Post-Convolved Splatting

Neophytos Neophytou   Klaus Mueller

Center for Visual Computing, Department of Computer Science, Stony Brook University

Abstract

One of the most expensive operations in volume rendering is the interpolation of samples in volume space. The number of samples, in turn, depends on the resolution of the final image. Hence, viewing the volume at high magnification will incur heavy computation. In this paper, we explore an approach that limits the number of samples to the resolution of the volume, independent of the magnification factor, using a cheap post-convolution process on the interpolated samples to generate the missing samples. For X-ray, this post-convolution is needed only once, after the volume is fully projected, while in full volume rendering, the post-convolution must be applied before each shading and compositing step. Using this technique, we are able to achieve speedups of two and more, without compromising rendering quality. We demonstrate our approach using an image-aligned sheet-buffered splatting algorithm, but our conclusions readily generalize to any volume rendering algorithm that advances across the volume in a slice-based fashion.

1. Introduction

Volumetric datasets are routinely produced in a great variety of application domains, with the most prominent being the medical field, both for clinical and educational purposes, and the computational sciences, for fluid flow and other types of simulations. More recent are applications for volumetric design, such as sculpting [1], as well as new data acquisition technologies, such as 3D Doppler radar [4] or cyber scanners. At the same time, we have also seen tremendous growth in the resolution of the devices that are used to visualize the datasets. As the number of available pixels increases (and often the size decreases), users want to view their datasets at higher resolutions. In many cases, this resolution exceeds that of the dataset and magnified viewing occurs. On the other hand, even if the volume is larger than the screen, users may want to zoom into certain volume areas, which again gives rise to a magnification.

Magnified viewing means that the resolution of the image exceeds that of the volume. The implications for this can be grasped by realizing that (i) the frequency content of a volume dataset is bounded from above by its grid sampling rate, and (ii) this upper frequency (the Nyquist frequency) cannot be exceeded, even if we resampled the volume at a higher rate, for the purpose of slicing or magnified volume rendering. In other words, we cannot gain any high-frequency detail in the densities just by oversampling, and if densities is all we're interested in for image generation, which is the case for X-ray, then we may just as well 3D-sample the volume at the volumes's Nyquist frequency and magnify the resulting image just before display. If, however, the interpolated densities are used as an input to another function, then this translation can potentially extend the frequency content of the interpolated signal beyond the volume's upper frequency bound. In fact, the color and opacity transfer functions as well as the shading operations used in full volume rendering represent such a frequency translation. This implies that we will not be able to perform the magnification after a full rendering, but only before the shading and coloring occurs.

An example for an algorithm that does not observe this constraint is the shear-warp algorithm [5]. It gains a fair share of its famous speed by performing all interpolation, shading, and compositing operations at volume resolution and then scaling up the resulting image via an inexpensive 2D warp (see [9] for a closer analysis). This yields a relative insensitivity to magnification in terms of runtime, but it generates blur in places of high geometric and color detail. On the other hand, Westenberg and Roerdink [11] used this idea for their wavelet compression-space X-ray volume rendering approach. Due to their restriction to the X-ray transform, they did not observe significant blurring artifacts.

The magnification of an image is generally performed via a convolution with a suitable filter. Hence, we shall call the technique of obtaining a magnified image from an image that was rendered at volume resolution *post-convolved volume rendering (PCVR)*. We found that PCVR, when used in the context of splatting [12], raises a few interesting peculiarities, which are mainly due to the required use of spherically-symmetric kernels (for reasons of efficiency). These issues are explained in the following sections of this paper, and our solutions for them are discussed as well.

The outline of our paper is as follows: First, Section 2 starts with a discussion of previous work related to PCVR, and then Section 3 provides our theoretical treatment of the subject. Section 4 presents details on our implementation, and Section 5 shows results both on visual quality and the speedups we were able to obtain. Finally, Section 6 ends this paper with conclusions.

2. Motivation and Previous Work

Westenberg and Roerdink [11] used a PCVR scheme to accelerate their hierarchical wavelet X-ray splatting tech-

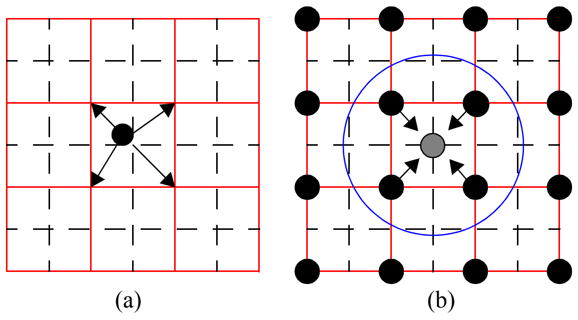


Figure 1: *Two-phase splatting for magnification=2: (a) splatting of a voxel into a (low-resolution) grid at volume resolution, (b) post-convolution of the low-resolution grid samples to give the high resolution grid samples.*

nique. In their work, they first calculate a decomposition of the original density volume into a wavelet basis, which gives rise to a set of wavelet coefficient volumes at decreasing resolution. Then for display, they project each volume separately (after culling irrelevant voxels determined by the wavelet coefficients) into images that match the resolution of the corresponding volume. The projection itself is performed by spreading the projected point into its 2x2 image pixel neighborhood, using bilinear weighting. After all points have been splatted in this way, the images are convolved with a 2D filter corresponding to the projection of the wavelet function (also known as the *footprint*) at that level. This process is illustrated in Fig. 1. Finally, the images are added to yield the final display. Significant speedups can be obtained because the bilinear spreading is much more efficient than a full footprint rasterization for each voxel. This is because bilinear spreading involves only 4 pixels, while the rasterization of the footprint for a kernel of radial extent=2 involves 16 pixels, at magnification=1. For larger magnifications this number scales by the square of the magnification factor. For wavelet-splatting the magnification factor is determined by the level of the wavelet volume and grows in a power of 2. So it is obvious why the PCVR brought great speedups, since the rasterization of large wavelet footprints would take much longer than the bilinear spreading.

Despite the obvious benefits and the innovative nature of Westenberg's and Roerdink's work, there are still a few open issues with this method. First, the authors only use a wavelet decomposition of 2-3 levels, and in their paper they only show the rendering results achieved with the 2-level wavelet decomposition, which is equivalent to a maximum magnification of 2. It is not clear at this point how the approach would behave for larger magnifications, since no formal analysis was given. Second, the approach was only investigated for X-ray rendering and not for full volume rendering with shading and compositing. Third, and finally, the effort for the 2D convolution on the image plane also has to be taken into account, especially if we need to perform it at every sampling slice, in full volume rendering. We would like to have an efficient scheme for this operation as well.

In this paper, we shall address these three issues. First, we will give a formal analysis of PCVR in the frequency domain, illustrated by image examples. Then we will show the application of the method for full volume rendering and describe an efficient method for post-convolution.

3. Theory

Splatting (also known as point-based volume rendering) [12] is an object-order projection algorithm, where each voxel is represented by a 3D basis function (identical for all voxels) and a density value. For image generation, all voxel basis functions b_i are projected to the screen, scaled in value by their corresponding voxel densities v_i, and accumulated in the pixels p_j:

$$p_j = \sum_{i=1}^{N} v_i b_{ij} \tag{1}$$

A projected 3D basis function gives rise to a 2D footprint that is rasterized (i.e., sampled) into the image grid, scaled in value by the voxel's density attribute. Since the footprint is identical for all voxels, it can be pre-computed and stored in a table (called the footprint table), and since the pre-computation is typically not fast, it is most efficient to use a basis function that is spherically symmetric such that it projects the same from all viewing directions.

3.1. Overview

We use the image-aligned sheet-buffered splatting algorithm [6] in our work. In essence, this algorithm interpolates, via splatting of pre-integrated basis-function slices, a series of image-aligned density sheets from the volume, and then shades, colors, and composites them. For X-ray, on the other hand, we can just add the density sheets, or even better, just rasterize and accumulate the fully pre-integrated basis functions themselves. In any case, the basis functions act as reconstruction kernels - they are reconstructing the continuous density function from their gridded voxel samples. Further, the grid on the sheet (slice) or image acts like a sampling process - it samples the reconstructed 3D volume into a 2D raster. Hence, we have a 3D reconstruction process immediately followed by a 3D resampling process, which generates gridded samples on a 2D slice of the 3D space. This is illustrated (for the 2D case) in Fig. 2.

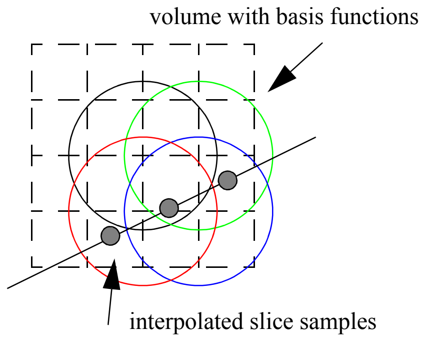


Figure 2: *Interpolation of a slice from volume basis functions.*

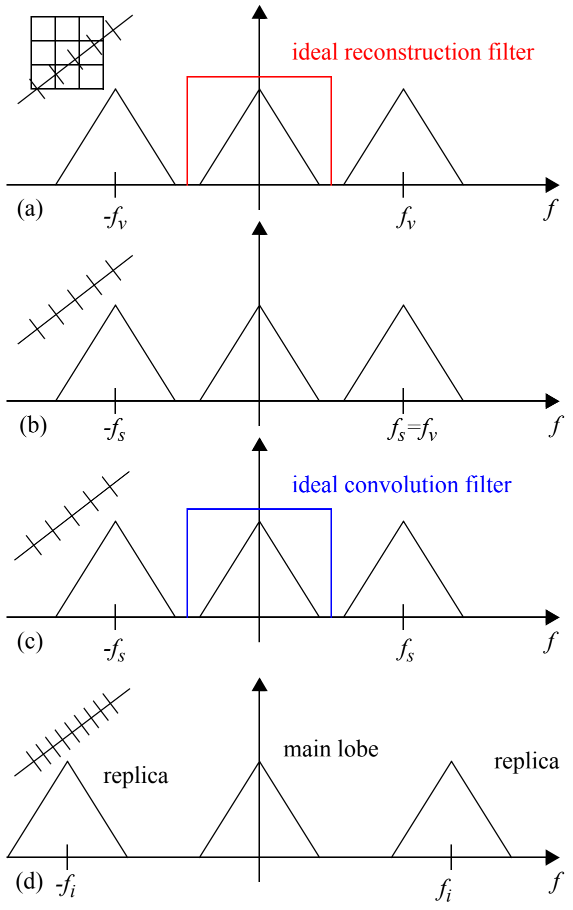


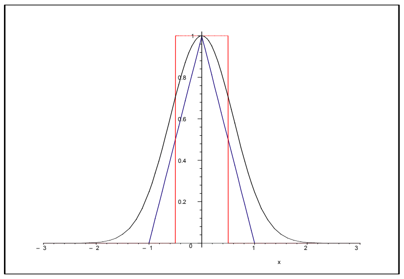
Figure 3: *PCVR in frequency space. Step 1: (a) reconstruction of the volume with bandwidth f_v and (b) sampling it into a slice at equal bandwidth $f_s = f_v$; Step 2: (c) reconstruction of the slice via convolution with a filter with bandwidth f_s and (d) resampling it into the desired high-resolution grid.*

The bandwidth of the 3D reconstruction filter must be set to that of the volume grid. If we don't minify, then the sampling rate of the slice or image is greater or equal that of the volume, and there is no need to lowpass-filter the 2D image. This is shown in Fig. 3. In the same figure, we also show what happens in the PCVR approach, for an ideal filter in the frequency domain (an infeasible sinc filter in the spatial domain). In step 1, we resample the reconstructed signal into a grid of equivalent bandwidth f_v, and in step 2, the post-convolution step, we reconstruct the signal with a filter of identical bandwidth and sample it into a grid of the desired resolution f_i. We see that for the ideal filter, no aliasing is produced and no signal is lost in this two-stage process. However, in practice we are not dealing with ideal filters, and this is where problems may occur. We shall now move into non-ideal scenarios, with non-ideal reconstruction filters, and see what happens there.

3.2. Stage 1: Splatting into a low-res buffer

Earlier we have justified that we need spherically symmetric kernels for splatting to work. We shall now have a look at a few possible practical candidates. Consider first Fig. 4 where we show (for 1D) the nearest-neighbor filter (the box filter), the linear filter, and the Gaussian filter $exp(-1.4 \cdot r^2)$ in both the spatial and the frequency domain. There we see that both the box and the linear filter go to zero at multiples of the sampling rate $f_v = 1.0$, which is where the signal replicas are located (see Fig. 3). So both box and linear filters have good suppression around the (usually) largest portion of the aliases, the signal around the DC term. The Gaussian, on the other hand, suppresses more signal in the passband ($f < 0.5$), but also suppresses better in the stopband ($f > 0.5$). Thus, we expect that the Gaussian will generate somewhat more blurry images, but with less artifacts due to aliasing.

(a)



(b)

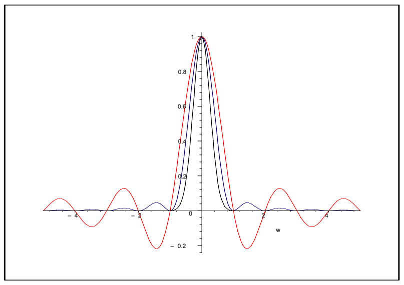


Figure 4: *Box (red), linear (blue), and Gaussian (black) in (a) the spatial and (b) the frequency domain.*

Note that the plots given in Fig. 4 only show the frequency curves of the 1D filters. For the reconstruction during rendering, however, we use 3D filters. Although the spatial domain 3D filter is given by simply plotting the 1D filter curve along the radial lines, we cannot use an equivalent method to get the 3D frequency response curve from the 1D spectrum. Instead we must use the Hankel transform to get the frequency response $H(f)$ of the spatial filter $h(x)$. This is defined as follows:

$$H(f) = 2\pi \int_0^\infty h(x) J_{0.5n-1}(2\pi x f) x^{0.5n} dx \qquad (2)$$

where n is the dimension, in our case $n=2,3$, and $J_{0.5n-1}(x)$ is the $(0.5n-1)$th order Bessel function of the first kind [2].

Fig. 5a shows the radial profiles of the Hankel-transformed radially symmetric box, linear, and Gaussian filters. Fig. 5b shows the 2D frequency plot of the linear filter, in the context of the "replica landscape". We observe from Fig. 5a that the

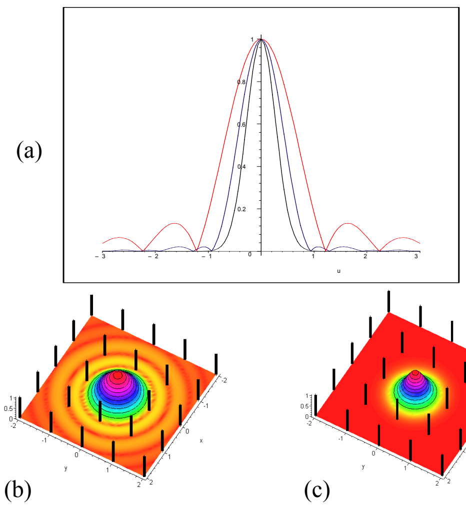


Figure 5: *(a) Radial profile of the 2D frequency responses for the radially symmetric box (red), linear (blue), and Gaussian (black) filters. (b) 2D plot of the frequency response of the linear filter, the black bars denote the DC locations of the signal replicas. (c) 2D plot of the frequency response of the Gaussian filter.*

curve for the linear filter goes to zero not only at multiples of 1.0, which suppresses low bands of the aliases along the major axes, but also at multiples of $\sqrt{2}$, which suppresses the low bands of the aliases along the diagonals (see Fig. 5b). However, we also see that the transformed Gaussian has much better suppression in the stop-band regions, at the expense of more blurring in the passband (as was observed in the 1D case as well). Note, that in these plots the zeros are not exactly at 1.0 and $\sqrt{2}$, but they could be moved there by increasing the kernel size slightly, via the Fourier scaling theorem.

The profiles of the 3D Hankel-transformed filters are shown in Fig. 6. We observe that the zeros at 1.0 and $\sqrt{2}$ are still there for the spherically symmetric linear filter's 3D frequency response (we shall call it 3D-L from now on). However, in a cubic cartesian frequency space, we ought to consider that there are three types of replica neighbors to the main spectrum (see also an illustration of one such Cubic Cartesian (CC) cell in Fig. 8a). The closest 6 (axis-aligned) neighbors are 1.0 away, which is where the 3D-L spectrum is low (or can be made low via scaling). Then there are 12 neighbors that are $\sqrt{2}$ away, where 3D-L is also low. However, there are also 8 neighbors that are $\sqrt{3}$ away (marked in blue in Fig. 8a), where 3D-L has a local maximum. This is a potential source for aliasing. The Gaussian 3D-G, on the other hand, has small values all around. Thus, we conclude that the reconstruction with 3D-L will likely tend to give more aliasing than 3D-G. Moreover, this aliasing is not removed by the subsequent convolution step. Since it is pre-aliasing for this second filtering step, it cannot be removed by subsequent filtering.

The bilinear spreading of points in Westenberg's and Roerdink's PCVR approach is synonymous with using a 3D-L filter for splatting (into a grid that has the same resolution than the volume). So we should expect aliased results, perhaps more visible at magnification levels larger than were used in their paper. On the other hand, the more traditional splatting with 3D-G will not yield aliasing. But before we go on and present results, we need to consider the post-convolution process as well.

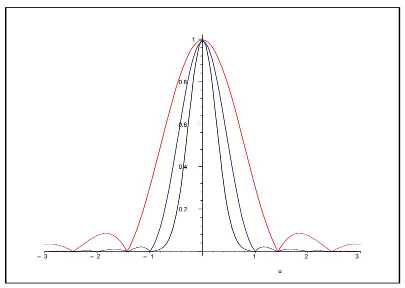


Figure 6: *Radial profile of the 3D frequency responses for the radially symmetric box (red), linear (blue), and Gaussian (black) filters.*

3.3. Stage 2: Post-convolution

The process of post-convolution is a 2D operation, and we would only use it for magnification. (For minification, the PCVR approach would not be used anyway as it does not offer any speed benefits.) Note, that the 2D convolution does not have to be performed with spherically symmetric filters. Axis-aligned filters, such as a bilinear filter, will be just as efficient. In fact, it is more efficient, since we can use it as a separable filter to speed up the convolution process (see Section 4 for more details).

The combined filter that results from the PCVR is the convolution of the 3D-G (the usual Gaussian employed for splatting) with the kernel used for post-convolution. If we use the 2D equivalent of the 3D-G for post-convolution, then the resulting Gaussian is $exp(-0.7 \cdot r^2)$ which is a stronger lowpass than the 3D-G. Therefore, it is actually better to use the bilinear filter for post-convolution, or a slimmer Gaussian, to keep the slice image sharp. Fig. 7 contrasts the combined filters that result from different filter pairs. Section 5 will show the implication on visual quality.

3.4. Potential of alternative grid topologies

Since it was somewhat disappointing to us that the (fast) bilinear spreading cannot not be applied in the general case, we also looked into the use of alternative voxel grid configurations. To that end, the BCC (Body-Centered Cartesian) grid has been found to provide the sparsest grid point distribution without compromising the signal content (under the condition that the signal is spherically bounded) [3]. It was used for splatting by [7][10]. The BCC grid gives rise to the FCC (Face Centered Cartesian) grid in the frequency domain. The FCC grid provides the closest 3D packing of signal spectra in the frequency domain [3]. This grid is shown in Fig. 8b. There we marked the 1-neighbors to the main spectra (those with distance 1.0) in red, the $\sqrt{2}$-neighbors in green, and the $\sqrt{3}$-

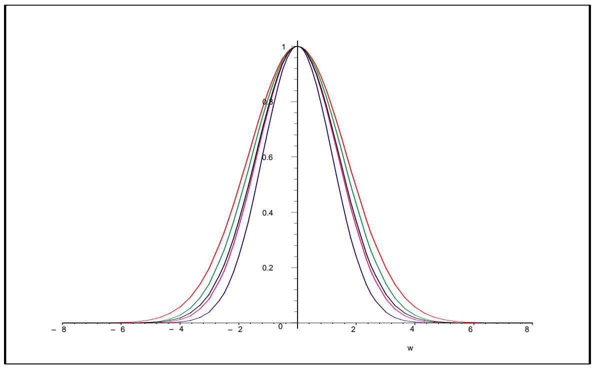


Figure 7: *Frequency plot of various combinations of filters, generated by convolving the filter used in stage 1 (splatting) with the filter used in stage 2 (post-convolution). We will use the following terminology: $G_{1.4}$: The traditional Gaussian with $\exp(-1.4 \cdot r^2)$; $G_{2.0}$: A slimmer Gaussian with $\exp(-2.0 \cdot r^2)$; B: Bilinear filter. From greater bandwidths towards smaller bandwidths, listing the splatting filter first and the post-convolution filter second:*

- *Red: $G_{1.4}$ used directly without post-convolution*
- *Green: $G_{2.0}$*B*
- *Black: $G_{2.0}$*$G_{2.0}$*
- *Magenta: $G_{1.4}$*B*
- *Blue: $G_{1.4}$*$G_{1.4}$*

neighbors in blue. There are 12 1-neighbors and 6 $\sqrt{2}$ -neighbors. (Note that there are more 1-neighbors than for the CC cell, which is why the FCC cell represents a closer packing.) But more importantly for this discussion, there are also 8 $\sqrt{3}$ -neighbors, exactly as many as in the standard CC grid. Due to this circumstance, we do not expect that the BCC grid will give better results than the CC grid for the bilinear spreading in the first phase of the PCVR.

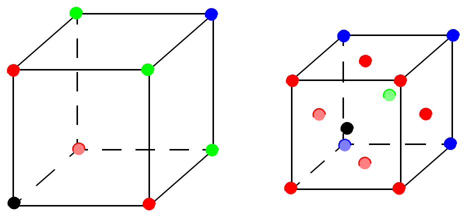


Figure 8: *(a) Cubic cartesian (CC) cell in frequency space, (b) Face centered cartesian (FCC) cell in frequency space. The black dot marks the location of the main spectrum. The red dots mark the 1-neighbors (the neighbors with distance=1), the green dots mark the $\sqrt{2}$ -neighbors, while the* red dots mark the $\sqrt{3}$ -neighbors.

4. Implementation Details

In our image-aligned sheet-buffered splatting pipeline [6], after the viewing transform is applied, each voxel is sliced during a front-to-back traversal of the volume. A sliced footprint of its basis function is then rasterized on each of the sheetbuffers it intersects. Since the density sheetbuffers match the image resolution, they are ready to be used for per-pixel classification and shading (post-classified rendering).

As explained in Section 3, PCVR consists of two distinct stages. For each sheetbuffer, the view-transformed volume is first sampled onto a grid of matching resolution (no scaling). Then a scaled convolution filter is used to resample the sheetbuffer to image resolution. After closely examining the PCVR process, a simple observation can be made: If the resulting image resolution is an exact multiple of the volume resolution, then all the pixels of the initial sampling grid (at volume resolution) coincide with pixels of the larger grid. This allows both stages to be performed on the same high resolution grid. Accessing the low resolution grid is just a matter of adjusting indices in the storage array. It also makes it possible to use the much more efficient discrete convolution using a separable kernel. It turns out that this is 3-4 times faster than a full convolution.

During the first stage a Gaussian kernel is used to splat the voxels into the low-resolution subgrid (see Fig. 9a). The radius of the kernel used is 2, so an area of 4x4 pixels is splatted on the low-res grid for each voxel. Here, after the viewing transform, voxels may project onto arbitrary image locations, so splatting requires a hi-resolution splatting kernel to be sampled by the pixels. This happens once per voxel, independent of the scale factor. As shown in Fig. 11h and Fig. 12h, this splatting stage just updates the corresponding pixels of the low-resolution subgrid.

At the second stage, the resulting sheetbuffer (with non-zero values only in the pixels of the low-res subgrid) is convolved with a 2D kernel that matches the image resolution in order to update the values of the hi-res grid of the sheetbuffer and fill in the gaps. Since the positions of the low-res grid pixels now coincide with the hi-res grid pixels, there is no need for a high-resolution kernel lookup table that can be sampled everywhere, we can just use discrete convolution (see Fig. 9b). The kernel is only sampled once to match the resolution of the target image. Further savings can be achieved by using a separable filter, like a Gaussian or the bilinear filter for this operation. As shown earlier, both will produce acceptable results in this 2D operation. However, the smaller bilinear kernel achieves much better performance in terms of speed (see Fig. 9c).

Our rendering engine is further optimized using an image-driven occlusion scheme. Simulating the effect of early ray termination in ray-casting, the system maintains an “occlusion map”. The occlusion test culls the voxel if its footprint area in the resulting image buffer is fully opaque. Additionally, in this implementation we are maintaining a tile structure which provides a coarse grain division of the image. Only the tiles with updated pixels are post-convolved using a cache-friendly separable convolution algorithm.

For X-Ray images the original rendering engine would keep updating the footprints of voxels into the same sheetbuffer, since no depth information is necessary for this visualization. After the whole volume is traversed, the final image is

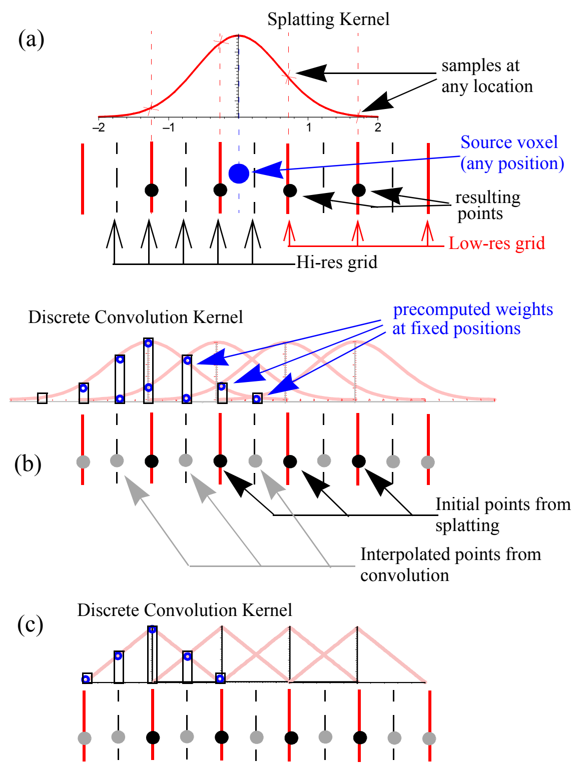

(a)
Splatting Kernel
samples at any location
Source voxel (any position)
resulting points
Low-res grid
Hi-res grid
Discrete Convolution Kernel
precomputed weights at fixed positions
(b)
Initial points from splatting
Interpolated points from convolution
Discrete Convolution Kernel
(c)

Figure 9: *Overview of the PCVR in our splatting pipeline: (a) splatting of a voxel into a (low-resolution) grid at volume resolution, (b) post-convolution of the low-resolution grid samples to give the high resolution grid samples, using Gaussian kernel, (c) using a linear bilinear filter (shown for 1D)*

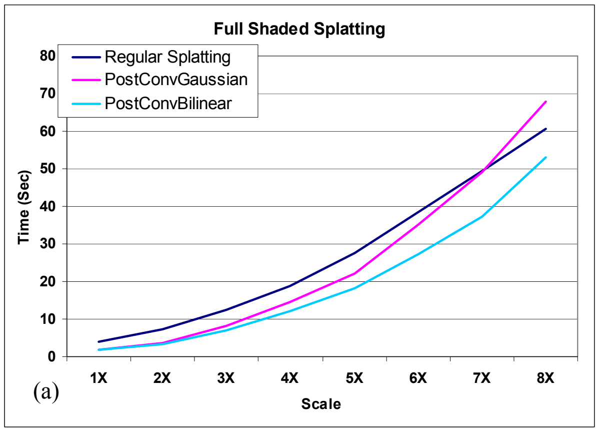
produced by adjusting the resulted intensities to fit the image range. The PCVR version of X-Ray splatting accumulates all of the splatted voxels in the low-resolution subgrid of the intensity buffer, and it convolves the resulting image only once in the end. As we can see in the results section, PCVR provides even higher speedups for X-Ray rendering.

5. Results

We now present the performance impact of applying PCVR to our current software implementation of image-aligned sheet-buffered splatting. In terms of performance we have observed speedups of two and more, depending on the scale factor. Fig. 10a and b show rendering times of the UNC CT Head dataset for both full shaded rendering and X-Ray rendering. The curves in Fig. 10a show the time taken to render the image using traditional splatting (blue curve), PCVR using a Gaussian kernel for the post-convolution (magenta), and PCVR using a bilinear filter for the post-convolution (sky blue). In the first graph, we observe significant reductions and speedups of about 2 for magnification < 5. After magnification=7 we can see that PCVR using a Gaussian post-convolution kernel is actually slower than regular splatting. This can be explained by the fact that full shaded splatting is image driven using an occlusion culling scheme. Our PCVR port of the splatter uses occlusion culling, on top of a coarse grain tile structure that depends on the size of the post-convolution kernel, and it drives the convolution process. The smaller bilinear kernel behaves much better under these circumstances.

In X-Ray imaging, PCVR seems to have a more obvious advantage. In this case all the voxels have to be splatted, and rasterization is the dominating operation. The advantage of PCVR is that it only splats at the resolution of the volume. Thus the rasterization operations are not affected at all by the image size. In addition, post-convolution only occurs once, at the end of the volume traversal.

In order to compare the quality of the proposed method we have rendered the CT head dataset both as an isosurface and as an X-Ray image. In Fig. 11a-c we show the head rendered using regular splatting, PCVR using a bilinear post-convolution filter, and PCVR using a Gaussian post-convolution filter. The expected effect of the post-convolution is slight blurring, as explained in the previous section. The (amplitude-magnified) difference images in Fig. 11e-g show that the differences are observed mainly at sharp edges and contours. Fig. 11d is a rendering using bilinear spreading as the first step of the process. This image illustrates the aliasing artifacts expected



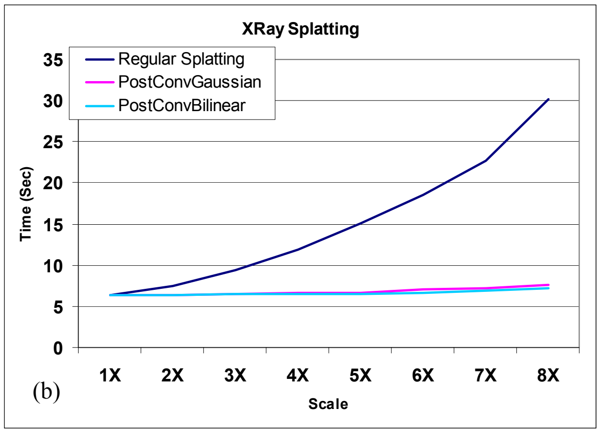


Figure 10: *Impact of PCVR on our splating pipeline for: (a) Full Volume rendering, (b) X-Ray Splatting. The graphs show time to render in seconds for scales of 1X to 8X.*

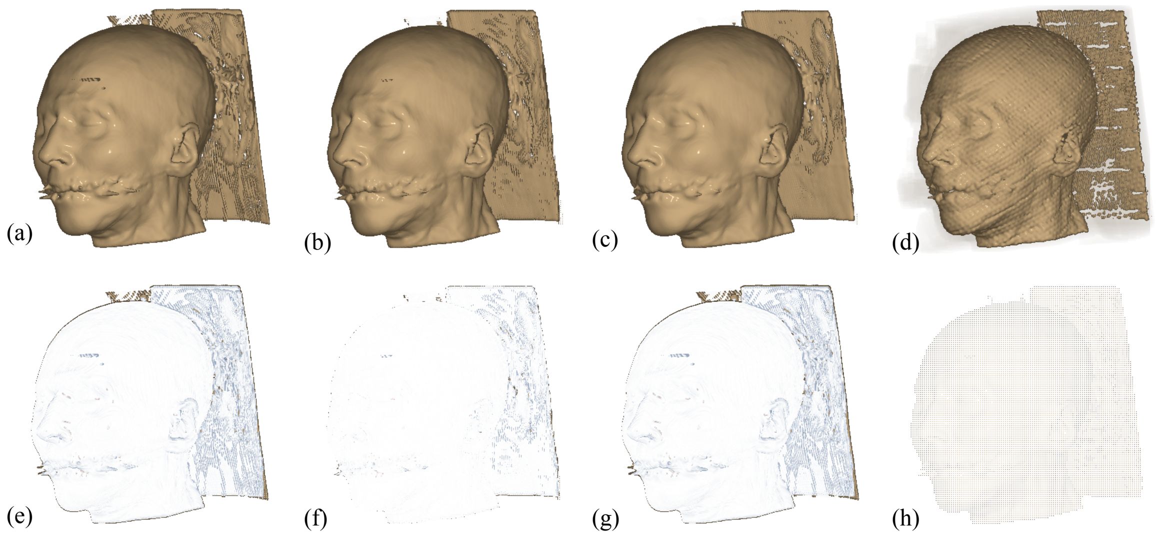


Figure 11: *Full rendering of CT Head using: (a) Regular Splatting, (b) PCVR with bilinear Post Convolution, (c) PCVR with gaussian Post Convolution, (d) PCVR with Bilinear Spreading (pre-convolution). (e),(f),(g): Difference Images of (a-b), (b-c), (a-c). (h) PCVR without the Post Convolution, to better demonstrate the process.*

from this approach. Finally, Fig. 11h is a rendering without the post-convolution step, to better demonstrate the algorithm.

As we observe the X-Ray images in Fig. 12, we can still observe a slight blurring effect visible mostly in the Gaussian post-convolved image of Fig. 12c. For the image rendered with bilinear splatting in Fig. 12d, we can already observe slight aliasing artifacts that appear as lines within the image. Again, these are caused by the less than perfect spherically symmetric bilinear filter used. The four images in the bottom row (Fig. 12i) have all been rendered using the bilinear spreading method for magnifications of 1X, 2X, 3X, and 4X. We can observe that for magnification 1 and 2, there no perceivable artifacts, as was also shown in [11]. However, as the magnification increases to 3 and higher, the aliasing becomes more visible in the image. Note that these artifacts are easier to detect at low magnifications in full shaded renderings, since there the gradient estimation and shading operations are very sensitive to even small errors. Finally, in Fig. 12k, we contrast these images with the alias-free image obtained with the 3D-G and subsequent bilinear convolution.

6. Conclusions

We have shown that post-convolved volume rendering (PCVR) can yield significant speed-ups for magnified viewing, especially for X-ray volume rendering. It reduces the rendering effort by moving all 3D interpolations that serve volume supersampling to a subsequent, less expensive 2D post-convolution stage. We discussed the method in the context of splatting, using spherically symmetric interpolation filters. Previous approaches suggested (for X-ray) to perform a simple bilinear spreading of voxels into the low-resolution grid, followed by a post-convolution into the desired high-resolution grid. We found (and justified), however, that this can lead to significant rendering artifacts, which are caused by aliasing in the first stage of the process. The aliasing comes from the fact that the bilinear spreading is equivalent to splatting a spherically symmetric linear filter kernel, whose frequency response is unfortunately non-zero at some of the signal replicas. Based on our analysis, we found that it is advised to instead use the traditional Gaussian in the first stage, which has a more favorable frequency response. On the other hand, it is permissible to use an efficient bilinear filter in the post-convolution stage, and we also found that for integer magnifications an efficient discrete post-convolution scheme can be employed, which provides additional significant speedups for PCVR. A downside of the PCVR approach is that it may lead to a slight increase in blurring since the combined filter is a convolution of two lowpass filters. Our experiments indicated, however, that the blurring is relatively minor.

The PCVR method has a number of application scenarios in which it can proof useful. The first is in software splatting, and, in fact, we use the PCVR scheme as a default option in our in-house software splatter. We also think that it can be useful in hardware implementations. We are currently porting our image-aligned sheet-buffered splatting approach to commodity hardware platforms, such as nVIDIA and ATi. For PCVR, one could rasterize the texture-splats into a low-resolution buffer and then project this buffer into a high resolution buffer, all in hardware. At the current time we cannot report any timings, but it may be beneficial to do this if the application turns out to be rasterization-bound. We have already seen the application of PCVR for the image generation from hierarchical basis functions, such as wavelets [11]. We think that our approach will facilitate higher magnifications and therefore more levels, artifact-free. We also think that PCVR may be useful for custom hardware implementations. For example, the VolumePro 500 [8] could only cast one ray per voxel. To increase resolution without having to cast more rays, a post-convolution module could be inserted before the shading pipeline but after interpolation. Similarly, the shear-warp algorithm could be modified by inserting a post-convolution module after each slice interpolation. Tiles could be used to limit the convolution effort. Finally, our findings are also relevant for medical imaging, in particular for CT reconstruction algorithms, which often use bilinear spreading in their projectors/backprojectors. The added accuracy obtained from using a better kernel may help to increase reconstruction fidelity.

Acknowledgements

This research was partially supported by NSF CAREER grant ACI-0093157 and DOE grant MO-068.

References

[1] A. Baerentzen, “Octree-based sculpting,” *Late Breaking Hot Topics IEEE Visualization’98*, pp. 9-12, 1998.

[2] R. Bracewell, *The Fourier Transform and its Applications - 3rd edition*, McGraw-Hill, 1999.

[3] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices and Groups*, 2nd edition, Springer Verlag, 1993.

[4] J. Jang, C. Shaw, W. Ribarsky, and N. Faust, “View-dependent multi-resolution splatting of non-uniform data,” *Eurographics/IEEE TGVC Visualization Symposium 2002*, pp. 125-132, 2001.

[5] P. Lacroute and M. Levoy, “Fast volume rendering using a shear-warp factorization of the viewing transformation,” *Proc. SIGGRAPH ‘94*, pp. 451- 458, 1994.

[6] K. Mueller, N. Shareef, J. Huang, and R. Crawfis, "High-quality splatting on rectilinear grids with efficient culling of occluded voxels," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 2, pp. 116-134, 1999.

[7] N. Neophytou and K. Mueller, “Space-time points: splatting in 4D,” *Symposium on Volume Visualization and Graphics 2002,* pp. 97-106, 2002.

[8] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, “The VolumePro real-time raycasting system,” *Proc. SIGGRAPH 99,* p. 251-260, LA, August 1999.

[9] J. Sweeney and K. Mueller, "Shear-Warp Deluxe: The Shear-Warp algorithm revisited," *Joint Eurographics - IEEE TCVG Visualization Symposium 2002*, pp. 95-104, Barcelona, Spain, May 2002.

[10] T. Theussl, T. Möller, and E. Gröller, “Optimal regular volume sampling,” *Proc. Visualization’01*, pp. 91-98, 2001.

[11] M. A. Westenberg and J. B. T. M. Roerdink, “X-ray volume rendering through two-stage splatting,” *Machine Graphics and Vision*, vol. 9, no. 1/2, pp. 307-314, 2000.

[12] L. Westover, “Footprint evaluation for volume rendering”, *Proc. SIGGRAPH’90*, pp. 367-376, 1990.

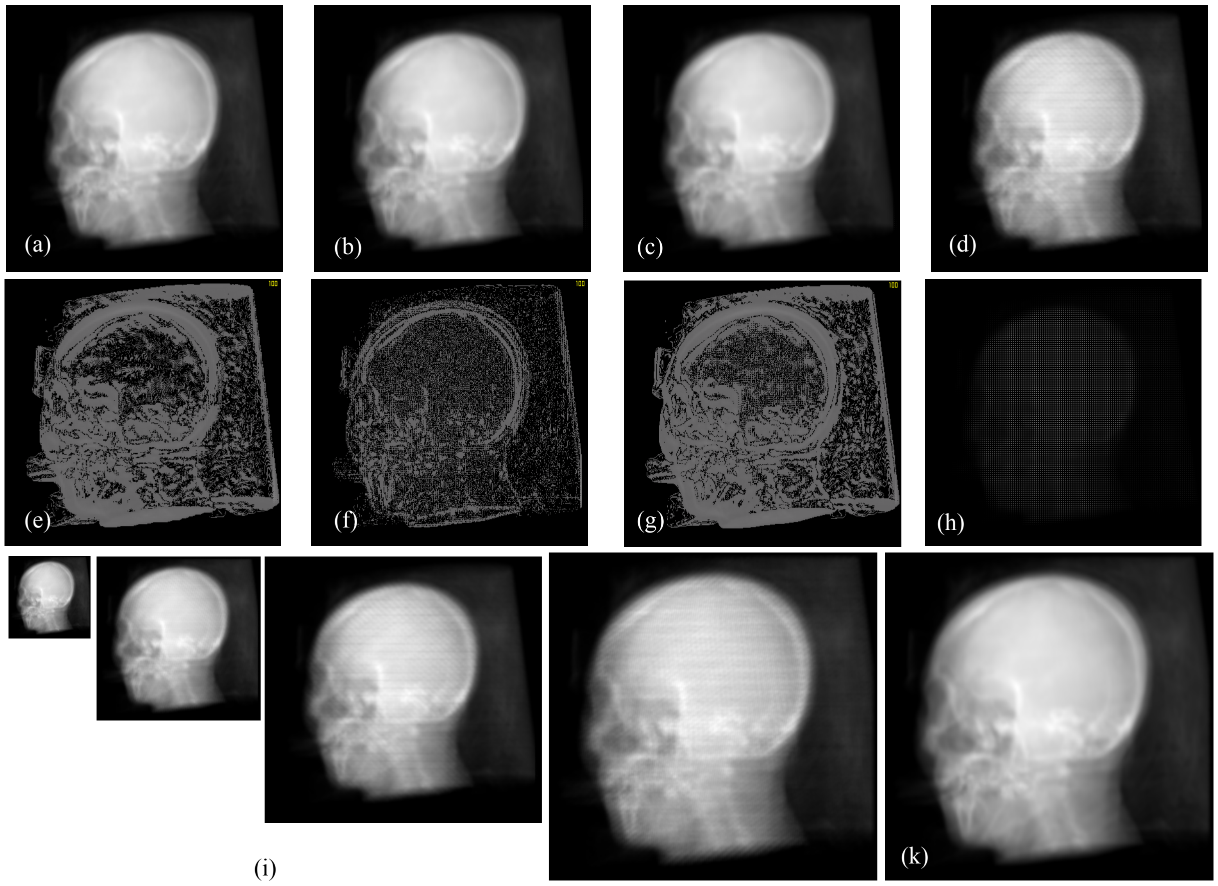


Figure 12: *X-Ray rendering of the CT Head using: (a) regular splatting, (b) PCVR with bilinear post-convolution, (c) PCVR with Gaussian post-convolution, (d) PCVR with bilinear spreading (pre-convolution). (e),(f),(g): amplified difference Images of (a-b), (b-c), (a-c). (h) PCVR without the post-convolution, (i) compares the image quality of PCVR with bilinear spreading at scales of 1X, 2X, 3X, 4X, (k) the image obtained with PCVR at 4X via Gaussian spreading and bilinear post-convolution.*