*S*pline and *W*avelets for image warping and projection

Stefan Horbelt

THESE N. 2397 (2001)

Thèse presentée au département de microtechnique
pour l'obention du grade de docteur en sciences
et acceptée sur propositon du jury:

Prof. M. Unser,
Directeur de thèse

Dr. P. Fua,
Rapporteur

Prof. M. Gross,
Rapporteur

Dr. F. Peyrin,
Rapporteur

Prof. M. Vetterli,
Rapporteur

ECOLE DE POLYTECHNIQUE FEDERALE DE LAUSANNE - 2001

Cover on first page design by A. Unser
Edited by © EPFL (2001)
Typesetting with LyX1.1.5 and LaTeX
Printing and bounding by Repro-EPFL

“On ne voit bien qu’avec le coeur,
l’essentiel est invisible pour les yeux.”

“Man sieht nur mit dem Herzen gut,
das Wesentliche ist für die Augen unsichtbar.”

—ANTOINE DE SAINT-EXUPÉRY
French pilot and poet, 1900–1944

Contents

Abstract

This thesis investigates advanced signal processing concepts and their application to geometric processing and transformations of images and volumes.

In the first part, we discuss the class of transformations that project volume data onto a plane using parallel line integrals; it is called the X-ray transform. In computer tomography (CT) the problem is to reconstruct the volume from these projections. We consider a basic setup with parallel projection and a geometry model in which the CT scanner rotates around one main axis of the volume. In this case, the problem is separable and reduces to the reconstruction of parallel images (slices of the volume). Each image is reconstructible from a series of 1D projections taken at different angular positions. The standard reconstruction algorithm is the filtered back-projection (FBP). We propose an alternative discretization of the Radon transform and of its inverse that is based on least-squares approximation and the convolution of splines. It improves the quality of the transform significantly.

Next we discuss volume rendering based on the X-ray transform. The volume is represented by a multiresolution wavelet decomposition. The wavelets are projected onto an adaptive multiresolution 2D grid. The multiresolution grid allows to speed up the rendering process especially at coarse scales.

In the second part of the thesis, we discuss transformations that warp images. In computer graphics, this is called texture mapping. Simple warps, such as shear, rotation, or zoom, can be computed by least-squares sampling, e.g. again with convolutions of splines. For more general warps there is not an easy continuous solution, if an analytical one exists at all. After a review of existing texture mapping methods, we propose a novel recursive one, which minimizes information loss. For this purpose, the texture is reconstructed from the mapped image and compared to the original texture. This algorithm outperforms the existing methods in terms of quality, but its complexity can be very high. A multiresolution version of the algorithm allows to keep the storage requirements and computational complexity within acceptable range.

Fast transmission of textures and 3D models over communication links requires low bitrate and progressive compression. We can achieve very low bitrate if we code textures only with the information necessary for a given view of the 3D scene. If the view changes, the missing information will be transmitted. This results in a progressive bitstream for an animated 3D scene. In contrast to recursive texture mapping, we do not back-project the texture, but come up with

a heuristic that predicts the information loss. This concept is called view-dependent scalability and we show how to apply it on DCT-based (as part of MPEG-4) and wavelet-based coders.

Last, we inspect the question of how to balance the bit budget of a jointly coded mesh and texture for the progressive and view-dependent transmission of a 3D model. By exhaustive search, we find the rate-distortion optimal path. By marginal analysis we find a close solution but at much lower costs (only two evaluated frames per step as compared to a full search).

Kurzfassung

Diese Doktorarbeit diskutiert fortgeschrittene Signalverarbeitungskonzepte und ihre Anwendung zur geometrischen Transformation von Bildern und Volumina.

Im ersten Teil behandeln wir eine Klasse von Transformationen, die Volumendaten auf eine Fläche durch parallele Linienintegrale projiziert; dies wird auch als Röntgenstrahlenprojektion bezeichnet. In der Computertomographie (CT) wird das Problem behandelt, Volumen von diesen Projektionen zu rekonstruieren.

Im folgenden beziehen wir uns auf Parallelprojektion und eine Geometrie bei der sich der CT Scanner um eine Volumenhauptachse dreht. Unter diesen Bedingungen ist das Problem separierbar und reduziert sich auf die Rekonstruktion von parallen Bildern (Volumenscheiben). Jedes Bild ist rekonstruierbar aus einer Serie von eindimensionalen Projektionen, die aus verschiedenen Projektionswinkeln aufgenommen wurden. Der Standardrekonstruktionalgorithmus ist die gefilterte Rückprojektion (FBP). Wir schlagen eine alternative Diskretisierung der Radon Transformation und ihrer Inversen vor, welche auf Projektion, die optimal im Sinne der kleinsten Quadrate sind, und auf Faltungen von Splines basiert. Sie verbessert die Qualität der Radon Transformation erheblich.

Anschließend betrachten wir die Visualisation von Volumen durch Parallelprojektion. Das Volumen wird hier durch eine Wavelet-basierte Multi-Skalen-Pyramide repräsentiert. Die Wavelets werden dabei auf ein adaptives zweidimensionales Gitter projiziert. Dieses Multi-Skalen-Gitter beschleunigt die Visualisation besonders für grobe Skalen.

Im zweiten Teil dieser Doktorarbeit konzentrieren wir uns auf Transformationen, die Bilder deformieren. In der Computergraphik nennt man dieses Verfahren *Texture Mapping*. Einfache Deformationen (*Warps*), wie Scheren, Drehen oder Zoomen, können durch Abtastung im Sinne der kleinsten Quadrate berechnet werden. Für allgemeinere Warps gibt es keine einfache kontinuierliche Lösung mehr, falls eine analytische Lösung überhaupt existiert. Nach einer Literaturübersicht der Texture Mapping Methoden schlagen wir eine neue, rekursive Methode vor, die den Informationsverlust minimiert. Zu diesem Zweck wird die Textur aus der projizierten Textur rekonstruiert und mit der ursprünglichen Textur verglichen. Dieser Algorithmus übertrifft die vorhandenen Methoden bezüglich Qualität, allerdings kann seine Komplexität sehr hoch sein. Die Multi-Skalen-Version des Algorithmus hält die Speicheranforderungen und den Berechnungsaufwand in einem akzeptablen Bereich.

Die Übertragung von Texturen und dreidimensionalen Modellen über Datenübertragungskanäle erfordert niedrige Datenraten und progressive Komprimierung. Wir können sehr niedrige Datenraten erzielen, wenn wir Texturen nur mit den Informationen kodieren, die für eine gegebene Sicht der dreidimensionalen Szene notwendig sind. Wenn sich der Blickwinkel ändert, werden die zusätzlich notwendigen Informationen übertragen. Für eine animierte 3D-Szene ergibt dies einen progressiven Datenstrom. Im Gegensatz zum rekursiven Texture Mapping, projizieren wir die Texture nicht zurück sondern sagen mittels einer Heuristik voraus, welche Information in der Textur unterdrückt wird. Dieses Konzept wird als ansichts-abhängige Skalierung bezeichnet, und wir zeigen, wie es sich in DCT-basierten (z.b. in MPEG-4) und Wavelet-basierten Komprimierungsverfahren einfügen lässt.

Zuletzt untersuchen wir die Frage, wie ein Kodierungsbudget aufgeteilt werden sollte, um ein dreidimensionales Gitter und die zugehörige Textur gemeinsam zu kodieren. Durch die vollständige Evaluierung aller Möglichkeiten finden wir den optimalen Weg, um das Kodierungsbudget progressiv zu verwenden. Durch marginale Suche erhalten wir eine Lösung, die der optimalen Lösung sehr nahe kommt, jedoch die Suchkosten wesentlich reduziert.

Version abrégée

Cette thèse étudie des concepts avancés de traitement des signaux et leur application aux transformations géométriques des images et des volumes. Dans la première partie, nous discutons la classe des transformations qui projettent des volumes sur un plan en utilisant des intégrales de lignes parallèles: la transformation des rayons X. En tomographie numérique (CT), le problème est de reconstruire le volume à partir de ces projections. Nous considérons l'hypothèse de base de la projection parallèle et un modèle de géométrie dans lequel le module de balayage de la CT tourne autour d'un axe principal du volume. Dans ce cas, le problème est séparable et se réduit à la reconstruction des images parallèles (tranches du volume). Chaque image est reconstructible à partir d'une série de projections 1D prises à différentes positions angulaires. L'algorithme standard de reconstruction est la rétro-projection filtrée (FBP). Nous proposons une nouvelle discrétisation de la transformation de Radon et de son inverse qui est basée sur l'approximation aux moindres carrés et sur la convolution de splines. La qualité de la transformation en est améliorée de manière significative.

Ensuite, nous traitons de la visualisation volumétrique basée sur la transformation des rayons X. Le volume est représenté par une décomposition en ondelettes multirésolution. Les ondelettes sont projetées sur une grille multirésolution 2D adaptative. La grille multirésolution accélère le processus de visualisation; en particulier, aux échelles grossières.

Dans la deuxième partie de la thèse, nous discutons les transformations qui déforment des images, ce qu'on appelle *plaquage de texture (texture mapping)* en infographie. Des déformations simples, telles que le cisaillement, la rotation ou le zoom peuvent être calculées par échantillonnage aux moindres carrés, par exemple avec des convolutions de splines. Il n'existe aucune solution analytique simple pour des déformations plus générales. Après un examen des méthodes existantes de texture mapping, nous proposons une nouvelle méthode récursive qui réduit au minimum la perte d'information. À cette fin, la texture est reconstruite à partir de l'image projetée et comparée à la texture initiale. Cet algorithme surpasse les méthodes existantes en termes de qualité, mais sa complexité peut être très élevée. Une version multirésolution de l'algorithme maintient la consommation de mémoire et la complexité de calcul dans une marge acceptable.

La transmission rapide des textures et des modèles 3D exige des bas débits et une compression progressive. Nous pouvons réaliser des débits très bas si nous codons les textures seulement avec l'information nécessaire pour une vue donnée de la scène 3D. Si la vue change,

l'information manquante sera transmise. Cela a pour conséquence un bitstream progressif pour une scène 3D animée. Contrairement au texture mapping récursif, nous ne projetons pas la texture en arrière, mais nous utilisons une méthode qui prévoit la perte d'information. Ce concept s'appelle "view dependent scalability" et nous montrons comment l'appliquer à un codeur basé sur la DCT (en tant qu'élément de MPEG-4), ansi qu'à un codeur basé sur des ondelettes.

Finalement, nous examinons comment équilibrer le budget en bits d'une maille et d'une texture conjointement codées pour la transmission progressive et vue-dépendante d'un modèle 3D. Par une technique de recherche exhaustive, nous trouvons le chemin optimal au sens du meilleur compromis taux/distortion. Par une analyse marginale, nous trouvons une solution proche mais d'un coût bien inférieur (seulement deux trames évaluées par étape par rapport à une recherche complète).

Chapter 1

Introduction

"I was born not knowing and have had only a little time to change that here and there."
—RICHARD P. FEYNMAN

1.1 Statement of problem

We speak of geometric image processing when we perform operations on images or volumes that are based on geometric transformation that involve the interpolation of in-between pixel values. The most basic examples are affine transformations such as resizing and rotation [105], the perspective projection in computer graphics [19], and more general warpings as in texture mapping [31]. The transformation may as well be combined with other convolution-like operators; for example, some form of integration to simulate the propagation of rays through a medium. This is typically the framework used for volume rendering [52, 65]. The inverse problem, the reconstruction of an image or volume from a collection of line integrals (or projections) [75] is of great interest as well because it is the basis for tomographic imaging (CT, PET, and SPECT) [80]. Finally, we investigate view-dependency scalability for the transmission and coding of image [43]; here, only the data of an image that is relevant for a given geometric transform is transmitted.

Geometric transformations are defined continuously. In contrast to this, digital computers are based on binary logic, limited memory, and discrete values. That is why continuous image transformations have to be computed using discrete algorithms. Often in geometric image transformations, the image data is resampled at variable sampling rates. The design of the sampling prefilter is crucial. The image is blurred when the filter takes away too much information; alias-

ing artifacts occur when the filter leaves too much. For non-affine geometric transformations, the filters are space-varying and not separable anymore. For these cases, we propose to use least-squares approximation and projection-based sampling. Astonishingly, researchers have not yet looked at geometric image processing from the least-square point of view, except in [101].

1.2 Investigated approach

We would like to solve continuously-defined image processing problems in a more accurate way using modern concepts in sampling theory such as least-square approximation [7], projection-based sampling [96], multi-resolution techniques [106], and rate-distortion analysis for coding. We identified four different application fields that involve geometric image processing: medical imaging, visualization, computer graphics and image coding. We have chosen to concentrate our effort on the following computational tasks:

- *Tomographic reconstruction* with the Radon transform and filtered back-projection.
- *Volume rendering* based on the calculation of line integrals.
- *Texture mapping*, which is the computer graphics denomination for image-coordinate transformation or warping.
- *View-dependent compression*, where image or volume data are compressed for a specified geometric transformation such that losses are not visible after the geometric transformation.

While computational solutions exist for most of these problems, we show that there is still room for improvement, especially if one carefully addresses the discretization aspect of the problem.

1.3 Organization of thesis

After the two introducing chapters, the thesis is split into two parts that describe the new contributions of this thesis on two different aspects of geometric image transformations. In the first part (Chapter 3-4), we deal with X-ray like transformations that project images and volumes by line integral. In the second part (Chapter 5-7), we concentrate on texture mapping and compression. The more detailed outline of the thesis is as follows.

In Chapter 2 we introduces the advanced signal processing tools used in this thesis. They are B-spline signal processing, projection-based sampling, multiresolution, wavelets, scalable coding and rate-distortion theory.

In Chapter 3 we derive a discretization of the Radon transform and its inverse, the filtered back-projection algorithm, using convolution of multiple B-splines. To accelerate the algorithm, we store the Radon kernels in look-up tables. [46, 48, 104]

In Chapter 4 we improve the wavelet splatting method for volume rendering by introducing two new concepts: First, we improve the quality by projecting splats on the spline-modeled grid by least-square approximation, similar to Chapter 3. Second, we accelerate the method by adapting the grid's sampling step to the scale of the wavelets at coarse scale. [51, 52]

In Chapter 5 we review the standard texture mapping algorithms. We then reformulate the texture mapping problem such as to minimize the loss of texture information. We propose a novel algorithm based on successive refinement combined with multiresolution pyramids. [49, 50]

In Chapter 6 we take advantage of the information loss of texture mapping to compress the texture. This produces a view-dependent simplified texture that is a good approximation of the original one once it is mapped. We apply this concept to stream scaled texture data for textured models at very low bitrate. The view-dependent concept has been successfully brought to the MPEG-4 standard. [40, 43, 45, 55]

In Chapter 7 we present a framework for the joint simplification of texture and mesh with respect to an error measure in screen space. We combine two efficient techniques, view-dependent texture coding from Chapter 7 and quadtree mesh simplification, to search for the optimal trade-off between the resolution of the mesh and the texture under constraint resource (bit budget). We show that marginal analysis performs close to the rate-distortion optimal solution. [39, 40]

Finally, the global conclusions are drawn and recommendations for a continuation of this work are given.

1.4 Main contributions

The presented work is a novel approach in applying and combining recent research results from B-spline signal processing, sampling theory, wavelets and coding to improve image processing algorithms in the fields of computer tomography, volume visualization, texture mapping and texture coding.

Its contributions can be summarized by the following points:

- Presentation of a novel discretization of the Radon transform and of its inverse using spline convolution kernels.
- First application of fractional B-splines to tomography thanks to a formula that computes the ramp filtering analytically.
- Proposition of a novel multi-resolution acceleration technique for volume rendering based on the least-squares projection of wavelets (droplets).

- Presentation of a novel least-square optimal texture mapping method by forward mapping and successive refinement.
- Identification of view-dependent filters in the FT, DCT and wavelet domains taking into account the interpolation models.
- Bringing the view-dependent concept as feature to the MPEG-4 compression standard and beyond.
- Construction of multiresolution pyramids that are view-dependent.
- Proposal of marginal analysis as a tool to balance the bit budget between view-dependent coded meshes and textures in the rate-distortion sense.

Chapter 2

Advanced Signal Processing Methods

"In science one tries to tell people,
in such a way as to be understood by everyone,
something that no one ever knew before.
But in poetry, it's the exact opposite."
—PAUL DIRAC

2.1 Outline

This chapter reviews the advanced signal processing methods that form the theoretical basis for this thesis. Splines [98, 99] provide a powerful framework for solving the proposed class of image processing problems. Instead of pixels, the image is seen as a continuous, piecewise polynomial function, which is represented by a linear combination of B-spline functions. What makes splines especially attractive in this context is that most computations can be made using digital filters. In Section 2.2.1 we derive the B-spline functions of integer and fractional degrees. The spline formalism extends Shannon's sampling theory [86] and provides a unifying view of continuous/discrete signal processing by using least-squares approximation and projection-based sampling (Section 2.3). The multiresolution property of splines makes them prime candidates for constructing multiresolution pyramids [9] and wavelets [106] (Section 2.4). Multiresolution

decomposition and rate distortion analysis (Section 2.5.1) are the essential ingredients for the progressive compression of images.

2.2 B-splines

B-splines are the basic atoms for constructing spline functions. They are popular functions, especially in computer graphics, because of their compact support and intuitive handling.

2.2.1 B-splines of integer degree

We define the centered B-spline of degree n as the n-fold autoconvolution of the box function:

$$\beta^0(x) = \begin{cases} 1, & \text{for } -\frac{1}{2} \le x < \frac{1}{2}, \\ 0, & \text{otherwise,} \end{cases} \tag{2.1}$$

$$\beta^n(x) = \beta^0 * \beta^{n-1}(x) = \underbrace{\beta^0 * \cdots * \beta^0}_{n+1 \text{ terms}}(x). \tag{2.2}$$

We also define the scaled version of the B-spline of degree n at width h:

$$\beta_h^n(x) = \frac{1}{h}\beta^n\left(\frac{x}{h}\right). \tag{2.3}$$

Note that these functions are normalized to have a unit integral.

Let us also define the one-sided power function x_+^n as

$$x_+^n = \begin{cases} x^n, & x \ge 0 \text{ and } n > 0, \\ 1, & x \ge 0 \text{ and } n = 0, \\ 0, & \text{otherwise,} \end{cases} \tag{2.4}$$

and the centered $(n+1)$-th finite-difference operator as

$$\Delta^{n+1} = \underbrace{\Delta^1 * \cdots * \Delta^1}_{n+1 \text{ terms}} = \sum_{k=0}^{n+1}(-1)^k \binom{n+1}{k} \delta\left(x + \left(\frac{n+1}{2} - k\right)\right), \tag{2.5}$$

where $\Delta^1 = \delta(x + \frac{1}{2}) - \delta(x - \frac{1}{2})$; δ denotes the Dirac delta.

Using this notation, β^0 can also be written as the central finite difference of the Heaviside step function x_+^0:

$$\beta^0 = \Delta^1 * x_+^0 = \left(x + \frac{1}{2}\right)_+^0 - \left(x - \frac{1}{2}\right)_+^0.$$

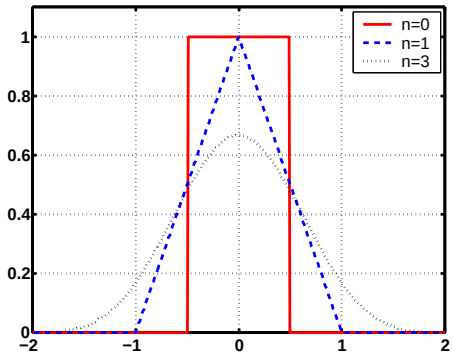


Figure 2.1: Centered B-splines of degree $n = 0, 1$ and 3

Thus we may rewrite β^n in terms of the following convolution operators:

$$\beta^n = \beta^0 * \cdots * \beta^0 = \left(\Delta^1 * x_+^0\right) * \cdots * \left(\Delta^1 * x_+^0\right).$$

Using the commutativity of convolutions operators, we get

$$\beta^n = \underbrace{\Delta^1 * \cdots * \Delta^1}_{n+1 \text{ terms}} * \underbrace{x_+^0 * \cdots * x_+^0}_{n+1 \text{ terms}},$$

which, as shown below, can be expressed in the more succinct form

$$\beta^n(x) = \Delta^{n+1} * \frac{x_+^n}{n!}, \tag{2.6}$$

where Δ^{n+1} and x_+^n are given by (2.5) and (2.4), respectively. The explicit form of the B-spline is thus:

$$\beta^n(x) = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} \frac{\left(x + \left(\frac{n+1}{2} - k\right)\right)_+^n}{n!}.$$

To illustrate (2.6), we depict the graphs of the B-splines of degrees $n \in \{0, 1, 3\}$ in Figure 2.1.

Proof of eq.(2.6): First, it is easy to establish that

$$x_+^{n-1} * x_+^0 = \int_0^x x_+^{n-1} dx = \frac{x_+^n}{n}.$$

By induction, we find that

$$\underbrace{x_+^0 * \cdots * x_+^0}_{n+1 \text{ terms}} = \frac{x_+^n}{n!}. \tag{2.7}$$

Next, we consider the convolution of iterated finite difference operators

$$\Delta^{n+1} = \underbrace{\Delta^1 * \cdots * \Delta^1}_{n+1 \text{ terms}}.$$

The transfer function (Fourier transform) of this convolution operator is

$$\widehat{\Delta}^{n+1}(\omega) = \left(e^{j\omega/2} - e^{-j\omega/2}\right)^{n+1},$$

which can be expanded to yield

$$\widehat{\Delta}^{n+1}(\omega) = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} e^{j\omega \cdot \left(\frac{n+1}{2} - k\right)}. \tag{2.8}$$

The explicit time-domain formula (2.5) is then obtained by interpreting the complex exponentials of (2.8) as time shifts. Combining (2.7) and (2.5) then yields (2.6). □

It follows from (2.1) that the Fourier transform of the centered B-spline is given by

$$\widehat{\beta}^n(\omega) = \text{sinc}^{n+1}\left(\frac{\omega}{2\pi}\right) = \left(\frac{\sin(\omega/2)}{\omega/2}\right)^{n+1} = \left(\frac{e^{j\omega/2} - e^{-j\omega/2}}{j\omega}\right)^{n+1}. \tag{2.9}$$

2.2.2 Fractional B-splines and derivatives

We now describe how to extend the B-spline concept to non-integer or fractional degrees α [102]. The causal, symmetric B-spline of degree α is defined in the Fourier domain by

$$\hat{\beta}_*^\alpha(\omega) = \frac{\left|1 - e^{-j\omega}\right|^{\alpha+1}}{|j\omega|^{\alpha+1}}. \tag{2.10}$$

This definition is essentially the same as (2.9). The main difference is that $\alpha = n$ is allowed to be non-integer. Note that we have a true equivalence only when α is odd, in which case the fractional B-spline is compactly supported. Otherwise it decays like $o\left(\frac{1}{x^{\alpha+2}}\right)$.

Here, we are especially interested in the fractional derivative properties of the fractional B-splines. The n-th derivative of the function $f(x)$ is defined in the Fourier domain as

$D^n f(x) \leftrightarrow (j\omega)^n \hat{f}(\omega)$, where $\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x) e^{-j\omega x} dx$ denotes the Fourier transform of $f(x)$. By extension, we define a symmetric version of fractional derivatives

$$D_*^\gamma f(x) \leftrightarrow |\omega|^\gamma \hat{f}(\omega), \tag{2.11}$$

where γ is any real number. Note that this derivative only corresponds to the usual one when γ is even.

The relevance for our purpose of the D_* operator is that it is a scaled version of the ramp filter $\hat{q}(\omega) = |\frac{\omega}{2\pi}|$, which plays a crucial role in tomography (cf. Chapter 3 and [46]). The key property is that we have a simple analytical formula for the fractional derivative of a fractional B-spline

$$D_*^\gamma \beta_*^\alpha(x) = \Delta_*^\gamma * \beta_*^{\alpha-\gamma}(x), \tag{2.12}$$

where Δ_*^γ is the fractional finite difference operator

$$\Delta_*^\gamma \leftrightarrow |1 - e^{-j\omega}|^\gamma.$$

The argument for the proof is as follows

$$D_*^\gamma \beta_*^\alpha(x) \leftrightarrow |j\omega|^\gamma \left| \frac{1 - e^{-j\omega}}{j\omega} \right|^{\alpha+1} = \left|1 - e^{-j\omega}\right|^\gamma \left| \frac{1 - e^{-j\omega}}{j\omega} \right|^{\alpha+1-\gamma}.$$

2.3 Projection-based sampling

Continuous-time signals have to be sampled before they can be processed digitally. The spline formalism described in [94] provides an attractive alternative to the classical formulation dictated by Shannon's sampling theorem [86]. The main idea is to use our basic atoms—e.g. the compactly supported B-splines—which are much more convenient to handle then the ideal sinc function which decays rather slowly. A good reason for using polynomial splines over other forms of interpolation is that they offer the best cost-performance tradeoff; this is a property that is now well documented in the literature [64, 92]. In particular, we refer to the extensive work of Meijering [72] who compared as many as 126 interpolators and concluded that splines were significantly better in all cases. One theoretical explanation for this superior performance is that the B-spline of degree n is the shortest and smoothest function that allows for reproduction of polynomials of degree n [6]. This polynomial reproduction property is essential in wavelet and approximation theory; it determines the approximation order ($L = n + 1$); i.e., the rate of decay of the approximation error as a function of the sampling step h [7, 103].

2.3.1 Polynomial splines

Here, we will consider polynomial splines of degree n defined on a uniform grid with step size h. These functions are made up of polynomial segments of degree n and of size h which are

joined together in a way that guarantees the continuity of the function and its derivatives up to order $n-1$. A fundamental result is that any such spline with spacing h can be represented in terms of its B-spline expansion [84]

$$f_h(x) = \sum_{k\in\mathbb{Z}} c(k)\beta_h^n(x-hk), \tag{2.13}$$

where the basis functions $\{\beta_h^n(x-hk)\}$ are the shifted versions of the B-splines at scale h defined in Section 2.2.1, eq.(2.3). The $c(k)$'s are the so-called B-spline coefficients; note that there is exactly one such coefficient per grid point.

The basic question in B-spline signal processing is how to determine the expansion coefficients in (2.13). There are essentially two approaches: interpolation and projection-based approximation.

2.3.2 Spline interpolation

This is the approach of choice when the signal is represented by its integer samples $f(k),\ k\in\mathbb{Z}$. It is then possible to determine the $c(k)$'s such that the model (2.13) interpolates the samples $f(k)$ exactly. This involves the solution of a linear system of equation. This problem is solved most effectively using the recursive filtering algorithm described in [94, 97, 98].

Spline interpolation amounts to fitting a sequence $f(k)$ with a spline of the form (2.13). The $c(k)$'s are determined by inverse filtering (cf. [94]):

$$c(k) = \left((b^n)^{-1} * f\right)(k), \tag{2.14}$$

where

$$(b^n)^{-1}(k) \leftrightarrow \frac{1}{\sum_{k\in\mathbb{Z}} \beta^n(k)e^{-j\omega k}} = \frac{1}{\sum_{l\in\mathbb{Z}} |\mathrm{sinc}(\frac{\omega}{2\pi}+l)|^{n+1}}.$$

By applying this spline interpolation to a Kronecker delta (unit impulse at origin), one obtains the so-called cardinal spline function which is denoted by $\varphi(x) = \beta_{\text{int}}^n(x)$. It has the interesting property that the cardinal spline coefficients are simply the sampled values of the continuous function: $c(k) = f(k)$. It is an interpolator of infinite support which resembles the sinc function [2]. This is another example of a valid spline basis functions. With this particular basis function in mind, one can interpret the B-spline interpolation algorithm (recursive filtering) as a change from the cardinal spline basis (where the samples are the coefficients) to the B-spline basis.

2.3.3 Least squares spline approximation

This is a refinement of straightforward sampling; it is applicable whenever the input function $f(x)$ is continuously defined. Here, the goal is to get the best spline representation $f_h(x)$ of $f(x)$ so that the integral of the square difference between $f(x)$ and $f_h(x)$ (squared L_2-norm)

is minimized. Mathematically, this corresponds to the orthogonal projection of $f(x)$ onto the spline space with step size h.

The optimal expansion coefficients are determined simply from the inner product with the dual basis functions (cf. [95]):

$$c(k) = h \left\langle f(x), \widetilde{\beta}_h(x - hk) \right\rangle . \tag{2.15}$$

The basis function $\beta(x)$ and its dual $\tilde{\beta}(x)$ (which is unique) form a bi-orthogonal basis system that fulfills the bi-orthogonality condition $\langle \beta(x), \widetilde{\beta}(x-k) \rangle = \delta(k)$. Note that the computation of the inner products (2.15) is akin to applying a prefilter to $f(x)$ and sampling thereafter. The main advantages of least-squares approximation over interpolation and sampling are a closer fit and the suppression of aliasing artifacts.

Interestingly, the role of basis functions and duals can be interchanged. This means that by performing the inner product between $f(x)$ and the B-splines $\{\beta_h^n(x-hk)\}$, one obtains the least squares coefficients of $f_h(x)$ in terms of the dual B-splines $\{\tilde{\beta}_h^n(x-hk)\}$. This is a property that will be used in the several of our algorithms because we know how to compute inner products with B-splines analytically. It is then possible to go back to the B-spline basis by performing a change of basis, which is equivalent to a digital filtering operation [99].

2.3.4 Least-squares sampling

The previous least-squares scheme also works for other functions that are not necessarily splines. It is possible to consider a signal representation in terms of more general shift-invariant basis functions generated from shifting a rescaled version of φ:

$$f_h(x) = \sum_{k \in \mathbb{Z}} c(k) \varphi \left(\frac{x}{h} - k \right), \tag{2.16}$$

where the expansion coefficients $c(k)$ are computed as

$$c(k) = \frac{1}{h} \left\langle f(x), \widetilde{\varphi} \left(\frac{x}{h} - k \right) \right\rangle \tag{2.17}$$

and where $\widetilde{\varphi}(x) \in V(\varphi)$ is the dual of $\varphi(x)$ [99]. The function space $V(\varphi)$ is the approximation space spanned by the integer shifted versions of the base function φ. The dual $\widetilde{\varphi}(x)$ is uniquely defined by

$$\widehat{\widetilde{\varphi}}(\omega) := \frac{\hat{\varphi}(\omega)}{\sum_{k \in \mathbb{Z}} |\hat{\varphi}(\omega - 2\pi k)|^2}, \tag{2.18}$$

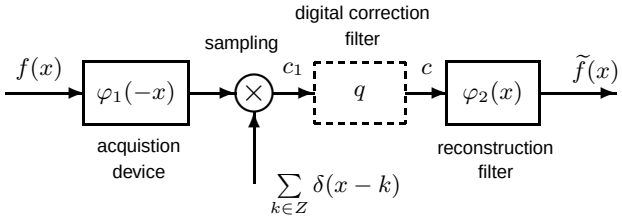


Figure 2.2: Sampling for non-ideal acquisition devices

which is biorthogonal to $\varphi(x)$. The first condition for the sampling expansion to be a faithful representation of $f(x)$ is that $\varphi(x)$ must generate a Riesz basis, which implies that there must exist two strictly positive constants A and B such that

$$A \leq \sum_{k\in\mathbb{Z}} |\hat{\varphi}(\omega + 2\pi k)|^2 \leq B. \tag{2.19}$$

The second condition is that $\varphi(x)$ satisfies

$$\sum_{k\in\mathbb{Z}} \varphi(x+k) = 1. \tag{2.20}$$

Indeed, one can prove that (cf. [95])

$$\lim_{h\to 0} f_h(x) = f(x)$$

in the L_2-sense, if and only if (2.20) is satisfied. This condition (2.20) is called the partition of unity. It plays a central role in wavelet theory. It is satisfied by B-splines, including the fractional ones with $\alpha \geq 0$.

Shannon's sampling theorem states that a band-limited function $f(x)$ can be reconstructed perfectly from its samples $f(hk)$, provided that the sampling step h is sufficiently small $h\omega_0 \leq \pi$, where ω_0 is the highest frequency of f. But real signals can not be truly band-limited[1]. The analysis function for band-limited signals (after Shannon) is $\varphi(x)$ =sinc(x); since it can be shown to be orthogonal, it is its own dual. Thus, the interpretation of (2.17) is that we need to convolve f with sinc$(\frac{x}{h})$ prior to resampling, which corresponds to low-pass filtering.

2.3.5 Oblique projection

In practice, the sampling prefilter is often specified a priori by the acquisition device, and not necessarily optimal. We will assume that the measurement of a function f is obtained by sampling its prefiltered version, $f * h$, which is equivalent to calculating the inner products

[1]On the one hand, real signals are time-limited, as they start and stop. Therefore they cannot be band-limited. One the other hand, real systems cannot vibrate at infinite frequency, ergo real signals are band-limited. What do you think they are? Find the answer to this paradox in [89].

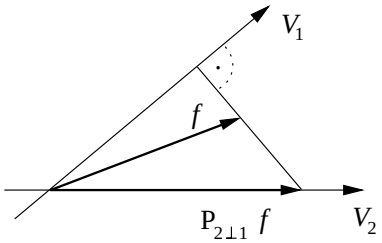


Figure 2.3: Principle of oblique projection onto $V(\varphi_2)$ perpendicular to $V(\varphi_1)$

$$c_1(k) = \langle f, \varphi_1(x-k) \rangle \,,$$

with analysis function $\varphi_1(x) = h(-x)$. We want to construct a representation of the form (2.16) with synthesis function $\varphi_2(x)$. The solution is to apply a suitable digital filter q as shown in Figure 2.2. We seek a signal approximation that, if re-injected into the system, provides the same solution. First we have to define the cross-correlation sequence:

$$a_{12}(k) = \langle \varphi_1(x-k), \varphi_2(x) \rangle \,.$$

Theorem 2 from [96] states: Let $f \in L_2$ be an unknown input function. Provided $\left|A_{12}(e^{j\omega})\right|^2 \geq m > 0$, then there exists a unique signal approximation $\tilde{f}$ in $V(\varphi_2)$ that is consistent with f in the sense that

$$\forall f \in L_2,\, c_1(k) = \langle f, \varphi_1(x-k) \rangle = \langle \tilde{f}, \varphi_1(x-k) \rangle.$$

The signal approximation is given by

$$\tilde{f} = \mathrm{P}_{2\perp 1}\{f(x)\} = \sum_{k \in \mathbb{Z}} (c_1 * q)(k) \cdot \varphi_2(x-k),$$

where the filter q is

$$q = (a_{12})^{-1}(k) \leftrightarrow Q(z) = \frac{1}{\sum_{k \in \mathbb{Z}} a_{12}(k) z^{-k}}.$$

The underlying projector $\mathrm{P}_{2\perp 1}$ is a projector from L_2 onto $V(\varphi_2)$ perpendicular to $V(\varphi_1)$ [96]. That is why this projection is called oblique projection (see Figure 2.3).

Type	Symbol	$P(e^{j\omega}) = \sum_{k\in\mathbb{Z}} p(k)e^{-j\omega k}$	Property
Basic	$\varphi = \beta$	1	Admissibilty $A \leq \left\|A_\varphi(e^{j\omega})\right\| \leq B$
Cardinal or interpolating	$\eta = \beta_{\text{int}}$	$\frac{1}{\sum_{k\in\mathbb{Z}} \hat{\beta}(\omega-2\pi k)}$	Interpolation $\beta_{\text{int}}(k) = \delta_k$
Orthonormal	$\phi = \beta_{\text{ortho}}$	$\frac{1}{\sqrt{\sum_{k\in\mathbb{Z}} \left\|\hat{\beta}(\omega-2\pi k)\right\|^2}}$	Orthonormality $\langle\Phi(x), \Phi(x-k)\rangle = \delta_k$
Dual	$\tilde{\beta} = \beta_{\text{dual}}$	$\frac{1}{\sum_{k\in\mathbb{Z}} \left\|\hat{\beta}(\omega-2\pi k)\right\|^2}$	Biorthonormality $\langle\tilde{\beta}(x), \beta(x-k)\rangle = \delta_k$

Table 2.1: Types of equivalent generating B-spline functions with their specific properties [95].

2.3.6 Equivalent spline basis

So far, we have encountered three types of B-spline functions: the basis ones (β), the duals $(\tilde{\beta})$ and the interpolating ones (β_{int}). In fact, we can construct many other equivalent spline functions of the form $\varphi_{\text{eq}} = \sum_{k\in Z} p(k)\beta(n-k)$, where $p(k)$ is an appropriate sequence of weights. The necessary and sufficient condition for $\{\beta(x-k)\}_{k\in\mathbb{Z}}$ to yield an equivalent Riesz basis $V(\beta)$ are that there exist two positive constants A and B such that $A \leq \left|P(e^{j\omega})\right|^2 \leq B$, where $P(e^{j\omega})$ is the Fourier transform of $p(k)$. The most important B-spline functions are summarized in Table 2.1. Note that the orthogonalized version β_{ortho} plays a specific role in wavelet theory [69] (see Section 2.4).

Some comments on the usefulness of the various presentations: If one is only concerned with the signal value at integers, then the cardinal representation is the most adequate one. If, on the other hand one wishes to perform computations involving values in between samples, the B-spline representation has the advantage of a signal expansion as in (2.16) with minimum support. This is especially true in higher dimensions where the cost of prefiltering is small in comparison to the computation of the expansion formula. An attractive feature of the above formulation is that the role of the analysis and synthesis function is interchangeable. For example, in parts of our algorithms (Chapter 3), fast prefiltering will be crucial; there, the dual spline representation will be a good choice as its analysis function (β) is of minimum support.

2.3.7 Error analysis and evaluation

The reconstruction of f in eq.(2.16) is not necessarily perfect, even though it improves as h gets smaller. We can estimate the least-square reconstruction error ε_h as a function of the sampling step h, and the Fourier transforms $\hat{f}(\omega)$ of the original function $f(x)$ (cf. [7]):

$$\varepsilon_h = \|f - f_h\|^2 = \frac{1}{2\pi}\int_{-\infty}^{\infty} E(h\omega)|\hat{f}(\omega)|^2 d\omega \tag{2.21}$$

where the error kernel $E(\omega)$ depends on $\hat{\varphi}(\omega)$, the Fourier transform of the basis function $\varphi(x)$, by

$$E(\omega) = 1 - \frac{|\hat{\varphi}(\omega)|^2}{\sum_{k\in\mathbb{Z}} |\hat{\varphi}(\omega + 2\pi k)|^2}.$$

This reconstruction error can serve as a criterion for comparing different signal representations. It may also be used to optimize the basis function [6]. It is a powerful tool to design more accurate algorithms, which are scalable (wavelets) and often more efficient (short support basis functions).

Least-squares is dead—long lives least-squares

It is a well-known fact in image processing that there does not yet exists a standard image quality measure that reflects the human perception of images and errors in images. So far the established standard error measure is a normalized square norm, the so-called peak signal-to-noise ratio (PSNR) expressed in decibels:

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}(e(k,l))^2}\right), \tag{2.22}$$

where $e(k,l)$ denotes the error between the original image and the reconstructed image (cardinal representation); both images are of size $N \times N$.

We discuss whether the least-squares norm is consistent with the notion of image quality used in this thesis' application fields.

1. Image compression [54, 106], where the *rate-distortion curve* is the basis to evaluate lossy, non-linear approximation methods. The problem of accurate evaluation of image distortion is still unsolved. The mainstream approach is to use the least-squares norm (L_2-norm). Potential alternatives are other L_p-norms, such as the absolute mean (L_1), maximum error (L_∞), or median norm. Lossless compression techniques (based on information theoretical tools) are applied to minimize remaining *redundancy.*

2. Medical imaging [1] has the highest demand for *reliable* and *meaning-full* images. This is achieved by image enhancement and artificial visualization clues. Processing the images in the least-squares optimal sense achieves the best measurable quality.

3. Computer graphics [19, 29, 65, 80, 81] is based on the human fascination for *esthetic* or *realistic* images, which is even harder to evaluate. Flawlessness of images might be the key; our least-squares approach reduces image *artifacts* like aliasing, flickering or blurring.

In conclusion the least-squares norm is the most commonly used image error measure.

2.4 Multiresolution representations

2.4.1 Image pyramids

Burt and Adelson [9] described the first multiresolution pyramids for coding. The image $g_0(\boldsymbol{x})$ is convolved with local smoothing functions (Gaussian, tent function, etc.) generating a low-pass filtered image $g_1(\boldsymbol{x}) = h(\boldsymbol{x}) * g_0(\boldsymbol{x})$ which is then subtracted from the original $l_1(\boldsymbol{x}) = g_0(\boldsymbol{x}) - g_1(\boldsymbol{x})$. This process is iterated by $g_{i+1}(\boldsymbol{x}) = h(\boldsymbol{x}) * g_i(\boldsymbol{x})$ and $l_{i+1}(\boldsymbol{x}) = g_i(\boldsymbol{x}) - g_{i+1}(\boldsymbol{x})$. It produces a multiresolution pyramid $\{g_i(\boldsymbol{x})\}$ called the *Gaussian pyramid* and a pyramid of difference images $\{l_i(\boldsymbol{x})\}$, the so-called *Laplacian pyramid.*

These pyramids are suitable for coding. The redundancy of images in the Gaussian pyramid is gradually removed from scale to scale. Progressive transmission starts by sending a coarse version $g_j(\boldsymbol{x})$ to give a first impression of the view. Then, little by little, more details come apparent by transmitting finer scales $l_i(x)$ of the Laplacian pyramid $g_{i-1}(\boldsymbol{x}) = g_i(\boldsymbol{x}) + l_i(\boldsymbol{x})$.

In computer graphics, similar multiresolution pyramids of textures at dyadic scales—the so called *Mipmap pyramids* [109]—are precalculated and stored. They allow to compute fast interpolated texture values; values at an intermediate scale are interpolated from the two closest Mipmap levels.

The downside of the pyramidal representation is its redundancy; it adds an overhead of $1/3$ compared to the size the original image or $1/7$ in the case of volumetric data.

More advanced image pyramids carefully adapt the smoothing functions to the representation space, e.g. to the continuous L_2-space in the case of L_2-pyramids [100], and to the discrete l_2-space in the case of l_2-pyramids [98]. Centered pyramids appear to be more suitable for computer vision, e.g. for image registration [93] or motion estimation. Non-centered pyramids are closely related to the wavelet decomposition.

2.4.2 Wavelet decomposition

The big advantage of the multiresolution representation with wavelets is its lack of redundancy; no overhead is added. The scaling function $\varphi(x)$ plays a crucial role in wavelet theory. A valid scaling function $\varphi(x)$ must fulfill the conditions of generating functions (Riesz basis (2.19) and partition of unity (2.20)), and, in addition, the two-scale relation

$$\varphi(x/2) = \sqrt{2} \sum_{k \in \mathbb{Z}} h(k) \varphi(x - k) \tag{2.23}$$

where $h(k)$ is the so-called refinement filter. In other words, the dilated version of φ must live in the same space as φ. Wavelet theory deals with a ladder of rescaled subspaces $V_i = V_{T=2^i}(\varphi) = \text{span}\{\varphi_{i,k}\}$ where the scaled and dilated version of φ is

$$\varphi_{i,k} = 2^{-i/2} \varphi(x/2^i - k).$$

If φ satisfies the two-scale relation, then these spaces are nested and form a multiresolution analysis of L_2. The wavelet function $\psi(x)$, which may be written as a linear combination of shifted φ's by

$$\psi(x/2) = \sum_{k\in\mathbb{Z}} g(k)\varphi(x-k),$$

is designed to generate a Riesz basis of the difference space $W_i = V_{i-1} - V_i$; these are also rescaled versions of each other but their pairwise intersections are all $\{0\}$, in contrast with the V_i's which are included hierarchically. Since the multiresolution is dense in L_2, any function $f(x) \in L_2$ can be represented by its wavelet expansion

$$f(x) = \sum_{k\in\mathbb{Z}} \sum_{i\in\mathbb{Z}} \left\langle f, \tilde{\psi}_{k,i} \right\rangle \psi_{k,i}, \tag{2.24}$$

where k and i are the position and the scale indices, respectively. The wavelet expansion works because the analysis and synthesis function form a biorthogonal basis of L_2 such that $\langle \psi_{k,i}, \psi_{l,j} \rangle = \delta_{k-l,i-j}$. For a detailed discussion of wavelets see the standard works of Daubechies [14], Mallat [70], Strang & Nguyen [91], and Vetterli & Kovačević [106].

2.5 Scalable coding

2.5.1 Rate-distortion analysis

Rate distortion theory [12] is useful to manage a limited amount of resources (rate) to achieve the best possible result (minimal distortion). Ideally more effort improves the results. Unfortunately, this is not always true for real problems. Still, if the problem has a concave[2] and monotonic decreasing rate-distortion (R-D) curve, then following this curve ensures the best trade-off between resource and result.

The slope of the R-D curve is proportional to the quality improvement achieved by a fixed amount of effort. This has implication on real problems that can be modeled by such curves. For example, a student has to learn for two exams. The result is the overall performance in both exams. He has to assign his limited time to study for each of the two exams. For each exam a R-D curve can be drawn and the current studying status corresponds to a point on the R-D curve. Most likely the slopes on the two R-D curves will be different. Learning for the exam with the higher gradient is the best choice. With time the learning progress will slow down and the slope of its R-D curve flattens. If it is smaller than the one for the other exam, then the student swaps subjects. This algorithm can go on until the time is up or the result is sufficient (see Figure 2.4).

The same principle applies to coding of signals and images. The decision of how to allocate bits is taken best in the rate-distortion optimal sense. It consists in coding first the coefficients that

[2] The second derivative of the curve is positive.

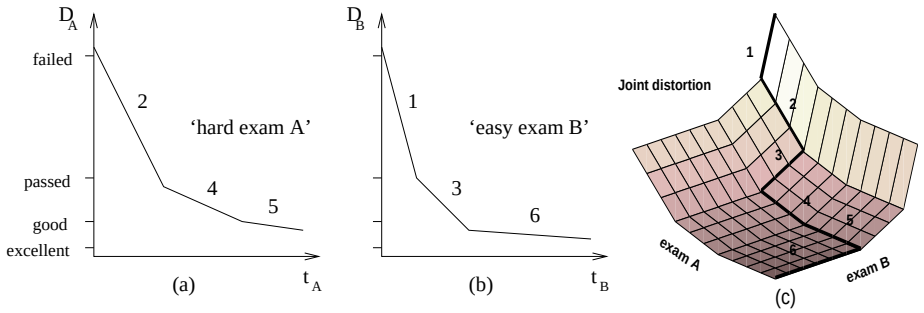


Figure 2.4: Rate-distortion optimal allocation of limited resources: Example of a student that studies for two exams. The time is the limited resource (rate), and the exam's result is reciprocal to the distortion. The best strategy is to study for the exam with the steepest slope. The line segments on the two R-D curves are enumerated in the optimal order (monotonic decreasing gradients). (a) R-D curve for hard exam A, e.g., Applied electrodynamics. (b) R-D curve for the easy exam B, e.g., Biology. (c) Joint R-D curve of both exams.

contribute most to the quality improvement of the reconstructed signal. For a well-conditioned problem, a local analysis of the R-D curve is called *marginal analysis*. There, a small bit budget is spent on all reasonable possibilities; the one with the best result is chosen and so forth. In Chapter 7 we will apply this principle for jointly coding meshes and textures.

2.5.2 Coding

We can distinguish between lossless coding and lossy coding. The first one, also called entropy coding, reconstructs the original digital signal perfectly. Entropy coding assigns the shortest codes to the most likely signal values. Given the probabilities p_i of each signal value x_i the entropy H is defined by $H = -\sum_i p_i \log_2 p_i$, measured in bits. It is the lower bound for the average length of the shortest description of a signal produced by a identically independent distributed source [12].

Image compression methods reduce the redundancy of an image by extracting the principal components of the image, e.g. by Fourier transforms (DCT) or subband decompositions (wavelets). Lossy coding selects only the most important components in the rate distortion sense. It then applies entropy coding to them, as well as a few other engineering tricks that make the code more concise (quantization, zig-zag scan, zerotree, etc.).

2.5.3 Progressive transmission

We have discussed two multiresolution representations: the image pyramid and the wavelet decomposition. Both allow for progressive transmission of images, first at coarse then at finer scale.

The non-redundancy of the wavelet representation makes a fine scale granularity possible and allows to create compact, progressive bitstreams. The compression can be optimally controlled by rate-distortion analysis.

We distinguish between four types of scalability:

- *Quantization scalability* permits to adjust the precision with which a coefficient is represented. The precision can be increased by adding more bits, which increases the number of quantization levels.
- *Spatial scalability* of images allows to view them already at lower resolution, while additional image details are not yet received.
- *Temporal scalability* of video signals allows to drop frames and to receive videos at lower frame rate.
- *View-dependent scalability* permits to transmit only the information that is necessary to render a three-dimensional scene from the user´s viewpoint; e.g., occluded objects or back faces of surfaces are invisible and thus dispensable.

The common setup in telecommunication is that data is transmitted through a channel from one source to one receiver. There are numerous variants of this setup, e.g. a server-to-client model or a peer-to-peer model, one-to-one or one-to-many connections, connection-based (telephone line) or packet-switched (Internet network) transmission. The quality-of-service may vary from application to application. In Chapter 6 we will stream 3D scenes, described by textured 3D models over the Internet. There we can imagine two basic communication scenarios. In the first, the user requests data for a specific scale—scale in the sense of the four scalabilities. The server compresses the data according to the request and streams it to the user. In the second one, one server sends the data to several receivers via multicast; in a multicast[3] communication scenario the data is sent in a tree-like structure from the sender (the root) over the network (the branches) to the receivers (the leaves). Each receiver extracts the information needed to decode the scene for its required scale. An intelligent network may forward only the information needed at the receivers; details not requested by child nodes are stripped off in the parent nodes to save bandwidth.

2.6 Summary

In this chapter we have introduced several advanced signal processing tools, namely B-splines, projection-based sampling, multiresolution, wavelets, rate-distortion theory, and scalable coding. In the next chapters, we will extend and apply these concepts to geometric transformations, projections and warps of images and volumes.

[3] one to many

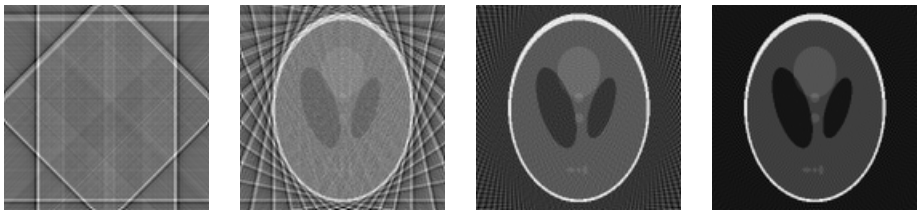
Chapter 3

Discretization of the Radon transform

"The experiment is the source of all certitude."
—LEONARDO DA VINCI

3.1 Introduction

The Radon transform of a two-dimensional function is given by the collection of its line-integrals (or projections); each ray is indexed by its distance t to the origin and its angle θ [75, 68] (see



4 angles 16 angles 64 angles 128 angles

Figure 3.1: The inverse Radon transform with increasing number of projection angles

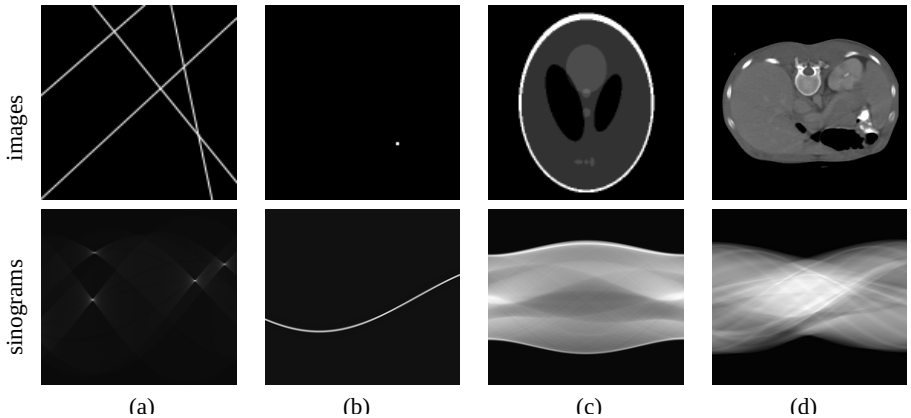


Figure 3.2: Radon transform of images: (a) Lines become points in the Radon domain and (b) points become sinusoids—therefore the name sinogram. (c) The Shepp-Logan phantom. (d) Intersection of the human torus.

Figures 3.1-3.3). The Radon transform plays a crucial role in medical imaging because it constitutes a good model of the tomographic acquisition process [15, 34, 73]. For instance, the forward step—or re-projection—that is required in most iterative reconstruction algorithms is equivalent to the computation of a Radon transform. Re-projection is also used explicitly for beam-hardening correction [58], streak suppression [32], and the removal of artifacts caused by the presence of radio-opaque material such as metallic implants [22]. Other applications of the Radon transform in image processing are the detection of lines (like the Hough transform) [23], and the recently proposed ridgelet transform [10], which is essentially a wavelet transform applied in the Radon domain.

An attractive feature of the continuously-defined Radon transform is that it has an exact inversion formula [68]. The digital implementation of this analytical formula leads to the standard filtered back-projection (FBP) algorithm, which goes back to the early 70s [76]. Despite the considerable research efforts devoted to alternative reconstruction techniques—in particular, algebraic (ART) [34] and statistical ones, including maximum likelihood [87] and Bayesian [8, 25, 28, 33]—the FBP is still the method of choice used in commercial CT scanners. It owes its success to the fact that it is direct, fast and reasonably simple to implement. Even though the standard implementation uses a rather rudimentary discretization—at least by modern standards, it has not been much improved over the years, except for the aspect of filter design [88]. One noteworthy exception is the work of Guédon et al. who derived an optimal reconstruction filter based on a piecewise-constant model of the image [27]. Some wavelet approaches can also be viewed as multi-scale variations on FBP [16, 74, 77].

The practical computation of the Radon transform or of its inverse necessarily involves some form of interpolation because the underlying mathematical objects are defined in the continuous

domain. The same also holds true for Fourier-based reconstruction techniques [24, 85, 107]. For efficiency reasons, practitioners tend to use relatively simple interpolation techniques such as piecewise constant or linear. One of the arguments that is often made in favor of the FBP is that it is less sensitive to interpolation errors than direct Fourier-based reconstruction [59, 90]. Thus, there appears to be a general perception that the role of interpolation is not predominant. Our purpose in this chapter is to investigate this issue in greater details and determine the extent to which the use of high quality interpolation models together with an optimal discretization can improve performance. Indeed, we will see that a careful design—i.e., the use of a good model and a sampling method that is optimal in the least squares sense—can make quite a difference. We have chosen to base our approach on splines because this type of representation offers the best cost-performance tradeoff for interpolation; this is a finding that was confirmed recently by three research groups in medical imaging [64, 72, 92].

The novelty of our approach, which uses splines both in the image and Radon domains, is fourfold: First, it is applicable in both directions for the computation of the Radon transform and of its inverse; in fact, the direct and inverse algorithms are duals of each other—they both use the same Radon kernels. Second, we explicitly fit a continuously-defined spline model to the sinogram (resp. to the image), instead of determining the values by resampling, as is done usually. Third, we select the spline basis function in a way that allows for an explicit solution with B-spline convolution kernels and therefore an optimal discretization of the Radon transform. Finally, the method is general and works for splines of any degree n. Choosing a larger degree (typ., $n = 3$) results in better quality but it also implies more computation. Thus, there is a compromise to be found.

This chapter is organized as follows: In Section 3.2, we derive an explicit formula of the spline convolution kernels and use it to derive our spline-based implementation of the Radon transform which is optimal in the least squares sense. In Section 3.3, we introduce the corresponding version of the FBP algorithm which uses spline kernels as well. In Section 3.4, we discuss implementation issues for both algorithms (direct and inverse). Finally, in Section 3.5, we present experimental results. In particular, we use the composition of back and forth transforms (Radon transform followed by its inverse) to investigate the influence of the various parameters (degree of the spline, spatial and angular sampling steps) on overall performance. We determine the best trade-off between computation time and image quality.

While we believe that our spline-formulation of the FBP is original, we are aware of three other instances where splines have been used for tomographic reconstruction. The first is a finite-element formulation of the problem using splines as basis functions [63]; it is essentially an algebraic method that requires the solution of a large system of linear equations. The second applies box splines—a multi-dimensional extension of B-splines—for improved approximation in an algorithm based on Fourier reconstruction [78]. The third is an approach where the ramp-filtering is evaluated analytically based on the continuously-defined derivative of the projection data [79]. Lastly, it is interesting to note that the benefit of bi-cubic interpolation—albeit not splines—has been demonstrated indirectly in rotation-based methods for iterative reconstruction [5].

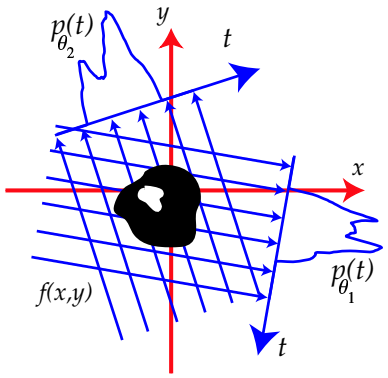


Figure 3.3: The Radon transform: Parallel projections of the object are depicted at two different angles.

3.2 Spline-based Radon transform

3.2.1 The Radon transform

The *Radon transform* $R_\theta f$ [75, 68] of an image $f(\boldsymbol{x})$, $\boldsymbol{x} = (x, y) \in \mathbb{R}^2$, is the set of line integrals along the direction $\boldsymbol{\theta}$ at the distance t from the origin

$$(R_\theta f)(t) = R_\theta\{f(\boldsymbol{x})\} = \int_{\mathsf{x}\in\mathbb{R}^2} f(\boldsymbol{x})\delta(t - \boldsymbol{x}^\top \cdot \boldsymbol{\theta})d\boldsymbol{x}, \tag{3.1}$$

where $\delta(t)$ is the Dirac impulse and $\boldsymbol{\theta} = (\cos\theta, \sin\theta)$ is a unit vector specifying the direction of the integration (see Figure 3.3). Note that R_θ, which is a Radon transform for one projection angle θ, is also called the *projection operator*; R on its own stands for the complete Radon transform with all angles θ.

The Radon transform is clearly a linear operator. Thus, if we represent the initial image as a sum of 2D basis functions (B-splines), we can compute its Radon transform exactly provided we know the Radon transform of the initial basis functions. Moreover, if we know the Radon transform of one prototype $\varphi(\boldsymbol{x})$, we can easily derive the Radon transform of all basis functions because of the shift-invariance property $R_\theta\{\varphi(\boldsymbol{x} - \boldsymbol{x}_0)\} = (R_\theta\varphi)(t - t_0)$ where $t_0 = \langle\boldsymbol{\theta}, \boldsymbol{x}_0\rangle$.

3.2.2 B-spline convolution kernels

To discretize the Radon transform using splines, we will need to convolve several B-splines of variable widths (see Figure 3.4). First, we define the scaled version of the B-spline of degree n

at width h

$$\beta_h^n(x) = \Delta_h^{n+1} * \frac{x_+^n}{n!} \tag{3.2}$$

in terms of the one-sided power function x_+^n defined in (2.4), and the centered $(n+1)$-th finite-difference operator scaled at width h:

$$\Delta_h^{n+1} = \underbrace{\Delta_h^1 * \cdots * \Delta_h^1}_{n+1 \text{ terms}} = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} \frac{\delta(x + h \cdot (\frac{n+1}{2} - k))}{h^{n+1}}. \tag{3.3}$$

The spline bi-kernel $\beta_{h_1,h_2}^{n_1,n_2}(x)$ is defined as the convolution of two B-splines of degrees n_1, n_2 and widths h_1, h_2:

$$\beta_{h_1,h_2}^{n_1,n_2}(x) = \beta_{h_1}^{n_1} * \beta_{h_2}^{n_2}(x). \tag{3.4}$$

In the same way, we define the spline m-kernel, which is the convolution of m different B-splines of degrees $n_1, \ldots, n_m$ and widths $h_1, \ldots, h_m$, by

$$\beta_{h_1,\ldots,h_m}^{n_1,\ldots,n_m}(x) = \beta_{h_1}^{n_1} * \cdots * \beta_{h_m}^{n_m}(x).$$

Proposition 3.1: The spline m-kernel can be computed as

$$\beta_{h_1,\ldots,h_m}^{n_1,\ldots,n_m}(x) = \Delta_{h_1,\ldots,h_m}^{n_1+1,\ldots,n_m+1} * \frac{x_+^{N_m}}{N_m!}, \tag{3.5}$$

where $N_m = m - 1 + \sum_{i=1}^m n_i$ and where $\Delta_{h_1,\ldots,h_m}^{n_1,\ldots,n_m} = \Delta_{h_1}^{n_1} * \cdots * \Delta_{h_m}^{n_m}$ is the convolution of several finite-difference operators at widths h_i defined by

$$\Delta_h^{n+1} = \underbrace{\Delta_h^1 * \cdots * \Delta_h^1}_{n+1 \text{ terms}} = \sum_{k=0}^{n+1} (-1)^k \binom{n+1}{k} \frac{\delta(x + h \cdot (\frac{n+1}{2} - k))}{h^{n+1}}. \tag{3.6}$$

The spline m-kernel is a non-uniform spline of degree N_m and its support is the sum of the supports of the convolved B-splines

$$\text{supp}(\beta_{h_1,\ldots,h_m}^{n_1,\ldots,n_m}(x)) = \left[-\sum_{i=1}^m h_i \left(n_i + 1\right)/2, \sum_{i=1}^m h_i \left(n_i + 1\right)/2 \right]. \tag{3.7}$$

Proof: Using the B-spline formula (3.2) we can express the spline *m*-kernels as

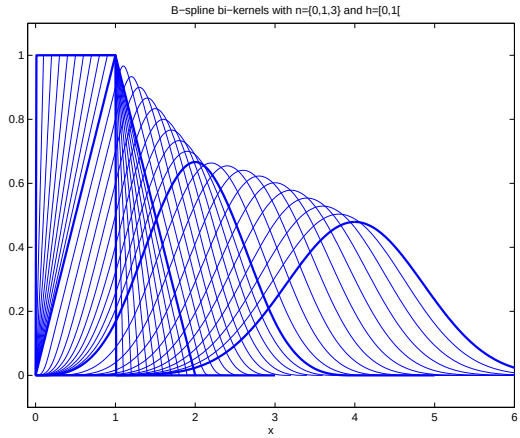


Figure 3.4: Spline bi-kernels by convolution of two causal B-splines $\beta_1^n * \beta_h^n(x)$ with $n = \{0, 1, 3\}$ and $h \in [0, 1]$. The kernels generate a smooth transition between B-splines of degree n and degree $2n + 1$. For $h = 0$ and $h = 1$, the kernels are B-splines, as $\lim_{h\to 0} \beta_1^n * \beta_h^n(x) = \beta_1^n(x)$ and $\beta_1^n * \beta_1^n(x) = \beta_1^{2n+1}(x)$.

$$\beta^{n_1,\dots,n_m}_{h_1,\dots,h_m}(x) = \left(\Delta^{n_1+1}_{h_1} * \frac{x_+^{n_1}}{n_1!}\right) * \cdots * \left(\Delta^{n_m+1}_{h_m} * \frac{x_+^{n_m}}{n_m!}\right).$$

Each term is interpreted as a convolution operator. Using the fact that the convolution operation is commutative, we get:

$$\beta^{n_1,\dots,n_m}_{h_1,\dots,h_m}(x) = \left(\Delta^{n_1+1}_{h_1} * \cdots * \Delta^{n_m+1}_{h_m}\right) * \left(\frac{x_+^{n_1}}{n_1!} * \cdots * \frac{x_+^{n_m}}{n_m!}\right).$$

Thanks to (2.7), this can be rewritten as

$$\beta^{n_1,\dots,n_m}_{h_1,\dots,h_m}(x) = \left(\Delta^{n_1+1}_{h_1} * \cdots * \Delta^{n_m+1}_{h_m}\right) * \frac{x_+^{N_m}}{N_m!}, \tag{3.8}$$

which is equivalent to (3.5). □

In the case where the h's are equal, the compound finite difference operator simplifies to

$$\Delta^{n_1+1,\dots,n_m+1}_{h,\dots,h} = \Delta^{N_m+1}_h. \tag{3.9}$$

We can use (3.5) and (3.9) to derive the well-known convolution property of two B-splines:

$$\beta_h^{n_1} * \beta_h^{n_2}(x) = \Delta_h^{n_1+n_2+2} * \frac{x_+^{n_1+n_2+1}}{(n_1+n_2+1)!} = \beta_h^{n_1+n_2+1}(x).$$

The spline m-kernel formula (3.5) expanded by (3.6) leads to a closed-form expression. For instance, with $m = 2$, we get the following explicit formula for the spline bi-kernel:

$$\beta_{h_1,h_2}^{n_1,n_2}(x) = \Delta_{h_1,h_2}^{n_1+1,n_2+1} * \frac{x_+^{n_1+n_2+1}}{(n_1+n_2+1)!}$$

$$= \sum_{k_1=0}^{n_1+1} \sum_{k_2=0}^{n_2+1} (-1)^{k_1+k_2} \binom{n_1+1}{k_1} \binom{n_2+1}{k_2}$$

$$\cdot \frac{\left(x + (\frac{n_1+1}{2} - k_1) \cdot h_1 + (\frac{n_2+1}{2} - k_2) \cdot h_2\right)_+^{n_1+n_2+1}}{(n_1+n_2+1)! \cdot h_1^{1+n_1} \cdot h_2^{1+n_2}}. \tag{3.10}$$

Examples of such bi-kernels are shown in Figure 3.4.

3.2.3 Radon transform of a B-spline

We now consider the case where the basis function φ is a separable B-spline.

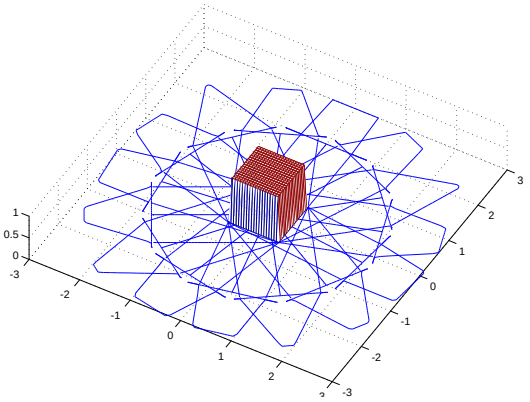
Proposition 3.2: Radon transform of the 2D B-spline
The Radon transform of the 2D separable B-spline $\beta_h^n(x,y) = \beta_h^n(x) \cdot \beta_h^n(y)$ of degree n is

$$R_\theta \beta_h^n(t) = \beta_{h|\cos\theta|}^n * \beta_{h|\sin\theta|}^n(t) = \beta_{h|\cos\theta|,h|\sin\theta|}^{n,n}(t), \tag{3.11}$$

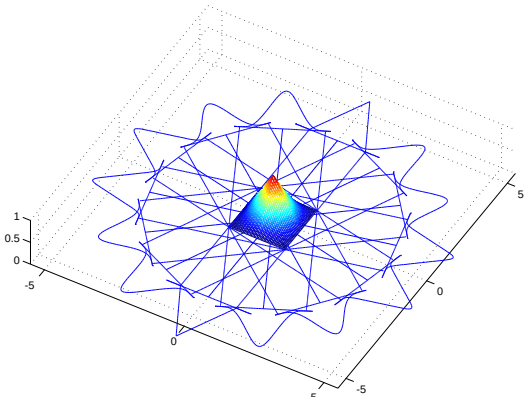
where $t = \boldsymbol{x}^\top \boldsymbol{\theta}$ and $\boldsymbol{\theta} = (\cos\theta, \sin\theta)$ with the projection angle θ; it is precisely a spline bi-kernel whose explicit form is given by (3.5) or (3.10).

The proof of Proposition 3.2 is easily derived using the Fourier slice theorem (see appendix 3.A, page 45).

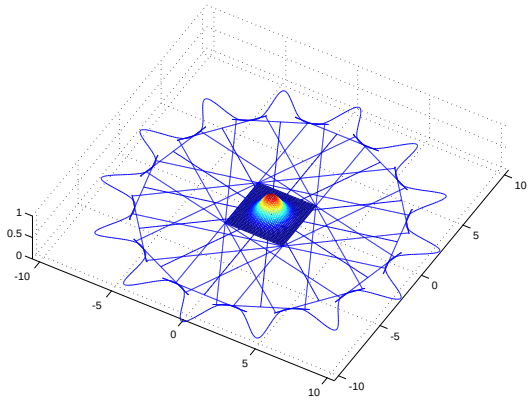
In Figure 3.5 the Radon transform of the 2D B-splines of several degrees n for various projection angles θ are shown. In the cases where $\cos\theta = 0$ or $\sin\theta = 0$, the Radon transform of the 2D B-spline reduces to a 1D B-spline of the same degree n, as $\lim_{h\to 0} \beta_h^n(x) = \delta(x)$. For angles where $|\cos\theta| = |\sin\theta|$, the Radon transform simplifies to a 1D B-spline of degree $2n+1$. For the angles in between, the spline bi-kernels range between these two extremes.



(a) $n = 0$



(b) $n = 1$



(c) $n = 3$

Figure 3.5: Spline Flowers: The Radon transform of the two-dimensional B-spline $\beta^n(x) \cdot \beta^n(y)$ is a B-spline convolution kernel. We show several examples. They are plotted on the floor, for various projection angles θ. At angles that are multiples of $\pi/4$, the B-spline convolution kernel simplifies to an ordinary B-spline.

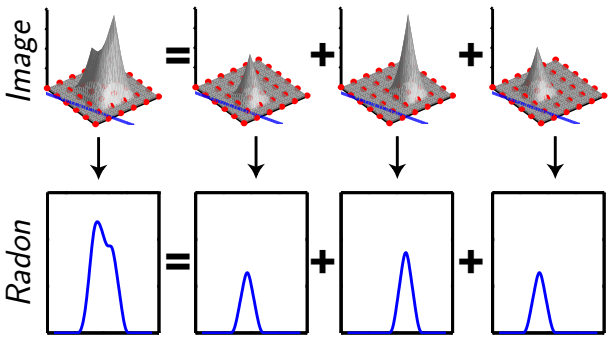


Figure 3.6: Principle of the spline-convolution Radon transform: The image is decomposed into 2D spline basis functions. Then, each basis function is X-ray projected. The X-ray projection of a 2D spline is a B-spline convolution kernel. By linearity, summing the X-ray projections of basis functions yields the X-ray projection of the whole image. The Radon transform of the image—the so-called sinogram—consists of the X-ray projections taken for a discrete set of projection angles.

3.2.4 Radon transform of the image

Let us now assume that the input image $f(\boldsymbol{x})$ is represented by a polynomial spline as a sum of shifted B-spline basis functions of degree n_1

$$f_h(\boldsymbol{x}) = \sum_{k,l \in \mathbb{Z}} c_{k,l} \beta_h^{n_1}(\boldsymbol{x} - h\boldsymbol{k}), \tag{3.12}$$

where $\boldsymbol{k} = (k, l)$.

In practice, the B-spline coefficients $c_{k,l}$ are specified such that the model (3.12) interpolates the given pixel values exactly, as described in Section 2.3.2. By using (3.11), (3.12), and the linearity of the Radon transform, we obtain the analytical expression of its projection at angle θ—the *sinogram* $g_\theta(t)$:

$$g_\theta(t) = R_\theta f_h(t) = \sum_{k,l \in \mathbb{Z}} c_{k,l} R_\theta \beta_h^{n_1}(t - h\boldsymbol{k}^\top \boldsymbol{\theta})$$

$$= \sum_{k,l \in \mathbb{Z}} c_{k,l} \beta_{h|\cos\theta|, h|\sin\theta|}^{n_1, n_1}(t - h\boldsymbol{k}^\top \boldsymbol{\theta}). \tag{3.13}$$

The principle of this technique is depicted in Figure 3.6.

3.2.5 Least-squares discretization

For the implementation purpose we will need to discretize the continuously-defined expression of the Radon transform (3.13). The simplest approach would be to sample $g_\theta(t)$ at an appropriate sampling step s.

Here, we want to be more sophisticated and approximate $g_\theta(t)$ by a polynomial spline of degree n_2 to minimize the approximation error in the L_2-sense. The solution is to compute the orthogonal projection as described in Section 2.3.3.

In order to make use of our spline convolution kernels, it is now appropriate to use a representation in terms of dual B-spline functions such that the analysis function is a B-spline. In other words, we now evaluate the best approximation of the sinogram $g_\theta(t)$ as:

$$g_{\theta,s}(t) = \sum_{i\in\mathbb{Z}} \underbrace{s\,\langle g_\theta(t), \beta_s^{n_2}(t-is)\rangle}_{\widetilde{e}_{i,\theta}} \cdot \tilde{\beta}_s^{n_2}(t-is), \tag{3.14}$$

where $g_\theta(t)$ is given by (3.13).

Here too, the approximation process is linear. Thus, to evaluate the inner product we look at what happens to one of the individual terms in (3.11):

$$P_s\{\beta^{n_1,n_1}_{h|\cos\theta|,h|\sin\theta|}\}(t-h\boldsymbol{k}^\top\boldsymbol{\theta}) = \sum_{i\in\mathbb{Z}} s\cdot d_{i,\theta,k,l}\cdot\tilde{\beta}_s^{n_2}(t-is), \tag{3.15}$$

where the coefficients $d_{i,\theta,k,l}$ are sampled values of the spline tri-kernel, or *Radon kernel*:

$$\begin{aligned} d_{i,\theta,k,l} &= \langle \beta^{n_1,n_1}_{h|\cos\theta|,h|\sin\theta|}(t-h\boldsymbol{k}^\top\boldsymbol{\theta}), \beta_s^{n_2}(t-is)\rangle_t \\ &= \beta^{n_1,n_1,n_2}_{h|\cos\theta|,h|\sin\theta|,s}(h\boldsymbol{k}^\top\boldsymbol{\theta}-is). \end{aligned} \tag{3.16}$$

The identification of this Radon kernel is the key to our approach. It allows us to reduce the method to the evaluation of the Radon kernel and yields an exact implementation. Putting everything together, we find that the least-squares coefficients $\widetilde{e}_{i,\theta}$ in (3.14) are given by

$$\widetilde{e}_{i,\theta} = \sum_{k,l\in\mathbb{Z}} d_{i,\theta,k,l}\cdot c_{k,l}\cdot s.$$

Here, s is the sampling step in the sinogram, h is the sampling step in the image, $c_{k,l}$ are the image coefficients (3.12) and $d_{i,\theta,k,l}$ the sampled values of the Radon kernels (3.16). Note that the spatial summation in (3.14) will only extend over the domain for which the tri-kernel is non-zero.

3.2.6 Analysis of least-squares approximation error

The above discretization is usually not exact and will introduce some approximation error depending on the sampling step s. The Radon transform of a 2D B-spline $\beta^{n_1}(x) \cdot \beta^{n_1}(y)$ of degree n_1 is a spline bi-kernel $\beta^{n_1,n_1}_{|\sin\theta|,|\cos\theta|}$ which is a non-uniform polynomial of degree $n_2 = 2n_1+1$. For $h = s$, the approximation space of the sinogram should have at least the same degree $n_2 = 2n_1 + 1$. Therefore, we suggest the following rule:

> *Choose a sinogram approximation space with a polynomial degree higher than (or equal to) the degree of the image approximation space. This ensure the smallest (observed) approximation error while keeping the computational cost low.*

To verify this rule, we compute the approximation error of the orthogonal projection f_s of the spline bi-kernel $f(t) = \beta^{n_1,n_1}_{|\sin\theta|,|\cos\theta|}(t)$ in the spline space with sampling step s using the general formula described in Section 2.3.7:

$$\varepsilon^2(s) = \|f - f_s\|^2_{\mathcal{L}_2} = \frac{1}{2\pi}\int_{-\infty}^{\infty} E(s\omega)|\hat{f}(\omega)|^2 d\omega. \tag{3.17}$$

Here, $\hat{f}(\omega)$ is the Fourier transform of the spline bi-kernel

$$\hat{f}(\omega) = |\sin\theta| \cdot \text{sinc}^{n_1+1}\left(\frac{\omega \cdot |\sin\theta|}{2\pi}\right) \cdot |\cos\theta| \cdot \text{sinc}^{n_1+1}\left(\frac{\omega \cdot |\cos\theta|}{2\pi}\right),$$

and $E(\omega)$, the error kernel, is given by

$$E(\omega) = 1 - \frac{|\hat{\beta}^{n_2}(\omega)|^2}{\sum_k |\hat{\beta}^{n_2}(\omega + 2k\pi)|^2}, \text{ with } \widehat{\beta}^{n_2}(\omega) = \text{sinc}^{n_2+1}(\frac{\omega}{2\pi}).$$

The approximation error is depicted in Figure 3.7; the error depends on four parameters: the degree of the image space n_1, the degree in the Radon space n_2, the sinogram sampling step s and the projection angle θ. Obviously, the approximation error can be decreased by increasing the spline degrees and/or by reducing the sampling step s, but at the expense of more computation complexity (see Section 3.4.3).

3.3 Spline-convolution filtered back-projection

We will now see that we can also use our B-spline convolution kernels to compute the inverse of the Radon transform. This yields a refined version of the filtered back-projection (FBP) algorithm for the reconstruction of tomographic images.

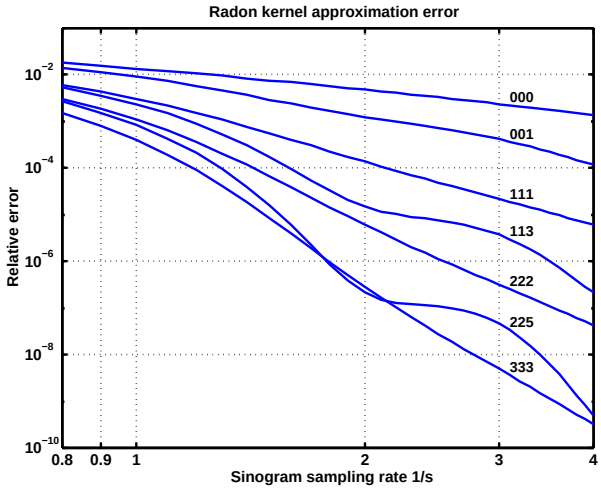


Figure 3.7: Approximation error (3.17) for various Radon kernel degrees $n_1 n_1 n_2$ and sinogram sampling rates $1/s$ averaged over the projection angles $\theta \in [0, \pi]$.

3.3.1 Filtered back-projection

The basis for the inverse Radon transform is the well-known identity (cf. [68])

$$f(\boldsymbol{x}) = (R^*(q * Rf))(\boldsymbol{x}), \tag{3.18}$$

where each projection $R_\theta f$ is convolved by a one-dimensional ramp filter q defined in the Fourier domain by $\hat{q}(\omega) = |\omega/2\pi|$. The back-projection operator R^* is the adjoint of R:

$$(R^* g_\theta)(\boldsymbol{x}) = R^* \{g_\theta(t)\} = \int_0^\pi g_\theta(\boldsymbol{x}^\top \cdot \boldsymbol{\theta}) d\theta, \tag{3.19}$$

where $g_\theta(t) = R_\theta f(t)$ is the sinogram line at angle θ. The widely-used filtered back-projection (FBP) algorithm corresponds to the direct discretization of the right-hand side of the inversion formula ($R^* q$) [76]. However, instead of the infinite ramp filter q, one usually uses an attenuated version $\hat{q}(\omega) = |\frac{\omega}{2\pi}| \hat{h}(\omega)$, where $\hat{h}(\omega)$ is a suitable spectral window (e.g. Shepp-Logan filter).

3.3.2 Spline-convolution filtered back-projection

The spline-convolution filtered back-projection implements the inverse of the spline-convolution Radon transform. Except for the additional ramp pre-filtering, the inverse method is a flow-graph transpose of the forward method. Because both the image and the sinogram are described by splines, we can again base the algorithm on the evaluation of the Radon kernel. This time, the

filtered sinogram is approximated using B-splines, while the resulting image is expanded using dual-splines. In the following, we give the necessary formulas for the three steps of our method.

Step 1. Ramp filter

In the first step, each sinogram line $\hat{g}_\theta(\omega) = \mathcal{F}_{1D}\{g_\theta(t)\}$ is filtered in the Fourier domain by $\hat{q}(\omega)$ (see Section 3.3.1)

$$\hat{p}_\theta(\omega) = \hat{g}_\theta(\omega) \cdot \hat{q}(\omega).$$

Step 2. B-spline approximation of the sinogram

In the second step, the filtered sinogram $p_\theta(t) = \mathcal{F}_{1D}^{-1}\{\hat{p}_\theta(\omega)\}$ is fitted with a spline of step size s:

$$p_{\theta,s}(t) = \sum_{i\in\mathbb{Z}} c_{\theta,i} \cdot \beta_s^n(t - is). \tag{3.20}$$

This can be done either using interpolation or projection (c.f. Chapter 2).

Step 3. Back-projection into the dual-spline space

In the third step, the back-projection $R_\theta^*\{p_{s,\theta}(t)\}$ is calculated and approximated in the image space, using dual-splines as basis functions

$$\tilde{f}_h(\vec{x}) = \sum_{k,l\in\mathbb{Z}} \tilde{c}_{k,l} \cdot \tilde{\beta}_h^n(\boldsymbol{x} - h\boldsymbol{k}), \tag{3.21}$$

where

$$\tilde{c}_{k,l} = h^2 \cdot \sum_{k,l\in\mathbb{Z}} \langle R_\theta^* p_{\theta,s}(\vec{x}), \beta_h^n(\boldsymbol{x} - h\boldsymbol{k})\rangle.$$

The crucial point is then to determine what happens if we backproject a B-spline basis function.

Proposition 3.3: For any given angle θ, the following adjoint relationship holds:

$$\langle f, R_\theta^* g\rangle = \langle R_\theta f, g\rangle,$$

where f stands for the image and g for the sinogram. □

The proof of Proposition 3.3 is given in Appendix 3.B (page 45).

Using Proposition 3.3, the coefficients $\tilde{c}_{k,l}$ are written as

$$\tilde{c}_{k,l} = \sum_{i\in\mathbb{Z},\theta} c_{\theta,i} \cdot h^2 \cdot \underbrace{\langle \beta_s^n(t - is), R_\theta \beta_h^n(t - h\boldsymbol{k}^\top\boldsymbol{\theta})\rangle_t}_{d_{i,\theta,k,l}}, \tag{3.22}$$

where $d_{i,\theta,k,l} = \beta^{n,n,n}_{h|\cos\theta|,h|\sin\theta|,s}(h\boldsymbol{k}^\top\boldsymbol{\theta} - is)$ is the same Radon kernel as in the spline-convolution Radon transform (3.16). Again, the spline spaces were chosen such that spline tri-kernels can be used. In practice, the sinogram is specified by its sample values. Thus, we can combine Steps 1 and 2 into a single filtering operation in the Fourier Domain using a slightly modified ramp filter.

In [46, 104] we derived modified ramp filters in the FFT domain for different methods to approximate the sinogram by spline (see Step 2). Using fractional splines (Section 2.3.5) instead of B-spline of integer degree or applying oblique projection (Section 2.2.2) improves the reconstruction quality. Applying the latter would allow to reduce the degree n_2 of the Radon kernel and thus shorten its support. We leave these interesting extensions open for future work.

3.4 Implementation

3.4.1 Spline convolution-based algorithms

Our algorithms are described in the Appendices 3.C and 3.D: The spline-convolution Radon transform in Algorithm 3.1 (page 46), and the spline-convolution filtered back-projection in Algorithm 3.2 (page 47).

The important formulas are summarized in Table 3.1. The key is the sampled *Radon kernel*, which is a convolution of three B-splines of various widths. In the first column of the table, we give the formulas to get the continuous functions from the coefficients. The coefficients are listed in the second column. In the third column, we specify the type of basis functions used for the representation. Whenever the Radon kernel is used, we start with B-splines and compute an approximation in terms of dual-splines. At the very end, the results are provided in the cardinal basis (pixel values) which involves an additional post-filtering step (re-sampling of the spline model).

3.4.2 Efficient computation of B-spline convolution kernels

The evaluation of spline *m*-kernels in Equation (3.5) requires the computation of polynomials. It gets more expensive when the degree increases or when the support (3.7) of the spline *m*-kernel increases. If the same spline *m*-kernel needs to be evaluated many times, it is efficient to precalculate the spline *m*-kernel.

In the case of the Radon transform, we precompute the Radon kernel—a spline tri-kernel—for various projection angles θ and store them in a lookup table (LUT). The Radon kernels are precomputed for N_θ angles between 0 and $\pi/4$ and for N_x values of x between 0 and half the support of the Radon kernel. The values for other angles or negative x can be deduced from the following symmetry properties of the Radon kernels:

	Sampled Radon kernel: $d_{i,\theta,k,l} = \langle \varphi_s(t-is), R_\theta \beta_h^n(t - k\cdot h\cos\theta - l\cdot h\sin\theta)\rangle_t$	Representation
1.	$f_h(x,y) = \sum_{k,l} c_{k,l}\cdot \beta_h^n(x-hk, y-hl)$,	B-spline
⇓	where $c_{k,l} = h^2 \left\langle \widetilde{\beta}_h^n(x-hk, y-hl), f(x,y)\right\rangle$.	
2.	$g_{\theta,s}(t) = \sum_i \widetilde{e}_{i,\theta}\cdot \widetilde{\varphi_s}(t-is)$,	Dual-spline
	where $\widetilde{e}_{i,\theta} = s\cdot \sum_{k,l} c_{k,l}\cdot d_{i,\theta,k,l}$.	
3.	$p_\theta(t) = (g_\theta * q)(t)$,	Continuous
⇓	where $\widehat{q}(\omega) = \lvert\omega/2\pi\rvert$.	
4.	$p_{\theta,s}(t) = \sum_i e_{i,\theta}\cdot \varphi_s(t-is)$,	B-spline
⇓	where $e_{i,\theta} = s\cdot \langle \widetilde{\varphi}_s(t-is), p_\theta(t)\rangle$.	
5.	$\widetilde{f_h}(x,y) = \sum_\theta \sum_{k,l} \widetilde{c}_{k,l}\cdot \widetilde{\beta}_h^n(x-hk, y-hl)$,	Dual-spline
	where $\widetilde{c}_{k,l} = h^2\cdot \sum_{i,\theta} e_{i,\theta}\cdot d_{i,\theta,k,l}$.	

Table 3.1: The *spline-convolution Radon transform* is specified in Steps 1 and 2. In Step 1, the original image is approximated in the B-spline space. In Step 2, the image is Radon transformed. The resulting sinogram is in the dual-spline space.

The *spline-convolution filtered back-projection* is specified in Steps 3 to 5. A practical application starts in Step 3 with a set of discrete measurement from the CT scanner $g_\theta(t)$. In Step 3, the sinogram is ramp-filtered in the Fourier domain, then, in Step 4, the filtered sinogram is approximated in the B-spline space. In the final step 5, the back-projection is performed and summed over all angles θ. The reconstructed image is in the dual-spline space.

$$\beta^{n_1,n_1,n_3}_{|\sin\theta|,|\cos\theta|,s}(x) = \begin{cases} \beta^{n_1,n_1,n_3}_{|\cos(\frac{\pi}{2}-\theta)|,|\sin(\frac{\pi}{2}-\theta)|,s}(x) & \text{for } \frac{\pi}{4} < \theta < \frac{\pi}{2} \\ \beta^{n_1,n_1,n_3}_{|\sin(\theta-\frac{\pi}{2})|,|\cos(\theta-\frac{\pi}{2})|,s}(x) & \text{for } \frac{\pi}{2} < \theta < \pi \end{cases}$$

$$\beta^{n_1,n_2,n_3}_{h_1,h_2,h_3}(x) = \beta^{n_1,n_2,n_3}_{h_1,h_2,h_3}(-x) \quad \text{for } x < 0.$$

The lookup table method can introduce errors if the table size $N_x \times N_\theta$ is too small. The spline-convolution algorithm was tested with Radon-kernel lookup tables with two different table resolutions: At a LUT resolution of 100×100, the PSNR of the reconstructed images decreases by approximately 1%. At a LUT resolution of 1000×1000, the error is insignificant ($< 0.001\,\%$). This has important practical implications as we found that the pre-calculated LUTs allow to accelerate the tri-kernel-based Radon transforms by one to two orders of magnitudes.

3.4.3 Computational complexity

The computational complexity of the spline-convolution algorithm is proportional to the number of evaluated image coefficients and to the support of the Radon kernel.

Support of the Radon-kernel

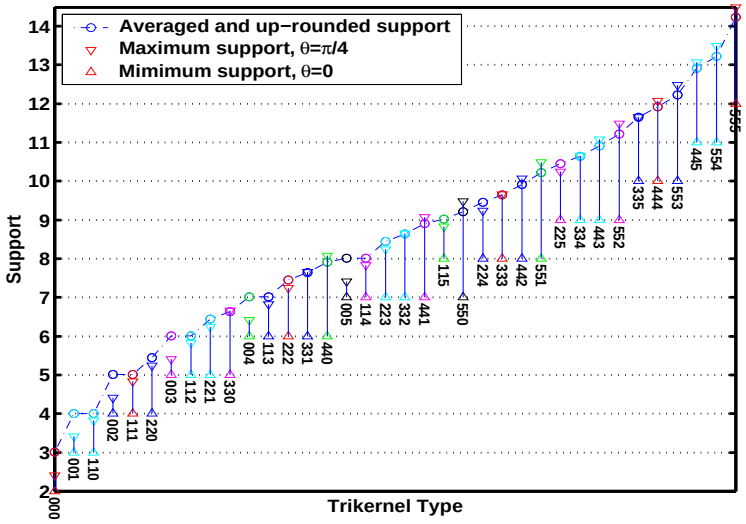


Figure 3.8: Trikernel support: The computation time is proportional to the number of sinogram coefficients that contribute to one image coefficient. This number corresponds the upper bound of the trikernel's support.

The support of the Radon kernel varies with the projection angle θ as

$$\operatorname{supp}(\beta^{n_1,n_1,n_2}_{h|\cos\theta|,h|\sin\theta|,s}) = h(|\cos\theta| + |\sin\theta|) \cdot (n_1 + 1) + s \cdot (n_2 + 1).$$

In Figure 3.8 the minimum (for $\theta = 0$) and maximum (for $\theta = \frac{\pi}{4}$) support for each degree type $n_1 n_1 n_2$ are depicted by triangles. The support is rounded up to the next integer value to count the number of sinogram knots evaluated by the Radon kernel. The average of the up-rounded supports over all projection angles is proportional to the computational complexity as

$$\overline{\operatorname{supp}}(\beta^{n_1,n_1,n_2}_{h|\cos\theta|,h|\sin\theta|,s}) = \frac{1}{K} \sum_{i=1}^{K} \left\lceil \operatorname{supp}(\beta^{n_1,n_1,n_2}_{h|\cos\theta_i|,h|\sin\theta_i|,s}) \right\rceil .$$

The averaged support increases with the sinogram degree n_2 and—by a $\frac{4}{\pi}$ greater factor—with the image degree n_1.

Complexity of spline-convolution Radon transform and of its inverse

The computational complexity of the spline-convolution Radon transform (and of its inverse) depends on the image size (M, N), the number of projection angles K, the sinogram sampling step s, the image sampling step h and the support of the spline tri-kernel. The computation time is proportional to

$$M \cdot N \cdot K \cdot \frac{1}{s} \cdot (\ \overline{\text{supp}}(\beta^{n_1,n_1,n_2}_{h|\cos\theta|,h|\sin\theta|,s}) + c), \tag{3.23}$$

where the constant c stands for some overhead.

3.5 Experimental results

In this section, we evaluate the joint performance of the proposed spline-convolution Radon transform R and of its inverse $q * R^*$. The spline-convolution Radon transform R of the image $f(x, y)$ is calculated first, resulting in the sinogram $g(t, \theta)$. Then, the inverse Radon transform R^* of the sinogram is calculated using the spline-convolution filtered back-projection (this scheme is depicted in Figure 3.9).

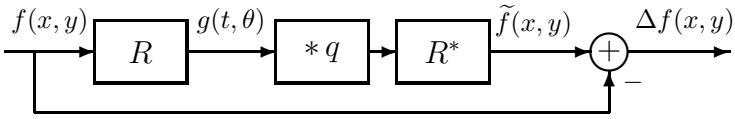


Figure 3.9: Evaluation scheme of Radon transform and its inverse

The quality of the reconstructed image is assessed using a standard measure of the peak signal-to-noise ratio (PSNR) in decibels:

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\frac{1}{N^2} \sum_{k=1}^{N} \sum_{l=1}^{N} (e(k,l))^2} \right), \tag{3.24}$$

where $e(k, l)$ denotes the error between the original image and the reconstructed image (cardinal representation) and where the maximum difference in amplitude is 255.

As test image, we use a properly sampled version of the *Shepp-Logan phantom* [88] at a resolution of $N \times N = 128 \times 128$ with pixel values between black (0) and white (255). If not

mentioned otherwise, the number of projection angles K is fixed to $K = 2N = 256$ and all sampling steps are set to one.

The spline tri-kernel degrees are chosen identical for the forward and the inverse Radon transforms. The tri-kernel degrees are specified by a 3 digit code: The first two digits stand for the two spline degrees in the x- and y-directions of the image space, the third digit for the spline degree of the sinogram space.

In the following sections, we investigate the relation between the spline degrees on the image and on the sinogram, the kernel support, the sinogram spatial and angular sampling, as well as the influences of these parameters on the image quality, on the computational complexity and on the runtime of the algorithms. The goal is to find out the best set of parameters for the proposed algorithms.

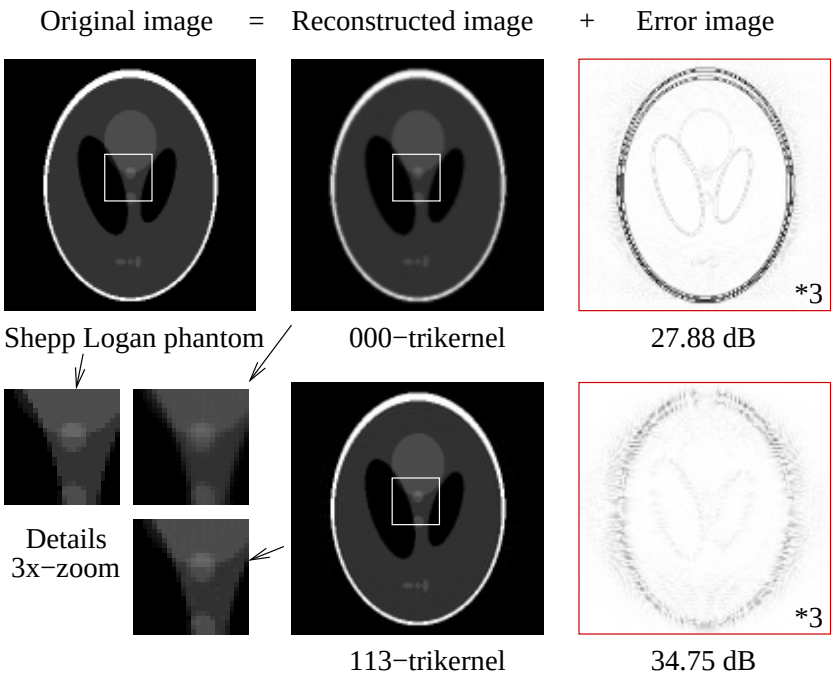


Figure 3.10: Spline-convolution filtered back-projection. The displayed error images are amplified by a factor of 3.

3.5.1 Degree on sinogram versus degree on image

Is it better to raise the degree of the spline model in the image or in the sinogram? This experiment evaluates 36 different spline tri-kernels with all possible combinations of spline degrees on

the image and on the sinogram between degrees 0 and 5. The experimental results are listed in Table 3.2; some of the error images are depicted in Figure 3.10. The PSNR is plotted versus the computation time in Figure 3.11. The envelope—a convex hull—represents the best compromise between computation time and image quality. We can conclude that it is better to use a degree on the sinogram that is equal or superior to the degree on the image.

Least-squares method

PSNR	degree n_2 on sinogram						
	$n_1 \backslash n_2$	0	1	2	3	4	5
	0	**27.88**	**30.38**	30.83	31.00	31.09	31.14
degree	1	30.45	**33.65**	**34.39**	**34.75**	34.96	35.09
n_1	2	30.93	34.19	34.82	**35.13**	**35.32**	**35.44**
on	3	31.03	34.28	34.90	35.22	35.41	**35.53**
image	4	31.11	34.39	34.99	35.29	35.47	**35.58**
	5	31.14	34.45	35.03	35.32	35.50	**35.62**

Table 3.2: Least-squares method: Reconstruction error of the Shepp-Logan phantom of size 128×128, recovered from 256 projections, using the spline-convolution algorithm. The index $n_2 = 0, 1, 2, 3, 4, 5$ on top indicates the degree of the 1D spline-space in the Radon transform domain and the indexes $n_1 = 0$, 1, 2, 3, 4, 5 on the left indicate the degree of the 2D spline space in the image domain. The tri-kernel used (both for projection and reconstruction) is given by $\beta^{n_1,n_1,n_2}_{|\sin\theta|,|\cos\theta|,1}(t)$. The bold numbers indicate solutions that lie on the quality-speed optimal curve in Figure 3.11.

3.5.2 Comparison with standard technique

The main differences between our implementation and the standard technique are two fold: First, we use better interpolation models (higher order splines). Second, we use a least squares discretization technique instead of straightforward resampling.

To answer the question of whether this is really worth the effort, we present results for the standard implementation (interpolation only), but with the same models as in our previous experiment to facilitate the comparison (cf. Table 3.3). To implement the Radon transform according to the standard method, we replace the integral by a Riemann sum with equally spaced points. The values of the points are calculated according to the interpolation model (3.12). Alternatively, we could also have resampled (3.13) which gives the exact line integral. Likewise, we resampled the spline-interpolated sinogram (3.20) to compute the back-projection.

Here too, the performance improves with the order of the model with a tendency to saturation for $n \geq 3$. For example, one gets a whole 5 dB improvement if one uses cubic splines instead of the bilinear interpolation used in most implementations (c.f. Table 3.3).

Now, if we compare Table 3.2 and 3.3, we observe that the least squares sampling provides

Standard spline-based method

PSNR	interpolation degree n_2 on sinogram for FBP						
degree	$n_1 \backslash n_2$	0	1	2	3	4	5
n_1	0	**27.49**	**28.25**	**29.48**	29.52	29.52	29.52
on	1	28.27	28.41	**30.11**	30.29	30.55	30.64
image	2	29.46	30.39	**33.41**	**33.68**	**34.00**	34.11
for	3	29.46	30.59	33.71	34.01	**34.35**	34.47
Radon	4	29.50	30.82	34.02	34.35	34.67	**34.78**
trans.	5	29.51	30.89	34.13	34.49	34.80	**34.91**

Table 3.3: Standard spline-based method: Reconstruction error of the Shepp-Logan phantom of size 128×128, recovered from 256 projections using standard algorithm. The Radon transform interpolates the image (with B-spline of degree n_1) and samples the projection on the sinogram. The back-projection interpolates the sinogram (with B-spline of degree n_2) and samples the back-projection on the image. The bold numbers indicate the solutions with the best trade-off between quality and speed.

an additional boost in performance, especially for lower order models (+4 dB for $n = 1$). Thus, it really makes sense to use the more sophisticated methods if the goal is to get the best images already at lower degrees.

3.5.3 Predictions versus measurements: runtime and quality

The predicted computation time (3.23) approximates well the measured computation time from Figure 3.11 with the parameters $h = 1$, $s = 1$, $M = N = 128$, $K = 256$, $s = 1$, $c \approx 1.142$; the correlation coefficient is 0.9995. From the analytical expression (3.17) of the approximation error for a B-spline we can estimate the expected image quality by $\widehat{\text{PSNR}} = 10 \log(255/\varepsilon^2(s))$. Figure 3.12 depicts the estimate PSNR versus the estimated computation time. A similar envelope as in Figure 3.11 is reproduced.

3.5.4 Smaller sampling step on the sinogram lines

Is it better to raise the degrees of the splines or to refine the sinogram sampling step? The answer is provided by Figure 3.13. It shows the reconstruction error (PSNR) versus the computation time. The Shepp-Logan head phantom was Radon transformed and backprojected using the spline tri-kernel-based FBP. The sinogram sampling steps are $s = 1, \frac{1}{2}, \frac{1}{4}$ and the spline tri-kernel degrees are taken from the envelope of Figure 3.11 (best cost/performance compromise). When the sampling step s on the sinogram lines gets finer, the image quality increases, but the computation time rises as well. One can see that a 111 tri-kernel at full sampling step $s = 1$ is as good and fast as the 000 tri-kernel at the finer sampling step $s = \frac{1}{2}$. But a finer sampling

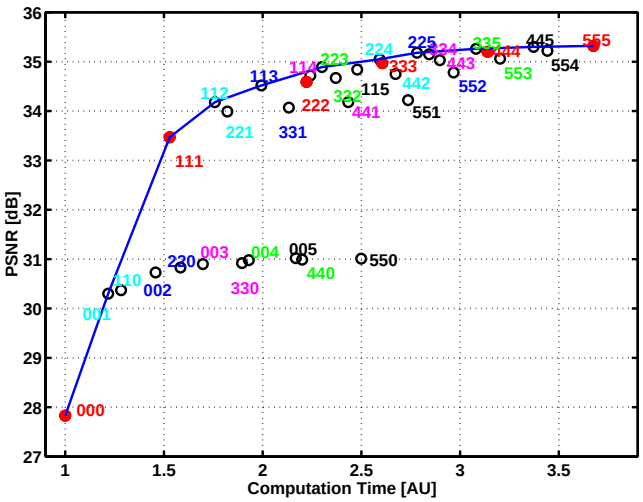


Figure 3.11: PSNR versus computation time in arbitrary units. By convention, the time is set to 1 for the 000-kernel. The first two numbers represent the degree on the image and the last the degree on the sinogram. For the 000-kernel, the absolute runtime is 5.5 seconds for the Radon transform and 6 seconds for the back-projection (on G3, $N = 128$, $K = 256$, $s = 1$). The envelope is the best compromise between speed and quality.

step requires that more sinogram data is acquired. For a fixed amount of sinogram data, e.g., measurements from a CT scanner, it is clear that a higher kernel degree yields a better quality of the reconstructed image. For finer sampling steps the kernels of higher degree quickly reach an upper-quality limit. Already for a quadric kernel degree, there is hardly a difference between two-times and four-times sinogram oversampling. Therefore, we can recommend to use either high-degree kernels at the sampling step $s = 1$, or the 111 tri-kernel at a sampling step of $s = \frac{1}{2}$. Four-times sinogram oversampling ($s = \frac{1}{4}$) with degrees higher than 0 result in insignificant improvements only.

3.5.5 Angular sampling step versus sinogram sampling step

Is is better to use a finer angular or a finer sinogram sampling step? In the FBP literature [27], the customary rule is to use four-times over-sampling on the sinogram and twice as many angles as the size of the image along one dimension. Our experiments verify this rule and suggest similar rules for higher approximation orders. In a large experiment (Table 3.4), we have evaluated the image quality and computation time for numerous projection angles, $K \in \{128, 192, 256, 384, 512\}$, and sinogram sampling step, $s \in \{1, \frac{1}{2}, \frac{1}{4}\}$, for tri-kernels of degrees on the envelope.

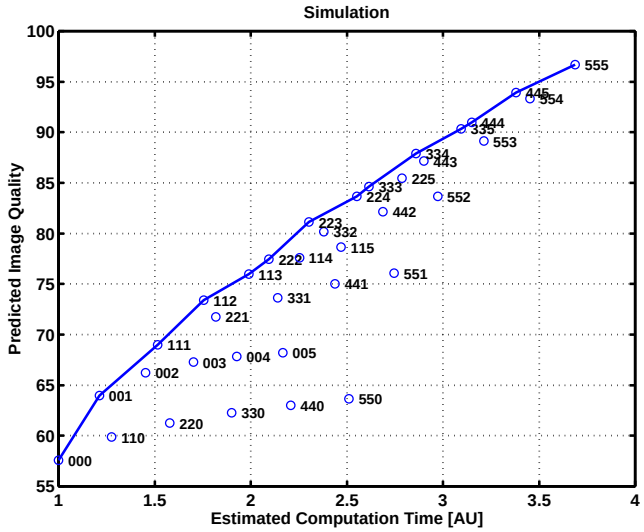


Figure 3.12: Predicted image quality over predicted computation time. We get a similar envelope as for the measured values (compare with Figure 3.11). These estimates are to be considered as upper bounds; there are additional source of error due to angular sampling which are not taken in account here.

For our previous experiments we chose $s = 1$ and $K = 2N$. Here, we find that it is faster and slightly better to use double sampling with $s = \frac{1}{2}$ and only half as much angles ($K = N$). This advantage is less significant for higher degrees. But higher degrees result in a much better image quality than lower ones.

The customary rule can be found in the case closest to the classical one (interpolation degree 0). There, an oversampling with $s = \frac{1}{4}$ still gains better quality for $K = N$ and $K = 2N$. It is faster to use higher-degree kernels to yield the best quality. In additional experiments, we have verified that the presented results reproduce as well for bigger image sizes when the ratio between the angle resolution and the image resolution is the same. Slight image quality improvements can be achieved with kernels of higher degrees $n = 2, 3$. We conclude that the best compromise between speed and quality is the following:

> *Use the spline-convolution Radon (and inverse Radon) transform with a tri-kernel degrees* $n_1 n_1 n_2 = 111$ *at double sinogram sampling step* $s = \frac{1}{2}$ *and twice as much angles* $K = 2N$ *as the image size* N.

In practice, the resolution of the measured CT data is limited by the maximum sinogram sampling rate and the maximum angular resolution of the CT scanner. There is a quality trade-off between the number of given angular projections and the size of the reconstructed image. A higher kernel degree allows for a similar quality at a bigger image size. But if the reconstructed image is too

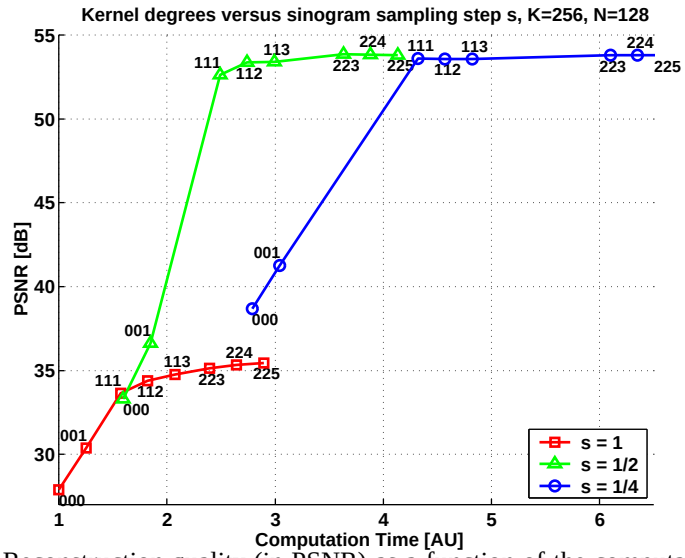


Figure 3.13: Reconstruction quality (in PSNR) as a function of the computation time and of the sampling step s on the sinogram lines. The time scale is normalized to 1 for the routine with the lowest tri-kernel degree (000). The first two numbers represent the degrees on the image and the last the degree on the sinogram.

big, then artifacts due to angular under-sampling occur in the border area of the image.

3.6 Conclusion

We derived an explicit formula for the convolution of multiple B-splines, which we called B-spline convolution kernels. We imposed a continuous B-spline model on the image and its Radon transform. We showed that the Radon transform of the basis function (a tensor-product B-spline) is a spline bi-kernel. The approximation of the spline bi-kernel in the dual-spline space corresponds to the sampled convolution of three B-splines, the so-called Radon kernel. With it, the Radon transform and its inverse are consistently and efficiently discretized using B-spline signal processing.

The approximation power of our method increases with the degree of the splines. The best compromise between computational complexity and approximation accuracy is achieved with Radon kernels with a sinogram degree n_2 that is higher than the image degree n_1. A degree $n_2 = 2n_1 + 1$ is especially efficient computationally. For higher degrees, the upper image-quality bound is already reached at two-times sinogram oversampling. This makes the commonly-recommended four-times oversampling superfluous and allows for better resolution of the reconstructed image for a fixed sinogram resolution.

Image quality in PSNR [dB] (computation time in [au]), N=128						
tri-	sampl.	Number for projection angles K				
kernel	step	K=128	K=192	K=256	K=384	K=512
000	s=1	27.77 (0.5)	*27.87 (0.8)*	*27.88 (1.0)*	*27.88 (1.5)*	*27.88 (2.0)*
000	s=1/2	32.64 (0.8)	33.14 (1.2)	33.29 (1.6)	*33.35 (2.4)*	*33.36 (3.2)*
000	s=1/4	35.54 (1.4)	37.78 (2.1)	38.68 (2.8)	39.08 (4.2)	*39.12 (5.6)*
001	s=1	30.23 (0.6)	*30.37 (0.9)*	*30.38 (1.3)*	*30.38 (1.9)*	*30.38 (2.5)*
001	s=1/2	34.58 (0.9)	36.10 (1.4)	36.63 (1.8)	*36.81 (2.8)*	*36.81 (3.7)*
001	s=1/4	35.91 (1.5)	39.34 (2.3)	41.25 (3.0)	42.39 (4.6)	*42.56 (6.1)*
111	s=1	32.97 (0.8)	33.55 (1.2)	*33.65 (1.6)*	*33.65 (2.4)*	*33.65 (3.2)*
111	s=1/2	37.83 (1.2)	44.30 (1.9)	52.63 (2.5)	*57.73 (3.7)*	*57.74 (5.0)*
111	s=1/4	37.67 (2.2)	44.16 (3.2)	53.59 (4.3)	73.06 (6.5)	*75.48 (8.7)*
113	s=1	33.88 (1.0)	34.66 (1.6)	*34.75 (2.1)*	*34.75 (3.1)*	*34.75 (4.2)*
113	s=1/2	37.67 (1.5)	44.16 (2.2)	53.41 (3.0)	*65.15 (4.5)*	*65.16 (6.0)*
113	s=1/4	37.66 (2.4)	44.15 (3.6)	53.57 (4.8)	74.14 (7.2)	*78.64 (9.7)*

Table 3.4: Sinogram resolutions: Comparison of angular resolution K versus sinogram sampling step s to evaluate the image quality and the computation time. The Shepp-Logan phantom's image size is $N = 128$. The time scale is normalized to 1 for the 000-kernel and $K = 256$. Increasing the angular sampling rate improves the image quality further only for high kernel degrees. The cursive PSNR values indicate saturated quality for the given tri-kernel and sampling step s. PSNR values above 60 dB can be considered as good as prefect (+6dB $\equiv$ 1 more bit precision).

We suggest to use the spline-convolution FBP (and Radon transform) with Radon kernels of degrees $n_1 n_1 n_2$ of 001, 111, 113 or 225, together with two-times sinogram oversampling and an angular resolution $K = 2N$ of twice the image size N along one axis.

3.7 Major achievements and future work

The main contribution in this chapter are the derivation of an explicit formulas for spline convolution kernels (Section 3.2.2) and its application to discretize the Radon transform.

In the context of spline-based FBP we had also two novel minor findings: First, that the fractional spline allows to calculate analytically the ramp filtering step of the FBP [104], and second, that oblique projection can improve the reconstruction as well [46]. The combination of these techniques with the Radon kernel is left open to future work.

Chapter appendix

3.A Proof of Proposition 3.2

Let $g_\theta(t) = R_\theta \beta^n(t)$ be the Radon transform of the tensor product of two B-splines. The Fourier slice theorem states the equivalence between the one-dimensional Fourier transform of a parallel projection and a 1D cut in the two-dimensional Fourier transform of an image:

$$\mathcal{F}_{1D}\left\{R_\theta f(\cdot)\right\}(\omega) = \mathcal{F}_{2D}\left\{f(\cdot,\cdot)\right\}(\omega\cos(\theta), \omega\sin(\theta)) \tag{3.25}$$

By performing a 1D inverse Fourier transform on both sides of (3.25), and from the scaling properties and convolution properties of the Fourier Transform, we deduce:

$$\begin{aligned} R_\theta \beta^n(t) &= \tfrac{1}{2\pi}\int_{-\infty}^{\infty} \hat{\beta}^n(\omega\cos\theta)\cdot\hat{\beta}^n(\omega\sin\theta)e^{i\omega t}d\omega \\ &= \tfrac{1}{|\cos\theta|\cdot|\sin\theta|}\beta^n\left(\tfrac{\cdot}{|\cos\theta|}\right) * \beta^n\left(\tfrac{\cdot}{|\sin\theta|}\right)(t) \\ &= \beta^n_{|\cos\theta|} * \beta^n_{|\sin\theta|}(t) \end{aligned}$$

Therefore, the Radon transform of a 2D B-spline can be written as a 1D convolution of two scaled B-splines. A further scaling of this results by h yields the result in Proposition 3.2.

3.B Proof of Proposition 3.3

We assume in this proof that the direction angle $\boldsymbol{\theta} = (0,1)$ of the Radon projection R_θ and of its inverse R_θ^* is aligned with the coordinate system (x,y). Otherwise, the coordinate system is rotated by an angle $-\theta$. Let f be the image and g the sinogram.

Then, $(R_\theta f)(x) = \int f(x,y)dy$ and $(R_\theta^* g)(x,y) = g(x)\ \forall y$.
The 2D scalar product of $R_\theta^* g$ and f is

$$\begin{aligned} \langle R_\theta^* g, f\rangle &= \langle (R_\theta^* g)(x,y), f(x,y)\rangle \\ &= \iint f(x,y)\cdot(R_\theta^* g)(x,y)dxdy = \iint f(x,y)\cdot g(x)dxdy \\ &= \iint f(x,y)dy\cdot g(x)dx = \int (R_\theta f)(x)\cdot g(x)dx \\ &= \langle (R_\theta f)(x), g(x)\rangle = \langle R_\theta f, g\rangle\,, \text{ q.e.d.} \end{aligned}$$

We have shown that the 2D scalar product in the image domain between a back-projected sinogram line and an image is equivalent to the 1D scalar product of the projected image and the sinogram line.

3.C Algorithm of spline-convolution Radon transform

Algorithm 3.1: Spline-convolution Radon transform

▷ **Input**: ImageCardinal of size (N_x, N_y), number of projection angles K

◁ **Output**: RadonCardinal of size $(\sqrt{N_x^2 + N_y^2}, K)$

1 ImageBspline=ChangeBasis2dCardinalToBasicSpline(ImageCardinal, n_1)

2 **FOR** $j := 0$ **TO** $K - 1$ *// project for each angle* $\theta[j]$

3 $h_1 = \sin(\theta[j]) \cdot h$

4 $h_2 = \cos(\theta[j]) \cdot h$

5 Support=TriKernelSupport$(n_1, n_1, n_2, h_1, h_2, s)$

6 **FOR** $k := 0$ **TO** $N_x - 1$

7 **FOR** $l := 0$ **TO** $N_y - 1$ *// sum over each pixel*

8 $t = x[k] \cdot h_1 + y[l] \cdot h_2$

9 $[i_{min}, i_{max}]$ =GetConcernedIndices$(t$, Support$)$

10 **FOR** $i := i_{min}$ **TO** i_{max} *// sum over the support*

11 RadonDual$[i, j]+ =$TriKernel$(n_1, n_1, n_2, |h_1|, |h_2|, s, t - i) \cdot$ $\cdot$ ImageBspline$[k, l]$

12 **END**

13 **END**

14 **END**

15 RadonCardinal$[:, j]$=ChangeBasis1dDualToCard(RadonDual$[:, j], n_2$)

16 **END**

Algorithm 3.1 describes the spline-convolution Radon transform: The image representation is changed from cardinal to the B-spline space in line 1. The algorithm loops over a discrete set of K projections angles (FOR in line 2) and over all image coefficients $c_{k,l}$ (FORs in lines 6-7). The support of the angle-dependent tri-kernel is computed in lines 3-5. A point $\boldsymbol{k} = (k, l)$ in the image projects to the position $t = h\boldsymbol{k}^\top \boldsymbol{\theta}$ on the sinogram (lines 8-9). The relative shift $t - i$ determines the points at which the tri-kernel is evaluated (lines 10-12). At the very end (line 15), the lines of the sinogram are changed from the dual-spline representation to the cardinal spline one to get the sample values.

3.D Algorithm of spline-convolution back-projection

Algorithm 3.2: Spline-convolution Back-Projection

▷ **Input**: RadonCardinal of size $(\sqrt{N_x^2 + N_y^2}, K)$, number of projection angles K, image size N_x, N_y

◁ **Output**: ImageCardinal of size (N_x, N_y)

1 **FOR** $j := 0$ **TO** $K - 1$ *// project for each angle* $\theta[j]$

2 RadonCardinal$[:, j]$ *// choose sinogram line and filter*

3 FilteredProjection=FilterInFourier(Projection,*'RamLak'*)

4 FilteredProjBspline=ChangeBasis1dCardinalToBspline(FilteredProjection, n_2)

5 $h_1 = \sin(\theta[j]) \cdot h$

6 $h_2 = \cos(\theta[j]) \cdot h$

7 Support=TriKernelSupport$(n_1, n_1, n_2, h_1, h_2, s)$

8 **FOR** $k := 0$ **TO** $N_x - 1$

9 **FOR** $l := 0$ **TO** $N_y - 1$ *// sum over each pixel*

10 $t = x[k] \cdot h_1 + y[l] \cdot h_2$

11 $[i_{min}, i_{max}]$ =GetConcernedIndices$(t,$ Support$)$

12 **FOR** $i := i_{min}$ **TO** i_{max} *// sum over the support*

13 ImageDual$[k, l]+ =$TriKernel$(n_1, n_1, n_2, h_1, h_2, s, t - i) \cdot$ $\cdot$ FilteredProjBspline$[i]$

14 **END**

15 **END**

16 **END**

17 RadonCardinal$[:, j]$=ChangeBasis1dDualToCard(RadonDual$[:, j], n_2$)

18 **END**

19 ImageCardinal=$\pi/K \cdot$ ChangeBasis2dDualToCardinal(ImageDual, n_1)

Algorithm 3.2 describes the spline-convolution filtered back-projection: It sums the back-projections over K projection angles (line 1). First each sinogram line (line 2) is filtered by the ramp filter q in Fourier Domain (line 3) and then approximated with B-splines (line 4). The support of the angle-dependent tri-kernel is computed in lines 5-7. The loops over all image coefficients (FORs in line 8-9) and the Radon kernels (line 12) are the same as in Algorithm 3.1, with one exception in line 13: The data is transformed in the opposite direction: from the Radon space to the image space. At the end (line 19) the reconstructed image is changed to the cardinal spline representation and divided by the number K of summed back-projections.

Chapter 4

Volume Rendering by Droplets

"We are each only one drop in a great ocean—but some of the drops sparkle."
—LAO TZU

4.1 Introduction

In this chapter, we will apply the same kind of sampling technique as for the Radon transform (cf. Chapter 3) to improve volume rendering. Volume rendering is a useful technique [65] in biomedical visualization, material testing, physics or weather forecast. It is computationally expensive and requires a substantial amount of memory and bandwidth. Fast implementations rely on hardware rendering and coarse approximations. In an interactive viewing scenario, a user would like to browse around and through a volume to explore the data and to get a good sensation of its three-dimensionality. One might accept the compromise of lower image resolution for the sake of fast feedback. Then, once a viewpoint of the volume has been chosen, the resolution increases again (cf. [26, 62, 67]).

We are interested in interactive volume rendering with multiresolution, similarly to Gross [26, 67] but with two important differences. First, instead of interpolation-based wavelet splatting, we use wavelet projections that are optimal in the least-square sense, hence better quality. Second, the computation of the projection is done on a grid whose sampling step size h adapts to the size of the basis function. This gives an additional speed-up at coarse resolutions.

In Section 4.2, we explain the two differences in detail. In Section 4.3, we compare the computational complexity of the usual wavelet splatting method with the proposed method.

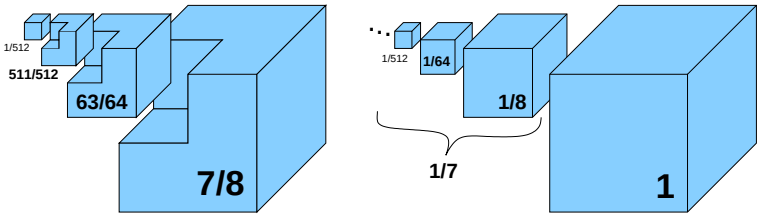


Figure 4.1: Multiresolution volumes: The wavelet decomposition (a) is non-redundant. The pyramid representation (b) adds an overhead of $1/7 \approx 14\%$.

4.2 Rendering in wavelet space

In the following, we project a volume of size M^3 on a screen of size N^2. This operation is called the X-ray transform—it is the most basic volume rendering method. The volume data is represented in a L_2-optimal multiresolution pyramid [7, 106] (see Figure 4.1).

4.2.1 The volume rendering integral

In its simplest form, the volume rendering integral is equal to the parallel projection p_θ in the direction $\boldsymbol{\theta}$ defined by

$$g_\theta(u,v) = p_\theta f(\boldsymbol{y}) = \int f(t \cdot \boldsymbol{\theta} + u \cdot \boldsymbol{e}_u + v \cdot \boldsymbol{e}_v) dt, \tag{4.1}$$

where $f(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^3$ is the volume, and where $g_\theta(\boldsymbol{y}),\ \boldsymbol{y} = (u,v) \in \mathbb{R}^2$ is the projection of the volume on a plane perpendicular to $\boldsymbol{\theta}$ and spanned by the vectors $\boldsymbol{e}_u$ and $\boldsymbol{e}_v$. In the last equation (4.1) we utilize three coordinate systems that have identical origins: Cartesian coordinates $\boldsymbol{x} = (x,y,z)$, spherical coordinates (α,β,r) and projection plane coordinates $\boldsymbol{u} = (t,u,v)$. The projection direction $\boldsymbol{\theta}$ is aligned with the normal $\boldsymbol{e}_t$ of the projection plane. It is spanned by $\boldsymbol{e}_u$ and $\boldsymbol{e}_v$, which can be expressed in Cartesian coordinates as a function of the two Euler angles α and β (see Figure 4.2).

The coordinate change between Cartesian coordinates $\boldsymbol{x}$ and projection plan coordinates $\boldsymbol{u}$ is a rotation: $\boldsymbol{x} = R \cdot \boldsymbol{u}$, where the rotation matrix is $R = [\boldsymbol{e}_t\ \ \boldsymbol{e}_u\ \ \boldsymbol{e}_v]$, or, written explicitly,

$$R = \begin{bmatrix} -\cos\alpha \cdot \cos\beta & \cos\alpha \cdot \sin\beta & -\sin\alpha \\ -\sin\alpha \cdot \cos\beta & \sin\alpha \cdot \sin\beta & \cos\beta \\ -\sin\beta & -\cos\alpha & 0 \end{bmatrix}.$$

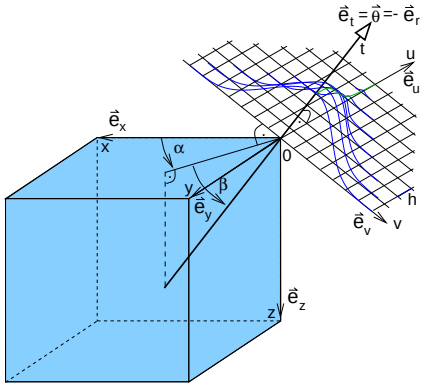


Figure 4.2: Coordinate systems used for parallel projection and Fourier slice theorem: Cartesian coordinates (x, y, z), spherical coordinates (α, β, r) and intersection plane coordinates (t, u, v).

4.2.2 Multiresolution projection

The volume data is represented in the wavelet domain by linear combination of a set of basis functions $\psi_k(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = \sum_{k \in \mathbb{R}^3} c(k)\psi_k(\boldsymbol{x}). \tag{4.2}$$

Using this decomposition, the volume rendering integral from (4.1) becomes

$$\begin{aligned} p_\theta f(\boldsymbol{y}) &= \sum_{k \in \mathbb{R}^3} c(k) \int \psi_k(t \cdot \boldsymbol{\theta} + u \cdot \boldsymbol{e}_u + v \cdot \boldsymbol{e}_v) dt \\ &= \sum_{k \in \mathbb{R}^3} c(k) \cdot \underbrace{p_\theta \psi_k(\boldsymbol{y})}_{\text{droplet } \chi_{\theta,k}(\mathsf{y})}, \end{aligned} \tag{4.3}$$

which is a linear combination of droplets $\chi_{\theta,k}(\boldsymbol{y}) = p_\theta \psi_k(\boldsymbol{y})$, where $\boldsymbol{y} = (u, v) \in \mathbb{R}^2$. Note that k may code for a combination of shifts and scaling (wavelet decomposition). The droplet is the splatted—or projected—version of the basis function [62] or wavelet [67]. We refer to our method by the term *projection* instead of *splatting*, as the latter is associated colloquially with a rather "dirty" process.

4.2.3 The B-spline droplet

We use the B-spline as an underlying separable basis function. It has numerous advantages, like compact support and explicit formulas, for details see Chapter 2 or [98]. The Fourier transform

of a B-spline of degree n is given by

$$\hat{\beta}^n(\omega) = \frac{\sin^{n+1}(\omega/2)}{\omega/2}. \tag{4.4}$$

In order to get the droplet, the three-dimensional B-spline function $\beta^n(\boldsymbol{x}) = \beta^n(x,y,z) = \beta^n(x)\beta^n(y)\beta^n(z)$ has to be projected along the direction $\boldsymbol{\theta}$.

The Fourier slice theorem states that in the Fourier domain the intersection of a volume by the plane spanned by $\boldsymbol{e}_u$ and $\boldsymbol{e}_v$ is equivalent to the 2D Fourier transform of the B-spline droplet $\chi_\theta(u,v)$:

$$\hat{g}(\omega_u,\omega_v) = \hat{\beta}^n(\omega_x)\hat{\beta}^n(\omega_y)\hat{\beta}^n(\omega_z), \tag{4.5}$$

with the substitution

$$\begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^\top = R \begin{bmatrix} \omega_u & \omega_v & \omega_t \end{bmatrix}^\top_{|\omega_t=0}. \tag{4.6}$$

Unfortunately there is no simple analytical formula of the projection of the 3D tensor product B-spline, as it was the case for the 2D tensor product B-spline in Chapter 3. Therefore, we calculate the droplet numerically using the above stated Fourier slice theorem. Interestingly, the B-spline droplet becomes almost radially symmetric for higher B-spline degrees.

Next, we use the 2D Fourier transform of the B-spline droplet to evaluate the approximation error.

4.2.4 Approximation on an adaptive grid

In Gross' et al. approach [67], the droplets are always calculated at the full resolution $h = 1$. Here, we investigate an alternative approach where the resolution of the grid h may be adapted to the size of the basis functions and where, in order to get high quality, we use least-squares approximation rather than the usual interpolation method.

The droplet $\chi_{\theta,j}(\boldsymbol{y})$ is approximated onto a reconstruction grid with the sampling step h_j using basis functions $\varphi(\frac{\mathsf{y}}{h_j} - l)$, where $h_j = 2^j h$ adapts to the scale j. We also use the fact that $\varphi(\boldsymbol{y})$ satisfies a two-scale relation to get a fast full-screen interpolation using digital filters.

With the approximation operator P_{h_j}, the approximation of the droplet is

$$P_{h_j}\chi_{\theta,j}(\boldsymbol{y}-\boldsymbol{z}) = \sum_{\boldsymbol{l}\in\mathbb{Z}^2} c_{h_j}(\boldsymbol{l})\varphi(\frac{\boldsymbol{y}}{h_j} - \boldsymbol{l}) \tag{4.7}$$

with

$$c_{h_j}(\boldsymbol{l}) = \langle \chi_{\theta,j}(\boldsymbol{y}-\boldsymbol{z}), \tilde{\varphi}(\frac{\boldsymbol{y}}{h_j} - \boldsymbol{l})\rangle \frac{1}{h_j^2}, \tag{4.8}$$

where $\tilde{\varphi}(\boldsymbol{y})$ is the dual function of $\varphi(\boldsymbol{y})$ (see Chapter 2).

h	Haar $n=0$	Linear $n=1$	Cubic $n=3$	Quintic $n=5$
1	32.6%	15.6%	11.0%	10.1%
0.75	25.2%	7.8%	3.4%	2.6%
0.5	17.2%	2.8%	0.34%	0.1%
0.25	8.7%	0.62%	0.010%	0.001%

Table 4.1: Approximation errors: Relative approximation errors of cubic B-spline droplet projected on grid with B-splines of degree n and sampling step h. We can choose an appropriate h to keep the approximation error below some threshold.

Here, $\chi_{\theta,j}(\boldsymbol{y}-\boldsymbol{z})$ denotes the continuously-defined B-spline droplet at direction $\boldsymbol{\theta}$, scale j and shift $\boldsymbol{z}$. The coefficients $c_{h_j}(\boldsymbol{l})$ are its discrete representation on the visualization grid (with sampling step h_j).

As the sum (4.3) is performed over large volumetric data, it is wise to select a function of *short support* for the dual basis function $\tilde{\varphi}(\boldsymbol{x})$ in (4.8) in order to get short and efficient filters. The cubic B-spline $\beta^3(\boldsymbol{x})$ is a good candidate. Ordinary sampling, as used in [67], is equivalent to the choice $\tilde{\varphi}(\boldsymbol{x})=\delta(\boldsymbol{x})$, but the quality is not as good as with the least-squares approximation.

The approximating error of a function f projected on a grid with the sampling step h is given in Section 2.3.7 by

$$\epsilon(h)=\|f-P_hf\|^2=\frac{1}{2\pi}\int_{-\infty}^{\infty}E(h\boldsymbol{\omega})|\hat{f}(\boldsymbol{\omega})|^2d\boldsymbol{\omega}, \tag{4.9}$$

where $\hat{f}(\omega)$ is the Fourier transform of the function f. The error kernel $E(\omega)$ is defined by

$$E(\boldsymbol{\omega})=1-\frac{|\hat{\varphi}(\boldsymbol{\omega})|^2}{\sum_n|\hat{\varphi}(\boldsymbol{\omega}+2n\pi)|^2}. \tag{4.10}$$

We evaluate the relative approximation error $\epsilon_r(h)=\|f-P_hf\|/\|f\|$ for droplets of cubic B-splines, $\hat{f}(\boldsymbol{\omega})=\hat{\beta}^3(\boldsymbol{\omega})$, using grid basis functions that are tensor-product B-splines of degree $n=0,1,3$. Cubic B-splines perform significantly better than linear or piecewise constants (see Table 4.1 and Figure 4.3).

4.2.5 Fast projection with table lookup

When the volume is rendered, equation (4.7) has to be evaluated for each wavelet coefficient and its corresponding droplet $\chi_{j,\theta}(\boldsymbol{y}-\boldsymbol{z})$. For a given projection direction $\boldsymbol{\theta}$ the shape of the droplet does not change; its size depends only on the scale j:

$$\chi_{j,\theta}(\boldsymbol{y}-\boldsymbol{z})=\chi_{j,\theta}(\frac{\boldsymbol{y}}{2^j}-\boldsymbol{z}).$$

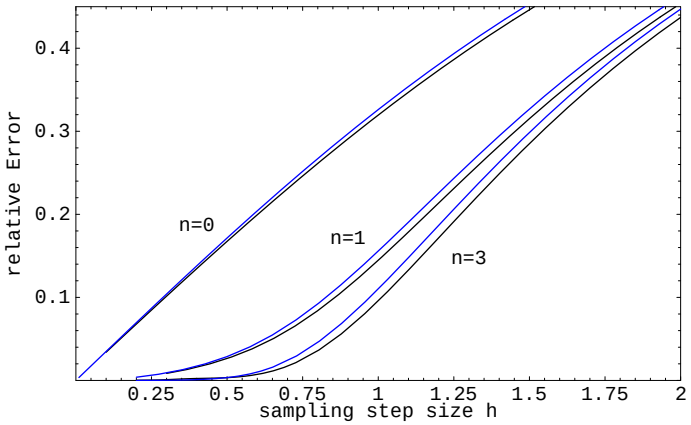


Figure 4.3: Approximation errors: Relative approximation error of cubic B-spline droplet projected on grid with B-splines of various degrees n and sampling steps h. For each degree, the plot shows the error for the shortest cut in blue and the slightly bigger error for the longest cut in black.

Furthermore, we adjust the scale of the grid to the scale of the droplet

$$\tilde{\varphi}(\frac{\boldsymbol{y}}{h_j} - \boldsymbol{l}) = \tilde{\varphi}(\frac{\boldsymbol{y}}{2^j h} - \boldsymbol{l}).$$

By substituting $\frac{\boldsymbol{y}}{2^j}$ by $\boldsymbol{y}$, we see that the scalar product (4.8), which determines the $c(\boldsymbol{l})$'s, does not depend on the scale j anymore. This means, for a fixed view, we can express (4.8) by a single kernel function $\xi_{h,\theta}(\boldsymbol{s})$, that depends only on $\boldsymbol{\theta}$ and h:

$$c_{h_j}(\boldsymbol{l}) = \frac{1}{4^j}\xi_{h,\theta}(\boldsymbol{s}), \tag{4.11}$$

and

$$\xi_{h,\theta}(\boldsymbol{s}) = \langle \frac{1}{h^2}\tilde{\varphi}(\frac{\boldsymbol{y}}{h}), \chi_\theta(\boldsymbol{y} + \boldsymbol{s})\rangle,$$

with a single shift parameter

$$\boldsymbol{s} = h\boldsymbol{l} - \boldsymbol{z},$$

where h is the sampling step on the grid and $\boldsymbol{z}$ the relative position at which the droplet is projected on the adapted grid. We calculate the kernel $\xi_{h,\theta}(\boldsymbol{s})$ once for the current viewing direction $\boldsymbol{\theta}$ at fine resolution in $\boldsymbol{s}$; the kernel is stored in a look-up table (LUT). Only a single LUT look-up is necessary to get one grid coefficients from (4.11). This method is essentially the 3D extension of the Radon algorithm in Chapter 3 using as well LUTs.

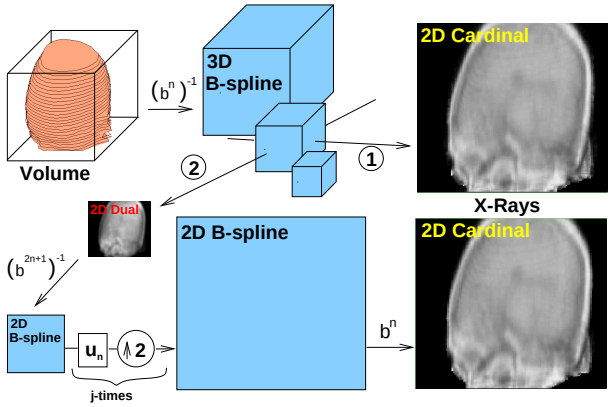


Figure 4.4: Volume rendering methods: The wavelet splatting method (1) splats at full resolution. The proposed wavelet projection method (2) approximates on a low resolution adaptive grid in dual space. It then interpolates the grid to full resolution in spline space and visualizes in cardinal space. Method (2) is faster at lower scale. As method (2) approximates on a coarser grid for the coarser scales, the quality can be slightly smaller than the quality of method (1). At full scale method (2) is slower but its quality (due to the least-squares approximation) is better than method (1).

4.3 Computational complexity

In this section, we compare our L_2-optimal wavelet projection method (droplets) with the wavelet splatting method proposed by Gross and Lippert [26, 67], see Figure 4.4.

Method 1: Wavelet splat at full resolution

Let us consider the 3D multiresolution wavelet decomposition of a volume. It is straightforward to splat each 3D coefficient to the rendering screen at full resolution as described in [67]. The wavelet splat is calculated once for the current viewing direction at full resolution. Then, for each coefficient, the splat is multiplied with the value of the coefficient and accumulated in the rendering buffer. The number of operations Op_1 depends on the size of the splat s^2 and on the number of wavelet coefficients M^3. We refer to this as Method 1. It needs Op_1 operations

$$\mathrm{Op}_1 = k_1 \cdot \underbrace{\left(\frac{M}{2^j}\right)^3}_{\text{voxels at scale } j} \cdot \underbrace{\left(2^j \frac{s+1}{h}\right)^2}_{\text{size of splat}} \sim 2^{-j} \cdot M^3, \tag{4.12}$$

where the support of the wavelet splat is $s(\Psi^n) = 2n - 1$. One operation corresponds to one addition and one multiplication.

Method 2: Droplet by least-squares optimal projection

The volume is expanded in multiple resolutions in the spline space. Here, the droplets are approximated in dual space on a grid with a sampling step *adapted* to the size of the basis function. The first part of the wavelet projection method needs $\mathrm{Op}_{2.1}$ operations

$$\mathrm{Op}_{2.1} = k_2 \cdot \left(\frac{M}{2^j}\right)^3 \cdot \left(\frac{s+1}{h}\right)^2 \sim 2^{-3j} \cdot M^3, \tag{4.13}$$

where the droplet support for the B-spline scaling function is $s(\varphi_n) = (n_v + 1) + (n_g + 1)$, or, if using the B-spline wavelet $s(\psi_n) = (2n_v - 1) + (n_g + 1)$. Note that the factor 2^j at the support s has disappeared in eq.(4.13) compared to eq.(4.12), because the sampling step h_j of the grid adapts to the scale j with $h_j = 2^j \cdot h$.

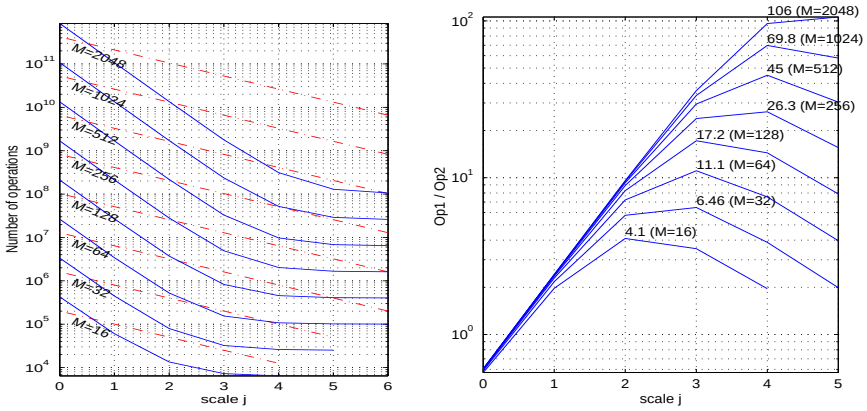
Once the volume has been projected on the grid, the projected data needs to be interpolated to the full-screen resolution N^2 and visualized in cardinal space. This is achieved in three steps: The first step changes from dual space to spline space, which is fast because the data size is small. The spline space has the shortest—fastest—upsampling interpolation filters. In the second step, these filters are used to reach the final full resolution. The last step changes from spline to the cardinal space for the purpose of visualization.

The total number of operations of all steps in Method 2 is:

$$\begin{aligned} \mathrm{Op}_2 &= k_2 \left(2^{-j} M\right)^3 \cdot (3n+1)^2 h^{-2} + \\ &+ k_3 N^2 \left(2^{-2j} h^{-2} + 2^{-1}\right) \cdot (2n+1)^2 \sim \\ &\sim 2^{-3j} \cdot M^3 + \frac{k_3}{k_2} N^2 \end{aligned} \tag{4.14}$$

Comparison of methods

The complexity of both methods are compared in Figure 4.5 for $h = 1$ (no oversampling) and $M = N$, which means that the rendering screen has the same resolution per dimension as the volume data. The complexity of Method 2, Op_2, is significantly lower than the complexity of Method 1, Op_1, for coarse scales $j > 1$. Method 2 is up to two order of magnitude faster at scale around $j \in \{4, 5\}$ and for large volumes $M \in \{512, 1024, 2048\}$. The complexity Op_2 decays with a factor 8^{-j} per scale j at small scales, until it reaches a constant that depends only on the screen size N^2. The factor 8^{-j} per scale j is already the theoretical limit, as the complexity cannot decrease faster than there are voxels per scale j. The complexity Op_1 falls only with a decay of 2^{-j}. A clever algorithm can adapt N and j to the user behavior, reduce the resolution (e.g. zoom into areas of interest) and take advantage of the high quality of the



(a) (b)

Figure 4.5: Complexity comparison of wavelet splatting methods (Op_1) and wavelet projection method (Op_2): (a) The number of operations are plotted in logarithmic scale; Op_1 as dashed line and Op_2 as solid line. The complexity is reduced with coarser scale: Op_2 decays faster with up to 2^{-3j} per scale j, whereas Op_1 decays with only 2^{-j}. (b) The ratio $\mathrm{Op}_1/\mathrm{Op}_2$ is plotted over the scales j and for several volume resolution M^3. For large M's, our proposed method can be up to 100-times faster than the first one.

interpolation. Essentially, Method 2 trades speed for quality, because the droplets at coarser scale are approximated also with a coarser sampling step.

Result images

We rendered the lobster volume with the linear B-spline model, see Figure 4.6. On an animated sequence of rendered volumes, one can see flickering artifacts for the nearest neighbor model (Haar, $n = 0$). When the model is improved to linear interpolation model $(n = 1)$ the flickering is significantly suppressed.

4.4 Conclusion

We have proposed a volume rendering method based on droplets and multiresolution approximation grids. It allows full control over the approximation error and achieves the smallest error for a given sampling step. To speed up computation, one may sacrifice some quality (see Figure 4.3 and 4.6) and adapt the rendering step to the size of the basis function. With this strategy, the proposed method can be up to two order of magnitude faster than standard ones [26]. The speed-up breaks down only at full resolution, where the complexity of the proposed algorithm

is 50% higher than the direct wavelet splatting method; but we end up with a higher quality result, especially when the data has a significant high frequency content. To conclude, wavelet projection with approximation grids may speed up interactive or progressive volume rendering; it yields higher quality for a given resolution.

4.5 Main achievements

- Replacement of interpolation-based splatting by least-squares approximation of the projected wavelet (droplet).
- Proposition an acceleration technique with a multi-resolution projection grid that adapts to the scale of the droplet.

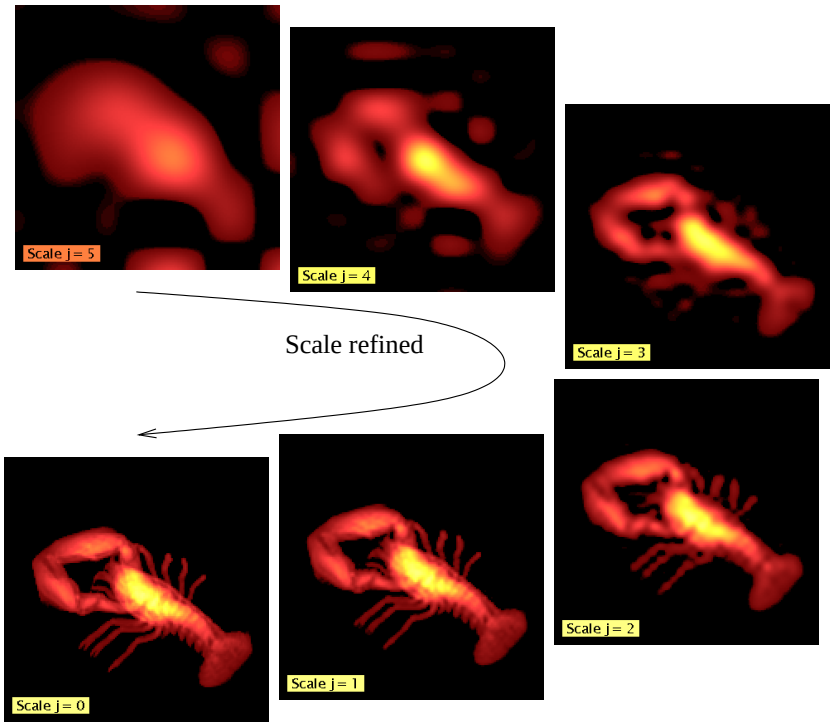


Figure 4.6: Multiresolution pyramid of a lobster at scales $j \in \{0, \ldots, 5\}$. The volume can be rendered faster at coarser scale, e.g., up to 64-times faster at scale $j = 2$. This allows a viewer to rotate the volume interactively. When the rotation stops a finer scaled version is rendered.

Chapter 5

Recursive Texture Mapping

"I hear and I forget. I see and I remember. I do and I understand."
—CONFUCIUS

5.1 Introduction

Raster-based image presentation is very common in digital imaging. The images are composed of regularly sampled pixels (e.g., used on LCD screen) or dithered points (e.g., used in newspapers), similar to a painting of the pointillist style that is composed of paintbrush spots. Another way to describe images is by geometric primitives, e.g., lines, boxes, circles, curved shapes, polygons that have attributes, like color, specularity, or patterns. This is similar to paintings of the cubist style. The drawback is that it is expensive to represent complex images in this kind of description.

Texture mapping merges the two approaches. It consists in warping a rastered image onto the surface of a 3D object. This is useful to simulate, for example, the wooden texture of a table. Volumetric textures are defined in three dimensions; they are used for example to describe a solid block of wood out off which a 3D surface is cut. This surface has already a well defined texture, which is consistent in 3D.

In this chapter, we show the limits of the standard texture-mapping methods and propose a novel algorithm that is based on least-squares approximation and successive refinement. In Section 5.2 we give a state of the art of texture mapping. In Section 5.3 we describe our novel recursive texture mapping algorithm. Later, in Chapter 6, we will discuss view-dependent texture mapping and, in Chapter 7, its application for joint mesh and texture coding.

5.2 State of the art

We summarize texture-mapping methods from several articles [11], [13], [18], [20], [30], [53], [109] and three surveys [31], [81], [108].

Nearest neighbor is the fastest method but it gives the worst quality. It maps the pixel's screen coordinates to texture coordinates and copies the value of the closest texel[1]. *Bilinear interpolation* combined with mipmapping pyramids is a common technique, since it is a good compromise between speed and quality. *Supersampling* gives the highest quality but is the slowest method.

The higher-quality methods generally use some kind of anti-aliasing filter. Typically, one determines the footprint of a screen pixel in texture space, and then takes the weighted sum over all texels inside the borders of this influence area. The weights are defined by an empirical filter function. The difference between the methods is the filter function. *Area sampling* uses a box filter, *bilinear interpolation* uses the tent filter, whereas *elliptical weighted area (EWA)* uses a circular Gaussian filter assuming a circular pixel support. Two acceleration methods based on tables are *mipmapping* pyramids and *summed-area tables*. Additionally, texture-mapping algorithms can be classified into *texture-space scan* and *screen-space scan* algorithms.

5.2.1 Supersampling

A high-quality, but computational expensive method is *supersampling*. It renders the image at higher resolution first, typically a 4×4 zooming factor, and then reduces the rendered image to the final resolution using common anti-aliasing filters. Note that we can improve the quality of the last reduction step by using projection-based sampling instead (cf. Chapter 2). The rendering costs are proportional to the superresolution; that is the price to pay for a simple but computational demanding method. We can use this method as a quality reference and look at faster algorithms.

5.2.2 Scan order

We can distinguish between *two-pass* and *one-pass* texture mapping methods. The two-pass methods map the texture first to an intermediate space and then from it to the screen space. This works fine for affine image transformations, which can be decomposed into a horizontal shearing and scaling followed by a vertical shearing and scaling. The advantage of these one-dimensional, separable transforms is that we can apply projection-based re-sampling, which give the best results in the least-square sense (c.f. Chapter 2) [48, 105]. For more general transformations, we have to consider one-pass texture mapping methods, which are not separable anymore. They are classified into two different algorithms:

1. The *screen space scan* algorithm computes for each screen pixel (x, y) its position (u, v) in texture space. The texel at (u, v) is copied to the pixel at (x, y). As the coordinates

[1] A texel is an elementary point in the texture. Texel comes from **tex**ture **el**ement, as pixel from picture element, or voxel from volume element.

(u, v) are in general non-integer, the value of the texture needs to be determined at intermediate coordinates, e.g. by choosing the *nearest neighbor* or by *bilinear interpolation*. The advantage of screen space scan is the constant cost per pixel.

2. The *texture space scan* algorithm takes each texel (u, v) and computes its coordinates in screen space (x, y). The texel is copied to the pixel closest to (x, y), or distributed between the neighboring pixels. The disadvantage of texture space scan is that the cost per pixel is not constant, but proportional to the texture size.

5.2.3 The pixel footprint and area sampling

The *pixel footprint* is defined as the support of the screen pixel mapped in the texture space. For arbitrary complex mappings, the pixel footprint can be irregularly shaped. However, for perspective projection, the quadrilateral defined by four projected points of the pixel extent provides a good approximation of the pixel footprint over texture space [11].

Texture mapping based on pixel footprint works as follows:

1. Determine the support of a pixel on the screen.
2. Map this support to the texture space; this determines the pixel footprint.
3. The texels inside the pixel footprint are weighted by a filter function, which is normalized to one. This filter function can be interpreted as the shape of the pixel mapped to the texture.
4. The sum of the weighted texels is assigned to the pixel.

The shape of the filter function is critical. The ideal low-pass filter is the separable 2D $\text{sinc}(\boldsymbol{x}) = \text{sinc}(x) \cdot \text{sinc}(y)$. However, it has infinite support and is thus impractical. Consequently, finite impulse filters are used. Examples are 2D tensor-product versions of the box function (*area sampling method*), the triangle, the cubic B-spline and the truncated Gaussian.

5.2.4 Mipmapping pyramid and trilinear interpolation

The mip prefix in mipmap [109] comes from the Latin *multum in parvo*—many things in a small place. The dyadic mipmapping pyramid is a series of downsampled[2] textures. Assuming that a pixel covers an approximately planar area in screen space, a square pixel maps into a quadrilateral in texture space. Except near singularities, e.g., vanishing points, the quadrilateral is approximately a parallelogram. The size of the parallelogram is given by the partial derivatives. They can be approximated by finite differences between the $(u,\ v)$ values of adjacent pixels.

The ordinary mipmap pyramid is square and dyadic. This means that the pixel footprint is approximated by one or two squares. The texture mapping process deforms the texture; locally

[2]Note that the coarsening is done best by projection-based downsampling.

the shrinking can be approximated by a texture minification factor z^{-1}. It is defined as the ratio between the original size of a texture patch and its size after the mapping—the larger z^{-1}, the smaller and coarser the mapped texture patch is. Ewins et al. [18] compare several methods to calculate the texture minification factor z^{-1} based on partial derivates, on finite differences, on hypotenuses, on cross-products, and on area ratios. Erring on the side of blurring, Heckbert [31] calculated the texture minification factor z^{-1} by the maximal hypotenuses length of the approximate quadrilateral:

$$z^{-1} = \max\left(\sqrt{u_x^2 + v_x^2}, \sqrt{u_y^2 + v_y^2}\right),$$

where (u_x, v_x) and (u_y, v_y) are the partial derivatives of the texture coordinates (u, v) in the screen coordinates (x, y). We can determine the mipmap level l from the texture minification factor z^{-1} by

$$l = \log_2 z^{-1}.$$

In most cases $\log_2 z^{-1}$ is non-integer. It would be possible, but impracticable to calculate the texture at that scale $\log_2 z^{-1}$ every time. Choosing the next finer dyadic mipmap-level $l_{\text{finer}} = \lfloor \log_2 z^{-1} \rfloor$ can lead to aliasing, because the texture is shrunk without proper prefiltering. Choosing the next coarser level $l_{\text{coarser}} = \lceil \log_2 z^{-1} \rceil$ leads to blurring, because the resolution is lower. The compromise is to interpolate the pixel value in each of the two adjacent levels and then to interpolate the results from both levels. The result from the coarser mipmap-level l_{coarser} contributes with a weight of $a = (z^{-1} - 2^{l_{\text{finer}}})/2^{l_{\text{finer}}} \in [0, 1]$ and the result from the finer mipmap level l with a weight of $1 - a$ to the final pixel. Due to the interpolation over three dimensions (x, y and j), mipmapping is also called *trilinear interpolation*, or trilinear filtering. Mipmapping is probably the most widely used texture filtering method because it yields acceptable results, is fast and requires little additional memory—in the limit, $1/3$ of the original texture size.

The drawback of mipmap pyramids is that they can give rise to flickering artifacts, visible when textured objects move. The reason is the following: When the pixel footprint moves across the texel boundary, the change in the pixel value is more abrupt at coarse scales than at the full resolution. Aliasing occurs when too fine a level is selected.

5.2.5 Summed area table

Mipmapping works with square pixel footprints, whereas *summed area table (SAT)* [13] works with more general rectangular footprints. Each entry in a summed area table $s(a, b)$ represents the sum of the intensities of all texels in a rectangle defined by the lower left corner $(0, 0)$ and the pixel of interest (a, b). Dividing by the number of pixels in the rectangle gives the average intensity. To get the sum of all texture pixels in a arbitrary rectangle defined by a lower left corner (a, c) and a upper right corner (b, d), we have to read only four table entries (see Figure 5.1) and calculate the sum of the rectangle with $s(b, d) - s(b, c) - s(a, d) + s(a, c)$. A SAT has the same size as the original texture, but the accumulated texel values have to be represented with higher

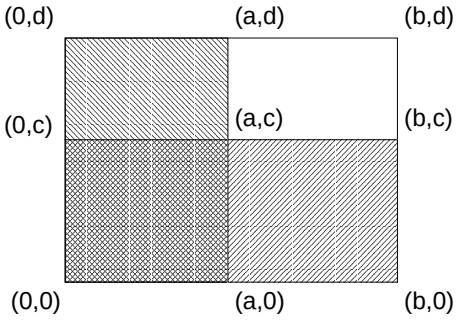


Figure 5.1: Summed area table

precision range, which requires more memory. SAT can only be used with a constant model (spline of degree $n = 0$). The quality of SAT is better than zero-degree mipmapping, since SAT filters approximate rectangular areas, not just square ones; but SAT is four-times slower than mipmapping $(n = 0)$.

The simple summed area table may result in image blur, when the pixel footprint is not a rectangle aligned coordinates. Glassner [21] has shown how rectangle can be successively redefined to approximate a quadrilateral.

Heckert [30] generalizes sum-area tables to space-variant filters with higher interpolation degrees by repeated integration of the box function (B-spline of degree 0). The algorithm differentiates the signal n-times until only an impulse comb remains and then uses repeated integration (simulating one-sided power functions) to reconstruct it; this is similar to a B-spline transform and reconstruction. The approach can be extended to any polynomial kernels, e.g., the Overhauser and Catmull-Rom cubic spline. The method is faster than direct convolution with wide kernels. The algorithm is a generalization of the add-incoming, subtract outgoing box blur algorithm.

5.2.6 Elliptical weighted average (EWA)

The *elliptical weighted average (EWA)* filter is an example of space-variant filter, because its support varies as the filter moves across the image. Space-variant filters are required when good anti-aliasing is needed and when non-affine mappings are used. Most texture filters consider square pixels. Nonlinear mappings, e.g., perspective projections, seriously distort square pixels, and may result in very large footprints especially near the horizon. However, circular regions in image space transform approximately to ellipses in texture space—except near vanishing points (e.g., horizon), where they transform into parabola or hyperbola. Hence, by considering isotropic pixels in the image space we can approximate their footprint by ellipses; these are characterized by the size, orientation and eccentricity.

The EWA algorithm is similar to the one described in Section 6.2.3, except that it uses an adapted circular weighting function; Heckbert [31] uses a Gaussian filter function. This ad-hoc

solution seems to work fine, but least-square approximation should yield better quality.

The drawback of EWA is the high computational cost. Recently two faster methods were proposed that approximate the EWA algorithm: Texram [83] provides higher visual quality than trilinear filtering with less complexity than EWA. It uses a series of circular filter along a line that approximates the length and slope of the major axis of EWA's elliptical footprint. Felines [71] (**F**ast **el**liptical **lines**), like Texram method, uses several isotropic probes along a line to implement an anisotropic filter. However it calculates the sampling line length more accurately and weights the filters by a Gaussian function. The visual quality of Felines is better than Texram for little additional costs, according to [71].

5.2.7 Forward mapping

Ghazanfarpour [20] proposed a forward texture mapping technique based on accurate convolutions which reduces aliasing in the final image. The algorithm loops over the texture and not the screen. Each texel is mapped to the screen and its weighted value is added to the screen pixel. The weighting is done by a radial filter function (such as a Gaussian). The weighting is normalized to ensure that a constant texture keeps the same constant value after the warping process. The runtime depends largely on the texture size. The authors did not propose multi-resolution or recursive schemes.

Another technique relies on forward and backward texture mapping in B-spline space [53]. This content-based hybrid method applies a *EWA filter combined with directional moving average filtering (EWADMA)* for blocks with a directional preference. The B-spline transform is performed by the recursive moving averaging interpolation. The interpolation function is a directional B-spline oriented at four angles $(0°, 45°, 90°, 135°)$. The authors compared the original texture with the back-mapped texture to evaluate the quality of their algorithm (for details see [53]).

5.2.8 Summary

We have reviewed most of the texture mapping methods. The best quality is achieved by super-resolution. For fixed resolution, EWA and its faster variants are the best known methods. The most common texture mapping method is mipmapping with trilinear interpolation. In the next section, we will describe a different approach to texture mapping. It minimizes the information loss of the texture mapping process by successive refinement.

5.3 Least-squares texture mapping

Our leading idea is to reformulate the texture mapping as an optimization problem for which the goal is to preserve the maximum amount of information in the mapped texture. In other words, when we undo the texture mapping, we would like to reconstruct the original texture as well as possible. We derive a solution that is optimal in the least-squares sense and that

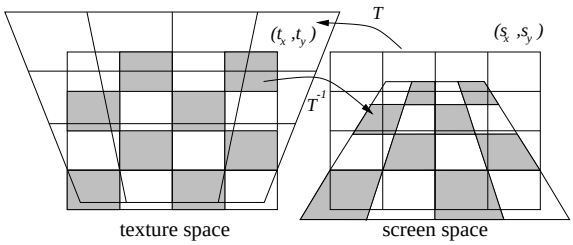


Figure 5.2: Transformation T between screen space and texture space

corresponds to the pseudo-inverse of the texture-mapping transformation. In practice, a first-order approximation of the least-squares solution is used as an initial estimate for the mapped texture. This initial solution is refined by successive approximation to yield the least-squares optimal result. In essence, the proposed multi-pass method acts like an adaptive anti-aliasing filter.

5.3.1 The texture-mapping problem

The texture-mapping problem is to map some input texture image f_t to produce a screen image f_s

$$f_s(s_x, s_y) \cong f_t(T(s_x, s_y)), \tag{5.1}$$

$$T(s_x, s_y) = (t_x, t_y), \tag{5.2}$$

where the geometrical transformation T—e.g., a perspective mapping on a 3D surface as seen in Figure 5.2—maps the screen coordinates (s_x, s_y) from the screen space into the texture coordinates (t_x, t_y) in the texture space. Assuming that T is invertible, the dual version of the problem is

$$f_t(t_x, t_y) \cong f_s(T^{-1}(t_x, t_y)), \tag{5.3}$$

$$T^{-1}(t_x, t_y) = (s_x, s_y), \tag{5.4}$$

where T^{-1} is the reverse mapping of T. We have used an approximate equality symbol in (5.1) and (5.3) because the mapping will not be perfect—both image and texture grids are discrete, which entails some loss of information.

5.3.2 Continuous-screen model

We introduce a continuous model f_s associated with the screen. This function lives in the screen space generated by the basis functions $\varphi_s(s_x - k, s_y - l)$ located at the pixel positions (k, l). The model is specified by

$$f_s(s_x, s_y) = \sum_{k\in\mathbb{Z}} \sum_{l\in\mathbb{Z}} c_s(k, l) \cdot \varphi_s(s_x - k, s_y - l), \tag{5.5}$$

where (s_x, s_y) are the screen coordinates, and φ_s is the generating function; for example, a separable B-spline or any other standard interpolation function.

We can now attempt to recover the original texture f_t by applying the inverse mapping (5.3)

$$\tilde{f}_t(t_x, t_y) = f_s(T^{-1}(t_x, t_y)). \tag{5.6}$$

5.3.3 Least-squares criterion

We assume that the texture model f_t is given. Then, we optimize the texture mapping on the screen f_s by minimizing the approximation error

$$\epsilon_t = \min_{c_s} \left\| f_t(t_x, t_y) - f_s(T^{-1}(t_x, t_y)) \right\|_{L_2}^2 .$$

Note that we measure the approximation error in the texture space because this is the space where the original texture f_t is specified.

5.3.4 Discrete texture mapping

In the following, we assume that the input texture is specified by its texel values $f_t(\boldsymbol{m})$, $\boldsymbol{m} = (m, n)$. Together with the inverse texture mapping (5.3) and with the continuous-screen model (5.5), the transformation of the screen coefficients $c_s(\boldsymbol{k})$, $\boldsymbol{k} = (k, l)$ to the texels f_t is described as

$$\tilde{f}_t(\boldsymbol{m}) = \sum_{\boldsymbol{k}\in\mathbb{Z}^2} c_s(\boldsymbol{k}) \cdot \varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k}). \tag{5.7}$$

In order to use matrix notation, we rearrange the two-dimensional arrays (images and arrays of coefficients) in raster-scan order to one-dimensional arrays:

$$\tilde{\boldsymbol{f}}_t = \left[\tilde{\boldsymbol{f}}_t(\boldsymbol{m})\right] = \left(\tilde{f}_t(1,1), \cdots, \tilde{f}_t(1,N), \cdots, \tilde{f}_t(M,1), \cdots, \tilde{f}_t(M,N)\right),$$

where (M, N) is the size of the texture. The same applies for $\boldsymbol{c}_s = [c_s(\boldsymbol{k})]$, where (K, L) is the size of the screen.

The key idea is to determine the $\boldsymbol{c}_s$ such that $\tilde{\boldsymbol{f}}_t$ is a good approximation of $\boldsymbol{f}_t$. The equation (5.7) in matrix form is

$$\tilde{\boldsymbol{f}}_t = \boldsymbol{A} \cdot \boldsymbol{c}_s, \tag{5.8}$$

$$\boldsymbol{A} = [a(\boldsymbol{m}, \boldsymbol{k})] = \left[\varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k})\right]. \tag{5.9}$$

In order to correctly reproduce a constant, we require that φ_s satisfies the partition of unity:

$$\sum_{k \in \mathbb{Z}} \varphi_s(x - k) = 1. \tag{5.10}$$

A direct implication is that the rows of $\boldsymbol{A}$ sum up to one:

$$\sum_{\boldsymbol{k}} a(\boldsymbol{m}, \boldsymbol{k}) = 1.$$

5.3.5 Least-squares solution

Given the original texture f_t, we want to minimize

$$\min_{\boldsymbol{c}_s} \left\| f_t(m,n) - \tilde{f}_t(m,n) \right\|_{l_2}^2 = \min_{\boldsymbol{c}_s} \| \boldsymbol{f}_t - \boldsymbol{A} \cdot \boldsymbol{c}_s \|_{l_2}^2 \tag{5.11}$$

to get the least-squares solution $\boldsymbol{c}_s$. The solution can be expressed explicitly in terms of $\boldsymbol{A}^+$, the pseudo-inverse of $\boldsymbol{A}$:

$$\boldsymbol{c}_s = (\boldsymbol{A}^\top \boldsymbol{A})^{-1} \boldsymbol{A}^\top \cdot \boldsymbol{f}_t = \boldsymbol{A}^+ \cdot \boldsymbol{f}_t.$$

Because of the large size of the involved matrices, the direct computation of the inverse $(\boldsymbol{A}^\top \boldsymbol{A})^{-1}$ is too expensive to be carried out in practice. In first approximation, we will assume that it is close to a diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1 \ldots \lambda_{\boldsymbol{k}}) \cong (\boldsymbol{A}^\top \boldsymbol{A})^{-1}$ and we write

$$\boldsymbol{A}^+ \cong \tilde{\boldsymbol{A}}^+ = \Lambda \cdot \boldsymbol{A}^\top = \left[\lambda_{\boldsymbol{k}} \cdot \varphi_s(T^{-1}\boldsymbol{m} - \boldsymbol{k})\right]. \tag{5.12}$$

The partition of unity (5.10) must again be fulfilled. This time, all columns of $\boldsymbol{A}^+$ have to sum up to one, due to the transposition of $\boldsymbol{A}$. We set the elements of Λ to be

$$\lambda_{\boldsymbol{k}} = \frac{1}{\sum_{\boldsymbol{m}} \widetilde{a}(\boldsymbol{m}, \boldsymbol{k})},$$

which ensures that the partition of unity is satisfied by the inverse approximation as well.

5.3.6 Successive approximation

The first-order approximation of the texture mapping is

$$\boldsymbol{c}_s = \boldsymbol{A}^+ \cdot \boldsymbol{f}_t \cong \Lambda \boldsymbol{A}^\top \cdot \boldsymbol{f}_t. \tag{5.13}$$

This solution can be improved by successive refinement, where the error correction term is $\boldsymbol{B} = (\boldsymbol{I} - \tilde{\boldsymbol{A}}^+ \boldsymbol{A})$. We give the successive least-squares texture-mapping algorithm in pseudo-code language:

$\boldsymbol{c}_s^0 \cong \widetilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t, \quad i = 0$ // initial approximation
DO $\quad i = i + 1$
  $\Delta \boldsymbol{f}_t = \boldsymbol{f}_t - \boldsymbol{A}\boldsymbol{c}_s^{i-1}$ // map back to get loss
  $\Delta \boldsymbol{c}_s = \widetilde{\boldsymbol{A}}^+ \Delta \boldsymbol{f}_t$ // map the loss to screen
  $\boldsymbol{c}_s^i = \boldsymbol{c}_s^{i-1} + \Delta \boldsymbol{c}_s$ // correct the screen
WHILE ($\|\Delta \boldsymbol{f}_t\| > \epsilon$ and $\|\Delta \boldsymbol{f}_t\|$ decreasing)

Rearranging the formulas, we get the recursive form

$$\boldsymbol{c}_s^i = \boldsymbol{B} \cdot \boldsymbol{c}_s^{i-1} + \widetilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t \tag{5.14}$$

and from it the closed form solution

$$\boldsymbol{c}_s^i = \boldsymbol{B}^i \cdot \boldsymbol{c}_s^0 + \sum_{k=0}^{i-1} \boldsymbol{B}^k \cdot \widetilde{\boldsymbol{A}}^+ \cdot \boldsymbol{f}_t. \tag{5.15}$$

The term with the initial approximation $\boldsymbol{c}_s^0$ of the screen in (5.15) vanishes with increasing iterations i. The conditions for convergence are that the matrix $\tilde{\boldsymbol{A}}^+$is well conditioned, and that the eigenvalues of $\boldsymbol{B} = (\boldsymbol{I} - \tilde{\boldsymbol{A}}^+ \boldsymbol{A})$ are smaller than 1. Since the matrices $\boldsymbol{A}$ and $\boldsymbol{A}^+$ strongly depend on the geometric mapping T, we cannot guarantee that the algorithm will converge for all possible configurations. In the unlikely event where the algorithm fails to converge, it is always possible to stabilize it by introducing a damping factor $\gamma < 1$ in the corrections.

The inversion of the matrix $\boldsymbol{A}$ makes sense only in areas where the texture is shrunk while mapped to the screen. Otherwise, the texture mapping problem is ill-posed and there are many possible solution. In the under-determined case, the texture is mapped by interpolation with the classical method instead (e.g., bilinear mipmapping). In the under-determined area the normalization weights $\lambda_{\boldsymbol{k}}$ are smaller than 1, because too few texels contribute to one screen pixel; typically, less than 4 for the bilinear case ($n = 1$). Based on these two rules, we determine the area on which to successively refine the screen.

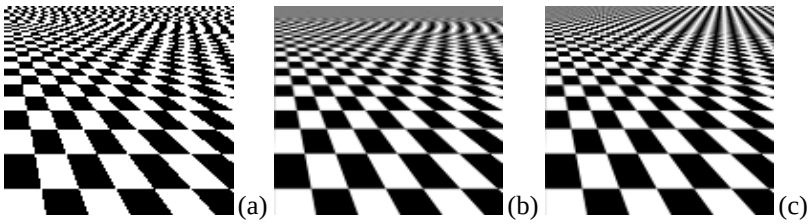


Figure 5.3: Comparison of map textures: (a) by nearest-neighbor method, (b) by trilinear mipmapping, (c) by successive refinement with trilinear mipmapping.

Border cases

There are two border cases where a screen pixel can map partially outside the defined texture area: 1. A pixel on the horizon maps to an infinitive area on the texture and beyond it. The ad-hoc solution is to blend between the texture and the background, e.g., by assigning the value of the background to the area outside the texture. 2. A pixel can be part of two planes (e.g. neighboring polygons): We suggest to calculate a separate solution for each plane and blend the values.

5.3.7 Multiresolution version

A pixel that is close to the horizon typically maps to a large texture area (see recursion at fine scale: Figure 5.4). For such large texture reduction factors, we switch to a lower texture resolution (see recursion at coarse scale: Figure 5.5). This reduces the computational cost.

We use a mipmap-like technique to combine the different resolutions. We calculate for each pixel locally the texture minification factor z^{-1} from the geometry of the projection (see Section 5.2.4) and take for each pixel the solution from the next finer scale $j_{\text{finer}} = \lfloor \log_2 z^{-1} \rfloor$. We are allowed to does this, because our successive refinement method suppresses aliasing; the results produced at finer scale are of better quality but also more costly. In Figure 5.6 we show the selected areas; we combine them by “copy-paste”. Further improvements can be achieved by blending the solutions obtained at the next finer scale $j_{\text{finer}} = \lfloor \log_2 z^{-1} \rfloor$ and at the next coarser scale $j_{\text{coarser}} = \lceil \log_2 z^{-1} \rceil$ scale (see Figure 5.7); this is similar to trilinear filtering. The quality of the result is better, but the computational cost is twice as high as the simple “copy-paste” method, because for each selected area the solution is calculated at two scales, before they are blended.

5.4 Experimental Results

To demonstrate our algorithm, we map a checkerboard onto an infinite plane (see definition of projection in Appendix 5.A). The results of the simple nearest-neighbor texture-mapping algorithm are shown in Figure 5.3a. This image exhibits severe aliasing artifacts. Trilinear mipmapping—a good standard method—reduces these effects (Figure 5.3b). The proposed method (Figure 5.3c) avoids the aliasing effects because it implicitly acts like an optimal anti-aliasing filter. Here, the underlying continuous model for the successive-refinement method is a bilinear spline.

The first three iterations of our recursive texture mapping algorithm are shown in Figure 5.4. The texture mapped to the screen is depicted in part (a); this screen is mapped back to the texture space (part (b)) and the error is evaluated in texture space (part (c)). The resolution of the checker board is high (16×16 pixels per checker), which is sufficient to render the details in the foreground. This resolution is computationally too expensive for the upper image part near the horizon (the background), although it eliminates the aliasing artifacts there. It is faster to reduce the resolution. We did so in Figure 5.5; there the texture resolution is 2×2 pixels per checker. The background is rendered correctly, but too few texels map to the foreground. Hence, we replace the foreground by the image produced with the higher texture resolution (Figure 5.4) or by a standard interpolated image.

The last two figures (5.4 and 5.5) make it evident that we have to combine texture from different scales to achieve a fast algorithm that samples the texture close to the resolution of the screen. The levels can be either combined by hard thresholding (see Figure 5.6) or, better, by blending between two adjacent mipmap levels (see Figure 5.7). The area in the foreground is filled up by an image obtained by standard trilinear texture mapping, which is the method of choice for oversampled areas of the mapped texture.

Area sampling corresponds to the first (and only) iteration of the proposed method, using a spline of degree $n = 0$ (nearest neighbor). For non-trivial cases (degrees $n > 0$), the successive-refinement method gets close to the optimal biorthogonal prefilter after several iterations. This filter resembles the filter of the EWA method, but tends to be much sharper. In addition, our method guarantees the partition of unity.

To verify our method, we consider the simple reduction-by-two case because the l_2-optimal solution is known and can be obtained by alternative means, as shown in [98]. Experimental results for different interpolation methods are shown in Figure 5.8. The cubic interpolators (Keys and B-spline) tend to give better results than the linear one. The PSNR in texture space (given in Figure 5.9) is higher and the visual appearance is slightly better.

The graph in Figure 5.9 gives the evolution of the error in texture space—the criterion optimized by the algorithm—and illustrates the convergence behavior. We note that the basis functions that are more localized (linear spline and Keys) converge in less iterations. The cubic spline needs more iterations but finally outperforms the other methods. In Figure 5.9 we verified that the algorithm converges to the l_2-optimal solution obtained by [98].

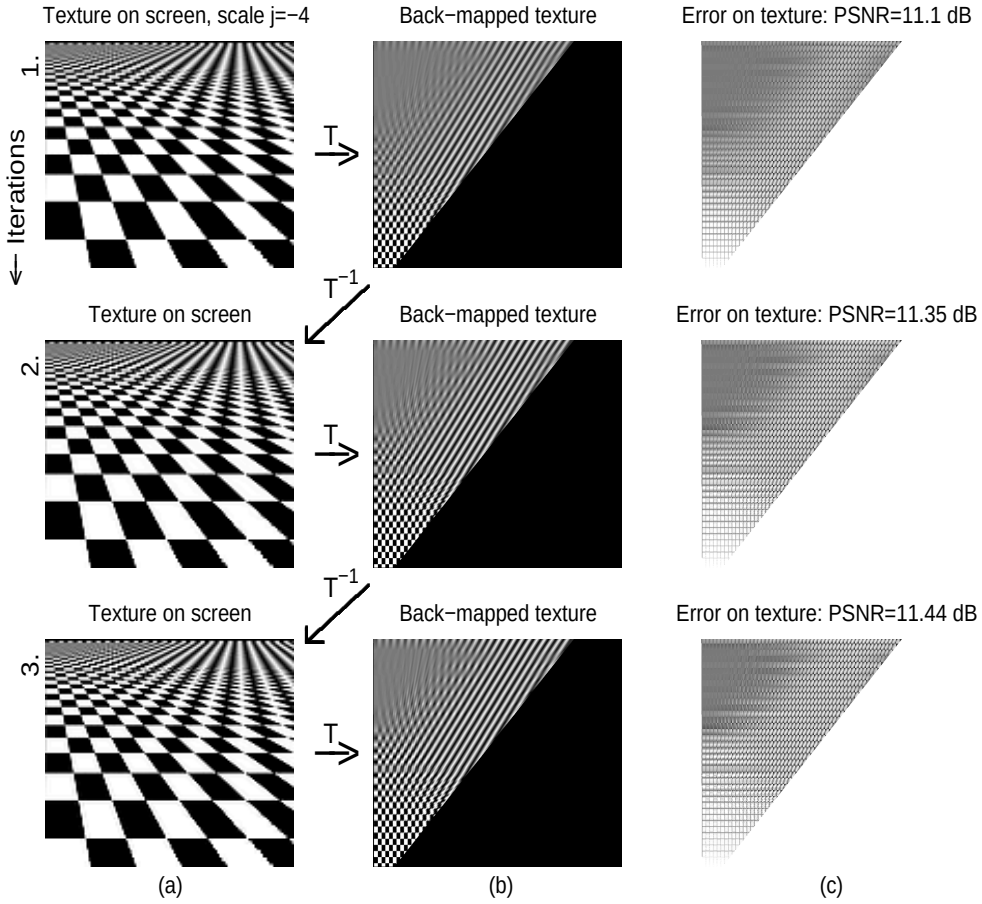


Figure 5.4: Recursive texture mapping of checker board at fine scale (first 3 iterations shown, bilinear interpolation). The scale of the texture is fine ($j = -4$, 16×16 pixels per checker), therefore the close foreground is rendered detailed enough, whereas the fare background is not rendered as the memory requirements are too high: (a) The texture is mapped to the screen. (b) The screen is mapped back to the texture space. (c) The error between (b) and the original texture decreasing with each iteration.

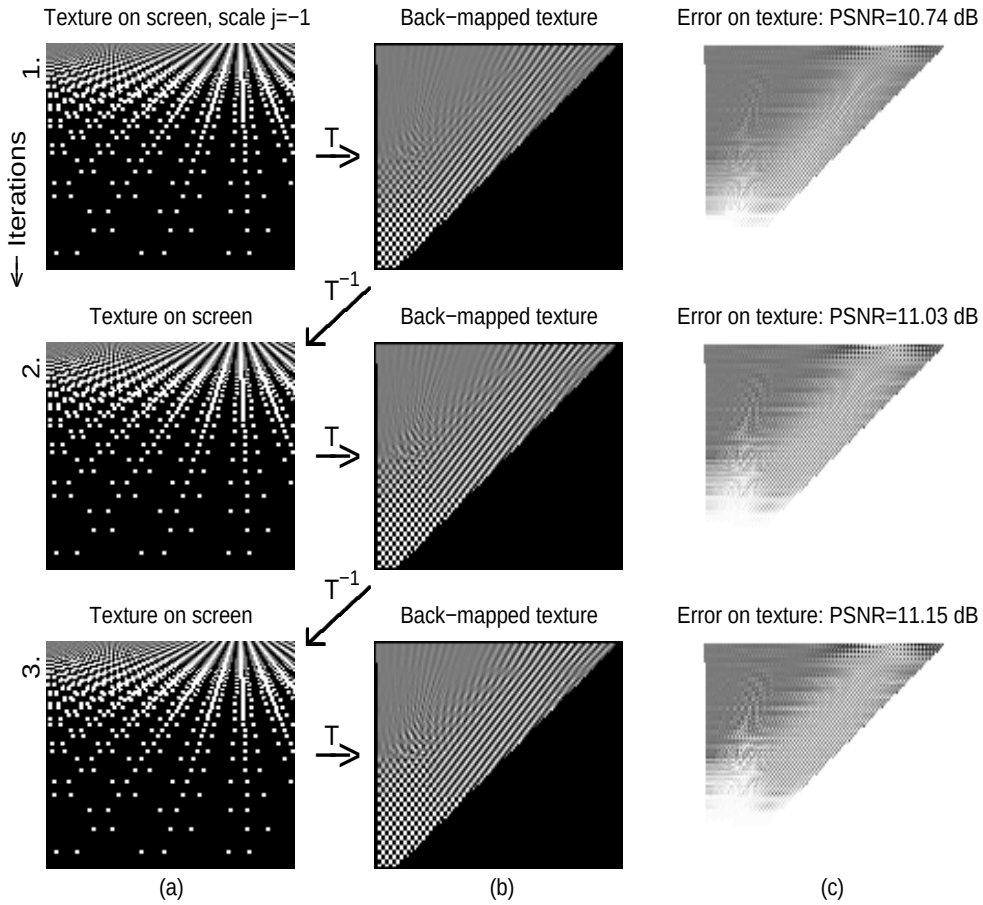


Figure 5.5: Recursive texture mapping of checker board at coarse scale (first 3 iterations shown, bilinear interpolation). The scale of the texture is coarse ($j = -1$, 2×2 pixels per checker). This resolution is sufficient to render the background fast, but the foreground is holey: (a) The texture is mapped to the screen. (b) The screen is mapped back to the texture space. (c) The error between (b) and the original texture decreasing with each iteration.

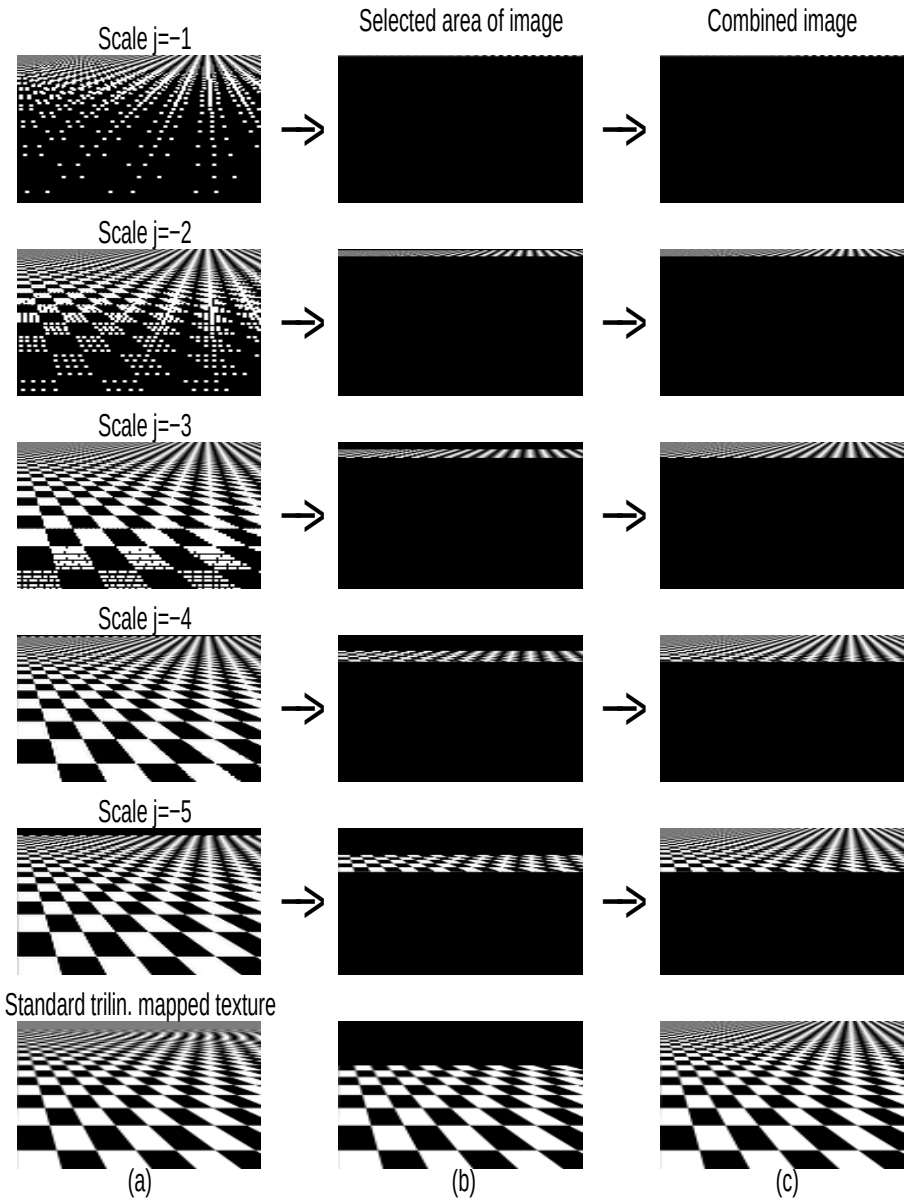


Figure 5.6: Multiresolution recursive texture mapping with *bilinear interpolation*: (a) The texture map is computed for each scale j separately. (b) According to the projection geometry the part of the image with the proper resolution is selected (a practical algorithm would compute only this area at that scale). (c) The selected areas are combined by "copy-paste" in the final image. In the last row we fill up the foreground of the image with a standard interpolated texture map.

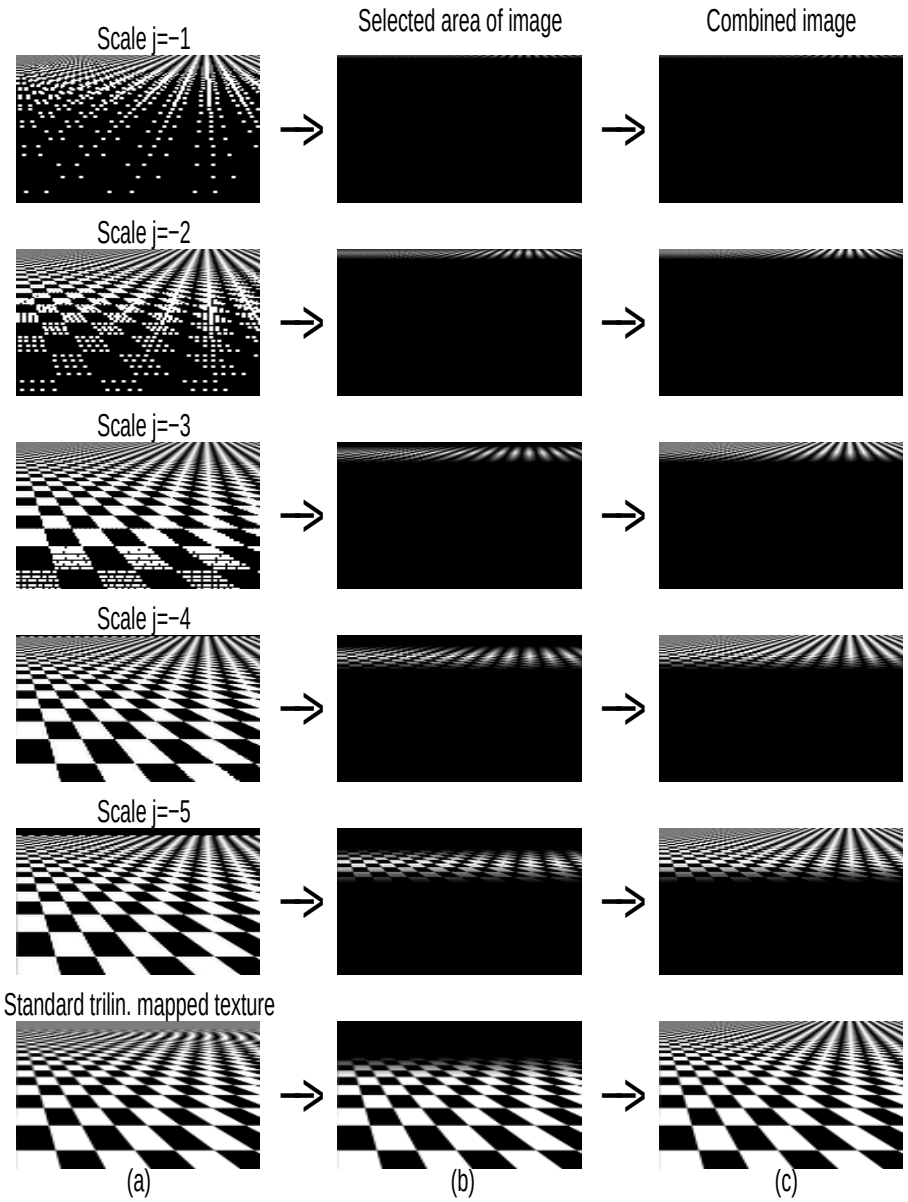


Figure 5.7: Multiresolution recursive texture mapping with *trilinear interpolation*. The difference to Figure 5.6 is that here we use trilinear interpolation to blend between the level. When you compare the second and third column you see that the select areas of two adjacent levels overlap. But added together they complete each other perfectly.

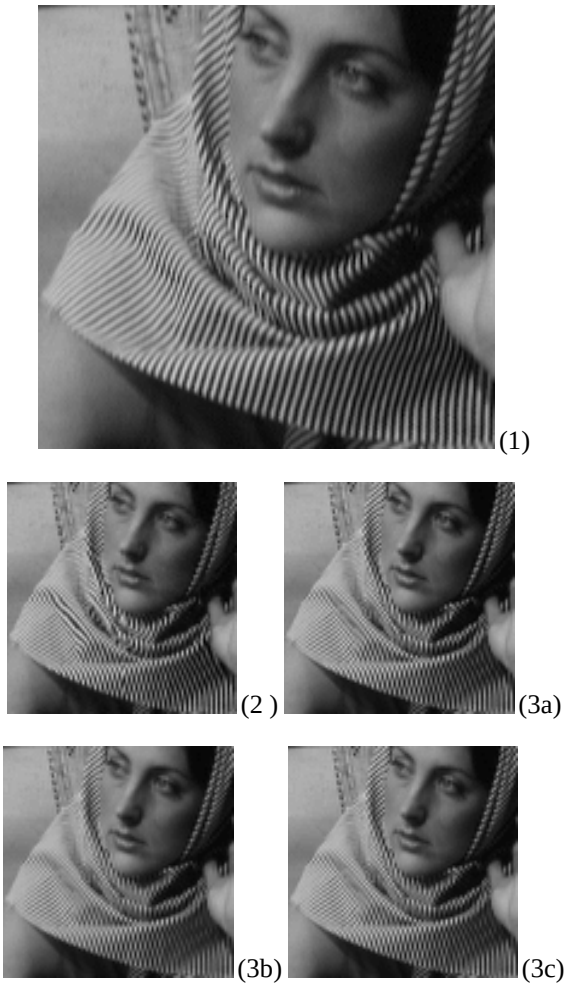


Figure 5.8: Reduction-by-two mapping: (1) Original texture, (2) Simple resampling (aliasing artifacts, PSNR=22.48 dB), (3) Successive refinement after 30 iterations with: (3a) bilinear spline (PSNR=23.14 dB), (3b) bicubic spline (PSNR dB=23.84 dB), (3c) cubic Keys kernel (PSNR=23.61 dB). The PSNR values for each iteration are given in Figure 5.9.

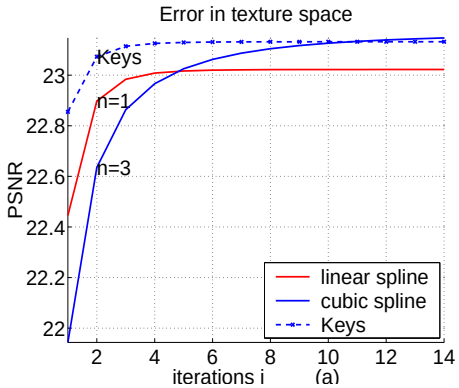


Figure 5.9: The reconstruction quality measured between back-projected texture and original texture increases with the number of iterations.

5.5 Conclusion

We have defined a novel way to solve the texture-mapping problem. The solution minimizes the loss of information and is optimal in the least-squares sense. The practical successive texture-mapping algorithm computes a first approximation of the pseudo inverse of the interpolation matrix $\boldsymbol{A}$ and then uses successive refinement to yield the optimal solution. Our method requires the geometric mapping T to be invertible. The images obtained are more detailed and reveal no aliasing artifacts. Moreover, if our screen model is piecewise constant, then the first iteration of our method is equivalent to area sampling. Our method works together with mipmapping pyramids and interpolation between the mipmap levels.

5.6 Main achievements

- Redefinition of the texture mapping problem to get a solution that minimizes the loss of information.
- Derivation a novel successive refinement texture mapping technique (with multiresolution) that is optimal in the least-squares sense.

Chapter appendix

5.A Perspective projection matrices

The perspective projection depicted in Figure 5.10 is described by the following concatenated matrices. The texture is represented by discrete values with the indices (m, n) and the screen by the indices (k, l). The model we used here is restricted to a tilt and zoom, but can be easily extended by additional rotation matrices (like in II).

	indices	coordinates
texture	(m, n)	(t_x, t_y, t_z)
screen	(k, l)	(s_x, s_y, s_z)

I. Texture indices (m, n) to texture coordinates (t_x, t_y, t_z) in 3D world:

$$\begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2/N_t & 0 & 0 & -(N_t-1)/N_t \\ 0 & 2/N_t & 0 & -(N_t-1)/N_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} m \\ n \\ 0 \\ 1 \end{pmatrix}$$

II. The texture plane is tilted by angle β:

$$\begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix}$$

III. The texture plane is a perspectively projection (with two parameters: focal length f and z-distance d to texture plane center), followed by normalization of the fourth coordinate:

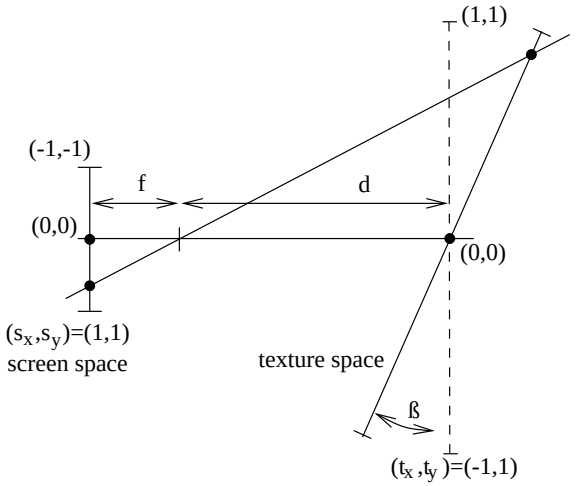


Figure 5.10: Perspective projection of a tilted texture plane

$$\begin{pmatrix} s_x \\ s_y \\ s_z \\ 1 \end{pmatrix} = \frac{1}{d \cdot t_z} \cdot \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 0 & 0 & d & 0 \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \\ t_z \\ 1 \end{pmatrix}$$

IV. From screen coordinates (s_x, s_y, s_z) in 3D world to 2D screen indices (k, l):

$$\begin{pmatrix} k \\ l \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} N_s/2 & 0 & 0 & (N_s-1)/2 \\ 0 & N_s/2 & 0 & (N_s-1)/2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -f \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x \\ s_y \\ s_z \\ 1 \end{pmatrix}$$

Chapter 6

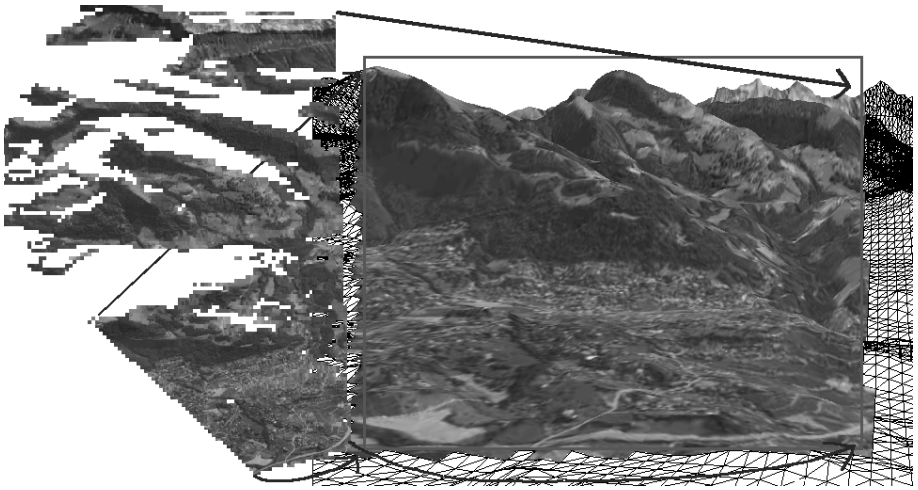
View-dependent Texture Mapping

"How far that little candle throws his beams!
So shines a good deed in a weary world."
—William Shakespeare

6.1 Introduction

In this chapter, we transmit textures that are mapped on surfaces of 3D objects. We take advantage of the losses of the texture mapping process (see also previous chapter) to simplify and compress the texture so that no losses are visible after the mapping. Many factors influence what is left of a texture after mapping and rendering. Three-dimensional scenes undergo a viewing projection, which distorts the textures further and which suppresses texture details. 3D textured objects have front sides that are visible and back sides that are hidden from our view. Parts of the surface can be hidden behind other objects or lie outside the view frustum. Details disappear at large viewing distances or under adverse illumination conditions (fog, shadow). If the kind of transformation that the texture undergoes is known, we can predict which information in the original texture contributes to the rendered scene and which does not. Our goal is to generate a view-dependent texture that has been simplified in advance such that no losses are perceivable after the mapping (see Figure 6.1).

In this chapter, we will derive a heuristic to determine the relevant information to generate the view-dependent texture. For this we analyze the warping process in the frequency and spatial domain. Next, we apply the results to DCT-based and wavelet-based coding. We achieve very high compression rates, first, because only the relevant information is transmitted, and second,



(a) (b)

Figure 6.1: View-dependent texture mapping: (a) The view-dependent texture is simplified. (b) It is then mapped on the mesh. Inside the view frustum (red box) the scene is perfectly rendered.

because we let coder and decoder share the information necessary to render the 3D scene, except for the texture data. This enables both coder and decoder to determine the view-dependent relevant parts of the texture; hence, transmission of this side information is superfluous. We show that view-dependent coding allows to stream texture data fast and at very low bitrates.

6.2 Filtering operations

In this section, we analyze the frequency content of the texture that undergoes a geometric transformation. In the first step, the image is mapped with different transformations (see Figure 6.2 (a)-(d)): zoom, tilt and perspective projection; perspective projection can be locally approximated by an affine transformation, e.g., here by a rotation followed by a tilt. In the second step, the mapped texture is mapped back to the original size. We can see that the back-mapped texture is blurred (see Figure 6.2 (a')-(d')), as during the texture mapping process some image content was lost. We can simulate this loss by directly filtering the original image in the Fourier domain. To determine the necessary filters, we inject an impulse (a black image with a white pixel in the center) in the warping/unwarping process and then compare the amplitudes of the original and back-mapped texture (see Figure 6.3). We are allowed to do this as the transformations are linear and approximately space-invariant. We perform the geometric transformation with different interpolation and approximation models: the nearest neighbor model $(n = 0)$, the linear spline

model ($n = 1$), and the cubic spline model ($n = 3$). In Figure 6.3, we can observe that the filters are fuzzy for $n = 0$, but they become sharper with increasing degrees $n = 1, 3$—theoretically they converge to the ideally band-limited filters for $n \to \infty$.

We have derived the ideal band-limited filters in [45]: For a tilt the filter is a low-pass filter along one frequency axis. For a zoom, the filter is low-pass on both frequency axes. A rotation of the image rotates its Fourier transform by the same rotation angle. A rotation followed by a tilt corresponds to a low-pass filter that was rotated by the rotation angle. As mentioned above, we use spline interpolation (for the rotation) and spline approximation (for the resizing) with splines of degree $n = 0, 1, 3$. Note that for real-time rendering of texture objects lower degree interpolation models are commonly used, as they are computationally less expensive even though the quality is worse. In the Fourier domain, this means that instead an ideal low-pass filter, we get a filter that is attenuated by the Fourier transform of a cardinal spline of degree n. For higher degrees the cardinal spline converts to the sinc function, ergo the ideal low-pass filter. This explains why the identified filters become sharper with increasing interpolation degrees n (Figure 6.3).

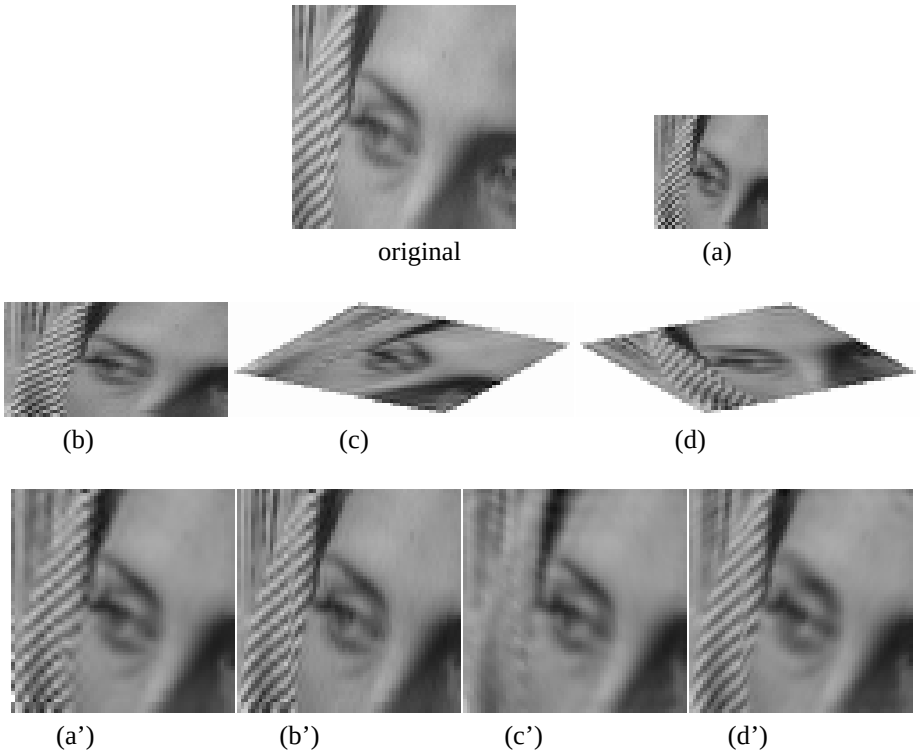


Figure 6.2: Geometric transformed part of "Barbara" image using *cubic* spline approximation ($n = 3$): (a) *zoom* by factor $\frac{1}{2}$. (b) *tilt* by 60° . (c) *rotation* by 30° *and tilt* by 70° . (d) *rotation* by -30° *and tilt* by 70° . The blurred images (a')-(d') are the back transformations of the mapped images (a)-(d) to the original size. Note that the blur depends on the geometric transformation. For example, the diagonal strip pattern is preserved best in (d') whereas it disappears in (c').

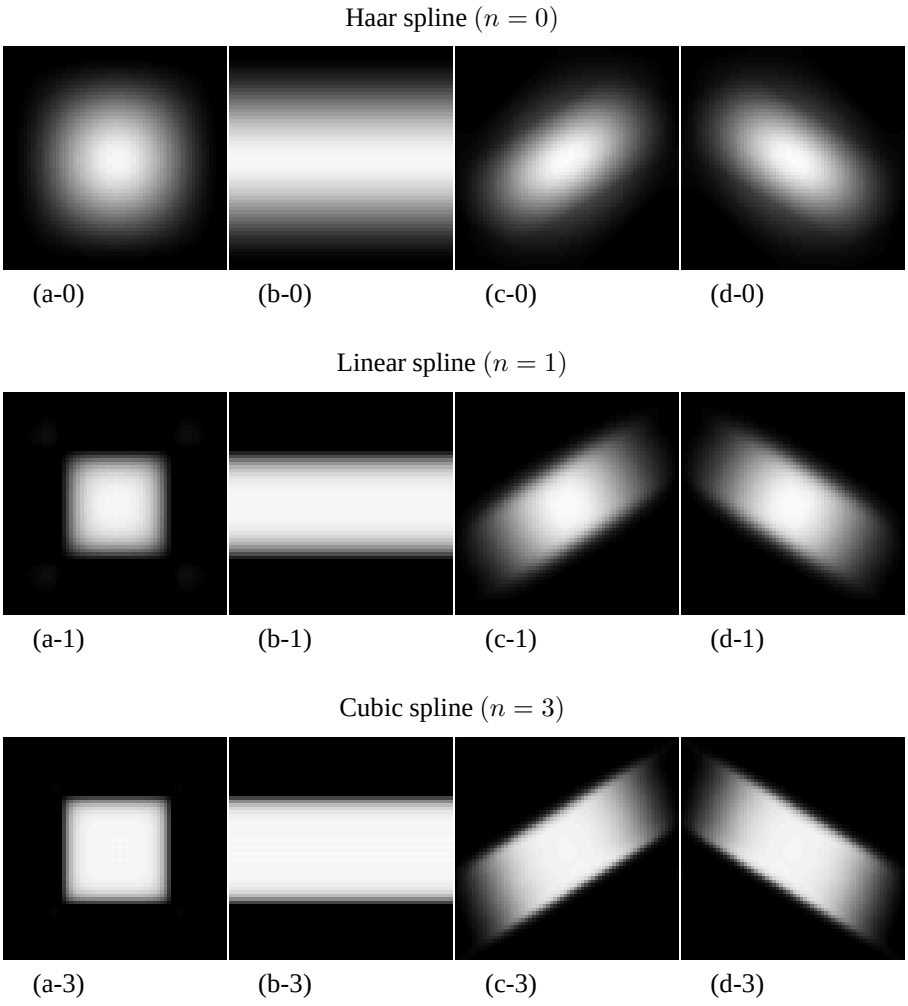
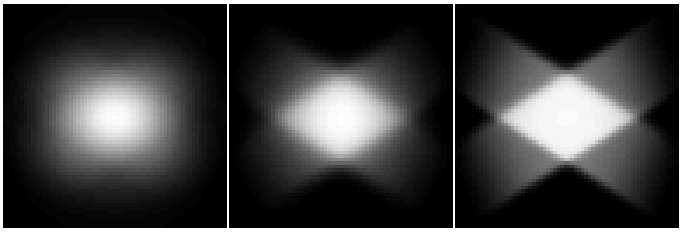


Figure 6.3: Fourier domain analysis of effect of geometric transformations using *Haar spline* ($n = 0$), *linear spline* ($n = 1$) and *cubic spline* ($n = 3$) approximation: (a-n) zoom by factor $\frac{1}{2}$, (b-n) tilt by 60°, (c$-n$) rotation by 30° and tilt by 70°, (d-n) rotation by -30° and tilt by 70° . The higher the degree n, the sharper the filters are.



(a) (b) (c)

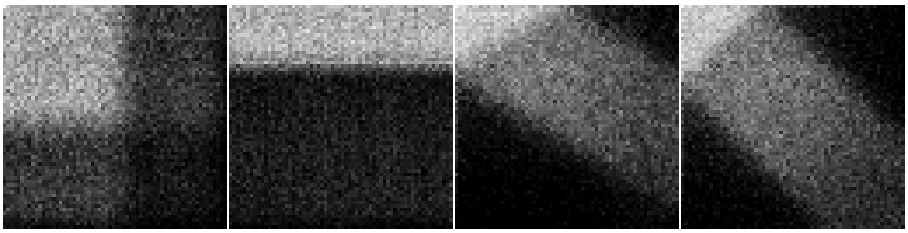
Figure 6.4: Butterfly filters: The Fourier transform has been restricted to an angle range of $0°$ to $90°$ by flipping all quadrants over to one quadrant. This corresponds approximately to the Fourier transform of similar filters in the DCT domain: The image was rotated by 30°and then tilted by 70°. (a) Haar spline ($n = 0$), (b) linear spline ($n = 1$), (c) cubic spline ($n = 3$). The higher the interpolation degree n, the sharper the butterfly filter.

6.3 Filtering in DCT and wavelet domain

Textures (images) are commonly compressed using the DCT or wavelet transform; that is why we try to determine the filters is the two domains. The Fourier transform (FT) can distinguish a rotation range of $180°$, but the DCT and wavelet transforms are limited to a rotation range of $90°$. One can approximately say that the Fourier spectrum of the DCT corresponds to the sum of the four quadrants of the FT flipped over to one quadrant. Therefore, if we analyze in the DCT transform a rotation by an angle α followed by a tilt (as in the Figure 6.3(c-3)), we will get the sum of the FTs of the image rotated at angles α and $-\alpha$ (Figures 6.3(c-3) + 6.3(d-3)). The resulting filters are depicted in Figure 6.4; they resemble the so-called butterfly filter.

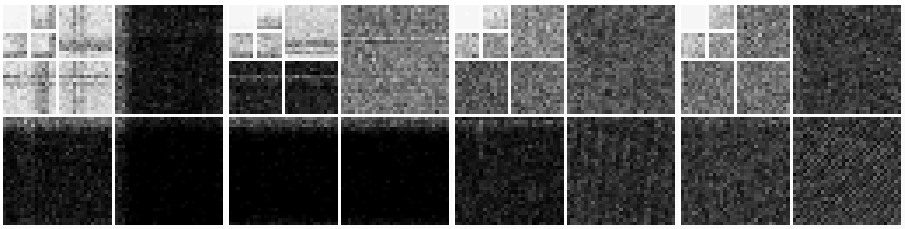
Similarly to the previous section, we identify the filters that simulate the loss of the mapping/unmapping process, but this time we identify them in the DCT and wavelet domains. In Figure 6.5, we apply four geometric transformations and analyze them in the DCT domain. Here, instead of injecting an impulse, we use an ensemble of random images and compare the coefficient amplitudes for each pair of image and back-mapped image: In (a) we shrink the texture by a factor 2, (b) we tilt it by $75°$, (c) we rotate the texture by $30°$ and tilt it by $75°$, (d) we rotate by $45°$ and tilt it by $75°$. The DCT filters (c) and (d) resemble the third quadrant of the butterfly filter in Figure 6.4c.

In Figure 6.6, we analyzed the same transformations in the wavelet domain. We observe that (a) the finest subbands are zero, (b) the two finest subband (LH and HH) in vertical direction are zero, while the amplitude in the finest HL subband is 50%. In (c) we can still see that the average amplitude of the subbands corresponds to the butterfly filter from Figure 6.4c. In (d) we rotate the texture by $45°$ degrees (followed by a tilt); there is no difference in the average amplitudes between the subbands (LH, HL). The coarsest subband HL and LH have an amplitude close to 100% as they correspond to the center of the butterfly filter. The finer the scale, the lower the



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

Figure 6.5: DCT domain analysis of effect of geometric transformations using cubic spline ($n = 3$). The filters are measured using an ensemble of random images. White areas are pass-bands, gray areas attenuate and black areas are stop-bands: (a) zoom by factor $\frac{1}{2}$, (b) tilt on y-axis by 75°(this corresponds to $\frac{1}{4}$ shrink on y-axis), (c) rotation 30° and tilt by 75°, (d) rotation 45° and tilt by 75°.



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

Figure 6.6: Wavelet domain analysis of effect of geometric transformations using cubic spline ($n = 3$). The filters are measured using an ensemble of random images: (a) zoom by factor $\frac{1}{2}$, (b) tilt on y-axis by 75°(this corresponds to $\frac{1}{4}$ shrink on y-axis), (c) rotation 30° and tilt by 75°, (d) rotation 45° and tilt by 75°.

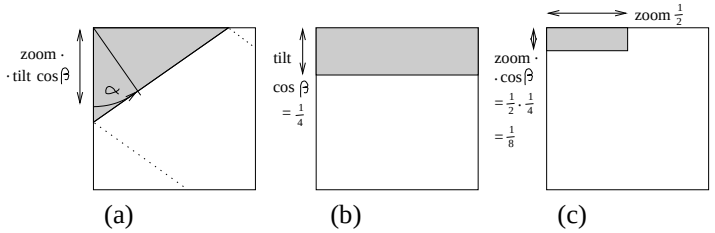


Figure 6.7: Approximation of butterfly filter in DCT domain: (a) tilt angle β and rotation angle α, (b) mask for tilt angle $\beta = 75°$, $\cos\beta = \frac{1}{4}$, (c) mask for zoom $z = \frac{1}{2}$ and tilt angle $\beta = 75°$, $\cos\beta = \frac{1}{4}$.

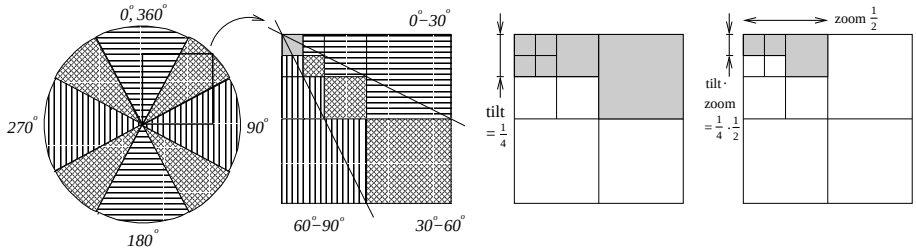


Figure 6.8: Approximation of butterfly filter in wavelet domain: (a) The directional selectivity of the wavelet transform is restricted to three directions. (b) mask for tilt angle $\beta = 75°$, $\cos\beta = \frac{1}{4}$. (c) mask for zoom $z = \frac{1}{2}$ and tilt angle $\beta = 75°$, $\cos\beta = \frac{1}{4}$.

average amplitude gets.

We can approximately realize the butterfly filters in the DCT and wavelet domain. Their shapes are determined by three parameters: the rotation angle α, the tilt angle β and the zoom factor z. See Figure 6.7 for the approximation in the DCT domain and Figure 6.8 for the approximation in the wavelet domain. For more details on mask models, see [37].

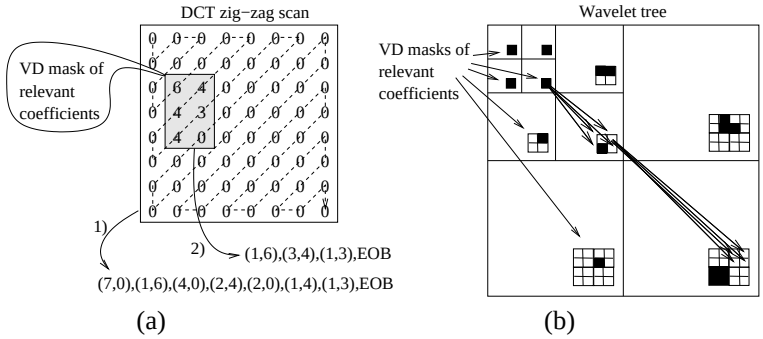


Figure 6.9: Coding of view-dependent texture and using VD masks to selected coefficients: (a) in DCT domain (8×8 block) using Zig-zag scan and runlength coding, (b) in wavelet domain.

6.4 View-dependent compression

6.4.1 View-dependent masking and coding

In the Section 6.3, we have approximated the filters in the DCT domain and the wavelet domain to simulate the losses of the geometric image transformations. We simplify these filters to a *decision mask*, or also-called *shrinking mask*, which indicates whether a coefficient is relevant or not. Therefore we use the words *masking* or *shrinking* for the view-dependent filtering of the coefficients. The coefficients remaining after applying the shrinking mask are necessary for the mapped texture. The mapped view-dependent texture can almost not be distinguished from the mapped original texture. If we code the filtered texture, we get a view-dependent compressed texture.

Figure 6.9a shows the coding of a view-dependent simplified 8x8 DCT block. The view-dependent (VD) mask in gray denotes the selected coefficients. There are two possibilities to run length code the DCT coefficients similar to JPEG (see Figure 6.9a): (1) the whole DCT block is coded. Due to the numerous zeros around the mask, the code becomes rather long, but the code is bitstream compliant with JPEG or MPEG. (2) only the coefficients belonging to the mask are coded. The mask must be either transmitted to the decoder or determined in the decoder by the view-dependent criteria. The code is more compact, but the coding process is more complex.

Figure 6.9b shows the view-dependent coding with wavelets. The mask is spread over all subbands and is locally interconnected through the wavelet tree hierarchy. For more details on coding see [37].

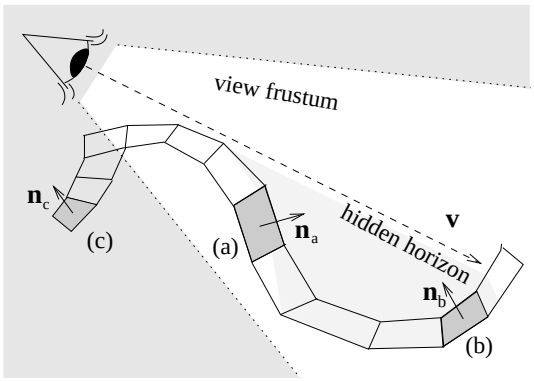


Figure 6.10: Spatial view-dependent criteria: (a) backface invisible as it points in viewing direction $\langle n_1, v \rangle > 0$, (b) polygon hidden under hidden horizon, not visible even though $\langle n_1, v \rangle < 0$, (c) polygon outside the view frustum (field-of-view).

6.4.2 Spatial criteria

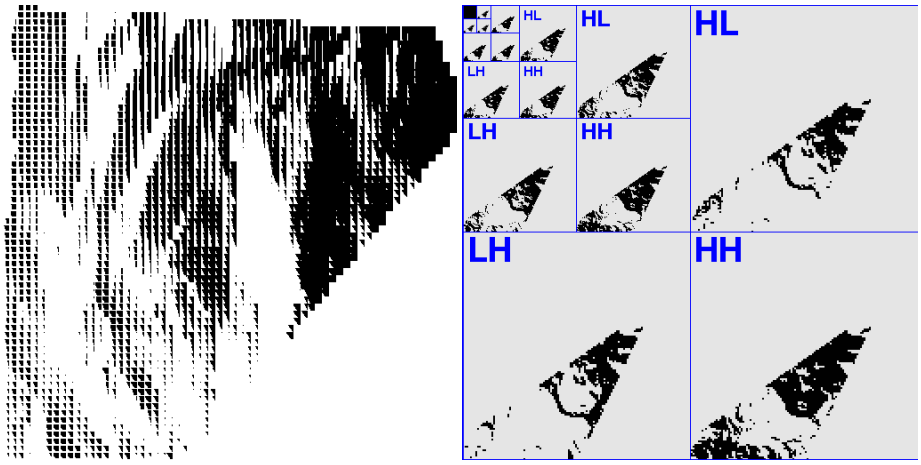
There are additional criteria in the spatial domain which determine the visible parts of a view-dependent texture. Let us assume the texture is mapped on a regular polygonal mesh. For each polygon we can analyze the part of the texture that is mapped to it. There are three geometric criteria which allow to neglect texture parts: First, the texture part is invisible because it is on the backface of a polygon pointing away from the viewer, this is also called *backface culling* (see Figure 6.10a). Second, the texture part is on a polygon that is hidden behind another one; for a closed surface we can apply a *hidden horizon* algorithm (see Figure 6.10b) to determine the hidden polygons. Third, the textured polygon lies outside the *view frustum* (see Figure 6.10c).

6.4.3 Shrinking masks

We create a shrinking mask (image) in which each black point corresponds to a coded coefficient and each white point to a view-dependent negligible coefficient. The mask visualizes the effects of the above described criteria. Note that the shrinking mask need not be binary (as we have implemented here), but weighted so that view-dependent less important coefficients are transmitted later or with less bit precision.

Intra masks

You can compare in Figure 6.11 the shrinking intra masks C_k for the DCT and the wavelet domain. For DCT the spatial and frequency criteria are applied on each 8x8 DCT block independently. For wavelets, we do the spatial analysis on the finest scales, and propagate the mask to the



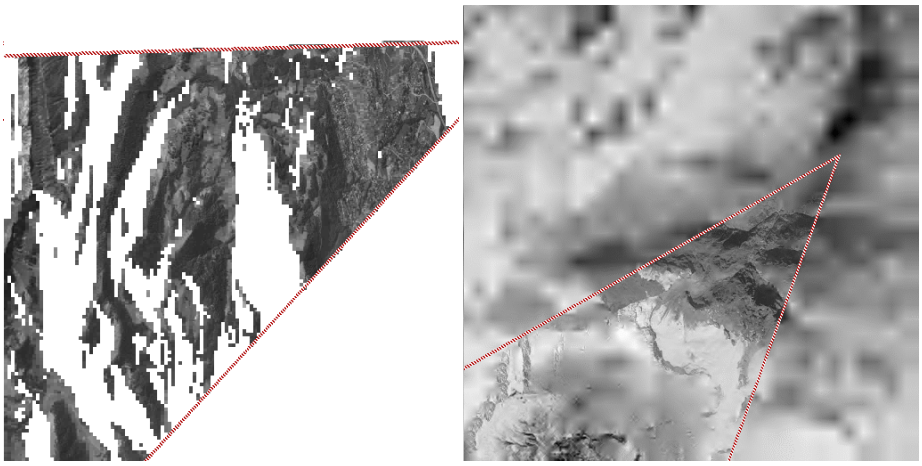
(a) (b)

Figure 6.11: View-dependent shrinking INTRA masks: (a) DCT domain with 8x8 blocks, (b) wavelet decomposition with 5 scaling levels. In the significance mask, each black point corresponds to a significant wavelet coefficient; each white point indicates a suppressed wavelet coefficient. Here, the implemented VD criteria are field-of-view, backface culling and the frequency domain criteria (zoom, tilt and rotation).

next coarser scales in the wavelet tree hierarchy (see as well Figure 6.9). If only one coefficient is relevant then the coefficient from the superior scale is relevant too. Figure 6.12 shows the result of applying the shrink mask to the texture in the DCT and wavelet domain. In Figure 6.13 the landscapes are rendered with these view-dependent simplified texture.

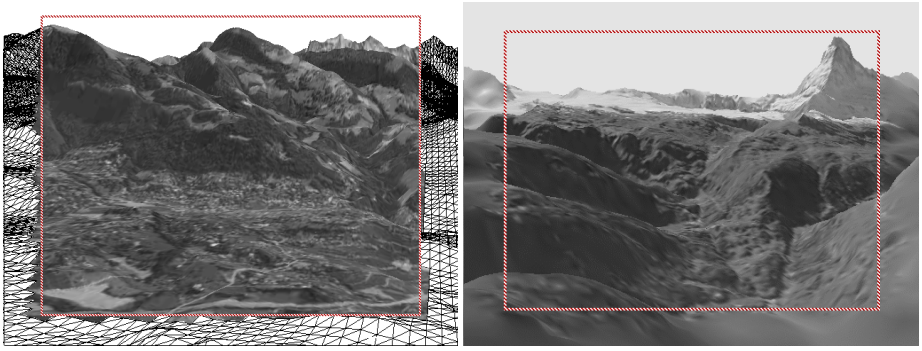
Inter masks

So far we have considered only mappings for a fixed viewpoint, but in an animated sequence the view frustum changes successively. This means the view-dependent shrinking mask changes as well. New coefficients are relevant, whereas older become superfluous. In a communications scheme where the texture is streamed according to the current view frustum, we code and send the new coefficients N_k. In order to determine N_k, we have to memorize the coefficients that have already been transmitted, S_{k-1} , and compare them to the coefficients C_k necessary at the current view frustum (or view position). The information as to which coefficients are necessary at the view position of the animation step k is described by the three binary masks: C_k denotes the current necessary coefficients (intra mask), N_k denotes the new coefficients (inter mask), and S_k denotes all transmitted coefficients. The three masks are related by



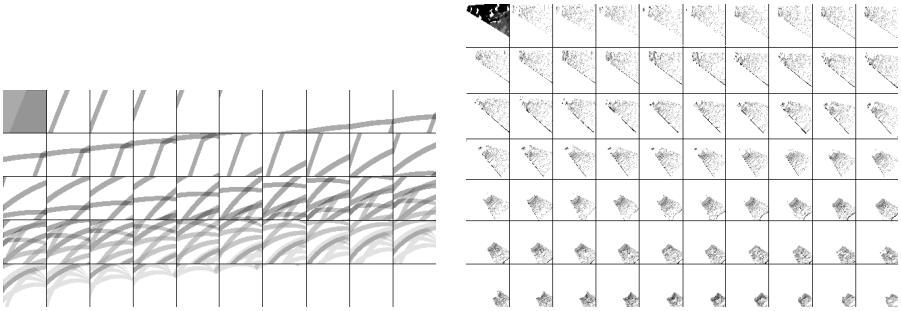
(a) (b)

Figure 6.12: View-dependent compressed texture with significance masks from Figure 6.11: (a) DCT domain, (b) wavelet domain. The view frustum is depicted by the red bold lines.



(a) (b)

Figure 6.13: Scene rendered with view-dependent compressed texture from Figure 6.12: (a) using DCT-based compressed texture, (b) using a texture compressed in wavelet domain. The view frustum is depicted by the red bold frame.



(a) (b)

Figure 6.14: Streaming of texture coefficients for sequence of images: The first frame (intra coded) is in the upper left corner, the last frame is in the lower right corner. Each black point indicates a transmitted DCT coefficient. (a) Zoom on tilted plane. Each strip moving across the texture, as time increases, contains DCT coefficients of the same frequency subband. (b) Fly-over landscape Blonay. One can see how the viewer position moves. The pattern looks like the *light cone* (view frustum) of a torch light moving across the landscape; the texture parts that are revealed by the *light* in each frame are transmitted.

$$S_{k-1} = \bigcup_{i=1}^{k-1} C_i = \bigcup_{i=1}^{k-1} N_i,$$

and

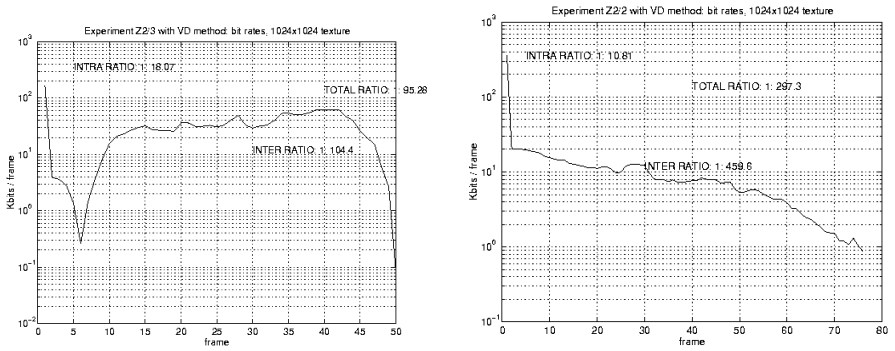
$$N_k = C_k \bigcap (\neg S_{k-1}) .$$

In the following, experimental section, we show the intra mask (first frame) and inter masks (following frames) for several sequences (see Figure 6.14).

6.5 Experimental results

Here, we show the intra and the inter masks for two sequences: A zoom on a tilting plane and a fly-over a landscape. These sequences have been produced during the core experiments of the MPEG-4 SNHC[1] standardization process [43, 41].

[1]MPEG stands for Motion Picture Expert Group. SNHC stands for Synthetic Natural Hybrid Coding.



(a) (b)

Figure 6.15: Bitrates for sequence of view-dependent streamed images: (a) Zoom on tilted plane. (b) Fly-over landscape Blonay.

6.5.1 Zoom on plane

This sequence starts with a square texture-mapped plane that is rotated by $\alpha = 30°$, tilted by $\beta = 20°$ and zoomed at far distance $z = \frac{1}{4}$. The sequence zooms onto the plane until all details are visible. Each strip in the sequence of inter masks corresponds to DCT coefficients of the same frequency subband (see Figure 6.14a). The bitrate per frame is quite steady and low (see Figure 6.15a).

6.5.2 Fly-over sequence

In this sequence we simulate a flight over a texture landscape, like in a flight simulation. It is interesting to observe in Figure 6.14b how the DCT coefficients are streamed as the view frustum glides over the landscape. The pattern looks like a torch light cone; when ever the light shines on new texture part (spatial criteria) then these parts are transmitted. The bitrates are given in Figure 6.15b. The comparison of the compression ratios of both sequences are given in Table 6.1. We have performed further experiments with wavelet-coded textures in Chapter 7.

6.5.3 Comparison with still image coding

The reason that view-dependent texture streaming works well for a moving sequence is that most of the texture data is not needed for the first frame, but only gradually in later frames—if at all. Another advantage is that the texture data necessary for the first frame arrives much faster at the receiver compared to the time it would take to receive the whole texture. If our view-dependent coding algorithm is good, then we would expect only little extra costs to stream all of the texture data coded view-dependently during a sequence (e.g. using MPEG-4) compared

Sequence	Flat zoom Z2/3	Fly over Z2/2
Kbits original frame	4866 kbits	3932 kbits
Kbits in first frame	170 kbits	333 kbits
INTRA compression ratio	1:28	1:12
Kbits/s @10 frames/s	294 kbits	75 kbits
INTER compression ratio	1:165	1:524
Average Kbits per frame	21.7 kbits	11.8 kbits
MEAN compression ratio	1:224	1:332

Table 6.1: Comparison of compression ratio of sequences: Intra compression is given for the first frame, the inter ratio for the following frame and the mean for all frames.

Sequence	Flat zoom Z2/3	Fly over Z2/2
VD transmitted	1.1 Mbits	0.9 Mbits
JPEG 75%	2.4 Mbits	2.4 Mbits
RGB raw data	24 Mbits	24 Mbits

Table 6.2: Comparison of total number of transmitted bits of view-dependent compression method (now in MPEG-4) and of a standard still image compression method (JPEG).

to a single transmission of the whole texture at the beginning (e.g. using JPEG). We compare in Table 6.2 the number of total transmitted bits of our method with the JPEG still image coding method. Note that in order to reduce the bitrate even further, one might exploit the redundancy between the already transmitted coefficients S_{k-1} and the currently transmitted coefficients N_k, as the receiver also knows the coefficients in S_{k-1}.

6.6 Conclusion

We analyzed the information loss of typical texture mappings in the frequency domain to derive mapping-dependent filters that simulate this loss. We approximate the filters in the DCT and wavelet domain to generate view-dependent compressed texture. Together with the spatial criteria, we can determine the texture coefficients that are necessary for the mapped texture. In an animated sequence, new coefficients become relevant as the view position (view frustum) moves in each frame; the new coefficients are streamed to the receiver. View-dependent compression allows to stream texture data at very small bitrates. The DCT-based view-dependent technique has been successfully introduced in the MPEG-4 SNHC standard.

6.7 Major achievements

- Identification of the view-dependent filters in the FT, DCT and wavelet domains taking into account the degree of the interpolation models.
- Development of the view-dependent texture compression method which is now part of the MPEG-4 standard.

Chapter 7

Joint Mesh and Texture Simplification

"Everything should be as simple as it is, but not simpler."
—ALBERT EINSTEIN

7.1 Introduction

In many computer graphic applications, a texture is mapped onto a meshed surface. Under complexity constraints, both texture and mesh need to be approximated. In this chapter[1], we present a framework for the joint simplification of texture and mesh with respect to an error measure in screen space. In order to achieve optimal operating points, we choose two efficient simplification algorithms: Optimal multiresolution mesh simplification based on a quadtree structure and multiresolution view-dependent texture compression based on wavelets (c.f. Chapter 6). Together, these two algorithms are used for joint simplification using an efficient heuristic based on marginal analysis. As an application example, we study the mapping of aerial orthophotographs onto a terrain model of the swiss alps, and show that the resolution of the terrain and the aerial photograph is efficiently traded-off.

[1]This work has been done in collaboration with Laurent Balmelli (balmelli@us.ibm.com).

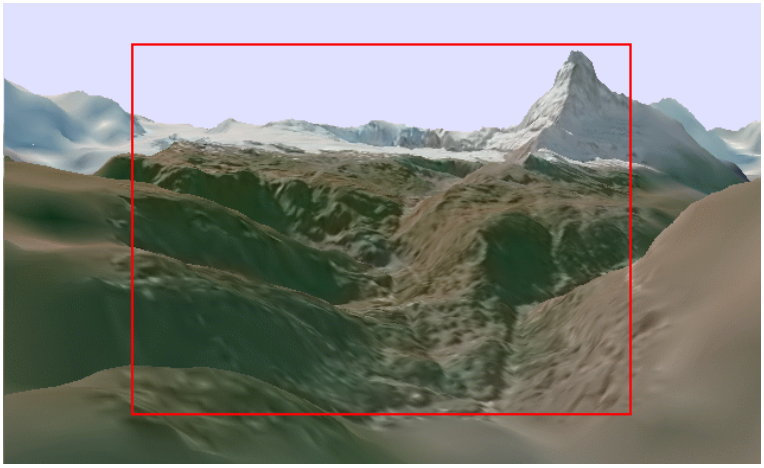


Figure 7.1: View-dependent compressed texture of Figure 7.6c) mapped on simplified mesh of Figure 7.2) seen from the viewer position. The field-of-view is depicted by the box. Outside the box the texture is blurred and the mesh is simplified.

7.2 Problem formulation

In this chapter, we address a simple yet basic question of computer graphics: given a surface specified by a mesh and a texture to be mapped on that mesh, what is the resolution required for the mesh and the texture to achieve the best combined display quality? This question is relevant every time we are resource-constrained, and we need to simplify meshes and/or textures mapped on the meshes. The resource constraint might be computational (complexity of rendering meshes and mapping textures), storage space or bandwidth (bitrate needed to represent, transmit or update a complex model and its associated texture information) or both. In such cases, it is critical to choose the correct resolution or level of detail for both the mesh and the texture mapped on it. If more resource become available (e.g. in progressive downloads over the Internet), it is important to know if the mesh or the texture needs to be refined first.

To make our discussion more concrete, consider a specific case where the interplay of texture and mesh resolution is particularly intuitive, namely very large terrain models with aerial photographs to be mapped on the terrain models. Consider two extreme cases. First, consider mapping a very fine texture (a very detailed aerial photograph) onto a very coarse terrain model. This gives an unsatisfactory result: While many details are available, the surface model is too simplistic compared to the texture resolution. At the other extreme, take a very detailed terrain model, but with an overly coarse texture (e.g. only one color per surface element). The mesh is now very complex, but the texture is trivial, thus not reaching a good trade-off. The above two extreme cases hint at a better solution, where the correct trade-off between mesh and tex-

ture resolution is found. Let us now formalize the problem. Given is a surface model at full resolution M_0, and a family of simplified models $\{M_i\}_{i=1\ldots N}$. We associate a cost function $C_M(M_j)$ to each model which reflects its complexity (number of bits needed to represent the model, or computational cost for rendering the model). Likewise, given a full resolution texture T_0, and a family of simplified textures $\{T_i\}_{i=1\ldots M}$, we associate a cost function $C_T(T_j)$ to each texture, which measures its complexity (again, bitrate or computational cost of rendering). Finally, for a given pair (M_i, T_j), we define a distortion measure $D(M_i, T_j)$ which evaluates the error in screen space between the image generated by the full resolution version (M_0, T_0) and the approximated version (M_i, T_j). Calling the image on the screen $I(M_i, T_j)$ to reflect its dependence on the underlying model and texture, a possible distortion[2] is the l_2-norm, or

$$D(M_i, T_j) = \|I(M_0, T_0) - I(M_i, T_j)\|_{l_2} \tag{7.1}$$

Note that $I(M_i, T_j)$ depends on the rendering process (e.g. lighting), but we assume the same rendering process for the various approximate images.

The statement of the problem is now the following: Given a total budget for the combined complexity of the mesh and texture:

$$C = C_M + C_T \tag{7.2}$$

find the pair (M_i, T_j) that minimizes $D(M_i, T_j), i \in [0 \ldots N], j \in [0 \ldots M]$:

$$D_{\text{min}} = \min_{i,j}(D(M_i, T_j)) \tag{7.3}$$

under the constraint that

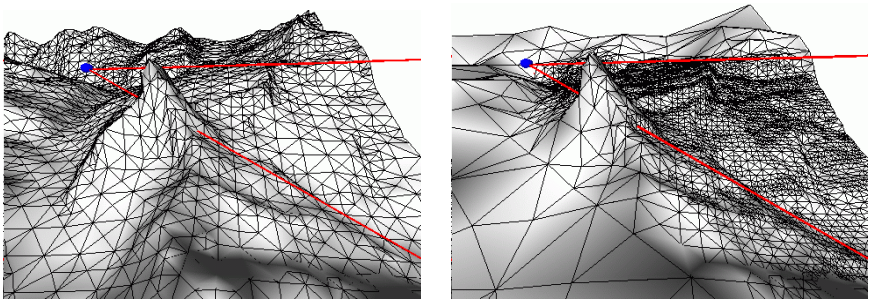
$$C_M(M_i) + C_T(T_j) \leq C. \tag{7.4}$$

Such a formulation is very reminiscent of compression systems, and indeed, when the cost is the required bitrate for model and texture, the above gives a best compression scheme in a operational rate-distortion sense.

The search for the minimum error in eq.(7.3) can be done by exhaustive search, but this is clearly impractical. Also, generating a set of models $\{M_i\}$ and a set of textures $\{T_i\}$ leading to various costs C_M and C_T can be very complex unless some structure is imposed. Thus it is critical to come up with computationally efficient methods for searching a wide variety of possible approximations. In the following, we focus our attention on the case where the cost is the bitrate required for representing the model and texture. This is a well defined cost, and it is very relevant for applications where storage and/or transmission is involved. Also, we restrict our attention to successive refinement or progressive models and textures. While this is a restricted class, it is a very important one for cases where interactivity and communication is involved (e.g. progressive downloads).

We now briefly overview our approach to progressive meshes and textures, to their joint simplification, and discuss the data set underlying our experiments.

[2]The problem of finding a distortion measure that reflects perceptual quality is still an open problem, especially for rendered models, see [82].



(a) (b)

Figure 7.2: Comparison of meshes of same complexity (courtesy of Laurent Balmelli): a) Global simplified mesh and b) view-dependent simplified mesh. Both meshes have ∼8K triangles. The original model has 131K triangles. The global simplification algorithm minimizes the overall approximation error of the mesh a). The view-dependent simplification algorithm minimizes the displacement of vertices projected in the screen space b). The viewer position is depicted by a disc and the view frustum by two lines. The triangles visible in the view frustum are more detailed in b) than in a).

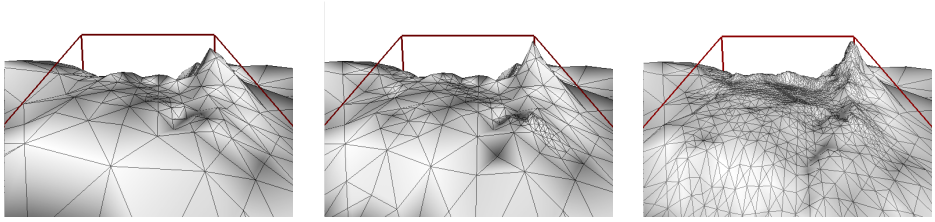
7.3 Solution method

We review two multiresolution approaches to mesh and texture simplification that we used in our joint simplification algorithm. Note that our joint simplification algorithm works with other simplification methods as well.

7.3.1 Multiresolution meshes

To construct a set of meshes of varying resolution, we use the algorithm described in [3]. This algorithm uses the same type of mesh connectivity as [66, 17], but it uses a vertex decimation approach to build the set of meshes rather than vertex insertion as in [66, 17] (decimation is also used in [17], but within a restricted setting). Approaches based on decimation yield better approximation quality in general. Moreover, [3] produces quasi-optimal decimation in a rate-distortion framework when using the l_2-norm as distortion functional. Therefore, we set a constraint in cost for each mesh and compute a series of approximations. For efficiency reason, the implementation is based on the quadtree described in [4]. Its traversal property allows to obtain fast processing of large datasets.

Figure 7.2 shows the difference between a view-independent (part a) and a view-dependent mesh simplification (part b). The view for which the mesh was simplified is determined by the viewer's position (depicted by a disc) and the view frustum (depicted by two lines). They are the same as in Figure 7.1, but here the scene is viewed from the side. Different mesh resolutions are



(a) (b) (c)

Figure 7.3: Series of mesh approximations: a) using 500, b) 1000, and c) 2000 triangles.

shown in Figure 7.3.

The error measure for the vertices is similar to [17]. For each vertex, we measure a displacement in the screen between its original position (obtained by projecting once the full resolution mesh) and its counterpart in the approximated mesh. Since an error is incurred only when a vertex is decimated, its position in the approximated mesh is obtained by interpolation using the remaining vertices.

In our experiments, multiresolution meshes were used, and the view dependency was taken into account by only counting mesh elements that are visible in the complexity measure. This is an approximation to a true view-dependent complexity of the mesh [35]. A complete, view-dependent mesh coder is beyond the scope of this work.

7.3.2 Multiresolution textures

Inspired from work in image compression using wavelets, we developed a multiresolution texture mapping based on wavelet methods [38]. That is, the texture is represented by its wavelet coefficients. These are pruned depending on the view point and the desired resolution. This is similar to the work in [43] about view-dependent compressed images, which predicts the content of projected textures in the frequency domain of the original texture. Given a full resolution view-dependent texture T_0 viewed at a distance d_0, a view-dependent multiresolution pyramid of simplified textures $\{T_i\}_{i=1\ldots M}$ is defined by exponentially increasing the distance $d_i = d_0 \cdot 2^{i/k}$, where k is the number of steps to double the resolution in the pyramid. A scalable bitstream for the set $\{T_i\}_{i=0\ldots M}$ is achieved by coding first the coefficients of the lowest resolution T_M, then adding the coefficients from $T_{M-1}, T_{M-2}, \ldots T_0$ until the full texture T_0 is reached. A decoder can stop the bitstream at any point to get a view-dependent texture at an intermediate resolution. For our experiments, we use a view-dependent wavelet-based coder [38, 43] with a scalability over 5 levels (Figure 7.6).

7.3.3 Efficient joint mesh and texture simplification

As outlined in the problem formulation, a brute force method consists in minimizing the screen distortion over all possible pairs of mesh and texture resolutions satisfying the bitrate constraint. Instead, we propose a method based on marginal analysis, a technique used in optimizing compression algorithms [110]. Using this method, a necessary condition for a pair (M_i, T_j) to be a candidate achieving a good trade-off between mesh and texture resolution is that removing a certain bit budget from either mesh or texture will lead to a similar increase in distortion. The intuition behind the result is simple: if it were not so, some of the bits could be moved from mesh to texture (or vice versa) in order to reduce the distortion. While this result is exact only when assuming independence of the mesh and texture (which is an approximation), the heuristic based on it is quite competitive. Note that it can be used either as a greedy refinement or a greedy pruning technique.

7.3.4 Data set underlying the experiments

Terrain models are very large scale meshes that require efficient simplification methods. Often, aerial photographs are available for texture mapping, and these photographs can have very high resolution, thus requiring approximation. An interesting terrain model is the one of the Alps; in particular, the region around the Matterhorn. We thus use digital terrain models of that geographical region as well as high resolution orthophotographs for our set of experiments.

7.4 Joint mesh-texture simplification

Both meshes and texture are usually treated separately in the current literature. The originality of our work is to take a joint approach to simplify a mesh and its texture: joint texture and mesh coding. In the previous section, we outlined methods to reduce/increase the resolution of textures mapped onto surfaces, as well as reduce/increase the resolution of surfaces defined by meshes. In both cases, the algorithms depend on the representation chosen for the texture and mesh, respectively. For textures, wavelet bases were used, and for meshes, an efficient quadtree structure was involved. In all cases, a smooth tradeoff between the complexity of the representation (as measured in bitrate) and quality of the representation (as measured by an l_2 error in world or screen space) is achieved.

Clearly, when mapping textures onto surfaces, there is a non-trivial interaction between the respective resolution of texture and mesh. For example, a high resolution texture will look pleasing visually (giving the “illusion” of high resolution) but an underlying coarse surface will lead to a large objective error (since the high resolution texture is not placed at the correct location).

The complex dependence between these effects would require an exhaustive search of the optimal trade-off. Instead, we propose to use an efficient heuristics that has proven to be efficient in other contexts where exhaustive search is too expensive [110]. The method, called marginal analysis, was presented intuitively in the introduction. It searches greedily for the best tradeoff

while reducing or increasing the resolution of both mesh and texture. In the pruning mode, a full resolution model, with full resolution texture mapped onto it, is successively coarsened. At each step, given a target complexity reduction of B bits, the best of reducing the mesh or the texture description by B bits is chosen (or possibly, reducing both by $B/2$). The degradation is simply computed by comparing the signal-to-noise ratio of the pruned versions with respect to the original, either in world space or in screen space (the latter being preferred). While this is a greedy approach, it performs quite close to an optimal exhaustive search in our experiments. Optimality would only be reached in general when the two entities are independent, in which case this becomes standard Lagrangian optimization [61]. In the tree growing mode, the converse operation is performed in order to find the better of increasing the resolution of texture or mesh. Note that tree growing is doubly greedy, and would not even perform optimally in the case of independence, and thus the pruning method is preferred.

The marginal analysis algorithm in tree pruning mode takes as input a given multiresolution pyramid of meshes M_i and texture T_j. The algorithm starts with the coarsest pair (M_N, T_M). It constructs a series of refined pairs (M_i, T_j) until the full resolution (M_0, T_0) is reached. The algorithm is described as follows:

Algorithm of marginal analysis for JMTS

▷ **Input**: Pairs (M_i, T_j) with **i** $= \{0, \ldots, N\}$ and **j** $= \{0, \ldots, M\}$

◁ **Output**: a set of progressive pairs (M_l, T_m), with $l \in$ **i** and $m \in$ **j**

1 **FOR (** $i = N, j = M$; $i \geq 0$ **AND** $j \geq 0$; **)**

2 **IF (** $D(M_{i-1}, T_j) \leq D(M_i, T_{j-1})$ **)**

3 { store the pair (M_{i-1}, T_j); $i = i - 1$; }

4 **ELSE** { store the pair (M_i, T_{j-1}); $j = j - 1$; }

7.5 Experimental results

In this section, we present experimental results on large terrain data and aerial photographs. We compare various rate configurations and the performance in both complexity and distortion are analyzed. The terrain model consists of a regular mesh of 256 by 256 vertices and the texture has 1024 by 1024 pixels resolution. To evaluate the quality of the rendered image, we measure the peak signal-to-noise ratio (PSNR) in the screen. The PSNR is a well established objective image error measure, albeit it does not reflect the subjective image quality perceived by a human

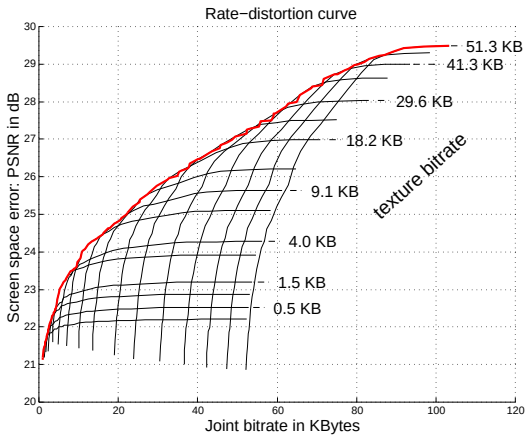


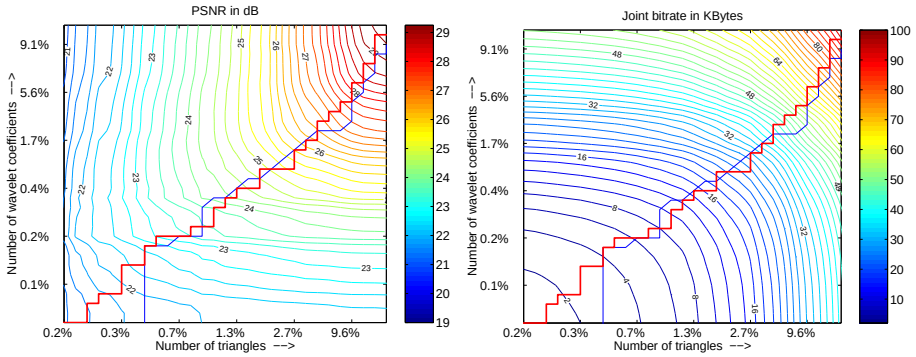
Figure 7.4: Experience 1 (exhaustive search): The bold envelope shows the optimal rate-distortion curve for joint mesh and texture simplification found by exhaustive search. The thin lines are lines of constant texture rate.

viewer [82]. However, given another error measure that decreases when the bitrate increases, the marginal analysis still works. Thus, more sophisticated perceptually based error models could be used as well.

We first conduct an exhaustive search experiment for a set of pairs (M_i, T_i) of meshes and textures optimized for a specific viewpoint. The viewer location, corresponding to the one in Figure 7.1, is such that only 10% of the terrain model is visible. To generate the approximation M_i and T_j, we increase exponentially the rate of the mesh and the texture from 0.1% to 10% of the available vertices and wavelet coefficients. In total, 32 textures and 60 meshes are generated giving a total of 1920 pairs. Figure 7.4 shows the result of the experiment: For each pair $(M_i, T_j)_{i=0\ldots59, j=0\ldots31}$, the PSNR with respect to the full resolution pair (M_0, T_0) is evaluated in the screen space. Each thin line in the graph corresponds to a fixed texture rate. The rate of the mesh increases on the x-axis, while the y-axis shows the resulting PSNR. The optimal path is the outer hull of all curves and is depicted by the bold line. Following this path gives the highest PSNR for a given bitrate.

The second experience consists in applying the marginal analysis and comparing the obtained path to the optimal path found by exhaustive search in Figure 7.4. Figure 7.5 shows, for all combinations of the 60 meshes and 32 textures, their PSNR and their joint rate, respectively. In both figures, the mesh complexity increases along the x axis, whereas the texture complexity increases along the y axis. The bold line shows the optimal path and the thin line shows the path found by marginal analysis.

Now, we visualize the solution space and compare rendered mesh-texture pairs (M_i, T_j) in the Figures 7.7-7.9. In Figure 7.7 we show rendered pairs (M_i, T_j) that have the same joint rate,



(a) (b)

Figure 7.5: Experience 2 (marginal analysis): (a) *Iso-lines of the PSNRs* of all mesh and texture combinations. The color of the iso-lines indicate the PSNR. (b) *Iso-lines of the joint rates* of all mesh and texture combinations. The color of the iso-lines indicate the joint bitrate. The thick (red) line shows the optimal path (in rate-distortion sense) found by exhaustive search. The thin (blue) line is the path found by marginal analysis. It is a good approximation of the optimal path.

but the quality of the images differs significantly. In Figure 7.8 we show rendered pairs (M_i, T_j) that have the same PSNR although the joint rate varies substantially. This shows that there is a trade-off between the resolutions of mesh and texture. In Figure 7.9 we present the optimal solution that follows the path found by marginal analysis. This path has a length of 84 frames. To find it, twice as much frames had to be rendered and their screen space error evaluated. Figure 7.5 shows that the path is close to the optimal path found by exhaustive search. Exhaustive search required to evaluate about 2000 mesh-texture combinations, whereas marginal analysis needed only 168 evaluations. We can conclude that marginal analysis is an efficient tool to select the pairs (M_i, T_i) that are close to optimal.

7.6 Conclusion and future work

The framework we set up to jointly simplify mesh and textures can be extended to other problems as well. For example, in all progressive transmission cases where refinements have to be sent, it is possible to use the marginal analysis idea to decide which information has to be refined first; for example, this may be applied for dealing with illumination changes (see Figure 7.12). Clearly, our marginal analysis algorithm is an efficient way to decide which refinements to send first. In conclusion, we have presented an efficient framework to solve the problem of jointly refining textures and meshes. The ingredients that were used include efficient multiresolution meshes on quadtrees and multiresolution texture mapping based on wavelets. Together with an algorithm

based on marginal analysis to search for the best refinement (tree growing) or simplification (tree pruning) of the combined texture and mesh (with respect to an objective screen space error measure, the l_2 norm), we obtained a new combined algorithm. In experimental results, the proposed algorithm performs very close to the optimal exhaustive method, and this at a fraction of the computational cost.

As future work, the view-dependent mesh compression has to be included in our scheme, in order to get a full compression method with joint-texture approximation. Possible mesh coding are discussed in [36, 60] and are a topic of current research.

7.7 Main achievements

The contributions of this chapter are the following ones: We introduce a framework to solve the joint mesh-texture simplification. We explained how to create view-dependent multiresolution image pyramids. We introduce an efficient heuristic to find a set of good pairs (M_i, T_j) of decreasing resolution. The method is based on marginal analysis. Extensive experiments show the importance of finding good trade-offs between mesh and texture resolution.

Chapter Appendix

7.A Plates

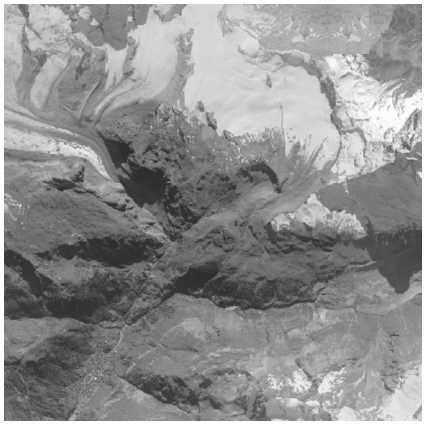
In the first plate (Figure 7.6), we demonstrate how view-dependent wavelet coding is applied to the orthophoto.

In the next plates (Figures 7.7-7.9), we explore the space of solutions found by exhaustive search and marginal analysis. The plates depicts the images for constant joint rate (Figure 7.7), the images for constant measured image quality (Figure 7.8), and the images found on the optimal path (Figure 7.9).

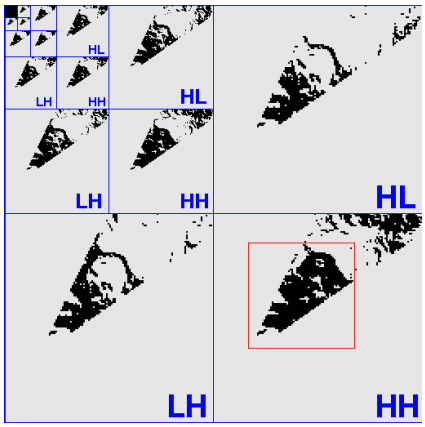
We put side by side a real picture of the Matterhorn and two computer generated images (one with the orthophoto and the other one with a synthetic texture). They are not easy to distinguish at first sight; guess yourself in Figure 7.10.

The orthophoto provided by Swissphoto AG is aligned with the topographical map (on which the DHM25 is based). The perspective correction did not take into account the distortion due to the height model (see Figure 7.11). We corrected the orthophoto to generate a realistic textured 3D model.

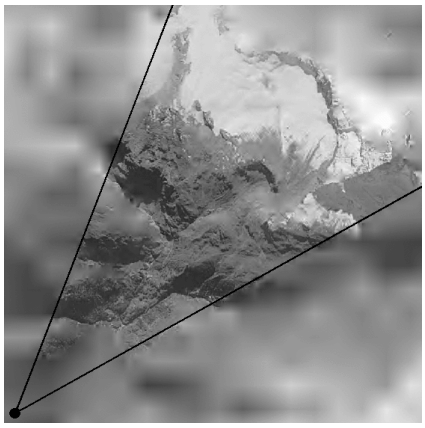
In Figure 7.12 we show pictures of the Matterhorn taken every hours with different illumination; watch the shadows move. The orthophoto in Figure 7.2a was taken around midday.



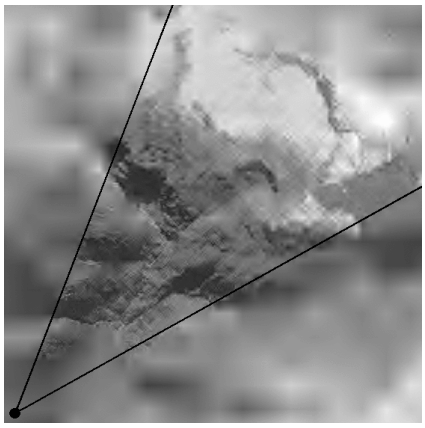
a) Texture detail (560 by 560 pixels) of the original texture (1024 by 1024 pixels). We zoomed on the part of the texture—depicted by the frame in b)—lying in the field-of-view to show in detail the effects of view-dependent simplification.



b) View-dependent wavelet decomposition with 5 levels: Each black point denotes a coded wavelet coefficient; each white point denotes a suppressed wavelet coefficient. The frame—shown only in finest HH subband—covers the area of the texture detail seen in a), c) and d).

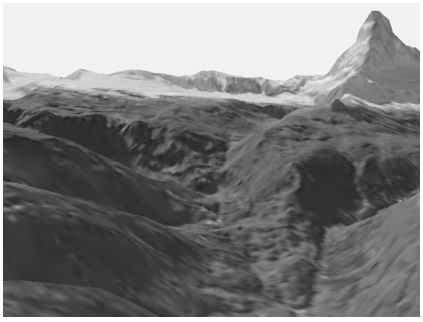


c) Texture detail coded only with the relevant wavelet coefficients b). The viewer position is depicted by a disc and the field-of-view by two lines. This view-dependent texture is optimized for the view seen in Figure 7.1.

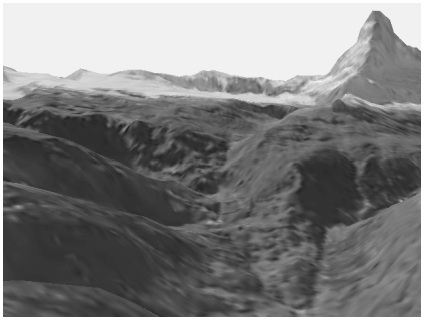


d) Texture detail coded with less wavelet coefficients than in c). We construct a multiresolution pyramid of view-dependent textures by progressively reducing the number of coded wavelet coefficients.

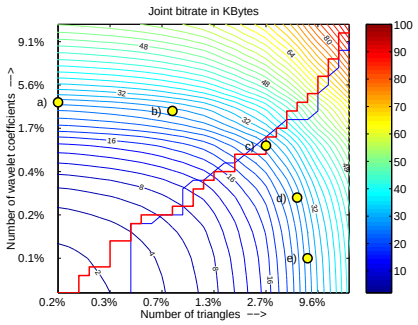
Figure 7.6: The principle of view-dependent texture coding using wavelets decomposition.



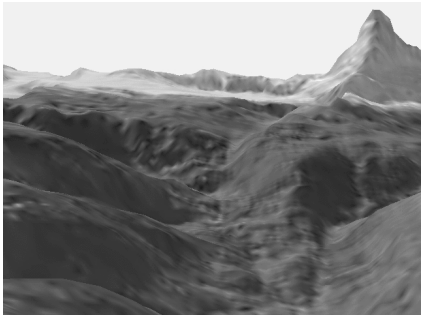
a) Bitrate=26.8KB, PSNR=21.1dB, texture=3.4%, mesh=0.2%



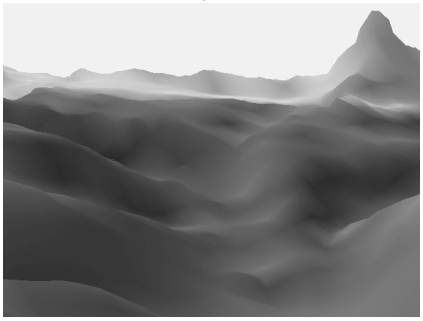
b) Bitrate=27.2KB, PSNR=24.0dB, texture=2.7%, mesh=0.8%



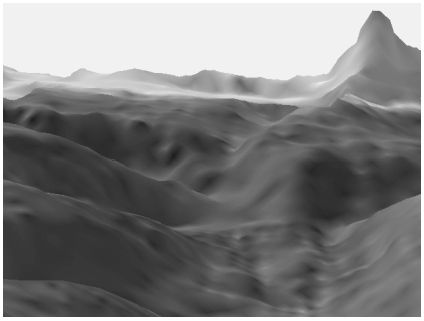
Iso-lines of constant joint-rate.



c) Bitrate=27.2KB, PSNR=25.5dB, texture=0.9%, mesh=2.7%

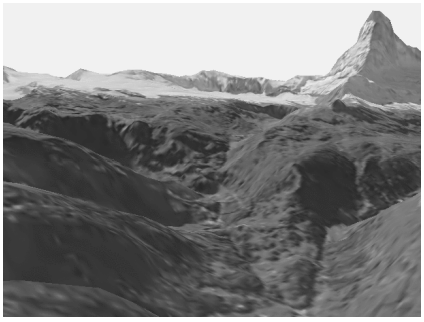


e) Bitrate=27.1KB, PSNR=22.7dB, texture=0.1%, mesh=4.9%

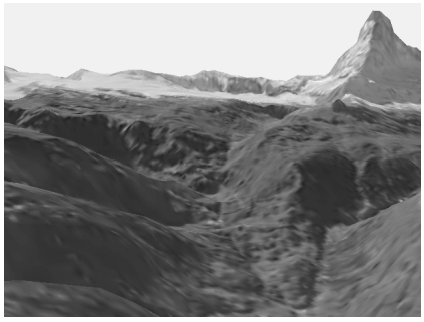


d) Bitrate=26.8KB, PSNR=24.2dB, texture=0.2%, mesh=4.2%

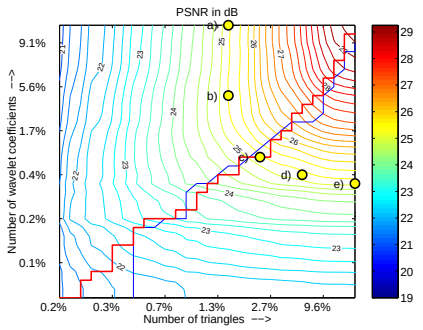
Figure 7.7: Trade-off between texture and mesh resolution with a *constant joint rate*. From a) to e) the texture resolution decreases and the mesh resolution increases. The best compromise in rate-distortion sense is c). The complexity is given in percent of the full model.



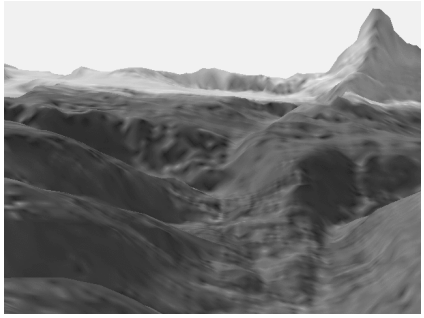
a) Bitrate=59.9KB, PSNR=25.2dB, texture=10.1%, mesh=1.6%



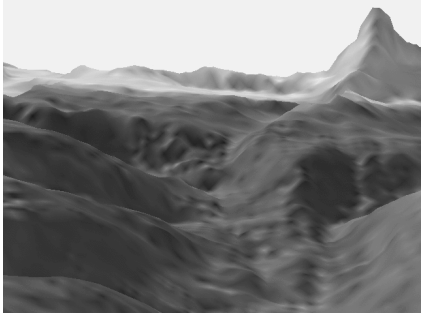
b) Bitrate=38.2KB, PSNR=25.3dB, texture=4.3%, mesh=1.6%



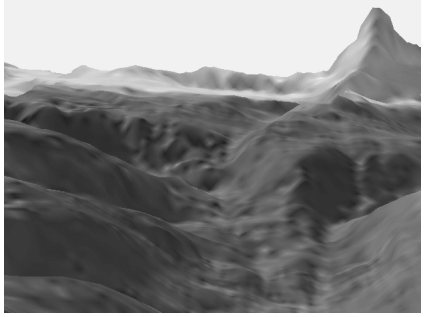
Iso-lines of constant PSNR.



c) Bitrate=23.7KB, PSNR=25.2dB, texture=0.7%, mesh=2.4%

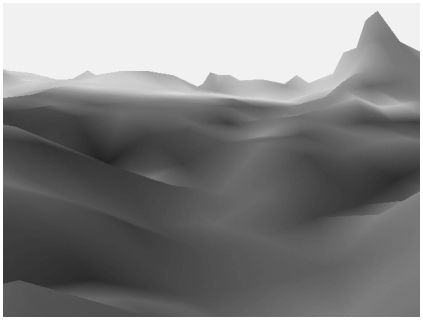


e) Bitrate=46.8KB, PSNR=25.1dB, texture=0.3%, mesh=7.5%

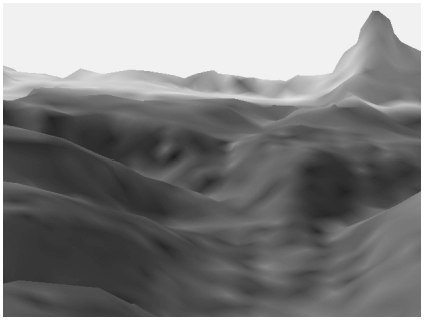


d) Bitrate=30.4KB, PSNR=25.2dB, texture=0.4%, mesh=4.2%

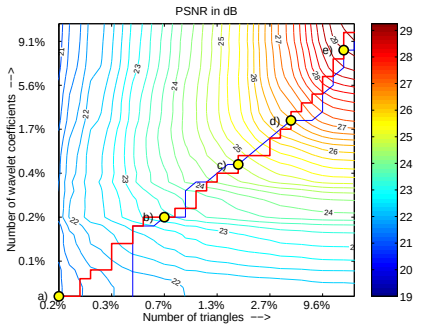
Figure 7.8: Trade-off between texture and mesh resolution with a *constant PSNR*. From a) to e) the texture resolution decreases and the mesh resolution increases. The best compromise is c). The complexity is given in percent of the full model. Although a human viewer may perceive that a) looks better than b), the measured PSNRs are similar. This is due to the low mesh resolution which causes small displacements of the textured triangles. Another case is d) and e): The refinement of the mesh does not improve the PSNR, because the texture resolution is low. Although the PSNRs are similar in the five images, the appearances differ significantly.



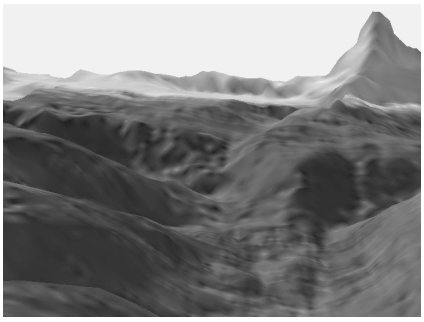
a) Bitrate=1.0KB, PSNR=21.2dB, texture=0.1%, mesh=0.2%



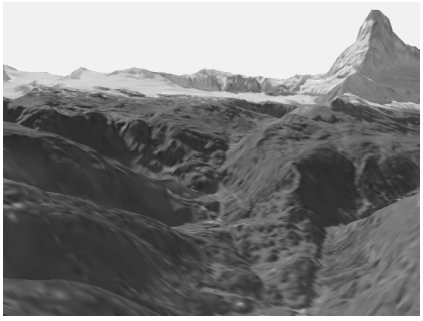
b) Bitrate=6.2KB, PSNR=23.2dB, texture=0.2%, mesh=0.7%



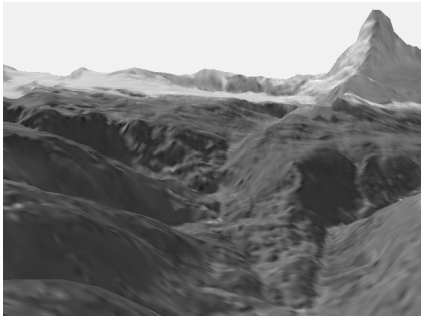
Iso-lines of constant PSNR. Solutions on optimal path



c) Bitrate=19.0KB, PSNR=24.7dB, texture=0.6%, mesh=1.8%



e) Bitrate=80.5KB, PSNR=29.0dB, texture=8.5%, mesh=6.8%

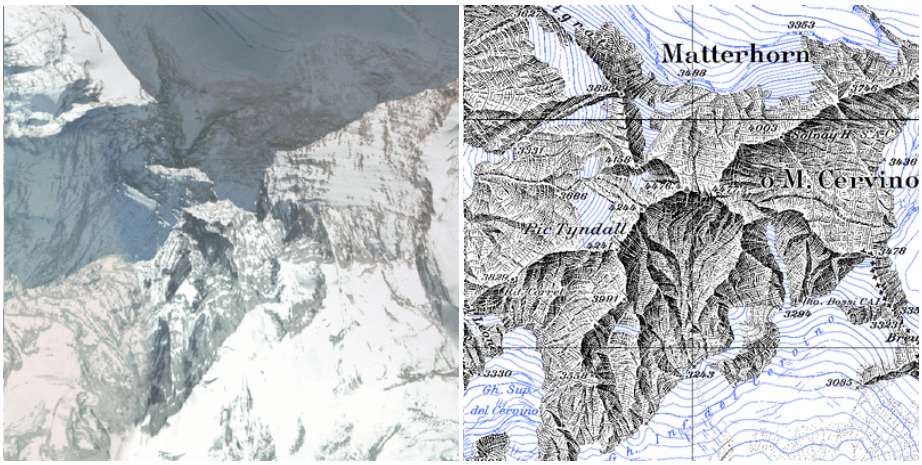


d) Bitrate=40.4KB, PSNR=26.5dB, texture=2.2%, mesh=3.6%

Figure 7.9: *The optimal path* of trade-offs between texture and mesh resolutions. From a) to e) the joint bitrate increases progressively. The bit budget is optimally balanced between the mesh and the texture. We can identify the mapped view-dependent textures from Figure 7.6. Texture 7.6c) is mapped on 7.9e) and the texture 7.6d) is mapped on 7.9c).

(a) (b) (c)

Figure 7.10: Views of Matterhorn from Grönergrat—You may guess first which one is a real picture: a) Rendered with Orthophoto, b) Rendered? Photo taken by digital camera, c) Rendered with synthetic texture (generated by Terragen). The mesh is based on DHM25 data.



(a) (b)

Figure 7.11: Position mismatching of the Matterhorn orthophoto (a, 5m per pixel) with geographic map (b, DHM25). The perspective correction was performed by Swissphoto AG without taking into account the distortion due to the height field. The camera was situated north of Matterhorn, therefore the south side of Matterhorn orthophoto was wrapped northwards.

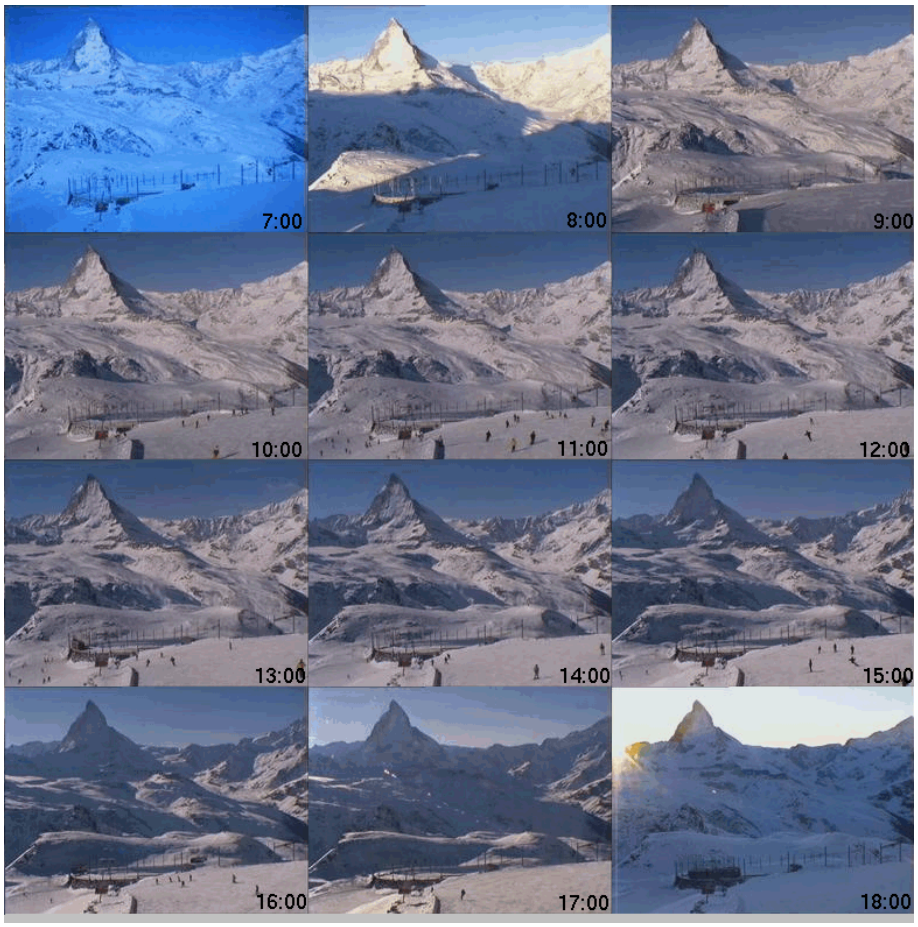


Figure 7.12: Matterhorn on 26 February 2001. Pictures taken from Grönergrat at the full hours from 7:00 (top left) to 18:00 (bottom right). Pictures courtesy of Topin webcam.

Chapter 8

General Conclusion

"Every man gets a narrower and narrower field of knowledge in which he must be an expert in order to compete with other people. The specialist knows more and more about less and less and finally knows everything about nothing."

— Konrad Lorenz

8.1 Summary

The presented work is a novel approach in applying and combining recent research results from B-spline signal processing, sampling theory, wavelets and coding to improve image processing algorithms in the fields of computer tomography, volume visualization, texture mapping and texture coding.

In the first part of this thesis, we discussed the Radon transform and its inverse; the key mathematical tools for the reconstruction of images in computer tomography and for the rendering of volumes by parallel projection.

We proposed a novel and exact discretization of the Radon transform and of its inverse based on least-squares approximation and the convolutions of splines. Both the sinogram and the reconstructed image are modeled by splines in such a way that we can calculate analytically the projection. Our method improves the reconstruction quality significantly compared to the standard FBP methods.

Next, we considered the (dual) problem of volume visualization by the X-ray transform. Wavelet splatting is a common acceleration technique for this kind of visualization. We proposed two improvements: First, we enhanced the quality by using least-squares approximation for the projected wavelets (droplets) instead of ordinary interpolation and sampling. Second, we projected onto a multiresolution grid that adapts to the scale of the droplet. This accelerates the rendering process at coarser scales by one to two order of magnitudes.

In the second part, we optimized texture mapping methods. First, we proposed a novel recursive method, which minimizes the information loss. It reconstructs the texture from the mapped texture and compares it to the original one. This algorithm is close to the l_2-optimal solution and outperforms the existing methods in terms of quality, but its complexity can be high. A multiresolution version of the algorithm allows to keep the storage requirements and computational complexity within acceptable range.

Next, we took advantage of the inevitable losses during the texture mapping process to compress textures such that after mapping no degradations are visible. We identified filters that simulate these losses in the frequency domain and, together with the viewing conditions, we approximated these filters in the DCT and wavelet domain. We applied them to generate view-dependent compressed textures that can be streamed at very low bitrates. This concept has been successfully introduced into the MPEG-4 compression standard.

Last, we investigated how to balance a bit budget to jointly code a view-dependent mesh and texture. By exhaustive search we found the solutions that are optimal in the rate-distortion sense. We proposed a marginal analysis algorithm that produces solution that are close to the optimal, but for much less computational costs.

8.2 Possible extensions

As a matter of fact, a research work is never really finished and can always be extended. Our intensive study on the combination of geometric image transformation with least-squares image processing revealed a number of aspects that need further investigation:

- Extend spline-based FBP algorithm to fan-beam geometry (irregular sampled sinogram corrected by warping).
- Combine advanced spline-based approximation of the sinogram (with fractional spline or oblique projection) with the Radon-kernel-based algorithm.
- Include noise reduction in the spline-based FBP algorithm (by fitting smoothing spline or spline wavelets).
- Volume rendering with additional volume shading (based of amplitude and direction of volume gradient vector of spline basis functions).
- Extend view-dependent compression with non-binary shrinking masks.
- Include full compression of the view-dependent mesh in the joint mesh-texture simplification joint mesh texture simplification experiments.
- Exploit joint mesh-texture simplification for streamed video sequences or animated scenes.

Acknowledgments

I'd like to take the opportunity to express my sincere gratitude to all people that generously gave me their support, time and energy.

First of all, I thank my advisor, Professor Michael Unser for his support, encouragement, and confidence. He is a very dedicated scientist and receptive supervisor.

I am very grateful to have worked together with my co-advisor, Professor Martin Vetterli. Martin's inexhaustible energy and his innovative thinking have always impressed me.

I would like to thank our ingenious mathematician Thierry Blu with whom it is amazing to work as he can explain and prove almost everything.

Many thanks to my students and especially my colleague Michael Liebling for great part of the C implementation of the computed tomography algorithms and our stimulating discussions.

Thanks to our first assistant Philippe Thevenaz for his precise remarks and help on the recursive texture mapping part. Special thanks to my colleague Laurent Balmelli which had the questionable privilege to work with me through the Christmas vacation 1999 to meet the SIGGRAPH deadline. More thanks to my former colleague Fred Jordan and our former supervisor Touradj Ebrahimi for the inspiring discussions and fruitful guidance during the MPEG-4 project.

I would also like to especially acknowledge the great effort that went into the reviewing of the drafts of this thesis by Michael Unser, Martin Vetterli, Eric Meijering, Arrate Munoz, and Michael Sühling. The thesis has benefited greatly from their suggestions. Any error that remain are purely my responsibility.

Many friends and fellow PhD from Michael's group and Martin's laboratory have also helped and inspired my along the way, including Jan Kybic, Manuela Feilner, Salvica Jonic, Gloria Menegaz, Paulo Prandoni, Pina Marziliano, David Hasler, Cristian Ciresan and Christian Schott.

I appreciated the great support from the staff in both laboratories. First, our computer supporting staff: Daniel Sage, Denice Deatrich, Claude Mary, Marie-Lorraine Bontron, and, last but not least, our hard-working secretaries: Martine Peter, Yvette Bernhard and Jocelyne Plantefol.

I want to thank my parents, Christa and Klaus Horbelt, for their love and encouragement.

Special thanks to Nadja Eschbaum for her love and understanding during the last three years. Without her patience I could not have finished this thesis.

Notations and Acronyms

Notations

α	fractional degree α of a polynomial function
α	rotation angle
β	tilt angle
$\beta_h^n(x)$	causal B-spline of degree n and width h
$\hat{\beta}^n(\omega)$	$= \text{sinc}^{n+1}(\omega/2\pi) = \sin(\omega/2)/(\omega/2)$.
$\beta_{h_1,\dots,h_m}^{n_1,\cdots,n_m}(x)$	The B-spline convolution kernel $\beta_{h_1,\dots,h_m}^{n_1,\cdots,n_m}(x) = \beta_{h_1}^{n_1} * \cdots * \beta_{h_m}^{n_m}(x)$
$\beta_*^\alpha(x)$	Symmetric fractional B-spline of degree α
$(b^n)^{-1}(k)$	B-spline prefilter
γ	fractional degree (real number)
$\delta(x)$	Dirac delta function
ε	approximation error
$\eta(x)$	cardinal spline
θ	projection angle
$\boldsymbol{\theta}$	unit vector $\boldsymbol{\theta} = (\cos\theta, \sin\theta)$ defining projection direction
λ	normalisation factors
ξ	kernel function
ν	frequency

$\xi(x)$	kernel function
$\varphi(x),\ \tilde{\varphi}(x)$	basis function $\varphi(x)$, and dual basis function $\tilde{\varphi}(x)$
$\chi(x)$	droplet function (projected wavelet function)
$\psi(x)$	wavelet function
ω	angular frequency $\omega = 2\pi\nu$
Δ	finite difference operator $\Delta = \delta(x) - \delta(x-1)$
Δ_*	symmetric finite difference operator $\Delta_* \leftrightarrow \lvert 1 - e^{-j\omega}\rvert$
Λ	normalisation diagonal matrix
Φ	wavelet
$x, \boldsymbol{x}$	variable x, vector $\boldsymbol{x} = (x, y)$
a_{12}	cross correlation between two functions
$c(k), c_k$	coefficients at shifts k
d	viewing distance
e	absolute error
f	focal distance
$f(x), f(\boldsymbol{x})$	function
$\hat{f}(\omega), \mathcal{F}$	Fourier transform of function $f(x) \leftrightarrow \hat{f}(\omega) = \int_{\infty}^{+\infty} f(x)e^{-j\omega x}dx$. The inverse Fourier transform of $\hat{f}(\omega)$ is defined by $f(x) = \frac{1}{2\pi}\int_{\infty}^{+\infty} \hat{f}(\omega)e^{j\omega x}dx$.
$g(x)$	image, sinogram
h	sampling step on image
$h(x)$	general filter
i, j	integer indices
j	scale
k, l	integer shifts $k, l \in Z$, screen indices

l	level in mipmap pyramid
l_2	Hilbert space of all discrete square-sumable functions.
l_2-norm	is $\|f\| = \left(\sum_{k=-\infty}^{+\infty} \|f(k)\|^2\right)^{1/2} = \sqrt{\langle f, f\rangle}$.
m, n	texture indices
n	integer degree of a polynomial function
$p(x)$	filtered sinogram
p_θ	parallel projection at angle θ
q	ramp filter
rect$(T\omega)$	$= 1$ for $\|T\omega\| < \pi$, otherwise 0
s	sampling step on sinogram
sinc(x)	$= \sin(\pi x)/\pi x$.
supp$(f(x))$	the support of a function $f(x)$ is the range $x \in [x_0, x_1]$ for which holds $f(x) = 0\ \forall x < x_0$ OR $x > x_1$.
s, t, u, v	continuous shift $s, t, u, v \in R$
s_x, s_y, s_z	screen coordinates
t_x, t_y, t_z	texture coordinates
$(x)_+^n$	one-sided power function
z	zoom factor
z^{-1}	minification factor

$\boldsymbol{A}$	matrix
$A,\ B$	strictly positive constants
$C(M, T)$	cost function
$D(M, T)$	distortion function
D_*	symmetric derivate $D_* f(x) \leftrightarrow \|\omega\|\hat{f}(\omega)$

$E(\omega)$	Fourier transform of error kernel
$H(\omega)$	Fourier transform of general filter
$K,\ L$	discrete sizes, angular resolution K, screen size $K \times L$
L_2	Hilbert space of all continuous square-integrable—in Lebesgue's sense—functions
L_2-norm	is $\|f\| = \left(\int_{-\infty}^{+\infty} \|f(x)\|^2 \, dx\right)^{1/2} = \sqrt{\langle f, f \rangle}$.
M_i	approximated mesh version
M, N	discrete sizes, image size $N \times N$ or $N \times M$
P	projection operator
$P_{2\perp 1}$	oblique projector operator
$R,\ R_\theta$	Radon transform R, and projection R_θ for one angle θ
$R^*,\ R_\theta^*$	Inverse Radon transform R^*, and backprojection operator R_θ^* for one angle θ
T_j	approximated texture version
$V(\varphi)$	space spanned by basis functions φ

Acronyms

ART	**A**lgebraic **R**econstruction **T**echnique
B-spline	**B**asic **spline** of degree n allows to construct all possible splines of degree n
CT	**C**omputer(-ized) **T**omography
DCT	**D**iscrete **C**osine **T**ransform. It is a good approximation of the Karhunen-Loève (KL) transform.
DHM25	**D**igital **H**eight **M**odel with **25** meter resolution
Droplet	Parallel projection of 3D wavelet onto a 2D surface, also called wavelet splat
EOB	**E**nd-**o**f-**B**lock code indicating the end of a coded DCT block.
EWA	**E**lliptical **W**eighted **A**verage is a texture mapping method with a spatial-variant filter that approximate the pixel footprint.
FBP	**F**iltered **B**ack-**P**rojection is a standard algorithm for CT reconstruction.
Felines	**F**ast **e**lliptical **lines** is a fast approximation of the EWA algorithm.
FFT	**F**ast **F**ourier **T**ransform is a fast version of the Fourier transform and has a computational complexity of $O(N \log N)$.
FOV	**F**ield-**o**f-**V**iew is the view frustum seen by virtual camera.
Intra	Intra coded image are coded in reference on to itself.
Inter	Inter coded image are coded in reference to previous or following frames.
ITU	**I**nternational **T**elecommunications **U**nion
JMTS	**J**oint **M**esh-**T**exture **S**implification
JPEG	**J**oint **P**icture **E**xpert **G**roup is a working group of ITU for still image coding standard.
Mip-map	Mip comes from the Latin *multum in parvo*—many things in a small place. A Mip-map is a pyramid of image at dyadic scales.
MPEG	**M**otion **P**icture **E**xpert **G**roup is working group ITU/SC29/WG11 for coding stream of images and multimedia data.
MRI	**M**agnetic **R**esonance **I**maging

PET **P**ositron **E**mission **T**omography is a medical imaging technique based on the annihilation of positron-emitting isotopes and tomographic reconstruction.

PSNR **P**eak **S**ignal-to-**N**oise **R**atio: $10\log(\text{signal peak}/\text{mean of noise})$

R-D **R**ate-**D**istortion

SAT **S**um **A**rea **T**able

SNHC **S**ynthetic **N**atural **H**ybrid **C**oding is a part of the MPEG-4 standard.

SPECT **S**ingle **P**ositron **E**mission **C**omputed **T**omography is a medical imaging technique based on nuclear medicine imaging and tomographic reconstruction.

Texture 1. An image warped or draped over a polygonal surface. 2. A repetitive pattern.

VD **V**iew-**D**ependent: Reducing information invisible from current point-of-view.

Wavelet Multiscale basis function

WYSIWYG What you see is what you get [visivig].

Bibliography

[1] A. Aldroubi and M. Unser, *Wavelets in Medicine and Biology*. CRC Press, 1986.

[2] A. Aldroubi, M. Unser and M. Eden, "Cardinal spline filters: stability and convergence to the ideal sinc interpolator", *Signal Processing*, vol. 28, no. 2, pp. 127–138, 1992.

[3] L. Balmelli, "Rate-distortion optimal mesh simplification for comunications", *Ph.D. thesis, Communication System Dept., Ecole Polytechnique Federale de Lausanne (EPF-ETH), Switzerland. available at lcavwww.epfl.ch/˜balmelli/*, , 2000.

[4] L. Balmelli, S. Ayer and M. Vetterli, "Efficient algorithms for embedded rendering of terrain models", *Proc. Int. Conf. Image Proc.*, vol. 2, pp. 914–918, Oct. 1998.

[5] E. V. R. D. Bella, A. B. Barclay, R. L. Eisner and R. W. Schafer, "A comparison of rotation-based methods for iterative reconstruction algorithms", *IEEE Trans. Nucl. Sci.*, vol. 43, no. 6, pp. 3370–3376, Dec. 1996.

[6] T. Blu, P. Thévenaz and M. Unser, "Moms: Maximal-order interpolation of minimal support", *IEEE Trans. on Image Proc.*, , 2001. to appear.

[7] T. Blu and M. Unser, "Quantitative Fourier analysis of approximation techniques: Part I—interpolators and projectors", *IEEE Trans. on Signal Proc.*, vol. 47, no. 10, pp. 2783–2795, Oct. 1999.

[8] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization", *IEEE Trans. on Image Proc.*, vol. 5, no. 4, pp. 480–492, Mar. 1996.

[9] P. Burt and E. Adelson, "The Laplacian pyramid as a compact code", *IEEE Trans. Commun.*, vol. 31, no. 4, pp. 337–345, 1983.

[10] E. J. Candés and D. L. Donoho, "Ridgelets: a key to higher-dimensional intermittency", *Phil. Trans. R. Soc. Lond.*, vol. 1760, no. 357, pp. 2495–2509, Sept. 1999.

[11] D. Cohen-Or, “Exact anti-aliasing of textured terrain models”, *Int. J. of Comp. Graphics*, vol. 13, no. 4, pp. 184–198, 1997.

[12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[13] F. C. Crow, “Summed-area tables for texture mapping”, *Computer-Graphics (Proc. of Siggraph)*, vol. 18, no. 3, pp. 207–212, July 1984.

[14] I. Daubechies, “Othrogonal bases of compactly supported wavelets”, *IEEE Trans. Inform. Th.*, vol. 41, pp. 909–996, 1988.

[15] S. R. Deans, *The Radon transform and Some of its Applications*. John Wiley, New York, 1983.

[16] A. H. Delaney and Y. Bresler, “Multiresolution tomographic reconstruction using wavelets”, *IEEE Trans. on Image Proc.*, vol. 4, no. 6, p. 799ff, 1995.

[17] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich and M. B. Mineev-Weinstein, “Roaming terrain: Real-time optimally adapting meshes”, *Proc. of IEEE Visualization*, , 1997.

[18] J. P. Ewins, M. D. Waller, M. White and P. F. Lister, “Mip-map level selection for texture mapping”, *IEEE Trans. Visual. and Comp. Graph.*, vol. 4, no. 4, pp. 317–329, Oct. 1998.

[19] J. D. Foley, *Computer Graphics, Principles and Practice*. Addison-Wesley, 1996.

[20] D. Ghazanfarpour and B. Peroche, “A high-quality filtering using forward texture mapping”, *Computer-Graphics (Proc. of Siggraph)*, vol. 15, no. 4, pp. 569–577, 1991.

[21] A. Glassner, “Adaptive precision in texture mapping”, *Computer-Graphics (Proc. of Siggraph)*, vol. 20, no. 4, pp. 297–306, Aug. 1986.

[22] G. Glover and N. Pelec, “An algorithm for the reduction of metal clip artifacts in ct reconstructions”, *J. on Med. Phys.*, vol. 8, no. 6, pp. 799–807, 1981.

[23] W. A. Goetz and H. J. Druckmüller, “A fast digital Radon transform”, *J. on Pattern Recognition*, vol. 29, pp. 711–718, 1996.

[24] D. Gottleib, B. Gustafsson and P. Forssen, “On the direct Fourier method for computer tomography”, *IEEE Trans. Med. Imag.*, vol. 19, no. 3, pp. 223–232, 2000.

[25] P. J. Green, “Bayesian reconstruction from emission tomography data using a modified em algorithm”, *IEEE Trans. Med. Imag.*, vol. 9, no. 1, pp. 84–93, Mar. 1990.

[26] M. H. Gross, L. Lippert and R. Dittrich, “Two methods for wavelet-based volume rendering”, *Comp. & Graphics*, vol. 21, no. 2, pp. 237–252, 1997.

[27] J.-P. Guédon and Y. Bizais, "Bandlimited and haar filtered back-projection reconstruction", *IEEE Trans. Med. Imag.*, vol. 12, no. 3, pp. 430–440, 1994.

[28] T. Hebert and R. Leahy, "A generalized EM algorithm for 3-d bayesian reconstruction from poisson data with gibbs priors", *IEEE Trans. Med. Imag.*, vol. 8, no. 2, pp. 194–202, Mar. 1990.

[29] P. Heckbert, Ed., *Graphics Gems I-IV*. Academic Press, Boston, 1994.

[30] P. S. Heckbert, "Filtering by repeated integration", *Computer-Graphics (Proc. of Siggraph)*, vol. 20, no. 4, pp. 286–293, Aug. 1986.

[31] P. S. Heckbert, "Survey of texture mapping", *IEEE Comp. Graphics and Appl.*, vol. 6, no. 11, pp. 56–67, 1986.

[32] G. Henrich, "A simple computational method for reducing streak artifacts in CT images", *Computerized-Tomography*, vol. 4, pp. 67–71, 1980.

[33] G. T. Herman and D. Odhner, "Performance evaluation of an iterative image reconstruction algorithm for positron emission tomography", *IEEE Trans. Med. Imag.*, vol. 3, no. 4, pp. 316–324, Dec. 1991.

[34] G. Herrmann, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, New York, 1980.

[35] H. Hoppe, "Progressive meshes", *Computer-Graphics (Proc. of Siggraph)*, pp. 99–108, 1996.

[36] H. Hoppe, "View-dependent refinement of progressive meshes", *Computer-Graphics (Proc. of Siggraph)*, pp. 189–198, 1997.

[37] S. Horbelt. "Progressive view-dependent texture coding for the transmission of virtual environments". Tech. Rep. doctoral school, EPFL/SSC, July 1997.

[38] S. Horbelt, "Geometric image processing", *Ph.D. thesis, Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland*, , 2001.

[39] S. Horbelt, L. Balmelli and M. Vetterli, "Joint mesh/texture coding using marginal analysis", *IEEE Trans. on Image Proc.*, , 2001.

[40] S. Horbelt, L. Balmelli and M. Vetterli. "Joint mesh/texture compression using marginal analysis". In *Proc. Int. Conf. Image Proc.*, Tessaloniki, Greece, Sep. 2001. submitted.

[41] S. Horbelt, F. Jordan and T. Ebrahimi. "Results of core experiment V1 using DCT". Tech. Rep. M2279, ITU-T/SC29/WG11, July 1997.

[42] S. Horbelt, F. Jordan and T. Ebrahimi. "Results of core experiment Y2". Tech. Rep. M1904, ITU-T/SC29/WG11, Apr. 1997.

[43] S. Horbelt, F. Jordan and T. Ebrahimi. "Streaming of photo-realistic texture mapped on 3d surfaces". In *IWSNHC3D'97: Int. Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging*, Sept 1997.

[44] S. Horbelt, F. Jordan and T. Ebrahimi. "View-dependent texture coding for transmission of virtual environments". Tech. Rep. M1723/X3, ITU-T/SC29/WG11, Feb. 1997.

[45] S. Horbelt, F. Jordan and T. Ebrahimi. "View-dependent texture coding for virtual environments". In *Int. Conf. on Image Proc. and its Appl. (IPA)*, Dublin, Ireland, vol. 1, pp. 433–437, July 1997.

[46] S. Horbelt, M. Liebling and M. Unser, "Discretisation of the Radon transform and its inverse by spline convolutions", *IEEE Trans. Med. Imag.*, , Nov. 1999. submitted.

[47] S. Horbelt, M. Liebling and M. Unser. "Filter design for filtered backprojection constrained by the interpolation model". In *Proc. Int. Conf. Image Proc.*, Tessaloniki, Greece, 2001. submitted.

[48] S. Horbelt, A. Munoz, T. Blu and M. Unser. "Spline kernels for continuous-space image processing". In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Proc.*, Istanbul, Turkey, vol. 4, pp. 2191–2194, June 2000.

[49] S. Horbelt, P. Thévenaz and M. Unser. "Texture mapping by successive refinement". In *Proc. Int. Conf. Image Proc.*, Vancouver, Canada, pp. 307–310, Sep. 2000.

[50] S. Horbelt, P. Thévenaz and M. Unser, "Least-squares texture mapping", *IEEE Trans. on Image Proc.*, , 2001. to be submitted.

[51] S. Horbelt, M. Unser and M. Vetterli. "Wavelet projections for volume rendering". In *Eurographics, Short Papers & Demos*, Milano, Italy, pp. 56–59, Sep. 1999.

[52] S. Horbelt, M. Vetterli and M. Unser. "High quality wavelet splatting for volume rendering". In *Wavelet Application Workshop*, Monte Verità, Ticino, Oct. 1998.

[53] C.-L. Huang and K.-C. Chen, "Directional moving average interpolation for texture mapping", *Graph. Models and Image Proc.*, vol. 58, no. 4, pp. 301–313, July 1996.

[54] A. K. Jain, *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[55] F. Jordan, S. Horbelt and T. Ebrahimi. "Photo-realistic texture mapped on 3d surface". In *Proc. Int. Conf. Image Proc.*, Santa Barbara, CA, vol. 2, pp. 390–393, 1997.

[56] F. Jordan, S. Horbelt and T. Ebrahimi. "Results of core experiment V1 using wavelets". M2278, ITU-T/SC29/WG11, July 1997.

[57] F. Jordan, S. Horbelt and T. Ebrahimi. “Results on core experiment X3 for texture coding”. Tech. Rep. M1737, ITU-T/SC29/WG11, Feb. 1997.

[58] P. M. Joseph and R. D. Spital, “A method for correcting bone included artifacts in computed tomography scanners”, *JCAT*, vol. 2, no. 1, pp. 101–108, 1978.

[59] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE press, 1988.

[60] A. Khodakovsky, P. Schröder and W. Sweldens, “Progressive geometry compression”, *Computer-Graphics (Proc. of Siggraph)*, pp. 271–278, 2000.

[61] K.Ramchandran and M.Vetterli, “Best wavelet packet bases in a rate-distortion sense”, *IEEE Trans. on Image Proc.*, vol. 2, no. 2, pp. 160–175, Apr. 1993.

[62] D. Laur and R. Hanrahan, “Hierarchical splatting: A progressive refinement algorithm for volume rendering”, *Computer-Graphics (Proc. of Siggraph)*, vol. 25, no. 4, pp. 285–289, 1991.

[63] M. Lautsch, “A spline inversion formula for the Radon transform”, *SIAM J. Math. Anal.*, vol. 26, no. 2, pp. 456–467, 1989.

[64] T. M. Lehmann, C. Gonner and K. Spitzer, “Survey: interpolation methods in medical image processing”, *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1049–1075, Nov. 1999.

[65] B. Lichtenbelt, R. Crane and S. Naqvi, *Introduction to Volume Rendering*. Hewlett-Packard, Prentice-Hall, 1998.

[66] P. Lindstrom, D. Koller, W. Ribarsky, L. Hodges, N. Faust and G. Turner, “Real-time continuous level of detail rendering of height fields”, *Computer-Graphics (Proc. of Siggraph)*, pp. 109–118, 1996.

[67] L. Lippert and M. Gross, “Fast wavelet-based volume rendering by accumulation of transparent texture maps”, *Eurographics*, vol. 14, no. 3, pp. 431–444, 1995.

[68] D. Ludwig, “The Radon transform on euclidean space”, *Comm. Pure and Applied Math.*, vol. 19, pp. 49–81, 1966.

[69] S. Mallat, “A theory for multiresolution signal decomposition: The wavelet respresentation”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 7, pp. 674–693, July 1989.

[70] S. Mallat, *A Wavelet Tour of Signal Processing*. Academic Press, San Diego, CA, 1998.

[71] J. McCormack, R. Perry, K. I. Farkas and N. P. Jouppi, “Feline: Fast elliptical lines for anisotropic texture mapping”, *Computer-Graphics (Proc. of Siggraph)*, pp. 243–250, 1999.

[72] E. H. W. Meijering, "Spline interpolation in medical imaging: Comparison with other convolution-based approaches", *Proc. EUSIPCO*, vol. 4, pp. 1989–1996, Sept. 2000.

[73] F. Natterer, *The mathematics of computerized tomography*. John Wiley, Chicester, 1986.

[74] T. Olson and J. DeStefano, "Wavelet localization of the Radon transform", *IEEE Trans. on Signal Proc.*, vol. 42, no. 8, pp. 2055–2067, 1994.

[75] J. Radon, "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten", *Berichte Sächsische Akademie der Wissenschaften, Math.-Phys. Kl.*, vol. 69, pp. 262–267, 1917.

[76] G. N. Ramachandran and A. V. Lakshminarayanan, "Three dimensional reconstructions from radiographs and electron micrographs: Application of convolution instead of Fourier transform", *Proc. Nat. Acad. Sci.*, vol. 68, pp. 2236–2240, 1971.

[77] F. Rashid-Farrokhi, K. J. R. Liu, C. A. Berenstein and D. Walnut, "Wavelet-based multiresolution local tomography", *IEEE Trans. on Image Proc.*, vol. 6, no. 10, pp. 1412–1430, 1997.

[78] M. Richter, "Use of box splines in computer tomography", *Computing*, vol. 61, pp. 133–150, 1998.

[79] P. G. L. Riviére and X. Pan, "Spline-based inverse Radon transform in two or three dimensions", *IEEE Trans. Nucl. Sci.*, vol. 45, no. 4, pp. 2224–2231, Aug. 1998.

[80] R. A. Robb, *Three Dimensional Biomedical Imaging, Principles and Practice*. Wiley-Liss, NY, 1998.

[81] D. F. Rogers, *Procedural elements for computer graphics*. 2nd. Ed., McGraw-Hill, 1998.

[82] H. Rushmeier, B. Rogowitz and C. Piatko, "Perceptual issues in substituting texture for geometry", *SPIE Proceedings, Human Vision and Electronic Imaging V, B. Rogowitz and T. Pappas, Editors*, vol. 3959, pp. 372–383, 2000.

[83] A. Schilling, G. Kittel and W. Strasser, "Texram: A smart memory for texturing", *IEEE Comp. Graphics and Appl.*, pp. 32–41, May 1996.

[84] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytical functions, part A:—on the problem of smoothing or graduation", *Quart. Applied Math.*, vol. 4, no. 1, pp. 45–99, 1946.

[85] H. Schomberg and J.Timm, "The gridding method for image reconstruction by Fourier transformation", *IEEE Trans. Med. Imag.*, vol. 14, pp. 596–607, 1995.

[86] C. E. Shannon, "Communication in the presence of noise", *Proc. IRE*, vol. 37, pp. 10–21, 1949.

[87] L. Shepp and Y. Yardy, "Maximum likelihood reconstruction for emission tomography", *IEEE Trans. Med. Imag.*, vol. 1, no. 2, pp. 113–122, 1982.

[88] L. A. Shepp and B. F. Logan, "The Fourier reconstruction of a head section", *IEEE Trans. Nucl. Sci.*, vol. 21, pp. 21–43, 1974.

[89] D. Slepian, "On bandwidth", *Proceedings of IEEE*, vol. 64, no. 3, pp. 292–300, Mar. 1975.

[90] H. Stark, J. Woods and R. Hingorani, "Direct Fourier reconstruction in computer tomography", *IEEE Trans. Acoust. Speech Signal Proc.*, vol. 29, pp. 237–244, 1981.

[91] G. Strang and T. Nguyen, *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

[92] P. Thévenaz, T. Blu and M. Unser, "Interpolation revisited", *IEEE Trans. Med. Imag.*, vol. 9, no. 17, pp. 739–758, July 2000.

[93] P. Thévenaz, U. E. Ruttimann and M. Unser, "A pyramid approach to subpixel registration based on intensity", *IEEE Trans. on Image Proc.*, vol. 7, no. 1, pp. 1–15, Jan. 1998.

[94] M. Unser, "Splines: a perfect fit for signal and image processing", *IEEE Signal Proc. Mag.*, vol. 16, no. 6, pp. 22–38, Nov. 1999.

[95] M. Unser, "Sampling–50 years after Shannon", *Proceedings of IEEE*, vol. 88, no. 4, pp. 569–587, Apr. 2000.

[96] M. Unser and A. Aldroubi, "A general sampling theory for non-ideal acquisition devices", *IEEE Trans. on Signal Proc.*, vol. 42, no. 11, pp. 2915–2925, Nov. 1994.

[97] M. Unser, A. Aldroubi and M. Eden, "Fast B-spline transforms for continuous image representation and interpolation", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 3, , Mar. 1991.

[98] M. Unser, A. Aldroubi and M. Eden, "B-spline signal processing: Part I—theory", *IEEE Trans. on Signal Proc.*, vol. 41, no. 2, pp. 821–833, Feb. 1993.

[99] M. Unser, A. Aldroubi and M. Eden, "B-spline signal processing: Part II—efficient design and applications", *IEEE Trans. on Signal Proc.*, vol. 41, no. 2, pp. 834–848, Feb. 1993.

[100] M. Unser, A. Aldroubi and M. Eden, "The L_2 polynomial spline pyramid", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 364–379, 1993.

[101] M. Unser, A. Aldroubi and M. Eden, "Enlargement or reduction of digital images with minimum loss of information", *IEEE Trans. on Image Proc.*, vol. 4, no. 3, pp. 247–258, 1995.

[102] M. Unser and T. Blu, "Fractional splines and wavelets", *SIAM Review*, vol. 42, no. 1, pp. 43–67, 2000.

[103] M. Unser and I. Daubechies, "On the approximation power of convolution-based least squares versus interpolation", *IEEE Trans. on Signal Proc.*, vol. 45, no. 7, pp. 1697–1711, 1997.

[104] M. Unser, S. Horbelt and T. Blu. "Fractional derivates, splines and tomography". In *Proc. EUSIPCO*, Tampere, Finnland, pp. –, Sep. 2000.

[105] M. Unser, P. Thévenaz and L. Yaroslavsky, "Convolution-based interpolation for fast, high quality rotation of images", *IEEE Trans. on Image Proc.*, vol. 4, no. 10, pp. 1371–1381, 1995.

[106] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[107] J. Walden, "Analysis of the direct Fourier method for computer tomography", *IEEE Trans. Med. Imag.*, vol. 19, no. 3, pp. 211–222, 2000.

[108] F. M. Weinhaus and V. Devarajan, "Texture mapping 3d models of real-world scenes", *ACM Computing Survey*, vol. 29, no. 4, pp. 325–365, Dec. 1997.

[109] L. Willliams, "Pyramidal parametrics", *Computer-Graphics (Proc. of Siggraph)*, vol. 17, pp. 1–11, 1983.

[110] Z. Xiong, K. Ramchandran and M. T. Orchard, "Space-frequency quantization for wavelet image coding", *IEEE Trans. on Image Proc.*, vol. 6, pp. 677–693, May 1997.

Curriculum vitæ

HORBELT Stefan

Audio-Visual Communications Laboratory
Biomedical Imaging Group
1015 EPFL DMT/IOA/BIG Lausanne
Switzerland

Phone: (+41) 21 693 5137
Email: horbelt@ieee.org

German
born 5th February 1969
Single

Home: (+41) 21 625 1039
URL: lcavwww.epfl.ch/horbelt

Current Research Areas

Image compression; Texture mapping; Volume rendering; Computer tomographic reconstruction

Education

1998–today *Ph.D. student*
Swiss Federal Institute of Technology (EPFL), Lausanne
- Thesis about Splines and Wavelets for Image Warping and Projection
- Practical application for high quality image processing and compression
- Advisors: Prof. Martin Vetterli (LCAV) and Prof. Michael Unser (BIG)

1996–1997 *Doctoral School in Communication Systems*
Swiss Federal Institute of Technology (EPFL), Lausanne
- Project on View-dependent texture coding, now part of MPEG-4
- Supervisors: Prof. Murat Kunt and Prof. Touradj Ebrahimi (LTS)
- GPA: 9.5/10.0

1989–1995 *M.Sc. in Electrical Engineering (Diplom-Ingenieur)*
Friedrich Alexander University (FAU), Erlangen-Nürnberg, Germany
- Fields of specialization: image processing, multimedia telecommunication
- Graduation mentioned "very good", GPA: 1.5/1.0

1994–1995 *Masters stage*
Institute Eurécom, Sophia Antipolis, France
- Master thesis about Bimodal Speech Recognition
- Supervisors: Prof. Bernd Girod (FAU/LNT) and Prof. Christian Wellekens

1992–1993 *Erasmus exchange student*
University College London (UCL), London
- Comparison of video compression algorithms

Awards

- EPFL/SSC fellowship (1997–2000): This 3-years fellowship is awarded every year to the two best students of the telecommunications doctoral school to complete a Ph.D.
- EPFL/SSC scholarship (1996–1997): This 1-year scholarship is given on competitive basis to the best students to attend the telecommunications doctoral school at EPFL, Lausanne.
- Erasmus exchange fellowship (1992–1993): Semester at the University College London.
- DAAD exchange fellowship (1992): Industrial stage in South Africa.

Professional Experience

1998–today	*Research and Teaching Assistant* **Swiss Federal Institute of Technology (EPFL), Lausanne** • TA for DSC graduate course on Wavelets and Applications • Supervision of students and contribution to laboratory infrastructure • Presentation at international conferences (Milano, Istanbul, Vancover)
1995–1996	*Research and Teaching Assistant* **Multimedia Communications Department, Institute Eurécom, France** • TA for undergraduate course on Image Analysis and Coding • Development of virtual teleconferencing application
Summer 1992	*Internship* **Siemens Ltd., Johannesburg, South Africa** • Diagnostic and communication systems of roll mill
1991–1992	*Interdisciplinary project realization* **High Frequency Institute, University Erlangen, Germany** • BodySway: Biomedical measurement system
Summer 1989	*Internship* **Siemens, UB Medizintechnik, Erlangen**
1988–1989	*Radio interception* **Military service, Germany**

Technical Skills

Signal Processing	Image Compression, Splines, Wavelets, High Quality Image Processing Computer Graphics, Biomedical Imaging and Reconstruction Multimedia Standards (MPEG 1/2/4, JPEG)
Computer	Unix, Linux, Windows 98/NT/2000 C/C++, Java, Perl, Python, OpenGL, Matlab, IDL, Khoros, Tcl/Tk, Pascal LaTeX,LyX, Word, PowerPoint, Excel, HTML, Photoshop, VHDL

Languages

German	Mother tongue
English	Fluent (oral and written)
French	Fluent (oral and written)

References

Professor Michael Unser Biomedical Imaging Group (BIG) CH-1015 EPFL-DMT, Lausanne, Switzerland	Email: Michael.Unser@epfl.ch Phone: +41 21 693 5175 URL: http://bigwww.epfl.ch/unser/
Professor Martin Vetterli Audio-Visual Communications Laboratory (LCAV) CH-1015 EPFL-DSC, Lausanne, Switzerland	Email: Martin.Vetterli@epfl.ch Phone: +41 21 693 5698 URL: http://lcavwww.epfl.ch/~vetterli/
Professor Touradj Ebrahimi Signal Processing Laboratory (LTS) CH-1015 EPFL-DE, Lausanne, Switzerland	Email: Touradj.Ebrahimi@epfl.ch Phone: +41 21 693 2606 URL: http://ltswww.epfl.ch/~ebrahimi/

Publications

Journal Papers

[1] S. Horbelt, M. Liebling, M. Unser, "Discretization of the Radon transform and its inverse by spline convolutions," *to appear in IEEE Trans. Med. Imag.*, 2001.

[2] S. Horbelt, L. Balmelli, M. Vetterli, "Joint mesh/texture coding using marginal analysis," *submitted to Trans. Imag. Proc.*, 2001.

[3] S. Horbelt, P. Thévenaz, M. Unser, "Least-squares texture mapping," *submitted to IEEE Trans. Imag. Proc.*, 2001.

Conference Papers

[4] S. Horbelt, M. Liebling, M. Unser, "Filter Design for Filtered Backprojection guided by the interpolation model," *submitted*, 2001.

[5] S. Horbelt, L. Balmelli, M. Vetterli, "Joint mesh/texture compression using marginal analysis," *to appear at IEEE Int. Conf. on Image Proc. (ICIP*), Thesaloniki, Greece, Oct 2001.

[6] S. Horbelt, P. Thévenaz, M. Unser, "Texture Mapping by Successive Refinement," *IEEE Int. Conf. on Image Proc. (ICIP*), pp. 307–310, Vancouver, Canada, Sep. 2000.

[7] S. Horbelt, A. Muñoz, T. Blu, M. Unser, "Spline Kernels for Continuous-Space Image Processing," *IEEE ICASSP*, vol. 4, pp. 2191–2194, Istanbul, Turkey, June 2000.

[8] M. Unser, S. Horbelt and T. Blu, "Fractional derivates, splines and tomography," *Proc. EUSIPCO'2000*, pp., Tampere, Finland, Sep., 2000.

[9] S. Horbelt, M. Unser, M. Vetterli, "Wavelet Projections for Volume Rendering," *Eurographics'99*, Short Papers & Demos, pp. 56–59, Milano, Italy, Sept. 1999.

[10] S. Horbelt, M. Vetterli, M. Unser, "High Quality Wavelet Splatting for Volume Rendering," *Wavelet Application Workshop*, Monte Verità, Ticino, 28 Sept.–2 Oct. 1998.

[11] S. Horbelt, F. Jordan, T. Ebrahimi, "Streaming of photo-realistic texture mapped on 3D surfaces," *IWSNHC3D'97: Int. Workshop on SNHC and 3D Imaging*, Sept. 1997, Rhodes, Greece.

[12] S. Horbelt, F. Jordan, T. Ebrahimi, "View dependent Texture Coding for Virtual Environments," *6th Int. Conf. on Image Proc. (IPA)*, vol. 1, pp. 433–437, Dublin, Ireland, July 1997.

[13] F. Jordan, S. Horbelt, T. Ebrahimi, "Streaming of photo-realistic texture mapped on 3D surface," *Int. Conf. on Image Proc. (ICIP)*, Santa Barbara, CA, vol.2, pp. 390–393 ,Oct. 1997.

[14] S. Horbelt, J. Crowcroft, "A hybrid BTC/ADCT video codex simulation bench," *7th Int. Workshop on Packet Video, Brisbane*, Australia, March 1996.

[15] S. Horbelt, J-L. Dugelay, "Active Contours for Lipreading—Combining Snakes with Templates," *15th GRETSI Symp. Signal and Imag. Proc.*, Juan les Pins, France, Sep. 1995.

ITU-T Documents

[16] S. Horbelt, F. Jordan, T. Ebrahimi, "View-Dependent Texture Coding for Transmission of Virtual Environments," *ITU-T/SC29/WG11/M1723/X3*, Sevilla, Feb.1997.

[17] S. Horbelt, F. Jordan, T. Ebrahimi, "Results of Core Experiment Y2," *ITU-T/SC29/WG11/M1904*, Bristol, April 1997.

[18] S. Horbelt, F. Jordan, T. Ebrahimi, "Results of Core Experiment V1 using DCT," *ITU-T/SC29/WG11/M2279*, Stockholm, July 1997.

[19] F. Jordan, S. Horbelt, T. Ebrahimi, "Results of Core Experiment V1 using Wavelets," *ITU-T/SC29/WG11/M2278*, Stockholm, July 1997.