X-Ray Rendering Through Two-Stage Splatting

Michel A. Westenberg and Jos B.T.M. Roerdink

Institute for Mathematics and Computing Science
University of Groningen
P.O. Box 800, 9700 AV Groningen, The Netherlands
E-mail: michel@cs.rug.nl, roe@cs.rug.nl

Abstract

We consider the splatting method for X-ray rendering of volume data sets, and introduce two-stage splatting. This variant improves rendering speed by separating the splatting process in two stages: coefficient accumulation and convolution. We incorporate this variant into wavelet splatting, and obtain speedups in rendering time up to a factor of three for Haar and B-spline wavelets.

1 Introduction

X-ray volume rendering remains one of the preferred techniques for medical applications, because physicians are well-trained in interpreting X-ray like images for diagnosis. Usual implementations are based on either ray-casting or splatting. Splatting [7] is an object order method in which the voxels are represented by 3-D reconstruction kernels. Integration of these kernels along the line of sight results in building blocks called 'splats' or 'footprints'. A mapping to the image plane by superposition of the footprints weighted by the voxel values forms the image in the view plane. In this paper, we discuss the splatting method, and we show that it is possible to split the splatting process in two stages: (i) coefficient accumulation and (ii) a final 2-D convolution with the footprint. This procedure saves many multiplications, and, as a consequence, reduces rendering time.

In addition, we incorporate the two-stage approach into *wavelet splatting*. This is a wavelet-based extension to splatting which was introduced by Lippert et al. [3], and is a multiresolution variant of X-ray volume rendering. Wavelet splatting modifies the splatting algorithm by using wavelets as 3-D reconstruction filters, so that data can be visualized at different levels of detail. One of the reasons to use wavelets in volume visualization is the desire to exchange volume data through systems such as the Internet. This has created a need for fast and efficient methods of transfer and display of 3-D data. Volume data may be stored on a central server, and (part of) the rendering may be performed on client systems to relieve the demand on the server capacity. Due to their size, transmission of the data sets is time consuming. This motivates the need for compression methods and mechanisms to visualize the data incrementally as it arrives ('progressive refinement'). For this purpose, wavelet-based multiresolution models have been developed, which allow systematic decomposition of the data into versions at different levels of resolution. Local level-of-detail (LOD) is another benefit of such methods, i.e., small, distant, or unimportant parts of the data are represented at lower resolution. Wavelet-based methods also have proved to perform well in compression of images [1] and volume data [5].

The remainder of this paper is organized as follows. In Section 2, we introduce the two-stage splatting algorithm. Section 3 describes wavelet splatting and the incorporation of two-stage splatting into this algorithm. In Section 4, we compare the standard implementation of wavelet splatting and the two-stage variant, and present timing results for rendering. Section 5 contains some conclusions.

2 Two-stage splatting

We discuss splatting in the context of X-ray rendering. The mathematical basis for this is the X-ray transform, well-known from computerized tomography [4]. This transform computes line integrals of a given function. In volume visualization, this amounts to simple addition of the values in a data volume along the viewing direction. For this particular case, we show that it is possible to separate the splatting process in two stages: (i) accumulation of the voxels in a weight image, and (ii) convolution of the weight image with the footprint. As will turn out, this separation reduces rendering time greatly.

Consider the line integrals of a continuous function $f(\mathbf{x})$, $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$, along a direction defined by a unit vector $\boldsymbol{\theta}$ (viewing direction). Let $\mathbf{u}$ and $\mathbf{v}$ be two mutually orthogonal unit vectors perpendicular to $\boldsymbol{\theta}$. The X-ray transform $\mathcal{P}_{\boldsymbol{\theta}} f$ of f is defined by

$$\begin{aligned} \mathcal{P}_{\boldsymbol{\theta}} f(u, v) &= \int_{\mathbb{R}^3} f(\mathbf{x}) \delta(\mathbf{x} \cdot \mathbf{u} - u) \delta(\mathbf{x} \cdot \mathbf{v} - v) \, \mathrm{d}\mathbf{x} \\ &= \int_{\mathbb{R}} f(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta}) \, \mathrm{d}t, \end{aligned} \tag{1}$$

where the dot denotes the inner product of vectors in $\mathbb{R}^3$. In the discrete case, we have a collection of samples $c_{k,l,m}$ of the function f. The splatting algorithm reconstructs f from its samples by convolution with a reconstruction filter ϕ, and then computes a mapping to the view plane. This is expressed by

$$\begin{aligned} f(x, y, z) &= \sum_{k,l,m} c_{k,l,m} \phi_{k,l,m}(x, y, z), \\ \mathcal{P}_{\boldsymbol{\theta}} f(u, v) &= \sum_{k,l,m} c_{k,l,m} \int_{\mathbb{R}} \phi_{k,l,m}(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta}) \, \mathrm{d}t, \end{aligned} \tag{2}$$

where $\phi_{k,l,m}(x, y, z) = \phi(x-k, y-l, z-m)$ (for simplicity, we assume that the sampling distances of the volume data are all equal to one). The integral over ϕ results in a two-dimensional function which is called a footprint. Rendering amounts to multiplication of the footprint by the corresponding coefficient $c_{k,l,m}$, and blending the result into the frame buffer in which the view plane image is accumulated. The original splatting algorithm [7] traverses the voxels in a back-to-front (of front-to-back) order to facilitate occlusion effects. For X-ray rendering, the voxels can be traversed in an arbitrary order, and blending corresponds to simple addition. This latter observation is exploited in two-stage splatting.

Define the footprint $F_{\boldsymbol{\theta}}$ by

$$F_{\boldsymbol{\theta}}(u, v) = \int_{\mathbb{R}} \phi(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta}) \, \mathrm{d}t.$$

Then, it is easy to derive that (2) is equivalent to

$$\mathcal{P}_{\boldsymbol{\theta}} f(u, v) = \sum_{\mathbf{k}} c_{\mathbf{k}} F_{\boldsymbol{\theta}}(u - \mathbf{k} \cdot \mathbf{u}, v - \mathbf{k} \cdot \mathbf{v}), \tag{3}$$

where $\mathbf{k} = (k, l, m)$. In the discrete case, the coordinates $(\mathbf{k} \cdot \mathbf{u}, \mathbf{k} \cdot \mathbf{v})$ of the projection of the position vector $\mathbf{k}$ are not always integer, so we need to interpolate between samples. There are a number of ways to implement this. If the reconstruction kernel is isotropic, the footprints are independent of the viewing direction. In this case, the footprint can be sampled with a very small sampling step, and interpolation in the frame buffer can be done by nearest neighbour interpolation. However, if the reconstruction kernel is non-isotropic, the footprints have to be computed again with each change of viewing direction. In this case, sampling the footprints with a small sampling step is computationally too expensive. Therefore, the footprints are sampled with the sampling step of the view plane, and interpolation in the frame buffer is done by bilinear interpolation. Since we use non-isotropic reconstruction kernels later on, we consider bilinear interpolation in the frame buffer.

Let $P[i, j] := \mathcal{P}_{\boldsymbol{\theta}}(i, j)$, $i, j = -N, -N+1, \ldots, N-1, N$, be a discretization of $\mathcal{P}_{\boldsymbol{\theta}}$. We want to express $P[i, j]$ as a combination of $F_{\boldsymbol{\theta}}[p, q]$, with p, q integer, and coefficients $c_{\mathbf{k}}$. Let $u_{\mathbf{k}} = \mathbf{k} \cdot \mathbf{u}$,

$u_{\mathbf{k}}^{\uparrow} = \lceil u_{\mathbf{k}} \rceil$ and $u_{\mathbf{k}}^{\downarrow} = \lfloor u_{\mathbf{k}} \rfloor$, with $v_{\mathbf{k}}$, $v_{\mathbf{k}}^{\uparrow}$ and $v_{\mathbf{k}}^{\downarrow}$ defined similarly. Here $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denote the ceiling and floor operator, respectively. Bilinear interpolation yields

$$\begin{aligned} P[i,j] = \sum_{\mathbf{k}} c_{\mathbf{k}} \{ & F_{\boldsymbol{\theta}}[i - u_{\mathbf{k}}^{\uparrow}, j - v_{\mathbf{k}}^{\uparrow}] \, (u_{\mathbf{k}} - u_{\mathbf{k}}^{\downarrow}) \, (v_{\mathbf{k}} - v_{\mathbf{k}}^{\downarrow}) \\ & + F_{\boldsymbol{\theta}}[i - u_{\mathbf{k}}^{\uparrow}, j - v_{\mathbf{k}}^{\downarrow}] \, (u_{\mathbf{k}} - u_{\mathbf{k}}^{\downarrow}) \, (v_{\mathbf{k}} - v_{\mathbf{k}}^{\uparrow}) \\ & + F_{\boldsymbol{\theta}}[i - u_{\mathbf{k}}^{\downarrow}, j - v_{\mathbf{k}}^{\uparrow}] \, (u_{\mathbf{k}} - u_{\mathbf{k}}^{\uparrow}) \, (v_{\mathbf{k}} - v_{\mathbf{k}}^{\downarrow}) \\ & + F_{\boldsymbol{\theta}}[i - u_{\mathbf{k}}^{\downarrow}, j - v_{\mathbf{k}}^{\downarrow}] \, (u_{\mathbf{k}} - u_{\mathbf{k}}^{\uparrow}) \, (v_{\mathbf{k}} - v_{\mathbf{k}}^{\uparrow}) \}. \end{aligned}$$

It is possible to split this expression in a part which involves the coefficients, and a part which involves the footprint. Define an array of weights $W[p,q]$, $p, q = -N, -N+1, \ldots, N-1, N$, such that $W[p,q]$ contains the accumulated contributions of coefficients $c_{\mathbf{k}}$ that map to position (p,q). Mathematically,

$$\begin{aligned} W[p,q] = \sum_{\mathbf{k}} c_{\mathbf{k}} \{ & \delta_{p,u_{\mathbf{k}}^{\uparrow}} \delta_{q,v_{\mathbf{k}}^{\uparrow}} (u_{\mathbf{k}} - u_{\mathbf{k}}^{\downarrow})(v_{\mathbf{k}} - v_{\mathbf{k}}^{\downarrow}) + \delta_{p,u_{\mathbf{k}}^{\uparrow}} \delta_{q,v_{\mathbf{k}}^{\downarrow}} (u_{\mathbf{k}} - u_{\mathbf{k}}^{\downarrow})(v_{\mathbf{k}} - v_{\mathbf{k}}^{\uparrow}) \\ & + \delta_{p,u_{\mathbf{k}}^{\downarrow}} \delta_{q,v_{\mathbf{k}}^{\uparrow}} (u_{\mathbf{k}} - u_{\mathbf{k}}^{\uparrow})(v_{\mathbf{k}} - v_{\mathbf{k}}^{\downarrow}) + \delta_{p,u_{\mathbf{k}}^{\downarrow}} \delta_{q,v_{\mathbf{k}}^{\downarrow}} (u_{\mathbf{k}} - u_{\mathbf{k}}^{\uparrow})(v_{\mathbf{k}} - v_{\mathbf{k}}^{\uparrow}) \}, \end{aligned} \tag{4}$$

where $\delta_{k,l}$ is the Kronecker delta function. Then, it is easy to verify that P is the result of convolving W and $F_{\boldsymbol{\theta}}$:

$$P[i,j] = \sum_{p,q} W[p,q] F_{\boldsymbol{\theta}}[i-p, j-q]. \tag{5}$$

Altogether, the splatting process is split in two stages. In the first stage, the voxels are projected on the view plane and accumulated in the array W as expressed by (4). The second stage computes a convolution of W with the footprint $F_{\boldsymbol{\theta}}$. Two-stage splatting is more efficient than a direct implementation of (3), where the footprint is blended into the frame buffer for each voxel. This requires interpolation of all elements of the footprint, whereas two-stage splatting only interpolates a single voxel, which saves many multiplications. The final convolution requires no interpolation. The details are summarized in pseudo-code in Algorithm 1.

Algorithm 1 *Two-stage splatting*

Input: 3-D array of voxels c, footprint $F_{\boldsymbol{\theta}}$, vectors $\mathbf{u}$ and $\mathbf{v}$ defining the view plane
Output: view plane image I

For all voxels (k,l,m) do
  ▷ Compute the image location (u,v) of the projection of voxel (k,l,m)
  $u = \mathbf{u} \cdot (k,l,m)$
  $v = \mathbf{v} \cdot (k,l,m)$
  ▷ Add $c_{k,l,m}$ to the array W by bilinear interpolation
  $\alpha = u - \lfloor u \rfloor$
  $\beta = v - \lfloor v \rfloor$
  $W(\lfloor u \rfloor, \lfloor v \rfloor) \mathrel{+}= (1-\alpha)(1-\beta) c_{k,l,m}$
  $W(\lceil u \rceil, \lfloor v \rfloor) \mathrel{+}= \alpha(1-\beta) c_{k,l,m}$
  $W(\lfloor u \rfloor, \lceil v \rceil) \mathrel{+}= (1-\alpha)\beta c_{k,l,m}$
  $W(\lceil u \rceil, \lceil v \rceil) \mathrel{+}= \alpha\beta c_{k,l,m}$
▷ Compute the convolution of W and the footprint $F_{\boldsymbol{\theta}}$
$I = F_{\boldsymbol{\theta}} * W$

3 Incorporation into Wavelet Splatting

Wavelets allow systematic decomposition of data into versions at different levels of resolution. This is of interest for volume visualization, since the lower resolution data can be processed much faster, which allows better interaction. Wavelet splatting [3] uses this property of wavelets, and computes low resolution images during user interaction. When interaction ceases, the images are refined incrementally to full resolution. For more details, see [6], where also a frequency domain approach to wavelet-based X-ray rendering is presented.

A 1-D biorthogonal wavelet basis can be constructed from a *scaling function* ϕ with associated wavelet ψ, and dual scaling function $\widetilde{\phi}$ and dual wavelet $\widetilde{\psi}$. The corresponding basis functions are $\{\phi_{j,k}\}$ and $\{\psi_{j,k}\}$, $j,k \in \mathbb{Z}$, where $\phi_{j,k}(x) = 2^{-j/2}\phi(2^{-j}x - k)$ and $\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)$. The dual basis functions are defined similarly. The parameters j and k denote scale and translation, respectively. From the 1-D basis, one constructs a 3-D separable wavelet basis (of the so-called nonstandard type) with eight basis functions, i.e. one scaling function $\Phi^0_{j,k,l,m}(x,y,z)$ and seven wavelet basis functions $\Psi^\tau_{j,k,l,m}(x,y,z)$, $\tau \in T = \{1,2,\ldots,7\}$ by taking tensor products of the 1-D scaling functions $\phi_{j,k}$ and wavelets $\psi_{j,k}$.

Wavelet splatting modifies the basic splatting algorithm in two ways: (i) it uses wavelets as reconstruction filters, and (ii) it provides a mechanism to visualize data at different levels of detail. First, the algorithm performs a 3-D wavelet decomposition of the volume data. Substitution of the expansion of f on this basis in (1) results in:

$$\begin{aligned}\mathcal{P}_{\boldsymbol{\theta}} f(u,v) &= \sum_{k,l,m} c^M_{k,l,m} \int_{\mathbb{R}} \Phi^0_{M,k,l,m}(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta})\,\mathrm{d}t \\ &+ \sum_{j=1}^{M} \sum_{\tau\in T} \sum_{k,l,m} d^{j,\tau}_{k,l,m} \int_{\mathbb{R}} \Psi^\tau_{j,k,l,m}(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta})\,\mathrm{d}t.\end{aligned}$$

Again, the integrals result in 2-D functions defined on the view plane: the footprints. The integrals have to be evaluated only once for a given viewing direction at the coarsest scale $j = M$ and translation $(k,l,m) = (0,0,0)$, yielding eight *prototype* footprints. The footprints for other scales and translations can be computed by rescaling and shifting. Prototype footprints can be computed efficiently by slicing their 3-D Fourier transforms. If analytical expressions exist for the Fourier transforms of the scaling function and wavelet, as is the case for the Haar and cardinal B-spline wavelets, no interpolation from discrete samples is necessary.

The incorporation of two-stage splatting into wavelet splatting is straightforward. Consider an approximation f^M at level M, which is computed from the approximation coefficients $c^M_{\mathbf{k}} := c^M_{k,l,m}$ only. It is possible to derive an equation similar to (3). The only difference is that, now, $F_{\boldsymbol{\theta}}$ is the result of integrating the 3-D scaling function $\Phi^0_{M,0,0,0}$ and that $\mathbf{k}\cdot\mathbf{u}$ is replaced by $2^M\mathbf{k}\cdot\mathbf{u}$, and $\mathbf{k}\cdot\mathbf{v}$ by $2^M\mathbf{k}\cdot\mathbf{v}$:

$$\begin{aligned}F^M_{\boldsymbol{\theta}}(u,v) &= \int_{\mathbb{R}} \Phi^0_{M,0,0,0}(u\mathbf{u} + v\mathbf{v} + t\boldsymbol{\theta})\,\mathrm{d}t, \\ \mathcal{P}_{\boldsymbol{\theta}} f^M(u,v) &= \sum_{\mathbf{k}} c^M_{\mathbf{k}} F^M_{\boldsymbol{\theta}}(u - 2^M\mathbf{k}\cdot\mathbf{u}, v - 2^M\mathbf{k}\cdot\mathbf{v}).\end{aligned} \tag{6}$$

A similar derivation can be made for terms involving the wavelets $\Psi^\tau_{j,k,l,m}$. Obviously, we can use the result from (5), and split the splatting process again in coefficient accumulation followed by convolution.

We can now formulate a rendering algorithm. This algorithm consists of two phases. The preprocessing phase computes a 3-D wavelet transform. The user controls the depth M of the transform, which is typically two or three, depending on the dimensions of the data set and the desired level of detail for the lowest resolution image. Next, a sequence of coefficients is constructed for each wavelet type and decomposition depth. For instance, a two-level wavelet decomposition results in one sequence of approximation coefficients at level 2, seven sequences of detail coefficients at level 2, and seven sequences of detail coefficients at level 1. The spatial position of each coefficient is stored in the sequence as well. Each sequence is then sorted in descending order with respect to magnitude, and all coefficients that are equal to zero are discarded.

The rendering phase performs the actual splatting and uses the convolution formula (5) to speed up the rendering process. After selection of the viewing direction $\boldsymbol{\theta}$, the prototype footprints are computed by Fourier slicing. From these, the footprints for lower levels are computed by scaling and downsampling. The algorithm computes a low resolution image from the approximation coefficients only, cf. (6). While the user interacts with the data, i.e. selects new viewing directions, only low resolution images are computed. If interaction ceases, the image is refined incrementally with the detail coefficients.

4 Results

We applied the rendering algorithm introduced in the previous section to three different data sets to investigate its performance. A phantom data set containing $128 \times 128 \times 128$ voxels at 8 bits per voxel was generated from a mathematical description of ellipsoids, as a 3-D variant of the Shepp-Logan head phantom used to assess the quality of tomographic reconstruction algorithms [2]. We also used a CT head data set of $128 \times 128 \times 128$ voxels at 16 bits per voxel, and an MR head data set of $256 \times 256 \times 256$ voxels at 16 bits per voxel. As basic wavelets both a Haar wavelet and a linear B-spline wavelet were used.

Timings were carried out on a PC with a 400 MHz Pentium II processor and 128 MB memory. A two-level wavelet transform was used for the phantom and CT head data sets, and a three-level wavelet transform for the MR head data set. This was done in order to have about the same number of approximation coefficients for all data sets. Another reason to choose these particular decomposition depths is that the decomposition depth cannot be set larger for the linear B-spline wavelets, due to the length of the filters used for the wavelet decomposition (41 coefficients). We compare the variant which blends the footprint in the frame buffer for each coefficient (*standard*) and the new algorithm which uses the convolution formula (*two-stage*). Table 1 shows a comparison of rendering times for both variants. For the Haar wavelet, two-stage splatting is about a factor of two faster, and for the linear B-spline wavelet more than a factor of three. Obviously, blending the footprint into the frame buffer for each coefficient has a severe impact on the performance. Two-stage splatting is also less dependent on the footprint size, which explains the higher speedup for the linear B-spline wavelet.

	Haar wavelet			*Linear B-spline wavelet*		
	Phantom (610219)	CT head (1323304)	MR head (10180887)	Phantom (1696748)	CT head (2023423)	MR head (12950884)
standard	1.51	3.15	26.58	6.76	7.79	58.75
two-stage	0.83	1.58	12.88	2.10	2.45	16.23

Table 1: *Comparison of rendering time (seconds) of the full data set for standard splatting and two-stage splatting. For the phantom and CT data, the decomposition depth was set to two, and for the MR data to three. Indicated in brackets are the number of coefficients left after removing all zero coefficients. The rendering times are averages over 100 views with angles uniformly spread over a range of 360 degrees.*

One of the reasons to use wavelets is that the data can be visualized at different levels of detail. While the user interacts with the data, only the approximation coefficients are used. We simulated this user interaction by taking 100 views with angles uniformly spread over a range of 360 degrees, and measured the frame rates of the two splatting variants. The timings include computation of the footprints, projection of the coefficients, and display of the image on the screen. The results are shown in Table 2, and show about the same speedups as in Table 1. The large drop in the frame rate for the MR head data set is due to the extra time spent to sample the footprints. When the decomposition depth goes from 2 to 3, the linear support of the footprints increases by a factor of two. Consequently, the size of the slice plane in the Fourier domain increases by a factor of four.

Figure 1 shows approximations at different levels of detail for the CT data set, with a linear B-spline as the basic wavelet. These images show that a level-one approximation is hardly distinguishable from the full reconstruction, providing an extra motivation for the use of wavelets.

	Haar wavelet			*Linear B-spline wavelet*		
	Phantom (11296)	CT head (21311)	MR head (21197)	Phantom (32767)	CT head (32767)	MR head (32767)
standard	11.0	6.7	1.9	3.0	3.0	1.0
two-stage	21.0	16.9	4.5	10.7	10.7	3.0

Table 2: *Comparison of number of frames per second between standard splatting and two-stage splatting. Indicated in brackets are the number of approximation coefficients. Only these coefficients were used to compute the frame rates.*

(a) 32767 (b) 262136 (c) 2023408

Figure 1: *Successive refinements for the CT head data set, with a linear B-spline as the basic wavelet. Indicated below each image is the number of coefficients used to generate the image. (a) level-2 approximation. (b) level-1 approximation. (c) full reconstruction. Differences between (b) and (c) are hardly visible.*

5 Conclusion

We have shown that, in the case of X-ray rendering, the splatting process can be split in two stages: (i) coefficient accumulation and (ii) convolution with the footprint. The results show that the rendering time of two-stage splatting is reduced by a factor of two to more than three in comparison to the variant in which the footprint is blended into the frame buffer for each coefficient. Two-stage splatting also proves to be less dependent on the size of the footprint. This is important for wavelet splatting for two reasons: (i) the support of the footprints doubles when the decomposition depth is increased with one level, and (ii) wavelets with large support allow the refinement step to terminate before a full reconstruction is reached. For instance, the linear B-spline wavelet shows good results using only 13% of the non-zero coefficients.

References

[1] R.A. DeVore, B. Jawerth, and B.J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38(2):719–746, 1992.

[2] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, New York, 1988.

[3] L. Lippert and M.H. Gross. Fast wavelet based volume rendering by accumulation of transparent texture maps. *Computer Graphics Forum*, 14(3):431–443, 1995.

[4] F. Natterer. *The Mathematics of Computerized Tomography*. B.G. Teubner and J. Wiley, 1986.

[5] J. Wang and H.K. Huang. Medical image compression by using three-dimensional wavelet transformation. *IEEE Transactions on Medical Imaging*, 15(4):547–554, 1996.

[6] Michel A. Westenberg and Jos B. T. M. Roerdink. Frequency domain volume rendering by the wavelet X-ray transform. *IEEE Trans. Image Process.*, 9(7):1249–1261, July 2000.

[7] L.A. Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, 1990.