UNIVERSITÄT ZU LÜBECK
INSTITUTE OF MATHEMATICS AND
IMAGE COMPUTING

Bildregistrierung mit dem Dämonenalgorithmus

Bachelorarbeit

im Rahmen des Studiengangs
Mathematik in Medizin und Lebenswissenschaften
der Universität zu Lübeck

vorgelegt von
Miriam Otto

ausgegeben und betreut von
Prof. Dr. Jan Modersitzki
Institute of Mathematics and Image Computing

mit Unterstützung von
Dr. Kanglin Chen und Alexander Derksen, M. Sc.
Institute of Mathematics and Image Computing

Lübeck, den 07. Oktober 2014

MIC

IM FOCUS DAS LEBEN

Erklärung

Ich versichere, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Lübeck, den 07. Oktober 2014

Inhaltsverzeichnis

1 Einleitung

Bildregistrierung ist eine wichtige Methode aus dem Bereich der Bildverarbeitung. Das Registrierungsproblem besteht darin, dass sich zwei Bilder, welche sich in Aufnahmezeitpunkt, Aufnahmeort oder Aufnahmeverfahren unterscheiden, durch eine gesuchte Transformation einander möglichst „ähnlich“ werden sollen. Dieses Problem wird gelöst, indem korrespondierende Strukturen in den beiden Bildern überlagert werden. Dazu wird in dieser Arbeit der sogenannte *Dämonenalgorithmus* [28] vorgestellt. Dieser basiert auf der Annahme, dass sogenannte Dämonen auf den Bildpunkten sitzen, deren Aufgabe es ist, das Verhalten der gesuchten Transformation zu kontrollieren.

Das Ziel dieser Arbeit ist es, den klassischen Dämonenalgorithmus nach Thirion mit dem diffeomorphen Dämonenalgorithmus [32] zu vergleichen.

Die vorliegende Arbeit gliedert sich in fünf Teile. Im Anschluss an diesen ersten einleitenden Teil sollen im zweiten Kapitel die Grundlagen der Bildregistrierung sowie Ideen verschiedener Ansätze zur Lösung des Registrierungsproblems vorgestellt werden. Im dritten Kapitel wird das Grundgerüst der Bildregistrierung diskretisiert. Dabei wird auf Interpolationsmethoden, Diskretisierung der Ableitung und den Multilevelansatz zur Registrierung eingegangen. Im anschließenden vierten Kapitel folgt die detaillierte Beschreibung des Dämonenalgorithmus. Die Kräfte werden mit Hilfe des optischen Flusses [16] hergeleitet und die diffeomorphe Erweiterung wird erläutert. Im letzten Kapitel werden Ergebnisse der Implementierung des Dämonenalgorithmus vorgestellt und der additive Algorithmus mit dem diffeomorphen verglichen.

Diese Arbeit zeigt, dass der Dämonenalgorithmus in der Lage ist, synthetische wie auch medizinische Bilddaten zu registrieren. Der diffeomorphe Dämonenalgortihmus führt zu besseren Ergebnissen bezüglich des SSD-Maßes [19] als der klassische Algorithmus.

2 Bildregistrierung

In diesem Kapitel soll eine kurze Einführung in die Bildregistrierung gegeben werden. Dazu wird zunächst die Motivation anhand einiger Beispiele erläutert. Anschließend werden die mathematischen Grundlagen beschrieben, wobei auch verschiedene Registrierungsansätze vorgestellt werden.

2.1 Motivation

Bei Bildregistrierung handelt es sich um ein wichtiges Verfahren der Bildverarbeitung, das bei vielen Anwendungen zum Einsatz kommt [18]. Ein Beispiel für einen Einsatzbereich ist das sogenannte Stitching [27], wo aus Satellitenbildern aus verschiedenen Perspektiven ein Bild der Umgebung zusammengesetzt wird. Die Umgebung wird so aufgenommen, dass benachbarte Bilder Überlappungen aufweisen, die dann registriert werden, um ein zusammenhängendes Bild zu bekommen. Ein besonders wichtiges Anwendungsfeld ist die Medizin, wo die Bildregistrierung beispielsweise bei der Diagnose oder beim Vergleich von prä- und postoperativ aufgenommenen Bilder eingesetzt wird [18]. Ein konkretes Beispiel [22] soll hier kurz erläutert werden. Um ein Gewebe auf zellulärer Ebene untersuchen und die Struktur erkennen zu können, wird das zu untersuchende Organ präpariert, eingewachst und anschließend in dünne Scheiben geschnitten, die unter dem Mikroskop untersucht werden. Dabei geht jedoch die 3D-Struktur des Organs verloren. Durch paarweise Bildregistrierung der einzelnen benachbarten Scheiben lässt sich diese wieder herleiten. Des Weiteren kann bei Magnetresonanzbildern, die eine lange Aufnahmezeit haben, die Bewegung der Patienten durch Bildregistrierung kompensiert werden [21].

Diese Beispiele verdeutlichen, dass die Bildregistrierung ein wichtiges Verfahren in vielen Anwendungsbereichen, insbesondere in der medizinischen Bildverarbeitung, ist.

2.2 Mathematische Grundlagen

Bei der Bildregistrierung soll ein Templatebild $\mathcal{T}$ so deformiert werden, dass es einem Referenzbild $\mathcal{R}$ möglichst „ähnlich“ ist [19]. Im Folgenden betrachten wir Bilder als hinreichend glatte Abbildungen eines rechteckigen Gebiets Ω in die reellen Zahlen, wobei $\Omega \subset \mathbb{R}^d$ mit $d = 2$ oder $d = 3$ gilt. In dieser Arbeit

beschränken wir uns auf 2-dimensionale Bilder, also

$$\mathcal{T}\colon \Omega \to \mathbb{R}, \quad \mathcal{R}\colon \Omega \to \mathbb{R}, \quad \Omega \subset \mathbb{R}^2.$$

Jedem Bildpunkt $x \in \Omega$ wird ein bestimmter Grauwert $\mathcal{T}(x)$ bzw. $\mathcal{R}(x)$ zugewiesen.

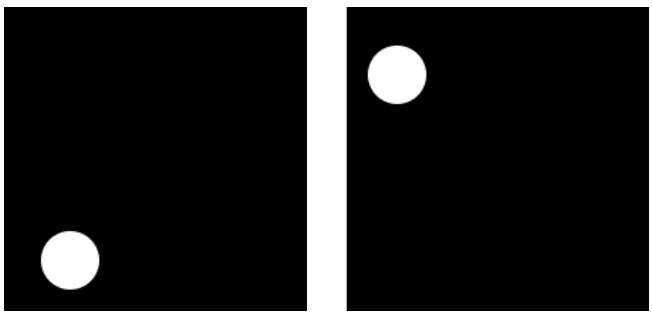
Eine Transformation wird als Abbildung $y\colon \mathbb{R}^2 \to \mathbb{R}^2$ beschrieben, die auf jeden Punkt des zu transformierenden Bildes angewendet wird. Das transformierte Bild ist $\mathcal{T}(y(x)) = \mathcal{T}[y](x) = \mathcal{T}[y]$.

Beschränkt man sich auf monomodale Bilder, also ein Datensatz, der mit nur einem bildgebenden Verfahren entstanden sind, so lässt sich das Problem der Bildregistrierung zusammenfassen als die Suche nach einer Transformation y, sodass idealerweise

$$\mathcal{T}[y] = \mathcal{R}$$

gilt. Das Registrierungsproblem ist die Suche nach einer solchen Transformation y, sodass eine Funktion $\mathcal{J}[y]$ minimiert wird. Diese Funktion misst beispielsweise die Distanz zwischen den einzelnen Bildpunkten des Referenz- und Templatebilds. Idealerweise sollte mit dem gesuchten y $\mathcal{T}[y] = \mathcal{R}$ gelten.

Das Registrierungsproblem ist jedoch nach Hadamard [13] schlecht gestellt, das heißt, es existieren immer mehrere Möglichkeiten, es zu lösen. Nimmt man beispielsweise an, die gesuchte Transformation sei eine Drehung um 90° im Uhrzeigersinn, so kann dieses Problem ebenso durch eine Drehung um 270° im Gegenuhrzeigersinn gelöst werden (vgl Abbildung 2.1). Eine Möglichkeit, das Problem in ein gut gestelltes zu überführen, ist die Einführung einer bestimmten Art von Abbildungen, den Diffeomorphismen, die im weiteren Verlauf dieser Arbeit noch behandelt werden.



(a) (b)

Abbildung 2.1: Beispiel für ein schlecht gestelltes Problem. Das Bild in (b) könnte aus (a) durch eine 90°-Drehung im Uhrzeigersinn, aber auch durch eine 270°-Drehung im Gegenuhrzeigersinn entstanden sein. Abbildung modifiziert aus [19].

Um Bilder auch differenzieren zu können, was für viele Registrierungsverfahren benötigt wird, werden sie im Folgenden als stetig differenzierbare Abbildun-

gen betrachte. Eine genauere Beschreibung der Ableitung von Bildern folgt in Kapitel 3.

2.3 Registrierungsansätze

Das Registrierungsproblem lässt sich mit verschiedenen Ansätzen lösen. Im Folgenden sollen zwei davon kurz vorgestellt werden.

Variationsproblem Bildregistrierung lässt sich als Variationsproblem der Form

$$\{\mathcal{J}[y] = \mathcal{D}[\mathcal{T}[y], \mathcal{R}] + \mathcal{S}[y]\} \xrightarrow{y} \min$$

ansehen [19]. Dabei wird die Funktion $\mathcal{J}[y]$ über alle möglichen Transformationen y minimiert. Hier bezeichnet $\mathcal{D}$ ein sogenanntes *Distanzmaß* [19], welches die räumliche Nähe korrespondierender Punkte misst und diese durch eine reelle Zahl beschreibt. Je kleiner diese Zahl ist, desto ähnlicher sind sich die Bilder in diesem Sinne. Die Funktion $\mathcal{S}[y]$ ist ein sogenannter *Regularisierer* [19], der das schlecht gestellte Problem in ein gut gestelltes überführt [19]. Des Weiteren soll er die Plausibilität einer Lösung y gewährleisten. Plausibilität bedeutet in diesem Zusammenhang, dass die Deformationen im Bild auch in der Realität möglich sein sollten; so sollten beispielsweise keine Punkte übereinander verschoben werden.

Dämonenansatz Ein weiterer Ansatz zur Lösung des Registrierungsproblems ist die Formulierung als Dämonenansatz [28]. Dieser basiert auf dem *optischen Fluss* [16], einem visuellen Phänomen aus der Natur. Der in dieser Arbeit behandelte Dämonenalgorithmus basiert auf diesem Ansatz. Auf die Bildpunkte eines der beiden Bilder werden sogenannte *Dämonen* gesetzt, die den optischen Fluss bestimmen und so die Bildpunkte verschieben. Obwohl dieser Ansatz nicht als Optimierungsproblem formuliert wurde, lässt sich zeigen, dass er dennoch als solches verstanden und gelöst werden kann [31], [30]. Eine detaillierte Beschreibung des Dämonenalgorithmus folgt in Kapitel 4.

3 Diskretisierung und Interpolation

Die Bilder $\mathcal{T}$ und $\mathcal{R}$ liegen nun allerdings in Form diskreter Datenpunkte vor und nicht als kontinuierliche Abbildung, wie in der Definition gefordert. Um eine kontinuierliche Abbildung zu bekommen, wird das diskrete Bild interpoliert. Dazu wird der Bildbereich $\Omega = (\omega^1, \omega^2) \times (\omega^3, \omega^4)$ in ein äquidistantes Gitter der Schrittweite h eingeteilt. Die gegebenen n Bildwerte zu den Punkten $x_j = [x_j^1, x_j^2], j = 1, ..., n$ werden in der Mitte der Zellen, das bedeutet an den Stellen $x_j = [\omega^1 + \frac{j-0.5}{h}, \omega^3 + \frac{j-0.5}{h}]$, angenommen, was man als *zellzentriertes Gitter* [19] bezeichnet. Für weitere Arten von Gittern siehe [19]. Um nun an jedem beliebigen Punkt des Bildbereichs einen Bildwert zu bekommen, wird das Bild auf dem Gitter interpoliert. Im Folgenden sollen einige Methoden dazu vorgestellt und verglichen werden. In dieser Arbeit gelte von nun an stets $h = 1$ und $\Omega = (0, n) \times (0, m)$ mit $n, m \in \mathbb{N}$.

3.1 Interpolationsmethoden

Interpolation ist ein wichtiger Bestandteil der Bildverarbeitung. In der Literatur findet man Interpolation von Bildern mit einem Spline-Ansatz beispielsweise in [29]. Auch in [4] und [33] wird die Spline-Interpolation behandelt.

Um die Grundidee der Interpolation und der verschiedenen Ansätze zu verstehen, bietet es sich an, zunächst die Interpolation nur im Eindimensionalen auf einem Intervall $I \subset \mathbb{R}$ zu betrachten und anschließend auf die zweidimensionale Anwendung zu übertragen. In diesem Abschnitt werden nun drei gängige Interpolationsansätze eindimensional beschrieben und verglichen. Die im Dämonenalgorithmus verwendete lineare Interpolation wird auch in zwei Dimensionen beschrieben.

Bei der Interpolation soll aus bereits bekannten Datenpunkten, den sogenannten Stützstellen, eine Funktion zur Bestimmung der Funktionswerte beliebiger anderer Punkte des Intervalls ermittelt werden. Diese Funktion $f\colon I \to \mathbb{R}$ wird Interpolante genannt und soll an den Stützstellen $x_i \in I$, $i = 1, ..., n$ die Bedingung

$$f(x_i) = \mathcal{B}(x_i)$$

erfüllen, wobei $\mathcal{B}(x_i)$ dem gegebenen Funktionswert an der Stützstelle entspricht. Nach [29] kann man die Interpolante als Linearkombination sogenannter Basisfunktionen $\varphi\colon \mathbb{R} \to \mathbb{R}$ darstellen. Diese Basisfunktionen werden zum

Punkt x_i hinverschoben sodass

$$f(x) = \sum_{i=1}^{n} c_i \varphi(x - x_i) \quad \text{für alle} \quad x \in I \tag{3.1}$$

gilt. Die $c_i \in \mathbb{R}$ sind hier Koeffizienten, die durch die vorgegebenen Werte des gegebenen Intervalls bestimmt werden. Es gibt nun verschiedene Möglichkeiten, die Basisfunktionen zu wählen. Die hier betrachteten Basisfunktionen zeichnen sich durch einen kompakten Träger aus. Als Träger einer Funktion bezeichnet man die abgeschlossene Menge aller Punkte des Definitionsbereichs, an denen die Funktion einen anderen Wert als 0 annimmt [7]. Ist dieser Träger kompakt, bedeutet dies, dass die Funktion nur auf einem beschränkten und abgeschlossenen Intervall nicht 0 ist. Dass die Basisfunktionen einen kompakten Träger haben, ist bei der Interpolation von Vorteil. Er bewirkt, dass für einen beliebigen Punkt $x \in I$ der Funktionswert $f(x)$ unter Verwendung nur weniger Basisfunktionen bestimmt werden kann. Drei in der Bildregistrierung häufig verwendete Arten von Basisfunktionen sind in Abbildung 3.1 dargestellt.

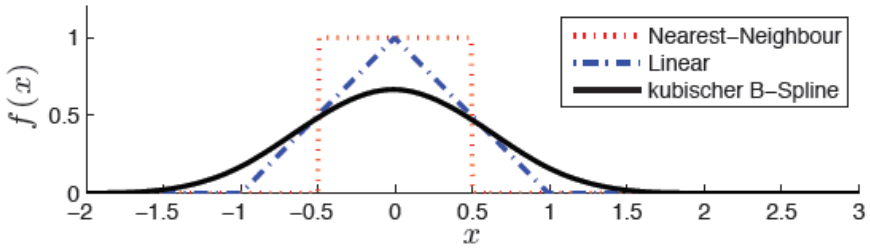


Abbildung 3.1: Basisfunktionen der Nearest-Neighbour-, linearen und kubischen Interpolation. Abbildung modifiziert aus [29].

B-Splines Bei *B-Splines* handelt es sich um genau die oben beschriebene stückweise Interpolationsmethode anhand von Basisfunktionen [23]. Ein B-Spline vom Grad n ist $(n-1)$-mal stetig differenzierbar und es werden $n+1$ umliegende Punkte zur Berechnung des neuen Funktionswerts benötigt [29]. Besonders die kubischen Splines sind eine gängige Methode bei Verfahren, die die zweite Ableitung benötigen [19].

Definition 3.1 (Faltung, [7])**.** *Seien* $f, g \in \mathcal{L}_1(\mathbb{R}) = \{h \colon \mathbb{R} \to \mathbb{R} : h \text{ ist messbar} \wedge \int_{\mathbb{R}} |h(x)|dx < \infty\}$*. Dann ist die* ***Faltung*** $f * g$ *definiert durch:*

$$(f * g)(y) := \int_{\mathbb{R}} f(x)g(y - x)dx.$$

Der B-Spline der Ordnung k entsteht durch Faltung der Rechteckfunktion (siehe Nearest-Neighbour-Interpolation) mit dem Spline der Ordnung $k - 1$ [4]:

$$\varphi^k(x) = (\varphi^0 * \varphi^{k-1})(x).$$

Hier sollen nun die B-Spline-Interpolation mit $n = 0$, welche der Nearest-Neighbour-Methode entspricht, sowie mit $n = 1$, was der linearen Interpolation entspricht, vorgestellt werden. Anhand dieser beiden Interpoaltionsmethoden niedriger Ordnung sollen Vor- und Nachteile bei der Bildregistrierung erläutert werden.

Nearest-Neighbour-Interpolation Bei der Nearest-Neighbour-Interpolation wird jedem $x \in I$ der Funktionswert $f(x_i)$ der nächstgelegenen Stützstelle zugewiesen; es ergibt sich für jede Zelle ein konstanter Funktionswert. Die Basisfunktion ist durch

$$\varphi^{\mathrm{NN}}(x) = \begin{cases} 1 & \text{für} \quad x \in (-\frac{1}{2}, \frac{1}{2}] \\ 0 & \text{sonst} \end{cases}$$

gegeben, was in (3.1) zu $c_i = \mathcal{B}(x_i)$ führt. Die Nearest-Neighbour-Interpolation ist an den Stützstellen nicht stetig und somit auch nicht differenzierbar. Zwischen den Stützstellen ist sie zwar differenzierbar, jedoch ist die Ableitung überall 0. Mit dieser Eigenschaft ist sie für Registrierungsansätze, bei denen die Ableitung benötigt wird, nicht geeignet. Der hier behandelte Dämonenalgorithmus zählt zu den ableitungsbasierten Registrierungsansätzen [28], sodass auch für diesen Ansatz die Nearest-Neighbour-Interpolation nicht zu empfehlen ist [34].

Lineare Interpolation Bei der linearen Interpolation wird der Funktionswert aus den zwei umliegenden Punkten gebildet. Sie hat die auch als „Hutfunktion“ [29] bezeichnete Basisfunktion

$$\varphi^{\mathrm{lin}} = \begin{cases} 1 + x & \text{für} \quad x \in (-1, 0] \\ 1 - x & \text{für} \quad x \in (0, 1] \\ 0 & \text{sonst.} \end{cases}$$

Die lineare Interpolation ist zwar stetig, jedoch wie die Nearest-Neighbour-Interpolation nur stückweise differenzierbar. Der Unterschied zur Nearest-Neighbour-Interpolation besteht darin, dass die Ableitung der linearen Interpolation auch Werte annimmt, die von 0 verschieden sind. Somit eignet sie sich für ableitungsbasierte Registrierungsansätze besser als die Nearest-Neighbour-Interpolation. An den Stützstellen, wo die Funktion nicht differenzierbar ist, wird einer der beiden Werte am Rand gewählt.

Bilineare Interpolation Die bilineare Interpolation entspricht der linearen Interpolation im Zweidimensionalen. Sie ist gut zu implementieren [34] und wurde auch von Thirion in [28] für seinen Dämonenalgorithmus vorgeschlagen. Sie weist außerdem ein gutes Gleichgewicht zwischen Genauigkeit und rechnerischem Aufwand auf [34]. Aus diesen Gründen wurde sie auch in dieser Arbeit für die Implementierung des Dämonenalgorithmus gewählt. Die gesuchte Interpolante $f\colon \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ kann für gegebene Werte $(x_i, y_j, \mathcal{B}(x_i, y_j)), i =$

$1, ..., n, j = 1, ..., m$ aus Ω als

$$f(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}\varphi_1(x - x_i)\varphi_2(y - y_j) \quad \text{für alle} \quad (x,y) \in \Omega$$

dargestellt werden [23]. Dabei sind $\varphi_1(x)\colon \mathbb{R} \to \mathbb{R}$ und $\varphi_2(y)\colon \mathbb{R} \to \mathbb{R}$ die eindimensionalen Basisfunktionen. Es werde nun ein $n \times m$-Bild betrachtet, auf dem bilinear interpoliert werden soll. Dazu werden $\varphi_1(x) = \varphi^{\text{lin}}(x)$ und $\varphi_2(y) = \varphi^{\text{lin}}(y)$ gewählt und die Interpolante ergibt sich zu

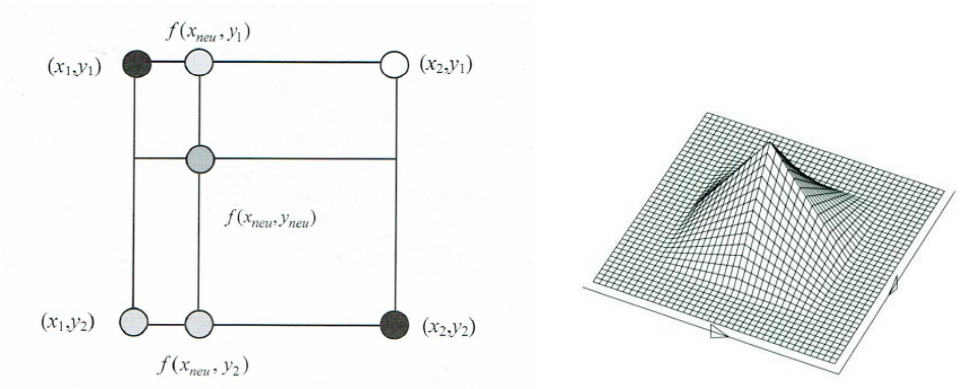
$$f(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}\varphi^{\text{lin}}(x - x_i)\varphi^{\text{lin}}(y - y_j) \quad \text{für alle} \quad (x,y) \in \Omega. \tag{3.2}$$

Das Produkt der Basisfunktionen $\varphi^{\text{lin}}(x)\varphi^{\text{lin}}(y)$ erzeugt analog zur Hutfunktion im Eindimensionalen eine Pyramide im Zweidimensionalen. Die Seitenflächen dieser Pyramide sind jedoch nicht eben, sondern aufgrund der auftretenden quadratischen Terme gekrümmt [23] (siehe Abbildung 3.2).

Es bezeichne nun $(x_{\text{neu}}, y_{\text{neu}}) \in \Omega$ den Punkt, an dem die Interpolante ausgewertet werden soll, sowie $(x_i, y_i) \in \Omega, i = 1, 2$ mit $(x_1, y_1) = (\lfloor x_{\text{neu}} \rfloor, \lfloor y_{\text{neu}} \rfloor)$ und $(x_2, y_2) = (\lceil x_{\text{neu}} \rceil, \lceil y_{\text{neu}} \rceil)$ die nächstgelegenen Gitterpunkte. Dann ist der neue Funktionswert $f(x_{\text{neu}}, y_{\text{neu}})$ nach Berechnung der Doppelsumme explizit gegeben durch

$$\begin{aligned} f(x_{\text{neu}}, y_{\text{neu}}) = {} & \mathcal{B}(x_1, y_1)(x_2 - x_{\text{neu}})(y_2 - y_{\text{neu}}) + \mathcal{B}(x_1, y_2)(x_2 - x_{\text{neu}})(y_{\text{neu}} - y_1) \\ & + \mathcal{B}(x_2, y_1)(x_{\text{neu}} - x_1)(y_2 - y_{\text{neu}}) + \mathcal{B}(x_2, y_2)(x_{\text{neu}} - x_1)(y_{\text{neu}} - y_1), \end{aligned}$$

wobei in (3.2) $c_{ij} = \mathcal{B}(x_i, y_j)$ gilt. Somit wird bei der bilinearen Interpolation der neue Bildwert durch die Werte der vier benachbarten Punkte berechnet [34].



(a) (b)

Abbildung 3.2: Illustration der bilinearen Interpolation. (a) Ermittlung des neuen Funktionswerts aus den vier umliegenden Punkten. (b) Bilinearer Basisspline. Beide Abbildungen modifiziert aus [23].

3.2 Multilevel-Registrierung

Bei der Multilevel-Registrierung [19] handelt es sich um ein Verfahren, das das Registrierungsproblem zunächst auf einer groben Auflösung löst und diese Lösung als Startwert für eine neue Registrierung auf einer höheren Auflösung verwendet. Die Lösungen auf groben Auflösungen beschreiben die Registrierung der groben Strukturen des Referenz- und Templatebilds. Die Lösung auf einer feinen Auflösung beschreibt die Registrierung der Details der beiden Bilder. Durch dieses Verfahren wird die Zielfunktion der Optimierung geglättet, was dazu führt, dass die Optimierung weniger häufig in lokalen Minima endet [19]. Dadurch endet das Verfahren öfter im globalen Minimum, was insgesamt zu besseren Registrierungsergebnissen führt. Auch die Konvergenz sowie die Laufzeit der Algorithmen profitieren im Allgemeinen von der Multilevel-Registrierung [11]. Wird ein Bild auf eine höhere Aulösung interpoliert, so bezeichnet man dies als Prolongation. Als Restriktion bezeichnet man das Verfahren, welches die Auflösung eines Bildes verringert.

Restriktion Die Restriktion des Bildes lässt sich durch Multiplikation mit einer bestimmten Matrix realisieren. Es gibt verschiedene Ansätze, die neuen Datenpunkte zu bestimmen; in dem hier vorgestellten Fall werden immer vier umliegende Punkte gewichtet zu einem neuen zusammengefasst. Am Rand wird angenommen, dass noch ein Punkt neben dem Randpunkt mit demselben Wert existiert. Das würde bedeuten, dass die Ableitung im Randpunkt in diese Richtung 0 ist [23]. Diese Randbedingung bezeichnet man als homogene Neumann-Randbedingung [23].

Wir betrachten zunächst den eindimensionalen Fall, bei dem aus einem Intervall mit $n \in \mathbb{N}$ Punkten $n/2$ werden sollen. Zur Vereinfachung wird davon ausgegangen, dass $n = 2^k, k \in \mathbb{N}$ gilt. Sollte n keine Zweierpotenz sein, müsste man vor jeder Restriktion überprüfen, ob es sich ohne Rest durch 2 teilen lässt. Der Restriktionsoperator R_n ist hier durch

$$R_n = \frac{1}{8}\begin{pmatrix} 4 & 3 & 1 & & & \\ 1 & 3 & 3 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 3 & 3 & 1 \\ & & & 1 & 3 & 4 \end{pmatrix} \quad \in \mathbb{R}^{\frac{n}{2} \times n}$$

gegeben [19]. Die Übertragung ins Zweidimensionale erfolgt durch Anwendung des Kronecker-Produkts [9] (Definition 3.2) . Hier soll ein Bild der Größe $n \times m$ auf die Größe $\frac{n}{2} \times \frac{m}{2}$ restringiert werden, das heißt aus nm Punkten werden $\frac{nm}{4}$.

Definition 3.2 (Kronecker-Produkt)**.** *Sei $A \in \mathbb{R}^{m \times n}$ und $B \in \mathbb{R}^{p \times q}$. Dann*

heißt die Matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1m}B \\ \vdots & & \vdots \\ a_{n1}B & \dots & a_{nm}B \end{pmatrix} \quad \in \mathbb{R}^{mp \times nq}$$

das ***Kronecker-Produkt*** *der Matrizen A und B.*

Mit dem Kronecker-Produkt ist der zweidimensionale Restriktionsoperator durch

$$\hat{R}_{nm} = (R_{\frac{n}{2}} \otimes I_m)(I_{2n} \otimes R_{\frac{m}{2}}) \quad \in \mathbb{R}^{\frac{nm}{4} \times nm}$$

definiert, wobei $I_n \in \mathbb{R}^{n \times n}$ die Einheitsmatrix bezeichne. Nun werden die Spalten des Bildes untereinander geschrieben und der Restriktionsoperator wird von links an dieses als Vektor geschriebene Bild multipliziert. Dies wird dadurch auf eine Größe von $\frac{nm}{4}$ restringiert.

Prolongation Bei der Prolongation ist ein analoges Vorgehen möglich und es wird zunächst wieder nur der eindimensionale Fall betrachtet. Der hier verwendete eindimensionale Prolongationsoperator fasst jeweils zwei benachbarte Punkte unterschiedlich gewichtet zu einem zusammen. Dabei wird jeder Punkt vierfach verwendet, sodass die doppelte Anzahl an Punkten entsteht. Der Prolongationsoperator entspricht dem transponierten Restriktionsoperator multipliziert mit zwei. Damit gelten am Rand wieder die homogenen Neumann-Randbedingungen und der Prolongationsoperator ist durch

$$P_n = 2R_{2n}^{\top} = \frac{1}{4} \begin{pmatrix} 4 & & & & & & \\ 3 & 1 & & & & & \\ 1 & 3 & & & & & \\ & 3 & 1 & & & & \\ & 1 & 3 & & & & \\ & & & \ddots & & & \\ & & & & 3 & 1 & \\ & & & & 1 & 3 & \\ & & & & & 4 & \end{pmatrix} \quad \in \mathbb{R}^{2n \times n}$$

gegeben. Auf das Zweidimensionale übertragen ergibt sich mit dem Kronecker-Produkt die Matrix

$$\hat{P}_{nm} = (P_n \otimes I_{2m})(I_n \otimes P_m) \quad \in \mathbb{R}^{4nm \times nm},$$

die analog zur Restriktion von links an das als Vektor geschriebene Bild multipliziert wird. Bei der Prolongation ändert sich dadurch die Größe des Bildes von $n \times m$ auf $2n \times 2m$ Punkte, das bedeutet, dass die Anzahl der Punkte vervierfacht wird.

3.3 Diskretisierung der Ableitung einer zweimal differenzierbaren Funktion

Da wir Bilder wie in Kapitel 2 als stetig differenzierbare Abbildungen betrachten, besitzen sie auch eine Ableitung, die für viele Registrierungsalgorithmen benötigt wird. Bei der Diskretisierung dieser Ableitung gibt es unter anderem die Möglichkeit, Rückwärts- oder Vorwärtsdifferenzenquotienten zu verwenden [23]. In dieser Implementierung wird die Rückwärtsdiskretisierung verwendet. Dazu betrachten wir die Taylorentwicklung [6] einer Funktion $f(x-h)\colon \mathbb{R} \to \mathbb{R}$ mit $f \in C^2(\mathbb{R})$ an der Entwicklungsstelle $x_0 = x$ bis zur ersten Ordnung:

$$f(x-h) = f(x) - hf'(x) + R_2(x).$$

Dabei bezeichnet $R_2(x)$ das Restglied der zweiten Ordnung. Diese Formel stellt man nach $f'(x)$ um und erhält mit

$$f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{1}{h}R_2(x)$$

den Rückwärtsdifferenzenquotienten der Funktion f. Der Ausdruck

$$\mathcal{D}^-(x) := \frac{f(x) - f(x-h)}{h} \tag{3.3}$$

wird als diskretisierte Ableitung im Punkt x verwendet und $\frac{1}{h}R_2(x)$ ist der Fehlerterm der Approximation. Dieser lässt sich nun noch durch

$$\frac{1}{h}|R_2(x)| \leq \frac{1}{h}\left|\frac{1}{2}f''(x)h^2\right| = \frac{h}{2}|f''(x)|$$

abschätzen [6] und man kann eine von h unabhängige Konstante $C > 0$ finden, sodass

$$h\frac{1}{2}|f''(x)| \leq hC$$

gilt. Nach [23] besitzt der Rückwärtsdifferenzenquotient die Konvergenzordnung eins.

4 Die Dämonenalgorithmen von Thirion und Vercauteren

In diesem Kapitel wird der Dämonenalgorithmus in zwei Variationen [28], [32] vorgestellt. Dieser Algorithmus basiert auf zwei physikalischen Prinzipien, der Diffusion sowie dem optischen Fluss, die miteinander verknüpft werden. Um das Vorgehen besser nachvollziehen zu können, wird darum im Folgenden zunächst das Prinzip des optischen Flusses erläutert. Anschließend wird dies auf den Dämonenalgorithmus übertragen. Im letzten Abschnitt des Kapitels werden Diffeomorphismen und ihre Berechnung erklärt und außerdem die Variante des Algorithmus, die zu diffeomorphen Transformationen führt, vorgestellt.

4.1 Optischer Fluss

Bei *optischem Fluss* handelt es sich um ein visuelles Prinzip aus der Natur, welches auch in der Bildverarbeitung häufig angewendet wird [28]. Bewegungen im Raum können nur als eine Projektion auf ein 2D-Bewegungsfeld wahrgenommen werden, die jedoch nicht exakt gemessen werden kann. Es kann lediglich die scheinbare Bewegung, die Wahrnehmung der Veränderung der Bildintensitäten, geschätzt werden. Diese Schätzung nennt man den optischen Fluss (siehe Abbildung 4.1). Er beschreibt in einem Video die Veränderung der Intensitäten in zwei aufeinanderfolgenden Bildern. Diese Bewegung wird als Vektorfeld dargestellt. Der optische Fluss und das wirkliche Bewegungsfeld sind nicht immer gleich. Betrachtet man beispielsweise eine sich drehende Kugel mit einer gleichmäßigen Oberfläche, so bewegt sie sich mit einem Bewegungsfeld in Richtung der Rotation [2]. Da die Oberfläche gleichmäßig ist, nimmt man jedoch keine Bewegung wahr, der optische Fluss ist also 0. Ein weiteres Beispiel ist in Abbildung 4.2 dargestellt.

In der Natur wird das Prinzip des optischen Flusses besonders von fliegenden Insekten genutzt [26]. Fliegt ein Insekt beispielsweise genau auf ein Hindernis zu, so wird der optische Fluss im Bereich des Hindernisses größer. Um Kollisionen zu vermeiden, wird die Flugbahn so korrigiert, dass sich der optische Fluss verringert. Abstände können ebenfalls durch das Prinzip des optischen Fluss geschätzt werden. Wird der optische Fluss größer, so kommt ein Objekt näher. Die Fluggeschwindigkeit kann dann wieder so angepasst werden, dass der optische Fluss verringert wird. Aber auch in der Bildverarbeitung ist der optische Fluss heutzutage eine wichtige Methode. Er wird beispielsweise beim Bau von Robotern eingesetzt, um ihnen eine Wahrnehmung der Umgebung zu

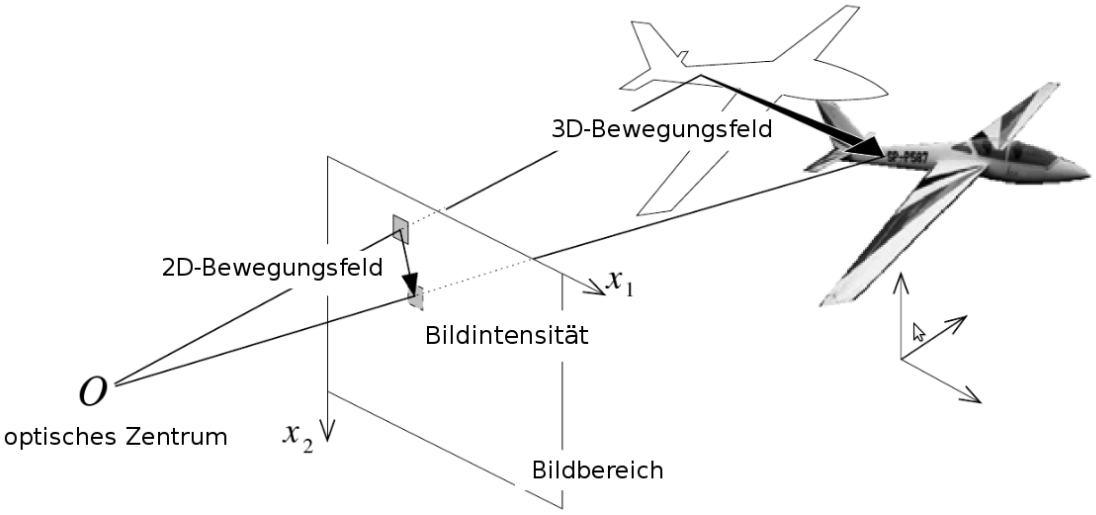


Abbildung 4.1: Illustration des optischen Flusses: Die 3D-Bewegung des Flugzeugs wird vom optischen Zentrum als Projektion auf das 2D-Bewegungsfeld wahrgenommen. Die Bildintensität kann geschätzt werden. Abbildung entnommen aus [2].

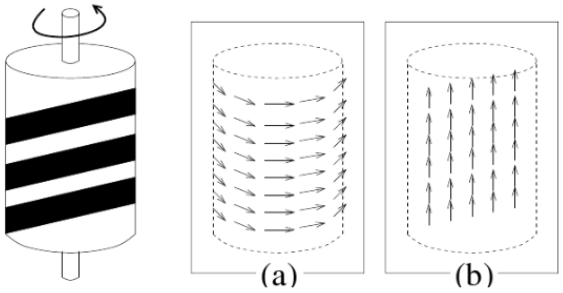


Abbildung 4.2: Beispiel für optischen Fluss: Das Bewegungsfeld einer Barbiersäule ist eine Drehung um die eigene Achse (a), während der optische Fluss nach oben zeigt (b).Entnommen aus [2].

ermöglichen [25].

Um den optischen Flusses zu berechnen, wird angenommen, dass die Intensität eines sich bewegenden Objekts über die Zeit konstant bleibt [2]. Es bezeichne $\mathcal{I}(x_1, x_2, t)\colon \Omega \times \mathbb{R} \to \mathbb{R}$ die Intensität eines Bildes im Punkt $x = (x_1, x_2) \in \mathbb{R}^2$ zum Zeitpunkt t. Dann wird angenommen, dass

$$\frac{\partial \mathcal{I}(x, y, t)}{\partial t} = 0$$

gilt [16]. Unter Anwendung der Kettenregel für Ableitungen bedeutet dies

$$\begin{aligned} & \frac{\partial \mathcal{I}}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial \mathcal{I}}{\partial x_2}\frac{\partial x_2}{\partial t} && = -\frac{\partial \mathcal{I}}{\partial t} \\ \Leftrightarrow \quad & \underbrace{\left(\frac{\partial \mathcal{I}}{\partial x_1}, \frac{\partial \mathcal{I}}{\partial x_2}\right)}_{=:\nabla \mathcal{I}} \underbrace{\begin{pmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{pmatrix}}_{=:v^\top} && = -\underbrace{\frac{\partial \mathcal{I}}{\partial t}}_{=:\mathcal{I}_t} \end{aligned}$$

Der Vektor v entspricht der Geschwindigkeit der Bewegung des Punktes, da diese als Weg pro Zeit definiert ist [14]. $\nabla \mathcal{I}$ ist die Ableitung der Intensität nach dem Ort und $\mathcal{I}_t$ die nach der Zeit. Somit gilt

$$v^\top \nabla \mathcal{I} = -\mathcal{I}_t.$$

Diskretisiert man die Zeit t, so kann die dann ebenfalls diskrete Funktion $\mathcal{I}$ auch als Video, also als eine Abfolge von Bildern, interpretiert werden. Die einzelnen Zeitpunkte t entsprechen dann den einzelnen Bildern des Videos.

Mit Hilfe der Diskretisierung lässt sich nun $\mathcal{I}_t$ noch anders ausdrücken. Wir verwenden hier Gleichung (3.3) für den Rückwärtsdifferenzenquotienten, wobei in diesem Fall $h = \Delta t$ gilt. Diese Formel kann ohne Weiteres auf diese mehrdimensionale Funktion übertragen werden, da in der eindimensionalen Variable t diskretisiert wird [7]. Nimmt man ohne Beschränkung der Allgemeinheit $\Delta t = 1$ an, so ergibt sich

$$\mathcal{I}_t(x,t) \approx \mathcal{I}(x,t) - \mathcal{I}(x,t-1),$$

was insgesamt zur *Gleichung des optischen Flusses*

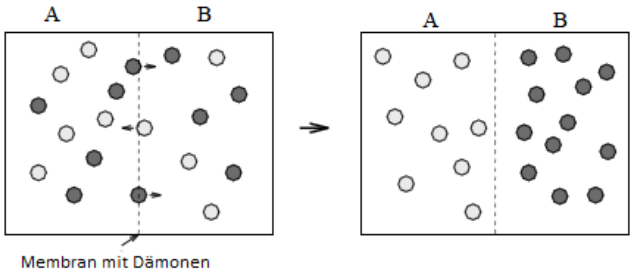
$$v^\top \nabla \mathcal{I} \approx -(\mathcal{I}(x,t) - \mathcal{I}(x,t-1)) \tag{4.1}$$

führt. Man sieht, dass aus dieser Gleichung lediglich der optische Fluss in Richtung des Gradienten der Intensität $\nabla \mathcal{I}$ bestimmt werden kann. Dieses Problem wird als Blendenproblem [17] bezeichnet und tritt auf, wenn eine Bewegung nicht vollständig sichtbar, also durch eine Blende abgedeckt ist. Dann kann der optische Fluss nur in Richtung des Gradienten des sichtbaren Teils wahrgenommem werden, während die Bewegung des gesamten Objekts eigentlich in eine andere Richtung erfolgen kann.Die Gleichung des optischen Flusses (4.1) wird im folgenden Abschnitt als Grundlage zur Berechnung der Kräfte im Dämonenalgorithmus verwendet.

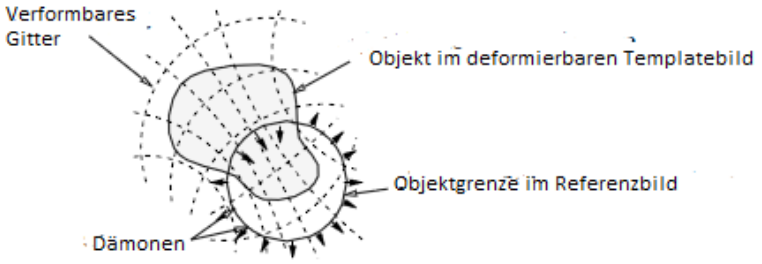
4.2 Thirions Dämonenalgorithmus

Im Folgenden soll der Dämonen-Algorithmus von Thirion [28] vorgestellt werden. Hierbei handelt es sich um eine Registrierungsmethode, die auf der Illustration eines thermodynamischen Paradoxons [14] basiert. Zur Illustration dieses Paradoxons betrachtet man zwei verschiedene Gase in einem Behälter, der durch eine semipermeable Membran in zwei Kammern unterteilt ist. An dieser Membran sitzen die sogenannten "Dämonen", die die Teilchen unterscheiden

können und beide Teilchenarten nur in jeweils eine Richtung durchlassen. Die Teilchen diffundieren nun durch die Membran, sodass mit der Zeit die beiden Gase vollständig getrennt sind (vgl. Abbildung 4.3). Da dies eine Abnahme der Entropie S [14] bedeutet, steht dieses Gedankenexperiment scheinbar im Widerspruch zum zweiten Hauptsatz der Thermodynamik. Dieser besagt, dass die Entropie in geschlossenen Systemen nicht abnehmen kann, es gilt stets $dS \geq 0$. Das Paradoxon wurde gelöst, da die Dämonen beim Erkennen der Teilchen so viel Entropie erzeugen, dass die Gesamtentropie des Systems zunimmt oder gleich bleibt [14].



(a)



(b)

Abbildung 4.3: (a) Illustration des Maxwellschen Gedankenexperiments. Links sind die Teilchen der Gase noch gemischt, rechts sind nach Diffusion durch die Membran mit den Dämonen getrennt. (b) Diffusionsmodel. Das Templatebild diffundiert als deformierbares Gitter durch das Referenzbild, an dessen Konturen sich Dämonen befinden. Beide Abbildungen entnommen aus [28]

In Analogie zu den Maxwellschen Dämonen wird nun bei der Bildregistrierung das Templatebild als deformierbares Gitter angenommen und die Punkte des Referenzbildes als semipermeable Membran, an der die Dämonen sitzen (vgl. Abbildung 4.3). Das Gitter „diffundiert"nun durch die Membran und jeder Punkt des Gitters, der auf einen Dämonen trifft, wird von diesem als „innerhalb" oder „außerhalb" gekennzeichnet. Außerdem wird eine Kraft in die

entsprechende Richtung berechnet, mit der die Punkte des Templatebilds verschoben werden [28].

Ein Vorteil des Dämonenalgorithmus ist, dass er relativ einfach zu implementieren und dabei doch eine sehr effektive Registrierungsmethode ist, für die lediglich die erste Ableitung des Bildes benötigt wird [28]. Jedoch müssen die zu registrierenden Objekte in den Bildern überlappen, da ansonsten alle Punkte als „außerhalb“ gekennzeichnet werden. Das führt dazu, dass das Templatebild nur in die Richtung seines Gradienten zusammen- oder auseinandergezogen wird, sich aber nicht bewegt. Nach der Idee des Algorithmus müssen die Strukturen überlappen, damit Diffusion stattfinden kann [28].

4.2.1 Berechnung der Dämonenkräfte

Die Berechnung der Dämonenkräfte erfolgt nach [28] durch die Gleichung des optischen Flusses. Es wird angenommen, dass es sich bei Referenz- und Templatebild um zwei benachbarte Bilder eines Videos handelt, sodass sich dieses Konzept übertragen lässt. Man kann nun in Gleichung (4.1) annehmen, dass die Intensität zum Zeitpunkt $t-1$ der des Referenzbildes und die Intensität zum Zeitpunkt t der des Templatebildes entspricht. Damit gilt $\mathcal{I}(x, t-1) = \mathcal{R}(x)$ sowie $\mathcal{I}(x,t) = \mathcal{T}(x)$ und (4.1) ergibt sich für die Bildregistrierung zu

$$v^\top \nabla \mathcal{T} = -(\mathcal{T} - \mathcal{R}). \tag{4.2}$$

Aus dieser Gleichung lässt sich nun ein Ausdruck für v zur Berechnung der Dämonenkräfte herleiten. Gleichung (4.2) reicht zur Bestimmung von v nicht aus, da es nur eine Gleichung mit zwei Unbekannten, den beiden Komponenten des Vektors v, ist. Aus diesem Grund werden wir nun einen Ausdruck für v unter der Nebenbedingung, dass die Länge von v minimal sein soll, herleiten. Für diese Herleitung nehmen wir an, dass $\nabla \mathcal{T} \neq 0$ gilt.

Zunächst zerlegen wir den Vektor v in einen Teil, der in die Richtung $\nabla \mathcal{T}$ zeigt und einen anderen, der in die orthogonale Richtung $(\nabla \mathcal{T})^\perp$ zeigt:

$$v = a\nabla \mathcal{T} + b(\nabla \mathcal{T})^\perp, \quad a, b \in \mathbb{R}.$$

Nun wird dieser Ausdruck in die linke Seite von (4.2) eingesetzt:

$$\begin{aligned} v^\top \nabla \mathcal{T} = (\nabla \mathcal{T})^\top v &= (\nabla \mathcal{T})^\top (a\nabla \mathcal{T} + b(\nabla \mathcal{T})^\perp) \\ &= a(\nabla \mathcal{T})^\top \nabla \mathcal{T} + \underbrace{b(\nabla \mathcal{T})^\top (\nabla \mathcal{T})^\perp}_{=0} \\ &= a(\nabla \mathcal{T})^\top \nabla \mathcal{T} = a\|\nabla \mathcal{T}\|_2^2. \end{aligned}$$

Somit erhält man die Gleichung

$$a\|\nabla \mathcal{T}\|_2^2 = -(\mathcal{T} - \mathcal{R}),$$

was zu

$$a = \frac{-(\mathcal{T} - \mathcal{R})}{\|\nabla \mathcal{T}\|_2^2}$$

führt. Nun wird die Nebenbedingung, dass die Länge von v minimal sein soll, verwendet. Die Länge von v lässt sich mit der Dreiecksungleichung abschätzen zu

$$\|v\|_2 = \|a\nabla\mathcal{T} + b(\nabla\mathcal{T})^\perp\|_2 \leq \|a\nabla\mathcal{T}\|_2 + \|b(\nabla\mathcal{T})^\perp\|_2.$$

Dieser Ausdruck wird bei a wie zuvor berechnet für $b = 0$ minimal. Somit lässt sich v als

$$v = a\nabla\mathcal{T}$$

darstellen, was zum Ausdruck zur Berechnung der Dämonenkräfte

$$v(x) = -\frac{(\mathcal{T}(x) - \mathcal{R}(x))\nabla\mathcal{T}(x)}{\|\nabla\mathcal{T}(x)\|_2^2}$$

führt.

Diese Gleichung ist jedoch für $\nabla\mathcal{T}(x) = 0$ nicht definiert, was sich durch einen zusätzlichen Summanden im Nenner lösen lässt. Auch mit einer Summe im Nenner lässt sich jedoch nicht sicherstellen, dass nie durch null geteilt wird; dieser Fall muss immer einzeln betrachtet werden. Wählt man als zusätzlichen Summanden die quadrierte Differenz von Template- und Referenzbild $(\mathcal{T}(x) - \mathcal{R}(x))^2$, so muss diese auf jeden Fall 0 sein, damit der Nenner 0 werden kann. Gilt jedoch $\mathcal{T}(x) = \mathcal{R}(x)$, so ist auch der Zähler 0. Setzt man also den Wert der Dämonenkraft im Fall $\|\nabla\mathcal{T}(x)\|_2^2 + (\mathcal{T}(x) - \mathcal{R}(x))^2 = 0$ auf $u(x) = 0$, so ergibt sich eine im Anwendungskontext sinnvolle Kraft; wenn die Bilder gleich sind, brauchen die Punkte nicht mehr verschoben zu werden. Die Dämonenkraft $u\colon \mathbb{R}^2 \to \mathbb{R}^2$ ergibt sich somit letztendlich zu

$$u(x) := -\frac{(\mathcal{T}(x) - \mathcal{R}(x))\nabla\mathcal{T}(x)}{\|\nabla\mathcal{T}(x)\|_2^2 + (\mathcal{T}(x) - \mathcal{R}(x))^2}, \tag{4.3}$$

wobei wie oben beschrieben $u(x) = 0$ gilt, falls $\|\nabla\mathcal{T}(x)\|_2^2 + (\mathcal{T}(x) - \mathcal{R}(x))^2 = 0$. Man sieht, dass $u(x)$ für $\mathcal{T}(x) = \mathcal{R}(x)$ 0 ist, also genau dann, wenn das Registrierungsproblem an diesem Punkt gelöst wurde. Auch der Fall, dass der Nenner 0 wird kann nur eintreten, falls $\mathcal{T}(x) = \mathcal{R}(x)$ erfüllt ist. Die Punkte werden durch die Dämonenkraft ausschließlich in Richtung $\pm\nabla\mathcal{T}$ verschoben. Gilt $\mathcal{T}(x) > \mathcal{R}(x)$, so ist die Richtung $-\nabla\mathcal{T}(x)$; gilt $\mathcal{T}(x) < \mathcal{R}(x)$, so ist die Richtung $\nabla\mathcal{T}$.

Nachdem die Verrückung $u(x)$ nach (4.3) berechnet wurde, wird diese im zweiten Schritt des Algorithmus geglättet. Dies geschieht durch die Faltung mit einem Gauß-Kern der Varianz σ_1^2 und Mittelwert $\mu = 0$. Dieser Gauß-Kern hat die Formel [24]

$$K_{\sigma^2}(x_1, x_2) = \frac{1}{2\pi\sigma} e^{\frac{-(x_1^2+x_2^2)}{2\sigma^2}} \tag{4.4}$$

und beschreibt die zweidimensionale Glockenkurve der Normalverteilung (vgl. Abbildung 4.4 (a)). Bei der Faltung mit dieser wird jeder Punkt durch das gewichtete Mittel seiner ihn umgebenden Punkte ersetzt und so werden die Daten geglättet. Da hier mit diskreten Daten gearbeitet wird, wird (4.4) durch eine

Algorithmus 1 Dämonen-Algorithmus mit Kräften nach (4.3) und additivem Update-Schritt (vgl. [31]).

Sei $V(x)$ das bekannte Startdeformationsfeld (in der ersten Iteration gilt $V(x) = x$). Dann ist eine Iteration:

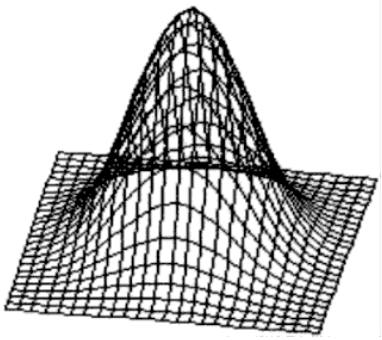
1. Berechne die Dämonenkraft für jeden Bildpunkt x_i nach (4.3):

$$u(x_i) = -\frac{(\mathcal{T}[V](x_i) - \mathcal{R}(x_i))\nabla\mathcal{T}[V](x_i)}{\|\nabla\mathcal{T}[V](x_i)\|_2^2 + (\mathcal{T}[V](x_i) - \mathcal{R}(x_i))^2}.$$

   Die Werte der kontinuierlichen Bilder werden durch Interpolation bestimmt.
2. Erste Glättung: $u \leftarrow K_{\sigma_1^2} * u$
3. Update-Schritt: $V = V + u$
4. Zweite Glättung: $V \leftarrow K_{\sigma_2^2} * V$

Matrix angenähert (siehe Abbildung 4.4 (b)). Die Größe der Matrix beeinflusst, wie stark geglättet wird.

Nach der Glättung der Verrückung erfolgt der Update-Schritt, bei dem die geglättete Verrückung auf das bestehende Vektorfeld addiert wird. Dieses neue Vektorfeld wird dann anschließend nochmals mit einem Gauß-Kern mit $\mu = 0$ geglättet. Da es sich um andere Daten als bei der ersten Glättung handelt, kann hier eine andere Varianz σ_2^2 verwendet werden.



(a)

$$\mathbf{G}_{5\times5} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

(b)

Abbildung 4.4: (a) Die zweidimensionale Glockenkurve der Normalverteilung. (b) Beispiel für eine Matrix zur Annäherung des Gauß-Kerns. Beide Abbildungen entnommen aus [].

Variationen des Dämonenalgorithmus In [28] werden als Variationsmöglichkeiten die Wahl der Dämonen und deren Position, der Raum der Transformationen und die Interpolationsmethode genannt. Im Folgenden sollen immer alle Bildpunkte

als Dämonen gewählt werden, da dies am einfachsten zu implementieren ist. Aus den in Kapitel 3 genannten Gründen wird stets bilinear interpoliert. Im Dämonenalgorithmus kann auch die Berechnung der Kräfte sowie die Art des Updates variiert werden [31].

4.2.2 Berechnung des neuen Deformationsfelds durch SSD-Minimierung

Das neue Deformationsfeld in Algorithmus 1 kann auch durch Minimierung des Distanzmaßes der Summe der quadrierten Differenzen (engl.: *sum of squared differences (SSD)*) berechnet werden [10]. Wie der Name bereits erahnen lässt, ist die Formel des SSD-Maßes eines Deformationsfelds $V : \Omega \to \mathbb{R}$ durch

$$SSD(V) = \frac{1}{2} \int_{\Omega} (\mathcal{T}[V(x)] - \mathcal{R}(x))^2 dx \xrightarrow{V} \min$$

gegeben. Da dies genau dann minimal wird, wenn der Integrand fast überall in Ω minimal ist, kann man das SSD auch punktweise betrachten und für jeden einzelnen Punkt über V minimieren [20], was zum Optimierungsproblem

$$SSD(V(x)) = \frac{1}{2}(\mathcal{T}[V(x)] - \mathcal{R}(x))^2 \xrightarrow{V} \min \quad \text{für fast alle } x \in \Omega \tag{4.5}$$

führt. Die Lösung von (4.5) erfolgt nun mittels eines Gradientenabstiegsverfahrens [20]. Da $\mathcal{T}$ und $\mathcal{R}$ als hinreichend glatt angesehen werden, ist $SSD(V)$ differenzierbar und mit dem Gradientenabstiegsverfahren ist das neue Deformationsfeld V_{n+1} durch

$$V_{n+1}(x) = V_n(x) - \alpha \nabla SSD(V_n(x))$$

gegeben, wobei $\alpha > 0$ die Schrittlänge bezeichnet. Mit $\nabla SSD(V_n(x)) = (\mathcal{T}[V_n(x)] - \mathcal{R}(x))\nabla\mathcal{T}[V_n(x)]$ ergibt sich somit

$$V_{n+1}(x) = V_n(x) - \alpha(\mathcal{T}[V_n(x)] - \mathcal{R}(x))\nabla\mathcal{T}[V_n(x)].$$

Der Term $-\alpha(\mathcal{T}[V_n(x)] - \mathcal{R}(x))\nabla\mathcal{T}[V_n(x)]$ kann als Update und so auch als Verrückung bzw. Dämonenkraft bei additivem Updateschritt interpretiert werden. Setzt man $\alpha = 1/\|\nabla\mathcal{T}[V_n(x)]\|^2 + (\mathcal{T}[V_n(x)] - \mathcal{R}(x))^2$ so erhält man die im vorherigen Abschnitt bereits vorgestellte Dämonenkraft (4.3).

4.3 Der diffeomorphe Dämonenalgorithmus nach Vercauteren

In Abschnitt 4.2 wurden bereits einige Variationen des Dämonenalgorithmus vorgestellt. In diesem Kapitel soll auf eine erstmals in [32] vorgestellte Abwandlung eingegangen werden, die zu diffeomorphen Transformationen führt. Ein Diffeomorphismus ist eine stetig differenzierbare Abbildung, deren Umkehrabbildung existiert und ebenfalls stetig differenzierbar ist. Dies wird im Dämonenalgorithmus durch das Berechnen des sogenannten Exponentials der Verrückung erreicht [30].

4.3.1 Diffeomorphismen und ihre Bedeutung für die Bildregistrierung

In der Bildregistrierung werden diffeomorphe Transformationen häufig als Bedingung für die Plausibilität einer Transformation angesehen. Da sie bijektiv sind, verhindern sie Falten und Zerreißen des Deformationsfelds. Eine Falte entsteht, wenn zwei Punkte auf denselben Funktionswert abbilden; dies wird durch die Injektivität der Transformation vermieden. Das Deformationsfeld zerreißt, wenn auf einen oder mehrere Funktionswerte gar nicht abgebildet wird. Dass dies nicht passiert, stellt die Surjektivität der Transformation sicher. Besonders bei dem Dämonenalgorithmus, bei dem der prinzipiell jede denkbare Transformation möglich ist, sollte darauf geachtet werden, dass die Transformation plausibel ist [30]. Es gibt jedoch auch Fälle, wie beispielsweise Bilder vor und nach einer Tumorresektion oder von verschiedenen Patienten, wo eine diffeomorphe Transformation nicht möglich ist, da die Bilder nicht die gleichen Strukturen darstellen [1]. Eine Möglichkeit, eine diffeomorphe Transformation zu garantieren, wäre, die Verrückung so stark zu glätten, dass sie keine Faltungen und Risse mehr aufweist. Dieser Ansatz ist jedoch nicht zu empfehlen, da bei einer zu starken Glättung die Genauigkeit der Registrierung leidet [5]. Eine alternative Methode zur Bestimmung von Diffeomorphismen wird im nächsten Abschnitt vorgestellt.

In [8] findet man folgende formale Definition für Diffeomorphismen:

Definition 4.1 (Diffeomorphismus)**.** *Es seien $G_1, G_2 \subset \mathbb{R}^n$ Gebiete und $f\colon G_1 \to G_2$ eine stetig differenzierbare Abbildung. f heißt ein **Diffeomorphismus**, wenn f bijektiv ist und $f^{-1}\colon G_2 \to G_1$ existiert und ebenfalls stetig differenzierbar ist.*

4.3.2 Berechnung des Diffeomorphismus

Mit dem Ansatz des optischen Flusses wurde in Kapitel 4.1 die Gleichung

$$\mathcal{T}_t(t,x) + v(t,x) \cdot \nabla \mathcal{T}(t,x) = 0$$

hergeleitet. Zusammen mit der Anfangsbedingung $\mathcal{T}(0,x) = \mathcal{T}_0(x)$ ist dies eine partielle Differentialgleichung erster Ordnung, also eine Differentialgleichung, die erste partielle Ableitungen und keine höheren enthält. Sie wird auch Transportgleichung genannt wird [15] und lässt sich mit der *Methode der Charakteristiken* [15] lösen. Wir werden diese Methode hier nicht komplett anwenden, da die Lösung, das Templatebild $\mathcal{T}(t,x)$, bereits bekannt ist und wir lediglich an der Funktion $v(t,x)$ interessiert sind. Zur Bestimmung dieser wird nur eine Gleichung aus der Methode der Charakteristiken benutzt; für eine genauere Beschreibung der Lösung der Transportgleichung siehe [15]. Nach der Methode der Charakteristiken gilt mit $t \in \mathbb{R}^+$ und $x \in R^2$ die gewöhnliche Differentialgleichung

$$\begin{cases} \frac{\partial}{\partial t}\Phi(t,x) & = v(t,\Phi(t,x)), \\ \Phi(0,x) & = x. \end{cases} \tag{4.6}$$

Die Funktion $\Phi\colon \mathbb{R}^+ \times \mathbb{R}^2 \to \mathbb{R}^2$ ist hierbei die sogenannte Charakteristik, die auf der Trajektorie $(t, \Phi(t,x))$ mit festem x und $t \in \mathbb{R}$ konstant ist. Sie beschreibt außerdem den allgemeinen Fluss zum Vektorfeld v, was in dieser Arbeit den Dämonenkräften entspricht.

Zur Berechnung des diffeomorphen Flusses $\Phi(t,x)$ muss also die Lösung des Anfangswertproblems (4.6) an der Stelle $t = 1$ berechnet werden [1]. Nach [23] lässt sich die Lösung von (4.6) durch

$$\Phi(1,x) = x + \int_0^1 v(s, \Phi(s,x))ds \tag{4.7}$$

berechnen. Diese Lösung $\Phi(1,x) = \exp(v)$ nennt man auch das Exponential des Vektorfelds v und sie ist diffeomorph. Um im Dämonenalgorithmus diffeomorphe Transformationen sicherzustellen und so Faltungen zu vermeiden, wird im Update-Schritt $\exp(u)$ mit dem gegebenen Deformationsfeld kompositiv verknüpft. Bei dieser Art der Verknüpfung bleibt die Transformation diffeomorph, was bei additivem Update nicht garantiert werden kann [1]. Der Rest des Algorithmus bleibt erhalten (vgl. Algorithmus 2).

Algorithmus 2 Diffeomorpher Dämonen-Algorithmus mit Kräften nach (4.3) (vgl. [31]).

Sei $V(x)$ das bekannte Startdeformationsfeld (in der ersten Iteration gilt $V(x) = x$). Dann ist eine Iteration:

1. Berechne die Dämonenkraft für jeden Bildpunkt x_i nach (4.3) Die Werte der kontinuierlichen Bilder werden durch Interpolation bestimmt.
2. Erste Glättung: $u \leftarrow K_{\sigma_1^2} * u$
3. Update-Schritt: $V = V \circ \exp(u)$
4. Zweite Glättung: $V \leftarrow K_{\sigma_2^2} * V$

Da von $v(x)$ nur numerische Werte bekannt sind und keine explizite Formel, kann das Integral aus (4.7) nicht exakt berechnet werden und die Lösung wird numerisch angenähert. Dazu sollen hier im Folgenden zwei Verfahren, das Euler- sowie das Runge-Kutta-Verfahren, vorgestellt und verglichen werden. Im Folgenden sei zur Vereinfachung $\Phi(t,x) = \Phi(t)$ bei konstantem x.

4.3.2.1 Lösen der Differentialgleichung mit dem Euler-Verfahren

In [1] wird das Euler-Verfahren zur Lösung der Differentialgleichung (4.6) verwendet. Zur Herleitung dieser Methode soll Gleichung (4.6) zunächst wie in [23] diskretisiert werden. Wir suchen nun also Näherungen $\varphi_j \approx \Phi(t_j)$ zu $N+1$ äquidistanten Stellen t_j in einem Intervall $[a,b]$. Da in diesem Fall konkret $\Phi(1) = \exp(u)$ gesucht ist, wählen wir das Intervall als $[a,b] = [0,1]$. Dann gilt für die Stützstellen wie in [23]

$$t_j = 0 + hj, \quad j = 0, 1, ..., N, \qquad h = \frac{1}{N},$$

wobei N zunächst beliebig gewählt werden kann. Nun muss noch die Ableitung diskretisiert werden. Wie in Kapitel 3 bereits beschrieben, wird hierzu ein Differenzenquotient, in diesem Fall vorwärts, verwendet, sodass

$$\Phi'(t_{j-1}) \approx \frac{\Phi(t_j) - \Phi(t_{j-1})}{h}$$

gilt. Setzt man dies in (4.6) ein, so erhält man

$$\frac{\Phi(t_j) - \Phi(t_{j-1})}{h} = u(\Phi(t_{j-1})).$$

Stellt man nun nach $\Phi(t_j)$ um, erhält man für die Näherung $\varphi_j = \Phi(t_j)$

$$\varphi_j = \varphi_{j-1} + hu(\varphi_{j-1}). \tag{4.8}$$

So kann man nun sukzessive die Werte im Intervall von links nach rechts berechnen und bekommt schließlich einen Wert für $\Phi(1)$. Das Euler-Verfahren gehört zur Gruppe der *Einschrittverfahren*. Deren Besonderheit liegt darin, dass der neue Wert φ_{j+1} nur von φ_j und nicht von weiteren Vorgängern abhängt. Allgemein gilt für Einschrittverfahren die Formel

$$\varphi_{j+1} = \varphi_j + hf(t_j, \varphi_j, t_{j+1}, \varphi_{j+1})$$

mit $h = t_{j+1} - t_j$ und $\varphi_0 = \Phi(0)$. Die Funktion $f\colon \mathbb{R}^4 \to \mathbb{R}$ wird als *Verfahrensfunktion* bezeichnet [23].

In [1] wird (4.8) als äquivalent zu

$$\varphi_j = (x + hu) \circ \varphi_{j-1}$$

beschrieben. Somit wäre beispielsweise für $N = 8$ die erste Näherung $\varphi_{\frac{1}{8}} = x + \frac{1}{8}u(x)$ und diese könnte nun immer wieder sukzessive mit den höheren Näherungen verknüpft werden:

$$\begin{aligned}
\varphi_{\frac{2}{8}} &= \varphi_{\frac{1}{8}} \circ \varphi_{\frac{1}{8}} \\
\varphi_{\frac{3}{8}} &= \varphi_{\frac{1}{8}} \circ \varphi_{\frac{2}{8}} \\
&\vdots \\
\varphi_1 &= \varphi_{\frac{1}{8}} \circ \varphi_{\frac{7}{8}}.
\end{aligned}$$

Wählt man nun für N eine Zweierpotenz, so kann man die Lösung mit einem sogenannten *scaling and squaring-Ansatz* [1] bestimmen. Bei diesem wird jede Näherung wieder mit sich selbst verknüpft, um die nächsthöhere Näherung zu erhalten:

$$\begin{aligned}
\varphi_{\frac{1}{4}} &= \varphi_{\frac{1}{8}} \circ \varphi_{\frac{1}{8}} \\
\varphi_{\frac{1}{2}} &= \varphi_{\frac{1}{4}} \circ \varphi_{\frac{1}{4}} \\
\varphi_1 &= \varphi_{\frac{1}{2}} \circ \varphi_{\frac{1}{2}}.
\end{aligned}$$

Das Euler-Verfahren ist ein sogenanntes einstufiges Verfahren, da es zur Berechnung von φ_{j+1} nur einen Funktionswert $u(\varphi_j)$ benötigt. Dadurch ist es wenig aufwändig, jedoch auch nicht so genau wie mehrstufige Verfahren [15]. Ein Beispiel für ein mehrstufiges Verfahren ist das Runge-Kutta-Verfahren, welches im nächsten Abschnitt vorgestellt wird.

4.3.2.2 Lösen der Differentialgleichung mit dem Runge-Kutta-Verfahren

Das Runge-Kutta-Verfahren ist ebenfalls eine Methode, um das Anfangswertproblem (4.6) zu lösen. Es gehört wie auch das Euler-Verfahren zur Gruppe der *Einschrittverfahren*. Im Folgenden gelte zur Berechnung von $\Phi(1)$, dass $h = 1$ ist.

Beim Runge-Kutta-Verfahren handelt es sich um ein mehrstufiges Verfahren, bei dem die Verfahrensfunktion als Linearkombination von m Funktionen dargestellt wird. Für das vierstufige Runge-Kutta-Verfahren, welches auch in dieser Implementierung verwendet wurde, gilt das folgende Schema [23]:

$$\begin{aligned} k_1 &:= u(t_j, \varphi_j) \\ k_2 &:= u(t_j + \tfrac{1}{2}, \varphi_j + \tfrac{1}{2}k_1) \\ k_3 &:= u(t_j + \tfrac{1}{2}, \varphi_j + \tfrac{1}{2}k_2) \\ k_4 &:= u(t_j + 1, \varphi_j + k_3) \\ \varphi_{j+1} &= \varphi_j + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned} \tag{4.9}$$

Da in diesem speziellen Fall $j = 0$ sowie $t_0 = 0$ und $\varphi_0 = x$ gilt, vereinfacht sich (4.9) zu

$$\begin{aligned} k_1 &:= u(0, x) \\ k_2 &:= u(\tfrac{1}{2}, x + \tfrac{1}{2}k_1) \\ k_3 &:= u(\tfrac{1}{2}, x + \tfrac{1}{2}k_2) \\ k_4 &:= u(1, x + k_3) \\ \varphi_{j+1} &= \varphi_j + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4). \end{aligned}$$

Da es sich bei dem Runge-Kutta-Verfahren um ein mehrstufiges Verfahren handelt, ist es aufwendiger zu rechnen als zum Beispiel das Euler-Verfahren. Dafür werden jedoch auch mehr Werte miteinbezogen und es ist somit genauer als ein einstufiges Verfahren [15]. Bei der Entstehung dieser Arbeit wurden beide Verfahren getestet. Da das Runge-Kutta-Verfahren bessere Ergebnisse lieferte, ist dies auch Teil der letztendlichen Implementierung.

5 Implementierung und Ergebnisse

Im Folgenden werden die Details der Implementierung des Dämonenalgorithmus beschrieben und diese Implementierung wird an synthetischen sowie medizinischen Bildern getestet. Dabei wird auch der Algorithmus nach Thirion mit additivem Update mit dem diffeomorphen Algorithmus nach Vercauteren verglichen. Außerdem soll der Einfluss des Glättungsparameters untersucht werden.

5.1 Abbruchkriterien

In der Implementierung wurden zwei Abbruchkriterien gewählt. Eines basiert auf der Summe der quadrierten Differenzen, dem SSD-Distanzmaß, welches wie in Kapitel 4 gezeigt durch den Dämonenalgorithmus minimiert wird. Diese Summe der Differenzen soll kleiner als eine Schranke $\tau > 0$ sein, welche als SSD-Wert vor der Registrierung geteilt durch 100 gewählt wird. Das zweite Kriterium greift, falls die Anzahl der bereits durchlaufenen Iterationen k die Anzahl der maximalen Iterationen k_{max} überschreitet. In dieser Implementierung wurde $k_{max} = 500$ gewählt. Ist also eine der Bedingungen

1. $\sum_{i=1}^{n} |\mathcal{T}[y(x_i)] - \mathcal{R}(x_i)|^2 < \tau$
2. $k > k_{max}$

erfüllt, so endet die Registrierung.

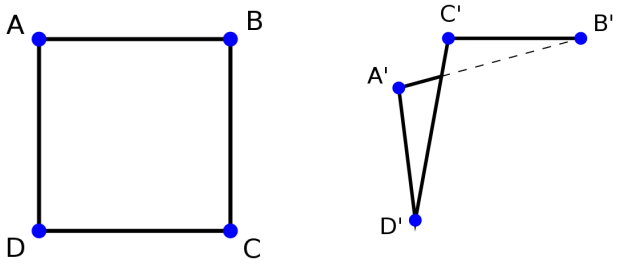
5.2 Parameterwahl

Beim Dämonenalgorithmus können die Parameter der Glättung gewählt werden. Diese sind zum einen die Varianz σ^2 und zum anderen die Größe des Filters. Die Filtergröße wird hier aus Symmetriegründen immer als ungerade Zahl gewählt. Je größer man beide Parameter wählt, umso glatter wird das Deformationsfeld. Dies verhindert zwar Faltungen, bedeutet aber auch Informationsverlust [5]; es muss ein gutes Mittelmaß zwischen beidem gefunden werden. Bei der Multilevelregistrierung empfiehlt es sich, die Varianz und die Filtergröße für jedes Level anzupassen, sodass auf jeder Auflösungsstufe verhältnismäßig gleich stark geglättet wird. In dieser Implementierung wurde die Filtergröße bei einer Bildgröße von $n \times n$ als nächstgrößere ungerade Zahl von $\lceil \frac{n}{20} \rceil$ berechnet. Die Varianz variiert in den einzelnen Abschnitten in Abhängigkeit von der Filtergröße, da unterschiedliche Daten untersucht wurden, die auch unterschiedliche stark streuen. Im Folgenden soll mit σ^2 immer die Varianz auf der

höchsten Auflösungsstufe bezeichnet werden. Zur besseren Vergleichbarkeit der Parameterauswirkung wurde zur Glättung des Updates und zur Glättung des Deformationsfelds der gleiche Parameter σ^2 gewählt.

5.3 Nachweis von faltenfreien Deformationsfeldern

Um nachzuweisen, dass bei der Registrierung keine Faltungen auftreten, wird der Flächeninhalt des Deformationsfelds berechnet. Geometrisch betrachtet bedeutet eine Faltung, dass bei einer Zelle eine Ecke ihre gegenüberliegende Kante überquert (vgl. Abbildung 5.1). Da dies jedoch auch passieren kann ohne dass sich das Vorzeichen des Flächeninhalts ändert, betrachtet man eine Zerlegung jeder Zelle in Dreiecke [12]. Tritt bei Dreiecken eine Faltung auf, ändert sich das Vorzeichen des Flächeninhalts in jedem Fall. Somit werden hier die Flächeninhalte aller vier möglichen Dreiecke (siehe Abbildung 5.2) berechnet und die Vorzeichen überprüft. Kommt kein negatives Vorzeichen vor, so ist die Transformation faltenfrei.

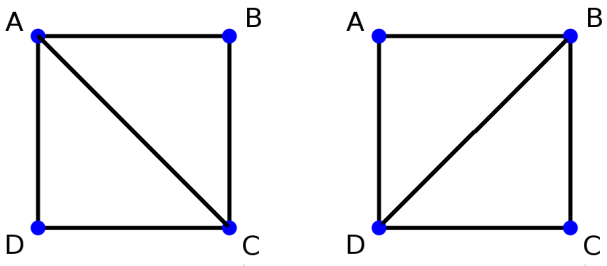


(a) (b)

Abbildung 5.1: (a) Undeformierte Zelle. Die Großbuchstaben bezeichnen die Eckpunkte. (b) Deformierte Zelle mit Faltung. Abbildung modifiziert aus [12].

5.4 Synthetische Bilder

Als synthetische Bilder werden zunächst zwei weiße Kreise auf schwarzem Grund der Auflösung 128×128 gewählt, von denen einer nach rechts unten verschoben ist (siehe Abbildung 5.3). Die Objekte in den Bildern überlappen, sodass der Algorithmus angewendet werden kann. Außerdem zeigen die Bilder dieselben Objekte, weshalb auch eine diffeomorphe Transformation möglich ist. Es soll einmal mit dem diffeomorphen und einmal mit dem additiven Dämonenalgorithmus registriert werden, wobei bei beiden zunächst $\sigma^2 = 1$ gewählt wird. Es wurde mit dem Multilevelansatz auf zwei Auflösungen registriert. An diesem Datensatz soll außerdem die Auswirkung des Glättungsparameters gezeigt werden. Dazu wird das Bild noch zwei weitere Male mit dem diffeomorphen Algorithmus



(a) (b)

Abbildung 5.2: (a) Erste mögliche Triangulation der Zelle. (b) Zweite mögliche Triangulation der Zelle. Insgesamt gibt es vier mögliche Dreiecke, die alle einen positiven Flächeninhalt haben müssen. Abbildung modifiziert aus [12].

registriert, einmal mit $\sigma^2 = 3$ und einmal mit $\sigma^2 = 0,4$. Die Ergebnisse der Registrierungen sind in den Abbildungen 5.4 und 5.5 dargestellt, die Laufzeiten sowie die SSD-Reduktion in Tabelle 5.1.

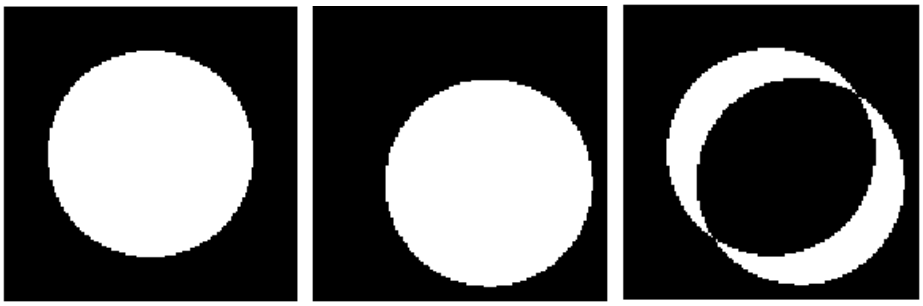


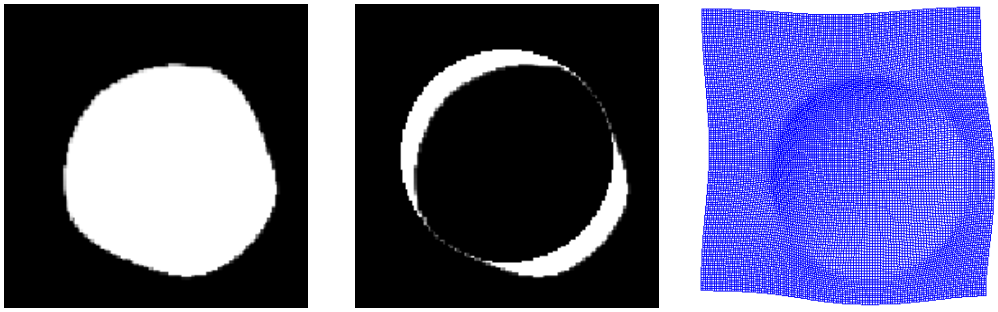
Abbildung 5.3: Synthetische Bilder. Links das Referenz-, in der Mitte das Templatebild, rechts die Differenz vor der Registrierung.

Algorithmus	σ^2	SSD-Reduktion	Laufzeit
additiv	1	55,99%	109,9 s
diffeomorph	1	99,05%	255,5 s
diffeomorph	3	54,27%	273,3 s
diffeomorph	0,4	98,23%	258,3 s

Tabelle 5.1: Vergleich der Registrierungen mit verschiedenem Glättungsparameter in SSD-Reduktion und Laufzeit.

5.5 Medizinische Bilder

Als medizinische Bilder wurden zwei coronale Schichten, die je aus einer Computertomographieaufnahme des Thorax genommen wurden, verwendet [3]. Sie



(a) additiv

(b) diffeomorph

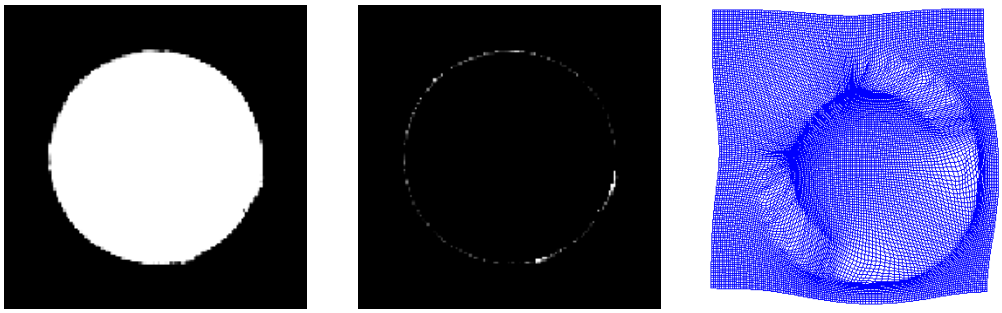
Abbildung 5.4: Ergebnisse der Registrierungen mit $\sigma^2 = 1$. Links die deformierten Templatebilder, in der Mitte die Differenz zwischen Referenz- und Templatebild nach der Registrierung, rechts das Deformationsfeld.

haben die Auflösung 128×128 und wurden bei maximaler bzw. minimaler Einatmung aufgenommen. Da es sich um 2D-Bilder handelt, haben sie keine klinische Relevanz da sich bei der Atmung kann Gewebe der Lunge aus der aufgenommenen Schicht herausbewegt haben kann. An diesen Bildern kann jedoch trotzdem die Funktionsweise des Algorithmus illustriert werden. Diese lässt sich dann ins dreidimensionale übertragen, womit auch Bilder mit klinischer Relevanz registriert werden können [28]. Als Glättungsparameter wurde $\sigma^2 = 0,6$ gewählt und es wurde wieder eine Multilevelregistrierung mit zwei verschiedenen Auflösungen durchgeführt. Wie bereits bei den synthetischen Bildern soll der additive Algorithmus mit dem diffeomorphen verglichen werden. Die Ergebnisse der Registrierungen sind in Abbildung 5.7 dargestellt, die SSD-Reduktion sowie Laufzeiten in Tabelle 5.2.

Algorithmus	σ^2	SSD-Reduktion	Laufzeit
additiv	$0,9$	$61,65\%$	$163,4$ s
diffeomorph	$0,9$	$79,48\%$	$229,3$ s

Tabelle 5.2: Vergleich der Registrierungen in SSD-Reduktion und Laufzeit.

(a) diffeomorph, $\sigma^2 = 0,4$



(b) diffeomorph, $\sigma^2 = 1$

(c) diffeomorph, $\sigma^2 = 3$

Abbildung 5.5: Ergebnisse der Registrierungen mit verschiedenem Glättungsparameter. Links die deformierten Templatebilder, in der Mitte die Differenz zwischen Referenz- und Templatebild nach der Registrierung, rechts das Deformationsfeld.

5.6 Diskussion

An den in diesem Kapitel vorgestellten Bildern sieht man, dass der Dämonenalgorithmus in der Lage ist, synthetische wie auch klinische Bilddaten zu registrieren und das SSD-Maß dieser Daten zu reduzieren. Dabei fallen in beiden Anwendungen Unterschiede zwischen dem additiven und dem diffeomorphen Dämonenalgorithmus auf.

Betrachtet man die Differenzbilder bei der Anwendung auf die synthetischen Daten (Abbildung 5.4), so sieht man, dass die Differenz mit dem diffeomorphen

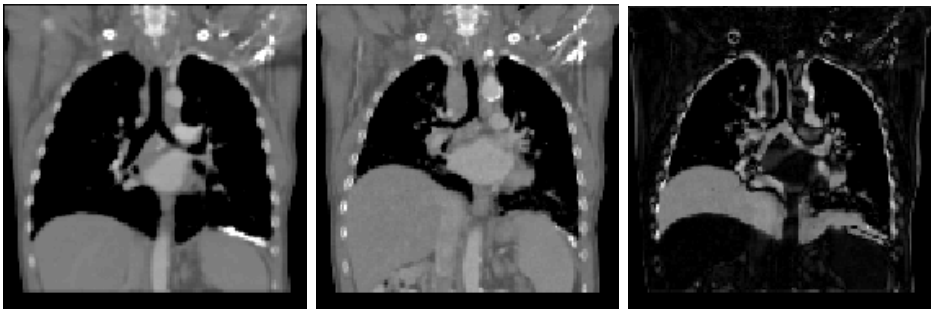
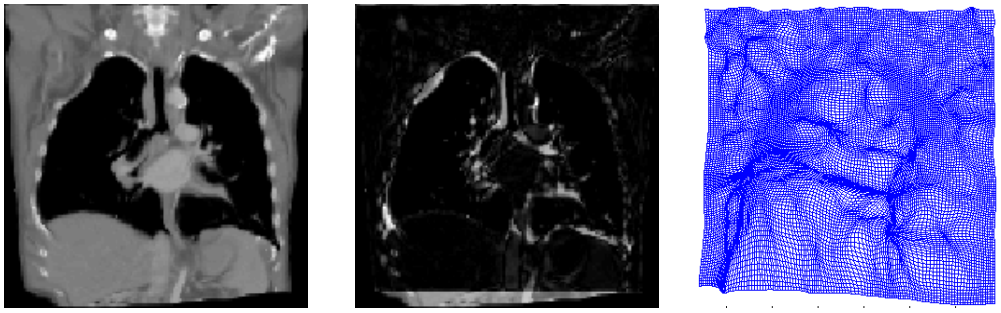
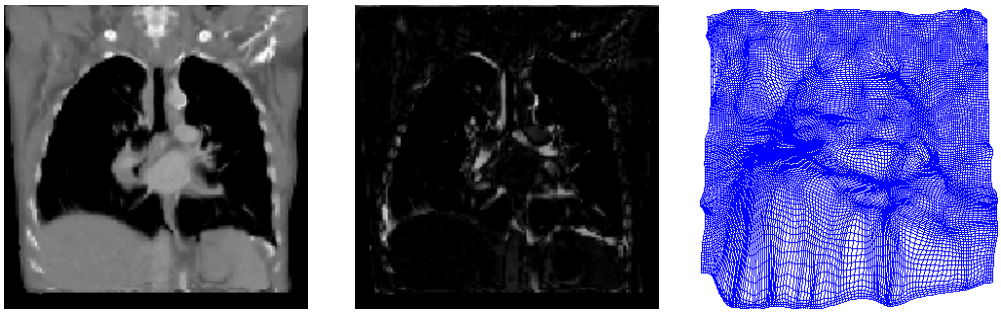


Abbildung 5.6: Medizinische Bilder einer Lunge. Links das Referenzbild bei maximaler Einatmung, in der Mitte das Templatebild bei maximaler Ausatmung, rechts die Differenz vor der Registrierung. Bilder entnommen aus [3].



(a) additiv



(b) diffeomorph

Abbildung 5.7: Ergebnisse der Registrierungen mit $\sigma^2 = 0,9$. Links die deformierten Templatebilder, in der Mitte die Differenz zwischen Referenz- und Templatebild nach der Registrierung, rechts das Deformationsfeld Originalbilder aus [3].

Dämonenalgorithmus im Vergleich zum additiven stärker minimiert wurde. Dies wird auch beim Vergleich der SSD-Reduktion (siehe Tabelle 5.1) deutlich, die beim diffeomorphen Algorithmus wesentlich höher ist. Beim additiven Algorithmus wurde der SSD-Wert um ungefähr 56% reduziert, wohingegen er beim diffeomorphen Algorithmus um ungefähr 99% gesunken ist. Vergleicht man die

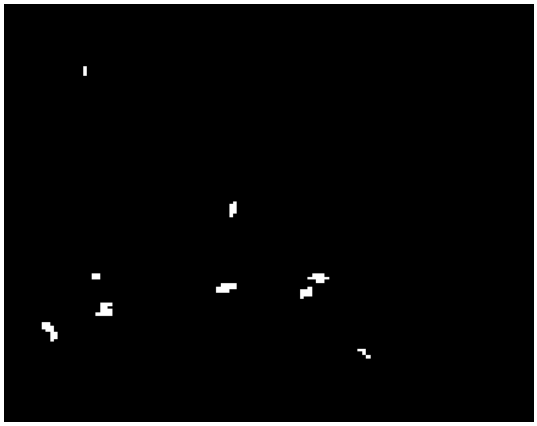


Abbildung 5.8: Binäre Darstellung des Flächeninhalts des Deformationsfelds des additiven Algorithmus bei $\sigma^2 = 0.9$. Weiße Punkte stehen für Zellen mit negativem Flächeninhalt, dort treten also Faltungen auf.

Laufzeiten, so bemerkt man, dass der additive Algorithmus bei gleichen Parametern schneller terminiert.

Beim Vergleich der Registrierungen mit verschiedenem Glättungsparameter (Abbildung 5.5) fällt auf, dass in diesem Fall bei $\sigma^2 = 1$ die größte SSD-Reduktion erreicht wird. Bei $\sigma^2 = 0.4$ ist der SSD-Wert ebenfalls im Bereich der Konvergenz. Die Registrierung mit $\sigma^2 = 3$ ähnelt die SSD-Reduktion sowie Aussehen des Deformationsfelds und deformiertem Templatebild eher der Registrierung mit dem additiven Algorithmus. Hier wurde keine Konvergenz nach dem ersten Konvergenzkriterium erreicht.

Bei der Registrierung der medizinischen Bilder (Abbildung 5.7) sind die Registrierungen mit dem additiven und dem diffeomorphen Algorithmus visuell annähernd gleich gut. Betrachtet man jedoch den SSD-Wert, so wird ersichtlich, dass auch hier die diffeomorphe Registrierung zu einem kleineren SSD-Wert führt. Zudem weist das Deformationsfeld des additiven Algorithmus, wie in Abbildung 5.8 gezeigt, Falten auf; es hat 152 Dreiecke mit negativem Flächeninhalt. Das Deformationsfeld des diffeomorphen Algorithmus hingegen ist faltenfrei; es hat kein einziges Dreieck mit negativem Flächeninhalt. Damit ist das Ergebnis dieses Algorithmus plausibler als das des additiven, da Falten in der Realität nicht sinnvoll interpretiert werden können [30].

5.7 Fazit

Insgesamt wurde in dieser Arbeit gezeigt, dass der Dämonenalgorithmus in der Lage ist, synthetische sowie klinische Bilddaten zu registrieren. Die Dämonenkräfte, die die Bildpunkte verschieben, werden dabei durch den optischen Fluss berechnet. Dieser wurde dann entweder additiv oder kompositiv, sodass eine diffeomorphe Transformation entsteht, mit dem bestehenden Deformationsfeld ver-

knüpft. Es hat sich gezeigt, dass der diffeomorphe Algorithmus bei gleichem Glättungsparameter bessere Ergebnisse bezüglich des SSD-Maßes als der additive Algorithmus liefert. Auch kann er ein faltenfreies Deformationsfeld garantieren, was die Plausibilität der Transformation erhöht.

Im nächsten Schritt könnte der Dämonenalgorithmus auch für dreidimensionale Bilder implementiert werden, um auch klinisch relevante Daten zu registrieren.

Literaturverzeichnis

[1] J. Ashburner. A fast diffeomorphic image registration algorithm. *NeuroImage 38*, 2007.

[2] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations*. Number Bd. 147 in Applied mathematical sciences. Springer, 2001.

[3] E. Castillo, R. Castillo, J. Martinez, M. Shenoy, and T. Guerrero. Four-dimensional deformable image registration using trajectory modeling. *Physics in Medicine and Biology*, 55(1):305, 2010.

[4] C. de Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York, 2001.

[5] D. Forsberg. *Robust Image Registration for Improved Clinical Efficiency*. PhD thesis, Linköping University, Department of Biomedical Engineering, 2013.

[6] O. Forster. *Analysis 1*. Vieweg, 2006.

[7] O. Forster. *Analysis 3*. Vieweg + Teubner, 2009.

[8] K. Fritzsche. *Grundkurs Analysis 2*. Elsevier GmbH, Spektrum Akademischer Verlag, München, 2006.

[9] G. M. Gramlich. *Lineare Algebra: Eine Einführung*. Pro BUSINESS, 2013.

[10] Alexandre Guimond, Alexis Roche, Nicholas Ayache, and Jean Meunier. Three-dimensional multimodal brain warping using the demons algorithm and adaptive intensity corrections. *Medical Imaging, IEEE Transactions on*, 20(1):58–69, 2001.

[11] E. Haber and J. Modersitzki. A multilevel method for image registration. *SIAM Journal on Scientific Computing*, 27(5):1594–1607, 2006.

[12] E. Haber and J. Modersitzki. Image registration with guaranteed displacement regularity. *International Journal of Computer Vision*, 71(3):361–372, 2007.

[13] J. Hadamard. *Lectures on the Cauchy Problem in Linear Partial Differential Equations*. Yale University Press, New Haven, 1923.

[14] D. Halliday, R. Resnick, and J. Walker. *Halliday Physik Bd. 1*. Wiley-VCH, 2009.

[15] M. Hanke-Bourgeois. *Grundlagen der Numerischen Mathematik und des*

Wissenschaftlichen Rechnens. Springer-Verlag, 2011.

[16] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence 17*, pages 185–203, 1981.

[17] B. Jähne. *Digital Image Processing*. Number 1 in Digital Image Processing. Springer, 2005.

[18] J. B. Maintz and M. A. Viergever. A survey of medical image registration. *Medical image analysis*, 2(1):1–36, 1998.

[19] J. Modersitzki. *Flexible Algorithms for Image Registration*. SIAM, 2009.

[20] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. 2006.

[21] L. Ruthotto, F. Gigengack, M. Burger, C. H. Wolters, X. Jiang, K. Schäfers, and J. Modersitzki. A simplified pipeline for motion correction in dual gated cardiac PET. In *Bildverarbeitung für die Medizin 2012*. Springer, 2012.

[22] Oliver Schmitt. *Die multimodale Architektonik des menschlichen Gehirns*. PhD thesis, 2001.

[23] H.-R. Schwarz and N. Köckler. *Numerische Mathematik*. Vieweg+Teubner, Wiesbaden, 2009.

[24] L.G. Shapiro and G.C. Stockman. *Computer Vision*. Prentice Hall, 2001.

[25] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *International Journal of Advanced Robotic Systems*, 4(1):13–16, 2007.

[26] M. Srinivasan. How insects infer range from visual motion. *Reviews of oculomotor research*, 5:139, 1993.

[27] R. Szeliski. Image alignment and stitching: A tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, 2006.

[28] J.-P. Thirion. Image matching as a diffusion process: an analogy with maxwell's demons. *Medical Image Analysis 2*, (3):243–260, 1998.

[29] P. Thévenaz, T. Blu, and M. Unser. Image interpolation and resampling. *Handbook of Medical Imaging, Processing and Analysis*, pages 393–420, 2000.

[30] T. Vercauteren. *Image Registration and Mosaicing for Dynamic In Vivo Fibered Confocal Microscopy*. PhD thesis, École des Mines de Paris, 2008.

[31] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache. Diffeomorphic demons: efficient non-parametric image registration. *NeuroImage*, 45(1):61–72, 2009.

[32] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Non-parametric diffeomorphic image registration with the demons algo-

rithm. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2007*, pages 319–326. Springer, 2007.

[33] G. Wahba. *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990.

[34] B. Zitova and J. Flusser. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.