Institut für Visualisierung und Interaktive Systeme
Universität Stuttgart
Universitätsstraße 38
D - 70569 Stuttgart

Bachelorarbeit Nr. 228

Nicht-lokale Glattheitsterme zweiter Ordnung zur Berechnung des optischen Flusses mit Variationsansätzen

Simon Rühle

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. Andrés Bruhn
Betreuer/in:	Prof. Dr.-Ing. Andrés Bruhn
Beginn am:	01. Juni 2015
Beendet am:	01. Dezember 2015
CR-Nummer:	G.1.6, G.1.8, I.2.10, I.4.8

ABSTRACT

In dieser Arbeit wird eines der grundlegenden Probleme des Bildverstehens, der optische Fluss, behandelt. Der optische Fluss gibt die Verschiebung eines Pixels von einem Bild zum nächsten Bild an, was einer Bewegung der Kamera oder von Gegenständen in der Realität entspricht. Der optische Fluss hat eine große Bedeutung für alle Anwendungsbereiche, in denen die Vorhersage oder Erkennung von Bewegungen wichtig ist, beispielsweise bei Fahrerassistenzsystemen im Auto. Zu Beginn wird eine Einführung in die Bildverarbeitung gegeben, sowie einer der klassischen Algorithmen zur Berechnung des optischen Flusses eingeführt, der Algorithmus von Horn und Schunck. Dieser wird im Anschluss mittels bilateraler, nicht-lokaler Komponenten erweitert, um dessen Schwächen zu beseitigen. Im Anschluss wird der neue Ansatz mit dem von Horn und Schunck auf Basis einiger Bildsequenzen evaluiert.

INHALTSVERZEICHNIS

1. EINLEITUNG

1.1 EINLEITUNG

Die Verwendung von Kameras, nicht nur zur Aufnahme von Bildern und Videos, sondern auch als Sensoren, nimmt stetig an Verbreitung zu. Dieser Trend zieht sich nicht nur durch wissenschaftliche Gebiete, sondern auch durch industrielle Anwendungen, sowie Anwendungen aus dem Automotive Bereich. Auch im Bereich der Endverbraucherprodukte nimmt die Verbreitung zu.
Um einige Beispiele zu nennen, sei die Steuerung von Spielen über spezielle Kameras erwähnt [1], ebenso wie die Vermessung in der Automatisierungstechnik und die Kollisionsvorraussage in modernen Automobilen [2], bis hin zum vollständig autonomen Fahren.
Bei den letzteren beiden Anwendungen spielt die möglichst genaue Bestimmung des optischen Flusses eine besondere Rolle. Mittels des optischen Flusses können Bewegungen zwischen Bildern berechnet werden und somit zur Vorhersage der zukünftigen Position von Objekten in der Szene verwendet werden. Für die Berechnung des optischen Flusses gibt es zwei grundlegende Herangehensweisen. Die erste ist die Berechnung der Bewegung einzelner markanter Punkte. Bei der zweiten Herangehensweise wird die Bewegung jedes Punktes im Bild bestimmt, es entsteht ein dichtes Flussfeld („Dense Optical Flow"). Somit liefern Methoden dieser Herangehensweise für jeden Punkt im Bild einen Vektor, der angibt, wo sich dieser Punkt im nächsten Bild befindet. Da eine möglichst genaue Bestimmung des Flusses in jedem Punkt die größte Flexibilität und Genauigkeit bietet und da somit auch für kleine und eventuell nicht markante Objekte der Fluss berechnet wird, behandelt diese Arbeit die Bestimmung von dichten Flussfeldern.

1.2 ÜBERBLICK ÜBER DAS THEMENGEBIET

Eine wegweisende Arbeit auf dem Gebiet der Berechnung des optischen Flusses ist die Arbeit von Horn und Schunck [3]. In dieser wird das Problem als Energieminimierung, basierend auf einigen Annahmen, aufgefasst. Die Energie besteht dabei aus einem Daten- und einem Glattheitsterm, die mittels einer Energieminimierung mit globalen, kontinuierlichen Optimierungsverfahren, den sogenannten Variationsansätzen, durchgeführt wird. Auf Basis dieser Herangehensweise wurden verschiedene andere Algorithmen vorgeschlagen, die meist auf eine Verbesserung des Datenterms (z.B. [4], [5]) oder Glattheitsterms (z.B. [6], [7]) abzielen.
Häufige Aspekte sind dabei die Berücksichtigung von Kanten im Bild, die auf Objekt-

grenzen und somit auf eine Änderung des Flusses hindeuten können. Weitere wichtige Aspekte sind das Erreichen einer größeren Robustheit gegen Störungen sowie eine größere Glättung in homogenen Gebieten. Eine Möglichkeit dafür ist die Verwendung nicht lokaler Glattheitsterme, wobei in [8] eines der ersten Verfahren für nicht-lokale Variationsrechnung zu finden ist. Durch die Nicht-Lokalität können lokale Störungen leichter kompensiert werden.
Eine weitere Möglichkeit zur Verbesserung ist eine Erhöhung des Grades der Glattheitsforderung und die Verwendung von Glättungstermen, die ähnlich einem bilateralen Filter sind [9]. Diese glätten relativ zu einer durch die Umgebung bestimmten Ebene, wodurch lineare Übergänge im Flussfeld im Vergleich zu einer absoluten Glättung, besser erhalten werden.
Bilaterale Filter [10] wurden hierbei als Ansatz für einen Glattheitsterm verwendet, da diese sowohl räumliche Informationen, also die Entfernung zum Mittelpunkt, als auch die farblichen Informationen in die Glättung miteinbeziehen und somit Kanten besser erhalten.
Zur Berechnung der minimalen Energie werden oft Variationsansätze [11] verwendet, es sind jedoch auch andere Verfahren möglich, wie in [12] oder [13] beschrieben.

1.3 ZIEL DER ARBEIT

In dieser Arbeit soll das Verfahren von Horn und Schunck [3] mittels eines Glattheitsterms höherer Ordnung und einer nicht-lokalen Umgebung zur Glättung erweitert werden, ähnlich zu [9]. Des Weiteren soll die Robustheit der einzelnen Terme durch geeignete Funktionen erhöht werden. Eine ausführliche Evaluierung mittels Testsequenzen, unter anderem aus dem Middleburry [14] Benchmark, soll zeigen, ob und wie viel der neue Term die Genauigkeit der Bestimmung des optischen Flusses in verschiedenen Szenarien verbessert.

1.4 AUFBAU DER ARBEIT

Im zweiten Kapitel werden einige Grundlagen und Konventionen zum optischen Fluss und zur Bildverarbeitung im Allgemeinen erläutert. Kapitel drei erläutert das Verfahren von Horn und Schunck, das danach in Kapitel vier durch einen nicht-lokalen Glattheitsterm zweiter Ordnung verbessert werden soll. Kapitel vier gibt auch einige Hinweise zur Implementierung des neuen Verfahrens. Das fünfte Kapitel behandelt Experimente mit dem neuen Verfahren und vergleicht dieses mit dem Ursprungsverfahren von Horn und Schunck. Das letzte Kapitel fasst die Erkenntnisse und Ergebnisse kurz zusammen und gibt einen Ausblick auf mögliche Verbesserungen.

2. GRUNDLAGEN

Das folgende Kapitel behandelt einige Grundlagen der Bildverarbeitung im Allgemeinen, sowie einige Grundlagen zum Optischen Fluss. Des Weiteren werden einige Konventionen und Schreibweisen erläutert.

2.1 BILD

Ein Bild ist eine Funktion f, die über einer zumeist zwei-dimensionalen Domäne Ω definiert ist [15], formal

$$f : \mathbf{x} \mapsto \mathbb{R}^n, \mathbf{x} \in \Omega \subset \mathbb{R}^m \text{ mit } m, n \in \mathbb{N}.$$

Hierbei ist meist $n = 2$, das Bild also zweidimensional. Der Definitionsbereich Ω ist typischerweise von rechteckiger Form, d.h. $\Omega = (0, \mathtt{Breite}) \times (0, \mathtt{Höhe})$.
Ist $m = 1$ so handelt es sich um ein Grauwertbild, bei $m = 3$ um ein Farbbild. Dennoch sind auch höhere Dimensionen möglich. $\mathbb{R}^n$ wird dabei als Wertebereich bezeichnet.
In dieser Arbeit wird, soweit nicht anders erwähnt, von $n = 2$ und $m = 1$ ausgegangen, also von einem zweidimensionalen Grauwertbild.
Das Argument $\mathbf{x}$ der Funktion ist hierbei von der Form $\mathbf{x} = (x_1, ..., x_m)^\top$, wobei im Zweidimensionalen x_1 die horizontale Koordinate (von links nach rechts) bezeichnet und x_2 die vertikale (von oben nach unten). Die Ableitung der Funktion f an der Stelle $\mathbf{x}$ nach x_j wird abkürzend als f_{x_j} bezeichnet.

2.2 SAMPLING UND QUANTISIERUNG

Da Bilder nicht kontinuierlich vorliegen, muss die Domäne Ω diskretisiert werden [15]. Die Funktion f ist also nur an bestimmten Punkten gegeben, meist Pixel genannt. Wird das Bild an diesen Punkten abgetastet spricht man von Sampling.
Der Abstand zweier Pixel wird oft normalisiert, unter Umständen ist dies jedoch nicht möglich oder sinnvoll. In diesen Fällen muss der Abstand der Pixel mitbetrachtet werden. Der Abstand zwischen zwei Pixeln wird hierbei mit h_{x_i} bezeichnet und kann je nach Anwendung in jeder Dimension verschieden sein. In dieser Arbeit wird zur Vereinfachung von $h_{x_1} = h_{x_2} = 1$ ausgegangen, werden andere h_{x_i} benötigt, so muss dies bei der Diskretisierung berücksichtigt werden, das Vorgehen ist jedoch analog.
Da der Wertebereich in der Praxis ebenfalls nicht kontinuierlich ist, wird dieser ebenfalls diskretisiert, was Quantisierung genannt wird. Bei einem Grauwertbild ist dieser Wertebereich üblicherweise $[0, 255]$ und kann somit in einem Byte gespeichert werden. Alternativ werden diese Werte für Berechnungen oft auf den Bereich $[0, 1]$ normalisiert.

2.3 OPTISCHER FLUSS

Der optische Fluss zwischen zwei Bildern gibt idealerweise die Verschiebung jedes Pixels im ersten Bild zu seiner Position im zweiten oder allgemein nächsten Bild an [16]. Der Fluss, also die Verschiebung der Pixel, in x- beziehungsweise x_1-Richtung wird als u bezeichnet, die Verschiebung in y- beziehungsweise x_2-Richtung als v. Das gesuchte Verschiebungsvektorfeld, das den optischen Fluss beschreibt, ist somit durch $(u(x_1, x_2), v(x_1, x_2))^\top$, oder kurz $(u(\mathbf{x}), v(\mathbf{x}))^\top$ gegeben. Ein Beispiel hierfür wird in Abbildung 2.2 gezeigt, in der der Fluss zwischen den zwei Bildern in 2.1 als Pfeile visualisiert wird.

Abbildung 2.1: Zwei aufeinanderfolgende Bilder aus dem KITTI Benchmark [17][18][19]

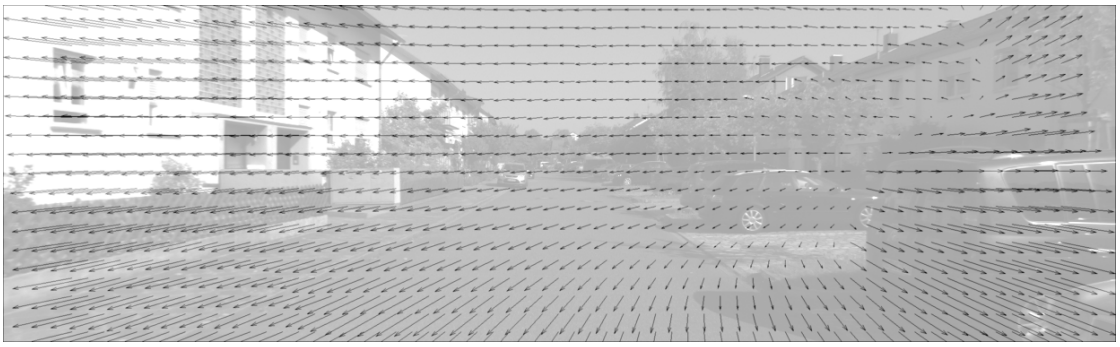


Abbildung 2.2: Berechneter optischer Fluss [20]. Visualisierung durch Pfeile

2.4 PARTIELLE ABLEITUNG UND GRADIENT

Wird eine Funktion mehrerer Variablen $f(x_1, ..., x_n)$ nach einem ihrer Argumente x_i abgeleitet, so entspricht dies der Ableitung der Funktion, als ob x_i nur eine Variable sei und alle anderen x_j konstant seien [11]. Diese sogenannte partielle Ableitung von f nach x_i wird als $\frac{\partial f}{\partial x_i}$ oder als f_{x_i} bezeichnet.
Der Vektor aller möglichen partiellen Ableitungen wird als Gradient bezeichnet

$$\nabla f(x_1, ..., x_n) = (\frac{\partial f}{\partial x_1} f, ..., \frac{\partial f}{\partial x_n} f)^\top. \tag{2.1}$$

Der Betrag des Gradienten ∇f ist hierbei gegeben durch

$$|\nabla f| = \sqrt{f_{x_1}^2 + ... + f_{x_n}^2}. \tag{2.2}$$

Um in einem realen und somit diskreten Bild eine partielle Ableitung berechnen zu können, wird eine Diskretisierung benötigt. Eine mögliche Diskretisierung ist gegeben durch

$$f_{x_1 i,j} \approx \frac{f_{i+1,j} - f_{i-1,j}}{2h_{x_1}} \tag{2.3}$$

$$f_{x_2 i,j} \approx \frac{f_{i,j+1} - f_{i,j-1}}{2h_{x_2}}. \tag{2.4}$$

Diese Art der Diskretisierung wird als „zentrale Differenz“ bezeichnet. Alternativ gibt es noch die „Vorwärtsdifferenz“

$$f_{x_1 i,j} \approx \frac{f_{i+1,j} - f_{i,j}}{h_{x_1}} \tag{2.5}$$

$$f_{x_2 i,j} \approx \frac{f_{i,j+1} - f_{i,j}}{h_{x_2}} \tag{2.6}$$

und die „Rückwärtsdifferenz“

$$f_{x_1 i,j} \approx \frac{f_{i,j} - f_{i-1,j}}{h_{x_1}} \tag{2.7}$$

$$f_{x_2 i,j} \approx \frac{f_{i,j} - f_{i,j-1}}{h_{x_2}}. \tag{2.8}$$

Dies sind die üblichen Diskretisierungen, wobei theoretisch weitere möglich wären, unter Einbeziehung einer größeren Umgebung. Die hier genannten Diskretisierungen werden zusammenfassend auch als „finite Differenzen“ bezeichnet.

2.5 DIVERGENZ

Die Divergenz einer Funktion von n Variablen ist nach [11] definiert als

$$div(f) = f_{x_1} + ... + f_{x_n}. \tag{2.9}$$

2.6 LAPLACE

Der Laplace-Operator [11] ist definiert als

$$\Delta f = div(\nabla f) = f_{x_1 x_1} + ... + f_{x_n x_n}. \tag{2.10}$$

2.7 TAYLORAPPROXIMATION

Die Taylorapproximation, welche auf der Taylorreihe aufbaut, nähert eine Funktion in der Nähe eines bekannten Punktes an [11]. Die Taylorreihe repräsentiert eine Funktion als unendliche Summe von Werten und Ableitungen an *einer* Stelle der Funktion und ist definiert als

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n, \tag{2.11}$$

wobei a die Stelle, die zur Berechnung verwendet wird, auch Entwicklungspunkt genannt, ist und $n! = n \cdot (n-1) \cdot ... \cdot 2 \cdot 1$ die Fakultät darstellt.
Die Taylorapproximation nähert mittels der Taylorreihe eine Funktion in einem Punkt, in dem diese nicht bekannt ist, an, indem dafür eine bekannte Stelle und deren Ableitungen verwendet werden. Da in der Praxis nur endlich viele Ableitungen der Funktion im Punkt a zur Verfügung stehen, kann dieses Verfahren den Wert $f(x)$ nur annähern. Somit erhält man für die Taylorapproximation

$$f(x) \approx \sum_{n=0}^{m} \frac{f^{(n)}(a)}{n!}(x-a)^n, \tag{2.12}$$

mit $m \in \mathbb{N}$. Ist die Funktion f hinreichend glatt und werden die Funktionswerte von f nur in einer kleinen Umgebung von a benötigt, so genügt meist eine Linearisierung der Funktion, d.h. eine Taylorapproximation mit $m = 1$.
Im zweidimensionalen Fall ist die Taylorreihe definiert als [15]

$$f(\mathbf{x}) = \sum_{n=0}^{\infty} \frac{(a_1 \partial_{x_1} + a_2 \partial_{x_2})^n \, f(\mathbf{x})}{n!}. \tag{2.13}$$

Somit ist die zweidimensionale Taylorapproximation gegeben durch:

$$f(\mathbf{x}) \approx \sum_{n=0}^{m} \frac{(a_1 \partial_{x_1} + a_2 \partial_{x_2})^n \, f(\mathbf{x})}{n!}. \tag{2.14}$$

3.1 OPTISCHER FLUSS NACH HORN UND SCHUNCK

Das in dieser Arbeit entwickelte Verfahren berechnet ein dichtes optisches Flussfeld und setzt dafür auf dem Algorithmus zur Bestimmung des optischen Flusses von Horn und Schunck auf [3], weswegen dieser nun ausführlich dargestellt wird. Der Algorithmus basiert hierbei auf zwei Annahmen:

1. Die Konstanzannahme, im Englischen als „constancy assumption“ bezeichnet, beschreibt die Annahme, dass sich die Helligkeit/Farbe in einem Punkt über die Zeit nicht ändert. Hierbei bezeichnet die Funktion $f(\mathbf{x}, t)$ mit $\mathbf{x} = (x_1, x_2)^\top \in \Omega$ die Helligkeit im Punkt $\mathbf{x}$. Der Bildbereich wird mit Ω bezeichnet. Aus dieser Annahme folgt

   $$f(\mathbf{x} + (u, v)^\top, t + 1) - f(\mathbf{x}, t) = 0, \tag{3.1}$$

   wobei u die Verschiebung in x_1-Richtung und v die Verschiebung in x_2-Richtung bezeichnet. Durch Linearsierung mittels einer Taylor-Reihe erhält man:

   $$f(\mathbf{x}, t) = f(\mathbf{x}, t) + f_{x_1}(\mathbf{x}, t)u + f_{x_2}(\mathbf{x}, t)v + f_t, \tag{3.2}$$

   und somit nach Subtraktion von $f(x, t)$:

   $$f_{x_1}(\mathbf{x}, t)u + f_{x_2}(\mathbf{x}, t)v + f_t = 0. \tag{3.3}$$

   Dabei ist anzumerken, dass hierbei angenommen wird, dass f hinreichend glatt ist und der Fluss $(u, v)^\top$ hinreichend klein, so, dass die Linearisierung eine gute Approximation an f darstellt.

2. Die Glattheitsannahme, im Englischen als „smoothness constraint“ bezeichnet, geht davon aus, dass innerhalb des Bildes räumlich nahe Pixel einen ähnlichen Fluss aufweisen. Dies bedeutet, dass das Feld des Flusses bis zu einem gewissen Grad glatt ist, sich also nicht jedes Pixel eines Bildes unabhängig von allen anderen umgebenden Pixeln an einen beliebigen Punkt bewegt, sondern benachbarte Pixel den gleichen Fluss besitzen. Dies macht Sinn, da benachbarte Pixel oft zum selben Objekt gehören. Dadurch gilt:

   $$|\nabla u|^2 = 0 \text{ und } |\nabla v|^2 = 0.$$

   In der Praxis wird davon ausgegangen, dass $|\nabla u|^2$ und $|\nabla v|^2$ klein sind.

Zur Berechnung des optischen Flusses benutzt der Algorithmus von Horn und Schunck das Prinzip der Energieminimierung. Das Energiefunktional sieht hierbei aus wie folgt:

$$E(u,v) = \int_{x\in\Omega} \underbrace{(f_{x_1}(\mathbf{x},t)u + f_{x_2}(\mathbf{x},t)v + f_t)^2}_{\text{Datenterm}} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{\text{Glattheitsterm}} \, d\mathbf{x}. \tag{3.4}$$

Das Energiefunktional setzt sich somit direkt aus den zwei vorherigen Annahmen zusammen und bestraft Abweichungen von diesen. Der vordere Term wird hierbei „Datenterm“ und der hintere „Glattheitsterm“ genannt. Dies kommt daher, dass der erste Term sich auf die Daten im Bild, also die Helligkeit, stützt, der hintere Term sich hingegen nur auf die Annahme stützt, dass der Fluss möglichst konstant ist, was zu einer Glättung führt. Die Gewichtung des Glattheitsterms wird durch den Parameter α gesteuert. Je größer der Wert für α, desto glatter ist das berechnete Verschiebungsvektorfeld.

3.2 VARIATIONSANSÄTZE

Am Ende des vorherigen Abschnittes wurde ein Funktional gegeben, das minimiert werden soll. Das Gebiet der Mathematik das sich mit der Minimierung von Funktionalen beschäftigt, ist die Variationsrechnung [11].
Das Problem aus dem vorherigen Abschnitt ist von der Form

$$E(u,v) = \int_{\texttt{textbfx}\in\Omega} F(\mathbf{x},u,v,u',v') \, d\mathbf{x}. \tag{3.5}$$

Zur Lösung von Problemen dieser Form können die folgenden Euler-Lagrange-Gleichungen verwendet werden:

$$0 = F_u - \frac{\partial}{\partial_{x_1}} F_{u_{x_1}} - \frac{\partial}{\partial_{x_2}} F_{u_{x_2}}, \tag{3.6}$$

$$0 = F_v - \frac{\partial}{\partial_{x_1}} F_{v_{x_1}} - \frac{\partial}{\partial_{x_2}} F_{v_{x_2}}. \tag{3.7}$$

Diese stellen notwendige Bedingungen für jeden Minimierer des allgemeinen Funktionals in (3.5) dar. Zudem gelten folgende Randbedingungen:

$$\begin{pmatrix} F_{u_{x_1}} \\ F_{u_{x_2}} \end{pmatrix}^\top \mathbf{n} = 0, \quad \begin{pmatrix} F_{v_{x_1}} \\ F_{v_{x_2}} \end{pmatrix}^\top \mathbf{n} = 0, \tag{3.8}$$

wobei $\mathbf{n}$ der Vektor ist, der über den Bildbereich am Rand hinaus zeigt.
Somit müssen zuerst die Ableitungen der Funktion F bestimmt werden. Im Fall von Horn und Schunk ist F:

$$F(\mathbf{x},u,v,u',v') = (f_{x_1}(\mathbf{x},t)u + f_{x_2}(\mathbf{x},t)v + f_t)^2 + \alpha(|\nabla u|^2 + |\nabla v|^2) \tag{3.9}$$

Somit werden für die Euler-Langrange-Gleichungen folgende Ableitungen benötigt:

$$F_u = 2(f_{x_1}u + f_{x_2}v + f_t)f_{x_1}\ , \tag{3.10}$$
$$F_v = 2(f_{x_1}u + f_{x_2}v + f_t)f_{x_2}\ , \tag{3.11}$$
$$F_{u_{x_1}} = 2\alpha u_{x_1}\ , \tag{3.12}$$
$$F_{u_{x_2}} = 2\alpha u_{x_2}\ , \tag{3.13}$$
$$F_{v_{x_1}} = 2\alpha v_{x_1}\ , \tag{3.14}$$
$$F_{v_{x_2}} = 2\alpha v_{x_2}\ . \tag{3.15}$$

Eingesetzt in die Euler-Lagrange-Gleichungen und nach Kürzung des Faktors 2 erhält man das folgende gekoppelte System von partiellen Differenzialgleichungen:

$$0 = f_{x_1}(f_{x_1}u + f_{x_2}v + f_t) - \alpha\Delta u, \tag{3.16}$$
$$0 = f_{x_2}(f_{x_1}u + f_{x_2}v + f_t) - \alpha\Delta v, \tag{3.17}$$

mit reflektierenden Neumann-Randbedingungen $\nabla u^\top \mathbf{n} = 0$ und $\nabla v^\top \mathbf{n} = 0$.

3.3 DISKRETISIERUNG

Da Bilder nicht kontinuierlich sind, sondern nur in diskreter Form vorliegen, müssen die Euler-Lagrange Gleichungen diskretisiert werden. Hierbei gilt $i \in [1, ..., N]$:

$$0 = f_{x_1 i}(f_{x_1 i}u_i + f_{x_2 i}v_i + f_{ti}) - \alpha \sum_{j \in N_2(i)} (u_j - u_i), \tag{3.18}$$
$$0 = f_{x_2 i}(f_{x_1 i}u_i + f_{x_2 i}v_i + f_{ti}) - \alpha \sum_{j \in N_2(i)} (v_j - v_i). \tag{3.19}$$

Der Laplace-Operator Δ für u und v wird hierbei mittels finiten Differenzen approximiert und $N_2(i)$ bezeichnet die direkt an das Pixel i angrenzenden Pixel (links, rechts, oben, unten).
Ziel ist es die u_i und v_i zu bestimmen. Da die Gleichungen ein großes, dünnbesetztes und lineares Gleichungssystem ergeben, kann dieses mittels eines iterativen Verfahrens gelöst werden. Zudem kann gezeigt werden, dass die Systemmatrix im Allgemeinen positiv definit ist [21].

3.4 LÖSER

Im Folgenden wird dafür das Gauß-Seidel-Verfahren verwendet, allerdings sind auch andere Verfahren, wie das Jacobi-Verfahren oder SOR mit geringen Änderungen möglich (alle beschrieben in [11]).
Im Folgenden werden drei Verfahren zu Lösung des Gleichungssystems verwendet,

das Jacobi-, Gauss-Seidel- und SOR-Verfahren [11]. Das Gleichungssystem des Problems hat die allgemeine Form $Ax = b$ oder ausführlicher (für ein Bild der Größe 3×2)

$$\underbrace{\left(\begin{pmatrix} f_{x_1}^2 & & & & & & \\ & f_{x_1}^2 & & & & & \\ & & f_{x_1}^2 & & & & \\ & & & f_{x_1}^2 & & & \\ & & & & f_{x_1}^2 & & \\ & & & & & f_{x_1}^2 & \\ C & & & & & & f_{x_1}^2 \end{pmatrix} \begin{pmatrix} C & & & & & & \\ & C & & & & & \\ & & C & & & & \\ & & & C & & & \\ & & & & C & & \\ & & & & & C & \\ & & & & & & C \end{pmatrix} \begin{pmatrix} f_{x_2}^2 & & & & & & \\ & f_{x_2}^2 & & & & & \\ & & f_{x_2}^2 & & & & \\ & & & f_{x_2}^2 & & & \\ & & & & f_{x_2}^2 & & \\ & & & & & f_{x_2}^2 & \\ & & & & & & f_{x_2}^2 \end{pmatrix} - \alpha \begin{pmatrix} -2 & 1 & & 1 & & & & & & & & \\ 1 & -3 & 1 & & 1 & & & & & & & \\ & 1 & -2 & & & 1 & & & & & & \\ 1 & & & -2 & 1 & & & & & & & \\ & 1 & & 1 & -3 & 1 & & & & & & \\ & & 1 & & 1 & -2 & & & & & & \\ & & & & & & -2 & 1 & & 1 & & \\ & & & & & & 1 & -3 & 1 & & 1 & \\ & & & & & & & 1 & -2 & & & 1 \\ & & & & & & 1 & & & -2 & 1 & \\ & & & & & & & 1 & & 1 & -3 & 1 \\ & & & & & & & & 1 & & 1 & -2 \end{pmatrix} \right)}_{A} \underbrace{\begin{pmatrix} u \\ u \\ u \\ u \\ u \\ u \\ v \\ v \\ v \\ v \\ v \\ v \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} D \\ D \\ D \\ D \\ D \\ D \\ E \\ E \\ E \\ E \\ E \\ E \end{pmatrix}}_{b},$$

(3.20)

wobei C, D und E als $C = f_{x_1} f_{x_2}$, $D = -f_{x_1} f_t$ und $E = -f_{x_2} f_t$ definiert sind.
Nach [22] kann eine Näherung der Lösung dieses Systems gefunden werden durch die Zerlegung

$$A = A_1 + A_2 \hookrightarrow (A_1 + A_2)\mathbf{x} = \mathbf{b} \Leftrightarrow A_1\mathbf{x} = \mathbf{b} - A_2\mathbf{x}. \tag{3.21}$$

Somit ist eine Fixpunktiteration der Form

$$\mathbf{x}^{k+1} = A_1^{-1}(\mathbf{b} - A_2\mathbf{x}^k) \tag{3.22}$$

möglich, wobei A_1^{-1} idealerweise einfach zu berechnen ist und A_1 gleichzeitig eine möglichst gute Approximation an A darstellen soll.

3.4.1 *Jacobi-Verfahren*

Beim Jacobi-Verfahren [11] wird die Matrix A in den Diagonalteil D, die obere Dreiecksmatrix U und die untere Dreiecksmatrix L aufgeteilt, d.h.

$$A = D - L - U. \tag{3.23}$$

Dies führt zur allgemeinen Form für das Jacobi-Verfahren von

$$\mathbf{x}^{k+1} = D^{-1}(\mathbf{b} + (L + U)\mathbf{x}^k). \tag{3.24}$$

Somit erhält man für die Methode von Horn und Schunck

$$u_i^{k+1} = \frac{-f_{x_1 i}(f_{x_2 i} v_i^k + f_{ti}) + \alpha \sum_{j \in N_{(i)}} u_j^k}{f_{x_1 i}^2 + \alpha |N(i)|}, \tag{3.25}$$

$$v_i^{k+1} = \frac{-f_{x_2 i}(f_{x_1 i} u_i^k + f_{ti}) + \alpha \sum_{j \in N_{(i)}} v_j^k}{f_{x_2 i}^2 + \alpha |N(i)|}. \tag{3.26}$$

Hierbei bezeichnet $N(i)$ die Nachbarschaft der direkt an i angrenzenden Pixel und $|N(i)|$ die Anzahl der direkt angrenzenden Nachbarn von i, maximal sind dies in diesem Fall vier (rechts, links, oben und unten). Es ist zu beachten, dass zur Berechnung die Werte der vorherigen Iteration u^k,v^k verwendet werden und somit bei einer Implementierung gesondert gespeichert werden müssen.

3.4.2 *Gauß-Seidel Verfahren*

Die Gauß-Seidel-Methode [11] nähert mit A_1 die Matrix A besser an und konvergiert somit schneller als die Jacobi-Methode. A_1 ist in diesem Fall $A_1 = D - L$, wodurch $A_2 = -U$. Dies führt zur allgemeinen Form

$$\mathbf{x}^{k+1} = (D - L)^{-1}(\mathbf{b} + U\mathbf{x}^k). \tag{3.27}$$

Somit erhält man für die Horn und Schunck Methode folgende Fixpunktiteration

$$u_i^{k+1} = \frac{-f_{x_1 i}(f_{x_2 i} v_i^k + f_{ti}) + \alpha \sum_{j \in N_-(i)} u_j^{k+1} - \alpha \sum_{j \in N_+(i)} u_j^k}{f_{x_1 i}^2 + \alpha |N(i)|}, \tag{3.28}$$

$$v_i^{k+1} = \frac{-f_{x_2 i}(f_{x_1 i} u_i^{k+1} + f_{ti}) + \alpha \sum_{j \in N_-(i)} v_j^{k+1} - \alpha \sum_{j \in N_+(i)} v_j^k}{f_{x_2 i}^2 + \alpha |N(i)|}. \tag{3.29}$$

$N_-(i)$ bezeichnet alle Pixel der Umgebung, die in der Iteration über das Bild vor dem Pixel i liegen und $N_+(i)$ alle die nach i liegen. Somit werden für jedes Pixel nacheinander die u_i und v_i berechnet und an den Stellen, an denen hier ein u_j beziehungsweise v_j vorkommt wird der aktuellste Wert verwendet, also entweder der Wert aus der vorherigen Iteration, initial null, oder der Wert aus der aktuellen Iteration.

3.4.3 *SOR-Verfahren*

Das SOR-Verfahren („successive over relaxation“) [11], im Deutschen auch Relaxationsverfahren genannt, kann die Konvergenzeigenschaften des Gauß-Seidel Verfahrens

verbessern. Hierbei wird statt in jedem Schritt einen komplett neuen Wert zu bestimmen der vorherige Wert mittels eines neu berechneten Werts korrigiert:

$$u_i^{k+1} = (1-w)u_i^k \tag{3.30}$$

$$+ w \cdot \frac{-f_{x_1 i}(f_{x_2 i} v_i^k + f_{ti}) + \alpha \sum_{j \in N_-(i)} u_j^{k+1} - \alpha \sum_{j \in N_+(i)} u_j^k}{f_{x_1 i}^2 + \alpha |N(i)|}, \tag{3.31}$$

$$v_i^{k+1} = (1-w)v_i^{k+1} \tag{3.32}$$

$$+ w \cdot \frac{-f_{x_2 i}(f_{x_1 i} u_i^{k+1} + f_{ti}) + \alpha \sum_{j \in N_-(i)} v_j^{k+1} - \alpha \sum_{j \in N_+(i)} v_j^k}{f_{x_2 i}^2 + \alpha |N(i)|}. \tag{3.33}$$

Für den Relaxationsparameter w muss hierbei gelten, dass $0 < w < 2$, wobei für $w = 1$ das Verfahren identisch mit dem Gauß-Seidel Verfahren ist. Vom SOR-Verfahren wird erst für $w > 1$ gesprochen (Überrelaxation).

3.5 BILATERAL FILTER

Nach der Vorstellung des Horn und Schunck Verfahrens werden nun die Grundlagen, die für den neuen Glättungsterm nötig sind, vorgestellt, die bilateralen Filter. Bilaterale Filter [10] dienen zur Glättung, beziehungsweise zur Entfernung von Störungen aus Bildern. Lineare Glättungsfilter bilden zur Glättung einen gewichteten Durchschnitt der Pixel um den zu glättenden Pixel. Dabei berücksichtigen die Gewichte der einzelnen Pixel sowohl den räumlichen Abstand als auch den Farbunterschied zum zentralen Pixel. Ein einfacher bilateraler Filter zur Glättung von Bildern mit f_i, dem Eingabepixel an der Stelle i und u_i, dem gefilterten Signal an dieser Stelle ist durch

$$h_i = \frac{\sum_j g(|f_i - f_j|^2) w(|\mathbf{x}_i - \mathbf{x}_j|^2) f_j}{\sum_j g(|f_i - f_j|^2) w(|\mathbf{x}_i - \mathbf{x}_j|^2)} \tag{3.34}$$

gegeben. Hierbei stellt $g(|f_i - f_j|^2)$ ein farbliches Gewicht dar, das mit größer werdender Differenz von f_i und f_j kleiner wird und $w(|\mathbf{x}_i - \mathbf{x}_j|^2)$ stellt ein räumliches Gewicht dar, das ebenfalls mit größer werdender Differenz kleiner wird. Die Summenvariable j kann hierbei entweder auf eine Region um das zentrale Pixel begrenzt werden, oder es wird die räumliche Gewichtungsfunktion in einer Art gewählt, so dass weiter entfernte Pixel ein Gewicht nahe null haben. Der Nenner stellt dabei eine Normalisierung der Gewichte dar, sodass die Summe aller Gewichte eins ergibt.
Eine Möglichkeit für die Gewichtungsfunktionen stellt die Gaussfunktion[11] dar, wobei dabei die Standardabweichungen σ die Stärke des Einflusses der Differenz darstellt. Die Gaussfunktion, auch Normalverteilung genannt, ist dabei definiert als

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{3.35}$$

beziehungsweise im zwei-dimensionalen Fall als

$$g(\mathbf{x}) = \frac{1}{2\pi\sigma^2} e^{-\frac{{x_1}^2+{x_2}^2}{2\sigma^2}}. \tag{3.36}$$

Die Abnahme der Gewichte $g(|f_i - f_j|^2)$ mit größer werdender Differenz führt zu einer besseren Erhaltung von Kanten, da Pixel mit anderem Grauwert bei der Glättung kaum Einfluss haben.
Wird als Funktion die Gaussfunktion verwendet, so gibt es zwei Parameter für den bilateralen Filter, die Standardabweichung σ der Funktion $g(x)$, im Folgenden als σ_g bezeichnet, und die Standardabweichung σ der Funktion $w(x)$, mit σ_w bezeichnet.
Die Auswirkung des Parameter σ_g wird in Abbildung 3.1 dargestellt. Oben links ist das Originalbild. Auf die anderen Bilder wurde ein bilateraler Filter mit $\sigma_w = 15$ und von oben Mitte nach unten rechts mit den σ_g von 2.5, 25, 35, 100 und 1000 angewendet. Für $\sigma_g \to \infty$ werden die Pixel der Nachbarschaft unabhängig von der Grauwertdifferenz gemittelt, somit entspricht dies einem Gaussfilter.
Abbildung 3.2 stellt die Auswirkung des Parameter σ_w dar. Oben links ist wieder das Originalbild. Die anderen Bilder wurden mittels eines bilateralen Filters mit $\sigma_g = 35$ und von oben Mitte nach unten rechts mit den σ_w von 1, 2.5, 4, 10 und 50 gefiltert. Bei $\sigma_w \to \infty$ erweitert sich die Nachbarschaft auf das ganze Bild, sodass global ähnliche Pixel gemittelt werden. Eine räumliche Gewichtung findet so gut wie nicht mehr statt.

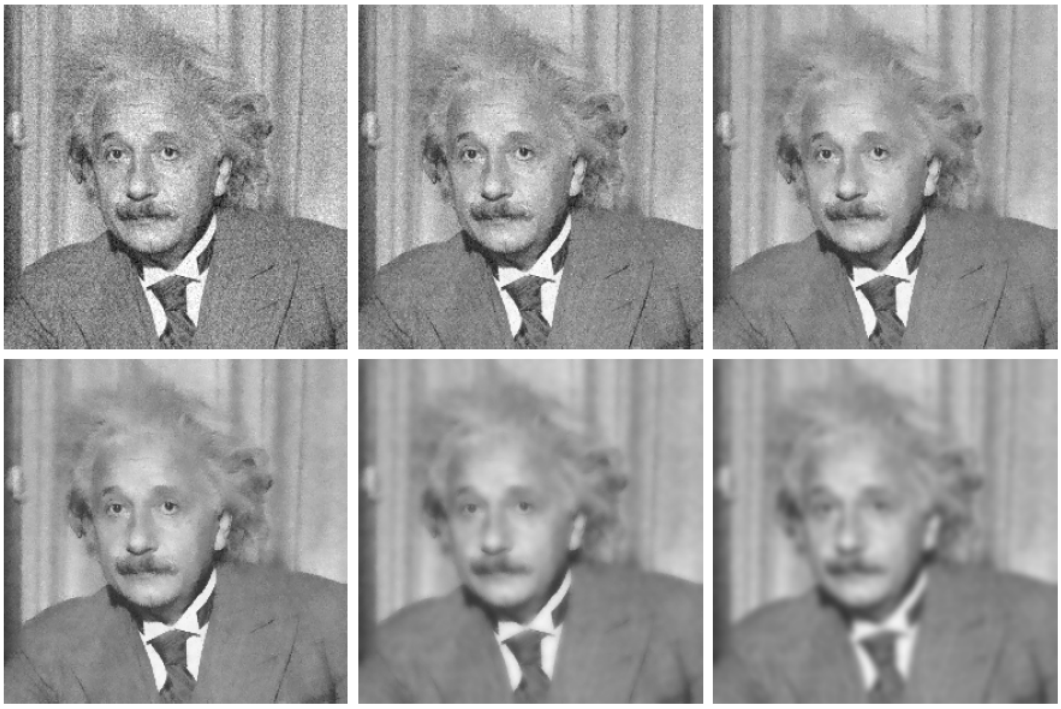


Abbildung 3.1: Wirkung des Parameters σ_g aus [15]

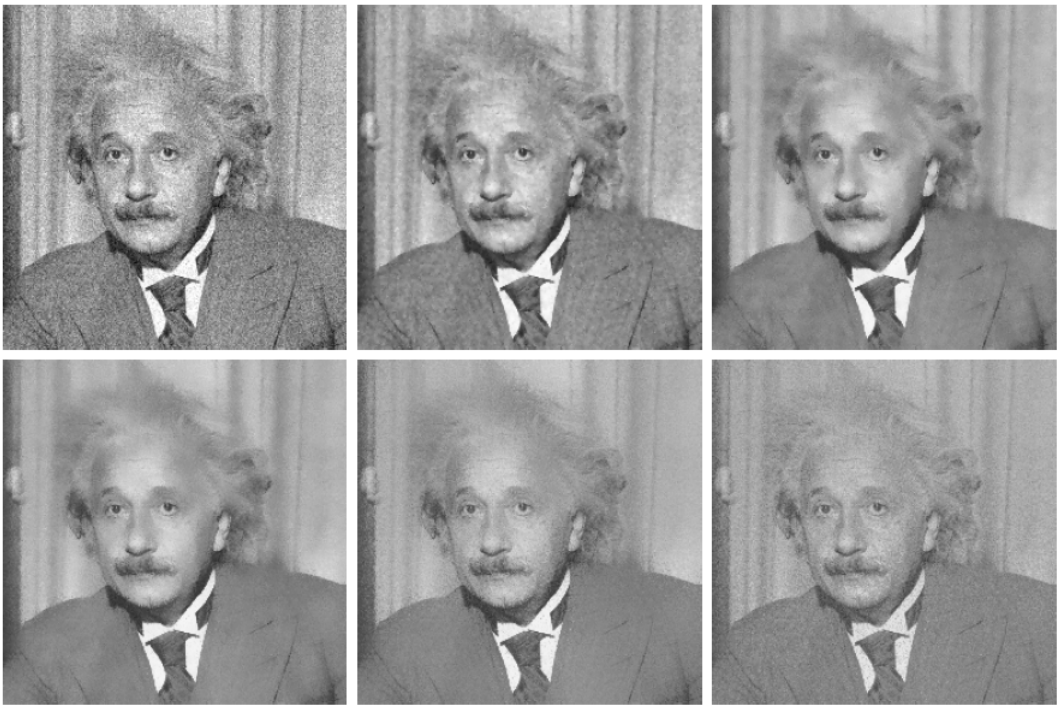


Abbildung 3.2: Wirkung des Parameters σ_w aus [15]

Verwendet man statt der direkten Differenz von f_i und f_j die abgewandelte Form $f_j - (f_i + \beta_i^1(\mathbf{j}_1 - \mathbf{i}_1) + \beta_i^2(\mathbf{j}_2 - \mathbf{i}_2))$, so wird eine Ebene durch die Funktionswerte f_i gelegt und somit werden zum Glätten statt der direkten Eingangswerte f_j, die um die Ebene korrigierten Werte verwendet. Dies ermöglicht auch in Bereichen mit linearen Verläufen der Funktionswerte eine möglichst genaue und passende Glättung des Eingangssignals [23] Auch hier ist eine Normalisierung notwendig. Dabei werden β^1 und β^2 zunächst aus dem Anfangssignal bestimmt und dann iterativ an die bearbeiteten Signale angepasst.

4.1 THEORETISCHE HERANGEHENSWEISE

4.1.1 *Quadratische Energiefunktion*

Das neue Verfahren basiert auf dem Verfahren von Horn und Schunck [3], soll jedoch nicht einen einfachen Glattheitsterm verwenden, sondern einen Glattheitsterm höherer Ordnung, der Kanten im Fluss besser erhält und gleichzeitig eine größere Umgebung in die Berechnung einbeziehen soll. Dazu wird die Annahme getroffen, dass Kanten in Fluss mit Kanten im Bild korrelieren. In einem ersten Schritt wird hierbei das Verfahren von Horn und Schunck um einen bilateralen Glattheitsterm sowie einen Term zur Glättung der eingeführten β^i ergänzt:

$$\begin{aligned} E(u,v) = &\int_{\mathbf{x}\in\Omega} (f_x u + f_y v + f_z)^2 \, d\mathbf{x} \\ &+ \alpha \int_{\mathbf{x}\in\Omega} \delta(\mathbf{x}) \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \left(\Phi(\mathbf{x},\hat{\mathbf{x}})(g_1(\mathbf{x},\hat{\mathbf{x}})^2 + g_2(\mathbf{x},\hat{\mathbf{x}})^2)\right) \, d\hat{\mathbf{x}} \, d\mathbf{x} \\ &+ \gamma \int_{\mathbf{x}\in\Omega} \left(|\nabla\beta^1|^2 + |\nabla\beta^2|^2 + |\nabla\beta^3|^2 + |\nabla\beta^4|^2\right) \, d\mathbf{x}. \end{aligned} \tag{4.1}$$

$N(\mathbf{x})$ eine zu wählende Nachbarschaft um $\mathbf{x}$.
Die Funktionen $g_1(\mathbf{x},\hat{\mathbf{x}})$ und $g_2(\mathbf{x},\hat{\mathbf{x}})$ sind definiert als

$$g_1(\mathbf{x},\hat{\mathbf{x}}) = u(\hat{\mathbf{x}}) - (u(\mathbf{x}) + \beta^1(\mathbf{x})(\hat{x}_1 - x_1) + \beta^2(\mathbf{x})(\hat{x}_2 - x_2)), \tag{4.2}$$

$$g_2(\mathbf{x},\hat{\mathbf{x}}) = v(\hat{\mathbf{x}}) - (v(\mathbf{x}) + \beta^3(\mathbf{x})(\hat{x}_1 - x_1) + \beta^4(\mathbf{x})(\hat{x}_2 - x_2)). \tag{4.3}$$

Die β_i sollen hierbei die Werte von u und v im Verhältnis zu einer Ebene durch die Umgebung anpassen, wodurch der Term $u(\mathbf{x}) + \beta^1(\mathbf{x})(\hat{x}_1 - x_1) + \beta^2(\mathbf{x})(\hat{x}_2 - x_2)$ beziehungsweise $v(\mathbf{x}) + \beta^3(\mathbf{x})(\hat{x}_1 - x_1) + \beta^4(\mathbf{x})(\hat{x}_2 - x_2)$ den Wert u beziehungsweise v in $\mathbf{x}$ in Relation zu dieser Ebene setzt.

Die Funktion $\Phi(\mathbf{x},\hat{\mathbf{x}})$ dient zur räumlichen Gewichtung der Nachbarschaft $N(\mathbf{x})$ im Glattheitsterm, wobei diese symetrisch zu wählen ist, also

$$\Phi(\mathbf{x},\hat{\mathbf{x}}) = \Phi(\hat{\mathbf{x}},\mathbf{x}) \tag{4.4}$$

gelten muss. Des Weiteren muss $\Phi(\mathbf{x},\hat{\mathbf{x}})$ immer größer oder gleich null sein und monoton fallen. Für diese Arbeit wird

$$\Phi(\mathbf{x},\hat{\mathbf{x}}) = \frac{1}{\sqrt{1 + \frac{(\hat{x}_1 - x_1)^2 + (\hat{x}_2 - x_2))^2}{\lambda^2}}} \tag{4.5}$$

verwendet.
Die Normalisierungsfunktion δ ist definiert als:

$$\delta(\mathbf{x}) = \frac{1}{\int_{\hat{\mathbf{x}} \in N(\mathbf{x})} \Phi(\mathbf{x}, \hat{\mathbf{x}})\, d\hat{\mathbf{x}}}.$$

Die Normalisierungsfunktion δ ermöglicht es, die Größe der Nachbarschaft $N(\mathbf{x})$ anzupassen, ohne das Gewicht α ändern zu müssen. Hierbei normalisiert die Funktion die räumlichen Gewichte der einzelnen Elemente der Nachbarschaft und macht α unabhängig von diesen und der Größe der Nachbarschaft.
Der letzte Term der Funktion sorgt für eine Glättung der β-Werte, da sonst die β-Werte in jedem Punkt eine beliebige Ebene darstellen könnten und somit keine Glättung von u und v mehr stattfindet. Der Term zwingt Ebenen in einem Umkreis ähnlich zu sein, die Ableitung der β^i soll also in jedem Punkt klein sein.

4.1.2 *Subquadratische Energiefunktion*

Zur Erhöhung der Robustheit werden für alle Terme zusätzliche Ψ-Funktionen eingeführt wodurch, man folgende Energiefunktion erhält:

$$\begin{aligned} E(u, v) = &\int_{\mathbf{x} \in \Omega} \Psi_1((f_x u + f_y v + f_z)^2)\, d\mathbf{x} \\ &+ \alpha \int_{\mathbf{x} \in \Omega} \delta(\mathbf{x}) \int_{\hat{\mathbf{x}} \in N(\mathbf{x})} \left(\Phi(\mathbf{x}, \hat{\mathbf{x}}) \Psi_2(g_1(\mathbf{x}, \hat{\mathbf{x}})^2 + g_2(\mathbf{x}, \hat{\mathbf{x}})^2)\right) d\hat{\mathbf{x}}\, d\mathbf{x} \\ &+ \gamma \int_{\mathbf{x} \in \Omega} \Psi_3 \left(|\nabla \beta^1|^2 + |\nabla \beta^2|^2 + |\nabla \beta^3|^2 + |\nabla \beta^4|^2\right) d\mathbf{x}. \end{aligned} \tag{4.6}$$

Die Funktionen $\Psi_1(s^2)$, $\Psi_2(s^2)$ und $\Psi_3(s^2)$ sind subquadratische Funktionen, um die Robustheit des jeweiligen Terms zu erhöhen, und müssen immer echt größer null sein und dabei monoton steigende, konvexe Funktionen sein, um eine eindeutige Minimierung zu gewährleisten.
In der Implementierung zu dieser Arbeit wird für die drei Ψ-Funktionen die Funktion

$$\Psi(s^2) = 2\lambda^2 \sqrt{1 + \frac{s^2}{\lambda^2}} - 2\lambda^2 \tag{4.7}$$

mit der zugehörigen Ableitung

$$\Psi'(s^2) = \frac{1}{\sqrt{1 + \frac{s^2}{\lambda^2}}} \tag{4.8}$$

verwendet.

Aus Gründen der Darstellung werden folgende abkürzenden Schreibweisen für die Ableitungen der Ψ Funktionen verwendet:

$$\Psi_1'(\mathbf{x}) = \Psi_1'((f_x u + f_y v + f_z)^2), \tag{4.9}$$

$$\Psi_2'(\mathbf{x}, \hat{\mathbf{x}}) = \Psi_2'(g_1(\mathbf{x}, \hat{\mathbf{x}})^2 + g_2(\mathbf{x}, \hat{\mathbf{x}})^2), \tag{4.10}$$

$$\Psi_2'(\hat{\mathbf{x}}, \mathbf{x}) = \Psi_2'(g_1(\hat{\mathbf{x}}, \mathbf{x})^2 + g_2(\hat{\mathbf{x}}, \mathbf{x})^2)., \tag{4.11}$$

$$\Psi_3'(\mathbf{x}) = \Psi_3'(|\nabla\beta^1|^2 + |\nabla\beta^2|^2 + |\nabla\beta^3|^2 + |\nabla\beta^4|^2). \tag{4.12}$$

Im weiteren Verlauf wird auch folgende verkürzende Schreibweise verwendet:

$$w(\mathbf{x}, \hat{\mathbf{x}}) = \Phi(\mathbf{x}, \hat{\mathbf{x}})\Psi_2'(\mathbf{x}, \hat{\mathbf{x}}). \tag{4.13}$$

4.1.3 *Variationsrechnung*

Das Energiefunktional $E(u, v)$ wird im Folgenden mittels Variationsrechnung minimiert. Hier werden die Euler-Lagrange-Gleichungen zu Hilfe genommen. Dafür werden die partiellen Ableitungen der zu integrierenden Funktion nach u,v, den β^i und $\beta^i_{x_1}$ sowie $\beta^i_{x_2}$ benötigt. Da die Energie minimiert werden soll, können gemeinsame konstante Faktoren aller Terme vernachlässigt werden.

$$\begin{aligned}
F_u &= 2\,\Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_1} \\
&\quad + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x}, \hat{\mathbf{x}})\Psi_2'(\mathbf{x}, \hat{\mathbf{x}})2g_1(\mathbf{x}, \hat{\mathbf{x}})g_{1_u}(\mathbf{x}, \hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\
&\quad + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\hat{\mathbf{x}}, \mathbf{x})\Psi_2'(\hat{\mathbf{x}}, \mathbf{x})2g_1(\hat{\mathbf{x}}, \mathbf{x})g_{1_u}(\hat{\mathbf{x}}, \mathbf{x})\,d\hat{\mathbf{x}} \\
&= 2\,\Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_1} \\
&\quad + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} (w(\mathbf{x}, \hat{\mathbf{x}}) + w(\hat{\mathbf{x}}, \mathbf{x}))(u(\mathbf{x}) - u(\hat{\mathbf{x}}))\,d\hat{\mathbf{x}} \\
&\quad + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \big((w(\mathbf{x}, \hat{\mathbf{x}})\beta^1(\mathbf{x}) + w(\hat{\mathbf{x}}, \mathbf{x})\beta^1(\hat{\mathbf{x}}))(\hat{x}_1 - x_1) \\
&\quad + (w(\mathbf{x}, \hat{\mathbf{x}})\beta^2(\mathbf{x}) + w(\hat{\mathbf{x}}, \mathbf{x})\beta^2(\hat{\mathbf{x}}))(\hat{x}_2 - x_2)\big)\,d\hat{\mathbf{x}}
\end{aligned} \tag{4.14}$$

$$
\begin{aligned}
F_v &= 2\,\Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_2} \\
&\quad + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x},\hat{\mathbf{x}})\Psi_2'(\mathbf{x},\hat{\mathbf{x}})2g_2(\mathbf{x},\hat{\mathbf{x}})g_{2_v}(\mathbf{x},\hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\
&\quad + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\hat{\mathbf{x}},\mathbf{x})\Psi_2'(\hat{\mathbf{x}},\mathbf{x})2g_2(\hat{\mathbf{x}},\mathbf{x})g_{2_v}(\hat{\mathbf{x}},\mathbf{x})\,d\hat{\mathbf{x}} \\
&= 2\,\Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_2} \\
&\quad + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} (w(\mathbf{x},\hat{\mathbf{x}}) + w(\hat{\mathbf{x}},\mathbf{x}))(v(\mathbf{x}) - v(\hat{\mathbf{x}}))\,d\hat{\mathbf{x}} \\
&\quad + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} [(w(\mathbf{x},\hat{\mathbf{x}})\beta^3(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^3(\hat{\mathbf{x}}))(\hat{x}_1 - x_1) \\
&\quad + (w(\mathbf{x},\hat{\mathbf{x}})\beta^4(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^4(\hat{\mathbf{x}}))(\hat{x}_2 - x_2)]\,d\hat{\mathbf{x}}
\end{aligned} \tag{4.15}
$$

Hierbei sind g_u und g_v die Ableitungen von g nach u und v, mit $g_u(\mathbf{x},\hat{\mathbf{x}}) = -1$, $g_u(\hat{\mathbf{x}},\mathbf{x}) = 1$, $g_v(\mathbf{x},\hat{\mathbf{x}}) = -1$ und $g_v(\hat{\mathbf{x}},\mathbf{x}) = 1$.
Das letzte Integral in F_u und F_v bezieht mit ein, dass ein Pixel nicht nur als Zentrum seiner eigenen Umgebung berücksichtigt werden muss, sondern auch alle Umgebungen, in denen es liegt, einen Einfluss haben. Dies sind genau die Umgebungen der Pixel, die in der Umgebung des zentralen Pixels liegen.
Darüber hinaus sind die Ableitungen nach β^1 und β^2 gegeben durch

$$
\begin{aligned}
F_{\beta^1} &= \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x},\hat{\mathbf{x}})\Psi_2'(\mathbf{x},\hat{\mathbf{x}})2g_1(\mathbf{x},\hat{\mathbf{x}})g_{1_{\beta^1}}(\mathbf{x},\hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\
&= 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^1(\mathbf{x})(\hat{x}_1 - x_1)^2\,d\hat{\mathbf{x}} \\
&\quad - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)(u(\hat{\mathbf{x}}) - u(\mathbf{x}) - \beta^2(\mathbf{x})(\hat{x}_2 - x_2))\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.16}
$$

$$
\begin{aligned}
F_{\beta^2} &= \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x},\hat{\mathbf{x}})\Psi_2'(\mathbf{x},\hat{\mathbf{x}})2g_1(\mathbf{x},\hat{\mathbf{x}})g_{1_{\beta^2}}(\mathbf{x},\hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\
&= 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^2(\mathbf{x})(\hat{x}_2 - x_2)^2\,d\hat{\mathbf{x}} \\
&\quad - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)(u(\hat{\mathbf{x}}) - u(\mathbf{x}) - \beta^1(\mathbf{x})(\hat{x}_1 - x_1))\,d\hat{\mathbf{x}}.
\end{aligned} \tag{4.17}
$$

Die Ableitungen nach β^3 und β^4 lauten dementsprechend:

$$\begin{aligned} F_{\beta^3} &= \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x},\hat{\mathbf{x}})\Psi_2'(\mathbf{x},\hat{\mathbf{x}})2g_2(\mathbf{x},\hat{\mathbf{x}})g_{2_{\beta^3}}(\mathbf{x},\hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\ &= 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^3(\mathbf{x})(\hat{x}_1 - x_1)^2\,d\hat{\mathbf{x}} \\ &\quad - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)(v(\hat{\mathbf{x}}) - v(\mathbf{x}) - \beta^4(\mathbf{x})(\hat{x}_2 - x_2))\,d\hat{\mathbf{x}}, \end{aligned} \tag{4.18}$$

$$\begin{aligned} F_{\beta^4} &= \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \Phi(\mathbf{x},\hat{\mathbf{x}})\Psi_2'(\mathbf{x},\hat{\mathbf{x}})2g_2(\mathbf{x},\hat{\mathbf{x}})g_{2_{\beta^4}}(\mathbf{x},\hat{\mathbf{x}})\,d\hat{\mathbf{x}} \\ &= 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^4(\mathbf{x})(\hat{x}_2 - x_2)^2\,d\hat{\mathbf{x}} \\ &\quad - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)(v(\hat{\mathbf{x}}) - v(\mathbf{x}) - \beta^3(\mathbf{x})(\hat{x}_1 - x_1))\,d\hat{\mathbf{x}}. \end{aligned} \tag{4.19}$$

Und schließlich sind die Ableitungen nach $\beta^i_{x_1}$ und $\beta^i_{x_2}$ durch

$$\begin{aligned} F_{\beta^i_{x_1}} &= 2\gamma\Psi_3'(\mathbf{x})\beta^i_{x_1}\,, \\ F_{\beta^i_{x_2}} &= 2\gamma\Psi_3'(\mathbf{x})\beta^i_{x_2}\,, \end{aligned} \tag{4.20}$$

gegeben für $i \in \{1,...,4\}$.
Die zuvor berechneten partiellen Ableitungen können nun in die allgemeinen Euler-Lagrange-Gleichungen eingesetzt werden,

$$F_u - \partial_{x_1}F_{u_{x_1}} - \partial_{x_2}F_{u_{x_2}} = 0, \tag{4.21}$$

$$F_v - \partial_{x_1}F_{v_{x_1}} - \partial_{x_2}F_{v_{x_2}} = 0, \tag{4.22}$$

$$\tag{4.23}$$

für u,v, sowie

$$F_{\beta^i} - \partial_{x_1}F_{\beta^i_{x_1}} - \partial_{x_2}F_{\beta^i_{x_2}} = 0, \tag{4.24}$$

$$\tag{4.25}$$

für $\beta^i, i \in 1,...,4$ mit den Randbedingungen $n^\top\nabla\beta^i = 0$.
Dies liefert:

$$
\begin{aligned}
0 = {} & \Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_1} \\
& + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} (w(\mathbf{x},\hat{\mathbf{x}}) + w(\hat{\mathbf{x}},\mathbf{x}))\,(u(\mathbf{x}) - u(\hat{\mathbf{x}}))\,d\hat{\mathbf{x}} \\
& + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \left(w(\mathbf{x},\hat{\mathbf{x}})\beta^1(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^1(\hat{\mathbf{x}})\right)(\hat{x}_1 - x_1) \\
& \quad + \left(w(\mathbf{x},\hat{\mathbf{x}})\beta^2(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^2(\hat{\mathbf{x}})\right)(\hat{x}_2 - x_2)\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.26}
$$

und

$$
\begin{aligned}
0 = {} & \Psi_1'(\mathbf{x})(f_{x_1}u + f_{x_2}v + f_z)f_{x_2} \\
& + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} (w(\mathbf{x},\hat{\mathbf{x}}) + w(\hat{\mathbf{x}},\mathbf{x}))(v(\mathbf{x}) - v(\hat{\mathbf{x}}))\,d\hat{\mathbf{x}} \\
& + \alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} \left(w(\mathbf{x},\hat{\mathbf{x}})\beta^3(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^3(\hat{\mathbf{x}})\right)(\hat{x}_1 - x_1) \\
& \quad + \left(w(\mathbf{x},\hat{\mathbf{x}})\beta^4(\mathbf{x}) + w(\hat{\mathbf{x}},\mathbf{x})\beta^4(\hat{\mathbf{x}})\right)(\hat{x}_2 - x_2)\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.27}
$$

sowie für die β^i:

$$
\begin{aligned}
0 = {} & -\operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^1(\mathbf{x})) \\
& + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^1(\mathbf{x})(\hat{x}_1 - x_1)^2\,d\hat{\mathbf{x}} \\
& - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)\left(u(\hat{\mathbf{x}}) - u(\mathbf{x}) - \beta^2(\mathbf{x})(\hat{x}_2 - x_2)\right)\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.28}
$$

$$
\begin{aligned}
0 = {} & -\operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^2(\mathbf{x})) \\
& + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^2(\mathbf{x})(\hat{x}_2 - x_2)^2\,d\hat{\mathbf{x}} \\
& - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_2 - x_2)\left(u(\hat{\mathbf{x}}) - u(\mathbf{x}) - \beta^1(\mathbf{x})(\hat{x}_1 - x_1)\right)\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.29}
$$

$$
\begin{aligned}
0 = {} & -\operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^3(\mathbf{x})) \\
& + 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^3(\mathbf{x})(\hat{x}_1 - x_1)^2\,d\hat{\mathbf{x}} \\
& - 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_1 - x_1)\left(v(\hat{\mathbf{x}}) - v(\mathbf{x}) - \beta^4(\mathbf{x})(\hat{x}_2 - x_2)\right)\,d\hat{\mathbf{x}},
\end{aligned} \tag{4.30}
$$

$$
\begin{aligned}
0 = &- \operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^4(\mathbf{x})) \\
&+ 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})\beta^4(\mathbf{x})(\hat{x}_1 - x_1)^2 \,\mathrm{d}\hat{\mathbf{x}} \\
&- 2\alpha\delta \int_{\hat{\mathbf{x}}\in N(\mathbf{x})} w(\mathbf{x},\hat{\mathbf{x}})(\hat{x}_2 - x_2)\left(v(\hat{\mathbf{x}}) - v(\mathbf{x}) - \beta^3(\mathbf{x})(\hat{x}_2 - x_2)\right) \mathrm{d}\hat{\mathbf{x}}.
\end{aligned} \tag{4.31}
$$

4.1.4 *Diskretisierung*

Im Folgenden werden die Funktionen für die Berechnung diskretisiert. Zur besseren Lesbarkeit sind i und j Vektoren welche die Position eines Pixels angeben ($i = (i_1, i_2)^\top$ und $j = (j_1, j_2)^\top$). Für u und v erhalten wir somit:

$$
\begin{aligned}
0 =&\Psi_{1_i}'(f_{x_{1i}}u_i + f_{x_{2i}}v_i + f_{z_i})f_{x_{1i}} \\
&+ \alpha\,\delta_i \sum_{j\in N(i)} (w_{i,j} + w_{j,i})(u_i - u_j) \\
&+ \alpha\,\delta_i \sum_{j\in N(i)} \left[(w_{i,j}\beta_i^1 + w_{j,i}\beta_j^1)(j_1 - i_1) + (w_{i,j}\beta_i^2 + w_{j,i}\beta_j^2)(j_2 - i_2)\right]
\end{aligned} \tag{4.32}
$$

$$
\begin{aligned}
0 =&\Psi_{1_i}'(f_{x_{1i}}u_i + f_{x_{2i}}v_i + f_{z_i})f_{x_{2i}} \\
&+ \alpha\,\delta_i \sum_{j\in N(i)} (w_{i,j} + w_{j,i})(v_i - v_j) \\
&+ \alpha\,\delta_i \sum_{j\in N(i)} \left[(w_{i,j}\beta_i^3 + w_{j,i}\beta_j^3)(j_1 - i_1) + (w_{i,j}\beta_i^4 + w_{j,i}\beta_j^4)(j_2 - i_2)\right]
\end{aligned} \tag{4.33}
$$

Bei den Funktionen β^k muss der Term $\operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^k(\mathbf{x}))$ diskretisiert werden:

$$
\begin{aligned}
\operatorname{div}(\Psi_3'(\mathbf{x})\nabla\beta^k(\mathbf{x})) &= \left(\Psi_3'(\mathbf{x})\beta_{x_1}^k\right)_{x_1} + \left(\Psi_3'(\mathbf{x})\beta_{x_2}^k\right)_{x_2} \\
&= \sum_{j\in N_2(i)} \left(\frac{\Psi_{3_i}' + \Psi_{3_j}'}{2}\right)\left(\beta_j^k - \beta_i^k\right),
\end{aligned} \tag{4.34}
$$

wobei $N_2(i)$ die direkt an das Pixel i angrenzenden Pixel bezeichnet (links, rechts, oben, unten). Hierbei wurden die $\beta_{x_j}^k$ ebenso wie die Ableitungen $(\Psi_3'(\mathbf{x})\beta_{x_j}^k)_{x_j}$ mittels finiter Differenzen approximiert.

Bei $\Psi_3(s^2) = s^2 \to \Psi_3'(s^2) = 1$ entspricht dies dem Laplace-Operator.
Somit erhalten wir folgende Gleichungen für $\beta^1, ..., \beta^4$:

$$\begin{aligned} 0 = &- \sum_{j \in N_2(i)} \left(\frac{\Psi'_{3_i} + \Psi'_{3_j}}{2} \right) (\beta_j^1 - \beta_i^1) \\ &+ \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} \beta_i^1 (j_1 - i_1)^2 \\ &- \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)(u_j - u_i - \beta_i^2 (j_2 - i_2)), \end{aligned} \tag{4.35}$$

$$\begin{aligned} 0 = &- \sum_{j \in N_2(i)} \left(\frac{\Psi'_{3_i} + \Psi'_{3_j}}{2} \right) (\beta_j^2 - \beta_i^2) \\ &+ \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} \beta_i^2 (j_2 - i_2)^2 \\ &- \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_2 - i_2)(u_j - u_i - \beta_i^1 (j_1 - i_1)), \end{aligned} \tag{4.36}$$

$$\begin{aligned} 0 = &- \sum_{j \in N_2(i)} \left(\frac{\Psi'_{3_i} + \Psi'_{3_j}}{2} \right) (\beta_j^3 - \beta_i^3) \\ &+ \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} \beta_i^3 (j_1 - i_1)^2 \\ &- \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)(u_j - u_i - \beta_i^4 (j_2 - i_2)), \end{aligned} \tag{4.37}$$

$$\begin{aligned} 0 = &- \sum_{j \in N_2(i)} \left(\frac{\Psi'_{3_i} + \Psi'_{3_j}}{2} \right) (\beta_j^4 - \beta_i^4) \\ &+ \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} \beta_i^4 (j_2 - i_2)^2 \\ &- \alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_2 - i_2)(u_j - u_i - \beta_i^3 (j_1 - i_1)). \end{aligned} \tag{4.38}$$

4.1.5 *Löser*

Um die u_i, $v_i$und die $\beta_{k_i}, k \in \{1, 2, 3, 4\}$ zu berechnen, werden die Gleichungen wie in 3.4 auf Seite 9 nach den zu berechnenden Werten umgeformt. Dabei definiert $N_2(i)$

die Umgebung eines Pixels zur Berechnung des diskreten Laplace-Operators analog zu 3.4. Dies sind also die direkt an ein Pixel angrenzenden Pixel (im zwei-dimensionalen maximal 4). Aus Gründen der Darstellung wird im Gegensatz zu 3.4 hier die Jacobi-Methode verwendet. Die anderen Methoden können analog zu 3.4 berechnet werden. Zur weiteren Vereinfachung der Darstellung wird von $h_{x_1} = h_{x_2} = 1$ ausgegangen.

$$
\begin{aligned}
u_i^{(k+1)} = & \frac{1}{\Psi'_{1_i} f_{x_1 i}^2 + \alpha\, \delta_i \sum_{j \in N(i)} (w_{i,j} + w_{j,i})} \\
& \Big(-\Psi'_{1_i} (f_{x_2 i} v_i^{(k)} + f_{z i}) f_{x_1 i} \\
& \quad + \alpha\, \delta_i \sum_{j \in N(i)} (w_{i,j} + w_{j,i}) u_j^{(k)} \\
& \quad - \alpha\, \delta_i \sum_{j \in N(i)} \left[(w_{i,j} \beta_i^1 + w_{j,i} \beta_j^1)(j_1 - i_1) + (w_{i,j} \beta_i^2 + w_{j,i} \beta_j^2)(j_2 - i_2) \right] \Big)
\end{aligned}
\tag{4.39}
$$

$$
\begin{aligned}
v_i^{(k+1)} = & \frac{1}{\Psi'_{1_i} f_{x_2 i}^2 + \alpha\, \delta_i \sum_{j \in N(i)} (w_{i,j} + w_{j,i})} \\
& \Big(-\Psi'_{1_i} (f_{x_1 i} u_i^{(k)} + f_{z i}) f_{x_2 i} \\
& \quad + \alpha\, \delta_i \sum_{j \in N(i)} (w_{i,j} + w_{j,i}) v_j^{(k)} \\
& \quad - \alpha\, \delta_i \sum_{j \in N(i)} \left[(w_{i,j} \beta_i^3 + w_{j,i} \beta_j^3)(j_1 - i_1) + (w_{i,j} \beta_i^4 + w_{j,i} \beta_j^4)(j_2 - i_2) \right] \Big)
\end{aligned}
\tag{4.40}
$$

$$
\beta_i^{1^{(k+1)}} = \frac{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)(u_j - u_i - \beta_i^{2^{(k)}} (j_2 - i_2)) + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i}) \beta_l^{1^{(k)}}}{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)^2 + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i})}
\tag{4.41}
$$

$$
\beta_i^{2^{(k+1)}} = \frac{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_2 - i_2)(u_j - u_i - \beta_i^{1^{(k)}} (j_1 - i_1)) + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i}) \beta_l^{2^{(k)}}}{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_2 - i_2)^2 + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i})}
\tag{4.42}
$$

$$
\beta_i^{3^{(k+1)}} = \frac{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)(v_j - v_i - \beta_i^{4^{(k)}} (j_2 - i_2)) + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i}) \beta_l^{3^{(k)}}}{\alpha\, \delta_i \sum_{j \in N(i)} w_{i,j} (j_1 - i_1)^2 + \frac{1}{2} \gamma \sum_{l \in N_2(i)} (\Psi'_{3_l} + \Psi'_{3_i})}
$$

(4.43)

$$\beta_i^{4^{(k+1)}} = \frac{\alpha\,\delta_i \sum_{j\in N(i)} w_{i,j}(j_2 - i_2)(v_j - v_i - \beta_i^{3^{(k)}}(j_1 - i_1)) + \frac{1}{2}\gamma \sum_{l\in N_2(i)}(\Psi'_{3_l} + \Psi'_{3_i})\beta_l^{4^{(k)}}}{\alpha\,\delta_i \sum_{j\in N(i)} w_{i,j}(j_2 - i_2)^2 + \frac{1}{2}\gamma \sum_{l\in N_2(i)}(\Psi'_{3_l} + \Psi'_{3_i})} \tag{4.44}$$

Somit werden die u, v und die β_i iterativ berechnet. Hierbei bezeichnet (k) das Ergebnis des vorherigen Rechenschritts und $(k+1)$ das neue Ergebnis. Dabei werden zuerst einige Iterationen die u und v mit konstanten β_i Werten berechnet und anschließend werden die u und v Werte konstant gehalten und die β_i berechnet. Dieses Vorgehen wird hierbei wiederholt ausgeführt. Näheres dazu in 4.2 auf Seite 25.

4.2 IMPLEMENTIERUNG

Im Folgenden werden einige wichtige Aspekte zur effizienten Implementierung behandelt und ein kurzer Einblick in das Vorgehen gegeben. Auch soll dieses Kapitel helfen eine Vorstellung vom Ablauf des Algorithmus und der Berechnung der Werte zu bekommen.

4.2.1 *Ablauf*

Es gibt insgesamt sechs verschiedenen Variablen pro Pixel die bestimmt werden sollen, u, v, β_1, β_2, β_3 und β_4, wobei die β_i letztlich nur zur Berechnung von u und v benötigt werden und somit nicht zum Ergebnis direkt gehören. Alle Variablen hängen dabei von den w_{ij} beziehungsweise w_{ji} ab, welche wieder von allen Variablen abhängen. Um nichtlineare Abhängigkeiten aufzulösen werden die w_{ij} eine gewisse Anzahl von Iterationen konstant gehalten und erst dann wieder aktualisiert [24][25][26]. Siehe dazu Abbildung 4.1 auf Seite 26.

Die Berechnung besteht somit aus mehreren Iterationen, wobei jede Iteration aus zwei Teilen besteht. Zuerst wird die Berechnung von u und v ausgeführt und danach die Berechnung der β^i. Jede der Berechnungen von u, v und den β^i besteht dabei wiederrum aus mehreren Sub-Iterationen, wobei in diesen jeweils eine gewisse Anzahl von Sub-Iterationen lang die w_{ij} konstant gehalten werden. Dies kommt daher, dass in jeder Sub-Iteration ein lineares Gleichunssystem gelöst wird. Dies geschieht entweder zur Berechnung der u, v oder der β^i. Somit wird das ursprünglich nichtlineare Problem durch eine Serie von Berechnungen linearer Gleichungssysteme gelöst.

4.2.2 *Vorausberechnung der w_{ij}*

Für die Berechnung von u, v und den β^i werden die w_{ij} benötigt und diese sind dabei einige Sub-Iterationen konstant. Somit empfiehlt es sich die w_{ij} für alle i zu berechnen und in einem Array zu speichern. Beim Abruf von w_{ij} muss somit im Array an Position i der j-te Wert abgerufen werden. Mit den w_{ji} wird analog verfahren. Der Speicher zur Speicherung der w_{ij} und w_{ji} sollte dabei nicht für jede Iteration neu allokiert werden, sondern einmal zentral allokiert werden und dann für alle Iterationen genutzt werden. Dies vermeidet die Fragmentierung des Heaps und erhöht, durch das Wegfallen unnötiger Allokierungen und Deallokierungen, die Geschwindigkeit.

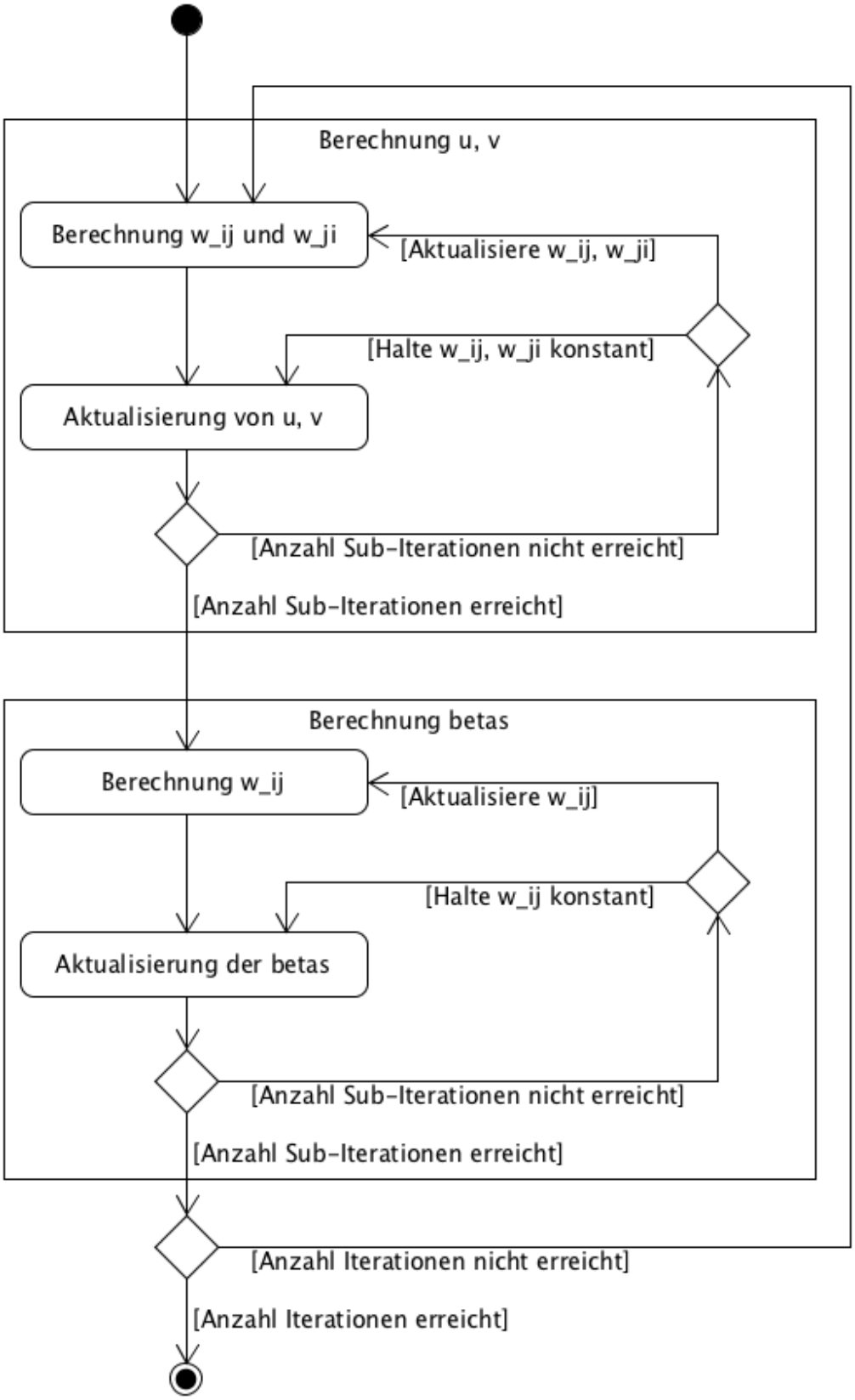


Abbildung 4.1: Ablaufdiagram des Algorithmus

4.2.3 *Praktische Implementierung*

Im Rahmen dieser Arbeit wurde das neue Verfahren implementiert. Als Programmiersprache kam dabei C++ zum Einsatz. Die Vorteile von C++ gegenüber C liegen dabei in der Möglichkeit der modularen Struktur und Objekorientierten Programmierung sowie in der Möglichkeit, mittels Templates einige Operationen zur Compilezeit zu berechnen und dennoch einfach Änderungen vornehmen zu können (z.B. wurde die Größe der Umgebung, $N(i)$, auf diese Weise implementiert).
Für die Berechnung der w_{ij} und w_{ji} wurden Buffer aus einem einmalig allokierten Memory-Pool verwendet, um die Geschwindigkeit und Effizienz zu erhöhen. Alle Teile des Algorithmus wurden nah an der theoretischen Arbeit implementiert. Zur schnelleren Konvergenz der Berechnung wurde das Verfahren mittels SOR implementiert mit einem Relaxationsparameter $w = 1,99$. Dieser Wert wurde experimentell als bester Wert festgestellt. Die implementierte Klasse wurde möglichst frei von Abhängigkeiten gehalten und benutzt nur Standard Libraries.

5.1 FEHLERMASSE

Für die Experimente werden im Folgenden einige Begrifflichkeiten und Maße zur Beurteilung der Genauigkeit der Algorithmen eingeführt. Hierbei bezeichnet „Ground-Truth“ den (annähernd) exakten optischen Fluss, der bei Benchmarks für Trainings- und Evaluierungszwecke gegeben ist. Der berechnete Fluss in x_1- bzw. x_2-Richtung wird im Folgenden als u bzw. v und im der Ground Truth Fluss als u_{GT} bzw. v_{GT} bezeichnet. Die im Folgenden vorgestellten und später für die Experimente verwendeten Fehlermaße sind:

- Winkelfehler [14]
- Absoluter Fehler [14]

5.1.1 *Winkelfehler*

Das erste Kriterium stellte der Winkelfehler dar [14]. Im Englischen wird dieses Fehlermaß als „Angular Error“ bezeichnet und im Folgenden damit als „AE“ abgekürzt. Dieses Fehlermaß berechnet den Winkel im 3D Raum zwischen $(u, v, 1)^\top$ und $(u_{GT}, v_{GT}, 1)^\top$.

$$AE = \cos^{-1}\left(\frac{1 + u \cdot u_{GT} + v \cdot v_{GT}}{\sqrt{1 + u^2 + v^2}\sqrt{1 + u_{GT}^2 + v_{GT}^2}}\right) \tag{5.1}$$

Zur Beschreibung der Abweichung eines Bildes wird das Maß RX verwendet. RX bezeichnet den Prozentsatz an Pixeln die einen Fehler von mehr als von x besitzen. So bedeutet $R2^\circ = 5,00$ beispielsweise, dass 5 Prozent aller Pixel mehr als 2° vom Winkel der Flussvektoren der Ground Truth abweichen.
Des Weiteren wird der durchschnittliche Winkelfehler bestimmt.
Zu beachten ist, dass dieses Fehlermaß durch die 1 in der Zeitkomponente relativ zur Größe der Verschiebung ist. Bei kleineren u und v fallen Fehler weniger stark ins Gewicht als bei großen Werten von u und v.

5.1.2 *Endpunktfehler*

Der absolute Fehler, im Englischen „Absolut Error“ oder „Endpoint Error“ , abgekürzt mit “EE“, gibt die absolute Abweichung von $(u, v)^\top$ zu $(u_{GT}, v_{GT})^\top$ an [14].

$$EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2} \tag{5.2}$$

Analog wie beim Winkelfehler bezeichnet RX die prozentuale Anzahl an Pixeln, deren Fluss um mehr als X vom Ground-Truth-Fluss abweicht.
Der absolute Fehler ist stärker praxisrelevant, da er nicht nur die Winkelungenauigkeit angibt, sondern die tatsächliche Differenz, was z.B. bei der Kollisionsschätzung in der Automobilindustrie extrem wichtig ist.
Als weiteres Maß wird der durchschnittliche absolute Fehler bestimmt.

5.2 BENCHMARKS

Zur Evaluierung des Algorithmus werden Sequenzen aus dem „Middelburry Optical Flow Evaluation“ Benchmark [14], die Sequenzen „Translating Trees“ (TT) und „Diverging Trees“ (TD) aus [27] und „Office“ aus [28] verwendet. Die „Middelburry Optical Flow Evaluation“, im Folgenden abkürzend als „Middelburry Benchmark“ bezeichnet, ist ein Benchmark zum Vergleich verschiedener Algorithmen zur Bestimmung des optischen Flusses. Hierbei kommen sowohl synthetische als auch unter kontrollierten Bedingungen aufgenommene echte Bilder zum Einsatz. Es gehört zu den bekanntesten Benchmarks für Algorithmen zur Bestimmung des optischen Flusses, beschränkt sich allerdings nicht nur auf Sequenzen mit kleinen Bewegungen. Im Folgenden werden die verwendeten Sequenzen abgebildet und die Gründe für die Auswahl der einzelnen Sequenzen ausgeführt.

Bild 1 | Bild 2 | Ground Truth

Tabelle 5.1: Middleburry Sequenz „RubberWhale“

Bild 1 | Bild 2 | Ground Truth

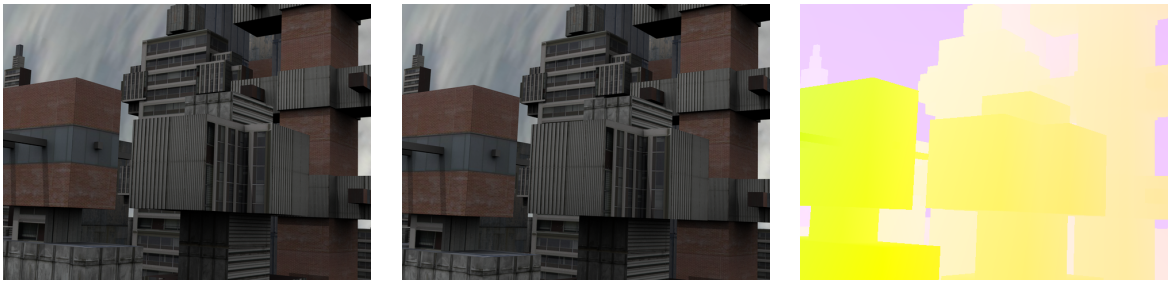


Tabelle 5.2: Middleburry Sequenz „Urban3“

Bild 1 | Bild 2 | Ground Truth

Tabelle 5.3: Sequenz „TT“

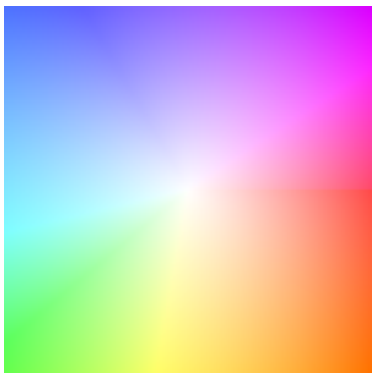
Bild 1	Bild 2	Ground Truth
		

Tabelle 5.4: Sequenz „TD“

Bild 1	Bild 2	Ground Truth
		

Tabelle 5.5: Sequenz „Office“

Die Sequenzen „RubberWhale“ (Tabelle 5.1) und „Urban3“ (Tabelle 5.2) aus dem Middleburry Benchmark sollen den Vergleich der Verfahren im Umgang mit realistischeren Begebenheiten möglich machen. Bei der Sequenz „RubberWhale“ bewegen sich die Gegenstände im Vordergrund hauptsächlich nach links, wobei sich der kreisförmige Gegenstand dabei leicht dreht, was an den verschiedenen Farben in der GroundTruth sichtbar wird. Der orangene Teil des Hintergrunds und der hölzerne, gitterförmige Teil bewegen sich aufeinander zu. In der Sequenz „Urban3“ bewegen sich die Gebäude nach unten und werden dabei ein wenig nach links gedreht. Die Grundbewegungsrichtung aller Gebäude ist dabei ähnlich, es handelt sich somit im Gegensatz zu „RubberWhale“ um eine statische Szene, bei der die Kamera bewegt wird. Beide Sequenzen enthalten Diskontinuitäten im Flussfeld, deutlich erkennbar beispielsweise bei „RubberWhale“ bei den Gegenständen im Vordergrund, oder bei „Urban3“ der Übergang zwischen Gebäuden und Himmel. Des Weiteren treten bei beiden Sequenzen Überdeckungen auf und „Urban3“ besitzt, im Gegensatz zu den anderen Sequenzen, größere Flüsse. Wichtig sind von diesen Eigenschaften vor allem die Diskontinuitäten im Flussfeld, da der neue Ansatz entwickelt wurde, um diese besser zu erhalten als dies beim Algorithmus von Horn und Schunck der Fall ist. Darum handelt es sich bei diesen Sequenzen um die zur Beurteilung essentiellen Sequenzen.
Die Sequenzen „TT“, „TD“ und „Office“ wurden gewählt, da diese einen gleichmäßigen und kleinen Fluss besitzen. Dabei stellen „TD“ und „Office“ einen Fluss durch eine Vergrößerung dar, und „TT“ einen Fluss durch eine einfache Verschiebung des Bildes nach rechts. Es gibt somit keine Verdeckungen, Diskontinuitäten oder sonstige Störungen im Fluss, die das Verfahren negativ beeinflussen könnten. Diese Sequenzen sollen eine Betrachtung der Algorithmen in optimalen Fällen, zumindest für den Algorithmus von Horn und Schunck, darstellen.

5.3 DURCHFÜHRUNG

Zur Bestimmung der Fehlerwerte werden die zu vergleichenden Algorithmen nacheinander auf die Bildsequenzen angewendet und im Anschluss die Übereinstimmung der Ergebnisse mit der Ground Truth ermittelt. Es werden nur Bereiche berücksichtigt, in denen der optische Fluss berechnet werden kann, sofern diese in der Ground Truth oder auf anderem Wege als solche Bereiche gekennzeichnet sind.
Es werden die Fehlermaße und Kriterien aus den vorhergehenden Abschnitten (Winkelfehler und Endpunktfehler) verwendet. Es wurden beide Algorithmen selbst mit dem SOR-Verfahren implementiert und diese Implementierungen für die Evaluation benutzt. Bei beiden Algorithmen wurden zusätzlich die Eingabebilder mittels eines Gauss-Filters vorgeglättet.

Die Parameter beim Horn und Schunck Verfahren sind:

- Standardabweichung σ für Vorglättung: 1,2
- Gewichtung Glättungsterm α: 2000
- Anzahl Iterationen: 2000
- Relaxationsparameter w: 1,95

Die Parameter beim neuen Verfahren sind:

- Standardabweichung σ für Vorglättung: 1,2
- Gewichtung Glättungsterm α: 35
- Gewichtung Glättungsterm der β^i (γ): 2000
- Anzahl Iterationen: 6
- Anzahl Sub-Iterationen: 15
- Anzahl Iterationen mit konstanten $w_{i,j}$: 8
- Relaxationsparameter w: 1,99

Es wurden für alle Sequenzen die gleichen Parameter verwendet, wobei die optimalen Parameter experimentell bestimmt wurden. Die Berechnungszeit für die Sequenz „Urban3“ (640×480 Pixel) betrug beim Verfahren von Horn und Schunck ungefähr $12,7$ Sekunden, beim neuen Ansatz ungefähr $40,4$ Sekunden. Die Zeiten wurden in einer virtuellen Maschine mit einem Einkernprozessor mit 1,8 GHz Grundtakt gemessen.

5.4 ERGEBNISSE

Im Folgenden werden die Ergebnisse der Evaluation dargestellt und diskutiert. Dabei werden anhand der Testsequenzen einige Vor- und Nachteile des neuen Ansatzes aufgezeigt.
Anhand der Testsequenz „RubberWhale“ (Tabelle 5.8) lassen sich die Vorteile des neuen Ansatzes klar erkennen. Die Kanten im Fluss sind klarer und weniger verschwommen und die kleinen Strukturen im Hintergrund (oben links) sind besser erhalten. Dies ist auch am runden Gegenstand in der Bildmitte erkennbar. Hier ist die Form im Fluss im Gegensatz zu Horn und Schunck deutlich zu erkennen inklusive des Lochs im Gegenstand.
An der Testsequenz „Urban3“ (Tabelle 5.9) lässt sich die bessere Erhaltung von Kanten im Fluss ebenfalls gut erkennen. Vor allem sichtbar wird dies im oberen mittleren Bereich des Bildes, in dem man eine deutliche Kante im Fluss erkennen kann. Generell

	Horn und Schunck			Neuer Ansatz		
Fehlermaß	R2,5	R5,0	R10,0	R2,5	R5,0	R10,0
RubberWhale	80,97	54,76	34,74	81,72	46,13	17,58
Urban3	94,64	83,88	69,26	91,06	82,55	67,60
Office	66,16	29,08	10,33	85,15	54,82	18,64
TT	70,52	12,02	5,96	68,74	21,92	6,46
TD	46,23	27,68	15,50	70,36	29,38	7,06

Tabelle 5.6: Winkelfehler des Horn und Schunck Algorithmus und des neuen Ansatzes im Vergleich.

	Horn und Schunck				Neuer Ansatz			
Fehlermaß	R0,5	R1,0	R2,0	Average	R0,5	R1,0	R2,0	Average
RubberWhale	26,73	13,04	4,30	0,46	13,84	7,03	3,07	0,35
Urban3	87,50	75,65	66,63	5,22	84,68	76,11	59,96	5,38
Office	0,86	0,02	0,00	0,11	1,76	0,08	0,00	0,15
TT	8,23	3,55	0,26	0,31	10,68	2,28	1,84	0,31
TD	9,00	1,44	0,00	0,16	4,99	3,54	0,05	0,17

Tabelle 5.7: Endpunktfehler des Horn und Schunck Algorithmus und des neuen Ansatzes im Vergleich.

sind die Strukturen besser und klarer als mit dem Verfahren von Horn und Schunck zu erkennen. Zugleich wird hier eine Schwäche beider Verfahren sichtbar, der Umgang mit großen Verschiebungen. Diese können von beiden Verfahren nicht korrekt ermittelt werden, hier wäre eine Berechnung des Flusses innerhalb eines Pyramidenschemas notwendig [29].
Nachteile des neuen Verfahrens lassen sich deutlich an den Sequenzen „TT“ (Tabelle 5.11), „TD“ (Tabelle 5.12) und „Office“ (Tabelle 5.10) erkennen. In diesen Sequenzen wurde der Fluss „künstlich“ erzeugt, im Falle von „TD“ und „Office“ durch eine Vergrößerung eines Ausschnitts des Bildes und in „TT“ durch Verschiebung. Somit entsprechen in diesen Bildern Kanten im Bild nicht unbedingt Kanten im Fluss wie es in vielen realen Sequenzen der Fall ist. Da dies Korrelation zwischen Kanten im Fluss und Kanten im Bild eine zentrale Annahme des neuen Ansatzes ist, kann man hier deutlich die Kanten des Bildes auch im Fluss erkennen. In diesem Fall ist dies jedoch ein ungewollter Effekt, da eine möglichst große Glattheit des Flusses die besten Ergebnisse erzielt.

Ground Truth | Horn & Schunck | Neuer Ansatz

Tabelle 5.8: Ergebnisse der Sequenz „RubberWhale“

Ground Truth | Horn & Schunck | Neuer Ansatz

Tabelle 5.9: Ergebnisse der Sequenz „Urban3“

Ground Truth | Horn & Schunck | Neuer Ansatz

Tabelle 5.10: Ergebnisse der Sequenz „Office“

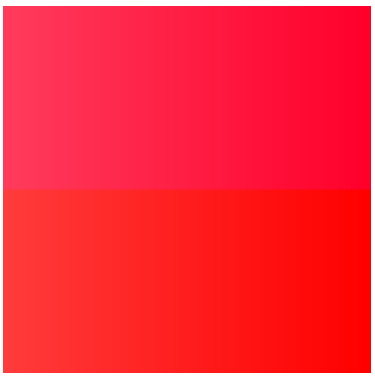
Ground Truth	Horn & Schunck	Neuer Ansatz
		

Tabelle 5.11: Ergebnisse der Sequenz „TT“

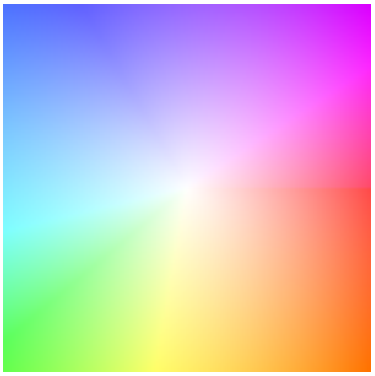
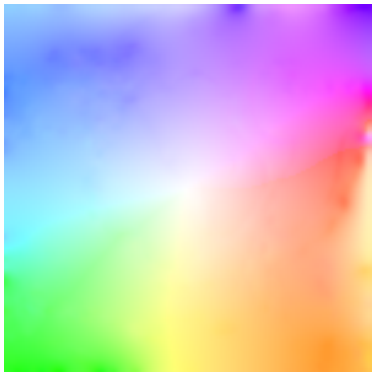
Ground Truth	Horn & Schunck	Neuer Ansatz
		

Tabelle 5.12: Ergebnisse der Sequenz „TD“

6. SCHLUSS

In dieser Arbeit wurde der klassische Variationsansatz von Horn und Schunck mittels eines nicht-lokalen bilateralen Glättungsterm zweiter Ordnung und einer robusteren Konstanzannahme erweitert, implementiert und getestet. Dabei hat sich gezeigt, dass diese Erweiterungen Vor- und Nachteile haben. So werden Kanten im Fluss, sofern diese eine Korrelation zu Kanten im Bild haben, besser erhalten als beim Verfahren von Horn und Schunck. Das Hauptproblem des alten wie des neuen Ansatzes stellen große Verschiebungen dar, die durch die Konstanzannahme nicht abgedeckt werden. Hier wäre die Einbettung in ein Pyramidenschema notwendig, wie zum Beispiel in [21] beschrieben. Eine weitere Verbesserung könnte die Verwendung von Glattheitstermen höherer Ordnung, als die verwendeten, darstellen. Auch könnte man den Algorithmus um einen Datenterm mit einer höheren Robustheit unter Beleuchtungsänderungen erweitern. Verbesserungen der Laufzeit ließen sich durch eine effizientere Implementierung erzielen.

7. ERKLÄRUNG

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman und A. Blake, "Real-time human pose recognition in parts from a single depth image", in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, S. 1297–1304.

[2] U. Franke, C. Rabe und S. K. Gehrig, "Kollisionsvermeidung durch raum-zeitliche Bildanalyse (Collision Avoidance based on Space-Time Image Analysis)", *It - Information Technology*, Bd. 49, Nr. 1, S. 25–32, 2007.

[3] B. K. P. Horn und B. G. Schunck, "Determining optical flow", *Artificial Intelligence*, Bd. 17, S. 185–203, 1981.

[4] C. Zach, T. Pock und H. Bischof, "A duality based approach for realtime TV-L1 optical flow", in *Proceedings German Conference on Pattern Recognition (DAGM)*, 2007, S. 214–223.

[5] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas und N. Navab, "Optical flow estimation with uncertainties through dynamic MRFs", in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[6] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers und H. Bischof, "Anisotropic Huber-L1 optical flow", in *Proceedings British Machine Vision Conference (BMVC)*, 2009.

[7] J. Weickert und C. Schnörr, "Variational optic flow computation with a spatio-temporal smoothness constraint", *Journal of Mathematical Imaging and Vision*, Bd. 14, Nr. 3, S. 245–255, 2001.

[8] G. Gilboa und S. Osher, "Nonlocal operators with applications to image processing", *SIAM Multiscale Modeling & Simulation*, Bd. 7, Nr. 3, S. 1005–1028, 2009.

[9] R. Ranftl, K. Bredies und T. Pock, "Non-local total generalized variation for optical flow estimation", in *Proceedings European Conference on Computer Vision (ECCV)*, 2014, S. 439–454.

[10] C. Tomasi und R. Manduchi, "Bilateral filtering for gray and color images", in *Proceedings IEEE International Conference on Computer Vision (ICCV)*, 1998, S. 839–846.

[11] M. M. Bronstein Semendjajew, *Taschenbuch der Mathematik - 5.Auflage*. Verlag Harri Deutsch, 2001.

[12] N. Setiawan, D. Aurrahman und C. W. Lee, "Optical flow in dynamic graph cuts", in *Proceedings IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2007, S. 288–291.

[13] S. Hermann und R. Klette, "Hierarchical scan-line dynamic programming for optical flow using semi-global matching", in *Proceedings Asia Conference Computer Vision (ACCV), Workshops*, 2013, S. 556–567.

[14] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. Black und R. Szeliski, "A database and evaluation methodology for optical flow", *International Journal of Computer Vision*, Bd. 92, Nr. 1, S. 1–31, 2011.

[15] A. Bruhn, "Vorlesungsskript Imaging Science, Sommersemester 2015",

[16] ——, "Vorlesungsskript Computer Vision, Wintersemester 2014/2015",

[17] A. Geiger, P. Lenz und R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite", in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, S. 3354–3361.

[18] A. Geiger, P. Lenz, C. Stiller und R. Urtasun, "Vision meets robotics: The KITTI dataset", *International Journal of Robotics Research (IJRR)*, 2013.

[19] J. Fritsch, T. Kuehnl und A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms", in *International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2013, S. 1693–1700.

[20] M. Werlberger, "Convex approaches for high performance video processing", *Dissertation, Graz University of Technology*, 2012.

[21] A. Bruhn, "Variational optic flow computation-accurate modelling and efficient numerics", *Dissertation, Saarland University*, 2006.

[22] Y. Saad, "Iterative Methods for Sparse Linear Systems",

[23] H. Takeda, S. Farsiu und P. Milanfar, "Higher order bilateral filters and their properties", in *Proceedings IS&T/SPIE Conference on Electronic Imaging*.

[24] T. F. Chan und P. Mulet, "On the convergence of the lagged diffusivity fixed point method in total variation image restoration", *SIAM Journal on Numerical Analysis*, Bd. 36, Nr. 2, S. 354–367, 1999.

[25] C. Vogel, *Computational Methods for Inverse Problems*. Society for Industrial und Applied Mathematics (SIAM), 2002.

[26] C. Frohn-Schauf, S. Henn und K. Witsch, "Nonlinear multigrid methods for total variation image denoising", *Computing and Visualization in Science*, Bd. 7, Nr. 3-4, S. 199–206, 2004.

[27] J. L. Barron, D. J. Fleet und S. S. Beauchemin, "Performance of optical flow techniques", *International Journal of Computer Vision*, Bd. 12, S. 43–77, 1994.

[28] B. Galvin, B. McCane, K. Novins, D. Mason and S. Mills, "Recovering motion fields: An evaluation of eight optical flow algorithms", in *Proceedings British Machine Vision Conference (BMVC)*, Bd. 98, 1998, S. 195–204.

[29] T. Brox, A. Bruhn, N. Papenberg und J. Weickert, "High accuracy optical flow estimation based on a theory for warping", in *Proceedings European Conference on Computer Vision (ECCV)*, 2004, S. 25–36.