Methods for Efficient, High Quality Volume Resampling in the Frequency Domain

Aili Li [*] Klaus Mueller [†] Thomas Ernst [‡]
Center for Visual Computing, Computer Science Department
Stony Brook University
Medical Department
Brookhaven National Laboratory

ABSTRACT

Resampling is a frequent task in visualization and medical imaging. It occurs whenever images or volumes are magnified, rotated, translated, or warped. Resampling is also an integral procedure in the registration of multi-modal datasets, such as CT, PET, and MRI, in the correction of motion artifacts in MRI, and in the alignment of temporal volume sequences in fMRI. It is well known that the quality of the resampling result depends heavily on the quality of the interpolation filter used. However, high-quality filters are rarely employed in practice due to their large spatial extents. In this paper, we explore a new resampling technique that operates in the frequency-domain where high- quality filtering is feasible. Further, unlike previous methods of this kind, our technique is not limited to integer-ratio scaling factors, but can resample image and volume datasets at any rate. This would usually require the application of slow Discrete Fourier Transforms (DFT) to return the data to the spatial domain. We studied two methods that successfully avoid these delays: the chirp-z transform and the FFTW package. We also outline techniques to avoid the ringing artifacts that may occur with frequency-domain filtering. Thus, our method can achieve high-quality interpolation at speeds that are usually associated with spatial filters of far lower quality.

CR Categories: I.4.3 [Image Processing]: Filtering, Enhancement; I.4.5 [Image Processing]: Transform Methods; I.4.10 [Image Processing]: Volumetric Image Representation; I.3.0 [Computer Graphics]: General

Keywords: resampling, filters, Fourier Transform

1 INTRODUCTION

The resampling of images and volumes is a frequently occurring task. A few examples are:

- Magnification (or upsampling), either for local zooms (as a magnifying glass) within a region of interest or for global enlargements.
- Minification (or downsampling), for general size reductions, image pyramids, mip-maps, and others.
- Interpolation of additional data along one or more axis directions, such as for the interpolation of additional slices in anisotropically acquired MRI or CT datasets.
- Rotation of images and volumes, required for the registration of multi-modal, multi-volume datasets, and for some volume renderers [9].
- Regular warps, such as perspective warps, for post-warp volume rendering with parallel rays.

The fidelity of the resampled image or volume is highly dependent on the quality of the resampling filter used in the process. Many evaluations of interpolation filters have become available over the years, both in terms of their quality and their computational effort [15, 17, 18, 21, 23], and a history of interpolation since the Babylonian times has been collected by Meijiring [16]. Generally, the wider the filter in the spatial domain, the better its quality (assuming proper parameter tuning). For example, a cubic convolution filter of half width=2 tends to give better interpolation results than a linear filter of half width=1. Frequency plots are often used to demonstrate the quality of a filter. Consider Fig. 1, which compares the box, linear, Gaussian, and *sinc* filters. A good filter has all of its energy in the passband (the frequencies below half the sampling rate - the main lobe) and little energy in the stopband (the frequencies above half the sampling rate - the side lobes). While the former reduces blurring effects, the latter limits aliasing. The theoretically best filter is the *sinc* filter, a box in the frequency domain, which has all of its energy in the passband and none in the stopband. The *sinc* filter, however, has infinite width in the spatial domain, and is therefore impractical to use. Marschner and Lobb [15] showed that a windowed *sinc* is a good compromise in that respect.

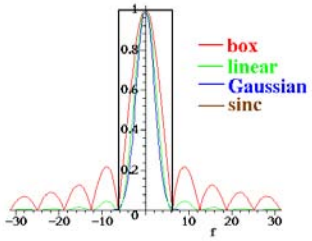


Figure 1: Some spatial interpolation filters in the frequency domain.

Alternatively, filtering can also be performed in the frequency domain. Here, the discrete spatial convolution is replaced by spectral multiplication. For a 1D discrete signal of length N, the cost of the $O(N \lg N)$ FFT (Fast Fourier Transform) and IFFT (Inverse FFT) is amortized if the filter size $W > 2 \lg N$. Thus, for $N = 256$, filters of $W > 16$ are most efficiently implemented in the frequency domain. A good example is computed tomography (CT) where the initial spectral ramp filtering is best transformed in the frequency domain, since for the standard $N = 256$, the width of the spatial-domain counterpart of the filter is much greater than 25 pixels [13]. These relationships extend readily into higher dimensions when the filter is separable, while it gets even more favorable to perform filtering in the frequency domain when the filter is non-separable. One should note, however, that long filters tend to have ringing artifacts in the filtered image. This makes the *sinc* filter, despite its theoretical superiority, a poor filter in practice, especially when strong edges are present in the input image. We shall discuss these issues further in the course of this paper.

[*]Email:aili@cs.sunysb.edu
[†]Email:mueller@cs.sunysb.edu
[‡]Email:ternst@bnl.gov

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

So far, we have only discussed frequency domain filtering as an efficient solution for performing discrete convolution, where the length of the signal remains constant. Discrete convolution may be performed for the purpose of edge filtering, blurring, bandpassing, and others. On the other hand, an image pyramid or mipmap can be created by filtering with a filter at half the image's bandwidth and then performing a subsequent downsampling by a factor of two, dropping every other sample and row. The frequency domain can be employed to perform this bandpassed downsampling efficiently, and this approach is often used in multirate systems [10], in which the sampling rate of a digital signal may be first increased or decreased before processing. Our work promotes the use of very efficient techniques for frequency domain-based processing, that is, the chirp-z transform [2, 22] and the FFTW system [11].

Our framework also proves particularly useful for the rotation of images and volumes, where scanline algorithms have been previously devised as a fast way to perform these operations [5, 12]. Scanline algorithms usually include several passes, and each pass only operates on one dimension of the image or volume. Our frequency-space approach also operates in this manner, and hence it readily applies to the resampling of images and volumes for affine transforms or warps.

Our paper is structured as follows. First, we review related work. Then, in Section 3, we describe the general theory of the methods used, i.e., the chirp-z transform and the FFTW package. In Section 4, we will discuss in detail our approach, in the specific application domain. Section 5 presents results, and Section 6 ends with conclusions.

2 Related Work

Upsampling by an integer factor, say k, can be achieved by transforming the signal into the frequency domain, and creating the $k-1$ intermediate series of samples via $k-1$ phase shifts, each followed by an inverse frequency transform. This method has been used by Chen et al. [6] to interpolate intermediate slices in anisotropically sampled MRI and CT datasets. A similar technique has been used by Unser et al. [27] in conjunction with a scheme that implemented rotations as a sequence of 1D shears, each performed in the frequency domain as phase shifts. Tong and Cox [8, 25] used the *chirp-z transform* [2, 22] to rotate 2D or 3D images. One 2D chirp-z transform, requiring four 2D FFTs, is sufficient for the 2D case [22], and two successive 3D chirp-z transforms requiring six 3D FFTs, are used for the 3D case [2]. All research groups have demonstrated that superior results can be obtained when using these frequency-space techniques over spatial-domain methods.

However, Chen et al. can only produce isotropic datasets if the ratio of anisotropy is an integer number, which is certainly not always the case. On the other hand, Unser et al. cannot scale an image or volume, which may be required when a dataset must be rotated and scaled to fit with the co-registered image of volume, respectively. Finally, Tong and Cox do not take advantage of the efficient decomposition of affine transformations into 1D operations. The limitations for the former two originate in the fact that scaling is equivalent to having more samples over the same spatial extent. Since, for efficiency, an FFT was used to perform the forward transform, requiring N being a power of two, the scaled signal would require an M that is not a power of two, which would no longer allow the use of the FFT for the backward transform. A slower DFT (Discrete Fourier Transform) must then be used instead, which at $O(N^2)$ is much slower than the FFT with $O(N \lg N)$.

The approach presented in our paper eliminates all of these constraints, enabling us to produce an arbitrary number of samples in the output signal. Since for arbitrary scale factors potentially at least one slow DFT is required, it is important to identify a fast implementation of the DFT. We use two approaches to facilitate this. First, we employ the 1D chirp-z transform. This method is, in essence, a clever way to implement a DFT as a series of 4 FFTs, which has lower complexity. More details on implementing the chirp-z transform can be found in Section 3. One drawback of the chirp-z transform is the need of handling multiple long-sequence FFTs, which can be troublesome for the interactive manipulation of larger volumes.

Another solution is to make use of a public-domain package called FFTW [11] (see also www.fftw.org). FFTW is an architecture-adaptive, high performance implementation of the Cooley-Tukey FFT algorithm [7]. It exploits the fact that the performance of a program is mostly determined by complicated interactions of the code with the processor pipeline and by the characteristics of the memory and the cache. But most importantly, due to its performance-driven decomposition of the FFT process into subtasks, sequences of non-power-of-two lengths suffer only little decrease in speed (less than 20%). More details on FFTW will be provided in Section 3, while our results will show that the architecture-aware FFTW proves superior to the complexity-aware chirp-z method, at least in the context of our goals.

In terms of spatial filtering, Keys [14] as well as Park and Schowengerdt [21] showed that the Catmull-Rom spline filter is optimal within the class of cardinal splines. Mitchell and Netravali [17] investigated the more general BC splines, which include cardinal splines, and found that BC splines satisfying $2C+B=1$ gave the best subjective results. These filters include the Catmull-Rom spline, where $B=0$ and $C=0.5$. Möller et al. [18] generalized Keys' method using a Taylor series expansion, by which it could also be demonstrated that the Catmull-Rom spline is the most accurate in the class of cardinal cubic splines. We therefore employ the Catmull-Rom cubic filter, in addition to the other popular, but computationally less expensive, box and linear filters, to compare the results we obtain with our frequency domain resampling approach.

3 Theory

In this section we will provide more details on the two alternative mechanisms available for our purposes, i.e., the chirp-z transform and the FFTW system. We will then compare the prospects of the two in a more theoretical context.

3.1 The Chirp-z Transform

The chirp-z transform was developed relatively early [2, 22] and is related to the fractional Fourier transform [1]. To explain how it works, let us first consider an N-long sequence F in the frequency domain, which we would like to transform into a sequence f of equal length in the spatial domain. This is written as follows:

$$f_k = \sum_{j=0}^{N-1} F_j e^{2\pi i \cdot j \frac{k}{N}} \tag{1}$$

The chirp-z transform generalizes this equation by substituting $1/N$ with a factor α:

$$f_k = \sum_{j=0}^{N-1} F_j e^{2\pi i \cdot jk\alpha} \tag{2}$$

where α can be any rational or even complex number. This parameter allows the output signal to have arbitrary grid spacing. In the general case, this transform must be performed by ways of a (costly)Inverse DFT (IDFT), and the chirp-z transform was introduced as an efficient alternative. It first rewrites $2jk = j^2 + k^2 - (k-j)^2$ and equation (2) becomes:

$$f_k = \sum_{j=0}^{N-1} F_j e^{\pi i \cdot [j^2 + k^2 - (k-j)^2]\alpha}$$

$$= e^{\pi i \cdot k^2 \alpha} \sum_{j=0}^{N-1} F_j e^{\pi i \cdot j^2 \alpha} e^{-\pi i \cdot (k-j)^2 \alpha} = e^{\pi i \cdot k^2 \alpha} \sum_{j=0}^{N-1} y_j z_{k-j} \qquad (3)$$

Here, y_j and z_{k-j} are N-long sequences that are defined as:

$$y_j = F_j e^{\pi i \cdot j^2 \alpha} \quad \text{and} \quad z_j = e^{-\pi i \cdot j^2 \alpha} \qquad (4)$$

The z_j is called the chirp-z signal, since its frequency increases with time, or index j (see the second row of Fig. 2 for an illustration). We observe that the sum in equation (3) is a discrete convolution of the two sequences in equation (4). Since these sequences are both N-long, it is more efficient to use a DFT approach that replaces the convolution by a multiplication in the co-domain. There are, however, two obstacles. First, the sequences y_j and z_j are not necessarily a power of two, which is best for an FFT-based solution (although FFTW would provide an efficient transform even in that case). Second, the z_j does not satisfy the DFT assumption of circular convolution, which expects the signals to be periodic and requires that $z_{k-j} = z_{k-j+N}$. But, in our case, when $(k-j) < 0$, $z_{k-j} = z_{j-k} \neq z_{k-j+N}$, due to the squaring of $(k-j)$ in equation (3). To cope, we can convert the summation into a form that fits the circular convolution requirement. First, we select an integer $p \geq N-1$ for which $2p$ is a power of two, and extend the sequences in (4) as follows:

$$\begin{aligned} y_j &= 0 \qquad N \leq j < 2p \\ z_j &= 0 \qquad N \leq j < 2p-N \\ z_j &= e^{-\pi i \cdot (j-2p)^2 \alpha} \qquad 2p-N \leq j < 2p \end{aligned} \qquad (5)$$

The sequences in equation (5) are illustrated in Fig. 2 below (in this figure, the reason for the $N-M$ inserted zeros will be explained later).

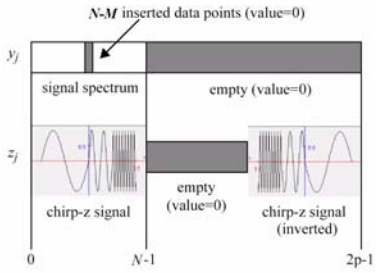


Figure 2: The chirp-z transform sequence. The first row shows the N-long y_j sequence that is kept as is, only padded with zeros .The second row shows the N-long z_j sequence, with the first part being the regular sequence (the chirp-z signal), and the second part being the first part, mirrored.The value $2p$ is chosen to be a power of two.

We observe that z_j now satisfies the circularity criterion. The convolution in equation (3) is then calculated as follows:

$$f_k = e^{\pi i \cdot k^2 \alpha} IFFT[FFT(y_j) \cdot FFT(z_j)] \qquad (6)$$

Both z_j and y_j are transformed with a $2p$-point FFT, the resulting sequences are multiplied bin per bin, and the result is transformed back with a $2p$-point IFFT and multiplied with the complex exponential. Note that we can only make use of the first N samples – the remaining $2p-N$ samples must be discarded.

3.2 The FFTW System

In the FFTW system, a Fourier transform is computed by an *executor*, which consists of highly optimized, composable blocks of C code called *codelets*. A codelet is a specialized piece of code that computes part of the transform. It is generated automatically by a *codelet generator*, which takes the computer's special architecture and configuration into account. The combination of codelets applied by the executor is specified by a special data structure called a *plan*. The plan is determined at runtime, before the computation begins, by a *planner*, which uses a dynamic programming algorithm to find a fast composition of codelets. The planner seeks to minimize the actual execution time, and not the number of floating point operations. To do so, it measures the run time of many plans and selects the fastest. The plan can be reused as many times as needed. If one only needs a single transform of a given size, the one-time cost of the planner becomes a significant cost-factor. But when many transforms of the same size are computed in a typical high-performance application, then the expensive initialization is amortized.

In our case, however, this size may vary. We get around this problem by generating all possible DFT plans beforehand and load these at start-up time. Since we only perform 1D DFTs, this is definitely feasible and dramatically reduces the execution time of operations on images or volumes.

4 Our Approach

In this section we will discuss our contributions. First, we will describe our implemented frequency space-based operators, i.e. 1D scaling and 1D translational shifts. All subsequent resampling applications will be decomposed into these basic 1D operators.

4.1 Scaling

Suppose we have a 1D signal $f(t)$, $0 \leq t \leq T$, sampled into discrete samples f_i, $0 \leq i < M$, spaced apart by a sample distance $\triangle t$. We would like to produce a magnified signal $g(t) = f(at)$, $0 \leq t \leq T/a$ with $0 < a \leq 1.0$, and sample it into a sequence g_i, $0 \leq i < N$, such that $(M-1) = a(N-1)$, and the sample distance being again $\triangle t$. We can look at this as a two-stage process. First, we sample $f(t)$ over the interval T at a rate $a\triangle t$ into the N-long sequence g_i, and then we pull it apart such that the sample distance is $\triangle t$ again (see Fig. 3 for an illustration of this procedure). While the second step is just an application of the Fourier scaling theorem [3], it is the first step that is the interesting one. We shall look at it in closer detail now.

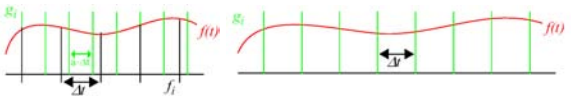


Figure 3: The original signal $f(t)$ which has been sampled into f_i (black), and is now being resampled into g_i (green), left: upsampled $f(t)$, right: $g(t)$ stretched such that the grid values are $\triangle t$ apart.

We want to realize the g_i upsampling using a frequency domain method. We start with an M-point DFT of the f_i which produces the complex series F_j, $0 \leq j < M$, in the frequency domain. We would like to output N samples, however, the DFT produced only M frequency-domain samples, thus $N-M$ samples must be added to the spectrum before transforming back to the spatial domain. But where do these samples go? Since $N > M$, yet we cover the same time interval T, we know that $g(t)$ has a higher sampling rate than $f(t)$. Thus its spectrum has a higher Nyquist rate. Yet, since both $f(t)$ and $g(t)$ have been sampled over the same interval T, the spacing of the samples in the frequency domain is identical. Hence, the $N-M$ samples must be added in the center of the spectrum, right at the fold-over frequency (see again Fig. 2). Further, these samples must be set to zero since their frequencies lie above the bandlimit of $f(t)$. On the other hand, if we want to produce a minified signal

$g(t) = f(at)$, $0 \le t < T/a$ with $a > 1$, then we need to drop $M - N$ samples right in the middle of the spectrum.

We can realize the inverse DFT of the augmented or reduced spectrum either with chirp-z or FFTW. Both will return an equivalent sequence, since neither of them uses approximations (the remaining differences will be due to round-off errors).

4.2 Translation

Translations can also be performed in the frequency domain, via phase shifts. This technique has been described in detail in [6, 27]. Briefly, if we wish to translate a signal $f(t)$ to $f(t-b)$, then the Fourier shift theorem [3] states:

$$F\{f(t-b)\} = F(u)e^{2\pi i \cdot bu} \tag{7}$$

that is, one simply multiplies the function's Fourier transform by the phase term (the exponential), and does the inverse Fourier transform. Accordingly, if we wish to shift an N-long discretized signal f_i by a factor of b, we can achieve this by obtaining its Fourier transform F_i, advancing all basis sinusoids i (i.,e., the phases of the Fourier coefficients F_j) by a factor of $-bi/((N-1)\triangle t)$ via phase shifts and then transforming the result back into the spatial domain. This is equivalent to reconstructing $f(t)$ with a *sinc* filter and resampling the resulting continuous signal according to the shifted grid. Assuming a bandlimited signal, this operation will not exhibit any filter artifacts, such as blurring or aliasing, which would have possibly been introduced with finite spatial filters.

4.3 Windowing and padding

The discrete Fourier transform assumes that the signal is periodic. If the end of the signal is not smoothly related to its start, then we may get discontinuities at the boundaries, leading to aliasing in the frequency domain. We have implemented two ways to deal with this: (1) windowing the signal in the spatial domain by an appropriate window filter, such as a Hanning window, or (2) padding the signal with a sequence that connects the end and the start of the sequence smoothly. We can use a simple linear interpolation between these end points for this. Note that after windowing, the side regions of the signal will be attenuated, compared to the central region, and that we can compensate for this effect by multiplying the signal by the reciprocal of the window function after the resampling has occurred.

Another source of artifacts is spectral leakage, caused by not exactly capturing an integer multiple of a sinusoidal oscillation in the signal. In fact, this is very likely to happen, especially when there is more than one frequency present. Windowing can also be used in this case, as a remedy for spectral leakage.

Finally, we have noticed that in some cases windowing is also required in the frequency domain to prevent ringing artifacts in scaling operations. Scaling in frequency space (without windowing) can be thought of as the following process: (1) the discrete M-long signal is transformed into frequency space, giving rise to a main spectrum with discrete frequencies and its replicas; (2) the replicas are removed by a box filter; (3) the remaining main spectrum is transformed back to generate a continuous signal in the spatial domain, (4) this signal is subsequently sampled to generate the desired N samples. This process is equivalent to convolving f_i with a *sinc* filter (a box in the frequency domain) with period $\triangle t$ and sampling the resulting signal with $a\triangle t$, where a is the scaling factor. No ringing occurs when $a = 1$. However, when $a \neq 1$, then the *sinc* function resulting from each original sample point is not only sampled at its peak and its zero crossings, but also within its non-zero sidelobes. This creates a ripple for each individual original sample point, which is most observable around strong edges [17].

The ringing can be reduced by using a window function on the sequence F_j before insertion of the additional zeros in the center of the spectrum. However, we have noticed that windowing is not needed when the scale factor is small, i.e., less than 2.0. In that case, the ringing does not affect a large, newly interpolated neighborhood, and we do not use the window. If a window must be applied, then ringing will be replaced by blur. We found the Welch window to be a good compromise between blur and ringing.

4.4 Applications

Since the *sinc* filter is separable, the described 1D implementations extend readily to higher dimensions. We simply perform first row-wise filtering in the x-axis, then along the y-axis, and finally, if we are dealing with volumes, along the z-axis. We shall now describe a variety of applications to which we have applied our resampling method. In the following, both chirp-z and FFTW can be used to transform the signal back into the spatial domain. To transform the signal into the frequency domain, either a regular FFT can be used if the signal length is a power of 2, or is extended to be a power of 2, or FFTW can also be employed.

4.4.1 Scaled Rotations

It is well-known that image and volume rotations can be decomposed into a series of 1D shears. Early such decompositions suffered from the so-called bottleneck problem [4, 12], where the contraction in intermediate sheared images complicated the implementation and could cause errors. More recent factorizations [20, 24] do not suffer from bottlenecks. For example, the 2D factorization could be written as follows:

$$\begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix} = \begin{bmatrix} 1 & -tan\frac{\alpha}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ sin\alpha & 1 \end{bmatrix} \begin{bmatrix} 1 & tan\frac{\alpha}{2} \\ 0 & 1 \end{bmatrix} \tag{8}$$

Here the right- and left-most matrices are both x-shears, implemented as unscaled x-shifts ($x' = x + ay$), while the center matrix is a y-shear, implemented as column-wise y-shifts ($y' = y + ax$). Unser et al. [27] used the phase shifting method to implement each of these shifts at *sinc* accuracy, in 2D. The work of Unser et al. [27] does not support scaled rotations, which in turn allows them to use the phase-shift property of the Fourier Transform. However, there are a variety of scenarios in which scaled rotations are necessary. For example, when registering CT with MRI or PET volumes, it is often the case that the MRI or PET volume has not only been acquired at a different orientation, but also at lower resolution. This requires a magnification followed by a rotation. Although we could first use the (accelerated) IDFT to perform a high-quality magnification and then use the method of Unser et al. for the rotation, we can eliminate the latter step with the following factorizations. In 2D the factorization is:

$$\begin{bmatrix} 1 & -tan\frac{\alpha}{2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ sin\alpha & S_y \end{bmatrix} \begin{bmatrix} S_x & S_y \cdot tan\frac{\alpha}{2} \\ 0 & 1 \end{bmatrix} \tag{9}$$

Here, S_x and S_y are the scale factors in x and y, and α is the rotation angle. With S_z being the scale factor in z, the 3D factorization is:

$$\begin{bmatrix} 1 & 0 & 0 \\ G & 1 & H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & A & B \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ E & F & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ C & 1 & D \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \tag{10}$$

where the factors A through H are expressions involving trigonometric functions of the 3D rotation angles α, θ, and φ. More specific detail is available in [5]. Combining scaling and rotation gives:

$$\begin{bmatrix} 1 & 0 & 0 \\ G & 1 & H \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & A & B \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ S_x \cdot E & F & Sz \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ S_x \cdot C & S_y & S_y \cdot D \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

All of these are scaled beam shifts (that is, the data are only shifted along their individual beam axis direction). In our approach, the 2D affine transform can be achieved in two phases, and the 3D affine transform can be achieved in four phases, where each phase is a scaled beam shift. For example, in the 2D case, the first phase is a row-wise scaled *xshift*: $x' = ax + by$, where the factor by is the translation constant and the factor a is the scale constant. Each scaled shift can be implemented by first transforming the spatial domain data into the frequency domain. The second step is, according to the desired number of output samples, adding zeros or dropping some of the highest frequencies in the spectrum. Here, windowing is optional. The third step accomplishes the phase shift. The final step is transforming the data back into the spatial domain.

4.4.2 Interpolation of intermediate slices

Volumes acquired with MRI, CT, confocal microscopy, and electron microscopy are often anisotropically sampled, i.e., their inter-slice distance is larger than their inter-pixel distance. Most of the times, the ratio of these two distances is not an integer value. In order to reduce staircasing artifacts in the volume viewing and rendering stage, it is advantageous to resample these datasets into an isotropic representation. The non-integer voxel cell aspect ratio prevents the phase shift method to be used for accurate resampling. Our method can be readily employed for this purpose. Suppose that the undersampled axis is z and we have N slices along z, then $M = N \cdot \triangle tz / \triangle tx$, where $\triangle tx$ and $\triangle tz$ are the grid spacings in the x (or y) and z axis directions. We then process each z-axis run and generate the M samples using the scaling method. Note that this will generate a complete new set of slices, due to the non-integer aspect ratio. Since none of the original slices will be part of the output data, it is even more important that an accurate resampler is used.

4.4.3 Perspective warps

These warps scale the volume slices at a factor related to the slice-orthogonal direction. Note that there may have been a prior rotation before the warp. The perspective warp, as well as other applications already described, can involve magnification factors of less than unity, i.e., $N < M$. In this case, we remove the $M - N$ central spectral components. Since this corresponds to lowpassing an image with a *sinc* filter prior to lowpassing, it can be advantageous to multiply the resulting spectrum with a window filter to avoid ringing. Our experiments indicated, however, that this was only needed if the removed frequency components had values significantly above zero.

5 Results

5.1 1D Analysis

In the following discussion, we shall refer to our implemented frequency space-based operators as Freq filters. Since the results using the chirp-z transform or FFTW are practically the same, only the execution time is relevant when comparing the two. We performed a quantitative analysis on four popular filters, i.e., box, linear, and cubic, and compared them with our frequency-domain Freq filter. In the cubic filter, $\alpha = -0.5$

$$box(t) = \begin{cases} 1 & \text{if } -0.5 < t \leq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

$$linear(t) = \begin{cases} -t & \text{if } -1 < t < 0 \\ 1.0 - t & \text{if } 0 \leq t < 1.0 \\ 0 & \text{otherwise} \end{cases}$$

$$cubic(t) = \begin{cases} (\alpha+2)|t|^3 - (\alpha+3)|t|^2 + 1 & \text{if } 0 < |t| \leq 1 \\ \alpha|t|^3 - 5\alpha|t|^2 + 8\alpha|t| - 4\alpha & \text{if } 1 < |t| \leq 2 \\ 0 & \text{if } 2 < |t| \end{cases}$$

Since all our filters are separable, we first conducted a quantitative analysis with a 1D signal. For this purpose, we constructed a signal synthesizer that allowed us to assemble any continuous signal as a sum of base frequencies. An example for such a signal is shown in Fig. 4, left. This particular signal is composed of 19 sinusoids with different phases. We sampled this signal just above the Nyquist rate, that is, twice per maximum oscillation. The resulting sample points are shown in Fig. 4, right. We did two experiments, shift and scale. In the shift experiment, we used the existing points to interpolate a new set of points a shift-distance away. Fig. 5, left, shows the result of the interpolation for a shift-distance of half the grid spacing. The black dots in this figure are the correct values of the analytical test signal at the shift-distance. We see that only our Freq filter interpolates these points exactly, while the cubic filter smoothes the signal somewhat and does not reconstruct it correctly. This can be expected since its passband, similar to all other non-*sinc* filters in Fig. 1, attenuates the higher frequencies closer to the Nyquist rate. A similar situation occurs when the signal is magnified, using the existing samples, an example of which is shown in Fig. 5, right. We then evaluated all filters at different shift and scale factors and computed the RMS error of the interpolated signal with the analytical one. The results are shown in Fig. 7. It is obvious that the frequency-space Freq filter excels in this task for all shift and scale factors evaluated.

5.2 Image and Volume Results

We then applied our method to image magnification. To demonstrate the fidelity of the individual filters, we started with a source image (the Lena image) and recursively magnified it by a small amount for 20 iterations. Note each scaling was implemented as scaling in the x direction followed by a scaling in the y direction, exploiting the separability property of the filters. Here, we also compared our filter with the Lanczos filter [26], a popular (spatial) windowed *sinc*-filter. The results of this experiment are presented in Fig. 6, left. We observe that the Freq filter seems to only minimally degrade the image quality, while the competing methods end with diminished results. Here, the box filter forms pixel clusters instead of blur since it does not generate intermediate pixel values. The experiment demonstrates the Freq filter's preservation of the Fourier spectrum, especially in the higher bands. Adding a Welch filter can remove the subtle ringing artifacts that are visible in some areas, however, at the price of some blurring.

Fig. 6, right, illustrates the performance of the Freq filter for the interpolation of extra slices in an anisotropically sampled CT volume dataset. In this dataset, the slice distance was 3.4 times the in-slice sampling rate. We observe that the isotropic volume generated by the Freq filter exhibits the least staircase artifacts.

Fig 9 shows a set of volume rendered images after scaling the engine dataset by a factor of 1.85 in each dimension and storing the result. The parameters used for the volume rendering were identical for all datasets. We observe that the results obtained with the Freq filter have considerably less ripple and noise artifacts than those obtained with the spatial filters. This can be closely observed in the magnifications, shown in Fig. 10 and Fig. 11.

Finally, we also tested the filters with the de-facto test dataset in volume rendering, the Marschner-Lobb (ML) function [15]. The ML function's frequency content is to 99.8% contained within the Nyquist limit and a significant amount of the signal content lies close to the Nyquist limit, positioning it as a challenging test for any filter. We scaled the 40^3 volume dataset in similar ways than the engine dataset, and the volume-rendered results are shown in

Fig. 12. We again observe that the Freq filter produces the least structural artifacts.

5.3 Time Analysis

We will now have a look at the time performance of the Freq filter in relation to its spatial contenders. We have measured the execution time for both the chirp-z transform and FFTW, and found that FFTW is at least 5 times faster than the chirp-z transform when doing image magnification. It is about 10 times faster when performing volume magnification. All the tests were done on a PC with Pentium-4 CPU and 512MB of memory.

FFTW also has a concept called *wisdom*, which can be used to create plans at a faster rate, provided some plans have already been created before. We found that it will usually take about 40 seconds to generate 1000 plans without importing wisdom, while it only takes 2 or 3 seconds if wisdom is present. For our experiments, we pre-created 1024 plans, due to different sequence lengths from $10 - 1024$. After generating or loading the plans, we just need to feed the input data, i.e., the sequence to be transformed, to the corresponding plan. This way, the plan generation time can be ignored, and the total execution time is only due to the resampling effort.

In Fig. 8, we report the collected execution times. On the left, we plot the execution times for scaling the chair volume (size: $64 \times 64 \times 64$) and on the right we plot the times required for the engine volume dataset (size: $128 \times 128 \times 112$). We see immediately that beyond a scale factor of 2.0, the Freq filter (in its FFTW implementation) wins over all spatial filters tested, even the small box filter. This is readily explained since the forward FFT occurs always at the same cost, only the backward FFT is scaled by the magnification factor. This drives down the cost ratio when compared to the spatial domain filters whose costs scale fully with the magnification factor.

6 Conclusions

We have described the use of frequency-domain filters for the accurate resampling of images and volumes at arbitrary magnification factors. In this work, we have tried two methods to implement our frequency domain (Freq) filter – the chirp-z transform and the public-domain FFTW package. Our approach provides a generalization of present frequency-space resampling methods, which cannot handle arbitrary scale factors. While the resampling results obtained with the Freq filter are much better than what can be achieved with common spatial filters, the significant complexity of the multiple FFT passes that are required is a drawback of the filter when implemented using the chirp-z method. However, when using FFTW for both DFT and IDFT, the execution time can be much reduced. In fact, it is even significantly faster than the spatial domain filters when the scaling factor is bigger than 2.0 for volumetric datasets. This shows the potential of frequency-space filtering to even be applied in conjunction with interactive image and volume manipulation applications. We conclude that frequency-space resampling represents a very good choice for high-quality region-of-interest zooms, the interpolation of extra slices in medical datasets, the correction of motion artifacts in MRI, the spatial alignment of temporal volumes in fMRI, and for multi-volume registration. In future work, we plan to investigate the Freq approach also in relation to higher-quality filters, such as those proposed in [18].

Acknowledgements

This research was supported by DOE grant MO-068 and NSF Career Grant ACI-0093157. We would like to thank the anonymous reviewers for their thoughtful comments.

References

[1] D. Bailey and P. Swarztrauber. The fractional Fourier transform and applications. *SIAM Review*, 33(3):389–404, 1991.

[2] L. Bluestein. A linear filtering approach to the computation of the discrete fourier transform. *IEEE Trans. Audio and Electro-Acoustics.*, AU-18, no. 4:451–455, 1970.

[3] R. Bracewell. *The Fourier Transform and Its Applications*. McGraw-Hill Book Company, 1965.

[4] E. Catmull and A. Smith. 3D transformations in scanline order. *Proc. SIGGRAPH'80*, pages 279–285, 1980.

[5] B. Chen and A. Kaufman. 3D volume rotation using shear transformations. *Graphical Models*, 62(4):308–322, 2000.

[6] Q. Chen, R. Crownover, and M. Weinhous. Subunity coordinate translation with fourier transform to achieve efficient and quality three-dimensional medical image interpolation. *Med. Phys.*, 26(9):1776–1782, 1999.

[7] J. Cooley and J. Tukey. An algorithm for the machine computation of the complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[8] R. Cox and R. Tong. Two- and three-dimensional image rotation using the FFT. *IEEE Trans. Img. Proc.*, 8(9):1297–1299, 1999.

[9] R. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Proceedings of SIGGRAPH'88*, 22(4):65–74, 1988.

[10] N. J. Fliege. *Multirate Digital Signal Processing: Multirate Systems - Filter Banks - Wavelets*. Wiley, 2000.

[11] M. Frigo and S.G. Johnson. FFTW: An adaptive software architecture for the fft. *In Proceedings of ICASSP '98(1998)*, 3:1381–1384, 1998.

[12] P. Hanrahan. Three-pass affine transforms for volume rendering. *Computer Graphics (San Diego Workshop on Vol. Vis.)*, 24(5):71–78, 1990.

[13] A. Kak and M. Slaney. *Principles of Computerized Tomography*. IEEE Press, 1988.

[14] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Trans. Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981.

[15] S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. *Proc. IEEE Visualization'94*, pages 100–107, 1994.

[16] E. Meijering. A chronology of interpolation. *Proc. IEEE*, 90(3):319–342, 2002.

[17] D. Mitchell and A. Netravali. Reconstruction filters in computer graphics. *Proc. SIGGRAPH '88*, pages 221–228, 1988.

[18] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a taylor series expansion. *IEEE Trans. Vis. Comp.*, 3(2):184–199, 1997.

[19] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing: 2nd Edition*. Prentice Hall, 1999.

[20] A. Paeth. A fast algorithm for general raster rotation. *Proc. Graphics Interface'86*, pages 77–81, 1986.

[21] S. Park and R. Schowengerdt. Image reconstruction by parametric cubic convolution. *Computer Vision, Graphics, and Image Processing*, 23:258–272, 1983.

[22] L. Rabiner, R. Schafer, and C. Rader. The chirp-z transform algorithm and its applications. *Bell Systems Technical Journal*, 48:1249–1292, 1969.

[23] P. Thvenaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Trans. Medical Imaging*, 19(7):739–758, 2000.

[24] T. Toffoli and J. Quick. Three-dimensional rotations by three shears. *Graphical Models and Image Processing: GMIP*, 59(2):89–95, 1997.

[25] R. Tong and R. Cox. Rotation of NMR images using the 2D chirp-z transform. *Magn. Reson. Med.*, 41:253–256, 1999.

[26] K. Turkowski. *Filters for common resampling tasks, Graphics Gems I*. Academic Press, 1990.

[27] M. Unser, P. Thevenaz, and L. Yaroslavsky. Convolution- based interpolation for fast, high-quality rotation of images. *IEEE Trans. Img. Proc.*, 4(10), 1995.

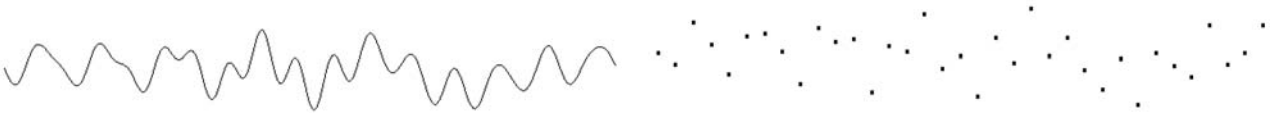


Figure 4: Left: original continuous signal with 19 frequencies. Right: a set of samples sampled at the Nyquist frequency, the number of samples is 38.

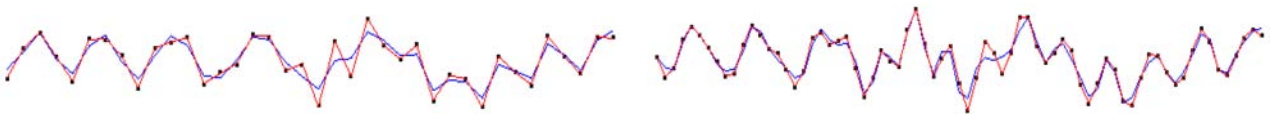


Figure 5: Interpolation from 38 samples per period. Left: the signal is translated by half the sampling period. Right: the signal is magnified by a factor of 1.85. In both graphs, the blue line is due to the cubic filter, while the red line is due to the Freq filter.

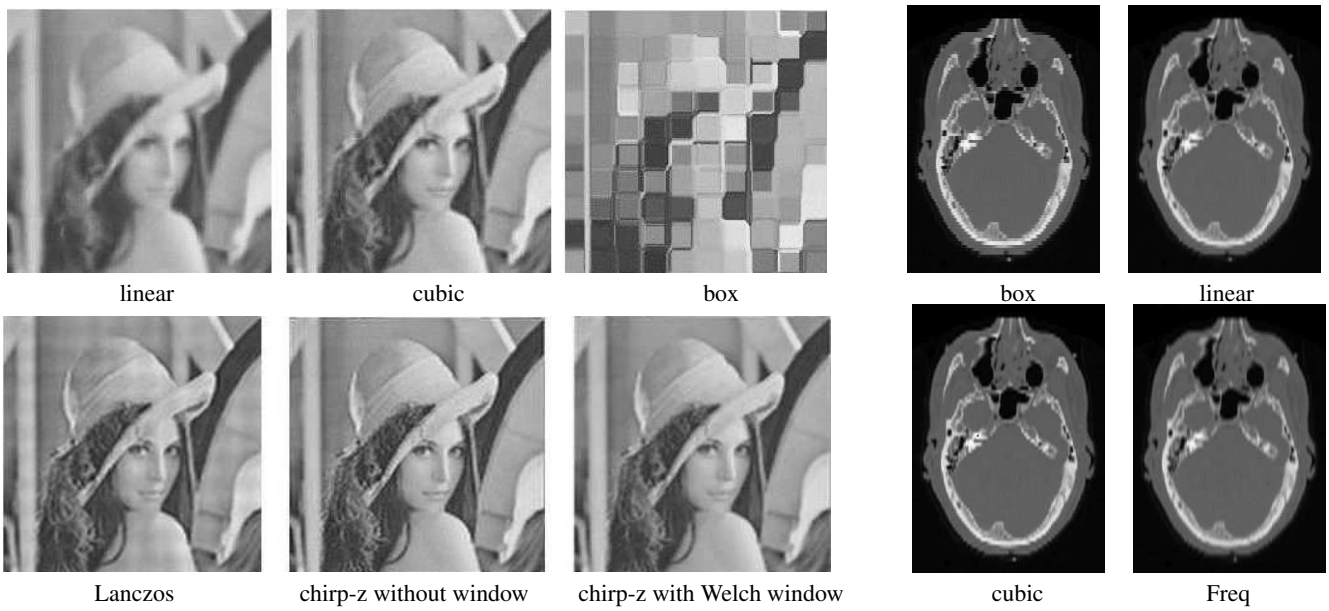


Figure 6: Left: The Lena image after a series of 20 magnifications of 10 pixels each. We observe that the image degradation with the frequency space, Freq, filter is minimal, while all other filters provide deteriorated results. The Freq filter seems to preserve the frequency spectrum well. Using the Welch window in frequency space gives less ringing artifacts, yet the amount of smoothing is small. Right: Interpolation of extra slices at a non-integer sampling rate. The original size of the voxels was $1 \times 1 \times 3.4$. The slices were inserted horizontally (y-direction), with respect to this figure. Shown here is a cut across many inserted slices. We observe that the frequency-transform method produces the least staircase artifacts (but slightly more blurring).

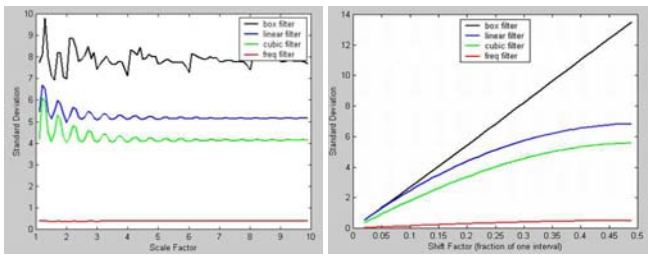


Figure 7: Errors with scale and shift. The RMS error between the original and the interpolated signals, using box, linear, cubic and Freq filter. Left: the behavior related to changing the scale factor from 1.0 to 10.0. Right: the trend when changing the shift factor from 0.0 to 0.5, where the latter means that we sampled right between two original samples.

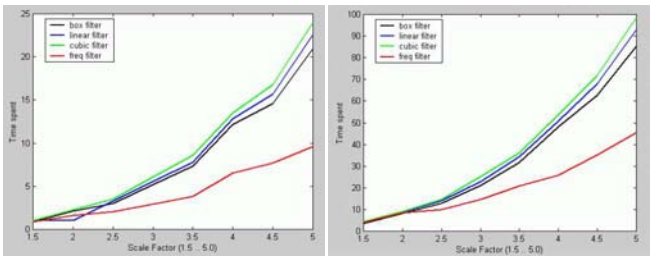
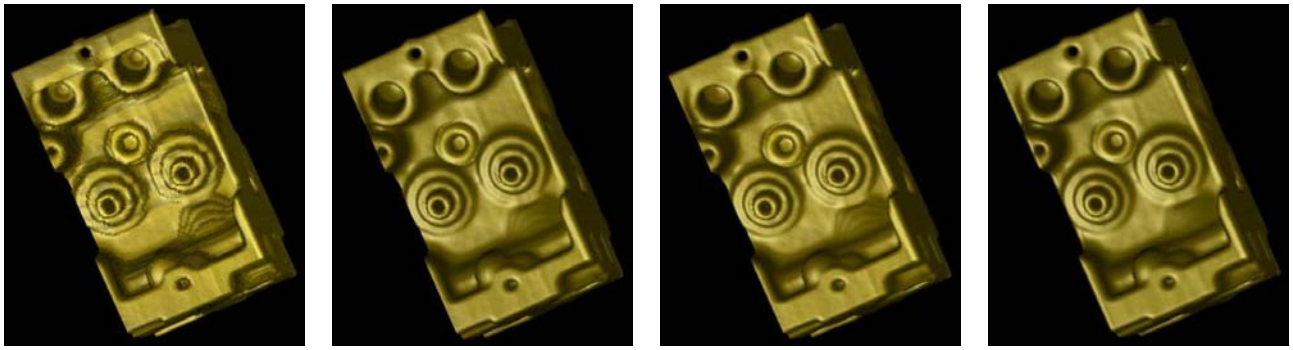
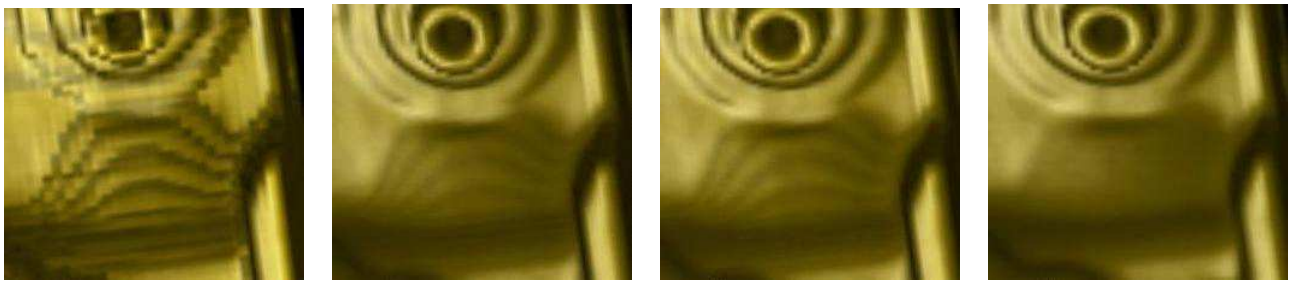


Figure 8: Execution times using the box, linear cubic and Freq filter to perform volume magnifications. Left: the time needed to magnify a $64 \times 64 \times 64$ volume from a factor of 1.5 to 5.0. Right: the time needed to do the same operation on a larger volume, the engine (size: $128 \times 128 \times 112$).



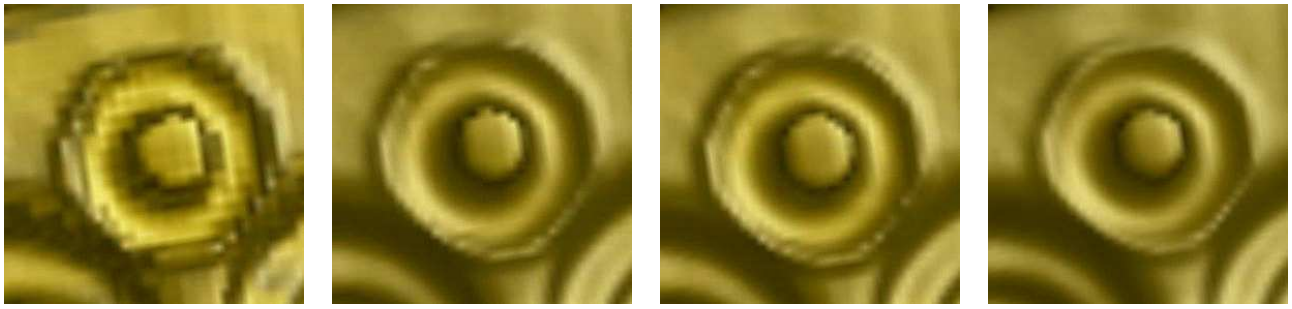
box linear cubic Freq

Figure 9: Results of scaling the engine volume dataset by a factor of 1.85 along each dimension using the box, linear, cubic, and the frequency space, Freq, filters. We notice ripple artifacts at various locations for box, linear, and cubic, which do not exist with Freq.



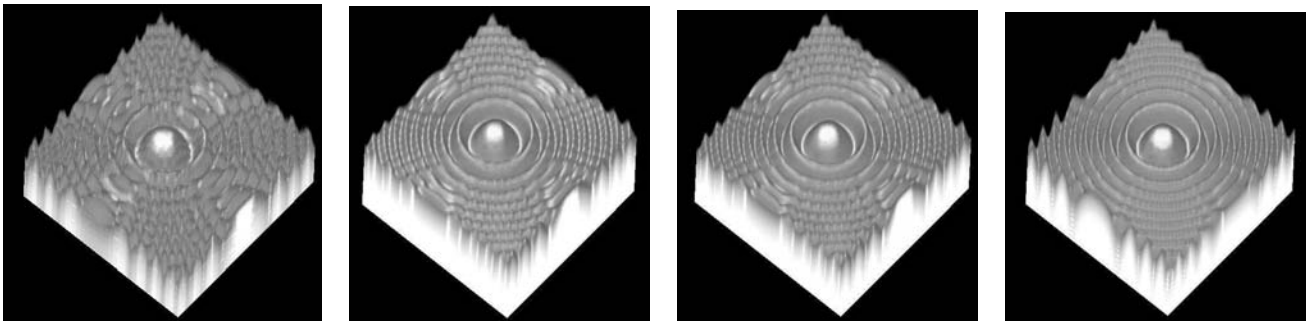
box linear cubic Freq

Figure 10: Magnified view of the volume shown in Figure 9. We observe that the frequency-space method produces no ripple artifacts.



box linear cubic Freq

Figure 11: Another magnified view onto the volume shown in Figure 9. Again, the frequency-space method produces the least artifacts.



box linear cubic Freq

Figure 12: The Marschner-Lobb volume, scaled up with the different filters and volume rendered at an iso-value of 128. Once again, the frequency-space method produces the least artifacts.