Wiener Splines

Markus H. Gross, David Kleiner
Computer Science Department
ETH Zurich, Switzerland
grossm@inf.ethz.ch

Abstract

We describe an alternative way of constructing interpolating B-spline curves, surfaces or volumes in Fourier space which can be used for visualization. In our approach the interpolation problem is considered from a signal processing point of view and is reduced to finding an inverse B-spline filter sequence. The Fourier approach encompasses some advantageous features, such as successive approximation, compression, fast convolution and hardware support. In addition, optimal Wiener filtering can be applied to remove noise and distortions from the initial data points and to compute a smooth, least-squares fitting 'Wiener spline'. Unlike traditional fitting methods, the described algorithm is simple and easy to implement. The performance of the presented method is illustrated by some examples showing the restoration of surfaces corrupted by various types of distortions.

1 Introduction

B-splines have been used widely and successfully in CAGD, computer graphics and visualization [1], since their versatility makes them irreplaceable for many tasks in geometric modeling. Moreover, most of the advanced commercially available modeling systems, such as ALIAS©, base on B-spline representations of curves and surfaces. However, in many applications [2], [3], the problem consists of constructing a spline through a set of given points rather than to work immediately on control points. As a consequence, the robust and efficient computation of interpolating B-spline curves and surfaces has been a fundamental task in geometric modeling.

The mathematical description of the B-spline interpolation problem can be found in many textbooks, tutorials and papers, such as [4], [5], [6], [7], [22] or others. The standard approach is to solve a sparse linear system of equations. For this purpose, we have to build and decompose a matrix, whose rows and columns are computed by evaluation of the B-spline basis functions at discrete positions in parameter space. In the case of cardinal B-spline interpolation the knots are equally spaced. Depending on the order of the B-spline, the resulting matrix is banded diagonal and respective algorithms perform correspondingly. That is, the entire interpolation step can be solved in $O(N)$, i.e. linear time [4].

Our motivation for the research presented in this paper was to point out an alternative by treating the interpolation problem from a signal processing point of view. This enables to carry over and to extend some findings from the vast amount of research in that area for the benefit of visualization. More precisely, in the case of uniform B-spline interpolation our task consists of constructing an inverse filter which is convolved over the sequence of interpolating points to provide the required control points. In this case, filtering can be accomplished in Fourier domain with the speed of FFT algorithms in $O(NlogN)$ – regardless of the spline order. In order to construct the required filters we base on some results of prior work, where Fourier transforms of uniform B-splines have been presented in [8], [9] or [10]. Since the Fourier approach implicitly assumes rotational symmetry special emphasis has to be put on the preservation of boundary conditions.

Another advantage of frequency domain constructions is the elegant embedding of least-squares fitting problems. Spectral estimations of noise and corruptions from the data enable us to compute a so-called Wiener filter [11] which provides a least-squares optimal reconstruction of the underlying function. The required parameters can be estimated efficiently for B-splines and thus provide the control mesh for a 'Wiener spline'.

One might argue that the computational advantages of the presented methods do not outperform in all cases. However, we demonstrate that the signal processing treatment of the interpolation problem allows us to apply efficient algorithms for fast convolution and FFT. Furthermore, advanced low-cost signal processors could support these operations for getting real-time performance. In addition Fourier domain interpolation comes along with very useful properties,

such as progressivity, multiresolution approximation, compression and error bounds and provides a fancy alternative implementation of the problem.

The paper is organized as follows: Without restricting generality all derivations are given for 1D parametric curves. For reasons of readability and consistency, section 2 briefly reviews some fundamentals of B-spline curves and explains the interpolation problem. In section 3 uniform B-splines are addressed and Fourier domain construction schemes are presented. Section 4 is dedicated to spectral analysis and Wiener filtering. Finally section 5 reports on some results to illustrate the performance of the method. Boldface characters are used throughout the paper to denote vector-valued variables.

2 B-Spline Interpolation

This section briefly summarizes some fundamental definitions and relationships associated with B-spline basis functions for curves and surfaces. Furthermore, the problem of computing interpolating B-spline curves is stressed. More general introductions to the issue are given for instance in [1] or [6]. In particular, we recommend interactive tools to study the properties of B-spline curves and surfaces, such as in [12].

2.1 Fundamentals of B-Splines: Revisited

A parametric B-spline curve $\boldsymbol{c}(u)$ can be considered as the following linear combination

$$\boldsymbol{c}(u) = \sum_{i=0}^{K-1} \boldsymbol{d}_i \phi_i^n(u) \ . \tag{1}$$

K: Number of basis functions.

Here, u denotes the parametric coordinate and $\boldsymbol{d}_i$ stands for the vector valued control vertices, i.e. $\boldsymbol{d}_i = (d_{ix}, d_{iy}, d_{iz})^T$, sometimes also called de Boor points. Needless to say that the upper equation provides a spatial position $\boldsymbol{c}(u) = (c_x(u), c_y(u), c_z(u))^T$ for each parameter value u and thus traces out the curve in $\boldsymbol{E}^3$.

The non-orthogonal B-spline basis functions ϕ_i are piecewise polynomials of degree n and can be computed recursively from a degree 0 prototype:

$$\begin{aligned} \phi_i^0(u) &= \begin{cases} 1 \text{ for } u \in [u_i, u_{i+1}] \\ 0 \text{ else} \end{cases} \ , \\ \phi_i^n(u) &= \frac{u \Leftrightarrow u_i}{u_{i+n} \Leftrightarrow u_i} \phi_i^{n-1}(u) \\ &\quad + \frac{u_{i+n+1} \Leftrightarrow u}{u_{i+n+1} \Leftrightarrow u_{i+1}} \phi_{i+1}^{n-1}(u) \ . \end{aligned} \tag{2}$$

Obviously, the so-called knot vector $\boldsymbol{u} = (u_0, .., u_{K+n})^T$ divides the parameter space into individual segments and accounts for the definition intervals of the bases whose local support is restricted to $n+1$ intervals.

We fix easily that the parameter spacings $\Delta_i = u_{i+1} \Leftrightarrow u_i$ can be either *uniform* or *non-uniform*, termini by which the respective curves and surfaces are referred to.

Thus, our curve is entirely defined by the following information:

- The degree n of the bases,
- the set of control vertices $\boldsymbol{d}_i$ and
- the knot vector $\boldsymbol{u}$.

In the case of uniform knot sequences the recurrence relation to construct cardinal B-spline bases holds

$$\phi^n(u) = \phi^{n-1}(u) * \phi^0(u) \ . \tag{3}$$

*: Convolution operator.

Conversely, (1) allows us to interpret the curve $\boldsymbol{c}(u)$ as a projection of an initial function $\boldsymbol{x}(u)$ onto the function space V, spanned by some *dual* bases Ψ. Along these lines, the control vertices are formally given by an inner product of $\boldsymbol{x}(u)$ and Ψ:

$$\boldsymbol{d}_i = \langle \boldsymbol{x}(u), \Psi_i(u) \rangle = \int_{-\infty}^{\infty} \boldsymbol{x}(u) \Psi_i(u) \mathrm{d}u \ , \quad \Psi_i(u) \in V \ . \tag{4}$$

Duality enforces the following relationships between the bases:

$$\langle \phi_i, \psi_j \rangle = \delta_{ij} \ . \tag{5}$$

δ_{ij}: Kronecker Delta function.

A thorough analysis of the properties of the duals are beyond the scope of our paper and we refer to [10] or [13] instead.

Note that the lower and upper bounds of the sum in (1) restrict the space V to a finite dimensionality. For parametric curves, the upper projection has to be computed separately for the x, y and z coordinates. In a similar way, projections of functions onto finite dimensional spaces are fundamental to the theory and application of finite element analysis (FEM) [14].

Tensor product surfaces can be constructed straightforwardly from parametric curves by computing 2D B-spline bases $\phi_i(u)\phi_j(v)$ from a tensor product of the 1D function spaces. The parametric surface $\boldsymbol{s}(u, v)$ is defined by:

$$\boldsymbol{s}(u, v) = \sum_{j=0}^{L-1} \sum_{i=0}^{K-1} \boldsymbol{d}_{ij} \phi_i(u) \phi_j(v) \ . \tag{6}$$

This approach carries out a set of *separable* 2D basis functions, one of which is illustrated for a bicubic 2D B-spline basis function in Fig. 1. Note again a fundamental property of B-splines, namely the local support, which equals the order $(n+1)^2$ of the polynomial.

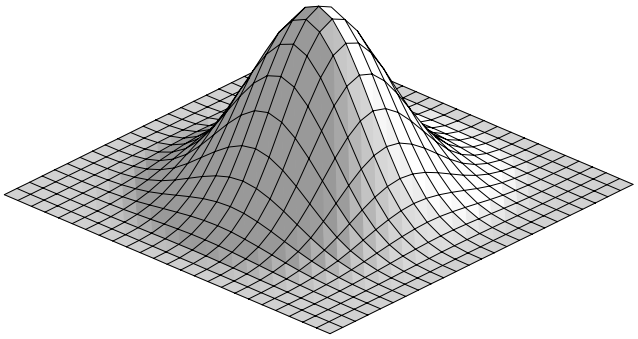


Fig. 1. Plot of a 2D bicubic B-spline basis function

2.2 The Interpolation Problem

Partition of unity of B-spline bases force the curve or surface to be bounded locally by the *convex hull* of the corresponding control vertices. In other words rather than passing through those points the curve or surface roughly approximates a set of predefined vertices. However, in many practical data modeling applications, the construction of interpolating B-splines is a critical issue. Fig. 2 illustrates the interpolation problem for a planar curve.

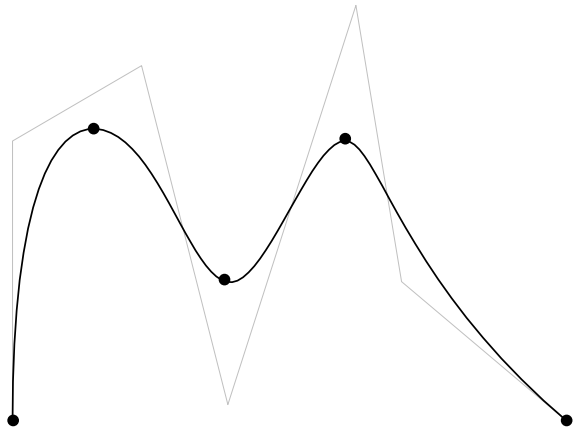


Fig. 2. Illustration of the interpolation problem: A cubic spline curve has to pass through a set of given points

In order to describe the interpolation problem mathematically let's restrict to parametric curves. Let $\boldsymbol{f} = (\boldsymbol{f}_0, .., \boldsymbol{f}_{K-1})^T$ be a vector of points to be interpolated. Each entry of $\boldsymbol{f}$ itself is a vector $\boldsymbol{f}_k = (f_{kx}, f_{ky}, f_{kz})$. Obviously, the problem collapses to finding the appropriate control vertices $\boldsymbol{d}_i$ of the curve, where for a given parameter value u_k the curve function has to pass through the point $\boldsymbol{f}_k$, or

$$\boldsymbol{f}_k = \boldsymbol{c}(u_k) = \sum_{i=0}^{K-1} \boldsymbol{d}_i \phi_i^n(u_k) \; . \tag{7}$$

Equation (7) can be rewritten in matrix form by

$$\begin{bmatrix} \boldsymbol{f}_0 \\ \cdots \\ \boldsymbol{f}_k \\ \cdots \\ \boldsymbol{f}_{K-1} \end{bmatrix} = \begin{bmatrix} \phi_0(u_0) & \phi_1(u_0) & \cdots & \phi_{K-1}(u_0) \\ \phi_0(u_1) & \cdots & \cdots & \cdots \\ \cdots & \cdots & \phi_i(u_k) & \cdots \\ \phi_0(u_{K-1}) & \cdots & \cdots & \phi_{K-1}(u_{K-1}) \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_0 \\ \cdots \\ \boldsymbol{d}_i \\ \cdots \\ \boldsymbol{d}_{K-1} \end{bmatrix} . \tag{8}$$

Hence, we have to solve a linear system of equations of type

$$\boldsymbol{f} = \boldsymbol{B} \cdot \boldsymbol{d} \; , \tag{9}$$

which apparently proceeds straightforwardly by computing the inverse matrix $\boldsymbol{B}^{-1}$ or by decomposition.

2.3 Boundary Conditions

The parameter values do not necessarily have to correspond to individual knots and thorough parameterization has turned out to be of critical importance for getting meaningful results. Due to the signal processing approach, we restrict subsequent analysis to knot-wise interpolation and direct our attention to the curve boundaries.

1) Floating End Conditions

Recalling that the curve needs $n+1$ active bases in its interval of definition we sum up to a total of $n + K \Leftrightarrow 1$ basis functions, required to fully represent the curve in $[u_0, .., u_{K-1}]$. K of them are given by (8) and the others can be selected to satisfy individual criteria.

2) Open End Conditions

We assume a knot vector of type $\boldsymbol{u} = [u_0, .., u_0, u_1, .., u_{K-1}, .., u_{K-1}]$, where the boundary knots are repeated up to the order $n+1$ of the curve. This popular technique forces the curve to interpolate the endpoints. In this case and for uneven degrees we end up with $K \Leftrightarrow 1$ equations to define the interior spline intervals and $n+1$ equations to specify

the boundary conditions. Endpoint interpolation costs 2 equations, namely $\boldsymbol{d}_0 = \boldsymbol{f}_0$ and $\boldsymbol{d}_{n+K-1} = \boldsymbol{f}_{K-1}$. The remaining $n \Leftrightarrow 1$ degrees of freedom can be set to satisfy additional criteria, like tangents, curvature and others, for instance to $\boldsymbol{t}_0 = \boldsymbol{d}_1 \Leftrightarrow \boldsymbol{d}_0$ and $\boldsymbol{t}_{K-1} = \boldsymbol{d}_{n+K-1} \Leftrightarrow \boldsymbol{d}_{n+K-2}$.

3) Periodic End Conditions

Unlike the open curves from above a periodic closed curve can be constructed by repeating the knot vector $[u_0, .., u_{K-1}]$ periodically with $u_K = u_0$, $u_{K+1} = u_1$ etc. At the same time the control vertices $[\boldsymbol{d}_0, .., \boldsymbol{d}_{K-1}]$ are repeated as well. In this case, (8) is sufficient to satisfy the interpolation and no additional degrees of freedom are left.

It has to be noted, however, that $\boldsymbol{B}$ can be arbitrarily ill-conditioned and singularities may arise for even degrees and coincidence of u_k with the knots [4]. In addition, unbalanced parameterization of the curve can lead to undesired oscillations, such as the ones illustrated in Fig. 3. Here, the curve has only been plotted in $5/6$ of the cyclic interval.

Due to the local support of the bases, $\boldsymbol{B}$ is in general a sparse banded matrix, where the width of the band depends on the degree n. Thus, sparse matrix algorithms perform decomposition in $O(N)$. We will elaborate on that subject in subsequent sections.

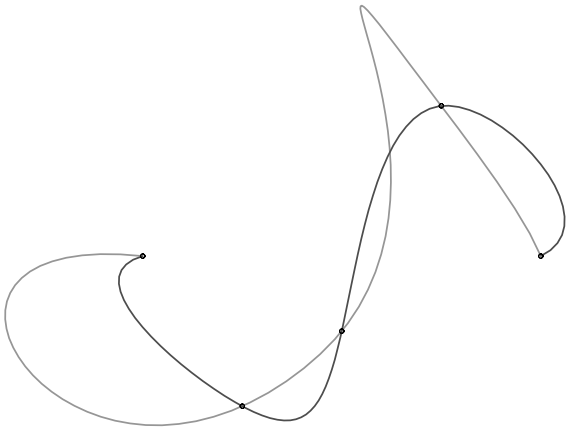


Fig. 3. Influence of the parameterization on the fairness of the curve for a periodic cubic spline (a) $\boldsymbol{u} = [0, 1, 2, 3, 4, 5, 0, 1, 2, 3]$ *(black curve)* (b) $\boldsymbol{u} = [0, 1.7, 2, 3.7, 4, 5, 0, 1.7, 2, 3.7]$ *(grey curve)*

3 Cardinal B-Spline Interpolation

In the following section, the case of uniform B-splines is discussed. Therefore, we start with an analysis of continuous convolution. By discretizing the representations, the interpolation problem can be implemented as a discrete convolution. One of the advantages of uniform B-splines is the cyclic symmetry, i.e. Toeplitz structure of the underlying matrix which allows one to compute an inverse filter sequence inverting our matrix $\boldsymbol{B}$. In this context constructions in Fourier space provide an elegant alternative.

3.1 B-Spline Interpolation as a Filter Operation

Recalling some fundamentals of signal processing [15], the continuous 1D convolution $\boldsymbol{f}(u)$ of an initial function $\boldsymbol{d}(t)$ with a kernel $\phi(u, t)$ is defined as:

$$\boldsymbol{f}(u) = \int_{-\infty}^{\infty} \boldsymbol{d}(t)\phi(u \Leftrightarrow t)\mathrm{d}t \; . \tag{10}$$

According to (4) we assume the function $\boldsymbol{f}(t)$ to trace the functional course of the inner product of $\boldsymbol{d}(u)$ and some kernel $\phi(u)$, continuously shifted along some parameter t. A discretization of (10) evaluates the integral at a finite number of positions $t_i = i\Delta$ and replaces it by the following sum:

$$\boldsymbol{f}(u) = \sum_{i=0}^{K-1} \boldsymbol{d}(i\Delta)\phi(u \Leftrightarrow i\Delta) \; . \tag{11}$$

Note that the upper equation conforms with (1), which actually defines our curve. Further discretization of $\boldsymbol{f}$ at positions $u_k = k\Delta$ and a *uniform knot spacing* set to $\Delta = 1$ results in

$$\begin{aligned} \boldsymbol{f}(k\Delta) = \boldsymbol{f}_k &= \sum_{i=0}^{K-1} \boldsymbol{d}(i\Delta)\phi(k\Delta \Leftrightarrow i\Delta) \\ &= \sum_{i=0}^{K-1} \boldsymbol{d}_i \phi(k \Leftrightarrow i) \end{aligned} \tag{12}$$

or in matrix notation as a special instance of (7)

$$\begin{bmatrix} \boldsymbol{f}_0 \\ \cdots \\ \boldsymbol{f}_k \\ \cdots \\ \boldsymbol{f}_{K-1} \end{bmatrix} = \begin{bmatrix} \phi_0 & \phi_{-1} & \cdots & \phi_{1-K} \\ \phi_1 & \phi_0 & \cdots & \phi_{2-K} \\ \phi_2 & \phi_1 & \cdots & \phi_{3-K} \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{K-1} & \phi_{K-2} & \cdots & \phi_0 \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_0 \\ \cdots \\ \boldsymbol{d}_i \\ \cdots \\ \boldsymbol{d}_{K-1} \end{bmatrix} , \tag{13}$$

where $\phi_i = \phi(i)$.

The rows of our matrix $\boldsymbol{B}$ are derived by mirroring and shifting an initial sequence $\{\phi_k\}$. It is easy to see that (13) represents a discrete convolution of the sequence of control vertices $\{\boldsymbol{d}_i\}$ with a discrete filter kernel given by $\{\phi_k\}$.

We sum up as follows: *A set of points $\{\boldsymbol{f}_k\}$ on the curve can be computed by filtering the control points with a B-spline low-pass filter kernel.*

$$\boldsymbol{f} = \boldsymbol{d} * \phi \ . \tag{14}$$

*: Convolution operator.

It follows immediately that one can solve the interpolation problem by a convolution of $\boldsymbol{f}$ with the inverse filter sequence $\{\psi_k\}$:

$$\boldsymbol{d} = \boldsymbol{f} * \psi \ . \tag{15}$$

In this case the required sequence $\{\psi_k\}$ forms the entries of the rows of $\boldsymbol{B}^{-1}$. As elucidated in subsequent sections this sequence can be constructed from a closed form representation in Fourier domain.

3.2 Stability

A further critical issue deserving attention is the stability of the interpolation problem. Therefore, we compute the condition number κ of the circular Toeplitz matrix $\boldsymbol{B}$ from above using the matrix-2 norm [16]. In this case, the norm is given directly by the ratio of the two extreme singular values σ which themselves correspond to the respective absolutes of the eigenvalues of the matrix:

$$\kappa(\boldsymbol{B}) = \|\boldsymbol{B}\| \cdot \left\|\boldsymbol{B}^{-1}\right\| = \frac{\sigma_{max}}{\sigma_{min}} \ . \tag{16}$$

For efficient computation of the eigenvalues we exploit their correspondence with the discrete Fourier transform for cyclic matrices. That is, given the closed form continuous description Φ_D of the FT of a zero-centered B-spline sequence $\{\phi_k\}$ (see [8], [10]) with uneven degree as[1]

$$\Phi_D = \phi_0 + 2 \sum_{i=1}^{\lfloor n+2 \rfloor + 1} \phi_i \cos(2\pi i f) \ . \tag{17}$$

We find the condition number easily by maximizing and minimizing (17). Some algebraic transforms which are omitted here for brevity reveal

$$\kappa(\boldsymbol{B}) = \left(1 \Leftrightarrow 4 \sum_{i=0}^{(n-2) \text{ DIV } 4} \phi_{2i+1} \right)^{-1} \ . \tag{18}$$

Table 1 depicts the condition number of $\boldsymbol{B}$ as a function of the B-spline degree n. We observe a moderate increase with n making the problem tractable for all spline orders used in practice.

[1] As a consequence the eigenvalues of $\boldsymbol{B}$ can be computed at the speed of DFT algorithms.

Table 1. Condition number of B as a function of the degree n of the B-spline

degree n	condition number κ
2	2.0
3	3.0
4	4.8
5	7.5
6	11.803
7	18.530
8	29.112
9	45.726
10	71.828
11	112.826
12	177.226
13	278.386

3.3 Fourier Domain Constructions

An elegant way of finding the dual sequence $\{\psi_k\}$ is to make use of Fourier domain descriptions. Especially, analytic formulations of the FT of B-spline bases, such as given in [8], enable us to construct the dual filter sequence without much effort.

Recalling the Fourier convolution theorem the continuous form of the curve equation of [8] converts to a product of the Fourier transforms $\boldsymbol{F}(f)$ and $\Phi(f)$, where f denotes the frequency:

$$\boldsymbol{F}(f) = \boldsymbol{D}(f)\Phi(f) \quad \text{or} \quad \boldsymbol{D}(f) = \boldsymbol{F}(f)\Psi(f) \ . \tag{19}$$

The upper equation gives us the relationship between the FT of the convolution kernel Φ and it's dual Ψ:

$$\Phi(f)\Psi(f) = 1 \ \Leftrightarrow \ \Psi(f) = \frac{1}{\Phi(f)} \ . \tag{20}$$

Hence, $\Psi(f)$ represents an inverse filter, which cancels the smoothing of $\boldsymbol{D}(f)$ by $\Phi(f)$.

In the discrete setting, we develop an analytic version of $\Psi_D(f)$ straightforwardly by inversion of (17):

$$\Psi_D(f) = \frac{1}{\phi_0 + 2 \sum_{i=1}^{\lfloor n+2 \rfloor + 1} \phi_i \cos(2\pi i f)} \ . \tag{21}$$

Note that both Φ_D and Ψ_D are continuous functions. Figure 4 illustrates the functional course of $\Psi_D(f)$ for different spline degrees.

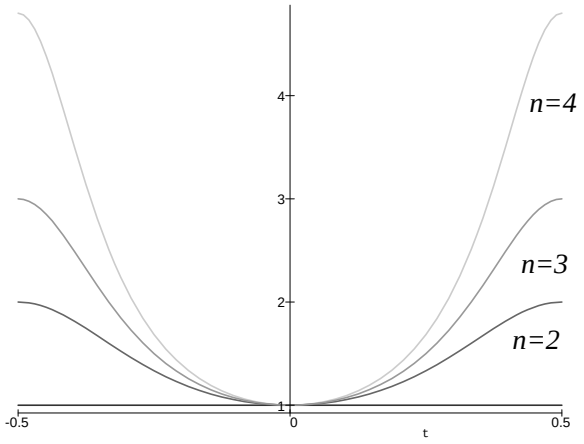


Fig. 4. Fourier transforms of the inverse filters $\Psi_D(f)$ for different degrees n

Note furthermore that $\Psi_D(f)$ has no poles in the interval of interest.

In order to implement this approach using inverse FFT, we have to discretize $\Psi_D(f)$ taking K samples in $[\Leftrightarrow 0.5, 0.5]$. Therefore, we generate a discrete sequence $\{\Psi_p\} = \{\Psi_D(f_p)\}$ of the inverse filter in Fourier space by sampling at frequency positions

$$f_p = \left(\Leftrightarrow \left(\frac{1}{2} + \frac{p}{K} \right) \right) \quad , \quad p = 0, .., K \Leftrightarrow 1 \ . \tag{22}$$

Pointwise multiplication of the discrete FT $\{\boldsymbol{F}(f_p)\}$ with the inverse filter sequence renders the FT of our control points, or[2]

$$\boldsymbol{D}_p = \boldsymbol{D}(f_p) = \boldsymbol{F}(f_p)\Psi_D(f_p) \ . \tag{23}$$

The control points $\boldsymbol{d}_k$ in spatial domain are computed by a discrete inverse Fourier transform (DIFT) of type:

$$\boldsymbol{d}_k = \frac{1}{K} \sum_{p=0}^{K-1} \boldsymbol{D}_p \mathrm{e}^{-i2\pi kp/K} \tag{24}$$

i: Complex operator.

Note that in the discrete implementation the frequency interval is shifted to $[0, 1]$.

A major advantage coming along with frequency computations is the energy concentration in the low-pass coefficients of the transform. This spectral decomposition allows us to perform versatile analysis, such as the separation of high frequency noise from the 'curve signal'. Most simply, we can define an upper cut-off frequency, $|f_g| < 0.5$ from which on all coefficients are set to zero. The resulting frequency window $\Gamma_g(f)$ is given by

$$\Gamma_g(f) = \begin{cases} 1 \text{ for } |f| \leq f_g \\ 0 \text{ for } |f| > f_g \end{cases} \tag{25}$$

and the 'denoised' control points by

$$\tilde{\boldsymbol{D}}(f) = \boldsymbol{F}(f)\boldsymbol{\Psi}(f)\Gamma_g(f) \ . \tag{26}$$

Due to the orthogonality of Fourier bases f_g governs the accuracy of the interpolation within L_2.

The Fourier setting described above assumes periodic end conditions, that is, it operates on a periodic knot vector and produces closed curves. However, most applications require open end conditions, which can be generated by evaluation of the curve in a subinterval of the parameter space. Unfortunately, this truncation may lead to undesired oscillations of the curve, such as illustrated in Fig. 5 and Fig. 6. The curve plotted in Fig. 5 exhibits oscillations at its ends that can be avoided by insertion of multiple end points. A triple end point produces a fair curve in this example.

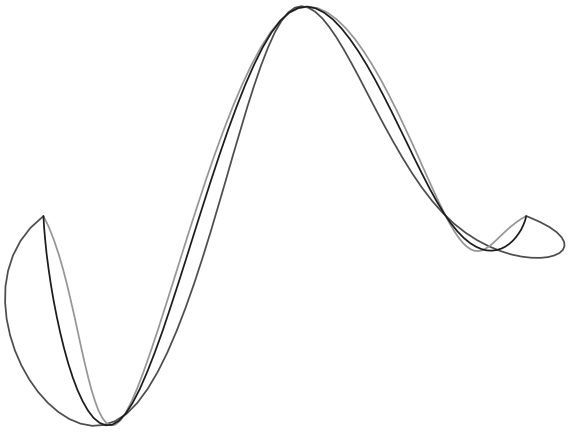
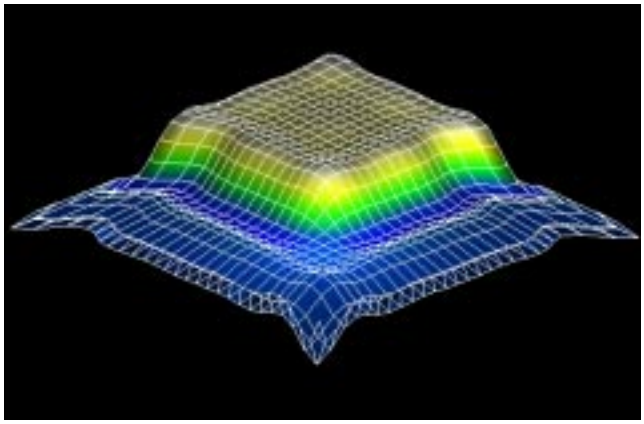


Fig. 5. Avoidance of oscillations generated by truncation. Insertion of multiple end points generates a fair interpolation curve (*grey:* single; *black:* double; *light grey:* triple)

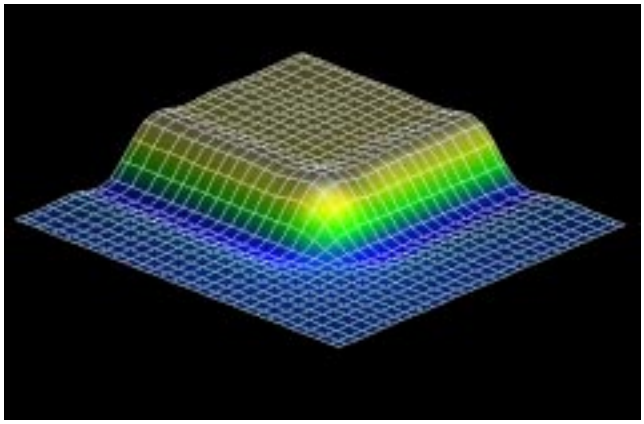
A similar behavior can be observed for 2D tensor product surfaces. In Fig. 6(a) we interpolated a step function with single end points. The parametric setting used to illustrate the phenomenon reveals the deviation of individual points due to truncation. Again, triple end points solve the problem and produce a fair surface, such as depicted in Fig. 6(b).

In order to exploit the speed of the FFT zeros have to be padded into the data vector up to the following power of 2.

[2] According to the fundamental relationships between Fourier and spatial domain discretization, the FT leads to a periodic repetition of the 'curve signal' in spatial domain. See literature, such as [17],[18] or [19].



(a)



(b)

Fig. 6. B-spline interpolation in 2D parametric settings (step function). (a) **single end points** (b) **triple end points**

4 Wiener Filtering

Besides interpolation, many practical applications in CAGD face the problem of coping with corrupted data samples $\boldsymbol{f}_k$. These curruptions are mostly caused by measuring systems of limited accuracy. In such cases, rather than interpolation the goal is often to fit a curve or surface through the points in a least squares sense. We will demonstrate that this least squares estimation is accomplished elegantly in Fourier space by exploiting the fundamental features of so-called Wiener filters [11]. Due to the rich literature on Wiener-Kalman problems[3] [20], we restrict our description to a brief summary and omit most mathematical details.

4.1 The Notion of the Wiener Filter

We start with continuous descriptions and assume our data $\boldsymbol{f}(u)$ to be distorted by an uncorrelated noise signal $\boldsymbol{r}(u)$. Our goal is to find the control vertices of the curve $\boldsymbol{c}(u)$, which smoothes $\boldsymbol{f}(u)$ in a least squares sense. Let

$$\boldsymbol{f}(u) = \boldsymbol{c}(u) + \boldsymbol{r}(u) \; . \tag{27}$$

The linearity of the Fourier transform translates the relation to:

$$\boldsymbol{F}(f) = \boldsymbol{C}(f) + \boldsymbol{R}(f) \; . \tag{28}$$

The so-called Wiener filter problem consists of finding a least squares optimal estimate $\boldsymbol{C}$ of the initial signal by design of an appropriate filter $W(f)$. It restores the 'curve signal' $\tilde{\boldsymbol{C}}(f)$ from noise

$$\tilde{\boldsymbol{C}}(f) = \boldsymbol{F}(f) \cdot W(f) \tag{29}$$

and thereby maximizes the signal-noise ratio in spatial and frequency domain with

$$\int_{-\infty}^{\infty} \left| \tilde{\boldsymbol{C}}(f) \Leftrightarrow \boldsymbol{C}(f) \right|^2 \mathrm{d}f = min \; . \tag{30}$$

It can be proofed that this optimal filter is constructed from the power spectra both of the initial signal $\boldsymbol{C}(f)$ and of the distorting noise $\boldsymbol{R}(f)$

$$W(f) = \frac{|\boldsymbol{C}(f)|^2}{|\boldsymbol{C}(f)| + |\boldsymbol{R}(f)|^2} \; . \tag{31}$$

Equation (31) is often called the *Wiener filter*. Given some estimates of the power spectra this simple and elegant equation allows the computation of least squares fitting curves.

Now we are in place to combine the results of section 3 with (31) and invoke our filter Ψ. As a consequence we obtain a least squares estimate $\boldsymbol{D}_{lq}(f)$ by

$$\boldsymbol{D}_{lq}(f) = \boldsymbol{F}(f)\Psi(f)W(f) \; . \tag{32}$$

In finite dimensional discrete settings the relations from above can be summarized as follows: An additional weighting of the sequence $\{\boldsymbol{D}_p\}$ computed by (23) with a Wiener filter sequence $\{W_p = W(f_p)\}$ provides us with the control points $\{\boldsymbol{D}_{lq(p)}\}$ of a curve, which fits through the data $\{\boldsymbol{F}_p\}$. The inverse Fourier transform renders the desired points in spatial domain $\{\boldsymbol{d}_k\}$.

[3] Norbert Wieners' fundamental work relates to linear estimation theory of stochastic signals. Later, recursive implementations of such filters using the notion of state variables became famous as Kalman filters [20]. They have enormous practical importance for signal detection, restauration and tracking.

Note that (31) can also be computed as a vector-valued filter $\boldsymbol{W}(f)$ treating each scalar component separately.

4.2 General Spectral Estimates

One of the difficulties of the Wiener filter is to compute robust spectral estimates of signal and noise. In cases of unknown data corruption we have to extract this information somehow from the power spectrum of the given data $\boldsymbol{F}(f)$, where

$$|\boldsymbol{C}(f)|^2 + |\boldsymbol{R}(f)|^2 = |\boldsymbol{F}(f)|^2 \ . \tag{33}$$

Generally, spectral estimation is nontrivial and in many cases there is no chance to get the initial signal back from the corrupted sequence. A fast and mostly robust way of estimation [21] is illustrated in Fig. 7. Assuming the signal to be smooth and band-limited the noise characteristics can be estimated by computation of an asymptotic regression line towards high frequencies. Assuming furthermore a linear or constant spectral behavior of the distortion, straightforward extrapolation reveals the signal.

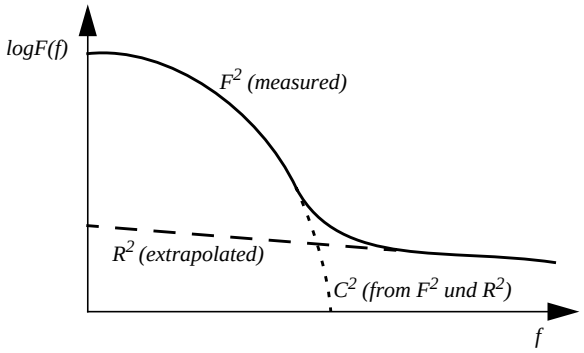
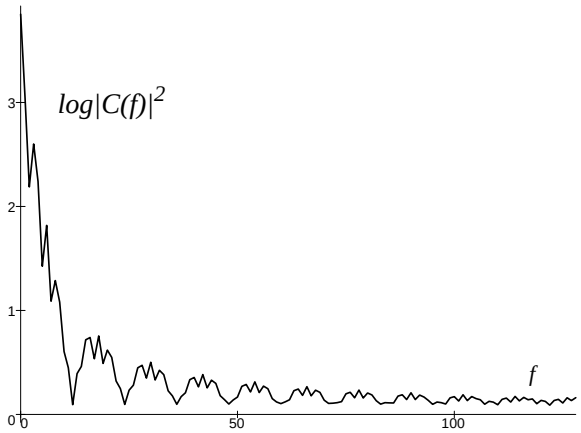
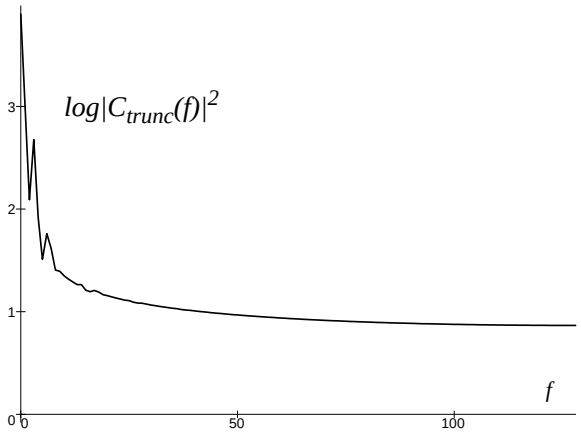


Fig. 7. Power spectrum of some measured data and its decomposition into signal and noise (see also [21])



(a)



(b)

Fig. 8. Illustration of the effects of truncation on the power spectrum of B-splines. (a) **Power spectrum of a periodic curve with 256 samples.** (b) **Truncation after 5/6 of the period**

4.3 Improved Estimates for B-Splines

For B-spline curves and surfaces, however, it is possible to incorporate more a priori knowledge into the estimation algorithm. Here, we consider the uncorrupted signal to represent a smooth 'low-pass' curve defined by a linear combination, such as in (1). For periodic curves and uniform knots the Fourier transform of (1) is obtained by

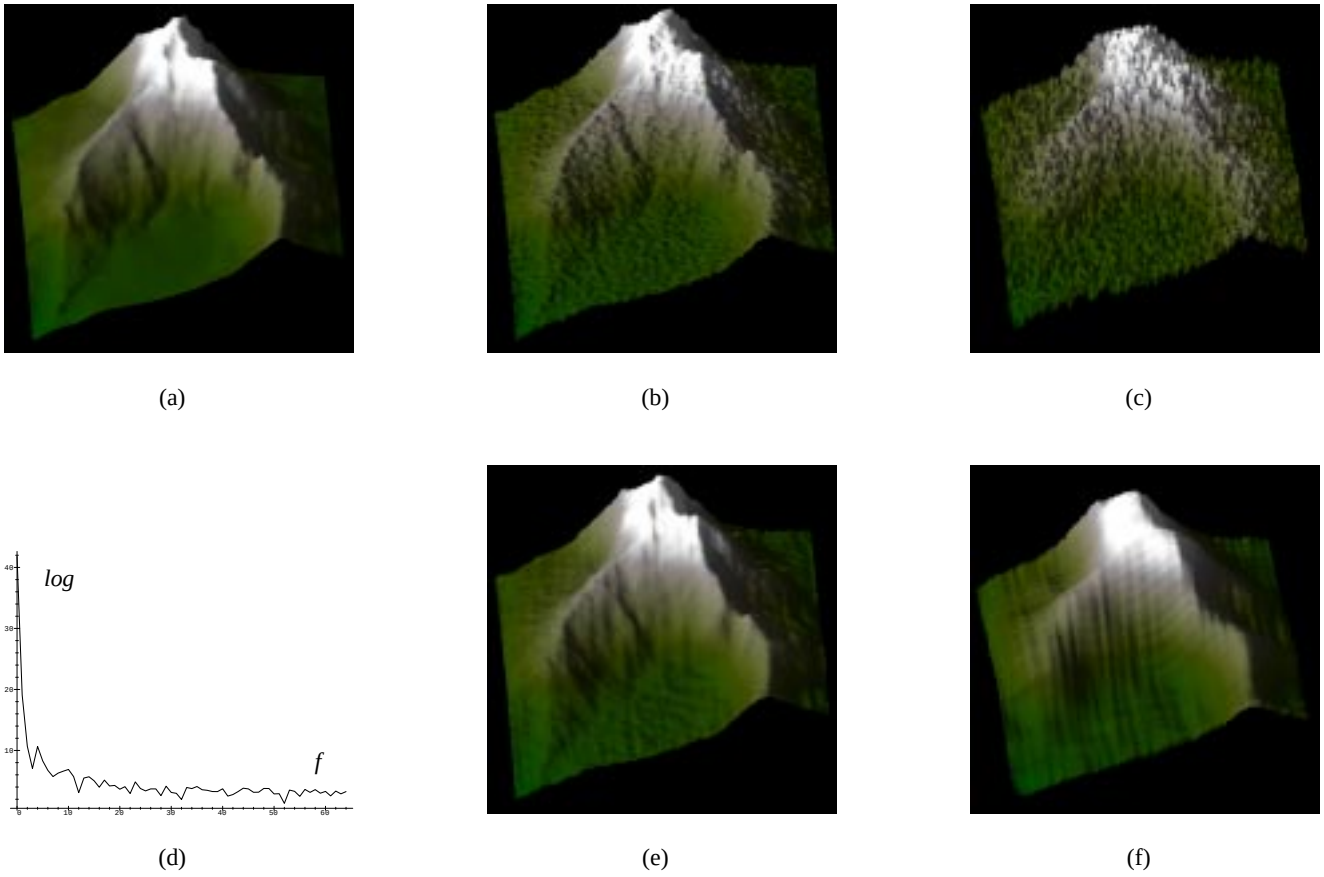
$$\boldsymbol{C}(f) = \mathrm{sinc}^n(f) \sum_{j=0}^{K-1} \boldsymbol{d}_j \cdot \mathrm{e}^{i2\pi f j} \tag{34}$$

because the shifting of individual basis functions translates to a phase shift of its Fourier transform. Consequently, a B-spline curve distorted by some noise yields to

$$\boldsymbol{C}_{noise}(f) = \mathrm{sinc}^n(f) \sum_{j=0}^{K-1} \boldsymbol{d}_j \cdot \mathrm{e}^{i2\pi f j} + \boldsymbol{R}(f) \ . \tag{35}$$

Thorough analysis of this equation shows that the signal vanishes at the zero crossings f_i of the $sinc$ polynomial. Obviously, the noise spectrum can be estimated from

$$\boldsymbol{B}_{noise}(f_i) = \boldsymbol{R}(f_i) \ . \tag{36}$$



(a) (b) (c)

(d) (e) (f)

Fig. 9. Data restauration using the Wiener Filters. (a) **Original data set.** (b) **Corrupted data (noise level = 0.11).** (c) **Corrupted data (noise level = 0.5).** (d) **Horizontal slice from the power spectrum.** (e) **Restauration of Fig. 9(b).** (f) **Restauration of Fig. 9(c). (Data source: Courtesy Bundesamt für Landestopographie, Bern, Switzerland)**

Unfortunately, these relations are only valid for periodic curves. Truncation to enforce open end conditions affects the Fourier transform of the curve and cancels out the zero crossings. In Fig. 8 a periodic curve is compared to a curve truncated after $5/6$ of the period.

We recommend to employ the asymptotic regression line method for open end conditions. In mere periodic settings the zero crossing search can be an appropriate choice.

5 Results

In the following section we present various applications to illustrate the performance and suitability of our method. It ranges from sheer parametric interpolation for image warping to removal of quantization noise in compressed geometry.

5.1 Error Norms

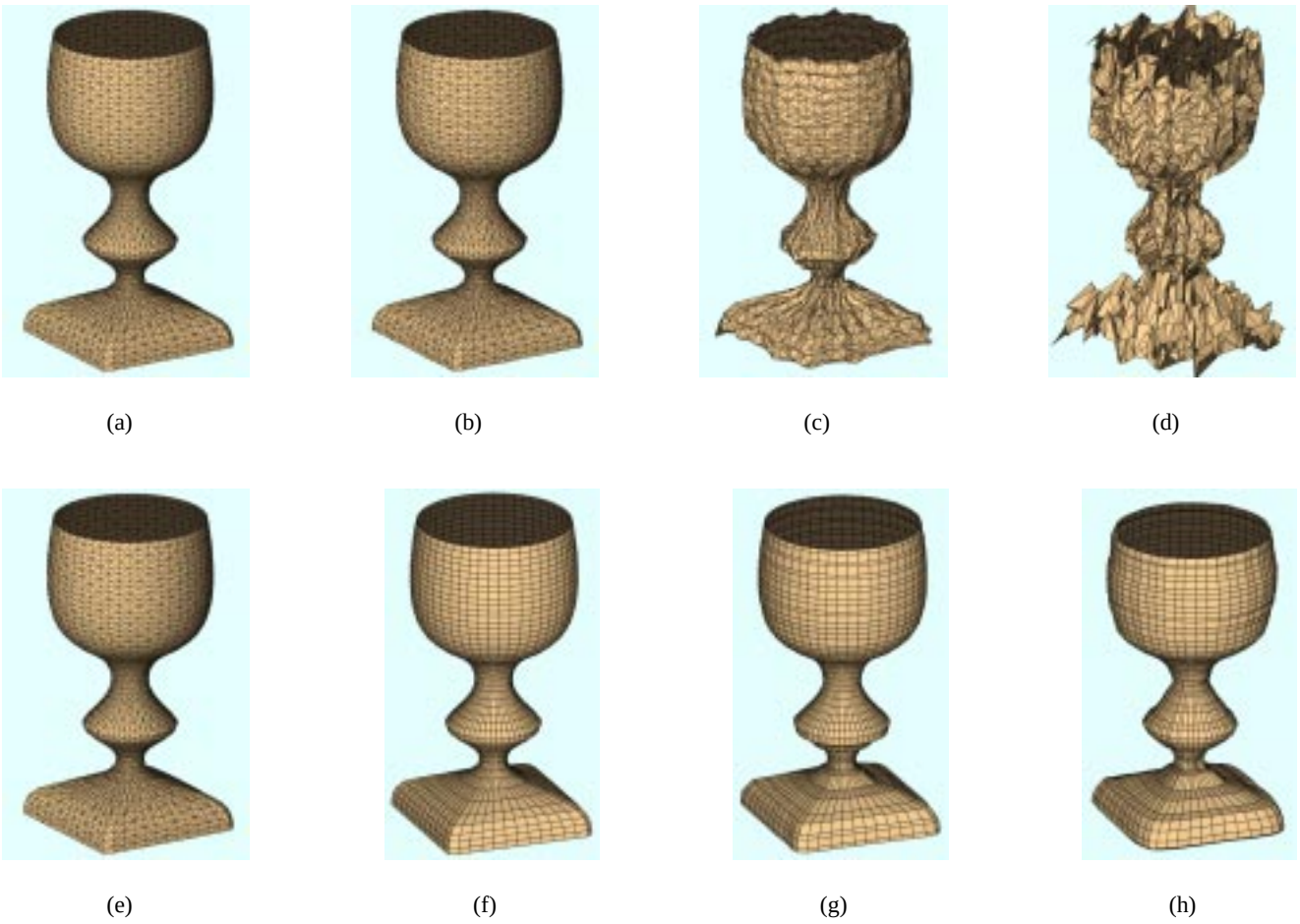
As with all signal processing approaches our most important measure to quantify the error imposed by the method is the signal-noise ratio defined as

$$\frac{\boldsymbol{C}}{\boldsymbol{R}} = 20 \cdot \log \frac{\boldsymbol{E}_C}{\boldsymbol{E}_R} \quad \text{(dB)} \,, \tag{37}$$

where $\boldsymbol{E}_C$ and $\boldsymbol{E}_R$ are the L_2-norms of $\boldsymbol{c}(u)$ and $\boldsymbol{r}(u)$. For geometric objects it is sometimes desirable to record the maximum $(L_{\inf})$ error. Therefore both errors will be used.

5.2 Interpolation Error

Table 2 reports the numerical error of the Fourier space method for different test data sets and 32 Bit floating point precision. The good conditioning of the problem keeps the numerical errors small.



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 10. Wiener Filtering of the parametric 'Goblet' data set corrupted by quantization noise. (a) – (d) **Different bit rates used for compression.** (e) – (h) **Resulting surfaces computed by the Wiener filter**

Table 2. Maximum interpolation error $(L_{\inf})$ **for different test data sets**

data set	Maximum error
Box function	$1.192\ 10^{-7}$
Matterhorn	$1.320\ 10^{-6}$

5.3 Noise Removal

The second example is a non-parametric digital terrain model of size 100x100 of the Matterhorn, Switzerland. In order to investigate the power of the Wiener filter, we corrupted the data artificially by white noise signals of different amplitude. Fig. 9(a) depicts the original data set along with two different noise signals. The restauration of the signal provides visually appealing results, where for large noise amplitudes the Wiener filter cancels out high frequency signal components, such as in Fig. 9(f). Specifically, Fig. 9(b) shows a 1D horizontal slice from the power spectrum. Table 3 gives some quantitative results. The Wiener filter, as a least squares estimator improves the L_2 error mostly for medium and low signal-noise ratios, not necessarily the $L_{\inf}$ error.

In order to further investigate the performance of the Wiener filter we tested parametric data sets distorted by noise of unknown spectral composition. The data sets were generated by a lossy geometry compression scheme, such as presented in [2]. Changing the number of bits used to encode the data creates some quantization noise. In Fig. 10 the initial data set is a smooth parametric B-spline surface sampled at size 100x100. In this example the performance of the Wiener filter is paramount. We observe that even for strong distortions (5 Bit quantization) the estimation produces a

Table 3. Performance of the Wiener filter on the digital terrain data set. Figures in the second row represent the Wiener filter

noise	$\boldsymbol{C}/\boldsymbol{R}$ ratio (dB)	L_2-error	$L_{\inf}$-error
0.11	78.48	2.21	2.58
	77.36	2.51	7.22
0.25	64.93	10.74	5.69
	65.57	10.0	9.23
0.5	52.96	44.48	11.5
	54.75	36.2	14.34
1.0	41.68	176.8	23.2
	43.9	136.2	20.6

Table 4. Error improvement of the Wiener filter for the 'Goblet' data set

quantization	L_2-error	$L_{\inf}$-error
10 Bits	$8.78\ 10^{-7}$	$3.20\ 10^{-3}$
	$5.13\ 10^{-6}$	$1.54\ 10^{-2}$
7 Bits	$3.63\ 10^{-5}$	0.0213
	$2.07\ 10^{-5}$	0.0144
5 Bits	$7.76\ 10^{-4}$	0.087
	$2.53\ 10^{-4}$	0.041

visually smooth and fair surface. Table 4 gives quantitative results.

5.4 Image Warping

Our last example relates to 2D interpolation for image warping. Here, an initial grid is defined interactively on the image to govern the warp. We assume the grid to be rectilinear in some 2D tensor product parameter space, spanned by the coordinates (u, v). The goal is to compute a smooth cubic interpolation of the warped target image based on the grid information. In this case the interpolation problem is twofold: In a first step, we construct a vector-valued parametric interpolation function $\boldsymbol{s}(u, v)$ which essentially computes the spatial position (s_x, s_y) in the initial image at each given coordinate pair (u_i, v_j) in parameter space. By equally spaced sampling of the parameter space we obtain a 2D matrix of positions in the initial image at any desired resolution. Now, given a spatial position (s_x, s_y) the second step consists of a non-parametric cubic B-spline interpolation/filtering in the original image. To obtain the pixel color the interpolation is computed independently for R, G and B. Note that the second step is an *anti-aliasing* of the results.

Some illustrations are given in Fig. 11 and Fig. 12, respectively. We observe that the cubic polynomial provides smooth interpolations and cancels out all aliasing artifacts while still preserving the image details. The parametric interpolation of image positions allows us to define fancy warp grids. In this context the Wiener filter could be used to denoise the initial image.

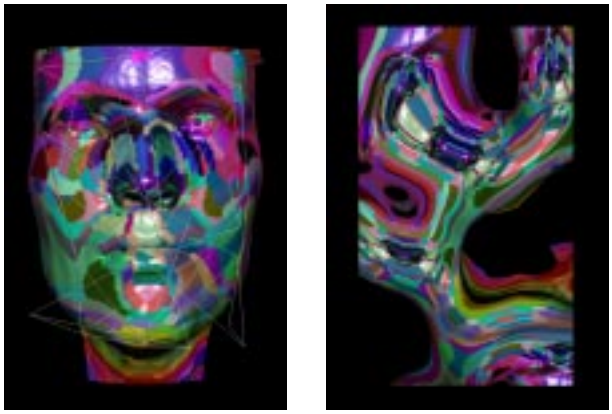
(a)

(b)

Fig. 11. Image Warping. (a) **Original image and interactively defined warp grid.** (b) **Resulting image**

6 Conclusions and Future Work

We presented an alternative way of computing B-spline interpolation by inverse filtering in Fourier space. Although the computational speed of the approach is limited by the complexity of FFT algorithms the spectral formulations provide various useful properties. Specifically, least squares



(a) (b)

Fig. 12. More fancy warp grids. (a) **Original image and grid.** (b) **Resulting image**

fitting problems and the removal of data corruptions can be tackled efficiently by using the notion of Wiener filters. Although the presented method performs surprisingly well for some of the examples, yet, the robust spectral estimation of unknown distortion sources is still an open issue and will be part of future investigations.

References

[1] R. Barthels, J. Beatty, and B. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Morgan Kaufmann, 1987.

[2] O. Staadt, M. Gross, and R. Weber. Multiresolution compression and reconstruction. *Proceedings of IEEE Visualization*, pages 337–364, 1997.

[3] L. Lippert, M. Gross, and C. Kurmann. Compression domain volume rendering for distributed environments. *Computer Graphics Forum*, 16(3):C95–C107, 1997.

[4] C. de Boor. *A Practical Guide to Splines*. Springer Verlag, Berlin, 1978.

[5] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design. 3rd. Edition*. Academic Press, 1992.

[6] W. Böhm et al. A survey of curve and surface methods for CAGD. *Computer Aided Geometric Design*, 1:1–60, 1984.

[7] M. Mummy. Hermite interpolation with b-splines. *Computer Aided Geometric Design*, 6:177–179, 1989.

[8] E. Catmull and R. Rom. A class of local interpolating splines. *Computer Aided Geometric Design*, pages 317–326, 1974.

[9] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing part II: Efficient design and applications. *IEEE Transactions on Signal Processing*, 41(2):834–848, 1993.

[10] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing part i: Theory. *IEEE Transactions on Signal Processing*, 41(2):821–832, 1993.

[11] C. Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[12] N. Wiener. Extrapolation, interpolation, and smoothing of stationary time series with engineering applications. *Internal Report, MIT*, 1949.

[13] A. Rockwood and P. Chambers. *Interactive Curves and Surfaces*. Morgan Kaufman, 1996.

[14] C. A. Micchelli. *Mathematical Aspects of Geometric Modeling*. SIAM, 1995.

[15] R. Koch and M. Gross et al. Simulating facial surgery using finite element models. *Computer Graphics Proceedings, Annual Conference Series, ACM Siggraph*, pages 421–428, 1996.

[16] A. Papoulis. *Signal Analysis*. McGraw-Hill, 1977.

[17] G. Golub and C. van Loan. *Matrix Computations. 3rd. Edition*. John Hopkins University Press, 1996.

[18] O. Brigham. *The Fast Fourier Transform*. Prentice Hall, 1974.

[19] H. Nussbaumer. *Fast Fourier Transforms and Convolution Algorithms*. Springer Verlag, 1982.

[20] D. Dudgeon and R. Mersereau. *Multidimensional Signal Processing*. Prentice Hall, 1984.

[21] M. Grewal and A. Andrews. *Kalman Filtering*. Prentice Hall, 1993.

[22] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C. 2nd Edition*. Cambridge University Press, 1992.