Multiresolution Data Handling for Visualization of Very Large Data Sets

Norbert Strobel[1], Chrisian Gosch[2], Jürgen Hesser[3]
and Christoph Poliwoda[4]

[1]Siemens Medical Solutions, 91052 Erlangen
[2]Volume Graphics, 69123 Heidelberg
[3]Uni Mannheim, 68131 Mannheim
[4]Volume Graphics, 69123 Heidelberg
Email: Norbert.Strobel@siemens.com

Abstract. We discuss some features of an experimental system for visualization of large (medical) volume data sets. Input voxel data sets are subdivided into blocks first. Then each block is decomposed into a multiresolution data representation by applying a reversible 3D integer Haar wavelet transform (S-transform). The resulting transform coefficients are encoded using a Golomb-Rice algorithm. For volume visualization, we selectively load and decompress blocks to a proper resolution. Then they are rendered into a common view. From our experiments, we learned that large volume data sets should preferably be stored using a multiresolution data representation. Depending on the size of the volume data set and the rendering mode, we also found that a block-based data representation can provide some advantages, but it may not always be the best choice.

1 Introduction

In medical and industrial applications, data sets may take up Giga bytes of memory. For efficient data retrieval over bandlimited networks and for volume visualization on computers with a limited amount of main memory (RAM), data processing and file storage have to be designed properly.

In addition, some applications such as diagnostic imaging require very high-quality data sets. This may rule out *lossy* compression to keep storage space and bandwidth requirements low. However, this does not mean that there is no room for image compression. It only implies that there must be an option to ultimately retrieve an exact copy of the original image.

Wavelet transform techniques offer a very elegant solution to this problem. They provide a mathematical framework for multiresolution data representation, and the wavelet basis functions usually decorrelate image input data rather well.

Among many examples in the literature, Ihm and Park, partitioned a given 3-D volume data set into so-called *unit blocks* to achieve both high compression ratios and fast random access [1]. In related work, Bajaj et al. extended this approach to 3D RGB images and light fields [2]. Nguyen and Saupe again reported

interesting performance results with block-based wavelet compression techniques [3]. Guthe et al. initially divided the input data set into cubic blocks as well [4]. Then they applied integer wavelet filters to each block. After separating the low pass and the highpass coefficients, they, however, grouped eight adjacent low pass blocks to obtain another cubic block for further processing. This approach is performed recursively until only one (low pass) block is left.

In this work, we investigate a multiresolution volume compression approach involving an integer version of the 3-D Haar wavelet transform (S-transform) and Golomb-Rice coding. Our goal was to investigate the advantages and problems encountered when rendering large volume data sets using a block-based data representation.

The remaining part of the paper is structured as follows: In the next section, we discuss how to compute a multiresolution data representation using the 3D S-transform. Then we outline the multiresolution visualization method used. Finally, we discuss our findings and offer some conclusions.

2 Multiresolution Data Representation

Similar to Ihm, Bajaj, Nguyen and Guthe, we partitioned our input data sets into blocks first. These blocks were then 3D wavelet transformed using an integer version of the Haar wavelet transform called S-transform [6]. The resulting transform coefficients were fed into a fast Golomb-Rice coder next and finally stored to disk.

2.1 Data Transform and Coding

The one-dimensional S-transform is extended to 3D by applying it first along image columns, then along image rows, and finally across successive slices in a volume data set.

The 3D S-transform takes a cube of eight adjacent voxels at pyramid level l and computes an average coefficient at level $l+1$. In addition, it produces seven associated differences or detail coefficients at level $l+1$. Since the 3D S-transform just involves addition, subtraction, and shift operations, it can be implemented very efficiently.

Depending on the initial block size, the 3D S-transform may be iterated by successively taking the average coefficients at each level as input for the next step. This way, a multiresolution representation of each input block can be computed. It consists of one subband with approximation coefficients (at the lowest spatial resolution). This low-resolution data set can be rendered directly. In addition, there are seven subbands comprising detail (or wavelet) coefficients at each level. The detail coefficients are needed to recursively recompute the original voxels.

A Golomb-Rice coder was used for entropy coding of the detail subbands [7]. Unlike Weinberger et al., who relied on contexts to optimize the Golomb-Rice encoding parameter for each sample, we determined a single parameter for each

Table 1. Compression results for three selected medical volume data sets. Note that the times for decoding and inverse transform are given in msec/block.

Name	CR	Decoding Time	Transformation Time
CT Head	2.05	7.33	2.77
MRI Head	2.88	7.92	3.33
CT Thorax	2.58	7.00	3.13

subband of each block and stored it as side information. Although suboptimal with respect to bit rate, decoding can be accelerated this way.

To assess the compression performance of our block-based approach, we took three medical volume data sets (16 bits per voxel) and divided them into blocks of size $32 \times 32 \times 32$ voxels. Next, a three-level S-transform was applied to each block.

In Table 1, we list the compression ratios (CRs), and how long it took to decode and inverse transform each block to its initial, i.e. original, resolution. Since the timing results are on a per-block basis, they have to be multiplied by the total number of blocks to get the overall time needed to decode and inverse transform a complete data set.

Our experiments were run on a Pentium II computer with 333 MHz and 256 MB RAM. No special SIMD commands were used to speed up the run-time performance.

Since the two-byte voxel data sets were compressed without any loss, compression ratios between two and three could be expected. Table 1 also shows that decompressing blocks comprising $32 \times 32 \times 32$ voxels took up to 11.25 msec on a 333 MHz Pentium II computer. This is equivalent to a decompression speed of about 5.5 MB/sec (MB: Mega byte). For a higher clock speed of 2 GHz, we may expect a decompression speed of around 33 MB/sec. If an even better computer was used, e.g., running at 4 GHz, the decompression speed should again double to 66 MB/sec. Then it will take about slightly less than eleven seconds to completely rebuild the $512 \times 512 \times 1440$ Visible Man fresh CT data set taking up 720 Mbytes uncompressed.

2.2 File Format and Data Structure

Starting with a block-based multiresolution data representation, two file formats for data storage become apparent. They are either

1. resolution-oriented, or
2. block-based.

In the first case, the data stored is sorted by resolution. That is, the low-resolution approximation (low pass) coefficients are followed by encoded detail (or wavelet) coefficients. This storage order facilitates volume rendering at increasingly higher spatial resolutions.

In the second case, the data is simply stored block by block. Such a storage format supports efficient rendering of volume regions comprised of a small number of blocks at the highest resolution.

We performed an experiment where we either linearly or randomly fetched data from the hard drive. Our results indicated that the storage order should match the render mode to avoid significant disk access delays mainly due to seek times.

As a consequence, the resolution-oriented storage format should be chosen when data sets are to be rendered at multiple resolutions. On the other hand, the block-based storage format appears superior when small regions of the original volume are to be rendered primarily at the highest resolution.

3 Multiresolution Visualization

A system for volume navigation was developed. Only the data currently contained in the visible volume is loaded block by block, decompressed and rendered onto a common image plane. The block data remains in main memory until a user-defined, previously set memory limit is exceeded.

Due to the multiresolution data representation, we can straightforwardly provide a volume preview simply by loading and rendering blocks at their lowest resolution.

For interactive volume visualization, we applied a view-based multiresolution data manager to achieve optimal RAM utilization. For a particular view, this data manager

- loads associated (compressed) voxel blocks from file,
- decompresses them to a particular resolution, and
- puts them into a block store for future use.

A hash-table is used for book-keeping and removal of blocks which went out of scope.

The resolution for each block is chosen either to be the maximum resolution that the user has chosen or a lower resolution, depending on the distance of the diverging rays traversing the block. This way, aliasing effects can be avoided by adapting the bandwidth of each block's data to the local ray density. Another approach to avoiding aliasing is to adapt the local ray density like Jung did with *two-phase perspective ray casting* [5].

4 Discussion and Conclusions

We designed and implemented a system for volume visualization. It transforms (cubic blocks of) voxel raw data into a multiresolution data representation first, and it uses a Golomb-Rice coding mechanism. The encoded transform coefficients are stored using a file format designed for fast random access. During visualization, the system selectively fetches compressed blocks from disk and

decompresses them into proper resolutions for rendering. A voxel block data management system tries to keep disk access to a minimum.

From our experiments, we conclude that a multiresolution volume data representation offers various interesting features for volume rendering. For example, we can selectively load only the low pass part of the multiresolution volume data set. This saves valuable bandwidth, and the data received is already sufficient to render views at a low spatial resolution. In addition, the low pass part of the volume data may require significantly less main memory than the full data set. In other words, a multiresolution data representation provides an interesting trade-off mechanism between rendering quality and system resources such as memory and bandwidth. Note however, that there is a computational overhead associated with a multiresolution storage format due to the need for decoding and reconstruction. The amount of overhead is determined by the resolution the data needs to be reconstructed to. This computational overhead can be kept small by using fast transforms and coding algorithms such as the 3D S-Transform and Golomb coding, respectively.

Depending on the size of the volume data set, we also found that block-based data storage and processing may provide some advantages, but it may not always be the best choice. The block-based data organization is clearly superior when only parts of the volume are to be loaded, e.g., when only a part of the overall volume is to be rendered. However, when dealing with volume data sets that fit into main memory, a block-based data structure seems to provide little advantages - in particular when taking into account the block management overhead.

References

1. Ihm I, Park S: Wavelet-Based 3D Compression Scheme for Interactive Visualization of Very Large Volume Data. Computer Graphics Forum 18:3–15, 1999.
2. Bajaj C, Ihm I, Park S: 3D RGB Compression for Interactive Applications. ACM Transactions on Graphics 20(1):10–38, 2001.
3. Nguyen K.G., Saupe D.: Rapid High Quality Compression of Volume Data for Visualization. Computer Graphics Forum 20 (3), 2001.
4. Guthe S, Wand M, Gonser J., et al.: Interactive Rendering of Large Volume Data Sets. IEEE Visualization, 2001.
5. Jung M: Two-phase perspective ray casting for interactive volume navigation. IEEE Visualization '97:183-189, 1997.
6. Wendler T and Meyer-Ebrecht D: Proposed standard for variable format picture processing and a codec approach to match diverse imaging devices. SPIE Proceedings, vol. 318:298-305, 1982.
7. Weinberger M, Seroussi G, Sapiro G: LOCO-I: A Low Complexity, Context-Based, Lossless Image Compression Algorithm. IEEE Data Compression Conference: 140–149, 1996.