A COMBINED IMAGE APPROACH TO COMPRESSION OF VOLUMETRIC DATA USING DELAUNAY TETRAHEDRALIZATION

R. Jovanovic*†*, R.A. Lorentz †

** Institute of Physics, University of Belgrade, Pregrevica 118, Zemun, Serbia, rakabog@yahoo.com*
† Texas A&M University at Qatar, Doha, PO Box 23874, Qatar, rudolph.lorentz@qatar.tamu.edu

Keywords: Compression, Delaunay tetrahedralization.

Abstract

We present a method for lossy compression of three dimensional gray scale images that is based on a 3D linear spline approximation to the image. We have extended an approach that has previously been successfully applied in two dimensions. In our method, we first select significant points in the data, and use them to create a 3D tetrahedralization. The tetrahedrons of the tetrahedralization are used as cells for a linear interpolation spline that gives an approximation of the original image. The compression is done by storing the positions of the vertices of the tetrahedralization and the values there instead of the value of the approximation at each grid point. We introduce a novel concept of using a smoothed version of the original image to improve the quality of the approximating spline. To increase the efficiency of the algorithm, we combine it with a refinement/decimation technique. We compare our compression technique to JPG2000 3D. We show that our algorithm performs similarly to, and in some cases even outperforms it, for high compression ratios. Our approach gives images that have significantly different properties than ones created using wavelets, and have the potential of being more suitable for some applications. In addition, this type of compression is particularly suitable for visualization.

1 Introduction

Volumetric data sets are being created and used much more frequently in the recent years since the appearance of more powerful computers and the fall in prices of equipment that can gather and digitalize data of this type. With the increased resolution of the images the volume of data has become tremendous. For example, to store a three dimensional data set whose dimension is equal to 1024 in each direction, and whose voxels are bytes, we need one gigabyte. This shows us the necessity of compression for volumetric data.

In this article we shall present a lossy 3D compression method for volumetric images. New compression algorithms for 3D data have been developed by adapting existing 2D methods. Previously, the Discrete Cosine Transform (DCT) has been extended to be used for the compression of volumetric data [2,22]. Currently, the most effective algorithms for compression of volumetric data use wavelets in three dimensions [4,5,18]. Wavelet compression of volumetric data has been standardized and is in part 10 of JPG2000 [20].

A completely different approach to the compression of 2D images is to approximate them using piecewise linear splines based on a triangulation. The compressed image file then contains only the vertices of the triangles and the values at the vertices [7,8,12,16]. The decompression consists of decompressing the vertices and their corresponding values, then carrying out a linear interpolation of the values to the pixel locations. This technique first determines the significant points of the image in the sense that they represent the geometry. We define a significant point to be a pair: the position and the corresponding value. Later these points are used to create a triangulation, and a linear spline which is an approximation of the image. In the cases where very high compression ratios are needed, the use of this type of method gave results that are equivalent to or even better than the standard JPG2000 [13]. This method has been extend to the third dimension for application on video data [9].

In this paper, we present an extension of this compression method using data dependent triangulations to the third dimension. The method is developed for gray scale 3D images. It is fully defined by giving a method for vertex selection, creation of linear splines and a coding system for storing the selected significant points.

To accelerate convergence, we introduce a novel concept of using a smoothed version of the original image to improve the quality of the approximation. This approach to compression has the additional advantage that visualization is accelerated. To visualize a compressed image, it is not necessary to decompress the entire image, since many visualization routines require only the vertices of the tetrahedrons and the values there, not all of the voxel values.

In 3D, instead of using triangles for creating linear splines, we use tetrahedrons. We have extended Silva's refinement algorithm [7] previously used on mammographic medical images to the third dimension. We further improve the quality of the approximation spline by adding a correction method for the values corresponding to significant points by taking into account a smoothed version of the original image. To increase its efficiency, we have combined it with the

refinement/decimation technique as proposed by H. Pedrini [15]. To make this possible, due to the great increase of calculation time when the problem is solved in the third dimension, we have developed a new approximate method for selecting those points which should be removed. We show that this approach applied to volumetric data compression gives results that are comparable and in some cases even better that the 3D version of JPG 2000 in terms of peak signal to noise ratio for high compression ratios. The compressed images have significantly different properties than ones created by wavelets which can make them more suitable for some applications.

The paper is organized in the following way. In the second section we give an introduction to creation of three dimensional data dependent tetrahedralization. In the next section we give details of the combined image approach that improves the quality of tetrahedralization. In the fourth section we explain the adaptation of the refinement/decimation technique for selection of significant points. In the next section, we give a short overview of the complete algorithm. In the fifth section we compare our method to Silva'the three dimensional version of JPG2000 and, finally, in the last section, we draw some conclusions.

2 Three dimensional data dependent Tetrahedralization

Two dimensional images in computer graphics are most commonly defined as discrete scalar fields $I:\Omega \to C$ with a domain $\Omega = \{0,...,i\} \times \{0,...,j\}$, as a regular grid of width i and the height j into the color space C. In the case of three dimensional data, $\Omega = \{0,..,i\} \times \{0,..,j\} \times \{0,..,k\}$ where k is the depth of the regular grid. In this paper we focus on gray scale images where C is $\{0,..,2^l -1\}$. We use the term significant point for a pair consisting of position and value, which is *(x,y,value)* in two dimensions and *(x,y,z,value)* in three dimensions. A tetrahedralization T of the domain Ω can be used for creating an approximation to the image I by linear interpolation of the function values from the vertices to the interior of each of the tetrahedrons.

In this section, we give a detailed explanation of the extension of Silva's algorithm for selecting significant points of two dimensional images for creating linear approximation splines [7] to the third dimension. In Silva's algorithm, the basic idea is to find a triangulation T of the domain with vertices chosen from among the grid points such that the linear spline based on T gives a good approximation to the data. It is only necessary to store the significant points (in a compressed form), because from this information, we can reconstruct the triangulation, and thus the linear spline using Delaunay triangulation.

To do this we shall first explain what a DT is in three dimensions. Next we show the changes that need to be made on the iterative algorithm for finding the significant points. First we point out that a direct extension of the two dimensional Delaunay triangulation can be done to the third dimension. In it, instead of using triangles we use tetrahedrons as basic cells. One important property of DT is that it is unique for a set of points in which no four lie on one circle. The Delaunay tetrahedralization now has the specific empty sphere property, that is, the circumscribing sphere of each cell of such a tetrahedralization does not contain any other vertex of the tetrahedralization in its interior. These tetrahedralizations are uniquely defined except in degenerate cases in which five points are cospherical. Even in this case, with some restrictions, it can be made unique [10].

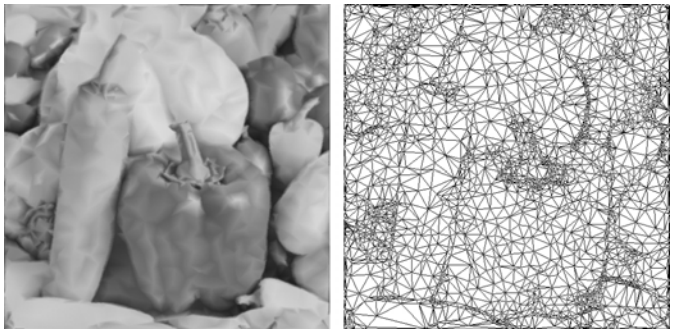

Figure 1. An example of triangulation and appropriate approximation

Silva's algorithms start with a simple triangulation containing a very small number of vertices to which we iteratively add new vertices to improve the quality of the approximation. This approach is called refinement. Opposite to this is decimation, where we start with a very fine triangulation containing vertices corresponding to all the pixels of the image. Then at each step we remove the vertices that decreases the quality of the approximation the least. Refinement techniques in general achieve results of lower quality but are significantly faster compared to decimation. Note that the vertices here are always grid points.

In Silva's algorithm, initially the four corner points of the image are chosen as significant points. These vertices create a triangulation consisting of the top left and bottom right triangle. Using these two triangles, an approximation spline L_0 is created. At each step i of the algorithm, a piecewise linear spline L_i is obtained. At odd refinement steps, a new point with the largest approximation error is selected and with it a new triangulation and corresponding linear spline L_{i+1} are created. The error for triangle t is the sum of errors of all points p inside of it. In the case of an even iteration, the point with the worst approximation is selected from that triangle which has the largest error. This process is repeated until the desired approximation quality is achieved or the maximum number of vertices is reached. An example is given in Figure 1. In Silva's algorithm, the values of the spline at its vertices is always taken to be the value of the image there. In our algorithm, we remove this restriction.

Silva's algorithm for selecting the significant points in three dimensions is changed in the following way. Initially we start with a tetrahedralization using the 8 corner vertices of the domain and tetrahedrons that a DT containing them generates. We use the refinement approach proposed by [7]. More precisely, we shall be switching between the globally worst approximated point and instead of choosing the triangle with

the biggest error, we choose the appropriate tetrahedron. To do this we need to define an error function for a tetrahedron Δ at iteration *i* as follows

$$E_i(\Delta) = \sum_{p \in \Delta \cap \Omega} |I(p) - L_i(p)|^2 \qquad (1)$$

where *I* is the image.

The error function in Equation 1 represents the sum of squares of errors over all the points inside the tetrahedron. We use the l^2 error instead of the absolute value as in the case of two dimensions, because we are trying to minimize the peak signal to noise ratio. We have observed in our tests, that better results are achieved in this way.

3 Combined image approach

In this section we present our combined image approach to the selection of significant points of the image. We alternate between the use of the original image and a smoothed version. The choice of which image will be used at any step of the algorithm, depends on the quality of the approximation achieved by the linear spline. Our approach, also separates the selection of position and values for a significant point

We first tried using a smoothed image obtained from applying a Gaussian blur filter [19]. The desired effect of removing problematic areas from the image is achieved but at the cost of some regions of the image becoming greatly distorted. Therefore instead of using Gaussian blur, we use the Bilateral filter which is given in article [21]. This filter, as does the Gaussian blur, computes a weighted average of pixel values in the neighbourhood, but instead of just using the distance between pixels it also takes into account corresponding values. This way, edges are just slightly blurred while noise reduction is achieved.

Let us define I^* as the image resulting from applying the Bilateral filter on the original image *I*. The initial idea was to use solely I^* instead of *I* in the algorithm described in the previous section. The problem with this approach is that the smoothed image does not approximate the original image well. One solution to this problem is to stop using the smoothed image when the approximation has reached a certain level of PSNR. We avoid this problem by combining the use of the smoothed image and the original one. This is done by dividing the selection of a significant point into two stages. In the first, we select the significant points position and in the second the corresponding value. In each of these stages we use one of the two images. We give an outline of our method in the following pseudo code:

```
Add corners of volume to DT
I* = BilateralFilter(I)
I_c = I*
WHILE(More Points Needed)
        IF (counter.\ mod.\ 2 == 0)
            p = MaxErrorPosition(L,I_c)
        ELSE
            p = MaxTetrahedronErrorPosition(L,I_c)
        Add p to DT
        IF (|I - L_I*(p) | < |I – L_I(p) |)
            L = L_I*(p)
        ELSE
            L = L_I(p)
        IF (PSNR(L) < α PSNR(I*))
            I_c=I*
        ELSE
            I_c=I
ENDWHILE
```

In the pseudo code, I_c stands for the image which is used in the current calculations. *p* is a point with position *(x; y; z)*. *L* is the linear spline corresponding to the current tetrahedralization. The function *MaxErrorPosition* gives us the position of that point for which the approximation spline *L* has biggest error to I_c. Similarly, *MaxTetrahedronErrorPosition* gives us the maximum error position inside the tetrahedron *D* with the greatest error *E(D)*. *α* is a fixed constant that regulates when we use the original or smoothed image for the selection of a significant point.
The algorithm starts by creating a Tetrahedralization from the 8 corner points of the volume. We set I_c to be the smoothed image *I** which we use in Silva's algorithm. As proposed by Silva, we switch selecting between *MaxErrorPosition* and *MaxTetrahedronErrorPosition* for selecting points depending on whether we are adding an odd or even vertex. When the selected position is added to the triangulation it generates a new *DT*. For the new position *p* and *DT* we check which of the values *I(p)* from the original or *I*(p)* from the smoothed image gives an interpolation spline $L_{I(p)}$ or $L_{I^*(p)}$ with higher quality approximation to the original image. We finally add a significant point *p* with a value that is equal to *I(p)* or *I*(p)*, depending on which gives a better approximation to our tetrahedralization. Before selecting the next point we check if the approximation function has reached a certain level of PSNR (αPSNR(I*)). If it has, we start using the original image *I* as the current image. We wish to point out that the two approximation splines $L_{I^*(p)}$ and $L_{I(p)}$ which appear in the algorithm are easily calculated together and the calculation of the second one does not greatly increase execution time. .

4 Refinement-Decimation

The results achieved by the above algorithm can be greatly improved by adding decimation/refinement steps as proposed by H. Pedrini [15]. The idea of this approach comes from the fact that some points, at the moment they are added to the set of significant points, greatly improve the approximation spline but as further ones are added they may become unnecessary in the sense that their removal would not greatly change the quality of the approximation function. One extreme example of generating unnecessary points using a refinement method is shown by Garland and Heckbert [11]. So instead of just adding new points to the set at some stages of the algorithm, we shall remove the least necessary ones.
There are several ways of selecting points for decimation. In the case of 2D decimation as given in [8], an exact approach

has been used. The vertex whose removal shall minimally decrease the PSNR of the approximation is chosen. To do this, we need to recalculate the triangulation and the approximation function in the changed triangles. In the case of two dimensions, the recalculation of the approximation function is not a great part of the overall calculation time of the algorithm. However, when we move to the third dimension and use tetrahedrons instead of triangles, this part of the algorithm becomes substantially more time consuming, a fact which we have experienced in our experiments. A different approach is given in [15] for the 2D version of the problem. Instead of calculating the exact effect of removing a point, an approximation to it is used. It consists of removing the point whose removal will least change the approximation.

In our algorithm, we implement a mix of exact and approximate methods for selecting which vertex shall be removed. It is obvious that at the time a new vertex is added we can easily see what will be the effect of its removal on the quality of the approximation. The error corresponding to vertex p can be seen as the sum of errors of all the tetrahedrons that are incident to it. The other tetrahedrons are not influenced by p. In Equation 2, we give the error connected with vertex p that is added as the n+1-st vertex of the tetrahedralization.

$$E_{n+1}(p) = \sum_{\Delta \in \Gamma_{n+1}} E_{n+1}(\Delta) \tag{2}$$

Here E_{n+1} represents the error functions, Γ_{n+1} represents the set of tetrahedrons that vertex p is incident to when $n+1$ significant points have been selected. At this stage it is necessary to calculate all $E_{n+1}(\Delta)$ for all tetrahedrons inside Γ_{n+1} to prepare the next step of the refinement algorithm. We shall define the error that corresponds to p at step n in Equation 3

$$E_n^{p^-}(p) = \sum_{\Delta \in \Gamma_n^{p^-}} E_n(\Delta) \tag{3}$$

In Equation 3, $\Gamma^{p^-}_n$ is the set of all the tetrahedrons that are in conflict with, or in other words their circumscribing sphere contains, p in the DT without p. This DT is exactly the tetrahedralization at step n. All the values of $E^{p^-}_n(D)$ are known before adding p so $E^{p^-}_n(p)$ is easily calculated. From the definition of the Delaunay tetrahedralization we know that volume $\Gamma^{p^-}_n$ is equivalent to Γ_{n+1}. So the effect of removing p from the tetrahedralization is the following.

$$ErrorChange(p, n+1) = E_{n+1}(p) - E_n^{p^-}(p) \tag{4}$$

ErrorChange gives us the the effect of removing p at step $n+1$. We wish to emphasize that the *ErrorChange* does not use the absolute value since removal of a point can in some cases even improve the approximation.

The problem is that when adding the point p, its neighbors are also affected due to the re-tetrahedralization that occurs. The current error for a neighboring point q is easily calculated using Equation 2, since all the E_{n+1} are known. To get the exact value of $E^{q^-}_n(q)$ we need to recalculate the tetrahedralization that exists after removing q and the corresponding approximation. This recalculation is as complex as the one for adding p and it needs to be done for all the neighbors. This makes it very inefficient. We avoid this by using an approximation of the error at previous step $E^{q^-}_n(q)$. This is done in the following way. Let us say q has been added at step $m+1$. At this moment it was easy to calculate the error $E^{q^-}_m(q)$. This value is stored. Let us define

$$\mu_{n+1}(p) = \sum_{\Delta \in \Gamma_{n+1}} \mu(\Delta) \tag{5}$$

In Equation 5, $\mu(\Delta)$ represents the volume of the tetrahedron Δ. $\mu_{n+1}(p)$ is the volume that is connected to a vertex p at step $n+1$ which is equal to the sum of volumes of all the tetrahedrons that are connected to it. Using μ we define the following approximation for $\tilde{E}^{q^-}_n(q)$

$$\tilde{E}_n^{q^-}(q) = E_m^{q^-}(q)\frac{\mu_{n+1}(q)}{\mu_m(q)} \tag{6}$$

$\tilde{E}^{q^-}_n(q)$ is equal to the error $E^{q^-}_m(q)$ just scaled to the new volume to which q is connected at step $n+1$. Now we shall remove vertices that have the minimal approximated error change, given in Equation 7, from the tetrahedralization.

$$ErrorChange^*(p, n+1) = E_{n+1}(p) - \tilde{E}_n^{p^-}(p) \tag{7}$$

5 Algorithm overview

The complete algorithm proceeds as follows. We first initialize the DT by adding the corner vertices of the brick that represents the data. We set the number of vertices that shall be decimated to half of the maximal size of our significant point set. In the main loop, we add vertices to the DT using the combined image approach until we reach the maximal allowed number of vertices. When we reach this number, we remove *NumOfDecimation* vertices from the significant point set. We set *NumOf Decimation* to its half and repeat this process until no vertex should be decimated. Finally the significant points, which are effectively a point cloud, were compressed using coding proposed in article [14].

The software is written in C++. In it, the calculation of the three dimensional DT is done using CGAL (Computational Geometry Algorithms Library) which is an open source project that provides efficient and reliable geometric algorithms in the form of a C++ library [1].We have conducted our tests on data sets that are a part of The Volume Library assembled by Stefan Roettger [17]. This library is a collection of CT Scans, MRI Scans, Laser scanning microscopy and computer generated data sets which have been taken from several industry and academic sources. We compare our algorithm to JPG2000 3D (Part 10 - JP3d) which is an extension of JPG 2000 for three dimensional data. In our tests, we have compared the achieved PSNR for a fixed number of bits per voxel for JPG 2000 3D and our method. We also analyze the effect of our improvements on the direct extension of Silva's algorithm [7].

6 Results and Discussion

In this section, we evaluate the effectiveness of using the previously given tetrahedralization algorithm for compression of volumetric data.

Our results are contained in Table 1 which compares the PSNR obtained when compressing to 0.03125 bits/voxel for Silva's method, for our method without using a smoothed image, for our method using a smoothed image and for JPEG2000/3D.

Previous research, done for two dimensional images, has shown that compression methods that use triangulation for creating approximating linear splines are most effective when very high compression ratios are desired [12]. Because of this, in our experiments, we choose 0.03125 bits per voxel. This bit rate was the minimal one used in [18]. We can see the results on various test images in Table 1.

File	Delaunay			JPG2000
	Silva	Dec	Dec+Com	3D
Dti-fa	30.00	30.45	**31.92**	31.59
Dti-md	36.70	37.21	**37.84**	34.51
Spheres	29.81	30.35	**31.88**	31.48
BluntFin	44.54	45.75	45.57	**46.96**
Daisy	34.72	35.12	**37.84**	37.51
Orange	29.60	30.21	31.14	**33.31**
Tomato	33.44	34.25	35.43	**37.31**
Baby	30.81	31.19	31.42	**32.86**
Vismale	32.44	33.03	33.10	**34.06**
Engine	38.35	38.79	39.17	**41.39**
AVERAGE	34.05	34.64	35.53	**36.09**

Table 1. Comparison of Peak Signal to Noise Ratio archived by JPG2000/3D and Delaunay tetrahedralization methods on different data sets using 0.03125 bits per voxel

The decimation improvement has shown its effectiveness in all the tested cases and shows an increase of PSNR compared to the direct extension of Silva's method in average from 34.05 to 34.64. The combination of decimation and the combined image approach would further increase the value of PSNR and in average it would be 35.53. Only in the case of BluntFin, did the use of a smoothed image not improve the PSNR when it was combined with decimation. The reason for this is that the original image is very smooth which means there are no problematic regions and further smoothing only distorts the image. This type of case can easily be recognized and avoided since the PSNR of the smoothed image is very high. Contrary to this in images that have a high level of discontinuity, like spheres and Daisy, the positive effect of this improvement is more evident. Our method in average does not perform as well as the JPG 2000 3D when PSNR is observed. But in some cases, it is more effective. In the case of 0.03125 bits used per voxel the difference in PSNR is relatively small 35.53 compared to 36.09. We have observed that the source of the error is significantly different for our method and JPG 2000. In our observation of we have noticed that in the case of JPG 2000 3D the error is small over the whole image, while with our method the error greatly varies from zero in some areas while being very high in others.

In the case of smaller sized images (Dti-fa, Dti-md, Spheres) when both improvements have been applied, the PSNR that is achieved even outperforms JPG 2000 3D. This can be explained by the fact that the number of bits needed to store a significant point depends on the image size. Because of this, a higher proportion of number of significant points to image size can be achieved for smaller images for the same number of bits per voxel, while in the case of wavelets no similar relation exists.

In many of the uses of volumetric data, it is visualized. In some cases this would be interactive. For this type of application, our method can prove to be more efficient than wavelet based methods. When visualizing compressed volumetric data in 3D, we first need to uncompress all the values in the grid and later apply one of the standard rendering methods. These methods are usually very slow due to the fact that a large amount of data needs to be processed. As for using triangles when rendering surfaces, it is possible to optimize the rendering of volumes using tetrahedrons. An overview of advantages of using tetrahedrons for different rendering techniques can be seen in the survey article [6]. This type of rendering is suitable for our compression method since it is based on tetrahedrons and secondly that full decompression is not needed but only the decompression of the vertices and the values.

7 Conclusion

In this paper we have presented a promising new method for compressing volumetric data that is based on the use of Delaunay tetrahedralization combined with linear splines. It is an extension of a known two dimensional refinement technique to the third dimension. We have given a detailed explanation of the changes that appear compared to the two dimensional problem. We improved the quality of the approximation spline by combining the use of original data with a smoothed version of it. This concept can also be applied to 2D images. Further, we have introduced a method that greatly decreases the calculation time needed for selection of significant points that should be removed when decimation steps are added to the basic algorithm.

We have compared our compression method to JPG 2000 3D on a variety of different data sets. In cases where very high compression ratios are desirable, our method has given results of similar, in some cases even better, quality when PSNR is observed. The artifacts that appear in the compressed image have different properties than the ones that appear when using JPG 2000 3D. Because of this difference, it could be more suitable for certain types of data. An additional advantage of our method is that the compressed data is actually a tetrahedralization of the original image which makes it more suitable for visualization, especially since full decompression is no longer needed. We believe that the results achieved, can be further improved by adapting some of the more complex methods that have been developed for two dimensional images

References

[1] *Cgal, Computational Geometry Algorithms Library*. 2010; Available from: http://www.cgal.org.

[2] G.P. Abousleman, M.W. Marcellin, and B.R. Hunt, *Compression of hyperspectral imagery using the 3-D DCT and hybrid DPCM/DCT.* Geoscience and Remote Sensing, IEEE Transactions on, 1995. **33**(1): p. 26 -34.

[3] S. Ait-Aoudia, F.-Z. Benhamida, and M.-A. Yousfi, *Lossless Compression of Volumetric Medical Data*, in *Computer and Information Sciences, ISCIS 2006*. 2006, Springer Berlin / Heidelberg. p. 563-571.

[4] H. Benoit-Cattin, et al., *3D medical image coding using separable 3D wavelet decomposition and lattice vector quantization.* Signal Processing, 1997. **59**(2): p. 139 - 153.

[5] A. Bilgin, G. Zweig, and M.W. Marcellin, *Three-Dimensional Image Compression With Integer Wavelet Transforms.* Appl. Opt., 2000. **39**(11): p. 1799-1814.

[6] P. Cignoni, C. Montani, and R. Scopigno, *Tetrahedra Based Volume Visualization*, in *Mathematical Visualization, Algorithms, Applications, and Numerics*. 1998, Springer Verlag. p. 3-18.

[7] L.S. da Silva and J. Scharcanski, *A lossless compression approach for mammographic digital images based on the Delaunay triangulation*, in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. 2005. p. II - 758-61.

[8] L. Demaret, N. Dyn, and A. Iske, *Image compression by linear splines over adaptive triangulations.* Signal Processing, 2006. **86**(7): p. 1604 - 1616.

[9] L. Demaret, A. Iske, and W. Khachabi, *Sparse Representation of Video Data by Adaptive Tetrahedralizations*, in *Locally adaptive filters in image and signal processing*. 2010. p. 197-220.

[10] T.-P. Fang and L.A. Piegl, *Delaunay triangulation in three dimensions*. Computer Graphics and Applications, IEEE, 1995. **15**(5): p. 62 -69.

[11] M. Garland and P. Heckbert, *Fast Polygonal Approximation of Terrains and Height Fields*. 1995, Computer Science Department, Carnegie Mellon University.

[12] B. Lehner, G. Umlauf, and B. Hamann, *Image Compression Using Data-Dependent Triangulations*, in *Advances in Visual Computing*. 2007, Springer Berlin / Heidelberg. p. 351-362.

[13] B. Lehner, G. Umlauf, and B. Hamann, *Survey of techniques for data-dependent triangulations*, in *GI Lecture Notes in Informatics, Visualization of Large and Unstructured Data Sets*. 2007. p. 178-187.

[14] T. Lewiner, et al., *GEncode: Geometry-Driven Compression in Arbitrary Dimension and Co-Dimension*, in *SIBGRAPI*. 2005. p. 249 - 256.

[15] H. Pedrini, *An Improved Refinement and Decimation Method for Adaptive Terrain Surface Approximation*, in *WSCG*. 2001. p. 103-109.

[16] V. Petrovic and F. Kuester, *Optimized Construction of Linear Approximations to Image Data.* Computer Graphics and Applications, Pacific Conference on, 2003: p. 487.

[17] S. Roettger. *The Volume Library*. 2006; Available from: http://www9.informatik.uni-erlangen.de/External/vollib/.

[18] P. Schelkens, et al., *Wavelet coding of volumetric medical datasets.* Medical Imaging, IEEE Transactions on, 2003. **22**(3): p. 441 -458.

[19] L.G. Shapiro and G.C. Stockman, *Computer Vision*. 2001: Prentence Hall. 137-150.

[20] D.S. Taubman, M.W. Marcellin, and M. Rabbani, *JPEG2000: Image Compression Fundamentals, Standards and Practice.* Journal of Electronic Imaging, 2002. **11**(2): p. 286-287.

[21] C. Tomasi and R. Manduchi, *Bilateral Filtering for Gray and Color Images*, in *ICCV-98*. 1998, IEEE Computer Society: Washington, DC, USA. p. 839.

[22] A.M. Vlaicu, et al., *New compression techniques for storage and transmission of 2D and 3D medical images*, in *Advanced Image and Video Communications and Storage Technologies*, N. Ohta, H.U. Lemke, and J.C. Lehureau, Editors. 1995, SPIE: Amsterdam, Netherlands. p. 370-377.