Angewandte Inpainting Verfahren zum Auffüllen von Aufdeckungen bei der 3D-Stereosynthese

Diplomarbeit

zur Erlangung des akademischen Grades
Diplominformatiker(in)

Humboldt-Universität zu Berlin
Mathematisch-Naturwissenschaftliche Fakultät II
Institut für Informatik

eingereicht von:	Adrian Jäkel
geboren am:	27.04.1983
in:	Jena

Gutachter:	Prof. Eisert, HU-Berlin
	Dr. Sebastian Knorr, imcube labs GmbH

eingereicht am: ..... verteidigt am: .....

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe. Weiterhin erkläre ich, eine Diplomarbeit in diesem Studiengebiet erstmalig einzureichen.

Berlin, den 4. Mai 2012

.....

Adrian Jäkel

Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die diese Diplomarbeit überhaupt erst ermöglicht haben und mich mit Rat, Tat und Beistand durch diese nicht immer leichte Phase geleitet haben.

Mein besonderer und tiefer Dank gilt daher zuerst meiner Mutter - ohne deren Unterstüzung ich nie hätte studieren können und die mit Geduld über meine immer größer werdende Semesteranzahl hinwegsah. Ich danke Ihr auch herzlich für das unermüdliche Korrekturlesen meiner Diplomarbeit und Ihre aufbauenden Worte in den anstrengenden Zeiten zum Ende hin.

Ich möchte mich außerdem bei Herrn Professor Eisert und Herrn Doktor Knorr bedanken, die mich bei meiner Diplomarbeit unterstützt, mir mit wertvollen Hinweisen und Tipps zur Seite gestanden und als Gutachter meiner Arbeit fungiert haben. Mein Dank geht auch an Herrn Doktor Kunter, der mir durch sein Fachwissen und seine Anregungen als Betreuer weitergeholfen hat.

Weiterhin danke ich meiner Freundin Nadine Andreas, die sich bereit erklärt hat, meine Diplomarbeit gegenzulesen, sich nicht gescheut hat, Kritik zu üben und Verbesserungsvorschläge vorzubringen und mir auch moralisch immer zu Seite stand.

Abkürzungsverzeichnis

2D	2-dimensional
3D	3-dimensional
DIBR	Depth Image Based Rendering
LGS	Lineares Gleichungssystem
MRF	Markov Random Field
PDG	Partielle Differentialgleichung
ROI	Region of Interest

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

3D ist in aller Munde. Wie auch immer man zu dem von den Einen als neue Ära beschriebenen, von den Anderen als vergänglichem Hype belächelten Thema stehen mag - die Technik ist unübersehbar auf den Vormarsch. Ursprünglich wieder aufgelebt durch Kinofilme wie *Avatar*, ist 3D Dank neuester erschwinglicher 3D-Fernseher-Technik auch im Consumerbereich angekommen und ganze TV-Produktionen werden heutzutage schon in einer dreidimensionalen Variante gefertigt. Da ist es nicht verwunderlich, dass vielfach der Wunsch nach 3D-Adaptionen beliebter Film- und Fernsehklassiker entsteht. Die nötige 2D-3D-Konvertierung ist jedoch zeit- bzw. kostenintensiv und bietet damit Ansätze zu einem interessanten Forschungsfeld, welches sich damit beschäftigt, Algorithmen und Verfahren zu finden, die diesen Vorgang möglichst gut automatisieren und beschleunigen können.

Abbildung 1.1.: Bei der 3D-Stereosynthese wird aus der linken Ansicht samt Tiefe (links) eine rechte Ansicht synthetisiert. Dabei entstehen Aufdeckungen (mitte). Diese können durch Methoden wie dem Inpainting gefüllt werden (rechts).

Im Gegensatz zur reinen 3D-Aufnahmetechnik kommen bei der nachträglichen Umwandlung von 2D-Material - der *3D-Stereosynthese* - verschiedene Verfahren zum Einsatz, die der Originalansicht z.B. eine Tiefenkarte hinzufügen und damit eine zweite Ansicht generieren (siehe Abbildung 1.1). Techniken wie das “Depth Image

Based Rendering" (DIBR) errechnen mit den an die einzelnen Pixel gekoppelten Tiefeninformationen perspektivisch korrekte Verschiebungen der Bildinhalte (vgl. Fehn [29]). In der Kombination des Originalbildes mit der synthetisierten Ansicht entsteht somit eine für den Menschen glaubhafte 3D-Wahrnehmung. Diese Umwandlung muß sich verschiedenen Problemen widmen, unter anderem entstehen sogenannte Aufdeckungen. Ähnlich wie beim menschlichen Sehen sind aus verschiedenen Blickwinkeln Elemente der Szene verdeckt oder eben nicht. Zum Auffüllen dieser Bereiche kommt eine Technik zum Einsatz, die schon länger in der Wissenschaft bekannt ist und untersucht wird. Das *Inpainting* (zu Deutsch in etwa "auffüllen" oder "einzeichnen") beschreibt als Begriff den Prozess des Wiederherstellens, Auffüllens oder Ersetzens von beschädigten oder fehlenden Teilen sowohl digitaler Bilder als auch Videos. Ziel ist es, ein Bild so zu rekonstruieren, dass es für den Betrachter weiterhin natürlich bzw. real aussieht. Anwendungsgebiete ergeben sich reichlich, nicht nur im Bereich der 3D-Stereosynthese. Inpainting wird zum Beispiel auch für die Rekonstruktion alter Fotografien verwendet oder zum Entfernen von unerwünschten Objekten in Bildern (siehe Abbildung 1.2).

Abbildung 1.2.: Inpainting, zum Entfernen von Objekten verwendet. Der Elefant (links) wurde aus dem Bild entfernt (rechts). Bildquelle: Drori et al. [24]

Die Technik des Inpaintings, bis dato ausgiebig in der Anwendung bei Bildern untersucht, sieht sich bei Videos und damit auch bei 3D-Sequenzen neuen Herausforderungen gegenüber. Um zu gewährleisten, dass die Ergebnisse auch beim Auffüllen von Videosequenzen natürlich aussehen, muss hierbei eine wichtige zusätzliche Eigenschaft berücksichtigt werden - die sogenannte zeitliche Konsistenz. Schon kleinste Unregelmäßigkeiten zwischen den Videoframes werden vom Betrachter sofort negativ wahrgenommen. Dies zu verhindern ist eine herausfordernde Aufgabe und damit vielfach Gegenstand aktueller Forschung.

1.2. Aufgabenstellung

Ausgehend von diesen Betrachtungen ist es das Ziel dieser Arbeit, die Methodik des Inpainting vorzustellen und näher zu untersuchen. Dabei soll zum einen in das Thema eingeführt und die Anwendung gängiger Inpainting-Verfahren demonstriert werden. Zum anderen soll darauf aufbauend der Rahmen auf die angesprochenen aktuellen Problematiken, insbesondere des Inpaintings bei der 3D-Stereosynthese, erweitert werden. Spezialisierte Inpainting-Verfahren sollen vorgestellt und ihr Verhalten bezüglich der erreichten zeitlichen Konsistenz untersucht und abschließend bewertet werden. Im Zuge dieser Arbeit sind dazu zwei Verfahren ausgesucht und für die Untersuchung implementiert worden.

Zur Einführung der Inpainting-Problematik wurde ein neueres Verfahren von Shen et al. [59] gewählt. Dieses Verfahren basiert auf einer in der Forschung weit verbreiteten und anerkannten patchbasierten Methode (siehe Kapitel 3.1 zu texturbasierten Verfahren), setzt jedoch neue Akzente durch eine Einbeziehung von Gradienten und der Rekonstruktion eines Bildes durch Lösung einer zugehörigen Poisson-Gleichung. Anhand dieses Verfahrens bietet sich die Möglichkeit, den Leser in das Thema einzuführen und eine der grundlegenden Techniken beim Inpainting darzustellen und zu erläutern. Die Tauglichkeit bei der 3D-Stereosynthese wird abschließend ebenfalls untersucht und spannt so den Bogen zum zweiten Verfahren.

Für die spezielle Anwendung bei Aufdeckungen in 3D-Synthese-Sequenzen wurde ein Verfahren von Cheng et. al [17] ausgewählt. Die Autoren dieser Veröffentlichung schlagen einen neuen Algorithmus zur Erzeugung von multiplen virtuellen Ansichten für autostereoskopische Displays vor. Ein wichtiger Bestandteil laut Veröffentlichung ist dabei die realistische und zeitkonsistente Wiederherstellung von Aufdeckungen, die bei der Generierung von einem für die dreidimensionale Darstellung nötigen zweiten Bild entstehen. Die vorgeschlagene Methode für eine natürliche und korrekte Auffüllung ist eine von mehreren aktuell in der Forschung diskutierten Ansätzen zum Inpainting in Videos. Aufgrund des ebenfalls in seiner Basis patchbasierten Verfahrens lässt sich seine Funktionsweise gut im Vergleich zu Verfahren gleichen Typs erschließen. In dieser Arbeit soll der grundlegende Algorithmus extrahiert sowie seine Tauglichkeit in der Anwendung auf konkrete 3D-Synthese-Sequenzen untersucht werden.

1.3. Aufbau der Arbeit

- **Kapitel 2** erklärt die grundlegende Notation des Inpaintings und die nötigen theoretischen Grundlagen zum Verständnis der bisherigen wissenschaftlichen Forschung und insbesondere von wichtigen Aspekten der zwei in dieser Arbeit vorgestellten Verfahren.
- **Kapitel 3** gibt einen Überblick über die bisherigen Arbeiten zum Thema und zum aktuellen Stand der Forschung. Beginnend mit den ersten Ansätzen des Einzelbild-Inpaintings wird der Verlauf der Forschung skizziert und die Entwicklung der verschiedenen Ansätze erläutert, um dann zum Schluss auf die Besonderheiten und unterschiedlichen Entwicklungen beim Video-Inpainting einzugehen.
- **Kapitel 4** geht detailliert auf die theoretische Funktionsweise und praktische Implementierung der ausgesuchten Verfahren ein, wobei jedem der zwei ein extra Abschnitt gewidmet wird. Zu Beginn demonstriert das erste Verfahren eine der grundlegenden Methoden des Inpaintings, während der zweite Abschnitt das spezielle zweite Verfahren mit dem Ziel der zeitkonsistenten Auffüllung von 3D-Synthese-Sequenzen vorstellt. Am Ende werden Besonderheiten, die im Zuge der Implementierung auftraten, erläutert.
- **Kapitel 5** erörtert die erzielten Resultate der implementierten Verfahren. Neben der Anwendung und Bewertung des ersten Verfahrens auf aus der Literatur bekannten Einzelbildern sowie dem Vergleich mit anderen Inpaintingverfahren, wird auch die Tauglichkeit bei den speziellen Aufdeckungen der 3D-Stereosynthese angesprochen. Im anschließend diskutierten zweiten Verfahren soll anhand der Anwendung auf konkrete 3D-Stereo-Sequenzen untersucht werden, inwiefern die gewünschte Zeitkonsistenz erreicht wurde.
- **Kapitel 6** fasst die Ergebnisse der Arbeit zusammen und gibt Ausblick auf mögliche Verbesserungen und Erweiterungen.

2. Theoretische Grundlagen

In diesem Kapitel sollen die notwendigen mathematischen und algorithmischen Grundlagen erläutert werden, die zum Verständnis der folgenden Kapitel benötigt werden. Nach der Festlegung der Notation der beim Inpainting üblichen Parameter folgt eine Beschreibung der für die Algorithmen benötigten mathematischen Sachverhalte. Dazu gehört neben den Begriffen des Gradienten und der Divergenz auch eine Erläuterung und Einordnung der Poisson-Gleichung.

2.1. Inpainting Notation

Die übliche Notation der Inpainting Literatur wird in dieser Arbeit übernommen (unterstützend dargestellt in Abbildung 2.1) und soll daher kurz erläutert werden. Im gegebenen Bild I definiert Ω den unbekannten, aufzufüllenden Bereich, sei es ein zu ersetzendes Objekt in der Bildbearbeitung oder eine während der 3D-Stereosynthese entstandene Aufdeckung. Weiterhin wird mit ψ_t der Zielpatch beschrieben, der aufgefüllt werden soll und der auf der Kontur $\delta\Omega$ des unbekannten Bereichs liegt. Der Patch ist zentriert auf den Pixel p, welcher die höchste zu ermittelnde Auffüllpriorität besitzt. Gesucht sind Quellpatches ψ_s im bekannten Bildbereich Φ, die ψ_t nach einem zu definierenden Maß möglichst ähnlich sind.

Abbildung 2.1.: Inpainting Notationen - Beschreibung siehe Text.

2.2. Gradienten und ihre Bedeutung in der Bildverarbeitung

Als Gradienten bezeichnet man im allgemeinen geometrischen Sinne den Anstieg an einem bestimmten Punkt. Für eine eindimensionale Funktion $f(x)$ wird so z.B. die Steigung an einem beliebigen Punkt x dargestellt und über die 1. Ableitung der

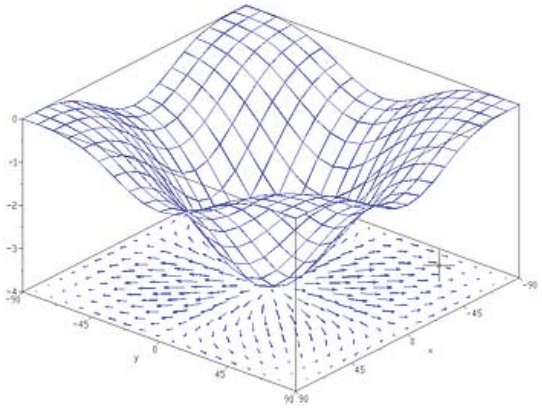


Abbildung 2.2.: Der Gradient der Funktion f(x,y) = -($\cos^2$x+$\cos^2$y)^2 visualisiert als projiziertes Vektorfeld auf der unteren Fläche

Funktion errechnet. Für eine zweidimensionale Funktion $f(x, y)$ erhält man einen Vektor, der in die Richtung des größten Anstiegs weist und dessen Betrag ein Maß für die Stärke desselben darstellt. (vgl. Abbildung 2.2)

Mathematisch gesehen ist der Gradient ein Differentialoperator, der auf ein Skalarfeld angewandt wird und ein Gradientenfeld genanntes Vektorfeld liefert. Er wird geschrieben als $\nabla f = grad(f)$, wobei ∇ den Nabla-Operator darstellt, definiert als Vektor, dessen Komponenten die partiellen Ableitungen $\frac{\partial f}{\partial x_i}$ von f in Richtung x_i sind.

Betrachtet man weiterhin Bilder als diskretisierte zweidimensionale Funktion, wobei jeder Bildpixel dem Wert der Funktion an der Stelle $f(x, y)$ entspricht, so errechnet sich der Gradient über die partiellen Ableitungen in x- und y-Richtung, welche dank der Diskretisierung auf das Pixelraster nun durch finite Differenzen[1] numerisch angenähert werden können. Zusammengefasst lassen sich folgende Sachverhalte formulieren:

- Gradient der Bildfunktion f

$$\nabla f = \left(\frac{\partial f}{\partial x}\ \frac{\partial f}{\partial y}\right) \tag{2.1}$$

- Betrag des Gradienten als Maß für den Anstieg

$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.2}$$

[1]Teil der einschlägigen Lehr-Literatur, z.B. W. Arendt und K.Urban [1] Kap. 9

- Richtung des Anstiegs

$$\Phi = \arctan\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right) \tag{2.3}$$

- Approximation der Ableitung durch finite Differenzen

$$\begin{aligned}\frac{\partial f}{\partial x} &\approx \frac{f(x+1,y) - f(x-1,y)}{2} \\ \frac{\partial f}{\partial y} &\approx \frac{f(x,y+1) - f(x,y-1)}{2}\end{aligned} \tag{2.4}$$

In der Bildverarbeitung stellen Gradienten also dar, wie schnell und in welche Richtung sich ein Bild verändert. Genutzt wird dies z.B. zur Strukturerkennung von Kanten oder Ecken in Bildern, denn an deren Position treten meist starke Änderungen in der Bildintensität auf. Im Besonderen lässt sich also feststellen, dass der Gradient eines Bildes auch ein Maß für dessen Struktur ist.

2.3. Poisson-Gleichung und ihre Anwendung

Theorie Die Poisson-Gleichung ist eine partielle Differentialgleichung zweiter Ordnung. Sie spielt bei der Modellierung von physikalischen und technischen Fragestellungen sowie der Lösung von Randwertproblemen eine große Rolle (z.B. in der Elektrostatik, der Akkustik oder der Thermodynamik). Ihre allgemeine Form lautet

$$\triangle u = f. \tag{2.5}$$

Dabei bezeichnet $\triangle$ den Laplace-Operator, f eine Funktion und u die gesuchte Lösung. Der Laplace-Operator ist im n-dimensionalen euklidischen Raum definiert als

$$\triangle u = div\,(grad\, u) = \nabla \cdot \nabla u = \nabla^2 u, \tag{2.6}$$

welcher die Summe der nicht gemischten partiellen Ableitungen zweiter Ordnung repräsentiert:

$$\triangle = \sum_{k=1}^{n} \frac{\partial^2}{\partial x_k^2}. \tag{2.7}$$

Wie zu sehen ist, gibt es hier eine Verbindung zum oben genannten Gradienten, erweitert um die Divergenz. Wenn der Gradient die Steilheit und Richtung des größten Anstieges in einem Punkt definiert, so definiert die Divergenz, wieviel sich an einem Punkt heraus- oder hineinbewegt (wird z.B. Luft an einem Punkt erhitzt, dehnt

sie sich in alle Richtungen aus; wird sie abgekühlt, zieht sie sich dort zusammen). Man erhält eine Vorstellung der Bewegung an diesem Punkt: Bei positiver Divergenz bewegt sich etwas zum Punkt hin und bei negativer bewegt es sich weg.

Betrachtet man Bilder als zweidimensionale Funktion, so ergibt sich folglich die Poisson-Gleichung in zweidimensionalen kartesischen Koordinaten wie folgt:

$$\triangle u = f \Longleftrightarrow \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) u(x,y) = f(x,y) \tag{2.8}$$

Dank der Diskretisierung auf das Pixelraster kann auch hier wieder eine numerische Lösung durch die finite Differenzen-Methode approximiert werden (vgl. auch Simeon Kap. 2 [62]):

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \approx u\,(x+1,y) + u\,(x-1,y) + u(x,y+1) + u(x,y-1) - 4u(x,y) \tag{2.9}$$

Aus Gleichung 2.9 lässt sich eine typische dünnbesetzte Bandmatrix der Form $Au = b$ bilden. Jedes $u_{x,y}$ ist eine unbekannte Variable u_i mit insgesamt N Variablen über alle unbekannten Pixel. Bekannte Werte von $u_{x,y}$ tragen zur rechten Seite bei.

$$\begin{pmatrix} -4 & 1 & & 1 & & \\ 1 & -4 & 1 & & 1 & \\ & 1 & -4 & 1 & & 1 \\ 1 & & 1 & -4 & 1 & \\ & 1 & & 1 & -4 & \vdots \\ & & 1 & & \cdots & \ddots \end{pmatrix}_{NxN} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_N \end{pmatrix} \tag{2.10}$$

Zur Lösung von 2.10 können daher gängige iterative Lösungsverfahren linearer Gleichungssysteme angewandt werden wie Gauss-Seidel, Successive Overrelaxation oder konjungierte Gradienten, deren nähere Besprechung hier den Rahmen sprengen würde. Der interessierte Leser sei z.B. auf [34] verwiesen.

Anwendung In der Bildverarbeitung lässt sich die Poisson-Gleichung bei verschiedenen Techniken einsetzen. So führen z.B. Pérez et al. [54] verschiedene Anwendungen, unter anderem etwa das nahtlose Klonen oder lokale Änderungen von Bildhelligkeit und -farbe vor. Fattal et al. [28] nutzen sie, um HDR-Bilder (Bilder mit hohem Kontrastumfang) für eine natürlich wirkende Darstellung auf Monitoren zu komprimieren.

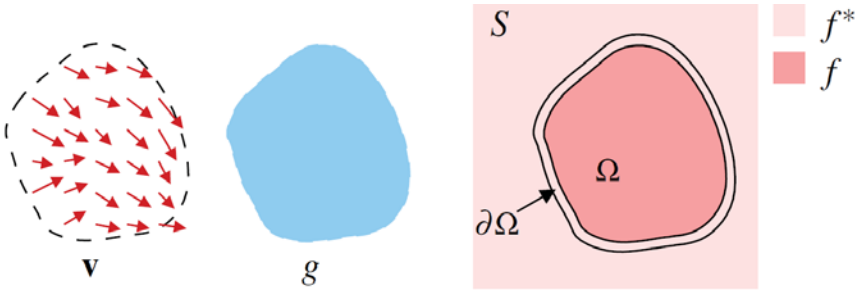


Abbildung 2.3.: Interpolations-Notation. Die unbekannte Funktion f interpoliert im Bereich Ω die Zielfunktion f^* und orientiert sich am Vektorfeld v, das ein Gradientenfeld einer Quellfunktion g sein kann oder nicht. Bildquelle Pérez [54].

Für den hier zu besprechenden Anwendungsfall des Inpainting interessiert vor allem das nahtlose Klonen.

Basierend auf der Erkenntnis, dass es vor allem starke Intensitätssprünge in Bildern sind, die dem Betrachter ins Auge fallen, wird versucht, die Intensitäten vom Rand unseres bekannten Bereichs möglichst weich fortzusetzen und im unbekannten Bereich zu interpolieren. Man betrachte dazu Abbildung 2.3. Dabei sei S eine abgeschlossene Teilmenge von $\mathbb{R}^2$ der Definitionsbereich eines Bildes und Ω eine abgeschlossene Teilmenge von S mit dem Rand $\delta\Omega$. Weiterhin sei f^* unsere bekannte Bildfunktion definiert über S ohne Ω und äquivalent f ein unbekannte Bildfunktion, definiert im Inneren von Ω. Ausserdem sei $\mathbf{v}$ ein Vektorfeld, ebenfalls definiert über Ω. Die einfachste Interpolation von f über Ω bezogen auf f^* wäre die Lösung des folgenden Minimierungsproblem:

$$\min_f \iint_\Omega |\nabla f|^2 \; mit \; f|_{\delta\Omega} = f^*|_{\delta\Omega}, \tag{2.11}$$

wobei ∇ der Nabla-Operator ist (siehe oben). Diese Minimierung muss der zugehörigen Euler-Lagrange Gleichung genügen:

$$\Delta f = 0 \; über \; \Omega \; mit \; f|_{\delta\Omega} = f^*|_{\delta\Omega}, \tag{2.12}$$

mit $\triangle$ als dem Laplace-Operator (siehe oben). Dies entspricht der Laplace-Gleichung (Poisson-Gleichung, wobei $f = 0$) mit Dirichlet-Randbedingung. Diese einfache Methode interpoliert den unbekannten Bereich aber nur unter Berücksichtigung der bekannten Randpixel und wirkt deshalb recht schnell verwaschen bzw. verschwommen (siehe Abbildung 2.4).

Wird jedoch das Minimierungsproblem um eine zusätzliche Bedingung erweitert, kann das verhindert werden. Die Interpolation soll die zusätzlichen Informationen der Gradienten als Orientierung nutzen. Das Minimierungsproblem lautet jetzt wie folgt:

Abbildung 2.4.: Interpolation von f über Ω mit der Laplace-Gleichung.

$$\min_f \iint_\Omega |\nabla f - \mathbf{v}|^2 \, mit \, f|_{\delta\Omega} = f^*|_{\delta\Omega}, \quad (2.13)$$

dessen Lösung die eindeutige Lösung der folgenden Poisson-Gleichung mit Dirichlet-Randbedingung ist:

$$\Delta f = div\mathbf{v} \, über \, \Omega \, mit \, f|_{\delta\Omega} = f^*|_{\delta\Omega}, \quad (2.14)$$

wobei $div\mathbf{v}$ die Divergenz (siehe oben) von v ist, mit $\mathbf{v} = \nabla g$ dem Gradientenfeld. Damit werden bei der Interpolation nicht nur die Randpixel berücksichtigt, sondern es wird auch versucht, so nah wie möglich an den gegebenen Gradienten zu bleiben (siehe Abbildung 2.5). Diese Eigenschaft wird im ersten vorgestellten Verfahren genutzt, um Bildinhalte möglichst natürlich in den unbekannten Bildbereich einzufügen.

(a) (b) (c)

Abbildung 2.5.: Beispiel des nahtlosen Klonens mithilfe der Poisson-Gleichung. Im Originalbild (a) wurde das Gesicht aus (b) durch Orientierung an seinen Gradienten eingefügt. Bild (c) zeigt das Resultat.

3. Stand der Forschung

In diesem Kapitel soll die Entwicklung des Inpaintings bis zum aktuellen Forschungsstand aufgezeigt und erläutert werden. Dies wird im größeren Maße für das Inpainting von Bildern geschehen, da dort die grundlegenden Algorithmen entwickelt worden sind. Aufbauend auf diesen, können dann die Besonderheiten des Video-Inpaintings erläutert und ihr Einfluss auf bisherige und aktuelle Arbeiten nachgezeichnet werden.

3.1. Inpainting für Einzelbilder

Betrachtet man die wissenschaftliche Entwicklung des Inpainting, lassen sich grob zwei Richtungen erkennen, die zur Lösungsfindung in der Forschung untersucht wurden. Das sind die *diffusionsbasierten* Verfahren auf der einen und die *texturbasierten* Verfahren auf der anderen Seite. Für jedes soll detailliert auf die wissenschaftlichen Entwicklungen eingegangen werden, beginnend mit je einem Beispiel der Herangehensweise zum besseren Verständnis der grundlegenden Methodik.

Diffusionsbasiert Die Arbeit zum *diffusionsbasierten* Verfahren von Bertalmio [9] kann als richtungsweisend gelten. In ihr wird erstmals der Begriff des Inpainting formuliert und mit der ihm heute innewohnenden Thematik verknüpft. Bertalmio beschreibt eine Analogie zur realen Welt, die aufgrund ihrer guten Verständlichkeit stellvertretend für diese Methodik hier wiedergegeben werden soll. Der Grundgedanke ist sich anzuschauen, wie ähnliche Probleme im Realen, genauer gesagt bei Restauratoren angegangen werden (vgl. Abbildung 3.1). Wenn diese alte Gemälde erneuern sollen, wird, wie Bertalmio herausfand, im Grunde einer simplen Methodik gefolgt:

Abbildung 3.1.: Klassisches Inpainting

1. Der Gesamteindruck des Bildes gibt eine bestimmte Richtung vor. Ein modernes Bild ist unter Umständen anders zu restaurieren als ein altes. Das Ergebnis soll, wie schon genannt, vom Betrachter nicht als Veränderung erkannt werden.

2. Strukturen wie etwa Kanten, Linien oder Ecken werden in den aufzufüllenden Bereich verlängert.

3. Die Bereiche, die sich durch die fortgesetzten Linien ergeben, werden mit Farben gefüllt, die denen am Rand gleichen.

4. Kleine, passende Details werden hinzugefügt, um die einzelnen Bereiche natürlicher aussehen zu lassen. Es wird also Textur hinzugefügt.

Die klassischen Inpainting-Algorithmen versuchen nun analog dazu, vorhandene Bildinformationen am Rand des aufzufüllenden Bereichs entlang sogenannter Isophoten (Linien entlang gleicher Grauwerte) in den unbekannten Bildbereich hinein zu propagieren. Sie diffundieren quasi hinein, daher der Name. Da der Fluss der Bildinformationen entlang der Bildgeometrie erfolgt, sind in der Literatur auch Bezeichnungen wie *geometriebasiert* oder *strukturbasiert* üblich. Traditionell werden zur mathematischen Modellierung dieses Ansatzes partielle Differentialgleichungen höherer Ordnung (PDG) oder Methoden der Variationsrechnung (vgl. Masnou und Morel [50]), einem Teilgebiet der Mathematik, das sich mit der Lösung von Minimierungs- bzw. Maximierungsaufgaben beschäftigt, verwendet. So beschreibt z.B. Bertalmio in [10] das Weiterführen der Bildinformationen analog zum physikalischen Strömungsprozess durch die aus der Physik bekannte Navier-Stokes-Gleichung. Im Laufe der Zeit sind eine Vielzahl weiterer Arbeiten mit verschiedenen Anpassungen, Veränderungen und Verbesserungsvorschlägen erschienen. Auch wenn sich aktuell das Interesse eher den texturbasierten und hybriden Modellen zuwendet, seien hier für den interessierten Leser einige genannt:

- Masnou und Morel z.B. in [49, 50]: Inpainting fehlender Bereiche durch das Verbinden von gleichwertigen Grauwertlinien bzw. Grauwertkonturen.

- Ballester et al. in [4, 5, 6]: Zusammenhängende Interpolation von Grauwerten und Gradient- bzw. Isophotenrichtung.

- Chan et al. in [14, 15, 16]: Verschiedene Methoden, basierend auf Total-Variation-Modellen, Diffusion unter Beachtung von Kurven und der Elastischen Theorie nach Euler.

- Tschumperlé et al. in [68, 69]: Diffusionprozess mit kurvenbasierter Kantenerhaltung.
- Bornemann und März in [12]: Modell basierend auf Kohärenztransport und Fast-Marching (vgl. [56, 57]).

Allen *diffusionsbasierten* Verfahren sind zwei Probleme gemein. Zum einen funktionieren sie am besten auf kleinen Bereichen. Etwa wenn kleinere Lücken und Löcher zu füllen sind wie Kratzer, Knicke oder Text in Bildern (siehe Abbildung 3.2a). Grosse Flächen werden jedoch durch den diffusionsbasierten Prozess stark weichgezeichnet (siehe Abbildung 3.2b). Zum anderen besitzen sie kein Wissen über die Textur eines Bildes oder größere lokale Zusammenhänge und sind somit meist nur schlecht in der Lage diese zu synthetisieren. Sie sind lokal in dem Sinne, dass sie aus sämtlichen Informationen, die im Bild zur Verfügung stehen, nur die Randinformationen einbeziehen können.

(a) Inpainting nach Bertalmio [9]. Bildquelle: ebd.

(b) Traditionelles Inpainting für grosse Flächen. Bildquelle: Criminisi [20].

Abbildung 3.2.: Beispiel für diffusionsbasiertes Inpainting

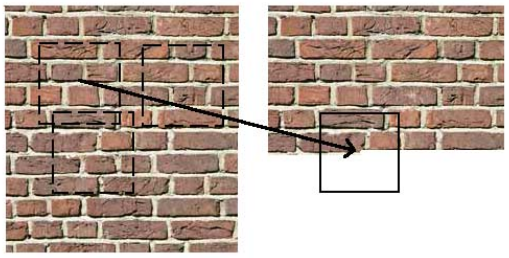


Abbildung 3.3.: Texturbasiertes Verfahren nach Efros und Leung [26]. Mehrere ähnliche Bereiche wurden gefunden und der mittlere Pixel aus einem davon wird in das Bild kopiert.

Texturbasiert Als richtungsweisend für die *texturbasierten* Verfahren sei hier die Arbeit von Efros und Leung [26] (vgl. auch Wei [71]) genannt, da in ihr bereits viele der später üblichen Konzepte aufgegriffen wurden. Das Grundprinzip ist wie folgt (siehe Abbildung 3.3):

1. Festlegen eines Pixels, der als nächstes aufgefüllt werden soll. Efros und Leung gehen dabei noch zeilenweise vor.

2. Festlegen eines Bereiches, z.B. die 8 den zu synthetisierenden Pixel umgebenden Nachbarpixel (auch Exemplar oder Patch genannt).

3. Suchen von anderen Bereichen, die dem vorher ausgewählten ähneln.

4. Zufällige Wahl eines der gefundenen Bereiche und Auffüllen des Pixels mit dem des entsprechenden Bereichs.

5. Wiederhole 1-4 bis das Bild fertig ist.

Der Ursprung des *texturbasierten* Inpainting liegt in der Anwendung aus Elementen der Textursynthese, die schon längere Zeit erforscht wurde (siehe hierzu z.B. [22, 31]). Die reine Textursynthese beschäftigt sich mit dem möglichst nahtlosen und fehlerfreien Replizieren von Mustern (Textur wie z.B. eine Steinwand, siehe auch Abbildung 3.4 - die größere Fläche ist aus der kleineren synthetisiert.).

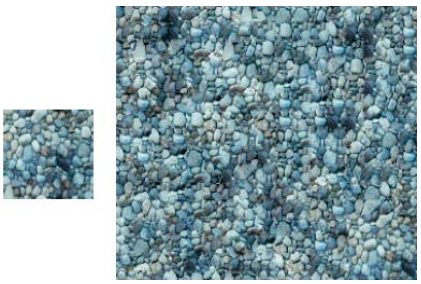


Abbildung 3.4.: Reine Textursynthese nach De Bonet [22].

Dazu wird die Textur als Markov Random Field (MRF) modelliert und ihre diversen stochastischen Merkmale bzw. Parameter in bekannten Bildteilen ermittelt. Anschließend können unbekannte Bildteile mithilfe dieser Informationen berechnet und synthetisiert werden (siehe dazu auch [21, 40, 46]). Für modernes Inpainting lässt sich das jedoch nur beschränkt verwenden (parametrische Textursynthese für Inpainting wurde z.B. von Igehy [32] und Levin [42] untersucht). Realistische Bilder bestehen in der Regel aus mehreren unterschiedlich texturierten Bereichen. Diese müssten segmentiert und einzeln durch die Verfahren bearbeitet werden, was in einem unnötig hohen Nutzeraufwand resultieren würde. Texturübergänge wären dabei auch undefiniert.

Inspiriert vom nicht-parametrisierten Inpainting-Verfahren nach Efros und Leung [26] erschienen eine Vielzahl von neuen Veröffentlichungen. Ein wichtiger Schritt war z.B. die Erkenntnis, dass nicht nur pixelweise, sondern gleich flächenweise Synthetisierung (von sogenannten Patches oder Exemplaren) der unbekannten Bereiche weiterhin vernünftige Resultate liefert, trotz der reduzierten Menge an verfügbaren Interpolationen. Dies bringt den positiven Nebeneffekt einer nicht unerheblichen Reduzierung von Rechenzeit mit sich. Verfahren dieser Art tauchen daher auch häufig als *exemplar-* oder *patchbasiert* betitelt in der Literatur auf. Untersucht wurde unter anderem:

- Position der zu synthetisierenden Patches (Benutzung eines fixen Rasters oder nicht, keine oder harte/weiche Überlappung aneinanderliegender Patches) (u.a. Drori [24]).
- Die Reihenfolge, in der aufgefüllt wird (zeilenweise, kreisförmig, unter Berücksichtigung geometrischer oder farblicher Einschränkungen) (u.a. Qin [55] oder Wu [76]).
- Der Suchraum für ähnliche Patches (in jeder Iteration der gesamte zur Verfügung stehende oder eingeschränkt durch den vorher ausgesuchten Patch).
- Das Maß für die Distanz von Patches und das Finden eines optimalen Kandidaten (simple Aufzählung, komplexe Graphstrukturen, Belief-Propagation etc.) (z.B. Kwok [41], Liu [45] oder Shen [59]).
- Erweitern des Suchraums durch z.B. skalierte und rotierte Varianten der Patches.
- Art des Auffüllens (gewichtete Pixel, Verknüpfung der Werte mehrerer Patches in [75] sowie Farb oder/und Gradientenwerte in [59]).

Der Kerngedanke des *textur-* oder auch *exemplarbasierten* Inpaintings ist das Ausnutzen der Lokalität und Stationarität von natürlichen Bildern. Das heisst, zwei unterschiedliche Nachbarschaften in einem Bild haben bis zu einem gewissen Grad ähnliche statistische Momente (diese Ähnlichkeit kann zum Beispiel über die Summe der quadrierten Differenzen der einzelnen Pixel bestimmt sein).

Wie Demanet et al. [23] hervorheben, ist das Verfahren mathematisch gesehen definiert als das Finden einer Korrespondenzkarte $\Gamma : \Omega \to \Phi$, welche jeder Position x in der zu füllenden Fläche Ω eine zugehörige Position $\Gamma(x) \in \Phi$ im bekannten Teil des Bildes zuweist. Die Pixel des unbekannten Bereichs würden dann aufgefüllt mit $I(x) = I(\Gamma(x))$. Dieses Verfahren ist relativ restriktiv (im englischen "greedy" genannt, also "gierig"), da es jeden zu füllenden Pixel/Patch nur einmal betrachtet und ihm einen endgültigen Wert zuweist. Daher ist die Reihenfolge, in der eingezeichnet wird, sehr wichtig, denn durch einmal falsch gefüllte Pixel wird der Fehler unter Umständen weiter in das Bild hineinpropagiert.

Abbildung 3.5.: Texturbasiertes Inpainting nach Criminisi [20]. Bildquelle: ebd.

Um dies zu verhindern wird z.B. eine prioritätsbasierte Füllreihenfolge vorgeschlagen (etwa in der richtungsweisenden Arbeit von Criminisi [20], siehe Abbildung 3.5), oder aber auch eine Mittelung mehrerer Patches in Wong et al. [75]. Andere Autoren hingegen haben eine globale Optimierung des Inpainting untersucht. Das bedeutet einfach ausgedrückt, dass die Korrespondenzkarte selbst als Bestandteil eines zu minimierenden Energiefunktionals betrachtet wird und eine bestmögliche Kombination aus Patches gesucht werden muss. Oder um es mathematisch zu formulieren: Angenommen die Ähnlichkeit ist definiert als die Summe der quadrierten Differenzen

$$d(N_a, N_b) = \sum_{x \in N_x} |I(a + x) - I(b + x)|^2,$$

wobei N_x die Nachbarschaft der Position x im Bild I ist, dann könnte die Inpainting Energie bzw. das Funktional, das minimiert werden soll, geschrieben werden als

$$E(\Gamma) = \sum_{a \in \Omega} d(N_a, N_{\Gamma(a)}).$$

Es wird also eine Korrespondenzenkarte gesucht, die am besten (im Sinne des Ähnlichkeitsmaßes) den unbekannten Bereich synthetisiert. Das Minimieren dieser Energie ist nicht trivial und diverse Arbeiten sind erschienen, um dieses Problem besser zu verstehen (z.B. in [2, 3]) oder alternative Optimierungspfade zu untersuchen (z.B. in [13, 36, 38, 64, 74, 80]).

Obwohl die *texturbasierten* Verfahren sehr gute Ergebnisse beim Einzeichnen von Textur und einfachen Strukturen zeigen, haben auch sie Schwächen bei komplizierten Strukturen, vor allem in großen zu füllenden Flächen. Daher sind Ideen entwickelt worden, *diffusions-* und *texturbasierte* Verfahren zu kombinieren. Dazu gehört z.B. Bertalmio [11], der ein Bild in Textur und Struktur zerlegt, um beide mit dem jeweils passenden Algorithmus zu füllen, und der die beiden Teile anschließend wieder zusammenführt. Eine andere Möglichkeit nach Sun [63] ist, manuell durch den Nutzer neben der Inpainting-Maske auch die Strukturlinien vorzugeben, an der sich dann der Algorithmus orientieren kann (siehe auch Barnes [7]). Eine aktuelle Arbeit in [83] schlägt vor dies zu automatisieren und die Struktur mithilfe der beschränkten Delauney-Triangulation zu ermitteln, an der wiederum im Anschluss eine Textursynthese erfolgen kann. Weitere hybride Verfahren sind u.a. von Bekir [8], Koppel [39], Wu [77] und Wang [70] vorgeschlagen worden.

Abschließend sei noch das Inpainting als Variante aus dem Gebiet der “sparse representation” erwähnt (“sparse”, engl. für “spärlich”, “wenig vorhanden”). Dieses Modell geht davon aus, dass man das Bild bzw. das Bildsignal durch lineare Kombination weniger verschiedener Basiselemente, genannt Atome, repäsentieren kann. Als Atome werden vielfältige mathematische Konstrukte wie Wavelets, Curvelets, Contourlets oder Bandlets verwendet. Dabei wird vorgeschlagen, das defekte Bildsignal (defekt aufgrund des unbekannten Inpaint-Bereiches) mithilfe dieser Atome wieder herzustellen. Für den interessierten Leser sei ausschnitthaft auf folgende Veröffentlichungen verwiesen: Du [25], Fadili [27], Koh [37], Mairal [47, 48], Mobahi [51], Shen [58] oder Xu [78].

3.2. Inpainting in Videos

Video ist im Endeffekt nicht mehr als die Aneinanderreihung von Einzelbildern (auch Frames genannt). Daher ist es nicht weiter verwunderlich, dass die ersten Algorithmen zum Video-Inpainting versuchten, die bis dato bekannten Methoden zur Einzelbild-Auffüllung zu verwenden (vgl. z.B. [10]). Wie jedoch schon eingangs erwähnt, ergeben sich einige Besonderheiten bei der Auffüllung von Bewegtbild-Lücken, die in der Folgezeit verstärkt untersucht wurden und aufgrund ihrer Komplexität immer noch Gegenstand der Forschung sind. Während beim Inpainting von Einzelbildern nur die Konsistenz der gegebenen lokalen Strukturen gewährleistet werden muss, ist beim Video auch die zeitliche Dimension zu berücksichtigen. Man kann sich leicht vorstellen, dass einzelne, in sich stimmig aufgefüllte Bilder, in der Kombination zu einer Videosequenz plötzlich starke Artefakte wie etwa Flackern aufweisen, weil sich z.B. allein schon die Farbwerte von Frame zu Frame zu stark unterscheiden. Hinzu kommen in den meisten Filmen komplizierte Objektbewegungen (einzelne oder mehrere, sich zueinander unterschiedlich bewegende oder mal näher oder weiter von der Kamera entfernte Objekte), gepaart mit verschiedensten Kamerabewegungen (Schwenk, Rotation, Zoom etc.), die es schwer machen eine konsistente Auffüllung zu erreichen.

Eine der ersten Arbeiten dazu kommt von Wexler et al. [72, 73], die in Anlehnung an das nichtparametrische Sampling von Efros und Leung (siehe [26]) eine Erweiterung des Suchraums in die zeitliche Dimension vornehmen, um dann über eine globale Optimierung zu versuchen, die zeitliche Konsistenz zu erreichen. Die relativ guten Resultate kommen jedoch aufgrund des enorm vergrößerten Suchraums auf Kosten der Rechenzeit (siehe Abbildung 3.6) zustande. Eine Reihe weiterer Veröffentlichungen ist erschienen, die die Szene anhand von Bewegungsschätzung (optischer Fluss etc.) in eine Hintergrundebene und ein oder mehrere bewegte Vordergrundebenen aufteilen (z.B. Cheung et al.[18], Patwardhan[35] oder Wexler [82]). Auf diese können dann unterschiedliche, den Gegebenheiten angepasste Verfahren angewandt werden. Für den Hintergrund ist dies meist eine Variante der texturbasierten Auffüllung, während Vordergrundobjekte mithilfe von Bewegungsmosaiken synthetisiert werden. Bugeau et al. [13] stellen eine Methode vor, die sich nur auf das Auffüllen von Hintergrund beschränkt, wobei dieser dann jedoch auch komplexer sein darf (z.B. durch sich verändernde Lichtverhältnisse, Texturen und Größenverhältnisse). Ling et al. [44] wiederum untersuchen das Wiederherstellen von Vordergrundobjekten, im Speziellen von Menschen, über einen Objektkonturerkennungs- und Synthetisiermechanismus. Eine

Abbildung 3.6.: Video-Inpainting nach Wexler et al. [72]. Bildquelle ebd.. Die relativ guten Ergebnisse erhält man auf Kosten der Rechenzeit. Das Beispiel (120 x 340px x 100 Frames mit 422.000 fehlenden Pixeln) braucht in einer Iteration über eine Stunde.

interessante Variante ist von Jia et al. [33] vorgestellt worden. Ihre Methode benutzt neben einer starken manuellen Komponente allerdings eine Vielzahl verschiedener Techniken, die den Gesamtprozess sehr kompliziert machen. Die Resultate sind gemäß den Rahmenbedingungen relativ gut. Unstimmigkeiten gibt es mitunter aufgrund der interpolierten Bewegung synthetisierter Objekte, die schnell unnatürlich wirken kann. Im Gegensatz zu den bisher vorgestellten Arbeiten, die von einer statischen Kamera ausgehen, ist hier bereits die Möglichkeit einer, wenn auch stark eingeschränkten, Kamerabewegung, vorgesehen (vergleichbar auch Patwardhan [53]). Shih et al. ([60] und [61]) beschäftigen sich explizit mit dem Problem der Schemenbildung ("ghosting artifacts") bei falschem Inpainting von zu entfernenden Objekten. Mit manueller Hilfe werden verschiedene Bewegungsebenen der Objekte und der Kamera extrahiert, um so durch ein erweitertes patchbasiertes Verfahren komplexere Kamerabewegungen wie Zoom und Rotation kompensieren zu können. Video-Inpainting, wenn auch unter simpleren Rahmenbedingungen, wird in [66] und [67] zur Entfernung von Videotext untersucht.

3.3. Einordnung der gewählten Verfahren

Im Rahmen dieser Arbeit wurden zwei textur- bzw. exemplarbasierte Verfahren ausgewählt. Die diffusionsbasierten Methoden sind, wie weiter oben angesprochen, nicht ausreichend in der Lage Strukturen und Bildinhalte von Aufdeckungen in der hier untersuchten Größe wieder herzustellen und wurden deshalb für die Implementierung verworfen. Der erste ausgewählte Algorithmus stammt von Shen et al. [59] Dieses Verfahren aus der Klasse der texturbasierten Einzelbildverfahren soll hier vorgestellt werden, da es, wie in der Einleitung schon erwähnt, neben dem für die Einführung in die übliche exemplarbasierte Inpainting-Methodik günstigen Algorithmus außerdem mit der Einbindung von Gradienten und der anschließenden Rekonstruktion des Bildes durch das Lösen einer Poisson-Gleichung eine interessante inhaltliche Erweiterung bietet. Obwohl der Algorithmus nicht auf die Anwendung auf Videos ausgelegt ist, lassen sich möglicherweise auch diesbezüglich einige Eigenschaften extrahieren, die bei anderen Verfahren von Vorteil sein könnten.

Das zweite Verfahren von Cheng et al.[17] ist durch die Ausrichtung auf die Wiederherstellung von Aufdeckungen, die bei der 3D-Stereosynthese entstehen, sehr gut für diese Arbeit geeignet. Dabei soll durch Integration von Tiefen- und Strukturinformation in die Suche nach passenden Bildteilen eine verbesserte Auffüllung des Einzelbildes erreicht werden sowie die gewichtete Einbeziehung von benachbarten Videoframes und ein finaler Optimierungsschritt die zeitliche Konsistenz zwischen den Frames gewährleisten. Im Vergleich zu anderen möglichen Verfahren, die in der aktuellen Forschung untersucht werden, bietet es sich des Weiteren auch im Sinne dieser Arbeit an, da es ebenfalls auf einer exemplarbasierten Auffüllung basiert. Der Leser ist nach der Lektüre des ersten Verfahrens bereits mit der Methodik vertraut und kann so die Funktionsweise gut nachvollziehen.

4. Algorithmen und Implementierung

Im folgenden Kapitel werden die beiden ausgesuchten Algorithmen im Detail vorgestellt und ihre Implementierung im Rahmen dieser Diplomarbeit erläutert. Das erste Verfahren nach Shen et al. [59] ist übersetzt betitelt als "Gradientenbasierte Bild-Vervollständigung durch Lösen einer Poisson-Gleichung" und kommt, wie schon genannt, aus der Klasse der textur- bzw. exemplarbasierten Inpainting-Varianten. Der Aufgabenstellung folgend soll es eine Einführung in eine der klassischen Methoden liefern, bietet jedoch aufgrund der Einbeziehung der Poisson-Bildbearbeitung (siehe Abschnitt 2.3) einige darüber hinaus interessante und zu untersuchende neue Aspekte. Das zweite Verfahren von Cheng et al. [17], übersetzt als ein "räumlich und zeitlich konsistenter neuartiger Ansichtssynthese-Algorithmus von Video-plus-Tiefe Sequenzen für autostereoskopische Displays", soll darauf aufbauend eine spezielle Methode vorstellen, die für die Aufdeckungen bei 3D-Synthese-Sequenzen konzipiert wurde. Begonnen wird mit einer kurzen Erläuterung der genutzen Entwicklungs- und Programmierwerkzeuge. In den anschließenden Abschnitten wird, nach einer überblickshaften theoretischen Betrachtung des jeweiligen Algorithmus, die konkrete Implementierung beschrieben. Dabei wurde versucht nah am Ursprungsverfahren zu bleiben, um dessen Eigenschaften und Wirksamkeit gut nachvollziehen zu können. Im Rahmen theoretischer Überlegungen sowie nach Erfahrungen innerhalb der Implementierung und den ersten Auswertungen der Qualität der Verfahren wurde die Implementierung angepasst, wobei jeweils in einem abschließenden Abschnitt Vorschläge und nötige Anpassungen in einer Kritik deutlich gemacht werden (siehe Abschnitte 4.2.3 und 4.3.3).

4.1. Programmierumgebung

Beide vorgestellten Verfahren wurden als einfache Konsolenanwendung in C++ mithilfe der Entwicklungsumgebung *Microsoft Visual Studio*[1] implementiert. Die Wahl von C++ bietet aufgrund der maschinennahen Programmierung gute Möglichkei-

[1] Visual Studio Startseite: `http://www.microsoft.com/visualstudio` [19]

ten für eine effiziente Umsetzung von rechenintensiven Implementierungsdetails. Da die gängigsten Grafik- und Videobearbeitungstools in C++ geschrieben sind, wie z.B. *Adobe Photoshop* oder *Autodesk Maya* (vgl. [43]), kann der Code später ohne großen Aufwand in andere Anwendungen eingebettet werden. Zur Vermeidung von integrationsspezifischen Details wurde in dieser Arbeit aber darauf verzichtet. Für die programmtechnische Verarbeitung von Bildinhalten wurde die weit verbreitete Bildverarbeitungsbibiliothek *OpenCV*[2] verwendet. Ursprünglich von Intel initiiert und inzwischen als freie Software verfügbar, liegt ihr Vorteil vor allem in der hohen Performanz und der großen Menge an verfügbaren Algorithmen. Des Weiteren wurde für das Lösen von Gleichungssystemen die ebenfalls frei verfügbare Bibliothek *Taucs*[3] verwendet. Zu erwähnen sei noch die Verwendung des proprietären 2D/3D Konvertierungstools *imcube cinema* der *imcube labs GmbH*[4] , mit dessen Hilfe die benötigten 3D-Stereobildsequenzen effizient und effektiv erzeugt werden konnten.

4.2. Einzelbild-Inpainting

Der Algorithmus von Shen et al.[59] verfolgt die Idee, gradientenbasierte Bildbearbeitung mit dem exemplarbasierten Inpainting zu kombinieren. Dazu wird der unbekannte Bereich mit passenden Gradienten-Patches aufgefüllt, aus denen anschließend ein Bild rekonstruiert wird. Die Gradientendivergenzen sollen dabei wie in Abschnitt 2.3 beschrieben als Orientierung dienen und eine möglichst natürlich wirkende Auffüllung ermöglichen.

4.2.1. Theorie

Gradientenbasiertes Patch-Füllen

Wie bereits erwähnt, ist die Füllreihenfolge für Patches bei den exemplarbasierten Verfahren von entscheidender Bedeutung, um ein natürlich aussehendes rekonstruiertes Bild zu erhalten. So nutzt z.B. Criminisi et al. [20] den Winkel zwischen Isophoten- und Normalen-Richtung, um die Priorität zu bestimmen, mit der bestimmte Bereiche aufgefüllt werden. Dadurch soll sichergestellt werden, dass Bildstrukturen vor den Texturen aufgefüllt werden. In dem hier vorgestellten Verfahren soll eine ähnliche Priorisierung benutzt werden.

[2] OpenCV DevZone: `http://code.opencv.org` [52]
[3] TAUCS, a Lib. of Sparse Linear Solvers: `http://http://www.tau.ac.il/~stoledo/taucs` [65]
[4] imcube labs GmbH: `http://www.imcube.com` [30]

Füllpriorität Die Priorität eines Patches Ψ_t zentriert um den Pixel $p_c \in \partial\Omega$ aus dem Rand des unbekannten Bildbereichs berechnet sich dabei analog zu Criminisi et al. [20] als das Produkt seines Konfidenzwertes mit einem Parameter, der die Struktur widerspiegelt:

$$P(p_c) = \Upsilon(p_c) \cdot G(p_c) \tag{4.1}$$

Dabei soll $\Upsilon(p_c)$, der Konfidenzterm des Patches, eine Aussage über die Verlässlichkeit der vorhandenen Bildinformationen abgeben. Je mehr Pixel des Patches im bekannten Teil des Bildes liegen, um so wahrscheinlicher ist es den Patch adäquat aufzufüllen. Andererseits soll $G(p_c)$, der Gradiententerm des Patches, eine Aussage über die Menge an Struktur (wie z.B. Kanten) in dem Bild geben. Je mehr Struktur in einem Patch vorhanden, umso eher soll er aufgefüllt werden. Die Struktur wird aus dem Gradientenfeld des Bildes in horizontaler und vertikaler Richtung ermittelt. Somit ergibt sich:

- Konfidenzterm
$$\Upsilon(p_c) = \frac{\sum_{q \in \Psi_t \cap (I \backslash \Omega)} C(q)}{|\Psi_t|}, \tag{4.2}$$
mit
$$C(p) = \begin{cases} 0, \forall p \in \Omega \\ 1, \forall p \in I \backslash \Omega \end{cases} \tag{4.3}$$

- Gradiententerm
$$G(p) = \frac{1}{|A|} \sum_{q \in A} \sqrt{G_x^2(q) + G_y^2(q)}, \tag{4.4}$$
wobei A die Nachbarschaft von Pixel p ist und $G = [G_x, G_y]$ das Gradientenfeld des Bildes in horizontaler und vertikaler Richtung.

Ähnlichkeitsmaß Nachdem der Patch mit der höchsten Füllpriorität ermittelt worden ist, wird im Bild nach einem Pendant gesucht, welches die größte Ähnlichkeit nach einem zu definierenden Abstandsmaß aufweist. In Anlehnung an Wexler et al. [73] entwickeln die Autoren dazu ein exponentielles Ähnlichkeitsmaß wie folgt:

$$s(\Psi_s, \Psi_t) = e^{d_c(\Psi_s, \Psi_t) + d_g(\Psi_s, \Psi_t)} \tag{4.5}$$

mit

$$d_c(\Psi_s, \Psi_t) = \sum_{(x,y)} ||\Psi_s^c(x,y) - \Psi_s^c(x,y)|| \tag{4.6}$$

$$d_g(\Psi_s, \Psi_t) = \sum_{(x,y)} ||\Psi_s^g(x,y) - \Psi_s^g(x,y)|| \tag{4.7}$$

Dabei repräsentieren Ψ_s^c bzw. Ψ_t^c die Farbinformationen und Ψ_s^g bzw. Ψ_t^g die Gradienteninformationen der entsprechenden Patches. Unbekannte Pixel werden mit Null gefüllt. Nun wird derjenige Patch ermittelt, welcher, laut den Autoren des Verfahrens, folgender Minimierung genügen muss:

$$\Psi_s = \arg\min_{\Psi_i \in \Phi} \frac{s(\Psi_i, \Psi_t)}{|\Psi_i|}, \tag{4.8}$$

wobei Ψ_i jeweils einen der möglichen Patches aus dem bekannten Bildbereich Φ darstellt. Man sucht also das Exemplar, dessen Differenzen mit dem Zielpatch sowohl bei Intensitäts(Farb)- als auch Strukturwerten der Pixel möglichst gering ist. Die Gradientenwerte des gefundenen Patches werden nun in den originalen Gradientenpatch (nur an den Stellen, wo auch unbekannte Pixel sind) kopiert und in der Maske wird entsprechend vermerkt, dass die Pixel gefüllt wurden. Diese Prozedur wird nun solange ausgeführt, bis alle Pixel gefüllt sind.

Bildrekonstruktion anhand der Gradienten

Nachdem der komplette unbekannte Bereich mit passenden Patches gefüllt wurde, können aus den Gradienten die entsprechenden Farbwerte wiederhergestellt werden. Dazu wird die Divergenz des aufgefüllten Bildes errechnet, was im Endeffekt wieder eine Gradientenberechnung des diesmal aufgefüllten Gradientenbilds ist. Anhand einer simplen Rückwärtsdifferenz-Berechnung für die Divergenzen wird sichergestellt, dass die Werte für die Poisson-Gleichung symmetrisch bleiben. Durch Einsetzen der Werte des Divergenzbildes sowie der Farbwerte am Rand des unbekannten Bereichs in die Poisson-Gleichung erhält man nach deren Lösung die interpolierten Farbwerte.

4.2.2. Implementierung

Der Pseudo-Code in Listing 1 soll den prinzipiellen Ablauf verdeutlichen.

Algorithmus 1 Gradientenbasiertes Inpainting

Input: Bild, Maske
Output: Aufgefülltes Bild

```
// Initialisierung :
Konfidenzwerte initialisieren // Gleichung 4.2
Gradientwerte initialisieren // Gleichung 4.4
Initiale Prioritäten errechnen // Gleichung 4.1

// Gradienten Auffüllung :
while maskiertePixel > 0 do
    Finde Höchste Priorität
    Suche besten Match // Gleichung 4.8
    Kopiere Gradientenwerte in maskierten Bereich
    Setze Konfidenzwerte auf 1 im gefüllten Bereich
    Setze Maskenwert auf 1 im gefüllten Bereich
    Aktualisiere Prioritäten
end while

Berechne Divergenzen

// Anwendung Poisson – Gleichung :
for maskierte Pixel do
    Bilde LGS mit den Divergenzwerten
    Bekannte Pixel am Rand tragen zur rechten Seite bei
end for
Löse Gleichungssystem
Übertrage Lösung in das Output-Bild
```

Initialisierung Der Algorithmus beginnt mit dem Laden des gewünschten Bildes inklusive einem entsprechenden Maskenbild (Abbildung 4.1), in welchem der Inpainting Bereich markiert ist. OpenCV lädt Bilder standardmäßig mit einer Bildtiefe von 8 Bit, d.h. die Farbwerte sind vom Typ eines *unsigned char* und nehmen Werte zwischen 0 und 255 an. Der Genauigkeit in den folgenden mathematischen Operationen wegen wird das Bild jedoch in eine 32 Bit Fließkomma-Darstellung (*float*) umgewandelt, deren Werte nun zwischen 0.0 und 1.0 liegen. Das Maskenbild wird als einfaches Grauwertbild geladen, wobei weiße Pixel (Farbwert 255) maskierte Pixel kennzeichnen. Für Gradient- und Konfidenzwerte sowie Priorität (siehe Gleichungen 4.1, 4.2 & 4.4) werden ebenfalls

Abbildung 4.1.: Bild mit zugehöriger Maske (der zu maskierende Bereich ist weiß)

Bilder erzeugt, um über die Pixelpositionen des Bildes die entsprechenden Terme leicht abrufen zu können. Die Konfidenzkarte wird bis auf den durch die Maske vorgegebenen Bereich mit dem Maximum, dem Wert 1, initialisiert und die Priorität im gesamten Bild auf 0 gesetzt. Für den Gradiententerm müssen zuerst die entsprechenden Gradienten in x- und y-Richtung berechnet werden. Dies geschieht separat für jeden Farbkanal und durch eine einfache Vorwärts-Differenz[5] $\triangle_h[f](x) = f(x+h) - f(x)$, mit f der Bildfunktion und $h = 1$, da auf das Pixelraster diskretisiert wird (siehe Abschnitt 2.2). Zur Verdeutlichung soll das in Abbildung 4.2 erläutert werden. Die beiden oberen 3x9 Raster stellen exemplarisch zwei Grauwertbilder dar, wobei jedes Kästchen einem Pixel entspricht und Grauwerte zwischen 0 (■ schwarz) und 255 (□ weiß) annehmen kann. Das untere der beiden Raster ist dabei die errechnete Gradientenansicht des oberen Rasters. Weiterhin wird schematisch angedeutet, wie in der untersten Pixelreihe durch Vorwärtsdifferenzen Gradientenwerte errechnet werden. Die Pfeile zeigen an, welche Pixelwerte voneinander subtrahiert werden und so das fertige Gradientenbild entsteht.

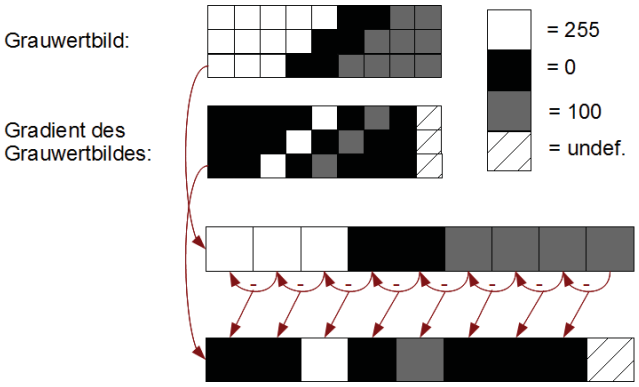


Abbildung 4.2.: Differenzen der Grauwerte ergeben Approximation des Gradienten. Weitere Beschreibung im Text.

[5]Die Symmetrie wird später durch eine Rückwärts-Differenz der Divergenzen wieder hergestellt.

Auf die Behandlung der Randfälle soll weiter unten eingegangen werden (vgl. Abschnitt 4.4). Aus den Gradienten in x- und y-Richtung wird nun für alle Pixel dessen Betrag errechnet (siehe Abbildung 4.3a), so dass anschließend für jeden Pixel des Randes um den unbekannten Bereich herum die Priorität nach Gleichung 4.1 errechnet werden kann. Da die Konfidenz und der Gradientenbetrag konsequenterweise schon im Vorfeld errechnet und gespeichert worden sind, brauchen diese nur noch auf Patchgröße aufsummiert werden. Es sei an dieser Stelle festgehalten, daß für die Implementierung einerseits das 1-Kanal-Gradientenbetragsbild zur Prioritäts- und Differenzberechnung erstellt wird, und andererseits die 3-Kanal-Gradientenbilder der x- und y-Richtung, die für den späteren Rekonstruktionsteil zuständig sind, beibehalten werden. Abbildung 4.3b zeigt eine beispielhafte Visualisierung der initialen Prioritäten, wobei grüne Pixel eher geringere und rote eher höhere Prioritäten symbolisieren. In diesem Beispiel hat der untere linke Bereich aufgrund des Schattenwurfs und der generell "unruhigeren" Struktur des Schnees (im Vergleich zum "glatten" Himmel) die höchste Priorität erhalten. Tabelle 4.1 listet zur Übersicht noch einmal alle Input-Daten der Implementierung, die bei der Initialisierung dieses Verfahren entstehen, auf.

(a) (b)

Abbildung 4.3.: Beispiel zur Initialisierung: (a) Visualisierung des Betrags der Gradienten (maskiert) (b) errechnete Füllpriorität, visualisiert als farbige Linie um den maskierten Bereich (rot = hohe Priorität, grün = niedrige Priorität)

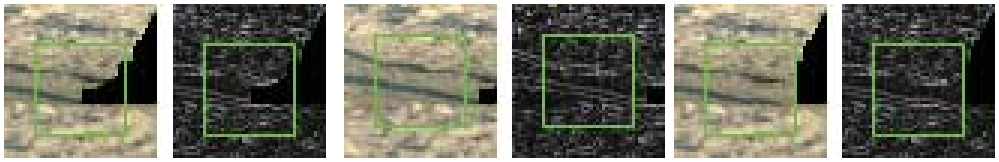
Hauptprogrammschleife Die folgenden Schritte werden solange wiederholt, bis der unbekannte Bereich komplett aufgefüllt ist. Begonnen wird mit der Ermittlung des Patches mit der höchsten Auffüllpriorität, also dem Patch, dessen Mittelpunkt auf dem Rand $\partial\Omega$ liegt und den höchsten Prioritätswert besitzt, wofür eine simple min-max-Funktion von *OpenCV* auf dem gespeicherten Prioritätsbild angewandt werden

Bezeichnung	Benutzung
Originalbild	Berechnung der anderen Daten und zur finalen Ausgabe
Maskenbild	Definiert den zu füllenden Bereich
X-Gradient	Gradientenbetragsbild, Divergenzbildung
Y-Gradient	Gradientenbetragsbild, Divergenzbildung
Konfidenzkarte	Prioritätsberechnung
Gradientenbetrag	Prioritätsberechnung, Differenzberechnung
Farbbetragsbild	Differenzberechnung der Patchsuche
Prioritätskarte	Zur Bestimmung des nächsten zu füllenden Patches

Tabelle 4.1.: Tabelle mit Input-Daten des Verfahrens

kann. Für diesen Patch wird nun im gesamten bekannten Bildbereich nach einem möglichst ähnlichen Patch gesucht. Um eine korrekte Vergleichbarkeit zu gewährleisten, werden keine Patche berücksichtigt, die zum Teil im maskierten Bereich liegen. Die Ähnlichkeit wird über das in Gleichung 4.8 definierte Maß ermittelt. Der zu vergleichende Patch wird dazu zeilen- und spaltenweise wie eine Maske über das Bild geschoben (siehe Abbildung 4.4). *OpenCV* bietet dafür die Möglichkeit in einem Bild eine sogenannte Region von Interesse (engl. abgekürzt ROI) festzulegen und zu verschieben, so dass nur auf dieser gearbeitet werden muss und das rechenintensive Allozieren von neuen Bildmatrizen vermieden wird. Während der Verschiebung des ROIs über das Bild, wird von jedem Pixel des Zielpatches und Quellpatches die absolute Differenz gebildet und aufsummiert, sowohl im Farb- als auch im Gradientenbild, um im Folgenden, wie in Gleichung 4.8 definiert, ein Ähnlichkeitsmaß für alle Patches zu errechnen. Dieser Teil des Algorithmus ist am zeitaufwendigsten, da hier Dutzende von Berechnungen ausgeführt werden müssen, und somit auch am interessantesten für mögliche Optimierungen (siehe Abschnitt 4.4). *OpenCV* bietet zur Berechnung neben grundlegenden mathematischen Matrixoperationen wie Addition, Subtraktion und Ähnlichem, auch weitergehende effiziente Möglichkeiten an, wie etwa Norm-Berechnung im L1- oder L2-Raum. Nachdem der Patch mit der größten Ähnlichkeit, also der geringsten errechneten Differenz, gefunden worden ist, werden die Pixel, die im unbekannten Bereich liegen, vom Quell- zum Zielpatch kopiert, jedoch nur im Gradientenbild, da dies allein für die spätere Poisson-Berechnung von Relevanz ist. Des Weiteren werden im Maskenbild an den neu

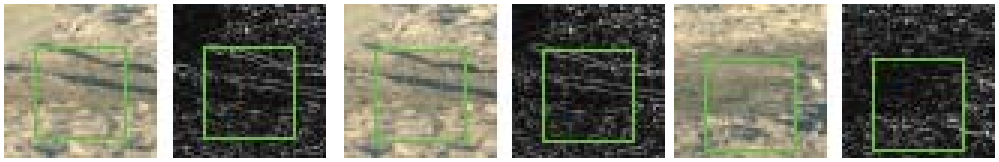
Abbildung 4.4.: Patchsuche, hier dargestellt für das Farbbild.



(a) (b) (c)

Abbildung 4.5.: Jeweils ein Ausschnitt aus dem Originalbild und dem Gradientenbild: (a) aufzufüllender Patch (b) gefundener bester Match (c) aufgefüllter Patch

aufgefüllten Positionen die Pixel von weiß auf schwarz gesetzt, um zu markieren, dass dieser Bereich fertig ist. Durch anschließendes Zählen der verbliebenen Maskenpixel ist nun bekannt, ob das Auffüllen beendet ist oder ob eine neue Iteration gestartet werden muss. Ist dies der Fall, so ergibt sich ein neuer unbekannter Bereich, für dessen Randpixel erneut die Prioritäten errechnet werden. Sobald dies erfolgt ist, kann eine neuer Schleifendurchgang ausgeführt werden. Abbildung 4.5 zeigt exemplarisch einen zu vergleichenden Patch, den als am besten passend gefundenen Patch sowie den fertig aufgefüllten, sowohl im Farbbild als auch im Gradientenbild. Abbildung 4.6 steht in Verbindung zu 4.5 und zeigt zur Demonstration der Funktionsfähigkeit der Patchsuche weitere gefundene Patche.

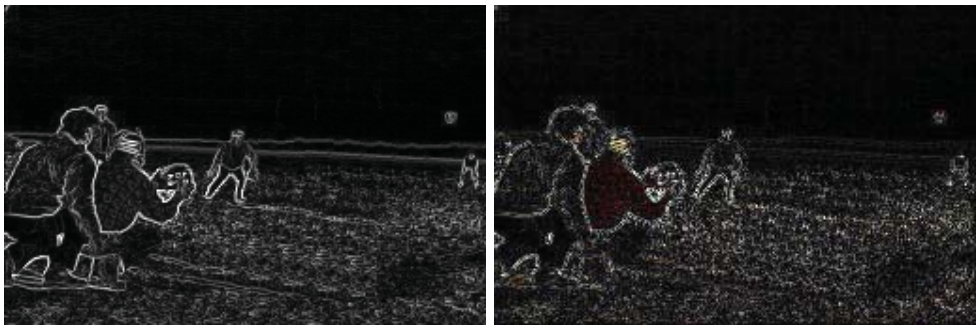


(a) (b) (c)

Abbildung 4.6.: Weitere zu dem in Abbildung 4.5a gezeigten Patch in absteigender Reihenfolge der Ähnlichkeit

Rekonstruktion Nachdem der durch die Maske definierte Bereich mit passenden Gradientenpatches komplett aufgefüllt wurde (Abbildung 4.7a), soll nun daraus das fertige Bild rekonstruiert werden, wozu in einem ersten Schritt entsprechend die Divergenzen für alle 3 Farbkanäle errechnet werden. Da vorher mit einer Vorwärtsdifferenz gearbeitet wurde, wird nun zur Wahrung der Symmetrie die Rückwärtsdifferenz errechnet (vgl. Abbildung 4.7b) und deren Ergebnisse zur Bildung eines Gleichungssystems der Form $Au = b$, wie oben beschrieben, herangezogen. Zur Lösung wird die angesprochene Bibliothek *Taucs* benutzt. Dazu wird zum einen mithilfe der Funktion *taucs_ccs_create* die Matrix A erstellt, welche mit den jeweiligen Faktoren der

Unbekannten u_i gefüllt wird (vgl. Formel 2.9). Zum zweiten wird ein Vektor b, der die rechte Seite der Gleichung repräsentiert, erstellt und mit den Divergenzwerten sowie bekannten Randpixeln gefüllt. Zur Erinnerung: jedes u_i entspricht einem der unbekannten Pixel $u(x, y)$ des zu füllenden Bildes und soll dem gegebenen Gradientenfeld möglichst gut angenähert werden, was der Lösung des LGS entspricht. Durch Aufruf der Funktion *taucs_linsolve* mit den genannten Variablen sowie der Option *"taucs.factor.LU=true"* wird die von *Taucs* zur Verfügung gestellte automatische LU-Dekomposition mit passendem Lösungsalgorithmus benutzt. Weitere Einzelheiten dazu können der entsprechenden Anleitung in [65] entnommen werden. Als Resultat liefert die Funktion den Lösungsvektor u mit den errechneten Bildwerten, die nach der Zuordnung zu den jeweiligen Bildpixeln abgespeichert werden können und das fertige Bild entstehen lassen. Die praktischen Anwendungsresultate werden in Kapitel 5 vorgestellt.



(a) (b)

Abbildung 4.7.: Rekonstruktion: (a) Fertig aufgefülltes Gradientenbild und (b) daraus errechnetes Divergenzbild

4.2.3. Kritik

Bei Betrachtung von Gleichung 4.8 fällt auf, dass die von Shen et al. Shen u. a. [59] vorgenommene Normierung mit der Anzahl der betrachteten Pixel an dieser Stelle nicht unbedingt verständlich erscheint. Analog dazu ist das zugehörige Abstandsmaß in Gleichung 4.5 kritisch zu betrachten. Es stellte sich heraus, dass die Ergebnisse derselben selbst bei kleinen Patchgrößen sehr hoch werden und bei Erhöhung der Patchgröße mitunter den Wertebereich gängiger Variablentypen ausreizen. Der Autor dieser Diplomarbeit schlägt deshalb vor und hat dies auch in der Implementierung umgesetzt, den Term $|\Psi_i|$ entsprechend in den Exponenten zu verlagern, um eine erwartete und plausiblere Normierung zwischen Null und Zwei zu erhalten. Des

Weiteren sei festgehalten, dass das Abstandsmaß nach Gleichung 4.8 durchaus Möglichkeiten bietet Gradienten- und Bildinformationen unterschiedlich zu gewichten. Darauf sind jedoch weder die Autoren des Verfahrens, noch der Autor dieser Arbeit näher eingegangen und dies bleibt zu untersuchen. Die beiden Terme sind in dieser Arbeit entsprechend gleichwertig in die Berechnung eingegangen.

4.3. Video-Inpainting

Der Inpainting-Algorithmus von Cheng et al. [17] untersucht die Erstellung von Video-plus-Tiefe Sequenzen für autostereoskopische Bildschirme. Das Hauptanliegen ist dabei örtlich und zeitlich konsistente Syntheseresultate der Aufdeckungen zu erhalten. Die lokale Konsistenz des exemplarbasierten Ansatzes wird wie gehabt durch eine Füllreihenfolge unter Benutzung der Bildgradienten und zusätzlicher Nutzung der Tiefeninformation ermöglicht, die sicherstellt, dass Bildstrukturen korrekt weitergeführt werden. Die zeitliche Konsistenz hingegen wird durch einen Optimierungsprozess forciert, der Farb- und Tiefenwerte, die mithilfe der benachbarten Frames geschätzt werden, verbessert.

4.3.1. Theorie

Ermitteln der Füll-Priorität Analog zum Inpainting von Bildern des ersten Verfahrens ist auch hier die Füllreihenfolge von entscheidender Bedeutung, um die lokalen Strukturen natürlich fortzuführen. Neben den Gradienten sind zusätzliche Tiefeninformationen für alle Pixel gegeben, die mit in die Bewertung einfließen sollen. Geometrisch betrachtet ist klar, dass die unbekannten Bereiche, die durch die 3D-Stereosynthese entstanden sind, im Sinne der Tiefe weiter hinten liegen müssen als Teile der angrenzenden Bildbereiche und folglich auch eher nach Hintergrundpixeln gesucht werden muss. Die Prioritätsberechnung eines Patches, zentriert an der Position (u, v) in der lokalen Umgebung $W(u, v)$ lautet daher wie folgt:

$$P(u,v) = \frac{\sqrt{G_{\parallel}(u,v) + G_{\perp}(u,v)}}{D(u,v) + \varepsilon}, \tag{4.9}$$

mit

$$
\begin{aligned}
G_{\parallel}(u,v) &= \sum_{(x,y)\in W_{(u,v)}} p(x,y)g_{\parallel}(x,y) \\
G_{\perp}(u,v) &= \sum_{(x,y)\in W_{(u,v)}} p(x,y)g_{\perp}(x,y)
\end{aligned}
\tag{4.10}
$$

und

$$
p(u,v) = \begin{cases} 0, & (u,v) \in \Omega \\ 1, & sonst \end{cases}. \tag{4.11}
$$

$G_{\parallel}$ und $G_{\perp}$sind die Summen der bekannten Gradientenbeträge $g_{\parallel}$ bzw. $g_{\perp}$ in horizontaler und vertikaler Richtung, wobei genannte Bekanntheit durch die Funktion p ermittelt wird. $D(u,v)$ beschreibt die Tiefe an der aktuellen Position und ε stellt sicher, dass nicht durch 0 dividiert wird und bietet die Möglichkeit, in geringem Maße die Tiefe in der Priorität unterschiedlich zu gewichten.

Ermitteln des besten Patches Nachdem der Patch mit der höchsten Priorität zum Auffüllen ermittelt wurde, folgt der Prozess der Suche nach passenden Korrespondenzen. Dazu soll zuerst der Suchraum näher definiert werden. Gesucht sind all die Patches in den benachbarten Quell- oder Referenzframes, die im zu synthetisierenden Zielframe $\tilde{I}(t)$ dem Patch $\tilde{I}(u,v,t)$ mit der höchsten Priorität am ähnlichsten sind. Dazu wird $\tilde{I}$ und $\tilde{D}$ definiert als die Menge aller Referenzframes (Farb- und Tiefenbilder) inklusive des, wenn vorhanden, vorher synthetisierten Frames:

$$
\tilde{I} = \{I(t-n^{-}), ..., I(t-n^{+}), \hat{I}(t-1)\} \tag{4.12}
$$

$$
\tilde{D} = \{D(t-n^{-}), ..., D(t-n^{+}), \hat{D}(t-1)\} \tag{4.13}
$$

Die Gesamtzahl der zu durchsuchenden Frames beträgt also $n^{-} + n^{+} + 2$, wobei n ein Parameter ist der festlegt, wieviel benachbarte Frames vor und nach dem aktuellen Frame einbezogen werden sollen (vgl. Abbildung 4.11). Zur Berechnung der Ähnlichkeit wird eine Kostenfunktion aufgestellt, die die Unterschiede zwischen dem ausgewählten Patch in $\hat{I}$ und allen Referenzbildern I_k aus $\tilde{I}$ mit den dazugehörigen Tiefenkarten D_k aus $\tilde{D}$ ermittelt. Der Index k referenziert entsprechend in der Menge aus Referenzframes den aktuellen zu vergleichenden Frame für die Gleichungen 4.12 und 4.13. Damit lautet die Kostenfunktion wie folgt.

$$C(u,v,t,k) = \min_{(\triangle u_k, \triangle v_k)} \sum_{(x,y)\in W_{(u,v)}} \{p(x,y,t) \cdot (||\hat{I}(x,y,t) - I_k(x+\triangle u_k, y+\triangle v_k)|| \\ + \lambda_D |\hat{D}(x,y,t) - D_k(x+\triangle u_k, y+\triangle v_k)| \\ + \alpha(1-p(x,y,t))|\hat{D}(x,y,t) - D_k(x+\triangle u_k, y+\triangle v_k)|\} \tag{4.14}$$

Der Patch wird dazu, wie schon beim ersten Verfahren, pixelweise, definiert durch den Translations-Vektor $(\triangle u_k, \triangle v_k)$, über die zu vergleichenden Bilder geschoben (vgl. Abbildung 4.4). Die Kostenfunktion setzt sich zusammen aus zwei Haupttermen, die über den Parameter α entsprechend gewichtet werden können. Der erste Term bezieht sich auf alle bekannten Farb- und Tiefenpixel, diese wiederum gewichtet über den Parameter λ_D. Der zweite Term bezieht sich auf alle unbekannten Tiefenpixel des Patches, was hilft, die Kohärenz der Tiefenkarte mit den im Suchprozess bereits ermittelten Werten zu wahren. Über die Summe der absoluten Differenzen der Pixelwerte an jeder Position (x, y) in unserem Nachbarschaftsbereich $W_{(u,v)}$ werden nun die Kosten ermittelt. Der Patch, dessen Kosten am niedrigsten sind, ist unserem Zielpatch damit am ähnlichsten. Es sei festgehalten, dass sich analog zum ersten Verfahren auch die Einbeziehung der Gradienteninformationen als sinnvoll erweisen könnte. Dies wurde von den Autoren dieses Verfahrens jedoch nicht weiter berücksichtigt.

Auffüllung Seien $(\triangle u_k^*, \triangle v_k^*)$ die optimalen Positionen in den einzelnen Referenzbildern k, gefunden durch die Kostenfunktion 4.14. Dann lassen sich nun die unbekannten Pixel über ein Schätzverfahren wie folgt berechnen:

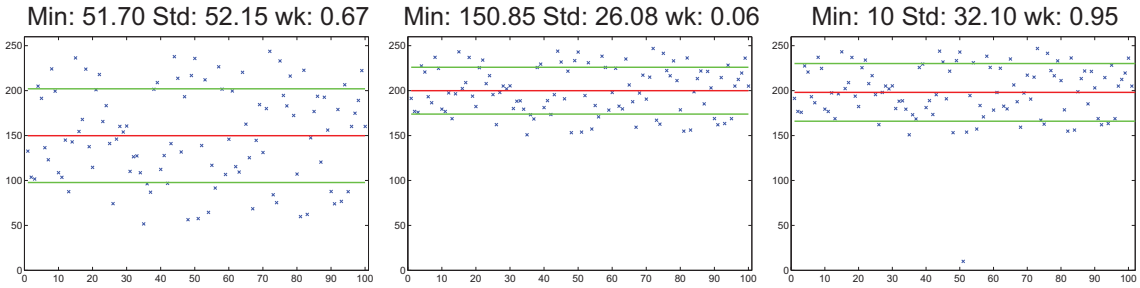
$$\tilde{I}(u,v,t) = \frac{\sum_k \omega_k I_k(u+\triangle u_k^*, v+\triangle v_k^*)}{\sum_k \omega_k} \tag{4.15}$$

$$\tilde{D}(u,v,t) = \frac{\sum_k \omega_k D_k(u+\triangle u_k^*, v+\triangle v_k^*)}{\sum_k \omega_k}. \tag{4.16}$$

Das Gewicht ω_k ist gegeben durch

$$\omega_k = \frac{2\hat{\sigma}_C^2}{C^2(u,v,t,k) + 2\hat{\sigma}_C^2}, \tag{4.17}$$

wobei $\hat{\sigma}_C$ die erwartete Standardabweichung der Kosten bezeichnet. Das Gewicht sagt etwas über die Qualität des gefundenen Patches im entsprechenden Frame k aus, und zwar über den Zusammenhang zwischen den minimalen Kosten und der Standardabweichung sämtlicher errechneter Kosten. Dies soll an einem kleinem Beispiel näher



(a) (b) (c)

Abbildung 4.8.: Veranschaulichung des Zusammenhangs von Standardabweichung, Minimalkosten und Gewichtung. Für alle gilt: rot = Mittelwert, grün = Standardabweichung, Wertebereich von 0 bis 250. (a) Werte zw. 50 - 250 ergeben eine ungenaue Aussage, daher nur mittlere Wichtung: $w_k = 0.67$ (b) Werte zw. 150 - 250 und ein hoher Minimalwert ergeben eine schlechte Wichtung: $w_k = 0.06$ (c) Werte zw. 150 - 250 und ein niedriger Minimalwert (beachte Wert $f(51) = 10$) ergeben eine gute Wichtung: $w_k = 0.95$

ausgeführt werden. In den Abbildungen 4.8 sind je 100 zufällige Zahlenwerte abgetragen, inklusive des Mittelwerts und der Standardabweichung, welche beispielhaft die errechneten Kosten repräsentieren sollen. Der Wertebereich sei der Einfachheit halber von 0 bis 255 angenommen, z.B. bei Differenzen einzelner Grauwertpixel. Betrachtet man nun Abbildung 4.8a, erkennt man, dass sich aufgrund der weiten Streuung der Kosten von 50 bis 250 keine genauen Aussagen über die Güte der Ergebnisse machen lassen. Daher ergibt sich mittels Gleichung 4.17 nur eine mittlere Wichtung mit $w_k = 0.67$. In Abbildung 4.8b hingegen bewegen sich die Kosten im oberen Drittel des Wertebereichs und auch der gefundene Minimalwert ist sehr hoch und konsequent folgt daraus eine schlechte Wichtung mit $w_k = 0.06$. Zu guter Letzt noch Abbildung 4.8c. Die Kosten liegen wieder zwischen 150 und 250, jedoch sagt ein niedriger Minimalwert aus, dass hier im Vergleich zu den anderen Kosten ein sehr passendes Ergebnis vorliegen muss und entsprechend fällt die Gewichtung mit $w_k = 0.95$ sehr gut aus.

Nachdem im zu synthetisierenden Bild und der Tiefenkarte alle unbekannten Pixel des Patches wie beschrieben aufgefüllt wurden, können nun die Prioritäten für die Kante des veränderten unbekannten Bereichs neu berechnet werden. Für den Patch mit der höchsten ermittelten neuen Priorität werden wieder die ähnlichsten in den benachbarten Frames gesucht und so weiter, bis sämtliche maskierte Pixel aufgefüllt worden sind. Wenn der aktuelle Frame aufgefüllt ist, kann der nächste Frame

entsprechend vervollständigt werden, bis letztendlich alle Frames der Videosequenz inklusive Tiefenkarten komplettiert sind.

Refinement Nachdem alle Frames erfolgreich mit Patches aufgefüllt wurden, soll nun ein weiterer Durchlauf auf Grundlage dieser Ergebnisse durchgeführt werden, um die Ergebnisse zu verbessern und Inkohärenzen zu optimieren. Dies verläuft ähnlich zum Algorithmus des vorherigen Abschnitts, nur wird statt eines lokalen Patches, der immer nur eine Teilmenge der Aufdeckung ersetzt, jetzt ein globaler verwendet, der alle Aufdeckungen beinhaltet. Die Einzelframeergebnisse sollen quasi noch einmal untereinander angepasst werden. Dementsprechend wird diesmal keine Prioritätsberechnung notwendig sein. Des Weiteren bezieht sich der Index $k = t - n^-, ..., t + n^+$ nur noch auf die Menge der im ersten Durchlauf synthetisierten Bilder und Tiefenkarten. Die zu berechnende Kostenfunktion verändert sich wie folgt:

$$C'(u,v,t,k) = \min_{(\triangle u_k, \triangle v_k)} \sum_{(x,y)\in W'_{(u,v)}} p'(x,y,t) \cdot (||\hat{I}(x,y,t) - \hat{I}(x+\triangle u_k, y+\triangle v_k, k)|| + \lambda_D |\hat{D}(x,y,t) - \hat{D}(x+\triangle u_k, y+\triangle v_k, k)|) \quad (4.18)$$

wobei $W'_{(u,v)}$ den globalen Bereich der Auffüllungen beschreibt und der Parameter λ_D wie gehabt zur Gewichtung zwischen Farb- und Tiefeninformationen dient. Die Funktion p' beschreibt analog zum vorherigen Abschnitt, ob ein Pixel (u, v) zum Zeitpunkt t im maskierten Bereich liegt und zusätzlich, ob dieser Pixel im Hintergrund liegt.

$$p'(u,v,t) = \begin{cases} 0, & (u,v) \in \Omega^t oder\, \hat{D}(u,v,t) > \gamma \\ 1, & sonst \end{cases} \quad (4.19)$$

Der Parameter γ definiert entsprechend den Schwellwert zwischen Vorder- und Hintergrund. Die Verdeckungsbereiche werden nun analog zu den Gleichungen 4.15 bis 4.17 berechnet und gefüllt, wobei C mit C'und $I_k \in \tilde{I}$ mit $\hat{I}$ ersetzt wird:

$$\tilde{I}(u,v,t) = \frac{\sum_k \omega_k \hat{I}(u+\triangle u_k^*, v+\triangle v_k^*, k)}{\sum_k \omega_k} \quad (4.20)$$

$$\tilde{D}(u,v,t) = \frac{\sum_k \omega_k \hat{D}(u+\triangle u_k^*, v+\triangle v_k^*, k)}{\sum_k \omega_k} \quad (4.21)$$

$$\omega_k = \frac{2\hat{\sigma}^2_{C'}}{C'^2(u,v,t,k) + 2\hat{\sigma}^2_{C'}} \quad (4.22)$$

4.3.2. Implementierung

Der Pseudo-Code in Listing 2 soll den prinzipiellen Ablauf verdeutlichen.

Algorithmus 2 Video-Inpainting

Input: Bilder, Masken, ggf. vorherige Syntheseergebnisse
Output: Synthetisierte Frames

```
// Initialisierung :
Gradientwerte initialisieren // Gleichung 4.10
Initiale Prioritäten errechnen // Gleichung 4.9

// Patchauffüllung :
for jeden Input-Frame do
    while maskierte Pixel > 0 do
        Finde höchste Priorität
        for jeder Referenzframe do
            Suche besten Match // Gleichung 4.14
            Berechne die Gewichtung des gefundenen Patches // Gleichung 4.17
        end for
        // Gleichung 4.15 und 4.16
        Kopiere gewichtete Farb- und Tiefenwerte in den maskierten Bereich
        Aktualisiere Prioritäten
    end while
end for

// Refinement :
for jeden zuvor synthetisierten Frame do
    Finde globales Aufdeckungsfenster
    Subtrahiere Vordergrundobjekte und Maske
    for alle Syntheseframes do
        Suche besten Match für Aufdeckungsfenster // Gleichung 4.18
        Berechne dessen Gewichtung // Gleichung 4.22
    end for
    // Gleichung 4.20 und 4.21
    Kopiere gewichtete Farb- und Tiefenwerte in die Aufdeckung
end for
```

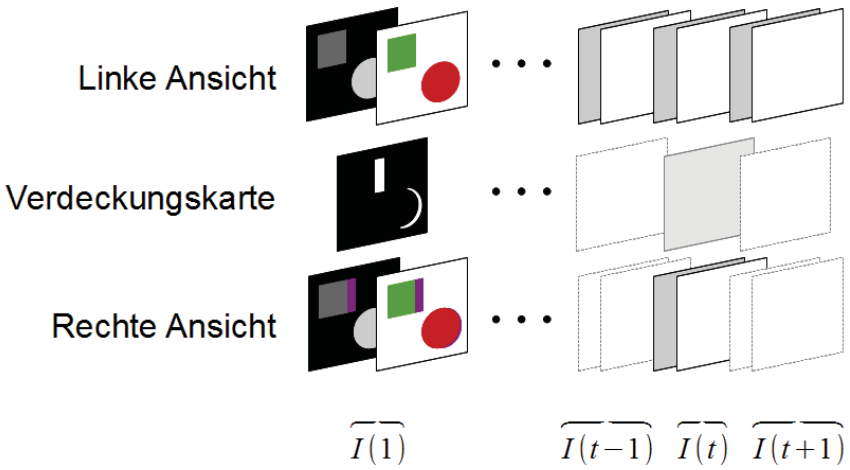


Abbildung 4.9.: Benötigte Eingabebilder, die zu laden sind.

Initialisierung

Der Algorithmus beginnt mit dem Laden der erforderlichen Bilder (siehe Abbildung 4.9). Im Folgenden wird davon ausgegangen, dass das vorhandene Bildmaterial die linke Ansicht repräsentiert und dass die, wie am Anfang des Kapitels unter 4.1 erläutert, mittels der *imcube cinema* Software synthetisierten Ansichten demzufolge für das rechte Auge sind und Aufdeckungen enthalten, die gefüllt werden sollen. Geladen werden also mindestens das linke Bild inklusive Tiefenkarte des aktuellen Frames sowie, wenn erforderlich, der vorherigen und folgenden Frames, je nachdem, wie weit dies per Parameter vorgegeben wurde. Hinzu kommt das aus der Stereosynthese stammende Bild und die Tiefenkarte für die rechte Ansicht inklusive einem Maskenbild, das die Aufdeckungen markiert, wobei weiße Pixel (Farbwert 255) maskierte Pixel kennzeichnen. Die Tiefenkarte ist ein Grauwertbild bei dem alle Tiefenwerte auf Werte zwischen 0 (schwarz) und 255 (weiß) gemapped werden. Je weiter ein Objekt im Hintergrund liegt, umso dunkler ist es. Wenn der Algorithmus nicht im allerersten Frame startet, wird weiterhin das fertig gefüllte Bild des letzten Durchlaufs benötigt.

Um die Menge der nötigen Daten besser zu bündeln, wurde entschieden ein *C struct* anzulegen, welches als Objekt einem Frame entsprechen soll. Dieses wiederum wird pro Frame in einer *hashmap* abgespeichert, wobei der Zugriff über die Framenummer geschieht und damit leichten Zugriff auf die Daten ermöglicht. Nun wird jeweils noch ein Gradientenbild in X- und Y-Richtung, ähnlich wie im gradientenbasierten Verfahren, angelegt und durch eine einfache in *OpenCV* bereitstehende Sobelberechnung

initialisiert (Betrachtung der Randbereiche in Abschnitt 4.4). Anschließend kann die Priorität am Rand $\delta\Omega$ der Aufdeckung erstmals ermittelt werden.

Prioritätsberechnung

Abbildung 4.10.: Darstellung der Prioritätsberechnung. Von links nach rechts: Tiefenkarte, Gradientenbeträge, Prioritäten bei Iteration 0, 35, 49 und 85 sowie ansteigend von grün nach rot. Erläuterungen im Text.

Zu Beginn jeder Iteration eines Frames werden die Prioritäten der Randpixel nach Gleichung 4.9 errechnet. Diese soll eine Position am Rand umso wichtiger für die nächste Auffüllung werten, je tiefer der Pixel im Hintergrund liegt oder umso mehr Struktur in Form von Gradienteninformation den Pixel umgibt. Abbildung 4.10 verdeutlicht dies und zeigt in den beiden ersten Ausschnitten ganz links die zugehörigen Tiefen- und Gradientenbilder des betrachteten Bereiches. Danach folgen Darstellungen der ermittelten Prioritäten zu verschiedenen Iterationszeitpunkten des Algorithmus. Der dritte Ausschnitt von links zeigt die initialen Prioritäten. Wie zu sehen ist, wird der Bereich oben am Kopf, der räumlich tiefer liegt, favorisiert, auch weil dort zusätzlich einiges an Struktur zu finden ist. In den folgenden Iterationen wird der untere Bereich an der Schulter der Figur priorisiert, da er ebenfalls mehr im Hintergrund liegt als beispielsweise die Statue direkt hinter dem Vordergrundobjekt (siehe vierter und fünfter Ausschnitt von links) und auch dort zuerst in der Region der Füße der rechts im Hintergrund befindlichen Statue, da dort mehr Gradienteninformationen vorhanden sind. Zuletzt werden die eher im Vordergrund liegenden Bereiche aufgefüllt (siehe Ausschnitt ganz rechts), wieder unter zusätzlicher Beachtung der Struktur (z.B. der Halskrause der Figur).

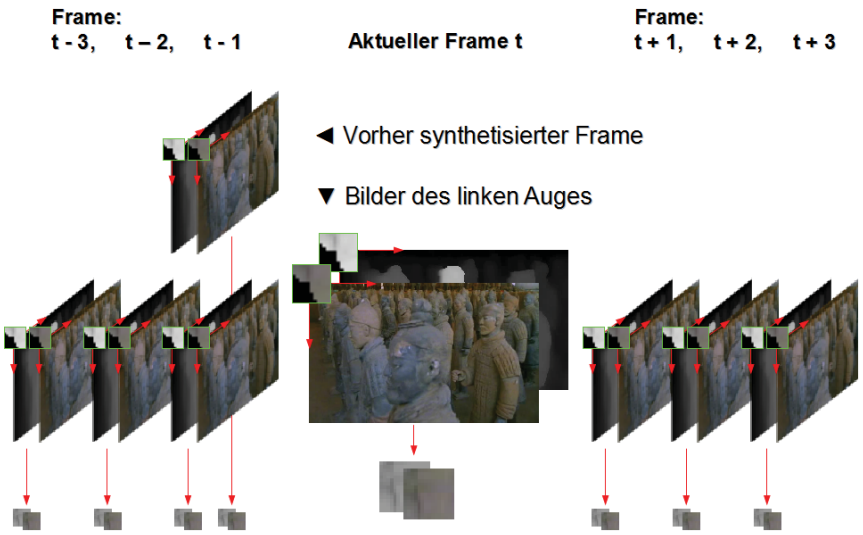


Abbildung 4.11.: Visualisierung der Suche in Farb- und Tiefenkarten aller Referenzframes. Dazu gehören das aktuelle Bild sowie die angrenzenden Nachbarbilder. Des Weiteren, wenn vorhanden, das Resultat der vorherigen Frame-Auffüllung. Aus allen Vergleichen ergibt sich ein ähnlichster Farb- und Tiefenpatch zur Auffüllung.

Hauptprogrammschleife

In der Hauptprogrammschleife werden nun alle vorhandenen Frames aufgefüllt. Dazu werden in einer Unterschleife die unbekannten Bildteile solange mit passenden Patches gefüllt, bis keine maskierten Pixel mehr vorhanden sind. Konkret wird im aktuellen Frame der Patch mit der höchsten Auffüllpriorität als Basis genommen, um in allen Referenzframes nach einem möglichst ähnlichen zu suchen. Dies funktioniert im Prinzip über Farb- und Tiefenwertdifferenzen. Betrachtet werden dazu die Kosten, definiert in Gleichung 4.14, aller möglichen Patches eines Referenzframes, und derjenige mit den geringsten ermittelten Kosten wird als Ergebnis zurückgegeben. Der zu vergleichende Patch wird dazu, wie in Abbildung 4.11 zu sehen ist, zeilen- und spaltenweise wie eine Maske über das jeweilige Referenzbild geschoben. Dabei werden zur Effizienzsteigerung wieder die bekannten ROIs (siehe oben) aus *OpenCV* und andere Optimierungen angewandt (vgl. Abschnitt 4.4). In jedem Frame wird sich die Position des Patches gemerkt, an der die Ähnlichkeit am höchsten ist. Des Weiteren müssen die errechneten Kosten aller anderen Patchvergleiche gespeichert werden, da dies benötigt wird, um die Standardabweichung aller Kosten einer Patchsuche zu ermitteln und im Folgenden die Gewichtung der einzelnen Frames zu erhalten. Die *C++*-Standardbibliothek bietet für die Speicherung ausreichende Containerklassen

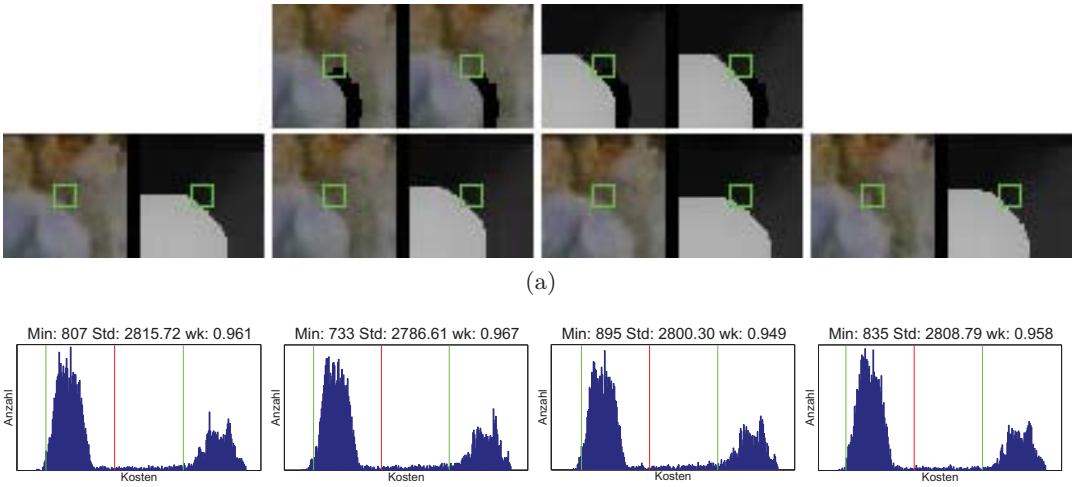


Abbildung 4.12.: Ergebnis der Patchsuche in Frame 2 - Iteration 6: (a) gefundene Farb- und Tiefenpatches, (b) das Histogramm aller Kosten pro durchsuchtem Referenzframe. Erläuterungen im Text.

an, wie etwa die *std::vector* Klasse, welche hier zum Einsatz kam. Abbildung 4.12 zeigt exemplarisch die Ergebnisse der Suche in den Referenzframes. Zu sehen sind jeweils die Farb- als auch die Tiefenkarten, zuerst die gesuchten Ausschnitte und das Ergebnis der Auffüllung (4.12a oben), gefolgt von den besten gefundenen Quellpatches (4.12a unten). Als gefundene Quellpatches sind hier von links nach rechts die Patches aus der linken Ansicht des aktuellen Frames, des vorherigen synthetisierten Ergebnisses, des vorherigen linken Nachbarframes sowie des nachfolgenden linken Nachbarframes angegeben. Nachdem die Suche abgeschlossen ist, sollen als Nächstes an den unbekannten Stellen des gesuchten Patches die Farb- und Tiefenwerte über die gewichteten Informationen aus den Nachbarframes zusammengesetzt werden, um eine erste Annäherung an unsere Auffüllung zu erhalten. Die konkrete Auffüllung erfolgt über die Gewichtung nach Formel 4.17. Abbildung 4.12b soll den Zusammenhang dahinter, der auch schon in Abschnitt 4.3.1 zur Sprache kam, kurz in der konkreten Anwendung beleuchten. Zu sehen sind die Histogramme der Kosten (von links nach rechts ansteigend) aus den Patchsuchen von Abbildung 4.12a (nur untere Reihe). Da nur ein grundlegender Sachverhalt dargestellt werden soll, wurde auf die genaue Beschriftung der Achsen verzichtet, um die Darstellung nicht unübersichtlicher zu machen. In rot ist der Mittelwert abgetragen und links und rechts davon jeweils in grün die Standardabweichung. Es fällt auf, dass die Histogramme aufgrund des ähnlichen Suchraums nicht stark voneinander abweichen, schließlich ist der Unterschied

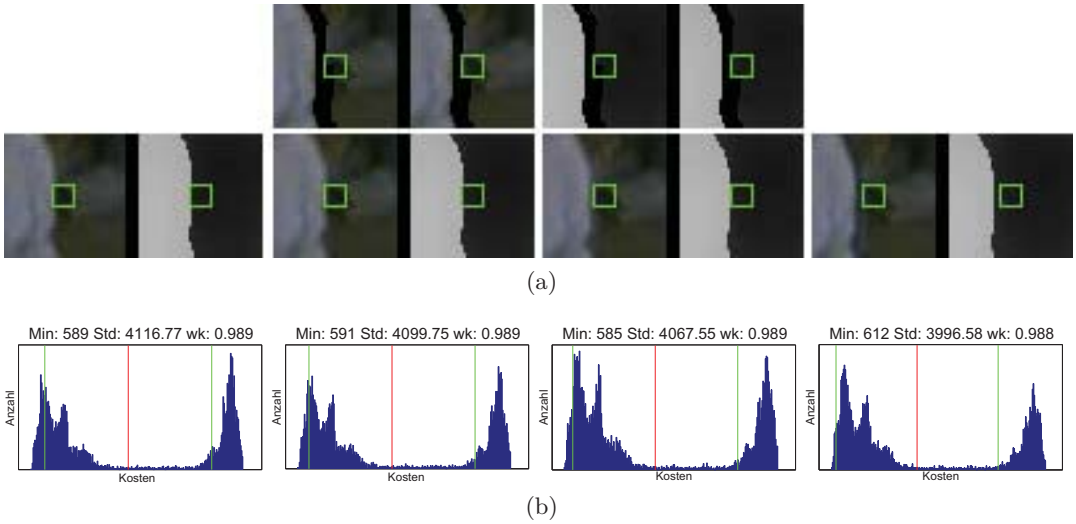


Abbildung 4.13.: Ergebnis der Patchsuche in Frame 0 - Iteration 8: (a) gefundene Farb- und Tiefenpatches, (b) das Histogramm aller Kosten pro Referenzframe. Erläuterungen im Text.

zwischen einzelnen Frames nicht allzu groß. Man kann erkennen, dass ein Großteil der Kosten verhältnismäßig klein ist (erster "Berg" im Histogramm aufgrund des ähnlichen Suchraums), aber zusätzlich noch ein kleinerer, aber nicht unerheblicher Anteil an sehr hohen ermittelten Kosten existiert (zweiter "Berg" durch alle schlecht passenden Patches). Im vorgestellten Beispiel liegt die Standardabweichung im Mittel bei 2800, die Minimalkosten zwischen 730 und 890. Die ermittelten Gewichte betragen damit für die Einzelframes im Schnitt 0.95, sind also sehr hoch, was aufgrund der subjektiv gut erscheinenden Qualität der gefundenen Patches auch gerechtfertigt sein mag. Bei dieser starken Gewichtung resultiert das Auffüllen jedoch quasi in einer Mittelung der gefunden Patches und ist dadurch durchaus kritisch zu betrachten (siehe Abschnitt 4.3.3). Abbildung 4.13 zeigt ein weiteres Beispiel der Patchfindung und -auffüllung. (Histogramme wiederum nur für die untere Reihe abgebildet.)

Alle Farb- bzw. Tiefenwerte der Nachbarframes an den gefundenen Positionen werden nun, jeweils multipliziert mit ihrem Gewicht, aufsummiert und durch die Summe aller Gewichte geteilt. Damit erhält man anschließend die fertigen Farb- und Tiefenwerte, die sich jeweils mit einer gewissen Wertigkeit aus den Referenzframes zusammensetzen und in den Zielpatch eingefügt werden können. Abschließend können im Maskenbild an den aufgefüllten Stellen die Pixel von weiß auf schwarz gesetzt werden, um zu markieren, dass dieser Bereich fertig ist. Durch darauf folgendes Zählen der verblie-

benen Maskenpixel ist des Weiteren bekannt, ob das Auffüllen beendet ist oder ob eine neue Iteration gestartet werden muss. Ist dies der Fall, so ergibt sich ein neuer unbekannter Bereich. Für die Randpixel werden erneut Prioritäten errechnet und ein neuer Schleifendurchgang kann ausgeführt werden.

Refinement

Das Refinement soll die bislang durch exemplarbasierte Auffüllung erhaltenen Frames noch einmal optimieren, um eine bessere Zeitkonsistenz zu erhalten und kleinere Fehler zu verbessern. Schon bei der vorherigen Auffüllung wurde, um dies zu erreichen, der Suchraum des Inpainting auf die Nachbarframes ausgedehnt und aus den erhaltenen Informationen eine gewichtete Schätzung der aufzufüllenden Bereiche gewonnen, die eine gewisse Angleichung der Bilddaten zwischen den Frames erzeugt. Die benötigten Frames liegen bereits in der angesprochenen *hashmap* von Frameobjekten vor und brauchen nicht mehr geladen zu werden, um im Folgenden als erstes den Bereich zu ermitteln, welcher die Aufdeckungen umschließt. Um dies zu erreichen, wird auf dem Maskenbild der Aufdeckungen des aktuellen Frames nach denjenigen (weißen) Pixeln gesucht, die am weitesten oben links und unten rechts liegen. Das resultierende Rechteck, welches dadurch aufgespannt wird, bildet den globalen Patch für die Hintergrundbereich-Suche (vgl. 4.3.1 Refinement). Dieser Patch wird nun analog zur lokalen Suche im vorherigen Abschnitt über die Referenzframes (in diesem Fall alle synthetisierten Frames) geschoben, um die Position des besten Match zu ermitteln (vgl. Gleichung 4.18). Dabei ist zu beachten, dass bei dem Vergleich nur alle Hintergrundpixel in den Vergleich mit einbezogen werden, die über einem gewissen Tiefenschwellwert liegen. So wird sichergestellt, dass möglichst relevante Ergebnisse für den Optimierungsprozess gefunden werden, da die zu optimierenden Bereiche ja im Hintergrund liegen. Die Kosten in den einzelnen Frames werden ermittelt, die Position der geringsten Kosten gespeichert und die Gewichtung errechnet. Anschließend werden die Pixel der ursprünglichen Auffüllungsregion über die gewichteten Informationen der Nachbarframes nach Gleichung 4.20 und 4.21 neu errechnet und entsprechend eingefügt (vgl. Abbildung 4.14).

4.3.3. Kritik

Nun soll abschließend noch einmal kritisch auf die von den Autoren Cheng et al. [17] vorgeschlagene und bereits als problematisch angesprochene Gewichtung der Nachbarframes eingegangen werden. Abgesehen davon, dass der Rechenaufwand für

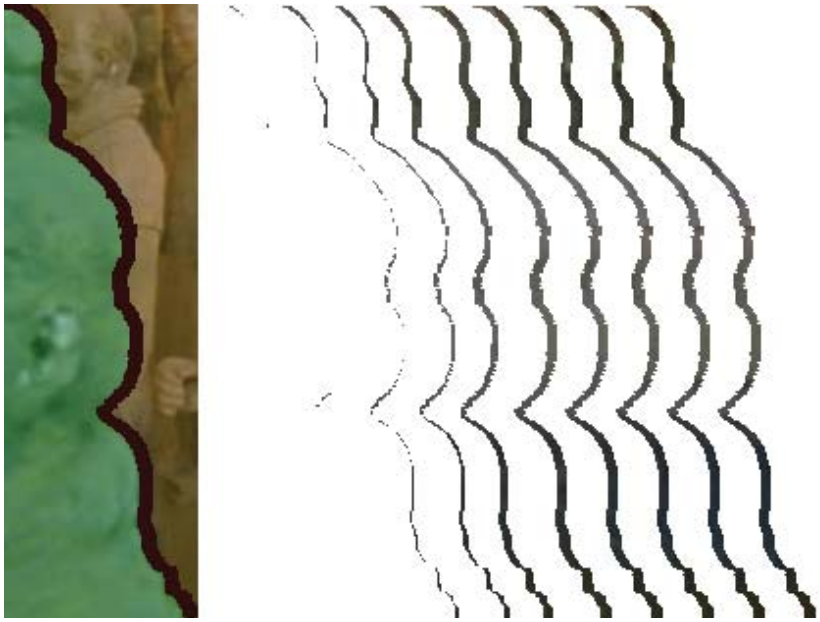


Abbildung 4.14.: Beispiel Refinement: Ausschnitt eines Frames (links) mit Vordergrundbereich (grün markiert), Aufdeckungsbereich (schwarz) und der zu vergleichende Teil (gelb markiert). Daneben die in den Nachbarframes gefundenen Bereiche für den Aufdeckungsbereich, die für die Optimierung in Betracht kommen.

eine repräsentative Aussage über die Standardabweichung aller Kosten sehr hoch ist (diese müssen schließlich erst einmal errechnet werden), gibt es noch einen anderen Aspekt zu beachten, der sich wie folgt darstellt. Bei der Betrachtung der möglichen Referenzframes ist davon auszugehen, dass es immer einigermaßen passende Patche zu finden gibt, da sich die Frames aufgrund ihrer Nachbarschaft sehr ähneln und sich damit in den meisten Fällen ein gutes Minimum im jeweiligen Bild finden lässt. Andererseits lässt sich feststellen, dass in einem natürlichen Bild, insgesamt gesehen, relativ heterogene Inhalte gegeben sind und damit eine breite Streuung der Werte auftritt bzw. ein große Standardabweichung zu sehen ist. Dies läuft darauf hinaus, was sich auch während der Implementierung beobachten ließ (vgl. Abbildung 4.15a), dass bei gegebener Berechnung nach Gleichung 4.17 für alle Patches hohe Gewichte zu erwarten sind. Wenn allerdings alle Bildinhalte der Nachbarframes hoch gewichtet werden, führt das letztendlich nur zu einer Mittelung und keiner sinnvollen Neubildung der Farbwerte. Außerdem werden selbst grob falsche Informationen im Vergleich zu ihrer geringen Aussagekraft zu hoch bewertet und führen zu falschen Ergebnissen in der Auffüllung (vgl. Abbildung 4.15b). Es scheint angebracht die Ermittelung der Gewichte zu überdenken und einen Weg zu finden ihre Aussagekraft zu verbessern.

(a) (b)

Abbildung 4.15.: Problematische Gewichtung: (a) Histogramm der Verteilung aller ermittelten Gewichte während der Auffüllung eines Frames - mehr als die Hälfte liegt über 0.8 (b) Bewusst eingebauter falscher Nachbarframe führt während der Patchsuche (aufzufüllender Patch links) zu einem ermittelten Gewicht des besten Patches (rechts) von "nur" 0.6. Das Ergebnis (mitte) führt zu sichtbaren Fehlern (aufgefüllter Bereich weist Rotfärbung auf).

Man könnte z.B. generell Patches mit sehr niedrigen Gewichten komplett aus der Auffüllung herauslassen. Dies zu verbessern bleibt zukünftigen Arbeiten überlassen.

Während der Implementierung des Refinements fiel weiterhin auf, dass die Tiefenwerte bei der Suche zwar entsprechend berücksichtigt wurden, jedoch nicht bei der anschließenden Auffüllung nach Gleichung 4.20 und 4.21. Dies führte in der Praxis dazu, dass Elemente des Vordergrunds das Ergebnis verfälschten und ein starker Schemen-Effekt zu beobachten war. Es wird daher vorgeschlagen, und dies wurde auch als Verbesserung in die Implementierung eingearbeitet, nur solche Pixel mit in die Auffüllung einfließen zu lassen, deren Tiefe unter dem bereits benutzten Schwellwert γ aus Gleichung 4.19 liegt (vgl. Abbildung 4.14 rechts: Nur Bildinformationen unterhalb eines definierten Tiefenschwellwertes werden angezeigt und zur Optimierung verwendet).

4.4. Besonderheiten der Implementierung

4.4.1. Verhalten am Bildrand

Ein in allen Bereichen der Bildbearbeitung auftauchendes Problem ist das Verhalten eines Algorithmus am Rand des Bildes oder allgemeiner am Rand definierter Bereiche. So kann z.B ein rechter oder linker Nachbarpixel fehlen, der für die Differenzenbildung während der Gradientenbestimmung benötigt wird (so. z.B. in Abbildung 4.2, die Pixel am Rand sind als undefiniert markiert). Je nach Problemstellung gibt es verschiedene Lösungsansätze. Eine Möglichkeit ist selbstverständlich den Algorithmus so zu programmieren, dass er die Randwerte nicht benötigt. Dies wird jedoch in seltensten Fällen zutreffen. Eher verbreitet ist in diesem Fall, das Bild so weit wie benötigt zu erweitern, entweder durch einfaches Fortführen der Randwerte oder komplizierter, durch Spiegeln des bekannten Randbereichs in den unbekannten Bereich, um nur einige der Möglichkeiten zu nennen. In der vorliegenden Arbeit wurden folgende Randbetrachtungen unternommen.

Gradientenberechnung Zu beachten ist hier, dass es zwei Randbereiche gibt, die berücksichtigt werden müssen. Neben dem normalen Bildrand gibt es auch den Rand $\delta\Omega$, also entlang des unbekannten bzw. maskierten Bereichs. Je nach implementierten Verfahren wurden unterschiedliche Möglichkeiten wahrgenommen.

Das erste Verfahren benutzt simple Vorwärts- und Rückwärtsdifferenzen, die am Bildrand nur jeweils bis zu den definierten Bereichen gehen. Am Rand des unbekannten Bereichs werden die Differenzen normal berechnet, jedoch wird anschließend der maskierte Bereich erweitert, um die fäschlicherweise eingeführten Kanten zu überdecken.

Beim Video-Inpainting wurde auf die in OpenCV zur Verfügung gestellte Funktion zur Sobel-Berechnung zurückgegriffen. Diese benutzt intern standardmäßig eine replizierende Randerweiterung, d.h. die letzten bekannten Bildwerte werden für den Bildrand übernommen. Am unbekannten Rand $\delta\Omega$ wurde eine spezielle eigene Form der Randreplikation benutzt. Alle bekannten Randpixel bekommen dabei den Gradientenwert ihres entsprechenden rechten bzw. oberen Pixelnachbarn.

Patchsuche am Rand Die Algorithmen erlauben auch maskierte Bereiche in der Bildrandzone, wobei damit alles gemeint ist, was in einem Bereich liegt, der so breit

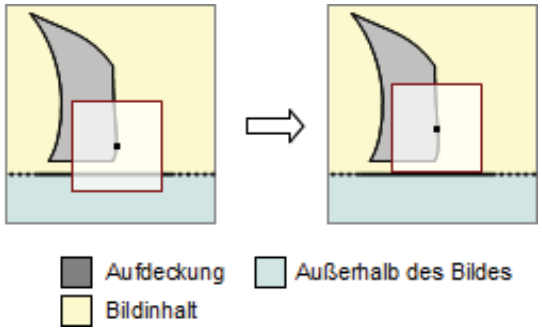


Abbildung 4.16.: Patchsuche am Bildrand: Beispiel der Verschiebung

ist wie ein Patch groß sein kann. Wird nach der Prioritätsberechnung festgestellt, dass der nächste Patch in diesem Bereich liegen würde, wird der Patch soweit ins Bild verschoben, dass er wieder vollkommen im Bildbereich liegt (siehe Abbildung 4.16). In beiden Verfahren wird keine Patchsuche auf dem Rand $\delta\Omega$ zugelassen, so dass es dort keiner speziellen Behandlung bedarf.

4.4.2. Rechenzeitoptimierung

Wie schon erwähnt, ist die Suche nach passenden Patches der rechenintensivste Teil. Jeder Patch muss einmal komplett über den Suchbereich geschoben werden, um die Differenzen zu ermitteln. Aktuelle Rechner sind in der Regel mit Mehrkernprozessoren ausgestattet, von denen ein rein sequentieller Algorithmus zum Patchvergleich jedoch nur einen Kern nutzen kann. Eine Parallelisierung bietet sich also an. In diesem Sinne wurde die Berechnung auf mögliche Ansatzpunkte untersucht und folgende Änderungen implementiert.

Parallelisierung Zu beachten ist bei einer Parallelisierung, dass keine Situationen erzeugt werden, in denen gleichzeitig auf bestimmte Daten von verschiedenen Seiten aus zugegriffen wird und diese dabei in Abhängigkeit von sich selbst verändert werden. Dies wird im vorliegenden Fall mittels einer Parallelisierungstechnik namens “map reduce” sichergestellt. Dabei wird in der “map”-Phase ein großes Problem in mehrere Teilprobleme zerlegt, dessen Ergebnisse dann in der “reduce”-Phase wieder zusammengeführt werden. Im konkreten Fall wird dies auf die Patchsuche angewandt. Das Hauptproblem ist hier definiert als das Finden eines vergleichbaren Patches im Bild. Dies wird zerlegt in die Suche eines passenden Patches in Teilbereichen des Bildes. Wenn die Ergebnisse der Teilbereiche vorhanden sind, werden diese untereinander verglichen und das beste Ergebnis ermittelt. Je nachdem wieviele Recheneinheiten

vorhanden sind, können also genauso viele Teilbereiche eines Bildes gleichzeitig durchsucht werden, was eine annähernd lineare Verbesserung der Rechenzeit mit sich bringt.

Einschränkung des Suchraumes Neben der Parallelisierung lässt sich die Rechenzeit auch leicht durch eine Reduzierung des Suchraumes verbessern. Während sich im Falle eines Einzelbildes statistisch betrachtet nur wenig über mögliche bevorzugte Bereiche der Patchsuche sagen lässt, sieht das bei der Multiframesuche schon anders aus. Es ist sehr viel wahrscheinlicher, dass gute Kandidaten auch im lokalen Umfeld des zu vergleichenden Patches in den anderen Frames liegen. Gerade im Bezug zu entstandenen Aufdeckungen der 3D-Stereosynthese bei Videos ist davon auszugehen, dass für diese in einigen der Nachbarframes die gesuchten Informationen in der Nähe liegen. Objekte, die eben noch verdecken, haben sich weiterbewegt und geben die gesuchten Hintergrundinformationen frei. Aus diesem Grund wurde im Video-Inpainting-Algorithmus der Suchraum auf einen vom Nutzer festgelegten Bereich um den aktuell am höchsten priorisierten Pixel beschränkt. Je nach Bildgröße wurden vom Autor Größen zwischen 50 und 200 Pixel benutzt.

5. Auswertung

Nachdem beide Verfahren im letzten Kapitel in der Theorie und der praktischen Implementierung vorgestellt wurden, sollen nun die Ergebnisse der Anwendung auf konkrete Bilder und Videos präsentiert werden. Diverse Einzelbilder werden im ersten Teil auf ihre glaubhafte und natürliche Auffüllung bei ausgesuchten exemplarbasierten Verfahren gegenüber dem hier vorgestellten gradientenbasierten Algorithmus untersucht, um dem Leser einen Eindruck der Möglichkeiten und Grenzen des Inpaintings zu verschaffen. Obwohl das erste Verfahren nicht speziell für Video ausgelegt ist, soll anhand einer Video-plus-Tiefe-Sequenz der Algorithmus auch auf seine Tauglichkeit für eine zeitkonsistente Auffüllung von 3D-Stereosynthese-Sequenzen getestet werden und so auch den Übergang zum zweiten Abschnitt des Kapitels ermöglichen. Anhand der dort untersuchten Videosequenzen werden die Resultate des zweiten Verfahrens vorgestellt und insbesondere auf die Einhaltung der geforderten Zeitkonsistenz geachtet. Eine Diskussion der Parameter soll den zweiten Teil abschließen.

Das Ziel von Inpainting ist, wie bereits erläutert, die für die normale Wahrnehmung ansprechende Auffüllung der beschädigten, fehlenden oder zu entfernenden Bereiche. Daher soll an dieser Stelle, wie in der Literatur üblich, die Bewertung aufgrund einer individuellen Beurteilung der Ergebnisse erfolgen. Aufgrund des textuellen Charakters dieser Arbeit ist eine Darstellung der Videos nicht möglich. Es wird jedoch, soweit möglich, anhand einzelner Frames auf hervorzuhebende Punkte eingegangen und eine subjektive Beurteilung der Videos abgegeben.

5.1. Gradientenbasierte Einzelbildauffüllung

Der Algorithmus von Shen et al. [59] wird im ersten Teil dieses Abschnitts auf verschiedene, aus der Inpainting-Literatur bekannte, natürlich strukturierte Abbildungen angewendet, um seine Funktionsfähigkeit zu demonstrieren und die Resultate zu diskutieren. Des Weiteren sind, soweit vorhanden, Ergebnisse von anderen Verfahren zum Vergleich danebengestellt, unter anderem auch eines klassischen diffusionsbasierten

Verfahrens. Es werden jeweils in einer Tabelle relevante Parameter wie Bildgröße, Inpaintingbereich, Patchgröße und Zeitdauer der Auffüllung mit Exemplaren aufgezählt. Die Zeitdauer zum Lösen der Poisson-Gleichung betrug aufgrund der gering besetzten Matrix in allen Fällen unter einer Sekunde und wird an dieser Stelle nicht weiter aufgeführt. Am Ende dieses Abschnitts wird überleitend der Algorithmus in Anwendung auf ein 3D-Stereosynthese-Resultat betrachtet.

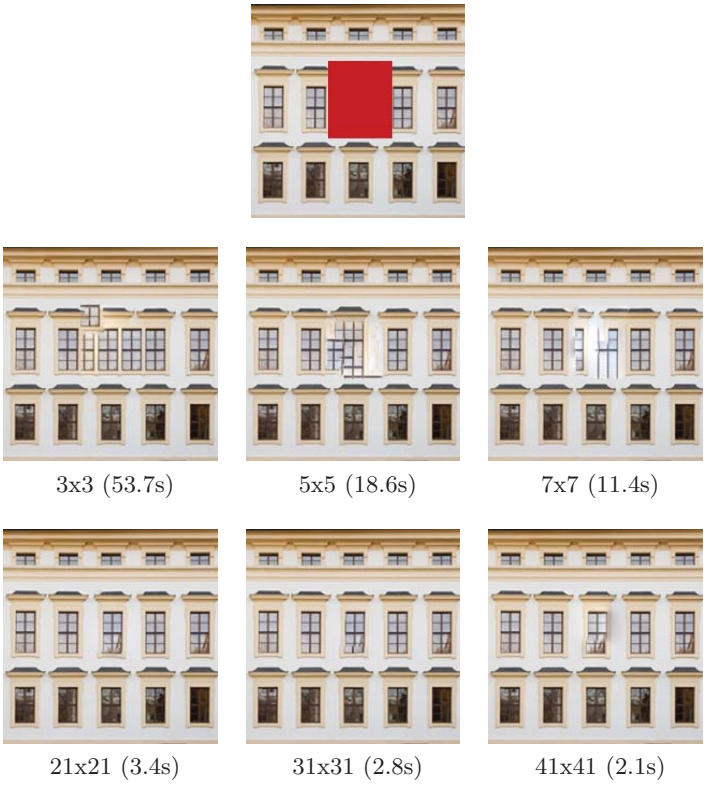


Abbildung 5.1.: Auswirkungen der Patchgröße. Bildquelle: Internet[1]. Die Bildunterschriften zeigen die jeweils genutzten Patchgröße sowie in Klammern die benötigte Zeit für das Ergebnis. [Bildgröße 200 x 200, unbekannter Bereich 60 x 72]

Eine konkrete Patchgröße wird bei vielen ähnlichen Verfahren in der Literatur nicht angegeben oder bestenfalls vage mit Texelgrößen assoziiert (Ein Texel ist ein Textur-Element und so wie ein Bild aus einer Menge von Pixeln besteht, so setzt sich eine Textur aus einer Menge von Texeln zusammen.) (vgl. [20, 63]). Die bei den vorgestellten Ergebnissen benutzte Patchgröße konnte daher nur ebenso empirisch ermittelt werden. Da sie jedoch ein entscheidender Faktor für die Qualität und Rechenzeit des

[1] `http://www.10ravens.com/textures/texture-buildings-00247-01-13948.html`

Algorithmus ist, soll dieser Parameter vor der Vorstellung der Ergebnisse noch einmal gesondert betrachtet werden. Man betrachte dazu Abbildung 5.1. Wie zu erkennen ist, sind kleine Patchgrößen (3x3 bis 7x7) nicht in der Lage größere Strukturzusammenhänge wiederherzustellen. Mit einer Größe von 21x21 Pixeln, was ungefähr der Breite eines Fensters in dem dargestellten Bild entspricht, ist die Auffüllung dagegen sehr glaubhaft. Bei den Werten darüber (31x31 und 41x41) hingegen ist der Patch schon wieder zu groß - es wird zuviel Inhalt auf einmal aufgefüllt und der Algorithmus hat keinen Spielraum für Zusammenhänge kleinerer Strukturen. Es kann außerdem vorkommen, dass dabei Informationen eingezeichnet werden, die in dem Kontext nicht mehr passen, aber aufgrund der Patchgröße noch mitgenommen wurden. Es lässt sich weiterhin ablesen, dass mit steigender Patchgröße die Rechenzeit sinkt. Dies ist logisch, da bei jeder Iteration weniger Differenzberechnungen ausgeführt werden müssen. Die Wahl der geeigneten Patchgröße ist somit auch immer ein Kompromiss zwischen gewünschter Qualität und toleriertem Zeitaufwand. Wie bereits eingangs erwähnt ist die optimale Patchgröße für jedes Bild neu zu ermitteln. Eine Automatik zu finden ist wünschenswert und sollte das Ziel künftiger Arbeiten sein. Im Folgenden nun die erhaltenen Ergebnisse.

Für das erste Verfahren wurden fünf Abbildungen ausgewählt. Zur Übersicht vorab zusammengefasst die wichtigen Kennzahlen der ausgewerteten Abbildungen, die man auch noch einmal in den Einzelabschnitten der Auswertung in den Tabellen 5.2 bis 5.6 aufgelistet findet. In der Zusammenfasung sieht man insbesondere auch die während der Auswertungen ermittelten günstigsten Patchgrößen.

Name	**Größe**$_{Bild}$	**Bereich** Ω	**Bildanteil**	**Größe**$_{Patch}$	**Dauer**
Baseball	256x170 px	05.342 px	~12 %	31x31 px	04 sec
Bungee	342x512 px	21.780 px	~12 %	27x27 px	80 sec
Elefant	384x256 px	14.532px	~15 %	41x41 px	18 sec
Jeep	374x266 px	17.163 px	~17 %	21x21 px	33 sec
Tonsoldaten	576x352 px	02.283 px	~02 %	07x07 px	51 sec

Tabelle 5.1.: Zusammenfassung der Eigenschaften und Parameter der folgenden ausgewerteten Abbildungen.

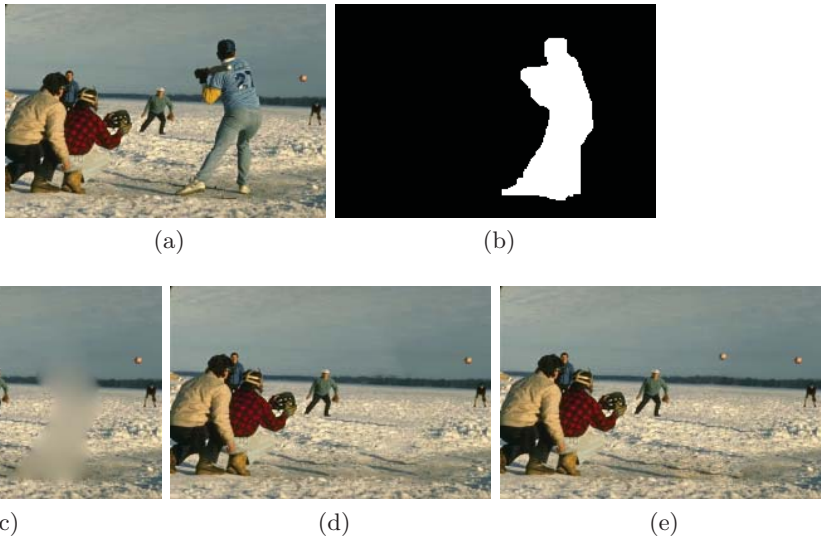
5.1.1. Abbildung Baseball

Bildgröße (b x h)	Bereich Ω	Bildanteil	Patchgröße	Inpaint-Zeit
256 x 170 px	5.342 px	~12 %	31x31 px	4 sec

Tabelle 5.2.: Bildeigenschaften und Parameter des Inpainting für Abbildung “Baseball”

Die Abbildung “Baseball” zeigt eine Szene im Freien. Der zu maskierende Bereich hat einen großen vertikalen Umfang und verläuft durch mehrere verschieden texturierte Gebiete. Verschiedene Strukturen aufgrund des Schnees und der Schatten sind zu beachten. Erschwerend kommt hinzu, dass die Jacke der einen und die Hose der anderen linken Person sehr ähnliche Farbwerte zum Schnee aufweisen. Dies macht den Algorithmus sehr anfällig für falsche Auffüllungen und erschwert das Finden einer guten Patchgröße. Die hier dargestellte ist mit 31x31 relativ groß im Vergleich zum Bild und dem Auffüllbereich, bringt jedoch den Vorteil der schnellen Berechnung mit sich. (Größere Darstellung und ein weiteres Beispiel im Anhang).

Wie zu erkennen ist, konnte der maskierte Bereich erfolgreich und plausibel aufgefüllt werden (siehe Abbildung 5.2e). Farben und Konturen sind visuell ansprechend fortgeführt worden, sowohl bei den Übergängen vom Schnee zur Baumreihe als auch von den Bäumen zum Himmel. Die kleinen Details im Schnee sehen authentisch aus und nach Meinung des Autors wurden auch die Schatten besser als in alternativen Inpaintings (siehe Alternative mit globaler Optimierung in Abbildung 5.2d) vervollständigt. Hinzu kommt die nicht unwesentlich höhere Komplexität bei Komodakis [38], die mit einer längeren Laufzeit einhergeht (Minuten im Vergleich zu Sekunden beim implementierten Algorithmus des Autors). Interessant ist, dass der vorgestellte Algorithmus die Auffüllung des Himmels so gewählt hat, dass auch noch einmal der Ball mit eingefügt wurde, was mit der großen Patchgröße zusammenhängt. Dies wiederum jedoch in einer Weise, dass das Bild für einen ahnungslosen Betrachter durchaus als visuell plausibel wahrgenommen wird. Bei der Komodakis-Variante hingegen sind bei näherer Betrachtung Farbsprünge am Rand des Inpaint-Bereichs erkennbar. Die Variante aus Abbildung 5.2c in Anlehnung an Bertalmio [9] ist zwar in der Lage die Farbgebung einzuhalten, bleibt jedoch stark verschwommen und ohne jede Struktur.



(a) (b)

(c) (d) (e)

Abbildung 5.2.: Resultat des gradientenbasierten Inpainting im Vergleich: (a) Original (b) Maske (c) traditionelle diffusionsbasierte Methode ähnlich [9] (d) Ergebnis durch globale Optimierung nach Komodakis [38] (e) Ergebnis der implementierten Gradientenmethode

5.1.2. Abbildung Bungee-Jumper

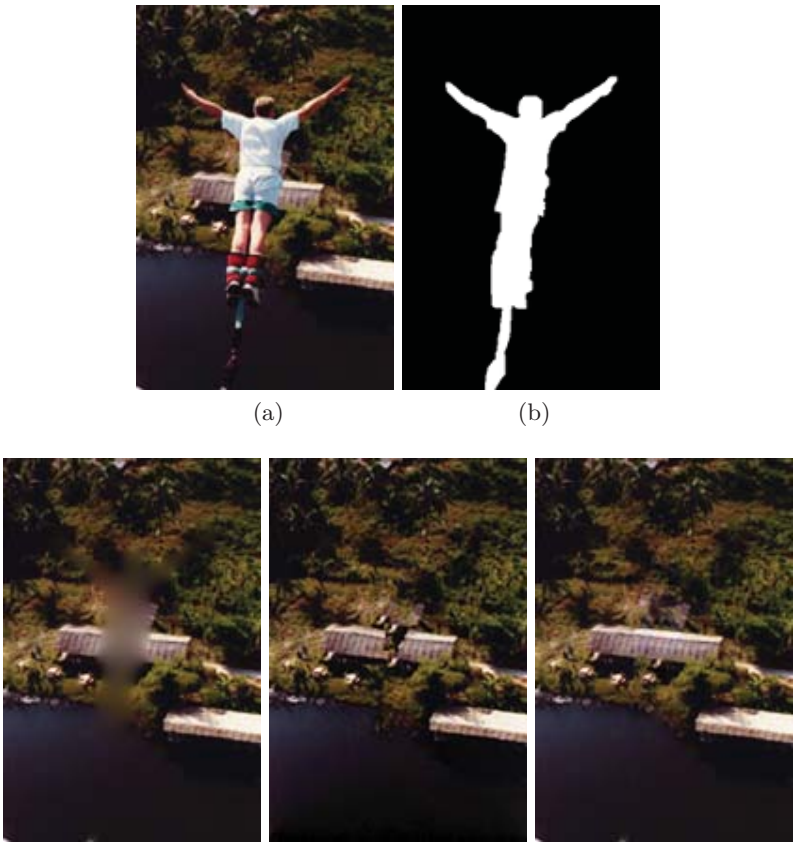
Bildgröße (b x h)	**Bereich Ω**	**Bildanteil**	**Patchgröße**	**Inpaint-Zeit**
342 x 512 px	21.780 px	~12 %	27x27 px	80 sec

Tabelle 5.3.: Bildeigenschaften und Parameter des Inpainting für Abbildung “Bungee”

Die Abbildung “Bungee-Jumper” zeigt eine der bekanntesten Szenen aus der Inpainting-Literatur. Der zu maskierende Bereich verläuft durch einen Großteil des Bildes und insbesondere durch eine stark strukturierte Dächeransicht, die es gilt visuell ansprechend aufzufüllen. Das Bild ist vergleichsweise groß, daher wurde auch ein größere Patchgröße gewählt, die dennoch im Rahmen der Dachstruktur bleibt. Aufgrund des großen Auffüllbereichs ist die Rechenzeit relativ lang. (Größere Darstellung im Anhang).

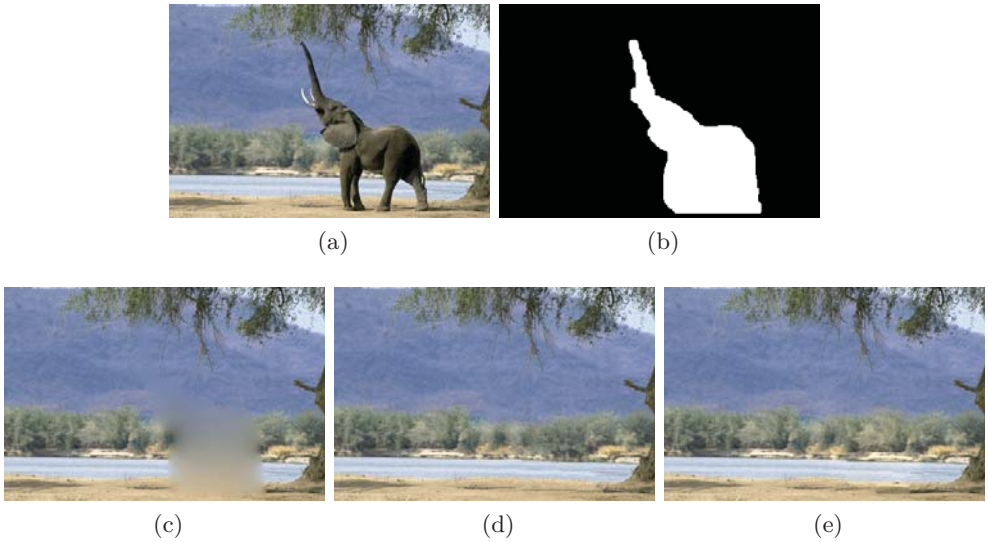
Wie zu erkennen ist, konnte der Algorithmus erfolgreich die Struktur des Daches vervollständigen (siehe Abbildung 5.3e). Die entsprechenden Kanten wurden gut vervollständigt und visuell plausibel gefüllt. Der Uferbereich wurde im Vergleich zu Criminisi [20] (siehe 5.3d) glaubwürdiger wiederhergestellt und Dank der Gradien-

teninterpolation ist anders als beim einfachen exemplarbasierten Inpainting keine Blockbildung erkennbar. Der Dschungelbereich lässt nicht vermuten, dass er computergeneriert ist. Nur an der oberen Dachkante sind leichte Doppelungen festzustellen, was vermutlich daran liegt, dass nicht genügend Bildinhalte vorhanden waren, um diese Stelle adäquat zu vervollständigen. Die Variante aus Abbildung 5.3c in Anlehnung an Bertalmio [9] kann das Bild nicht gut auffüllen, zu stark ist die Weichzeichnung des diffusionsbasierten Ansatzes.



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

Abbildung 5.3.: Resultat des gradientenbasierten Inpainting im Vergleich: (a) Original (b) Maske (c) traditionelle diffusionsbasierte Methode ähnlich [9] (d) exemplarbasierte Methode ähnlich Criminisi [20] (e) Ergebnis der implementierten Gradientenmethode



(a) (b)

(c) (d) (e)

Abbildung 5.4.: Resultat des gradientenbasierten Inpainting im Vergleich: (a) Original (b) Maske (c) traditionelle diffusionsbasierte Methode ähnlich [9] (d) Methode nach Drori [24] (e) Ergebnis der implementierten Gradientenmethode

5.1.3. Abbildung Elefant

Bildgröße (b x h)	**Bereich Ω**	**Bildanteil**	**Patchgröße**	**Inpaint-Zeit**
384 x 256 px	14.532px	~15 %	41x41 px	18 sec

Tabelle 5.4.: Bildeigenschaften und Parameter des Inpainting für Abbildung “Elefant”

Die Abbildung “Elefant” zeigt eine Szene im Freien. Der zu entfernende Elefant, der komplett maskiert wird, überlagert mehrere verschieden texturierte Gebiete, die adäquat aufgefüllt werden müssen. Unterschiedlich starke Strukturlinien, die sich durch den Übergang von Sand zu Wasser usw. ergeben, müssen korrekt fortgeführt werden. Für dieses Bild wurde ebenfalls eine möglichst großer Patch ermittelt, der dennoch in der Lage ist das Bild natürlich aufzufüllen, um die Rechenzeit zu verbessern. (Größere Darstellung und ein weiteres Beispiel im Anhang).

In Abbildung 5.4e ist das erfolgreiche Ergebnis des implementierten Verfahrens zu sehen. Die einzelnen Texturbereiche von Berg, Wasser und Sand konnten wiederhergestellt werden, mithilfe der Gradienteninterpolation ohne erkennbare Blockbildung. Die Übergänge am Wasser als auch andere für den Betrachter wichtige Strukturlinien, sind adäquat aufgefüllt worden. Im Vergleich mit dem alternativen Inpainting nach

Drori [24] (siehe Abbildung 5.4d) fällt auf, dass die Schattenlinie im Sandbereich nicht ganz so sauber aussieht, was dem normalen Betrachter jedoch nicht weiter negativ ins Auge fällt. Das Verfahren nach Drori ist bekannt für seine guten Ergebnisse bei solchen Problemstellungen, jedoch aufgrund einer komplizierten Patchsuche mehrere Größenordnungen rechenaufwendiger (bis zu 2 Stunden nach eigenen Angaben im Vergleich zu 18 Sekunden des vorgestellten Algorithmus). Die Variante aus Abbildung 5.2c in Anlehnung an Bertalmio [9] ist, wie schon vermutet, nicht in der Lage Struktur zu vollenden und von einer starken Weichzeichnung geprägt.

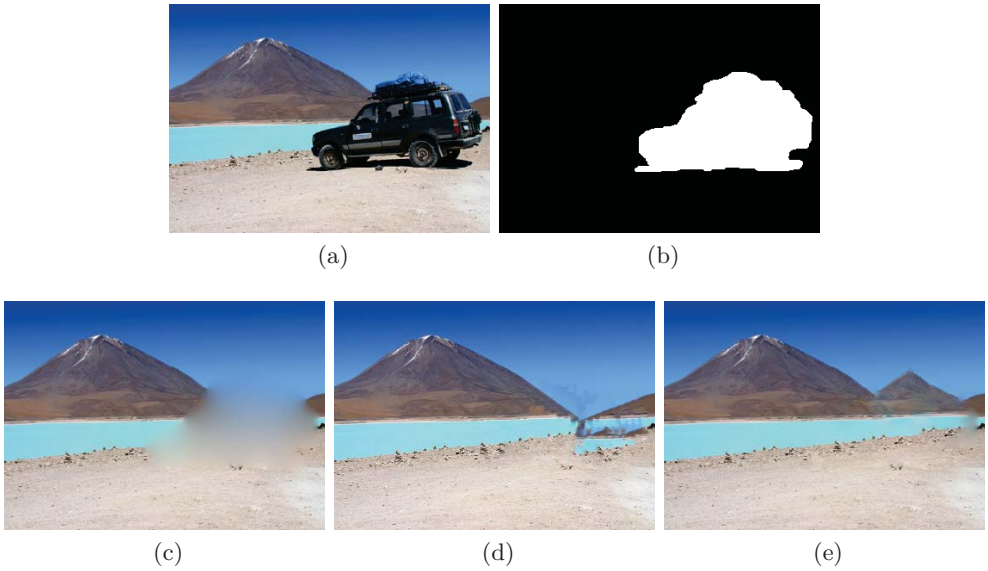
5.1.4. Abbildung Jeep

Bildgröße (b x h)	**Bereich Ω**	**Bildanteil**	**Patchgröße**	**Inpaint-Zeit**
374 x 266 px	17.163 px	~17 %	21x21 px	33 sec

Tabelle 5.5.: Bildeigenschaften und Parameter des Inpainting für Abbildung “Jeep”

Die Abbildung “Jeep” ist ebenfalls eine Szene in der Natur. Ähnlich wie bei der Abbildung “Elefant” soll das Fahrzeug komplett entfernt werden. Dass der maskierte Bereich recht weit am rechten Rand liegt, macht eine ansprechende Auffüllung unter Umständen anspruchsvoll, da wenig umliegende Informationen vorhanden sind. Des Weiteren führen an dieser Stelle die Strukturlinien zusammen, die durch den Übergang von Wasser zu Berg bzw. Sand entstehen, was das Finden eines passenden Patches ebenfalls erschweren kann. Neben diesen Strukturlinien kommt erschwerend eine für den Betrachter prägnante Strukturlinie von Berg zu Himmel dazu, die es gilt hinreichend zu verbinden. (Größere Darstellung im Anhang).

Die erfolgte Auffüllung ist in Abbildung 5.5e zu sehen. Wie zu erkennen, ist das Ergebnis nicht mehr ganz so gut wie bei den vorherigen Resultaten, deutet jedoch auch beispielhaft auf ein bekanntes Problem des exemplarbasierten Inpainting hin. Aufgrund des restriktiven Charakters der Auffüllung können, wie schon weiter oben beschrieben, einmal getroffene Fehlentscheidungen nicht mehr rückgängig gemacht werden. Auch in diesem Fall wurden einige ungünstige Patches ausgewählt, was z.B. zu dem sichtbaren Artefakt am rechten Bildrand führt. Im Vergleich zum reinen patch-basierten Inpainting nach Criminisi [20] (siehe Abbildung 5.5d) lässt sich jedoch eine deutlich gesteigerte Bildqualität feststellen, unter anderem, da die gradienten-geleitete Interpolation viele der Problembereiche kaschieren oder zumindest abschwächen kann. Das diffusionsbasierte Verfahren in Anlehnung an Bertalmio [9] (siehe Abbildung



(a) (b)

(c) (d) (e)

Abbildung 5.5.: Resultat des gradientenbasierten Inpainting im Vergleich: (a) Original (b) Maske (c) traditionelle diffusionsbasierte Methode ähnlich [9] (d) exemplarbasierte Methode ähnlich Criminisi [20] (e) Ergebnis der implementierten Gradientenmethode

5.5c) ist wie erwartet stark verschwommen. Die fehlende Struktur fällt vor allem beim fehlenden Übergang von Himmel zu Berg auf.

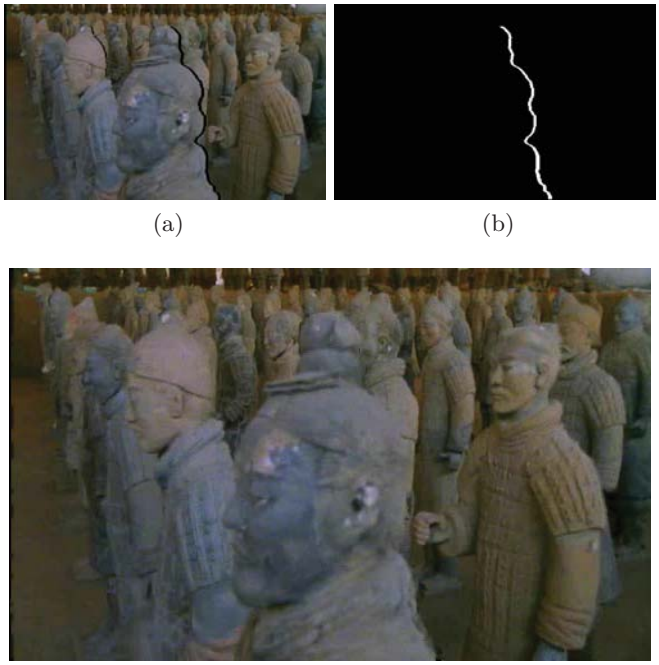
5.1.5. Abbildung Tonsoldaten

Bildgröße (b x h)	**Bereich Ω**	**Bildanteil**	**Patchgröße**	**Inpaint-Zeit**
576 x 352 px	2.283 px	~2 %	7x7 px	51 sec

Tabelle 5.6.: Bildeigenschaften und Parameter des Inpainting für Abbildung “Tonsoldaten”

Die Abbildung “Tonsoldaten” ist der erste Frame einer zu füllenden Bildfolge (Bild mit Aufdeckungen in Abbildung 5.6a und zu füllende Bereiche in Abbildung 5.6b) aus der Sequenz “Tonsoldaten” (siehe Abschnitt 5.2.1) und soll das Verfahren in Anwendung auf ein in der 3D-Stereosynthese erzeugtes Bild demonstrieren. Der maskierte Bereich setzt sich in diesem Fall aus den erhaltenen Aufdeckungen zusammen. Da von einer linken Quellkamera aus die rechte Ansicht synthetisiert wurde, sind die Aufdeckungen also an den rechten Seiten der durch die Tiefe voneinander abgesetzten Objekte (Tonsoldaten) zu finden. Je weiter die Tiefenunterschiede, umso breiter die Aufdeckung. Es wurde eine relativ kleine Patchgröße gewählt, um in den dünnen

Aufdeckungsbereichen individuell auf die lokale Umgebung zu reagieren. Dies erklärt auch die höhere Rechendauer. (Größere Darstellung im Anhang).



(a) (b)

(c)

Abbildung 5.6.: Inpaint eines stereosynthetisierten Bildes mit Aufdeckungen: (a) Original mit Aufdeckungen (b) Maske (c) Resultat der Auffüllung

Das Ergebnis, wie in Abbildung 5.7 zu sehen, ist dabei abhängig von den Prämissen des Verfahrens zu bewerten. Es wird erfolgreich vorhandene Struktur fortgesetzt (vgl. Abbildung 5.7a, obere Hervorhebung) und die Auffüllungen sind Dank der Gradienteninterpolation ohne erkennbare Block- oder Kantenbildung stimmig integriert. Für eine Einzelbildvervollständigung ist dies bis auf einige kleinere Artefakte durchaus präsentabel. Im Kontext der 3D-Videosynthese werden jedoch mehrere Bilder benötigt und jedes Bild wird unabhängig voneinander aufgefüllt. Der Algorithmus besitzt keinerlei Information über Tiefe oder zeitliche Veränderungen und die daraus resultierenden Abhängigkeiten der benötigten Farbinformationen (vgl. Abbildung 5.7a, untere Hervorhebung). Es wird somit jeder Frame an örtlich-zeitlich zusammenhängenden Positionen unterschiedlich aufgefüllt - je nach lokalen Entscheidungen des Algorithmus auf den einzelnen Frame bezogen, was zum erwarteten zeitinkonsistenz-bedingten

Flackern in der Wahrnehmung einer Videosequenz führt. Dies sieht man deutlich an den verschiedenartigen Auffüllungen, die für Frame 1-5 durchgeführt wurden (vgl. Abbildung 5.7b). Hier zeigen sich die Grenzen eines texturbasierten Verfahrens wie dem nach Shen et. al [59], das für Videosequenzen nicht hinreichend geeignet ist. Somit muss man für diese Aufgabenstellung spezifische Verfahren einsetzen, die die Zeitkonsistenz berücksichtigen, wie das Verfahren nach Cheng et. al [17].

(a)

(b)

Abbildung 5.7.: Inpaint eines stereosynthetisierten Bildes mit Aufdeckungen: (a) Ausschnitte aus dem Resultat: korrekte Strukturvervollständigung [oben], aber Auffüllung von links und rechts [unten] (b) Ausschnitte aus Frame 1-5 der "Tonsoldatensequenz" demonstrieren die zeitliche Inkonsistenz. Man beachte die unterschiedlichen Auffüllungen.

5.2. 3D-Videosynthese-Auffüllung

Der vorgestellte Algorithmus von Cheng et al. [17] wurde auf verschiedene Videosequenzen, die aus der 3D-Stereosynthese stammen, angewandt, um seine Funktionsfähigkeit zu untersuchen. Bei allen Sequenzen wird von einer linken Quellkamera ausgegangen und entsprechend werden mithilfe der eingangs erwähnten *imcube cinema* Software aus den linken Originalbildern und Tiefenkarten neben den korrespondierenden rechten Bildern und Tiefenkarten auch die Aufdeckungskarten für alle drei Sequenzen herausgerendert. Die erste Sequenz ist dem Paper der Autoren selber entnommen, in dessen verlinktem Begleitmaterial Zhang [79] Quellvideos (Bild und Tiefe) zu finden sind. Da der beschriebene Schritt zur Verbesserung der Tiefenkarten nicht Teil der Arbeit ist, bleiben diese nur von mittelmäßiger Qualität, was im Rahmen dieser Untersuchung jedoch ausreichend ist. Die anderen beiden Sequenzen sind von der *imcube labs GmbH* bereit gestellt. In einer Tabelle werden jeweils relevante Parameter wie Bildgröße, Aufdeckungsbereich, Patchgröße, Zeitdauer zur Auffüllung eines Frames (Index I), Zeitdauer des Refinements (Index R), Anzahl der genutzten Nachbarframes usw. genannt. Die Gesamtzahl der Frames bezieht sich dabei nur auf die für die finale Videosequenz genutzten Frames. Die Wahl der Patchgrößen ist wie bereits im vorherigen Abschnitt, empirisch erfolgt. Aufgrund der Unterschiede und Komplexität der Szenen ist eine Festlegung im Vorfeld schwer durchführbar. Unter anderem ist es hilfreich darauf zu achten, eine an die Aufdeckungen angepasste Patchgröße zu wählen (vgl. Abschnitt 5.2.4). Wie in Abschnitt 4.4 vorgeschlagen, wurden die Möglichkeiten zur Rechenzeitoptimierung genutzt. Neben einer vollen Auslastung der zur Verfügung stehenden Prozessorkerne[2], wurde, nach Einschätzung der Szenen durch den Autor, ein Suchraum von 101x101 Pixeln um den jeweils aktuellen Patch festgelegt. Dies reicht aus, um die interessanten Bildinformationen der Referenzframes einzuschließen und etwaige Kamera- oder Objektbewegungen zu berücksichtigen. Im letzten Abschnitt dieses Unterkapitels soll diesbezüglich dann noch einmal auf die Auswirkungen der Patchgröße sowie der Anzahl an Nachbarframes auf die Qualität und die daraus resultierende Rechenzeit eingegangen werden (siehe Abschnitt 5.2.4). Weiterhin sei darauf aufmerksam gemacht, dass in den Resultaten, bis auf die endgültigen Ergebnisse des Refinement, darauf verzichtet wurde, die ebenfalls aufgefüllten Tiefenkarten mit anzuzeigen, um nicht den Rahmen der Darstellung zu sprengen. Der interessierte Leser findet hierzu Beispiele im Anhang, neben einzelnen Resultaten in größerer Auflösung zur besseren Ansicht.Es folgen die Ergebnisse der Auswertung.

[2] AMD Athlon II X4 640 Prozessor mit 4 Kernen á 3 GHz

Das Verfahren von Cheng et al. [17] wurde auf drei Video-Sequenzen angewendet. Vorab eine Zusammenfassung der wichtigsten Kennzahlen, die sich auch in den einzelnen Abschnitten der Auswertung wiederfinden.

Name	**Framegröße (b x h)**	**Frameanzahl**	**Bereich Ω**
Tonsoldaten	576x352 px	36	~1.800 px
Soccer	1280x720 px	100	~4.500 px
Breakdancer	1280x720 px	51	~3.800 px

Tabelle 5.7.: Zusammenfassung der Eigenschaften der im Folgenden ausgewerteten Sequenzen.

Name	**Patchgröße**	**Nachbarframes**	**Zeit$_I$/ Frame**	**Zeit$_R$/ Frame**
Tonsoldaten	7x7 px	4	~100 sec	~15 sec
Soccer	9x9 px	4	~240 sec	~100 sec
Breakdancer	11x11 px	4	~300 sec	~700 sec

Tabelle 5.8.: Zusammenfassung der Parameter und der benötigten Rechenzeit der im Folgenden ausgewerteten Sequenzen.

5.2.1. Sequenz “Tonsoldaten”

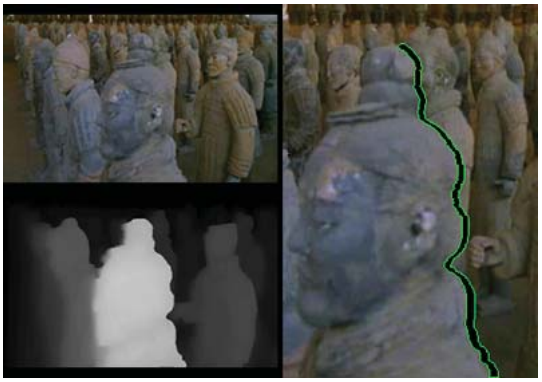


Abbildung 5.8.: Frame 0 der Sequenz “Tonsoldaten”: linke Ansicht (o.l.) mit Tiefenkarte (u.l.) und Ausschnitt der zugehörigen rechten Ansicht mit Aufdeckung (rechts)

Framegröße (b x h)	**Frameanzahl**	**Bereich Ω**	**Patchgröße**
576x352 px	36	~1.800 px	7x7 px
	Nachbarframes	**Zeit$_I$/ Frame**	**Zeit$_R$/ Frame**
	4	~100 sec	~15 sec

Tabelle 5.9.: Eigenschaften und Parameter des Inpainting für Sequenz “Tonsoldaten”

Die Sequenz “Tonsoldaten” stellt eine einfache horizontale Kamerafahrt von links nach rechts über die bekannten Tonsoldaten aus China dar. Der Hintergrund ist statisch, es gibt jedoch eine Vielzahl von Objekten mit unterschiedlichen Tiefen. Das vorliegende Material weist eine starke Kompression auf, was sich z.B. in zu erkennender Blockbildung äußert und auch die Farbgebung für den Algorithmus erschwerend beeinflusst. Das Hauptaugenmerk soll auf der großen Figur im Vordergrund liegen, da hier die größten Disparitäten und die auffälligste Bewegung liegt (siehe Abbildung 5.8).

Abbildung 5.9 zeigt die fertigen, aufgefüllten Ergebnisse der Frames 0 bis 5. Wie zu erkennen ist, sind die Auffüllungen durchaus stimmig, jedoch lässt der zeitliche Verlauf noch zu wünschen übrig. So ist z.B. zwischen Frame 0 und 2 am Bodenbereich ein Sprung der Farbwerte sichtbar, der sich auch stark in der Videoansicht bemerkbar macht. Abbildung 5.10 zeigt die durch das Refinement des Verfahrens erhaltenen verbesserten Ergebnisse in Farb- und Tiefenbildern. Wie zu sehen ist, konnten die vorher gewonnenen Frames in Bezug auf Bild- und Tiefeninhalte sowie deren zeitliche Konsistenz zwischen den Frames noch einmal gut verbessert werden. Bei Frame 0

Abbildung 5.9.: Ausschnitt des Ergebnisses der exemplarbasierten Auffüllung. Von links nach rechts Frame 0 bis 5. Grün markiert den Bereich der Auffüllung. Gute Füllergebnisse, Strukturen wie z.B. die Halskrause im Hintegrund wurden korrekt fortgesetzt. Am Boden sind temporale Artefakte im Frameverlauf erkennbar.

fällt eine fehlerhafte Auffüllung am Rücken der hinteren Figur auf, die so vorher nicht vorhanden war, aber durch die Einbeziehung der Nachbarframes integriert wurde. Bei der Betrachtung der endgültigen Videosequenz durch den Autor fällt an einigen Stellen eine Art Korona am Rand der Auffüllung auf, welche jedoch an den eingangs erwähnten mittelmäßigen Tiefenkarten liegt, bzw. auch an der daraus entstandenen Aufdeckungskarte. Einige Pixel, die zu einem Teil noch zum Vordergrund gehören, wurden dadurch in den Hintergrund verschoben und stören das Synthese-Resultat. Dies ließe sich jedoch durch eine manuelle Erweiterung des maskierten Bereichs verhindern. Im Vergleich zur reinen Auffüllung erscheint die optimierte Sequenz um einiges ruhiger, also zeitkonsistenter.

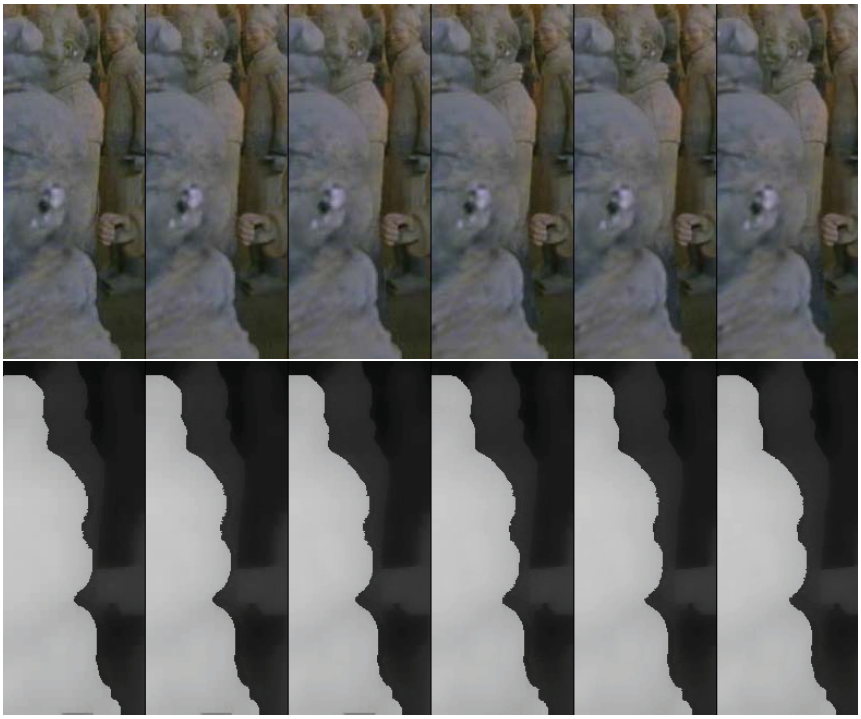


Abbildung 5.10.: Ausschnitt des Ergebnisses des Refinements der Sequenz “Tonsoldaten”. Von links nach rechts Frame 0 bis 5. Obere Reihe optimierte Bilder, untere Reihe optimierte Tiefenkarten. Die Frames wurden einander angeglichen und die zeitliche Inkonsistenz verringert.

5.2.2. Sequenz “Soccer”

Abbildung 5.11.: Frame 0 der Sequenz “Soccer”: linke Ansicht (u.l.) mit Tiefenkarte (o.l.) und Ausschnitt der zugehörigen rechten Ansicht mit Aufdeckung (rechts)

Framegröße (b x h)	**Frameanzahl**	**Bereich Ω**	**Patchgröße**
1280 x 720 px	100	~4.500 px	9x9 px
	Nachbarframes	**Zeit$_I$/ Frame**	**Zeit$_R$/ Frame**
	4	~240 sec	~100 sec

Tabelle 5.10.: Eigenschaften und Parameter des Inpainting für Sequenz “Soccer”

Die Sequenz “Soccer” stellt eine einfache Szene ohne Kamerabewegung und mit statischem Hintergrund dar. Die Tiefe des Bildes ist relativ einfach gehalten. Das Hauptaugenmerk liegt auf der Figur des Fussballspielers, dessen Bewegung ein kompliziertes Muster aus Aufdeckungen erzeugt (siehe Abbildung 5.11). Des Weiteren wirft er einen Schatten, der eine gewisse Dynamik in den Hintergrund bringt und die korrekte Auffüllung verkompliziert. Im Folgenden nun die Ergebnisse der Auffüllung und des Refinements.

Abbildung 5.12 zeigt je einen ersten Bildausschnitt der Ergebnisse von Frame 0 bis Frame 5 des Fußballers. Die Resultate sind durchwachsen: einerseits wurden die Farb- und Strukturinformationen an den meisten Stellen visuell ansprechend gefüllt, andererseits wurde an einigen Stellen die Schattenkante falsch fortgesetzt, was zum Flackern in der Videowahrnehmung führt (Beispiele sind mit roten Pfeilen markiert). An anderen Stellen wurden je nach Frame leicht divergierende Farbwerte eingesetzt, die ebenfalls die Videoqualität einschränken (Beispiele sind mit rotem Quadrat und Kreis markiert). Abbildung 5.13 zeigt einen zweiten Ausschnitt der Ergebnisse des Fußballers. Dargestellt sind hier die Frames 33 bis 35, die genau den Anfang des anspruchsvollen Wechsels von sichtbaren zu nicht sichtbaren Schatten wiedergeben. Wie zu erkennen ist, versagt der Algorithmus an dieser Stelle und es werden eher helle Bildinhalte aufgefüllt, was zu einer abrupten Änderung in der Schattendarstellung im Zeitverlauf führt. Abbildung 5.14 zeigt die Ergebnisse des Refinements für einen Ausschnitt des Bildes und den dazugehörigen Tiefenkarten. Zur besseren Einordnung in den Bildinhalt wurde auf den Rahmen um den zu optimierenden Bereich herum verzichtet. Wie zu erkennen, wurden die einzelnen Frames, soweit es die exemplarbasierte Auffüllungsgrundlage zulässt, optimiert und einander angeglichen. Die zeitliche Inkonsistenz, die sich im vorherigen Schritt durch falsche Auffüllungen z.B. an den Schattenkanten ergaben, wird so gut wie möglich auf den Zeitverlauf angepasst, um störende Artefakte in der Videodarstellung zu vermeiden. Die Tiefenkarten sind im Gegensatz dazu durchweg sehr gut wiederhergestellt worden, was zu einem gewissen Anteil auch daran liegt, dass diese recht simpel gehalten waren. Bei der Ansicht der

entstandenen Videosequenz durch den Autor fiel vor allem die fehlerhafte Schattenauffüllung ins Auge. Es lässt sich jedoch feststellen, dass durch den Optimierungsschritt der Gesamteindruck über den Zeitverlauf hinweg ruhiger geworden ist, also weniger Flackern auftritt und die Zeitkonsistenz zufriedenstellend ist.

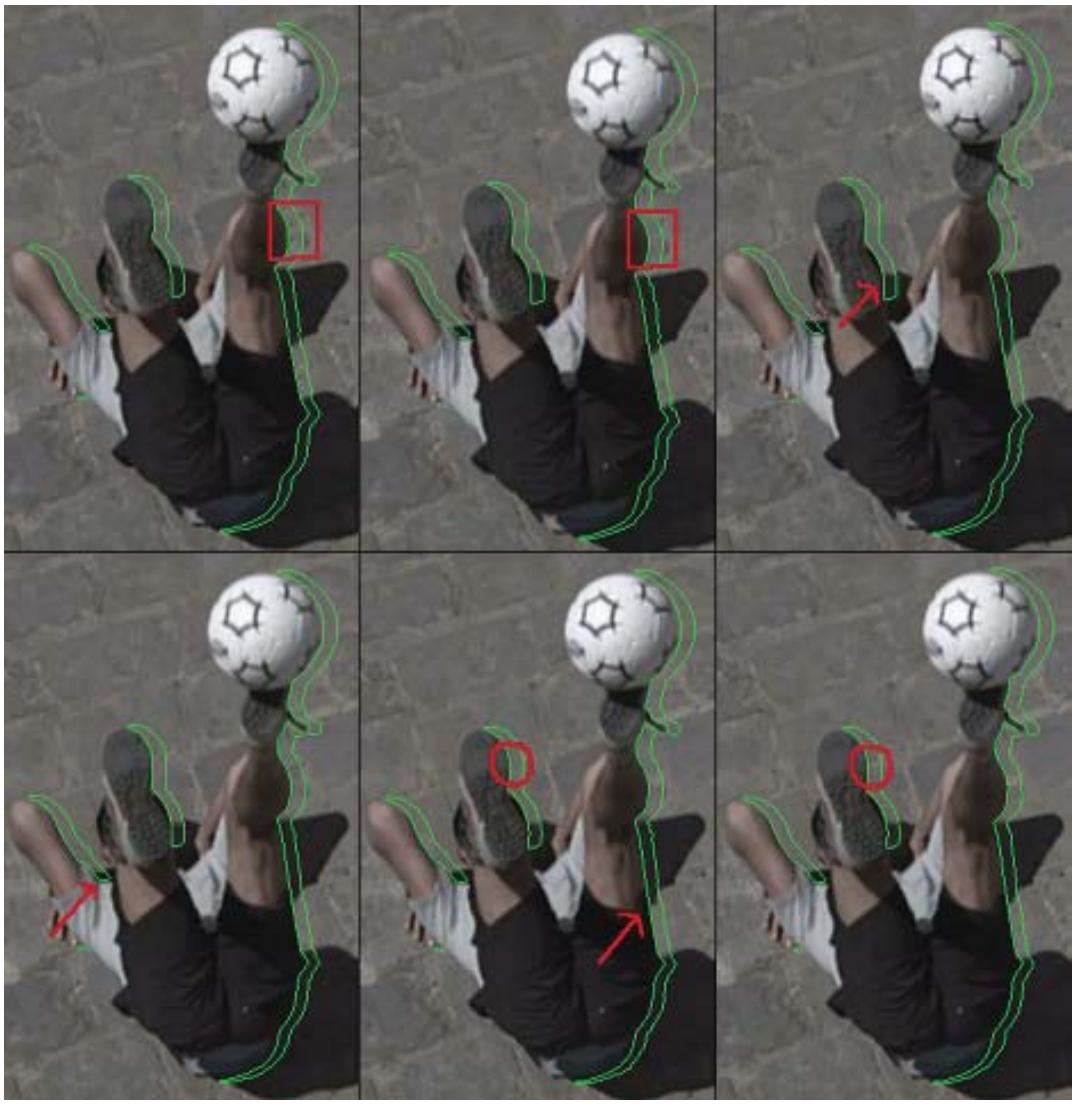


Abbildung 5.12.: Ausschnitt 1 des Ergebnisses der exemplarbasierten Auffüllung. Von links oben nach rechts unten Frame 0 bis 5. Grün markiert den Bereich der Auffüllung. Einerseits sind die Farb- und Strukturinformationen an vielen Stellen visuell ansprechend gefüllt worden, andererseits wird an diversen Stellen die Schattenkante falsch fortgesetzt, was zum Flackern in der Videowahrnehmung führt. Fehlerhafte Stellen sind mit roten Markierungen hervorgehoben.

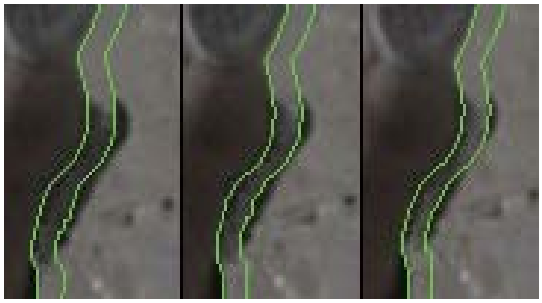


Abbildung 5.13.: Ausschnitt 2 des Ergebnisses der exemplarbasierten Auffüllung. Von links nach rechts Frame 33 bis 35. Grün markiert den Bereich der Auffüllung. Fehlerhaftes Ergebnis - der Schatten wird nicht korrekt gefüllt.

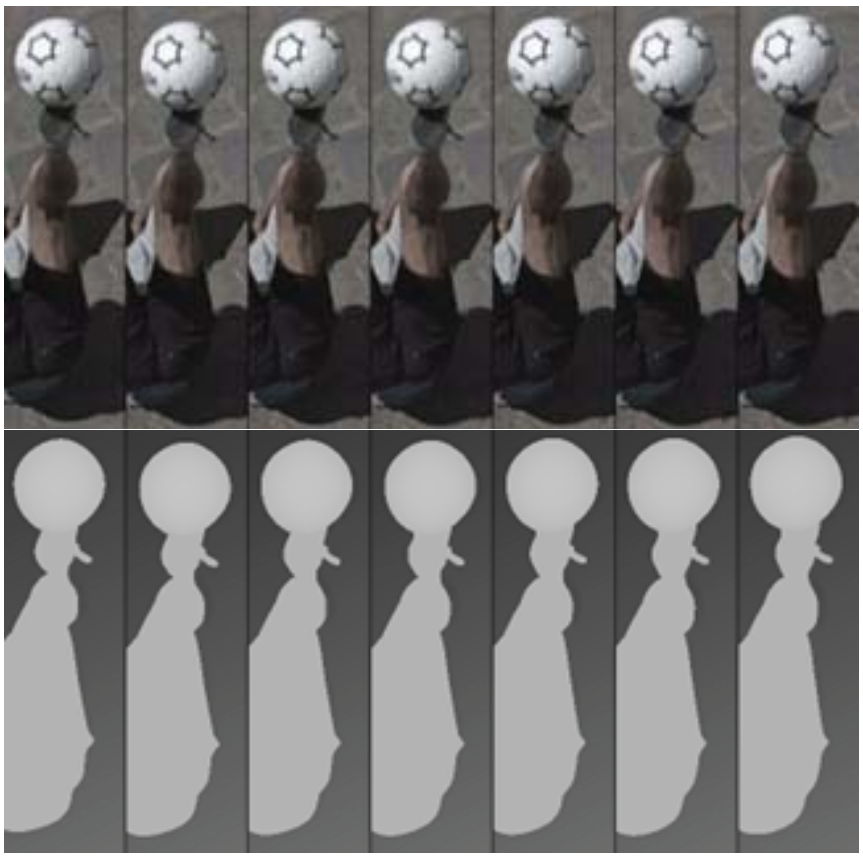


Abbildung 5.14.: Ausschnitt des Ergebnisses des Refinements der Sequenz “Soccer”. Von links nach rechts Frame 0 bis 6. Obere Reihe optimierte Bilder, untere Reihe optimierte Tiefenkarten. Die Frames wurden einander angeglichen und die zeitliche Inkonsistenz verringert.

5.2.3. Sequenz “Breakdancer”

Abbildung 5.15.: Frame 0 der Sequenz “Breakdancer”: linke Ansicht (u.l.) mit Tiefenkarte (o.l.) und Ausschnitt der zugehörigen rechten Ansicht mit Aufdeckung (rechts)

Framegröße (b x h)	**Frameanzahl**	**Bereich Ω**	**Patchgröße**
1280 x 720 px	51	~3.800 px	11x11 px
	Nachbarframes	**Zeit$_I$/ Frame**	**Zeit$_R$/ Frame**
	4	~300 sec	~700 sec

Tabelle 5.11.: Eigenschaften und Parameter des Inpainting für Sequenz “Breakdancer”

Die Sequenz “Breakdancer” stellt eine einfache Szene ohne Kamerabewegung und statischem Hintergrund dar. Die Tiefe des Bildes ist relativ einfach gehalten. Das Hauptaugenmerk liegt auf der Figur des Breakdancers, dessen Bewegung ein kompliziertes Muster aus Aufdeckungen erzeugt (siehe Abbildung 5.15). Des Weiteren besteht der Hintergrund aus einer komplexen Textur mit starkem Strukturanteil, die schon geringe Fehler bei der Auffüllung sichtbar macht.

Abbildung 5.16.: Ausschnitt 1 des Ergebnisses der exemplarbasierten Auffüllung. Von links nach rechts Frame 0 bis 4. Grün markiert den Bereich der Auffüllung. Gute Füllergebnisse, Strukturen wie z.B. das Geländer im Hintergrund wurden korrekt fortgesetzt. Nur an der Fahne sind Artefakte zu erkennen.

Abbildung 5.16 zeigt in einem ersten Ausschnitt der Ergebnisse von Frame 0 bis Frame 4 ein Detail des Breakdancers. Die Auffüllungen können im Großen und Ganzen als gut bezeichnet werden. Die meisten der vorhandenen Farben und Strukturlinien wurden korrekt fortgesetzt, bis auf den obersten Rand der Ballustrade, die vereinzelt falsche Kantenfortführungen enthält (z.B. in Frame 0).

Abbildung 5.17.: Ausschnitt 2 des Ergebnisses der exemplarbasierten Auffüllung. Von links nach rechts Frame 0 bis 4. Grün markiert den Bereich der Auffüllung. Mäßige Ergebnisse, die hellgraue Struktur am Kopf wurde nicht fortgesetzt und auch die Häusertextur ist teils sehr unterschiedlich eingezeichnet worden, was zu einem wahrnehmbaren Flackern in der Videodarstellung führt.

Abbildung 5.17 zeigt einen zweiten Ausschnitt der Ergebnisse von Frame 0 bis Frame 4 vom Kopf des Breakdancers. Wie zu sehen ist, wurde die Textur der Häuserwand

von rechts bis zum Kopf aufgefüllt, jedoch war der Algorithmus nicht in der Lage, die hellgraue Struktur korrekt nach unten fortzusetzen. Die einzelnen Auffüllungen unterscheiden sich mitunter stark, was in der Videoansicht als starkes Flackern auffällt. So wird z.B. der schon angesprochene graue Balken unterschiedlich weit fortgeführt. In Frame 2 (3.v.l) ist ebenfalls deutlich zu erkennen, dass die Häusertextur im Hintergrund im Vergleich zu den anderen Frames abweichend aufgefüllt wurde.

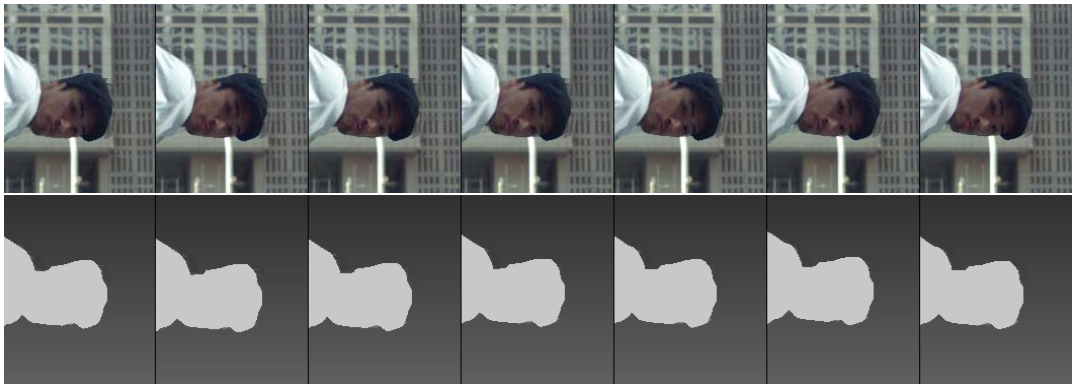
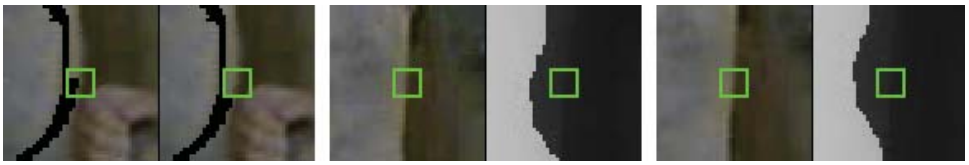


Abbildung 5.18.: Ausschnitt des Ergebnisses des Refinements der Sequenz “Breakdancer”. Von links nach rechts Frame 0 bis 6. Obere Reihe optimierte Bilder, untere Reihe optimierte Tiefenkarten. Die Frames wurden einander angeglichen und die zeitliche Inkonsistenz verringert.

Abbildung 5.18 zeigt die Ergebnisse des Refinements für einen Ausschnitt des Bildes und der entsprechenden Tiefenkarten. Zur besseren Einordnung in den Bildinhalt wurde auf den Rahmen um den zu optimierenden Bereich verzichtet. Wie zu erkennen, wurden die einzelnen Frames, soweit es die exemplarbasierte Auffüllungsgrundlage zulässt, gut optimiert und einander angeglichen. Die Häusertextur ist in allen Frames annähernd identisch. Bei näherer Betrachtung der fehlerhaften Fortführung der hellgrauen Struktur fällt auf, dass diese auf dem hier dargestellten Frameverlauf gut interpoliert wurde. In der Videodarstellung fällt dieses Element zwar immer noch als “falsch” auf, wird jedoch nicht mehr als störend flackernd wahrgenommen. Die Tiefenkarten sind im Endergebnis als gut einzuschätzen. Die grundlegende Tiefe ist vorhanden und ein oder zwei leicht abweichende Pixel sollten in der Tiefenwahrnehmung nicht auffallen. Der im Vergleich lange Zeitraum, der in dieser Sequenz für das Refinement gebraucht wird, liegt im relativ großen globalen Patch begründet, welcher in dieser Sequenz aufgrund der unterschiedlichen Aufdeckungen um einiges größer ausfällt als bei den anderen beiden Sequenzen (vgl. folgender Abschnitt 5.2.4).

(a) Patchgröße 43x43 px



(b) Patchgröße 15x15 px

Abbildung 5.19.: Auswirkung der Patchgröße [Sequenz Tonsoldaten, Frame 0]: Abgebildet von links nach rechts ist je ein beispielhafter zu füllender Patch, das aufgefüllte Ergebnis und die gefundenen Patches in der linken Ansicht sowie im ersten Nachbarframe (a) Ein zu großer Patch bezieht fälschlicherweise Informationen des Vordergrundobjektes mit ein und führt zu fehlerhafter Auffüllung. (b) Eine kleinere Wahl der Patchgröße, angepasst an die Aufdeckung, vermeidet diesen Fehler.

Bei der durch den Autor durchgeführten Beurteilung der fertigen Videosequenz lässt sich feststellen, dass mit dem Refinement eine gute zeitkonsistente Darstellung der unterschiedlichen Auffüllungsergebnisse erreicht wurde. Diese waren jedoch nicht ausreichend in der Lage, die komplexe Hintergrundstruktur zufriedenstellend zu füllen, was wiederum in der Betrachtung negativ auffällt.

5.2.4. Parameter und Rechenzeit

Der vorgestellte Algorithmus bietet einige Möglichkeiten der Parametrisierung, die auch in der Implementierung möglichst variabel gehalten wurden und hier kurz vorgestellt werden sollen. Zur Verfügung steht dabei, neben der üblichen Möglichkeit die Patchgröße zu variieren, auch die Anzahl der einzubeziehenden Nachbarframes. Des Weiteren bietet sich an, über einen Parameter in die Patchsuche einzugreifen und Farb- sowie Tiefenwerte unterschiedlich gewichtet in die Kostenfunktion einfließen zu lassen. In den oben vorgestellten Ergebnissen wurde, wenn nicht anders ausgewiesen, mit vier Nachbarframes gearbeitet und die Farb- und Tiefeninformation als gleichwertig betrachtet.

Die genutzten und empirisch ermittelten Patchgrößen waren den Tabellen zu entnehmen. Sie sind jedoch, wie in allen exemplarbasierten Inpaintingverfahren, von

entscheidender Bedeutung für die Qualität der Auffüllung. Durch die Wahl größerer Kantenlängen lässt sich beispielsweise die Rechenzeit merklich reduzieren. Das führt jedoch unweigerlich dazu, dass schneller fehlerhafte Bildteile kopiert werden, auf deren Grundlage dann meist weitere unpassende Patche ausgesucht werden. Wie bei ähnlichen patchbasierten Verfahren ist es nicht trivial eine automatische Größenbestimmung zu finden und dieser Punkt bleibt zu untersuchen. Bei Videosequenzen ist sogar davon auszugehen, dass je nach Szene eine dynamische Anpassung der Patchgröße für ein optimales Ergebnis vonnöten ist (vgl. Zhang [81]). Im Zuge der Besonderheiten des hier vorgestellten Verfahren konnte jedoch ein grobe Eingrenzung vorgenommen werden. Aufgrund der speziellen Verortung der Aufdeckungen bei der 3D-Stereosynthese (Diese liegen immer an der Seite der Tiefensprünge von Objekten), ist es für die Qualität von Vorteil, wenn die Patchgröße nicht über den Bereich der Aufdeckung reicht. Durch die Suche in den Ausgangsbildern der Synthese würden von dort genau die falschen Bildteile übernommen werden, wie Abbildung 5.19 demonstriert.

Die Anzahl der Nachbarframes ist der zweite wichtige Parameter mit Einfluss auf Qualität und Rechenzeit. Jeder zusätzlich in die Suche einbezogene Frame muss für den zu füllenden Patch durchsucht werden. Andererseits geht möglicherweise wertvolle Bildinformation verloren, wenn dieser Parameter zu klein gewählt wird. Ein verdeckter Bildbereich mag gerade in dem nicht mehr beachteten Frame sichtbar geworden sein. Es bleibt jedoch festzuhalten, dass zuviele einbezogene Frames die Qualität der Auffüllung eher negativ beeinflussen. Wie in Kapitel 4.3.2 hervorgehoben wurde, trägt die ungünstige Gewichtung in diesem Fall dazu bei, zu viele Bildinformationen mit in die Auffüllung einfließen zu lassen und somit das Resultat unnötig weichzuzeichnen. Es scheint erstrebenswert, die Gewichtsfunktion in zukünftigen Arbeiten noch einmal näher zu untersuchen, um die Informationen der Nachbarframes adäquat zu nutzen. Die Wahl von vier Nachbarframes hat sich in der Praxis als guter Kompromiss von Rechenzeit und Bildinformation erwiesen.

Um ein besseres Verständnis für die Zusammenhänge von Patchgröße, Nachbarframes und Rechenzeit zu bekommen, wurde der Algorithmus mit unterschiedlich eingestellten Parametern auf die im letzten Abschnitt vorgestellten Sequenzen angewendet und die benötigte Rechenzeit gemessen[3]. In allen Fällen wurde mit einem Suchraum von 50 Pixeln um den aktuell priorisierten Pixel gearbeitet. Tabelle 5.12

[3]Testsystem: AMD Athlon II X4 640 Prozessor mit 4 Kernen á 3 GHz, 4 GB RAM

Nachbarframes	Zeit Auffüllung	Zeit Refinement
1	30 sec / Frame	05 sec / Frame
2	44 sec / Frame	09 sec / Frame
4	71 sec / Frame	15 sec / Frame
6	93 sec / Frame	22 sec / Frame

Patchgröße 9x9 px

Patchgröße	Zeit Auffüllung	Zeit Refinement
3x3	422 sec / Frame	16 sec / Frame
5x5	154 sec / Frame	15 sec / Frame
7x7	96 sec / Frame	15 sec / Frame
21x21	37 sec / Frame	16 sec / Frame

Nachbarframes 4

Tabelle 5.12.: Rechenzeit Sequenz Tonsoldaten. [Framegröße 576 x 352 px, Inpaintbereich 1660 px, Globaler Patch 97 x 312 px]

zeigt die Ergebnisse der Tonsoldaten-Sequenz und die Tabellen 5.13 und 5.14 entsprechend für die Soccer und Breakdancer-Sequenzen. Es wurde exemplarisch je ein Frame der Sequenz gewählt und in einem ersten Durchlauf, bei fester Patchgröße von 9x9 Pixeln, mit 1, 2, 4 und 6 Nachbarframes getestet. In einem zweiten Durchlauf wurden bei vier festen Nachbarframes je Patchgrößen von 3x3, 5x5, 7x7 und 21x21 Pixeln benutzt und die Rechenzeit gemessen. Dabei ist festzuhalten, dass die Resultate in gewissem Maße auch immer davon abhängen, wie sich der unbekannte Bereich im Laufe der Auffüllung verändert. Je nach Form, Füllposition und -priorität werden unterschiedlich viele Pixel in einer Iteration eingezeichnet. Dennoch lassen sich einige grundlegende Aussagen treffen. Bei Betrachtung der Resultate der Tonsoldaten-Sequenz ist z.B. zu erkennen, dass mit steigender Anzahl der Nachbarframes auch die Rechenzeit in einem konstanten aber geringen Maße ansteigt. Dagegen nimmt die Komplexität quadratisch zu, wenn wir die Patchgröße verringern. Will man die Rechenzeit verringern, bietet es sich also eher an eine Patchvergrößerung in Betracht zu ziehen, als die Verringerung der Nachbarframes. Auch hier muss letztendlich empirisch, wie beim ersten Verfahren, ein Kompromiss zwischen Qualität und toleriertem Rechenaufwand gefunden werden. Man sieht des Weiteren, dass auch die Zeit fürs Refinement bei Veränderung der Nachbarframe-Anzahl in geringem Maße linear ansteigt. Dagegen hat die Veränderung der Patchgröße keine Auswirkung zur Folge, da beim Refinement mit einem globalen Patch gearbeitet wird. Gleiches lässt sich bei den beiden anderen Sequenzen feststellen.

Nachbarframes	Zeit Auffüllung	Zeit Refinement
1	130 sec / Frame	36 sec / Frame
2	168 sec / Frame	61 sec / Frame
4	240 sec / Frame	110 sec / Frame
6	313 sec / Frame	158 sec / Frame

Patchgröße 9x9 px

Patchgröße	Zeit Auffüllung	Zeit Refinement
3x3	1620 sec / Frame	109 sec / Frame
5x5	627 sec / Frame	110 sec / Frame
7x7	334 sec / Frame	109 sec / Frame
21x21	100 sec / Frame	109 sec / Frame

4 Nachbarframes

Tabelle 5.13.: Rechenzeit Sequenz Soccer. [Framegröße 1280 x 720 px, Inpaintbereich 4515 px, Globaler Patch 226 x 376 px]

Nachbarframes	Zeit Auffüllung	Zeit Refinement
1	143 sec / Frame	231 sec / Frame
2	186 sec / Frame	391 sec / Frame
4	206 sec / Frame	696 sec / Frame
6	350 sec / Frame	1019 sec / Frame

Patchgröße 9x9 px

Patchgröße	Zeit Auffüllung	Zeit Refinement
3x3	1637 sec / Frame	680 sec / Frame
5x5	608 sec / Frame	688 sec / Frame
7x7	370 sec / Frame	683 sec / Frame
21x21	135 sec / Frame	682sec / Frame

4 Nachbarframes

Tabelle 5.14.: Rechenzeit Sequenz Breakdancer. [Framegröße 1280 x 720 px, Inpaintbereich 4113 px, Globaler Patch 501 x 449 px]

Die Unterschiede in der benötigten Auffüllzeit erklären sich durch die unterschiedlich großen Aufdeckungsbereiche. Eine Bereich von 1660 Pixeln bei der Tonsoldaten-Sequenz lässt sich mit einem Patch von 9x9 Pixeln und vier Nachbarframes entsprechend drei mal so schnell auffüllen wie der fast 3 mal so große Bereich von 4515 Pixeln der Soccer-Sequenz. Ebenso lassen sich die Unterschiede in der Zeit fürs Refinement mit der Größe des jeweils ermittelten globalen Patches erklären. Der globale Patch der Tonsoldaten-Sequenz ist mit 97x312 Pixeln um Einiges kleiner als der der

Breakdancer-Sequenz und damit fallen weit weniger Rechenoperationen an, die sich entsprechend in der Zeit niederschlagen.

In der Patchsuche fließen Farb- und Tiefenwerte in die Kostenberechnung mit ein und entscheiden darüber, ob ein gefundener Patch letztendlich zur Auffüllung verwendet werden soll. Hier lässt sich bei Bedarf über einen Parameter nachjustieren, ob in bestimmten Szenen die genaue Wiederherstellung der Farbinformationen wichtiger ist als die Beachtung der Tiefe. Dies sollte jedoch mit Bedacht erfolgen, da damit natürlich auch Bildteile des Vordergrunds mit genutzt werden. In den hier untersuchten Sequenzen war das jedoch nicht von Belang, daher wurden Farb- und Tiefenwerte gleichwertig behandelt.

5.3. Fazit

Die in Kapitel 4 vorgestellten und implementierten Inpainting-Verfahren wurden in diesem Kapitel in ihrer konkreten Anwendung präsentiert und die Ergebnisse untersucht.

Das auf Einzelbilder ausgerichtete erste Verfahren war dabei in der Lage Bildinhalte stimmig sowie natürlich wirkend aufzufüllen. Durch die Einbeziehung der Gradienteninformationen in die Patchsuche konnten Bildstrukturen glaubhafter wieder hergestellt werden als bei vergleichbaren Verfahren (z.B. Criminisi et al.[20]). Auch ist die Rechenzeit konkurrenzfähig zu Methoden vergleichbarer Bildqualität (z.B. Drori et al.[24]), wenn auch noch nicht für die Echtzeit-Bildverarbeitung geeignet. Vor allem Dank der neuartigen Einbindung einer Gradienteninterpolation in die Auffüllung konnten weiche Übergänge ohne störende Kanten, wie sonst oft bei exemplarbasierten Verfahren gesehen, in das Bild erreicht werden. Aufgrund der fehlenden Tiefeninformation war aber zu erwarten, dass sich das erste Verfahren nicht so gut bei Aufdeckungen der 3D-Stereosynthese behaupten würde, was mittels einer Testsequenz aus der Videosequenz "Tonsoldaten" untersucht wurde. Das Ergebnis der Untersuchung zeigt, dass nur die Einzelbildauffüllung überzeugen kann. Bei der Auswertung der Einhaltung der Zeitkonsistenz stellte sich heraus, dass man hier deutliche Abstriche machen muss. Ohne Beachtung der intraframe-spezifischen Zusammenhänge war das Verfahren nicht in der Lage, einen sanften Frameverlauf ohne Flackern zu produzieren.

Anschließend wurden die Möglichkeiten der Auffüllung speziell bei 3D-Stereosynthese-

Sequenzen unter Verwendung des zweiten Verfahrens dargestellt. Durch die Integration von Tiefeninformationen sowohl in die Auffüllpriorität als auch in die Patchsuche konnte eine korrekte Auswahl an Patches sichergestellt werden, um Bereiche, die im Hintergrund liegen, auch nur mit ebensolchen Informationen zu füllen. Dank der Suche in benachbarten Frames war es möglich auch verdeckte Bildbereiche zu finden. Es fiel jedoch auf, dass die Auffüllung noch ihre Schwächen hat. Mitunter ungenügend fortgeführte Strukturen würden sich z.B. durch eine Integration von Gradientenwerten in die Patchsuche (wie beim ersten Verfahren) besser auffüllen lassen. Mithilfe der vorgeschlagenen Gewichtung der unterschiedlichen Bildteile aus anderen Frames sollten mögliche Fehler verringert werden. Jedoch hat sich in der Praxis gezeigt, dass es aufgrund von konzeptionellen Schwächen der Gewichtsberechnung, wie in Abschnitt 4.3.3 erläutert, auch zu ungenügenden, weil weichgezeichneten, bis hin zu fehlerhaften Auffüllungen kommen kann. Die vielfach hervorgehobene und wichtige Bedingung der Zeitkonsistenz konnte vor allem Dank des Optimierungsschrittes gut eingehalten werden, wobei auch hier durch die ungenaue Gewichung mit der daraus folgenden Mittelung der Farbwerte eine Qualitätsreduzierung zu sehen war. Bezüglich der Komplexität lässt sich festhalten, dass auch dieses Verfahren noch weit davon entfernt ist innerhalb akzeptabler Rechenzeiten zu agieren, um beispielsweise eine flüssige Nutzerinteraktion zu ermöglichen.

6. Zusammenfassung und Ausblick

Die vorliegende Diplomarbeit beschäftigte sich mit dem Forschungsgebiet des Inpaintings. Zunächst wurde in der Arbeit der aktuelle Forschungsstand auf dem Gebiet und die nötigen theoretischen Grundlagen aufgezeigt. Auf dieser Basis wurden zwei fortgeschrittene Inpaintingverfahren vorgestellt, implementiert und in ihrer praktischen Anwendung untersucht. Das Hauptaugenmerk lag dabei zum einen auf der zügigen und für den Betrachter natürlich aussehenden Auffüllung - einer Grundprämisse des Inpaintings - und zum anderen bei der speziellen Anwendung für Aufdeckungen, die durch die 3D-Ansichtsynthese entstehen und insbesondere im Zusammenhang mit der Videodarstellung besonderen Anforderungen genügen müssen (Stichwort: zeitliche Konsistenz).

Im ersten Teil wurde eine interessante Variante der texturbasierten Verfahren vorgestellt, die von den Autoren des Verfahrens um wichtige Elemente der gradientenbasierten Bildbearbeitung ergänzt worden ist. Dieses Verfahren ist ursprünglich für Einzelbildanwendung ausgelegt worden und kann dort in Bereichen der Bildreparatur oder Bildretusche seine Stärken beweisen. Wie die Resultate zeigen, ist es dadurch möglich, qualitativ hochwertige Bildauffüllungen in Einzelbildern zu erreichen, die im Vergleich zu anderen Verfahren aus der Literatur auch relativ schnell erfolgen. Eine hervorzuhebende Stärke ist die Vermeidung von harten Farbkanten an Randbereichen, die bei den exemplarbasierten Verfahren oft auftreten. Durch Einbeziehung der Gradienteninformationen in den Prioritätsterm konnten lineare Strukturen gut vervollständigt werden, jedoch gab es, wie in vergleichbaren Arbeiten aus der Literatur, Probleme bei der Fortführung von kurvigen Elementen. Dort würde es sich anbieten, anspruchsvollere Strukturfindungsalgorithmen zu integrieren, auf denen dann das untersuchte Verfahren aufbauen könnte. Es bleibt zu untersuchen, inwiefern sich die Parameterfindung für Werte wie z.B. die Patchgröße automatisieren lässt. Momentan muss dies noch manuell erfolgen. Wie zu erwarten und gezeigt, eignet sich dieses Verfahren in seiner grundlegenden Form nicht für die Auffüllung von tiefenbasierten Aufdeckungen oder die erweiterte Anwendung auf Videos. Es ist jedoch gut vorstellbar,

einige Elemente für andere Verfahren zu übernehmen, um z.B. mithilfe der Gradienteninterpolation eine noch natürlichere Integration von Auffüllung zu erreichen, auch in Videos.

Im zweiten Teil wurde ein Verfahren vorgestellt und implementiert, dass sich mit dem Inpainting von Videos beschäftigt und sich dabei speziell auf die Aufdeckungen konzentriert, die bei der 3D-Ansichtssynthese entstehen. Der exemplarbasierte Ansatz, über eine zusätzlich tiefenbeschränkte und auf Nachbarframes erweiterte Patchsuche für den aufzufüllenden Frame passende Bildinhalte zu finden, ist sinnvoll und funktioniert wie gezeigt grundsätzlich, wenn auch je nach Szene mehr oder weniger gut. Die modellbedingten Schwierigkeiten des exemplarbasierten Ansatzes wirken auch hier nach. Die Restriktivität der Entscheidung für einen einmal aufgefüllten Bereich und die Folgefehler lassen sich auch durch späteres Refinement kaum überdecken. Eine verbesserte Auffüllung ließe sich unter anderem erreichen, indem für den Algorithmus mehr passende Patches zur Verfügung gestellt werden - z.B. durch rotierte Varianten der bereits existierenden Patches. Es ist auch vorstellbar, eine mitunter auftretende typische Blockbildung durch unterschiedliche Ränder der gefüllten Patches mit der Gradienteninterpolation des ersten Verfahrens zu verbessern. Eine gute Auffüllung ist, wie bei der Tonsoldaten-Sequenz erläutert wurde, stark von einer korrekten Auffüllungskarte abhängig. Halbtransparente Elemente von Vordergrundobjekten (z.B. durch Bewegungsunschärfe) oder ungenaue Tiefenkarten resultieren in Bildinhalten, die im Vorfeld fälschlicherweise in den Hintergrund synthetisiert werden und zu einer falschen Patchfindung führen. Es bleibt zu untersuchen, wie dies in Zukunft gelöst werden kann. Des Weiteren ist, wie diskutiert, die verwendete Gewichtung fraglich. Die dadurch generierte Mittelung der Bildinhalte erzeugt ohne Frage eine temporale Konsistenz über die betrachteten Bereiche, jedoch auf Kosten der Bildqualität.

Es bleibt festzuhalten, dass das zweite Verfahren zu diesem Zeitpunkt nur mittelmäßige Ergebnisse liefern kann. Die prinzipielle Idee sollte weiterverfolgt, jedoch Elemente wie die tiefenabhängige Bildsuche weiter ausgebaut werden. Es bleibt z.B. zu untersuchen, inwiefern durch eine geänderte Gewichtsfunktion noch stärker als bisher zwischen gefundenen guten Patches unterschieden werden kann, um lokale Merkmale zu stärken und genannte Weichzeichnung zu vermeiden. Unpassende Patches mit hohen Kosten sollten vielleicht überhaupt nicht einbezogen werden. Gerade auch in Bezug auf Rechenzeit ist die Abhängigkeit von globalen Bildstatistiken wie der Standardabweichung der Kosten aller Vergleiche zweifelhaft und in der Praxis nicht

anwendbar.

Es wurde gezeigt, dass das zweite Verfahren eine vergleichsweise lange Laufzeit hat, bis es zu einem Ergebnis kommt. Insbesondere in einer Anwendung mit Nutzerinteraktion ist das noch nicht ausreichend schnell und muss verbessert werden. Obwohl in der vorliegenden Arbeit schon einige Anstrengungen diesbezüglich unternommen worden sind, gibt es sicherlich noch vielfältige weitere Optimierungsmöglichkeiten. So lässt sich z.B. eine Erweiterung der vorhandenen Parallelisierung auf vorhandene Grafikkarten-Hardware gut vorstellen. Auch birgt der Programmcode sicher noch einiges an Verbesserungspotential um z.B. das volle Potential der benutzten *OpenCV*-Bibliothek auszunutzen.

Abschließend lässt sich sagen, dass das Auffüllen von Videos nach wie vor ein hochkomplexes Thema ist, zu dessen Verständnis jedoch hoffentlich diese und andere Arbeiten beitragen werden - und sie damit Raum schaffen für neue Ansätze und Ideen.

Algorithmenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literatur

[1] W. Arendt und K. Urban. *Partielle Differenzialgleichungen: Eine Einführung in analytische und numerische Methoden*. Spektrum Akademischer Verlag, 2010.

[2] Pablo Arias u. a. "A Variational Framework for Exemplar-Based Image Inpainting". In: *International Journal of Computer Vision* 93 (3 2011), S. 319–347.

[3] Jean-François Aujol, Saïd Ladjal und Simon Masnou. "Exemplar-Based Inpainting from a Variational Point of View". In: *SIAM Journal on Mathematical Analysis* 42.3 (2009), S. 1246–1285.

[4] C. Ballester, V. Caselles und J. Verdera. "A variational model for disocclusion". In: *2003 International Conference on Image Processing*. Bd. 3. IEEE, Nov. 2003, III–677.

[5] C. Ballester u. a. "A Variational Model for Filling-In Gray Level and Color Images". In: *IEEE International Conference on Computer Vision* 1 (2001), S. 10.

[6] C. Ballester u. a. "Filling-in by joint interpolation of vector fields and gray levels". In: *IEEE Trans. Image Processing* 10 (2001), S. 1200–1211.

[7] Connelly Barnes u. a. "PatchMatch: a randomized correspondence algorithm for structural image editing". In: *ACM SIGGRAPH 2009 papers*. SIGGRAPH '09. New Orleans, Louisiana: ACM, 2009, 24:1–24:11.

[8] D. Bekir u. a. "An Image Completion Method Using Decomposition". In: *EURASIP Journal on Advances in Signal Processing* 2011 (2011), S. 15.

[9] Marcelo Bertalmio u. a. "Image inpainting". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, S. 417–424.

[10] M. Bertalmio, A. L. Bertozzi und G. Sapiro. “Navier-Stokes, fluid dynamics, and image and video inpainting”. In: *Proc. IEEE Computer Vision and Pattern Recognition (CVPR*. 2001, S. 355–362.

[11] M. Bertalmio u. a. “Simultaneous structure and texture image inpainting”. In: *IEEE Transactions on Image Processing* 12.8 (2003), S. 882–889.

[12] F. Bornemann und T. März. “Fast image inpainting based on coherence transport”. In: *Journal of Mathematical Imaging and Vision* 28.3 (2007), S. 259–278.

[13] Aurélie Bugeau u. a. “Coherent Background Video Inpainting through Kalman Smoothing along Trajectories”. Anglais. In: *Coherent Background Video Inpainting through Kalman Smoothing along Trajectories*. 2010, S. 8.

[14] Tony F. Chan und Jianhong Shen. “Mathematical Models for Local Nontexture Inpaintings”. In: *SIAM J. Appl. Math* 62 (2002), S. 1019–1043.

[15] Tony F. Chan und Jianhong Shen. “Non-Texture Inpainting by Curvature-Driven Diffusions (CDD)”. In: *J. Visual Comm. Image Rep* 12 (2000), S. 436–449.

[16] Tony F. Chan u. a. “Euler's Elastica And Curvature Based Inpaintings”. In: (2002).

[17] C M Cheng, S J Lin und Shang-hong Lai. “Spatio-Temporally Consistent Novel View Synthesis Algorithm From Video-Plus-Depth Sequences for Autostereoscopic Displays”. In: *Ieee Transactions On Broadcasting* 57.2 (2011), S. 523–532.

[18] Sen-Ching S. Cheung, Jian Zhao und M. V. Venkatesh. “Efficient Object-Based Video Inpainting”. In: *Proc. IEEE Int Image Processing Conf.* 2006, S. 705–708.

[19] Microsoft Corporation. *Visual Studio Startseite*. 2012. URL: `http://www.microsoft.com/germany/visualstudio` (besucht am 10. 04. 2012).

[20] A. Criminisi, P. Perez und K. Toyama. “Region filling and object removal by exemplar-based image inpainting”. In: *IEEE T Image Process* 13.9 (2004), S. 1200–1212.

[21] George R. Cross und Anil K. Jain. “Markov Random Field Texture Models”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 5.1 (Jan. 1983), S. 25–39.

[22] Jeremy S. De Bonet. “Multiresolution sampling procedure for analysis and synthesis of texture images”. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, S. 361–368.

[23] Laurent Demanet, Bing Song und Tony Chan. *Image Inpainting by Correspondence Maps: a Deterministic Approach*. Techn. Ber. 2003.

[24] Iddo Drori, Daniel Cohen-Or und Hezy Yeshurun. “Fragment-Based Image Completion”. In: *ACM Transactions on graphics* 22 (2003), S. 303–312.

[25] Xiaojun Du, Dongwook Cho und Tien D. Bui. “Image segmentation and inpainting using hierarchical level set and texture mapping”. In: *Signal Process.* 91.4 (Apr. 2011), S. 852–863.

[26] Alexei Efros und Thomas Leung. “Texture Synthesis by Non-parametric Sampling”. In: *In International Conference on Computer Vision*. 1999, S. 1033–1038.

[27] MJ Fadili, J.L. Starck und F. Murtagh. “Inpainting and zooming using sparse representations”. In: *The Computer Journal* 52.1 (2009), S. 64–79.

[28] Raanan Fattal, Dani Lischinski und Michael Werman. “Gradient domain high dynamic range compression”. In: *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’02. San Antonio, Texas: ACM, 2002, S. 249–256.

[29] Christoph Fehn. “Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV”. In: *Proceedings of SPIE* 5291.2 (2004), S. 93–104.

[30] imcube labs GmbH. *imcube 3D solutions*. 2012. URL: `http://www.imcube.com` (besucht am 19. 04. 2012).

[31] David J. Heeger und James R. Bergen. “Pyramid-based texture analysis/synthesis”. In: *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’95. New York, NY, USA: ACM, 1995, S. 229–238.

[32] Homan Igehy und Lucas Pereira. “Image Replacement through Texture Synthesis”. In: *In International Conference on Image Processing*. 1997, S. 186–189.

[33] Jiaya Jia u. a. “Video Repairing under Variable Illumination Using Cyclic Motions”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (2006), S. 832–839.

[34] Christian Kanzow. *Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren (Springer-Lehrbuch)*. Springer, 15. Sep. 2004.

[35] G. Sapiro K. A. Patwardhan und M. Bertalmio. “Video inpainting of occluding and occluded objects”. In: *Image Processing, 2005. ICIP 2005. IEEE International Conference on*. Bd. 2. II. 2005, S. 69–72.

[36] Norihiko Kawai, Tomokazu Sato und Naokazu Yokoya. “Image Inpainting Considering Brightness Change and Spatial Locality of Textures and Its Evaluation”. In: *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*. PSIVT ’09. Tokyo, Japan: Springer-Verlag, 2008, S. 271–282.

[37] Min-Sung Koh und Esteban Rodriguez-Marek. “Turbo inpainting: Iterative K-SVD with a new dictionary”. In: *2009 IEEE International Workshop on Multimedia Signal Processing* 1 (2009), S. 1–6.

[38] Nikos Komodakis. “Image Completion Using Global Optimization”. In: *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*. CVPR ’06. Washington, DC, USA: IEEE Computer Society, 2006, S. 442–452.

[39] M. Koppel, D. Doshkov und P. Ndjiki-Nya. “Fully automatic inpainting method for complex image content”. In: *Proc. 10th Workshop Image Analysis for Multimedia Interactive Services WIAMIS ’09*. 2009, S. 189–192.

[40] Vivek Kwatra u. a. “Texture optimization for example-based synthesis”. In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH ’05. Los Angeles, California: ACM, 2005, S. 795–802.

[41] Tsz-Ho Kwok, Hoi Sheung und C. C. L. Wang. “Fast Query for Exemplar-Based Image Completion”. In: *IEEE Transactions on Image Processing* 19.12 (2010), S. 3106–3115.

[42] Anat Levin, Assaf Zomet und Yair Weiss. “Learning How to Inpaint from Global Image Statistics”. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. ICCV ’03. Washington, DC, USA: IEEE Computer Society, 2003, S. 305–.

[43] Vincent Lextrait. *The Programming Languages Beacon*. 2012. URL: `http://www.lextrait.com/vincent/implementations.html` (besucht am 25. 04. 2012).

[44] Chih-Hung Ling u. a. “Virtual Contour Guided Video Object Inpainting Using Posture Mapping and Retrieval”. In: *IEEE Transactions on Multimedia* 13.2 (2011), S. 292–302.

[45] Liping Liu und Haisheng Li. “Image Completion Based on Weighting Patch Match”. In: *Proceedings of the 2009 Fifth International Conference on Image and Graphics*. ICIG ’09. Washington, DC, USA: IEEE Computer Society, 2009, S. 824–829.

[46] M.R. Luettgen u. a. “Multiscale representations of Markov random fields”. In: *IEEE Transactions on Signal Processing* 41.12 (1993), S. 3377–3396.

[47] Julien Mairal, Michael Elad und Guillermo Sapiro. “Sparse Representation for Color Image Restoration”. In: *IEEE Transactions on Image Processing* 17.1 (2008), S. 53–69.

[48] Julien Mairal, Guillermo Sapiro und Michael Elad. *Learning multiscale sparse representations for image and video restoration*. Techn. Ber. 2007.

[49] Simon Masnou. “Disocclusion: a variational approach using level lines”. In: *IEEE Transactions on Image Processing* 11.2 (2002), S. 68–76.

[50] Simon Masnou und J.M. Morel. “Level lines based disocclusion”. In: *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*. IEEE. 1998, S. 259–263.

[51] Hossein Mobahi, Shankar R. Rao und Yi Ma. “Data-driven image completion by image patch subspaces”. In: *Proceedings of the 27th conference on Picture Coding Symposium*. PCS’09. Chicago, IL, USA: IEEE Press, 2009, S. 241–244.

[52] OpenCV. *OpenCV DevZone*. 2012. URL: `http://code.opencv.org`.

[53] K. A. Patwardhan, G. Sapiro und M. Bertalmio. “Video Inpainting Under Constrained Camera Motion”. In: *IEEE Transactions On Image Processing* 16.2 (Feb. 2007), S. 545–553.

[54] Patrick Pérez, Michel Gangnet und Andrew Blake. “Poisson image editing”. In: *ACM SIGGRAPH 2003 Papers*. SIGGRAPH ’03. San Diego, California: ACM, 2003, S. 313–318.

[55] Y.B. Qin und F. Wang. “A curvature constraint Exemplar-based image inpainting”. In: 2010, S. 263–267.

[56] J. A. Sethian. “A Fast Marching Level Set Method for Monotonically Advancing Fronts”. In: *Proc. Nat. Acad. Sci.* 1995, S. 1591–1595.

[57] James Albert Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, 1999.

[58] Bin Shen u. a. "Image inpainting via sparse representation". In: *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*. ICASSP '09. Washington, DC, USA: IEEE Computer Society, 2009, S. 697–700.

[59] Jianbing Shen u. a. "Gradient based image completion by solving the Poisson equation". In: *Computers & Graphics* 31.1 (2007), S. 119–126.

[60] T. K. Shih, N. C. Tang und Jenq-Neng Hwang. "Exemplar-Based Video Inpainting Without Ghost Shadow Artifacts by Maintaining Temporal Continuity". In: 19.3 (2009), S. 347–360.

[61] T.K. Shih, N.C. Tang und J.N. Hwang. "Ghost Shadow Removal in Multi-Layered Video Inpaintinga". In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, S. 1471–1474.

[62] B. Simeon. "Numerik partieller Differentialgleichungen". In: *Lecture Notes, Munich Garching* (2004).

[63] Jian Sun u. a. "Image completion with structure propagation". In: *ACM SIGGRAPH 2005 Papers*. SIGGRAPH '05. Los Angeles, California: ACM, 2005, S. 861–868.

[64] Huang Ting u. a. "Image inpainting by global structure and texture propagation". In: *Proceedings of the 15th international conference on Multimedia*. MULTIMEDIA '07. Augsburg, Germany: ACM, 2007, S. 517–520.

[65] Sivan Toledo. *TAUCS, a Library of Sparse Linear Solvers*. 2003. URL: `http://www.tau.ac.il/~stoledo/taucs` (besucht am 09. 04. 2012).

[66] Tsung-Han Tsai und Chih-Lun Fang. "Structural Videotext Regions Completion with Temporal-Spatial Consistency". In: *Proc. IEEE Int. Conf. Sensor Networks, Ubiquitous and Trustworthy Computing SUTC '08*. 2008, S. 256–261.

[67] Tsung-Han Tsai und Chih-Lun Fang. "Text-Video Completion Using Structure Repair and Texture Propagation". In: 13.1 (2011), S. 29–39.

[68] David Tschumperlé. "Fast Anisotropic Smoothing of Multi-Valued Images using Curvature-Preserving PDE's". In: *Int. J. Comput. Vision* 68.1 (Juni 2006), S. 65–82.

[69] David Tschumperle und Rachid Deriche. “Vector-Valued Image Regularization with PDEs: A Common Framework for Different Applications”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27.4 (Apr. 2005), S. 506–517.

[70] Minqin Wang, Guoqiang Han und Yongqiu Tu. “Edge-Based Image Completing Guided by Region Segmentation”. In: *Proc. ISECS Int. Colloquium Computing, Communication, Control, and Management CCCM ’08*. Bd. 1. 2008, S. 152–156.

[71] Li-Yi Wei und Marc Levoy. “Fast texture synthesis using tree-structured vector quantization”. In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, S. 479–488.

[72] Yonatan Wexler, Eli Shechtman und Michal Irani. “Space-Time Completion of Video”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007), S. 463–476.

[73] Y. Wexler, E. Shechtman und M. Irani. “Space-Time Video Completion”. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 1 (2004), S. 120–127.

[74] B. Wohlberg. “Inpainting by Joint Optimization of Linear Combinations of Exemplars”. In: 18.1 (2011), S. 75–78.

[75] A. Wong und J. Orchard. “A nonlocal-means approach to exemplar-based inpainting”. In: *Proc. 15th IEEE Int. Conf. Image Processing ICIP 2008*. 2008, S. 2600–2603.

[76] Jiying Wu und Q. Ruan. “Object Removal By Cross Isophotes Exemplar-based Inpainting”. In: *Proc. 18th Int. Conf. Pattern Recognition ICPR 2006*. Bd. 3. 2006, S. 810–813.

[77] Jiying Wu und Qiuqi Ruan. “A novel hybrid image inpainting model”. In: *Proc. Int. Conf. Audio, Language and Image Processing ICALIP 2008*. 2008, S. 138–142.

[78] Zongben Xu und Jian Sun. “Image inpainting by patch propagation using patch sparsity”. In: *Trans. Img. Proc.* 19.5 (Mai 2010), S. 1153–1165.

[79] Guofeng Zhang. *Depth Video Recovery*. 2009. URL: `http://www.cad.zju.edu.cn/home/gfzhang/projects/videodepth` (besucht am 19. 04. 2012).

[80] Ranran Zhang, Youdong Ding und Shuhan Wei. “Image Inpainting Algorithm Based on Successive Elimination”. In: *Proceedings of the 2009 Eigth IEEE/ACIS International Conference on Computer and Information Science*. ICIS ’09. Washington, DC, USA: IEEE Computer Society, 2009, S. 1111–1114.

[81] S. Zhang und X. Zhou. “An improved scheme for Criminisi’s inpainting algorithm”. In: *Image and Signal Processing (CISP), 2011 4th International Congress on*. Bd. 4. IEEE. 2011, S. 2048–2051.

[82] Yunjun Zhang, Jiangjian Xiao und Mubarak Shah. “Motion Layer Based Object Removal in Videos”. In: *Applications of Computer Vision and the IEEE Workshop on Motion and Video Computing, IEEE Workshop on* 1 (2005), S. 516–521.

[83] Han Zhou u. a. “Image Completion Using Constrained Delaunay Triangulation”. In: *Proceedings of the 2011 Fourth International Conference on Intelligent Computation Technology and Automation - Volume 02*. ICICTA ’11. Washington, DC, USA: IEEE Computer Society, 2011, S. 568–571.

A. Ergebnisse des ersten Verfahrens

Abbildung Baseball

Abbildung A.1.: Vergrößerte Darstellung der Abbildung Baseball [256x170px, Patchgröße 31x31, Dauer 4,3s]

Abbildung A.2.: Beispiel der Abbildung Baseball mit geringerer Patchgröße[256x170 px, Patchgröße 7x7, Dauer 15s]

Abbildung Bungee-Jumper

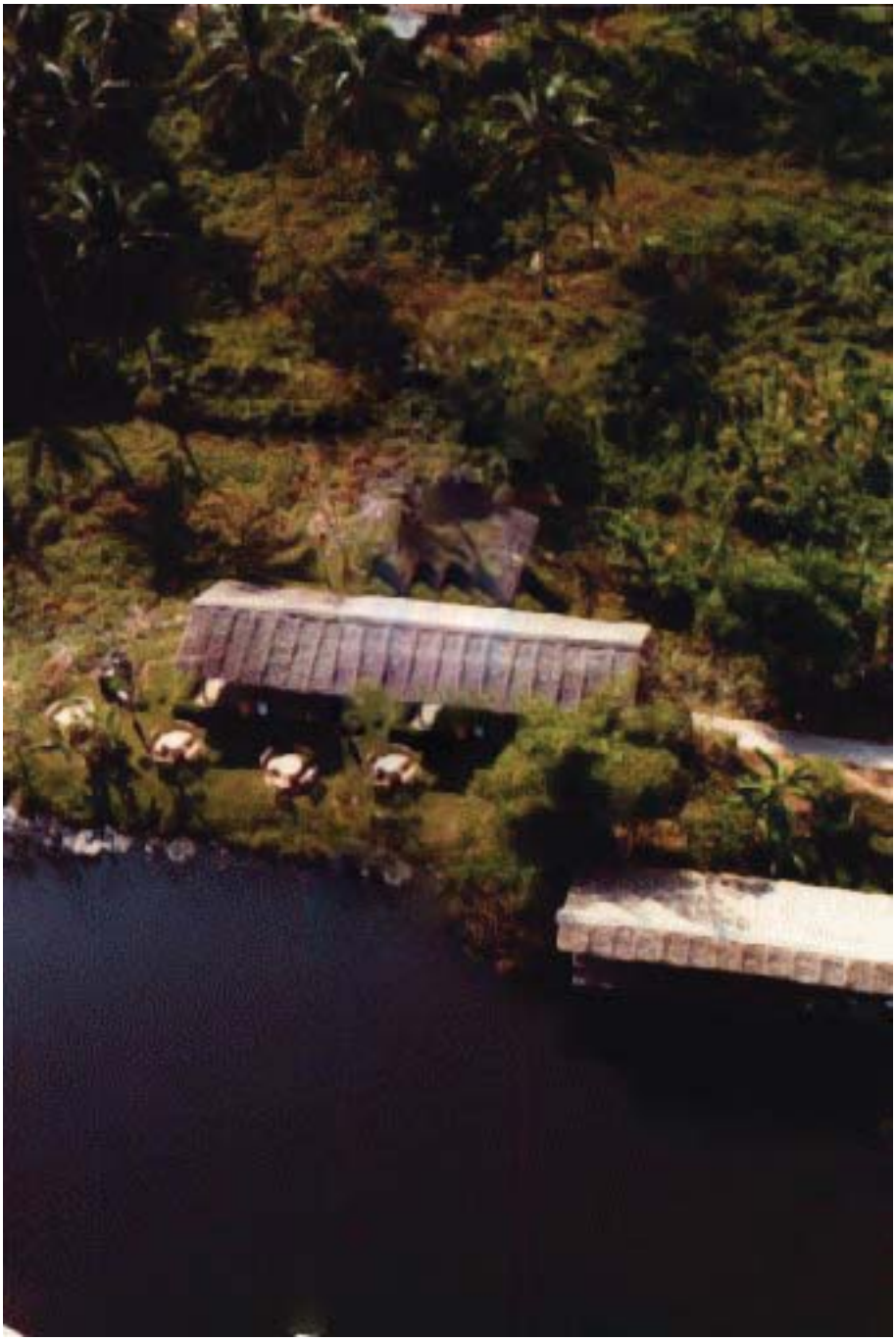


Abbildung A.3.: Vergrößerte Darstellung der Abbildung Bungee-Jumper [342x512 px, Patchgröße 27x27, Dauer 80s]

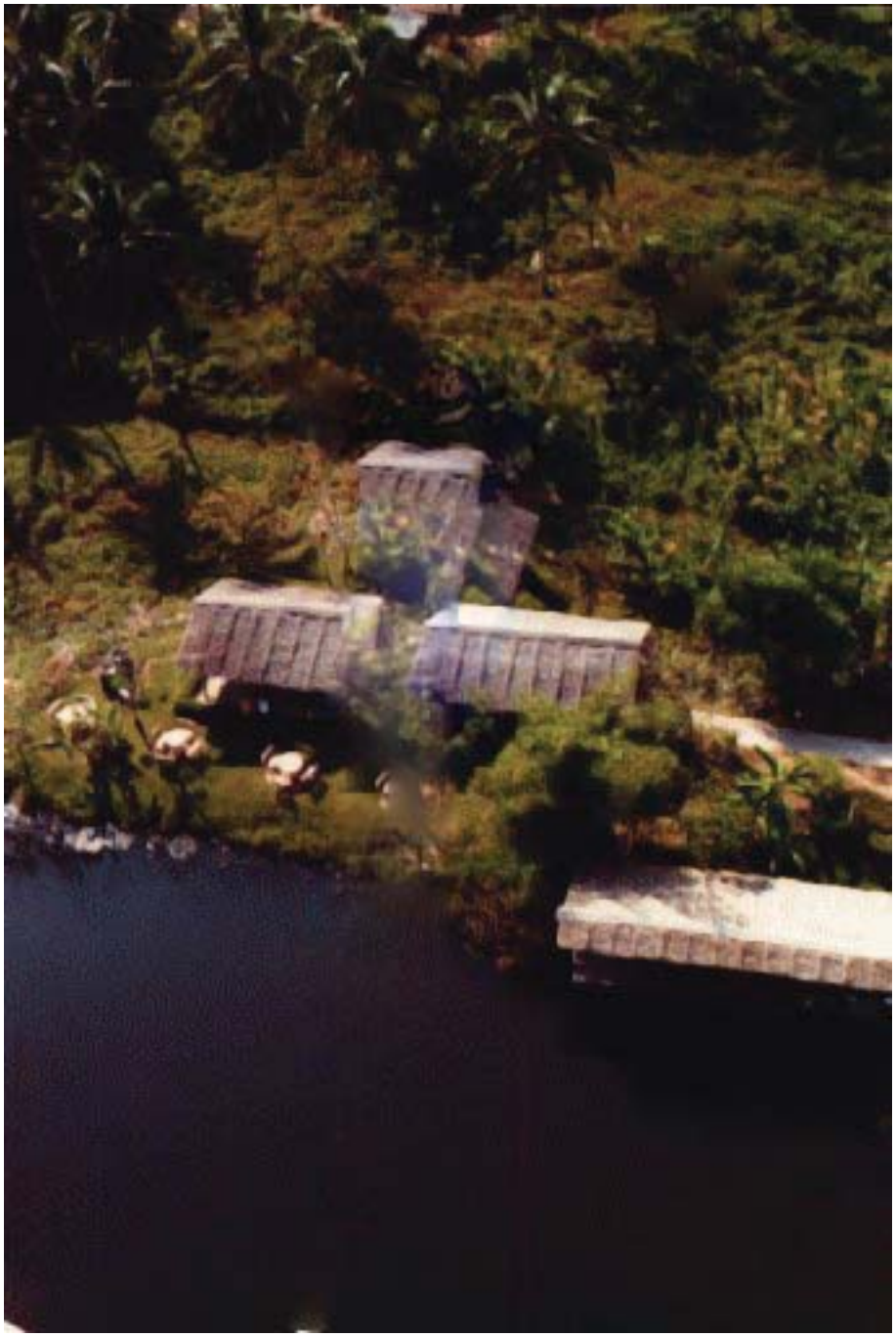


Abbildung A.4.: Beispiel der Abbildung Bungee-Jumper mit zu großer Patchgröße - Die Dachstruktur wird unterbrochen [342x512 px, Patchgröße 43x43, Dauer 63s]

Abbildung Elefant

Abbildung A.5.: Vergrößerte Darstellung der Abbildung Elefant [364x266 px, Patchgröße 41x41, Dauer 18s]

Abbildung A.6.: Beipsiel der Abbildung Elefant mit geringerer Patchgröße [364x266 px, Patchgröße 21x21, Dauer 30s]

Abbildung Jeep

Abbildung A.7.: Größere Darstellung der Abbildung Jeep [364x66 px, Patchgröße 21x21, Dauer 33s]

Abbildung A.8.: Weiteres Beispiel der Abbildung Jeep - Die Bergkante ist nicht überall korrekt fortgeführt worden [364x66 px, Patchgröße 17x17, Dauer 38s]

Abbildung Tonsoldaten

Abbildung A.9.: Größere Darstellung der Abbildung Tonsoldaten - MAn beachte die zeitliche Inkonsistenz [Ausschnitt Frame 0-3, 576x352 px, Patchgröße 7x7, Dauer ~50s]

B. Ergebnisse des zweiten Verfahrens

Sequenz Tonsoldaten

Abbildung B.1.: Sequenz Tonsoldaten: Auffüllung vor dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 576x352 px, 4 Nachbarframes, 15x15 px Patchgröße]

Abbildung B.2.: Sequenz Tonsoldaten Tiefenkarte: Auffüllung vor dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 576x352 px, 4 Nachbarframes, 15x15 px Patchgröße]

Abbildung B.3.: Sequenz Tonsoldaten: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 576x352 px, 4 Nachbarframes, 15x15 px Patchgröße]

Abbildung B.4.: Sequenz Tonsoldaten Tiefenkarte: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 576x352 px, 4 Nachbarframes, 15x15 px Patchgröße]

Sequenz Soccer

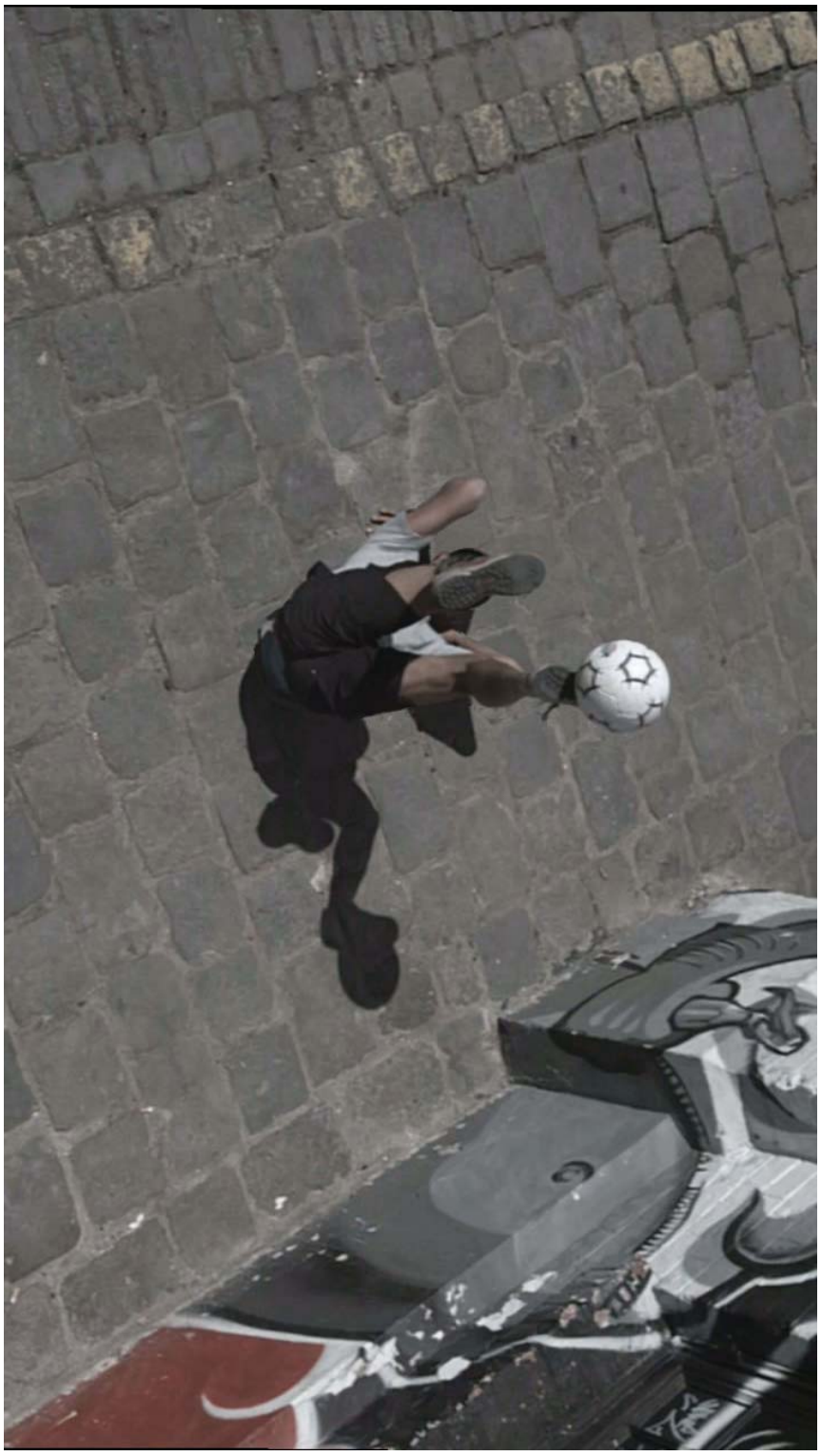


Abbildung B.5.: Sequenz Soccer: Auffüllung vor dem Refinement [Frame 0, 1280x720 px, 4 Nachbarframes, 9x9 px Patchgröße]

Abbildung B.6.: Sequenz Soccer Tiefenkarte: Auffüllung vor dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 9x9 px Patchgröße]

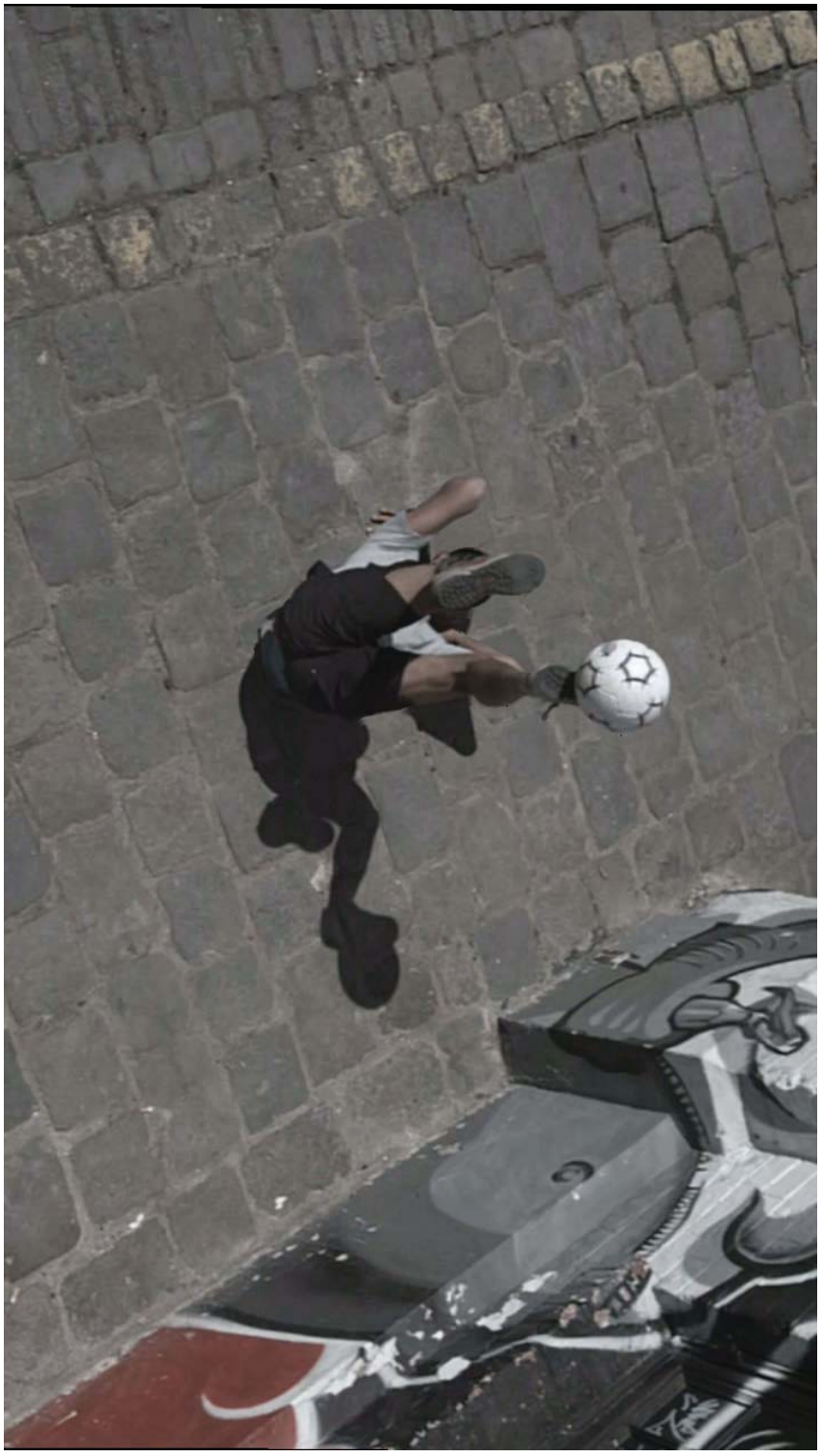


Abbildung B.7.: Sequenz Soccer: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 9x9 px Patchgröße]

Abbildung B.8.: Sequenz Soccer Tiefenkarte: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 9x9 px Patchgröße]

Sequenz Breakdancer

Abbildung B.9.: Sequenz Breakdancer: Auffüllung vor dem Refinement [Frame 0, 1280x720 px, 4 Nachbarframes, 11x11 px Patchgröße]

Abbildung B.10.: Sequenz Breakdancer Tiefenkarte: Auffüllung vor dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 11x11 px Patchgröße]

Abbildung B.11.: Sequenz Breakdancer: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 11x11 px Patchgröße]

Abbildung B.12.: Sequenz Breakdancer Tiefenkarte: Auffüllung nach dem Refinement und nur an der Aufdeckung des großen Vordergrundobjekts [Frame 0, 1280x720 px, 4 Nachbarframes, 11x11 px Patchgröße]