Textursynthese und -analyse für Anwendungen der Kartographie und Luftbildauswertung

Vom Fachbereich Elektrotechnik und Informationstechnik
der Universität Hannover
zur Erlangung des akademischen Grades

Doktor-Ingenieur

genehmigte

Dissertation

von

Dipl.-Phys. Oliver Stahlhut

geboren am 14. August 1972 in Hildesheim

2004

1. Referent: Prof. Dr.-Ing. C.-E. Liedtke

2. Referent: Prof. Dr.-Ing. habil. Monika Sester

Tag der Promotion: 28.07.2004

Vorwort

Diese Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung der Universität Hannover.

Herrn Prof. Dr.-Ing. C.-E. Liedtke gilt mein besonderer Dank für die hervorragenden Arbeitsmöglichkeiten, die angenehme Arbeitsatmosphäre sowie die Übernahme des Hauptreferats. Durch sein Engagement und die vielfältigen Anregungen zum Thema hat er das Entstehen dieser Arbeit maßgeblich gefördert.

Frau Prof. Dr.-Ing. habil. M. Sester danke ich für die Übernahme des Koreferats.

Der Erfolg wissenschaftlicher Tätigkeit hängt stark von dem gegebenen Umfeld ab. Die Kollegen haben sich stets durch Hilfsbereitschaft und fachlichen Rat ausgezeichnet und sich immer Zeit für kritische Diskussionen genommen. Dies gilt insbesondere für Dipl.-Ing. Jochen Wingbermühle, der entscheidende Phasen meiner Arbeit begleitet hat. Für gute Teamarbeit danke ich außerdem Dr.-Ing. Martin Pahl, Dr.-Ing. Jürgen Bückner, Dipl.-Ing. Hellward Broszio, Dr.-Ing. Sebastian Weik und allen anderen wissenschaftlichen Mitarbeitern. Die Durchsicht und das Korrekturlesen einer Arbeit erfordert Zeit und Konzentration - dafür danke ich Dipl.-Ing. Torsten Wiebesiek, Dipl.-Math. Kai Cordes und meinem Vater Erwin Stahlhut. Eine besondere Verbundenheit habe ich zum Systemmanagement des Institutes, das immer für eine perfekte Infrastruktur gesorgt hat; daher Dank an Dipl.-Ing.(FH) Matthias Schuh, Dr.-Ing. Heiko Münkel und Dipl.-Ing. Nikolaus Meine. Bei der Firma TopoSys und dem LGN Niedersachsen möchte ich mich für die Bereitstellung der Fernerkundungsdaten bedanken.

Daß das Fertigstellen einer Dissertation nicht „nebenbei“ zu bewerkstelligen ist, mußte auch meine Familie mittragen. Daher ein besonderes Dankeschön an meine Frau Manuela und meinen Sohn Jan Eric Stahlhut.

Hannover, Dezember 2003

Eine elektronische Version dieser Arbeit, welche die Bilder in Farbe enthält, kann über die Seite `http://www.tnt.uni-hannover.de/~stahlhut/diss.html` erreicht werden.

Kurzfassung

Inhalt der vorliegenden Arbeit ist die Entwicklung eines Verfahrens für die multiskalige Synthese und Analyse von Bildtexturen mit speziellen Anwendungen im Bereich der Kartographie und Fernerkundung. Der entwickelte Algorithmus kann, ausgehend von einem Bildauschnitt, der eine beliebige, natürliche Textur darstellt, eine visuell ähnliche Textur jeder vorgebenen Größe erzeugen. Die Synthese erfolgt dabei über ein Texturmodell, das implizit durch die Texturprobe gebildet wird, ohne Kenntnis des eigentlichen Texturentstehungsprozesses. Die praktische Anwendung der Textursynthese liegt in der Erzeugung künstlicher Luftbilder für mobile Navigationsaufgaben. Gleichzeitig läßt sich das zugrundeliegende Texturmodell für die überwachte Segmentierung und Klassifikation von Bildtexturen in Fernerkundungsdaten einsetzen und stellt damit eine Basistechnologie für Umweltüberwachungsaufgaben oder eine automatische Aktualisierung von Geoinformationssystemen.

Der Algorithmus nutzt ein statistisches Markov-Zufallsfeld-Texturmodell, das geeignet ist, sowohl regelmäßige als auch stochastische Texturen zu beschreiben. Der eigentliche Syntheseprozeß besteht aus Bildähnlichkeitsmessungen zwischen lokalen Umgebungen der Synthesepositionen und der Texturprobe. Das Verfahren überträgt kleine zusammenhängende Blöcke von der Texturprobe auf das Syntheseprodukt und erhält damit den lokalen Texturzusammenhang bei hoher Ausführungsgeschwindigkeit. Durch den Einsatz einer Auflösungspyramide werden alle auftretenden Strukturgrößen gleichwertig verarbeitet. Die Synthese erlaubt Randbedingungen und erzeugt kachelbare Ergebnisse. Eine Auswertung der Nutzungsstatistik der Texturprobe zur Laufzeit und getrennte Farbverarbeitung erhöhen die Synthesequalität gegenüber anderen Ansätzen. Ein mehrschichtiges Beschleunigungskonzept ermöglicht durch Ausnutzung von Hardware-beschleunigter, paralleler Vektorverarbeitung und Vektorquantisierung hohe Syntheseraten für die anwendungsspezifische Verarbeitung großer Texturen.

Stichwörter: Textur, Synthese, Segmentierung, Markov-Zufallsfelder, MRF, Fernerkundung, Luftbilder

Abstract

This work describes a novel, multi-scale texture processing algorithm, which is applied for synthesis of aerial images and segmentation of remote sensing data. The algorithm was designed for the tileless generation of large images of arbitrary size from small sample images of natural textures. The synthesised texture shows features that are visually similar to the sample over a wide frequency range. The synthesis utilises an implicit texture model provided by the texture sample itself and has no knowledge about the original texture creation process. Practical applications of the algorithm include synthesis of artifical aerial images for mobile navigation purposes and supervised texture classification of remote sensing data as a base technology for environmental monitoring and automatic update of geographic information systems.

The algorithm is based on a Markov random field (MRF) texture model which is suitable for representing regular as well as stochastic textures. The actual synthesis process measures the similarity of local neighbourhoods at the synthesis position and the texture sample. Small texture patches are copied from the sample to the synthesis surface, thus preserving the local texture coherence, while achieving a high texturing rate at the same time. As an extension to the MRF texture model a multi-scale image pyramid captures and reproduces texture features over a wide frequency range. The synthesis allows incorporation of the original texture sample and produces tileable results. An evaluation of the samples' usage statistic during runtime and separate colour processing increase the synthesis quality compared to previous techniques. A multilayer acceleration scheme provides hardware accelerated parallel vector operations and vector quantisation which enables the handling of large textures for the desired applications.

Key words: texture, synthesis, segmentation, Markov Random Fields, MRF, remote sensing, aerial images

Inhaltsverzeichnis

Symbolverzeichnis

$\mathcal{N}$	Menge der Referenznachbarschaftsvektoren, Seite 42
$\mathcal{V}$	Vektorraum, in dem die Vektoren aus der Menge $\mathcal{N}$ dargestellt werden, Seite 49
$\overline{U}$	Mittlere Anzahl der Referenztexturblockzugriffe, Seite 66
σ	Codebuch der Vektorquantisierung, Seite 53
$\vec{N}_j^{ref}$	Nachbarschaftsumgebung des Referenztexturblocks j ausgedrückt als linearer Vektor, Seite 42
$\vec{N}^{syn}$	Nachbarschaftsumgebung der aktuellen Syntheseposition ausgedrückt als linearer Vektor, Seite 42
B	Texturblock, quadratische Gruppe von Bildelementen, Seite 20
B_j^{ref}	Referenztexturblock j, Seite 20
B_u^{syn}	Syntheseblock u, Seite 20
d_N	Dimension der Vektoren $\vec{N}_j$ und des Vektorraums $\mathcal{V}$, Seite 49
$F(N_i, N_j)$	Ähnlichkeitsmaß für zwei Nachbarschaftsumgebungen, Seite 21
I	Eingangsbild/-textur, Seite 9
I_0	Originalauflösung des Eingangsbildes I, Seite 9
I_l	Auflösungsstufe l des Eingangsbildes I, Seite 9
I_{L-1}	niedrigste Auflösungsstufe des Eingangsbildes I, Seite 34
L	Anzahl der Auflösungsstufen der Bildpyramide, Seite 9
$L\{B_u^{syn}\}$	Liste der Syntheseblöcke, Seite 20
$L_1(\vec{N}_i, \vec{N}_j)$	L_1-Metrik/Norm, Summe der absoluten Differenzen zweier Vektoren, City-Block-Abstand, Seite 40

$L_2(\vec{N}_i, \vec{N}_j)$	L_2-Metrik/Norm, Summe der quadratischen Differenzen zweier Vektoren, Euklidischer-Abstand, Seite 40
L_∞	L_∞-Metrik/Norm, Seite 50
$N(B_j^{ref})$	Nachbarschaftsumgebung des Referenztexturblocks j, Seite 20
$N(B_u^{syn})$	Nachbarschaftsumgebung des Syntheseblocks u, Seite 20
$N(P)$	Nachbarschaftsumgebung des Pixels P, Seite 17
P	Pixel, Bildelement, Seite 17
$U(B_j^{ref})$	Zugriffszähler für den Referenztexturblock B_j^{ref}, Seite 66
W_B	(ungerade) Breite/Höhe des Texturblocks B, Seite 20
W_N	(ungerade) Breite der Nachbarschaftsumgebung N, Seite 20
W_N^{eff}	effektive Breite der multiskaligen Nachbarschaftsumgebung N, Seite 24
W_N^{erw}	(ungerade) Breite des multiskaligen Erweiterungsanteils der Nachbarschaftsumgebung N, Seite 25
$\|\mathcal{V}\|$	Anzahl der Vektoren in $\mathcal{V}$, Seite 55
p	Anzahl der Partitionen bei der Vektorquantisierung, Seite 56

1 Einleitung

1.1 Texturen - Analyse, Modelle, Synthese und Segmentierung

Der Begriff „Textur“ hat seinen Ursprung im lateinischen *textura* und heißt wörtlich übersetzt Gewebe, Weben oder Struktur. Er ist eng mit dem Begriff „Struktur“ verbunden, im lateinischen *structura*: Zusammensetzung, Aufbau. Die Struktur beschreibt in einem größeren Rahmen die Beziehungen zwischen Objekten und deren Anordnung. Die Textur hingegen bezieht sich auf visuelle oder haptische Eigenschaften einer Oberfläche und legt gleichzeitig die lokale Struktur fest. Texturen werden mit Attributen wie Weichheit, Rauheit, Feinheit, Grobheit, Körnigkeit, Richtung, Regularität, Zufälligkeit, etc. beschrieben [49] [43]. Diese sprachlichen Bezeichnungen für die visuelle, haptische bzw. taktile Ausprägung von Texturen sind aber nicht präzise genug, um eine allgemeine, formale Definition abzuleiten.

Die vorliegende Arbeit widmet sich der Reproduktion von Bildtexturen, den zugrundeliegenden Texturmodellen und Aspekten der Bildanalyse mit Hilfe der Textur. In der digitalen Bildverarbeitung bezeichnet der Begriff der Bildtextur *Bilder von natürlich texturierten Oberflächen* bzw. *simulierte Muster, die Texturen natürlicher Oberflächen approximieren* [42]. Diese Bilder werden entweder durch Filterung und Abtastung einer analogen Vorlage, wie z.B. Foto, Druck oder Bild eines optischen Systems, gewonnen oder direkt im Bildverarbeitungssystem erzeugt. Die Abbildung einer natürlichen Oberflächentextur liefert eine Bildtextur, die im Rechner als grauwertige, farbige oder allgemein mehrkanalige, zweidimensionale Matrix von Signalwerten aus einer beschränkten Signalwertmenge vorliegt. Entscheidend für die Charakteristik der Bildtextur sind der Aufnahmesensor und die Beleuchtung, welche die Wiedergabe einer bestimmten, realen Textur sehr unterschiedlich beeinflussen. Die speziellen Eigenschaften, die eine Textur von einem beliebigen Bild unterscheiden, werden in einem Texturmodell erfaßt.

Bildtexturen lassen sich durch elementare Grundbestandteile und deren räumliche Anordnung definieren. Diese **Tex**tur**el**emente werden als Texton [60] [61] oder Texel [51] bezeichnet. Ein Texel wird in der Bildmatrix durch eine spezifische Gruppierung von Bildelementen (picture elements, Pixel) dargestellt. Historisch gesehen, wurde in regelmäßige und stochastische Texturen unterteilt [49]. Regelmäßige Bildtexturen werden durch eine bestimmte Menge unterschiedlicher Texel und eine geeignete Positionierungsvorschrift gebildet. Stochastische Texturen haben sehr kleine Texel oder auch nur Pixel,

deren Anordnung nicht deterministisch ist, so daß sich keine Positionierungsvorschrift angeben läßt. Die Grenzen zwischen diesen Extremen sind jedoch fließend, natürliche Texturen sind eine Mischung aus regelmäßiger und stochastischer Textur, wie in Abbildung 1.1 dargestellt.

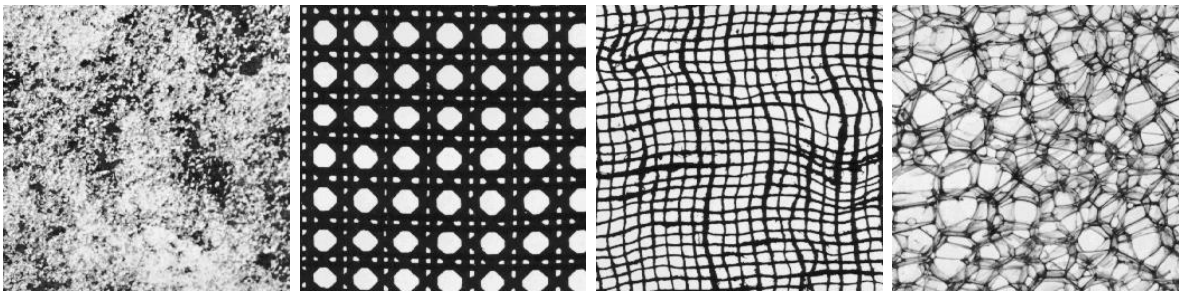


Abbildung 1.1: Natürliche Texturen aus [12]: stochastisch (D100), regelmäßig (D101) und Mischformen (D103, D111).

Anstatt für beide Bildtexturklassen unterschiedliche rechnergestützte Beschreibungen einzuführen, müssen geeignete Texturmodelle gleichzeitig die stochastische Komponente und die Regelmäßigkeit erfassen. So kann eine Textur z.B. als Texelhierarchie mit räumlicher Selbstähnlichkeit, d.h. statistischer Wiederholung der Texel, dargestellt werden. In der Literatur finden sich zahlreiche Ansätze für die Beschreibung von Bildtexturen in Form von physikalischen [98] [104] [105], fraktalen [34] [69], statistischen [25] [82] [37] [54] [109] [85] oder statistisch-deterministischen [35] Modellen.

Statistische Bildmodelle umfassen prinzipiell die Menge aller Bilder und sind deshalb auch geeignet, jede beliebige Bildtextur zu modellieren. Der Bildraum ist unvorstellbar groß, jedoch wird nur ein winziger Bruchteil durch natürliche Texturen gebildet [64] [88]. Gleichzeitig wird angenommen, daß sich biologische Sehprozesse im Laufe der Evolution an die statistischen Merkmale der beobachteten Signale angepaßt haben [4] und daher eine direkte Verbindung zwischen Sehen und Bildstatistik existiert. Damit die statistische Bildmodellierung handhabbar wird, müssen eingeschränkte Texturmodelle wie z.B. Markov-Zufallsfelder [38] verwendet werden. Grundlegende Arbeiten zur statistischen Charakterisierung von Texturen sind von Julesz [59] erbracht worden, der postuliert hat, daß sich Texturen durch die Betrachtung der Statistik der Ordnung $N > 0$ unterscheiden lassen. Daraus folgt, daß Texturen, die dieselbe Ausprägung gewisser statistischer Merkmale haben, von einem menschlichen Beobachter als identisch bzw. ähnlich angesehen werden. Für die Modellierung einer Textur genügt es also, die „richtigen“ statistischen Merkmale für die Modellbildung zu identifizieren und die Parameter dieser Merkmale durch Analyse einer Texturprobe zu bestimmen.

Das statistische Texturmodell kann anschließend für die Synthese einer beliebig dimensionierten Bildtextur eingesetzt werden, die der Modelltextur visuell ähnlich ist und gleichzeitig keine Diskontinuitäten aufweist. Eine andere Anwendung des Texturmodells ist die Segmentierung, d.h. das Unterteilen eines Bildes in unterschiedliche Bildregionen,

die nach dem visuellen Verständnis zusammengehörig sind, d.h. die gleiche homogene Bildtextur aufweisen.

Die Möglichkeiten zur Erzeugung und Variation von Texturen sind unbegrenzt. Um dennoch eine Vergleichbarkeit von Texturverarbeitungsalgorithmen zu erreichen, existieren de facto Standards in Form von Texturkatalogen, die Proben von unterschiedlichsten, natürlichen Texturen zur Verfügung stellen. Wichtigste Vertreter sind das Fotoalbum von Brodatz [12] und der digitale „VisTex“ Katalog [81]. Diese Quellen sind auch die Grundlage für die Texturbeispiele dieser Arbeit, die den Schwerpunkt auf die Reproduktion von Texturen - der Textursynthese - legt. Ein universeller Textursynthese-Algorithmus ermöglicht eine Vielzahl von Anwendungen, wie Restauration, Kompression und Segmentierung von Bildern oder auch Oberflächentexturierung von dreidimensionalen Objekten, um realistisch wirkende, virtuelle Welten zu erzeugen. Die speziellen Anwendungen der vorliegenden Arbeit liegen im Bereich der Verarbeitung von Fernerkundungsdaten für die Synthese künstlicher Luftbilder und der Klassifikation von Luftbildtexturen.

1.2 Textursynthese - Stand der Technik

Die Aufgabe einer Textursynthese kann wie folgt beschrieben werden: Bildtextur sei definiert als ein visuelles Muster, gebildet aus den Signalwerten an diskreten Stützstellen eines unendlichen Gitters. Aus einer endlichen Stichprobe dieser Textur sollen andere Stichproben derselben unendlichen Textur erzeugt werden. Notwendige Voraussetzung ist, daß die Probe groß genug ist, um die Stationarität und die elementaren Texel zu erfassen. In den folgenden Abschnitten sollen unterschiedliche Klassen von Verfahren vorgestellt werden, die eine Textursynthese ermöglichen.

1.2.1 Prozedurale Algorithmen

Eine prozedurale Synthese erzeugt eine Textur nicht auf Basis einer Texturprobe, sondern generiert sie mit einem mathematischen Algorithmus [80]. Grundlage dieser Ansätze sind meist Basisfunktionen mit spezieller Rauschcharakteristik, wie z.B. das Perlin-Rauschen [30], die jedem Bildelement einen zufälligen Signalwert zuordnen, wobei zwischen den Signalwerten unterschiedlicher Elemente Korrelationen bestehen. Die Rauschfunktionen bestimmen das Maß an Zufälligkeit für die Initialisierung der mathematischen Algorithmen, welche die prozedurale Erzeugung von Texturen wie Gras, Wasser, Schnee, Wolken, Sterne, Feuer, Holz oder Marmor durchführen. Die prozeduralen Texturen sind keine natürlichen Texturen, sondern approximieren diese mehr oder weniger befriedigend. Die Berechnungen sind sehr zeitaufwendig und müssen in Fließkommadarstellung erfolgen [53]. Historisch gesehen, sind diese Verfahren in der Computergrafik entstanden und erlaubten erstmals, die Oberfläche von dreidimensionalen Objekten als Kontinuum zu texturieren. Zuvor wurde die Oberfläche durch Abbildung von Ausschnitten natürlicher

Texturen (texture mapping) bebildert, um diese detaillierter und komplexer wirken zu lassen, als ihre Geometrie es eigentlich festlegt.

Mit prozeduraler Textursynthese können nur bestimmte Texturklassen erzeugt werden. Die Verfahren sind nicht allgemein anwendbar und erzielen keine realistisch wirkenden Texturen, so daß sie nicht der Ausgangspunkt für eine weitere Entwicklung sein können.

1.2.2 Physikalische Simulation

Verfahren dieser Klasse erzeugen Texturen direkt durch Simulation eines physikalischen Entstehungsprozesses. Insbesondere natürliche Muster von Fell, Schuppen, Panzern, Schneckengehäusen und Haut lassen sich durch Diffusions-/Reaktionsprozesse oder zellulare Texturierung künstlich erzeugen [98] [104] [105]. Auch Erosionsprozesse an Oberflächen von Metallen oder Mineralien lassen sich durch Simulation eindrucksvoll reproduzieren [29] [28]. Natürliche Texturen werden allerdings durch sehr unterschiedliche physikalische Prozesse erzeugt, deren Beschreibung nur in speziellen Fällen möglich ist. Daher decken diese Verfahren nur einen kleinen Teil möglicher Texturen ab und können insbesondere für die Beschreibung von Objekten in Fernerkundungsdaten, wie für die Anwendungen in Kapitel 5 beabsichtigt, nicht eingesetzt werden.

1.2.3 Kachelung und Texture Patching

Die Kachelung von Texturen ist die einfachste Methode, um aus einer vorgegebenen Texturprobe eine größere Ausgabetextur zu synthetisieren. Eine Kachelung durch wiederholtes Aneinandersetzen der Texturprobe führt im allgemeinen zu Diskontinuitäten an den Übergängen zwischen zwei Teilstücken, die nur durch torodiale Anschlußbedingungen des oberen und unteren sowie linken und rechten Randes der Texturprobe vermieden werden können. Diese Randbedingungen werden von natürlichen Texturen meistens nicht erfüllt und können bei einfacher Herangehensweise nur durch Spiegelung oder Überblendung approximiert werden. Durch die Kachelung werden außerdem die globalen Strukturmerkmale der Textur beeinflußt, da eine sichtbare, schachbrettartige Repetition auftritt.

Das „Texture Patching“ (übersetzt: Aneinandersetzen von Texturausschnitten) ist an das Prinzip der Kachelung angelehnt. Diese Verfahren kopieren zufällig oder aufgrund von Randbedingungen ausgewählte, große Blöcke der Originaltextur und optimieren die Anschlußbedingungen [106] [31] [70] [27]. Beim *Image Quilting* [31] überlappen sich die kopierten Blöcke auf der Ausgabetextur. In den Überlappungsbereichen werden Schnitte berechnet, die für optimale Anschlußbedingungen der Texturblöcke sorgen. Dies geschieht durch Aufstellen und Minimierung einer Kostenfunktion für Anschlußfehler. Aufgrund des unterschiedlichen Schnittverlaufs ergeben sich beliebig berandete, aneinander anschließende Texturregionen (texture patches). In einem finalen Nachverarbeitungsschritt werden die Übergänge zwischen den Patches optimiert. Obwohl das *Image Quil-*

ting kein aufwendiges Texturmodell benötigt, erreicht es bei geringem Rechenaufwand qualitativ hochwertige Ergebnisse, die mit anderen aktuellen Verfahren konkurrieren können.

1.2.4 Synthese durch Merkmalsanpassung

Für die Synthese mit Merkmalsanpassung werden Texturen durch eine festgelegte Menge von (statistischen) Merkmalen modelliert. Der Syntheseprozeß versucht, ein rauschinitialisiertes Bild in eine Textur zu überführen, indem die ausgewählten Merkmale von Ausgabetextur und Texturprobe schrittweise aneinander angepaßt werden. Alle bekannten Methoden verwenden eine steuerbare Auflösungspyramide für eine frequenzabhängige Adaption von Referenz- und Ausgabetextur. Die einzelnen Auflösungsstufen der Pyramiden werden richtungsgefiltert, um die Merkmalsanpassung in Abhängigkeit von R verschiedenen Richtungskomponenten der Eingangstextur durchzuführen. Die frequenz- und orientierungsabhängige Verarbeitung von Merkmalen entspricht nach dem heutigen Erkenntnisstand dem menschlichen Wahrnehmungsprozeß [103] [5] [72]. Das erste Verfahren von Heeger und Bergen [54] betrachtet ausschließlich die Histogrammverteilungen in den richtungsgefilterten Frequenzbändern. Mit diesem Ansatz können stochastische Texturen sehr gut reproduziert werden; stärker strukturierte Texturen, wie z.B. Bilder von Mauerwerk, werden durch die Histogramme allein nicht ausreichend beschrieben. De Bonet [9] erweitert den Ansatz durch Berücksichtigung der Abhängigkeiten struktureller Merkmale über die einzelnen Auflösungsstufen hinweg. Dabei nutzt das Verfahren eine Kachelung der Texturprobe, die unter Erhaltung der Merkmale schrittweise randomisiert wird. Dieses Verfahren kann eine Vielzahl von Texturen wiedergeben, liefert aber wie [54] die besten Ergebnisse für stochastische Texturen. Die Arbeiten von Portilla und Simoncelli [92] [84] [85] betrachten statistische Merkmale höherer Ordnung mit Wavelet-Koeffizienten. Durch die Modellierung lokaler Nachbarschaftsbeziehungen wird eine vergleichsweise hohe Synthesequalität sowohl für stochastische als auch für regelmäßige Texturen erzielt.

1.2.5 Markov-Zufallsfelder

Zufallsfelder ermöglichen die Repräsentation von Bildern bzw. Texturen in Form eines Wahrscheinlichkeitsmodells. Markov Zufallsfelder (Markov Random Fields - MRF) beschränken sich für die Beschreibung von Texturen auf lokale und stationäre Zufallsprozesse [25] [82] [109] [79] [32]. Stationär bedeutet, daß die Nachbarschaftsumgebung eines beliebigen Bildelements der Textur an jeder Bildposition ähnlich aussieht, d.h. vergleichbare Signalwertverteilungen aufweist. Lokal bedeutet, daß der Signalwert eines Bildelements auf Basis seiner Nachbarschaftsumgebung geschätzt werden kann und dabei unabhängig von den Signalwerten des restlichen Bildes ist, siehe auch Grundlagen MRF Abschnitt 2.4.

Anhand der Texturprobe werden die statistischen Abhängigkeiten zwischen Paaren (Statistik 2. Ordnung) oder auch Gruppen von Bildelementen (Statistik höherer Ordnung) des Markov-Zufallsfeldes ermittelt und parametrisch, z.B. mit Gibbs-Potentialen [42] [109] oder nicht-parametrisch in Form von mehrdimensionalen Histogrammen [79], als Texturbeschreibung abgelegt und für den eigentlichen Syntheseprozeß abgetastet.

Markov Zufallsfelder stellen eine gute Approximation für eine Vielzahl von Texturen dar und liefern im allgemeinen gute Syntheseergebnisse. Neuere Ansätze mit MRF-Texturmodellen verzichten auf die explizite Darstellung der statistischen Abhängigkeiten innerhalb einer lokalen Nachbarschaftumgebung und messen die Wahrscheinlichkeiten für das Auftreten eines Signalswertes direkt durch Bildvergleich zwischen Syntheseumgebung und Texturprobe [32] [101] [52]. Diese Verfahren sind äußerst rechenintensiv und wurden, obwohl bereits 1981 in [36] vorgeschlagen, erst durch leistungsfähige Rechnersysteme realisierbar.

Die bekannten Implementationen sind sehr unterschiedlich und zum Teil Mischformen verschiedener Textursynthesemethodiken. So nutzt [101] zusätzlich Auflösungspyramiden, um die Größe der lokalen Nachbarschaftsumgebung effektiv zu erhöhen und gleichzeitig lokale *und* globale Merkmale der Texturprobe zu reproduzieren. In [3] und [55] wird das MRF-Texturmodell um eine Signalwert-Prädiktion erweitert, die das Verfahren um Größenordnungen beschleunigt. Die Prädiktion beruht auf der Annahme, daß es eine hohe Wahrscheinlichkeit dafür gibt, daß benachbarte Bildelemente der Synthese aus benachbarten Bildelementen der Texturprobe stammen und daher direkte Kopiervorgänge ohne Messung der Wahrscheinlichkeitsverteilung zulässig sind. Das Verfahren überträgt größere zusammenhängende Gebiete und kann daher auch als Hybridverfahren von MRF und „Texture Patching“ angesehen werden.

1.3 Grenzen der beschriebenen Verfahren

Prozedurale Algorithmen und physikalische Simulation können nur bestimmte Klassen von Texturen erzeugen und sind nicht geeignet, Vorgaben in Form einer Texturprobe für die Synthese direkt zu verarbeiten. Sie erfordern eine aufwendige, manuelle Anpassung des zugrundeliegenden Modells an die visuelle Erscheinung jeder Texturprobe, was für die meisten Texturen nicht möglich ist, und sind daher für weitergehende Entwicklungen ungeeignet.

Aus der Klasse der Syntheseverfahren mit Merkmalsanpassung sind die neuesten Implementationen in der Lage, eine Vielzahl von unterschiedlichen Texturen mit einer vergleichsweise hohen Synthesequalität zu reproduzieren. Die Merkmalsanpassung basiert auf einem statistischen Texturmodell, das durch eine Auswahl von geeigneten Merkmalen definiert wird. Es müssen die Merkmale identifiziert und modelliert werden, die für den menschlichen Sehprozeß relevant, d.h. für die visuelle Unterscheidbarkeit von Texturen verantwortlich sind. Da die Funktionen des Sehapparates noch nicht vollständig verstan-

den sind, ist es schwierig, die „richtigen“ Merkmale zu bestimmen. Im Zuge der Verbesserung dieser Verfahrensklasse wurde die Menge der Merkmale immer weiter vergrößert und damit auch die Komplexität erhöht, was letztlich einen hohen Rechenaufwand nach sich zieht. Die neueste Entwicklung [85] steht als MATLAB-Code zur Verfügung [83] und konnte in den direkten Vergleich mit anderen Verfahren gestellt werden. Generell zeigt sich, daß bei bestimmten Texturen unterschiedliche Frequenzanteile durchmischt werden und hochfrequente Strukturen durch Glättung von Ecken und Kanten zum Teil verloren gehen, was auch von [101] bemerkt wurde. Diese Verfahrensklasse kann insbesondere den Vergleich mit neueren MRF-Algorithmen weder in punkto Synthesegeschwindigkeit noch in der Synthesequalität bestehen.

Texture Patching Algorithmen überzeugen mit hohen Texturierungsgeschwindigkeiten durch Blocktransfer, weisen aber auch einige Nachteile auf. Zum einen ist die unterliegende Blockstruktur nicht zu jedem Texturtyp „kompatibel“ und kann zu Artefakten an den Blockübergängen führen, die auch durch einen optimierten Schnitt [31] oder Überblendung [70] nicht ausgeglichen werden können. Zum anderen modellieren die Verfahren die globale Struktur der Eingangstextur nicht. Durch den Blocktransfer wird die lokale Statistik erhalten, die Reproduktion der niedrigen Frequenzen der Globalstruktur wird aber durch die Breite der Überlappzone bei der Ausrichtung der Blöcke festgelegt. Wird die Überlappzone vergrößert oder die Blockgröße verkleinert, geht das Texture Patching in ein MRF-Verfahren ohne Auflösungshierarchie über.

Alle Markov-Zufallsfeld-Texturmodelle reproduzieren prinzipbedingt nur die Textureigenschaften, die von der betrachteten Nachbarschaftsumgebung umschlossen werden. Globale Strukturen, welche die Größe des Markov-Zufallsfelds überschreiten, liegen außerhalb des Texturmodells und werden daher nicht erfaßt. Die MRF-Verfahren, die eine explizite, parametrische Beschreibung der statistischen Abhängigkeiten innerhalb des Zufallsfeldes nutzen, um diese für den Syntheseprozeß abzutasten, sind sehr langsam, da sie iterativ arbeiten und sich keine sinnvolle Konvergenzbedingung aufstellen läßt [32]. Neuere MRF-Algorithmen, die nicht parametrisch mit Bildvergleichen arbeiten, erlauben zum einen die Reproduktion von Statistik höherer Ordnung, zum anderen ist es möglich, Auflösungspyramiden zu integrieren, um einen größeren Frequenzbereich bei der Strukturreproduktion zu erfassen [101]. Nachteil dieser Verfahren ist, daß sie in jedem Syntheseschritt immer nur ein Bildelement erzeugen und daher extrem rechenintensiv sind. Die bekannten Verfahren neigen außerdem zu Selbstwiederholung bestimmter Texturelemente.

1.4 Lösungsansatz

Für die vom Stand der Technik ausgehende Entwicklung eines neuen Textursyntheseverfahrens steht die Synthesequalität im Vordergrund. Betrachtet man die zum Teil sehr guten Ergebnisse der unterschiedlichen, bekannten Verfahren, wird schnell deutlich, daß die Markov-Zufallsfeld-Texturmodelle die besten Resultate produzieren. Dies gilt insbeson-

dere für neuere Implementationen, die auf eine explizite, parametrische Darstellung der Feldstatistik verzichten und implizit mit Bildvergleichen arbeiten [32] [101]. Daher sind diese Ansätze die Basis für den Entwurf eines neuen, verbesserten Syntheseverfahrens. Der fortwährende Bildvergleich von Syntheseumgebung und Texturprobe eröffnet die Möglichkeit, nicht nur einzelne Bildelemente, sondern *kleine* Bildblöcke zu übertragen, so daß eine Mischform aus „Texture Patching“ und MRF-Modell entsteht. Die Synthese von Bildblöcken schafft den Vorteil, automatisch die lokalen Textureigenschaften zu erhalten [31] [70]. Anders als beim *Image Quilting* [31] sollen aber keine optimalen Schnitte in der Überlappzone von großen Blöcke erfolgen, sondern die Blockgröße als Regler für die Synthesegeschwindigkeit dienen. Das Verfahren soll im Gegensatz zu [101] eine Synthese unter Randbedingungen mit einem Durchlauf ausführen. Dabei sollen kachelbare Ergebnisse erzeugt und Strukturen über einen großen Frequenzbereich reproduziert werden können. Die letzten drei Punkte erfordern den Einsatz einer Auflösungspyramide [101] und eines lokal-abhängigen, variablen Markov-Zufallsfelds [32]. Die Beschleunigung des Verfahrens wird in Verbesserung zu [101] mit einer mehrschichtigen Strategie erzielt, die neben einer geeigneten Vektorquantisierung die Parallelisierungsfähigkeiten moderner Rechnerarchitekturen ausnutzt, um hohe Syntheseraten erzielen zu können. Die Verarbeitung von mehrbändigen Farbtexturen kann bei gleicher oder höherer Qualität durch Dekomposition und einen Nachverarbeitungsschritt beschleunigt werden. Eine kontinuierliche Auswertung der Übertragungsstatistik während der Synthese unterdrückt die Selbstwiederholungsprobleme, die bei anderen Verfahren [32] [101] beobachtet werden. Gleichzeitig eröffnet die Untersuchung geeigneter Bildvergleichsmetriken und der Arbeitsweise des Algorithmus einen neuen Ansatzpunkt für die Verfahrensbeschleunigung.

Im folgenden Kapitel 2 werden einige grundlegenden Aspekte zu den Themen Auflösungspyramiden, Bildähnlichkeit, Statistik und Markov-Zufallsfelder behandelt. Die Implementation des Algorithmus mit der Umsetzung der einzelnen Anforderungspunkte sowie die Optimierung der Verfahrensparameter werden in Kapitel 3 beschrieben, gefolgt von Syntheseergebnissen in Kapitel 4. In den Kapiteln 3 und 4 werden nur homogene, natürliche Texturen aus den bereits erwähnten Bildkatalogen [12] und [81] verwendet, um eine Vergleichbarkeit mit den Ergebnissen anderer Literaturquellen herzustellen und die Universalität des Algorithmus zu demonstrieren.

Eine hohe Synthesequalität und Ausführungsgeschwindigkeit ist die Voraussetzung für die im Anwendungskapitel 5 beschriebene Synthese künstlicher Luftbilder und die Textursegmentierung von Fernerkundungsdaten, die das breite Einsatzspektrum des entwickelten Algorithmus veranschaulichen.

2 Grundlagen

In den folgenden vier Abschnitten werden in kurzer Form einige Grundlagen erläutert, die das theoretische Gerüst für die Algorithmik des Textursyntheseverfahrens bilden. Dazu zählen die Auflösungspyramide, die genutzt wird, um ein breites Spektrum an Texturfrequenzen zu reproduzieren, die Messung der Bildähnlichkeit als Kernfunktion des Verfahrens und statistische Methoden, die eine wesentliche Rolle bei den Vergleichsverfahren für Textursynthese und -segmentierung spielen. Das Kapitel endet mit einer kurzen Einführung in statistische Bildmodelle, die auf das hier verwendete Markov-Zufallsfeld und dessen unterschiedliche Realisierungen führt.

2.1 Die Auflösungspyramide

Eine Auflösungspyramide ist eine in der graphischen Datenverarbeitung häufig verwendete Repräsentationsform eines digitalen Bildes. Sie besteht aus L Auflösungsstufen I_l, wobei I_0 - die unterste Stufe der Pyramide - das Bild selbst ist. Eine Stufe I_{l+1} wird durch Filterung und Unterabtastung von I_l erzeugt (Reduktion). Ordnet man die einzelnen Stufen zentriert übereinander an, ergibt sich eine Pyramide wie in Abbildung 2.1.

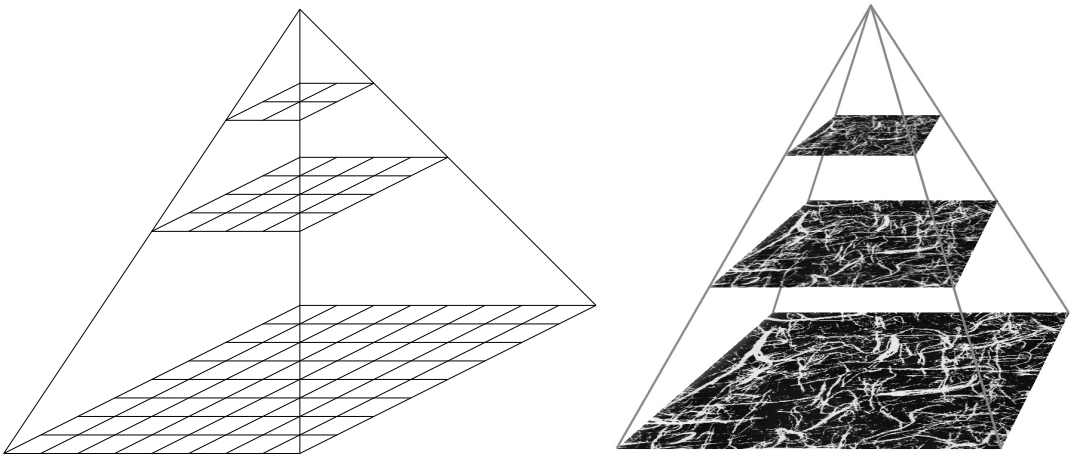


Abbildung 2.1: Auflösungspyramiden. Links schematisch die Pyramide einer 8×8 Bildmatrix, rechts eine typische Textur in 3 Auflösungsebenen.

In der Praxis wird üblicherweise eine Halbierung der Grenzfrequenz und Unterabtastung mit Faktor 2 zwischen zwei aufeinanderfolgenden Stufen durchgeführt. Dabei reduziert sich die Bildfläche von I_l auf ein Viertel in I_{l+1}. Andere Abtastfaktoren werden praktisch nicht benutzt, da sie eine aufwendige Konvertierung unterschiedlicher Bildraster erfordern. Um Aliasing-Artefakte zu verhindern, muß I_l vor der Unterabtastung tiefpaßgefiltert werden. Das Filter muß symmetrisch und normiert sein, um richtungsunabhängig zu filtern und Grauwertverschiebungen zu verhindern. Da er in der Pyramide mehrfach hintereinander verwendet wird, sollte er bei Reihenschaltung seine Tiefpaß-Charakteristik nicht ändern und einem einfach angewendeten Tiefpaßfilter mit stärkerer Filterwirkung entsprechen. Diesen Anforderungen genügt ein Gauß-Filter, das außerdem kantenglättend wirkt, dafür aber kein Überschwingen an Bildkanten (Ringing) aufweist. Eine einfache Approximation eines Gaußfilters ist der Filterkern

$$\vec{g} = (0.05, 0.25, 0.4, 0.25, 0.05) \tag{2.1}$$

der als 5×5 Filtermatrix

$$w(i,j) = \vec{g}^t * \vec{g} \tag{2.2}$$

mit der jeweiligen Auflösungsstufe I_l gefaltet wird [14]

$$I_{l+1}(x,y) = \sum_{j=-2}^{2} \sum_{i=-2}^{2} w(i,j) I_l(2 \cdot x + i, 2 \cdot y + j). \tag{2.3}$$

Um die Fensteroperation der Faltung vollständig zu definieren, werden die Bildränder von I_l gespiegelt. Gleichung 2.3 beinhaltet über die Adressierung $2 \cdot x$ und $2 \cdot y$ gleichzeitig die Unterabtastung, d.h. für die Bildkantenlängen gilt:

$$dx(I_{l+1}) = dx(I_l)/2 \qquad dy(I_{l+1}) = dy(I_l)/2. \tag{2.4}$$

Um Zuordnungsprobleme in der späteren Anwendung zu vermeiden, sollten $dx(I_l)$ und $dy(I_l)$ für alle l gerade Zahlen sein. Durch die Quantisierung der Pixelkoordinaten und den Abtastfaktor 2 gehen ungeradzahlige Pixelzeilen an den Bildrändern verloren.

Die gesamte Pyramide wird durch wiederholtes Anwenden von Filterung mit Unterabtastung (Reduktion) erzeugt.

2.2 Bildähnlichkeit

Viele analysierende Verfahren der digitalen Bildverarbeitung erfordern einen Vergleich von Bildern oder Bildausschnitten und eine Berechenbarkeit der Bildähnlichkeit. Prinzipiell existieren beliebig viele Möglichkeiten, ein Ähnlichkeitsmaß zu definieren. Ein geeignetes Maß muß aber dem Verständnis des menschlichen Betrachters entsprechen und läßt sich nur durch genauere Untersuchung des menschlichen Sehens und Erkennens ableiten, was eine Aufgabe der visuell-kognitiven Wissenschaften ist.

Nach heutiger Auffassung gliedert sich das Sehen und Erkennen in zwei Phasen - dem präattentiven und dem attentiven Prozeß. Beim präattentiven Prozeß wird die Szene als Ganzes erfaßt und stark ausgeprägte Merkmale, wie Kanten und deren Orientierung, Kontraste, Farben und Bewegungen wahrgenommen. Dieser Vorgang erfolgt parallel und mit kurzer Verzögerung. In der attentiven Phase wird eine „Region-of-interest“ auf die Merkmale der ersten Phase gerichtet. Hierbei werden kleinere Details wahrgenommen und Objekte oder Regionen segmentiert. Der Wahrnehmungsprozeß der attentiven Phase ist stark durch den Erfahrungsschatz des Individuums geprägt und eher als subjektiv zu bezeichnen. Daher ist die Modellierung des Sehens und Erkennens mit dem Ziel einer Berechenbarkeit eines Bildeindrucks eine äußerst komplizierte Aufgabe, die nach heutigem Stand der Technik nicht gelöst ist.

Ein geeigneter Ansatz ist die frequenz- und orientierungsabhängige Verarbeitung von Merkmalen der präattentiven Phase, z.B. Kanten und Farben um ein Maß für die Ähnlichkeit von Bildregionen berechnen zu können [5] [67] [8] [87]. Der Aufwand für die Berechnung derartiger Merkmale ist allerdings enorm und wird meist nur für einzelne Bildvergleiche durchgeführt. Das in dieser Arbeit entwickelte Verfahren beruht auf der Berechnung von Tausenden bis mehreren Millionen Bildähnlichkeiten und ist daher auf einfache, schnell zu berechnende Ähnlichkeitsmaße wie z.B. die L_1 oder L_2-Norm angewiesen, deren Eigenschaften im Rahmen der Beschreibung des Algorithmus in Abschnitt 3.5.3 diskutiert werden.

2.3 Statistische Texturmerkmale

Für die Charakterisierung einer Textur können statistische Methoden eingesetzt werden [59], beispielsweise durch Berechnung statistischer Merkmale lokaler Nachbarschaftsumgebungen. Der Begriff der Statistik n-ter Ordnung bezeichnet die Anzahl der an der Bildung eines Merkmals beteiligten Bildelemente, die zueinander feste, relative Positionen haben.

Im Fall der Statistik 1. Ordnung wird von einem Bildelement und der korrespondierenden Umgebung ausgegangen. Typische Merkmale sind die Momente und zentralen Momente k-ter Ordnung der Signalverteilung im betrachteten Bildbereich:

$$\mu_k = \sum_{s=1}^{N} p(s)s^k \qquad \sigma_k^2 = \sum_{s=1}^{N} (s-\mu_1)^k p(s) \qquad 0 \leq p(s) \leq 1, \sum_{s=1}^{N} p(s) = 1 \tag{2.5}$$

Dabei ist $p(s)$ die relative Häufigkeit des Signalwerts s (μ_1: Mittelwert, σ_2: Varianz).

Statistik 2. Ordnung berechnet die Signalabhängigkeiten zwischen jeweils zwei Bildelementen (Pixelpaare, Cliquen, siehe Abbbildung 2.2) innerhalb einer Nachbarschaft, die in den sogenannten Co-occurence-Matrizen dargestellt werden. Für jede Clique ergibt sich eine eigene Matrix, deren Größe durch die Grauwertdynamik bestimmt wird. So ergibt sich bei einer Dynamik von 8 Bit und damit 256 Graustufen eine Matrix der Größe

256×256. In der Praxis wird üblicherweise eine Quantisierung auf 32 oder weniger Grauwertstufen vorgenommen, um den Rechen- und Darstellungsaufwand zu begrenzen. In einer grundlegenden Arbeit hat Haralick [50] Vorschläge für geeignete statistische Merkmale zur Beschreibung von Texturen gemacht, die aus den Co-occurence-Matrizen ermittelt werden können. Diese sogenannten 14 Haralick-Momente umfassen unter anderem Kontrast, Homogenität, Entropie und Korrelation.

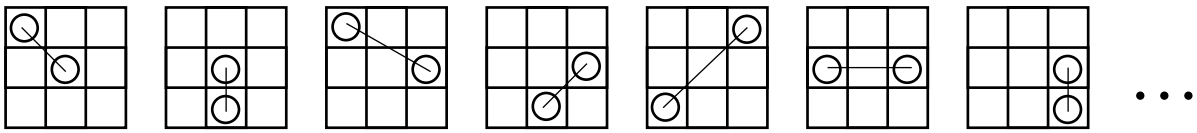


Abbildung 2.2: Cliquen zur Bestimmung von Statistikmomenten 2. Ordnung.

Analog zur Statistik 2. Ordnung wird die Statistik 3. Ordnung durch Cliquen definiert, die aus drei Bildelementen mit fester, relativer Position innerhalb einer Nachbarschaftsumgebung bestehen. Der Beschreibungsaufwand erhöht sich signifikant, da nun dreidimensionale Co-occurence-Matrizen erstellt werden müssen und mehr Möglichkeiten für die Bildung einer dreielementigen Clique existieren. Dies ist ein Grund, warum diese Darstellungsform in der Verfahrensentwicklung selten genutzt wird. Ein weiterer Aspekt ist die niedrige Empfindlichkeit des menschlichen Sehapparates auf entprechende statistische Merkmale, die dazu führt, daß bestimmte Texturen, die sich lediglich in Momenten höherer Statistiken unterscheiden, vom Menschen nicht differenziert werden können, siehe Beispiel in Abschnitt 5.2. Generell gilt diese Ununterscheidbarkeit allerdings nicht [19] [20] [62].

In der Literatur sind weitere Methoden für die Bestimmung statistischer Texturmerkmale beschrieben, eine Übersicht gibt [103]. Rechentechnisch weniger aufwendig ist z.B. die Berechnung von Textur-Energie-Merkmalen nach Laws [68]. Dazu werden zunächst die sogenannten „Mikrostatistiken“ der Textur durch Faltung mit einem Satz verschiedener 5×5-Faltungsmasken erzeugt. Anschließend wird durch eine Fensteroperation lokal die Energie der Ergebnisse gemessen und daraus Merkmalsvektoren zusammengestellt. Laws hat einen Satz von 25 Merkmalen definiert und exemplarisch demonstriert, daß diese leistungsfähiger als die Haralick-Momente sein können [68].

2.4 Markov-Zufallsfelder

Das in den folgenden Kapiteln beschriebene Verfahren nutzt ein Markov-Zufallsfeld (Markov Random Field, MRF) als Texturmodell. An dieser Stelle sollen zunächst in Anlehnung an [42] einige grundlegende Begriffe erläutert werden.

Die Repräsentation von Bildern erfolgt durch Gitter von Bildpunkten. Das endliche, rechteckige und regelmäßige Gitter $\mathbf{R}$ ist eine geordnete Menge von Gitterpunkten mit

den Koordinaten (m, n) und einer Größe von M Zeilen und N Spalten:

$$\mathbf{R} = [(m, n) : m = 0, ..., M - 1;\ n = 0, ..., N - 1]. \tag{2.6}$$

In Kurzschreibweise sollen die Koordinaten (m, n) mit i bezeichnet werden. Ein Bild $\mathbf{s}$ besteht aus einem Gitter, dem für jeden Gitterpunkt i Werte s_i einer endlichen Signalmenge $\mathbf{U}$ zugeordnet werden. Im folgenden werden nur grauwertige Bilder $\mathbf{s}$ mit 8Bit-Dynamik und 256 Graustufen betrachtet:

$$\mathbf{s} = [s_i : i \in \mathbf{R};\ s_i \in \mathbf{U} = \{0, ..., 255\}]. \tag{2.7}$$

Ein Zufallsfeld $\mathbf{Z}$ mit Elementen Z_i, die zufällig aus den s_i gewählt wurden, kann als Wahrscheinlichkeitsmodell für Bilder betrachtet werden [24]:

$$\mathbf{Z} = [Z_i : i \in \mathbf{R}]. \tag{2.8}$$

Wenn $\mathcal{S}$ die Menge aller möglichen Bilder $\mathbf{s}$ ist, kann ein Zufallsfeld $\mathbf{Z}$ durch eine diskrete Verteilungsfunktion beschrieben werden, welche die Wahrscheinlichkeiten $P(\mathbf{Z} = \mathbf{s})$ wiedergibt. Dabei gilt für alle $\mathbf{s} \in \mathcal{S}$, daß die Einzelwahrscheinlichkeiten $P(\mathbf{Z} = \mathbf{s}) \geq 0$ sind und $\sum_{\mathbf{s}\in\mathcal{S}} P(\mathbf{Z} = \mathbf{s}) = 1$ ist. Jedes $P(\mathbf{Z} = \mathbf{s})$ ist extrem klein, da die Mächtigkeit von $\mathcal{S}$ aufgrund der Kombinatorik $|\mathcal{S}| = |\mathbf{U}|^{|\mathbf{R}|}$ sehr groß ist. Es muß nun ein Weg gefunden werden, aus gegebenen Wahrscheinlichkeiten $P(\mathbf{Z} = \mathbf{s})$ ein Bildmodell aufzustellen, das die Wahrscheinlichkeitsverteilung annähernd reproduziert, d.h. hohe Wahrscheinlichkeiten für die gewünschten Bilder und eine Wahrscheinlichkeit nahe 0 für alle anderen Bilder liefert. Dazu sollen nun die bedingten Wahrscheinlichkeiten für Signalwerte einzelner Bildelemente Z_i des Zufallsfeldes bei bekannter Feldumgebung betrachtet werden.

Sei nun $\mathbf{R}^i$ das Gitter $\mathbf{R}$ vermindert um das Element i. Ebenso sind $\mathbf{Z}^i$ das Zufallsfeld $\mathbf{Z}$ ohne Element Z_i und $\mathbf{s}^i$ das Bild $\mathbf{s}$ ohne s_i. Die bedingte Wahrscheinlichkeit $P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i)$, daß das Element Z_i des Zufallsfelds den Wert u hat, kann bei Annahme von festen Werten $\mathbf{s}^i$ für den Rest des Bildes nach der Bayes'schen Formel berechnet werden [58]:

$$P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i) = \frac{P(Z_i = u \wedge \mathbf{Z}^i = \mathbf{s}^i)}{P(\mathbf{Z}^i = \mathbf{s}^i)}. \tag{2.9}$$

Für die Summe über alle Werte der Signalmenge gilt: $\sum_{u\in\mathbf{U}} P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i) = 1$.

Da die Umgebung $\mathbf{Z}^i$ des Elements Z_i alle übrigen Punkte des Gitters beinhaltet, verlangt das Aufstellen einer geeigneten Verteilungsfunktion die Berücksichtigung von sehr vielen Variablen, was zu einem sehr komplexen und schwer handhabbaren Problem führt. Eine Vereinfachung wäre eine Dekomposition der Berechnung in voneinander unabhängige Teile, falls dieses möglich ist (Markov-Eigenschaft). Dazu sollen nun nur die Nachbarn j des Gitterpunktes i berücksichtigt werden, deren Signalwerte s_j einen Einfluß auf die bedingte Wahrscheinlichkeit $P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i)$ haben. Gibt es eine derartige Untermenge von Gitterpunkten j, läßt sich das Problem der Berechnung der bedingten Wahrscheinlichkeiten dekomponieren. Der Einfluß der Elemente s_j auf Z_i wird auch als Interaktion

benachbarter Bildelemente (pixel interaction) bezeichnet. Die Menge der Nachbargitterpunkte j von i ist die Nachbarschaft $\mathbf{N}_i$, die Menge aller $\mathbf{N}_i$ das Nachbarschaftssystem $\mathbf{N_S}$ [7].

Ein Markov-Zufallsfeld (MRF) ist ein Zufallsfeld für das die Menge aller $\mathbf{N}_i$ eine Untermenge des Gitters $\mathbf{R}$ ist [24], d.h. für die Mächtigkeiten gilt:

$$|i \cup \mathbf{N}_i| < |\mathbf{R}| \quad \forall i \in \mathbf{R}. \tag{2.10}$$

Markov-Eigenschaft bedeutet Unabhängigkeit von Komponenten zu einigen anderen, was im Falle des Markov-Zufallsfeldes als Bildmodell mit *Lokalität* gleichzusetzen ist: die bedingte Wahrscheinlichkeit $P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i)$ für das Auftreten eines Signalwertes u an der Stelle i hängt nicht von dem ganzen Bild, sondern nur von den Signalwerten s_i der begrenzten Nachbarschaft $\mathbf{N}_i$ ab.

Beispiele für Nachbarschaftsumgebungen $\mathbf{N}_i$ sind in Abbildung 2.3 dargestellt. In der Praxis werden aufgrund der Berechnungskomplexität nur kleine Markov-Zufallsfelder betrachtet. Ein Verfahren für die Abschätzung der optimalen Größe der Nachbarschaft in Abhängigkeit der betrachteten Textur wird in [96] vorgestellt.

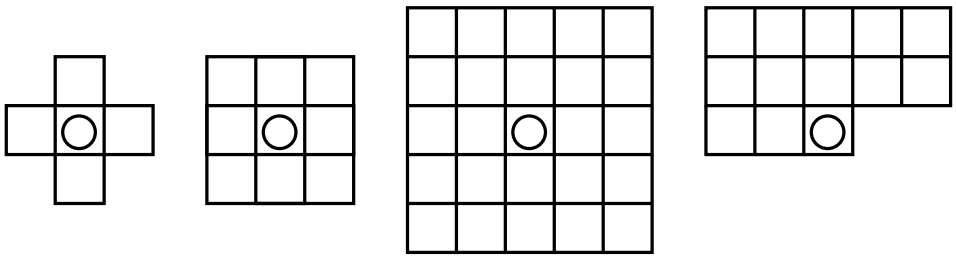


Abbildung 2.3: Nachbarschaftsumgebungen. Symmetrisch und kausal (rechts, im Sinne einer spalten- und zeilenweisen Verschiebung).

Die Realisierung eines Markov-Zufallsfeld-Texturmodells kann über unterschiedliche Ansätze erfolgen. Viele Verfahren setzen direkt das Konzept der interagierenden Pixelpaare (Cliquen) um. Das Clifford-Hammersley Theorem besagt, daß jedes Markov-Zufallsfeld durch eine Gibbs-Wahrscheinlichkeitsverteilung beschrieben werden kann [7]. Die Interaktion der Cliquen wird mit (Gibbs-)Potentialen dargestellt. Diese Repräsentationsform der Markov- bzw. Gibbs-Zufallsfelder wird z.B. von [42], [109] und [33] verwendet. Die Interaktionsstruktur, also die geometrische Anordnung der Cliquen, und die Stärke der Potentiale werden aus einer Texturprobe ermittelt und bilden das parametrische Wahrscheinlichkeitsmodell der Textur. Für Textursynthese oder -segmentierung wird das Modell in einem iterativen Prozeß abgefragt. Ein nicht-parametrisches Modell [79] stellt anhand einer Texturprobe Co-occurence-Matrizen (2D-Signalhistogramme) für die Cliquen auf, siehe Abschnitt 2.3, die direkt für eine Schätzung der bedingten Wahrscheinlichkeit $P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i)$ genutzt werden können.

Neben diesen Ansätzen mit *expliziter* Repräsentation der Cliquen und der ihnen zugeordneten Verteilungsfunktionen in parametrischer oder nicht-parametrischer Form werden in neueren Literaturquellen [32] [101] [52] Ansätze verfolgt, die ein MRF-Texturmodell über direkte Bildvergleiche realisieren. Diese werden im folgenden als *implizite* Modelle bezeichnet und sollen genauer beschrieben werden, da sie die Grundlage für den im Rahmen dieser Arbeit entwickelten Algorithmus sind.

Aus einer Texturprobe I_0, die Teil der unendlichen Textur I_∞ ist, soll eine visuell ähnliche Textur I_{syn} synthetisiert werden. Das Markov-Zufallsfeld wird mit Hilfe einer konstanten, quadratischen Nachbarschaft $\mathbf{N}_i$ mit der ungeraden Breite W_N um die möglichen Gitterpunkte i beschrieben. Um den Signalwert u an einer Stelle i zu bestimmen, muß, wie oben beschrieben, die bedingte Wahrscheinlichkeit $P(Z_i = u|\mathbf{Z}^i = \mathbf{s}^i)$ bei gegebener Nachbarschaftsumgebung berechnet werden. Dazu wird anhand von I_∞ die Menge $K(i)$ aller Bildelemente k erzeugt, deren Nachbarschaften $\mathbf{N_k}$ identisch mit $\mathbf{N}_i$ sind:

$$K(i) = \{k \in I_\infty | \Delta(\mathbf{N}_k, \mathbf{N}_i) = 0\} \tag{2.11}$$

Δ ist eine Funktion, welche die visuelle Ähnlichkeit zweier gleich großer Bildausschnitte mißt. Die bedingte Wahrscheinlichkeitsverteilung für die Signalwerte u an der Position i kann dann aus dem Histogramm der Menge $K(i)$ geschätzt werden. Da die unendliche Textur I_∞ nicht bekannt ist, sondern nur ein Ausschnitt I_0 zur Verfügung steht, kann es sein, daß die Menge $K(i)$ leer ist, weil $\Delta \neq 0 \quad \forall k$. Daher wird zunächst $\Delta(\mathbf{N}_k, \mathbf{N}_i) = min.$ gesucht und anschließend die Menge K_i mit einem Schwellwert ϵ und $\Delta \leq min. + \epsilon$ definiert. Die meisten Verfahren verzichten auf die explizite Berechnung und Abtastung des Histogramms von $K(i)$ und setzen $\epsilon \equiv 0$, d.h. $K(i)$ wird einelementig. Diese Vorgehensweise widerspricht dem theoretischen Ansatz, führt in der Praxis insbesondere für kleine Texturproben I_0 aber zu vergleichbaren Ergebnissen.

Prinzipiell kann mit der geschilderten Methode nur ein Loch von der Größe eines Bildelements verschlossen werden. Um die eigentliche Textursynthese durchführen zu können, werden bei der Berechnung der Funktion Δ immer nur die bereits definierten Bildelemente der Nachbarschaft $\mathbf{N}_i$ genutzt und alle anderen Elemente maskiert.

Ein implizites MRF-Modell besticht durch eine vergleichweise einfache Realisierung, da es ausschließlich mit Bildvergleichen und ohne komplexe, parametrische Darstellung interagierender Cliquen auskommt. Gleichzeitig ist es vollständiger als ein explizites Modell, weil innerhalb der Nachbarschaftsumgebung Statistiken höherer Ordnung erfaßt werden. Nachteilig ist der hohe Rechenaufwand durch die häufige Messung der Umgebungsähnlichkeit für die Synthese jedes einzelnen Bildelements.

3 Algorithmus für die Textursynthese

Der folgende Hauptteil der Arbeit beschreibt die Konzeption und Implementierung des Algorithmus. Basierend auf den Vorgaben aus Abschnitt 1.4 wird ein Textursyntheseverfahren mit implizitem Markov-Zufallsfeld-Texturmodell entwickelt. Die Implementation des Verfahrens und damit auch der prinzipielle Ablauf einer Synthese ist abhängig von der gestellten Aufgabe. In der Praxis ergeben sich meist zwei typische Szenarien:

- Eine „leere“, d.h. zunächst undefinierte Fläche, soll mit einer Textur gefüllt werden.
- Eine Fläche ist zum Teil mit einer Textur initialisiert, die als Ausgangspunkt für eine Synthese dienen kann.

Im zweiten Fall kann die bestehende Information zum Beispiel eine Kante einer Textur sein, die in eine bestimmte Richtung erweitert werden soll. Üblicherweise werden aber Teilstücke von Texturen auf der Ausgabefläche verteilt und dann miteinander „verwachsen“ oder aber auch Löcher in einer bestehenden Textur verschlossen, siehe auch Abschnitt 4.2.

Die Anwendungsfälle unterscheiden sich also in der Initialisierung der Eingangsdaten. Zunächst soll der allgemeinere Fall einer Synthese mit Randbedingungen in Form einer teil-initialisierten Textur dargestellt werden. Die Synthese einer vollständig undefinierten Fläche wird dann in Abschnitt 3.4 beschrieben. Abbildung 3.1 zeigt einen Bildausschnitt I_0 einer unvollständigen Textur. Die schwarze Fläche des Bildes I_0 ist undefiniert und soll im Verlauf der Textursynthese vervollständigt werden.

Die grundlegende Idee ist, zusammenhängende Blöcke aus der wohldefinierten Referenztextur auf die undefinierten Bereiche zu übertragen. Die Auswahl des jeweils optimalen Blocks erfolgt aufgrund seiner Nachbarschaftsumgebung - an eine undefinierte Stelle des Bildes wird der Block der Referenztextur übertragen, dessen unmittelbare Nachbarschaft der Nachbarschaft an der Syntheseposition am ähnlichsten ist.

Diese Vorgehensweise entspricht dem impliziten Markov-Zufallsfeld Ansatz für die Synthese einzelner Bildelemente: Ein Pixel $P_{Referenz}$ der Referenztextur läßt sich gut an einer Position $X_{Synthese}$ einfügen, wenn die Nachbarschaftsumgebung $N(P_{Referenz})$ der zugehörigen Nachbarschaftsumgebung $N(X_{Synthese})$ ähnlich ist, siehe Abschnitt 2.4. Allerdings verarbeitet der Algorithmus nicht einzelne Bildelemente, sondern zusammenhängende Texturinformation in Form eines Blocks um $P_{Referenz}$. Die Synthese findet somit in der Texeldomäne und nicht in der Pixeldomäne wie bei anderen Ansätzen statt.

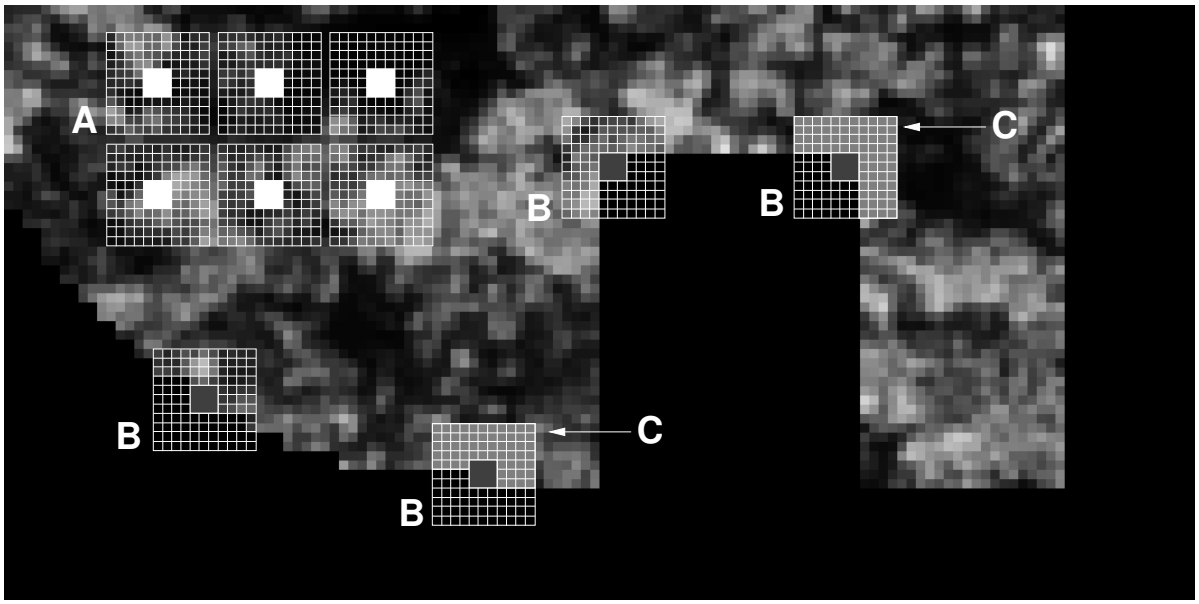


Abbildung 3.1: Synthesekomponenten: Texturausschnitt auf undefiniertem, schwarzen Hintergrund, Syntheseblöcke (dunkelgrau in B), Referenzblöcke (weiß in A), symmetrische Blocknachbarschaften (weiße Gitter von A und B), variable Nachbarschaftmaske (hellgrau/C in den Blocknachbarschaften).

Für die Darstellung des Algorithmus wird als erstes die Synthese einer einzelnen Auflösungsstufe betrachtet. Die multiskalige Erweiterung des Verfahrens mittels einer Auflösungspyramide wird im darauf folgenden Abschnitt 3.3 beschrieben. Zunächst müssen aber die verschiedenen Darstellungsformen und Zusatzinformationen, die im Laufe der Synthese entstehen, erläutert werden, um die Eigenschaften der Textursynthese beschreiben und diskutieren zu können.

3.1 Ein- und Ausgabedaten

Abbildung 3.2 zeigt die unterschiedlichen Informationen, die während einer Synthese verarbeitet bzw. erzeugt werden. Oben abgebildet sind die Eingangsdaten in Form von unvollständiger Textur und Arbeitsmaske. Die zu bearbeitenden Texturen sind bei diesem Verfahren zunächst grauwertig, d.h. ein Pixel kann durch ein Byte, entsprechend 256 Graustufen, repräsentiert werden. Die Behandlung von Farbtexturen wird in Abschnitt 3.8 beschrieben. Die Arbeitsmaske markiert die zu bearbeitende Region und die Texturreferenz. Unten in Abbildung 3.2 sind die Syntheseprodukte dargestellt. Links das Syntheseergebnis, dann eine Kopierkarte und die Flächennutzungsübersicht. Die Kopierkarte kodiert für jeden Punkt auf der synthetisierten Fläche die ursprünglichen Koordinaten in den Referenzdaten. Die x- und y-Koordinaten werden während der Synthese in zwei separaten Bildern abgelegt und für die Darstellung in der Kopierkarte als roter

und grüner Kanal eines RGB-Bildes auf den Wertebereich von jeweils [0..255] normiert. Die Kopierkarte visualisiert unter anderem das Syntheseverhalten des Algorithmus. Die Flächennutzungsübersicht gibt Aufschluß über die verwendeten Pixel, bzw. Blöcke der Texturreferenz. Für jeden von der Referenz auf die Synthesefläche übertragenen Block werden in der Flächennutzungskarte Zähler an den korrespondierenden Pixelpositionen inkrementiert. Nach der Synthese kann diese Nutzungsstatistik - z.B. wie hier als grauwertiges Bild auf [0..255] normiert - einen Hinweis auf die Gleichmäßigkeit der Nutzung der Referenztextur geben. Den Kopier- und Flächennutzungskarten kommen somit besondere Bedeutungen zu, die in den Abschnitten 3.6.6 und 3.7 behandelt werden.

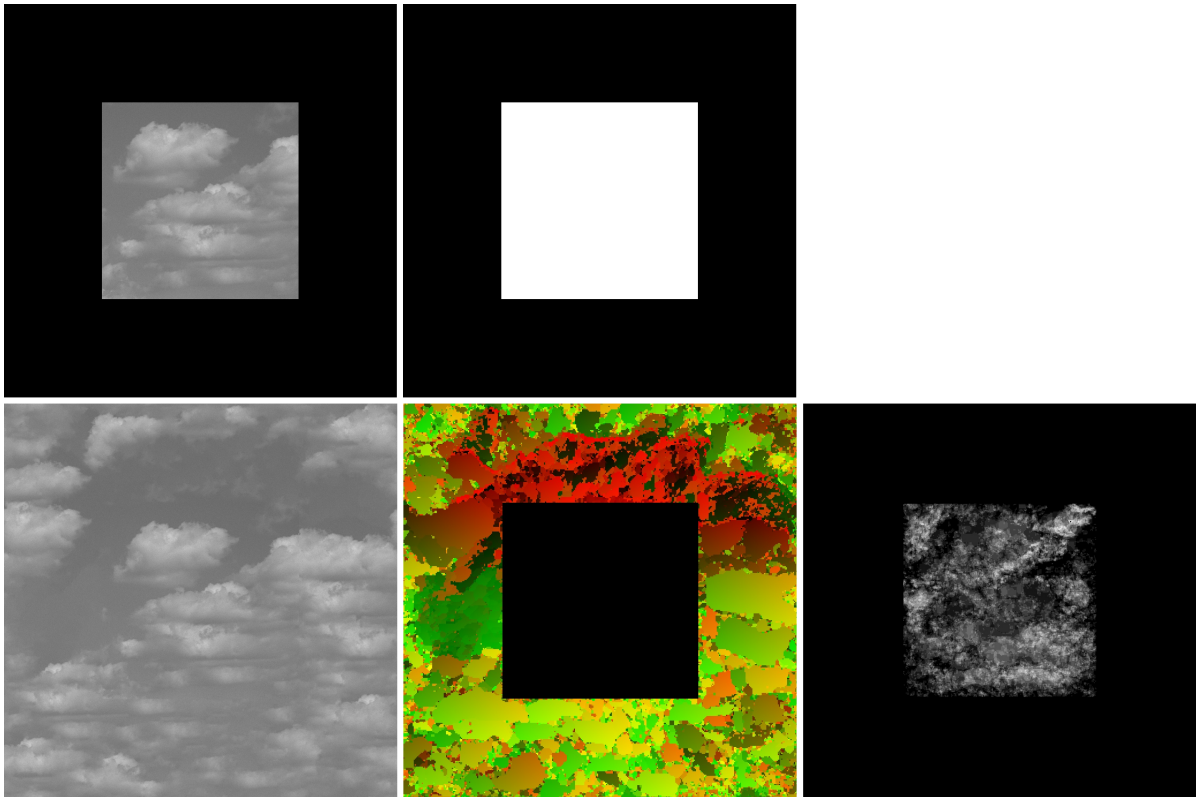


Abbildung 3.2: Syntheseinformation. Oben: unvollständige Textur und Arbeitsmaske, unten: synthetisierte Textur, Kopierkarte und Flächennutzung.

3.2 Synthese einer Auflösungsstufe

3.2.1 Strategie

Im ersten Schritt muß eine Strategie für das blockweise Texturieren der undefinierten Fläche entwickelt werden. Ein Ansatz für eine geeignete Strategie läßt sich aus der Betrachtung eines Puzzlespieles ableiten. Dort werden nicht wahllos oder in einem ra-

sterorientierten Verfahren die Puzzleteilchen aneinandergesetzt, sondern zuerst Bereiche gefüllt, die am meisten Umgebungsinformation, d.h. fertige Puzzleabschnitte aufweisen. Jedes zusätzlich (korrekt) angebrachte Puzzlestück verbessert die Information für das Ansetzen weiterer Teilchen. Analog zu dieser „Puzzle"-Strategie soll auch der Texturierungsalgorithmus arbeiten. Das Verfahren muß mit der Synthese dort beginnen, wo die Nachbarschaftsumgebung eines Syntheseblocks am besten definiert ist. Nach jedem einzelnen Syntheseschritt werden diese optimalen Positionen neu bestimmt und das Verfahren fortgeführt, bis die gesamte undefinierte Fläche vervollständigt ist.

3.2.2 Identifikation der Syntheseblöcke

I_0 aus Abbildung 3.1 wird zunächst in nicht überlappende Blöcke B_i mit ungerader Blockkantenlänge W_B zerlegt. In der Praxis ist $W_B \in [3, 5, 7, 9, 11]$. Auch eine Pixelsynthese mit $W_B = 1$ kann durchgeführt werden und ermöglicht so den Vergleich mit älteren Verfahren, die pixelbasiert arbeiten. Im nächsten Schritt wird aus der Menge der B_i eine Liste der Syntheseblöcke $L\{B_u^{syn}\}$ erzeugt, in die alle Blöcke B_i eingetragen werden, die ganz oder teilweise undefinierte Textur aufweisen. Die vier dunkelgrauen, von einem weißen Gitter eingeschlossenen Blöcke in Abbildung 3.1 (B) sind beispielsweise Kandidaten für die Liste $L\{B_u^{syn}\}$. Die weißen Gitter markieren die Geometrie der verwendeten Nachbarschaft N. Es handelt sich um eine symmetrische, quadratische Pixelnachbarschaft des eingeschlossenen Blocks $N(B_u^{syn})$ mit Breite W_N. Diese Blocknachbarschaften beinhalten sowohl definierte als auch undefinierte (hier: schwarze) Pixel. Die Liste $L\{B_u^{syn}\}$ wird nun absteigend nach der Anzahl der definierten Pixel $P^{def}(N(B_u^{syn}))$ der Nachbarschaftsumgebung jedes Listenelements sortiert. Am Anfang der Liste stehen nach der Sortierung die Syntheseblöcke, deren Nachbarschaft am besten beschrieben ist.

3.2.3 Bestimmung der Referenztextur

Im nächsten Schritt muß die Referenzinformation für die Textursynthese identifiziert werden. Das Verfahren ermöglicht die Nutzung von zwei unterschiedlichen Quellen für Referenztextur. Zum einen kann ein Bild einer Referenztextur zusammen mit einer Selektionsmaske von außen vorgegeben werden, zum anderen kann auch die unvollständige Textur selbst als Quelle dienen. Die Bildmaske, die den undefinierten, zu synthetisierenden Bereich kennzeichnet (schwarze Fläche in Abbildung 3.1), definiert umgekehrt - invertiert - den Bereich, der als Referenz genutzt werden kann. In diesem Fall dürfen jedoch nur Blöcke B_j^{ref} verwendet werden, deren Nachbarschaft $N(B_j^{ref})$ vollständig in der wohldefinierten Referenztextur liegt. Die Position dieser Blöcke wird pixelgenau verwaltet, um alle in der Synthese auftretenden Anschlußbedingungen erfüllen zu können. Sechs Beispiele für Referenztexturblöcke sind in Abbildung 3.1 (A) weiß gekennzeichnet. Wie bereits bei den Syntheseblöcken steht die symmetrische Nachbarschaft $N(B_j^{ref})$ stellvertretend für den Block B_j^{ref}.

3.2.4 Syntheseschleife

Die Syntheseschleife beginnt mit der Bestimmung der Nachbarschaft N des ersten Syntheseblocks B_0^{syn} der Blockliste $L\{B_u^{syn}\}$. Diese Nachbarschaft $N(B_0^{syn})$ wird mit den Nachbarschaften *aller* Referenzblöcke $N(B_j^{ref})$ verglichen und der Referenzblock auf die Syntheseposition B_0^{syn} übertragen, der die ähnlichste Nachbarschaft besitzt. Das Ähnlichkeitsmaß ist eine Funktion $F(N(B_0^{syn}), N(B_j^{ref}))$ - ausgedrückt durch eine Abstandsmetrik -, deren globales Minimum die optimale Referenznachbarschaft und damit den optimalen Referenzblock kennzeichnet. In den Abschnitten 2.2 und 3.5.3 wird näher auf geeignete Abstandsmaße eingegangen - für alle folgenden Synthesebeispiele wird zunächst die L_1-Norm verwendet. Der Block B_0^{syn} wird aus der Blockliste $L\{B_u^{syn}\}$ entfernt und die Listenpositionen aller zu B_0^{syn} geometrisch benachbarten Blöcke $B_{\Delta(0)}^{syn}$ aktualisiert. Geometrisch benachbart bedeutet in diesem Fall Evaluierung der L_∞-Norm mit einem Abstand $\Delta_{max}(W_B, W_N)$. Letzteres ist notwendig, da die Synthese ein dynamischer Prozeß ist. Durch die Synthese des Blocks B_0^{syn} wurde unter Umständen die Anzahl der definierten Pixel der Nachbarschaften $N(B_{\Delta(0)}^{syn})$ und somit auch die Sortierung verändert. Die Syntheseschleife wird fortgesetzt, bis alle Blöcke aus der Liste $L\{B_u^{syn}\}$ entfernt wurden und damit alle undefinierten Bereiche von I_0 gefüllt worden sind.

Das Sortierkriterium $P^{def}(N(B_u^{syn}))$ bewirkt, daß im Sinne der zugrundeliegenden Strategie immer die Blöcke als erste synthetisiert werden, welche die am besten definierte Nachbarschaft aufweisen. Dieses Verhalten des Algorithmus ist in Abbildung 3.3 veranschaulicht. Die beiden Blöcke in den Ecken des fehlenden Rechtecks werden als erste erzeugt. Das Rechteck wächst zu und bildet mit der kantigen Begrenzung im linken Bildbereich den Kondensationskeim für den weiteren Fortschritt der Synthese.

Um die ständige Neusortierung der Liste zu beschleunigen, kann diese in eine Hash-Tabelle mit $P^{def}(N(B_u^{syn}))$ als Hash-Schlüssel überführt werden. Eine spatiale Indextabelle, welche die Elemente der Hash-Tabelle indiziert, kann die Identifikation von benachbarten Blöcken $(B_{\Delta(u)}^{syn})$ deutlich effizienter gestalten. Sie verbessert die Listenverwaltung dadurch, daß der Suchprozeß, dessen Laufzeit von der Anzahl der Listenelemente und somit quadratisch von der Bildgröße abhängt, durch eine konstante Bearbeitungszeit ersetzt wird. Die konstante Bearbeitungszeit ergibt sich aus der direkten Indizierung aller relevanten Listenelemente.

3.2.5 Alternative Strategie

Für bestimmte Anwendungen ist es sinnvoll, die „Puzzle“-Strategie in einer modifizierten Form zu verwenden. Anstatt das Texturwachstum von einem oder mehreren optimalen Kondensationskeimen ausgehen zu lassen, können bestehende Texturflächen von den Rändern ausgehend gleichmäßig erweitert werden. Für die Abarbeitung der Syntheseblockliste $L\{B_u^{syn}\}$ bedeutet dies, daß alle Blöcke nacheinander synthetisiert werden,

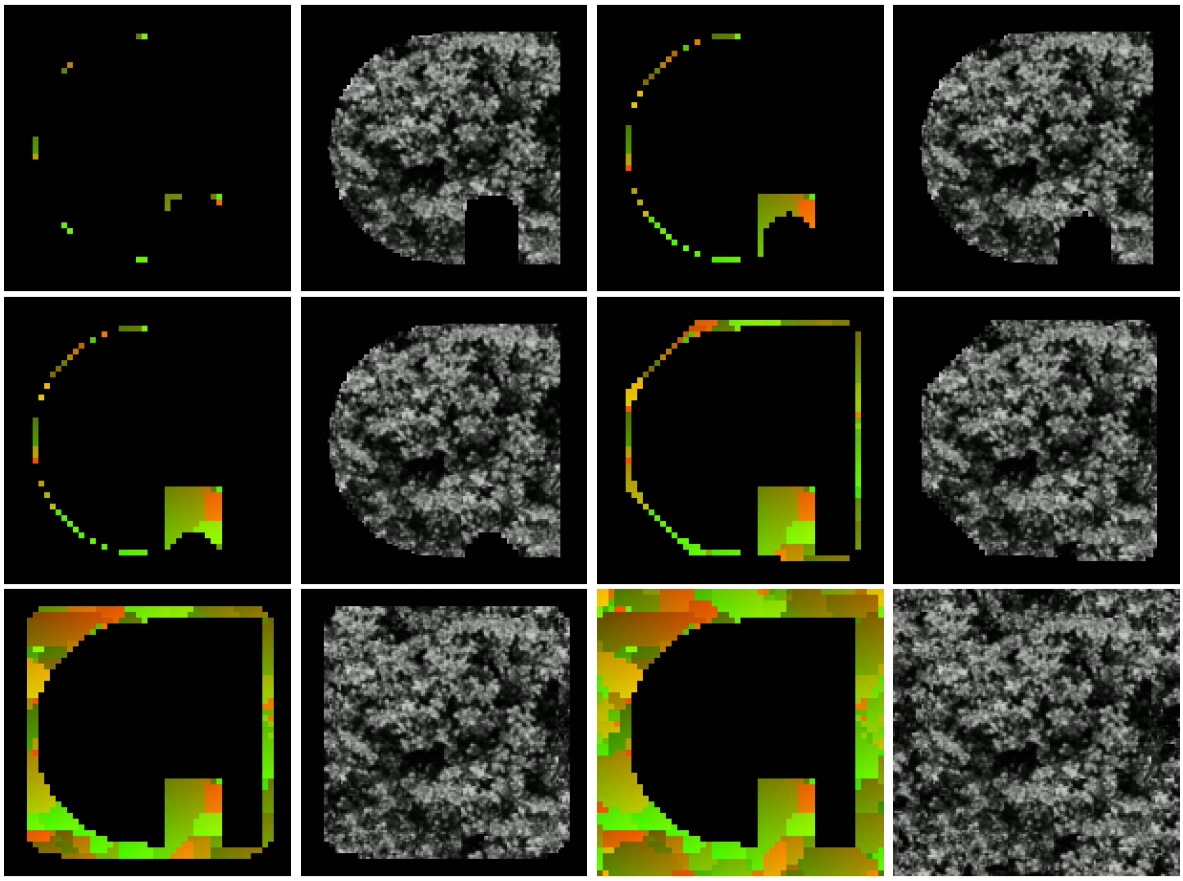


Abbildung 3.3: Synthesesequenz. Jeweils links: Kopierkarte, rechts Syntheseprodukt.

deren Nachbarschaft ein Mindestmaß an definierten Pixeln besitzt. Als minimale Besetzung kann z.B. 1/4 der Gesamtzahl der Nachbarschaftspixel gewählt werden, was einem Eckpunkt in der bereits bestehenden Textur entspricht. Der Ablauf erfolgt dabei ohne Neusortierung der Liste nach Synthese eines einzelnen Blocks. Erst wenn keine Blocknachbarschaft das Besetzungskriterium mehr erfüllt, wird die Liste entsprechend des aktuellen Synthesezustands neu geordnet.

Diese alternative Strategie kommt ohne die speicherintensive, spatiale Indextabelle aus und erhöht die Synthesegeschwindigkeit durch deutlich weniger Sortierungsläufe. Die Ergebnisse erreichen in etwa die Qualität der „Puzzle“-Strategie.

3.2.6 Behandlung der Bildränder

Wie in Abbildung 3.1 zu sehen, sind die Nachbarschaftsumgebungen der Syntheseblöcke nicht immer vollständig definiert. Dies gilt an den Kantenbereichen der zu erweiternden Texturstücke und insbesondere auch an den vier Bildrändern von I_0. Würde man diesen

Sachverhalt ignorieren und den Vergleich der Blocknachbarschaften auch mit undefinierten Pixeln durchführen, wäre der Syntheseprozeß nicht *kausal*, da auf Information zugegriffen wird, die zum jeweiligen Zeitpunkt noch nicht vorhanden ist. Um Kausalität der Synthese zu gewährleisten, wird daher mit einer variablen Maske gearbeitet, die für den Nachbarschaftsvergleich die undefinierten Pixel von Synthese- *und* Referenzblock-Nachbarschaft maskiert. In Abbildung 3.1 sind zwei Beispiele für variable Masken in hellgrau (C) eingezeichnet. Die Bezeichnung „variabel“ beschreibt, daß die Form der Maske für jeden einzelnen Syntheseblock neu bestimmt werden muß, da sie sich aus der Historie des bisherigen Syntheseverlaufs ableitet und nicht allein aus der initialen Arbeitsmaske berechnet werden kann.

Die Bildränder könnten prinzipiell mit der gleichen Methode behandelt werden. Sehr viel eleganter ist es jedoch, die Bildfläche I_0 als Torus zu betrachten. Durch Modulo-Operation beim Koordinatenzugriff

$$(x, y) = (x \ mod \ I_l^{width}, y \ mod \ I_l^{height}) \tag{3.1}$$

erhält man torodiale Anschlußbedingungen, d.h. linker und rechter, sowie oberer und unterer Bildrand sind identisch. Eine Blocknachbarschaft, die links aus dem Bild herausragt, adressiert dann Pixel an dem rechten Bildrand. Durch das Synthetisieren auf einer Torusoberfläche ergibt sich ein weiterer entscheidender Vorteil: das Syntheseprodukt läßt sich an sich selbst anschließen, um noch größere Flächen über eine Kachelung zu formen, siehe Abschnitt 4.2.

3.3 Synthese mit einer Auflösungspyramide

Alle Synthesealgorithmen, die implizit oder explizit mit Markov-Zufallsfeld-Texturmodellen arbeiten, haben das Defizit, daß sie nur Strukturen in der Größe des Analysefensters korrekt wiedergeben können. Dies gilt auch für das in Abschnitt 3.2 beschriebene Verfahren. Um globale Textureigenschaften zu erfassen, müssen die Dimensionen des Nachbarschaftsfensters auf das größte wiederzugebende Texturmerkmal angepaßt werden.

Für die Qualität der Synthese mit Randbedingungen in Form von teil-texturierten Oberflächen, die vervollständigt werden sollen, hat die Größe des Nachbarschaftsfensters eine weitere Bedeutung. Sie bestimmt den Abstand, über den verschiedene Texturteile aneinander angeglichen werden können und muß so gewählt werden, daß das Fenster fehlende Bereiche zwischen Texturregionen überbrücken kann, um ein optimales - geschlossenes - Syntheseergebnis zu erhalten.

Die aus diesen beiden Punkten folgende Forderung nach großen Nachbarschaftsfenstern hat den Nachteil, daß sich der Rechenaufwand dramatisch erhöht.

Eine bessere Methode ist die getrennte Verarbeitung von niedrigen und hohen Frequenzanteilen der Textur in einer Auflösungspyramide. Dabei werden die globalen Merk-

male einer Textur in einer niedrigen Auflösungsstufe optimal reproduziert und diese „grobe“ Textur beim Fortschreiten von niedrigen zu hohen Auflösungsstufen immer weiter verfeinert. Die effektive Fenstergröße der Auflösungspyramide bestimmt sich durch die Fenstergröße in der niedrigsten Auflösungsstufe multipliziert mit der 2er-Potenz der Anzahl der folgenden Auflösungsstufen in der Pyramide: $W_N^{eff} = W_N \cdot 2^{L-1}$. Eine Pyramide mit sechs Auflösungsstufen erreicht somit ein $2^5 = 32$-fach größeres Nachbarschaftsfenster.

Für den Algorithmus bedeutet dies, daß die zu bearbeitende, unvollständige Textur und die optionale Texturreferenz in eine Pyramide zerlegt werden und die niedrigste Auflösungsstufe der Pyramide mit einer einfachen Synthese, wie in Abschnitt 3.2 beschrieben, erstellt wird. Alle weiteren Pyramidenebenen werden ebenso verarbeitet, jedoch muß die quadratische, symmetrische Blocknachbarschaft für eine Verbindung zwischen den aufeinanderfolgenden Auflösungsstufen erweitert werden, um die Syntheseergebnisse der jeweils vorhergehenden Auflösungsstufe zu berücksichtigen. Auf diese Weise werden niederfrequente Strukturen über die Pyramidenstufen propagiert und bleiben bis zur höchsten Auflösungsstufe erhalten.

3.3.1 Erzeugung der Pyramide

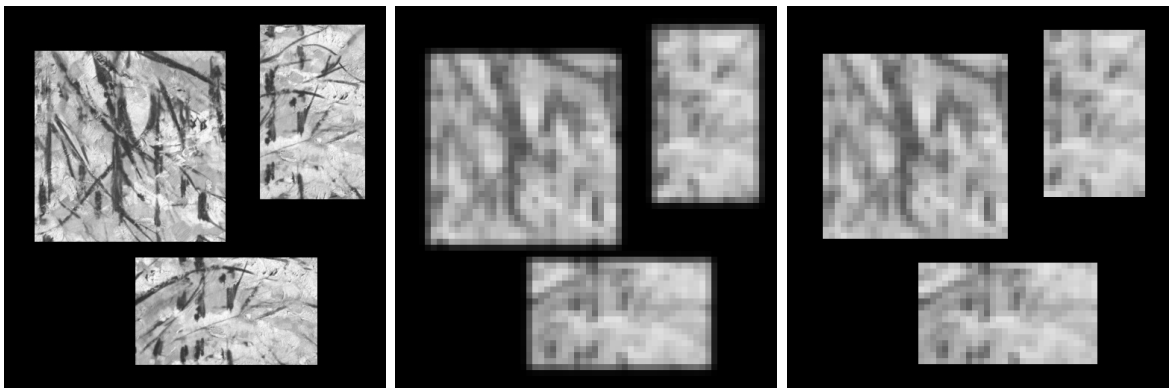


Abbildung 3.4: Filterung. I_0 Eingangsdaten, I_3 Gauß-gefiltert, I_3 Gauß-gefiltert mit Maskenverarbeitung.

Die im Grundlagenkapitel 2 beschriebene Methode zur Erstellung einer Auflösungspyramide eignet sich gut für die frequenzabhängige Zerlegung einer vollständig definierten Referenztextur, wie z.B. in Abbildung 2.1 rechts zu sehen. Für die Synthese mit Randbedingungen müssen aber auch Bilddaten mit unvollständigen Texturen in eine Pyramide zerlegt werden. Dort genügt eine Behandlung der Bildkanten durch Spiegelung nicht, da sich auch innerhalb der Bildfläche Berandungen der Texturflächen befinden, die durch eine Arbeitsmaske beschrieben werden, siehe Abbildungen 3.2 und 3.4. Wendet man die in Abschnitt 2.1 beschriebene Erzeugung einer Bildpyramide auf derartige Eingangsdaten an, werden an den Begrenzungen der Texturflächen Anteile des Hintergrundes mit

Anteilen der Textur vermischt. Da der Hintergrund typischerweise schwarz ist, führt dies zu einer Abdunkelung und Ausdehnung der Texturflächenränder in den Pyramidenstufen $I_{l>0}$, siehe Abbildung 3.4 Mitte.

Um die Vermischung von Hintergrund und Textur in den Randbereichen zu verhindern, wird parallel zur Gauß'schen Filterung die Arbeitsmaske ausgewertet, die den Verlauf der Texturränder kennzeichnet. Nicht definierte Anteile im Term I_l der Gleichung 2.3 werden, wenn möglich, durch Spiegelung erzeugt oder es wird anstelle der Faltung mit 2.2 eine Mittelwertbildung durchgeführt. In Abbildung 3.4 rechts ist ein Ergebnis einer Filterung mit gleichzeitiger Maskenverarbeitung dargestellt.

3.3.2 Erweiterung des Nachbarschaftssystems

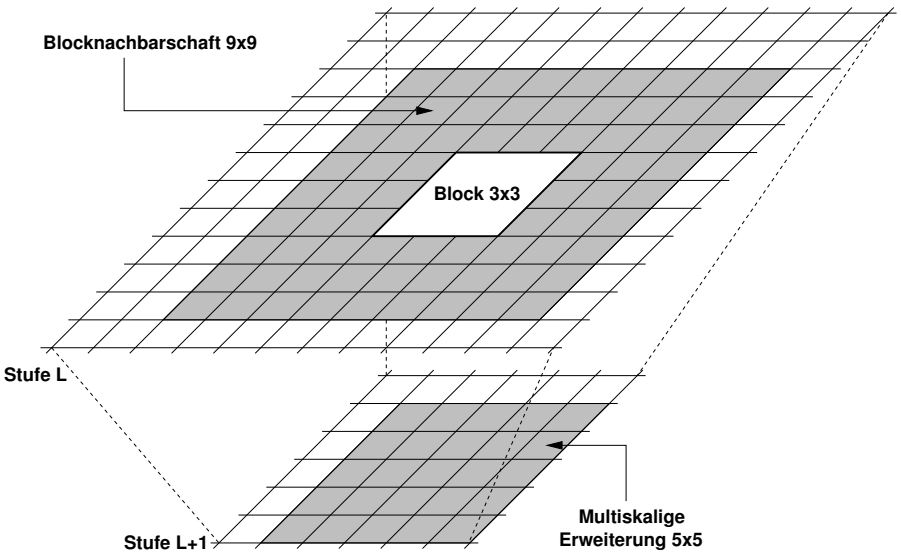


Abbildung 3.5: Symmetrische, multiskalige Blocknachbarschaft.

Um die Ergebnisse einer Pyramidenstufe I_{l+1} mit niedrigerer Auflösung bei der Synthese von I_l nutzen zu können, muß das bisher verwendete symmetrische Nachbarschaftssystem um eine multiskalige Komponente erweitert werden. Grundsätzlich muß diese Erweiterung nur die jeweils niedrigere Auflösungsstufe I_{l+1} berücksichtigen, da diese durch die hierarchische Anordnung die Textur/Frequenz-Informationen **aller** niedrigeren Auflösungsstufen $I_{k>l}$ beinhaltet. Ein erweitertes Nachbarschaftssystem ist in Abbildung 3.5 dargestellt.

Beide Anteile der Nachbarschaftsumgebung haben immer ungerade Kantenlängen W_N und W_N^{erw}. Die Größe des Anteils in Auflösungsstufe I_{l+1} wird so angepaßt, daß sie größer gleich der Fläche des Anteils in I_l ist. Da I_{l+1} durch Unterabtastung mit Faktor 2 aus I_l hervorgegangen ist, ergibt sich die Tabelle 3.1 für die Kantenlängen W_N und W_N^{erw}.

Stufe I_l ($p_1 = W_N \times W_N$)	Stufe I_{l+1} ($p_2 = W_N^{erw} \times W_N^{erw}$)	Pixelanzahl ($= p_1 - W_B^2 + p_2; W_B = 3$)
5×5	3×3	25
7×7	5×5	65
9×9	5×5	97
11×11	7×7	161
13×13	7×7	209
15×15	9×9	297

Tabelle 3.1: Multiskalige Nachbarschaftsgrößen und Gesamtzahl der Pixel einer Nachbarschaft bei einer Syntheseblockgröße von $W_B = 3$.

Beim Ähnlichkeitsvergleich einer Synthese- mit allen Referenznachbarschaften werden die Distanzmaße beider Nachbarschaftsanteile separat berechnet. Die Gewichtung der Anteile zueinander kann vom Anwender festgelegt werden. Standardmäßig wird versucht, eine Gleichgewichtung entsprechend der Flächenverhältnisse der beiden Nachbarschaftsanteile zu erreichen.

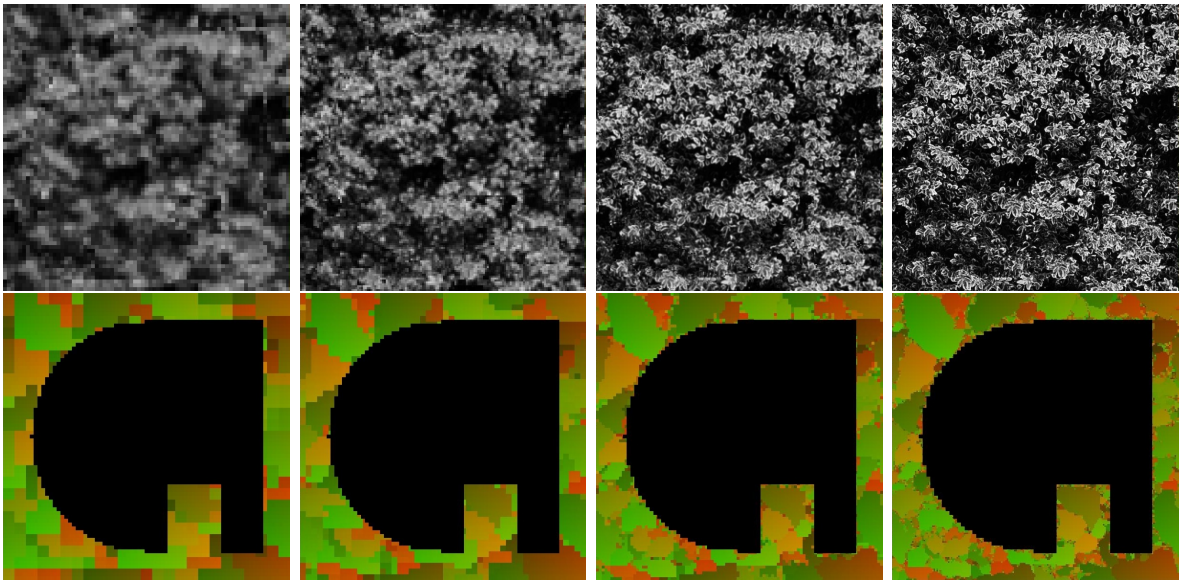


Abbildung 3.6: Multiskalige Textursynthese.

Abbildung 3.6 zeigt die Synthese von vier aufeinanderfolgenden Auflösungsstufen zusammen mit der korrespondierenden Kopierkarte. Die Kopierkarten zeigen sehr deutlich, daß die Synthese nicht nur einzelne Texturblöcke überträgt, sondern daß diese Blöcke größere zusammenhängende Texturregionen formen. Dabei bleiben die initial definierten Gebiete über die Auflösungshierarchie erhalten. Die Randbereiche bzw. die Übergänge zwischen zwei benachbarten Texturbereichen werden von Stufe zu Stufe immer weiter verfeinert.

3.3.3 Koordinatenquantisierung

Durch die effiziente Repräsentation der Ein- und Ausgabetextur in einer Pyramide wird die Synthese sowohl in Qualität als auch in der Ausführungsgeschwindigkeit verbessert. Der Zugriff auf die Teilbänder der Pyramide mit einer erweiterten Nachbarschaftsumgebung erfordert allerdings eine genaue Betrachtung von möglichen Quantisierungsproblemen. Wie im Grundlagenabschnitt 2.1 beschrieben, wird eine niedrigere Auflösungsstufe der Pyramide durch Tiefpaßfilterung mit Halbierung der Grenzfrequenz und Unterabtastung mit Faktor 2 konstruiert. Bei der Unterabtastung geht keine Information verloren - nach dem Abtasttheorem läßt sich das tiefpaßgefilterte Bild durch Überabtastung und Interpolation fehlerfrei rekonstruieren. An dieser Stelle deutet sich aber ein erstes Problem für die Nutzung der erweiterten Nachbarschaftsumgebung an. Aus Effizienzgründen arbeitet der Erweiterungsanteil direkt auf den niedrig aufgelösten, unterabgetasteten Bilddaten I_{l+1}, wobei auf eine zeitaufwendige Interpolation verzichtet wird. Daher gibt es eine direkte, feste Zuordnung von Bildpunkten $P_a = (x_a, y_a)$ in I_l und zugehörigen Bildpunkten $P_b = (x_b, y_b)$ in I_{l+1} durch $(x_b, y_b) = (x_a/2, y_a/2)$. Dabei sind x_a, y_a, x_b, y_b ganzzahlige Koordinaten. Der Quantisierungsfaktor 2 bei der Koordinatenberechnung bewirkt, daß eine Verschiebung eines Nachbarschaftsanteils in I_l um 1 nur an jeweils geraden Koordinaten (x_a, y_a) zu einer Verschiebung des Erweiterungsanteil in I_{l+1} führt.

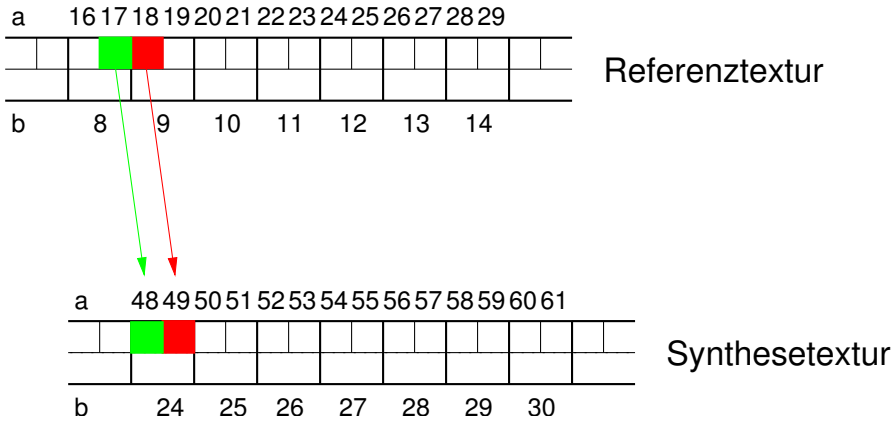


Abbildung 3.7: Koordinatenquantisierung. Oben Texturquelle, unten Syntheseposition. Eindimensionale Darstellung der Koordinaten $[a, b]$ mit $b = a/2$.

Anhand der Abbildung 3.7 soll eine mögliche Auswirkung der festen Zuordnung zwischen I_l und I_{l+1} erläutert werden. Es genügt die Betrachtung des eindimensionalen Problems in der Notation $\square[\square_a, \square_b]$ mit derselben Bedingung $\square_b = \square_a/2$. Der Syntheseprozeß soll zunächst aufgrund der Ähnlichkeit der Nachbarschaftsumgebungen ein Pixel von $R[17, 8]$ der Referenz auf die Koordinate $S[48, 24]$ der Ausgabetextur kopiert haben. Wenn dieses Pixel Teil einer zusammenhängenden Region wäre, sollte nun das Pixel $R[18, 9]$ der Referenz angeschlossen, d.h. auf $S[49, 24]$ der Ausgabetextur kopiert werden. Für den Nachbarschaftsvergleich wird nun die Umgebung um $S[49, 24]$ gebildet und mit allen Referenznachbarschaften verglichen. Der Erweiterungsanteil der Nachbarschaft um

$S_b = 24$ in I_{l+1} korrespondiert, da sich aufgrund der Quantisierung die Koordinate S_b nicht geändert hat, am besten mit $R_b = 8$ der Referenz. Das am besten passende Pixel der Referenz liegt jedoch in $R[18, 9]$, so daß es abhängig von der Referenztextur unter Umständen eine andere Referenz-Position gibt, die aufgrund der gewichteten Bewertung der beiden Nachbarschaftsanteile eine bessere Gesamtbewertung erhält. Ein derartiger Wechsel in der Position der Referenzquelle kann sich nun bei jedem weiteren Vorrücken der Syntheseposition wiederholen und führt zu Oszillationen, die in der Kopierkarte sichtbar werden.

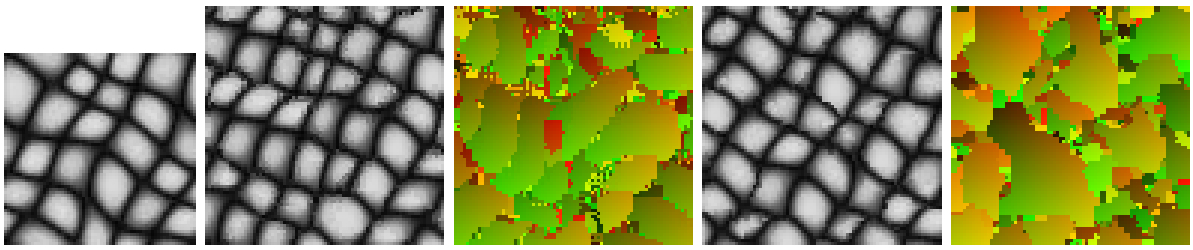


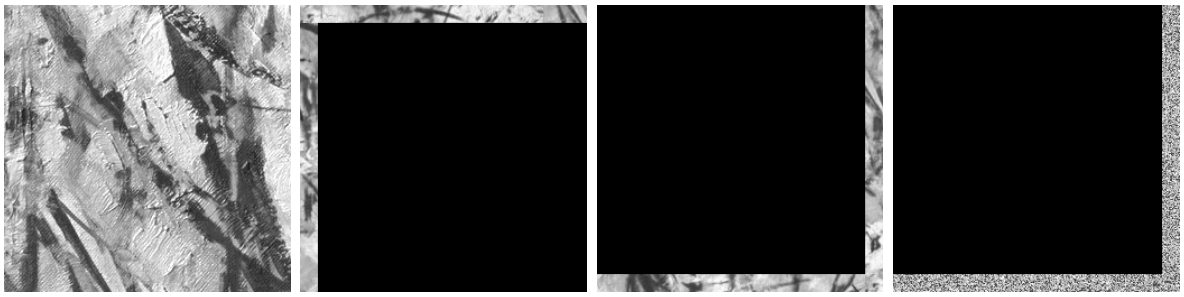
Abbildung 3.8: Quantisierungsoszillationen der multiskaligen Synthese. Links Referenztextur (S01, 64 × 64), Mitte: gestörtes Syntheseergebnis (80 × 80) mit in der Kopierkarte sichtbaren Oszillationen, rechts: Resultat mit Paritätskontrolle und stärkeren Kantenartefakten.

Für bestimmte Texturen konnten diese Oszillationen der multiskaligen Synthese tatsächlich beobachtet werden. In zwei Dimensionen zeigt sich der Wechsel der Texturquelle in Form von neben- oder untereinander angeordneten Bändern, bzw. schachbrettartig strukturierten Flächen in der Kopierkarte, siehe Abbildung 3.8 Mitte. Das abwechselnde Kopieren aus zwei unterschiedlichen Referenztexturbereichen führt bei einigen Texturen zu einer Verschlechterung der Syntheseergebnisse.

Ein Ansatz zur Vermeidung der Quantisierungsartefakte ist eine Paritätskontrolle beim Vergleich von Referenz- und Synthesenachbarschaft. Die Koordinatenparität des Nachbarschaftsanteils in I_l wird geprüft und für den Ähnlichkeitsvergleich nur Referenznachbarschaften mit derselben Parität zugelassen. Dabei gibt es vier unterschiedliche Werte: x-Koordinate gerade oder ungerade kombiniert mit y-Koordinate gerade bzw. ungerade. Diese vier Möglichkeiten erhalten die Paritätswerte $1, 2, 3, 4$. Durch Anwendung der Randbedingung gleicher Parität in Synthese und Referenz verschwinden die Oszillationen, wie in Abbildung 3.8 rechts dargestellt. Gleichzeitig wird die Zahl der Nachbarschaftsvergleiche auf ein Viertel reduziert und die Synthese um Faktor 4 beschleunigt. Nachteil der Paritätsbedingung ist, daß durch die Einschränkung der möglichen Texturreferenzquellen weniger Anschlußmöglichkeiten für Pixel bzw. Blöcke gefunden werden können und somit die Übergänge zwischen zusammenhängenden Regionen ein gröberes Erscheinungsbild mit stärkerer Kantenbildung zeigen. Für eine optimale Synthesequalität muß abhängig von der jeweiligen Textur entschieden werden, ob der Einsatz einer Paritätskontrolle sinnvoll ist oder nicht.

3.4 Initialisierung der Vollsynthese

Am Anfang des Kapitels 3 wurde beschrieben, daß die Textursynthese in Abhängigkeit des Verwendungszwecks mit unterschiedlich initialisierten Eingangsdaten arbeitet. Zunächst wurde der allgemeinere Fall einer teil-initialisierten Textur erläutert und dabei die Komponenten Texturblock, symmetrische Blocknachbarschaft, Referenztextur, Arbeits- und variable Maske, Auflösungspyramide, sowie Kopierkarte und Nutzungsstatistik eingeführt.



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

Abbildung 3.9: Initialisierung der Vollsynthese. a) Referenztextur, b) prinzipielle Anordnung der Initialisierungsstreifen, c) Synthesebeginn an Offset $(0, 0)$ und d) Rauschinitialisierung.

Die Textursynthese einer undefinierten Fläche mittels einer Referenztextur - im folgenden Vollsynthese genannt - ist ein Spezialfall der Synthese mit Randbedingungen. Prinzipiell benötigt auch die Vollsynthese eine Initialisierung der Ausgabefläche, da ein Nachbarschaftsvergleich zwischen Referenz und Syntheseposition nur erfolgen kann, wenn zumindest ein Anteil der Blocknachbarschaft an der jeweiligen Syntheseposition mit Texturinformation definiert ist. Im einfachsten Fall können dies zwei schmale Texturstreifen in Breite des halben Nachbarschaftsfensters sein, wie in Abbildung 3.9 b) zu sehen. Diese Anordnung ermöglicht den spalten- und zeilenweisen Aufbau der Textur mit einer vereinfachten L-förmigen Blocknachbarschaft, siehe Abbildung 3.10. Durch die Definition der Randbereiche ist die Blocknachbarschaft an jeder Syntheseposition kausal, da bei spalten- und zeilenweisem Vorrücken der Syntheseposition nur auf bereits definierte Textur zugegriffen wird. Daher kann auch auf die variable Maske der symmetrischen Blocknachbarschaft verzichtet werden.

Die Position des horizontalen und vertikalen Initialisierungsstreifens ist beliebig - sie definiert lediglich den Offset des Startpunktes der Synthese, die wie beschrieben mit torodialen Anschlußbedingungen arbeitet. Soll die Synthese an der Nullposition des Bildes beginnen, müssen die Streifen an den rechten und unteren Bildrand verschoben werden, siehe Abbildung 3.9 c). Beide Streifen können im Verlauf der Synthese auch überschrieben

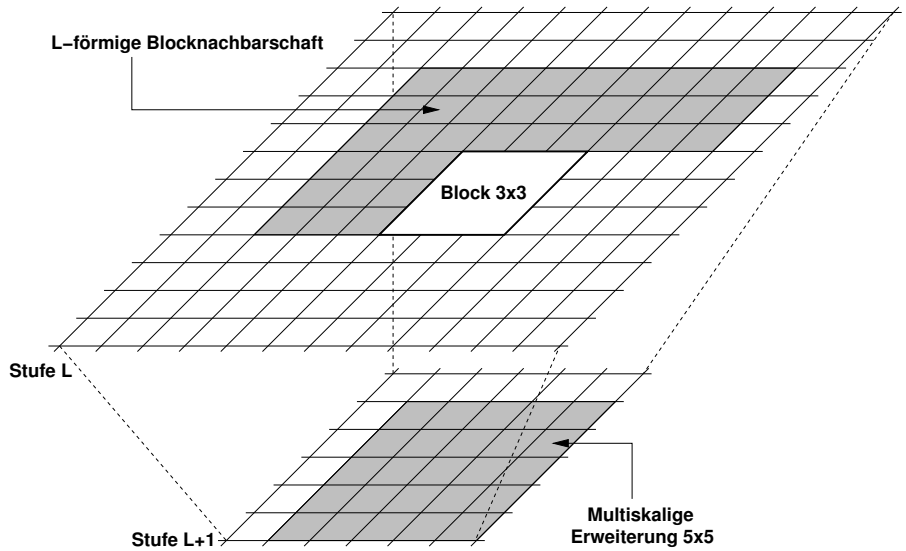


Abbildung 3.10: L-förmige, multiskalige Blocknachbarschaft für die Vollsynthese.

werden, was insbesondere dann sinnvoll ist, wenn sie aus mehreren Teilstücken zusammengesetzt wurden und sichtbare Kantenübergänge aufweisen. Die aufwendige manuelle Präparation der Ausgabefläche mit Bildstreifen läßt sich durch eine Rauschinitialisierung ersetzen. Anstatt Streifen der Referenztextur auf die Ausgabefläche zu kopieren, kann eine zufällige Textur basierend auf den Referenzdaten erzeugt werden. Benötigt wird eine Rauschquelle, die die Histogramm-Verteilung der Referenztextur aufweist. Eine einfache Realisierung ist das Adressieren der Menge der Referenztextur-Bildpunkte mit einer gleichverteilten Rauschquelle und direktes Kopieren auf die Synthesevorlage, siehe Abbildung 3.9 d). Die Grauwertverteilung der Referenztextur bleibt damit zwar erhalten, die Struktur geht jedoch verloren - sie wird erst im Verlauf der Synthese wiederhergestellt.

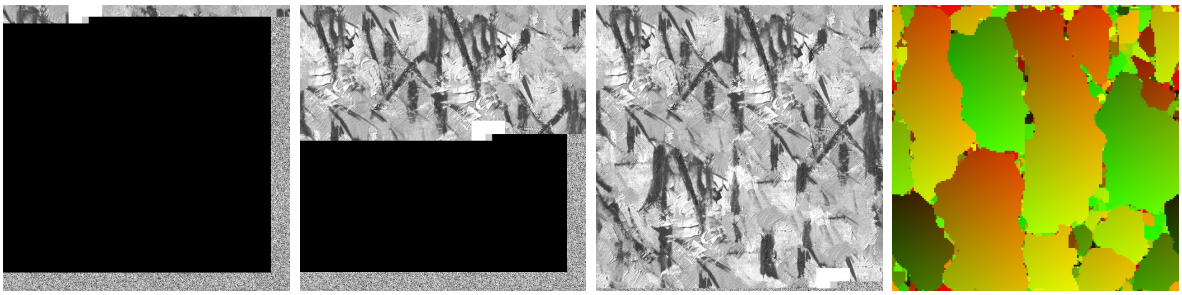


Abbildung 3.11: Vollsynthese. Zwischenergebnisse und Kopierkarte.

Abbildung 3.11 zeigt Momentaufnahmen des Ablaufs einer multiskaligen Synthese der Textur aus Abbildung 3.9 links mit vier Auflösungsstufen. Dargestellt sind die Zwischenergebnisse der höchsten Auflösungsstufe, die Nachbarschaftsumgebung (überhöht) und die finale Kopierkarte.

Das Initialisieren mit strukturfreiem, Histogramm-angepaßtem Rauschen anstelle von Texturstreifen bewirkt ein nicht-deterministisches Entstehen der Textur mit hohen Variationsmöglichkeiten. Gleichzeitig sorgt es jedoch für ein hohes Maß an Störungen in unmittelbarer Nähe der Rauschstreifen, da selten größere zusammenhängende Gebiete der Texturreferenz kopiert werden. Ursache hierfür ist, daß der Nachbarschaftsvergleich von Texturreferenz und Syntheseposition, bedingt durch die Rauschwerte in der Nachbarschaft der Syntheseblöcke, immer wieder zu anderen, optimal anschließenden Blöcken aus der Texturreferenz führt. Sobald der Syntheseprozeß sich von dem horizontalen Rauschstreifen entfernt und auf bereits synthetisierte, „richtige“ Textur zugreift, werden wieder größere zusammenhängende Gebiete übertragen und die hochfrequenten Störungen verschwinden. Da die Rauschinitialisierung nur wenige Bildzeilen bzw. -spalten belegt, sind die beschriebenen Artefakte für den Gesamtprozeß nicht relevant und können bei Bedarf auch nachträglich abgeschnitten werden.

3.5 Einfluß und Optimierung der Verfahrensparameter

In diesem Abschnitt soll die Auswirkung der Variation aller in der Beschreibung des Algorithmus eingeführten Parameter und Größen auf den Syntheseprozeß detailliert diskutiert werden. Neben der Einzelbetrachtung werden auch die Abhängigkeiten zwischen den Parametern beschrieben und Regeln für die Bestimmung der problemspezifisch optimalen Arbeitspunkte aufgestellt.

3.5.1 Nachbarschaft und Bildpyramide

Für die hier beschriebene, implizite Markov-Zufallsfeld Methode gilt wie für alle MRF-Ansätze, daß die exakte Reproduktion der Merkmale der Texturreferenz im Syntheseprozeß von der Größe der verwendeten Nachbarschaftsumgebung abhängt. Wie in Abschnitt 3.3 beschrieben, muß die Breite W_N des (symmetrischen) Nachbarschaftsfensters N so gewählt werden, daß N das größte wiederzugebende Texturmerkmal vollständig erfaßt. Der Einfluß der Größe W_N auf die einfache Vollsynthese mit einer Auflösungsstufe ist in Abbildung 3.12 dargestellt.

Die niedrigste Frequenz und damit größte Wellenlänge im oberen Beispiel in Abbildung 3.12 beträgt in etwa 40 Pixel. Man sieht sehr deutlich, wie die globale Struktur der Synthese sich in Abhängigkeit von der Fenstergröße immer weiter dem Original annähert. Analog dazu äußert sich die Veränderung von W_N in der Kopierkarte durch stärker zusammenhängende Regionen. Je größer das Fenster, desto höher ist der Anteil der zusammenhängenden Flächen, was dazu führt, daß die globale Anordnung der Textur immer besser wiedergegeben wird. Die visuelle Qualität der Syntheseergebnisse nimmt bei sehr großem W_N wieder ab, wie in Abbildung 3.12 rechts ($W_N = 41$) zu sehen. Bei weiterer Erhöhung der Fenstergröße tendiert der Algorithmus dazu, immer wieder

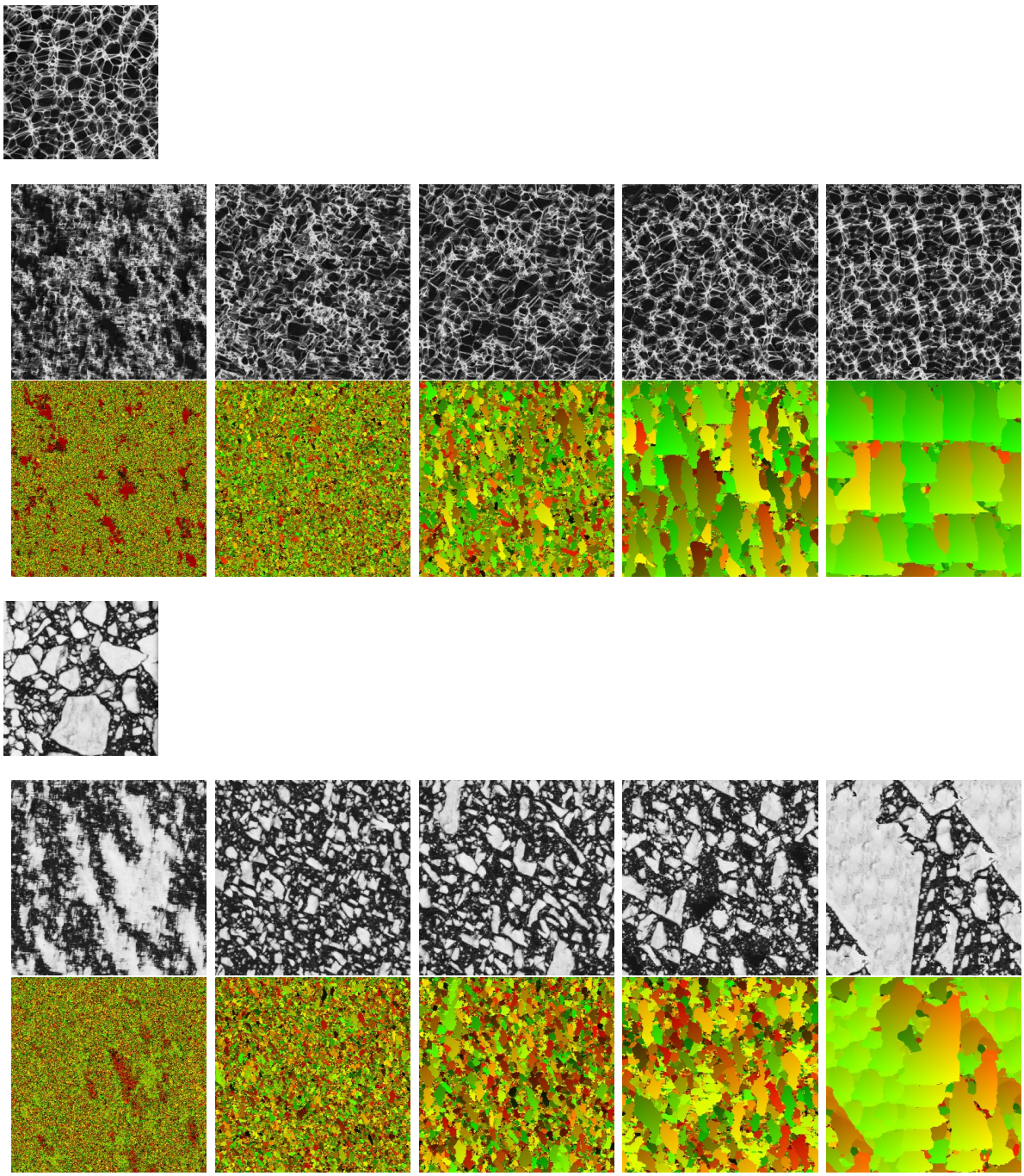


Abbildung 3.12: Einfache Synthese mit einer Auflösungsstufe. Referenztexturen (D111, D62; 256 × 256), Ergebnisse (320 × 320) mit $W_N = 3, 5, 9, 17, 41$ und Kopierkarten.

identische Referenzregionen an sich selbst anzuschließen. Ein Grund für dieses Verhalten liegt in der zunehmenden Beschränkung der Referenztexturfläche, da, wie in Abschnitt 3.2.3 erläutert, das Nachbarschaftsfenster vollständig innerhalb der Texturreferenz liegen

muß. Ein weiterer Grund für die Repetitionen und Maßnahmen zur Beseitigung der Ursachen werden ausführlich in Abschnitt 3.7 diskutiert.

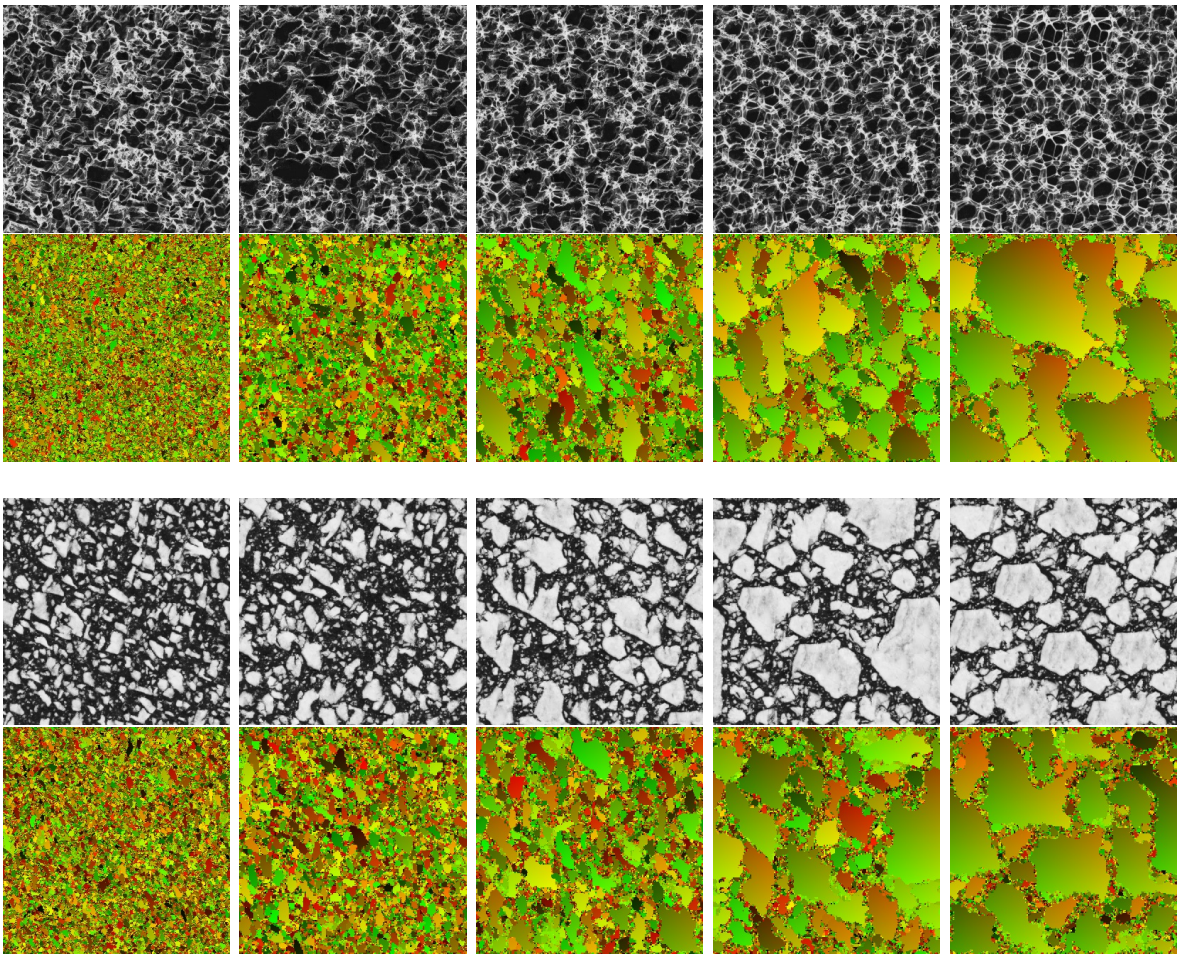


Abbildung 3.13: Synthese mit Auflösungspyramide. Ergebnisse (D111,D62; 320 × 320) mit $W_N = 5, L = 1, 2, 3, 4, 5$ und Kopierkarten.

Die Ergebnisse der einfachen Textursynthese sollen nun den Resultaten einer multiskaligen Textursynthese gegenübergestellt werden. In Abschnitt 3.3 wurde beschrieben, wie sich die effektive Größe des Nachbarschaftsfensters einer Pyramide berechnen läßt: $W_N^{eff} = W_N \cdot 2^{L-1}$, wobei L die Anzahl der Pyramidenstufen ist. Abbildung 3.13 zeigt eine multiskalige Synthese mit $W_N = 5$ und Variation von L. Die Darstellung veranschaulicht den großen Vorteil einer Synthese mit Auflösungspyramide. Trotz des vergleichsweise kleinen Nachbarschaftsfensters von $W_N = 5$ werden die globalen Strukturen schon bei einer Pyramidentiefe von $L = 4 \Longrightarrow W_N^{eff} = 40$ deutlich besser reproduziert als bei jedem Ergebnis aus Abbildung 3.12.

Der Gewinn durch den Einsatz der Auflösungspyramide äußert sich aber nicht nur in der Qualität, sondern auch in der Berechnungszeit, da die Größe der Nachbarschaft

W_N die Rechenzeit quadratisch beeinflußt, die größere Fläche der Auflösungspyramide aber nur linear eingeht. Für eine Abschätzung des Laufzeitfaktors soll von einer großen Textur ausgegangen werden, d.h. die unterschiedliche Ausdehnung der Randbereiche kann vernachlässigt werden. Dann gilt, daß für eine einfache Vollsynthese mit einer Auflösungsstufe, $W_N = 17$ und einer Syntheseblockgröße $W_B = 1$ aufgrund der L-förmigen Nachbarschaft $(17^2 - 1)/2 = 144$ Nachbarschaftspixel verglichen werden müssen. Bei einer multiskaligen Synthese mit vier Auflösungsstufen $L = 4$, $W_N = 5$ bzw. $W_N^{eff} = 40$ ergeben sich aufgrund der erweiterten Nachbarschaftsumgebung nach Tabelle 3.1 $(5^2 - 1)/2 + 3^2 = 21$ zu vergleichende Nachbarschaftspixel. Die Gesamtfläche der hier verwendeten Auflösungspyramide ist immer kleiner als 4/3 der Fläche der höchsten Auflösung. Der Rechenaufwand reduziert sich daher um einen Faktor $144/21 * \frac{3}{4} \approx 5$. Die multiskalige Synthese benötigt bei sichtbar besserer Qualität durch die größere, effektive Nachbarschaft also nur 20 Prozent der Rechenzeit und ermöglicht dadurch die Verarbeitung von sehr großen Texturen, wie sie die Anwendungen in Kapitel 5 erfordern.

Im folgenden wird daher ausschließlich die multiskalige Textursynthese betrachtet. Die multiskalige Synthese mit einer Auflösungspyramide besteht, wie in Abschnitt 3.3 beschrieben, aus zwei aufeinander aufbauenden Schritten. Als erstes wird die Pyramidenstufe mit der niedrigsten Auflösung I_{L-1} mit einer einfachen Synthese erzeugt. Anschließend erfolgt die Verfeinerung des niedrig aufgelösten Syntheseergebnisses I_{L-1} mit höherfrequenten Anteilen der folgenden Auflösungsstufen über eine Synthese mit erweiterter Nachbarschaft N. Wie müssen die Parameter der beiden Teilschritte für ein optimales Ergebnis gewählt werden?

Eine sinnvolle Obergrenze für L bestimmt sich aus der Bildgröße der Eingangsdaten. In der Pyramidenstufe I_{L-1} mit der niedrigsten Auflösung muß genügend Bildmaterial für die einfache Synthese mit Fenstergröße W_N vorhanden sein. Auch hier entsteht ansonsten, bedingt durch die notwendigen Randbereiche für die Referenztextur, eine Repetierung weniger Texturelemente, siehe Abbildung 3.13 unten rechts ($L = 5$). Für die einfache Synthese der niedrigsten Auflösungsstufe I_{L-1} gilt prinzipiell alles, was bisher im Zusammenhang mit der Variation von W_N diskutiert wurde, siehe Abbildung 3.12, wobei $W_N \leq 11$ sein sollte. Wird ein größeres W_N benötigt, um die globale Struktur geeignet zu reproduzieren, sollte aus Effizienzgründen nicht W_N, sondern die Anzahl der Auflösungsstufen L erhöht werden. Eine weitere Bedingung für die Wahl von W_N und damit auch L ergibt sich aus der Größe der Syntheseblöcke. Da die Nachbarschaftsumgebung N den zu übertragenen Synthese- bzw. Referenzblock einschließt und repräsentiert, muß für die Bestimmung des Distanzmaßes der Parameter W_N größer W_B gewählt werden. Abgeleitet aus der Minimalbedingung $W_B = 1, W_N = 3$ wird $W_N \approx 3 \cdot W_B$ gewählt.

Für den zweiten Schritt - die stufenweise Verfeinerung von I_{L-1} über die Auflösungspyramide - reicht theoretisch eine erweiterte Nachbarschaft N mit $W_N = 5$ für I_l und $W_N^{erw} = 3$ für I_{l+1} aus (s. Tabelle 3.1), da die Grenzfrequenz bei aufeinanderfolgenden Pyramidenstufen halbiert wurde und das 3×3-Fenster die höchste auftretende Frequenz in der niedrigen Auflösungsstufe I_{l+1} erfassen kann. Abbildung 3.14 zeigt, daß die globale Struktur der niedrigsten Auflösungsstufe I_{L-1} beim Synthetisieren der höherfrequenten

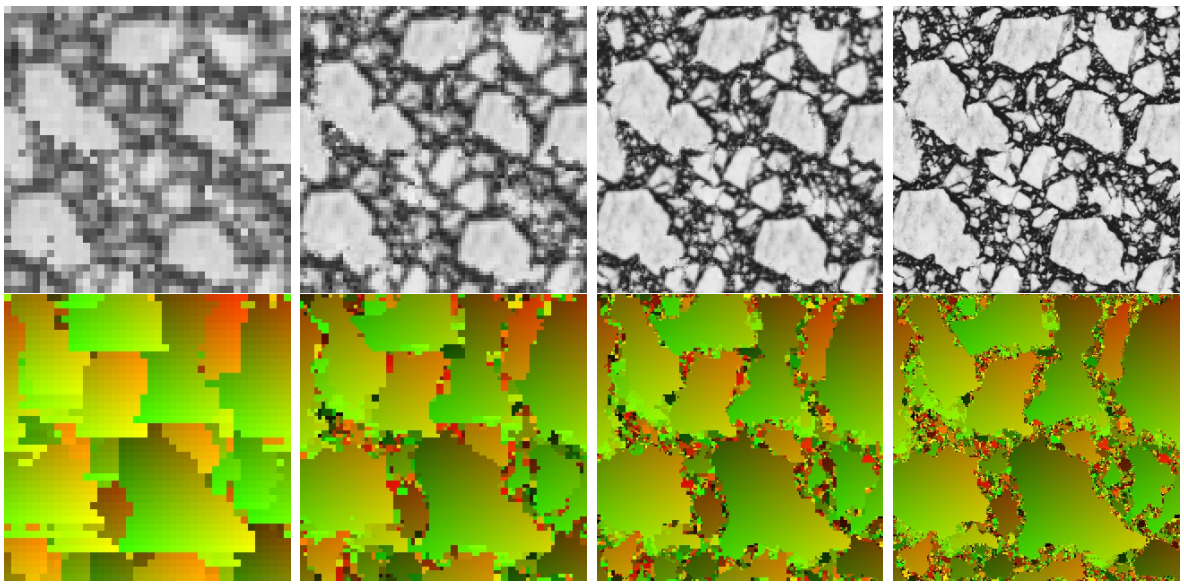


Abbildung 3.14: Synthese von D62 mit $W_N = 5, W_N^{erw} = 3$. Ergebnisse skaliert auf 320×320, Auflösungsstufen $l = 3, 2, 1, 0$ und Kopierkarten.

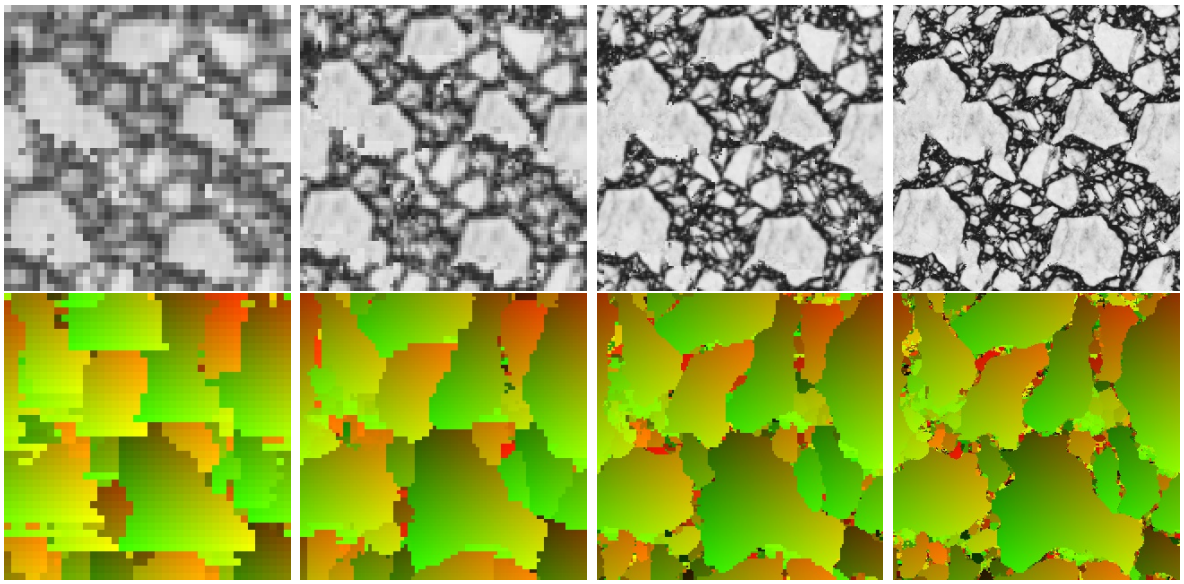


Abbildung 3.15: Synthese von D62 mit $W_N = 11, W_N^{erw} = 7$. Ergebnisse skaliert auf 320×320, Auflösungsstufen $l = 3, 2, 1, 0$ und Kopierkarten.

Auflösungsstufen erhalten bleibt und die Übergänge zwischen den zusammenhängenden Gebieten kontinuierlich durch neue, kleinere Regionen verbessert werden. Durch die verhältnismäßig kleine, kurzreichweitige Nachbarschaftsumgebung mit $W_N = 5$ und $W_N^{erw} = 3$ ergeben sich für die Optimierung der Regionenübergänge vielfältige Auswahlmöglichkeiten für die Texturquelle. Dies erklärt die große Anzahl der kleinen Tex-

turregionen zwischen den größeren zusammenhängenden Flächen in der Kopierkarte.

Wird ausgehend von derselben, niedrigsten Auflösungsstufe I_{L-1} die Reichweite der erweiterten Nachbarschaft N erhöht, z.B. auf $W_N = 11, W_N^{erw} = 7$, siehe Abbildung 3.15, wird durch das größere Fenster der globale Zusammenhang der Elemente besser erfaßt, gleichzeitig aber die Auswahlmöglichkeiten in der Referenztextur stärker beschränkt. Man spricht hier auch von dem *„Fluch der Dimensionalität“* (curse of dimensionality) [100] [55]: bei gegebener Texturreferenz endlicher Größe ist für ein kleines Fenster um die Syntheseposition die Wahrscheinlichkeit, mehrere ähnliche Nachbarschaftsumgebungen zu finden, höher, als für ein großes Fenster, welches das Erscheinungsbild genauer festlegt. Der Wechsel in der Texturquelle erfolgt hier seltener und abrupter, was Diskontinuitäten zur Folge hat. Das Syntheseverfahren modelliert die Übergänge zwischen den zusammenhängenden Flächen nicht durch Einfügen kleinerer Texturregionen, sondern durch Verschiebung der Grenzen der großen, zusammenhängenden Flächen, siehe Abbildung 3.15. Diese Verschiebung erfolgt automatisch so, daß ein optimaler Übergang zwischen den jeweils angrenzenden Flächen erreicht wird. Das Verhalten des Verfahrens entspricht damit der Berechnung eines optimalen Schnitts zwischen zwei überlappenden Regionen, wie es z.B. beim *Image Quilting* [31] durchgeführt wird.

Ein weiterer Parameter der multiskaligen Synthese ist der Gewichtungsfaktor für die Erweiterung des Nachbarschaftssystems N, siehe Abschnitt 3.3.2. In der Praxis wird standardmäßig eine Gleichgewichtung beider Anteile eingestellt, d.h. der Erweiterungsanteil wird mit dem Quotienten der Pixelzahl beider Anteile multipliziert. Dieser Parameter kann aber auch frei gewählt werden. Es ist möglich, durch niedrigere Gewichtung der Nachbarschaftserweiterung die Wiedergabe der lokalen Strukturen zu verbessern - insbesondere die erwähnten Übergänge zwischen den großen, zusammenhängenden Texturregionen aus den niederfrequenten Auflösungsstufen. Gleichzeitig nimmt dann aber die globale Kohärenz ab. Andererseits kann eine stärkere Betonung auf die exakte Reproduktion der globalen Texturmerkmale gelegt werden, was im Gegenzug zu verstärktem Auftreten von Artefakten in der lokalen Struktur führt, wie z.B. starke Gradienten an dem Übergang zwischen zwei Texturflächen. Vorteilhafterweise werden Artefakte an den Übergängen zwischen Texturregionen beim Fortschreiten der Synthese über die aufeinanderfolgenden Auflösungsstufen der Pyramide abgeschwächt.

Abbildung 3.16 zeigt eine Variation des Gewichtungsfaktors. Bei einer 4-fachen Verstärkung des Erweiterungsanteils wird die globale Struktur der niederfrequenten Auflösungsstufe I_{L-1}, links im Bild, positionsgenau wiedergegeben. Dies gilt insbesondere für die beiden dunkleren Texturmerkmale im unteren Bildbereich. Gleichzeitig zeigen sich Artefakte in der lokalen Struktur - die Übergänge zwischen den einzelnen, zusammenhängenden Regionen sind erkennbar. Bei Gleichgewichtung mit Faktor 1 werden die lokalen Fortsetzungen der Textur deutlich besser synthetisiert und durch einen Faktor 0.5 nochmals optimiert. Letzteres führt allerdings zu einer Veränderung der ursprünglichen globalen Anordnung, was bei einer multiskaligen Synthese nicht erwünscht ist.

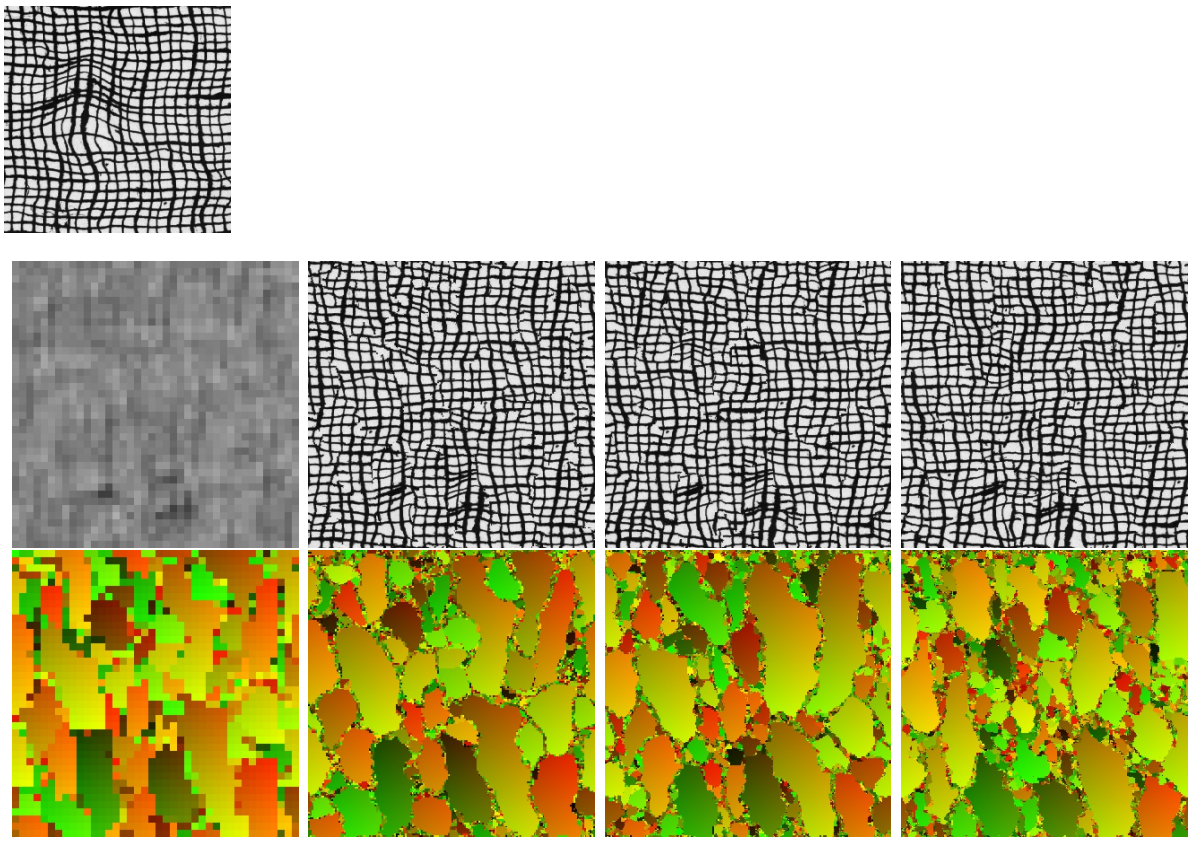


Abbildung 3.16: Gewichtung der Nachbarschaftsanteile. Referenztextur (D104, 256 × 256), Auflösungsstufe $l = 3$ skaliert auf 320 × 320 als Basis, Syntheseergebnisse mit $W_N = 7, L = 4$ und Gewichtungsfaktoren $4.0, 1.0, 0.5$.

3.5.2 Texelgröße

Wie in Abschnitt 3.2 beschrieben, überträgt der Synthesealgorithmus nicht einzelne Pixel von der Texturreferenz auf die Ausgabefläche, sondern kleine, zusammenhängende quadratische Blöcke. Der Vorteil dieses Ansatzes ist, daß durch die Übertragung einer kompakten Pixelgruppe die lokale Kohärenz innerhalb des Blocks erhalten bleibt. Im Vergleich zu einer Pixelsynthese bedeutet dies niedrigeres Bildrauschen, weil eine stärkere Kontinuität bei der Positionswahl in der Texturquelle besteht. Es wurde bereits gezeigt, daß das Verfahren für die Reproduktion der Globalstruktur größere, zusammenhängende Regionen aus der Referenz kopiert. Innerhalb dieser Regionen liefern Block- und Pixeltransfer identische Ergebnisse. Grundsätzlich verringert der Blocktransfer aber die laterale Auflösung der Kopierkarte und damit auch die Auswahl der Anschlußmöglichkeiten in den Übergangszonen zwischen den großen Regionen, was in Form von Kantenartefakten sichtbar werden kann. Diese Artefakte werden bei einer multiskaligen Synthese von einer Auflösungsstufe zur nächsten verringert und sind in der höchsten Auflösung meist nur in der feinen Detailinformation auszumachen. Abbildung 3.17 zeigt ausgehend

von der gleichen niedrigen Auflösungsstufe I_3 eine Variation der Texelgröße bei gleicher Nachbarschaftsgröße $W_N = 11$. Die Wahl einer optimalen Blockgröße W_B hängt von der Ausprägung und der Größe der zu synthetisierenden Textur zusammen. Inhomogen strukturierte Texturen erfordern eine kleinere Blockgröße, regelmäßige sowie stochastische Texturen können mit größeren Transferblöcken erzeugt bzw. qualitativ verbessert werden, siehe Abbildung 3.18.

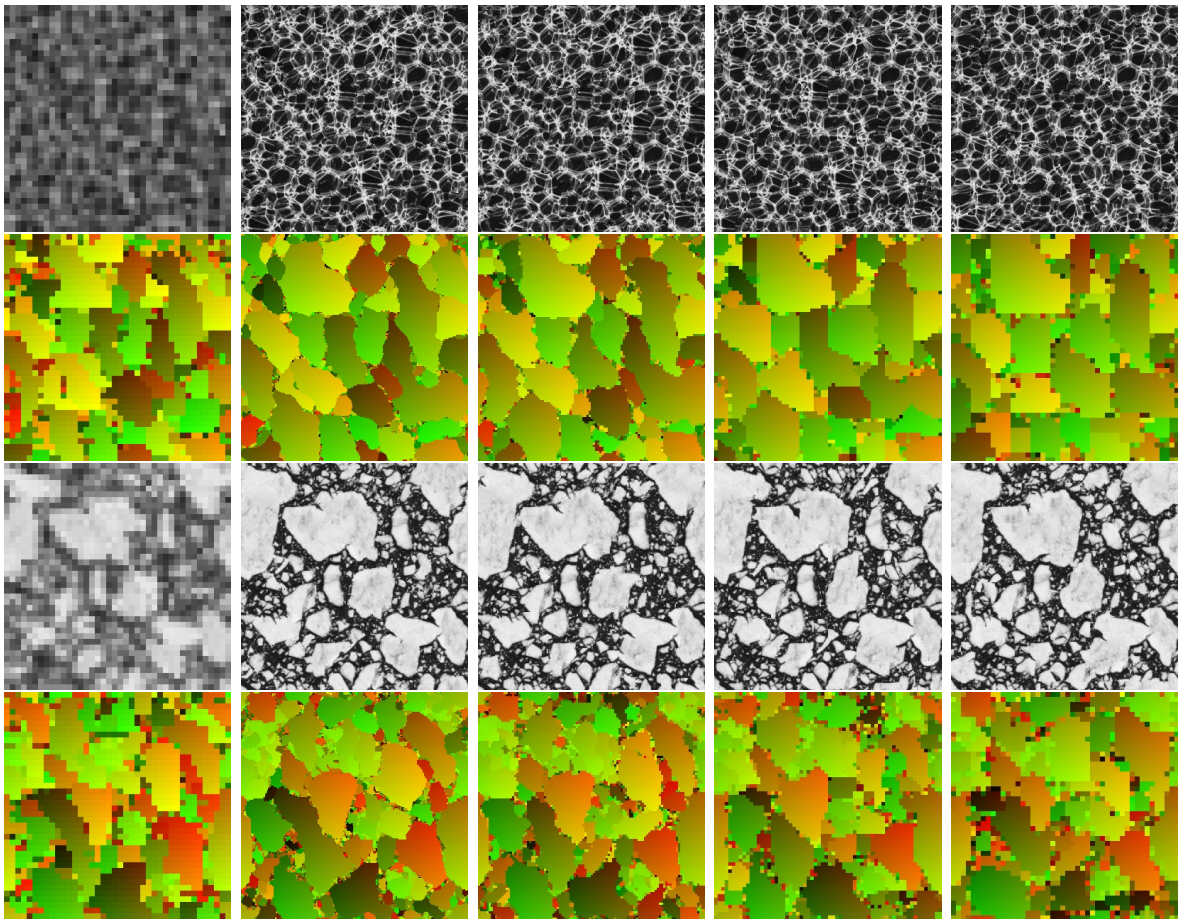


Abbildung 3.17: Variation der Texelgröße. Auflösungsstufe $l = 3$ skaliert auf 320×320 als Basis, Syntheseergebnisse mit $L = 4, W_N = 11, W_B = 1, 3, 5, 7$ (D111, D62).

Neben der lokalen Kohärenz der Syntheseblöcke ist der wesentliche Vorteil der Blockbasierten Synthese die höhere Ausführungsgeschwindigkeit. Schon bei einer Blockgröße von 3×3 wird das Verfahren gegenüber einer Pixelsynthese um fast eine Größenordnung beschleunigt. In der Praxis hat sich gezeigt, daß die meisten Texturen ohne signifikanten Qualitätsverlust mit einer Blockgröße von 3×3 oder 5×5 synthetisiert werden können, was gegenüber anderen Verfahren eine Beschleunigung von Faktor 9 bzw. 25 ermöglicht.

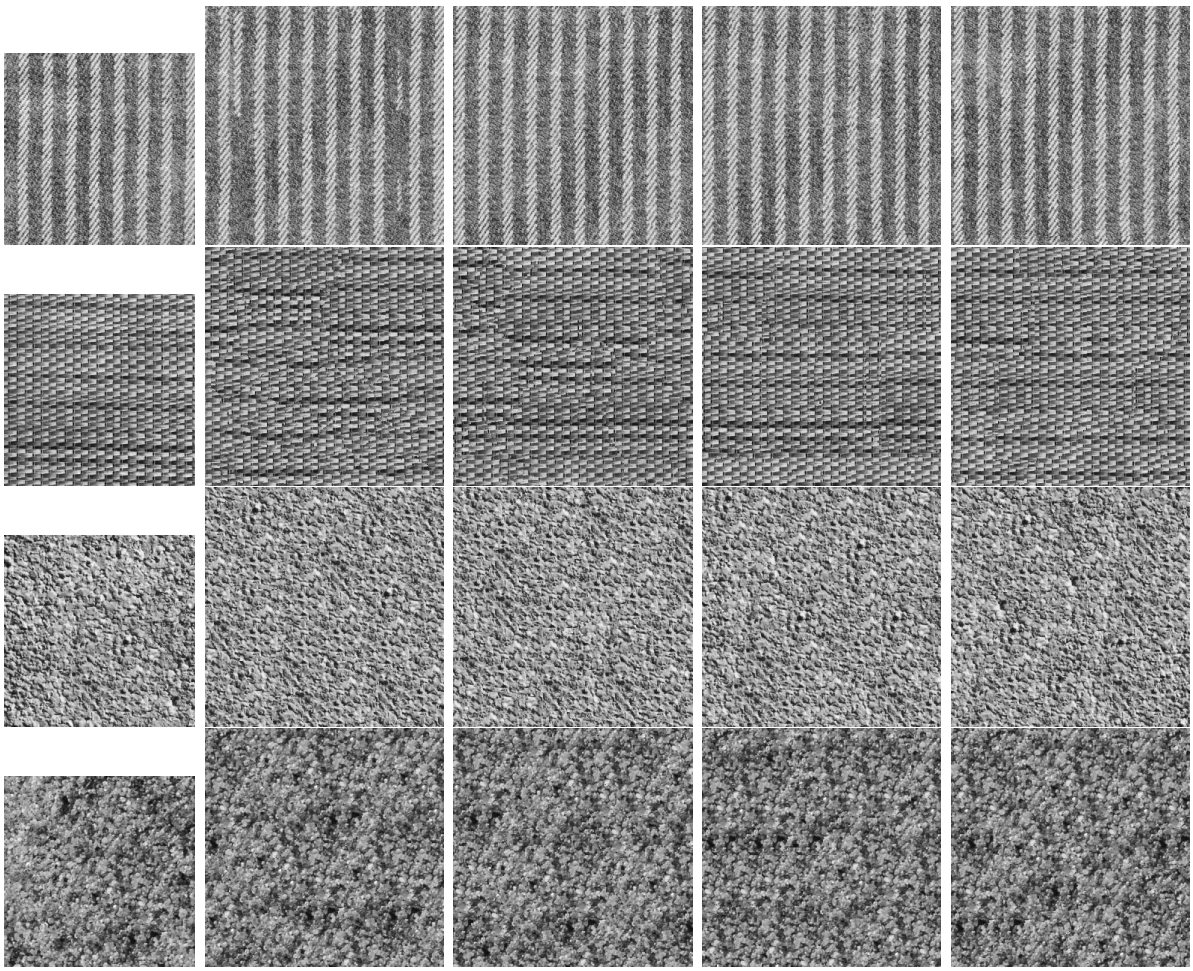


Abbildung 3.18: Variation der Texelgröße bei regelmäßigen und zufälligen Texturen. Referenzdaten (D11, D55, D4, D28, 256×256), Synthese (320×320) mit $L = 4$, $W_N = 11$, $W_B = 1, 3, 5, 7$.

3.5.3 Metriken

Im Grundlagenkapitel Abschnitt 2.2 wurde der Begriff der Bildähnlichkeit eingeführt und in Abschnitt 3.2 in Form eines berechenbaren Distanzmaßes zur Messung der Ähnlichkeit von zwei Blocknachbarschaften verwendet. Das Distanzmaß ist essentieller Bestandteil des impliziten MRF-Ansatzes, da zur Bestimmung des optimal passenden Texturblocks für eine beliebige Syntheseposition die Umgebung dieser Position mit den Umgebungen aller Referenztexturblöcke verglichen werden muß.

Für das Verfahren stellt sich nun das Problem, eine Vorschrift zur Messung der Ähnlichkeit zweier Bildregionen zu entwickeln, welche die Berechnung jedes einzelnen Abstands so effizient wie möglich gestaltet. Eine Referenztextur der Größe 1000×1000 erfordert rund eine Million Distanzmaßberechnungen für die Synthese eines einzelnen Blocks. Die

Forderung nach Effizienz bedeutet auch, daß Zugeständnisse bezüglich der optimalen Anpassung an die Eigenschaften des menschlichen Sehapparats gemacht werden müssen. Dies bedingt letztlich schlechtere, visuelle Resultate der Textursynthese, die prinzipiell eine höhere Qualität liefern könnte, was momentan für die Verarbeitung von großen Texturen aber nicht realisierbar ist.

Ein häufig verwendetes Maß zum Bestimmen der Ähnlichkeit von zwei Bildregionen ist der euklidische Abstand, die L_2-Norm. Bei der Berechnung wird die Summe der quadratischen Grauwertdifferenzen der Pixel bestimmt - je kleiner die L_2-Norm, desto ähnlicher sind sich die Blocknachbarschaften $N(B^{syn})$ und $N(B^{ref})$:

$$L_2(N(B^{syn}), N(B^{ref})) = \sqrt{\sum_{i=0}^{l} \left(P_i^{syn} - P_i^{ref}\right)^2}. \tag{3.2}$$

Die L_2-Norm ist keine befriedigende Approximation für den menschlichen Seh- und Erkennungsprozeß, da Kantenverläufe und Positionen von Ecken nicht empfindlich genug detektiert werden. Für die Synthese bedeutet dies, daß starke Gradienten der Texturreferenz im Ergebnis nicht exakt genug aneinander angeschlossen und fortgesetzt werden, was insbesondere bei der Reproduktion von stark strukturierten, regelmäßigen Strukturen zum Tragen kommt. Entscheidender Vorteil der L_2-Norm ist die einfache Berechenbarkeit. Da nur ein relatives Abstandsmaß benötigt wird, kann auf das Radizieren verzichtet und L_2^2 ausschließlich mit Multiplikationen und Additionen bestimmt werden.

Eine weitere, einfache Realisierung eines Distanzmaßes ist die Anwendung der im Bereich der Videokodierung beliebten L_1-Norm, d.h. Berechnung der Summe der absoluten Grauwertdifferenzen der Pixel in der Nachbarschaftsumgebung:

$$L_1(N(B^{syn}), N(B^{ref})) = \sum_{i=0}^{l} |P_i^{syn} - P_i^{ref}|. \tag{3.3}$$

Die Bestimmung von L_1 benötigt wegen der Betragsbildung mehr Rechenzeit als die von L_2. Moderne Mikroprozessoren sind aber mit Befehlserweiterungen ausgestattet, die eine parallele Verarbeitung des Summationsterms aus Gleichung 3.3 durchführen können, siehe Abschnitt 3.6.2.

Im Versuch zeigt sich, daß die L_2-Norm, die eine Grauwertdifferenz quadratisch „bestraft“, zu einem häufigeren Wechsel der Texturquelle in den Referenzdaten führt, siehe auch [52]. In der Kopierkarte äußert sich dieses Verhalten durch kleinere, zusammenhängenden Regionen, die gleichmäßiger verteilt sind. Die L_1-Norm erzeugt ein stärker ausgeprägtes Patchwork - damit aber auch eine visuell bessere Qualität, da grössere, zusammenhängende Bereiche übertragen werden. Abbildung 3.19 vergleicht die Syntheseprodukte von L_1- und L_2-Norm.

Im direkten Vergleich sind sich die Ergebnisse recht ähnlich. Bei der multiskaligen Synthese erzielt L_1 sowohl bei regelmäßigen als auch stochastischen Strukturen geringe Vorteile. Durch die stärker zusammenhängenden Regionen der L_1-Norm sind im Ergebnis

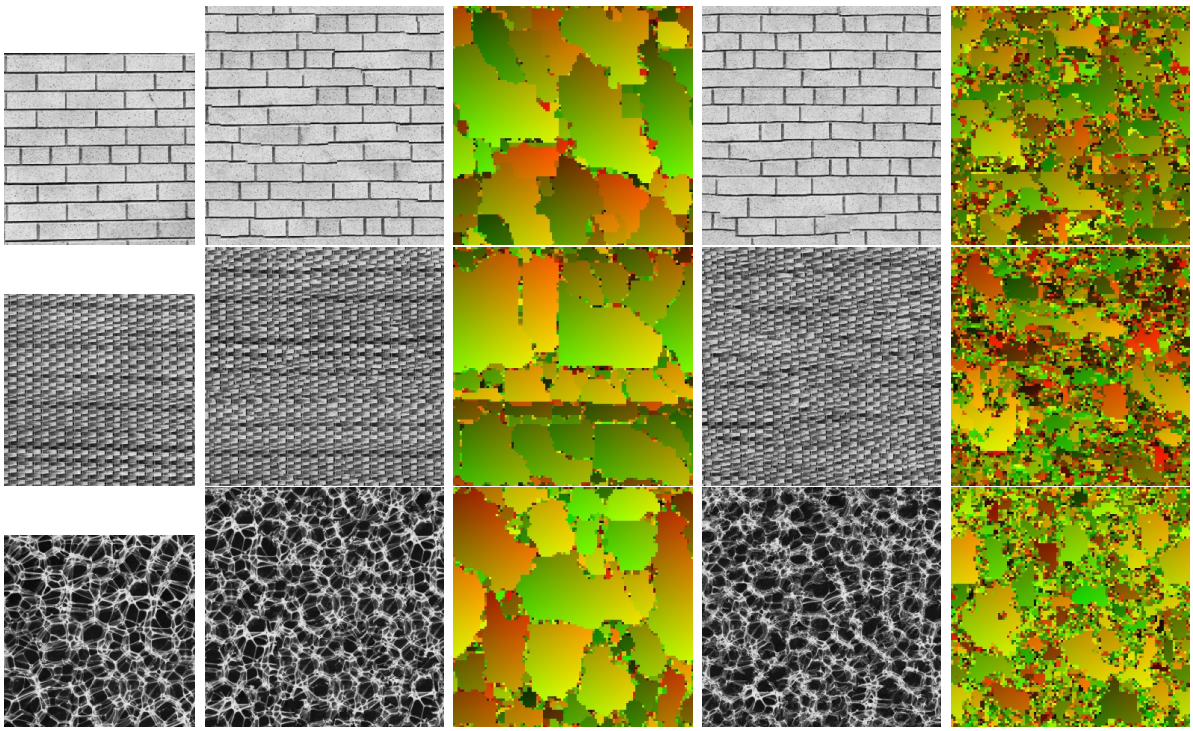


Abbildung 3.19: L_1/L_2-Norm. Referenztexturen (D26, D55, D111; 256×256), Synthese L_1 und Kopierkarte, L_2 und Kopierkarte ($W_N = 7, W_B = 3, L = 4$).

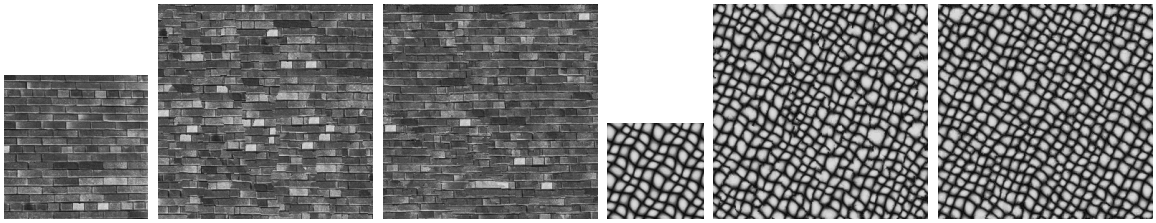


Abbildung 3.20: L_1/L_2-Norm. Referenztextur (D94, 200×200), Synthese (300×300) von L_1 und L_2 ($W_N = 21, W_B = 1, L = 1$), Referenztextur (S01, 128×128), Synthese (300×300) von L_1 und L_2 ($W_N = 7, W_B = 1, L = 2$).

weniger Kantenfragmente vorhanden, was für ein ruhigeres Erscheinungsbild und geringeres Rauschen als bei L_2 sorgt. In den Untersuchungen ließ sich nur für wenige Texturen bei geringer (effektiver) Breite von W_N ein Vorteil für die L_2-Norm herausarbeiten. Zwei Beispiele sind in Abbildung 3.20 gezeigt. Beide Metriken genügen nicht dem Anspruch, ein optimales Maß für visuelle Bildähnlichkeit zu sein. Dennoch ermöglichen sie eine befriedigende Wiedergabe von unterschiedlichen Texturtypen.

3.6 Beschleunigung

In der bisherigen Darstellung des Algorithmus wurde mit Ausnahme der Paritätskontrolle zur Kompensation von Quantisierungsartefakten in Abschnitt 3.3.3 und der Veränderung der Texelgröße in Abschnitt 3.5.2 noch keine Methodik beschrieben, die zu einer Beschleunigung des Verfahrens führt. Die vorgestellte Synthese arbeitet mit implizitem Texturmodell, daher entfällt der Analyseschritt, den andere Verfahren für die explizite Modellbildung ausführen müssen. Die Laufzeit wird mit nahezu 100 Prozent durch den Ähnlichkeitsvergleich von Nachbarschaftsumgebungen bestimmt. Die Ausführungsgeschwindigkeit des Vergleichs ist somit ein Ausgangspunkt für weitere Optimierungen.

Zunächst sollen jedoch alle Faktoren, welche die Rechenzeit beeinflussen, abgeschätzt werden. Durch die Größe der Blocknachbarschaft ($\simeq W_N^2$) wird die Laufzeit des einzelnen Nachbarschaftsvergleichs festgelegt. Die Gesamtzahl der Nachbarschaftsvergleiche hängt linear von den Größen der Referenztextur und der undefinierten Fläche der Ausgabetextur ab. Die Laufzeit des Verfahrens ist also von der Ordnung $O(W_N^2\ I_{ref}\ I_{syn})$. Durch den Einsatz einer Auflösungspyramide konnte W_N im Vergleich zur einfachen Synthese um mehr als eine Größenordnung reduziert und das Verfahren um einen Faktor größer Hundert beschleunigt werden. Da die Größe des Syntheseergebnisses festgelegt ist, kann eine weitere Geschwindigkeitssteigerung nur über die Größe der Referenztextur oder rein technische Verbesserungen in der Implementierung erfolgen. Die technischen Aspekte Vektordarstellung und mehrstufige Parallelisierung werden in den folgenden Abschnitten diskutiert. Anschließend werden Methoden für die Vektorquantisierung und den effizienten Zugriff auf die Referenztextur betrachtet. Als letztes wird eine Analyse des impliziten Texturmodells durchgeführt, welches eine weitere Möglichkeit zur Verfahrensbeschleunigung aufzeigt.

3.6.1 Vektordarstellung

Für die Synthese jedes einzelnen Texturblocks muß ein Vergleich der multiskaligen Blocknachbarschaft mit *allen* Nachbarschaftsumgebungen der Texturreferenz durchgeführt werden. Dieser Vorgang wird ständig wiederholt und erfordert für die Bestimmung des Distanzmaßes eine zeitaufwendige, direkte Adressierung der Pixelzeilen der Referenznachbarschaften im Bild. Die Anordnung der Nachbarschaftsumgebungen im linearen Speicher bedingt außerdem einen fragmentierten Speicherzugriff. Die erste Optimierung für die Berechnung der Distanzmaße ist daher eine Transformation der zweidimensionalen Blocknachbarschaften der Texturreferenz in eindimensionale Vektoren, wie in Abbildung 3.21 dargestellt.

Jede einzelne multiskalige Blocknachbarschaft $N(B_j^{ref})$ wird einmal ermittelt und beide Nachbarschaftsanteile in einem linearen Vektor $\vec{N}(B_j^{ref})$, kurz $\vec{N}_j^{ref}$, gespeichert. Diese bilden die Menge der Nachbarschaftsvektoren $\mathcal{N} = \{\vec{N}_j^{ref}\}$, die in einem zusammenhängenden Speicherblock abgelegt wird. Während des Syntheseprozesses wird eben-

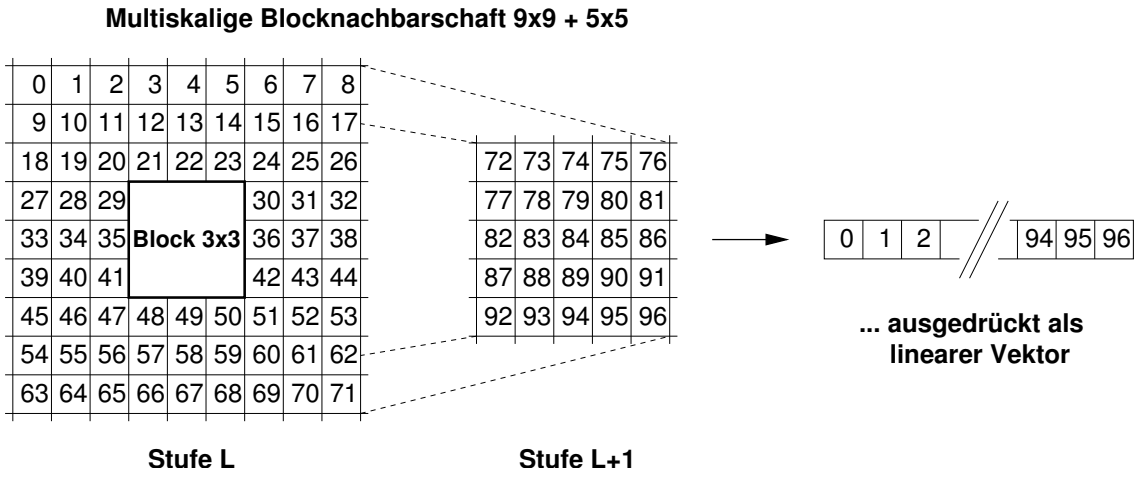


Abbildung 3.21: Multiskalige Blocknachbarschaft als linearer Vektor.

so die aktuelle Synthesenachbarschaft in einen linearen Vektor $\vec{N}^{syn}$ transformiert und dann in der Vektormenge $\mathcal{N}$ nach dem Vektor $\vec{N}_j^{ref}$ mit dem kleinsten Abstand zu $\vec{N}^{syn}$ gesucht. Bei jeder einzelnen Abstandsmessung werden die beiden Anteile der multiskaligen Blocknachbarschaft in dem Referenzvektor als zwei Teilvektoren behandelt, die Abstandsmaße unabhängig voneinander bestimmt und gewichtet addiert, siehe Abschnitte 3.3.2 und 3.5.1. Die variable Maske der Synthese mit Randbedingungen, siehe Abschnitt 3.2.6, wird ebenfalls als linearer Vektor dargestellt und maskiert vor der Berechnung jedes Distanzmaßes die nicht definierten Komponenten. Da die Texturreferenz bei dieser Repräsentationsform nur einmal direkt im Bild ermittelt werden muß, konnte eine Geschwindigkeitssteigerung von ungefähr einer Größenordnung erzielt werden. Nachteil der expliziten Erzeugung von $\mathcal{N}$ ist der hohe Speicherbedarf, der abhängig von der Größe der Nachbarschaftsumgebung und der Referenztextur ist. Für eine typische Referenztextur von 2000×2000 Bildelementen und $W_N = 9, W_N^{erw} = 5$ werden ungefähr 500 Megabyte benötigt!

3.6.2 Parallele Vektorverarbeitung

In Abschnitt 3.5.3 wurde die Anwendung der L_1- und L_2-Metrik als Ähnlichkeitsmaß beschrieben. Der Unterschied der beiden Distanzmaße in der Synthesequalität ist gering bei leichten Vorteilen für die L_1-Norm für bestimmte Referenztexturen. Die Auswahl des Ähnlichkeitsmaßes richtet sich daher nach rein implementativen Gesichtspunkten.

Moderne Prozessoren verfügen neben den üblichen Registern und Befehlssätzen über Erweiterungen, die speziell auf Anwendungen der digitalen Signalverarbeitung zugeschnitten sind, wie z.B. AltiVec™ [74], MMX™ [73] oder auch VIS [94]. Bei diesen Erweiterungen handelt es sich um Parallelarchitekturen mit kurzer Vektorlänge. Sie stellen eine Zwischenstufe zwischen den sequentiell arbeitenden Systemen und den Parallelar-

chitekturen mit langen Vektoren der Supercomputer dar. Die Verarbeitung von langen Vektoren mit Hunderten von Elementen findet Anwendung im wissenschaftlichen Bereich, kurze Vektorlängen werden häufig in Multimedia-Applikationen benötigt.

Die Prozessorerweiterungen ermöglichen es, einen Befehl auf mehrere Datenelemente gleichzeitig auszuführen (SIMD - single instructions, multiple data). Die Parallelarchitekturen mit kurzer Vektorlänge sind 64 oder 128 Bit breit und erlauben die parallele Verarbeitung von 8 bzw. 16 Byte-Werten in einem Taktzyklus. Zu den parallelen Operationen zählt unter anderem die Berechnung der Summe der absoluten Abstände der Vektorkomponenten. Aufgrund dieser hardwareunterstützten Parallelisierungsmöglichkeit wird in der hier entwickelten Textursynthese die L_1-Norm favorisiert.

Auf der verwendeten PC-Hardware kommt die von Intel® eingeführte MMX™-Befehlserweiterung zum Einsatz, die auf allen x86-Prozessoren verfügbar ist. Die MMX™-Erweiterung bietet acht 64Bit-Datenregister und 57 assoziierte Assemblerbefehle, insbesondere die hier verwendete Berechnung der L_1-Norm. Da jede Komponente eines Nachbarschaftsvektors $\vec{N}$ ein grauwertiges Pixel mit einem Byte Größe repräsentiert, kann ein 64Bit-Register acht Vektorkomponenten parallel verarbeiten. Prinzipiell ist die Berechnung der L_1-Norm für die Textursynthese ein Anwendungsfall für Parallelarchitekturen mit langen Vektoren - typische Vektorlängen reichen von 20 bei kleinen Nachbarschaften W_N bis zu 200 oder mehr bei großen Syntheseblöcken W_B und daraus resultierendem großen W_N. Für MMX™ mit kurzer Vektorlänge wird die L_1-Norm der Nachbarschaftsvektoren daher sequentiell in 8 Byte-Blöcken berechnet und die Zwischensummen addiert. Dabei muß die Identität von MMX™- und Fließkomma-Registern berücksichtigt werden, die eine gleichzeitige Nutzung der Parallelisierungsoperationen und Fließkommaberechnungen unmöglich macht. Letzteres ist für das Syntheseverfahren aber nicht kritisch, da z.B. die Gewichtungsoperation in Fließkommadarstellung erst nach der L_1-Norm Berechnung erfolgt.

Durch den Einsatz einer 8fach-parallelen L_1-Norm Berechnung sollte sich auch der Synthesealgorithmus um mindestens Faktor 8 beschleunigen lassen. Der Faktor kann prinzipiell noch größer sein, da die parallele Verarbeitung in drei Taktzyklen erfolgt (Teilvektoren laden, L_1-Norm, Zwischensumme) und sequentiell achtmal fünf Taktzyklen (zwei Vektorkomponenten laden, Differenz, Vorzeichen, Zwischensumme) benötigt werden. Aufgrund der reinen Berechnungszyklen wäre also theoretisch eine Beschleunigung um Faktor 13 denkbar. Tatsächlich wird aber nur ein kleinerer Faktor zwischen 2 und 8 gemessen, der durch die Vektorlänge und die Gesamtsystemleistung beeinflußt wird.

Für beide Berechnungsarten - parallel und sequentiell - bestimmt die Vektorlänge das Verhältnis von Rechen- und Speicherzugriffsoperationen zu Anzahl der Operationen für die Ablaufsteuerung. Die Ablaufsteuerung besteht aus Funktionsaufruf, Schleifen und Zählern für die Vektorkomponenten, Vektorunterteilung bei paralleler Verarbeitung, Gewichtung der Vektoranteile, sowie bedingten Sprungbefehlen bei der sequentiellen Verarbeitung. Für die absolute Laufzeit einer L_1-Norm Berechnung gilt: je kürzer der Vektor, desto höher der Anteil an Steuerungsoperationen und desto geringer der Ge-

winn durch schnellere Ausführung der eigentlichen Berechnung. Abbildung 3.22 zeigt die Abhängigkeit von Vektorlänge und erzieltem Beschleunigungsfaktor bei paralleler Verarbeitung.

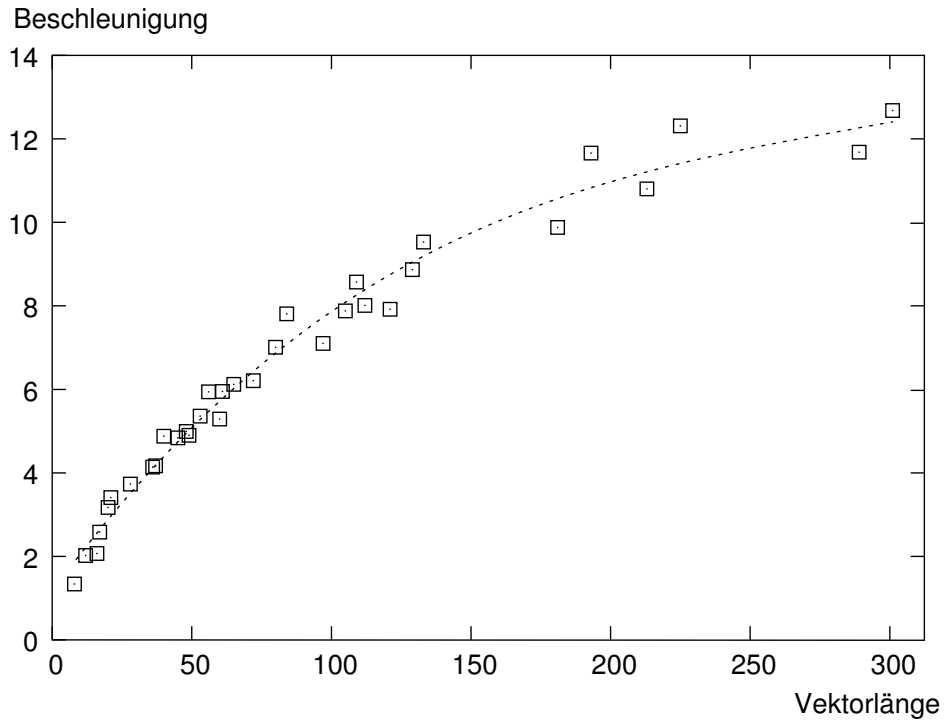


Abbildung 3.22: Beschleunigung der L_1-Norm-Berechnung durch parallele Vektorverarbeitung mit unterschiedlicher Vektorlänge. Messung bei $3GHz$-Systemtakt und kleiner, vollständig gecachter Vektormenge $\mathcal{N}$.

Eine weitere Begrenzung für die Parallelisierung des Synthesealgorithmus ist die effektive Speicherbandbreite, die für die Übertragung von Nachbarschaftsvektoren zur Verfügung steht. Die Rechenleistung kann durch parallele Verarbeitung nur dann signifikant gesteigert werden, wenn dem Rechenwerk kontinuierlich eine ausreichende Datenmenge zur Verarbeitung zur Verfügung steht. Der Systemspeicher gliedert sich bei PC-Hardware in die drei Teilbereiche Firstlevel-Cache, Secondlevel-Cache und Hauptspeicher. Die Cachespeicher arbeiten mit Systemtakt, der Hauptspeicher wird mit eigenem Takt über ein externes Speicherinterface angebunden. Bei einem Rechner mit $3GHz$ Systemtakt und 64Bit Datenbus ergibt sich für die Cachespeicher ein theoretischer Datendurchsatz von $24GB/s$. Da alle Speicher Latenzzeiten beim Zugriff haben und unterschiedlich weit von dem Rechenwerk entfernt sind, werden diese theoretischen Werte in der Praxis nicht erreicht. Beim Lesezugriff auf den kleinen Firstlevel-Cache werden $11GB/s$, auf den größeren Secondlevel-Cache $8GB/s$ gemessen. Die theoretische Bandbreite des z.Zt. schnellsten und hier verwendeten PC-Speicherinterfaces beträgt $6.4GB/s$ ($200MHz$ Systemtakt, 64Bit-Datenbus, 4 Übertragungen pro Signalflanke), wobei in einer optimierten Anwendung $\approx 2.3GB/s$ für Lesezugriff erzielt werden, da auch hier das Initiieren

der Übertragung von Datenblöcken mit Latenzzeiten verbunden ist. Um den Einfluß der effektiven Speicherbandbreite unabhängig von der Vektorlänge zu bestimmen, werden im folgenden nur lange Vektoren mit 100 Komponenten betrachtet.

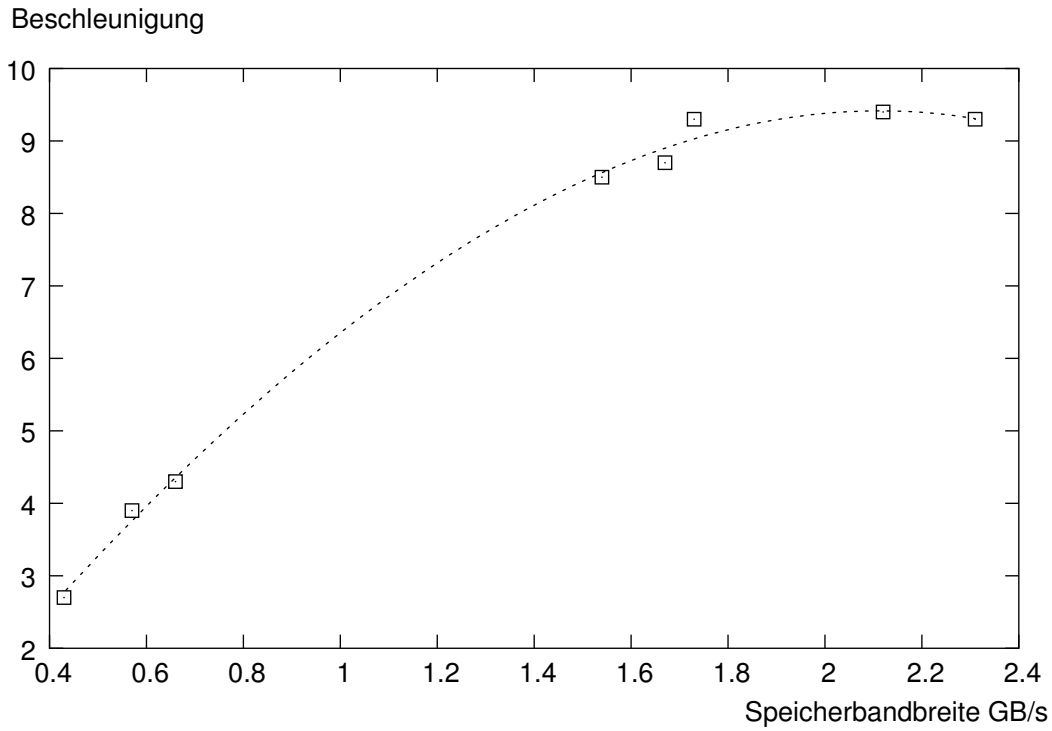


Abbildung 3.23: Beschleunigung der L_1-Norm-Berechnung durch parallele Verarbeitung großer Vektormengen $\mathcal{N}$ bei unterschiedlicher, effektiver Hauptspeicherbandbreite.

Für kleine Vektormengen $\mathcal{N}$, die vollständig im Cache gehalten werden können, liegt der Beschleunigungsfaktor durch parallele L_1-Norm Berechnung mit MMX$^{\text{TM}}$ bei ≈ 8 (siehe auch Abbildung 3.22) und ist unabhängig vom Systemtakt. Bei schnellem Cachespeicherzugriff sind die Datentransferzeiten also kleiner oder gleich den Laufzeiten der Berechnungsoperationen, so daß sich die 8fach-Parallelisierung des Speicherzugriffs und der Berechnung gewinnbringend einsetzen läßt.

Für die Textursynthese typische Vektormengen $\mathcal{N}$ mit mehreren Megabyte Größe können nicht in dem schnellen, Prozessor-assoziierten Cache gehalten werden und müssen über das generelle Speicherinterface aus dem Hauptspeicher geladen werden. Dabei werden nicht einzelne Bytes, sondern parallel 8-, 16- oder 32-Byteblöcke geladen, die dann im schnelleren Cache zur Verfügung stehen. Der Datentransfer vom Hauptspeicher muß prinzipiell so schnell erfolgen, daß zwischen den einzelnen Berechnungen keine zusätzlichen Wartezyklen ausgeführt werden müssen. Diese Randbedingung wird auf Rechnern mit mindestens $1.5GB/s$ effektiver Hauptspeicherbandbreite erfüllt. Eine weitere Erhöhung der Speicherbandbreite, z.B. auf die oben beschriebenen $2.3GB/s$, bewirkt

keine signifikant höhere Beschleunigung der Textursynthese und wird erst in Zukunft, bei noch leistungsfähigeren Prozessoren, notwendig sein. Systeme mit deutlich niedriger Hauptspeicherbandbreite, z.B. $0.7GB/s$, sind nicht in der Lage, die höheren Anforderungen an den Speicherdurchsatz bei paralleler Verarbeitung zu erfüllen - hier wurden Beschleunigungsfaktoren von 3 oder niedriger gemessen, siehe Abbildung 3.23.

3.6.3 Symmetrische Partitionierung von Vektormengen

Aufbauend auf der parallelen Verarbeitung der einzelnen Vektoren wird im nächsten Schritt eine parallele Verarbeitung der Vektormenge $\mathcal{N}$ realisiert. Es ist zulässig, $\mathcal{N}$ in beliebig viele Teilmengen aufzuspalten und den minimalen L_1-Abstand für jede dieser Teilmengen zu berechnen. Nach der parallelen Berechnung der lokalen Abstandminima wird das globale Minimum aus den lokalen Minima ermittelt. Das parallele Berechnen der lokalen Minima der Teilmengen erfolgt auf Rechnerarchitekturen mit mehreren Prozessoren. Diese Systeme werden als SMP-Architekturen (Symmetric Multiprocessing) bezeichnet, bei denen sich alle Prozessoren gleichwertig, gemeinsamen Hauptspeicher teilen. Derartige Systeme skalieren in der Systemleistung annähernd linear bis 8 oder 16 Prozessoren. Darüber hinaus werden die Anforderungen an das Design für einen gleichwertigen Speicherzugriff aller Prozessoren zu hoch, so daß man kleinere Multiprozessoreinheiten mit assoziiertem Speicher über Hochgeschwindigkeitsdatenverbindungen (z.B. Hypertransport [57]) zu noch größeren Clustern gekoppelt werden. Letzteres bezeichnet man als NUMA-Architektur (Non Unified Memory Access).

Die Nachbarschaftsvektormenge $\mathcal{N}$ der Texturreferenz kann für n Prozessoren symmetrisch in n Teilmengen aufgeteilt werden. Parallel laufende Programmroutinen (Threads) bestimmen die lokalen Abstandsmaße in den Teilmengen und liefern das Ergebnis an einen Masterprozeß, der das globale Minimum bestimmt. Der gesamte Programmablauf wird mit Semaphoren synchronisiert. Die Funktionalität wurde auf SMP-Systemen mit 2 und 4 Prozessoren verifiziert. Die effektive Beschleunigung des Verfahrens hängt stark von der Konzeption der Gesamtarchitektur ab. Wie in Abschnitt 3.6.2 beschrieben, ist auch hier die zur Verfügung stehende, effektive Speicherbandbreite der begrenzende Faktor. Da SMP-Architekturen auf *gemeinsamen* Speicher zugreifen, muß der Datentransfer vom Speicher zu den Prozessoren kontinuierlich und so schnell erfolgen, daß die Prozessoren möglichst wenig Wartezyklen ausführen müssen. Die Speicherbandbreite sollte dementsprechend an die Zahl der Prozessoren angepaßt sein. In der Praxis wird diese Anforderung durch parallele Adressierung von mehreren Speicherbänken erreicht, was einen erheblichen Designaufwand bedeutet und letztlich auch den Preis dieser Systeme bestimmt.

Zusätzlich zur Beschleunigung durch parallele Berechnung mit MMX$^{\text{TM}}$ konnte durch Aufspaltung der Vektormenge $\mathcal{N}$ auf einem Zweiprozessorsystem ein Beschleunigungsfaktor 1.8, auf einem Vierprozessorsystem ein Faktor 2.7 erzielt werden. Der Aufwand für das Multithreading und die Kosten der Systeme werden durch diesen Gewinn al-

lerdings nicht gerechtfertigt. Geeigneter ist der parallele Einsatz von preiswerten, aber leistungsfähigen Einzelprozessorsystemen, wie im folgenden Abschnitt diskutiert wird.

3.6.4 Verteilung von Vektorteilmengen

Aus den bisherigen Erkenntnissen zur Parallelverarbeitung von Vektoren bzw. Vektormengen läßt sich ableiten, daß eine weitere Steigerung der Verarbeitungsgeschwindigkeit nur mit hohem Aufwand auf einem einzelnen System realisierbar ist. Der nächste naheliegende Schritt ist daher die Verteilung der Rechenaufgabe auf mehrere, preiswerte Einzelsysteme in einem Netzwerkverbund - eine Clusterlösung. Im Abschnitt 3.6.3 wurde dargestellt, daß sich die Nachbarschaftsvektormenge $\mathcal{N}$ der Texturreferenz in Teilmengen aufspalten läßt, die unabhängig voneinander verarbeitet werden können. Analog dazu können die Teilmengen auch auf mehrere unterschiedliche Rechner (Clienten) übertragen werden, die jeweils die lokalen Minima in den Teilmengen ermitteln. Ein zentraler Rechner (Server), der die gesamte Vektormenge vorhält, dient dabei der Prozesssteuerung und bestimmt das globale Minimum aus den Resultaten der Clienten. Die Aufteilung auf unterschiedliche Rechner ist deshalb möglich, weil die Vektormenge $\mathcal{N}$ über die Laufzeit der Textursynthese konstant bleibt und vor Beginn der Synthese über ein Netzwerk verteilt werden kann. Dieser durch die Netzwerkbandbreite begrenzte, zeitintensive Vorgang findet nur einmal statt - in der weiteren Kommunikation überträgt der Server Vergleichsvektoren an die Clienten, die nach der Berechnung Indizes auf die lokalen Minima zurückliefern. Kritische Punkte sind dabei die Latenzzeit und die Bandbreite für den Datenaustausch zwischen Server und den Clienten. Die Clusterverteilung der Vektormenge beschleunigt den Syntheseprozeß nur dann, wenn die Zeit für die Berechnung eines lokalen Minimums groß gegen die Übertragungszeit für die Netzwerkpakete mit den Vergleichsvektoren und Indizes ist. Weiterhin müssen alle Clienten synchron arbeiten, d.h. die Bestimmung der Minima der Teilmengen muß von allen Clienten in derselben Rechenzeit ausgeführt werden. Diese Bedingung ergibt sich aus der Kausalität des Syntheseprozesses - das globale Minimum resultiert aus den Teilergebnissen und ist Grundlage für die Definition des nächsten (kausalen) Vergleichsvektors.

Die Kommunikation zwischen Server und Clienten wird über das verbindungsorientierte Protokoll TCP/IP mit einer festgelegten Menge unterschiedlicher Steuerungsbefehle durchgeführt. Eine Übersicht gibt Tabelle 3.2. Der Server erhält vom Anwender eine Liste aller Clienten, die in den Cluster integriert werden sollen. Die Synchronisierung der Clienten erfolgt durch dynamische Anpassung der Vektorteilmengen entsprechend der zur Verfügung stehenden Ressourcen. Dazu werden alle Clienten anhand einer Standardvektormenge kalibriert und die vollständige Vektormenge $\mathcal{N}$ der Texturreferenz in ungleich große Teilmengen aufgeteilt. Die Partitionierung richtet sich dabei nach den Ergebnissen der Kalibrierroutine und sorgt dafür, daß alle Teilmengen auf den Clienten in derselben Zeit verarbeitet werden können. Die Bearbeitungszeit wird während der eigentlichen Synthese kontinuierlich gemessen und Abweichungen aufgrund anderweitiger Überlastung einzelner Ressourcen detektiert. Kurzfristige Laständerungen im

Bereich von einigen Sekunden werden ignoriert, länger andauernde Abweichungen durch Rekalibrierung und erneute Vektormengenverteilung kompensiert. Neben diesem Load-Balancing des Clusters werden Totalausfälle von Clienten erkannt und diese aus dem Cluster entfernt.

Befehl	Beschreibung	Antwort
'C'	Aufruf der Client-Kalibrierroutine	Laufzeit in Sekunden
'N'	Übertragung einer Vektorteilmenge	-
'V'	Übertragung eines Vergleichsvektors	Index des ähnlichsten Vektors
'I'	Identifikation	Rechneridentifikation
'Q'	Stopsignal	-

Tabelle 3.2: Verteiltes Rechnen. Steuerungsbefehle für Server-Client Kommunikation.

Der Clusterbetrieb hat sich insbesondere wegen der intelligenten Kalibrierung und Fehlerbehandlung als sehr robust erwiesen und ermöglicht eine weitere Beschleunigung durch Parallelisierung um einen Faktor, der proportional zur Anzahl der Clienten n_c ist und durch den Anteil der Netzwerkkommunikation an der Gesamtlaufzeit beeinflußt wird. Bei Verarbeitung von großen Vektormengen wird eine Beschleunigung von bis zu $0.95 \cdot n_c$ erreicht.

3.6.5 Vektorraumpartitionierung

Die vollständige Suche in der Vektormenge $\mathcal{N}$ der Texturreferenz kann durch die in den letzten Abschnitten vorgestellten Parallelisierungsmaßnahmen MMX™, SMP und Clusterrechnen erheblich beschleunigt werden, stößt aber bei sehr großen Mengen und langen Vektoren immer noch in den Bereich von mehreren Stunden oder Tagen Rechenzeit vor. Gefragt ist eine Alternative zur vollständigen Suche, welche die Synthesegeschwindigkeit signifikant erhöht.

Durch die Vektordarstellung der Nachbarschaftsumgebungen kann der bildhafte Nachbarschaftsvergleich mathematisch gesehen in eine Suche nach dem nächsten Nachbarn in einem hochdimensionalen Vektorraum überführt werden. Die Nachbarschaftsvektoren aus $\mathcal{N}$ sind Punkte in dem Vektorraum $\mathcal{V}$. Die Dimension d_N der Nachbarschaftsvektoren und damit auch des Vektorraums $\mathcal{V}$ wird durch die Größen W_N und W_N^{erw} der multiskaligen Nachbarschaftsumgebung bestimmt. Die Suche nach der ähnlichsten Referenznachbarschaft entspricht der Suche nach einem Vektor $\vec{N}_j^{ref}$, dessen Endpunkt den kleinsten Abstand zum Endpunkt eines Suchvektors $\vec{N}^{syn}$ hat. Der Abstand ist entsprechend der verwendeten Metrik der Euklidische (L_2-Norm) oder der City-Block-Abstand (L_1-Norm). Eine typische Größe für die Dimension d_N der Nachbarschaftsvektoren und damit auch des Vektorraums $\mathcal{V}$ kann der Abbildung 3.21 entnommen werden. Sie beträgt in diesem Beispiel $d_N = 97$. Derartig hohe Dimensionalitäten verschließen sich einfacher Suchverfahren [77] und werden meist mit Vektorraumpartitionierungen behandelt.

Im folgenden wird also neben der Vektormenge $\mathcal{N}$ auch der Vektorraum $\mathcal{V}$ betrachtet, in dem die Vektoren aus $\mathcal{N}$ dargestellt werden. Eine Partitionierung des Vektorraums $\mathcal{V}$ unterteilt diesen in Vektorteilräume, die wiederum Teilmengen von $\mathcal{N}$ beinhalten können. Es muß eine geeignete Methodik zur Partitionierung entwickelt werden, die die Suche nach dem nächsten Nachbarn effizient gestaltet und genau den Vektor aus $\mathcal{N}$ als Suchergebnis liefert, den die vollständige Suche in $\mathcal{N}$ ergeben würde. Letztere Bedingung wird von den meisten Partitionierungsverfahren nicht erfüllt, so daß für die Textursynthese nur suboptimale Zielvektoren identifiziert werden, welche die visuelle Qualität des Ergebnisses negativ beeinflussen können. Im folgenden sollen zwei Klassen geeigneter Verfahren vorgestellt werden: eine geometrische Partitionierung mit einem Hyperkubus und die Vektorquantisierung.

Suche mit Hyperkubus

Die Suche mit Hyperkubus reduziert die Anzahl der Vektorvergleiche durch eine Vorauswahl der Vektoren aus $\mathcal{N}$ [77]. Es werden nur die Vektoren $\vec{N}_j^{ref}$ aus $\mathcal{N}$ betrachtet, die in der Umgebung des Vergleichsvektors $\vec{N}^{syn}$ liegen. Die Umgebung wird durch einen mehrdimensionalen „Würfel“ - den Hyperkubus - mit Kantenlänge ϵ definiert, der um den Vergleichsvektor $\vec{N}^{syn}$ aufgespannt wird. Die Seitenflächen des Hyperkubus werden aus Schnitten von Hyperebenen gebildet, die aus der Selektion der Koordinatenintervalle ϵ für alle Vektorkomponenten entstehen. Abbildung 3.24 zeigt ein zweidimensionales Beispiel. Zuerst werden alle Vektoren ausgewählt (grün/rot), deren nullte Komponente (x) innerhalb der Hyperebenen um den Suchvektor (gelb) liegen, dann wird für die erste Komponente (y) der Vorgang für die Teilmenge aus dem letzten Schritt wiederholt. Für die rot markierten Vektoren wird im letzten Schritt der L_1-Abstand zum Suchvektor berechnet und der Zielvektor mit dem geringsten Abstand ausgewählt.

Das eigentliche Verfahren besteht darin, möglichst schnell die Teilmenge der Vektoren zu berechnen, die im Hyperkubus liegt. Die Vektoren $\vec{N}_j^{ref}$ mit Dimension d_N aus $\mathcal{N}$ bilden die vollständige Kandidatenmenge. Analog zum oben beschriebenen Beispiel werden sukzessiv die Vektoren aus der Kandidatenmenge entfernt, die in der Komponente $d \in \{0, .., d_{N-1}\}$ eine Grauwertdifferenz größer $\epsilon/2$ zur Komponente d des Suchvektors $\vec{N}^{syn}$ aufweisen. In [77] wird dieser Prozeß durch das Erstellen von nach Grauwert sortierten Listen aller Vektorkomponenten der $\vec{N}_j^{ref}$ beschleunigt. Deutlich weniger speicherintensiv und schneller ist die Implementierung einer selbstverkleinernden Vektorindextabelle mit variablen Zeigern [23]. Dabei wird für jedes Vektorelemement ein Integer-Wert gespeichert, der die Sprungweite zum nächsten Kandidatenvektor festlegt. Initial werden diese Indizes alle auf 1 gesetzt. Das Entfernen von Vektoren aus der Kandidatenmenge entspricht nun einer Anpassung der des vorherigen Vektors assoziierten Sprungweite auf den folgenden Vektor. Die wiederholte Anwendung des Löschverfahrens für alle $d \in \{0, .., d_{N-1}\}$ liefert aus der Kandidatenmenge die Vektoren, die innerhalb des Hyperkubus liegen. Diese reduzierte Kandidatenmenge wird auf den minimalen L_1-Abstand zum Suchvektor geprüft und liefert so den Zielvektor.

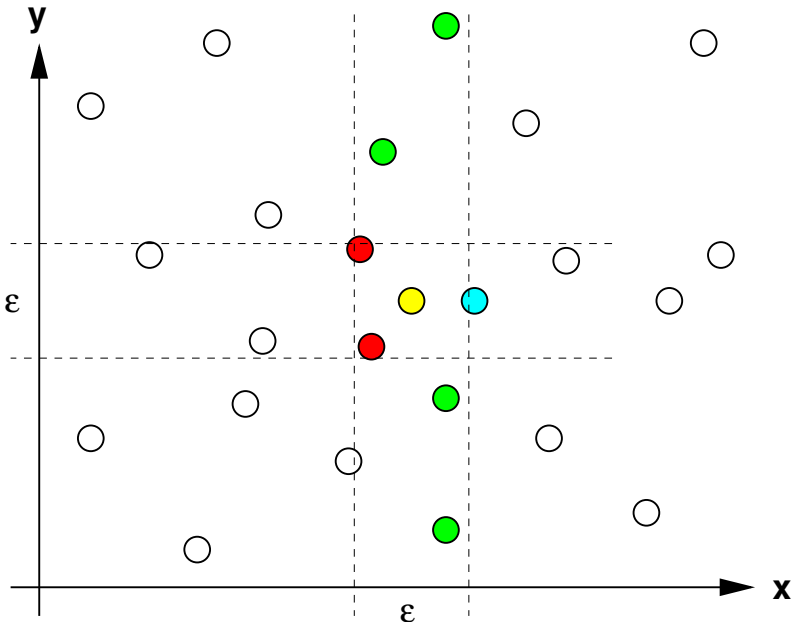


Abbildung 3.24: Suche mit Hyperkubus. Vektorendpunkte (weisse Kreise), Suchvektorendpunkt (gelb), Vektorauswahl nach Prüfung der ersten Komponente (grün/rot), Vektorauswahl nach Prüfung der zweiten Komponente (rot), Vektor mit kürzestem L_1-Abstand (blau).

Die Vektorpartitionierung mit dem Hyperkubus ist fehlerbehaftet, da der Abstand vom Kubuszentrum zur Kubusoberfläche nicht nach L_1-, sondern nach L_∞-Norm konstant ist. Es ist also möglich, daß Vektoren aussortiert werden, die im Sinne der L_1-Metrik näher an dem Suchvektor liegen als der ermittelte Zielvektor (blauer Kreis in Abb. 3.24).

Das Suchverfahren konvergiert schnell, wenn die Hyperkubus Kantenlänge ϵ auf einen kleinen Wert gesetzt und bei jeder Iteration entlang der Vektorkomponenten viele Vektoren der Kandidatenmenge gelöscht werden können. Im Idealfall enthält die Kandidatenmenge nach der vollständigen Aussortierung aller Komponenten exakt einen Zielvektor. Ist ϵ zu groß gewählt, wird auch die finale Kandidatenmenge groß - im Extremfall stellt sich wieder das Problem einer vollständigen Suche. Ist ϵ hingegen zu klein, befindet sich kein Vektor in dem Hyperkubus und die Suche muß zeitaufwendig mit einem größeren ϵ wiederholt werden. Die Auswahl eines geeigneten Wertes für ϵ hängt von der bearbeiteten Textur und dem Zustand der Textursynthese ab. So wird am Anfang einer Vollsynthese ein großes ϵ benötigt, um dem häufigen Wechsel der Texturquelle bedingt durch die Rauschinitialisierung Rechnung zu tragen, siehe Abschnitt 3.4. Im weiteren Verlauf, wenn zusammenhängende Gebiete übertragen werden, kann der Wert für ϵ hingegen kleiner sein. Daher wurde die Suche mit Hyperkubus mit dynamischer ϵ-Anpassung realisiert. Im Laufe der Synthese wird ϵ immer dann verkleinert, wenn ein Grenzwert für die Anzahl von Kandidatenvektoren im Hyperkubus überschritten wird. Bei leerer Kandidatenmenge wird ϵ um 1/3 vergrößert.

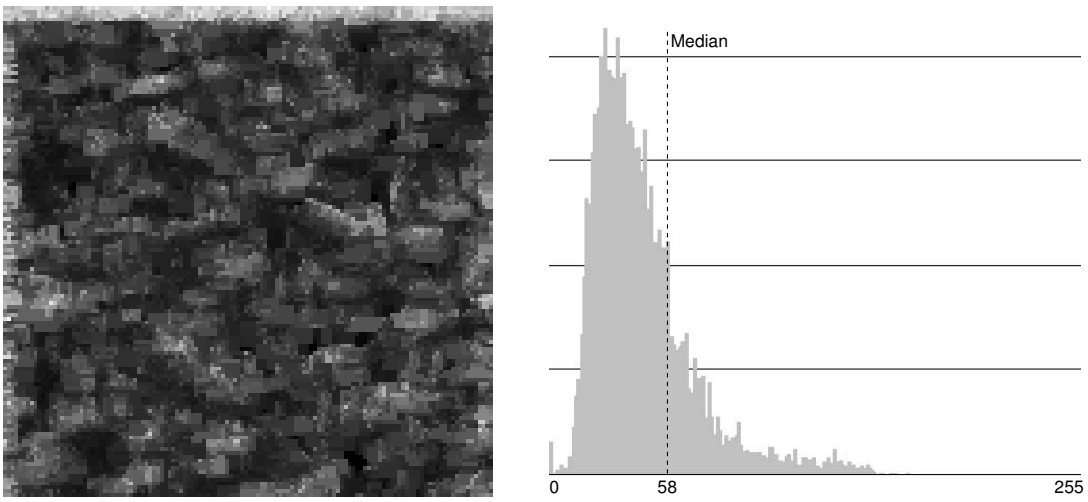


Abbildung 3.25: Breite ϵ des Hyperkubus bei einer Textursynthese. Links eine Karte der ϵ Werte für die Bestimmung der Zielvektoren. Dunkle Punkte entsprechen einem kleinen, helle Punkte einem großem ϵ. Rechts: Histogramm.

Abbildung 3.25 zeigt für eine typische Textursynthese den Wert ϵ, der notwendig war, um einen Zielvektor zu ermitteln. In der Grauwertkodierung bedeuten dunkle Punkte ein kleines, helle Punkte ein großes ϵ. Aus mehreren derartigen Syntheseergebnissen konnte eine durchschnittliche Kantenlänge des Hyperkubus von ≈ 60 abgeleitet werden, um zu einem Suchergebnis zu gelangen, siehe Abbildung 3.25 rechts. Verglichen mit dem absoluten Grauwertbereich von [0..255] ist dieser ϵ-Wert relativ groß, was sich dadurch erklärt, daß ein Textursynthese-Vektorraum üblicherweise eine hohe Dimensionalität aufweist und dünn besetzt, d.h. überwiegend „leer“ ist. Unter anderen Voraussetzungen kann das Verfahren wesentlicher effizienter sein. In [77] kam ein $\epsilon < 3$ zum Einsatz. Trotz des großen Suchbereiches und der damit verbundenen langsamen Aussortierung von Kandidatenvektoren ist die Suche mit Hyperkubus und dynamischer ϵ-Anpassung um einen Faktor 4 schneller als eine vollständige Suche. Unglücklicherweise kann diese Methode aufgrund der Bearbeitung einzelner Vektorkomponenten nicht mit anderen Parallelisierungsansätzen wie MMX$^{\text{TM}}$ kombiniert werden und bietet damit keine Möglichkeit für weitere Beschleunigungsmaßnahmen. Allein der Einsatz von MMX$^{\text{TM}}$ ermöglicht eine Beschleunigung der vollständigen Suche um einen Faktor acht oder höher, so daß die Vektorraumpartitionierung mit Hyperkubus nur optional genutzt und die Auswirkungen auf die Synthesequalität durch fehlerbehaftete Anwendung der L_∞-Norm für die Bestimmung des minimalen L_1-Abstandes nicht weiter untersucht werden.

Vektorquantisierung

Gegeben sei ein Vektorraum $\mathcal{V}^d$ mit Dimensionalität d. Die Vektorquantisierung bildet eine große Trainings- oder Referenzmenge $R = \{\vec{r}_1, ..., \vec{r}_m\} \in \mathcal{V}^d$ auf die kleinere Menge

$\sigma = \{\vec{s}_1, ..., \vec{s}_n\} \in \mathcal{V}^d$, $m > n$ ab. Die Vektoren $\vec{s}_k \in \sigma$ repräsentieren mehrere Vektoren $\vec{r}_j$ der Referenzmenge R und unterteilen den Vektorraum $\mathcal{V}^d$ in n Partitionen, die auch als Cluster bezeichnet werden. Das Ersetzen jedes Vektors $\vec{r}_j$ durch den Repräsentanten seines Teilraums $\vec{s}_k$ wird als Quantisierung mit Codebuch σ bezeichnet. Der durch die Vektorquantisierung auftretende Fehler, dargestellt durch eine Metrik $\delta(\vec{r}_j, \vec{s}_k)$, heißt Verzerrung. Die Aufgabe eines Quantisierungsalgorithmus ist es, ein Codebuch σ zu finden, für das die Verzerrung minimal wird, wobei die Repräsentanten $\vec{s}_k$ im Zentrum der zugehörigen Teilräume liegen. Abbildung 3.26 zeigt ein Beispiel für die Aufteilung des $\mathcal{V}^2$ - ein Voronoi-Diagramm. Die Grenzen der Voronoi-Gebiete ergeben sich aus Teilen von Mittelsenkrechten zur Verbindungsstrecke zwischen jeweils zwei $\vec{s}_k$ (schwarze Punkte). Bei hochdimensionalen Vektorräumen, wie dem $\mathcal{V}$ der Textursynthese, werden aus den Mittelsenkrechten Hyperebenen, die nicht zu veranschaulichen sind, aber analog zum Beispiel im $\mathcal{V}^2$ eine Vektorraumpartitionierung beschreiben.

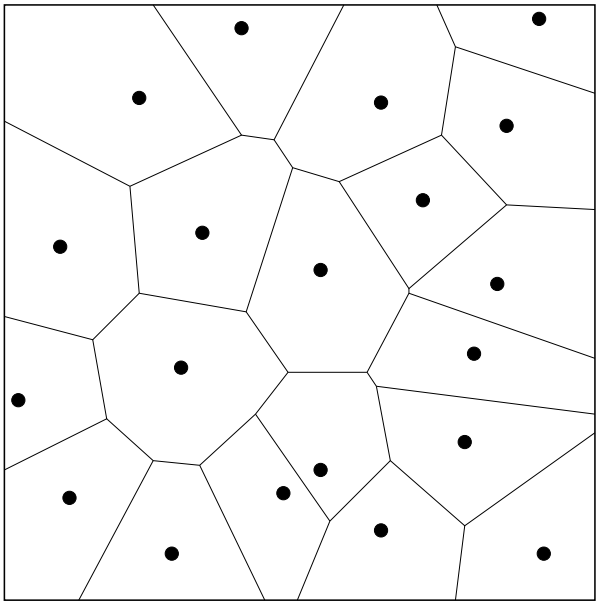


Abbildung 3.26: Partitionierung des $\mathcal{V}^2$. Voronoi-Diagramm mit 20 Clustern.

Zur Beschleunigung der Textursynthese wird die Vektorquantisierung als mehrstufiges Suchverfahren genutzt. Die $\vec{r}_j$ sind hierbei die Nachbarschaftsvektoren $\vec{N}_j^{ref}$ der Referenztextur. Der Vektorraum $\mathcal{V}$ wird partitioniert und ein Vergleich der Teilraumrepräsentanten $\vec{s}_k$ mit dem Suchvektor $\vec{N}^{syn}$ durchgeführt, um die „ähnlichste“ Partition zu finden. Anschließend wird innerhalb der gewählten Partition der optimale Zielvektor aus der Menge der assoziierten Referenzvektoren $\vec{r}_j$ bestimmt. Der Zielvektor ist in jedem Fall ein Element der Referenzmenge R, so daß bei der Synthese kein Quantisierungsfehler durch die Verzerrung $\delta(\vec{r}_j, \vec{s}_k)$ auftritt, da die $\vec{s}_k$ nicht direkt, sondern nur als Verzweigungen für den Suchprozeß genutzt werden. Im Vergleich zur vollständigen Suche entstehen durch die Vektorraumpartitionierung dennoch Fehler, da ein im Sinne der L_1-Norm global optimales Codebuch σ meistens nicht berechnet werden kann.

Um das optimale Ergebnis der vollständigen Suche zu erreichen, muß ein Codebuch σ gefunden werden, das gewährleistet, daß der gewählte partielle Vektorraum den optimalen Zielvektor enthält. Diese Bedingung kann nur dann erfüllt werden, wenn in der Menge der Vektoren bestimmte Eigenschaften wie Vorzugsrichtungen, Cluster, etc. vorhanden sind, die eine geeignete Unterteilung des Vektorraums ermöglichen. Für hochdimensionale, dünn besetzte Vektorräume ist die Bestimmung einer optimalen Partitionierung sehr rechenaufwendig oder sogar unmöglich. Daher liefern fast alle Vektorraumpartitionierungen ein suboptimales Syntheseergebnis. Es ist nun die Aufgabe, ein geeignetes Verfahren zur Erstellung eines Codebuchs σ zu finden, das eine Partitionierung und ein daraus resultierendes Syntheseergebnis ermöglicht, das von dem Ergebnis der vollständigen Suche visuell ununterscheidbar ist.

Die Bestimmung eines optimalen Codebuchs σ für eine große Zahl von Vektoren $R = \{\vec{r}_1, ..., \vec{r}_m\} \in \mathcal{V}^d$ mit großem d ist äußerst rechenintensiv, so daß prinzipiell nur mit Näherungslösungen gearbeitet wird. Aus der Literatur sind unterschiedliche Ansätze bekannt [39]. Der einfachste Ansatz, die Monte-Carlo-Methode, wählt n Vektoren $\vec{s}_k$ zufällig aus der Referenzmenge R. Dieses Verfahren ist sehr schnell ($O(n)$), wobei das Ziel eines optimalen Codebuchs allerdings dem Zufall überlassen bleibt. Ein anderer Ansatz ist das Vektor-Pruning. In der Ausgangssituation ist das Codebuch identisch mit der Referenzmenge ($\sigma = R$). Dann werden sukzessiv die Vektoren $\vec{s}_i$ aus σ entfernt, welche die geringste Gesamtverzerrung $\Delta(\vec{s}_i)$ verursachen, d.h. daß

$$\Delta(\vec{s}_i) = \sum_{j \in \sigma} \delta(\vec{s}_i, \vec{s}_k) \tag{3.4}$$

minimal ist. Dieser Vorgang wird wiederholt, bis das Codebuch die gewünschte Größe hat. Umgekehrt kann auch mit einem leeren Codebuch begonnen werden, dem schrittweise Vektoren hinzugefügt werden, bis eine vorgegebene Gesamtverzerrung unterschritten wird. Dabei bestimmt der Anwender gezielt den zulässigen Verzerrungsgrad und nicht die abstrakte Größe des Codebuchs. Die Laufzeit dieser Verfahren ist von der Größenordnung $O(m^2 n)$.

TSVQ

Eine weitere Methode, die für die Anwendung im Bereich der Textursynthese von Wei [101] vorgeschlagen wurde, ist die n-stufige Vektorquantisierung mit hierarchischer Baumstruktur (tree structured vector quantisation: TSVQ). Bei diesem Verfahren wird der Schwerpunkt der Vektormenge im Vektorraum bestimmt, eine geeignete, binäre Partitionierung um diesen Schwerpunkt berechnet und dann rekursiv dieselben Schritte auf die Vektormengen in den beiden Teilpartitionen ausgeführt. Das Codebuch σ spiegelt dann die durch den rekursiven Prozeß entstehende baumartige Struktur wieder und speichert für jede Verzweigung die beiden Repräsentantenvektoren der Teilpartitionen $\vec{s}_k^{(0,1)}$. Die Blätter des Baumes bestehen aus den (nicht mehr teilbaren) Referenz- bzw. Zielvektoren $\vec{r}_j$. Für die Bestimmung des ähnlichsten Referenzvektors $\vec{r}_j$ zu einem gegebenen

Suchvektor wird letzterer an jeder Verzweigung mit den Codebuchvektoren $\vec{s}_k^{(0,1)}$ verglichen und entsprechend des Abstandsmaßes entlang der Baumstruktur verzweigt, bis ein Blatt mit einem Zielvektor erreicht wird. Wenn $|\mathcal{V}|$ die Anzahl der Vektoren in $\mathcal{V}$ ist, werden die Blätter durch die binären Entscheidungen in $\log_2(|\mathcal{V}|)$ Schritten erreicht, was die Suche deutlich beschleunigt. Bei der Umsetzung dieses Verfahrens zeigt sich jedoch, daß die Partitionierung mit TSVQ nur dann gute Ergebnisse liefert, wenn nicht nur das Blatt mit dem eigentlichen Zielvektor ausgewählt, sondern im Baum rückwärts gelaufen und auch andere Zielvektoren verglichen werden [101], da die hierarchische, binäre TSVQ-Partitionierung an die Verteilung der Nachbarschaftsvektoren im hochdimensionalen, dünn besetzten Vektorraum der Textursynthese nicht optimal angepaßt ist. Die Tiefe des Rückwärtslaufens in der Baumstruktur (Backtracking) reguliert die Qualität des Syntheseergebnisses, wobei das jeweils ausreichende Maß variiert und von der verwendeten Textur abhängt. Da in der Praxis entweder eine hohe Backtrackingtiefe notwendig ist oder weichzeichnende Eigenschaften von TSVQ im Ergebnis sichtbar werden [3] [95], wird in dieser Arbeit das im folgenden beschriebene LBG-Verfahren zur Partitionierung des Vektorraums favorisiert.

LBG-Algorithmus

Wie bereits beschrieben, ist die Suche nach einer optimalen Partitionierung eines hochdimensionalen, dünn besetzten Vektorraums sehr aufwendig. Sie kann aber effizienter gestaltet werden, wenn ein Modell für die zu erwartende Verteilung der Referenzvektoren im Vektorraum zugrunde gelegt wird. Daher stellt sich die Frage nach den speziellen Eigenschaften der in der Textursynthese verwendeten Nachbarschaftsvektoren. Die Nachbarschaftsvektoren repräsentieren die zweidimensionalen, bildhaften Intensitätsverteilungen der Nachbarschaftsumgebungen/-fenster. Diese Verteilungen sind nicht immer einzigartig, wobei der Grad der Variation von der betrachteten Textur abhängt. Im Fall von stark strukturierten, eher regelmäßigen Texturen wie z.B. Mauerwerk, Gitter, Textilien, etc. werden beim Verschieben des Nachbarschaftsfensters immer wieder ähnliche, aber nicht unbedingt identische Grauwertverteilungen beobachtet. Damit führen diese Texturen zu einer Clusterbildung im Vektorraum, wobei die Clusterzentren die unterschiedlichen Elementarzellen der Textur repräsentieren. Bei abnehmenden Grad an Regelmäßigkeit wird das Clusterverhalten schwächer. Im Extremfall einer rein stochastischen Textur (Rauschen) erhält man eine annähernd gleichmäßige Verteilung der Nachbarschaftsvektoren im Raum. Da sich stochastische Texturen auch ohne aufwendige Textursyntheseverfahren erzeugen lassen, werden hier fast ausschließlich Texturen mit einer strukturellen Komponente betrachtet. Daher ist es sinnvoll, eine Vektorraumpartitionierung zu wählen, die gezielt Vektorcluster modelliert.

Der LBG-Algorithmus, benannt nach seinen Erfindern Linde, Buzo und Gray [71], ist eine effiziente Erweiterung der oben beschriebenen Monte-Carlo-Methode und geeignet für die Repräsentation geclusterter Verteilungen. Das Codebuch σ wird zunächst durch zufällig aus der Referenzmenge $R = \{\vec{r}_1, ..., \vec{r}_m\} \in \mathcal{V}^d$ ausgewählte Vektoren gebildet,

wobei die Selektion „ohne Zurücklegen“ erfolgt, d.h. jeder Vektor $\vec{r}_j$ kann nur einmal ausgewählt werden. Die Vektoren dieses initialen Codebuchs $\sigma = \{\vec{s}_1, ..., \vec{s}_n\} \in R, n \ll m$ werden auch als Saatvektoren (seed vectors) bezeichnet. Im zweiten Verfahrensschritt werden die $\vec{r}_j \in R$ jeweils dem Saatvektor zugeordnet, der den geringsten Abstand hat. Dadurch werden n disjunkte Cluster $P_j = \{\vec{r}_1^j, ..., \vec{r}_l^j\}$ gebildet. Im dritten Schritt werden alle Saatvektoren $\vec{s}_k$ in den Schwerpunkt des ihnen zugeordneten Clusters P_j, verschoben, siehe Abbildung 3.27.

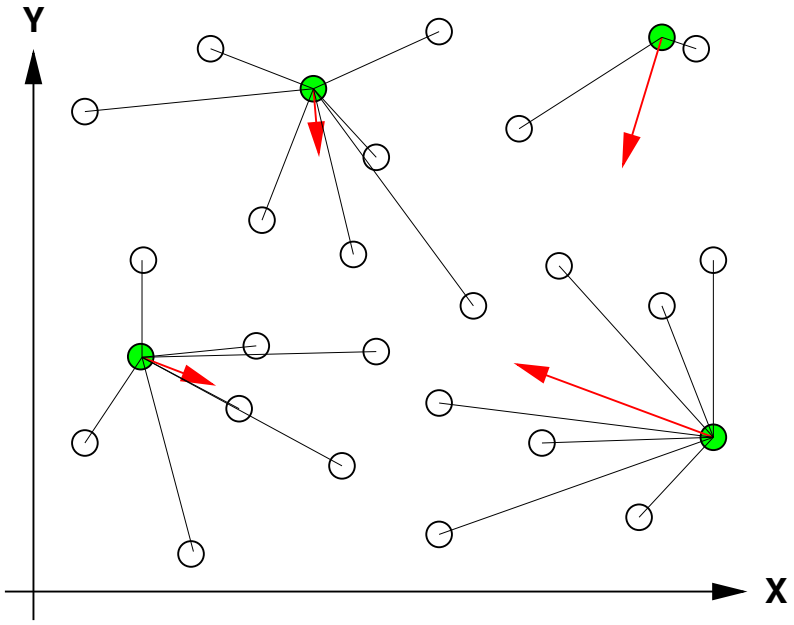


Abbildung 3.27: LBG-Partitionierung: Vektorendpunkte (weisse Kreise), Saatvektorendpunkte (grüne Kreise), iterative Verschiebung (rote Pfeile) der Saatvektorendpunkte in Richtung des jeweiligen Clusterschwerpunkts.

Anschließend werden die Verfahrensschritte zwei und drei so lange wiederholt, bis sich die Positionen der Saatvektoren und damit das Codebuch σ stabilisieren, d.h. die Änderung Δ der Summe der absoluten Verschiebungen der Saatvektoren von zwei aufeinanderfolgenden Iterationen unter einen festgelegten Schwellwert (z.B. $1.02 \equiv 2\%$) fällt. Das LBG-Verfahren ist sehr robust und konvergiert schnell. In bestimmten Situationen ist es dennoch möglich, daß es durch plötzliche Wechsel der Zuordnungen zwischen Referenz- und Saatvektoren zu einer temporären Vergrößerung von Δ kommt. Nach der Fertigstellung des Codebuches sind die Saatvektoren die Schwerpunktsvektoren der Vektorcluster $\vec{s}_k$.

Wie am Anfang dieses Abschnitts beschrieben, wird das mit dem LBG-Verfahren ermittelte Codebuch in der Textursynthese für ein zweistufiges Suchverfahren genutzt, indem zu einem gegebenen Suchvektor zunächst die „ähnlichste“ Partition mittels der Schwerpunktsvektoren $\vec{s}_k$ gesucht und dann innerhalb dieser Partition der optimale Zielvektor $\vec{r}_j$ bestimmt wird, siehe Abbildung 3.28.

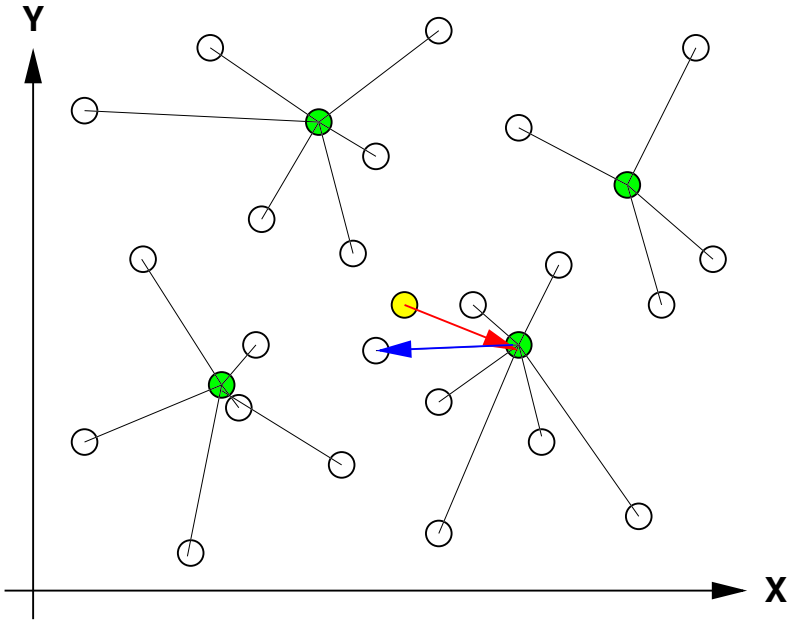


Abbildung 3.28: Vektorauswahl nach Stabilisierung des LBG-Verfahrens. Vektorendpunkte (weisse Kreise), Suchvektorendpunkt (gelber Kreis), Endpunkte der Schwerpunktsvektoren (grüne Kreise), Zuordnung der optimalen Partition (roter Pfeil), Zuordnung des optimalen Zielvektors (blauer Pfeil).

Die Anzahl der Abstandsberechnungen für die Bestimmung des Codebuchs σ hängt von der Zahl der Referenzvektoren $|\mathcal{V}|$, der Partitionen p und der Iterationen it ab. Die Laufzeit des LBG-Verfahrens ist damit von der Größenordnung $O(|\mathcal{V}|\ p\ it)$ und der Einsatz nur dann sinnvoll, wenn das erstellte Codebuch in der Synthese häufig genutzt wird. Für den Fall, daß kleine Texturen aus einer großen Referenzdatenmenge erzeugt werden sollen, ist es unter Umständen besser, eine vollständige Suche in der Vektormenge $\mathcal{N}$ durchzuführen. Die Zeit für die Erstellung des Codebuchs muß also klein gegenüber der Laufzeit der vollständigen Suche sein, wobei auch das LBG-Verfahren selbst durch Einsatz von MMX$^{\text{TM}}$ für die L_1-Abstandsberechnung beschleunigt werden kann. Um die Vektorraumpartitionierung im eigentlichen Suchprozeß optimal nutzen zu können, muß bei der Implementierung der Zugriff auf das Codebuch effizient gestaltet werden. Für das LBG-Verfahren wird jeder Referenzvektor $\vec{r}_j$ mit einem Index auf den zugehörigen Partitionsrepräsentanten $\vec{s}_k$ versehen. Die Indizes werden während der Iterationen bis zur Stabilisierung des Codebuchs ständig angepaßt. Damit werden die Zuordnungen zwar festgelegt, die Vektoren im Speicher aber nicht reorganisiert. Der Zugriff auf die Partitionen während des Suchprozeß ist daher aus speicherphysikalischer Sicht hochgradig fragmentiert, und da die Burst-Qualitäten der Hauptspeicheranbindung (s. oben) nicht genutzt werden können, unter Umständen ebenso aufwendig wie eine vollständige Suche. Daher müssen die Referenzvektoren im Speicher nach Fertigstellung des Codebuchs entsprechend der Partitionierung sequentiell sortiert werden.

Der zweistufige Suchprozeß über die quantisierten Schwerpunktsvektoren des Codebuchs und die zugehörigen Teilmengen der Referenzvektoren führt nicht immer zu dem optimalen Referenzvektor, den die vollständige Suche liefern würde. Durch die suboptimale Vektorauswahl weist das LBG-Resultat im direkten Vergleich der Syntheseergebnisse lokale Grauwertdifferenzen auf. In Abbildung 3.29 sind für vier Synthesebeispiele die Anzahl der Pixel mit Grauwertdifferenz gegenüber der Anzahl der verwendeten Partitionen aufgetragen. Es zeigt sich, daß der Anteil der veränderten Pixel bei höheren Partitionszahlen ein Drittel der Gesamtzahl aller Pixel erreichen kann. Auch der Kurvenverlauf der Summe der absoluten Grauwertdifferenzen entspricht dem der Abbildung 3.29.

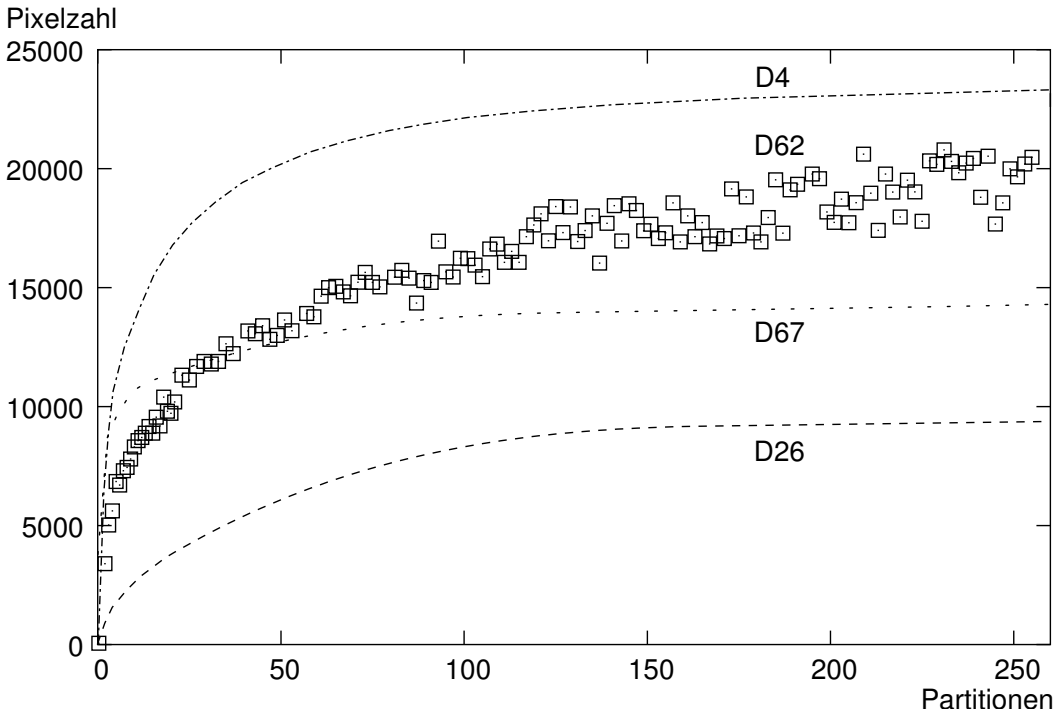


Abbildung 3.29: Fehlzuordnungen durch LBG-gestütztes Suchverfahren. Anzahl der im Vergleich zur vollständigen Suche unterschiedlichen Pixel in Abhängigkeit von der Zahl der LBG-Partitionen. Syntheseprodukte und Referenztexturen $256 \times 256 = 65536$ Pixel (Messwerte D62, approximierte Kurven für D4, D26, D67). Ausgangsbasis ist die Auflösungsstufe $l = 1$, die mit vollständiger Suche berechnet wurde.

In Abbildung 3.30 sind die Syntheseergebnisse der Auflösungsstufe $l = 0$ von vollständiger und LBG-gestützter Suche gegenübergestellt. Beide Resultate wurden ausgehend von derselben Auflösungsstufe $l = 1$ erzeugt. Wie am Anfang beschrieben, eignet sich der LBG-Algorithmus insbesondere dann, wenn die Referenzvektoren Cluster im Vektorraum bilden. Dies gilt speziell für regelmäßige, einfach strukturierte Texturen, wie D26 und D67, rechts in Abbildung 3.30. Für diese Texturen ist der visuelle Eindruck beider

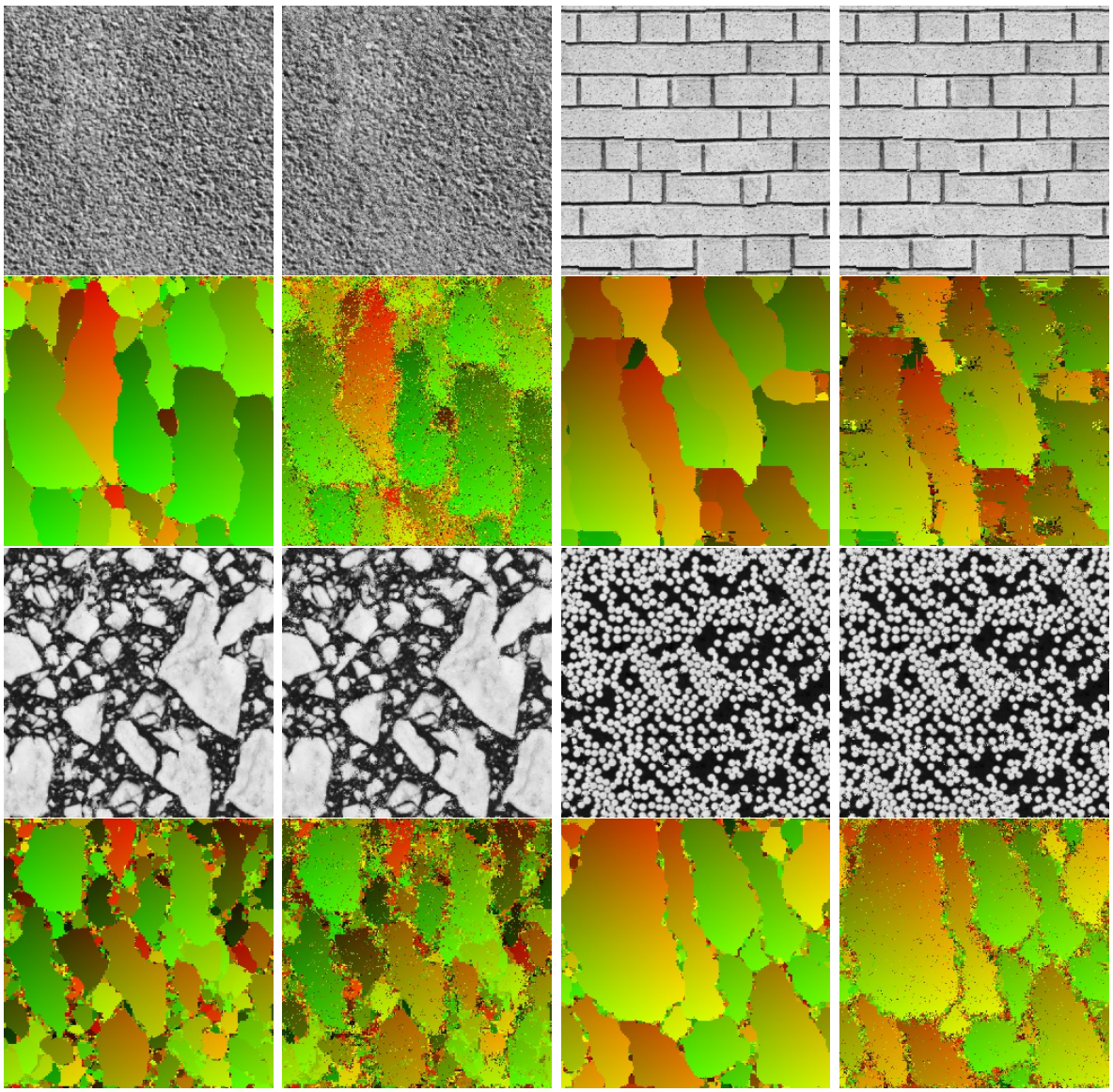


Abbildung 3.30: Visuelle Qualität des LBG-gestützen Suchverfahrens. Jeweils links Syntheseergebnis mit vollständiger, rechts mit LBG-gestützter Suche für Auflösungsstufe $l = 0$ mit 200 LBG-Partitionen, darunter die zugehörigen Kopierkarten. Referenz und Produkt 256×256 Pixel (D4, D26, D62, D67).

Ergebnisse im wesentlichen identisch. Bei D26 sind die Grauwertveränderungen voneinander abhängig, so daß sich in der Kopierkarte kleinere, kompakte Veränderungszentren erkennen lassen. Bei D67 sind die LBG-Suchergebnisse nahezu optimal, d.h. Ergebnis und die zugehörige Kopierkarte weisen wenig Störungen auf. Die Texturen D4 und D62 haben eine stärkere stochastische Komponente. Daher ist der hochdimensionale Vektorraum gleichmäßig besetzt und weist wenig (D62) oder keine Cluster (D4) auf. Die

LBG-gestütze Suche produziert in diesem Fall deutlich mehr Störungen durch suboptimale Suchergebnisse, wie in den Syntheseprodukten und den Kopierkarten zu erkennen ist.

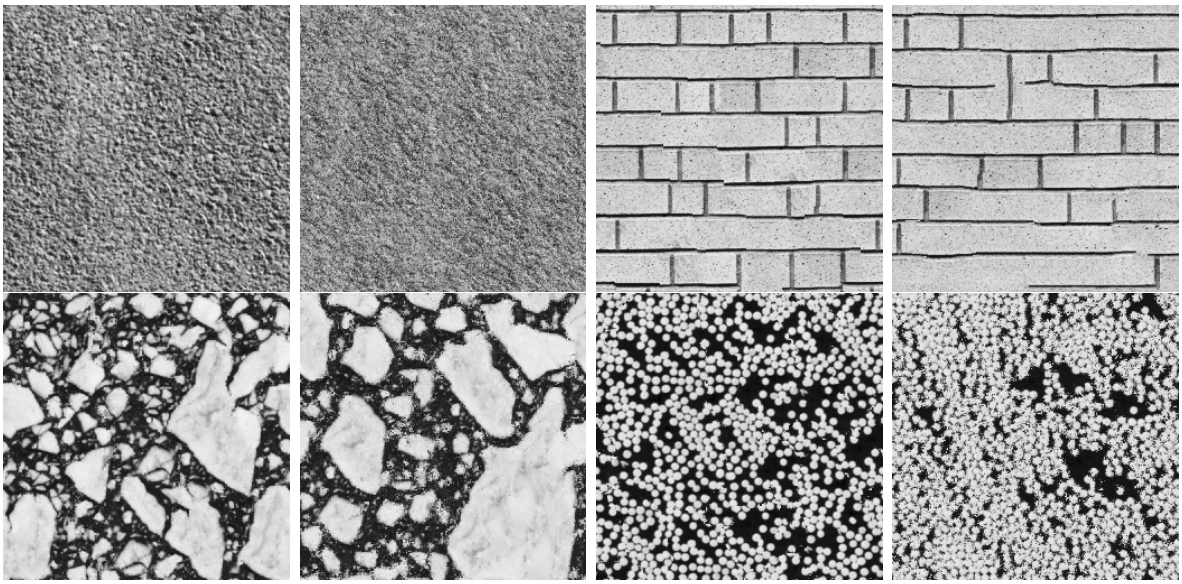


Abbildung 3.31: Visuelle Qualität des LBG-gestützen Suchverfahrens. Jeweils links Syntheseergebnis mit vollständiger, rechts mit LBG-gestützter Suche. Alle Auflösungsstufen mit 200 LBG-Partitionen. Referenz und Produkt 256×256 Pixel (D4, D26, D62, D67).

Abbildung 3.31 stellt Syntheseresultate gegenüber, bei denen für alle Auflösungsstufen L jeweils vollständige und LBG-unterstützte Suche angewendet wurde. Aufgrund der Fehlerfortpflanzung suboptimaler LBG-Suchergebnisse von der niedrigsten zur höchsten Auflösungsstufe erscheinen die LBG-basierenden Resultate insgesamt weicher und störungsbehaftet. Lediglich die Textur D26 wird aufgrund ihrer geclusterten Vektorverteilung hinreichend gut über die Suche mit LBG-Vektorquantisierung reproduziert. Dieses Ergebnis ist ein wichtiger Aspekt bei der Entwicklung der im folgenden Abschnitt beschriebenen, allgemeinen Suchstrategie.

Nach der Fehlerbetrachtung soll nun der Laufzeitgewinn durch das LBG-gestützte Suchverfahren ohne Berücksichtigung der Rechenzeit für das Erstellen des Codebuches ermittelt werden. Für eine Abschätzung der Beschleunigung f_B des Suchprozesses soll von einer homogenen Vektorraumpartitionierung ausgegangen werden, d.h. alle Partitionen p_i haben eine gleich große Besetzung. Die theoretische Obergrenze für f_b hängt dann von der Anzahl der Partitionen p und Vektoren $|\mathcal{V}|$ ab und berechnet sich aufgrund der Zweistufigkeit des Suchverfahrens zu $f_B = |\mathcal{V}|/(p + |\mathcal{V}|/p)$. Aus $f_b = max$ ergibt sich $p = \sqrt{|\mathcal{V}|}$ und $f_B^{max} = \sqrt{|\mathcal{V}|}/2$. In diesem Fall müssen $\sqrt{|\mathcal{V}|}$ Vektoren des Codebuches und anschließend $\sqrt{|\mathcal{V}|}$ Vektoren in der gewählten Partition untersucht werden. Tatsächlich wird dieser Beschleunigungsfaktor in der Praxis aus zwei Gründen nicht er-

reicht. Zum einen ist die Besetzung der LBG-Partitionen aufgrund der Verteilung der Referenzvektoren im hochdimensionalen Vektorraum nicht homogen. In der Praxis zeigt sich, daß die Größe der einzelnen LBG-Partitionen durchaus um einen Faktor 5 oder mehr variieren kann. Zum anderen kann die effektive Nutzung der einzelnen Partitionen stark schwanken, da nicht alle Bereiche der Texturreferenz gleich häufig für das Syntheseprodukt genutzt werden. Der theoretische Laufzeitgewinn muß also um einen Faktor, der sich aus der Nutzungsstatistik zur Laufzeit ergibt, verringert werden. Dieser ist abhängig vom Datenmaterial und liegt z.B. für $p = 100$ zwischen ≈ 15 und 50. In Abbildung 3.32 sind für die bereits untersuchten vier Referenztexturen D4, D26, D62 und D67 die gemessenen Beschleunigungsfaktoren in Abhängigkeit von der Anzahl der Partitionen dargestellt. Das Maximum wird entsprechend der theoretischen Grenze bei 256 Partitionen oder früher erreicht. Für die in Abbildung 3.30 mit $p = 200$ berechneten Texturen wurde mit Ausnahme von D62 die jeweils optimale Beschleunigung erreicht.

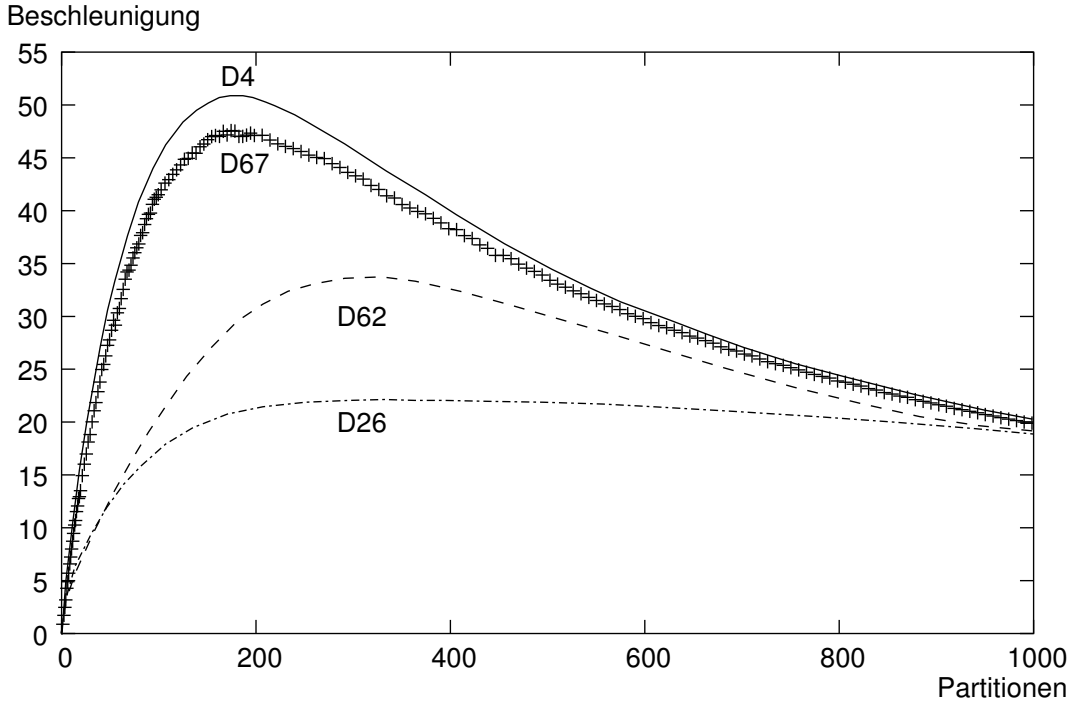


Abbildung 3.32: Beschleunigung durch LBG-gestütztes Suchverfahren. Syntheseprodukte und Referenztexturen 256×256 Pixel (Messwerte D67, approximierte Kurven für D4, D26, D62).

3.6.6 Kopierkarten

Abbildung 3.33 zeigt das Ergebnis einer Synthese mit Randbedingungen, bei der ein einzelner Referenztexturblock flächig erweitert wurde. Dieses und vorherige Bespiele wie Abbildung 3.6 auf Seite 26 demonstrieren die grundlegenden Eigenschaften des Textur-

syntheseverfahrens. Obwohl nur kleine Blöcke von der Referenztextur auf die Syntheseoberfläche übertragen werden, bilden diese wie beim „Texture Patching“ [31] [70] größere zusammenhängende Regionen, was auf den Ähnlichkeitsvergleich der Blocknachbarschaften von Referenz und Synthese zurückzuführen ist, siehe Abschnitte 3.3 und 3.5.1.

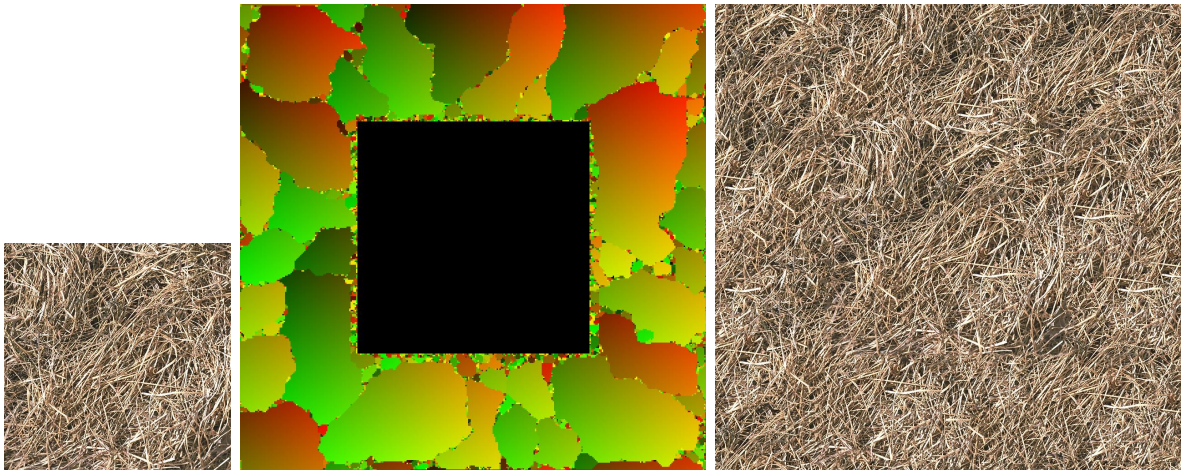


Abbildung 3.33: Grundlegende Syntheseeigenschaften. Das Verfahren kopiert größere zusammenhängende Regionen aus der Texturreferenz.

Die zusammenhängenden Gebiete haben unterschiedliche, zufällige Berandungen und schließen optimal aneinander an. Die Anordnung dieser zufällig berandeten Texturregionen wird schon in der untersten Stufe der Auflösungspyramide festgelegt und bleibt weitgehend konstant. Während der Synthese werden hauptsächlich die Übergangsbereiche verändert bzw. optimiert. Wie viel von der inneren Fläche der Texturregionen über die Auflösungsstufen erhalten bleibt, bzw. wie groß und variabel die Übergangszonen sind, hängt von der vom Benutzer spezifizierten Gewichtung und Größe der erweiterten Nachbarschaftsumgebung ab, siehe Abschnitte 3.3.2 und 3.5.1. Für den Fall der Gleichgewichtung beider Anteile, wie sie standardmäßig eingestellt wird, läßt sich im Experiment ein Flächenverhältnis der veränderlichen Übergangszonen zur Gesamtfläche von 1 zu 2 für niedrige Auflösungsstufen ermitteln.

Dies ist der Ansatzpunkt für eine Beschleunigungsmethode, die im Gegensatz zu den bisher beschriebenen Verfahren nicht auf einer Optimierung des Suchprozesses basiert, sondern das charakteristische Verhalten des Verfahrens bei der Oberflächengenerierung ausnutzt, also modellspezifisch ist. Die zusammenhängenden Regionen prägen sich in den Kopierkarten durch homogene, zweidimensionale Gradienten mit einer Steigung von 1 aus und können mit einem $n \times n$-Filterkern derselben Charakteristik detektiert werden. Die Breite der Pufferzonen, welche die Übergänge zwischen benachbarten Teilregionen maskieren, wird gleichzeitig über die Größe des Filterkerns festgelegt. Die Kopierkarte wird damit in eine Kopiermaske für die folgende Auflösungsstufe der Pyramide überführt. Der aufwendige Syntheseprozeß wird anschließend nur in den Übergangszonen ausgeführt. Die gleichbleibenden Regionen werden direkt aus der Referenztextur ko-

piert, wobei die Koordinaten aus der Kopiermaske der vorherigen Auflösungsstufe extrapoliert werden. Bei vergleichsweise geringem zusätzlichen Speicherbedarf durch die zusätzliche Kopiermaske kann das Verfahren signifikant beschleunigt werden. Da das Flächenverhältnis von Übergangszonen zu Gesamtfläche für höhere Auflösungsstufen immer kleiner wird, werden in der Praxis Beschleunigungsfaktoren von 2 bei niedrigen bis zu 4 bei hohen Auflösungsstufen erreicht.

Bei Anwendung der Kopiermaske muß grundsätzlich der Einfluß auf das Syntheseergebnis berücksichtigt werden. Wie oben beschrieben, ist die Kopiermaske nur dann gültig, wenn die gewählten Syntheseparameter tatsächlich zu konstanten Texturregionen führen. Durch das Fixieren homogener Bereiche mit der Kopiermaske wird letztlich ein Eingriff in das implizite Texturmodell des Verfahrens vorgenommen. Für stark variable Regionen, wie rechts in Abbildung 3.16 auf Seite 37, beeinträchtigt die Festlegung bestimmter Flächenanteile unter Umständen die visuelle Qualität des Syntheseprodukts. Daher ist vor dem Einsatz einer Kopiermaske die Flächenkonstanz in mehreren aufeinanderfolgenden, niedrigen Auflösungsstufen zu validieren.

3.6.7 Strategie

Alle im Abschnitt 3.6 beschriebenen Methoden zur Beschleunigung des Nachbarschaftsvergleichs von Texturreferenz und Textursyntheseblöcken kommen auf jeder Auflösungsstufe der Bildpyramide zum Einsatz. Die Verfahren können einzeln oder in Kombination angewendet werden und lassen sich in drei unterschiedliche Klassen aufteilen:

- Verlustfrei
- Modellspezifisch
- Verlustbehaftet.

Zu den verlustfreien Methoden zählen ganz allgemein die Vektordarstellung der Nachbarschaftsumgebungen sowie die parallele Verarbeitung der Vektoren über MMX™ SMP und verteiltes Rechnen. Das modellspezifische Beschleunigungsverfahren über die Kopierkarten ist dann verlustfrei, wenn durch geeignete Parameterwahl keine Modellverletzung auftritt. Jede Art von Vektorraumpartitionierung über Vektorquantisierung ist dagegen verlustbehaftet, da suboptimale Ergebnisse erzielt werden und sich Fehler in der Auflösungspyramide fortpflanzen und verstärken.

Für die Entwicklung einer Strategie zur Anwendung der verschiedenen Verfahren innerhalb der Textursynthese müssen zunächst Randbedingungen definiert werden. Prinzipiell erhöht sich der Rechenaufwand von Stufe zu Stufe der Auflösungspyramide um einen Faktor 16, da sich sowohl die Fläche der Referenztextur als auch die der Syntheseoberfläche jeweils um einen Faktor 4 vergrößert. Allein die Synthesezeit für die höchste Auflösungsstufe $l = 0$ definiert also die Laufzeit des Syntheseprozesses. Dies bedeutet im

Umkehrschluß, daß die Synthese einer niedrigen Auflösungsstufe im allgemeinen recht schnell berechnet werden kann. Daher wird für die unteren Auflösungsstufen der Bildpyramide die verlustfreie, vollständige Suche mit MMX^{TM} eingesetzt. Für die hohen Auflösungsstufen $l = 0, 1$ wird normalerweise die Vektorraumpartitionierung mit LBG-Vektorquantisierung in Verbindung mit den Kopierkarten gewählt. Für mittlere Stufen und bei Texturen, die sich nicht optimal mit der LBG-gestützen Suche umsetzen lassen, kann optional das verteilte Rechnen aktiviert werden. Der Benutzer entscheidet aufgrund der Zwischenergebnisse und Laufzeitabschätzungen, die das Syntheseverfahren erstellt, welche der unterschiedlichen Methoden auf den einzelnen Pyramidenstufen eingesetzt werden.

Im Vergleich mit anderen aktuellen Syntheseverfahren wird mit dieser Beschleunigungsstrategie bei höherer Qualität eine vergleichbare oder bessere Texturierungsgeschwindigkeit erreicht [95]. In Tabelle 3.3 sind die Rechenzeiten der unterschiedlichen Beschleunigungsverfahren für die Beispiele aus den Abbildungen 4.1 und 4.2 dargestellt. Die Beispiele selbst sind mit der schnellsten Kombination in der rechten Tabellenspalte erstellt worden.

l	Basis	+Parität	+MMX	+Kopierkarte	+Cluster	+LBG	opt.
4	0.05	0.05	0.02	0.02	0.02	-	0.02
3	2.01	0.51	0.12	0.12	0.74	-	0.12
2	37.51	9.40	2.04	2.04	3.45	-	2.04
1	642.26	161.44	35.03	22.94	13.26	-	13.26
0	10914.00	2740.46	569.13	259.26	89.94	4.86	4.86
$\sum$	11595.83	2911.86	606.34	284.38	107.41	-	20.30
Faktor	1	4	19	41	108	-	571

Tabelle 3.3: Laufzeit Textursynthese. Eingangsdaten 512×512 Punkte, Ausgangsdaten 1024×1024 Punkte, Parameter: $L = 5$, $W_N = 7$, $W_B = 3$, $2.53GHz$ CPU, Cluster mit 4 CPUs, Angaben in Sekunden $[s]$.

3.7 Optimierung der Flächennutzung

Ein generelles Problem der auf einem impliziten MRF-Ansatz basierenden Textursyntheseverfahren ist die Eigenschaft, bestimmte Referenztexturregionen wiederholt an sich selbst anzuschließen [32] [31]. Ursache dafür ist das dem Algorithmus zugrundeliegende Prinzip, daß die Auswahl von Referenztexturblöcken für den Syntheseprozeß aufgrund der Ähnlichkeit der Nachbarschaftsumgebungen von Referenzblöcken und Syntheseposition erfolgt. Wie in den vorherigen Abschnitten beschrieben, wird dafür in der Menge der Nachbarschaftsvektoren $\mathcal{N}$ nach dem Referenzvektor gesucht, dessen Abstand zum gegebenen Suchvektor am geringsten ist. Dabei kann es vorkommen, daß die Synthese

ausschließlich eine bestimmte Teilmenge von $\mathcal{N}$ nutzt, wodurch ständig dasselbe Muster kopiert wird. Betrachtet man die Verteilung der Vektoren aus $\mathcal{N}$ im Vektorraum $\mathcal{V}$, zeigt sich, daß insbesondere bei inhomogenen Referenztexturen bzw. inhomogener Besetzung von $\mathcal{V}$ die Abstände zwischen verschiedenen Referenzvektorclustern sehr groß sein können, so daß bei der Bewertung in Form der L_1- oder L_2-Norm eine in sich geschlossene Referenzregion für den weiteren Syntheseverlauf bevorzugt wird. Ein weiterer Nachteil der Selbstwiederholung ist, daß nur ein kleiner Teil der Referenztextur für das Syntheseprodukt genutzt wird, gleichzeitig aber die gesamte Referenzvektormenge $\mathcal{N}$ für die Synthese jedes einzelnen Texturblocks durchsucht werden muß, wenn keine LBG-gestütze Suche durchgeführt wird. Der im folgenden beschriebene Lösungsansatz für dieses grundlegende Problem ermöglicht eine deutlich höhere Synthesequalität und die Behandlung von inhomogenen Texturen.

Zur Laufzeit der Synthese kann die Tendenz zur Selbstwiederholung durch Auswertung der Nutzungsstatistik detektiert werden. In Abschnitt 3.1 auf Seite 18 wurde die Flächennutzungskarte des Syntheseverfahrens vorgestellt. Diese Karte besteht aus 32Bit-Zählern, die mit den Bildelementen der Referenztextur korrespondieren. Im Verlauf der Synthese werden für jeden übertragenen Texturblock B_j^{ref} die Zähler der assozierten Pixel um 1 inkrementiert. Nach der Fertigstellung einer Auflösungsstufe l gibt die (normierte) Flächennutzungskarte Aufschluß über die Häufigkeit, mit der die einzelnen Bereiche der Referenztextur für das Syntheseergebnis verarbeitet wurden.

Die Betrachtung der Nutzungsstatistik ist zunächst einmal eine wertvolle Information bei der Optimierung der Verfahrensparameter. Die Größe der Blocknachbarschaft W_N und die Pyramidentiefe L beeinflussen den Nutzungsgrad der Referenztextur, da in den niedrigen Auflösungsstufen die globale Anordnung von Texturregionen und damit die prinzipielle Verwendung der Referenztextur für den Syntheseprozeß festgelegt wird.

3.7.1 Passive Varianzoptimierung

Eine weitere interessante Anwendung der Flächennutzungskarte wird durch die Vollsynthese 3.4 gegeben. Da diese von einer Zufallsverteilung ausgeht, ergeben sich bei verschiedenen Läufen des Verfahrens mit derselben Parametereinstellung unterschiedliche, aber ähnliche Ausgangstexturen. Der visuelle Unterschied zweier unterschiedlich initialisierter Syntheseergebnisse läßt sich durch einen menschlichen Betrachter bewerten, der der einen oder anderen Ausgangstextur aufgrund von Struktur- oder Häufungsmerkmalen den Vorzug gibt. Alternativ kann für Bewertungsfragen ein objektives Maß wie z.B. die Gleichmäßigkeit der Flächennutzung über die Varianz berechnet werden. Ein niedrigerer Varianzwert bedeutet dabei eine homogenere, gleichmäßigere Flächenausnutzung als ein höherer Wert. Das Verfahren bietet einen passiven Varianzoptimierungsmodus, bei dem für jeden Syntheselauf der aktuelle Varianzwert der Flächennutzungskarte ermittelt und mit dem von vorherigen Programmläufen verglichen wird. Nach einer vom Benutzer definierten Anzahl von Iterationen oder maximalen Programmlaufzeit wird der Optimie-

rungsprozeß beendet und das Syntheseergebnis mit der kleinsten Varianz ausgegeben. Die passive Varianzoptimierung ist insbesondere für schnell zu berechnende, niedrige Auflösungsstufen sinnvoll, wenn das Ziel einer Synthese eine möglichst homogene Nutzung aller globalen Referenztexturmerkmale ist.

3.7.2 Aktive Varianzoptimierung

Neben der passiven Varianzminimierung über Ausnutzung des Zufallsprozesses bei der Vollsynthese ermöglicht die Flächennutzungskarte auch eine aktive Varianzoptimierung durch gezielten Eingriff in den Syntheseprozeß. Dazu wird zur Laufzeit der Synthese neben der lokalen Nutzungsstatistik für die einzelnen Pixel bzw. Blöcke der Referenztextur $U(B_j^{ref})$ die globale, mittlere Flächennutzung $\overline{U}$ berechnet. Das Verhältnis von $\overline{U}$ und $U(B_j^{ref})$ kann dann als zusätzlicher Bewertungsterm bei der Berechnung des Abstandsmaßes zweier Nachbarschaftsvektoren eingesetzt werden:

$$L(\vec{N}(B^{syn}), \vec{N}(B_j^{ref})) = \sum_{i=0}^{l} |P_i^{syn} - P_i^{ref}| + f(U(B_j^{ref}), \overline{U}). \tag{3.5}$$

Der Einfluß des Bewertungsterms wird über die Funktion f definiert, die z.B. als Multiplikator oder Exponentialfunktion von $U(B_j^{ref})/\overline{U}$ implementiert wird. Durch die Erweiterung des Bewertungsmaßes werden die Vektoren „bestraft“, deren assozierte Texturblöcke B_j^{ref} häufiger als andere während der Synthese genutzt werden. Daraus folgt, daß nicht immer der im Sinne der L_1-Metrik optimal passende Vektor, sondern gegebenenfalls ein anderer Vektor gewählt wird, der einen größeren L_1-Abstand aufweist, aber im bisherigen Syntheseverlauf seltener genutzt wurde.

Der zusätzliche Bewertungsterm führt zu einer homogeneren Nutzung der vorhandenen Referenztextur, die sich in einer niedrigeren Varianz der Flächennutzungskarte wiederspiegelt. Gleichzeitig kann die Qualität der Übergänge zwischen benachbarten, zusammenhängenden Regionen schlechter werden, da durch das Kriterium f nicht mehr ausschließlich nach dem optimalen Anschluß gesucht wird. Der visuelle Gesamteindruck wird durch die Varianzoptimierung aber grundsätzlich verbessert, da die Selbstwiederholung effektiv begrenzt wird. Abbildung 3.34 zeigt einen Vergleich zwischen verbesserter und herkömmlicher Flächennutzung. Die obere Zeile wurde ohne Auswertung der Flächennutzungsstatistik im Laufe des Syntheseprozesses erstellt und zeigt selbstwiederholende Elemente, die sich auch in der ungleichmäßigen Verwertung der Referenztextur ausprägen (siehe Flächennutzungskarte, links). Für die Ergebnisse der unteren Reihe wurde die Bewertung der Flächennutzung bei der Berechnung des Abstandmaßes berücksichtigt, was sich in der homogeneren Flächennutzungskarte wiederspiegelt und zu deutlich weniger Artefakten im Syntheseprodukt führt.

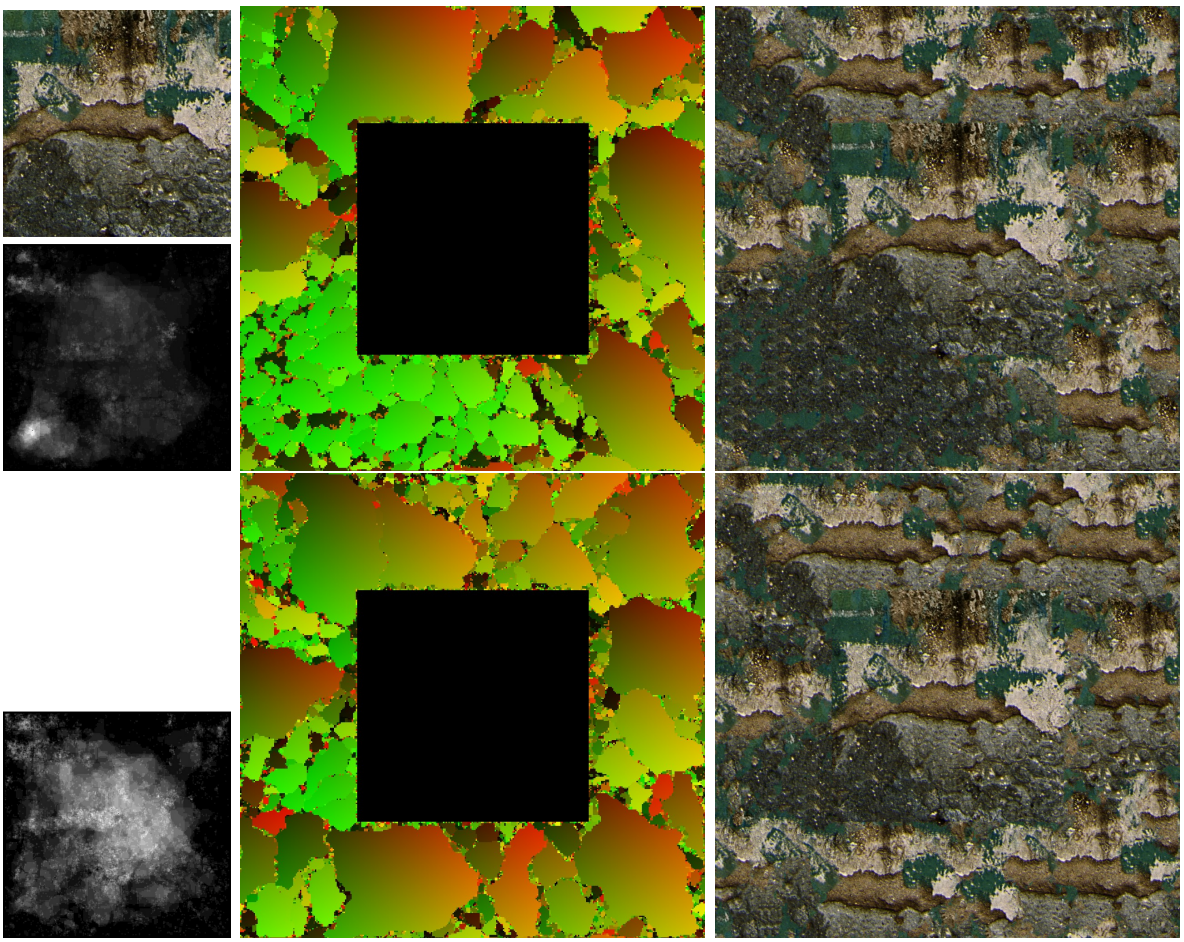


Abbildung 3.34: Aktive Optimierung der Flächennutzung. Oben links: Referenztextur, jeweils von links nach rechts: Flächennutzungskarte, Kopierkarte und Syntheseprodukt. Obere Reihe ohne, untere mit Varianzoptimierung.

3.8 Farbverarbeitung

Das bisher vorgestellte Verfahren zur Textursynthese ist, wie in Abschnitt 3.1 beschrieben, ausschließlich für die Synthese von grauwertigen Eingangstexturen geeignet. Für das Einbringen von Farbinformation in den Syntheseprozeß sind zwei unterschiedliche Ansätze denkbar. Die zusätzliche Information kann in jedem Verfahrensschritt direkt, z.B. als 3-Band Rot-Grün-Blau-Textur (RGB), verarbeitet werden [101]. Dabei wird die zeitintensive Berechnung der L_1- oder L_2-Norm auf Basis des Bildpunkttripels ausgeführt. Diese Vorgehensweise hat den Nachteil, daß sich Rechenzeit und Speicherbedarf für die Nachbarschaftsvektormenge verdreifachen und durch die dreifach höhere Dimensionalität der Referenzvektoren $\vec{N}_j^{ref}$ die Auswahlmöglichkeiten für die Referenztexturquelle signifikant reduziert werden (*curse of dimensionality*). Außerdem ist die Abstandsmessung in einem RGB-Farbraum für die Ermittlung der Bildähnlichkeit ungeeignet [2].

Eine weniger aufwendige Alternative ist die an Verfahren der Videokodierung angelehnte, getrennte Verarbeitung von Luminanz- und Chrominanzinformation [55]. Dazu wird zunächst der Luminanzkanal einer Farbeingangstextur extrahiert bzw. ein RGB-Datensatz in eine grauwertige Textur konvertiert und anschließend eine Synthese in der bisher beschriebenen Form durchgeführt. Nach Fertigstellung der höchsten Auflösungsstufe $l = 0$ der Bildpyramide kann die Kopierkarte dazu genutzt werden, Texturblöcke aus der farbigen Eingangstextur analog zur Anordnung im grauwertigen Syntheseergebnis zu einem farbigen Syntheseergebnis zusammenzustellen. Voraussetzung für die getrennte Verarbeitung von Farb- und Strukturinformation ist eine starke Korrelation beider Komponenten, so daß bei einer Synthese der Luminanzinformation eine Annahme über die zugehörige Farbinformation gemacht werden kann. Diese Korrelation ist nicht bei allen Texturen gegeben. Mit der VisTex Bibliothek [81] wurden zahlreiche Synthesetests durchgeführt, die gezeigt haben, daß die Mehrzahl der dort katalogisierten Texturen die Korrelationsbedingung insoweit erfüllen, daß in den Syntheseprodukten keine auffälligen Farbartefakte erkennbar waren. Repräsentative Beispiele sind in den Abbildungen 3.33 und 3.34 sowie dem Ergebnisteil 4 dargestellt.

Abbildung 3.35 zeigt eine Textur, die identische Strukturen mit unterschiedlicher Farbinformation aufweist und somit die gestellte Randbedingung nicht erfüllt. Da die Farbinformation bei der Ähnlichkeitsmessung nicht ausgewertet wird, kommt es zu Artefakten an den Übergängen zwischen zusammenhängenden Gebieten. Das grauwertige Syntheseergebnis ist hingegen „fehlerfrei“. Trotz der schwachen Korrelation von Farbe und Struktur ist das Syntheseergebnis insgesamt noch durchaus zufriedenstellend.

Abbildung 3.35: Farbverarbeitung. Von links nach rechts: die Vollsynthese liefert einen guten Gesamteindruck, im Detail werden Artefakte an den Rändern zusammenhängender Gebiete sichtbar, die durch die grauwertige Synthese nicht behandelt werden.

4 Experimentelle Ergebnisse

In Kapitel 3 wurden alle wesentlichen Aspekte des entwickelten Textursynthese-Algorithmus, wie Ansatz, Implementierung und Parameteroptimierung behandelt. Im folgenden werden verschiedene Syntheseergebnisse gezeigt und diskutiert, die aus der optimierten Anwendung des beschriebenen Verfahrens resultieren. Ein direkter Vergleich mit anderen Verfahren hinsichtlich Synthesequalität und -geschwindigkeit wurde in [95] durchgeführt.

4.1 Vollsynthese

Die in Abbildung 4.1 dargestellten Syntheseprodukte wurden über eine Vollsynthese mit Rauschinitialisierung 3.4 und Optimierung der Flächennutzung erstellt.

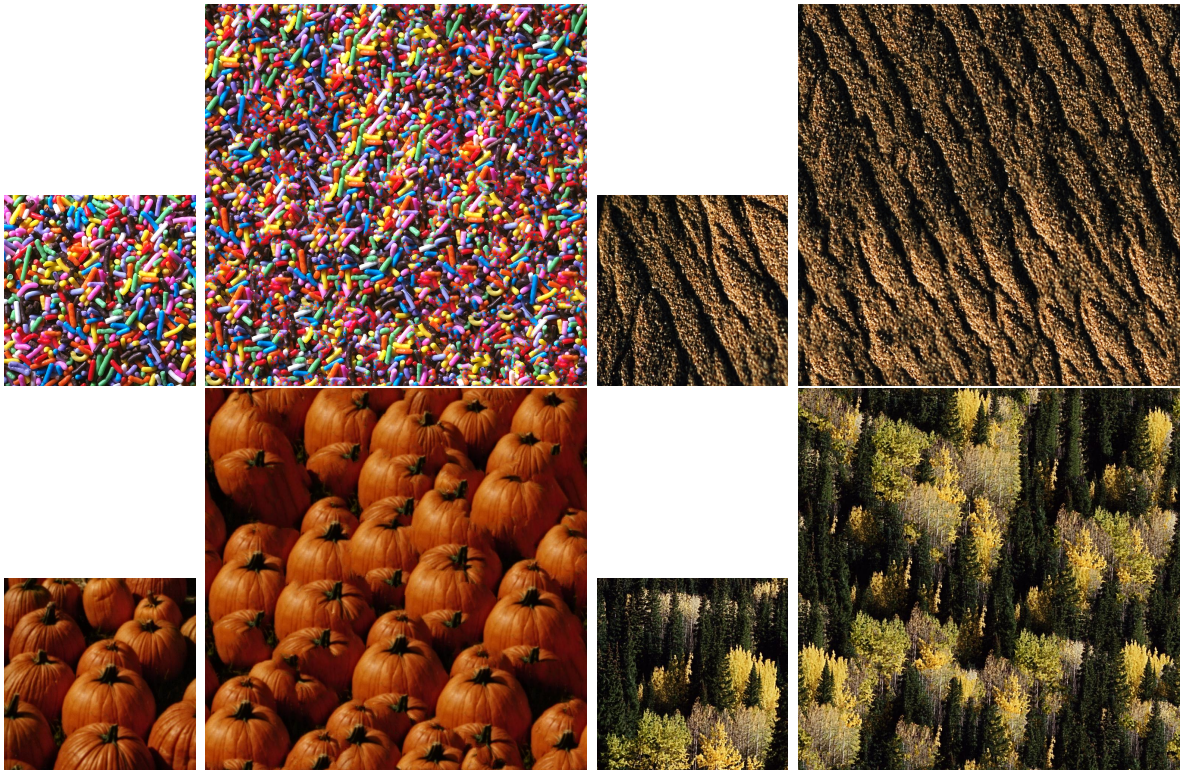


Abbildung 4.1: Ergebnisse Vollsynthese (1). Links Referenztextur (512×512 Pixel, VisTex Bibliothek [81]), rechts Syntheseergebnis (1024×1024 Pixel).

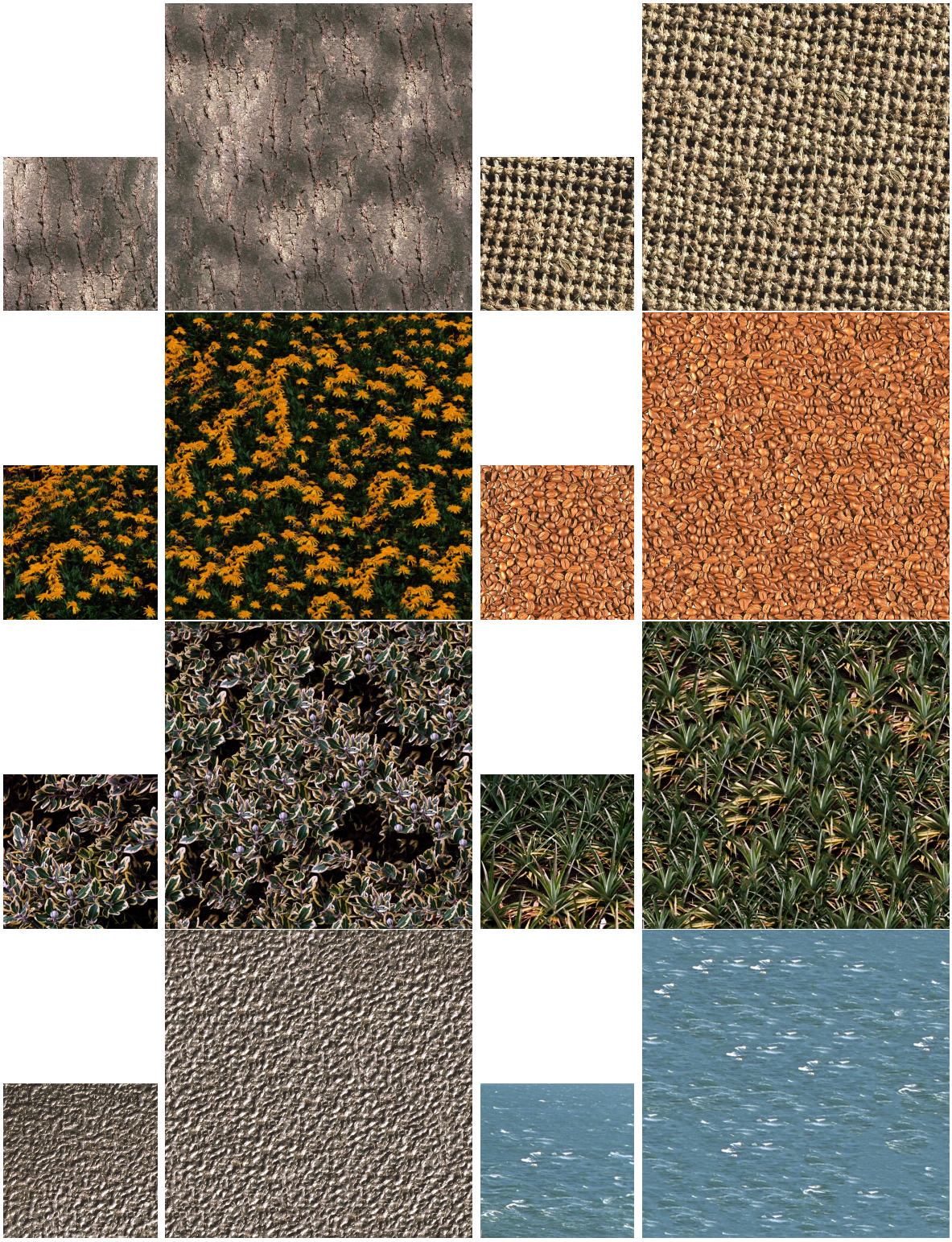


Abbildung 4.2: Ergebnisse Vollsynthese (2). Links Referenztextur (512×512 Pixel, VisTex Bibliothek [81]), rechts Syntheseergebnis (1024×1024 Pixel).

Als Parameter für die Synthese der Ergebnisse aus Abbildung 4.1 wurden fünf Auflösungsstufen $L = 5$, eine Blocknachbarschaftsbreite mit $W_N = 9$ und Blockgröße $W_B = 3$ gewählt. Die Fläche der 512×512 Pixel großen Referenz wurde auf 1024×1024 vervierfacht, wobei für die Synthese der letzten Auflösungsstufe eine LBG-Partitionierung mit $p = 50$ Partitionen berechnet wurde. Die Synthesezeit betrug weniger als 30 Sekunden pro Beispiel auf einer $3GHz$-CPU. Für die Synthese der Baumkronen wurde eine Varianzminimierung der Flächennutzungsstatistik durchgeführt; alle anderen Ergebnisse sind mit einer Varianzoptimierung der Flächennutzungskarte zur Syntheselaufzeit erzielt worden. Letzteres liefert insbesondere bei der Synthese der Sandtextur oben rechts ein interessantes Ergebnis, da die Referenztextur unterschiedliche Helligkeitszonen mit gleicher Struktur aufweist. Der Algorithmus synthetisiert zunächst aus der dunkleren Zone der Texturreferenz und wird durch die immer schlechtere Bewertung der dunklen Zonen in der Nutzungsstatistik dazu „gezwungen“, hellere Bereiche nahtlos anzuschließen. Der Einfluß des Bewertungsterms für die Flächennutzung wurde für dieses Beispiel sehr gering gewählt, um die Steuerbarkeit des Prozesses zu demonstrieren. Abbildung 4.2 zeigt Resultate der Vollsynthese, bei denen mit den gleichen Parametern, aber ohne Optimierung der Flächennutzung gearbeitet wurde, weil Selbstwiederholung bei den dargestellten Referenztexturen nicht störend in Erscheinung tritt. Die Metalltextur unten links weist ähnliche Eigenschaften wie die Sandtextur aus Abbildung 4.1 auf. Da ohne Auswertung der Flächennutzungsstatistik synthetisiert wurde, verweilt der Syntheseal-gorithmus dauerhaft in dem helleren Texturbereich unten rechts in der Referenz.

Für alle Ergebnisse gilt, daß durch das Übertragen von zusammenhängenden und optimal aneinander anschließenden Regionen die Charakterisitik der Textur auch ohne tiefergehendes Verständnis des eigentlichen Texturentstehungsprozesses sehr gut reproduziert wird. Bei inhomogenen Texturen wird eine der Texturreferenz vergleichbare Merkmalsverteilung durch Optimierung der Flächennutzung erreicht.

4.2 Synthese mit Randbedingungen

In Abbildung 4.3 sind Ergebnisse der Textursynthese mit Randbedingungen dargestellt. Jeweils oben ist die unvollständige Arbeitsfläche, darunter das Syntheseprodukt abgebildet. Die verwendete Arbeitsmaske entspricht der undefinierten Arbeitsfläche. Als Texturreferenz wurde ausschließlich die definierte Arbeitsfläche und keine weitere, externe Quelle genutzt. Die Syntheseparameter für die Erstellung der 1024×1024 Pixel großen Ausgangsflächen waren $L = 5, W_N = 9, W_B = 3$ oder $L = 6, W_N = 7, W_B = 3$. Abbildung 4.4 zeigt weitere Syntheseergebnisse, bei denen aufgrund der Inhomogenität der Referenztexturen eine Optimierung der Flächennutzung erforderlich war.

Wie in Abschnitt 3.2.6 beschrieben, wird die Syntheseoberfläche als Torus betrachtet, d.h. linke und rechte sowie obere und untere Kante des Bildes schließen nahtlos aneinander an. Die Syntheseprodukte sind somit Kacheln, die an sich selbst angeschlossen werden können, was in Abbildung 4.5 veranschaulicht ist.

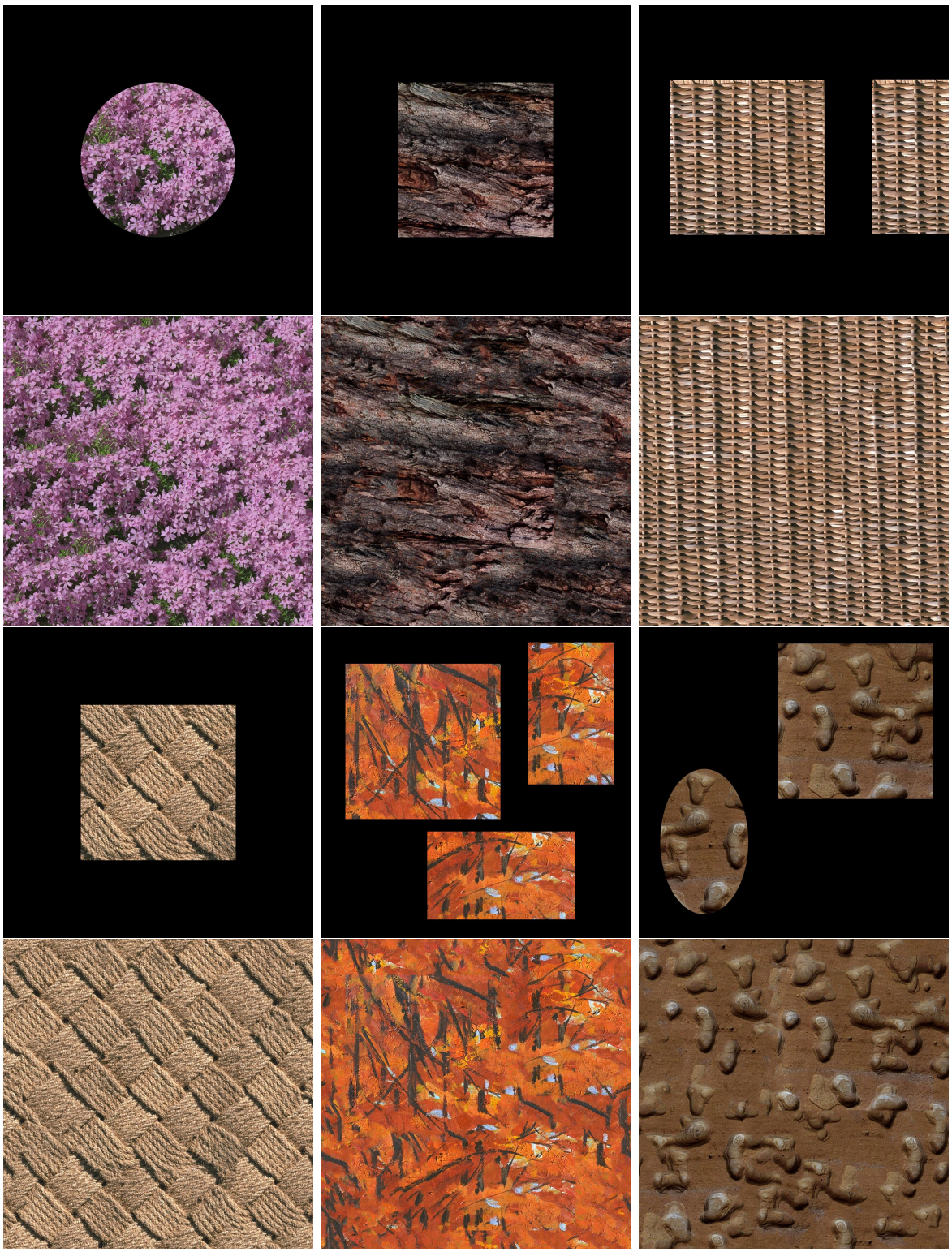


Abbildung 4.3: Ergebnisse Textursynthese mit Randbedingungen (1). Jeweils oben unvollständige Arbeitsfläche mit Referenztextur aus der VisTex Bibliothek [81], unten das Syntheseergebnis (1024×1024 Pixel).

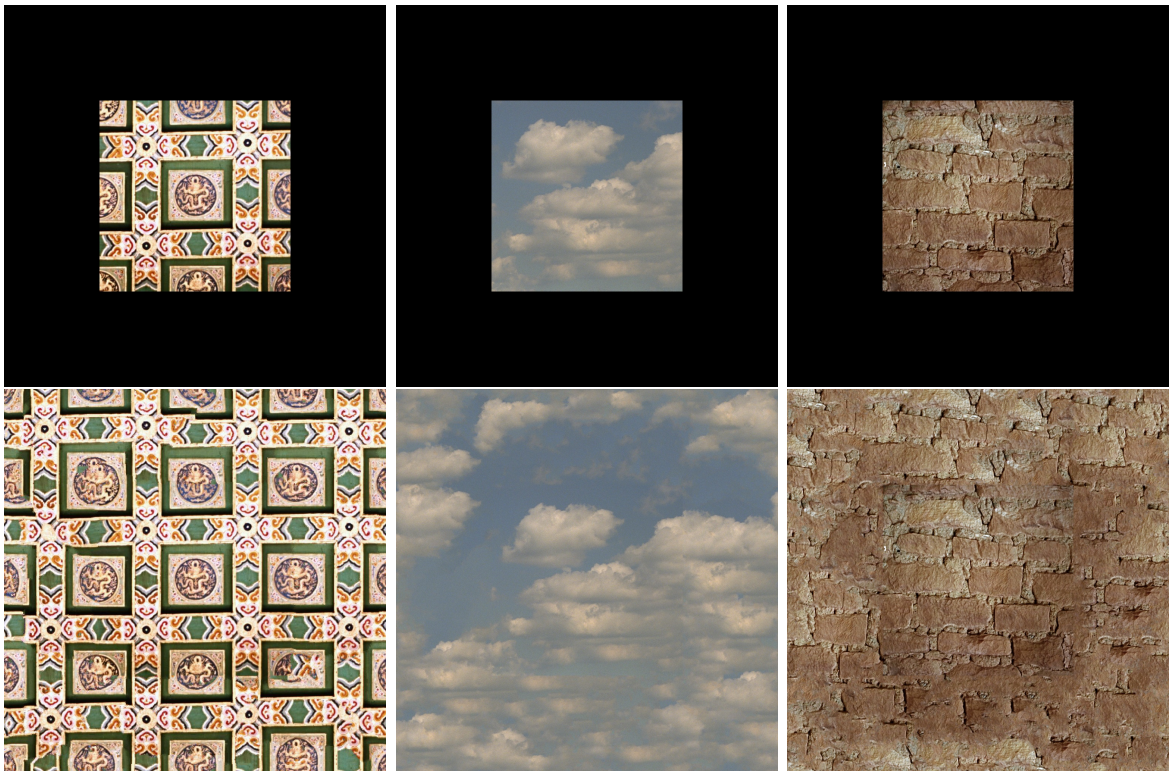


Abbildung 4.4: Ergebnisse Textursynthese mit Randbedingungen (2). Jeweils oben unvollständige Arbeitsfläche mit Referenztextur aus der VisTex Bibliothek [81], unten das Syntheseergebnis (1024 × 1024 Pixel).

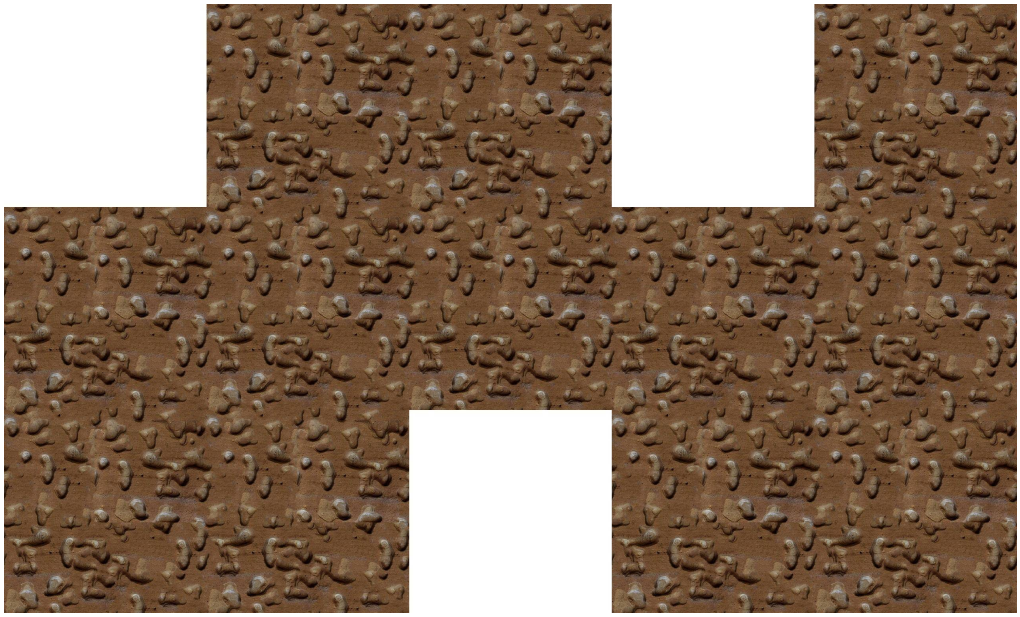


Abbildung 4.5: Texturkacheln: Aufgrund der Torusanschlußbedingungen der Syntheseoberfläche können Syntheseprodukte nahtlos an sich selbst angeschlossen werden, Beispiel Steintexturkachel aus Abbildung 4.3 unten rechts.

4.3 Mischen von Texturen

Durch die Nutzung einer Auflösungspyramide ergibt sich eine weitere Eingriffsmöglichkeit in den Textursyntheseprozeß: das Mischen von lokalen und globalen Merkmalen. So kann z.B. eine gefilterte und unterabgetastete Auflösungsstufe einer großen Referenztextur genutzt werden, um eine niedrige Auflösungsstufe zu synthetisieren, die eine genaue Reproduktion der globalen Merkmale liefert. In höheren Auflösungsstufen werden anschließend nur kleine, ausgewählte Ausschnitte der großen Referenztextur für die Verfeinerung der globalen Struktur genutzt. Die Idee für diesen Ansatz ist, daß die lokale Struktur oft redundant ist und die Verarbeitung der vollständigen Referenztextur einen hohen rechnerischen Aufwand bedeutet. Durch geeignete Auswahl der Texturreferenz für höhere Auflösungsstufen kann diese Redundanz verringert und die Texturierungsgeschwindigkeit erhöht werden.

Das Mischen von verschiedenen Texturauflösungen erlaubt auch einen interaktiven Eingriff in den Syntheseprozeß. Globale Merkmale können vom Anwender modifiziert werden, bevor die höheren Auflösungsstufen erzeugt werden. Abbildung 4.6 veranschaulicht beide Prinzipien. Nach der Synthese der Pyramidenstufe $l = 2$ mit der vollständigen Referenztextur (obere Reihe) wurde dem niederfrequenten Zwischenergebnis ein Text überlagert. Anschließend werden mit einem kleineren Ausschnitt der Referenztextur die Pyramidenstufen $l = 1$ (nicht dargestellt) und $l = 0$ erzeugt. Der überlagerte Text integriert sich in die Globalstruktur des Ergebnisses.

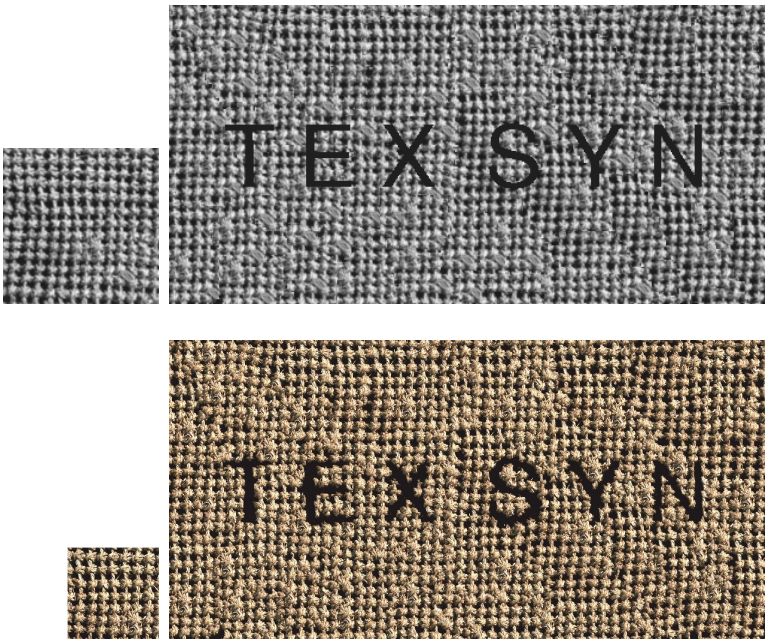


Abbildung 4.6: Mischen von Texturen. Oben: Auflösungsstufe $l = 2$, Referenztextur und Syntheseprodukt mit überlagertem Text, unten: Stufe $l = 0$, Referenztextur und Syntheseprodukt.

5 Anwendungen

5.1 Erzeugung künstlicher Luftbilder

5.1.1 Szenario

Die Themen Navigation und ortsbezogene Dienste (location based services, LBS) gewinnen in der Informationsgesellschaft immer mehr an Bedeutung. Durch GPS (Global Positioning System [40] [26]) ist eine präzise, satellitengestützte Positionsbestimmung im Freien möglich; die flächendeckend verfügbare GSM-Mobilfunktechnologie (Global System for Mobile communications [86] [46]) ermöglicht über Triangulation eine Ortbestimmung auch in geschlossenen Umgebungen und die mobile Nutzung von Datendiensten. Durch den Fortschritt in der Mikroelektronik sind leistungsfähige, mobile Endgeräte mit Anbindung an GSM und GPS zum Massenartikel geworden. Typische Anwendungen sind neben der Kommunikation die Navigation im Fahrzeug oder zu Fuß sowie die Abfrage von aktuellen Zusatzinformationen zum jeweiligen Aufenthaltsort. Die Grundlage dafür ist digitales Kartenmaterial der Umgebung, das je nach Speicherleistung des Endgerätes lokal verfügbar ist (onboard) oder zum Zeitpunkt der Anfrage ausschnittsweise über Datendienste übertragen wird (offboard). Durch Kommunikation mit einem externen Dienstleister ist es prinzipiell möglich, den Onboard-Informationsstand ständig zu aktualisieren bzw. zu erweitern.

Die grafische Visualisierung der meist vektoriellen Karteninformation wird durch Symbolik unterstützt, wobei bestimmte Informationen dem Anwendungszweck entsprechend hervorgehoben werden. Der Benutzer muß die Kartendarstellung interpretieren und in Bezug auf seine Umgebung registrieren, um die Zusatzinformation zur Zielfindung einsetzen zu können. Ein Problem tritt dann auf, wenn dem Anwender die Karteninterpretation nicht gelingt, weil er z.B. nicht ausreichend mit der verwendeten Symbolik vertraut ist.

Ein Lösungsansatz ist nun, die spezielle symbolische Darstellung der Kartographie durch den Menschen vertraute, natürliche Texturen zu ersetzen. So können z.B. Waldgebiete, die in der Karte durch Grüntöne und Signaturen mit Λ- und Ω-ähnlichen Symbolen (Nadel-/Laubwald) gekennzeichnet sind, durch photographische Waldtextur aus einem Luftbild ersetzt werden. Dabei muß berücksichtigt werden, daß die Karteninterpretation durch die zusätzliche bildliche Textur unterstützt und nicht beeinträchtigt wird. Die Verwendung von echten Luftbildern mit überlagerter symbolischer Information schließt

sich aus, da die individuelle Übertragung von Bilddaten insbesondere für den mobilen Einsatz kostspielig und zeitaufwendig ist. Die Erzeugung natürlicher Texturen als Ersatz für Kartensymbolik läßt sich mit viel geringerem Aufwand durch eine Textursynthese bewerkstelligen. Die geometrischen Daten der flächenhaften Kartenobjekte liegen bereits onboard vor oder werden wie bisher an das Endgerät übertragen. Sie liefern die Berandung der Synthesebereiche. Über den Objekttyp wird onboard aus einem Texturkatalog eine kleine, aber repräsentative Referenztextur gewählt und die Synthese ausgeführt. Nach Fertigstellung der unterschiedlichen Objekttypen steht eine Karte zur Verfügung, die visuell große Ähnlichkeit zum Luftbild hat und der symbolische Information überlagert werden kann. Durch die Nutzung einer Textursynthese anstelle einfacher Repetierung von Texturelementen wird eine wesentlich höhere Darstellungsqualität erzielt. Der zusätzliche Aufwand für diese neue Methodik beschränkt sich auf die Onboard-Haltung eines kompakten Texturreferenzkatalogs und der Rechenzeit der Synthese.

5.1.2 Datenbasis

Die Applikation nutzt ein Geographisches Informationssystem (GIS), für Deutschland das länderübergreifende ATKIS (Amtliches Topographisch-Kartographisches Informationssystem [45]), auf das im Rahmen dieser Arbeit zugegriffen werden konnte. Ein Geoinformationssystem ermöglicht die Modellierung, Verwaltung und Analyse raumbezogener Daten und ihrer Beziehungen [13]. Kern jedes GIS ist eine spatiale Datenbank, die durch hierarchische Indizierungstechniken raumbezogene Anfragen auf den Datenstamm effizient ausführen kann. Weitere Datenquelle ist ein regionsspezifischer Referenztexturkatalog aller für die Visualisierung relevanten GIS-Objektarten, der aus repräsentativen Luftbildausschnitten erstellt wird. Abbildung 5.1 zeigt die flächenhaften GIS-Objekte der Szene „Ravensburg“ und eine Referenztexturauswahl.

5.1.3 Luftbildsynthese

Zunächst werden für die Darstellung relevante, linienhafte Objekte, wie z.B. Verkehrswege, erzeugt. Diese sind im Geoinformationssystem als Vektorpfad abgelegt und besitzen ein Attribut für die Objektbreite. Aufgrund der im Vergleich zu flächenhaften Objekten geringen Ausdehnung werden diese Strukturen nicht synthetisiert, sondern direkt als Linien mit unterschiedlicher Linienbreite gezeichnet. Für jede Objektart wird eine geeignete, einheitliche Farbe aus den Referenztexturen ermittelt. In Abbildung 5.2 sind für die im folgenden betrachtete Beispielszene „Ravensburg/Oberzell“ die ATKIS-Objekttypen „Straße“ (3101), „Wege“ (3102), „Schiene“ (3201) und „Fluß“ (5101) erzeugt worden.

Nach Erstellung der relevanten, linienhaften Objekte wird die Textursynthese für die flächenhaften Objektarten durchgeführt, die im Geoinformationssystem als Polygone bzw. geschlossene Vektorpfade verwaltet werden. Das eingesetzte Verfahren ist dabei eine spezielle Abwandlung der in Kapitel 3 beschriebenen Textursynthese mit Randbedingun-

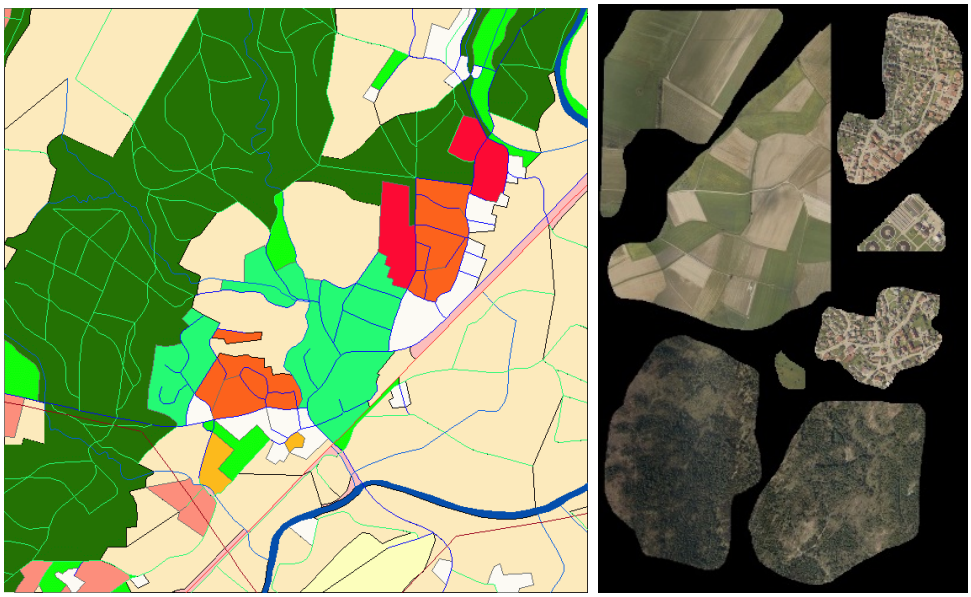


Abbildung 5.1: Datenbasis. Links flächenhafte GIS-Objekte, rechts Referenztexturen für verschiedene Objektklassen.

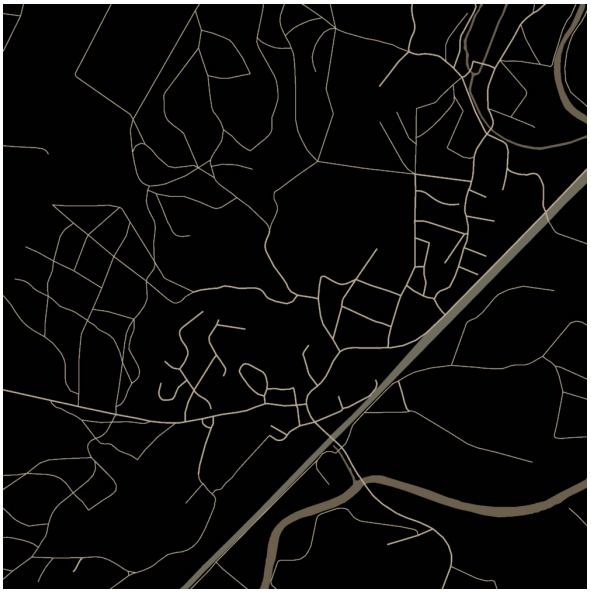


Abbildung 5.2: Erzeugung linienhafter GIS-Objekte. Umsetzung der ATKIS-Vektorbeschreibung für die Objektarten Straße, Wege, Schiene und Fluß.

gen. Die Vorgaben für den Syntheseprozeß bestehen aus den vektoriellen Flächenberandungen der zu erzeugenden Objekte und der auf der Arbeitsfläche bereits synthetisierten Textur. Im ersten Schritt besteht letztere nur aus den bereits gezeichneten, linienhaften Strukturen. Es wird eine Objektart ausgewählt und deren Polygonbeschreibung in eine

Arbeitsmaske überführt, siehe Abbildung 5.3 links. Im Gegensatz zu der in Abschnitt 3.1 beschriebenen Arbeitsmaske, die das zu bearbeitende Gebiet und die Texturreferenz maskiert, müssen hier drei Fälle unterschieden werden: Arbeitsgebiet, definierte und unbekannte Fläche. Das Verfahren synthetisiert in den Arbeitsgebieten und paßt dafür die variable Nachbarschaftsmaske, siehe Abschnitt 3.2.6, nur an neu synthetisierte und definierte, aber nicht an unbekannte Flächen an, um Kausalität des Syntheseprozesses zu gewährleisten.

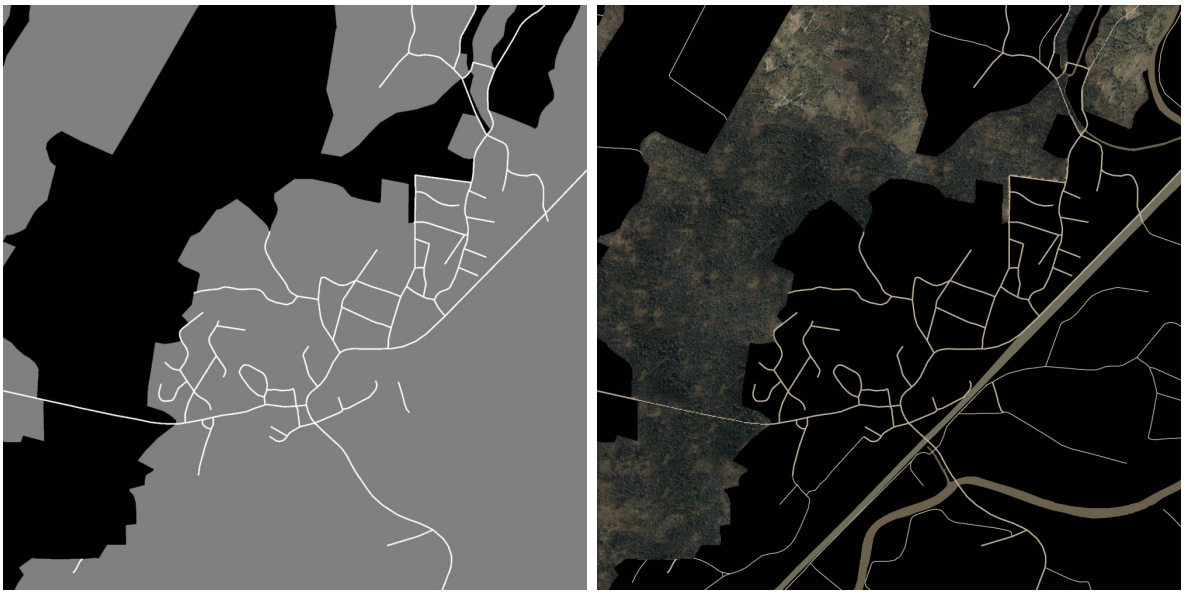


Abbildung 5.3: Synthese flächenhafter Objektarten (1). Links Arbeitsmaske Objektart „Wald“: Arbeitsgebiet (schwarz), definierte Umgebung (weiß), unbekannte Umgebung (grau). Rechts Syntheseresultat „Wald“ aufbauend auf den linienhaften Strukturen aus Abbildung 5.2 und der Wald-Referenztextur aus Abbildung 5.1 rechts.

Ein Problem stellt die Auswahlreihenfolge der verschiedenen, flächenhaften Objektarten dar. Im ersten Schritt werden ausschließlich linienhafte Objekte erzeugt, die als Startpunkt für die Textursynthese dienen können. Die Linienstrukturen haben aber eine geringe Ausdehnung und sind untexturiert, so daß sich entlang eines Linienverlaufs immer identische Randbedingungen einstellen, was letztlich zu geringen Variationen im Syntheseprodukt führt. Daher muß für die Synthese von Luftbildern die im Abschnitt 3.7 beschriebene Varianzoptimierung eingesetzt werden, um die Uniformität der Startbedingungen auszugleichen. Die visuelle Synthesequalität von strukturierten Texturen wird durch Varianzoptimierung eher beeinträchtigt, als die von stochastischen Texturen. Daher werden zuerst Objektarten wie Wald, Grünland oder Acker synthetisiert und anschließend - mit besseren Randbedingungen und schwächerer Varianzoptimierung - stärker strukturierte Texturen, wie sie z.B. die Objektarten Siedlungs- oder Industriegebiet aufweisen. Optimal wäre eine quasi gleichzeitige Erzeugung der verschiedenen, flächenhaften Objekte, was Gegenstand weiterer Untersuchungen sein kann.

Die rechte Seite von Abbildung 5.3 zeigt das Syntheseergebnis der Objektart „Wald“ (4107). Bei dieser speziellen Texturklasse muß berücksichtigt werden, daß aus der Luft mit hoher Wahrscheinlichkeit nur breite Straßen innerhalb der Waldgebiete zu erkennen sind. Daher wurde die Objektart „Wege“ für die Festlegung der Randbedingungen von vornherein ausgeblendet. Abbildung 5.4 zeigt exemplarisch die weiteren Stufen „Ackerland“ (4101) und „Ortslage„ (2101) der Objekterzeugung. In den Arbeitsmasken werden die in vorangegangenen Schritten bearbeiteten Flächen als „definiert“ gekennzeichnet.

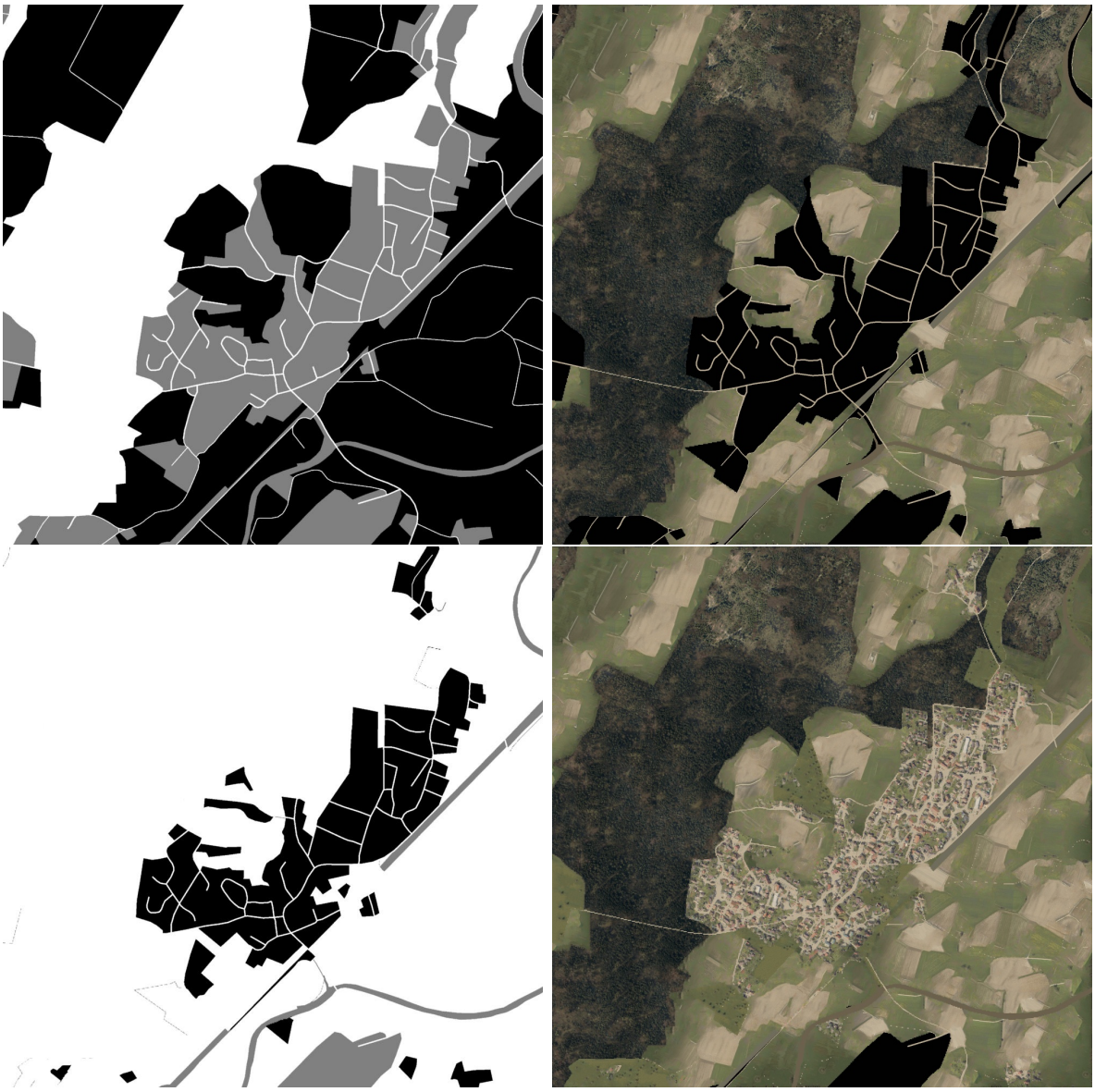


Abbildung 5.4: Synthese flächenhafter Objektarten (2). Links Arbeitsmasken der Objektarten „Ackerland“ und „Ortslage“, rechts Syntheseresultate mit Referenztextur aus Abbildung 5.1 rechts.

Nach Erzeugung der Objektarten „Grünland“ (4102) und „Industrie“ (2112) ist das synthetische Luftbild der Beispielszene fertiggestellt. Abbildung 5.5 zeigt oben links das Resultat und unten vier Ausschnittsvergrößerungen. Die Auflösung des künstlichen Luftbildes ist prinzipiell nur durch die Referenztextur begrenzt. Das Beispiel wurde mit einer Bodenauflösung von 1 Meter/Pixel berechnet, wodurch Häuser bereits sehr detailliert reproduziert werden. Dabei kann es sich als problematisch erweisen, daß ein zu hoher Detaillierungsgrad den Eindruck erweckt, eine reale Darstellung zu nutzen.

Abbildung 5.5: Synthetisches Luftbild (links oben) mit vier Ausschnittsvergrößerungen (untere Reihe) und Orthophoto (rechts oben).

Im direkten Vergleich werden die Unterschiede zwischen echtem und synthetischem Luftbild schnell deutlich, siehe Abbildung 5.5 oben. Die Übergänge zwischen aneinandergrenzenden, unterschiedlichen Objektarten sind im synthetischen Ergebnis zu hart, es finden sich zahlreiche Syntheseartefakte, z.B. Fehler in der Globalstruktur, Aliasing an linienhaften Randbegrenzungen aufgrund des Synthese-Blocktransfers und Repetierung von bestimmten Texturelementen. Außerdem wird durch die eingeschränkte Zahl von Objektarten die natürliche Vielfalt und Komplexität des Luftbild-Orthophotos nicht erreicht. Dennoch stellt das Syntheseprodukt eine Approximation der realen Szene dar und ist eine geeignete Grundlage für die eingangs beschriebenen Anwendungen.

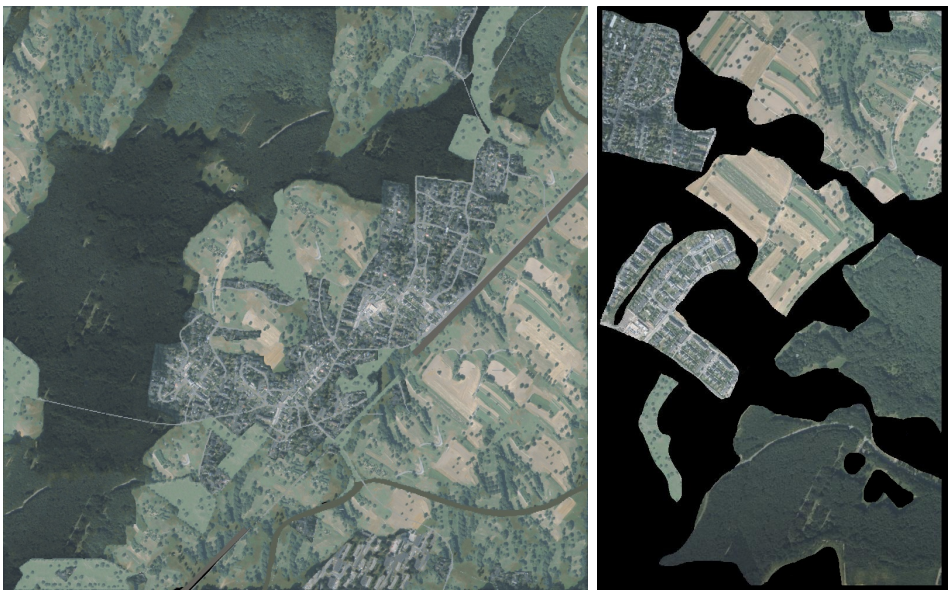


Abbildung 5.6: Synthetisches Luftbild mit Variation des Referenzkatalogs.

Abbildung 5.6 zeigt ein Syntheseergebnis, das ausgehend von den gleichen Geoinformationsdaten mit einer Variation des Referenztexturkatalogs erzeugt wurde. Aus dem Vergleich der beiden synthetischen Luftbilder läßt sich ableiten, daß der visuelle Gesamteindruck nicht nur durch die Synthesequalität, sondern auch durch Eigenschaften des Luftbildes und die Auswahl der Referenztextur bestimmt wird. Das Luftbild bzw. Orthophoto als Quelle der Referenztextur kann Farbabweichungen, Unschärfen, partielle Abschattungen, etc. aufweisen, die im synthetischen Luftbild entsprechend wiedergegeben werden. Dies muß bei der Erstellung von Referenztexturkatalogen berücksichtigt werden, die gleichzeitig auf ihre Eignung für die Überlagerung mit symbolischer Information, siehe Abschnitt 5.1.4, zu prüfen sind. Mit der Auswahl der Referenztextur werden auch strukturelle Aspekte der Reproduktion beeinflußt. Die Beispielszene beschreibt einen eher ländlichen Raum, der für die Objektart „Ortslage“ einzelne Häuser mit größeren Grundstücken verlangt. Diese Bedingung wird durch Verwendung entsprechender Referenztextur erfüllt, siehe Abbildung 5.5 oben links. Wird nun eine Referenztextur für die Ortslage gewählt, die Vorstadtbereiche mit Mehrfamilienhäusern und kleineren Grundstücken wiedergibt, erscheint das Syntheseresultat kompakter, siehe Abbildung 5.6. Ebenso spiegeln sich regionale Unterschiede in der Landschaftsentwicklung in den strukturellen Merkmalen wieder. In der Referenztextur der Objektart „Ackerland“ des alternativen Texturkatalogs treten häufig solitäre Bäume auf, die entsprechend oft im Syntheseergebnis reproduziert werden. In der Realität der Beispielszene existieren derartige Strukturen nicht, siehe Abbildung 5.5 oben rechts. Ein optimaler Referenztexturkatalog sollte regional diversifiziert sein und somit unterschiedliche Ausprägungen der verschiedenen Objektarten des Geoinformationssystems bereitstellen.

5.1.4 Kartographische Generalisierung

Wie bereits beschrieben, wird für die Erzeugung der synthetischen Luftbilder auf das Geoinformationssystem ATKIS zugegriffen. Die im ATKIS verwalteten vektoriellen oder polygonalen Objektbeschreibungen basieren auf Vermessungsdaten und geben im Rahmen der erreichbaren Genauigkeit die realen Begrenzungen von flächenhaften bzw. den Verlauf von linienhaften Objekten wieder. Da bei einer verkleinerten Darstellung der Geoinformationsdaten für die Anwendung relevante Informationen verloren gehen können, müssen diese in ein vereinfachtes, zweckbezogenes Kartenmodell überführt werden.

Vereinfachung bedeutet kartographische Generalisierung, also die Auswahl des Wichtigsten, Wesentlichen und dessen zielgerichtete Verallgemeinerung. Nach [47] wird sie in rein geometrische (Vereinfachung, Vergrößerung, Verdrängung) und geometrisch-begriffliche Generalisierung (Zusammenfassen, Selektieren, Klassifizieren, Bewerten) unterteilt. Für die geometrische Generalisierung der Geoinformationsdaten existieren Verfahren [91], die speziell an die Anforderungen des mobilen Einsatzes adaptiert sind [11] [48].

Abbildung 5.7 zeigt zwei Karten der Beispielszene - eine symbolische und eine, die mit Textursynthese auf Basis der generalisierten Objektbeschreibung erstellt wurde. Der synthetisierten Karte wurden ausgewählte, symbolische Informationen überlagert, wie sie z.B. für eine Navigationsanwendung benötigt werden.

Abbildung 5.7: Symbolische und synthetische Karte.

Das Vergrößerungsprinzip der Generalisierung kann in Form einer Auflösungsvariation der Referenztextur umgesetzt werden. Damit entfernt man sich von dem realistischen Charakter des Luftbildes und bewegt sich in Richtung der symbolischen Darstellung. In Abbildung 5.8 sind synthetische Karten mit einer um Faktor zwei bzw. vier erhöhten Tex-

turauflösung dargestellt. Verglichen mit der maßstabsrichtigen Darstellung werden strukturelle Fehler des Syntheseresultats durch die Auflösungssteigerung stärker betont. Wie bereits oben erwähnt, wird durch die hohe Detaillierung eine Korrespondenz von Elementen des Syntheseresultats (z.B. Gebäuden) zur tatsächlichen Anordnung dieser Elemente suggeriert, die in der Realität nicht vorhanden ist. Abbildung 5.8 veranschaulicht dieses Problem insbesondere durch die Objektart „Ortslage“. Aufgrund dieser Nachteile wird deutlich, daß eine Generalisierung durch nicht maßstabsgerechte Textursynthese keinen erhöhten Nutzen für die Interpretation der Darstellung bewirkt.

Abbildung 5.8: Synthetische Karte mit erhöhter Referenztexturauflösung. Links zweifache, rechts vierfache Auflösung.

Anhand der Beispiele wird demonstriert, daß die Textursynthese für Karten auf Basis eines generalisierten Geoinformationssystems es ermöglicht, gewisse Teile der Symbolik durch selbsterklärende, natürliche Texturen zu ersetzen. Dieses Verfahren kann als eine unterstützende, neue Visualisierungsform angesehen werden, die es erlaubt, die Erläuterung des Kartenmaterials in Form von Legenden auf wenige, anwendungsspezifische Elemente zu reduzieren und damit einen schnelleren Zugang zur Information zu erhalten. Der methodische Aufbau in Form einer schrittweisen Synthese der einzelnen Objektarten, wie für die vorgestellten Ergebnisse umgesetzt, bildet die Basis für die weitere Verfahrensentwicklung. Durch Nutzung von zusätzlichen GIS-Objektarten, wie z.B. die Begrenzung von Flurstücken, werden die Randbedingungen für den Syntheseprozeß genauer definiert, wodurch eine höhere Qualität insbesondere für stark strukturierte Texturen erzielt werden kann.

5.2 Segmentierung von Fernerkundungsdaten

5.2.1 Allgemeine Definition

Die Bildsegmentierung soll eine sinnvolle Zuordnung von Bildpunkten zu Objekten durchführen (Klassifikation von Bildpunkten) oder verschiedene Bildpunkte zu sinnvollen Objekten zusammenfassen (Partitionierung des Bildes). Der Begriff „sinnvoll“ bezieht sich dabei auf das intuitive, menschliche Verständnis für die gestellte Aufgabe, welches durch Erfahrung und Wissen geprägt wird.

In der digitalen Bildverarbeitung wird in überwachte und nicht-überwachte Segmentierungsverfahren unterschieden. Die überwachten Verfahren erhalten als Form von Wissen Beispiel- bzw. Anlernbilder, mit denen festgelegt wird, wie die zu unterscheidenden Klassen aussehen. Das Verfahren erzeugt auf Basis der Referenzbilder eine Segmentierung eines Eingangsbildes, die als Labelbild ausgegeben wird. Das Labelbild kodiert für jeden Bildpunkt die Klassenzugehörigkeit. Nicht-überwachte Segmentierungsverfahren verzichten auf die Referenzbilder und führen eine Klassifikation bzw. Partitionierung über Berechnung von statistischen Bildmerkmalen durch [56] [66] [93].

Bildsegmentierungen können regionen- oder kantenbasiert durchgeführt werden. Regionenbasierte Ansätze verwenden meist lokale Umgebungen (Markov-Eigenschaft) für die Klassifikation von Bildpunkten. Die Klassenzuordnung für Bildpunkte im Grenzbereich zweier benachbarter, unterschiedlicher Regionen ist grundsätzlich ungenau, da die lokale Umgebung dort Bildpunkte beider Klassen gleichzeitig beinhaltet (Koronaeffekt). In diesen Übergangszonen kann ein kantenbasiertes Verfahren bessere Ergebnisse liefern. Dazu werden beispielsweise Konturpunkte über Gradientenoperationen bestimmt und die relevanten Punkte für den Kantenverlauf miteinander verbunden [63].

5.2.2 Anwendungsgebiete

Segmentierungsverfahren spielen in der digitalen Bildverarbeitung eine wichtige Rolle und sind in fast jeder Anwendung zu finden. Bildgebende Verfahren der Medizin wie Ultraschall oder Röntgen liefern Daten, die für die Diagnostik und Planung von Maßnahmen analysiert und somit segmentiert werden müssen [65] [90]. In der industriellen Produktion wird Qualitätskontrolle mit Hilfe von Bildverarbeitung durchgeführt. Die Segmentierung ist dabei Grundlage für die Bestimmung von Lage, Form und Größe von Objekten, um Abweichungen zu Sollvorgaben automatisch erkennen zu können [21] [1].

Eine sehr anspruchsvolle Anwendung ist die Analyse von Fernerkundungsdaten, z.B. für Umweltüberwachungsaufgaben [75] [89] und Aktualisierung von Geoinformationssystemen [76] [16]. Wegen der großen anfallenden Bilddatenmengen und der hohen Komplexität des Datenmaterials werden robuste und produktive Techniken für die Analyse und Objektextraktion benötigt. Neuere Entwicklungen für diese Aufgaben sind wissensba-

sierte Bildinterpretationssysteme, die den Analyseablauf steuern und kontrollieren [78] [17]. Diese Systeme verfügen über ein variables Szenenmodell, das in einem hierarchischen Aufbau von Knoten, die Objekte repräsentieren, und Kanten, die Objektrelationen wiederspiegeln, das *a priori* Wissen über die unterschiedlichen, erwarteten Szenenbestandteile modelliert. Das Szenenmodell wird im Laufe der Analyse in ein Instanzennetz überführt. Dafür verfügen die Knoten über Anbindungen an Bildsegmentierungsverfahren zur Klassifikation der Eingangsdaten (Top-Down) und Operatoren für die Bewertung und Gruppierung der Segmentierungsergebnisse (Bottom-Up) [18] [15].

Das Ergebnis und die Effizienz einer wissensbasierten Bildinterpretation wird also wesentlich von der Qualität der Segmentierungsergebnisse bestimmt. Übersegmentierung führt zu einem erheblichen Aufwand bei der Bewertung; Fehlklassifikation läßt sich nur durch gleichzeitige Bewertung der Ausgaben unterschiedlicher Segmentierungsverfahren erkennen. Daher ist es wichtig, für die auftretenden Objekttypen verschiedene, robuste Segmentierungsverfahren in das Interpretationssystem zu integrieren. Bei der Analyse von Fernerkundungsdaten wird die regionenbasierte Textursegmentierung zur Unterteilung und Klassifikation von flächenhaften Objekten eingesetzt.

5.2.3 Übertragung der Textursynthese auf die Textursegmentierung

Die meisten der in Kapitel 1 beschriebenen Textursyntheseverfahren lassen sich in modifizierter Form auch für eine regionenbasierte Textursegmentierung anwenden. Sie verwenden Texturmodelle, die für die Unterscheidung von verschiedenen Texturklassen ebenso geeignet sind wie für die Texturreproduktion. Dies gilt insbesondere für die Ansätze mit explizitem Markov-Zufallsfeld-Texturmodell, wie z.B. [41] [37] [107], die seit vielen Jahren für beide Anwendungsfelder eingesetzt werden. Das im Rahmen dieser Arbeit entwickelte Syntheseverfahren arbeitet mit einem impliziten Markov-Zufallsfeld-Texturmodell und kann so adaptiert werden, daß es eine überwachte, regionenbasierte Textursegmentierung für eine Auflösungsstufe der Bildpyramide ausführt. Eine multiskalige Erweiterung, die Segmentierungsergebnisse unterschiedlicher Auflösungsstufen fusioniert, wird über eine Nachverarbeitung realisiert. Das Verfahren wird gemäß dem Ansatz im folgenden Text mit „iMRF“ bezeichnet.

Textursegmentierung einer Auflösungsstufe

Um die Segmentierung im Eingangsbild einer beliebigen Stufe der Auflösungspyramide durchzuführen, wird die Nachbarschaftsumgebung $\vec{N}(P^{seg})$ des jeweils zu klassifizierenden Pixels P^{seg} mit allen Umgebungen $\vec{N}(P_j^{ref})$ der bereits klassifizierten Anlerntextur verglichen. Die Anlerntextur entspricht dabei der Referenztextur des ursprünglichen Syntheseverfahrens, die Klassenzuordnung der Anlerntextur wird in Form eines Referenzlabelbildes $K(P_j^{ref})$ verwaltet. Als Ähnlichkeitsmaße werden wie bisher der L_1- oder L_2-Abstand der Nachbarschaftsvektoren verwendet. Zusammen mit der L_1-Norm können

auch alle in Abschnitt 3.6 beschriebenen Beschleunigungsverfahren eingesetzt werden. Nach Bestimmung des ähnlichsten Nachbarschaftsvektors $\vec{N}(P^{ref}_{min})$ zum gegebenen Suchvektor $\vec{N}(P^{seg})$ wird der Position von P^{seg} im Ausgabe-Labelbild die Klasse $K(P^{ref}_{min})$ zugewiesen, siehe Abbildung 5.9.

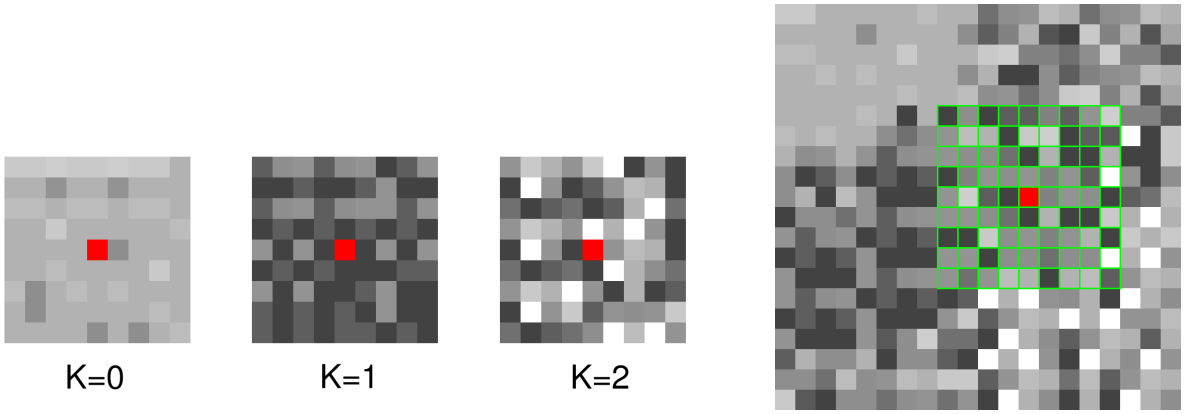


Abbildung 5.9: Prinzip der Segmentierung. Jede Nachbarschaftsumgebung des zu klassifizierenden Bildes (rechts) wird mit allen Nachbarschaftsumgebungen der Referenztextur verglichen (links drei Bespiele mit Klassenzuordnung $K = 0, 1, 2$). Über die Klassenzuordnung der ähnlichsten Referenzumgebung wird die Klassenzugehörigkeit des aktuellen Pixels festgelegt.

Wie in Abschnitt 3.2.3 beschrieben, werden nur die einbeschriebenen, vollständig definierten, symmetrischen Nachbarschaftsumgebungen der Referenztextur genutzt, so daß auf die variable Maske aus Abschnitt 3.2.6 verzichtet werden kann. Die Operation erfolgt im Bild spalten- und zeilenweise, wobei prinzipiell jede beliebige Reihenfolge zulässig ist. Durch den impliziten Markov-Zufallsfeld Ansatz entfällt - wie schon zuvor bei der Textursynthese - die Notwendigkeit, ein explizites Texturmodell durch eine Analyse der Referenztextur zu erstellen.

Um die Qualität des Segmentierungsverfahren bewerten zu können, wird zunächst auf Eingangsdaten zurückgegriffen, die aus natürlichen Texturen manuell zusammengestellt wurden, und für die somit ein ideales Segmentierungsergebnis bekannt ist. Abbildung 5.10 zeigt Anlern- und zugehöriges Labelbild sowie die Eingangsdaten und die bekannte, ideale Segmentierung. Die Regionen für das Anlernbild wurden direkt aus dem Eingangsbild entnommen. Durch Vergleich der idealen Segmentierung mit dem jeweiligen Segmentierungsergebnis läßt sich die Anzahl der unterschiedlichen Bildpunkte ermitteln und damit ein relatives Gütemaß berechnen.

Zusätzlich wird ein Vergleich mit den Ergebnissen eines Verfahrens von Gimel'farb und Zalesny [108] durchgeführt, das ein explizites, parametrisches Markov-Zufallsfeld Texturmodell nutzt und bisher als Top-Down-Operator für Textursegmentierung im Rahmen des wissensbasierten Bildinterpretationssystems „geoAIDA“ eingesetzt wird. Das Vergleichsverfahren wird zur Unterscheidung im folgenden Text mit „eMRF“ bezeichnet.

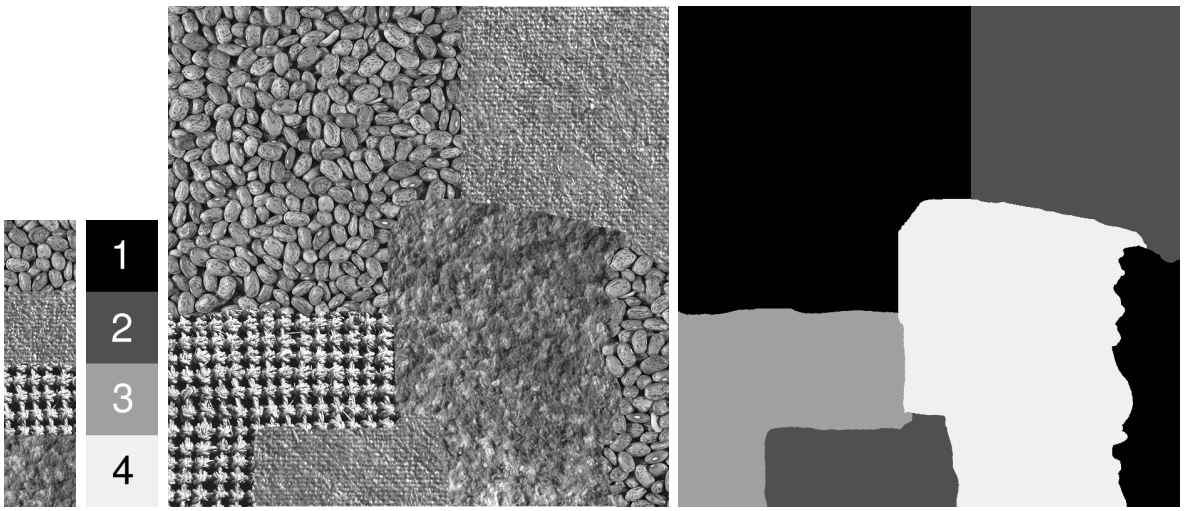


Abbildung 5.10: Testdaten für die Textursegmentierung. Von links nach rechts: Anlernbild, zugehöriges Labelbild (Nummerierung nur in dieser Darstellung für die Ergebnisdiskussion), Texturcollage als Eingangsdaten und ideale Segmentierung.

Abbildung 5.11 zeigt die Segmentierungsergebnisse beider Verfahren bei einer Nachbarschaftsumgebung von 9×9 Bildpunkten und maximal 16 Cliquen (siehe Abschnitt 2.4) für die Berechnung des expliziten Texturmodells. Die eMRF-Segmentierung liefert sehr unterschiedliche Ergebnisse für die einzelnen Texturtypen, wobei die Klassifikationsquote insgesamt nur 51.5% erreicht. Die Knotentextur (3) wird sehr gut von den anderen differenziert, was auf die ausgeprägten, strukturellen Merkmale und die starke Inhomo-

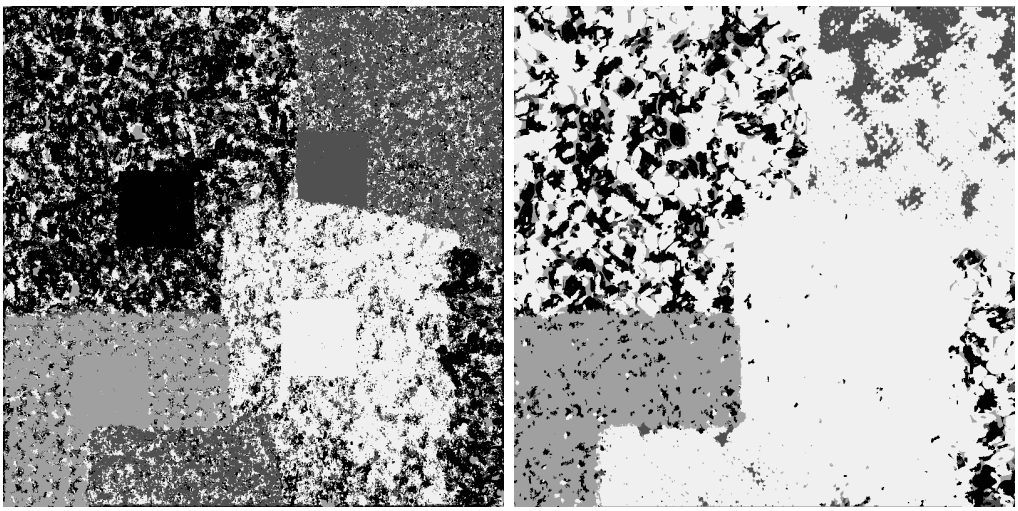


Abbildung 5.11: Segmentierung einer Auflösungsstufe. Links: entwickeltes iMRF Verfahren, rechts: eMRF Vergleichsalgorithmus.

genität des Grauwerthistogramms zurückzuführen ist. Die Trennung der beiden homogeneren Texturen (2)(4) gelingt dem zugrundeliegenden Texturmodell jedoch nicht. Das neu entwickelte iMRF-Verfahren erreicht eine Klassifikationsquote von 72.1%. Aufgrund des direkten Nachbarschaftsvergleichs werden die Bereiche, aus denen die Anlerntexturen entnommen wurden, richtig klassifiziert (homogene Quadrate im Segmentierungsergebnis). Werden diese selbstsegmentierten Anlernbereiche nicht berücksichtigt, beträgt die bereinigte Klassifikationsquote 68.9%. Das iMRF-Ergebnis weist interessanterweise keine großflächigen Fehlklassifikationen auf. Die Fehler besitzen eher eine Rauschcharakteristik und resultieren aus einem zu klein gewählten Nachbarschaftsfenster, wodurch größere, strukturelle Zusammenhänge nicht erkannt werden. Eine Verbesserung des Ergebnisses kann über zwei verschiedene Ansätze erzielt werden: durch eine größere Nachbarschaftsumgebung mit multiskaliger Segmentierung, wie im folgenden Abschnitt beschrieben, oder mittels Rauschreduktion in einer Nachverarbeitung. Die Fehlklassifikationen können als lokale, hochfrequente Störungen mit geringer räumlicher Ausdehnung betrachtet werden. Unterschreitet die Größe dieser Störungen einen vorgegebenen Schwellwert, werden sie der Umgebung zugeordnet. Das Ergebnis einer solchen Operation ist in Abbildung 5.12 dargestellt.

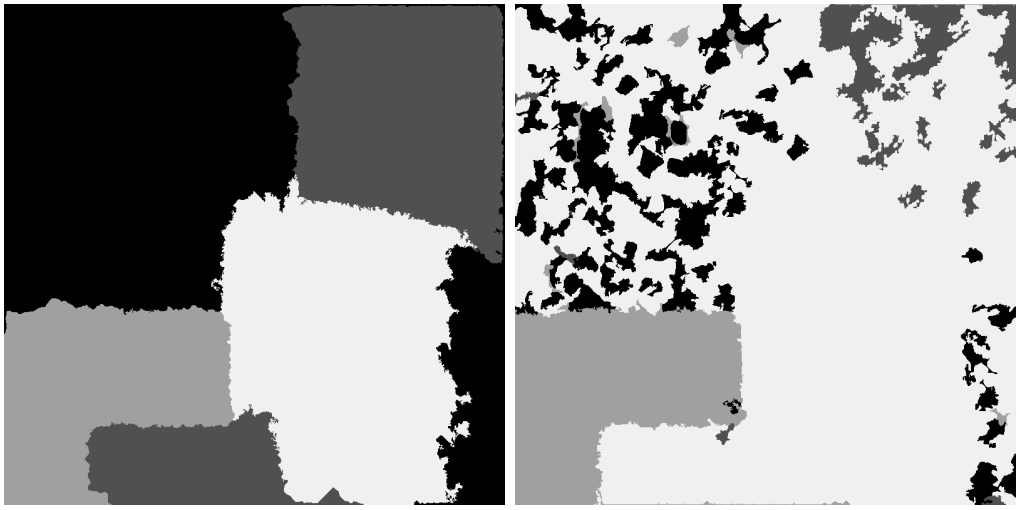


Abbildung 5.12: Rauschreduktion. Nachverarbeitung der Segmentierungsergebnisse aus Abbildung 5.11.

Die iMRF-Klassifikationsquote wird durch die Nachverarbeitung auf 97.7% erhöht. Die verbleibenden Fehlklassifikationen finden sich in den Übergangszonen zwischen den Texturen und resultieren aus dem Korona-Effekt der regionenbasierten Segmentierung. Eine weitere Verbesserung ist nur über anwendungsspezifische Modellannahmen für Kantenverläufe erzielbar. So haben z.B. Grenzen zwischen zwei Ackerflächen in einem Luftbild im allgemeinen einen geraden Verlauf. Unter Ausnutzung dieses Vorwissens kann eine kantenbasierte Segmentierung eine optimierte Trennung der Übergangszonen erzielen. Das eMRF-Segmentierungsergebnis kann durch eine Rauschreduktion nur unwesentlich verbessert werden, siehe Abbildung 5.12 rechts. Hier steigt die Klassifikationsquote auf

maximal 54%, da das Modell einer hochfrequenten Störung die beobachteten Segmentierungsfehler nicht unterstützt.

Textursegmentierung mit Auflösungspyramide

Die Anwendung einer Auflösungspyramide zur Vergrößerung des Nachbarschaftsfensters mit Breite W_N auf ein effektives Maß $W_N^{eff} = W_N \cdot 2^{L-1}$ wurde für die Textursynthese in Abschnitt 3.5.1 beschrieben. Dieser multiskalige Ansatz kann auch auf die Textursegmentierung übertragen werden.

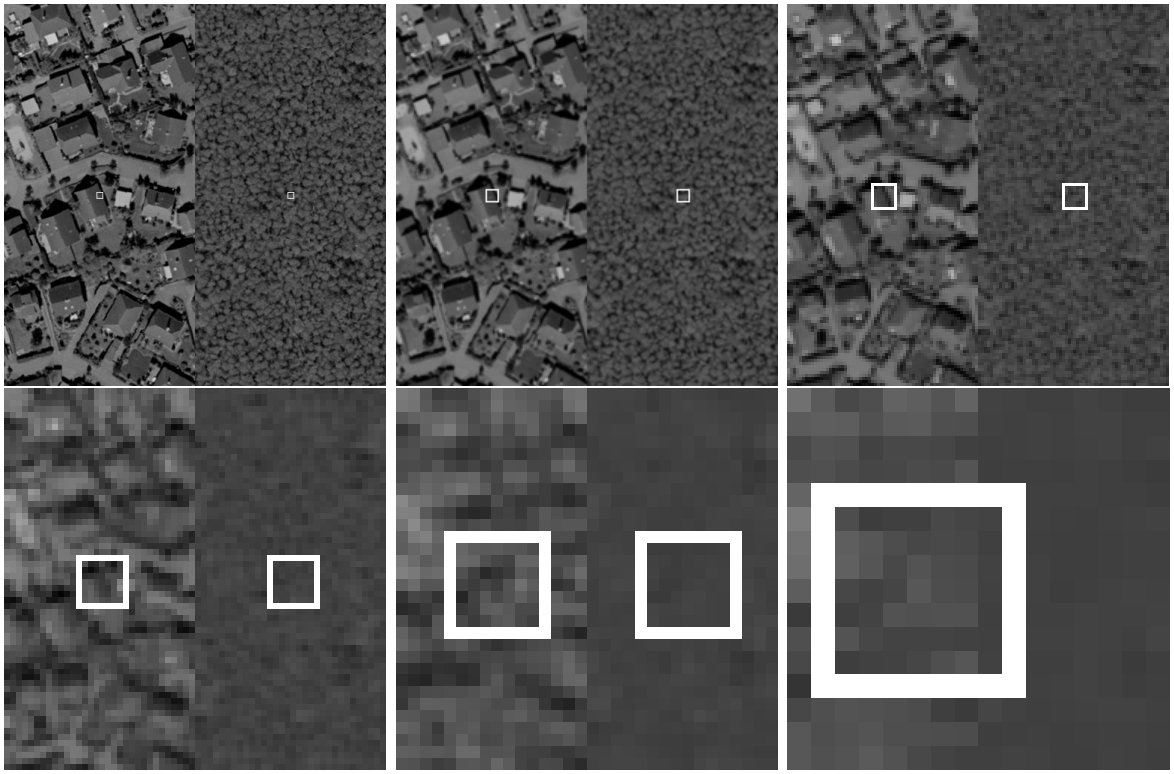


Abbildung 5.13: Auflösungspyramide und Reichweite eines 9×9-Nachbarschaftsfenster.

Abbildung 5.13 zeigt Luftbildauschnitte von einem Siedlungs- und einem Waldgebiet zusammen mit dem Erfassungsbereich eines 9×9-Analysefensters. In der höchsten Auflösungsstufe werden keine Merkmale erfaßt, die für die Textur „Siedlung“ spezifisch sind. Daher wird ein Segmentierungsergebnis dieser Auflösungsstufe stark fehlerbehaftet sein. Erst in den niedrigen Auflösungen beinhaltet die Umgebung globale, strukturelle Merkmale der Textur, die für eine Klassifikation geeignet sind. Umgekehrt lassen sich homogenere Texturen, wie das Waldgebiet, besser in den mittleren Auflösungsstufen klassifizieren, weil in niedrigen Stufen nur noch grauwert-homogene Flächen vorhanden sind.

Aufgrund der Abhängigkeit der Klassifikationsquote von der Auflösungsstufe ist eine direkte Kopplung der Stufen, z.B. über eine erweiterte Nachbarschaftsumgebung, siehe Textursynthese Abschnitt 3.3.2, nicht sinnvoll. Die Segmentierungen der einzelnen Auflösungsstufen l werden daher separat ausgeführt, die Teilergebnisse bewertet und unter Berücksichtigung der auflösungsspezifischen Klassifikationsgüte fusioniert. Die Messung der Klassifikationsgüte erfolgt anhand des Anlernbildes, das eine ideale Segmentierung darstellt. Dazu wird jede Anlerntextur in zwei Hälften unterteilt und eine Testsegmentierung jeweils einer Hälfte mit den Referenzdaten der anderen Hälfte und aller anderen Anlerntexturen durchgeführt. Diese Test- bzw. „Selbstsegmentierung“ wird für beide Hälften aller Anlerntexturen und alle L Auflösungsstufen wiederholt. Als Ergebnis erhält man für jede der Texturklassen eine Tabelle, welche die Klassifikationsgüte in Abhängigkeit von l wiedergibt. Für die Anlerntextur „Siedlung“ der Abbildung 5.13 zeigt Tabelle 5.1 das Ergebnis der Testsegmentierung mit iMRF-Verfahren.

Auflösungsstufe l	klassifizierte Bildpunkte		Quote
	richtig	falsch	
0	163577	92455	0.64
1	40355	22141	0.65
2	9665	5215	0.65
3	3360	729	0.82
4	593	79	0.88

Tabelle 5.1: Ergebnis der Selbstsegmentierung für die Anlerntextur „Siedlung“.

Wie nach Abbildung 5.13 zu erwarten war, ist die Klassifikation der Siedlungstextur in niedrigeren Auflösungsstufen zuverlässiger. Anhand der Tabellendaten aller Anlerntexturen lassen sich normierte Gewichtungsfaktoren w_l^k berechnen, welche die Segmentierungsergebnisse der verschiedenen Auflösungsstufen, entsprechend der Klassifikationsgüte, stärker oder schwächer gewichten [22]. Prinzipiell genügt es, ausschließlich die Segmentierungsergebnisse der texturspezifischen, optimalen Auflösungsstufe zu verwenden. Variiert diese optimale Stufe aber stark für die verschiedenen, zu trennenden Texturklassen, führt dies zu unterschiedlichen Quantisierungen der Regionsberandungen. Um das zu vermeiden, werden alle Teilergebnisse der einzelnen Auflösungsstufen durch Überabtastung und Interpolation auf die Größe der Auflösungsstufe $l = 0$ erweitert. Anschließend wird das multiskalige Segmentierungsergebnis berechnet, indem für jeden Bildpunkt die Klassenzuordnung gewählt wird, die in allen Auflösungsstufen am häufigsten auftritt. Für die Berechnung der Häufigkeit wird die auflösungsabhängige Zuverlässigkeit der Klassenzuordnung über die Gewichtungsfaktoren w_l^k berücksichtigt.

In Abbildung 5.14 sind die multiskaligen Segmentierungsergebnisse der beiden Verfahren dargestellt. Die Klassifikationsquote des neu entwickelten iMRF-Verfahrens wird von ursprünglich 72.1% auf 80.5% erhöht. Sehr viel deutlicher profitiert das Vergleichsverfahren von der Bildpyramide, das eine Verbesserung der Klassifikationsquote von 51.5% auf 86.7% erzielt. Wird zusätzlich die oben beschriebene Rauschreduktion durch Zuordnung

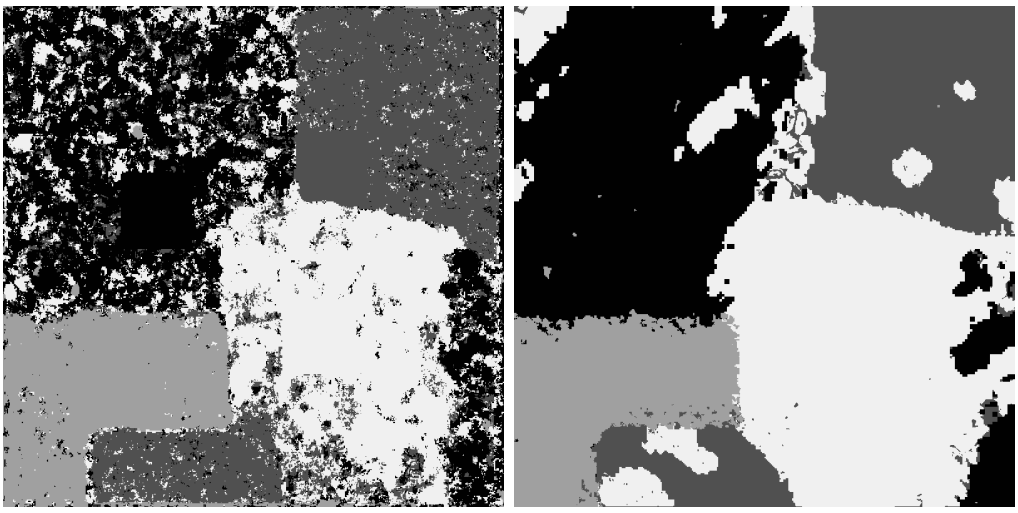


Abbildung 5.14: Multiskalige Segmentierung. Links: iMRF-Verfahren, rechts: eMRF-Vergleichsalgorithmus.

der hochfrequenten Signalanteile zum Hintergrund durchgeführt, erhöht sich die Klassifikationsquote auf 95.8% für iMRF bzw. 87.7% für eMRF. Der kleinere Wert von 95.8% verglichen mit 97.7% bei Segmentierung einer Auflösungsstufe und Rauschreduktion erklärt sich durch den ungenaueren Kantenverlauf in den niedrigeren Auflösungsstufen der Bildpyramide.

Das explizite Modell des eMRF-Vergleichsverfahrens berücksichtigt nur die Statistik 1. und 2. Ordnung, siehe Abschnitt 2.4. Das entwickelte iMRF-Verfahren mit dem impliziten Markov-Zufallsfeld Texturmodell erfaßt durch den direkten Nachbarschaftsvergleich auch höhere statistische Momente. Abbildung 5.15 demonstriert dies anhand einer

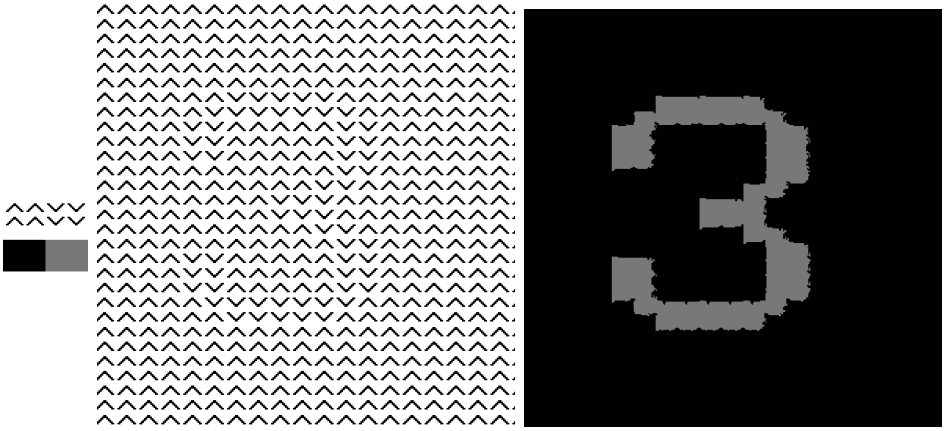


Abbildung 5.15: Statistik 3. Ordnung. Anlernbild (aus [103]) und Klassenzuordnung, Eingangsdaten, iMRF-Segmentierung.

Textur mit unterschiedlich orientierten Winkelelementen. eMRF modelliert nur Punkt-zu-Punkt-Abhängigkeiten und kann daher den Zusammenhang, der aus der Verbindung zweier Diagonallinien resultiert, nicht wiedergeben. Dementsprechend werden die in dem Eingangsbild unterschiedlich angeordneten Winkelelemente nicht segmentiert und alle Punkte des Bildes derselben Klasse zugeordnet. Dem implizit vollständigen iMRF-Modell gelingt die Klassifikation. Somit können auch Eingangsdaten verarbeitet werden, die der menschliche Sehapparat aufgrund seiner Unempfindlichkeit für höhere statistische Momente nur schwerlich oder gar nicht segmentieren kann.

5.2.4 Anwendung auf Fernerkundungsdaten

Für den Test und Vergleich von Segmentierungsverfahren werden in der Literatur meistens manuell erzeugte Eingangsdaten mit bekannter Referenzsegmentierung verwendet, so auch im vorangegangenen Abschnitt. Auf diese Weise läßt sich die Effizienz der Klassentrennung objektiv bewerten. Im praktischen Einsatz sind die Anforderungen an ein Segmentierungsverfahren ungleich höher. Im Gegensatz zu den synthetischen Texturcollagen lassen sich hier im allgemeinen keine repräsentativen Anlerntexturen zusammenstellen. Dies gilt vor allem für die Segmentierung von Fernerkundungsdaten, die eine Vielzahl von Objekten wiedergeben und deren Komplexität somit deutlicher höher ist. Abbildung 5.16 zeigt vier Anlerntexturen „Ackerland“, „Wald“, „Siedlung“ und „Industrie“, die dem Orthophoto in Abbildung 5.18 links oben entnommen wurden.

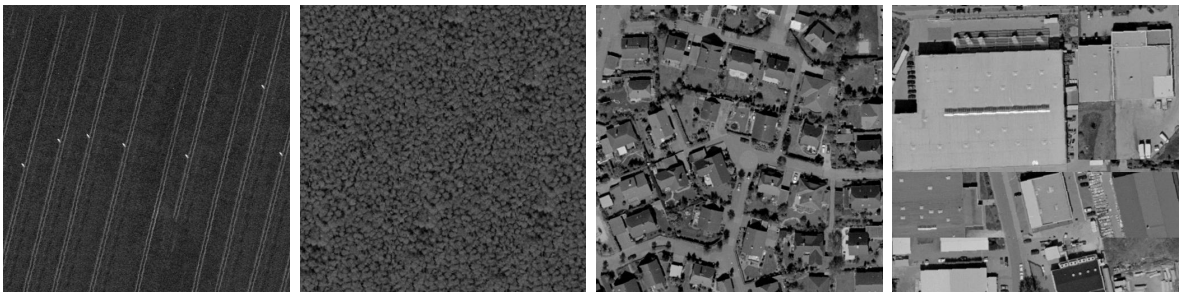


Abbildung 5.16: Anlerntexturen Ackerland, Wald, Siedlung, Industrie.

Diese Anlerntexturen werden zur multiskaligen Segmentierung der vollständigen Luftbildszene herangezogen und mit der manuell erstellten Referenzsegmentierung aus Abbildung 5.18 oben rechts verglichen. Wie oben beschrieben, wird zuerst eine Selbstsegmentierung der Anlerntexturen durchgeführt, um für jede Texturklasse normierte Gewichtungsfaktoren w_l^k zur Berücksichtigung auflösungsabhängiger, unterschiedlicher Klassifikationsgüten zu berechnen, siehe Abbildung 5.17. Die Klassifikationsquoten der multiskaligen Segmentierung sind für beide Verfahren in Tabelle 5.2 dargestellt. Insgesamt ist das Segmentierungsergebnis mit iMRF um circa 5% besser als das von eMRF. Die Ursache für den vergleichsweise geringen Vorteil von iMRF liegt in der Fehlklassifikation von Wald- und Ackerland-Texturen. So wurden die großflächigen Felder, die in

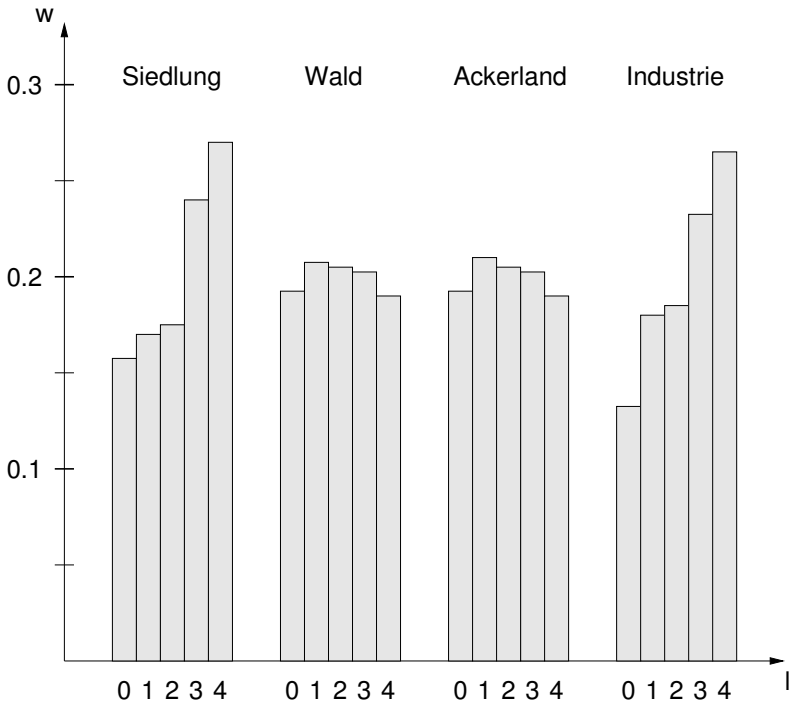


Abbildung 5.17: Normierte Gewichtungsfaktoren w_l^k für die Texturklassen aus Abbildung 5.16 und fünf Auflösungsstufen l.

Abbildung 5.18 mit einem „F“ versehen sind, fälschlicherweise als Wald- bzw. Industriegebiet klassifiziert. Diese Einteilung resultiert aus nicht repräsentativen Anlerntexturen. So entspricht die Grauwertverteilung des hellen, unteren Felds eher der eines Industriegebietes. Während die Anlerntextur für „Ackerland“ parallele Strukturen aufweist, ist das obere fehlklassifizierte Feld hingegen homogen und der Anlerntextur „Wald“ viel ähnlicher. Dagegen ist die Klassifikation der komplexen Strukturen „Siedlung“ und „Industrie“ mit dem neu entwickelten iMRF-Verfahren deutlich zuverlässiger als mit dem eMRF-Vergleichsverfahren.

	Siedlung	Wald	Ackerland	Industrie	gesamt
iMRF	89.0	45.5	53.8	63.6	70.5
eMRF	57.2	73.9	62.3	21.0	65.6

Tabelle 5.2: Klassifikationsquoten bei Segmentierung von Fernerkundungsdaten.

Zusammenfassend läßt sich sagen, daß das entwickelte multiskalige Segmentierungsverfahren mit implizitem Texturmodell über Markov-Zufallsfelder bei beschränktem Umfang der zu unterscheidenden Texturklassen und repräsentativer Auswahl von Anlerntexturen hohe Klassifikationsquoten von über 90% erzielen kann. Dabei erreicht es auch ohne multiskalige Analyse eine um ca. 10% höhere Klassifikationsquote als die bisher verwendete Textursegmentierung mit explizitem Modell. Komplexere Aufgaben wie die

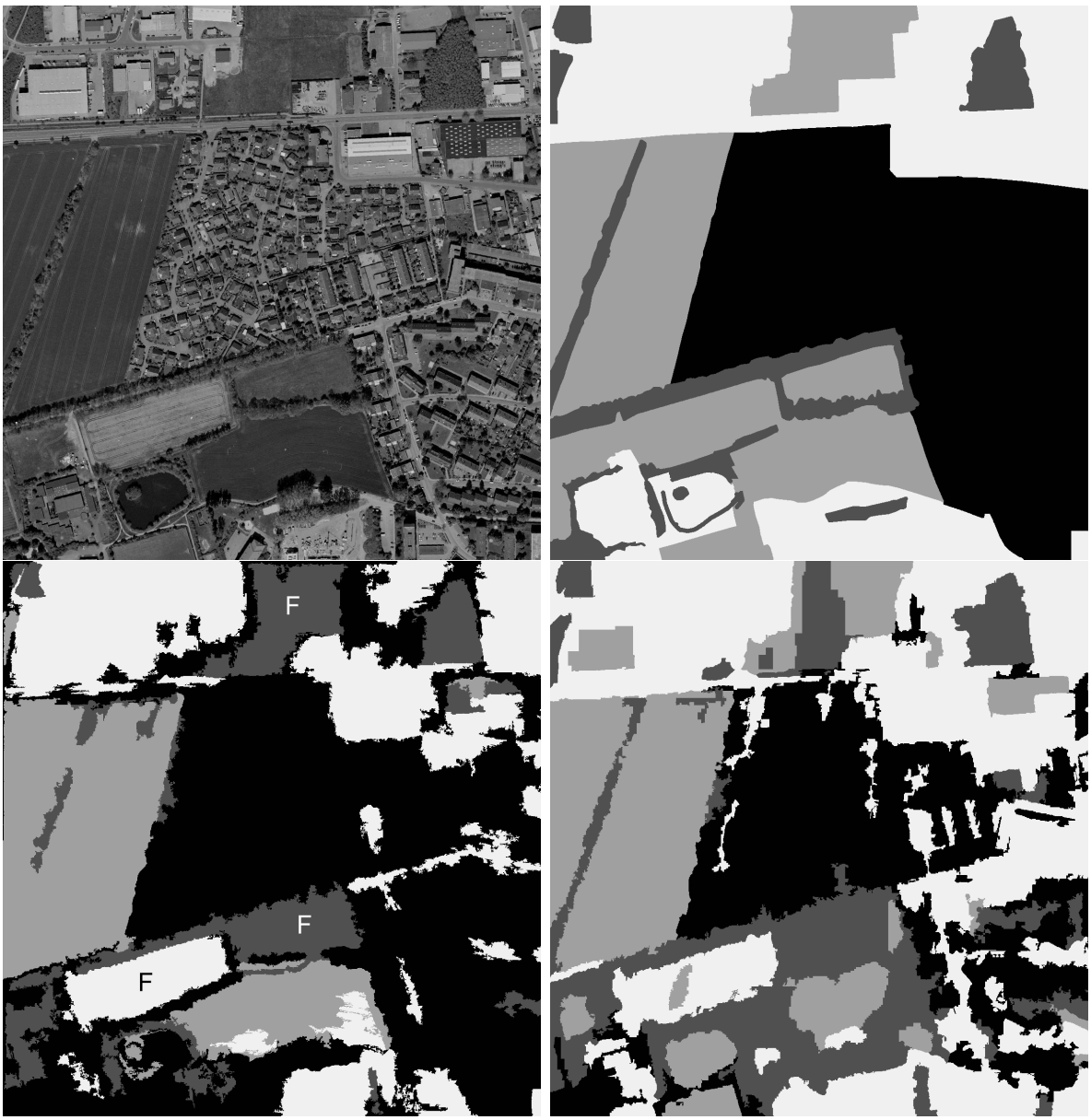


Abbildung 5.18: Luftbildsegmentierung. Orthophoto und manuell erstellte Referenzsegmentierung, Ergebnisse iMRF und eMRF.

Segmentierung von Luftbildern, bei denen die Auswahl von Anlerntexturen aufgrund der Vielzahl unterschiedlicher Objektarten pauschaler erfolgen muß, werden von der entwickelten iMRF-Segmentierung zufriedenstellend gelöst. Auch dabei wird eine höhere Klassifikationsquote als mit dem Vergleichsverfahren erzielt, wobei insbesondere komplexe Texturen wie Siedlungs- und Industriegebiete deutlich genauer separiert werden.

6 Zusammenfassung und Ausblick

Inhalt der vorliegenden Arbeit ist die Entwicklung eines Verfahrens für die multiskalige Synthese und Analyse von Bildtexturen mit speziellen Anwendungen im Bereich der Kartographie und Fernerkundung. Der entwickelte Algorithmus kann ausgehend von einem Bildauschnitt, der eine beliebige, natürliche Textur darstellt, eine visuell ähnliche Textur jeder vorgebenen Größe erzeugen. Die Synthese erfolgt dabei über ein Texturmodell, das implizit durch die Texturprobe gebildet wird, ohne Kenntnis des eigentlichen Texturentstehungsprozesses. Die angestrebte, praktische Anwendung der Textursynthese liegt in der Erzeugung künstlicher Luftbilder für mobile Navigationsaufgaben. Gleichzeitig läßt sich das zugrundeliegende Texturmodell für die überwachte Segmentierung und Klassifikation von Bildtexturen in Fernerkundungsdaten einsetzen und stellt damit eine Basistechnologie für Umweltüberwachungsaufgaben oder eine automatische Aktualisierung von Geoinformationssystemen.

Bei der Verfahrensentwicklung steht die Aufgabe der Textursynthese im Vordergrund. Der Algorithmus nutzt ein statistisches Markov-Zufallsfeld-Texturmodell, das geeignet ist, sowohl regelmäßige als auch stochastische Texturen sowie jede Mischform aus diesen zu beschreiben. Der eigentliche Syntheseprozeß besteht aus einer Vielzahl von bildhaften Vergleichen, bei denen die Ähnlichkeit der lokalen Umgebung einer Syntheseposition zu lokalen Umgebungen in der Texturprobe gemessen wird. Aufgrund der Bildähnlichkeit der Nachbarschaften findet der Algorithmus geeignete Texturinformation, die von der Texturprobe auf die Syntheseposition übertragen wird. Markov-Zufallsfelder behandeln eine Textur als lokal und stationär und erfassen prinzipbedingt nur die Textureigenschaften, die von der betrachteten Nachbarschaftsumgebung umschlossen werden. Die hier eingesetzte Kombination von Markov-Zufallsfeldern und Auflösungspyramiden erweitert die effektive Nachbarschaftsgröße und ermöglicht so die gleichwertige Verarbeitung aller auftretenden Strukturgrößen. Als Mischform bisheriger Syntheseverfahren überträgt der Algorithmus kleine zusammenhängende Blöcke von der Texturprobe auf das Syntheseprodukt und erhält damit den lokalen Texturzusammenhang bei gleichzeitiger Steigerung der Ausführungsgeschwindigkeit. Bedingt durch den Blocktransfer können lokale Diskontinuitäten entstehen, die aber abhängig von der betrachteten Textur und Blockgröße unter die Wahrnehmungsschwelle fallen. Für die Synthese sind Randbedingungen zulässig, d.h. die Texturprobe selbst oder aber auch eine andere vorgegebene Textur kann Teil des synthetisierten Produktes sein. Durch torodiale Anschlußbedingungen der Synthesefläche werden kachelbare Ergebnisse erzeugt, die mit stetigen Übergängen an sich selbst angeschlossen werden können. Fehlentwicklungen des Aufbaus der erzeugten Textur werden durch Auswertung der Nutzungsstatistik der Texturprobe bereits während

der Laufzeit erkannt und kompensiert. Der zeitkritische Kern des Verfahrens ist der Bildvergleich der Nachbarschaftsumgebungen, welcher Ansatzpunkt für ein mehrschichtiges Beschleunigungskonzept ist. Jede Nachbarschaftsumgebung wird zunächst in einen hochdimensionalen Vektor überführt. Ein Bildvergleich entspricht dann einer Abstandsmessung zwischen Vektoren, die durch Parallelverarbeitung mit speziellen Hardwarebefehlen effizient ausgeführt wird. Die Vielzahl der Bildvergleiche führt auf eine Suche des nächsten Nachbarn im Nachbarschaftsvektorraum. Dieser Suchprozeß wird durch gleichzeitige Nutzung mehrerer Prozessoren und Vektorquantisierung beschleunigt. Beinhaltet die Texturprobe Farbinformation, wird diese getrennt verarbeitet und in einem Nachverarbeitungsschritt in das Syntheseprodukt eingebracht. Durch diese Verfahrensweise sinkt die Komplexität der Berechnungen, was sich wiederum positiv auf die Syntheserate auswirkt.

Anhand von Beispielen mit natürlichen Farbtexturen wurde die Qualität der Vollsynthese und Synthese mit Randbedingungen demonstriert. Das Verfahren erreicht aufgrund der Auflösungspyramide und der Auswertung der Nutzungsstatistik zur Laufzeit eine im Vergleich zu vorherigen Ansätzen höhere Synthesequalität und ermöglicht durch die hohen Syntheseraten die Verarbeitung großer Texturen für spezielle Anwendungen. Als erste, neue Anwendung wurde die Erzeugung künstlicher Luftbilder für die Kartographie gezeigt. Die synthetischen Luftbilder ersetzen Teile der Kartensymbolik durch dem Menschen vertrautere, natürliche Texturen. Sie stellen damit eine unterstützende, neue Visualisierungsform dar, die es erlaubt, die Erläuterung des Kartenmaterials auf wenige, anwendungsspezifische Elemente zu reduzieren und damit einen schnelleren Zugang zur Information zu erhalten. Die zweite Anwendung setzt den entwickelten Algorithmus in leicht modifizierter Form als Operator für die Textursegmentierung und -klassifikation von Fernerkundungsdaten im Rahmen eines wissensbasierten Bildinterpretationssystems ein. Die Leistungsfähigkeit der multiskaligen Segmentierung wurde mit künstlichen Testdaten und einer realen Luftbildszene verifiziert. Im direkten Vergleich zum bisher verwendeten Markov-Zufallsfeld-Segmentierungsverfahren konnte eine deutliche Verbesserung, insbesondere bei komplexen Strukturen erzielt werden.

Geeignete Ansatzpunkte für Verbesserungen des Algorithmus wurden im Rahmen der Parameteroptimierung und des Ergebnis- bzw. Anwendungskapitels diskutiert. Im einzelnen sind dies die Bildvergleichsmetrik, die durch genauere Betrachtung von Kantenverläufen z.B. über Auswertung der Hamming-Distanz [10] in Laplace-Auflösungspyramiden verbessert werden kann, komplexere Bewertungskriterien für die Nutzungsstatistik oder die Optimierung der Anschlüsse der synthetisierten Texturblöcke. Die gleichzeitige Verarbeitung von Struktur und Textur kann insbesondere bei der Synthese mit Randbedingungen Vorteile erzielen [6]. Das vorgestellte Verfahren ist nicht statisch, sondern bietet ein flexibeles Gerüst für künftige Entwicklungen und Anwendungen. Denkbar ist eine radiometrischen Entzerrung von Fernerkundungsdaten durch Textursynthese in Schattenbereichen oder eine überwachte Hintergrundsegmentierung über Schwellwertbildung. Eine bereits geplante Erweiterung ist die Texturierung dreidimensional beschriebener Oberflächen [102] [99] [97] [44] für Virtual Reality Anwendungen.

Literaturverzeichnis

[1] F. Ade, M. Peter, M. Rutishauser, M. Trobina und A. Ylä-Jääski. Vision for a 3-D Object Manipulation System. In: *Proceedings 14th DAGM Symposium* (Hg. S. Fuchs), S. 117–124. Springer Verlag, 1992.

[2] S. Albin, G. Rougeron, B. Peroche und A. Tremeau. Quality Image Metrics for Synthetic Images Based on Perceptual Color Differences. *IEEE Transactions on Image Processing*, 11(9):961–971, 2002.

[3] M. Ashikhmin. Synthesizing natural textures. In: *Proceedings of SIGGRAPH 2001* (Hg. E. Fiume), S. 217–226. ACM Press / ACM SIGGRAPH, 2001.

[4] J. J. Atick. Could information theory provide an ecological theory of sensory processing ? *Network: Computation in Neural Systems*, 3:213–251, 1992.

[5] J. R. Bergen und E. H. Adelson. Early vision and texture perception. *Nature*, 333:363–364, 1988.

[6] M. Bertalmio, G. S. L. Vese und S. Osher. Simultaneous Structure and Texture Image Inpainting. *IEEE Transactions on Image Processing*, 12(8):882–889, 2003.

[7] J. E. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society B*, 36:192–236, 1974.

[8] M. R. Bolin und G. W. Meyer. A Perceptually Based Adaptive Sampling Algorithm. In: *Siggraph 1998, Computer Graphics Proceedings*, S. 299–309. Addison Wesley Longman, 1998.

[9] J. S. D. Bonet. Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, S. 361–368. ACM Press/Addison-Wesley Publishing Co., 1997.

[10] A. Bookstein, V. A. Kulyukin und T. Raita. Generalized Hamming Distance. *Information Retrieval*, 5:353–375, 2002.

[11] C. Brenner und M. Sester. Continuous Generalization for Small Mobile Displays. In: *Int. Conference on Next Generation Geospatial Information, Boston MA*. 2003.

[12] P. Brodatz. *Textures: A Photographic Album for Artists and Designers*. Dover Publications, New York, 1966.

[13] P. A. Burrough und R. A. McDonnell. *Principles of Geographical Information Systems (Spatial Information Systems)*. Clarendon Press, 1998.

[14] P. J. Burt und E. H. Adelson. The Laplacian Pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31,4:532–540, 1983.

[15] J. Bückner, S. Müller, M. Pahl und O. Stahlhut. Semantic Interpretation of Remote Sensing Data. In: *Proceedings PCV02, ISPRS Commission III Symposium* (Hg. R. Kalliany, F. Leberl und F. Fraundorfer), Bd. XXXIV Teil 3A+B, S. 62–66. ISPRS Council Publications, 2002.

[16] J. Bückner, M. Pahl und O. Stahlhut. Radiometric Equalization of Remote Sensing Data by Utilization of Laserscan Data. In: *IEEE CCECE: Canadian Conference on Electrical and Computer Engineering, Toronto*, Bd. 2, S. 1111–1116. 2001.

[17] J. Bückner, M. Pahl, O. Stahlhut und C.-E. Liedtke. geoAIDA - A Knowledge Based Automatic Image Data Analyser for Remote Sensing Data. In: *Proceedings ICSC Congress CIMA, Bangor, Wales, UK*. NAISO Academic Press, 2001.

[18] J. Bückner, M. Pahl, O. Stahlhut und C.-E. Liedtke. A Knowledge-Based System for Context Dependent Evaluation of Remote Sensing Data. In: *Proceedings 24th DAGM Symposium* (Hg. L. V. Gool), S. 58–65. Springer-Verlag, Berlin Heidelberg, 2002.

[19] T. M. Caelli und B. Julesz. On perceptual analyzers underlying visual texture discrimination. Part I. *Biological Cybernetics*, 28:167–175, 1978.

[20] T. M. Caelli, B. Julesz und E. N. Gilbert. On perceptual analyzers underlying visual texture discrimination. Part II. *Biological Cybernetics*, 29:201–214, 1978.

[21] B. Claus, C. Daul und R. Rösch. Qualität von Holzoberflächen: Farbe und Maserung. In: *Proceedings 18th DAGM Symposium* (Hg. B. Jähne), S. 199–208. Springer Verlag, 1996.

[22] K. Cordes. *Übertragung eines Textursyntheseverfahrens auf die Textursegmentierung von Fernerkundungsdaten*. Diplomarbeit, Institut für Informatik, Universität Hannover, 2003.

[23] K. Cordes und O. Stahlhut. An efficient algorithm for high dimensional hypercube searching. *To be published*, 2004.

[24] H. Cramer und M. R. Leadbetter. *Stationary and Related Stochastic Processes*. John Wiley and Sons, New York, 1967.

[25] G. R. Cross und A. K. Jain. Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(1):25–39, 1983.

[26] P. H. Dana. Global Positioning System Overview. Techn. Ber., University of Colorado at Boulder, Department of Geography, 2000.
URL `http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html`

[27] J.-M. Dischler, K. Maritaud, B. Levy und D. Ghazanfarpour. Texture Particles. In: *Proceedings of the Eurographics conference 2002* (Hg. G. Drettakis und H.-P. Seidel), Bd. 21. Blackwell Publishers, 2002.

[28] J. Dorsey, A. Edelman, J. Legakis, H. W. Jensen und H. K. Pedersen. Modeling and Rendering of Weathered Stone. In: *Siggraph 1999, Computer Graphics Proceedings*, S. 225–234. Addison Wesley Longman, Los Angeles, 1999.

[29] J. Dorsey und P. Hanrahan. Modeling and Rendering of Metallic Patinas. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, S. 387–396. ACM Press, 1996.

[30] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin und S. Worley. *Texturing and modeling: a procedural approach*. Academic Press Professional, Inc., 1994.

[31] A. A. Efros und W. T. Freeman. Image quilting for texture synthesis and transfer. In: *Proceedings of SIGGRAPH 2001* (Hg. E. Fiume), S. 341–346. ACM Press / ACM SIGGRAPH, 2001.

[32] A. A. Efros und T. K. Leung. Texture Synthesis by Non-parametric Sampling. In: *ICCV (2)*, S. 1033–1038. 1999.

[33] I. M. Elfadel und R. W. Picard. Gibbs random fields, cooccurences and texture modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):24–37, 1994.

[34] A. Fournier, D. Fussel und L. Carpenter. Computer Rendering of Stochastic Models. *Communications of the ACM*, 25(6):371–384, 1982.

[35] J. M. Francos, A. Z. Meiri und B. Porat. A unified texture model based on a 2-d wold-like decomposition. *IEEE Transactions on Signal Processing*, 41:2665–2678, 1993.

[36] D. D. Garber. *Computational Models for Texture Analysis and Texture Synthesis*. Dissertation, University of Southern California, Image Processing Institute, 1981.

[37] D. Geman. Random fields and inverse problems in imaging. *Lecture Notes in Mathematics*, 1427:113–193, 1991.

[38] S. Geman und D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[39] A. Gersho und R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[40] I. A. Getting. The Global Positioning System. *IEEE Spectrum*, 30(12):36–47, 1993.

[41] G. L. Gimel'farb. Texture Modeling by Multiple Pairwise Pixel Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(11):1110–1114, 1996.

[42] G. L. Gimel'farb. *Image Textures and Gibbs Random Fields*. Kluwer Academic Publishers, 1999.

[43] R. C. Gonzalez und P. Wintz. *Digital Image Processing*. Addison-Wesley, 1987.

[44] G. Gorla, V. Interrante und G. Sapiro. Texture Synthesis for 3D Shape Representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):512–524, 2003.

[45] D. Grüreich. ATKIS - A Topographic Information System as a Basis for a GIS and Digital Cartography in West Germany. *Geologisches Jahrbuch*, A122:207–215, 1992.

[46] GSM Association. GSM Technology. Techn. Ber., GSM Association, 2003. URL `http://www.gsmworld.com/technology/index.shtml`

[47] G. Hake und D. Grünreich. *Kartographie*. de Gruyter, 1994.

[48] M. Hampe und M. Sester. Real-Time Integration and Generalization of Spatial Data for Mobile Applications. In: *Maps and the Internet 2002, Geowissenschaftliche Mitteilungen, Heft Nr.60*. Technische Universität Wien, 2002.

[49] R. M. Haralick. Statistical and Structural Approaches to Texture. *Proceedings of IEEE*, 67(5):786–804, 1979.

[50] R. M. Haralick, K. Shanmugan und I. Dinstein. Textural Features for image classification. *Proc. IEEE Transactions on Systems, Man and Cybernetics*, 3:610–621, 1973.

[51] R. M. Haralick und L. G. Shapiro. *Computer and Robot Vision*, Bd. 1. Addison-Wesley, MA, 1992.

[52] P. Harrison. A non-hierarchical procedure for re-synthesis of complex textures. In: *9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, S. 190–197. 2001.

[53] J. C. Hart, N. Carr, M. Kameya, S. A. Tibbitts und T. J. Coleman. Antialiased parameterized solid texturing simplified for consumer-level hardware implementation. In: *Proceedings of the 1999 Eurographics/SIGGRAPH workshop on Graphics hardware*, S. 45–53. ACM Press, 1999.

[54] D. J. Heeger und J. R. Bergen. Pyramid-based texture analysis/synthesis. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, S. 229–238. ACM Press, 1995.

[55] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless und D. H. Salesin. Image Analogies. In: *Proceedings of SIGGRAPH 2001* (Hg. E. Fiume), S. 327–340. ACM Press / ACM SIGGRAPH, 2001.

[56] T. Hofmann, J. Puzicha und J. M. Buhmann. Unsupervised Texture Segmentation in a Deterministic Annealing Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):803–818, 1998.

[57] HyperTransport Consortium. HyperTransport™ Technology. Techn. Ber., HyperTransport Consortium, 2003.
URL `http://www.hypertransport.org/technology.html`

[58] F. Jondral und A. Wiesler. *Wahrscheinlichkeitsrechnung und stochastische Prozesse*. Teubner, Stuttgart, 2002.

[59] B. Julesz. Visual pattern discrimination. *IRE Transactions on Information Theory*, 8(2):84–92, 1962.

[60] B. Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290:91–97, 1981.

[61] B. Julesz und J. R. Bergen. Textons: the fundamental elements in preattentive vision and perception of texture. *Bell Systems Technical Journal*, 62(6):1619–1645, 1983.

[62] B. Julesz, E. N. Gilbert und J. D. Victor. Visual discrimination of textures with identical third-order statistics. *Biological Cybernetics*, 31:137–140, 1978.

[63] B. Jähne. *Digitale Bildverarbeitung*. Springer Verlag, 2001.

[64] D. Kersten. Predictability and redundancy of natural images. *Journal of the Optical Society of America A*, 4(12):2395–2400, 1987.

[65] L. Kuhnert und O. Burgert. Texturbasierte Segmentierung. Techn. Ber., IAIM, Universität Karlsruhe, 2003.

[66] V. Lakshmanan, V. E. DeBrunner und R. Rabin. Nested Partitions Using Texture Segmentation. In: *5th IEEE Southwest Symposium on Image Analysis and Interpretation*. 2002.

[67] C. Lambrecht und J. Farrell. Perceptual quality metric for digitally coded color images. In: *Proceedings of the European Signal Processing Conference*, S. 1175–1178. 1996.

[68] K. Laws. *Textured Image Segmentation*. Dissertation, University of Southern California, 1980.

[69] J. P. Lewis. Generalized stochastic subdivision. *ACM Transactions on Graphics (TOG)*, 6(3):167–190, 1987.

[70] L. Liang, C. Liu, Y.-Q. Xu, B. Guo und H.-Y. Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG)*, 20(3):127–150, 2001.

[71] Y. Linde, A. Buzo und R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28:84–95, 1980.

[72] J. Malik und P. Perona. Preattentive Texture Discrimination with Early Vision Mechanisms. *Journal of the Optical Society of America A*, 7(5):929–931, 1990.

[73] M. Mittal, A. Peleg und U. Weiser. MMX™ Technology Architecture Overview. Techn. Ber., Intel Corporation, 1997.
URL `http://www.intel.com/technology/itj/q31997/pdf/archite.pdf`

[74] Motorola. AltiVec™ Technology at a glance. Techn. Ber., Motorola Inc., 2002.
URL `http://www.motorola.com/altivec`

[75] S. Müller, R. Feitosa, G. Mota, D. da Costa, V. da Silva und K. Tanisaki. GEOAIDA Applied to SPOT Satellite Image Interpretation. In: *Proceedings URBAN 2003 - 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*. 2003.

[76] S. Müller, M. Weis, C.-E. Liedtke und M. Pahl. Automatic Quality Surveillance of GIS Data with GEOAIDA. In: *Proceedings PIA 2003 - ISPRS Conference on Photogrammetric Image Analysis*. 2003.

[77] S. A. Nene und S. K. Nayar. A Simple Algorithm for Nearest Neighbor Search in High Dimensions. *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1997.

[78] M. Neubert und G. Meinel. Evaluation of segmentation programs for high resolution remote sensing applications. In: *Proceedings of the ISPRS Joint Workshop High Resolution Mapping from Space 2003*. 2003.

[79] R. Paget und D. Longstaff. Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random field. *IEEE Transactions on Image Processing*, 7(6):925–931, 1998.

[80] K. Perlin. An image synthesizer. In: *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, S. 287–296. ACM Press, 1985.

[81] R. Pickard, C. Graszyk, S. Mann, J. Wachmann, L. Pickard und L. Campbell. VisTex: Vision Texture Database. Techn. Ber., Massachusetts Institute of Technology Media Laboratory, Cambridge, MA, 1995.
URL `http://www-white.media.mit.edu/vismod/imagery/VisionTexture.`

[82] K. Popat und R. W. Picard. Novel cluster-based probability model for texture synthesis, classification and compression. In: *Proceedings SPIE Visual Communications and Image Processing*, S. 756–768. 1993.

[83] J. Portilla und E. Simoncelli. Representation and Synthesis of Visual Texture. Techn. Ber., Laboratory for Computational Vision, New York University, 2001.
URL `http://www.cns.nyu.edu/~eero/texture/`

[84] J. Portilla und E. P. Simoncelli. Texture modelling and synthesis using joint statistics of complex wavelet coefficients. In: *IEEE Workshop on Statistical and Computational Theories of Vision*. 1999.

[85] J. Portilla und E. P. Simoncelli. A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. *International Journal of Computer Vision*, 40(1):49–70, 2000.

[86] M. Rahnema. Overview of the GSM system and protocol architecture. *IEEE Communications Magazine*, 31(4):92–100, 1993.

[87] M. Ramasubramanian, S. N. Pattanaik und D. P. Greenberg. A Perceptually Based Physical Error Metric for Realistic Image Synthesis. In: *Siggraph 1999, Computer Graphics Proceedings* (Hg. A. Rockwood), S. 73–82. Addison Wesley Longman, 1999.

[88] D. L. Ruderman und W. Bialek. Statistics of natural images: Scaling in the woods. *Physical Review Letters*, 73(6):814–817, 1994.

[89] C. J. van der Sande, S. M. de Jong und A. P. J. de Roo. A segmentation and classification approach of IKONOS-2 imagery for land cover mapping to assist flood risk and flood damage assessment. *International Journal of Applied Earth Observation and Geoinformation*, 4(3):217–229, 2003.

[90] A. Schröder. *Bilddatenmanagement zur herzchirurgischen Operationsplanung: Bedeutung des Konzeptes der problemorientierten Segmentierung und Visualisierung in der hierarchischen Datenanalyse*. Dissertation, Medizinische Fakultät Heidelberg, 1999.

[91] M. Sester. Generalization Based on Least Squares Adjustment. *International Archives of Photogrammetry and Remote Sensing, Amsterdam, Netherlands*, XXXIII(Part B4):931–938, 2000.

[92] E. Simoncelli und J. Portilla. Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes. In: *5th IEEE International Conference on Image Processing*, Bd. I. IEEE Computer Society, Chicago, 4-7 1998.

[93] W. Skarbek und A. Koschan. Colour image segmentation - a survey. Techn. Ber., Technische Universität Berlin, 1994.

[94] L. Spracklen. The VIS Instruction Set. Techn. Ber., SUN Microsystems, Inc., VIS Engineering, Processor and Network Products, 2003.
URL `http://www.sun.com/processors/vis/`

[95] O. Stahlhut. Extending Natural Textures with Multi-scale Synthesis. In: *Proceedings of Vision, Video and Graphics 2003 (Eurographics PE)*, S. 221–230. Eurographics Association, 2003.

[96] S. Stan, G. Palubinskas und M. Datcu. Bayesian selection of the neighbourhood order for Gauss-Markov texture models. *Pattern Recognition Letters*, 23:1229–1238, 2002.

[97] X. Tong, J. Zhang, L. Liu, X. Wang, B. Guo und H.-Y. Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, S. 665–672. ACM Press, 2002.

[98] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. In: *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, S. 289–298. ACM Press, 1991.

[99] G. Turk. Texture synthesis on surfaces. In: *Proceedings of SIGGRAPH 2001* (Hg. E. Fiume), S. 347–354. ACM Press / ACM SIGGRAPH, 2001.

[100] L.-Y. Wei. *Texture Synthesis by Fixed Neighborhood Searching*. Dissertation, Stanford University, Department of Electrical Engineering, 2001.

[101] L.-Y. Wei und M. Levoy. Fast Texture Synthesis Using Tree-Structured Vector Quantization. In: *Siggraph 2000, Computer Graphics Proceedings*, S. 479–488. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[102] L.-Y. Wei und M. Levoy. Texture Synthesis Over Arbitrary Manifold Surfaces. In: *Proceedings of SIGGRAPH 2001* (Hg. E. Fiume), S. 355–360. ACM Press / ACM SIGGRAPH, 2001.

[103] D. Wermser. *Automatische Texturauswertung auf der Basis einer Analyse des visuellen Systems*. VDI-Verlag, 1985.

[104] A. Witkin und M. Kass. Reaction-diffusion textures. In: *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, S. 299–308. ACM Press, 1991.

[105] S. Worley. A cellular texture basis function. In: *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, S. 291–294. ACM Press, 1996.

[106] Y.-Q. Xu, B. Guo und H. Shum. Chaos Mosaic: Fast and Memory Efficient Texture Synthesis. Techn. Ber., Microsoft Research, 2000.
URL `http://research.microsoft.com/research/pubs/view.aspx?msr_tr_id=MSR-TR-2000-32`

[107] X. Yang und J. Liu. Unsupervised texture segmentation with one-step mean shift and boundary Markov random fields. *Pattern Recognition Letters*, 22:1073–1081, 2001.

[108] A. V. Zalesny. Homogeneity & texture. General approach. In: *Proceedings 12th IAPR International Conference on Pattern Recognition*, Bd. 1, S. 592–594. IEEE Computer Society Press, 1994.

[109] S. C. Zhu, Y. Wu und D. Mumford. Filters, Random Fields and Maximum Entropy FRAME): Towards a Unified Theory for Texture Modeling. *International Journal of Computer Vision*, 27(2):107–126, 1998.

Lebenslauf

Persönliche Daten

Name:	Oliver Stahlhut
geboren am:	14.08.1972 in Hildesheim
Staatsangehörigkeit:	deutsch
Familienstand:	verheiratet, 1 Sohn

Schulausbildung

1978 - 1984	Grundschule und Orientierungsstufe Elze
1984 - 1991	Gymnasium Sarstedt, Abschluß: Abitur

Hochschulstudium

1991 - 1997	Studium der Physik an der Universität Hannover Studienschwerpunkte: Experimentalphysik und Informatik Diplomarbeit am Institut für Quantenoptik: „Zeitaufgelöste Fouriertransformationsspektroskopie“ Abschluß: Diplom (Dipl.-Phys.)

Berufstätigkeit

1997 - 2003	Wissenschaftlicher Mitarbeiter an der Universität Hannover, Institut für Theoretische Nachrichtentechnik
seit Januar 2004	Mitarbeiter der Benecke-Kaliko AG, Hannover