# Verfahren zur verbesserten Approximation von Lichtverteilungen in der fotorealistischen Bildsynthese

Dissertation
zur Erlangung des Grades des
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik
von

Dipl.-Inform. Georg Pietrek

Lehrstuhl VII - Graphische Systeme
Fachbereich Informatik
Universität Dortmund

Dortmund
2001

Tag der mündlichen Prüfung: 23.4.2001

Dekan: Prof. Dr. Bernd Reusch

Gutachter:

Prof. Dr. Heinrich Müller
Universität Dortmund

Prof. Dr.-Ing. Claudio Moraga
Universität Dortmund

Prof. Dr. Reinhard Klein
Universität Bonn

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

Bei der fotorealistischen Bildsynthese geht es darum, aus der Beschreibung einer dreidimensionalen Szene, die aus Lichtquellen und Oberflächen, die mit Licht interagieren, besteht, realistisch wirkende Rasterbilder zu erzeugen (Abb. D.1, Seite 131). Allgemein akzeptierte Grundlage dieser Berechnung ist die Bildsynthesegleichung („Rendering-Equation“) von Kajiya [41]:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} \rho_{bd}(\mathbf{x}, \omega_o, \omega)\, L_i(\mathbf{x}, \omega)\, \cos\theta \, \mathrm{d}\omega$$

$L$ steht dabei für die Strahlungsdichte („Radiance“), die von einem Oberflächenpunkt $\mathbf{x}$ in eine Raumrichtung $\omega_o$ abgegeben wird. Sie setzt sich aus der Eigenstrahlung $L_e$ der Oberfläche an dem betrachteten Punkt in die gegebene Richtung sowie dem reflektierten Licht zusammen. Das reflektierte Licht ergibt sich durch Aufintegration des einfallenden Lichtes $L_i$ über alle relevanten Raumrichtungen $\omega$, welches noch mit einem Reflexionsfaktor $\rho_{bd}$ an der betrachteten Oberfläche gedämpft wird, der durch die bidirektionale Reflexionsverteilungsfunktion („BRDF“) gegeben ist. In der dargestellten Form umfaßt die Gleichung nur die Reflexion. Sie kann auf Brechungseffekte verallgemeinert werden, die jedoch in dieser Arbeit nicht Gegenstand der Untersuchung sind.

Die Bildsynthesegleichung läßt sich im allgemeinen nicht analytisch lösen. Für ihre näherungsweise Lösung wurden in der Vergangenheit eine Reihe von Verfahren entwickelt, die sich zu einem großen Teil auch auf Einschränkungen beziehen. So hat insbesondere die Beschränkung auf uniform diffuse Oberflächen in Form der sogenannten „Radiosity“-Berechnung viel Aufmerksamkeit gefunden. Durch die Richtungsunabhängigkeit der Reflexion vereinfacht sich die Gleichung in diesem Fall signifikant zur „Radiosity-Gleichung“.

Die bekannten Vorgehensweisen zur Lösung der Rendering-Equation lassen sich grob in Monte Carlo-Abtastverfahren und Finite-Elemente-Verfahren einteilen.

## 1.1 Monte Carlo-Ray Tracing mit adaptiven Dichtefunktionen

Monte Carlo-Ray Tracing ist das Verfahren, das zur Zeit zur Simulation aller optischen Effekte am praktikabelsten ist. Es basiert auf dem mathematischen Verfahren der Monte Carlo-Integration zur Berechnung von Integralen [42].

Basisoperation eines Monte Carlo-Ray Tracers ist die punktweise Abtastung des zu integrierenden Signals. Im Falle der Bildgenerierung handelt es sich dabei um die Abtastung der Hemisphäre, die durch Verfolgung von Strahlen in einzelne Richtungen erfolgt. Die Strahlverfolgung ist eine aufwendige Operation, da sie die Verdeckungen in der Szene berücksichtigen muß und die meisten Verfahren am Auftreffpunkt eines Strahls die rekursive Verfolgung eines oder mehrerer neuer Strahlen erfordern. Daraus resultiert, daß die Berechnung einer einzelnen Abtastung bereits relativ teuer (gemessen in Rechenzeit) ist. Man sollte deshalb danach streben, aus der Information der berechneten Abtastung maximalen Nutzen zu ziehen. Glassner formuliert dieses Prinzip in [28] als:

> **The Sampler's Credo:** Every sample is precious.

In dieser Arbeit wird ein Ansatz zur Effizienzsteigerung der Monte Carlo-Bilderzeugung vorgestellt, der eine mögliche Realisierung dieses Credos präsentiert.

Die mathematische Literatur zur Monte Carlo-Integration listet verschiedene Ansätze zur Effizienzsteigerung auf. Eine systematische Untersuchung der Anwendung dieser Ansätze auf Monte Carlo-Ray Tracing findet sich in [49]. Das in der vorliegenden Arbeit entwickelte Verfahren beruht auf der Effizienzsteigerung durch Importance Sampling. Die Bilderzeugung erfolgt in mehreren Durchläufen. Jeder Durchlauf besteht aus einem normalen Monte Carlo-Ray Tracing-Verfahren, wobei zusätzlich die in den einzelnen Abtastungen enthaltenen Informationen über den Verlauf des Integranden dazu genutzt werden, eine Approximation des Integranden zu berechnen. Nach dem Durchlauf werden aus den Approximationen neue Wahrscheinlichkeitsdichten bestimmt, die im folgenden Durchlauf benutzt werden. Die Wahrscheinlichkeitsdichten werden über hierarchischen Basisfunktionen repräsentiert. Die durchgeführten empirischen Untersuchungen zeigen eine wahrnehmbare Qualitätssteigerung der generierten Bilder bei etwa gleichem Berechnungsaufwand.

## 1.2 Rekonstruktion der Radiosity-Funktion

Beim Finite-Elemente-Ansatz wird die Lichtverteilungsfunktion über einem Funktionsraum repräsentiert, der von einem Satz gegebener Basisfunktionen aufgespannt wird. Diese Basisfunktionen werden üblicherweise über einem Netz definiert, das die Oberflächen der Szene in Teilflächen zerlegt. Der wohl bekannteste Vertreter dieser Vorgehensweise ist das klassische Radiosity-Verfahren

[29, 14]. Bei diesem Verfahren wird die Lichtverteilungsfunktion über stückweise konstanten Basisfunktionen repräsentiert. Stückweise konstante Basisfunktionen sind jedoch für die Visualisierung aufgrund der sichtbaren Facettierung, die zudem noch durch den Mach-Band-Effekt [24] verstärkt wird, nicht gut geeignet. Deshalb erfolgt vor der Bilderzeugung üblicherweise die Umwandlung der berechneten Approximation in eine andere Darstellung, die eine höhere visuelle Qualität der Bilder erreichen soll. Diese Umwandlung wird im Rahmen des Radiosity-Verfahrens als Rekonstruktion der Radiosity-Funktion bezeichnet [15].

In dieser Arbeit wird die Rekonstruktion als Interpolationsaufgabe betrachtet. Es zeigt sich, daß es sich dabei im allgemeinen um eine Streudateninterpolation handelt. Für die Streudateninterpolation sind in der Literatur eine Vielzahl von Verfahren bekannt. Es wird eine Reihe der Verfahren auf das Rekonstruktionsproblem übertragen. Die durchgeführten empirischen Untersuchungen zeigen, daß solche Verfahren tatsächlich zu visuell ansprechenden Bildern führen. Vorteil des hier gewählten Ansatzes ist, daß er, anders als etwa der Übergang zu anderen Basisfunktionen, in bestehende Radiosity-Programme integriert werden kann, ohne Änderungen im Berechnungsverfahren zu erfordern.

## 1.3 Testumgebung

Die Beurteilung der Qualität der in dieser Arbeit entwickelten Verfahren geschieht in großem Umfang empirisch. Hierfür wurde eine Testumgebung geschaffen, die flexibel änderbar und erweiterbar ist. Dies wird durch eine objektorientierte Software-Architektur erreicht. Ein weiterer wichtiger Aspekt, der bei der Entwicklung der Testumgebung beachtet wurde, ist die Vergleichbarkeit der rechnerischen Effizienz verschiedener Lösungsverfahren. Verschiedene Verfahren zur Bilderzeugung haben häufig Teilaufgaben gemeinsam. Bei unterschiedlicher Implementierung der Lösung der Teilaufgaben kann es zu Effizienzunterschieden kommen, die nicht dem Gesamtverfahren zuzuschreiben sind. Aus diesem Grund sind Algorithmen und Testumgebung so angelegt, daß nur in wirklich unterschiedlichen Teilen unterschiedliche Algorithmen und deren Implementierung zum Einsatz kommen.

Kern der Architektur sind zwei objektorientierte Bibliotheken. Die eine modelliert mathematische Konzepte, insbesondere die hierarchische Funktionsdarstellung in Wavelet-Basen und die Monte Carlo-Integration. Die zweite Bibliothek realisiert verschiedene Verfahren zur fotorealistischen Bilderzeugung.

## 1.4 Aufbau der Arbeit

Im folgenden Kapitel wird der neu entwickelte Ansatz zur Verwendung adaptiver Dichtefunktionen für Anwendungen der Monte Carlo-Integration vorgestellt. Der Ansatz wird anwendungsneutral als mathematisches Modell entwickelt, läßt

sich prinzipiell also auch für Anwendungen der Monte Carlo-Integration in anderen Gebieten außerhalb der Bildsynthese nutzen.

Im Kapitel 3 wird gezeigt, wie der neu entwickelte mathematische Ansatz zur Effizienzsteigerung von Monte Carlo-Ray Tracing eingesetzt werden kann. Der Einsatz erfolgt in einem auf Pfadverfolgung basierenden Ray Tracing-Verfahren. Das hieraus resultierende neue Verfahren bietet viele Freiheitsgrade. Ein wichtiger Bestandteil dieses Kapitels ist deshalb die Untersuchung, wie sich die Wahl der Parameter auf die Effizienz des Verfahrens auswirkt.

Das nächste Kapitel stellt dar, wie Verfahren der Streudateninterpolation zur Rekonstruktion der Radiosity-Funktion benutzt werden können. Die Bewertung der verschiedenen Verfahren erfolgt durch Anwendung auf Radiosity-Ergebnisse, die prototypische Lichtverteilungen über Flächen darstellen. Die Beurteilung der Qualität kann dann visuell und über ein numerisches Fehlermaß erfolgen.

Die objektorientierte Architektur der verwendeten Testumgebung wird in Kapitel 5 vorgestellt. Insbesondere wird hier gezeigt, mit welchen Mechanismen die Flexibilität und Erweiterbarkeit der Testumgebung erreicht werden.

Die ersten drei Anhänge enthalten eine Darstellung der in der Arbeit benötigten mathematischen Grundlagen. Anhang A erläutert die verwendeten stochastischen Grundbegriffe. Die aus dem Bereich der Linearen Algebra benutzten Begriffe, insbesondere der Funktionsapproximation, werden im folgenden Anhang B vorgestellt. Anhang C behandelt die Darstellung von Funktionen in hierarchischen Basen, insbesondere in Wavelet-Darstellung.

Die Anhänge E und F enthalten die Resultate der mit den in Kapitel 3 und 4 vorgestellten Verfahren berechneten Bilder. Anhang E beinhaltet die zur Beurteilung unterschiedlicher Parametrisierungen des Ray Tracing-Verfahrens durchgeführten Berechnungen. In Anhang F befinden sich die Ergebnisse verschiedener Radiosity-Rekonstruktionen. Aus technischen Gründen (Farbdruck) sind die Seiten der Anhänge D und E einseitig bedruckt.

# Kapitel 2

# Monte Carlo-Integration mit adaptiven hierarchischen Dichtefunktionen

In diesem Kapitel wird ein Ansatz zur Steigerung der Effizienz des Monte Carlo-Integrationsverfahrens vorgestellt. Das Verfahren arbeitet in mehreren Durchläufen. In jedem Durchlauf gelangt man als Nebenprodukt der Integration zu einer Approximation des Integranden. Aus der Approximation wird eine Wahrscheinlichkeitsdichte erzeugt, die im folgenden Durchlauf des Verfahrens eingesetzt wird.

In diesem Kapitel benutzte mathematische Konzepte werden in den Anhängen A bis C vorgestellt. Es handelt sich hierbei um stochastische Begriffe (Anhang A), Grundbegriffe der Linearen Algebra, insbesondere Funktionsapproximation (Anhang B) und die Darstellung von Funktionen in hierarchischen Wavelet-Basen (Anhang C).

## 2.1 Monte Carlo-Integration

Gegeben seien eine nach einer Dichte $p$ verteilte Zufallsvariable $X \sim p$ und eine Funktion $g : [0,1] \to \mathbb{R}$. Nach (A.8) (Anhang A, Seite 103) und dem Gesetz der großen Zahl (A.11) (Seite 104) läßt sich der Erwartungswert $E(g(X))$ abschätzen als [42]

$$E(g(X)) = \int_0^1 g(x)\, p(x)\, \mathrm{d}x \approx \frac{1}{m} \sum_{i=1}^{m} g(x_i) \tag{2.1}$$

wobei $x_i$ nach der Wahrscheinlichkeitsdichte $p$ verteilte Zufallszahlen sind. Hiermit bietet sich die Möglichkeit, ein Integral durch Ausführung einer Anzahl von Zufallsexperimenten näherungsweise zu berechnen. Dieses ist die Basis des Monte Carlo-Integrationsverfahrens. Die in (2.1) dargestellte Berechnung wird im folgenden auch als *1. Form der Monte Carlo-Integration* bezeichnet.

Für die Anwendung von (2.1) ist es erforderlich, daß der Integrand als Produkt einer Funktion $g$ mit einer Wahrscheinlichkeitsdichte $p$ vorliegt. Dies ist in der Praxis im allgemeinen nicht der Fall. Mit der Substitution

$$f(x) = g(x)\,p(x) \tag{2.2}$$

kann man die allgemein anwendbare Form

$$\begin{aligned} \int_0^1 f(x)\,\mathrm{d}x \quad &= \quad \int_0^1 g(x)\,p(x)\,\mathrm{d}x \\ &\overset{(2.1)}{\approx} \quad \frac{1}{m}\sum_{i=1}^{m} g(x_i) \\ &\overset{(2.2)}{=} \quad \frac{1}{m}\sum_{i=1}^{m} \frac{f(x_i)}{p(x_i)} \end{aligned}$$

herleiten. Die verwendete Wahrscheinlichkeitsdichte $p$ muß die Bedingung

$$f(x) \neq 0 \quad \Rightarrow \quad p(x) \neq 0$$

erfüllen, kann aber ansonsten beliebig gewählt werden. Insbesondere kann sie für $f(x) = 0$ beliebige Werte annehmen.

Ein beliebiges Integral $I$ läßt sich dann mit

$$I = \int_0^1 f(x)\,\mathrm{d}x \approx \frac{1}{m}\sum_{i=1}^{m} \frac{f(x_i)}{p(x_i)} \tag{2.3}$$

abschätzen. Dieser Ansatz wird im folgenden *2. Form der Monte Carlo-Integration* genannt.

### 2.1.1 Höherdimensionale Integration

Ein großer Vorteil des Monte Carlo-Verfahrens ist, daß es sich einfach auf höherdimensionale Integration verallgemeinern läßt [42, 73, 35, 65].

Gegeben sei eine Funktion $f : [0,1]^d \to \mathbb{R}$. Dann läßt sich das Integral annähern als

$$\int \cdots \int f(\mathbf{x})\,\mathrm{d}x_1 \,\ldots\, \mathrm{d}x_d \approx \frac{1}{m}\sum_{i=1}^{m} \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} \tag{2.4}$$

wobei die $\mathbf{x}_i \in [0,1]^d$ nach der $d$-dimensionalen Wahrscheinlichkeitsdichtefunktion („WDF“) $p(\mathbf{x})$ erzeugte *Zufallsvektoren* sind. Auch hier muß die WDF die Bedingung

$$f(\mathbf{x}) \neq 0 \quad \Rightarrow \quad p(\mathbf{x}) \neq 0$$

erfüllen.

## 2.2 Varianzminderung

Eine nach (2.1) oder (2.3) berechnete Schätzung ist selbst wiederum eine Zufallsvariable, deren Erwartungswert das zu berechnende Integral ist. Wenn man, wie im Falle der Monte Carlo-Integration, den Wert einer Zufallsvariablen als Ergebnis einer durchzuführenden Berechnung ansieht, quantifiziert die Standardabweichung $\sigma$ den zu erwartenden Fehler des Resultats. Primäres Ziel einer numerischen Berechnung ist die Erreichung eines möglichst geringen Fehlers, hier also die Verringerung der Standardabweichung.

In der mathematischen Literatur wird dieser Aspekt unter dem Begriff der Varianzminderung („Dispersionsverkleinerung“) behandelt. Aufgrund der Beziehung $\sigma = \sqrt{DX}$ zwischen Standardabweichung und Varianz führt eine Verringerung der Varianz gleichzeitig zu einer Verringerung der Standardabweichung.

Es sei angemerkt, daß eine geringere Varianz ein genaueres Resultat nicht garantiert. Da der Fehler der Berechnung wiederum eine Zufallsgröße mit der Standardabweichung als Erwartungswert ist, können auch große Abweichungen auftreten, werden mit geringerer Varianz aber immer unwahrscheinlicher. Dieser Zusammenhang wird durch die *Tschebyscheffsche Ungleichung* [6] quantifiziert:

$$Prob\left(|X - EX| \geq \varepsilon\right) \leq \frac{DX}{\varepsilon^2} \qquad \text{für } \varepsilon > 0$$

Im folgenden werden die drei in der Praxis wahrscheinlich am häufigsten angewandten Techniken zur Varianzminderung vorgestellt, die auch in dem in Kapitel 3 vorgestellten Verfahren eingesetzt werden. Für weitere Ansätze der Varianzminderung sei auf die mathematische Literatur verwiesen [30, 42, 73, 22, 35, 65], im Falle der Anwendung von Monte Carlo-Methoden zur fotorealistischen Bilderzeugung insbesondere auf die Arbeit von Lafortune [49].

### 2.2.1 Erhöhung der Samplezahl

Der Term $f(x_i)/p(x_i)$ wird als *primärer Schätzer* des Integrals bezeichnet, während der Mittelwert mehrerer primärer Schätzer

$$\frac{1}{m}\sum_{i=1}^{m}\frac{f(x_i)}{p(x_i)}$$

*sekundärer Schätzer* genannt wird [75]. Der sekundäre Schätzer ist besser als der primäre, weil er niedrigere Varianz hat. Bei unabhängig erzeugten identisch verteilten Zufallszahlen gilt nämlich [30, 42]:

$$D\left(\frac{1}{m}\sum_{i=1}^{m}\frac{f(x_i)}{p(x_i)}\right) = \frac{1}{m}D\left(\frac{f(x_i)}{p(x_i)}\right) \tag{2.5}$$

Die Standardabweichung gibt den zu erwartenden Fehler der Schätzung an. Da sie als Wurzel der Varianz berechnet wird, verhält sich damit der Fehler proportional zu $1/\sqrt{m}$. Dies ist ein ungünstiges Verhältnis, denn daraus folgt,

daß zur Halbierung des Fehlers die Zahl der Samples vervierfacht werden muß. Später gezogene Samples tragen immer weniger zur Verringerung des Fehlers bei.

Trotzdem ist für höherdimensionale Integrationsaufgaben der Monte Carlo-Ansatz oftmals ein gut geeignetes Berechnungsverfahren. Die Verallgemeinerung eines Quadraturverfahrens mit $k$ Stützstellen erfordert für eine $d$-dimensionale Integration $k^d$ Funktionsberechnungen, was für wachsendes $d$ schnell zu inakzeptablem Aufwand führt, während die Qualität der Monte Carlo-Berechnung aufgrund von (2.5) proportional zur Zahl der Samples aber unabhängig von der Dimension des Integrationsbereiches ist.

### 2.2.2 Importance Sampling

Die Varianz des primären Schätzers wird als

$$\begin{aligned} D\left(\frac{f(x)}{p(x)}\right) &\overset{(A.9)}{=} E\left(\left(\frac{f(x)}{p(x)}\right)^2\right) - \left(E\left(\frac{f(x)}{p(x)}\right)\right)^2 \\ &= \int_0^1 \left(\frac{f(x)}{p(x)}\right)^2 p(x)\,\mathrm{d}x - I^2 \\ &= \int_0^1 \frac{f^2(x)}{p(x)}\,\mathrm{d}x - I^2 \end{aligned} \tag{2.6}$$

berechnet. Varianzminderung durch Importance Sampling („Verfahren der wesentlichen Stichprobe“) basiert darauf, Dichtefunktionen zu suchen, die diesen Wert minimieren [30, 42, 73, 22, 35, 65]. Bei der Bestimmung des Minimums von (2.6) handelt es sich um ein Variationsproblem, für das sich in [42] eine auf Lagrangeschen Multiplikatoren beruhende Lösung findet. Diese besagt, daß die optimale Dichtefunktion $p_{opt}(x)$ eine normierte Version des absoluten Integranden ist:

$$p_{opt}(x) = \frac{|f(x)|}{\int_0^1 |f(t)|\,\mathrm{d}t}$$

Wenn man voraussetzt, daß der Integrand im gesamten Integrationsbereich nicht-negativ ist, vereinfacht sich dies zu:

$$p_{opt}(x) = \frac{f(x)}{I} \tag{2.7}$$

Da alle in dieser Arbeit auftretenden Integranden diese Bedingung erfüllen, wird im folgenden (2.7) benutzt, um die optimale Dichte zu definieren.

Setzt man die durch (2.7) bestimmte optimale Dichte in der 2. Form der Monte Carlo-Integration ein, erkennt man, daß jeder primäre Schätzer bereits das exakte Resultat liefert:

$$\frac{f(x_i)}{p_{opt}(x_i)} = \frac{f(x_i)}{\frac{f(x_i)}{I}} = I$$

Tatsächlich reduziert die optimale Dichte die Varianz des Verfahrens auf Null.

Die Benutzung der optimalen Dichte ist in der Praxis nicht möglich, da sie bereits die Kenntnis des zu berechnenden Wertes $I$ erfordert, so daß dann gar keine Integration mehr erforderlich wäre. Darüber hinaus ist in vielen Fällen keine geschlossene Darstellung des Integranden $f(x)$ bekannt, sondern wird nur für einzelne Samples $x$ durch Punktabtastung bestimmt. Es zeigt sich aber, daß auch eine Dichtefunktion, deren Verlauf nicht der optimalen Dichte entspricht, sondern diese nur annähert, zu einer deutlichen Verringerung der Varianz führt. Je besser die Dichtefunktion die optimale Dichte approximiert, desto stärker ist die Varianzminderung.

Verfahren zur Varianzminderung durch Importance Sampling basieren darauf, aus Approximationen des Integranden Dichtefunktionen zu gewinnen, deren Verwendung dann zu Monte Carlo-Verfahren mit geringerer Varianz führen [55, 48]. Die Approximationen können durch a priori-Kenntnisse des Integranden oder durch im Verlauf der Berechnung gewonnene Informationen generiert werden. Aus der Approximation $\hat{f}(x)$ wird die Dichtefunktion durch Normierung gewonnen

$$p(x) = \frac{\hat{f}(x)}{\int_0^1 \hat{f}(x)\,\mathrm{d}x} \tag{2.8}$$

wobei die Bedingung der Positivität (siehe A.1.1) gewährleistet werden muß.

### 2.2.3 Stratifikation

Eine andere häufig angewandte Methode zur Varianzminderung ist als *Stratifikation* („Verfahren der geschichteten Stichprobe“) bekannt [30, 42, 73, 22, 35, 65]. Hierbei wird der Integrationsbereich $\Omega$ in $k$ Teilbereiche $\Omega_j$ unterteilt. Das Integral kann dann als Summe der Integrale über die Teilbereiche dargestellt werden:

$$\int_\Omega f(x)\,\mathrm{d}x = \sum_{j=1}^{k} \int_{\Omega_j} f(t)\,\mathrm{d}t$$

Jedes Teilintegral kann dann mit einem Monte Carlo-Verfahren berechnet werden, wobei jedem Teilbereich eine eigene WDF $p_j$ zugewiesen werden kann. Es ist üblich, nur ein Sample pro Teilbereich zu verwenden, da es bei höherer Samplezahl effizienter ist, die Zahl der Teilbereiche zu erhöhen statt mehrere Samples pro Teilbereich zu verwenden. Die Varianz der Gesamtschätzung ergibt sich dann als Summe der Varianzen der Teilschätzungen:

$$D\left(\sum_{j=1}^{k} \frac{f(x_j)}{p_j(x_j)}\right) = \sum_{j=1}^{k} D\left(\frac{f(x_j)}{p_j(x_j)}\right) \tag{2.9}$$

wobei Sample $x_j$ nach der WDF $p_j$ erzeugt wird.

Die Varianzminderung durch Stratifikation resultiert daraus, daß die Varianz der Schätzer für die Teilintegrale oftmals signifikant kleiner ist als die Varianz

für einen Schätzer des Gesamtintegrals. Betrachtet man z. B. den Integrand $f(x) = \sin\left(\frac{\pi}{2}x\right)$, so hat ein Schätzer mit uniformer WDF für $\Omega = [0,1]$ eine Varianz von $\sigma^2 = 0.0947$. Unterteilt man den Integrationsbereich in zwei Teilbereiche $\Omega_1 = [0, 1/2]$ und $\Omega_2 = [1/2, 1]$, dann hat ein Schätzer in einem Teilbereich jeweils eine Varianz $\sigma_1^2 = 0.0107$ und $\sigma_2^2 = 0.0019$. Eine stratifizierte Monte Carlo-Integration mit zwei Samples hat damit nach (2.9) eine Varianz von $\sigma_{st}^2 = \sigma_1^2 + \sigma_2^2 = 0.0126$, während eine normale Monte Carlo-Integration mit zwei Samples nach (2.5) zu einer Varianz von $\sigma_{mc}^2 = \frac{1}{2}\sigma^2 = 0.0474$ führt. Dieser Effekt der Varianzminderung durch Stratifikation wird für höhere Samplezahlen noch stärker.

Stratifikation ist ein gutes Mittel, wenn Importance Sampling nicht einsetzbar ist, weil z. B. kein Wissen über den Verlauf des Integranden verfügbar ist. Der Einsatz von Stratifikation kann aber auch Nachteile haben. Während normale Monte Carlo-Integration nach einer beliebigen Samplezahl unterbrochen oder auch zur Verbesserung des Ergebnisses durch Berechnung zusätzlicher Samples verfeinert werden kann, ist man beim Einsatz von Stratifikation auf eine vorher fixierte Samplezahl $k$ festgelegt. Das Weglassen von Samples ist nicht zulässig, weil jedes einzelne Sample für ein bestimmtes Teilgebiet des Integrationsgebietes steht. Zusätzliche Samples lassen sich nur erzeugen, indem das Integrationsgebiet neu stratifiziert wird und dann aus beiden Berechnungen ein nach Samplezahl gewichteter Mittelwert gebildet wird. Die fixierte Samplezahl wird insbesondere für höherdimensionale Integration problematisch. Liegt ein $d$-dimensionaler Integrationsbereich vor, in dem jede Achse in $k$ Teilintervalle stratifiziert werden soll, so sind $k^d$ Samples nötig. Dies führt selbst für kleine $k$ schnell zu unakzeptablen Samplezahlen. Aus diesem Grund ist der Einsatz von Stratifikation für höherdimensionale Integration nicht üblich.

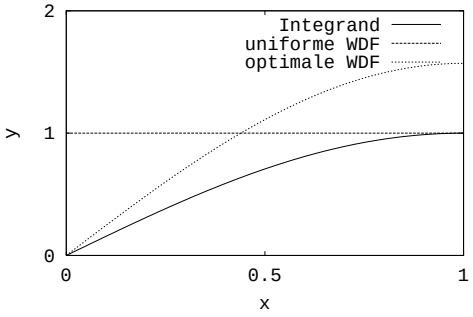
### 2.2.4 Beispiel

Die Auswirkungen der oben vorgestellten Ansätze zur Varianzminderung sollen an einem Beispiel verdeutlicht werden, um auch einen Eindruck zu gewinnen, welche quantitativen Verbesserungen erreichbar sind. Die Werte wurden auf der Grundlage von (2.6) mit dem Programm *Maple V* [34] berechnet. Es wurde die Varianz verschiedener Monte Carlo-Ansätze für die Berechnung des Integrals

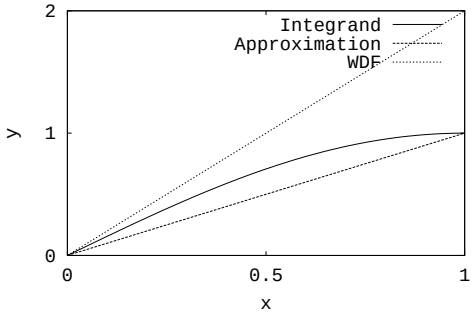
$$I = \int_0^1 \sin\left(\frac{\pi}{2}x\right)\,\mathrm{d}x = \frac{2}{\pi}$$

bestimmt.

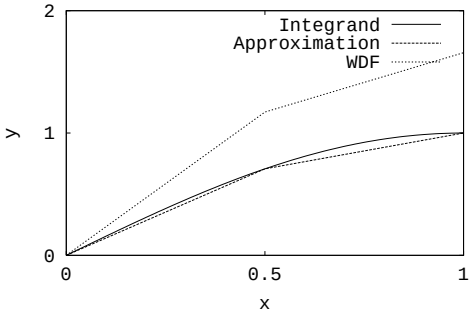
Als erster Ansatz für ein Importance Sampling wurde eine stückweise lineare Approximation des Integranden generiert, indem an den Stützstellen 0, 1/4, 1/2, 3/4 und 1 der Funktionswert des Integranden berechnet und mit 1, 2 bzw. 4 Segmenten stückweise linear interpoliert wurde (Abb. 2.1). Die so erhaltenen Approximationen müssen normiert werden, um Wahrscheinlichkeitsdichtefunktionen zu erhalten. Für die Normierung wird das Integral über die Approximation benötigt. Dieses ist in diesem Fall einfach zu berechnen, da die einzelnen



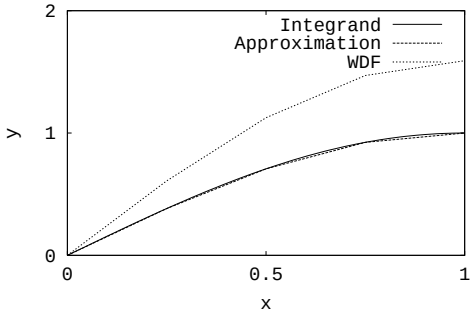

uniforme/optimale Dichte



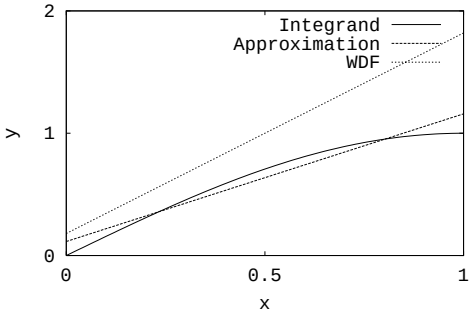

lin. Approximation,
1 Segment



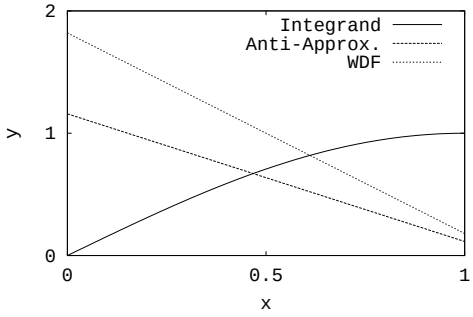

lin. Approximation,
2 Segmente




lin. Approximation,
4 Segmente




lin. $L^2$-Approximation




Anti-Approximation

Abbildung 2.1: Integrand $\sin\left(\frac{\pi}{2}x\right)$, Approximationsfunktion und Wahrscheinlichkeitsdichtefunktion (=normierte Approximationsfunktion) für unterschiedliche Approximationsansätze

| | Samples | | |
|---|---|---|---|
| | 1 | 2 | 4 |
| **uniforme Dichte** | 0.0947 | 0.0474 | 0.0237 |
| **Stratifikation** | 0.0947 | 0.0126 | 0.00160 |
| **Importance Sampling** | | | |
| lin. Appr., 1 Segment | 0.00678 | 0.00339 | 0.00170 |
| lin. Appr., 2 Segmente | 0.000267 | 0.000133 | 0.0000667 |
| lin. Appr., 4 Segmente | 0.0000143 | 0.00000716 | 0.00000358 |
| lin. $L^2$-Approximation | 0.00614 | 0.00307 | 0.00154 |
| lin. Anti-Approximation | 0.593 | 0.297 | 0.148 |
| **optimale Dichte** | 0 | 0 | 0 |

Tabelle 2.1: Varianz für Monte Carlo-Integration, Integrand $\sin\left(\frac{\pi}{2}x\right)$

Segmente der Approximation Trapezform haben. Die so berechneten Dichtefunktionen haben für $x = 0$ Nullstellen, sind aber zulässig, da der Integrand an dieser Stelle ebenfalls den Wert 0 annimmt. Tabelle 2.1 zeigt, daß mit zunehmender Qualität der Approximation die Varianz des Resultats deutlich sinkt.

Als klassischer Ansatz der Funktionsapproximation wurde eine lineare $L^2$-Approximation des Integranden berechnet und daraus wieder eine WDF bestimmt. Es zeigt sich, daß mit dieser WDF ebenfalls eine Varianzminderung erreichbar ist.

Um die Auswirkungen einer ungünstigen WDF zu untersuchen, wurde aus der $L^2$-Approximation eine „Anti-Approximation“ gebildet, indem sie an der Achse $x = 1/2$ gespiegelt wurde. Die daraus erhaltene WDF ist im Sinne des Importance Sampling eine schlechte WDF und führt im Gegensatz zu allen anderen Dichtefunktionen zu einer deutlich höheren Varianz.

Die optimale Dichte ist wie oben gesehen der skalierte Integrand

$$p_{opt}(x) = \frac{1}{I} \sin\left(\frac{\pi}{2}x\right)$$

Sie liefert für jede Samplezahl das korrekte Ergebnis und erreicht somit eine Varianz von 0. Die Dichte nimmt für $x = 0$ den Wert 0 an, ist aber zulässig, da der Integrand an dieser Stelle ebenfalls eine Nullstelle hat.

Zusammenfassend zeigt dieses Beispiel, daß Stratifikation und Importance Sampling zu deutlichen Varianzminderungen führen können, wobei Importance Sampling für gute Dichtefunktionen zu besseren Resultaten führt als die Stratifikation. Die ebenfalls benutzte Anti-Approximation ist ein Beispiel dafür, daß eine ungünstige Dichtefunktion die Qualität eines Monte Carlo-Verfahrens deutlich verschlechtern kann. Somit bleibt als zweite Erkenntnis festzuhalten, daß die Qualität eines auf Importance Sampling basierenden Monte Carlo-Verfahrens von der Güte der verwendeten Approximation abhängt.

## 2.3 Effizienz der Monte Carlo-Integration

Die Effizienz eines Integrationsverfahrens setzt Rechenzeit und Qualität des berechneten Ergebnisses in Beziehung zueinander. Ein Verfahren ist dann effizienter, wenn es

- in gleicher Zeit ein Ergebnis besserer Qualität bzw.
- in kürzerer Zeit ein Ergebnis gleicher Qualität

berechnet. Die Qualität eines Verfahrens wird als der zu erwartende Fehler gemessen, im Falle der Monte Carlo-Integration ist es üblich, die Varianz als Qualitätsmaß zu verwenden.

Kalos und Whitlock [42] stellen fest, daß die Größe

$$Q_1 := t_1 \, \sigma_1^2$$

ein Effizienzmaß für Monte Carlo-Verfahren darstellt, wobei $t_1$ die Rechenzeit des Verfahrens für ein Sample und $\sigma_1^2$ die Varianz dieser Ein-Sample-Monte Carlo-Berechnung ist. Bessere Monte Carlo-Algorithmen führen in der Regel zu einem kleineren $\sigma_1$ benötigen aber auch mehr Rechenzeit. Die Frage ist dann, ob die Steigerung der Rechenzeit durch die Erhöhung der Qualität kompensiert wird. Das ist genau dann der Fall, wenn das Maß $Q_1$ eines gegebene Verfahrens kleiner ist als das eines Vergleichsverfahrens.

Arvo und Kirk [1] benutzen im wesentlichen dasselbe Effizienzmaß, aber sie verwenden den Kehrwert

$$e_1 := \frac{1}{t_1 \, \sigma_1^2}, \tag{2.10}$$

so daß in ihrem Maß der Algorithmus mit dem höchsten Wert der effizienteste ist.

Die Effizienzmaße von Kalos/Whitlock und Arvo/Kirk sind für Ein-Sample-Monte Carlo-Verfahren, also nur für primäre Schätzer definiert. Ein Verfahren mit $m$ Samples wird $m\,t_1$ Rechenzeit benötigen und unter der Annahme, daß (2.5) gilt, zu einer Varianz $\sigma^2 = \frac{1}{m}\sigma_1^2$ führen. Dann ist das Effizienzmaß

$$\begin{aligned} e_m &:= \frac{1}{t \, \sigma^2} \\ &= \frac{1}{m \, t_1 \, \frac{1}{m} \, \sigma_1^2} \\ &= \frac{1}{t_1 \, \sigma_1^2} \\ &= e_1 \end{aligned} \tag{2.11}$$

eine von der Zahl der Samples unabhängige Konstante und identisch mit dem von Arvo/Kirk definierten Effizienzmaß.

Es sei angemerkt, daß für einige Techniken zur Varianzminderung wie z. B. stratifizierte Sample-Generierung die Beziehung (2.5) nicht gilt und die durch

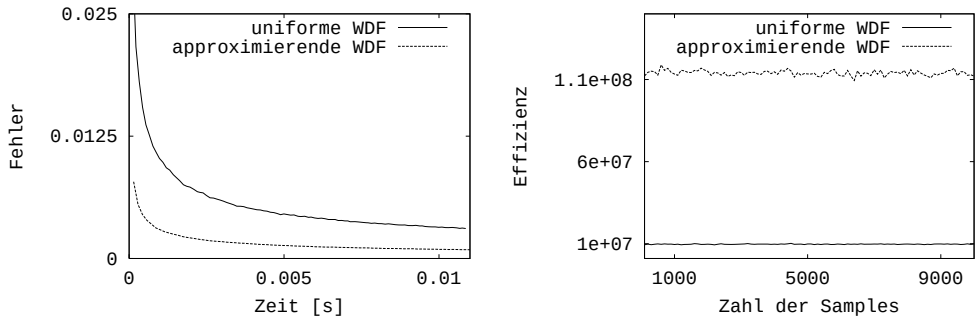



Abbildung 2.2: Fehler (links) und Effizienz (rechts) der Monte Carlo-Integration von $\sin\left(\frac{\pi}{2}x\right)$

(2.11) definierte Effizienz im allgemeinen keine Konstante ist. In diesen Fällen kann (2.11) nicht als von der Samplezahl unabhängiges Effizienzmaß benutzt werden.

### 2.3.1 Beispiel

Ein praktisches Beispiel soll die Eigenschaften des Effizienzmaßes (2.11) zeigen. Das Integral

$$\int_0^1 \sin\left(\frac{\pi}{2}x\right)\,\mathrm{d}x = \frac{2}{\pi}$$

wurde mit Monte Carlo-Integration berechnet. Dazu wurden eine uniforme und eine lineare WDF benutzt. Bei der linearen WDF handelt es sich um die aus Beispiel 2.2.4 bekannte aus der $L^2$-Approximation generierte WDF. Für unterschiedliche Samplezahlen — nämlich 100, 200, 300, ..., 10000 — wurden die Rechenzeiten und Schätzungen der Varianz bestimmt, indem für jede Samplezahl 10000 Monte Carlo-Berechnungen durchgeführt wurden.

Die Ergebnisse dieser Versuche sind in Abb. 2.2 dargestellt. Das linke Diagramm zeigt die Zeit-Fehler-Kurven für die benutzten Dichtefunktionen. Aus dem Diagramm ist erkennbar, daß die aus der $L^2$-Approximation gewonnene WDF zum effizienteren Monte Carlo-Verfahren führt, da das Ergebnis für eine gegebene Zeit einen geringeren Fehler aufweist. Das rechte Diagramm zeigt die nach (2.11) berechnete Effizienz. Es ist zwar eine gewisse aus der stochastischen Natur des Verfahrens resultierende Schwankung erkennbar, die Kurven belegen aber hinreichend, daß die so berechnete Effizienz tatsächlich eine von der Samplezahl unabhängige Konstante ist. Außerdem zeigt das Effizienzmaß wie erwartet, daß die Benutzung der approximierenden WDF das effizientere Verfahren ergibt.

Derselbe Versuch wurde für stratifizierte Samplegenerierung durchgeführt. Aus Abb. 2.3 wird erkennbar, daß in diesem Fall die durch (2.11) definierte Effizienz

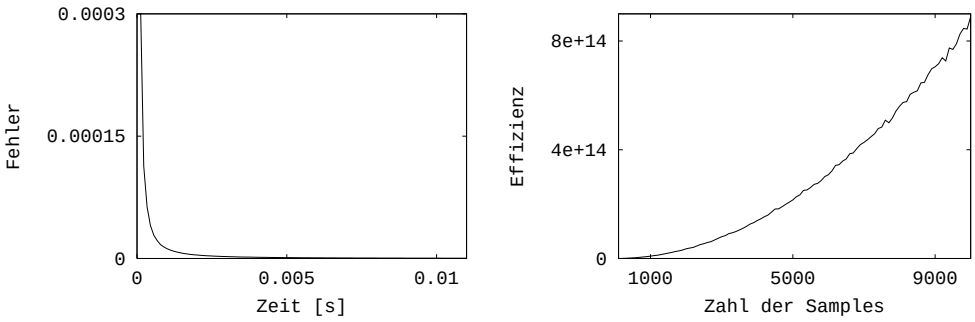



Abbildung 2.3: Fehler (links) und Effizienz (rechts) der Monte Carlo-Integration von $\sin\left(\frac{\pi}{2}x\right)$ mit stratifizierter Sample-Generierung

tatsächlich keine Konstante ist.

## 2.4 Adaptive Dichtefunktionen

In diesem Abschnitt wird ein Verfahren vorgestellt, das aus den Samples einer Monte Carlo-Integration eine Approximation $\hat{f}(x)$ des Integranden $f(x)$ konstruiert. Aus dieser Approximation $\hat{f}$ kann dann eine neue Wahrscheinlichkeitsdichte $p^+$ erzeugt werden, die in weiteren Durchgängen der Monte Carlo-Integration genutzt werden kann. Aus Abschnitt 2.2.2 ist bekannt, daß eine Dichtefunktion, die aus einer möglichst genauen Approximation des Integranden gewonnen wird, zu einer Varianzminderung führt. Deshalb ist zu erwarten, daß mit der neu generierten Wahrscheinlichkeitsdichte $p^+$ der folgende Durchgang der Monte Carlo-Integration eine geringere Varianz des Ergebnisses aufweist.

### 2.4.1 Dichtefunktionen in Basisdarstellung

Sei $N = \{N_j(x) \in L^2(0,1), j = 1,\ldots,n\}$ eine Funktionsbasis deren Basisfunktionen als skalierte Wahrscheinlichkeitsdichten $p_j(x)$ geschrieben werden können:

$$\begin{aligned} N_j(x) &= w_j\, p_j(x) \\ \int_0^1 p_j(x)\,\mathrm{d}x &= 1 \end{aligned} \tag{2.12}$$

Dies gilt für alle Basen mit nicht-negativen Basisfunktionen, z. B. für alle B-Spline-Basen. Die Skalierungsfaktoren $w_j$ erhält man als

$$w_j = \int_0^1 N_j(x)\,\mathrm{d}x \tag{2.13}$$

Sei $p(x)$ eine in dieser Basis $N$ dargestellte Wahrscheinlichkeitsdichte:

$$p(x) = \sum_{j=1}^{n} c_j \, N_j(x)$$

Außerdem sei vorausgesetzt, daß Verfahren gegeben sind, Zufallszahlen entsprechend den Dichtefunktionen $p_j$ zu erzeugen. Dann ist es möglich, eine Zufallszahl $x_i \sim p$ zu erzeugen, indem erst der Index $j$ einer Basisfunktion ausgewählt und dann eine Zufallszahl gemäß der Dichte $p_j$ generiert wird [30, 42]. Die Auswahl des Index $j$ hat so zu erfolgen, daß für die Wahrscheinlichkeit

$$Prob(\text{Wahl von } j) = \frac{c_j \, w_j}{\sum\limits_{k=1}^{n} c_k \, w_k} \tag{2.14}$$

gilt [42]. Die Auswahl von $j$ erfolgt also nach einer diskreten, durch (2.14) definierten Wahrscheinlichkeitsdichte.

So erhält man für jeden Index $j$ eine Menge $S^j$ von Zufallszahlen

$$S^j = \{x_k^j, k = 1, \ldots, n_j\}$$

mit $x_k^j \sim p_j$. Die Vereinigung dieser Mengen

$$S = \cup S^j$$

ergibt die Menge $S$ von Zufallszahlen $x_i \sim p$, die gemäß Gleichung (2.3) dazu benutzt werden kann, eine Schätzung des gesuchten Integrals zu berechnen:

$$\int_0^1 f(x) \, \mathrm{d}x \approx I_{est} = \frac{1}{m} \sum_{j=1}^{n} \sum_{k=1}^{n_j} \frac{f(x_k^j)}{p(x_k^j)} \tag{2.15}$$

### 2.4.2 Approximation des Integranden

In den für die Berechnung von (2.15) benötigten Funktionswerten $f(x_k^j)$ ist wertvolle Information über den Funktionsverlauf des Integranden enthalten. Grundidee des hier vorgestellten neuen Verfahrens zur Monte Carlo-Integration mit adaptiven Dichtefunktionen ist, diese Information zu nutzen, um daraus eine Approximation des Integranden zu konstruieren.

In diesem Abschnitt wird gezeigt, wie aus den Samplemengen $S^j$ mit zugehörigen Funktionswerten $f(x_k^j)$ eine $L^2$-Approximation

$$\hat{f} = \sum_{j=1}^{n} c_j^\star \, N_j(x)$$

des Integranden $f$ in der Basis $N$ berechnet werden kann.

Im folgenden wird vorausgesetzt, daß der Integrand $f(x)$ im Integrationsbereich keine negativen Werte annimmt. Das vorgestellte Verfahren ist auch anwendbar, wenn diese Bedingung nicht erfüllt ist. In diesem Fall ist eine Approximation des absoluten Integranden $|f(x)|$ zu berechnen. Auf die dann notwendigen Unterschiede zum vorgestellten Verfahren wird an entsprechender Stelle hingewiesen.

#### 2.4.2.1 Orthonormale Basis

Für eine orthonormale Basis $N$ werden die Koeffizienten $c_j^\star$ nach (B.1) (Anhang B, Seite 114) als

$$\begin{aligned} c_j^\star &\overset{(B.1)}{=} \langle f, N_j \rangle \\ &= \int_0^1 f(x)\, N_j(x)\, \mathrm{d}x \\ &\overset{(2.12)}{=} \int_0^1 f(x)\, w_j\, p_j(x)\, \mathrm{d}x \end{aligned} \tag{2.16}$$

berechnet. Das Integral (2.16) hat die geeignete Struktur, um mit der 1. Form der Monte Carlo-Integration (2.1) berechnet zu werden:

$$\begin{aligned} c_j^\star &\overset{(2.1)}{\approx} \frac{1}{n_j} \sum_{k=1}^{n_j} w_j\, f(x_k^j) \\ &= \frac{w_j}{n_j} \sum_{k=1}^{n_j} f(x_k^j) \end{aligned} \tag{2.17}$$

Nimmt der Integrand auch negative Werte an, ist $f(x_k^j)$ in (2.17) durch den Absolutwert $\left|f(x_k^j)\right|$ zu ersetzen.

Wie man sieht, werden die nach $p_j$ verteilten Samples der Menge $S^j$ dazu benutzt, eine Monte Carlo-Schätzung des zur Basisfunktion $j$ gehörenden Koeffizienten $c_j^\star$ zu erhalten. Da keiner der in (2.17) auftretenden Terme einen negativen Wert annimmt, kann auch keiner der berechneten Koeffizienten $c_j^\star$ negativ werden.

Es sei darauf hingewiesen, daß die berechneten Koeffizienten $c_j^\star$ stochastisch bestimmte Näherungen der Koeffizienten des tatsächlichen Proximums sind. Das Gesetz der großen Zahl besagt aber, daß die so bestimmte Näherung mit steigender Samplezahl gegen das tatsächliche Proximum konvergiert.

#### 2.4.2.2 Nicht-orthonormale Basis

Ist die Basis $N$ nicht orthonormal, läßt sich nach (B.3) (Anhang B, Seite 115) eine Darstellung der $L^2$-Approximation in der dualen Basis $\tilde{N}$ darstellen:

$$\begin{aligned} \tilde{c}_j^\star &\overset{(B.3)}{=} \langle f, N_j \rangle \\ &\approx \frac{w_j}{n_j} \sum_{k=1}^{n_j} f(x_k^j) \end{aligned} \tag{2.18}$$

Falls der Integrand negative Werte annehmen kann, muß er auch hier durch den Absolutwert ersetzt werden.

Ein Basiswechsel liefert die Koeffizienten der Approximation in der primären Basis:

$$\mathbf{c}_j^\star = \mathbf{M}_{\tilde{N}\to N}\tilde{\mathbf{c}}_j^\star \tag{2.19}$$

Die in (2.18) auftretenden Terme sind ebenfalls nicht-negativ. Daraus folgt zwar, daß die Koeffizienten $\tilde{c}_j^\star$ ebenfalls nicht-negativ sind. Diese Eigenschaft kann aber durch den Wechsel in die primäre Basis verloren gehen, so daß eventuell negative Koeffizienten $c_j^\star$ auftreten.

#### 2.4.2.3 Bestimmung fehlender Koeffizienten

Die Verteilung von Samples auf die Basisfunktionen erfolgt nach der in (2.14) bestimmten Wahrscheinlichkeit. Bei geringer Samplezahl kann es passieren, daß auf eine Basisfunktion $j$ gar keine Samples entfallen, eine Schätzung des Koeffizienten nach (2.17) bzw. (2.18) ist dann natürlich nicht möglich. Um trotzdem zu einem Wert für den Koeffizienten zu gelangen, bieten sich verschiedene Möglichkeiten:

**zusätzliche Samples** Es können ein oder mehrere zusätzliche Samples $x_k^j \sim p_j$ generiert werden, um (2.17) bzw. (2.18) doch anzuwenden. Das führt aber, insbesondere wegen der dann notwendig werdenden Funktionsberechnungen $f(x_k^j)$, zu zusätzlichem Aufwand, der keinen Beitrag zur Berechnung des Integrals liefert.

**lokaler Ansatz** Der Koeffizient kann als Mittelwert aus benachbarten Koeffizienten berechnet werden. Dieser Ansatz ist sinnvoll, wenn von lokaler Stetigkeit des Integranden ausgegangen werden kann. Sind benachbarte Koeffizienten mangels Samples ebenfalls unbestimmt, muß die Suche auf deren Nachbarn fortgesetzt werden.

**globaler Ansatz** Der Koeffizient kann aus dem errechneten Integral $I_{est}$ geschätzt werden. Setzt man den Integranden als konstant $f_{est}(x) = I_{est}$ an, erhält man für eine orthonormale Basis den Koeffizient

$$\begin{aligned} c_j^\star &= \langle f_{est}, N_j\rangle \\ &= \int_0^1 I_{est}\, N_j(x)\,\mathrm{d}x \\ &\overset{(2.13)}{=} I_{est}\, w_j \end{aligned}$$

bzw. für eine nicht-orthonormale Basis

$$\tilde{c}_j^\star = I_{est}\, w_j.$$

Dieser Ansatz sollte gewählt werden, wenn man keine lokale Stetigkeit des Integranden voraussetzt.

### 2.4.3 Verfeinerte Dichtefunktion

Aus der berechneten Approximation $\hat{f}$ soll jetzt eine verfeinerte Dichtefunktion $p^+$ gewonnen werden. Dabei ist auf die Einhaltung der in Abschnitt A.1.1 genannten Eigenschaften einer Dichtefunktion zu achten. Da im Falle einer nichtorthonormalen Basis auch negative Koeffizienten auftreten können, erfolgt erst eine Korrektur der Koeffizienten. Eventuell will man auch einen Sicherheitssockel einfügen, um zu verhindern, daß in bestimmten Bereiche des Integrationsgebietes keine Samples mehr generiert werden. Aus der so korrigierten Approximation wird dann durch Normierung eine Dichtefunktion erzeugt. Die Approximation wird dadurch nur geringfügig verändert, da die Zahl der zu korrigierenden Koeffizienten niedrig und deren absoluter Wert im Vergleich zu den anderen Koeffizienten klein ist.

#### 2.4.3.1 Korrektur negativer Koeffizienten

Um aus der Approximation $\hat{f}$ durch Normierung eine WDF gewinnen zu können, muß gewährleistet sein, daß $\hat{f}$ eine nicht-negative Funktion ist. Aufgrund der Voraussetzung in Abschnitt 2.4.1 sind die Basisfunktionen nichtnegative Funktionen. Nur bei Auftreten eines negativen Koeffizienten kann es passieren, daß $\hat{f}$ negative Werte annimmt. Dies kann verhindert werden, indem alle negativen Koeffizienten zumindest auf den Wert 0 angehoben werden.

#### 2.4.3.2 Sicherheitssockel

Aufgrund der stochastischen Bestimmung der Koeffizienten ist es möglich, daß einzelne Koeffizienten zu klein geschätzt werden. Wird aus dieser Schätzung eine WDF berechnet, die als Grundlage weiterer Durchläufe des Verfahrens dient, kann dies dazu führen, daß in diesem Bereich wenige oder gar keine Samples generiert werden. Dann wird eventuell das Integral falsch berechnet und die Fehlschätzung der Koeffizienten nicht mehr korrigiert. Um dies zu verhindern, kann es sinnvoll sein, einen gewissen Sicherheitssockel einzuführen, indem man alle Koeffizienten unterhalb einer gewissen Schwelle bis zu dieser Schwelle anhebt. Es ist sinnvoll, diese Schwelle am globalen Funktionsverlauf zu orientieren:

$$\text{wenn } c_j^\star < \tau\, I_{est}\, w_j, \text{ dann setze } c_j^\star := \tau\, I_{est}\, w_j$$

Der Faktor $\tau > 0$ legt die Höhe dieses Sicherheitssockels fest. Ein hoher Wert schützt besser gegen die Fehlschätzung von Koeffizienten, verschlechtert aber eventuell die erreichbare Qualität, da die Dichtefunktion in den Bereichen, in denen der Integrand klein wird oder tatsächlich auf 0 fällt, nicht nach unten folgen kann. In der Praxis hat sich ein Wert $\tau = 1/10$ gut bewährt.

#### 2.4.3.3 Normierung

Nach der Korrektur muß $\hat{f}$ normiert werden, um die Dichtefunktion $p^+$ zu erhalten. Das dafür benötigte Integral ist leicht als

$$\begin{aligned}\int_0^1 \hat{f}(x)\,\mathrm{d}x &= \int_0^1 \left(\sum_{j=1}^n c_j^\star\, N_j(x)\right)\mathrm{d}x \\ &= \sum_{j=1}^n c_j^\star \int_0^1 N_j(x)\,\mathrm{d}x \\ &\overset{(2.12)}{=} \sum_{j=1}^n c_j^\star\, w_j\end{aligned}$$

zu berechnen. Die somit erhaltene Dichtefunktion

$$p^+(x) = \sum_{j=1}^n \frac{c_j^\star}{\sum_{k=1}^n c_k^\star\, w_k} N_j(x) \tag{2.20}$$

kann dann in einem neuen Monte Carlo-Durchlauf als neue Dichtefunktion verwendet werden.

### 2.4.4 Beispiel

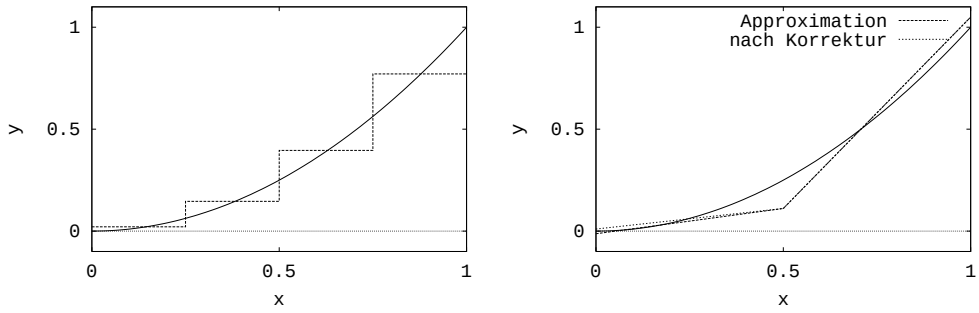
Ein Beispiel soll anschaulich die Funktionsweise des hier vorgestellten adaptiven Verfahrens demonstrieren. Als Integrand wurde die Funktion $f(x) = x^2$ benutzt. Bei Monte Carlo-Integration mit uniformer Dichte beträgt die Varianz des primären Schätzers $\sigma_u^2 = 0.00222$. Beginnend mit uniformer Dichte wurde das adaptive Verfahren mit jeweils 40 Samples für eine orthonormale und eine nicht-orthonormale Basis durchgeführt.

#### 2.4.4.1 Orthonormale Basis

Als Beispiel für eine orthonormale Basis wurde die Basis $\Phi^2$ einer Haar-MSA benutzt. Es handelt sich hierbei um eine Basis für den Raum der stückweise konstanten Funktionen über den Intervallen $[0, 1/4]$, $[1/4, 1/2]$, $[1/2, 3/4]$, $[3/4, 1]$.

Die nach dem oben beschriebenen Verfahren durchgeführte adaptive Monte Carlo-Integration führt dazu, daß 8 Samples für die erste Basisfunktion, 13 für die zweite, 10 für die dritte und 9 für die vierte erzeugt werden. Das entspricht ziemlich gut dem nach (2.14) errechneten Erwartungswert von 10 Samples pro Basisfunktion. Nach (2.17) erhält man aus diesen Samples die Koeffizienten der Approximation:

$$\mathbf{c}^\star = \begin{bmatrix} 0.00825 \\ 0.0644 \\ 0.191 \\ 0.390 \end{bmatrix}$$




stückweise konstant stückweise linear

Abbildung 2.4: Approximation des Integranden $f(x) = x^2$

Das linke Diagramm von Abb. 2.4 zeigt den Verlauf der Approximation. Obwohl hier eine relativ geringe Samplezahl benutzt wurde, ist diese Approximation schon nahe am exakten (mit *Maple V* berechneten) Proximum:

$$\mathbf{c}_{prox} = \begin{bmatrix} 1/96 \\ 7/96 \\ 19/96 \\ 37/96 \end{bmatrix} \approx \begin{bmatrix} 0.0104 \\ 0.0729 \\ 0.198 \\ 0.385 \end{bmatrix}$$

Die aus der Approximation erhaltene Dichtefunktion führt zu einer Varianz $\sigma_o^2 = 0.000182$ des primären Schätzers. Die aus der Adaption erhaltene Dichtefunktion verringert also die Varianz um mehr als den Faktor 12.

#### 2.4.4.2 Nicht-Orthonormale Basis

Als nicht-orthonormale Basis wurde die Basis $\Phi^1$ einer linearen B-Spline-MSA mit Knotenvektor $[0, 0, 1/2, 1, 1]$ verwendet.

Aus der ersten Basisfunktion werden 8 Samples, aus der zweiten 23 und aus der dritten 9 generiert. Aufgrund (2.14) ergibt sich für die erste und dritte Basisfunktion ein Erwartungswert von 10, für die zweite Basisfunktion ein Erwartungswert von 20 Samples, was gut mit der Beobachtung übereinstimmt. Aus diesen Samples erhält man nach (2.18) die Koeffizienten einer Approximation in der dualen Basis $\tilde{\Phi}^1$:

$$\tilde{\mathbf{c}}^\star = \begin{bmatrix} 0.00707 \\ 0.123 \\ 0.184 \end{bmatrix}$$

Auch hier ist trotz der geringen Samplezahl bereits eine gute Übereinstimmung mit dem exakten Proximum feststellbar:

$$\tilde{\mathbf{c}}_{prox} = \begin{bmatrix} 1/96 \\ 7/48 \\ 17/96 \end{bmatrix} \approx \begin{bmatrix} 0.0104 \\ 0.146 \\ 0.177 \end{bmatrix}$$

Da die neue Dichtefunktion in der primären Basis $\Phi^1$ dargestellt werden soll, wird danach ein Basiswechsel durchgeführt:

$$\begin{aligned}
\mathbf{c}^\star &= \mathbf{M}_{\tilde{\Phi}^1 \to \Phi^1} \tilde{\mathbf{c}}^\star \\
&= \begin{bmatrix} 7 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 7 \end{bmatrix} \begin{bmatrix} 0.00707 \\ 0.123 \\ 0.184 \end{bmatrix} \\
&= \begin{bmatrix} -0.0131 \\ 0.111 \\ 1.05 \end{bmatrix}
\end{aligned}$$

Wie zu erkennen ist, tritt hier tatsächlich nach dem Wechsel in die primäre Basis ein negativer Koeffizient auf, so daß eine Korrektur des Koeffizienten nötig wird. Setzt man einen Schwellwert von $\tau = 1/100$, erhält man als endgültige Approximation:

$$\mathbf{c}^\star = \begin{bmatrix} 0.01 \\ 0.111 \\ 1.05 \end{bmatrix}$$

Der Verlauf der Approximation vor und nach der Korrektur ist im rechten Diagramm von Abb. 2.4 dargestellt. Hieraus wird durch Normierung eine Dichtefunktion gewonnen, die zu einer Varianz $\sigma^2_{no} = 0.000173$ führt. Auch hier verringert sich die Varianz um mehr als den Faktor 12.

### 2.4.5 Hinweise zur Implementierung

Aus der Beschreibung des Verfahrens kann man den Eindruck gewinnen, daß es nötig ist, alle Samples $x^j_k$ mit ihren Funktionswerten $f(x^j_k)$ zu speichern, um (2.17) bzw. (2.18) berechnen zu können. Es reicht aber, für jede Basisfunktion $j$ einen Speicher zur Summenbildung und einen Zähler $n_j$ vorzusehen. Während der Berechnung wird jeder Funktionswert sofort zum zugehörigen Speicher addiert und der Zähler inkrementiert. Zum Schluß muß dann nur noch der Speicher mit $w_j/n_j$ multipliziert werden.

Hier zeigt sich auch, daß das Verfahren wenig zusätzlichen Speicher ($n$ Summen und $n$ Zähler) und wenig zusätzlichen Aufwand (eine Addition und eine Inkrementierung) während der Berechnung erfordert. Am Ende jedes Durchlaufs sind die Korrektur der Koeffizienten und die Normierung nach (2.20) zu berechnen sowie für nicht-orthonormale Basen zusätzlich ein Basiswechsel.

Werden mehrere adaptive Durchläufe des Verfahrens durchgeführt, kann man die Schätzung der Koeffizienten verbessern, indem man nicht jeweils eine neue Approximation berechnet, sondern alte und neue Schätzung nach Samplezahl gewichtet miteinander verrechnet. Für eine orthonormale Basis erfolgt die Berechnung mit

$$c^\star_j = \frac{n^{alt}_j \, c^{\star alt}_j + w_j \sum_{k=1}^{n_j} f(x^j_k)}{n^{alt}_j + n_j}$$

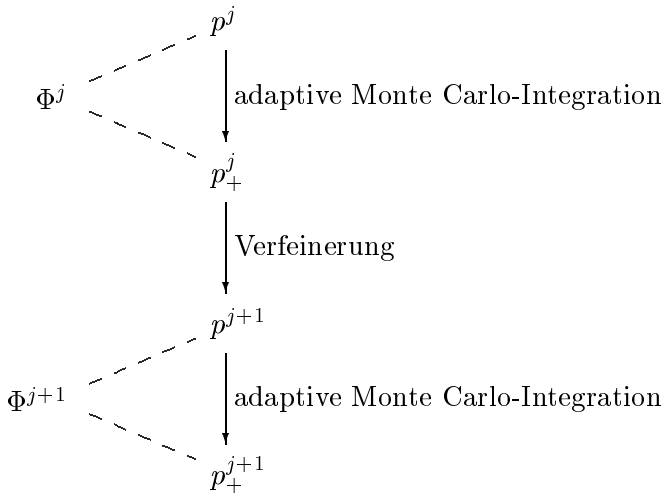



Abbildung 2.5: Hierarchische Adaption der Dichtefunktion

statt (2.17). Für eine nicht-orthonormale Basis wird (2.18) durch

$$\tilde{c}_j^\star = \frac{n_j^{alt}\,\tilde{c}_j^{\star alt} + w_j \sum_{k=1}^{n_j} f(x_k^j)}{n_j^{alt} + n_j}$$

ersetzt.

## 2.5 Adaptive hierarchische Dichtefunktionen

Das bis jetzt vorgestellte Verfahren erlaubt eine Adaption der Dichtefunktion innerhalb einer gegebenen Basis $N$. Benutzt man als Basis die Skalierungs-Basis einer Multi-Skalen-Analyse, kann man zu einer hierarchischen Adaption der Dichtefunktion gelangen, indem man nach jeder Adaption eine Verfeinerung der Darstellung auf die nächstfeinere Skala der MSA vornimmt.

Wenn man auf Skala $j$ der MSA startet, wird die Dichtefunktion $p^j(x)$ in der Basis $\Phi^j$ dargestellt. Nach dem oben beschriebenen adaptiven Monte Carlo-Verfahren gelangt man nach dem ersten Durchlauf zu einer verfeinerten Dichte $p_+^j(x)$. Vor dem nächsten Durchlauf des Monte Carlo-Verfahrens wird diese Dichtefunktion auf die Skala $j+1$ der MSA verfeinert, man erhält dann die Dichte $p^{j+1}(x)$ in der Basis $\Phi^{j+1}$. Diese Verfeinerung ist ein Basiswechsel, der Funktionsverlauf wird hierbei nicht verändert. Im nächsten Durchlauf der Monte Carlo-Integration wird nun auf der feineren Skala der MSA eine neue Dichte $p_+^{j+1}$ adaptiert. Dieser Ansatz kann über mehrere Skalen der MSA fortgesetzt werden. Abb. 2.5 veranschaulicht die Vorgehensweise. Die gestrichelten Verbindungen zeigen, in welcher Basis die entsprechende Dichtefunktion repräsentiert wird.

Die Verfeinerung der Funktion $p_+^j(x)$ entspricht einem Syntheseschritt ohne Detail-Koeffizienten. Sei $\mathbf{c}^{j+}$ der nach (2.20) berechnete Koeffizientenvektor von

$p_+^j(x)$ in der Basis $\Phi^j$, dann erhält man den Koeffizientenvektor von $p^{j+1}(x)$ in der Basis $\Phi^{j+1}$ als:

$$\mathbf{c}^{j+1} = \mathbf{P}^{j+1}\,\mathbf{c}^{j+}$$

### 2.5.1 Verteilung der Samples auf die Durchläufe

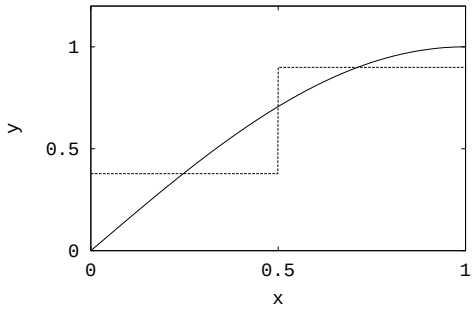
Das vorgestellte Verfahren arbeitet in mehreren Durchläufen, wobei sich die WDF von Durchlauf zu Durchlauf auf immer feineren Skalen der MSA adaptiv an den Integranden annähert. Jeder Durchlauf liefert eine Schätzung des zu bestimmenden Integrals. Diese Schätzungen werden gemäß der in jedem Durchlauf benutzten Samplezahl zu einer Gesamtschätzung verrechnet. Es ist nun zu klären, wie die Samples auf die einzelnen Durchläufe des Verfahrens verteilt werden sollten. Da spätere Durchläufe mit einer feiner adaptierten WDF arbeiten, ist zu erwarten, daß in diesen Durchläufen berechnete Werte eine geringere Varianz aufweisen. Deshalb sollte man möglichst wenige Samples in den ersten Durchläufen des Verfahrens einsetzen und möglichst viele Samples in späteren Durchläufen. Eine Einschränkung dieses Prinzips ergibt sich dadurch, daß bei einer zu geringen Samplezahl, die nach (2.17) bzw. (2.18) geschätzten Koeffizienten mit zu großem Fehler behaftet sind. Das kann zu schlechter Approximationsqualität und dann, wie im auch im Beispiel 2.2.4 gesehen, zu einer Verschlechterung der Varianz führen. Aus diesen Überlegungen resultiert als Prinzip für die Verteilung der Samples auf die Durchläufe des Verfahrens, daß in frühen Durchläufen möglichst wenige Samples eingesetzt werden sollten, aber so viele, daß die Adaption eine gute Approximation des Integranden liefert. Die größte Zahl der Samples sollte in späteren Durchläufen eingesetzt werden, da diese Samples dann von der besseren Qualität der adaptierten WDF profitieren. Welche Sampleverteilung zu guten Ergebnissen führt, ist abhängig vom Anwendungsfall und muß eventuell durch Versuche bestimmt werden.

### 2.5.2 Beispiel

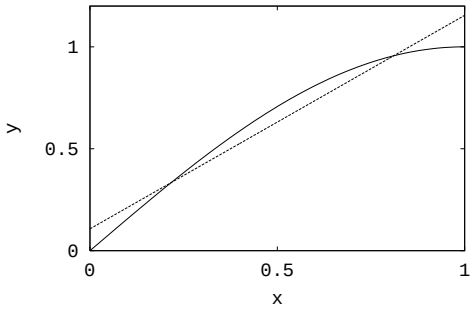
Das Verfahren zur Monte Carlo-Integration mit adaptiven hierarchischen Dichtefunktionen soll mit der Berechnung des Integrals

$$I = \int_0^1 \sin\left(\frac{\pi}{2}x\right)\,\mathrm{d}x = \frac{2}{\pi}$$

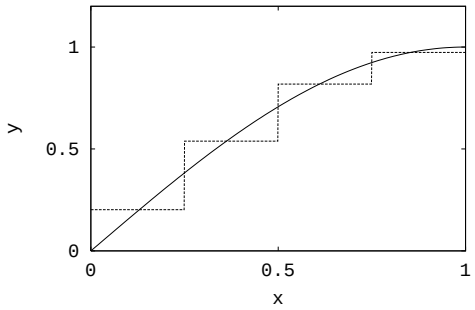
getestet werden. Dazu wurde das Integral mit einem normalen, nicht-adaptiven Monte Carlo-Verfahren und mit dem neuen adaptiven Verfahren berechnet. Als orthonormale Wavelet-Konstruktion wurde eine Haar-MSA benutzt. Die Verfeinerung der Dichtefunktion erfolgte über die Skalen 1, 2 und 3 dieser MSA. Eine Adaption auf Skala 0 ist sinnlos, da diese Skala nur eine Basisfunktion hat. Eine MSA mit linearen B-Splines dient als Beispiel für eine nicht-orthonormale Konstruktion. Die Adaption erfolgte über die Skalen 0, 1, 2 dieser MSA.



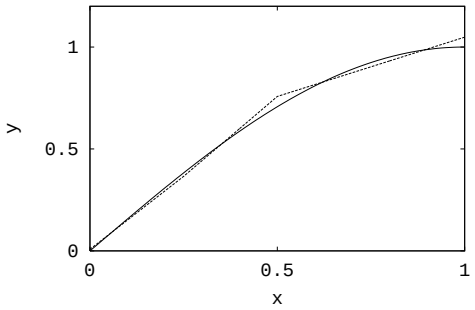

Haar-MSA, Skala 1



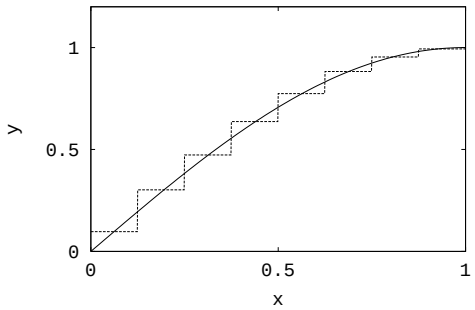

lin. B-Spline-MSA, Skala 0



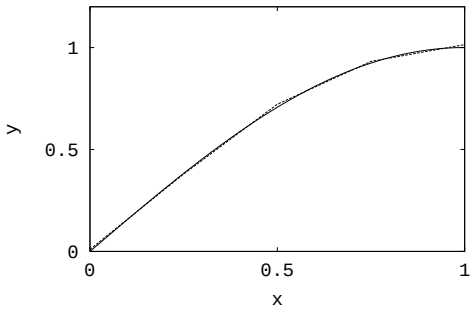

Haar-MSA, Skala 2




lin. B-Spline-MSA, Skala 1




Haar-MSA, Skala 3




lin. B-Spline-MSA, Skala 2

Abbildung 2.6: Hierarchische Adaption der Approximation, Integrand $\sin\left(\frac{\pi}{2}x\right)$

| Durchlauf | Haar-MSA | lin. B-Splines-MSA |
|---|---|---|
| 1 | 0.0947 | 0.0947 |
| 2 | 0.0388 | 0.00586 |
| 3 | 0.0132 | 0.000372 |
| 4 | 0.00432 | 0.00009 |

Tabelle 2.2: Varianz für Monte Carlo-Integration mit adaptiven, hierarchischen Wahrscheinlichkeitsdichten, Integrand $\sin\left(\frac{\pi}{2}x\right)$

| | Samples | | Zeit | Varianz | Beschleunigung |
|---|---|---|---|---|---|
| | Anzahl | Verteilung | [s] | $\sigma_1^2$ | (Faktor) |
| nicht-adaptiv | $4 \cdot 10^6$ | | 4.23 | 0.0947 | |
| | $13 \cdot 10^6$ | | 13.74 | 0.0947 | |
| Haar-Basis | $4 \cdot 10^6$ | 1-1-1-1 | 9.88 | 0.0379 | 1.07 |
| | $13 \cdot 10^6$ | 1-1-1-10 | 33.03 | 0.0146 | 2.70 |
| lin. B-Splines | $4 \cdot 10^6$ | 1-1-1-1 | 11.41 | 0.0253 | 1.39 |
| | $13 \cdot 10^6$ | 1-1-1-10 | 38.82 | 0.00783 | 4.28 |

Tabelle 2.3: Effizienz von Monte Carlo-Integration mit adaptiven hierarchischen Wahrscheinlichkeitsdichten, Integrand $\sin\left(\frac{\pi}{2}x\right)$ (Zeiten: Celeron 400 MHz)

Abb. 2.6 zeigt, wie sich die Approximation des Integranden von Skala zu Skala verfeinert und immer stärker an den Integranden annähert. Die daraus resultierende Verbesserung der Varianz ist aus Tabelle 2.2 ersichtlich. In beiden Fällen, wird mit uniformer Dichte gestartet, deshalb weisen beide Verfahren im ersten Durchlauf dieselbe Varianz auf. In den weiteren Durchläufen sinkt mit zunehmender Güte der Dichtefunktion die Varianz. Aus Abb. 2.6 ist ersichtlich, daß sich der Integrand in der linearen B-Spline-Basis offensichtlich besser annähern läßt als in der stückweise konstanten Haar-Basis. Daraus resultiert die deutlich stärkere Verringerung der Varianz für die B-Spline-MSA.

Tabelle 2.3 zeigt die Ergebnisse. Um die Auswirkungen unterschiedlicher Verteilungen der Samples auf die Durchläufe zu testen, wurden 4 Millionen Samples mit jeweils 1 Million Samples pro Durchlauf und 13 Millionen Samples mit 1 Million Samples in den ersten drei Durchläufen und 10 Millionen Samples im letzten Durchlauf benutzt. Wie man sieht, führt das neue adaptive Verfahren bei gleicher Samplezahl zu höherem Zeitbedarf. Da die Varianz aber stärker abnimmt, als die Zeit zunimmt, ist das Verfahren tatsächlich effizienter als das nicht-adaptive. Außerdem ist aus den Ergebnissen erkennbar, daß es vorteilhaft ist, möglichst viele Samples im letzten Durchlauf des Verfahrens zu benutzen, was den im Abschnitt 2.5.1 gemachten Überlegung zur Verteilung der Samples auf die Durchläufe entspricht.

## 2.6 Adaptive höherdimensionale Dichtefunktionen

Das in 2.4 vorgestellte Verfahrens läßt sich einfach auf höherdimensionale Integrationsaufgaben verallgemeinern. Das Prinzip wird im folgenden für eine zweidimensionale Integrationsaufgabe

$$I = \int_0^1 \int_0^1 f(u,v)\,\mathrm{d}u\,\mathrm{d}v$$

gezeigt und läßt sich ebenso auf höherdimensionale Aufgaben übertragen.

### 2.6.1 Dichtefunktionen in Tensorprodukt-Basisdarstellung

Die Dichtefunktion sei in einer Tensorprodukt-Basisdarstellung gegeben. $N^u$ und $N^v$ seien Funktionsbasen mit $\dim N^u = n^u$ und $\dim N^v = n^v$. Natürlich können $N^u$ und $N^v$ auch identisch sein. Weiter sei wie in Abschnitt 2.4.1 jede Basisfunktion eine skalierte Wahrscheinlichkeitsdichte

$$N_j^u(u) = w_j^u \, p_j^u(u)$$
$$N_k^v(v) = w_j^v \, p_j^v(v)$$

und es seien Verfahren gegeben, die Zufallszahlen entsprechend den Dichtefunktionen $p_j^u$ und $p_k^v$ erzeugen.

Ausgehend von einer Dichtefunktion

$$p(u,v) = \sum_{j=1}^{n^u} \sum_{k=1}^{n^v} c_{j,k} \, N_j^u(u) \, N_k^v(v)$$

in einer Tensorprodukt-Basisdarstellung lassen sich dann Zufallsvektoren $(u_i, v_i) \sim p(u,v)$ erzeugen, indem erst zufällig ein Indexpaar $(j,k)$ gemäß

$$Prob(\text{Wahl von } (j,k)) = \frac{c_{j,k} \, w_j^u \, w_k^v}{\sum\limits_{l=1}^{n^u} \sum\limits_{m=1}^{n^v} c_{l,m} \, w_l^u \, w_m^v}$$

gewählt und dann ein Zufallspaar gemäß der Dichte

$$p_{j,k}(u,v) = p_j^u(u) \, p_k^v(v)$$

erzeugt wird [42, 35]. In diesem Fall lassen sich die Zufallszahlen

$$u_i \sim p_j^u(u)$$
$$v_i \sim p_k^v(v)$$

unabhängig voneinander erzeugen, mit denen eine Monte Carlo-Berechnung des Integrals erfolgt:

$$\int_0^1 \int_0^1 f(u,v) \, \mathrm{d}u \, \mathrm{d}v \approx I_{est} = \frac{1}{m} \sum_{i=1}^{m} \frac{f(u_i, v_i)}{p(u_i, v_i)}$$

### 2.6.2 Approximation des Integranden

Aus der im vorhergehenden Abschnitt beschriebenen Samplegenerierung erhält man Mengen

$$S^{j,k} = \{(u_l^{j,k}, v_l^{j,k}), l = 1, \ldots, n_{j,k}\}$$

von Zufallspaaren $(u_l^{j,k}, v_l^{j,k}) \sim p_{j,k}(u,v)$, aus denen sich die Koeffizienten einer $L^2$-Approximation des Integranden in der Tensorprodukt-Basis berechnen lassen.

Sind beide Basen $N^u$ und $N^v$ orthonormal, ergibt die Samplemenge $S^{j,k}$ zusammen mit den zugehörigen Funktionswerten $f(u_l^{j,k}, v_l^{j,k})$ eine Abschätzung des Koeffizienten $c_{j,k}^\star$:

$$\begin{aligned} c_{j,k}^\star &= \langle\langle f, N_j^u\rangle, N_k^v\rangle \\ &= \int_0^1\int_0^1 f(u,v)\, N_j^u(u)\, N_k^v(v)\,\mathrm{d}u\,\mathrm{d}v \\ &\approx \frac{w_j^u\, w_k^v}{n_{j,k}} \sum_{l=1}^{n_{j,k}} \frac{f(u_l^{j,k}, v_l^{j,k})}{p(u_l^{j,k}, v_l^{j,k})} \end{aligned} \tag{2.21}$$

Sind beide Basen nicht-orthogonal erhält man eine Schätzung des Koeffizienten

$$\tilde{c}_{j,k}^\star \approx \frac{w_j^u\, w_k^v}{n_{j,k}} \sum_{l=1}^{n_{j,k}} \frac{f(u_l^{j,k}, v_l^{j,k})}{p(u_l^{j,k}, v_l^{j,k})} \tag{2.22}$$

in der aus $\tilde{N}^u$ und $\tilde{N}^v$ gebildeten dualen Tensorprodukt-Basis. Auch hier führt ein Basiswechsel die Darstellung zurück in die primäre Tensorprodukt-Basis.

Es ist auch möglich, daß eine Basis orthonormal ist und die andere nicht. Dann erhält man eine Approximation in einer gemischten Basis, die entlang einer Achse aus primären Basisfunktionen und entlang der anderen Achse aus dualen Basisfunktionen besteht. Ein nur für die Achse mit nicht-orthonormaler Basis durchgeführter Basiswechsel führt dann wiederum in eine Repräsentation in rein primärer Basisdarstellung.

### 2.6.3 Verfeinerte Dichtefunktion

Wie in 2.4.3.1 und 2.4.3.2 für die eindimensionale Anwendung beschrieben, müssen negative bzw. zu kleine Koeffizienten korrigiert werden. Bei Benutzung einer adaptiven Schwelle $\tau$ läßt sich auch im höherdimensionalen Fall ein Sicherheitssockel realisieren:

$$\text{wenn } c_{j,k}^\star < \tau\, I_{est}\, w_j^u\, w_k^v, \text{ dann setze } c_{j,k}^\star := \tau\, I_{est}\, w_j^u\, w_k^v$$

Durch Normierung erhält man die verfeinerte Dichtefunktion

$$p^+(u,v) = \sum_{j=1}^{n^u}\sum_{k=1}^{n^v} \frac{c_{j,k}^\star}{\sum_{l=1}^{n^u}\sum_{m=1}^{n^v} c_{l,m}^\star\, w_l^u\, w_m^v} N_j^u(u)\, N_k^v(v)$$

### 2.6.4 Adaptive hierarchische Dichtefunktionen

Der unter 2.5 beschriebene Ansatz zur Verwendung hierarchischer Dichtefunktionen läßt sich ebenfalls auf höherdimensionale Anwendungen übertragen.

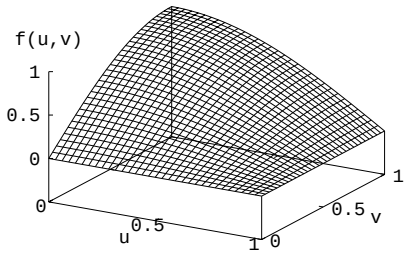



Abbildung 2.7: Integrand $f(u,v) = \sin\left(\frac{\pi}{2}(1-u)\right)\,\sin\left(\frac{\pi}{2}v\right)$

Zur Vereinfachung der Schreibweise sei im folgenden angenommen, daß entlang der Koordinatenrichtungen dieselbe Multi-Skalen-Analyse benutzt werde.[1] Die Dichtefunktion $p_+^j(u,v)$ wird dann in der aus $\Phi^j(u)$ und $\Phi^j(v)$ gebildeten Tensorprodukt-Basis dargestellt:

$$p_+^j(u,v) = \sum_{k=1}^{m^j}\sum_{l=1}^{m^j} c_{k,l}^{j+}\,\phi^j(u)\,\phi^j(v)$$

Diese Dichtefunktion wird auf die nächste Skala verfeinert, indem erst ein Verfeinerungsschritt in jeder Zeile der Matrix $\mathbf{C}^{j+}$ durchgeführt und danach in der daraus resultierenden Matrix jeweils ein Verfeinerungsschritt in jeder Spalte durchgeführt wird. Die so erhaltene Matrix $\mathbf{C}^{j+1}$ stellt die Koeffizientenmatrix der Dichtefunktion $p^{j+1}(u,v)$ in der aus $\Phi^{j+1}(u)$ und $\Phi^{j+1}(v)$ gebildeten Tensorprodukt-Basis dar. Mit dieser Dichtefunktion wird der nächste Durchlauf des adaptiven Monte Carlo-Verfahrens gestartet.

#### 2.6.4.1 Beispiel

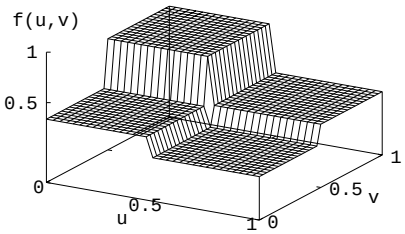
Für den in Abb. 2.7 dargestellten Integranden

$$f(u,v) = \sin\left(\frac{\pi}{2}(1-u)\right)\,\sin\left(\frac{\pi}{2}v\right)$$

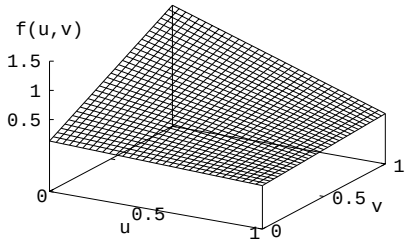
wurde das oben vorgestellte Verfahren mit adaptiven hierarchischen Dichtefunktionen durchgeführt. Als orthonormale Wavelet-Konstruktion wurde eine Haar-MSA verwendet, die Adaption der Dichtefunktion erfolgte über die Skalen 1, 2 und 3. Als nicht-orthonormale MSA wurde eine lineare B-Spline-MSA mit äquidistant unterteilten Knotenvektoren und mehrfachen Endknoten verwendet. Die Dichtefunktion wurde über die Skalen 0, 1 und 2 adaptiert.

Abb. 2.8 zeigt die Adaption der Dichtefunktion nach dem ersten, zweiten und dritten Durchlauf des Verfahrens mit jeweils $10^6$ Samples pro Durchlauf.

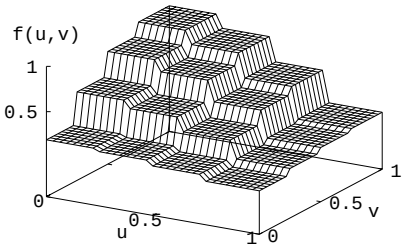
[1] Die in Kapitel 5 vorgestellte `WaveFunc`-Bibliothek unterstützt tatsächlich die Verwendung unterschiedlicher MSAs für $u$- und $v$-Richtung.



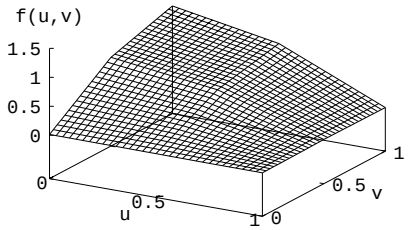

Haar-MSA, Skala 1



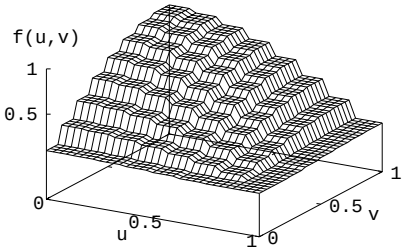

lin. B-Spline-MSA, Skala 0



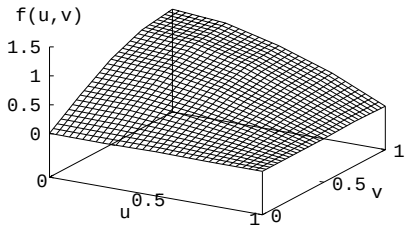

Haar-MSA, Skala 2




lin. B-Spline-MSA, Skala 1




Haar-MSA, Skala 3




lin. B-Spline-MSA, Skala 2

Abbildung 2.8: Hierarchische Adaption der höherdimensionaler Wahrscheinlichkeitsdichten ($10^6$ Samples pro Durchlauf)

| Durchlauf | Haar-MSA | lin. B-Splines-MSA |
|---|---|---|
| 1 | 0.0857 | 0.0857 |
| 2 | 0.0332 | 0.005 |
| 3 | 0.0111 | 0.000251 |
| 4 | 0.00361 | 0.0000733 |

Tabelle 2.4: Varianz für Monte Carlo-Integration mit höherdimensionalen Wahrscheinlichkeitsdichten ($10^6$ Samples pro Durchlauf)

| | Samples | | Zeit | Varianz | Beschleunigung |
|---|---|---|---|---|---|
| | Anzahl | Verteilung | [s] | $\sigma_1^2$ | (Faktor) |
| nicht-adaptiv | $4 \cdot 10^6$ | | 8.69 | 0.0857 | |
| | $13 \cdot 10^6$ | | 27.86 | 0.0857 | |
| Haar-Basis | $4 \cdot 10^6$ | 1-1-1-1 | 19.96 | 0.0334 | 1.12 |
| | $13 \cdot 10^6$ | 1-1-1-10 | 65.31 | 0.0128 | 2.86 |
| lin. B-Splines | $4 \cdot 10^6$ | 1-1-1-1 | 24.94 | 0.0228 | 1.31 |
| | $13 \cdot 10^6$ | 1-1-1-10 | 82.13 | 0.00706 | 4.12 |

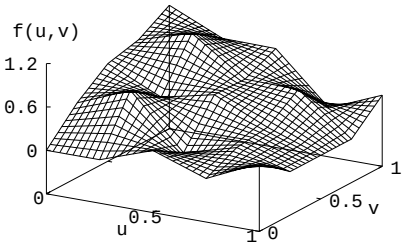
Tabelle 2.5: Effizienz von Monte Carlo-Integration mit höherdimensionalen Wahrscheinlichkeitsdichten (Zeiten: Celeron 400 MHz)

Die in jedem Durchlauf erreichte Varianz des primären Schätzers ist aus Tabelle 2.4 ersichtlich. Beide Verfahren starten im ersten Durchlauf mit einer uniformen Dichte und weisen dieselbe Varianz auf. In jedem weiteren Durchlauf wird die aus dem vorhergehenden Durchlauf erhaltene Dichtefunktion (jeweils dargestellt in Abb. 2.8) benutzt und führt zu immer kleineren Werten der Varianz. Die lineare B-Spline-MSA erreicht deutlich niedrigere Werte, da sie eine bessere Approximation des glatten Integranden ermöglicht als die stückweise konstante Approximation in der Haar-MSA.

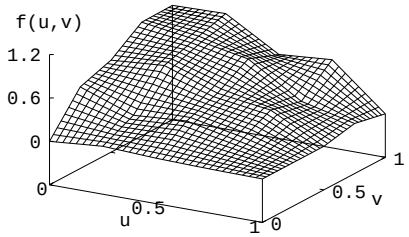
Um zu prüfen, ob sich tatsächlich eine Verbesserung der Effizienz ergibt, wurden die adaptiven Verfahren mit einer nicht-adaptiven Monte Carlo-Integration mit uniformer Sample-Verteilung verglichen. Tabelle 2.5 zeigt die für eine bestimmte Samplezahl benötigte Zeit sowie die erreichte Varianz, die sich im Fall der adaptiven Verfahren als mit der Samplezahl gewichteter Mittelwert über die einzelnen Durchläufe ergibt. Wie man erkennt, sinkt die Varianz jeweils mit einem größeren Faktor als die Zeit wächst. Die Samples wurden auf die 4 Durchläufe im Verhältnis $1:1:1:1$ bzw. $1:1:1:10$ verteilt. Auch hier zeigt sich, daß die Effizienz sich verbessert, wenn im letzten Durchlauf mehr Samples benutzt werden als in frühen Durchläufen.

#### 2.6.4.2 Konvergenz

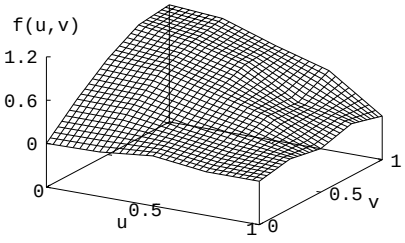
Die für das obige Beispiel durchgeführten Versuche haben gezeigt, daß adaptive Dichtefunktionen mit höhergradigen Funktionsbasen, z. B. linearen, quadratischen oder kubischen B-Spline-Basen, empfindlich auf zu geringe Samplezahlen reagieren. Abb. 2.9 zeigt, daß bei geringen Samplezahlen die Qualität der Approximation schlecht wird und infolgedessen die Varianz des primären Schätzers steigt. Das liegt daran, daß bei zu geringer Samplezahl die nach (2.22) berechneten Koeffizienten mit einem zu großen Fehler behaftet sind. Bei einer praktischen Anwendung dieses Verfahrens ist also darauf zu achten, daß so viele Samples zur Adaption benutzt werden, daß die Approximation nahe genug an das tatsächliche Proximum herankommt. Statt einer Erhöhung der Samplezahl ist auch eine Verringerung des Grades der Basisfunktionen oder der Skala der MSA möglich.



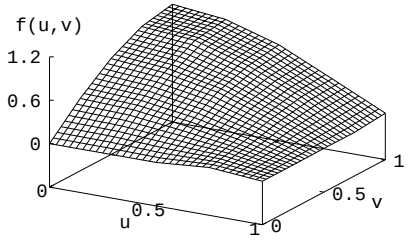

$10^2$ Samples pro Durchlauf
$\sigma_1^2 = 0.0145$




$10^3$ Samples pro Durchlauf
$\sigma_1^2 = 0.00188$




$10^4$ Samples pro Durchlauf
$\sigma_1^2 = 0.00170$




$10^5$ Samples pro Durchlauf
$\sigma_1^2 = 0.000420$

Abbildung 2.9: Konvergenz höherdimensionaler Wahrscheinlichkeitsdichten, lin. B-Spline-MSA, Skala 2

# Kapitel 3

# Monte Carlo-Ray Tracing mit adaptiven hierarchischen Dichtefunktionen

In diesem Kapitel wird gezeigt, wie der mathematische Ansatz zur Monte Carlo-Integration mit adaptiven hierarchischen Dichtefunktionen zur Effizienzsteigerung von Monte Carlo-Ray Tracing-Verfahren eingesetzt werden kann. Als Anwendung wird ein auf Pfadverfolgung basierendes Verfahren zur fotorealistischen Bilderzeugung entwickelt. Mit Versuchsreihen wird der Einfluß unterschiedlicher Parametrisierungen auf die Qualität des Verfahrens untersucht.

## 3.1 Monte Carlo-Ray Tracing

Physikalische Grundlage fotorealistischer Bilderzeugung ist die *Bildsynthese-Gleichung* (*rendering equation*). Sie wurde von Kajiya ursprünglich in Form einer Drei-Punkt-Transportgleichung formuliert [41]. Für die hier vorgestellte Anwendung ist eine Punkt-Richtungs-Darstellung besser geeignet:

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} \rho_{bd}(\mathbf{x}, \omega_o, \omega)\, L_i(\mathbf{x}, \omega)\, \cos\theta \, \mathrm{d}\omega \tag{3.1}$$

$L(\mathbf{x}, \omega_o)$ ist die Radiance, die den Punkt $\mathbf{x}$ in Richtung $\omega_o = (\theta_o, \phi_o)$ verläßt, $L_e$ ist die Eigenemission der Fläche und $\rho_{bd}$ ist der bidirektionale Reflexionskoeffizient (*bidirectional reflection distribution function*, kurz: *BRDF*). Integrationsbereich ist die Hemisphäre $\Omega$ aller Einfallsrichtungen $\omega = (\theta, \phi)$ über dem Punkt $\mathbf{x}$. $(\theta, \phi)$ ist die Repräsentation der Richtung $\omega$ in sphärischen Koordinaten der Hemisphäre über dem Punkt $\mathbf{x}$. In der hier dargestellten Form wird nur Reflexion berücksichtigt, Transmission läßt sich natürlich leicht zufügen.

Gleichung (3.1) beschreibt das Energie-Gleichgewicht am Punkt $\mathbf{x}$: Das in eine bestimmte Richtung $\omega_o$ ausfallende Licht setzt sich zusammen aus Eigenemission $L_e$ der Fläche und dem reflektierten Anteil des aus allen Richtungen der

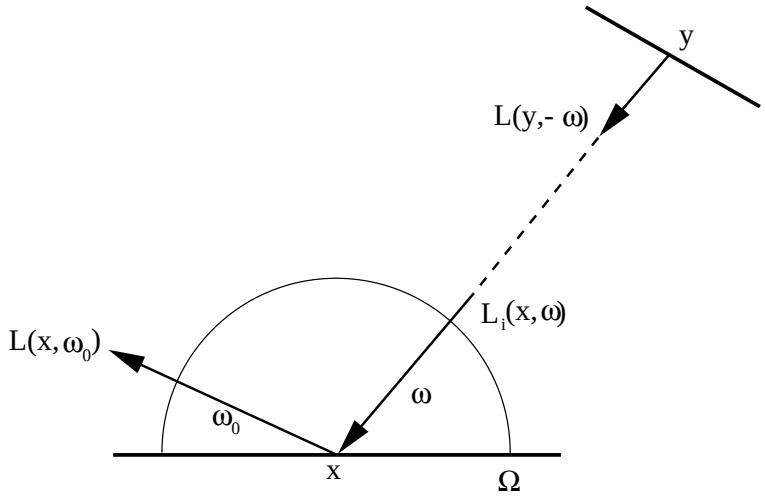



Abbildung 3.1: Bildsynthese-Gleichung

Hemisphäre einfallenden Lichts $L_i$. Die Lösung dieser Gleichung wird dadurch erschwert, daß das einfallende Licht $L_i$ als ausfallendes Licht $L(\mathbf{y}, -\omega)$ von einem anderen Punkt $\mathbf{y}$ stammt und dort natürlich wieder durch die Gleichung (3.1) bestimmt wird (Abb. 3.1). Der Punkt $\mathbf{y}$ ist der erste Punkt auf einer Fläche der Szene, auf den man trifft, wenn man sich von Punkt $\mathbf{x}$ in Richtung $\omega$ bewegt. Dieses ist die Grundoperation der Strahlverfolgung, man kann sie auch in Form einer Ray Tracing-Funktion formulieren

$$\mathbf{y} = r(\mathbf{x}, \omega)$$

und erhält durch Einsetzen eine Darstellung als Integralgleichung

$$L(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\Omega} \rho_{bd}(\mathbf{x}, \omega_o, \omega)\, L(r(\mathbf{x}, \omega), -\omega)\, \cos\theta \, \mathrm{d}\omega. \tag{3.2}$$

Es kann passieren, daß ein Strahl keine Fläche der Szene trifft, in diesem Fall wird $L(r(\mathbf{x}, \omega), -\omega)$ auf Null oder auf eine vorgegebene Umgebungshelligkeit $L_a$ gesetzt.

Verfahren, die die Integralgleichung (3.2) mit Monte Carlo-Methoden lösen, werden als *Monte Carlo-Ray Tracing*-Verfahren bezeichnet. In den folgenden Abschnitten werden zwei Varianten des Monte Carlo-Ray Tracing vorgestellt, nämlich *verteiltes Ray Tracing* und *Pfadverfolgung*.

### 3.1.1 Verteiltes Ray Tracing

Verteiltes Ray Tracing (*distributed ray tracing*) [16] entspricht einer Monte Carlo-Berechnung von (3.2) [75] und hat den Namen daher, daß Strahlen über die Hemisphäre der Einfallsrichtungen verteilt werden:

$$L(\mathbf{x}, \omega_o) \approx L_e(\mathbf{x}, \omega_o) + \frac{1}{m} \sum_{j=1}^{m} \frac{\rho_{bd}(\mathbf{x}, \omega_o, \omega_j)\, L(r(\mathbf{x}, \omega_j), -\omega_j)\, \cos\theta_j}{p(\omega_j)}. \tag{3.3}$$

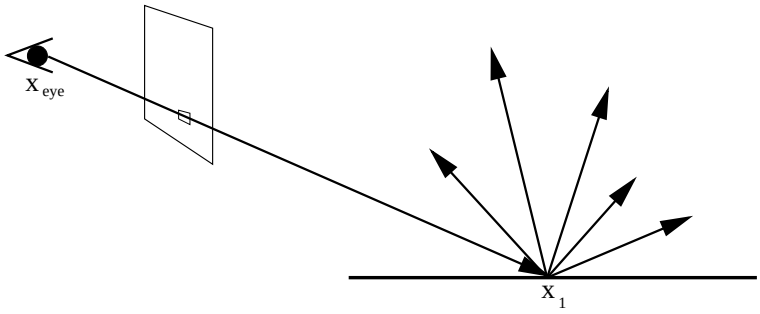



Abbildung 3.2: Verteiltes Ray Tracing

Die Richtungen $\omega_j = (\theta_j, \phi_j)$ werden nach einer über die Hemisphäre definierten WDF $p(\omega)$ erzeugt. Häufig wird als Dichtefunktion die normierte BRDF $\rho_{bd}(\mathbf{x}, \omega_o, \omega)$ bzw. $\rho_{bd}(\mathbf{x}, \omega_o, \omega) \cos\theta$ gewählt. Die Berechnung des einfallenden Lichts $L(r(\mathbf{x}, \omega_j), -\omega_j)$ erfolgt rekursiv auf dieselbe Weise.

Ein Bild wird mit diesem Verfahren erzeugt, indem man an einem Augenpunkt $\mathbf{x}_{eye}$ startet. Der in Pixel gerasterte Ausschnitt der Bildebene definiert die Startrichtungen der Primärstrahlen. Von $\mathbf{x}_{eye}$ wird ein Strahl durch den Mittelpunkt eines Pixels verfolgt. Dieser Strahl trifft den Punkt $\mathbf{x}_1$ in der Szene. Von dort werden $m$ Strahlen in zufällig gewählte Richtungen verfolgt (Abb. 3.2). Das von diesem Punkt in Richtung des Auges ausgestrahlte Licht wird nach (3.3) berechnet und als Helligkeit des Pixels gespeichert. Da vom ersten Trefferpunkt rekursiv Strahlen in $m$ Richtungen verfolgt werden, können bis zu $m$ neue Trefferpunkte gefunden werden, von denen wiederum $m$ Strahlen ausgehen, so daß es in der zweiten Generation bis zu $m^2$ Trefferpunkte gibt. Wie man sieht, kann dies im ungünstigsten Fall zu einer exponentiellen Zunahme der Strahlenzahl mit der Rekursionstiefe führen. Aus diesem Grund und um unendliche Rekursion zu verhindern, muß die Anwendung eines der unten beschriebenen Verfahren zur Pfadterminierung erfolgen.

### 3.1.2 Pfadverfolgung

Die zweite Variante des Monte Carlo-Ray Tracing, die Pfadverfolgung (*path tracing*) [41], wird oftmals aus einer Reihenentwicklung der Bildsynthese-Gleichung hergeleitet, so z. B. in [75], sie kann aber auch als Spezialfall des verteilten Ray Tracing angesehen werden. Führt man das verteilte Ray Tracing mit $m = 1$, also mit nur einem Sample pro Integration aus, führt das dazu, daß man einen *Pfad* $\mathbf{x_{eye}}$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, ... durch die Szene verfolgt (Abb. 3.3). Dieser Pfad entsteht dadurch, daß von jedem bereits getroffenen Punkt $\mathbf{x}_i$ eine Zufallsrichtung gewählt und ein Strahl in diese Richtung verfolgt wird. Der dort getroffene Punkt $\mathbf{x}_{i+1}$ ist die nächste Station auf dem Pfad.

Im Verlauf des Pfades wird an jedem Punkt ein auf (3.2) basierendes Integral mit nur einem Sample geschätzt. Aufgrund der in Kapitel 2 vorgestellten Eigenschaften der Monte Carlo-Integration ist zu erwarten, daß diese Schätzun-

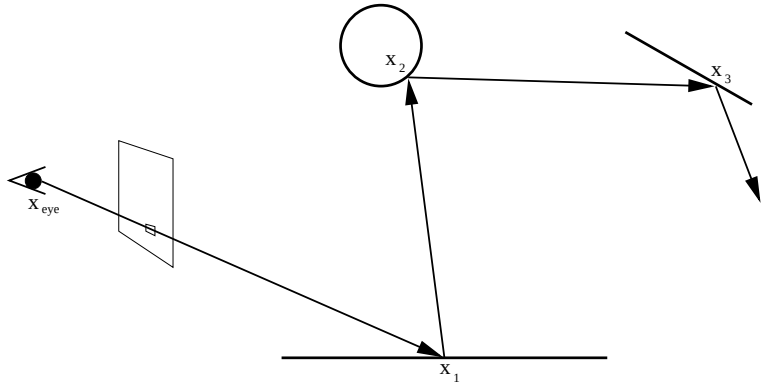



Abbildung 3.3: Pfadverfolgung

gen hohe Fehler aufweisen. Man kann die Schätzung aber dadurch verbessern, daß man mehrere Pfade verfolgt und deren Resultate zu einer Pixel-Helligkeit mittelt. Dieser Ansatz hat den Vorteil, daß keine exponentielle Zunahme der Strahlanzahl mit der Rekursionstiefe auftritt. Er ist deshalb in der Praxis dem verteilten Ray Tracing oftmals überlegen [1].

Entlang des Pfades kann man ein Gewicht $w$ mitführen, das angibt, welcher Anteil des vom nächsten Trefferpunkt $\mathbf{x}_i$ ausgestrahlten Lichts in die Helligkeit des Pixels eingeht. Für den ersten Trefferpunkt $\mathbf{x}_1$ ist das Gewicht $w_1 := 1$. Das Gewicht der weiteren Trefferpunkte ergibt sich dann aus (3.3) als

$$w_{i+1} = w_i \frac{\rho_{bd}(\mathbf{x}_i, \omega_{oi}, \omega_i) \cos\theta_i}{p(\omega_i)}$$

wobei $\omega_{oi}$ die Ausfallrichtung des Lichts am Punkt $\mathbf{x}_i$ und $\omega_i = (\theta_i, \phi_i)$ die zufällig erzeugte Einfallsrichtung ist.

### 3.1.3 Stratifikation

Ein signifikanter Anteil der Helligkeit eines Punktes in der Szene ist die direkt von Lichtquellen einfallende Beleuchtung. Beim zufälligen Abtasten der Hemisphäre werden Lichtquellen manchmal getroffen und manchmal verfehlt, was dazu führt, daß die Helligkeit benachbarter Pixel stark schwanken kann. Je kleiner die Lichtquellen sind, um so geringer ist die Wahrscheinlichkeit, daß Strahlen auf sie treffen. Erst der Einsatz sehr hoher Samplezahlen führt dann zu brauchbaren Resultaten. Deutliche Verbesserungen kann man hier durch Stratifikation erzielen. Die Hemisphäre wird hierbei in Regionen unterteilt, aus denen direkte Beleuchtung von Lichtquellen einfällt, und in den Rest der Hemisphäre, aus dem nur indirektes, reflektiertes Licht stammt [27] (Abb. 3.4).

Für eine Realisierung dieses Ansatzes ist eine Kenntnis der tatsächlichen Unterteilung der Hemisphäre nicht erforderlich. Die Stratifikation der Hemisphäre erfolgt implizit dadurch, daß gezielt Strahlen in Richtung der Lichtquellen verfolgt werden. Die Richtung dieser Strahlen kann z. B. dadurch festgelegt werden,

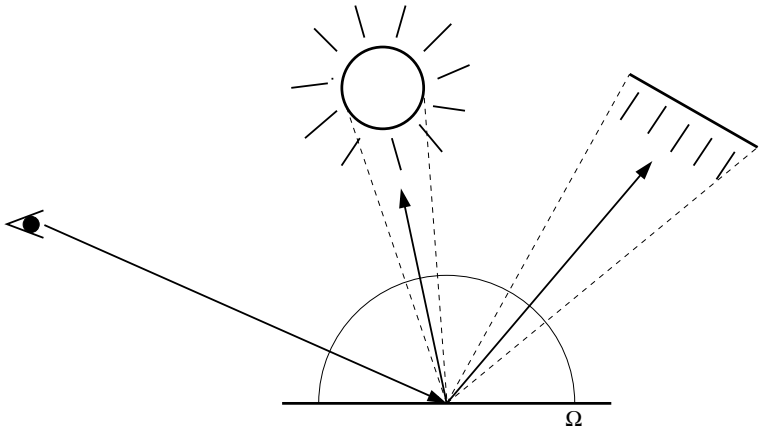



Abbildung 3.4: Stratifikation

daß zufällige Punkte auf der Lichtquelle als Ziel der Strahlen ausgewählt werden. Danach werden über die Hemisphäre verteilte Samples erzeugt, die die indirekte, nicht direkt von Lichtquellen stammende Beleuchtung abtasten. Damit keine Beleuchtungsanteile mehrfach in das Resultat eingehen, gibt es zwei Möglichkeiten:

- Wird bei der Abtastung der Lichtquellen auch das von ihnen reflektierte Licht berechnet, darf ein zur Abtastung der Hemisphäre erzeugter Strahl nicht benutzt werden, falls er auf eine Lichtquelle trifft. In diesem Fall wird er verworfen und ein neuer Strahl erzeugt. Dieser Ansatz entspricht einer stratifizierten Berechnung des Integrals in (3.2). Jede Lichtquelle stellt jeweils ein Stratum dar, ein weiteres Stratum ist die restliche, nicht von Lichtquellen bedeckte Hemisphäre.

- Wenn bei der Abtastung der Lichtquellen nur deren Eigenemission $L_e$ erfaßt wird, dann erfolgt eine normale Abtastung der Hemisphäre, die aber von den Trefferpunkten nicht die Eigenemission sondern nur das reflektierte Licht berücksichtigen darf. Das entspricht einer Aufspaltung des Integrals in getrennte Integration der direkten Beleuchtung $L_e$ und der indirekten, reflektierten Beleuchtung $L_r$:

$$\begin{aligned} L(\mathbf{x},\omega_o) &= L_e(\mathbf{x},\omega_o) + L_r(\mathbf{x},\omega_o) \\ L_r(\mathbf{x},\omega_o) &= \int_\Omega \rho_{bd}(\mathbf{x},\omega_o,\omega)\, L_e(r(\mathbf{x},\omega),-\omega)\, \cos\theta \,\mathrm{d}\omega \\ &\quad + \int_\Omega \rho_{bd}(\mathbf{x},\omega_o,\omega)\, L_r(r(\mathbf{x},\omega),-\omega)\, \cos\theta \,\mathrm{d}\omega \end{aligned}$$

  Die Abtastung der Lichtquellen entspricht einer stratifizierten Berechnung des ersten Integrals, wobei wieder jede Lichtquelle ein Stratum bildet. Die Berechnung des zweiten Integrals erfolgt durch die Abtastung der Hemisphäre als normale, nicht-stratifizierte Monte Carlo-Berechnung.

### 3.1.4 Pfadterminierung

Ein auf Grundlage von (3.3) berechneter Pfad endet nur, wenn der letzte Strahl des Pfades kein Objekt trifft oder wenn das getroffene Objekt nicht reflektiert. Da ein Pfad prinzipiell unendlich lang werden kann, ist es nötig, einen Mechanismus vorzusehen, der Pfade terminiert. Im allgemeinen wird mit zunehmender Pfadlänge der Beitrag der folgenden Trefferpunkte zur Gesamthelligkeit des Pixels immer kleiner. Ein einfacher Ansatz zur Terminierung besteht deshalb darin, eine feste Pfadlänge vorzugeben, und bei Erreichen dieser Pfadlänge die Rekursion abzubrechen, indem keine Reflexion mehr berechnet wird. Statt einer festen Pfadlänge kann bei der Berechnung auch das Gewicht $w_i$ des aktuellen Trefferpunktes $\mathbf{x}_i$ mitgeführt werden. Der Pfad kann dann bei Unterschreiten eines Mindestgewichts terminiert werden. Diese Ansätze führen einen kleinen, systematischen Fehler in die Berechnung ein, da für den letzten Schritt des Pfades die Gleichung (3.2) verfälscht wird und nur $L_e$ in die Rechnung eingeht. Oftmals bleibt dieser Fehler in den erzeugten Bildern unsichtbar, er kann aber für bestimmte Szenen signifikant werden.

Eine Möglichkeit zur Pfadterminierung ohne systematischen Fehler ist als *Russisches Roulette* bekannt [1]. Hat ein Pfad eine bestimmte Länge erreicht bzw. ein bestimmtes Gewicht unterschritten, wird mit einer Zufallsentscheidung bestimmt, ob der Pfad terminiert oder fortgesetzt wird. Mit einer vorgegebenen Wahrscheinlichkeit $p_a$ wird der Pfad beendet. Wenn die Zufallsentscheidung eine Fortsetzung des Pfades ergibt (mit Wahrscheinlichkeit $1 - p_a$), wird das Gewicht des aktuellen Trefferpunktes um den Faktor $1/(1 - p_a)$ erhöht. Der Erwartungswert $E(W)$ des Gewichts ändert sich hierdurch nicht. Wenn $w_i$ das Gewicht vor Anwendung von Russischen Roulette war, gilt:

$$\begin{aligned} E(W) &= p_a \cdot 0 + (1 - p_a)\frac{w_i}{1 - p_a} \\ &= w_i \end{aligned}$$

Daraus resultiert, daß dieser Ansatz keinen systematischen Fehler in die Berechnung einführt.

### 3.1.5 Variationen des Ray Tracing-Verfahrens

Ray Tracing-Verfahren beginnen beim Auge eines virtuellen Betrachters und versuchen Wege zu den Lichtquellen zu finden. Den umgekehrten Weg gehen Particle Tracing-Verfahren. Sie verschiessen Energieeinheiten von den Lichtquellen und suchen Wege in das Auge [1, 62]. Dieser Ansatz entspricht der Monte Carlo-Berechnung der zur Bildsynthese-Gleichung (3.1) dualen Potential-Gleichung [63].

Aus einer Kombination von Ray Tracing und Particle Tracing erhält man bidirektionale Verfahren [50, 84, 64]. Es werden Teilpfade konstruiert, die vom Auge und je einer Lichtquelle starten. Aus einer paarweisen Verknüpfung der Trefferpunkte dieser Teilpfade erhält man Abschätzungen für das über den Gesamtpfad fließende Licht.

Basierend auf der Erkenntnis, daß für eine effiziente Integration die Gleichverteilung der Abtastpunkte wichtiger ist als die Zufälligkeit, gibt es eine Klasse von Monte Carlo-Verfahren, die im normalen Monte Carlo-Ansatz die Zufallsfolgen durch deterministisch bestimmte Zahlenfolgen ersetzen [61]. Solche Verfahren werden als *Quasi-Monte Carlo-Verfahren* bezeichnet. Alexander Keller hat die Anwendung dieses Ansatzes auf Probleme der Computergrafik untersucht [45]. Unter anderem stellt er ein auf dem Quasi-Monte Carlo-Ansatz basierendes Particle Tracing-Verfahren zur Lösung der Bildsynthese-Gleichung vor [43], das unter Nutzung moderner Grafikhardware ein sehr schnelles Bilderzeugungsverfahren ermöglicht [44]. Er konnte zeigen, daß der Wechsel auf Quasi-Monte Carlo-Verfahren Anwendungen im Bereich der Computergrafik bis zu einem Faktor 5 beschleunigen kann.

## 3.2 Bestehende Ansätze zur Nutzung adaptiver Dichtefunktionen

Bei den Ansätzen zur Benutzung adaptiver Dichtefunktionen kann man zwischen zwei Klassen von Verfahren unterscheiden.

Die erste Klasse besteht aus Zwei-Schritt-Verfahren, die in einem ersten Schritt Dichtefunktionen konstruieren, welche im zweiten Schritt für ein Importance Sampling benutzt werden [39, 40, 82].

Bustillo verwendet eine neuronale Gas-Struktur, um Information über Beleuchtung in der Szene zu speichern [7]. Im ersten Schritt werden Strahlen von den Lichtquellen verschossen und in der neuronalen Gas-Struktur gespeichert. Im zweiten Schritt wird die in der Gas-Struktur gespeicherte Information für ein evolutionäres Importance Sampling benutzt.

Zur zweiten Klasse gehören Mehr-Schritt-Verfahren, die adaptiv Dichtefunktionen generieren, so daß spätere Schritte von den in früheren Schritten adaptierten Dichtefunktionen profitieren können. Von Dutré und Willems stammen Verfahren zum Particle Tracing. Das erste Verfahren basiert auf der VEGAS-Methode [55] und erzeugt adaptive Dichtefunktionen an jeder Lichtquelle [19]. Eine Weiterentwicklung dieses Verfahrens unterteilt die Flächen der Szene in Teilflächen, plaziert auf jede Teilfläche eine stückweise konstante Dichtefunktion, die im Verlauf des Particle Tracing adaptiert wird [20]. Eine adaptive Raum-/Richtungs-Unterteilung in Form eines 5D-Baumes wird von Lafortune und Willems in einem Ray Tracing-Verfahren benutzt [51]. In diesem Baum wird adaptiv eine stückweise konstante Repräsentation der einfallenden Beleuchtung $L_i$ erzeugt, die für ein Importance Sampling bei der Generierung von Reflexionsstrahlen benutzt wird.

Veach und Guibas präsentieren eine auf Metropolis-Sampling basierende Methode, die gegen die optimale Wahrscheinlichkeitsdichte konvergiert, ohne daß eine Repräsentation der WDF gespeichert wird [85]. Beginnend mit einer initialen Menge von Transportpfaden, die z. B. durch bidirektionale Pfadverfolgung

[84, 52] gewonnen wird, erfolgt eine lokale Suche im Raum aller Pfade, indem zufällige Mutationen der Pfade generiert werden.

## 3.3 Nutzung adaptiver hierarchischer Dichtefunktionen

Um den in Kapitel 2 vorgestellten Ansatz zur Nutzung adaptiver hierarchischer Dichtefunktionen beim Monte Carlo-Ray Tracing anzuwenden, muß die zugrundeliegende Gleichung (3.2) in eine geeignete Form gebracht werden.

### 3.3.1 Mathematischer Ansatz

Gleichung (3.2) enthält eine Integration über die Hemisphäre. Diese kann auch als Doppelintegral über sphärische Koordinaten formuliert werden:

$$\int_{\Omega} f(\omega)\,\mathrm{d}\omega = \int_{0}^{\frac{\pi}{2}} \int_{0}^{2\pi} f(\theta,\phi)\,\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi \tag{3.4}$$

Auf dieser Grundlage kann der in Kapitel 2 beschriebene Ansatz mit zweidimensionalen adaptiven Dichtefunktionen benutzt werden. Dafür ist eine Abbildung vom Definitionsbereich $[0,\pi/2]\times[0,2\pi]$ auf $[0,1]\times[0,1]$ und umgekehrt erforderlich. Damit über $[0,1]\times[0,1]$ nach einer Dichtefunktion $p$ erzeugte Zufallsvektoren korrekt auf Richtungen der Hemisphäre abgebildet werden, sollte diese Abbildung flächentreu sein, nur dann wird z. B. eine uniforme Verteilung über $[0,1]\times[0,1]$ auf eine Gleichverteilung der Richtungen über die Hemisphäre abgebildet. Diese Bedingungen werden erfüllt durch die Abbildung $M : [0,\pi/2]\times[0,2\pi] \to [0,1]\times[0,1]$:

$$M(\theta,\phi) = (1-\cos\theta, \frac{\phi}{2\pi})$$

mit der Umkehrung $M^{-1} : [0,1]\times[0,1] \to [0,\pi/2]\times[0,2\pi]$:

$$M^{-1}(u,v) = (\arccos(1-u), 2\pi v)$$

Mit dieser Abbildung und Anwendung der Substitutionsregel erhält man aus (3.4) ein über $[0,1]\times[0,1]$ berechnetes Integral:

$$\int_{\Omega} f(\omega)\,\mathrm{d}\omega = \int_{0}^{1}\int_{0}^{1} f(\arccos(1-u), 2\pi v)\,2\pi\,\mathrm{d}u\,\mathrm{d}v$$

Durch Anwendung dieser Substitution auf (3.2) ergibt sich die Monte Carlo-Lösung

$$L(\mathbf{x},\omega_o) \approx L_e(\mathbf{x},\omega_o) + \frac{1}{m}\sum_{j=1}^{m} 2\pi \frac{\rho_{bd}(\mathbf{x},\omega_o,\omega_j)\,L(r(\mathbf{x},\omega_j),-\omega_j)\,\cos\theta_j}{p(u_j,v_j)} \tag{3.5}$$

mit $\omega_j = (\theta_j, \phi_j) = (\arccos(1 - u_j), 2\pi v_j)$.

Ein direkt auf (3.5) basierendes Ray Tracing-Verfahren wurde von Stefan Legner realisiert [54]. Am Trefferpunkt des primären Augenstrahles wird eine Abtastung der Hemisphäre durchgeführt. Die über die Hemisphäre einfallende Beleuchtung $L_i$ wird in einer auf der Haar-MSA basierenden Dichtefunktion adaptiert. In mehreren Durchläufen wird so die Hemisphäre abgetastet. Es konnte gezeigt werden, daß dieser Ansatz effizienter ist als eine nicht-adaptive Abtastung der Hemisphäre. Dieses Verfahren hat aber noch Nachteile. Für die Generation von Strahlen an sekundären Trefferpunkten stehen keine adaptiven Dichtefunktionen zur Verfügung. Außerdem wird jede Dichtefunktion nach der Benutzung verworfen, so daß die Berechnung späterer Pixel in keiner Weise von vorher generierten Dichtefunktionen profitiert.

Um zu einem Verfahren zu gelangen, das einmal adaptierte Dichten nicht verwirft und auch die optimierte Verfolgung rekursiv erzeugter Strahlen unterstützt, wurde ein anderer Ansatz gewählt.

### 3.3.2 Verfahren

Es wird ein Pfadverfolgungs-Verfahren angewandt, das in mehreren Durchläufen arbeitet. Adaptive Dichtefunktionen werden den Flächen der Szene zugeordnet. Wird im Laufe der Pfadverfolgung eine Fläche getroffen, dann wird die zugeordnete Dichtefunktion dazu benutzt, die Strahlrichtung zur Fortsetzung des Pfades zu generieren. Das über diese Pfadfortsetzung einfallende Licht wird entsprechend des in Kapitel 2 beschriebenen Verfahrens dazu benutzt, Daten zur Adaption der Dichtefunktion zu sammeln. Die Bilderzeugung erfolgt in mehreren Durchläufen. In jedem Durchlauf wird eine bestimmte Zahl von Strahlen pro Pixel verschossen. Nach jedem Durchlauf werden die Dichtefunktionen adaptiert und verfeinert, so daß im folgenden Durchlauf die Pfadverfolgung die verbesserten Dichtefunktionen nutzt. Vorteil dieses Verfahrens ist, daß für jeden Trefferpunkt im Laufe der Rekursion eine Dichtefunktion verfügbar ist und daß keine Dichtefunktionen verworfen werden, sondern diese von Durchlauf zu Durchlauf verbessert werden.

Jeder Durchlauf dieses Verfahrens liefert ein Bild, wobei damit zu rechnen ist, daß spätere Durchläufe von den adaptierten Dichtefunktionen profitieren und bessere Qualität liefern sollten. Im praktischen Einsatz wird man ähnlich wie oben in Abschnitt 2.5.1 beschrieben in frühen Durchläufen, die hauptsächlich der Adaption dienen, wenige Samples pro Pixel benutzen und in späteren mehr. Das Gesamtbild wird als ein mit der entsprechenden Samplezahl gewertetes Mittel der aus den Durchläufen stammenden Bilder bestimmt.

**Anmerkung:** Da die während der Adaption berechneten Samples schlechtere Qualität aufweisen als die in späteren Durchläufen berechneten Samples, kann man fragen, ob die Einrechnung dieser Samples zu einer Verschlechterung des Gesamtergebnisses führt. Die Beantwortung dieser Frage hängt davon ab, welche relative Qualitätsverbesserung mit der adaptierten Dichtefunktion erreichbar ist. Entsprechende Untersuchungen haben gezeigt, daß man für die

berechneten Szenen bessere Resultate erhält, wenn man die Adaptions-Samples in das Gesamtergebnis einrechnet.

Die nach (3.5) adaptierten Dichtefunktionen haben einen zweidimensionalen Definitionsbereich, sind aber zusätzlich abhängig vom Punkt $\mathbf{x}$ und der Ausfallsrichtung $\omega_o$. Wie in [20] angemerkt, besteht eine Lösungsmöglichkeit darin, Dichtefunktionen über einem sechsdimensionalen Definitionsbereich zu adaptieren. Gegen diese Lösung spricht aber der hohe Speicherbedarf einer sechsdimensionalen Tensorprodukt-Darstellung und insbesondere die Gefahr, daß die zur Konvergenz der Approximation nötige Samplezahl zu groß wird. Deshalb sollte ein anderer Weg benutzt werden.

Die Orts-Abhängigkeit wird beseitigt, indem die Flächen der Szene in Teilflächen unterteilt und jeder Teilfläche eine Dichtefunktion zugeordnet wird. Zu einem Trefferpunkt wird dann die Teilfläche bestimmt, in der dieser Punkt liegt, und die zu der Teilfläche gehörende Dichtefunktion benutzt. Bei genügend kleiner Unterteilung werden sich die optimalen Dichtefunktionen zwischen zwei Punkten innerhalb einer Teilfläche nur noch so wenig unterscheiden, daß es gerechtfertigt ist, eine Dichtefunktion für die gesamte Teilfläche zu benutzen. Selbst wenn die Dichtefunktion einmal von der optimalen abweichen sollte, führt dies, wie aus Kapitel 2 bekannt, nicht zu fehlerhaften Ergebnissen, sondern höchstens zu höherer Varianz des Ergebnisses.

Um die Abhängigkeit von der Ausfallsrichtung zu behandeln, gibt es mehrere Möglichkeiten. Statt die Dichtefunktion an den Integranden anzunähern, ist es möglich, sie nur an die einfallende Beleuchtung, also an $L(r(\mathbf{x},\omega),-\omega)\cos\theta$ zu adaptieren. Samples können dann entweder direkt aus dieser Dichtefunktion generiert werden oder es wird zu einem Ausfallswinkel $\omega_o$ eine neue Dichtefunktion generiert, nach der dann das Sample erzeugt wird. Die neue Dichtefunktion muß in diesem Fall als Produkt der BRDF $\rho_{bd}(\mathbf{x},\omega_o,\omega)$ mit der Approximation der einfallenden Beleuchtung berechnet werden. Dieser Ansatz hat aber den Nachteil, daß die Sample-Generierung sehr teuer wird, da die Generierung jedes einzelnen Samples die Konstruktion einer neuen Dichte erfordert.

Zu einer deutlich besseren Lösung gelangt man, wenn man unterschiedlich danach vorgeht, ob eine BRDF mit gerichtetem, spekularen Reflexionsverhalten oder eine diffuse BRDF vorliegt. Im Falle gerichteter Reflexion überwiegt der aus dem Kegel der Hauptreflexionsrichtung reflektierte Lichtanteil die aus anderen Richtungen der Hemisphäre einfallenden Anteile so stark, daß der Einsatz adaptiver Dichten ganz entfallen kann und die Strahlrichtungen ausschließlich nach der BRDF generiert werden. Für diffuse Reflexion entfällt die Richtungsabhängigkeit der BRDF, so daß eine Adaption der Dichtefunktion an die einfallende Beleuchtung $L(r(\mathbf{x},\omega),-\omega)\cos\theta$ die korrekte Dichtefunktion ergibt. Liegt eine BRDF mit gemischten Eigenschaften vor, können spekular und diffus reflektierter Anteil unabhängig voneinander berechnet und dann addiert werden.

Wird das Verfahren mit Stratifikation der direkten Beleuchtung angewandt, darf die Adaption der Dichtefunktionen nur nach dem indirekt einfallenden Licht erfolgen, da die Strahlen zu den Lichtquellen deterministisch erzeugt werden. In

diesem Fall wird natürlich auch nur die Berechnung der indirekten Beleuchtung verbessert. Deshalb ist zu erwarten, daß das Verfahren umso mehr Vorteile bringt, je wichtiger die indirekte Beleuchtung in einer Szene ist. Das Verfahren sollte also genau in den Fällen Vorteile bringen, die für Ray Tracing-Verfahren schwierige Aufgaben darstellen.

Das Verfahren zur Bilderzeugung mit adaptiven hierarchischen Dichtefunktionen läßt sich zusammenfassend so darstellen:

- diffus reflektierende Flächen bzw. Flächen mit diffus reflektierendem Anteil der BRDF werden in Teilflächen unterteilt, jeder Teilfläche wird eine Dichtefunktion zugewiesen,
- in mehreren Durchläufen werden mit Pfadverfolgung Bilder erzeugt, an diffus reflektierenden Flächen wird die zugehörige Dichte zur Generierung der Strahlrichtung verwendet, das aus der Richtung einfallende Licht wird zur Adaption der Dichtefunktion benutzt,
- nach jedem Durchlauf werden die Dichtefunktionen adaptiert und verfeinert,
- aus den Bildern der einzelnen Durchläufe wird ein Gesamtbild als nach Samplezahl gewichtetes Mittel berechnet.

## 3.4 Bewertung der Ergebnisse

Ergebnis von Monte Carlo-Ray Tracing ist die Generierung von Bildern, die einem Betrachter einen realistischen Eindruck der dargestellten Szene bieten sollen. Insofern ist ein wichtiges Kriterium zur Beurteilung eines Verfahrens der visuelle Eindruck eines Betrachters. Monte Carlo-Ray Tracing ist aber auch, wie oben dargestellt, die Anwendung eines numerischen Integrationsverfahrens, so daß man quantitative Aussagen über die Qualität der Berechnungen machen kann.

### 3.4.1 Fehlernorm

Wenn ein Referenzbild vorhanden ist, kann ein numerischer Fehler für das generierte Bild berechnet werden, indem man die Differenz dieser beiden Bilder in einer Norm mißt. Wendet man dafür z. B. eine der in Anhang B vorgestellten $L^p$-Normen an, dann wird die Differenz zweier Bilder als

$$\varepsilon_p = \left( \frac{1}{n_h \, n_w} \sum_{i=1}^{n_h} \sum_{j=1}^{n_w} |r(i,j) - p(i,j)|^p \right)^{\frac{1}{p}} \tag{3.6}$$

berechnet. Die Bilder haben eine Dimension von $n_h \cdot n_w$ Pixeln, $r(i,j)$ ist das Pixel in Zeile $i$ und Spalte $j$ des Referenzbildes, $p(i,j)$ das entsprechende Pixel des berechneten Bildes. Handelt es sich bei den berechneten Bildern um

Farbbilder, werden sie vor der Fehlerberechnung in Grauwertbilder konvertiert oder der in (3.6) benutzte Betragsoperator muß im verwendeten Farbmodell arbeiten.

#### 3.4.1.1 $L^2$-Norm

Es ist üblich, die Varianz oder Standardabweichung des Resultats als Qualitätsmaß für ein Monte Carlo-Verfahren zu verwenden. Beim Monte Carlo-Ray Tracing ist nicht ein einzelner Wert das Ergebnis der Berechnung sondern ein aus vielen Pixeln bestehendes Bild. Die Berechnung jedes einzelnen Pixels erfordert eine Integration nach (3.2), so daß jedes Pixel eine individuelle Varianz hat. Man kann dem berechneten Bild eine Varianz als Mittelwert der Varianzen der einzelnen Pixel zuweisen.

Existiert ein Referenzbild zu dem berechneten Bild, dann ist der nach (3.6) berechnete $L^2$-Fehler $\varepsilon_2$ eine Schätzung der über das Bild gemittelten Standardabweichung. Der quadrierte $L^2$-Fehler ergibt eine Schätzung der über alle Pixel gemittelten Varianz. Der absolute Fehler eines Pixels, berechnet mit $|r(i,j) - p(i,j)|$, kann als Schätzung der Standardabweichung angesehen werden. Quadriert man diesen Wert, erhält man eine Schätzung der Varianz. Bildet man wie in (3.6) den Mittelwert über alle Pixel, ergibt sich eine Schätzung der mittleren Varianz, deren Wurzel wiederum eine Schätzung der Standardabweichung darstellt. Aus diesen Gründen ist der $L^2$-Fehler das gebräuchliche Maß zur Bewertung von Monte Carlo-Ray Tracing-Verfahren.

Eine Anwendung des Verfahrens zeigt, daß bei gleicher Samplezahl die Benutzung adaptiver Dichtefunktionen zu deutlich besserer visueller Qualität führt (Abb. E.1). Betrachtet man aber den in der $L^2$-Norm gemessenen Fehler als Funktion der Samplezahl (Abb. E.2), erkennt man, daß in dieser Norm erst für viel größere Samplezahlen ein Absinken des Fehlers unter die Kurve für nicht-adaptives Ray Tracing erfolgt. Dies läßt sich erklären, wenn man die Art der Fehler in den generierten Bildern untersucht. Das nicht-adaptive Ray Tracing erzeugt überwiegend ein dunkles Rauschen, viele Pixel werden zu dunkel berechnet. Die adaptiven Verfahren zeigen ein deutlich schwächeres Rauschen, weshalb sie visuell auch als deutlich besser eingeschätzt werden. Es treten zusätzlich einige Ausreißer auf, die deutlich nach oben abweichen, diese Pixel sind um ein Vielfaches zu hell. Das wird erkennbar, wenn man den Fehler in der $L^\infty$-Norm betrachtet (Abb. E.3). Diese Norm mißt den maximalen Fehler, liefert also den Fehlerwert des Pixels mit der größten Abweichung zum Referenzbild. Da in der $L^2$-Norm große Abweichungen mit höherem Gewicht eingehen als kleine Abweichungen, führen diese größeren maximalen Fehler auch zu einem größeren $L^2$-Fehler.

Der größere maximale Fehler der adaptiven Verfahren resultiert aus dem Faktor (Gleichung (3.5))

$$2\pi \frac{\rho_{bd}(\mathbf{x}, \omega_o, \omega_j)\ \cos\theta_j}{p(u_j, v_j)}$$

mit dem das Gewicht an einem Trefferpunkt multipliziert wird. Wird ein Samplepaar $(u_j, v_j)$ mit einem kleinen Wert $p(u_j, v_j)$ erzeugt, führt das zu einer starken Erhöhung des Gewichts und damit zu den beobachteten Ausreißern. Solche Samplepaare haben eine geringe Wahrscheinlichkeit, treten also nur selten auf. Das stimmt mit der Beobachtung überein, daß in einem Bild mit 300·300 Pixeln maximal einige hundert Ausreißer auftreten. Mit zunehmender Samplezahl sollten sich solche Ausreißer ausmitteln, was im Diagramm (Abb. E.3) durch Absinken des Fehlers mit zunehmender Samplezahl erkennbar ist.

Nicht-adaptives Ray Tracing weist diese Ausreißer nicht auf, da bei Wahl einer WDF proportional zu $\rho_{bd}(\mathbf{x}, \omega_o, \omega) \cos\theta$ das Gewicht im Laufe eines Pfades von Trefferpunkt zu Trefferpunkt abnimmt.

#### 3.4.1.2 Modifizierte $L^2$-Norm

Alle Szenen sind so modelliert, daß bei der Darstellung des Bildes ein Radiance-Wert von 1 auf die maximale Helligkeit des Ausgabemediums abgebildet wird. Wie man in Abb. E.3 sieht, ist für niedrige Samplezahlen der maximale Fehler weit über 100, also um mehr als ein Hundertfaches höher als die maximal darstellbare Helligkeit. Also treten die zur Verschlechterung der $L^2$-Norm führenden Fehler in Pixeln mit einer Helligkeit auf, die gar nicht mehr darstellbar ist und somit in der visuellen Bewertung gar nicht erkennbar werden. Deshalb wird eine modifizierte $L^2$-Norm benutzt, die den darstellbaren Helligkeitsbereich berücksichtigt. Im Referenzbild und im generierten Bild werden alle Pixel, deren Helligkeit über dem maximal darstellbaren Wert liegt, auf die maximal darstellbare Helligkeit korrigiert. Danach wird nach (3.6) der $L^2$-Fehler berechnet. Dadurch wird erreicht, daß in die Berechnung des numerischen Fehlers nur darstellbare Pixelhelligkeiten eingehen. Das so modifizierte $L^2$-Fehlermaß korreliert gut mit dem visuellen Eindruck (Abb. E.4).

### 3.4.2 Qualität und Effizienz

In Abschnitt 2.3 ist bereits gezeigt worden, daß es kein allgemein einsetzbares Effizienzmaß zur Bewertung von Monte Carlo-Verfahren gibt. Im folgenden sollen folgende Begriffe zur Beurteilung eines Monte Carlo-Verfahrens benutzt werden:

**Qualität** Für die Beurteilung der numerischen Qualität eines Verfahrens wird der Fehler als Funktion der Samplezahl dargestellt. Die pro Sample benötigte Zeit hängt von der Implementierung und Hardware ab. Die Betrachtung der Qualität zeigt, ob unabhängig von der vorhandenen Implementierung und Hardware eine prinzipielle Verbesserung des berechneten Ergebnisses erreichbar ist.

**Effizienz** Die tatsächliche Effizienz eines Verfahrens erfordert natürlich eine Berücksichtigung der Rechenzeit. Zur Beurteilung wird der Fehler in einem Diagramm als Funktion der Zeit aufgetragen. Ein Verfahren ist dann

effizienter, wenn es in gleicher Zeit zu einem kleineren Fehler führt (Abb. E.5) oder denselben Fehler in kürzerer Zeit erreicht (Abb. E.6). Bei der Beurteilung dieser Diagramme ist zu beachten, daß eine verhältnismäßig kleine Verringerung des Fehlers bereits einer deutlichen Zeiteinsparung entsprechen kann.

## 3.5 Ergebnisse

Um Aussagen über das Verhalten des implementierten Verfahrens machen zu können, wurden Bilder mit unterschiedlichen Parametrisierungen berechnet. Ziel war es, das Verhalten des Verfahrens für unterschiedliche Basisfunktionen, Tiefe der MSA und Szenen-Unterteilungen sowie hierarchische bzw. nichthierarchische Adaption zu untersuchen. Diese Berechnungen haben insgesamt mehrere Monate an CPU-Zeit benötigt. Die Ergebnisse sind im Anhang E zusammengefaßt.

Zum Test des Verfahrens wurden drei Szenen benutzt:

**Szene 1** In der ersten Szene sind fast alle Flächen direkt beleuchtet. Als Referenz wurde mit nicht-adaptivem Monte Carlo-Ray Tracing ein Bild mit 64000 Samples pro Pixel berechnet (Abb. E.7).

**Szene 2** Die zweite Szene enthält einen signifikanten Anteil indirekter Beleuchtung. Die Lichtquellen strahlen gegen die Decke, so daß die meisten Flächen der Szene ihr Licht erst nach mindestens einer Reflexion erhalten. Die Referenz wurde ebenfalls mit nicht-adaptivem Monte Carlo-Ray Tracing unter Einsatz von 96000 Samples pro Pixel berechnet (Abb. E.20).

**Szene 3** Die ersten beiden Szenen sind geschlossen, Licht strahlt auf jeden Punkt aus der ganzen Hemisphäre ein. Die dritte Szene wurde aus der zweiten durch Weglassen der vorderen, hinteren und rechten Wand erzeugt. Dadurch kann das Verhalten des Verfahrens untersucht werden, wenn die Beleuchtung nur über Teile der Hemisphäre einfällt. Die Referenzlösung wurde nicht-adaptiv mit 96000 Samples pro Pixel berechnet (Abb. E.41).

Für einige Untersuchungen ist es nötig, Aussagen über die mit einem bestimmten Ansatz prinzipiell erreichbare Qualität zu machen. Um die Qualität eines Verfahrens, insbesondere die mit einer bestimmten adaptierten Dichtefunktion erreichbare Qualität, zu messen, wurde jeweils der Fehler eines mit 100 Samples pro Pixel berechneten Bildes benutzt. Im Gegensatz zum oben beschriebenen Bilderzeugungsverfahren werden die in den ersten Durchläufen der Berechnung verschossenen Samples nicht für das Ergebnisbild verwendet. Sie dienen nur der Adaption der Dichtefunktionen. Damit wird erreicht, daß das erzeugte Bildes ausschließlich von der adaptierten Dichtefunktion abhängt.

### 3.5.1 Konvergenz

Die ersten Untersuchungen betreffen das Konvergenzverhalten der adaptierten Dichtefunktionen. Die mit adaptiven Dichtefunktionen maximal erreichbare Qualität ist abhängig von der verwendeten Funktionsbasis, der Tiefe der benutzten MSA und der Unterteilung der Flächen in der Szene. Außerdem muß eine ausreichende Zahl von Samples für die Adaption eingesetzt werden, damit die Dichtefunktionen gegen das Proximum konvergieren. Dieses Konvergenzverhalten kann man untersuchen, indem man die Bildqualität als Funktion der für die Adaption benutzten Samplezahl aufträgt. Die Bildqualität wird wie oben beschrieben als Fehler eines in einem zweiten Durchlauf mit 100 Samples pro Pixel erzeugten Bildes bestimmt.

Die Ergebnisse dieser Berechnungen für Szene 2 sind in den Abb. E.21, E.22 (ohne Stratifikation) sowie E.23, E.24 (mit Stratifikation) dargestellt. Die Abb. E.21 und E.23 zeigen, wie sich die Konvergenz bei gegebener Szenenunterteilung in Abhängigkeit von der Skala der MSA verhält. Aus den Abb. E.22 und E.24 ist erkennbar, wie sich bei gegebener Skala der MSA (hier Skala 3) eine Verfeinerung der Szenen-Unterteilung auswirkt. Aus den Diagrammen läßt sich ablesen, daß im allgemeinen ungefähr 50 Samples zur Adaption ausreichend sind, um Konvergenz zu erreichen. Für höhere Skalen, feinere Szenen-Unterteilung oder höhergradige Basisfunktionen sind aber teilweise deutlich mehr Adaptions-Samples erforderlich. Eine weitere wichtige Eigenschaft, die insbesondere für die Verfahren ohne Stratifikation sichtbar wird, ist, daß eine zu geringe Adaptions-Samplezahl zu einer schlechten Qualität der adaptierten Dichte führt, teilweise sogar zu schlechteren Resultaten als nicht-adaptives Ray Tracing.

Die Diagramme machen auch deutlich, daß die Benutzung höherer Skalen bzw. feinerer Szenen-Unterteilungen nur bis zu einer gewissen Grenze zu einer Verbesserung der Qualität führt. Weitere Verfeinerungen ergeben keine Vorteile mehr. Eine genauere Untersuchung dieser Grenzen erfolgt im nächsten Abschnitt.

### 3.5.2 Maximal erreichbare Qualität

Die mit adaptiven Dichtefunktionen maximal erreichbare Qualität ist abhängig von der verwendeten Funktionsbasis, der Tiefe der benutzten MSA und der Unterteilung der Flächen in der Szene. Aus den Diagrammen zur Konvergenz im vorigen Abschnitt ist erkennbar, daß 500 Samples pro Pixel zur Adaption im allgemeinen ausreichend sind, Konvergenz der adaptierten Dichtefunktion zu erreichen. Die Endpunkte der zur Untersuchung der Konvergenz betrachteten Diagramme stellen somit ein Maß für die maximal mit einer bestimmten Parametrisierung erreichbaren Qualität dar.

#### 3.5.2.1 Szene 1

Abb. E.8 zeigt die maximal erreichbare Qualität für Verfahren ohne Stratifikation. Für höhere Skalen verbessert sich die Qualität, ab Skala 3 beginnen die

Kurven abzuflachen. Während Haar-MSA sowie lineare und quadratische B-Spline-MSA für höhere Skalen gegen fast identische Werte konvergieren, führt der Einsatz kubischer B-Splines zu schlechteren Ergebnissen, teilweise schlechter als nicht-adaptives Ray Tracing. Für kubische Basisfunktionen ist eine Verschlechterung der Qualität mit zunehmender Szenen-Unterteilung erkennbar. Für Basisfunktionen niedrigen Grades werden keine großen Unterschiede zwischen verschiedenen Szenen-Unterteilungen sichtbar.

Mit Stratifikation werden für Szene 1 kaum Qualitätsverbesserungen im Vergleich zum nicht-adaptiven Ray Tracing erzielt (Abb. E.9). Der Einsatz höhergradiger Basisfunktionen, insbesondere kubischer B-Splines, führt teilweise sogar zu deutlichen Verschlechterungen. Es zeigen sich keine großen Abhängigkeiten der erreichbaren Qualität von der Skala der MSA oder der Szenen-Unterteilung.

#### 3.5.2.2 Szene 2

In Szene 2 sind mit dem adaptiven Ansatz offensichtlich viel deutlichere Qualitätsverbesserungen erreichbar (Abb. E.25, E.26). Die Verfahren ohne und mit Stratifikation der direkten Beleuchtung verhalten sich sehr ähnlich. Die besten Qualitäten werden mit der linearen B-Spline-MSA erzielt, dicht gefolgt von der Haar-MSA, während quadratische und kubische B-Spline-MSA schlechtere Qualitäten erreichen. Die Qualität steigt allgemein mit höherer Skala, auch hier wird ab Skala 3 ein Abflachen der Kurven sichtbar. Die Qualität verbessert sich von einer $2 \cdot 2$-Szenen-Unterteilung zu einer $4 \cdot 4$-Unterteilung. Feinere Unterteilungen führen nicht mehr zu deutlichen Verbesserungen.

#### 3.5.2.3 Szene 3

In Szene 3 sind die größten relativen Verbesserungen erkennbar (Abb. E.42, E.43). Wieder ist zu beobachten, daß mit höheren Skalen bessere Qualität erreichbar ist, wobei wieder bei Skala 3 eine, hier aber nicht so stark ausgeprägte, Abflachung der Kurven einsetzt. Bis zu einer $8 \cdot 8$-Szenen-Unterteilung ist eine Verbesserung erkennbar, eine weitere Unterteilung bringt keine Vorteile mehr. In allen Fällen werden mit linearer B-Spline-MSA die besten Resultate erzielt, Haar-MSA und quadratische B-Spline-MSA erreichen fast identische Ergebnisse, deutlich schlechter ist wiederum die kubische B-Spline-MSA.

### 3.5.3 Qualität

Die im vorigen Abschnitt untersuchte maximal erreichbare Qualität erlaubt Aussagen darüber, welche Dichtefunktionen ohne Berücksichtigung des Aufwands für die Adaption zu den besten Ergebnissen führen, zeigt also einen Grenzwert für die erreichbare Qualität. Im oben vorgestellten Bilderzeugungsverfahren gehen aber auch die während der Adaptions-Durchläufe generierten Samples in die Bildberechnung ein. Die Qualität des Gesamtverfahrens bestimmt man also aus den Bildern, die die Resultate aller Durchläufe benutzen.

#### 3.5.3.1 Szene 1

Eine Beurteilung der visuellen Qualität ermöglichen die in den Abb. E.10, E.11 (ohne Stratifikation) sowie E.13, E.14 (mit Stratifikation) gezeigten Bilder. Eine Betrachtung der Samplezahl-Fehler-Kurven (Abb. E.12, E.15) ergibt, daß für diese Szene nur in wenigen Fällen, nämlich Haar-MSA und lineare B-Spline-MSA ohne Stratifikation, eine geringe Qualitätsverbesserung erzielt werden kann. Dies entspricht auch dem visuellen Eindruck, in den Bildern sind keine Qualitätsunterschiede erkennbar. Dieses Resultat ist auch aufgrund der bereits oben getroffenen Feststellung zu erwarten, daß die adaptiven Dichtefunktionen in dieser Szene kaum Vorteile gegenüber nicht-adaptivem Ray Tracing erreichen.

#### 3.5.3.2 Szene 2

Deutlich bessere Resultate lassen sich für die zweite Szene erzielen. Die Betrachtung der erzeugten Bilder (Abb. E.27, E.28 sowie E.30, E.31) zeigt insbesondere für die Verfahren ohne Stratifikation eine Verringerung des Rauschens bei Einsatz der Haar-MSA, linearen B-Spline-MSA und in geringerem Maß für die quadratische B-Spline-MSA. Dies entspricht den Samplezahl-Fehler-Kurven (Abb. E.29, E.32). Haar-MSA und lineare B-Spline-MSA liefern fast identische Qualitätsverbesserungen, die quadratische B-Spline-MSA führt zu Resultaten knapp unterhalb des nicht-adaptiven Verfahrens, während der Einsatz einer kubischen B-Spline-MSA meistens zu einer Verschlechterung der Bildqualität führt. Unterschiedliche Szenen-Unterteilungen haben keinen großen Einfluß auf die Ergebnisse. Nur die Haar-MSA zeigt im Verfahren mit Stratifikation für feinere Szenen-Unterteilungen eine Verbesserung der Qualität.

#### 3.5.3.3 Szene 3

Die dritte Szene führt zu ähnlichen Ergebnissen wie die zweite. In den Bildern (Abb. E.44, E.45 sowie E.47, E.48) ist bis zu quadratischen Basisfunktionen eine Abnahme des Rauschens sichtbar. Haar-MSA und lineare B-Spline-MSA liefern fast identische Resultate und bringen die größten Verbesserungen (Abb. E.46, E.49). Die Qualität der kubischen B-Spline-MSA liegt ungefähr gleichauf mit dem nicht-adaptiven Verfahren. Die mit quadratischer B-Spline-MSA erreichbare Qualität liegt ziemlich genau in der Mitte. Für diese Szene sind kaum Abhängigkeiten von der Szenen-Unterteilung erkennbar.

### 3.5.4 Effizienz

Ein Vergleich der Effizienz verschiedener Ansätze läßt sich durch Betrachtung von Fehler-Zeit-Kurven durchführen. Ein Verfahren ist effizienter, wenn seine Kurve unterhalb derjenigen des Vergleichsverfahrens liegt.

#### 3.5.4.1 Szene 1

Die entsprechenden Fehler-Zeit-Kurven für Szene 1 sind in den Abb. E.16, E.17 (ohne Stratifikation) sowie E.18, E.19 (mit Stratifikation) dargestellt. Die Abb. E.16 und E.18 erlauben einen Vergleich der Effizienz verschiedener Basisfunktionen bei gegebener Szenen-Unterteilung, während aus den Abb. E.17 und E.19 erkennbar wird, wie sich bei gegebener Basis die Szenen-Unterteilung auf die Effizienz auswirkt.

Betrachtet man die ohne Stratifikation berechneten Ergebnisse, sieht man, daß die Benutzung der Haar-MSA und linearer B-Spline-MSA zu ähnlichen Ergebnissen wie nicht-adaptives Ray Tracing führt. Nur für die Haar-MSA sind in zwei Fällen kleine Verbesserungen gegenüber dem nicht-adaptiven Ray Tracing feststellbar. Deutliche Abhängigkeiten der Effizienz von der Szenen-Unterteilung sind in den Abb. E.17 und E.19 nicht erkennbar. Die feinste Szenen-Unterteilung führt oftmals zur schlechtesten Effizienz.

Mit Stratifikation der direkten Beleuchtung erreicht für diese Szene keines der Verfahren eine höhere Effizienz als nicht-adaptives Ray Tracing. Die Benutzung der Haar-MSA und linearer B-Spline-MSA führt bei nicht zu feiner Szenen-Unterteilung zu ähnlicher Effizienz wie das nicht-adaptive Ray Tracing (Abb. E.18). Auffällig ist, daß die Rangfolge der Effizienz genau dem Grad der Basisfunktionen entspricht. Eine deutliche Abhängigkeit der Effizienz von der Szenen-Unterteilung läßt sich nur bei Benutzung quadratischer B-Spline-MSAs erkennen (Abb. E.19). Die Effizienz der Verfahren nimmt in diesem Fall mit feinerer Unterteilung deutlich ab.

In Szene 1 sind keine signifikanten Verbesserungen der Effizienz erreicht worden, in den meisten Fällen waren die Verfahren deutlich ineffizienter. Betrachtet man die Geometrie von Szene 1, wird deutlich, daß an jedem Punkt der Szene Licht ziemlich gleichmäßig aus allen Richtungen einfällt. In solchen Konstellationen ist durch den Einsatz adaptiver Dichtefunktionen keine Verbesserung erreichbar. Dies hat sich bereits in Abschnitt 3.5.2.1 gezeigt, da für diese Szene nur geringe Qualitätsverbesserungen erreichbar waren.

#### 3.5.4.2 Szene 2

Für Verfahren ohne Stratifikation ergibt der Einsatz der Haar-MSA und der linearen B-Spline-MSA eine deutliche Steigerung der Effizienz, wobei bei höheren Samplezahlen die Haar-MSA noch etwas bessere Ergebnisse liefert (Abb. E.33). Der Einsatz höhergradiger Basisfunktionen lohnt sich nicht, die Effizienz ist deutlich schlechter als die des nicht-adaptiven Verfahrens. Aus Abb. E.34 ist erkennbar, daß das Optimum bei relativ feinen $8 \cdot 8$ bzw. $16 \cdot 16$-Szenen-Unterteilungen erreicht wird.

Bei den Verfahren mit Stratifikation ist ebenfalls für die Haar-MSA und die lineare B-Spline-MSA eine Effizienzsteigerung sichtbar (Abb. E.35). Bei feinerer Szenen-Unterteilung und höheren Samplezahlen ist die Haar-MSA am effizientesten. Die quadratische B-Spline-MSA kommt zwar näher an die Effizienz des

| | Zeit [s] | | |
|---|---|---|---|
| Fehler | nicht-adaptiv | Haar-Basis | Faktor |
| 0.286 | 273 | 55 | 4.96 |
| 0.256 | 595 | 114 | 5.20 |
| 0.227 | 845 | 232 | 3.64 |
| 0.181 | 1657 | 454 | 3.65 |
| 0.143 | 2889 | 927 | 3.12 |
| 0.123 | 4121 | 1423 | 2.90 |
| 0.108 | 5564 | 1937 | 2.87 |
| 0.098 | 6746 | 2441 | 2.76 |

Tabelle 3.1: Verringerung der Rechenzeit, Szene 2, ohne Stratifikation

nicht-adaptiven Ray Tracing heran, der Einsatz höhergradiger Basisfunktionen lohnt sich aber auch hier nicht. Das Optimum wird hier mit den mittleren $4 \cdot 4$ bzw. $8 \cdot 8$-Unterteilungen erreicht (Abb. E.36), wobei sich nur geringe Abhängigkeiten zwischen Effizienz und Szenen-Unterteilung beobachten lassen.

Wie oben bei der Beschreibung des Verfahrens bereits vermutet, führt die Benutzung adaptiver Dichtefunktionen dann zu Effizienzsteigerungen, wenn indirekte Beleuchtung einen signifikanten Anteil an der Gesamthelligkeit hat, wie es in dieser Szene der Fall ist.

#### 3.5.4.3 Szene 3

Für Verfahren ohne Stratifikation führt die Benutzung der Haar-MSA bzw. der linearen B-Spline-MSA zu deutlichen Effizienzsteigerungen (Abb. E.50). Auch hier lohnt sich der Einsatz höhergradiger Basisfunktionen nicht. Das Optimum wird bis zum Grad 2 mit der $8 \cdot 8$-Szenen-Unterteilung erreicht, für kubische B-Splines erst mit $16 \cdot 16$ (Abb. E.51). Der deutlichste Abstand ist für den Übergang von einer $2 \cdot 2$ zu feineren Unterteilungen erkennbar.

Mit Stratifikation zeigen Haar-MSA und lineare B-Spline-MSA deutliche Verbesserungen, in dieser Szene läßt sich auch mit quadratischer B-Spline-MSA bessere Effizienz erzielen (Abb. E.52). Die Abhängigkeit von der Szenen-Unterteilung ist gering, das Optimum erreichen alle Verfahren mit einer $8 \cdot 8$-Szenen-Unterteilung (Abb. E.53).

### 3.5.5 Beschleunigung

Bei der Beurteilung eines Bilderzeugungsverfahrens ist die erreichbare Beschleunigung eines der wichtigsten Kriterien. Hier geht es also um die Einsparung von Rechenzeit bei gleicher Qualität des erzeugten Bildes. Der erreichbare Beschleunigungsfaktor bei Benutzung der Haar-MSA ist für Szene 2 in den Tabellen 3.1 (ohne Stratifikation) und 3.2 (mit Stratifikation) berechnet. Dieselben Ergebnisse für Szene 3 finden sich in den Tabellen 3.3 (ohne Stratifikation) und 3.4 (mit Stratifikation).

| | Zeit [s] | | |
|---|---|---|---|
| Fehler | nicht-adaptiv | Haar-Basis | Faktor |
| 0.205 | 1021 | 355 | 2.88 |
| 0.175 | 2224 | 753 | 2.95 |
| 0.150 | 3638 | 1511 | 2.41 |
| 0.108 | 8118 | 3110 | 2.61 |
| 0.075 | 17009 | 6383 | 2.66 |
| 0.060 | 27441 | 9707 | 2.83 |
| 0.051 | 39583 | 13051 | 3.03 |
| 0.045 | 50990 | 16375 | 3.11 |

Tabelle 3.2: Verringerung der Rechenzeit, Szene 2, mit Stratifikation

| | Zeit [s] | | |
|---|---|---|---|
| Fehler | nicht-adaptiv | Haar-Basis | Faktor |
| 0.202 | 389 | 24 | 16.08 |
| 0.179 | 563 | 50 | 11.16 |
| 0.163 | 677 | 107 | 6.35 |
| 0.130 | 1190 | 218 | 5.47 |
| 0.104 | 1941 | 445 | 4.36 |
| 0.089 | 2614 | 678 | 3.86 |
| 0.080 | 3289 | 913 | 3.60 |
| 0.073 | 3868 | 1144 | 3.38 |

Tabelle 3.3: Verringerung der Rechenzeit, Szene 3, ohne Stratifikation

| | Zeit [s] | | |
|---|---|---|---|
| Fehler | nicht-adaptiv | Haar-Basis | Faktor |
| 0.140 | 576 | 123 | 4.68 |
| 0.122 | 921 | 266 | 3.46 |
| 0.107 | 1301 | 546 | 2.38 |
| 0.075 | 2618 | 1188 | 2.20 |
| 0.050 | 6218 | 2485 | 2.50 |
| 0.039 | 10629 | 3792 | 2.80 |
| 0.032 | 15424 | 5103 | 3.02 |
| 0.028 | 20269 | 6408 | 3.16 |

Tabelle 3.4: Verringerung der Rechenzeit, Szene 3, mit Stratifikation

In beiden Szenen ergibt das Verfahren ohne Stratifikation für niedrige Samplezahlen hohe Faktoren. Szene 3 ist so konstruiert, daß für viele Punkte der Szene aus großen Bereichen der Hemisphäre keine Beleuchtung einfällt. Eine adaptive Dichtefunktion, die die Samples entsprechend verteilt, sollte hier deutliche Vorteile bringen. Wie man sieht, ist hier tatsächlich eine Beschleunigung bis zum Faktor 16 erreicht worden. Für höhere Samplezahlen liegt die erzielte Beschleunigung für beide Szenen mit und ohne Stratifikation ungefähr bei Faktor 3.

### 3.5.6 Hierarchische Dichtefunktionen

Eine weitere wichtige Untersuchung betrifft die Frage, ob sich die Adaption der Dichtefunktionen über mehrere Hierachiestufen lohnt oder ob es effizienter ist, die Dichtefunktion auf einer fixen Stufe zu adaptieren, also nicht hierarchisch zu arbeiten.

Bei der hierarchischen Konstruktion einer Funktions-Approximation startet man mit der Konstruktion einer groben Darstellung und konzentriert die Arbeit dann auf die Bereiche, die aufgrund eines hohen Absolutwertes der Funktion bzw. aufgrund eines hohen Gradienten als wichtig eingestuft werden. Diese Art des Vorgehens ist inhärent in der hierarchischen Adaption der Dichtefunktionen enthalten. Da mehr Samples in den Bereichen generiert werden, wo die Dichtefunktion hohe Werte hat, ist die Approximation in diesen Bereichen automatisch besser. Dieser Effekt muß sich insbesondere bemerkbar machen, wenn die Zahl der Samples in den frühen Adaptions-Durchläufen minimiert wird. Ein Nachteil des hierarchischen Vorgehens könnte daraus resultieren, daß von Stufe zu Stufe die Basisdarstellung der Dichtefunktion neu berechnet wird, während man beim nicht-hierarchischen Arbeiten auf Grundlage der in Abschnitt 2.4.5 gegebenen Hinweise die Samples akkumulieren kann.

Zuerst soll untersucht werden, ob die aus hierarchischer Adaption generierte Dichtefunktion eine bessere Qualität liefert als eine nicht-hierarchisch erzeugte Dichte. Hierfür wurde mit möglichst geringen Samplezahlen in 3 Durchläufen eine Adaption durchgeführt und dann zur Qualitätsbestimmung im vierten Durchlauf ein Bild mit 100 Samples pro Pixel erzeugt. Um den Einfluß der Verteilung der Samples auf die verschiedenen Durchläufe zu erkennen (Abschnitt 2.5.1), wurde in einem Versuch gleiche Samplezahl in jedem Durchlauf eingesetzt, so daß zur Adaption 1-1-1, 2-2-2, 3-3-3, ... Samples im jeweiligen Durchlauf benutzt wurden. Im zweiten Versuch wurde die Zahl der Samples von Durchlauf zu Durchlauf verdoppelt, was zu Sampleverteilungen von 1-2-4, 2-4-8, 3-6-12, ... führt. Die Ergebnisse dieser Versuche für Szene 2 finden sich in Abb. E.37 (ohne Stratifikation) und in Abb. E.39 (mit Stratifikation direkter Beleuchtung). Bei Benutzung der Haar-MSA und der 1-2-4-Sampleverteilung sind hierarchische und nicht-hierarchische Adaption gleichwertig. Für andere Sampleverteilungen und lineare Basisfunktionen führt nicht-hierarchische Adaption bei gleichem Zeitaufwand für die Adaption zu einer besseren Qualität der Dichtefunktion als hierarchisches Vorgehen.

Wie hierarchische Adaption die Effizienz des Gesamtverfahrens beeinflußt, ist in Abb. E.38 (ohne Stratifikation) und E.40 erkennbar. Für die Haar-MSA und lineare B-Spline-MSA sind nur geringe Unterschiede sichtbar, die nicht-hierarchische Adaption liefert etwas bessere Ergebnisse. Im Falle der höhergradigen B-Spline-MSAs ist der nicht-hierarchische Ansatz deutlich effizienter.

## 3.6 Zusammenfassung

Die verschiedenen Varianten des Monte Carlo-Ray Tracing sind Verfahren zur Lösung der Bildsynthese-Gleichung, die auf dem Prinzip der Monte Carlo-Integration basieren. Aus der Theorie der Monte Carlo-Integration ist bekannt, daß eine Verbesserung der Resultate durch Verwendung von Dichtefunktionen erreicht werden kann, die möglichst exakt den Verlauf des Integranden annähern. Basierend auf dem mathematischen Ansatz zur Verwendung adaptiver hierarchischer Dichtefunktionen aus Kapitel 2 wurde ein Monte Carlo-Ray Tracing mit Pfadverfolgung realisiert, das in mehreren Durchläufen arbeitet und adaptiv optimierte Dichtefunktionen zur Effizienzsteigerung benutzt.

Es konnte gezeigt werden, daß der Einsatz adaptiver Dichtefunktionen zu einer Effizienzverbesserung führen kann. Der hier benutzte mathematische Ansatz erlaubt die Erprobung verschiedener Basisfunktionen und hierarchische bzw. nicht-hierarchische Darstellungen. Es wurde gezeigt, daß sich der Einsatz stückweise konstanter und stückweise linearer Basisfunktionen lohnt, während höhergradige, stückweise quadratische bzw. kubische, Basisfunktionen keinen Vorteil bringen. Hierarchische Adaption hat keine Verbesserung der Resultate gezeigt, im Falle der höhergradigen Basisfunktionen führte sie sogar zu einer Verschlechterung. Neben der Wahl der Funktionsbasen hat das Verfahren weitere Parameter, nämlich Szenen-Unterteilung und Tiefe der MSA. Es hat sich gezeigt, daß die Wahl dieser Parameter für die Effizienz nicht kritisch ist, so daß man beim Einsatz des Verfahrens nicht mit großem Aufwand zur Parameter-Einstellung rechnen muß.

Bemerkenswert an dem hier realisierten Ansatz ist, daß im Gegensatz zu anderen Verfahren eine tatsächliche Steigerung der Effizienz gemessen an der benötigten Rechenzeit beobachtet werden kann. Um eine tatsächliche Effizienzverbesserung zu erzielen, muß darauf geachtet werden, daß die Adaption der Dichtefunktionen und die Erzeugung von Samples nach diesen Dichtefunktionen mit geringstmöglichem Mehraufwand erreicht werden.

# Kapitel 4

# Anwendung von Interpolationsverfahren zur Radiosity-Rekonstruktion

Eine Radiosity-Berechnung liefert als Ergebnis eine Approximation $\hat{B}$ der Radiosity-Funktion $B$. Meistens ist diese Approximation nicht geeignet, sie direkt darzustellen, weil z. B. die Verwendung stückweise konstanter Basisfunktionen zu sichtbaren Artefakten führt. Der Rekonstruktions-Schritt der Radiosity-Pipeline bildet die Lösung $\hat{B}$ in eine andere, für die Darstellung besser geeignete Approximation $\tilde{B}$ ab. In diesem Kapitel wird ein Ansatz vorgestellt, die Rekonstruktion als Interpolationsaufgabe anzusehen und mit bekannten Verfahren zur Interpolation von Daten über regelmäßigen Gittern bzw. von Streudaten zu lösen.

## 4.1 Radiosity-Verfahren

Radiosity-Verfahren [29, 13, 14, 12] berechnen fotorealistische Bilder auf der Grundlage der Radiosity-Gleichung. Man erhält die Radiosity-Gleichung aus der allgemeinen Bildsynthese-Gleichung, indem man alle Lichtinteraktionen auf rein diffuse Reflexion einschränkt. Dadurch entfallen in der Bildsynthese-Gleichung alle Richtungsabhängigkeiten und sie vereinfacht sich zur Radiosity-Gleichung:

$$B(\mathbf{x}) = E(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int\limits_{\mathbf{y} \in S} B(\mathbf{y})\, G(\mathbf{x}, \mathbf{y})\, V(\mathbf{x}, \mathbf{y})\, \mathrm{d}A_{\mathbf{y}} \tag{4.1}$$

Integriert wird über alle Flächen der Szene. $B(\mathbf{x})$ ist die Radiosity am Flächenpunkt $\mathbf{x}$, $E(\mathbf{x})$ die Eigenemission an diesem Punkt, $\rho(\mathbf{x})$ die diffuse Reflektivität und $\mathrm{d}A_{\mathbf{y}}$ die differentielle Fläche am Punkt $\mathbf{y}$. $V(\mathbf{x}, \mathbf{y})$ ist die Sichtbarkeitsfunktion, sie ist 1, wenn die Sicht zwischen den Punkten $\mathbf{x}$ und $\mathbf{y}$ unverdeckt ist, sonst 0.

$$G(\mathbf{x}, \mathbf{y}) = \frac{\cos\theta_{\mathbf{x}}\, \cos\theta_{\mathbf{y}}}{d(\mathbf{x}, \mathbf{y})^2}$$

ist ein Geometrie-Term. $\theta_{\mathbf{x}}$ und $\theta_{\mathbf{y}}$ sind die Winkel zwischen der Verbindungsstrecke der Punkte $\mathbf{x}$ und $\mathbf{y}$ sowie dem Normalenvektor an dem jeweiligen Punkt.

Eigenemission $E$ und Reflexionskoeffizient $\rho$ jeder Fläche werden bei der Modellierung der Szene festgelegt, gesucht ist die Radiosity-Funktion $B$. Für die durch (4.1) definierte Integralgleichung ist im allgemeinen keine geschlossene Lösung bestimmbar. Eine Möglichkeit, zu einer näherungsweisen Lösung zu gelangen, ist die Anwendung von Finite-Elemente-Methoden [71]. Im folgenden wird hier der Ansatz entwickelt, zu dem man bei Benutzung stückweise konstanter Basisfunktionen gelangt. Die aus der Benutzung allgemeiner Basisfunktionen resultierenden Ansätze werden in [15] vorgestellt.

Die Benutzung stückweise konstanter Basisfunktionen bedeutet, daß die gesamte Szene in $n$ Teilflächen $A_1, \ldots, A_n$ unterteilt wird, die *Patches* oder *Elemente* genannt werden. Eigenemission, Reflexionskoeffizient und Radiosity werden über jedes Element als konstant vorausgesetzt, so daß pro Element jeweils nur ein Wert $E_i$, $\rho_i$, $B_i$ benötigt wird. Dadurch wird das Problem in einen finiten Funktionsraum abgebildet und man erhält eine vereinfachte Radiosity-Gleichung

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j \, F_{ij} \tag{4.2}$$

mit den Formfaktoren

$$F_{ij} = \frac{1}{\pi \, A_i} \int\limits_{\mathbf{x} \in A_i} \int\limits_{\mathbf{y} \in A_j} G(\mathbf{x}, \mathbf{y}) \, V(\mathbf{x}, \mathbf{y}) \, \mathrm{d}A_{\mathbf{y}} \, \mathrm{d}A_{\mathbf{x}} \tag{4.3}$$

die den Energietransport zwischen zwei Elementen quantifizieren. Gleichung (4.2) gilt für jedes Element $i$, man erhält also ein lineares Gleichungssystem, dessen Lösung die gesuchten Werte $B_i$ liefert. Dadurch wird eine stückweise konstante Näherung $\hat{B}$ bestimmt:

$$\hat{B}(\mathbf{x}) := B_i \quad \text{für} \quad \mathbf{x} \in A_i$$

Diese Definition von $\hat{B}$ ist eindeutig, da die $A_i$ eine disjunkte Zerlegung der Szene darstellen.

Abb. 4.1 zeigt die wesentlichen Schritte, die ein Radiosity-Verfahren zur Berechnung eines Bildes durchläuft [15]. Der erste Schritt besteht aus der Modellierung der Szene bzw. dem Einlesen der Szenenbeschreibung. Hier werden die Geometrie der Objekte und deren optische Eigenschaften definiert. Die Schritte 2 bis 4 stellen die eigentliche Berechnung der Radiosity-Lösung $\hat{B}$ dar, sie bestehen aus Unterteilung der Flächen in Elemente, Berechnung der Formfaktoren und Lösung des dadurch definierten linearen Gleichungssystems. Die Schritte 2 bis 4 sind in vielen Verfahren zu einem Schritt verschmolzen oder werden zyklisch mehrfach durchlaufen, was durch die rückwärts gerichteten Pfeile symbolisiert wird. Die direkte Benutzung der Lösung $\hat{B}$ für die Darstellung der Szene führt, insbesondere bei einer stückweise konstanten Lösung, zu unbefriedigenden Ergebnissen, da die visuelle Wahrnehmung des Menschen sehr empfindlich auf Unstetigkeiten reagiert. Deshalb wird vor der Darstellung des Ergebnisses eine

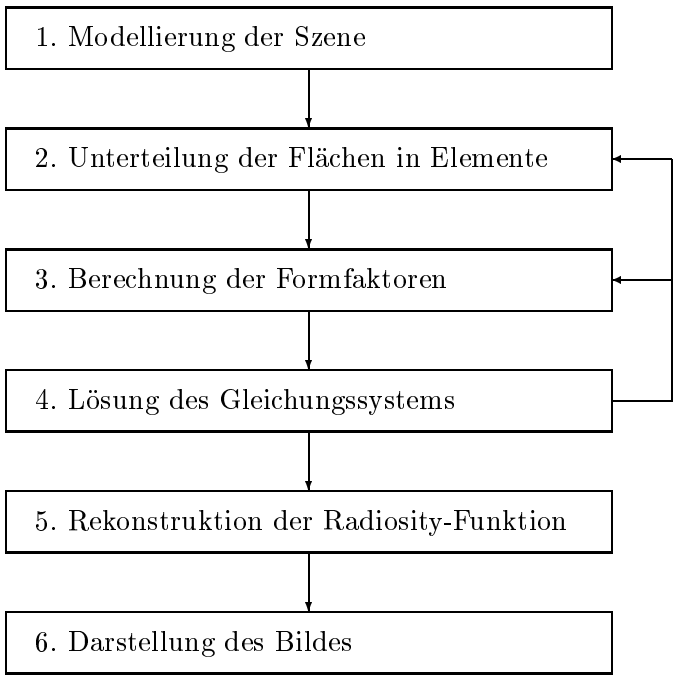



Abbildung 4.1: Radiosity-Pipeline (nach [15])

Umwandlung der Lösung $\hat{B}$ in eine neue Darstellung $\tilde{B}$ durchgeführt, wobei an die neue Darstellung die Anforderung gestellt wird, insbesondere die visuelle Qualität des erzeugten Bildes zu verbessern. Ein großer Vorteil von Radiosity-Verfahren ist, daß die berechnete Lösung unabhängig vom Betrachterstandpunkt ist. Deshalb kann die Bilddarstellung (Schritt 6) beliebig oft für neue Betrachterstandpunkte wiederholt werden.

## 4.2 Bestehende Ansätze zur Rekonstruktion

Die meisten Radiosity-Programme berechnen stückweise konstante Lösungen über regelmäßigen Gittern oder adaptiven, Quadtree-artigen Unterteilungen der Flächen in der Szene. Die Radiosity eines Flächenstücks wird von diesen Programmen als konstant angesetzt.

Es gibt viele bekannte Interpolationsverfahren für reguläre Gitter, am häufigsten wird bilineare Interpolation angewandt. Um bestimmte Nachteile, wie z. B. Mach-Bänder, zu vermeiden, werden manchmal Interpolationsverfahren höherer Ordnung benutzt.

Quadtree-Unterteilungen erzeugen eine ungleichmäßige Verteilung der Datenpunkte auf der Fläche. Auch hier ist bilineare Interpolation innerhalb der Quadtree-Zellen möglich. Dafür ist erst eine Umrechnung der Radiosity auf die Eckpunkte der Quadtree-Zellen nötig, z. B. durch Mittelung der Radiosity-Werte umliegender Zellen. Ein sichtbares Artefakt dieses Ansatzes sind aber

T-Stellen-Probleme an den Kanten, wo verschieden fein unterteilte Zellen aneinandergrenzen. Eine übliche Methode zur Behebung dieser Probleme ist, eine Triangulierung zu erzeugen und innerhalb dieser bilinear zu interpolieren. Da starke Unterschiede des Unterteilungsgrades benachbarter Zellen hierbei immer noch zu Problemen führen, wird außerdem der Quadtree so eingeschränkt, daß benachbarte Zellen maximal um eine Stufe im Baum auseinanderliegen [75]. Dadurch werden aber zusätzliche Patches erzeugt, die natürlich zusätzlichen Aufwand für die Berechnung der Radiosity-Lösung erfordern. Die prinzipiellen Nachteile bilinearer Interpolation bleiben trotzdem erhalten.

Von Salesin et. al. stammt ein Ansatz zur stetigen Radiosity-Rekonstruktion mit Clough-Tocher-Elementen, die aus kubischen Dreiecks-Bézier-Patches konstruiert werden [70]. Ergebnis dieses eher aufwendigen Ansatzes ist eine $C^1$-stetige Rekonstruktion. Unstetigkeiten an Schattengrenzen können durch bewußtes Fallenlassen von Stetigkeitsbedingungen der Clough-Tocher-Konstruktion berücksichtigt werden. Bastos et. al. benutzen bikubische Hermite-Interpolation für reguläre Gitter [4] oder Quadtree-Unterteilungen [3]. Die für die Hermite-Interpolation benötigten Ableitungen werden aus den Radiosity-Werten geschätzt. Unstetigkeiten können durch Einfügen zusätzlicher Kontrollpunkte an den Schattenkanten erfaßt werden. Diese Ansätze haben den Vorteil, Unstetigkeiten korrekt in der Rekonstruktion wiedergeben zu können. Die meisten Radiosity-Programme bieten aber keine Verfahren zur expliziten Konstruktion von Schattenkanten, weil dafür die Implementierung aufwendiger Verfahren und viel Rechenzeit nötig sind.

Kobbelt et. al. sehen, wie es in dem in dieser Arbeit vorgestellten Ansatz auch getan wird, die Radiosity-Rekonstruktion als ein Interpolationsproblem mit unregelmäßig verteilten Stützstellen an [46]. Sie benutzen ein angepaßtes Flächen-Unterteilungsschema zur Generierung der Radiosity-Funktion. Das Verfahren liefert glatte Interpolationen und erhält scharfe Schattenkanten, falls sie durch entsprechend feine Unterteilung im Quadtree repräsentiert sind. An Schattenkanten kann eventuell Überschwingen sichtbar werden, die Autoren erwägen an diesen Stellen die Anwendung eines Tiefpaß-Filters.

## 4.3 Rekonstruktion als Interpolationsaufgabe

Die Rekonstruktion der Radiosity-Funktion ist insofern eine schwierige Aufgabe, als in gewisser Weise widersprüchliche Ziele erreicht werden sollen. Ziel der Rekonstruktion ist eine visuell hochwertige Darstellung des Helligkeitsverlaufs. Da das Auge empfindlich auf Unstetigkeiten im Helligkeitsverlauf und in der ersten Ableitung reagiert, strebt man eine möglichst glatte Rekonstruktion an. Eine Gefahr ist dann aber, daß harte Schattenkanten nicht korrekt wiedergegeben werden. Es kann passieren, daß sie zu stark geglättet werden oder daß die Rekonstruktion an harten Schattenkanten zum Schwingen neigt.

Die Rekonstruktion der Radiosity-Funktion soll hier auf eine neue Weise betrachtet werden, indem sie als Interpolationsproblem formuliert wird.

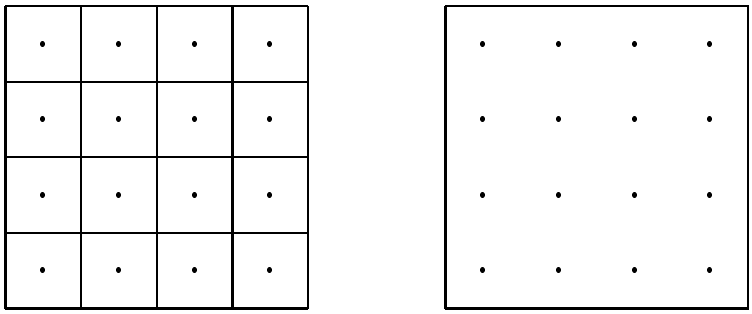


Abbildung 4.2: Reguläres Gitter und Stützstellen

Gegeben sei ein Punkt $\mathbf{q} = (u_q, v_q)$ auf einer Fläche, für den der Wert der Radiosity-Funktion $B(\mathbf{q})$ bestimmt werden soll. Interpolationsverfahren liefern eine Näherung $\tilde{B}(\mathbf{q})$, die aus der vorgegebenen Menge von Stützstellen $\hat{B}(\mathbf{p}_j)$ berechnet wird.

Ausgangspunkt der Interpolation ist die Radiosity-Lösung eines Verfahrens, das mit stückweise konstanten Basisfunktionen und regelmäßigen Gitter-Unterteilungen oder adaptiven, Quadtree-artigen Unterteilungen der Flächen arbeitet. Als Repräsentant eines Elements und Stützstelle $\mathbf{p}_j$ dient jeweils der Element-Mittelpunkt. Für die zwei Arten von Unterteilungen wurden zwei Gruppen von Interpolationsverfahren implementiert, die im folgenden vorgestellt werden.

Ein charakteristisches Problem vieler Interpolationsverfahren ist, daß sie beim Versuch, eine Stufenfunktion zu interpolieren, Überschwingen produzieren. Da sich in einer Vorarbeit [83] gezeigt hat, daß solche zum Überschwingen neigenden Verfahren nicht gut zur Radiosity-Rekonstruktion geeignet sind, wird jedes Verfahren mit der Interpolation einer Stufenfunktion erprobt. Die Stufenfunktion wurde über einem 8·8-Gitter von Stützstellen interpoliert. Die dargestellte Kurve zeigt die Interpolationsfunktion entlang einer Linie mit fixem $v$.

## 4.4 Interpolation auf Gittern

Frühe Versionen des Radiosity-Verfahrens [29] benutzen regelmäßige Unterteilungen der Flächen, bei denen die Stützstellen auf einem Gitter angeordnet sind (Abb. 4.2). Es wurden zwei Interpolationsverfahren für Gitter-Unterteilungen angewendet, nämlich die weit verbreitete *bilineare Interpolation* und als Vertreter höhergradiger Ansätze die *bikubische Spline-Interpolation.*

### 4.4.1 Bilineare Interpolation

Eines der einfachsten und bekanntesten Verfahren ist die bilineare Interpolation. Sie besteht aus der Anwendung von linearer Interpolation in $u$- und $v$-Richtung. Sind $\mathbf{p}_{i_1} = (u_{i_1}, v_{i_1})$, $\mathbf{p}_{i_2} = (u_{i_2}, v_{i_2})$, $\mathbf{p}_{i_3} = (u_{i_3}, v_{i_3})$ und $\mathbf{p}_{i_4} = (u_{i_4}, v_{i_4})$ die dem

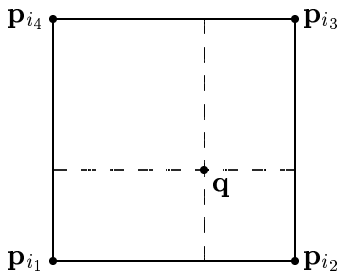



Abbildung 4.3: Bilineare Interpolation

Anfragepunkt $\mathbf{q}$ benachbarten Stützstellen im Gitter (Abb. 4.3), dann wird die Interpolation als

$$\begin{aligned} s &= (u_q - u_{i_1})/(u_{i_2} - u_{i_1}) \\ t &= (v_q - v_{i_1})/(v_{i_4} - v_{i_1}) \\ \tilde{B}(\mathbf{q}) &= (1-s)\,(1-t)\,\hat{B}(\mathbf{p}_{i_1}) \\ &+ s\,(1-t)\,\hat{B}(\mathbf{p}_{i_2}) \\ &+ s\,t\,\hat{B}(\mathbf{p}_{i_3}) \\ &+ (1-s)\,t\,\hat{B}(\mathbf{p}_{i_4}) \end{aligned}$$

berechnet.

Die Anwendung dieses Verfahrens zur Interpolation einer Stufenfunktion ist in Abb.4.4 gezeigt.

### 4.4.2 Bikubische Spline-Interpolation

Bilineare Interpolation weist keine Unstetigkeiten der Helligkeit auf, erzeugt aber eine Interpolation mit Unstetigkeiten in der ersten Ableitung. Solche Unstetigkeiten der ersten Ableitung werden vom Auge in Form von Mach-Bändern erkannt. Eine Möglichkeit, eine auch in der ersten Ableitung stetige

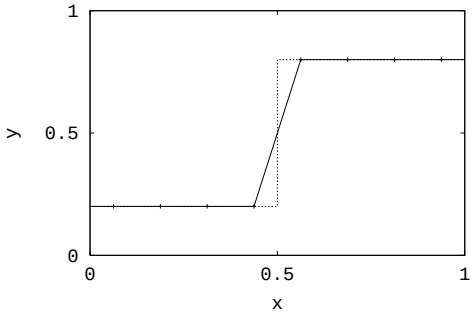



Abbildung 4.4: Stufenfunktion: Bilineare Interpolation

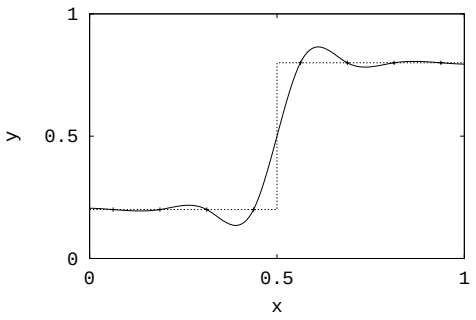



Abbildung 4.5: Stufenfenfunktion: Bikubische Spline-Interpolation

Interpolation zu erhalten, ist es, höhergradige Interpolationsverfahren einzusetzen. Als höhergradiges Verfahren für Gitterdaten wurde die bikubische Spline-Interpolation gewählt.

Die bikubische Spline-Interpolation besteht aus der Hintereinanderschaltung kubischer Spline-Interpolation für Kurven. In jeder Zeile des Gitters der Stützstellen wird eine Spline-Interpolation für den Parameter $u_q$ berechnet. Auf die hieraus erhaltenen Datenpunkte wird dann nochmals eine Spline-Interpolation zum Parameter $v_q$ bestimmt. Es wurden kubische Splines mit natürlichen Endbedingungen benutzt [23], was bedeutet, daß die zweite Ableitung an den Enden der Kurve Null ist. Die Implementierung wurde anhand von [67] erstellt.

Bikubische Spline-Interpolation führt zwar zu einer stetigen Interpolation, kann aber, wie Abb. 4.5 zeigt, zu Überschwingen führen. Dieser Ansatz wurde bereits in [83] zur Radiosity-Rekonstruktion benutzt. Dort hat sich gezeigt, daß die Überschwinger visuell noch störender sind als die durch bilineare Interpolation verursachten Mach-Band-Effekte.

## 4.5 Streudaten-Interpolation

Moderne Radiosity-Verfahren arbeiten meistens mit einer adaptiven, an den Helligkeitsverlauf angepaßten Unterteilung der Flächen. Eingeführt wurden diese adaptiven Verfahren von Cohen et. al. [14]. Dieses Konzept wurde bis zum hierarchischen Radiosity-Verfahren [31] weiterentwickelt. Gemeinsam ist diesem Verfahren eine hierarchische, Quadtree-artige Unterteilung der Flächen (Abb. 4.6).

Die für Gitter-Unterteilungen benutzten Verfahren bestehen im Prinzip aus Kurven-Schemata, die abwechselnd in $u$- und $v$-Richtung angewendet werden. Dies wird durch durch die Regularität der Unterteilung ermöglicht. Für Quadtree-Unterteilungen ist dieser Ansatz nicht anwendbar. Quadtree-Unterteilungen sind zwar nicht vollkommen irregulär, es ist aber nicht einfach, Interpolations-Verfahren anzugeben, die die spezielle Unterteilungsstruktur nutzen können. Läßt man die zugrundeliegende Quadtree-Struktur wegfallen und

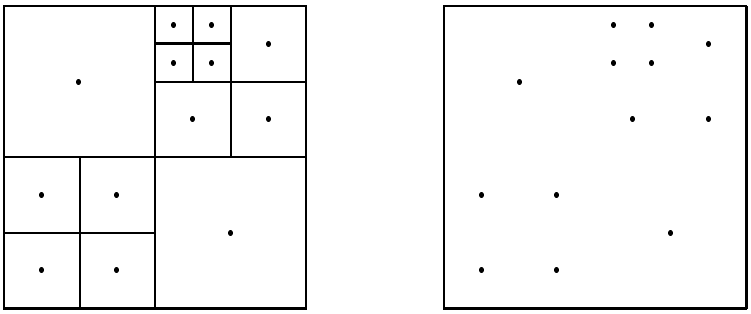


Abbildung 4.6: Quadtree-Unterteilung und Stützstellen

betrachtet nur noch die resultierenden Datenpunkte (wie rechts in Abb. 4.6), bietet sich der Einsatz von Verfahren zur Streudaten-Interpolation an. Kennzeichnend für diese Verfahren ist, daß sie keine Anforderungen an die Lage der Stützstellen stellen.

Verfahren zur Streudateninterpolation lassen sich in zwei Klassen einteilen [69]. Die erste Klasse besteht aus Verfahren, die den Funktionswert an einer Stelle $\mathbf{q}$ durch gewichtete Mittelung umgebender Datenpunkte bestimmen. Aus dieser Klasse wurden die *abstandsgewichtete Interpolation* und die *Natural-Neighbour-Interpolation* angewandt. Die zweite Klasse besteht aus Verfahren, die eine Interpolationsfunktion als lineare Kombination geeigneter Basisfunktionen darstellt. Als Verfahren dieser Klasse wurde die *Hardysche Multiquadriken-Methode* benutzt.

### 4.5.1 Abstandsgewichtete Interpolation

Eines der ältesten Verfahren zur Streudaten-Interpolation ist die abstandsgewichtete Interpolation, auch als Shepard-Verfahren bekannt [72]. Die Interpolation wird als gewichteter Mittelwert der Funktionswerte an den Stützstellen berechnet:

$$\tilde{B}(\mathbf{q}) = \sum_{i=1}^{n} w_i(\mathbf{q})\, \hat{B}(\mathbf{p}_i) \tag{4.4}$$

wobei das Gewicht $w_i(\mathbf{q})$ eines Datenpunktes vom Abstand zwischen $\mathbf{q}$ und der Stützstelle $\mathbf{p}_i$ abhängt. Die Gewichtsfunktionen müssen die Bedingungen

$$\begin{aligned} w_i(\mathbf{p}_i) &= 1 \\ \sum_{i=1}^{n} w_i(\mathbf{q}) &= 1 \\ w_i(\mathbf{q}) &\geq 0 \end{aligned} \tag{4.5}$$

für beliebige Anfragepunkte $\mathbf{q}$ erfüllen. Shepard wählte

$$w_i(\mathbf{q}) = \frac{\sigma_i(\mathbf{q})}{\sum_{i=1}^{n} \sigma_i(\mathbf{q})}$$

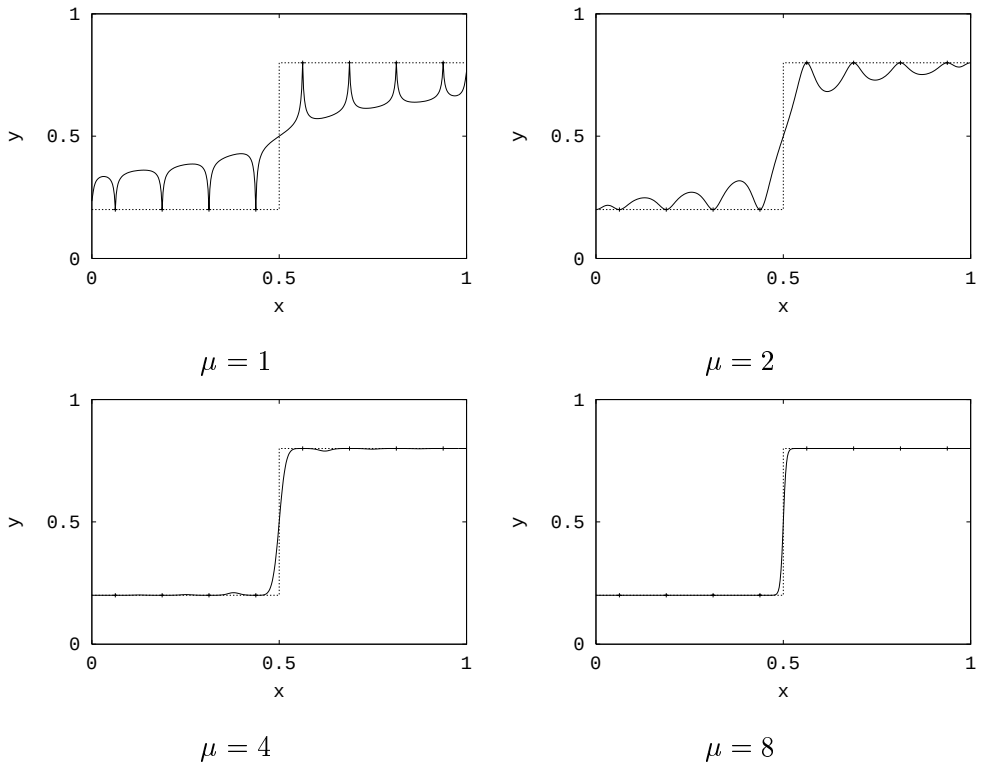



Abbildung 4.7: Stufenfunktion: Abstandsgewichtete Interpolation

$$\sigma_i(\mathbf{q}) \quad = \quad \frac{1}{d(\mathbf{q}, \mathbf{p}_i)^\mu}$$

was zur Interpolation

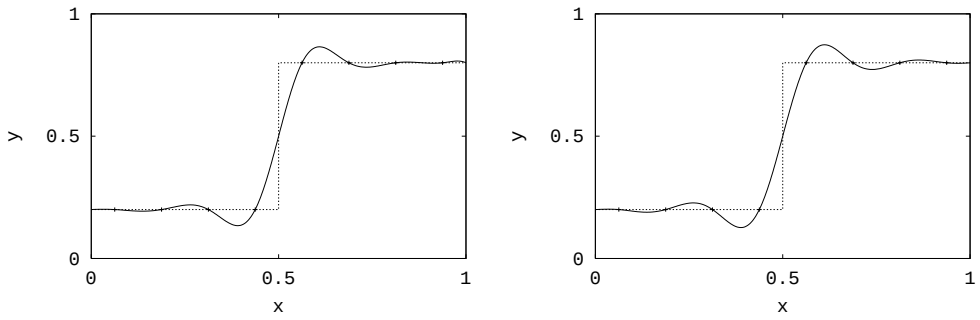
$$\tilde{B}(\mathbf{q}) = \frac{\sum_{i=1}^{n} d(\mathbf{q}, \mathbf{p}_i)^{-\mu} \, \hat{B}(\mathbf{p}_i)}{\sum_{i=1}^{n} d(\mathbf{q}, \mathbf{p}_i)^{-\mu}}$$

führt. Der Exponent $\mu$ bestimmt die Glattheit der Interpolation. Für $\mu \leq 1$ ist die Interpolation an den Stützstellen unstetig und weist dort Spitzen auf. Für $\mu > 1$ ist die Interpolation stetig, an den Stützstellen ist die erste Ableitung immer Null.

Der Einfluß von $\mu$ auf das Ergebnis wird in Abb. 4.7 gezeigt. Hier ist das Verfahren zur Interpolation einer Stufenfunktion benutzt worden. Für die Radiosity-Rekonstruktion wurde der Wert $\mu = 4$ verwendet.

### 4.5.2 Hardysche Multiquadriken-Methode

Die Hardysche Multiquadriken-Methode gehört zur Klasse der Interpolationsverfahren mit *radialen Basisfunktionen*. Charakteristisch für die Basisfunktionen $f_i$ ist, daß ihr Wert nur vom Abstand zur Stützstelle $\mathbf{p}_i$ abhängt, sie sind




$\mu = 0.5$ $\mu = -0.5$

Abbildung 4.8: Stufenfenfunktion: Hardysche Multiquadriken-Methode

also radialsymmetrisch zur Stützstelle. Die Interpolation wird in Basisdarstellung als

$$\tilde{B}(\mathbf{q}) = \sum_{i=1}^{n} \alpha_i \, f_i \left( d \left( \mathbf{q}, \mathbf{p}_i \right) \right) \tag{4.6}$$

angesetzt. Hardy schlug als radiale Basisfunktion

$$f_i(d) = (d^2 + r^2)^{\mu} \qquad r > 0, \mu \neq 0$$

mit $\mu = 0.5$ [32] oder alternativ $\mu = -0.5$ [33] vor.

Die Größe $r$ bestimmt die Rundheit der Interpolation in der Nähe der Stützstellen, sie kann prinzipiell frei gewählt werden. Hier wurde der Vorschlag von Eck [21] benutzt, jedem Stützpunkt ein individuelles

$$r_i = \min_{i \neq j} d(\mathbf{p}_i, \mathbf{p}_j)$$

zuzuordnen. Durch Einsetzen der Interpolationsbedingungen

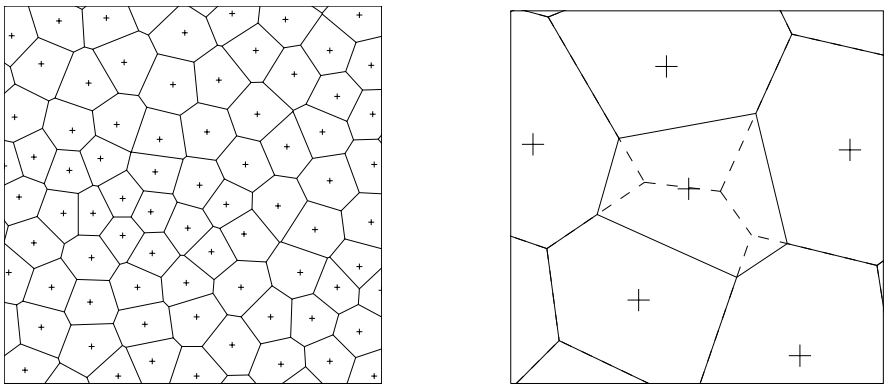
$$\tilde{B}(\mathbf{p}_i) = \hat{B}(\mathbf{p}_i), i = 1, \ldots, n$$

in (4.6) erhält man ein lineares Gleichungssystem, dessen Lösung die benötigten $\alpha_i$ liefert. Zur Lösung des Gleichungssystems wurde ein Gauß-Jordan-Verfahren aus [67] benutzt.

Abb. 4.8 zeigt die Anwendung dieses Verfahrens zur Interpolation einer Stufenfunktion. Wie man sieht, neigt dieses Verfahren hier ähnlich wie die bikubische Spline-Interpolation zum Überschwingen.

### 4.5.3 Natural-Neighbour-Interpolation

Solange die in (4.5) aufgestellten Bedingungen erfüllt sind, lassen sich auch andere Verfahren mit gewichteter Mittelwertbildung konstruieren. Ein Verfahren, das die Gewichte $w_i$ aus der durch die Stützstellen induzierten Voronoi-Zerlegung der Fläche bestimmt, ist die Natural-Neighbour-Interpolation von



Voronoi-Zerlegung | Voronoi-Region für Punkt $\mathbf{q}$

Abbildung 4.9: Voronoi-Zerlegung (nach [69])

Sibson [74]. Das linke Bild in Abb. 4.9 zeigt die Voronoi-Zerlegung einer Fläche in Voronoi-Regionen $\mathcal{VR}(\mathbf{p}_i)$. Um den Radiosity-Wert am Punkt $\mathbf{q}$ zu rekonstruieren, werden die $w_i$ als die relative Überlappung der existierenden Regionen $\mathcal{VR}(\mathbf{p}_i)$ mit der Region $\mathcal{VR}(\mathbf{q})$, die durch Zufügen des neuen Punktes $\mathbf{q}$ in der Voronoi-Zerlegung entstehen würde:

$$
\begin{aligned}
w_i(\mathbf{q}) &= \frac{v\left(\mathcal{VR}(\mathbf{p}_i) \cap \mathcal{VR}(\mathbf{q})\right)}{v\left(\mathcal{VR}(\mathbf{q})\right)} \\
\mathcal{VR}(\mathbf{x}) &= \left\{\mathbf{p} \in \mathbb{R}^2 | d(\mathbf{p},\mathbf{x}) \leq d(\mathbf{p},\mathbf{p}_j), \forall j = 1,\ldots,n\right\}
\end{aligned}
$$

$v$ mißt das Volumen von Punktmengen, da hier eine zweidimensionale Anwendung vorliegt, ist dies also der Flächeninhalt der entsprechenden Gebiete. Die $w_i(\mathbf{q})$ werden auch als *Natural-Neighbour-Koordinaten* bezeichnet. Durch Einsetzen dieser Gewichtsfunktionen in (4.4) erhält man die gesuchte Interpolation $\tilde{B}$.

Normalerweise überlappt $\mathcal{VR}(\mathbf{q})$ nur die Voronoi-Regionen weniger benachbarter Stützstellen $\mathbf{p}_i$, daraus resultiert, daß die Natural-Neighbour-Interpolation

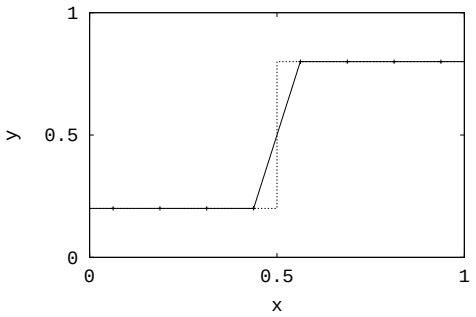



Abbildung 4.10: Stufenfenfunktion: Natural-Neighbour-Interpolation

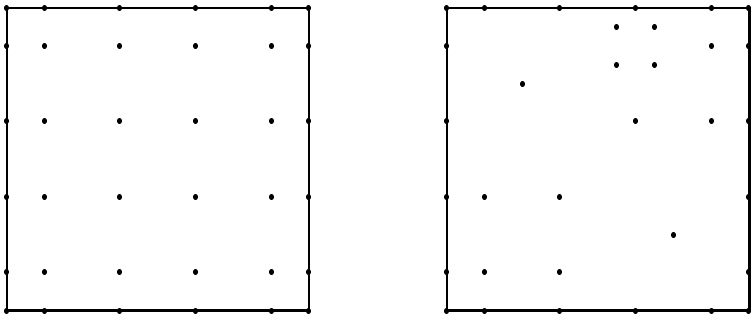



Abbildung 4.11: Einfügen zusätzlicher Rand-Stützstellen

ein lokal interpolierendes Verfahren ist. Als Implementierung dieses Verfahrens wurde das Programm `nngridr` von Dave Watson [86] benutzt.

Abb. 4.10 zeigt, daß sich dieses Verfahren bei der Interpolation der Stufenfunktion wie eine lineare Interpolation verhält.

## 4.6 Einfügen zusätzlicher Rand-Stützstellen

Wählt man die Stützstellen wie in den Abb. 4.2 und 4.6 dargestellt, dann gibt es auf der Fläche einen Rand außerhalb der konvexen Hülle der Stützstellen. In einer ersten Implementierung des Verfahrens hat sich gezeigt, daß insbesondere die Natural-Neighbour-Interpolation hierauf kritisch reagiert [37]. Dieses Problem läßt sich beheben, indem zusätzliche Stützstellen entlang der Ränder der Fläche zugefügt werden. Abb. 4.11 zeigt, wie diese zusätzlichen Stützstellen in der Gitter- bzw. Quadtree-Unterteilung positioniert werden können. Diese Wahl der Stützstellen führt dazu, daß die aufgetretenen Fehler vermieden werden [38]. Als Funktionswert an diesen neuen Stützstellen wird der Funktionswert der Approximation $\hat{B}$ an diesen Stellen eingesetzt.

## 4.7 Ergebnisse

Die oben vorgestellten Interpolationsverfahren sind in verschiedenen Anwendungsfällen getestet worden. Dafür wurde ein Satz von Szenen ausgewählt, der prototypisch bestimmte in der Praxis auftretende Konfigurationen umfaßt. Variiert wurde dabei zwischen langsamem, weichem Schattenverlauf und hartem Schatten sowie Ausrichtung der Schattenkante entlang der Elementunterteilung bzw. diagonal dazu. Für harte Schatten ergeben sich noch Unterschiede daraus, ob die Schattenkante auf Elementgrenzen fällt oder innerhalb der Elemente verläuft. Daraus resultieren die folgenden Szenen:

**Szene 1** weicher Schatten, parallel zu Unterteilungskanten

**Szene 2** weicher Schatten, diagonal zu Unterteilungskanten

**Szene 3a, 3b** harter Schatten, parallel zu Unterteilungskanten

**Szene 4a, 4b** harter Schatten, diagonal zu Unterteilungskanten

Bewertet wurde die Rekonstruktion der Radiosity-Funktion jeweils für die Bodenfläche der Szene. Als Referenz diente eine Radiosity-Lösung mit einer 400·400-Element-Unterteilung der Bodenfläche. Die Radiosity-Rekonstruktion wurde jeweils als Textur mit 400·400 Pixeln berechnet und zur Darstellung auf die Bodenfläche aufgetragen. Die berechneten Texturen lassen sich direkt mit der berechneten Referenzlösung vergleichen. Dies erfolgt einerseits in der Berechnung eines Fehlermasses in der $L^1$, $L^2$ und $L^\infty$-Norm für die Differenz zwischen Rekonstruktion und Referenz. Außerdem werden die Differenzen in Form von Differenzbildern visualisiert, die es erlauben, typische Fehler der Rekonstruktion zu erkennen. In den Differenzbildern repräsentiert mittleres Grau Übereinstimmung zwischen Rekonstruktion und Referenz. In hellen Gebieten ist die berechnete Rekonstruktion heller als die Referenzlösung, in dunklen Gebieten dunkler.

### 4.7.1 Rechenzeiten

Eingabe für die Gitter-basierten Verfahren war die Radiosity-Lösung einer 16·16 Gitter-Unterteilung der Bodenfläche. Die Quadtree-Unterteilung wurde adaptiv erzeugt. Ein Element wurde weiter unterteilt, wenn die probeweise berechneten Radiosity-Werte der 4 Unterelemente mehr als eine bestimmte Schwelle voneinander abwichen. Hieraus resultieren unterschiedliche Elementanzahlen für die verschiedenen Szenen, sie sind aus Tabelle 4.1 ersichtlich. Die Tabelle zeigt auch die für die Texturberechnung nötigen Rechenzeiten. Insbesondere die Verfahren zur Streudaten-Interpolation benötigen deutlich mehr Rechenzeit als die für Radiosity-Programme übliche bilineare Interpolation. In der Praxis sind die hier vorgestellten Verfahren trotzdem einsetzbar, da die Radiosity-Berechnung bis zum Schritt der Rekonstruktion als Vorverarbeitung gelten kann. Wenn die Radiosity-Rekonstruktion, wie in dieser Realisierung auch geschehen, als Textur berechnet wird, kann die Szene auf moderner Hardware interaktiv betrachtet werden, unabhängig davon, welches Verfahren zur Rekonstruktion eingesetzt worden ist.

### 4.7.2 Einfügen zusätzlicher Rand-Stützstellen

Die Natural-Neighbour-Interpolation führt wie oben bereits bemerkt zu sichtbaren Fehlern außerhalb der konvexen Hülle der Datenpunkte. Dies ist an den Differenzbildern in der linken Spalte der Abbildungen F.31 und F.32 erkennbar. Der in Abschnitt 4.6 vorgestellte Ansatz, diese Fehler durch Einfügen zusätzlicher Rand-Stützstellen zu vermeiden, löst dieses Problem, wie die rechte Spalte der Abbildungen zeigt. Die anderen Verfahren haben sich als unempfindlich hinsichtlich dieses Effekts erwiesen.

| | Szene | | | | | |
|---|---|---|---|---|---|---|
| Interpolationsverfahren | 1 | 2 | 3a | 3b | 4a | 4b |
| Referenz | 727.2 | 712.9 | 659.0 | 660.2 | 651.0 | 650.5 |
| **Gitter-Unterteilung** | | | | | | |
| Zahl der Elemente | 256 | 256 | 256 | 256 | 256 | 256 |
| stückweise konstant | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| bilinear | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| bikubisch | 6.5 | 6.5 | 6.5 | 6.5 | 6.4 | 6.4 |
| abstandsgewichtet | 57.7 | 57.6 | 57.6 | 57.7 | 57.7 | 57.8 |
| Hardysche MQ | 61.9 | 61.9 | 61.7 | 61.7 | 61.9 | 61.7 |
| Natural-Neighbour | 18.4 | 18.5 | 18.5 | 19.4 | 18.9 | 18.6 |
| **Quadtree-Unterteilung** | | | | | | |
| Zahl der Elemente | 100 | 154 | 232 | 301 | 631 | 511 |
| stückweise konstant | 0.2 | 0.2 | 0.3 | 0.2 | 0.2 | 0.2 |
| abstandsgewichtet | 27.0 | 37.6 | 53.0 | 66.6 | 133.4 | 109.4 |
| Hardysche MQ | 27.2 | 38.6 | 56.1 | 72.3 | 170.6 | 129.6 |
| Natural-Neighbour | 11.5 | 12.9 | 18.3 | 18.7 | 36.7 | 32.5 |

Tabelle 4.1: Rechenzeiten (in s) für Radiosity-Rekonstruktion (Celeron 400 MHz)

### 4.7.3 Visuelle Qualität

Die Szenen 1 und 2 weisen weiche Schattenkanten auf. Hier führen mit Ausnahme der abstandsgewichteten Interpolation alle Interpolationsverfahren zu visuell akzeptablen Ergebnissen (Abb. F.2, F.3, F.7, F.8). Betrachtet man die Differenzbilder (Abb. F.4, F.5, F.9, F.10), sieht man, daß für diese Szene die Hardysche Multiquadriken-Methoden die geringsten erkennbaren Differenzen aufweist. Insbesondere für Szene 2 sind in den Differenzbildern fast keine Fehler erkennbar.

Die Szenen 3a und 3b weisen harte, parallel zur Unterteilung verlaufende Schattenkanten auf. In diesen Konfigurationen fallen die Verfahren auf, die zum Überschwingen neigen. Dies macht sich in den Bildern (Abb. F.12, F.13, F.17, F.18) und in den Differenzbildern (Abb. F.14, F.15, F.19, F.20) als helle und dunkle Streifen parallel zur Schattenkante bemerkbar. Die Natural-Neighbour-Interpolation liefert insbesondere für die Quadtree-Unterteilungen die besten visuellen Ergebnisse.

Die Szenen 4a und 4b testen das Verhalten der Verfahren für harte, diagonal zur Unterteilung verlaufende Schattenkanten (Abb. F.22, F.23, F.27, F.28). Überschwingen wird hier weniger deutlich, ist aber im Differenzbild zur Gitter-Unterteilung von Szene 4a (Abb. F.24) bei der bikubischen Spline-Interpolation und der Hardyschen Multiquadriken-Methode noch sichtbar. Auch hier zeigt die Betrachtung der Differenzbilder (Abb. F.24, F.25, F.29, F.30) für die Natural-Neighbour-Interpolation die geringsten Abweichungen.

Wie oben bereits bemerkt, verschwindet bei Anwendung abstandsgewichteter Interpolation an den Stützstellen die erste Ableitung. Dies führt jeweils zu einer Plateaubildung an den Stützstellen. In den Bildern führt das zu hellen bzw. dunklen Kreisen um die Stützstellen, die die zugrundeliegende Gitter- oder Quadtree-Unterteilung sichtbar werden lassen. Eine Veränderung des $\mu$-Parameters löst dieses Problem nicht, er führt nur zu Veränderungen des Durchmessers dieser Kreise.

### 4.7.4 Numerische Qualität

Eine Betrachtung der berechneten Fehlerwerte zeigt, daß die visuelle Qualität oftmals gut mit den Fehlerwerten korreliert. Es fällt aber auf, daß sich typische Artefakte bestimmter Verfahren, die zu einer Minderung der visuellen Qualität führen, nicht unbedingt im numerischen Fehler erkennbar werden. So weist die abstandsgewichtete Interpolation in der $L^2$-Norm für fast alle Szenen den größten Fehler auf, hat aber für Szene 3a mit Gitter-Unterteilung plötzlich den kleinsten Fehler (Tabelle F.5). In derselben Szene fällt auf, daß das visuell störende Überschwingen an Schattenkanten nicht im $L^2$-Fehler erkennbar wird.

Der in Abschnitt 4.6 beschriebene Ansatz, Rand-Stützstellen zuzufügen, führt für die Natural-Neighbour-Interpolation zu einer deutlichen Verringerung der numerischen Fehler, bis hin zu einer Halbierung. Dies stimmt mit der Beobachtung überein, daß dieses Verfahren ohne Rand-Stützstellen visuell erkennbare Fehler erzeugt, die durch Einfügen dieser Stützstellen beseitigt werden. Bei den anderen Verfahren resultiert das Zufügen der Rand-Stützstellen in geringen Veränderungen des Fehlers. Für die Hardyschen Multiquadriken führen die Randstützstellen in Szene 1 und 2 zu einer leichten Erhöhung des numerischen Fehlers, in den Szenen 3a, 3b und 4a, 4b führen sie für die Quadtree-Unterteilungen zu kleineren Fehlern.

## 4.8 Zusammenfassung

Der Schritt der Radiosity-Rekonstruktion hat die Aufgabe, die aus einem Radiosity-Verfahren stammende Lösung in eine visuell befriedigende Darstellung umzuwandeln. Die Formulierung dieser Aufgabe als ein Interpolationsproblem erlaubt den Einsatz bekannter Interpolationsverfahren für reguläre Gitter-Unterteilung und für die in der Praxis gebräuchlicheren adaptiven Quadtree-Unterteilungen. Für die unregelmäßige Verteilung der Datenpunkte in Quadtree-Unterteilungen bieten sich Verfahren zur Streudaten-Interpolation an. Aus der Gruppe der implementierten Verfahren hat sich die Natural-Neighbour-Interpolation als bestes Verfahren für alle Arten von Szenen bewährt. Dieses Verfahren ist sowohl in der Lage, den hohen Gradienten im Bereich scharfer Schattenkanten zu erhalten, ohne Überschwingen zu produzieren, als auch Helligkeitsverläufe im Bereich sanfter Schattenkanten ohne visuelle Artefakte darzustellen. Die Radiosity-Rekonstruktion wird in Form von Texturen

erzeugt, was die interaktive Darstellung der Szenen auf moderner Grafikhardware ermöglicht.

# Kapitel 5

# Realisierung

Für die Implementierung der in den Kapiteln 2 bis 4 vorgestellten Verfahren wurde ein objektorientierter Ansatz gewählt. Die entstandene Realisierung besteht aus über 350 Klassen. In diesem Kapitel soll nur das Design derjenigen Komponenten präsentiert werden, die wesentlich für die Realisierung der in den vorigen Kapiteln entwickelten Verfahren sind.

## 5.1 Objektorientierte Software-Entwicklung

### 5.1.1 Objektorientierung

In klassischer Sichtweise besteht ein Programm aus einer Folge von Anweisungen, die Schritt für Schritt von einem Computer ausgeführt werden. Diese Sichtweise legt den Schwerpunkt der Betrachtung auf die ausgeführten Operationen. Technisch gesehen ist diese Sichtweise korrekt, denn letztendlich muß jedes Programm in eine Folge von Operationen der Maschinensprache überführt werden, die der Prozessor des benutzten Rechners zur Verfügung stellt. Die ersten Programme mußten tatsächlich in der Maschinensprache des benutzten Rechners geschrieben werden, weil andere Möglichkeiten noch gar nicht zur Verfügung standen. Man begann aber bald, neue Programmiersprachen und -paradigma zu entwickeln, die immer stärker von der Maschinenebene abstrahieren und näher an die Begriffswelt der Anwendungen heranführen.

Ein objektorientiertes Programm besteht zur Laufzeit aus einer Menge von unterscheidbaren *Objekten*. Diese Menge ist nicht fix, während des Programmlaufs können Objekte erzeugt und vernichtet werden. Ein Objekt ist die Repräsentation eines realen oder virtuellen Konzepts der Anwendung innerhalb des Computers. Jedes Objekt faßt Daten und zugehörige Funktionen in einer Entität zusammen. Dies entspricht einer Realisierung des Konzepts eines *abstrakten Datentyps*. Die Daten werden innerhalb des Objekts in Form von *Attributen* gespeichert. Ein Objekt hat zu jedem Zeitpunkt seiner Existenz eine Belegung der Attribute mit Werten. Jedes Objekt hat eine eindeutige Identität. Haben

zwei Objekte zu einem Zeitpunkt identische Attributwerte, sind sie trotzdem eigenständige, unterscheidbare Objekte. Das Verhalten eines Objekts wird durch eine Schnittstelle, bestehend aus einer Menge von *Operationen*, spezifiziert. Der Zugriff auf ein Objekt erfolgt ausschließlich über die definierte Schnittstelle. Dadurch wird erreicht, daß die Implementierung des Objekts (Attribute und Realisierung der Operationen) vom restlichen Programm *abgekapselt* ist und geändert werden kann, ohne daß andere Programmteile davon betroffen sind. Objekte mit identischer Datenstruktur (Attribute) und identischem Verhalten (Operationen) gehören einer *Klasse* an. Ein Objekt wird auch als *Instanz* seiner Klasse bezeichnet, die Erzeugung eines Objekts wird *Instantiierung* genannt.

Design und Implementierung objektorientierter Software erfolgen auf dem Abstraktionsniveau der Klasse. Für eine umfassende Übersicht sei auf [58, 68, 5, 25] verwiesen. Ein wichtiges Merkmal objektorientierter Software-Entwicklung ist die Möglichkeit, *Beziehungen* zwischen Klassen zu definieren. Eine zwischen Klassen definierte Beziehung wird zur Laufzeit des Programms jeweils durch eine Beziehung zwischen Objekten der entsprechenden Klassen realisiert. Neben der allgemeinen Beziehung, der *Assoziation*, gibt es die spezielleren Formen der *Aggregation*, die eine Teil-Ganzes-Beziehung repräsentiert, und der *Vererbung*. Die Vererbung erlaubt es, Klassen in Hierarchien anzuordnen. In einer Vererbungsbeziehung erbt die Unterklasse von der Oberklasse die Attribute und Operationen. In der Unterklasse können weitere Attribute und Operationen zugefügt oder Operationen der Oberklasse durch neue Implementierungen ersetzt werden. An den Stellen im Programm, an denen ein Objekt der Oberklasse verwendet wird, kann stattdessen ein Objekt einer Unterklasse benutzt werden, wobei es sich sowohl um eine direkte Unterklasse als auch um eine im Vererbungsbaum tiefer liegende, indirekte Unterklasse handeln kann. Aus dieser Eigenschaft und der Möglichkeit, die Implementierung einer Operation in einer Unterklasse zu ersetzen, folgt die Eigenschaft der *Polymorphie* von Objekten: Der Aufruf einer Operation kann, je nachdem zu welcher Klasse ein konkretes Objekt gehört, zur Ausführung einer anderen Implementierung der Operation führen.

### 5.1.2 Vorteile objektorientierter Software-Entwicklung

Nach Meyer [58] erwartet man durch die objektorientierte Softwareentwicklung eine leichtere Erreichbarkeit der folgenden Ziele:

- Korrektheit
- Robustheit
- Benutzerfreundlichkeit
- Wartbarkeit

Die ersten drei Ziele stellen wünschenswerte Eigenschaften jedes Software-Produkts dar.

Unter Wartbarkeit versteht man die Eigenschaft, die Software leicht ändern zu können. Da erfolgreiche kommerzielle Software-Produkte über mehrere Jahre im Einsatz sind und in dieser Zeit sowohl an veränderte Einsatzbedingungen angepaßt als auch um zusätzlich geforderte Funktionalität erweitert werden müssen, kommt diesem Aspekt in der Praxis große Bedeutung zu. Die für Wartung nötigen Aufwendungen übersteigen oftmals die Kosten zur Realisierung der ersten Programmversion.

Bei herkömmlichen Ansätzen der Software-Entwicklung, wie z. B. der *Strukturierten Analyse* [18, 2], basiert die Architektur des Systems auf der Funktionalität der Anwendung. Spätere Änderungen der Funktionalität lassen sich in diesen Ansätzen aber nur schwer realisieren. Theoretisch wäre jedesmal eine Anpassung der Architektur notwendig, was in der Praxis nicht realisierbar ist, da der hierfür nötige Aufwand nahe an den einer Neu-Implementierung herankommen kann. Deshalb versucht man in der Praxis, neue Funktionalität innerhalb der bestehenden Architektur zuzufügen. Nach einer Reihe von Änderungen wird die Diskrepanz zwischen Architektur und Funktionalität dann aber so groß, daß weitere Änderungen im Rahmen der Architektur nicht mehr realisiert werden können, die Software muß dann tatsächlich durch eine Neu-Implementierung ersetzt werden.

Bei objektorientierter Software-Entwicklung basiert die Architektur des Systems nicht auf der Funktionalität sondern auf den Objekten der Anwendung und ihren Beziehungen [58, 68]. Die resultierende Architektur ist langfristig gesehen stabiler. So kann man z. B. davon ausgehen, daß eine Anwendung aus dem Gebiet der Computergrafik unabhängig von der konkreten Funktionalität mit Objekten wie Punkten, Vektoren und geometrischen Körpern umgehen wird. Änderungen der Funktionalität lassen sich dann im Rahmen dieser objektorientierten Architektur durch Änderung vorhandener oder Bereitstellung neuer Operationen realisieren. Auf objektorientierten Architekturen basierende Anwendungen sind also leichter änderbar und lassen sich langfristig besser warten als auf funktionalen Architekturen basierende Software. An dieser Stelle sei angemerkt, daß objektorientierte Software-Entwicklung zwar langfristig gesehen kostengünstiger ist, daß aber die Realisierung der ersten Version eventuell aufwendiger ist. Die für die erste Version geleistete Arbeit zahlt sich später in besserer Wartbarkeit aus.

Bei Software, die im Rahmen wissenschaftlicher Forschung entwickelt wird, handelt es sich meistens um Prototypen, die die Realisierbarkeit neuer Konzepte verifizieren aber bei weitem nicht das Qualitätsniveau kommerziell genutzter Software erreichen sollen. Von den oben genannten Qualitätszielen der Software-Entwicklung sind Robustheit und Benutzerfreundlichkeit oftmals von geringerem Interesse, da diese Software fast ausschließlich von Fachleuten, meistens vom Autor der Software selbst, benutzt wird. Auch Wartbarkeit im Sinne von langfristiger Anpassung an sich ändernde Anforderungen ist aufgrund des Prototyp-Charakters dieser Software meistens nicht erforderlich. In diesem Anwendungsgebiet kann aber die Wartbarkeit für kurzfristige Änderungen ein ausschlaggebendes Argument für eine objektorientierte Vorgehensweise sein. Die Erstellung wissenschaftlicher Software hat oftmals die Entwicklung neuer

Algorithmen zum Ziel, so daß die Funktionalität der Anwendung zu Beginn noch gar nicht klar definiert ist und erst im Laufe der Realisierung, auch auf der Grundlage von mit der Software gemachten Erfahrungen, entwickelt wird. Auch in diesem Fall liefert eine objektorientierte Vorgehensweise eine flexible Architektur, die gute Voraussetzungen bietet, als Experimentierumgebung zur Entwicklung neuer Algorithmen zu dienen [56, 76, 26].

## 5.2 Designziele

Bei dem im Rahmen dieser Arbeit entwickelten Software-System war, wie im vorigen Abschnitt beschrieben, das wichtigste Ziel, eine flexible Testumgebung zu realisieren. Diese Umgebung soll die Entwicklung und den Test neuer Verfahren im Gebiet der fotorealistischen Bilderzeugung ermöglichen. Neben diesem Hauptziel sollte auch ein Vergleich der Effizienz konkurrierender Verfahren erreicht werden. Liegen solche Verfahren in unterschiedlichen Implementierungen vor, sind Effizienz-Aussagen eventuell irreführend, da die unterschiedliche Qualität der Implementierungen die Ergebnisse verfälschen kann. Zutreffendere Aussagen erhält man, wenn die Implementierungen soweit wie möglich dieselbe Code-Basis benutzen und sich tatsächlich nur in den Teilen unterscheiden, in denen unterschiedliche Algorithmen zum Einsatz kommen.

Beide Ziele lassen sich in einem objektorientierten Design erreichen, das es erlaubt, an klar definierten Stellen des Systems unterschiedliche Realisierungen von Algorithmen einsetzen zu können, ohne daß der Rest des Systems dadurch beeinflußt wird. Gamma et. al. formulieren dieses Grundprinzip als [25]:

> The focus here is on *encapsulating the concept that varies*, a theme of many design patterns.

Erreichen läßt sich diese Austauschbarkeit von Algorithmen bzw. Algorithmus-Implementierungen dadurch, daß eine Schnittstelle spezifiziert wird, die allgemein genug ist, um die verschiedenen Varianten als Realisierungen dieser Schnittstelle zu verwirklichen.

Es ist möglich, innerhalb einer Klasse eine Operation ohne zugehörige Implementierung zu spezifizieren. Eine solche Operation wird als *abstrakt* bezeichnet (C++ *pure virtual method*). Eine Klasse, die mindestens eine abstrakte Operation enthält, wird dadurch zu einer *abstrakten* Klasse. Von einer abstrakten Klasse können keine Instanzen erzeugt werden. Nur von Unterklassen, die tatsächlich Implementierungen zu den abstrakten Operationen der Oberklasse zur Verfügung stellen, können Instanzen generiert werden. Abstrakte Klassen können dazu benutzt werden, Schnittstellen zu definieren, während die konkreten Unterklassen Realisierungen des durch die Schnittstelle spezifizierten Konzepts bereitstellen. Wenn gewährleistet wird, daß andere Teile des Programms auf Objekte dieser Klassen nur über die durch die abstrakte Oberklasse definierte Schnittstelle zugreifen, lassen sich tatsächlich verschiedene Implementierungen dadurch austauschen, daß jeweils Objekte unterschiedlicher Unterklassen

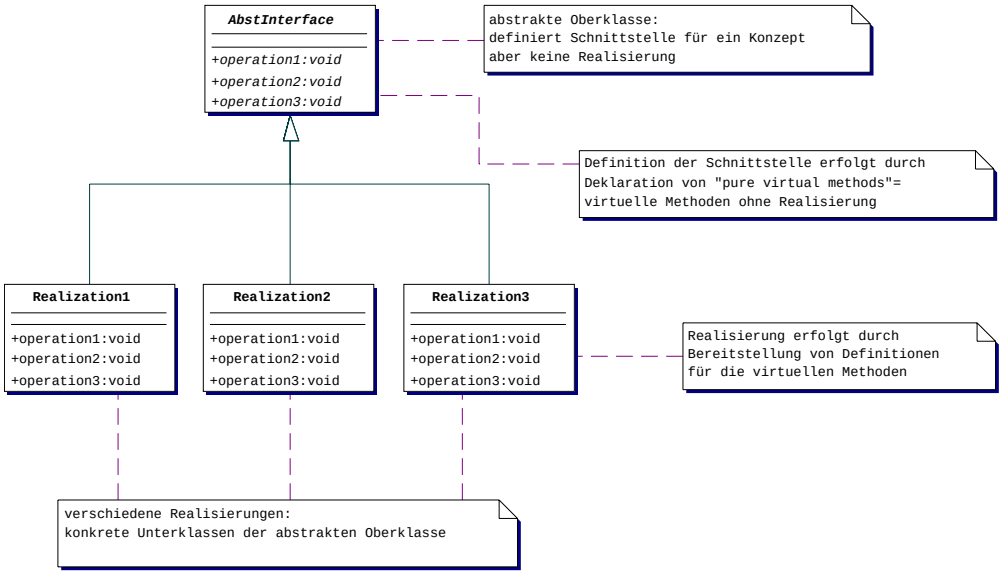



Abbildung 5.1: Grundprinzip für Austauschbarkeit von Realisierungen

instantiiert und benutzt werden. Dieses in Abb. 5.1 dargestellte Prinzip ist das in dem hier präsentierten Software-System am häufigsten verwendete Grundmuster.

## 5.3 Realisierung

Die Implementierung der in den Kapiteln 2 bis 4 vorgestellten Verfahren wurde als objektorientierte Anwendung verwirklicht. Der Entwurf erfolgte in OMT-Notation [68] (Werkzeug: Stp/OMT), später in UML-Notation [5] (Werkzeug: Together C++). Die Programmierung erfolgte in der Programmiersprache C++ [81, 80]. Als nützliche Informationsquellen zu einem umfassenden Verständnis der Sprache sei auf [59, 60, 36] verwiesen. Mit der speziellen Problematik der Realisierung großer Softwareprojekte in C++ beschäftigt sich [53].

## 5.4 Architektur

Im Systementwurf wurde zur Realisierung eine geschichtete Architektur gewählt (Abb. 5.2). Die unteren Schichten bilden drei Klassenbibliotheken. Die unterste Schicht, die `BaseLib`-Bibliothek, besteht aus einer Sammlung allgemein einsetzbarer Basisklassen. Die darüberliegende Schicht ist in der `WaveFunc`-Bibliothek realisiert, die Wavelet-Darstellungen von Funktionen und insbesondere die in Kapitel 2 vorgestellten Verfahren zur Monte Carlo-Integration mit adaptiven Dichtefunktionen enthält. Die nächste Schicht bildet die `RadaRT`-Bibliothek mit Implementierungen der in den Kapiteln 3 und 4 vorgestellten Ansätze zur fotorealistischen Bilderzeugung. Eine auf Grundlage dieser Bibliotheken realisierte

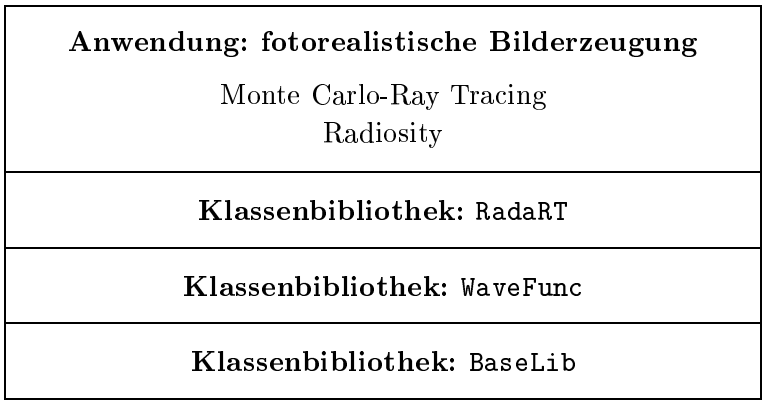



Abbildung 5.2: Systementwurf: geschichtete Architektur

Anwendung stellt die oberste Schicht dar. Höhere Schichten können auf darunterliegende Schichten zugreifen, aber nicht umgekehrt. Durch dieses Prinzip werden zyklische Abhängigkeiten zwischen den Schichten vermieden.

## 5.5 `BaseLib`-Klassenbibliothek

Die unterste Schicht besteht aus der Klassenbibliothek `BaseLib` mit allgemein nutzbaren Basisklassen. Sie enthält verschiedene Realisierungen von Container-Klassen, zu Diagnosezwecken einsetzbare Debug-Klassen und eine Exception-Hierarchie, die es erlaubt, möglicherweise auftretende Fehlersituationen zu signalisieren. Außerdem werden die Repräsentation von Punkten und Vektoren im dreidimensionalen Raum sowie geometrische Transformationen unterstützt.

### 5.5.1 Container

Die Container-Klassen liegen in Form von Template-Realisierungen vor. Dadurch ist es möglich, Objekte unterschiedlicher Klassen typsicher in Containern zu speichern.

#### 5.5.1.1 Arrays

Die erste Gruppe der Container-Klassen besteht aus ein- und zweidimensionalen Arrays. Dabei wird unterschieden zwischen Arrays, deren Dimension

- zur Übersetzungszeit bekannt ist: `StaticArray`, `StaticArray2D`,
- bei der Erzeugung festgelegt wird: `Array`, `Array2D` oder
- während der Laufzeit verändert werden kann: `DynamicArray`, `DynamicArray2D`.

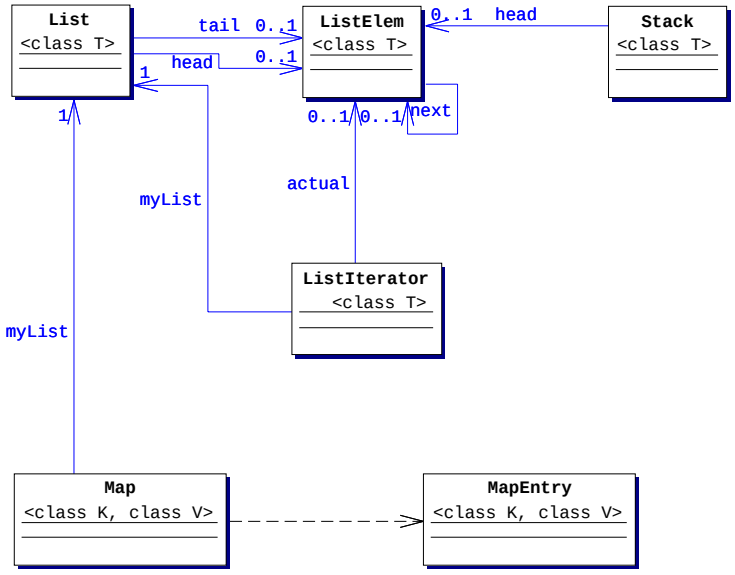



Abbildung 5.3: Klassendiagramm für listenbasierte Container

#### 5.5.1.2 Listenbasierte Container

Die zweite Gruppe bilden listenbasierte Container-Klassen (Abb. 5.3). Basis für diese Container ist die Template-Klasse `List`, die eine Implementierung einfach verketteter Listen zur Verfügung stellt. Die Elemente der Liste werden in `ListElem`-Objekten gespeichert. Das Durchlaufen einer Liste wird durch die Iterator-Klasse `ListIterator` ermöglicht.

Das Konzept des Stapelspeichers wird in der Template-Klasse `Stack` in Form einer Liste realisiert, an der jeweils vorne Elemente zugefügt bzw. entfernt werden.

Eine Implementierung eines assoziativen Arrays liegt mit der Template-Klasse `Map` vor. Je ein Schlüssel-Wert-Paar wird in einem `MapEntry`-Objekt gespeichert. Diese Assoziationspaare werden in einer Liste gehalten. Schlüssel und Wert können Objekte beliebiger, in den Template-Argumenten vorgegebener Klassen sein.

### 5.5.2 Punkte, Vektoren, 3D-Transformationen

Die nächste Gruppe von Klassen stellt die in der Computergrafik allgemein einsetzbaren Konzepte von Punkten, Vektoren und Transformationen im dreidimensionalen Raum zur Verfügung. Punkte werden in der Klasse `Point`, Vektoren in der Klasse `Vector` realisiert. Beide Klassen stellen Unterklassen eines Vektors der Dimension vier dar, in dem die homogenen Raumkoordinaten des Punktes bzw. Vektors gespeichert werden. Die vierte Koordinate eines Punktes ist 1, die eines Vektor 0. Die Operationen der Klassen stellen allgemein sinnvolle

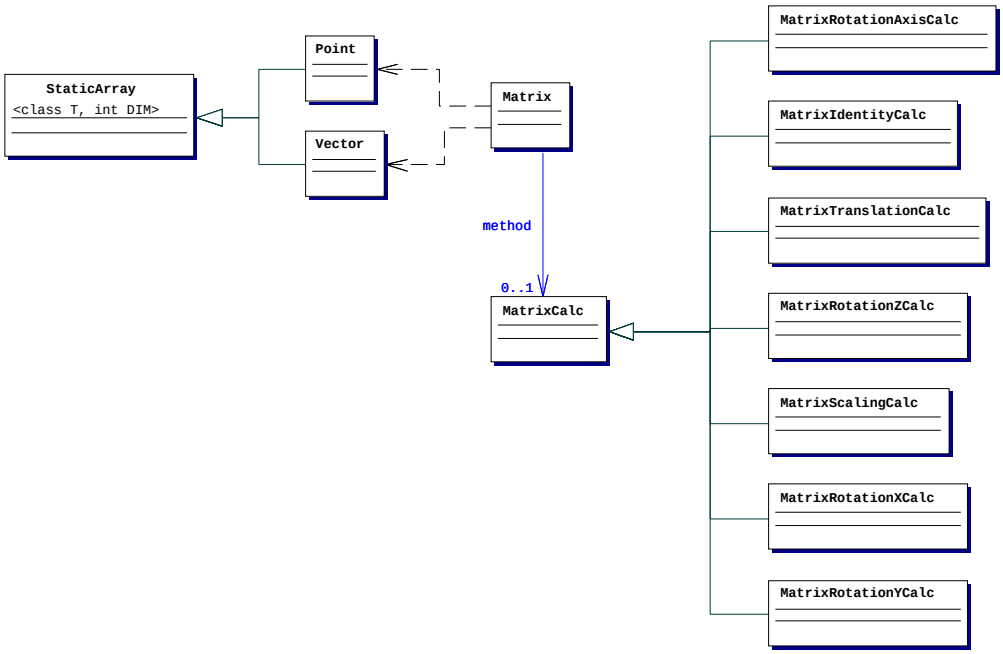



Abbildung 5.4: Klassendiagramm für Punkte, Vektoren, 3D-Transformationen

Funktionalität wie Längenbestimmung eines Vektors, Addition eines Vektors zu einem Punkt oder Subtraktion von Vektoren zur Verfügung.

Transformationen werden durch die Klasse `Matrix` repräsentiert. Die Operationen umfassen Transformationen von Punkten und Vektoren sowie Matrix-spezifische Operationen wie Multiplikation, Transposition oder Invertierung. Hat die durch die Matrix repräsentierte Transformation eine spezielle Form, wie z. B. eine Skalierung oder eine Rotation um eine bestimmte Achse, dann enthält die Matrix einen Verweis auf eine `MatrixCalc`-Unterklasse, die eine für die spezielle Transformation optimierte Implementierung bietet. Gibt es zu der Transformation keine spezielle Implementierung, erfolgt die Berechnung in Form einer Matrix-Multiplikation innerhalb der `Matrix`-Klasse.

## 5.6 `WaveFunc`-Klassenbibliothek

Die `WaveFunc`-Bibliothek ermöglicht die Verwendung von Funktionen in Basis- und Wavelet-Darstellung. Sie enthält die Funktionalität, beliebige, nicht-negative Funktionen über ein- oder zweidimensionalem Definitionsbereich als Wahrscheinlichkeitsdichten zu normieren und effizient Samples entsprechend dieser Dichten zu generieren. Insbesondere sind im Rahmen dieser Bibliothek auch die im Kapitel 2 vorgestellten Verfahren zur Monte Carlo-Integration mit adaptiven Wahrscheinlichkeitsdichten in Basis- und Wavelet-Darstellung implementiert worden.

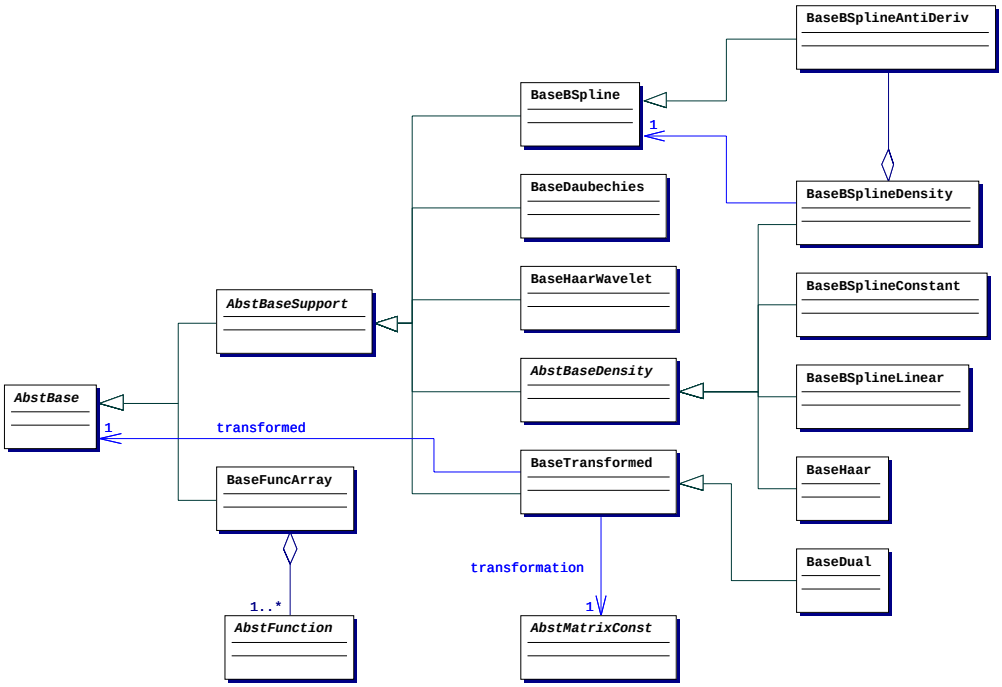



Abbildung 5.5: Klassendiagramm für Funktionsbasen

### 5.6.1 Funktionsbasen

Funktionen in Basis-Darstellung sind eines der meistbenutzten Konzepte in dieser Bibliothek. Deshalb sollen hier erst die verfügbaren Funktionsbasen vorgestellt werden (Abb. 5.5). Oberklasse aller Realisierungen von Funktionsbasen ist die abstrakte Klasse `AbstBase`. Sie definiert die Schnittstelle von Funktionsbasen, die hauptsächlich aus Abfrage der Dimension der Basis, Berechnung des Funktionswerts der Basisfunktionen zu einem gegebenen Argument $x$ und Abfrage des Trägers einer Basisfunktion besteht. Der Funktionswert kann für eine einzelne Basisfunktion oder für alle Basisfunktionen gleichzeitig berechnet werden, das Ergebnis ist in letzterem Fall ein Vektor. Die Möglichkeit der gleichzeitigen Berechnung aller Basisfunktionen ist oftmals effizienter, als die Berechnung einzelner Werte. Im Rahmen von Wavelet-Konstruktionen sind Funktionsbasen mit lokalem Träger der Basisfunktionen vorteilhaft. Das Ergebnis der Berechnung ist dann ein dünn besetzter Vektor. Im Falle einer quadratischen B-Spline-Basis sind z. B. nie mehr als drei Einträge des Vektors von 0 verschieden. Deshalb wird auch eine Möglichkeit zur Abfrage des Ergebnisses in einer Repräsentation als dünn besetzter Vektor geboten. Diese Eigenschaft kann zur Implementierung optimierter Algorithmen genutzt werden, deren Laufzeit unabhängig von der Dimension der Basis ist. Außerdem ist eine Operation zur Berechnung des inneren Produkts einer Funktion mit allen Basisfunktionen vorhanden. Diese Operation kann z. B. zur Projektion einer Funktion in eine Basis benutzt werden. Die Berechnung erfolgt durch numerische Integration unter Nutzung der Information über die Träger der Basisfunktionen.

Die allgemeinste Implementierung stellt in der Klasse `BaseFuncArray` eine Funktionsbasis als Array von Funktionen dar. Da kein Wissen über die einzelnen Basisfunktionen verfügbar ist, lassen sich keine optimierten Implementierungen der Operationen verwirklichen.

Alle weiteren Funktionsbasen sind Unterklasse der abstrakten Klasse `AbstBaseSupport`, die zur Klasse `AbstBase` ein Array mit `Support`-Objekten zur Speicherung der Träger der Basisfunktionen zufügt.

B-Spline-Basen sind in der Klasse `BaseBSpline` implementiert. Sie werden durch Angabe des Grades und eines Knotenvektors definiert.

Die Klasse `BaseDaubechies` bietet einen Basis für orthonormale Wavelet-Konstruktionen.

Die Wavelet-Basen einer Haar-MRA sind mit der Klasse `BaseHaarWavelet` verfügbar, während die Skalierungs-Basen einer solchen MRA in der Klasse `BaseHaar` vorliegen.

Die Klasse `BaseHaar` gehört zu einer besonderen Gruppe von Funktionsbasen, die Basisfunktionen enthalten, die als Wahrscheinlichkeitsdichten nutzbar sind. Alle diese Klassen sind Unterklasse der abstrakten Klasse `AbstBaseDensity`. Neben den Haar-Basen handelt es sich hierbei um B-Spline-Basen. Neben der allgemeinen Realisierung in der Klasse `BaseBSplineDensity`, gibt es für B-Splines der Grade 0 und 1 spezielle Realisierungen (`BaseBSplineConstant`, `BaseBSplineLinear`). Diese speziellen Klassen erlauben die Benutzung optimierter Implementierungen, die deutlich schneller sind als die allgemeine Realisierung. Um eine effiziente Sample-Generierung zu erreichen, enthalten `BaseBSplineDensity`-Objekte Stammfunktionen der Wahrscheinlichkeitsdichten. Die zur Darstellung dieser Stammfunktionen benötigte B-Spline-Basis ist als Verweis auf ein `BaseBSplineAntiDeriv`-Objekt enthalten.

In der Klasse `BaseTransformed` lassen sich aus einer Transformation erhaltene Basen darstellen. Ein `BaseTransformed`-Objekt enthält einen Verweis auf die transformierte Basis sowie auf die Transformationsmatrix in Form eines `AbstMatrixConst`-Objekts. Ein Spezialfall einer transformierten Basis ist die duale Basis, die als Unterklasse `BaseDual` vorliegt. Sie erhält zur Initialisierung einen Verweis auf die primäre Basis, die Transformationsmatrix wird dann berechnet.

### 5.6.2 Multi-Skalen-Analysen

Wichtiges Hilfsmittel bei der Repräsentation von Funktionen in Wavelet-Darstellung sind Möglichkeiten zur Darstellung dünn besetzter Matrizen und Vektoren, weil nur dann eine effiziente Realisierung der Analyse- und Synthese-Operationen ermöglicht wird.

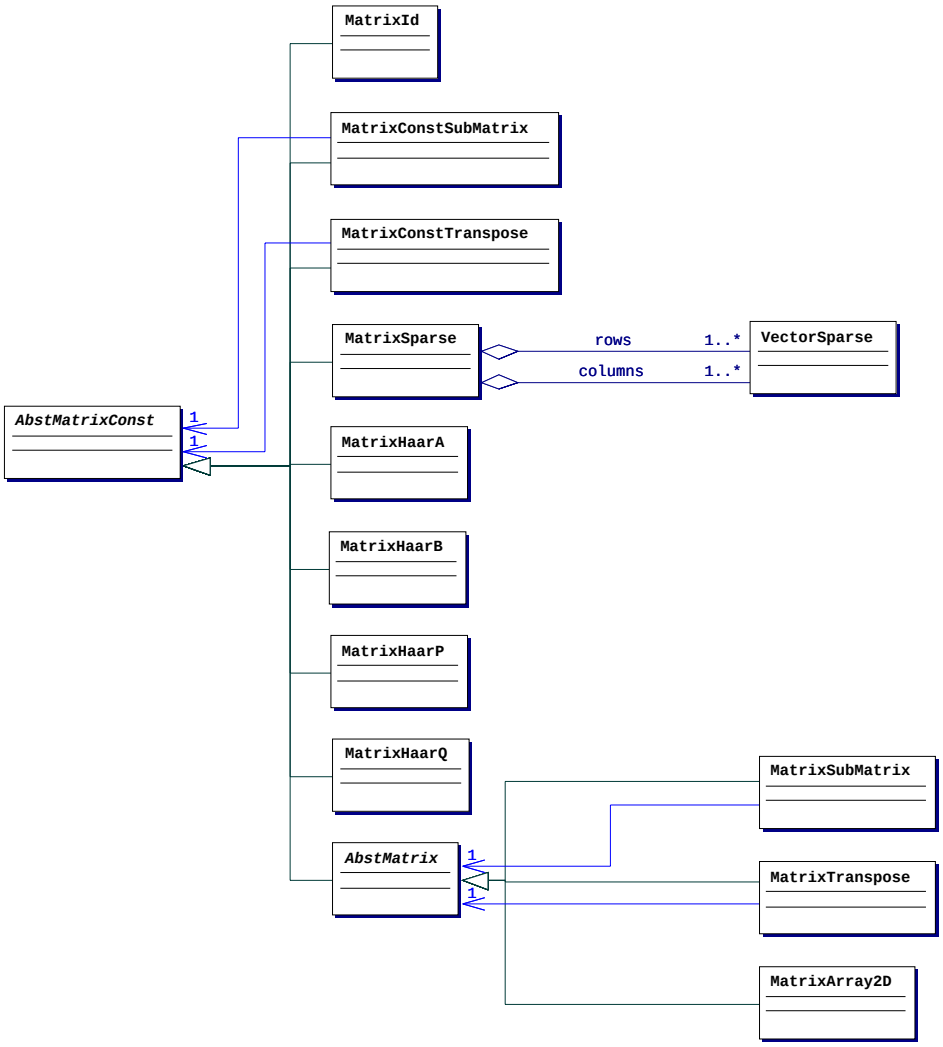



Abbildung 5.6: Klassendiagramm für Matrizen

### 5.6.3 Matrizen

Die abstrakte Klasse `AbstMatrixConst` ist Oberklasse für alle Matrix-Klassen (Abb. 5.6). Sie definiert als Schnittstelle Operationen zur Abfrage der Matrix-Dimensionen und lesenden Zugriff auf die Matrix-Elemente sowie allgemeine Realisierungen der Multiplikation mit einem Vektor ($v \cdot M$ bzw. $M \cdot v$). Außerdem enthält sie optimierte Implementierungen für die Multiplikation mit dünn besetzten Vektoren.

Dünn besetzte Matrizen werden in der Klasse `MatrixSparse` dargestellt. Sie enthält dünn besetzte Vektoren (`VectorSparse`) als Spalten und Zeilen.

Die Klassen `MatrixHaarA`, `MatrixHaarB`, `MatrixHaarP` und `MatrixHaarQ` enthalten spezialisierte Implementierungen der Filter-Matrizen einer Haar-MSA.

Die Klasse `MatrixId` enthält die Identitätsabbildung. `MatrixConstSubMatrix`

und `MatrixConstTranspose` erlauben den Zugriff auf eine Untermatrix bzw. eine transponierte Matrix.

Die Unterklassen der abstrakten Klasse `AbstMatrix` erlauben zusätzlich zu den oben genannten Operationen auch schreibenden Zugriff auf Matrix-Elemente. Die Klasse `MatrixArray2D` speichert eine Matrix als zweidimensionales Array. Die Klassen `MatrixSubMatrix` und `MatrixTranspose` realisieren Zugriff auf eine Untermatrix bzw. eine transponierte Matrix.

### 5.6.4 Vektoren

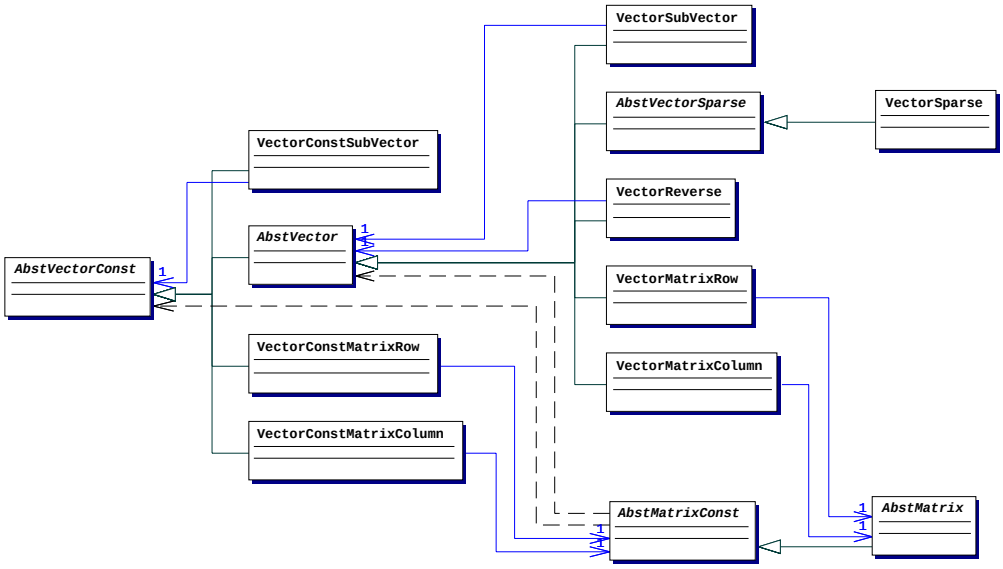



Abbildung 5.7: Klassendiagramm für Vektoren

Oberklasse aller Vektor-Darstellungen ist `AbstVectorConst`. Operationen sind Abfrage der Dimension und lesender Zugriff auf die Vektor-Elemente. Direkte Unterklassen sind `VectorConstSubVector`, die Zugriff auf einen Teilvektor realisiert, sowie `VectorConstMatrixRow` und `VectorConstMatrixColumn`, die Zeilen bzw. Spalten einer Matrix als Vektoren darstellen.

Unterklassen von `AbstVector` bieten zusätzlich schreibenden Zugriff auf Vektor-Elemente. Auch hier gibt es Klassen zum Zugriff auf einen Teilvektor (`VectorSubVector`), auf Matrix-Zeilen (`VectorMatrixRow`) bzw. -Spalten (`VectorMatrixColumn`).

Die Klasse `VectorSparse` stellt zur Zeit die einzige Implementierung des Konzepts dünn besetzter Vektoren dar. Nur eine Teilfolge des Vektors enthält von Null verschiedene Einträge. Es ist möglich, Beginn und Ende dieser Teilfolge abzufragen sowie zu ändern. Außerdem enthält sie eine optimierte Operation zur Berechnung des Skalarprodukts mit einem zweiten Vektor.

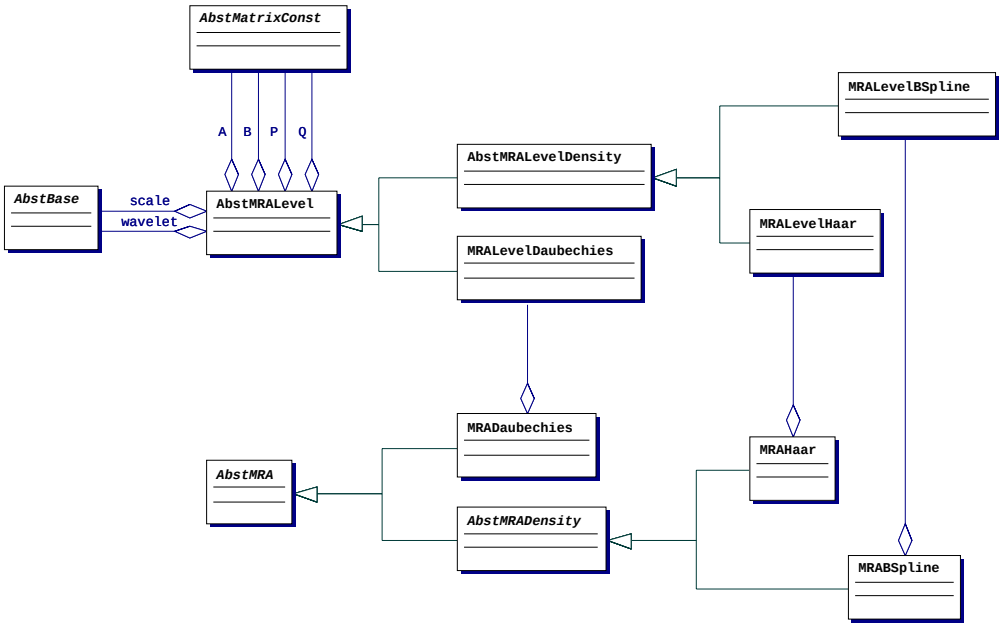



Abbildung 5.8: Klassendiagramm für Multi-Skalen-Analysen

### 5.6.5 Multi-Skalen-Analysen

Die abstrakte Klasse `AbstMRA` ist Oberklasse für alle verfügbaren Multi-Skalen-Analysen. Sie enthält Operationen zum Zugriff auf die Skalen sowie zur Analyse und Synthese von MSA-Darstellungen. Als Realisierung einer höhergradigen orthonormalen MSA gibt es `MRADaubechies`.

Multi-Skalen-Analysen, deren Skalierungs-Basen auch als Wahrscheinlichkeitsdichten nutzbar sind, haben eine wichtige Bedeutung für den in Kapitel 2 beschriebenen adaptiven Ansatz. Sie sind als Unterklassen der abstrakten Klasse `AbstMRADensity` vorhanden. Es handelt sich um eine Realisierung der Haar-MSA (`MRAHaar`) und B-Spline-MSA (`MRABSpline`).

Jede MSA enthält die zu ihr gehörenden Skalen. Oberklasse der Skalen ist die abstrakte Klasse `AbstMRALevel`. Sie enthält Verweise auf die Skalierungs- und Wavelet-Basis dieser Skala und auf die Analyse- und Synthese-Filter. Zu jeder MSA-Klasse gibt es eine entsprechende Skalen-Klasse, ein `MRAHaar`-Objekt enthält z. B. `MRALevelHaar`-Objekte als Skalen. Die Skala einer Haar-MSA (`MRALevelHaar`) enthält natürlich `BaseHaar`- und `BaseHaarWavelet`-Basen und spezialisierte Filter `MatrixHaarA`, `MatrixHaarB`, `MatrixHaarP`, `MatrixHaarQ`. Der Benutzer einer Haar-MSA braucht den konkreten Typ dieser Basen und Filter nicht zu kennen, da er ausschließlich über die in `AbstMRALevel` definierte Schnittstelle auf die Skala zugreift. Streng genommen braucht er sogar den Typ der MSA nicht zu kennen, da er sie über die in `AbstMRA` definierte Schnittstelle benutzt. Dieses ist ein konkretes Beispiel für die Anwendung des in Abschnitt 5.2 vorgestellten Grundprinzips.

### 5.6.6 Funktionen

Abstrakte Oberklasse aller Funktionen ist die Klasse `AbstFunction` (Abb. 5.9). Jede Funktion enthält den Träger in Form eines `Support`-Objekts. Der Träger ist als Folge von Intervallen dargestellt. Außerdem enthält eine Funktion eine Operation zur Berechnung des Funktionswertes für ein Argument $x$ und zur Berechnung des inneren Produkts mit einer zweiten Funktion. Die Berechnung des inneren Produkts erfolgt numerisch unter Ausnutzung der Träger-Information.

Funktionen in Basis-Darstellung liegen in der Klasse `FuncBase` vor. Objekte dieser Klasse enthalten einen Verweis auf eine Funktionsbasis (`AbstBase`) sowie einen Vektor, der die Koeffizienten der Funktion in der gegebenen Basis speichert. Handelt es sich bei der Basis speziell um eine `AbstBaseDensity`-Unterklasse, kann die Funktion spezialisiert als `FuncBaseDensity`-Objekt dargestellt werden.

Die Klasse `FuncMRA` implementiert Funktionen in einer MSA-Darstellung. Ein `FuncMRA`-Objekt enthält einen Verweis auf die zugehörige Multi-Skalen-Analyse (`AbstMRA`) sowie Vektoren zur Speicherung der Skalierungs- und Wavelet-Koeffizienten in den verschiedenen Skalen. Ist die MSA speziell eine `AbstMRADensity`-Klasse, kann die Funktion spezialisiert als `FuncMRADensity` dargestellt werden.

Die weiteren Funktions-Repräsentationen werden innerhalb der Bibliothek nicht benutzt, sind aber nützlich zur Erstellung von Anwendungen, die die `WaveFunc`-Bibliothek verwenden. Es handelt sich um:

**FuncBaseFunction:** Selektion einer Basisfunktion aus einer Basis

**FuncSampled:** stückweise lineare Interpolation einer abgetasteten Funktion, die in Form von $(x, y)$-Wertepaaren vorliegt

**FuncDeriv:** (numerisch bestimmte) Ableitung einer Funktion

**FuncPolynom:** $x^n$

**FuncOpMul:** $f(x) \cdot g(x)$

**FuncOpDiv:** $f(x)/g(x)$

**FuncOpSub:** $f(x) - g(x)$

**FuncOpAdd:** $f(x) + g(x)$

**FuncOpReparam:** $f(g(x))$ (Reparametrisierung)

**FuncConstant:** konstante Funktion

**FuncCFunction:** in der Programmiersprache C implementierte Funktion (erlaubt z. B. Nutzung der Funktionen aus der Standard-Bibliothek `math.h`)

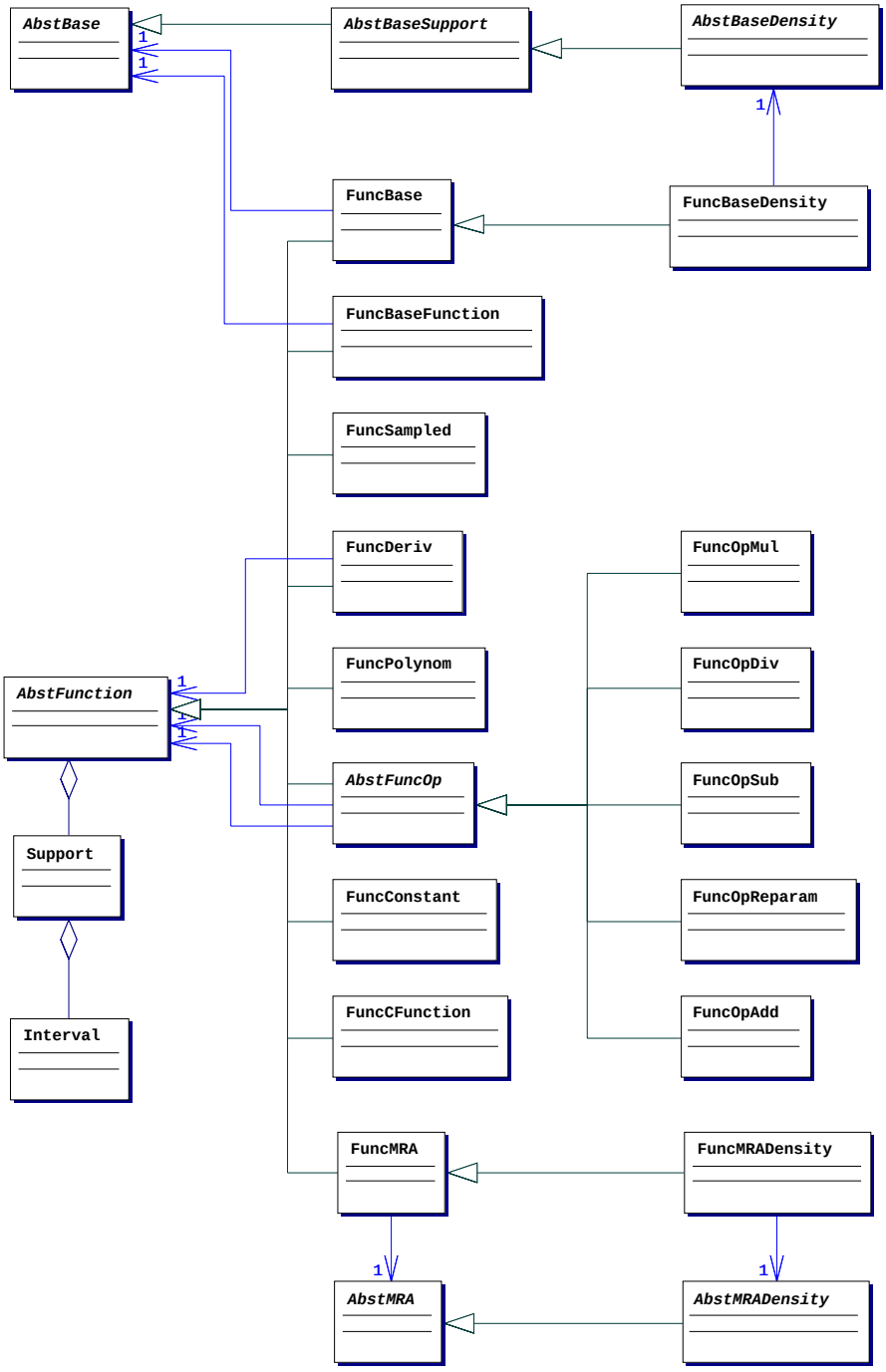



Abbildung 5.9: Klassendiagramm für Funktionen

### 5.6.7 Adaptive 2D-Dichtefunktionen

Da für die im Kapitel 3 vorgestellte Anwendung Dichtefunktionen über einem zweidimensionalen Definitionsbereich benutzt werden, soll hier deren Realisierung vorgestellt werden.

Abstrakte Oberklasse für Funktionen über einem zweidimensionalen Definitionsbereich ist `AbstFunc2D`. Sie definiert als Schnittstelle die Berechnung des Funktionswertes. Das Argument $(u, v)$ wird in Form eines Objekts der Klasse `Arg2D` übergeben.

Wahrscheinlichkeitsdichten sind selbst Funktionen, also Unterklassen von `AbstFunc2D`. Ihre gemeinsame Oberklasse ist `AbstDensity2D`. Sie enthalten einen Verweis auf die zugrundeliegende Funktion, aus der sie durch Normierung hervorgehen. Ihre Schnittstelle enthält Operationen zur Monte Carlo-Integration sowie zur Abfrage einzelner Samples mit zugehörigem Gewicht.

Oberklasse für Verfahren mit adaptiven Dichtefunktionen ist die Klasse `AbstDensity2DAdaptive`. Die Unterklasse `Density2DAdaptive` realisiert das in Abschnitt 2.6 vorgestellte Verfahren zur Integration mit adaptiven Dichtefunktionen in Tensorprodukt-Darstellung. Die adaptiv erzeugte Approximation wird in dem assoziierten `Func2DTensorDensity`-Objekt gespeichert. Wichtige Eigenschaft für die Anwendbarkeit des Verfahrens ist, daß ein `Func2DTensorDensity`-Objekt zwei `AbstBaseDensity`-Objekte als Funktionsbasen benutzt, während die allgemeine `Func2DTensor`-Darstellung auf `AbstBase`-Objekte als Basen verweist. Die Unterklasse `Density2DAdaptiveDeterm` enthält zusätzlich Operationen, die Samples deterministisch so auf Basisfunktionen verteilen, daß auf jede Basisfunktion eine vorgegebene Mindestzahl von Samples entfällt.

In den Unterklassen der abstrakten Klasse `AbstDensity2DMRA` befindet sich die Realisierung des in Abschnitt 2.6.4 vorgestellten Verfahrens zur Integration mit adaptiven hierarchischen Dichtefunktionen. Die Adaption innerhalb einer bestimmten Skala erfolgt unter Benutzung eines `Density2DAdaptive`-Objekts, deshalb hat `AbstDensity2DMRA` Verweise auf diese Klasse und auf die in einem `Func2DTensorDensity`-Objekt gespeicherte Approximation. Die Speicherung der Approximation in der hierarchischen MSA-Darstellung erfolgt in einem `Func2DMRADensity`-Objekt. Auch hier ist natürlich Voraussetzung, daß die Skalierungs-Basen der eingesetzten MSAs als Dichtefunktionen nutzbar sind, deshalb werden MSAs vom Typ `AbstMRADensity` benutzt. Die Klasse `Func2DMRADensity` ist Unterklasse von `Func2DMRANonStandard`, die Speicherung erfolgt also in der Non-Standard-Konstruktion. Auch für die Standard-Konstruktion gibt es eine Realisierung in Form der Klasse `Func2DStandard` (im Klassendiagramm nicht dargestellt). Oberklasse aller Funktionen in MSA-Darstellung ist `AbstFunc2DMRA`.

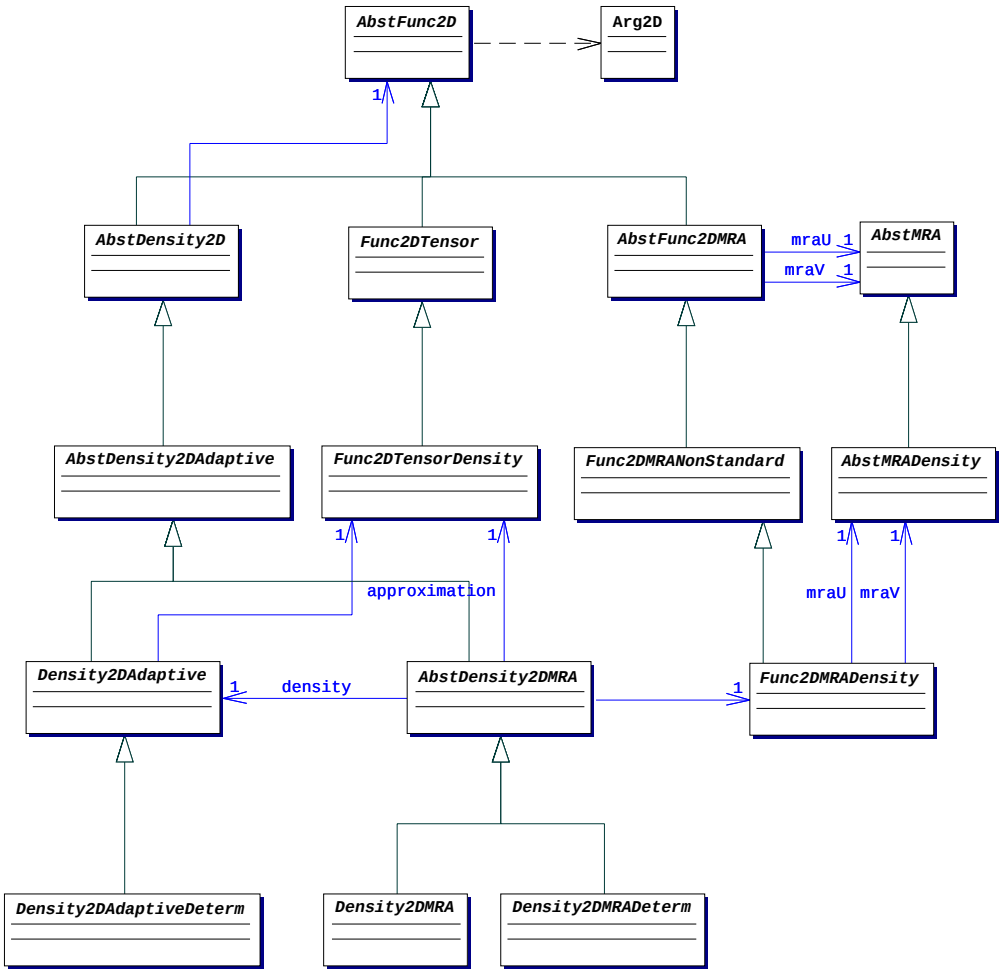



Abbildung 5.10: Klassendiagramm für adaptive 2D-Dichtefunktionen

## 5.7 RadaRT-Klassenbibliothek

Die in den Kapiteln 3 und 4 vorgestellten Verfahren zur Bilderzeugung sind im Rahmen der RadaRT-Bibliothek entwickelt worden. Diese Bibliothek hat den Namen daher, daß sie Radiosity- und Ray Tracing-Techniken kombiniert (**Rad**iosity **a**nd **R**ay **T**racing).

### 5.7.1 Hierarchie der Szenen-Objekte

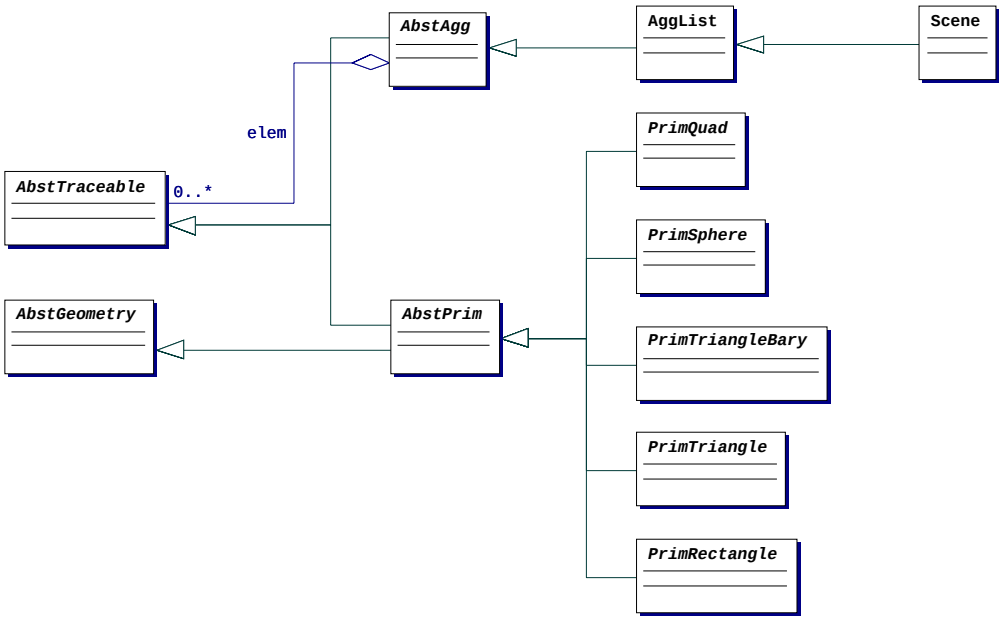



Abbildung 5.11: Klassendiagramm für Hierarchie der Szenen-Objekte

Ziel fotorealistischer Bilderzeugung ist die Generierung von Bildern virtueller Szenen. Eine Szene wird in Form einer Menge von geometrischen Körpern, im folgenden Szenen-Objekte genannt, dargestellt, die gewisse optische Eigenschaften (Lichtemission, Reflexionsverhalten) haben und an bestimmten Positionen im Raum plaziert sind.

Abstrakte Oberklasse für diese Szenen-Objekte ist `AbstTraceable`. Wie der Name dieser Klasse bereits signalisiert, definiert sie die Schnittstelle der grundlegenden Ray Tracing-Funktionalität, es handelt sich um die in Abschnitt 3.1 vorgestellte Ray Tracing-Funktion.

Die Repräsentation der Szene kann hierarchisch organisiert werden, indem aggregierte Objekte wiederum weitere `AbstTraceable`-Objekte enthalten. Oberklasse für aggregierte Objekte ist die abstrakte Klasse `AbstAgg`. Die bekannten Ray Tracing-Beschleunigungstechniken [27] lassen sich als solche Aggregationen implementieren. Zur Zeit existiert nur die listenbasierte Aggregation `AggList`. Eine Szene wird mit der Klasse `Scene` als Unterklasse von `AggList` realisiert.

Sie fügt zu `AggList` weitere Funktionalität zur Radiosity-Berechnung und Bilderzeugung mit Ray Tracing hinzu.

Szenen-Objekte, die tatsächlich geometrische Körper darstellen, sind Unterklassen der abstrakten Klasse `AbstPrim`. Die für die Geometrie relevanten Konzepte, wie Berechnung eines Punktes, Normalenvektors, partieller Ableitungen oder Flächeninhalt eines Teilgebiets der Fläche, werden in der Oberklasse `AbstGeometry` definiert. Zusätzlich enthält `AbstPrim` Möglichkeiten zur Speicherung optischer Eigenschaften (mehr dazu in Abschnitt 5.7.5).

Implementierungen geometrischer Körper liegen in Form der Klassen `PrimTriangle` (Dreieck), `PrimTriangleBary` (Dreieck mit Parametrisierung in baryzentrischen Koordinaten), `PrimRectangle` (Rechteck), `PrimQuad` (Viereck) und `PrimSphere` (Kugel) vor. Jede dieser Geometrien stellt eine parametrisierte Fläche dar. Jede Klasse muß mindestens eine Implementierung zur Berechnung eines Punktes der Fläche zur Verfügung stellen. Für die anderen Operationen gibt es in `AbstGeometry` Implementierungen, die in den Unterklassen durch für die jeweilige Geometrie optimierte Implementierungen ersetzt werden können. Um die in `AbstTraceable` definierte Schnittstelle zu erfüllen, muß jede Klasse außerdem die Implementierung der Ray Tracing-Funktion bieten, d. h. eine Methode, die zu einem Strahl den Punkt bestimmt, an dem der Strahl das Objekt trifft.

### 5.7.2 Strahlverfolgung

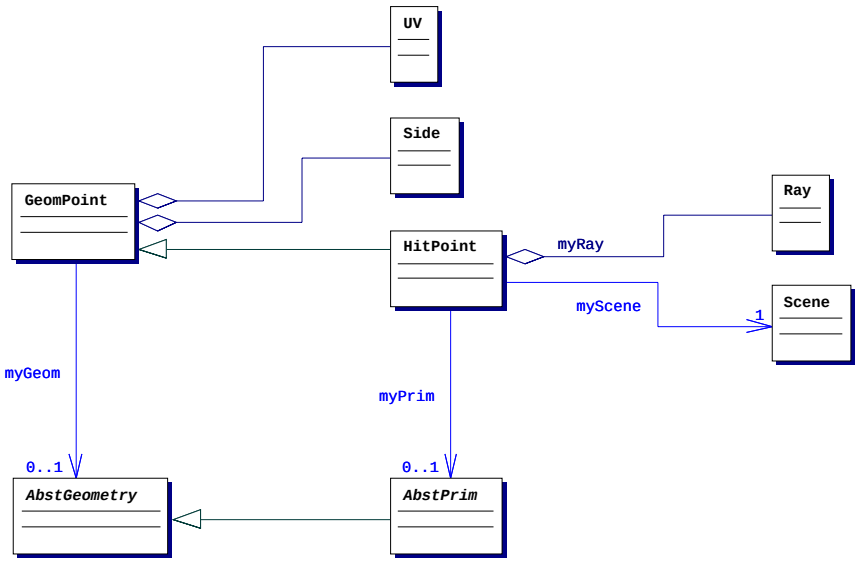



Abbildung 5.12: Klassendiagramm für Strahlverfolgung

Kern der Strahlverfolgung ist die Klasse `HitPoint` (Abb. 5.12). Sie enthält den zu verfolgenden Strahl als Objekt der Klasse `Ray` und einen Verweis auf die Szene, in der der Strahl verfolgt werden soll. Durch Aufruf der Ray Tracing-

Funktionen der in der Szene enthaltenen Objekte werden Schnittpunkte des Strahls mit den Objekten bestimmt. Jeweils der zum Strahlanfang naheliegendste Punkt bleibt im `HitPoint`-Objekt gespeichert. Es handelt sich hierbei um die Geometrie des Punktes selbst, die dadurch enthalten ist, daß `HitPoint` eine Unterklasse von `GeomPoint` ist, und einen Verweis auf das vom Strahl getroffene `AbstPrim`-Objekt, das benötigt wird, um Zugriff auf die optischen Eigenschaften des getroffenen Punktes zu erhalten.

Ein `GeomPoint` repräsentiert einen Punkt auf einer Fläche. Die Fläche, auf der der Punkt liegt, wird durch einen Verweis auf ein `AbstGeometry`-Objekt bezeichnet. Die lokalen, der Parametrisierung der Fläche entsprechenden Koordinaten sind in einem `UV`-Paar gespeichert. Das enthaltene `Side`-Objekt legt fest, ob der Punkt auf der Vorder- oder Rückseite der Fläche liegt. Vorteil dieser Darstellung eines Punktes ist, daß die geometrischen Eigenschaften in der lokalen Umgebung des Punktes, wie Normalenvektor, Richtungsableitungen oder Flächeninhalt einer Delta-Umgebung, abrufbar sind. Dies sind Informationen, die zur Implementierung von Ray Tracing- oder Radiosity-Beleuchtungsverfahren erforderlich sind. Das Konzept des `GeomPoint` ermöglicht leichten Zugriff auf diese Daten über eine wohldefinierte Schnittstelle. Eine Kenntnis des tatsächlich getroffenen Körpers ist für die Benutzung nicht notwendig.

### 5.7.3 Bildgenerierung

Die Bildgenerierung erfolgt innerhalb des `RadaRT`-Systems grundsätzlich durch Strahlverfolgung, wobei aber sehr unterschiedliche Ansätze alternativ genutzt werden können. Oberklasse aller realisierten Verfahren ist die abstrakte Klasse `AbstPixSampler` (Abb. 5.13). Diese Klasse kapselt das Konzept der Berechnung eines einzelnen Pixels. Ein Pixel wird als Teilfläche der virtuellen Bildebene angesehen. Ein dem `AbstPixSampler` zugeordnetes `AbstUVSampler`-Objekt enthält das Verfahren zur Verteilung der Primärstrahlen über die Pixelfläche. Ein Primärstrahl wird hier in lokalen Koordinaten eines Punktes der Bildebene repräsentiert. Die von der Abbildungstransformation abhängige Umrechnung in einen Strahl erfolgt in einem dem `AbstPixSampler` zugeordneten `AbstView`-Objekt. Zur Zeit sind Parallel-Projektion sowie Zentral-Projektion in Form der Klassen `ViewParallel` sowie `ViewPerspective` realisiert. Das Konzept ist aber allgemein genug, um beliebige weitere Perspektiven, wie z. B. Fischaugen-Perspektive, zu verwirklichen. Auch hier gilt natürlich wieder, daß neue Perspektiven ohne Änderungen an anderen Teilen des Systems zugefügt werden können.

Die einfachsten, nicht auf stochastischem Ray Tracing basierenden Verfahren sind in den Klassen `PixSamplerClassic` und `PixSamplerDoF` verfügbar. Sie dienen hauptsächlich zur Darstellung einer mit dem Radiosity-Verfahren berechneten Lösung. Ist einer Fläche eine Radiosity-Darstellung zugewiesen (näheres hierzu in Abschnitt 5.7.6), wird deren Wert am Trefferpunkt als Helligkeit verwendet. Ist die Fläche ausschließlich oder zusätzlich als perfekt spiegelnd

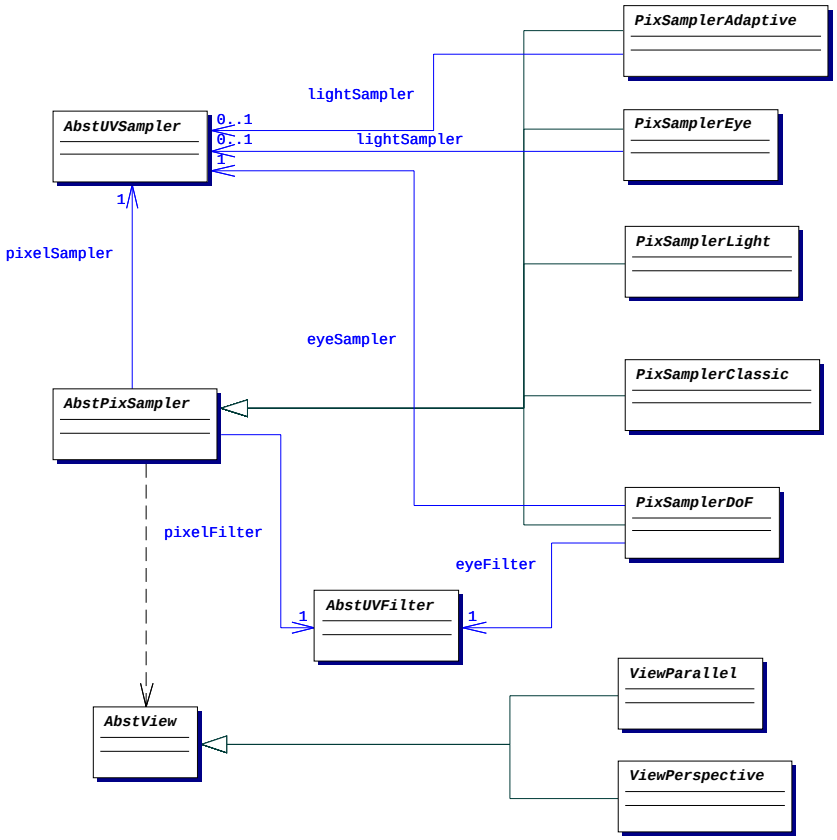



Abbildung 5.13: Klassendiagramm für Bildgenerierung

markiert, wird außerdem ein Strahl rekursiv in die Spiegelungsrichtung verfolgt. Die Klasse `PixSamplerDoF` unterscheidet sich von `PixSamplerClassic` dadurch, daß sie zusätzlich Tiefenschärfe realisiert, indem unter Benutzung eines weiteren `AbstUVSampler`-Objekts die Startpunkte der Strahlen über eine virtuelle Pupille verteilt werden.

Auf Pfadverfolgung basierende stochastische Ray Tracing-Verfahren sind in den Klassen `PixSamplerEye` und `PixSamplerLight` verfügbar. Die Klasse `PixSamplerEye` realisiert eine beim Auge des virtuellen Betrachters startende Pfadverfolgung, während `PixSamplerLight` ein umgekehrt arbeitendes, an den Lichtquellen beginnendes Verfahren enthält. Das in Kapitel 3 präsentierte Verfahren zur Pfadverfolgung unter Nutzung adaptiver hierarchischer Dichtefunktionen liegt in der Klasse `PixSamplerAdaptive` vor. Objekten der Klassen `PixSamplerEye` und `PixSamplerLight` kann optional ein `AbstUVSampler`-Objekt zugewiesen werden. Ist diese Referenz definiert, dann wird der direkte Anteil der Beleuchtung wie in Abschnitt 3.1.3 beschrieben durch Abtastung der Lichtquelle berechnet. Das `AbstUVSampler`-Objekt wird dabei dazu benutzt, die Abtastpunkte auf der Lichtquelle auszuwählen.

Die zur Verteilung der Primärstrahlen über die Pixelfläche bzw. zur Bestimmung von Abtastpunkten auf einer flächigen Lichtquelle benutzten Verfahren,

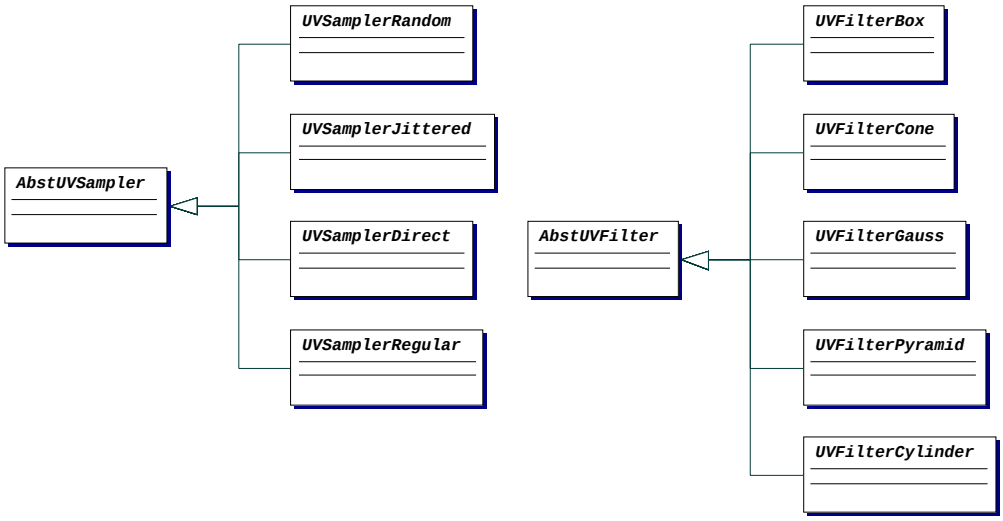



Abbildung 5.14: Klassendiagramm für Primärstrahl-Generierung/Filterung

sind als Unterklassen von `AbstUVSampler` realisiert (Abb. 5.14). Es handelt sich um:

**UVSamplerDirect:** jeder Strahl geht exakt durch die Pixelmitte

**UVSamplerRegular:** die rechteckige Pixelfläche wird in ein regelmäßiges Gitter unterteilt, durch den Mittelpunkt jeder Gitterzelle wird ein Strahl geschossen

**UVSamplerJittered:** die rechteckige Pixelfläche wird in ein regelmäßiges Gitter unterteilt, ein Strahl wird durch jeweils einen zufällig gewählten Punkt einer Gitterzelle geschossen

**UVSamplerRandom:** Strahlen werden zufällig über eine rechteckige Pixelfläche verteilt

Auch hier ist wieder klar die Realisierung des Grundprinzips erkennbar, daß verschiedene Strategien in Form austauschbarer Klassen realisiert sind, die ohne Änderung des restlichen Systems ausgetauscht werden können. Eine sinnvolle, durch Zufügen einer neuen Klasse erreichbare Erweiterung, wäre z. B. die Implementierung einer Klasse, die die Pixelfläche nicht als Rechteck sondern als Kreis betrachtet.

Die zur Filterung über die Pixelfläche nutzbaren Verfahren befinden sich in Unterklassen der Klasse `AbstUVFilter`. Zu einer gegeben lokalen `UV`-Koordinate berechnen sie ein Gewicht. Die Farbe des Pixels wird dann als gewichteter Mittelwert über alle Strahlen bestimmt. Vorhanden sind als Filter mit Gleichgewichtung über die Pixelfläche Würfel (`UVFilterBox`) oder Zylinder (`UVFilterCylinder`), als Filter mit linearem Abfall von der Pixelmitte Pyramide (`UVFilterPyramid`) und Kegel (`UVFilterCone`) sowie ein Gaußscher

Filter (`UVFilterGauss`). Wie man sieht, kann man die in der `AbstUVSampler`-Hierarchie fehlenden Generierungsmethoden für kreisförmige Pixelflächen auch durch Einsatz von Filtern mit kreisförmigem Träger ersetzen.

### 5.7.4 Bildspeicherung

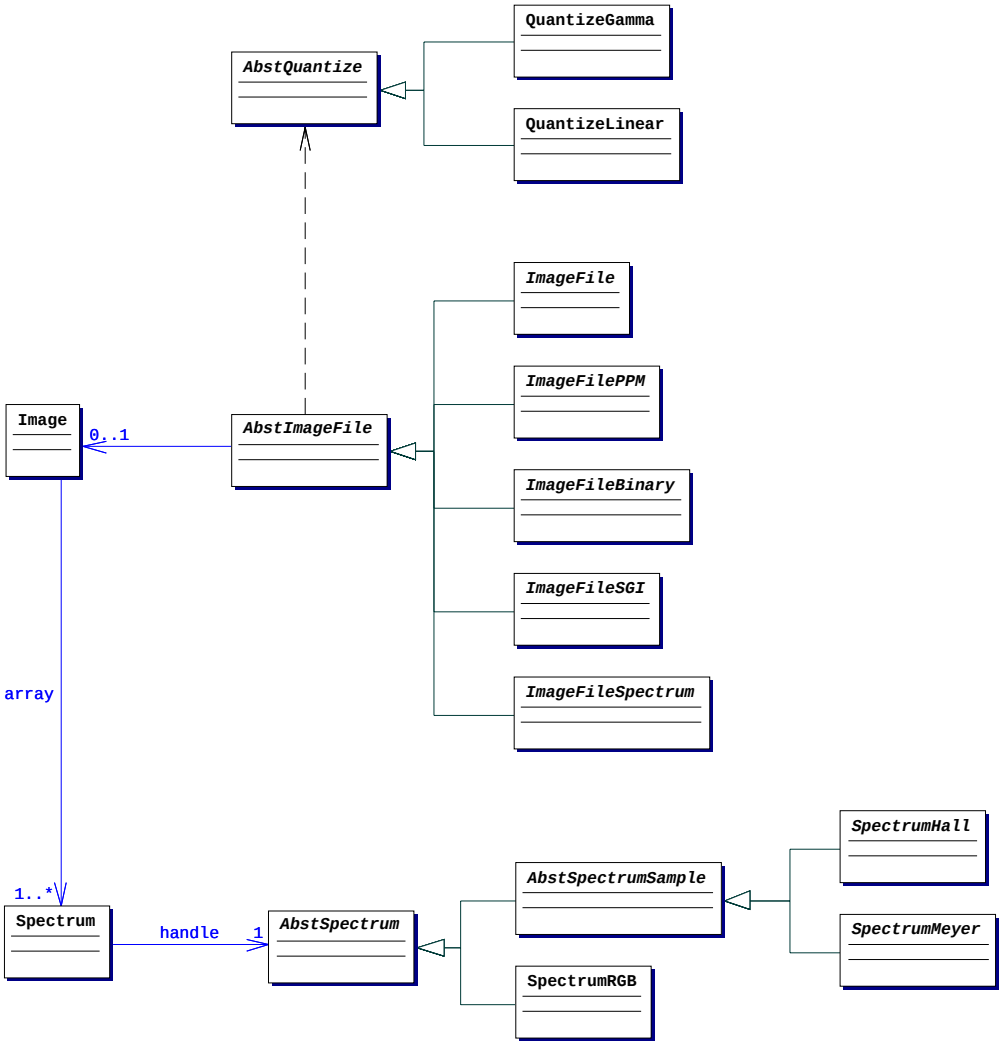



Abbildung 5.15: Klassendiagramm für Bildspeicherung

Zur Speicherung von Beleuchtungsquantitäten innerhalb der `RadaRT`-Bibliothek dient die Klasse `Spectrum` (Abb. 5.15). Es handelt sich dabei z. B. um Radiance-Werte eines Strahls oder Pixels oder um Radiosity-Werte einer Fläche. Es liegen verschiedene Implementierungen zur Spektrendarstellung vor. Diese sind Unterklassen von `AbstSpectrum`. Ein `Spectrum`-Objekt kapselt eine Referenz auf ein `AbstSpectrum`-Objekt und hat als wichtigste Funktionalität die Erzeugung und Vernichtung von `AbstSpectrum`-Objekten zu realisieren, während das Spektrum betreffende Operationen an das `AbstSpectrum`-Objekt delegiert werden.

Dadurch werden Benutzer dieser Klasse von der Aufgabe befreit, Spektrum-Instanzen passender Typen jeweils zu erzeugen und zu vernichten. Stroustrup bezeichnet eine Klasse dieser Funktionalität als *handle class* [81].

Ein Bild ist in der Klasse `Image` als zweidimensionales Array von Pixeln (`Spectrum`-Objekte) gespeichert. Verschiedene Verfahren zum Lesen und Schreiben unterschiedlicher Bildformate sind in Form von `AbstImageFile`-Unterklassen realisiert. Der naheliegende Gedanke verschiedene Bildformate als Unterklassen von `Image` zu realisieren, hat den Nachteil, daß ein Bild dann nur in einem Format gelesen und geschrieben werden kann. Deshalb wurde hier der Ansatz gewählt, die Verfahren zum Lesen und Schreiben unterschiedlicher Formate in einer von `Image` unabhängigen Hierarchie zu kapseln. Ein neues Bildformat läßt sich leicht in Form einer neuen Klasse zufügen.

Einige Bildformate erforden eine Umrechnung der Beleuchtungswerte auf andere Wertebereiche sowie eventuell eine Quantisierung. Es existieren Klassen zur linearen Abbildung (`QuantizeLinear`) und zu einer Gamma-korrigierten Umrechnung [24] (`QuantizeGamme`).

### 5.7.5 Monte Carlo-Ray Tracing mit adaptiven Dichtefunktionen

Die in der `RadaRT`-Bibliothek verfügbaren Körper sind grundsätzlich als parametrisierte Flächen dargestellt, um für alle Körper auch Radiosity-Berechnungen zu ermöglichen. Jeder Fläche, die wie in Abschnitt 5.7.1 dargestellt eine Unterklasse von `AbstPrim` ist, können optische Eigenschaften zugewiesen werden (Abb. 5.16). Diese Eigenschaften können für Vorder- und Rückseite unterschiedlich sein. Es handelt sich dabei um die Eigenschaft, Licht abzustrahlen und um das Reflexionsverhalten.

Emittiert eine Seite einer Fläche Licht, wird dies in der Szene durch Zuweisung eines `AbstLightSource`-Objektes modelliert. Zur Zeit ist nur rein diffuses Strahlungsverhalten in Form der Klasse `LightSourceDiffuse` realisiert. Auch hier können andere Charakteristiken wieder leicht durch Zufügen einer neuen Unterklasse von `AbstLightSource` ermöglicht werden.

Das Reflexionsverhalten wird in Form einer BRDF dargestellt. Allgemeine Eigenschaften sowie die gemeinsame Schnittstelle werden in der abstrakten Klasse `AbstBRDF` definiert. Realisierungen liegen in Form rein diffuser Reflexion (`BRDFDiffuse`) und perfekt spiegelnder Reflexion (`BRDFMirror`) vor. Optional hat ein `AbstBRDF`-Objekt einen Verweis auf ein `DensityGrid`-Objekt. Ein `DensityGrid` enthält in einer Gitter-Unterteilung der Fläche adaptive, hierarchische Dichtefunktionen. Ist der Verweis auf ein solches Objekt definiert, wird automatisch das in Kapitel 3 vorgestellte Verfahren zur Nutzung adaptiver Dichtefunktionen eingesetzt. Ist eine solche Referenz nicht vorhanden, werden die Samples nach der BRDF einschließlich des Kosinus-Terms verteilt. Da der Verweis optional ist, kann für jede Seite einer Fläche einzeln festgelegt werden, ob das adaptive Verfahren zum Einsatz kommen soll.

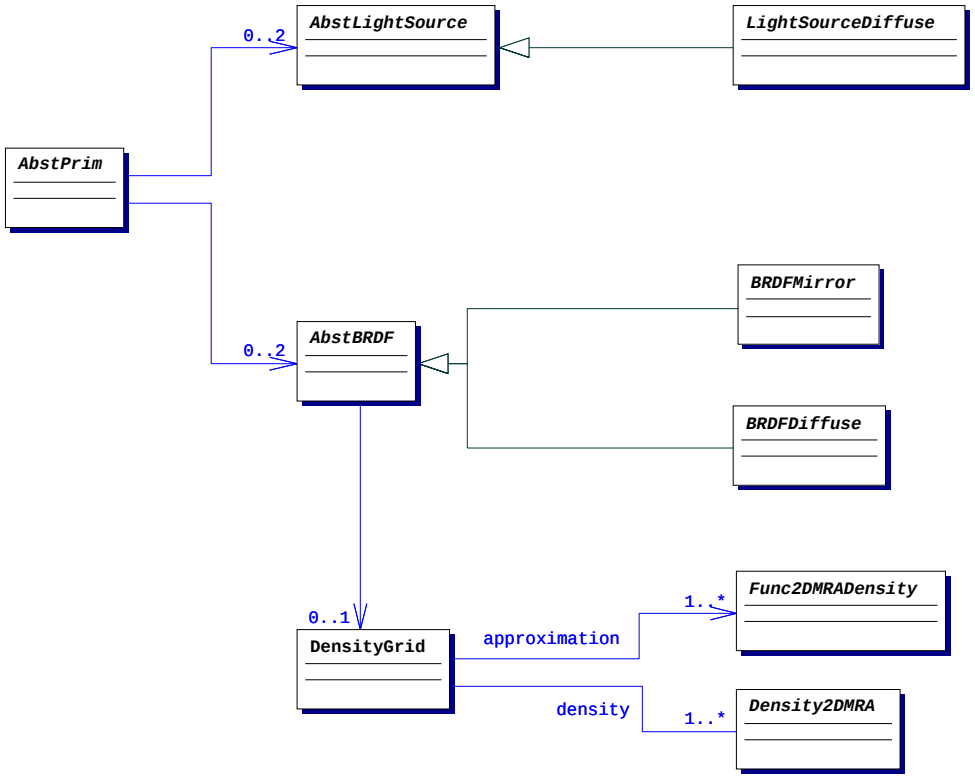



Abbildung 5.16: Klassendiagramm für Monte Carlo-Ray Tracing mit adaptiven Dichtefunktionen

Der Verweis von einem `DensityGrid`-Objekt zur Approximation (`Func2DMRADensity`) und zur Dichtefunktion (`Density2DMRA`) ist die einzige Verbindung zwischen der `WaveFunc`-Bibliothek und der `RadaRT`-Bibliothek. Hier zeigt sich deutlich, wie klar in der Architektur des Systems das mathematische Konzept von der Anwendung getrennt werden konnte.

### 5.7.6 Radiosity-Berechnung

Um eine Radiosity-Berechnung durchzuführen, müssen den Flächen der Szene (`AbstPrim`-Objekte) Möglichkeiten zur Speicherung der Radiosity-Funktion zugewiesen werden. Diese liegen in Form von Unterklassen der abstrakten Oberklasse `AbstRadioSurface` vor. Zur Zeit sind die gitterartige Unterteilung in der Klasse `RadioSurfaceGrid` und eine hierarchische, Quadtree-artige Unterteilung in der Klasse `RadioSurfaceQuadtree` verfügbar. Eine Radiosity-Darstellung wird stückweise konstant in Form von Patches gespeichert. Für die beiden Realisierungen gibt es korrespondierende Patch-Typen (`RadioPatchConstant` bzw. `RadioPatchQuadtree`), die Unterklassen des in der Klasse `AbstRadioPatch` vorliegenden Konzepts eines Radiosity-Patches sind. Eine Radiosity-Berechnung erfolgt durch Berechnung des Energieaustausches zwischen zwei Patches. Da alle Zugriffe jeweils über die in den abstrakten Oberklassen definierten Schnittstel-

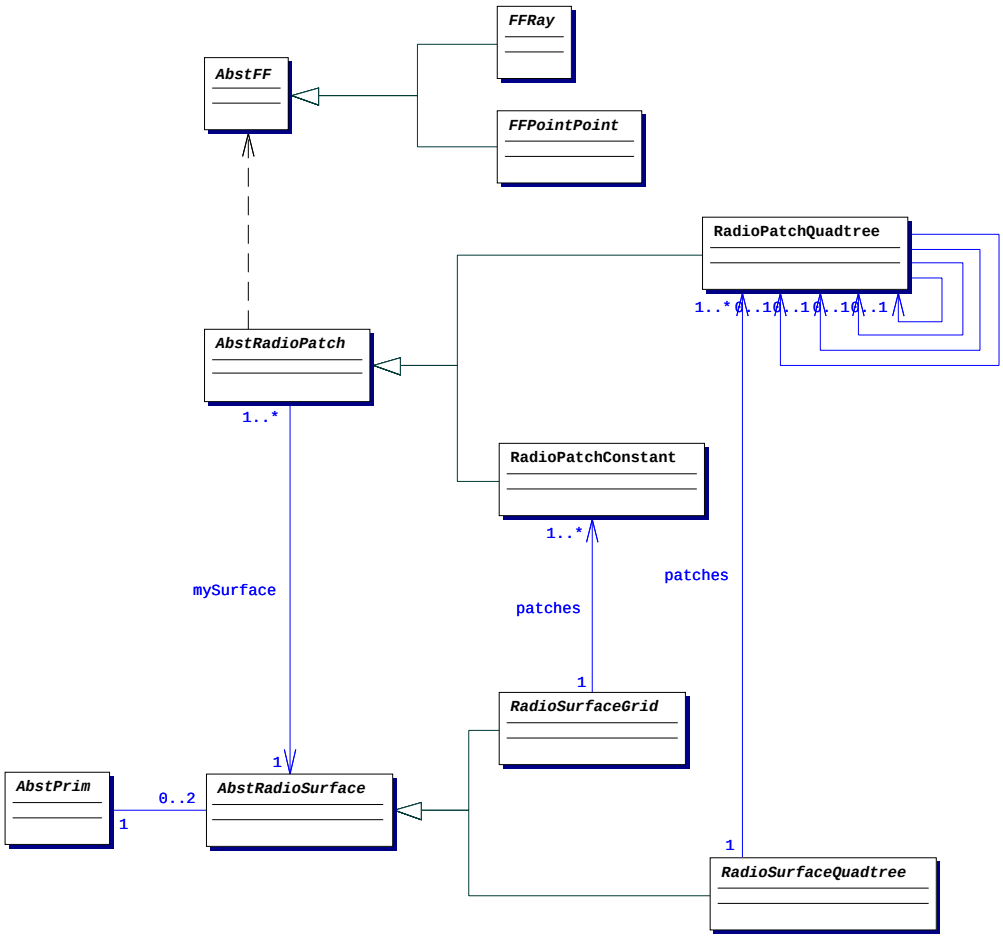



Abbildung 5.17: Klassendiagramm für Radiosity-Berechnung

len erfolgen, können in einer Szene beliebige `AbstRadioSurface`-Realisierungen gemischt eingesetzt werden.

Wichtigste Aufgabe zur Berechnung des Energieaustausches zweier Patches ist die Formfaktor-Berechnung. Verschiedene Verfahren zur Formfaktor-Berechnung sind als Unterklassen der abstrakten Klasse `AbstFF` implementiert. Eine schnelle Lösung ist in der Klasse `FFPointPoint` verfügbar. Der Formfaktor wird als differentieller Formfaktor zwischen den Patch-Mittelpunkten bestimmt. Eine exaktere, aber aufwendigere Lösung befindet sich in der Klasse `FFRay`. Die beiden Patches werden dafür in Sub-Patches unterteilt, zwischen denen dann ebenfalls eine Punkt-Punkt-Formfaktorberechnung erfolgt. Die Qualität dieser Berechnung wird durch die Feinheit der Patch-Unterteilung gesteuert. Beide Verfahren nutzen die im System vorhandene Ray Tracing-Funktionalität zur Verdeckungsberechnung. Auch hier können neue Verfahren zur Formfaktor-Berechnung in Form neuer Klassen zugefügt werden.

Eine Radiosity-Lösung kann auf beliebigen, nicht-planaren Flächen erfolgen. Einzige Bedingung ist, daß die Flächen parametrisierbar sind. Die Realisie-

rung der Formfaktor- und Radiosity-Berechnung basiert auf einer differential-geometrischen Schnittstelle, die in Form der Klasse `AbstGeometry` zur Verfügung gestellt wird. Wie in [76] ausgeführt ist eine Radiosity-Berechnung auf nicht-planaren Flächen eine in Radiosity-Programmen sehr selten realisierte Funktionalität. Im Rahmen der hier entwickelten Architektur ließ sich dieses Konzept leicht integrieren, weil klar zwischen der Geometrie eines Körpers (`AbstGeometry`) und der Darstellung der Radiosity (`AbstRadioSurface`) unterschieden wird. Eine `AbstRadioSurface`-Implementierung läßt sich in dieser Architektur mit jeder beliebigen Geometrie kombinieren, während die Architektur vieler Radiosity-Programme ausschließlich auf die Benutzung von Polygonen eingestellt ist. Wichtigster Unterschied zu einer Radiosity-Berechnung auf planaren Flächen ist eine Anpassung der Formfaktor-Berechnung, die ohne die differential-geometrische Darstellung der Flächen nicht möglich gewesen wäre.

# Kapitel 6

# Zusammenfassung

Gegenstand der fotorealistischen Bildsynthese ist die Erzeugung realistisch wirkender Rasterbilder aus dreidimensionalen Szenen durch die Berechnung von Lichtverteilungen. Grundlage der Berechnung ist die Bildsynthesegleichung (3.1) von Kajiya, als deren Lösung sich die gesuchte Lichtverteilung ergibt. Die Bildsynthesegleichung ist im allgemeinen nicht analytisch lösbar. Zur näherungsweisen Lösung sind im wesentlichen zwei Vorgehensweisen bekannt: Monte Carlo-Abtastverfahren und Finite-Elemente-Verfahren. In dieser Arbeit wird eine Verbesserung von Monte Carlo-Strahlverfolgung entwickelt, die bei gegebener Stichprobenanzahl Bilder signifikant besserer Qualität als bisher bekannte Verfahren liefert. Dies wird durch adaptive hierarchische Dichtefunktionen erreicht. Für die Berechnung von Bildern für den Fall der uniformen diffusen Reflexion („Radiosity-Berechnung“) nach dem klassischen Finite-Elemente-Ansatz mit konstanten Basisfunktionen werden eine Reihe von Interpolationsverfahren für die resultierende Radiosity-Funktion zur Verbesserung der visuellen Bilddarstellung angegeben. Diese basieren insbesondere auf Verfahren für die Streudateninterpolation. Durch empirische Analyse wird eine Bewertung der untersuchten Verfahren durchgeführt. Für die Implementierung und Analyse der Verfahren wurde eine Software-Architektur entwickelt. Designziele waren die Entwicklung einer flexibel änderbaren und erweiterbaren Testumgebung sowie die Ermöglichung der vergleichbaren empirischen Effizienzanalyse verschiedener Lösungsverfahren, indem soweit wie möglich dieselbe Implementierungsbasis genutzt wird und nur in wirklich unterschiedlichen Teilen unterschiedliche Algorithmen und deren Implementierung zum Einsatz kommen.

Beide Ansätze lassen sich relativ einfach und erfolgversprechend in existierende Systeme zur Bilderzeugung integrieren.

Wie bereits in Kapitel 3 ausgeführt, lohnt sich der Einsatz adaptiver Dichtefunktionen umso stärker, je ungerichteter die BRDF der betrachteten Fläche ist, also genau dann, wenn herkömmliches Monte Carlo-Ray Tracing ineffizient wird. Somit bietet es sich an, adaptive Dichtefunktionen auf den Flächen der Szene einzusetzen, die einen signifikanten diffusen Reflexions-Anteil aufweisen. An überwiegend gerichtet spiegelnden Flächen kann mit dem herkömmlichen Ansatz gearbeitet werden, die Strahlen nach der BRDF zu verteilen. Der Einsatz

adaptiver Dichtefunktionen kann zum Beispiel abhängig von einem Schwellwert für den diffusen Anteil der Reflexion für bestimmte Flächen in einer Szene aktiviert werden.

Eine lohnenswerte Erweiterung des in dieser Arbeit vorgestellten Verfahrens wäre die Kombination der Nutzung adaptiver Dichtefunktionen mit dem Quasi-Monte Carlo-Ansatz [45]. Auch der Quasi-Monte Carlo-Ansatz profitiert von verbesserten Dichtefunktionen, insofern ist aus der Kombination dieser Ansätze eine weitere Effizienzsteigerung zu erwarten.

Das in Kapitel 4 vorgestellte Verfahren zur verbesserten Interpolation von Radiosity-Funktionen kann in jedes auf stückweise konstanten Basisfunktionen basierende Radiosity-Verfahren, wie progressives oder hierarchisches Radiosity, integriert werden. Auch hier kann der Einsatz selektiv auf wichtige Flächen der Szene beschränkt werden. So ist zum Beispiel die visuelle Qualität von Schattenverläufen auf Böden und Wänden eines Raumes wichtig, weil Fehler in diesen Bereichen dem Betrachter stärker auffallen als auf kleineren Oberflächen der Szene. Eine Möglichkeit ist es also, den neuen Ansatz für alle Flächen einzusetzen, die eine gewisse Größe überschreiten, während die Radiosity-Funktionen kleinerer Flächen mit einfacheren Verfahren interpoliert werden. Der vorgestellte Ansatz läßt sich sowohl in die strahlbasierte Bilderzeugung integrieren, indem die Interpolationsfunktionen gespeichert und an getroffenen Punkten der Flächen jeweils neu ausgewertet werden, als auch in die interaktive, hardwarebasierte Darstellung, indem aus den Interpolationsfunktionen Texturen generiert werden.

# Anhang A

# Stochastische Grundbegriffe

In diesem Anhang werden die zum Verständnis des Monte Carlo-Integrationsverfahrens benötigten Begriffe aus dem Gebiet der Wahrscheinlichkeitsrechnung vorgestellt.

## A.1 Zufallsvariablen

Eine *Zufallsvariable* oder *Zufallsgröße* $X$ ist eine reelle Variable, die in Abhängigkeit vom Zufall verschiedene Werte annimmt. Zufallsvariablen sind oftmals Ausgänge von Experimenten wie der zufällige Fehler einer Messung oder die Lebensdauer einer Glühlampe.

Die mathematischen Eigenschaften einer Zufallsvariable $X$ werden durch ihre *Verteilungsfunktion* beschrieben. Die Verteilungsfunktion $P(x)$ der Zufallsvariablen $X$ gibt die Wahrscheinlichkeit dafür an, daß die Zufallsvariable einen Wert kleiner als $x$ annimmt:

$$P(x) = Prob(X < x) \tag{A.1}$$

Die Wahrscheinlichkeit dafür, daß die Zufallsvariable in ein gegebenes halboffenes Intervall fällt, ist dann:

$$Prob(a \leq X < b) = P(b) - P(a)$$

Charakteristische Eigenschaften der Verteilungsfunktion sind [6]:

- Grenzwerte:

$$\lim_{x \to -\infty} P(x) = 0 \tag{A.2}$$

$$\lim_{x \to \infty} P(x) = 1 \tag{A.3}$$

- $P(x)$ ist monoton nicht abnehmend ($x_1 < x_2 \Rightarrow P(x_1) \leq P(x_2)$)
- $P(x)$ ist linksseitig stetig

### A.1.1 Stetige Zufallsvariablen

Eine Zufallsvariable heißt stetig, wenn sich ihre Verteilungsfunktion darstellen läßt als:

$$P(x) = \int_{-\infty}^{x} p(t)\,\mathrm{d}t \tag{A.4}$$

Die Funktion $p$ heißt *Dichte der Verteilung* oder *Wahrscheinlichkeitsdichtefunktion* („WDF“, „probability density function“, „pdf“).

Aus den charakteristischen Eigenschaften der Verteilungsfunktion folgen bestimmte Eigenschaften der Dichte:

- aus der Monotonie der Verteilungsfunktion folgt, daß die Dichte eine nichtnegative Funktion ist:

  $$p(x) \geq 0 \quad \text{für alle } x \in \mathbb{R}$$

- aus dem Grenzwert (A.3) der Verteilungsfunktion ergibt sich:

  $$\int_{-\infty}^{\infty} p(x)\,\mathrm{d}x = 1 \tag{A.5}$$

  Man sagt zu dieser Eigenschaft auch, $p$ hat *Einheitsmasse*.

- bei gegebener Dichte berechnet sich die Wahrscheinlichkeit, daß die Zufallsvariable in ein vorgegebenes halboffenes Intervall fällt, als:

  $$Prob(a \leq X < b) = \int_{a}^{b} p(x)\,\mathrm{d}x$$

Ist eine Zufallsvariable $X$ entsprechend einer Dichte $p$ verteilt, wird das durch $X \sim p$ dargestellt.

### A.1.2 Gleichverteilte Zufallsvariablen

Ein wichtiger Spezialfall stetiger Zufallsvariablen sind gleichverteilte Zufallsvariablen. Eine Zufallsvariable heißt gleichverteilt im Intervall $[a, b]$, wenn ihre Dichte in diesem Intervall konstant und außerhalb des Intervalls 0 ist. Aus (A.5) folgt dann

$$p(x) = \frac{1}{b-a}$$

und aus (A.2), (A.3), (A.4) folgt

$$P(x) = \begin{cases} 0 & \text{für} \quad x \leq a \\ \dfrac{x-a}{b-a} & \text{für} \quad a < x < b \\ 1 & \text{für} \quad b \leq x \end{cases}$$

Anschaulich gesehen nimmt eine gleichverteilte Zufallsvariable jeden beliebigen Wert aus dem Intervall $[a, b]$ mit gleicher Wahrscheinlichkeit an.

## A.2 Erwartungswert

$$EX := \int_{-\infty}^{\infty} x\, p(x)\, \mathrm{d}x$$

wird *Erwartungswert* von $X$ genannt. Der Erwartungswert ist ein Maß für das Zentrum einer Verteilung, wenn man ihn als Schwerpunkt der Massenverteilung deutet, die durch die Massendichte $p$ beschrieben wird.

Der Erwartungswert hat folgende Eigenschaften:

$$\begin{aligned} E(aX+b) &= a\,EX + b && \text{(A.6)} \\ E(X+Y) &= EX + EY && \text{(A.7)} \\ E(g(X)) &= \int_{-\infty}^{\infty} g(x)\, p(x)\, \mathrm{d}x && \text{(A.8)} \end{aligned}$$

mit Konstanten $a, b \in \mathbb{R}$ und einer Funktion $g : \mathbb{R} \to \mathbb{R}$.

## A.3 Varianz

$$DX := \int_{-\infty}^{\infty} (x - EX)^2\, p(x)\, \mathrm{d}x$$

heißt *Varianz* von $X$. Die Varianz ist ein Maß für die Streuung der Verteilung um den Erwartungswert.

Es gilt

$$\begin{aligned} DX &= E(X - EX)^2 \\ &= E(X^2 - 2X\,EX + (EX)^2) \\ &= EX^2 - E(2X\,EX) + E(EX)^2 \\ &= EX^2 - 2(EX)^2 + (EX)^2 \\ &= EX^2 - (EX)^2 && \text{(A.9)} \end{aligned}$$

## A.4 Standardabweichung

Die Quadratwurzel der Varianz

$$\sigma := \sqrt{DX}$$

heißt *Streuung* oder *Standardabweichung* von $X$. Sie gibt die zu erwartende Abweichung der Zufallsvariablen von ihrem Erwartungswert an.

## A.5 Gesetz der großen Zahl

Gegeben sei eine Folge $\{X_n\}$ unabhängiger Zufallsvariablen, die alle nach derselben Dichte $p$ verteilt sind. Unabhängigkeit bedeutet, daß der Wert, den eine Zufallsvariable $X_i$ annimmt, nicht vom Wert beeinflußt wird, den andere Zufallsvariablen annehmen. Man bezeichnet $\{X_n\}$ auch als eine Folge *unabhängiger identisch verteilter Zufallsvariablen*. Eine solche Folge kann z. B. durch wiederholtes Ausführen eines Zufallsexperiments erzeugt werden, wobei der Ausgang des $i$-ten Experiments den Wert von $X_i$ ergibt.

Bildet man den Mittelwert über diese Folge von Zufallsvariablen, erhält man eine Abschätzung für den Erwartungswert einer Zufallsvariablen $X \sim p$:

$$EX \approx \frac{1}{n} \sum_{i=1}^{n} X_i \tag{A.10}$$

Das *starke Gesetz der großen Zahl* macht eine Aussage über die Konvergenz dieser Abschätzung:

$$Prob\left( EX = \lim_{n \to \infty} \frac{1}{n} \sum_{i=1}^{n} X_i \right) = 1 \tag{A.11}$$

Anschaulich gesehen besagt dieses Gesetz, daß die Abschätzung (A.10) für genügend hohe Samplezahlen gegen den Erwartungswert $EX$ konvergiert.

# Anhang B

# Grundbegriffe der Linearen Algebra

In dieser Arbeit werden viele Begriffe aus dem Gebiet der Linearen Algebra benutzt. Bei den betrachteten Vektorräumen handelt es sich speziell um Funktionsräume. Manche Begriffe klingen in diesem Zusammenhang ungewohnt (z. B. Orthogonalität zweier Funktionen), deshalb werden in diesem Anhang die benötigten Grundbegriffe insbesondere auch für Funktionsräume erläutert.

## B.1 Gruppen

**Definition B.1 (Gruppe)** *Eine Gruppe besteht aus einer Menge $G$ und einer Operation $\circ$, die jedem geordneten Paar $(a, b)$ von Elementen aus $G$ eindeutig ein mit $a \circ b$ bezeichnetes Element von $G$ so zuordnet, daß folgende Axiome erfüllt sind $(a, b, c \in G)$:*

1. $(a \circ b) \circ c = a \circ (b \circ c)$ *(Assoziativgesetz)*
2. *Es gibt ein Element $e \in G$ mit $e \circ a = a$ für alle $a \in G$ (neutrales Element)*
3. *Zu jedem $a \in G$ existiert ein Element $a' \in G$ mit $a' \circ a = e$ (inverses Element)*

*Die Gruppe heißt abelsche oder kommutative Gruppe, wenn außerdem folgendes Axiom erfüllt ist:*

4. $a \circ b = b \circ a$ *(Kommutativgesetz)*

Das erste Axiom besagt, daß die Klammerung beliebig ist, man kann deshalb ganz auf sie verzichten, was im folgenden Text auch getan wird.

Es läßt sich zeigen, daß das Element $e$ durch die Axiome eindeutig bestimmt ist, es wird das *neutrale Element* der Gruppe genannt. Weiter läßt sich zeigen,

daß das Element $a'$ durch $a$ eindeutig bestimmt wird, man nennt dieses Element *inverses Element* von $a$ und schreibt es im allgemeinen als $a^{-1}$. Für Beweise sei auf [47] verwiesen.

Bei additiv geschriebenen Gruppen, in denen die Gruppenverknüpfung als Operator + geschrieben wird, bezeichnet man das neutrale Element mit 0 und das zu $a$ inverse Element mit $-a$.

**Beispiel B.1** *Die Menge der reellen Zahlen bildet hinsichtlich der gewöhnlichen Addition eine abelsche Gruppe (additive Gruppe der reellen Zahlen). Das neutrale Element ist die Zahl 0, das zu a inverse Element ist die Zahl $-a$.*

**Beispiel B.2** *Es sei $V = \{(x, y)$ mit $x, y \in \mathbb{R}\}$ und $+$ die Operation $(x_1, y_1) + (x_2, y_2) = (x_1 + x_2, y_1 + y_2)$. Dann ist $V$ mit diesem Operator eine abelsche Gruppe.*

Im Beispiel B.2 ist zu beachten, daß das Symbol + in dem Ausdruck $(x_1, y_1) + (x_2, y_2)$ die Gruppenverknüpfung für $V$ bezeichnet, während das gleiche Symbol im Ausdruck $x_1 + x_2$ die gewöhnliche Addition der reellen Zahlen bezeichnet. Aus den Operanden folgt jeweils, welche Operation durch das Symbol bezeichnet wird.

**Beispiel B.3** *Es sei $L^2(0, 1)$ die Menge aller Funktionen $f(x) : [0, 1] \to \mathbb{R}$, für die $\int_{x=0}^{1} |f(x)|^2 \, dx < \infty$ gilt. Die Operation $+$ zweier Funktionen $f(x)$, $g(x)$ aus $L^2(0, 1)$ sei definiert als $(f + g)(x) \equiv f(x) + g(x)$. Dann ist $L^2(0, 1)$ mit dieser Operation eine abelsche Gruppe. Neutrales Element ist die Nullfunktion, die konstant 0 als Funktionswert liefert.*

Bei multiplikativen Gruppen, in denen die Gruppenverknüpfung als Operator $\cdot$ geschrieben wird, bezeichnet man das neutrale Element mit 1. Oftmals wird der Verknüpfungsoperator weggelassen, statt $a \cdot b$ wird kurz $ab$ geschrieben.

**Beispiel B.4** *Die Menge der von 0 verschiedenen reellen Zahlen bildet hinsichtlich der gewöhnlichen Multiplikation eine abelsche Gruppe (multiplikative Gruppe der reellen Zahlen). Das neutrale Element ist die Zahl 1, das zu a inverse Element ist die Zahl $\frac{1}{a}$.*

## B.2 Körper

**Definition B.2 (Körper)** *Ein Körper besteht aus einer Menge $K$ und zwei Operationen $+$ und $\cdot$, die jedem geordneten Paar $(a, b)$ von Elementen aus $K$ eindeutig ein Element $a + b$ bzw. $a \cdot b$ von $K$ so zuordnen, daß folgende Axiome erfüllt sind $(a, b, c \in K)$:*

1. $(a + b) + c = a + (b + c)$ *(Assoziativität der Addition)*

2. $a + b = b + a$ *(Kommutativität der Addition)*

3. *Es gibt ein Element* $0 \in K$ *mit* $0 + a = a$ *für alle* $a \in K$

4. *Zu jedem* $a \in K$ *existiert ein Element* $-a$ *in* $K$ *mit* $(-a) + a = 0$

5. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$ *(Assoziativität der Multiplikation)*

6. *Es gibt ein Element* $1 \in K$ *mit* $1 \cdot a = a$ *für alle* $a \in K$

7. *Zu jedem* $a \in K$ *mit* $a \neq 0$ *existiert ein Element* $a^{-1}$ *in* $K$ *mit* $a^{-1} \cdot a = 1$

8. $a \cdot (b + c) = a \cdot b + a \cdot c$ *und* $(b + c) \cdot a = b \cdot a + c \cdot a$ *(Distributivität)*

9. $1 \neq 0$

*Gilt zusätzlich das Axiom*

10. $a \cdot b = b \cdot a$ *(Kommutativität der Multiplikation)*

*so wird* $K$ *ein kommutativer Körper genannt.*

Eine übliche Konvention zur Klammereinsparung ist die Regel, daß die Multiplikation stärker bindet als die Addition, diese Regel ist in Axiom 8 bereits vorausgesetzt worden.

Aus den Axiomen ist ersichtlich, daß ein Körper bezüglich der Addition eine abelsche Gruppe bildet und daß die von 0 verschiedenen Elemente bezüglich der Multiplikation eine Gruppe bilden, im Falle eines kommutativen Körpers ebenfalls eine abelsche Gruppe.

**Beispiel B.5** *Die reellen Zahlen bilden hinsichtlich der üblichen Addition und Multiplikation einen kommutativen Körper.*

## B.3 Vektorräume

**Definition B.3 (Vektorraum)** *Ein Vektorraum besteht aus einer abelschen Gruppe* $V$*, deren Elemente Vektoren genannt werden, einem kommutativen Skalarenkörper* $K$*, dessen Elemente Skalare genannt werden, und einer Multiplikation, die jedem geordneten Paar* $(a, \mathbf{v})$ *mit* $a \in K$ *und* $\mathbf{v} \in V$ *eindeutig einen Vektor* $a\mathbf{v} \in V$ *so zuordnet, daß folgende Axiome erfüllt sind* ($a, b \in K$, $\mathbf{u}, \mathbf{v} \in V$)*:*

1. $(ab)\mathbf{u} = a(b\mathbf{u})$ *(Assoziativität)*

2. $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
   $(a + b)\mathbf{u} = a\mathbf{u} + b\mathbf{u}$
   *(Distributivität)*

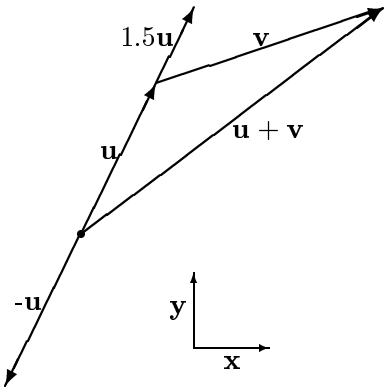



Abbildung B.1: Vektoren in der Ebene

*3.* $1\mathbf{u} = \mathbf{u}$

Um die Notation zu vereinfachen, wird festgelegt, daß die Multiplikation mit Skalaren stärker bindet als die Vektoraddition, statt $(a\mathbf{u}) + \mathbf{v}$ wird also kurz $a\mathbf{u} + \mathbf{v}$ geschrieben.

Auch hier muß wieder darauf geachtet werden, daß das gleiche Symbol verschiedene Operationen bezeichnet, so wird im Ausdruck $(a+b)\mathbf{u}$ mit dem Symbol + die Addition im Skalarenkörper bezeichnet, während es sich im Ausdruck $a\mathbf{u} + b\mathbf{u}$ um die Vektoraddition, also die Gruppenoperation von $V$ handelt.

Wenn der Skalarenkörper $K$ explizit benannt werden soll, spricht man von einem Vektorraum $V$ über $K$. Im folgenden wird grundsätzlich als Skalarenkörper der Körper $\mathbb{R}$ der reellen Zahlen mit der üblichen Addition und Multiplikation vorausgesetzt.

**Beispiel B.6** *Die in Beispiel B.2 vorgestellte Gruppe bildet zusammen mit der Multiplikation* $a \cdot (x, y) = (ax, ay)$ *(*$a \in \mathbb{R}$*,* $(x, y) \in V$*) einen Vektorraum über* $\mathbb{R}$*. Man kann sich die Elemente z. B. anschaulich als Pfeile in der Ebene vorstellen. Die Komponenten* $x$ *und* $y$ *entsprechen dann den Anteilen in* $x$*- und* $y$*-Richtung des Koordinatensystems. Die Vektoraddition entspricht anschaulich dem Aneinanderhängen zweier Pfeile, die Multiplikation mit einem Skalar* $a \in \mathbb{R}$ *ergibt den Pfeil, dessen Länge der* $|a|$*-fachen Länge des ursprünglichen Pfeils entspricht, falls* $a$ *negativ ist, wird die Pfeilrichtung umgekehrt (Abb. B.1).*

**Beispiel B.7** *Die in Beispiel B.3 vorgestellte Gruppe* $L^2(0, 1)$ *bildet zusammen mit der folgendermaßen definierten Multiplikation* $(a \cdot f)(x) \equiv a \cdot f(x)$ *(*$a \in \mathbb{R}$*,* $f(x) \in L^2(0, 1)$*) einen Vektorraum über dem Skalarenkörper* $\mathbb{R}$*.*

Ein Raum, wie der in Beispiel B.7 vorgestellte $L^2(0, 1)$, dessen Vektoren mathematische Funktionen sind, wird auch als *Funktionsraum* bezeichnet.

## B.4 Basis

**Definition B.4 (Lineare Unabhängigkeit)** *Endlich viele Vektoren* $\mathbf{v}_1, \ldots, \mathbf{v}_n$ *eines Vektorraums* $V$ *über* $K$ *heißen linear unabhängig, wenn aus* $c_1\mathbf{v}_1 + \cdots + c_n\mathbf{v}_n = \mathbf{0}$, $c_1, \ldots, c_n \in K$ *stets* $c_1 = \cdots = c_n = 0$ *folgt.*

*Eine Teilmenge* $M$ *eines Vektorraums* $V$ *heißt linear unabhängig, wenn je endlich viele verschiedene Vektoren aus* $M$ *linear unabhängig sind.*

**Beispiel B.8** *Je zwei Funktionen* $x^j, x^k \in L^2(0,1)$, $j,k \in \mathbb{N}$ *sind genau dann linear unabhängig, wenn* $j \neq k$. *Hieraus folgt, daß jede Teilmenge der Menge* $\{x^i \in L^2(0,1), i \in \mathbb{N}\}$ *linear unabhängig ist.*

**Definition B.5 (Basis eines Vektorraums)** *Eine Teilmenge* $B$ *eines Vektorraums* $V$ *heißt eine Basis von* $V$, *wenn* $B$ *linear unabhängig ist und den ganzen Raum* $V$ *aufspannt, das heißt, wenn jeder Vektor* $x \in V$ *als*

$$v = \sum_j c_j \mathbf{b}_j \qquad c_j \in \mathbb{R} \qquad \mathbf{b}_j \in B$$

*geschrieben werden kann.*

*Wenn* $B$ *eine endliche Zahl von Elementen* $\mathbf{b}_1, \ldots, \mathbf{b}_n$ *hat, ist* $V$ *endlich-dimensional und hat die Dimension* $n$. *Sonst wird* $V$ *als unendlich-dimensional bezeichnet.*

Der von einer Basis $B$ aufgespannte Vektorraum wird im folgenden als $V_B$ bezeichnet.

**Beispiel B.9** *Eine Basis für den aus Beispiel B.6 bekannten Vektorraum, bilden die beiden in Abb. B.1 dargestellten Vektoren* $\mathbf{x} = (1,0)$ *und* $\mathbf{y} = (0,1)$. *Vektor* $\mathbf{u}$ *hat in dieser Basis z. B. die Koeffizienten* $c_1 = 1$, $c_2 = 2$ *und somit die Darstellung* $\mathbf{u} = \mathbf{x} + 2\mathbf{y}$. *Dieser Vektorraum hat die Dimension 2.*

Die Basis eines Funktionsraumes wird auch als *Funktionsbasis* bezeichnet. Ihre Elemente werden *Basisfunktionen* genannt.

**Beispiel B.10** *Die aus Beispiel B.8 bekannte Menge* $\{x^i \in L^2(0,1), i \in \mathbb{N}\}$ *ist eine unendlich-dimensionale Basis für den Raum der Polynome* $\{a_0 + a_1 x + \ldots + a_n x^n, a_0, \ldots, a_n \in \mathbb{R}, n \in \mathbb{N}\}$.

*Der Vektorraum* $L^2(0,1)$ *ist unendlich-dimensional, da der Raum der Polynome eine Teilmenge von* $L^2(0,1)$ *und, wie oben gesehen, unendlich-dimensional ist.*

Spannen zwei Basen $N$ und $M$ denselben Vektorraum $V = V_N = V_M$ auf, kann man einen beliebigen Vektor $\mathbf{v} \in V$ in jeder dieser Basen darstellen. Der Wechsel von der Darstellung des Vektors in einer Basis in eine andere läßt sich als Matrix darstellen. $\mathbf{c} = [c_j]$ seien die als Spaltenvektor geschriebenen

Koeffizienten von $\mathbf{v}$ in der Basis $N$. Dann beschreibt eine quadratische Matrix $\mathbf{M}_{N\to M}$ den *Basiswechsel*. Die Darstellung von $\mathbf{v}$ in der Basis $M$ erhält man dann als $\mathbf{d} = \mathbf{M}_{N\to M}\,\mathbf{c}$. Die umgekehrte Transformation ist mit $\mathbf{M}_{M\to N} = \mathbf{M}_{N\to M}^{-1}$ gegeben: $\mathbf{c} = \mathbf{M}_{N\to M}^{-1}\,\mathbf{d}$.

**Beispiel B.11** *$G = \{g_0(x) = x, g_1(x) = 1\}$ und $L = \{l_0(x) = x, l_1(x) = 1 - x\}$ sind zwei Basen für den Raum der linearen Funktionen $p(x) = mx + b$. Die Funktion $3x + 2$ hat in der Basis $G$ die Koeffizienten $c_0 = 3$, $c_1 = 2$. In der Basis $L$ hat sie die Darstellung:*

$$\mathbf{d} = \mathbf{M}_{G\to L}\,\mathbf{c} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

*Eine Kontrolle ergibt:*

$$5\,l_0(x) + 2\,l_1(x) = 5x + 2(1 - x) = 3x + 2$$

*Die Umkehrung erhält man als:*

$$\mathbf{c} = \mathbf{M}_{L\to G}\,\mathbf{d} = \mathbf{M}_{G\to L}^{-1}\,\mathbf{d} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

## B.5 Inneres Produkt, Orthogonalität

**Definition B.6 (Inneres Produkt)** *$V$ sei ein Vektorraum. Ein inneres Produkt $\langle\cdot,\cdot\rangle$ ist eine Abbildung von $V \times V$ nach $\mathbb{R}$, die folgende Eigenschaften erfüllt ($\mathbf{u}, \mathbf{v} \in V$, $a, b \in \mathbb{R}$):*

1. $\langle \mathbf{u}, \mathbf{v}\rangle = \langle \mathbf{v}, \mathbf{u}\rangle$ *(Symmetrie)*
2. $\langle a\mathbf{u} + b\mathbf{v}, \mathbf{w}\rangle = a\langle \mathbf{u}, \mathbf{w}\rangle + b\langle \mathbf{v}, \mathbf{w}\rangle$ *(Bilinearität)*
3. $\langle \mathbf{u}, \mathbf{u}\rangle > 0$ *für alle* $\mathbf{u} \neq 0$ *(positiv definit)*

Das innere Produkt wird auch als *Skalarprodukt* bezeichnet.

**Beispiel B.12** *Für den in Beispiel B.6 genannten Vektorraum $V$ definiert $\langle (x_1, y_1), (x_2, y_2)\rangle := x_1x_2 + y_1y_2$ ein inneres Produkt.*

**Beispiel B.13** *Als Standard-Skalarprodukt für Funktionsräume wie dem aus Beispiel B.7 bekannten $L^2(0, 1)$ wird $\langle f, g\rangle := \int_0^1 f(x)\,g(x)\,dx$ benutzt.*

*Eine andere Möglichkeit zur Definition eines Skalarprodukts erhält man mit einer positiven Gewichtsfunktion $w(x)$ als $\langle f, g\rangle := \int_0^1 w(x)\,f(x)\,g(x)\,dx$.*

Im folgenden soll, soweit nichts anderes angegeben ist, grundsätzlich das in Beispiel B.13 definierte Standard-Skalarprodukt benutzt werden.

Mit dem inneren Produkt kann der Begriff der Orthogonalität eingeführt werden.

**Definition B.7 (Orthogonalität)** *Zwei Vektoren* $\mathbf{u}, \mathbf{v}$ *werden als orthogonal (senkrecht) bezeichnet, wenn* $\langle \mathbf{u}, \mathbf{v} \rangle = 0$.

**Beispiel B.14** *Die Funktionen*

$$c_0(x) := \begin{cases} 1 & x < 1/2 \\ 0 & sonst \end{cases} \qquad c_1(x) := \begin{cases} 0 & x < 1/2 \\ 1 & sonst \end{cases}$$

*sind orthogonal zueinander:*

$$\begin{aligned} \langle c_0, c_1 \rangle &= \int_0^1 c_0(x) \cdot c_1(x) dx \\ &= \int_0^{1/2} 1 \cdot 0 \, dx + \int_{1/2}^1 0 \cdot 1 \, dx \\ &= 0 \end{aligned}$$

*Die Funktionen* $l_0(x) := x$ *und* $l_1(x) := 1 - x$ *sind nicht orthogonal zueinander:*

$$\begin{aligned} \langle l_0, l_1 \rangle &= \int_0^1 x\,(1-x)\; dx \\ &= \frac{1}{2}x^2 - \frac{1}{3}x^3 \Big|_0^1 \\ &= \frac{1}{6} \end{aligned}$$

Es läßt sich leicht zeigen, daß eine Menge gegenseitig orthogonaler Vektoren auch linear unabhängig ist, für Beweise sei wieder auf [47] verwiesen.

**Definition B.8 (Orthogonale Basis)** *Eine Basis, deren Vektoren gegenseitig orthogonal sind, heißt orthogonale Basis.*

**Beispiel B.15** *Die aus Beispiel B.14 bekannten Funktionen* $c_o(x)$, $c_1(x)$ *bilden eine orthogonale Basis für den Raum der auf den Intervallen* $[0, 1/2)$, $[1/2, 1]$ *stückweise konstanten Funktionen.*

*Die Menge* $L = \{l_o(x) = x, l_1(x) = 1 - x\}$ *ist eine nicht-orthogonale Basis für den Raum der linearen Funktionen* $g(x) = mx + b$.

## B.6 Norm

**Definition B.9 (Norm)** *Ein Vektorraum* $V$ *heißt normierter Raum, wenn jedem Element* $\mathbf{v} \in V$ *eine reelle Zahl* $\|\mathbf{v}\| \geq 0$, *genannt die Norm des Vektors* $\mathbf{v}$, *zugeordnet ist und diese Norm die folgenden Eigenschaften hat (*$\mathbf{u}, \mathbf{v} \in V$, $a \in \mathbb{R}$*):*

1. $\|\mathbf{v}\| = 0$ *nur für* $\mathbf{v} = \mathbf{0}$
2. $\|a\mathbf{v}\| = |a|\,\|\mathbf{v}\|$
3. $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ *(Dreiecksungleichung)*

Eine Norm ist anschaulich gesehen eine Abbildung, die einem Vektor eine Länge zuordnet.

**Definition B.10 ($p$-Normen)** *In endlich-dimensionalen Vektorräumen ist die Familie der p-Normen definiert durch:*

$$\|\mathbf{u}\|_p := \left( \sum_j |u_j|^p \right)^{\frac{1}{p}}$$

Wenn $p$ gegen Unendlich geht, erhält man die Maximums-Norm:

$$\|\mathbf{u}\|_\infty = \max_j |u_j|$$

**Definition B.11 ($L^p$-Normen)** *In Funktionsräumen definiert man die $L^p$-Normen als:*

$$\|f\|_p := \left( \int_0^1 |f(x)|^p \, dx \right)^{\frac{1}{p}}$$

Die am häufigsten verwendete Norm ist die $L^2$-Norm, die bei Verwendung des Standard-Skalarprodukts auch als $\|f\| = \sqrt{\langle f, f \rangle}$ geschrieben werden kann.

Wenn $p$ gegen Unendlich geht, erhält man die Maximums-Norm für Funktionen:

$$\|f\|_\infty = \max_{x \in [0,1]} |f(x)|$$

**Definition B.12 (Orthonormale Basis)** *Ein Vektor* $\mathbf{u}$ *mit* $\|\mathbf{u}\| = 1$ *wird normiert genannt. Eine orthogonale Basis* $\{u_i\}$*, deren Vektoren in der $L^2$-Norm normiert sind, wird als orthonormale Basis bezeichnet. Das ist gleichwertig mit*

$$\langle \mathbf{u}_j, \mathbf{u}_k \rangle = \delta_{j,k}$$

**Beispiel B.16** *Die Vektoren* $\mathbf{x}$ *und* $\mathbf{y}$ *aus Beispiel B.9 sind eine orthonormale Basis für den Vektorraum aus Beispiel B.6.*

**Beispiel B.17** *Die Funktionen*

$$h_0(x) := \begin{cases} \sqrt{2} & x < 1/2 \\ 0 & sonst \end{cases} \qquad h_1(x) := \begin{cases} 0 & x < 1/2 \\ \sqrt{2} & sonst \end{cases}$$

*bilden eine orthonormale Basis für den Raum der auf den Intervallen* $[0, 1/2)$, $[1/2, 1]$ *stückweise konstanten Funktionen:*

$$\begin{aligned}
\langle h_0, h_0 \rangle &= \int_0^{1/2} \sqrt{2} \cdot \sqrt{2}\, dx &&= 1 \\
\langle h_0, h_1 \rangle &= \int_0^{1/2} \sqrt{2} \cdot 0\, dx + \int_{1/2}^{1} 0 \cdot \sqrt{2}\, dx &&= 0 \\
\langle h_1, h_0 \rangle &= \int_0^{1/2} 0 \cdot \sqrt{2}\, dx + \int_{1/2}^{1} \sqrt{2} \cdot 0\, dx &&= 0 \\
\langle h_1, h_1 \rangle &= \int_{1/2}^{1} \sqrt{2} \cdot \sqrt{2}\, dx &&= 1
\end{aligned}$$

## B.7 Funktionsapproximation

Im folgenden sei vorausgesetzt, daß alle Funktionen Elemente eines Funktionsraumes mit innerem Produkt und einer Norm sind. Man kann z. B. annehmen, daß alle Funktionen aus dem Funktionsraum $L^2(0, 1)$ stammen, das Standard-Skalarprodukt und eine der $L^p$-Normen benutzt werden.

Eine beliebige Funktion $f(x)$ ist im allgemeinen nicht in einer gegebenen Funktionsbasis $N = \{N_j(x), j = 1, \ldots, n\}$ darstellbar, das bedeutet, im allgemeinen ist $f(x)$ nicht in $V_N$ enthalten. Oftmals steht man aber vor der Aufgabe, die Funktion $\hat{f}(x) \in V_N$ zu finden, die möglichst nah an die gegebene Funktion $f(x)$ herankommt. Um den Abstand der Funktionen zu messen, benutzt man die gegebene Norm.

**Definition B.13 (Proximum)** *Zu einer gegebenen Funktion* $f(x)$ *und einer Funktionsbasis* $N$ *ist die Funktion* $\hat{f}(x) \in V_N$ *mit*

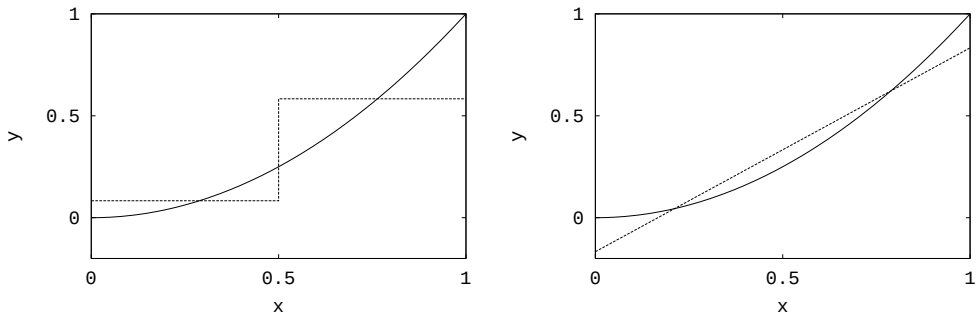
$$\hat{f} := \inf_{v \in V_N} \|f - v\|$$

*das Proximum von* $f$ *in* $N$*.*

Man bezeichnet $\hat{f}(x)$ auch als *Approximation* von $f$ in $N$ oder kann die Verwendung einer bestimmten Norm anzeigen, indem man z. B. von der $L^2$-Approximation von $f$ in $N$ spricht.

Bei Verwendung des Standard-Skalarprodukts und der $L^2$-Norm hat das Proximum $\hat{f}$ die spezielle Eigenschaft, daß die Differenzfunktion $r(x) := f(x) - \hat{f}(x)$ orthogonal zu allen Basisfunktionen ist, das heißt, sie enthält keinen Anteil mehr, der in der Basis darstellbar ist.

Die Bestimmung des Proximums besteht aus der konkreten Aufgabe der Berechnung der Koeffizienten $c_j$ von $\hat{f}(x) = \sum_j c_j N_j(x)$. Man spricht in diesem Zusammenhang auch davon, daß die *Projektion* von $f$ in die Basis $N$ gesucht




stückweise konstant (Basis $H$) linear (Basis $L$)

Abbildung B.2: Approximation von $f(x) = x^2$

wird. Im Falle einer orthonormalen Basis erhält man die Projektion als

$$\begin{aligned} c_j &= \langle f, N_j \rangle \\ \hat{f}(x) &= \sum_j \langle f, N_j \rangle \, N_j(x) \end{aligned} \tag{B.1}$$

**Beispiel B.18** *Das Proximum der Funktion $f(x) = x^2$ in der aus Beispiel B.17 bekannten Basis $H = \{h_0(x), h_1(x)\}$ ist:*

$$\begin{aligned} \hat{f}(x) &= \int_0^{1/2} \sqrt{2}\, x^2 \, dx \cdot h_0(x) + \int_{1/2}^{1} \sqrt{2}\, x^2 \, dx \cdot h_1(x) \\ &= \frac{1}{24}\sqrt{2}\, h_0(x) + \frac{7}{24}\sqrt{2}\, h_1(x) \end{aligned}$$

*(Abb. B.2, links)*

Ist die Basis nicht orthonormal, kann man die Projektion berechnen, falls die duale Basis $\tilde{N}$ bekannt ist.

**Definition B.14 (Duale Funktionsbasis)** *Die Basis $\tilde{N} = \{\tilde{N}_j(x)\}$ bezeichnet man als duale Basis zur (primären) Basis $N$, wenn*

$$\langle N_j, \tilde{N}_k \rangle = \delta_{j,k} \qquad j, k = 1, \ldots, n$$

*gilt.*

**Beispiel B.19** *Die zu $L = \{l_0(x) = x, l_1(x) = 1 - x\}$ duale Basis ist $\tilde{L} =$*

$\{\tilde{l}_0(x) = 6x - 2, \tilde{l}_1(x) = 4 - 6x\}$:

$$\begin{aligned}
\langle l_0, \tilde{l}_0 \rangle &= \int_0^1 x(6x-2)\,dx &&= 2x^3 - x^2\big|_0^1 &&= 1 \\
\langle l_0, \tilde{l}_1 \rangle &= \int_0^1 x(4-6x)\,dx &&= -2x^3 + 2x^2\big|_0^1 &&= 0 \\
\langle l_1, \tilde{l}_0 \rangle &= \int_0^1 (1-x)(6x-2)\,dx &&= -2x^3 + 4x^2 - 2x\big|_0^1 &&= 0 \\
\langle l_1, \tilde{l}_1 \rangle &= \int_0^1 (1-x)(4-6x)\,dx &&= 2x^3 - 5x^2 + 4x\big|_0^1 &&= 1
\end{aligned}$$

Wenn die duale Basis $\tilde{N}$ bekannt ist, erhält man die Darstellung von $\hat{f}(x)$ in der Basis $N$ als

$$\begin{aligned}
c_j &= \langle f, \tilde{N}_j \rangle \\
\hat{f}(x) &= \sum_{j=1}^{n} \langle f, \tilde{N}_j \rangle\, N_j(x)
\end{aligned} \tag{B.2}$$

Man kann auch die Repräsentation von $\hat{f}(x)$ in der dualen Basis bestimmen:

$$\begin{aligned}
\tilde{c}_j &= \langle f, N_j \rangle \\
\hat{f}(x) &= \sum_{j=1}^{n} \langle f, N_j \rangle\, \tilde{N}_j(x)
\end{aligned} \tag{B.3}$$

**Beispiel B.20** *Das Proximum der Funktion $f(x) = x^2$ in der aus Beispiel B.19 bekannten Basis $L$ ist:*

$$\begin{aligned}
\hat{f}(x) &= \int_0^{1/2} x^2\, \tilde{l}_0(x)\,dx \cdot l_0(x) + \int_{1/2}^{1} x^2\, \tilde{l}_1(x)\,dx \cdot l_1(x) \\
&= \frac{5}{6}\, l_0(x) - \frac{1}{6}\, l_1(x)
\end{aligned}$$

*(Abb. B.2, rechts)*

# Anhang C

# Funktionsdarstellungen in hierarchischen Basen

Als Grundlage zur Darstellung von Funktionen in hierarchischen Funktionsbasen soll hier die Wavelet-Darstellung dienen.

Die Wavelet-Transformation ist aus den Bestrebungen einer Weiterentwicklung der Fourier-Transformation entstanden. Während die Fourier-Analyse gute Möglichkeiten zur Untersuchung der Frequenzanteile eines Signals liefert, gibt sie keine Hinweise zur Lokalisierung dieser Anteile. Die Wavelet-Transformation hat sich diesem Ansatz insofern als überlegen erwiesen, als sie einen mathematischen Rahmen liefert, in dem die Transformation eine gleichzeitige Lokalisierung des Signals im Orts- und Frequenzbereich liefert.

Die Wavelet-Transformation und ihre Anwendungen sind ein aktuelles Forschungsgebiet der Mathematik. Weiterführend sei auf [8, 57] verwiesen. Eine lesenswerte Einführung findet sich im Wavelet-Primer [77, 78]. Anwendungen im Bereich der Computergrafik werden in [79] vorgestellt.

## C.1 Multi-Skalen-Analyse

Eine *Multi-Skalen-Analyse* (kurz *MSA*) ermöglicht die Darstellung einer Funktion in verschiedenen Genauigkeitsgraden. Die Funktion kann auf verschiedenen *Skalen* der MSA betrachtet werden.

Die gröbste Skala einer MSA wird als Skala 0 bezeichnet, die folgenden Skalen werden aufsteigend numeriert. Im folgenden Text werden viele Begriffe definiert, die jeweils zu einer Skala $j$ der MSA gehören. Die Zugehörigkeit wird jeweils durch einen hochgestellten Index gekennzeichnet, diese Schreibweise ist nicht mit Potenzierung zu verwechseln.

### C.1.1 Funktionsbasen

Grundlage einer MSA ist eine Folge geschachtelter Funktionsräume:

$$V^0 \subset V^1 \subset V^2 \subset \cdots$$

Die Dimension des Funktionsraumes $V^j$ werde mit $m^j$ bezeichnet. Da die Auflösung der Funktionen von Skala zu Skala feiner wird, gilt:

$$m^0 < m^1 < m^2 < \cdots$$

Die Basisfunktionen $\phi_i^j(x), i = 1, \ldots, m^j$ des Funktionsraums $V^j$ werden als *Skalierungsfunktionen* bezeichnet. Da es später nützlich ist, bestimmte Beziehungen in Matrixschreibweise darzustellen, schreiben wir die Basis $\Phi^j$ von $V^j$ manchmal als Zeilenvektor:

$$\mathbf{\Phi}^j(x) := \begin{bmatrix} \phi_1^j(x) & \cdots & \phi_{m^j}^j(x) \end{bmatrix}$$

Da jeder $V^j$ ein echter Unterraum des nächsthöheren Funktionsraums $V^{j+1}$ ist, kann man nach einer Darstellung für die „Lücke“ zwischen diesen Räumen suchen. Es handelt sich hierbei um die *Wavelet-Räume* $W^j$. Der Funktionsraum $W^j$ ist das *Komplement* von $V^j$ in $V^{j+1}$. Das bedeutet, jede Funktion in $V^{j+1}$ kann als Summe einer Funktion aus $V^j$ und einer Funktion aus $W^j$ geschrieben werden. Die Dimension $n^j$ des Wavelet-Raums $W^j$ ist dann natürlich die Differenz der Dimension zweier aufeinanderfolgender Skalierungs-Räume:

$$n^j = m^{j+1} - m^j$$

Die Basisfunktionen $\psi_i^j(x), i = 1, \ldots, n^j$ des Wavelet-Raums $W^j$ werden *Waveletfunktionen* oder kurz *Wavelets* genannt. Auch hier ist es nützlich, die Basis $\Psi^j$ manchmal als Zeilenvenktor darzustellen:

$$\mathbf{\Psi}^j(x) := \begin{bmatrix} \psi_1^j(x) & \cdots & \psi_{n^j}^j(x) \end{bmatrix}$$

### C.1.2 Beziehungen zwischen Basen

Wenn die Basen der Funktionsräume als Zeilenvektoren geschrieben werden, lassen sich bestimmte Beziehungen durch Matrizen darstellen:

- Da die Skalierungs-Räume eine Folge ineinander geschachtelter Funktionsräume sind, läßt sich jede Basisfunktion eines Raumes $V^{j-1}$ als Linearkombination von Basisfunktionen der nächsthöheren Skala $V^j$ darstellen. Das bedeutet, daß es eine Matrix $\mathbf{P}^j$ gibt, so daß

  $$\mathbf{\Phi}^{j-1}(x) = \mathbf{\Phi}^j(x)\,\mathbf{P}^j \tag{C.1}$$

  gilt. $\mathbf{P}^j$ ist eine $m^j \times m^{j-1}$-Matrix.

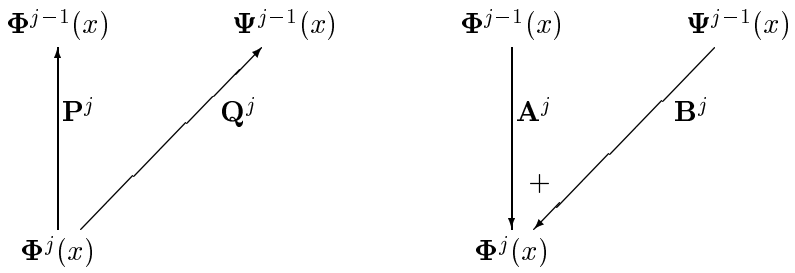



Abbildung C.1: Beziehungen zwischen Funktionsbasen einer MSA

- $W^{j-1}$ ist ein Unterraum von $V^j$, also kann jede Basisfunktion von $W^{j-1}$ als Linearkombination der Basisfunktionen von $V^j$ dargestellt werden. Das bedeutet, es gibt eine Matrix $\mathbf{Q}^j$, für die

$$\mathbf{\Psi}^{j-1}(x) = \mathbf{\Phi}^j(x)\,\mathbf{Q}^j \tag{C.2}$$

  gilt. $\mathbf{Q}^j$ ist eine $m^j \times n^{j-1}$-Matrix.

- Da $V^{j-1}$ und $W^{j-1}$ gemeinsam den Raum $V^j$ aufspannen, läßt sich jede Basisfunktion von $V^j$ als Linearkombination der Basisfunktionen von $V^{j-1}$ und $W^{j-1}$ darstellen. Es gibt also Matrizen $\mathbf{A}^j$ und $\mathbf{B}^j$, so daß

$$\mathbf{\Psi}^j(x) = \mathbf{\Phi}^{j-1}(x)\,\mathbf{A}^j + \mathbf{\Psi}^{j-1}(x)\,\mathbf{B}^j \tag{C.3}$$

  gilt. $\mathbf{A}^j$ ist eine $m^{j-1} \times m^j$-Matrix, $\mathbf{B}^j$ eine $n^{j-1} \times m^j$-Matrix.

Die durch die jeweiligen Matrizen ausgedrückten Beziehungen der Funktionsbasen zueinander sind in der Abbildung C.1 veranschaulicht.

Die Matrizen $\mathbf{A}^j$, $\mathbf{B}^j$, $\mathbf{P}^j$, $\mathbf{Q}^j$ einer Skala $j$ erfüllen die folgende Relation:

$$\left[\frac{\mathbf{A}^j}{\mathbf{B}^j}\right] = \left[\mathbf{P}^j|\mathbf{Q}^j\right]^{-1} \tag{C.4}$$

Diese Darstellung benutzt Block-Matrix-Notation: die Matrizen werden über- bzw. nebeneinander in eine größere Matrix geschrieben.

### C.1.3 Analyse

Zu einer gegebenen Funktion $f(x)$ gibt es auf jeder Skala der Hierarchie eine Approximation $\hat{f}^j(x)$. Natürlich läßt sich $\hat{f}^j(x)$ in der Basis dieser Skala, also in den Skalierungsfunktionen darstellen:

$$\hat{f}^j(x) = \sum_{i=1}^{m^j} c_i^j\,\phi_i^j(x)$$

Die Koeffizienten dieser Basisdarstellung lassen sich als Spaltenvektor $\mathbf{c}^j$ schreiben:

$$\mathbf{c}^j = \begin{bmatrix} c_1^j \\ \vdots \\ c_{m^j}^j \end{bmatrix}$$

Damit läßt sich die Basisdarstellung auch kurz als

$$\hat{f}^j(x) = \mathbf{\Phi}^j(x)\,\mathbf{c}^j$$

schreiben.

Die Koeffizienten $\mathbf{c}^{j-1}$ der Approximation $\hat{f}^{j-1}(x)$ auf der gröberen Skala $j-1$ kann man durch eine Matrix-Multiplikation

$$\mathbf{c}^{j-1} = \mathbf{A}^j\,\mathbf{c}^j$$

erhalten.

Da $\Phi^{j-1}$ weniger Basisfunktionen enthält als $\Phi^j$, geht ein gewisser Anteil von Detail-Information verloren. Da $\Phi^{j-1}$ zusammen mit $\Psi^{j-1}$ eine Basis des Raums $V^j$ bildet, kann dieser verlorene Anteil in der Wavelet-Basis der Skala $j-1$ dargestellt werden. Die Koeffizienten dieses Anteils werden als Spaltenvektor $\mathbf{d}^{j-1}$ geschrieben:

$$\mathbf{d}^{j-1} = \begin{bmatrix} d_1^{j-1} \\ \vdots \\ d_{n^j}^{j-1} \end{bmatrix}$$

Diese Koeffizienten werden mit einer weiteren Matrix-Multiplikation berechnet:

$$\mathbf{d}^{j-1} = \mathbf{B}^j\,\mathbf{c}^j$$

Zusammen ergibt dies eine Darstellung der Funktion $\hat{f}^j(x)$ in den Skalierungs- und Waveletfunktionen der Skala $j-1$:

$$\begin{aligned} \hat{f}^j(x) &= \sum_{i=1}^{m^{j-1}} c_i^{j-1}\,\phi_i^{j-1}(x) + \sum_{i=1}^{n^{j-1}} d_i^{j-1}\,\psi_i^{j-1}(x) \\ &= \mathbf{c}^{j-1}\,\mathbf{\Phi}^{j-1}(x) + \mathbf{d}^{j-1}\,\mathbf{\Psi}^{j-1}(x) \end{aligned}$$

Der Prozeß des Aufspaltens der Koeffizienten $\mathbf{c}^j$ in eine gröbere Version $\mathbf{c}^{j-1}$ und einen Detail-Anteil $\mathbf{d}^{j-1}$ wird *Analyse* oder *Dekomposition* genannt. Die Matrizen $\mathbf{A}^j$ und $\mathbf{B}^j$ werden deshalb auch *Analyse-Filter* genannt.

Beginnt man die Analyse auf einer Skala $j$ und wiederholt diesen Vorgang, bis man zur Skala 0 gelangt, erhält man die *Wavelet-Transformation* der Funktion $\hat{f}^j$(x):

$$\begin{aligned} \hat{f}^j(x) &= \sum_{i=1}^{m^0} c_i^0\,\phi_i^0(x) + \sum_{k=0}^{j-1}\sum_{i=1}^{n^k} d_i^k\,\psi_i^k(x) \\ &= \mathbf{c}^0\,\mathbf{\Phi}^0(x) + \sum_{k=0}^{j-1} \mathbf{d}^k\,\mathbf{\Psi}^k(x) \end{aligned}$$

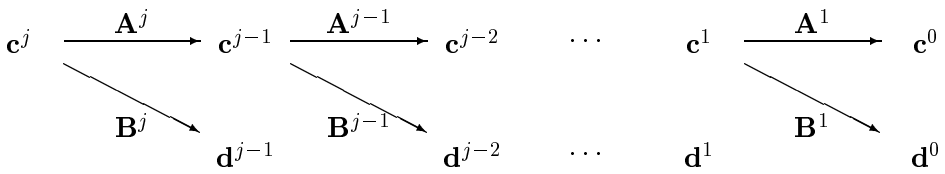



Abbildung C.2: Analyse

Der Prozeß der Transformation von $\mathbf{c}^j$ zu $\mathbf{c}^0$, $\mathbf{d}^0$, $\mathbf{d}^1$, ..., $\mathbf{d}^{j-1}$ wird in Abb. C.2 veranschaulicht.

Die Wavelet-Transformation ist mathematisch gesehen ein Basiswechsel, sie ist verlustfrei und läßt sich, wie im nächsten Abschnitt dargestellt, umkehren. Der durch $\mathbf{c}^0$ dargestellte Anteil stellt ein Grundsignal dar, zu dem mit feineren Skalen $j$ zunehmend höhere Frequenzanteile addiert werden. Die Koeffizienten $d_i^j$ repräsentieren nicht nur mit größerem $j$ höhere Frequenzen, zusätzlich liefert der Index $i$ Informationen über den Ort des Anteils. Hieraus resultiert die oben bereits genannte Eigenschaft der Wavelet-Transformation, eine gleichzeitige Lokalisierung des Signals im Orts- und Frequenzbereich des Signals zu liefern.

Da die meisten Signale nicht alle Frequenzen an allen Orten aufweisen, sind viele der Koeffizienten $d_i^j$ Null oder nahe an Null. Auf dieser Grundlage arbeiten Wavelet-basierte Verfahren zur Datenkompression, indem sie kleine Koeffizienten weglassen.

### C.1.4 Rekonstruktion

Die Koeffizienten $\mathbf{c}^j$ können aus den Koeffizienten $\mathbf{c}^{j-1}$, $\mathbf{d}^{j-1}$ der gröberen Skala $j-1$ unter Benutzung der Matrizen $\mathbf{P}^j$ und $\mathbf{Q}^j$ berechnet werden:

$$\mathbf{c}^j = \mathbf{P}^j\,\mathbf{c}^{j-1} + \mathbf{Q}^j\,\mathbf{d}^{j-1}$$

Da man hier davon sprechen kann, daß das feinere Signal aus der gröberen Darstellung rekonstruiert wird, wird dieser Vorgang *Rekonstruktion* oder auch *Synthese* genannt. $\mathbf{P}^j$ und $\mathbf{Q}^j$ werden entsprechend als *Synthese-Filter* bezeichnet.

Durch wiederholte Anwendung des Rekonstruktionsschrittes kann aus den Koeffizienten $\mathbf{c}^0$, $\mathbf{d}^0$, $\mathbf{d}^1$, ..., $\mathbf{d}^{j-1}$ der Wavelet-Transformation die Darstellung $\mathbf{c}^j$ rekonstruiert werden. Diese Umkehrung der Wavelet-Transformation wird in Abb. C.3 veranschaulicht.

## C.2 Entwurf einer Multi-Skalen-Analyse

Jede Multi-Skalen-Analyse, die die oben beschriebenen Eigenschaften besitzt, kann als Grundlage zur Wavelet-Darstellung von Funktionen dienen. Man hat

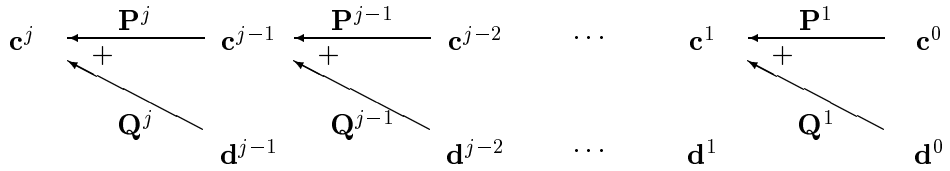



Abbildung C.3: Rekonstruktion

dabei gewisse Freiheitsgrade, so daß MSAs entworfen werden können, die für die jeweilige Anwendung nützliche Eigenschaften haben.

Zum Entwurf einer MSA gehören drei Schritte:

**Wahl der Skalierungs-Basen** $\Phi^j$ Dadurch werden die Funktionsräume $V^j$ und die Synthese-Filter $\mathbf{P}^j$ festgelegt. Außerdem hat die Wahl dieser Basen Einfluß auf die Approximation der Funktion auf den verschiedenen Skalen der MSA.

**Wahl eines inneren Produkts** Diese Wahl legt die Standardnorm und den Begriff der Orthogonalität fest. Unter Verwendung der Standardnorm wird hierdurch auch das Fehlermaß festgelegt, nach dem die Approximationen auf den verschiedenen Skalen der MSA bestimmt werden. Manchmal ist es sinnvoll, das innere Produkt einem anwendungsbezogenen Fehlermaß anzupassen.

**Wahl der Wavelet-Basen** $\Psi^j$ Durch diese Wahl werden die Synthese-Filter $\mathbf{Q}^j$ und dadurch indirekt auch die Analyse-Filter $\mathbf{A}^j$ und $\mathbf{B}^j$ festgelegt.

### C.2.1 Orthonormale Wavelets

In einer orthonormalen Basis sind alle Basisvektoren normiert und paarweise senkrecht zueinander. In mathematischer Notation lassen sich diese Bedingungen als

$$\begin{aligned}
\langle \phi_k^j, \phi_l^j \rangle &= \delta_{k,l} \qquad \text{für} \quad k = 1, \ldots, m^j \quad l = 1, \ldots, m^j \\
\langle \psi_k^j, \psi_l^j \rangle &= \delta_{k,l} \qquad \text{für} \quad k = 1, \ldots, n^j \quad l = 1, \ldots, n^j \\
\langle \phi_k^j, \psi_l^j \rangle &= 0 \qquad\quad \text{für} \quad k = 1, \ldots, m^j \quad l = 1, \ldots, n^j
\end{aligned}$$

darstellen.

Die angenehmste Eigenschaft einer orthonormalen Wavelet-Konstruktion ist, daß sie eine einfache Berechnung des Approximationsfehlers ermöglicht, der sich bei Weglassen einzelner Koeffizienten ergibt. Dadurch sind effiziente Verfahren zur Kompression eines Signals mit vorgegebener Fehlerschranke möglich: Die Koeffizienten werden in aufsteigender Reihenfolge ihres Absolutwertes sortiert und dann von vorne beginnend gestrichen, bis zum ersten Mal das Fehlermaß überschritten wird.

Für weitere Informationen zu orthonormalen Wavelet-Konstruktionen sei insbesondere auf die Arbeiten von I. Daubechies verwiesen [17, 11].

#### C.2.1.1 Haar-MSA

Abb. C.4 zeigt als Beispiel einer orthonormalen MSA die Basisfunktionen der ersten Skalen der Haar-MSA, die stückweise konstante Basisfunktionen benutzt.

### C.2.2 Semiorthogonale Wavelets

Es sind nicht viele Basen zur Erzeugung orthonormaler MSAs bekannt. Außerdem sind oftmals andere Eigenschaften der Basisfunktionen erwünscht, die sich in orthonormalen Konstruktionen nicht erreichen lassen:

**kompakter Träger** Die gegenseitige Überlappung der Träger der Basisfunktionen bestimmt, wie dünn die Besetzung der Filtermatrizen ist und damit, welchen Aufwand Analyse und Synthese erfordern. Je kompakter die Träger der Basisfunktionen, desto dünner besetzt sind die Matrizen.

**Glattheit** Je glatter die Basisfunktionen sind, desto besser lassen sich glatte Approximationen erreichen, diese Anforderung steht im gewissen Gegensatz zur erstgenannten, da sie in der Regel zu einem breiteren Träger führt.

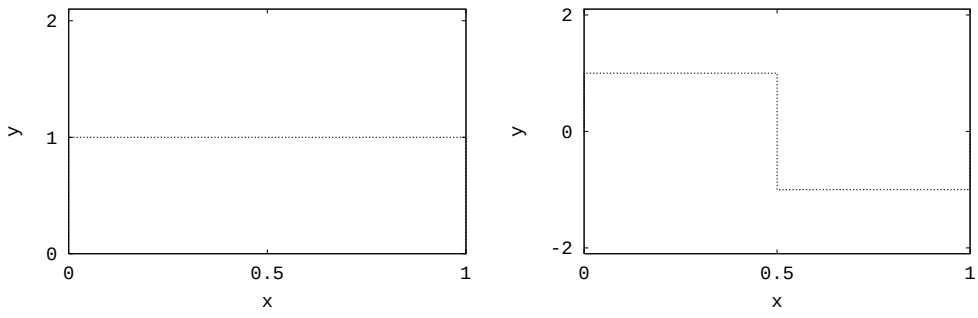
**Symmetrie** Für manche Anwendungen sind bezüglich ihrer Mitte symmetrische bzw. antisymmetrische Waveletfunktionen erwünscht, um Verzerrungen zu vermeiden.

**Verschwindende Momente** Ein Wavelet $\psi(x)$ hat $n$ verschwindende Momente, wenn $\int \psi(x)\, x^k \,\mathrm{d}x$ identisch Null ist für $k = 0, \ldots, n-1$ aber nicht für $k = n$. Verwendung solcher Wavelets führt bei bestimmten Funktionen zu dünn besetzten Koeffizientenvektoren.

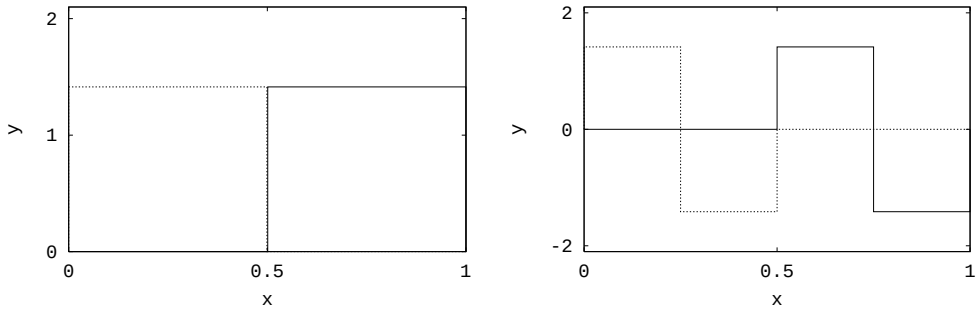
Wird die einschränkende Bedingung der Orthonormalität fallengelassen, lassen sich die obigen Eigenschaften leichter erreichen. Man spricht von einer semiorthogonalen Konstruktion, wenn die Waveletfunktionen senkrecht zu den Skalierungsfunktionen derselben Skala sind, es wird aber nicht mehr gefordert, daß Skalierungsfunktionen bzw. Wavelets einer Skala senkrecht zueinander stehen. Zu erfüllen ist also die Bedingung:

$$\langle \phi_k^j, \psi_l^j \rangle \quad = \quad 0 \qquad \text{für} \quad k = 1, \ldots, m^j \quad l = 1, \ldots, n^j$$

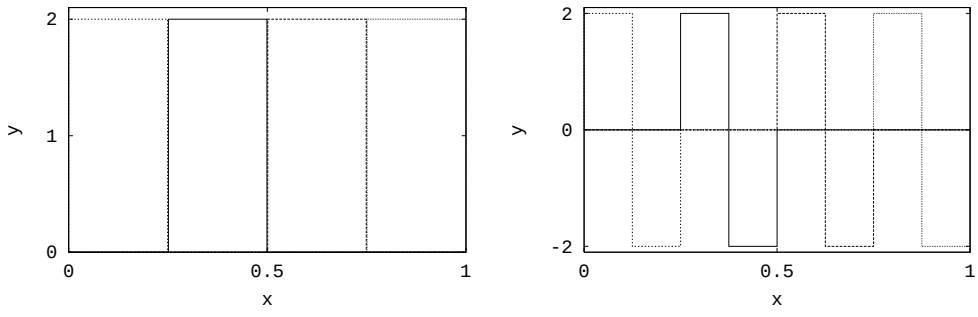
Orthonormale Wavelets sind offensichtlich ein Spezialfall der semiorthogonalen Konstruktion. Das Verfahren zur effizienten Kompression einer Basisdarstellung läßt sich auf semiorthogonale MSAs nicht mehr anwenden. Ein Vorteil semiorthogonaler MSAs ist, daß die Repräsentation eines Signals auf Skala $i < j$ eine $L^2$-Approximation der Darstellung auf Skala $j$ ist.




Skala 0




Skala 1




Skala 2

Abbildung C.4: Haar-MSA, links: Skalierungsfunktionen, rechts: Wavelets

### C.2.2.1 B-Spline-MSA

Als Beispiele für semiorthogonale Wavelet-Konstruktionen zeigen die Abb. C.5 und C.6 die Basisfunktionen der ersten Skalen einer linearen bzw. quadratischen B-Spline-MSA. Als Skalierungsfunktionen werden die B-Splines auf dem Intervall $[0, 1]$ mit äquidistantem Knotenvektor und mehrfachen Endknoten verwendet. Die Wavelets sind so konstruiert, daß sie minimalen Träger haben [10, 9].

### C.2.3 Biorthogonale Wavelets

Ein Nachteil semiorthogonaler Konstruktionen ist, daß die Analyse-Filter im Gegensatz zu den Synthese-Filtern oftmals nicht dünn besetzt sind.

Die biorthogonale Wavelet-Konstruktion läßt auch die Bedingung der Orthogonalität von Skalierungsfunktionen und Wavelets einer Skala fallen, führt aber folgende Bedingungen für die dualen Basen ein:

$$\begin{aligned} \langle \phi_k^j, \tilde{\psi}_l^j \rangle &= 0 \qquad \text{für} \quad k = 1, \ldots, m^j \quad l = 1, \ldots, n^j \\ \langle \psi_k^j, \tilde{\phi}_l^j \rangle &= 0 \qquad \text{für} \quad k = 1, \ldots, n^j \quad l = 1, \ldots, m^j \end{aligned}$$

Mit biorthogonalen Konstruktionen lassen sich zwar dünn besetzte Filter sowohl für Analyse als auch Synthese erreichen, es sei aber darauf hingewiesen, daß die Repräsentationen des Signals zwischen verschiedenen Skalen keine $L^2$-Approximationen mehr sind, weshalb Anwendungen, die ein Signal mit verschiedenem Detaillierungsgrad betrachten, auf eine biorthogonale MSA nicht unbedingt anwendbar sind.

## C.3 Höherdimensionale Wavelet-Transformationen

Höherdimensionale Wavelet-Konstruktionen werden meistens als Tensorprodukt-Ansätze verwirklicht.

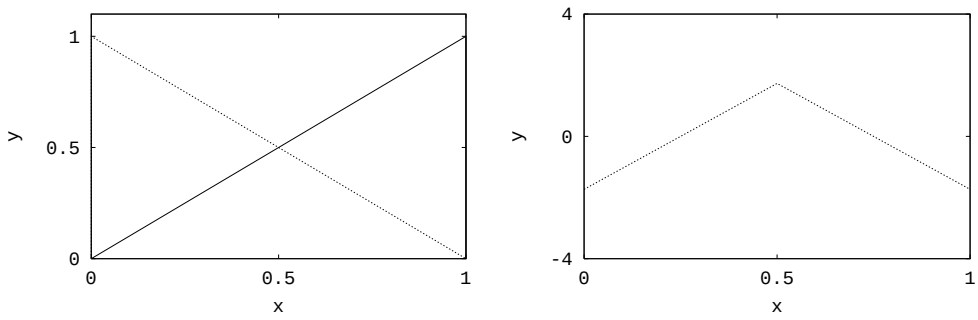
### C.3.1 Zweidimensionale Wavelet-Transformationen

Im zweidimensionalen Anwendungsfall ist die Transformation einer Funktion $f(u, v)$ zu bestimmen. Für die im folgenden vorgestellten Tensorprodukt-Konstruktionen sei angenommen, daß in $u$- und $v$-Richtung dieselbe MSA verwendet werde, prinzipiell ist es natürlich möglich, verschiedene MSAs zu verwenden, z. B. in $u$-Richtung eine Haar-MSA und in $v$-Richtung eine quadratische B-Spline-MSA.[1]

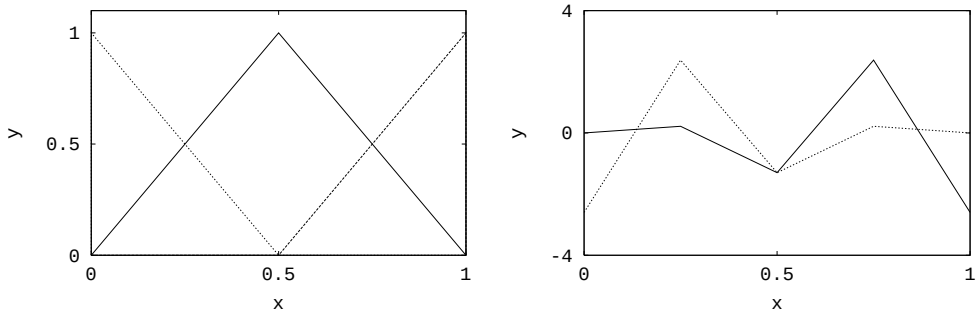
Beide hier vorgestellten Konstruktionen beginnen mit einer Tensorprodukt-Darstellung der Approximation $\hat{f}(u, v)$ auf Skala $j$:

$$\mathbf{C}^j = \sum_{k=1}^{m^j} \sum_{l=1}^{m^j} c_{k,l}^j \, \phi_k^j(u) \, \phi_l^j(v)$$

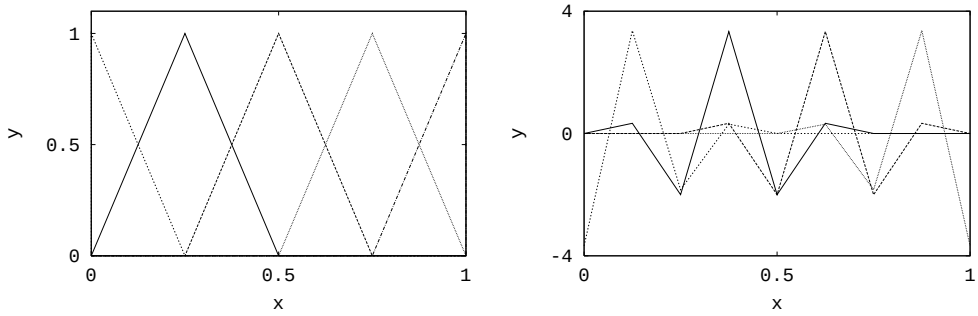
[1] Die in Kapitel 5 vorgestellte `WaveFunc`-Bibliothek unterstützt tatsächlich die Verwendung unterschiedlicher MSAs für $u$- und $v$-Richtung.




Skala 0

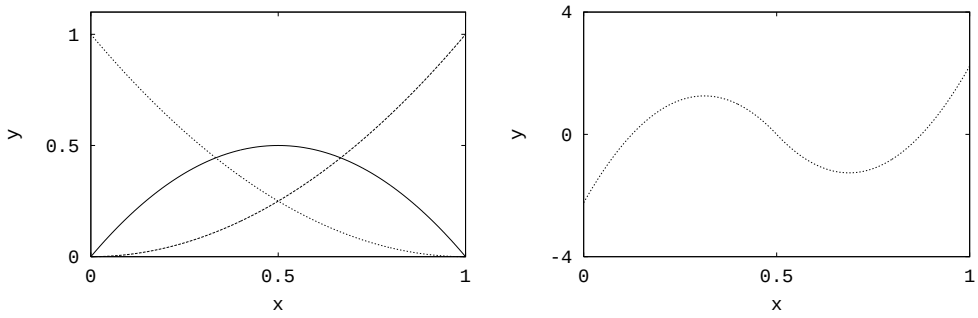



Skala 1

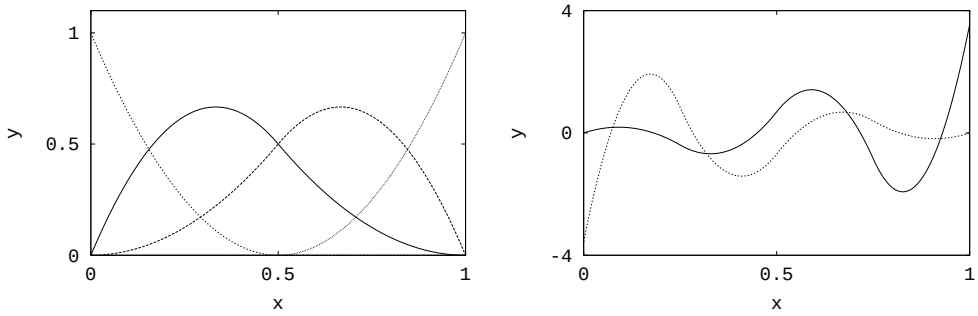



Skala 2

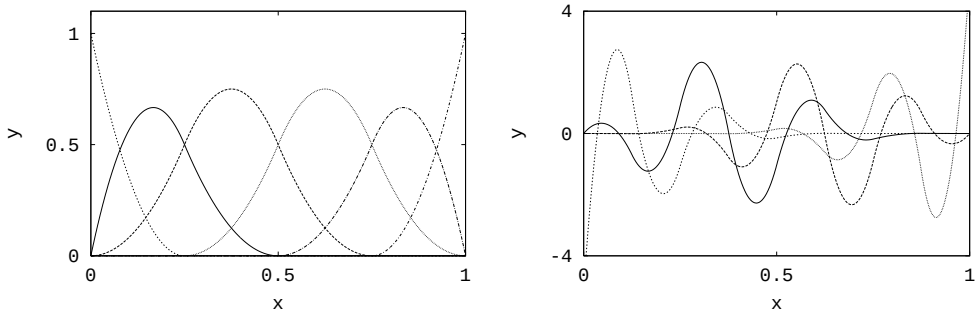
Abbildung C.5: lineare B-Spline-MSA, links: Skalierungsfunktionen, rechts: Wavelets




Skala 0




Skala 1




Skala 2

Abbildung C.6: quadratische B-Spline-MSA, links: Skalierungsfunktionen, rechts: Wavelets

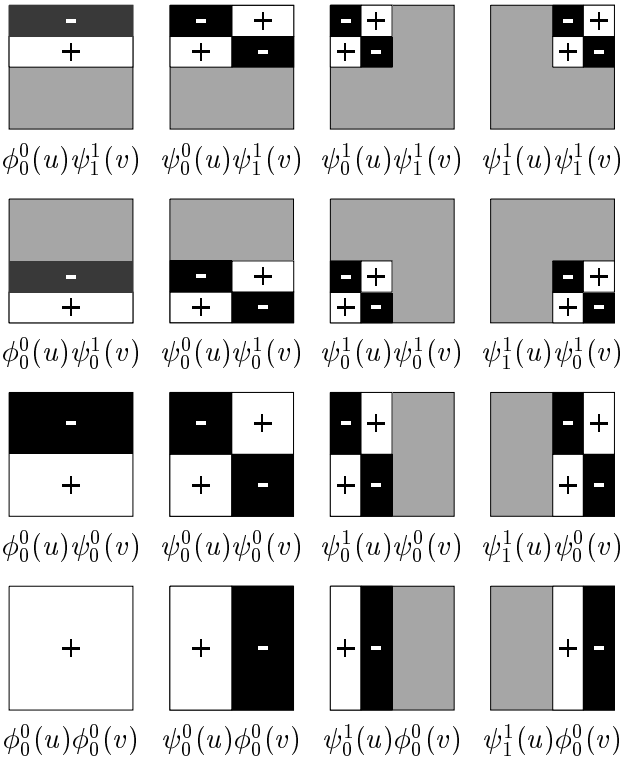



Abbildung C.7: Basisfunktionen der Standard-Konstruktion für Haar-MSA

Die Matrix $\mathbf{C}^j$ ist die $m^j \times m^j$-Koeffizientenmatrix dieser Tensorprodukt-Darstellung.

#### C.3.1.1 Standard-Konstruktion

Die Standard-Konstruktion wendet auf jede Zeile der Matrix $\mathbf{C}^j$ eine eindimensionale Wavelet-Transformation an. Die daraus resultierenden Koeffizienten werden zeilenweise in eine neue Matrix geschrieben, auf die dann spaltenweise jeweils wieder eine eindimensionale Wavelet-Transformation angewendet wird. Die Basis der so generierten Wavelet-Transformation erhält man als die Tensorprodukte aller Paare von Basisfunktionen der Basen $\Phi^0(u)$, $\Psi^0(u)$, ..., $\Psi^{j-1}(u)$ mit den Basisfunktionen aus den Basen $\Phi^0(v)$, $\Psi^0(v)$, ..., $\Psi^{j-1}(v)$, das heißt, die Basis ist die echte Tensorprodukt-Basis zur eindimensionalen Wavelet-Darstellung. Deshalb wird diese Konstruktion auch als Standard-Konstruktion bezeichnet. Für manche Anwendungen ist ein Nachteil dieses Ansatzes, daß viele Tensorprodukt-Basisfunktionen eine langen, schmalen Träger im zweidimensionalen Definitionsbereich aufweisen. Abb. C.7 zeigt die aus dieser Konstruktion resultierenden Basisfunktionen bei Benutzung der Haar-MSA.

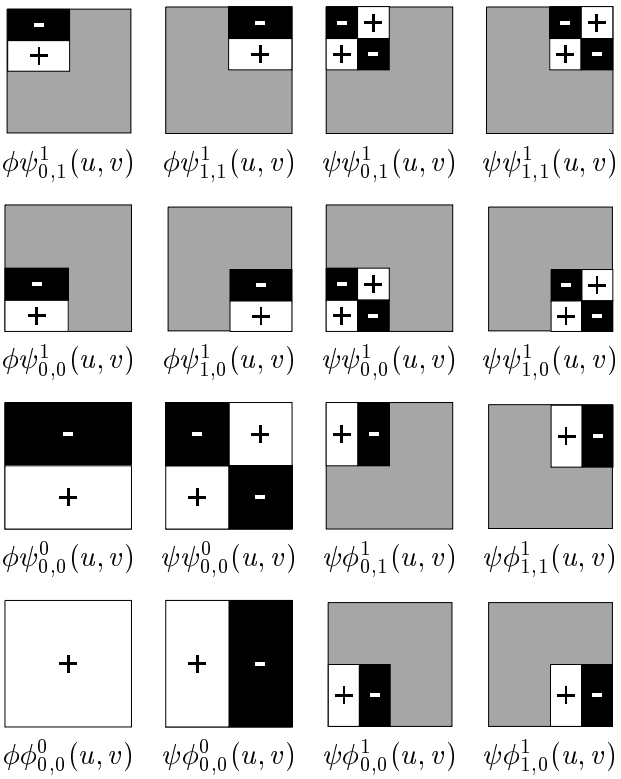



Abbildung C.8: Basisfunktionen der Non-Standard-Konstruktion für Haar-MSA

#### C.3.1.2 Non-Standard-Konstruktion

Die Non-Standard-Konstruktion wendet auf jede Zeile der Matrix $\mathbf{C}^j$ einen Analyse-Schritt der eindimensionale Wavelet-Transformation an. Die daraus resultierenden Koeffizienten werden wiederum zeilenweise in eine neue Matrix geschrieben, auf die dann spaltenweise jeweils wieder ein Analyse-Schritt angewendet wird. Die hieraus resultierende Ergebnismatrix enthält als Untermatrix die Koeffizienten $\mathbf{C}^{j-1}$ der Approximation $\hat{f}^{j-1}(u,v)$ in der gröberen Skala $j-1$. Auf diese Untermatrix wird das Verfahren rekursiv bis zur Skala 0 fortgesetzt. Die aus diesem Ansatz resultierenden Basisfuntktionen bei Verwendung der Haar-MSA sind in Abb. C.8 dargestellt.

### C.3.2 Verallgemeinerung auf höhere Dimensionen

Wie man sieht, basieren beide Konstruktionen auf Anwendungen der eindimensionalen Wavelet-Transformation entlang einzelner Achsenrichtungen. Sowohl der Standard- als auch der Non-Standard-Ansatz lassen sich deshalb relativ einfach auf höhere Dimensionen verallgemeinern. Normalerweise ist man dabei nicht an der expliziten Konstruktion der Basisfunktionen interessiert, sondern an Verfahren zur Analyse und Rekonstruktion höherdimensionaler Funktionsdarstellungen. Diese Operationen können durch wiederholte Anwendung

der Analyse- und Syntheseschritte für eindimensionale Funktionsdarstellungen realisiert werden. Die Speicherung der Koeffizienten der höherdimensionalen Funktionsdarstellungen muß dafür in Datenstrukturen erfolgen, die Zugriff auf eindimensionale Teilvektoren entlang unterschiedlicher Koordinatenrichtungen ermöglichen.

# Anhang D

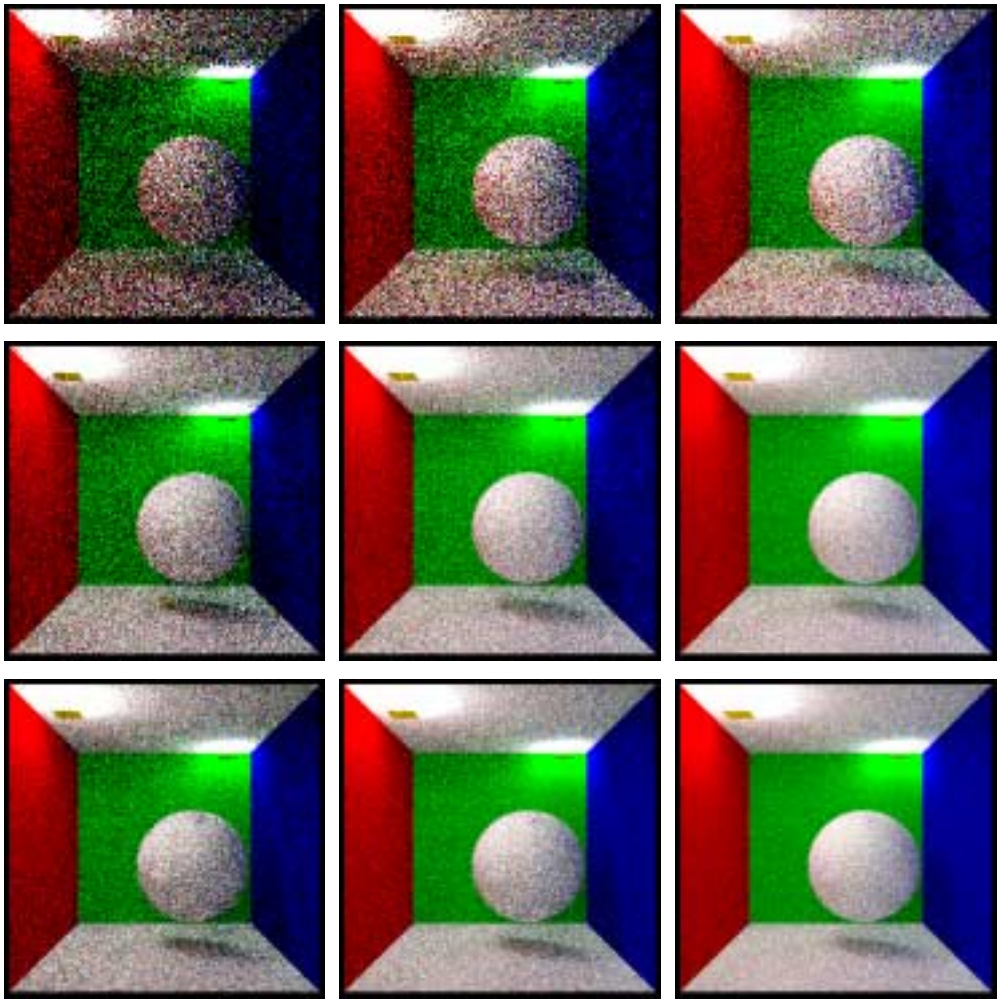
# Beispiel für fotorealistische Bilderzeugung

Abbildung D.1: Mit Radiosity-Verfahren erzeugte Ansicht einer Innenraum-Szene [66]

# Anhang E

# Ergebnisse der Monte Carlo-Bilderzeugung

## E.1 Bewertung der Ergebnisse



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.1: Bildqualität, Szene 2, ohne Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA mit Tiefe 4, Patch-Unterteilung $4 \cdot 4$, Haar-MSA mit Tiefe 5, Patch-Unterteilung $8 \cdot 8$

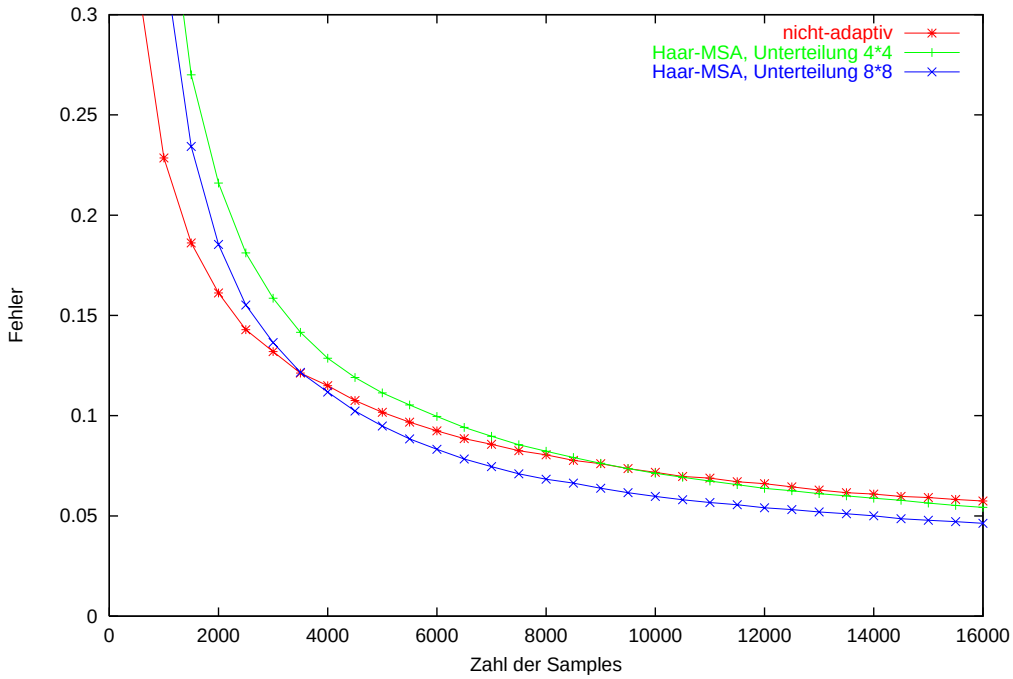



Abbildung E.2: Fehler in $L^2$-Norm

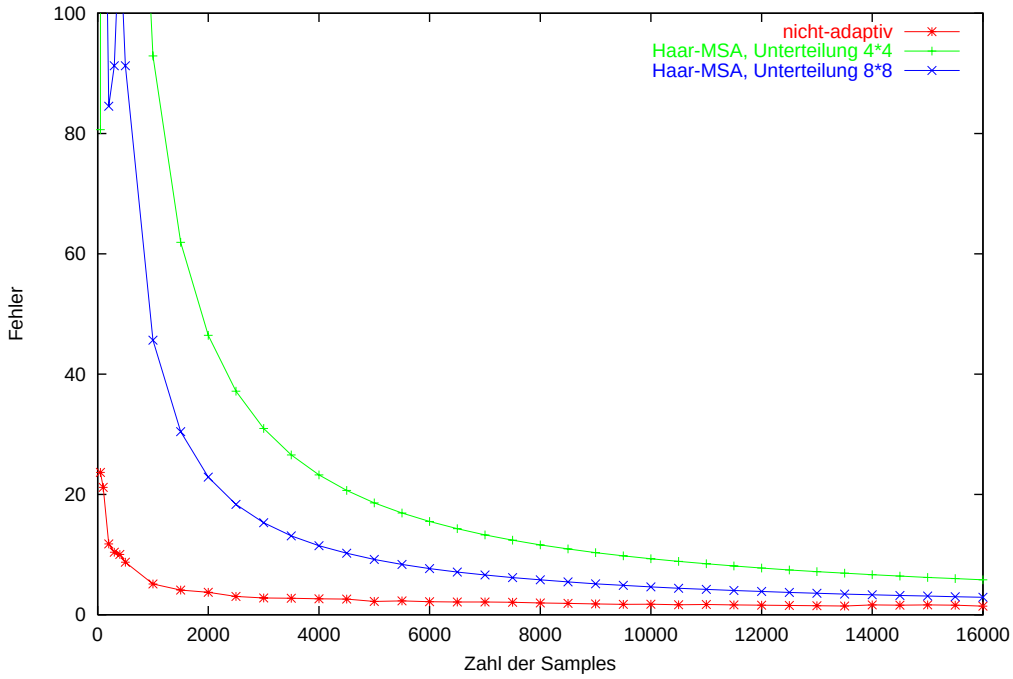



Abbildung E.3: Fehler in $L^\infty$-Norm

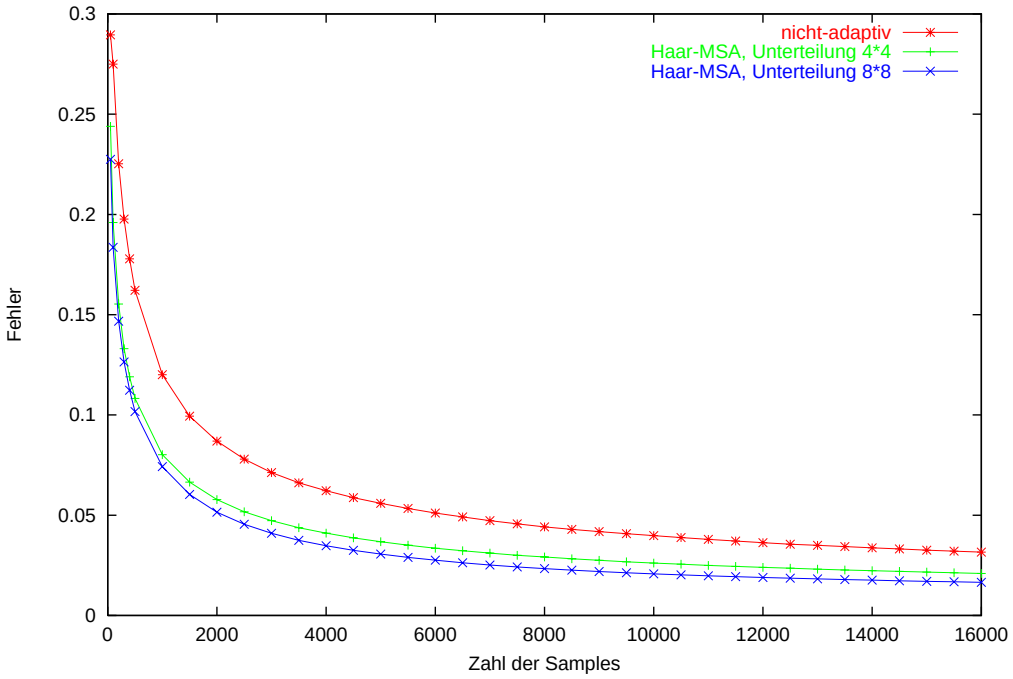



Abbildung E.4: Fehler in modifizierter $L^2$-Norm

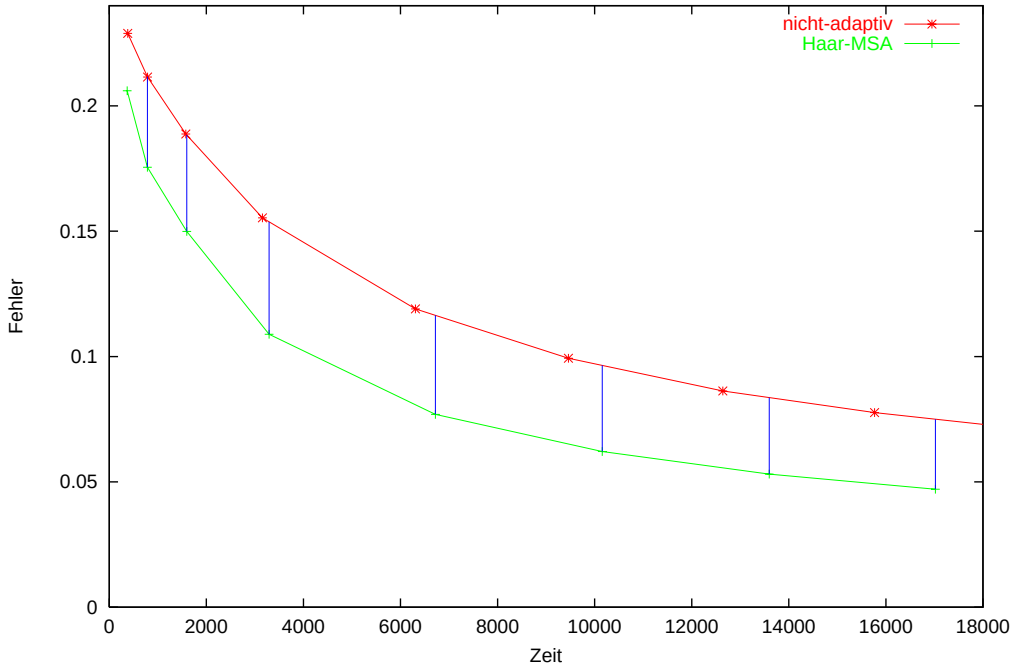



Abbildung E.5: Verringerung des Fehlers bei gleicher Rechenzeit

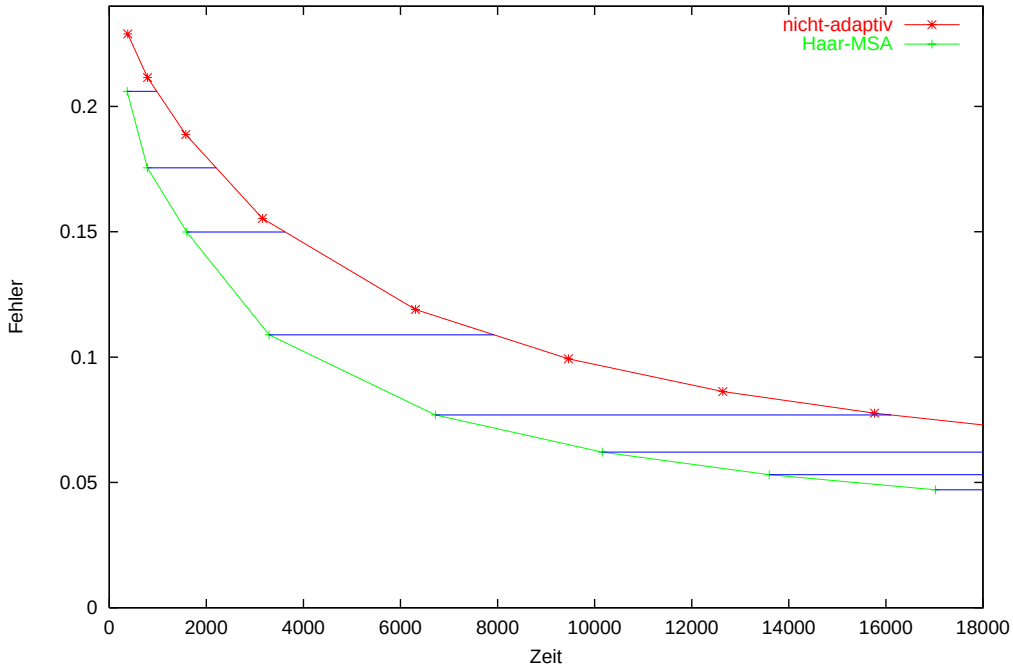



Abbildung E.6: Verringerung der Rechenzeit bei gleichem Fehler

## E.2 Szene 1

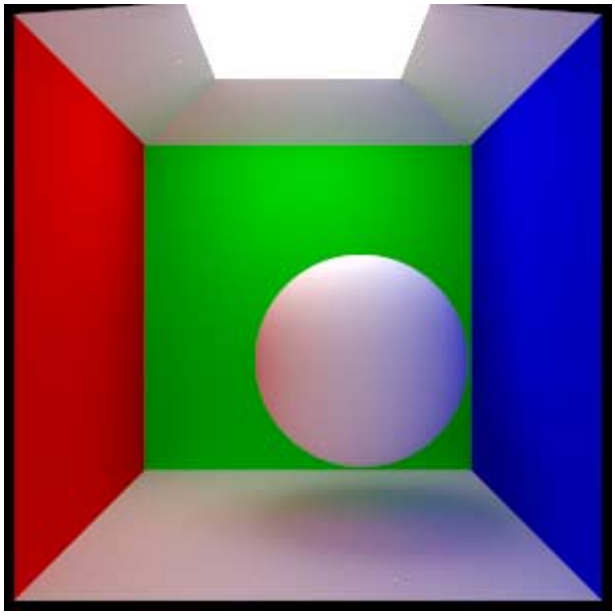


Abbildung E.7: Szene 1, Referenzlösung (nicht-adaptiv, mit Stratifikation, 64000 Samples pro Pixel)

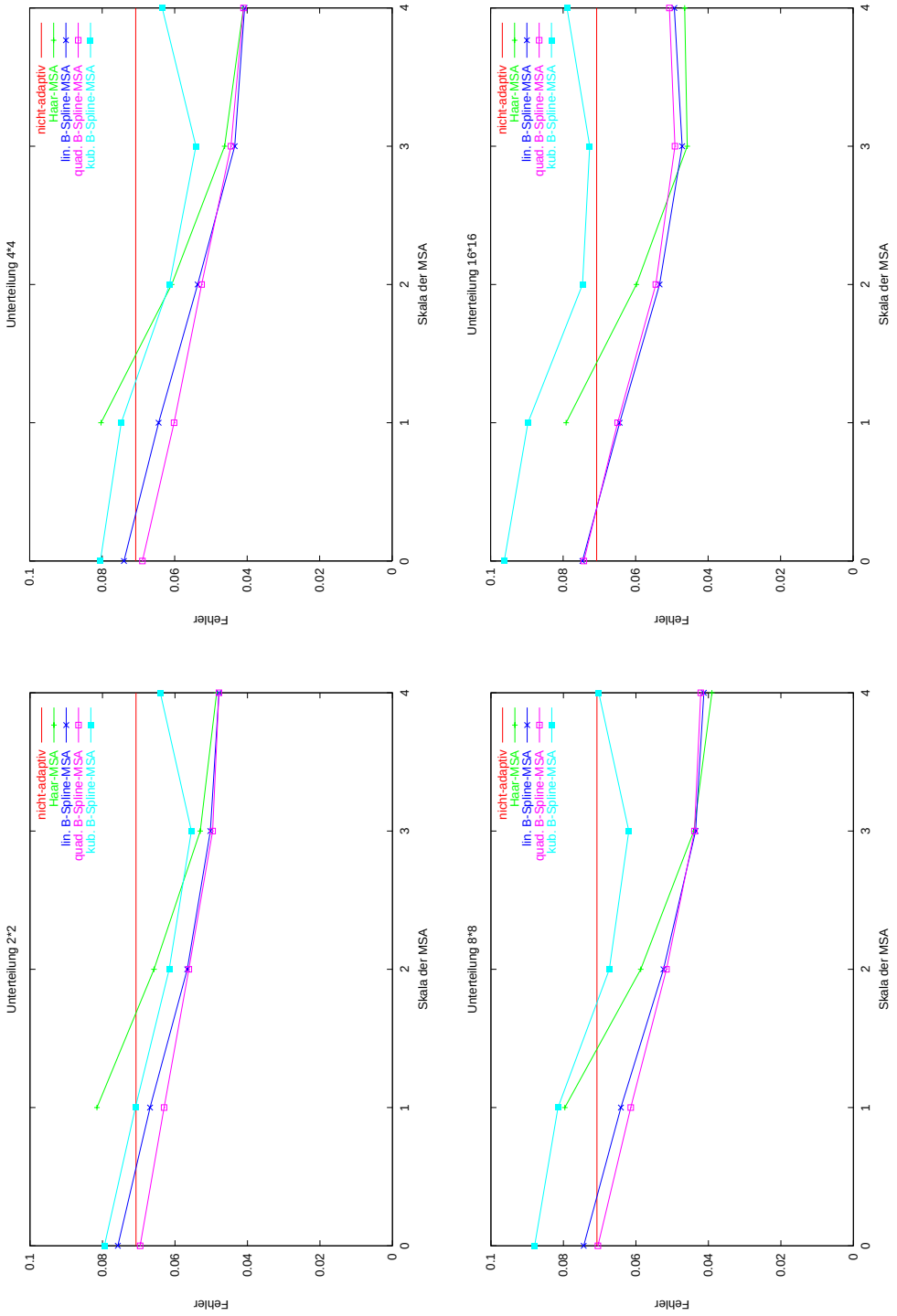



Abbildung E.8: Maximal erreichbare Qualität, Szene 1, ohne Stratifikation

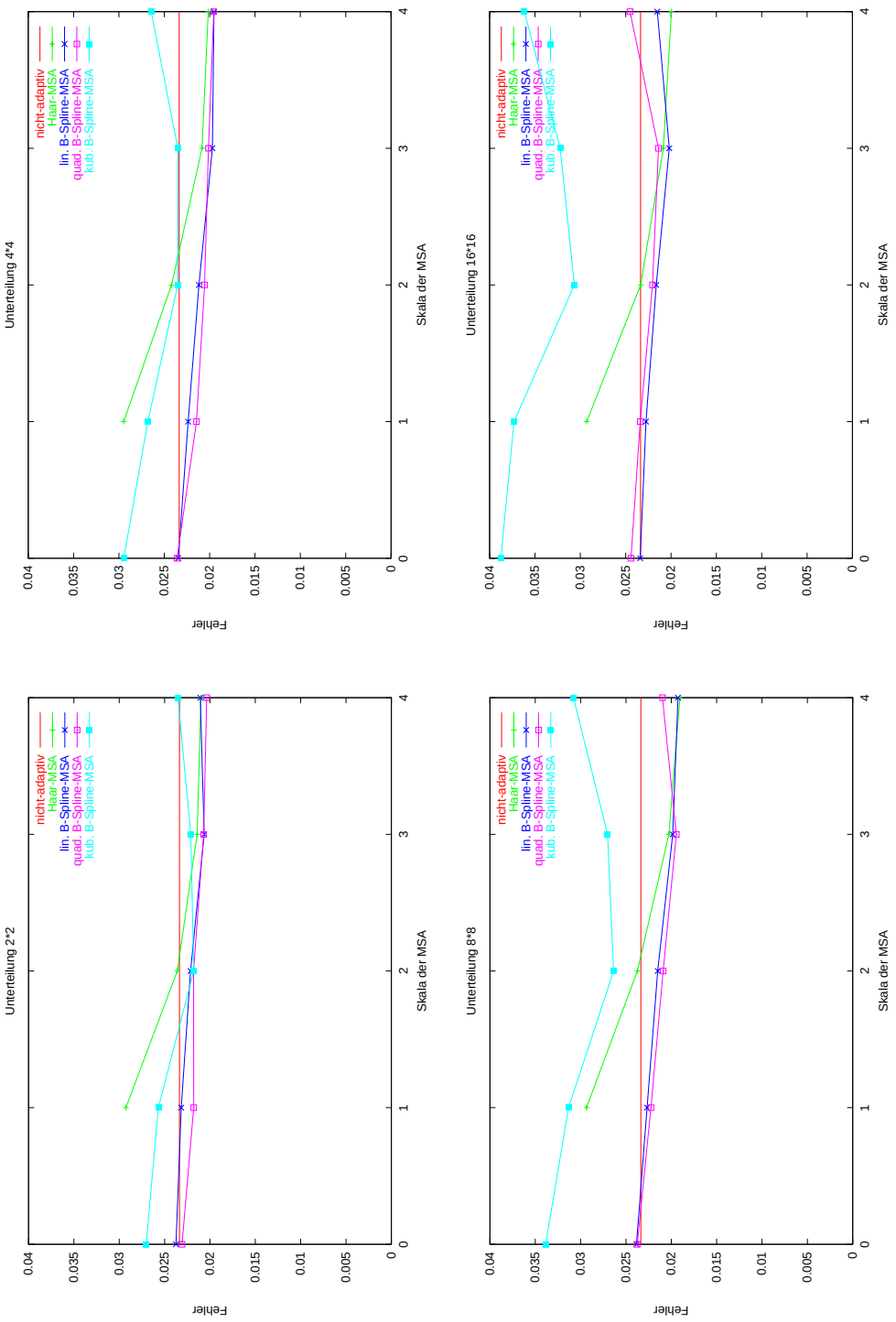
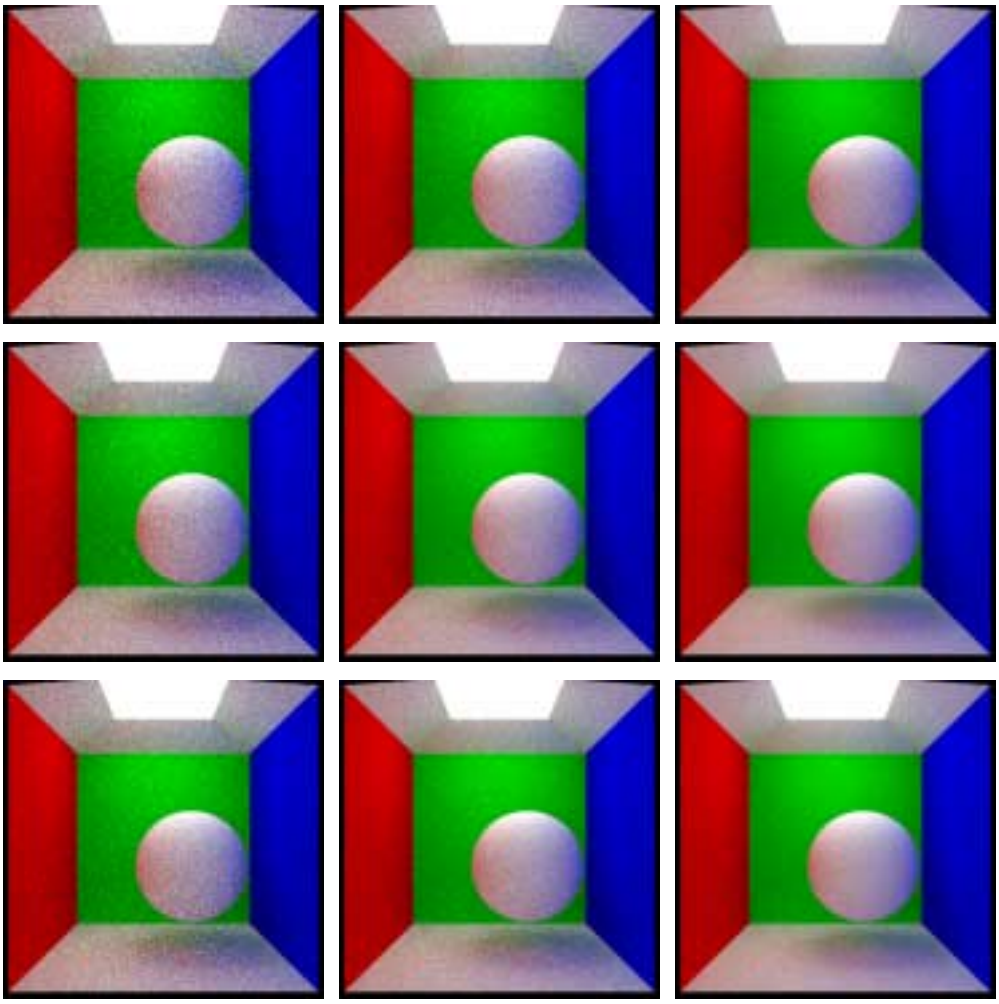
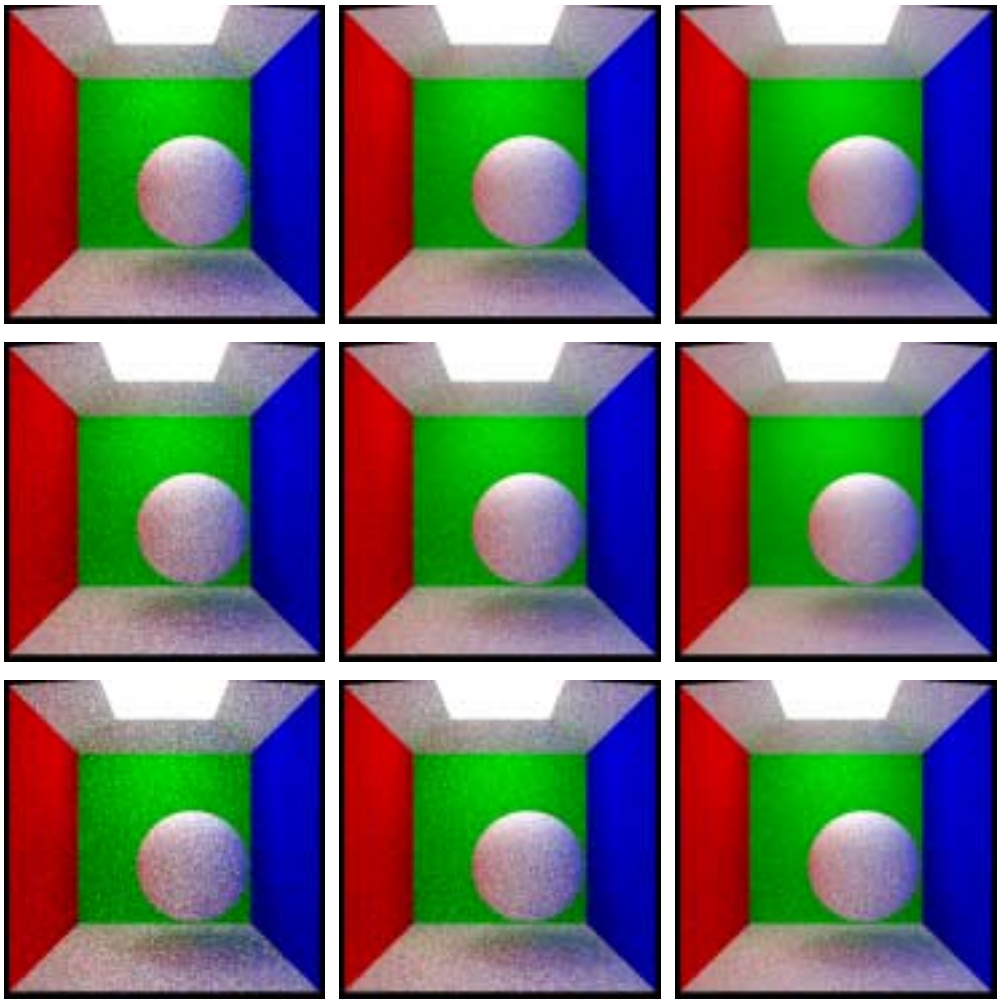



Abbildung E.9: Maximal erreichbare Qualität, Szene 1, mit Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.10: Bildqualität, Szene 1, ohne Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.11: Bildqualität, Szene 1, ohne Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

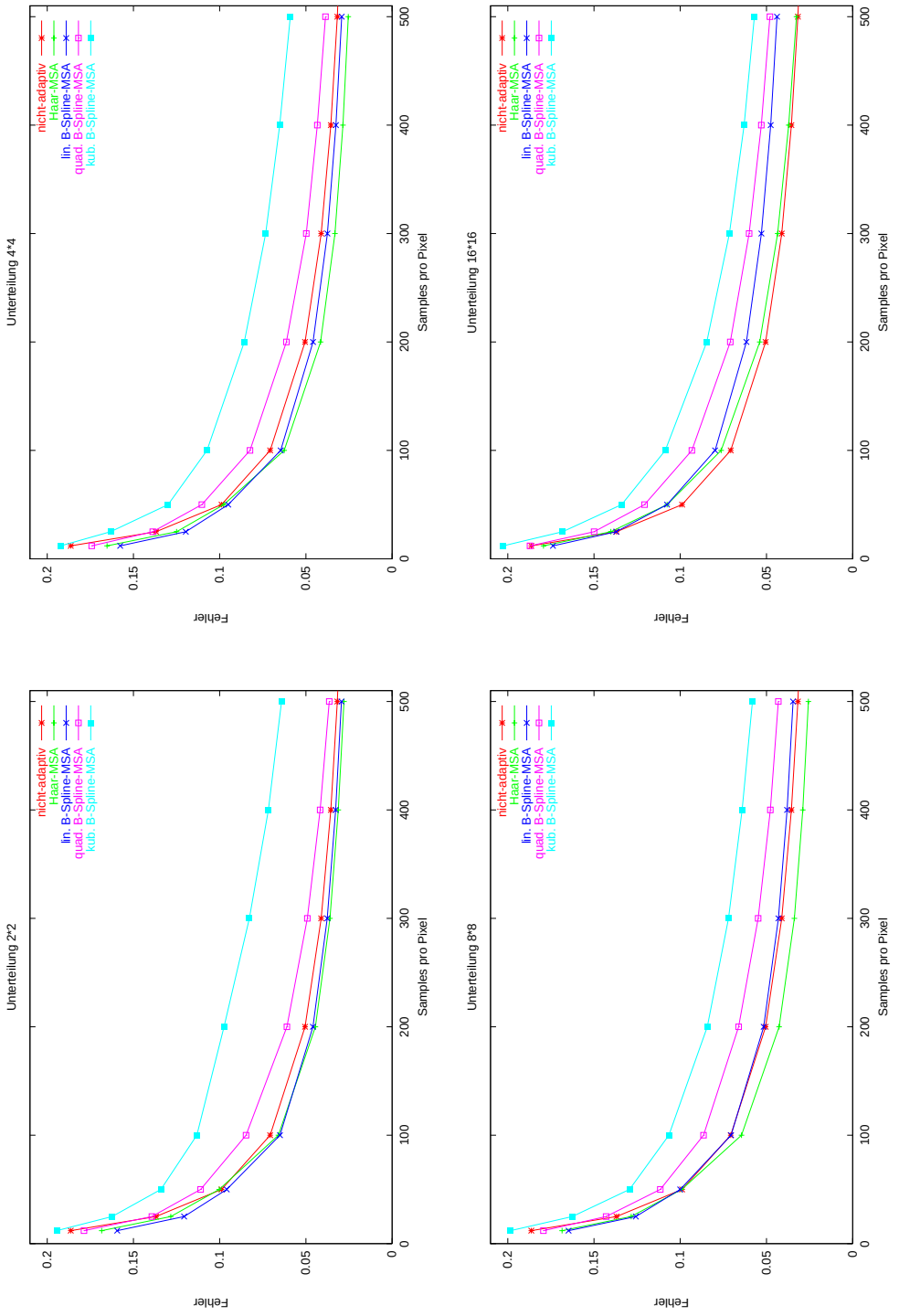
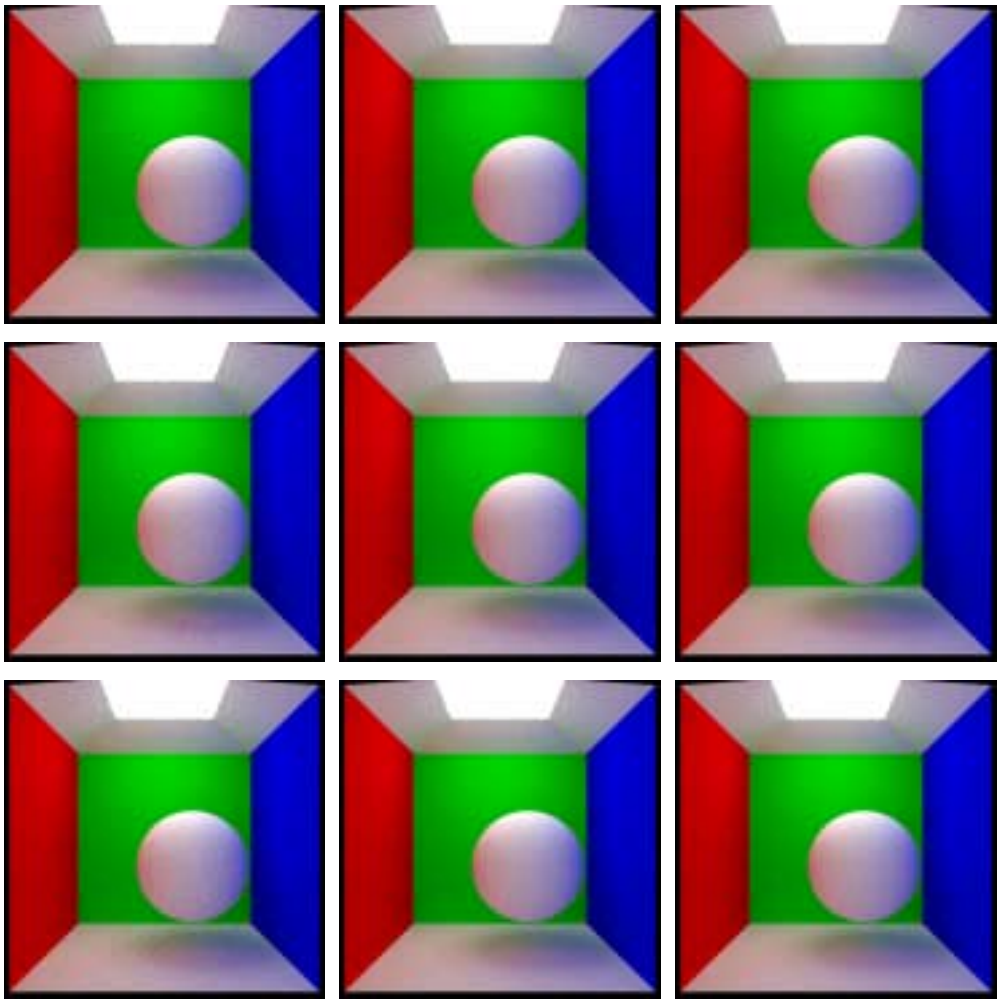
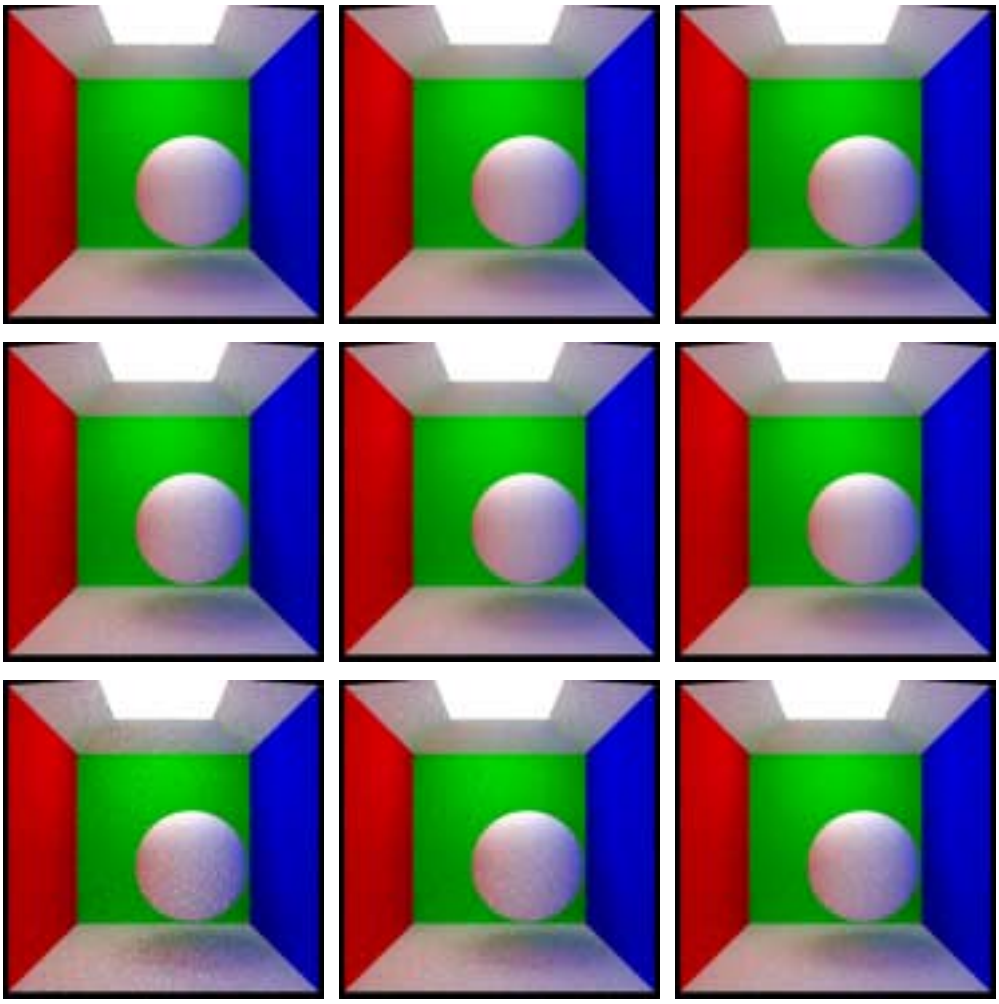



Abbildung E.12: Fehler als Funktion der Samplezahl, Szene 1, ohne Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.13: Bildqualität, Szene 1, mit Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.14: Bildqualität, Szene 1, mit Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

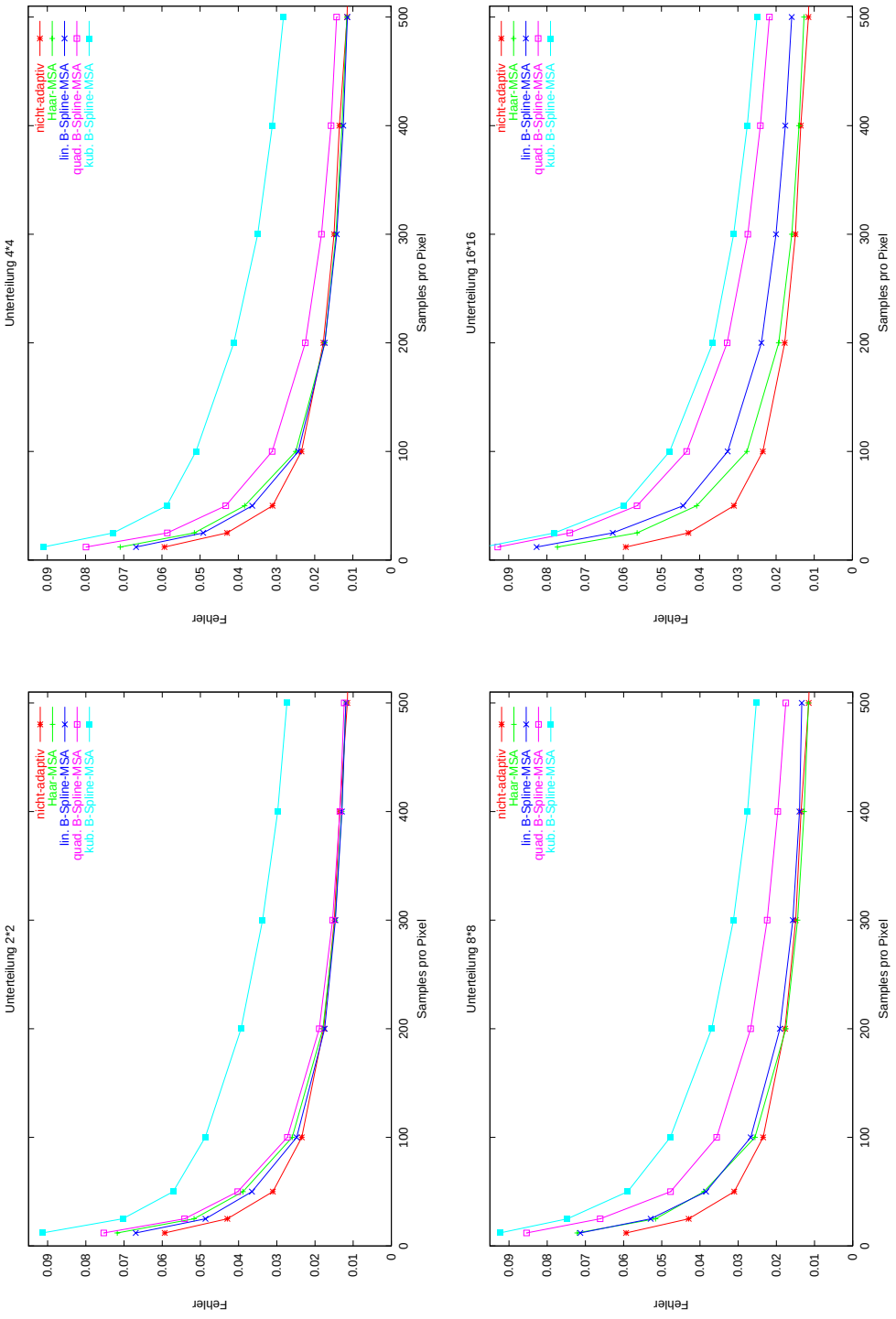



Abbildung E.15: Fehler als Funktion der Samplezahl, Szene 1, mit Stratifikation

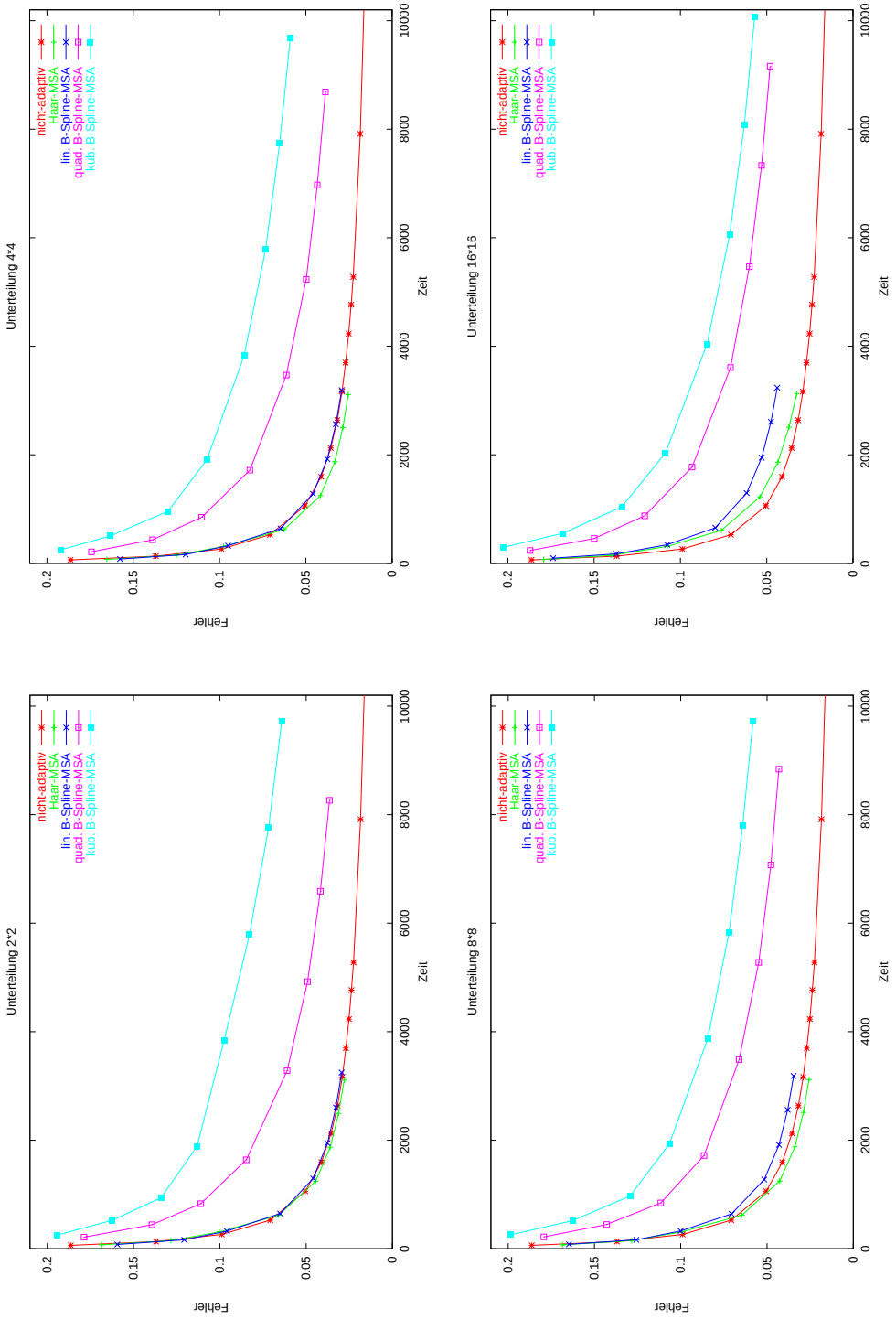



Abbildung E.16: Fehler als Funktion der Zeit, Szene 1, ohne Stratifikation

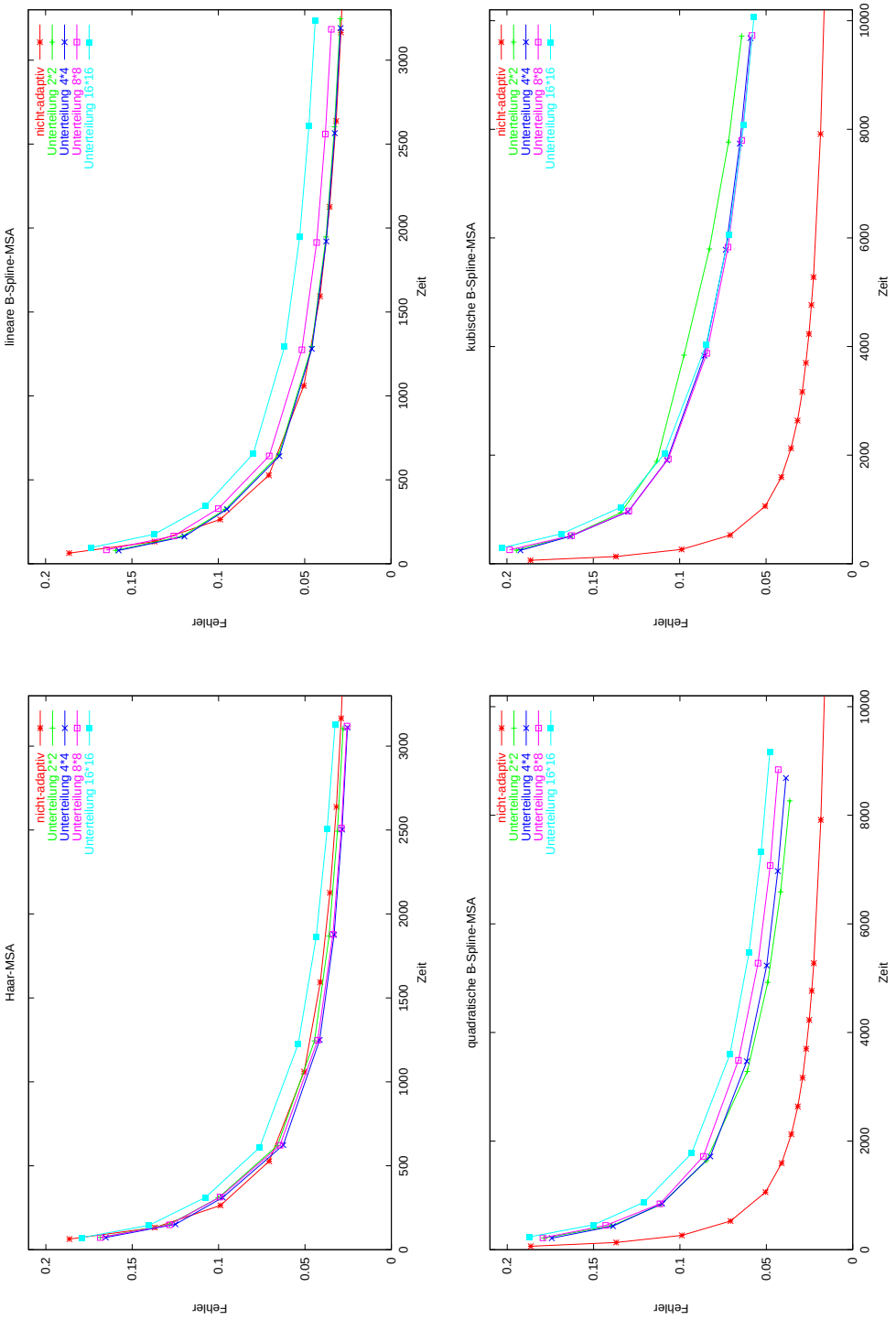



Abbildung E.17: Fehler als Funktion der Zeit, Szene 1, ohne Stratifikation

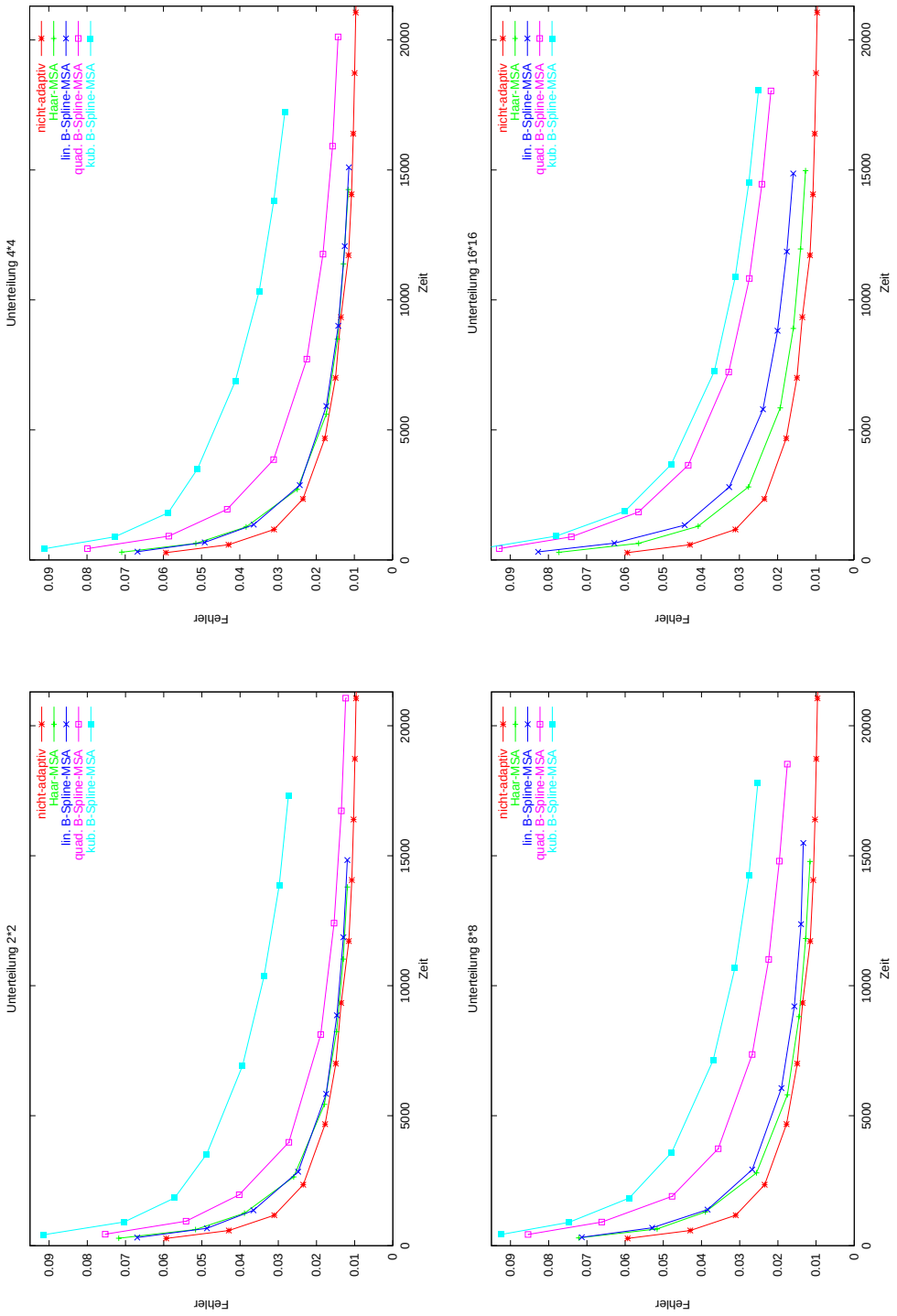



Abbildung E.18: Fehler als Funktion der Zeit, Szene 1, mit Stratifikation

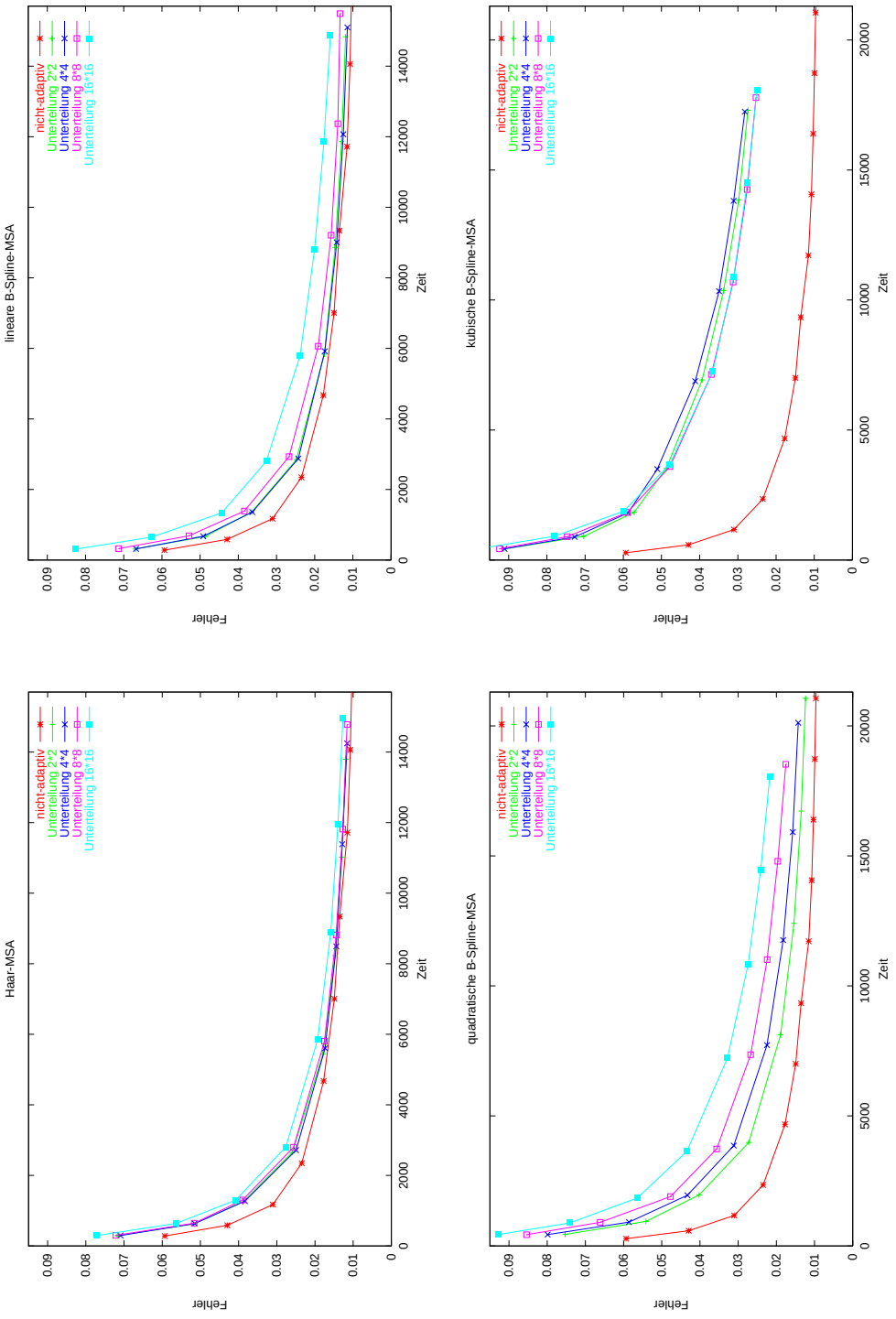



Abbildung E.19: Fehler als Funktion der Zeit, Szene 1, mit Stratifikation

## E.3 Szene 2

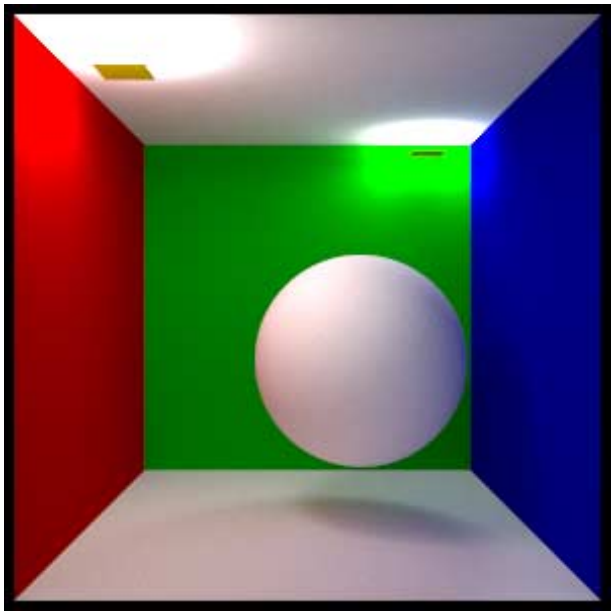


Abbildung E.20: Szene 2, Referenzlösung (nicht-adaptiv, mit Stratifikation, 96000 Samples pro Pixel)

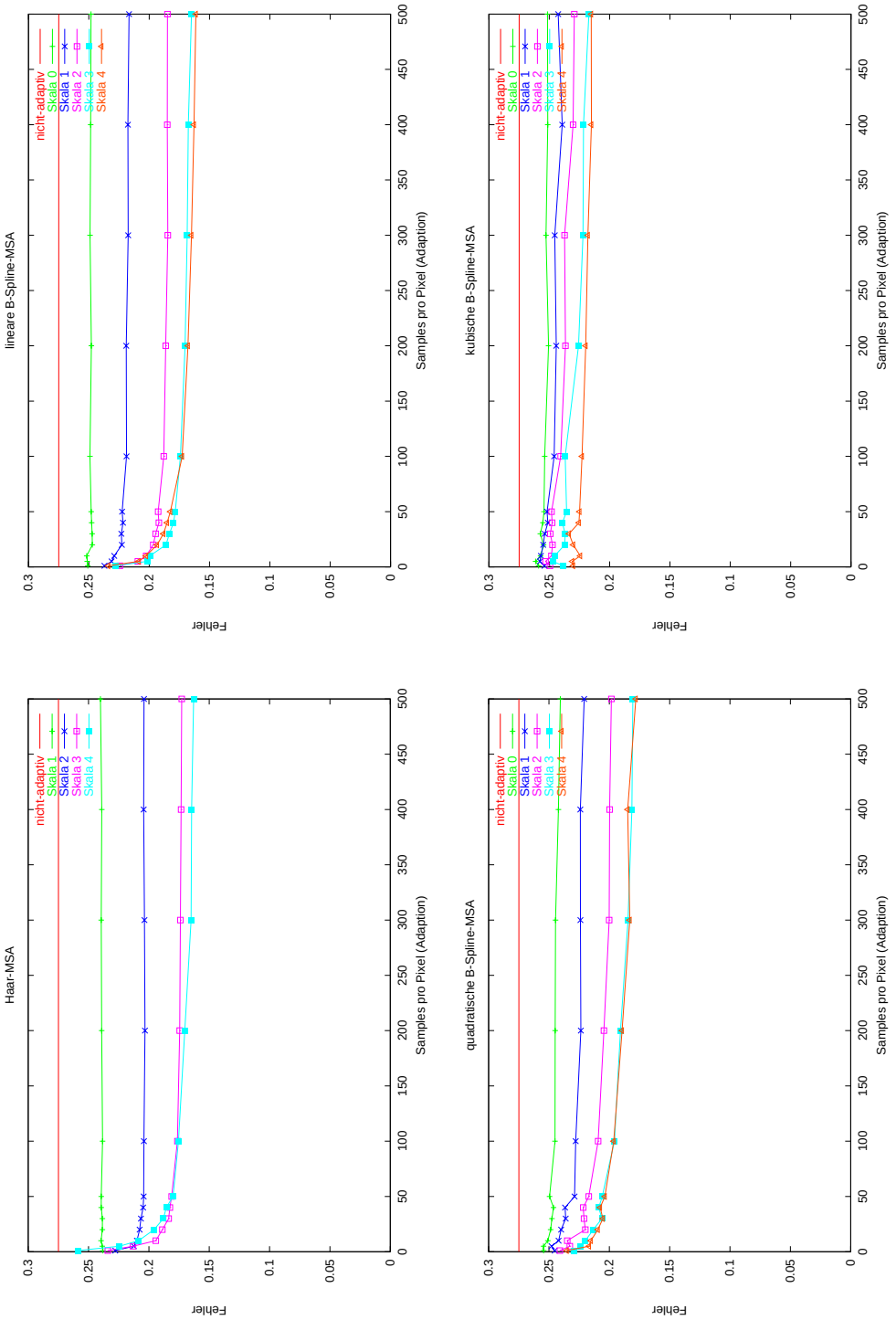



Abbildung E.21: Qualität als Funktion der Adaptions-Samplezahl, Szene 2, ohne Stratifikation, Unterteilung $4 \cdot 4$

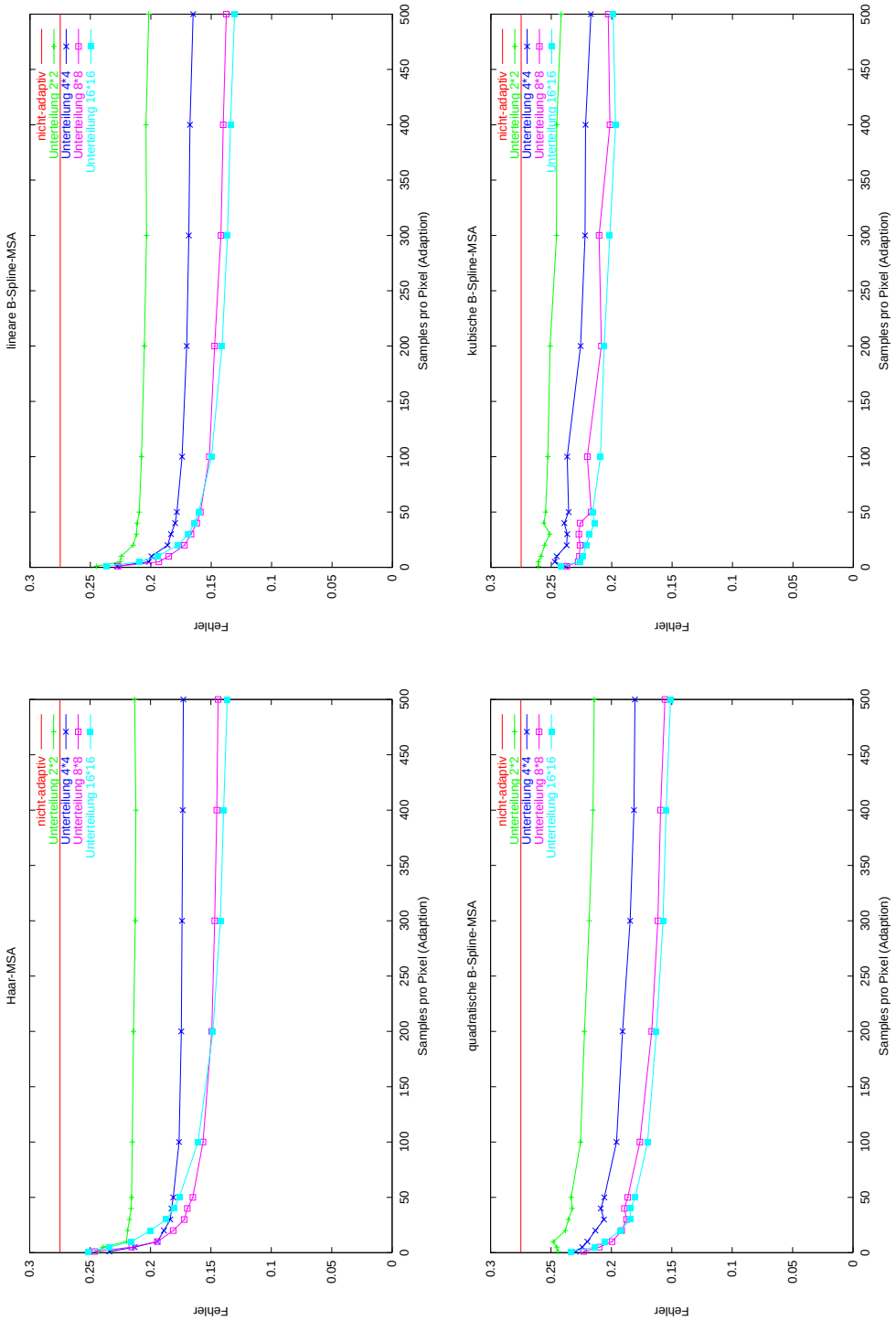



Abbildung E.22: Qualität als Funktion der Adaptions-Samplezahl, Szene 2, ohne Stratifikation, Skala 3

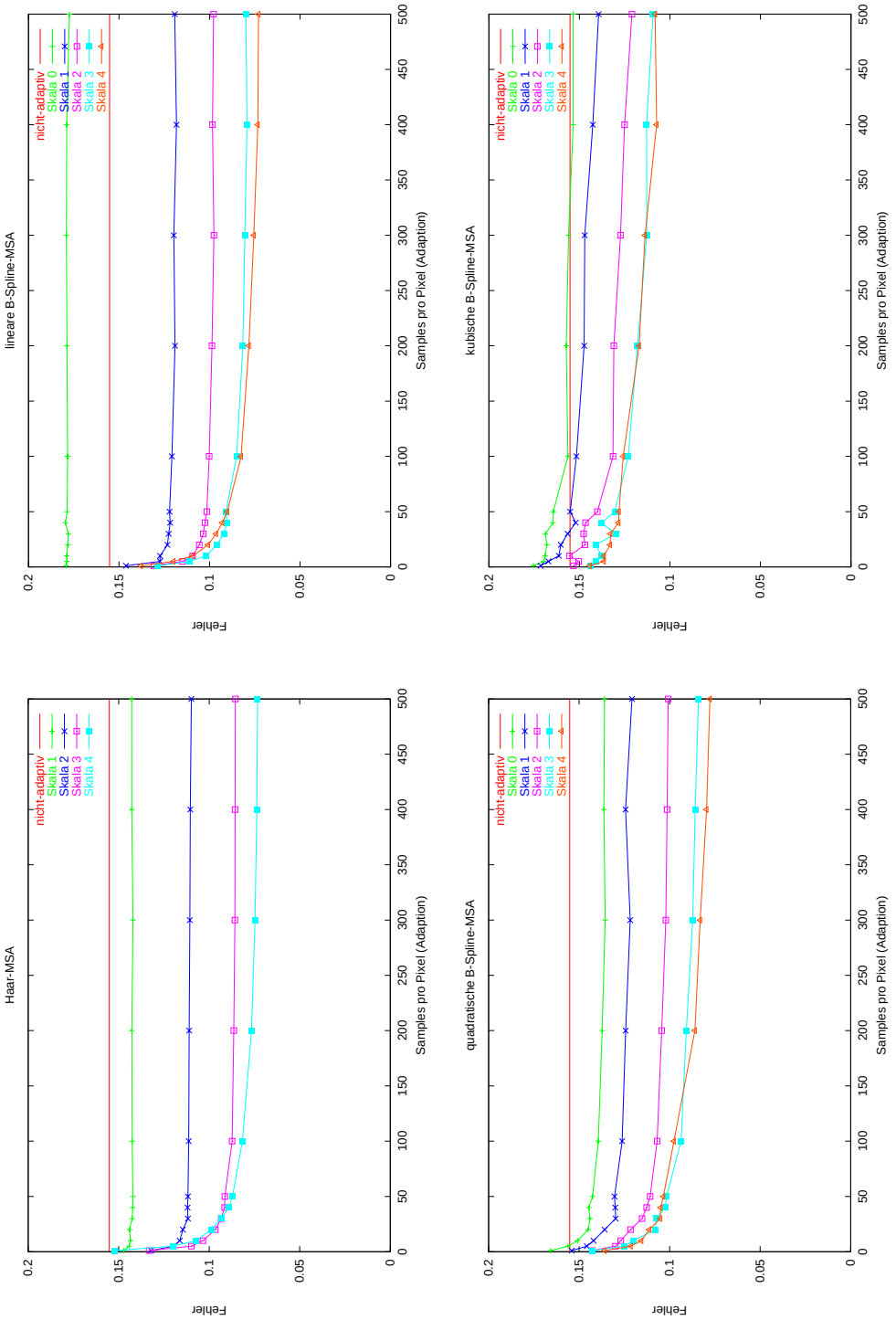



Abbildung E.23: Qualität als Funktion der Adaptions-Samplezahl, Szene 2, mit Stratifikation, Unterteilung $4 \cdot 4$

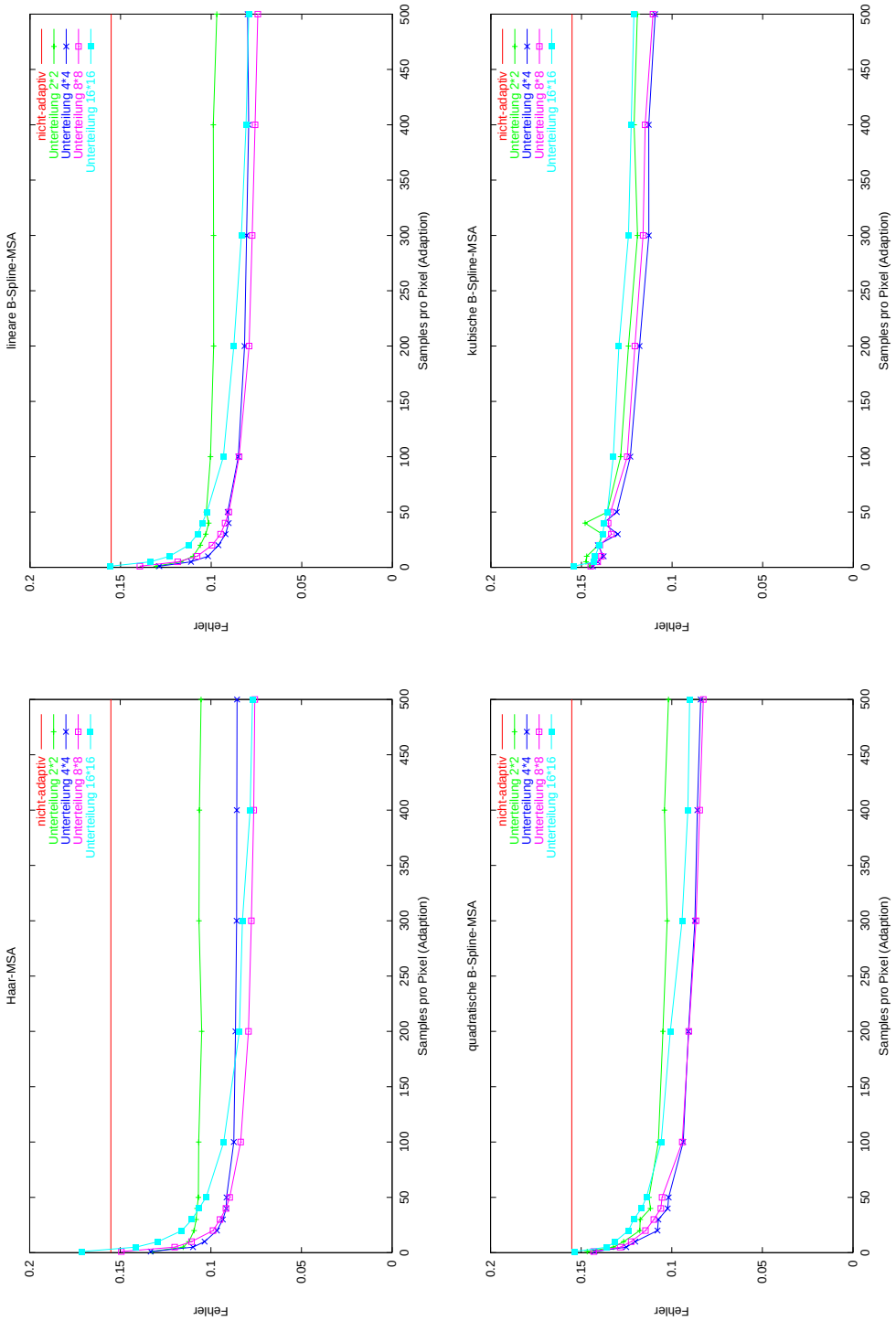



Abbildung E.24: Qualität als Funktion der Adaptions-Samplezahl, Szene 2, mit Stratifikation, Skala 3

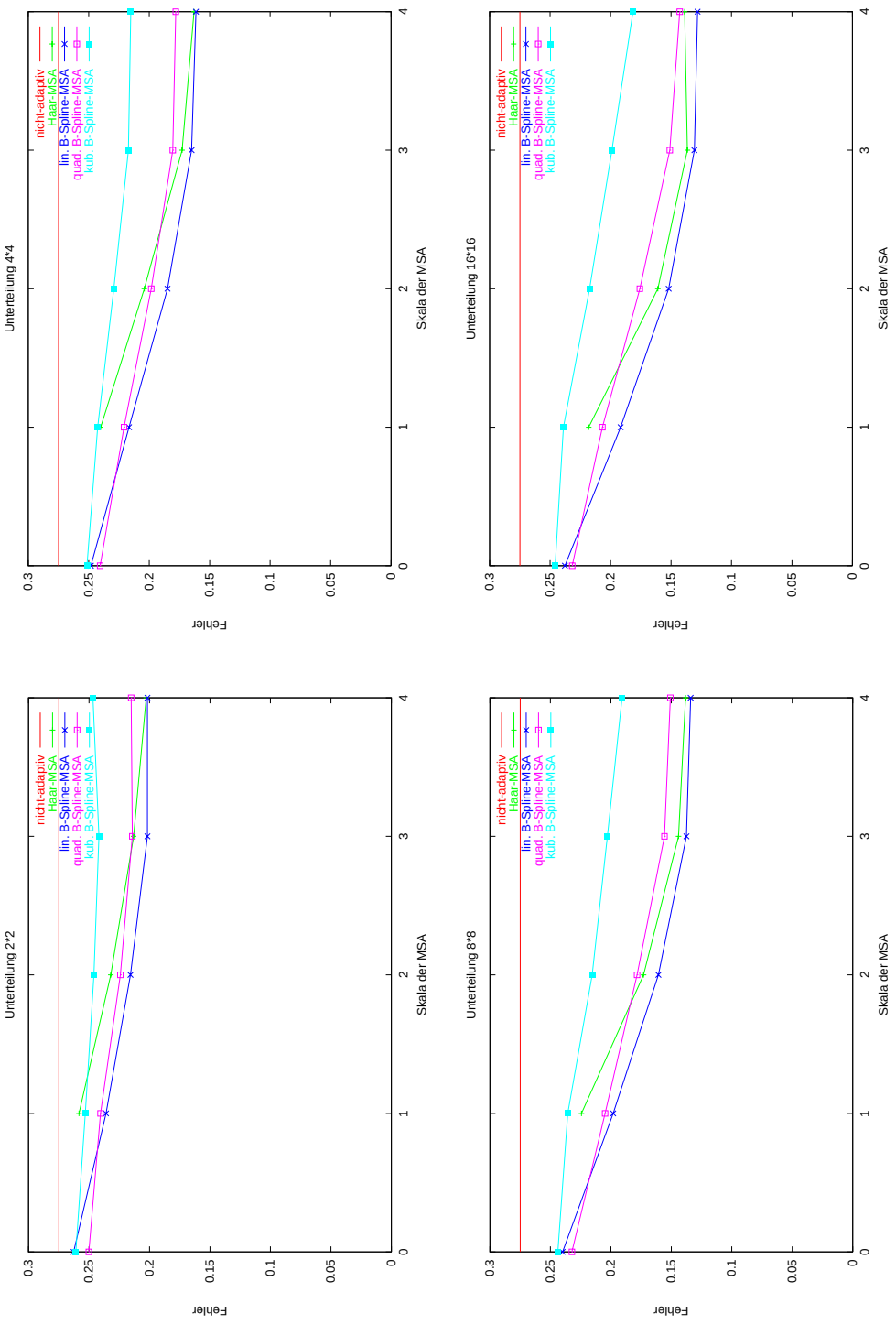



Abbildung E.25: Maximal erreichbare Qualität, Szene 2, ohne Stratifikation

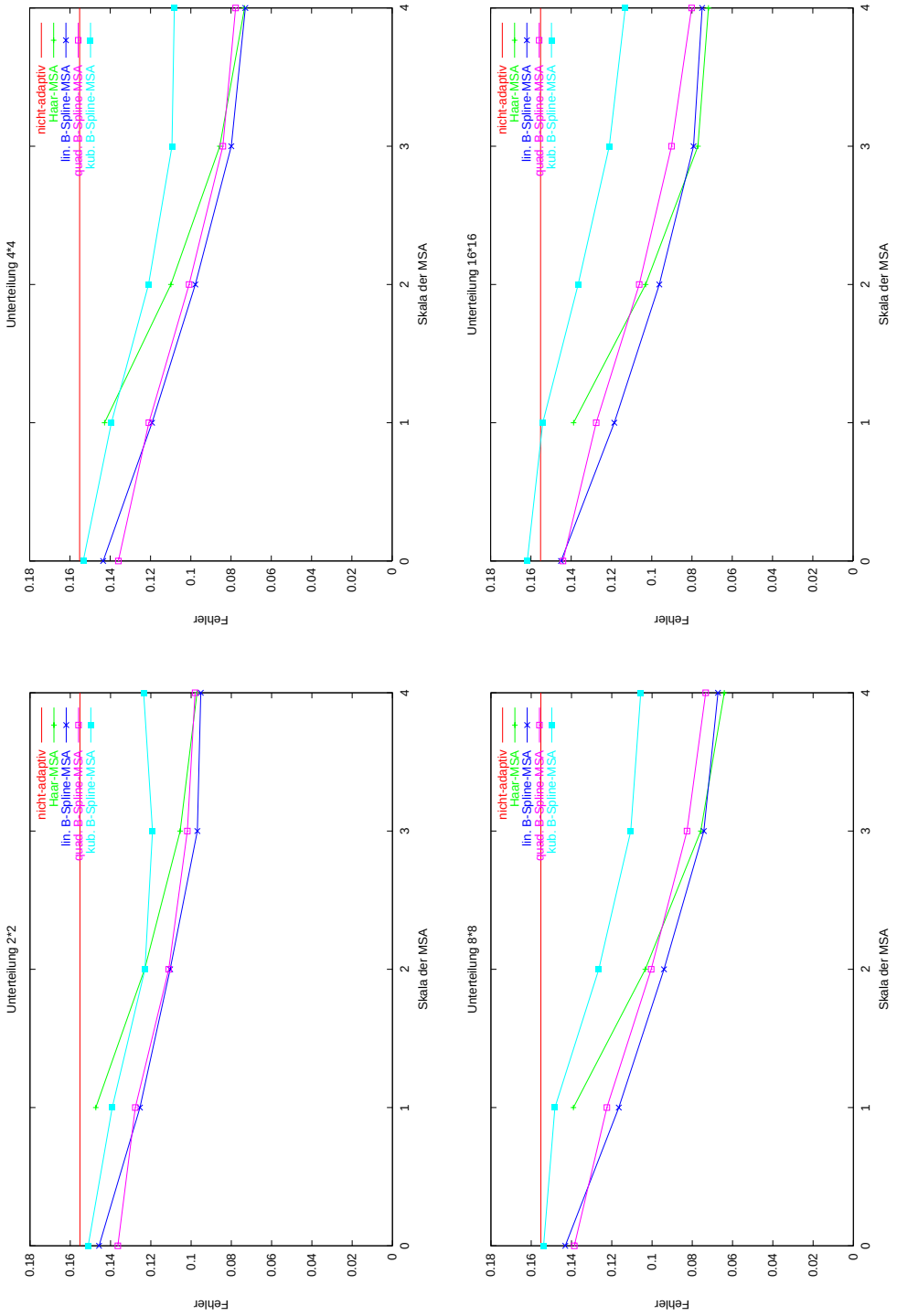
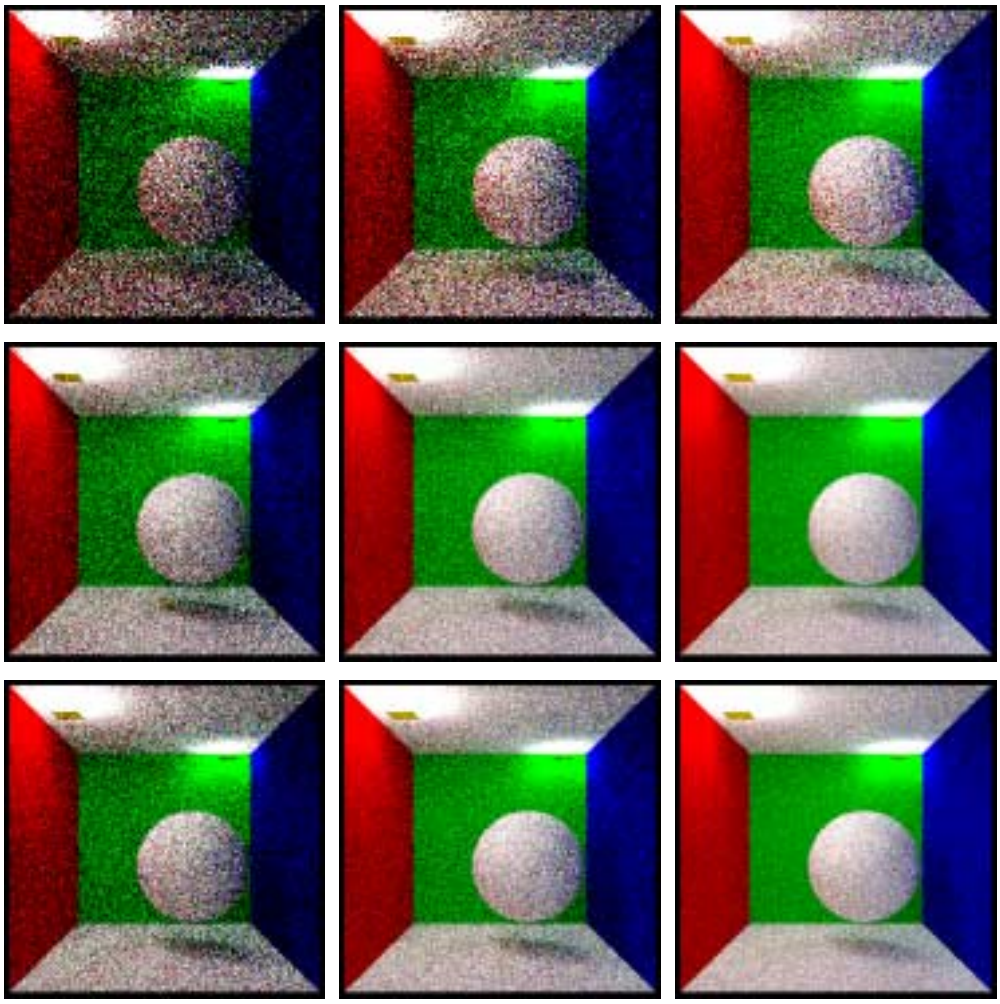
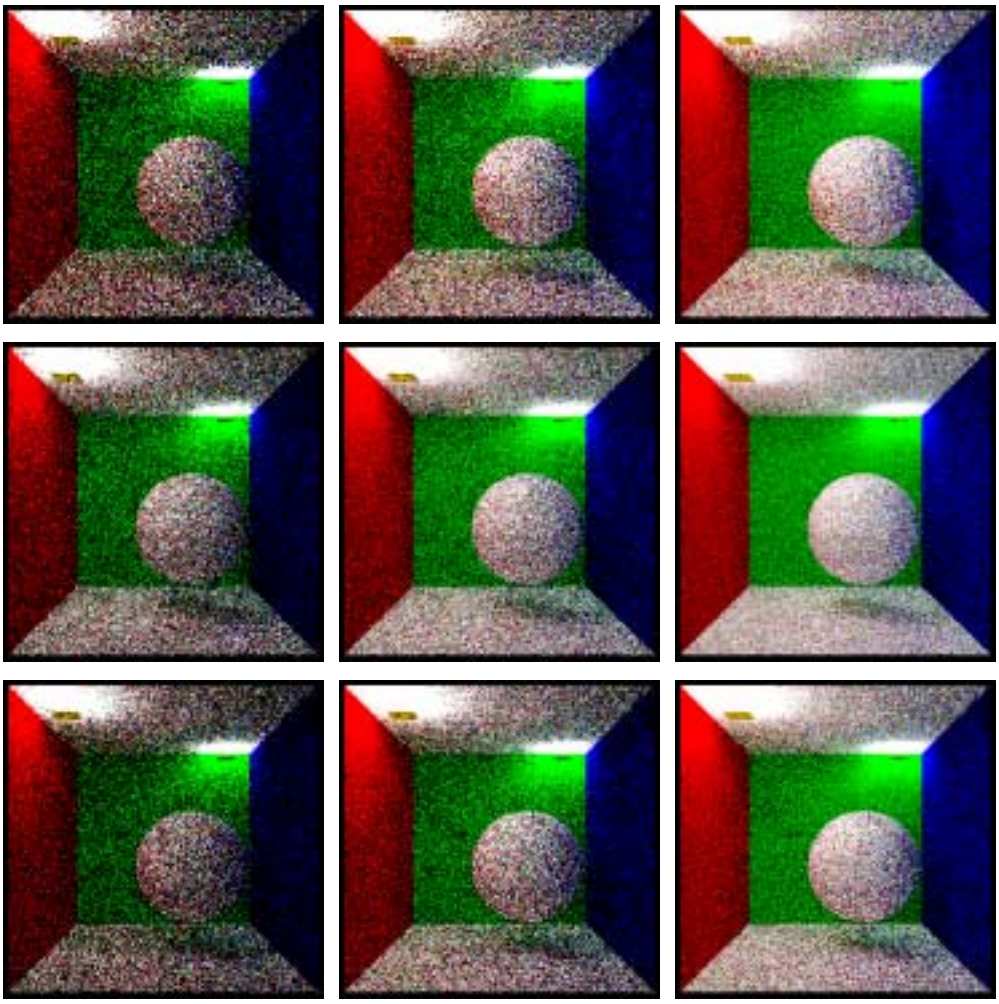



Abbildung E.26: Maximal erreichbare Qualität, Szene 2, mit Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.27: Bildqualität, Szene 2, ohne Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.28: Bildqualität, Szene 2, ohne Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

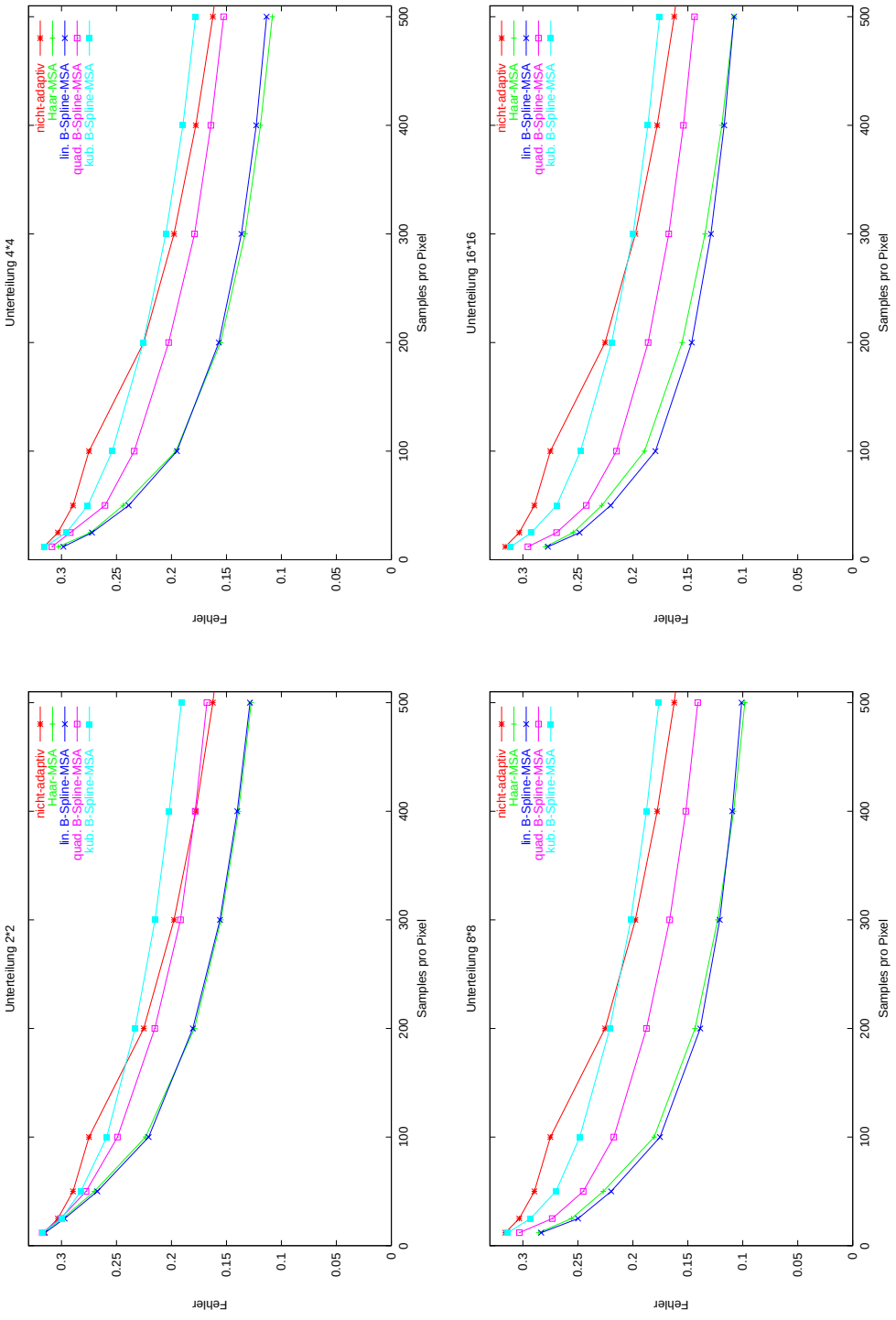
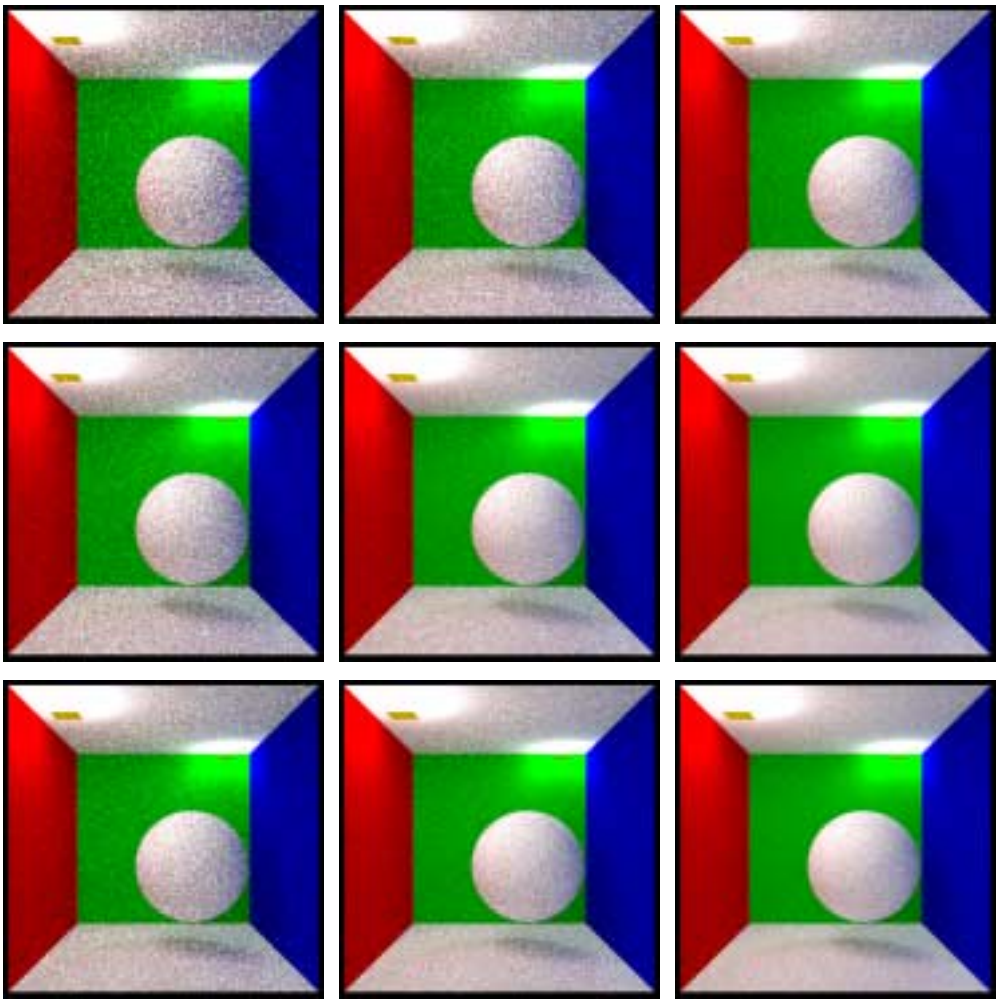
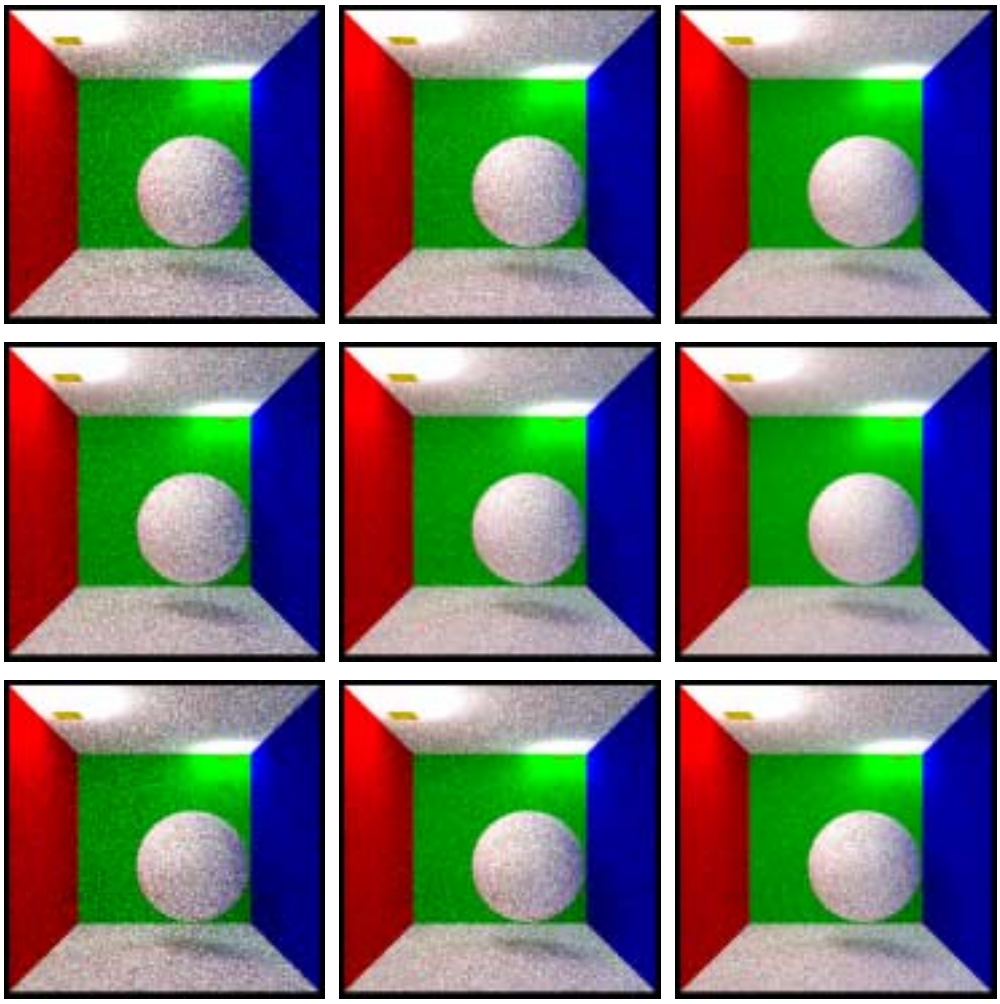



Abbildung E.29: Fehler als Funktion der Samplezahl, Szene 2, ohne Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.30: Bildqualität, Szene 2, mit Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.31: Bildqualität, Szene 2, mit Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

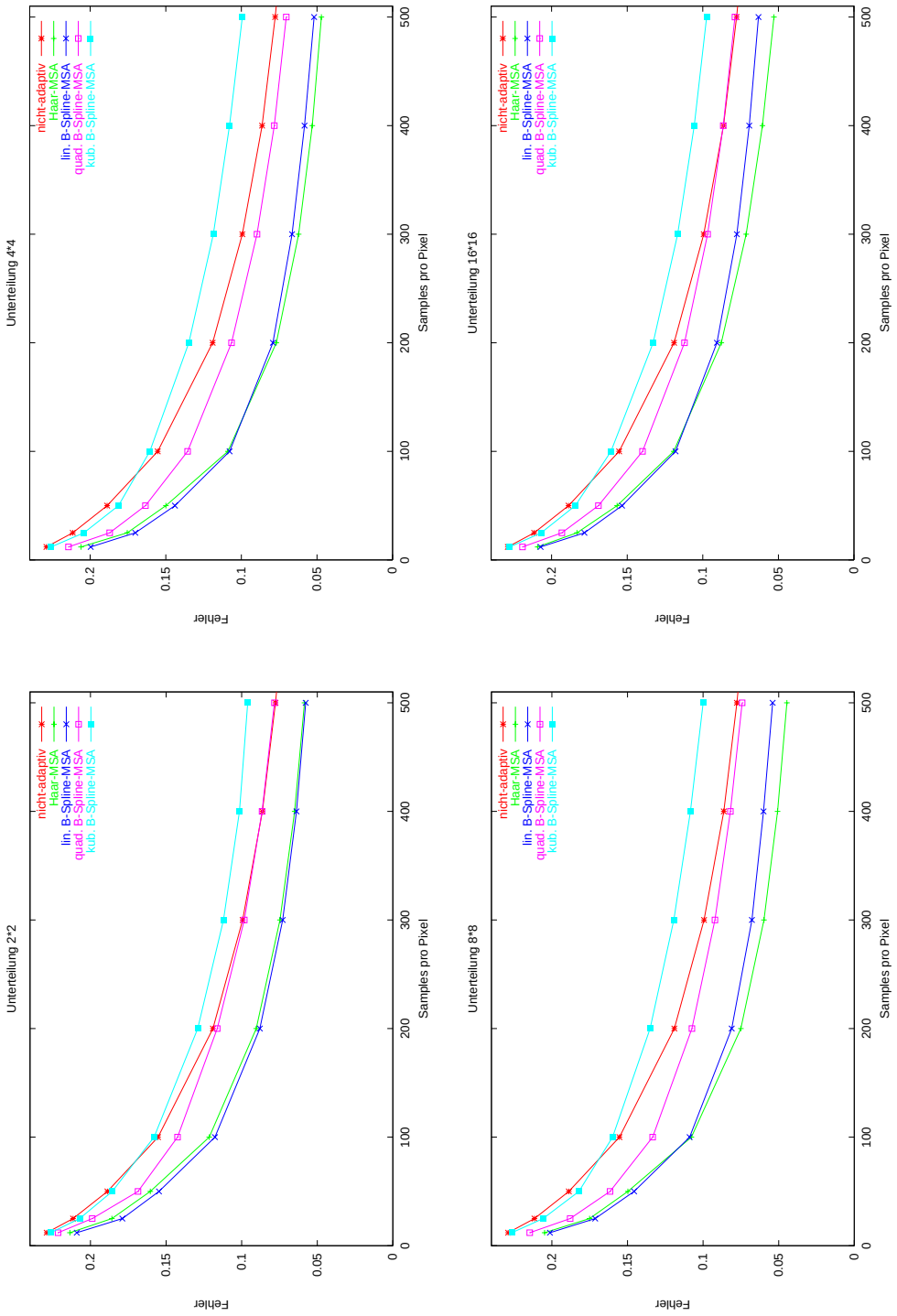



Abbildung E.32: Fehler als Funktion der Samplezahl, Szene 2, mit Stratifikation

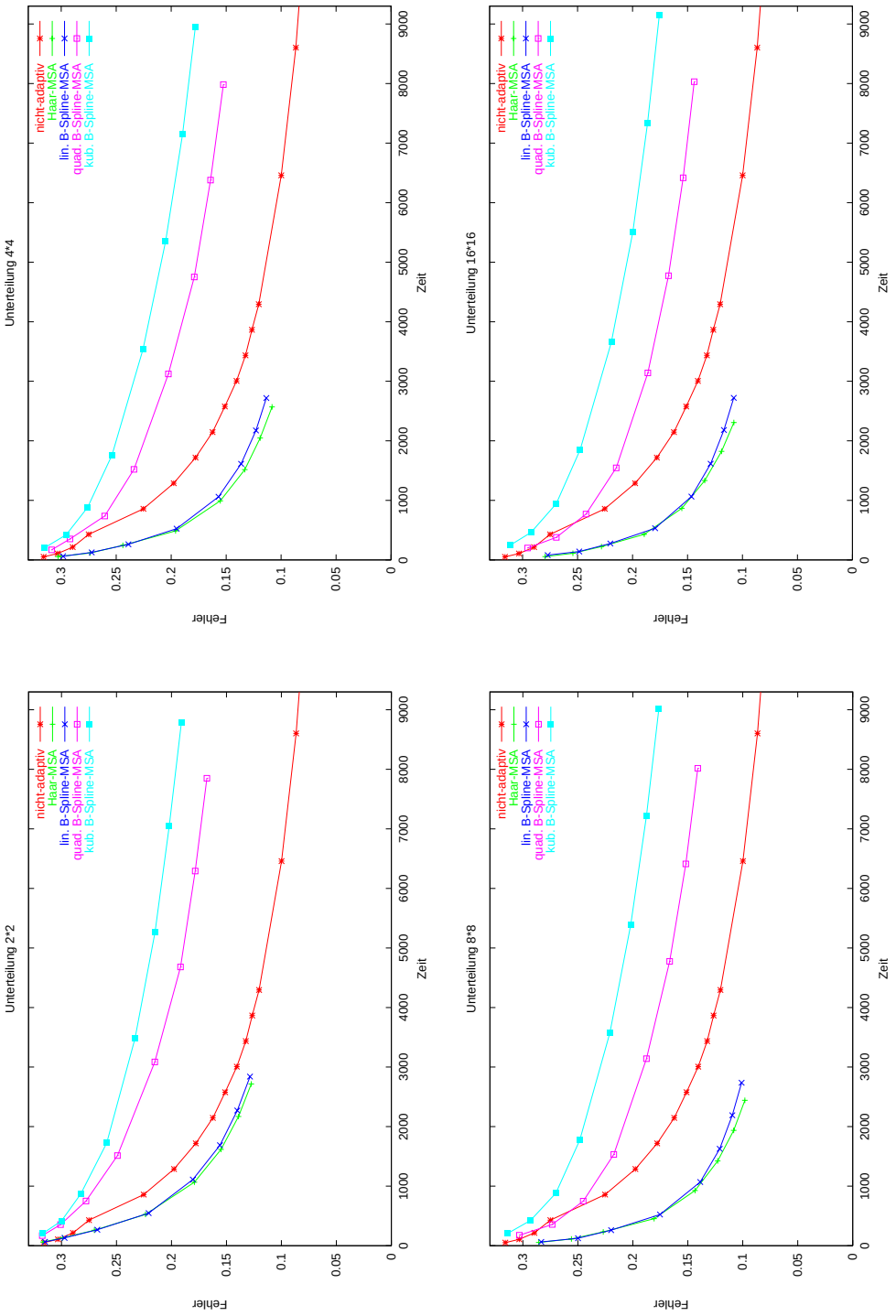



Abbildung E.33: Fehler als Funktion der Zeit, Szene 2, ohne Stratifikation

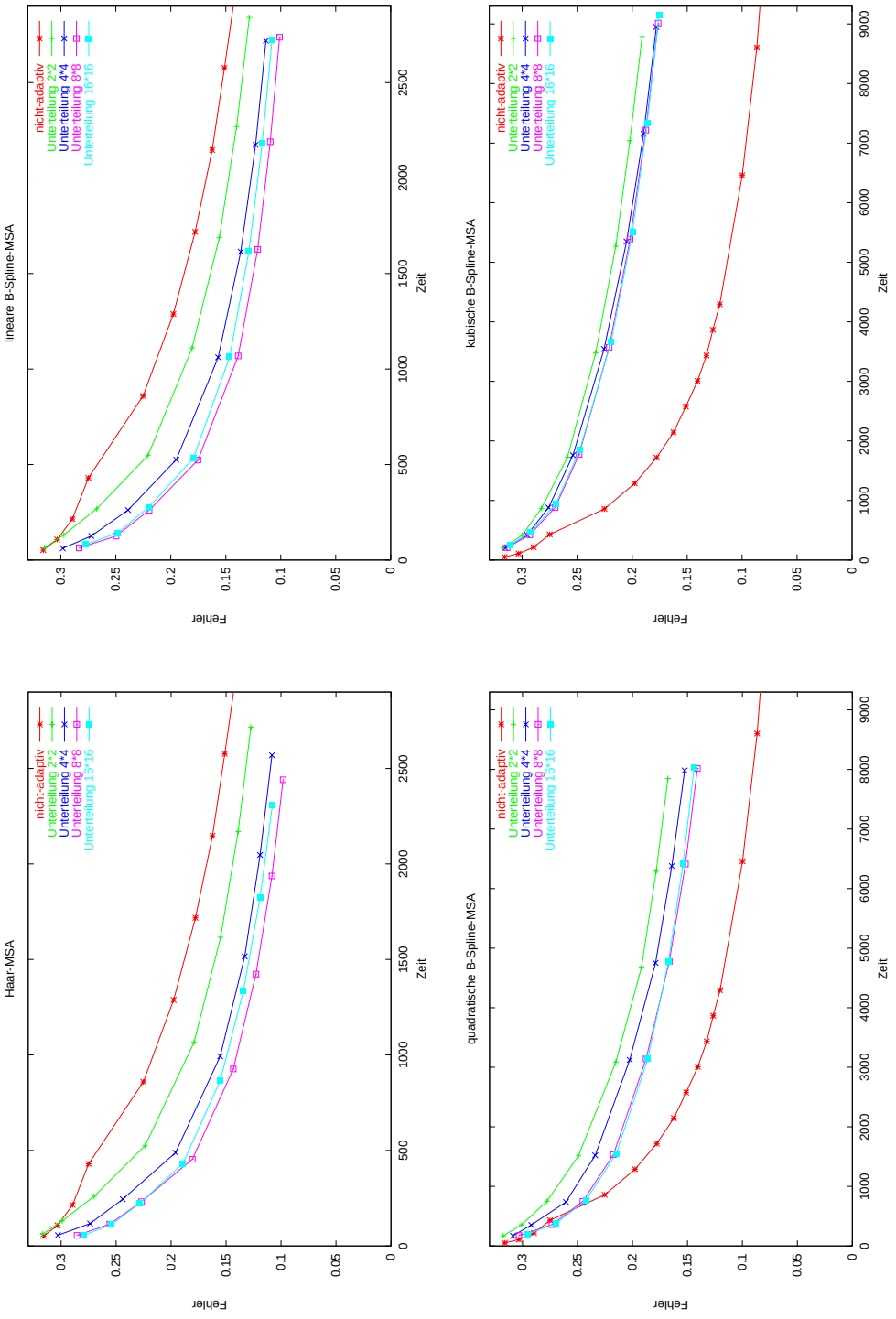



Abbildung E.34: Fehler als Funktion der Zeit, Szene 2, ohne Stratifikation

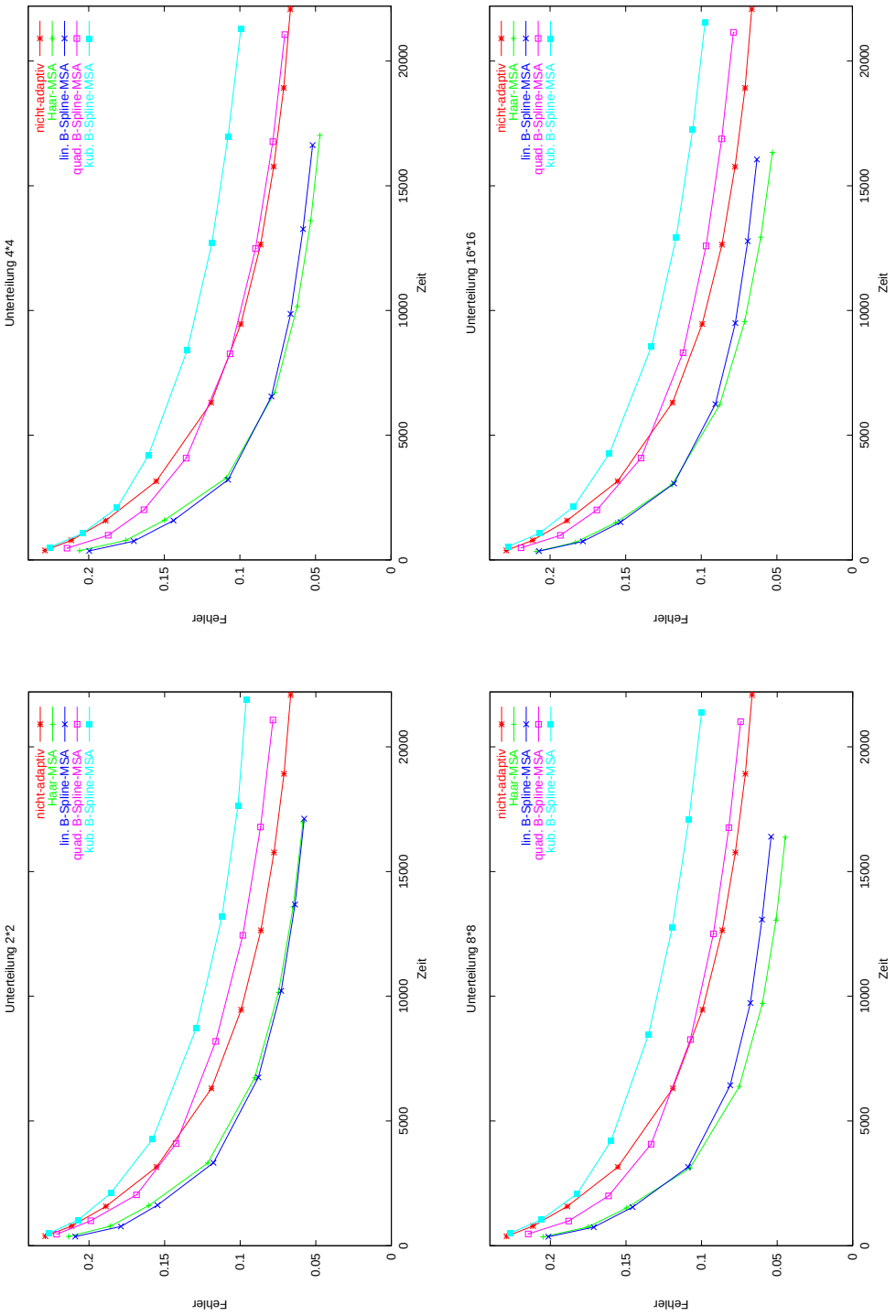



Abbildung E.35: Fehler als Funktion der Zeit, Szene 2, mit Stratifikation

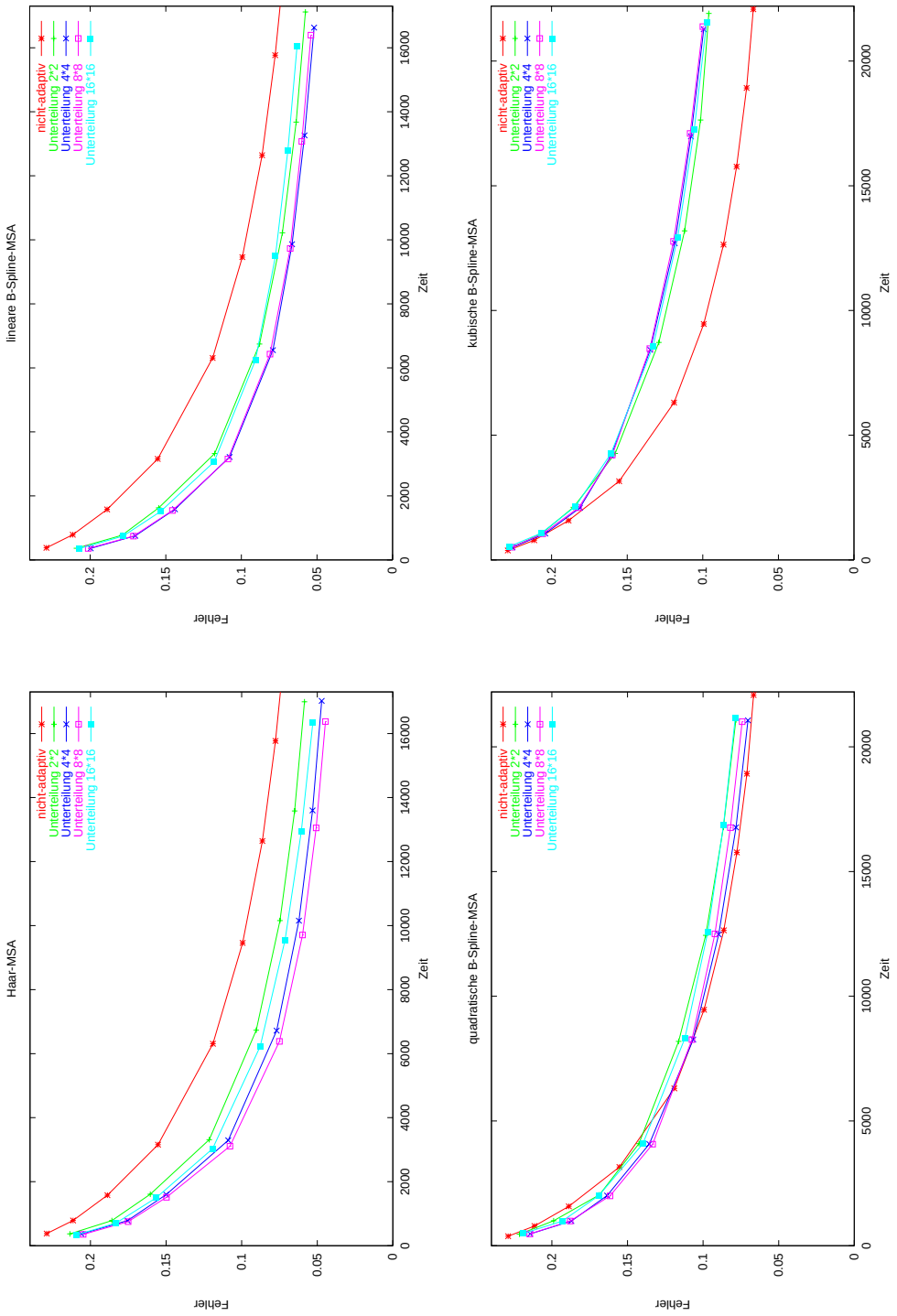



Abbildung E.36: Fehler als Funktion der Zeit, Szene 2, mit Stratifikation

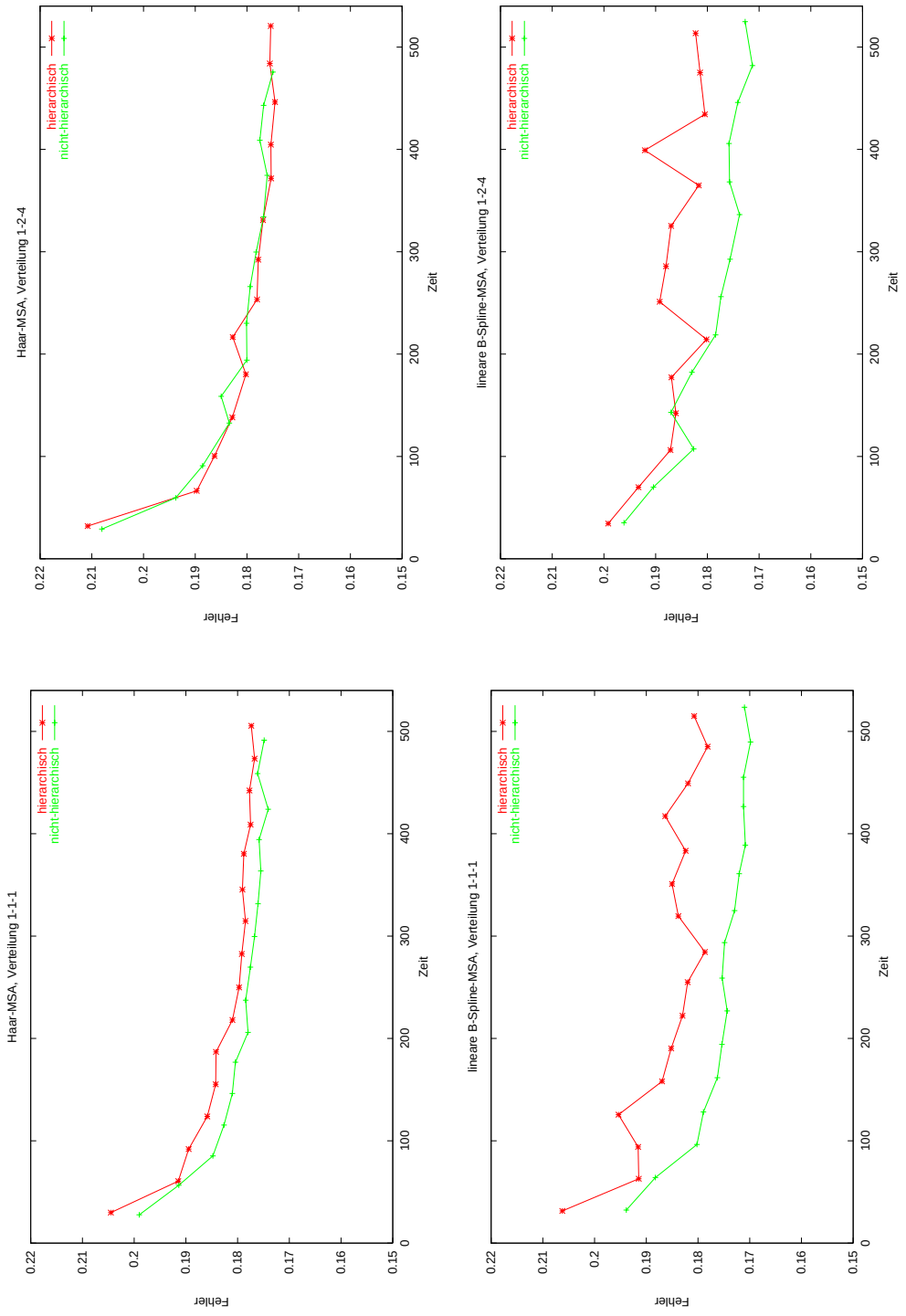



Abbildung E.37: Qualität als Funktion der Adaptions-Samplezahl und Verteilung, Szene 2, ohne Stratifikation

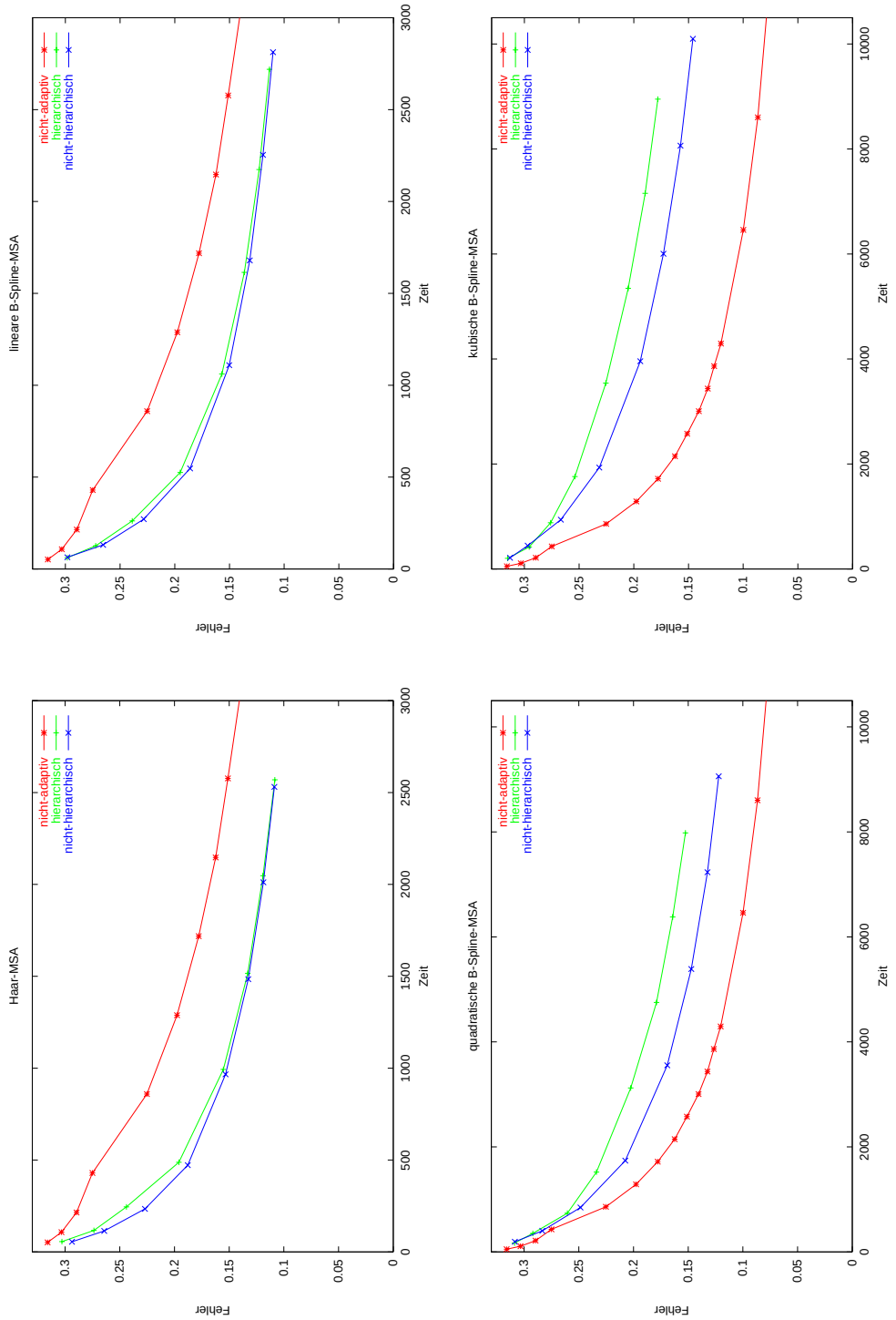



Abbildung E.38: Fehler als Funktion der Zeit, Szene 2, ohne Stratifikation, Unterteilung $4 \cdot 4$

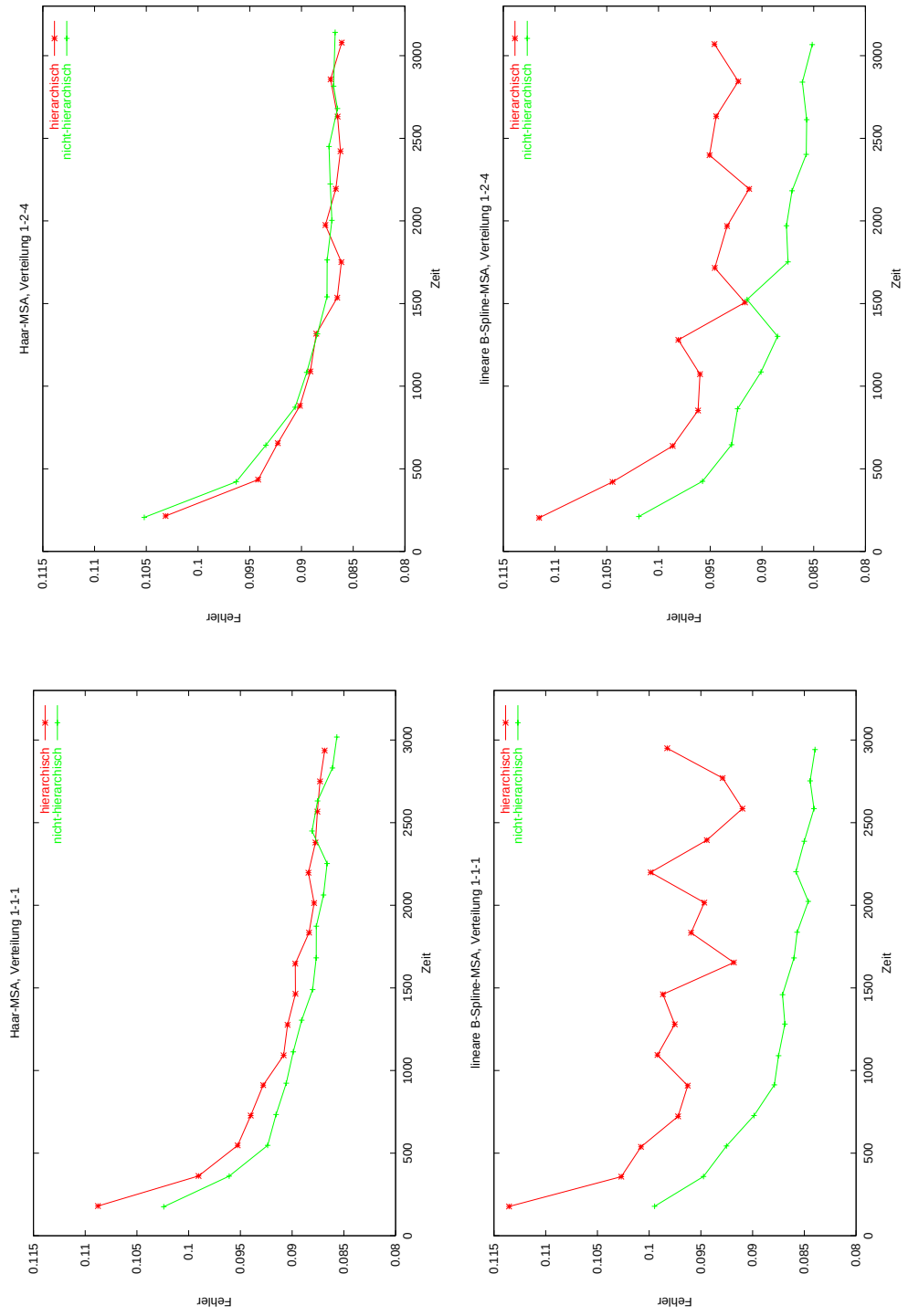



Abbildung E.39: Qualität als Funktion der Adaptions-Samplezahl und Verteilung, Szene 2, mit Stratifikation

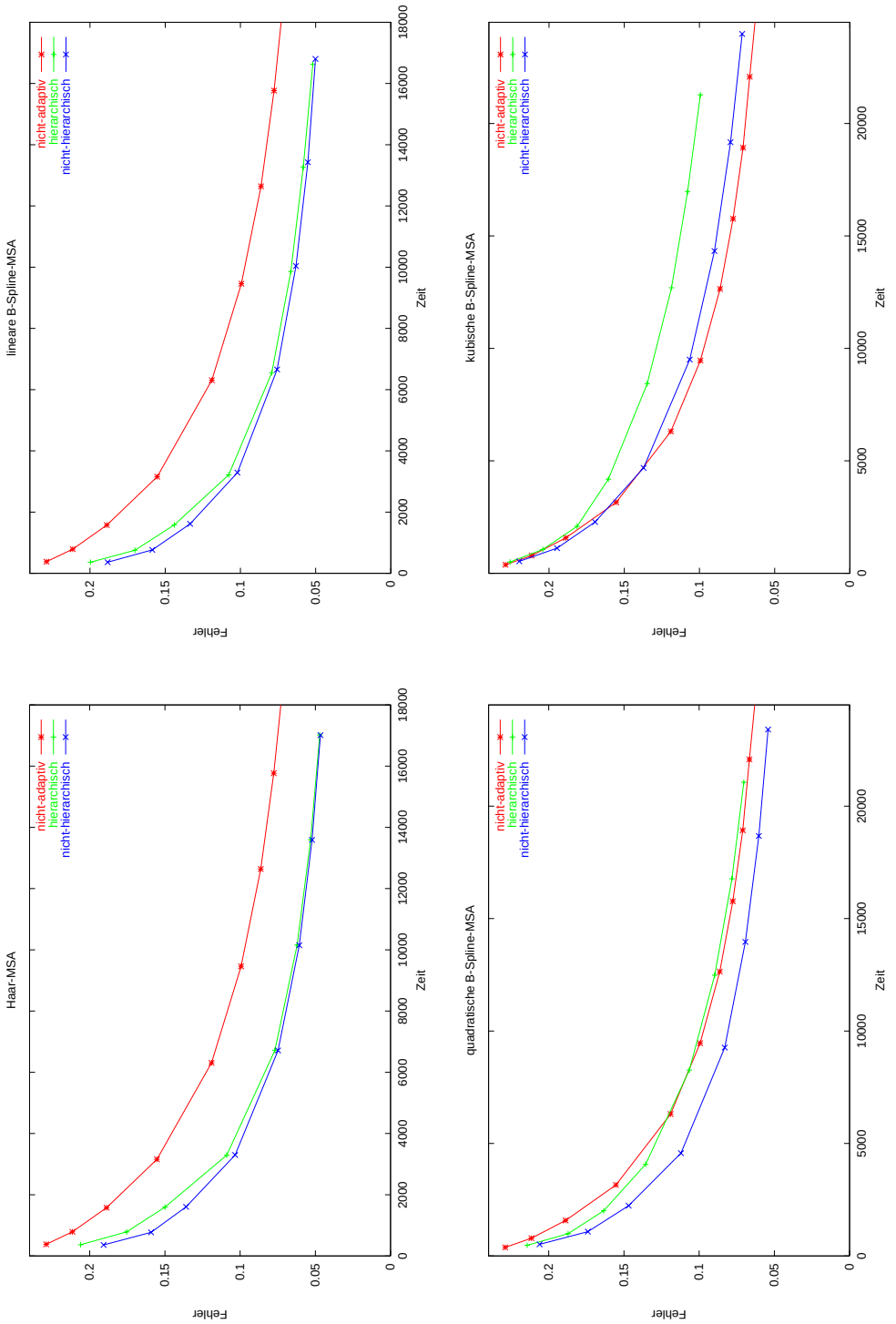



Abbildung E.40: Fehler als Funktion der Zeit, Szene 2, mit Stratifikation, Unterteilung $4 \cdot 4$

## E.4 Szene 3

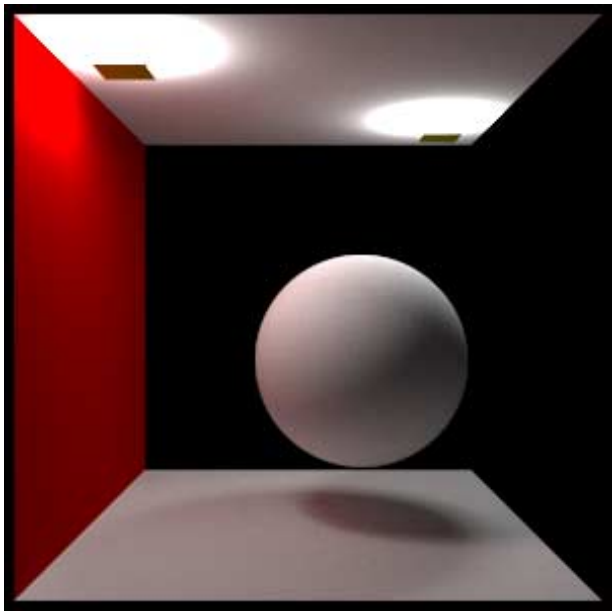


Abbildung E.41: Szene 3, Referenzlösung (nicht-adaptiv, mit Stratifikation, 96000 Samples pro Pixel)

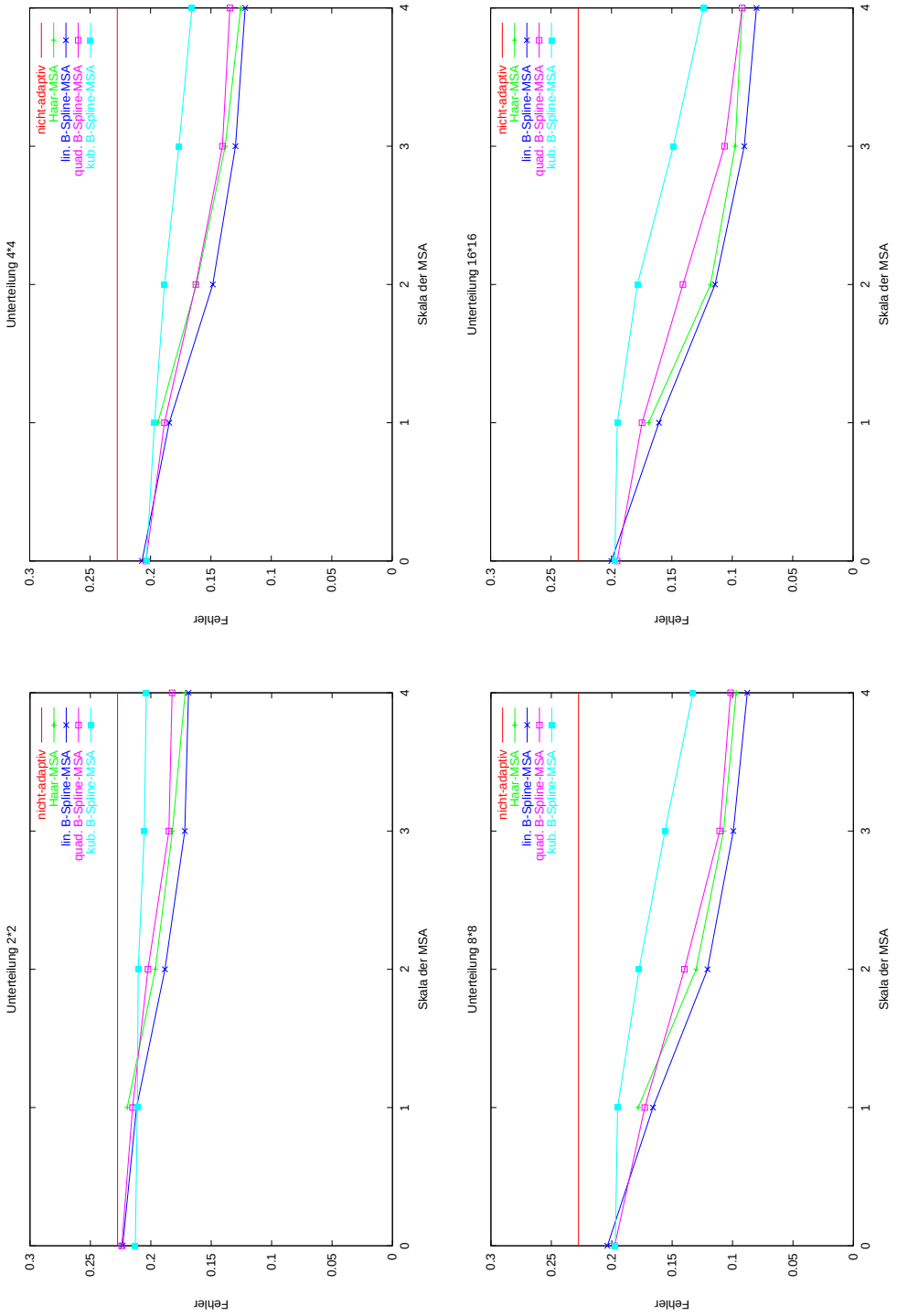



Abbildung E.42: Maximal erreichbare Qualität, Szene 3, ohne Stratifikation

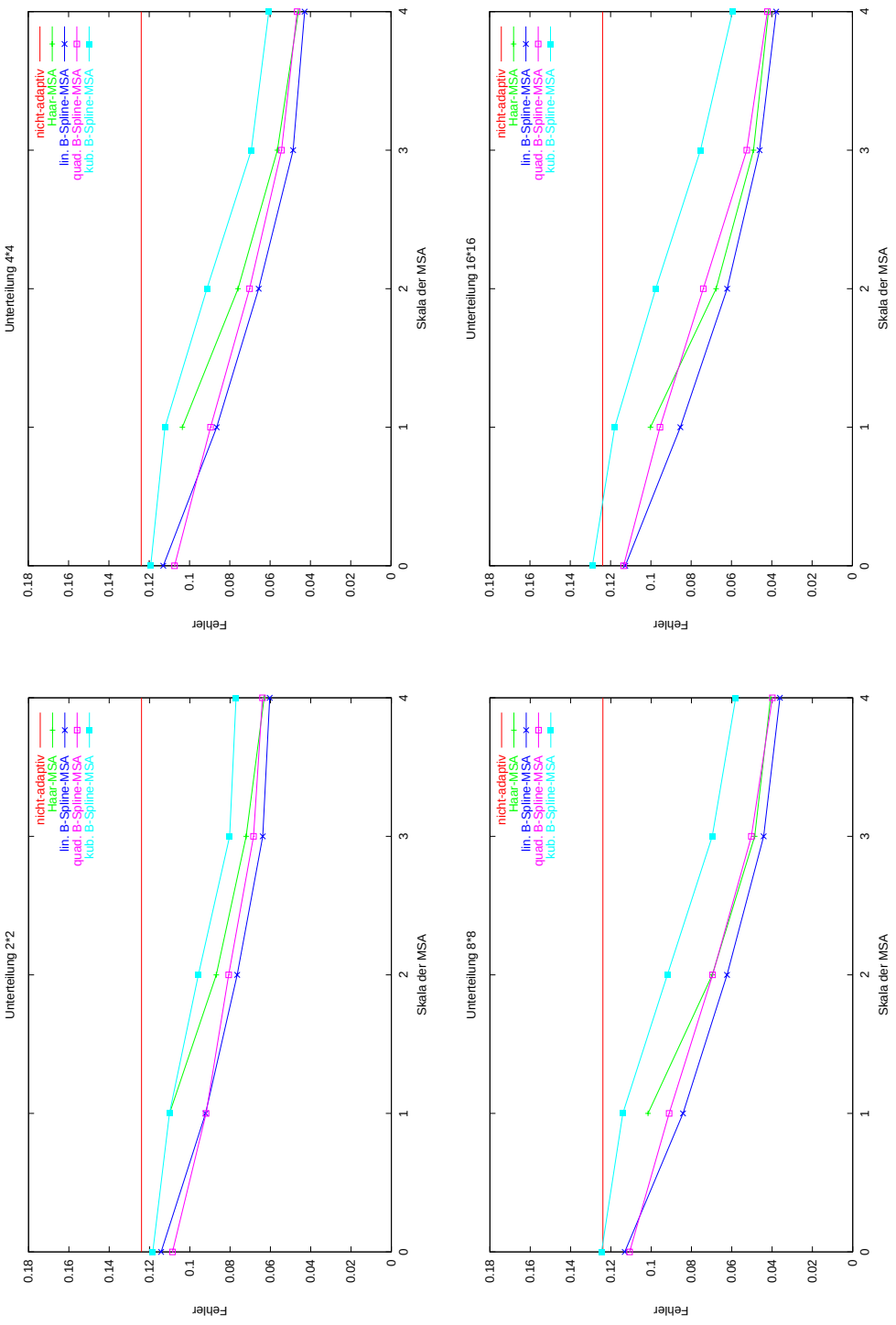
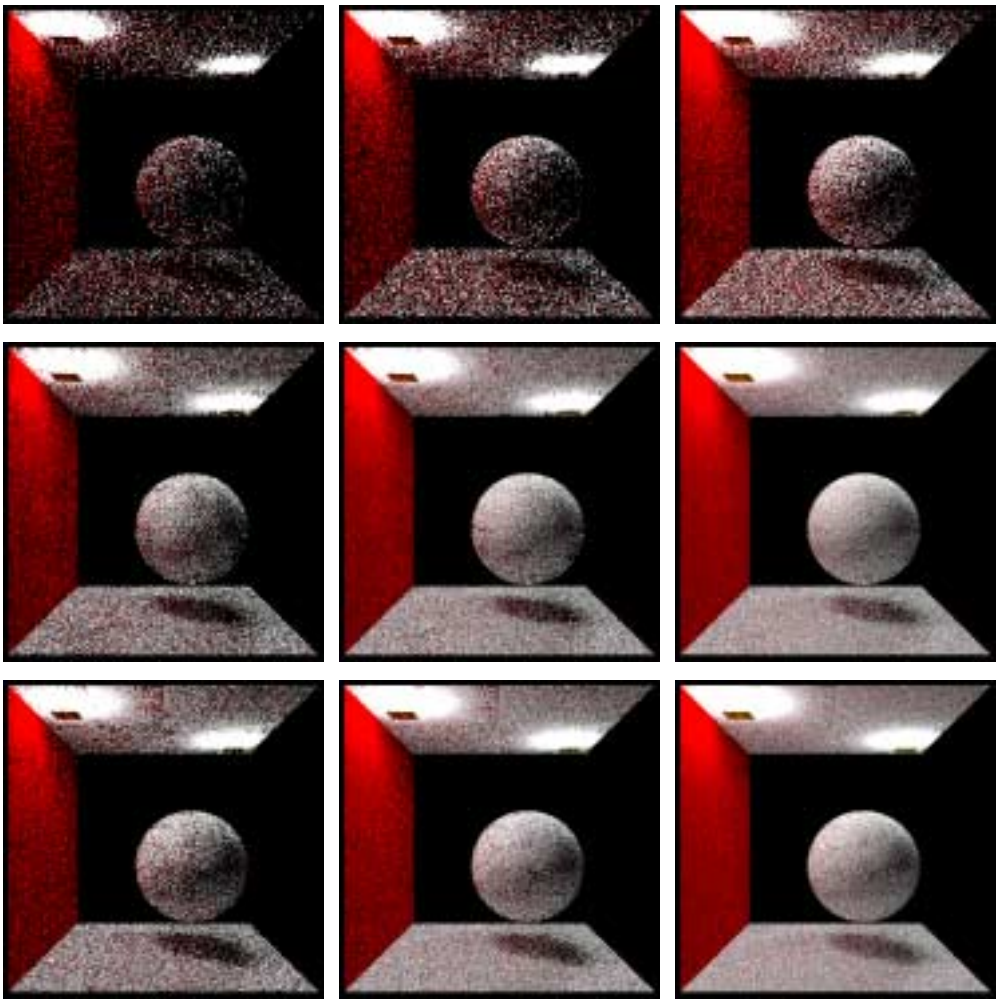
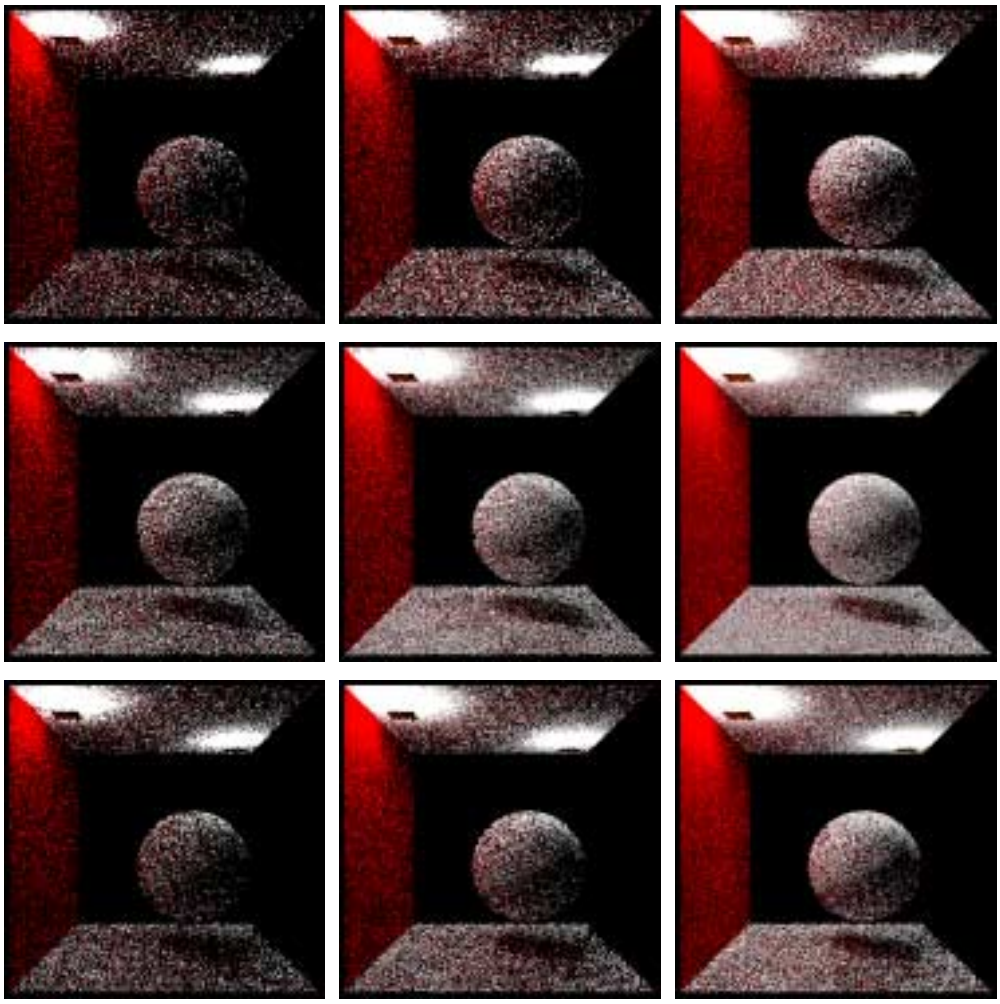



Abbildung E.43: Maximal erreichbare Qualität, Szene 3, mit Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.44: Bildqualität, Szene 3, ohne Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.45: Bildqualität, Szene 3, ohne Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

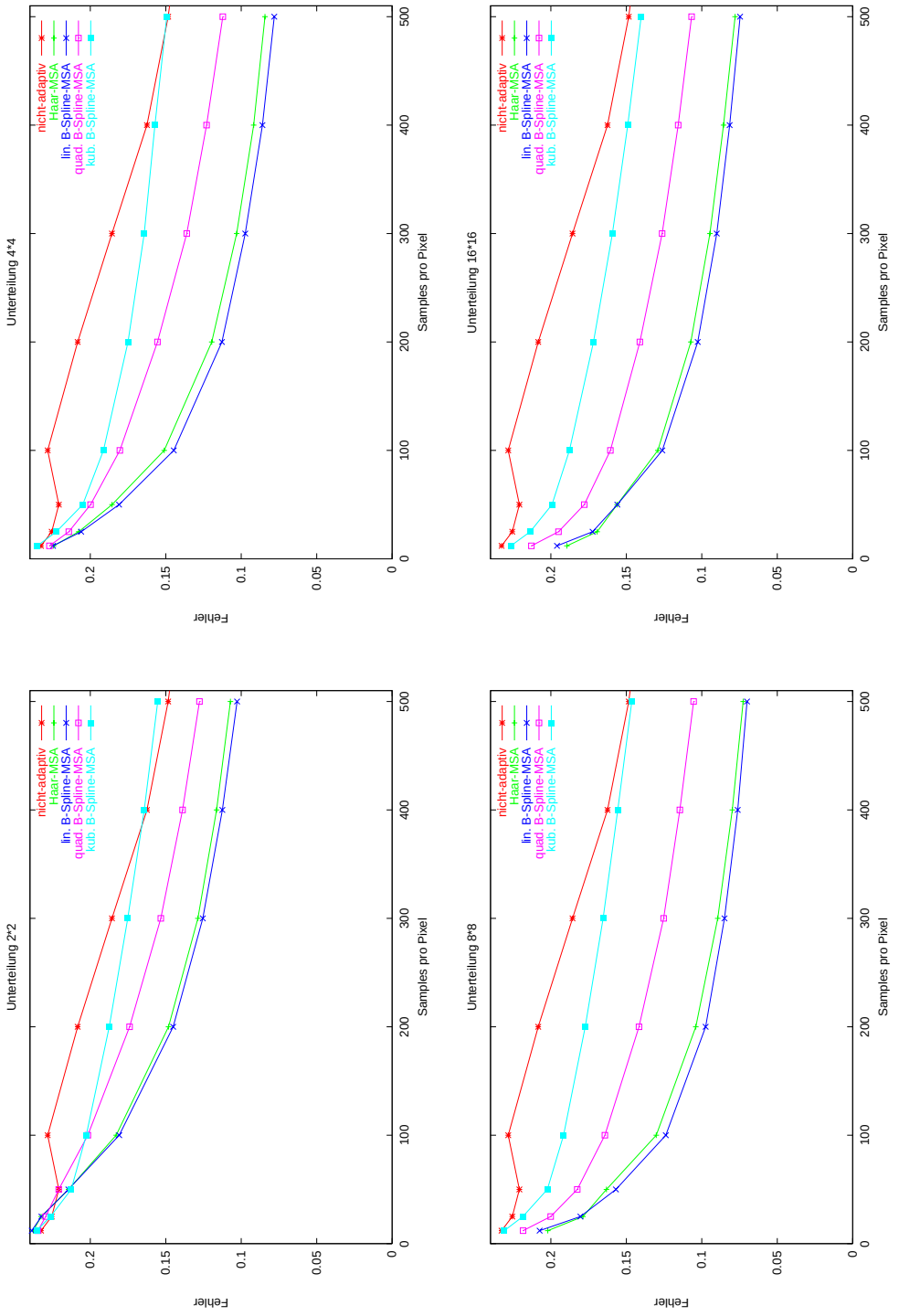
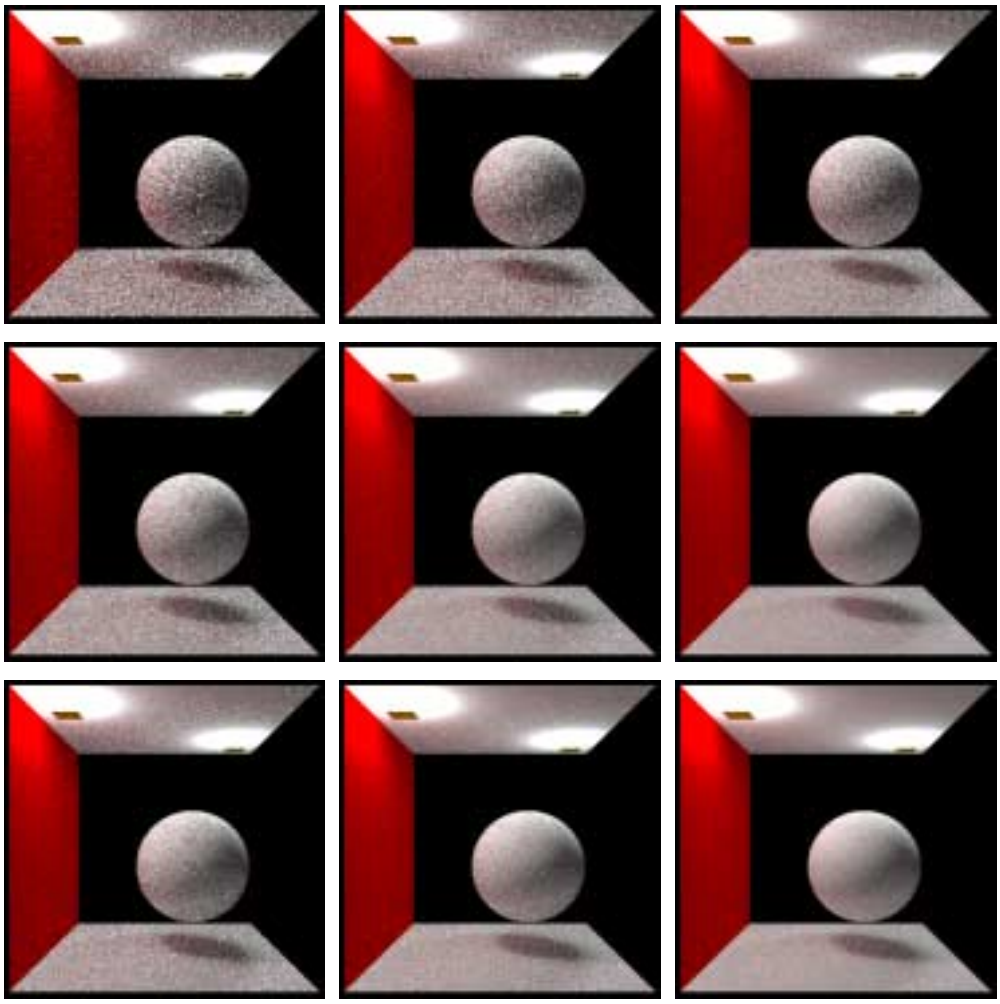
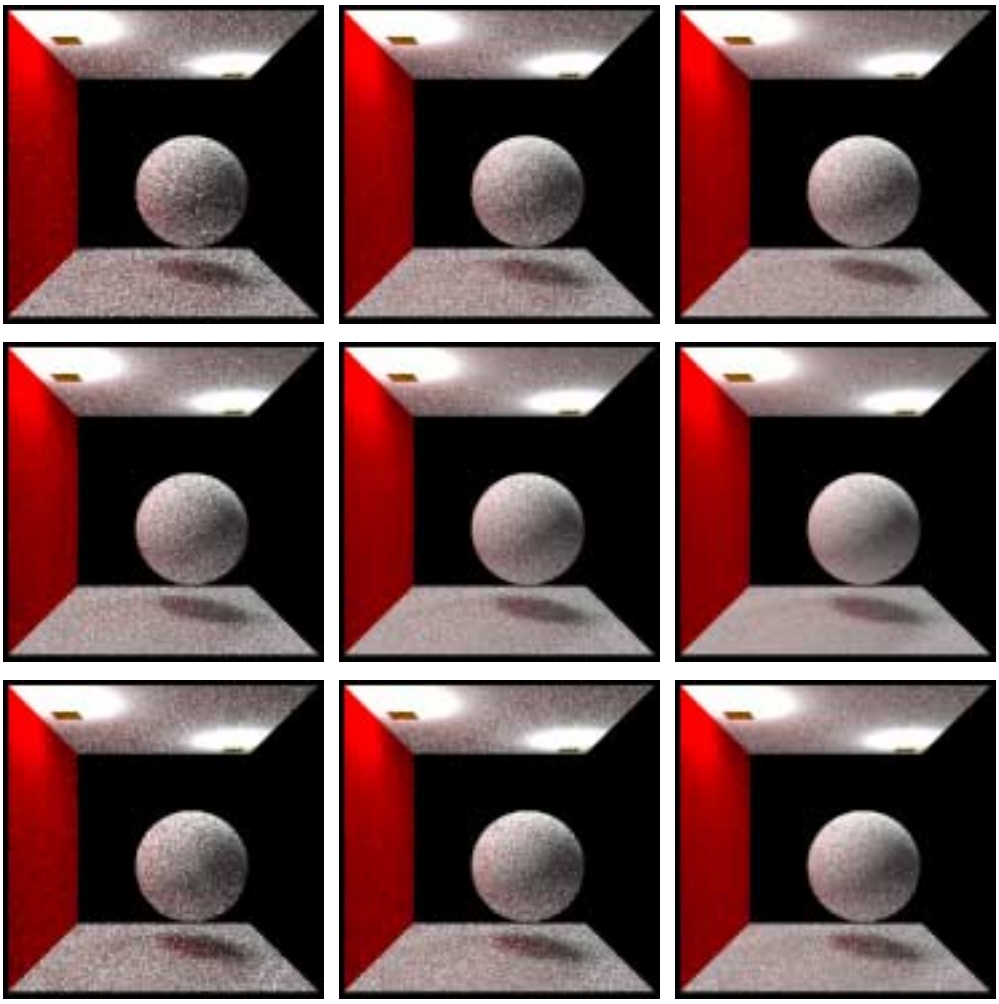



Abbildung E.46: Fehler als Funktion der Samplezahl, Szene 3, ohne Stratifikation



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.47: Bildqualität, Szene 3, mit Stratifikation, von oben nach unten: nicht-adaptiv, Haar-MSA, lin. B-Spline-MSA



100 Samples pro Pixel 200 Samples pro Pixel 400 Samples pro Pixel

Abbildung E.48: Bildqualität, Szene 3, mit Stratifikation, von oben nach unten: nicht-adaptiv, quad. B-Spline-MSA, kub. B-Spline-MSA

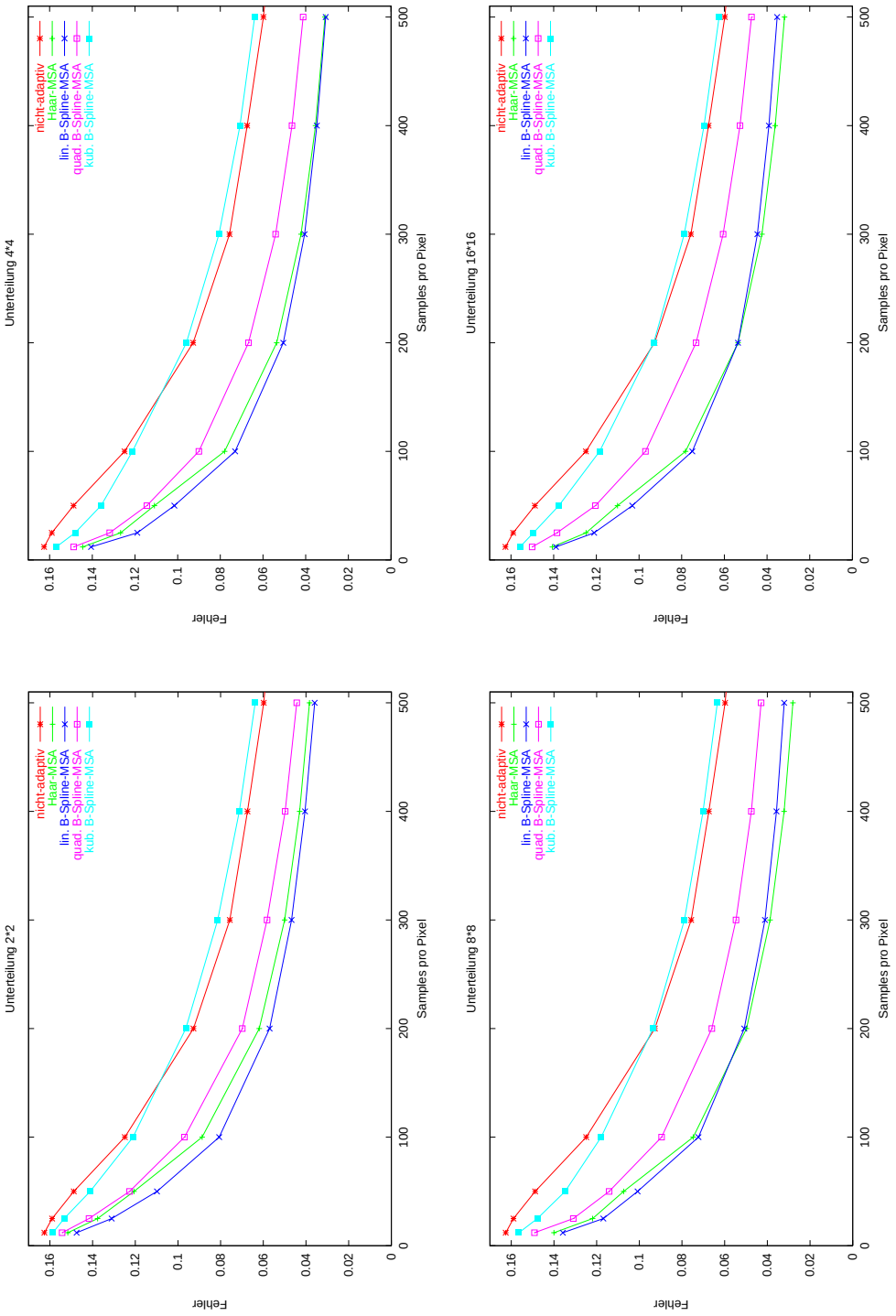



Abbildung E.49: Fehler als Funktion der Samplezahl, Szene 3, mit Stratifikation

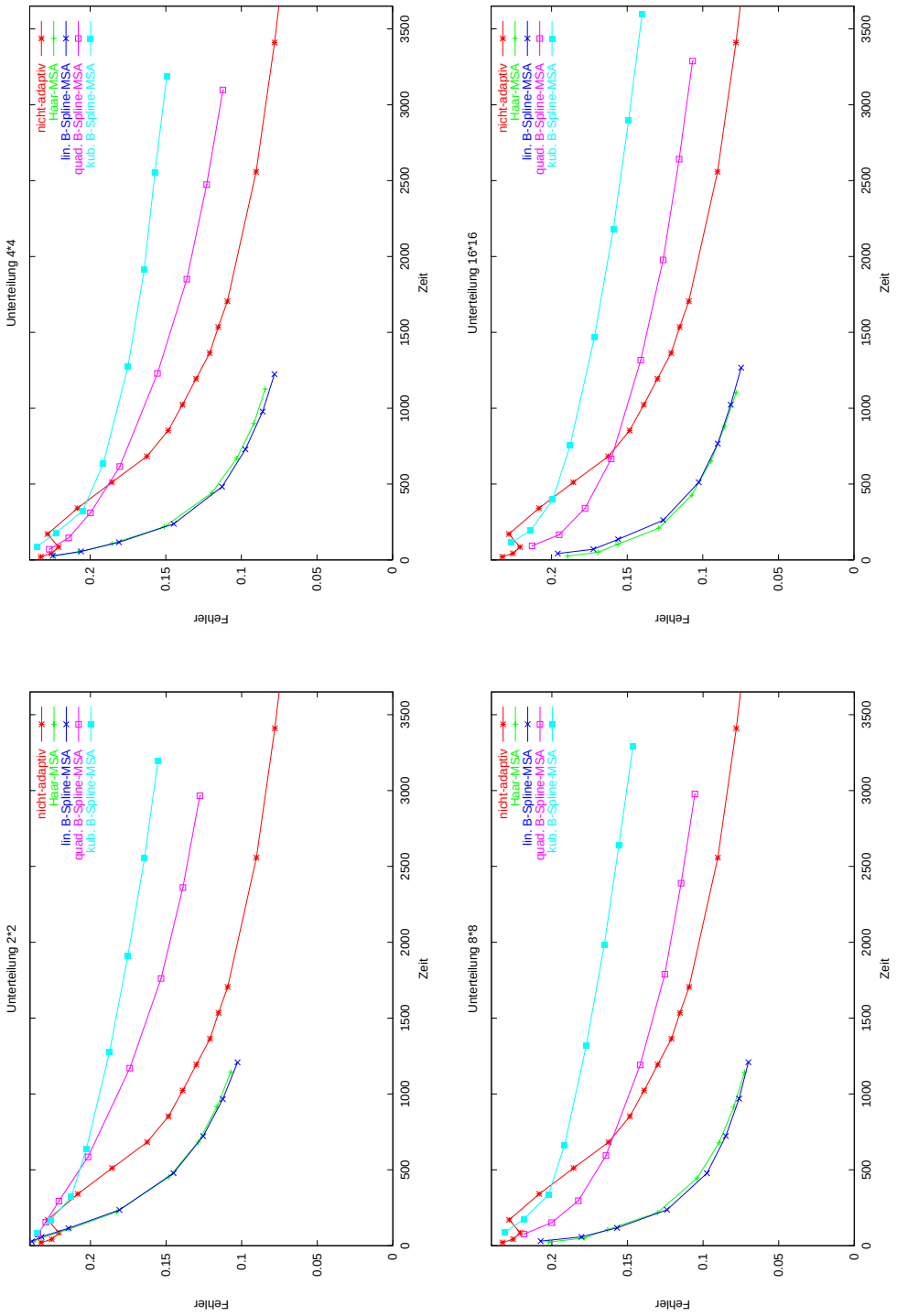



Abbildung E.50: Fehler als Funktion der Zeit, Szene 3, ohne Stratifikation

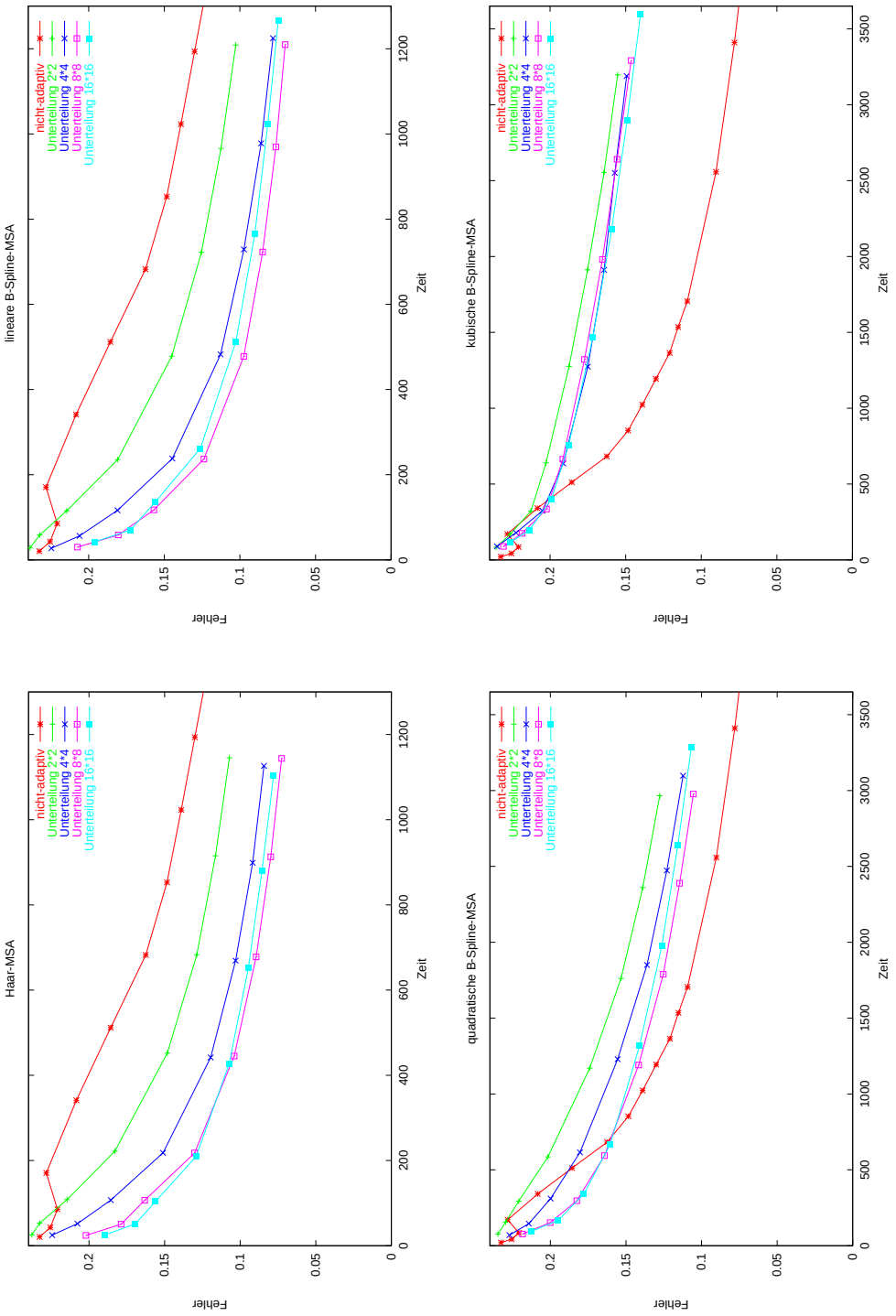



Abbildung E.51: Fehler als Funktion der Zeit, Szene 3, ohne Stratifikation

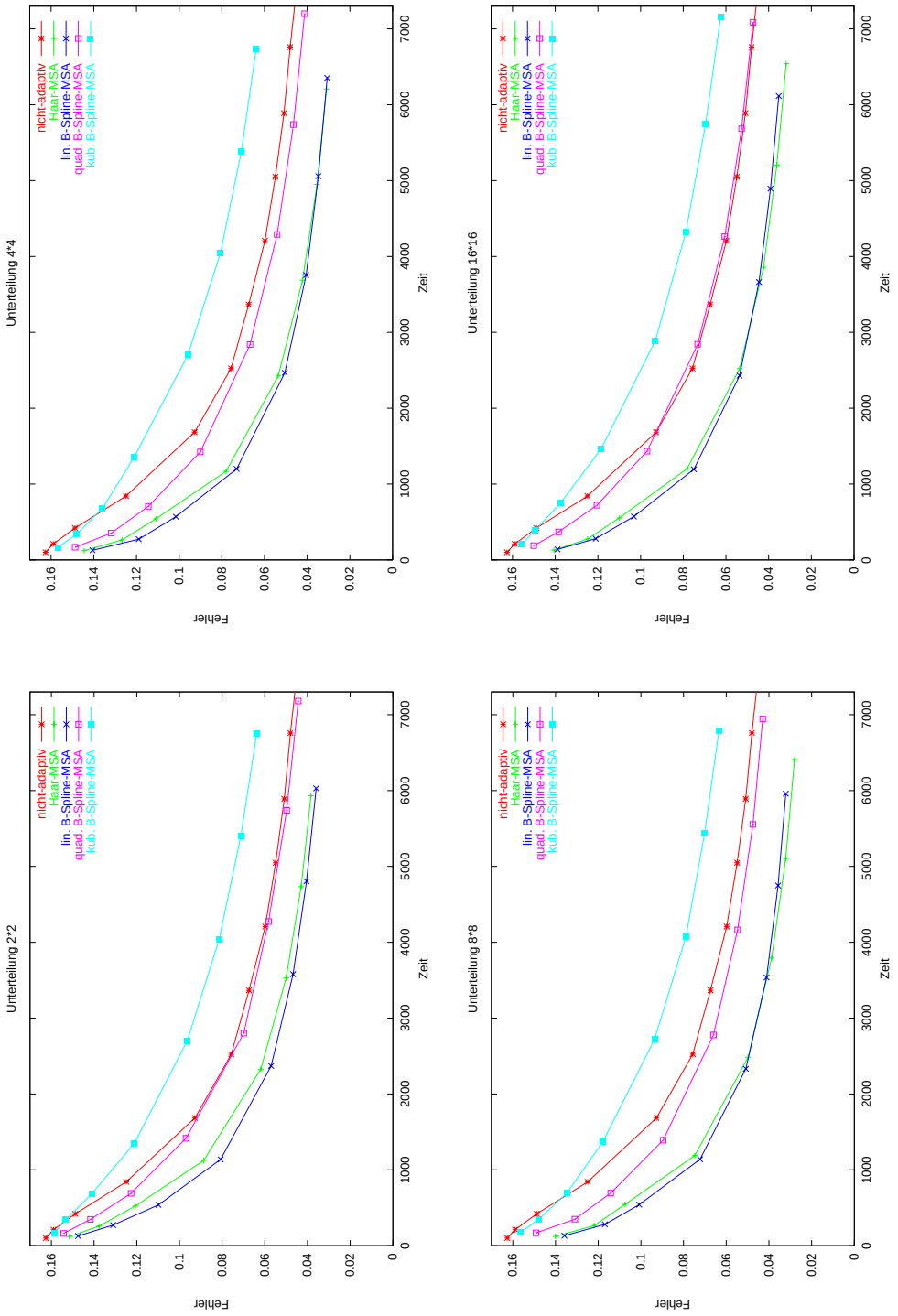



Abbildung E.52: Fehler als Funktion der Zeit, Szene 3, mit Stratifikation

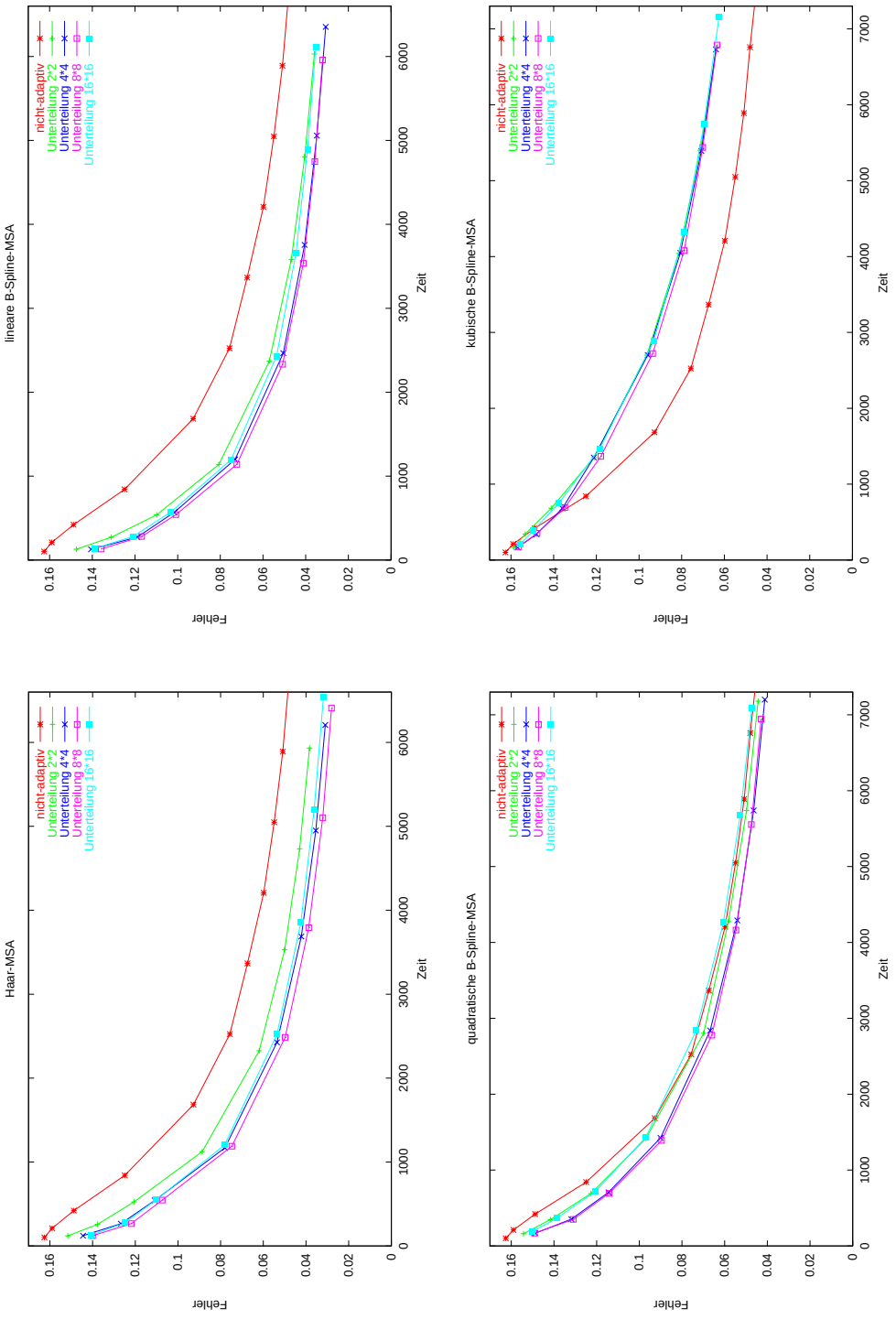



Abbildung E.53: Fehler als Funktion der Zeit, Szene 3, mit Stratifikation

# Anhang F

# Ergebnisse der Radiosity-Rekonstruktion

## F.1 Szene 1

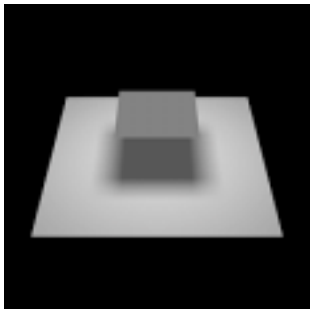


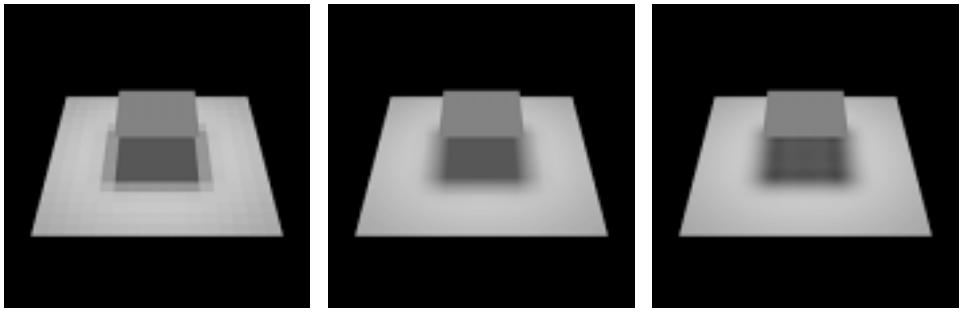
Abbildung F.1: Szene 1, Referenz

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0183 | 0.0327 | 0.223 |
| bilinear | 0.00464 | 0.00849 | 0.0394 |
| bikubisch | 0.00506 | 0.00979 | 0.0539 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0118 | 0.0186 | 0.0974 |
| Hardysche MQ | 0.00601 | 0.0105 | 0.056 |
| Natural-Neighbour | 0.00817 | 0.0163 | 0.0749 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0118 | 0.0187 | 0.0973 |
| Hardysche MQ | 0.00661 | 0.0108 | 0.0561 |
| Natural-Neighbour | 0.00477 | 0.00843 | 0.0391 |

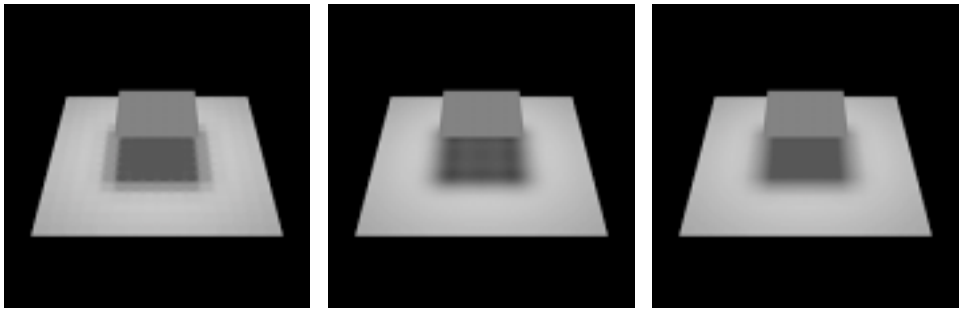
Tabelle F.1: Szene 1, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0227 | 0.0332 | 0.158 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0282 | 0.038 | 0.14 |
| Hardysche MQ | 0.0084 | 0.0131 | 0.0708 |
| Natural-Neighbour | 0.024 | 0.0304 | 0.0749 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0285 | 0.0381 | 0.14 |
| Hardysche MQ | 0.0101 | 0.0146 | 0.0708 |
| Natural-Neighbour | 0.019 | 0.0247 | 0.0749 |

Tabelle F.2: Szene 1, Fehler für Quadtree-Unterteilung

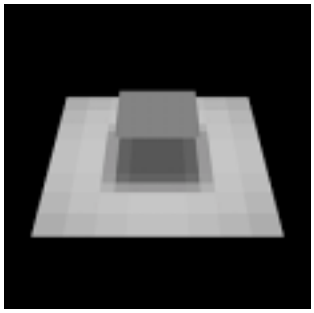


stückweise konstant | bilinear | bikubisch

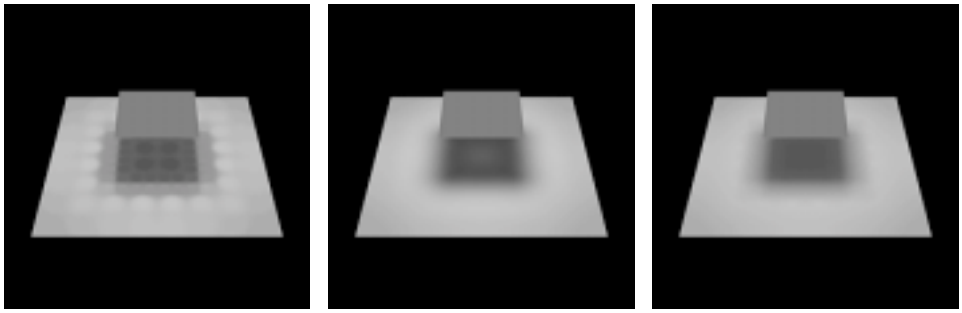


abstandsgewichtet | Hardysche MQ | Natural-Neighbour

Abbildung F.2: Szene 1, Gitter-Unterteilung



stückweise konstant



abstandsgewichtet | Hardysche MQ | Natural-Neighbour

Abbildung F.3: Szene 1, Quadtree-Unterteilung

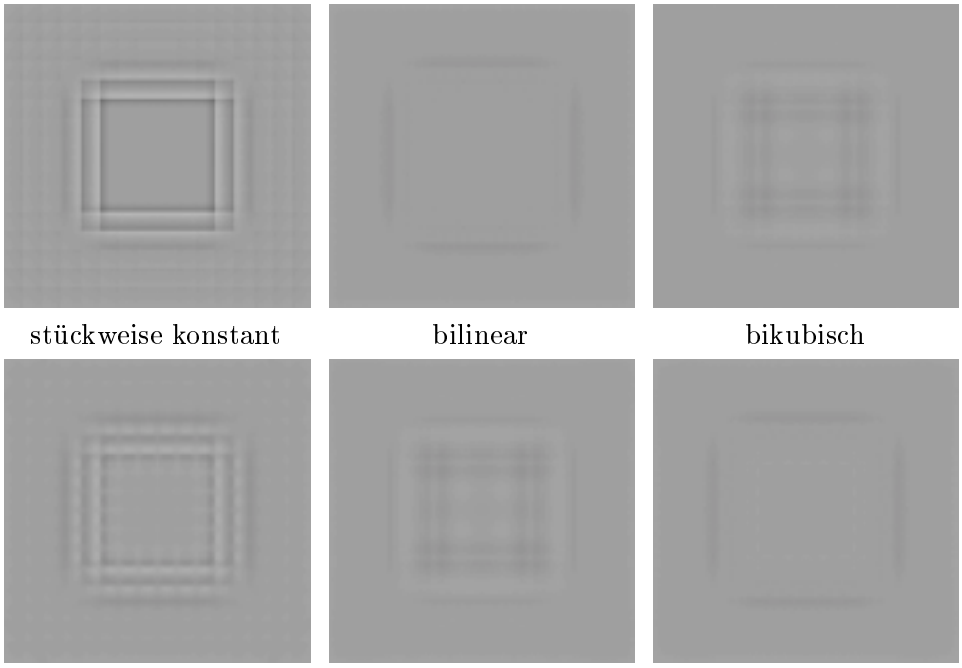



Abbildung F.4: Differenzbilder Szene 1, Gitter-Unterteilung

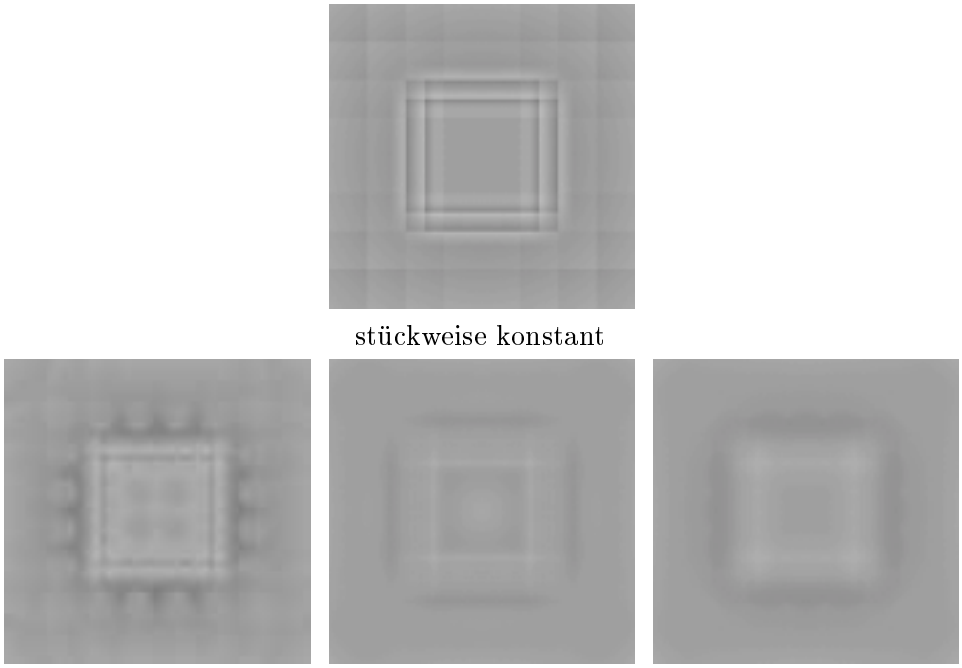



Abbildung F.5: Differenzbilder Szene 1, Quadtree-Unterteilung

## F.2 Szene 2

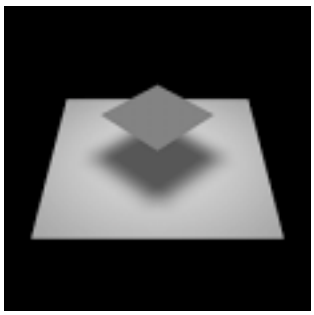


Abbildung F.6: Szene 2, Referenz

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0167 | 0.0308 | 0.189 |
| bilinear | 0.00628 | 0.0102 | 0.0429 |
| bikubisch | 0.00158 | 0.00319 | 0.0209 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0114 | 0.0189 | 0.105 |
| Hardysche MQ | 0.00212 | 0.00357 | 0.0241 |
| Natural-Neighbour | 0.01 | 0.0174 | 0.0749 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0114 | 0.0189 | 0.105 |
| Hardysche MQ | 0.00267 | 0.00434 | 0.0218 |
| Natural-Neighbour | 0.00663 | 0.0104 | 0.0449 |

Tabelle F.3: Szene 2, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0205 | 0.0312 | 0.272 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0237 | 0.0329 | 0.137 |
| Hardysche MQ | 0.00553 | 0.00767 | 0.0426 |
| Natural-Neighbour | 0.0187 | 0.0255 | 0.0719 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0241 | 0.0331 | 0.136 |
| Hardysche MQ | 0.00748 | 0.0101 | 0.0426 |
| Natural-Neighbour | 0.0137 | 0.0185 | 0.062 |

Tabelle F.4: Szene 2, Fehler für Quadtree-Unterteilung

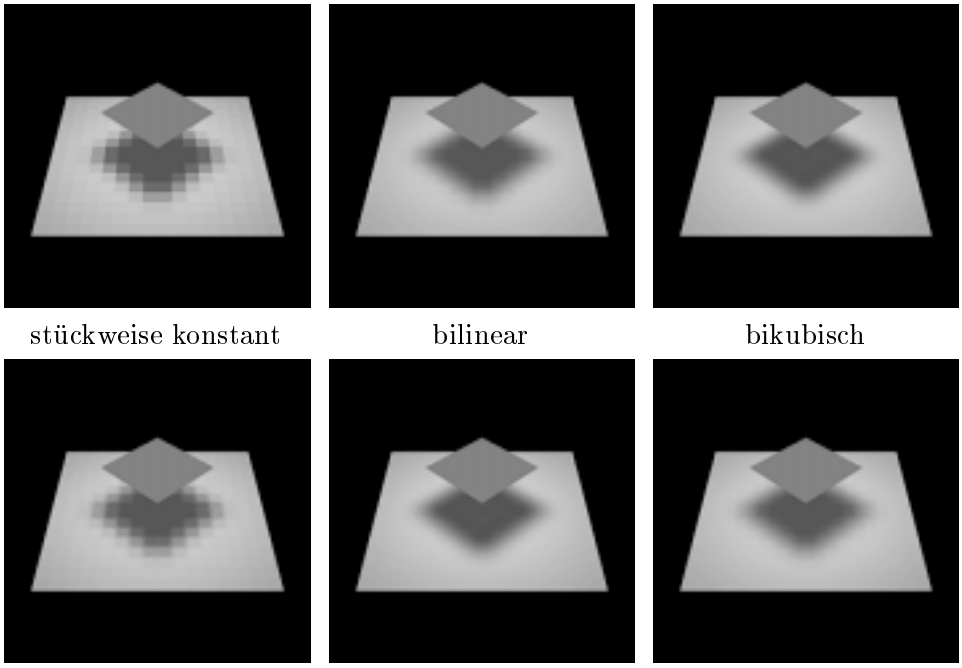



Abbildung F.7: Szene 2, Gitter-Unterteilung

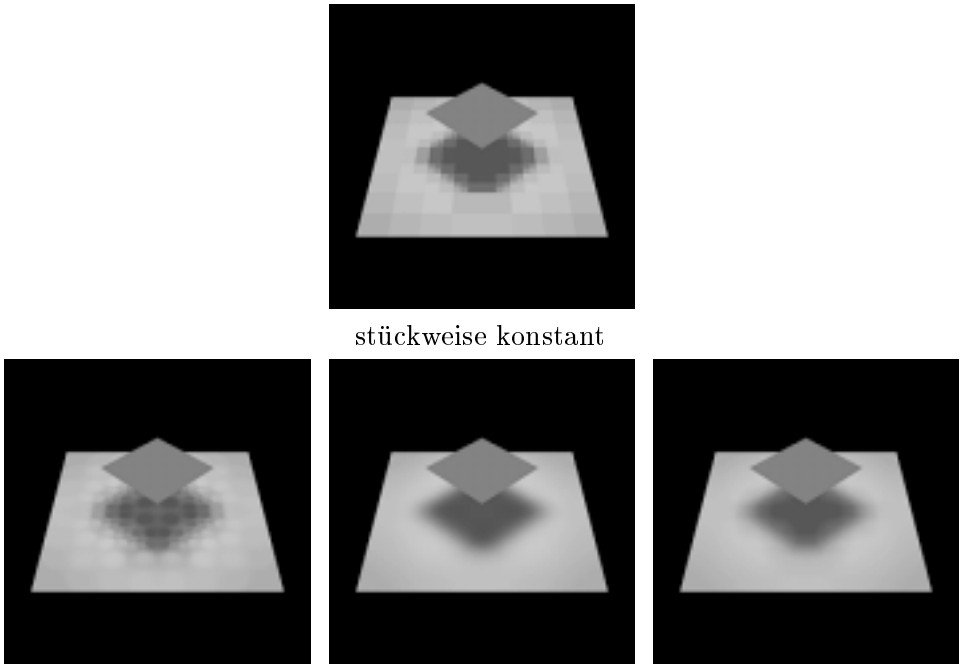



Abbildung F.8: Szene 2, Quadtree-Unterteilung

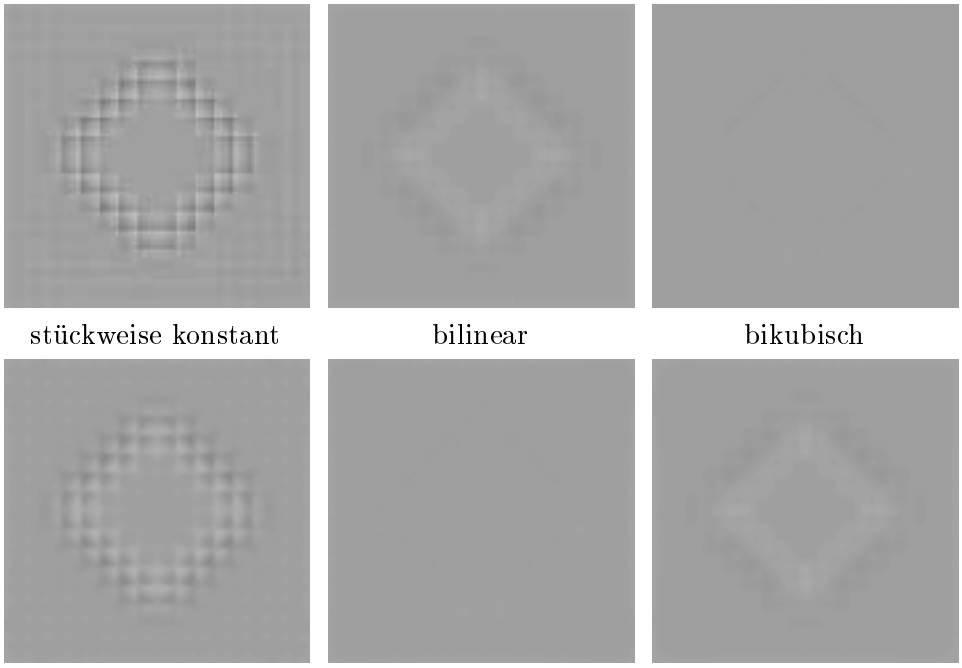



Abbildung F.9: Differenzbilder Szene 2, Gitter-Unterteilung

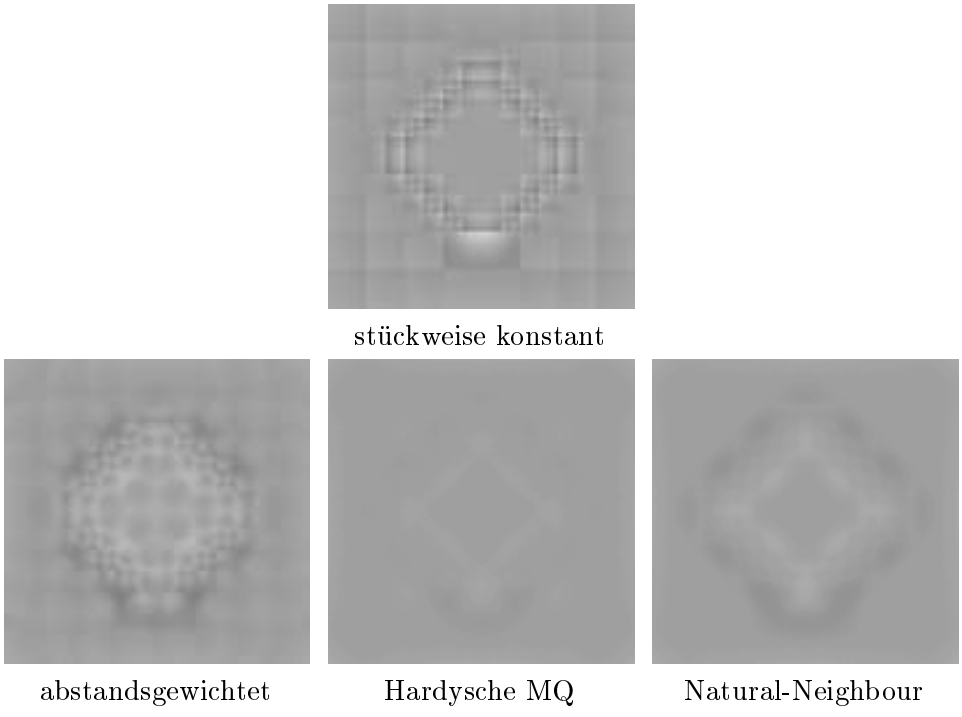



Abbildung F.10: Differenzbilder Szene 2, Quadtree-Unterteilung

## F.3 Szene 3a

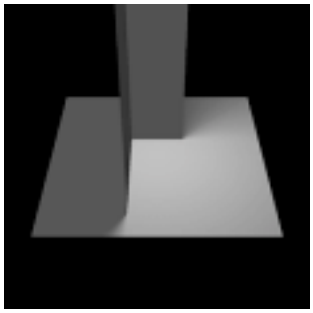


Abbildung F.11: Szene 3a, Referenz

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.00512 | 0.0146 | 0.3 |
| bilinear | 0.00738 | 0.029 | 0.321 |
| bikubisch | 0.0096 | 0.0287 | 0.32 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0071 | 0.0215 | 0.312 |
| Hardysche MQ | 0.0105 | 0.0291 | 0.319 |
| Natural-Neighbour | 0.0125 | 0.0366 | 0.323 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.00726 | 0.0217 | 0.312 |
| Hardysche MQ | 0.0108 | 0.0293 | 0.319 |
| Natural-Neighbour | 0.00803 | 0.0294 | 0.323 |

Tabelle F.5: Szene 3a, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.007 | 0.0119 | 0.152 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0417 | 0.0617 | 0.303 |
| Hardysche MQ | 0.0257 | 0.0463 | 0.316 |
| Natural-Neighbour | 0.0248 | 0.0479 | 0.325 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0359 | 0.058 | 0.304 |
| Hardysche MQ | 0.0223 | 0.0437 | 0.317 |
| Natural-Neighbour | 0.0141 | 0.0395 | 0.325 |

Tabelle F.6: Szene 3a, Fehler für Quadtree-Unterteilung

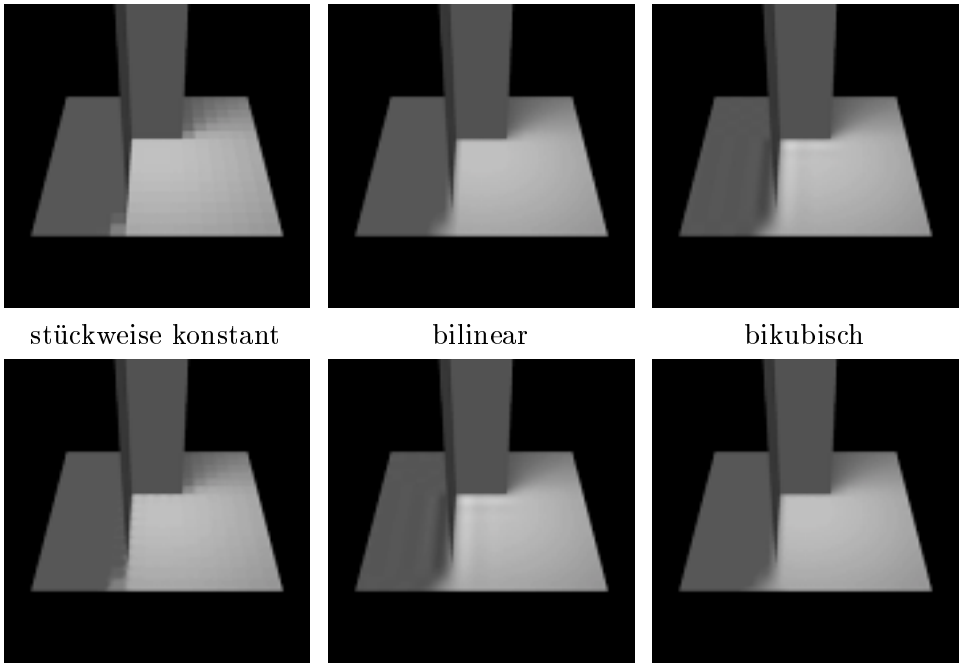



Abbildung F.12: Szene 3a, Gitter-Unterteilung

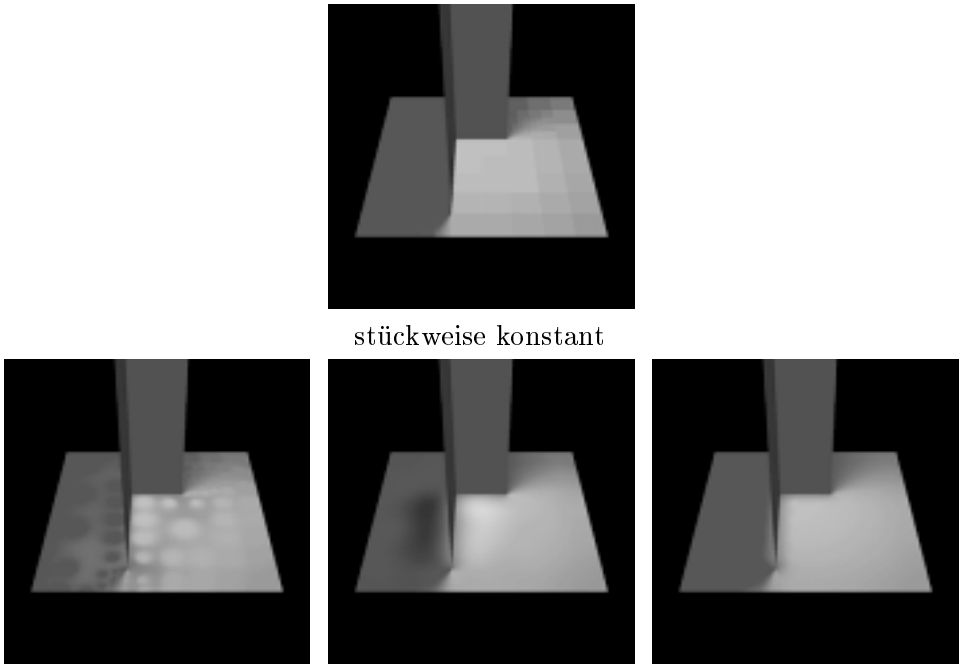



Abbildung F.13: Szene 3a, Quadtree-Unterteilung

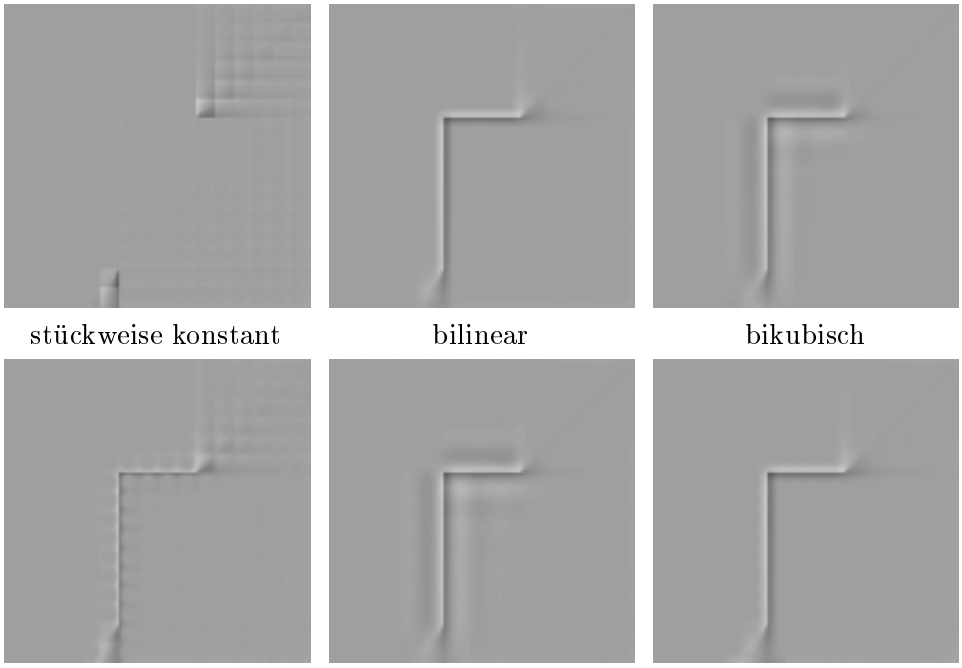



Abbildung F.14: Differenzbilder Szene 3a, Gitter-Unterteilung

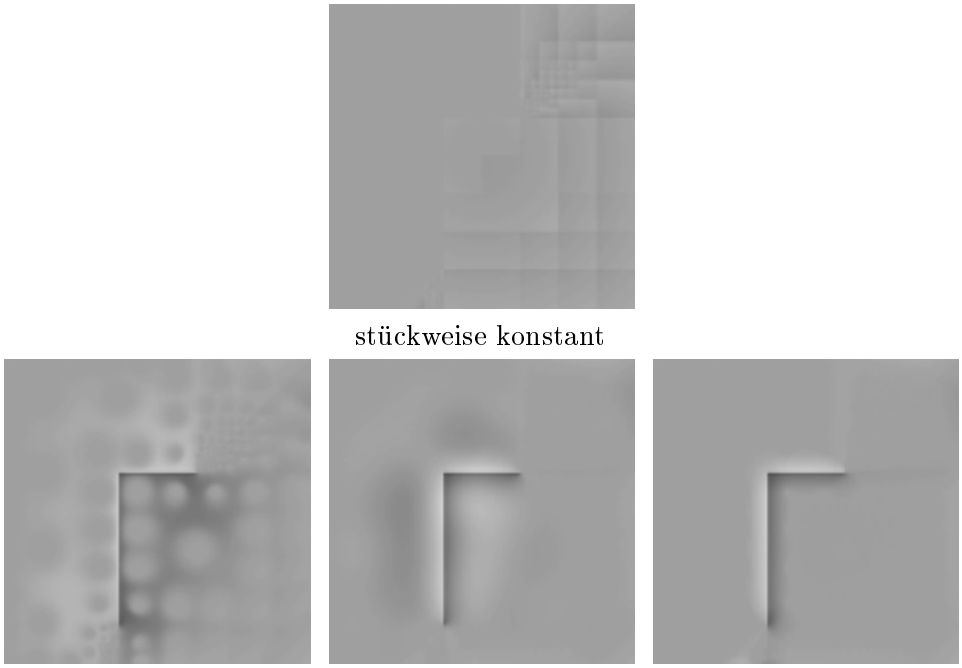



Abbildung F.15: Differenzbilder Szene 3a, Quadtree-Unterteilung

## F.4 Szene 3b

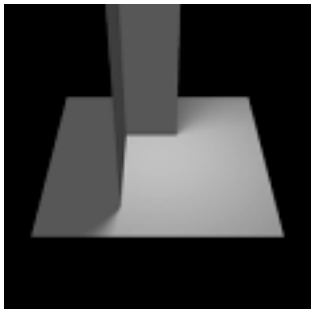


Abbildung F.16: Szene 3b, Referenz

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0153 | 0.0484 | 0.333 |
| bilinear | 0.0121 | 0.0385 | 0.314 |
| bikubisch | 0.0108 | 0.0347 | 0.311 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0143 | 0.0438 | 0.325 |
| Hardysche MQ | 0.0109 | 0.0344 | 0.311 |
| Natural-Neighbour | 0.0169 | 0.0442 | 0.315 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0143 | 0.0438 | 0.325 |
| Hardysche MQ | 0.0113 | 0.0345 | 0.311 |
| Natural-Neighbour | 0.0125 | 0.0386 | 0.315 |

Tabelle F.7: Szene 3b, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.00793 | 0.0155 | 0.329 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0407 | 0.0558 | 0.273 |
| Hardysche MQ | 0.0244 | 0.0413 | 0.265 |
| Natural-Neighbour | 0.0158 | 0.0296 | 0.291 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0327 | 0.0482 | 0.273 |
| Hardysche MQ | 0.0225 | 0.0399 | 0.265 |
| Natural-Neighbour | 0.00626 | 0.0182 | 0.291 |

Tabelle F.8: Szene 3b, Fehler für Quadtree-Unterteilung

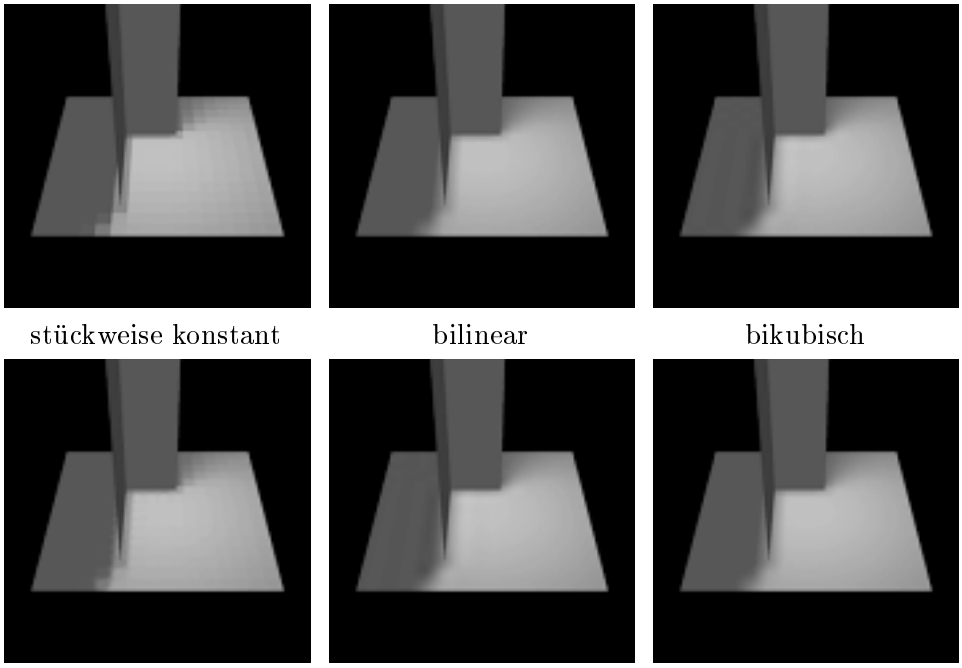



Abbildung F.17: Szene 3b, Gitter-Unterteilung

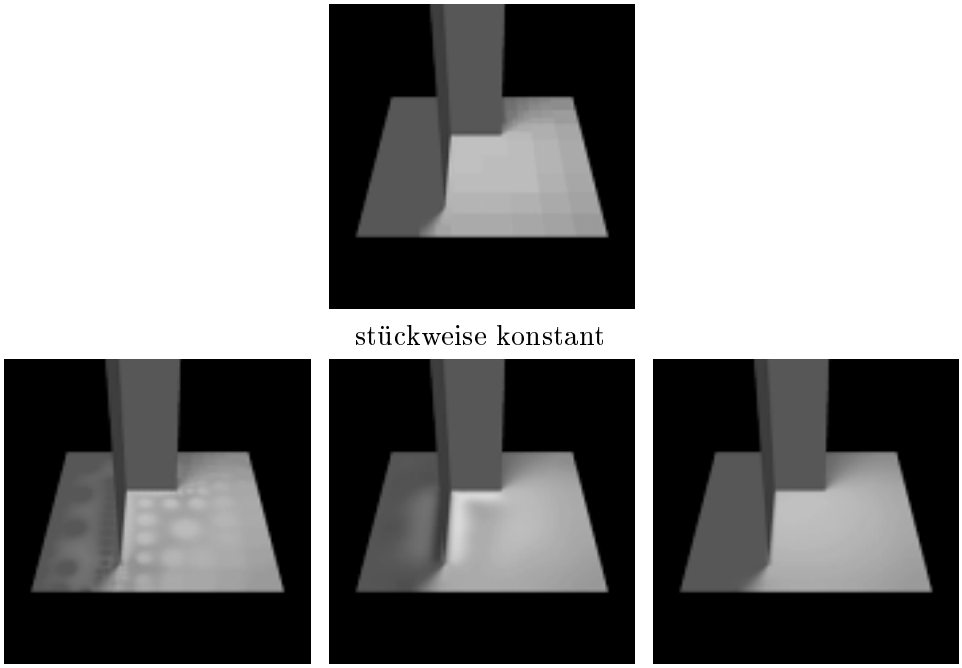



Abbildung F.18: Szene 3b, Quadtree-Unterteilung

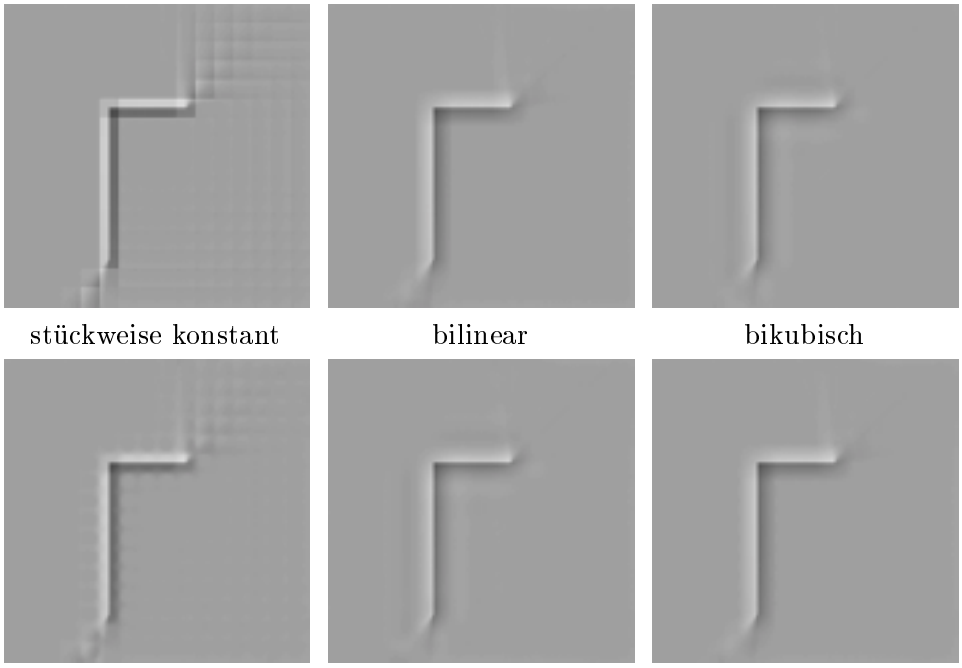



Abbildung F.19: Differenzbilder Szene 3b, Gitter-Unterteilung

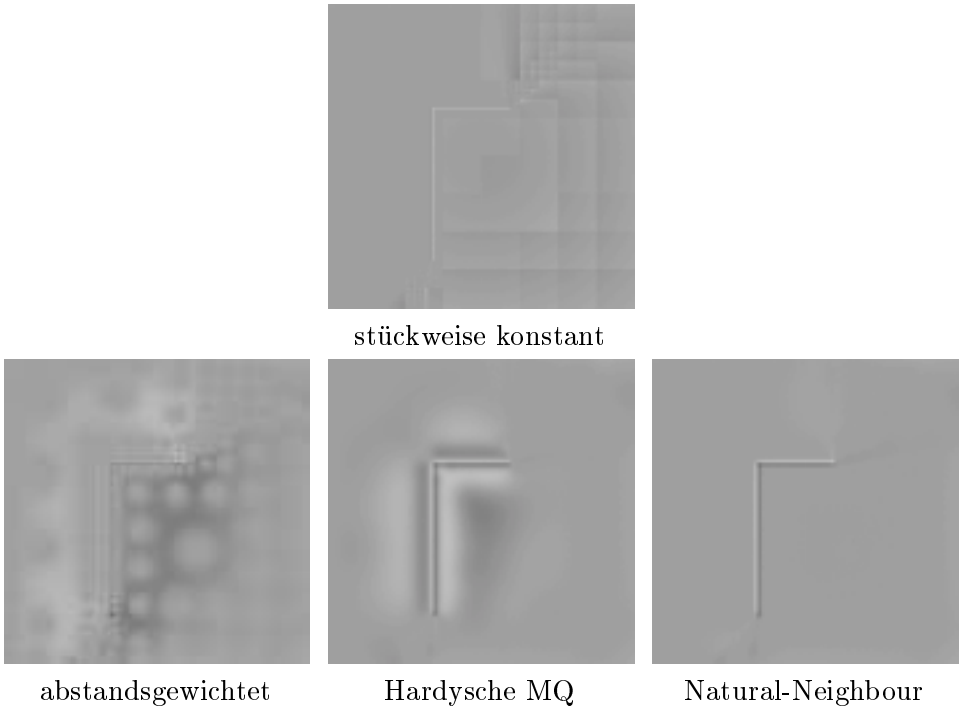



Abbildung F.20: Differenzbilder Szene 3b, Quadtree-Unterteilung

## F.5 Szene 4a

Abbildung F.21: Szene 4a, Referenz

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0124 | 0.0463 | 0.332 |
| bilinear | 0.0105 | 0.037 | 0.319 |
| bikubisch | 0.00882 | 0.0343 | 0.32 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0121 | 0.0409 | 0.332 |
| Hardysche MQ | 0.00904 | 0.0344 | 0.321 |
| Natural-Neighbour | 0.0152 | 0.0413 | 0.314 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0122 | 0.041 | 0.332 |
| Hardysche MQ | 0.00949 | 0.0345 | 0.321 |
| Natural-Neighbour | 0.011 | 0.0369 | 0.314 |

Tabelle F.9: Szene 4a, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.00746 | 0.0139 | 0.277 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0812 | 0.107 | 0.291 |
| Hardysche MQ | 0.00672 | 0.012 | 0.176 |
| Natural-Neighbour | 0.0103 | 0.0183 | 0.251 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0669 | 0.0939 | 0.291 |
| Hardysche MQ | 0.00554 | 0.011 | 0.176 |
| Natural-Neighbour | 0.00511 | 0.0102 | 0.251 |

Tabelle F.10: Szene 4a, Fehler für Quadtree-Unterteilung

stückweise konstant | bilinear | bikubisch

abstandsgewichtet | Hardysche MQ | Natural-Neighbour

Abbildung F.22: Szene 4a, Gitter-Unterteilung

stückweise konstant

abstandsgewichtet | Hardysche MQ | Natural-Neighbour

Abbildung F.23: Szene 4a, Quadtree-Unterteilung

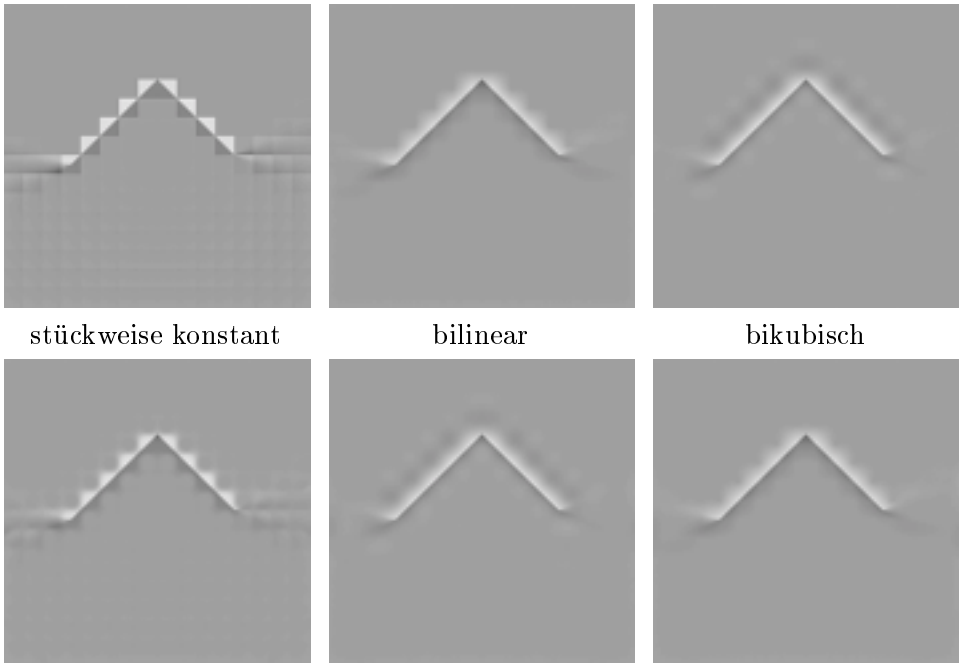



Abbildung F.24: Differenzbilder Szene 4a, Gitter-Unterteilung

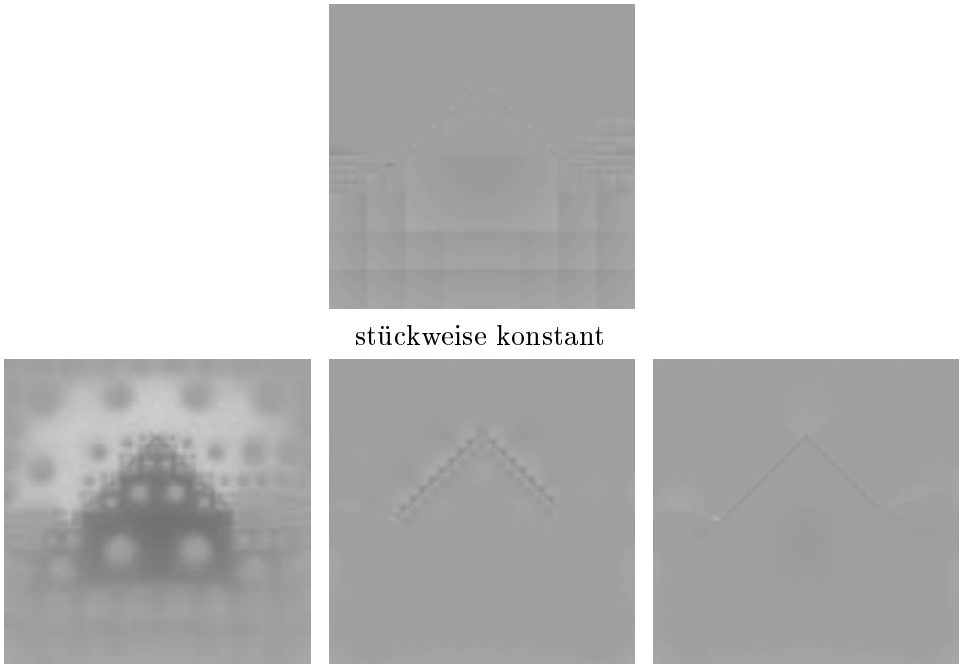



Abbildung F.25: Differenzbilder Szene 4a, Quadtree-Unterteilung

## F.6 Szene 4b

Abbildung F.26: Szene 4b

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.0119 | 0.0417 | 0.441 |
| bilinear | 0.0102 | 0.0345 | 0.314 |
| bikubisch | 0.00834 | 0.0309 | 0.311 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0113 | 0.035 | 0.389 |
| Hardysche MQ | 0.00855 | 0.0309 | 0.308 |
| Natural-Neighbour | 0.0152 | 0.0399 | 0.305 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0114 | 0.0351 | 0.389 |
| Hardysche MQ | 0.00906 | 0.031 | 0.307 |
| Natural-Neighbour | 0.0108 | 0.0346 | 0.305 |

Tabelle F.11: Szene 4b, Fehler für Gitter-Unterteilung

| Interpolationsverfahren | $L^1$ | $L^2$ | $L^\infty$ |
|---|---|---|---|
| stückweise konstant | 0.00968 | 0.0233 | 0.422 |
| ohne Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0732 | 0.1 | 0.314 |
| Hardysche MQ | 0.00872 | 0.0199 | 0.302 |
| Natural-Neighbour | 0.015 | 0.0292 | 0.273 |
| mit Rand-Stützstellen | | | |
| abstandsgewichtet | 0.0589 | 0.0851 | 0.314 |
| Hardysche MQ | 0.00809 | 0.0191 | 0.302 |
| Natural-Neighbour | 0.00901 | 0.0215 | 0.273 |

Tabelle F.12: Szene 4b, Fehler für Quadtree-Unterteilung

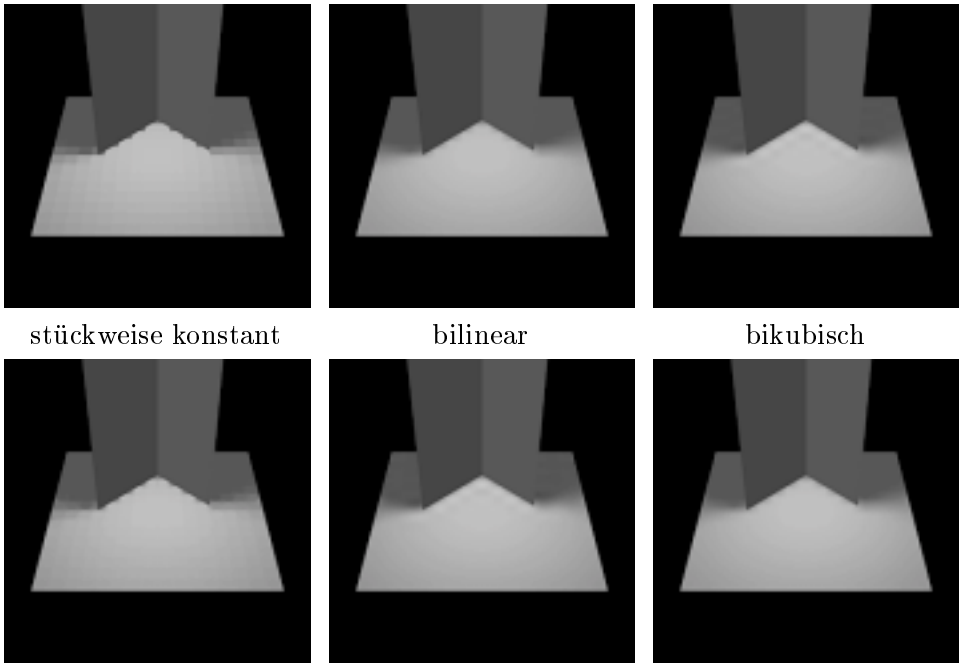



Abbildung F.27: Szene 4b, Gitter-Unterteilung

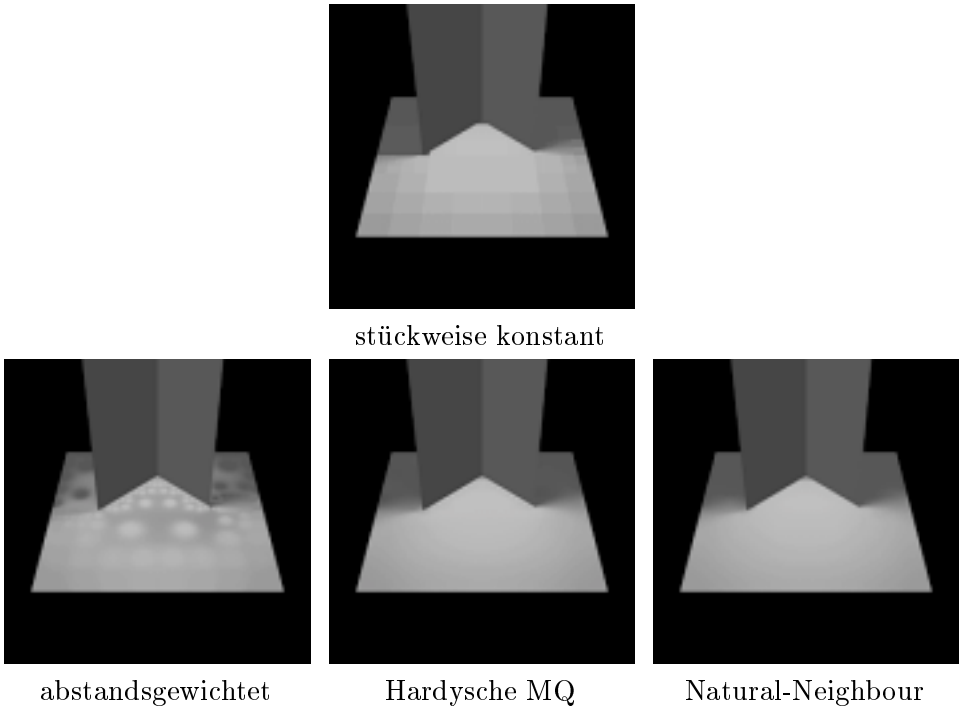



Abbildung F.28: Szene 4b, Quadtree-Unterteilung

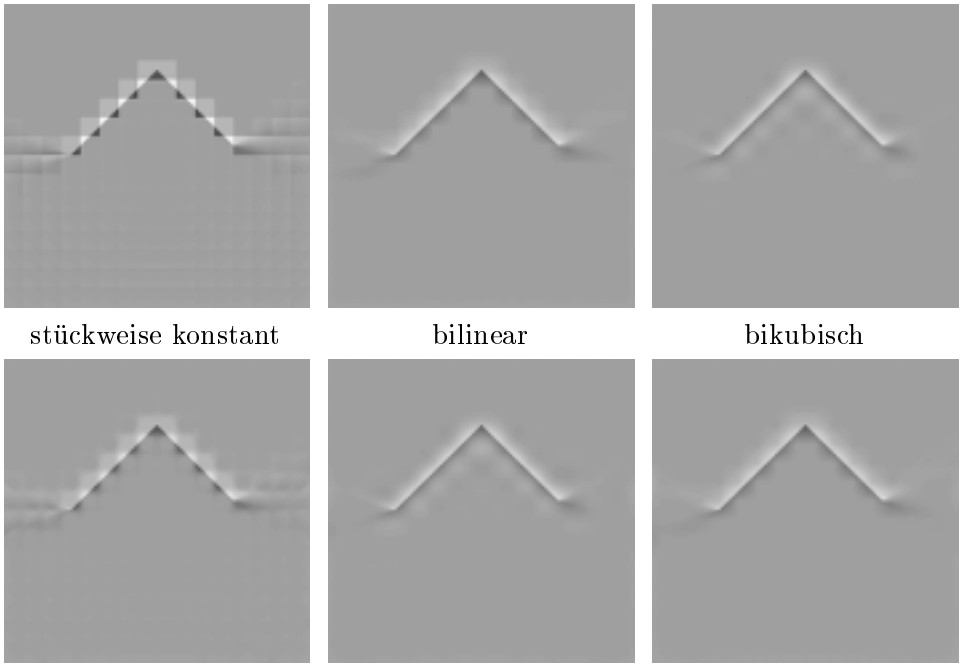



Abbildung F.29: Differenzbilder Szene 4b, Gitter-Unterteilung

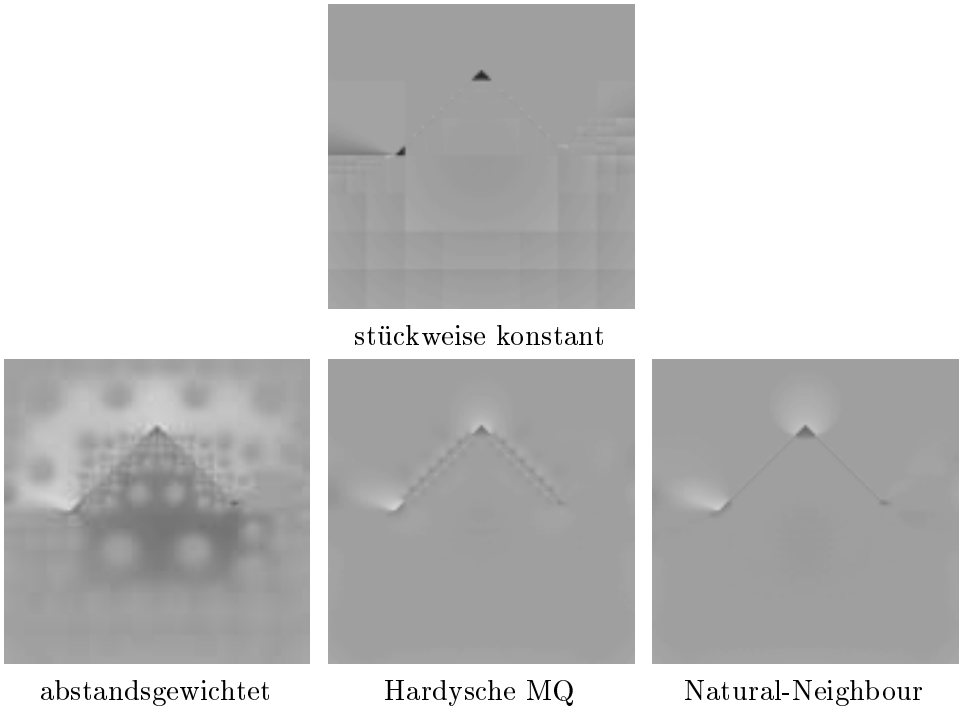
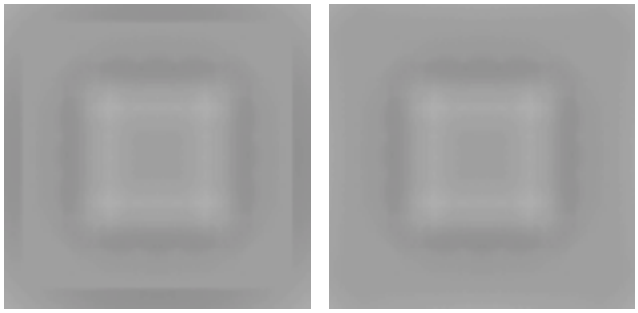
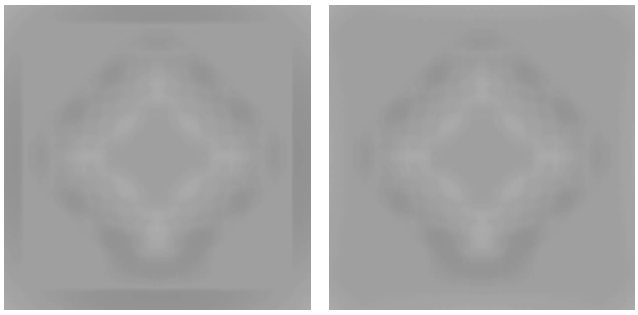



Abbildung F.30: Differenzbilder Szene 4b, Quadtree-Unterteilung

## F.7 Einfügen zusätzlicher Rand-Stützstellen



Szene 1



Szene 2

Szene 3a

Abbildung F.31: Einfügen zusätzlicher Rand-Stützstellen, links: ohne Rand-Stützstellen, rechts: mit Rand-Stützstellen

Szene 3b

Szene 4a

Szene 4b

Abbildung F.32: Einfügen zusätzlicher Rand-Stützstellen, links: ohne Rand-Stützstellen, rechts: mit Rand-Stützstellen

# Literaturverzeichnis


[1] Arvo, J. und D. Kirk: *Particle Transport and Image Synthesis.*
Computer Graphics, 24(4):63–66, August 1990.

[2] Balzert, H.: *Lehrbuch der Software-Technik.*
Spektrum Akademischer Verlag, 1996.

[3] Bastos, R., M. Goslin und H. Zhang: *Efficient Rendering of Radiosity Using Textures and Bicubic Reconstruction.*
Technischer Bericht Dept. of CS, U. of North Carolina at Chapel Hill, 1996. TR 96-025.

[4] Bastos, R. M., A. A. de Sousa und F. N. Ferreira: *Reconstruction of Illumination Functions using Bicubic Hermite Interpolation.*
In: *Proceedings of Fourth Eurographics Workshop on Rendering*, Seiten 317–326, 1993.

[5] Booch, G., J. Rumbaugh und I. Jacobson: *The Unified Modeling Language User Guide.*
Addison-Wesley, 1998.

[6] Bronstein, I. N. und K. A. Semendjajew: *Taschenbuch der Mathematik.*
Verlag Harri Deutsch, Thun, Frankfurt/Main, 1987.

[7] Bustillo, E.: *A Neuro-Evolutionary Unbiased Global Illumination Algorithm.*
In: Dorsey, Julie und Philipp Slusallek (Herausgeber): *Eurographics Rendering Workshop 1997*, Seiten 263–274, New York City, NY, Juni 1997. Eurographics, Springer Wien.

[8] Chui, C. K.: *An Introduction to Wavelets.*
Academic Press, 1992.

[9] Chui, C. K. und J. Z. Wang: *A General Framework of Compactly Supported Splines and Wavelets.*
J. Approx. Theory, 71(3):263–304, 1992.

[10] Chui, C. K. und J. Z. Wang: *On Compactly Supported Spline Wavelets and a Duality Principle.*
Trans. Amer. Math. Soc., 330:903–915, 1992.

[11] Cohen, A., I. Daubechies, B. Jawerth und P. Vial: *Multiresolution analysis, wavelets and fast algorithms on an interval.*
C. R. Acad. Sci. Paris, t. 316, Serie I, Seiten 417–421, 1993.

[12] Cohen, M. F., S. E. Chen, J. R. Wallace und D. P. Greenberg: *A Progressive Refinement Approach to Fast Radiosity Image Generation.* Computer Graphics, 22(4):75–84, August 1988.

[13] Cohen, M. F. und D. P. Greenberg: *The Hemi-Cube: A Radiosity Solution for Complex Environments.* In: *Computer Graphics (ACM SIGGRAPH '85 Proceedings)*, Band 19, Seiten 31–40, August 1985.

[14] Cohen, M. F., D. P. Greenberg, D. S. Immel und P. J. Brock: *An Efficient Radiosity Approach for Realistic Image Synthesis.* IEEE Computer Graphics and Applications, 6(3):26–35, März 1986.

[15] Cohen, M. F. und J. R. Wallace: *Radiosity and Realistic Image Synthesis.* Academic Press Professional, Boston, MA, 1993.

[16] Cook, R. L., T. Porter und L. Carpenter: *Distributed Ray Tracing.* Computer Graphics, 18(3):137–145, Juli 1984.

[17] Daubechies, I.: *Orthonormal Bases of Compactly Supported Wavelets.* Comm. Pure Applied Math., XLI(41):909–996, November 1988.

[18] DeMarco, T.: *Structured Analysis and System Specification.* Englewood Cliffs: Yourdon Press, 1978.

[19] Dutré, P. und Y. D. Willems: *Importance-Driven Monte Carlo Light Tracing.* In: *Fifth Eurographics Workshop on Rendering*, Seiten 185–194, Darmstadt, Germany, Juni 1994.

[20] Dutré, P. und Y. D. Willems: *Potential-Driven Monte Carlo Particle Tracing for Diffuse Environments with Adaptive Probability Density Functions.* In: Hanrahan, P. M. und W. Purgathofer (Herausgeber): *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, Seiten 306–315, New York, NY, 1995. Springer-Verlag.

[21] Eck, M.: *Interpolationsmethoden zur Rekonstruktion von 3D-Oberflächen aus ebenen Schnittfolgen.* CAD und Computergraphik, 13(5):109–120, Februar 1991.

[22] Ermakov, S. M.: *Die Monte-Carlo-Methode und verwandte Fragen.* Oldenbourg, München, 1975.

[23] Farin, G.: *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, San Diego, Dritte Auflage, 1993.

[24] Fellner, W. D.: *Computer Grafik.* BI Wissenschaftsverlag, 1988.

[25] Gamma, E., R. Helm, R. Johnson und J. Vlissides: *Design Patterns, Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994.

[26] Garmann, R., R. Mencl und G. Pietrek: *MoCaRT++ Ein objektorientiertes Programmsystem für Monte-Carlo-Bildsynthese.*
it+ti Informationstechnik und Technische Informatik, 40(1):12–19, 1998.

[27] Glassner, A. S.: *An Introduction to Ray Tracing.*
Academic Press, 1989.

[28] Glassner, A. S.: *Principles of Digital Image Synthesis.*
Morgan Kaufmann, San Francisco, CA, 1995.

[29] Goral, C. M., K. K. Torrance, D. P. Greenberg und B. Battaile: *Modelling the Interaction of Light Between Diffuse Surfaces.*
Computer Graphics, 18(3):213–222, Juli 1984.

[30] Hammersley, J. M. und D. C. Handscomb: *Monte Carlo Methods.*
Methuen, London, 1967.

[31] Hanrahan, P., D. Salzman und L. Aupperle: *A rapid hierarchical radiosity algorithm.*
In: Sederberg, Thomas W. (Herausgeber): *Computer Graphics (SIGGRAPH '91 Proceedings)*, Band 25, Seiten 197–206, Juli 1991.

[32] Hardy, R. L.: *Multiquadric equations of topography and other irregular surfaces.*
Journal of Geophysical Research, 76(8):1905–1915, 1971.

[33] Hardy, R. L. und W. M. Göpfert: *Least squares prediction of gravity anomalies, geodial undulations and deflections of the vertical with multiquadric harmonic functions.*
Geophysical Research Letters, 2:423–426, 1975.

[34] Heal, K. M., M. L. Hansen und K. M. Rickard: *Maple V Learning Guide.*
Springer-Verlag, Berlin, Germany / Heidelberg, Germany / London, UK / etc., 1996.
With the editorial assistance of J. S. Devitt. Based in part on the work of B. W. Char.

[35] Hengartner, W. und R. Theodorescu: *Einführung in die Monte-Carlo-Methode.*
Hanser, München, 1978.

[36] Henricson, M. und E. Nyquist: *Industrial Strength C++, Rules and Recommendations.*
Prentice Hall, 1996.

[37] Hinkenjann, A. und G. Pietrek: *Reconstructing Radiosity by Scattered Data Interpolation.*
In: *Proceedings of WSCG'98, Sixth International Conference in Central Europe on Computer Graphics and Visualization*, Februar 1998.
held at University of West Bohemia, Plzen, Czech Republic, 9-13 February 1998.

[38] Hinkenjann, A. und G. Pietrek: *Using Scattered Data Interpolation for Radiosity Reconstruction.*

In: Wolter, Franz-Erich und Nicholas M. Patrikalakis (Herausgeber): *Proceedings of the Conference on Computer Graphics International 1998 (CGI-98)*, Seiten 715–721, Los Alamitos, California, Juni 22–26 1998. IEEE Computer Society.

[39] Jensen, H. W.: *Importance Driven Path Tracing Using the Photon Map.* In: Hanrahan, P. M. und W. Purgathofer (Herausgeber): *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, Seiten 326–335. Springer-Verlag, New York, NY, 1995.

[40] Jensen, H. W.: *Global Illumination Using Photon Maps.* In: *Proceedings of the Seventh Eurographics Workshop on Rendering*, Porto, Portugal, Juni 1996.

[41] Kajiya, J. T.: *The Rendering Equation.* Computer Graphics, 20(4):143–150, August 1986.

[42] Kalos, M. und P. Whitlock: *Monte Carlo Methods.* Wiley & Sons, New York, 1986.

[43] Keller, A.: *Quasi-Monte Carlo Radiosity.* In: Pueyo, Xavier und Peter Schröder (Herausgeber): *Eurographics Rendering Workshop 1996*, Seiten 101–110, New York City, NY, Juni 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.

[44] Keller, A.: *Instant Radiosity.* Computer Graphics, 31(Annual Conference Series):49–56, August 1997.

[45] Keller, A.: *Quasi-Monte Carlo Methods for Photorealistic Image Synthesis.* Doktorarbeit, Universität Kaiserlautern, Juni 1997. ISBN 3-8265-3330-5.

[46] Kobbelt, L., M. Stamminger und H-P. Seidel: *Using Subdivision on Hierarchical Data to Reconstruct Radiosity Distribution.* Computer Graphics Forum, 16(3):347–356, August 1997. Proceedings of Eurographics '97. ISSN 1067-7055.

[47] Kowalsky, H.-J.: *Lineare Algebra.* Walter de Gruyter, Berlin, New York, 9. Auflage, 1979.

[48] Kul'chitskij, O. Yu. und S. V. Skrobotov: *Adaptive algorithm of Monte Carlo type for calculating the integral characteristics of complex systems.* Autom. Remote Control, 47:812–818, 1986.

[49] Lafortune, E.: *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering.* Ph.D. thesis, Department of Computer Science, Katholieke Universitiet Leuven, Leuven, Belgium, Februar 1996.

[50] Lafortune, E. P. und Y. D. Willems: *Bi-directional Path Tracing.*

In: SANTO, H. P. (Herausgeber): *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, Seiten 145–153, Alvor, Portugal, Dezember 1993.

[51] LAFORTUNE, E. P. und Y. D. WILLEMS: *A 5D Tree to Reduce the Variance of Monte Carlo Ray Tracing.*
In: HANRAHAN, P. M. und W. PURGATHOFER (Herausgeber): *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, Seiten 11–20, New York, NY, 1995. Springer-Verlag.

[52] LAFORTUNE, E. P. und Y. D. WILLEMS: *Rendering Participating Media with Bidirectional Path Tracing.*
In: PUEYO, XAVIER und PETER SCHRÖDER (Herausgeber): *Eurographics Rendering Workshop 1996*, Seiten 91–100, New York City, NY, Juni 1996. Eurographics, Springer Wien.
ISBN 3-211-82883-4.

[53] LAKOS, J.: *Large Scale C++ Software Design.*
Addison-Wesley, 1996.

[54] LEGNER, S.: *Monte Carlo-Ray Tracing mit adaptiven Dichtefunktionen in Wavelet-Darstellung.*
Diplomarbeit, Universität Dortmund, 1998.

[55] LEPAGE, G.: *A New Algorithm for Adaptive Multidimensional Integration.*
Journal of Computational Physics, 27:192–203, 1978.

[56] LITFIN, D.: *Eine objekt-orientierte Entwicklungsumgebung für Beleuchtungsverfahren.*
Diplomarbeit, Universität Dortmund, 1993.

[57] LOUIS, A. K., P. MAASS und A. RIEDER: *Wavelets, Theorie und Anwendungen.*
Teubner Studienbücher, 1994.

[58] MEYER, B.: *Object-oriented Software Construction.*
Prentice-Hall, 1988.

[59] MEYERS, S.: *Effective C++.*
Addison-Wesley, 1992.

[60] MEYERS, S.: *More Effective C++.*
Addison-Wesley, 1996.

[61] NIEDERREITER, H.: *Random Number Generation and Quasi-Monte Carlo Methods*, Band 63 der Reihe *CBMS-NSF regional conference series in Appl. Math.*
SIAM, Philadelphia, 1992.

[62] PATTANAIK, S. N. und S. P. MUDUR: *Computation of Global Illumination by Monte Carlo Simulation of the Particle Model of Light.*
Third Eurographics Workshop on Rendering, Seiten 71–83, Mai 1992.

[63] PATTANAIK, S. N. und S. P. MUDUR: *Efficient potential equation solutions for global illumination computation.*
Computers and Graphics, 17(4):387–396, Juli–August 1993.

[64] Peter, I. und G. Pietrek: *Importance Driven Construction of Photon Maps.*
In: *9th Eurographics Workshop on Rendering.* Eurographics, Juni 1998.
held in Vienna, Austria, June 29th–July 1st 1998.

[65] Piehler, J.: *Simulationsmethoden.*
Teubner, Leipzig, 1976.

[66] Pietrek, G.: *Fast Calculation of Accurate Form Factors.*
In: Cohen, Michael F., Claude Puech und Francois Sillion (Herausgeber): *Fourth Eurographics Workshop on Rendering*, Seiten 201–220. Eurographics, Juni 1993.
held in Paris, France, 14–16 June 1993.

[67] Press, W. H., S. A. Teukolsky, W. T. Vetterling und B. P. Flannery: *Numerical Recipes in C.*
Cambridge University Press, Zweite Auflage, 1992.

[68] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy und W. Lorensen: *Object-oriented Modeling and Design.*
Prentice-Hall, 1991.

[69] Ruprecht, D.: *Geometrische Deformationen als Werkzeug in der grafischen Datenverarbeitung.*
Doktorarbeit, University of Dortmund, 1994.
published by Verlag Shaker, 1995, ISBN 3-8265-0451-8.

[70] Salesin, D., D. Lischinski und T. DeRose: *Reconstructing Illumination Functions with Selected Discontinuities.*
In: *Third Eurographics Workshop on Rendering*, Seiten 99–112, Bristol, UK, Mai 1992.

[71] Schwarz, H. R.: *Methode der finiten Elemente.*
B. G. Teubner, Stuttgart, 1991.

[72] Shepard, D.: *A two-dimensional interpolation function for irregularly-spaced data.*
In: *Proceedings of the ACM national conference*, Seiten 517–524, 1968.

[73] Shreider, Yu. A. und N. P. Buslenko: *The Monte Carlo method.*
Pergamon Press, Oxford, 1967.

[74] Sibson, R.: *A Brief Description of Natural Neighbour Interpolation.*
In: Barnett, Vic (Herausgeber): *Interpreting Multivariate Data*, Seiten 21–36. John Wiley & Sons Ltd., New York, 1981.

[75] Sillion, F. und C. Puech: *Radiosity and Global Illumination.*
Morgan Kaufmann, San Francisco, CA, 1994.

[76] Slusallek, P.: *Vision - An Architecture for Physically Based Image Synthesis.*
Ph.D. thesis, Computer Graphics Group, University of Erlangen, Erlangen, Germany, Juni 1995.

[77] Stollnitz, E. J., T. D. deRose und D. H. Salesin: *Wavelets for Computer Graphics: A Primer, Part I.*

Technischer Bericht University of Washington, 1995.

[78] Stollnitz, E. J., T. D. deRose und D. H. Salesin: *Wavelets for Computer Graphics: A Primer, Part II.*
Technischer Bericht University of Washington, 1995.

[79] Stollnitz, E. J., T. D. DeRose und D. H. Salesin: *Wavelets for Computer Graphics.*
Morgan Kaufmann, 1996.

[80] Stroustrup, B.: *The Design and Evolution of C++.*
Addison-Wesley, 1994.

[81] Stroustrup, B.: *The C++ Programming Language.*
Addison-Wesley, Dritte Auflage, 1997.

[82] Ureña, C. und J. C. Torres: *Improved Irradiance Computation by Importance Sampling.*
In: Dorsey, Julie und Philipp Slusallek (Herausgeber): *Eurographics Rendering Workshop 1997*, Seiten 275–284, New York City, NY, Juni 1997. Eurographics, Springer Wien.

[83] Urwani, Z.: *Fotorealistische Bilderzeugung durch Kombination von Ray-Tracing- und Radiosity-Verfahren.*
Diplomarbeit, Universität Dortmund, 1993.

[84] Veach, E. und L. Guibas: *Bidirectional Estimators for Light Transport.*
In: *Fifth Eurographics Workshop on Rendering*, Seiten 147–162, Darmstadt, Germany, Juni 1994.

[85] Veach, E. und L. J. Guibas: *Metropolis Light Transport.*
Computer Graphics, 31(3A):65–76, August 1997.

[86] Watson, D.: `nngridr`*, An Implementation of Natural Neighbour Interpolation.*
David F. Watson, P.O. Box 734, Claremont, WA 6010, Australia, 1994, `www.iinet.com.au/~watson/nngridr.html`.