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

Bachelorarbeit Nr. 174

Kombination von Regularisierern verschiedener Ordnung zur Berechnung des optischen Flusses

Sebastian Richter

Studiengang:	Informatik
Prüfer/in:	Prof. Dr.-Ing. Andrés Bruhn
Betreuer/in:	Prof. Dr.-Ing. Andrés Bruhn
Beginn am:	1. Juni 2014
Beendet am:	27. Oktober 2014
CR-Nummer:	I.4.8, I.4.3, I.4.6, I.2.10

ZUSAMMENFASSUNG

Betrachtet man zwei oder mehr Bilder der gleichen Szene, zum Beispiel auch aus einer Videosequenz, stellt man fest, dass eine räumliche Verschiebung der Objekte stattgefunden hat. Diese Verschiebung wird optischer Fluss genannt. Es gibt zahlreiche Algorithmen zur Berechnung des optischen Flusses. Die meisten haben ihre Stärken und Schwächen, abhängig davon, auf welche Art von Bildern sie angewendet werden. Allerdings müssen Annahmen getroffen werden, da die Lösung sonst nicht eindeutig wird. Verschiedene Ansätze und Annahmen werden je nach Bildtyp besser oder schlechter erfüllt. Ein Beispiel dafür ist die Ordnung des Regularisierers des Glattheitsterms. Bei Verwendung der ersten Ordnung wird ein konstanter optischer Fluss angenommen. Das ist dann der Fall, wenn sich das Objekt parallel zur Bildebene bewegt, zum Beispiel bei Bildern, die seitlich aus einem fahrenden Zug gemacht wurden. Verwendet man Glattheitsterme zweiter Ordnung, geht man von einem linearen optischen Fluss aus. Ein Anwendungsbeispiel hierfür wäre die Bilder einer Frontkamera eines Autos. Grundsätzlich kann ein Regularisierer zweiter Ordnung auch konstante Flussfelder erzeugen, ist aber deutlich anfälliger gegenüber Rauschen. In dieser Arbeit wurde ein Verfahren entwickelt, das sowohl mit der ersten als auch der zweiten Ordnung arbeiten kann und sogar selbstständig entscheiden kann, welch von beiden Annahmen für eine gegebene Bildsequenz besser geeignet ist. Dazu wurde zunächst das Verfahren von Horn und Schunck nachimplementiert und mit der Jakobimethode oder dem Gauß-Seidel Verfahren gelöst. Anschließend wird dieser Algorithmus um einige Funktionen erweitert, wie SOR-Verfahren, Gradientenkonstanz, Warping und robusten Datentermen. Danach wird der Algorithmus um einen Regularisierer zweiter Ordnung erweitert. Zum Schluss werden einige Ergebnisse mit erster oder zweiter Ordnung und mit einer Kombination aus beiden beschrieben und diskutiert.

INHALTSVERZEICHNIS

1 EINLEITUNG 1
- 1.1 Motivation . . . 1
- 1.2 Aufgabenstellung . . . 1
- 1.3 Verwandte Arbeiten . . . 2
- 1.4 Gliederung . . . 3

2 GRUNDLAGEN 5
- 2.1 Verfahren von Horn und Schunck . . . 5
- 2.2 Euler-Lagrange Gleichungen . . . 7
- 2.3 Diskretisierung . . . 7
- 2.4 Iterative Solver . . . 8
- 2.5 Jacobi-Methode . . . 9
- 2.6 Gauß-Seidel Verfahren . . . 9
- 2.7 Erste evaluation . . . 10

3 ERWEITERUNGEN 13
- 3.1 Konstanzannahmen höherer Ordnung . . . 13
- 3.2 Warping . . . 13
  - 3.2.1 Backward Registration . . . 14
- 3.3 Robuste Daten- und Glattheitsterme . . . 15
  - 3.3.1 Fluss-gesteuerter robuster Glattheitsterm . . . 16
- 3.4 Succesive Over-Relaxation Verfahren . . . 17
- 3.5 Evaluation . . . 18

4 GLATTHEITSTERME ZWEITER ORDNUNG 19
- 4.1 Modellierung . . . 19
- 4.2 Herleitung des Stencils . . . 20

5 EVALUATION 25
- 5.1 Ergebnisse mit Regularisierung erster Ordnung . . . 25
- 5.2 Ergebnisse mit Regularisierung zweiter Ordnung . . . 31
- 5.3 Kombination der beiden Regularisierer . . . 31
- 5.4 Automatische Erkennung der Ordnung . . . 34

6 ZUSAMMENFASSUNG UND AUSBLICK 35
- 6.1 Zusammenfassung . . . 35
- 6.2 Ausblick . . . 35

LITERATUR 37

A ANHANG 39
- A.1 Ergebnistabellen . . . 39

DANKSAGUNG 41
ERKLÄRUNG 43

1. EINLEITUNG

Die Extraktion von Bewegungsinformation aus Bildfolgen ist eines der zentralen Probleme des Maschinensehens. Typischerweise ist man in diesem Zusammenhang an der Bestimmung des Verschiebungsvektorfeldes zwischen zwei aufeinander folgenden Bildern der Bildfolge interessiert. Dieses Verschiebungsvektorfeld wird in der Literatur auch als optischer Fluss bezeichnet. Seit den ersten wegweisenden Arbeiten von Horn und Schunck auf diesem Gebiet (AI 1981) haben sich globale kontinuierliche Optimierungsmethoden – sogenannte Variationsansätze – als populäre und inzwischen auch sehr genaue Verfahren zur Bestimmung des optischen Flusses etabliert. Solche Optimierungsmethoden basieren auf der Minimierung geeigneter Energiefunktionalen, die Abweichungen von verschiedenen Modellannahmen bestrafen: Während Konstanzannahmen eine Zuordnung zugehöriger Merkmale in aufeinander folgenden Bildern ermöglichen sollen, sorgen räumliche Glattheitsannahmen dafür, dass die oft nicht eindeutige Lösung der Konstanzannahmen regularisiert, d.h. eindeutig gemacht wird.

1.1 MOTIVATION

Bei der Modellierung von Variationsansätzen für bestimmte Anwendungsgebiete (z.B. für Fahrerassistenzsysteme) tritt vor allem die Frage auf, welche Glattheitsterme für die Berechnung des optischen Flusses besonders geeignet sind und wie diese am besten kombiniert, d.h. gewichtet werden können. Während Glattheitsterme erster Ordnung besonders gute Ergebnisse bei fronto-paralleler Bewegung (d.h. bei Bewegung parallel zur Bildebene) erzielen, sind Glattheitsterme zweiter Ordnung besonders bei solchen Bewegungen nützlich, die durch die Eigenbewegung der Kamera entstehen. Offensichtlich ist es wünschenswert die Eigenschaften der beiden Glattheitsterme verschiedener Ordnung zu kombinieren, um eine qualitativ hochwertige Schätzung in beiden Fällen sicherzustellen,.

1.2 AUFGABENSTELLUNG

Ziel der Arbeit ist es deshalb zunächst, ein Basisverfahren für die Berechnung des optischen Flusses zu implementieren. Als Ausgangspunkt soll hierbei das Verfahren von Bruhn et al. (IJCV 2005) dienen. Anschließend sollen neben dem Regularisierer erster Ordnung, der bereits im Modell von Bruhn et al. enthalten ist, auch ein geeignet Regularisierer zweiter Ordnung implementiert werden. Beide Regularisierer sollen dann anhand von Standardbenchmarks (z.B. Middlebury, KITTI, MPI Sintel) untersucht und

verglichen werden. Insbesondere sollen hierbei globale Mechanismen zur Wahl des geeigneten Glattheitsterms entwickelt und evaluiert werden.

1.3 VERWANDTE ARBEITEN

Grundlage des verwendeten Algorithmus sind die Ergebnisse von Horn und Schunck aus dem Jahre 1981. Sie beschrieben den optischen Fluss als Minimierungsproblem eines Energiefunktionals mit einem Daten- und einem Glattheitsterm [1].
In dem Artikel "High Accuracy Optical Flow Estimation Based on a Theory for Warping"[2] von Brox et al. vom Mai 2004 werden einige der Erweiterungen beschrieben, die auch in dieser Arbeit verwendet wurden. Es werden die Vorteile der einzelnen Anpassungen beschrieben und die Ergebnisse mit anderen gängigen Verfahren in Bezug auf ihre Fehler verglichen. Dabei stellten die Verfasser fest, dass ihr Verfahren zum Teil deutlich besser ist als alle Vergleichsalgorithmen. Allerdings gibt es heute noch bessere Verfahren.
Grundlage der Subquadratischen Bestrafungsfunktion ist der Artikel "The Robust Estimation of Multiple Motions: Parametric and Piecewise-Smooth Flow Fields"[3]. Darin wird auch ein Verfahren beschrieben, in dem nicht von einem konstanten optischen Fluss ausgegangen wird, sondern von einem stückweise konstantem optischen Fluss. Das verbessert die Berechnung an Kanten, an denen zwei Bewegungsrichtungen oder -Stärken nahe beieinander liegen, beispielsweise ein Objekt bewegt sich schneller als der Hintergrund.
Ein Vergleich von verschiedenen robusten Glattheitstermen, flussgetriebene, datengetriebene, isotrope und anisotrope Verfahren, wurde in der Arbeit Ä Theoretical Framework for Convex Regularizers in PDE-Based Computation of Image Motion"[4] durchgeführt.
Eine sehr genaue Beschreibung der Funktionsweise des SOR-Verfahrens ist der Artikel Ïterative Mehtods for Solving Partial Equations of Elliptic Type"von David Young aus dem Jahre 1954 [5]. Auch erklärt er genau den Übergang von numerischen Lösungen von Grenzwertproblemen mit partiellen Differentialgleichungen zu einem linearen System.
Grundlage der Variationsrechnungen, die in dieser Arbeit verwendet wurden, entstammen dem Buch "Calculus of Variations"[6].
Eine genaue Analyse und Erklärung für einen Regularisierer zweiter Ordnung gibt es in dem Artikel "Learning Brightness Transfer Functions for the Joint Recovery of Illumination Changes and Optical Flow"[7].
Die Evaluation des Algorithmus geschah anhand der Middlebury Sequenzen [11] und der KITTI Vision Banchmark Suite [8].

1.4 GLIEDERUNG

Im 2. Kapitel werden zuerst die mathematischen Grundlagen wie iterative Löser für lineare Gleichungssysteme und die Euler-Lagrange Gleichungen erklärt. Im Anschluss wird der Basisalgorithmus von Horn und Schunck erklärt, welche Annahmen getroffen werden, um die Lösung eindeutig zu machen und wie die Gleichungssysteme gelöst werden. In Kapitel 3 wird dieses Verfahren nach und nach um verschiedene Konzepte erweitert, die das Ergebnis verbessern können. Im 4. Kapitel wird der bisher verwendete Regularisierer erster Ordnung durch einen der zweiten Ordnung ersetzt. In Kapitel 5 werden mit den verschiedenen Regularisierern Tests durchgeführt und die Ergebnisse miteinander Verglichen. Außerdem wird eine Kombination aus den verschiedenen Verfahren betrachtet und bewertet.

2. GRUNDLAGEN

An dieser Stelle sollen zunächst die (mathematischen) Grundlagen erklärt werden. Im Nachfolgenden Abschnitt soll das verwendete Verfahren beschrieben werden, beginnend mit einem einfachen Algorithmus, der Schritt für Schritt verändert und angepasst wird, um die Qualität der Ergebnisse zu verbessern.

2.1 VERFAHREN VON HORN UND SCHUNCK

Bereits 1981 veröffentlichten Horn und Schunck ein Verfahren zur Berechnung des optischen Flusses [1]. Gesucht wird das Verschiebungsvektorfeld zwischen zwei Bildern, also wie weit ein Pixel von einem Bild zum nächsten verschoben wurde. Aufgrund der Diskretisierung des Bildes in X und Y Richtung, wird dieses in zwei Komponenten u und v aufgeteilt, die für die Verschiebung in x beziehungsweise y Richtung stehen. Der Bildraum wird Ω genannt. Die Bilder werden dabei als Funktion f(x,y,t) betrachtet. X und Y sind die Koordinaten der Pixel, t und t+1 sind das erste beziehungsweise das zweite Bild.
Zunächst müssen wir die einzelnen Pixel auf den unterschiedlichen Bildern wiedererkennen. Dies geschieht anhand der Annahme, dass der Pixel in beiden Bildern den gleichen Grauwert hat (Grauwertkonstanzannahme):

$$f(x, y, t) = f(x + u, y + v, t + 1). \tag{2.1}$$

Linearisiert man das Problem mit einer Taylorentwicklung, erhält man

$$f_x u + f_y v + f_t = 0 \tag{2.2}$$

Diese Annahme hat drei Nachteile: Zum einen können dadurch nur an den Ecken die Bewegungen richtig erkannt werden. Denn bei einer Fläche mit konstantem Grauwert gibt es mehrere Möglichkeiten, welche Bewegung das Objekt gemacht hat, da viele Pixel den gleichen Grauwert haben. Auch bei Kanten kann nur Teil des optischen Flusses korrekt berechnet werden, der senkrecht zur Kante steht. Dieses Problem wird Aperaturproblem genannt. Zum anderen kann es sein, dass die beiden Bilder leicht unterschiedliche Beleuchtungsbedingungen haben, so dass sich der Grauwert eines Pixels verändert. Daraus folgt, es könnte irgendwo im Bild einen Pixel geben, der zufällig den gleichen Grauwert hat und dadurch als Korrespondenzpunkt gewählt wird. Dadurch ist die Lösung nicht immer eindeutig.

Um diese Probleme zu lösen, wird eine weitere Annahme getroffen: Der optische Fluss soll möglichst konstant sein (Glattheitsannahme):

$$|\nabla u|^2 + |\nabla v|^2 = 0 \tag{2.3}$$

Das ermöglicht das Berechnen des optischen Flusses in großen Flächen und verhindert, dass Pixel mit zufälligen anderen Pixeln verrechnet werden.
Da weder die erste noch die zweite Annahme erfüllt sein wird, beschränken wir uns auf eine Minimierung der beiden Terme und führen eine Gewichtung von beiden ein. Die grundsätzliche Gleichung ist hierbei:

$$E(u,v) = \int_\Omega D(u,v) + \alpha * S(u,v)\,dx\,dy \tag{2.4}$$

wobei D(u,v) der Datenterm die Abweichungen von der Konstanzannahme bestraft und der Glattheitsterm S(u,v) die Abweichung der Glattheit der Lösung bestraft. Alpha ist der einstellbaren Parameter zur Gewichtung der beiden Terme. Wählt man α = 0, ignoriert man die Glattheit und achtet nur auf den Datenterm Der optische Fluss (u,v) ist dabei das globale Minimum des oben genannten Energiefunktionals.
Wir definieren den Bewegungstensor J wie folgt, um Diskretisierung und die Schreibweise zu vereinfachen:

$$J = \nabla f \nabla f^\top = \begin{pmatrix} f_x^2 & f_x f_y & f_x f_t \\ f_x f_y & f_y^2 & f_y f_t \\ f_x f_t & f_y f_t & f_t^2 \end{pmatrix} \tag{2.5}$$

Nun lassen sich der Daten- und der Glattheitsterm für das Verfahren von Horn und Schunck schreiben als:

$$D(u,v) = w^\top J w \tag{2.6}$$

$$S(u,v) = |\nabla u|^2 + |\nabla v|^2 \tag{2.7}$$

mit $w = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$ und der Grauwertskonstanzannahme

$$w^\top J w = (f_x u + f_y v + f t)^2 = 0 \tag{2.8}$$

Setzt man die Annahme in die Gleichung 2.4 ein, erhält man

$$E(u,v) = \int_\Omega w^\top J w + \alpha * (|\nabla u|^2 + |\nabla v|^2)\,dx\,dy \tag{2.9}$$

$$= (f_x u + f_y v + f t)^2 + \alpha * (|\nabla u|^2 + |\nabla v|^2)\,dx\,dy \tag{2.10}$$

Dieses Verfahren bewirkt, dass an Kanten eine Bewegung erkannt und berechnet werden kann und in Flächen die Informationen des Nachbarn aufgefüllt wird. Das Minimum des Energiefunktionals wird mithilfe der Euler-Lagrange Gleichung bestimmt.

2.2 EULER-LAGRANGE GLEICHUNGEN

Gegeben sei ein (Energie-)Funktional $E = \int_a^b F(x, y, u, v, u_x, u_y, v_x, v_y)dxdy$, das von einer Funktion f(u,v), deren Ableitung u_x, u_y, v_x, v_y und der Zeit t abhängt. Gesucht sei nun das globale Minimum (oder Maximum) von E beziehungsweise q(t), so dass E extremal wird. Notwendige Bedingung für einen Extremwert ist, dass die Euler-Lagrange-Gleichung erfüllt wird [5]:

$$0 \stackrel{!}{=} F_u - \frac{\partial}{\partial t}F_{u_x} - \frac{\partial}{\partial t}F_{u_y} \tag{2.11}$$

$$0 \stackrel{!}{=} F_v - \frac{\partial}{\partial t}F_{v_x} - \frac{\partial}{\partial t}F_{v_y} \tag{2.12}$$

In unserem Fall gilt:

$$E(u, v) = \int_\Omega F(x, y, u, v, u_x, u_y, v_x, v_y)dxdy \tag{2.13}$$

$$\begin{aligned} \Rightarrow 0 &\stackrel{!}{=} F_u - \frac{\partial}{\partial x}F_{u_x} - \frac{\partial}{\partial y}F_{u_y} \\ 0 &\stackrel{!}{=} F_v - \frac{\partial}{\partial x}F_{v_x} - \frac{\partial}{\partial y}F_{v_y} \end{aligned} \tag{2.14}$$

mit Neumann Randbedingungen

$$n^\top \begin{pmatrix} F_{u_x} \\ F_{u_y} \end{pmatrix} = 0 = n^\top \begin{pmatrix} F_{v_x} \\ F_{v_y} \end{pmatrix} \tag{2.15}$$

Im Fall von Horn und Schunck gilt

$$F = w^\top J w + \alpha(|\nabla u|^2 + |\nabla v|^2) \tag{2.16}$$

$$= J_{11}u^2 + J_{22}v^2 + J_{33} + 2J_{12}uv + 2J_{13}u + 2j_{23}v + \alpha(u_x^2 + u_y^2 + v_x^2 + v_y^2) \tag{2.17}$$

Dadurch erhällt man:

$$0 = J_{11}u + J_{12}v + J_{13} - \alpha\Delta u \tag{2.18}$$

$$0 = J_{12}u + J_{22}v + J_{23} - \alpha\Delta v \tag{2.19}$$

$J_{i,j}$ ist dabei der Eintrag (i,j) der Matrix J.

2.3 DISKRETISIERUNG

Da wir mit diskreten Werten arbeiten, müssen u, v, $J_{11} - J_{33}, \Delta u$ und Δv diskretisiert werden. Daher wird aus u und v wird $u_{i,j} = u(i * h_x, j * h_y)$ beziehungsweise

$v_{i,j} = v(i * h_x, j * h_y)$ mit der Gittergröße hx und hy. Die diskreten Einträge des Bewegungsternsors können wie folgt berechnet werden:

- $[J_{11}]_{i,j} = [f_x]^2_{i,j}$
- $[J_{22}]_{i,j} = [f_y]^2_{i,j}$
- $[J_{33}]_{i,j} = [f_t]^2_{i,j}$
- $[J_{12}]_{i,j} = [f_x]_{i,j} * [f_y]_{i,j}$
- $[J_{13}]_{i,j} = [f_x]_{i,j} * [f_t]_{i,j}$
- $[J_{23}]_{i,j} = [f_y]_{i,j} * [f_t]_{i,j}$

Das wiederum führt dazu, dass f_x, f_y, f_t diskretisiert werden müssen und daher auch f. Dies geschieht durch:

$$f_{i,j,t} = f(i * h_x, j * h_y, t) \tag{2.20}$$

Die Ableitungen von f in x und y Richtung werden mit gemittelten zentralen Differenzen approximiert, f_t mit Vorwärtsdifferenzen:

$$[f_x]_{i,j} \approx \frac{1}{2}\left(\frac{f_{i+1,j,t+1} - f_{i-1,j,t+1}}{2h_x} + \frac{f_{i+1,j,t} - f_{i-1,j,t}}{2h_x}\right) \tag{2.21}$$

$$[f_y]_{i,j} \approx \frac{1}{2}\left(\frac{f_{i,j+1,t+1} - f_{i,j-1,t+1}}{2h_x} + \frac{f_{i,j+1,t} - f_{i,j-1,t}}{2h_x}\right) \tag{2.22}$$

$$[f_t]_{i,j} \approx \frac{f_{i,j,t+1} - f_{i,j,t}}{h_t} \tag{2.23}$$

$\Delta u = u_{xx} + u_{yy}$ bezeihungsweise $\Delta v = v_{xx} + v_{yy}$ werden approximiert durch:

$$\Delta u \approx \frac{u_{i+1,j} - u_{i,j}}{h_x^2} - \frac{u_{i,j} - u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - u_{i,j}}{h_y^2} - \frac{u_{i,j} - u_{i,j-1}}{h_y^2} \tag{2.24}$$

$$\Delta v \approx \frac{v_{i+1,j} - v_{i,j}}{h_x^2} - \frac{v_{i,j} - v_{i-1,j}}{h_x^2} + \frac{v_{i,j+1} - v_{i,j}}{h_y^2} - \frac{v_{i,j} - v_{i,j-1}}{h_y^2} \tag{2.25}$$

2.4 ITERATIVE SOLVER

Ein lineares Gleichungsystem der Art Ax = b nach x zu Lösen ist in der Theorie sehr einfach (Beipsiel Gaußsches Lösungsverfahren). Dieses Verfahren hat den Nachteil, dass für eine Matrix der Größe n^2 bis zu $O(n^3)$ Operationen nötig sind. Der derzeit beste bekannte Algorithmus benötigt immerhin noch $O(n^{2,376})$ Operationen. Da bei den weiteren Berechnungen n der doppelten Anzahl von Pixeln entspricht, und auch

Bilder mit bis zu 1 Megapixel berechnet werden sollen, sind diese Verfahren nicht geeignet. Die Idee ist nun, eine Zerlegung zu finden für die gilt:

$$A = A_1 + A_2 \Rightarrow (A_1 + A_2)x = b \Leftrightarrow A_1 x = b - A_2 x \tag{2.26}$$

Eine Fixpunktiteration berechnet dann aus einem bekannten oder geratenem x^k einen besseren Schätzwert x^{k+1}:

$$A_1 x^{k+1} = b - A_2 x^k \tag{2.27}$$

$$\Leftrightarrow x^{k+1} = A_1^{-1}(b - A_2 x^k) \tag{2.28}$$

A_1 sollte A möglichst gut approximieren, gleichzeitig leicht zu invertieren sein, also zum Beispiel eine Diagonalmatrix. für $k \to \infty$ konvergiert das Verfahren gegen die Lösung des linearen Gleichungssystems.

2.5 JACOBI-METHODE

Für die Berechnung des nächsten Schrittes $u_{i,j}^{k+1}$ beziehungsweise $v_{i,j}^{k+1}$ aus den alten Zwischenergebnissen $u_{i,j}^k$ und $v_{i,j}^k$ wird die Jakobi Methode angewendet. A wird dabei in D (Diagonale von U) und (L+U)(unterer und oberer Dreiecksmatrix) zerlegt. Daraus folgt:

$$x^{k+1} = D^{-1}(b + (L + U)x^k) \Leftrightarrow x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{i,j} x_j^k \right) \tag{2.29}$$

Auf die Horn und Schunck Methode angewendet, ergibt sich:

$$u_{i,j}^{k+1} = \frac{(-[J_{13}]_{i,j} - ([J_{12}]_{i,j} * v_{i,j}^k - \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{u_{\tilde{i},\tilde{j}}^k}{h_l^2}))}{[J_{11}]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_k^2}} \tag{2.30}$$

$$v_{i,j}^{k+1} = \frac{(-[J_{23}]_{i,j} - ([J_{12}]_{i,j} * u_{i,j}^k - \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{v_{\tilde{i},\tilde{j}}^k}{h_l^2}))}{[J_{22}]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_k^2}} \tag{2.31}$$

Eine Möglichkeit der Verbesserung ist das Glätten der Eingangsbilder mit einem Gauß-filter mit der Standartabweichung σ . Das verringert den Einfluss von Rauschen. Außerdem wird das Bild unendlich oft ableitbar, was für die weiteren verfahren wichtig wird.

2.6 GAUSS-SEIDEL VERFAHREN

Der Unterschied zwischen der Jacobi Methode und dem Gauß-Seidel Verfahren ist, dass bei letzterem die Zwischenergebnisse der aktuellen Iteration für die Berechnung

Abbildung 2.1: Die farbliche Darstellung des optischen Flusses. Der Farbton gibt die Richtung und die Helligkeit die Länge der Verschiebung an.

der weiteren Ergebnisse mitverwendet werden. Formal führt das zu folgender Gleichung:

$$u_{i,j}^{k+1} = \frac{(-[J_{13}]_{i,j} - ([J_{12}]_{i,j} * v_{i,j}^{k} - \alpha \sum_{l \in x,y} \sum_{N_l^-(i,j)} \frac{u_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} - \alpha \sum_{l \in x,y} \sum_{N_l^+(i,j)} \frac{u_{\tilde{i},\tilde{j}}^{k}}{h_l^2}))}{[J_{11}]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_k^2}} \tag{2.32}$$

$$v_{i,j}^{k+1} = \frac{(-[J_{23}]_{i,j} - ([J_{12}]_{i,j} * u_{i,j}^{k+1} - \alpha \sum_{l \in x,y} \sum_{N_l^-(i,j)} \frac{v_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} - \alpha \sum_{l \in x,y} \sum_{N_l^+(i,j)} \frac{v_{\tilde{i},\tilde{j}}^{k}}{h_l^2}))}{[J_{22}]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\tilde{i},\tilde{j}) \in N_l(i,j)} \frac{1}{h_k^2}} \tag{2.33}$$

wobei $N_l^-(i,j)$ die Pixel der Nachbarschaft von i,j bezeichnet, die in der aktuellen Iteration bereits neu berechnet wurden und $N_l^+(i,j)$ diejenigen Pixel bezeichnet, die noch nicht neu berechnet wurden. Obwohl die Gleichung zunächst komplizierter aussieht, bringt sie zwei wesentliche Vorteile: Die alten Ergebnisse müssen nicht solange gespeichert werden, bis die neue Matrix komplett berechnet wurde, man spart sich also den Speicher für die alten u und v. Zum anderen ist das Verfahren schneller, da neue, bessere Zwischenergebnisse schneller propagiert werden.

2.7 ERSTE EVALUATION

Um berechneten Ergebnisse und den Einfluss der verschiedenen Erweiterungen einschätzen zu können, wurden an dieser Stelle bereits Tests durchgeführt. Grundlage dafür war die Yosemite Sequece with clouds. Als Fehlermaße des berechneten Flussfeldes u^e gegenüber der Ground Truth u^twerden der Äverage Angular Error"(AAE)

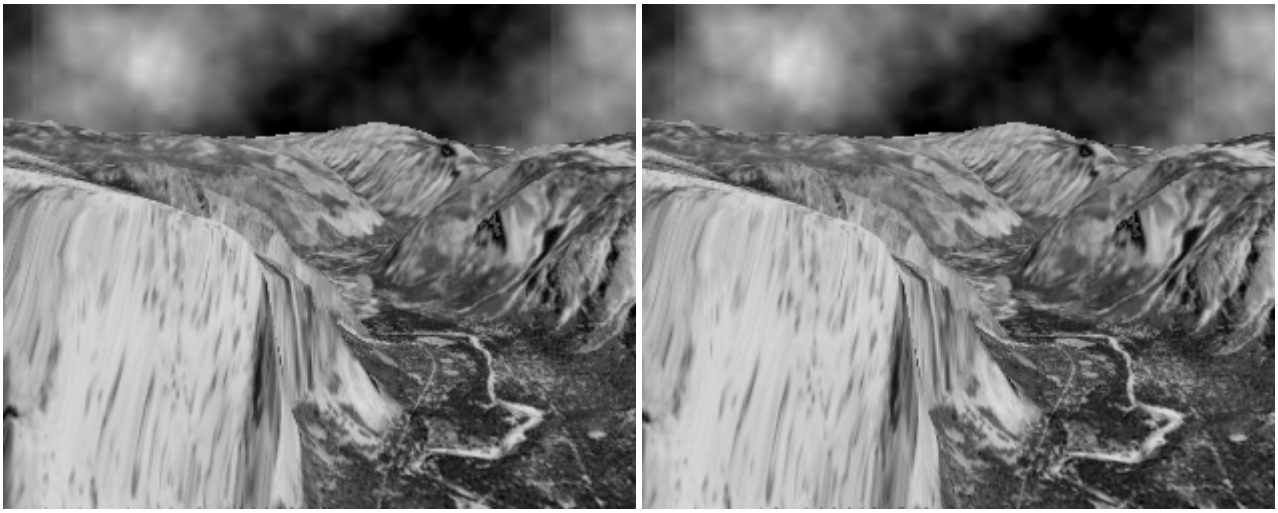

Abbildung 2.2: Die Bilder 8 und 9 der Yosemite Sequenz

und der Äverage Endpoint Error"(AEE) verwendet. Bei einem Bild mit N * M Pixeln berechnen sich die Fehler wie folgt:

$$AAE(u^e, u^t) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \arccos \left(\frac{u^t_{i,j} u^e_{i,j} + v^t_{i,j} v^e_{i,j} + 1}{\sqrt{{u^t_{i,j}}^2 {v^t_{i,j}}^2 + 1} \sqrt{{u^e_{i,j}}^2 {v^e_{i,j}}^2 + 1}} \right) \tag{2.34}$$

$$AEE(u^t, u^e) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \sqrt{(u^t_{i,j} - u^e_{i,j})^2 + (v^t_{i,j} - v^e_{i,j}])^2} \tag{2.35}$$

Mit der Jakobimethode kann man Ergebnisse mit einem durchschnittlichen AAE von ca $7,61°$ erreichen.

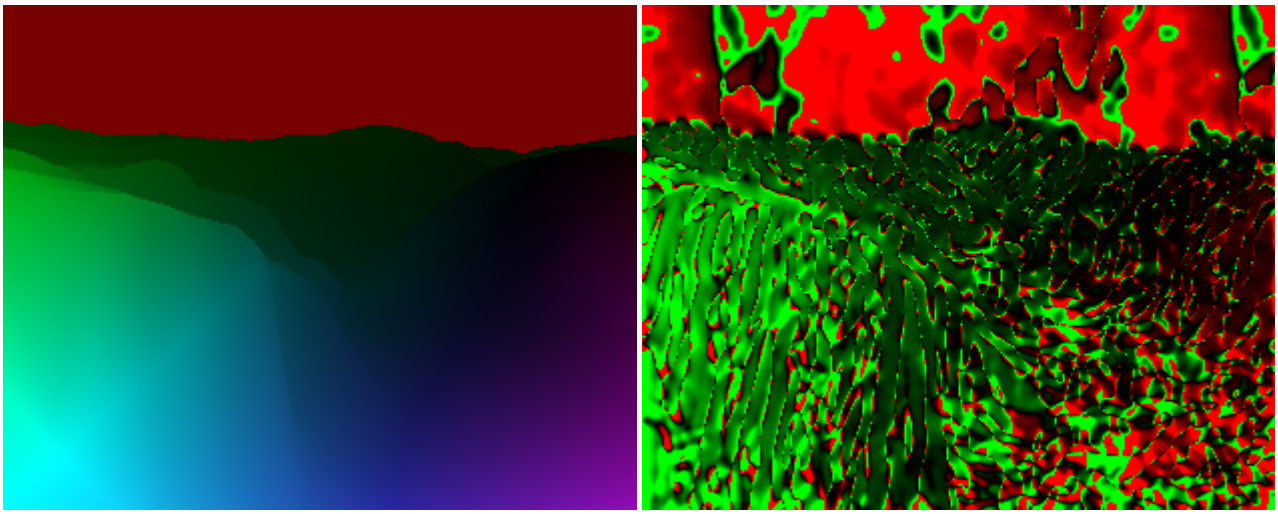

Abbildung 2.3: Links: Die Ground Truth der Yosemite Sequenz. Rechts: Das berechnete Ergebnis mit Basisalgorithmus von Horn und Schunck, ohne Glattheitsterm, Sigma = 2,0. Der AAE ist 43,527 und der AEE 7,956. Quelle: Niklas Kaulitz

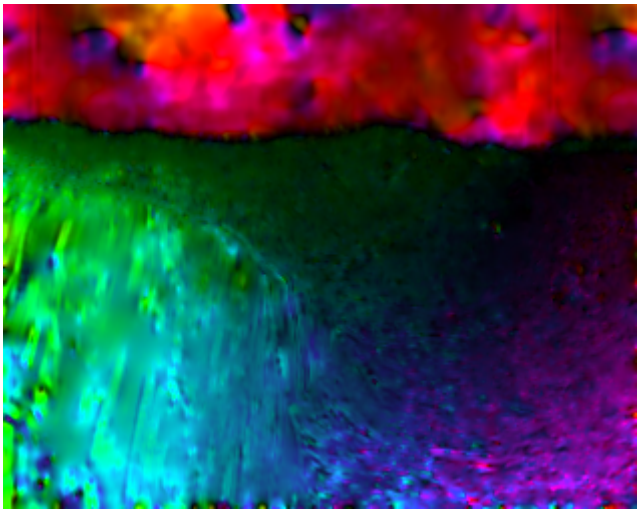

Abbildung 2.4: Horn und Schunck Algorithmus mit Grauwertkonstanz und Glattheitsterm, Sigma = 1,0, alpha = 15, 200 Iterationen. Der AAe liegt bei 11,488 und der AEE bei 0,725. Quelle: Niklas Kaulitz

3. ERWEITERUNGEN

Nachdem wir nun einen funktionierenden, aber noch recht ungenauen Algorithmus zur Bestimmung des optischen Flusses haben, werden nun einige Konzepte eingeführt, um das Ergebnis zu verbessern und um die Berechnung zu beschleunigen.

3.1 KONSTANZANNAHMEN HÖHERER ORDNUNG

Um das Verfahren weiter zu verbessern, wird statt der Grauwertkonstanzannahme nun die Konstanz der Ableitung vorausgesetzt. Das führt zu einer Invarianz unter globalen additiven Beleuchtungsänderungen, aber auch zu erhöhter Sensibilität durch Rauschen. Statt den Formeln 2.18 und 2.19 verwendet man nun:

$$f_x(x, y, t) - f_x(x + u, y + v, t + 1) = 0 \tag{3.1}$$
$$f_y(x, y, t) - f_y(x + u, y + v, t + 1) = 0 \tag{3.2}$$

Durch die Linearisierung ergibt sich:

$$f_{xx}u + f_{xy}v + f_{xt} = 0 \tag{3.3}$$
$$f_{yx}u + f_{yy}v + f_{yt} = 0 \tag{3.4}$$

Nun setzt man diese Annahme in das Horn und Schunck verfahren ein:

$$E(u, v) = \int_\Omega (f_{xx}u + f_{xy}v + f_{xt})^2 + (f_{yx}u + f_{yy}v + f_{yt})^2 \alpha * (|\nabla u|^2 + |\nabla v|^2) dx dy \tag{3.5}$$

Die entsprechenden Euler-Lagrange Gleichungen lauten

$$f_{xx}(f_{xx}u + f_{xy}v + f_{xt}) + f_{yx}(f_{yx}u + f_{yy}v + f_{yt}) - \alpha\Delta u = 0 \tag{3.6}$$
$$f_{xy}(f_{xx}u + f_{xy}v + f_{xt}) + f_{yy}(f_{yx}u + f_{yy}v + f_{yt}) - \alpha\Delta u = 0 \tag{3.7}$$

und werden analog zu 2.18 und 2.19 mit 2.32 und 2.33 gelöst.
Die Konstanzannahmen können sich auf noch höhere Ableitungen beziehen, zum Beispiel auf die Hesse-Matrix des Bildes. Je höher die Ordnung, desto mehr Gleichungen ergeben sich.

3.2 WARPING

In dem bisherigen Verfahren ist das Energiefunktional wegen der Linearisierung der Konstanzannahme konvex. Dadurch können nur kleine Verschiebungen korrekt berechnet werden. Bei großen Verschiebungen ist diese Linearisierung nicht möglich, daher ist auch das Energiefunktional nicht konvex. Daher kann das Ergebnis in lokalen

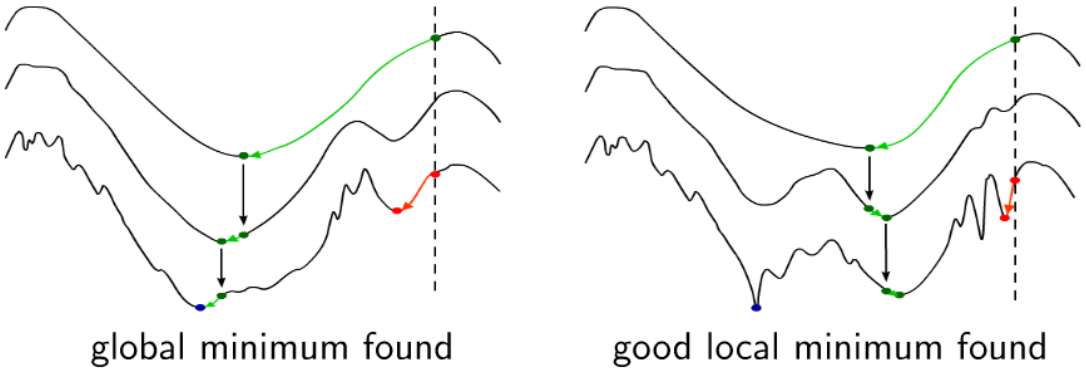

Abbildung 3.1: 2 Beispiele, bei denen ein besseres Minimum durch die Warping-Strategie gefunden wurde. Die gestrichelte Linie gibt die Startwerte an. Grün ist der Verlauf des Zwischenergebnisses mit Warping, rot ist der Verlauf ohne Warping. Quelle: Vorlesung "Correspondence Problems in Computer Vision", Andrés Bruhn, 2014

Minima „steckenbleiben“. Um das zu verhindern kann die Warping-Strategie benutzt werden [2]. Hierbei wird das Bild zunächst in einer gröberen Auflösung betrachtet und dabei eine lokale Lösung berechnet. Dieses Ergebnis wird auf der nächst feineren Auflösung als Startwert benutzt. Das verringert die Gefahr, in einem schlechten lokalen Minimum zu enden und erhöht die Chance, das globale Minimum oder zumindest ein gutes lokales Minimum zu finden. Zusätzliche Parameter sind nun der Faktor, um den die Pixelzahl verringert wird pro Warpingstufe, und die maximale Anzahl von Warpingstufen, die ausgeführt werden soll.

3.2.1 *Backward Registration*

Bisher werden für jeden Iterationsschritt die Bilder f(x,y,t) und f(x,y,t+1) verwendet. Nun soll aber das bisher berechnete Vektorfeld (u,v) genau die Verschiebung zwischen den beiden Bildern angeben. Man kann daher das zweite Bild um den bisherigen Fehler kompensieren, um für die nächste Warpingstufe nur noch ein kleineres Flussfeld berechnen zu müssen. Daher berechnet man nach jedem Warplevel $f(x+u^k, y+v^k, t+1)$ aus f(x,y, t+1) und (u^k, v^k). Mit bilinearer Interpolation lässt sich dabei Subpixel Präzision erreichen:

$$\begin{aligned}[f(x+u^k, y+v^k, t+1)]_{i,j} =& (1-\epsilon_u)(1-\epsilon_v) f_{(i+\bar{u}),(j+\bar{v}),t+1} \\ &+ (\epsilon_u)(1-\epsilon_v) f_{(i+\bar{u})+1,(j+\bar{v}),t+1} \\ &+ (1-\epsilon_u)(\epsilon_v) f_{(i+\bar{u}),(j+\bar{v})+1,t+1} \\ &+ (\epsilon_u)(\epsilon_v) f_{(i+\bar{u})+1,(j+\bar{v})+1,t+1}\end{aligned} \tag{3.8}$$

mit $u = \bar{u} + \epsilon_u$ und $v = \bar{v} + \epsilon_v$. $\bar{u}$ gibt dabei die ganzzahlige Pixelverschiebung und ϵ_u die Subpixelverschiebung an. Dadurch werden nach jedem Warplevel nur noch die Unterschiede zwischen der bisherigen Lösung und dem Originalbild berechnet. Die Werte von u und v werden dadurch immer präziser. Die Gesamtlösung ist die Summe aller (u,v) von allen Stufen.

3.3 ROBUSTE DATEN- UND GLATTHEITSTERME

Der bisherige Algorithmus ist relativ anfällig gegenüber Ausreißern im Datenterm, zum Beispiel durch Rauschen. Um deren Einfluss zu verringern, wird der quadratische Datenterm aus 2.9 durch einen linearen ersetzt [3]:

$$E(u,v) = \int_\Omega \Psi(w^\top J w) + \alpha * (|\nabla u|^2 + |\nabla v|^2) dx dy \tag{3.9}$$

mit $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$. Die zugehörigen Euler-Lagrangegleichungen sind

$$0 = \Psi'(w^\top J w)(J_{11}u + J_{12}v + J_{13}) - \alpha \Delta u \tag{3.10}$$

$$0 = \Psi'(w^\top J w)(J_{12}u + J_{22}v + J_{23}) - \alpha \Delta u \tag{3.11}$$

mit

$$\Psi'(s^2) = \frac{\delta \Psi(s^2)}{\delta s^2} = \frac{1}{2\sqrt{s^2 + \epsilon^2}} \tag{3.12}$$

Diese Formel ist nun nichtlinear in u und v. Ausreißer im Datenterm werden dadurch herunter gewichtet. Nach der Diskretisierung entsteht folgendes nichtlineare Gleichungssystem:

$$0 = [\Psi']_{i,j}([J_{11}]_{i,j}u_{i,j} + [J_{12}]_{i,j}v_{i,j} + [J_{13}]_{i,j}) - \alpha \sum_{l \in x,y} \sum_{N_l(i,j)} \frac{u_{\tilde{i},\tilde{j}} - u_{i,j}}{h_l^2} \tag{3.13}$$

$$0 = [\Psi']_{i,j}([J_{12}]_{i,j}u_{i,j} + [J_{22}]_{i,j}v_{i,j} + [J_{23}]_{i,j}) - \alpha \sum_{l \in x,y} \sum_{N_l(i,j)} \frac{v_{\tilde{i},\tilde{j}} - v_{i,j}}{h_l^2} \tag{3.14}$$

für i = 1...N und j = 1...M mit

$$[\Psi']_{i,j} = \Psi'(w_{i,j}^\top J_{i,j} w_{i,j}) = \Psi' \left(\begin{pmatrix} u_{i,j} \\ v_{i,j} \\ 1 \end{pmatrix}^\top \begin{pmatrix} [J_{11}]_{i,j} [J_{12}]_{i,j} [J_{13}]_{i,j} \\ [J_{12}]_{i,j} [J_{22}]_{i,j} [J_{23}]_{i,j} \\ [J_{13}]_{i,j} [J_{23}]_{i,j} [J_{33}]_{i,j} \end{pmatrix} \begin{pmatrix} u_{i,j} \\ v_{i,j} \\ 1 \end{pmatrix} \right) \tag{3.15}$$

Das Problem dabei ist, dass das nichtlineare Gleichungssystem nicht direkt mit einem iterativen Löser berechnet werden kann. Da eine exakte Berechnung aufgrund

der Größe der Daten nicht in Frage kommt, wird stattdessen wird eine Reihe von linearen Gleichungen gelöst. Dafür wird ein $\Psi'()$ fixiert und damit das nun lineare Gleichungssystem für u und v gelöst:

$$0 = [\Psi']^k_{i,j}([J_{11}]_{i,j}u_{i,j} + [J_{12}]_{i,j}v_{i,j} + [J_{13}]_{i,j}) - \alpha \sum_{l \in x,y} \sum_{N_l(i,j)} \frac{u_{\tilde{i},\tilde{j}} - u_{i,j}}{h_l^2} \tag{3.16}$$

$$0 = [\Psi']^k_{i,j}([J_{12}]_{i,j}u_{i,j} + [J_{22}]_{i,j}v_{i,j} + [J_{23}]_{i,j}) - \alpha \sum_{l \in x,y} \sum_{N_l(i,j)} \frac{v_{\tilde{i},\tilde{j}} - v_{i,j}}{h_l^2} \tag{3.17}$$

Anschließend wird $\Psi'()^k + 1$ berechnet. Dieses Verfahren nennt sich "lagged nonlinearity method"(hinterher hinkende, nichtlineare Methode), da das $\Psi'()$ der Berechnung von u und v hinterher hinkt. Für jedes lineare System ist nur eine ungefähre Lösung notwendig, um eine gute finale Lösung zu erhalten.
Die Vorteile dieser Erweiterung sind bessere Ergebnisse, da Ausreißer und Rauschen weniger gewichtet werden. Das Verfahren kann auf beliebige Konstanzannahmen angewandt werden, es kann sogar aus mehreren die "beste"herausfinden, da schlechte weniger gewichtet werden. Durch die Berechnung über eine Reihe von linearen Gleichungssystemen entsteht nur ein geringer Programmieraufwand. Allerdings ist die Berechnung relativ zeitaufwendig, da das nichtlineare System mit zwei statt wie bisher mit einer Schleife gelöst werden muss. Außerdem können kleine Strukturen verschwinden, da sie als Ausreißer betrachtet werden.

3.3.1 *Fluss-gesteuerter robuster Glattheitsterm*

Neben dem Datenterm kann man auch den Glattheitsterm durch eine subquadratischen Term ersetzen:

$$E(u,v) = \int_\Omega w^\top J w + \alpha \Psi(|\nabla u|^2 + |\nabla v|^2) dx dy \tag{3.18}$$

mit $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$. Dies erlaubt stückweise glatte Verschiebungsfelder. Noch besser jedoch sind gerichtete stückweise konstante Verschiebungsfelder. Das erreicht man durch einen anisotropischen, flussgesteuerten Glattheitsterm:

$$E(u,v) = \int_\Omega w^\top J w + \alpha \operatorname{tr} \Psi(\nabla u \nabla u^\top + \nabla v \nabla v^\top) dx dy \tag{3.19}$$

mit

$$\Psi(A) = (e_1, e_2) \begin{pmatrix} \Psi(\lambda_1) & 0 \\ 0 & \Psi(\lambda_2) \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} \tag{3.20}$$

Die zugehörigen Euler-Lagrange Gleichungen sind:

$$0 = J_{11}u + J_{12}v + J_{13} - \alpha \operatorname{div}(D\Delta u) \tag{3.21}$$

$$0 = J_{12}u + J_{22}v + J_{23} - \alpha \operatorname{div}(D\Delta v) \tag{3.22}$$

D ist dabei der Diffusionstensor, eine 2x2 Matrix. Vergleiche die verschiedenen Methoden:

$$\text{Horn/Schunck} : D = I \tag{3.23}$$

$$\text{Flussgesteuert, Isotrop} : D = \Psi'(|\nabla u|^2 + |\nabla v|^2)I \tag{3.24}$$

$$\text{Flussgesteuert, Anisotrop} : D = \Psi'(\nabla u \nabla u^\top + \nabla v \nabla v^\top) \tag{3.25}$$

Dabei gilt wie beim Datenterm: $\Psi'(s^2) = \frac{1}{2\sqrt{s^2+\epsilon^2}}$ mit kleinem $\epsilon > 0$. Die diskretisierung erfolgt wie im Kapitel zuvor.

3.4 SUCCESIVE OVER-RELAXATION VERFAHREN

Der letzte Schritt zu dem verwendeten Basisverfahren ist die Beschleunigung des iterativen Lösers, dem Gauß-Seidel Verfahren. Dazu wird das SOR-Verfahren (Succesive Over Relaxation, Überrelaxationsverfahren) angewandt [5]. Allgemein wird für das neue Zwischenergebnis eine Art gewichtetes Mittel aus dem alten Zwischenergebnis und einem normal berechnetem Ergebnis genommen:

$$x_i^{k+1} = (1-\omega)x_i^k + \omega\, \bar{x}_i^{k+1} \tag{3.26}$$

mit dem Ergebnis des Gauß-Seidel Verfahrens $\bar{x}_i^{k+1}$ und dem overrelaxation Parameter $\omega \in [1,2)$. Typischerweise ist $\omega > 1$(Extrapolation). Das Verfahren erfüllt folgende Eigenschaften: Für $\omega = 1$ ist es gleich der normalen Gauß-Seidel Methode. Für $\omega \in [u,2)$ konvergiert es gegen die Lösung des linearen Gleichungssystems. Bei geschickt gewähltem ω ist es 1-2 Größenordnungen schneller als das alte Verfahren. Wendet man dieses Verfahren auf unseren Horn-Schunck Algorithmus an, ergeben sich folgende Gleichungen:

$$u_{i,j}^{k+1} = (1-\omega)u_{i,j}^k + \omega \frac{(-[J_{13}]_{i,j} - ([J_{12}]_{i,j} * v_{i,j}^k - \alpha \sum_{l\in x,y} \sum_{N_l^-(i,j)} \frac{u_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} - \alpha \sum_{l\in x,y} \sum_{N_l^+(i,j)} \frac{u_{\tilde{i},\tilde{j}}^{k}}{h_l^2}))}{[J_{11}]_{i,j} + \alpha \sum_{l\in x,y} \sum_{(\tilde{i},\tilde{j})\in N_l(i,j)} \frac{1}{h_k^2}} \tag{3.27}$$

$$v_{i,j}^{k+1} = (1-\omega)v_{i,j}^k + \omega \frac{(-[J_{23}]_{i,j} - ([J_{12}]_{i,j} * u_{i,j}^{k+1} - \alpha \sum_{l\in x,y} \sum_{N_l^-(i,j)} \frac{v_{\tilde{i},\tilde{j}}^{k+1}}{h_l^2} - \alpha \sum_{l\in x,y} \sum_{N_l^+(i,j)} \frac{v_{\tilde{i},\tilde{j}}^{k}}{h_l^2}))}{[J_{22}]_{i,j} + \alpha \sum_{l\in x,y} \sum_{(\tilde{i},\tilde{j})\in N_l(i,j)} \frac{1}{h_k^2}} \tag{3.28}$$

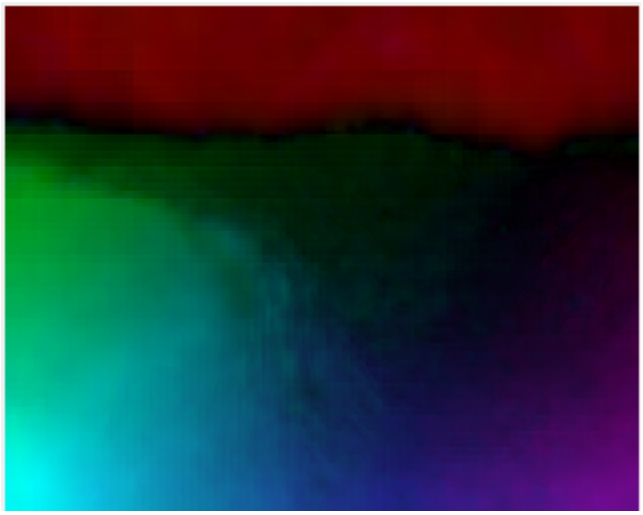

Abbildung 3.2: Der optische Fluss der Yosemite Sequenz. Parameter: $\alpha = 15.\eta = 0,6.\sigma = 1.\omega = 1,9.\epsilon = 0,1$. Warplevel = 10. inner iterations = 10. outer iterations = 20. Der AAE ist 4,329

3.5 EVALUATION

Nachdem sich das Verfahren nun deutlich verändert hat, werfen wir nun erneut einen Blick auf die Yosemite Sequenz. Das alte Verfahren lieferte einen AAE von $7,61°$. Mit der Umstellung auf Gradientenkonstanz lassen sich Ergebnisse mit einem AAE von circa $5,91°$ erreichen. Durch das Warping und die Backwardregistration verbessert sich der Wert dadurch auf $5,14°$. Verwendet man die robusten Daten- und Glattheitsterme statt dem Warping, lassen sich Fehler von $5,21°$ erreichen. Kombiniert man alle bisher vorgestellten Konzepte, kann man Werte von $4,3°$ erreichen.

4. GLATTHEITSTERME ZWEITER ORDNUNG

Das im vorherigen Kapitel beschriebene Verfahren von Horn und Schunck mit allen Erweiterungen benutzt Glattheitsterme 1. Ordnung. Hierbei werden für jedes Pixel des Flussfeldes nur die vier direkten Nachbarn betrachtet. Wie in der Einleitung und Aufgabenstellung beschrieben ist, gibt es aber auch Fälle, in denen die 2. Ordnung besser ist. Dies ist zum Beispiel dann der Fall, wenn das Bild nicht parallel zur Bildebene verschoben wird, sondern sich die Kamera auf ein Objekt zubewegt, wie zum Beispiel bei einer Frontkamera eines Autos. Generell gilt zwar, dass ein Regularisierer zweiter Ordnung bei (lokal) konstantem optischen Fluss die gleiche Lösung berechnen kann. Allerdings ist hier der Algorithmus anfälliger auf Rauschen und anderen Störungen. Vergleiche dazu Interpolation oder Least-Square-Fit mit unterschiedlichen Ordnungen: Liegen zwei Messwerte auf einer Geraden parallel zur X Achse, werden sowohl lineare als auch konstante Approximation die gleiche Gerade als Ergebnis liefern. Liegt nur ein Punkt etwas daneben (wie zum Beispiel durch Messfehler), wird die lineare Interpolation ein grundsätzlich anderes Ergebnis liefern, während die konstante Approximation eine ähnliche Lösung wie zuvor liefert, nämlich den Mittelwert aus beiden Punkten.

4.1 MODELLIERUNG

Der Glattheitsterm der 1. Ordnung wird nun durch einen der 2. Ordnung ersetzt. Um die Regularisierer der verschiedenen Ordnungen besser voneinander unterscheiden zu können, nennen wir die Gewichtung des Glattheitsterms nun β. Das Energiefunktional ändert sich nun zu:

$$E(u,v) = \int_\Omega D(u,v) + \beta\Psi(\|\mathscr{H}u\|_F^2 + \|\mathscr{H}v\|_F^2) \tag{4.1}$$

wobei $\|\mathscr{H}v\|_F$ die Frobeniusnorm der Hessematrix ist [7]. Das führt zu folgenden Euler-Lagrange Gleichungen:

$$F_u - \frac{\partial}{\partial x}F_{u_x} - \frac{\partial}{\partial y}F_{u_y} - \frac{\partial}{\partial xx}F_{u_{xx}} - \frac{\partial}{\partial xy}F_{u_{xy}} - \frac{\partial}{\partial yx}F_{u_{yx}} - \frac{\partial}{\partial yy}F_{u_{yy}} = 0 \tag{4.2}$$

$$F_v - \frac{\partial}{\partial x}F_{v_x} - \frac{\partial}{\partial y}F_{v_y} - \frac{\partial}{\partial xx}F_{v_{xx}} - \frac{\partial}{\partial xy}F_{v_{xy}} - \frac{\partial}{\partial yx}F_{v_{yx}} - \frac{\partial}{\partial yy}F_{v_{yy}} = 0 \tag{4.3}$$

mit den Randbedingungen

$$h^\top \begin{pmatrix} F_{u_{xx}} \\ F_{u_{xy}} \end{pmatrix} = 0 \tag{4.4}$$

$$h^\top \begin{pmatrix} F_{u_{xy}} \\ F_{u_{yy}} \end{pmatrix} = 0 \tag{4.5}$$

$$h^\top \begin{pmatrix} F_{v_{xx}} \\ F_{v_{xy}} \end{pmatrix} = 0 \tag{4.6}$$

$$h^\top \begin{pmatrix} F_{v_{xy}} \\ F_{v_{yy}} \end{pmatrix} = 0 \tag{4.7}$$

Damit ergibt sich folgendes Energiefunktional, das wie bisher minimiert werden soll:

$$E(w) = \int_\Omega w^\top J w + \beta\Psi\left(\left(\frac{\partial^2}{\partial x^2}u\right)^2 + 2\left(\frac{\partial^2}{\partial x \partial y}u\right) + \left(\frac{\partial^2}{\partial y^2}u\right)^2 + \left(\frac{\partial^2}{\partial x^2}v\right)^2 + 2\left(\frac{\partial^2}{\partial x \partial y}v\right) + \left(\frac{\partial^2}{\partial y^2}v\right)^2\right) dx dy \tag{4.8}$$

4.2 HERLEITUNG DES STENCILS

Da die Berechnung des Datenterms bekannt ist und u und v symmetrisch sind, wird im folgenden nur die Herleitung des Stencils von u beschrieben. Im Gegensatz zum bisherigen Verfahren wird die Gleichung zuerst diskretisiert und dann abgeleitet. Dadurch können die Randbedingungen gleich berücksichtigt werden und müssen nicht am Schluss berechnet werden. Nach der Diskretisierung und Näherung für die 2.Ableitung ergibt sich für u:

$$\sum_{i=bx+1}^{bx+nx-2} \sum_{j=by+1}^{by+ny-2} \beta\Psi_{i,j}\left(\left(\frac{u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)}}{h_x^2}\right)^2 + 2\left(\frac{u_{(i+1,j+1)} - u_{(i-1,j+1)} - u_{(i+1,j-1)} + u_{(i-1,j-1)}}{4h_x h_y}\right)^2 + \left(\frac{u_{(i,j+1)} - 2u_{(i,j)} + u_{(i,j-1)}}{h_y^2}\right)^2\right) \tag{4.9}$$

Um diese Summe zu minimieren, muss der Term nun nach jedem Pixel abgeleitet werden. Dabei merken wir uns für später, in welchen Termen ein Pixel vorkommt.

Dies wird beispielhaft für den ersten der drei Summanden durchgerechnet.
Wir wollen nun

$$\sum_{i=bx+1}^{bx+nx-2} \sum_{j=by+1}^{by+ny-2} \beta\Psi_{i,j} \left(\frac{u_{(i+1,j)} - 2u_{(i,j)} + u_{(i-1,j)}}{h_x^2} \right)^2 \tag{4.10}$$

nach einem Pixel P(k,l) ableiten. P hängt aber nur von u(i,j) ab, wenn gilt:

$$(i = k-1 \vee \tag{4.11}$$
$$i = k \vee \tag{4.12}$$
$$i = k+1) \tag{4.13}$$
$$\wedge j = l \tag{4.14}$$

Betrachten wir zunächst den Fall 4.11. Substituieren wir nun in 4.10 i mit (k-1), ergibt sich:

$$\beta\Psi_{k-1,l} \left(\left(\frac{u_{(k-1+1,l)} - 2u_{(k-1,l)} + u_{(k-1-1,l)}}{h_x^2} \right)^2 \right) \tag{4.15}$$

Abgeleitet sieht der Term dann so aus:

$$\beta\Psi'_{k-1,l} \left(\frac{u_{(k,l)} - 2u_{(k-1,l)} + u_{(k-2,l)}}{h_x^2} \right) \frac{2}{h_x^2} =: ① \tag{4.16}$$

Dabei wird auf folgende Werte von u zugegriffen: $u_{k,j}, u_{k-1,l}, u_{k-2,l}$. Das führt zu folgenden Randbedingungen:

$$(bx + 2 \leqslant k \leqslant bx + nx - 1) \wedge (by \leqslant l \leqslant by + ny - 1) \tag{4.17}$$

Außerdem bedeutet das, dass in diesem Fall für das Pixel P[k,l] genau die Werte u[k-2,l] bis u[k,l] für die Berechnung benötigt werden. Wir konstruieren uns nun eine Schablone (Stencil) für die Näherung zweiter Ordnung. Wir benötigen dafür eine 5x5 Matrix. Nach dem ersten Schritt sieht diese nun so aus:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ ① & -2 * ① & ① & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{4.18}$$

Zu beachten ist dabei, dass wenn Bedingung 4.17 nicht gilt, alle 3 Werte aus der Matrix fallen.
Auf die gleiche Art und Weise kann man sich für die anderen zwei Fälle sowie für den

zweiten und dritten Summand aus 4.9 die insgesamt zehn Bedingungen und Faktoren herleiten. Diese sind:

$$① := \beta\Psi'_{k-1,l}\left(u_{(k,l)} - 2u_{(k-1,l)} + u_{(k-2,l)}\right) * \frac{2}{h_x^4} \tag{4.19}$$

$$(bx + 2 \leqslant k \leqslant bx + nx - 1) \wedge (by \leqslant l \leqslant by + ny - 1) \tag{4.20}$$

$$② := \beta\Psi'_{k,l}\left(u_{(k+1,l)} - 2u_{(k,l)} + u_{(k-1,l)}\right) * \frac{-4}{h_x^4} \tag{4.21}$$

$$(bx + 1 \leqslant k \leqslant bx + nx - 2) \wedge (by \leqslant l \leqslant by + ny - 1) \tag{4.22}$$

$$③ := \beta\Psi'_{k+1,l}\left(u_{(k+2,l)} - 2u_{(k+1,l)} + u_{(k,l)}\right) * \frac{2}{h_x^4} \tag{4.23}$$

$$(bx \leqslant k \leqslant bx + nx - 3) \wedge (by \leqslant l \leqslant by + ny - 1) \tag{4.24}$$

$$④ := \beta\Psi_{k-1,l-1}\left(u_{(k,l)} - u_{(k-2,l)} - u_{(k,l-2)} + u_{(k-2,l-2)}\right)\left(\frac{1}{4h_x^2h_y^2}\right) \tag{4.25}$$

$$(bx + 2 \leqslant k \leqslant bx + nx - 1)\,(by + 2 \leqslant l \leqslant by + ny - 1) \tag{4.26}$$

$$⑤ := \beta\Psi_{k+1,l-1}\left(u_{k+2,l} - u_{k,l} - u_{k+2,l-2} + u_{k,l-2}\right)\left(\frac{-1}{4h_x^2h_y^2}\right) \tag{4.27}$$

$$(bx \leqslant k \leqslant bx + nx - 3)\,(by + 2 \leqslant l \leqslant by + ny - 1) \tag{4.28}$$

$$⑥ := \beta\Psi_{k-1,l+1}\left(u_{k,l+2} - u_{k-2,l+2} - u_{k,l} + u_{k-2,l}\right)\left(\frac{-1}{4h_x^2h_y^2}\right) \tag{4.29}$$

$$(bx + 2 \leqslant k \leqslant bx + nx - 1)\,(by \leqslant l \leqslant by + ny - 3) \tag{4.30}$$

$$⑦ := \beta\Psi_{k+1,l+1}\left(u_{k+2,l+2} - u_{k,l+2} - u_{k+2,l} + u_{k,l}\right)\left(\frac{1}{4h_x^2h_y^2}\right) \tag{4.31}$$

$$(bx \leqslant k \leqslant bx + nx - 3)\,(by \leqslant l \leqslant by + ny - 3) \tag{4.32}$$

$$⑧ := \beta\Psi'_{k,l}\left(u_{(k,l)} - 2u_{(k,l-1)} + u_{(k,l-2)}\right) * \frac{2}{h_x^4} \tag{4.33}$$

$$(bx \leqslant k \leqslant bx + nx - 1) \wedge (by + 2 \leqslant l \leqslant by + ny - 1) \tag{4.34}$$

$$⑨ := \beta\Psi'_{k,l}\left(u_{(k,l+1)} - 2u_{(k,l)} + u_{(k,l-1)}\right) * \frac{-4}{h_x^4} \tag{4.35}$$

$$(bx \leqslant k \leqslant bx + nx - 1) \wedge (by + 1 \leqslant l \leqslant by + ny - 2) \tag{4.36}$$

$$⑩ := \beta\Psi'_{k,l}\left(u_{(k,l+2)} - 2u_{(k,l+1)} + u_{(k,l)}\right) * \frac{2}{h_x^4} \tag{4.37}$$

$$(bx \leqslant k \leqslant bx + nx - 1) \wedge (by \leqslant l \leqslant by + ny - 3)$$

Richtig in die Matrix-Schablone eingesetzt, sieht diese so aus:

$$\left(\begin{array}{c|c|c|c|c} ④ & 0 & -④- ⑤+ ⑧ & 0 & ⑤ \\ \hline 0 & 0 & -2\ ⑧\ -⑨ & 0 & 0 \\ \hline ①-④-⑥ & -2①-② & \begin{array}{c}① - 2② +③ + \\ ④+⑤+⑥+⑦ \\ +⑧ +2⑨+⑩\end{array} & -②-2③ & ③ - ⑤-⑦ \\ \hline 0 & 0 & -⑨ -2⑩ & 0 & 0 \\ \hline ⑥ & 0 & -⑥-⑦+⑩ & 0 & ⑦ \end{array}\right) \tag{4.38}$$

Diese Matrix wird nun für jeden Pixel des zu berechnenden Bildes erzeugt, wobei jedes mal die Bedingungen neu überprüft werden müssen.

5. EVALUATION

In diesem Kapitel sollen die beiden Verfahren, also Regularisierer erste Ordnung und zweite Ordnung, miteinander verglichen werden. Es werden Beispiele gezeigt, in denen das eine oder das andere Verfahren besser geeignet ist. Außerdem soll eine Kombination aus beiden Verfahren getestet werden, bei der die jeweiligen Nachbarschaften mit unterschiedlichen Gewichten addiert werden. Als Testsequenzen werden nun jeweils vier ausgewählte Bildpaare aus dem Middlebury Benchmark und der KITTI Vision Banchmark Suite verwendet. Für Middlebury wurden die Testsequenzen Grove2, Grove 3, Urban 2 und Urban 3 verwendet und dabei jeweils die Bilder 10 und 11. Sie sind Beispiele, bei denen sich die Kamera in einer virtuellen Szene etwas nach rechts oder links bewegt. Für KITTI sind es die Sequenzen 11, 15, 44 und 74, auch hier wurden die Bilder 10 und 11 verwendet. Diese Bilder sind durch eine Frontkamera eines Fahrenden Autos entstanden. Zu erwarten ist, dass bei Middlelbury die erste Ordnung und bei KITTI die zweite Ordnung besser ist.
Um die berechneten Flussfelder zu visualisieren, wird aus ihnen ein Bild berechnet, in dem Richtung und Länge des Verschiebungsvektors codiert sind. Der Farbton gibt die Richtung und die Helligkeit gibt die Länge der Verschiebung an, siehe Abbildung 2.1.

5.1 ERGEBNISSE MIT REGULARISIERUNG ERSTER ORDNUNG

Der Algorithmus benötigt eine Reihe von Parametern, die einen Einfluss auf die Qualität der Ergebnisse haben, abhängig von der Art des Bildes. Folgende Parameter können verändert werden:

- α: Die Gewichtung zwischen Datenterm und Glattheitsterm, siehe Formel 2.4
- η: Die relative Größe des Bildes beim nächsten Warplevel, siehe Kapitel 3.2
- σ: Die Standartabweichung des Gaußfilters, siehe Kapitel 2.5
- ϵ: Der Parameter für den Subquadratischen Datenterm, siehe Kapitel3.3
- ω: Der Parameter für das SOR Verfahren, siehe Kapitel 3.4
- Warplevel: Gibt die maximale Anzahl an Warpleveln an. Hier wurde das Warplevel so gewählt, dass das kleinste Bild mindestens die Größe vier mal vier Pixel hat.
- Inner Iteration: Anzahl der Iterationen mit dem SOR-Verfahren, bevor Ψ aktualisiert wird, siehe Kapitel 3.3

Abbildung 5.1: Die Originalbilder der KITTI-Sequenzen. Von oben nach unten: Sequenz 11, 15, 44, 74

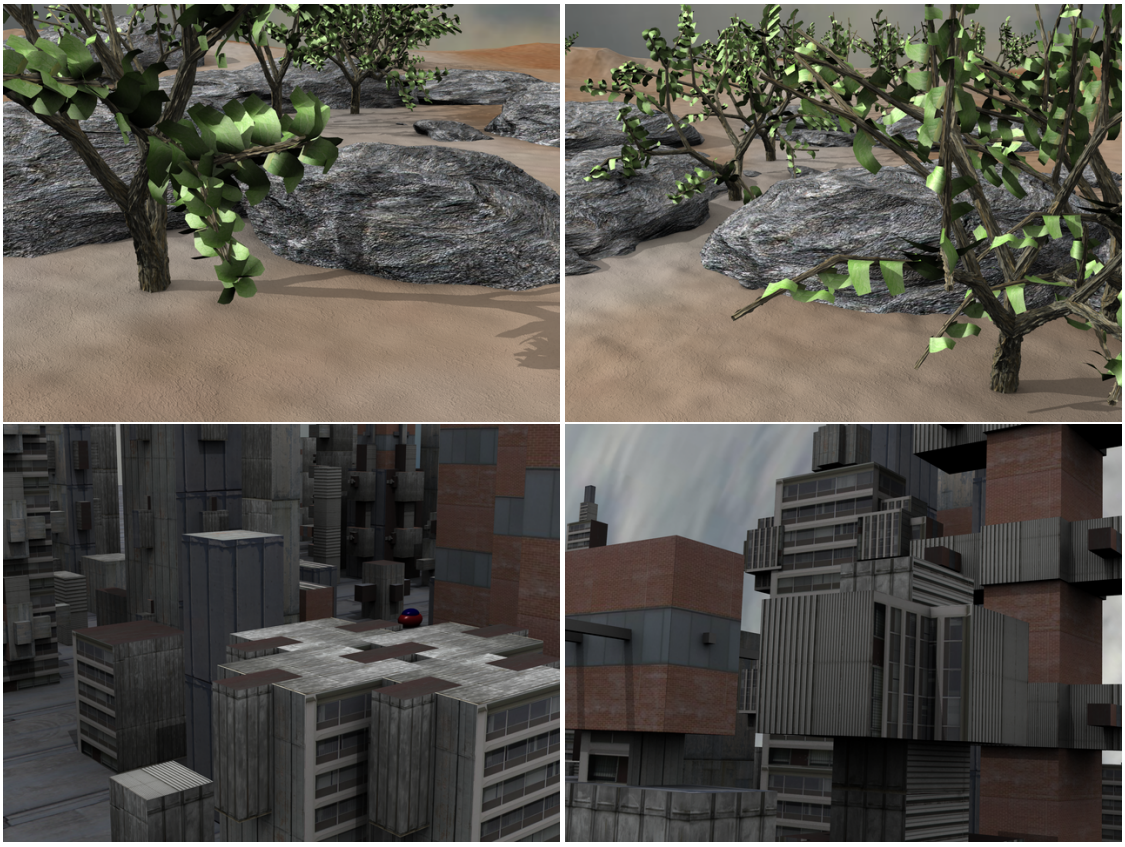

Abbildung 5.2: Die Originalbilder der Middlerbury-Sequenzen. Links oben: Grove2. Rechts oben: Grove3. Links unten: Urban2. Rechts unten: Urban3

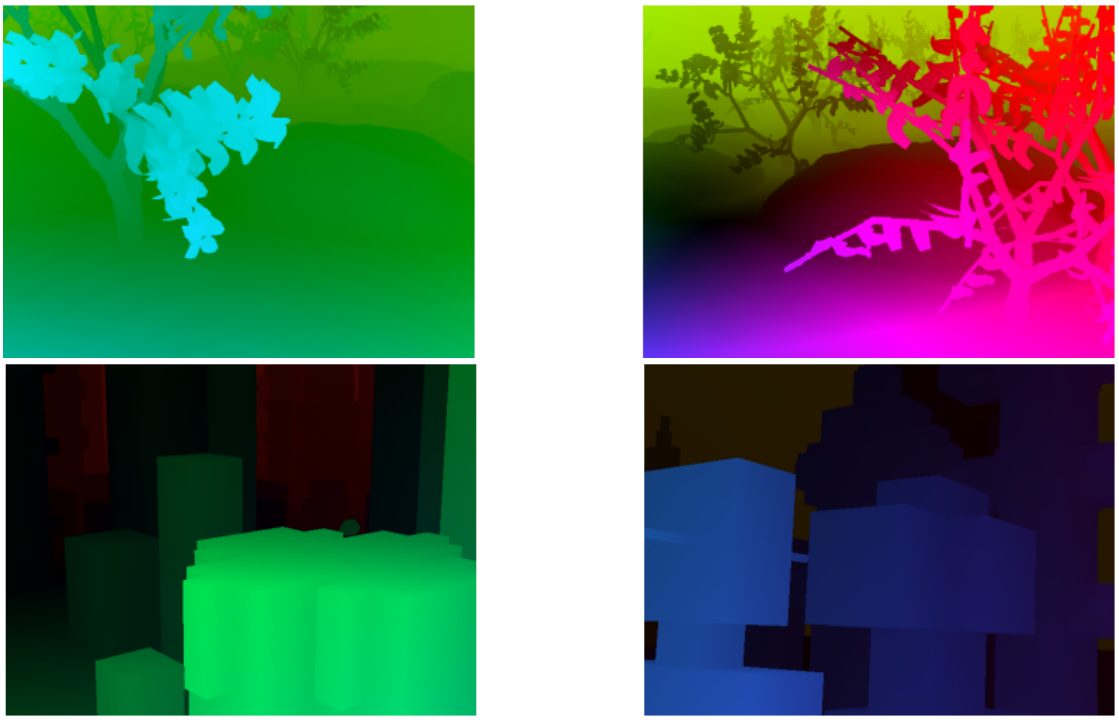

Abbildung 5.3: Ground Truth der Middlebury Sequenzen. Links oben: Grove 2. Rechts oben: Grove 3. Links unten: Urban 2. Rechts unten: Urban 3. Beachte: die Bilder Urban 2und 3 sind im Vergleich zu Grove 2 und 3 dunkler, da der optische Fluss größere Werte hat.

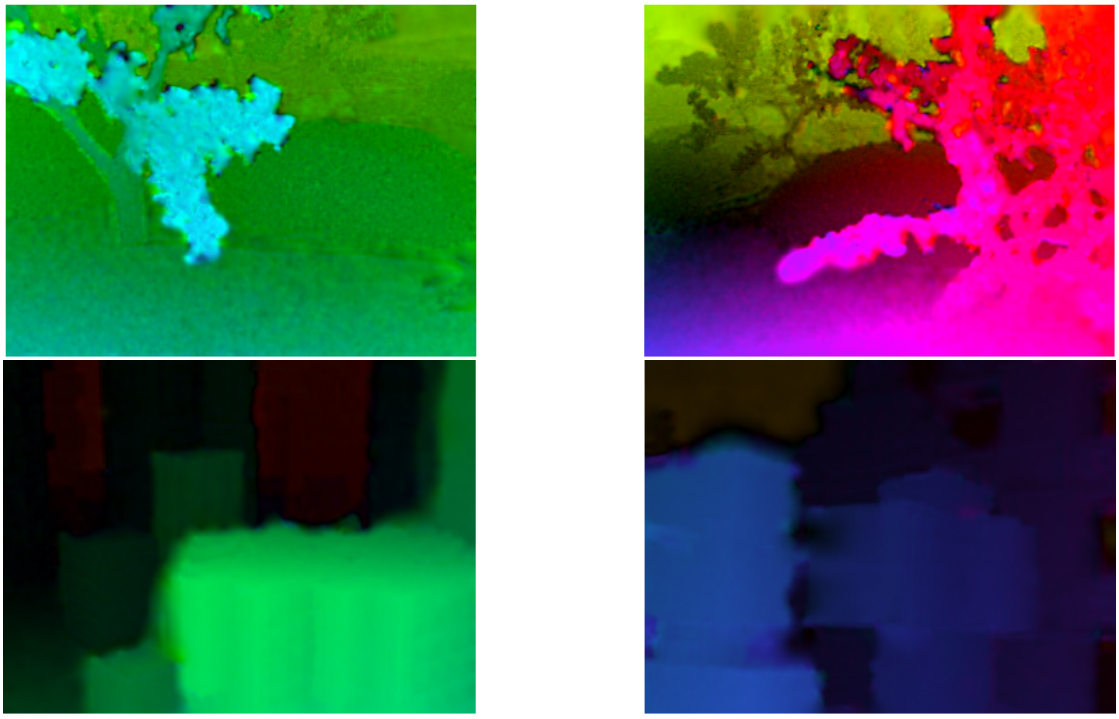

Abbildung 5.4: Die berechneten Ergebnisse bei folgenden Parametereinstellungen: $\alpha = 3{,}63352$, $\eta = 0{,}85$, $\sigma = 1$, $\epsilon = 0{,}1$, $\omega = 1{,}97$, inner Iteration = 5, outer Iterations = 10. Die AEE sind: 0,2967 (Grove 2), 0,8698 (Grove 3), 0,6296 (Urban 2), 0,9637 (Urban 3). Durchschnitt: 0,6899375

- Outer Itaration: Anzahl der Schritte, wie oft Ψ aktualisiert wird, siehe Kapitel 3.3

Um möglichst gute Ergebnisse zu erzielen beziehungsweise um möglichst gute Parametereinstellungen zu ermitteln, wurden Schätzwerte verwendet und anschließend ein Parameter so lange verändert, bis ein möglichst guter Wert ermittelt wurde (im Vergleich mit der Ground Truth). Betrachtet wurde dabei jeweils der Mittelwert für die vier Bilder aus einer Serie (KITTI oder Middlelbury). Danach wurden die anderen Parameter ebenfalls optimiert, bis ein möglichst gutes Gesamtergebnis erreicht wurde.

Es zeigt sich schnell, dass die Ergebnisse beim Middlebury Test deutlich besser ausfallen als bei KITTI. Das liegt unter anderem an der Tatsache, dass die Middlebury Testbilder digital erstellt wurden und es sich bei KITTI um Bilder, die aus einem fahrenden Auto heraus aufgenommen wurden, handelt. Dazu kommt, dass gerade bei KITTI die Annahme, der optische Fluss sei (lokal) konstant, was bei der Regularisierung erster Ordnung angenommen wird, nicht erfüllt ist.

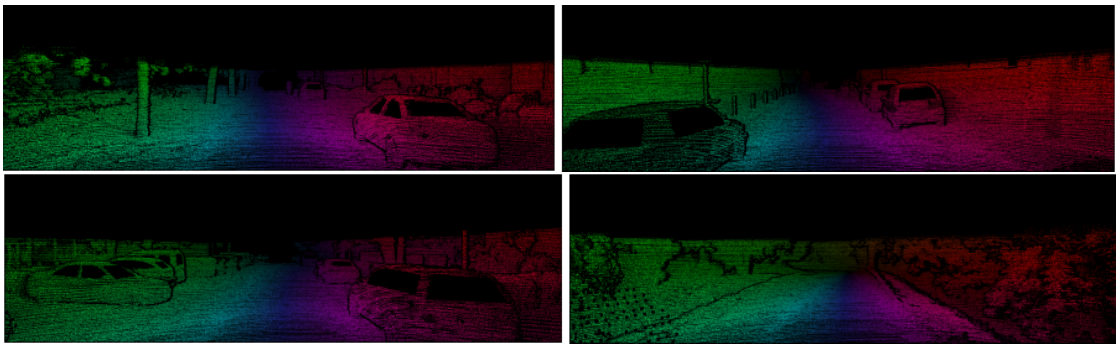

Abbildung 5.5: Die Groundtruth der KITTI Serien, von links nach rechts und von oben nach unten: Die Serien 11, 15, 44,74. Die Groundtruth ist nur an Kanten definiert, Flächen werden daher schwarz angezeigt. Aufgrund der großen Unterschiede im Betrag des optischen Flusses (Verschiebung um einige wenige Pixel in der Mitte bis ungefähr 100 Pixel am Rand) wurde bei allen KITTI Bildern eine logarithmische Skalierung der Helligkeit gewählt.

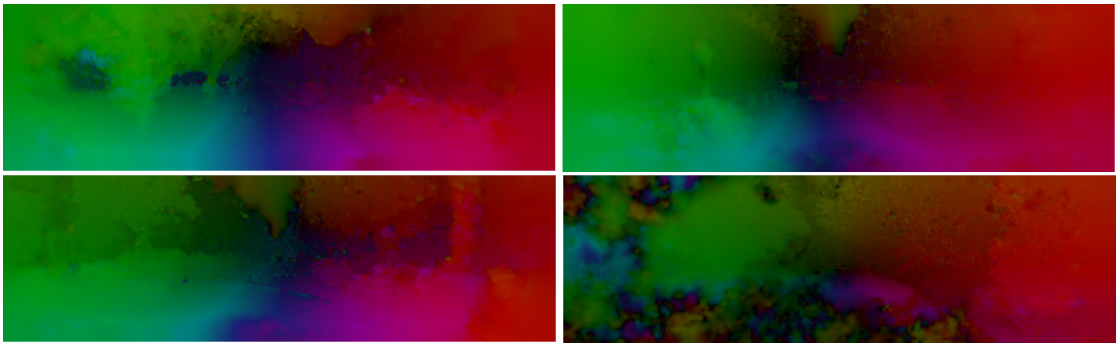

Abbildung 5.6: Die Berechneten Bilder mit den Parametern:$\alpha = 1$, $\eta = 0{,}85$, $\sigma = 1$, $\epsilon = 0{,}1$, $\omega = 1{,}97$, inner Iteration = 5, outer Iterations = 10. Die AEE sind: 8,378(Sequenz 11, oben links), 8,2545(Sequenz 15, oben rechts), 6,3997(Sequenz 44, unten links), 24,108(Sequenz 74, unten rechts). Durchschnitt: 11,78505

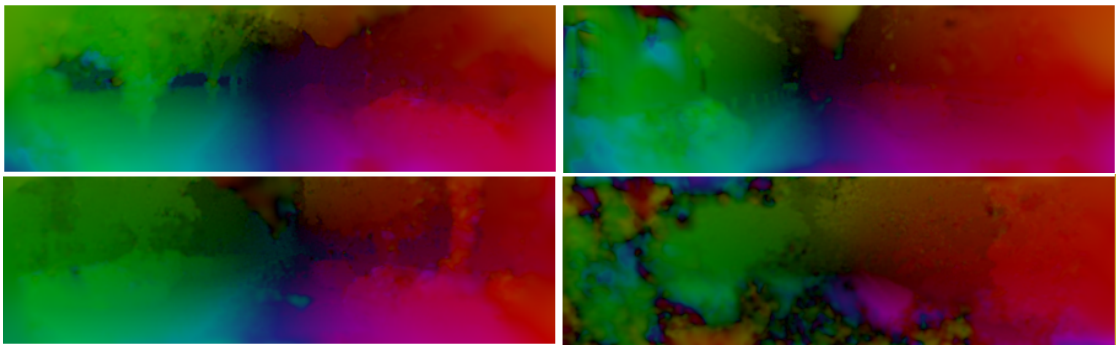

Abbildung 5.7: Das Ergebnis der Berechnung der optischen Flusses mit einem Regularisierer zweiter Ordnung. Beta = 2, η = 0,87, σ = 1, ϵ = 0,1, ω = 1,97, Warplevel = 80, inner Iterations = 5, outer Iterations = 10. Die AEE sind 8,469 (Sequenz 11, oben links), 8,552 (Sequenz 15, oben rechts), 4,425(Sequenz 44, unten links), 22,69(Sequenz 74, unten rechts). Der durchschnittliche AEE ist 11,12825

5.2 ERGEBNISSE MIT REGULARISIERUNG ZWEITER ORDNUNG

Die einstellbaren Parameter bei einem Regularisierer zweiter Ordnung sind die gleichen wie bei dem der ersten Ordnung. Allerdings ersetzten wir dasα nun durch ein β, das ebenfalls eine Gewichtung zwischen Glattheitsterm und Datenterm angibt, allerdings soll für die spätere Anwendung, wenn beide Regularisierer kombiniert werden, eine Unterscheidung zwischen den beiden Ordnungen und ihrer jeweiligen Gewichtung möglich sein. Das Finden von guten Parametern wurde wie oben über das Variieren einzelner Parameter durchgeführt. Bei beiden Serien sieht man, dass es eine relativ genau zu berechnende Sequenz gibt (Urban2 beziehungsweise Sequenz 44) und eine Sequenz mit sehr schlechtem Ergebnis (Urban3 und Sequenz 74). Die beiden anderen liegen jeweils mit ähnlichem AEE in der Mitte. Das gilt sowohl für Regularisierer erster als auch zweiter Ordnung.
Man sieht, dass bei KITTI der Regularisierer zweiter Ordnung bessere Ergebnisse liefert und bei Middlebury der Regularisierer erster Ordnung und damit die Erwartungen vom Anfang des Kapitels erfüllt werden. Allerdings sind die Unterschiede in Relation zu den absoluten Ergebnissen bei KITTI minimal.

5.3 KOMBINATION DER BEIDEN REGULARISIERER

Als nächster Schritt werden die beiden Möglichkeiten kombiniert. Es gibt nun sowohl den Parameterα als auch β. Es werden nun für beide Ordnungen eine Nachbarschaft berechnet, diese mit dem entsprechendem Parameter multipliziert und anschließend addiert. Die Hoffnung ist dabei, die beiden Stärken der einzelnen Verfahren zu kombinieren und noch bessere Ergebnisse zu erzeugen.

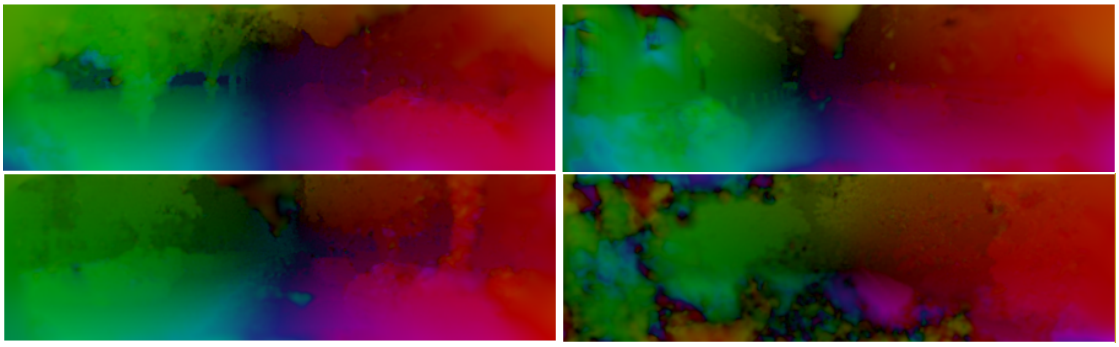

Abbildung 5.8: Erneuter Test mit zweiter Ordnung. Die Parameter sind wie in 5.7, außer σ = 0,75. Die Ergebnisse der drei besten Testsequenzen ist etwas besser, das Ergebnis der Sequenz 77 ist schlechter: Die AEE sind 8,453 (Sequenz 11, oben links), 8,531 (Sequenz 15, oben rechts), 4,383(Sequenz 44, unten links), 22,783(Sequenz 74, unten rechts). Der durchschnittliche AEE ist 11,0375

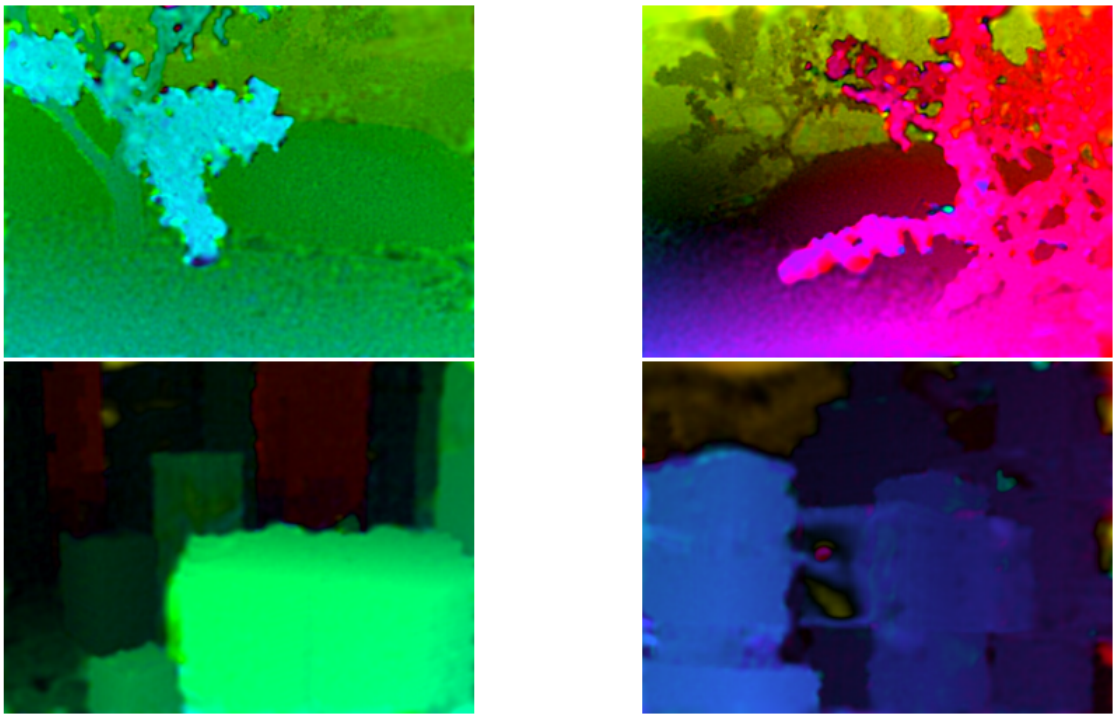

Abbildung 5.9: Die vier Ergebnisse der Middlerbury Testreihe mit den Parametern β = 3, η = 0.9, σ = 1, ω = 1,97, inner Iterations = 5, outer Iterations = 10. Die AEE sind 0,316 (Grove2, oben links), 0,918 (Grove3, oben rechts), 0,633 (Urban2, unten links), 1,233 (Urban3, unten rechts), Durchschnitt: 0,775 Auf drei der Bilder sind deutliche Artefakte zu erkennen, die meistens ein Zeichen von overfitting sind.

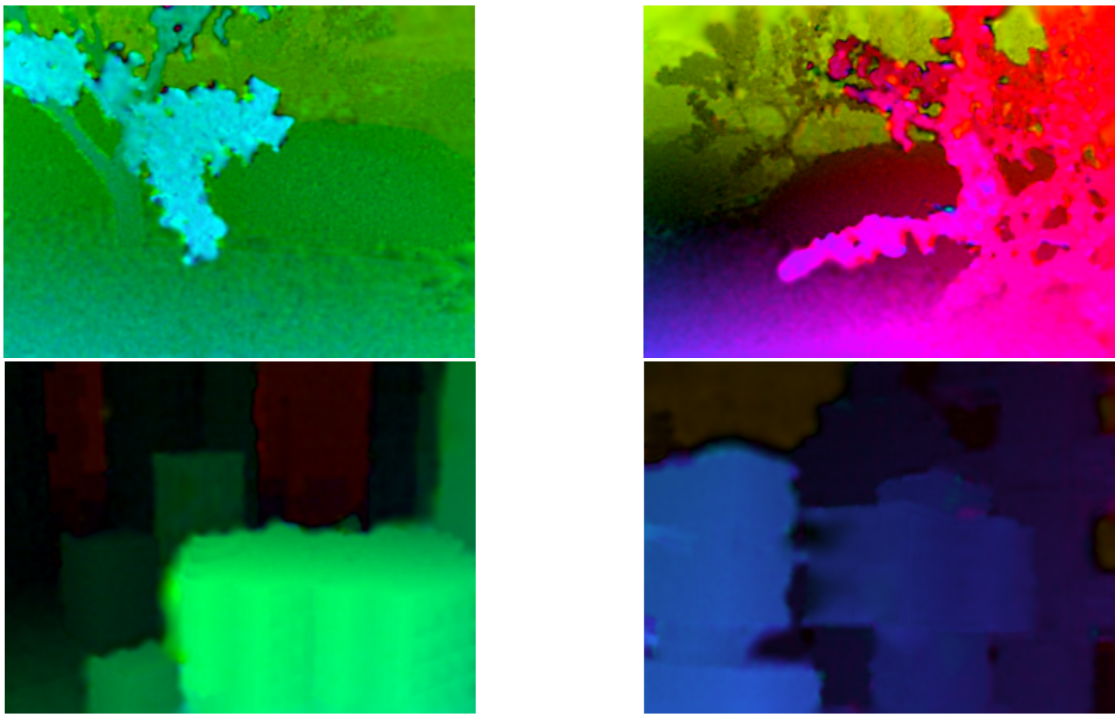

Abbildung 5.10: Die vier Ergebnisse der Middlerbury Testreihe mit kombinierten Regularisierern mit den Parameternα = 4, β = 1,5, η = 0.95, σ = 1, ω = 1,97, inner Iterations = 5, outer Iterations = 10. Die AEE sind 0,294 (Grove2, oben links), 0,839 (Grove3, oben rechts), 0,58 (Urban2, unten links), 0,834 (Urban3, unten rechts), Durchschnitt: 0,775. Die Artefakte sind nun nicht mehr erkennbar.

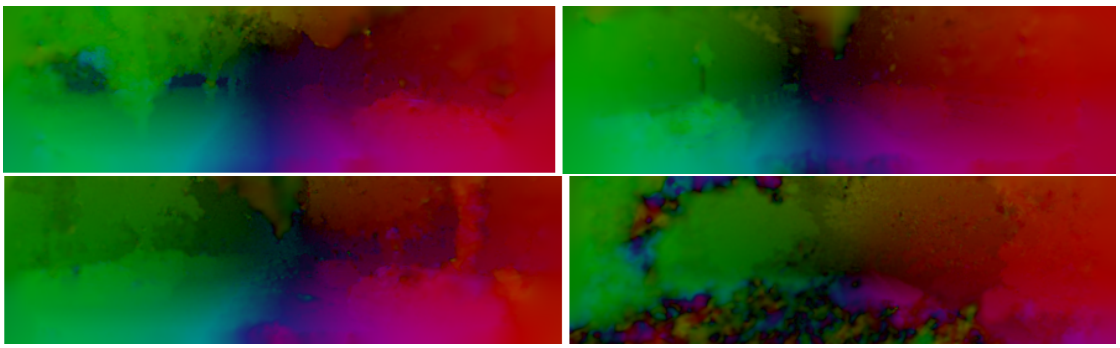

Abbildung 5.11: Berechnung des optischen Flusses mit Kombination beider Ordnungen und den Parameternα = 0,3, β = 0,6, η = 0.85, σ = 1, ω = 1,97, inner Iterations = 5, outer Iterations = 10. Die AEE sind 6,8652 (Sequenz 11, oben links), 6,1724 (Sequenz 15, oben rechts), 4,3778 (Sequenz 44, unten links), 22,228 (Sequenz 74, unten rechts). Der durchschnittliche AEE ist 9,91085

5.4 AUTOMATISCHE ERKENNUNG DER ORDNUNG

Bisher wurden die beiden Parameter α und β manuell vom Benutzer eingestellt. Es wäre allerdings interessant, ein Verfahren zu entwickeln, das automatisch, das heißt nur anhand der Eingabedaten erkennt, welches Verfahren besser geeignet ist und die Gewichtung der beiden Parameter vornimmt. Ein Ansatz dazu wäre, man berechnet das Flussfeld zunächst wie bisher, und untersucht anschließend, welche der beiden Bedingungen, also lokal konstantes oder lokal lineares Flussfeld, besser zutrifft. Es wurde im Rahmen dieser Arbeit folgendes Verfahren implementiert: Zunächst wird das Flussfeld mit einem Regularisierer zweiter Ordnung berechnet. Danach wurde über jedes Pixel iteriert, der Mittelwert der vier Nachbarn und des zentralen Pixel berechnet und anschließend die Differenz der fünf Pixel zum Mittelwert. Die Abweichung aller Pixel wird gemittelt. Liegt dieser Mittelwert unter einer gewissen Grenze, kann man davon ausgehen, dass die Annahme des lokal konstantem Flussfeldes gerechtfertigt ist. Ist das der Fall, kann die Berechnung erneut durchgeführt werden, diesmal mit einem Regularisierer erster Ordnung. Dabei ist die Wahl der Grenze entscheidend.
Man kann leicht erkennen, dass dieses Verfahren im besten Fall, das heißt wenn der richtige Regularisierer gewählt wird, das Ergebnis des besseren der beiden Annahmen liefert. Die Ergebnisse einer Kombination können jedoch nicht übertroffen werden. Dazu kommt, dass die Threshhold nicht allgemein angegeben werden kann. Betrachtet man zum Beispiel Pixel in der Nähe einer Kante, wird die Abweichung zum Mittelwert höher sein als in einer gleichmäßigen Fläche. Das bedeutet, dass auf Bildern mit vielen Kanten der Grenzwert höher eingestellt werden muss als in Bildern mit vielen flachen Regionen.
Eine Möglichkeit, dieses Problem zu beheben, ist die Gewichtung der einzelnen Differenzen mit ψ_s, siehe Kapitel 3.3. ψ_s soll den Einfluss des Smoothnessterms an Kanten verringern. Dadurch fallen für das Mittel aus allen Pixeln hauptsächlich die Pixel in flachen Regionen ins Gewicht, bei denen auch der Glattheitsterm eine große Rolle spielt.
Bei Tests zeigte sich jedoch, dass die berechnete mittlere Abweichung stark vom gewählten Glattheitsparameter β abhängt: Je kleiner β, desto größer ist die Abweichung. Das liegt daran, dass bei kleinerem β die Lösung weniger glatt ist und daher die Abweichungen größer sind. Das führt wiederum dazu, dass sich der Grenzwert, wann erste und wann zweite Ordnung gewählt werden soll, noch schwieriger zu finden und allgemein zu bestimmen ist.
Da man bei dieser Methode ebenso wie bei der Kombination aus dem vorangegangenen Unterkapiteln zwei Parameter hat, nämlich β und den Grenzwert beziehungsweise α, β, aber weniger verschiedene Einstellmöglichkeiten, wurde diese Art der Verbesserung nicht weiter untersucht. Die besten Ergebnisse sind gleich der besten Ergebnisse mit Regularisierer erster oder zweiter Ordnung.

6. ZUSAMMENFASSUNG UND AUSBLICK

Nach umfassenden Test und Analysen sollen die Ergebnisse in diesem letzten Kapitel zusammengefasst werden. Insbesondere soll geklärt werden, ob die im Vorfeld getroffenen Vermutungen erfüllt wurden oder nicht.

6.1 ZUSAMMENFASSUNG

Zu Beginn der Arbeit wurde der Algorithmus von Horn und Schunck implementiert und um einige Konzepte erweitert. Getestet wurde dieses Verfahren mit der Yosemite Sequece with Clouds. Anschließend wurde das Verfahren um einen Regularisierer zweiter Ordnung erweitert und mithilfe der Middlebury Benchmark und der KITTI Vision Banchmark Suite getestet. Die Test wurden sowohl mit erster und zweiter Ordnung durchgeführt als auch mit einer Kombination aus beiden Möglichkeiten. Dabei zeigte sich, dass bei den Middlebury Sequenzen die Kombination aus beiden Verfahren die besten Ergebnisse lieferte (bei hohem Faktor erste Ordnung), gefolgt von nur erster Ordnung. Test mit nur zweiter Ordnung lieferten die schlechtesten Ergebnisse. Bei den KITTI Serien war ebenfalls die Kombination am besten, gefolgt von den Durchläufen mit nur zweiter Ordnung. Damit wurde die These, bei Bildern mit einer Frontalbewegung der Kamera ist ein Regularisierer zweiter Ordnung besser und bei Parallelbewegungen der Regularisierer erster Ordnung, bestätigt. Allerdings sind die Unterschiede kleiner als erwartet. Ebenfalls erfüllt wurde die Vermutung, dass bei einer Kombination beider Verfahren die Ergebnisse noch besser werden, vor allem, wenn der bessere geeignete Term stärker gewichtet wird.

6.2 AUSBLICK

Es gibt eine Reihe anderer Verfahren, die deutlich bessere Ergebnisse liefern als der Algorithmus, der in dieser Arbeit verwendet wird. Viele davon nutzen weitere Informationen, wie Stereokameras, mehr als zwei Bilder aus einer Serie oder ähnliches. Zum einen kann man daher das Basisverfahren noch weiter verbessern. Um die Berechnung des Optischen Flusses anhand der verschiedenen Ordnungen noch weiter zu verbessern, könnte eine lokale Methode verwendet werden, die an einzelnen Punkten oder in einzelnen Regionen eine Gewichtung für alpha und beta bestimmt. So ein Verfahren könnte sich lokal auf die verschiedenen Bedingungen des Bildes einstellen.

LITERATUR

[1] B. K. Horn und B. G. Schunck, "Determining optical flow", *Artificial Intelligence*, Bd. 17, Nr. 1–3, S. 185 –203, 1981, ISSN: 0004-3702. DOI: http://dx.doi.org/10.1016/0004-3702(81)90024-2.

[2] T. Brox, A. Bruhn, N. Papenberg und J. Weickert, "High accuracy optical flow estimation based on a theory for warping", English, in *Computer Vision - ECCV 2004*, Ser. Lecture Notes in Computer Science, T. Pajdla und J. Matas, Hrsg., Bd. 3024, Springer Berlin Heidelberg, 2004, S. 25–36. DOI: 10.1007/978-3-540-24673-2_3.

[3] M. J. Black und P. Anandan, "The robust estimation of multiple motions: parametric and piecewise-smooth flow fields", *Computer Vision and Image Understanding*, Bd. 63, Nr. 1, S. 75 –104, 1996. DOI: http://dx.doi.org/10.1006/cviu.1996.0006.

[4] J. Weicker und C. Schnörr, "A theoretical framework for convex regularizers in pde-based computation of image motion", 2000.

[5] D. Young, "Iterative methods for solving partial difference equations of elliptic type", *Transactions of the American Mathematical Society*, S. 92–111, 1954. Adresse: http://www.jstor.org/stable/1990745.

[6] I. M. Gelfand und S. V. Fomin, *Calculus of Variations*, R. A. Silverman, Hrsg. 2000.

[7] O. Demetz, M. Stoll, S. Volz, J. Weickert und A. Bruhn, "Learning brightness transfer functions for the joint recovery of illumination changes and optical flow", 2014.

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

[9] A. Bruhn, *Vorlesung correspondence problems in computer vision*, Universtität Stuttgart, Deutschland, 2014. Adresse: http://www.vis.uni-stuttgart.de/nc/lehre/details/typ/vorlesung/2049/158.html.

[10] M. J. Black und P. Anandan, "A framework for the robust estimation of optical flow", in *Computer Vision, 1993. Proceedings., Fourth International Conference on*, 1993, S. 231–236. DOI: 10.1109/ICCV.1993.378214.

[11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black und R. Szeliski, "A database and evaluation methodology for optical flow", 2009.

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

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

A. ANHANG

A.1 ERGEBNISTABELLEN

Im folgenden sind die Tabellen aufgelistet, welche für die Auswertung verwendet wurden.

Abb.	**Verfahren**	**Parameter**						**Ergebnisse: AEE**				
Nr.	Ordnung	α	β	η	σ	ϵ	ω	1	2	3	4	durchschnitt
5.4	1.	3,63	-	0,85	1	0,1	1,97	0,297	0,870	0,630	0,964	0,690
5.9	2.	-	3	0,9	1	0,1	1,97	0,316	0,918	0,633	1,233	0,775
5.10	kombi	4	1,5	0,95	1	0,1	1,97	0,294	0,839	0,580	0,834	0,63675
5.6	1.	1	-	0,85	1	0,1	1,97	8,378	8,255	6,400	24,108	11,78505
5.7	2.	-	2	0,87	1	0,1	1,97	8,469	8,552	4,425	22,69	11,034
5.8	2.	-	2	0,87	0,75	0,1	1,97	8,453	8,531	4,383	22,783	11,0375
5.11	kombi	0,3	0,6	0,85	1	0,1	1,97	6,865	6,172	4,378	22,228	9,911

Tabelle A.1: Testszenario 1. Vergleich der drei entwickelten Verfahren

DANKSAGUNG

An dieser Stelle möchte ich mich ganz herzlich bei meinem Professor und Betreuern, Prof. Dr.-Ing. Andrés Bruhn, für die ausführliche Beratung und Hilfestellung bedanken. Er konnte mir immer schnell und kompetent behilflich sein, wenn ich nicht weiter wusste. Ebenfalls bedanken möchte ich mich bei meinem Kommilitonen Niklas Kaulitz, der eine ähnliche Arbeit geschrieben hat und mit dem ich zusammen die Grundlagen des Programmierteils erarbeitet habe.

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