Continuous Fourier Volume Rendering of Irregularly Sampled Data Using Anisotropic RBFs

H.Quynh Dinh [a]

[a] *Department of Computer Science*
Stevens Institute of Technology
quynh@cs.stevens.edu
phone: 201-216-5321
fax: 201-216-8249

Neophytos Neophytou

Department of Computer Science
Stony Brook University

Klaus Mueller

Department of Computer Science
Stony Brook University

Abstract

We describe a Fourier Volume Rendering (FVR) algorithm for datasets that are irregularly sampled and require anisotropic (*e.g.*, elliptical) kernels for reconstruction. We sample the continuous frequency spectrum of such datasets by computing the continuous Fourier transform of the spatial interpolation kernel which is a radially symmetric basis function (RBF) that may be anisotropically scaled. While in the frequency domain, we can apply low, band, and high-pass filters and arbitrary magnification and minification of the dataset before performing an inverse 2D Fourier transform to obtain the X-ray projection. Our algorithm is particularly amenable to implementation on commodity programmable graphics boards, and can interactively render X-rays for datasets on the order of tens of thousands of points. We describe the theoretical considerations to properly sample the frequency spectrum of anisotropic RBFs to avoid aliasing in the resulting X-ray and present a practical method for datasets with high sampling requirements. A significant benefit of our algorithm is that it can be applied to anisotropic RBFs that have been fitted to data through optimization techniques, allowing the incorporation of advanced data-sensitive constraints, such as smoothness, sharpness, and feature preservation.

Key words: Fourier volume rendering, frequency domain resampling, radial basis functions, X-ray imaging

1 Introduction

Fourier Volume Rendering (FVR) generates projection images (X-rays) of a 3D volume of N^3 size in $O(N^2 logN)$ time rather than the $O(N^3)$ complexity of typical volume rendering [8,12,14]. X-rays are practical for displaying volumes because all internal structures are shown in one visualization. X-rays often provide a useful first look at the data, particularly when no true surface exists, and doctors still rely extensively on this type of visualization.

FVR requires a one-time pre-processing step to transform the 3D data into the frequency domain via a discrete Fast Fourier Transform (FFT) (an $O(N^3 logN)$ operation). Conventional FFT requires that the data be uniformly (regularly) sampled. The Non-uniform Discrete Fourier Transform (NDFT) overcomes this limitation, although with limits on the degree of irregularity [15]. Using any discrete form of the Fourier Transform leads to a discrete frequency spectrum that must then be resampled via interpolation to generate arbitrary views in FVR [14]. The interpolation kernel must be carefully designed to avoid aliasing, and the spatial data must be pre-multiplied prior to applying the Discrete Fourier Transform (DFT) to avoid attenuation in the resulting X-ray images.

Our contribution is an algorithm that directly samples at arbitrary resolution the continuous frequency spectrum of irregularly sampled datasets. Instead of using the discrete Fourier transform, we sample the continuous Fourier transform of the spatial interpolation kernel – a radial basis function (RBF) which may be anisotropically scaled and located at irregular spatial intervals. Note that we do not resample the data in the spatial domain to obtain a regular volume that is then brought to the frequency domain via the FFT. With a continuous frequency spectrum, our FVR algorithm does not need to interpolate discrete frequency samples or deal with aliasing in the frequency domain. However, we will need to sample the frequency spectrum with sufficient resolution to avoid aliasing in the resulting X-ray (spatial domain). We call this approach *continuous Fourier Volume Rendering.*

While in the frequency domain, we can apply conventional signal processing filters, such as low, band, and high-pass filters. Arbitrary magnification and minification are possible by, respectively, increasing the frequency samples or by zero-padding. We can also use this algorithm to generate a regular, albeit dense, sampling of the irregular dataset by performing an inverse 3D DFT. Finally, our algorithm is particularly amenable to implementation on the Graphics Processing Unit (GPU) of commodity programmable graphics boards and can be used to interactively render X-rays for datasets on the order of tens of thousands of points.

Our method generalizes [5] where the authors sample the continuous frequency spectrum of isotropic RBFs to visualize meshless simulations using collocation and particle hydrodynamics. They do not address anisotropic RBFs and the issues therein such as frequency sampling requirements. In this paper, we describe the the-

oretical considerations in properly sampling the frequency spectrum of anisotropic RBFs to avoid aliasing, present a practical method for challenging datasets with high sampling requirements, and describe an optimal GPU algorithm for FVR of irregularly sampled datasets using anisotropic RBFs.

In Section 2, we review Fourier Volume Rendering and radial basis functions for spatial data interpolation, describe continuous FVR for irregularly sampled data in Section 3, and present examples and results in Section 4.

2 Previous Work

Because our work extends conventional FVR to irregularly sampled data, we review FVR in detail. We also discuss the particular RBFs used in our paper and place our contributions in the context of prevous work in frequency domain resampling.

2.1 Fourier Volume Rendering (FVR)

Fourier Volume Rendering was first presented separately by Dunne [8], Levoy [12], and Malzbender [14]. FVR generates an X-ray projection (summed volume rendering) of a 3D volume using the Fourier Projection-Slice Theorem. The theorem states that a 2D slice passing through the origin of the 3D Fourier transform is the 2D Fourier transform of the projection of the 3D space in the direction orthogonal to the slice. Using the notation presented in [14], we define an image plane whose normal vector t together with u and v form an orthonormal basis in $\mathbb{R}^3$. Given a continuous 3D distribution $f(x,y,z)$, the parallel projection $p(u,v)$ of f onto the image plane is:

$$p(u,v) = \int_{-\infty}^{\infty} f(t,u,v)dt \tag{1}$$

The Fourier Transform of $f(x,y,z)$ is $F(\omega_x,\omega_y,\omega_z)$ and the plane in frequency space corresponding to the image plane is defined by orthonormal vectors, $W_u = (w_{ux},w_{uy},w_{uz})$ and $W_v = (w_{vx},w_{vy},w_{vz})$. Using the Fourier Projection-Slice Theorem, $p(u,v)$ is obtained via the inverse Fourier Transform of a spectrum slice, $P(\omega_u,\omega_v)$:

$$p(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} P(\omega_u,\omega_v)e^{j2\pi(u\omega_u+v\omega_v)}d\omega_u,d\omega_v \tag{2}$$

where,

$$P(\omega_u,\omega_v) = F(w_{ux}\omega_u + w_{vx}\omega_v, w_{uy}\omega_u + w_{vy}\omega_v, w_{uz}\omega_u + w_{vz}\omega_v) \tag{3}$$

In conventional FVR, the 3D Fourier Transform of the volume is computed in a pre-processing step via the discrete FFT. Interpolating a 2D slice and transforming

it back to the spatial domain via an inverse 2D DFT is an $O(N^2 logN)$ operation, which is much more efficient than the $O(N^3)$ of volume rendering but requires careful design of a frequency domain interpolation kernel to avoid aliasing and attenuation in the spatial domain [14]. A GPU-enabled FVR algorithm has been developed, but deals primarily with mapping the rendering stage (frequency slice extraction and interpolation of frequency samples) on the GPU. Computing the discrete 3D frequency spectrum remains a pre-processing step [24]. In our new algorithm, we directly sample the continuous frequency spectrum of the data and avoid the convolution and pre-multiplication of the data needed to appropriately resample a slice of the discrete frequency spectrum. We will, however, need to sample the frequency spectrum with sufficient resolution to avoid aliasing in the spatial domain (resulting X-ray) and address these issues in Section 3.1.

2.2 Interpolation via Radial Basis Functions

A summation of weighted radial basis functions (RBFs) of the following form has been used as a data interpolant in many domains:

$$f(\vec{x}) = \sum_{i=1}^{n} w_i \phi_i(\vec{x} - \vec{c}_i) \tag{4}$$

In the above equation, ϕ_i are the interpolation kernels (typically, a truncated Gaussian); c_i are the data points (centers of the kernels); and w_i are densities associated with each data point. For volume rendering, Equation 4 is convolution, and the centers and densities are obtained using kernel fitting algorithms such as [9]. RBFs have also been used to encode scalar fields [10,25] and for reconstructing implicit surfaces [4,7,16,19,20]. In this paper, we use the splatting kernel ($\phi(r) = 2^{-r^2}$) often found in volume rendering, though other RBFs may be applied to our algorithm.

2.3 Non-uniform DFT

The non-uniform discrete Fourier transform (NDFT) computes the discrete frequency spectrum for non-uniformly sampled data. Methods to compute the NDFT in 1D include the direct method, Horner's and the Goertzel algorithm [15]. Unfortunately, there are no similar extensions to 2D and beyond because even if the sample points are distinct, there is no guarantee that the NDFT matrix is not singular. There are two special cases in which the 2D NDFT matrix can be guaranteed to be non-singular. These are: (1) rectangular grids whose cell width or height may vary, and (2) non-uniform sampling on parallel lines. [21] uses the NDFT to regrid originally irregular datasets into a regular domain and enable interactive exploration and zooming-in of these datasets using FVR. His approach handles irregular sampling, but not anisotropically scaled kernels. As a result, gaps are apparent in the resulting X-ray. The novel RBF approach that we present handles the more general case of irregularly sampled, anisotropically scaled data.

2.4 Frequency Domain Resampling

A number of publications highlight the benefits of frequency domain resampling. In [13], Li *et al.* showed that even when a lower-quality filter is used (e.g., cubic or linear) in the frequency domain, a significant performance advantage is achieved. A similar result was arrived at in [1] where an algorithm based on Shear-Warp Factorization [11,22] was used. Our new approach enables high resolution sampling of the frequency domain because we compute the *continuous* frequency spectrum of our data set using radial basis functions. Hence, we do not need to resample, or interpolate, discrete frequency samples.

2.5 Computing the Frequency Spectrum of Isotropic RBFs

The continuous Fourier transform of a summation of weighted RBFs, as derived in [3], is:

$$F(\omega) = \sum_{i=1}^{N} w_i e^{-j2\pi\omega\vec{c}_i} \Phi_i(|\omega|) \tag{5}$$

In the above equation, $\Phi_i(|\omega|)$ is the Fourier transform of the RBF ϕ_i at each data point; w_i are the weights (or values) of the N scattered samples; and $e^{-j2\pi\omega\vec{c}_i}$ is due to applying the Fourier Shift Theorem to RBFs centered at $\vec{c}_i$. A more general formulation is derived in [15]. For regularly sampled datasets, ϕ_i is isotropic and identical (homogeneous) for all data points, and hence the i subscript can be dropped and $\Phi(|\omega|)$ can be pulled outside of the summation:

$$F(\omega) = \Phi(|\omega|) \sum_{i=1}^{N} w_i e^{-j2\pi\omega\vec{c}_i} \tag{6}$$

In this case, $\Phi(|\omega|)$ is a radially symmetric function and essentially behaves as a low-pass filter. Corrigan and Wallin use this homogeneous isotropic FVR formulation to visualize the results of meshless simulation using collocation and smoothed particle hydrodynamics [5]. In our approach, we allow ϕ_i to be scaled uniquely and anisotropically for each data point.

2.5.1 Spatial Interpolation Kernels for FVR

The RBF we use is the Gaussian kernel used in splatting ($\phi(r) = 2^{-r^2}$) whose generalized Fourier Transform is:

$$\frac{1}{\ln^{\frac{3}{2}} 2} \pi^{\frac{3}{2}} e^{-\pi^2 \frac{r^2}{\ln 2}} \tag{7}$$

We note that other RBFs can be applied to the FVR algorithm presented here so long as they do not approach infinity anywhere along their base of support. Figure 1

are examples of using an isotropic kernel with continuous FVR on several regularly sampled datasets. Note that the color in the X-ray renderings throughout the paper are generated by applying a transfer function to the 2D X-ray images from FVR not to the original volumetric dataset.

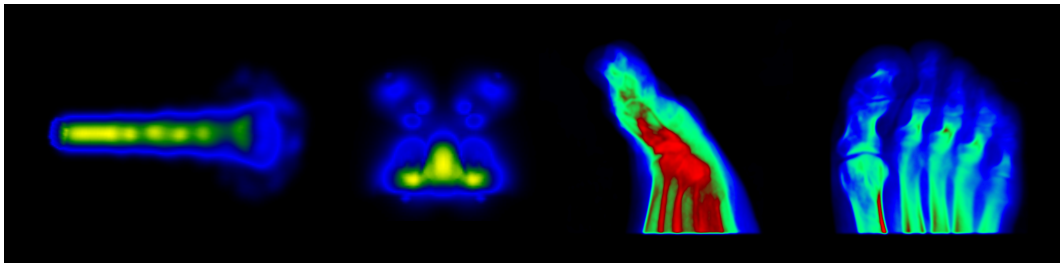

Fig. 1. Continuous FVR of data with uniform sampling using isotropic kernels (left to right): fuel injection into a combustion chamber, spatial probability distribution of electrons in a protein molecule, and rotational C-arm x-ray scan of a human foot.

3 Computing the Frequency Spectrum of Anisotropic RBFs

In the general formulation (Equation 5), the radial basis functions are not homogeneous. Each RBF is individually and non-uniformly scaled about its center, resulting in an anisotropic RBF such that the direction and magnitude of anisotropy can differ from one data point to the next. Elliptical RBFs are those that have been scaled non-uniformly along 3 principle directions. Anisotropic scaling may be the result of sampling data on curvilinear or non-uniform rectilinear grids (thus requiring an anisotropic reconstruction kernel) or from kernel fitting optimization techniques [7,9] that generate elliptical RBFs and allow the incorporation of more advanced data-sensitive constraints, such as smoothness, sharpness, and feature preservation. To compute the frequency spectrum for anisotropic RBFs, we apply the Fourier Scaling Theorem [2]. We note that although our sample datasets are composed of elliptical RBFs, our method can be applied to RBFs that have undergone arbitrary transformations.

We must address two key issues to achieve continuous FVR on the general anisotropic formulation. The first is to define the sampling requirements in frequency space to ensure that high frequency information is retained while preventing aliasing in the spatial domain (resulting X-ray). The second is how to optimize sampling of the frequency spectrum since $\Phi_i(|\omega|)$ cannot be pulled out of the equation.

3.1 Sampling Density

Because we sample the continuous frequency spectrum, we avoid aliasing in frequency space, but may encounter aliasing in the spatial domain. The inverse DFT assumes periodicity of the spatial signal, and so, copies of the reconstructed signal

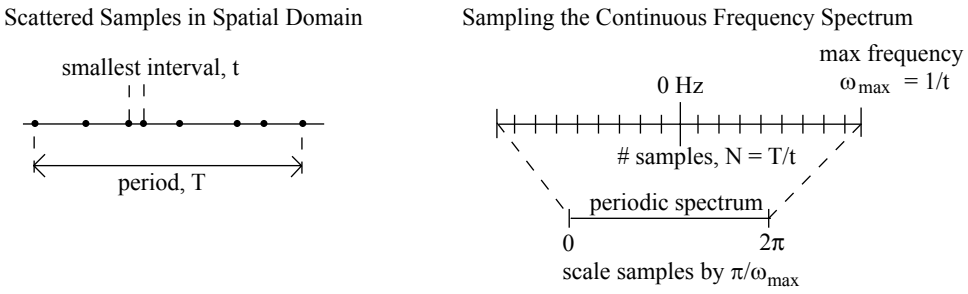

Fig. 2. Sampling density required in the frequency domain given spatial domain samples spanning period T with t as the smallest interval between two samples. The maximum frequency corresponds to the smallest spatial interval, $\omega_{max} = 1/t$, and the number of frequency samples N depends on the maximum number of spatial samples that can occur within the period T given the smallest interval ($N = T/t$).

may overlap. Aliasing can be avoided if we sample the frequency spectrum densely enough. Figure 2 relates the density of input spatial samples to the maximum frequency and resolution required for sampling the frequency spectrum to completely capture spatial variation and avoid aliasing. As we increase the number of samples, the spatial period increases, reducing overlap between copies. For data sampled on a regular grid, we can compute the number of frequency samples required based on the dimensions of the dataset [14]. If the sampling is irregular, we can avoid aliasing by assuming regular sampling at the smallest spatial interval t and over-sample to $N = T/t$. The smallest spatial interval also determines the maximum frequency: $\omega_{max} = 1/t$. We uniformly sample the frequency spectrum with N samples in the range of frequences $-\omega_{max} < \omega < \omega_{max}$. Because the IFT assumes a frequency period of 0 to 2π, we must rescale our samples by π/ω_{max}.

3.2 Sampling Density for Anisotropic RBFs

For anisotropic RBFs, the required sampling density depends on RBF scaling. When the scaling is less than 1 (compressed RBFs), the data points are, in effect, brought closer together. The minimum spatial interval is further reduced, requiring a denser sampling of the frequency spectrum: $N = T/(s*t)$ and $\omega_{max} = 1/(s*t)$ where s is the minimum RBF scaling. As a result, N and ω_{max} increase as s approaches 0. Most datasets do not exhibit excessive compression, and N and ω_{max} do not become prohibitively large. We have one dataset that is an example of excessive compression and variation in sampling density – the Blunt Fin. To handle this challenging dataset, we must partition the data as described next.

3.2.1 A Dataset With Excessive Compression and Variation in Sampling Density

The Blunt Fin (Figure 3) contains a small region of very dense samples surrounded by sparser samples of elliptical RBFs. This leads to two problems: (1) as described above, the dense sampling means a very small minimum spatial interval and a very

high max frequency, and (2) the intensity of the resulting X-ray varies widely, causing the dense region to overwhelm the rest of the X-ray. As suggested in [17], the solution to (2) is to normalize the X-ray by dividing it with another X-ray image that is generated using a weight of 1.0 for all data points. Although this is effective in equalizing the X-ray intensities (see Figure 3a), oscillation or ringing due to an insufficient number of frequency samples (problem 1) is amplified. When we compute a square frequency image below the max frequency component, we are essentially applying a box filter which is a sinc function in the spatial domain, leading to ringing. We can apply a Hamming window in the frequency domain to reduce ringing, but the result is data leakage beyond the original boundaries of the dataset and loss of high frequency detail (Figure 3b). We can completely eliminate leakage and oscillation by significantly low-pass filtering in the frequency domain, but even more spatial features are lost in the process.

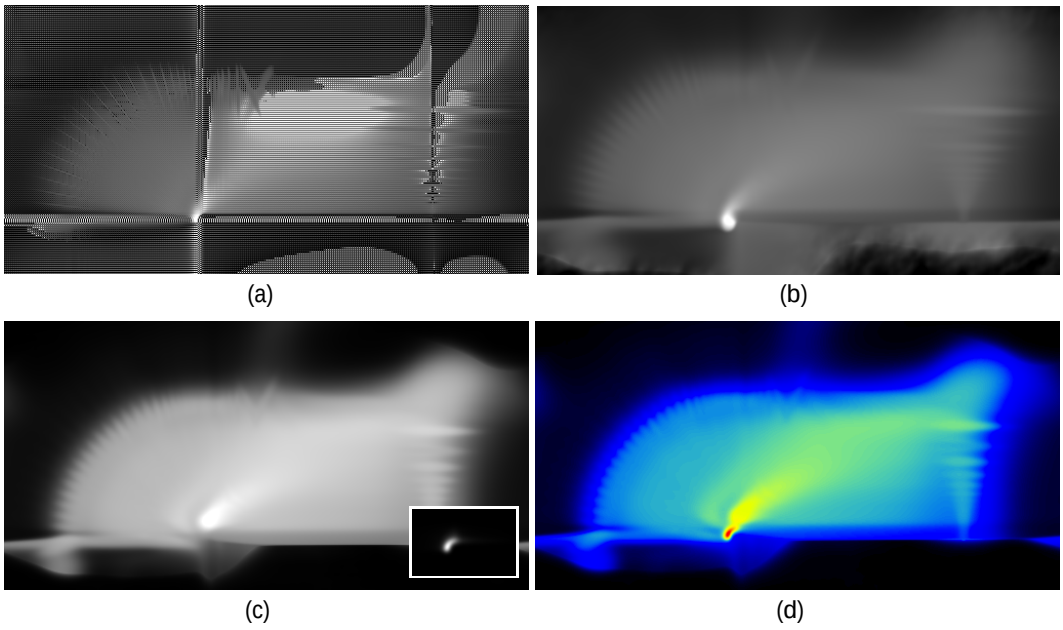

Fig. 3. (a) X-ray of Blunt Fin with normalization; (b) with windowing; (c) partitioned into two datasets, and (d) resulting X-ray of composited partitions with color transfer.

Because the problems are due to a wide variation in sampling density, we partition the dataset into regions of more common density, and in doing so, retain high frequency information. In Figure 3c, we separate the high density region from the rest of the dataset. We then combine the two resulting X-rays into a single image since X-ray imaging is additive (Figure 3d). The partitioned X-rays are free of ringing, but data leakage is still apparent. Partitioning the dataset further reduces leakage at the cost of maintaining and compositing X-ray partitions. The Blunt Fin is an example of excessive variation in sampling density. All other irregularly sampled datasets we have encountered do not require partitioning.

3.3 Computing the Frequency Spectrum on the GPU

To compute the frequency spectrum, we compute only half the spectrum using Equation 5, and compute the second half using conjugate symmetry. We avoid evaluating the Fourier Transform of the RBF (Φ) for every frequency sample by precomputing Φ and storing it in a look-up table. This freedom allows us to use arbitrary precision when computing Φ and linearly interpolate between values in the look-up table. In practice, we build a 1D look-up table with a resolution of 4096 samples and have found no visible difference from explicitly evaluating Φ for each frequency sample. We compute a slice of the frequency spectrum by rendering a quad to accumulate the shift factor $e^{-j2\pi\omega\vec{c}_i}$ for each location ω in parallel. For uniform isotropic RBFs, the texture storing Φ is simply multipled to the quad once for the entire image. For elliptical RBFs, the texture is anisotropically scaled, multiplied, and accumulated on a per pixel basis. For a 3D frequency spectrum, N depth slices are computed using the above method for 2D slices. In Figure 7, we compare CPU versus GPU times for computing a 2D slice of the frequency spectrum.

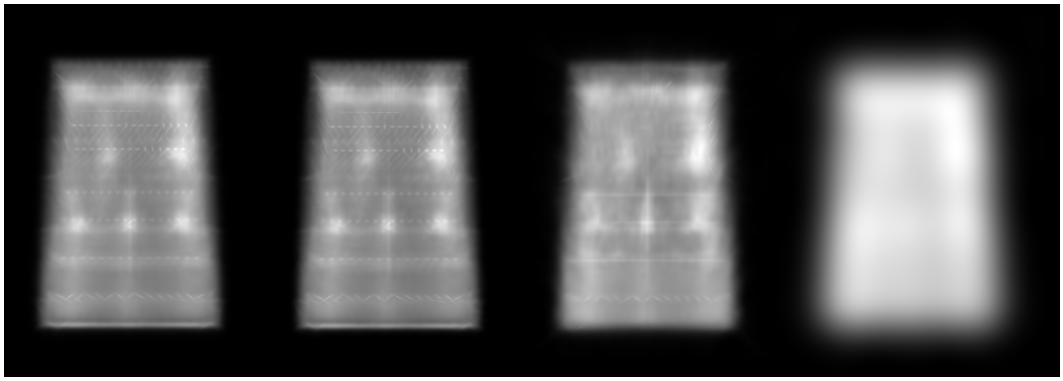

Fig. 4. X-rays of the Combustion Engine generated by (left to right): computing the frequency spectrum on the CPU, on the GPU, using interpolation on the GPU, and using isotropic RBFs. Using homogeneous isotropic RBFs for all data points is the most efficient, but sharp features are lost. Interpolation (3rd X-ray) is more efficient that computing the spectrum on the GPU, but some blurring is apparent. The first two X-rays are virtually identical, and computing the spectrum on the GPU is significantly faster than on the CPU.

Our initial GPU implementation described above rotates and anisotropically scales RBFs in the fragment shader on a per pixel basis because the Fourier Transform of anisotropic RBFs cannot be pulled out of the summation of Equation 5. As commonly known, the amount of computation in the fragment shader should be kept to a minimum for faster runtimes. In an optimized algorithm, we rotate and anisotropically scale the texture coordinates used to texture map the RBF footprint in the CPU. We then rely on the GPU's linear interpolation of the texture coordinates to sample the RBF footprint in the fragment shader. This optimization does indeed reduce the computation time for the frequency spectrum. However, some blurring is apparent in the resulting X-rays due to interpolation of the texture coordinates as shown in Figure 4 (3rd X-ray). In Figure 8, we compare timing results for computing the frequency spectrum on the GPU for isotropic RBFs and anisotropic RBFs

using both the original and optimized GPU algorithms.

4 Results

Continuous FVR can operate in two modes. In the first mode, we directly compute a 2D slice of the frequency spectrum at arbitrary resolution. In the second mode, we pre-compute a 3D frequency spectrum and interactively slice the volume to obtain a 2D slice to which the inverse FT is applied. The latter mode is conventional FVR as presented in [8,12,14] and requires sampling the 3D frequency only once. Doing so constrains the resolution of the spectrum and X-ray, resulting in aliasing artifacts or blurring when zoomed-in as shown in the left and center panels of Figure 5. By directly computing a 2D slice of the frequency spectrum, we are able to generate accurate high resolution X-rays during arbitrary zoom-in (right panel of Figure 5).

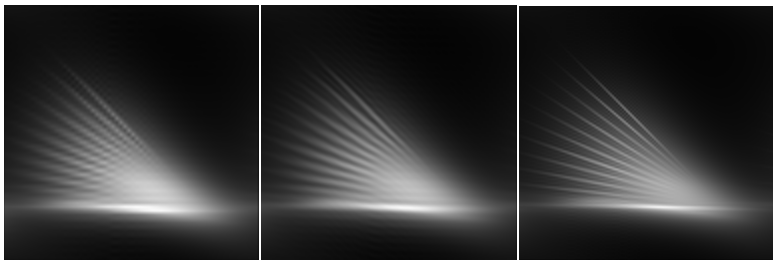

Fig. 5. Close-up of left portion of Blunt Fin: Zooming into a pre-computed 64×64 X-ray image reveals aliasing artifacts (left). Zero-padding in frequency space reduces aliasing (center), but by directly computing the 2D frequency spectrum (right), our approach achieves high resolution magnification without artifacts (sharp features are preserved).

We now present rendering and timing results on various anisotropic datasets and show several applications including zooming (magnification) and frequency-sensitive X-ray rendering. We show only timing results for directly computing 2D frequency spectrums since 3D frequency spectrums are computed slice-by-slice using our GPU algorithm for 2D slices. As previously noted, the color in the X-rays is generated by applying a transfer function to the resulting 2D X-ray images, not to the original volumetric dataset. We briefly describe the irregularly sampled datasets.

4.1 Data Sets

We apply continuous Fourier Volume Rendering to three types of irregularly sampled datasets that require anisotropically scaled kernels. The SPX (4,011 pts), Combustion Engine (46,805 pts), and Blunt Fin (15,256 pts) datasets were generated by fitting elliptical basis functions to a local Delauney triangulation of the dataset [9]. The computational fluid dynamics (CFD) data is a volume of particle concentrations from the New York Harbor area acquired from the New York Harbor Observing and Prediction Systems (NYHOPS). The CFD data (3,652 pts) shown in Figure 6 is a close-up of the entrance to the New York harbor. The Ghiradelli Square

dataset (2,762 pts) came from voxel coloring [6] and anisotropic RBF fitting based on principle components analysis within a neighborhood of each data point [7].

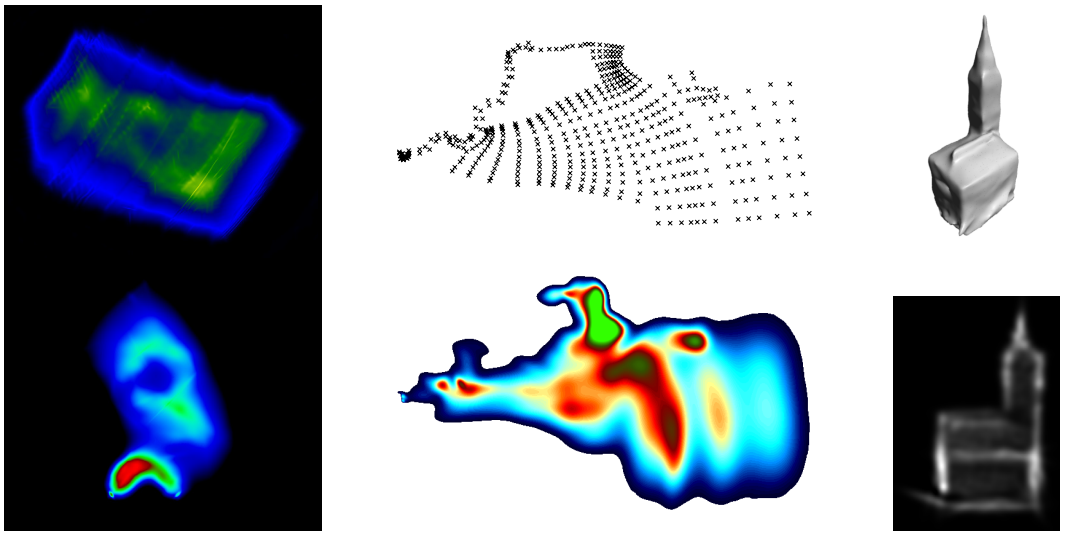

Fig. 6. Continuous FVR of irregularly sampled datasets: Combustion Engine (top-left), SPX (bottom-left), CFD particle concentrations (center), and Ghiradelli Square (right). Input data for CFD data and Ghiradelli Square are shown above their X-rays.

4.2 X-rays and Timing Results

Visually, our X-ray results are similar to splatting and more continuous than Non-uniform DFT because it allows X-rays of arbitrary resolution to be generated. NDFT does not handle anisotropic kernels, and sampling artifacts (smoothed over gaps between data points) are apparent in the reconstruction results [21].

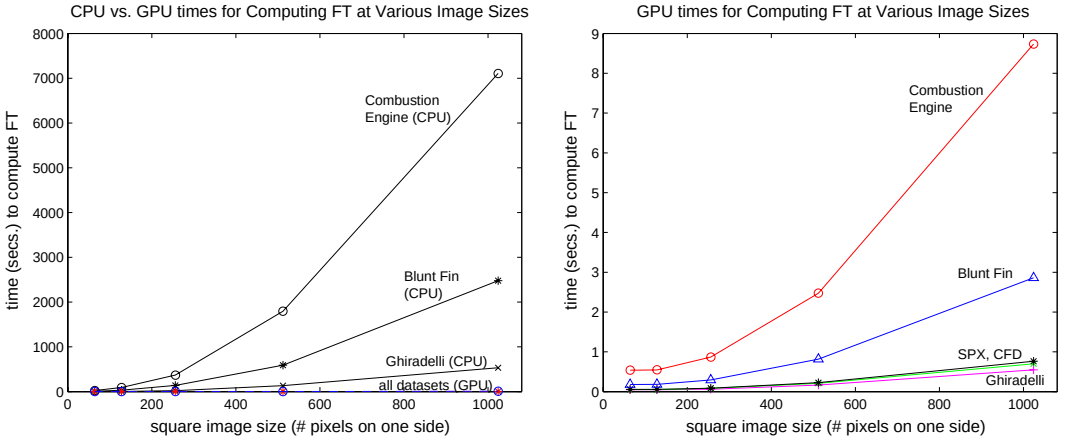

Fig. 7. Left: CPU and GPU times for computing a 2D frequency spectrum of anisotropic datasets. Right: Close-up of GPU times for all datasets.

An advantage of our algorithm is that it is highly amenable to computation on the GPU (GPU times are orders of magnitude faster than CPU times). In Figure 7, we compare the time to compute a 2D slice of the frequency spectrum on the CPU

with the time to compute it on the GPU (a close-up of GPU times for the different datasets are shown on the right). With an Nvidia GeForce 8800 GT and at an image resolution of 256 $\times$ 256, we achieve 3.4 fps on the Blunt Fin dataset with 15,256 data points and 1.2 fps on the Combustion Engine dataset with 46,805. CPU times are not under 1 second for any of the datasets at the same image resolution. The plots also show the nearly linear relationship between rendering time and total image size (in the plots, image size is indicated by the number of pixels along one side of a square image, not the total 2D image size). As with all GPU algorithms, ours will become more interactive with future generations of graphics cards.

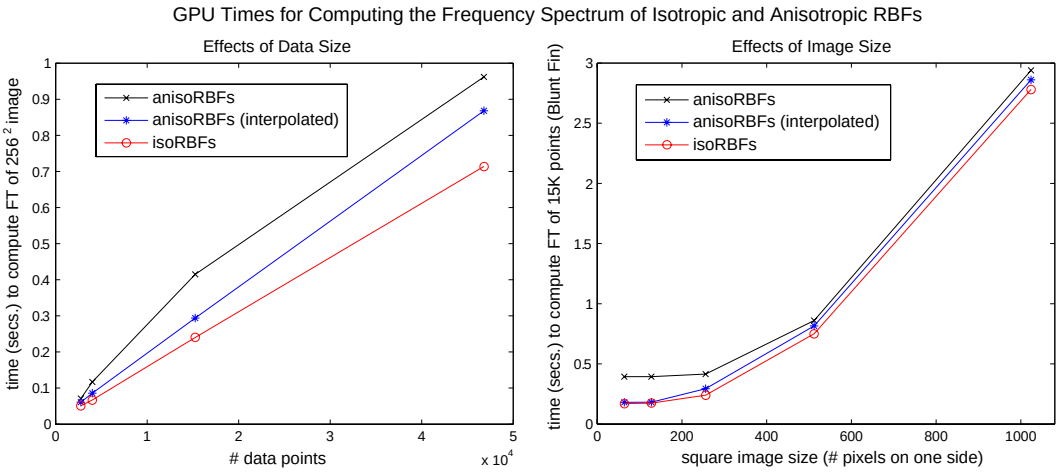

Fig. 8. GPU times as a function of number of data points (left) and image size (right) for computing a 2D frequency spectrum using isotropic and anisotropic RBFs and an optimized GPU algorithm (via interpolation of texture coordinates).

In Figure 8, we compare timing results recorded on the Nvidia GeForce 8800 GT for computing the frequency spectrum on the GPU for isotropic RBFs and anisotropic RBFs using both the unoptimized (per fragment computation of anisotropically scaled texture coordinates) and optimized (interpolated texture coordinates) GPU algorithms. Actual timing data for rending 512 $\times$ 512 images are recorded in Table 1. As expected, computing the frequency spectrum using homogeneous, isotropic RBFs is fastest because of the simplification provided by this less general formulation, and the optimized GPU algorithm for anisotropic RBFs is faster than the unoptimized algorithm. Interestingly, the difference in rendering time between the three GPU algorithms remains almost constant regardless of image size or number of data points, particularly as we go beyond 10,000 data points and images greater than 256 $\times$ 256 in size. This is in contrast to the plots of Figure 7 which show that the reduction in rendering time achieved by the GPU over the CPU becomes more significant as image and data size grows. Hence, the more accurate, per fragment, anisotropic scaling of texture coordinates is scalable and is not much more expensive at large image and data sizes.

Although GPU-accelerated splatting remains faster than GPU-enabled continuous FVR, there are two key benefits of continuous FVR over splatting: (1) signal pro-

Dataset	#points	CPU	GPU-Isotropic	GPU	GPU-optimized
Fuel	13,731	205	0.68	-	-
Neghip	121,586	1,869	5.89	-	-
Foot	4,854,701	23,641	234.66	-	-
Ghiradelli Square	2,762	135	0.15	0.17	0.16
CFD	3,652	170	0.19	0.21	0.21
SPX	4,011	197	0.20	0.23	0.23
Blunt Fin	15,256	591	0.75	0.86	0.82
Combustion Engine	46,805	1,795	2.27	2.51	2.47

Table 1
Rendering times (seconds) for 512×512 images. The first 3 datasets are regularly sampled and reconstructed using homogeneous, isotropic RBFs. In the last two columns, X-rays are rendered on the GPU using anisotropic RBFs which vary in scale from point to point.

cessing operations such as filtering, minification, and magnification are simplified because they can be applied in the frequency domain, and (2) continuous FVR provides greater flexibility with respect to the degree of anisotropy applied to RBFs. In [18], a GPU algorithm was developed for image aligned splatting of elliptical kernels which relies on transforming the general ellipsoid to a unit sphere. Our approach can be applied to RBFs that have undergone arbitrary transformations because the transformations are applied to texture coordinates used to sample the texture storing the FT of the isotropic RBF (Φ). This transformation may be an affine transformation represented by a 4×4 transformation matrix, or it can be a non-linear transformation stored in a displacement texture. In addition to these benefits, continuous FVR maintains high resolution at arbitrary zoom factors because the frequency spectrum can be explicitly sampled at any resolution, unlike conventional FVR which requires re-computing the 3D frequency spectrum at the desired resolution for high fidelity zoom-in.

Splatting exploits the locality of RBFs in the spatial domain, and hence, the computation time depends significantly on the size of the ellipsoid. This dependence is evident in the low framerates for datasets with highly stretched ellipsoids (Blunt Fin and Combustion Engine) versus better framerates for larger but more regular datasets [18]. In our approach, the frequency domain shift factor ($e^{-j2\pi\omega\vec{c}_i}$) required to sample the continuous frequency spectrum of shifted RBFs has infinite support. Thus, the Fourier Transform of every RBF must be evaluated for each frequency sample (leading to the slower performance when compared with splatting). We found early discarding of fragments ineffective (few fragments were being discarded), and the additional branching resulted in even longer compute times. As a result, the degree of anisotropy of the dataset does not affect our runtime which is linearly dependent on the size of the dataset and X-ray image that is generated.

4.3 Filtering, Zooming, and Frequency-sensitive X-rays

Once we have a frequency spectrum of the dataset, we can apply basic signal processing before restoring the data to the spatial domain via the inverse DFT. Typical low, band, and high-pass operations are implemented via a multiplication of the frequency spectrum with a box filter which retains the desired frequencies and zeros out the remaining frequencies. Depending on the filtering or zooming operation, this can be achieved on the graphics hardware by either scaling up or down the texture coordinates of the spectral footprint of the RBFs (stored as a texture) or the coordinates of the quadrilateral to which the texture is mapped. For low-pass filtering, the high frequency information must be clamped to zero while still retaining the same frequency range. This translates to texture mapping only the low frequency portion of the spectrum to the quadrilateral. For band or high-pass filtering, the low frequencies are clamped to zero by multiplying an inverse Gaussian to the frequency spectrum to avoid ringing which would occur if a black quad is simply drawn over the low frequencies. Figure 9 shows low and high-pass filtering of the Combustion Engine dataset.

For minification (zoom-out) and magnification (zoom-in), the range of the frequency spectrum needs to be scaled. We achieve this by scaling the quadrilateral to which the full spectral footprint of the RBFs are drawn (scale up for zooming-out and scale down for zooming-in). When zooming-in, we can apply zero-padding to the scaled-down spectrum to reduce aliasing or completely re-compute the 2D spectrum to display high-resolution detail without artifacts as shown in the center and right panels of Figure 5.

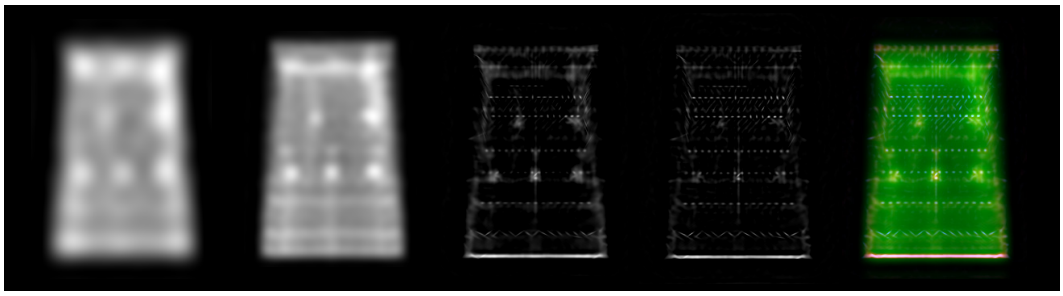

Fig. 9. Left to right: Low to high-pass filtering of the Combustion Engine. Far right: Color is applied based on frequency band instead of opacity for a frequency-sensitive transfer function.

With a frequency spectrum, we are able to generate *frequency-sensitive* X-rays by applying a transfer function based on frequency bands rather than opacity values. We do so by transforming each band back into the spatial domain via the inverse DFT, applying a transfer function to the resulting filtered X-ray, and blending all X-ray bands into a final image. An example is shown in Figure 9 (far right) where high-frequency details (sharp features) are high-lighted.

5 Conclusions and Future Work

We have described a continuous FVR algorithm for irregularly sampled datasets that uses the continuous Fourier Transform of anisotropic RBFs to generate X-rays of arbitrary resolution. With a continuous frequency spectrum, we are able to easily filter, minify and magnify the dataset, and generate frequency-sensitive X-rays. We note that although our sample datasets are composed of elliptical basis functions, our method can be applied to RBFs that have undergone arbitrary transformations.

In future work, we will investigate how shading and depth cues can be incorporated into continuous FVR. Such cues are already possible in discrete FVR [23]. The Fourier Transform of the gradient magnitude of RBFs is quite complex, however, and this is the primary obstacle. For example, the FT of the gradient magnitude of Gaussians requires the Kummer function. Finally, the RBFs and their Fourier Transforms are scalable to higher dimenions. We will explore how continuous FVR can be applied to time-varying 3D data in future work.

References

[1] M. Artner, T. Möller, I. Viola, M. Gröller, High-quality volume rendering with resampling in the frequency domain, TCVG Symp. on Visualization (2005) 1–9.

[2] E. Brigham, The Fast Fourier Transform and Its Applications.

[3] M. Buhmann, Radial Basis Functions.

[4] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, Reconstruction and representation of 3d objects with radial basis functions, SIGGRAPH.

[5] A. Corrigan, J. Wallin, Visualization of meshless simulations using fourier volume rendering, ECCOMAS Conf. on Meshless Methods.

[6] W. Culbertson, T. Malzbender, G. G. Slabaugh, Generalized voxel coloring, Vision Algorithms: Theory and Practice. (LNCS) 1883 (1999) 67–74.

[7] H. Dinh, G. Turk, G. Slabaugh, Reconstructing surfaces using anisotropic basis functions, IEEE Intl Conf on Computer Vision (ICCV) (2001) 606–613.

[8] S. Dunne, S. Napel, B. Rutt, Fast reprojection of volume data, Proc. of the 1st Conference on Visualization in Biomedical Computing (1990) 11–18.

[9] W. Hong, N. Neophytou, K. Mueller, A. Kaufman, Constructing 3d elliptical gaussians for irregular data, Math Found. of Sci. Visualization, Computer Graphics, and Massive Data Exploration.

[10] Y. Jang, M. Weiler, M. Hopf, J. Huang, D. Ebert, K. Gaither, Interactively visualizing procedurally encoded scalar fields, Proc. of EG/IEEE TCVG Symposium on Visualization VisSym (2004) 35–44.

[11] P. Lacroute, M. Levoy, Fast volume rendering using a shear-warp factorization of the viewing transformation, SIGGRAPH 94 (1994) 451–458.

[12] M. Levoy, Volume rendering using the fourier projection-slice theorem, SIGGRAPH (1992) 61–69.

[13] A. Li, K. Mueller, T. Ernst, Methods for efficient, high quality volume resampling in the frequency domain, Proc. of IEEE Visualization (2004) 3–10.

[14] T. Malzbender, Fourier volume rendering, ACM Trans. on Graphics (TOG) 12 (3) (1993) 233–250.

[15] F. Marvasti, Nonuniform Sampling: Theory and Practicse.

[16] B. Morse, T. Yoo, P. Rheingans, D. Chen, K. Subramanian, Interpolating implicit surfaces from scattered data using compactly supported radial basis functions, Shape Modeling Intl (2001) 89–98.

[17] N. Neophytos, K. Mueller, Gpu accelerated image aligned splatting, Proc. of Int'l. Workshop on Volume Graphics (2005) 197–205.

[18] N. Neophytou, K. Mueller, K. McDonnell, X. G. W. Hong, H. Qin, A. Kaufman, Gpu-accelerated volume splatting with elliptical rbfs, IEEE-TVGC Eurographics EuroVis 2006.

[19] G. Nielson, Scattered data modeling, Computer Graphics and Applications (1993) 60–70.

[20] V. Savchenko, A. Pasko, O.G.Okunev, T.L.Kunii, Function representation of solids reconstructed from scattered surface points and contours, Computer Graphics Forum 14 (4) (1995) 181–188.

[21] P. Stark, Fourier volume rendering of irregular data sets, MS Thesis, Simon Fraser University.

[22] J. Sweeney, K. Mueller, Shear-warp deluxe: The shear-warp algorithm revisited, IEEE TCVG Symposium on Visualization 2002 (2002) 95–104.

[23] T. Totsuka, M. Levoy, Frequency domain volume rendering, SIGGRAPH 27 (1993) 271–278.

[24] I. Viola, A. Kanitsar, M. Gröller, Gpu-based frequency domain volume rendering, Proc. of the 20th Spring Conference on Computer graphics (2004) 55–64.

[25] M. Weiler, R. Botchen, S. Stegmaier, T. Ertl, J. Huang, Y. Jang, D. Ebert, K. Gaither, Interactively visualizing procedurally encoded scalar fields, IEEE CG&A 25 (5) (2005) 72–81.