3-D Haar Wavelet Transformation in Java

Pujita Pinnamaneni
Dept. of Electrical and Computer Engineering, Mississippi State University
Joerg Meyer
Dept. of Computer Science, Mississippi State University
Email: {pp2|jmeyer}@erc.msstate.edu

Abstract

Internet has connected different parts of the world through a network link. Researchers in bio-informatics and biology have always wanted to share information for the progress of their research. But the data, they wanted to transfer are large-scale data sets which cannot be transferred over the existing network link in a considerable amount of time. Hence, methods to compress these large volumes of data are found. Wavelets are excellent data compression tools [1]. This paper discusses the Haar wavelet transformation to transform large-scale biological data sets. As we are more inclined towards viewing our data in 3-D we have implemented the Haar wavelet transform in 3-D.

1. Introduction

A Wavelet transformation converts data from the spatial into the frequency domain and then stores each component with a corresponding matching resolution scale [1]. Wavelets are used to represent different levels of detail. The Haar wavelet is one of the simplest wavelet transforms. This paper deals with the 3-D transformation of large data sets using a Haar wavelet transform. Image compression is important for reducing data storage cost and transmission costs in slow network channels. Biologists and researchers all over the world have wanted to access large data sets, which occupy gigabytes to terabytes of memory space. Storing such large data sets onto one's local hard drive is usually very costly. Hence efforts have been made to establish a central repository where these data sets are stored. When researchers want to access these data, these data sets are transmitted over the network. But sending such large data sets over the network in a reasonable amount of time considering the speed of the network channel can also be a huge challenge. This is where compression techniques come into act. Various compression

techniques have been studied before, but wavelets have shown to be more efficient than many other methods. Using the Haar wavelet transform, these huge data sets are transformed to considerably smaller representations, which are then transmitted over the network at higher speeds. The data sets we are dealing with are stored in a volumetric data file format, which supports 3-D meshes. There are three different formats, which we call *vols*, *volb* and *volc* respectively. The *vols* file format stores data as 8 bit scalar values. The *volb* file format stores data as 24 bit RGB-alpha values. The color component values are single byte each with either '0' alpha value or a single byte alpha value. The *volc* file format stores data as 64 bit RGB-alpha-beta values. The color component values are 10 bits for red, 12 bits for green and 10 bits for blue. The alpha and the beta color components are truncated to a short each. The data set used for testing our algorithm is a CT scan of a human brain (*ctbrain.vols*). This data set consists of a series of 231 slices where each slice has been stored having 512 X 512 elements. The size of this file is 60,555,264 bytes (~ 57.8 GB). In the 3-D transformation technique proposed we have implemented a method to transform the data sets by storing them in a 3-D array and applying the Haar wavelet transformation algorithm in *x*-, *y*- and *z*- directions. The implementation of the Haar wavelet will be discussed in the following section. The transformation and the reconstruction part of the algorithm are illustrated with images.

2. Background

Image compression has been under a wide discussion over the past few years. Researchers have been talking about various methods of image compression schemes, e.g., JPEG, Synthetic Aperture Radar (SAR) [9], Multiwavelet image compression techniques [7], and Multi-resolution Trees [8], each having their own representation and optimization procedures. In most cases, compression has only been discussed in two dimensions. In the past few years, volume rendering of images has become more and more important. Hence implementation of a compression technique that could enable volume rendering (visualizing image in 3-D space) has inspired us to use the Haar wavelet transformation technique for efficient transformation and, more importantly, faster reconstruction of 3-D image data. Wavelets are local in both frequency and time domain. This helps to preserve features and represent functions with discontinuities and

sharp spikes in a more compact way and still achieve a reasonable approximation[1]. The detail coefficients usually have very small values. Sometimes they can be neglected or discretized in order to achieve better compression rates (RLE, LZW). These features make wavelets an excellent data compression tool. Compression techniques using wavelets have also proved advantageous when evaluating the image quality of the compressed images. Wavelets have proved to be faster in terms of their computational complexity as compared with other transforms such as the Fast Fourier Transform (FFT). We are making use of these features of wavelets and design an algorithm for a 3-D transformation technique, which will be discussed in the following section.

3. Haar Wavelet Transformation algorithm

The Haar wavelet transform is one of the simplest and basic transformations from the space domain to a local frequency domain. Haar wavelets are being used for the image transformation technique proposed here. To get a better idea about the implementation of this wavelet in image compression, we try to illustrate the procedure with a simple example.

Assume we have a one-dimensional image with an eight-pixel resolution, where the pixels have the following values[3]:

7 5 3 9 3 7 5 3

By applying the Haar wavelet transform we can represent this image in terms of a low-resolution image and a set of detail coefficients. The transformed data coefficients are obtained by averaging two consecutive pixels, while the detail coefficients represent the difference between the average and one of the two consecutive pixels. So the above image will be represented as follows after the first cycle:

Transformed coefficients: 6 6 5 4 Detail coefficients: 1 –3 –2 1

Now the original image can be represented by a four pixel transformed image ((a+b)/2) after the first cycle and a four pixel image containing the detail coefficients ((a-b)/2). Recursively iterating this algorithm leads to an image that is reduced by a factor of two for each cycle. The detail coefficients are required to reconstruct the image.

Reconstruction of the original image involves adding and subtracting the detail coefficients to and from the subsequent transformation coefficients for each cycle. During reconstruction of the original image of the previous example, adding and subtracting the transformation and detail coefficients obtain the first pair of pixels.

For interactive rendering, the low-resolution image is transmitted first. The detail coefficients are transmitted at a later time to obtain the next higher level of detail or to complete the reconstruction of the image.

In 2D wavelet transformation, structures are defined in 2-D and the transformation algorithm is applied in *x*-direction first, and then in the *y*-direction. Details about data structures and our implementation are discussed in the following section. Similarly, in 3-D wavelet transformation the structures are defined in 3-D and the transformation algorithm is applied in *x*-, *y*- and *z*- direction successively.

4. Implementation

Haar wavelet transformation is one of the simplest wavelet transforms and its implementation for compressing an image is illustrated in this section.

As discussed earlier, very large-scale data sets are considered for this transformation technique. The data set that has been used to test our prototype implementation is a CT scan of a human brain that has been scanned in 231 slices. Each individual slice has a resolution of 512 x 512 pixels. The structures used in this implementation are 3-D arrays. The data set is stored in a way that each slice is represented as a 2D array, where all slices are stacked along the third dimension for all the 231 cross-sections of the data set. The array sizes are expressed in powers of two. Mathematically, the original resolution of the images is converted into the next larger power of two, and the array sizes are initialized accordingly. For the above data set, the

3-D array defined is 512 x 512 x 256. While loading the slices into the 3-D array, care is taken to fill in the extra space allotted by zeros.

For the first cycle, the transformation algorithm is first run along the *x*-direction for all the 231 slices.

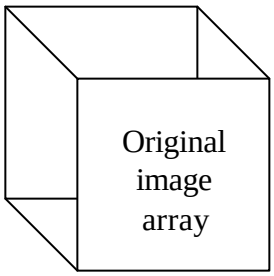


Figure 1: Original volume

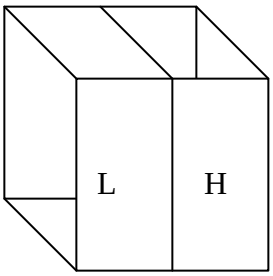


Figure 2: First run: *x*-direction

As shown in the image above (Figure 2), the image array is split into two halves containing the transformed data and the detail coefficients. The transformed data coefficients are the results of the low-pass filter while the detail coefficients are the results of the high-pass filter. After transforming the data set in the *x*-direction, this 3-D array is then transformed along the *y*-direction. The resulting 3-D array is as shown below.

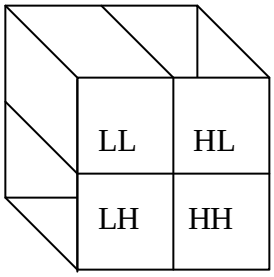


Figure 3: Second run: *y*-direction

Finally, the 3-D array is transformed along the *z*-direction and the resulting 3-D array is represented as shown below.

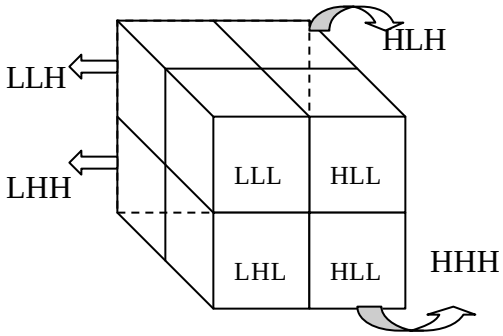


Figure 4: Third run: *z*-direction

One cycle of our algorithm is defined as the completion of three runs, one for each spatial direction (see Figure 4). The final array structure gives us a transformed version of the 3-D volume (the LLL segment) and the detail coefficients, which are used for the reconstruction, in the other seven octants. The indexing of the array is defined such that the transformed coefficients are stored in the upper-left-front corner for each successive cycle. Each time a volume is transformed the size of the original volume is reduced to one-eighth of the original size. The detail coefficients in the other segments are stored because they are required for the reconstruction of the original image.

During consecutive cycles, only the upper left portion of the transformed volume is transformed leaving the detail coefficients intact. The data set is transformed to an extent where the transformed volume has one element on each side. The 3-D array can be visualized as shown below after the second cycle.

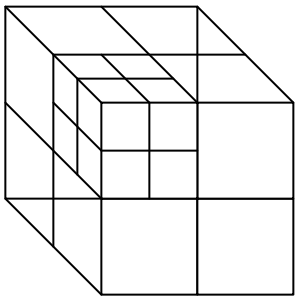


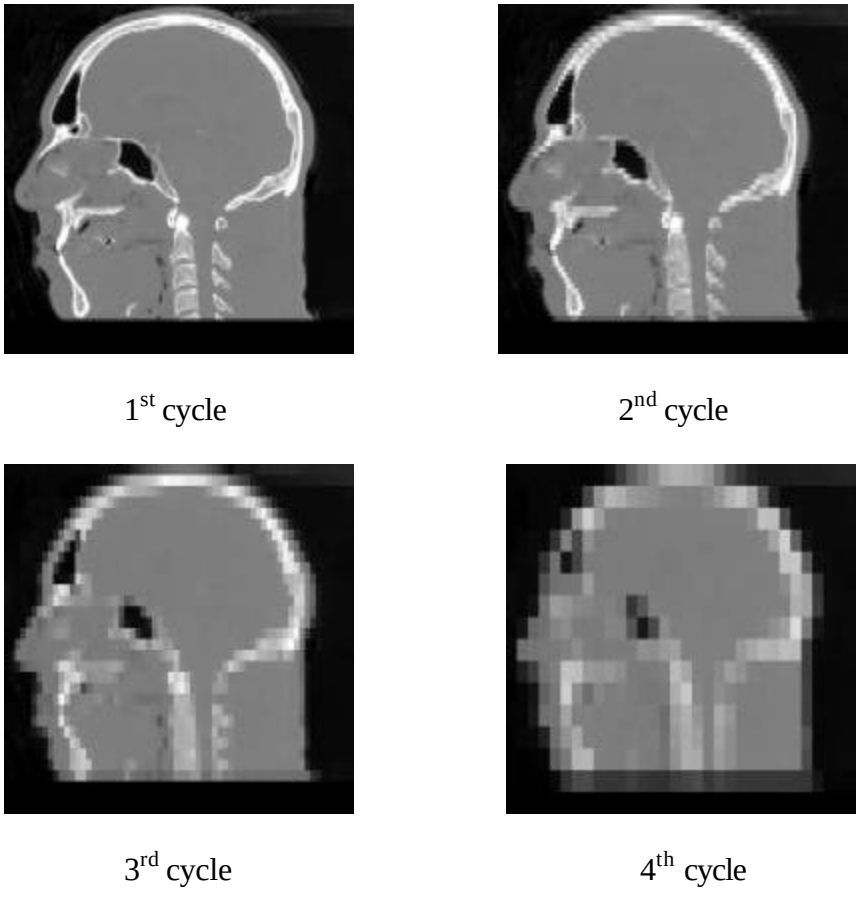
Figure 5: After two cycles

For the reconstruction, the detail coefficients resulting from each cycle are added and subtracted to the respective data coefficients to retrieve the original pixels values of the next higher level. A bottom to top approach is followed to get back the original image

from the transformed image. At each step of the reconstruction, the detail coefficients are added to and subtracted from the transformed data coefficients in order to retrieve the next pixel values. This stepwise retrieval of the pixel values for each cycle enables progressive reconstruction of the original image.

5. Results

We applied our algorithm to a CT scan of a human brain (512 x 512 x 231). The following images show a cross-section of the 3-D volume at three different levels of detail (Figure 6).



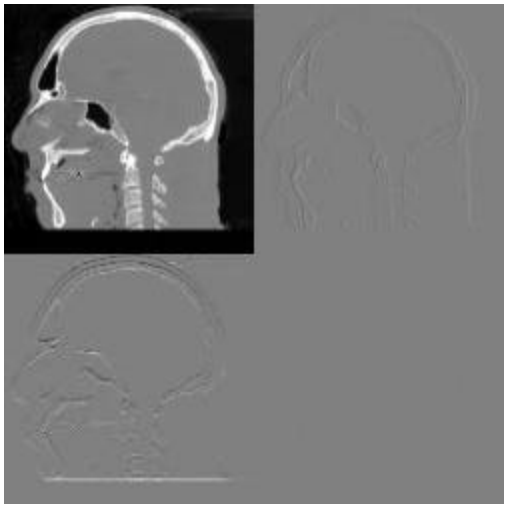
1^{st} cycle 2^{nd} cycle

3^{rd} cycle 4^{th} cycle

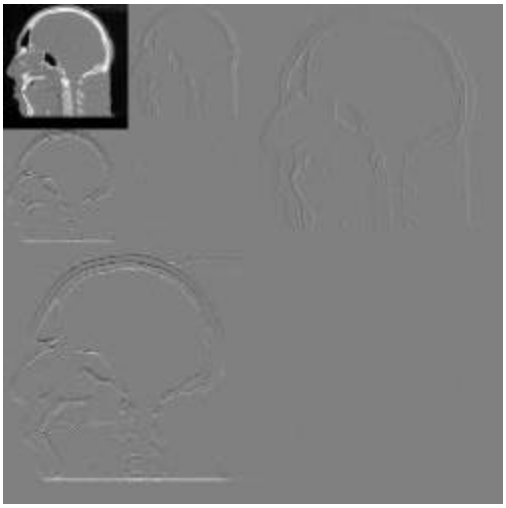
Figure 6: Compressed images in successive cycles

Successive cycle reduces the size of the original volume by a factor of eight (2^3). By applying a run-length encoding scheme, we were able to obtain much better compression rates for our 3-D algorithm than for a 2-D algorithm. This makes it possible

to transmit a low-pass filtered data set from a server to a rendering client over a low- bandwidth network, and then successively transmit more and more data to improve the resolution and quality of the 3-D reconstruction. The data set is stored on a server with large storage capacity, while the Java3-D-based rendering client runs on a desktop machine.

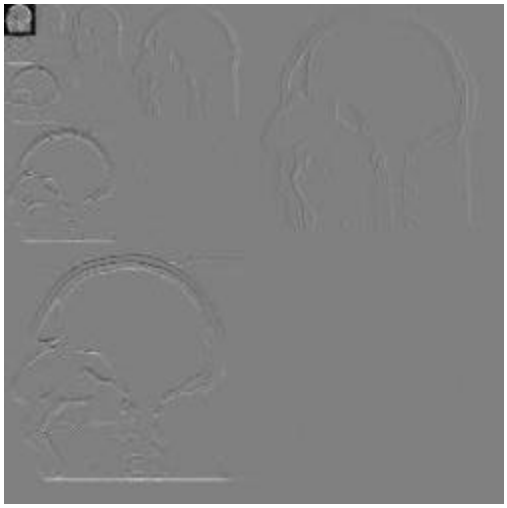


1st cycle



2nd cycle

3rd cycle



4th cycle

Figure 7: Compressed image data and detail coefficients after four cycles

6. Conclusions

Haar wavelet transformation has enabled the transformation of huge data sets in 3-D. The image quality of the transformed image has also been satisfactory. This paper has discussed the implementation of the transformation algorithm where the entire data set is transformed. Efforts are being taken to extract sub-volumes of the data set and transform the selected volumes in order to decrease the computational time of transforming unwanted data. Future work might involve transformation of certain data at a higher resolution as compared to the rest of the data to enable the user to have a better view a particular region of interest, while rest of the data is provided as context information.

Acknowledgements

We thank Arthur J. Olson (The Scripps Institute, La Jolla, CA) for providing the sample data sets. This project was funded by the National Partnership for Advanced Computational Infrastructure (NPACI) under award no. 10195430 00120410.

References

[1] Brani Vidakovic and Peter Müller, “Wavelets for kids,” A Tutorial Introduction, Institute of Statistics and Decision Science, Duke University, Durham, NC, 1991.

[2] Chao, Hongyang; Fisher, Paul, “An Approach to Fast Integer Reversible Wavelet Transforms for Image Compression,” Computer and Information Science Inc., 3401 E. University, Suite 104, Denton, TX 76208, in "Advances in Computational Mathematics: Guangzhou, China - The proceedings of Guangzhou International Symposium on Computational Mathematics," Guangzhou, P. R. China, August11-15, 1997.

[3] Eric J. Stollnitz, Tony D. DeRose and David H. Salesin, ”Wavelets for Computer Graphics : A Primer Part 1,” IEEE Computer Graphics and Applications, May 1995.

[4] Ghavamnia, Mohammad H., Yang Xue D., Direct Rendering of Laplacian Pyramid Compressed Volume Data, Proceedings Visualization '95, Atlanta, Georgia, October 29 - November 3, 1995.

[5] Ingrid Daubechies, Ten Lectures on Wavelets, Society of Industrial and Applied

Mathematics, Philadelphia, Pennsylvania, 1992.

[6] Knittel, Guenter, “High-Speed Volume Rendering Using Redundant Block Compression,” Proceedings Visualization '95, Atlanta, Georgia, October 29 - November 3, 1995.

[7] Michael B. Martin and Amy E. Bell, “New Image Compression Techniques Using Multiwavelets and Multiwavelets Packets,” IEEE Trans. Image Processing, Vol. 10, pp. 500-510, April 2001.

[8] Wilhelms, Jane, and Allen Van Gelder, “Multi-dimensional Trees for Controlled Volume Rendering and Compression,” 1994 ACM Symposium on Volume Visualization, pp. 27-34, October, 1994.

[9] Zhaohui Zeng and Ian G. Cumming, “SAR Image data Compression using Tree–Structured Wavelet Transform,” IEEE Trans. Geosciences and Remote sensing, Vol. 39, No. 3, pp. 546-552, March 2001.