Volume Graphics (2005)
E. Gröller, I. Fujishiro (Editors)

Volumetric Reconstruction, Compression and Rendering of Natural Phenomena from Multi-Video Data

Lukas Ahrenberg, Ivo Ihrke and Marcus Magnor†

Graphics-Optics-Vision, Max-Planck-Institut für Informatik, Germany

Abstract

Lately, new methods for the acquisition of time-varying, volumetric data for photo-realistic rendering of semi-transparent, volumetric phenomena like fire and smoke have been developed. This paper presents a wavelet-coding and rendering approach for these volumetric sequences that exploits spatial as well as temporal coherence in the data. A space partitioning tree allows for efficient storage and real-time rendering of dynamic, volumetric data on common PC hardware.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Digitizing and scanning I.4.5 [Image Processing and Computer Vision]: ReconstructionI.4.2 [Image Processing and Computer Vision]: Compression (Coding)I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Dynamic, volumetric phenomena in nature such as fire, smoke and swirling fog, are challenging to simulate realistically in computer graphics [FSJ01, NFJ02, ZWF*03]. These are often evolving in a more or less semi-chaotic fashion over time.

To obtain computer models of such phenomena that are suitable for photorealistic image synthesis, optical tomography is introduced in [IM04] as a suitable method to reconstruct volumetric time-varying models from camera images. It is applied to reconstruct three-dimensional volumetric models of flames, but is applicable to thin smoke reconstruction as well.

This article presents a coding and rendering approach for this new kind of data. The data is characterized by high storage cost, but is temporally correlated due to the evolving properties of the underlying phenomena. This facilitates a space-time wavelet compression approach, treating the dataset as a four-dimensional volume. After wavelet transformation the continuity coefficients are stored in a 4D space-partitioning tree which allows for progressive transmission and effective reconstruction. We achieve considerable savings in storage space. The considerably reduced bandwidth consumption then enables real-time rendering frame rates.

The paper is organized as follows: In section 2 we review related work. Section 3 gives some background on the reconstruction of transparent, volumetric data from camera images to give an idea of the nature of our data sets. Section 4 discusses the wavelet compression and our data structure which is essential for real-time rendering. The rendering algorithm is covered in section 5. Finally, we present experiments and results in section 6 and conclude the paper by discussing future work in section 7.

2. Related Work

To reconstruct volumetric phenomena there have been approaches to extend surface reconstruction by taking transparency into account [BV99, SG98]. Computerized tomographic methods have been applied to rigid body reconstructions [GW99]. Transparent, volumetric phenomena are treated by Hasinoff et al. [HK03, Has02]. In Ref. [HK03] the *flame sheet decomposition* algorithm is developed, which reconstructs a surface (the flame sheet) with varying transparency and color. 3D - CT reconstruction is used to generate time-varying, volumetric models of flames in [IM04].

There are a few different approaches to the problem of compressing time-varying volumetric data. Shen

† e-mail: [ahrenberg | ihrke | magnor]@mpi-sb.mpg.de

et.al. [SCM99] implement a structure they call the TSP-tree which is an extended octree structure to represent both spatial and temporal information. Ma and Shen [MS00] also investigate the effects of quantization on time-varying, volumetric data.

Westerman [Wes95] proposes a method where each timeframe of the volume sequence is individually wavelet-compressed. The coefficients are then compared temporally to determine so called time-features which are used to compute the evolution of regions in the volume.

Most similar to the compression scheme in this paper, Linsen et.al. [LPD*02] propose a framework and lifting-scheme for wavelet transforming time-varying volumetric data, treating time as the fourth dimension.

The focus of this paper is on representing dynamic, natural phenomena in an efficient way in order to render directly from this representation. We can rely on a high temporal correlation of the data and apply a 4D wavelet compression to the volume-time space. In contrast to other approaches, we do not use scalar-valued, or color-indexed voxels, but work directly with RGB floating point values.

3. Tomographic Reconstruction of Flames and Smoke

We use the approach presented in [IM04] to generate time-varying volumetric descriptions of fire and smoke. Here, we give a short summary.

We record multi-video sequences of fire or smoke using a multi-camera array consisting of 8 calibrated cameras. Using this data, we compute a linear system of equations that describes the simultaneous rendering of the 8 camera views in terms of unknown voxel values. This linear system is then solved in a non-negative least squares sense. This scheme is comprised of an algebraic CT reconstruction method [KS01], which, unlike Radon transform [Rad17] based methods, allows incorporating additional constraints into the equations of the linear system. We use this flexibility to restrict the solution of the linear system to the visual hull of the volumetric phenomenon. Thus, only voxel values contained in the visual hull are estimated by the reconstruction method. This makes it possible to acquire dynamic, volumetric models of fire and smoke that allow a photo-realistic rendering of the phenomenon from arbitrary viewpoints.

3.1. Image Formation Model

We use the image formation model of Hasinoff et al. [HK03] for our reconstruction algorithm. The fire is modeled as a 3D density field ϕ of fire reaction products, i.e. soot particles. Image intensity is related to the density of luminous particles in the fire. It is valid for fire that obeys the following assumptions:

- Negligible absorption/scattering - this is valid for fire not substantially obscured by smoke, and

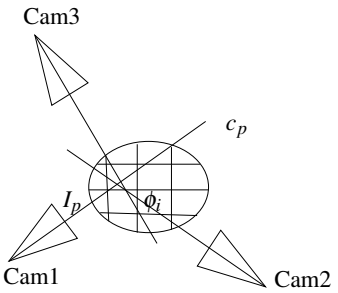

Figure 1: *The reconstruction process: The coefficients of the basis functions ϕ_i are restricted by several camera views. I_p is the image intensity and c_p the backprojected ray of pixel p.*

- Proportional self-emission - brightness depends on the density of the soot particles only.

Additionally, the model works for smoke reconstruction in the following case:

- The smoke is uniformly lit, and
- The smoke is optically thin i.e. there is mostly single scattering.

These assumptions allow us to treat the smoke in a similar way as the self-emissive medium.

3.2. Basic Equations

The image formation model is formulated for every pixel p:

$$I_p = \int_c \phi \; ds + I_{bg}. \tag{1}$$

Here I_p is pixel p's intensity, c is a curve through 3D space, ϕ is the density field of soot particles, and I_{bg} is the background intensity. Curve c is the backprojected ray of pixel p. We approximate every pixel by one ray through the density field.

In order to discretize this equation, which is valid in the whole of $\mathbb{R}^3$, we discretize the part of $\mathbb{R}^3$ that is covered by the volumetric phenomenon using a set of basis functions with local support. Eq. (1) then reads

$$I_p = \int_c \left(\sum_i a_i \phi_i \right) ds + I_{bg}, \tag{2}$$

which is reformulated to

$$I_p = \sum_i a_i \left(\int_c \phi_i \; ds \right) + I_{bg}, \tag{3}$$

yielding the linear system of equations

$$\mathbf{p} = \mathbf{S}\mathbf{a} + \mathbf{b}. \tag{4}$$

The background term **b** is subtracted from the pixel values **p** in a pre-processing step

$$(\mathbf{p}-\mathbf{b}) = \mathbf{S}\mathbf{a}. \tag{5}$$

The simplest basis functions ϕ_i are the box basis functions

$$\phi_i^{Box}(x,y,z) = \begin{cases} 1 & \begin{array}{l} x^i_{min} < x \leq x^i_{max} \\ y^i_{min} < y \leq y^i_{max} \\ z^i_{min} < z \leq z^i_{max} \end{array} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

This choice of basis functions directly results in a voxel model reconstruction, whereas for different basis functions there is an additional interpolation step that needs to be performed to construct a voxel model from the coefficients.

It would be nice to use wavelet basis functions already for the reconstruction process. This is unfortunately complicated by the difficulty of enforcing non-negative voxel values after the interpolation step which is necessary to obtain a physically plausible reconstruction. The difficulty stems from the negative parts of the wavelet basis functions. For non-negative basis functions we simply need to enforce a non-negative solution to the system of equations (4).

3.3. Restrictions

The reconstruction method is able to recover three-dimensional, dynamic voxel models of fully transparent, light-emitting phenomena. In its current version, it can not cope with obstructors present in the volumetric phenomenon. This is similar to artifacts caused by inlays or other metal implants in medical computerized tomographic imaging. Furthermore we require that the pixels of the recorded images are not saturated, and that the entire phenomenon is considered, i.e. no light is emitted outside the discretization, except from the background term. For the full details of this method the interested reader is referred to [IM04].

4. Wavelet Compression of 4D Volumetric Data

The volumetric frames of the reconstructed volumetric animation form a 4-dimensional volume. In this section we propose a 4D wavelet compression scheme to significantly reduce the memory requirements of this volume representation.

4.1. Wavelets

Wavelet compression is nowadays a well-known approach to data reduction. We will therefore just give a brief introduction to the part of the theory useful for our application and point to other sources, such as [Dau92] and [SDS96], for a thorough introduction.

We will assume that f is a function on our data volume and can be written as a linear combination of basis functions

$$f = \sum_{i=0}^{N-1} c_i B_i \tag{7}$$

where N is the number of data elements in the volume.

Wavelet theory is based on two sets of basis functions, the scaling functions

$$\phi_{k,n} = 2^{\frac{k}{2}}\phi(2^k - n), \tag{8}$$

and the wavelet functions

$$\psi_{k,n} = 2^{\frac{k}{2}}\psi(2^k - n). \tag{9}$$

The scaling function down-samples a part of the signal, *scaling* its size and preserving the low frequency, smoothed information. The wavelet function, on the other hand, is chosen so that it can represent the high-frequency information, that is, the details lost by the low-pass filtering of the scaling function. Both the scaling and wavelet functions are scaled and translated versions of a mother function, and form a multi-resolution hierarchy. We will use these properties later when constructing our data-structure for representing the wavelet compressed data. Once a set of wavelet basis functions and corresponding scaling functions are chosen, the recovery of coefficients can be thought of as a recursive process. Start with the original data and use the scaling functions to down-sample the data to half its size, computing the scaling coefficients of this level. At the same time the wavelet functions compute another set of coefficients that represent the high-frequency information.

Now we take the down-sampled data and repeat this process, ending up with half of the original data size, and a new set of coefficients. This can be done until only one scaling function is needed to represent the down-sampled data. Starting from this one scaling value, $s_{0,0}$, the scaled function of the next higher resolution is computed as

$$f^1 = s_{0,0}\phi_{0,0} + w_{0,0}\psi_{0,0}, \tag{10}$$

and for any other level after this as

$$f^{m+1} = f^m + \sum_t w_{m,t}\psi_{m,t} \tag{11}$$

where the sum over t represents the different translations on this scale. Thus f is a linear combination of the basis functions $\phi_{0,0}$ and $\psi_{k,n}$, $0 \leq k < M$, where M is the number of scaling levels. Wavelet basis functions have compact support, and for our application we consider only orthonormal basis functions. For our four-dimensional data we construct the basis functions as the tensor product of one-dimensional

wavelet and scaling functions. This corresponds to the different combinations of the wavelet or scaling function along the coordinate directions.

4.2. Compression

For many data sets a low pass filtering will yield a good approximation of the original set, and thus the high frequency wavelet coefficients will be small. The basic principle of compression is thresholding small coefficients to zero. For basis functions with good interpolating properties, many coefficients can be dropped without degrading data quality. This technique results in high compression ratios, storing only non-zero coefficients.

As the fourth dimension of our data set is time, we can not automatically expect a good correlation along this data axis. However, the focus of our method is to reconstruct natural phenomena, which must evolve continuously over time. Although this evolution may be chaotic in the long run, we can still expect correlation on a frame-by-frame basis given that the data was captured with high enough temporal resolution. From the sequence of thresholded coefficients, we keep only the ones that are non-zero in value, as well as their corresponding basis functions. In order to store this information and allow for both small size and fast access, we need a compact data structure.

4.3. The Hexadeca-tree data structure

Because a wavelet basis function has local support and is used to refine the value of a coarser scale function, the support of a basis function of side s with index k will span the region of support of the $s/2$ sized functions of index $k+1$. That is, for a basis function, $\psi_{k,t}$, of scale $k > 0$, one can find basis functions of a coarser scale $j < k$, so that their support contains the support of $\psi_{k,t}$. Just as observed in section 4.1, the support of the basis functions divide the original support of the data set into sub sets.

The basis functions form a space partitioning tree analogous to quad- and octrees in 2D and 3D. Using this tree we let each node represent all basis functions of a specific scale and translation. The child-nodes will be those basis functions refining the value along their parent node's support, subdividing them. Figure 2 shows a binary tree for the 1D case of the situation described above. In this case, there would be two basis functions, one wavelet and one scaling function per node. However for reconstruction purposes it is sufficient to store only the wavelet basis functions and their corresponding coefficients, except for the root node which also contains the scaling coefficient and the scaling basis function. Higher-dimensional trees will have more functions of the same support, as there are more ways to combine the basis functions. Generally, for a n-dimensional space, $2^n - 1$ wavelet basis functions will have the same support. All basis functions with the same support are represented by the same tree node. In our 4D case, there will be 16 different basis functions defined having the same support. Of these, the 15 wavelet basis function coefficients will be stored in the nodes. The 16:th, being the scaling function coefficient, is not required for reconstruction, except at the coarsest scale.

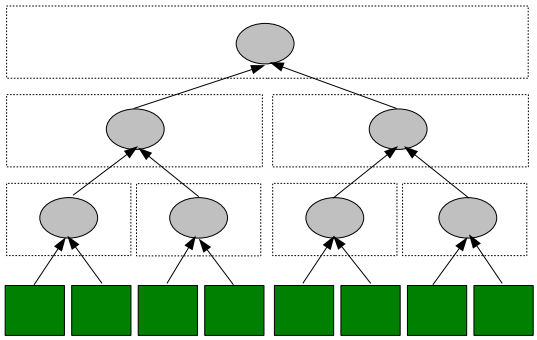

Figure 2: *A binary tree representation. The squares represent the original data-points, and the ellipse nodes contain a basis function and the corresponding coefficient. The dashed box around the node indicates the width of the support of the function.*

At the top level of the tree, we have the coarsest scale, represented by the scaling function coefficient and the wavelet coefficient refining it. The support of the root node being effectively the whole data size. A the children of a node will then subdivide the support of a parent in a hierarchical manner.

If all basis functions are multiplied with the corresponding coefficients and summed up, the result will be the original function. In our 4D approach, the result will not be a binary tree, but one where each node has 16 children, representing the 16 equally sized sub-cubes of a 4D hypercube. We will refer to this structure as a *hexadeca-tree*.

We can reduce the memory usage of the data-structure drastically by pruning the tree in a bottom-up approach after compressing the coefficients. Because the leaf-nodes with zero-coefficients do not contribute to the reconstructed signal, they can be removed from the tree. If all children of a node are removed, it becomes a leaf and the same test can be applied again until we find a node that can not be removed. Figure 3 shows a simple, binary tree example.

4.4. Implementing the data-structure

The pruning method described above leaves us with a smaller tree than we started with. But it cannot do anything about those nodes which are not leaves and still contain many zero-valued coefficients. This case can be common when dealing with 4D data, as each node contains several coefficients. If only one of these coefficients of a leaf is non-zero, it will be kept. Therefore we choose to have a

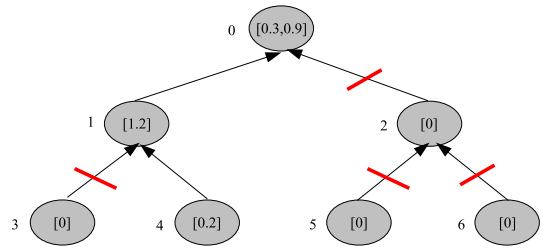

Figure 3: *Pruning the nodes from the bottom up in a binary space partitioning tree. Starting with node 6, we can remove it because its coefficient is zero. Node 5 can be removed for the same reason, and node 2 now becomes a leaf. This node can be removed as it is 0. Node 4 is saved and node 3 is removed, but then the pruning stops since there are no more leaves to check.*

data structure of dynamic size to represent the nodes of the hexadeca-tree.

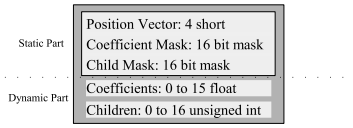

Figure 4: *The node data-structure. The position vector and the masks make up the static part, present in every node, while the coefficient and the children sections are of dynamic size.*

A schematic of the node data-structure is depicted in Figure 4. For each node we store a static and a dynamic part. The static part contains the node position and two 16 bit masks. The 'Coefficient Mask' specifies which of the possible basis functions in the node have non-zero coefficients. The 'Child Mask' defines which of the children exist in a pruned tree. The dynamic part contains just the non-zero coefficients and index offsets to any child nodes, as indicated by the masks. Every node stores basis functions having the same support, which are the ones resulting from the tensor product producing the four-dimensional basis functions from the one-dimensional mother wavelet. The reason we only need to store 15 coefficients per node, although there are 16 different functions resulting from such an operation, is that one of them is the scaling function, which is not needed for reconstructing the data. The only scaling function coefficient needed is the one at the coarsest scale, as indicated in section 4.1, and this one is stored first in the array as a special case.

The compressed and pruned tree is an efficient representation for transmission and rendering. There is, of course, some potential overhead in this representation, as in the worst case all 15 coefficients will be set in the dynamic section. However this is rarely the case. The nodes are stored

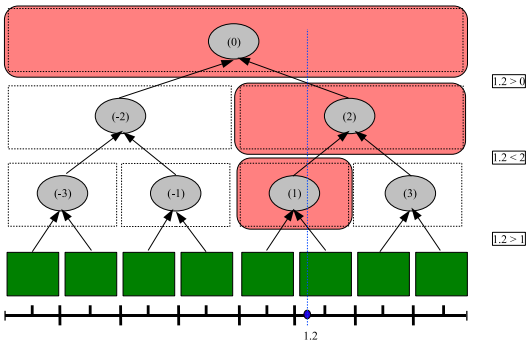

Figure 5: *Traversal of a basis node tree, reconstructing the value at position 1.2. The method starts at the root node, and checks in which child the coordinate lies. Nodes traveled through are highlighted.*

breadth-first in an array. This facilitates progressive decoding so that time-, transmission- or memory-critical applications need only read and decode a part of the tree to obtain approximated rendering results. The approach is similar to the spatial orientation trees in the SPIHT codec for images [SP96].

5. Data Reconstruction and Rendering

5.1. Data Reconstruction

To compute the function value at a point x, it is not necessary to first compute the whole data set f and then evaluate $f(x)$. Instead, we observe that a point located in the support of one node is bound to be located in the support of one of its children in the hexadeca-tree. That is

$$f(x) = \sum_{\Omega} c_i B_i \tag{12}$$

where $B_i \in \Omega$ implies that x lies in the local support of B_i. Because the children of a node sub-divide its support, it is simple to compute Ω, starting at the root node. Given that a node contains x it is only necessary to check in which of the node's children the point lies. We do this recursively until a leaf node is reached. The reconstruction sum will thus take the form of a traversal through the space partition tree. The method is depicted in Figure 5.

This is an efficient way of reconstructing subsets of a wavelet decomposed function, and is similar to traditional hierarchical volume rendering strategies [Lev90, LH91, KCY93]. We know that the support of the next basis functions in the sum will be contained within the support of the current. The number of summations needed to reconstruct a value in a direct approach will thus be the depth of the tree, which is logarithmically dependent on the resolution of the data. For our approach this is an important observation as the reconstruction is a 3D volume intersection of a

```
Given the view-matrix, M.
for every voxel position p do
 let v = [p,0,1];
 let u = M*v;
 let this_node = root node;
 let value = scaling coefficient;
 repeat while this_node exists
  for all basis B in this_node do
   value += coefficient * B(u);
  end for;
  for every child of this_node do
   if child is in this_node.support
    this_node = child;
   end if;
  end for;
 end repeat;
 set Image[p[0],p[1]] += value;
end for;
```

Figure 6: *Algorithm for rendering an image view from a hexadeca-tree representation.*

4D dataset. Thus only a subset of all data points needs to be reconstructed in each step.

5.2. Rendering

To render a view of our tomographic data we need to integrate a 3D volume along the viewing direction. By translating this volume along the temporal axis in our 4D data-space we can render time-slices of the volume. If we first rotate the view-volume around the spatial (x,y,z) axis we can also view the scene from an arbitrary position centered around the origin. This mapping is a 5x5 view-matrix for 4-dimensional homogeneous coordinates. The view matrix, M, for orthogonal projection is arranged as

$$M = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & 0 & 0 \\ r_{1,0} & r_{1,1} & r_{1,2} & 0 & 0 \\ r_{2,0} & r_{2,1} & r_{2,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (13)$$

The upper 3x3 sub-matrix consists of a 3D transformation matrix that aligns the viewing volume with the current viewing direction. t is the the translation along the temporal axis. The matrix transforms coordinates from the viewing volume coordinate system into the 4D data coordinate system of the hexadeca-tree.

We use the algorithm in Figure 6 to render the data. Every voxel in our viewing volume is transformed into data-space coordinates by multiplication with the view matrix M. The value is reconstructed by summing the node coefficient-basis function products while traversing the hexadeca-tree as described above. Every node with support containing the voxel is traversed. The algorithm stops when it reaches a leaf.

The rendering algorithm can be sped up by storing a 3D bounding box with every temporal step. It costs some more memory, but can be a good trade-off if the data occupies only a part of the volume. For our current approach we have assumed an orthographic projection, and thus have a block-shaped viewing volume. A perspective projection can also be implemented by using a viewing frustum instead of a block. The coordinates can then be projected before accumulating the results in the image.

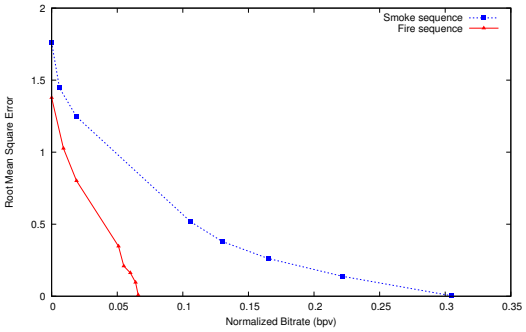

Figure 7: *RMS error varying with bit rates of the hexadeca-tree memory print.*

The volume can be integrated as part of a virtual scene by dynamically textured billboards [Sch95]. The current viewing direction of the camera is computed, together with the animation time. The volume at the current time is then rendered to a 2D texture, as seen from the camera. The intensity-value of the texel is used as the alpha value. The texture is then mapped onto a polygon translated to the volume position and rotated to always face the camera.

6. Results

We have used two different data sets, one of smoke and the other of fire. Both sequences are of 64x64x64 voxels in size and contain 64 frames. A sequence consists of 64^4 voxels, each having a RGB floating point value, yielding a raw data size of 192 MB. We used the Haar wavelets as basis functions. The rendering was performed using the bill-boarding technique described in the previous section implemented in OpenGL on a Pentium 4 workstation. We performed compression and rendering tests using both data sets.

Figure 7 shows how the Root Mean Square (RMS) error of the reconstructed volumetric sequence varies with the number of bits per voxel needed to represent the sequence at different compression levels. The normalized bit rate is computed as the RAM memory size of the hexadeca-tree data structure (as opposed to the storage-size on disk, where we use zlib to compress the data, and usually gain 25–30% smaller files) divided by the original sequence data size. As can be seen in the figure, the 'Fire' sequence has a very

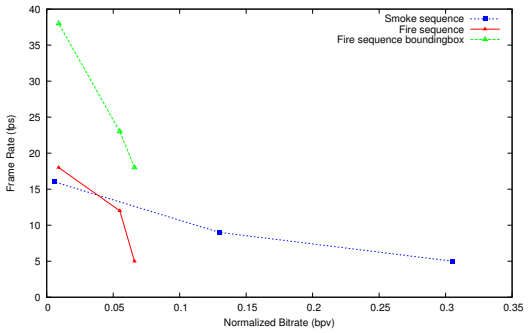

Figure 8: *Frame rate decreasing with higher bit rates.*

low maximum bit rate at almost no error cost at all. However, as the bit rate decreases the error increases drastically. The low maximum bit rate comes from the high number of zero voxels in the original data; the fire is located approximately at the center of the volume, and although varying in shape, it almost never occupies a majority of the volume. The zeros will of course be removed when using the hexadeca-tree representation, and cause a major compression without any quality loss. The dependence of quality on bitrate can be explained by the fast movement of the flame, changing much faster than the frame rate of our camera system. Thus, two concurrent volumes in the flame sequence are not well-enough correlated and compression in the temporal dimension of the hexadeca-tree will be more sensitive to coefficient removal. The 'Smoke' sequence puts the wavelet compression scheme to better use. It has just as chaotic, dynamic behavior as the fire, but spans more of the volume and evolves slower. As seen from the graph, the maximum bit rate is higher than that of the 'Fire' sequence, but it also degrades more slowly and allows for good compression. Figure 9 shows the visual quality for some of the bit rates in the graph. As can be seen, we achieve visually pleasing results, even at moderately low bitrates, such as the fire at 0.051 and the smoke at 0.106. However, at some point the error rises too high and the volume integration fails.

We have implemented a viewer which loads a hexadeca-tree array into memory and generates textures on the fly given a viewing direction and position, as described in Section 5. The highest bit rates for the 'Smoke' and 'Fire' sequences renders at 5 and 7 fps. We also precalculated bounding boxes for the frames in both sequences which results in a performance gain to 18 fps for the 'Fire' sequence. As expected, the frame rate increases with lower bit rates. Figure 8 depicts this.

7. Conclusions and Future Work

We have shown how volumetric models of dynamically changing, chaotic phenomena of fire and smoke can be rendered. Using models captured from the real world might prove to be a valuable alternative to simulated models in some computer graphics applications. We have devised a wavelet compression scheme for the time-varying models, treating them as a 4D data space. The wavelet coefficients are stored in a hexadeca-tree structure which allows for progressive decoding and fast reconstruction. The volume data can be incorporated into interactive 3D models by a dynamic bill-boarding approach.

We plan to continue our work by extending the compression and rendering capabilities of our system. For this paper we have used 64^3 volumes, which gives satisfactory results. However, we will have higher resolution volumes in the near future, which should give even better visual results. Because the Haar wavelet basis has limited interpolating properties, a logical future step in compression will be to try out other wavelet basis functions. Although our "divide-and-conquer" software rendering approach is efficient and can render directly from the hexadeca-tree representation, a number of interesting approaches using graphics hardware have been proposed [LG95, EKE01, GWGS02, BCF03]. We would like to investigate the use of these methods in our scheme to see if we can improve performance.

References

[BCF03] BINOTTO A. P. D., COMBA J., FREITAS C. M. D. S.: Real-time volume rendering of time-varying data using a fragment-shader compression approach. In *IEEE Symposium on Parallel and Large-Data Visualization and Graphics* (2003), pp. 69–76.

[BV99] BONET J. S. D., VIOLA P. A.: Roxels: Responsibility Weighted 3D Volume Reconstruction. In *Proc. International Conference on Computer Vision (ICCV '99)* (1999), pp. 418 – 425.

[Dau92] DAUBECHIES I.: *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, 1992.

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), ACM Press, pp. 9–16.

[FSJ01] FEDKIW R., STAM J., JENSEN H. W.: Visual Simulation of Smoke. *Proceedings of SIGGRAPH* (August 2001), 15–22.

[GW99] GERING D. T., WELLS III W. M.: Object Modeling using Tomography and Photography. In *Proc. of IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes* (June 1999), pp. 11–18.

[GWGS02] GUTHE S., WAND M., GONSER J., STRASSER W.: Interactive rendering of large volume data sets. In *VIS '02: Proceedings of the conference on Visualization '02* (2002), IEEE Computer Society, pp. 53–60.

[Has02] HASINOFF S. W.: Three-Dimensional Reconstruction of Fire from Images. MSc Thesis, University of Toronto, Department of Computer Science, 2002.

[HK03] HASINOFF S. W., KUTULAKOS K. N.: Photo-Consistent 3D Fire by Flame-Sheet Decomposition. In *In Proc. 9th IEEE International Conference on Computer Vision (ICCV '03)* (2003), pp. 1184 – 1191.

[IM04] IHRKE I., MAGNOR M.: Image-Based Tomographic Reconstruction of Flames . *ACM Siggraph / Eurographics Symposium Proceedings, Symposium on Computer Animation* (June 2004), 367–375.

[KCY93] KAUFMAN A., COHEN D., YAGEL R.: Volume graphics. *Computer 26*, 7 (1993), 51–64.

[KS01] KAK A. C., SLANLEY M.: *Principles of Computerized Tomographic Imaging*. Society of Industrial and Applied Mathematics, 2001.

[Lev90] LEVOY M.: Efficient ray tracing of volume data. *ACM Trans. Graph. 9*, 3 (1990), 245–261.

[LG95] LIPPERT L., GROSS M. H.: Fast wavelet based volume rendering by accumulation of transparent texture maps. *Comput. Graph. Forum 14*, 3 (1995), 431–444.

[LH91] LAUR D., HANRAHAN P.: Hierarchical splatting: a progressive refinement algorithm for volume rendering. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), ACM Press, pp. 285–288.

[LPD*02] LINSEN L., PASCUCCI V., DUCHAINEAU M. A., HAMANN B., JOY K. I.: Hierarchical representation of time-varying volume data with "4th-root-of-2" subdivision and quadrilinear b-spline wavelets. In *Pacific Conference on Computer Graphics and Applications* (2002), pp. 346–355.

[MS00] MA K.-L., SHEN H.: Compression and accelerated rendering of time-varying volume data. In *Proceedings of the Workshop on Computer Graphics and Virtual Reality* (2000).

[NFJ02] NGUYEN D. Q., FEDKIW R., JENSEN H. W.: Physically Based Modelling and Animation of Fire. *ACM Transactions on Graphics 21*, 3 (July 2002), 721–728.

[Rad17] RADON J.: Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. Ber. Ver. Sachs. Akad. Wiss. Leipzig, Math-Phys. Kl., 69:262 277, April 1917.

[Sch95] SCHAUFLER G.: Dynamically Generated Impostors. *GI Workshop on Modeling, Virtual Worlds* (Nov. 1995), 129–135.

[SCM99] SHEN H.-W., CHIANG L.-J., MA K.-L.: A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. In *VIS '99: Proceedings of the conference on Visualization '99* (Los Alamitos, CA, USA, 1999), IEEE Computer Society Press, pp. 371–377.

[SDS96] STOLLNITZ E. J., DEROSE T. D., SALESIN D. H.: *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., 1996.

[SG98] SZELISKI R., GOLLAND P.: Stereo Matching with Transparency and Matting. In *Proceedings Sixth International Converence on Computer Vision* (1998), pp. 517–524.

[SP96] SAID A., PEARLMAN W. A.: A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology 6* (1996), 243–250.

[Wes95] WESTERMANN R.: Compression domain rendering of time-resolved volume data. In *IEEE Visualization* (1995), pp. 168–175.

[ZWF*03] ZHAO Y., WEI X., FAN Z., KAUFMAN A., QIN H.: Voxels on Fire. In *Proc. 14th IEEE Visualization Conference (VIS'03)* (October 2003), pp. 271–278.