UNIVERSITÄT
DES
SAARLANDES

Ein Bildeditor für Osmose und Diffusion

Masterarbeit

zur Erlangung des akademischen Grades
Master of Science
im Studiengang Angewandte Mathematik
der Naturwissenschaftlich-Technischen Fakultät I
- Mathematik und Informatik -
der Universität des Saarlandes

von

Nane Senta Neu

Saarbrücken, 30. Juli 2018

Angefertigt unter der Betreuung von
Prof. Dr. Joachim Weickert

Zweitgutachter
Prof. Dr. Thomas Schuster

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Saarbrücken, 30. Juli 2018

Nane Senta Neu

INHALTSVERZEICHNIS

NOTATIONEN UND ABKÜRZUNGEN

Die folgenden Bezeichnungen sind für die ganze Arbeit gültig:

$\mathbb{R}$	=	Der Körper der reellen Zahlen,
$[a,b]$	=	$\{x \in \mathbb{R} : a \leq x \leq b\}$, das abgeschlossene Intervall von a bis b,
(a,b)	=	$\{x \in \mathbb{R} : a < x < b\}$, das offene Intervall von a bis b,
$[a,b)$	=	$\{x \in \mathbb{R} : a \leq x < b\}$, das halboffene Intervall von a bis b,
$(a,b) \times (c,d)$	=	$\{(x,y) \in \mathbb{R}^2 : x \in (a,b), y \in (c,d)\}$,
$\langle \boldsymbol{x}, \boldsymbol{y} \rangle$	=	$\boldsymbol{x}^\top \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$, das Skalarprodukt von $\boldsymbol{x}$ und $\boldsymbol{y} \in \mathbb{R}^n$,
$\|\boldsymbol{x}\|$	=	$\sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$, die Norm des Vektors $\boldsymbol{x} \in \mathbb{R}^n$,
$\lfloor x \rfloor$	=	$\max_{k \in \mathbb{Z}, k \leq x} k$, die untere Gaußklammer,
$C^k(M)$	=	Raum der k-mal stetig differenzierbaren Funktionen von M nach $\mathbb{R}$,
$\frac{\partial}{\partial x_i} f(\boldsymbol{a})$	=	$\partial_{x_i} f(\boldsymbol{a}) = f_{x_i}(\boldsymbol{a}) := \lim_{h \to 0} \frac{f(a_1,\ldots,a_i+h,\ldots,a_n)-f(\boldsymbol{a})}{h}$, die partielle Ableitung von $f \in C^n(\mathbb{R}^n)$ nach x_i,
$\boldsymbol{\nabla} f$	=	$\left(\frac{\partial}{\partial x_1} f(\boldsymbol{a}), \ldots, \frac{\partial}{\partial x_n} f(\boldsymbol{a})\right)$, der Gradient von $f : \mathbb{R}^n \to \mathbb{R}$,
div $\boldsymbol{f}$	=	$\sum_{i=1}^{n} \frac{\partial}{\partial x_i} \boldsymbol{f}(\boldsymbol{a})$, die Divergenz von $f : \mathbb{R}^n \to \mathbb{R}^n$,
$\boldsymbol{\Delta}$	=	(div$\boldsymbol{\nabla}$), der Laplace Operator,
$\binom{n}{k}$	=	$\frac{n!}{(n-k)!\,k!}$, der Binomialkoeffizient von n über k,
ODE	=	gewöhnliche Differentialgleichung (ordinary differential equation),
PDE	=	partielle Differentialgleichung (partial differential equation).

EINLEITUNG

Die Forschungsarbeiten der mathematischen Bildanalyse und -verarbeitung beinhalten auch Computerprogramme, mit denen die aktuelle Forschung getestet und angewandt wird. Im Falle der Bildverarbeitung sind dies meist Konsolenanwendungen, die dem Benutzer erlauben, Grauwertbilder im PGM-Format bzw. Farbwertbilder im PPM-Format zu bearbeiten. Dabei geschieht die Interaktion mit dem Benutzer allein über die Konsole: Bilder einlesen, die entsprechenden numerischen Methoden auswählen, die dazugehörigen Parameter einstellen und das Endresultat abspeichern. Ist man mit dem Resultat nicht zufrieden oder möchte einfach verschiedene Einstellungen testen, so muss das Programm jedes Mal neu gestartet und alle Einstellungen erneut vorgenommen werden. Um beispielsweise ein Szenario zu testen, bei dem man nur betrachten will, was die Änderung eines einzelnen Parameters bewirkt, bedeutet diese Vorgehensweise einen Mehraufwand im Vergleich zu einem System, das es erlaubt, tatsächlich nur die Parameter neu einzugeben, die verändert werden sollen. Dazu kommt, dass die volle Ausschöpfung dieser Programme den Programmieren sowie denen vorbehalten ist, die sich tagtäglich mit der dahinterliegenden Theorie auseinandersetzen. In vielen Fällen ist das ausreichend, zum Beispiel dann, wenn es darum geht, Forschungsergebnisse darzustellen.
Bei manchen Verfahren, wie etwa der Osmose, ist es von Vorteil, die Bearbeitung einfacher zu gestalten. Osmose hat für eine allgemeine Nutzerschaft interessante Anwendungsgebiete, wie das Reduzieren und Entfernen von Schatten und dem nahtlosen Ineinanderkopieren von Bildern.
Hier hakt diese Arbeit ein. Der Schwerpunkt dieser Arbeit liegt in der Entwicklung eines Bildeditors für Osmose und Diffusion. Die Benutzung der Bildverarbeitungsalgorithmen wird hierbei durch einfache Dateneingabe und Parametereinstellung sowie eine grafische Benutzeroberfläche erleichtert.
Der Editor wurde so konzipiert, dass er von zwei Benutzergruppen verwendet werden kann. Zum einen von einem sogenannten *Practitioner*, dem es egal ist, welche Algorithmen und numerischen Methoden im Hintergrund ablaufen und der an einem Resultat in einem guten Schnelligkeits-Qualitäts-Verhältnis interessiert ist. Zum anderen von einem sogenannten *Expert*, der sich mit den theoretischen Eigenschaften auskennt und daher mehr Einstellungsfreiheiten bei der Wahl der numerischen Methode und den dazugehörigen Parametern

genießen möchte.
Die Verfahren, die der Editor anbietet, reichen von linearer Diffusion und homogenem diffusionsbasiertem Inpainting zu osmotischer Schattenreduktion und nahtlosem Ineinanderkopieren zweier Bilder.
Es können Bilder in verschiedenen Formaten eingelesen werden (PNG, JPG, BMP, PPM und PGM) und in den PNG- und PPM-Formaten abgespeichert werden. Darüber hinaus gibt es auch die Möglichkeit, mit einem Pinsel in die Bilder zu zeichnen und dadurch ganz eigene Kreationen zu erstellen.

Der schriftliche Teil der Arbeit liegt hiermit vor; während der Quellcode und die ausführbaren Dateien des Editors für Linux und Windows auf der beigelegten CD zu finden sind.
In den ersten beiden Kapiteln werden die Prozesse der Diffusion und Osmose erklärt und ausgehend von diesem physikalischen Verständnis jeweils ein Analogon zur mathematischen Bildverarbeitung gezogen. Es werden verschiedene Anwendungen der linearen Diffusion und Osmose beschrieben. Darunter fallen der homogene Diffusionsfilter und diffusionsbasiertes Inpainting sowie Schattenreduktion und nahtloses Ineinanderkopieren von zwei Bildern mittels Osmose.
Kapitel 3 befasst sich damit, die kontinuierliche Theorie in eine diskrete Theorie umzuwandeln. Dabei werden Diskretisierungen für die beiden Prozesse angegeben und die Verfahren vorgestellt, die in dem Editor verwendet werden können. Im Fall der Diffusion sind das das explizite Schema und zwei explizite Beschleunigungen: FED und FSI. Im Fall der Osmose wird zusätzlich zum explizten auch ein implizites Schema angeboten, das das entstehende Gleichungssystem mit Hilfe der BiCGStab-Methode löst.
Zum Schluss wird in Kapitel 4 das Framework Qt vorgestellt, das für die Entwicklung des Editors genutzt wurde. Im gleichen Zuge wird die grafische Oberfläche des Editors beschrieben und seine Benutzung erklärt.

Danksagung

An dieser Stelle möchte ich allen danken, die mich während meines Studiums unterstützt und begleitet haben.
Mein besonderer Dank gilt dabei Prof. Dr. Joachim Weickert für das interessante Thema und die intensive Betreuung und Anleitung während der Entstehungsphase dieser Masterarbeit. Außerdem bedanke ich mich herzlich bei meinen Eltern, die mir zur Seite standen, durch finanzielle Unterstützung das Mathematikstudium ermöglichten und denen vor allem aufgefallen ist, dass auch mein zweiter Vorname auf der Arbeit stehen sollte.
Ein zusätzliches großes Dankeschön geht an alle, die mich bei der Erstellung dieser Arbeit unterstützt haben.

1 GRUNDLAGEN

In diesem Kapitel werden die physikalisch-chemischen Prozesse der Diffusion und Osmose behandelt und ihre Analogien in der digitalen Bildbearbeitung dargestellt.

1.1 Diffusion

1.1.1 Definition

Diffusion (*lat. diffundere »ausströmen«*):
„Unter Diffusion versteht man den Transportvorgang, in deren Verlauf Teilchen infolge ihrer Wärmebewegung [...] von Orten höherer Konzentration zu Orten niedriger Konzentration gelangen, sodass allmählich ein Konzentrationsausgleich stattfindet.“ ([11], Seite 115).

Somit beschreibt Diffusion einen Prozess, bei dem Konzentrationsunterschiede ausgeglichen werden. Er wird mit Hilfe des *1. Fick'schen Gesetzes* beschrieben:

$$\boldsymbol{j} = -\boldsymbol{D} \cdot \boldsymbol{\nabla} u. \tag{1.1}$$

Dabei erzeugt das *Konzentrationsgefälle* $\boldsymbol{\nabla} u := (\partial_x u, \partial_y u)^\top$ den *Diffusionsfluss* $\boldsymbol{j} = (j_1, j_2)^\top$. Das Verhältnis, in dem die beiden zueinander stehen, wird durch den *Diffusionstensor* $\boldsymbol{D}$, einer symmetrischen, positiv definiten 2×2 Matrix, angegeben. Es handelt sich um *isotrope* Diffusion, wenn $\boldsymbol{j}$ und $\boldsymbol{\nabla} u$ parallel zueinander sind, $\boldsymbol{D}$ kann dann durch eine positive, skalarwertige *Diffusivität* g ersetzt werden. Im allgemeinen Fall, wenn $\boldsymbol{j}$ und $\boldsymbol{\nabla} u$ nicht parallel sind, spricht man von *anisotroper* Diffusion mit Diffusionstensor $\boldsymbol{D}$. Ist $\boldsymbol{D}$ über die gesamte Bildfläche konstant, so spricht man von *homogener* Diffusion und von *inhomogener* Diffusion, falls $\boldsymbol{D}$ nicht konstant ist [17].
Da der Diffusionsprozess einen reinen Transportvorgang beschreibt, der nur die vorhandenen Teilchen miteinander vermischt, ohne neue Teilchen hinzuzufügen oder zu entfernen, bleibt die Gesamtheit der Teilchen während des

Prozesses gleich. Dadurch ergibt sich die *Kontinuitätsgleichung*

$$\partial_t u = -\text{div } \boldsymbol{j}, \tag{1.2}$$

wobei t die Zeit und $\text{div}\boldsymbol{j} = \partial_x j_1 + \partial_y j_2$ die *Divergenz* von $\boldsymbol{j}$ beschreibt. Zusammen mit (1.1) ergibt sich die *Diffusionsgleichung*

$$\partial_t \boldsymbol{u} = \text{div}(\boldsymbol{D} \cdot \boldsymbol{\nabla} u), \tag{1.3}$$

welche in der Physik auch als *Wärmeleitungsgleichung* bekannt ist.

Um diese physikalische Definition nun auf die digitale Bildverarbeitung anzuwenden, wird vorerst angenommen, dass ein Grauwertbild mit Pixelbreite N und Pixelhöhe M bearbeitet werden soll. Der Definitionsbereich ist somit die Bildfläche, ein rechteckiges Gebiet $\Omega := [0, N) \times [0, M) \subset \mathbb{R}^2$. Das Bild selbst wird als eine beschränkte Funktion $f : \Omega \to \mathbb{R}$ beschrieben, die Werte im Bereich $[0, 255]$ annimmt. Dabei werden die Grauwerte eines Bildes als Konzentrationen interpretiert. Der Diffusionsfilter berechnet die gefilterte Version $u(x, y, t)$ vom Bild $f(x, y)$ als Lösung der Diffusionsgleichung (1.3) mit

$$u(x, y, 0) = f(x, y) \tag{1.4}$$

als Anfangsbedingung. Es werden gespiegelte Randbedingungen, sogenannte *homogene Neumann Bedingungen* angenommen, d.h. es gilt $\partial_{\boldsymbol{n}} u = 0$ für einen Normalenvektor $\boldsymbol{n}$ am Bildrand $\partial\Omega$, dabei ist $\partial_{\boldsymbol{n}} := \boldsymbol{n}^\top \boldsymbol{\nabla} \boldsymbol{n}$.
Es wird auch $\boldsymbol{x} := (x, y) \in \Omega$ geschrieben.

In [17] beschreibt Weickert das Konzept der Skalenräume. Er erklärt die Äquivalenz von Gauß'schen Skalenräumen zu linearer Diffusion: Ein Gauß-Filter mit Varianz σ^2 ist äquivalent zu einem linearen Diffusionsfilter mit Stoppzeit $T = \frac{1}{2}\sigma^2$. Daraus ergeben sich folgende theoretische Eigenschaften für das Diffusionsmodell:

1. **Gutgestelltheit und Regularität:** Es existiert eine eindeutige Lösung $u(\boldsymbol{x}, t)$ mit $u \in C^\infty(\Omega \times [0, \infty))$ und $u(., t)$ hängt stetig von f bzgl. der L^2-Norm ab.

2. **Erhaltung des mittleren Grauwerts:**
   Da der Diffusionsprozess in Divergenzform vorliegt erhält er den mittleren Grauwert μ_f des Anfangsbildes f :

$$\frac{1}{|\Omega|} \int_\Omega u(\boldsymbol{x}, t) d\boldsymbol{x} = \frac{1}{|\Omega|} \int_\Omega f(\boldsymbol{x}) d\boldsymbol{x} =: \mu_f \quad \forall t > 0. \tag{1.5}$$

3. **Maximum-Minimum-Prinzip:**
   Es genügt einem *Maximum-Minimum-Prinzip* :

$$\inf f \leq u(\boldsymbol{x}, t) \leq \sup f \qquad \forall \boldsymbol{x} \in \Omega, \quad \forall t > 0. \tag{1.6}$$

4. **Lyapunov Funktionale**

$$V(t) := \int_{\Omega} r(u(\boldsymbol{x}, t))d\boldsymbol{x} \tag{1.7}$$

ist eine Lyapunov Funktion für alle konvexen Funktionen $r \in C^2(\mathbb{R})$, d.h. $V(t)$ ist eine fallende, nach unten beschränkte Funktion.

5. **Konvergenz gegen ein konstantes Bild:**
$u(\boldsymbol{x}, t)$ konvergiert gegen den mittleren Grauwert von f:

$$\lim_{t\to\infty} u(\boldsymbol{x}, t) = \mu_f. \tag{1.8}$$

1.1.2 Beispiele

Es gibt verschiedene Möglichkeiten, den Diffusionstensor $\boldsymbol{D}$ bzw. die Diffusivität g zu wählen.
Um Rauschen aus einem Bild zu entfernen, wird g so gewählt, dass an den Kanten des Originalbildes weniger diffundiert wird als in den homogenen Flächen des Bildes. Ein gutes Beispiel hierfür ist die *Perona-Malik-Diffusivität*

$$g(|\boldsymbol{\nabla} u|^2) := \frac{1}{1 + \frac{|\nabla u|^2}{\lambda^2}}, \tag{1.9}$$

wobei λ als Kontrastparameter gilt, d.h. die Stellen, an denen $|\boldsymbol{\nabla} u| > \lambda$ gilt werden vom Algorithmus als Kanten wahrgenommen, da dort große Kontrastunterschiede herrschen. Der Diffusionsprozess wird dort deutlich geschwächt und die Kante bleibt auch in der gefilterten Version erhalten. Abbildung 1.1 zeigt, wie sich der *Perona-Malik-Filter* verhält.

Ein Beispiel, das statt einer skalarwertigen Diffusivität g den Diffusionstensor $\boldsymbol{D}$ nutzt, findet man in der anisotropen Diffusion. Abbildung 1.2 zeigt die gefilterten Versionen zweier verschiedener nichtlinearer, anisotroper Diffusionsprozesse: *Mean-curvature motion* und *Coherence-enhancing anisotropic diffusion.*

Um die nun angerissenen verschiedenen Arten der Diffusionsfilter auf einen Blick zu zeigen und ihre Unterschiede herauszustellen, zeigt Abbildung 1.3 die Auswirkungen von linearer und nichtlinearer isotroper sowie nichtlinearer anisotroper Diffusion auf ein Ausgangsbild, das aus einem Dreieck und einem Rechteck besteht und zu 70 % durch Rauschen gestört wurde. Lineare Diffusion kann zwar das Rauschen komplett entfernen, verwischt aber auch die Kanten der geometrischen Figuren. Es ist nicht klar zu sagen, wo genau die ursprünglichen Kanten verlaufen. Nichtlineare isotrope Diffusion (g wird

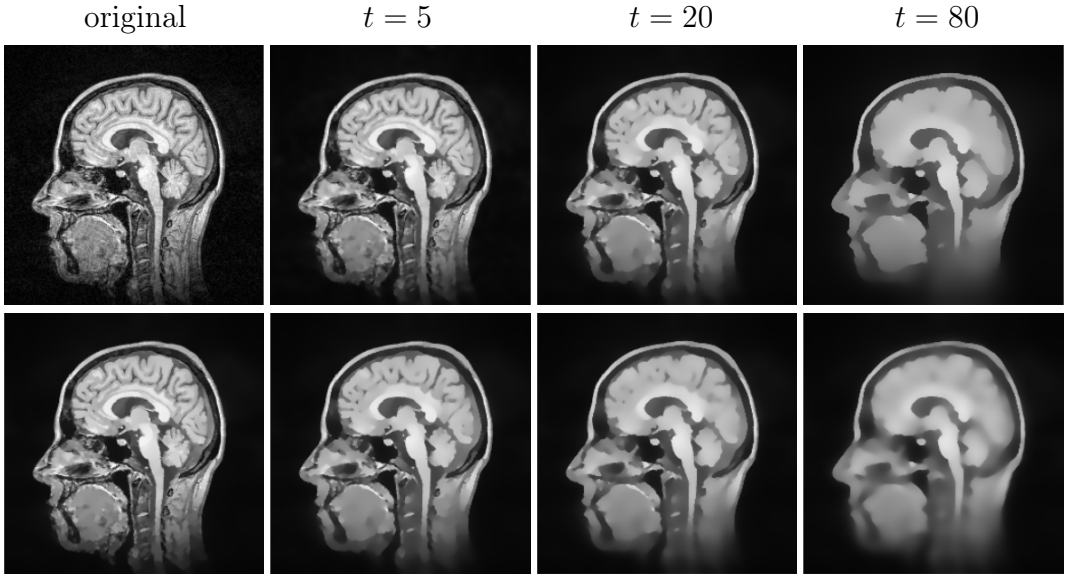

Abb. 1.1. Beispiel des Diffusionsvorganges anhand der Perona-Malik-Diffusivität: Die obere Reihe zeigt den Einfluss der Zeit auf das Original-Bild ($\lambda = 4$), die untere Reihe den des Kontrastparameters λ ($t = 20$) [15].

durch die Perona-Malik Diffusivität (1.9) beschrieben) hingegen belässt die ursprünglichen Kanten in annähernd ihrem Originalzustand. Allerdings hat die an den Kanten herabgesetzte Diffusion den Nachteil, dass diese immer noch stark verrauscht sind. Nichtlineare anisotrope Diffusion besitzt die Vorteile der beiden vorher genannten Verfahren. Sie kombiniert die gute Rauschreduktion der linearen Diffusion mit der Kantenerhaltung der nichtlinearen isotropen Diffusion. Allerdings werden Ecken mehr abgerundet als im nichtlinearen isotropen Fall, da anisotrope Diffusion das Bild entlang der Kanten glättet [15].

Obwohl nichtlineare (an)isotrope Diffusion interessante und spannende Anwendungsgebiete eröffnen, beschränkt sich die vorliegende Arbeit auf den Einfachsten der Diffusionsvorgänge: Lineare Diffusion mit Diffusionstensor $\boldsymbol{D} = \boldsymbol{I}$. Damit gilt für die *Diffusionsgleichung* (1.3)

$$\begin{aligned} \partial_t u &= \operatorname{div}(\boldsymbol{\nabla} u) \\ &= \boldsymbol{\Delta} u \\ &= u_{xx} + u_{yy}, \end{aligned} \tag{1.10}$$

wobei $\boldsymbol{\Delta}$ den *Laplace-Operator* beschreibt [17].

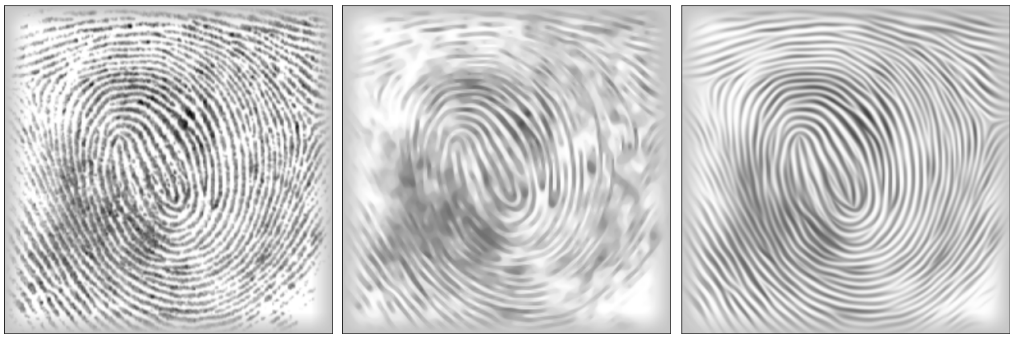

Abb. 1.2. Beispiel für Anisotrope Diffusion: **Links**: Original. **Mitte**: Anisotrope Diffusion mittels *Mean-curvature motion*, $t = 5$. **Rechts**: *Coherence-enhancing anisotropic diffusion*, $\sigma = 0.5, \rho = 4, t = 20$ [17].

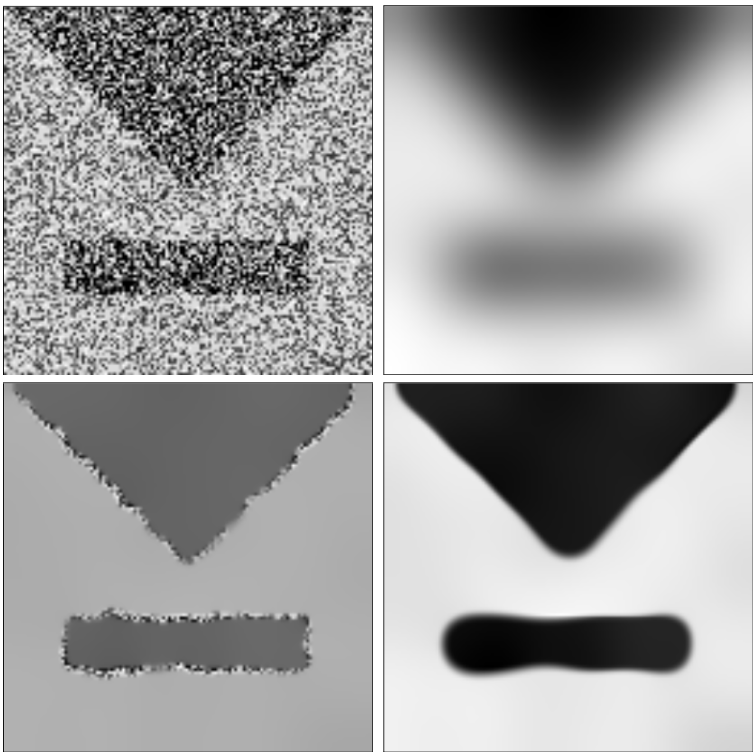

Abb. 1.3. Beispiel für verschiedene Diffusivitäten: **Oben Links**: Original. **Oben Rechts**: Lineare Diffusion. **Unten Links**: Nichtlineare isotrope Diffusion. **Unten Rechts**: Nichtlineare anisotrope Diffusion [17].

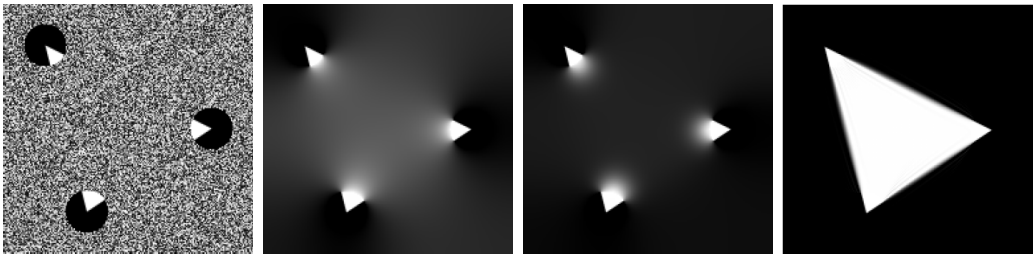

Abb. 1.4. Beispiel: Diffusionsbasiertes Inpainting mit folgenden Diffusionstensoren: **Von links nach rechts**: Startbild; Linear isotrop: $\boldsymbol{D} = \boldsymbol{I}$; Nichtlinear isotrop $\boldsymbol{D} = g(|\boldsymbol{\nabla} u|^2)\boldsymbol{I}$; Nichtlinear anisotrop: $D(|\boldsymbol{\nabla} u|^2)$ [8].

1.1.3 Diffusionsbasiertes Inpainting

In der Bildkompression wird *diffusionsbasiertes Inpainting* genutzt, um ein Bild von einer sehr kleinen Menge an bekannten Daten zu rekonstruieren. Diese Daten liegen in der Teilmenge $K \subset \Omega$, die auch *Inpainting-Maske* genannt wird. Während des Diffusionsvorgangs breiten sich die bekannten Daten auf den noch unbekannten Teil des Bildes aus, während die Daten in der Maske unverändert bleiben. Diese Ausbreitung wird durch

$$\begin{aligned} u(\boldsymbol{x}) &= f(\boldsymbol{x}) && \text{falls } \boldsymbol{x} \in \Omega, \\ \partial_t u &= \operatorname{div}(\boldsymbol{D}\boldsymbol{\nabla} u) && \text{falls } \boldsymbol{x} \in \Omega \backslash K \end{aligned} \tag{1.11}$$

beschrieben. Da das Bild f möglichst genau rekonstruieren werden soll, ist man an dem stätionären Zustand dieses Gleichungssystems interessiert. Dieser erfüllt $\partial_t u = 0$, womit (1.11) auch als Interpolation

$$\chi_K(\boldsymbol{x}) \cdot (u(\boldsymbol{x}) - f(\boldsymbol{x})) - (1 - \chi_K(\boldsymbol{x})) \cdot \operatorname{div}(\boldsymbol{D}\boldsymbol{\nabla} u) = 0 \tag{1.12}$$

mit der charakteristischen Funktion χ_K:

$$\chi_K(\boldsymbol{x}) = \begin{cases} 1, & \text{falls } \boldsymbol{x} \in K, \\ 0, & \text{falls } \boldsymbol{x} \notin K, \end{cases} \tag{1.13}$$

beschrieben werden kann. Die Lösung u genügt somit $u = f$ in den gegebenen Pixeln und $\operatorname{div}(\boldsymbol{D}\boldsymbol{\nabla} u) = 0$ in den unbekannten Pixeln.
Abbildung 1.4 zeigt diffusionsbasiertes Inpainting mit verschiedenen Diffusionstensoren. Die Inpainting-Maske ist dabei durch die nicht-verrauschten Pixel im Startbild gegeben. Man sieht, dass der Diffusionstensor eine wichtige Rolle beim Inpainting spielt. In den beiden isotropen Fällen geht die angedeutete Struktur der Maske fast komplett verloren, wobei es im nichtlinearen Fall zu erahnen ist, wie die Struktur aussehen könnte. Das beste Ergebnis liefert die nichtlineare anisotrope Diffusion mit dem Diffusionstensor für *Edge-enhancing*

anisotropic diffusion. Dieser Tensor adaptiert die Struktur, die gegeben ist und führt sie weiter, was in dem gewählten Beispiel ein optimales Ergebnis bewirkt. Eine gut gewählte Maske sowie ein gut gewählter Diffusionstensor reichen also aus, um sinnvolle Bildkompression zu betreiben. Da man nur die Bilddaten in den Maskenpunkten benötigt, reicht es aus, diese zu speichern und anzugeben, mit welchem Diffusionstensor die Rekonstruktion durchgeführt wird. Schwieriger ist es, sinnvolle Maskenpunkte zu finden, die eine möglichst gute Rekonstruktion ermöglichen. [8].

Der Editor dieser Arbeit ermöglicht diffusionsbasiertes Inpainting mit Diffusionstensor $\boldsymbol{D} = \boldsymbol{I}$.

1.2 Osmose

1.2.1 Definition

Osmose (*griech. osmos »Stoß«*):
„Unter Osmose versteht man einen auf das Lösungsmittel beschränkten Diffusionsvorgang, der auftritt, wenn zwei gleichartige Lösungen unterschiedlicher Konzentration durch eine semipermeable Membran getrennt sind. Durch diese können nur Moleküle des Lösungsmittels von einer Lösung in die andere diffundieren; der gelöste Stoff, dessen Moleküle zu groß sind, wird zurückgehalten. Um einen Konzentrationsausgleich zu bewirken, diffundieren mehr Lösungsmittelmoleküle in den Bereich höherer Konzentration als umgekehrt. [...]“ ([11], Seite 327).

Somit gilt auch Osmose als ein Diffusionsprozess; der Konzentrationsaustausch geht allerdings gerichtet vonstatten. Sie beschreibt einen Transportvorgang durch eine semipermeable (halbdurchlässige) Membran, sodass in ihrem stationären Zustand die Flüssigkeiten zu beiden Seiten der Membran verschiedene Konzentrationen haben. Man kann deswegen auch von unsymmetrischer Diffusion sprechen.

Um das Gleichgewicht der Osmose zu beschreiben, muss erst bestimmt werden, wie der Fluss der Konzentrationen dargestellt wird. Die semipermeable Membran wird durch ein *Fluss-Vektorfeld* $\boldsymbol{d}$ beschrieben, das die Osmose steuert. Der Fluss $\boldsymbol{j}$ setzt sich dabei wie folgt zusammen:
Aus dem *1. Fick'schen Gesetz*

$$\boldsymbol{j}_{dif} = -\boldsymbol{D} \cdot \boldsymbol{\nabla} u \tag{1.14}$$

und dem Fluss-Vektorfeld $\boldsymbol{d}$, der zu dem Konvektionsterm

$$\boldsymbol{j}_{konv} = \boldsymbol{d} \cdot u \tag{1.15}$$

führt. Zusammen ergibt sich

$$\begin{aligned} \boldsymbol{j} &= \boldsymbol{j}_{dif} + \boldsymbol{j}_{konv} \\ &= -\boldsymbol{D} \cdot \boldsymbol{\nabla} u + \boldsymbol{d} \cdot u. \end{aligned} \tag{1.16}$$

Da Osmose als Abwandlung eines Diffusionsprozesses einen reinen Transportprozess beschreibt und daher die Gesamtheit der Teilchen nicht beeinflusst, gilt auch hier die Kontinuitätsgleichung (1.2) und es ergibt sich die *Drift-Diffusionsgleichung* (auch *Konvektions-Diffusionsgleichung* genannt)

$$\partial_t u = \operatorname{div}(\boldsymbol{D} \cdot \boldsymbol{\nabla} u - \boldsymbol{d} \cdot u). \tag{1.17}$$

Betrachtet man wie im vorliegenden Fall einen linearen Osmoseprozess, wird aus dem Diffusionstensor $\boldsymbol{D}$ der Einheitsoperator und (1.17) wird zu

$$\partial_t u = \boldsymbol{\Delta} u - \operatorname{div}(\boldsymbol{d} \cdot u). \tag{1.18}$$

In der statistischen Physik nennt man diese Gleichung auch die *Fokker-Planck-Gleichung*, in der Wahrscheinlichkeitstheorie die *Kolmogorov-Vorwärtsgleichung*.

Auch diese physikalisch-chemische Definition kann auf die digitale Bildverarbeitung angewendet werden. Wie oben wird dabei die semipermeable Membran durch das Vektorfeld $\boldsymbol{d}$ dargestellt, welches für jedes Pixel im Bild angibt, in welche Richtung es „*fließen*“ darf. Dieses Vektorfeld heißt von nun an *Drift-Vektorfeld*.

Analog zum Diffusionsvorgang entsprechen auch hier die Grauwerte den Konzentrationen. Das Bild muss allerdings ein positives Bild $f : \Omega \to \mathbb{R}^+$ sein. Die Bildfunktion f darf also nur positive Werte annehmen, also kann es gegebenenfalls von Nöten sein, das Originalbild zu skalieren, z. B. von $[0, 255] \to [1, 256]$. Die semipermeable Membran wird durch das Drift-Vektorfeld $\boldsymbol{d} : \Omega \to \mathbb{R}^2$ dargestellt, wobei hier für die Randvektoren $\boldsymbol{n}^\top \mathbf{d} = 0$ auf $\partial\Omega$ gilt. Der Osmosefilter berechnet analog zum Diffusionsfilter das gefilterte Bild $u(x, y, t)$ durch $f(x, y)$ als Lösung der *Drift-Diffusionsgleichung* (1.18) auf $\Omega \times (0, T)$ mit

$$u(x, y, 0) = f(x, y) \tag{1.19}$$

als Anfangsbedingung und homogenen Neumann Bedingungen $\partial_{\boldsymbol{n}} u = 0$ auf $\partial\Omega$ für eine Normale $\boldsymbol{n}$.

Dieses Modell bringt folgende Eigenschaften mit sich [14]:

1. **Erhaltung des mittleren Grauwerts:**
   Wie bei der Diffusion liegt der Osmoseprozess in Divergenzform vor und hält deswegen den mittleren Grauwert μ_f des Anfangsbildes f aufrecht:

$$\frac{1}{|\Omega|} \int_\Omega u(\boldsymbol{x}, t) d\boldsymbol{x} = \frac{1}{|\Omega|} \int_\Omega f(\boldsymbol{x}) d\boldsymbol{x} =: \mu_f \quad \forall t > 0. \tag{1.20}$$

2. **Erhaltung der Positivität:**
   Obwohl es für den Osmoseprozess kein *Maximum-Minimum-Prinzip* gibt, da er dagegen verstoßen kann, bleibt seine Lösung für alle Zeiten t positiv:

$$u(\boldsymbol{x}, t) > 0 \qquad \forall \boldsymbol{x} \in \Omega, \quad \forall t > 0. \tag{1.21}$$

3. **Konvergenz gegen einen nichttrivialen stationären Zustand:**
   Der einzige Unterschied zwischen homogener Diffusion und kontinuierlicher linearer Osmose ist das Drift-Vektorfeld $\boldsymbol{d}$. Dieses Vektorfeld ist das Werkzeug, welches die Konvergenz der Osmose steuert: Genügt $\boldsymbol{d}$

$$\boldsymbol{d} = \boldsymbol{\nabla}(\ln v) = \frac{\boldsymbol{\nabla} v}{v} \tag{1.22}$$

   für ein positives Bild v, so konvergiert der Osmoseprozess gegen v bis auf eine multiplikative Konstante, die die Erhaltung des mittleren Grauwerts des Anfangsbildes f sichert. Also kreiert Osmose nichttriviale stationäre Zustände. Dies ist ein fundamentaler Unterschied zur linearen Diffusion, die nur triviale, also einfarbige Bilder als stationäre Zustände liefert.

Im Gegensatz zur Diffusion ist der stationäre Zustand der Osmose ein nichttriviales (nicht-konstantes) Bild. Dieser Zustand wird fast ausschließlich durch das Drift-Vektorfeld $\boldsymbol{d}$ bestimmt. Vom ursprünglichen Anfangsbild f wird nur der mittlere Grauwert μ_f übernommen. Da das Drift-Vektorfeld eine so große Rolle beim Modellieren der Osmosefilter spielt, wird es im nächsten Kapitel genauer untersucht. Dabei werden auch die Hauptanwendungsgebiete der Osmose in der digitalen Bildverarbeitung herausgestellt.

1.3 Bemerkungen

1.3.1 Farbbilder

Obwohl im Vorangegangenen nur von Grauwertbildern gesprochen wurde, kann man sowohl bei der Diffusion als auch bei der Osmose die genannten Verfahren auch auf Farbbilder anwenden. Ein Farbbild wird als Funktion $f : \Omega \to \mathbb{R}^3$ interpretiert, dabei beschreibt jede Komponente einen Kanal im RGB-Farbraum. Um Farbbilder zu filtern, wendet man im linearen Fall den Filter auf jeden Kanal separat an.
Im nichtlinearen Fall ist diese separate Anwendung nicht die optimale Lösung, da die Kanäle an unterschiedlichen Stellen verschiedene Strukturelemente besitzen können. Daher werden für nichtlineare isotrope Diffusion *gekoppelte Diffusivitäten* und für nichtlineare anisotrope Diffusion *gekoppelte Diffusionstensoren* verwendet. Es ergeben sich die Diffusionsgleichungen

$$\partial_t u_i = \operatorname{div}(\boldsymbol{G} \cdot \boldsymbol{\nabla} u_i), \qquad \forall i \in \{1, \ldots, m\} \tag{1.23}$$

mit m Kanälen und der gekoppelten Diffusivität $\boldsymbol{G}$. Bei RGB-Bildern gilt $m = 3$. Gerig et al liefern in [3] mit $\boldsymbol{G} = g\left(\sum_{k=1}^{m} |\boldsymbol{\nabla} u_k|\right)$ ein Beispiel für die gekoppelte Diffusivität im isotropen Fall. Dabei synchronisiert $\sum_{k=1}^{m} |\boldsymbol{\nabla} u_k|$ die Evolution in allen Kanälen. Ein Beispiel für den anisotropen Fall wird von Weickert in [16] mit $\boldsymbol{G} = g\left(\sum_{k=1}^{m} \boldsymbol{\nabla} u_k \, \boldsymbol{\nabla} u_k^{\top}\right)$ gegeben.

1.3.2 Implementierung und Diskretisierung

In diesem Kapitel wurde nur die kontinuierliche Formulierung der Diffusion und Osmose erläutert. Da die Bilder, die bearbeitet werden, in einer digitalen, also einer diskreten Umgebung vorliegen, werden in Kapitel 3 explizite und implizite Diskretisierungen für Diffusion sowie Osmose behandelt.

2 ANWENDUNGEN DER OSMOSE

In diesem Kapitel geht es darum wie das Drift-Vektorfeld den Osmoseprozess steuert. Es wird erst allgemein ein Modell beschrieben, wie man zu einem gegebenem Bild ein Vektorfeld findet, das im stationären Zustand das Originalbild wiedergibt; dann werden Veränderungen an den Drift-Vektoren vorgenommen, die verschiedene Anwendungsgebiete ermöglichen wie zum Beispiel das Entfernen von Schatten aus einem Bild oder nahtloses Ineinanderkopieren von zwei verschiedenen Bildern. Schließlich wird eine Möglichkeit gezeigt, wie man mittels Osmose effiziente Bildkompression betreiben kann.

2.1 Kanonische Drift-Vektoren

In Kapitel 1 wurde eine Theorie für die kontinuierliche lineare Osmose beschrieben. Es wurde gezeigt, dass der Osmose Prozess gegen einen nichttrivialen stationären Zustand konvergiert. Genügt das Drift-Vektorfeld $\boldsymbol{d}$ der Gleichung

$$\boldsymbol{d} = \boldsymbol{\nabla}(\ln v) = \frac{\boldsymbol{\nabla} v}{v} \tag{2.1}$$

für ein positives Bild v, so konvergiert der Prozess gegen v bis auf den mittleren Grauwert. Deswegen wird v als *Steuerungsbild* und $\boldsymbol{d}[v] := \frac{\boldsymbol{\nabla} v}{v}$ als sein *kanonisches Drift-Vektorfeld* beschrieben. Somit gibt $\boldsymbol{d}[v]$ vor, wie der stationäre Zustand einer osmotischen Evolution aussieht.
Abbildung 2.1 veranschaulicht diesen Sachverhalt anhand des RGB-Bildes des Mandrills. Nutzt man den Vorgang für Farbbilder, wird für jeden der drei RGB-Kanäle ein separates Drift-Vektorfeld berechnet. Als Anfangsbild wird ein konstantes Bild genommen, welches die mittleren Grauwerte des Mandrill-Bildes in jedem Kanal besitzt.

Interessant wird Osmose dann, wenn man Veränderungen am Drift-Vektorfeld vornimmt. Dies kann zum Beispiel geschehen, indem man die Vektoren an manchen Stellen auf Null setzt oder die kanonischen Drift-Vektoren von zwei

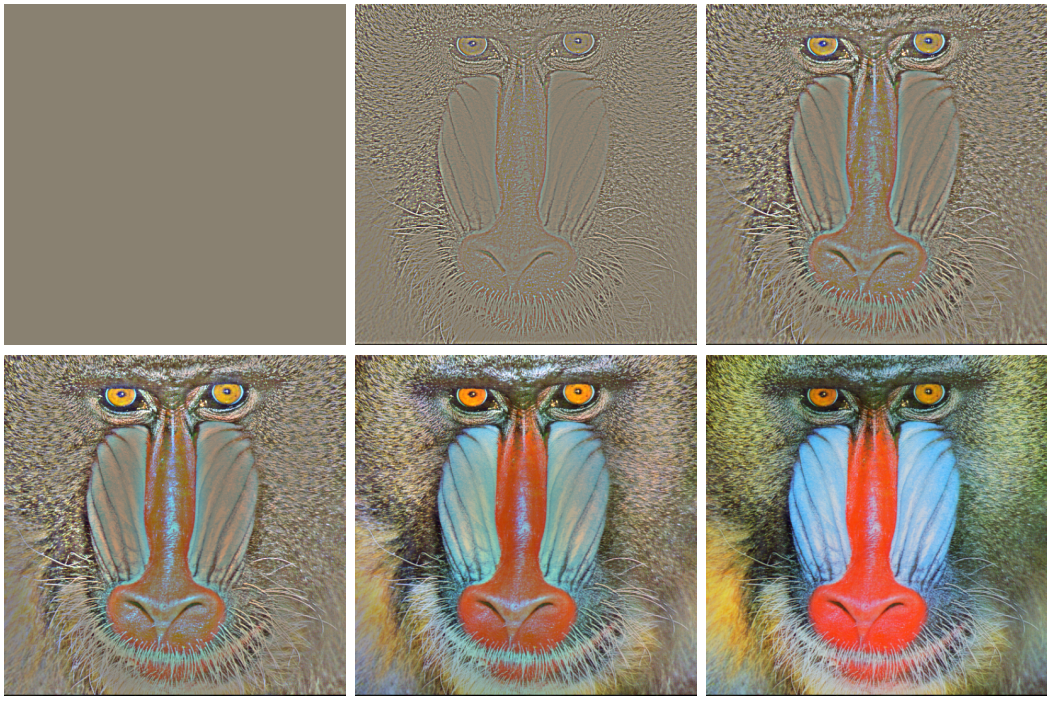

Abb. 2.1. Konvergenz gegen ein gegebenes Bild v. Das Originalbild (512 x 512 Pixel) ist durch ein konstantes Bild mit den mittleren Grauwerten des Mandrill-Bildes in jedem RGB-Kanal gegeben. Das kanonische Drift-Vektorfeld wurde für jeden Kanal separat aus den Kanälen des Mandrill-Bildes berechnet. **Von oben links nach unten rechts**: $t = 0;\ 2,72;\ 15,3;\ 69,1;\ 1140;\ 250.000$ [18].

verschiedenen Bildern verwendet. Solche Veränderungen ergeben im Allgemeinen nicht-integrierbare Daten, aber trotzdem wird der Osmoseprozess einen stationären Zustand erzeugen, der versucht, einen möglichst guten Kompromiss zwischen den auftretenden Konflikten darzustellen. Dies führt zu den folgenden Anwendungsgebieten der Osmose.

2.2 Schattenreduktion

In [18] wird beschrieben, wie Schatten aus einem Bild durch Osmose entfernt werden können. Im Allgemeinen sind Schatten multiplikative Änderungen der ursprünglichen Bildfarbe. Da Osmose invariant unter multiplikativen Grauwert-Skalierungen ist, haben Schatten in einem Bild nur an ihren Grenzen Auswirkungen auf die Drift-Vektoren des Bildes. Setzt man die Drift-Vektoren an den Schattengrenzen auf Null, reduziert man innerhalb der Grenzen die Schatten. Wählt man dabei die Stoppzeit groß genug, so wird der Schatten

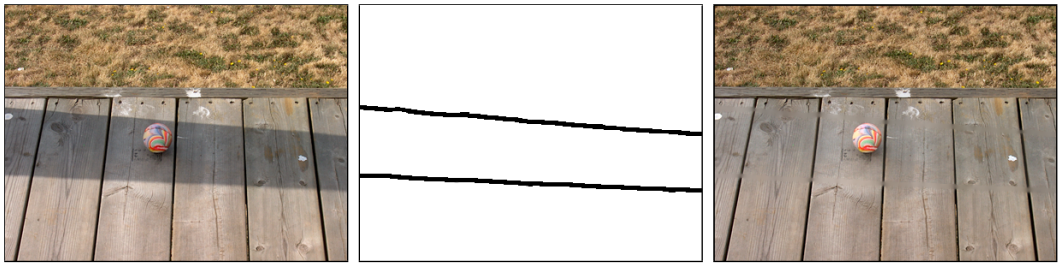

Abb. 2.2. Schatten Entfernung. **Oben links**: Original Bild (400 x 299 Pixel) [18]. **Oben rechts**: Selbst definierte Schattengrenzen. **Unten**: Osmotische Rekonstruktion mit dem Original Bild als Anfangsbild mit dem Programm dieser Arbeit.

komplett entfernt. An den Stellen, an denen der Drift-Vektor auf Null gesetzt wurde, wird homogene Diffusion ausgeführt. Aus diesem Grund sollte man die Schattengrenzen nicht zu breit eintragen, denn dann ergibt sich ein Bild, was an den Grenzen sehr unscharf ist, während der Rest des Bildes die ursprüngliche Schärfe behält.

In Abbildung 2.2 sieht man, wie Osmose Schatten aus einem Bild entfernt. Dabei fällt auf, dass die Rekonstruktion in den Regionen, die nicht vom Schatten betroffen sind, dunkler aussieht als das Original. Dies liegt daran, dass Osmose den mittleren Grauwert des Originals erhält und dieser aufgrund des noch vorhandenen Schattens niedriger ist als der gewünschte mittlere Grauwert der Rekonstruktion ohne Schatten. Dies kann behoben werden, indem man die Rekonstruktion so skaliert, dass die Farbwerte in der Region, die nicht vom Schatten betroffen ist, in beiden Bildern übereinstimmen.

2.3 Nahtloses Ineinanderkopieren von Bildern

Das zweite Beispiel, das Weickert et al. in [18] anführen, um zu zeigen, wie man Osmose mit veränderten Drift-Vektoren anwenden kann, findet sich im nahtlosen Ineinanderkopieren von zwei Bildern. Dabei möchte man einen oder mehrere Teile aus einem Bild in ein anderes kopieren, ohne dass es eine harte Kante oder einen sichtbaren Übergang gibt. In Abbildung 2.3 wird das Problem dargestellt: Zwei Bilder f_1 und f_2 sollen in solcher Art und Weise ineinander kopiert werden, sodass f_2 Bildinformationen in f_1 ersetzt. Dabei liegt das Originalbild f_1 in der rechteckigen Umgebung Ω. Der Teil von f_2, der eingefügt werden soll, ist durch den Bereich Γ mit Rand $\partial\Gamma$ gegeben.
Dazu setzt man das Drift-Vektorfeld folgendermaßen zusammen: Man nutzt die kanonischen Drift-Vektoren von f_1 in $\Omega\backslash\Gamma$ und die von f_2 in Γ. An der Grenze $\partial\Gamma$ wird das arithmetische Mittel der beiden Drift-Vektoren verwen-

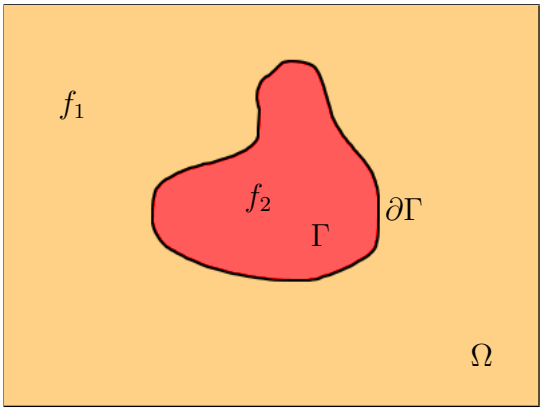

Abb. 2.3. Diagramm: Ineinanderkopieren

Abb. 2.4. Nahtloses Ineinanderkopieren. **Links außen**: Bild 1, Euler (460 x 600 Pixel)[19]. **Links Mitte**: Bild 2, Lagrange [20]. **Rechts Mitte**: Osmotische Rekonstruktion mit Bild 1 als Anfangsbild. **Rechts außen**: Maske.

det. Nimmt man als Anfangsbild f_1 auf der gesamten Bildfläche Ω, so bekommt man als stationären Zustand das ineinanderkopierte Bild von f_2 in f_1.

Dies kann unter Anderem dazu genutzt werden, um einer Person ein anderes Gesicht zu geben wie in Abbildung 2.4 deutlich wird. Hier wurde das Gesicht von Lagrange in das Bild von Euler kopiert. Es ist zu erkennen, dass die Farbwerte des ersten Bildes erhalten bleiben. Da das Bild von Lagrange im Gesicht kontrastreicher ist, als das von Euler, ist auch die Rekonstruktion an dieser Stelle kontrastreicher. Das lässt sich darauf zurückführen, dass die kanonischen Drift-Vektoren diese Kontraste automatisch mitliefern.

In Abbildung 2.5 fällt auf, dass die Erhaltung des mittleren Grauwertes dazu führen kann, dass die Rekonstruktion des ineinanderkopierten Bildes heller oder dunkler aussieht als erwartet. Im vorliegenden Fall ist der Bereich des Anfangsbildes, welcher nicht das Gesicht beinhaltet sehr dunkel. Dadurch dass das Gesicht jedoch sehr hell ist, ist der mittlere Grauwert des ersten Bildes ins-

gesamt heller, als wenn man nur den Bereich ohne das Gesicht sieht. Deswegen denkt man bei der Rekonstruktion, dass das Bild aufgehellt wurde.

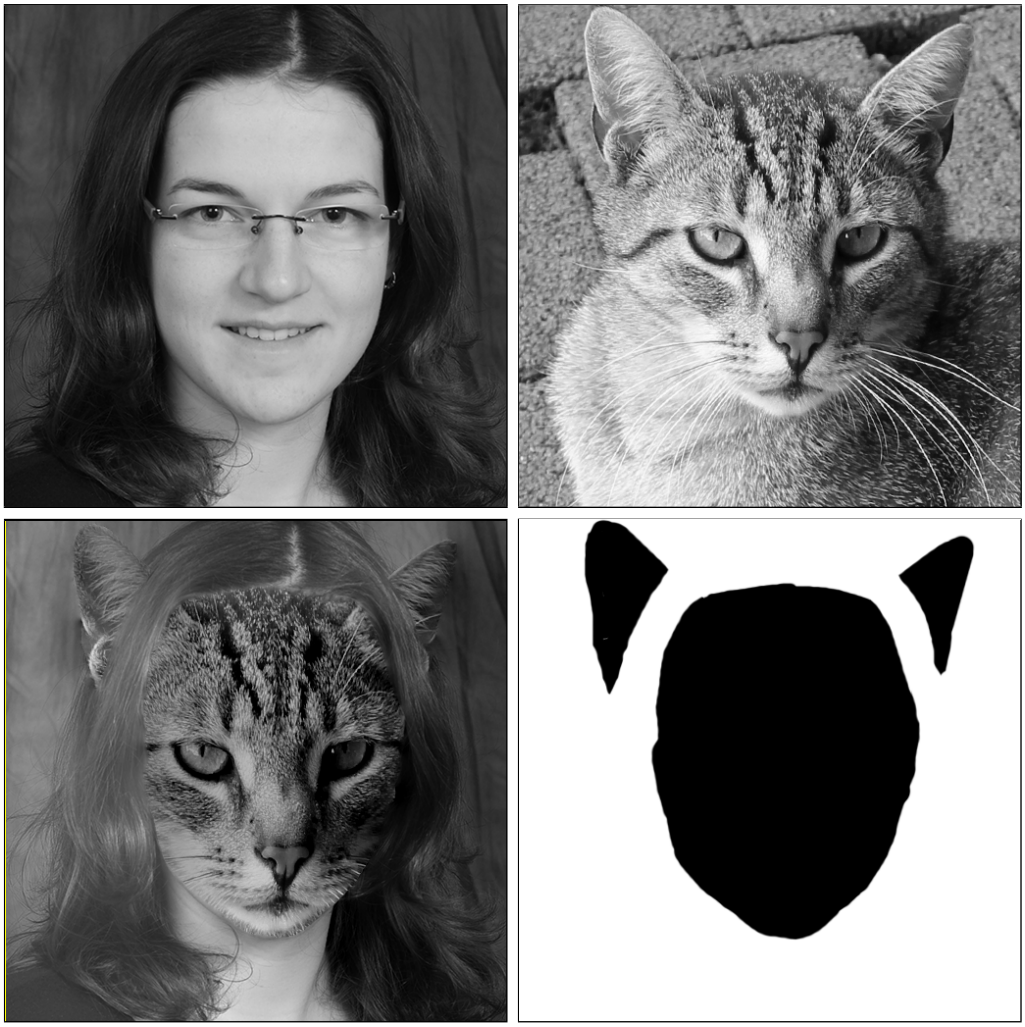

Abb. 2.5. Nahtloses Ineinanderkopieren. **Oben links**: Bild 1, Nane (512 x 512 Pixel). **Oben rechts**: Bild 2, Katze. **Unten links**: Osmotische Rekonstruktion mit Bild 1 als Anfangsbild. **Unten rechts**: Maske.

2.4 Codierung

Osmose kann auch zur Bildkompression verwendet werden. Das Verfahren hierzu wird in [18] kurz angerissen. Dazu wird ähnlich wie bei anderen PDE-basierten Methoden vorgegangen. Zuerst werden die Kanten des zu komprimierenden Bildes lokalisiert. Dazu kann man zum Beispiel Cannys Ansatz zur

Abb. 2.6. Codierung Mittels Osmose. **Links**: Original (512 x 512 Pixel). **Mitte**: Canny-Kanten, die etwa 9,6 % des Bildes ausmachen [1]. **Rechts**: Rekonstruktion mit mittlerem Grauwert des Originals und seinen kanonischen Drift-Vektoren in den Kanten [18].

Kantenerkennung nutzen [1]. Sind die Kantenpositionen bekannt, speichert man ein Bild ab, das zusätzlich zu den Kanten ihre beiden Nachbarpixel aus dem Originalbild speichert. Dadurch sind die Farben auf beiden Seiten der Kante gegeben. Zudem werden auch die Randpixel gespeichert. Zur Rekonstruktion wird dieses Bild mittels Diffusion interpoliert. Das bedeutet man löst die Diffusionsgleichung (1.3) für alle Pixel, die noch nicht bekannt sind, während die schon gegebenen Pixel unverändert bleiben [7]. Um Osmose für einen solchen Kompressionsprozess zu nutzen, reicht es aus, die Länge der Drift-Vektoren in den Kanten zu speichern. Die Richtung der Vektoren ergibt sich aus den Kanten selbst, da die Drift-Vektoren orthogonal zu den Kanten stehen. Alle anderen Drift-Vektoren werden auf Null gesetzt, sodass an diesen Stellen die Diffusions-Interpolation stattfindet.

3 ALGORITHMEN

In Kapitel 1 wurden das Diffusions- und Osmoseverfahren anhand ihrer physikalischen Definitionen erklärt und daraufhin die Anwendbarkeit in der digitalen Bildbearbeitung beschrieben. Da die digitale Welt keine stetigen Gleichungen versteht, müssen diese beiden Verfahren nun diskretisiert werden. Somit wird in diesem Kapitel die diskrete Theorie für Diffusion und Osmose sowie die Algorithmen, die von dem Programm dieser Arbeit benutzt werden, vorgestellt.

3.1 Diffusion

3.1.1 Diskretisierung

Die Diffusionsgleichung (1.10) liegt in kontinuierlicher Form vor. Um sie in einer Implementierung, d.h. in einer diskreten Umgebung anwenden zu können, benötigt man eine diskrete Darstellung der Gleichung. Dazu wird mit Hilfe der Taylor-Reihen-Entwicklung eine Approximation der ersten und zweiten Ableitung durch *finite Differenzen* hergeleitet. Die räumliche Diskretisierung von (1.10) ergibt sich damit als

$$\partial_t u = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_1^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_2^2}. \tag{3.1}$$

Dabei steht $u_{i,j} := u(x_i, y_j)$ für das Bild u im Pixel (i, j), mit $x_i := (i - \frac{1}{2})h_1$, $y_j := (j - \frac{1}{2})h_2$ und den Schrittweiten h_1 in x-Richtung und h_2 in y-Richtung. Somit lässt sich (3.1) als Matrix-Vektor-Produkt

$$\partial_t \boldsymbol{u} = \boldsymbol{A}\ \boldsymbol{u}(t) \tag{3.2}$$

mit

$$\boldsymbol{u}(0) = \boldsymbol{f}$$

als Anfangsbedingung schreiben, wobei u und f je durch einen Vektor $\boldsymbol{u},\ \boldsymbol{f} \in \mathbb{R}^p$ mit $p := N \cdot M$ als die Anzahl der Pixel dargestellt werden. Die Pixel (i, j)

werden dabei von einem einfachen Index $k(i,j)$ dargestellt. Betrachtet man die Menge $\mathcal{N}_n(k)$ als die Menge der Nachbarn in n-Richtung, so ergibt sich $\boldsymbol{A}$ als symmetrische $p \times p$-Matrix mit den Einträgen

$$a_{k,l} := \begin{cases} \dfrac{1}{h_n^2}, & \text{falls } l \in \mathcal{N}_n(k), \\ -\displaystyle\sum_{n=1}^{2} \sum_{l \in \mathcal{N}_n(k)} \dfrac{1}{h_n^2}, & \text{falls } l = k, \\ 0, & \text{ansonsten.} \end{cases} \tag{3.3}$$

Dabei sieht man, dass sich die Spalten von $\boldsymbol{A}$ zu 0 aufsummieren und die Einträge außerhalb der Diagonalen nichtnegativ sind.

Da der Diffusionsprozess eine zeitliche Evolution darstellt, muss auch die linke Seite in (3.1) diskretisiert werden. In dieser Arbeit werden zwei Arten behandelt, wie man diese zeitliche Diskretisierung vornehmen kann. Zuerst wird das *explizite Schema*

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}\ \boldsymbol{u}^k \tag{3.4}$$

mit τ als Zeitschritt und k als Iterationsschritt zur Zeit $k\tau$ betrachtet. Setzt man $\boldsymbol{P} := \boldsymbol{I} + \tau \boldsymbol{A}$, mit der Einheitsmatrix $\boldsymbol{I}$, ergibt sich

$$\boldsymbol{u}^{k+1} = \boldsymbol{P}\ \boldsymbol{u}^k \tag{3.5}$$

und (3.1) wird zu

$$\begin{aligned} u_{i,j}^{k+1} = &\left(1 - 2\frac{\tau}{h_1^2} - 2\frac{\tau}{h_2^2}\right)\ u_{i,j}^k \\ &+ \frac{\tau}{h_1^2}\ (u_{i+1,j}^k + u_{i-1,j}^k) \\ &+ \frac{\tau}{h_2^2}\ (u_{i,j+1}^k + u_{i,j-1}^k). \end{aligned} \tag{3.6}$$

Nutzt man wie im vorliegenden Fall homogene Neumann Randbedingungen (s. Kapitel 1), werden sie in den Randfällen $x, y = 0$ und $x = Nh_1$, $y = Mh_2$ zu

$$\frac{u_{1,j}^k - u_{0,j}^k}{h_1} = 0,\ \frac{u_{N+1,j}^k - u_{N,j}^k}{h_1} = 0 \qquad \forall j \in (0, M),$$

$$\frac{u_{i,1}^k - u_{i,0}^k}{h_2} = 0,\ \frac{u_{i,M+1}^k - u_{i,M}^k}{h_2} = 0 \qquad \forall i \in (0, N).$$

Das ergibt $u_{0,j}^k := u_{1,j}^k$, $u_{N+1,j}^k := u_{N,j}^k$ und $u_{i,0}^k := u_{i,1}^k$, $u_{i,M+1}^k := u_{i,M}^k$ $\forall i \in (0, N),\ j \in (0, M)$. Damit kann nun das explizite Schema implementiert werden, ohne extra auf die Randfälle eingehen zu müssen.

Die zweite zeitliche Diskretisierung ist durch das *implizite Schema*

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}\ \boldsymbol{u}^{k+1} \tag{3.7}$$

gegeben. Es ergibt sich ein lineares Gleichungssystem mit der Unbekannten $\boldsymbol{u}^{k+1}$. Ist die Matrix invertierbar, gilt (3.5) mit $\boldsymbol{P} := (\boldsymbol{I} - \tau \boldsymbol{A})^{-1}$.

Dadurch ergeben sich folgende Eigenschaften:

Proposition 1
Sei $\boldsymbol{f} \in \mathbb{R}^p$, $\boldsymbol{u}^k$ wird durch

$$\begin{aligned} \boldsymbol{u}^0 &= \boldsymbol{f}, \\ \boldsymbol{u}^{k+1} &= \boldsymbol{P}(\boldsymbol{u}^k)\ \boldsymbol{u}^k, \quad \forall k > 0 \end{aligned} \tag{3.8}$$

berechnet, wobei $\boldsymbol{P}$ folgende Eigenschaften erfüllt:
(D1) $\boldsymbol{P}$ ist stetig in $\boldsymbol{u}^k \quad \forall i, j$.
(D2) $\boldsymbol{P}$ ist symmetrisch.
(D3) Alle Spaltensummen von $\boldsymbol{P}$ ergeben 1.
(D4) $\boldsymbol{P}$ hat nur nichtnegative Einträge.
(D5) $\boldsymbol{P}$ hat eine positive Diagonale.
(D6) $\boldsymbol{P}$ ist irreduzibel.

Dann gelten:

(a) Es existiert eine eindeutige Lösung $\boldsymbol{u}^k$, die stetig von $\boldsymbol{f}$ abhängt $\forall k > 0$.

(b) Der mittlere Grauwert $\mu_{\boldsymbol{f}}$ des Anfangsbildes $\boldsymbol{f}$ bleibt erhalten:

$$\frac{1}{p}\sum_{i=1}^{p} \boldsymbol{u}_i^k = \frac{1}{p}\sum_{i=1}^{p} \boldsymbol{f}_i =: \mu_{\boldsymbol{f}} \qquad \forall k > 0. \tag{3.9}$$

(c) Es gilt das Maximum-Minimum-Prinzip:

$$\min_{1 \le j \le p} \boldsymbol{f}_j \le \boldsymbol{u}_i^k \le \max_{1 \le j \le p} \boldsymbol{f}_j \qquad \forall i \in \{1, \ldots, p\}, \forall k > 0. \tag{3.10}$$

(d) Die Folge

$$V^k := \sum_{i=1}^{p} r(\boldsymbol{u}_i^k) \tag{3.11}$$

ist eine Lyapunov-Folge für alle konvexen Funktionen $r \in C$: Fallend und nach unten beschränkt.

(e) Die Lösung von (3.8) konvergiert gegen einen konstanten stationären Zustand

$$\lim_{k\to\infty} \boldsymbol{u}_i^k = \mu_{\boldsymbol{f}} \qquad \forall i \in \{1,\dots,p\} \tag{3.12}$$

Beweis:
Der Beweis kann in [17] nachgelesen werden.

□

Mit diesen theoretischen Eigenschaften lässt sich eine Proposition formulieren, die für Systeme, wie in (3.2) leicht überprüfbare Bedingungen zur Konvergenz der Diffusionsevolution ergeben.

Proposition 2
Sei $\boldsymbol{f} \in \mathbb{R}^p$ und betrachte man das System

$$\begin{aligned} \partial_t \boldsymbol{u} &= \boldsymbol{A}(\boldsymbol{u})\ \boldsymbol{u}, \\ \boldsymbol{u}(0) &= \boldsymbol{f}, \end{aligned} \tag{3.13}$$

wobei $\boldsymbol{A}$ folgende Eigenschaften erfüllt:
(S1) $\boldsymbol{A}$ ist lipschitz-stetig in $\boldsymbol{u}$ für jede beschränkte Teilmenge in $\mathbb{R}^p$.
(S2) $\boldsymbol{A}$ ist symmetrisch.
(S3) Alle Spaltensummen von $\boldsymbol{A}$ ergeben 0.
(S4) $\boldsymbol{A}$ hat nichtnegative Einträge außerhalb der Diagonalen.
(S5) $\boldsymbol{A}$ ist irreduzibel.

Dann gelten:

(a) Das explizite Schema (3.4) erfüllt (D1)-(D6), falls gilt

$$\tau < \frac{1}{\max\limits_i |a_{i,i}|}. \tag{3.14}$$

(b) Das implizite Schema (3.7) erfüllt (D1)-(D6) für jeden Zeitschritt $\tau > 0$.

Beweis:
Der Beweis kann in [17] nachgelesen werden.

□

Man beachte, dass diese theoretischen Resultate nicht nur für den linearen Fall in (1.10) gelten, sondern auch für den allgemeineren Fall, der in (1.3) beschrieben wurde [17].
Hat man die räumlichen Schrittweiten $h_1 = h_2 = 1$, so ergibt sich für die Zeitschritt-Beschränkung (3.14) im expliziten Schema $\tau < \frac{1}{4}$.
Man sieht also, dass das explizite Schema (3.6) zwar einfach und leicht zu implementieren, allerdings auch ineffizient ist, um große Stoppzeiten zu erreichen.

Möchte man zum Beispiel eine Stoppzeit von 1000 erreichen, braucht man im expliziten Fall mit der Beschränkung $\tau < \frac{1}{4}$ mehr als 4000 Iterationen. Da im impliziten Fall beliebig große Zeitschritte gewählt werden können, ist dies wesentlich effektiver, sofern man einen effizienten und schnellen Löser für lineare Gleichungssysteme zur Verfügung hat. Da es sich hier um ein symmetrisches Gleichungssystem handelt, kann man z.B. die Cholesky-Zerlegung verwenden [10]. Im Folgenden werden Beschleunigungen für den expliziten Fall untersucht.

3.1.2 FED - Fast Explicit Diffusion

Das in (3.1.1) gezeigte explizite Schema ist zwar einfach zu implementieren und eignet sich gut für parallele Programmierung, z. B. auf Grafikkarten, aber es ist trotzdem sehr langsam. Die Beschränkung der Zeitschrittgröße auf $\tau < \frac{1}{4}$ führt in n Schritten nur zu einer Stoppzeit der Ordnung $\mathcal{O}(n)$. Das macht das explizite Schema ineffizient.
Grewenig et al. stellen in [4] ein explizites Verfahren vor, das eine Stoppzeit der Ordnung $\mathcal{O}(n^2)$ erreicht: Das *Fast Explicit Diffusion Scheme (FED Schema)*. Zur einfachen Herleitung wird das explizite Schema der eindimensionalen linearen Diffusion

$$\partial_t u = \partial_{xx} u \tag{3.15}$$

betrachtet. Ein Gitter der Größe h liefert hierfür die Diskretisierung

$$\frac{u_i^{k+1} - u_i^k}{\tau} = \frac{u_{i+1}^k - 2u_i^k + u_{i-1}^k}{h^2}. \tag{3.16}$$

Dies kann auch als Gauß'sche Faltung mit Kern

$\frac{\tau}{h^2}$	$1 - 2\frac{\tau}{h^2}$	$\frac{\tau}{h^2}$

betrachtet werden.

Sei $\boldsymbol{B}_{2n+1}$ ein Box-Filter der Länge $2n+1$. Er wird durch den Kern

$\frac{1}{2n+1}$	$\frac{1}{2n+1}$	...	$\frac{1}{2n+1}$	$\frac{1}{2n+1}$

repräsentiert. Durch die Symmetrie dieses Kerns ist sein Erwartungswert $\mu = 0$ und seine Varianz ist durch

$$\begin{aligned}
\operatorname{var}(\boldsymbol{B}_{2n+1}) &= \sum_{k=-\infty}^{\infty} (kh - \mu) \cdot \boldsymbol{B}_{2n+1}(kh) &&= \sum_{k=-n}^{n} (kh - 0)^2 \cdot \frac{1}{2n+1} \\
&= \frac{h^2}{2n+1} \sum_{k=-n}^{n} k^2 &&= \frac{2h^2}{2n+1} \sum_{k=1}^{n} k^2 \\
&= \frac{2h^2}{2n+1} \cdot \frac{n(n+1)(2n+1)}{6} &&= h^2 \cdot \frac{n(n+1)}{3}
\end{aligned} \tag{3.17}$$

gegeben. Laut dem zentralen Grenzwertsatz konvergieren iterierte Box-Filter gegen einen Gauß-Filter. In der Praxis liefern schon 3 bis 5 Iterationen eine gute Approximation. Iteriert man m Box-Filter der Länge $2n+1$, ergibt sich aufgrund der Additivität der Varianz unter der Faltung eine Annäherung an den Gauß-Filter mit Varianz

$$\sigma^2 = m \cdot \mathrm{var}(\boldsymbol{B}_{2n+1}) \tag{3.18}$$

an. Da die Gauß'sche Faltung mit Varianz σ^2 äquivalent zur linearen Diffusion mit Stoppzeit $T = \frac{1}{2}\sigma^2$ ist, konvergiert ein Box-Filter der Länge $2n+1$ also gegen das Ergebnis des linearen Diffusionsfilters mit Stoppzeit

$$t_n := \frac{1}{2} \cdot \mathrm{var}(\boldsymbol{B}_{2n+1}) = \frac{h^2}{2} \cdot \frac{n^2+n}{3}. \tag{3.19}$$

Nun wächst die Stoppzeit t_n quadratisch in n und da der Box-Filter nichtnegative Gewichte besitzt, die sich zu 1 aufsummieren, ist er immer stabil [14]. Grewenig et al. haben dazu folgendes Theorem aufgestellt [4]:

Theorem 1
Ein Box-Filter der Länge $2n+1$ kann in n explizite Diffusions-Kerne

$\frac{\tau_l}{h^2}$	$1 - 2\frac{\tau_l}{h^2}$	$\frac{\tau_l}{h^2}$

mit *sich verändernden* Zeitschrittgrößen

$$\tau_l = \frac{h^2}{2} \cdot \frac{1}{2\cos\left(\pi \frac{2l+1}{4n+2}\right)} \qquad (l = 0, \ldots, n-1) \tag{3.20}$$

und der entsprechenden Stoppzeit

$$t_n := \sum_{l=0}^{n-1} \tau_l = \frac{h^2}{3}\binom{n+1}{2} \tag{3.21}$$

faktorisiert werden.

Beweis:
Der Beweis hierzu kann in [4] nachgelesen werden.

□

Ein solcher Zyklus der Diffusionszeit t_n wird FED Zyklus genannt. Wegen seiner Äquivalenz zum Box-Filter ist auch FED stabil, obwohl etwa die Hälfte der Teil-Zeitschrittgrößen τ_l das Stabilitätskriterium $\tau_l < \frac{h^2}{2}$ verletzen.
Man beachte, dass das FED Schema im linearen Fall keine Beschleunigung des Diffusionsverfahrens bedeutet, denn die Implementierung von iterierten

Box-Filtern ist genauso effizient. Allerdings kann man dieses Schema verallgemeinern und erweitern, sodass es für beliebige Diffusionsfilter gilt.
Betrachtet man einen beliebigen isotropen oder anisotropen Diffusionsfilter, zu dem ein explizites Schema der Form (3.4) mit stabiler Zeitschrittgröße τ vorliegt, kann dieses Schema analog zur eindimensionalen linearen Diffusion beschleunigt werden: Man ersetzt die stabile Zeitschrittgröße τ durch einen Zyklus von n sich verändernden Teilzeitschrittgrößen $\tau_0 \dots \tau_{n-1}$. Wie im eindimensionalen Fall werden diese Schrittgrößen vom Box-Filter abgeleitet. Es wird dabei die stabile Zeitschrittgröße $\frac{h^2}{2}$ der eindimensionalen Diffusion in (3.20) durch die nun geltende stabile Zeitschrittgröße τ ersetzt:

$$\tau_l = \tau \cdot \frac{1}{2\cos\left(\pi \frac{2l+1}{4n+2}\right)} \qquad (l = 0, \dots, n-1). \tag{3.22}$$

Dies führt zu einem expliziten Schema, das u^{k+1} aus u^k in einem Zyklus mit n Zwischenschritten der Schrittgrößen $\tau_0 \dots \tau_{n-1}$ berechnet:

$$u^{k+\frac{l+1}{n}} = \left(\boldsymbol{I} + \tau_l \boldsymbol{A}(u^k)\right) u^{k+\frac{l}{n}}. \tag{3.23}$$

Einer dieser FED Zyklen kann als ein einziger *Superzeitschritt* betrachtet werden, seine Schrittgröße ist durch

$$t_n = \sum_{l=0}^{n-1} \tau_l = \tau \cdot \frac{n^2 + n}{3} \tag{3.24}$$

gegeben. Wie im linearen Fall sollte dieses Schema iteriert werden um den Diffusionsfilter gut zu approximieren, d.h. es sollten mehrere Zyklen hintereinander angewandt werden. Allerdings sind meist 3 bis 5 Zyklen ausreichend.
Damit kann FED für lineare oder nichtlineare, isotrope oder anisotrope Diffusionsfilter jeder Dimension genutzt werden; die einzige Einschränkung ist, dass die Matrix $\boldsymbol{A}$ symmetrisch sein und während eines FED Zyklus konstant bleiben muss [4].
Da man bei einer Implementierung nie exakt rechnen kann, muss man auch hier auf Rundungsfehler achten. Diese treten vor allem dann auf, wenn die Zeitschrittgrößen τ_l in aufsteigender Reihenfolge benutzt werden. Um diese Fehler im Zaum zu halten, werden die τ_l nach dem Prinzip der κ-Zyklen permutiert [2]. Dadurch ergibt sich ein Schema, das so einfach zu implementieren ist, wie das explizite Schema, durch seine Struktur jedoch eine quadratische Verbesserung der Evolutionszeit darstellt.

3.1.3 FSI - Fast Semi-iterative Diffusion

In [5] stellen Hafner et al. ein weiteres Schema zur Beschleunigung von Diffusionsfiltern vor, das sogenannte *Fast Semi-iterative Scheme (FSI Schema).*

Wie FED benutzt es als Grundlage den iterierten Box-Filter. Die Herleitung erfolgt bis (3.19) analog zum FED-Schema, jedoch wird die Beschleunigung durch Extrapolation der vorangegangenen Iterationen erreicht. Daraus wird folgendes Theorem abgeleitet:

Theorem 2
Ein Box-Filter $\boldsymbol{B}_{2n+1}$ der Länge $2n+1$ kann iterativ durch n explizite lineare Diffusionsschritte konstruiert werden:

$$\boldsymbol{B}_{2l+3} = \alpha_l(\boldsymbol{I} - \tau\boldsymbol{A})\boldsymbol{B}_{2l+1} + (1-\alpha_l)\boldsymbol{B}_{2l-1} \qquad (l = 0, \ldots, n-1) \tag{3.25}$$

mit $\tau := \frac{h^2}{2}$, $\alpha_l := \frac{4l+2}{2l+3}$, $\boldsymbol{B}_{-1} := \boldsymbol{I}$.

Bemerkung:
Für $k = 0$ ist $\boldsymbol{B}_3 = \boldsymbol{I} + \frac{h^2}{3}\boldsymbol{A}$ ein einzelner Diffusionsschritt mit Schrittgröße $\frac{2}{3}\tau$.
Beweis:
Die Behauptung lässt sich durch Induktion beweisen [5].

□

Um lineare Diffusion zu approximieren, wird auch hier der Box-Filter iteriert, d.h. (3.25) wird zyklisch angewendet, wobei der m-te Zyklus der Länge n durch

$$u^{m,k+1} = \alpha_k(\boldsymbol{I} + \tau\boldsymbol{A})u^{m,k} + (1-\alpha_k)u^{m,k-1} \tag{3.26}$$

mit $u^{m-1} := u^{m,0}$, $\alpha_k = \frac{4k+2}{2k+3}$ für $k = 0, \ldots, n-1$ gegeben ist. Für den nächsten Zyklus gilt $u^{m+1,0} := u^{m,n}$. Dies kann nun verallgemeinert und auf beliebige Diffusionsfilter angewendet werden, die ein explizites Schema des Typs

$$u^{k+1} = (\boldsymbol{I} + \tau\boldsymbol{A}(u^k))u^k \tag{3.27}$$

mit einer negativ semidefiniten, symmetrischen Matrix $\boldsymbol{A}$ besitzen. Die Zeitschrittgröße τ wird dabei durch $0 < \tau < \rho(\boldsymbol{A}(u^k))$ eingeschränkt, wobei ρ den Spektralradius $\rho(\boldsymbol{A}) := \max_i |\lambda_i|$ von $\boldsymbol{A}$ beschreibt. Dies sichert wegen der Symmetrie von A die Stabilität in der euklidischen Norm. Es lässt sich das FSI Schema

$$u^{m,k+1} = \alpha_k(\boldsymbol{I} + \tau\boldsymbol{A}(u^{m,k}))u^{m,k} + (1-\alpha_k)u^{m,k-1} \tag{3.28}$$

mit $u^{m-1} := u^{m,0}$, $\alpha_k = \frac{4k+2}{2k+3}$ für $k = 0, \ldots, n-1$ für beliebige Diffusionsfilter ableiten. Es zeigt den m-ten Zyklus der Länge n. Durch Wiederanwenden des Zyklus erreicht man die gewünschte Stoppzeit. Aufgrund der Symmetrie von $\boldsymbol{A}$ ist jede Iteration in einer entsprechenden Norm stabil [5]. Man sieht nun, dass das FSI Schema im eindimensionalen Fall äquivalent zum FED Schema ist, da beide äquivalent zum Box-Filter $\boldsymbol{B}_{2n+1}$ sind.

Im Fall von nichtlinearen Problemen bietet FSI allerdings wesentliche Vorteile gegenüber FED, da FSI durch die semi-iterative Struktur nichtlineare Aktualisierungen der Matrix $\boldsymbol{A}$ während eines Zyklus zulässt. Außerdem müssen die Teilzeitschritte bei FSI nicht neugeordnet werden, da schon $0 < \tau < \rho(\boldsymbol{A})$ für die Teilzeitschritte τ_l gilt.

Wie in [5] nachzulesen ist, ist dies nicht das einzige Gebiet, auf dem FSI Anwendung findet. Das Schema wird auch für andere Probleme verwendet wie zum Beispiel zum Lösen von linearen Gleichungssystemen, konvexen Optimierungsproblemen und sogar von bedingten konvexen Optimierungsproblemen.

3.2 Osmose

3.2.1 Diskretisierung

Um ein analoges System für den Osmosevorgang zu erhalten, wird die Drift-Diffusionsgleichung (1.18) diskretisiert. Wie bei der Diffusion werden finite Differenzen benutzt und man erhält für

$$\begin{aligned} \partial_t u &= \boldsymbol{\Delta} u - \operatorname{div}(\boldsymbol{d} \cdot u) \\ &= \partial_{xx} u + \partial_{yy} u - \partial_x(\mathrm{d}_1 u) - \partial_y(\mathrm{d}_2 u) \end{aligned} \tag{3.29}$$

die räumliche Diskretisierung

$$\begin{aligned} \partial_t u = \quad & u_{i,j} \left(-\frac{2}{h_1^2} - \frac{2}{h_2^2} - \frac{d_{1,i+\frac{1}{2},j}}{2h_1} + \frac{d_{1,i-\frac{1}{2},j}}{2h_1} - \frac{d_{2,i,j+\frac{1}{2}}}{2h_2} + \frac{d_{2,i,j-\frac{1}{2}}}{2h_2} \right) \\ & + u_{i+1,j} \left(\frac{1}{h_1^2} - \frac{d_{1,i+\frac{1}{2},j}}{2h_1} \right) + u_{i-1,j} \left(\frac{1}{h_1^2} + \frac{d_{1,i-\frac{1}{2},j}}{2h_1} \right) \\ & + u_{i,j+1} \left(\frac{1}{h_2^2} - \frac{d_{2,i,j+\frac{1}{2}}}{2h_2} \right) + u_{i,j-1} \left(\frac{1}{h_2^2} + \frac{d_{2,i,j-\frac{1}{2}}}{2h_2} \right), \end{aligned} \tag{3.30}$$

mit $u_{i,j}$ als Approximation im Punkt $((i-\frac{1}{2})h_1, (j-\frac{1}{2})h_2)^\top$, h_1 und h_2 als die Schrittweiten in x-, bzw. y-Richtung und $\boldsymbol{d} = (d_1, d_2)^\top$. Nutzt man homogene Neumann-Bedingungen und setzt man die Drift-Vektoren über die Grenzen auf Null, so gilt dies auch für die Randpunkte. Im Folgenden werden nur Drift-Vektorfelder $\boldsymbol{d} = (d_1(\boldsymbol{x}), d_2(\boldsymbol{x}))^\top$ benutzt, die

$$|d_1(\boldsymbol{x})| < \frac{2}{h_1}, \qquad |d_2(\boldsymbol{x})| < \frac{2}{h_2} \qquad \forall \boldsymbol{x} \in \Omega \tag{3.31}$$

genügen.

Für ein positives Bild v erhält man eine diskrete Approximation seines kanonischen Drift-Vektorfeldes $\boldsymbol{d} = \frac{\boldsymbol{\nabla} v}{v}$ zwischen den Gitterpunkten durch

$$d_{1,i+\frac{1}{2},j} = \frac{2(v_{i+1,j} - v_{i,j})}{h_1(v_{i+1,j} + v_{i,j})}, \qquad d_{2,i,j+\frac{1}{2}} = \frac{2(v_{i,j+1} - v_{i,j})}{h_2(v_{i,j+1} + v_{i,j})}. \tag{3.32}$$

Man kann somit (3.30) als gewöhnliches Differentialgleichungssystem sehen und als Matrix-Vektor-Produkt

$$\partial_t \boldsymbol{u} = \boldsymbol{A} \cdot \boldsymbol{u}(t) \tag{3.33}$$

mit

$$\boldsymbol{u}(0) = \boldsymbol{f} \tag{3.34}$$

als Anfangsbedingung schreiben, wobei die Bilder u und f durch die Vektoren $\boldsymbol{u}, \boldsymbol{f} \in \mathbb{R}^p$ mit einfacher Indizierung dargestellt werden, $p := N \cdot M$ ist dabei die Pixelanzahl der Bilder. Durch die Drift-Vektoren wird $\boldsymbol{A}$ zu einer unsymmetrischen $p \times p$-Matrix, was einen wesentlichen Unterschied zum Diffusionsszenario darstellt, da Diffusion zu symmetrischen Matrizen führt. Da die Gewichte der vier Nachbarn von $u_{i,j}$ in (3.30) mit der Einschränkung von (3.31) positiv sind, hat $\boldsymbol{A}$ nichtnegative Einträge außerhalb der Diagonalen. Außerdem addieren sich alle Spalten von $\boldsymbol{A}$ zu Null auf und $\boldsymbol{A}$ ist irreduzibel
Auch hier werden zwei Arten der zeitlichen Diskretisierung betrachtet. Der einfachste Fall ist das *explizite Schema*

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}\ \boldsymbol{u}^k \tag{3.35}$$

mit $\tau > 0$ als Zeitschritt und k als Approximation der Zeit $k\tau$. Mit $\boldsymbol{P} := \boldsymbol{I} + \tau \boldsymbol{A}$ wird daraus

$$\boldsymbol{u}^{k+1} = \boldsymbol{P}\ \boldsymbol{u}^k. \tag{3.36}$$

Für dieses Schema ergibt sich aus (3.30) nach Umstellen und Neusortieren

$$\begin{aligned} u_{i,j}^{k+1} = \ u_{i,j}^k + \ & \frac{\tau}{h_1^2}\left(u_{i+1,j}^k - 2u_{i,j}^k + u_{i-1,j}^k\right) \\ + \ & \frac{\tau}{h_2^2}\left(u_{i,j+1}^k - 2u_{i,j}^k + u_{i,j-1}^k\right) \\ + \ & \frac{\tau}{2h_1}\left(d_{1,i-\frac{1}{2},j}(u_{i-1,j}^k + u_{i,j}^k) - d_{1,i+\frac{1}{2},j}(u_{i+1,j}^k + u_{i,j}^k)\right) \\ + \ & \frac{\tau}{2h_2}\left(d_{2,i,j-\frac{1}{2}}(u_{i,j-1}^k + u_{i,j}^k) - d_{2,i,j+\frac{1}{2}}(u_{i,j+1}^k + u_{i,j}^k)\right). \end{aligned} \tag{3.37}$$

Eine alternative zeitliche Diskretisierung ist durch das *implizite Schema*

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^k}{\tau} = \boldsymbol{A}\ \boldsymbol{u}^{k+1} \tag{3.38}$$

gegeben. Um es zu lösen muss ein lineares Gleichungssystem mit der Unbekannten $\boldsymbol{u}^{k+1}$ gelöst werden. Falls die Matrix invertierbar ist, kann man das System wie in (3.36) mit $\boldsymbol{P} := (\boldsymbol{I} - \tau\ \boldsymbol{A})^{-1}$ notieren.

Es ergeben sich folgende Eigenschaften:

Proposition 3
Sei $\boldsymbol{f} \in \mathbb{R}_+^p$, betrachte

$$\boldsymbol{u}^0 = \boldsymbol{f} \tag{3.39}$$
$$\boldsymbol{u}^{k+1} = \boldsymbol{P}\ \boldsymbol{u}^k \qquad (k = 0, 1, \dots) \tag{3.40}$$

mit der (unsymmetrischen) Matrix $\boldsymbol{P} \in \mathbb{R}^{p \times p}$, die die folgenden Eigenschaften erfüllt:
(DLO1) Alle Spaltensummen von $\boldsymbol{P}$ ergeben 1.
(DLO2) $\boldsymbol{P}$ ist nichtnegativ.
(DLO3) $\boldsymbol{P}$ ist irreduzibel.
(DLO4) $\boldsymbol{P}$ hat nur positive Diagonaleinträge.

Dann gelten:

(a) Der mittlere Grauwert $\mu_{\boldsymbol{f}}$ des Anfangsbildes $\boldsymbol{f}$ bleibt erhalten:

$$\frac{1}{p}\sum_{i=1}^{p} \boldsymbol{u}_i^k = \frac{1}{p}\sum_{i=1}^{p} \boldsymbol{f}_i =: \mu_{\boldsymbol{f}} \qquad \forall k > 0 \tag{3.41}$$

(b) Die Positivität bleibt erhalten:

$$\boldsymbol{u}_i^k > 0 \qquad \forall i \in \{1, \dots, p\}, \forall k > 0 \tag{3.42}$$

(c) Es existiert ein eindeutiger stationärer Zustand für $k \to \infty$. Er ist gegeben durch den Eigenvektor $\boldsymbol{v} \in \mathbb{R}_+^p$ von $\boldsymbol{P}$ zum Eigenwert 1, der den gleichen mittleren Grauwert wie das Anfangsbild $\boldsymbol{f}$ besitzt.

Beweis:
Der Beweis kann in [13] nachgelesen werden.

□

Damit wird sichergestellt, dass ein Filter den mittleren Grauwert des Originalbildes und die Positivität erhält und man die volle Kontrolle über den stationären Zustand hat.
Wendet man diese diskrete Osmose Theorie nun auf das explizite sowie das implizite Schema an, sieht man, dass sie nicht nur für die parabolische zeitliche Entwicklung sondern auch für den elliptischen stationären Zustand genutzt werden kann:

Proposition 4
Sei $\boldsymbol{f} \in \mathbb{R}^p_+$ und sei die semidiskrete lineare Osmose Entwicklung

$$\begin{aligned} \boldsymbol{u}(0) &= \boldsymbol{f} && (3.43) \\ \boldsymbol{u}'(t) &= \boldsymbol{A}\ \boldsymbol{u}(t) && (3.44) \end{aligned}$$

mit der (unsymmetrischen) Matrix $\boldsymbol{A} = (a_{i,j}) \in \mathbb{R}^{p\times p}$ gegeben, die folgende Eigenschaften erfüllt:
(SLO1) Alle Spaltensummen von $\boldsymbol{A}$ ergeben 0.
(SLO2) $\boldsymbol{A}$ hat nur nichtnegative Einträge außerhalb der Diagonalen.
(SLO3) $\boldsymbol{A}$ ist irreduzibel.

Dann gelten:

(a) Das explizite Schema

$$\boldsymbol{u}^{k+1} = (\boldsymbol{I} + \tau \boldsymbol{A})\boldsymbol{u}^k \qquad (3.45)$$

erfüllt die Bedingungen (DLO1) - (DLO4) für diskrete lineare Osmose Prozesse, falls gilt

$$\tau < \frac{1}{|a_{i,i}|} \qquad \forall i \in \{1, \ldots, p\}. \qquad (3.46)$$

(b) Das implizite Schema

$$(\boldsymbol{I} - \tau \boldsymbol{A})\ \boldsymbol{u}^{k+1} = \boldsymbol{u}_k \qquad (3.47)$$

erfüllt die Bedingungen (DLO1) - (DLO4) für alle $\tau > 0$.

Beweis:
Auch dieser Beweis kann in [13] nachgelesen werden.

□

Gilt nun für die Gittergrößen $h_1 = h_2 = 1$, so ist $|d_1(x)| < 2$, $|d_2(x)| < 2$ und man erhält aus (3.46) $\tau < \frac{1}{8}$ als Einschränkung für die Zeitschrittgröße des expliziten Schemas. Diese Einschränkung macht das explizite Schema der Osmose doppelt so langsam wie im Falle der Diffusion. Da das implizite Schema beliebig große Zeitschritte zulässt, ist es wünschenswert einen schnellen Löser für unsymmetrische Gleichungssysteme zu finden. In [13] wurden mehrere Löser getestet, darunter SOR, Gauß-Seidel und BiCGStab. Da BiCGStab das beste Ergebnis lieferte, wird dieser Löser in dieser Arbeit verwendet.

3.2.2 BiCGStab

Das oben gezeigte explizite Schema (3.37) ist einfach zu implementieren, braucht aber sehr lange, um große Stoppzeiten zu erreichen, da ein Zeitschritt auf $\tau < \frac{1}{8}$ beschränkt ist um Stabilität des Verfahrens zu garantieren. Somit ist es hoch ineffizient. Auch sind Beschleunigungsverfahren wie FED und FSI, die bei der Diffusion sehr gute Ergebnisse erzielt haben, hier nicht zu empfehlen. Das Fehlen der Symmetrie von $\boldsymbol{A}$ führt dazu, dass die Erhaltung der Positivität nicht garaniert werden kann.
Betrachtet man das implizite Schema

$$(\boldsymbol{I} - \tau \boldsymbol{A})u^{k+1} = u^k, \tag{3.48}$$

erfüllt es die Bedingungen (DLO1)-(DLO4) von Poposition 3 für alle Zeitschrittgrößen τ. Da es bei Osmose auf den stationären Zustand ankommt, ist ein Schema, welches große Zeitschrittgrößen zulässt, unabdingbar. Löst man das parabolische Osmose Schema

$$w = (\boldsymbol{I} + \tau \boldsymbol{A})w \tag{3.49}$$

für den expliziten Fall und

$$(\boldsymbol{I} - \tau \boldsymbol{A})w = w \tag{3.50}$$

für den impliziten Fall, sind die Bedingungen (DLO1)-(DLO4) automatisch erfüllt. Da dadurch die Erhaltung des mittleren Grauwerts des Anfangsbildes $\boldsymbol{f}$ und der Positivität gewährleistet ist, ist die Lösung dieses Problems eindeutig. Zudem kommt, dass beide Gleichungen (3.48) und (3.49) zu der räumlichen Diskretisierung

$$\boldsymbol{A}w = \boldsymbol{0} \tag{3.51}$$

der elliptischen Gleichung

$$\boldsymbol{\nabla} u - \operatorname{div}(\boldsymbol{d}u) = \boldsymbol{0} \tag{3.52}$$

mit homogenen Neumann Bedingungen äquivalent sind. Das bedeutet, dass das Schema für jeden stabilen Zeitschritt τ gegen den korrekten stationären Zustand w konvergiert. Da das explizite Schema durch die strenge Zeitschritt-Beschränkung ineffizient ist, kann ein implizites Schema sehr effizient sein, sofern es einen schnellen Löser für unsymmetrische lineare Gleichungssysteme benutzt.
Ein solcher Löser ist die *BiCGStab*-Methode von van der Vorst [12]. BiCGStab ist eine Variante der *Konjugierten Gradienten* Methode für unsymmetrische Systeme. Obwohl es keine formale Konvergenztheorie dazu gibt, konvergiert BiCGStab in der Praxis sehr schnell. Im Anhang A wird die Methode näher

erläutert. An dieser Stelle ist anzumerken, dass das Programm dieser Arbeit teilweise Quellcode benutzt, der von Herrn Prof. Dr. Joachim Weickert bereitgestellt wurde. Der bereitgestellte Code wird benutzt, um die Rekonstruktion eines Bildes im Fall der Osmose unter Verwendung des impliziten Schemas mit der BiCGStab Methode zu berechnen. Er wurde leicht verändert um sich in die Struktur einer C++-Implementierung einzugliedern. Im Quellcode geht eindeutig hervor, um welchen Code es sich dabei handelt.
Da der stationäre Zustand das Ziel bei den Anwendungsgebieten der Osmose ist, steht nun die Frage im Raum, welche Stoppzeit benötigt wird, um diesen zu erreichen.
Die Konvergenzgeschwindigkeit einer PDE-Entwicklung wird durch ihre höchste Ableitung festgelegt. Bei der Osmose findet sich diese im Diffusionsterm. Nun weiß man, dass homogene Diffusion äquivalent zur Gauß'schen Faltung ist und die Diffusionszeit T im Verhältnis

$$T = \frac{1}{2}\sigma^2 \tag{3.53}$$

zur Standardabweichung σ des Gauß-Filters steht. Um das komplette Bild abzudecken, sollte σ nicht kleiner als die Bilddiagonale sein, welche bei einem Bild mit $N \times M$ Pixeln der Größe $h_1 \times h_2$ die Länge

$$d := \sqrt{(Nh_1)^2 + (Mh_2)^2} \tag{3.54}$$

besitzt. Daraus folgt, dass die Stoppzeit mindestens

$$T = \frac{(Nh_1)^2 + (Mh_2)^2}{2} \tag{3.55}$$

betragen sollte. Tabelle 3.1 zeigt die daraus resultierenden Stoppzeiten für verschieden große Bilder.

Bildgröße	T	Iterationszahl mit $\tau < \frac{1}{8}$ (explizit)
128 x 128	16.384	>130.000
256 x 256	65.536	>520.000
512 x 512	262.144	>2.000.000
1024 x 1024	1.048.576	> 8.000.000
200 x 300	65.000	>520.000
300 x 400	125.000	> 1.000.000
500 x 1000	625.000	> 5.000.000

Tabelle 3.1. Verschiedene Stoppzeiten für unterschiedliche Bildgrößen mit Pixelgröße $h_1 = h_2 = 1$.

In [13] wurden das explizite und das implizite Schema mit BiCGStab für verschiedene Bildgrößen untersucht. Für den Zeitschritt im expliziten Verfahren wurde $\tau = 0,12$ gewählt, im impliziten Verfahren $\tau = 10^5$. Es ergaben sich folgende Zeiten und Iterationszahlen:

	explizit		implizit		
Bildgröße	Zeit(s)	Iterationen	Zeit(s)	Iterationen	Beschleunigung
100 x 115	14,689	61.184	0,3179	2	46,2
200 x 230	$359,49$[1]	240.115	4,5454	2	79,1
400 x 460	$4.487,6$[2]	948.484	61,909	3	72,5

Tabelle 3.2. Rechenzeiten in Sekunden und Iterationsanzahl für verschiedene Bildgrößen und unterschiedliche Osmose Schemata [13].

Es ist deutlich erkennbar, dass das implizite Schema, welches das nichtsymmetrische Gleichungssystem mittels BiCGStab löst, wesentlich schneller ist. Es erzielt eine Beschleunigung bis zum Faktor 79 gegenüber dem expliziten Schema.

[1] 359,49 s $\approx$ 5,99 min

[2] 4.487,6 s $\approx$ 74,79 min $\approx 1\frac{1}{4}$ h

4 INTERFACE

In diesem Kapitel wird die grafische Benutzerschnittstelle (Interface) sowie das Framework Qt, mit dem das Programm dieser Arbeit (Editor) entwickelt wurde, vorgestellt und erklärt. Man kann dieses Kapitel auch als Benutzerhandbuch für den Editor betrachten.

4.1 Qt

4.1.1 Ursprünge und Historie

Qt ist ein Framework, welches die systemübergreifende Entwicklung grafischer Benutzeroberflächen (Graphical User Interface, GUI) in der Sprache C++ ermöglicht. Es bietet verschiedene Werkzeuge zur Programmentwicklung an, darunter befindet sich sowohl ein Editor zur grafischen Erstellung des Programms als auch eine Entwicklungsumgebung. Diese Umgebung ermöglicht es, den gleichen Code auf verschiedenen Betriebssystemen zu übersetzen und jeweils eine ausführbare Datei für das System zu erstellen.

Qt wurde Anfang der 1990er Jahre von den zwei Norwegern Haavard Nord und Eirik Chambe-Eng entwickelt. Die beiden arbeiteten 1990 an der Softwareentwicklung für Ultraschallgeräte in einem Trondheimer Krankenhaus. Ihre Software musste allerdings unter Unix, Macintosh und Microsoft laufen, was für sie einen erheblichen Mehraufwand bedeutete. In diesem Zuge beschlossen sie, ein systemübergreifendes GUI Framework zu entwickeln. Ihr Produkt nannten sie Qt, gesprochen wie das englische Wort *"cute"*, was übersetzt süß, hübsch, aber auch pfiffig bedeutet.
Sie gründeten die Firma Trolltech mit dem Hauptprodukt Qt. Diese Firma wurde in der Zwischenzeit von Nokia und später von der finnischen Firma Digia aufgekauft. Momentan ist sie unter dem Namen Qt Company eine Sparte von Digia [9] [21].

4.1.2 Entwicklung

In der vorliegenden Arbeit wurde Qt5 mit der Version 5.6.0 benutzt. Sie bietet die Entwicklungsumgebung *QtCreator* sowie den *QtDesigner* zur GUI-Entwicklung an.

QtCreator ist speziell auf Qt-Projekte ausgelegt. In der Programmiersprache C++ wird dadurch systemübergreifende Softwareentwicklung ermöglicht. Es erstellt beim Kompilieren automatisch die entsprechenden Zwischendateien, die für das aktuell benutzte Betriebssystem (Unix, Macintosh, Microsoft) benötigt werden und übersetzt diese mit dem systemeigenen Compiler in ausführbare Dateien.
Hat man auf einem Microsoft-Rechner ein Programm entwickelt, möchte aber zusätzlich, dass es unter Linux läuft, so benötigt man nur die gleiche Qt-Version auf einem Linux-Rechner. Man öffnet das Projekt mit QtCreator auf dem Linux-System und kompiliert es. Es wird eine unter Linux ausführbare Datei ausgegeben, ohne dass man das Programm komplett neu schreiben muss. Im schlimmsten Fall sind kleine Änderungen im Quellcode nötig, die den Sprachspezifikationen der verschiedenen Compiler zuzuschreiben sind.

Da Qt ein universell einsetzbares Framework ist, wurde es auch für den vorliegenden Editor verwendet. Auf der beiliegenden CD befindet sich jeweils eine Version für Linux und Windows sowie der Quellcode, der unter Linux kompliliert wurde.

4.2 Oberfläche

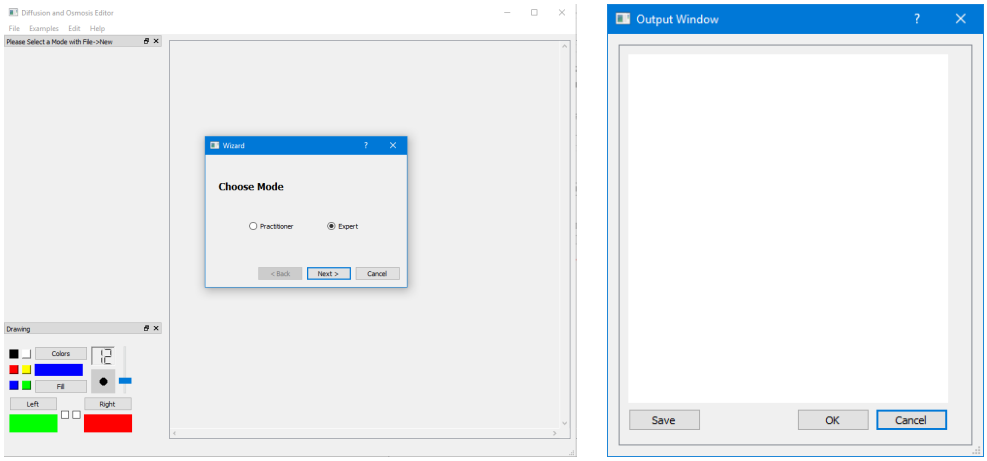

Abb. 4.1. Oberfläche des Editors. **Links (a):** Mainwindow beim Start mit dem Wizard im Vordergrund. **Rechts (b):** Outputwindow.

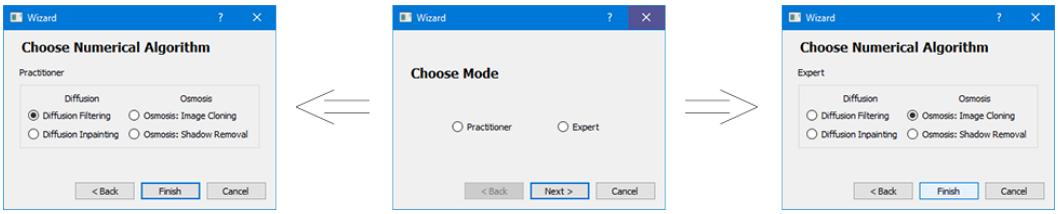

Abb. 4.2. Schematische Darstellung des Wizards mit seinen Auswahlmöglichkeiten.

Öffnet man den Editor, so sieht man zuerst das Hauptfenster (*Mainwindow*) im Hintergrund und den Auswahldialog (*Wizard*) im Vordergrund (s. Abb. 4.1(a)). Über den Wizard wählt man den numerischen Algorithmus aus, mit dem man arbeiten möchte. Dazu gibt es 4 verschiedene Methoden in 2 Modi. Man kann sich zuerst zwischen dem Practitioner- und dem Expert-Modus entscheiden. Hat man diese Entscheidung getroffen, wählt man zwischen *Diffusion Filtering, Diffusion Inpainting, Osmosis Image Cloning* oder *Osmosis Shadow Removal* aus (s. Abb. 4.2). Die entsprechende Oberfläche öffnet sich im Mainwindow, sobald man im Wizard auf Finish geklickt hat. Wählt man im Wizard nichts aus und schließt den Dialog, so findet man ein leeres Mainwindow vor, das links einen Button **New** beinhaltet. Damit kann man den Wizard wieder aufrufen. Alternativ findet man in der Menüleiste unter **File** den gleichen Befehl. Das Tastatur-Kürzel dafür ist Strg+N. Hat man einen Diffusions- oder Osmosevorgang durchgeführt, erscheint das Resultat danach in einem externen

Dialog, dem Ausgabefenster (*Outputwindow*) (s. Abb.4.1(b)).

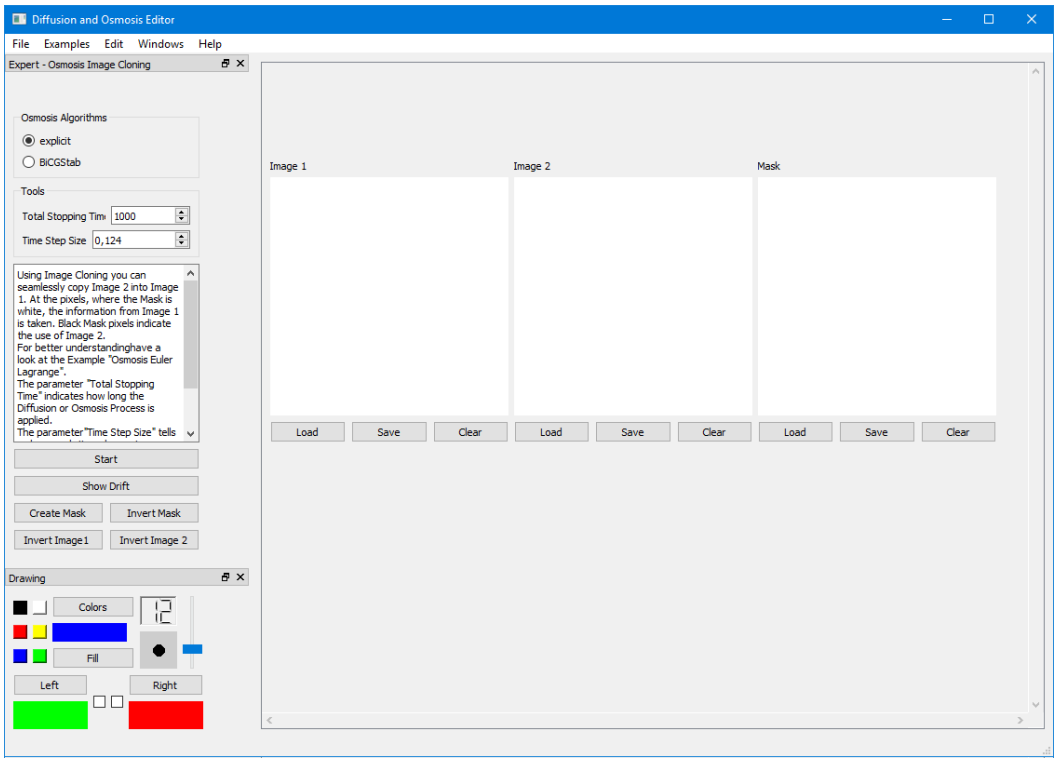

Abb. 4.3. Mainwindow mit der Ansicht für Osmosis Image Cloning im Expert-Modus.

Das Mainwindow ist wie in Abbildung 4.3 aufgebaut.
Oben befindet sich die Menüleiste.
Darunter befinden sich links zwei Dock Widgets, die untereinander angeordnet sind: Das Widget **Tools** (s. (4.2.2)), das die Werkzeuge beinhaltet, mit denen man die Parameter für die einzelnen Verfahren einstellen kann; und das Widget **Drawing** (s. (4.2.3)), das die Eigenschaften des Pinsels festlegt.
Im mittleren Teil des Fensters befinden sich je nach Methodenauswahl bis zu drei Bilder. Jedes Bild verfügt über drei Buttons, die sich darunter befinden: **Load**, **Save** und **Clear**. Sie ermöglichen das Laden und Speichern der Bilder sowie das Löschen der Farbinformationen. **Load** öffnet einen Dialog, der dem Benutzer erlaubt, Bilder zu öffnen. Es können Bilddateien der Formate PNG, JPG, BMP, PPM und PGM ausgewählt werden. Die Bildgröße darf dabei 1024 x 1024 Pixel nicht überschreiten. **Save** ermöglicht das Speichern der Bilder im PNG- oder PPM-Format. Der **Clear**-Button löscht die Farbinformationen, indem er das Bild mit weißer Farbe füllt.

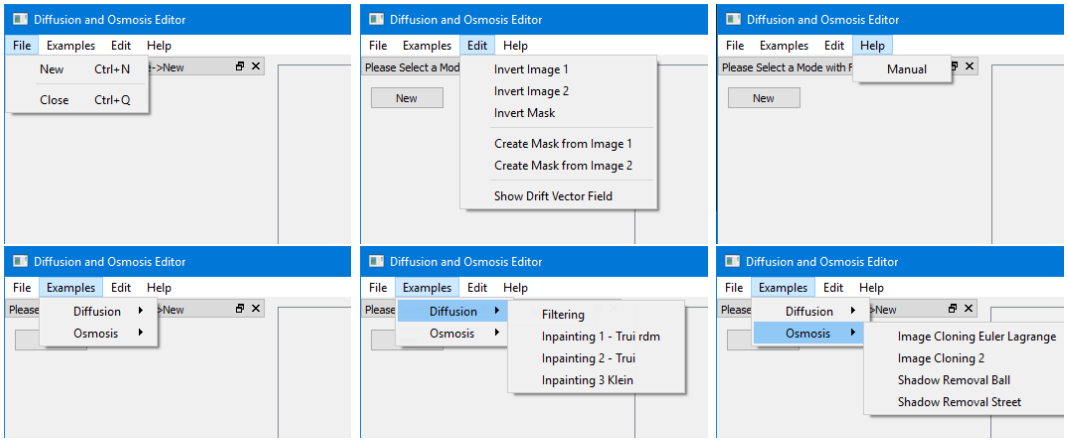

Abb. 4.4. Ansicht der Einträge der Menüleiste

4.2.1 Menüleiste

Abbildung 4.4 zeigt die einzelnen Menüpunkte.
Unter dem Punkt **File** findet man die Befehle **New** und **Close**. Ersteres erzeugt einen neuen Wizard, um eine andere numerische Methode auszuwählen. Letzteres schließt das Programm.
Der Punkt **Examples** lädt je vier Beispiele zu Diffusion und Osmose. Man sollte bei den Beispielen darauf achten, dass man vorher die richtige Methode ausgewählt hat.
Edit bringt ein paar Bearbeitungsschritte mit sich. So können die Bilder wie die Maske invertiert oder aus beiden Bildern eine Maske erzeugt werden. Dabei ist zu beachten, dass die Maske an allen Stellen, die im Bild weiß sind, auch weiß ist und an allen anderen Stellen schwarz. Es ist auch möglich, sich das Drift-Vektorfeld für die momentan eingestellte Methode anzeigen zu lassen. Dabei wird die Richtung der Vektoren durch verschiedene Farben gekennzeichnet (vgl. Abb. 4.5).
Unter dem Punkt **Help** öffnet sich mit Klick auf **Manual** eine PDF-Datei mit diesem Kapitel der Arbeit als Benutzerhandbuch.

Als Beispiel zeigt Abbildung 4.5 wie zuerst mit dem Pinsel in das erste Bild gezeichnet wurde und daraus mittels **Create Mask** die Maske generiert wurde. Dann wurden das Bild sowie die Maske invertiert. Als Letztes sieht man die Visualisierung des Drift-Vektorfeldes, nachdem man Osmosis Shadow Removal ausgewählt und das Ball-Beispiel geladen hat. Es ist deutlich erkennbar, wo die Drift-Vektoren auf Null gesetzt wurden.

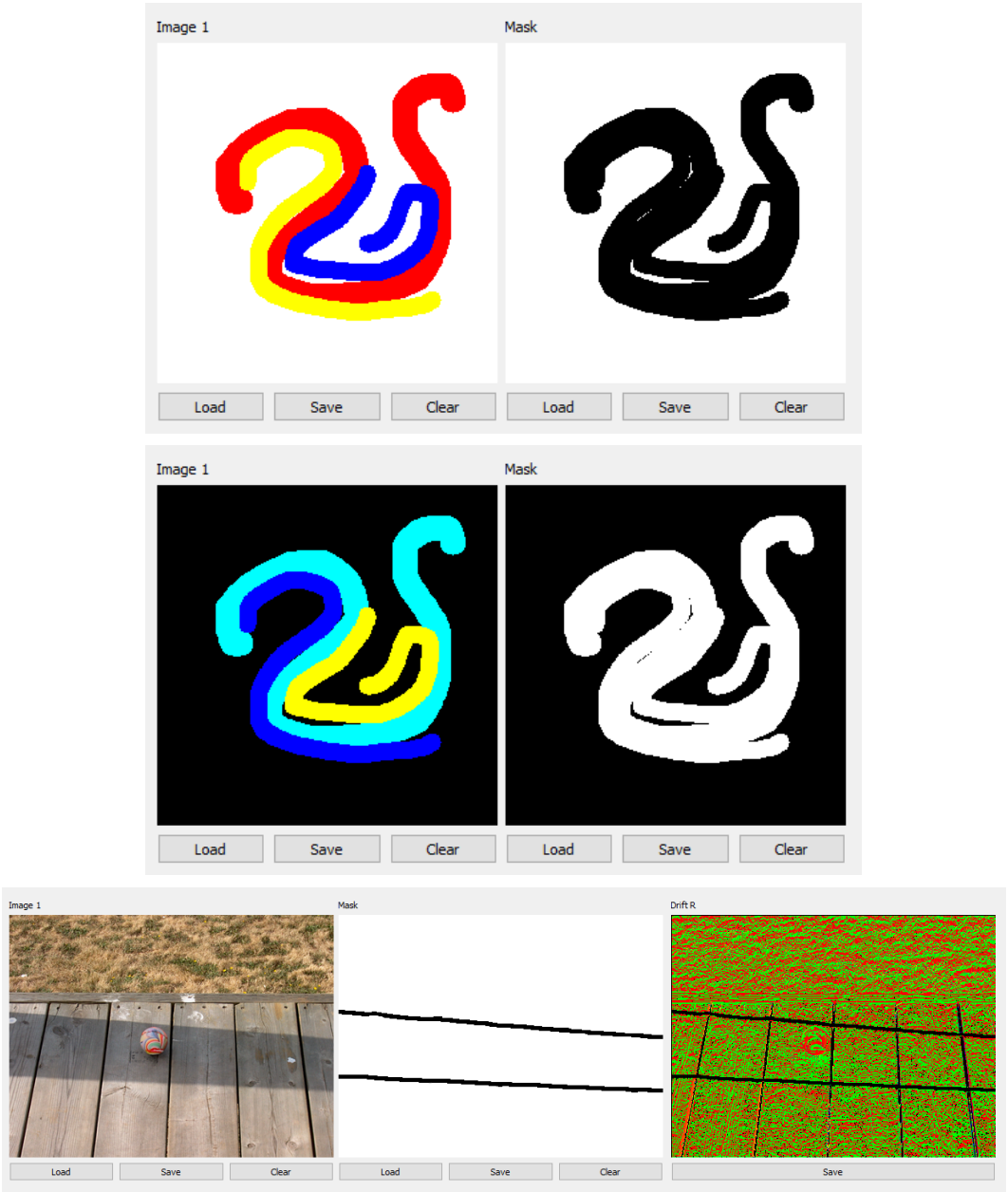

Abb. 4.5. Beispiel der Buttons **Create Mask**, **Invert** und **Show Drift**. **Oben**: Nach dem Zeichnen in Bild 1 wurde aus dem Bild eine Maske erzeugt. **Mitte**: Bild 1 sowie die Maske wurden invertiert. **Unten**: Visualisierung des Drift-Vektorfeldes in der Auswahl Osmosis Shadow Removal.

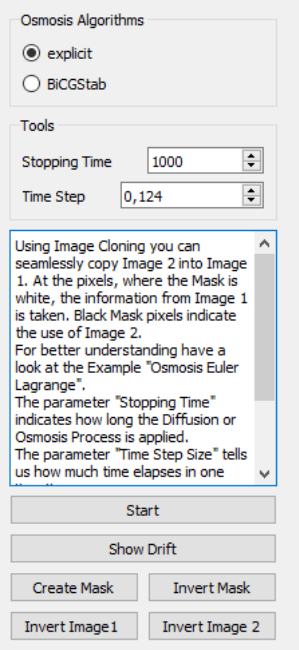

Abb. 4.6. Ansicht der Tools für einen Osmose Algorithmus im Expert-Modus

4.2.2 Tools

Die einzelnen Parameter der Gruppe **Tools** passen sich der vorherigen Auswahl im Wizard an. Im Expert-Modus kann man zusätzlich auswählen, mit welchem Algorithmus das Bild bearbeitet werden soll.
Darunter findet sich eine Textbox, die zu jeder Auswahl eine kleine Erklärung liefert. Abbildung 4.6 zeigt die Parameter und Auswahlmöglichkeiten anhand des Beispiels Image Cloning im Expert-Modus. Diese werden im Einzelnen in (4.3.1) beschrieben.
Der Button **Start** lässt den gewählten Algorithmus mit den eingestellten Parametern starten. Das kann je nach Rechenleistung des Computers und den Einstellungen etwas dauern. Währenddessen kann das Programm nicht weiter genutzt werden. Ist der Rechenprozess zu Ende, wird das Resultat als Bild im Outputwindow ausgegeben.
Die restlichen Buttons sind äquivalent zu den Befehlen unter **Edit**, wobei der Button **Create Mask** eine Maske aus dem ersten Bild erzeugt. Der Titel dieses Widgets gibt immer an, in welchem Modus man sich gerade befindet und welche Methode ausgewählt wurde.

4.2.3 Drawing

Abbildung 4.7 zeigt das Dock Widget **Drawing**. Hier werden die Eigenschaften des Pinsels, mit dem man in die Bilder zeichnen kann, eingestellt. In die beiden Bilder kann mit Farbe gezeichnet werden, in die Maske nur mit schwarz-weiß. Zeichnet man in die Maske mit einer von weiß verschiedenen Farbe, erscheint diese automatisch als schwarz.
Die 6 kleinen, bunten Buttons bieten einen Schnellzugriff auf die Grundfarben schwarz, weiß, rot, gelb, blau und grün. Der Button **Colors** öffnet einen Farbdialog, indem man eine Farbe mittels ihres HTML Codes oder über den Farbwähler auswählen kann (s. Abb. 4.7). Der Button **Fill** füllt das erste Bild komplett mit der darüber angegebenen Farbe aus. Der Slider rechts gibt die Pinselgröße an. Zusätzlich besteht die Möglichkeit, links oder rechts neben dem eigentlichen Pinselstrich einen Weiteren zu zeichnen. Dazu wird eine oder beide der Checkboxen unten links bzw. rechts ausgewählt. Die Farbfelder daneben geben an, in welcher Farbe diese Nebenlinien gezeichnet werden. Diese lässt sich ändern, wenn man auf die Buttons **Left** und **Right** klickt, die einen Farbdialog öffnen.

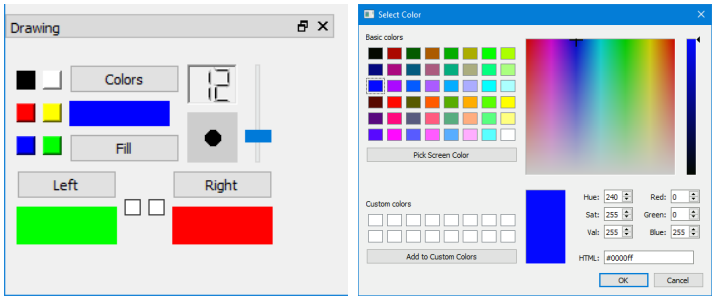

Abb. 4.7. Drawing. **Links (a):** Zeichenwerkzeuge. **Rechts (b):**Farbdialog.

4.3 Benutzung des Editors

4.3.1 Expert-Modus

Im Expert-Modus kann unter den Punkten Diffusion Filtering, Diffusion Inpainting, Osmosis Image Cloning und Osmosis Shadow Removal ausgewählt werden.
Je nach dem, welche Wahl getroffen wurde, werden die entsprechenden UI-Elemente im Mainwindow angezeigt. Die Abbildungen 4.8 und 4.9 zeigen die Ansichten des Mainwindows für Diffusion und Osmose.

Diffusion Filtering

Entscheidet man sich für den Punkt Diffusion Filtering, wird dem Anwender eine Oberfläche angezeigt, die ihm ermöglicht, auf ein Bild homogene Diffusion anzuwenden.
Es wird ein Bild gezeigt, man kann den Algorithmus wählen, mit dem gerechnet werden soll und die Parameter einstellen.

Diffusion Inpainting

Unter dem Punkt Diffusion Inpainting werden dem Benutzer zwei Bilder angezeigt, Bild 1 und die Maske. Die Maske ist ein schwarz-weiß Bild, deren weiße Pixel indizieren, dass in den entsprechenden Pixeln des Bildes 1 diffundiert wird. Die Pixel, die den schwarzen Pixel der Maske entsprechen, bleiben während des Diffusionsvorgangs unverändert. Außerdem kann man den Algorithmus sowie die Parameter einstellen.

Parameterwahl: Diffusion

Bei den beiden oben genannten Verfahren kann gewählt werden, welcher Algorithmus (explizit, FED, FSI) benutzt werden soll. Dabei können folgende Parameter eingestellt werden:

Stopping Time: Die Stoppzeit gibt an, nach welcher Zeit der Diffusionsvorgang enden soll.

Time Step: Der Zeitschritt besagt, wie groß ein einzelner Zeitschritt ist. Aus Stabilitätsgründen darf er 0.24 nicht überschreiten (s. Kapitel 3).

Nr. of cycles: Die Anzahl der Zykel gibt bei FED und FSI die Zahl der äußeren Zykel an, die durchlaufen werden.

Die Zahl der Iterationen hängt davon ab, ob man das explizite Verfahren oder FED bzw. FSI wählt. Als Beispiel benötigt eine Stoppzeit von 1000 mit Zeitschritt 0,24 im expliziten Fall mindestens $\lfloor 1000/0,24 \rfloor = 4166$ Iterationen.
Im Fall der beschleunigten Diffusion wird die Anzahl der Iterationen wie folgt festgelegt. Zuerst muss bestimmt werden, wie viele Teilzeitschritte in einem äußeren Zykel gemacht werden. Diese Anzahl bekommt man über die Formel $n := \lfloor 0.5 \cdot \left(\sqrt{1 + 24\frac{T}{m}} - 1 \right) \rfloor + 1$ mit T als Stoppzeit und m als Anzahl der äußeren Zykel, sie ergibt sich aus den Algorithmen für FED und FSI. Hat man diese Zahl ermittelt, ergibt sich die Iterationszahl als $n \cdot m$. Für eine Stoppzeit von 1000 ergeben sich somit bei 5 äußeren Zykel $35 \cdot 5 = 175$ Iterationen.

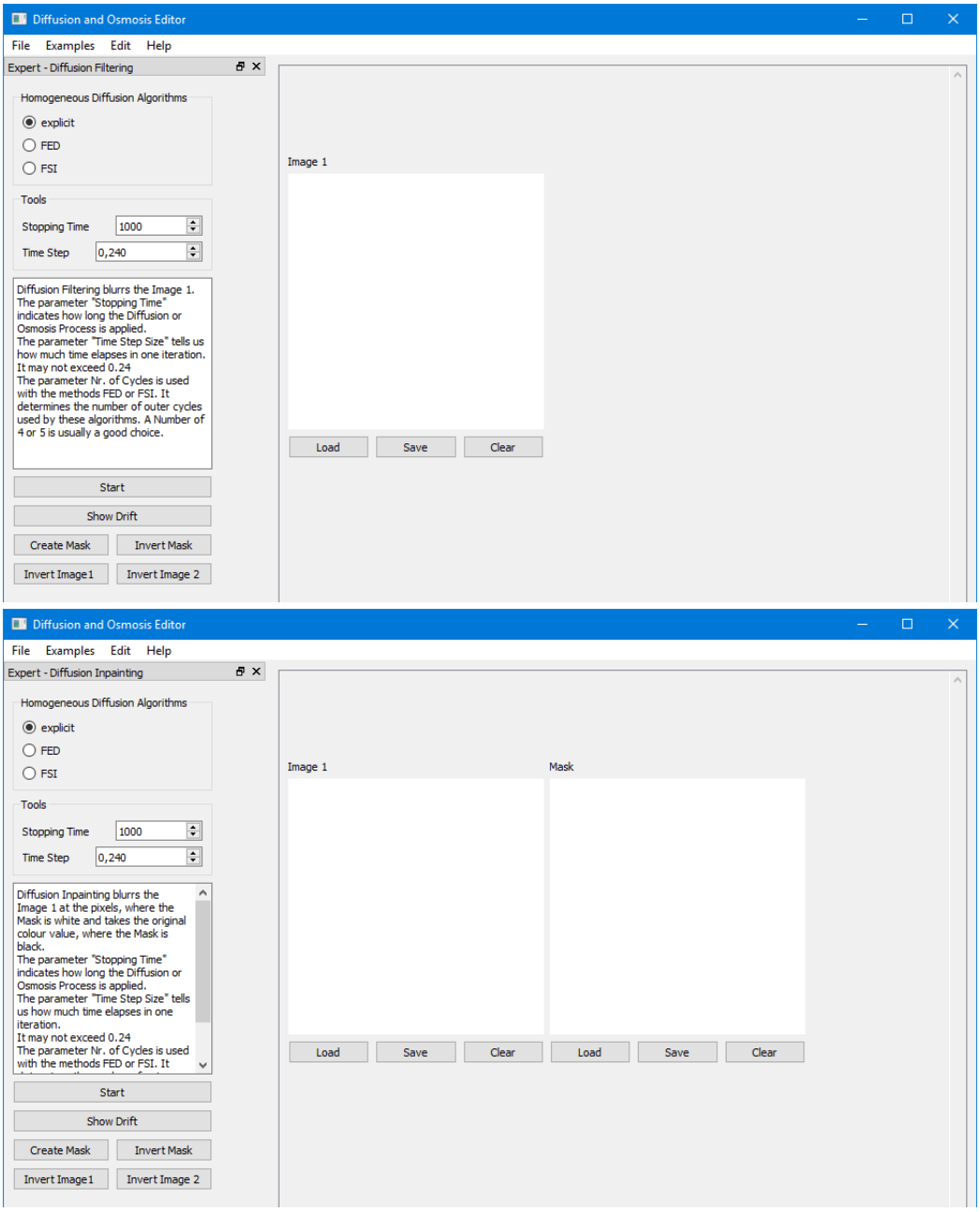

Abb. 4.8. Ansicht des Mainwindows nach Auswahl der Diffusionsfilter. **Oben**: Diffusion Filtering. **Unten**: Diffusion Inpainting.

Osmosis Image Cloning

Unter dem Punkt Osmosis Image Cloning werden drei Bildflächen angezeigt: zwei Bilder und eine Maske. Hier hat man die Möglichkeit, zwei Bilder nahtlos ineinander zu kopieren. An den Stellen, an denen die Maske schwarz gefärbt ist, wird Bild 2 in Bild 1 kopiert. Als Anfangsbild wird Bild 1 genommen. Die weißen Pixel der Maske indizieren die Verwendung der kanonischen Drift-Vektoren des ersten Bildes und die schwarzen Pixel die des zweiten Bildes. An der Grenze zwischen weißen und schwarzen Pixeln wird das arithmetische Mittel der kanonischen Drift-Vektoren beider Bilder genutzt. Dadurch ergibt sich bei Anwendung des Osmose Filters ein Bild, welches den entsprechenden Teil des zweiten Bildes in das erste Bild kopiert, ohne dass es eine ersichtlichen Übergang gibt.

Osmosis Shadow Removal

Wählt man Osmosis Shadow Removal aus, werden zwei Bildflächen angezeigt: Ein Bild und eine Maske. Um Schatten aus einem Bild zu entfernen, werden die Schattengrenzen mit schwarz in die Maske eingetragen während der Rest der Maske weiß bleibt. Es werden die kanonischen Drift-Vektoren des Anfangsbildes genutzt, die an den schwarzen Stellen der Maske auf Null gesetzt werden. Wegen der multiplikativen Invarianz des Osmose-Verfahrens und dem Fakt, dass Schatten in der Regel multiplikative Veränderungen der Farbe bedeuten, wird an diesen Stellen die Information, dass das Bild zum Schatten übergeht nicht weitergegeben und man erhält bei genügend großer Stoppzeit ein schattenfreies Bild.

Parameterwahl: Osmose

Wie bei den Diffusionsmodi kann auch bei der Osmose der Algorithmus (explizit, BiCGStab) gewählt werden. Dabei können bis auf die Anzahl der Zyklen die gleichen Parameter, wie bei den Diffusionsverfahren eingestellt werden:

Stopping Time: Die Stoppzeit gibt an, nach welcher Zeit der Osmosevorgang enden soll.

Time Step: Der Zeitschritt besagt, wie groß ein einzelner Zeitschritt ist. Aus Stabilitätsgründen darf er im expliziten Fall 0.124 nicht überschreiten (s. Kapitel 3). Im impliziten Fall, also wenn man BiCGStab ausgewählt hat, darf der Zeitschritt beliebig groß sein. Man sollte allerdings beachten, dass er nicht größer als die Stoppzeit ist.

Hier ergibt sich die Anzahl der Iterationen im expliziten wie im impliziten Fall als der Quotient von Stoppzeit durch Zeitschritt.

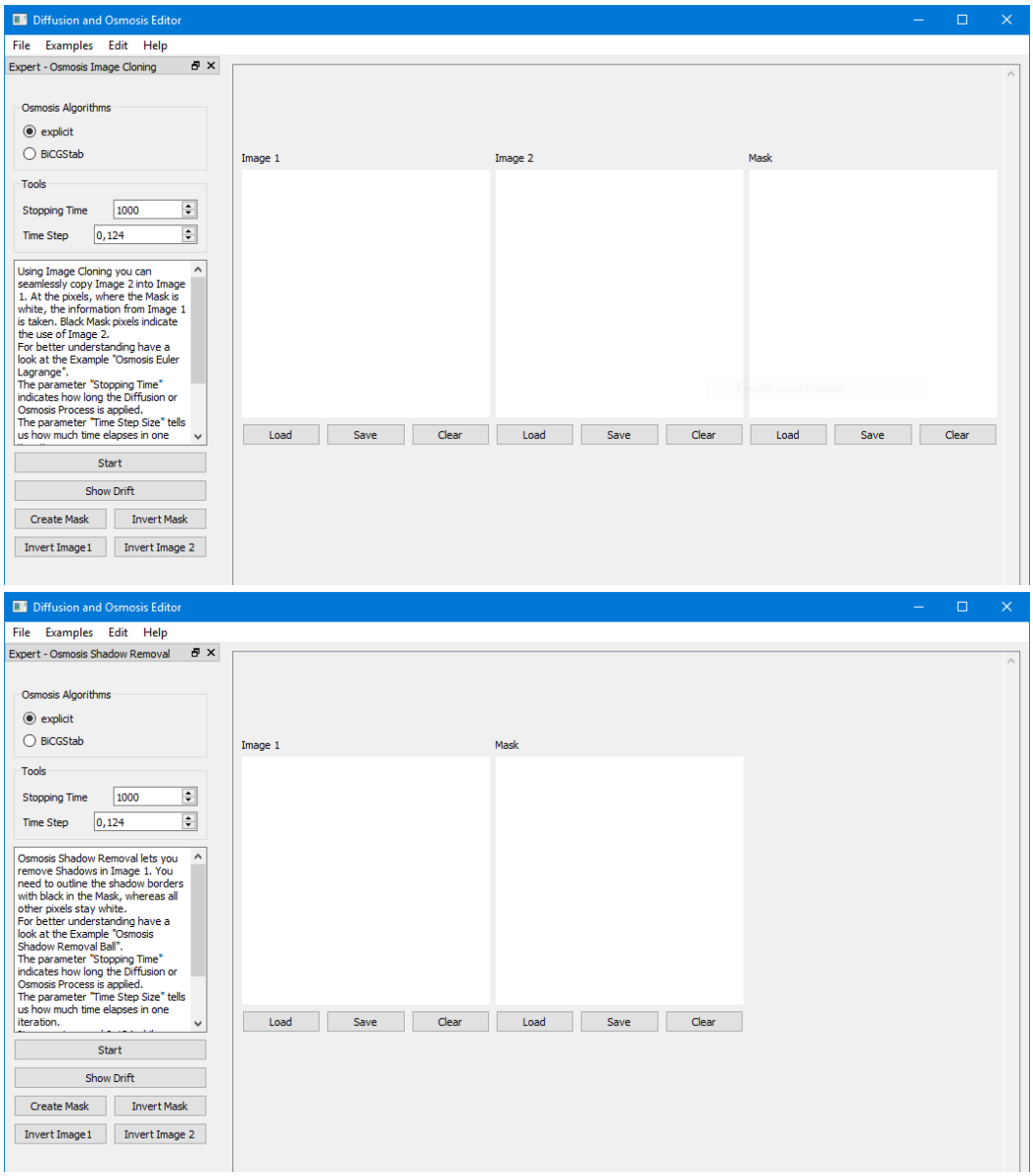

Abb. 4.9. Ansicht des Mainwindows nach Auswahl für Osmose. **Oben**: Osmosis Image Cloning. **Unten**: Osmosis Shadow Removal.

4.3.2 Practitioner-Modus

Wird im Wizard statt dem Expert-Modus der Practitioner-Modus gewählt, so erscheint die gleiche Auswahl wie vorher. Allerdings wird man zu einer vereinfachten Oberfläche des Mainwindows geleitet.
Wie im Expert-Modus erscheinen der Auswahl entsprechend eine, zwei oder drei Bildflächen (vgl. Abb. 4.8 und 4.9). Es gibt jedoch keine Algorithmen-Auswahl. Bei den Diffusionsverfahren wird das FSI Verfahren und bei den Osmosverfahren das BiCGStab Verfahren angewendet.
Dabei kann man jeweils den Parameter *Strength* beeinflussen, welcher der totalen Stoppzeit entspricht und aus Gründen der Benutzerfreundlichkeit umbenannt wurde. Die anderen Parameter werden automatisch wie in Tabelle 4.1 eingestellt. Diese Einstellungen erfolgen aufgrund der Stabilitätskriterien von Kapitel 3 im Fall der Diffusion, im Fall der Osmose wird ein von der Stoppzeit abhängiger Zeitschritt berechnet, um zu verhindern, dass die Stoppzeit versehentlich kleiner als der Zeitschritt gewählt wird.

	Diffusion	Osmose
Zeitschrittgröße	0,24	Stoppzeit/5
Anzahl der Zyklen	5	–

Tabelle 4.1. Parametereinstellung Practitioner

4.4 Beispiele

In diesem Abschnitt werden verschiedene Beispiele dargestellt, wie der Editor genutzt werden kann.
Als Beispiel für Diffusion Inpainting dient Abbildung 4.10. Sie beschreibt, wie die Anordnung der Pixel in der Inpainting Maske das Resultat beeinträchtigt.
In Abbildung 4.11 wird gezeigt, wie das Drift-Vektorfeld visualisiert wird. Statt die Vektorfelder bei Farbbildern für alle drei Kanäle anzuzeigen, wird aufgrund ihrer Ähnlichkeit nur das Drift-Vektorfeld des roten Kanals verwendet.
Schließlich werden in den Abbildungen 4.12 und 4.13 die kanonischen Drift-Vektorfelder mit den bearbeiteten Feldern verglichen.
Es ist spannend zu sehen, wie sich die Drift-Vektorfelder im Fall des Shadow Removal verhalten. Abbildung 4.12 zeigt eindeutig, dass im Drift-Vektorfeld der Schatten nur durch seine Grenzen definiert ist.

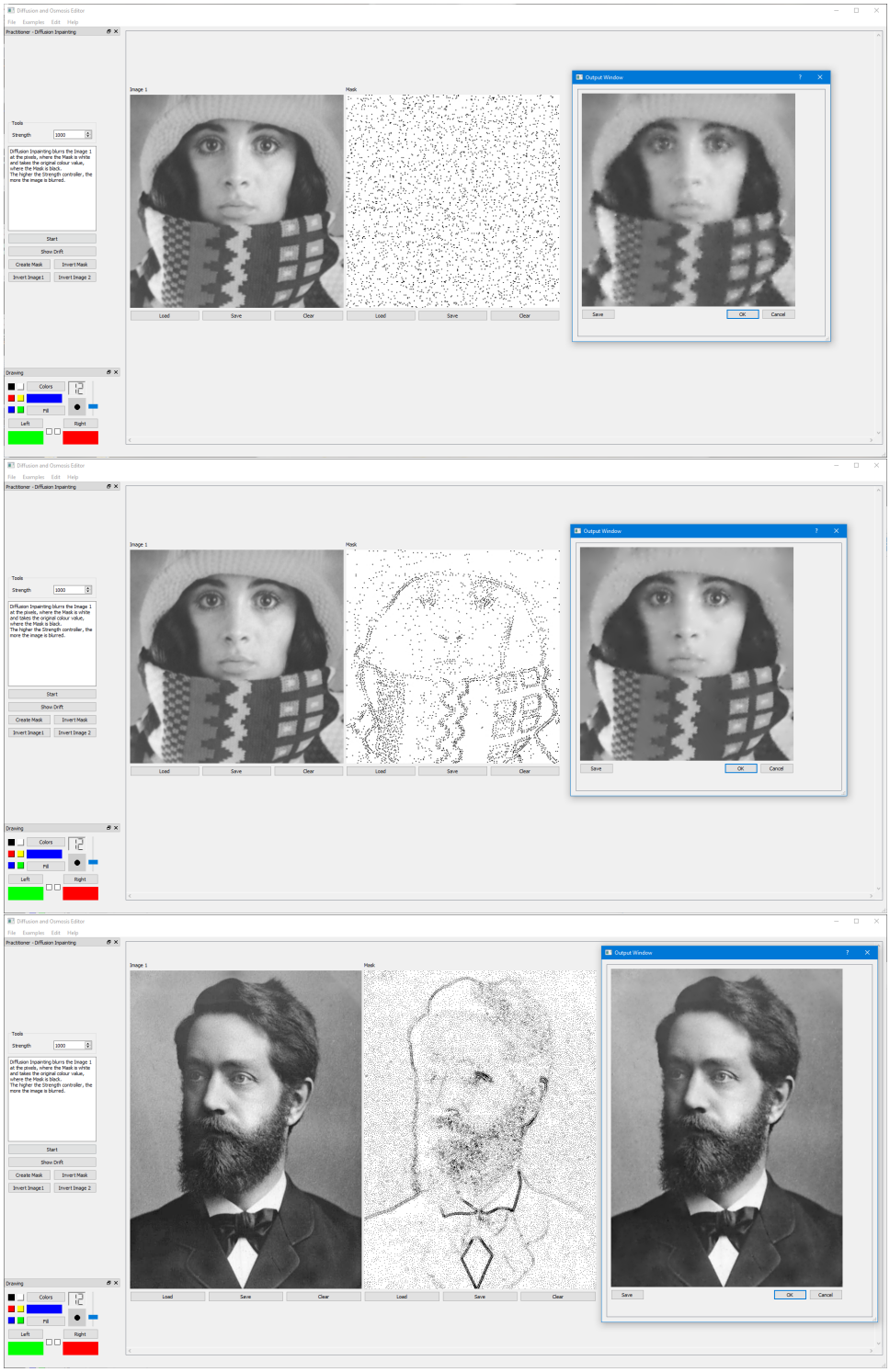

Abb. 4.10. Diffusion Inpainting mit Masken und Ausgaben. **Oben**: Trui mit einer Maske aus 5% der Pixel, zufällig angeordnet. **Mitte**: Trui mit einer Maske aus 5% der Pixel, sinnvoll angeordnet. **Unten**: Klein mit sinnvoll angeordneter Maske.

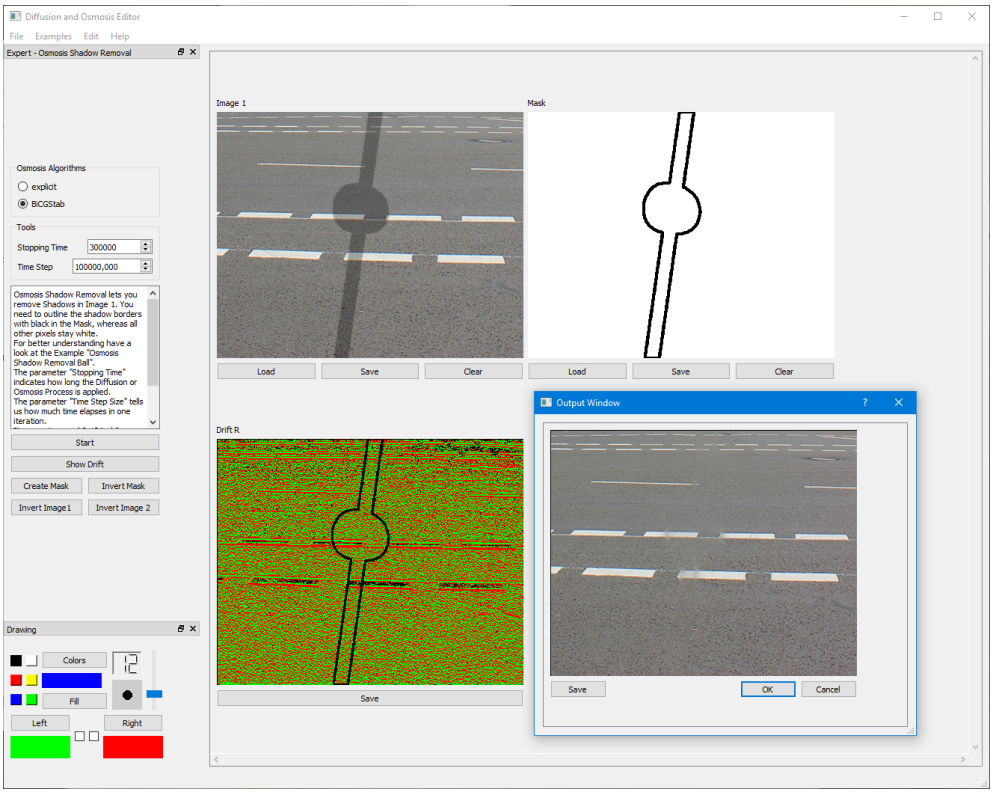

Abb. 4.11. Shadow Removal. Beispiel Street mit Drift-Vektorfeld und Ausgabe. Die schwarzen Stellen in der Maske geben an, wo das Drift-Vektorfeld auf Null gesetzt wurde.

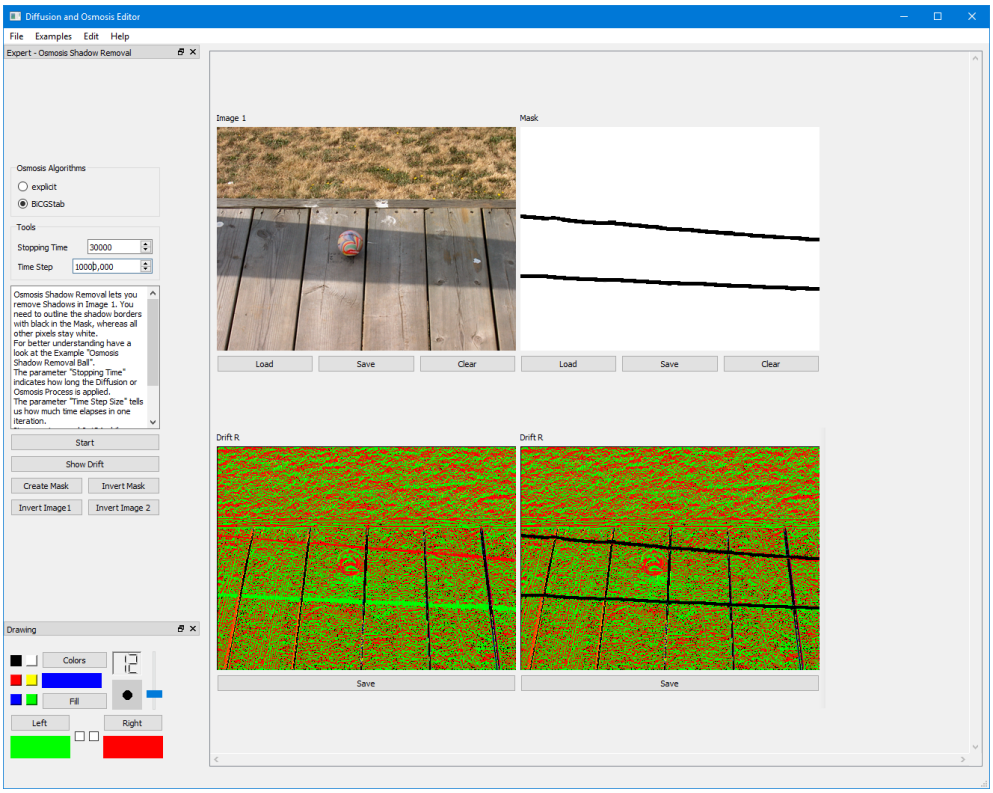

Abb. 4.12. Shadow Removal. Beispiel Ball mit Vergleich der Drift-Vektoren. Unten rechts sieht man das kanonische Drift-Vektorfeld des Anfangsbildes, unten links das bearbeitete Drift-Vektorfeld mit Schattengrenzen.

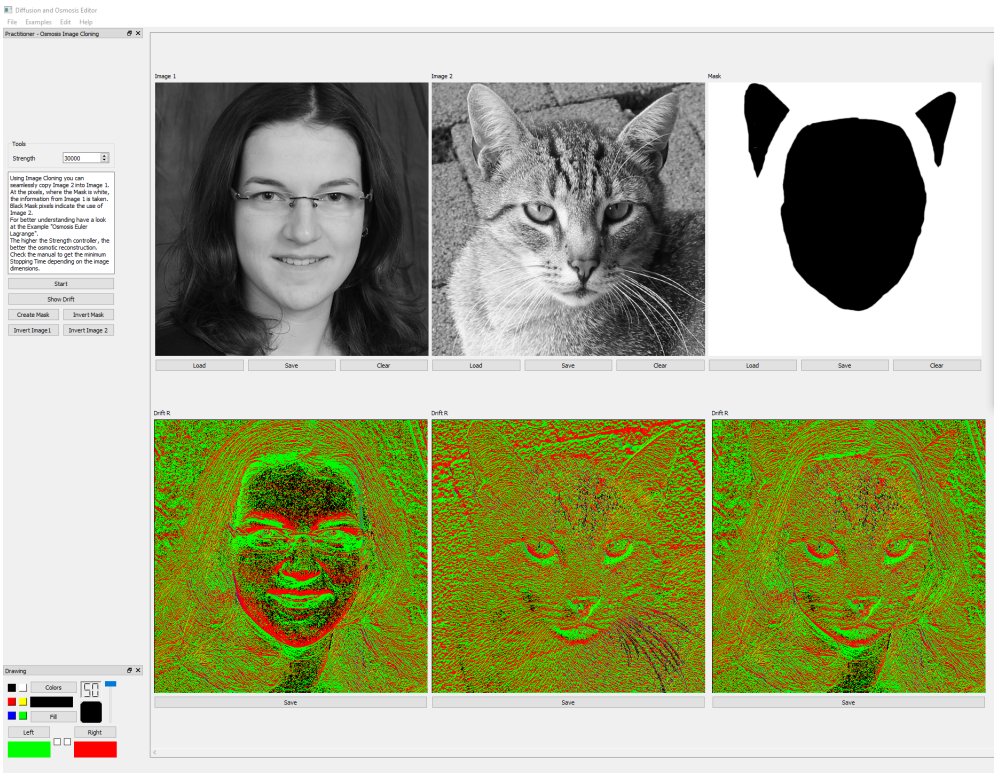

Abb. 4.13. Image Cloning. Beispiel Nane mit Vergleich der Drift-Vektoren. Die untere Reihe zeigt die Drift-Vektorfelder zu den entsprechenden Bildern. Das Feld unten links zeigt das ineinanderkopierte Drift-Vektorfeld.

5 FAZIT UND AUSBLICK

5.1 Fazit

Im Rahmen dieser Arbeit wurde erfolgreich ein leicht zu bedienender, benutzerfreundlicher Editor entwickelt, um Osmose für die Schattenreduktion und für das nahtlose Ineinanderkopieren zweier Bilder anzuwenden. Zusätzlich kann damit homogene Diffusion sowie diffusionsbasiertes Inpainting genutzt werden. Durch die Trennung in die Modi Expert und Practitioner wird ein breites Spektrum an potentiellen Benutzern angesprochen. Durch seine Bedienerfreundlichkeit ermöglicht der Editor allgemeinen Anwendern (Practitioner) die einfache Verbesserung bzw. Bearbeitung von digitalen Bildern. Zudem bietet er im Expert-Modus anspruchsvollen Nutzern umfassende Gestaltungsmöglichkeiten angesichts der Algorithmenauswahl und Parametereinstellung.
Da der Editor Funktionalitäten besitzt, um selbst mit dem Pinsel zu zeichnen, besitzt er auch einen künstlerischen Mehrwert und engagierte Künstler können durch Anwendung der verschiedenen Verfahren auf ihre Zeichnung interessante Werke entstehen lassen.

5.2 Ausblick

Der Editor kann auf verschiedene Arten weiterentwickelt werden.

Zuerst kann das Spektrum der genutzten Verfahren erweitert werden. Das bedeutet, das weitere Diffusionsfilter zur Bildverarbeitung angeboten werden können. Insbesondere nichtlineare anisotrope Diffusion stellt ein interessantes Anwendungsgebiet dar.
Um den Editor dahingehend weiterzuentwickeln, müssten entweder verschiedene Diffusionstensoren schon vorprogrammiert sein, sodass der Nutzer nur einen auswählen muss und die entsprechenden Parameter einstellt; oder aber man entwickelt eine Oberfläche, die es ermöglicht, eigene, benutzerdefinierte Diffusionstensoren einzugeben. Diese sollte die theoretischen Eigenschaften

des Skalenraumkonzeptes für die nichtlineare anisotrope Diffusion prüfen. Eine solche Umgebung kann automatisch auch auf lineare und nichtlineare isotrope Diffusion angewandt werden.

Dann können die numerischen Verfahren parallelisiert werden. Dies ermöglicht ihre Berechnung zum Beispiel auf einer Grafikkarte und bietet dadurch eine starke Beschleunigung. Damit wird eine interaktive Benutzung des Editors ermöglicht und gerade im Fall der Osmose wird dies zu einer interessanten und anspruchsvollen Aufgabe, da im impliziten Fall ein unsymmetrisches Gleichungssystem zu lösen ist.

Des Weiteren kann der Osmosefilter in schon bestehende, professionelle Bildbearbeitungsprogramme eingebunden werden. Diese Programme bieten oft eine Möglichkeit, eigene Plug-Ins zu entwickeln, diese selbst zu nutzen und sie zu verbreiten.

ANHANG A

Das BiCGStab-Verfahren

Die Herleitung des Verfahrens wird z.B. in [6] beschrieben.
Sei das lineare Gleichungssystem

$$Ax = b \tag{A.1}$$

mit einer regulären Matrix $A \in \mathbb{R}^{n \times n}$ gegeben. Die Lösung von (A.1) kann wie folgt bestimmt werden:

1. Wähle Startvektor $x^0 \in \mathbb{R}^n$, setze $r^0 := b - Ax^0$, $p^0 := r^0$.
2. Wähle $\hat{r}^0$ mit $\langle r^0, \hat{r}^0 \rangle \neq 0$.
3. für $k = 0, 1, 2 \cdots$

   $\alpha_k := \frac{\langle r^k, \hat{r}^0 \rangle}{\langle Ap^k, \hat{r}^0 \rangle}$;

   $s^k := r^k - \alpha_k A p^k$;

   $\omega_{k+1} := \frac{\langle As^k, s^k \rangle}{\langle As^k, As^k \rangle}$;

   $x^{k+1} := x^k + \alpha_k p^k + \omega_{k+1} s^k$;

   $r^{k+1} := s^k - \omega_{k+1} A s^k$;

   $\beta_k := \frac{\alpha_k}{\omega_{k+1}} \cdot \frac{\langle r^{k+1}, \hat{r}^0 \rangle}{\langle r^k, \hat{r}^0 \rangle}$;

   $p^{k+1} := r^{k+1} + \beta_k \left(p^k - \omega_{k+1} A p^k \right)$;

Ein mögliches Abbruchkriterium ist $\|r^k\| < \epsilon$.

LITERATURVERZEICHNIS

[1] J. Canny. *A computational approach to edge detection.* In IEEE Transactions on Pattern Analysis and Machine Intelligence, Nr. 8: S. 679-698. 1986.

[2] W. Gentzsch; A. Schlüter. *Über ein Einschrittverfahren mit zyklischer Schrittweitenänderung zur Lösung parabolischer Differentialgleichungen.* In: ZAMM, Zeitschrift für Angewandte Mathematik und Mechanik, Nr. 58: S. 415-416. 1978.

[3] G. Gerig; O. Kübler; R. Kikinis; F. A. Jolesz. *Nonlinear anisotropic filtering with a diffusion tensor.* In: IEEE Transactions of Medical Imaging, Nr. 11, S. 221-232. 1992.

[4] S. Grewenig; J. Weickert; A. Bruhn. *From Box Filtering to Fast Explicit Diffusion.* In: Goesele M., Roth S., Kuijper A., Schiele B., Schindler K. (Hrsg.): Pattern Recognition. DAGM 2010. Lecture Notes in Computer Science, Nr. 6376: S. 533-542. Springer, Berlin, 2010.

[5] D. Hafner; P. Ochs; J. Weickert; M. Reißel; S. Grewenig. *FSI Schemes : Fast Semi-Iterative Solvers for PDEs and Optimisation Methods.* In: Rosenhahn B., Andres B. (Hrsg.): Pattern Recognition. GCPR 2016. Lecture Notes in Computer Science, Nr. 9796, S. 91-102. Springer, Cham, 2016.

[6] C. Kanzow. *Numerik linearer Gleichungssysteme: Direkte und iterative Verfahren.* Springer Berlin Heidelberg, 2005.

[7] M. Mainberger; J. Weickert. *Edge-based image compression with homogeneous diffusion.* In: X. Jiang, N. Petkov (Hrsg.): Computer Analysis of Images and Patterns. Lecture Notes in Computer Science, Nr. 5702: S. 476-483, Springer, Berlin. 2009.

[8] P. Peter. *Image Compression.* Universität des Saarlandes. Vorlesungsskript Wintersemester 2014.

[9] Qt. *Our Story.* `https://www.qt.io/company#story`. Letzte Überprüfung: 21. Juli 2018.

[10] S. Rjasanow. *Praktische Mathematik*. Universität des Saarlandes. Vorlesungsskript Sommersemester 2011.

[11] Bibliographisches Institut (Mannheim). Redaktion Schule und Lernen. *Chemie: das Fachlexikon von A-Z*. Dudenverlag, 2007.

[12] H. van der Vorst. *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*. In: SIAM Journal on Scientific and Statistical Computing, Nr. 13 (2): S. 631-644. 1992.

[13] O. Vogel; K. Hagenburg; J. Weickert; S. Setzer. *A Fully Discrete Theory for Linear Osmosis Filtering*. In: Kuijper A., Bredies K., Pock T., Bischof H. (Hrsg.): Scale Space and Variational Methods in Computer Vision. SSVM 2013. Lecture Notes in Computer Science, Nr. 7893: S. 368-379. Springer, Berlin, 2013.

[14] J. Weickert. *Differential Equations in Image Processing*. Universität des Saarlandes. Vorlesungsskript Wintersemester 2015.

[15] J. Weickert. *Image Processing and Computer Vision*. Universität des Saarlandes. Vorlesungsskript Wintersemester 2013.

[16] J. Weickert. *Scale-space properties of nonlinear diffusion filtering with a diffusion tensor*. In: Technical Report 110, Laboratory of Technomathematics, University of Kaiserslautern. 1994.

[17] J. Weickert. *Anisotropic Diffusion in Image Processing*. Teubner, Stuttgart, 1998.

[18] J. Weickert; K. Hagenburg; M. Breuß; O. Vogel. *Linear Osmosis Models for Visual Computing*. In: Heyden A., Kahl F., Olsson C., Oskarsson M., Tai XC. (Hrsg.): Energy Minimization Methods in Computer Vision and Pattern Recognition. EMMCVPR 2013. Lecture Notes in Computer Science, Nr. 8081: S. 26-39. Springer, Berlin, 2013.

[19] Wikimedia. *Bild Euler*. `https://commons.wikimedia.org/wiki/File:Leonhard_Euler_by_Handmann.png`. Letzte Überprüfung: 21. Juli 2018.

[20] Wikimedia. *Bild Lagrange*. `https://commons.wikimedia.org/wiki/File:Joseph_Louis_Lagrange.jpg`. Letzte Überprüfung: 21. Juli 2018.

[21] Wikipedia. *Qt (Bibliothek) - Wikipedia Artikel*. `https://de.wikipedia.org/wiki/Qt_(Bibliothek)`. Letzte Überprüfung: 21. Juli 2018.

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS