CFD-Techniken zur Visualisierung von Strömungen und Bildbearbeitung

Dissertation
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

Der Fakultät für Mathematik der
Technischen Universität Dortmund
vorgelegt von

Jens Friedrich Acker

im August 2016

Dissertation
CFD-Techniken zur Visualisierung von Strömungen und Bildbearbeitung

Fakultät für Mathematik
Technische Universität Dortmund

Erstgutachter: Prof. Dr. S. Turek

Zweitgutachter: PD Dr. A. Rademacher

Tag der mündlichen Prüfung: 29.9.2016

Danksagung

Die vorliegende Arbeit entstand während meiner Arbeit am Lehrstuhl für numerische Mathematik (LS III) an der Universität Dortmund, welche später in Technische Universität Dortmund umbenannt wurde. Die vorgestellten Bildverarbeitungstechniken beruhen auf Vorarbeiten von Prof. Joachim Weickert von der Saarland Universität und die Visualisierungsmethode für instationäre Strömungen auf Vorarbeiten von Prof. Dr. Martin Rumpf vom Institut für numerische Simulation der Universität Bonn. Bei letzterem Institut war ich zeitweilig angestellt, um ein dort zuvor geschriebenes Visualisierungsprogramm zu erweitern und zu verbessern.
Ich möchte an dieser Stelle meinen Dank an Professor Stefan Turek[1] aussprechen, ohne dessen langjährige Unterstützung und Betreuung diese Arbeit wohl nicht möglich gewesen wäre. Auch meinen lieben Kollegen vom Lehrstuhl III möchte ich für ihre Unterstützung, interessante Gespräche und Hilfe bei der Zähmung widerwilliger Software danken.

[1] Lehrstuhl III der Fakultät Mathematik der TU Dortmund.

Einleitung

Bildverarbeitung bestand früher hauptsächlich aus geschickt konstruierten digitalen Filtern beziehungsweise Filterbänken, später auch Fourier- und Wavelet-Analysen. Digitale Filter können oft als Anwendungen von Spezialfällen partieller Differentialgleichungen interpretiert werden. Untersucht man nun die zugrunde liegenden partiellen Differentialgleichungen, so ergeben sich interessante Modifikationen ("nichtlineare anisotrope Diffusion"), die keinen digitalen Filtern mehr entsprechen und wesentlich mehr Möglichkeiten zur Bildverarbeitung bieten [Wei98, TD02]. Beispiele unterschiedlicher Anwendungen dieser Art von Erweiterungen zur Bildwiederherstellung befinden sich in den Abschnitten 1.3 und 1.4. Obwohl in vielen Fällen immer noch digitale Filter Verwendung finden, werden heutzutage immer mehr nichtlineare Filter und Verfahren basierend auf (partiellen) Differentialgleichungen verwendet.
Lässt man die Nichtlinearität des Diffusionsoperators von einem vorgegebenen Strömungsfeld abhängen und fügt noch einen Transportterm hinzu, so kann man eine Textur, z.B. einfaches Zufallsrauschen, entlang der Strömung zu linienartigen Strukturen verschmieren. Diese "Schlieren" werden mit der Zeit immer gröber und müssen entweder neu initialisiert werden oder mit feineren Lösungen kombiniert werden. Das Ergebnis kann zur besseren Visualisierung mehrskaliger Strömungsprozesse in 2D verwendet werden, wie in Abschnitt 3 zu sehen ist. Der 3D-Fall wird angesprochen, aber wegen des unverhältnismäßigen Rechenaufwandes für eine passende Visualisierung der Ergebnisse nicht weiter ausgeführt.

Schlüsselwörter

Bildverarbeitung, Bildergänzung, Perona–Malik, Anisotrope Diffusion, Visualisierung, Partielle Differenzialgleichung.

Verwendete Symbole

$\vec{x}$**:** Koordinaten eines Punktes. Kann im 2D Fall (x, y) oder (x_1, x_2) bezeichnen oder im 3D Fall (x, y, z) oder (x_1, x_2, x_3).

$\frac{\partial}{\partial \mathbf{x_i}}$**:** Partielle Ableitung nach x_i.

$\nabla \mathbf{u}$**:** Gradient der Funktion u. Wenn diese n Ortsparameter x_i besitzt, so ist dies der Vektor $(\frac{\partial}{\partial x_1}u, \ldots, \frac{\partial}{\partial x_n}u)^\top$.

div $\vec{v}$ oder $\nabla \cdot \vec{v}$**:** Divergenz der Vektorfunktion $\vec{v}$. Wenn diese n Ortsparameter x_i besitzt, so ist dies $\sum_{i=1}^{n} \frac{\partial}{\partial x_i} v_i$. Bei matrixwertigen Funktionen $\mathbf{v} = (\vec{v}_1 \ \ldots \ \vec{v}_m)$ gilt $\overrightarrow{\text{div}}\ \mathbf{v} = (\text{div}\ \vec{v}_1, \ldots, \text{div}\ \vec{v}_m)^\top$.

$\mathbf{\Delta u}$**:** Laplace Operator angewendet auf die Funktion u. Dieser ist definiert als $\Delta u := \sum_{i=1}^{n} \frac{\partial^2}{\partial {x_i}^2} u = \text{div}\ \nabla u$.

trace A: Spur der Matrix $A := (a_{ij})_{i,j=1}^{n}$. Die Spur ist definiert als $\operatorname{trace} A := \sum_{i=1}^{n} a_{ii}$.

$\mathcal{H}$ u: Die Hesse-Matrix der Funktion u mit n Ortsparametern. $\mathcal{H}\, u := \left(\frac{\partial^2}{\partial x_i \partial x_j} u\right)_{i,j=1}^{n}$.

Id: Identitätsabbildung beziehungsweise die zugehörige Matrix. In der Form Id_n bezeichnet es auch eine (n,n)-Matrix mit Einsen auf der Hauptdiagonalen und sonst nur Nullen.

$\tilde{\mathbf{v}} \| \tilde{\mathbf{w}}$: Der Vektor $\vec{v}$ ist kollinear zum Vektor $\vec{w}$.

$\mathbf{K}_\rho$: Gaußkern der Varianz σ auf $\mathbb{R}^n$ definiert durch $K_\rho(\vec{x}) := \frac{1}{(\sqrt{2\pi}\sigma)^n}\, e^{-\frac{\|\vec{x}\|^2}{2\sigma^2}}$.

$\mathbf{W^{k,p}(\Omega)}$: Sobolevraum. Alle reellwertigen $L^p(\Omega)$-Funktionen, deren schwache Ableitungen bis zur k-ten Ableitung in $L^p(\Omega)$ liegen.

$\mathbf{H^k}$: Hilbertraum basierend auf einem Sobolevraum. $H^k := W^{k,2}$.

$\mathbf{H_0^k(\Omega)}$: Für $k \geq 1$ sei $H_0^k(\Omega) := \{v \in H^k \mid v_{|\partial\Omega} = 0\}$.

$\mathbf{L_0^2(\Omega)}$: $L_0^2(\Omega) := \{v \in L^2(\Omega) \mid \int_\Omega v(x)\, \mathrm{d}\vec{x} = 0\}$. [2]

$\mathbf{f_{|D}}$: Einschränkung des Definitionsbereiches der Funktion f auf die Menge D. Bei einem Raum von Funktionen V sei $V_{|D} := \{f_{|D} \mid \forall f \in V\}$.

mod(a, b): Rest der Division von $a \geq 0$ durch $b > 0$.

[2] L^2-Funktionen sind Klassen von Funktionen, die sich auf Nullmengen, wie dem Rand von Ω, unterscheiden können, obwohl sie zur selben Klasse gehören. Die Vorgabe eines Nullrandes macht daher keinen Sinn.

Inhaltsverzeichnis

Kapitel 1

Bildwiederherstellung

Bilder können unterschiedliche Mängel aufweisen, die entweder technisch bedingt bei der Aufnahme entstehen oder später durch Alterungsprozesse, Lagerungsfehler oder durch unsachgemäße Handhabung auftreten. Ein typischer technischer Fehler, der bei der Aufnahme von Bildern entsteht, besteht aus einem gewissen Rauschen, das die Bildinformation überlagert oder punktweise die Bildinformation ersetzt. Die Ursache ist ein unvermeidliches Grundrauschen, das Teil der Bildsensoren in modernen Kameras ist, oder einzelne fehlerhafte Zellen in diesen Sensoren. Das Grundrauschen tritt in vielen Fällen normalverteilt auf, wohingegen einzelne fehlerhafte Zellen punktweise Störungen erzeugen, welche auch als "Salz und Pfeffer"-Rauschen[1] bezeichnet werden. Für diese Art von Fehlern eignen sich klassische Ansätze der Bildverarbeitung, die auf Frequenzfiltern und Untersuchung der statistischen Korrelation der Bildinformation beruhen.
Eine ganz andere Art von Fehlern entsteht, wenn Bilder ausgedruckt wurden oder fototechnisch aus einem Negativ eine Papierversion des Positivbildes erstellt wurden und diese Bilder bei der Lagerung, Handhabung oder durch Alterung beschädigt werden. Diese Art von Fehlern zeigen sich als Flecken oder "Risse" im Bild. Das bedeutet, in einzelnen Regionen ist die Bildinformation verändert worden oder fehlt ganz. Diese Stellen müssen möglichst unauffällig geflickt werden. Kanten, die durch diese Fehlstellen unterbrochen wurden, müssen passend fortgesetzt werden, um beschädigte Teile von Strukturen und Formen des Bildes zu reparieren. Diese Fortsetzung sollte aber zum Beispiel keine Kreisränder mit Geraden reparieren, da dies visuell sehr auffällig wäre und im Bild vorhandene Formen verändern würde. Ein großes Problem sind Muster, deren Details feiner als die zu reparierenden Risse sind. Glatte Ergänzungen in einer gemusterten Region fallen visuell sehr stark auf, da das menschliche Wahrnehmungsvermögen sehr gut im Erkennen von Mustern ist und Unterbrechungen oder lokale Veränderungen von erkannten Mustern besonders auffällig sind. Diese Stellen sollten besser durch Einfügen von unbeschädigten Musterteilen aus anderen Bildregionen repariert werden. B. A. Youssef[2] hat entlang dieser Idee schon einen Algorithmus zu Bildwiederherstellung erstellt.

[1] In der englischsprachigen Literatur auch als "salt & pepper noise" bezeichnet.

[2] Computer Based Engineering Application Department, Informatics Intitute, Moubark City for Scientitic research and Technological Applications, Alexandria, Ägypten.

Auch hat er die hier aufgegriffene Idee, angeregt durch einen Vorschlag durch Bertalmio, Bertozzi und Sapiro in [BBS01], zur Bildergänzung vorgeschlagen, wobei mit modifizierter Navier-Stokes-Wirbelstärken-Formulierung auf Tensorproduktgittern mit finiter Volumendiskretisierung und Runge-Kutta Zeitschrittverfahren gearbeitet werden sollte. Die Modifizierung der Navier-Stokes-Gleichungen bestand dabei aus dem Ersetzen des Laplace-Operators durch einen nichtlinearen Diffusionsoperator wie Perona-Malik (siehe Abschnitt 1.3.1) oder NAD (siehe Abschnitt 1.3.2).

1.1 Problemstellungen

Um das Problem der Bildwiederherstellung abstrahieren zu können, muss man zuerst Bilder etwas anders als nur als Ansammlung von Pixeln darstellen. Im Folgenden, sofern nicht speziell anders definiert, beschränken wir uns O.B.d.A. auf ein quadratisches Gebiet $\Omega := (-1;1)^2$ und definieren ein (Graustufen-) Bild als Abbildung $I : \Omega \to [0;1]$.
Es gibt die verschiedensten Arten, wie man eine solche Funktion aus den Pixeldaten des zu bearbeitenden Bildes erhalten kann und man muss die Wahl der Methode jeweils auf den verwendeten Lösungsansatz anpassen. Falls ein Ansatz eine Bildfunktion in $C^1(\Omega)$ voraussetzt, sollte man zum Beispiel keine bilineare Interpolation verwenden. Eine ungeschickte Wahl der Methode kann zu unnötigen Interpolationsfehlern, schlecht definierten Problemen oder Ergebnissen außerhalb des ursprünglichen Wertebereiches (zum Beispiel Überschwingen bei Interpolation mit höherer Ordnung an Bildkanten, Gibbs'sches Phänomen) führen.
Wenn Pixeldaten mit Bildpunkten im Wertebereich [0;1] bilinear auf einem regelmäßigen Tensorproduktgitter interpoliert werden, ergibt sich eine Bildfunktion als Startvorgabe einer Rekonstruktion, die stetig im gesamten Gebiet und zumindest bis auf eine Nullmenge stetig differenzierbar ist. Die Problemstellen bei dieser Interpolationsmethode liegen dabei auf den Knoten und Kanten eines durch die Pixel gebildeten Tensorproduktgitters. Dort kann der Gradient der Bildfunktion Sprünge aufweisen. Ist die Gitterweite h, kann allerdings die Norm des Gradienten an allen anderen Stellen durch $\sqrt{2}h^{-1}$ beschränkt werden. Die so erzeugte Funktion liegt in $H^1(\Omega)$.

1.1.1 Entrauschung

Der Vorgang des Entrauschens versucht, aus einem gestörten Bild $\tilde{I} : \Omega \to [0;1]$ das ursprüngliche Bild I durch Entfernen der Störung so gut wie möglich wiederherzustellen. Dabei sind im Grunde zwei Arten des Rauschens zu beachten:

Überlagerung Die Störung läßt sich als Addition einer Rauschfunktion $\eta_\delta : \Omega \to [-\delta;\delta]$ zur Bildfunktion darstellen. Das bedeutet $\tilde{I} = I + \eta_\delta$. Dabei muss beachtet werden, zum Beispiel durch Abschneiden an den Intervallgrenzen, dass $\tilde{I}$ immer innerhalb des Intervalls $[0;1]$ bleibt!

Punktweise Ersetzung Hier kommen im Grunde zwei Rauschfunktionen $\eta, \theta : \Omega \to [0;1]$ mit gleichmäßiger Verteilung und ein Parameter δ zur Anwendung. Das gestörte Bild ergibt sich nun als

$$\tilde{I}(\vec{x}) = \begin{cases} \theta(\vec{x}) & \text{falls } \eta(\vec{x}) \leq \delta \text{ ist,} \\ I(\vec{x}) & \text{sonst.} \end{cases} \tag{1.1}$$

Als Variante kann die Funktion θ auch als $\theta : \Omega \to \{0, 1\}$ definiert sein.

(a) Originalbild (b) Überlagerung (c) Ersetzung

Abbildung 1.1.1: Beispiele für unterschiedliche Rauscharten

1.1.2 Bildergänzung

Dabei liegen ein oder mehrere zusammenhängende Gebiete Ω_i vor, in denen jede Bildinformation fehlt. Dies kann durch Beschädingungen im Bildmaterial entstehen oder es wurden absichtlich Bereiche maskiert, um Objekte verschwinden zu lassen.
Die Bildergänzung versucht, für diese Bereiche die Bildinformationen aus den vorhandenen Informationen wiederherzustellen oder durch passendes Füllmaterial zu einem vollständigen Bild zu ergänzen. Hierbei werden oft nur die Informationen aus der unmittelbaren Umgebung der einzelnen Ω_i benutzt. Eine Methode dafür wird in Abschnitt 1.4 vorgestellt. Beispiele dazu sind in Abschnitt 1.4.4 zu finden.

1.2 Verbindung klassischer Algorithmen mit PDEs

Klassische (Filter-)Algorithmen der Bildverarbeitung definieren oft eine sogenannte Maske ("Stencil"), das heißt eine beschränkte Umgebung fest vorgegebener Form, die auf den zu bearbeitenden Bildpunkt verschoben wird und die angibt, welche Punkte zur Berechnung des neuen Bildpunktes herangezogen werden sollen. Verbreitet sind der sogenannte 5-Punkte-Stern, das bedeutet der Bildpunkt und die direkten Nachbarn (Oben, Unten, Rechts und Links), und der 9-Punkte-Stern, der zusätzlich noch die direkten diagonalen Nachbarn enthält (siehe Abbildung 1.2.1).

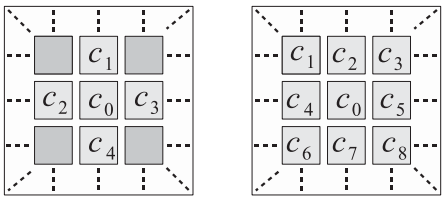


Abbildung 1.2.1: Masken für den 5– und 9–Punktestern

Bei normalen linearen Filteralgorithmen wird zu jedem dieser Maskenpunkte ein Gewichtungsfaktor c_i angegeben. Um den Inhalt des neuen Bildpunktes zu erhalten, wird nun der Wert jedes Bildpunktes der Maske mit dem Gewichtungsfaktor multipliziert und danach addiert. Bei nichtlinearen Filtern können diese Gewichtungsfaktoren zusätzlich von der Bildinformation oder anderen Daten abhängen. Zum Beispiel könnte man die Stärke eines Weichzeichners vom Ergebnis eines Kantendetektionsfilters abhängig machen oder einen anisotropen Weichzeichner durch ein vorgegebenes Vektorfeld steuern lassen.
Bei allen Algorithmen dieser Art muß man darauf achten, wie Maskenpunkte außerhalb des Bildes behandelt werden sollen. Dafür gibt es verschiedene Möglichkeiten:

- Angabe eines festen Wertes, der zum Beispiel einer vorher festgelegten Hintergrundfarbe entspricht.
- Fortsetzung durch Spiegelung der Werte.
- Extrapolation der Werte über den Rand hinaus.
- Fortsetzung am gegenüberliegenden Rand. Ein Bild wird damit topologisch zu einem Torus geschlossen.

1.2.1 Weichzeichnen

Das Weichzeichnen entspricht einem Diffusionsprozess, der Bildwerte eines Bildes I entlang des Gradienten ∇I verwischt. Dabei wird der Mittelwert der Bildwerte des gesamten Bildes nicht verändert. Dieser Prozess lässt sich auf verschiedene Arten formulieren:

- Als Faltung des Bildes I mit dem Gaußkern K_σ (siehe Seite 4).
- Als Lösung $u(t, \vec{x})$ der Wärmeleitungsgleichung

$$\begin{aligned} u_t - \Delta u &= 0 && \text{in } (0,\infty) \times \mathbb{R}^2 \\ u(0,\vec{x}) &= I(\vec{x}) && \text{in } \mathbb{R}^2 \end{aligned} \tag{1.2}$$

zum Zeitpunkt $T = \frac{1}{2}\sigma^2$. Wenn I in $C(\mathbb{R}^2) \cap L^\infty(\mathbb{R}^2)$ ist, so gibt es nur eine Lösung[3] $u \in C_1^2((0,\tilde{T}) \times \mathbb{R}^2) \cap C([0,\tilde{T}] \times \mathbb{R}^2)$, welche die Wachstumsbedingung

$$\|u(t,x)\| \leq A \exp^{a\|x\|^2} \qquad (x \in \mathbb{R}^2, 0 \leq t \leq \tilde{T})$$

für Konstanten $A, a > 0$ erfüllt [Eva98, Theorem 7 auf Seite 58]. Da dies für beliebige $\tilde{T}$ gilt, kann es auch auf den Fall $T < \tilde{T}$ angewandt werden. Die Faltung von I mit dem Gaußkern erfüllt sowohl die Differentialgleichung als auch die Wachstumsbedingung und ist damit die eindeutige Lösung. u ist sogar in $C^\infty((0,\infty) \times \mathbb{R}^2)$ und

$$\lim_{(t,x)\to(0,x_0),x\in\mathbb{R}^2,t>0} u(t,x) = I(x_0) \qquad \forall x_0 \in \mathbb{R}^2$$

laut [Eva98, Theorem 1 auf Seite 47].

Über die Faltung mit dem Gaußkern ist eine analytische Lösung für Anfangsbedingungen mit der passenden Glattheit angebbar. Allerdings sind nicht für alle möglichen Faltungsintegrale passende Stammfunktionen einfach bestimmbar, weshalb eine numerische Berechnung einer guten approximativen Lösung als Alternative benutzt wird.
Außerdem sind die Anfangsbedingungen nicht immer von passender Glattheit und das zu untersuchende Gebiet Ω ist in der Regel nicht der $\mathbb{R}^d$, sondern eine offene beschränkte nichtleere zusammenhängende Teilmenge des $\mathbb{R}^d$. Zuerst schränke man das Problem auf Ω und den zugehörigen Raum-Zeit-Zylinder $\Omega_T := (0;T] \times \Omega$ ein und erhalten nach [Eva98, Seite 350] die Gleichungen

$$\begin{aligned} u_t - \Delta u &= 0 && \text{in } \Omega_T \\ u &= 0 && \text{auf } [0;T] \times \partial\Omega \\ u(0,\cdot) &= I(\cdot) && \text{in } \Omega. \end{aligned} \tag{1.3}$$

Eine Aufweichung dieses Problems über eine variationelle Formulierung ergibt, dass auch schwache Lösungen u im Raum $L^2(0,T;H_0^1(\Omega))$ und u_t im Raum $L^2(0,T;H^{-1}(\Omega))$ für Anfangswerte $I \in L^2(\Omega)$ existieren [Eva98, Kapitel 7.1.1 und 7.1.2, Seite 352-354]. d ist hier die Dimension des Raumes. Die schwache variationelle Formulierung der Gleichung (1.3) lautet nun für fast alle $t \in (0,T)$:

$$\langle u_t, \phi \rangle + \int_\Omega \nabla u \cdot \nabla \phi \, \mathrm{d}\vec{x} = 0 \qquad \forall \phi \in H_0^1(\Omega) \tag{1.4}$$

mit der dualen Paarung $\langle u_t, \phi \rangle$ von $H^{-1}(\Omega)$ mit $H_0^1(\Omega)$ nach [Eva98, Seite 283].
Zur numerischen Lösung des Problems muss (1.4) zeitlich und räumlich diskretisiert werden. Zur zeitlichen Diskretisierung stehen eine Vielzahl von Zeitschrittverfahren bereit. Beispiele sind expliziter und impliziter Euler, Crank-Nicholson,

[3] $u \in C_1^2(\cdot)$ bedeute: u ist zweimal stetig differenzierbar im Ort und einmal stetig differenzierbar in der Zeit

Runge-Kutta, lineare Mehrschrittverfahren und Fraktional-Step. Verfahren mit festen oder adaptiven Zeitschritten mittels Fehlerschätzern, von denen auch einige zur Auswahl stehen, sind ebenfalls für dieses Problem ausgiebig in der Literatur zu finden. Die ersten drei angeführten Beispiele lassen sich als "one-step-θ"-Ansatz mit Hilfe eines freien Parameters θ so formulieren:

$$\frac{u_{n+1} - u_n}{t_{n+1}} - \Delta(\theta u_{n+1} + (1-\theta)u_n) = 0 \qquad \text{in } \Omega \tag{1.5}$$

$$u_0 = I \qquad \text{in } \Omega. \tag{1.6}$$

Wobei t_n die Weite des n-ten Zeitschritts bezeichnet und u_n die Lösung im n-ten Zeitschritt. Das implizite Euler-Verfahren ergibt sich für $\theta = 1$, das explizite für $\theta = 0$ und das Crank-Nicholson-Verfahren für $\theta = \frac{1}{2}$. Beim impliziten Euler und Crank-Nicholson-Verfahren muss bei jedem Zeitschritt folgendes Problem gelöst werden:

$$(Id - t_{n+1}\theta\Delta)u_{n+1} = (Id + t_{n+1}(1-\theta)\Delta)u_n \qquad \text{in } \Omega.$$

Zur örtlichen Diskretisierung bieten sich Standardmethoden wie finite Differenzenverfahren oder finite Elelemente (FEM) an. Letztere können auch auf unstrukturierten Gittern verwendet werden. Falls auf einem Gitter gerechnet wird, das ein achsenparalleles Tensorproduktgitter darstellt, so sollte eine Diskretisierung des Laplace-Operators mittels des zentralen Differenzenverfahrens in Betracht gezogen werden, da diese Diskretisierung sowohl einfach zu implementieren als auch von zweiter Ordnung ist. Wenn man auf unstrukturierten Gittern arbeitet oder noch höhere Genauigkeit haben will, sollte ein FEM-Ansatz mit Elementen höherer Ordnung benutzt werden. Finite Elemente, welche einen Ansatzraum höherer Ordnung verwenden, erlauben eine höhere Genauigkeit bei gleicher Gitterweite zu erreichen, insofern die Lösung keine starken Gradienten aufweist. Allerdings sollte man darauf achten, auch ein Zeitschrittverfahren passender Ordnung zu verwenden, um die höhere Genauigkeit (im Raum) für jeden einzelnen Zeitschritt nicht zu vergeuden.

1.3 Entrauschung mit PDEs

Natürlich dämpft ein einfaches Weichzeichnen additives[4] Rauschen weg. Allerdings trifft dies auch auf die wiederherzustellenden Bilddaten zu. Punktuelle Störungen durch sogenanntes Salz & Pfeffer-Rauschen (siehe auch Funktion (1.1)) können sich sogar als störende Flecken ausbreiten. Als Beispiel dafür kann man Originaldaten mit konstantem Wert 0 annehmen und an zufälligen Gitterpunkten den Wert 1 einstreuen. Nach kurzer Anwendung der Wärmeleitungsgleichung (1.2) (T klein) oder Faltung mit K_σ, ist das Bild von grauen Flecken übersät. Ein geeignetes Entrauschungsverfahren sollte also starke punktuelle Abweichungen von den umgebenden Bilddaten erkennen können und diese Stellen, mit zu der Umgebung passenden Bilddaten, auffüllen. Falls das Originalbild allerdings genau solche

[4] Ein normalverteiltes Rauschen, welches mit den Originaldaten durch Addition kombiniert wird.

punktweisen Einstreuungen enthält, werden diese natürlich mit entfernt. Es wird daher immer mit der impliziten Annahme gearbeitet, dass dies nicht der Fall ist. Ein anderes Problem sind scharfe Kanten und Ecken. Diese Bildstrukturen sollten möglichst an derselben Position und ohne Veränderung der Form erhalten bleiben, da sie oft wichtige Bildinhalte darstellen. Zur Behebung dieses Problems wurden in der Vergangenheit verschiedene Ansätze verwendet. Ein naheliegender Ansatz liegt in der Modifizierung des Diffusionsoperators. Im den folgenden Abschnitten werden dazu zwei mögliche Ansätze angeführt.
Da diese, aus jenen Ansätzen ergebenden, Methoden normalerweise nur auf Bilddaten angewendet werden, werden üblicherweise gleichmäßige Tensorprodukgitter und finite Differenzenansätze zur Implementierung verwendet. Um diese Methoden auch zur Entrauschung oder Rekonstruktion von relevanten Daten oder Strukturen für eine größere Menge von Anwendungsfällen[5] leichter zugänglich zu machen, ist ein Übergang auf unstrukturierte Gitter wünschenswert. Es ist zwar möglich, diese Daten zuerst auf ein Tensorproduktgitter zu interpolieren und nach der Verarbeitung auf das alte Gitter erneut zu interpolieren, dabei entstehen aber zusätzlich zwei Interpolationsfehler. Ansätze mit finiten Elementen bieten sich daher an. Dafür ist eine Umformulierung der partiellen Differentialgleichungen in eine variationelle Form nötig.

1.3.1 Perona-Malik

Unter der Annahme, dass Bildkanten hohe lokale Gradientennormen besitzen, kann man die Stärke des Diffusionsoperators in solchen Regionen herabsetzen. Dadurch wird der Diffusionsoperator nichtlinear. Wenn die Diffusion nur von einer Funktion $g(\|\nabla I\|^2)$ abhängt, so handelt es sich um den sogenannten Perona-Malik-Ansatz zur Rauschentfernung (1.7). Dabei ist $g(\cdot) \in C^1(\{s \in \mathbb{R} | \, s \geq 0\})$ normalerweise eine monoton fallende C^1-Funktion mit $\lim_{s\to\infty} g(s) = 0$ und $g(0) = 1$:

$$\begin{aligned} \partial_t u - \operatorname{div}(g(\|\nabla u\|^2) \cdot \nabla u) &= 0 \qquad \text{auf } (0,\infty) \times \Omega, \\ \partial_n u &= 0 \qquad \text{auf } (0,\infty) \times \partial\Omega, \\ u(0,\cdot) &= I(\cdot) \quad \text{auf } \Omega. \end{aligned} \tag{1.7}$$

Um das Problem wohldefiniert zu machen, ist es rein formal notwendig, dass u zuerst zu u_σ regularisiert[6] wird, bevor $g(\|\nabla I_\sigma\|^2)$ berechnet wird, damit die Norm des Gradienten beschränkt bleibt und somit die Diffusion nicht ohne Untergrenze gegen 0 strebt.
Obwohl der Diffusionsoperator rein formal isotrop erscheint, tritt in der Anwendung durch dessen Nichtlinearität ein anisotropes Verhalten zutage, welches die Diffusion tangential zur Gradientenrichtung herabsetzt. Dieses Verhalten kann

[5] Damit sind Datensätze mit nicht frei wählbaren Messpunkten gemeint, wie zum Beispiel Daten aus Wetterstationen.

[6] Zum Beispiel durch Hinzufügen von etwas Diffusion mittels einer Faltung mit einem Gaußkern kleiner Varianz. Eine Erweiterung von u auf ganz $\mathbb{R}^2$, zum Beispiel als Funktion mit auf Ω beschränktem Träger ist dazu notwendig. Dadurch gilt sogar $u_\sigma \in C^\infty(\Omega)$.

man, wie in [CBFAB97, KDA97] gezeigt, auch mathematisch herleiten, indem man lokal auf ein anderes Koordinatensystem (ξ, η) wechselt, das tangential und orthogonal zur Gradientenrichtung ausgerichtet wird. Dadurch ändert sich die Differentialgleichung in (1.7) zu:

$$\partial_t u - g_\xi u_{\xi\xi} - g_\eta u_{\eta\eta} = 0$$

mit $g_\xi = g(\|\nabla u\|^2)$ und $g_\eta = g'(\|\nabla u\|^2)\|\nabla u\| + g(\|\nabla u\|^2)$. $u_{\xi\xi}$ und $u_{\eta\eta}$ sind hier die jeweiligen zweiten Richtungsableitungen von u.
Die Verwendung des Perona-Malik-Ansatzes entfernt zwar sehr gut einfaches additives Bildrauschen unter Beibehaltung der Existenz starker Bildkanten, sofern eine passende Funktion $g(\cdot)$[7] gewählt wird (siehe auch [Wei98, S. 15-17]). Es gibt aber Probleme bei der Erhaltung der Position dieser Bildkanten und Abrundung von Ecken. Bei zu langer Anwendung des Verfahrens tendieren scharf abgetrennte einfach zusammenhängende Bildbereiche dazu, zuerst abzurunden und zu deformieren, um danach im Laufe der Zeit zuerst zu einer Kreisform zusammengezogen und danach langsam aufgelöst zu werden. Dies kann als Skalenübergang von kleinskaligen zu grobskaligen Bilddetails betrachtet werden. Ein Beispiel für die Evolution eines Beispielbildes mit einspringendem schwarzen Dreieck und schwarzem Rechteckbalken auf weißem Hintergrund ist bei [Wei98, S. 117] zu finden. Das verrauschte Originalbild und das Ergebnis einer eigenen Rechnung basierend auf der variationellen FEM-Variante (1.8) des Perona-Malik-Ansatzes, nach partieller Integration mit zusätzlicher kontrastverstärkender rechten Seite[8], ist in Abbildung 1.3.1 zu sehen. V bezeichnet hier einen zum Ansatzraum passenden Testraum (zum Beispiel $H^1(\Omega)$ für schwache Lösungsspaare $(u, \partial_t u)$ in den Räumen $L^2(0,T;H^1(\Omega))$ und $L^2(0,T;H^{-1}(\Omega))$). Eine kurze Analyse der anisotropen Eigenschaften des Perona-Malik-Operators ist bei [Tsc02, Seite 39-40] zu finden:

$$\begin{aligned} \langle \partial_t u, \psi \rangle + \int_\Omega g(\|\nabla u\|^2) \cdot \nabla u \cdot \nabla \psi \, \mathrm{d}\vec{x} &= \int_\Omega f(u) \cdot \psi \, \mathrm{d}\vec{x} \quad \forall \psi \in V, \\ u(0,\cdot) &= I(\cdot) \qquad \text{in } \Omega, \end{aligned} \tag{1.8}$$

was in impliziter Form die Erfüllung einer homogenen Neumann-Randbedingung fordert.
Wie schon zuvor kommt die duale Paarung zur Anwendung. Im schwachen Fall erhält man damit $g(\|\nabla u\|^2) \cdot \partial_{\vec{n}} u = 0$ und nach Division durch $g(\|\nabla u\|^2)$ die Neumann-Randbedingung aus (1.7) allerdings nur für fast alle Punkte auf $\partial\Omega$.
Falls das zu entfernende Rauschen eine ähnliche Größenskala und lokale Gradientennormen wie Muster und Oberflächenstrukturen von dargestellten Objekten enthält, können diese nicht von Rauschen unterschieden werden und werden mit entfernt. Stark vereinfachend gesagt, verschwinden zuerst die Feinstrukturen des

[7] Zum Beispiel die von Weickert verwendete Funktion $g(s^2) = \left(1 + \frac{s^2}{\lambda^2}\right)^{-1}$, $\lambda > 0$ oder die von Perona-Malik vorgeschlagene Funktion $g(s^2) = \exp(-\frac{s^2}{K^2})$, $K > 0$.

[8] Ersetze 0 auf der rechten Seite durch (2.6), da ansonsten das Ergebnis gegen den Mittelwert strebt.

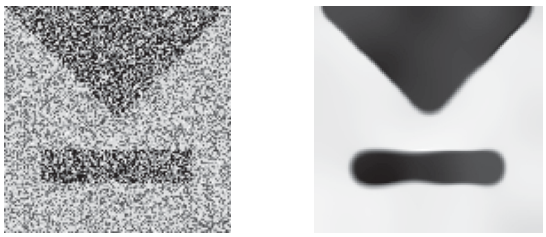


Abbildung 1.3.1: Verrauschtes Testbild und Ergebnis von Perona-Malik mit Kontrastverstärkung.

Bildes und später, mit wachsender Simulationszeit, Strukturen auf gröberen Skalen. Dies stellt für sich ein weiteres Problem in der Anwendung dar: *"Wie lange soll gerechnet werden?"*
Die optimale Zeit kann, je nach Rauschen und lokaler Bildstruktur, an verschiedenen Stellen des Bildes unterschiedlich sein. Unter Umständen müssen verschiedene Bereiche des Bildes getrennt mit unterschiedlichen Parametern behandelt und die Ergebnisse danach zusammengefügt werden.

1.3.2 Nichtlineare anisotrope Diffusion (NAD)

Wie im Abschnitt 1.3.1 behandelt, kann ein anisotropes, lokal an die Gegebenheiten ausgerichtetes, Verhalten des Diffusionsoperators von Vorteil bei der Erhaltung von wichtigen Bildstrukturen sein. Um dieses Verhalten besser zu kontrollieren, wird im folgenden Ansatz (1.9) die Diffusion durch das Einschieben einer positiv definiten und symmetrischen Matrixfunktion mit reellen Koeffizienten $A(\vec{v}, \nabla I)$ modifiziert, die Kontrolle über die Diffusionsstärke in einem lokalen Koordinatensystem geben soll:

$$\begin{aligned} \partial_t u - \operatorname{div}(A(\vec{v}, \nabla u)\nabla u) = 0 \qquad & \text{auf } \mathbb{R}^+ \times \Omega, \\ A(\vec{v}, \nabla u)\partial_n u = 0 \qquad & \text{auf } \mathbb{R}^+ \times \partial\Omega, \\ u(0, \cdot) = I(\cdot) \quad & \text{auf } \Omega. \end{aligned} \tag{1.9}$$

Bei der Verwendung des in (1.9) verwendeten Operators, wird über ein Vektorfeld lokal ein Orthonormalsystem erstellt, dessen erster Vektor in Richtung des Vektorfeldes $\vec{v}$ geht. Die Basisvektoren stehen als Spalten in der Matrix $B(\vec{v})$. Die Diffusionsstärke in Richtung des Vektorfeldes wird durch $\alpha(\|\vec{v}\|)$ und normal dazu durch $g(\|\nabla I\|)$ bestimmt. Diese Formulierung des Operators kann auch für Anwendungsfälle mit mehr als zwei Dimensionen verwendet werden. Die im Diffusionsoperator verwendete Matrix hat damit folgende Struktur:

$$A(\vec{v}, \nabla u) := B(\vec{v}) \begin{pmatrix} \alpha(\|\vec{v}\|) & 0 \\ 0 & g(\|\nabla u\|)\mathrm{Id}_{d-1} \end{pmatrix} B(\vec{v})^\top \tag{1.10}$$

Eine alternative Formulierung dieser Definition wäre

$$A(\vec{v}, \nabla u) := \alpha(\|\vec{v}\|)\frac{\vec{v}\vec{v}^\top}{\|\vec{v}\|^2} + g(\|\nabla u\|)\left(\mathrm{Id}_d - \frac{\vec{v}\vec{v}^\top}{\|\vec{v}\|^2}\right), \tag{1.11}$$

welche die Matrix $B(\vec{v})$ eliminiert.
Im Falle von konstanten Koeffizienten in der Matrix A, existieren positive Eigenwerte λ_1 und λ_2 und eine dazu gehörende Orthonormalbasis bestehend aus Eigenvektoren $\vec{v}_1$ und $\vec{v}_2$ der Länge 1. Sei nun $B = (\vec{v}_1\, \vec{v}_2)$, dann gilt

$$A = B \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} B^\top. \tag{1.12}$$

Oder anders ausgedrückt, nach einer Koordinatentransformation des Problems auf die vorher erwähnte Orthonormalbasis wird A zu $\tilde{A} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$.
Eine zusätzliche Skalierung der Koordinaten kann das Problem weiter transformieren und die Matrix zu $\bar{A} = \mathrm{Id}_d$ reduzieren. Dies bedeutet, dass in diesem sehr speziellen Fall ein anisotroper Diffusionsoperator äquivalent zu einem isotropen Diffusionsoperator auf einem transformierten Koordinatensystem, Rechengebiet beziehungsweise Gitter ist.
Wenn $A(\vec{v})$ nur vom Vektorfeld $\vec{v}$ abhängt, so kann man versuchen (zumindest lokal) eine ortsabhängige Schar von Orthonormalbasen und somit lokal unterschiedlicher Transformationen/Verzerrungen zu finden, welche dieses Problem äquivalent zu isotroper Diffusion macht. Im Falle der vollständigen Nichtlinearität ($A(\vec{v}, \nabla u)$) ist dies im Allgemeinen leider nicht mehr möglich, wodurch dieser Ansatz für eine weitere Untersuchung des NAD-Operators nicht zur Verfügung steht.
Die Nichtlinearität tritt, in der hier vorliegenden Form des Operators, zunächst in der Gestalt der Abhängigkeit der Diffusionsstärke (orthogonal zur Vektorfeldrichtung) von der Norm des Bildgradienten auf. Bei anderer Wahl der Funktionen α, g und zusätzlicher Koppelung durch $\vec{v} = \nabla u$, kann das Perona-Malik-Verfahren als Sonderfall von NAD mit $\alpha(s) = g(s)$ angesehen werden.[9] Andererseits kann die Funktion g auch durch eine Konstante ersetzt werden. Falls das Vektorfeld $\vec{v}$ unabhängig von u ist, wird das Problem damit vollständig linear.
Eine Anwendung dieses Operators liegt darin, den Anfangsdaten entlang der Stromlinien eines vorgegebenes Vektorfeld (das auch zeitabhängig sein kann!) sichtbaren linienartige Strukturen "aufzuprägen". Durch eine passende Wahl der Funktion g werden dabei die einzelnen Linien voneinander getrennt und feine Strukturen mit der Zeit dicker und grobskaliger.
Eine Anwendung in der Bildverarbeitung könnte darin liegen, während einer Vorverarbeitung lokale Strukturen zu entdecken und diese Informationen, als Vektorfeld aufbereitet, dann zur Steuerung des NAD-Operators bei der Entrauschung zu verwenden. Dies könnte zum Beispiel durch Ausrichtung der stärksten Diffusion entlang von Kanten erfolgen.
Zur numerischen Berechnung einer approximativen Lösung durch ein passendes Zeitschrittverfahren ist allerdings eine Linearisierung des Problems[10] nötig. Die

[9] **Achtung:** Die Funktion g bei Perona-Malik ist nicht identisch mit der hier verwendeten Funktion g! Bei Perona-Malik ist der Funktionsparameter $\|\nabla u\|^2$, hier jedoch $\|\nabla u\|$.

[10] Siehe Gleichung (1.13) mit $u_n := u(t_n, \cdot)$, $\vec{v}_n := \vec{v}(t_n, \cdot)$ und der Zeitschrittweite Δt_n im n-ten Zeitschritt.

Auswertung der Funktionen B, α und g finden dazu mittels Daten zum vorherigen Zeitschritt statt. Der Ablauf des so modifizierten Verfahrens besteht nun eigentlich aus zwei miteinander verbundenen Verfahren, deren Schritte abwechselnd ausgeführt werden. Zuerst werden mit dem ersten Verfahren vorsichtig erste Strukturen entdeckt. Zum Beispiel durch eine Glättung $\tilde{u}_n = K_\sigma * u_n$, was Strukturinformationen mittels $\nabla\tilde{u}_n$ liefert oder durch Auswertungen der Eigenvektoren und Eigenwerte des Strukturtensors $J_\rho(\nabla\tilde{u}_n)$ (siehe dazu Abschnitt 1.3.5). Danach wird mit dem zweiten Verfahren (der linearisierten NAD) kurze Zeit entlang dieser Strukturen diffundiert (zum Beispiel $\vec{v}_n = \nabla\tilde{u}_n$), danach diese diffundierten Daten erneut zur Strukturerkennung mit dem ersten Verfahren benutzt, und so weiter bis zum gewünschten Endzeitpunkt oder bis ein Abbruchkriterium erreicht wurde.
Da als Operator nicht $A(\vec{v}_{n+1}, \nabla\tilde{u}_{n+1})$ verwendet wird, liegt bei der verwendeten Zeitdiskretisierung nicht das implizite Euler-Verfahren vor, sondern eine semiimplizite Variante. Die Verwendung des Operators $A(\vec{v}_n, \nabla\tilde{u}_n)$ stellt bei ausreichend kleinem Zeitschritt aber einen relativ kleinen Fehler dar, der bei dem verwendeten Verfahren 1. Ordnung keine sichtbaren Abweichungen erzeugt, und somit für Zwecke der Bildbearbeitung und Visualisierung genau genug ist. Zudem wird ein zeitaufwendiges Fixpunktverfahren vermieden, das den Rechenaufwand zusätzlich explodieren lassen würde. Als Alternative könnte man aber auch völlig explizit arbeiten, man müsste aber in jedem Zeitschritt die passende CFL-Bedingung bestimmen, was jeweils eine Analyse des Operators $A(\vec{v}_n, \nabla\tilde{u}_n)$ über alle Punkte des Rechengebietes hinweg erfordert.

$$\begin{aligned} \frac{u_{n+1} - u_n}{\Delta t_{n+1}} - \operatorname{div}(A(\vec{v}_n, \nabla\tilde{u}_n)\nabla u_{n+1}) = 0 \qquad & \text{auf } \mathbb{R}^+ \times \Omega, \\ A(\vec{v}_n, \nabla\tilde{u}_n)\partial_n u_{n+1} = 0 \qquad & \text{auf } \mathbb{R}^+ \times \partial\Omega, \\ u(0,\cdot) = I(\cdot) \qquad & \text{auf } \Omega. \end{aligned} \tag{1.13}$$

Eine Methode, um solche Strukturen zu entdecken, besteht in der Berechnung des "Strukturtensors" und wird im Abschnitt 1.3.5 besprochen. Denkbare Heuristiken für Abbruchkriterien könnten anhand von Fourier- oder Wavelettransformation[11] der aktuellen Ergebnisse entwickelt werden. Das Verhältnis von niederen zu hohen Frequenzen könnte als Näherung des Signal-Rausch-Verhältnisses[12] benutzt werden, wobei sich bei der Berechnung auf unstrukturierten Gittern wieder neue Probleme ergeben. Diese Heuristiken sind natürlich nicht universell anwendbar, da sich zum Beispiel Bilder mit kleinkarierten Mustern sehr von Bildern mit sanfteren Strukturen (Nahaufnahme eines Blattes, Landschaftsbilder von Sanddünen in der Wüste, usw.) doch sehr in ihrer Frequenzverteilung unterscheiden. Diese Unterschiede können natürlich in unterschiedlichen Abschnitten ein und desselben Bildes auftreten. Eine Aufteilung und getrennte Behandlung dieser Bildteile kann unter diesen Umständen nötig werden.

[11] Zum Beispiel durch eine Projektion der Ergebnisse auf ein virtuelles Koordinatenraster mit Auswertung der so gewonnenen Punktdaten.

[12] SRV oder SNR im Englischen

1.3.3 Ein alternativer Ansatz für anisotrope Operatoren

Ein anderer Ansatz, der für mehrdimensionale Datensätze mit starker Korrelation (zum Beispiel Farbbilder) besser geeignet wäre, wurde von David Tschumperlé und Rachid Deriche am INRIA Odyssée Labor der Sophia-Antipolis in Frankreich entwickelt und untersucht.
Wie in [TD05b, Abschnitte 1.3 und 3.2] angeführt, ist die Ausrichtung der Diffusion nicht allein durch die angegebene Richtung bestimmt, sondern wird ähnlich wie beim Perona-Malik-Operator wegen der Abhängigkeit von lokalen Bildgradienten auch von diesen beeinflusst. Sei $\mathbf{D} := (d_{i,j})$ ein Divergenztensor, dann gilt:

$$\operatorname{div}(\mathbf{D}\nabla u) = \operatorname{trace}(\mathbf{D}\,\mathcal{H}(u)) + \nabla u^\top \overrightarrow{\operatorname{div}}(\mathbf{D}) \tag{1.14}$$

mit

$$\overrightarrow{\operatorname{div}}(\mathbf{D}) := \begin{pmatrix} \operatorname{div}\left((d_{1,1},\ d_{1,2})^\top\right) \\ \operatorname{div}\left((d_{2,1},\ d_{2,2})^\top\right) \end{pmatrix}.$$

Die bisherige NAD-Formulierung des Diffusionsoperators enthält also einen Term, der direkt vom lokalen Bildgradienten und der lokalen Änderung des Divergenztensors abhängt. Da letztere ebenfalls vom lokalen Bildgradienten abhängt, ergibt sich eine weitere indirekte Abhängigkeit vom lokalen Bildgradienten. Alternativ kann der Divergenztensor auch von den Spektralelementen des Strukturtensors (siehe Abschnitt 1.3.5) abhängen, wodurch größere Strukturen besser von lokalem Rauschen getrennt und somit das Ergebnis der Entrauschung verbessert wird.
Um eine bessere Kontrolle über die lokale Diffusionsrichtung zu erhalten, kann ein Wechsel zu einem Ansatz mittels einer Hessematrix versucht werden. Hier der Ansatz von David Tschumperlé[13] (Notation angepasst, $\vec{\xi} \perp \vec{\nu}$, $\|\vec{\xi}\| = \|\vec{\nu}\| = 1$):

$$\partial_t u = c_1 \frac{\partial^2 u}{\partial \vec{\xi}^2} + c_2 \frac{\partial^2 u}{\partial \vec{\nu}^2} = \operatorname{trace}(\mathbf{T}\,\mathcal{H}(u)) \tag{1.15}$$

mit $\mathbf{T} := c_1 \cdot \vec{\xi}\vec{\xi}^\top + c_2 \cdot \vec{\nu}\vec{\nu}^\top$. Dabei ist $\frac{\partial^2}{\partial \vec{\xi}^2}$ die 2. Richtungsableitung in Richtung $\vec{\xi}$.
Die auf mehrdimensionale Datenfelder[14] erweiterte Variante (siehe [TD05b, Abschnitt 4]) für ein solches Datenfeld $\vec{u}$ mit Komponenten u_i und $\vec{u}(0,\cdot) = \vec{I}(\cdot)$ lautet entsprechend:

$$\partial_t \vec{u} = \sum_{j=1}^{n} \operatorname{trace}(\mathbf{A^{ij}}\,\mathcal{H}(u_i)) \quad (i = 1..n) \tag{1.16}$$

mit den Diffusionstensoren A^{ij}, die auch alle NAD-Operatoren sein können.
In [TD05b, Abschnitte 3.1 und 6] wurde ein interessanter Ansatz dazu vorgestellt, der anstatt mit finiten Differenzen, durch Filterung mit lokal variierende Masken beziehungsweise Faltung mit orientierten normalisierten Gaußkernen berechnet

[13] Notation angepasst, Formel (7) aus der Veröffentlichung für den Spezialfall einer Komponente.

[14] Ein Beispiel sind Farbbilder im RGB, HSV, YCbCr oder anderen Farbräumen.

wird, was keine 2. Ableitungen mehr benötigt und das Maximumsprinzip erhält. Der Nachteil liegt dabei im immensen Rechenaufwand, da für jeden Bildpunkt und für jeden Zeitschritt ein eigener Gaußkern zu Faltung berechnet werden muss. Um auch hierfür eine Berechnung mit bilinearen finiten Elementen und somit einer Implementierung in den hier verwendeten numerischen Werkzeugen möglich zu machen, ist ein eine schwache Formulierung des Problems mit (schwachen) Ableitungen höchstens 1. Ordnung nötig.[15] Ein einfacher variationeller Ansatz mit nachfolgender Umformulierung mittels partieller Integration wäre:

$$0 = \int_\Omega (\partial_t u - \operatorname{trace}(\mathbf{T}\,\mathcal{H}(u))) \cdot \phi \,\mathrm{d}\vec{x} \tag{1.17}$$

$$= \int_\Omega \left(\partial_t u + \nabla u^\top \overrightarrow{\operatorname{div}}(\mathbf{T}) - \operatorname{div}(\mathbf{T}\nabla u)\right) \cdot \phi \,\mathrm{d}\vec{x} \tag{1.18}$$

$$= \int_\Omega \left(\partial_t u + \nabla u^\top \overrightarrow{\operatorname{div}}(\mathbf{T})\right) \cdot \phi + (\mathbf{T}\nabla u) \cdot \nabla\phi \,\mathrm{d}\vec{x} - \int_{\partial\Omega} (\mathbf{T}\nabla u) \cdot \phi \,\mathrm{d}\vec{n} \qquad \forall \phi \in V. \tag{1.19}$$

Der durch die partielle Integration erzeugte Randterm kann weggelassen werden, wenn Testräume mit einer wohldefinierten Spur auf $\partial\Omega$ und der Eigenschaft $\phi(\vec{x}) = 0\ \forall \vec{x} \in \partial\Omega$ verwendet werden. In der schwachen Form (1.19) werden (schwache) 1. Ableitungen für die Koeffizienten des Divergenztensors $\mathbf{T}$ benötigt, was etwas höhere Anforderungen an den Raum stellt, aus denen die Komponentenfuntionen des Divergenztensors stammen. Wegen der Ähnlichkeit der DGL mit parabolischen Gleichungen zweiter Ordnung, sollten bei geeigneten Randbedingungen, Startwerten und unter Vermeidung von Singularitäten, folgenden Eigenschaften der Lösungen parabolischer Gleichungen zweiter Ordnung zu erwarten sein:

- Starke Lösungen u der DGL (1.15) beziehungsweise (1.17) sollten bei wohldefiniertem Diffusionstensor[16] mindestens einmal stetig in der Zeit ableitbar, zweimal stetig ableitbar im Raum und stetig bis zum Rand sein. Oder anders formuliert: $u \in C_1^2(\Omega \times (0, t_{max})) \cap C(\bar{\Omega} \times (0, t_{max}))$ mit Testraum $V = \{v \in C^\infty(\Omega \times (0, t_{max})) \cap C(\bar{\Omega} \times (0, t_{max})) | v(x) = 0\, \forall x \in \partial\Omega\}$. (Notation der Räume aus [Eva98, S. 54-55].)

- Schwache Lösungen sollten (bei genügender Regularität der Start- und Randwerte) als Paar von Abbildungen $u : (0; t_{max}] \to H_0^1(\Omega)$ und $u_t : (0; t_{max}] \to H^{-1}(\Omega)$ vorliegen, für die $u \in L^2(0, t_{max}; H_0^1(\Omega))$ und $u_t \in L^2(0, t_{max}; H^{-1}(\Omega)$ gilt. Der dafür verwendete Testraum ist $V = H_0^1(\Omega)$. (Notationen aus [Eva98, S. 351-352].)

[15] Das Numerikpaket FEAST stellt auch finite Elemente höherer Ordnung bereit. Eine Diskretisierung mit Elementen 2. Ordnung wäre auch denkbar.

[16] Symmetrisch, positiv definit, koerzitiv und einmal stetig ableitbaren Komponentenfunktionen auf $\bar{\Omega}$ im hier vorliegenden Anwendungsfall.

Es wurde allerdings nicht ausführlich untersucht, unter welchen Umständen sich diese Eigenschaften wirklich auf Lösungen von (1.15) und (1.17) übertragen. Daher sind diese Angaben nur unter Vorbehalt und erfordern wahrscheinlich für jede konkrete Wahl der Komponentenfunktionen eine eigene Überprüfung der Regularitätseigenschaften. Eine Gruppierung von Problemen mit ähnlichen Komponentenfunktionen in Problemklassen könnte dies vielleicht vereinfachen.

1.3.4 Anmerkungen zur Implementierung

Bei der numerischen Implementierung einer FEM-Methode zur Lösung des Problems (1.19), kann man bei der Assemblierung der Diffusionsmatrix jeweils alle Ableitungen für Koeffizienten des Diffusionstensors punktweise neu berechnen, was eine punktweise Berechnung der Ortsableitungen der $d_{i,j}$ Komponenten des Divergenztensors $\mathbf{D}$ erfordert oder man kann versuchen, auch die Kompomenten des Divergenztensors als FEM-diskretisierte Datenfelder anzulegen und die Ableitungen mittels der FEM-Diskretisierung als Linearkombinationen der Ableitungen der FEM-Ansatzfunktionen abzulesen. Letztere Methode wurde von mir im schwierigeren NATD-Fall (Gleichung (2.4)) anhand eines Testbeispiels mit zufriedenstellenden Ergebnissen ausprobiert und scheint für Bildverarbeitungszwecke genau genug zu sein. Für Divergenztensoren, deren Komponentenfunktionen nur schwach ableitbar sind, sind Fälle denkbar, für die deren Ableitungen nicht in allen Punkten klar definiert sind. In solchen Fällen ist bei ausreichend vorhandenem Speicher immerhin noch die zweite Methode anwendbar.

1.3.5 Der Strukturtensor

Da sowohl Strukturen im Originalbild, als auch das zugefügte Rauschen, große Gradienten erzeugen, ist es immer schwer, anhand gestörter lokaler Informationen, größere Strukturen wie Kanten, Ecken und Punkte korrekt zu identifizieren und bei der Rekonstruktion zu erhalten. Verfahren, die nur die lokalen Informationen betrachten, werden daher immer dazu neigen Bilddetails "abzuschleifen" und kleinskalige Strukturen nach und nach zu entfernen.
Aus diesem Grund ist es von Vorteil, Informationen aus einem größeren Umfeld mit einzubeziehen und somit den Einfluss des Rauschens auf die Erkennung zu vermindern. Dies kann über die Untersuchung der Eigenwerte des sogenannten "Strukturtensors" erfolgen (vergleiche auch [Wei98, S. 56–57]).

Definition 1.3.1. Sei u_σ das Ergebnis der Faltung von $\tilde{u}$ mit einem Gaußkern K_σ mit Varianz σ auf $\mathbb{R}^2$ mit anschließender Einschränkung von u_σ auf Ω. $\tilde{u}$ ist dabei die Erweiterung von u auf $\mathbb{R}^2$ als Funktion mit beschränktem Träger auf Ω:

$$\tilde{u}(t,\vec{x}) := \begin{cases} u(t,\vec{x}) & \text{für } \vec{x} \in \Omega \\ 0 & \text{sonst.} \end{cases} \tag{1.20}$$

Damit ist $u_\sigma : [0; t_{max}] \times \mathbb{R}^2 \to \mathbb{R}$ definiert als

$$u_\sigma(t, \vec{x}) := (K_\sigma * \tilde{u})(t, \vec{x}) = \int_{\mathbb{R}^2} K_\sigma(\vec{x}-\vec{y})\tilde{u}(t, \vec{y})\,\mathrm{d}\vec{y} = \int_{\Omega} K_\sigma(\vec{x}-\vec{y})u(t, \vec{y})\,\mathrm{d}\vec{y} \quad \forall \vec{x} \in \Omega \tag{1.21}$$

Anmerkung. *Falls für festes $t \in [0; t_{max}]$ gilt, dass $\sup_{\vec{x}\in\Omega} \|u(t, \vec{x})\| < \infty$ und integrierbar ist, gilt $u_\sigma(t, \cdot) \in C^\infty(\Omega)$, somit ist auch $\nabla u_\sigma(t, \cdot)$ auf ganz Ω definiert. Dies folgt daraus, dass sich alle Ableitungen in das Integral hineinziehen lassen. Somit sind die Ableitungen von $u_\sigma(t, \cdot)$ Faltungen von $u(t, \cdot)$ mit den Ableitungen von $K_\sigma \in C^\infty(\mathbb{R}^2)$.*

Definition 1.3.2. Der Strukturtensor der Dimension n sei die positiv-semidefinite symmetrische Matrix $J_\rho(\nabla u_\sigma)$:

$$J_\rho(\nabla u_\sigma) := K_\rho * (\nabla u_\sigma \nabla u_\sigma{}^\top) = \begin{pmatrix} j_{1,1} & \cdots & j_{1,n} \\ \vdots & \ddots & \vdots \\ j_{n,1} & \cdots & j_{n,n} \end{pmatrix}. \tag{1.22}$$

Im Weiteren werden nur Strukturtensoren mit $n = 2$ betrachtet.
Das Ziel ist, den Strukturtensor zu Erkennung von relevanten Bildstrukturen einer gewissen Detailgröße zu benutzen, während viel kleinere Details und Rauschen möglichst ignoriert werden sollen. Die Parameter σ und ρ bezeichnen zwei verschiedene Skalen, die in die Bildung des Strukturtensors einfließen. Es sollte dabei immer $\sigma \ll \rho$ sein.
Details und Rauschen einer Größenordnung unterhalb der Varianz σ werden durch die Faltung von u mit K_σ "geglättet" und nebenbei ∇u einer Regularisierung unterzogen. Die Eigenvektoren und Eigenwerte von $J_0(\nabla u_\sigma) := \nabla u_\sigma \nabla u_\sigma{}^\top$ liefern *erweiterte lokale Informationen* über die Richtung mit der stärksten Graustufenänderung (minus Rauschen) beziehungsweise über die bevorzugte lokale Ausrichtung (*Koherenzrichtung*), was zum Beispiel ein Hinweis auf eine potentielle Kante sein kann. Mit derart lokalisierten Informationen lassen sich allerdings schwerlich Kanten oder Ecken sicher identifizieren, da nicht sicher ist, ob die Ausrichtung Teil einer größeren Struktur ist oder nur eine lokale Störung der Daten widerspiegelt. Details zur Berechnung und Klassifizierung der Eigenwerte in verschiedene Fälle (Kanten, Ecken und Punkte) sind in [Wei98, S. 56–57] zu finden.
Sollen Bilddetails einer Größe um ρ erkannt werden, so müssen die Daten innerhalb einer "Nachbarschaft" der Größe ρ betrachtet werden. Dieser gewichtete Mittelwert der Koherenzrichtung wird durch die abschließende Faltung von J_0 mit K_ρ erreicht. Falls u schon rauschfrei ist, so kann $\sigma = 0$ gesetzt werden, das heißt ∇u_σ durch ∇u ersetzt werden. Die aus J_ρ gewonnene Koherenzrichtung kann anstelle von $\nabla u^\top$ auch zur Bildergänzung eingesetzt werden (vergleiche auch mit Abschnitt 1.4.1).
Durch den Ansatz $\det(J_\rho - \mu\,\mathrm{Id}) = 0$ erhält man die beiden Eigenwerte $\mu_{1,2}$

$$\mu_{1,2} = \frac{j_{11} + j_{22}}{2} \pm \sqrt{\frac{(j_{11} - j_{22})^2}{4} + {j_{12}}^2}, \tag{1.23}$$

und durch Lösen von $(J_\rho - \mu\,\mathrm{Id})\vec{v} = 0$ und Auswählen von Lösungsvektoren der Länge 1 erhält man die zugehörigen orthonormalen Eigenvektoren $\vec{v}_1$ und $\vec{v}_2$, welche folgende Kolinearitätseigenschaften aufweisen:

$$\vec{v}_1 \| \begin{pmatrix} 2j_{12} \\ j_{22} - j_{11} + \sqrt{(j_{11} - j_{22})^2 + 4j_{12}^2} \end{pmatrix} \tag{1.24}$$

$$\vec{v}_2 \| \begin{pmatrix} j_{22} - j_{11} + \sqrt{(j_{11} - j_{22})^2 + 4j_{12}^2} \\ -2j_{12} \end{pmatrix}. \tag{1.25}$$

Die Orthogonalität der je von $\vec{v}_1$ und $\vec{v}_2$ aufgespannten Eigenräume kann im Fall $\mu_1 \neq \mu_2$ durch die Symmetrie von J_ρ begründet werden. Im Fall $\mu_1 = \mu_2$ gilt:

$$\begin{pmatrix} 2j_{12} \\ j_{22} - j_{11} \end{pmatrix} \perp \begin{pmatrix} j_{22} - j_{11} \\ -2j_{12} \end{pmatrix} \Rightarrow \vec{v}_1 \perp \vec{v}_2.$$

Wird der Diffusionsoperator A in Abschnitt 1.3.2 beziehungsweise im Problem (1.9) in Abhängigkeit vom Strukturtensor $J\rho$ ($\rho \geq 0$) gewählt, so gibt es ein theoretisches Resultat in [Wei98, S. 58], welches Wohlgestelltheit, Regularität und ein Extremumsprinzip für die Lösung von (1.9) liefert. Dafür werden neben $I \in L^\infty(\Omega)$, mit $\sigma > 0$ regularisiertem ∇u, $T > 0$, $a := \operatorname{ess\,inf}_\Omega(I)$ und $b := \operatorname{ess\,sup}_\Omega(I)$ zusätzlich drei Vorbedingungen an den Diffusionsoperator A gestellt:

(C1) Glattheit: Für $A : \mathbb{R}^{2\times 2} \to \mathbb{R}^{2\times 2}$ sei
$A \in C^\infty(\mathbb{R}^{2\times 2})$.

(C2) Symmetrie:
$A(J)$ symmetrisch für alle symmetrischen Matrizen $J \in \mathbb{R}^{2\times 2}$.

(C3) Uniforme positive Definitheit:
Für alle $w \in L^\infty(\Omega, \mathbb{R}^2)$ mit $|w(\vec{x})| \leq K$ auf $\bar{\Omega}$ existiert eine positive untere Grenze $\nu(K)$ für die Eigenwerte von $A(J_\rho(w))$.

Unter diesen Bedingungen liefert das Theorem 1 aus [Wei98, S. 58][17] die folgenden Aussagen : Es existiert eine eindeutige (schwache) Lösung $u(t, \vec{x})$ der (analog zu (1.8)) variationellen schwachen Formulierung des Problems (1.9) mit

$$u \in C([0,T]; L^2(\Omega)) \cap L^2(0,T; H^1\Omega) \tag{1.26}$$

$$\partial_t u \in L^2(0,T; H^1\Omega). \tag{1.27}$$

Zudem gilt $u \in C^\infty((0,T] \times \bar{\Omega})$. Diese Lösung hängt stetig von I bezüglich der Norm $\|\cdot\|_{L^2(\Omega)}$ ab und erfüllt das Extremumsprinzip: $\exists a, b \in \mathbb{R}$ mit

$$a \leq u(t, \vec{x}) \leq b \quad \text{auf } \Omega \times (0,T]. \tag{1.28}$$

Mit $\rho = 0$ ergibt sich als Sonderfall auch eine Aussage für Anwendungen ohne die Verwendung des Strukturtensors.

[17] Notation wurde angepasst.

1.3.6 Beispiele

Man kann selbst ohne Verwendung des Strukturtensors gute Resultate mit (1.9) erhalten. In den Abbildungen 1.3.2 und 1.3.3 wurden zwei verrauschte Testbilder[18] mit einem kantenverstärkenden Diffusionsoperator nach [Wei98, S. 114][19] entrauscht. Die verwendete Diffusionsfunktion $g(\cdot)$ in (1.10) lautet:

$$g(s) := \begin{cases} 1 & \text{falls } s \leq 0 \text{ ist,} \\ 1 - \exp\left(\frac{-C_m}{(s/c)^m}\right) & \text{falls } s > 0 \text{ ist.} \end{cases} \text{ für } C_m > 0,\ c{>}0 \tag{1.29}$$

Für $m = 4$ lautet $C_4 = 3.31488$.
Das Problem (1.9) wurde für die numerische Berechnung linearisiert, mit implizitem Euler auf bilinearen finiten Viereckselementen (Q_1-Ansatzraum) diskretisiert und mit den Zeitschrittweiten $\Delta t = 10^{-2}$ und $\Delta t = 10^{-3}$ auf einem strukturierten Gitter gerechnet, bis das Rauschen aus der Lösung verschwunden war. Die Lösung wurde dabei jeweils durch Lösen eines isotropen Diffusionsproblems regularisiert. Zur weiteren Vereinfachung wurde die Randbedingung in (1.9) durch eine einfacher zu implementierende Dirichlet-Randbedingung ersetzt, welche die Startlösung am Rand festhält.

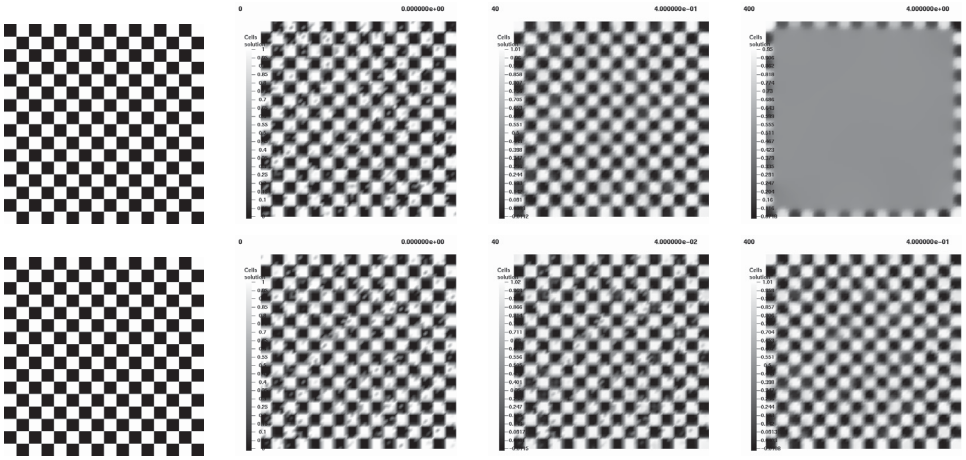


Abbildung 1.3.2: Entrauschung eines Schachbrettmusters (Original ganz links) mit den Zeitschrittweiten $\Delta t = 10^{-2}$ (oben) und $\Delta t = 10^{-3}$ (unten). ($c = 1$)

Die Ideen hinter der speziellen Auswahl der Testbeispiele waren folgende:

Schachbrettmuster: Um die Auswirkung der Entrauschung auf Bilddetails mit fester Skalengröße zu verdeutlichen. Wie in Abbildung 1.3.2(oben) zu sehen ist, können diese Details durch eine zu große Zeitschrittweite oder zu starke Diffusion ausgelöscht werden.

Zielscheibe: Die einzelnen Kreise testen, ob die Krümmung von Rändern einen Einfluss auf das Ergebnis hat. Die Krümmung von Kreisen ist, wie bekannt

[18] Es wurde sogenanntes "Salz-und-Pfeffer"-Rauschen verwendet.
[19] Es wurde mit $m = 4$ die gleiche Diffusionsfunktion gewählt, die auch Weickert anwendet.

sein dürfte, $\kappa = r^{-1}$ mit r :=Kreisradius. Zudem sind durch die Innen- und Außenränder der Kreisringe jeweils konvex und konkav gekrümmte Ränder vorhanden. Wie in Abbildung 1.3.3 zu sehen ist, bleibt die Form der Ränder ausreichend gut erhalten. Ein von der Krümmung abhängender Einfluss auf die Diffusionsstärke ist dabei nicht erkennbar.

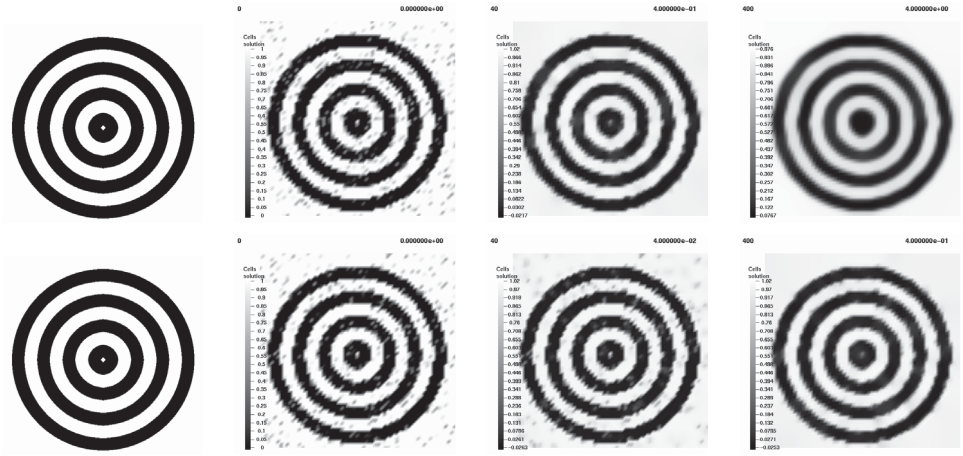


Abbildung 1.3.3: Entrauschung eines Zielscheibenmusters (Original ganz links) mit den Zeitschrittweiten $\Delta t = 10^{-2}$ (oben) und $\Delta t = 10^{-3}$ (unten). ($c = 1$)

Bei der Anwendung des NAD-Verfahrens muss immer darauf geachtet werden, dass man nicht zu lange rechnet, da die Lösung ohne einen entsprechenden Gegeneinfluss, wie zum Beispiel eine kontrastverstärkende rechte Seite oder einen Strafterm ($\delta(I - u)$, $\delta > 0$) auf der rechten Seite, für $t \to \infty$ gegen den Mittelwert der Startlösung I strebt. Auch wenn die Randwerte fest vorgegeben wurden, strebt die Lösung im Inneren gegen diesen Mittelwert (siehe Abbildung 1.3.2 oben rechts). Ein Nachweis dieses Verhaltens kann bei [Wei98, S. 67-72] gefunden werden. Dort wurde zusätzlich die anisotrope nichtlineare Diffusion mit Lyapunov-Funktionalen in Verbindung gebracht. Dies wurde später auch in [RSW00] vertieft, indem zuerst Fälle mit nichtlinearer isotroper Diffusion in Bezug zu Lyapunov-Funktionalen gesetzt und damit Beschränkungen und Abklingverhalten ([RSW00, Seite 103-107]) hergeleitet werden. In [RSW00, Seite 107-108] werden die Fälle nichtlinearer anisotroper Diffusion (und Fälle nichtkonvexer Diffusionsfunktionen) durch eine Serie von quadratischen Regularisierungsfunktionalen approximiert, wodurch die in [RSW00, Seite 103-107] zuvor hergeleiteten Resultate auch hierfür angewendet werden können.

1.4 Bildergänzung ("Inpainting")

Die Ergänzung von fehlenden Bildteilen unterscheidet sich von der Bildentrauschung dadurch, dass nur auf Teilbereichen des Bildes operiert wird und im Inneren der zu reparierenden Bereiche ("Löcher") keinerlei oder nur stark beschädigte

Daten vorliegen. Zur Ergänzung stehen somit nur Informationen aus Bildbereichen abzüglich der Löcher zu Verfügung. Ein "korrekt" repariertes beziehungsweise ergänztes Bild sei nun eines, bei dem diese Lücken möglichst "natürlich" im Bezug zum restlichen Bild aufgefüllt wurden. Das bedeutet, dass die aufgefüllten Inhalte möglichst visuell unauffällig und von derselben Struktur wie die Umgebung sein sollten.
Manche Methoden versuchen, diese Löcher mit Texturen oder vorhandenen Bildbausteinen des intakten Bildes auszubessern. Andere wiederum versuchen Strukturen, wie zum Beispiel Linien und Farbflächen, die an die Ränder der Lücken stoßen, über die Lücken hinweg miteinander zu verbinden.
Eine mögliche Methode[20] zerteilt dabei zuerst das Bild in Kacheln fester Größe. Danach wird anhand einer passenden Metrik die beste Übereinstimmung zwischen Randstellen und Kachel gefunden und fehlende Bildinformationen durch die der Kachel ersetzt. Dieser "Suche und Ersetze"-Schritt wird solange ausgeführt, bis alle Lücken geschlossen wurden. Diese Methode erfordert am Anfang eine sehr aufwendige Suche der besten Kachel bezüglich aller Möglichkeiten, eine Kachel an den Rand anzufügen. Nachdem der Rand durch das Einfügen der Kachel verändert worden ist, können theoretisch die Suchergebnisse für Einfügepositionen, welche keinerlei Überschneidung mit dem geänderten Randabschnitt aufweisen, zur Verbesserung der Laufzeit wiederverwendet werden. Diese Methode ist besonders zur Rekonstruktion von Mustern und texturierten Objekten geeignet, da diese im Allgemeinen eine periodische Struktur aufweisen und somit fehlende Musterteile durch passende Stücke, die an anderer Stelle vorhanden sind, wieder vervollständigt werden können.
Die hier vorgestellte Methode[21] verwendet einen ähnlichen Ansatz wie [YA07] und benutzt lediglich die Informationen in der direkten Umgebung des Randes der Fehlstellen für die Bildergänzung und kommt ohne Suchen aus. Stattdessen werden die Bildinformationen in eine inkompressible Strömung verwandelt und simuliert, wie sich diese innerhalb der Löcher verhält. Entgegen der von Bayumy A Youssef verwendeten Anwendung der Methode im Rahmen einer Wirbelstromformulierung des Navier-Stokes-Problems, wird hier direkt die gewöhnliche Navier-Stokes-Formulierung des inkompressiblen Strömungsproblems (1.30) und (leicht modifizierte) numerische Programme der FeatFlow-Programmsuite verwendet, welche dieses Problem lösen können.
Diese Methode kann aber keine Texturen fortsetzen, sondern zielt auf die Generierung eines glatten Übergangs zwischen Randteilen und der Rekonstruktion gekrümmter Ränder ab. Falls sich die Löcher in einer stark texturierten Region befinden, ist zu erwarten, dass die erzeugten weichen Übergänge als sehr auffällig wahrgenommen werden. Dies führt zur Vermutung, dass sich diese Methode eher zur Beseitigung kleinerer Kratzer eignet, als dazu größere Bildteile zu ersetzen.
Aus Gründen der Klarheit wird die verwendete Variante der inkompressiblen

[20] Vorgeführt von Dr. Bayumy A Youssef, Informatics Institute Moubark City for Scientific research, Alexandria, Ägypten, während eines Besuches an der TU Dortmund.

[21] Ebenfalls vorgeschlagen von Bayumy A Youssef, Informatics Institute Moubark City for Scientific research, Alexandria, Ägypten

stationären Navier-Stokes Gleichung mit Geschwindigkeitsfeld $\vec{v}$, Druck p und Dirichlet-Randwerten $\vec{g}$ hier explizit angegeben:

$$\begin{cases} (\vec{v} \cdot \nabla)\vec{v} - \nu\Delta\vec{v} - \nabla p = 0 \\ \operatorname{div} \vec{v} = 0 \\ \vec{v}_{|\partial\Omega} = \vec{g} \end{cases} \tag{1.30}$$

1.4.1 Bilder und Stromfunktionen

Im Rahmen dieser Arbeit wurden hauptsächlich Graustufenbilder untersucht. Diese lassen sich jeweils als eine Funktion $I : \Omega \to [0; 1]$ darstellen. Betrachten wir nun eine Kante als typische Bildstruktur und unterbrechen diese mit einem Loch.

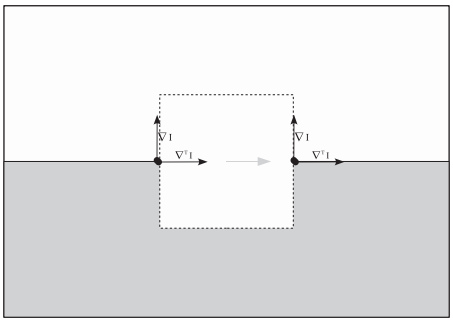


Abbildung 1.4.1: Eine Kante unterbrochen durch ein quadratisches Loch (gestrichelt eingezeichnet).

Wird die Normale $\nabla^\perp I := (\frac{\partial I}{\partial x_2}, -\frac{\partial I}{\partial x_1})^\top$ des Gradienten ∇I gebildet, so läuft diese entlang dieser Kante und es gilt $\|\nabla^\perp I(x)\| = \|\nabla I(x)\| \ \forall x \in \Omega$. Die Reparatur dieser Kante entspricht einer Verbindung der Kantenenden innerhalb des Loches, welche einen Übergang der Normalenvektoren auf beiden Seiten des Loches liefert. Interpretiert man nun diese Normalenvektoren als Strömungsgeschwindigkeit $\vec{v} := \nabla^\perp I$, so entspricht die Bildfunktion I (bis auf eine Konstante) der sogenannten "Stromfunktion" Ψ der Strömung v. Diese Ideen stammen aus den Arbeiten von Bertalmío, Bertozzi und Sapiro bezüglich "Image Inpainting" (siehe [BBS02]).
Durch die Verwandlung von I in ein Strömungsfeld $\vec{v}$ erhalten wir nun Strömungsdaten am Rande des Loches, die wir im Inneren des Loches ergänzen müssen. Da punktweise $\operatorname{div} \vec{v} = \partial_x(\partial_y I) + \partial_y(-\partial_x I) = 0$ gilt, ist auch das Integral der Flüsse über den Rand jeder einzelnen Fehlstelle gleich Null[22]. Somit bereiten die Strömungsdaten am Rand in diesem Punkt keine Probleme, die zu ergänzende Strömung als inkompressibel zu modellieren, wodurch eine Navier-Stokes-Formulierung des Problemes mit Dirichlet-Randbedingungen naheliegend ist. Zudem ist eine stationäre Strömung erwünscht, aber wohl nicht für alle Fälle erreichbar. Details dazu werden im Abschnitt 1.4.2 beschrieben. Das Bildergänzungspro-

[22] Errechnen der Strömung über den Rand des Loches mittels der Gauss-Formel.

blem wird in ein Strömungsproblem umgedeutet, welches dann mit Programmen zur Lösung von Navier-Stokes-Problemen berechnet werden kann.
Nachdem die Strömung berechnet wurde, muss nur noch die zugehörige Stromfunktion berechnet werden. Diese entspricht nun, bis auf eine Konstante, der gesuchten Bildergänzung. Diese Konstante kann durch Vergleich der errechneten Stromfunktion mit den vorhandenen Bildinformationen am Rande errechnet werden. Dies kann zum Beispiel durch die Differenz eines beliebigen Bildwertes am Rande mit der dort errechneten Stromfunktion oder durch einen Ansatz als Fehlerminimierer bezüglich einer passenden Norm über den Rand der Fehlstelle erfolgen. Letztere Methode sollte wegen des zusätzlichen Aufwandes nur angewandt werden, sollte es zu sichbaren Artefakten durch numerische Fehler bei der Rekonstruktion der Stromfunktion aus der errechneten Strömung kommt.
Zu Bildverarbeitungszwecken hat sich die einfache Differenz bei den in Abschnitt 1.4.4 angeführten Beispielproblemen als völlig ausreichend gezeigt (siehe Abbildungen 1.4.11, 1.4.16 und 1.4.18). Daher wurde auf eine Implementierung der letzteren Methode verzichtet.
Die hier vorgestellte Rekonstruktionsmethode hat das inherente Problem, dass nur die Gradienteninformation auf dem Rand der Fehlstelle zur Rekonstruktion herangezogen wird, welche höchstens Informationen über eine zwei Pixel tiefe Schicht um die Fehlstellen beinhalten. Die so nutzbare Informationsmenge ist in ihrem Wachstum linear zur Länge des Randes (bemessen in Pixeln) beschränkt. Die zu rekonstruierende Bildinformation ist in ihrem Wachstum aber nur linear zur Fläche (bemessen in Pixeln der Fehlstelle) beschränkt. Dies geht gut für Probleme, bei denen das Verhältnis der Fläche der Fehlstelle zur Länge des Randes klein bleibt oder die Fehlstelle auch mit Bildinformation von geringer Informationsmenge ausgebessert werden kann. In anderen Worten, für dünne Kratzer entgegen der lokalen Bildstrukturen, kleine fast punktförmige Löcher oder Bereiche mit sehr geringer Variation.

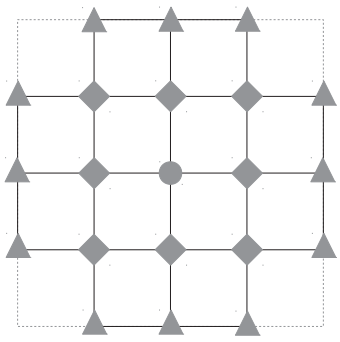


Abbildung 1.4.2: Ein zu rekonstruierendes Pixel (Kreis), Pixel auf dem Rand der Fehlstelle (Rauten) und zusätzliche Pixel, welche die Gradienteninformationen auf dem Rand der Fehlstelle beieinflussen können (Dreiecke).

Um die Machbarkeit der Bildergänzung abzuschätzen, betrachte man die Situation nun genauer. Jedes Graustufenpixel kann, wenn 8 Bits verwendet werden, $M = 256$ verschiedene Zustände annehmen. Wenn das Innere der Fehlstelle N_F

verschiedene Pixel enthält, so kann das Innere der Fehlstelle genau M^{N_F} verschiedene Zustände annehmen. Die Gradienten entlang des Randes hängen von den Pixel am Rand und den restlichen Pixeln der Zellen ab, welche die jeweilige Randkante enthalten. Enthält der Rand N_R Pixel, so kann die Zahl der, die Randdaten beeinflussenden Zellen N_D, durch

$$N_R \leq N_D \leq \frac{3}{2} \cdot N_R$$

beschränkt werden. Dies kann aus der Abbildung 1.4.2 des Extremfalles eines einzelnen Pixels abgeleitet werden.[23] Die Gradienten am Rande der Fehlstelle können höchstens M^{N_D} verschiedene Zustände annehmen. Damit ergibt sich eine obere Schranke für die Anzahl der rekonstruierten Inhalte der Fehlstellen. Wird nun das Verhältnis der möglichen Zustände des Inneren der Fehlstelle zu den möglichen Randdaten größer als 1, so gibt es Bilder, die keinesfalls perfekt rekonstruiert werden können. Sei nun $\rho := \frac{M^{N_F}}{M^{N_D}} = M^{N_F - N_D}$ mit der Abschätzung:

$$\frac{M^{N_F}}{M^{1.5 \cdot N_R}} \leq \rho = \frac{M^{N_F}}{M^{N_D}} \leq \frac{M^{N_F}}{M^{N_R}}$$

Durch eine Logarithmierung zur Basis M ergibt sich daraus:

$$N_F - 1.5 \cdot N_R \leq \rho_M := \log_M \rho = N_F - N_D \leq N_F - N_R$$

Die Schranke $\rho \leq 1$ kann nun als $\rho_M \leq 0$ beziehungsweise als $N_F \leq N_D$ umformuliert werden. Die Beschränkungen von N_D durch N_R liefern dabei die einfacher auszurechnende hinreichende Bedingung $N_F \leq N_R$ für die Erfüllung dieser Schranke.
Ein weiteres Problem liegt darin, dass die rekonstruierten Informationen sehr stark von der Güte der angegebenen Gradientendaten abhängen. Auch ein kleiner Fehler bei der Gewinnung der Gradientendaten kann zu sichtbaren Artefakten führen. Eine besondere Sorgfalt bei der Gewinnung guter gitterunabhängiger Gradientendaten ist daher angebracht!
Der hier verwendete Ansatz hat zudem die Eigenschaft, dass die Ergebnisse nicht symmetrisch gegenüber Invertierungen sind. Dies bedeutet, wenn das in Abbildung 1.4.1 gezeigte Bild invertiert, rekonstruiert und danach wieder invertiert wird, wird das Ergebnis im Allgemeinen nicht der direkten Rekonstruktion des Bildes entsprechen. Ein Beispiel dafür wäre, dass eine scharfe Bildkante, die einem in die Fehlstelle zeigenden Strömungsvektor entspricht, eine Jetströmung induziert, in der inversen Situation, welche einem aus der Fehlstelle zeigenden Strömungsvektor entspricht, aber einen Sog induziert, der Material aus der gesamten Umgebung ansaugt und somit ein völlig anderes Strömungsverhalten aufweist. Die so erzeugten Jetströmungen entsprechen Fortsetzungen der Bildkanten in die Fehlstelle hinein, die durch Diffusionsprozesse verwischt werden. Ist die

[23] In diesem Extrembeispiel ist $N_R = 8$ und $N_D = 12$. Die 4 Zellen mit dem fehlerhaften Pixel als Ecke können korrumpierte Informationen enthalten und müssen ignoriert werden!

zu überbrückende Fehlstelle zu lang, so kann sich, je nach Reynoldszahl des Problems, diese Jetströmung langsam auffächern (kleine Reynoldszahl, noch laminare Strömung). Das bedeutet die Bildkante verwischt entlang dieser Strömung zusehends. Bei zu hoher Reynoldszahl des Problems kann eine Jetströmung auch zu Strukturen mit Wirbeln[24] führen, in denen sich die Bildkante ebenfalls auflöst.
Mathematisch formuliert wird I durch $\tilde{I} := 1 - I$ ersetzt. Daraus folgt $\tilde{v} = \nabla^{\perp}\tilde{I} = -\nabla^{\perp}I = -\vec{v}$ für die Daten auf $\partial\Omega$ und es ergibt sich durch das Ersetzen von $\vec{v}$ durch $\tilde{v}$ im Gleichungssystem (1.30) das folgende neue Gleichungssystem:

$$\begin{cases} \tilde{v} \cdot \nabla\tilde{v} - \nu\Delta\tilde{v} - \nabla\tilde{p} = 0 \\ \operatorname{div}\tilde{v} = 0 \\ \tilde{v}_{|\partial\Omega} = -\nabla^{\perp}I = -g \end{cases} \tag{1.31}$$

Dies liefert natürlich im Allgemeinen $\tilde{v} \neq \vec{v}$ in Inneren des Rechengebietes Ω und somit existiert für die zugehörige Stromfunktion kein $c \in \mathbb{R}$, so dass $\tilde{I} = 1 - I + c$ gilt[25].
Um eine Bildkante korrekt durch eine Fehlstelle fortzusetzen, muss eine Strömung die Fehlstelle durchqueren und dann von der entsprechenden Fortsetzung der Kante angesaugt zu werden. Wenn mehrere fortzusetzende Kanten nahe beieinander liegen, kann es dabei zu Zuordnungsproblemen oder zu Problemen wegen gegenläufiger Strömungen kommen. Diese wurden bei meinen Tests insbesondere bei der Rekonstruktion von Haarstrukturen zum Problem. Teilweise wirken die Rekonstruktionen von Haaren wie abgeschnitten, da ein Teil der Bildkanten nicht zur gegenüberliegenden Seite der Fehlstelle fortgesetzt wurden, sondern zu nahe liegenden inversen Kanten "umgebogen" wurden. Als Beispiel dafür müssen nur die Haare des rekonstruierten "Lena"-Testbildes aus Abschnitt 1.4.4 genauer betrachtet werden.

1.4.2 Implementierung

Das FEM-Paket FeatFlow enthält verschiedene Programme zur Berechnung von Strömungen, darunter auch "cc2d_movbc", welches stationäre (und instatinäre) inkompressible Navier-Stokes-Probleme (1.30) mit fiktiven Rändern berechnen kann, "pp2d", welches Navier-Stokes-Probleme rein instationär rechnet und eine Variante "pp2d_movbc" von "pp2d" mit fiktiven Rändern, welche von mir dahingehend modifiziert wurde, dass die fiktiven Ränder über eine eingelesene Bildmaske definiert werden. Dabei kommen jeweils FEM-Viereckselemente mit rotiert bilinearem Ansatzraum[26] für die Komponenten der Geschwindigkeit und

[24] Als kleines Beispiel für eine Jetströmung, welche Verwirbelungen induziert ist das folgende Beispiel aus "The Virtual Album of Fluid Flow" geeignet: `http://www.featflow.de/album/jetstream.html`.

[25] Es musste hier eine frei wählbare Konstante eingefügt werden, da die Stromfunktionen nur bis auf eine Konstante definiert sind und für die Rekonstruktionen beider Stromfunktionen aus den Strömungen die verwendeten Konstanten unterschiedlich sein können.

[26] $\tilde{Q}_1 = \operatorname{span}\{1, x, y, x^2 - y^2\}$, nichtkonform, Freiheitsgrade in den Seitenmittelpunkten. (Rannacher-Turek-Element)

stückweise konstantem Ansatzraum[27] für den Druck zur Anwendung. Die Freiheitsgrade der errechneten Stromfunktion befinden sich dabei in den Ecken des Viereckgitters. Das Programm "cc2d_movbc" wurde nicht gewählt, da der verwendete nichtlineare Löser teilweise Konvergenzprobleme bei der direkten Lösung der hier vorkommenden stationären Probleme aufwies und der instationäre Löser langsamer als der von "pp2d_movbc" war.

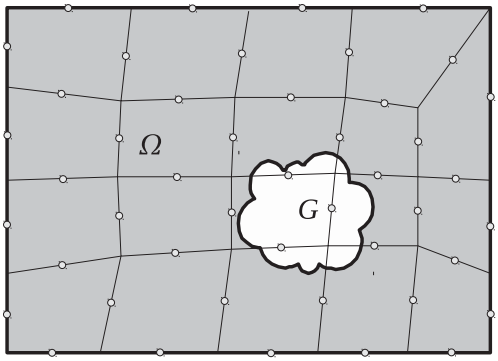


Abbildung 1.4.3: Gebiet Ω mit einem Loch G und Viereckgitter (Seitenmittelpunkte markiert).

Für Rechnungen, mit den rein analytisch definierte Beispielen aus den Abbildungen 1.4.5 und 1.4.6, kann der Gradient und somit die Strömungsstärke auch als vorher definierte Funktion bereitgestellt werden. Dies wurde auch benutzt, um die Ergebnisse der Abbildung 1.4.7 zu erhalten. Für die Anwendung auf Bilddaten muss ein anderer Weg beschritten werden.
Wie schon zuvor erwähnt, wurde das Programm "pp2d" von mir so modifiziert, dass neben dem Rechengitter auch Bilddaten und eine Bildmaske einließt. Aus der Bildmaske wird das eigentliche Rechengebiet bestimmt, welches durch einen fiktiven Rand umschlossen wird. Die Bilddaten werden als bilinear interpolierte Textur auf die Menge $[-1;1]^2$ abgebildet und die interpolierten Daten werden punktweise als Funktion bereitgestellt. Da die Bilddaten äquivalent zur Stromfunktion sein sollen, werden diese Daten im Folgenden in den Knoten des Gitters abgefragt. Die verwendeten Gitter waren so gewählt, dass in den zu rekonstruierenden Bereichen für jedes Pixel immer ein Knoten vorhanden ist, um dieses Pixel darstellen zu können. Rechnungen auf verfeinerten Gittern, zum Beispiel für Mehrgitterlöser, sind ebenfalls möglich. Für die abschließende Rekonstruktion der Bilddaten aus den errechneten Strömungsdaten sind aber lediglich die Ergebnisse auf dem Gitterlevel, der der Pixelstruktur entspricht, nötig, wobei das Programm die passende Stromfunktion für diesen Gitterlevel gleich mit ausrechnet. Diese Stromfunktionswerte können nun benutzt werden, um die Fehlstellen im Bild auszufüllen. Das Ergebnis kann dabei stark von ν abhängen (siehe Abbildungen 1.4.7).
Gegeben sei ein Gebiet Ω, welches den zu reparierenden Bildabschnitt enthält und mit einem Viereckgitter (siehe Abbildung 1.4.3) versehen wurde. Der Bildabschnitt wird auf den FEM-Lösungsraum der Stromfunktion projeziert, indem die

[27] P_0 in jedem Element, nichtkonform, Freiheitsgrad im Mittelpunkt.

bilinear interpolierten Bilddaten in den Knotenpositionen abgefragt werden, wobei die fehlenden Bildinformationen gleich Null gesetzt werden. Aus diesen Bildinformationen, die als eine FEM-diskretisierte Stromfunktion I aufgefasst werden können, lassen sich Näherungen für ∇I und $\vec{v}_0 := \nabla^\perp I$ errechnen, wobei die Informationen für alle Gitterzellen, welche Ecken an der Position fehlender Bildinformationen enthalten, als fehlerhaft anzunehmen sind. Aus diesem Grund müssen alle Geschwindigkeitsinformationen, welche zu solchen Zellen gehören, bei der Berechnung ignoriert werden. Alle Seitenmittelpunkte, die innerhalb der fehlenden Bildinformationen liegen oder zu Gitterzellen gehören, die fehlende Bildinformationen enthalten, werden nun als *Innen* und die restlichen Seitenmittelpunkte als *Außen* beziehungsweise *Rand* markiert. Bei der Generierung des Geschwindigkeitsfeldes $\vec{v}_0$ ist ebenfalls darauf zu achten, dass die Inkompressibilitätsbedingung erfüllt wird ($\operatorname{div} \vec{v}_0 = 0$) und die entsprechende Stromfunktion bis auf eine Konstante wieder I ist. Wäre die Inkompressibilitätsbedingung nicht erfüllt, so kann es zu Randbedingungen $\vec{g} = \vec{v}_0|_{\partial\Omega}$ führen, welche nicht konsistent mit dem zu lösenden Problem (1.30) sind.

Da "pp2d" nur rein instationäre Simulationen ausführt, kann mit diesem Programm die stationäre Lösung nicht direkt ausgerechnet werden. Um zu einer stationären Lösung zu kommen, wurden die Simulationen mit adaptiver Zeitschrittweitensteuerung so lange gerechnet, bis sich die Lösungen nicht mehr wesentlich änderten. Es wurde der implizite Euler als Zeitschrittverfahren gewählt, da diese Methode eine starke numerische Dämpfung aufweist, welche im stationären Limes egal ist, den Weg zu solch einer Lösung aber beschleunigt.[28]

Bei einigen der hier verwendeten Beispielen lagen die unbeschädigten Bildinformationen vor. Daher wurden dort, zur Vereinfachung der Implementierung, die Strömungswerte für das Originalbild ausgerechnet und danach die Strömung für innere Kantenmittelpunkte auf Null gesetzt. Macht man dies nicht, so müssen bei den angepassten Gittern aus Abschnitt 1.4.3 zwei zusätzliche Zellschichten statt einer hinzugefügt werden, damit die Näherungen für die tangentialen Geschwindigkeitsanteile korrekt berechnet werden. Dies kann durch das Wachstum des maskierten Bereiches um zwei Pixel, zum Beispiel mittels des Bildbearbeitungsprogrammes GIMP, erreicht werden, bevor daraus Gitter erzeugt werden. Der Grund für dieses Vorgehen liegt in der Art und Weise begründet, wie der tangentiale Anteil der Strömung berechnet wird:

1. Bilineare Interpolation der Bilddaten in die Ecken der Zellen.
   Anmerkung: *Bei Gittern, welche dieselbe Auflösung wie das zu bearbeitende Bild aufweisen, entspricht die diskretisierte Startlösung in den Ecken der Zellen exakt den Bilddaten.*

2. Berechnung des Gradienten der Bildinformation in der Zellmitte durch die Bilddaten in den Ecken.

[28] Weitere Details zu der internen Arbeitsweise von "pp2d" ist im Handbuch der FEATFLOW-Programmsuite zu finden: `http://www.featflow.de/media/download/featflow.ps.gz`.

3. Interpolation der Gradientendaten in den Zellmitten in die Mitten der Zellkanten. Dabei werden die Daten jeweils benachbarter Zellmitten gewichtet gemittelt.

4. Ermittlung des Normalanteils dieses Gradienten. Dies ist der tangentiale Anteil der gesuchten Strömung.

Die fehlerhaft Information in einer Ecke wird durch den zweiten Vorgang in die Zellmitten aller Zellen transportiert, welche diese Ecke gemeinsam haben und durch den dritten Vorgang in alle Kantenmitten dieser Zellen weitergetragen. Falls dieser dritte Schritt so modifiziert wird, dass in Fällen, bei der die Daten einer Zellmitte korrumpiert wurde, diese Daten ignoriert beziehungsweise aus den anderen Daten geschätzt werden, so könnte die Interpolation der Gradienten dort adaptiv durch eine Extrapolation ersetzt werden. Mit dieser Modifikation könnte man versuchen auch mit einer anstatt zwei Zellschichten auszukommen und einen unnötigen Informationsverlust zu vermeiden. Bei einem Anwendungsbeispiel von Bertalmío und Sapiro (siehe Abschnitt 1.4.4) waren die Ergebnisse für Rechnungen ohne tangentialen Anteil oder mit um zwei Bildpunkten vergrößerten Bildmasken besser als die Ergebnisse für die einseitig extrapolierten Strömungen, was meiner Meinung nach dafür spricht, dass die Fehlstellen über die maskierten Bereiche hinausgeragt haben beziehungsweise die Masken etwas zu knapp bemessen waren. Bei einseitiger Extrapolation und um einem Bildpunkt vergrößerte Maske wird dieser Teil heller als die umgebenden Hautpartien rekonstruiert, bei den anderen Methoden nicht. Derselbe Effekt tritt etwas schwächer auch bei der Rekonstruktion durch Bertalmío und Sapiro auf. Man beachte dazu die untere Augenpartie des mittleren Kindes (siehe untere linke Teilabbildung 1.4.12).
Versuche von mir, Startlösungen beziehungsweise Randdaten für die Bildergänzung durch einfache Projektion auf den FEM-Raum zu erhalten, haben sich in der Praxis leider nicht als genau genug herausgestellt. Insbesondere war der Gesamtfluss über den Rand der einzelnen Gebiete nicht mehr Null, was der Inkompressibilitätsbedingung widerspricht. Die Folge war, dass das Navier-Stokes-Problem mit diesen Randdaten nicht mehr wohl gestellt beziehungsweise unlösbar war. Um dieses Problem zu beheben, musste ein anderer Ansatz gewählt werden.
Der neue Ansatz lag darin, die Berechnungsmethode, welche aus einer Strömung eine Stromfunktion errechnet und auch von FeatFlow verwendet wird, umzukehren. Bei dieser wird, ausgehend von einer Ecke, die Differenz zum Wert der nächsten Ecke entlang der verbindenden Kante durch Auswertung des normalen Flusses über diese Kante bestimmt. Wenn nun die Werte in den Ecken einer Kante bekannt sind, so geben die Differenzen dieser Werte die Stärke des normalen Flusses über die Elementkante an. Dies liefert zwar wohlgestellte Randdaten, aber die erhaltenen Flüsse haben keinerlei tangentialen Anteil bezüglich der Elementkanten und hängen, da sie immer exakt normal zu den Kanten am Rand stehen, zu sehr von der Lage dieser Kanten und somit vom verwendeten Rechengitters ab. Der Unterschied zwischen ursprünglicher Strömung und einer auf die normalen Anteile reduzierte Strömung wird in Abbildung 1.4.4 verdeutlicht.

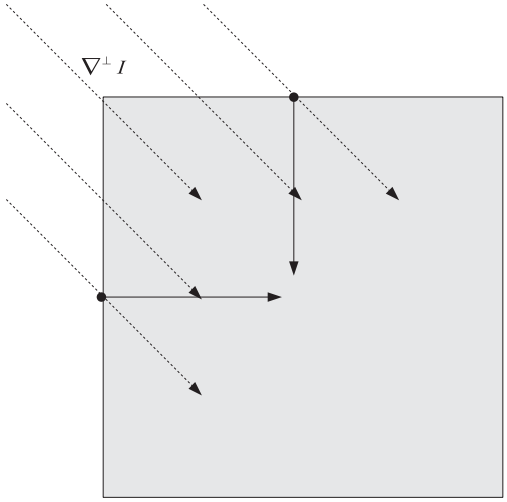


Abbildung 1.4.4: Normale Anteile einer Strömung, welche diagonal zu den Gitterkanten verläuft.

Dieses Problem konnte behoben werden, indem die beiden Ansätze zu einem neuen, hybriden, Ansatz verschmolzen wurden. Die Idee besteht darin, den normalen und tangentialen Anteil des Flusses getrennt voneinander zu bestimmen und anschließend zu vereinen. Der normale Anteil kann durch die vorher beschriebene Umkehrung der Berechnungsmethode zur Errechnung der Stromfunktion einer Strömung erhalten werden. Den fehlenden tangentialen Anteil kann man nun aus der vorher als zu ungenau bestimmten Projektion auf den FEM-Raum extrahieren. Wird dieser zusätzliche tangentiale Anteil auf die normalen Anteile der Strömung aufaddiert, werden die Werte der, zu dieser Strömung gehörenden, Stromfunktion nicht verändert! Die resultierende Strömung hat immer noch einen verschwindenden Gesamtfluss über den Rand des Rechengebietes und die Stromfunktion entspricht (bis auf eine Konstante) den originalen Bilddaten. Der beigemischte tangentiale Anteil verhindert, dass alle Strömungsdaten strikt am jeweils verwendeten Gitter ausgerichtet werden. Dies würde bei Bildkanten, welche diagonal zu den Gitterkanten liegen, genau dieselben Randdaten liefern als bei Bildkanten, welche dieselbe Ausrichtung wie die Gitterkanten besitzen, was natürlich zu einer fehlerhaften Rekonstruktion dieser Kanten führen würde. Zudem würde es Probleme mit Zellen geben, die zwei nebeneinander liegende Kanten am Rand aufweisen. Falls dort eine Strömung vorliegt, welche diagonal zu diesen Kanten liegt, so würden sich Einzelströungen ergeben, die im 90°-Winkel aufeinanderstoßen oder voneinander wegstreben (siehe Abbildung 1.4.4 zur Verdeutlichung).
Die Werte von $\vec{v}$ auf dem inneren Rand liefern Dirichlet-Randwerte für ein Navier-Stokes-Problem, welches für ausreichend großes ν im Laufe der Zeit stationär wird, beziehungsweise für das die stationären inkompresiblen Navier-Stoke-Gleichungen (1.30) eine eindeutige Lösung besitzen.
Das zur Lösung verwendete Programm erfordert, dass das mit *Innen* bezeichnete Gebiet zusammenhängend ist. Daher müssen die einzelnen Komponenten nicht zusammenhängender Fehlstellen einzeln gerechnet werden. Es ist dabei zu beach-

ten, dass die einzelnen Komponenten durch ausreichend Gitterzellen voneinander getrennt sind, um wohldefinierte Randdaten zu erhalten. Falls die Erreichung eines ausreichenden Abstandes nicht möglich sein sollte, müssen die Fehlstellen an den Stellen mit zu geringem Abstand durch "Brücken" miteinander verbunden werden. Dies bewirkt eine Verschmelzung der so verbundenen Fehlstellen zu einer einzigen neuen Fehlstelle, mit der das Programm arbeiten kann.
Ein Beispiel für eine Bildrekonstruktion ist in Abbildung 1.4.5 zu sehen, bei dem das Bild ein radialer Gradient ist[29], der linear mit der Distanz zum Mittelpunkt des Gebietes abnimmt. Die Fehlstelle ist ein asymmetrisch platziertes Kreuz, welches alle vier Seiten des Gebietes berührt. Die Darstellung des errechneten Druckes (rechte Abbildung) zeigt die Konturen des Kreuzes. Wie anhand der linken Abbildung zu sehen ist, wird die Krümmung des Bildgradienten ohne sichbare Abweichungen wiederhergestellt. Andere Methoden zur Bildrekonstruktion, welche auf der Minimierung der TV-Norm[30] beruhen, haben das Problem, dass sie die Bildwerte an den Rändern der Fehlstelle mit geraden Verbindungen zu überbrücken versuchen, womit gekrümmte Strukturen nicht rekonstruierbar sind und bei diesem Beispiel die Abweichung sofort sichtbar gewesen wäre.

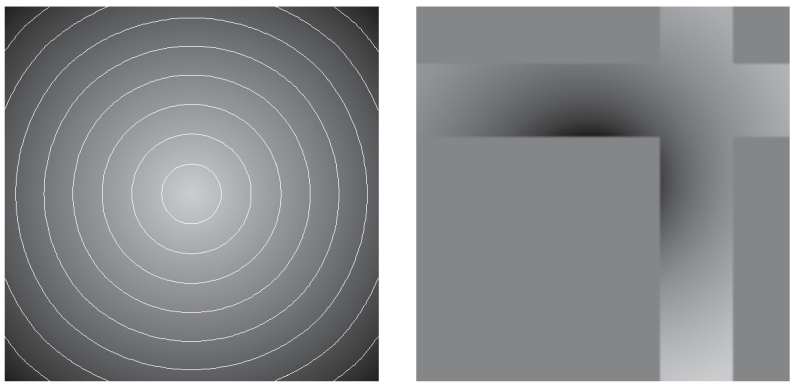


Abbildung 1.4.5: Ergänztes Bild (Stromfunktion mit Isolinien) und Druckwerte für eine asymmetrisch platzierte kreuzförmige Fehlstelle.

Die Abhängigkeit der rekonstruierten Darstellung von der Diffusion ν zeigt sich vor allen in uneindeutigen Situationen und großflächigen Fehlstellen. Ein Beispiel dafür ist in Abbildung 1.4.6 zu sehen. Es handelt sich dabei um zwei gleich große nicht berührende schwarze Kreise mit einer kreisförmigen Fehlstelle, welche beide Kreise symmetrisch schneidet. Die Kreise sind mit ringförmigen Übergangsbereichen versehen, welche die Bildgradienten limitieren sollen. Die Rekonstruktion kann nun entweder die Kreise durch eine nach innen gebogene Überbrückung zu einem einzigen hantelförmigen Objekt verbinden (Abbildungen 1.4.7(a)–(c)) oder aber die zwei Kreise wiederherstellen (Abbildungen 1.4.7(d)–(f)) werden. Im letz-

[29] Darstellung mit Isolinien, die kein Teil des Bildes selbst sind.
[30] Norm der "Totalen Variation".

teren Fall werden durch den Wiederherstellungsprozess interne Wirbel induziert, welche zu einer Erweiterung des ursprünglichen Wertebereiches der Stromfunktion führen. Dies bedeutet genauer, dass die Stromfunktion im Inneren der Fehlstelle nicht auf den Wertebereich der Stromfunktion auf dem Rand der Fehlstelle beschränkt bleibt. Es gibt für diesen Ansatz *kein* Maximumsprinzip. Die internen Wirbel entsprechen "Flecken" im rekonstruierten Bild. Die Stromfunktion kann nun entweder fest durch die Werte auf dem Rand der Fehlstelle limitiert werden, wobei die Bilddaten an Glattheit verlieren, oder aber ν wird solange erhöht, bis alle internen Wirbel verschwunden sind und die Strömung laminar wird, was aber die Kanten des Bildes durch Verwischen zerstören würde.

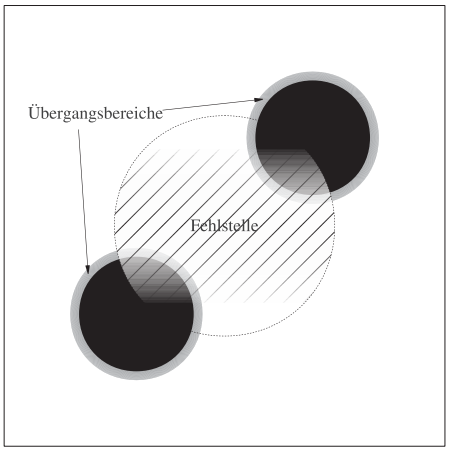
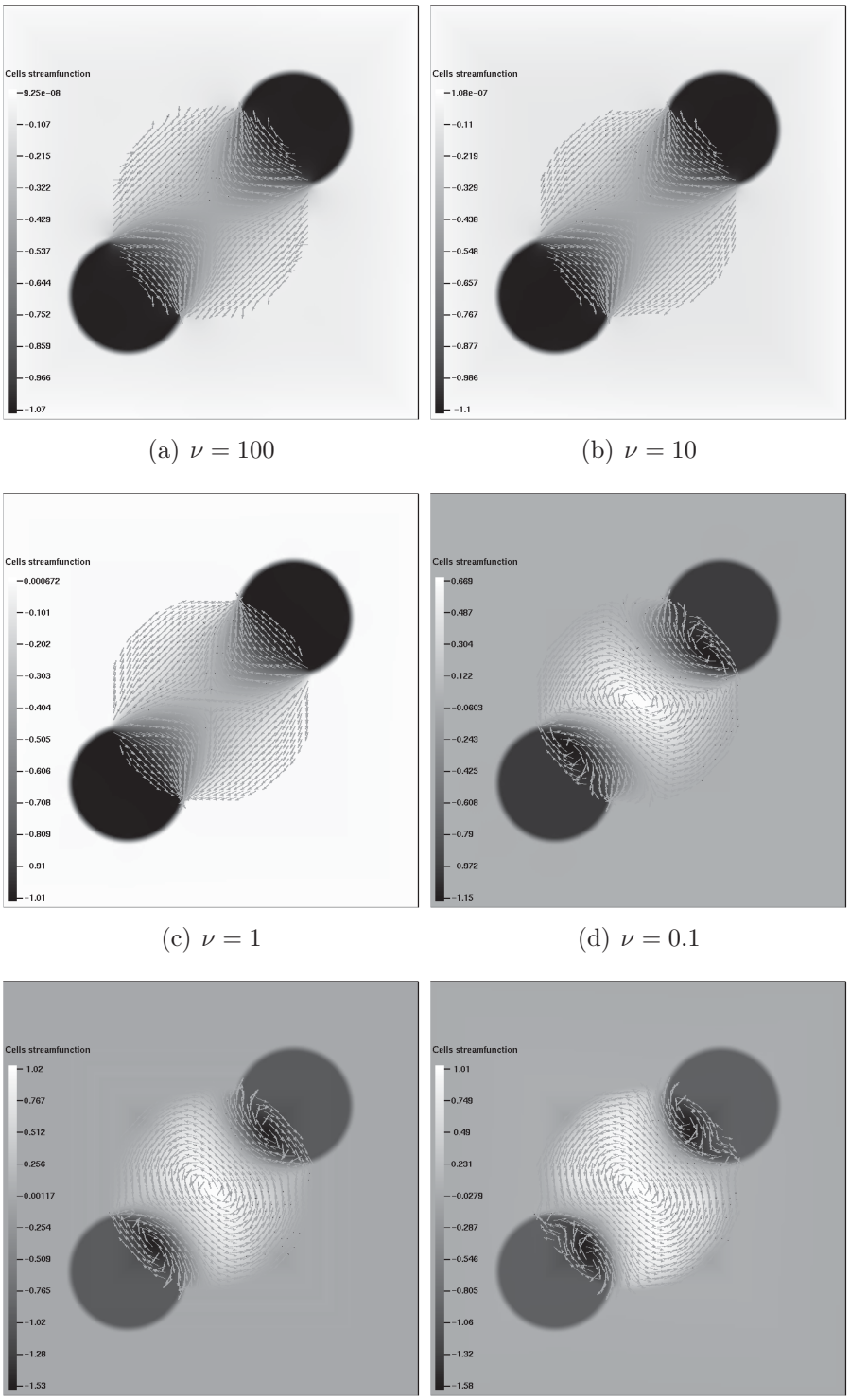


Abbildung 1.4.6: Skizze der Problemstellung.

Ein zu großer Wert von ν ist aber auch nicht wünschenswert, da ansonsten die rekonstruierten Bildinformationen zu stark diffundiert werden, insbesondere harte Kanten zerfließen sichtbar beim Eintritt in die Fehlstelle. ν sollte also möglichst gerade so groß gewählt werden, dass eventuelle interne Wirbel verschwinden oder zumindest deren Anzahl minimiert wird. Dies könnte zum Beispiel durch die Erkennung geschlossener Isolinien in der errechneten Stromfunktion oder durch die Suche nach lokalen Minima und Maxima der Stromfunktion geschehen. Beides zeigt das Vorhandensein von internen Wirbeln an.
Der Ansatz hinter der hier vorgestellten Methode beruht auf der "fließenden" Verbindung von Randdetails, was relativ glatte "Flicken" für die Löcher in der Bildinformation ergibt. Die Methode wird somit Probleme beim Auffüllen von gemusterten Bildregionen haben. Dies ist zwar bei kleinen Löchern, dünnen Kratzern oder Bildregionen ohne viele Details nicht allzu auffällig, das Auge eines Betrachters erwartet aber in gemusterten Regionen eine regelmäßige Fortführung des Musters. Abweichungen von einem erkannten Muster treten optisch stark hervor, was diese Methode für das Reparieren von größeren Löchern in gemusterten Bildregionen ungeeignet werden lässt.
Dieses Problem soll an einem einfachen Schachbrettmuster vorgeführt werden. Die Abbildung 1.4.8 zeigt ein 3x3-Schachbrettmuster mit einem quadratischen Loch als Testbeispiel. Das Loch deckt dabei das mittlere schwarze Quadrat komplett



(a) $\nu = 100$ (b) $\nu = 10$

(c) $\nu = 1$ (d) $\nu = 0.1$

(e) $\nu = 0.05$ (f) $\nu = 0.01$

Abbildung 1.4.7: Ergebnisse einer Bildwiederherstellung für unterschiedliche Werte von ν. Die entsprechenden Strömungen wurden mit Vektorpfeilen dargestellt. Man beachte die Erweiterung des Wertebereiches bei (d)-(f).

ab, was in einer Entfernung desselben enden wird. Auch die vier Ecken der umliegenden Quadrate werden ebenfalls von diesem Loch abgedeckt. Es ist zu erwarten, dass die Ecken in abgerundeter Form rekonstruiert werden.

Für die Bildrekonstruktion wurde ein reguläres äquidistantes Viereckgitter mit 65536 Gitterzellen[31] verwendet. Die Ergebnisse für drei verschiedene Werte von ν sind in den Abbildungen 1.4.9(a)–(c) zu sehen. Wie erwartet, verschwindet in allen drei Fällen das mittlere Quadrat des Schachbrettmusters aus der Rekonstruktion. Dies führt zu einer sehr auffälligen Unterbrechung des Musters.

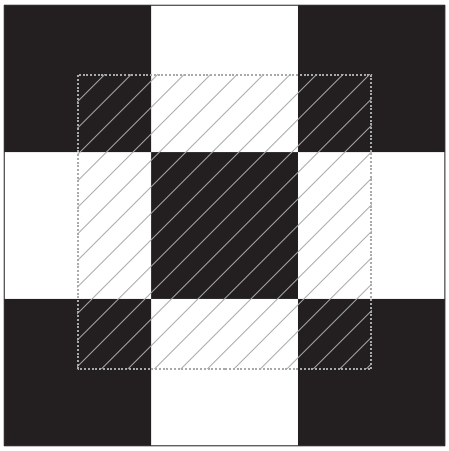


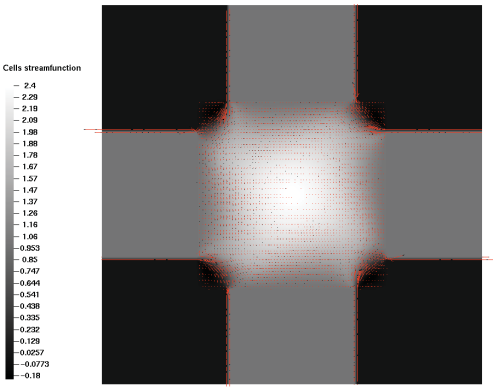
Abbildung 1.4.8: 3x3 Schachbrettmuster mit angedeutetem Loch (gestrichelt).

Auch hier ist die Wahl eines ausreichend großen ν wichtig. Für zu kleine Werte (siehe Abbildung 1.4.9(a)) wird wieder ein interner Wirbel erzeugt, der ein Überschreiten des ursprünglichen Wertebereiches der Stromfunktion bewirkt. Dies ist in der Abbildung als heller Fleck zu sehen. Durch den veränderten Wertebereich werden zudem ursprünglich weiße Bereiche nun grau dargestellt. Auch die Abrundung der rekonstruierten Ecken ist zu erkennen. Dabei werden die Kanten, wie in den Abbildungen 1.4.9 zu sehen ist, stark verwischt.

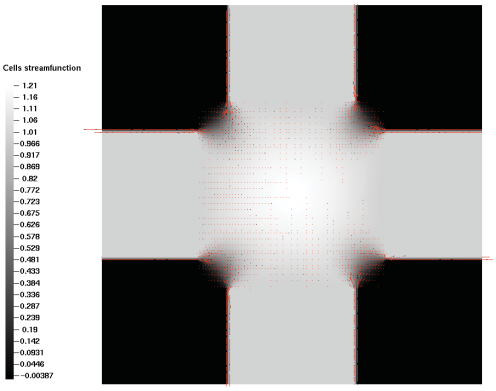
Das Verfahren kann in einigen Fällen auch bei Mustern angewandt werden, sofern die Fehlstellen wesentlich kleiner als die lokalen Details des Musters sind. Im Fall des Schachbrettmusters wäre dies der Fall, wenn die Fehlstelle wesentlich kleiner als eines der Quadrate ist. Die Rekonstruktion ist auch in diesem Falle nur dann perfekt, wenn die Fehlstelle innerhalb eines der Quadrate liegt. Ansonsten gibt es Abweichungen vom Original, aber es verschwindet kein Quadrat vollständig.

Eine ebenfalls PDE-basierte Methode, welche sich auch für Farbbilder eignet, wurde von David Tschumperlé und Rachid Deriche entwickelt. Dabei werden die Bilddaten auf den Rändern der Fehlstellen auf strukturerhaltende Art ins Innere der Fehlstelle diffundiert. Dabei werden auch Strukturen über die einzelnen Farbkanäle hinweg erhalten. Das Ergebnis liegt vor, wenn die Lösung in den stationären

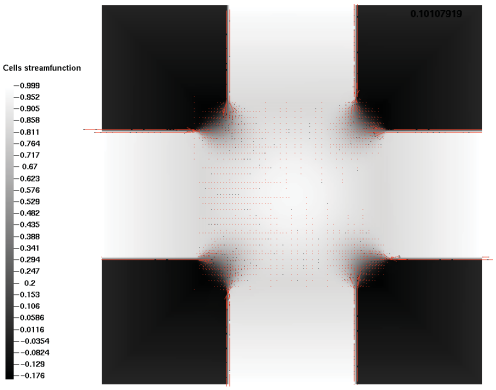
[31] 256x256-Gitter.



(a) $\nu = 0.1$



(b) $\nu = 1$



(c) $\nu = 10$

Abbildung 1.4.9: Bildrekonstruktionen für verschieden Werte von ν. Die Rechnung erfolgte auf einem regulären Viereckgitter mit 65536 äquidistanten Gitterzellen. Die entsprechenden Strömungen wurden mit Vektorpfeilen dargestellt.

Zustand übergeht. Die verwendete PDE mit Startlösung $\vec{I} = (I_1, \ldots, I_n)^\top$ ist

$$\begin{aligned} \frac{\partial u_i}{\partial t} &= \operatorname{trace}(\mathbf{T}\,\mathcal{H}(u_i)) && (i = 1, \ldots, n) \\ u_i(0, \cdot) &= I_i(\cdot) && (i = 1, \ldots, n), \end{aligned} \tag{1.32}$$

wobei $\mathbf{T}$ ein punktweise definierter Tensor ist, welcher aus den Spektralelementen des geglätteten Strukturtensors errechnet wird und $\vec{u} = (u_1, \ldots, u_n)^\top$ liefert rekonstruierte Bildwerte aus dem $\mathbb{R}^n$ (hier $n = 3$). Details und Beispiele dazu sind in [TD05a, Seite 7-10] zu finden.
Marcello Bertalmío und Guillermo Sapiro[32] haben in Zusammenarbeit mit Vicent Caselles und Colorma Ballester[33] ebenfalls eine Methode präsentiert (siehe [BSCB11]), bei der Informationen von den Rändern der Fehlstellen mittels anisotroper Diffusion entlang der Isophoten $\nabla^\perp I$ ins Innere gelangen. Dabei wird im stationären Zustand geometrisch die Gleichung $\nabla(\text{"Maß der Glätte"}) \cdot \nabla^\perp I = 0$ gelöst. Dies bedeutet, dass ein vorgegebenes Maß der Glätte bezüglich der Bildfunktion konstant entlang der Isophoten sein soll. Etwas ältere Arbeiten von Bertalmío, Bertozzi und Sapiro zum Thema Bildrekonstruktion sind [BBS01, BBS02].

1.4.3 Beschleunigung durch angepasste Gitter

Bei der bisher verwendeten Berechnungsmethode wurde nur auf einem relativ kleinen Teil des Rechengitters wirklich gerechnet. Informationen, welche für die Rechnung völlig überflüssig sind, müssen im Hauptspeicher des Rechners gespeichert und beim Lösen ausgelesen werden. Dies verschwendet Speicherplatz und Zeit beim Speichertransfer, während die iterativen Löser angewendet werden. Dies führt wiederum zu einer drastischen Verlangsamung der Berechnungen.
Um dieses Problem zu beheben, wurde von mir ein spezielles Python-Programm geschrieben, welches aus einer Beschreibung der Fehlstellen mittels einer Bildmaske für jeden zusammenhängenden Bereich ein Rechengitter komplett mit passender Randparametrisierung erzeugt. Das vollständige Programm wird in Anhang B aufgelistet.
Das Programm nimmt quadratische monochrome[34] PNG-Bilder im Format $N \times N$ an und bildet diese auf die Knoten eines virtuellen Tensorproduktgitters mit $(N-1) \times (N-1)$ Zellen ab, welches das Gebiet $[-1; 1]^2$ ausfüllt. Weiß bedeutet dabei "Fehlstelle" und Schwarz "keine Fehlstelle". Da eine Zelle im Rechengebiet sein muss, sobald auch nur eine Ecke der Zelle innerhalb der Fehlstelle liegt, wird nun eine Karte erstellt, welche solche Zellen als "zu einem Rechengebiete gehörend" markiert. Alle anderen Zellen sind in dieser Karte als "nicht zugehörig markiert". Im nächsten Verarbeitungsschritt wird diese Karte systematisch nach markierten Zellen abgesucht. Sobald solch eine Zelle gefunden wurde, werden alle Zellen, die über eine Kantennachbarschaft zur selben Zusammenhangskomponente gehören, mittels eines Breitensuchalgorithmus festgestellt, in eine eigene Karte bezüglich dieser Zusammenhangskomponente eingetragen und die Markierungen dieser Zellen aus der ursprünglichen Karte gelöscht. Dies wird nun solange fortgeführt, solange noch markierte Zellen gefunden werden können oder bis die Karte ganz durchsucht wurde. Das Ergebnis dieses Verarbeitungsschrittes ist eine Liste mit einzelnen Karten.

[32] Electrical and Computer Engineering, University of Minnesota.
[33] Escola Superior Politecnica, Universitat Pompeu Fabra.
[34] Nur 1 Bit pro Pixel. Jedes Pixel ist entweder schwarz oder weiß.

Nun wird für jede dieser Karten eine lokale Zellen- und Knotennummerierung erstellt und eine Liste mit Koordinaten für die einzelnen Knoten abgespeichert. Dann wird sowohl zeilen- als auch spaltenweise achsenparallele Strahlen mittels eines Algorithmus durch die Kanten von Zellen geschickt. Damit kann man bestimmen, welche Kanten an den Übergängen von "Außen" nach "Innen" und umgekehrt liegen und diese mit der korrekten Orientierung in eine Liste einfügt. Aus diesen Kantenlisten können nun sehr einfach die Anzahl der Ränder und die Ränder selbst bestimmt werden. Die Ränder werden dabei nach folgendem Algorithmus bestimmt:

1. Setze die Menge R=$\emptyset$.
2. Füge für jede Übergangskante E das Knotentupel (e_1, e_2) zu R hinzu. Die Reihenfolge bezeichnet dabei die Orientierung der Kante.
3. Solange Tupel $K = (k_1, \ldots, k_n)$ und $L = (l_1, \ldots, l_m)$ mit $k_n = l_1$ gefunden werden können:
   (a) Entferne K und L aus S: $S := S \setminus \{K, L\}$
   (b) Verschmelze K und S zu M und füge dies in S ein: $S := S \cup \{(k_1, \ldots, k_n, l_2, \ldots, l_m)\}$
4. (Fast) Fertig!

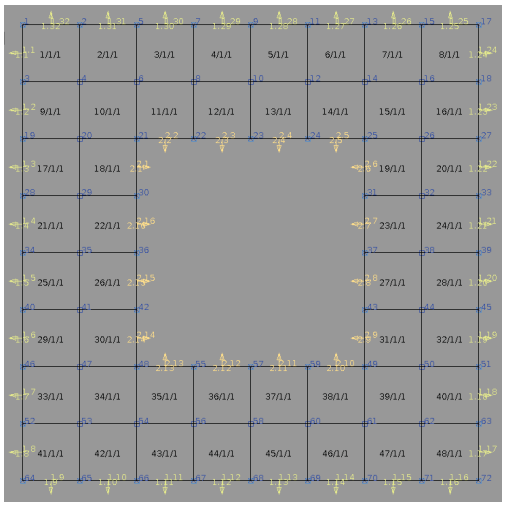
Die Tupel in R bezeichnen nun orientierte Ränder, die aber nachbearbeitet werden müssen. Es können "Schleifen" vorliegen, dass bedeutet mehrfaches Vorkommen eines Knotens innerhalb eines Randes. Ebenso kann es sein, dass es Knoten gibt, die auf mehreren Rändern liegen. Durch eine Nachbearbeitung der Ränder lassen sich Ränder mit gemeinsamen Knoten zu einem Rand mit Schleifen verschmelzen, welche danach durch Einfügen von zusätzlichen Knoten aufgetrennt werden. Dabei muss auch die Knoteninformation der Zellen angepasst werden. Die Verschmelzung der Ränder $K = (k_1, \ldots, k_{r-1}, q, k_{r+1}, \ldots, k_n)$ und $L = (l_1, \ldots, l_{s-1}, q, l_{s+1}, \ldots, l_m)$ mit gemeinsamen Knoten q sei dabei:

$$M = (k_1, \ldots, k_{r-1}, q, l_{s+1}, \ldots, l_m, l_1, \ldots, l_{s-1}, q, k_{r+1}, \ldots, k_n).$$

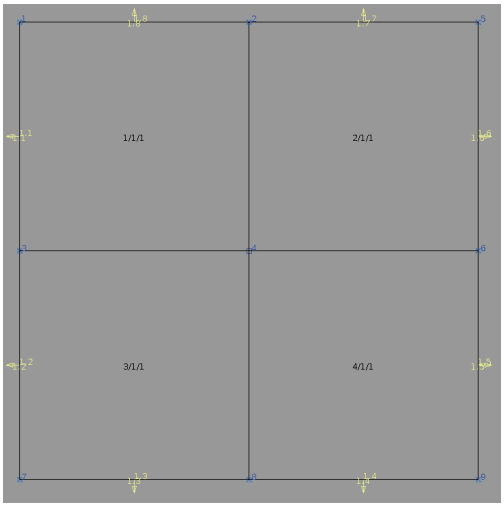
Schleifen lassen sich sehr einfach als doppeltes Vorkommen ein und derselben Knotennummer q erkennen. Aufgrund der Struktur des ursprünglichen Tensorproduktgitters, kann dabei eine Knotennummer höchstens zweimal in einem Rand vorkommen. Dies liegt daran, dass die Schleifen und Berührungen von zwei verschiedenen Rändern hier nur bei Zellen vorkommen, die über einen gemeinsamen Knoten diagonal miteinander verbunden sind. Das Auftrennen erfolgt durch das Erzeugen einer neuen, noch nicht verwendeten Knotennummer $\tilde{q}$, welche einen Knoten mit denselben Koordinaten wie q bezeichnet, und anschließendes Ersetzen des zweiten Vorkommens von q durch dieses $\tilde{q}$. Dabei muss auch die Knoteninformation der Zellen angepasst werden, so dass eine der beiden Zellen, die als Knoten q enthält, diesen durch $\tilde{q}$ ersetzt.

Die Knotenkoordinaten, Randlisten und Zellinformationen werden nun benutzt, um Gitter im PRM/TRI-Format herauszuschreiben, welche sich zur Rechnung mit FeatFlow-Programmen eignen.

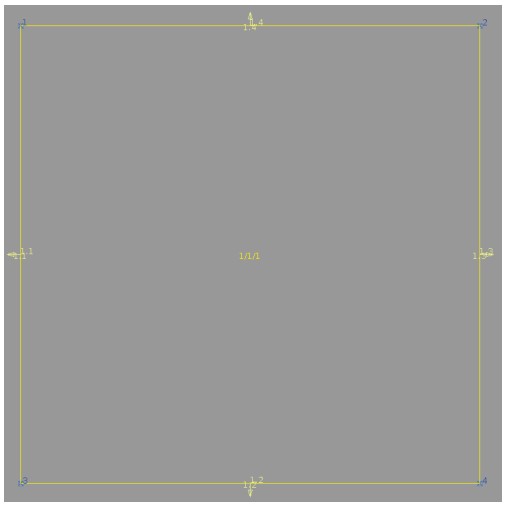
(a) Bildmaske



(b) Gitter für den weißen Ring



(c) Gitter für den mittleren weißen Punkt



(d) Gitter für den Punkt unten Rechts

Abbildung 1.4.10: Bildmaske (Format 11x11) für ein Tensorproduktgitter mit 10x10 Zellen und die daraus erzeugten Gitter. Jedes weiße Quadrat entspricht einem Knoten mit beschädigter Information. *Achtung:* Die erzeugten Gitter wurden auf volle Größe skaliert!

Ein kleines, sehr überschaubares Beispiel ist in Abbildung 1.4.10 zu finden. Oben links ist eine kleine Bildmaske zu sehen, welche angibt, welche Knoten "Innen" sein müssen. Die entsprechende Zellmaske hat nun drei Zusammenhangskomponenten bezüglich der Kantennachbarschaft. Die erste Zusammenhangskomponente entspricht einem Ring (Abbildung oben rechts), die zweite einem 2×2 Gitter (Abbildung unten links) und einem Gitter bestehend aus einer einzigen Zelle (Abbildung unten rechts).

Diese Umwandlung einer Maske, welche die Fehlstellen beschreibt, in eine Folge von Gittern ist somit durch dieses Python-Programm vollständig automatisiert und benötigt keinerlei Benutzereingriff!
Zwar könnte man etwas Ähnliches erreichen, indem man das Bild in rechteckige Teilbereiche zerlegt, welche jeweils eine Fehlstelle umschließen ("bounding box"-Ansatz) und dort mit Tensorproduktgittern rechnen, aber insbesondere bei langen dünnen und schräg liegenden Kratzern, würden damit noch viel zu viele unnötige Zellen in den Rechengittern vorhanden sein und kaum Beschleunigung der Rechnung auftreten. Dies wird sofort deutlich, wenn man sich überlegt, dass die Zellenzahl für einen dünnen diagonal liegenden Kratzer bei dieser Methode quadratisch mit der Länge des Kratzers wächst, aber bei der Verwendung eines angepassten Gitters lediglich linear.
Durch die Implementierung eines integrierten Systems würde hier noch weiteres Potential zur Beschleunigung und Erhöhung des Benutzerkomforts des Verfahrens vorliegen. Solch ein integriertes System würde bei der Erstellung der Masken helfen und für jede Maske automatisch die Anzahl der Teilrechnungen, die Gitter und die passenden Konfigurationsdateien für die numerischen Simulationen erzeugen, diese parallel und lastenangepasst auf verfügbare Rechenknoten eines Rechnerverbundes/Großrechners verteilen und die resultierenden Ergebnisse einsammeln. Danach würde solch ein System entweder selbst das korrigierte Bild aus diesen Ergebnissen zusammenfügen oder diese Aufgabe an ein Visualisierungsprogramm, wie zum Beispiel ParaView[35] weiterleiten.

1.4.4 Anwendungsbeispiele

Bisher wurde die hier vorgestellte Methode zur Bildergänzung nur auf künstlich erzeugte Beispiele angewendet. Es folgen nun Anwendungen auf natürliche Bilder, um die Tauglichkeit in der Praxis zu untersuchen.
Die folgenden Beispiele wurden mit einer modifizierten Variante von "pp2d" gerechnet, welche schon im Abschnitt 1.4.2 erwähnt wurde. Diese Variante liest die Maskenbeschreibung der Fehlstellen ein und bestimmt daraus einen fiktiven Rand. Zudem werden die Ergebnisse für jede einzelne Fehlstelle getrennt auf angepassten Gittern gerechnet (siehe Abschnitt 1.4.3) und die Ergebnisse mittels einer geschickten Anwendung des Visualisierungsprogrammes ParaView wieder zusammengefügt. Genauer beschrieben erfolgte dies, indem das Originalbild als Textur auf ein quadratisches Gebiet mit den passenden Abmessungen gelegt wurde und die einzelnen Ergebnisse als jeweils eigene GMV-Dateien eingelesen und visualisiert wurden. Dabei lagen diese vor der Textur, so dass dort die Textur durch die Darstellung der Rechenergebnisse überdeckt wurde. Die Fläche, auf welche ParaView die Textur projezieren soll, sollte an allen Seiten um eine halbe Pixelbreite des ursprünglichen Bildes größer sein, damit alles wieder zusammenpasst!
Zwar wäre es präziser gewesen, wenn die Rechenergebnisse direkt auf die Pixel des zu restaurierenden Bildes übertragen worden wären. Die hier erreichte Genauig-

[35] Dies kann mittels einer eigens generierten PVSM-Datei geschehen.

keit sollte jedoch für eine qualitative Beurteilung der hier vorgestellten Methode ausreichen.

"Lenna"-Testbild

Als erstes Beispiel wird ein Klassiker der Bildverarbeitung herangezogen, das als sogenanntes "Lenna"-Testbild (mitunter auch als "Lena") bekannte Motiv.[36] Hier wird lediglich ein 174×174 Ausschnitt des Gesichtes für das Beispiel verwendet.

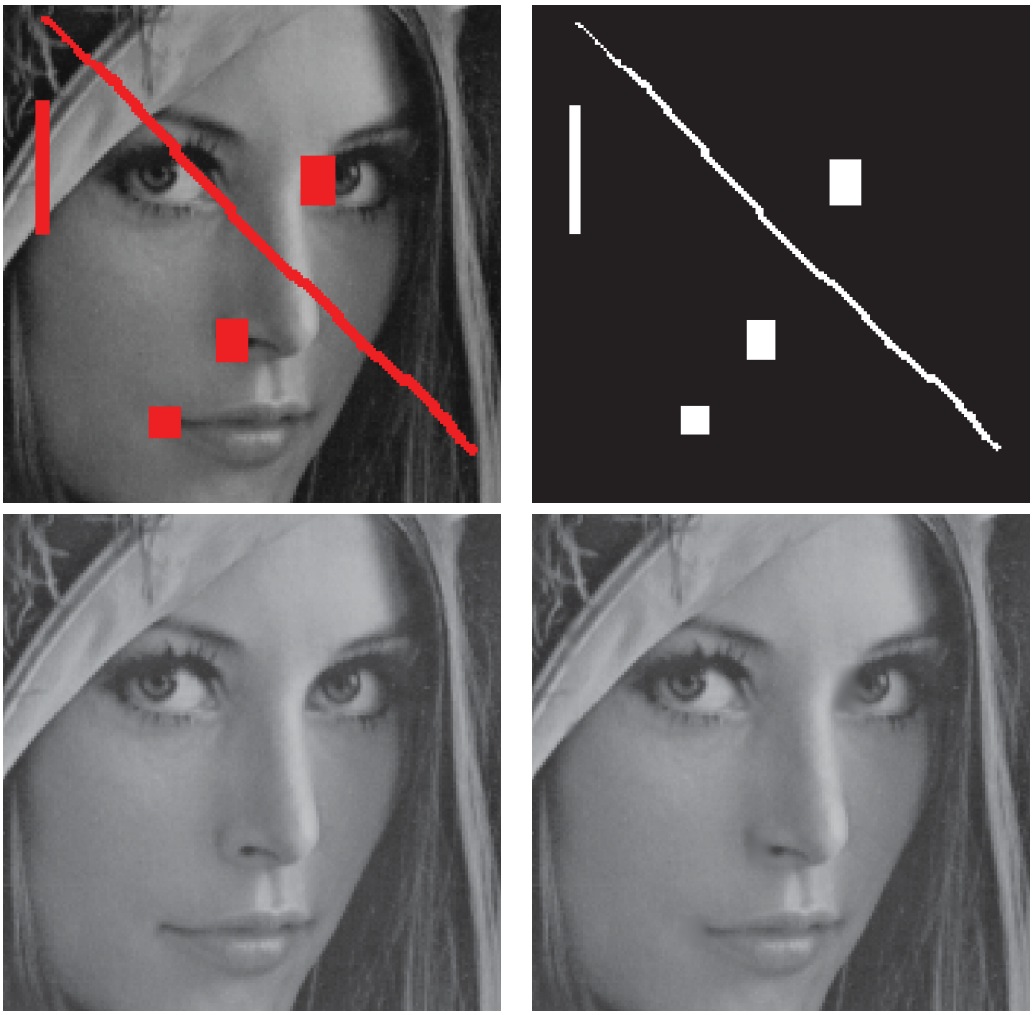


Abbildung 1.4.11: Von links oben nach rechts unten: Originalbild mit rot markiertem Fehlstellen, Fehlstellen als Bitmaske, Originalbild und rekonstruiertes Bild für $\nu = 0.02$.

Das Gesicht wurde mit zwei dünnen Kratzern und drei flächigen Schadstellen versehen (siehe Abbildung 1.4.11 links oben). Einer der Kratzer ging durch die Hutkrempe, durch eines der Augen, quer über die Nase, Wange und schließlich durch

[36] Beruht auf einem Foto aus der Zeitschrift Playboy. Weitere Informationen zu diesem Bild gibt es im Wikipedia-Artikel *https://secure.wikimedia.org/wikipedia/en/wiki/Lenna*.

die Haare. Diese Bereiche bereiteten jeweils unterschiedliche Probleme, welche sich die hier vorgestellte Methode stellen musste. Die flächigen Schadstellen lagen im Winkel eines der Augen, am Rand eines Nasenloches und in einem Mundwinkel. Dies sind alles Bereiche, welche gekrümmte Bilddetails enthalten.
Für diese fünf Bereiche wurden mit dem in Abschnitt 1.4.3 besprochenen Programm angepasste Gitter erstellt. Das angepasste Gitter für den großen langen diagonalen Kratzer hat dabei lediglich 744 Zellen und 1046 Knoten. Ein Gitter, welches das ganze Bild umfassen würde, hätte $173^2 = 29929$ Zellen und $174^2 = 30276$ Knoten. Die Rechnungen wurden aus technischen Gründen[37] auf einem einmal verfeinerten Gitter durchgeführt, aber auf dem Originalgitter herausgeschrieben.
Wie durch einen Vergleich von Abbildung 1.4.11 unten links und rechts ersichtlich ist, werden die meisten Bilddetails gut repariert, wobei die Kanten der Hutkrempe aber leicht an Schärfe verloren haben. Das Nasenloch, die Ecke des Auges und der Mundwinkel sehen sehr natürlich aus, wobei im restaurierten Bild der Mundwinkel etwas kürzer ist und das angedeutete Lächeln verschwunden ist. Probleme gibt es beim Haar, da dort die Bildgradienten ∇I sehr klein sind. Damit ist die Strömung dort auch sehr schwach und diffundiert zu stark. Dies erzeugt eine Unterbrechung in den Haarstrukturen, welche aber nur bei genauem Hinsehen sichtbar ist.

Drei Kinder

Das nächste Beispiel wurde häufig in diversen Veröffentlichungen von Marcelo Bertalmío und Guillermo Sapiro verwendet und kann auch als Beispiel (siehe [Ber]) auf der Internetseite von Bertalmío gefunden werden. Es besteht aus einem alten Foto, das drei kleine Mädchen zeigt und erhebliche Schädigungen aufweist, die sehr danach aussehen, als ob das Bild sowohl gefaltet, als auch zerknittert worden wäre. Um eine bessere Vergleichbarkeit mit den Resultaten anderer Arbeiten zu erhalten, ging ich von derselben Maskierung der Fehlstellen aus, welche auch Bertalmío und Sapiro verwendeten (siehe Abbildung 1.4.12). Meine Programmcodes verarbeiten momentan nur quadratische Bilder, weshalb das Bild und die Maske zur Rekonstruktion passend aufgefüllt und später wieder beschnitten wurden. Wie schon in Abschnitt 1.4.2 beschrieben, muss bei diesem Beispiel die verwendete Bildmaske vergrößert werden. Die vergrößerten Bildmasken sind in Abbildung 1.4.13 zu sehen. Die in Abschnitt 1.4.3 beschriebene Beschleunigung durch angepasste Gitter erzeugt bei einer um einen Pixel vergrößerte Bildmaske genau zehn und bei einer um zwei Pixel vergrößerte Bildmaske genau neun voneinander unabhängige Einzelgitter. Für jedes dieser Gitter muss eine getrennte Rechnung durchgeführt werden, deren Ergebnisse später mit dem Ausgangsbild zu einer Gesamtrekonstruktion des Bildes zusammengeführt werden. Die einzelnen Rechnungen sind dabei völlig unabhängig voneinander und können parallel ausgeführt werden.

[37] FeatFlow-Programme verwenden einen Mehrgitterlöser, der mindestens eine einmalige Verfeinerung des Gitters benötigt.

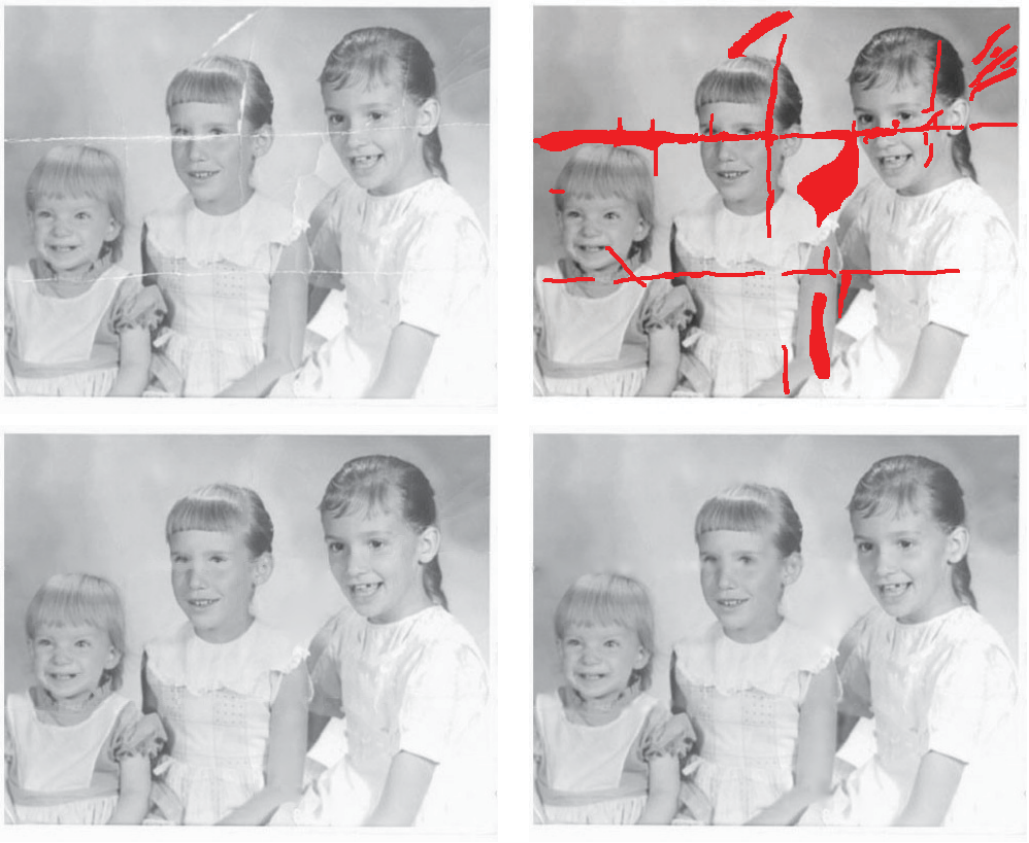


Abbildung 1.4.12: Von links Oben nach rechts Unten: Originales Testbild, Testbild mit rot markierten Fehlstellen, Rekonstruktion durch Bertalmío und Sapiro, eines meiner Ergebnisse für eine um zwei Pixel größere Maske, interpoliertem tangentialem Anteil und $\nu = 0.1$.

Die Abbildung 1.4.12 zeigt das beschädigte Originalbild, die verwendete Markierung und einen Vergleich eines Ergebnisses von Bertalmío mit dem, meiner Meinung nach besten Ergebnis, welches ich mit dem hier vorgestellten Verfahren erzeugen konnte. Ein Vergleich der rechten Wange des rechten Mädchens und des Überganges des linken Arms des mittleren Mädchens zum Rock des rechten Mädchens, zeigt zusätzliche Flecken in der Nähe von ausgeprägten Kanten. Diese treten dadurch auf, dass entlang der Kanten eine starke Strömung vorliegt, welche interne Wirbel induzieren kann. Diese Wirbel entsprechen Flecken im rekonstruierten Bild und können, wie schon zuvor erwähnt, auch Werte in der Stromfunktion erzeugen, welche nicht mehr im Wertebereich des ursprünglichen Bildes liegen. Will man diese Flecken vermeiden, so muss der Diffusionskoeffizient so hoch gewählt werden, dass die Strömung laminar wird. Dies hat aber den negativen Effekt, dass Bildkanten sehr stark geglättet werden. Das Ergebnis von Bertalío ist in diesem Bereich besser als die hier vorgestellte Methode. Vergleicht man aber das rechte Auge und den Nasenrücken des mittleren Mädchens bei beiden Verfahren, so liefert sie meiner eigenen Meinung nach bessere Ergebnisse.

An diesem Beispiel lassen sich auch verschiedene Methoden vergleichen, welche

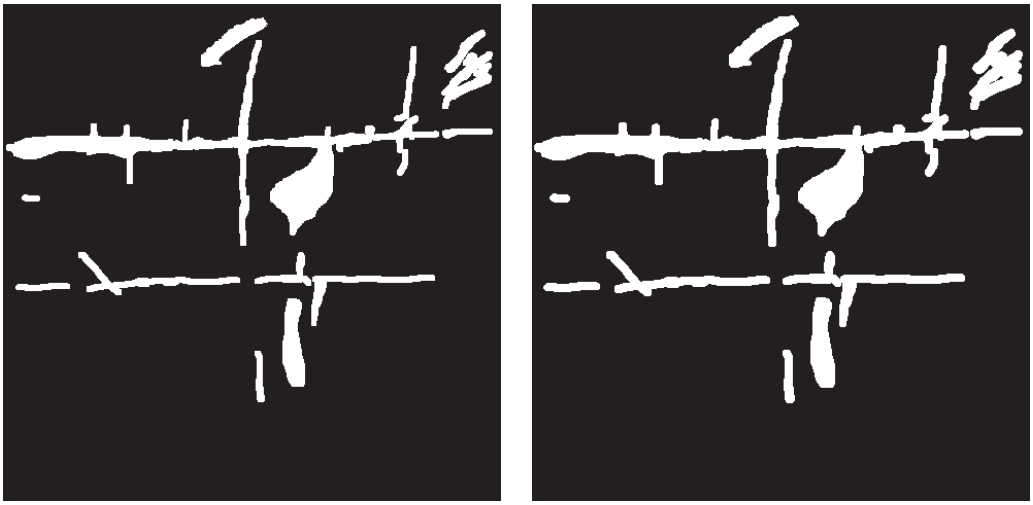


Abbildung 1.4.13: Bildmasken: Links um einen Pixel, Rechts um zwei Pixel vergrößert.

sich in der Rekonstruktion des tangentialen Anteils der Strömung unterscheiden. Die Abbildung 1.4.14 zeigt die Ergebnisse für vier verschiedene Varianten. Die einfachste Variante liegt darin, den tangentialen Anteil einfach zu ignorieren. Wie schon vorher beschrieben, ist die so entstehende Strömung sehr von der Ausrichtung des verwendeten Gitters abhängig. Dies führt bei Bildkanten, welche diagonal zu den Gitterkanten liegen, und bei gekrümmten Bildstrukturen zu Problemen. In dem hier verwendeten Beispiel werden diese Probleme aber von den Problemen bei der richtigen Wahl der Bildmaske überschattet, so dass das Ergebnis für diese Rekonstruktionsmethode überraschenderweise besser aussieht, als für die anderen Methoden, mit Ausnahme der Rekonstruktion mit vergrößerter Bildmaske.
Der Einfluss des verwendeten Diffusionskoeffizienten auf das rekonstruierte Bild kann anhand der Vergleiche in Abbildung 1.4.15 gesehen werden. Zudem wird noch einmal eine Gegenüberstellung der Resultate von zwei unterschiedlichen Methoden zur Behandlung des tangentialen Anteils durchgeführt.

Abbildung 1.4.14: Vergleich der Resultate für verschiedene Rekonstruktionsmethoden (ν=0.1). Von links Oben nach rechts Unten: Ohne tangentialen Anteil, interpolierter tangentialen Anteil, adaptiv extrapolierter tangentialen Anteil, interpolierter tangentialen Anteil mit vergrößerter Maske.

Abbildung 1.4.15: Gegenüberstellung der Resultate der Bildrekonstruktionen für verschiedene Diffusionswerte und Methoden. Von Oben nach Unten: ν=1, 0.1, 0.01 und 0.001. Links wurde der tangentialen Anteil der Strömung bezüglich der Gitterkanten ignoriert und Rechts unter Verwendung einer um zwei Pixel größeren Maske interpoliert.

Gänsestall

Dieses Beispiel ist ein Blick auf einen Gänsestall[38] mit einer Gans und ihren Küken. Für diese Beispiel wird die Graustufenversion des Bildes verwendet. Es ist mit 1200×1200 Bildpunkten schon sehr viel detailreicher als die vorherigen Beispiele und enthält sehr viele Bereiche mit unterschiedlichen Charakteristiken, unter ihnen auch stark strukturierte Bereiche mit Stroh, welche sehr viele kontrastreiche Kanten enthalten.
Als erstes soll versucht werden, eines der Küken aus dem Bild zu entfernen. Dazu wurde mit GIMP der Bereich des Kükens markiert und diese Information in eine Bildmaske umgewandelt. Aus dieser Maske wurde ein angepasstes Gitter mit 24612 Zellen und 25099 Knoten erzeugt. Ein Tensorproduktgitter für das ganze Bild hätte $1199^2 = 1437601$ Zellen und $1200^2 = 1440000$ Knoten gehabt, also ungefähr 58 mal so viele Zellen wie das angepasste Gitter.
Wie in Abbildung 1.4.16 rechts unten zu sehen ist, wurde der Bereich des Kükens durch einen glatten Flicken aufgefüllt. Die Kanten des umgebenden Strohs stoßen dabei an den Rand an. Dies erzeugt, je nach Richtung des Bildgradienten ∇I, bei der Rechnung einzelne Strömungsjets, welche in das Gebiet einströmen und Randbereiche, an denen ein starker Sog herrscht. Die dünnen Jets lösen sich schnell auf und hinterlassen im rekonstruierten Bild ein fleckiges Muster, was nicht zur Struktur des umgebenden Strohs passt. Der Randbereich, welcher das Federkleid des Nachbarkükens berührt, wird dagegen viel besser behandelt.
Durch die Bildmaske wurden dort die Spitzen der Flaumfedern abgeschnitten. Die Bildrekonstruktion versucht die Federn fortzusetzen und schafft es hier sogar über eine gewisse Strecke, bevor die Strukturen zerfließen und im Rest des Flickens aufgehen.
Versuche, die Strukturen des Strohs durch eine Verringerung der Diffusion besser fortzusetzen, führten leider zu unbefriedigenden Ergebnissen. Die Ergebnisse von Rechnungen für verschiedene Werte der Diffusion ν finden sich in Abbildung 1.4.17. Zwischen $\nu = 0.1$ und $\nu = 0.02$ traten interne Wirbel auf, welche den Wertebereich verschieben[39] und weiße und schwarze Flecken in der Darstellung erzeugen. Je kleiner der Wert von ν wurde, desto ausgeprägter, kontraststärker und auffälliger wurden die Flecken innerhalb des rekonstruierten Bereiches. Zudem ist zu sehen, dass die Flaumfedern des Nachbarkükens bei kleineren Diffusionswerten kürzer wurden.
Die Vermutung liegt nahe, dass das Problem bei der Entfernung des Kükens darin liegt, dass es sich vor einem stark strukturierten Hintergrund aus Stroh befindet und der von den Strohstrukturen zu überbrückende Bereich sehr viel größer als die Dicke der Strohhalme war. Würde dieselbe Rechnung vor einem weniger strukturierten Hintergrund durchgeführt werden, würde ein glatter Flicken für die Fehlstelle weniger auffallen oder keine starken internen Wirbel durch Jetströmungen induziert werden. Auch im Fall eines dünnen Kratzers durch das Küken,

[38] Das Motiv wurde von mir im Frühjahr 2011 in der Dortmunder Innenstadt aufgenommen. Zu dieser Zeit fand eine Art Ausstellung mit verschiedenen Tieren in der Fußgängerzone statt.
[39] Hier wurden die Werte bei der Visualisierung fest auf das Intervall $[0; 1]$ beschnitten

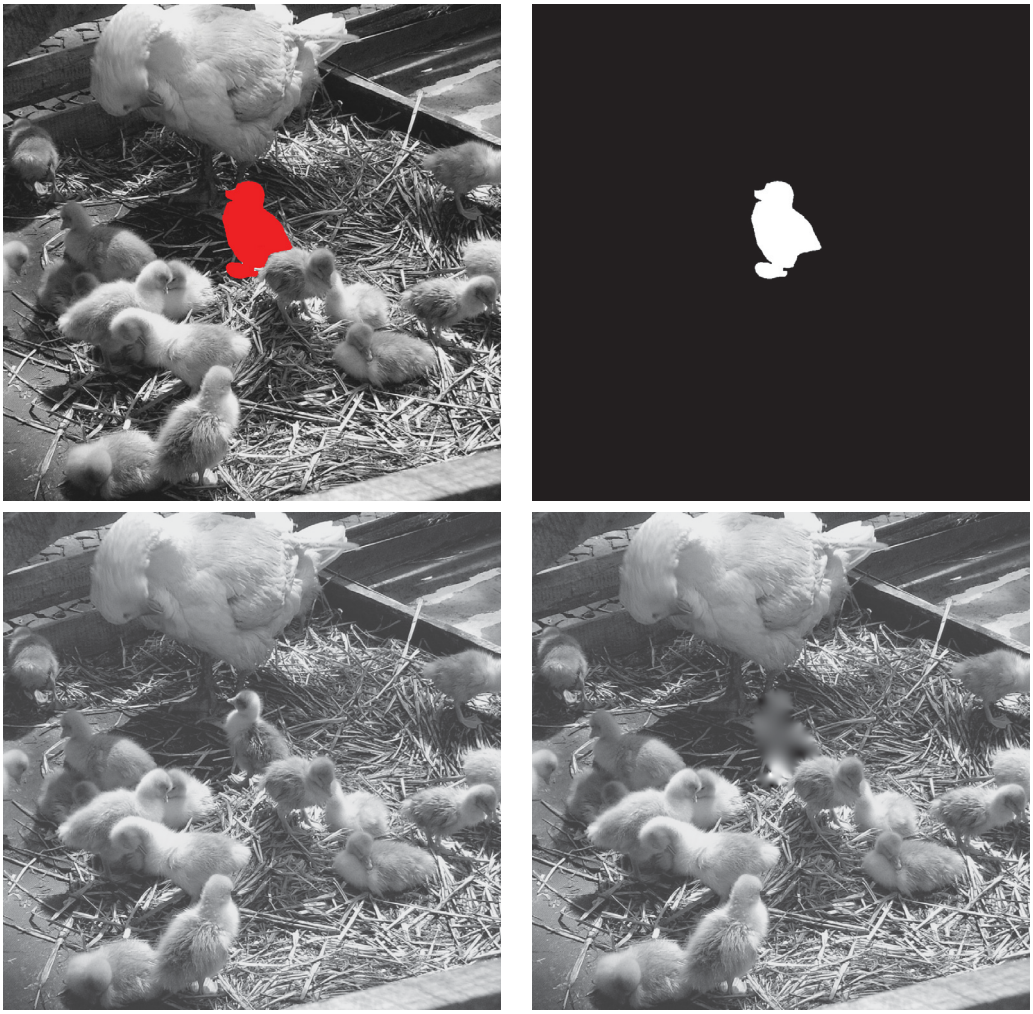


Abbildung 1.4.16: Von links oben nach rechts unten: Originalbild mit rot markiertem Küken, das Küken als Bitmaske, Originalbild und rekonstruiertes Bild für $\nu = 1.0$.

anstatt einer kompletten Entfernung des Kükens, würde wohl weniger Probleme bereiten, da einfach kein Platz für die Entwicklung eines großen internen Wirbels innerhalb eines solchen Kratzers ist.

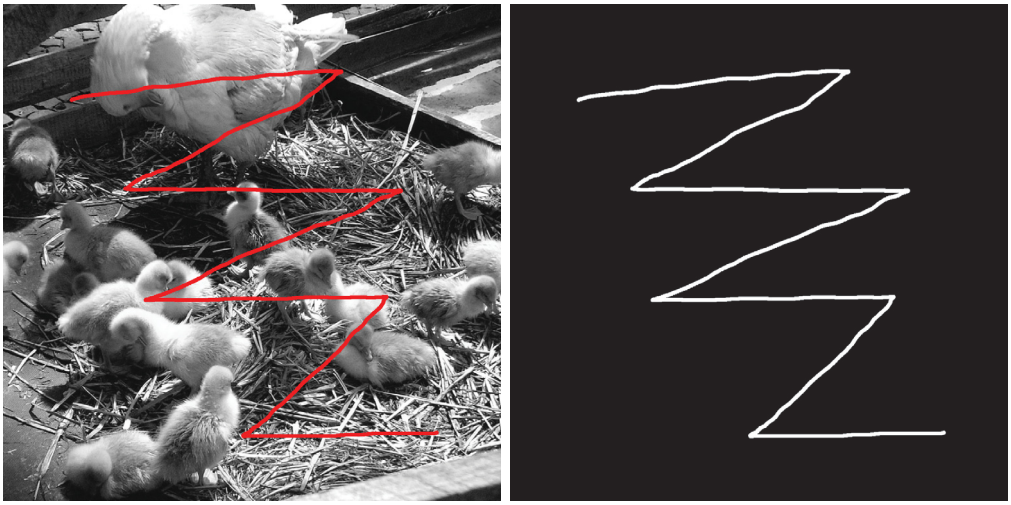
Letzteres wurde ebenfalls ausprobiert. In Abbildung 1.4.18 wurde ein langer dünner Zick-Zack-Kratzer quer über das Bildmotiv gelegt. Das angepasste Gitter dazu besitzt 42910 Zellen und 47670 Knoten. Obwohl der Kratzer über sehr unterschiedliche Bereiche verläuft, fiel das Ergebnis sehr zufriedenstellend aus. Es erfordert schon einen sehr genauen Vergleich von Originalbild und rekonstruiertem Bild, um die Veränderungen zu finden. Die Federstrukturen und die Schnäbel der Küken wurden sehr gut rekonstruiert und auch die Strohhalme, welche nicht gerade längs der Richtung des Kratzer liegen, wurden gut fortgesetzt.

Abbildung 1.4.17: Rekonstruktionen für verschiedene Werte von ν. Von links oben nach rechts unten: $\nu = 0.2$, $\nu = 0.1$, $\nu = 0.02$ und $\nu = 0.001$.

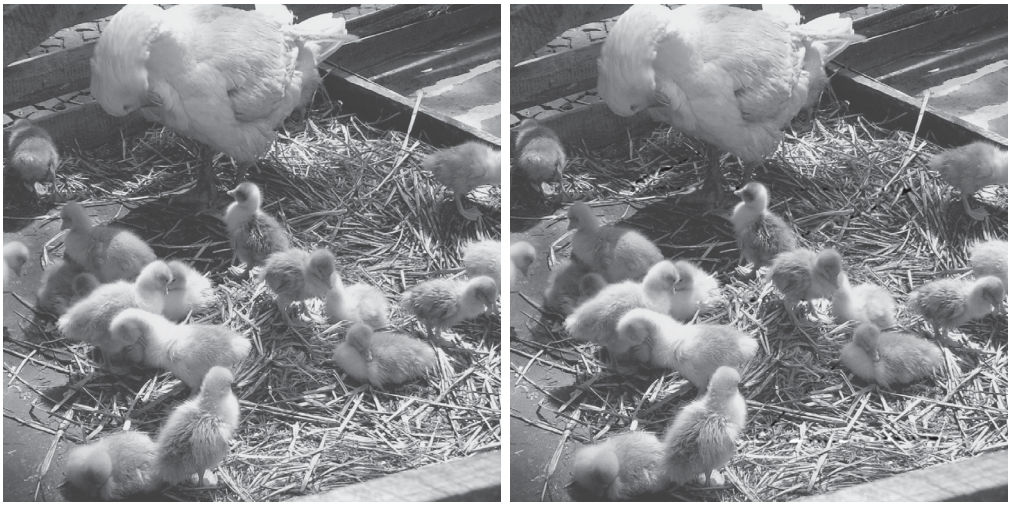
1.4.5 Erweiterung auf Farbbilder

Um nicht nur Graustufenbilder, sondern auch Farbbilder verarbeiten zu können, wurde in diesem Abschnitt eine Idee wieder aufgegriffen, welche auch bei der Definition des JPEG-Standards[40] zur Anwendung gekommen war.
Dabei wird das ursprüngliche Farbbild, das hier im RGB-Farbmodell mit den Farbkanälen für Rot, Grün und Blau so umgewandelt, dass man ein Bild erhält, in dem sich die Bildinformation mit höchster Wichtigkeit/Korrelation konzentriert und zwei weitere, welche weniger wichtige Extrainformationen kodieren. Beim JPEG-Standard wurde dabei das YCbCr-Farbmodell zur Umwandlung verwendet. Eine solche Zerlegung des Gänsestall-Bildes wurde mit Hilfe des Bildbearbeitungsprogramms GIMP durchgeführt. Die hier verwendete Einstellung für die Konversion war "YCbCr ITU R709". Das Originalbild und die drei Farbkanäle sind in der Abbildung 1.4.19 zu sehen. Der Y-Kanal entspricht dabei einer Graustufendarstel-

[40] ISO/IEC 10918



(a) Originalbild mit Kratzer (rot markiert) (b) Kratzer als Bitmaske



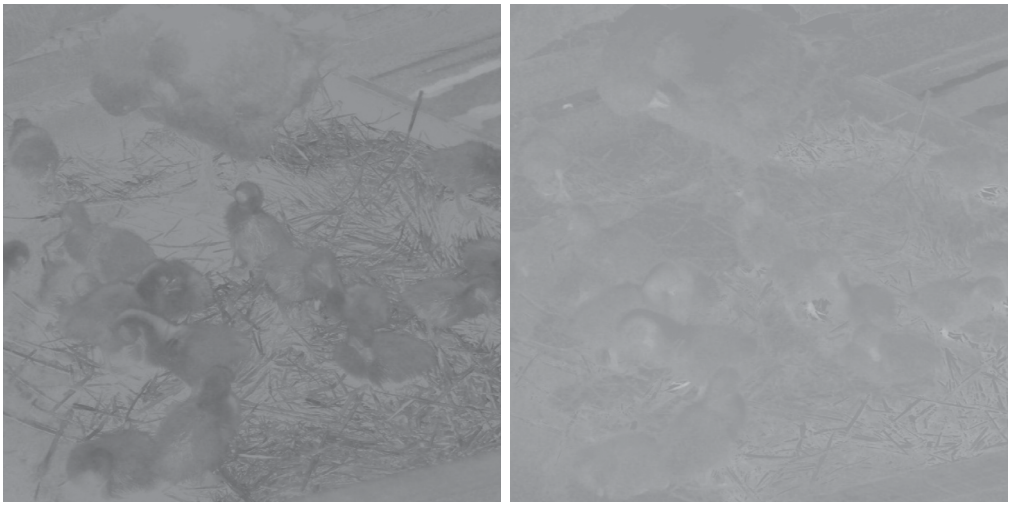
(c) Originalbild (d) rekonstruiertes Bild

Abbildung 1.4.18: Versuch einen Kratzer aus dem Gänsestallbild zu entfernen.

lung des Farbbildes und weist im Wesentlichen dieselbe Kontraststärke, wie das Farbbild auf. Die beiden anderen Farbkanäle wirken eher ausgebleicht und kontrastarm. Dies hat die Folge, dass die darin vorkommenden Bildgradienten und somit auch die entsprechenden Strömungsgeschwindigkeiten sehr viel kleiner sind.

Es wurden dieselben Zick-Zack-Kratzer aus dem vorangegangenen Beispiel zum Test herangezogen. Danach wurde die Bildrestauration für die einzelnen Farbkanäle getrennt durchgeführt. Dabei haben sich die verminderten Bildgradienten in den Cb- und Cr-Kanälen positiv auf die Rechenzeit ausgewirkt, da die adaptive Zeitschrittweitensteuerung des modifizierten “pp2d” größere Zeitschrittweiten wählen konnte und jeweils auch schneller zu einem stationären Ergebnis gekommen war. Mit ParaView wurden auch hier die einzelnen Rechnungen zusammengefasst und drei restaurierte Bilder erstellt. Mit Hilfe von Gimp wurden die drei Einzelbilder

(a) RGB-Farbbild (b) Y

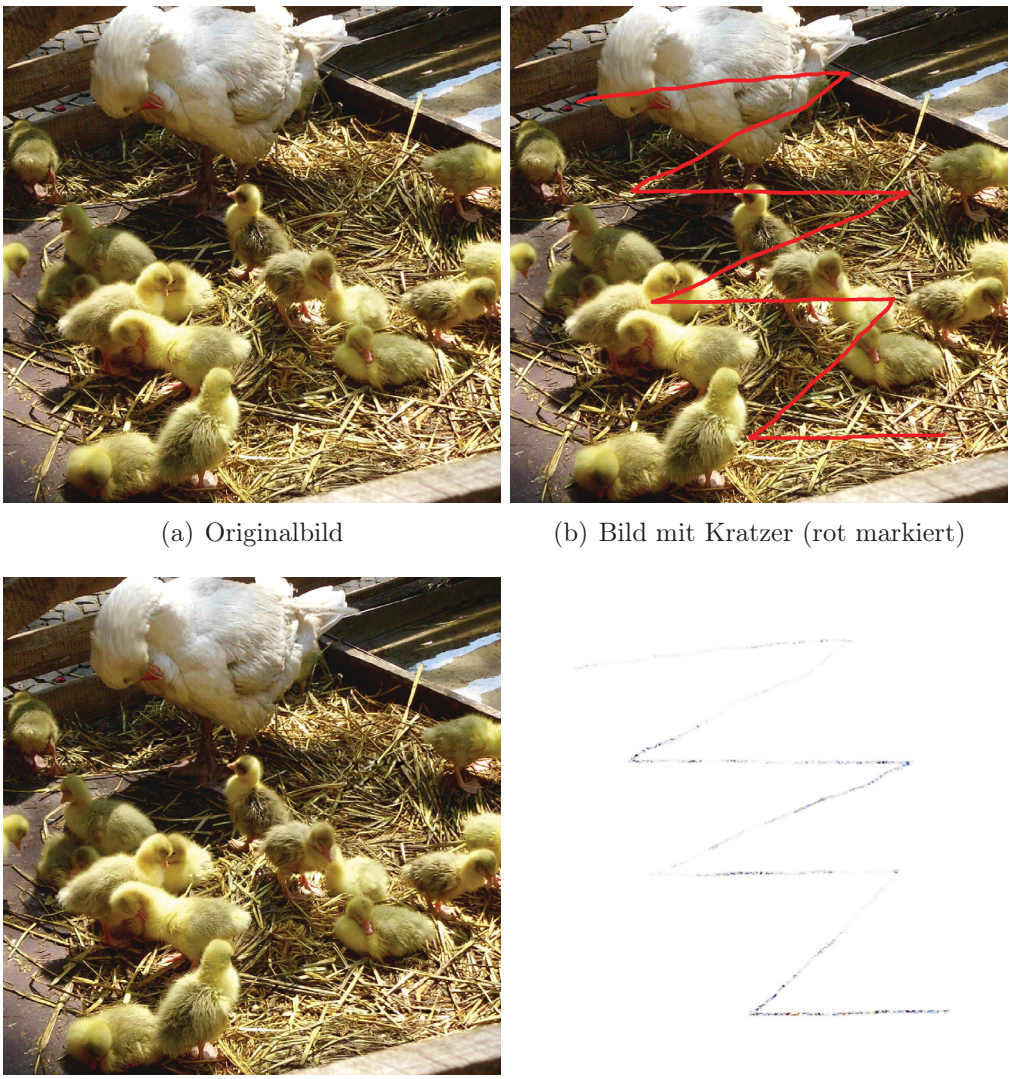


(c) Cb (d) Cr

Abbildung 1.4.19: Gänsestall als Farbbild und seine Zerlegung in Komponenten nach YCbCr ITU R709.

wieder zu einem Farbbild zusammengefasst und ins RGB-Farmodell rückkonvertiert. Ein Vergleich zwischen Originalbild und restauriertem Bild ist in Abbildung 1.4.20 zu sehen. Mit Hilfe von GIMP wurde auch eine Differenz der beiden Bilder zur Verdeutlichung der feinen Unterschiede angefertigt.

Die Qualität der Wiederherstellung kann sich dabei durchaus sehen lassen. Obwohl die Farbkanäle getrennt restauriert wurden, treten bei diesem Beispiel kaum Verfärbungen auf. Lediglich im unteren Bereich des Kratzers ist eine Stelle, bei der der Kratzer fast parallel zur lokalen Ausrichtung der Strohhalme ist und die Qualität der Rekonstruktion abnimmt. Dies ist aber lediglich eine Bestätigung einer erwarteten Schwäche des ursprünglichen Verfahrens, ist also kein Problem der Erweiterung auf Farbbilder, da auch dort in diesem Problembereich keine erkennbaren Fehlfarben auftreten.



(a) Originalbild (b) Bild mit Kratzer (rot markiert)

(c) rekonstruiertes Bild (d) Differenzbild (invertierte Darstellung)

Abbildung 1.4.20: Rekonstruktion eines Farbbildes

1.4.6 Fazit

Die hier vorgestellte Methode hat ihre Eignung für die Rekonstruktion von Graustufenbildern mit Kratzern gezeigt, sofern die Dicke der Kratzer nicht zu groß in Relation der zu überbrückenden Bilddetails ist (siehe Abbildung 1.18(d)). Auch Farbbilder können nach einer geeigneten Transformation des Farbmodells ausreichend gut damit behandelt werden.

Sofern die Kratzer geeignet in einer Bildmaske markiert worden sind, kann die Erzeugung der Rechengitter automatisiert ablaufen. Bei der Feineinstellung der Diffusionsstärke und Simulationsdauer (und diverser anderer Einstellungen der numerischen Software) der einzelnen Bereiche wird aber noch der Eingriff eines Benutzers benötigt. Falls man sich mit festen Werten von ν und einem passenden Abbruchkriterium für die Simulation zufrieden gibt, kann wahrscheinlich auch

dieser Bereich des Verfahrens automatisiert werden.
Es besteht Potential zur Verbesserung der hier vorgestellten Methode, indem die Diffusion nichtlinear von der Ströhmungsstärke abhängig gemacht wird und in Bereichen hoher Strömungsstärke die Diffusion drosselt. Dies würde der Verringerung der Diffusion in Bereichen großer Bildgradienten entsprechen und wäre vergleichbar mit einem Perona-Malik-Ansatz.[41] Auch der Einsatz des NAD-Operators (1.10) aus Abschnitt 1.3.2 anstelle des normalen Diffusionsoperators in der Navier-Stokes-Gleichung wäre hier denkbar, um weitere Möglichkeiten zur Optimierung des Ergebnisses zur erhalten. Dabei würde allerdings die Ausrichtung des lokalen Koordinatensystems, bezüglich dessen der Operator ausgerichtet ist, ebenfalls von der berechneten Strömung anstatt einem vorgegebenen Vektorfeld abhängen, was eine weitere nichtlineare Abhängigkeit hinzufügen würde. Dies ermöglicht im Gegenzug aber zusätzliche Möglichkeiten zur Optimierung der Bildwiederherstellung, die Möglichkeiten für weitere Forschungen bezüglich dieses Ansatzes bieten.
Die vorgestellte Methode verwendet zudem Gitter, welche Ausschnitte eines regelmäßig strukturierten Tensorproduktgitters sind. Dadurch wurde die Möglichkeit des FEM-Ansatzes, auch mit unstrukturierten Gittern arbeiten zu können, nicht verwendet. Eine Implementierung dieses "Inpainting"-Verfahrens mittels leichter zu implementierender finiter Differenzen-Ansätze (FD) ist damit ebenfalls möglich.

[41] Dies gilt da $\vec{v} = \nabla^{\perp} I \Rightarrow \|\vec{v}\| = \|\nabla^{\perp} I\| = \|\nabla I\|$ und somit eine Steuerung der Diffusion über ein Funktion $g(\|\vec{v}\|)$, der über $g(\|\nabla I\|)$ entspricht.

Kapitel 2

Nichtlineare anisotrope Transport-Diffusion (NATD)

Der NATD-Ansatz ergänzt die NAD Differentialgleichung aus Abschnitt 1.3.2 um einen Transportterm. Die Anisotropie des Diffusionsoperators wird dabei entlang eines lokalen, entlang der Transportrichtung orientierten, orthogonalen Koordinatensystems so ausgerichtet, dass die stärkste Diffusion in Richtung des Transports erfolgt. In Bereichen ohne Transport, in denen keine Ausrichtung definiert ist[1], soll der Operator beziehungsweise die Funktionen, welche die Diffusionsstärke für die verschiedenen Ausrichtungen liefern, mit ausreichender Glätte in den gewöhnlichen isotropen Diffusionsoperator übergehen. Damit kann der NATD-Operator dazu benutzt werden, eine beliebige Textur hauptsächlich in einer vorgegebenen Richtung durch Diffusion zu verschmieren und gleichzeitig entlang derselben Richtung zu transportieren. Eine Trennung der Hauptdiffusionsrichtung und Transportrichtung ist zwar möglich, ist aber hier nicht sinnvoll.
Der NATD-Operator L lautet, mit dem übernommenen Diffusionsoperator A des NAD Operators, folgendermaßen:

$$L(\vec{v}, \nabla I) := \vec{v} \cdot \nabla I - \operatorname{div}(A(\vec{v}, \|\nabla I\|)\nabla I). \tag{2.1}$$

Hierbei ist, wie schon zuvor, $\vec{v}$ ein vorgegebenes Vektorfeld auf dem Gebiet Ω und ∇I der Gradient der skalaren Funktion I auf Ω, welche Konzentrationen, Bildwerte und ähnliches bezeichnen kann. Der Operator L kommt in der Gleichung

$$\partial_t I + L(\vec{v}, \nabla I) = f(I) \tag{2.2}$$

zur Anwendung.
In dieser Form kann die Wohldefiniertheit des Problems sowohl für NATD als auch für NAD und Perona-Malik nicht allgemein garantiert werden, da der zur Berechnung von $A(\vec{v}, \|\nabla I\|)$ nötige Term $\|\nabla I\|$ nicht beschränkt ist oder keine ausreichende Glattheit vorweist. Um dieses Problem zu beheben, kann eine, zum Beispiel durch Faltung mit einem Gaußkern oder durch die Verwendung einer

[1] Dies bezeichnet Stellen mit $\nabla I = \vec{0}$.

Lösung eines Wärmeleitungsproblems mit Startlösung I, regularisierte Bildfunktion I_σ zur Berechnung von $A(\vec{v}, \|\nabla I_\sigma\|)$ benutzt werden. Mit dieser Modifikation erhält man

$$L_\sigma(\vec{v}, \nabla I) := \vec{v} \cdot \nabla I - \operatorname{div}(A(\vec{v}, \|\nabla I_\sigma\|)\nabla I). \tag{2.3}$$

Für eventuelle praktische Implementierungen hat dies aber wenig Relevanz, so dass man sich dabei auf (2.1) beschränken kann. Eine zusätzliche Glättung bedeutet in eventuellen Anwendungen meist nur einen rechnerischen Mehraufwand, dem aber kein gleichwertiger Ausgleich durch Verbesserung der Ergebnisse gegenübersteht. Da ein Transportterm enthalten ist, müssen aber numerische Implementierungen, die diesen Operator verwenden, zusätzlich stabilisiert werden. Dies wird später in Abschnitt 3.2.2 näher erläutert.

Die in Kapitel 3 präsentierte Anwendung berechnet Lösungen von (2.2) und verwendet nur reine Rauschtexturen als Startlösungen, die aus gleichverteiltem Rauschen im Intervall $[0; 1]$ gewonnen wurden. Die Anwendung des NATD-Operators auf dieses Rauschen erzeugt ausgerichtete linienartige Strukturen, welche mit der Zeit breiter und gröber werden. Dies bedeutet auch, dass feinere, weniger ausgeprägte Linien von benachbarten Linien absorbiert werden, was auch bedeutet, dass die Skala der sichtbaren Strukturen innerhalb des Skalenraumes zu gröberen Skalen hin wandert. Dieser Effekt ist gut in den Abbildungen in Abschnitt 2.3 zu sehen.

In einer Variation des dort angeführten Beispiels, kann der Effekt der anisotropen Diffusion zusammen mit dem Transportoperator in Zusammenarbeit mit einer kontrastverstärkenden rechten Seite (2.6) auch in Abbildung 2.0.1(b) betrachtet werden. Obwohl in diesem Beispiel hohe Kontraste vorhanden sind, tritt nur eine minimale Verschmierung der Trennlinie zwischen schwarzem und weißem Bereich auf. In beiden Beispielen wurde dasselbe Strömungsfeld benutzt.[2]

Um ein ähnliches Ergebnis zur erhalten, ist alternativ auch die Verwendung des Perona-Malik–Operators zusammen mit einem Transportoperator denkbar, da die nichtlineare Abhängigkeit der Diffusionsstärke von den lokalen Bildgradienten eine Diffusion normal zu den Bildgradienten und eine andere (schwächere) Diffusion/Antidiffusion in Abhängigkeit von einer Fluxfunktion [Wei98, p.15-17] entlang der Bildgradienten verursacht. Dieses Verhalten ist somit analog zur Aufspaltung des NAD/NATD-Operators und der Verwendung einer schwächeren Diffusion entlang starker Bildgradienten. Allerdings hat man damit weniger Kontrolle, als durch eine direkte Kontrolle der Diffusion über getrennte Funktionen für die einzelnen Richtungen.

[2] Ausrichtung normal zur Radialrichtung, Strömung skaliert mit dem Sinus des Abstandes vom Mittelpunkt des Kreises, konstant in der Zeit, alle Stromlinien sind geschlossene Kreise um denselben Mittelpunkt.

(a) Startzustand (b) Endzustand

Abbildung 2.0.1: NATD-Operator angewendet auf ein Schwarz/Weiß-Gebiet. Strömungsfeld tangential zur Radialrichtung und Stärke definiert als Sinusfunktion vom Radius. Parameter: c=300, ϵ=0.01, ρ=8.

2.1 Allgemeiner Ansatz

Ω sei ein beschränktes Gebiet mit einem stückweise Lipschitz-stetigem Rand. Im allgemeinen Fall sei zu jedem Zeitpunkt $t \in [0, \infty)$ das Strömungsfeld durch die Funktion $\vec{v} : [0, \infty) \times \Omega \to \mathbb{R}^d$ gegeben. Der symmetrische anisotrope Diffusionsoperator $A : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^{d \times d}$ ist, wie schon zuvor in Abschnitt 1.3.2 der Operator (1.10), in seiner Ausrichtung von einem Vektor und in seiner Stärke sowohl von der euklidischen Norm dieses Vektors, als auch von der Norm des Gradienten der Lösung des Problems abhängig.
Durch letztere Abhängigkeit wird der Operator und somit das gesamte Problem (2.4) nichtlinear. Es wird nun (2.1) in $\partial_t I + L(\vec{v}, \nabla I) = f(I)$ eingesetzt, um die folgende Differentialgleichung mit Anfangsbedingung I_0 für das NATD-Problem zu erhalten:

$$\begin{aligned} \partial_t I + \vec{v} \cdot \nabla I - \operatorname{div}(A(\vec{v}, \|\nabla I\|)\nabla I) = f(I) \quad &\text{in } \mathbb{R}^+ \times \Omega, \\ I(0, \cdot) = I_0(\cdot) \quad &\text{in } \Omega. \end{aligned} \tag{2.4}$$

Diese Gleichung muss allerdings noch um zusätzliche Randbedingungen für $\mathbb{R}^+ \times \partial\Omega$ ergänzt werden. Dies kann zum Beispiel

$$A(\vec{v}, \|\nabla I\|)\partial_n I = 0 \qquad \text{on } \mathbb{R}^+ \times \partial\Omega \tag{2.5}$$

sein. Oder wenn man Strömungen mit Ein- und Ausströmrand verwendet, kann man eine sogenannte "Do Nothing"-Randbedingung am Ausströmrand verwenden. Für den Einströmrand kann eine Funktion mit zeit- und ortsabhängigen Rauschdaten zur Definition einer Dirichlet-Randbedingung verwendet werden. Dies wird

auch bei der Anwendung der NATD-Gleichung zur Visualisierung von Strömungen verwendet, welche in Abschnitt 3 mit zugehörigen Beispielen in Abschnitt 3.3 besprochen wird.

Ein Beispiel für eine nützliche rechte Seite ist die folgende kontrastverstärkende Funktion $f : [0,1] \to \mathbb{R}^+$:

$$f(I) = \rho \cdot \big((2\,I - 1) - (2\,I - 1)^3\big) \tag{2.6}$$

Die Idee hinter dieser, auch von Rumpf und Preusser verwendeten [BPR01, PR99] Funktion liegt darin, dass Orte mit Werten im Bereich $(0.5; 1)$ mit einer "Quelle" ($f > 0$) und Werte im Bereich $(0; 0.5)$ mit einer "Senke" ($f < 0$) versehen werden. Um eventuelle Ausreißer wieder in das Intervall $[0; 1]$ zurückzuführen, ist ein Vorzeichenwechsel bei 0 und 1 nützlich. Die Werte 0, 0.5 und 1 sollten Nullstellen von f sein. Das (2.6) diese Eigenschaften erfüllt, kann auch direkt anhand des zugehörigen Graphens (siehe Abbildung 2.1.1) beurteilt werden.

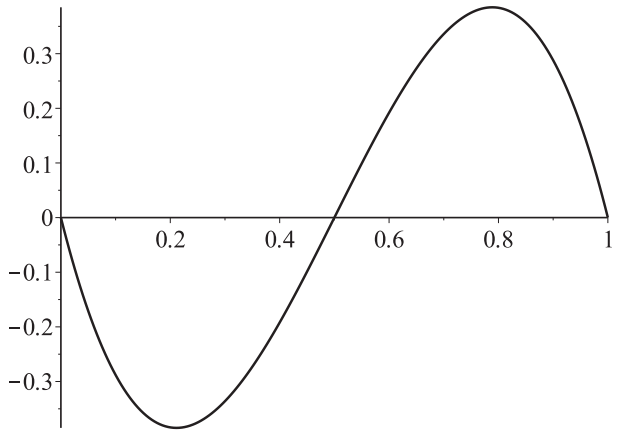


Abbildung 2.1.1: Plot von $f(I)$ aus (2.6) im Intervall $[0; 1]$ mit $\rho = 1$.

Die einfachste (polynomielle) Funktion, die alle diese Vorgaben erfüllt, ist das angegebene Polynom 3. Grades. Der Ansatz von Preusser und Rumpf zur Herleitung dieser Funktion basiert dabei auf Flux-Betrachtungen. Die Steigung im Punkte 0.5, welche direkt mit dem Grad der Kontrastverstärkung korrespondiert, lässt sich nun einfach über den Parameter $\rho \geq 0$ kontrollieren. Bei Verwendung eines (elementweise polynomiellen) FEM-Ansatzes auf die variationelle Formulierung des NATD-Problems (siehe (2.7)) hat ein Polynom zudem den Vorteil, dass sich die dabei entstehende rechte Seite durch eine Quadraturformel ausreichenden Grades exakt integrieren lässt.[3]

[3]Grad der zu integrierenden Funktion ist: 3 + Grad der Testfunktion.

2.2 Variationelle Formulierung für FEM

Die schwache Formulierung des NATD-Problems mit Neumann-Randbedingungen lautet: Finde ein Paar $(I, \partial_t I)$ aus den Räumen $L^2(0,T;H^1(\Omega))$ und $L^2(0,T;H^{-1}(\Omega))$, welche folgende variationelle Gleichung erfüllt

$$\begin{aligned} \langle \partial_t I, \phi \rangle + \int_\Omega \phi \cdot (\vec{v}^\top \nabla I) + \nabla \phi^\top A(\vec{v}, \|\nabla I\|) \nabla I \, \mathrm{d}\vec{x} &= \int_\Omega \phi \cdot f(I) \, \mathrm{d}\vec{x} \quad \forall \phi \in H^1(\Omega), \\ I(0,\cdot) &= I_0(\cdot) \qquad \text{in } \Omega. \end{aligned} \tag{2.7}$$

Die Wahl des Testraumes $H^1(\Omega)$ und das Weglassen des eigentlich entstehenden Randtermes impliziert die homogene Neumann-Randbedingung.
Nur in den wenigsten Fällen würde sich aber damit direkt eine analytische Lösung finden lassen. Als nächstes werden daher vorbereitende Schritte für eine Berechnung numerischer Lösungen mit der FEM-Methode angeführt. Dafür wird das kontinuierliche Problem (2.7) zuerst in der Zeit (2.8) und danach im Raum diskretisiert.

2.2.1 Diskretisierung in der Zeit

Sei nun δt_n die Zeitschrittweite der Zeitdiskretisierung im n-ten Zeitschritt. Dann lautet die linearisierte semi-implizite Euler Formulierung von (2.7): Finde für jeden Zeitschritt eine Lösung $I_n \in H^1(\Omega)$ mit

$$\begin{aligned} &\int_\Omega \phi \left\{ \frac{I_n - I_{n-1}}{\delta t_n} + \vec{v}_n \cdot \nabla I_n \right\} \\ &+ \nabla \phi \cdot A(\vec{v}_n, \|\nabla I_{n-1}\|) \nabla I_n \, \mathrm{d}\vec{x} = \int_\Omega \phi f(I_{n-1}) \, \mathrm{d}\vec{x} \quad \forall \phi \in H^1(\Omega), \\ &I(0,\cdot) = I_0(\cdot) \qquad \text{in } \Omega. \end{aligned} \tag{2.8}$$

Auch hier sind die zusätzlich benötigten Randbedingungen nicht zu vergessen!
Die Verwendung des semi-impliziten Eulers ist hierbei natürlich nicht zwingend. Wenn die Zeitschritte klein genug sind, um die dazu passende CFL-Bedingung zu erfüllen, sind natürlich auch wieder expliziter Euler, Runge-Kutta oder Crank-Nicholson verwendbar. Bei einer Verwendung des expliziten Eulers kann man später in Abschnitt 2.2.2 die Vermeidung von Lösungen von Gleichungssystemen gegen kleinere Zeitschritte eintauschen. Was insgesamt schneller ist, hängt danach von den verwendeten Lösern, der Konditionierung der Matrix A, der konkreten Implementierung der numerischen Software und eventuell verwendeter spezialisierter[4] Hardware ab.

[4]Grafikkarten von Nvidia können mittels CUDA® und anderen Methoden zur Beschleunigung numerischer Berechnungen verwendet werden. Weiterführende Informationen sind auch bei `https://developer.nvidia.com/accelerated-computing` zu finden.

2.2.2 Diskretisierung im Raum

Das Gebiet Ω sei nun durch ein konvexes, Viereckszellengitter T mit Innenwinkeln $0 < \theta < 180$ polygonal approximiert, so dass keine Zelle des Gitters über Ω hinausragt. Wobei $\Omega_h := \bigcup_{K\in T} K$ und K hier jeweils einzelne Zellen des Gitters bezeichne. Der Raum der bilinearen FEM-Funktionen auf dem Referenzelement $\hat{K} := [-1;1]^2$ sei mit Q_1 bezeichnet. Für jedes Element K des Gitters gibt es nun eine invertierbare bilineare Abbildung $R_K : \hat{K} \to K$, welche das Referenzelement $\hat{K}$auf das Element K abbildet. Damit kann der Ansatz- beziehungsweise Testraum V und V_0 durch die FEM-Räume $V_h := \{\psi \in H^1 : (\psi \circ R_K) \in Q_1 \, \forall K \in T\}$ und $V_{0,h} := \{\psi \in H_0^1 : (\psi \circ R_K) \in Q_1 \ \forall K \in T\}$ mit abzählbaren Knotenbasen $\{\psi_i\}_{i=1}^{N}$ und $\{\phi_i\}_{i=1}^{N_0}$ ersetzt werden. Die Lösung I wird nun für jeden Zeitschritt als $I_{h,n} := \sum_{i=1}^{N} u_{i,n}\psi_i$ mit den Koeffizienten $u_{i,n} \in \mathbb{R}$ dargestellt. Dies ergibt die FEM-Diskretisierung des Problems: Finde für jeden Zeitschritt t_n ein $I_{h,n} \in V_h$ mit

$$\begin{aligned} &\int_{\Omega_h} \phi_i \left\{ \frac{I_{h,n} - I_{h,n-1}}{\delta t_n} + \vec{v}_n \cdot \nabla I_{h,n} \right\} \\ &+\nabla\phi_i \cdot A(\vec{v}_{h,n}, \|\nabla I_{h,n-1}\|)\nabla I_{h,n} \, \mathrm{d}\vec{x} = \int_{\Omega_h} \phi_i f(I_{h,n-1}) \, \mathrm{d}\vec{x} \quad \forall \phi_i \in V_{0,h}, \\ &I_{h,0} = I_{0,h}. \end{aligned} \tag{2.9}$$

Dabei ist $\vec{v}_{h,n}$ die Strömungsfunktion im n-ten Zeitschritt, welche im Ort auf Ω_h eingeschränkt wurde und $I_{0,h}$ die L^2-Projektion der Startlösung I_0 auf den FEM-Ansatzraum.
Da das Problem (2.9) für jeden einzelnen Zeitschritt in linearisierter Form vorliegt, kann es durch Verwendung der Definition von $I_{h,n}$ in jedem Zeitschritt in ein Gleichungssystem mit den Unbekannten $u_{i,n}$ umgeschrieben werden. Für diese Darstellungsform wurden von mir Programme geschrieben, welche Probleme dieser Art unter Zuhilfenahme der FEAST-Numerikbibliothek lösen konnten. Beispiele dafür sind in Abbildung 2.0.1, Abbildung 2.3.2 und in Abschnitt 3.3 zu sehen.

2.3 Beispiel: Kreisströmung

Um ein einfaches Beispiel für die Anwendung des NATD-Operators zu konstruieren, wurde von mir ein Kreis mit Radius 1 mit dem Nullpunkt als Mittelpunkt als Gebiet Ω gewählt. Auf diesem Gebiet wird ein konstantes Strömungsfeld $\vec{v}$ mit in sich geschlossenen Stromlinien und Nullrandbedingung $\vec{v}_{|\partial\Omega} = (0,0)^\top$ definiert:

$$\vec{v}(x_1,x_2) := \begin{cases} \begin{pmatrix} 0 \\ 0 \end{pmatrix} & \text{wenn } x_1^2 + x_2^2 = 0 \text{ oder } x_1^2 + x_2^2 = 1, \\ \frac{\sin(4\pi\sqrt{x_1^2+x_2^2})}{2\sqrt{x_1^2+x_2^2}} \begin{pmatrix} -x_2 \\ x_1 \end{pmatrix} & \text{sonst} \end{cases} \tag{2.10}$$

Eine Vektordarstellung dieses Strömungsfeldes ist in Abbildung 2.2(a) zu sehen. Wie klar aus der Definition der Strömung ersichtlich ist, ist diese nur vom Abstand zum Nullpunkt $r := \sqrt{x_1^2 + x_2^2}$ abhängig. Daher könnte man die Koordinatenparameter x_1 und x_2 auch durch r ersetzen. Die Strömungsstärke beträgt somit $\|\vec{v}(r)\| = \left|\frac{1}{2}\sin(4\pi r)\right|$.
Für die räumliche Diskretisierung wurde ein hierarchisches, durch iterative Verfeinerung eines Grobgitters erzeugtes, Rechengitter verwendet. Das verwendete Grobgitter ist in Abbildung 2.3.1 zu sehen. Dieses Gitter weist eine 120°-Symmetrie auf, welche auch unter gleichmäßiger Verfeinerung erhalten bleibt. Der innere Kern wird aus drei Rauten gebildet, die von einem Ring von Elementen umgeben sind. Dieser Ring verhindert, dass am Rand durch die Verfeinerung Elemente mit einem Winkel von nahezu 180° gebildet werden, welche schlecht für die Konditionierung der bei der Lösung auftretenden Matrizen wäre. Dieses Grobgitter wurde mittels GRID3D[5] aus der originalen FEATFLOW-Gitterbeschreibung (siehe Anhang C.1) in das Gitterformat von FEAST konvertiert. Dieses Grobgitter mit 9 Zellen wurde für die Rechnung 7 mal gleichmäßig verfeinert, was ein Gitter mit genau $9 * 4^7 = 147456$ Zellen erzeugt.

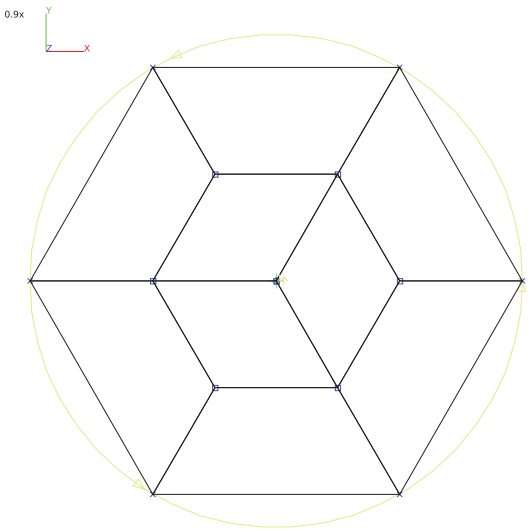
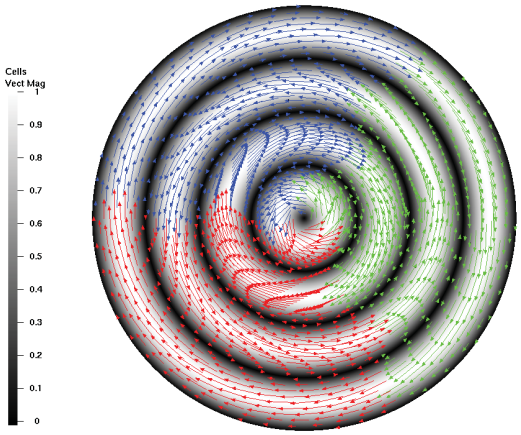


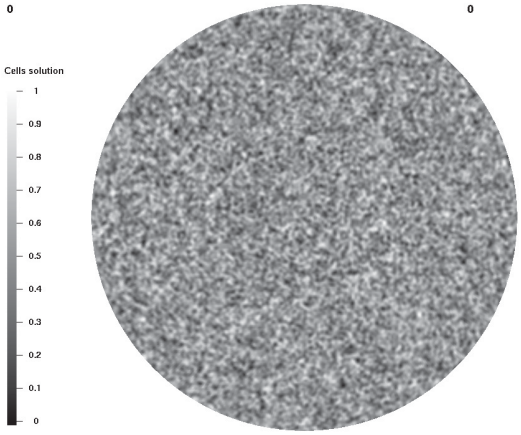
Abbildung 2.3.1: Grobgitter für einen Kreis mit Radius 1.

Die Berechnungen wurden mit einem von mir geschriebenen FEAST-Programm durchgeführt, welches das NATD-Problem (2.4), wie in Abschnitt 2.2.1 beschrieben, mit implizitem Euler in der Zeit und, wie in Abschnitt 2.2.2 beschrieben, mit unstrukturierten finiten Elementen mit Q_1-Ansatzraum auf dem Referenzelement im Raum diskretisiert. Zur Stabilisierung des Transportterms wurde Stromliniendiffusion (siehe dazu auch Abschnitt 3.2.2) verwendet.
Die Startlösung wurde erzeugt, indem eine Textur mit 257x257 Bildpunkten mit gleichverteiltem Rauschen aus dem Intervall $[0; 1]$ gefüllt wird. Danach wird diese

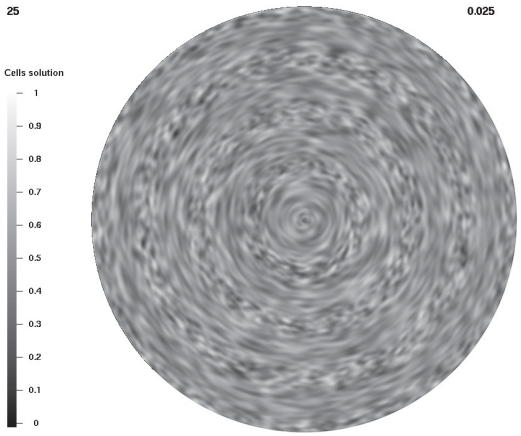
[5] `http://www.feast.tu-dortmund.de/downloads.html`



(a) Vektorfeld



(b) Startlösung



(c) Endlösung

Abbildung 2.3.2: Beispiel für die Anwendung des NATD-Operators.

Textur über das Rechengitter gelegt. An den Knoten des Gitters werden, mittels bilinearer Interpolation der Werte der nächsten Bildpunkte, jeweils Werte

zur Initialisierung der diskreten Startlösung errechnet. Das Ergebnis davon ist in Abbildung 2.2(b) zu sehen.
Die Diffusionsfunktionen für dieses Beispiel wurden aus Abschnitt 3.2.1 übernommen. Die Parameter um ein Resultat wie in Abbildung 2.2(c) zu erhalten lauten: $\Delta t = 10^{-3}$, $v_{min} = 10^{-3}$, $\beta = 10$, $\epsilon = 10^{-3}$, $c = 300$, $\rho = 3.0$ und $sd = 0.8$. Nach einer Simulationszeit von 0.025 Zeiteinheiten erhält man dann das gesuchte Resultat.
Man kann aber auch analytisch, zum Beispiel durch die Auswertung der Funktion

$$I_0(x_1, x_2) := \begin{cases} 0 & \text{für } x_1 < x_2 \\ 1 & \text{für } x_1 \geq x_2 \end{cases},$$

ein einfaches schwarz-weißes Testmuster vorgeben, wodurch man das Beispiel aus Abbildung 2.0.1 erhält.

Kapitel 3

Darstellung von Strömungsstrukturen

Bekanntermaßen sind die Analyse und Nachbearbeitung von Rohdaten und Ergebnissen die Grundaufgaben von wissenschaftlichen Visualisierungen. Fortschrittliche Multiskalenvisualisierungsmethoden werden zur Darstellung und Analyse der Strukturen von instationären Strömungsfeldern benötigt, für die Standardmethoden ungeeignet sein können. Eine große Auswahl von Visualisierungstechniken sowohl für stationäre als auch instationäre Strömungsfelder in 2D und 3D wurden in den letzten Jahren präsentiert. Die Auswahl reicht von Partikelverfolgungstechniken [TB96, Wij93] über texturbasierten Methoden [DPR00, Wij91, CL93, SK97, IG97] bis zur Erkennung und Extraktion von Strukturen in 3D Strömungsfeldern [CPC90, TJ82, HWM88, JH95]. Eine Übersicht über die Kategorien verfügbarer Visualisierungsmethoden kann bei [LHD+04] gefunden werden. Da das Gebiet der Visualisierung laufendem Wachstum durch neue Methoden und Variationen unterworfen ist, kann diese Übersicht nicht vollständig sein.
Die Diskretisierung, die in [BPR01] verwendet wird, basiert auf Quadtree- und Octtree-Gittern, die für reale Anwendungen ungeeignet sind, da die von CFD-Rechnungen für reale Anwendungen gelieferten Strömungsdaten eher auf unstrukturierten als auf gleichmäßigen Rechteckgittern[1] vorliegen. Eine vorherige Interpolation der errechneten Daten auf ein Tensorproduktgitter würde zum Einen in den meisten Fällen sehr viel mehr Zellen und damit mehr Speicher verbrauchen und zum Anderen einen zusätzlichen Interpolationsfehler erzeugen. Der erhöhte Verbrauch an Zellen liegt vor allem daran, dass es unmöglich ist, mit äquidistanten Tensorproduktgittern eine lokale Anpassung der Gitterweite durchzuführen, zum Beispiel um eine Randschicht um einen umströmten Zylinder zu legen, was zu einer großen Verschwendung durch unnötig feine Gitter in anderen Bereichen führt. Um diese Probleme zu vermeiden, werden für die hier vorgestellte Methode nur unstrukturierte Gitter in Verbindung mit einer FEM-Diskretisierung der auftretenden Transport-Diffusions-Probleme verwendet. Zur Stabilisierung der Transportterme wird Stromliniendiffusion (SD) verwendet und es werden verschiedene iterative Krylov-Raumlöser oder Mehrgitterverfahren zur Lösung der auftretenden

[1] Im Weiteren als Tensorproduktgitter bezeichnet.

nichtsymmetrischen Gleichungen miteinander verglichen.
Dies stellt eine echte Erweiterung der verfügbaren Möglichkeiten zur Visualisierung auch komplexerer Problemstellungen dar. Zudem bietet die Verwendung des FEM-Ansatzes (und die verwendete numerische Bibliothek FEAST) die Möglichkeit, anstatt der verwendeten Viereckselementen mit Q_1-Ansatzraum auch andere Elemente mit höhergradigem Ansatzraum zu verwenden.
Das Visualisierungsverfahren wird an verschiedenen Anwendungsbeispielen vorgeführt, zum Beispiel Wirbelströmungen hinter einem Zylinder oder einer Venturipumpe. Teilweise wurde dieses Visualisierungsverfahren auf unstrukturierten Gittern zuvor von mir in Vorträgen vorgeführt, die ich im Rahmen der "GAMM[2] 2006"-Tagung in Berlin und des DFG-Programms SPP 1114 gehalten habe und die in verkürzter Form in [DKMT08] veröffentlicht wurden.
Zudem werden die Effekte verschiedener Parametereinstellungen mittels Beispielen analysiert und eine eigene Überblendungsstrategie vorgestellt, deren Start durch eine von mir erdachte Anpassung einen weniger unebenen Start hat. Anhand dieser Beispiele wird gezeigt werden, dass die NATD-Technik ein vorzüglicher Kandidat zur effizienten Darstellung von instationären Strömungen mit komplizierten Mehrskalenstrukturen in Raum und Zeit ist.

3.1 Problemstellung

Strömungen werden mathematisch normalerweise in zwei verschiedenen Formen dargestellt, die sich aus unterschiedlichen Betrachtungsweisen ergeben.
In der sogenannten Lagrange-Darstellung wird die Bewegung jedes Punktes beginnend vom Zeitpunkt $t = 0$ über die Zeit hinweg verfolgt. Das bedeutet, dass für jeden Punkt $\vec{x}_0$ eine Funktion $\vec{x} : [0;T] \to \Omega$ mit $\vec{x}(0) = \vec{x}_0$ existiert. Der Betrachtungsfokus wandert also mit der Strömung mit. Solange eine Bewegung vorliegt[3], kann sogar für jeden Zeitpunkt ein lokales Koordinatensystem basierend auf dem Tangential- und Normalvektor der Bewegung eindeutig definiert werden. Die Besonderheiten bei Ein- und Ausströmung in Ω werden aus Gründen der Übersichtlichkeit hier ignoriert. Bei stationären Strömungen entspricht die Bildmenge dieser Funktionen den Stromlinien.[4]
In der Euler-Darstellung bleibt der Betrachtungsfokus fixiert. Für jeden Punkt wird das Strömungsfeld über die Zeit hinweg, beziehungsweise die Abbildung $\vec{u} : \Omega \times [O;T] \to \mathbb{R}^n$, betrachtet $(\vec{u}(\cdot, 0) := u_0)$.
Zur numerischen Berechnung beider Darstellungsformen existieren jeweils sehr unterschiedliche Diskretisierungen und Berechnungsmethoden mit unterschiedlichen Anwendungsbereichen:

[2]Gesellschaft für Angewandte Mathematik und Mechanik e.V.

[3]Ohne eine Bewegung ist der Tangentialvektor zur Bewegungsrichtung gleich dem Nullvektor, wodurch sich keine eindeutige Richtung definieren lässt.

[4]Stromlinien = Isolinien der Stromfunktion im stationären 2D-Fall.

Lagrange: Simulation des Verhaltens von sehr vielen einzelnen Partikeln und ihre Interaktionen untereinander, mit den Gebietsrändern beziehungsweise Kontaktflächen, und eventuell vorgegebenen Kräften, wie zum Beispiel der Erdbeschleunigung. Um dies zu durchführen zu können, werden Partikel in Einzelbereichen zu "Zellen" zusammengefasst und das Verhalten dieser Partikel im statistischen Mittel durch die Bewegung dieser Zellen ausgedrückt. Dies wird zum Beispiel in der Strukturmechanik und der Finite-Elemente-Methode (FEM) angewendet. FEM ermöglicht dabei die Verwendung von Ansätzen höherer Ordnung, welche bei entsprechender Glattheit der Lösung eine höhere Genauigkeit bei gleicher Zellenzahl oder die Reduktion der Zellen bei gleicher Genauigkeit ermöglicht. Der Ansatz eignet sich aber wenig für kompressible Strömungen.

Euler: Auch hier wird das Gebiet in "Zellen" eingeteilt, die jedoch an fester Position verharren. Dabei wird die Ein- und Ausströmung über die Seiten dieser Zellen und die enthaltene Substanz simuliert. Die dabei übliche Methode ist die Finite-Volumen-Methode (FVM). Dies wird bei der Simulation von Strömungen (kompressibel und inkompressibel) verwendet, ist aber nur ein Ansatz 1. Ordnung.

Für inkompressible Fluide kann aber auch die Finite-Elemente-Methode zur Anwendung kommen. Ein Beispiel dafür sind die Programme aus dem FeatFlow-Numerikpaket oder Feast-Numerikpaket, wobei letzteres finite Elemente höherer Ordnung erlaubt.

Die Visualisierung von Strömungen durch Partikelverfolgung, bei der eine vorher berechnete Strömung durch die Bewegung einzelner Partikel dargestellt wird, gehört dabei zu den Lagrange-Darstellungen.
Neben diesen beiden (klassischen) Ansätzen gibt es auch noch die Lattice-Bolzmann-Methode (LBM), die auf der Lattice-Bolzmann-Gleichung beruht, bei der die Menge aller möglichen Zustände[5] auf einem Gitter diskretisiert wird und deren Änderung durch Strömung und Kollision[6] simuliert wird.
Egal in welcher Form diese Daten nun vorliegen, beziehungsweise berechnet wurden, müssen diese Daten am Ende so aufbereitet und dargestellt (visualisiert) werden, dass die relevanten[7] Informationen für den Betrachter erkennbar werden. Im Weiteren sollen hier die Wirbelstrukturen der Strömung und deren zeitliche Entwicklung von Interesse sein.
Eine Strömung kann nun sowohl kleinskalige, als auch grobskalige Strukturen, sowie langsame Strömungen, als auch sehr schnelle Strömungen gleichzeitig enthalten. Wenn zusätzlich am Anfang der Visualisierung sogar Unklarheiten bezüglich der Lage und Art der Strömungsstrukturen[8] bestehen, ist ein Verfahren nötig,

[5] Phasenraum. Enthält im Gegensatz zum Zustandsraum keine Zeitkomponente

[6] Dies sind die zwei Prozesse, die im einfachsten Fall des Lattice-Bolzmann-Algorithmus in jedem Zeitschritt durchgeführt werden.

[7] Was genau "relevant" ist, hängt sehr von der jeweiligen Problemstellung ab!

[8] Zum Beispiel: Wirbel und laminare Bereiche.

dass die Strukturen aus verschiedenen Skalen gleichzeitig extrahieren und in einer Darstellung vereinen kann. Dazu eignen sich die zuvor erwähnten texturbasierten Techniken.

3.2 Texturbasierte Technik

Texturbasierte Techniken beruhen darauf, dass die Wirkung der Strömung auf ein vorgegebenes Muster/Textur mittels vorgegebener Operatoren sichtbar gemacht wird. Um zu vermeiden, dass irrtümlich Strukturen der Textur als Struturen der Strömung fehlinterpretiert werden, sollte die Textur idealerweise keine erkennbaren Muster aufweisen. Andererseits sollten die Auswirkungen der Strömung gut erkennbar sein, was (lokal) feinskalige Strukturen erfordert.

Beide Anforderungen lassen sich durch Verwendung von (gleichverteiltem) Rauschen erfüllen. Um dabei keine Strukturen zu erzeugen, sollte der verwendete Pseudozufallszahlengenerator eine gewisse Mindestgüte aufweisen. Falls eine Wiederholbarkeit der Berechnungen erwünscht sein sollte, zum Beispiel um die Auswirkung von unterschiedlichen Parameterwahlen zu vergleichen, so sollte der Startwert des Pseudozufallszahlengenerators fest vorgebbar sein. Da auch die Qualität des Zufallszahlengenerators zwischen Betriebssystemen und Compiler-Umgebungen variiert, sollte in diesen Fällen eine selbst bereitgestellte Programmroutine verwendet werden. Es gibt zum Beispiel eine freie Implementierung für Fortran 90 des Mersenne-Twisters MT19937[9], die sich für diese Anwendung eignet.

Wenn nun die Werten in den Knoten des Gitters mit Rauschwerten einfach direkt mit Zufallswerten gefüllt werden, so ergeben sich bei unstrukturierten Gittern in den meisten Fällen visuelle Artefakte, die auf unterschiedliche lokale Gitterweiten zurückgeführt werden können. Die unterschiedlichen Gitterweiten korrespondieren bei diesem Ansatz direkt mit lokal unterschiedlichen Skalengrößen. Man kann somit unterschiedliche Bereiche des Gitter im Ergebnis der Initialisierung sehen. Ein Beispiel dafür sind die feinen Randschichten in Gittern, die für gewöhnlich um zu umströmende Hindernisse gelegt werden.

Um dies zu vermeiden oder zumindest zu reduzieren, wurde in den hier verwendeten Programmen von mir ein Umweg gewählt. Dabei wird ein Bild mit zum Gebiet passendem Höhen- und Seitenverhältnis erzeugt, dessen Auflösung in Pixeln gerade fein genug ist, um in etwa der mittleren Gitterweite zu entsprechen. Diese Textur wird mit Rauschdaten gefüllt, optional durch die Anwendung eines Diffusionsfilters geringfügig diffundiert und somit regularisiert, und danach auf die Knoten des Gitters abgebildet[10]. Letzteres, indem für jede Knotenkoordinate ein Punkt in der Textur bestimmt wird und die nächstliegenden Pixelwerte bilinear interpoliert werden. Diese Vorgehensweise verhindert, dass Bereiche mit sehr feiner Gitterweite optisch hervorstechen. Die Methode kann aber nicht verhindern, dass Bereiche mit zu grobem Gitter auffallen. Zumindest wird eine gewisse

[9] `http://jblevins.org/mirror/amiller/mt19937.f90`

[10] Im Englischen als “texture mapping” bekannt.

Entkoppelung der lokalen Gitterweite von der Skalengröße der Rauschdaten ermöglicht.
Dieses Problem tritt nicht auf, wenn man sich wie bei [DPR00, BPR00] auf äquidistante Tensorproduktgitter beschränkt. Bei [BPR00] werden die Knoten/Pixel in einem regulären Gitter mit Rauschdaten gefüllt und danach mittels Faltung regularisiert. Im Einströmungsbereich des Randes wird eine 1D Textur mit Rauschdaten gefüllt und dann mittels einer Faltung mit einem Gaußkern regularisiert und dies zur Definition der Randdaten über eine Projektion verwendet. Damit endet die Ähnlichkeit der Vorgehensweisen aber. Im Inneren wird die Technik auf Faltungen entlang der Stromlinien zurückgeführt und mittels Linienintegralmethoden (LIC) berechnet.
Die hier verwendete Projektion zum Auffüllen des gesamten Gebietes kann als eine Erweiterung der 1D-Methode interpretiert werden, die eine Erweiterung der Gesamtmethode auf unstrukturierte Gitter ermöglicht. Zudem können bei den Vorarbeiten von Preusser und Rumpf nur Eingabedateien mit konstanter Zeitdifferenz verwendet werden, die jeweils konstant in der Zeit bis zur nächsten Eingabedatei verwendet werden. Auch dies wurde von mir dahingehend erweitert, dass auch Eingabedateien mit variabler Zeitdifferenz verarbeitet werden können, die sogar linear in der Zeit interpoliert werden. Dies ermöglicht es, Ergebnisse von Strömungssimulationen mit adaptiver Zeitschrittweitensteuerung verwenden zu können. Siehe dazu auch Abschnitt 3.2.4.

3.2.1 Einstellung der Parameter

Auf die verwendete Texturen wirken im Wesentlichen zwei Prozesse ein: Transport und Diffusion. Der Transportprozess soll die Textur entlang eines Richtungsfeldes/Strömung möglichst perfekt bewegen, was in der numerischen Implementierung allerdings immer mit künstlicher Diffusion zur Stabilisierung verbunden ist, welche zu minimieren ist, als auch numerischer Diffusion die Teil der numerischen Verfahren selbst ist. Der Diffusionsprozess soll die Textur in kontrollierter Weise anisotrop verschmieren. Damit die Strukturen auch in Bereichen mit geringer Strömung gut dargestellt werden, sollte eine Mindestdiffusion vorgegeben werden. Dies ist indirekt durch die Angabe einer Mindestströmungsgeschwindigkeit $v_{min} > 0$ möglich. Da für den Fall $\|\vec{v}\| = 0$ keinerlei Richtung definiert ist, wird in diesem Fall ein isotroper Diffusionsoperator mit Diffusionsstärke $\alpha(v_{min})$ verwendet. Dieser Fall tritt zum Beispiel an Gebietsrändern mit Haftrandbedingung und im Zentrum eines Wirbels auf.
Sei nun der Zeitschritt der NATD-Gleichung mit Δt angegeben, und der Parameter $\beta > 0$, so sei, wie in [BPR01], die Funktion $\alpha : \mathbb{R} \to \mathbb{R}^+$ im Matrixoperator A (siehe Gleichung (1.10)) folgendermaßen definiert:

$$\alpha(\|\vec{v}(x,t)\|) := \frac{\beta^2 \max(\|\vec{v}(x,t)\|, v_{min})^2 \Delta t}{2}. \tag{3.1}$$

In den Beispielen wird $\beta = 10$ und $v_{min} = 0.05$ benutzt. Für $\|\vec{v}(x,t)\| = v_{min}$ liegt eine Unstetigkeit in der Ableitung von α vor. Dies ist zwar für praktische

Anwendungen ohne Probleme, für eventuelle theoretische Untersuchungen sollte aber diese Stelle geglättet werden oder die Funktion durch

$$\alpha(\|\vec{v}(x,t)\|) := \frac{\beta^2(\|\vec{v}(x,t)\|^2 + \gamma)\Delta t}{2}$$

ersetzt werden, mit $\gamma > 0$ sehr klein gewählt. Diese Variante von α ist in $C^\infty(\mathbb{R})$ und es existiert für festes Δt ein $\delta > 0$ mit $\alpha(s) \geq \delta > 0 \; \forall s \in \mathbb{R}$.
Wenn orthogonal zur Strömungsrichtung auch etwas Diffusion in Abhängigkeit des Texturgradienten hinzugefügt wird, können feinere parallel laufende Linien miteinander verschmelzen. Dabei wird bei stärkeren Gradienten diese Diffusion herabgesetzt. Dafür ist eine passend gewählte Funktion $g : \mathbb{R} \to \mathbb{R}^+$ zuständig, für die es verschiedene brauchbare Auswahlmöglichkeiten gibt, die durch verschiedene Ansätze in der Bildverarbeitung inspiriert sind:

Konstanter Wert: $g(s) := \epsilon$

Minimale Flächen: $g(s) := \frac{\epsilon}{1+c\cdot s^2}$. Siehe auch [PM87].[11]

Perona-Malik: $g(s) := \epsilon \cdot e^{-c\cdot s^2}$

Geman-McClure: $g(s) := \frac{\epsilon}{(1+c\cdot s^2)^2}$

Damit diese Linien aber nicht zu sehr durch Diffusion verschwimmen oder die Textur zu einem einheitlichen Grau zerfließt, ist eine zusätzliche allgemeine Verstärkung der Bildkontraste beziehungsweise der Bildgradienten wünschenswert. Dies alles kann mittels des in (2.4) beschriebenen Ansatzes mit der in (2.6) beschriebenen rechten Seite[12] erreicht werden. Die so erzeugten Strukturen werden damit zwar mit der Zeit grobskaliger, aber je nach Einstellung der Kontrastverstärkung ausreichend sichtbar oder sogar scharf abgegrenzte schwarz-weiße Linien.[13]
Der enthaltene Transportterm muss natürlich für eine numerische Implementierung des Verfahrens stabilisiert werden. Wenn Stromliniendiffusion zur Stabilisierung des Transportterms benutzt wird, tritt die zusätzliche numerische Diffusion in Hauptrichtung der gewünschten anisotropen Diffusion auf. Es bietet sich deshalb an, dies durch eine Addition der zusätzlich benötigten Diffusion zum Ergebnis der Funktion $\alpha(\cdot)$ zu Implementieren.
Aus theoretischen Gründen der Wohlgestelltheit ist es besser, dass der Diffusionsoperator A nicht direkt von den Gradienten der Textur I abhängt, sondern von den Gradienten einer regularisierten Textur I_σ. Siehe dazu auch [KK98, CLMC92]. In der praktischen Anwendung kann aber im Allgemeinen $\sigma = 0$ gesetzt werden, beziehungsweise die Regularisierung weggelassen werden.

[11] Standard von M. Rumpf und T. Preusser.

[12] Wenn nicht explizit anders angegeben, wird im Weiteren $\rho = 80$ verwendet.

[13] Eine gute Vorstellung von den Strukturen ergibt sich, wenn man sich "Schlierenmuster" in erhitztem Wasser innerhalb eines Wasserkochers mit transparenten Seitenwänden ansieht. Oder einfach einen handelsüblichen Teekocher mit Glaskanne im Betrieb von der Seite betrachtet. Die dort Betrachteten Muster werden durch einen variierenden Brechungsindex erzeugt und benötigen eine passende Beleuchtung, stellen aber sogar eine Möglichkeit zur Betrachtung der Struktur einer dreidimensionalen Konvektionsströmung dar!

3.2.2 Stabilisierung mittels Stromliniendiffusion (SD)

Wie schon zuvor in Abschnitt 3.2.1 angemerkt, ist der Diffusionsoperator schon in Strömungsrichtung und dazu orthogonal stehender Richtung aufgespaltet. Sei nun

$$Re_{loc} := \frac{\|\vec{v}\|_{loc}\, h_{loc}}{\alpha(\|\vec{v}\|)} \qquad sd := sd_{par}\, h_{loc} \frac{Re_{loc}}{1 + Re_{loc}}. \tag{3.2}$$

Dann lautet der stabilisierte Diffusionsoperator $\tilde{A}(\vec{v}, \nabla I)$ wie folgt:

$$\tilde{A}(\vec{v}, \nabla I) = B(v) \begin{pmatrix} \alpha(\|\vec{v}\|) + sd & 0 \\ 0 & g(\|\nabla I\|)\mathrm{Id}_{d-1} \end{pmatrix} B(v)^T. \tag{3.3}$$

Wobei in der Formulierung schon die Erweiterung auf Dimensionen $d > 2$ angedeutet wird.
Der vom Anwender zur wählende Parameter sd_{par} liegt im Intervall $(0; 2)$ und h_{loc} ist die lokale Gitterweite, die analog zu $\|\vec{v}\|_{loc}$ die lokale Geschwindigkeit bezeichnet (siehe [Tur99, Seite 119-122] für weitere Details). Die Vorteile in der Wahl dieser Stabilisierung liegt in der einfachen Implementierbarkeit auf unstrukturierten Gittern[14], der hohen Robustheit der Stabilisierung ohne allzu übermäßige Diffusion und darin, dass es eine *lineare* Stabilisierung ist, deren für jeden Zeitschritt zu lösenden Probleme, im Gegensatz zu TVD[15]-Methoden, effizient mit iterativen Lösern gelöst werden können.
Da es sich bei SD aber um eine lineare Stabilisierung handelt, gibt es eventuell auftretende numerische Oszillationen. Daher kann die richtige Wahl von sd_{par} kritisch für die Genauigkeit, Robustheit und Effizienz des numerischen Verfahrens sein. Falls auf die Linearität des Diffusionsoperators verzichtet werden kann, so könnte man versuchen $\alpha(\|\vec{v}\|) + sd$ durch $\max\{\alpha(\|\vec{v}\|), sd\}$ zu ersetzen. In diesem Fall können aber auch fortschrittlichere Techniken wie TVD- und FCT[16]-Methoden zur Stabilisierung gewählt werden.

3.2.3 Optionale zusätzliche Crosswind-Stabilisierung (CW)

Extrem anisotrope Diffusionsoperatoren können laut [JK07, JK08] ebenfalls eine Instabilität aufweisen und Oszillationen in der Lösung erzeugen. Dies kann durch eine zusätzliche "Crosswind"-Stabilisierung verhindert werden. Sein nun

$$cw := \max(0, \|\vec{v}\| h_{loc}^{\frac{3}{2}} - g(\|\nabla I\|)). \tag{3.4}$$

Damit ändert sich (3.3) zu

$$\tilde{A}(\vec{v}, \nabla I) = B(v) \begin{pmatrix} \alpha(\|\vec{v}\|) + sd & 0 \\ 0 & (g(\|\nabla I\|) + cw)\mathrm{Id}_{d-1} \end{pmatrix} B(v)^T. \tag{3.5}$$

[14] Insbesondere da der Diffusionsoperator sowieso schon passend separiert ist!
[15] "Total Variation Diminishing"
[16] "Flux Corrected Transport"

Bei eigenen Versuchen, mit den in dieser Arbeit verwendeten Funktionen $g(\cdot)$, war allerdings kein sichtbarer Unterschied zu bemerken, so dass diese zusätzliche Stabilisierungsmethode zwar erwähnt, im Weiteren aber nicht mehr verwendet wurde. Bei extremen Funktionen $g(\cdot)$ könnte die CW-Stabilisierung aber notwendig werden, weshalb diese Methode hier trotzdem erwähnt wird.

3.2.4 Angabe des Strömungsfelds

Noch nicht erwähnt wurde, wie das zu visualisierende Strömungsfeld angegeben wird. Dafür sind im Anwendungsbereich drei verschiedene Möglichkeiten zu finden:

1. **Rein analytisch:** Dabei wird das Strömungsfeld als Funktion $\vec{v}(x,t)$ durch eine Formel ausgedrückt. Dies findet entweder vor Start des Programms direkt im Programmcode statt oder später, falls ein Laufzeitinterpreter vorhanden ist, der Benutzereingaben in ausführbare Funktionen übersetzt. Dies ist sehr geeignet zur Visualisierung von Testbeispielen, für welche analytische Lösungen vorliegen.

2. **Synchron zur Berechnung:** Dabei wird die Visualisierung nach jedem Zeitschritt der Berechnung um denselben Zeitraum weitergeführt. Dabei drängt sich eine Parallelisierung der Berechnungs- und Visualisierungsprozesse auf, bei denen der erste Prozess seine Ergebnisse an den zweiten Prozess weiterreicht[17] und während der Visualisierung die Simulationsrechnung weiterführt.

   Nachteile dieser Methode sind, dass bei Änderungen der Visualisierungseinstellungen, die ganze Rechnung wiederholt werden muss, und die Zeitschritte zur Strömungsberechnung oft sehr viel kleiner sind, als die Zeitschritte, die für eine ausreichend genaue Visualisierung nötig sind.

   Die Vorteile liegen in der höheren Genauigkeit der Datensätze und dem geringen Bedarf an Speicherplatz für eine ansonsten nötige Zwischenspeicherung von Strömungsdaten.

3. **Asynchron zur Berechnung:** Hierbei wird in einem ersten Schritt die Strömung simuliert und die resultierenden Strömungsinformationen entweder in regelmäßigen oder adaptiv gewählten Zeitschritten in Form von Dateien mit enthaltenen Zeitstempeln gespeichert. Die eigentliche Simulation erfolgt erst nachträglich, wobei die einzelnen Dateien im Laufe der Visualisierung einzeln eingelesen und in der Zeit interpoliert werden.

   Die Nachteile liegen im hohen Speicherverbrauch und einer Verringerung der Genauigkeit der Strömungsinformation.

   Der Vorteil liegt in der Entkoppelung der Strömungsberechnung und der Visualisierung, die mit unterschiedlichen Parametern wiederholt werden kann, ohne die Strömungsdaten neu zu berechnen.

[17] Mittels einer Interprozesskommunikation, zum Beispiel MPI.

In den im Umfang dieser Arbeit verwendeten Programmen wird zwar die analytische Vorgabe in einzelnen Testproblemen verwendet, Hauptziel war aber die Verwendung von vorberechneten Datensätzen mit, nicht notwendig gleichbleibender, Zeitdifferenz zwischen den Dateien. Dies ist notwendig, wenn man die Ausgabe von Simulationsrechnungen mit adaptiver Zeitschrittweitensteuerung visualisieren will. Die Ausgabe der Visualisierung soll dabei in gleichbleibenden Zeitintervallen erfolgen, damit später brauchbare Animationen erstellt werden können.

Damit das Ergebnis der Visualisierung eine sinnvolle Darstellung der Strömung ergibt, muss aber zugleich die Synchronizität der Simulation und Visualisierung gewährleistet werden. Oder in anderen Worten, dass Zeitschritte in der Simulation und der Visualisierung dasselbe bedeuten. Dafür ist bei Datensätzen mit wechselnder Zeitdifferenz zwischen den Datensätzen der schon erwähnte Zeitstempel nötig. In den hier verwendeten Programmen liegen die Strömungsdaten im GMV-Format[18] vor, welches diesen Zeitstempel im Eintrag "probtime" bereitstellen kann.

Beides wird durch eine eigene Implementierung einer linearen Interpolation in der Zeit erfüllt, welche selbständig die passenden Eingabedateien findet, die Zeitstempel extrahiert und für eine gegebene Zeit genau die dazu passenden Dateien für eine Interpolation einließt. Nebenbei wird unnötiges Neueinlesen von Datensätzen vermieden.

Eine genauere Beschreibung des dafür benutzten Algorithmus besteht darin, dass zwei Puffer verwendet werden, die abwechselnd mit den eingelesenen Strömungsfeldern befüllt werden, bis die Zeitstempel des einen Puffer kleiner gleich und des anderen Puffers größer als die gesuchte Zielzeit sind. Für nachfolgende Anfragen wird zuerst überprüft, ob sich die gesuchte Zeit immer noch zwischen den Zeitstempeln der Puffer befindet. Falls ja, werden die Daten einfach linear interpoliert. Trifft dies aber nicht zu, werden erneut Strömungsdaten aus Dateien eingelesen, bis sich die gesuchte Zielzeit wieder zwischen den Zeitstempeln der Puffer befindet.

Bei diesem Algorithmus wird vereinfachend angenommen, dass die gesuchten Zielzeiten in der Abfolge der Eingabedateien eine streng monoton wachsende Folge bilden, was in den benutzten Programmen der Fall ist. Falls eine Interpolation höherer Ordnung gewünscht wird, so müsste man den Algorithmus um zusätzliche Puffer erweitern und die Interpolationsformel anpassen, die benutzt wird, um aus den Puffern die Daten der gesuchten Zielzeit zu interpolieren. Das Grundgerüst des Algorithmus sollte sich aber, bis auf die Startphase und Endphase[19], nicht wesentlich ändern.

3.2.5 Überblendungsmethode

Die Strukturen, die sich durch die Anwendung der texturbasierten Visualisierungstechnik ergeben, werden durch die angewendete Diffusion mit fortschreitender Simulationszeit immer grobskaliger. Bei Verwendung der rechten Seite $f = 0$,

[18] **G**eneral **M**esh **V**iewer: `http://www.generalmeshviewer.com`

[19] Bereiche um die zeitlichen Intervallgrenzen.

tendiert das Resultat von Gleichung (2.4) für $t \to \infty$ sogar zum Mittelwert.[20] Dies hat als Nebeneffekt, dass kleinskalige Strukturen, wie zum Beispiel Wirbel in Gebietsecken, nicht mehr dargestellt werden. Es wäre sehr gut, wenn sich die Darstellungsskala irgendwie um einen mittleren Skalenbereich herum stabilisieren ließe. Um eine einheitliche Darstellungsskala für das Gesamtgebiet zu erhalten, ist eine zumindest annähernd gleichmäßige Verteilung der Gitterpunkte zu beachten, da ansonsten die ungleichen Skalen sehr auffällige und unschöne visuelle Artefakte im Endresultat ergeben. Zudem können Zonen mit groben Gitterstrukturen, die stromaufwärts liegen, verhindern, dass stromabwärts liegende feine Gitterstrukturen auch feine Skalen darstellen können. In anderen Worten: Sobald eine Textur in ein Gebiet mit grobem Gitter transportiert wird, gehen alle feinen Strukturen verloren und werden auch nicht durch einen Weitertransport in eine Zone mit feinem Gitter wieder rekonstruiert!

Um eine Stabilisierung der Skalen entlang der Zeitachse zumindest zu approximieren, kann man verschiedene Lösungen unterschiedlicher Skalen zu einer einzigen Lösung verbinden. Dabei sollen die Lösungen im mittleren Skalenbereich einen stärkeren Anteil besitzen, als klein- und grobskalige Lösungen. Dies wird durch eine gewichtete Überblendung von Lösungen erreicht, die zu unterschiedlichen Zeiten gestartet wurden, und somit unterschiedliche Skalen repräsentieren. Dabei sind unterschiedliche Überblendungsmethoden möglich: trigonometrische Funktionen, interpolierende Splines und noch viele weitere, die hier aber nicht weiter ausgeführt werden. In der hier vorgestellten Methode werden Bézier-Splines zusammen mit einer speziellen, von mir zur Reduktion von Unebenheiten am Anfang erdachten, Startphase verwendet.

Innerhalb dieser Startphase wird die Überblendungsmethode von einer Lösung bis zur vollständigen Anzahl n_{tot} von Lösungen aufgebaut. Dabei lassen sich zwar Unebenheiten in der visuellen Qualität nicht vermeiden, diese sollte aber durch die hier angegebene Methode wesentlich reduziert werden. Die momentan verwendete Anzahl von Lösungen sei n_{akt} und die Zeit zwischen Wechseln der Lösungen oder (Re-)Initialisierungen sei Δt_{blend}. Die zur Überblendung verwendeten Lösungen werden in einer Array-Struktur verwaltet. Nachdem eine Zeitdifferenz von Δt_{blend} vergangen ist, wird in der Startphase eine neu mit Rauschen initialisierte Lösung an den Anfang des Array eingefügt und alle bisher vorhandenen Lösungen um eine Indexposition höher verschoben.[21] Sobald $n_{akt} = n_{tot}$, beziehungsweise das Array voll ist, wird die älteste Lösung durch eine neue Rauschlösung ersetzt oder überschrieben und der Inhalt des Arrays im Kreis verschoben[22], so dass die zweitälteste Lösung zur neuen ältesten Lösung wird.

Die eigentliche Überblendung ist in zwei Teile aufgespaltet. Der erste Teil besteht aus der Funktion *ParCalc*, welche die aktuelle Simulationszeit t in das Intervall $[0, 1]$ abbildet und dem zweiten Teil, der aus der Funktion *BlendCoeffCalc* besteht,

[20] Bei Problemen ohne Ein- und Ausströmung. Ansonsten gibt um den Einströmbereich herum, durch die Verwendung von wechselnden Zufallswerten, immer einen kleinskaligen Bereich.

[21] Aus Effizienzgründen befinden sich natürlich nur Verweise/Zeiger auf die Lösungen in diesem Array!

[22] In der englischsprachigen Literatur als “ring shift” bekannt.

welche diesen abgebildeten Wert benutzt, um ein Array mit Gewichten zu errechnen. Diese Gewichte werden durch die Auswertung der n_{tot} Berstein-Polynome des Grades $n_{tot} - 1$ berechnet, und dienen als Gewichtung bei der Summation der Einzellösungen in eine Gesamtlösung.
Das Spezielle an der vorgestellten Methode liegt in der Funktion *ParCalc*, die wie folgt definiert ist, wobei "$flag$" eine boolesche Variable bezeichnet, die anfangs auf "falsch" gesetzt ist, aber beim Erreichen der gewünschten Anzahl von Lösungen auf "wahr" gesetzt wird, Δt_{blend} das Zeitintervall bis zu einer Erneuerung einer Lösung und t die Zeitdifferenz zur Startzeit:

$$ParCalc(t, \Delta t_{blend}, n_{akt}, n_{tot}, flag) := \begin{cases} \frac{\text{mod } (t, \Delta t_{blend})}{\Delta t_{blend}\, n_{tot}} + \frac{n_{tot}-1}{2\, n_{tot}} & \text{wenn } flag\text{=“wahr”} \\ \frac{t}{\Delta t_{blend}\, n_{tot}\, n_{akt}} + \frac{n_{akt}-1}{2\, n_{akt}} & \text{wenn } flag\text{=“falsch”} \end{cases}$$

Die hier hier vorgestellte konkrete Implementierung einer Überblendungsmethode ist zwar eine unabhängige Eigenentwicklung, es ist aber möglich, dass jemand anderes bei einer ähnlichen Problemstellung auf dieselbe Idee gekommen ist.

3.3 2D-Beispiele

In diesem Kapitel wird die texturbasierte NATD-Visualisierungstechnik an verschiedenen Strömungsbeispielen vorgeführt. Nach einer kurzen Vorstellung der verwendeten numerischen Methoden, wird neben Standardbeispielen für Strömungssimulationen (siehe Abschnitte 3.3.3 und 3.3.4) auch die komplexere Strömungsstruktur einer Venturipumpe untersucht (siehe Abschnitt 3.3.5).

3.3.1 Verwendete numerische Methoden

Die Strömungen selbst wurden mittels der Programme[23] aus dem FeatFlow[24]-Numerikpaket berechnet. Alle verwendeten Programme benutzten hierarchisch erzeugte Viereckgitter mit einer $\tilde{Q}_1/P_0$[25] FEM-Diskretisierung im Ort. Die Diskretisierung in der Zeit erfolgt über ein Fractional-Step-θ-Schema. Als Löser kommt das Mehrgitterverfahren mit Vanka-Glätter zur Anwendung. Die errechneten Druckwerte werden nicht weiter benötigt und später ignoriert. Trotzdem werden diese voreingestellt in die Ecken des Gitters interpoliert abgespeichert. Die Freiheitsgrade der Geschwindigkeit liegen in den Kantenmittelpunkten und werden zur Ausgabe in GMV-Dateien ebenfalls in die Ecken der Elemente interpoliert. Jede Ausgabedatei enthält zudem einen Zeitstempel im Datenfeld "probtime", welcher die Synchronizität der berechneten Strömung mit der Visualisierung ermöglicht.

[23] pp2d/cc2d/bouss
[24] Version 1.3
[25] $\tilde{Q}_1$: rotiert bilineares (Rannacher-Turek-)Element.

Im Grunde ist die Quelle dieser Daten aber egal, sofern sie in einem Format vorliegen, welche das nachfolgende Programm korrekt interpretieren kann. Sofern ein passender Ausgabefilter verwendet/geschrieben wird, könnten auch andere Numerikpakete als Datenquellen in Frage kommen.

Mit den vorberechneten Ausgabedaten arbeitet das Programm zur Berechnung der Visualisierungen und erzeugt wiederum neue GMV-Ausgabedateien, welche von konventionellen Visualisierungsprogrammen (GMV, ParaView[26]) betrachtet werden können. Im Gegensatz zu den Programmen, welche die Strömungsdaten berechnet haben, arbeitet dieses mit den Hilfsroutinen des Feast-Numerikpakets als Basis. Obwohl dasselbe Berechnungsgitter verwendet wird, werden nun aber Q_1-Elemente zur FEM-Diskretisierung im Ort verwendet. Da die zu visualisierenden Strömungsdaten in interpolierter Form vorliegen (Seitenmitten $\rightarrow$ Knoten), passt die Diskretisierung auch gut mit diesen Daten zusammen und erfordert keine weitere Interpolation, welche nur weitere Fehler verursachen würde.

Wie schon erwähnt wurde zur Zeitdiskretisierung ein variables "one-step-θ"-Schema implementiert, welches Einstellungen zwischen explizitem Euler ($\theta = 0$), Crank-Nicholson ($\theta = \frac{1}{2}$) und implizitem Euler ($\theta = 1$) erlaubt. In den angeführten Beispielen wurde allerdings nur der implizite Euler verwendet. Obwohl damit zusätzliche Diffusion eingeführt wird, können damit größere Zeitschrittweiten benutzt werden und die Berechnungen der Visualisierung beschleunigt werden. Die errechneten Lösungen sind für reine Visualisierungszwecke ausreichend genau. Zusammen mit der höheren Stabilität des Verfahrens kann eine adaptive Zeitschrittweitensteuerung für jede Einzellösung vermieden werden, was die Komplexität des Programms reduziert. Falls eine höhere Genauigkeit oder eine bessere Kontrolle über diese gewünscht wird, so sollte eine adaptive Zeitschrittweitensteuerung getrennt für jede Einzellösung und Crank-Nicholson, beziehungsweise ein anderes Zeitschrittverfahren mit einer Ordnung höher als 1, wie zum Beispiel Runge-Kutta-Verfahren, verwendet werden.

Um eine, für die Erzeugung einer Animation nützliche, Ausgabe zu erhalten, werden die Ergebnisse der Visualisierungsberechnungen mit konstanter Zeitdifferenz erzeugt, selbst wenn die ursprünglichen Strömungsdaten mit einem adaptivem Zeitschrittverfahren berechnet und mit variierender Zeitdifferenz erzeugt wurden.

3.3.2 Vergleich verschiedener Löser

Ein anisotroper Diffusionsoperator ist bekanntermaßen äquivalent zu einem isotropen Diffusionsoperator auf einem anisotropen Rechengitter. Dies kann auch lokal als eine auf die Gitterknoten wirkende Koordinatenabbildung betrachtet werden. Daher ist anzunehmen, dass Löser, die auch für sehr anisotrope Gitter geeignet sind, sich auch für anisotrope Diffusionsoperatoren eignen. Bezüglich dieses Themas wird hier auf [Kös04] verwiesen.

Die variationelle Formulierung des NATD-Problems und dessen FEM-Diskretisie-

[26] `http://www.paraview.org`

Stufe	Zellen	Knoten
1	20	34
2	80	107
3	320	373
4	1280	1385
5	5120	5329
6	20480	20897
7	81920	82753

Tabelle 3.3.1: Liste der Knoten- und Zellzahlen für verschiedene Gitterverfeinerungsstufen.

rung (impliziter Euler) erzeugt lineare Unterprobleme der Form

$$(M + \Delta t\ S)x_{n+1} = M\ x_n + \Delta t\ M\ f(x_n) \tag{3.6}$$

mit der Massematrix M und der asymmetrischen Steifigkeitsmatrix S.
Der Transportoperator zerstört die Symmetrie des NATD-Operators, ist der Grund für die Asymmetrie der Steifigkeitsmatrix und bewirkt, dass keine Löser eingesetzt werden können, welche symmetrische positiv definite Operatoren benötigen. Das bedeutet CG-Löser und Glätter basierend auf Cholesky-Zerlegungen können nicht verwendet werden. Direkte Löser sind wegen der großen Anzahl von Unbekannten und der durch die eventuell hohen Anisotropie des Diffusionsoperator erwartbaren Konditionszahlen auch unpassend.
Deshalb wurden hier für dieses Problem nur Iterative Löser beziehungsweise Mehrgitterverfahren mit unterschiedlichen Glättern am Beispiel der Venturipumpe (siehe Abschnitt 3.3.5) getestet. Zu erwarten ist, dass sich für diese Art von Problem Methoden basierend auf Krylov-Räumen sehr gut eignen werden. Siehe dazu auch [van92]. Da hier PDE-Probleme 2. Ordnung gelöst werden, ist die Konditionierung der Probleme stark von der Gitterweite abhängig ($O(h^{-2})$). Da auch die Anisotropie im Diffusionsoperator sehr groß werden kann, werden die meisten Iterationsverfahren Konvergenzprobleme oder zumindest schlechte Konvergenzraten aufweisen. Dieses Problem kann durch verschiedene Vorkonditionierer abgemildert werden. Insbesondere sind mutmaßlich hierarchische Mehrgitterverfahren, deren Konvergenzraten weniger abhängig von der Konditionierung von S für verschiedene Gitterweiten sind, für diesen Zweck geeignet.
FEAST erlaubt eine Vielzahl von Lösern zu neuen Lösern zu kombinieren. Es wurden etliche verschiedene Löserkombinationen getestet und drei Repräsentanten ausgewählt. Die damit jeweils erreichten Konvergenzraten wurden tabellarisch erfasst (siehe Tabelle 3.3.2).
Wie aus Tabelle 3.3.2 zu entnehmen ist, kann BiCG-stab mit einem Mehrgitterlöser (MG mit ADITRIGS[27] als Glätter) als Vorkonditionierer die Stärken der jeweiligen Verfahren kombinieren und liefert sehr gute Konvergenzraten. Insbesondere bei Verfeinerungsstufe 7 ergaben sich Konvergenzraten, die um den Faktor

[27] **A**lternat**i**ng **tri**-diagonal **G**auß-**S**eidel.

Stufe	$\Delta t = 0.001$	#It.	$\Delta t = 0.002$	#It.	$\Delta t = 0.003$	#It.
5	0.08753	6	0.08345	6	0.12957	7
6	0.08973	6	0.11818	7	0.19461	9
7	0.20600	9	0.37173	15	0.50550	21

Stufe	$\Delta t = 0.001$	#It.	$\Delta t = 0.002$	#It.	$\Delta t = 0.003$	#It.
5	0.00001	1	0.00008	2	0.00030	2
6	0.00003	2	0.00043	2	0.00985	3
7	0.00027	2	0.06115	5	0.60475	28

Stufe	$\Delta t = 0.001$	#It.	$\Delta t = 0.002$	#It.	$\Delta t = 0.003$	#It.
5	0.00001	1	0.00001	2	0.00008	2
6	0.00001	1	0.00059	2	0.00275	3
7	0.00001	1	0.00641	3	0.05587	5

Tabelle 3.3.2: Vergleich der Konvergenzraten für verschiedene Verfeinerungsstufen, Zeitschrittweiten und Löser. Mit ADITRIGS vorkonditionierter BiCG-stab (Oben), MG mit ADITRIGS Glätter, 8 Glättungsschritten und F-Zyklus (Mitte), mit MG/ADITRIGS vorkonditionierter BICG-stab (Unten). Das Rauschfeld wurde durch Sinuswellen ersetzt um die Ergebnisse vergleichbarer zu machen, und es mussten jeweils sechs Stellen an Genauigkeit gewonnen werden.

10 besser waren, als bei dem zweitbesten Kandidaten (MG mit ADITRIGS als Glätter).
Die Ursache für die starke Verbesserung der Konvergenzrate des BiCG-stab-Lösers, beim Wechsel von ADITRIGS zu MG Vorkonditionierung, liegt vermutlich in der Eigenschaft von Mehrgitterverfahren, hohe Frequenzanteile im Fehler stark zu dämpfen, was den Hauptteil des Fehlers im niederfrequenten Anteil vereint. Wenn man dies auf einem hierarchisch generierten Tensorproduktgitter mit festen Gitterweiten betrachten würde, so könnte man eine DFT[28] auf dem Fehler durchführen. Die möglichen Frequenzwerte auf dem Grobgitter der Hierarchie ist um den Faktor 2^{1-l} geringer als auf dem verfeinerten Gitter, wenn l die Anzahl der Gitterebenen bezeichnet. Deshalb kann das Krylovraumverfahren (BiCG-stab) sich darauf konzentrieren, hauptsächlich die wenigen so konzentrierten Fehlerfrequenzen zu entfernen.
Obwohl FEAST eine große Bandbreite an Möglichkeiten zulässt, sich Löser nach eigenen Vorstellungen zusammenzusetzen, sind nicht alle Möglichkeiten sinnvoll. Innerhalb eines Schrittes des vorkonditionierten BiCG-stab-Algorithmus wird der Vorkonditionierer mehrfach aufgerufen. Es wird davon ausgegangen, dass der Vorkonditionierer zumindest innerhalb eines BiCG-stab-Schrittes konstant bleibt. Daher wird eine feste Zahl von Mehrgitter und Glättungsschritten vorgegeben (1*MG, 8*ADITRIGS), obwohl FEAST auch eine dynamische Wahl über den exakten oder relativen Fehler[29] zuließe.

[28] **D**iskrete **F**ourier **T**ransformation.

[29] Mit "Fehler" ist hier die Größe des Defekts, beziehungsweise des Fehlers in der rechten Seite des zu lösenden linearen Gleichungssystems gemeint.

Die hier verglichenen Löser sind nicht extra für Parallelverarbeitung mittels Gebietszerlegung[30] optimiert worden. FEAST erlaubt die Zerlegung des Rechengebietes in Einzelbereiche (Paralleblöcke) und die Löser können auf verschiedenen Ebenen arbeiten: Global, Parallelblock und Matrixblock. Dabei korrelieren Matrixblöcke zu lokalen Tensorproduktgittern, die aus einzelnen Zellen des Grobgitters erzeugt wurden. Diese lokalen regulären Matrixstrukturen können benutzt werden um sowohl alle Matrix-Vektor-Operationen, als auch Prolongations- und Restriktions-Operationen für Mehrgitterverfahren, sowie auch schnelle und robuste Glätter zu konstruieren. Diese Strukturen ermöglichen es auch, den "Cache"-Speicher von Prozessoren und "Pipelining"-Effekte auszunutzen. Siehe dazu auch [TBK99].
Es wurden zusätzliche Tests mit Löservarianten durchgeführt, die das Mehrgitterverfahren mit einem weiteren Mehrgitterverfahren auf Parallelblock- oder sogar Matrixblockebene als Glätter kombiniert haben. Da die Konvergenzraten dieser Ergebnisse aber nicht wesentlich besser waren, werden diese im Vergleich nicht angeführt.
Selbsterstellte Kombinationen von Lösern können als Textdatei eingelesen werden. Hier ist die Beschreibung[31] des BiCG-stab Lösers mit MG-Vorkonditionierung und ADITRIGS-Glätter innerhalb des Mehrgitterverfahrens, der auch in den folgenden Beispielen verwendet wurde:

```
SOLVER=BICG, MAXITER=10, TOL=REL:1e-6
# preconditioned indirectly, globally, relative error <10^(-6)
PREC=INDIRECT_ON_SD, DAMP=1.0
  # by data-parallel multigrid, perform exactly one iteration
  SOLVER=MG, MAXITER=1, TOL=IGNORE, CYCLE=F:8:0, CGCDAMP=ADAP
    # smoothed by a per-macro multigrid
    SMOOTHER=ADITRIGS, DAMP=1.0
    COARSE=UMFPACK
```

[30] In englischen Veröffentlichungen bekannt als "domain decomposition".

[31] Die Syntax wird in der FEAST-Dokumentation beschrieben. Informationen zu FEAST sind bei `http://www.feast.tu-dortmund.de/index.html` zu finden.

3.3.3 Wirbelströmung hinter einem Zylinder

Bei diesem Beispiel handelt es sich um einen schon klassischen Testfall für Strömungssimulationen. Aus Gründen der Vergleichbarkeit wurde das SB25-Gitter (siehe Abbildung 3.3.1) aus einer älteren Version des "Album of Fluid Motion" des Lehrstuhls III der mathematischen Fakultät der TU-Dortmund entnommen. Da hier, wie auch in den folgenden Beispielen, dimensionslos gerechnet wurde, sind die Angaben nicht in SI-Einheiten, wie Meter oder Meter/Sekunde, sondern in *abstrakten Einheiten*.
Das Rechengebiet ist 2.9 Einheiten lang, 0.62 Einheit hoch und besitzt ein kreisförmiges Loch mit Radius 0.08, dessen Mittelpunkt bei $(0.52, 0.28)$ leicht asymmetrisch liegt, um die Bildung der, für dieses Beispiel charakteristischen, Wirbelkette zu beschleunigen. Eine Kopie der Gitterbeschreibung im FeatFlow-Format ist in Anhang C.2 zu finden. Die Einströmung besitzt ein parabolisches Profil mit einer maximalen Einströmung von 1 Geschwindigkeitseinheit, wodurch sich ein mittlerer Fluss von 2/3 Volumeneinheiten pro Zeiteinheit ergibt. Die für die Strömungsrechnung verwendete Viskosität war 10^{-3}. Zur numerischen Berechnung der Strömung wurde das FeatFlow-Programm CC2D benutzt und die Strömungsdaten wurden in Zeitintervallen von 0.03 Zeiteinheiten abgespeichert. Die Rechnung wurde auf Verfeinerungsstufe 6 mit 26016 Knoten und 25600 Zellen durchgeführt.

Abbildung 3.3.1: Grobgitter für die numerische Rechnung.

Unter Verwendung desselben Grobgitters als Makrogitter für mein Feast-Programm, wurden Visualisierungen mit den folgenden gemeinsamen Parametern gerechnet: 5 Verfeinerungen der Makros (ergibt genauso viele Knoten und Zellen wie bei der Rechnung mit CC2D), 4 überblendete Einzellösungen, neue Lösungen alle 41 Zeitschritte, Zeitschrittverfahren mit implizitem Euler, $\Delta t = 0.01$, $v_{min} = 0.01$, $\beta = 1.5$, $\epsilon = 0.01$, $c = 300$, $\rho = 3.0$ und $sd = 0.8$. Die Parameter beziehen sich dabei teilweise auf die Formeln für die tangentiale Diffusion, wie in (3.1) beschrieben, und die normal dazu stehende Diffusion, beschrieben durch die von Minimalflächen inspirierten Form für $g(\cdot)$ (siehe Seite 74). Das Ergebnis dieser Rechnung ist in der Abbildungsserie 3.3.2 zu sehen.
Hinter dem Zylinder entsteht ein Unterdruck, welcher die Flüssigkeit zuerst in zwei gegenläufigen Wirbeln zurücksaugt. Das angesaugte Material wird dann durch wechselseitige Wirbelablösung wieder abgesondert. Die Gesammtströmung erhält durch diese wechselseitige Aktivität eine Wellenform mit eingelagerten Wirbeln in den Bergen und Tälern der Wellen. Sie wird auch als Kármánsche Wirbelstraße bezeichnet und ist benannt nach Theodore von Kármán (1881-1963), der sie 1911 als erster nachgewiesen und berechnet hat. Die Geschichte und Eigenschaften dieser Wirbelstraßen wird etwas ausführlicher auch in [ENC] erläutert.

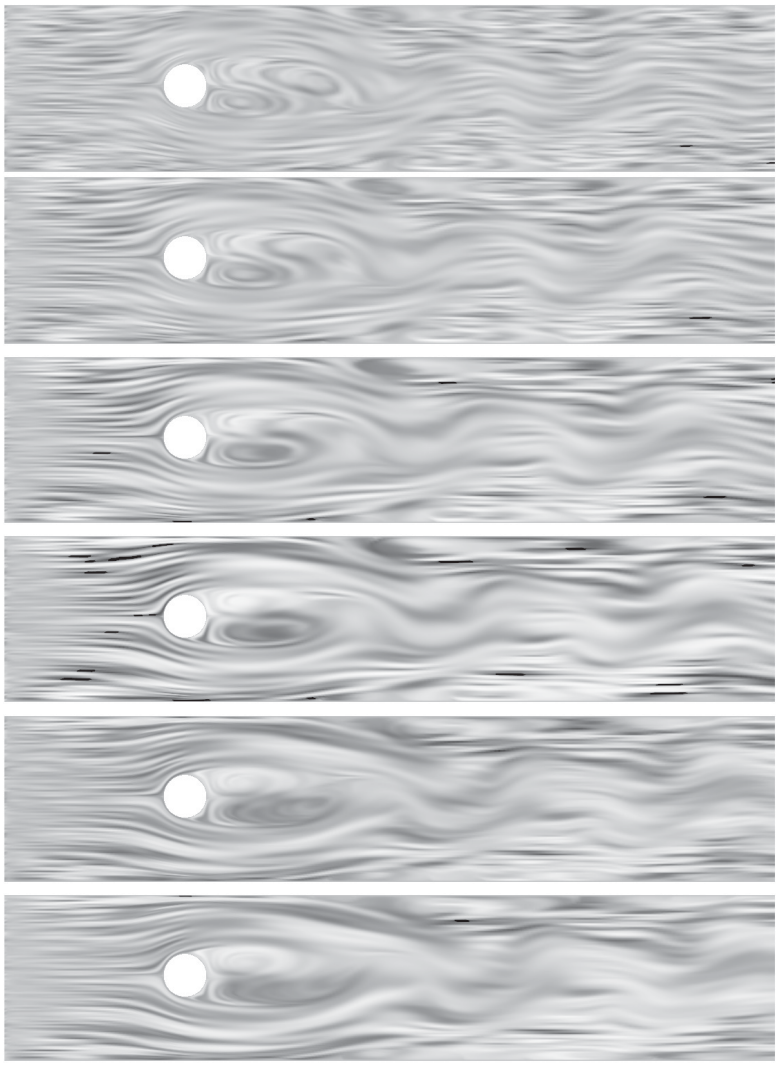


Abbildung 3.3.2: Visualisierung der Wirbelablösung hinter einem Zylinder.

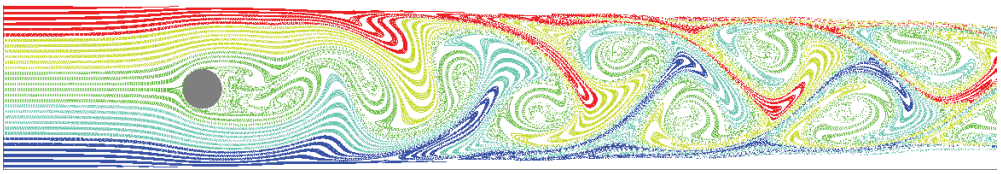


Abbildung 3.3.3: Partikelverfolgung mit Partikelquellen entlang des Einströmbereiches. Den Quellen wurden fünf verschiedene Farben zugewiesen, was die Faltung der zuerst laminaren Strömung durch die Wirbel illustriert.

Die so entstehenden Wirbelschleppen lassen sich auch gut mit alternativen Visualisierungsmethoden darstellen. In den Abbildungen 3.3.3 und 3.3.4 wurde ein ähnliches Problem, in der ersten Abbildung mittels Partikelverfolgung und in der zweiten Abbildung mittels parallel zur Strömungsberechnung ausgeführter Kon-

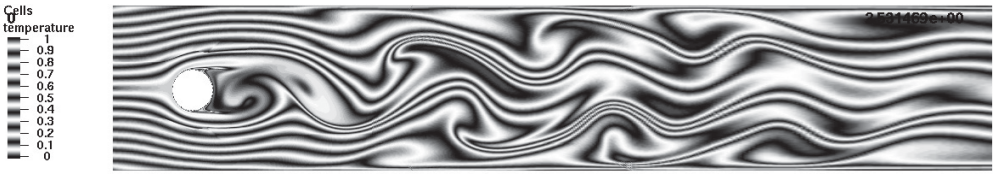


Abbildung 3.3.4: Ergebnis einer Berechnung mit BOUSS2D. Die Konzentration verläuft linear von 0 zu 1 entlang der Einströmungskante. Die speziell gewählte Farbpalette erzeugt eine Darstellung ähnlich einem Interferenzmuster.

zentrationsberechnung (FeatFlow-Programm BOUSS2D) und geschickt gewählter Farbpalette, visualisiert. Die Partikelverfolgung bietet für dieses Beispiel eine gute Alternative, was aber zum Teil auch daran liegt, dass die zu visualisierenden Strömungsstrukturen und gute Positionen[32] für Partikelquellen bekannt sind.

3.3.4 Stufenströmung

In diesem Beispiel wurde die Strömung über eine rückwärtsgerichtete Stufe untersucht. Das Rechengebiet ist 10 Einheiten Lang, 1 Einheit hoch und besitzt eine Stufe der Höhe 0.5, genau 2 Einheiten vom links liegenden Einströmrand entfernt. Die Einströmung besitzt ein parabolisches Profil mit einer maximalen Einströmung von 1 Geschwindigkeitseinheit, wodurch sich ein mittlerer Fluss von 2/3 Volumeneinheiten pro Zeiteinheit ergibt. Die für die Strömungsrechnung verwendete Viskosität war 10^{-4}. Zur numerischen Berechnung der Strömung wurde das FeatFlow-Programm BOUSS benutzt und die Strömungsdaten wurden in Zeitintervallen von 0.2 Zeiteinheiten abgespeichert. Das Grobgitter für diese Rechnung ist in Anhang C.3 zu finden und auch in Abbildung 3.3.5 zu sehen. Die Rechnung wurde auf Verfeinerungsstufe 7 mit 90945 Knoten und 90112 Zellen durchgeführt.

Abbildung 3.3.5: Grobgitter für die numerische Rechnung.

Unter Verwendung desselben Grobgitters als Makrogitter für das von mir für Feast geschriebene Programm zur Berechnung der Visualisierungsdaten, wurde mit den folgenden Parametern gerechnet: 6 Verfeinerungen der Makros (ergibt genauso viele Knoten und Zellen wie bei der Rechnung mit BOUSS), 4 überblendete Einzellösungen, neue Lösungen alle 8 Zeitschritte, Zeitschrittverfahren mit implizitem Euler, $\Delta t = 0.025$, $v_{min} = 0.005$, $\beta = 10.0$, $\epsilon = 0.01$, $c = 150$, $\rho = 6.0$ und $sd = 1.0$. Das Ergebnis dieser Rechnung ist in der Abbildungsserie 3.3.6 zu sehen und zeigt sehr komplexe zeitveränderliche Wirbelstrukturen.

[32]Eine weitere gute Position ist im direkten "Windschatten" des Kreises/Cylinders. Dort positionierten Partikel werden von der dortigen Unterdruckzone angesaugt und dann auf die Wirbelschleppen verteilt.

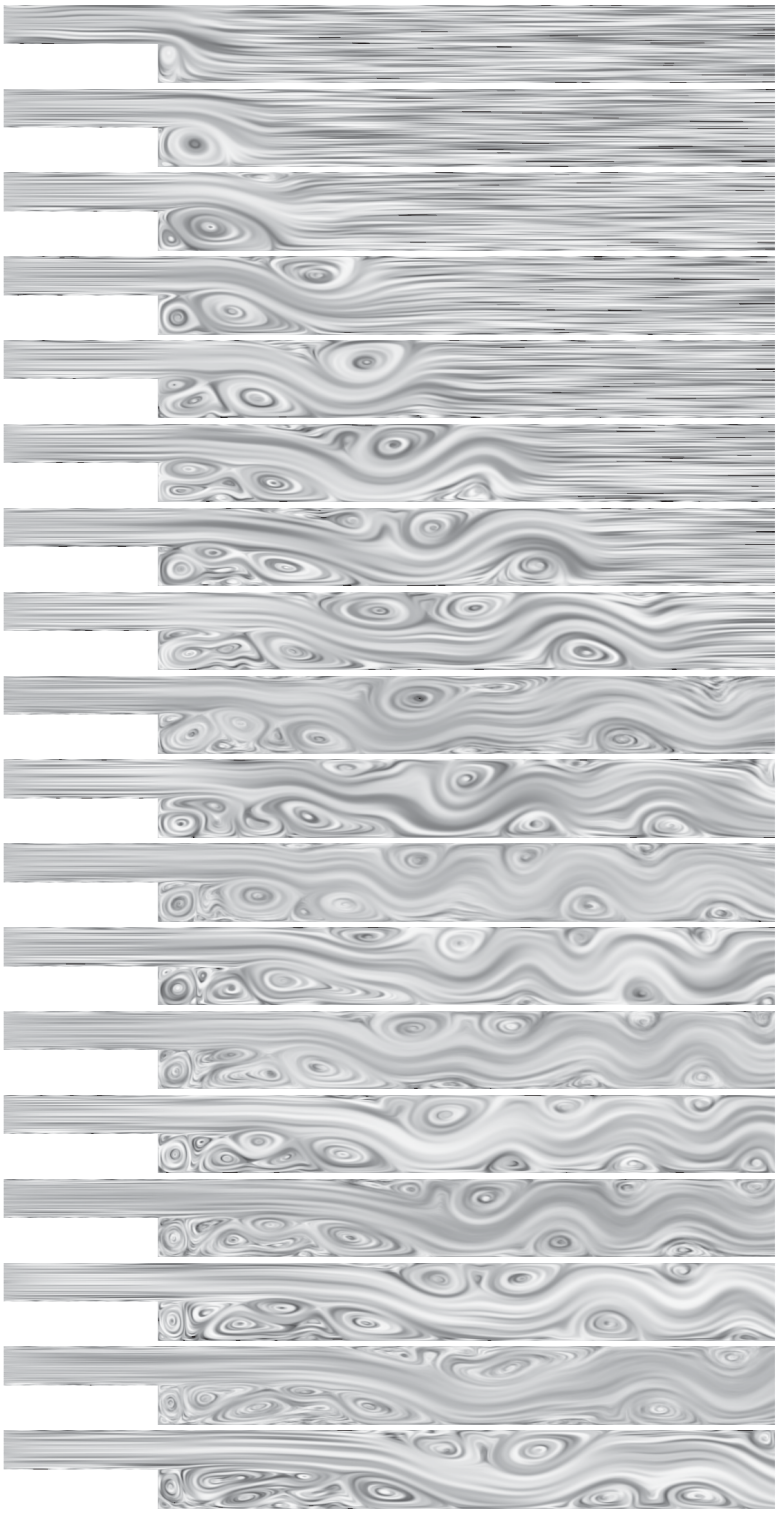


Abbildung 3.3.6: Visualisierung der Strömung über eine rückwärtsgerichtete Stufe.

3.3.5 Venturipumpe

Diese besondere Art von Pumpe/Düse[33] wurde von Giovanni Battista Venturi[34] erfunden und beruht auf der Erzeugung eines Unterdrucks innerhalb einer, in der Mitte verengten, Röhre durch eine schnelle Strömung (*Bernoulli-Prinzip*). Die Venturipumpe kann zum Beispiel zum Auspumpen von Segelbooten durch ihre Eigenbewegung benutzt werden. Eine Skizze des schematischen Aufbaus befindet sich in Abbildung 3.3.7.

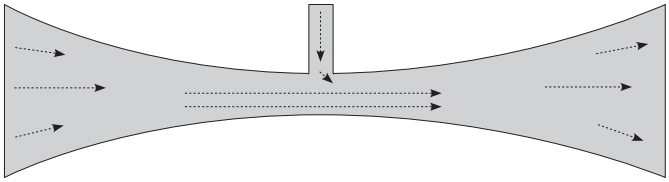
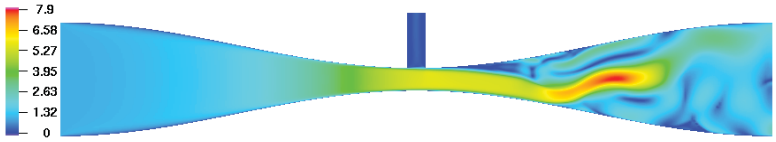
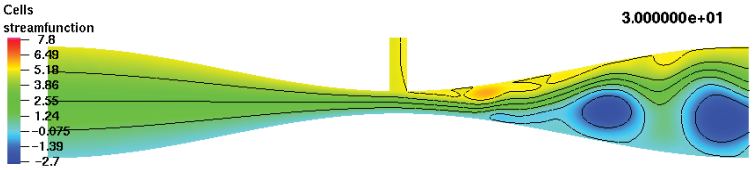


Abbildung 3.3.7: Schematischer Aufbau einer Venturipumpe.

Die Visualisierung der komplexen Wirbelstrukturen und der sich ergebenden Materialströmungen ist mit Schwierigkeiten behaftet. Keine der Euler-Methoden, welche die Stromfunktion oder die Strömungsstärke benutzen, zeigen viel von diesen Strukturen (siehe Abbildung 3.3.8(a) und (b)). Auch die Lagrange-Methode der Partikelverfolgung hat ihre Probleme, insofern sie nur Strömungsstrukturen zeigt, in denen auch eine Partikelquelle platziert wurde (siehe Abbildung 3.3.8(c)).



(a) Strömungsstärke



(b) Stromfunktion

(c) Partikelverfolgung

Abbildung 3.3.8: Anwendung "klassischer" Visualisierungsmethoden.

Die NATD-Methode liefert viel bessere Resultate, wie in der Abbildungsserie 3.3.9 zu sehen ist. Sogar sehr kleine Wirbelstrukturen wurden durch diese Methode im

[33] `https://de.wikipedia.org/wiki/Venturi-Düse`

[34] * 1746 in Bibbiano, †1822 in Reggio nell'Emilia.

Detail sichtbar. Dabei wurden jeweils vier Darstellungen von unterschiedlicher Skalengröße miteinander kombiniert, was für die vorliege Anwendung völlig ausreicht, um eine nahezu konstant bleibende Darstellungsqualität zu erreichen. In der kontrastverstärkenden rechten Seite wurde $\rho = 80$ gesetzt. Die verwendete Zeitschrittweite für die Berechnung der Darstellung betrug $\Delta t = 0.005$ auf einem Rechengitter mit 82753 Knoten.
Um den Einfluss der verschiedenen Parameter zu illustrieren, wurden auch Vergleichsrechnungen mit variierenden Parametern und nur einer Lösung durchgeführt. Wie in den Darstellungen von Abbildung 3.3.10 zu sehen ist, liefert ein zu kleiner Wert von ρ nur eine verwaschene Darstellung mit Andeutungen von Strukturen. Ein zu großer Wert bewirkt aber eine "überzeichnete" Darstellung, welche schwächere Details beziehungsweise Feinheiten der Strömung eliminiert. Eine moderate Einstellung von ρ (10-15 ist oft geeignet) ist wichtig für eine ausgewogene Darstellung.
Auch die Stärke der tangentialen Diffusion, welche über den Parameter β gesteuert wird, hat einen großen Einfluss auf die Darstellung. Ist β zu klein, so werden die vorhandenen Strukturen in den Texturen nicht erfolgreich zu linienartigen Strukturen verbunden. Dies ist gut in der Abbildungsserie 3.3.11 zu sehen.
Um die Wichtigkeit einer gewissen Mindestanzahl von gleichzeitig Lösungen zu unterschiedlichen Skalen zu verdeutlichen, wurde eine Rechnung mit nur zwei gleichzeitigen Lösungen berechnet und in der Abbildungsserie 3.3.12 hier präsentiert. Wie klar ersichtlich ist, springt die Darstellungsskala hin und her. Tests an den verschiedenen Beispielen haben ergeben, dass vier gleichzeitige Lösungen ausreichend für eine stabile Darstellung sind. Eine weitere Erhöhung der Anzahl der Lösungen verbessert die Qualität nur noch geringfügig, erfordert aber eine merkliche Erhöhung der Rechenzeit, da eine weitere unabhängige Lösung berechnet werden muss. Daher ist in der Praxis davon abzuraten.

3.4 3D-Beispiele

Zur Anwendung des NATD-Operators (2.1) auf Dimensionen $d > 2$, muss der enthaltene NAD-Operator entsprechend (1.10) angepasst werden. Damit werden die Rauschfelder wie zuvor durch die stärkere Diffusion in Strömungsrichtung zu linienartigen Strukturen zusammengeführt. Die schwächere Diffusion, welche durch die Funktion $g(\cdot)$ bestimmt wird, wirkt dabei lokal auf die Ebene, die orthogonal zur Strömungsrichtung liegt.
Obwohl sich diese Methode auch auf 3D-Strömungsdaten anwenden lässt, ist eine brauchbare Darstellung der Ergebnisse sehr schwer zu erreichen. Um die entstandenen linearen Strukturen ("Schlieren") im Inneren des Gebietes sehen zu können, müsste man in das Innere des Gebietes hineinsehen können und gleichzeitig verhindern, dass sich die Schlieren gegenseitig verdecken. Ein Versuch die Struktur einer 3D-Strömung durch alleinige Anzeige der Daten in Gebieten mit größerer Strömungsnorm zu visualisieren ist, wie in Abbildung 3.4.1 zu sehen ist, leider nicht sehr überzeugend. Für die eigentliche grafische Darstellung der resultieren-

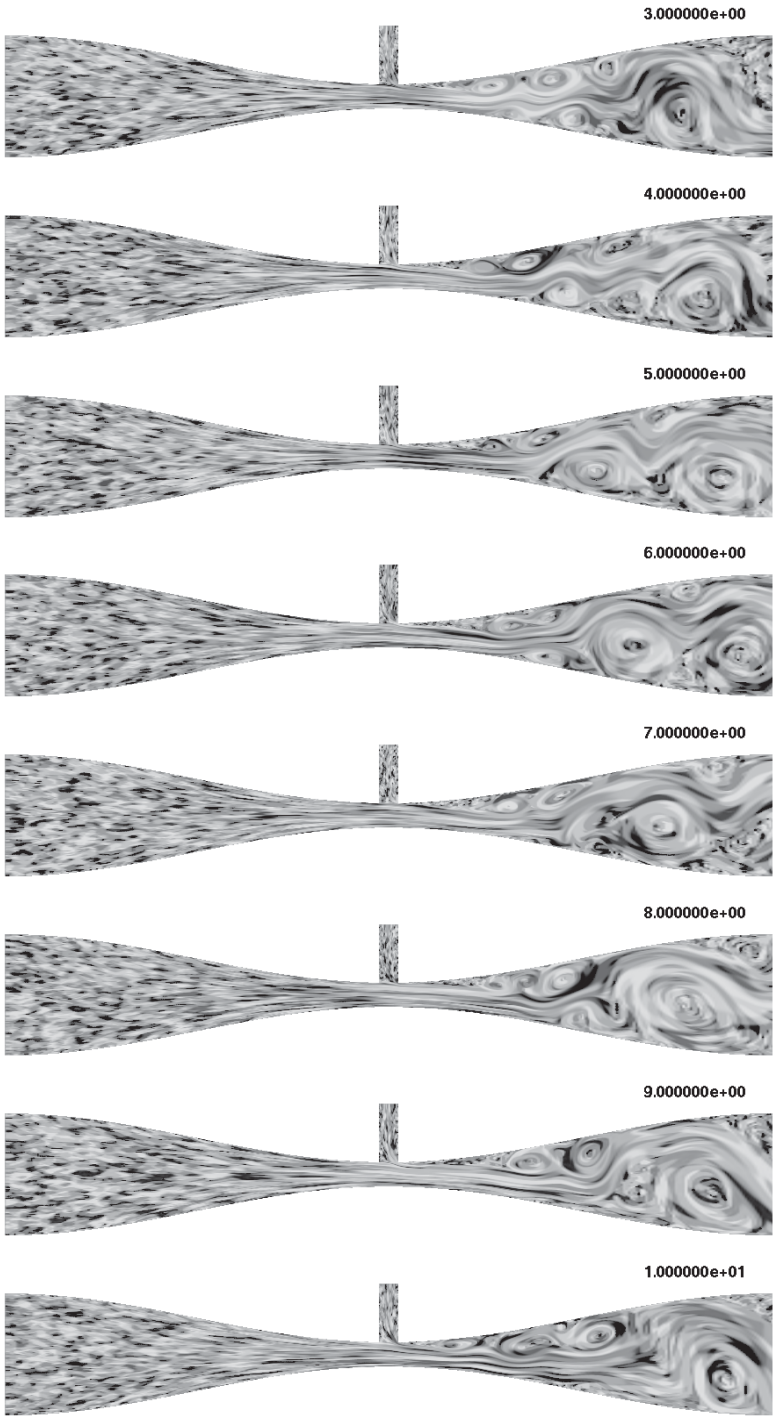


Abbildung 3.3.9: Multiskalenvisualisierung einer Venturipumpe.

den Daten wurde dabei GMV verwendet.

Etwas ähnliches kann man bei einem Wasserkocher mit transparenten Seitenwänden sehen. Der unterschiedliche Brechungsindex von heißem und kaltem Wasser ermöglicht es, die vom Heizelement aufsteigende Konvektionsströmung zu sehen.

(a) $\rho = 0.1$

(b) $\rho = 1$

(c) $\rho = 10$

(d) $\rho = 80$

Abbildung 3.3.10: Visualisierung mit verschiedenen Werten von ρ. (Keine Überblendung)

Eine geeignete Beleuchtung, Hintergrund und die Möglichkeit, den Betrachtungspunkt oder die Position der Lichtquellen zu verändern, sind für ein optimales Ergebnis wünschenswert.

Um dies im Rechner zu replizieren, könnte zum Beispiel ein volumetrisches Raytracing, mit durch die Ergebnisse moduliertem Brechungsindex, durchgeführt werden, was einen Rechenaufwand bedeutet, der wohl wesentlich größer als die ursprüngliche Strömungsberechnung ist, sich im Gegensatz zu dieser, wegen der Unabhängigkeit der einzelnen Strahlen voneinander, aber wohl besser parallelisieren ließe. Um die dreidimensionale Struktur der Strömung korrekt einem Betrachter zu vermitteln, könnte man zum Beispiel den Betrachtungspunkt veränderlich machen. Dies könnte durch die Vorgabe eines Kamerapfades oder Unmengen an Rechenaufwand für eine interaktive Echtzeitberechnung ermöglicht. Der Aufwand für die Visualisierung der Strömungsberechnung ist somit für eine praktische Anwendung viel zu groß. Die Verwendung der hier vorgestellten Visualisierungsmethode für 3D-Strömungsdaten wurde daher nicht weiter verfolgt.

(a) $\beta = 0.1$

(b) $\beta = 1$

(c) $\beta = 10$

Abbildung 3.3.11: Visualisierung zu verschiedenen Werten von β und konstantem ρ. (Keine Überblendung)

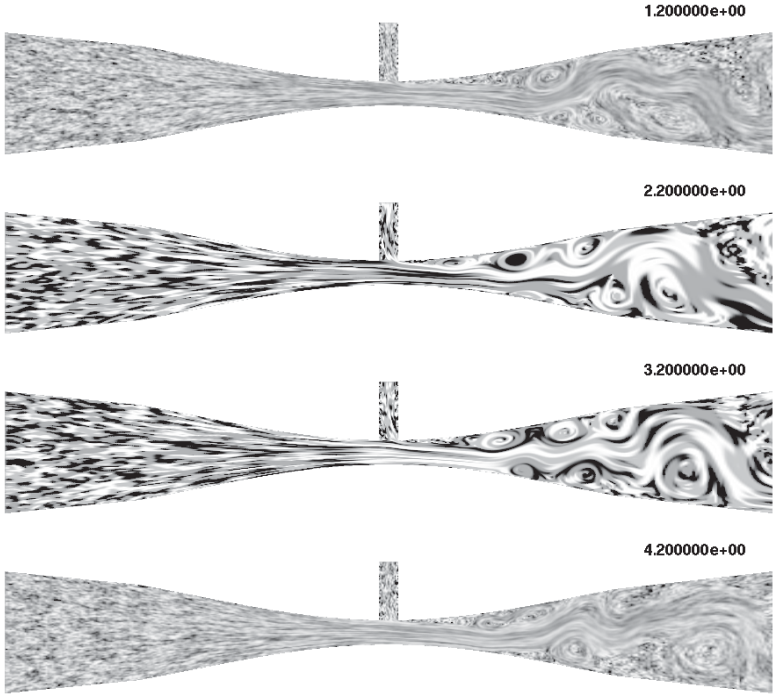


Abbildung 3.3.12: Unstete Dartellung bei Visualisierung mit nur zwei überblendeten Lösungen.

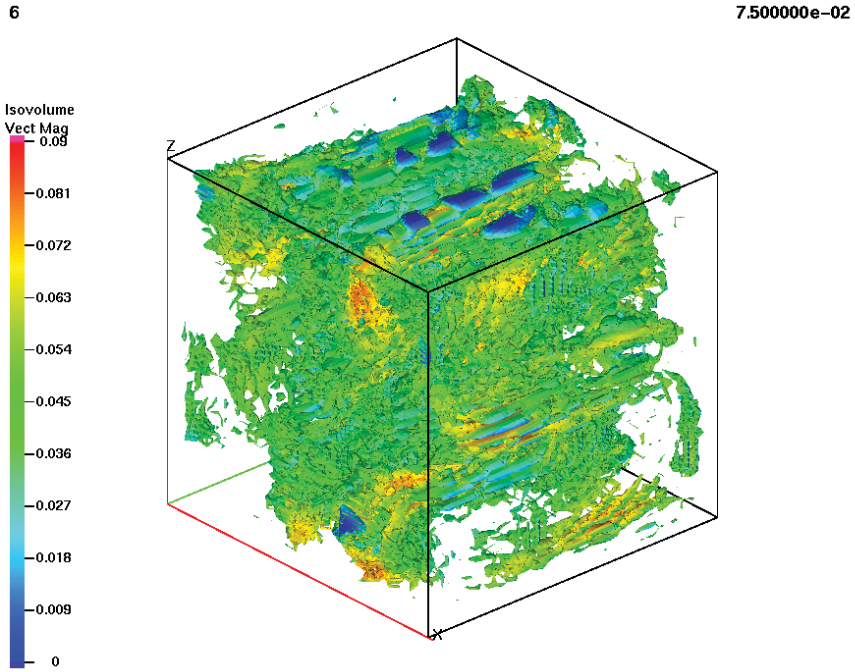


Abbildung 3.4.1: Darstellung einer Bénard-Strömung in 3D.

3.4.1 Verwendete numerische Methoden

Die Software, die zur Berechnung der Texturdarstellung der 3D-Strömung verwendet wurde (Visualisierung siehe Abbildung 3.4.1), wurde von mir im Laufe einer zeitweiligen Arbeit in der Forschungsgruppe um Professor Martin Rumpf [35] am "Institute for Numerical Simulation"[36] der Rheinischen Friedrich-Wilhelms-Universität Bonn geschrieben. Von dieser Gruppe wurden auch die Strömungsdaten der 3D-Strömung bereitgestellt.
Es gab dort zwar schon einen Vorläufer, der mit finiten Differenzen und Upwind-Stabilisierung gearbeitet hat, der aber von mir durch ein von Grund auf neu geschriebenes Programm ersetzt wurde. Dieses kann sowohl 2D- als auch 3D-Fälle rechnen, benutzt bilineare/trilineare FEM mit Stromliniendiffusion und kann zusätzlich auch variierende Zeitintervalle in den Eingabedaten verarbeiten. Für die Integration der rechten Seite können auch verschiedene Quadraturregeln eingestellt werden. Das Zeitschrittverfahren lässt sich mittels eines Parameters von explizitem zu implizitem Euler umstellen. Der Nachteil ist aber, dass es nur auf einem äquidistanten Tensorproduktgitter auf einen quadratische/kubischem Gebiet arbeitet! Wie zuvor erwähnt, wurde die resultierende Darstellung in eine GMV-Datei geschrieben und danach mit GMV die Abbildung 3.4.1 erstellt. Dabei wurde ein Isovolumen durch einen ausgewählten Bereich der Texturwerte erzeugt und basierend auf der Norm der Geschwindigkeit eingefärbt.

[35] E-Mail: martin.rumpf@ins.uni-bonn.de
[36] `http://numod.ins.uni-bonn.de/organisation/home.htm`

Zusammenfassung

In der vorliegenden Arbeit wurden so verschiedene Themenkomplexe wie Bildentrauschung und Visualisierung von Strömungen unter einem gemeinsamen Aspekt zusammengeführt: der Anwendung von partiellen Differentialgleichungen, finiten Elementen und modifizierten CFD-Programmen auf diese Probleme. Dabei wurden die Themen jeweils in eine ähnliche Richtung entwickelt, so dass der NAD-Operator in verschiedener Form zur Anwendung kam. Im Falle der Visualisierung von Strömungen allerdings in einer um einen Transportterm erweiterten Form (NATD).
Im ersten Themenkomplex Bildentrauschung wurden vorherige Ansätze, wie zum Beispiel Faltung mit einem Gaußkern und Perona-Malik angeführt, um die Eigenschaften und Möglichkeiten von nichtlinearer anisotroper Diffusion (NAD mit/ohne Strukturtensor) aufzuzeigen. Nach einer Erweiterung durch einen Transportterm wurde später der NATD-Operator dazu benutzt, um unter Steuerung durch vorberechnete Strömungsfelder, Rauschtexturen so zu verändern, dass Visualisierungen dieser Strömungsfelder entstehen, die auch für problematische Strömungen geeignet sind, welche Strukturen auf unterschiedlichen Geschwindigkeitsskalen aufweisen oder über die keinerlei Vorabinformationen vorliegen. Die Eignung dieser Visualisierungsmethode für diese Art von Strömungen wurde anhand von Beispielen gezeigt. Insbesondere für vorher unklare Strömungsstrukturen ist diese Methode besser als andere Techniken, wie zum Beispiel Partikelverfolgung, deren Probleme auch in Anhang A.2 besprochen werden. Die Änderung des in den Vorarbeiten von Martin Rumpf und Tobias Preusser gezeigten Ansatzes auf einen variationellen FEM-Ansatz ermöglicht, auch mit unstrukturierten Gittern arbeiten zu können und auch die möglichen Anwendungsbereiche stark zu erweitern.
Zusätzlich wurde in Abschnitt 1.4 bezüglich des Themas Bildergänzung ("Inpainting") ein interessanter Ansatz zur Verwendung von CFD-Techniken vorgestellt und dessen Vor- und Nachteile an Beispielen getestet. Dabei wurde ein Ansatz von Bayumy A Youssef, der mit einer Wirbelstromformulierung des imkompressiblen Navier-Stokes (iNS) Problems auf finiten Volumen (FV) arbeitet, auf einen FEM-Ansatz mit der Standardformulierung des (iNS) umgestellt und in eine Form gebracht, dass vorhandene Numeriksoftware (FeatFlow) damit arbeiten konnte. Um die Bildergänzung zu beschleunigen, wurde ein Programm geschrieben (Quellcode siehe Anhang B), das aus benutzergenerierten Angabe von Schadstellen in Form einer Bildmaske, automatisch optimierte Rechengitter generiert, welche zusammenhängende Segmente des ursprünglichen Tensorproduktgitters darstellen. Dabei traten bei der Erzeugung der Randparametrisierung (wird sowohl für Feast als auch FeatFlow benötigt) diverse Problemfälle auf, die aber programmintern durch zwei zusätzliche Arbeitsschritte behoben werden konnten. Dies reduzierte die Größe der zu rechnenden Probleme stark und führte zu einer starken Beschleunigung der Rechnungen. Siehe dazu auch mein Fazit in Abschnitt 1.4.6, das ich bezüglich dieser Methode gezogen habe.

Literaturverzeichnis

[AT00] Acker, J. ; Turek, S.: Postprocessing of FEATFLOW data with the particle tracing tool GMVPT (Version 1.2) / Fakultät für Mathematik, TU Dortmund. 2000. – Forschungsbericht. – Ergebnisberichte des Instituts für Angewandte Mathematik, Nummer 193

[BBS01] Bertalmio, M. ; Bertozzi, A. L. ; Sapiro, G.: Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting. In: *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2001

[BBS02] Bertalmio, M. ; Bertozzi, A. L. ; Sapiro, G.: *Navier-Stokes, Fluid Dynamics, and Image and Video Inpainting.* 2002. – *http://iie.fing.edu.uy/investigacion/grupos/gti/charlas/m-20021220/final-cvpr.pdf*

[Ber] `http://www.dtic.upf.edu/~mbertalmio/restoration2.html`

[BPR00] Becker, J. ; Preusser, T. ; Rumpf, M.: PDE Methods in Flow Simulation Post Processing. In: *Computing and Visualization in Science* 3 (2000), Nr. 3, S. 159–167

[BPR01] Bürkle, D. ; Preusser, T. ; Rumpf, M.: Transport and Anisotropic diffusion in time-dependent flow visualization. In: *Proceedings Visualization '01*, 2001

[BSCB11] Bertalmio, M. ; Sapiro, G. ; Caselles, V. ; Ballester, C.: *Image Inpainting.* Mai 2011. – *http://serv2.ist.psu.edu:8080/viewdoc/download;jsessionid=1F25206A551858795AC2402A50D89855?doi=10.1.1.107.8630&rep=rep1&type=pdf*

[CBFAB97] Charbonnier, P. ; Blanc-Féraud, M. ; Aubert, G. ; Barlaud, M.: Deterministic edge-preserving regularization in computing imaging. In: *IEEE Transactions on Image Processing* Bd. 6(2), 1997, S. 298–311

[CL93] Cabral, B. ; Leedom, L.: Imaging Vector Fields Using Line Integral Convolution. In: Kajiya, James T. (Hrsg.): *Computer Graphics (SIGGRAPH '93 Proceedings)* Bd. 27, 1993, S. 263–272

[CLMC92] Catté, F. ; Lions, P.-L. ; Morel, J.-M. ; Coll, T.: Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. In: *SIAM J. Numer. Anal.* 29 (1992), Nr. 1, S. 182–193

[CPC90] Chong, M. S. ; Perry, A.E. ; Cantwell, B. J.: A general classification of three-dimensional flow fields. In: *Phys. Fluids A* 2 (1990), Nr. 5, S. 765–777

[DKMT08] *Kapitel* 4.6. In: Dahlhaus, Rainer (Hrsg.) ; Kurths, Jürgen (Hrsg.) ; Maass, Peter (Hrsg.) ; Timmer, Jens (Hrsg.): *Mathematical Methods in Time Series Analysis and Digital Image Processing*. Springer-Verlag Berlin Heidelberg, 2008. – ISBN 978–3–540–75631–6, S. 138–151

[DPR00] Diewald, U. ; Preusser, T. ; Rumpf, M.: Anisotropic Diffusion in Vector Field Visualization on Euclidean Domains and Surfaces. In: *Trans. Vis. and Comp. Graphics* 6 (2000), Nr. 2, S. 139–149

[ENC] *Von Kármán vortex shedding*. `http://www.encyclopediaofmath.org/index.php?title=Von_K%C3%A1rm%C3%A1n_vortex_shedding&oldid=23554`, Abruf: 26.6.2016

[Eva98] Evans, Lawrence C.: *Graduate Studies in Mathematics*. Bd. 19: *Partial differential Equations*. American Mathematical Society, 1998. ISSN 1065–7339

[HWM88] Hunt, J. C. R. ; Wray, A. A. ; Moin, P.: Eddies, Stream and convergence zones in turbulent flow fields / Center for turbulence research. 1988 (CTR-S88). – Forschungsbericht

[IG97] Interrante, V. ; Grosch, C.: Stragegies for Effectively Visualizing 3D Flow with Volume LIC. In: *Proceedings Visualization '97*, 1997, S. 285–292

[JH95] Jeong, J. ; Hussain, F.: On the identification of a vortex. In: *Journal of Fluid Mechanics* 285 (1995), S. 69–94

[JK07] John, Volker ; Knobloch, Petr: On spurious oscillations at layers diminishing (SOLD) methods for convection–diffusion equations: Part I–A review. In: *Computer Methods in Applied Mechanics and Engineering* 196 (2007), Nr. 17, S. 2197–2215

[JK08] John, Volker ; Knobloch, Petr: On spurious oscillations at layers diminishing (SOLD) methods for convection–diffusion equations: Part II–Analysis for P1 and Q1 finite elements. In: *Computer Methods in Applied Mechanics and Engineering* 197 (2008), Nr. 21, S. 1997–2014

[KDA97] Kornprobst, P. ; Deriche, R. ; Aubert, G.: Nonlinear operators in image restoration. In: *Proceedings of the Internation Conference on Computer Vision and Pattern Recognition* IEEE Conputer Society, IEEE, June 1997, S. 325–331

[KK98] Kawohl, B. ; Kutev, N.: Maximum and comparison principle for one-dimensional anisotropic diffusion. In: *Math. Ann.* 311 (1) (1998), S. 107–123

[Kös04] Köster, M.: *Robuste Mehrgitter-Krylowraum-Techniken für FEM-Verfahren*, TU Dortmund, Fakultät für Mathematik, Lehrstuhl 3 für Angewandte Mathematik und Numerik, Diploma Thesis, 2004

[LHD+04] Laramee, R.S. ; Hausser, H. ; Doleisch, H. ; Vrolijk, B. ; Post, F.H. ; Weiskopf, D.: The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. In: *Computer Graphics forum* 23 (2004), Nr. 2, S. 203–221

[PM87] Perona, P. ; Malik, J.: Scale space and edge detection using anisotropic diffusion. In: *IEEE Computer Society Workshop on Computer Vision*, 1987

[PR99] Preusser, T. ; Rumpf, M.: Anisotropic Nonlinear Diffusion in Flow Visualization. In: *Proceedings Visualization 1999*, 1999, S. 325–332

[RSW00] Radmoser, Ester ; Scherzer, Otmar ; Weickert, Joachim: Scale-Space Properties of Nonstationary Iterative Regularization Methods. In: *Journal of Visual Communication and Image Representation* 11 (2000), June, Nr. 2, S. 96–114. `http://dx.doi.org/doi:10.1006/jvci.1999.0437`. – DOI doi:10.1006/jvci.1999.0437

[SK97] Shen, H.-W. ; Kao, D. L.: UFLIC: A Line Integral Convolution Algorithm For Visualizing Unsteady Flows. In: *Proceedings Visualization '97*, 1997, S. 317–322

[TB96] Turk, G. ; Banks, D.: Image-guided streamline placement. In: *Proc. 23rd annual conference on Computer graphics, August 4 - 9, 1996, New Orleans, LA USA*, ACM Press, 1996

[TBK99] Turek, S. ; Becker, Ch. ; Kilian, S.: Consequences of modern hardware design for numerical simulations and their realization in FEAST. In: P. Amesto, P. Berger, M. Dayde, I. Duff, V. Fraysse, L. Giraud, D. Ruiz (Hrsg.): *Proceedings Euro-Par'99 Parallel Processing*, 1999. – Toulouse, France, August/September 1999

[TD02] Tschumperle, D. ; Deriche, R.: Vector-Valued Image Regularization with PDE's :a Common Framework for Different Applications

/ I.N.R.I.A. 2004 Rte des Lucioles, BP 93,06560 Sophia Antipolis, France, 2002. – Forschungsbericht

[TD05a] Tschumperle, D. ; Deriche, R.: Vector-Valued Image Regularization with PDEs: a Common Framework for Different Applications. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), April, Nr. 4, S. 1–12

[TD05b] Tschumperlé, David ; Deriche, Rachid: Vector-Valued Image Regularization with PDEs: A Common Framework for Different Applications. In: *IEEE transactions on pattern analysis and machine intelligence* 27 (2005), april, Nr. 4, S. 506–517. `http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.87`. – DOI http://doi.ieeecomputersociety.org/10.1109/TPAMI.2005.87. – ISSN 0162–8828

[TJ82] Tobak, M. ; J., Peake D.: Topology of 3D separated flow. In: *Ann. Rev. Fluid Mech.* 14 (1982), S. 61–85

[Tsc02] Tschumperlé, David: *Thesis: PDE's Based Regularization of Multivalued Images and Applications*, Universitè de Nice - Sophia Antipolis, Diss., Dezember 2002

[Tur99] Turek, S.: *LNCSE*. Bd. 6: *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*. Springer, 1999

[van92] van der Vorst, H. A.: Bi-CGStab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. In: *SIAM J. Sci. Stat. Comp.* 13 (1992), Nr. 2, S. 631–644

[Wei98] Weickert, J.: *Anisotropic diffusion in image processing*. Teubner, 1998. ISSN ISBN 3–519–02606–6

[Wij91] Wijk, J. J.: Spot noise-Texture Synthesis for Data Visualization. In: Sederberg, T.W̃. (Hrsg.): *Computer Graphics (SIGGRAPH '91 Proceedings)* Bd. 25, 1991, S. 309–318

[Wij93] Wijk, J. J.: Flow visualization with surface particles. In: *IEEE Computer Graphics and Applications* 13 (1993), Juli, Nr. 4, S. 18–24

[YA07] Youssef, B. A. ; Atta, E. H.: Digital image inpainting using finite volume approach and the Navier-Stokes equations. In: *Proceedings of the 9th WSEAS international conference on Mathematical and computational methods in science and engineering*. Stevens Point, Wisconsin, USA : World Scientific and Engineering Academy and Society (WSEAS), 2007 (MACMESE'07). – ISBN 978–960–6766–12–1, 305–311

Anhang A

Partikelverfolgung

A.1 Motivation

Die bei den Strömungsberechnungen anfallenden Daten wie Druck, Strömungsstärke und Richtung lassen sich für jeden Zeitpunkt punktweise zumindest interpoliert angeben. Bildet man diese Werte auf Farbskalen oder im Fall der Richtung auf Vektorpfeile ab, so lassen sich erste visuelle Darstellungen einer Strömung erstellen. Dies hat allerdings zwei Probleme.
Das erste Problem besteht darin, dass nicht alle Strömungsprozesse innerhalb der gleichen Skalen ablaufen. Dies können zum Beispiel Strömungen und Wirbel mit stark unterschiedlicher Geschwindigkeit sein. Man kann versuchen, diese einzeln durch Untersuchung von Teilbereichen der Werteskalen zu finden, aber eine gemeinsame Darstellung in einem Bild ist schwer. Zumal zuerst ein Verdacht über die Präsenz nicht-offensichtlicher Strömungen vorhanden sein muss.
Das zweite Problem liegt in der Betrachtung der Werte in einzelnen Punkten. Bei zeitveränderlichen Strömungen interessiert es mehr, wie das Wasser[1] in der Summe der Einflüsse über die Zeit fließt. Oder in anderen Worten, welchen Pfad jeder Wassertropfen nimmt. Diese Lagrange-Darstellung der Strömung lässt sich zum Teil mittels eines Programms zur Partikelverfolgung realisieren. Dabei werden masselose Partikel an bestimmte Stellen gesetzt und verfolgt, wie sie weiter transportiert werden. Hierbei muss man allerdings schon eine Vermutung haben, wo sich lohnende Stellen für Partikelquellen befinden und zudem abwarten, bis sich die Partikel von diesen Quellen ausgebreitet haben. Etwas Ähnliches erhält man, wenn man eine Substanzkonzentration oder Temperatur durch das errechnete Strömungsfeld transportieren lässt. Dabei tritt aber immer Diffusion auf, welche Strukturen aufweicht.
Ein gutes Beispiel dafür ist der DFG-Benchmark von 1995/96 mit Wirbelstraßen hinter einem Zylinder in 2D[2]. Die "normalen" Visualisierungsmethoden zeigen nur kleine Ausschläge an. Die Partikelverfolgung zeigt aber die Wirbelstraßen[3] auf.

[1] oder eine andere inkompressible Flüsigkeit, die sich mit den Navier-Stokes-Gleichungen modellieren lässt.

[2] *http://www.featflow.de/album/catalog/bench_cyl_2d/data.html*

[3] *http://www.featflow.de/album/channel_cyl.html*

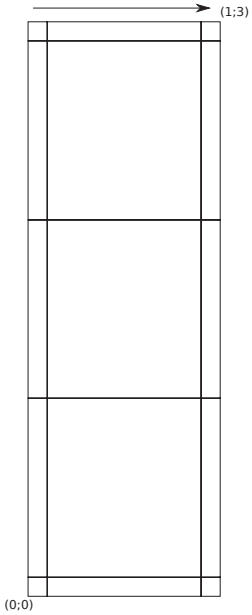
Für letztere Visualisierungen wurde von mir ein eigener "Partikel Tracer" names GMVPT geschrieben. Mehr dazu ist in [AT00] zu finden. Diese Programm ist allerdings speziell an die von FeatFlow generierten GMV-Dateien angepasst worden. Eine Alternative dazu bietet der "ParticleTracer"-Filter des Visualisierungsprogrammes ParaView[4], womit sich Strömungsdaten in einer großen Bandbreite von Eingangsformaten verarbeiten lassen.

A.2 Ein Beispiel mit Problemen

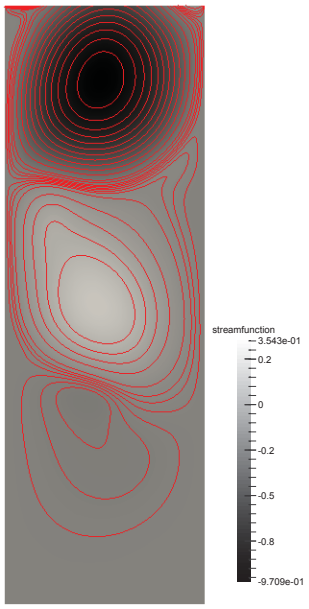
Ein Beispiel für eines der Probleme kann mittels des sogenannten "Driven Cavity"-Problems illustriert werden. Dabei wird für ein rechteckiges Gebiet eine Strömung tangential zur oberen Kante vorgegeben. Diese induziert direkt bei einem quadratischen Gebiet einen größeren Hauptwirbel mit geringerer Geschwindigkeit und indirekt weitere kleinere Wirbel von noch langsamerer Rotation in den vier Ecken. Bei einem Gebiet vom Format 1xN türmen sich die größeren Wirbel entlang des Gebietes auf, wobei der erste Hauptwirbel (Primärwirbel) einen langsameren Sekundärwirbel und dieser einen noch langsameren Tertiärwirbel induziert. Dies setzt sich theoretisch immer weiter fort, bis das Gebietsende erreicht oder die Energie durch Diffusion zerstreut wurde.
Die dabei auftretenden Wirbel befinden sich bezüglich ihrer Rotationsgeschwindigkeit auf sehr unterschiedlichen Skalen. Deshalb ist es sehr schwer, die komplette Wirbelstruktur in einem Bild zu visualisieren. Falls die oben genannte Wirbelstruktur unbekannt ist, so können Wirbel leicht übersehen werden.
Als konkretes Beispiel führe ich hier ein Beispiel vom Format 1x3 an (siehe Abbildungen A.2.1). Bei einer konstanten tangentialen Strömung der Stärke 10 erhält man, je nach Viskosität, zwischen 3 und 4 Hauptwirbel. Es wurde hier eine Viskosität von 1/500 verwendet, um nur drei Hauptwirbel zu erhalten. Wie in den Abbildungen A.1(b) und A.1(c) zu sehen ist, lassen sich die ersten beiden Hauptwirbel noch einigermaßen erkennen. Der dritte Hauptwirbel ist aber nur schwach sichtbar. In der Stromfunktionsdarstellung ist er nur anhand geschickt gewählter Isolinien erkennbar.
Eine Visualisierung mittels eines eingebrachten Stoffes/Temperatur lässt nach 30 simulierten Zeiteinheiten den dritten Wirbel erahnen (siehe Abbildung A.1(d). Da im Einströmungsbereich eine relativ hohe Geschwindigkeit vorgegeben wurde, muss zur Simulation ein relativ kleiner Zeitschritt benutzt werden. Die adaptive Zeitschrittweitensteuerung hatte sich am Ende auf eine Zeitschritt von circa 0.00294 Zeiteinheiten eingependelt.[5] Da im Rest des Gebietes wesentlich langsamere Strömungen vorhanden sind, ist es allein der Einströmbereich der verhindert, dass längere Zeitschrittweiten verwendet werden können und somit die Gesamtsimulation lange dauert.
Die so berechneten Strömungsdaten wurden abgespeichert und als Grundlage für

[4] *http://www.paraview.org/overview/*

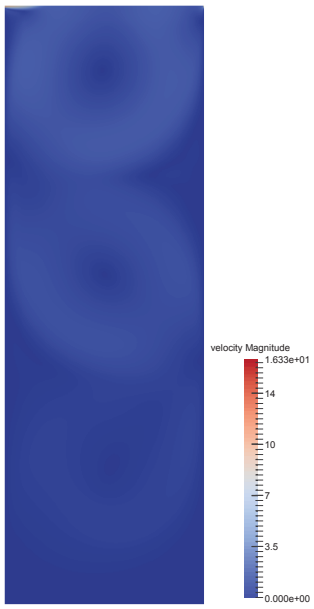
[5] Verwendetes Programm: BOUSS aus FeatFlow 1.3 mit Crank-Nicolson als Zeitschrittverfahren.



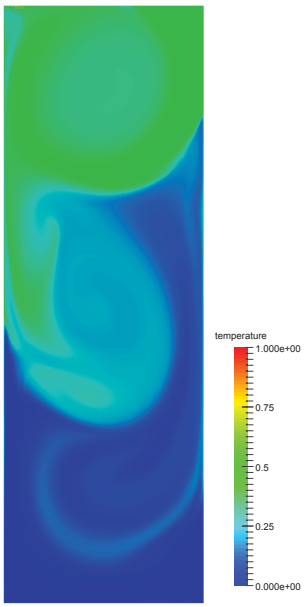
(a) Grobgitter und vorgegebene Strömung (konstant 10.0)



(b) Stromfunktion mit Isolinien (ungleichmäßig verteilt)



(c) Geschwindigkeit



(d) Konzentration

(e) Partikelverfolgung

Abbildung A.2.1: “Driven Cavity” auf einem Gebiet vom Format 1x3 Einheiten.

eine Partikelverfolgung mittels GMVPT verwendet. Zur Partikelsimulation wurde ein rechteckiger Block aus 6400 Partikeln in den obersten Bereich eingesetzt und dies regelmäßig alle 0.28 Zeiteinheiten wiederholt. Die Simulationszeitschrittweite lagen dabei bei $0.14 \cdot 10^{-3}$. Die Simulation wurde solange durchgeführt, bis Partikel in den dritten Wirbel vorgedrungen waren.
Dabei gehen Partikel leicht verloren oder bleiben an den Wänden hängen, da nur ein explizites Verfahren 1. Ordnung zur Bewegungsberechnung verwendet wird und selbst geschlossene Wirbelströmungen daher Partikel schnell nach außen wegdrücken und die Simulation verfälschen. Dieses Problem wird sowohl mit wachsender Zeitschrittweite als auch Strömungsgeschwindigkeit stärker, so dass deshalb nur eine sehr kleine Zeitschrittweite benutzt werden sollte.
Da es lange dauert, bis die Partikel in den dritten Wirbel vordringen, ist mit dieser Simulation am Anfang keine Aussage über diesen Wirbel möglich. Dafür müsste eine weitere Simulation angefertigt werden, mit zusätzlichen Partikelquellen in den nun bekannten Wirbelpositionen. Kleine Wirbel in den Ecken erscheinen kaum in dieser Darstellung. Dies verdeutlicht, dass für eine gute Partikelsimulation die Positionierung der Partikelquellen sehr wichtig ist! Gute Positionen ergeben sich aber erst, wenn schon Informationen oder ausreichende Vermutungen über die Strömungsstruktur vorliegen.
Bei dem hier vorgestellten Beispiel sind die Wirbel stationär. Bei Problemen mit wandernden Startpositionen von Wirbeln, bei denen unklar ist, wo genau die Wirbel auftreten werden, müsste man wandernde Partikelquellen und unter Umständen sogar eine Wirbelerkennung einführen, welche die Partikelquellen mit den Wirbeln mitbewegt. Bei sehr komplexen Strömung würde man damit aber auch an Grenzen stoßen, da Wirbel sich auch teilen, verschmelzen, neu bilden oder ganz verschwinden können. Würde man aber das ganze Gebiet regelmäßig mit Partikeln füllen, so würde die Anzahl der Partikel schnell sehr groß werden, die Simulation verlangsamen und die Übersichtlichkeit einschränken.
Die in dieser Arbeit vorgeführte Visualisierungsmethode aus Abschnitt 3.3 hat diese Probleme nicht, ist aber numerisch aufwändiger. Wenn die Struktur der Strömung durch diese Methode erkennbar wird, kann dies allerdings dazu benutzt werden, um gute Positionen für Partikelquellen für eine nachfolgende Visualisierung mittels Partikelverfolgung zu finden.

Anhang B

PNGtoFEAT

Das Programm PNGtoFEAT erwartet eine Bilddatei im PNG-Format mit nur einem Farbkanal und schwarzen und weißen Pixeln, welche eine Maske von Fehlstellen eines anderen Bildes mit derselben Auflösung darstellen. Daraus wird eine Folge von Rechengittern im FeatFlow-Format erzeugt, die zur Bildwiederherstellung genutzt werden.
Schwarze Pixel bedeuten dabei, dass das Bild an dieser Stelle unbeschädigt ist, alle anderen Graustufen dagegen, dass eine Schadstelle vorliegt. Anstatt PNG kann zudem auch jedes Bildformat verwendet werden, dass vom "Image"-Modul von Python unterstützt wird.
Achtung: Das Programm nimmt implizit an, dass die Maske zu einem Gebiet der Größe $[-1;1]^2$ korrespondiert. Ist dies nicht der Fall, so muss die Funktion "gridCoords" angepasst werden! Zudem wird der Dateiname am ersten Punkt in Präfix und Suffix aufgespalten. Daher sollte neben dem Punkt für die Dateierweiterung kein weiterer Punkt im Dateinamen vorkommen.

pngtofeat.py

```
#!/usr/bin/env python
# -*- coding: iso-8859-15 -*-

#PNGTOFEAT von Jens Friedrich Acker
#Programm fuer Python 2.7

from sys import argv
from itertools import count,izip
import Image

def genCellMask(nx,ny,Img):
  #Leere ZellMaske
  C=[[False for i in xrange(nx)] for j in xrange(ny)]
  #Teste die Knoten einer Zelle
  for j in xrange(ny):
    for i in xrange(nx):
      C[j][i]=(Img[i,j]!=0)or(Img[i+1,j]!=0)or(Img[i,j+1]!=0)or(Img[i+1,j+1]!=0)
  return C
```

```
def gridCoords(i,j,nx,ny):
  return ((2.0/nx)*i-1.0,1.0-(2.0/ny)*j)

def genMeshFromMask(nx,ny,CellMask):
  ydiff=nx+1
  NodeDict={}
  NrNodes=0
  Nodes=[]
  Cells=[]
  def mapNode(N,coords,NrNodes,Nodes,NodeDict):
    if NodeDict.has_key(N):
      return (NodeDict[N],NrNodes)
    else:
      NrNodes+=1
      NodeDict[N]=NrNodes
      Nodes.append(coords)
      return (NrNodes,NrNodes)

  for j in xrange(ny):
    for i in xrange(nx):
      if CellMask[j][i]:
        N1=j*ydiff+i+1
        N2=N1+ydiff
        N3=N1+ydiff+1
        N4=N1+1
        (N1,NrNodes)=mapNode(N1,gridCoords(i,j,nx,ny),NrNodes,Nodes,NodeDict)
        (N4,NrNodes)=mapNode(N4,gridCoords(i+1,j,nx,ny),NrNodes,Nodes,NodeDict)
        (N2,NrNodes)=mapNode(N2,gridCoords(i,j+1,nx,ny),NrNodes,Nodes,NodeDict)
        (N3,NrNodes)=mapNode(N3,gridCoords(i+1,j+1,nx,ny),NrNodes,Nodes,NodeDict)
        Cells.append((N1,N2,N3,N4))
  return (Nodes,Cells,NodeDict)

def genEdgeLists(nx,ny,CellMask,NodeDict):
  """
  Benutzt einen Scannline-Algorithmus um Randkanten
  mit korrekter Orientierung zu erzeugen.
  """
  ydiff=nx+1
  Edges=[]
  #Scanne zuerst entlang der Zeilen
  for j in xrange(ny):
    Inside=False
    for i in xrange(nx):
      if Inside and not CellMask[j][i]:
        Edges.append((NodeDict[ydiff*(j+1)+i+1],NodeDict[ydiff*j+i+1]))
        Inside=False
      elif (not Inside) and CellMask[j][i]:
        Edges.append((NodeDict[ydiff*j+i+1],NodeDict[ydiff*(j+1)+i+1]))
        Inside=True
    #Letzte Kante der Zeile getrennt überprüfen
    if Inside:
      Edges.append((NodeDict[ydiff*(j+1)+nx+1],NodeDict[ydiff*j+nx+1]))
      Inside=False
```

```
  #Scanne nun entlang der Spalten
  for i in xrange(nx):
    Inside=False
    for j in xrange(ny):
      if Inside and not CellMask[j][i]:
        Edges.append((NodeDict[ydiff*j+i+1],NodeDict[ydiff*j+i+2]))
        Inside=False
      elif (not Inside) and CellMask[j][i]:
        Edges.append((NodeDict[ydiff*j+i+2],NodeDict[ydiff*j+i+1]))
        Inside=True
    #Letzte Kante der Spalte getrennt überprüfen
    if Inside:
      Edges.append((NodeDict[ydiff*ny+i+1],NodeDict[ydiff*ny+i+2]))
      Inside=False
  return Edges

def fuseEdges(Edges,Nodes,Cells):
  """
  Verschmelze die Randkanten zu einzelnen Rändern und gebe diese
  als Liste von Rändern aus. Die Liste der Randkanten wird dabei
  verbraucht und sollte danach leer sein.
  """
  B=[]
  while Edges:
    Found=False
    E=Edges.pop()
    for i in xrange(len(Edges)):
      if(Edges[i][-1]==E[0]):
        #Verschmelzen
        Edges[i]=Edges[i][0:-1]+E[:]
        Found=True
        break
    if not Found:
      if E[-1]!=E[0]:
        print "Fehler: Nicht geschlossener Rand:",E
      E=E[:-1]
      B.append(E)
  #Verschmelze Schleifen
  tmp=[]
  while B:
    E=B.pop()
    Found=True
    while Found:
      Found=False
      for i in xrange(len(B)):
        #Haben die Ränder E und B[i] einen gemeinsamen Knoten
        c=set(E) & set(B[i])
        #Wenn ja, verschmelze die Ränder
        if c:
          Found=True
          b=B.pop(i)
          #Verschmelzung an irgendeinem gemeinsamen Knoten
          CNode=c.pop()
          Idx1=E.index(CNode)
```

```
          Idx2=b.index(CNode)
          E=E[:Idx1]+b[Idx2:]+b[:Idx2]+E[Idx1:]
          #Dieser neue Rand kann Verbindungen mit weiteren
          #Schleifen besitzen. Daher muss die gesammte Suche
          #mit E erneut durchgeführt werden.
          break
    tmp.append(E)
  B=tmp
  #Auftrennen von Randschleifen
  #Untersuche jeden einzelnen Rand
  NrNodes=len(Nodes)
  for j in xrange(len(B)):
    E=B[j]
    #Teste von Hinten, ob der Index des ersten Auftretens eines
    #Knoten mit dem momentanen Index übereinstimmt.
    for i in reversed(xrange(len(E))):
      Idx=E.index(E[i])
      if Idx!=i:
        #Ein doppeltes Auftreten eines Knotens bzw.
        #eine interne Schleife wurde gefunden.
        #An dieser Stelle muss der Knoten durch
        #einen neuen Knoten ersetzt werden
        NrNodes=NrNodes+1
        Nodes.append(tuple(Nodes[E[i]-1]))
        B[j]=E[:i]+(NrNodes,)+E[i+1:]
        N=E[Idx]
        Ne=E[(i+1)%len(E)]
        #Suche das erste Auftreten des Knotens N gefolgt von Ne
        #in der Zellliste und ersetze ihn durch NrNodes
        for i in xrange(len(Cells)):
          if N in Cells[i]:
            tmp=list(Cells[i])
            Idx=tmp.index(N)
            if tmp[(Idx+1)%len(tmp)]==Ne:
              tmp[tmp.index(N)]=NrNodes
              Cells[i]=tuple(tmp)
              break
  #Bringe die Ränder in die erwartete Reihenfolge.
  B.sort(cmp=lambda x,y: cmp(min(x),min(y)))
  return B

def writeFeat(NamePrefix,Nodes,Cells,Boundaries):
  #print "Verwende Namenspräfix \"%s\"."%NamePrefix
  #Baue Mapping für Randknoten auf
  ParamMap={}
  #Gebe die Parametrisierung aus.
  with open(NamePrefix+".prm","w") as PrmFile:
    Parameters=""
    PrmFile.write("NBCT\n%d\n"%len(Boundaries))
    for (Boundary,BoundaryNr) in izip(Boundaries,count(1)):
      PrmFile.write("IBCT\n%d\n"%BoundaryNr)
      PrmFile.write("NCOMP\n%d\n"%len(Boundary))
      PrmFile.write("ITYP NSPLINES NPAR\n")
      for (NodeNr,ParamVal) in izip(Boundary,count(0)):
```

```
        ParamMap[NodeNr]=(BoundaryNr,ParamVal)
      for Idx in xrange(len(Boundary)):
        IdxNext=(Idx+1)%len(Boundary)
        PrmFile.write("1 1 2\n")
        #Hänge Linienparamter an
        StartCoords=Nodes[Boundary[Idx]-1]
        EndCoords=Nodes[Boundary[IdxNext]-1]
        Difference=tuple(map(lambda a,b:a-b,EndCoords,StartCoords))
        Parameters+="%.8f %.8f\n"%StartCoords
        Parameters+="%.8f %.8f\n"%Difference
    PrmFile.write("PARAMETERS\n")
    PrmFile.write(Parameters)
  #Gebe das Gitter aus.
  with open(NamePrefix+".tri","w") as TriFile:
    TriFile.write("Coarse mesh written by pngtofeat.\n")
    TriFile.write("Parametrisierung PARXC, PARYC, TMAXC\n")
    HeaderData=(len(Cells),len(Nodes),len(Boundaries))
    TriFile.write("%d %d 0 4 %d \tNEL NVT NMT NVE NBCT\n"%HeaderData)
    #Ausgabe der Knotenkoordinaten bzw. Parameterwerte
    TriFile.write("DCORVG\n")
    for Idx in xrange(1,HeaderData[1]+1):
      if ParamMap.has_key(Idx):
        TriFile.write("%d 0\n"%ParamMap[Idx][1])
      else:
        TriFile.write("%.8f %.8f\n"%Nodes[Idx-1])
    #Ausgabe der Gitterzellen
    TriFile.write("KVERT\n")
    for Cell in Cells:
      TriFile.write("%d %d %d %d\n"%Cell)
    #Ausgabe der Knoteneigenschaften
    TriFile.write("KNPR\n")
    for Idx in xrange(1,HeaderData[1]+1):
      if ParamMap.has_key(Idx):
        TriFile.write("%d\n"%ParamMap[Idx][0])
      else:
        TriFile.write("0\n")
    #Ausgabe der Knoten mit minimalem und maximalem Parameterwert
    #für jeden Rand
    TriFile.write("KMM\n")
    for Boundary in Boundaries:
      TriFile.write("%d %d\n"%(Boundary[0],Boundary[-1]))
  #Fertig

def findComponent(C,Mask,i,j,nx,ny):
  """
  Iterativer Algorithmus zum finden aller Zellen in einer
  gemeinsamen Zusammenhangskomponente, wobei der Zusammenhang
  über die Zellkanten bestimmt ist.
  """
  def Inside(i,j,nx,ny):
    return ((i>=0)and(i<nx)and(j>=0)and(j<ny))
  def TestAndMark(a,b,nx,ny,C,Mask,S):
    if Inside(a,b,nx,ny):
      if Mask[b][a]:
```

```
        C[b][a]=True
        Mask[b][a]=False
        S.add((a,b))
  #Test ob die Indices ausserhalb liegen. Wenn ja, tue nichts!
  if not Inside(i,j,nx,ny):
    return
  #Liegt die Zelle im Markierten Bereich. Wenn nicht, tue nicht!
  if not Mask[j][i]:
    return
  #Initialisiere Set aktiver Suchzellen
  S=set()
  #Markiere die Position (i,j) in C und entferne sie aus Mask
  C[j][i]=True
  Mask[j][i]=False
  S.add((i,j))
  while len(S)>0:
    (x,y)=S.pop()
    #Teste die vier direkten Nachbarn von (x,y)
    TestAndMark(x-1,y,nx,ny,C,Mask,S)
    TestAndMark(x+1,y,nx,ny,C,Mask,S)
    TestAndMark(x,y-1,nx,ny,C,Mask,S)
    TestAndMark(x,y+1,nx,ny,C,Mask,S)
  #Fertig

def splitCellMask(nx,ny,CellMask):
  CellMasks=[]
  for j in xrange(ny):
    for i in xrange(nx):
      if CellMask[j][i]:
        #Generiere leere Zellmaske C
        C=[[False for x in xrange(nx)] for y in xrange(ny)]
        #Bestimme Zusammenhangskomponente
        findComponent(C,CellMask,i,j,nx,ny)
        #Sichere Zusammenhangskomponente
        CellMasks.append(C)
  return CellMasks

# Test auf korrekte Anzahl der Kommandozeilenparameter
if len(argv)!=2 :
  print """
  Falsche Parameteranzahl!

  Aufruf: pngtofeat filename.png
  Ausgabe: filename.prm & filename.tri
  """
else:
  print "PNGTOFEAT Version 2"
  print "==================\n"
  #Bild öffnen und Inhalt überprüfen
  ImgFile=Image.open(argv[1])
  Img=ImgFile.load()
  (NX,NY)=ImgFile.size
  if (NX<=1) or (NY<=1):
    print "Fehler: Bildformat kleiner als 2x2! Momentanes Format: %dx%d"%(NX,NY)
```

```
  exit()
(m,M)=ImgFile.getextrema()
if M==0:
  print "Fehler: Leere Maske. Es gibt nichts zu tun!"
  exit()
#Eigentliches Hauptprogramm beginnt erst hier.
NamePrefix=argv[1].rsplit(".",1)[0]
print "Bildformat: %dx%d"%(NX,NY)
CellMask=genCellMask(NX-1,NY-1,Img)
CellMasks=splitCellMask(NX-1,NY-1,CellMask)
if len(CellMasks)==1:
  print "Ein Einzelgebiet gefunden."
else:
  print "%d Einzelgebiete gefunden."%len(CellMasks)
for (CellMask,Idx) in izip(CellMasks,count(1)):
  print "Bearbeite Gebiet Nr. %d."%Idx
  (Nodes,Cells,NodeDict)=genMeshFromMask(NX-1,NY-1,CellMask)
  if len(Cells)==0:
    print "Keine Zelle im Rechengebiet!"
  elif len(Cells)==1:
    print "Genau eine Zelle im Rechengebiet."
  else:
    print "%d Zellen im Rechengebiet."%len(Cells)
  BoundaryEdges=genEdgeLists(NX-1,NY-1,CellMask,NodeDict)
  Boundaries=fuseEdges(BoundaryEdges,Nodes,Cells)
  if len(Boundaries)==0:
    print "Kein Rand gefunden!"
  elif len(Boundaries)==1:
    print "Genau ein Rand gefunden."
  else:
    print "%d Ränder gefunden."%len(Boundaries)
  print "Schreibe Gebietsdaten heraus..."
  writeFeat(NamePrefix+"_%d"%Idx,Nodes,Cells,Boundaries)
print "Fertig."
```

Anhang C

Grobgitter

In diesem Addendum werden diverse Grobgitter gesammelt, die bei den Rechnungen verwendet wurden. Mit diesen sollte es etwas einfacher sein, eigene Rechnungen zur Überprüfung der Beispiele durchzuführen.

C.1 TRISYM

C.1.1 trisym.prm

```
NBCT
1
IBCT
1
NCOMP
1
ITYP NSPLINE NPAR
2 1 3
PARAMETERS
0.00000 0.00000
1.00000 0.0E0
0.0 6.283185307179586
```

C.1.2 trisym.tri

```
Gitterbeschreibung fuer trisym
Parametrisierung PARCX, PARCY, TMAXC
9  13 0  4 1   NEL NVT NMT NVE NBCT
DCORVG
0.0 0.0
0.166666666667 0.0
0.333333333334 0.0
0.500000000001 0.0
0.666666666668 0.0
0.833333333335 0.0
0.00000 0.00000
0.50000 0.00000
-0.50000 0.00000
0.25000 0.43301
-0.25000 0.43301
-0.25000 -0.43301
0.25000 -0.43301
KVERT
8 1 2 10
11 10 2 3
9 11 3 4
12 9 4 5
13 12 5 6
8 13 6 1
10 7 13 8
9 7 10 11
9 12 13 7
KNPR
1 1 1 1 1 1 0 0 0 0 0 0 0
KMM
1 6
```

C.2 SB25

C.2.1 sb25.prm

```
NBCT
2
IBCT
1
NCOMP
4
ITYP NSPLINE NPAR
1 1 2
1 1 2
1 1 2
1 1 2
IBCT
2
NCOMP
1
ITYP NSPLINE NPAR
2 1 3
PARAMETERS
-0.16000 -0.04000
2.90000 0.00000
2.74000 -0.04000
0.00000 0.62000
2.74000 0.57999
-2.90000 0.00000
-0.16000 0.57999
0.00000 -0.62000
0.52000 0.28000
0.08000 0.0E0
10.995574287564276 4.71238898038469
```

C.2.2 sb25.tri

```
Gitterbeschreibung fuer sb25
Parametrisierung PARCX, PARCY, TMAXC
25  38 0  4 2   NEL NVT NMT NVE NBCT
DCORVG
0.0 0.0
0.16666666666 0.0
0.33333333333 0.0
0.49999999999 0.0
0.66666666666 0.0
0.83333333333 0.0
3.0 0.0
3.35 0.0
3.67 0.0
0.0 0.0
2.835 0.0
0.31850 0.36298
0.31850 0.16459
0.165 0.0
```

```
0.52149 0.47500
2.765 0.0
0.52149 0.08299
0.235 0.0
2.695 0.0
0.72450 0.36298
0.72450 0.16299
0.305 0.0
2.55 0.0
1.14500 0.36298
1.14500 0.16299
0.45 0.0
2.38 0.0
1.63800 0.36298
1.63800 0.16299
0.62 0.0
2.16000 0.36298
2.16000 0.16299
2.2 0.0
0.8 0.0
2.0 0.0
1.0 0.0
1.65 0.0
1.33 0.0
KVERT
10 14 13 9
9 13 12 8
8 12 11 7
11 12 15 16
3 4 15 12
2 3 12 13
1 2 13 17
13 14 18 17
16 15 20 19
4 5 20 15
5 6 21 20
1 17 21 6
17 18 22 21
19 20 24 23
20 21 25 24
21 22 26 25
23 24 28 27
24 25 29 28
25 26 30 29
27 28 31 33
28 29 32 31
29 30 34 32
34 36 38 32
32 38 37 31
31 37 35 33
KNPR
2 2 2 2 2 2 1 1 1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 1
KMM
10 9 1 6
```

C.3 Rückwärtsgerichtete Stufe

C.3.1 step.prm

```
NBCT
  1
IBCT
  1
NCOMP
  6
ITYP NSPLINE NPAR
  1     1     2
  1     1     2
  1     1     2
  1     1     2
  1     1     2
  1     1     2
PARAMETERS
 10 0
 0 1
 10 1
 -10 0
 0 1
 0 -0.5
 0 0.5
 2 0
 2 0.5
 0 -0.5
 2 0
 8 0
```

C.3.2 step.tri

```
Coarse mesh 2D
converted by tri_param2D_writer
22 36 0 4 1 NEL NVT NMT NVE NBCT
DCORVG
2 0
2.5 0
3 0
1.899999999999999911 0
1 0.75
3.5 0
2 0.75
4 0
5 0
1.75 0
2.5 0.5
5.0625 0
1.675000000000000044 0
3.25 0.5
5.15625 0
1.600000000000000089 0
```

```
4 0.5
5.25 0
1.5 0
5 0.5
5.375 0
1.399999999999999911 0
6 0.5
5.5 0
1.300000000000000044 0
7 0.5
5.625 0
1.200000000000000178 0
8 0.5
5.75 0
1.100000000000000089 0
9 0.5
5.875 0
1 0
0.5 0
0 0
KVERT
 1 2 5 4
 2 3 6 5
 4 5 7 10
 5 6 8 7
 7 8 11 10
 8 9 12 11
 10 11 14 13
 11 12 15 14
 13 14 17 16
 14 15 18 17
 16 17 20 19
 17 18 21 20
 19 20 23 22
 20 21 24 23
 22 23 26 25
 23 24 27 26
 25 26 29 28
 26 27 30 29
 28 29 32 31
 29 30 33 32
 31 32 35 34
 32 33 36 35
KNPR
1 1 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1
KMM
 36 33
```