DOI: 10.1111/cgf.14753
EUROGRAPHICS 2023 / K. Myszkowski and M. Nießner
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 2

One Step Further Beyond Trilinear Interpolation and Central Differences: Triquadratic Reconstruction and its Analytic Derivatives at the Cost of One Additional Texture Fetch

Balázs Csébfalvi

Budapest University of Technology and Economics, Department of Control Engineering and Information Technology, Hungary

Abstract

Recently, it has been shown that the quality of GPU-based trilinear volume resampling can be significantly improved if the six additional trilinear samples evaluated for the gradient estimation also contribute to the reconstruction of the underlying function [Csé19]. Although this improvement increases the approximation order from two to three without any extra cost, the continuity order remains C^0. In this paper, we go one step further showing that a C^1 continuous triquadratic B-spline reconstruction and its analytic partial derivatives can be evaluated by taking only one more trilinear sample into account. Thus, our method is the first volume-resampling technique that is nearly as fast as trilinear interpolation combined with on-the-fly central differencing, but provides a higher-quality reconstruction together with a consistent analytic gradient calculation. Furthermore, we show that our fast evaluation scheme can also be adapted to the Mitchell-Netravali [MN88] notch filter, for which a fast GPU implementation has not been known so far.

CCS Concepts

• ***Computing methodologies*** → ***Volumetric models;*** *Image processing; Texturing;*

1. Introduction

For several decades, trilinear interpolation has been used as a de facto standard resampling technique for direct volume rendering. Although tensor-product extensions of higher-order interpolation filters are well-known to produce higher image quality, they are rarely used in practice. This is mainly because of their additional computational cost, which increases exponentially in higher dimensions. For example, applying a Catmull-Rom spline interpolation [CR74, Key81] instead of a linear interpolation, the number of original samples to count with is 4, 16, and 64 in one, two, and three dimensions, respectively. In the practice of direct volume rendering [CCF94, WE98, KW03, EHK*06], one can remedy this problem by using fast GPU implementations [SH05, RMtHRS08, Csé18] that evaluate the result of the convolution filtering as a linear combination of trilinear samples, which are provided by the texture-sampling hardware. Following this approach, a tricubic Catmull-Rom spline interpolation can be evaluated from eight trilinear texture samples [Csé18], rather than from 64 nearest-neighbor texture samples. As in current GPUs the cost of a trilinear texture fetch is nearly the same as that of a nearest-neighbor texture fetch, this simplification already leads to a reasonable trade-off between rendering speed and image quality, but only for isosurface rendering. Using, for example, first-hit ray casting for rendering isosurfaces, the cost of the gradient estimation is negligible as gradients need to be evaluated at most once for each ray to calculate the shaded colors of the potentially hit surface points [WE98, HLRSR09]. Nevertheless, in semitransparent direct volume rendering [Lev88, KW03], gradients are necessary to evaluate for each sample position along the rays, which still makes the tricubic Catmull-Rom spline interpolation a very expensive solution. Its combination with on-the-fly central differencing would require 48 additional trilinear texture fetches, which can hardly be considered as a reasonable extra cost for the quality improvement. Currently, a fast GPU-based solution for calculating the analytic derivatives of a Catmull-Rom spline reconstruction is not known. To the best of our knowledge, such a fast analytic derivative-filtering technique has been published only for a tricubic B-spline reconstruction so far [SH05]. However, the estimation of each partial derivative requires eight additional trilinear texture fetches. Therefore, the total number of texture fetches needed for gradient estimation is 24, which is not a reasonable trade-off either.

Recently, it has been shown that, instead of treating the tasks of function reconstruction and gradient estimation completely separately, it is worthwhile to improve the function reconstruction by reusing the same trilinear texture samples that have already been used for gradient estimation [Csé19]. More concretely, the approximation order of the standard linear interpolation was proposed to be increased from two to three by reusing the six additional trilinear samples that need to be evaluated for gradient estimation anyway. Consequently, the quality of the function reconstruction

is improved for free, as new texture fetches are not required and the additional arithmetic operations are performed in the shadow of the texture fetches. Moreover, the function reconstruction and the gradient estimation are more consistent in an approximation-theoretic sense, as they both provide the exact solutions for quadratic polynomials, unlike in case of trilinear interpolation combined with central differencing. Nevertheless, in a strict theoretical sense, the reconstructed gradients are still inconsistent as they are not the analytic gradients of the reconstructed function. Furthermore, in spite of the increased approximation order, the reconstructed function is still only C^0 continuous just as the pure trilinear reconstruction. C^1 continuity is guaranteed only along the grid lines, where the reconstruction is equivalent to a 1D Catmull-Rom spline interpolation. As the impulse response can be considered to be a nonseparable extension of the 1D Catmull-Rom spline, throughout this paper, we will refer to this technique [Csé19] as a Nonseparable Catmull-Rom Spline Interpolation (NCRSI).

In this paper, we propose two triquadratic volume-resampling techniques that use exactly the same set of trilinear samples for function reconstruction and for gradient estimation as well, which ensures the optimal utilization of resources. Compared to NCRSI, the improvements are the following:

- **Consistent gradient estimation:** Unlike NCRSI, which is based on a central-differencing gradient estimation, our techniques provide a fully consistent gradient field, which is the analytic gradient of the reconstructed function.
- **Higher order of continuity:** NCRSI guarantees an approximation order of three, but its order of continuity is just C^0. Our techniques also maintain an approximation order of three, but result in C^1 continuous reconstructed functions. The C^1 continuity stems from the fact that we evaluate separable tensor-product extensions of C^1 continuous quadratic filters, such as the quadratic B-spline or the Mitchell-Netravali notch filter [MN88].
- **Reduced postaliasing:** Mitchell and Netravali proposed a notch filter as a special case of BC-splines [MN88], which has very good antialiasing properties. We show that this filter can be decomposed to a convolution of a discrete filter and a quadratic B-spline filter, where the discrete filter does not increase the number of voxels. Based on this decomposition, we can efficiently evaluate this filter and its analytic derivatives from the same eight trilinear texture samples, in the same way as for a quadratic B-spline reconstruction. We show that the Mitchell-Netravali notch filter significantly reduces the annoying staircase aliasing compared to NCRSI.

Due to these advantageous properties, we think that the fast volume-resampling techniques proposed in this paper are even better candidates for replacing the standard trilinear interpolation than the recently published NCRSI [Csé19].

2. Related Work

Several researchers demonstrated the superiority of higher-order filters over the standard linear interpolation in terms of numerical accuracy and visual quality as well [MN88, ML94, THG00, LME04, MMMY97, MMK*98, BU99b, BTU99, BU99a, CBU05, Csé08]. However, in GPU-accelerated direct volume rendering [CCF94, WE98, KW03, EHK*06], it is relatively difficult to compete with trilinear interpolation. Although it produces lower quality, it is very efficient to evaluate due to its hardwired implementation. Therefore, in many practical applications, it is still considered to be a better trade-off between image quality and rendering speed than any higher-order filtering technique mentioned above.

The first higher-order volume resampling technique optimized especially for the GPU was published in 2005 by Sigg and Hadwiger [SH05]. They recommended a fast evaluation of a cubic B-spline filtering as a linear combination of eight trilinear texture samples. Meanwhile, this classical method was also proposed to be further optimized [RMtHRS08] and adapted to a fast Catmull-Rom spline filtering [Csé18]. However, combined with derivative estimation schemes evaluating either central differences or the analytic derivative filters, these techniques become too expensive computationally. Therefore, they are more suitable for isosurface rendering rather than for semitransparent direct volume rendering. To overcome these difficulties, NCRSI [Csé19] was proposed as an efficient two-in-one solution for both function reconstruction and gradient estimation. NCRSI is able to exactly reconstruct quadratic surfaces and their normals from their discrete volumetric representations at the cost of a lower-quality trilinear interpolation combined with on-the-fly central differencing. In fact, using NCRSI, a Catmull-Rom spline interpolation is applied along the edges of the cubic cells, and inside the cells the underlying function is reconstructed as a polynomial that fits onto the edge profiles. This approach guarantees a C^1 continuous reconstruction along the grid lines, but provides only C^0 continuity at the faces of the cubic cells. Therefore, in this paper, we propose a slightly slower volume resampling that results in a C^1 continuous triquadratic B-spline reconstruction. Compared to the seven texture fetches required by NCRSI, we take eight texture samples, which are used for both reconstructing the underlying function and for calculating the analytic derivatives of the reconstruction. Thus, at the cost of only one more texture fetch, a higher-quality volume resampling can be achieved.

Our method is similar to the technique published by Sigg and Hadwiger [SH05] (see a brief summary in Section 2.1), but instead of a tricubic filtering we use a triquadratic filtering. Although, in this way, we reduce both the order of continuity and the approximation order by one, in Section 3.1, we show that a triquadratic filtering is still more appropriate for semitransparent volume rendering as the analytic gradients of the triquadratic reconstruction can be evaluated for free without taking any extra trilinear texture samples.

2.1. Fast Tricubic B-Spline Filtering

In 1D, a cubic B-spline reconstruction at position x is calculated as a normalized weighted sum of four neighboring samples f_{i-1}, f_i, f_{i+1}, f_{i+2}:

$$f_{\text{cub}}(x) = f_{i-1}w_0(\alpha) + f_i w_1(\alpha) + f_{i+1}w_2(\alpha) + f_{i+2}w_3(\alpha), \quad (1)$$

where $x = i + \alpha$ for $i \in \mathbb{Z}$ and $\alpha \in [0,1)$ being the integer and fractional parts of x, respectively. The weights corresponding to the

cubic B-spline are defined as follows:

$$\begin{aligned} w_0(\alpha) &= \tfrac{1}{6}\left(-\alpha^3+3\alpha^2-3\alpha+1\right), \\ w_1(\alpha) &= \tfrac{1}{6}\left(3\alpha^3-6\alpha^2+4\right), \\ w_2(\alpha) &= \tfrac{1}{6}\left(-3\alpha^3+3\alpha^2+3\alpha+1\right), \\ w_3(\alpha) &= \tfrac{1}{6}\alpha^3. \end{aligned} \tag{2}$$

The hardwired linear interpolation can be exploited if $f_{\text{cub}}(x)$ is evaluated as a linear combination of the following two terms [SH05]:

$$\begin{aligned} f_{\text{cub}}(x) = \\ g_0(\alpha)\cdot \underbrace{\left(f_{i-1}\frac{w_0(\alpha)}{g_0(\alpha)}+f_i\frac{w_1(\alpha)}{g_0(\alpha)}\right)}_{\text{linear interpolation at } i-1+\frac{w_1(\alpha)}{g_0(\alpha)}} + \\ g_1(\alpha)\cdot \underbrace{\left(f_{i+1}\frac{w_2(\alpha)}{g_1(\alpha)}+f_{i+2}\frac{w_3(\alpha)}{g_1(\alpha)}\right)}_{\text{linear interpolation at } i+1+\frac{w_3(\alpha)}{g_1(\alpha)}} \\ = g_0(\alpha)\cdot f_{\text{lin}}\left(i-1+\tfrac{w_1(\alpha)}{g_0(\alpha)}\right)+g_1(\alpha)\cdot f_{\text{lin}}\left(i+1+\tfrac{w_3(\alpha)}{g_1(\alpha)}\right), \end{aligned} \tag{3}$$

where $g_0(\alpha)=w_0(\alpha)+w_1(\alpha)$, $g_1(\alpha)=w_2(\alpha)+w_3(\alpha)$, and $f_{\text{lin}}(x)$ is a linear interpolation at position x. The first term is evaluated as a linear interpolation between f_{i-1} and f_i at position $i-1+\frac{w_1(\alpha)}{g_0(\alpha)}$, while the second term is evaluated as a linear interpolation between f_{i+1} and f_{i+2} at position $i+1+\frac{w_3(\alpha)}{g_1(\alpha)}$. Applying the same evaluation scheme, Sigg and Hadwiger [SH05] proposed to use the analytic derivative of the cubic B-spline for derivative filtering. This also requires two linearly interpolated samples, but at different positions determined by the weights corresponding to the analytic derivative of the cubic B-spline. Thus, in 1D, the total number of linear samples necessary for function and derivative reconstruction is four. Consequently, the 3D tensor-product extension requires eight trilinear samples for the trivariate function reconstruction, while the reconstruction of each partial derivative requires eight additional trilinear samples. Therefore, in 3D, the total number of trilinear samples required for function and gradient reconstruction is 32, which can hardly be competitive to the seven trilinear samples necessary for either NCRSI [Csé19] or the trilinear interpolation combined with on-the-fly central differencing.

3. Fast Evaluation of Triquadratic Reconstructions and their Analytic Derivatives

In this section our major contribution is presented, namely, we propose efficient implementations for two well-known reconstruction filters: the triquadratic B-spline, and the Mitchell-Netravali notch filter [MN88]. Both filters guarantee C^1 continuity and an approximation order of three. In both cases, the reconstruction is evaluated from eight trilinear texture samples, which are reused for calculating the analytic derivatives of the reconstructed function as well. Thus, the derivatives are provided for free. Consequently, our implementations are especially suitable for semitransparent volume rendering, where the gradients for the shading computation need to be estimated for each sample along the rays.

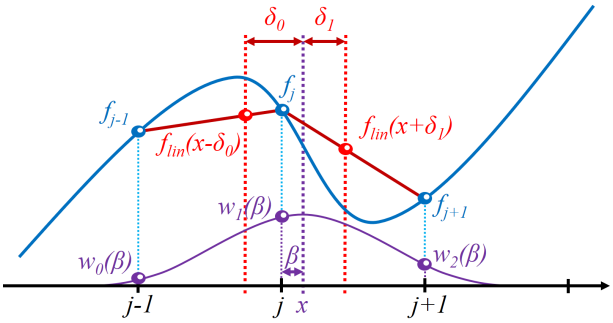


Figure 1: *Illustration of a quadratic B-spline reconstruction using our fast evaluation scheme. The red dots depict the two linear samples that are used for both function reconstruction and for analytic derivative calculation. The blue dots represent the original samples, while the purple dots depict the samples of the quadratic B-spline, which are used for weighting the original samples.*

3.1. Triquadratic B-Spline Filtering

In order to optimize also for the gradient estimation, we propose to reconstruct the underlying function by using a triquadratic B-spline filter rather than a tricubic B-spline filter. In 1D, a quadratic B-spline reconstruction at position x is calculated as a normalized weighted sum of three neighboring samples f_{j-1}, f_j, f_{j+1} (see Figure 1):

$$f_{\text{quad}}(x) = f_{j-1}w_0(\beta)+f_jw_1(\beta)+f_{j+1}w_2(\beta), \tag{4}$$

where $x=j+\beta$ for $j\in\mathbb{Z}$ and $\beta\in\left[-\frac{1}{2},\frac{1}{2}\right)$. Note that $j=\lfloor x+\frac{1}{2}\rfloor$ is now the closest integer to x, unlike in Equation 1, where $i=\lfloor x\rfloor$ is the integer part of x. The weights corresponding to the quadratic B-spline are defined as follows:

$$\begin{aligned} w_0(\beta) &= \tfrac{1}{2}\left(\beta-\tfrac{1}{2}\right)^2, \quad w_1(\beta) = \tfrac{3}{4}-\beta^2, \\ w_2(\beta) &= \tfrac{1}{2}\left(\beta+\tfrac{1}{2}\right)^2. \end{aligned} \tag{5}$$

Similarly to the fast cubic B-spline filtering [SH05], we evaluate $f_{\text{quad}}(x)$ as a linear combination of two terms obtained by two separate linear interpolations:

$$f_{\text{quad}}(x) = g_0(\beta)\cdot f_{\text{lin}}(x-\delta_0(\beta))+g_1(\beta)\cdot f_{\text{lin}}(x+\delta_1(\beta)), \tag{6}$$

where the first and the second linear samples are evaluated at positions $x-\delta_0(\beta)$ and $x+\delta_1(\beta)$, respectively. It is important to note that these two positions are not unique, unlike in case of cubic B-spline filtering discussed in Section 2.1, where the positions for the two separate linear interpolations are completely determined by the weights of the cubic B-spline. In contrast, the result of the quadratic

B-spline reconstruction can be equivalently expressed by different definitions of δ_0, δ_1, g_0, and g_1. For example, a symmetric definition is $g_0(\beta) = g_1(\beta) = \frac{1}{2}$ and $\delta_0(\beta) = \delta_1(\beta) = \frac{1}{4} + \beta^2$, which is easy to check to be a valid solution for Equation 6 being equal to Equation 4. These two equations form an under-determined linear equation system that gives an additional degree of freedom for optimizing offsets δ_0 and δ_1 for an efficient derivative estimation. More precisely, we can determine these offsets such that the corresponding linear samples $f_{\text{lin}}(x - \delta_0(\beta))$ and $f_{\text{lin}}(x + \delta_1(\beta))$ can be used not just for the quadratic B-spline reconstruction, but for calculating the analytic derivative of the reconstructed function as well. This optimization leads to the following solution:

$$\begin{aligned} \delta_0(\beta) &= \tfrac{1}{4} + \tfrac{1}{2}\beta, \quad & \delta_1(\beta) &= \tfrac{1}{4} - \tfrac{1}{2}\beta, \\ g_0(\beta) &= \tfrac{1}{2} - \beta, & g_1(\beta) &= \tfrac{1}{2} + \beta, \end{aligned} \tag{7}$$

Substituting these definitions into Equation 6, we obtain

$$\begin{aligned} & g_0(\beta) \cdot f_{\text{lin}}(x - \delta_0(\beta)) + g_1(\beta) \cdot f_{\text{lin}}(x + \delta_1(\beta)) = \\ & \left(\tfrac{1}{2} - \beta\right) f_{\text{lin}}\left(x - \tfrac{1}{4} - \tfrac{1}{2}\beta\right) + \left(\tfrac{1}{2} + \beta\right) f_{\text{lin}}\left(x + \tfrac{1}{4} - \tfrac{1}{2}\beta\right) = \\ & \left(\tfrac{1}{2} - \beta\right) f_{\text{lin}}\left(j + \beta - \tfrac{1}{4} - \tfrac{1}{2}\beta\right) + \\ & \left(\tfrac{1}{2} + \beta\right) f_{\text{lin}}\left(j + \beta + \tfrac{1}{4} - \tfrac{1}{2}\beta\right) = \\ & \left(\tfrac{1}{2} - \beta\right) f_{\text{lin}}\left(j - 1 + \tfrac{3}{4} + \tfrac{1}{2}\beta\right) + \left(\tfrac{1}{2} + \beta\right) f_{\text{lin}}\left(j + \tfrac{1}{4} + \tfrac{1}{2}\beta\right) = \\ & \left(\tfrac{1}{2} - \beta\right)\left[f_{j-1}\left(\tfrac{1}{4} - \tfrac{1}{2}\beta\right) + f_j\left(\tfrac{3}{4} + \tfrac{1}{2}\beta\right)\right] + \\ & \left(\tfrac{1}{2} + \beta\right)\left[f_j\left(\tfrac{3}{4} - \tfrac{1}{2}\beta\right) + f_{j+1}\left(\tfrac{1}{4} + \tfrac{1}{2}\beta\right)\right] = \\ & f_{j-1} w_0(\beta) + f_j w_1(\beta) + f_{j+1} w_2(\beta) = f_{\text{quad}}(x). \end{aligned} \tag{8}$$

Thus, setting the offsets δ_0 and δ_1 and the weights g_0 and g_1 according to Equation 7, a quadratic B-spline reconstruction can be evaluated from two linear samples indeed based on Equation 6. Now we show that exactly the same linear samples can also be used for analytic derivative filtering. We propose the following derivative-estimation scheme:

$$f'_{\text{quad}}(x) = 2[f_{\text{lin}}(x + \delta_1(\beta)) - f_{\text{lin}}(x - \delta_0(\beta))]. \tag{9}$$

Substituting Equation 7 into Equation 9, we obtain

$$\begin{aligned} & 2[f_{\text{lin}}(x + \delta_1(\beta)) - f_{\text{lin}}(x - \delta_0(\beta))] = \\ & 2 f_{\text{lin}}\left(j + \tfrac{1}{4} + \tfrac{1}{2}\beta\right) - 2 f_{\text{lin}}\left(j - 1 + \tfrac{3}{4} + \tfrac{1}{2}\beta\right) = \\ & 2\left[f_j\left(\tfrac{3}{4} - \tfrac{1}{2}\beta\right) + f_{j+1}\left(\tfrac{1}{4} + \tfrac{1}{2}\beta\right)\right] - \\ & 2\left[f_{j-1}\left(\tfrac{1}{4} - \tfrac{1}{2}\beta\right) + f_j\left(\tfrac{3}{4} + \tfrac{1}{2}\beta\right)\right] = \\ & f_{j-1} w'_0(\beta) + f_j w'_1(\beta) + f_{j+1} w'_2(\beta) = f'_{\text{quad}}(x). \end{aligned} \tag{10}$$

Thus, using the same linear samples $f_{\text{lin}}(x - \delta_0(\beta))$ and $f_{\text{lin}}(x + \delta_1(\beta))$ as for the quadratic B-spline reconstruction, the analytic derivative of the reconstructed function can also be efficiently determined. Analogously, the 3D tensor-product extension of this reconstruction scheme requires the same eight trilinear samples for the trivariate function reconstruction and for the analytic gradient computation as well. As a consequence, compared to trilinear interpolation combined with central differencing, only one additional trilinear texture fetch is necessary, while compared to tricubic B-spline reconstruction combined with analytic gradient filtering (see Section 2.1), the number of necessary texture fetches is reduced by a factor of four. Therefore, our triquadratic B-spline reconstruction scheme represents a very good trade-off between quality improvement and computational cost.

3.2. Mitchell-Netravali Notch Filtering

Mitchell and Netravali introduced a family of BC-spline filters [MN88], which are defined by parameters B and C:

$$\varphi_{\text{BC}}(x) = \frac{1}{6}\begin{cases} (12 - 9B - 6C)|x|^3 + (-18 + 12B + 6C)|x|^2 \\ +(6 - 2B) \text{ if } |x| \le 1, \\ (-B - 6C)|x|^3 + (6B + 30C)|x|^2 \\ +(-12B - 48C)|x| + (8B + 24C) \text{ if } 1 < |x| \le 2, \\ 0 \text{ if } |x| > 2. \end{cases} \tag{11}$$

In order to minimize the postaliasing effect, Mitchell and Netravali proposed parameters $B = \frac{3}{2}$ and $C = -\frac{1}{4}$ resulting in a piecewise quadratic filter:

$$\varphi_{\text{notch}}(x) = \frac{1}{4}\begin{cases} -|x|^2 + 2 \text{ if } |x| \le 1, \\ |x|^2 - 4|x| + 4 \text{ if } 1 < |x| \le 2, \\ 0 \text{ if } |x| > 2. \end{cases} \tag{12}$$

This is a notch filter as its frequency response equals to zero at the Nyquist frequency, which has been shown to be an advantageous property in terms of antialiasing [MN88].

Our key idea for efficiently evaluating the Mitchell-Netravali notch filter is a decomposition of the filter kernel to a convolution of a discrete filter and a quadratic B-spline:

$$\varphi_{\text{notch}}(x) = (p * \varphi_{\text{quad}})(x) = \int_{-\infty}^{-\infty} p(y)\varphi_{\text{quad}}(x - y)dy = \tag{13}$$

$$\frac{1}{2}\varphi_{\text{quad}}\left(x - \frac{1}{2}\right) + \frac{1}{2}\varphi_{\text{quad}}\left(x + \frac{1}{2}\right),$$

where $p(x) = \frac{1}{2}\left[\delta\left(x - \frac{1}{2}\right) + \delta\left(x + \frac{1}{2}\right)\right]$ and the quadratic B-spline is defined as

$$\varphi_{\text{quad}}(x) = \frac{1}{8}\begin{cases} -8|x|^2 + 6 \text{ if } |x| \le \frac{1}{2}, \\ 4|x|^2 - 12|x| + 9 \text{ if } \frac{1}{2} < |x| \le \frac{3}{2}, \\ 0 \text{ if } |x| > \frac{3}{2}. \end{cases} \tag{14}$$

Note that the discrete filtering by p can be performed in a preprocessing, creating new discrete samples a_k by taking the average of each pair of the neighboring original discrete samples: $a_k = (f_k + f_{k+1})/2$. Assuming a periodic extension, the number of

the discrete samples remains the same, but the new samples are associated to sample positions shifted by $\frac{1}{2}$. Exploiting Equation 13, the result of the Mitchell-Netravali notch filtering can be evaluated by simply convolving the new discrete samples a_k by the quadratic B-spline:

$$f_{\text{notch}}(x) = \sum_k a_k \varphi_{\text{quad}}\left(x - \frac{1}{2} - k\right). \tag{15}$$

As we have simplified the Mitchell-Netravali notch filtering to a quadratic B-spline filtering of precalculated samples, we can now use exactly the same fast evaluation scheme to calculate $f_{\text{notch}}(x)$ and its analytic derivatives as for the quadratic B-spline reconstruction (see Section 3.1). Note that, using a 3D tensor product-extension, the Mitchell-Netravali notch filtering is evaluated also from eight trilinear texture samples similarly to the triquadratic B-spline reconstruction, but these trilinear samples are now taken from a precalculated volume, where each voxel is obtained by averaging eight neighboring voxels from the original volume using the 3D extension of the discrete filter p. To the best of our knowledge, the fact that the Mitchell-Netravali notch filtering can be decomposed to a discrete filtering and a consecutive quadratic B-spline filtering has not been recognized so far. Consequently, such a decomposition has not been exploited for a fast implementation yet.

3.3. Prefiltering

It is well known that the quadratic B-spline filter is not interpolating but based on the concept of *generalized interpolation* [BTU99] the original samples can be prefiltered such that the reconstructed function still exactly reproduces the original samples. To do so, instead of the original samples f_k, the prefiltered samples c_k are convolved with the reconstruction filter φ:

$$\tilde{f}(x) = \sum_k c_k \varphi(x-k), \tag{16}$$

where c_k are determined such that the interpolation condition is satisfied:

$$\tilde{f}(j) = \sum_k c_k \varphi(j-k) = f_j. \tag{17}$$

Note that, generalized interpolation does not lead to a loss of information, as it exactly reproduces the original samples at the discrete sample positions just as an interpolating filter applied on the original data. As shown by Blu at al. [BTU99], the prefiltered samples c_k are obtained from the original samples f_k by using the sampled reconstruction filter for a deconvolution prefiltering.

Thus, to make our quadratic B-spline reconstruction interpolating, we have to use the sampled quadratic B-spline as a deconvolution prefilter:

$$q_{\text{quad}}(x) = \varphi_{\text{quad}}(x) \sum_k \delta(x-k) = \frac{1}{8}\delta(x-1) + \frac{3}{4}\delta(x) + \frac{1}{8}\delta(x+1). \tag{18}$$

The deconvolution is easy to implement in the frequency domain as a division by the DTFT of $q_{\text{quad}}(x)$, which is defined as

$$\hat{q}_{\text{quad}}(\omega) = \frac{3+\cos(\omega)}{4}. \tag{19}$$

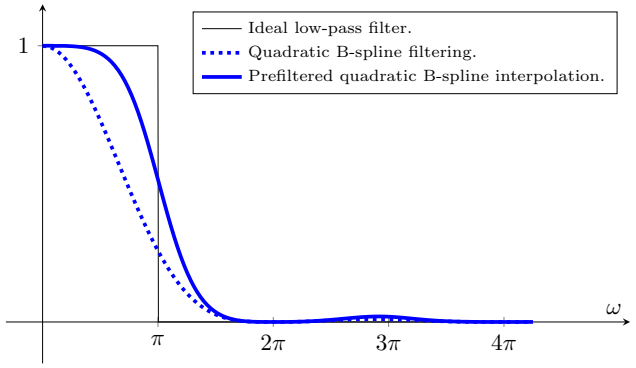


Figure 2: *Magnitudes of the frequency responses $\hat{\varphi}_{quad}(\omega)/\hat{q}_{quad}(\omega)$ and $\hat{\varphi}_{quad}(\omega)$, which correspond to the quadratic B-spline reconstruction with and without prefiltering, respectively.*

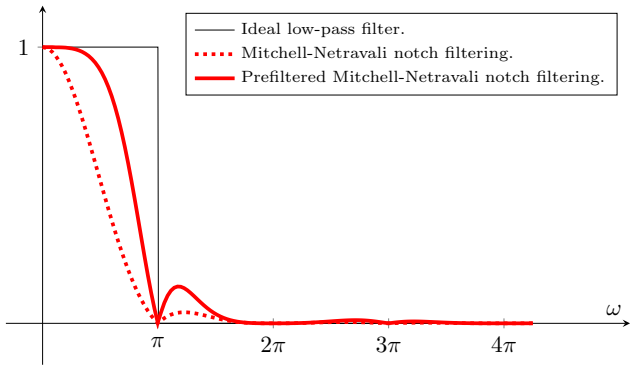


Figure 3: *Magnitudes of the frequency responses $\hat{\varphi}_{notch}(\omega)/\hat{q}_{notch}(\omega)$ and $\hat{\varphi}_{notch}(\omega)$, which correspond to the Mitchell-Netravali notch filtering with and without prefiltering, respectively.*

Using prefiltering, the resultant frequency response is $\hat{\varphi}_{\text{quad}}(\omega)/\hat{q}_{\text{quad}}(\omega)$, where the frequency response of the quadratic B-spline filter φ_{quad} is $\hat{\varphi}_{\text{quad}}(\omega) = \text{sinc}^3(\omega) = \frac{\sin^3(\omega/2)}{(\omega/2)^3}$. Figure 2 shows the frequency responses with and without prefiltering. Note that the prefiltering for generalized interpolation improves the passband behavior, significantly decreasing the oversmoothing effect.

Generalized interpolation cannot be adapted to the Mitchell-Netravali notch filter φ_{notch}, as the DTFT of the sampled filter results in zero-crossings, which would lead to a division by zero. Therefore, we propose a discrete prefilter that optimally exploits the approximation power of φ_{notch} and leads to a quasi-interpolation of order three. The frequency response of φ_{notch} is

$$\hat{\varphi}_{\text{notch}}(\omega) = \cos(\omega/2)\text{sinc}^3(\omega) = \cos(\omega/2)\frac{\sin^3(\omega/2)}{(\omega/2)^3}. \tag{20}$$

Note that $\hat{\varphi}_{\text{notch}}$ fulfills the necessary Strang-Fix conditions [SF71] for a quasi-interpolation of order three, as it guarantees zero-crossings of order three at nonzero integer multiples of 2π. We

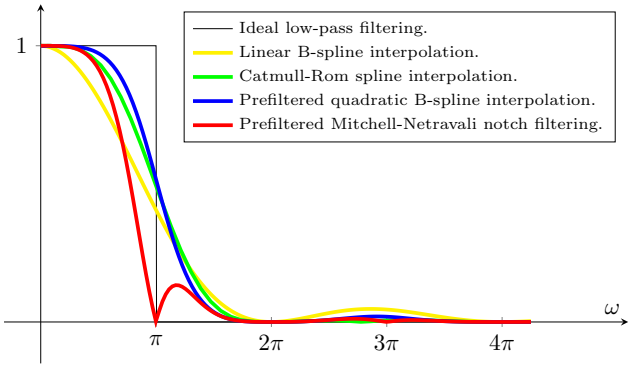


Figure 4: *Frequency responses of the proposed quadratic filters compared to that of the linear B-spline and the Catmull-Rom spline.*

derive a deconvolution prefilter q_{notch} such that a satisfactory condition [CBU05] for a quasi-interpolation of order three is also satisfied:

$$\hat{q}_{\text{notch}}(\omega) = \hat{\varphi}_{\text{notch}}(\omega) + O(\omega^3). \tag{21}$$

We propose to use the deconvolution prefilter previously applied for the interpolating quadratic B-spline reconstruction twice, that is, $q_{\text{notch}} = q_{\text{quad}} * q_{\text{quad}}$ and $\hat{q}_{\text{notch}}(\omega) = \hat{q}_{\text{quad}}(\omega) \cdot \hat{q}_{\text{quad}}(\omega)$. The Taylor series expansions of $\hat{q}_{\text{notch}}(\omega)$ and $\hat{\varphi}_{\text{notch}}(\omega)$ around $\omega = 0$ are

$$\hat{q}_{\text{notch}}(\omega) = 1 - \frac{1}{4}\omega^2 + O(\omega^4), \tag{22}$$

and

$$\hat{\varphi}_{\text{notch}}(\omega) = 1 - \frac{1}{4}\omega^2 + O(\omega^3), \tag{23}$$

respectively. Thus, $\hat{q}_{\text{notch}}(\omega)$ indeed satisfies Equation 21. Therefore, applying the deconvolution filter q_{quad} twice, the Mitchell-Netravali notch filtering can be made quasi-interpolating of order three being equivalent to the interpolating prefiltered quadratic B-spline reconstruction in terms of approximation order. Figure 3 shows the frequency responses corresponding to the Mitchell-Netravali notch filtering with and without prefiltering. Note that the proposed prefilter drastically decreases the oversmoothing effect similarly to the prefilter applied for the quadratic B-spline reconstruction.

Figure 4 shows the frequency responses of the proposed quadratic filters compared to that of the linear B-spline and the Catmull-Rom spline. The standard linear interpolation is the worst in detail-preservation as its frequency response drastically deviates from the ideal passband behavior leading to the highest oversmoothing [ML94]. The prefiltered quadratic B-spline interpolation performs better than the Catmull-Rom spline interpolation, decreasing not just the oversmoothing effect, but also the postaliasing effect [ML94], which stems from the deviation from the ideal stopband behavior. Note that the proposed prefiltered Mitchell-Netravali notch filtering performs the best in terms of antialiasing, though its oversmoothing is slightly higher than that of the prefiltered quadratic B-spline interpolation or the Catmull-Rom spline interpolation.

3.4. 3D Extension

Our techniques can be easily extended to 3D using a separable tensor-product extension. Accordingly, the triquadratic B-spline reconstruction is evaluated as a linear combination of eight trilinear samples as follows:

$$\begin{aligned}
&f_{\text{quad}}(x,y,z) = \\
&g_0(\beta_x)g_0(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y-\delta_0(\beta_y), z-\delta_0(\beta_z)) + \\
&g_1(\beta_x)g_0(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y-\delta_0(\beta_y), z-\delta_0(\beta_z)) + \\
&g_0(\beta_x)g_1(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y+\delta_1(\beta_y), z-\delta_0(\beta_z)) + \\
&g_1(\beta_x)g_1(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y+\delta_1(\beta_y), z-\delta_0(\beta_z)) + \\
&g_0(\beta_x)g_0(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y-\delta_0(\beta_y), z+\delta_1(\beta_z)) + \\
&g_1(\beta_x)g_0(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y-\delta_0(\beta_y), z+\delta_1(\beta_z)) + \\
&g_0(\beta_x)g_1(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y+\delta_1(\beta_y), z+\delta_1(\beta_z)) + \\
&g_1(\beta_x)g_1(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y+\delta_1(\beta_y), z+\delta_1(\beta_z)),
\end{aligned} \tag{24}$$

where $f_{\text{trilin}}(x,y,z)$ is the result of a trilinear interpolation at position $[x,y,z]$, while $\beta_x = x - \lfloor x + \frac{1}{2} \rfloor$, $\beta_y = y - \lfloor y + \frac{1}{2} \rfloor$, and $\beta_z = z - \lfloor z + \frac{1}{2} \rfloor$.

For the evaluation of the partial derivatives of the reconstruction, exactly the same eight trilinear samples can be used. For example, the partial derivative along the x-axis is evaluated by applying our quadratic B-spline filtering (see Equation 6) along the y-axis as well as along the z-axis, while along the x-axis our analytic derivative filtering (see Equation 9) is applied:

$$\begin{aligned}
&\tfrac{\partial}{\partial x} f_{\text{quad}}(x,y,z) = \\
&2[g_1(\beta_x)g_0(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y-\delta_0(\beta_y), z-\delta_0(\beta_z)) - \\
&g_0(\beta_x)g_0(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y-\delta_0(\beta_y), z-\delta_0(\beta_z))] + \\
&2[g_1(\beta_x)g_1(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y+\delta_1(\beta_y), z-\delta_0(\beta_z)) - \\
&g_0(\beta_x)g_1(\beta_y)g_0(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y+\delta_1(\beta_y), z-\delta_0(\beta_z))] + \\
&2[g_1(\beta_x)g_0(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y-\delta_0(\beta_y), z+\delta_1(\beta_z)) - \\
&g_0(\beta_x)g_0(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y-\delta_0(\beta_y), z+\delta_1(\beta_z))] + \\
&2[g_1(\beta_x)g_1(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x+\delta_1(\beta_x), y+\delta_1(\beta_y), z+\delta_1(\beta_z)) - \\
&g_0(\beta_x)g_1(\beta_y)g_1(\beta_z) \cdot f_{\text{trilin}}(x-\delta_0(\beta_x), y+\delta_1(\beta_y), z+\delta_1(\beta_z))].
\end{aligned} \tag{25}$$

The partial derivatives along the y-axis or along the z-axis can be calculated analogously using also the same trilinear samples that are evaluated anyway for the triquadratic B-spline reconstruction.

4. GPU Implementation

Based on our fast evaluation scheme introduced in Section 3.1 and extended to 3D in Section 3.4, a triquadratic B-spline filtering can be implemented using the following GLSL code:

```
vec4 resampleGradientAndDensity(vec3 position)
{
  vec3 scaled_position = position * size - 0.5;
  vec3 beta = scaled_position - round(scaled_position);

  vec3 g0 = 0.5 - beta;
  vec3 delta0 = (0.5 + beta) * 0.5;

  vec3 position0 = position - delta0 / size;
  vec3 position1 = position0 + 0.5 / size;

  vec4 s0 = vec4(
    texture(samplerUnit,
      vec3(position0.x, position0.y, position0.z)).r,
    texture(samplerUnit,
      vec3(position0.x, position1.y, position0.z)).r,
    texture(samplerUnit,
      vec3(position0.x, position0.y, position1.z)).r,
    texture(samplerUnit,
      vec3(position0.x, position1.y, position1.z)).r);

  vec4 s1 = vec4(
    texture(samplerUnit,
      vec3(position1.x, position0.y, position0.z)).r,
    texture(samplerUnit,
      vec3(position1.x, position1.y, position0.z)).r,
    texture(samplerUnit,
      vec3(position1.x, position0.y, position1.z)).r,
    texture(samplerUnit,
      vec3(position1.x, position1.y, position1.z)).r);

  vec4 s_xy0z0_xy1z0_xy0z1_xy1z1 = mix(s1, s0, g0.x);
  vec4 s_dxy0z0_dxy1z0_dxy0z1_dxy1z1 = s1 - s0;
  vec4 s_xyz0_xyz1_dxyz0_dxyz1 = mix(
    vec4(s_xy0z0_xy1z0_xy0z1_xy1z1.yw,
         s_dxy0z0_dxy1z0_dxy0z1_dxy1z1.yw),
    vec4(s_xy0z0_xy1z0_xy0z1_xy1z1.xz,
         s_dxy0z0_dxy1z0_dxy0z1_dxy1z1.xz), g0.y);
  vec2 s_xdyz0_xdyz1 =
         s_xy0z0_xy1z0_xy0z1_xy1z1.yw -
         s_xy0z0_xy1z0_xy0z1_xy1z1.xz;
  vec3 s_xyz_dxyz_xdyz = mix(
    vec3(s_xyz0_xyz1_dxyz0_dxyz1.yw, s_xdyz0_xdyz1.y),
    vec3(s_xyz0_xyz1_dxyz0_dxyz1.xz, s_xdyz0_xdyz1.x), g0.z);
  float s_xydz =
         s_xyz0_xyz1_dxyz0_dxyz1.y -
         s_xyz0_xyz1_dxyz0_dxyz1.x;

  return vec4(
    normalize(vec3(s_xyz_dxyz_xdyz.y, s_xyz_dxyz_xdyz.z, s_xydz)),
    s_xyz_dxyz_xdyz.x);
}
```

The resolution of the volume is represented by the `vec3` variable `size`, which is passed to the fragment shader as a uniform input parameter. Variable `position1` is calculated from `position0`, exploiting that $\delta_0 + \delta_1 = \frac{1}{2}$ (see Equation 7). The first three components of the returned `vec4` variable form the normalized gradient, while the fourth component is the sample of the reconstructed density function. Note that the estimated gradient as well as the interpolated density are both calculated from exactly the same trilinear texture samples. For the Mitchell-Netravali notch filtering, we use the same source code as the discrete filtering by p (see Equation 13) is performed in a preprocessing without increasing the number of voxels.

5. Practical Evaluation

We tested our method on both synthetic and real-world data. Figure 5 shows the well-known Marschner-Lobb (ML) test signal [ML94] rendered by semitransparent volume rendering using different reconstruction techniques combined with on-the-fly gradient estimation. Note that a prefiltered triquadratic B-spline interpolation provides significantly higher image quality than a trilinear interpolation combined with central differencing, and shows less apparent postaliasing artifacts than NCRSI [Csé19], while the rendering performance is just slightly decreased due to our fast implementation. The frame rates were measured on an nVidia GeForce GT 720M graphics card. As shown in Figure 6, the prefiltered triquadratic B-spline interpolation performs the best in terms of asymptotic error behavior. On the other hand, the Mitchell-Netravali notch filter combined with our discrete deconvolution

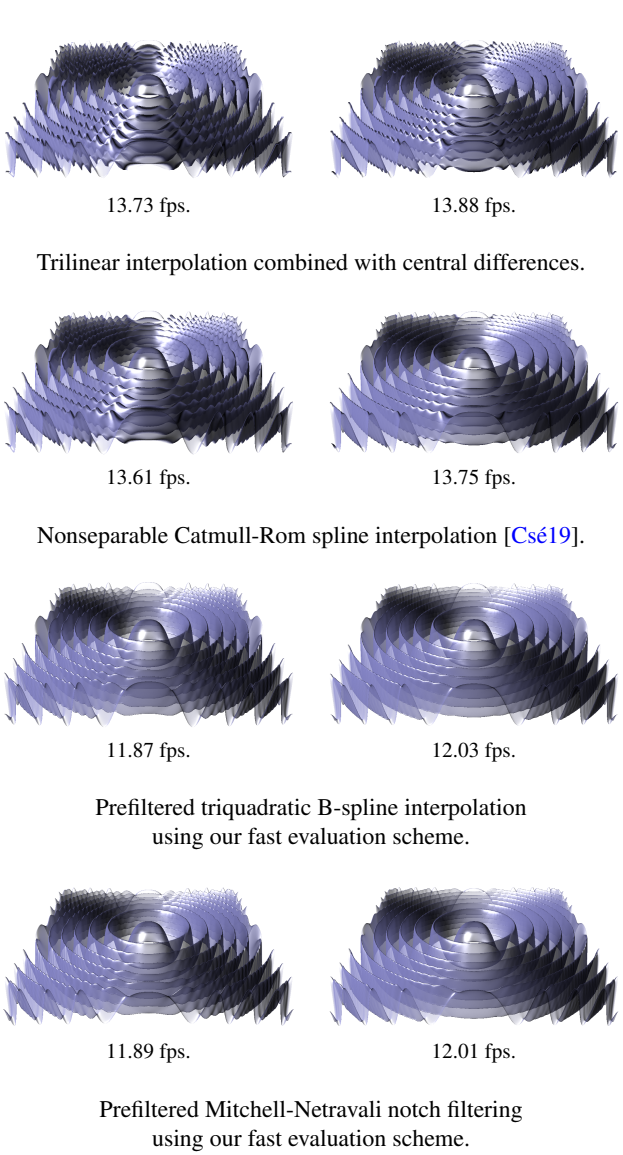


Figure 5: *Semitransparent volume rendering of the Marschner-Lobb test signal [ML94] using different reconstruction techniques combined with on-the-fly gradient estimation.*

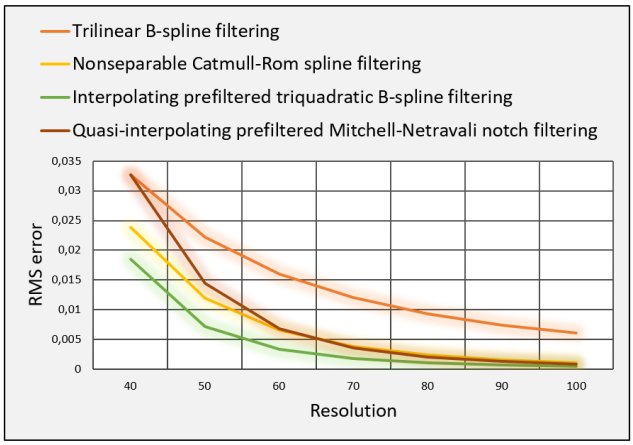


Figure 6: *RMS error of the ML reconstructions depending on the sampling resolution, which is set to be the same along the three major axis.*

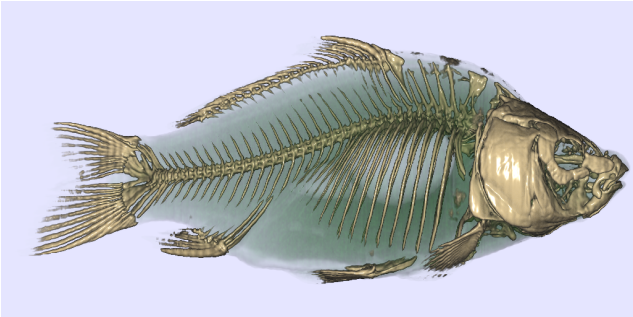
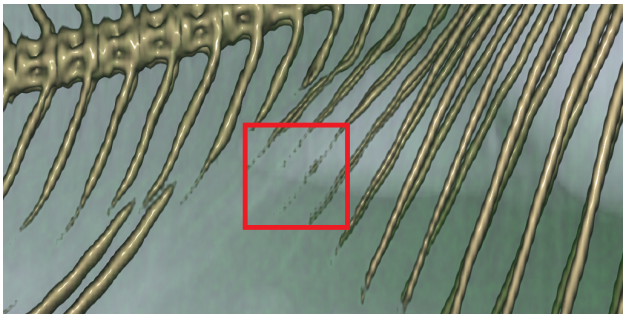


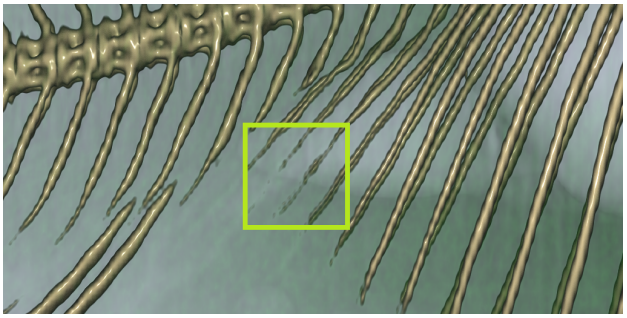
Figure 7: *CT scan of a carp visualized by semitransparent volume rendering using trilinear interpolation combined with on-the-fly central differencing.*

prefilter shows the least postaliasing artifacts (see Figure 5), while the rendering performance is the same as for the triquadratic B-spline interpolation.

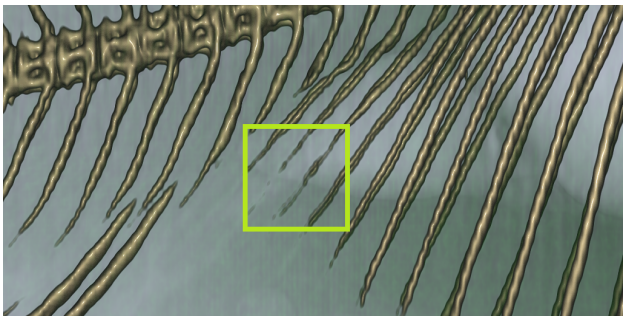
We also rendered a CT scan of a carp to see how the different reconstruction techniques perform in terms of antialiasing and detail-preservation. We applied a transfer function that consists of two triangle-shaped peaks. The first peak is placed at the representative value of the soft tissue, while the second peak is located at the representative value of the bone. To appropriately visualize the skeleton, the height of the first peak is significantly lower than that of the second one. Therefore, lower opacity values are assigned to the soft tissue than to the bone. The full-screen rendering by using trilinear interpolation is shown in Figure 7. The drawbacks of this standard resampling technique become apparent if we zoom into the details. Figure 8 shows that a trilinear interpolation introduces severe staircase aliasing and cannot completely reproduce the ribs because of its higher oversmoothing effect. NCRSI [Csé19] is able to preserve more details but still shows aliasing artifacts.



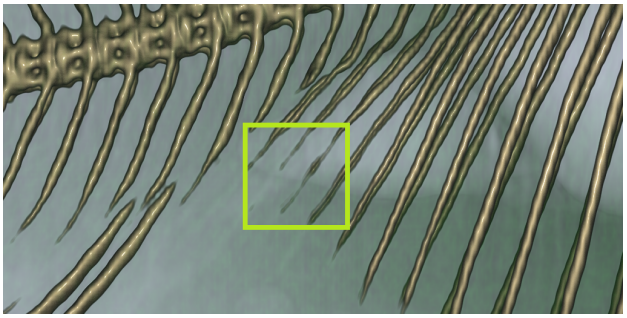
Trilinear interpolation combined with central differences: 7.14 fps.



Nonseparable Catmull-Rom spline interpolation [Csé19]: 7.17 fps.

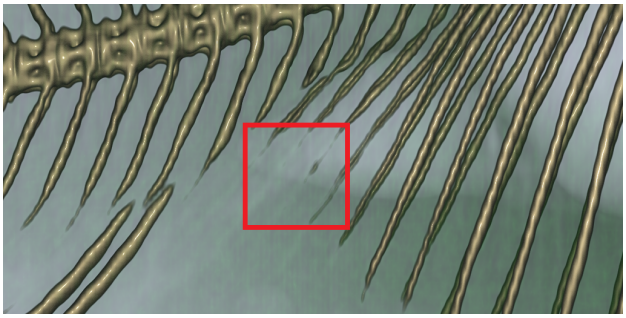


Prefiltered triquadratic B-spline interpolation
using our fast evaluation scheme: 6.69 fps.

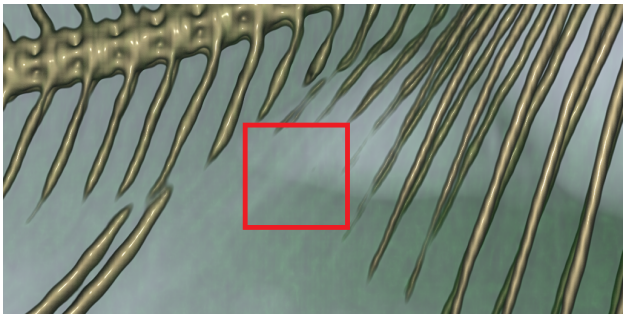


Prefiltered Mitchell-Netravali notch filtering
using our fast evaluation scheme: 6.74 fps.

Figure 8: *CT scan of a carp visualized by semitransparent volume rendering using different reconstruction techniques combined with on-the-fly gradient estimation. The region of interest is shown by a red square if the details are removed, while the green square indicates that the details are sufficiently preserved.*



Triquadratic B-spline interpolation without prefiltering.



Mitchell-Netravali notch filtering without prefiltering.

Figure 9: *Without the proposed prefilters, both the triquadratic B-spline reconstruction and the triquadratic Mitchell-Netravali notch filtering shows severe oversmoothing removing the high-frequency details. This is especially apparent in the region of interest shown by the red squares.*

The prefiltered triquadratic B-spline interpolation performs similarly in terms of both postaliasing and detail preservation. However, using the Mitchell-Netravali notch filter combined with our discrete deconvolution prefilter, the annoying staircase artifacts can be removed almost completely, without compromising the reconstruction of the high-frequency details. Here we also emphasize that, due to our fast evaluation scheme, the rendering performance is just slightly decreased compared to that of the trilinear interpolation or NCRSI. To demonstrate the significance of the discrete deconvolution prefilters proposed in Section 3.3, we also rendered the carp data set using the triquadratic B-spline reconstruction and the triquadratic Mitchell-Netravali notch filtering without prefiltering. In this case, as shown in Figure 9, both techniques result in significant oversmoothing, removing the fine details.

6. Conclusion

In this paper, we have shown that high-quality triquadratic reconstructions together with the analytic partial derivatives of the reconstructed functions can be efficiently evaluated on the GPU by taking only eight trilinear texture samples around each sample position. This requires only one additional texture fetch compared to the seven texture fetches necessary for the conventional, but lower-quality trilinear interpolation combined with on-the-fly central differencing. Our fast GPU implementations are especially suitable for semitransparent volume rendering, where the gradients need to be evaluated for the shading computations for each sample point along the rays. To the best of our knowledge, the only known GPU-accelerated volume-resampling technique that uses the same texture samples for both gradient estimation and a higher-order function reconstruction is a Nonseparable Catmull-Rom Spline Interpolation (NCRSI) [Csé19], which was recently proposed as a candidate for being a new standard tool for volume resampling. Although our method is slightly slower, it shows several attractive properties. NCRSI results in a C^0 continuous reconstruction and the gradients proposed to be evaluated by central differencing are not the analytic gradients of the reconstructed function. In contrast, our triquadratic reconstruction schemes are not just equivalent to NCRSI in terms of approximation order, but also guarantee C^1 continuous function reconstructions and provide the analytic derivatives of the reconstructed functions for free without requiring additional texture fetches.

We proposed fast implementations for a prefiltered triquadratic B-spline interpolation and also for a quasi-interpolating prefiltered triquadratic Mitchell-Netravali notch filtering. According to our frequency-domain analysis, the first technique is recommended for reproducing the most high-frequency details, while the second one is recommended for applications, where the most important aspect is the suppression of the annoying staircase artifacts.

Theoretically, it has been recognized for the first time in this paper, that the well-known Mitchell-Netravali notch filtering can be decomposed to a discrete filtering and a consecutive quadratic B-spline filtering, and this decomposition can be exploited for evaluating the reconstruction and its analytic partial derivatives from exactly the same eight trilinear samples. Furthermore, we also derived a deconvolution prefilter that optimally exploits the approximation power of the Mitchell-Netravali notch filter, and makes the reconstruction quasi-interpolating of order three. Using the proposed prefilter, the oversmoothing effect can be drastically reduced, while the advantageous antialiasing properties that stem from the notch filtering can still be maintained.

Acknowledgments

This work has been supported by the project OTKA K–124124. The carp data set is courtesy of Michael Scheuring, Computer Graphics Group, University of Erlangen.

Funder name: Hungarian Scientific Research Fund, DOI: 10.13039/501100003549, Grant Numbers: K-124124

Funder name: Magyar Tudományos Akadémia, DOI: 10.13039/501100003825, Grant Numbers:

References

[BTU99] BLU T., THÉVENAZ P., UNSER M.: Generalized interpolation: Higher quality at no additional cost. In *Proceedings of IEEE International Conference on Image Processing* (1999), pp. 667–671. 2, 5

[BU99a] BLU T., UNSER M.: Approximation error for quasi-interpolators and (multi-)wavelet expansions. *Applied and Computational Harmonic Analysis 6*, 2 (1999), 219–251. 2

[BU99b] BLU T., UNSER M.: Quantitative Fourier analysis of approximation techniques: Part I - Interpolators and projectors. *IEEE Transactions on Signal Processing 47*, 10 (1999), 2783–2795. 2

[CBU05] CONDAT L., BLU T., UNSER M.: Beyond interpolation: Optimal reconstruction by quasi-interpolation. In *Proceedings of the IEEE International Conference on Image Processing* (2005), pp. 33–36. 2, 6

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of IEEE Symposium on Volume Visualization* (1994), pp. 91–98. 1, 2

[CR74] CATMULL E., ROM R.: A class of local interpolating splines. *Computer Aided Geometric Design* (1974), 317–326. 1

[Csé08] CSÉBFALVI B.: An evaluation of prefiltered reconstruction schemes for volume rendering. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 289–301. 2

[Csé18] CSÉBFALVI B.: Fast Catmull-Rom spline interpolation for high-quality texture sampling. *Computer Graphics Forum (Proceedings of EUROGRAPHICS) 37*, 2 (2018), 455–462. 1, 2

[Csé19] CSÉBFALVI B.: Beyond trilinear interpolation: Higher quality for free. *ACM Transactions on Graphics (Proceedings of SIGGRAPH) 38*, 4 (2019). 1, 2, 3, 7, 8, 9

[EHK*06] ENGEL K., HADWIGER M., KNISS J., RECK-SALAMA C., WEISKOPF D.: *Real-Time Volume Graphics*. AK Peters Ltd., 2006. 1, 2

[HLRSR09] HADWIGER M., LJUNG P., RECK-SALAMA C., ROPINSKI T.: Advanced illumination techniques for GPU-based volume raycasting. In *ACM SIGGRAPH Course Notes* (2009). 1

[Key81] KEYS R. G.: Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-29*, 6 (1981), 1153–1160. 1

[KW03] KRÜGER J., WESTERMANN R.: Acceleration techniques for GPU-based volume rendering. In *Proceedings of IEEE Visualization* (2003), pp. 38–45. 1, 2

[Lev88] LEVOY M.: Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 3 (1988), 29–37. 1

[LME04] LI A., MUELLER K., ERNST T.: Methods for efficient, high quality volume resampling in the frequency domain. In *Proceedings of IEEE Visualization* (2004), pp. 3–10. 2

[ML94] MARSCHNER S., LOBB R.: An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization* (1994), pp. 100–107. 2, 6, 7

[MMK*98] MÖLLER T., MUELLER K., KURZION Y., MACHIRAJU R., YAGEL R.: Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization* (1998), pp. 143–151. 2

[MMMY97] MÖLLER T., MACHIRAJU R., MUELLER K., YAGEL R.: Evaluation and design of filters using a Taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics 3*, 2 (1997), 184–199. 2

[MN88] MITCHELL D., NETRAVALI A.: Reconstruction filters in computer graphics. In *Proceedings of SIGGRAPH* (1988), pp. 221–228. 1, 2, 3, 4

[RMtHRS08] RUIJTERS D., M. TER HAAR ROMENY B., SUETENS P.: Efficient GPU-based texture interpolation using uniform B-splines. *Journal of Computer Tools 13*, 4 (2008), 61–69. 1, 2

[SF71] STRANG G., FIX G.: A Fourier analysis of the finite element variational method. In *Constructive Aspects of Functional Analysis* (1971), pp. 796–830. 5

[SH05] SIGG C., HADWIGER M.: Fast third-order texture filtering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation* (2005), Matt Pharr (ed.), Addison-Wesley, pp. 313–329. 1, 2, 3

[THG00] THEUSSL T., HAUSER H., GRÖLLER M. E.: Mastering windows: Improving reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization* (2000), pp. 101–108. 2

[WE98] WESTERMANN R., ERTL T.: Efficiently using graphics hardware in volume rendering applications. In *Proceedings of SIGGRAPH* (1998), pp. 169–176. 1, 2