An Anti-Aliasing Technique for Splatting

J. Edward Swan II[1,2,3] **Klaus Mueller**[2] **Torsten Möller**[1,2] **Naeem Shareef**[1,2] **Roger Crawfis**[1,2] **Roni Yagel**[1,2]

[1]Advanced Computing Center for the Arts and Design
[2]Department of Computer and Information Science
The Ohio State University*

[3]The Naval Research Laboratory
Washington, DC†

ABSTRACT

Splatting is a popular direct volume rendering algorithm. However, the algorithm does not correctly render cases where the volume sampling rate is higher than the image sampling rate (e.g. more than one voxel maps into a pixel). This situation arises with orthographic projections of high-resolution volumes, as well as with perspective projections of volumes of any resolution. The result is potentially severe spatial and temporal aliasing artifacts. Some volume ray-casting algorithms avoid these artifacts by employing reconstruction kernels which vary in width as the rays diverge. Unlike ray-casting algorithms, existing splatting algorithms do not have an equivalent mechanism for avoiding these artifacts. In this paper we propose such a mechanism, which delivers high-quality splatted images and has the potential for a very efficient hardware implementation.

Keywords and Phrases: volume rendering, splatting, direct volume rendering, resampling, reconstruction, anti-aliasing, perspective projection.

1. INTRODUCTION

In the past several years, direct volume rendering has emerged as an important technology in the fields of computer graphics and scientific visualization, and splatting is one of several popular techniques for direct volume rendering. The majority of images produced through direct volume rendering have used orthographic projections, in part because such projections are useful in many of the application areas (such as biomedical and fluid flow visualization) which have initially motivated work in volume rendering. Perspective projections offer a viewpoint which more naturally correlates to the way we perceive the physical world, and perspective projections are essential when it is desirable to "fly through" the data — flight simulators are one example. A perspective projection of a volume dataset gives another perceptual cue which can be employed when comprehending spatial relationships.

Any volume rendering algorithm which supports perspective projections has to deal with the problem of non-uniform sampling produced by diverging viewing rays. If not addressed this can result in potentially severe aliasing artifacts. Although other volume rendering approaches have dealt with this problem (e.g. ray-casting [15] and shear-warp [5][6]), to date the problem has not been addressed in the context of splatting. In this paper, we present a modification to the splatting algorithm which prevents the aliasing that arises from this non-uniform sampling. The same type of resampling problems occur with an orthographic projection if the volume resolution is higher than the image resolution (e.g. if many voxels project into each pixel). Our modified splatting algorithm also avoids aliasing in this situation.

*{mueller, moeller, shareef, crawfis, yagel}@cis.ohio-state.edu
†swan@acm.org

In the next section we describe the splatting algorithm and related previous work, and then we give some advantages and disadvantages of splatting as compared to other volume rendering techniques. In Section 4 we describe our anti-aliasing technique and argue for its correctness. We follow this with implementation details and example images. In Section 6 we discuss our findings and indicate areas of future work.

2. PREVIOUS WORK

The splatting technique has been used to directly render volumes of various grid structures [11][21] and for both scalar [8][21][22] [23] and vector fields [4]. The basic algorithm, first described by Westover [21], projects each voxel to the screen and composites it into an accumulating image. It solves the hidden surface problem by using a painter's algorithm: it visits the voxels in either a back-to-front or front-to-back order, with closer voxels overwriting farther voxels. Splatting is an *object-order* algorithm: the resulting image is built up voxel-by-voxel. This is in contrast to volume rendering by *ray-casting*, which is an image-order algorithm that builds up the resulting image pixel by pixel.

As each voxel is projected onto the image plane, the voxel's energy is spread over the image raster using a reconstruction kernel centered at the voxel's projection point. This reconstruction kernel is called a "splat"; its name comes from the colorful analogy of throwing a snowball against a wall, with the spreading energy analogous to the "splatting" snow. Conceptually, the splat is considered a spherically symmetric 3D reconstruction kernel centered at a voxel. But because the splat is reconstructed into a 2D image raster, it can be implemented as a 2D reconstruction kernel. This 2D kernel, called a "footprint function", contains the integration of the 3D kernel along one dimension. Because the 3D kernel is spherically symmetric, it does not matter along which axis this integration is performed. The integration is usually pre-computed, and the footprint function is represented as a finely sampled 2D lookup table. The 2D table is centered at the projection point and sampled by the pixels which lie within its extent. Each pixel composites the value it already contains with the new value from the footprint table. Under certain conditions (regular volume grid spacing, orthographic view projection, radially symmetric splat kernel) the footprint table can be computed once and used unmodified for all voxels. Under different conditions, the footprint function will vary, and consequently must be re-computed for each view (when there is a non-symmetric kernel) and possibly for each voxel (when there is a perspective projection).

Recent work has extended the original splatting algorithm to achieve higher quality as well as faster rendering. To improve image quality, in later work Westover [22][23] first accumulates splats onto a 2D sheet that is aligned with the volume axis most parallel to the view plane, and then composites the sheets in depth order into the image with a matting operation. Image quality is also affected by the size, shape, and type of the reconstruction kernel used. Laur and Hanrahan [8] change the size of a splat based upon the cell it represents in an octree representation of the volume. Mao [11] uses spherical and ellipsoidal kernels with varying sizes

to splat non-rectilinear grids. Mueller and Yagel [14] use an image-order splatting approach which improves accuracy when using a perspective projection. And while to date most splatting implementations have used a Gaussian reconstruction kernel, other kernel types can generate higher quality images. Max [12] and Crawfis and Max [4] propose quadratic spline functions, optimized for certain conditions, as splat kernels.

To improve rendering speed, Westover [22] maps view dependent footprints with a circular or elliptical shape to a generic footprint table which only needs to be computed once. Laur and Hanrahan [8] approximate splats with a triangle mesh and use graphics hardware to quickly scan convert the footprint. Crawfis and Max [4] and Yagel et al. [26] also use texture mapping hardware to quickly render splats represented as textures mapped to polygons. Splatting can also be accelerated by preprocessing the volume and culling voxels which will not contribute to the final image. Laur and Hanrahan [8] cull with an octree structure, and Yagel et al. [26] extract and store only the most visually significant voxels.

3. ADVANTAGES AND DISADVANTAGES OF SPLATTING

In this section we compare splatting to other rendering algorithms. When listing the disadvantages of splatting, we distinguish between inherent problems and those that are due to inaccuracies in current splatting implementations.

3.1 Advantages of Splatting

The main advantage of splatting over ray-casting is that splatting is inherently faster. In ray-casting, reconstruction is performed for each sample point along the ray. At each sample point a k^3 convolution filter is applied. Even if, on the average, each of the n^3 voxels are sampled only once, ray-casting has a complexity of *at least* k^3n^3. In splatting, on the other hand, the convolution is precomputed, and every voxel is splatted exactly once. Each splat requires k^2 compositing operations. Therefore, one can expect a complexity of *at most* k^2n^3. This gives splatting an inherent speed advantage. An additional benefit is that one can afford to employ larger reconstruction kernels and improve the accuracy of splatting, incurring an $O(k^2)$ penalty instead of an $O(k^3)$ penalty.

Because splatting is an object-order rendering algorithm, it has a simple, static parallel decomposition [9][23], where the volume raster is evenly divided among the processors. It is more difficult to distribute the data with ray-driven approaches, because each ray might need to access many different parts of the volume raster.

Splatting is trivially accelerated by ignoring empty voxels. It can be accelerated further by extracting and storing just those voxels which contribute to the final image [26], which prevents traversing the entire volume raster. This is equivalent to similar acceleration techniques for volume ray-casting, such as space-leaping [25] or fitted extents [19], which accelerate ray-casting by quickly traversing empty space.

Because splatting generates images in a strict front-to-back or back-to-front order, observing the partially created images can give insight into the data which is not available from image-order techniques. In particular, with a back-to-front ordering, partial images reveal interior structures, while with a front-to-back ordering it is possible to terminate the rendering early [14].

Finally, splatting is the preferred volume rendering technique when the desired result is an X-ray projection image instead of the usual composited image [14]. This is because the summation of pre-integrated reconstruction kernels is both faster and more accurate than ray-casting approaches, which require the summation of many discrete sums. Creating X-ray projection images from volumes is an important step in the reconstruction algorithms employed by tomographic medical imaging devices [14][3], such as CT and PET.

3.2 Inherent Disadvantages of Splatting

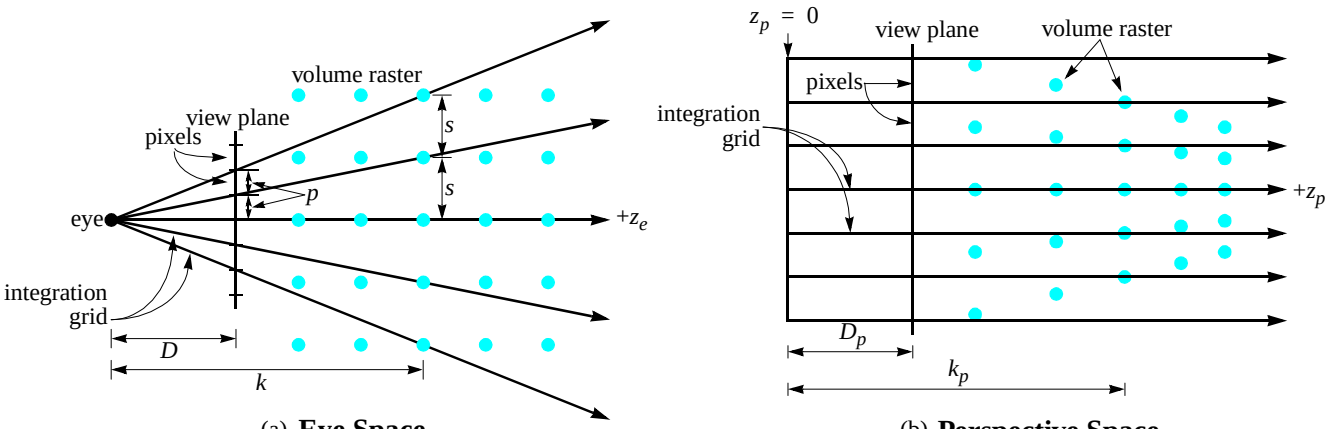
There are some disadvantages inherent to the splatting method. One is that while an ideal volume renderer first performs the process of reconstruction and then the process of integration (or composition) for the entire volume, splatting forces both reconstruction and integration to be performed on a per-splat basis. The result is incorrect where the splats overlap, and the splats must overlap to ensure a smooth image. This problem is particularly noticeable when the traversal order of the volume raster changes during an animation [22].

Another disadvantage of splatting lies in the ordering of the classification, shading, and reconstruction steps. For efficiency reasons, in splatting both (transfer function-based) classification and shading are usually applied to the data *prior* to reconstruction. This is also commonly done in ray-casting [7]. However, this produces correct results only if both classification and shading are linear operators. The result of employing a non-linear classification or illumination model may cause the appearance of pseudo-features that do not exist in the original data. For example, if we want to find the color and opacity at the center point between the two data values a and b using linear interpolation, we would compute $(C(a) + C(b))/2$ performing classification first, but the correct value would be $C((a + b)/2)$ performing interpolation first. Clearly, if C is a non-linear operator, these two results will be different. Requiring C to be linear generally means that the shading model can only model diffuse illumination. While methods exist for ray-casting that perform classification and shading *after* reconstruction [2][13][17], this is not possible in splatting.

3.3 Implementation-Based Disadvantages of Splatting

With a ray-casting volume rendering algorithm it is easy to terminate the rays early when using a front-to-back compositing scheme, which can substantially accelerate rendering. Although not reported in the literature, early termination could potentially be implemented for splatting by employing the dynamic screen mechanism [18] (also used by [6] for shear-warp volume rendering). Also, the ray-driven splatting [14] implementation can support early ray termination.

While ray-casting of volumes was originally implemented for both orthographic and perspective viewing, splatting was fully implemented only for orthographic viewing. Although ray-casting has to include some mechanism to deal with the non-uniform reconstruction that is necessary with diverging viewing rays, it seems splatting needs to address several more inaccuracies. For the following discussion, it is useful to adopt the definitions given in [4], [14], and [26] which view the footprint table as a polygon in world space centered at the voxel position with the pre-integrated filter kernel function texture-mapped onto it. As is described in [14], when mapping the footprint polygon onto the screen an accurate perspective splatting implementation must: (1) align the footprint polygon perpendicularly with the projector (sight ray) that goes through the polygon center; (2) perspectively project it to the screen to get its screen extent, and (3) ensure that the projector (sight ray) for every pixel that falls within this extent traverses the polygon at a perpendicular angle as well. All three conditions are violated in Westover's splatting algorithm [21]. Mueller and Yagel [14] give a voxel-driven splatting approach that takes care of con-



(a) **Eye Space** (b) **Perspective Space**

FIGURE 1. Resampling the volume raster onto the integration grid. (a) In eye space. (b) In perspective space.

dition (1) and (2), and a ray-driven approach that fulfills all three conditions.

4. AN ANTI-ALIASING TECHNIQUE FOR SPLATTING

In this section we describe our splatting-based anti-aliasing method and argue for its correctness. In Section 4.1 we describe why anti-aliasing is needed for volume rendering algorithms. In Section 4.2 we develop an expression (Equation 2) which, if satisfied by a given volume rendering algorithm, indicates that the algorithm will not produce the sample-rate aliasing artifacts that arise from the resampling phase of the rendering process. In Section 4.3 we describe our anti-aliasing method, and in Section 4.4 we show that our method satisfies the equation developed in Section 4.2, which argues for the correctness of the method. Section 4.5 describes the anti-aliasing method's frequency domain characteristics and discusses the effects of using a non-ideal reconstruction kernel. Finally, Section 4.6 analyzes the error that results from the way the technique estimates the local sampling rate.

4.1 The Need for Anti-Aliasing in Volume Rendering

The process of volume rendering is based on the integration (or composition), along an *integration grid*, of the volume raster. This integration grid is composed of *sight projectors* (or *rays*) which pass from the eye point, through the view plane, and into the volume raster. Before this integration can occur, the volume raster has to be reconstructed and then resampled along the integration grid. This is illustrated in Figure 1 for a perspective view of the volume, where the volume raster is shown as a lattice of dots, and the integration grid is shown as a series of rays, cast through pixels, which traverse the volume raster. Figure 1a shows the scene in eye space, where the eye is located at point $(0, 0, 0)$ and is looking down the positive z-axis (denoted $+z_e$). The perspective transformation means the integration grid diverges as it traverses the volume. Figure 1b shows the same scene in perspective space, after perspective transformation and perspective division. Here the volume raster is distorted according to the perspective transformation, and the integration grid lines are parallel. Because of this the eye is no longer located at a point, but can be considered the plane $z_p = 0$.

The reconstruction and resampling of the volume raster onto the integration grid has to be done properly to avoid aliasing artifacts. Ideally, aliasing is avoided by (1) sampling above the Nyquist limit, and (2) reconstructing with an ideal filter. The aliasing that results from an insufficient sampling rate (below the Nyquist limit) is called *prealiasing* — it is caused by energy from the alias spectra spilling over into the primary spectrum. The aliasing that results from a non-ideal reconstruction filter is called *postaliasing* — it is caused by the non-ideal filter picking up energy from the alias spectra beyond the Nyquist limit (see Figure 5 and [16][10][24]). In practice, it is not possible to implement an ideal reconstruction filter, and so criteria (2) cannot be achieved — any realizable filter inevitably results in a tradeoff among aliasing, blurring, and other artifacts [10]. However, reconstruction filters previously used for splatting, in particular Gaussian filters, contribute very little postaliasing at the cost of substantial blurring [10]. In current splatting implementations, the great majority of aliasing is prealiasing; it arises from not achieving criteria (1). Therefore, in the rest of this paper, when referring to the term 'aliasing' we generally mean prealiasing, or aliasing that results from not sampling above the Nyquist limit.

It may be possible to sample above the Nyquist limit, but if this is not possible then aliasing can also be avoided by low-pass filtering the volume to reduce its frequency content. For an orthographic view this low-pass filtering must be applied to the entire volume, but for a perspective view low-pass filtering may only be required for a portion of the volume. This can most easily be seen in perspective space (Figure 1b). Note that there is a distance along the $+z_p$ axis, denoted k_p, where the sampling rate of the volume raster and the sampling rate of the integration grid are the same. Previous to this distance there is less than one voxel per pixel, and beyond this distance there is more than one voxel per pixel. When there is more than one voxel per pixel, the volume raster can contain frequency information which is higher than the integration grid can represent, and so aliasing is possible. In the next section this concept is developed into an equation.

Note that same concept holds in eye space (Figure 1a). Here there is an equivalent distance along the $+z_e$ axis, denoted k, where the sampling rates of the volume raster and the integration grid are the same (note that in general $k \neq k_p$, but the two distances are related by the perspective transformation). Aliasing artifacts can occur beyond k, when the distance between adjacent rays is greater than one voxel.

Volume ray-casting algorithms generally perform the reconstruction in eye space. Some avoid aliasing by employing recon-

struction kernels which become larger as the rays diverge [15][17]. This provides an amount of low-pass filtering which is proportional to the distance between the rays. Splatting algorithms generally perform the reconstruction in perspective space. Unlike ray-casting algorithms, existing splatting algorithms do not have an equivalent mechanism to avoid aliasing. In this paper we propose such a mechanism.

4.2 Necessary Conditions to Avoid Aliasing

In this section we give the conditions which are necessary for the volume rendering resampling process to avoid introducing sample-rate aliasing artifacts into the integration grid samples. Let s be the volume raster grid spacing (Figure 1a), and $\rho = 1/s$ be the volume raster sampling rate. The volume raster contains aliasing when either (1) the sampled function is not bandlimited, or (2) the function is bandlimited at frequency w but the sampling rate ρ is below the Nyquist limit: $\rho < 2w$.

If the first condition holds then aliasing will be present no matter how large ρ becomes. However, if the function is bandlimited at w, then as long as

$$\rho \geq 2w \tag{1}$$

there is no aliasing in the volume raster. Assuming that Equation 1 is true, our job is to resample the volume raster onto the integration grid in a manner that guarantees that no aliasing is introduced.

Let ϕ represent the sampling frequency of the integration grid. For a perspective projection the integration grid diverges (Figure 1a) and therefore ϕ is a function of distance along the z_e axis: $\phi = \phi(d)$, where d is the distance. For an orthographic projection we can still express $\phi(d)$ as a function, but it will have a constant value. As illustrated in Figure 1, at k the sampling rates of the volume raster and the integration grid are the same: $\phi(k) = \rho$.

The distance k means there are two cases to consider:

- **Case 1**: $d < k$. This is the case for the portion of the grid in Figure 1a previous to k. Here $\phi(d) > \rho$, and if Equation 1 holds then $\phi(d) > 2w$ and there is no aliasing.
- **Case 2**: $d \geq k$. This is the case for the portion of the grid in Figure 1a beyond k. Here $\phi(d) \leq \rho$, and so it may be that $\phi(d) < 2w$. If this is the case, then the integration grid will contain an aliased signal once d is large enough.

This argument shows that, given Equation 1, volume rendering algorithms do not have to perform anti-aliasing as long as Case 1 holds (previous to the distance k). However, once Case 2 holds (beyond k), it is necessary to low-pass filter the volume raster to avoid aliasing. Ideally the amount of this filtering is a function of d*, and reduces the highest frequency in the volume raster from w to $\tilde{w}(d)$. To avoid aliasing there must be enough low-pass filtering so that

$$\phi(d) \geq 2\tilde{w}(d). \tag{2}$$

By showing that this equation holds for a particular volume rendering technique, we can claim that the technique does not introduce sample-rate aliasing artifacts when resampling from the volume raster onto the integration grid.

*This is because a portion of the volume raster may not require any filtering (Case 1), and for the portion that does require filtering (Case 2), if there is a perspective projection then the amount of filtering required is itself a function of d.

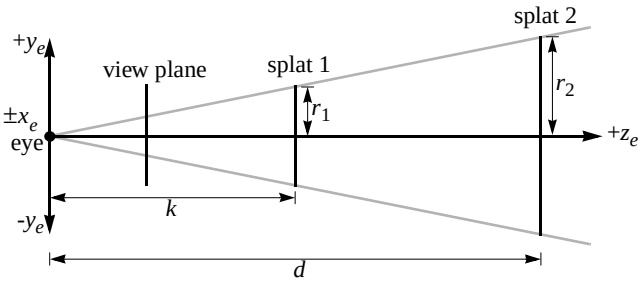


FIGURE 2. The geometry for attenuating the energy of splat 2.

4.3 An Anti-Aliasing Method for Splatting

As mentioned in Section 4.1 above, volume ray-casting algorithms avoid aliasing by using reconstruction kernels which increase in size as the integration grid rays diverge, which satisfies Equation 2. In this section we give a similar anti-aliasing algorithm for splatting.

As shown in Figure 1a, at distance k the ratio of the volume raster sampling frequency ρ and the integration grid sampling frequency $\phi(d)$ is one-to-one. k can be calculated from similar triangles:

$$k = s\frac{D}{p}, \tag{3}$$

where s is the sample spacing of the volume raster, p is the extent of a pixel, and D is the distance from the eye point to the screen.

Figure 3 gives a "side view" of splatting as implemented by Westover [21][22][23], as well as our anti-aliasing method. In Figure 3 the y-axis is drawn vertically, the z-axis is drawn horizontally, the x-axis comes out of and goes into the page, and diagrams are shown in both eye space (x_e, y_e, z_e) and perspective space (x_p, y_p, z_p). The top row illustrates standard splatting. As in Figure 1, D is the distance of the view plane from the eye point, and k is calculated from Equation 3. For this example we are rendering a single row of splats, which are equally spaced along the z_e axis (Figure 3a). Each splat is the same size in eye space. Figure 3b shows the same scene in perspective space. Here D_p and k_p are D and k expressed in z_p coordinates. As expected, because of the non-linear perspective transformation, the splat spacing is now non-uniform along the z_p axis, and the size of the splats decreases with increasing distance from the eye.

The bottom row illustrates our anti-aliasing method. Previous to k we draw splats the same size in eye space (Figure 3c). Beginning at k, we scale the splats so they become larger with increasing distance from the view plane. This scaling is proportional to the viewing frustum, and is given in Equation 4 below. Figure 3d shows what happens in perspective space. Previous to k_p we draw the splats with decreasing sizes according to the perspective transformation. Beginning at k_p, we draw all splats the same size, so splats with a z_p coordinate greater than k_p are the same size as splats with a z_p coordinate equal to k_p.

Figure 2 gives the geometry for scaling splats drawn beyond k. If a splat drawn at distance k has the radius r_1, then the radius r_2 of a splat drawn at distance $d > k$ is the projection of r_1 along the viewing frustum. This is calculated by similar triangles:

$$r_2 = r_1\frac{d}{k}. \tag{4}$$

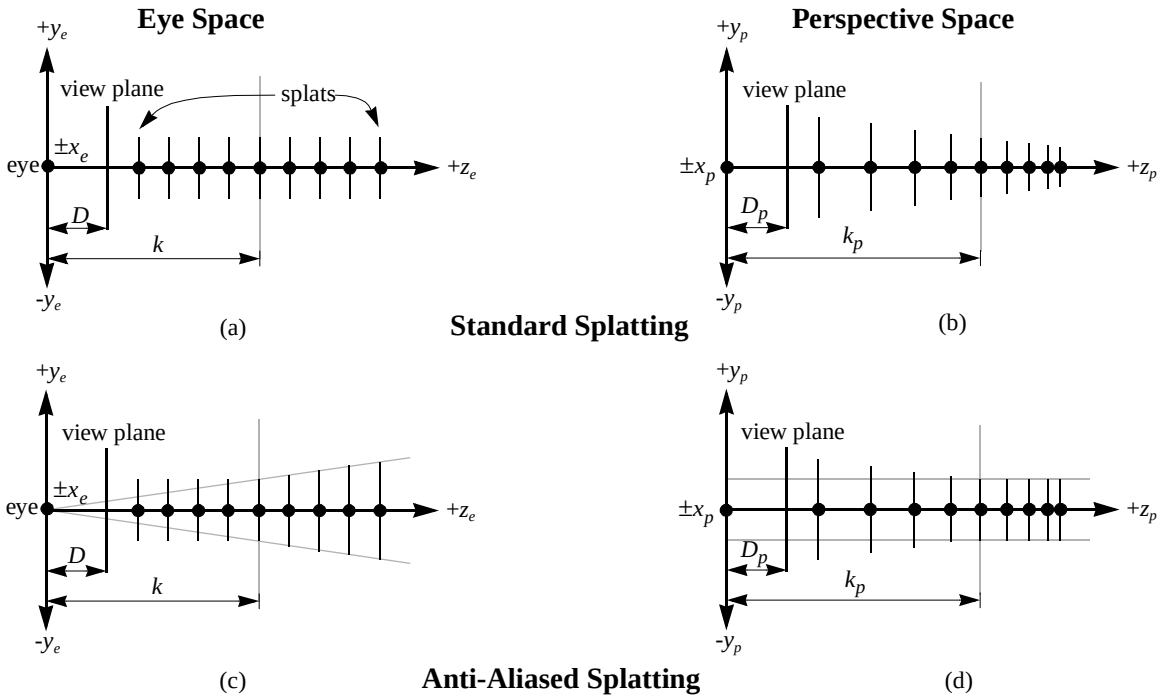


FIGURE 3. A comparison of the standard splatting method (top) with our anti-aliased method (bottom). (a) The standard splatting method in eye space. (b) The standard splatting method in perspective space. (c) The anti-aliased splatting method in eye space. (d) The anti-aliased splatting method in perspective space.

Scaling the splats drawn beyond k is not enough to provide anti-aliasing, however. In addition to scaling, the energy that these splats contribute to the image needs to be the same as it would have been if they had not been scaled. As shown in Figure 2, both splat 1 and splat 2 project to the same sized area on the view plane. Because they are composited into the view plane in the form of two-dimensional "footprint" filter kernels [22], the amount of energy the splats contribute to the view plane is proportional to their areas. We want the amount of energy per unit area contributed by the splats to be the same. We accomplish this by attenuating the energy of splat 2 according to the ratio of the areas of the splats:

$$E_2 = \frac{A_1}{A_2} E_1, \tag{5}$$

where A_1, A_2 are the areas of the splats and E_1, E_2 are some energy measure for the splats. Examples of energy measures include the volume under the splat kernel or the alpha channel of the polygon defining the 2D splat footprint.

Assume for now that the filter kernel is a circle for both splats. Then the areas of the splats are $A_1 = \pi r_1^2$ and $A_2 = \pi r_2^2$. By Equation 4 we can express A_2 in terms of r_1:

$$A_2 = \pi\left(r_1 \frac{d}{k}\right)^2. \tag{6}$$

Then

$$E_2 = \left(\frac{\pi r_1^2}{\pi\left(r_1 \frac{d}{k}\right)^2}\right) E_1 \tag{7}$$

which can be simplified to

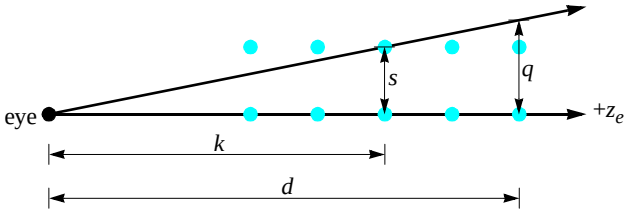


FIGURE 4. Calculating the integration grid sampling frequency.

$$E_2 = \left(\frac{k}{d}\right)^2 E_1. \tag{8}$$

Although here we have derived Equation 8 using circular filter kernels, we can also use any other two-dimensional shape for the kernel, such as an ellipse, square, rectangle, parallelogram, etc. and derive the same equation. We use Equation 8 to attenuate the energy of all splats drawn beyond k.

4.4 Correctness of the Method

We now demonstrate that Equation 2 holds for our anti-aliasing technique. We begin by deriving expressions for the two functions in Equation 2 — $\phi(d)$ (the integration grid sampling frequency at d) and $\tilde{w}(d)$ (the maximum volume raster frequency at d).

We derive the integration grid sampling frequency $\phi(d)$ with a similar-triangles argument. Consider Figure 4, where q is the integration grid spacing at distance d from the eyepoint. By similar triangles

$$\frac{s}{k} = \frac{q}{d}, \tag{9}$$

which can be written as

$$\frac{1}{q} = \frac{1}{s} \cdot \frac{k}{d}. \tag{10}$$

Now $1/s = \rho$, and $1/q$ is simply $\phi(d)$, the integration grid sampling frequency at d. Thus we have

$$\phi(d) = \rho \cdot \frac{k}{d}. \tag{11}$$

The maximum volume raster frequency $\tilde{w}(d)$ can be derived from the scaling property of the Fourier transform [1]:

$$f(at) \leftrightarrow \left|\frac{1}{a}\right| F\left(\frac{1}{a}\omega\right), \tag{12}$$

where "$\leftrightarrow$" indicates a Fourier transform pair. This shows that widening a function by the factor a in the spatial domain is equivalent to narrowing the function in the frequency domain by the factor $1/a$. In our anti-aliasing method, the widening for the splat kernels drawn beyond k is given by Equation 4:

$$a = \frac{d}{k}. \tag{13}$$

Thus we have

$$f\left(\frac{d}{k}t\right) \leftrightarrow \left|\frac{k}{d}\right| F\left(\frac{k}{d}\omega\right), \tag{14}$$

which shows that as the splat kernels are widened by d/k, the frequency components of the function they reconstruct are narrowed by k/d. Since all the frequencies of the volume raster are attenuated by k/d, the maximum frequency w is attenuated by the same amount, and we have:

$$\tilde{w}(d) = w \cdot \frac{k}{d}. \tag{15}$$

Now we are ready to show that our technique satisfies Equation 2. We start with Equation 1: $\rho \geq 2w$, which implies that the volume raster has sampled the function above the Nyquist limit. Multiplying both sides by k/d we have

$$\rho \cdot \frac{k}{d} \geq 2\left(w \cdot \frac{k}{d}\right), \tag{16}$$

which we can write as:

$$\phi(d) \geq 2\tilde{w}(d). \tag{17}$$

This derivation says that if the volume raster has sampled the function above the Nyquist limit, our anti-aliasing technique provides enough low-pass filtering so that aliasing is not introduced when the volume raster is resampled onto the integration grid. Note that this derivation only deals with the prealiasing that results from an inadequate sampling rate — it does not address the aliasing or blurring effects which result from using a non-ideal reconstruction filter.

4.5 Frequency Domain Behavior

This section describes the frequency-domain behavior of the anti-aliasing method, and it illustrates the effects of reconstructing with a non-ideal reconstruction kernel. Figure 5 compares the frequency-domain behavior of the standard splatting technique with the new anti-aliasing technique. For clarity, the diagrams are all drawn in one dimension; the extension to three dimensions is straightforward.

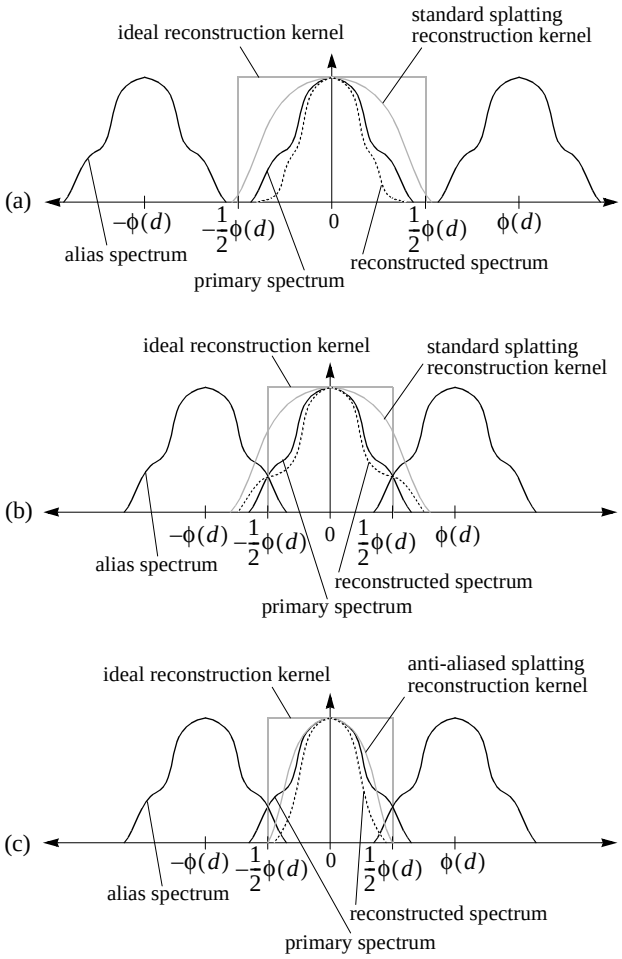


FIGURE 5. The behavior of the anti-aliased method in the frequency domain. (a) Previous to k ($d < k$). (b) Beyond k ($d \geq k$); standard splatting method. (c) Beyond k ($d \geq k$); anti-aliased splatting method.

Figure 5a shows what happens in the frequency domain when resampling from the volume raster onto the integration grid previous to the distance k (e.g. $d < k$). There is a primary spectrum at the origin of frequency space, and alias spectra replicated at regular intervals of $\phi(d)$, the integration grid frequency for the distance d. Assuming Equation 1 is true, then $\phi(d) > 2w$ and the primary and alias spectra do not overlap, and therefore there is no prealiasing [16][10]. As shown, an ideal reconstruction filter would have the value 1 between $-(1/2)\phi(d)$ and $(1/2)\phi(d)$, and 0 elsewhere, which would yield the primary spectrum. Of course, an ideal filter cannot be implemented, and a standard, realizable splatting reconstruction kernel is also shown. This filter blurs the primary spectrum, yielding the reconstructed spectrum as shown. A realizable filter also typically picks up some energy beyond the Nyquist limit of $|(1/2)\phi(d)|$, and so it is susceptible to postaliasing as well [16][10].

Figure 5b shows what happens in standard splatting beyond the distance k (e.g. $d \geq k$). Here $\phi(d)$ has shrunk, which pulls the alias spectra into the primary spectrum, resulting in prealiasing. An

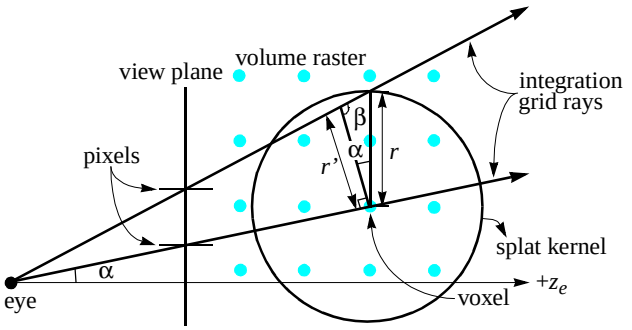


FIGURE 6. Analysis of the integration grid sampling rate error.

ideal reconstruction kernel would shrink as well — thus getting larger in the spatial domain — which would pick up some prealiasing but reject the postaliasing. A standard splatting reconstruction kernel stays the same size, so it picks up both more of the prealiasing and the postaliasing. Figure 5c shows what happens with the anti-aliasing method at the same location — the splatting reconstruction kernel shrinks in the frequency domain according to *d*. This eliminates almost all of the postaliasing and most of the prealiasing, at the expense of blurring.

4.6 Integration Grid Sampling Rate Error

As calculated from Figure 1 and Equation 3, the integration grid sampling rate at position *k* is only exactly correct for splats centered on the $+z_e$ axis. In this section we analyze the amount of error incurred for splats that do not lie along this axis.

The geometry of the calculation is given in Figure 6. For the two rays shown, let *r* be the length, calculated from Equation 4, of the radius of the splat. A more accurate way to approximate the integration grid sampling rate is to measure the "perpendicular" distance between the two rays, given by r' (as shown r' is only perpendicular to the bottom ray; it is not quite perpendicular to the top ray). As calculated from the angle α, the error is given by $r' \approx r\cos\alpha$, where we say "approximately equal" because the angle β is not exactly a right angle. This means that Equation 4 underestimates $\tilde{w}(d)$ by a small amount, and thus voxels which do not lie on the $+z_e$ axis receive a greater amount of low-pass filtering than they actually require. For example, at the view cone boundary with $\alpha = 30°$ Equation 4 chooses a kernel that is about $1.0/0.87 = 1.15$ times larger than it needs to be. It is important to note, however, that because this error results in more low-pass filtering than is actually required, Equation 2 is still satisfied and so this error does not result in aliasing.

5. RESULTS

In our implementation of this algorithm we make use of rendering hardware to quickly draw the splats, in a manner similar to [4], [14], and [26]. For each splat we draw a polygon in world space centered at the voxel position. The polygon is rotated so it is perpendicular to the ray passing from the eye point through the voxel position. The splat kernel is pre-computed and stored in a 256×256 table which is texture mapped onto the polygon by the rendering hardware. We use the optimal cubic spline splat kernel reported in [4]. We attenuate the alpha channel of the polygon as a measure of splat energy when evaluating Equation 8, and then composite the semi-transparent splat polygon into the screen buffer.

Our renderer is a modified version of the "splat renderer" reported in [26]. For a given volume we extract and store a subset of the voxels. For each voxel we evaluate a transfer function $t = F(\nabla, \rho)$, where ∇ and ρ are the gradient and density of the voxel, respectively; we include the voxel in the subset if *t* exceeds a user-defined threshold. We store this volume subset as a 2D array of splat rows, where each row contains only the extracted voxels. Each row is implemented as an array of voxels, but the voxels are not necessarily contiguous, and so we must store each voxel's location and normal vector. In general each row may contain a different numbers of voxels. Despite not storing the empty voxels, we can still traverse this data structure in either a back-to-front or front-to-back order.

Figure 7 (see color plates) shows a collection of images obtained from both a standard splatting algorithm and the anti-aliased algorithm reported in this paper. The images in the left column are rendered without anti-aliasing, while the images in the right column are rendered with anti-aliasing.

Figures 7 (a) and (b) show a $498 \times 398 \times 1$ volume consisting of a single sheet, where alternate 10×10 squares are colored either red or white to create a checkerboard effect. The resulting dataset contains 198K splats which are rendered into a 270×157 image. In Figure 7 (a) a black line is drawn at the distance *k*; beyond this line there is more than one voxel per pixel. As expected, the left-hand image shows strong aliasing effects, but these are smoothed out in the right-hand image.

Figures 7 (c) and (d) show a $512 \times 512 \times 103$ volume containing a terrain dataset acquired from a satellite photograph and a corresponding height field. The resulting dataset contains 386K splats (this is more than the expected 512^2 splats because extra splats are required to fill in the "holes" formed where adjacent splats differ in height). Each column of splats is given the color of the corresponding pixel from the satellite photograph. The dataset is rendered into a 270×175 image. In Figure 7 (c) a black line is drawn at the distance *k*. The left-hand image shows strong aliasing in the upper half of the image (containing about 90% of the data); when animated, these regions show prominent flickering and strobing effects. In the right-hand image these regions have been smoothed out; and although this technique does not address temporal aliasing effects, when animated these regions are free of flickering and strobing effects.

Figures 7 (e) and (f) show a $420 \times 468 \times 62$ volume containing a microtubule study acquired from confocal microscopy. The resulting dataset contains 103K splats which are rendered into a 270×270 image. Unlike the previous images, where the viewpoint is set so that the datasets disappear into the horizon and thus the splats have a great range of sizes (covering several pixels to much less than a single pixel), in this image the entire dataset is visible, and the range of splat sizes is much smaller. However, because the volume has a higher resolution than the image it is still liable to aliasing effects (all of the splats are drawn beyond the distance *k*). This is shown in the left-hand image, which contains jagged artifacts that shimmer when animated. In the right-hand image these effects have been smoothed out; when animated this shimmering effect disappears.

6. FUTURE WORK

An area of future work is modifying our implementation to take better advantage of the rendering hardware. As reported in [26], depending on the machine and the number of extracted voxels, our renderer can give real-time performance. However, our current implementation does not take full advantage of the rendering speeds offered by the graphics hardware. Since the anti-aliased splats defined by Equation 4 project to the same size on the screen, it would be more efficient to draw the splats directly on the screen

instead of drawing the scaled splats in eye space. We are currently looking into hardware-supported *bitblt* operations to provide optimal rendering speeds.

Another area of future work is an effort in the opposite direction. In this project there is a mismatch between the type of rendering algorithm we have written and the available programming tools: the Silicon Graphics rendering hardware we used is designed to accelerate scenes using traditional surface graphics primitives such as polygons and spline surfaces; it does not contain an optimized splat primitive. To this end, new hardware architectures which better support operations which are common in volume rendering are needed. Some possible avenues of exploration are:

- Extended *bitblt*-like operators that can be sub-pixel centered and subsequently composited.
- Hardware support for point rendering using different reconstruction kernels (cubic, Gaussian, etc.) with common footprints (circular, elliptical, etc.). Potentially this offers a far more efficient implementation of splatting than hardware texture mapping.
- A higher resolution alpha channel to allow for the accurate accumulation of very transparent splats.
- Splat primitives with automatic size scaling based on their *z*-depth.

A splat primitive has properties of both simple points (always lying in the projection plane) and texture maps (non-linear intensities across the primitive). We are currently working on scanline algorithms to efficiently render splat primitives, with the goal of future hardware implementations.

The implications of our technique for rendering other types of objects needs to be explored. As shown in Figure 7 (a)–(d), one can employ our technique for traditional texture mapping. Our method obviously provides an accurate solution to the texture sampling problem — a solution which is much more accurate than either mip-maps or summed area tables [20]. We are exploring this point-based approach to viewing and rendering as an alternative to the traditional scanline rendering of discrete objects, and we are exploring its applications for texture mapping, image-base rendering, and volume rendering.

7. ACKNOWLEDGMENTS

We acknowledge Daniel Cohen-Or for the terrain dataset and Noran Instruments for the confocal dataset. We acknowledge The Ohio Visualization Lab at The Ohio Supercomputer Center for computer access.

REFERENCES

[1] Bracewell, R.N., *The Fourier Transform and Its Applications* (2nd edition), McGraw-Hill, 1978.

[2] Bentum, M.J., Lichtenbelt, B.B.A., Malzbender, T., "Frequency Analysis of Gradient Estimators in Volume Rendering", *IEEE Transactions on Visualization and Computer Graphics*, 2(3), September 1996, pp. 242–254.

[3] Cabral, B., Cam, N., and Foran, J., "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware", In proceedings of *1994 Symposium on Volume Visualization* (Washington, DC, October 17–18), IEEE Computer Society Press, 1994, pp. 91–98.

[4] Crawfis, R., and Max, N., "Texture Splats for 3D Scalar and Vector Field Visualization", In proceedings of *Visualization '93* (San Jose, CA, October 25–29), IEEE Computer Society Press, 1993, pp. 261–266.

[5] Drebin, R.A., Carpenter, L., and Hanrahan, P., "Volume Rendering", *Computer Graphics* (Proceedings of SIGGRAPH), 22(4), August 1988, pp. 65–74.

[6] Lacroute, P. and Levoy, M. "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", *Computer Graphics* (Proceedings of SIGGRAPH), July 1994, pp. 451–458.

[7] Levoy, M., "Display of Surfaces from Volume Data", *IEEE Computer Graphics and Applications*, 8(5), May 1988, pp. 29–37.

[8] Laur, D. and Hanrahan, P., "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering," *Computer Graphics* (Proceedings of SIGGRAPH), 25(4), July 1991, pp. 285–288.

[9] Yagel, R. and Machiraju, R., "Data-Parallel Volume Rendering Algorithms", *The Visual Computer*, 11(6), 1995, pp. 319–338.

[10] Marschner, S.R. and Lobb, R.J., "An Evaluation of Reconstruction Filters for Volume Rendering", In proceedings of *Visualization '94* (Washington, DC, October 17–21), IEEE Computer Society Press, 1994, pp. 100–107.

[11] Mao, X., "Splatting of Non Rectilinear Volumes Through Stochastic Resampling," *IEEE Transactions on Visualization and Computer Graphics*, 2(2), 1996, pp. 156–170.

[12] Max, N., "An Optimal Filter for Image Reconstruction," In *Graphics Gem II*, J. Arvo, Editor, Academic Press, pp. 101–104.

[13] Möller, T., Machiraju, R., Mueller, K., and Yagel, R., "Classification and Local Error Estimation of Interpolation and Derivative Filters for Volume Rendering", In proceedings of *1996 Symposium on Volume Visualization* (San Francisco, CA, October 27 – November 1), IEEE Computer Society Press, 1996, pp. 71–78.

[14] Mueller, K. and Yagel, R., "Fast Perspective Volume Rendering with Splatting by Utilizing a Ray-Driven Approach", In proceedings of *Visualization '96* (San Francisco, CA, October 27 – November 1), IEEE Computer Society Press, 1996, pp. 65–72.

[15] Novins, K.L., Sillion, F.X., and Greenberg, D.P., "An Efficient Method for Volume Rendering using Perspective Projection", *Computer Graphics* (Proceedings of San Diego Workshop on Volume Visualization), 24(5), November 1990, pp. 95–102.

[16] Oppenheim, A.V., and Shafer, R.W., *Digital Signal Processing*, Prentice-Hall, 1975.

[17] Pfister, H. and Kaufman, A., "Cube-4 — A Scalable Architecture for Real-Time Volume Rendering", In proceedings of *Visualization '96* (San Francisco, CA, October 27 – November 1), IEEE Computer Society Press, 1996, pp. 47–54.

[18] Reynolds R.A., Gordon D., and Chen L., "A Dynamic Screen Technique for Shaded Graphics Display of Slice-Represented Objects", *Computer Vision, Graphics, and Image Processing*, 38(3), June 1987, pp. 275–298.

[19] Sobierajski L.M. and Avila R.S., "Hardware Acceleration for Volumetric Ray Tracing," In proceedings of *Visualization '95* (Atlanta, GA, October 29 – November 3), IEEE Computer Society Press, 1995, pp. 27–34.

[20] Swan, J. Edward II, *Object-Order Rendering of Discrete Objects*, Ph.D. Dissertation, Department of Computer and Information Science, The Ohio State University, 1997.

[21] Westover, L.A., "Interactive Volume Rendering", In proceedings of *Volume Visualization Workshop* (Chapel Hill, NC, May 18–19), Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1989, pp. 9–16.

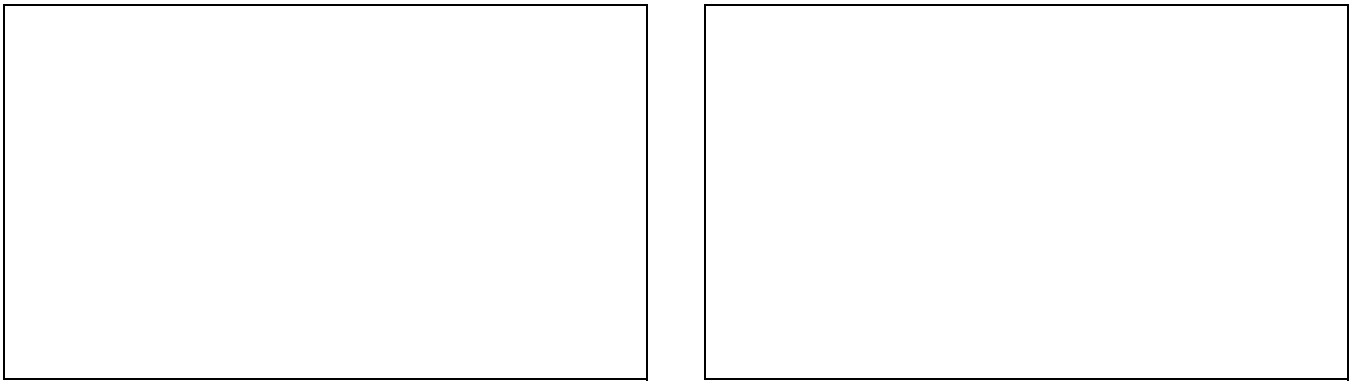
[22] Westover L.A., "Footprint Evaluation for Volume Rendering", *Computer Graphics* (Proceedings of SIGGRAPH), 24(4), August 1990, pp. 367–376.

[23] Westover, L.A., *SPLATTING: A Parallel, Feed-Forward Volume Rendering Algorithm*, Ph.D. Dissertation, Department of Computer Science, The University of North Carolina at Chapel Hill, 1991.

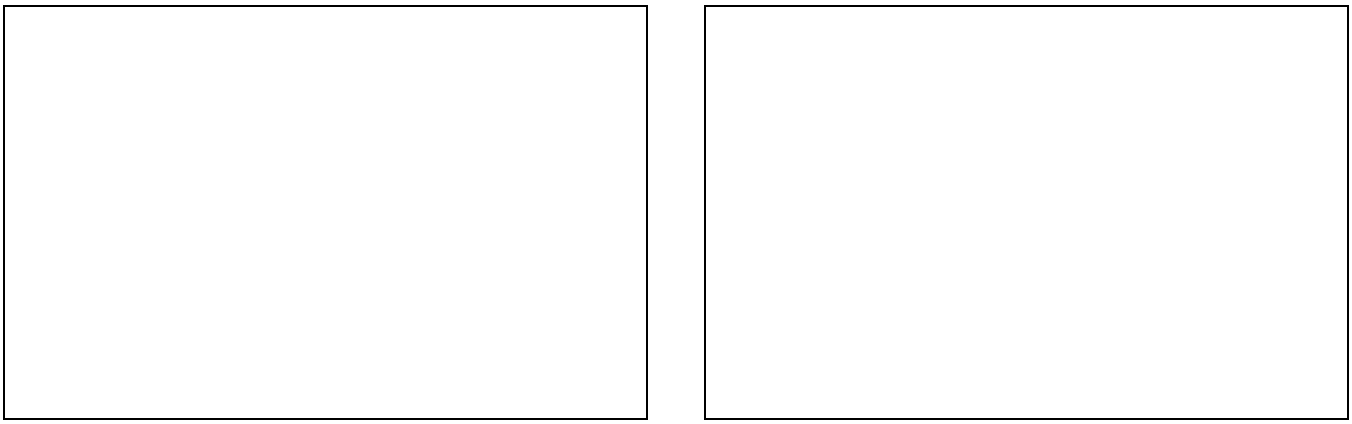
[24] Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, 1989.

[25] Yagel, R., and Shi, Z., "Accelerating Volume Animation by Space-Leaping", In proceedings of *Visualization '93* (San Jose, CA, October 25–29), IEEE Computer Society Press, 1993, pp. 62–69.

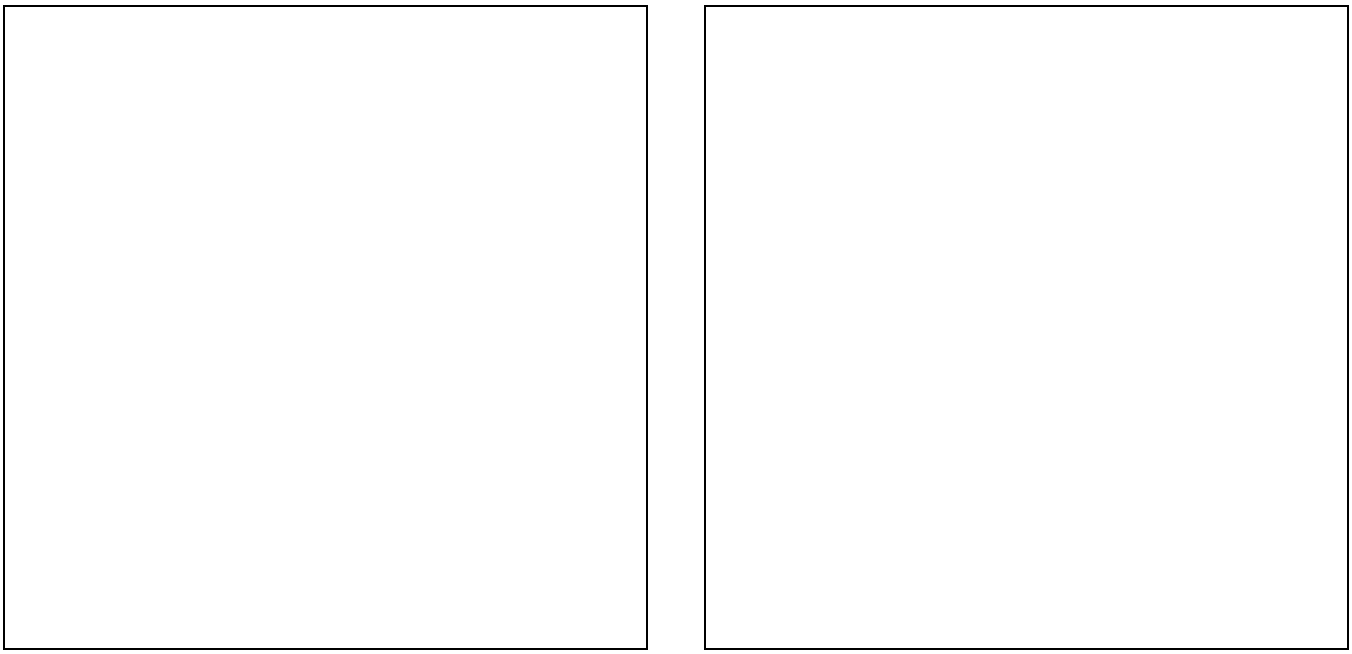
[26] Yagel, R., Ebert, D.S., Scott, J., and Kurzion, Y., "Grouping Volume Renderers for Enhanced Visualization in Computational Fluid Dynamics," *IEEE Transactions on Visualization and Computer Graphics*, 1(2), 1995, pp. 117–132.



(a) (b)



(c) (d)



(e) (f)

FIGURE 7. Rendered images. The left column is rendered with standard splatting, the right column is rendered with the anti-aliased splatting algorithm. (a)–(b) Checkerboard pattern. (c)–(d) Terrain dataset from satellite and mapping data. (e)–(f) Microtubule dataset acquired from confocal microscopy.