Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3712

Fortgeschrittene Segmentierungs-basierte Verfahren zur Bestimmung des Optischen Flusses

Fabian Andres

Studiengang: Informatik

Prüfer/in: Prof. Dr.-Ing. Andrés Bruhn

Betreuer/in: Prof. Dr.-Ing. Andrés Bruhn

Beginn am: 17. Februar 2015

Beendet am: 19. August 2015

CR-Nummer: I.4.6, I.4.8, I.4.10

Kurzfassung

Die Berechnung des Optischen Flusses ist eines der zentralen Probleme der Computer-Vision. Hierbei wird ein Vektorfeld zwischen zwei aufeinanderfolgenden Bildern einer Bildersequenz bestimmt, das als die Bewegung der Objekte interpretiert werden kann. Häufig wird dieses Problem durch mathematische Modelle, sogenannte Variationsansätze, beschrieben. Diese setzen die Minimierung eines Kostenfunktionals voraus, das Abweichungen der aufgestellten Modellannahmen bestraft. Typischerweise wird dies durch eine pixelbasierte Variante durchgeführt. In den letzten Jahren hat sich jedoch die Integration von Segmentinformation als Erfolg versprechend gezeigt. Das Ziel dieser Arbeit ist es, die Berechnung des Optischen Flusses auf verschiedene segmentbasierte Verfahren umzustellen. Dabei werden die Pixel der Bilder durch eine Mean-Shift-Segmentierung zu inhaltlichen Objekten zusammengefasst, was zu einer besseren Abgrenzung der Objekte führt. Hierbei werden zwei verschiedene Ansätze verfolgt. Zum einen die Umstellung auf segmentbasierte Methoden, die die Bewegung ganzer Segmente durch eine konstante und affine Darstellung modellieren und so eine Reduzierung der Variablen ermöglichen. Zum anderen durch pixelbasierte Methoden, die durch eine zuvor berechnete Segmentierung an den Kanten unterstützt werden.

Abstract

The calculation of the optical flow is one of the central problems in computer vision. Here, a vector field between two consecutive images of an image sequence is to be found, which can be interpreted as the movement of objects. Often this problem is described by mathematical models, so-called variational methods. These require the minimization of a cost funtional which punishes deviations of the established model assumptions. Typically, this is performed by a pixel-based method. In recent years, however, the integration of segment information has shown promising results. The aim of this work is to perform the calculation of the optical flow in several segment-based methods. The pixels of the images are therefor combined by a mean shift segmentation, resulting in a better definition of the objects. Here, two different approaches were followed. Firstly, switching to segment-based methods that model the movement of entire segments by a constant and affine representation, thus allowing a reduction in variables. Secondly, by pixel-based methods that are supported by a previously computed segmentation at the edges.

Inhaltsverzeichnis

1 Einleitung

Die Berechnung des Optischen Flusses ist eines der zentralen Probleme der Computer Vision. Ziel dabei ist es, bei einer gegebenen Bildsequenz, ein Vektorfeld zu bestimmen, das alle Punkte des ersten Bildes in die neue Position des darauffolgenden Bildes überführt. Dieses Vektorfeld kann als die Bewegung der Objekte interpretiert werden. Das Konzept des Optischer Fluss wurde bereits im Jahr 1950 von Gibson eingeführt um das Verhalten der Tiere durch visuelle Stimuli zu beschreiben [Gib50]. Mit zunehmendem Fortschritt und neuen technologischen Möglichkeiten fand der Optische Fluss Einzug in das maschinelle Sehen und leistet heute einen wichtigen Beitrag bei der Interaktion von Maschinen mit ihrer Umgebung. Die Berechnung des Optischen Flusses kann für eine Vielzahl an visuellen Aufgaben eingesetzt werden. Dabei kann er für die Lösung des Korrespondenzproblems oder als Informationsquelle in komplexe Anwendungen integriert werden. Einen wichtigen Beitrag leistet der Optische Fluss beispielsweise in der Erkennung von beweglichen und stationären Objekten. In der Automobilindustrie kann so durch eine elektronische Überwachung ein sichereres Umfeld aller Beteiligten geschaffen werden. Hierbei werden bewegliche Objekte wie Passanten oder andere Fahrzeuge vom Hintergrund getrennt und ihre Bewegung im Verhältnis zur eigenen Geschwindigkeit evaluiert. So kann in kritischen Situationen der Bremsvorgang ohne Handeln des Fahrers automatisch eingeleitet werden.

Ein weiteres Anwendungsgebiet, indem der Optische Fluss zum Einsatz kommt, ist die Robotik. Er dient dabei als wichtige Informationsquelle für die Navigation der Maschinen.

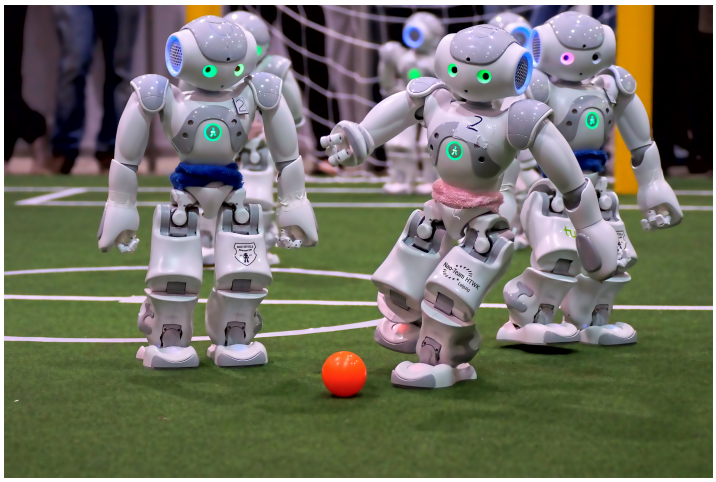


Abbildung 1.1: Fußball spielende Roboter auf der CeBIT 2013 [NT].

In Abbildung 1.1 sind Fußball spielende Roboter zu sehen. Um viele Spielzüge durchführen zu können, muss der Roboter ein Verständnis seiner Umgebung, der Mitspieler und der Gegner haben. So kann mithilfe des Optischen Flusses ein Angriff des Gegners jedoch auch ein offen stehendes Tor oder ein freier Mitspieler erkannt und eine entsprechende Maßnahme eingeleitet werden.

Ein weiteres Gebiet bildet die Komprimierung von Videos. Hierbei werden nur in einem bestimmten Intervall Bilder abgespeichert und die dazwischenliegenden Bilder durch Unterschiede der Schlüsselbilder kodiert, was zu einer Verringerung des benötigten Speichers führt. Ein ähnliches Prinzip verfolgt die Komprimierung von Videodaten, bei denen nur Bilder abgespeichert werden in denen Bewegung vorhanden ist. So können in einem stundenlangen Überwachungsvideo nur die Szenen abgespeichert werden, in denen sich ein bewegliches Objekt vor der Kamera befindet.

Typischerweise wird bei der Berechnung jeder Punkt durch einen Pixel repräsentiert und durch Lösung des Korrespondenzproblems die Bewegung ermittelt. Eine der größten Herausforderungen spielen dabei die Kanten der Bildobjekte, da an diesen Stellen häufig ein Wechsel des Flusses durch unterschiedliche Bewegung entsteht. Um die Berechnung dieser Stellen zu verbessern, wurde der Fokus in den letzten Jahren zunehmend auf eine Kombination des Optischen Flusses mit segmentierten Bilden gelegt. Durch die Zusammenfassung der Pixel zu inhaltlichen Regionen kann so eine perfekte Erhaltung der Kanten erzielt werden.

1.1 Ziel der Arbeit

Das Ziel dieser Arbeit ist es, den Optischen Fluss und die Segmentierung der Bilder durch verschiedene Modelle miteinander zu verbinden und so ein besseres Ergebnis speziell in den Regionen der Kanten zu erhalten. Dabei wurden zwei verschiedene Methodiken bei der Modellierung betrachtet. Die erste Variante bestimmt den Fluss durch segmentweise Verschiebung der kompletten Objekte, wodurch der Fluss auf ein segmentbasiertes Modell umgestellt wird. Die zweite Variante behandelt den Fluss pixelbasiert, wobei die Segmentierung jedoch auf unterschiedliche Weise eingebracht wird, um die problematischen Stellen zu verbessern.

1.2 Verwandte Arbeiten

In diesem Abschnitt werden existierende Arbeiten vorgestellt, die eine ähnliche Vorgehensweise wie die aktuelle Arbeit haben. Da es eine Vielzahl an Arbeiten zur Berechnung des Optischen Flusses gibt, werden hier nur Arbeiten vorgestellt, die den Fluss mithilfe einer Segmentierung bestimmen. Dabei wird mit der Idee angefangen die Bilder zu segmentieren und den Fluss durch ein parametrisches Modell zu berechnen. Eine der ersten Arbeiten, welche die Segmentierung mit in den Optischen Fluss brachte, waren Ju *et al.* [JBJ96], die die Bilder in rechteckige Segmente aufteilen und diese durch ein affines Modell berechnen. Black und Jepson [BJ96] hingegen segmentieren die Bilder zunächst und berechnen für jedes Segment ein affines Modell unterschiedlicher Ordnung, da nicht jedes Segment durch ein komplexes Modell berechnet werden muss. Xu *et al.* [XCJ08] berechnen ebenfalls eine Segmentierung, bei der jedes Segment durch ein affines Modell dargestellt wird, das anschließend

global durch eine Konfidenzmap korrigiert wird um starre Bewegung der Segmente zu beheben. Eine weitere Methode ist Segmentkandidaten zu bestimmen und diese anschließend zuzuweisen. Vogel *et al.* [VSR13] weisen jedem Pixel ein Segment zu und jedem Segment eine dreidimensionale Ebene. Für die Regularisierung an den Kanten wird der Abstand der Ebenen an den Segmentgrenzen bestraft. Chen *et al.* [CJL+13] formulieren das Problem als Bewegungsproblem von Segmenten und verwenden eine Nearest-Neighbour-Approximationen um ein initiales Bewegungsfeld zu berechnen. Anschließend wird ein Set aus Ähnlichkeitsdeformationen für die Segmentkandidaten erstellt. Cremers und Soatto [CS05] segmentieren das Bild und berechnen den Fluss der Segmente durch kontinuierliche Optimierung der Grenzlänge der Segmente. Brox *et al.* [BBW06] verwenden eine kombinierte Berechnung des Optischen Flusses und der Segmentierung und optimieren die Segmente anhand von Level-Set-Funktionen. Unger *et al.* [UWPB12] berechnen den optischen Fluss durch eine 2D affine Parametrisierung und Okklusionsterm. Eine weitere Methode basiert auf einem geschichtetem Modell der Segmente. Sun *et al.* [SSB12] segmentieren die Bilder und benutzen eine affine Bewegung um das Bild in Schichten zu unterteilen und so den Fluss für jede Ebene zu regulieren.

Ein nah verwandtes Problem des Optischen Flusses ist die Berechnung der Tiefe aus einer statischen Szene. Hierfür müssen die Bildpunkte aus beiden Ansichten verbunden werden, um die Szene dreidimensional zu rekonstruieren. Eine Vorarbeit zu dieser Arbeit befasste sich mit der dreidimensionalen Rekonstruktion der Szene, bei der die Bilder segmentiert und anschließend durch eine konstante sowie affine Modellierung berechnet wurden [And14]. Ben-Ari und Sochen verwenden ein Mumford-Shah Variationsmodell mit einem Okklusions- und Stetigkeitsterm [BAS10].

1.3 Aufbau der Arbeit

Im folgenden Abschnitt wird der Aufbau dieser Arbeit erklärt. Zuerst erfolgt eine Einführung und Erklärung der benötigten mathematischen Operatoren und Verfahren, um ein Variationsmodell zur Berechnung des Optischen Flusses aufzustellen (Kapitel 2). Danach wird das Basisverfahren aufgebaut und ausführlich Schritt für Schritt hergeleitet (Kapitel 3). Im Anschluss erfolgt die Herleitung der segmentbasierten Verfahren mit einer Einführung in die Segmentierung (Kapitel 4). Darauf werden die pixelbasierten Verfahren behandelt, die durch verschiedene Varianten der Segmentierung unterstützt werden (Kapitel 5). Um die Qualität der Methoden zu evaluieren, werden danach Experimente anhand verschiedener Bildsequenzen durchgeführt und auf ihre Stärken und Schwächen überprüft (Kapitel 6). Im letzten Kapitel folgt eine Zusammenfassung der Arbeit und ein Ausblick über mögliche Erweiterungen, sowie ein abschließendes Fazit (Kapitel 7).

2 Grundlagen

Für die Berechnung des Optischen Flusses sind eine Vielzahl an verschiedenen Operatoren und Verfahren notwendig, welche sich über mehrere Teilgebiete der Mathematik erstrecken. In dem folgenden Kapitel werden deshalb wichtige Operatoren und grundlegende Funktionen sowie Eigenschaften erläutert, die zum Verständnis der späteren Vorgehensweise beitragen. Hinzu kommt ein Einblick in den Aufbau und der Extraktion der benötigten Daten, genauer gesagt der strukturelle Aufbau von digitalen Bildern und den verschiedenen Farbräumen, die in diesem Verfahren ihre Anwendung finden.

2.1 Mathematisches Grundwissen

Im folgenden Abschnitt werden einige Operatoren und Funktionen anhand von Beispielen erklärt und ihre jeweilige Funktion in der Berechnung des Optischen Flusses gezeigt. Ein Großteil befindet sich im Bereich der Analysis, welche sich mit der Untersuchung von reellen oder komplexen Funktionen hinsichtlich der Stetigkeit oder den Grenzwerten beschäftigt. Ein Hauptaugenmerk ist jedoch die Betrachtung der Differenzier- sowie Integrierbarkeit. Hinzu kommt die lineare Algebra, in welcher die Lösung linearer Gleichungsysteme sowie lineare Abbildungen im Mittelpunkt stehen.

2.1.1 Skalarfeld

Bei dem Skalarfeld handelt es sich um eine Funktion f, welche von $\mathbb{R}^n \to \mathbb{R}$ abbildet. Es entsteht somit ein Feld, welches jedem Punkt im Raum $f(x_1...x_n)$ ein Skalar zuordnet. Häufige Verwendung findet es beispielsweise in der Darstellung von Höhenunterschieden einer Landkarte oder von Temperaturunterschieden.

2.1.2 Vektorfeld

Im Gegensatz zu einem Skalarfeld bildet ein Vektorfeld von $\mathbb{R}^n \to \mathbb{R}^m$ ab. Hier wird jedem Punkt im Raum ein Vektor zugeordnet. Das Vektorfeld wird häufig für die Darstellung von Magnetfeldern verwendet, in welchem jeder Punkt des Magnetfelds in eine bestimmte Richtung und Stärke wirkt. Auch der Optische Fluss wird typischerweise durch ein Vektorfeld dargestellt, indem die Richtung und Länge der Verschiebung durch einen Vektor dargestellt wird. Veranschaulicht wird dies in Abbildung 2.1.

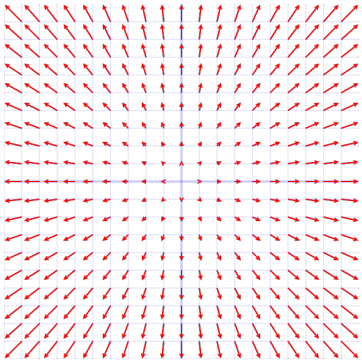


Abbildung 2.1: Plot eines möglichen Verschiebungsvektorfelds, in dem sich auf die Mitte des Bildes zubewegt wird (Zoom-In).

2.1.3 Differentialrechnung

Die Differentialrechnung ist zusammen mit der Integralrechnung eine der wichtigsten Bereiche der Analysis. Sie beschreibt die lokale sowie globale Veränderung von Funktionen. Unterschieden wird hier zwischen dem Differenzenquotient und Differentialquotient.

Differenzenquotient

$$\varphi(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{2.1}$$

Der Differenzenquotient wird oft verwendet, um die lokale Ableitung einer Funktion zu bestimmen. Hierzu dienen x_0 und x_1 als Eingrenzung des zu berechnenden Bereichs, welcher oft auch als Schrittlänge oder Schrittweite bezeichnet und positiv gewählt wird. $f(x_0)$ und $f(x_1)$ beschreiben die Werte, welche die Funktion f durch Einsetzen der Punkte an den jeweiligen Stellen annimmt. Durch den Quotient der beiden Differenzen erhält man die lokale Steigung der Funktion f im Intervall $[x_0; x_1]$.

Differentialquotient

$$\lim_{x \to 0} \frac{f(x_0 + h) - f(x_0)}{h} \tag{2.2}$$

Der Differentialquotient hingegen beschreibt die Ableitung der Funktion f in jedem Punkt. Hierzu wird die Schrittweite nach $h = x_1 - x_0$ umgestellt und eingesetzt. Eine Funktion ist im Punkt x_0 differenzierbar, wenn der Grenzwert $h \to 0$ existiert, die Schrittweite demzufolge unendlich klein ist. Um die Ableitung einer Funktion zu bestimmen, werden die Ketten-, Produkt- sowie die Quotientenregel verwendet. Häufige Schreibweisen des Differentialquotienten, der von nun an so abgekürzt wird, sind $f'(x)$, f_x oder $\frac{df}{dx}$, wobei immer nach der Variable x abgeleitet wird. Ein Beispiel für beide Varianten der Differenzierung ist in Abbildung 2.2 zu sehen.

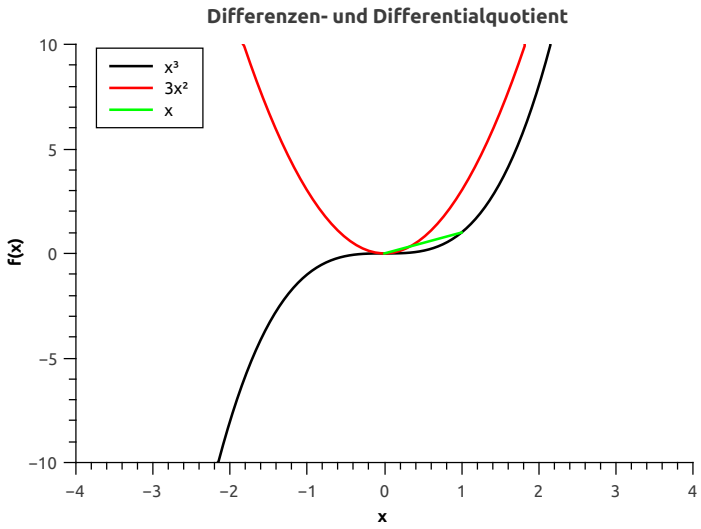


Abbildung 2.2: Plot der Funktion x^3 (Schwarz) mit der ersten Ableitung (Rot) und der lokalen Ableitung (Grün) zwischen dem Intervall 0 und 1.

Nabla-Operator

Der Nabla-Operator ∇ kann als Vektor angesehen werden, der für die Beschreibung von Differentialoperatoren verwendet wird, zum Beispiel dem Gradienten, der Divergenz oder der Rotation. Da all diese Operatoren die Ableitung einer Funktion gemein haben, bestehen die Komponenten des Nabla-Operators aus den partiellen Ableitungen der jeweiligen Funktion:

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix} \qquad \text{oder} \qquad \nabla^T = \left(\frac{\partial}{\partial x_1} \cdots \frac{\partial}{\partial x_n} \right). \tag{2.3}$$

Er kann als Spaltenvektor ∇ verwendet werden, wie es bei dem Gradienten der Fall ist oder als Zeilenvektor ∇^T, welcher für die Divergenz benötigt wird.

Gradient

Durch Anwendung des Nabla-Operators auf eine skalare Funktion f wird jedem Punkt im Raum $\mathbb{R}^n$ ein Vektor zugeordnet, welcher in die Richtung des stärksten Anstiegs zeigt. Der zugehörige Betrag des Vektors beschreibt den Grad der Steigung. Das entstandene Vektorfeld wird auch der Gradient von f genannt. Berechnet wird er durch die Multiplikation des Nabla-Operators mit der Funktion, auf welche er angewendet wird:

$$\text{grad}(f) = \nabla f(x_1...x_n) = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix} f(x_1...x_n) = \begin{pmatrix} \frac{\partial}{\partial x_1} f(x_1...x_n) \\ \vdots \\ \frac{\partial}{\partial x_n} f(x_1...x_n) \end{pmatrix} = \begin{pmatrix} f_{x_1} \\ \vdots \\ f_{x_n} \end{pmatrix}. \tag{2.4}$$

Divergenz

Wendet man den Nabla-Operator auf eine Vektorfunktion f an, wird jedem Punkt im Raum $\mathbb{R}^n$ ein Skalar zugeordnet, der beschreibt, wie stark benachbarte Vektoren sich voneinander wegbewegen (divergieren). Das entstandene Skalarfeld wird Divergenz genannt. Es gibt mehrere Möglichkeiten die Divergenz zu interpretieren. Stellt man sich das Vektorfeld als Strömung in einem geschlossenem System vor, beschreiben positive Werte, auch Quellen (sources) genannt, Orte bei denen neue Flüssigkeit einströmt. Negative Werte dagegen stellen Senken (sinks) dar, bei denen Flüssigkeit entweicht. Berechnet wird die Divergenz durch das Skalarprodukt des Nabla-Operators und dem Vektorfeld f:

$$\text{div}(f) = \nabla^T \cdot f = \begin{pmatrix} \frac{\partial}{\partial x_1} & ... & \frac{\partial}{\partial x_n} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} = \frac{\partial}{\partial x_1} f_1 + ... + \frac{\partial}{\partial} f_n. \tag{2.5}$$

2.1.4 Integralrechnung

Das Integral ist mit der Differentialrechnung der wichtigste Bereich der Analysis. Wird das Differential gefolgt von dem Integral oder umgekehrt auf eine Funktion f angewendet, entsteht bei Vernachlässigung der Integrationskonstante wiederum die ursprüngliche Funktion f. Das Integral ist daher die inverse Operation zur Differentialrechnung. Bei der Integralrechnung wird zwischen dem bestimmten Integral und dem unbestimmten Integral unterschieden.

Bestimmtes Integral

$$\int_a^b f(x)dx \tag{2.6}$$

Das bestimmte Integral ordnet einer Funktion f einen Zahlenwert zu. Dieser Berechnet sich aus einem festgelegten Intervall in der Funktion, der zwischen a und b liegt und beschreibt die Fläche die zwischen der Funktion und der x-Achse eingeschlossen wird. Häufige Anwendung findet das bestimmte Integral in Extremwertproblemen, wo beispielsweise die kleinste Lösung eines Problems gesucht wird.

Unbestimmtes Integral

$$\int f(x)dx = F(x) + C \tag{2.7}$$

Das unbestimmte Integral ordnet einer Funktion f eine Menge an Funktionen zu, den sogenannten Stammfunktionen, welche mit F abgekürzt werden. Wie bereits angesprochen, bilden diese Funktionen wenn sie erneut differenziert werden, die ursprüngliche Funktion f. Da die Integration jedoch nicht eindeutig ist, da bei der Ableitung einer Stammfunktion die Integrationskonstante C wegfällt, existieren unendlich viele Stammfunktionen. Ein Hauptsatz des Integrals besagt, dass sich aus jeder Stammfunktion wiederum das bestimmte Integral berechnen lässt. In Abbildung 2.3 sind beide Integrale anhand eines Beispiels dargestellt.

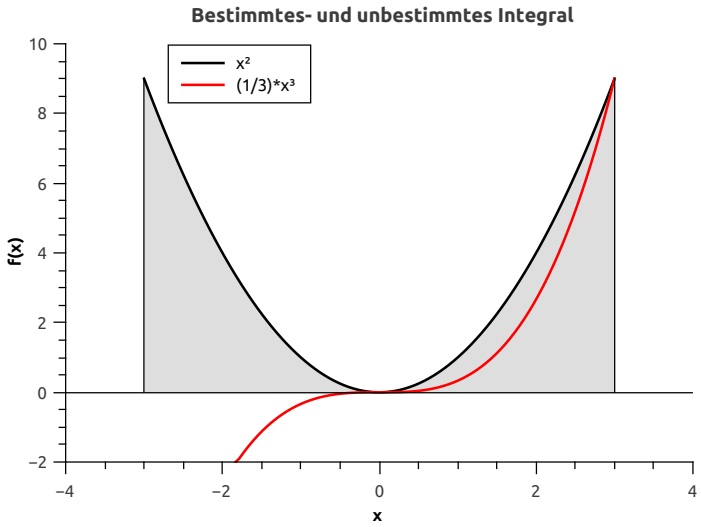


Abbildung 2.3: Plot der Funktion x^2 (Schwarz) mit der Fläche des bestimmten Integrals zwischen -3 und 3 (Grau) und einer Stammfunktion $\frac{1}{3}x^3$ (Rot) mit Integrationskonstante = 0.

2.1.5 Eigenwertproblem

Das Eigenwertproblem beschreibt wesentliche Eigenschaften eines linearen Gleichungssystem oder einer Matrix. Bestandteile sind Eigenvektoren und ihre zugehörigen Eigenwerte, welche unter Voraussetzung einer quadratischen Matrix der Form n $\times$ n wie folgt berechnet werden können:

$$Ax = \lambda x \qquad \text{oder} \qquad (A - \lambda E) \cdot x = 0 \tag{2.8}$$

Die Definition des Eigenvektors besagt: Existiert ein Vektor $x \neq 0$, welcher mit einer Matrix A multipliziert wird und obige Gleichung erfüllt, folglich seine Richtung beibehält und nur durch seine Länge verändert wird, spricht man von einem Eigenvektor. Die Länge λ ist als Eigenwert definiert

und gibt an, wie stark der Vektor gestreckt oder gestaucht wird. Ein Beispiel eines Eigenvektors ist in Abbildung 2.4 veranschaulicht.

Für die Berechnung der Eigenwerte muss die Determinante der Matrix $A - \lambda E$ berechnet werden, das so entstehende Polynom wird auch Charakteristisches Polynom genannt. Hiervon müssen nun die Nullstellen berechnet werden und man erhält die Eigenwerte λ_i. Für die zugehörigen Eigenvektoren x_i muss λ_i in die Matrix eingesetzt und das jeweils resultierende homogene Gleichungssystem gelöst werden.

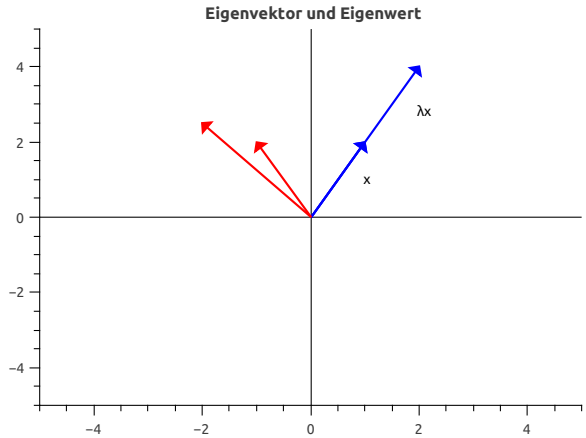


Abbildung 2.4: Beispiel eines Eigenvektors mit zugehörigem Eigenwert (blau). Die Richtung hat sich im Gegensatz zu dem Vektor in rot nicht geändert.

2.1.6 Spur

Die Spur einer Matrix ist definiert als die Summe der Hauptdiagonale einer n $\times$ n Matrix. Sie kann nur auf quadratische Matrizen angewendet werden und berechnet ein Skalar. Eine besondere Eigenschaft der Spur ist, dass sie der Summe der Eigenwerte entspricht:

$$\mathrm{Spur}(A) = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + ... + a_{nn} = \sum_{i=1}^{n} \lambda_i. \tag{2.9}$$

2.1.7 Laplace Operator

Der Laplace-Operator Δ ordnet jedem Punkt einer skalaren Funktion f im Raum $\mathbb{R}^n$ ein Skalar zu. Er wird berechnet durch die Divergenz des Gradienten von f und beschreibt die Quelldichte des Gradientenfelds. Ein mögliches Anwendungsgebiet ist die Beschreibung der Diffusion von Teilchen, kurzum der natürlichen Ausbreitung von Stoffen. In der Physik wird der Laplace-Operator auch oft für die Beschreibung von der Massedichte eines Gravitationspotentials verwendet. Er kann wie folgt dargestellt werden:

$$\Delta f = \mathrm{div}(\mathrm{grad}(f)) = \nabla^T \cdot (\nabla f) = \begin{pmatrix} \frac{\partial}{\partial x_1} & \dots & \frac{\partial}{\partial x_n} \end{pmatrix} \begin{pmatrix} \frac{\partial}{\partial x_1} f \\ \vdots \\ \frac{\partial}{\partial x_n} f \end{pmatrix} = \frac{\partial^2}{\partial^2 x_1} f + \dots + \frac{\partial^2}{\partial^2 x_n} f. \quad (2.10)$$

2.1.8 Konvergenz

Ein wichtiger Bereich der Analysis ist die Bestimmung des Grenzwertes einer Funktion oder Reihe. Grenzwerte werden heute für viele Beweisformen verwendet, beispielsweise für die Definitionen von Stetigkeit oder der Differential- und Integralrechnung. Betrachtet man den Limes (auch Limit) einer Funktion können 2 Fälle auftreten. Sollte eine Funktion gegen einen bestimmten Wert tendieren, welchem sie sich mit steigendem oder fallendem x immer weiter annähert, spricht man von einer konvergenten Funktion. Das genaue Gegenteil ist die Divergenz, wobei kein Grenzwert existiert.

2.1.9 Differentialgleichung

Eine Differentialgleichung (DLG) ist eine Gleichung, welche die Beziehungen zwischen einer Funktion und ihren Ableitungen beschreibt. Es wird somit eine Funktion gesucht, die diese Gleichung erfüllt. Sie wird oft in der Naturwissenschaft benutzt, um bestimmte Phänomene zu beschreiben. Dabei bezeichnet die Funktion eine physikalische Größe und die Ableitung ihre Veränderungsrate, die durch die Gleichung miteinander in Verbindung gebracht werden. Da Differentialgleichungen jedoch äußerst komplexe Formen annehmen können, existiert kein allgemeines Lösungsverfahren, sondern muss oft durch numerische Verfahren approximiert werden. Die allgemeine Form lautet:

$$F(x, y(x), y'(x), ..., y^n(x)) = 0 \quad (2.11)$$

wobei y^n für die n-te Ableitung steht.

2.1.10 Funktional

Ein Funktional beschreibt eine Funktion, die einen Vektor als Eingabeparameter bekommt und anschließend auf ein Skalar abbildet. Häufig wird jedoch ein Vektorraum übergeben, dessen Elemente selbst Funktionen sind und den gleichen Definitionsbereich besitzen. Dies wird als Funktionenraum bezeichnet. Ein Funktional ist somit eine Funktion, die weitere Funktionen als Argument enthalten kann:

$$G(f(x)) = \int_\Omega f(x) dx. \quad (2.12)$$

Ein Teilgebiet der Mathematik, die sich mit der Lösung von Funktionalen beschäftigt, ist die Variationsrechnung, die Mitte des 18. Jahrhunderts von Leonhard Euler und Joseph-Louis Lagrange entwickelt wurde.

2.1.11 Variationsrechnung

Die Variationsrechnung wurde entwickelt, um die Lösung eines Funktionals zu bestimmen. Bei der gewöhnlichen Differentialrechnung besteht ein wichtiger Bestandteil darin Extremwerte zu finden, folglich einen Punkt oder Punkte der x-Achse, bei welchem eine Funktion $f(x)$ auf einen Hoch-, Tief- oder Sattelpunkt abbildet. Die Variationsrechnung hingegen beschäftigt sich mit einem ähnlichen Prinzip, mit dem Unterschied, dass nun eine Funktion gesucht wird, für dessen Integral in Abhängigkeit der Funktion selbst und ihren Ableitungen ein Extremum annimmt. Die Aufgabe besteht somit darin, einen kompletten Funktionsverlauf zu finden, für den der Integralausdruck ein Extremum annimmt. Dies wird auch Extremal genannt. Die allgemeine Form lautet:

$$I[y] = \int_a^b F(x, y(x), y'(x)...y^n(x))dx. \tag{2.13}$$

Der Hauptsatz der Variationsrechnung beruht auf der Euler-Lagrange-Differentialgleichung. Dies ist die zentrale These, die zur Berechnung eines Funktionals verwendet wird. Ähnlich wie bei der Differenzialrechnung stellt sie eine notwendige Bedingung für das Annähern an ein Extremal dar, welche durch Veränderung im Umfeld der Lösung ermittelt wird.

2.1.12 Taylorreihe

Die Taylorreihe ist eine Approximierung einer Funktion f, die durch die unendliche Summe ihrer Ableitungen an einem bestimmten Punkt ausgewertet wird. Voraussetzung dafür ist allerdings die Stetigkeit der Funktion und die Existenz einer stetigen Ableitung bis einschließlich der Ableitung der höchsten verwendeten Ordnung. Üblicherweise wird die Taylorreihe nach einer bestimmten Anzahl an Termen abgebrochen, wenn der approximierte Bereich hinreichend angenähert ist. Die ursprüngliche Funktion kann so bereits mit wenigen Termen approximiert werden. Das so entstehende Polynom wird auch Taylorpolynom genannt:

$$T(f) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k. \tag{2.14}$$

Die Taylorpolynom wird so um den Punkt a und mit steigender Anzahl verwendeter Terme genauer an die Funktion angenähert. $f^{(k)}$ steht hier für die k-te Ableitung von f, $k!$ bezeichnet die Fakultät.

Linearisierung

Eine weitere Eigenschaft der Taylorreihe ist die Erzeugung von linearen Funktionen. Da nichtlineare Funktionen oder nichtlineare Differentialgleichungen eine hohe Komplexität haben und die Theorie von linearen Funktionen besser erforscht und umfangreicher ist, wird deshalb oft zur Vereinfachung des Problems ein lineares Polynom approximiert. Dies geschieht durch den frühzeitigen Abbruch der Taylorreihe, genauer gesagt, bereits nach dem zweiten Term:

$$L(f) = f(a) + \frac{\partial}{\partial x} f(a) \cdot (x - a). \tag{2.15}$$

Das so entstehende Polynom verläuft tangential durch den Entwicklungspunkt a. Zu beachten ist allerdings, dass bei sprunghaften Funktionen mit zunehmender Entfernung vom Entwicklungspunkt keine Genauigkeit der Werte garantiert werden kann und die Approximation nur bei glatten Funktionen einer guten Repräsentation entspricht. In Abbildung 2.5 wird die Taylorreihe und eine entsprechende Linearisierung anhand eines Beispiels erklärt.

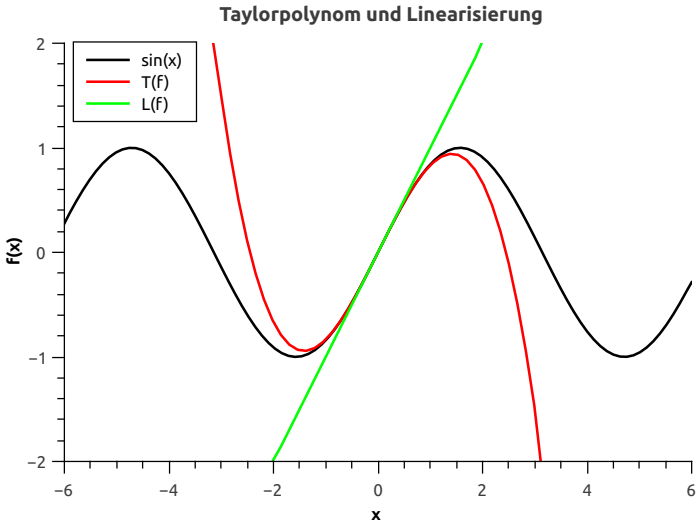
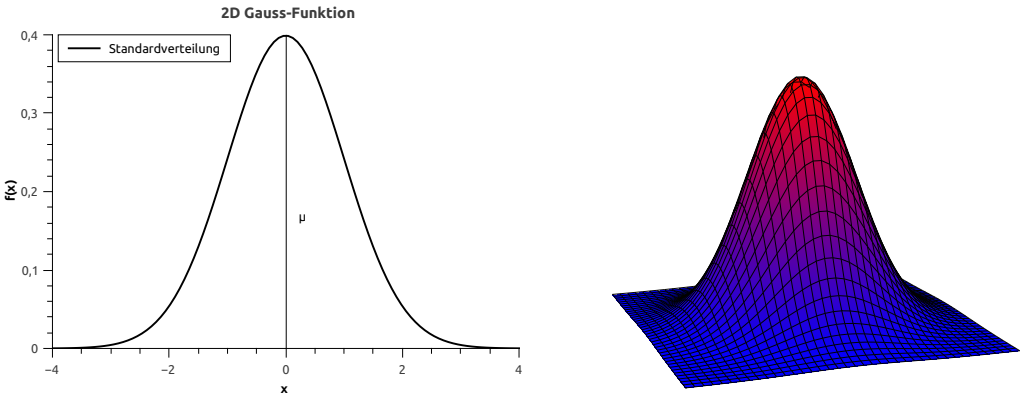


Abbildung 2.5: Plot der Funktion $\sin(x)$ (Schwarz) mit dem Taylorpolynom 3. Grades (Rot) und der Linearisierung (Grün) um den Entwicklungspunkt 0.

2.1.13 Gauß-Verteilung

Die Gauß-Verteilung, auch Normalverteilung, gehört zu den wichtigsten Funktionen der Wahrscheinlichkeitsrechnung. Sie wird oft in der Statistik oder der Beschreibung der Natur eingesetzt und drückt die stetige Verteilung von unabhängigen Zufallsvariablen aus. Der Parameter μ beschreibt dabei den Erwartungswert, an dessen Stelle der höchste Wert auftritt. Die Standardabweichung σ steht für Breite der Normalverteilung. Die Normalverteilung kann mit folgender Funktion beschrieben werden:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{+\infty} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx. \tag{2.16}$$



(a) 2-Dimensionale Gauß-Funktion

(b) 3-Dimensionale Gauß-Funktion

Abbildung 2.6: 2D und 3D Gauß-Funktion mit Erwartungswert $\mu = 0$ und Standardabweichung $\sigma^2 = 1$.

Dichtefunktion

Die Dichtefunktion der Gauß-Verteilung heißt Gauß-Funktion. Durch ihre Ähnlichkeit mit einer Glocke wird sie auch oft Glockenfunktion genannt und wird wie folgt beschrieben:

$$f(x, \mu, \sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{\det \sum}} e^{-\frac{1}{2}(x-\mu)^T \sum^{-1} (x-\mu)}. \tag{2.17}$$

Sie beschreibt die Wahrscheinlichkeit eines Erwartungswerts μ, welcher den wahrscheinlichsten Fall und den Hochpunkt darstellt, sowie der Abweichung von diesem Wert durch zufällige Einflüsse. Je größer diese Abweichung ist, desto unwahrscheinlicher ist das auftreten dieses Falls. $\sum$ steht für die $n \times n$ symmetrische Kovarianzmatrix, welche das Verhalten der Abweichung um den Erwartungswert beschreibt und wie folgt aussieht:

$$\sum = \begin{pmatrix} \sigma_{11} & \cdots & \sigma_{1n} \\ \vdots & & \vdots \\ \sigma_{n1} & \cdots & \sigma_{nn} \end{pmatrix}. \tag{2.18}$$

Mithilfe der Abweichung σ lässt sich zudem sehr leicht der Wertebereich festlegen, da der Intervall von -3σ bis $+3\sigma$ etwa 99.7% der Fläche einnimmt. Der Normierungsterm hat die Funktion des sicheren Ereignisses. Dies bedeutet, dass alle Wahrscheinlichkeiten addiert oder die Fläche unter der Funktion 1 ergeben müssen. In Abbildung 2.6 sind Beispiele für zwei- sowie den dreidimensionalen Fall der Funktion abgebildet.

Faltung

Mit der Faltung wird ein Operator beschrieben, welcher eine bestimmte Variante der Verknüpfung zweier Funktionen darstellt. Sie drückt das gewichtete Mittel einer Funktion f aus, welche mit einer weiteren Funktion g gewichtet wird und so die Funktion $(f * g)$ erzeugt. Die Berechnung der Faltung der Funktionen f und g erfolgt durch:

$$(f * g)(x) = \int_{\mathbb{R}^{\times}} f(\tau)g(x - \tau)d\tau. \tag{2.19}$$

Geometrisch gesehen werden dabei alle Punkte von f, welche sich mit g überlagern durch den Funktionswert von g gewichtet und der Mittelwert an dem Punkt τ ausgewertet. Eine häufig verwendete Gewichtung entspricht der zuvor behandelten Gauß-Funktion, was auch glätten genannt wird. Durch die Faltung von f mit dem Gaußkern werden spezielle Eigenschaften an die Funktion übertragen, wie die unendliche Differenzierbarkeit der neuen Funktion. Ein Beispiel der Faltung ist in Abbildung 2.7 zu sehen.

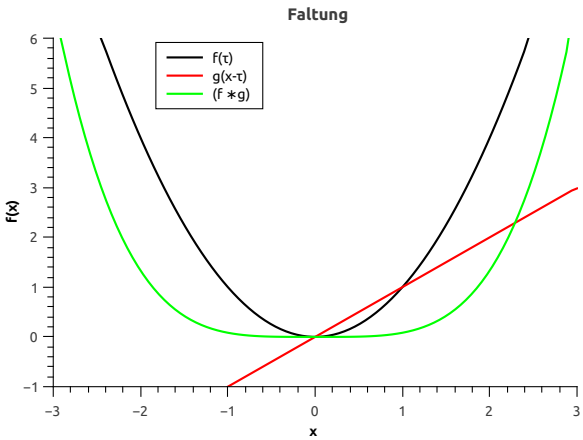


Abbildung 2.7: Plot der Faltung in dem Definitionsbereich -3 bis 3 von x^2 (Schwarz) mit der Gewichtsfunktion x (Rot) und der dadurch entstandenen Funktion $\frac{x^4}{12}$ (Grün).

2.2 Bilder

Für die Berechnung des Flusses sind allerlei Informationen notwendig, welche einzig aus den vorhandenen Bilder des Videos oder der Bilderfolge extrahiert werden müssen. Deshalb wird im folgenden Kapitel eine Beschreibung des Aufbaus und der Verwendung von Bildern erläutert.

2.2.1 Bildeigenschaften

Ein Bild wird durch verschiedene Parameter definiert. Zum einen durch die Form, die durch das Seitenverhältnis bestimmt wird. Zum anderen durch die Auflösung, welche oft als Indiz für die Qualität eines Bildes steht.

Seitenverhältnis

Die klassischen analogen sowie digitale Bilder bestehen aus einem Seitenverhältnis, das die Breite und Höhe des Bildes angibt. Ein häufig verwendetes Format war das $4:3$ Seitenverhältnis. Dies entspricht einer Breite, die 1.33 mal so lang ist wie Höhe. Dies hatte den Hintergrund, dass bereits existierende Sensoren von Monitoren benutzt wurden, welche zu dieser Zeit Standard waren. Heutzutage haben sich jedoch die $3:2$ und $16:9$ Formate durchgesetzt.

Auflösung

Die Auflösung eines Bildes beschreibt die Gesamtzahl an Bildpunkten einer Rastergrafik. Jedes digitale Foto hat eine bestimmte Anzahl an Bildpunkten in der Breite sowie in der Höhe. Multipliziert ergeben diese Werte die Auflösung des Bildes. Für die Darstellung ergeben sich somit 2 Varianten. Zum einen die Gesamtzahl an Punkten, die in der Fotografie oft in Megapixel angegeben wird, zum anderen als Multiplikation der Breite und Höhe, woraus sich leicht das Seitenverhältnis erkennen lässt. Eine Grundregel lautet, je größer die Auflösung eines Bildes ist, desto feiner und detailreicher kann es dargestellt werden.

Pixel

Die Bildpunkte eines Bildes, umgangssprachlich Pixel genannt (Kurzform für picture element), enthalten die farbliche Information und sind das kleinste veränderliche Element eines Bildes. Sie sind durch den Farbraum und die Farbtiefe definiert. Diese werden typischerweise durch die Kombination verschiedener Komponenten dargestellt, wie beispielsweise den Farbkanälen Rot, Grün und Blau. Durch die Aufteilung in sogenannte Subpixel, entsteht so durch Vermischung der Farbkanäle, der Eindruck einer bestimmten Farbe. Abbildung 2.8 zeigt eine schematische Darstellung der Pixel eines LCD-Displays.

2.2.2 Farbräume

Farben werden innerhalb eines Computers mittels Farbmodellen beschrieben. Jeder Farbe wird so eine Kombination der jeweiligen Parameter des entsprechenden Modells zugeordnet. Dies lässt sich dreidimensional in einem Farbraum darstellen, wobei jedem Punkt eine bestimmte Koordinate zugewiesen wird. Alle darstellbaren Farben lassen sich so durch einen geometrischen Körper anzeigen, dem sogenannten Farbkörper. In dieser Arbeit wurden in erster Linie zwei Farbmodelle verwendet,

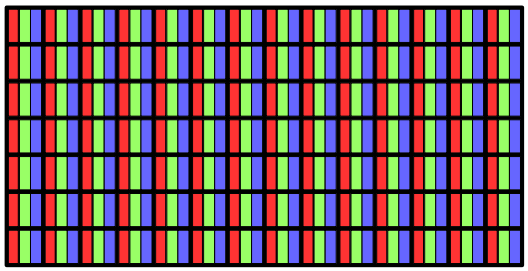


Abbildung 2.8: Schematischer Aufbau eines Bildes auf einem LCD-Display mit roten, grünen und blauen Subpixel.

welche jeweils ihre Vor- und Nachteile haben und in folgendem Abschnitt etwas näher erklärt werden.

RGB

Der RGB-Farbraum ist der bekannteste und am häufigsten verwendete Farbraum. Er besteht aus den drei Komponenten Rot, Grün und Blau, welche an das farbliche Sehen des Menschen angelehnt sind und dort für die verschiedenen Zapfen des Auges stehen. Jeder Farbwert eines Kanals wird mit 8 Bit dargestellt, was einer Unterteilung von 256 Farben pro Kanal entspricht. Durch additive Kombination der Kanäle lassen sich so bis zu 16.777.216 Farben erzeugen.

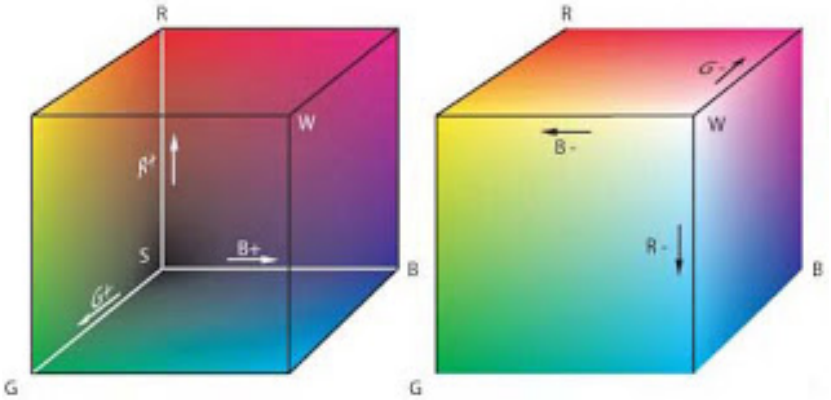


Abbildung 2.9: Darstellung des RGB-Farbkörpers [Fra] mit dem Fokus auf dunklere Farben (links) und helleren Farben (rechts).

Bei der Betrachtung von Abbildung 2.9 ist zu erkennen dass die Werte (0,0,0) Schwarz und (255,255,255) Weiß entsprechen. Alle anderen Farbwerte, bei welchen die drei Kanäle den selben Wert aufweisen, stellen verschiedenste Grautöne zwischen Schwarz und Weiß dar.

Lab

Der Lab-Farbraum beschreibt alle wahrnehmbaren Farben, somit sind mehr Farben darstellbar als mit RGB. Er besteht aus den Komponenten L, a und b, wobei L für die Lightness oder Helligkeit steht und Werte von 0 bis 100 annehmen kann. Schwarz wird mit 0 erzeugt und Weiß mit 100. Der a-Kanal beschreibt die Rot/Grün Komponente der Farbe und kann Werte zwischen -128 für Grün und $+127$ für Rot annehmen. Der b-Kanal hingegen stellt die Blau/gelb Komponente dar, mit -128 für Blau und $+127$ für Gelb.

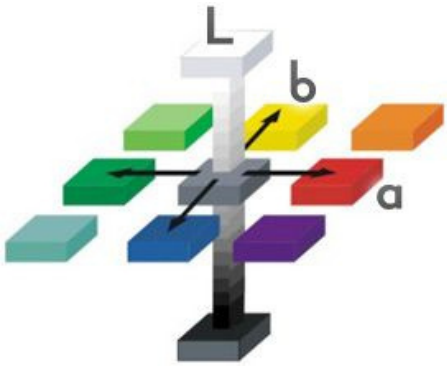


Abbildung 2.10: Darstellung des Lab-Farbkörpers [Ste].

Bei der Betrachtung von Abbildung 2.10 ist zu erkennen, dass durch die Festlegung von a und b auf die Werte 0 alleine durch die Veränderung des L-Kanals, alle Grautöne zwischen schwarz und weiß erzeugt werden können. Gleiches gilt bei allen anderen Farben. Somit können bestimmte Farben und ihre jeweiligen Abweichungen der Helligkeit gezielt einem Ort im Farbkörper zugeordnet werden. Dies ist insbesondere bei dem Vergleich ähnlicher Farben hilfreich.

2.2.3 Kontinuierliche Bilder

Die kontinuierliche Darstellung von Bildern am Computer ist im eigentlichen Sinne nicht möglich, da hierfür eine Funktion benötigt wird, welche in jedem Punkt definiert sein müsste. Ungeachtet dessen, helfen kontinuierliche Darstellungen von Bildern bei der späteren Modellierung und Umsetzung des Verfahrens. Unter anderem ermöglichen sie eine höhere Präzision, da sie eine Berechnung im Subpixel-Bereich zulassen. Sie können durch folgende Form dargestellt werden:

$$f(x, y). \tag{2.20}$$

Die Funktion f steht nun für die dreidimensionale Interpretation eines Bildes. Die x- und y-Achse steht für die Breite und Höhe eines Bildes und bildet auf den entsprechenden Farb- oder Grauwert ab.

2.2.4 Diskrete Bilder

Die diskrete Darstellung von Bildern wird typischerweise durch eine zeilenweise Konkatenation von Arrays realisiert. Hierzu werden die Bilder anhand der Auflösung und des Seitenverhältnisses in ein Gitter unterteilt, wobei jedem Eintrag der Farbwert an der entsprechende Stelle zugewiesen und wie folgt aufgeschrieben wird:

$$f = \begin{pmatrix} 200 & 150 & 100 \\ 200 & 150 & 100 \\ 100 & 100 & 100 \end{pmatrix} \tag{2.21}$$

Der zweidimensionale Array f zeigt einen Ausschnitt eines Bildes der Größe 3×3, die jeweiligen Einträge beschreiben den Grauwert oder einen bestimmten Farbkanal eines Pixels.

3 Basisverfahren

Für die meisten Lebewesen sind visuelle Wahrnehmungen ein wichtiger Bestandteil der Interaktion mit der Umgebung. So wird von dem Menschen annähernd 80 % der Informationen durch optische Reize aufgenommen und verarbeitet. Die Bewegung von Objekten spielt dabei eine besondere Rolle. Das Gehirn verarbeitet bestimmte Bewegungen und reagiert instinktiv auf die Situation. So werden bestimmte Reflexe wie beispielsweise der Schutz des Kopfes aktiviert, welcher durch ein schnell näher kommendes Objekt in Richtung der Person ausgelöst wird. Das Ziel des Optischen Flusses ist, dieses Verhalten auf den Computer zu übertragen und ihm so die Interaktion mit der Umgebung durch visuelle Reize zu ermöglichen. Hierzu werden aufeinanderfolgende Bilder aus einer Videosequenz analysiert und versucht, die Verschiebung der Objekte zu errechnen. Oft genügen hier bereits zwei Bilder um eine Bewegung zu bestimmen. Dabei wird jedem Pixel des ersten Bildes ein Pixel aus dem zweiten Bild zugeordnet, welcher sich durch Bewegung und/oder einen anderen Blickwinkel der Kamera verändert hat. Diese Beziehung zu erstellen ist im Vergleich zum Menschen allerdings nicht so trivial wie es auf den ersten Blick scheint, da ein Computer keinerlei Verbindung zwischen den Bildern erkennt. In der Computer-Vision wurden deshalb einige mathematische Modelle zur Beschreibung dieses Verhaltens entwickelt, welche im nächsten Abschnitt näher erläutert werden.

3.1 Geschichte und Entwicklung

Die ersten Modelle des Optischen Flusses funktionierten rein auf der Erkennung eines bestimmten Farb- oder Grauwerts. Die simpelste Variante davon wurde durch einfache Vergleiche der Pixelfarbe zwischen den Bildern erzielt. Dies hatte allerdings den gravierenden Nachteil, dass viele Pixel die gleiche Farbe haben können und so ein wildes Vektorfeld ohne Ordnung entstehen konnte. Als Folge dessen wurden lokale diskrete Methoden wie das Block-Matching Verfahren entwickelt. Hier wird jedem Pixel eine gewisse Nachbarschaft und Distanz eingeräumt, in welcher er sich aller Voraussicht nach bewegt. So konnte der maximale Bewegungsraum eingegrenzt und durch Vergleiche der Nachbarschaft eine Übereinstimmung der Pixel sichergestellt werden. Ein Vorteil dieser Methoden ist, dass sie durch einfache Vergleiche überaus performant sind, allerdings keine genaue Schätzung der Bewegung zulassen. Mit zunehmender Leistung der Computer konnten jedoch vermehrt rechenintensive Verfahren verwendet werden, was zur Entwicklung von kontinuierlichen Modellen, wie das Verfahren von Lucas-Kanade [LK81] führte. Hiermit konnte die Bewegung explizit durch ein mathematisches Modell berechnet werden. Im Jahr 1981 wurde schließlich das erste globale kontinuierliche Verfahren von Horn und Schunck [HS81] entwickelt, welches sich bis heute als zuverlässige Grundlage vieler Modelle herausgestellt hat und auch Grundlage dieser Arbeit ist.

3.2 Grundlagen des Optischen Flusses

Im nächsten Kapitel werden einige grundlegende Bedingungen definiert, aus dessen Komponenten sich das Problem der Berechnung des Optischen Flusses in ein mathematisches Modell zusammenfassen lässt. Zudem folgt eine Erklärung der einzelnen Komponenten, welche einen genauen Einblick in die Funktion und der Idee dahinter gewährt.

3.2.1 Vorglättung

Bevor die eigentlichen Komponenten genauer untersucht werden, wird zuerst ein grundlegender Vorverarbeitungsschritt des Optischen Flusses erläutert, welcher in der Regel vor allen anderen durchgeführt wird. Bei digitalen Bildern kommt es häufig vor, dass bei der Aufnahme wenig oder gar kein Licht auf bestimmte Pixel fällt und so eine großer Unterschied bezüglich der Helligkeit und Farbe auftritt. Dies wird auch Bildrauschen genannt. Da diese Stellen später leicht zu Fehler führen können, werden die Bilder deshalb mit dem Gaußkern der Standardabweichung σ gefaltet und auf das Initialbild f_0 angewendet:

$$f = K_\sigma * f_0. \tag{3.1}$$

Durch Analyse der benachbarten Pixel können so einzelne Ausreißer, die nichts mit dem Bildinhalt zu tun haben effektiv beseitigt werden. Der Parameter σ sollte jedoch nicht zu hoch sein, da sonst das Bild verschwimmt.

3.2.2 Grauwertkonstanz

Die Grauwertkonstanz ist eine Gleichung, welche das Verhältnis des Grauwerts eines Pixels aus Bild 1 mit dem Grauwert eines Pixels aus Bild 2 vergleicht und wie folgt berechnet werden kann:

$$f(x+u, y+v, t+1) = f(x, y, t). \tag{3.2}$$

Sie besteht aus einer Bildfunktionen f, die zu einer gegebenen Koordinate x,y und dem Zeitpunkt t den Grauwert liefert. Da dieser Pixel sich durch die Verschiebung mit großer Wahrscheinlichkeit im darauffolgenden Bild zum Zeitpunkt $t+1$ nicht mehr an dieser Stelle befindet, wird ein Pixel gesucht der den gleichen Grauwert besitzt, jedoch durch u in x-Richtung sowie v in y-Richtung verschoben ist. Die Annahme geht davon aus, dass bei zwei Bildern, die zeitlich gesehen direkt aufeinander folgen, keine oder nur vernachlässigbare Unterschiede in ihrer Helligkeit aufweisen. Dies lässt sich dadurch erklären, dass der zeitliche Abstand zwischen den beiden Bildern je nach Bildrate nur wenige Millisekunden darstellt und angesichts dessen, kein entscheidender Unterschied sichtbar ist. Durch Subtraktion der rechten Seite

$$f(x+u, y+v, t+1) - f(x, y, t) = 0 \tag{3.3}$$

lässt sich die Gleichung gleich null setzen und so ein Unterschied der beiden Pixel durch Abweichungen von 0 erkennen.

3.2.3 Glattheitsannahme

Die Glattheitsannahme fand erst durch das Verfahren von Horn und Schunck Einzug in die Berechnung des Optischen Flusses. Sie beschreibt das Verhältnis des Flusses benachbarter Pixel.

$$|\nabla u|^2 + |\nabla v|^2 = 0 \tag{3.4}$$

Der Fluss wird für jeden Pixel durch einen Vektor mit den beiden Funktionen $u(x, y)$ in x-Richtung sowie $v(x, y)$ in y-Richtung bestimmt, welche die jeweilige Verschiebung an den Koordinaten x und y darstellen. Die Glattheitsannahme geht davon aus, dass benachbarte Pixel durch Verschiebung in eine Richtung, auch im Nachhinein noch benachbart sind, somit die gleiche Bewegung haben sollten. Im normalen Sachverhalt lässt sich dies dadurch erklären, dass Objekte aus vielen Pixel bestehen. Wird beispielsweise eine Tasse auf einem Tisch um 10 Zentimeter verschoben, werden alle Pixel die die Tasse darstellen in die gleiche Richtung bewegt. Formuliert wird dies durch den Gradienten von u und v, wobei ein geringer Wert für nahezu identische Verschiebung steht und ein großer Wert für eine Bewegung in unterschiedliche Richtungen oder einer anderen Intensität der Verschiebung. Dies ist beispielsweise bei dem Übergang zwischen zwei verschiedenen Objekten der Fall und in Abbildung 3.1 anhand eines Schemas verdeutlicht.

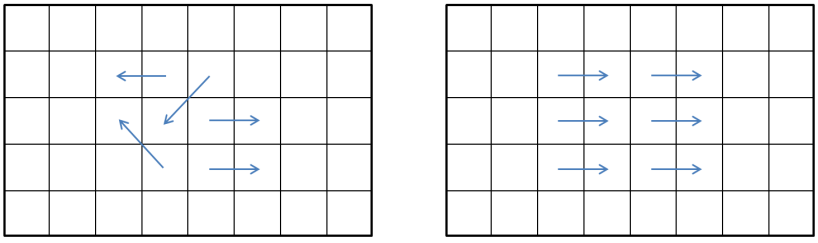


Abbildung 3.1: Zu sehen ist eine schematische Darstellung der Glattheit. **Links:** Nur wenig Glattheit ist vorhanden, die Verschiebung findet in viele Richtungen statt. **Rechts:** Die Verschiebung ist einheitlich in eine Richtung.

3.2.4 Darstellung des Vektorfelds

Für die Auswertung der späteren Vektorfelder wird eine geeignete Darstellung benötigt, die nicht nur einfach zu lesen ist, sondern auch möglichst viele Informationen beinhaltet. Im Laufe der Zeit haben sich zwei Varianten bewährt, auf welche im folgenden Abschnitt näher eingegangen wird.

Vektorielle Darstellung

Das Vektorfeld mithilfe von Vektoren darzustellen, klingt auf den ersten Blick schlüssig, da bekanntermaßen die Verschiebung jedes Pixels durch einen Vektor berechnet wird. So kann sehr leicht der Ursprung und die jeweilige Verschiebung beobachtet werden indem das Flussfeld mit dem ursprünglichen Bild überlagert wird. Für dichte Vektorfelder, welche in dieser Arbeit berechnet werden, sind vektorielle Darstellungen jedoch weniger geeignet, da die Vektoren selbst viel Platz benötigen und Pixel überdecken. Als Folge dessen können nur wenige Pixel dargestellt werden und es können schnell Bildstörungen entstehen, falls sich die Vektoren kreuzen. Abbildung 3.2 zeigt ein mögliches Vektorfeld.

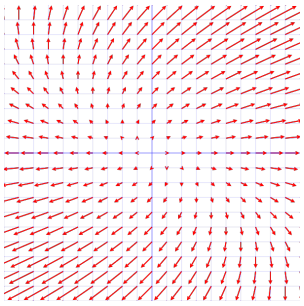


Abbildung 3.2: Vektordarstellung eines Flussfeldes mit eingezeichneten Vektoren.

Farbliche Darstellung

Eine andere Variante der Darstellung ist, die Vektoren farblich zu kodieren. Hierfür wird ein Template benötigt, das jeder Richtung und Länge des Vektors eine andere Farbe zuweist.

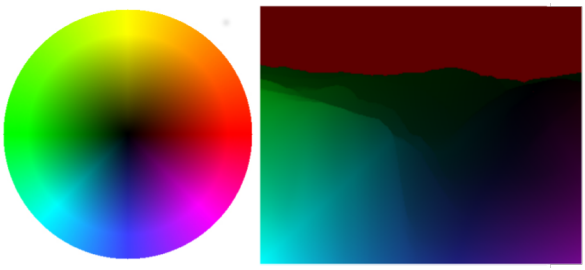


Abbildung 3.3: Farbdarstellung mit Templatekreis und einem visuell dargestellten Flussfeld.

Für das Template wird ein Farbkreis gewählt, welcher die Farben des RGB-Raums abdeckt. Wobei das Zentrum des Kreises für den Nullvektor oder für keine Verschiebung steht und mit schwarz kodiert wird. Die maximal auftretende Länge des Vektors wird auf den Rand gesetzt und mit der größten Intensität der jeweiligen Farbe dargestellt. Kürzere Vektoren erhalten somit einen Farbverlauf zwischen der Richtungsfarbe und schwarz. Der Vorteil gegenüber der vektoriellen Darstellung ist, dass

jeder Pixel angezeigt und leicht an der Helligkeit sowie Farbe die jeweilige Verschiebung abgelesen werden kann und Bewegungsübergänge besser sichtbar sind. Ein Beispiel eine Farbdarstellung und dem zugehörigen Farbkreis ist aus Abbildung 3.3 zu entnehmen.

3.3 Ansatz von Horn und Schunck

Im folgenden Abschnitt wird der Variationsansatz nach dem Funktional von Horn und Schunck aufgebaut, welches die Grundlage des Basisverfahrens darstellt. Hierzu wird die Grauwertkonstanz mit der Glattheitsannahme im Rahmen eines gemeinsamen Energiefunktionals kombiniert:

$$E(u,v) = \int_\Omega \underbrace{(f(x+u, y+v, t+1) - f(x,y,t))^2}_{Datenterm} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{Glattheitsterm} \, dxdy. \tag{3.5}$$

Das Energiefunktional E wird zum einen aus dem Datenterm und dem Glattheitsterm zusammengesetzt, welche Abweichungen erkennen und ihnen eine Energie (Kosten) zuordnen. Dabei gilt, je größer die Energie, desto schlechter erfüllt die Verschiebung die zuvor getroffene Annahme hinsichtlich der Grauwertkonstanz und Glattheit. Der Datenterm besteht aus den Informationen, welche direkt aus den Bildern gelesen werden können und in diesem Fall aus der quadrierten Grauwertkonstanz besteht. Sie bestraft Abweichungen des Grauwerts aus Bild 1 und Bild 2 und ordnet ihnen eine Energie $>= 0$ zu, wobei 0 für einen identischen Pixel steht und die Energie mit steigendem Unterschied wächst. Hinzukommt der Glattheitsterm, der nach dem gleichen Prinzip handelt, jedoch Abweichungen in der Glattheit des Flusses in x- und y-Richtung mit einer höheren Energie bestraft. Die Regularisierungskonstante α wird für das Verhältnis der zwei Terme verwendet, mit welcher sich der Fokus auf den jeweiligen Term verstärken lässt. Wird α ein großer Wert zugewiesen, haben schon kleine Abweichungen in der Glattheit Einfluss auf die Gesamtenergie. Das berechnete Flussfeld wird somit glatter. Am Ende wird das bestimmte Integral über den Bildbereich Ω ermittelt. Geometrisch gesehen wird so die Fläche der Funktion bestimmt, was diskret der Summe aller Pixel entspricht. Das Ziel ist in Abhängigkeit von u und v das kleinste Volumen (die Summe der Kosten) zu finden, da dieses für die minimale Energie und unter Betrachtung aller Pixel für die allgemein beste Lösung steht. Ein mögliches Flussfeld wird in Abbildung 3.4 dargestellt.

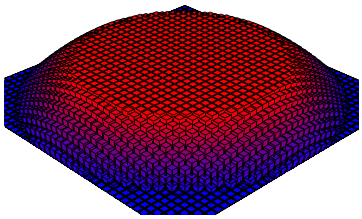


Abbildung 3.4: Eine möglicher Plot der Energie für jeden Pixel. Die Energie wird für jeden Pixel jeweils durch einen Balken dargestellt. Je kleiner das Volumen, desto besser die Lösung.

3.3.1 Aperturproblem

Das Aperturproblem ist ein häufig auftretendes Problem in der Bildverarbeitung. Es beschreibt die Problematik, bei welcher eine Verschiebung durch fehlenden Kontrast oder Struktur nicht eindeutig zuordenbar ist.

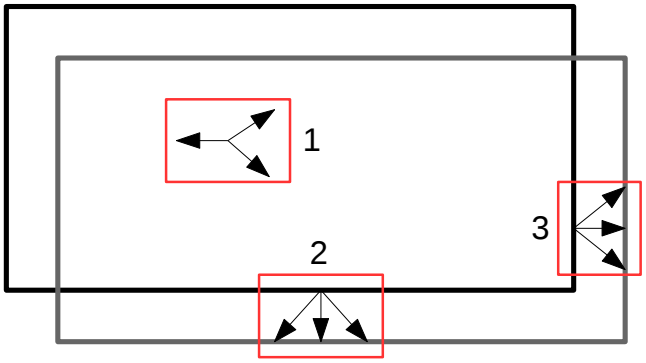


Abbildung 3.5: Schematische Darstellung der Aperturproblems mit 3 Fällen. **Rot:** Verschiebung in keiner oder nur in einer Richtung erkennbar, jedoch nicht auf der orthogonalen Achse.

Da die Erkennung der Verschiebung über den Datenterm nur lokal erfolgt und nur ein kleiner Ausschnitt des Bildes verarbeitet wird, können Situationen entstehen in denen die Richtung der Bewegung nicht eindeutig erkennbar ist. Um den Fluss für einen Pixel zu berechnen und ihn zuzuweisen, muss der Pixel sich von seiner Umgebung durch Helligkeit, Kontrast oder Struktur abheben. Abbildung 3.5 zeigt ein Rechteck, das nach unten rechts verschoben wird, wobei die kleinen Rechtecke den aktuellen Bereich markieren. Das Aperturproblem kann in 2 verschiedenen Varianten auftreten. Zum einen in komplett homogenen Flächen, wie in Beispiel 1. Aufgrund fehlender Struktur, lässt sich hier die Richtung nicht erkennen, da alle Pixel untereinander die Bedingung der Grauwertkonstanz erfüllen. Beispiel 2 und 3 illustrieren einen zum Teil erkennbaren Fluss. Die Verschiebung ist in Richtung des Bildgradienten sichtbar, jedoch kann nichts über die orthogonale Richtung gesagt werden, da nur eine Gleichung für zwei Unbekannte vorhanden ist. Zur Lösung des Aperturproblems an diesen Stellen tritt in diesem Zusammenhang der Glattheitsterm in Kraft. Sollte die Grauwertkonstanz keine Lösung liefern, versucht der Glattheitsterm die Energie der Pixel so klein wie möglich zu halten. Dies hat zur Folge, dass der Fluss eines Pixel auf alle Nachbarn übertragen wird, da ein glattes Ergebnis ohne Beachtung der Grauwertkonstanz, einer Energie von 0 und der besten Lösung entspricht. Somit kann durch Propagation von u und v der Fluss für alle Pixel bestimmt werden.

3.3.2 Farbe

Das aktuelle Funktional Horn und Schunck betrachtet bisher nur Grauwertbilder, aus denen die Information anhand der Helligkeitsunterschiede extrahiert wird. Farbbilder enthalten jedoch weit mehr Informationen, da die Farbe aus mehreren Farbkanälen zusammengestellt wird, welche einzeln betrachtet werden können. Hierzu kann die Grauwertkonstanz 3.2.2 auf die Farbkanäle erweitert und in drei Gleichungen aufgeteilt werden:

$$\begin{aligned} f_1(x+u, y+v, t+1) - f_1(x,y,t) &= 0, \\ f_2(x+u, y+v, t+1) - f_2(x,y,t) &= 0, \\ f_3(x+u, y+v, t+1) - f_3(x,y,t) &= 0 \end{aligned} \tag{3.6}$$

wobei f_1, f_2, f_3 für die jeweiligen Farbkanäle des RGB-Farbraums stehen. Auf diese Weise kann das Flussfeld anhand aller drei Gleichungen berechnet werden, indem sie die bisherige Grauwertkonstanz im Energiefunktional ersetzen:

$$E(u,v) = \int_\Omega \underbrace{\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2}_{Datenterm} + \alpha \underbrace{(|\nabla u|^2 + |\nabla v|^2)}_{Glattheitsterm} \, dxdy. \tag{3.7}$$

3.3.3 Subquadratische Bestrafungsfunktion

Die Methode von Horn und Schunck setzt eine globale Glattheit des Flusses voraus. Dies hat zur Folge dass speziell auf ein glattes Ergebnis geachtet wird. Bei der Minimierung der Energie werden deshalb Flüsse bevorzugt, welche ein glatteres Ergebnis liefern. Viele Objekte bewegen sich allerdings in verschiedene Richtungen und haben deshalb einen unterschiedlichen Fluss. Dies ist besonders an den Übergängen zwischen den Objekten problematisch, da hier Sprünge im Fluss zu Unrecht mit einer hohen Energie bestraft werden. Um den Einfluss dieser Stellen zu vermindern, wird die Funktion Ψ eingeführt, welche die Bestrafung verringert und somit die Kanten besser erhalten soll. Dazu kann zum Beispiel die subquadratische Charbonnier-Bestrafungsfunktion [CBFAB94] verwendet werden:

$$\Psi(s^2) = 2\epsilon^2 \cdot \sqrt{1 + \frac{s^2}{\epsilon^2}} \tag{3.8}$$

wobei s^2 durch $|\nabla u|^2 + |\nabla v|^2$ gegeben ist. Der Parameter ϵ stellt hier einen Kontrastparameter dar, der angibt, welche Sprünge im Flussfeld erhalten bleiben sollen und welche geglättet werden. Ein solcher Glattheitsterm wird in der Literatur auch als isotroper flussgetriebener Glattheitsterm bezeichnet.

Das gleiche Prinzip kann darüber hinaus für Ausreißer des Datenterms angewendet werden. Pixel können zum einen durch Rauschen einen anderen Farbwert annehmen, zum anderen können Objekte, die an den Rändern aus dem Bild fließen, die Energie erhöhen da sie in dem darauffolgenden Bild

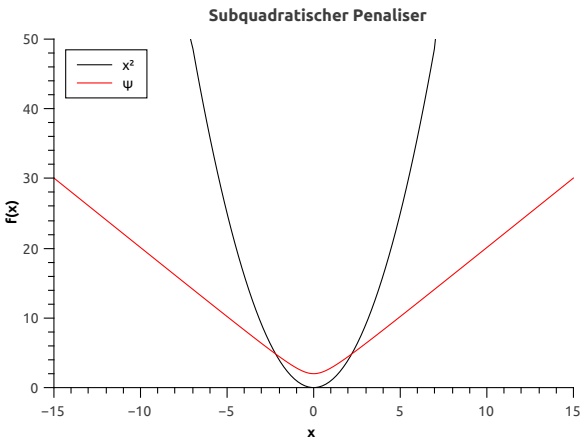


Abbildung 3.6: Vergleich zwischen der quadratischen (Schwarz) und subquadratischen (Rot) Bestrafungsfunktion.

nicht zu lokalisieren sind. Als Argument s^2 bekommt die Funktion den Datenterm überliefert. Die Regularisierung übernimmt auch hier ein Kontrastparameter ϵ. Das Funktional der Horn und Schunck Methode mit robustem Daten- und Glattheitsterm sieht demnach wie folgt aus:

$$E(u,v) = \int_\Omega \underbrace{\Psi_D \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2 \right)}_{Datenterm} + \alpha \underbrace{\Psi_S(|\nabla u|^2 + |\nabla v|^2)}_{Glattheitsterm} \, dxdy \tag{3.9}$$

wobei ϵ_D und ϵ_S die Kontrastparameter der subquadratischen Bestrafungsfunktionen im Daten- und Glattheitsterm sind. Der Unterschied zwischen einer quadratischen und einer subquadratischen Funktion ist in Abbildung 3.6 dargestellt.

3.3.4 Minimierung

Die Minimierung eines Funktionals wie es bei der Methode von Horn und Schunk vorliegt, hat Ähnlichkeit mit der Minimierung von gewöhnlichen Funktionen. Dazu wird die notwendige Bedingung für Extremwerte benötigt, bei der die Ableitung an diesen Punkten 0 sein muss. Voraussetzung dafür ist eine zweifach stetig differenzierbare Funktion der Form

$$E(u,v) = \int_\Omega F(x, y, u, v, u_x, u_y, v_x, v_y) \, dxdy \tag{3.10}$$

wobei u und v die gesuchten Funktionen beschreiben, u_x, u_y und v_x, v_y die partiellen Ableitungen sind und der stationäre Punkt durch x und y beschrieben wird.

3.3.5 Euler-Lagrange-Gleichung

Für die Lösung des obigen Funktionals muss die Variationsrechnung herangezogen werden. Diese sagt aus, dass die sogenannten Euler-Lagrange-Gleichungen eine notwendige Bedingung für jeden Minimierer darstellen. Dazu wird die optimale Lösung $u(x, y)$ und $v(x, y)$ angenommen und diese mit einer Vielzahl an Perturbationsfunktionen der Form $\epsilon\mu(x, y)$ variiert. Beispiele der Perturbatiosn-funktionen sind aus Abbildung 3.7 zu entnehmen. Die Variable ϵ steht für den Variierungsparameter, der μ verändert [CH93]. Das Problem kann durch folgende Gleichung formuliert werden:

$$E = \int_{\Omega} F(x, y, U, V, U_x, U_y, V_x, V_y)\, dxdy \tag{3.11}$$

mit den Substitutionen

$$U = u + \epsilon_1\mu_1, \qquad U_x = u_x + \epsilon_1\mu_{1_x}, \qquad U_y = u_y + \epsilon_1\mu_{1_y}, \tag{3.12}$$

$$V = v + \epsilon_2\mu_2, \qquad V_x = v_x + \epsilon_2\mu_{2_x}, \qquad V_y = v_y + \epsilon_2\mu_{2_y}. \tag{3.13}$$

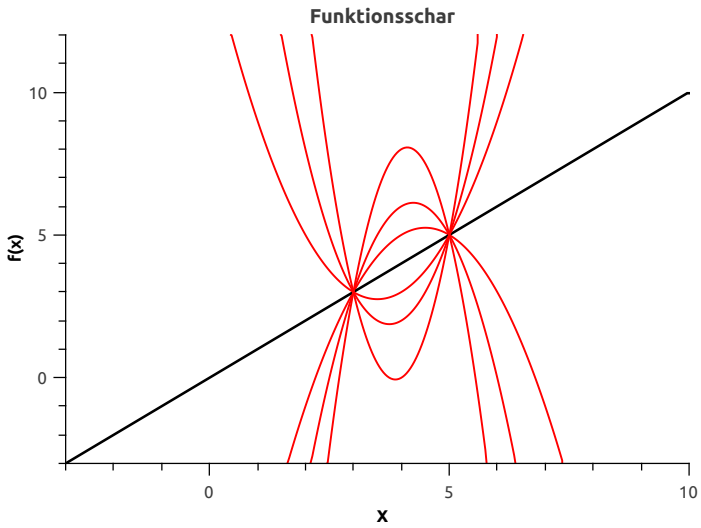


Abbildung 3.7: Schema der Funktionsschar eines Funktionals mit der angenommenen Lösung (Schwarz) und einigen Funktionen μ (Rot), welche sich alle in den Grenzen schneiden und mit ϵ skaliert wurden. Zu erkennen ist, dass bei $\epsilon = 0$, sich beide Funktionen überschneiden, was der optimalen Lösung gleich kommt.

So lässt sich jeweils eine reellwertige Funktion zweier Variablen (ϵ_1, ϵ_2) konstruieren, die bei $\epsilon_1 = \epsilon_2 = 0$ ein Minimum haben muss, da dort $U = u$ und $V = v$ gilt:

$$
\begin{aligned}
\left.\frac{\partial E}{\partial \epsilon_1}\right|_{\epsilon_1=0} &= \int_\Omega \frac{\partial F}{\partial U} \cdot \frac{\partial U}{\partial \epsilon_1} + \frac{\partial F}{\partial U_x} \cdot \frac{\partial U_x}{\partial \epsilon_1} + \frac{\partial F}{\partial U_y} \cdot \frac{\partial U_y}{\partial \epsilon_1}\, dxdy \Bigg|_{\epsilon_1=0} = 0, \\
\left.\frac{\partial E}{\partial \epsilon_2}\right|_{\epsilon_2=0} &= \int_\Omega \frac{\partial F}{\partial V} \cdot \frac{\partial V}{\partial \epsilon_2} + \frac{\partial F}{\partial V_x} \cdot \frac{\partial V_x}{\partial \epsilon_2} + \frac{\partial F}{\partial V_y} \cdot \frac{\partial V_y}{\partial \epsilon_2}\, dxdy \Bigg|_{\epsilon_2=0} = 0.
\end{aligned}
\tag{3.14}
$$

Die partiellen Ableitungen lassen sich durch Ableitung der Gleichungen 3.12 und 3.13 nach ϵ_1 und ϵ_2 bestimmen. Wird die Funktion am Minimum ausgewertet, das bedeutet wird ϵ_1 und ϵ_2 gleich 0 gesetzt, können so die gesuchten Funktionen u und v entsprechend eingesetzt werden:

$$
\begin{aligned}
\frac{\partial E}{\partial \epsilon_1} &= \int_\Omega \frac{\partial F}{\partial u} \cdot \mu_1 + \frac{\partial F}{\partial u_x} \cdot \mu_{1_x} + \frac{\partial F}{\partial u_y} \cdot \mu_{1_y}\, dxdy = 0 \\
\frac{\partial E}{\partial \epsilon_2} &= \int_\Omega \frac{\partial F}{\partial v} \cdot \mu_2 + \frac{\partial F}{\partial v_x} \cdot \mu_{2_x} + \frac{\partial F}{\partial v_y} \cdot \mu_{2_y}\, dxdy = 0
\end{aligned}
\tag{3.15}
$$

Anschließend können die beiden hinteren Terme mit partieller Integration integriert und die Funktionen μ_1 sowie μ_2 ausgeklammert werden:

$$
\begin{aligned}
\int_\Omega \left(\frac{\partial F}{\partial u} - \frac{\partial F}{\partial x}\left(\frac{\partial F}{\partial u_x}\right) - \frac{\partial F}{\partial y}\left(\frac{\partial F}{\partial u_y}\right)\right) \cdot \mu_1\, dxdy + \left.\frac{\partial F}{\partial u_x}\right|_{\partial\Omega} \mu_1 + \left.\frac{\partial F}{\partial u_y}\right|_{\partial\Omega} \mu_1 = 0 \\
\int_\Omega \left(\frac{\partial F}{\partial v} - \frac{\partial F}{\partial x}\left(\frac{\partial F}{\partial v_x}\right) - \frac{\partial F}{\partial y}\left(\frac{\partial F}{\partial v_y}\right)\right) \cdot \mu_2\, dxdy + \left.\frac{\partial F}{\partial v_x}\right|_{\partial\Omega} \mu_2 + \left.\frac{\partial F}{\partial v_y}\right|_{\partial\Omega} \mu_2 = 0
\end{aligned}
\tag{3.16}
$$

Da diese Gleichungen für alle Perturbationsfunktionen μ_1 und μ_2 gelten müssen, müssen sie auch für die Unterklasse von Funktionen erfüllt sein, für die μ_1 und μ_2 am Rand 0 sind. Damit reicht es zunächst nur das Integral zu berücksichtigen. Um zu garantieren, dass das Integral für alle Perturbationsfunktionen 0 ist, muss an den Rändern gelten:

$$
\frac{\partial F}{\partial u_x} + \frac{\partial F}{\partial u_y} = 0, \tag{3.17}
$$

$$
\frac{\partial F}{\partial v_x} + \frac{\partial F}{\partial v_y} = 0. \tag{3.18}
$$

Um in beiden Gleichungen aus 3.16 das Minimum für die Funktionsschar μ_1 und μ_2 zu erhalten, muss somit der innere Teil der Klammer gleich null sein, womit sich die finalen Euler-Lagrange-Gleichungen mit zugehörigen natürlichen Randbedingungen bilden:

$$
\begin{aligned}
F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} = 0, \\
F_v - \frac{\partial}{\partial x} F_{v_x} - \frac{\partial}{\partial y} F_{v_y} = 0
\end{aligned}
\tag{3.19}
$$

mit den umformulierten natürlichen Randbedingungen

$$
\begin{aligned}
n^T \begin{pmatrix} F_{u_x} \\ F_{u_y} \end{pmatrix} = 0, \\
n^T \begin{pmatrix} F_{v_x} \\ F_{v_y} \end{pmatrix} = 0,
\end{aligned}
\tag{3.20}
$$

wobei n^T der transponierten Normale an den Rändern entspricht. Das Euler-Lagrange Framework lässt durch weitere Variablen und Funktionen erweitern, wobei für jede neue Funktion eine weitere Gleichung in Abhängigkeit dieser Funktion entsteht. Im Falle neuer Variablen entstehen weitere Terme, die an die jeweilige Gleichung analog zu den bisherigen Termen angehängt werden.

Im nächsten Schritt wird das Euler-Lagrange-Framework auf das bestehende Funktional angewendet. Dabei müssen die einzelnen Komponenten des Funktionals berechnet werden. Hierbei wird u und u_x als unabhängig betrachtet und anschließend mithilfe der Kettenregel nach den Variablen u und v abgeleitet. Zuerst wird der Glattheitsterm zur besseren Übersicht ausmultipliziert:

$$
F = \Psi_D \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2 \right) + \alpha \Psi_S (u_x^2 + u_y^2 + v_x^2 + v_y^2). \tag{3.21}
$$

Danach werden die benötigten partiellen Ableitungen berechnet:

$$
\begin{aligned}
F_u &= \Psi_D' \cdot 2 \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{ix}(x+u, y+v, t+1) \right), \\
F_v &= \Psi_D' \cdot 2 \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{iy}(x+u, y+v, t+1) \right),
\end{aligned}
\tag{3.22}
$$

$$
\begin{aligned}
F_{u_x} &= \alpha \Psi_S' 2 u_x, \\
F_{u_y} &= \alpha \Psi_S' 2 u_y, \\
F_{v_x} &= \alpha \Psi_S' 2 v_x, \\
F_{v_y} &= \alpha \Psi_S' 2 v_y.
\end{aligned}
\tag{3.23}
$$

Werden die einzelnen Komponenten in das Framework der Euler-Lagrange-Gleichung eingesetzt, entstehen die beiden Gleichungen für u und v, welche es zu minimieren gilt:

$$\begin{aligned} 0 &= \Psi'_D \cdot 2\xi_1 - \frac{\partial}{\partial x}\alpha\Psi'_S 2u_x - \frac{\partial}{\partial y}\alpha\Psi'_S 2u_y, \\ 0 &= \Psi'_D \cdot 2\xi_2 - \frac{\partial}{\partial x}\alpha\Psi'_S 2v_x - \frac{\partial}{\partial y}\alpha\Psi'_S 2v_y \end{aligned} \tag{3.24}$$

mit

$$\begin{aligned} \xi_1 &= \sum_{i=1}^{3}(f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{ix}(x+u, y+v, t+1), \\ \xi_2 &= \sum_{i=1}^{3}(f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{iy}(x+u, y+v, t+1). \end{aligned} \tag{3.25}$$

Anschließend wird die rechte Seite noch durch zwei geteilt und die restlichen partiellen Ableitung auf eine einheitlich Form gebracht. Diese können mithilfe der Divergenz zusammengefasst werden:

$$\begin{aligned} 0 &= \Psi'_D \cdot \sum_{i=1}^{3}(f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{ix}(x+u, y+v, t+1) \\ &\quad - \alpha \operatorname{div}(\Psi'_S \nabla u), \\ 0 &= \Psi'_D \cdot \sum_{i=1}^{3}(f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{iy}(x+u, y+v, t+1) \\ &\quad - \alpha \operatorname{div}(\Psi'_S \nabla v), \end{aligned} \tag{3.26}$$

wobei die Ableitungen der beiden Ψ-Funktionen folgendem entsprechen:

$$\Psi'_D = \frac{1}{\sqrt{1 + \frac{s^2}{\epsilon_D^2}}} \qquad \text{mit} \qquad s^2 = \sum_{1}^{3}(f_i(x+u, y+v, t+1) - f_i(x,y,t))^2, \tag{3.27}$$

$$\Psi'_S = \frac{1}{\sqrt{1 + \frac{s^2}{\epsilon_S^2}}} \qquad \text{mit} \qquad s^2 = (|\nabla u|^2 + |\nabla v|^2). \tag{3.28}$$

Neumann Randbedingung

Zur Lösung von Differentialgleichungen die auf einem geschlossenem Intervall definiert sind, müssen zusätzlich noch Randbedingungen spezifiziert werden, die das Verhalten am Rand vorgeben. Häufig verwendete Bedingungen sind hier Dirichlet-Randbedingungen [GT15], welche Funktionswerte des Randes Festlegen oder Neumann-Randbedingungen [GT15], die Werte für die Ableitung festlegen. In diesem Fall ergeben sich die Randbedingungen automatisch bei der Herleitung der Euler-Lagrange-Gleichungen (siehe 3.3.5):

$$\begin{aligned} n^T \begin{pmatrix} F_{u_x} \\ F_{u_y} \end{pmatrix} &= 0, \\ n^T \begin{pmatrix} F_{v_x} \\ F_{v_y} \end{pmatrix} &= 0, \end{aligned} \tag{3.29}$$

wobei n^T die transponierte Normale über den Rand ist und $F_{u_x}, F_{u_y}, F_{v_x}, F_{v_y}$ für die bereits bekannten Komponenten der Euler-Lagrange-Gleichung stehen. Jede Euler-Lagrange-Gleichung hat somit ihre eigene individuelle Randbedingung, welche die Werte in Richtung der Normale festlegt. Für diesen Fall sehen die Bedingungen wie folgt aus:

$$\begin{aligned} n^T \begin{pmatrix} 2\alpha\Psi'_S u_x \\ 2\alpha\Psi'_S u_y \end{pmatrix} = 0 &\iff n^T \nabla u = 0, \\ n^T \begin{pmatrix} 2\alpha\Psi'_S v_x \\ 2\alpha\Psi'_S v_y \end{pmatrix} = 0 &\iff n^T \nabla v = 0. \end{aligned} \tag{3.30}$$

Da alles gleich null gesetzt wird, kann $2\alpha\Psi'_S$ herausgezogen werden und fällt durch die Division weg. Die restliche Gleichung besagt, dass die Ableitung in Richtung der Normale 0 betragen muss.

3.3.6 Warping

Um den rechnerischen Aufwand und die Komplexität zu verringern, wird im herkömmlichen Horn und Schunck Verfahren eine Linearisierung des Datenterms vorgenommen. Hierdurch entsteht eine strikt konvexe Funktion mit einem globalen Minimum, welche die bisher nur als Argument der Bilder vorkommenden Funktionen u und v faktorisiert und lösbar macht. Dies hat allerdings zur Folge, dass die Bestimmung größerer Verschiebungen durch die Simplifizierung des Datenterms erschwert wird, da dieser mit steigender Entfernung stark an Genauigkeit verliert (siehe Sektion 2.1.12). Zur Erhaltung der großen Verschiebungen wird die Linearisierung deshalb hinausgezögert und die Minimierung mit der ursprünglichen nicht-konvexen Form vorgenommen (siehe 3.26), die einem genaueren Modell des Optischen Flusses entspricht. Der Unterschied zwischen einer konvexen und nicht-konvexen Funktion ist Abbildung 3.8 dargestellt.

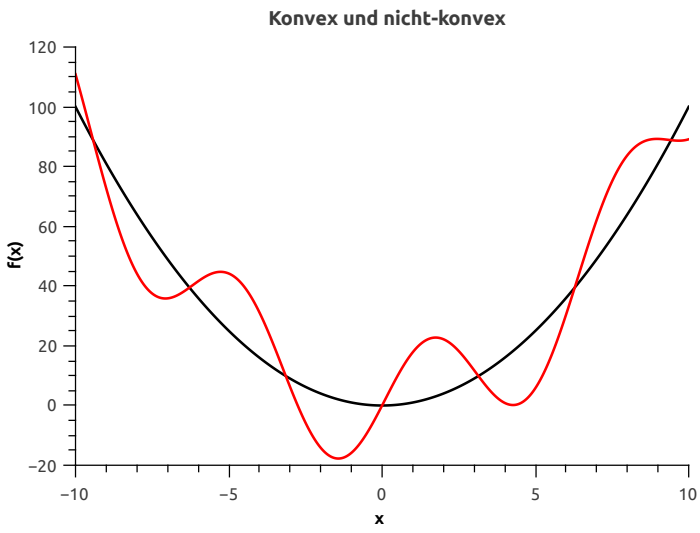


Abbildung 3.8: Illustration einer konvexen Funktion (Schwarz), bei der die Verbindung zweier beliebiger Punkte immer oberhalb der Funktion liegt und einer nicht-konvexen Funktion (Rot) bei der das nicht der Fall ist.

Dieses Energiefunktional besitzt jedoch den Nachteil, dass u sowie v implizit in der Funktion f stecken. Um dies zu lösen wird zunächst die Euler-Lagrange-Gleichung berechnet (Gleichung 3.26) und dort eine Fixpunktiteration [BBPW04] eingeführt um das ursprüngliche nicht-konvexe nicht-lineare Problem durch eine Folge von konvexen nicht-linearen Problemen zu approximieren. Dies geschieht in zwei Schritten:

$$\begin{aligned}
0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(x+u^{k+1}, y+v^{k+1}, t+1) - f_i(x,y,t)) \cdot f_{ix}(x+u^k, y+v^k, t+1) \\
&\quad - \alpha \operatorname{div}(\Psi'_S \nabla u^{k+1}), \\
0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(x+u^{k+1}, y+v^{k+1}, t+1) - f_i(x,y,t)) \cdot f_{iy}(x+u^k, y+v^k, t+1) \\
&\quad - \alpha \operatorname{div}(\Psi'_S \nabla v^{k+1}).
\end{aligned} \tag{3.31}$$

Im ersten Schritt werden Iterationschritte für u und v eingeführt, wobei die neue Variable k für den aktuellen Zeitschritt der Iteration steht und $k+1$ für den direkt folgenden. Anschließend wird u und v der Iteration $k+1$ aufgeteilt. Zum einen in den bereits berechneten Fluss u und v aus den älteren Iterationsschritten und dem Inkrement der Verschiebung du und dv, welches die Bewegung zwischen k und $k+1$ darstellt:

$$
\begin{aligned}
u^{k+1} &= u^k + du^k, \\
v^{k+1} &= v^k + du^k.
\end{aligned}
\tag{3.32}
$$

u und v setzen sich demnach aus vielen kleinen Verschiebungen von du und dv zusammen, wobei für den Datenterm eine implizite Berechnung aus beiden Zeitpunkten und bei dem Glattheitsterm eine semi-implizite Variante anhand des zweiten Zeitpunkts gewählt wird. Beide Varianten sind der expliziten Berechnung vorzuziehen, da sie eine schnellere Konvergenz aufweisen. Eingesetzt in die Fixpunkt-Iteration bildet dies das Schema zur inkrementellen Berechnung des Flusses:

$$
\begin{aligned}
0 = \Psi'_D \cdot \sum_{i=1}^{3} &(f_i(x + u^k + du^k, y + v^k + dv^k, t+1) - f_i(x,y,t)) \cdot f_{ix}(x + u^k, y + v^k, t+1) \\
&- \alpha \, div(\Psi'_S \nabla(u^k + du^k)), \\
0 = \Psi'_D \cdot \sum_{i=1}^{3} &(f_i(x + u^k + du^k, y + v^k + dv^k, t+1) - f_i(x,y,t)) \cdot f_{iy}(x + u^k, y + v^k, t+1) \\
&- \alpha \, div(\Psi'_S \nabla(v^k + dv^k)).
\end{aligned}
\tag{3.33}
$$

In einem zweiten Schritt wird das Funktional linearisiert, da sich die Flusskomponenten zum Teil als Argument innerhalb der Bildfunktionen befinden und die spätere Gleichung nach diesen aufgelöst werden soll. Dies geschieht jedoch nur in Abhängigkeit von du und dv, da sich der komplette Fluss aus der Summe dieser zusammensetzt. Dafür muss die Taylorreihe auf die entsprechende Form für Vektoren angepasst werden:

$$
f(x) \approx f(a) + (x-a)^T \nabla_3 f(a). \tag{3.34}
$$

Anschließend wird um den Punkt $a = (x + u^k, y + v^k, t+1)$ nach du und dv linearisiert.

$$
\begin{aligned}
f_i(x + u^k + du^k, y + v^k + dv^k, t+1) \approx\ & f_i(x + u^k, y + v^k, t+1) \\
&+ \begin{pmatrix} x + u^k + du^k - (x + u^k) \\ y + v^k + dv^k - (y + v^k) \\ (t+1) - (t+1) \end{pmatrix}^T \\
&\cdot \begin{pmatrix} f_{ix}(x + u^k, y + v^k, t+1) \\ f_{iy}(x + u^k, y + v^k, t+1) \\ f_{it}(x + u^k, y + v^k, t+1) \end{pmatrix}.
\end{aligned}
\tag{3.35}
$$

Durch Subtraktion der einzelnen Komponenten lässt sich die Gleichung vereinfachen:

$$f_i(x+u^k, y+v^k, t+1) + \begin{pmatrix} du^k \\ dv^k \\ 0 \end{pmatrix} \begin{pmatrix} f_{ix}(x+u^k, y+v^k, t+1) \\ f_{iy}(x+u^k, y+v^k, t+1) \\ f_{it}(x+u^k, y+v^k, t+1) \end{pmatrix}. \tag{3.36}$$

Um die Linearisierung noch einmal auf die bekannte Form zu bringen, können die Vektoren ausmultipliziert werden:

$$f_i(x+u^k, y+v^k, t+1) + f_{ix}(x+u^k, y+v^k, t+1)du^k + f_{iy}(x+u^k, y+v^k, t+1)dv^k. \tag{3.37}$$

Dies wird anschließend in die ursprüngliche Gleichung eingesetzt und vereinfacht:

$$\begin{aligned}
0 = \Psi'_D \cdot \sum_{i=1}^{3} (f_{ix}(x+u^k, y+v^k, t+1)du^k + f_{iy}(x+u^k, y+v^k, t+1)dv^k & \tag{3.38}\\
+ \underbrace{f_i(x+u^k, y+v^k, t+1) - f_i(x,y,t)}_{\approx f_{it}}) \cdot f_{ix}(x+u^k, y+v^k, t+1) & \tag{3.39}\\
- \alpha \operatorname{div}(\Psi'_S \nabla u^k) - \alpha \operatorname{div}(\Psi'_S \nabla du^k), & \tag{3.40}\\
0 = \Psi'_D \cdot \sum_{i=1}^{3} (f_{ix}(x+u^k, y+v^k, t+1)du^k + f_{iy}(x+u^k, y+v^k, t+1)dv^k & \tag{3.41}\\
+ \underbrace{f_i(x+u^k, y+v^k, t+1) - f_i(x,y,t)}_{\approx f_{it}}) \cdot f_{iy}(x+u^k, y+v^k, t+1) & \tag{3.42}\\
- \alpha \operatorname{div}(\Psi'_S \nabla v^k) - \alpha \operatorname{div}(\Psi'_S \nabla dv^k). & \tag{3.43}
\end{aligned}$$

Durch die Linearisierung hat sich eine Subtraktion der zeitlichen Komponente t gebildet. Da eine Subtraktion bei Bild 2 von Bild 1 nichts anderes als eine Ableitung hinsichtlich der Zeit approximiert, können beide Terme durch $f_{it}(x+u^k, y+v^k, t+1)$ ersetzt werden. Im letzten Schritt muss die Klammer ausmultipliziert werden:

$$\begin{aligned}
0 &= \Psi'_D \cdot (J^k_{11}du^k + J^k_{12}dv^k + J^k_{13}) - \alpha(\operatorname{div}(\Psi'_S \nabla u^k) - \operatorname{div}(\Psi'_S \nabla du^k)),\\
0 &= \Psi'_D \cdot (J^k_{12}du^k + J^k_{22}dv^k + J^k_{23}) - \alpha(\operatorname{div}(\Psi'_S \nabla v^k) - \operatorname{div}(\Psi'_S \nabla dv^k)),
\end{aligned} \tag{3.44}$$

wobei $J^k_{11}, J^k_{12}, J^k_{13}, J^k_{22}$ und J^k_{23} Einträge des Bewegungstensors

$$J^k = \sum_{i=1}^{3} \begin{pmatrix} f_{ix}^2 & f_{ix}f_{iy} & f_{ix}f_{it} \\ f_{iy}f_{ix} & f_{iy}^2 & f_{iy}f_{it} \\ f_{it}f_{ix} & f_{it}f_{iy} & f_{it}^2 \end{pmatrix} \tag{3.45}$$

an der Stelle $(x+u^k, y+v^k, t+1)$ darstellen.

Hierarchische Minimierung

Im letzten Schritt muss die inkrementelle Fixpunktiteration in eine sogenannte Coarse-to-Fine-Pyramide eingebettet werden, um das Problem der lokalen Minima zu lösen. Dabei werden die Bilder stufenweise auf verschiedenste Auflösungen herunterskaliert, wobei immer mehr Details durch die begrenzte Anzahl an Pixel verschwinden und so eine simplere Variante des Bildes entsteht. Eine mögliche Pyramide ist in Abbildung 3.9 zu sehen. Dies hat zum einen den Vorteil, dass dadurch viele lokale Minima verschwinden, da nur eine grobe (coarse) Repräsentation des Bildes übrig bleibt. Es bedeutet allerdings nicht, dass zwangsweise das globale Minimum gefunden wird, sondern die Iteration auch in einem guten lokalen Minimum sein Ende finden kann. Zum anderen verringert sich die Distanz der Verschiebungen zunehmend mit dem Level der Pyramide. Eine Bewegung, welche im Originalbild 100 Pixel beträgt, kann auf der Skalierung mit der kleinsten Auflösung nur noch einen einzigen Pixel ausmachen. Da der Fluss bereits auf dem gröbsten Level berechnet wird, wo Verschiebungen allesamt klein ausfallen und mit jedem feineren Level nur das Inkrement, was ebenfalls klein ist, berechnet und aufsummiert werden muss, entspricht die linearisierte Konstanzannahme hiermit einer guten und geeigneten Approximation des Problems.

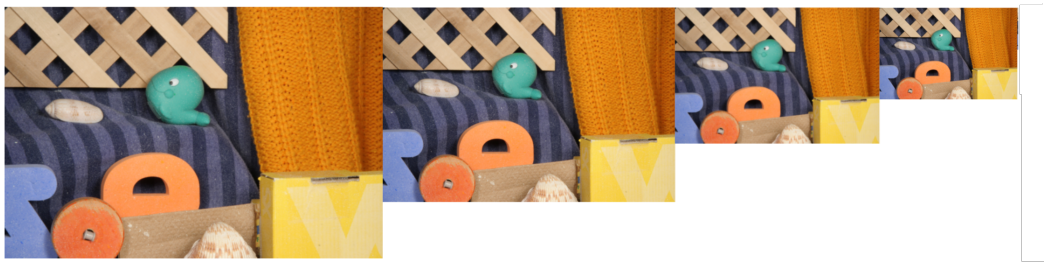


Abbildung 3.9: Darstellung einer Coarse-to-Fine-Pyramide mit vier Leveln, die aus verschieden aufgelösten Bildern besteht und von links nach rechts an Genauigkeit verliert.

Rückwärtsregistrierung

Da der Fluss auf jedem Level durch den Unterschied beider Bilder berechnet wird, muss der bereits bekannte Fluss in die Bilder mit einfließen. Um die Voraussetzungen für ein kleines Inkrement zu gewährleisten, ist es deshalb notwendig das zweite Bild nach jedem Level um den bereits berechneten Fluss zu verschieben (warpen). Dadurch wird jedem Pixel des zweiten Bildes der aktuelle Vorwärts-Fluss mit

$$\hat{f}_2(x, y, t) = f_2(x + du^k, y + dv^k, t + 1) \tag{3.46}$$

zugeordnet und an seiner Position anhand des zuvor berechneten Flusses platziert. Hierbei wird ein kompensiertes zweites Bild generiert, das ab diesem Zeitpunkt den Platz des originalen zweiten Bildes übernimmt und für weitere Berechnungen verwendet wird. Abbildung 3.10 zeigt das Schema der Rückwärtsregistrierung anhand eines kleines Ausschnitts.

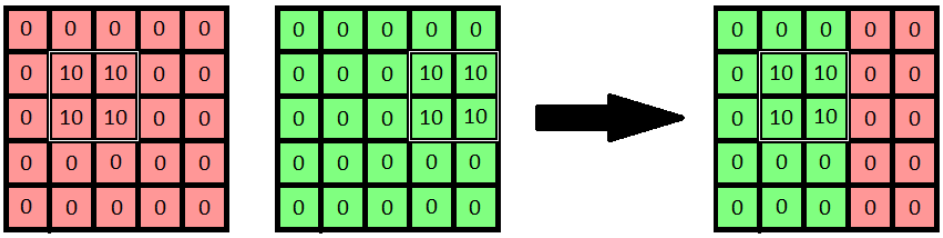


Abbildung 3.10: Berechnung des gewarpten Bildes. Der negative Fluss (hier $u = 2$) wird von Bild 2 benutzt um Bild 1 mithilfe des aktuellen Flusses darzustellen. Durch Objekte außerhalb des Bildbereichs kann so eine Kombination aus beiden Bildern entstehen.

Zusammenfassung

Da das Warping aus vielen Schritten besteht, werden zur besseren Übersicht nochmal alle relevanten Vorgehensweisen in diesem Abschnitt zusammengefasst und der Algorithmus anschaulich dargestellt.

1. Verschiedene Level der Pyramide durch Skalierung der Originalbilder erstellen.
2. Auf dem gröbsten Level beginnen und Startwerte u^0 und v^0 mit 0 initialisieren.
3. Für jedes Level:
   a) Inkremente du^k und dv^k berechnen.
   b) Inkremente du^k und dv^k auf Gesamtfluss u^k und v^k addieren.
   c) Bekannten Fluss u^{k+1} und v^{k+1} durch Interpolation auf nächstfeineres Level skalieren.
   d) Das zweite Bild durch Rückwärsregistrierung um den aktuellen Fluss warpen.
4. Schritt 3 mit Bild 1 und gewarptem Bild wiederholen bis originales Level erreicht ist.
5. Fluss u und v ist komplett berechnet.

3.3.7 Diskretisierung

Für die Berechnung und Implementierung des Funktionals durch ein numerisches Verfahren muss dieses von der kontinuierlichen Iterationsgleichung zuerst in eine diskrete Gleichung umwandelt werden. Dies ist notwendig, da Funktionen an unendlich vielen Punkten definiert sind, Computer allerdings nur eine begrenzte Anzahl an Speicher zur Verfügung haben. Zu diesem Zweck wird eine diskrete Teilmenge des kontinuierlichen Funktionals erstellt, welche nur an bestimmten Punkten definiert ist. In diesem speziellen Fall bilden diese Punkte die Koordinaten oder Pixel der Bilder. Im folgenden Abschnitt werden deshalb alle vorkommenden Funktionen und Variablen auf das Gitter der Bilder angepasst. Angefangen mit den bisher bekannten Gleichungen:

$$
\begin{aligned}
0 &= \Psi'_D \cdot (J^k_{11} du^k + J^k_{12} dv^k + J^k_{13}) - \alpha(\operatorname{div}(\Psi'_S \nabla u^k) + \operatorname{div}(\Psi'_S \nabla du^k)), \\
0 &= \Psi'_D \cdot (J^k_{12} du^k + J^k_{22} dv^k + J^k_{23}) - \alpha(\operatorname{div}(\Psi'_S \nabla v^k) + \operatorname{div}(\Psi'_S \nabla dv^k)).
\end{aligned} \tag{3.47}
$$

Finite Differenzen

Für die Approximation von Ableitungen auf einem Gitter werden finite Differenzen (siehe Abschnitt 2.1.3) zur Hilfe genommen. Hierbei handelt es sich um Gleichungen, welche die Steigung zwischen zwei Punkten durch deren Differenz bilden. Typischerweise stehen dabei drei Varianten zur Verfügung:

Vorwärtsdifferenz Die Vorwärtsdifferenz bildet die Ableitung durch den Funktionswert an der Stelle x und zieht diesen von dem um h verschobenen Funktionswert ab. Am Ende wird durch den Abstand der beiden Punkte dividiert. Die Vorwärtsdifferenz betrachtet somit die Steigung, die sich rechts von x befindet und wird wie folgt berechnet:

$$
f'(x) = \frac{f(x+h) - f(x)}{h}. \tag{3.48}
$$

Rückwärtsdifferenz Die Rückwärtsdifferenz bildet die Ableitung wiederum durch die Differenz des Funktionswerts von x, wobei der zweite Punkt die Verschiebung um h in entgegengesetzter Richtung markiert. Am Ende wird ebenfalls durch den Abstand dividiert. Die Rückwärtsdifferenz betrachtet damit die Steigung, die sich links von x befindet und wird wie folgt bestimmt:

$$
f'(x) = \frac{f(x) - f(x-h)}{h}. \tag{3.49}
$$

Zentrale Differenz Die zentrale Differenz ist eine Mischung aus beiden vorherigen Varianten. Sie betrachtet die Differenz zweier Punkte, welche jeweils um h verschoben sind. Dabei bezieht sie die Information beider Seiten mit ein, was zu einer genaueren Schätzung am Punkt x führt. Weil Schritte in beiden Richtungen betrachtet werden, muss der Abstand der Punkte auf $2h$ erhöht werden:

$$
f'(x) = \frac{f(x+h) - f(x-h)}{2h}. \tag{3.50}
$$

Diskretisierung der Komponenten

Bei näherer Betrachtung der Euler-Lagrange-Gleichungen, heben sich 4 Komponenten hervor, welche Schritt für Schritt diskretisiert werden müssen. Für die Diskretisierung der Flussvariablen u, v und du, dv ist nicht viel Aufwand erforderlich, es reicht aus die Funktionen an den entsprechenden Pixel auszuwerten:

$$u_{i,j} = u(i \cdot h_x, j \cdot h_y), \tag{3.51}$$
$$v_{i,j} = v(i \cdot h_x, j \cdot h_y), \tag{3.52}$$
$$du_{i,j} = du(i \cdot h_x, j \cdot h_y), \tag{3.53}$$
$$dv_{i,j} = dv(i \cdot h_x, j \cdot h_y), \tag{3.54}$$

wobei sich das Gitter h_x und h_y je nach Auflösung der Level verändert. Anschließend müssen die Einträge des Bewegungstensors J berechnet werden. Dieser stellt sich aus den partiellen Ableitungen f_x, f_y und f_t zusammen, welche wiederum f benötigen. Die Diskretisierung von f funktioniert nach dem gleichen Schema wie es bei den Flussvariablen der Fall war:

$$f_{i,j,t} = f(i \cdot h_x, j \cdot h_y, t), \tag{3.55}$$
$$f_{i,j,t+1} = f(i \cdot h_x, j \cdot h_y, t+1). \tag{3.56}$$

Für die Tensoreinträge wird die zuvor behandelte zentrale Differenz verwendet, da sie von den drei Varianten die genauesten Ergebnisse liefert:

$$[f_x]_{i,j} = \frac{1}{2}\left(\frac{f_{i+1,j,t+1} - f_{i-1,j,t+1}}{2h_x}\right) + \left(\frac{f_{i+1,j,t} - f_{i-1,j,t}}{2h_x}\right), \tag{3.57}$$
$$[f_y]_{i,j} = \frac{1}{2}\left(\frac{f_{i,j+1,t+1} - f_{i,j-1,t+1}}{2h_y}\right) + \left(\frac{f_{i,j+1,t} - f_{i,j-1,t}}{2h_y}\right), \tag{3.58}$$
$$[f_t]_{i,j} = f_{i,j,t+1} - f_{i,j,t}. \tag{3.59}$$

Zur robusteren Approximation wurde die durchschnittliche zentrale Differenz beider Bilder genommen. Die einzelnen Einträge des Tensors lassen sich anschließend durch Multiplikation der partiellen Ableitungen definieren:

$$\begin{aligned}
[J^k_{11}]_{i,j} &= [f_x^2]_{i,j}, \\
[J^k_{12}]_{i,j} &= [f_x]_{i,j}[f_y]_{i,j}, \\
[J^k_{13}]_{i,j} &= [f_x]_{i,j}[f_t]_{i,j}, \\
[J^k_{22}]_{i,j} &= [f_y^2]_{i,j}, \\
[J^k_{23}]_{i,j} &= [f_y]_{i,j}[f_t]_{i,j}.
\end{aligned} \tag{3.60}$$

Der Faktor Ψ'_D kann analog zu den bisherigen Funktionen an den Pixel ausgewertet werden, da alle Komponenten der Funktion bereits diskretisiert wurden:

$$[\Psi'_D]_{i,j} = \Psi'_D \left(\begin{pmatrix} du^k_{i,j} \\ dv^k_{i,j} \\ 1 \end{pmatrix}^T \begin{pmatrix} [J^k_{11}]_{i,j} & [J^k_{12}]_{i,j} & [J^k_{13}]_{i,j} \\ [J^k_{21}]_{i,j} & [J^k_{22}]_{i,j} & [J^k_{23}]_{i,j} \\ [J^k_{31}]_{i,j} & [J^k_{32}]_{i,j} & [J^k_{33}]_{i,j} \end{pmatrix} \begin{pmatrix} du^k_{i,j} \\ dv^k_{i,j} \\ 1 \end{pmatrix} \right). \tag{3.61}$$

Der Glattheitsterm kann zunächst allgemeiner umgestellt werden. In späteren Abschnitten werden weitere Modelle folgen, die durch diese Formulierung einfacher dargestellt werden können, da sie alle Varianten auf eine einheitliche Form bringt:

$$\begin{aligned} 0 &= \Psi'_D \cdot (J^k_{11} du^k + J^k_{12} dv^k + J^k_{13}) - \alpha(\operatorname{div}(D\nabla u^k) + \operatorname{div}(D\nabla du^k)), \\ 0 &= \Psi'_D \cdot (J^k_{12} du^k + J^k_{22} dv^k + J^k_{23}) - \alpha(\operatorname{div}(D\nabla v^k) + \operatorname{div}(D\nabla dv^k)). \end{aligned} \tag{3.62}$$

Dazu wird der skalare Wert Ψ'_S durch den Diffusionstensor D ersetzt, welcher eine positiv definite Matrix darstellt. So können spätere Erweiterungen allein durch Veränderung von D ausgedrückt werden. In diesem Beispiel nimmt D folgende Form an:

$$D = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \Psi'_S(|\nabla(u^k + du^k)|^2 + |\nabla(v^k + dv^k|^2))\, I, \tag{3.63}$$

wobei der vordere Teil den bereits bekannten Faktor Ψ'_S darstellt, welcher mit der Einheitsmatrix I der Größe 2×2 multipliziert wird. Da die Diskretisierung für alle 4 Terme gleich abläuft, wird der Ablauf nur anhand von du dargestellt. Für die anderen Flussvariablen lässt sich die Vorgehensweise dementsprechend übertragen. Für den ersten Schritt muss die Divergenz aufgelöst und D mit dem Gradienten multipliziert werden, wobei die Einträge b abseits der Hauptdiagonale wegfallen:

$$\operatorname{div}(D\,\nabla du^k) = \frac{\partial}{\partial x}(a\, du^k_x) + \frac{\partial}{\partial y}(c\, du^k_y). \tag{3.64}$$

Anschließend folgt die Approximation der inneren und äußeren Ableitung durch zentrale Differenzen. Da für die vollständige Diskretisierung zwei ineinander geschachtelte Ableitungen approximiert werden müssen, wird der Abstand pro Ableitung auf die Hälfte reduziert, sodass später ein einheitlicher Abstand von einem Pixel entsteht. Die Gleichungen sieht nach der ersten Ableitung wie folgt aus:

$$\begin{aligned} &\frac{(a\, du^k_x)_{i+\frac{1}{2},j} - (a\, du^k_x)_{i-\frac{1}{2},j}}{h_x} + \frac{(c\, du^k_y)_{i,j+\frac{1}{2}} - (c\, du^k_y)_{i,j-\frac{1}{2}}}{h_y} \\ = &\frac{a_{i+\frac{1}{2},j}\, [du^k_x]_{i+\frac{1}{2},j} - a_{i-\frac{1}{2},j}\, [du^k_x]_{i-\frac{1}{2},j}}{h_x} + \frac{c_{i,j+\frac{1}{2}}\, [du^k_y]_{i,j+\frac{1}{2}} - c_{i,j-\frac{1}{2}}\, [du^k_y]_{i,j-\frac{1}{2}}}{h_y}. \end{aligned} \tag{3.65}$$

Im nächsten Schritt wird die zweite zentrale Differenz für die Ableitungen du_x^k und du_y^k vorgenommen:

$$\begin{aligned} &\frac{a_{i+\frac{1}{2},j}\,\frac{du_{i+1,j}^k - du_{i,j}^k}{h_x} - a_{i-\frac{1}{2},j}\,\frac{du_{i,j}^k - du_{i-1,j}^k}{h_x}}{h_x} \\ +&\frac{c_{i,j+\frac{1}{2}}\,\frac{du_{i,j+1}^k - du_{i,j}^k}{h_y} - c_{i,j-\frac{1}{2}}\,\frac{du_{i,j}^k - du_{i,j-1}^k}{h_y}}{h_y}. \end{aligned} \tag{3.66}$$

Zuletzt wird das Minus in den Term hineingezogen und a, c an den Pixelzwischenräumen ausgewertet. Da Pixelzwischenräume allerdings nicht definiert sind und somit keine Werte existieren, wird der Durchschnitt der zwei benachbarten Pixel genommen. Da $[\Psi'_S] = a = c$ erhält man schließlich:

$$\begin{aligned} &\frac{[\Psi'_S]_{i+1,j} + [\Psi'_S]_{i,j}}{2} \cdot \frac{du_{i+1,j}^k - du_{i,j}^k}{h_x} \\ +&\frac{[\Psi'_S]_{i-1,j} + [\Psi'_S]_{i,j}}{2} \cdot \frac{du_{i-1,j}^k - du_{i,j}^k}{h_x^2} \\ +&\frac{[\Psi'_S]_{i,j+1} + [\Psi'_S]_{i,j}}{2} \cdot \frac{du_{i,j+1}^k - du_{i,j}^k}{h_y^2} \\ +&\frac{[\Psi'_S]_{i,j-1} + [\Psi'_S]_{i,j}}{2} \cdot \frac{du_{i,j-1}^k - du_{i,j}^k}{h_y^2}. \end{aligned} \tag{3.67}$$

Führt man die Diskretisierung für alle Terme durch ist zu erkennen, dass die Terme die Summe der Differenz des zentralen Pixel i, j zu den 4 Nachbarn darstellen. Dies lässt sich leicht durch eine Schablone auf die jeweiligen Pixel anwenden, wobei die Zellen für die Pixel an den jeweiligen Positionen des Flusses u und v stehen und jeweils mit den durchschnittlichen Werten von Ψ' multipliziert werden. Die berechnete Schablone des Glattheitsterms ist Tabelle 3.1 zu entnehmen.

	$\frac{[\Psi'_S]_{i,j+1}+[\Psi'_S]_{i,j}}{2}$	
$\frac{[\Psi'_S]_{i-1,j}+[\Psi'_S]_{i,j}}{2}$	$-\frac{[\Psi'_S]_{i+1,j}+[\Psi'_S]_{i,j}}{2}$ $-\frac{[\Psi'_S]_{i-1,j}+[\Psi'_S]_{i,j}}{2}$ $-\frac{[\Psi'_S]_{i,j+1}+[\Psi'_S]_{i,j}}{2}$ $-\frac{[\Psi'_S]_{i,j-1}+[\Psi'_S]_{i,j}}{2}$	$\frac{[\Psi'_S]_{i+1,j}+[\Psi'_S]_{i,j}}{2}$
	$\frac{[\Psi'_S]_{i,j-1}+[\Psi'_S]_{i,j}}{2}$	

Tabelle 3.1: Schablone des Glattheitsterm mit den jeweiligen Gewichten der Pixel.

Durch das Einfügen in die Gleichung , erhält man somit die finalen diskreten Euler-Lagrange-Gleichungen. Die 4 Terme des Glattheitsterms lassen sich in zwei Summen zusammenfassen, wobei $\sum_{l \in x,y}$ für je zwei Terme von x und y stehen und $\sum_{(\bar{i},\bar{j}) \in N_l(i,j)}$ die Nachbarn $\bar{i}, \bar{j}$ von i, j in Richtung l definieren:

$$
\begin{aligned}
0 =& [\Psi'_D]_{i,j} \cdot ([J^k_{11}]_{i,j} du^k_{i,j} + [J^k_{12}]_{i,j} dv^k_{i,j} [J^k_{13}]_{i,j}) \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}} + [\Psi'_S]_{i,j}}{2} \cdot \frac{u^k_{\bar{i},\bar{j}} - u^k_{i,j}}{h_l^2} \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}} + [\Psi'_S]_{i,j}}{2} \cdot \frac{du^k_{\bar{i},\bar{j}} - du^k_{i,j}}{h_l^2}, \\
0 =& [\Psi'_D]_{i,j} \cdot ([J^k_{12}]_{i,j} du^k_{i,j} + [J^k_{22}]_{i,j} dv^k_{i,j} [J^k_{23}]_{i,j}) \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}} + [\Psi'_S]_{i,j}}{2} \cdot \frac{v^k_{\bar{i},\bar{j}} - v^k_{i,j}}{h_l^2} \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}} + [\Psi'_S]_{i,j}}{2} \cdot \frac{dv^k_{\bar{i},\bar{j}} - dv^k_{i,j}}{h_l^2}.
\end{aligned}
\tag{3.68}
$$

3.3.8 Lösung

In diesem Abschnitt wird der Algorithmus hergeleitet, der zur Lösung benötigt wird. Da für jeden Pixel zwei Gleichungen vorliegen, welche alle gemeinsam unter Beachtung der anderen Gleichungen gelöst werden müssen, lässt sich das Problem als Gleichungssystem formulieren, das der Anzahl an Pixel $2 \cdot x \cdot y$ entspricht. Verfahren wie der Gaußalgorithmus lassen sich bei so einer Größenordnung allerdings nicht mehr verwenden, da er eine Laufzeit von $O(n^3)$ besitzt, die mit der Anzahl an Pixel kubisch ansteigt. Aus diesem Grund werden hierfür Iterationsverfahren benutzt, die auf bereits berechneten Werten aufbauen und sich der Lösung schrittweise annähern. Die Berechnung in dieser Arbeit wird durch ein sogenanntes Splitting-Verfahren gelöst, welches durch geschicktes Trennen der Matrix mit einem Anfangsvektor zum Ergebnis konvergiert und nach einer hinreichenden Anzahl an Iterationsschritten abgebrochen wird.

Lagged Non-linearity

Da beide Ψ-Funktionen von u und v abhängen, rücken die Flussvariablen bei der Ableitung in den Nenner wodurch ein nicht-lineares Gleichungssystem entsteht. Dies ist jedoch wegen der hohen Komplexität und des daraus folgenden rechnerischen Aufwands nicht anwendbar. Zur Lösung dieses Problems, wird das nicht-lineare Gleichungssystem als Serie von linearen Gleichungssystemen gelöst, indem die Faktoren Ψ' im Iterationsschritt m vorberechnet werden und anschließend für eine bestimmte Anzahl an Iterationen eingefroren werden. Dadurch entsteht eine ungenaue Lösung eines

linearen Gleichungssystems, das für die Konvergenz allerdings ausreichend ist. Eingesetzt in die bisherige Gleichung ergibt sich folgender Schritt:

$$
\begin{aligned}
0 =& [\Psi'_D]^m_{i,j} \cdot ([J^k_{11}]_{i,j} du^{k,m+1}_{i,j} + [J^k_{12}]_{i,j} dv^{k,m+1}_{i,j} [J^k_{13}]_{i,j}) \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m+1}_{\bar{i},\bar{j}} - u^{k,m+1}_{i,j}}{h_l^2} \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{du^{k,m+1}_{\bar{i},\bar{j}} - du^{k,m+1}_{i,j}}{h_l^2}, \\
0 =& [\Psi'_D]^m_{i,j} \cdot ([J_{12}]_{i,j} du^{k,m+1}_{i,j} + [J^k_{22}]_{i,j} dv^{k,m+1}_{i,j} [J^k_{23}]_{i,j}) \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{v^{k,m+1}_{\bar{i},\bar{j}} - v^{k,m+1}_{i,j}}{h_l^2} \\
&- \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{dv^{k,m+1}_{\bar{i},\bar{j}} - dv^{k,m+1}_{i,j}}{h_l^2}.
\end{aligned}
\tag{3.69}
$$

Jacobi-Verfahren

Das Jacobi-Verfahren ist ein Splitting-Verfahren, das die Matrix eines Gleichungssystems in zwei Bereiche aufteilt und anschließend durch eine Fixpunktiteration sich der Lösung annähert. Angefangen mit der Matrix-Vektor Schreibweise

$$
Ap = q. \tag{3.70}
$$

Es trennt die Matrix zum einen in die Diagonalmatrix A_1 und den Rest A_2 auf. Dies hat den Hintergrund, dass A_1 zur Bestimmung von p im weiteren Verlauf invertiert werden muss und Diagonalmatrizen sehr leicht zu invertieren sind. Die Matrix kann somit durch folgende Teilmatrizen dargestellt werden:

$$
A = A_1 + A_2. \tag{3.71}
$$

Im Anschluss kann die Substitution eingefügt, die Gleichung nach p umgestellt und die Fixpunktiteration angewendet werden:

$$
\begin{aligned}
(A_1 + A_2)p &= q \\
A_1 p + A_2 p &= q \\
A_1 p &= q - A_2 p \\
p^{n+1} &= A_1^{-1}(q - A_2 p^n).
\end{aligned}
\tag{3.72}
$$

Dieses Schema kann nun auf die diskrete Euler-Lagrange Gleichung angewendet werden. Da beide Gleichungen nach dem exakt selben Prinzip ablaufen, wird die Herleitung nur für du durchgeführt. Die Umstellung für dv erfolgt nach der gleichen Vorgehensweise. Im ersten Schritt werden alle Komponenten, welche nicht von du abhängen auf die andere Seite gebracht um auf die $Ap = q$ Form zu kommen:

$$\begin{aligned}
&[\Psi'_D]^m_{i,j}[J^k_{11}]_{i,j}du^{k,m+1}_{i,j} - \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{du^{k,m+1}_{\bar{i},\bar{j}} - du^{k,m+1}_{i,j}}{h_l^2} \\
&= -[\Psi'_D]^m_{i,j}[J^k_{12}]_{i,j}dv^{k,m+1} - [\Psi'_D]^m_{i,j}[J^k_{13}]_{i,j} \\
&\quad + \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m+1}_{\bar{i},\bar{j}} - u^{k,m+1}_{i,j}}{h_l^2}.
\end{aligned} \tag{3.73}$$

Im nächsten Schritt wird die Matrix zerlegt, der nicht-diagonale Teil auf die rechte Seite gebracht und du^k ausgeklammert:

$$\begin{aligned}
&\left([\Psi'_D]^m_{i,j}[J^k_{11}]_{i,j} + \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{1}{h_l^2}\right) \cdot du^{k,m+1}_{i,j} \\
&= -[\Psi'_D]^m_{i,j}[J^k_{12}]_{i,j}dv^{k,m+1} - [\Psi'_D]^m_{i,j}[J^k_{13}]_{i,j} \\
&\quad + \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m+1}_{\bar{i},\bar{j}} + du^{k,m+1}_{\bar{i},\bar{j}} - u^{k,m+1}_{i,j}}{h_l^2}.
\end{aligned} \tag{3.74}$$

Zuletzt wird der diagonale Teil durch Multiplikation mit der Inversen auf die andere Seite gebracht:

$$du^{k,m,n+1}_{i,j} = \frac{\xi_1 + \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \xi_2}{[\Psi'_D]^m_{i,j}[J^k_{11}]_{i,j} + \alpha \sum_{l\in x,y} \sum_{(\bar{i},\bar{j})\in N_l(i,j)} \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{1}{h_l^2}} \tag{3.75}$$

mit

$$\begin{aligned}
\xi_1 &= -[\Psi'_D]^m_{i,j}[J^k_{12}]_{i,j}dv^{k,m,n} - [\Psi'_D]^m_{i,j}[J^k_{13}]_{i,j}, \\
\xi_2 &= \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m,n}_{\bar{i},\bar{j}} + du^{k,m,n}_{\bar{i},\bar{j}} - u^{k,m,n}_{i,j}}{h_l^2}.
\end{aligned} \tag{3.76}$$

Gauß-Seidel-Verfahren

Die Gauß-Seidel-Methode ist eine Erweiterung der Jacobi-Methode. Da die Diagonalmatrix nur eine ungenaue Approximierung der ursprünglichen Matrix ist, wird zusätzlich der untere Teil der Matrix A zu A_1 hinzugefügt. Dadurch entsteht eine Dreiecksmatrix, welche immer noch recht simpel zu invertieren ist und die Laufzeit der Konvergenz ca. um den Faktor zwei verbessert:

$$du_{i,j}^{k,m,n+1} = \frac{\xi_1 + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l^-(i,j)} \xi_2 + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l^+(i,j)} \xi_3}{[\Psi'_D]_{i,j}^m [J_{11}^k]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}}^m + [\Psi'_S]_{i,j}^m}{2} \cdot \frac{1}{h_l^2}} \tag{3.77}$$

mit

$$\xi_1 = -[\Psi'_D]_{i,j}^m [J_{12}^k]_{i,j} dv^{k,m,n} - [\Psi'_D]_{i,j}^m [J_{13}^k]_{i,j}, \tag{3.78}$$

$$\xi_2 = \frac{[\Psi'_S]_{\bar{i},\bar{j}}^m + [\Psi'_S]_{i,j}^m}{2} \cdot \frac{u_{\bar{i},\bar{j}}^{k,m,n+1} + du_{\bar{i},\bar{j}}^{k,m,n+1} - u_{i,j}^{k,m,n+1}}{h_l^2}, \tag{3.79}$$

$$\xi_3 = \frac{[\Psi'_S]_{\bar{i},\bar{j}}^m + [\Psi'_S]_{i,j}^m}{2} \cdot \frac{u_{\bar{i},\bar{j}}^{k,m,n} + du_{\bar{i},\bar{j}}^{k,m,n} - u_{i,j}^{k,m,n}}{h_l^2}, \tag{3.80}$$

wobei N^- die Komponenten der unteren Dreiecksmatrix darstellen und bereits für Iterationsschritt $n+1$ berechnet wurden. N^+ steht für die Komponenten der oberen Dreiecksmatrix, welche noch nicht erreicht wurden und deshalb noch im Iterationsschritt n sind. Zusätzlich wird Speicher in der Größe von du und dv eingespart, da die neu berechneten Werte direkt in die aktuelle Iteration mit einfließen, wodurch eine zusätzliche Speicherung der neuen Werte überflüssig wird.

Successive Over-Relaxation-Verfahren

Der SOR-Löser [Mei11] ist der finale Löser und eine Erweiterung der Gauß-Seidel-Methode, der eine noch schnellere Konvergenz aufweist. Dieser ergibt sich mittels überrelaxation der Gauß-Seidel-Methode der Form:

$$du_{i,j}^{k,m,n+1} = (1-\omega) du_{i,j}^{k,m,n} + \omega \left(\frac{\xi_1 + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l^-(i,j)} \xi_2 + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l^+(i,j)} \xi_3}{[\Psi'_D]_{i,j}^m [J_{11}^k]_{i,j} + \alpha \sum_{l \in x,y} \sum_{(\bar{i},\bar{j}) \in N_l(i,j)} \frac{[\Psi'_S]_{\bar{i},\bar{j}}^m + [\Psi'_S]_{i,j}^m}{2} \cdot \frac{1}{h_l^2}} \right) \tag{3.81}$$

mit

$$\xi_1 = -[\Psi'_D]^m_{i,j}[J^k_{12}]_{i,j}dv^{k,m,n} - [\Psi'_D]^m_{i,j}[J^k_{13}]_{i,j}, \tag{3.82}$$

$$\xi_2 = \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m,n+1}_{\bar{i},\bar{j}} + du^{k,m,n+1}_{\bar{i},\bar{j}} - u^{k,m,n+1}_{i,j}}{h_l^2}, \tag{3.83}$$

$$\xi_3 = \frac{[\Psi'_S]^m_{\bar{i},\bar{j}} + [\Psi'_S]^m_{i,j}}{2} \cdot \frac{u^{k,m,n}_{\bar{i},\bar{j}} + du^{k,m,n}_{\bar{i},\bar{j}} - u^{k,m,n}_{i,j}}{h_l^2}. \tag{3.84}$$

Der Fall $\omega = 1, \omega \in (0, 2)$ beschreibt den speziellen Fall, wodurch wieder das Gauß-Seidel-Verfahren entsteht.

4 Segmentierungsgestütze Methoden

Im nächsten Kapitel werden segmentbasierte Algorithmen besprochen, die auf der zuvor behandelten Grundlage von Horn und Schunck aufbauen. Die Idee hinter diesen Methoden ist die Verschiebung nicht länger für jeden Pixel einzeln zu betrachten, sondern jedes auf den Bildern dargestellte Objekt als Ganzes zu sehen. Dadurch entsteht eine drastische Reduzierung der Variablen, was besonders Einfluss auf die Laufzeit und die Berechnung der Kanten durch festgelegte Grenzen hat. Zuerst wird jedoch eine Einführung in die Segmentierung gegeben und anhand des Mean-Shift-Algorithmus durchgeführt, welcher anschließend durch ein Region-Merging-Verfahren erweitert wird. Danach werden die einzelnen Methoden zur Bestimmung des Flusses hergeleitet.

4.1 Segmentierung

Unter der Segmentierung eines Bildes versteht man das Zusammenfassen einzelner Pixel zu inhaltlich zusammenhängenden Regionen. Das Ziel dabei ist es, die Darstellung der Bilder so zu vereinfachen, dass die Objekte der Bilder leichter zu erkennen und zu verarbeiten sind. Im optimalen Fall bestünde das Bild aus zusammenhängenden Regionen, wobei jeder Pixel einer Region zugeordnet ist, die für ein Objekt steht. Dies nennt man eine überdeckungsfreie, vollständige und zusammenhängende Segmentierung. Einige Anwendungsgebiete sind beispielsweise die Klassifizierung von Objekten, Gesichts- und Schrifterkennung oder auch die Komprimierung von Videodateien. Eine mögliche Segmentierung ist in Abbildung 4.1 zu sehen.

Abbildung 4.1: Segmentierung des ersten Bildes der Hydrangea Testsequenz. **Links:** Originales Bild. **Rechts:** Segmentiertes Bild.

4.1.1 Mean-Shift Verfahren

Das Mean-Shift Verfahren [CM02] ist eine vielseitig anwendbare Methode zur Bestimmung der Dichtemaxima einer Punktmenge. Häufige Anwendung findet es als Clusteralgortihmus zur Gruppierung von Datensätzen oder zur visuellen Verfolgung von Objekten, es liefert jedoch auch hervorragende Ergebnisse bei der Segmentierung von Bildern. Dazu wird das sogenannte Parzen-Fenster zur Hilfe genommen.

Parzen-Fenster

Die Parzen-Fenster-Methode ist eine Kerndichteschätzung zur Bestimmung der Wahrscheinlichkeit von einer Zufallsgröße:

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{x-x_i}. \tag{4.1}$$

wobei die Dichte durch die Summe der Punkte x_i in einer bestimmten Region definiert wird. Herkömmliche Verfahren wie Erstellung eines Histogramms liefern jedoch nur unzureichend genaue diskrete Ergebnisse, da oft von einer stetigen Verteilung ausgegangen werden kann. Um eine kontinuierliche Funktion zu erzeugen, wird deshalb für jeden Punkt x ein Fenster der Größe h erzeugt, das alle Punkte x_i anhand des Abstands gewichtet. Dies führt zu einer Erweiterung von Gleichung 4.1:

$$p(x) = \frac{1}{nh^d} \sum_{i=1}^{n} \varphi \left(\left\| \frac{x - x_i}{h} \right\|^2 \right). \tag{4.2}$$

Seien n die Anzahl gegebener Punkte und d die Dimensionen. Dann wird der räumliche Support jenes Punktes x_i auf ein Fenster der Größe h erweitert. Die Warscheinlichkeitsdichte wird dann nicht mehr durch die Summe von Punkten sondern durch eine Summe von Fenstern dargestellt. Eine Veranschaulichung des Fenster ist in Abbildung 4.2 dargestellt.

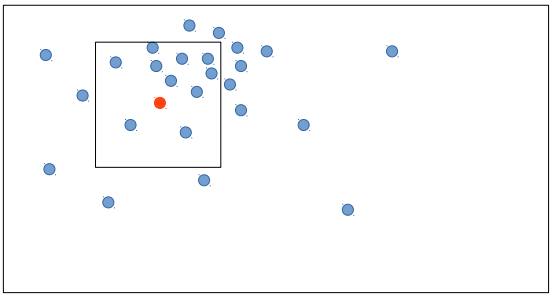


Abbildung 4.2: Beispiel einer zweidimensionalen Verteilung von Punkten mit aktuellem Punkt x (Rot) und dem dazugehörigen Fenster.

Mean-Shift

Bei der Mean-Shift Methode wird jedoch nach der größten Dichte einer Punktemenge gesucht. Die notwendige Bedingung für einen Hochpunkt ist die Ableitung der Dichtefunktion gleich null zu setzen. Durch Anwendung der Kettenregel kann so zuerst die äußere Ableitung von φ bestimmt und mit der inneren Ableitung multipliziert werden. Die Gleichung wird anschließend gleich 0 gesetzt:

$$\nabla f(x) = \frac{1}{nh^{d+2}} \sum_{i=1}^{n} (x - x_i) \varphi' \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) = 0. \tag{4.3}$$

Als nächstes wird die Funktion $g(x)$ definiert, unter der Annahme, dass die Ableitung für alle $x > 0$ existiert:

$$g(x) = -\varphi'(x). \tag{4.4}$$

Durch Einsetzen von $g(x)$ in die Dichtefunktion verändern sich die Vorzeichen, sodass x und x_i im Faktor vertauscht werden. Anschließend kann die Funktion vereinfacht werden, indem die Klammer ausmultipliziert wird und die Faktoren durch Division wegfallen:

$$\begin{aligned}
0 &= \frac{1}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \\
&= \frac{1}{nh^{d+2}} \sum_{i=1}^{n} x_i \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) - \frac{1}{nh^{d+2}} \sum_{i=1}^{n} x \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \\
&= \sum_{i=1}^{n} x_i \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) - \sum_{i=1}^{n} x \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \\
&= \left[\sum_{i=1}^{n} g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right) \right] \left[\frac{\sum_{i=1}^{n} x_i \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right)} - x \right] \\
m_{h,g}(x) &= \frac{\sum_{i=1}^{n} x_i \, g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right)}{\sum_{i=1}^{n} g \left(\left\| \frac{x - x_i}{h} \right\|^2 \right)} - x.
\end{aligned} \tag{4.5}$$

Die dadurch entstandene Gleichung zeigt den Mean-Shift-Vektor. Er stellt die Verschiebung des Punktes x in Richtung der stärksten Steigung der Dichte dar. Der Vektor setzt sich aus der Summe aller Punkte innerhalb des Fensters zusammen. Diese werden jeweils mit ihrem dazugehörigen Gewicht des Abstands zu x multipliziert und anschließend durch die Summe der jeweiligen Gewichte geteilt. Der so errechnete neue Standort des Punktes rückt damit ein Stück näher zu dem gesuchten Hochpunkt. Durch Subtraktion des neuen Punktes mit dem alten Punkt, entsteht der Mean-Shift-Vektor. Eine Darstellung der Iterationen kann aus Abbildung 4.3 entnommen werden.

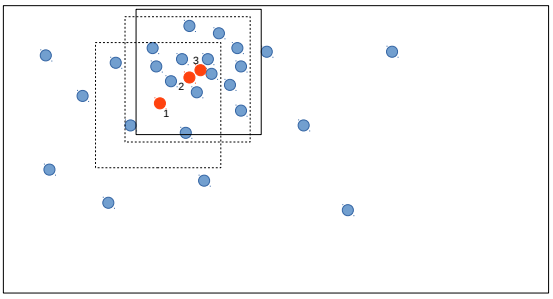


Abbildung 4.3: Schema des Mean-Shift-Verfahrens. Punkt 1 wird durch Analyse seiner Umgebung zur größten Dichte stückweise herangezogen, bis er im Maximum ankommt.

Für die Anwendung auf Farbbilder wird ein fünfdimensionaler Raum betrachtet. Bilder werden typischerweise durch ihre räumliche Komponente x, y repräsentiert, wobei jeder Pixel einen bestimmten Farbwert R, G, B besitzt und jede dieser Komponenten als eigene Dimension aufgefasst wird. Zuvor müssen die Bilder allerdings von dem RGB- in den Lab-Farbraum (siehe Abschnitt 2.2.2) konvertiert werden, da dort gleichartige Farben eine ähnliche Position annehmen und dem Abstand eine größere Aussagekraft in Bezug auf die Ähnlichkeit einer Farbe zukommt. Ein Pixel kann dementsprechend durch folgenden Punkt des fünfdimensionalen Raums dargestellt werden:

$$x_i = \begin{pmatrix} x \\ y \\ L \\ a \\ b \end{pmatrix}. \tag{4.6}$$

Da der Raum jedoch in zwei unterschiedliche Bereiche getrennt ist die inhaltlich nicht miteinander in Verbindung stehen, ist es nicht sinnvoll, diese gleich zu gewichten. Die räumliche Nähe zweier Pixel sagt beispielsweise nichts über die Ähnlichkeit der Farbe aus. Aufgrund dessen wird zur Berechnung des Abstands der fünfdimensionale Raum getrennt und zwei Gewichtungsfunktionen verwendet. Zum einen eine zweidimensionale Funktion für die räumliche Komponente und zum anderen eine dreidimensionale für die farbliche Komponente. Hierfür wird bei beiden Varianten die Gaußfunktion verwendet, welche anschließend miteinander multipliziert werden. Die Fensterfunktion lautet damit:

$$G(h_s, h_r) = g\left(\left\|\frac{x^s}{h_s}\right\|^2\right) \cdot g\left(\left\|\frac{x^r}{h_r}\right\|^2\right) \tag{4.7}$$

wobei x^s eine Koordinate des räumlichen (spatial) Raumes und x^r den farblichen (range) Raum darstellt. Die Variablen h_s und h_r bestimmen die Größe der Fensterfunktion in Raum- bzw. Farbrichtung. In Abbildung 4.4 sind beide Räume mit ihren jeweiligen Gewichtsfunktionen grafisch dargestellt.

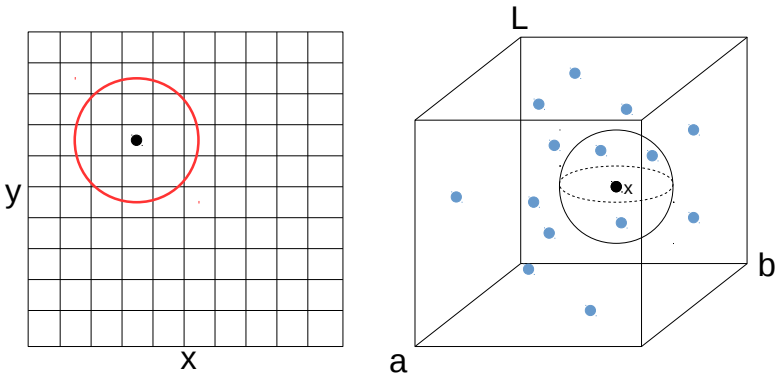


Abbildung 4.4: Darstellung beider Räume. **Links:** Räumliche Komponente mit Gitter der Pixel. **Rechts:** Farbliche Komponente.

Dieser Vorgang wird solange iteriert, bis der Mean-Shift-Vektor eine kleinere Länge als ein definierter Schwellwert besitzt. Dies ist nötig, da die Länge des Vektors sich mit jeder Iteration in Richtung des Grenzwerts verringert und es zu viel Zeit in Anspruch nehmen würde, eine perfekte Lösung zu erzielen. Hat jeder Pixel seinen entgültigen Punkt eingenommen, werden alle Punkte die zum gleichen Dichtemaximum konvergieren zu einem Segment gruppiert. Zur Implementierung wurde eine modifizierte Version des EDISON Systems [CGM02] verwendet. Eine Reihe an Segmentierungen sind in Abbildung 4.5 zu sehen.

4.1.2 Region-Merging

Der Mean-Shift Algorithmus stößt jedoch bei gewissen Situationen an seine Grenzen. Gerade bei Bereichen mit starkem Rauschen ist die farbliche Distanz der Pixel oft zu groß, sodass sie außerhalb des Fensters sind und nicht zu einem Segment zusammengefügt werden. Dies hat zur Folge, dass diese Stellen eine sehr starke Übersegmentierung aufweisen. So kann es durchaus vorkommen, dass dort jeder Pixel sein eigenes Segment bekommt. Natürlich kann die Distanz für den farblichen Raum angehoben werden um die Ausreißer mit einzubeziehen, was jedoch oft zu einem Verlust an Objekten an anderen Stellen führt. In Abbildung 4.6 wurde eine Segmentierung der Grove2 Testsequenz mit $h_s = 7, h_r = 15$ durchgeführt. Zu erkennen ist, dass sich die Steine nur kaum von dem ursprünglichen Bild unterschieden, da dort besonders viel Rauschen vorhanden ist. Die Kantenmap zeigt zudem eine äußerst große Anzahl an Segmenten auf den Steinen. Um dies etwas besser in den Griff zu bekommen, wird die Mean-Shift-Segmentierung um einen Region-Merging-Algorithmus [NZZW10] erweitert, der kleine Segmente nach und nach mit seinen Nachbarn vereint, bis diese eine festgelegte Größe überschreiten. Dazu wird eine Regionen-Adjazenzliste erstellt, die für jedes Segment die zugehörigen Nachbarn abspeichert. Jedem Segment wird anschließend eine Position in der Liste und die aktuelle Segmentnummer zugeordnet. Die Liste wird nun nach zu kleinen Segmenten durchsucht und für den Fall eines Treffers, mit dem Nachbarn vereint, der nach der farblichen Information zufolge die

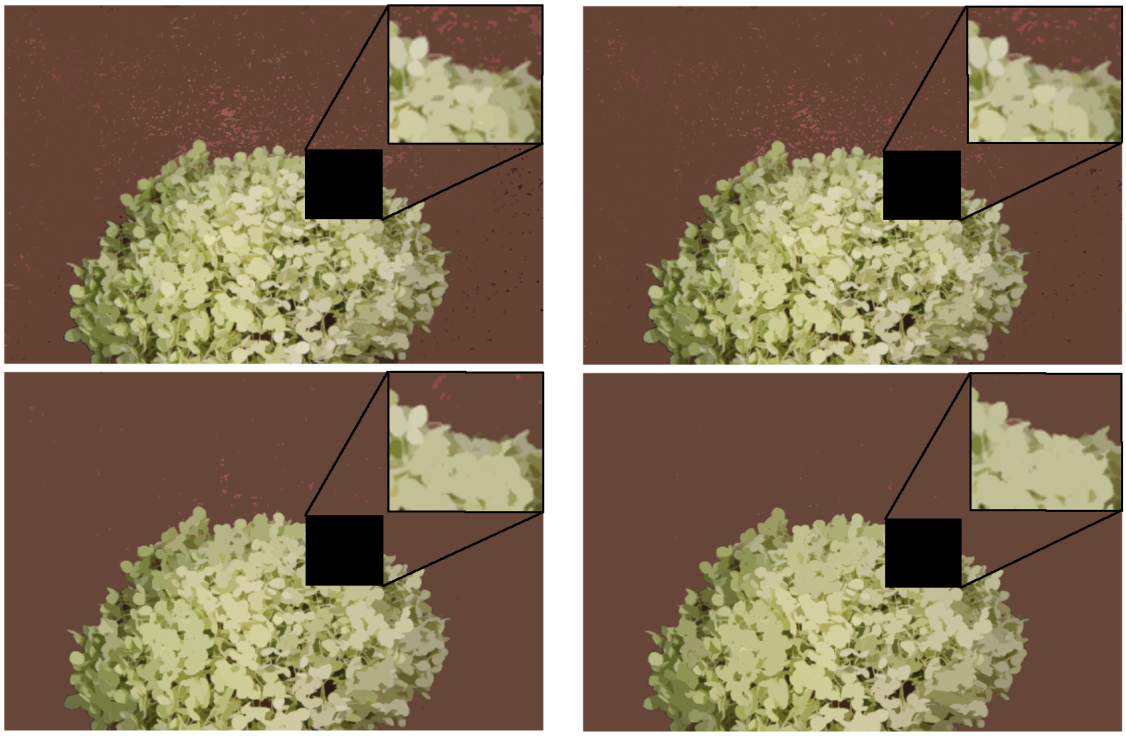


Abbildung 4.5: Segmentierung der Hydrangea Testseqeuenz **Oben links:** $h_s = 7, h_r = 7$. **Oben rechts:** $h_s = 15, h_r = 7$, **Unten links:** $h_s = 7, h_r = 15$, **Unten rechts:** $h_s = 15, h_r = 15$

größte Ähnlichkeit aufweist. Am Ende der Iteration wird eine transitive Hülle erstellt, da Segmente oft durch mehrere Vereinigungen erzeugt wurden und das neue Segment vielleicht bereits mit einem anderen Segment verschmolzen wurde. So entsteht eine Liste die jedem kleinen Segment die entgültige Segmentnummer zuordnet, in welcher es aufgenommen wird. Dies wird solange wiederholt bis keine kleinen Segment mehr vorhanden sind. Abbildung 4.7 zeigt eine anschauliche Darstellung einer kleinen Liste mit 10 Segmenten und ihren jeweiligen Nachbarn, in der die erste Iteration zu sehen ist.

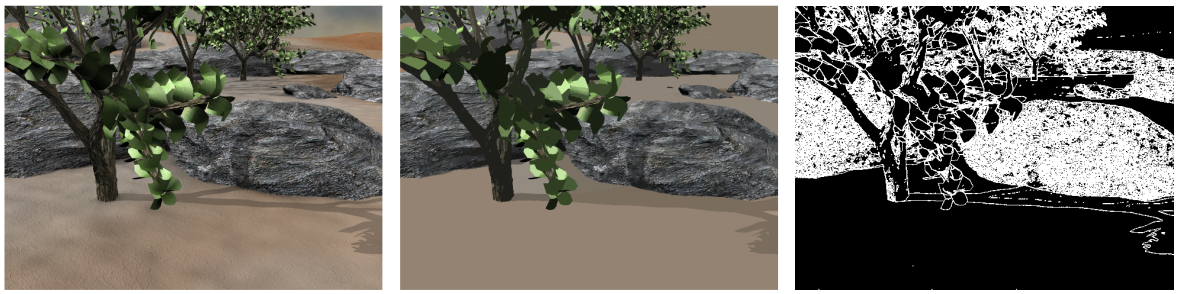


Abbildung 4.6: Segmentierung der Grove2 Testsequenz. **Links:** Originalbild, **mitte:** Segmentiertes Bild, **rechts:** Kantenmap der Segmentierung mit weißen Kanten. Bei den Steinen bildet sich eine starke Übersegmentierung.

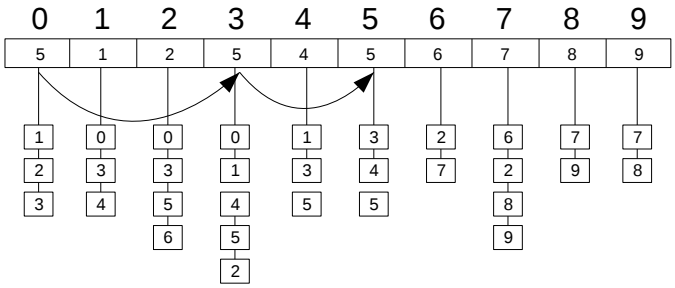


Abbildung 4.7: Schema des Region-Merging Algorithmus. Segment 0 und 3 stellen kleine Segmente dar. Zuerst wird 0 von 3 aufgenommen, welches anschließend mit 5 vereint wird.

Die obere Zahlenreihe zeigt hier die ursprüngliche Segmentnummer, darunter ist die aktuelle Segmentnummer zu sehen, die durch Vereinigung mit den Nachbarn entstanden ist. Nachfolgend in Abbildung 4.8 sind ein paar Ergebnisse des Region-Merging-Verfahrens zu sehen, die an der Grove2 Testsequenz durchgeführt wurden:

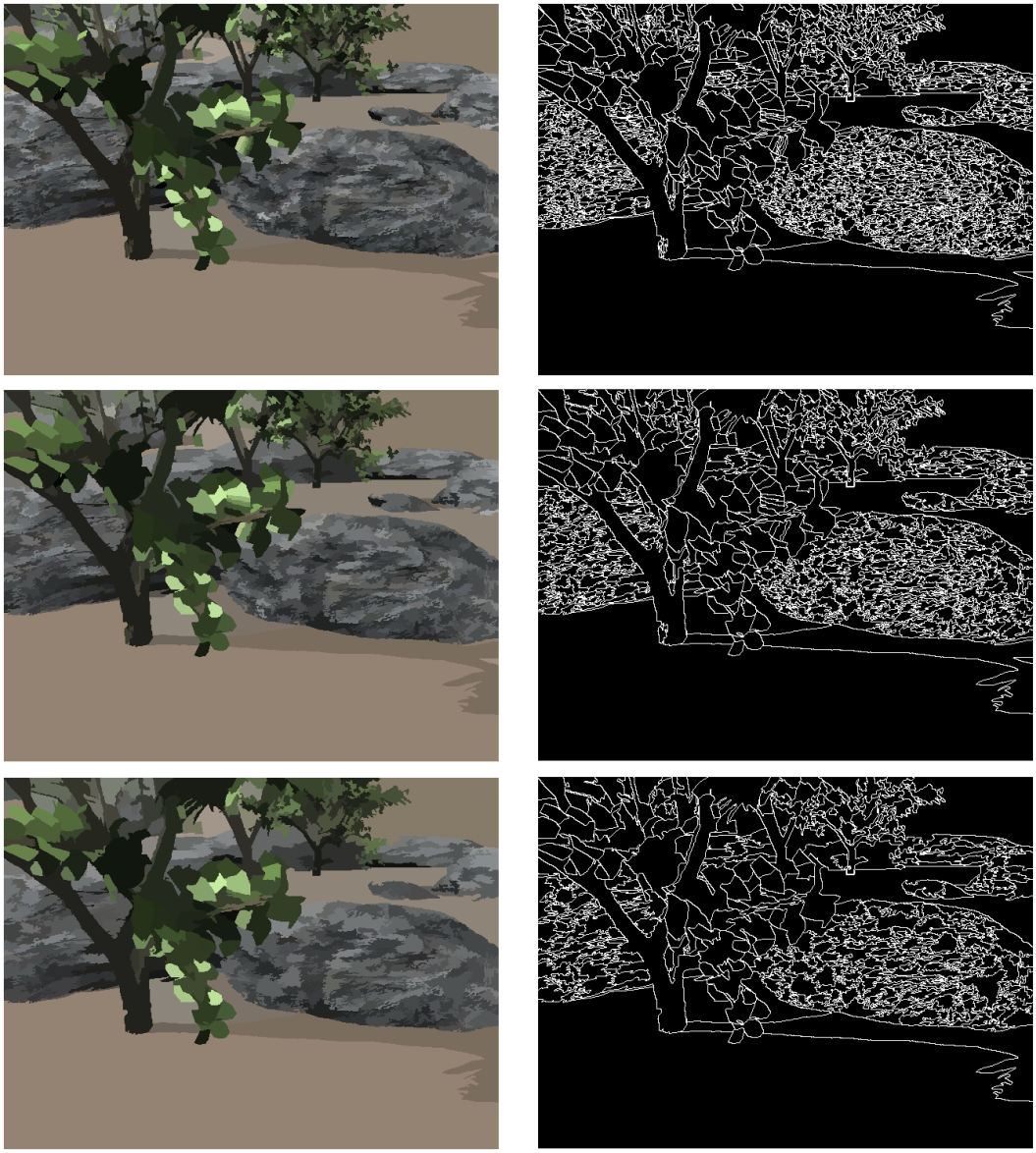


Abbildung 4.8: Von oben nach unten: Grove2 Testsequenz mit Filterung aller Segmente die kleiner als 50, 100, 200 Pixel sind.

4.2 Methoden

Für die nachfolgenden Methoden wurde ein segmentbasiertes Flussmodell erstellt, das den Fluss nicht länger für jeden einzelnen Pixel bestimmt, sondern diesen anhand inhaltlicher Objekte des Bildes ermittelt. Dazu wird durch das Mean-Shift Verfahren eine Segmentierung von Bild 1 berechnet und anschließend Rauschen und kleine Segmente durch Region-Merging entfernt. Danach wird jedem Pixel eine Segmentnummer zugeordnet. Dabei wird die Verschiebung ähnlich dem Basisverfahren mit Daten- und Glattheitsterm ermittelt, mit dem Unterschied, dass nun für jedes Segment eine Flussvariable berechnet wird, welche später auf die Pixel innerhalb des Segments übertragen wird. Nach dieser Vorgehensweise wurden zwei verschiedenen Varianten entwickelt.

4.2.1 Konstanter Segmentfluss

Das erste Verfahren bestimmt den Fluss über konstante Verschiebung der Segmente und kommt dem Modell des Basisverfahrens recht nahe. Hier findet eine direkte Übertragung der Terme statt, wobei eine Anpassung auf eine segmentbasierte Berechnung erfolgt. Dabei wird für alle Segmente des Bildes der Fluss der Segmente berechnet und anschließend direkt auf die zugehörigen Pixel innerhalb des Segments übertragen:

$$\begin{aligned} u_{i,j} &= u_l \ \text{ mit } \ i,j \in \Omega_l, \\ v_{i,j} &= v_l \ \text{ mit } \ i,j \in \Omega_l. \end{aligned} \tag{4.8}$$

4.2.2 Affiner Segmentfluss

Die zweite Methode stellt den Fluss auf ein affin parametrisiertes Flussmodell um. Dabei wird davon ausgegangen, dass Objekte sich nicht statisch bewegen, sondern innerhalb der Segmente selbst unterschiedliche Flüsse entstehen können. Der Fluss wird deshalb durch die Parameterdarstellung einer Ebene modelliert, die je nach Position der Pixel ihnen eine unterschiedliche Verschiebung zuweisen kann. Die verschiedenen Pixel können durch folgende Parametrisierung beschrieben werden:

$$\begin{aligned} u_{i,j} &= a_l \cdot x + b_l \cdot y + c_l \ \text{ mit } \ i,j \in \Omega_l, \\ v_{i,j} &= d_l \cdot x + e_l \cdot y + f_l \ \text{ mit } \ i,j \in \Omega_l. \end{aligned} \tag{4.9}$$

Durch die Parametrisierung vergrößert sich die Anzahl der Flussvariablen u und v auf sechs, da jede zukünftig durch drei Parameter der Ebene definiert ist. Die Verschiebung in x-Richtung wird durch a, b, c und die Verschiebung in y-Richtung durch d, e, f bestimmt.

4.3 Ansatz

Für die Umstellung auf ein segmentbasiertes Verfahren müssen Änderungen am Daten- sowie Glattheitsterm durchgeführt werden. Da die Konstanzannahme nicht länger mit Pixeln arbeitet, sondern die Segmente miteinander vergleicht, wird eine Variable l eingeführt, welche jedem Pixel das zugehörige Segment zuordnet. Als Folge dessen wird die Energie der Segmente anhand der Summe aller Abweichungen der Pixel berechnet, die sich in Segment l befinden, und am Ende durch die Anzahl der Pixel normiert. Dies ist notwendig, da Segmente selten eine gleiche Anzahl an Pixel besitzen und sonst größere Segmente mehr Einfluss auf das Ergebnis haben. Der Glattheitsterm wird ähnlich der Basisverfahrens durch die Steigung zwischen den Segmenten berechnet, indem die Differenz des Flusses von Segment l mit allen Nachbarn j verglichen wird. Im Vergleich zu Pixel nehmen Segmente jedoch keine feste Position im Bild ein, weshalb theoretisch jeder Randpixel eines Segmentes einen unterschiedlichen Nachbar haben kann. Um die Anzahl der Nachbarn zu regulieren wird deshalb der Faktor $\frac{1}{|N(l)|}$ vorangestellt, um die Segmente zu normieren. Die Summe der Energie aller Segmente bildet schließlich die Gesamtenergie.

Konstanter Segmentfluss

Für den konstanten Fluss kann der Datenterm im Grunde von dem Basisverfahren übernommen werden. Einzig die Beziehung der Pixel innerhalb der Segmente muss geregelt werden. Die Energie des Segments wird daher durch die Summe der Energie aller Pixel innerhalb von l berechnet und anschließend normiert. Unter Beachtung aller Änderungen, kann das Funktional wie folgt formuliert werden:

$$\begin{aligned} E(u,v) = &\sum_{l=1}^{m} \Psi_S \left(\frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(x+u_l, y+v_l, t+1) - f_i(x,y,t))^2 \, dxdy \right) \\ &+ \alpha \sum_{l=1}^{m} \frac{1}{|N(l)|} \sum_{j \in N(l)} \Psi_S((u_l - u_j)^2 + (v_l - v_j)^2). \end{aligned} \tag{4.10}$$

Die Abweichung der Glattheit wird wie bisher durch Unterschiede zwischen den benachbarten Elementen bestimmt. In diesem Fall die Abweichung der Segmente l zu ihren Nachbarn j, welche anschließend durch die Anzahl normiert werden. Der Datenterm wird über den Segmentbereich Ω_l bestimmt. $\sum_{l=1}^{m}$ summiert die Energie aller Segmente auf und bildet die Gesamtenergie.

Affiner Segmentfluss

Für die Umstellung auf ein affines Datenmodell können die Flussvariablen u und v im Datenterm durch Substitution der Gleichungen 4.9 erstellt werden. Zu beachten ist, dass das Funktional nun von sechs Flussvariablen abhängt und deshalb die Glattheit von all diesen gefordert wird. Eingesetzt in das Funktional der konstanten Methode bildet dies das Funktional der affinen Methode:

$$E(a,b,c,d,e,f) = \sum_{l=1}^{m} \Psi_S \left(\frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} \xi_1^2 \, dxdy \right) + \alpha \sum_{l=1}^{m} \frac{1}{|N(l)|} \sum_{j \in N(l)} \Psi_S(\xi_2) \tag{4.11}$$

mit

$$\xi_1 = f_i(x + (a_l x + b_l y + c_l), y + (d_l x + e_l y + f_l), t+1) - f_i(x,y,t), \tag{4.12}$$

$$\xi_2 = (a_l - a_j)^2 + (b_l - b_j)^2 + (c_l - c_j)^2 + (d_l - d_j)^2 + (e_l - e_j)^2 + (f_l - f_j)^2. \tag{4.13}$$

4.4 Minimierung

Die Minimierung solcher Energiefunktionale wird ähnlich des Basisverfahrens durchgeführt, indem das Funktional abgeleitet und gleich 0 gesetzt wird. Da die Modellierung der Glattheit ohne die Gradienten des Flusses erfolgt und es sich um ein semi-diskretes Funktional handelt, in dem jedoch die Anzahl der Flussvariablen diskret ist, kann auf die Benutzung des Euler-Lagrange-Frameworks verzichtet und das Funktional nach den Flussvariablen gewöhnlich differenziert werden.

Konstanter Segmentfluss

Im nächsten Schritt kann nach den Flussvariablen u und v abgeleitet werden, wodurch wie bisher zwei Gleichungen entstehen. Dazu wird im Datenterm mithilfe der Kettenregel die äußere Ableitung mit Ψ'_D bestimmt, die mit der inneren Ableitung des quadratischen Teils multipliziert wird:

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x},\hat{y},t+1) - f_i(x,y,t)) \cdot f_{ix}(\hat{x},\hat{y},t+1) \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (u_l - u_j), \end{aligned} \tag{4.14}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x},\hat{y},t+1) - f_i(x,y,t)) \cdot f_{iy}(\hat{x},\hat{y},t+1) \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (v_l - v_j) \end{aligned} \tag{4.15}$$

mit

$$\hat{x} = x + u_l, \tag{4.16}$$
$$\hat{y} = y + v_l. \tag{4.17}$$

Der Glattheitsterm kann auf die gleiche Weise berechnet werden. Der Faktor $\frac{N_l+N_j}{N_l \cdot N_j}$ entsteht durch Normierung der Nachbarn. Da jedes Segment l mit dem Nachbarn j gleichzeitig auch ein Nachbar von j ist, existiert für jedes Segmentpaar ein doppelter Vergleich. Deshalb erfolgt eine Normierung anhand der Anzahl beider Segmentnachbarn, sodass jedes Segmentpaar nur einmal ins Gewicht fällt.

Affiner Segmentfluss

Die Minimierung des affinen Modells erfolgt über die Differenzierung nach allen sechs Variablen. Hierbei wird wie bereits bei der konstanten Methode die Kettenregel mehrfach auf den Datenterm angewendet. Zu beachten ist, dass durch die Substitution von u und v eine weitere Ableitung durchgeführt werden muss und so bei der Ableitung der Bildfunktion ein weiterer Faktor von x und y entsteht, da diese ebenfalls von a, b, d und e abhängen. Die Gleichungen aller Flussvariablen sehen demnach wie folgt aus:

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1) \cdot x \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (a_l - a_j), \end{aligned} \tag{4.18}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1) \cdot y \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (b_l - b_j), \end{aligned} \tag{4.19}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1) \cdot 1 \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (c_l - c_j), \end{aligned} \tag{4.20}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{iy}(\hat{x}, \hat{y}, t+1) \cdot x \, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (d_l - d_j), \end{aligned} \tag{4.21}$$

$$
\begin{aligned}
0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{iy}(\hat{x}, \hat{y}, t+1) \cdot y \, dxdy \\
+ \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (e_l - e_j),
\end{aligned}
\tag{4.22}
$$

$$
\begin{aligned}
0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{iy}(\hat{x}, \hat{y}, t+1) \cdot 1 \, dxdy \\
+ \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (f_l - f_j)
\end{aligned}
\tag{4.23}
$$

mit

$$
\begin{aligned}
\hat{x} &= x + (a_l x + b_l y + c_l), \\
\hat{y} &= y + (d_l x + e_l y + f_l).
\end{aligned}
\tag{4.24}
$$

4.5 Warping

Da bei den segmentbasierten Varianten die gleichen Probleme hinsichtlich der impliziten Abhängigkeit der Flussvariablen innerhalb der Bildfunktionen und großer Verschiebungen aufreten, wird die gleiche Coarse-to-Fine-Strategie wie im Basisverfahren verwendet. Wobei mit der Erstellung einer Fixpunktiteration für die jeweiligen Flussvariablen und der anschließenden Einführung des Inkrements der Level (siehe Abschnitt 3.3.6) angefangen wird. Nach der Berechnung eines Levels wird der Fluss der Segmente u_l, v_l auf jeden Pixel des Segments übertragen und für das Warping auf die nächsthöhere Ebene verwendet. Um für jedes Level eine entsprechende Repräsentation der Segmente zu gewährleisten, wird die Segmentierung für jedes Level durch einen Nearest-Neighbour-Verfahren skaliert, da eine Mittelung der Segmente nicht anwendbar ist. Dabei kann es vorkommen dass kleine Segmente je nach Stärke der Skalierung entfernt werden und erst ab einem bestimmten Level erstmals auftreten. Segmente die durch die Skalierung verschwunden sind, wird deshalb ein vernachlässigbarer Fluss zugeordnet und spielen erst ab ihrem ersten Auftreten eine Rolle.

Konstanter Segmentfluss

Um die Flussvariablen u_l, v_l auf das Warping vorzubereiten, muss ihnen ein Iterationsschritt zugewiesen werden, wobei sich die Zuordnung bei den segmentbasierten nicht von den pixelbasierten Methoden unterscheidet und für eine schnellere Konvergenz diese implizit im Datenterm und semi-implizit im Glattheitsterm vergeben werden. Anschließend kann der Fluss aus Iterationsschritt $k+1$ mit den Gleichungen aus 3.32 substituiert werden, womit sich der Fokus auf das Flussinkrement

du_l^k und dv_l^k lenkt. Um die Gleichung nach diesen Variablen umzustellen, wird am Schluss eine Linearisierung durch die Taylorreihe berechnet:

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1)\, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (u_l^k + du_l^k - u_j^k - du_j^k), \end{aligned} \tag{4.25}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{iy}(\hat{x}, \hat{y}, t+1)\, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (v_l^k + dv_l^k - v_j^k - dv_j^k). \end{aligned} \tag{4.26}$$

mit

$$\begin{aligned} \hat{x} &= x + u_l^k + du_l^k, \\ \hat{y} &= y + v_l^k + dv_l^k. \end{aligned} \tag{4.27}$$

Durch die Linearisierung des semi-impliziten Teils des Datenterms entsteht eine Differenz des zweiten und ersten Bildes, welche als zeitliche Änderung aufgefasst werden kann und mit der partiellen Ableitung f_t bezeichnet wird:

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_{ix} du_l^k + f_{iy} dv_l^k + f_{it}) \cdot f_{ix}\, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (u_l^k + du_l^k - u_j^k - du_j^k), \end{aligned} \tag{4.28}$$

$$\begin{aligned} 0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_{ix} du_l^k + f_{iy} dv_l^k + f_{it}) \cdot f_{iy}\, dxdy \\ + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (v_l^k + dv_l^k - v_j^k - dv_j^k). \end{aligned} \tag{4.29}$$

Für eine bessere Darstellung kann der Datenterm auf die bekannte Bewegungstensor Notation umgeformt werden. Dazu wird die Klammer ausmultipliziert und die Normierung, Integration über den Bildbereich und Farbe mit eingebunden. Die dadurch entstehende Gleichung hat große Ähnlichkeit

mit dem Basisverfahren (siehe 3.44) und kann mit einer kleinen Ausnahme bezüglich des Glattheitsterm auf die gleiche Weise implementiert werden:

$$\begin{aligned} 0 &= \Psi'_D \cdot (\hat{J}^k_{11} du^k_l + \hat{J}^k_{12} dv^k_l + \hat{J}^k_{13}) + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (u^k_l + du^k_l - u^k_j - du^k_j), \\ 0 &= \Psi'_D \cdot (\hat{J}_{12^k} du^k_l + \hat{J}^k_{22} dv^k_l + \hat{J}^k_{23}) + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (v^k_l + dv^k_l - v^k_j - dv^k_j) \end{aligned} \tag{4.30}$$

mit dem Bewegungstensor $\hat{J}$, auf den sich in den folgenden Abschnitten der konstanten Methode bezogen wird:

$$\hat{J}^k = \frac{1}{|\Omega_l|} \sum_{i=1}^{3} \int_{\Omega_l} \begin{pmatrix} f^2_{ix} & f_{ix} f_{iy} & f_{ix} f_{it} \\ f_{iy} f_{ix} & f^2_{iy} & f_{iy} f_{it} \\ f_{it} f_{ix} & f_{it} f_{iy} & f^2_{it} \end{pmatrix} dxdy. \tag{4.31}$$

Affiner Segmentfluss

Aufgrund der Tatsache, dass die Vorgehensweise der Fixpunktitertation, der Einführung eines Inkrements und der Linearisierung bei allen sechs Gleichungen bis auf wenige Unterschiede gleich abläuft, wird die Herleitung nur für die Variable da durchgeführt und anschließend analog auf die restlichen fünf Gleichungen angewendet. Die Einführung des Inkrements unterschiedet sich hinsichtlich zum Basisverfahren nur durch die Anzahl der Variablen und kann auf die gleiche Weise durchgeführt werden:

$$0 = \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1) \cdot x \, dxdy \tag{4.32}$$

$$+ \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (a^k_l + da^k_l - a^k_j - da^k_l) \tag{4.33}$$

mit

$$\begin{aligned} \hat{x} &= x + ((a^k_l + da^k_l)x + (b^k_l + db^k_l)y + (c^k_l + dc^k_l)), \\ \hat{y} &= y + ((d^k_l + dd^k_l)x + (e^k_l + de^k_l)y + (f^k_l + df^k_l)). \end{aligned} \tag{4.34}$$

Da der Fluss in diesem Modell durch eine Ebene dargestellt wird, hat die Position der Pixel innerhalb eines Segments Einfluss auf die berechnete Verschiebung. Dies hat den Grund, dass in einer schiefen Ebene der Fluss durch den Funktionswert dieser Ebene repräsentiert wird und einige der Pixel

geometrisch gesehen höher liegen als andere und so einen unterschiedlichen Wert erhalten. Die linearisierten Gleichungen nehmen dadurch folgende Form an:

$$\begin{aligned} 0 =& \Psi'_D \cdot \frac{1}{|\Omega_l|} \int_{\Omega_l} \sum_{i=1}^{3} ((f_{ix}x)da_l^k + (f_{ix}y)db_l^k + f_{ix}dc_l^k + (f_{iy}x)dd_l^k + (f_{iy}y)de_l^k + f_{iy}df_l^k + f_t) \\ & \cdot f_{ix} \cdot x \, dxdy + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (a_l^k + da_l^k - a_j^k - da_l^k). \end{aligned} \tag{4.35}$$

Um die gewohnte Schreibweise des Bewegungstensors zu erhalten, kann anschließend die Klammer ausmultipliziert werden und die verschiedenen Faktoren und Normierungen mit hineingezogen werden. Dadurch entsteht eine Gleichung die ähnlich der konstanten Variante ist, jedoch von allen sechs Variablen abhängt. Die bisherigen Schritte können für alle Variablen auf die gleiche Weise durchgeführt werden und nehmen schließlich diese Form an:

$$0 = \Psi'_D \cdot D_1 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (a_l^k + da_l^k - a_j^k - da_l^k), \tag{4.36}$$

$$0 = \Psi'_D \cdot D_2 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (b_l^k + db_l^k - b_j^k - db_l^k), \tag{4.37}$$

$$0 = \Psi'_D \cdot D_3 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (c_l^k + dc_l^k - c_j^k - dc_l^k), \tag{4.38}$$

$$0 = \Psi'_D \cdot D_4 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (d_l^k + dd_l^k - d_j^k - dd_l^k), \tag{4.39}$$

$$0 = \Psi'_D \cdot D_5 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (e_l^k + de_l^k - e_j^k - de_l^k), \tag{4.40}$$

$$0 = \Psi'_D \cdot D_6 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} \Psi'_S \cdot (f_l^k + df_l^k - f_j^k - df_l^k), \tag{4.41}$$

wobei die Datenterme D_1 - D_6 mit folgenden Gleichungen ersetzt werden:

$$\begin{aligned} D_1 &= \hat{J}_{11}^k da_l^k + \hat{J}_{12}^k db_l^k + \hat{J}_{13}^k dc_l^k + \hat{J}_{14}^k dd_l^k + \hat{J}_{15}^k de_l^k + \hat{J}_{16}^k df_l^k + \hat{J}_{17}^k, \\ D_2 &= \hat{J}_{12}^k da_l^k + \hat{J}_{22}^k db_l^k + \hat{J}_{23}^k dc_l^k + \hat{J}_{24}^k dd_l^k + \hat{J}_{25}^k de_l^k + \hat{J}_{26}^k df_l^k + \hat{J}_{27}^k, \\ D_3 &= \hat{J}_{13}^k da_l^k + \hat{J}_{23}^k db_l^k + \hat{J}_{33}^k dc_l^k + \hat{J}_{34}^k dd_l^k + \hat{J}_{35}^k de_l^k + \hat{J}_{36}^k df_l^k + \hat{J}_{37}^k, \\ D_4 &= \hat{J}_{14}^k da_l^k + \hat{J}_{24}^k db_l^k + \hat{J}_{34}^k dc_l^k + \hat{J}_{44}^k dd_l^k + \hat{J}_{45}^k de_l^k + \hat{J}_{46}^k df_l^k + \hat{J}_{47}^k, \\ D_5 &= \hat{J}_{15}^k da_l^k + \hat{J}_{25}^k db_l^k + \hat{J}_{35}^k dc_l^k + \hat{J}_{45}^k dd_l^k + \hat{J}_{55}^k de_l^k + \hat{J}_{56}^k df_l^k + \hat{J}_{57}^k, \\ D_6 &= \hat{J}_{16}^k da_l^k + \hat{J}_{26}^k db_l^k + \hat{J}_{36}^k dc_l^k + \hat{J}_{46}^k dd_l^k + \hat{J}_{56}^k de_l^k + \hat{J}_{66}^k df_l^k + \hat{J}_{67}^k, \end{aligned}$$

mit dem Bewegungstensor $\hat{J}$, auf den sich in den folgenden Abschnitten der affinen Variante bezogen wird. Durch die Erhöhung der Flussvariablen verändert sich der Bewegungstensor auf eine 7×7 Matrix:

$$\hat{J}^k = \frac{1}{|\Omega_l|} \sum_{i=1}^{3} \int_{\Omega_l} \begin{pmatrix} (f_x^2)x^2 & (f_x^2)xy & (f_x^2)x & (f_x f_y)x^2 & (f_x f_y)xy & (f_x f_y)x & (f_x f_t)x \\ (f_x^2)yx & (f_x^2)y^2 & (f_x^2)y & (f_x f_y)yx & (f_x f_y)y^2 & (f_x f_y)y & (f_x f_t)y \\ (f_x^2)x & (f_x^2)y & (f_x^2) & (f_x f_y)x & (f_x f_y)y & (f_x f_y) & (f_x f_t) \\ (f_y f_x)x^2 & (f_y f_x)xy & (f_y f_x)x & (f_y^2)x^2 & (f_y^2)xy & (f_y^2) & (f_y f_t)x \\ (f_y f_x)yx & (f_y f_x)y^2 & (f_y f_x)y & (f_y^2)yx & (f_y^2)y^2 & (f_y^2)y & (f_y f_t)y \\ (f_y f_x)x & (f_y f_x)y & (f_y f_x) & (f_y^2)x & (f_y^2)y & (f_y^2) & (f_y f_t) \\ (f_t f_x)x & (f_t f_x)y & (f_t f_x) & (f_t f_y)x & (f_t f_y)y & (f_t f_y) & (f_t^2) \end{pmatrix} dxdy. \tag{4.42}$$

4.6 Diskretisierung

Die Diskretisierung wie sie in Abschnitt 3.3.7 besprochen wurde, kann für viele Komponenten der segmentbasierten Varianten angewendet werden. Da zusätzlich die ursprüngliche Form des Funktional schon semi-diskret formuliert ist, beschränkt sich die Diskretisierung auf die Bewegungstensoreinträge und die beiden Ψ'-Funktionen, mit dem diskreten Bewegungstensor:

$$J_l^k = \frac{1}{|\Omega_l|} \sum_{k=1}^{3} \sum_{i,j \in l} \begin{pmatrix} [J_{11}^k]_{i,j} & \cdots & [J_{1n}^k]_{i,j} \\ \vdots & & \vdots \\ [J_{n1}^k]_{i,j} & \cdots & [J_{nn}^k]_{i,j} \end{pmatrix}. \tag{4.43}$$

Für die Tensoreinträge ist eine diskrete Darstellung der Funktion f und ihrer Ableitungen f_x, f_y und f_t notwendig, welche analog zum Basisverfahren durch finite Differenzen zu ermitteln sind. Das Integral des Tensoreintrags von Segment l kann durch die Summe der Pixel ersetzt werden, somit besteht l aus der Summe aller Tensoreinträge der Pixel innerhalb Segment l. Anschließend wird eine Normierung durch die Division der Anzahl der Pixel vorgenommen. Die Funktionen Ψ' können analog zu 3.61 aufgebaut und berechnet werden:

$$[\Psi']_l = \Psi' \left(\begin{pmatrix} [dx_1^k]_l \\ \vdots \\ [dx_{n-1}^k]_l \\ 1 \end{pmatrix}^T \begin{pmatrix} [J_{11}^k]_l & \cdots & [J_{1n}^k]_l \\ \vdots & & \vdots \\ [J_{n1}^k]_l & \cdots & [J_{nn}^k]_l \end{pmatrix} \begin{pmatrix} [dx_1^k]_l \\ \vdots \\ [dx_{n-1^k}]_l \\ 1 \end{pmatrix} \right), \tag{4.44}$$

wobei $[dx_n^k]_l$ für die n-te Flussvariable und $[J_{pq}^k]_l$ für den Bewegungstensoreintrag des Segments l an der Stelle pq steht.

Konstanter Segmentfluss

Für die finalen diskreten Gleichungen müssen die zuvor berechneten Komponenten eingefügt werden. Auf diese Weise entsteht eine Berechnung der Flusskomponenten du und dv mit:

$$
\begin{aligned}
0 &= [\Psi'_D]_l \cdot ([\hat{J}^k_{11}]_l du^k_l + [\hat{J}^k_{12}]_l dv^k_l + [\hat{J}^k_{13}]_l) + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (u^k_l + du^k_l - u^k_j - du^k_j), \\
0 &= [\Psi'_D]_l \cdot ([\hat{J}^k_{12}]_l du^k_l + [\hat{J}^k_{22}]_l dv^k_l + [\hat{J}^k_{23}]_l) + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (v^k_l + dv^k_l - v^k_j - dv^k_j).
\end{aligned}
\tag{4.45}
$$

Affiner Segmentfluss

Die affine Variante wird entsprechend der konstanten Methode aufgebaut. Hierfür werden die Komponenten für alle sechs Gleichungen eingefügt und können anschließend dem Löser übergeben werden:

$$0 = [\Psi'_D]_l \cdot D_1 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (a^k_l + da^k_l - a^k_j - da^k_l), \tag{4.46}$$

$$0 = [\Psi'_D]_l \cdot D_2 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (b^k_l + db^k_l - b^k_j - db^k_l), \tag{4.47}$$

$$0 = [\Psi'_D]_l \cdot D_3 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (c^k_l + dc^k_l - c^k_j - dc^k_l), \tag{4.48}$$

$$0 = [\Psi'_D]_l \cdot D_4 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (d^k_l + dd^k_l - d^k_j - dd^k_l), \tag{4.49}$$

$$0 = [\Psi'_D]_l \cdot D_5 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (e^k_l + de^k_l - e^k_j - de^k_l), \tag{4.50}$$

$$0 = [\Psi'_D]_l \cdot D_6 + \alpha \sum_{j \in N_l} \frac{|N(l)| + |N(j)|}{|N(l)| \cdot |N(j)|} [\Psi'_S]_l \cdot (f^k_l + df^k_l - f^k_j - df^k_l), \tag{4.51}$$

wobei der Datenterm aus den Gleichungen D_1 - D_6 besteht und mit folgenden Einträgen des affinen Bewegungstensors erstellt wird:

$$
\begin{aligned}
D_1 &= [\hat{J}^k_{11}]_l da^k_l + [\hat{J}^k_{12}]_l db^k_l + [\hat{J}^k_{13}]_l dc^k_l + [\hat{J}^k_{14}]_l dd^k_l + [\hat{J}^k_{15}]_l de^k_l + [\hat{J}^k_{16}]_l df^k_l + [\hat{J}^k_{17}]_l,\\
D_2 &= [\hat{J}^k_{12}]_l da^k_l + [\hat{J}^k_{22}]_l db^k_l + [\hat{J}^k_{23}]_l dc^k_l + [\hat{J}^k_{24}]_l dd^k_l + [\hat{J}^k_{25}]_l de^k_l + [\hat{J}^k_{26}]_l df^k_l + [\hat{J}^k_{27}]_l,\\
D_3 &= [\hat{J}^k_{13}]_l da^k_l + [\hat{J}^k_{23}]_l db^k_l + [\hat{J}^k_{33}]_l dc^k_l + [\hat{J}^k_{34}]_l dd^k_l + [\hat{J}^k_{35}]_l de^k_l + [\hat{J}^k_{36}]_l df^k_l + [\hat{J}^k_{37}]_l,\\
D_4 &= [\hat{J}^k_{14}]_l da^k_l + [\hat{J}^k_{24}]_l db^k_l + [\hat{J}^k_{34}]_l dc^k_l + [\hat{J}^k_{44}]_l dd^k_l + [\hat{J}^k_{45}]_l de^k_l + [\hat{J}^k_{46}]_l df^k_l + [\hat{J}^k_{47}]_l,\\
D_5 &= [\hat{J}^k_{15}]_l da^k_l + [\hat{J}^k_{25}]_l db^k_l + [\hat{J}^k_{35}]_l dc^k_l + [\hat{J}^k_{45}]_l dd^k_l + [\hat{J}^k_{55}]_l de^k_l + [\hat{J}^k_{56}]_l df^k_l + [\hat{J}^k_{57}]_l,\\
D_6 &= [\hat{J}^k_{16}]_l da^k_l + [\hat{J}^k_{26}]_l db^k_l + [\hat{J}^k_{36}]_l dc^k_l + [\hat{J}^k_{46}]_l dd^k_l + [\hat{J}^k_{56}]_l de^k_l + [\hat{J}^k_{66}]_l df^k_l + [\hat{J}^k_{67}]_l.
\end{aligned}
$$

4.7 Lösung

Für die Lösung der Gleichungen müssen diese anschließend auf das Lagged-Nonlinearity-Verfahren mit SOR-Löser umgestellt werden. Da beide Varianten sich nur in der Anzahl der Gleichungen unterscheiden und vom Prinzip her gleich aufgebaut sind, lässt sich das Lösungsverfahren als Framework beider segmentbasierten Methoden definieren:

$$
[dx_p]^{k,m,n+1}_l = (1-\omega)[dx_p]^{k,m,n}_l + \omega \left(\frac{\xi_1 + \alpha \sum_{j\in N_l^-} \frac{|N(l)|+|N(j)|}{|N(l)|\cdot|N(j)|}\xi_2 + \alpha \sum_{j\in N_l^+} \frac{|N(l)|+|N(j)|}{|N(l)|\cdot|N(j)|}\xi_3}{[\Psi'_D]^m_l [J^k_{pp}]_l + [\Psi'_S]_l \sum_{N_l} \frac{|N(l)|+|N(j)|}{|N(l)|\cdot|N(j)|}\cdot 1} \right) \tag{4.52}
$$

wobei d_{x_p} für die p-te Variable und J_{pp} für den pp-ten Eintrag des jeweiligen Bewegungstensors steht, mit $(x_1, x_2) = (u, v)$ und $(x_1, x_2, x_3, x_4, x_5, x_6) = (a, b, c, d, e, f)$.

Konstanter Segmentfluss

Die drei Komponenten des SOR-Lösers nehmen demnach für u und v folgende Form an und entstehen durch Umstellung der diskreten Lösungsgleichung nach dem Splittingverfahren:

u:

$$
\begin{aligned}
\xi_1 &= [\Psi'_D]^m_l(-[J^k_{12}]_l dv^{k,m,n}_l - [J^k_{13}]),\\
\xi_2 &= [\Psi'_S]^m_l \cdot (u^{k,m,n+1}_j + du^{k,m,n+1}_j - u^{k,m,n+1}_l),\\
\xi_3 &= [\Psi'_S]^m_l \cdot (u^{k,m,n}_j + du^{k,m,n}_j - u^{k,m,n}_l),
\end{aligned}
$$

v:

$$
\begin{aligned}
\xi_1 &= [\Psi'_D]^m_l(-[J^k_{12}]_l du^{k,m,n+1}_l - [J^k_{13}]),\\
\xi_2 &= [\Psi'_S]^m_l \cdot (v^{k,m,n+1}_j + dv^{k,m,n+1}_j - v^{k,m,n+1}_l),\\
\xi_3 &= [\Psi'_S]^m_l \cdot (v^{k,m,n}_j + dv^{k,m,n}_j - v^{k,m,n}_l).
\end{aligned}
$$

Affiner Segmentfluss

Aufgrund der identischen Umstellung unter den sechs Flussvariablen der affinen Methode, werden die Substitutionen nur für da aufgeführt. Die anderen Gleichungen können analog nach dem Schema des Basisverfahrens in Abschnitt 3.3.8 berechnet werden:

$$\xi_1 = [\Psi'_D]_l^m(-[J_{12}^k]_l db_l^{k,m,n} - [J_{13}^k] dc_l^{k,m,n} - [J_{14}^k] dd_l^{k,m,n}) \tag{4.53}$$

$$+ [\Psi'_D]_l^m(-[J_{15}^k]_l de_l^{k,m,n} - [J_{16}^k]_l df_l^{k,m,n} - [J_{17}^k]_l), \tag{4.54}$$

$$\xi_2 = [\Psi'_S]_l^m \cdot (a_j^{k,m,n+1} + da_j^{k,m,n+1} - a_l^{k,m,n+1}), \tag{4.55}$$

$$\xi_3 = [\Psi'_S]_l^m \cdot (a_j^{k,m,n} + da_j^{k,m,n} - a_l^{k,m,n}). \tag{4.56}$$

4.8 Erweiterung

Im nächsten Abschnitt wird ein weiterer Ansatz eingeführt, der auf ein Problem der segmentbasierten Varianten eingeht und diese mithilfe einer Erweiterung der Segmentierung verbessert. In den bisherigen Methoden des letzten Abschnitts wurde versucht die Verschiebung durch stückweise konstante oder affine Verschiebung zu modellieren. Diese Techniken decken jedoch nicht alle Möglichkeiten einer Bewegung ab. Die konstante Methode lässt nur eine Verschiebung für alle Pixel des Segments zu. Dies wurde durch die affine Beschreibung verbessert indem lineare Änderungen innerhalb der Segmente erlaubt wurden. Da die Segmente jedoch immer als Ganzes gesehen und verschoben werden, lassen sich beispielsweise keine Rotationen oder ein Zoom an Objekte realisieren, da hierfür Teile des Segments sich in unterschiedliche Richtungen bewegen müssen. Aus diesem Grund wird eine Modifizierung der Segmentierung mithilfe eines simplen pixelbasierten Verfahren durchgeführt, das den Fluss vorberechnet und diese Informationen an das Mean-Shift-Verfahren weitergibt. Dadurch entsteht eine Verfeinerung der entsprechenden Bereiche durch Trennung der Segmente in ihre unterschiedlichen Flüsse, welche anschließend von den bekannten Verfahren separat berechnet werden können.

4.8.1 Segmentierung mit vorberechnetem Fluss

Um das Flussfeld mit in die Segmentierung einzubringen, müssen die Bilder in ihrer Dimension erweitert werden. In Abschnitt 4.1.1 bestanden die Bilder aus dem zweidimensionalen räumlichen Teil und dem dreidimensionalen farblichen Bereich. In der folgenden Variante kommt ein zweidimensionaler Flussbereich hinzu, der die Verschiebung der x- und y-Achse darstellt. Somit wird von einem siebendimensionalen Wertebereich ausgegangen, in dem ein Punkt durch folgenden Vektor dargestellt werden kann:

$$x_i = \begin{pmatrix} x & y & L & a & b & u & v \end{pmatrix}^T. \tag{4.57}$$

Die Berechnung des Mean-Shift-Verfahrens kann analog zu der bisher behandelten Methode durchgeführt werden. Einzig eine dritte Gewichtungsfunktion muss für die Berechnung des Abstands der Flusskomponenten innerhalb des Flussbereichs hinzugefügt werden. Der Mean-Shift Vektor ändert sich so gesehen selbst nicht, jedoch wird das Gewicht g um einen weiteren Teil erweitert:

$$m_{h,g}(x) = \frac{\sum_{i=1}^{n} x_i \, g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)} - x, \tag{4.58}$$

wobei anstatt $g\left(\left\|\frac{x-x_i}{h}\right\|^2\right)$ folgende Fensterfunktion verwendet wird

$$G(h_s, h_r, h_f) = g\left(\left\|\frac{x^s}{h_s}\right\|^2\right) \cdot g\left(\left\|\frac{x^r}{h_r}\right\|^2\right) \cdot g\left(\left\|\frac{x^f}{h_f}\right\|^2\right) \tag{4.59}$$

die sich aus den Abständen des Raumes x^s, der Farbe x^r und dem Fluss x^f zusammensetzt. Durch Änderung des Flussparameters lassen sich so unterschiedlich starke Segmentierungen in den Bereichen erzeugen, die eine hohe Änderungsrate des Flusses besitzen. In Abbildung 4.9 sind einige Segmentierungen mit eingezeichneten Kanten abgebildet. Bei einem Vergleich des Flusses und den Segmentierungen ist leicht zu erkennen, dass gerade die Bereiche mit einer hohen Änderungsrate des Flusses eine Vielzahl an neuen Segmenten erhalten haben. Besonders auffällig ist dies bei dem Rad im unteren Abschnitt oder der Decke in der Mitte des Bildes. Die Rotation des Rades kann somit besser durch viele kleine Segmente berechnet werden.

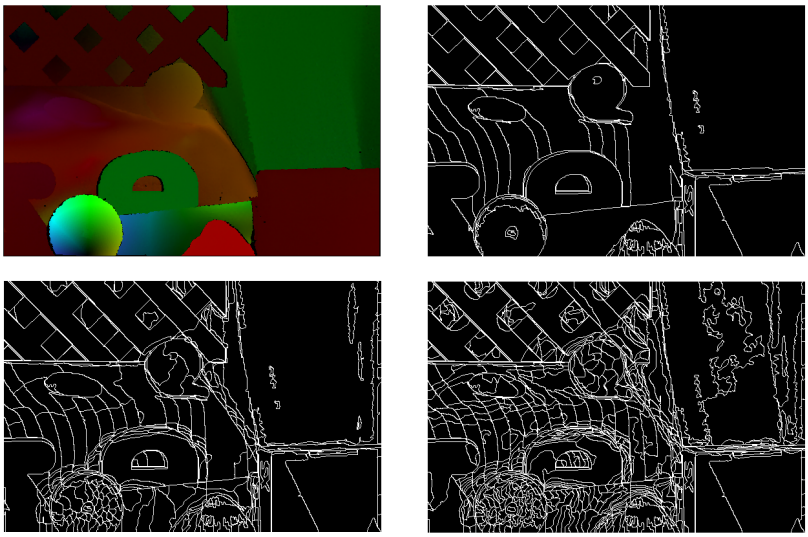


Abbildung 4.9: Darstellung verschieden starker Teilung der Segmente mit unterschiedlichem Fluss. **Links oben:** Flussfeld. **Rechts oben:** Segmentierung ohne Vorberechnung. **Links unten:** Teilung der Segmente ab einem Flussunterschied von 0.5. **Rechts unten:** Teilung ab einem Unterschied von 0.2.

5 Pixelbasierte Methoden

Im bisherigen Teil dieser Arbeit wurde die Segmentierung genutzt um die Flussvariablen mithilfe von Segmenten darzustellen und so eine Verringerung an Variablen und einer besseren Erhaltung der Grenzen zu erzeugen. Eine weitere Variante Segmente in den Optischen Fluss mit einzubinden ist eine Kombination aus pixelbasierter Berechnung und den Vorteilen eines segmentierten Bildes. Da die Segmentierung eine gute Repräsentation der inhaltlichen Objekte darstellt und pixelbasierte Methoden durch ihre feine Granularität durch einzelne Pixel gute Ergebnisse bei komplexeren Verschiebungen erhalten, ist es sinnvoll diese Beziehung etwas genauer zu untersuchen und die Vorteile beider Verfahren miteinander zu verbinden. Das Ziel dieses Abschnitts besteht darin, die Glättung an den Kanten durch eine Segmentierung stärker zu unterbinden und einen schärferen verlauf der Objekte zu erhalten, bei gleichzeitiger Verfeinerung des Flusses innerhalb der Segmente durch eine pixelbasierte Berechnung.

5.1 Methoden

Nach diesem Grundgedanken wurden drei pixelbasierte Modelle entwickelt wovon die Erste ein Basisverfahren für die späteren Methoden darstellt und zu Vergleichszwecken zusätzlich hergeleitet wird. Für die anderen Verfahren wurden verschiedene Segmentierungen vorbereitet, die Einfluss auf das Flussverhalten der Pixel haben.

5.1.1 Anisotroper Regularisierer

Die erste pixelbasierte Variante stellt das isotrope Basisverfahren (siehe Abschnitt 3) auf einen anisotropen Glattheitsterm um und bildet das neue Basisverfahren der folgenden Methoden. Bis zu diesem Punkt wurde die Diffusion richtungsunabhängig behandelt. In Bezug auf die unterschiedliche Beschaffenheit des Flusses durch Kanten und homogenen Flächen ist es jedoch sinnvoll derartige Unterschiede anders zu behandeln. Zur Verbesserung kann deshalb eine Unterscheidung eingeführt werden, die die Struktur des Flusses analysiert und die Stärke der Diffusion richtungsabhängig regelt. So kann der Fluss über eine Kante reduziert und entlang einer Kante geglättet werden. Dazu ist allerdings eine Betrachtung aller Nachbarpixel notwendig, womit sich der Glattheitsterm von den bisherigen vier auf acht Nachbarpixel erweitert.

5.1.2 Regularisierung durch Kantenmap

Für die zweite Variante wird zusätzlich eine Kantenmap zur Regelung der Diffusion verwendet. Diese wird aus der Segmentierung des Bildes erstellt und erzeugt zwischen allen Pixeln, deren Nachbarn in einem anderen Segment liegen eine Grenze, welche die Diffusion in dieser Richtung stark heruntergewichtet. So entsteht eine zusätzlich Barriere zu der flussgetriebenen Regulierung, welche die Information der Segmentierung nutzt um Kanten besser zu erhalten. Dies ist jedoch nicht mit einem bildgetriebenen Glattheitsterm zu verwechseln, bei welchem die Struktur des Bildes analysiert wird und anhand der Kanten der Fluss gelenkt wird. Die Regelung des Flusses erfolgt weiterhin durch die Struktur von u und v und wird lediglich durch die Kantenmap stärker reduziert. Bildgetriebene Verfahren haben oft den Nachteil, dass eine Übersegmentierung erfolgt, die den Fluss zudem an ungewollten Stellen verändert. Um diesen Effekt zu verringern, sollten die Parameter der Segmentierung etwas höher gewählt werden um so möglichst die wahren Kanten zu erhalten.

5.1.3 Komplementärer Regularisierer

Die letzte Methode verknüpft die Vorteile einer bildgetriebenen mit der einer flussgetriebenen Methode [ZBW+09]. Dazu wird nicht wie bisher, der Fluss als leitende Kraft für die Erkennung von Objektkanten verwendet, sondern die Struktur der Segmentierung an sich. Da die Segmente aus einer konstanten Farbe und somit aus homogenen Flächen bestehen, können unerwünschte Kanten herausgefiltert und nur die Objektkanten verwendet werden. Anhand dieser Kanteninformation lässt sich die Diffusion in zwei verschiedene Bereiche aufteilen. Für den Fall einer Kante, ist es wünschenswert diese maximal zu erhalten. Die orthogonale Richtung entlang der Kante, soll jedoch möglichst glatt sein. Um dies zu realisieren wird für jede der zwei Bereiche eine unterschiedliche Funktion verwendet. In Richtung einer Kante kommt die Perona-Malik Diffusionsfunktion [PM90]zum Einsatz, welche eine verstärkende Wirkung auf Kanten hat, entlang einer Kante wird dagegen wie gehabt die Charbonnier-Funktion verwendet.

5.2 Ansatz

Für die Umstellung auf einen anisotropen Glattheitsterm ist es notwendig die Analyse der Nachbarn auf alle acht zu erweitern und die diagonalen Pixel miteinzubeziehen. Dafür muss der bisher verwendete Glattheitsterm so abgeändert werden, dass eine richtungsbezogene Aussage erstellt werden kann. Hierfür kann der Strukturtensor [FG87] genutzt werden. Er ist eine 2×2 positiv semidefinite Matrix, welche mithilfe der Gradientenrichtung die Struktur von homogenen Flächen, über Kanten und Ecken unterscheidet.

Anisotroper Regularisierer

Für das anisotrope Energiefunktional wird der bisherige isotrope Glattheitsterm durch den Strukturtensor ersetzt, wobei der Fluss von u und v separat untersucht und am Ende addiert wird. Hierbei wird die Struktur des Flusses auf Unebenheiten untersucht, die in den meisten Fällen für den Übergang

zwischen verschiedenen Objekten stehen und die Diffusion anhand dieser geregelt wird. Da es sich um eine 2×2 Matrix handelt, wird anschließend noch die Spur berechnet um später eine skalarwertige Funktion zu erhalten. Das Funktional mit anisotropem Regularisierer sieht damit wie folgt aus:

$$E(u,v) = \int_\Omega \Psi_D \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2 \right) \tag{5.1}$$

$$+ \alpha \, tr \, \Psi_S (\nabla u \nabla u^T + \nabla v \nabla v^T) \, dxdy. \tag{5.2}$$

Kantenmap Regularisierer

Die Kantenmap wird verwendet um den Fluss an Pixelübergängen verschiedener Segmente fast vollständig zu reduzieren. Dazu wird für jeden Pixel eine Nachbarschaft in allen Richtungen abgespeichert und entsprechend markiert, falls eine Kante zwischen diesen Pixeln vorhanden ist. Für ein genaueres Ergebnis werden die Kanten zudem nicht auf den Pixel selbst ausgewertet, was zu einer späteren Mittelung der Kanten in der jeweiligen Richtung führen würde, sondern direkt in den Pixelzwischenräumen gespeichert, was eine direkte Multiplikation der Kantenmap im späteren Verlauf ermöglicht. Die Kanten werden hier als Funktion $g_{Edge}(f_1)$ repräsentiert da nur eine Segmentierung von Bild 1 erzeugt wird. Diese kann mit dem bereits bekannten Glattheitsterm der anisotropen Variante multipliziert werden:

$$E(u,v) = \int_\Omega \Psi_D \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2 \right) \tag{5.3}$$

$$+ \alpha \cdot g_{Edge}(f_1) \cdot tr \, \Psi_S (\nabla u \nabla u^T + \nabla v \nabla v^T) \, dxdy. \tag{5.4}$$

Komplementärer Regularisierer

Die dritte Methode spaltet den Glattheitsterm in die zuvor erwähnten Teile auf, wobei die Perona-Malik Funktion, hier mit Ψ_{S_1} dargestellt, die Regulierung der Vektoren in r_1-Richtung übernimmt. Tritt in der Segmentierung eine Kante auf verläuft diese stets in Richtung r_1. Dabei wird der Fluss u und v jeweils separat untersucht. Die zweite Funktion ist für die orthogonalen Vektoren r_2 verantwortlich und steht für die Charbonnier-Funktion. Sie hat die Aufgabe entlang der Kante in Richtung r_2 den Fluss zu glätten. In den Bereichen innerhalb eines Segmentes werden beide Richtungen beliebig gewählt und es wird eine gewöhnliche Glättung durchgeführt. Dies kann wie folgt realisiert werden:

$$E(u,v) = \int_\Omega \Psi_D \left(\sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2 \right) \tag{5.5}$$

$$+ \alpha (\Psi_{S_1}((\nabla u^T r_1)^2 + (\nabla v^T r_1)^2) + \Psi_{S_2}((\nabla u^T r_2)^2 + (\nabla v^T r_2)^2)) \, dxdy. \tag{5.6}$$

5.3 Minimierung

Um die Energiefunktionale der pixelbasierten Methoden zu lösen werden die bereits bekannten Euler-Lagrange-Gleichungen aus Abschnitt 3.3.5 verwendet und auf die gewohnte Form der Divergenz gebracht. Durch die Umstellung auf einen anisotropen Glattheitsterm verändert sich der Diffusionstensor D allerdings von einem skalaren Wert zu einer 2×2 Matrix. Für ein besseres Ergebnis wird D zusätzlich mit einem Gaußkern geglättet, was hier nur der Vollständigkeit halber erwähnt wird, da dieser Schritt nicht in das Energiefunktional mit einbezogen werden kann. Angefangen mit den Euler-Lagrange-Gleichungen:

$$\begin{aligned} 0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{ix}(x+u, y+v, t+1) \\ &\quad - \alpha \operatorname{div}(D \nabla u) \\ 0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t)) \cdot f_{iy}(x+u, y+v, t+1) \\ &\quad - \alpha \operatorname{div}(D \nabla v). \end{aligned} \tag{5.7}$$

Anisotroper Regularisierer

Da Ψ'_S bisher nur für skalarwertige Argumente definiert wurde, bei den anisotropen Methoden jedoch eine 2×2 Matrix vorliegt, muss dieser Fall noch etwas genauer untersucht werden. Die Aufgabe von Ψ'_S ist, Sprünge des Flusses zu regulieren und so die Diffusion an diesen Richtungen weitgehend zu stoppen. Für den Fall des Strukturtensors bedeutet dies eine Zerlegung der Matrix in seine Eigenvektoren und Eigenwerte. Ψ'_S kann so gezielt auf den Eigenwert angewendet werden, da dieser nichts anderes als den Ausschlag oder die Stärke der Kante in Richtung des Eigenvektors angibt. Mittels der Eigenzusammensetzung kann die Matrix anschließend neu gebildet werden. Die Ψ-Funktionen und D nehmen dabei folgende Form an:

$$\Psi'_D = \frac{1}{\sqrt{1 + \frac{s^2}{\epsilon_D^2}}} \qquad \text{und} \qquad s^2 = \sum_{1}^{3} (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2, \tag{5.8}$$

$$D = \Psi'_S \begin{pmatrix} u_x^2 + v_x^2 & u_x u_y + v_x v_y \\ u_y u_x + v_y v_x & u_y^2 + v_y^2 \end{pmatrix} = (e_1, e_2) \begin{pmatrix} \Psi'_S(\lambda_1) & 0 \\ 0 & \Psi'_S(\lambda_2) \end{pmatrix} (e_1, e_2)^T. \tag{5.9}$$

Kantenmap Regularisierer

Die Vorgehensweise der anisotropen Methode kann für die Kantenmap übernommen werden und unterschiedet sich in erster Linie dadurch, dass die Kantenmap an D heranmultipliziert wird. Durch die Richtungsabhängigkeit der Kantenmap muss diese mit in die Divergenz gezogen werden und verstärkt so die Reduzierung der Diffusion weiter anhand der Informationen der Segmentierung. Diese kann ebenfalls wie bei der anisotropen Methode durch eine direkte Anwendung auf die Eigenwerte des Strukturtensors durchgeführt werden:

$$\Psi'_D = \frac{1}{\sqrt{1+\frac{s^2}{\epsilon^2}}} \qquad \text{und} \qquad s^2 = \sum_1^3 (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2, \tag{5.10}$$

$$D = g_{Edge}(f_1) \cdot \Psi'_S \begin{pmatrix} u_x^2 + v_x^2 & u_x u_y + v_x v_y \\ u_y u_x + v_y v_x & u_y^2 + v_y^2 \end{pmatrix} \tag{5.11}$$

$$= (e_1, e_2) \begin{pmatrix} g_{Edge}(f_1) \cdot \Psi'_S(\lambda_1) & 0 \\ 0 & g_{Edge}(f_1) \cdot \Psi'_S(\lambda_2) \end{pmatrix} (e_1, e_2)^T. \tag{5.12}$$

Komplementärer Regularisierer

Für den komplementären Ansatz wird zuerst der Strukturtensor der Segmentierung erstellt, aus welchem sich durch Eigenzerlegung die Richtungsvekoren r_1 und r_2 berechnen lassen, wobei r_1 die Richtung des größeren Eigenwerts darstellt und im Fall einer Kante auf diese zeigt. r_2 bestimmt somit den orthogonalen Vektor entlang dieser Kante. Der Diffusionstensor D lässt sich dementsprechend durch die Skalarmultiplikation aus Ψ_{S_1} und dem direkten Produkt aus r_1 mit sich selbst bilden, was zu einer Kantenerhaltung durch Perona-Malik führt. Die Multiplikation von Ψ_{S_2} und dem direkten Produkt aus r_2 mit sich selbst erzeugt eine Glättung entlang dieser Kante.

$$\Psi'_D = \frac{1}{\sqrt{1+\frac{s^2}{\epsilon_D^2}}} \qquad \text{und} \qquad s^2 = \sum_1^3 (f_i(x+u, y+v, t+1) - f_i(x,y,t))^2, \tag{5.13}$$

$$D = \Psi'_{S_1} \cdot \begin{pmatrix} r_{11}^2 & r_{11} r_{12} \\ r_{12} r_{11} & r_{12}^2 \end{pmatrix} + \Psi'_{S_2} \cdot \begin{pmatrix} r_{21}^2 & r_{21} r_{22} \\ r_{22} r_{21} & r_{22}^2, \end{pmatrix} \tag{5.14}$$

wobei die Argumente s_2 im Gegensatz zu den bisherigen Methoden einem Skalar entsprechen. Zu beobachten ist, dass so eine Verbindung zwischen den bildgetriebenen und flussgetriebenen Methoden entsteht, wobei die Richtung der Diffusion durch die Bildinformation anhand der Richtungsvektoren

der Segmentierung gesteuert wird, das Ausmaß der Reduzierung jedoch durch die Struktur des Flusses geregelt wird. Die beiden Ψ'_S Funktionen nehmen dabei folgende Form an:

$$\Psi'_{S_1} = \frac{1}{1 + \frac{s^2}{\lambda^2}} \qquad \text{und} \qquad s^2 = r_1^T(\nabla u \nabla u^T + \nabla v \nabla v^T) r_1, \tag{5.15}$$

$$\Psi'_{S_2} = \frac{1}{\sqrt{1 + \frac{s^2}{\epsilon_D^2}}} \qquad \text{und} \qquad s^2 = r_2^T(\nabla u \nabla u^T + \nabla v \nabla v^T) r_2. \tag{5.16}$$

Der Parameter λ stellt den Kontrastparameter dar, der die Vorwärts- von der Rückwärtsdiffusion trennt.

5.4 Warping

Für die Anwendung der Methoden auf größere Verschiebungen, wird die bekannte Coarse-to-Fine-Strategie (siehe Abschnitt 3.3.6) aus den bisherigen Verfahren verwendet. Da bei der Umstellung von isotrop auf anisotrop nur eine Erhöhung der Anzahl der betrachteten Nachbarn stattfindet, kann das Framework des isotropen Basisverfahrens angewendet und die Fixpunktiteration so auf alle drei anisotrope Verfahren analog aufgebaut werden, indem für die beiden Flussvariablen u und v ein Inkrement mit der Regel $u^{k+1} = u^k + du^k$ sowie $v^{k+1} = v^k + dv^k$ eingesetzt wird:

$$\begin{aligned} 0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{ix}(\hat{x}, \hat{y}, t+1) - \alpha \operatorname{div}(D\, \nabla u^k) - \alpha \operatorname{div}(D\, \nabla du^k), \\ 0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_i(\hat{x}, \hat{y}, t+1) - f_i(x, y, t)) \cdot f_{iy}(\hat{x}, \hat{y}, t+1) - \alpha \operatorname{div}(D\, \nabla v^k) - \alpha \operatorname{div}(D\, \nabla dv^k) \end{aligned} \tag{5.17}$$

mit

$$\begin{aligned} \hat{x} &= x + u^k + du^k, \\ \hat{y} &= y + v^k + dv^k. \end{aligned} \tag{5.18}$$

Anschließend wird durch die Taylorreihe der Datenterm linearisiert um die gesuchten Variablen auszuklammern und so eine Berechnung des Inkrements zu ermöglichen. Hier kann der Datenterm auf die bekannte Bewegungstensornotation umgeformt werden. Da sich die Funktionale nur im Glattheitsterm unterscheiden, ist dieser Schritt für die anisotropen Verfahren identisch. Die linearisierten Gleichungen sind durch

$$
\begin{aligned}
0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_{ix} du^k + f_{iy} dv^k + f_{it}) \cdot f_{ix} - \alpha \operatorname{div}(D \nabla u^k) - \alpha \operatorname{div}(D \nabla du^k), \\
0 &= \Psi'_D \cdot \sum_{i=1}^{3} (f_{ix} du^k + f_{iy} dv^k + f_{it}) \cdot f_{iy} - \alpha \operatorname{div}(D \nabla v^k) - \alpha \operatorname{div}(D \nabla dv^k)
\end{aligned}
\tag{5.19}
$$

gegeben. Durch Ausmultiplikation der Klammer können die partiellen Ableitungen durch die Tensoreinträge von J beschrieben werden.:

$$
\begin{aligned}
0 &= \Psi'_D \cdot (J^k_{11} du^k + J^k_{12} dv^k + J^k_{13}) - \alpha \operatorname{div}(D \nabla u^k) - \alpha \operatorname{div}(D \nabla du^k), \\
0 &= \Psi'_D \cdot (J^k_{12} du^k + J^k_{22} dv^k + J^k_{23}) - \alpha \operatorname{div}(D \nabla v^k) - \alpha \operatorname{div}(D \nabla dv^k)
\end{aligned}
\tag{5.20}
$$

mit dem Bewegungstensor

$$
J^k = \sum_{i=1}^{3} \begin{pmatrix} f_{ix}^2 & f_{ix} f_{iy} & f_{ix} f_{it} \\ f_{iy} f_{ix} & f_{iy}^2 & f_{iy} f_{it} \\ f_{it} f_{ix} & f_{it} f_{iy} & f_{it}^2 \end{pmatrix}. \tag{5.21}
$$

Anisotroper Regularisierer

Durch das Warping muss der Tensor D im Glattheitsterm angepasst werden, indem das Inkrement ebenfalls für die Flussvariablen eingesetzt wird. Dies führt zu einer einfachen Substitution der Einträge in Matrix D mit den Gleichungen aus 3.32. Zur Lösung wird die Gleichung anschließend in die Coarse-to-Fine-Pyramide eingebettet. Der Diffusionstensor sieht dabei folgendermaßen aus:

$$
D = \Psi'_S \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{5.22}
$$

mit

$$
a = (u_x + du_x)^2 + (v_x + dv_x)^2, \tag{5.23}
$$

$$
b = (u_x + du_x)(u_y + du_y) + (v_x + dv_x)(v_y + dv_y), \tag{5.24}
$$

$$
c = (u_y + du_y)(u_x + du_x) + (v_y + dv_y)(v_x + dv_x), \tag{5.25}
$$

$$
d = (u_y + du_y)^2 + (v_y + dv_y)^2. \tag{5.26}
$$

Kantenmap Regularisierer

Für die Methode der Kantenmap kann die Vorgehensweise der anisotropen Variante exakt übernommen werden, da sich das Verfahren einzig durch die spätere Regulierung der Kantenmap unterscheidet und diese unabhängig vom Fluss ist. Die Einträge von D nehmen deshalb die gleichen Werte wie in Gleichung 5.22 an

Komplementärer Regularisierer

Da bei der komplementären Methode beide Tensoren aus dem Strukturtensor der Segmentierung bestehen und der Fluss somit nur als skalares Argument in beiden Funktionen Ψ_{S_1} und Ψ_{S_2} integriert ist, reicht es die Faktoren anzupassen indem u^{k+1}, v^{k+1} mit $u^k + du^k$ und $v^k + dv^k$ ersetzt werden. Die Argumente s^2 der Ψ-Funktionen nehmen somit folgende Form an:

$$D = \Psi'_{S_1} \cdot \begin{pmatrix} r_{11}^2 & r_{11}r_{12} \\ r_{12}r_{11} & r_{12}^2 \end{pmatrix} + \Psi'_{S_2} \cdot \begin{pmatrix} r_{21}^2 & r_{21}r_{22} \\ r_{22}r_{21} & r_{22}^2 \end{pmatrix}, \tag{5.27}$$

$$\Psi'_{S_1} = \frac{1}{1 + \frac{s^2}{\lambda^2}} \quad \text{und} \quad s^2 = r_1^T(\nabla(u+du)\nabla(u+du)^T + \nabla(v+dv)\nabla(v+dv)^T)r_1, \tag{5.28}$$

$$\Psi'_{S_2} = \frac{1}{\sqrt{1 + \frac{s^2}{\epsilon^2}}} \quad \text{und} \quad s^2 = r_2^T(\nabla(u+du)\nabla(u+du)^T + \nabla(v+dv)\nabla(v+dv)^T)r_2. \tag{5.29}$$

5.5 Diskretisierung

Die Diskretisierung der pixelbasierten anisotropen Methoden kann für die drei Varianten auf die Diskretisierung des Frameworks aus Gleichung 3.62 reduziert werden, mit dem Unterschied dass im anisotropen Fall die diagonalen Einträge des Diffusionstensors ebenfalls vorhanden sind. Die zu diskretisierenden Gleichungen sind dabei:

$$\begin{aligned} 0 &= \Psi'_D \cdot (J_{11}^k du^k + J_{12}^k dv^k + J_{13}^k) - \alpha \operatorname{div}(D\,\nabla u^k) - \alpha \operatorname{div}(D\,\nabla du^k), \\ 0 &= \Psi'_D \cdot (J_{12}^k du^k + J_{22}^k dv^k + J_{23}^k) - \alpha \operatorname{div}(D\,\nabla v^k) - \alpha \operatorname{div}(D\,\nabla dv^k). \end{aligned} \tag{5.30}$$

Mit der Standarddiskretisierung, welche bei den isotropen Methoden verwendet wird, lässt sich allerdings nur eine unzureichend genaue Schablone bilden. Da bei einer positiv semidefiniten Matrix die Eigenschaft der positiven Werte nur für die Einträge a und d garantiert wird und b, c willkürliche Vorzeichen haben können, ist eine Positivität der Nachbarn in der Schablone damit nicht gegeben. Aus diesem Grund wird die Diskretisierung durch den Non-Negativity-Stencil [Wei98] vorgenommen:

$\frac{\lvert b_{i-1,j+1}\rvert - b_{i-1,j+1}}{4h_1h_2}$ $+\frac{\lvert b_{i,j}-b_{i,j}\rvert}{4h_1h_2}$	$\frac{c_{i,j+1}+c_{i,j}}{2h_2^2} - \frac{\lvert b_{i,j+1}\rvert+\lvert b_{i,j}\rvert}{2h_1h_2}$	$\frac{\lvert b_{i+1,j+1}\rvert - b_{i+1,j+1}}{4h_1h_2}$ $+\frac{\lvert b_{i,j}+b_{i,j}\rvert}{4h_1h_2}$
$\frac{a_{i-1,j}+a_{i,j}}{2h_1^2}$ $-\frac{\lvert b_{i-1,j}\rvert+\lvert b_{i,j}\rvert}{2h_1h_2}$	$-\frac{a_{i-1,j}+2a_{i,j}+a_{i+1,j}}{2h_1^2}$ $-\frac{\lvert b_{i-1,j+1}\rvert-b_{i-1,j+1}+\lvert b_{i+1,j+1}\rvert+b_{i+1,j+1}}{4h_1h_2}$ $-\frac{\lvert b_{i-1,j-1}\rvert+b_{i-1,j-1}+\lvert b_{i+1,j-1}\rvert-b_{i+1,j-1}}{4h_1h_2}$ $+\frac{\lvert b_{i-1,j}\rvert+\lvert b_{i+1,j}\rvert+\lvert b_{i,j-1}\rvert+\lvert b_{i,j+1}\rvert+2\lvert b_{i,j}\rvert}{2h_1h_2}$ $-\frac{c_{i,j-1}+2c_{i,j}+c_{i,j+1}}{2h_2^2}$	$\frac{a_{i+1,j}+a_{i,j}}{2h_1^2}$ $-\frac{\lvert b_{i+1,j}\rvert+\lvert b_{i,j}\rvert}{2h_1h_2}$
$\frac{\lvert b_{i-1,j-1}\rvert+b_{i-1,j-1}}{4h_1h_2}$ $+\frac{\lvert b_{i,j}\rvert+b_{i,j}}{4h_1h_2}$	$\frac{c_{i,j-1}+c_{i,j}}{2h_2^2} - \frac{\lvert b_{i,j-1}\rvert+\lvert b_{i,j}\rvert}{2h_1h_2}$	$\frac{\lvert b_{i+1,j-1}\rvert - b_{i+1,j-1}}{4h_1h_2}$ $+\frac{\lvert b_{i,j}\rvert-b_{i,j}}{4h_1h_2}$

Tabelle 5.1: Schablone der anisotropen Diskretisierung durch den Non-Negativity-Stencil.

Diese Maske ist ähnlich der Schablone aus Abbildung 3.1 und zeigt die Gewichtung der benachbarten Pixel inklusive des zentralen Pixels. Dabei wird durch eine Diskretisierung zweiter Ordnung ein Schablone gebildet, die die gemischten Terme b beinhält und gleichzeitig positive Gewichte der Nachbarn sicherstellt. Die Einträge a, b, c an den Stellen i, j bilden dabei die Einträge des Diffusionstensors D an den jeweiligen Positionen der Pixel. h_1 und h_2 stehen für die Schrittlänge in x- sowie y-Richtung. Die restlichen Komponenten können analog zum Basisverfahren (siehe Abschnitt 3.3.7) diskretisiert werden. Die finalen Euler-Lagrange-Gleichungen für die anisotropen Methoden sehen demnach wie folgt aus:

$$
\begin{aligned}
0 =& [\Psi'_D]_{i,j} \cdot ([J^k_{11}]_{i,j} du^k_{i,j} + [J^k_{12}]_{i,j} dv^k_{i,j} [J^k_{13}]_{i,j}) \\
&- \alpha \sum_{(\bar{i},\bar{j}) \in N(i,j)} w_{i,j,\bar{i},\bar{j}} \cdot (u^k_{\bar{i},\bar{j}} - u^k_{i,j}) \\
&- \alpha \sum_{(\bar{i},\bar{j}) \in N(i,j)} w_{i,j,\bar{i},\bar{j}} \cdot (du^k_{\bar{i},\bar{j}} - du^k_{i,j}), \\
0 =& [\Psi'_D]_{i,j} \cdot ([J^k_{12}]_{i,j} du^k_{i,j} + [J^k_{22}]_{i,j} dv^k_{i,j} [J^k_{23}]_{i,j}) \\
&- \alpha \sum_{(\bar{i},\bar{j}) \in N(i,j)} w_{i,j,\bar{i},\bar{j}} \cdot (v^k_{\bar{i},\bar{j}} - v^k_{i,j}) \\
&- \alpha \sum_{(\bar{i},\bar{j}) \in N(i,j)} w_{i,j,\bar{i},\bar{j}} \cdot (dv^k_{\bar{i},\bar{j}} - dv^k_{i,j})
\end{aligned}
\tag{5.31}
$$

wobei die Summe der anisotropen Varianten über alle 8 Nachbarn geht und mit dem jeweiligen Gewicht $w_{i,j,\bar{i},\bar{j}}$ der Schablone $D_{i,j}$ multipliziert wird.

5.6 Lösung

Da sich alle anisotropen Methoden nur durch den Diffusionstensor D unterscheiden, lassen sich die drei Methoden durch folgende Gleichung nach Umstellung auf den SOR-Löser berechnen. Dazu wird $D_{i,j}$ durch den jeweiligen Diffusionstensor aus Abschnitt 5.4 ersetzt und mit der Non-Negativity-Schablone diskretisiert:

$$du_{i,j}^{k,m,n+1} = (1-\omega)du_{i,j}^{k,m,n} + \omega \left(\frac{\xi_1 + \alpha \sum_{(\bar{i},\bar{j})\in N^-(i,j)} \xi_2 + \alpha \sum_{(\bar{i},\bar{j})\in N^+(i,j)} \xi_3}{[\Psi'_D]_{i,j}^m [J_{11}^k]_{i,j} + \alpha \sum_{(\bar{i},\bar{j})\in N(i,j)} w_{i,j,\bar{i},\bar{j}}} \right) \tag{5.32}$$

mit den Einträgen

$$\xi_1 = -[\Psi'_D]_{i,j}^m [J_{12}^k]_{i,j} dv^{k,m,n} - [\Psi'_D]_{i,j}^m [J_{13}^k]_{i,j}, \tag{5.33}$$

$$\xi_2 = w_{i,j,\bar{i},\bar{j}} \cdot (u_{\bar{i},\bar{j}}^{k,m,n+1} + du_{\bar{i},\bar{j}}^{k,m,n+1} - u_{i,j}^{k,m,n+1}), \tag{5.34}$$

$$\xi_3 = w_{i,j,\bar{i},\bar{j}} \cdot (u_{\bar{i},\bar{j}}^{k,m,n} + du_{\bar{i},\bar{j}}^{k,m,n} - u_{i,j}^{k,m,n}). \tag{5.35}$$

wobei $w_{i,j,\bar{i},\bar{j}}$ für die jeweilige Kombination der Einträge aus der Schablone der anisotropen Diskretisierung steht. Die Lösung für v kann analog zu u durchgeführt werden.

6 Evaluation

Im nächsten Abschnitt erfolgt eine Evaluation der entwickelten Algorithmen, in der die Stärken und Schwächen der einzelnen Methoden betrachtet werden. Hierfür werden verschiedene Experimente durchgeführt und diese anhand zweier Fehlermaße miteinander verglichen. Der Fokus liegt dabei auf den Auswirkungen der Segmentierung, die in den meisten Methoden Einzug gefunden hat.

6.1 Experimente und Fehlermaße

Um die berechneten Ergebnisse qualitativ einschätzen zu können, werden Fehlermaße benötigt. Häufig werden Messverfahren benutzt, die den berechneten Fluss mit einer sogenannten Ground Truth abgleichen. Die Ground Truth ist die perfekte Lösung des Korrespondenzproblems eines Bilderpaares und wird zu Vergleichszwecken mit einigen Datensätzen mitgeliefert. Hierbei handelt es sich meist um synthetische erzeugte Sequenzen. In der Regel werden die beiden folgenden Fehlermaße verwendet.

Durchschnittlicher Winkelfehler

Der durchschnittliche Winkelfehler (auch Average Angular Error oder AAE) vergleicht die einzelnen Flussvektoren mit denen der Ground Truth bereitgestellten Vektoren und bestraft diese anhand Abweichungen des Winkels. Dabei wird der Winkel für die Bewegung jedes Pixels des Bildes verglichen und aufsummiert. Der durchschnittliche Fehler wird anschließend erhalten, indem durch die Anzahl der Pixel $N \cdot M$ dividiert wird:

$$AAE(u^t, u^e) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} arccos \left(\frac{u_{i,j}^t u_{i,j}^e + v_{i,j}^t v_{i,j}^e + 1}{\sqrt{(u_{i,j}^t)^2 + (v_{i,j}^t)^2 + 1} \sqrt{(u_{i,j}^e)^2 + (v_{i,j}^e)^2 + 1}} \right) \tag{6.1}$$

Durchschnittlicher Endpunktfehler

Der durchschnittliche Endpunktfehler (auch Average Endpoint Error oder AEE) vergleicht die Stelle des Endpunktes des Vektors mit dem der Ground Truth. Hierbei werden Abweichungen der Verschiebung von u und v mit den Flussvariablen der Ground Truth quadratisch bestraft.

$$AEE(u^t, u^e) = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \sqrt{(u^t_{i,j} - u^e_{i,j})^2 + (v^t_{i,j} - v^e_{i,j})^2} \tag{6.2}$$

6.1.1 Testsquenzen

Für die Evaluation wurden Testsequenzen des Middlebury Benchmarks [BSL+11] verwendet. Hierbei handelt es sich um speziell entwickelte Bildfolgen, die unterschiedliche Schwierigkeiten der Optischen-Fluss-Berechnung abdecken und die Verfahren so auf ihre allgemeine Tauglichkeit testen. Abbildung 6.1 zeigt alle verwendeten Bilderpaare mit der zugehörigen Ground Truth, die für die Evaluation verwendet wurden. Der Schwerpunkt der Sequenzen Dimetrodon, Hydrangea und Rubberwhale liegt dabei auf der versteckten Textur, wobei Urban2 und Grove durch eine größere Verschiebung hervorstehen.

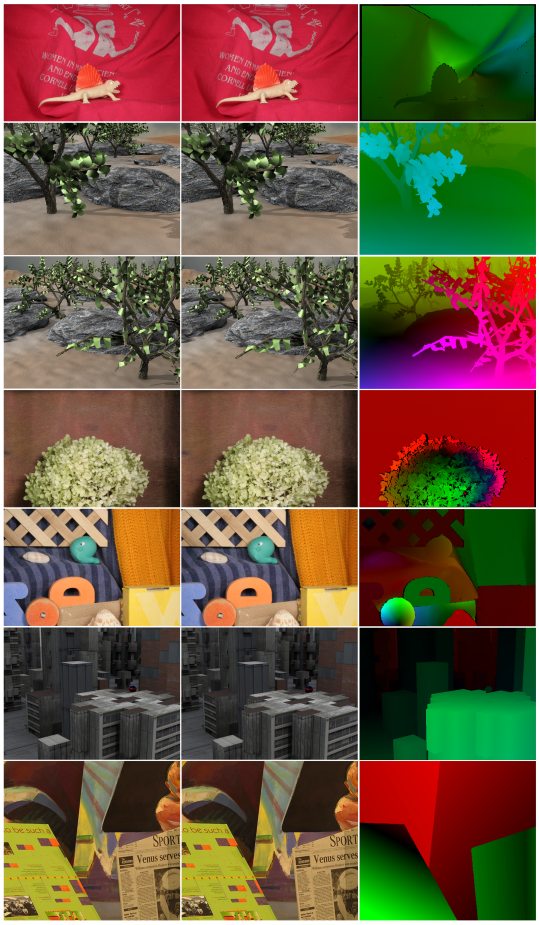


Abbildung 6.1: Testsequenzen des Middlebury Benchmarks. **Von links nach rechts:** Referenzbild, zweites Bild, Ground Truth. **Von oben nach unten:** Dimetrodon, Grove2, Grove3, Hydrangea, Rubberwhale, Urban2, Venus.

6.1.2 Segmentbasierte Verfahren

Der nächste Abschnitt stellt einen Vergleich zwischen der konstanten und affinen Variante sowie der Erweiterung der konstanten Methode durch einen vorberechneten Fluss auf. Dabei wird im ersten Teil der Evaluation auf die Fehlermessung eingegangen und ein näherer Einblick in die Qualität der Methoden gegeben. Daraufhin wird anhand von mehreren Testsequenzen ein visuelles Fazit gebildet, indem aufgezeigt wird, welche Stellen des Bildes für das entsprechende Abschneiden verantwortlich sind. Für ein einheitliches Ergebnis wurde für jedes Verfahren die besten Parameter ermittelt, die den kleinsten Fehler aufweisen und diese für alle Testsequenzen beibehalten, um eine allgemeine Repräsentation zu bieten. Für die Evaluation wurde ein Laptop mit einem Intel i5-4210U Prozessor und einem Takt von 1.7 GHz verwendet.

		Dimetrodon	Grove2	Grove3	Hydrangea
Konstant	AAE	17.1050	11.6073	19.9875	7.8138
	AEE	0.8449	0.7660	1.8828	**0.6841**
	Time	0:20:242	0:36:310	0:37:002	0:25:403
		Rubberwhale	Urban2	Venus	
	AAE	18.5826	15.7118	14.1053	
	AEE	0.5792	2.7916	1.2423	
	Time	0:22:107	0:34:006	0:17:383	
Affin		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	19.1193	15.3375	17.6098	28.7923
	AEE	1.1058	1.2585	4.4363	2.9219
	Time	0:21:254	0:45:999	0:50:789	0:25:403
		Rubberwhale	Urban2	Venus	
	AAE	20.9855	15.8162	17.6152	
	AEE	0.6537	3.2496	2.8653	
	Time	0:23:673	1:01:609	0:31:348	
Vorberechnet		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	**7.5278**	**7.4612**	**7.8091**	**7.2409**
	AEE	**0.4008**	**0.5056**	**0.5202**	0.8428
	Time	0:37:266	1:00:449	0:57:675	0:42:855
		Rubberwhale	Urban2	Venus	
	AAE	**12.8743**	**11.3492**	**12.7438**	
	AEE	**0.4058**	**1.6443**	**0.8965**	
	Time	0:39:541	0:59:556	0:31:480	

Tabelle 6.1: Vergleich der segmentbasierten Methoden anhand der Fehlermessverfahren AAE und AEE für alle bisher vorgestellten Testsequenzen des Middlebury Benchmarks mit zusätzlicher Laufzeit der Verfahren. Die beste Lösung wurde unterstrichen und fett markiert.

Im nächsten Abschnitt folgt einer genaue Analyse der Methoden aus Tabelle 6.1 mithilfe einer Reihe an Testsequenzen, die in Abbildung 6.2, 6.3, 6.4 und Abbildung 6.5 dargestellt werden. Die Parameter die dabei zu Berechnung benutzt wurden, lassen sich in Tabelle 6.2 nachlesen.

	α	ϵ_D	ϵ_S	h_s	h_r	h_f
Konstant	4	0.1	200	7	15	
Affin	100000	0.01	0.05	7	15	
Vorberechnet	17	0.1	170	7	15	2

Tabelle 6.2: Verwendete Parameter der drei Varianten, welche für die Bildsequenzen beibehalten wurden.

Konstante Methode In Abbildung 6.2 lässt sich die erste Problematik der konstanten Methode erkennen. Da die Segmentierung durch die minimalen farblichen Änderungen diese zu einem großem Segment zusammenfügt, werden viele unterschiedliche Regionen, die normalerweise einen verschiedenen Fluss haben, durch ein einziges Segment repräsentiert. Diese lassen sich damit nicht darstellen, da alle Pixel in einem Segment denselben Fluss haben. Im Beispiel der Decke entsteht somit ein Fluss, der nur an wenigen Stellen zutrifft. Die Stärken der Segmentierung lassen sich hingegen an den Objektkanten erkennen. Besonders gut ist dies an der Echse zu sehen, die sehr scharf erhalten wird, die Glattheit jedoch nicht exakt an die Nachbarn angepasst wird, wodurch größere Sprünge im Fluss entstehen. In Abbildung 6.3 ist eine klare Trennung der Objekte mit scharfen Kanten zu erkennen, die Segmente leiden jedoch auch hier an dem Flussverhalten innerhalb der Segmente und werden nicht korrekt dargestellt. Durch diese Faktoren entsteht ein recht hoher Winkel- und Endpunktfehler, da viele Bereiche der Segmente durch die konstante Verschiebung nicht zutreffend sind.

Affine Methode Die affine Methode löst die Problematik der konstanten Methode nur bedingt besser. Sie kann die Flüsse innerhalb der Segmente zwar durch lineare Änderung darstellen, diese ist in vielen Fällen jedoch nicht ausreichend um viele der komplexeren Flüsse darzustellen. In Abbildung 6.2 ist der Übergang der Segmente zwar etwas besser gelungen, womit ein leichter Verlauf des Flusses über das Segment der Decke entstanden ist, da dies jedoch als Mittelung aller Flüsse der Pixel innerhalb des Segments angesehen werden kann, trifft diese Berechnung nur an wenigen Stellen zu. In Bezug auf die Kanten wurden diese auch hier sehr gut erhalten. Die Umstellung auf eine affine Modellierung hat an einigen Stellen zu einer besseren Qualität geführt, durch die Möglichkeit den Fluss mit einem Verlauf darzustellen. Doch gerade diese Eigenschaft, führt bei einigen Bildern auch zu den größten Fehlern, da große Segmente wie in Abbildung 6.4 an viele unterschiedliche Flüsse angrenzen und so leicht verfälscht werden können.

Vorberechnete Methode Die dritte Methode geht die Granularität der Segmente mit einem vorberechneten pixelbasierten Fluss an und trennt bereits vor dem eigentlichen Verfahren die Segmente auf und lässt damit eine genauere Verteilung zu. Wie in Abbildung 6.2 zu erkennen ist, erhält die zuvor aus einem Segment bestehende Decke so weitere Segmente an den Stellen, wo ein anderer Fluss auftritt. So können viele Flüsse, die zuvor durch den kleinen farblichen Unterschied von dem Mean-Shift-Verfahren zu einem großen Segment zusammengefügt wurden, nun separat berechnet werden.

Mit steigender Anzahl an Unterteilungen ist so eine bessere Berechnung komplexer Bewegungen möglich. Das gleiche Ergebnis ist zudem bei Abbildung 6.3 zu erkennen. Das Rad sowie der Karton im unteren Bildbereich erhalten zusätzliche Segmente an diesen Stellen, wodurch die Rotation besser darzustellen ist. Zusätzlich werden Segmente besser erhalten, wie an der Muschel zu erkennen ist. Die Flusskomponente trägt dort zu einer stärkeren Separation des Hintergrunds bei und erhält die Form besser. Ein Nachteil dieser Methode ist allerdings, dass sie stark von der pixelbasierten Berechnung abhängt. Ist diese ungenau, entstehen besonders an den Objektkanten neue zusätzliche Segmente die bei der späteren segmentbasierten Berechnung zu Fehler führen können.

Zusammenfassung Zu den segmentbasierten Methoden lässt sich abschließend sagen, dass die Vorberechnung des Flusses einen deutlichen Vorteil, gegenüber den herkömmlichen Varianten in Bezug auf die Granularität der Segmente, bietet. Dies spiegelt sich auch deutlich in Tabelle 6.1 wider. Methode drei erlangt hier bei allen getesteten Sequenzen die besten Werte bei ähnlicher Laufzeit, verglichen mit der affinen Methode.

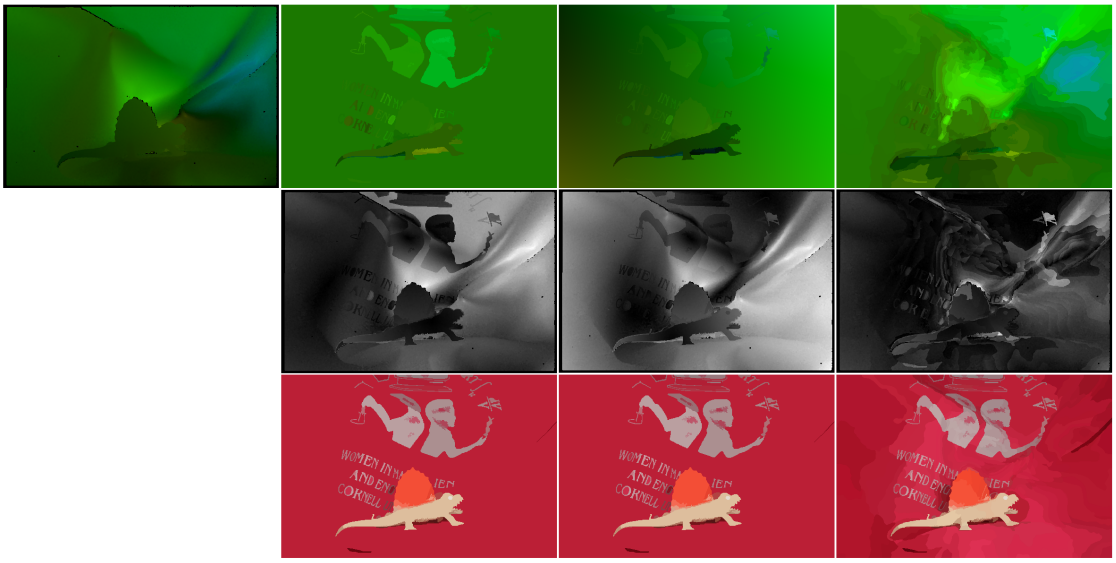


Abbildung 6.2: Ergebnisse der segmentbasierten Varianten der Dimetrodon Testsequenz. **Von links nach rechts:** Ground Truth, konstante, affine und vorberechnete Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

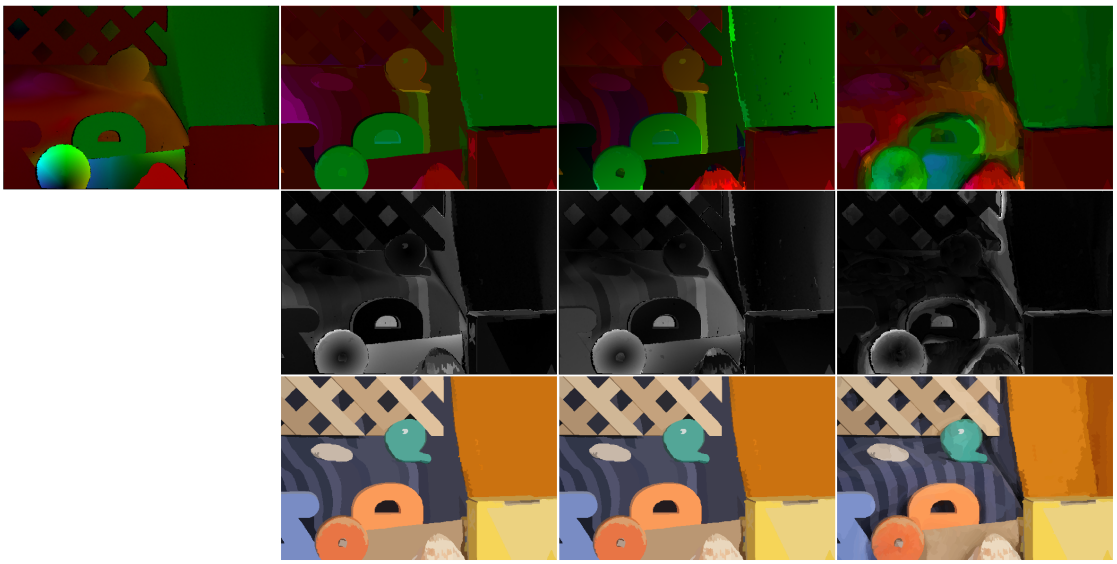


Abbildung 6.3: Ergebnisse der segmentbasierten Varianten der Rubberhwale Testsequenz. **Von links nach rechts:** Ground Truth, konstante, affine und vorberechnete Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

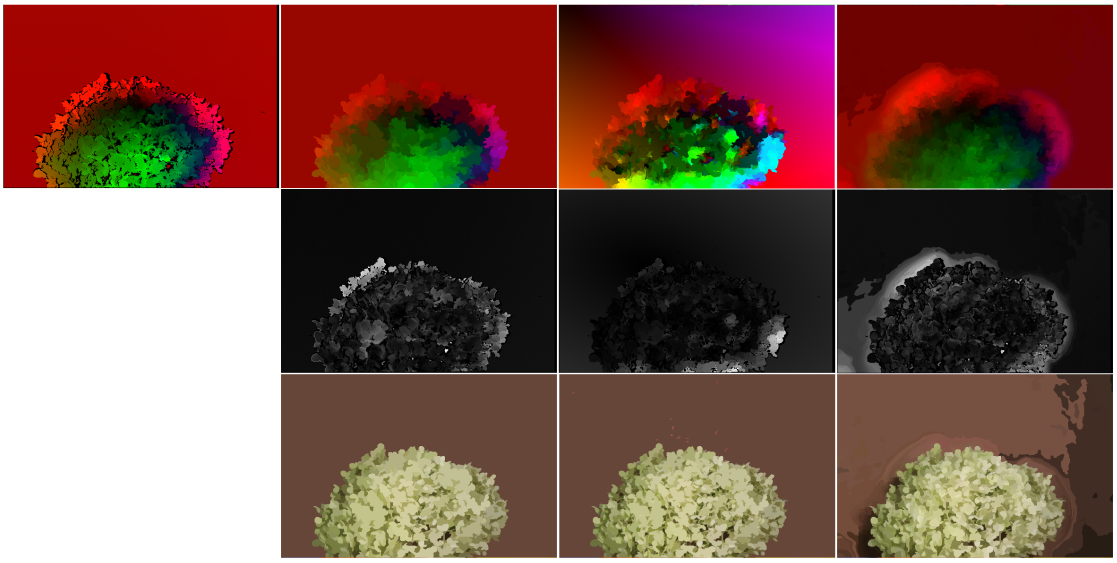


Abbildung 6.4: Ergebnisse der segmentbasierten Varianten der Hydrangea Testsequenz. **Von links nach rechts:** Ground Truth, konstante, affine und vorberechnete Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

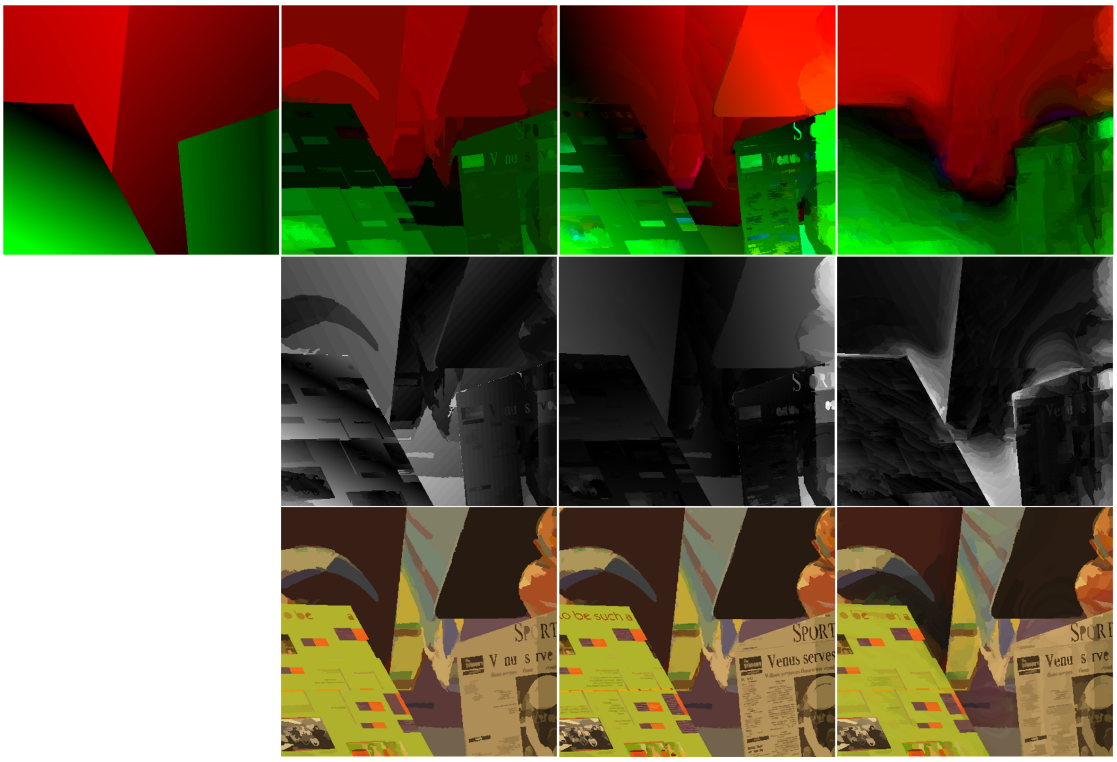


Abbildung 6.5: Ergebnisse der segmentbasierten Varianten der Venus Testsequenz. **Von links nach rechts:** Ground Truth, konstante, affine und vorberechnete Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

6.1.3 Pixelbasierte Verfahren

Im nächsten Abschnitt erfolgt ein Vergleich zwischen den pixelbasierten Verfahren, zu welchen auch das Basisverfahren aus Kapitel 3 gehört und zu Vergleichszwecken hier ebenfalls getestet wird. Wie bereits bei den segmentbasierten Methoden wurden die Verfahren durch die bereits bekannten Testsequenzen von Middlebury evaluiert und die Ergebnisse in Tabelle 6.3 zusammengefasst. Zudem erfolgt ein Vergleich anhand der Bilder und es wird auf die jeweiligen Stärken und Schwächen der Methoden im Zusammenhang mit der Segmentierung eingegangen. Die verwendeten Parameter der Methoden sind in Tabelle 6.4 wiederzufinden und wurden ebenfalls für jede de vier Varianten allgemein für alle getesteten Sequenzen optimiert.

Isotrop		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	3.0936	3.5877	8.0364	2.7819
	AEE	0.1604	0.2501	0.8716	0.2588
	Time	3:56:962	5:12:699	5:13:714	3:45:573
		Rubberwhale	Urban2	Venus	
	AAE	6.4409	5.5863	8.9861	
	AEE	0.1932	0.7440	0.5172	
	Time	3:52:595	5:21:199	2:37:914	
Anisotrop		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	3.5289	3.7730	7.7967	3.0602
	AEE	0.1799	0.2735	0.7956	0.2607
	Time	3:47:409	5:09:159	5:14:413	4:01:096
		Rubberwhale	Urban2	Venus	
	AAE	6.1196	4.8668	9.5868	
	AEE	0.1853	0.6190	0.5568	
	Time	3:49:441	5:06:510	2:48:258	
Kantenmap		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	3.5761	3.8208	7.8287	3.0724
	AEE	0.1823	0.2780	0.7922	0.2588
	Time	3:59:901	5:27:434	5:20:413	3:59:527
		Rubberwhale	Urban2	Venus	
	AAE	5.9096	5.2454	7.7568	
	AEE	0.1796	0.6881	0.4706	
	Time	3:58:177	5:29:633	2:51:023	
Komplementär		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	**3.0601**	**2.8015**	**6.7846**	**2.5961**
	AEE	**0.1584**	**0.1997**	**0.7335**	**0.2167**
	Time	3:56:601	5:26:935	5:23:887	4:05:420
		Rubberwhale	Urban2	Venus	
	AAE	**5.1879**	**3.3286**	**6.7696**	
	AEE	**0.1504**	**0.3733**	**0.3868**	
	Time	3:59:907	5:20:710	2:46:519	

Tabelle 6.3: Vergleich der pixelbasierten Methoden anhand der AAE und AEE Fehler für alle bisher vorgestellten Testsequenzen des Middlebury Benchmarks mit zusätzlicher Laufzeit der Verfahren. Die beste Lösung wurde unterstrichen und fett markiert.

Anisotrop Im Gegensatz zum Basisverfahren, das den Fluss nur durch die Stärke einer Kante regulierte, liefert die anisotrope Methode durch die zusätzliche Analyse der Richtung, eine bessere Erhaltung der Kanten. Da die Grundlage des Basisverfahrens allerdings generell zu einer Überglättung neigt, erfolgt die Diffusion weiterhin über einige Kanten, was beispielsweise in Abbildung 6.9 zu einem Verwischen der Kante an dem Gebäude im Vordergrund führt.

	α	ϵ_D	ϵ_S	h_s	h_r
Isotrop	10	0.1	200	7	15
Anisotrop	3	0.1	190	7	15
Kantenmap	3	0.1	200	7	15
Komplementär	15	0.1	180	7	15

Tabelle 6.4: Verwendete Parameter der vier Verfahren, welche für alle Bildsequenzen beibehalten wurden.

Kantenmap Die Methode der Kantenmap zeigt auf den ersten Blick visuell nur wenig Unterschiede zur anisotropen Methode und hat auch bei den meisten Testsequenzen mit ähnlichen Resultaten wie die anisotrope Methode abgeschnitten. Bei genauerer Betrachtung sind jedoch schärfere Kanten an den Objektübergängen in Abbildung 6.9 zu erkennen, welche allerdings durch die Überglättung über die Kanten hinaus ein wenig verschwimmen und so nur einen kleinen Unterschied in der Qualität ausmachen. Zudem hängt die Methode stark von einer korrekt berechneten Kantenmap ab, da fehlende oder zusätzliche Kanten keine Auswirkung auf den Fluss haben oder ihn stören können. Tabelle 6.3 zeigt dass in manchen Testsequenzen zwar ein etwas besseres Ergebnis vorhanden ist, in den anderen Bildern jedoch die anisotrope Methode auf Augenhöhe ist.

Komplementär Die dritte Methode zeigt besonders bei den Kanten seine Stärken und erzeugt durch die Kombination von bildgetriebener Richtung und einer flussgetriebenen Intensität der Diffusion ein äußerst scharfes Bild an den Objektübergängen. Die Überglättung, welche in den bisherigen drei Methoden vorhanden war, kann durch die Segmentierung an vielen Stellen komplett verhindert werden. Gegenüber herkömmlichen bildgetriebenen Methoden kann zudem ein glatteres Ergebnis erzeugt werden, da keine Übersegmentierung entsteht und meist nur Objektkanten vorhanden sind. Wie in Abbildung 6.9 an den Häusern zu erkennen ist, ist die Diffusion über die Gebäudekanten fast vollständig verschwunden. In Abbildung 6.6 ist dies ebenfalls am Stamm des Baumes zu erkennen der sich deutlicher vom Hintergrund trennt.

Zusammenfassung Bei den hier vorgestellten pixelbasierten Methoden geht die komplementäre Variante als klarer Sieger hervor und erzeugt in Bezug auf die Kanten die besten Ergebnisse. Dies lässt sich auch durch Tabelle 6.3 bestätigen indem die komplementäre Variante in allen getesteten Sequenzen ausnahmslos die Besten Ergebnisse erzielt, bei nur geringfügig längerer Laufzeit.

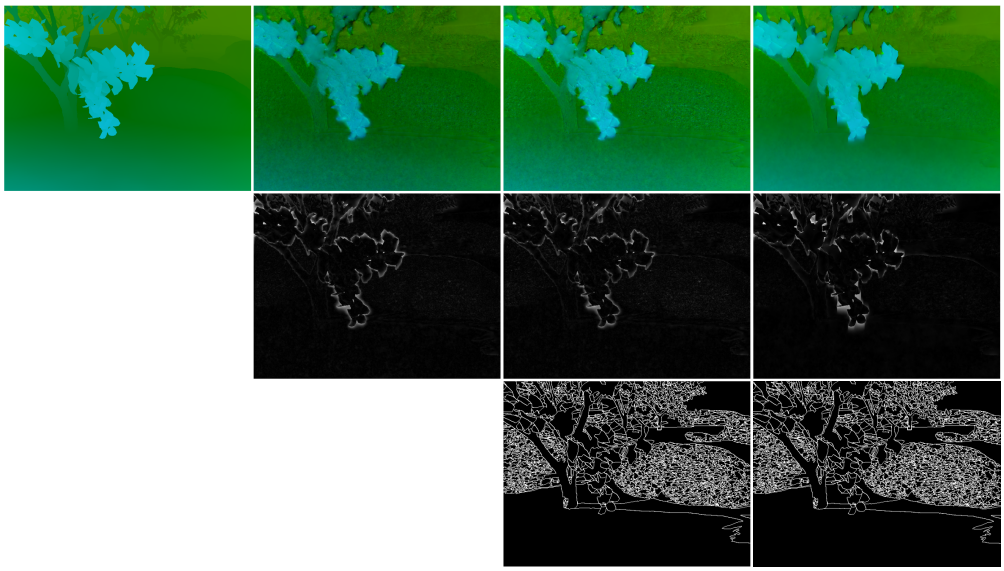


Abbildung 6.6: Ergebnisse der pixelbasierten Varianten der Grove2 Testsequenz. **Von links nach rechts:** Ground Truth, anisotrope, Kantenmap und komplementäre Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

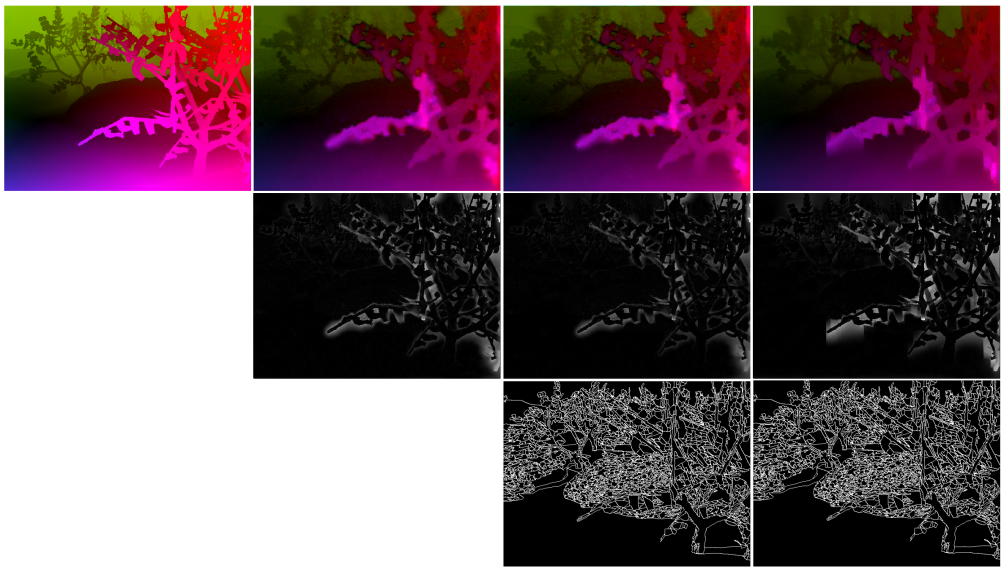


Abbildung 6.7: Ergebnisse der pixelbasierten Varianten der Grove3 Testsequenz. **Von links nach rechts:** Ground Truth, anisotrope, Kantenmap und komplementäre Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

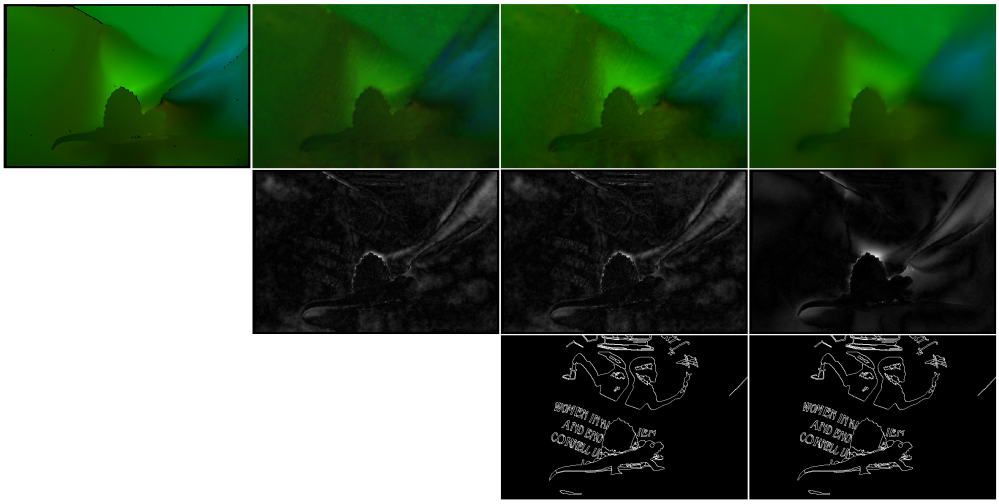


Abbildung 6.8: Ergebnisse der pixelbasierten Varianten der Dimetrodon Testsequenz. **Von links nach rechts:** Ground Truth, anisotrope, Kantenmap und komplementäre Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

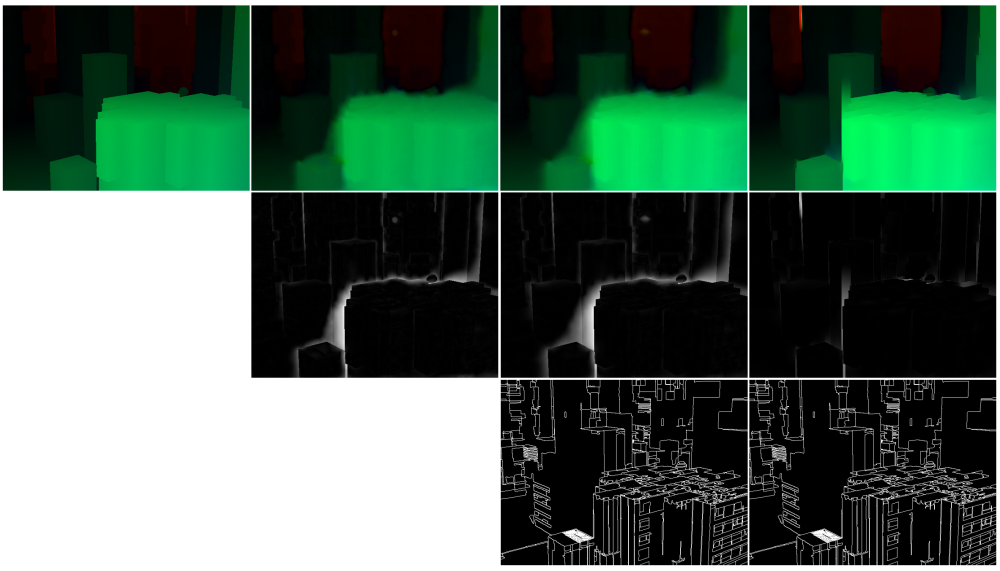


Abbildung 6.9: Ergebnisse der pixelbasierten Varianten der Urban2 Testsequenz. **Von links nach rechts:** Ground Truth, anisotrope, Kantenmap und komplementäre Methode, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

6.1.4 Vergleich segmentbasierter und pixelbasierter Verfahren

In diesem Abschnitt wird einer kleiner Vergleich zwischen den segmentbasierten und den pixelbasierten Methoden vorgenommen, wobei jeweils die beste Methode der jeweiligen Art gegenübergestellt wird. Die Ergebnisse der Methoden sind zudem in Tabelle 6.5 nochmals aufgeführt.

		Dimetrodon	Grove2	Grove3	Hydrangea
Vorberechnet	AAE	7.5278	7.4612	7.8091	7.2409
	AEE	0.4008	0.5056	0.5202	0.8428
	Time	**0:37:266**	**1:00:449**	**0:57:675**	**0:42:855**
		Rubberwhale	Urban2	Venus	
	AAE	12.9756	11.3492	12.7438	
	AEE	0.4127	1.6443	0.8965	
	Time	**0:41:163**	**0:59:556**	**0:31:480**	
Komplementär		Dimetrodon	Grove2	Grove3	Hydrangea
	AAE	**3.0601**	**2.8015**	**6.7846**	**2.5961**
	AEE	**0.1584**	**0.1997**	**0.7335**	**0.2167**
	Time	3:56:601	5:26:935	5:23:887	4:05:420
		Rubberwhale	Urban2	Venus	
	AAE	**5.1879**	**3.3286**	**6.7696**	
	AEE	**0.1504**	**0.3733**	**0.3868**	
	Time	3:59:907	5:20:710	2:46:519	

Tabelle 6.5: Vergleich der segmentbasierten Methode mit vorberechnetem Fluss und der komplementären pixelbasierten Methode anhand des AAE und AEE Fehlers. Die Beste Lösung wurde unterstrichen und fett markiert.

Wie in dem bisherigen Teil der Evaluation zu sehen war, haben die segmentbasierten Verfahren oft den Nachteil, dass komplexe Verschiebungen wie eine Vergrößerung oder Rotation zu Schwierigkeiten in der Berechnung dieser führt. Die affine Methode konnte dies nur Bedingt verbessern und führte zu ähnlichen Problemen wie die konstante Methode. Durch die Vorberechnung des Flusses durch eine pixelbasierte Methode konnte dies zwar etwas besser dargestellt werden, da die Vorberechnung allerdings nicht perfekt ist, können so neue Fehlerquellen in der segmentbasierten Variante entstehen. Durch die Eingrenzung der Objekte durch Segmente, entstehen jedoch sehr scharfe Kanten ohne diese im Funktional besonders zu berücksichtigen, dabei können jedoch Fehler in der Glattheit der Segmente auftreten. Zudem verringert sich die Anzahl der Variablen erheblich was zu einer drastischen Reduzierung der Laufzeit führt. Im Schnitt war so die segmentbasierte Variante um den Faktor 5.7 schneller als die pixelbasierte Methode. Die segmentbasierten Varianten werden jedoch durch die pixelbasierten Methoden qualitativ in den Schatten gestellt. So kann die komplementäre Methode die Objekte oft detaillierter berechnen als dies bei den Segmenten möglich ist. Zudem können mit Hilfe der Kanten des segmentierten Bildes zusätzlich sehr scharfe Kanten erzeugt werden. Abschließend lässt sich über die zwei verschiedenen Vefahrensansätze sagen, dass die hier getesteten segmentbasierten

Verfahren eher in zeitkritischen Anwendungen ihren Einsatz finden, da sie eine sehr schnelle Laufzeit besitzen. Diese können jedoch nicht an die Qualität der pixelbasierten Methoden anknüpfen.

6.1.5 Einfluss der Segmentgröße

Im nächsten Abschnitt wird der Einfluss der Segmentgröße auf die Qualität der Bilder untersucht. Dafür wurde der Fluss aller Testsequenzen mit der komplementären Methode und drei verschiedenen Kantenmaps berechnet, welche sich jeweils durch die Größe und dementsprechend ihrer Anzahl der Segmente unterscheiden.

Komplementär	Dimetrodon	Segmentzahl	102	121	302
		AAE	3.0601	3.0601	**3.0478**
		AEE	0.1584	0.1584	**0.1580**
	Grove2	Segmentzahl	855	1033	1333
		AAE	2.8281	**2.8015**	2.8533
		AEE	**0.1991**	0.1997	0.2013
	Grove3	Segmentzahl	1019	1260	1573
		AAE	6.7931	**6.7846**	6.7908
		AEE	0.7377	0.7335	**0.7329**
	Hydrangea	Segmentzahl	218	389	648
		AAE	2.6179	**2.5961**	2.6016
		AEE	0.2188	**0.2167**	0.2190
	Rubberwhale	Segmentzahl	83	149	1796
		AAE	5.5221	5.2773	**5.2068**
		AEE	0.1617	0.1553	**0.1530**
	Urban2	Segmentzahl	124	411	1796
		AAE	3.4559	3.3286	**3.1249**
		AEE	0.4232	0.3733	**0.3704**
	Venus	Segmentzahl	202	261	484
		AAE	**6.7541**	6.7696	6.8437
		AEE	0.3897	**0.3868**	0.3928

Tabelle 6.6: Vergleich der komplementären Methode mit unterschiedlicher Anzahl an Segmenten. Die beste Lösung wurde unterstrichen und fett markiert.

In Tabelle 6.6 sind die Ergebnisse der verwendeten Bilder abgebildet. Zu sehen ist, dass bei den meisten Bildern eine Verbesserung der Qualität durch eine Erhöhung der Segmente stattfindet. Dies lässt sich dadurch erklären, dass speziell komplexere Objekte aus vielen Segmenten bestehen und bei einer größeren Segmentierung, einige Teile vom Objekt abgeschnitten werden können und einem falschen Objekt zugeordnet werden. In Abbildung 6.10 ist dies gut an dem Gebäude im Vordergrund zu erkennen. Im hinteren Bereich des Gebäudes kommen mit steigender Anzahl an Segmenten neue Teile

hinzu, welche zuvor entfernt wurden. Der Nachteil einer Übersegmentierung hat allerdings zur Folge, dass viele Kanten innerhalb der Objekte hinzukommen und so die Diffusion auch an diesem Stellen gebremst wird, was zu Artefakten führen kann. In vielen Fällen bringt die Erhaltung der Objekte jedoch eine größere Verbesserung, als die durch Übersegmentierung verursachte Verschlechterung. In den hier getesteten Bildern, führte deshalb eine Erhöhung der Segmente im Schnitt zu einem besseren Ergebnis.

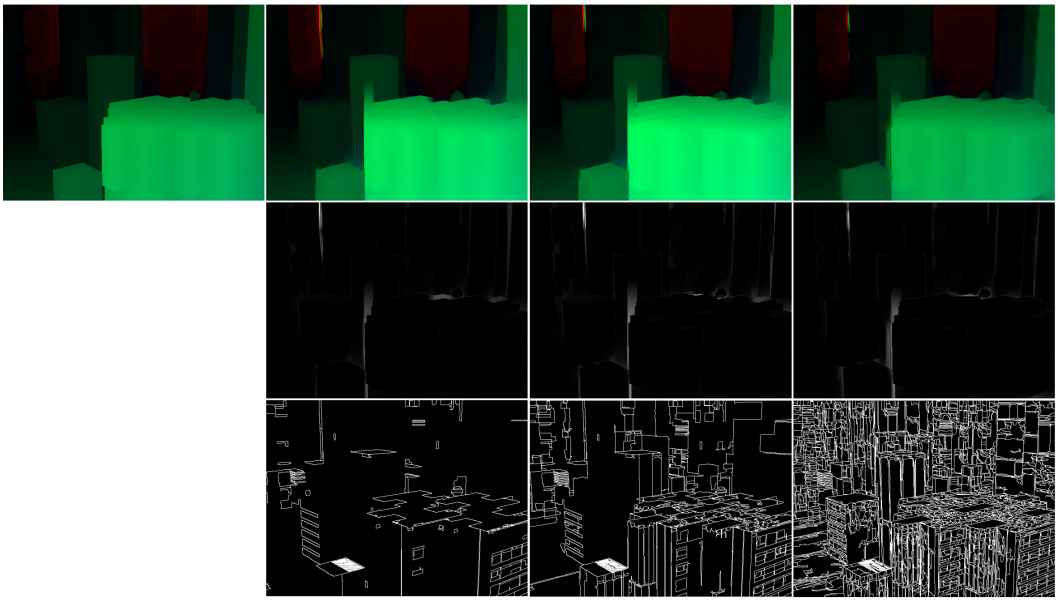


Abbildung 6.10: Ergebnisse der komplementären Variante der Urban2 Testsequenz bei einer wechselnden Anzahl und Größe an Segmenten. **Von links nach rechts:** Ground Truth, Kantenmap mit wenigen Segmenten, Kantenmap mit bisher verwendeten Segmenten und Kantenmap mit vielen Segmenten, **Von oben nach unten:** Flussfeld, Fehler, Segmentierung der Bilder.

6.1.6 Vergleich mit anderen Verfahren

In diesem Abschnitt wird die Arbeit mit einigen aktuellen Verfahren aus der Literatur verglichen. Da die komplentäre pixelbasierte Methode die besten Ergebnisse lieferte, wird diese Variante herangezogen. Für den Vergleich werden die bisherigen Testsequenzen verwendet und nach dem durchschnittlichen Endpunktfehler ausgewertet. Die Fehler inklusive des durchschnittlichen Fehlers sind in Tabelle 6.7 abgebildet. Wie zu sehen ist, ist die Schätzgenauigkeit des entwickelten Verfahrens trotz Verbesserung gegenüber dem Basisverfahren etwas schlechter als die aktuellen Verfahren der Literatur. Da diese jedoch zusätzliche Konzepte wie die Fundamentalmatrix [WCPB09], mehrere Eingangsbilder [VBVZ11], [SSB12] oder stückweise parametrische Bewegung in Form von Homographiemodellen verwenden, besteht berechtigte Hoffnung, dass sich die entwickelten Modelle noch weiter verbessern lassen.

	Avg.	Dimetrodon	Grove2	Grove3	Hydrangea	Rubberwhale	Urban2	Venus
Dieses Verfahren	0.317	0.158	0.199	0.733	0.216	0.155	0.373	0.386
Wedel *et al.* [WCPB09]	0.237	0.190	0.140	0.560	0.150	0.080	0.290	0.250
Volz *et al.* [VBVZ11]	0.209	X	0.114	0.540	0.134	0.071	0.187	X
Yang *et al.* [YL15]	0.180	0.118	0.095	0.445	0.146	0.072	0.196	0.190
Sun *et al.* [SSB12]	0.163	0.126	0.080	0.336	0.175	0.062	0.175	0.191

Tabelle 6.7: Vergleich der pixelbasierten komplementären Methode mit anderen aktuellen Verfahren der Literatur.

7 Zusammenfassung und Ausblick

In diesem Kapitel folgt eine Zusammenfassung der Diplomarbeit in der die Abschnitte noch einmal besprochen werden und ein kleines Fazit erstellt wird. Im Anschluss gibt es einen kurzen Ausblick der mögliche Ansatzpunkte für Erweiterungen der Verfahren thematisiert.

Zusammenfassung

Das Ziel dieser Arbeit war es die Berechnung des Optischen Flusses durch verschiedene Varianten mit den Vorteilen der Segmentierung zu kombinieren. Zugrunde lag ein pixelbasiertes Basisverfahren, mit robustem Daten- sowie Glattheitsterm, das durch weitere Terme und Verfahren erweitert wurde. Dabei wurden zwei verschiedene Methodiken entwickelt, die die Modellierung des Flusses verschieden angehen. Als erstes lag der Schwerpunkt auf den segmentbasierten Varianten. Hier wurden die Bilder zuerst mit dem Mean-Shift-Verfahren segmentiert, anschließend durch einen Region-Merging-Algotihmus kleine Segmente entfernt und jedem Pixel innerhalb des Segments die gleiche Flussvariable zugeordnet. Dies führte zu einer starken Reduzierung an Variablen und wirkte sich dementsprechend positiv auf die Laufzeit aus und lies eine genaue Erhaltung der Kanten zu. Um die starre Bewegung der ersten Methode zu verbessern, wurde eine affine Modellierung innerhalb der Segmente eingeführt um dort einen variableren Fluss zuzulassen. Erweitert wurde dies mit einer Vorberechnung durch eine simple pixelbasierte Methode, um die Segmente weiter zu trennen und die Granularität zu erhöhen. Der zweite Schwerpunkt lag auf den pixelbasierten Methoden, welche zur besseren Erhaltung der Kanten durch eine Kantenmap unterstützt wurden. Dabei wurde auf einen anisotropen Glattheitsterm umgestellt, um die Diffusion richtungsabhängig zu behandeln. Um die Diffusion an unerwünschten Stellen noch stärker zu stoppen, wurden die Segmentkanten als Stoppkriterium verwendet. Die letzte Methode kombinierte schließlich die bild- und flussgetriebenen Verfahren miteinander, indem die Richtung der Diffusion aus den segmentierten Bildern gelesen und anhand dieser, eine unterschiedliche Behandlung der Bildstruktur durchgeführt wurde. Die Evaluation hat am Ende gezeigt, dass die segmentbasierten Methoden eine schnelle Laufzeit im Vergleich zu den pixelbasierten Methoden hatten, die Kanten gut erhielten, bei der Glattheit der Segmente jedoch Probleme verursachten, wodurch grobe Übergänge entstanden. Die pixelbasierten Methoden konnten durch ihre feinere Granularität ein glatteres und allgemein besseres Ergebnis berechnen, was durch die Ergänzung der Kantenmap zu einer besseren Qualität der Kanten führte. Dies hatte jedoch im Gegensatz zu den segmentbasierten Methoden eine deutlich höhere Laufzeit zur Folge. Die komplementäre Methode zeigte schließlich, dass Kanten durch eine vorherigen Segmentierung auch bei einer pixelbasierten Methode sehr gut erhalten werden können. Verdeutlicht wurde dies dadurch, dass bei allen Testsequenzen so das beste Ergebnis berechnet wurde.

Ausblick

In diesem Abschnitt werden mögliche Aspekte zur Verbesserung der segment- sowie pixelbasierten Varianten vorgeschlagen, welche die Qualität weiter verbessern können.

Segmentbasiert Methoden

Glattheit Die segmentbasierten Methoden hatten Schwierigkeiten bei den Segmenten für einen glatten Übergang zu sorgen. Oft wurde eine komplett andere Richtung berechnet oder bei einem höheren Wert des Regularisierungsparameters gleich eine Vielzahl an Segmenten an den Fluss eines Segments angepasst, sodass dort eine starke Überglättung entstand. Durch die unterschiedliche Anzahl an Nachbarn, ist es an diesen Stellen schwierig eine Glattheit zu erstellen, die bei allen Nachbarn zu guten Ergebnissen führt. Um dies zu verbessern, kann eine Überprüfung der Segmentkanten vorgenommen werden wie von Vogel *et al.* [VSR13] vorgeschlagen. So kann die Glattheit zwischen den Segmenten angepasst und für einen flüssigeren Übergang gesorgt werden.

Segmentmodellierung Ein weiterer Schritt, der eine erhebliche Verbesserung der segmentbasierten Methoden zur Folge hat, ist eine geeigneteres Modell der Segmente, um die starre Verschiebung etwas zu entschärfen und so die Granularität zu erhöhen. Die Kombination aus einem vorberechneten Flussfeld durch ein pixelbasiertes Verfahren kommt diesem Problem zwar etwas entgegen, kann es aber nicht vollständig beheben, weshalb eine grundlegende Änderung der Modellieren notwendig ist. Black und Jepson [BJ96] gehen dieses Problem durch eine variable Modellierung an, welche Situationsabhängig auch eine höhere Ordnung für die Segmente zulässt. Eine weitere Methode wäre den Fluss nachträglich zu korrigieren, was Xu *et al.* [XCJ08] durch eine Konfidenzmap bewerkstelligen.

Pixelbasierte Methoden

Datenterm Für die pixelbasierten Methoden kann eine Erweiterung des Datenterms helfen, indem zusätzlich zu der Grauwertkonstanz eine Gradientkonstanz eingeführt wird, um globaler Illumination besser entgegen zu wirken und so schattige Bereiche besser berechnen zu können. Eine weiterer Ansatz ist die Verwendung von Basisfunktionen zu Schützung von Beleuchtungsunterschieden, wie Demetz *et al.* [DSV+14] vorschlagen.

Okklusion Bei der Verschiebung von Objekten werden oft dahinterliegende Objekte oder Teile des Hintergrunds verdeckt, welche in dem zweiten Bild nicht wiederzufinden sind, da diese erst durch die Verschiebung des Objekts im Vordergrund sichtbar wurden. Da hier keine genaue Schätzung des Flusses möglich ist und oft nur die Glattheit benachbarter Pixel übernommen wird, kann ein Okklusionsterm hinzugefügt werden, der die Energie an diesen Stellen besser regelt. Ein solcher Term wurde beispielsweise in [UWPB12] vorgeschlagen.

Literaturverzeichnis

[And14] F. Andres. Ein segmentierungsgestützter Variationsansatz zur Stereorekonstruktion. *Studienarbeit, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart*, 2014. (Zitiert auf Seite 7)

[BAS10] R. Ben-Ari, N. Sochen. Stereo matching with Mumford-Shah regularization and occlusion handling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):2071–2084, 2010. (Zitiert auf Seite 7)

[BBPW04] T. Brox, A. Bruhn, N. Papenberg, J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. European Conference on Computer Vision (ECCV)*, S. 25–36. Springer, 2004. (Zitiert auf Seite 38)

[BBW06] T. Brox, A. Bruhn, J. Weickert. Variational motion segmentation with level sets. In *Proc. European Conference on Computer Vision (ECCV)*, S. 471–483. Springer, 2006. (Zitiert auf Seite 7)

[BJ96] M. J. Black, A. D. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):972–986, 1996. (Zitiert auf den Seiten 6 und 102)

[BSL+11] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, R. Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. (Zitiert auf Seite 86)

[CBFAB94] P. Charbonnier, L. Blanc-Féraud, G. Aubert, M. Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proc. IEEE International Conference on Image Processing (ICIP)*, Band 2, S. 168–172. IEEE, 1994. (Zitiert auf Seite 31)

[CGM02] C. M. Christoudias, B. Georgescu, P. Meer. Synergism in low level vision. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, Band 4, S. 150–155. IEEE, 2002. (Zitiert auf Seite 57)

[CH93] R. Courant, D. Hilbert. *Methoden der mathematischen Physik*. Springer-Verlag, 1993. (Zitiert auf Seite 33)

[CJL+13] Z. Chen, H. Jin, Z. Lin, S. Cohen, Y. Wu. Large displacement optical flow from nearest neighbor fields. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 2443–2450. IEEE, 2013. (Zitiert auf Seite 7)

[CM02] D. Comaniciu, P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. (Zitiert auf Seite 54)

[CS05] D. Cremers, S. Soatto. Motion competition: A variational approach to piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, 2005. (Zitiert auf Seite 7)

[DSV+14] O. Demetz, M. Stoll, S. Volz, J. Weickert, A. Bruhn. Learning brightness transfer functions for the joint recovery of illumination changes and optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, S. 455–471. Springer, 2014. (Zitiert auf Seite 102)

[FG87] W. Förstner, E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data*, S. 281–305. 1987. (Zitiert auf Seite 76)

[Fra] H. Frank. RGB-Cube. http://upload.wikimedia.org/wikipedia/commons/0/03/RGB_farbwuerfel.jpg. (Zitiert auf Seite 21)

[Gib50] J. J. Gibson. *The perception of the visual world.* Houghton Mifflin, 1950. (Zitiert auf Seite 5)

[GT15] D. Gilbarg, N. S. Trudinger. *Elliptic Partial Differential Equations of Second Order.* Springer, 2015. (Zitiert auf Seite 37)

[HS81] B. K. Horn, B. G. Schunck. Determining optical flow. In *Artificial Intelligence*, Band 17, S. 185–203. 1981. (Zitiert auf Seite 25)

[JBJ96] S. X. Ju, M. J. Black, A. D. Jepson. Skin and bones: Multi-layer, locally affine, optical flow and regularization with transparency. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 307–314. IEEE, 1996. (Zitiert auf Seite 6)

[LK81] B. D. Lucas, T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Journal of Computer Vision*, Band 81, S. 674–679. 1981. (Zitiert auf Seite 25)

[Mei11] A. Meister. *Numerik linearer Gleichungssysteme.* Springer, 2011. (Zitiert auf Seite 50)

[NT] H. L. Nao-Team. Fussball-Roboter. http://www.htwk-leipzig.de/fileadmin/prorektorw/news/2013/HTWK_Leipzig_Nao_Team_Cebit2013C.jpg. (Zitiert auf Seite 5)

[NZZW10] J. Ning, L. Zhang, D. Zhang, C. Wu. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition*, 43(2):445–456, 2010. (Zitiert auf Seite 57)

[PM90] P. Perona, J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990. (Zitiert auf Seite 76)

[SSB12] D. Sun, E. B. Sudderth, M. J. Black. Layered segmentation and optical flow estimation over time. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 1768–1775. IEEE, 2012. (Zitiert auf den Seiten 7, 98 und 99)

[Ste] K. Steve. Lab-Cube. http://3.bp.blogspot.com/-YmBZR_8iMEs/UQm1NTktUXI/AAAAAAAAAXg/34KpXk2T1L8/s1600/lab.jpg. (Zitiert auf Seite 22)

[UWPB12] M. Unger, M. Werlberger, T. Pock, H. Bischof. Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 1878–1885. IEEE, 2012. (Zitiert auf den Seiten 7 und 102)

[VBVZ11] S. Volz, A. Bruhn, L. Valgaerts, H. Zimmer. Modeling temporal coherence for optical flow. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, S. 1116–1123. IEEE, 2011. (Zitiert auf den Seiten 98 und 99)

[VSR13] C. Vogel, K. Schindler, S. Roth. Piecewise rigid scene flow. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, S. 1377–1384. IEEE, 2013. (Zitiert auf den Seiten 7 und 102)

[WCPB09] A. Wedel, D. Cremers, T. Pock, H. Bischof. Structure-and motion-adaptive regularization for high accuracy optic flow. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, S. 1663–1668. IEEE, 2009. (Zitiert auf den Seiten 98 und 99)

[Wei98] J. Weickert. *Anisotropic Diffusion in Image Processing*, Band 1. Teubner Stuttgart, 1998. (Zitiert auf Seite 82)

[XCJ08] L. Xu, J. Chen, J. Jia. A segmentation based variational model for accurate optical flow estimation. In *Proc. European Conference on Computer Vision (ECCV)*, S. 671–684. Springer, 2008. (Zitiert auf den Seiten 6 und 102)

[YL15] J. Yang, H. Li. Dense, Accurate Optical Flow Estimation with Piecewise Parametric Model. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 1019–1027. 2015. (Zitiert auf Seite 99)

[ZBW+09] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn, H.-P. Seidel. Complementary optic flow. In *Proc. Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, S. 207–220. Springer, 2009. (Zitiert auf Seite 76)

Alle URLs wurden zuletzt am 17. 08. 2015 geprüft.

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