Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Berechnung des Szenenflusses mit Variationsansätzen

Steffen Nüssle

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. Andrés Bruhn
Betreuer:	Michael Stoll, M.Sc. M.Sc. B.Sc. Daniel Maurer, M.Sc. B.Sc.
Beginn am:	2. November 2016
Beendet am:	3. Mai 2017
CR-Nummer:	I.2.10, I.4.8

Kurzfassung

Die Berechnung des Szenenflusses ist momentan eines der anspruchsvollsten Themen im Bereich des Maschinensehens. Viele Szenenflussverfahren verwenden eine 2D-Parametrisierung, um die 3D-Bewegung einer Szene in der Bildebene zu bestimmen. Ist die Anzahl der Kameraansichten vorab nicht bekannt, haben derartige Verfahren jedoch den Nachteil, die zusätzlichen Informationen durch weitere Ansichten nicht verwenden zu können. Die in dieser Arbeit präsentierte Szenenflussmethode, basierend auf der Arbeit von Basha *et al.* [9], verwendet eine 3D-parametrisierte Punktewolke, um die Tiefe und die 3D-Bewegung an jedem Bildpunkt einer frei wählbaren Referenzkamera zu berechnen. Durch diese 3D-Parametrisierung wird eine einfache Erweiterung des Verfahrens auf eine beliebige Anzahl verschiedener Kameraansichten ermöglicht. Zusätzlich wird das vorgestellte Verfahren durch die Verwendung der Gradientenkonstanzannahme robust gegenüber Beleuchtungsunterschieden zwischen den jeweiligen Eingabebildern gemacht. In einer anschließenden Evaluation wird die Funktionstüchtigkeit dieser Konzepte untersucht und sichergestellt. Weiter wird in der Evaluation gezeigt, dass das Verfahren durch vorab kalibrierte Modellparameter immer noch vernünftige Ergebnisse in realitätsechten Szenen liefern kann.

Danksagungen

Ich möchte mich an dieser Stelle bei allen Personen bedanken, die zum Gelingen meiner Arbeit beigetragen haben.

- Großer Dank gebührt vor allem Prof. Dr.-Ing. *Andrés Bruhn*, der mich durch seine enthusiastische Vorlesungen für das Maschinensehen begeistern konnte und mir das Thema dieser Arbeit bereitgestellt hat.
- Weiter möchte ich mich bei meinen Betreuern *Michael Stoll* und *Daniel Maurer* für die wertvolle Unterstützung bedanken.
- *Ulrike Rieger* und *Gerd Plagemann* möchte ich meinen Dank für das fortwährende Korrekturlesen aussprechen.
- Als letztes möchte ich meinen Eltern *Monika* und *Artur Nüssle*, die mir dieses Studium ermöglicht haben, danken.

Oberjesingen, den 03. Mai 2017

Steffen Nüssle

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Der Szenenfluss beschreibt die dreidimensionale Bewegung von Oberflächenpunkten in einer dreidimensionalen Szene. Mit der ersten wissenschaftlichen Veröffentlichung von Vedula *et al.* [30] ist die Berechnung des Szenenflusses seit nunmehr 17 Jahren ein zentrales und herausforderndes Forschungsthema im Bereich des Maschinensehens (Computer Vision). Obwohl es mittlerweile eine Vielzahl an Methoden für das Berechnen des Szenenflusses gibt, scheinen die Grenzen des Machbaren an Präzision und Effizienz noch lange nicht erreicht [33]. Des Weiteren existieren in realitätsechten Szenen noch ungelöste Probleme, so dass deren Szenenfluss noch von keiner bisher bekannten Methode zufriedenstellend gut berechnet werden konnte [21].

Die Kenntnis über 3D-Bewegungen im Raum kann vielfältig eingesetzt werden. Stoyanov [28] beschreibt ein solches System, welches bei minimal-invasiven Operationen assistiert. Arbeiten von Lenz *et al.* [18] sowie von Menze *et al.* [21, 22] verwenden den Szenenfluss zur Detektion von Personen und Objekten im Straßenverkehr. Ein Beispiel für solch ein Szenario ist in Abbildung 1.1 dargestellt. Aktive Fahrassistenzsysteme und autonome Fahrzeuge können anhand der Daten z.B. Maßnahmen gegen eine eventuelle Kollision ergreifen. Lui *et al.* [19] benutzt u.a. den Szenenfluss, um die Kopfhaltung einer Person zu erfassen. Diese Information wiederum erlaubt es, Rückschlüsse bezüglich des Aufmerksamkeitszentrums einer Person zu machen und ermöglicht zudem, z.B. über eine Kopfsteuerung, eine Alternative zu traditionellen Eingabegeräten für körperlich benachteiligte Personen.

Für die Berechnung des Szenenflusses werden häufig zwei zeitlich kurz aufeinanderfolgende Stereosequenzen verwendet. Eine Stereosequenz besteht aus zwei oder mehr Bildern einer Szene, die zeitgleich von unterschiedlich positionierten Kameras aufgenommen wurden. Damit die Bewegungen der einzelnen Oberflächenpunkte verfolgt werden können, ist es notwendig, die Bildpunkte ihrem jeweiligen Pendant

in den anderen Bildern zuzuordnen. Für derartige Korrespondenzprobleme hat sich die Verwendung von kontinuierlichen globalen Optimierungsmethoden, sogenannte Variationsansätze, etabliert. Ein Variationsansatz minimiert hierfür ein Energiefunktional, das Abweichungen von geeigneten Konstanz- und Glattheitsannahmen bestraft. Da der optische Fluss als die Projektion des Szenenflusses auf die Bildebene verstanden werden kann, bedienen sich viele Methoden einer 2D-Parametrisierung, die die Berechnung des Szenenflusses im 2D-Bildbereich ausführt. Diese Vorgehensweise hat den Vorteil, dass ähnliche Konzepte wie aus dem Bereich des optischen Flusses auf die Szenenflussberechnung übertragen werden können. Ein Nachteil dieser Methode ist jedoch, dass sich eine Erweiterung auf zusätzliche Ansichten als schwierig gestaltet, da die Projektion des Szenenflusses abhängig von der Position der jeweiligen Kamera ist. Solch eine Erweiterung kann wünschenswert sein, da sie für gewöhnlich die Häufigkeit an nicht eindeutigen Korrespondenzen reduziert und somit die Gesamtqualität der Ergebnisse verbessern kann [9]. Abhilfe schafft eine 3D-Parametrisierung, die die Tiefe und den Szenenfluss direkt im 3D-Raum, ausgehend von einer frei wählbaren Referenzkamera, an jedem Bildpunkt modelliert.

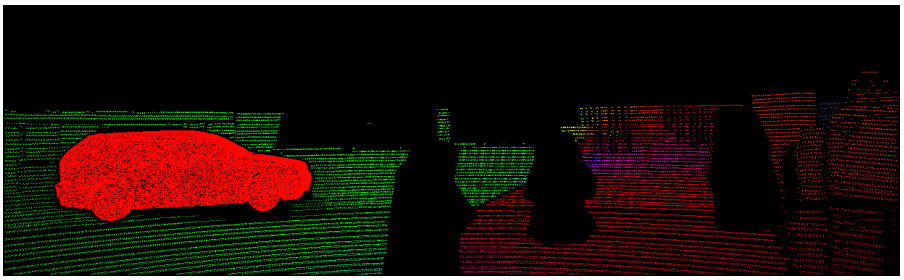
1.2. Zielsetzung

Das Ziel dieser Arbeit ist es, eine tiefengetriebene Szenenflussmethode basierend auf der Veröffentlichung von Basha *et al.* [9] zu entwickeln. Hierfür steht ein Lehrstuhlframework der Universität Stuttgart, Abteilung Intelligente Systeme, zur Extraktion des optischen Flusses zur Verfügung. Das Framework soll um die entsprechenden Funktionen zur Berechnung des Szenenflusses erweitert werden. Der linearisierte Datenterm wird hierfür in die Tensornotation [5, 6] integriert und das entwickelte Szenenflussverfahren wird abschließend mittels verschiedener Datensätze evaluiert.

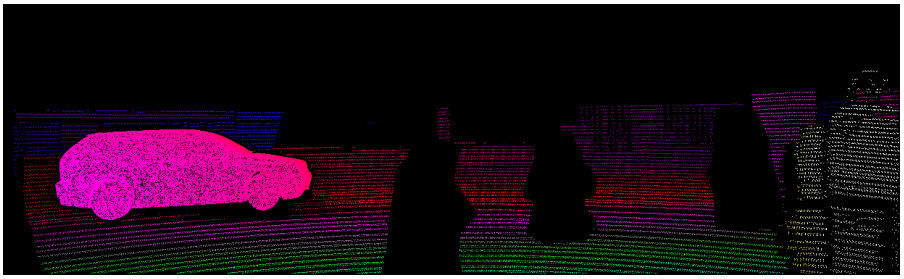
1.3. Verwandte Arbeiten

Die erste wegweisende Arbeit von Vedula *et al.* [30] präsentiert drei verschiedene Vorgehensweisen, die in Kombination mit dem optischen Fluss benutzt werden können, um den Szenenfluss zu berechnen. Die Wahl einer Vorgehensweise ist hierbei abhängig vom Kenntnisstand

(a) Referenzbild einer Szene.



(b) 2D-Projektion des Szenenflusses in Farbwertkodierung.



(c) Tiefenwerte der Szene in Farbwertkodierung.

Abbildung 1.1.: Szenenfluss- und Tiefenwerte aus dem KITTI Datensatz [21]. Mithilfe dieser Werte kann beispielsweise die Bewegung von Personen und Objekten im Straßenverkehr detektiert werden.

über die Struktur der Szenen und von eventuell bekannten Stereokorrespondenzen.
Die Arbeiten von Huguet und Devernay [16], Wedel *et al.* [31] sowie Valgaerts *et al.* [29] haben das Szenenflussproblem in einen Variationsansatz, basierend auf der optischen Fluss Methode von Brox *et al.* [3], eingebettet. Wedel *et al.* [31] separiert die Berechnungen für die Tiefen- und Bewegungswerte, um das Verfahren in Echtzeit durchführen zu können. Valgaerts *et al.* [29] unterscheidet sich in zwei wesentlichen Punkten von den anderen Arbeiten: (1) die Methode setzt keinen kalibrierten Stereoaufbau der Kameras voraus, d.h. die extrinsischen

Kameraparameter müssen nicht vorab bekannt sein. (2) Da Kanten im Bewegungsfluss nicht mit Kanten im Tiefenfeld übereinstimmen müssen, benutzen die Autoren getrennte Kostenfunktionen für die jeweiligen Glattheitsterme, um etwaige Unterbrechungen zwischen den einzelnen Flussfeldern zu erlauben. Alle drei Methoden [16, 29, 31] verwenden eine 2D-Parametrisierung für die Beschreibung des Szenenflusses.
Basha *et al.* [9] verfassten als Erste eine Arbeit, die eine 3D-Parametrisierung in Form einer Punktewolke verwendet hat. Die Punktewolke wird hierbei durch die Rückprojektion von Bildpunkten in die 3D-Szene gewonnen. Das Verfahren ist ebenfalls in einen Variationsansatz eingebettet. Die Arbeit von Menze und Geiger [21, 22] bedient sich hingegen kleiner ebener Flächen, sogenannten *Superpixeln*, um gezielt Objekte - und nicht einzelne Bildpunkte - in einer dynamischen Szene zu verfolgen. Diese Methode ist vor allem auf die Erfassung von Bewegungen im Straßenverkehr ausgelegt.
RGB-D Methoden, wie die von Herbst *et al.* [15] sowie Zanfir und Sminchisescu [35], bedienen sich neben den eigentlichen Eingabebildern noch zusätzlichen, meist von Sensoren gelieferten Tiefeninformationen. Hierdurch wird ein Stereoaufbau der Kameras überflüssig und dem Verfahren genügen Bilder einer einzelnen Kamera. Zudem erlaubt dieses Vorgehen die Formulierung von zusätzlichen Konstanzannahmen für die Tiefe. Ein Überblick und Vergleich über diverse Szenenflussverfahren ist in der Arbeit von Yan und Xiang [33] gegeben.
Abschließend sei die Arbeit von Maurer [20], die ein tiefen-getriebenes Verfahren zur Stereorekonstruktion auf Basis der 3D-Parametrisierung von Basha *et al.* vorstellt, erwähnt. Das hier vorgestellte Szenenflussverfahren baut somit auf der selben Grundlage wie die Arbeit von Maurer auf und kann als eine Erweiterung seines Stereoverfahrens angesehen werden.

1.4. Gliederung und Vorgehensweise

Der Aufbau der Arbeit ist wie folgt: Das Kapitel 2 soll eine Basis für die in der Arbeit verwendete Notation und die vorgestellte Methode schaffen. Das Kapitel 3 beschreibt dann das Szenenflussverfahren im Detail. Hiernach wird die Methode in Kapitel 4 ausführlich anhand von verschiedenen Datensätzen evaluiert. Schließlich wird die Arbeit in Kapitel 5 zusammengefasst und mögliche Verbesserungen des Verfahrens in Aussicht gestellt.

2. Grundlagen

Sinn und Zweck dieses Kapitels ist es, eine gemeinsame Grundlage für die Notation und die später in dieser Arbeit vorgestellte Methode und die darin verwendeten Konzepte zu schaffen. Zuerst werden die homogenen Koordinaten, die eine häufige Anwendung im anschließenden Abschnitt zum Lochkameramodell finden, vorgestellt. Das Lochkameramodell dient als Grundlage für die 3D Parametrisierung des in Kapitel 3 vorgestellten Szenenflussverfahrens. Den Abschluss dieses Kapitels bildet eine kleine Einführung in die Variationsrechnung, die benötigt wird, um das von dem Verfahren formulierte Energiefunktional minimieren zu können.

2.1. Homogene Koordinaten

Homogene Koordinaten haben u.a. die für diese Arbeit wertvolle Eigenschaft, nichtlineare affine und projektive Transformationen durch eine Matrixmultiplikation darstellen zu können [14]. Gegenüber kartesischen Koordinaten sind Gleichungen mit homogenen Koordinaten häufig einfacher und leichter zu verstehen. Homogene Koordinaten werden gebildet, indem eine 1 an die bereits bestehenden kartesischen Koordinaten angehängt wird. Die Transformation $\mathcal{H}$ von nichthomogenen Koordinaten $\mathbf{x} \in \mathbb{R}^n$ in homogene Koordinaten $\tilde{\mathbf{x}} \in \mathbb{R}^{n+1}$ kann also wie folgt ausgedrückt werden:

$$\begin{aligned} \mathcal{H} : \mathbb{R}^n &\to \mathbb{R}^{n+1} \\ \mathbf{x} = (x_1, \cdots, x_n)^\top &\to \tilde{\mathbf{x}} = (x_1, \cdots, x_n, 1)^\top \ . \end{aligned} \tag{2.1}$$

Die Rücktransformation $\mathcal{H}^{-1}$ erfolgt durch das Entfernen der zuvor angehängten 1. Im allgemeinen Fall wird jedoch durch etwaige Berechnungen u.a. die Koordinate an letzter Stelle verändert. In diesem Fall können die kartesischen Werte der homogenen Koordinaten durch

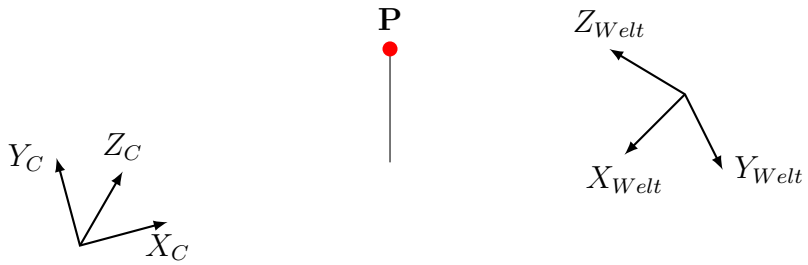


Abbildung 2.1.: Die Koordinaten des Punktes $\mathbf{P}$ im Kamerakoordinatensystem sind nicht identisch mit denen des Weltkoordinatensystems.

eine Division des Elementes an letzter Stelle wieder hergestellt werden. Die Rücktransformation ist also gegeben zu:

$$\begin{aligned} \mathcal{H}^{-1} &: \mathbb{R}^{n+1} \to \mathbb{R}^n \\ \tilde{\mathbf{x}} = (x_1, \cdots, x_n, x_{n+1})^\top &\to \mathbf{x} = (\frac{x_1}{x_{n+1}}, \cdots, \frac{x_n}{x_{n+1}})^\top . \end{aligned} \tag{2.2}$$

Hierdurch offenbaren die homogenen Koordinaten ihre projektive Eigenschaft, denn alle Punkte, die durch die Gerade $(\lambda x_1, \lambda x_2, \cdots, \lambda)^\top$ mit $\lambda \neq 0$ beschrieben sind, projizieren sich durch die Rücktransformation jeweils auf den selben Punkt in kartesischen Koordinaten. Die folgende Gleichung beschreibt diesen Zusammenhang:

$$\begin{aligned} &\mathcal{H}^{-1}\left((\lambda x_1, \lambda x_2, \cdots, \lambda x_n, \lambda)^\top\right) \\ &= \mathcal{H}^{-1}\left((x_1, x_2, \cdots, x_n, 1)^\top\right) \quad \text{, für alle } \lambda \neq 0 \\ &= (x_1, x_2, \cdots, x_n)^\top . \end{aligned} \tag{2.3}$$

Einen Spezialfall bilden homogene Koordinaten, die eine 0 an letzter Stelle enthalten. Da die Division durch 0 nicht definiert ist, können derartige Koordinaten nicht rücktransformiert werden. Stattdessen werden diese Koordinaten als unendlich weit entfernte Punkte, die nicht in kartesischen Koordinaten ausgedrückt werden können, verstanden [24].

2.2. Das Lochkameramodell

Im Bereich des Maschinensehens ist das Lochkameramodell eine gebräuchliche Annäherung für Kameras mit lichtempfindlichen Sensoren

[14]. Das Modell beschreibt die Projektion π eines Punktes $\mathbf{P} \in \mathbb{R}^3$ auf die Bildebene $\Omega \subset \mathbb{R}^2$ einer Kamera C. Das Ergebnis der Projektion ist der Punkt $\mathbf{p} \in \Omega$. Die Projektion π wird durch die *vollständige Projektionsmatrix* $\mathbf{M}$, die sich selbst wieder aus der *extrinsischen Matrix* $\mathbf{M}_{\text{Ext}}$, der *Kameraprojektionsmatrix* $\mathbf{M}_{\text{Kamera}}$ und der *intrinsischen Matrix* $\mathbf{M}_{\text{Int}}$ zusammensetzt, beschrieben. Im Folgenden werden diese Matrizen und deren Nutzen erläutert, bevor anschließend die vollständige Projektionsmatrix gebildet wird.

2.2.1. Extrinsische Matrix

Die 4×4 extrinsischen Matrix $\mathbf{M}_{\text{Ext}}$ enthält die extrinsischen Parameter einer Kamera und wird verwendet, um die Position des Punktes $\mathbf{P} = (X, Y, Z)^\top$ vom Weltkoordinatensystem in das entsprechende 3D Kamerakoordinatensystem, das bezüglich der Lochblende der Kamera definiert ist, zu transformieren. Abbildung 2.1 verdeutlicht den Unterschied zwischen den Koordinatensystemen. Die Koordinaten des Punktes $\mathbf{P}$ bezüglich des Kamerakoordinatensystems sind mit der folgenden Gleichung gegeben:

$$\tilde{\mathbf{P}}_{\text{Kamera}} = \mathbf{M}_{\text{Ext}} \cdot \tilde{\mathbf{P}} \,. \tag{2.4}$$

Die Elemente der extrinsische Matrix werden durch die Orientierung $\mathbf{R}$ und die Translation $\mathbf{t}$ relativ zum Ursprung des Weltkoordinatensystems gebildet.

$$\mathbf{M}_{\text{Ext}} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \,. \tag{2.5}$$

Die quadratische Matrix $\mathbf{R}$ wird auch als 3×3 Rotationsmatrix bezeichnet, während $\mathbf{t}$ ein 3×1 Translationsvektor ist [10]. Durch die Verwendung der zuvor eingeführten homogenen Koordinaten lassen sich die Rotation und Translation kompakt durch die in Gleichung 2.5 beschriebene Matrix darstellen. Mit drei Wahlmöglichkeiten für die Position $(t_1, t_2, t_3)^\top$ der Kamera und drei Wahlmöglichkeiten für die Rotation (ϕ_X, ϕ_Y, ϕ_Z) [10] um die jeweiligen Achsen verfügt die extrinsische Matrix über 6 verschiedene Freiheitsgrade. Zudem hat die extrinsische Matrix den vollen Rang und ist somit invertierbar.

Die invertierte Matrix $\mathbf{M}_{\text{Ext}}^{-1}$ beschreibt die Rücktransformation von Kamerakoordinaten zu Weltkoordinaten mittels

$$\tilde{\mathbf{P}} = \mathbf{M}_{\text{Ext}}^{-1} \cdot \tilde{\mathbf{P}}_{\text{Kamera}} \,. \tag{2.6}$$

Die Elemente der inversen Matrix $\mathbf{M}_{\text{Ext}}^{-1}$ können durch die Gleichung

$$\mathbf{M}_{\text{Ext}}^{-1} = \begin{pmatrix} \mathbf{R}^{-1} & -\mathbf{R}^{-1}\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R}^{\top} & -\mathbf{R}^{\top}\mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{2.7}$$

bestimmt werden [10]. Dies ist für gewöhnlich weniger aufwändig als das Invertieren einer allgemeinen 4×4 Matrix. Für den weiteren Verlauf dieser Sektion wird, ohne Beschränkung der Allgemeinheit, angenommen, dass das Weltkoordinatensystem mit dem Kamerakoordinatensystem übereinstimmt und keine Transformation zwischen Welt- und Kamerakoordinaten vorgenommen werden muss. In diesem Fall kann die extrinsische Matrix durch die Einheitsmatrix $\mathbf{I}$ repräsentiert werden und es gilt:

$$\tilde{\mathbf{P}}_{\text{Kamera}} = \mathbf{I} \cdot \tilde{\mathbf{P}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \tag{2.8}$$

2.2.2. Kameraprojektionsmatrix

Die 3×4 Kameraprojektionsmatrix beschreibt die Projektion von 3D-Kamerakoordinaten $\mathbf{P}_{\text{Kamera}} \in \mathbb{R}^3$ zu 2D-Kamerakoordinaten $\mathbf{p}_{\text{Bild}} \in \Omega$, wobei $\Omega \subset \mathbb{R}^2$ die Bildebene der Lochkamera bezeichnet. Die 2D-Kamerakoordinaten haben ihren Ursprung in der Bildmitte c. Die Bildmitte ist als die Stelle, an der sich die Z-Achse des Kamerakoordinatensystems mit der Bildebene Ω schneidet, definiert. Aufgrund dieser Eigenschaft wird die Z-Achse häufig auch als Hauptachse bezeichnet. Der Abstand zwischen Bildebene und Kamerazentrum C wird als Brennweite bezeichnet. Um eine Punktspiegelung des aufgenommenen Bildes zu verhindern, befindet sich die Bildebene im Gegensatz zu realen Kameras vor dem Kamerazentrum. Abbildung 2.2 zeigt den Aufbau eines solchen Szenarios. Bei der Bestimmung der Kameraprojektionsmatrix hilft eine seitliche Betrachtung der Szene entlang der Kamera X-Achse. Dies ist in Abbildung 2.3 dargestellt. Aus

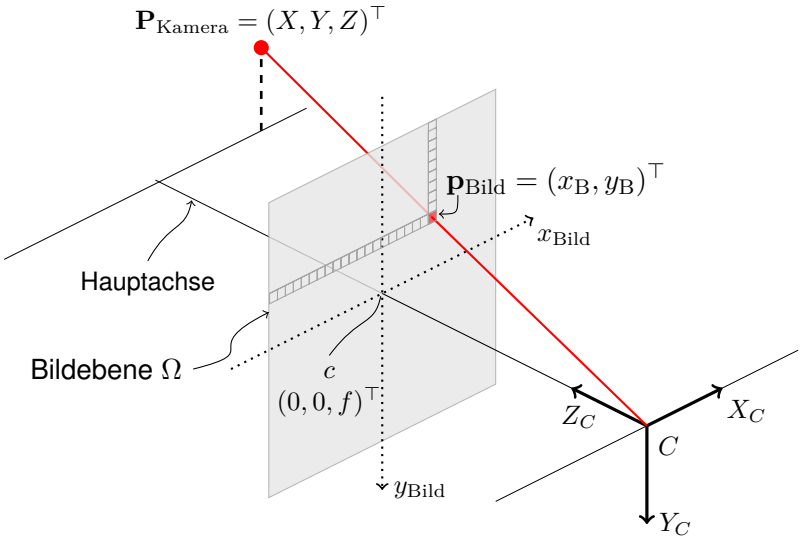


Abbildung 2.2.: Der 3D Punkt $\mathbf{P}_{\text{Kamera}}$ wird auf den Bildpunkt $\mathbf{p}_{\text{Bild}}$ der Bildebene Ω projiziert.

dieser Abbildung wird mithilfe des Strahlensatzes folgende Beziehung zwischen den Koordinaten klar:

$$\frac{y_{\text{B}}}{Y} = \frac{f}{Z} \,. \tag{2.9}$$

Eine weitere Ansicht von oben, also entlang der Kamera Y-Achse, hilft, analog zum vorherigen Verfahren, die Beziehung bezüglich der X- und Z-Koordinaten der jeweiligen Punkte zu ermitteln.

$$\frac{x_{\text{B}}}{X} = \frac{f}{Z} \,. \tag{2.10}$$

Aus den Gleichungen 2.9 und 2.10 und der Umstellung nach x und y lassen sich die Koordinaten des Punktes $\mathbf{p}_{\text{Bild}}$ in Bezug auf den Bildmittelpunkt wie folgt bestimmen:

$$\mathbf{p}_{\text{Bild}} = \begin{pmatrix} x_{\text{B}} \\ y_{\text{B}} \end{pmatrix} = \begin{pmatrix} f \cdot X/Z \\ f \cdot Y/Z \end{pmatrix} \,. \tag{2.11}$$

Mithilfe der homogenen Koordinaten kann der Punkt $\mathbf{p}$ etwas eleganter formuliert werden:

$$\tilde{\mathbf{p}}_{\text{Bild}} = \begin{pmatrix} f \cdot X \\ f \cdot Y \\ Z \end{pmatrix} \,. \tag{2.12}$$

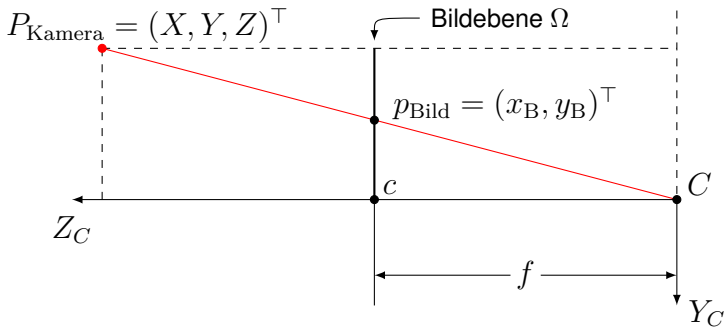


Abbildung 2.3.: Seitliche Ansicht der Geometrie des Lochkameramodells entlang der X-Achse der Kamera.

Die Projektion $\mathbf{P}_{\text{Kamera}} \rightarrow \mathbf{p}_{\text{Bild}}$ lässt sich nun als Multiplikation mit der Kamera Projektionsmatrix $\mathbf{M}_{\text{Kamera}}$ darstellen:

$$\tilde{\mathbf{p}}_{\text{Bild}} = \mathbf{M}_{\text{Kamera}} \cdot \tilde{\mathbf{P}}_{\text{Kamera}} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \tag{2.13}$$

Da es sich hier um eine 3×4 Matrix handelt, kann leider die Rücktransformation mangels einer inversen Matrix nicht definiert werden. Dies lässt sich mit dem Informationsverlust der Tiefe bei der Projektion auf die Bildebene erklären. Es ist jedoch möglich, die Position des abgebildeten Punktes $\mathbf{P}_{\text{Kamera}}$ in Abhängigkeit einer zuvor bekannten Tiefe Z zu formulieren. Dieser Ansatz wird später in der Arbeit noch einmal aufgegriffen.

2.2.3. Intrinsische Matrix

Bisher wurde beschrieben wie die extrinsische Matrix und die Kameraprojektionsmatrix die Projektion eines Punktes mit beliebigem Koordinatenursprung auf einen Punkt in der Bildebene einer Kamera abbilden. Leider ist eine letzte Koordinatentransformation in Form der intrinsischen 3×3 Matrix $\mathbf{M}_{\text{Int}}$, die die intrinsischen Parameter einer Kamera enthält, nötig. Bei der Verarbeitung von Bildern mit einem Computer werden für gewöhnlich Bildpunkte mit zwei positiven ganzzahligen Indizes addressiert. Der Ursprung, also der Bildpunkt $\mathbf{p} = (0, 0)^\top$, befindet sich in der oberen linken Bildecke. Ausgehend von dem Bildpunktursprung sind die Koordinaten der Bildmitte mit

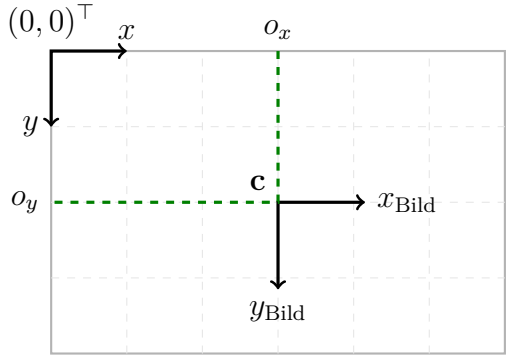


Abbildung 2.4.: Zusammenhang des Bild- und Bildpunktkoordinatensystems.

$(o_x, o_y)^\top$ modelliert. Abbildung 2.4 zeigt den Zusammenhang der beiden Koordinatensysteme. Die Koordinaten des Punktes $\mathbf{p}_{\text{Bild}}$ sind in einer zuvor bestimmten Längeneinheit berechnet worden. Um diese Koordinaten in Indizes für das Bildpunktkoordinatensystem umzuwandeln, werden die Skalierungsfaktoren n_x und n_y mit der Einheit [Bildpunkte / Längeneinheit] verwendet. Um das Modell einfach zu halten, wird in dieser Arbeit von einem orthogonalen Bildpunktkoordinatensystem ausgegangen. Die Parameter o_x, o_y sowie n_x und n_y bilden die 4 Freiheitsgrade der intrinsischen Matrix

$$\mathbf{M}_{\text{Int}} = \begin{pmatrix} n_x & 0 & o_x \\ 0 & n_y & o_y \\ 0 & 0 & 1 \end{pmatrix} . \tag{2.14}$$

Die Koordinatentransformation $\mathbf{p}_{\text{Bild}} \rightarrow \mathbf{p}$ kann nun mithilfe der intrinsischen Matrix mit folgender Gleichung beschrieben werden:

$$\begin{aligned} \tilde{\mathbf{p}} &= \mathbf{M}_{\text{Int}} \cdot \tilde{\mathbf{p}}_{\text{Bild}} \\ &= \begin{pmatrix} n_x & 0 & o_x \\ 0 & n_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f \cdot X \\ f \cdot Y \\ Z \end{pmatrix} . \end{aligned} \tag{2.15}$$

Die intrinsische Matrix hat, wie zuvor die extrinsische Matrix, vollen Rang und die Rücktransformation $\mathbf{p} \rightarrow \mathbf{p}_{\text{Bild}}$ ist damit über die invertierte intrinsische Matrix $\mathbf{M}_{\text{Int}}^{-1}$ definiert.

2.2.4. Vollständige Projektionsmatrix

Nachdem alle nötigen Matrizen definiert und beschrieben worden sind, kann nun die vollständige 3×4 Projektionsmatrix $\mathbf{M}$, die die Projektion $\pi : \mathbb{R}^3 \to \Omega$ beschreibt, gebildet werden. Hierfür werden die zuvor besprochenen Matrizen miteinander multipliziert:

$$\begin{aligned}\mathbf{M} &= \mathbf{M}_{\text{Int}} \cdot \mathbf{M}_{\text{Kamera}} \cdot \mathbf{M}_{\text{Ext}} \\ &= \begin{pmatrix} n_x & 0 & o_x \\ 0 & n_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \end{pmatrix} .\end{aligned} \tag{2.16}$$

Die Projektion eines Punktes $\mathbf{P} = (X, Y, Z)^\top$ auf den Bildpunkt $\mathbf{p} = (x, y)^\top$ lässt sich nun kompakt mit der vollständigen Projektionsmatrix und homogenen Koordinaten repräsentieren:

$$\tilde{\mathbf{p}} = \mathbf{M} \cdot \tilde{\mathbf{P}} = \begin{pmatrix} M_{11} \cdot X + M_{12} \cdot Y + M_{13} \cdot Z + M_{14} \\ M_{21} \cdot X + M_{22} \cdot Y + M_{23} \cdot Z + M_{24} \\ M_{31} \cdot X + M_{32} \cdot Y + M_{33} \cdot Z + M_{34} \end{pmatrix} . \tag{2.17}$$

Aufgrund der nicht invertierbaren Kameraprojektionsmatrix $\mathbf{M}_{\text{Kamera}}$ ist auch die vollständige Projektionsmatrix $\mathbf{M}$ nicht invertierbar. Die vollständige Projektionsmatrix verfügt insgesamt über 10 Freiheitsgrade, die sich aus den 6 Freiheitsgraden der extrinsischen Matrix und den 4 Freiheitsgraden der intrinsischen Matrix zusammensetzen. Möchte man schiefwinklige Bildpunktkoordinatensysteme modellieren, kommt ein weiterer Freiheitsgrad für die intrinsische Matrix hinzu. Sind alle Freiheitsgrade, und somit die vollständige Projektionsmatrix, bekannt, so spricht man von einem vollständig kalibrierten Kameraaufbau. Eine Übersicht der Koordinatensysteme und der verwendeten Matrizen im Lochkameramodell ist in Abbildung 2.5 zu finden.

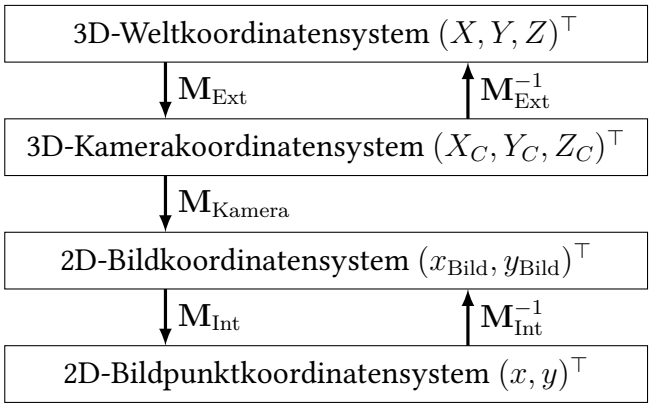


Abbildung 2.5.: Koordinatensysteme im Lochkameramodell

2.2.5. Kamerakalibrierungsmatrix

Abschließend wird die 3×3 Kamerakalibrierungsmatrix $\mathbf{K}$, die aus der intrinsischen Matrix und der Kameraprojektionsmatrix gebildet werden kann, vorgestellt. Die Kamerakalibrierungsmatrix beinhaltet alle kameraspezifischen Parameter und wird häufig zusammen mit der extrinsischen Matrix für Testdatensätze bereitgestellt. Die Matrix $\mathbf{K}$ kann wie folgt gebildet werden:

$$\begin{aligned}\begin{pmatrix}\mathbf{K} & \mathbf{0}\end{pmatrix} &= \mathbf{M}_{\text{Int}} \cdot \mathbf{M}_{\text{Kamera}} \\ &= \begin{pmatrix} n_x \cdot f & 0 & o_x & 0 \\ 0 & n_y \cdot f & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\ \mathbf{K} &= \begin{pmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{pmatrix}\end{aligned} \tag{2.18}$$

mit $s_x = n_x \cdot f$ und $s_y = n_y \cdot f$. Der Ausdruck $\begin{pmatrix}\mathbf{K} & \mathbf{0}\end{pmatrix}$ beschreibt somit die direkte Koordinatentransformation $\mathbf{P}_{\text{Kamera}} \rightarrow \mathbf{p}$. Die Darstellung der Gleichung 2.16 verändert sich unter Verwendung der Kamerakalibrierungsmatrix wie folgt:

$$\begin{aligned}\mathbf{M} &= \begin{pmatrix}\mathbf{K} & \mathbf{0}\end{pmatrix} \cdot \mathbf{M}_{\text{Ext}} \\ &= \begin{pmatrix} s_x & 0 & o_x & 0 \\ 0 & s_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}.\end{aligned} \tag{2.19}$$

2.3. Variationsrechnung

Die Variationsrechnung befasst sich mit der Berechnung von Extrema und ist somit geeignet für das Lösen von Optimierungsproblemen [7]. Wie bereits in der Einleitung dieser Arbeit erwähnt, wird die Berechnung des Szenenflusses in einem globalen, kontinuierlichen Optimierungsproblem, einem Variationsansatz, der ein geeignetes Energiefunktional minimiert, eingebettet. Ein Energiefunktional $E(u(x))$ ist, pragmatisch ausgedrückt, eine Funktion, die selbst wiederum aus Funktionen und deren Ableitungen besteht und kann durch die Gleichung

$$E(u(x)) = \int_a^b F(x, u(x), u_x(x))\, dx \tag{2.20}$$

ausgedrückt werden. Da in dem Energiefunktional nur die 1. Ableitung von $u(x)$ enthalten ist, wird auch von einem Funktional 1. Ordnung gesprochen [11]. $C^n_{[a,b]}$ bezeichne nun die Menge aller Funktionen mit einer kontinuierlichen stetigen Ableitung n-ter Ordnung im Interval $[a, b] \in \mathbb{R}$. Um das Energiefunktional zu minimieren, muss eine geeignete Funktion $u(x) \in C^2_{[a,b]}$ die sogenannte *Euler-Lagrange-Gleichung* erfüllen. Für das obige Energiefunktional ergibt sich die Euler-Lagrange-Gleichung zu

$$0 = F_u(x, u(x), u_x(x)) - \frac{\partial}{\partial x} F_{u_x}(x, u(x), u_x(x)) \tag{2.21}$$

mit den Randbedingungen

$$\begin{aligned} 0 &= F_{u_x}(a, u(a), u_x(a)) \,, \\ 0 &= F_{u_x}(b, u(b), u_x(b)) \,. \end{aligned} \tag{2.22}$$

Ein Beweis für dieses Theorem ist in [17] beschrieben. Löst man also die Euler-Lagrange-Gleichung, erhält man die Funktion $u(x)$, die das Energiefunktional $E(u(x))$ minimiert. Das Energiefunktional in Gleichung 2.20 berücksichtigt jedoch nur die reelle Variable x. Da die im nächsten Kapitel vorgestellte Szenenflussmethode mit 2D-Bildern arbeitet, ist eine Erweiterung des Variationsansatzes notwendig. Um die Gleichungen weiterhin kompakt zu halten, werden die Funktionsargumente, sofern diese eindeutig sind, im weiteren Verlauf der Arbeit impliziert. Gegeben sei nun das Energiefunktional

$$E(u(x, y)) = \int_\Omega F(x, y, u, u_x, u_y)\, dxdy \,, \tag{2.23}$$

welches über den Bildbereich Ω in den Richtungen x, y integriert. Laut [8] ergibt sich dann die Euler-Lagrange-Gleichung zu

$$0 = F_u - \frac{\partial}{\partial x} F_{u_x} - \frac{\partial}{\partial y} F_{u_y} \tag{2.24}$$

mit den natürlichen Randbedingungen

$$\mathbf{0} = \mathbf{n}^\top \begin{pmatrix} F_{u_x} \\ F_{u_y} \end{pmatrix} \tag{2.25}$$

auf $\partial\Omega$, wobei $\mathbf{n}$ den normierten Normalenvektor bezeichnet. Eine Herleitung dieser Randbedingungen ist im Anhang von [20] zu finden.

Auch dieser Variationsansatz ist noch immer nicht ausreichend, um das später vorgestellte Szenenflussproblem zu modellieren, so dass eine letzte Erweiterung notwendig wird. Da das Szenenflussverfahren mehrere Größen gleichzeitig berechnet, muss hierfür die Zahl der verwendeten Funktionen, die das Energiefunktional aufbauen, ebenfalls erhöht werden. Ein Energiefunktional mit dem Funktionsvektor $\mathbf{u}$, bestehend aus n gesuchten Funktionen $(u_1, u_2, \cdots, u_n)$, kann durch die Gleichung

$$E(\mathbf{u}(x,y)) = \int_\Omega F(x, y, \mathbf{u}, \mathcal{J}_\mathbf{u})\, dxdy\,, \tag{2.26}$$

in der $\mathcal{J}$ die Jacobi-Matrix ist, beschrieben werden. Die entsprechende Euler-Lagrange-Gleichung, welche die Minimierer des Funktionals erfüllen müssen, wird hierdurch zu einem Gleichungssystem aus n Gleichungen [7]:

$$\begin{aligned}
0 &= F_{u_1} - \frac{\partial}{\partial x} F_{u_{1_x}} - \frac{\partial}{\partial y} F_{u_{1_y}}\,, \\
0 &= F_{u_2} - \frac{\partial}{\partial x} F_{u_{2_x}} - \frac{\partial}{\partial y} F_{u_{2_y}}\,, \\
&\cdots \qquad\quad \cdots \qquad\quad \cdots \\
0 &= F_{u_n} - \frac{\partial}{\partial x} F_{u_{n_x}} - \frac{\partial}{\partial y} F_{u_{n_y}}
\end{aligned} \tag{2.27}$$

mit den natürlichen Randbedingungen

$$0 = \mathbf{n}^\top \begin{pmatrix} F_{u_{1_x}} \\ F_{u_{1_y}} \end{pmatrix} = \mathbf{n}^\top \begin{pmatrix} F_{u_{2_x}} \\ F_{u_{2_y}} \end{pmatrix} = \cdots = \mathbf{n}^\top \begin{pmatrix} F_{u_{n_x}} \\ F_{u_{n_y}} \end{pmatrix} \quad . \tag{2.28}$$

Zuletzt sei angemerkt, dass im Falle von streng konvexen Energiefunktionalen die berechneten Funktionen, die das Funktional minimieren, genau eine Lösung besitzen.

3. Das Szenenflussverfahren

Dieses Kapitel beschreibt die Funktionsweise des auf der Arbeit von Basha *et al.* [9] basierenden Szenenflussverfahrens. Dies geschieht in mehreren Schritten. Zuerst wird die verwendete 3D-Parametrisierung mithilfe des Lochkamera Modells hergeleitet. Anschließend wird das Energiefunktional des Verfahrens formuliert und die für die Minimierung notwendigen Euler-Lagrange-Gleichungen abgeleitet. Die Minimierung selbst bedarf aufgrund des nichtlinearen und nicht konvexen Energiefunktionals einer aufwändigeren Verfahrensweise, deren Erklärung sich dann anschließt. Das Ende dieses Kapitels bildet die Erweiterung des Verfahrens durch die Gradientenkonstanzannahme.

3.1. Parametrisierung

Das Szenenflussverfahren verwendet eine 3D-Punktewolke $\mathbb{P} \subset \mathbb{R}^3$ als Parametrisierung. Die Koordinaten eines in der Punktewolke enthaltenen Punktes $\mathbf{P} \in \mathbb{P}$ sind bezüglich einer frei wählbaren Referenzkamera C_0 zu verstehen. Weiter wird vorausgesetzt, dass das Koordinatensystem der Kamera C_0 mit dem Weltkoordinatensystem übereinstimmt und dass $\mathbf{p}_i \in \Omega_i$ der auf die Bildebene $\Omega_i \subset \mathbb{R}^2$ mittels der vollständigen Projektionsmatrix $\mathbf{M}_i$ der Kamera C_i projizierte Punkt $\mathbf{P}$ ist. Die Parametrisierung für ein System bestehend aus zwei Kameras ist in Abbildung 3.1 dargestellt. Da dem Verfahren als Eingabe lediglich die Bilder und damit die 2D-Koordinaten der Bildpunkte zur Verfügung stehen, sind die 3D-Koordinaten eines Punktes $\mathbf{P}$ in der Punktewolke $\mathbb{P}$ zunächst unbekannt. Im Folgenden wird aus diesem Grund die Rückprojektion eines Punktes $\mathbf{p}_0$ in der Bildebene der Referenzkamera in die 3D-Punktewolke beschrieben.

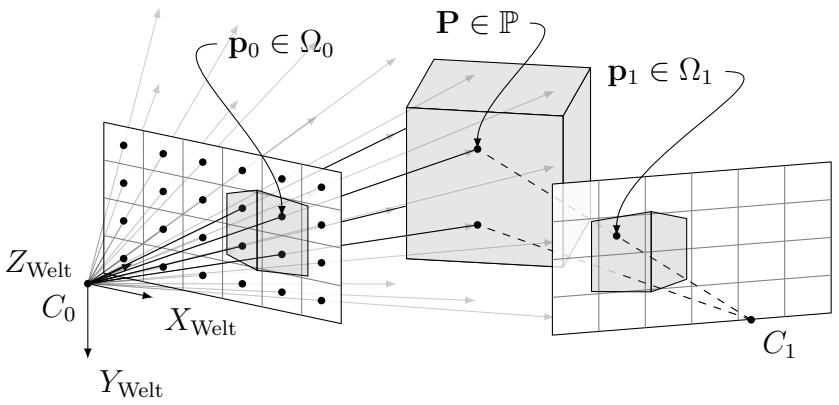


Abbildung 3.1.: Parametrisierung des Szenenflussverfahrens. 3D-Punkte liegen auf einem optischen Strahl durch die Bildebene und das Kamerazentrum C_0, dessen Koordinatensystem mit dem Weltkoordinatensystem übereinstimmt. Abbildung aus [20].

Auf Basis des Lochkameramodells aus dem Grundlagenteil dieser Arbeit sind die Koordinaten des Punktes $\tilde{\mathbf{p}}_0$ wie folgt bekannt:

$$\tilde{\mathbf{p}}_0 = \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \mathbf{M}_0 \cdot \tilde{\mathbf{P}} = \begin{pmatrix} \mathbf{K}_0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{R}_0 & \mathbf{t}_0 \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \tag{3.1}$$

Unter der Voraussetzung, dass das Kamerakoordinatensystem mit dem Weltkoordinatensystem übereinstimmt, kann die extrinsische Matrix durch die Einheitsmatrix ersetzt werden und es ergibt sich

$$\begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{K}_0 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} s_{x_0} & 0 & o_{x_0} & 0 \\ 0 & s_{y_0} & o_{y_0} & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} . \tag{3.2}$$

Durch Ausmultiplizieren und anschließendes Rücktransformieren der homogenen Koordinaten erhält man:

$$\begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} = \begin{pmatrix} X \cdot s_{x_0} + Z \cdot o_{x_0} \\ Y \cdot s_{y_0} + Z \cdot o_{y_0} \\ Z \end{pmatrix} ,$$

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 1/Z \cdot (X \cdot s_{x_0} + Z \cdot o_{x_0}) \\ 1/Z \cdot (Y \cdot s_{y_0} + Z \cdot o_{y_0}) \end{pmatrix} . \tag{3.3}$$

Da die Koordinaten $(X, Y, Z)^\top$ des 3D-Punktes gesucht werden, muss die Gleichung entsprechend der Variablen umgestellt werden. Es existieren jedoch nur zwei Gleichungen für drei Variablen, so dass das Gleichungssystem unterbestimmt ist. Dies bedeutet, dass die Werte zweier Variablen als Funktion der anderen Variablen ausgedrückt werden können. Wie schon bereits im Abschnitt zum Lochkameramodell erwähnt, ist dies durch den Verlust der Tiefeninformation (Ω_i hat keine Z-Achse) bei der Projektion auf die Bildebene zu erklären. Auflösen nach $(X, Y)^\top$ liefert:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = Z \begin{pmatrix} (x_0 - o_{x_0})/s_{x_0} \\ (y_0 - o_{y_0})/s_{y_0} \end{pmatrix} . \tag{3.4}$$

Die erhaltene Gleichung beschreibt den optischen Strahl durch das Kamerazentrum C_0 und die Bildebene Ω_0. Bei bekannter Tiefe eines Bildpunktes kann somit die 3D-Position des Punktes $P \in \mathbb{P}$ mit der Gleichung

$$\mathbf{P} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = Z \begin{pmatrix} (x_0 - o_{x_0})/s_{x_0} \\ (y_0 - o_{y_0})/s_{y_0} \\ 1 \end{pmatrix} \tag{3.5}$$

rekonstruiert werden. Mithilfe dieser Gleichung können die Koordinaten eines Punktes $\mathbf{p}_i \in \Omega_i$ auf der Bildebene einer Kamera C_i nun wie folgt bestimmt werden. Im Grundlagenteil dieser Arbeit ist die Projektion $\mathbf{P} \to \mathbf{p}_i$ durch die Gleichung

$$\tilde{\mathbf{p}}_\mathbf{i} = \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \end{pmatrix} = \mathbf{M}_i \cdot \tilde{\mathbf{P}} \tag{3.6}$$

beschrieben. Eine Rücktransformation zu euklidischen Koordinaten liefert

$$\mathbf{p}_\mathbf{i} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} \tilde{x}_i/\tilde{z}_i \\ \tilde{y}_i/\tilde{z}_i \end{pmatrix} = \frac{1}{[\mathbf{M}_i]_3 \cdot \tilde{\mathbf{P}}} \begin{pmatrix} [\mathbf{M}_i]_1 \cdot \tilde{\mathbf{P}} \\ [\mathbf{M}_i]_2 \cdot \tilde{\mathbf{P}} \end{pmatrix} , \tag{3.7}$$

wobei $[\mathbf{M}_i]_j$ die j-te Zeile der Matrix $\mathbf{M}_i$ bezeichnet. Hieraus ergibt sich für die Koordinaten x_i und y_i:

$$\begin{aligned} x_i &= \frac{[M_i]_1}{[M_i]_3} = \frac{M_{i,11} \cdot X + M_{i,12} \cdot Y + M_{i,13} \cdot Z + M_{i,14}}{M_{i,31} \cdot X + M_{i,32} \cdot Y + M_{i,33} \cdot Z + M_{i,34}} \,, \\ y_i &= \frac{[M_i]_2}{[M_i]_3} = \frac{M_{i,21} \cdot X + M_{i,22} \cdot Y + M_{i,23} \cdot Z + M_{i,24}}{M_{i,31} \cdot X + M_{i,32} \cdot Y + M_{i,33} \cdot Z + M_{i,34}} \,. \end{aligned} \tag{3.8}$$

Mithilfe der Gleichung 3.5 können die Variablen X, Y substituiert werden. Anschließendes Ausklammern von Z liefert dann

$$\begin{aligned} x_i &= \frac{M_{i,11} \cdot Z \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,12} \cdot Z \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + Z \cdot M_{i,13} + M_{i,14}}{M_{i,31} \cdot Z \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,32} \cdot Z \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + Z \cdot M_{i,33} + M_{i,34}} \\ &= \frac{Z \cdot \overbrace{\left(M_{i,11} \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,12} \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + M_{i,13} \right)}^{a} + \overbrace{M_{i,14}}^{b}}{Z \cdot \underbrace{\left(M_{i,31} \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,32} \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + M_{i,33} \right)}_{c} + \underbrace{M_{i,34}}_{d}} \\ &= \frac{a \cdot Z + b}{c \cdot Z + d} \,, \end{aligned}$$

$$\begin{aligned} y_i &= \frac{M_{i,21} \cdot Z \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,22} \cdot Z \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + Z \cdot M_{i,23} + M_{i,24}}{M_{i,31} \cdot Z \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,32} \cdot Z \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + Z \cdot M_{i,33} + M_{i,34}} \\ &= \frac{Z \cdot \overbrace{\left(M_{i,21} \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,22} \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + M_{i,23} \right)}^{\check{a}} + \overbrace{M_{i,24}}^{\check{b}}}{Z \cdot \underbrace{\left(M_{i,31} \cdot \frac{x_0 - o_{x_0}}{s_{x_0}} + M_{i,32} \cdot \frac{y_0 - o_{y_0}}{s_{y_0}} + M_{i,33} \right)}_{c} + \underbrace{M_{i,34}}_{d}} \\ &= \frac{\check{a} \cdot Z + \check{b}}{c \cdot Z + d} \,. \end{aligned} \tag{3.9}$$

Mithilfe dieser Gleichung können also die Koordinaten eines Punktes $\mathbf{p}_i = (x_i, y_i)^\top$ in der Bildebene Ω_i durch die Koordinaten des Punktes $\mathbf{p}_0 = (x_0, y_0)^\top$ in der Bildebene Ω_0 der Referenzkamera zum Zeitpunkt

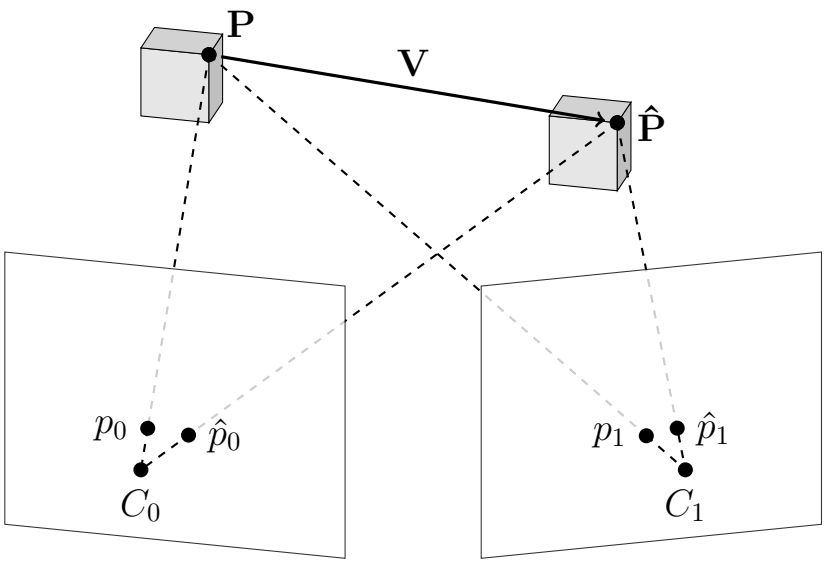


Abbildung 3.2.: Der 3D-Punkt $\mathbf{P}$ wird zum Zeitpunkt t auf die Punkte $\mathbf{p}_0$ und $\mathbf{p}_1$ der Kameras C_0 und C_1 projiziert. Nach der Verschiebung $\mathbf{V}$ zum Zeitpunkt $t+1$ wird die neue Position $\hat{\mathbf{P}}$ auf die Punkte $\hat{\mathbf{p}}_0$ und $\hat{\mathbf{p}}_1$ projiziert.

t bestimmt werden. Diese Parametrisierung ist ausreichend, um die Tiefe einer statischen Szene zu berechnen (siehe [20]). Zu dem Zeitpunkt $t+1$ hat sich ein 3D-Punkte $\mathbf{P} \in \mathbb{P}$ jedoch um den Szenenfluss, also um den Bewegungsvektor $\mathbf{V} = (u, v, w)^\top$, in der Szene verschoben. Die Koordinaten des so neu entstandenen Punktes $\hat{\mathbf{P}}$ sind

$$\hat{\mathbf{P}} = \mathbf{P} + \mathbf{V} = \begin{pmatrix} X+u \\ Y+v \\ Z+w \end{pmatrix} \quad . \tag{3.10}$$

Die projizierten Koordinaten $\hat{\mathbf{p}}_i = (\hat{x}_i, \hat{y}_i)^\top$ des bewegten Punktes $\hat{\mathbf{P}}$ können analog zum Fall des Zeitpunktes t berechnet werden. Um die Notation kompakt zu halten, wird die Gleichung 3.9 erweitert. Die Koordinaten $(\hat{x}_i, \hat{y}_i)^\top$ können dann mit

$$\begin{aligned} \hat{x}_i &= \frac{a \cdot Z + M_{i,11} \cdot u + M_{i,12} \cdot v + M_{i,13} \cdot w + b}{c \cdot Z + M_{i,31} \cdot u + M_{i,32} \cdot v + M_{i,33} \cdot w + d} \,, \\ \hat{y}_i &= \frac{a \cdot Z + M_{i,21} \cdot u + M_{i,22} \cdot v + M_{i,23} \cdot w + b}{c \cdot Z + M_{i,31} \cdot u + M_{i,32} \cdot v + M_{i,33} \cdot w + d} \,, \end{aligned} \tag{3.11}$$

berechnet werden. Die Abbildung 3.2 verdeutlicht den Zusammenhang der Punkte $\mathbf{P}$ und $\hat{\mathbf{P}}$ und deren Projektion auf die Bildebenen zweier Kameras.

Insgesamt verwendet also die hier vorgestellte Parametrisierung vier Unbekannte pro Bildpunkt. Die Tiefe Z wird verwendet, um die Rückprojektion des Bildpunktes in die 3D-Punktewolke zu ermöglichen. Dies wiederum wird benötigt, um Korrespondenzen zwischen den einzelnen Bildern zum Zeitpunkt t zu etablieren. Der Vektor $\mathbf{V} = (u, v, w)^\top$ beschreibt den Szenenfluss und wird zusammen mit Z benötigt, um Korrespondenzen zwischen Bildern der Zeitpunkte t und $t + 1$ zu finden.

3.2. Modellierung des Variationsansatzes

Unter Verwendung der zuvor eingeführten Parametrisierung kann nun der Variationsansatz des Szenenflussverfahrens formuliert werden. Zentraler Bestandteil des Verfahrens ist ein Energiefunktional, durch dessen Minimierung die Tiefe $Z(x, y)$ und der Szenenfluss $\mathbf{V}(x, y)$ bestimmt werden können. Die allgemeine Form dieses Energiefunktionals besteht aus einem Datenterm E_{Data} und einem Glattheitsterm E_{Smooth}, wie die folgende Gleichung zeigt:

$$E(Z, \mathbf{V}) = E_{\text{Data}}(Z, \mathbf{V}) + E_{\text{Smooth}}(Z, \mathbf{V}) \, . \tag{3.12}$$

Das Zusammenspiel der beiden Terme E_{Data} und E_{Smooth} ist dabei wie folgt: Während der Datenterm geeignete Modellannahmen beinhaltet, welche die Tiefe und den Szenenfluss mittels zusammengehöriger Bildpunkte zwischen den einzelnen Bildern ermittelt, dient der Glattheitsterm als Regularisierer an den Stellen im Bild, an denen eine eindeutige Lösung mittels des Datenterms aufgrund von Verdeckungen, Ausreißern oder dem Blendenproblem [1] nicht gefunden werden kann. Im Folgenden werden die beiden Terme nacheinander genauer betrachtet.

Die Modellannahme des Datenterms ist die Helligkeitswertkonstanzannahme (engl.: **b**rightness **c**onstancy assumption) , die von Lambert'schen Oberflächen in der aufgenommenen Szene ausgeht. Mit der zuvor eingeführten Parametrisierung bedeutet dies, dass die Intensität eines projizierten Punktes der Punktewolke, vor und nach der Bewe-

gung $\mathbf{V}$, konstant in den jeweiligen Bildern der Kameras ist. Hierfür wird der Datenterm wiederum in drei Funktionen unterteilt:

$$E_{Data}(Z, \mathbf{V}) = \int_{\Omega_0} BC_m(Z, \mathbf{V}) + BC_{s1}(Z) + BC_{s2}(Z, \mathbf{V})\, dxdy\ . \tag{3.13}$$

Die Funktion $BC_m(Z, \mathbf{V})$ bestraft hierbei Abweichungen der Intensität zwischen den einzelnen Bildern $I_i(x, y, t)$ und $I_i(x, y, t+1)$ einer Kamera C_i vor und nach der 3D-Bewegung $\mathbf{V}$. Gleichermaßen bestraft die Funktion $BC_{s1}(Z)$ Abweichungen der Intensität zwischen dem Referenzbild $I_0(x, y, t)$ und den anderen Bildern des Zeitpunktes t. Die selbe Funktionalität übernimmt der Term $BC_{s2}(Z, \mathbf{V})$ für die Bilder des Zeitpunktes $t+1$. Für ein System bestehend aus N Kameras geben die folgende Gleichungen Aufschluss über den genauen Aufbau dieser drei Funktionen:

$$\begin{aligned} BC_m(Z, \mathbf{V}) &= \sum_{i=0}^{N-1} \Psi(|I_i(\mathbf{p}_i, t) - I_i(\hat{\mathbf{p}}_i, t+1)|^2)\ , \\ BC_{s1}(Z) &= \sum_{i=1}^{N-1} \Psi(|I_0(\mathbf{p}_0, t) - I_i(\mathbf{p}_i, t)|^2)\ , \\ BC_{s2}(Z, \mathbf{V}) &= \sum_{i=1}^{N-1} \Psi(|I_0(\hat{\mathbf{p}}_0, t+1) - I_i(\hat{\mathbf{p}}_i, t+1)|^2)\ . \end{aligned} \tag{3.14}$$

In den Gleichungen dient $\Psi(s^2)$ als eine nicht quadratische Kostenfunktion, um das Verfahren robust gegen Ausreißer und Signalrauschen zu machen [2]. Zu diesem Zweck wird die regularisierte Betragsfunktion verwendet.

$$\Psi(s^2) = \sqrt{s^2 + \epsilon^2} \quad , \epsilon = 0.001\ . \tag{3.15}$$

Weiter wird, um größere Verschiebungen zugehöriger Bildpunkte zwischen den Bildern zu erlauben, der Datenterm nicht linearisiert [3].

Nach der Erläuterung des Datenterms wird nun auf den Glattheitsterm $E_{Smooth}(Z, \mathbf{V})$ eingegangen. Der Glattheitsterm setzt eine gewisse abschnittsbedingte Glattheit der Tiefe Z und des Szenenflusses $\mathbf{V}$ voraus. Da Unstetigkeiten in den jeweiligen Tiefen- bzw. Bewegungsfelder unabhängig voneinander sind, werden die beiden Größen hierbei getrennt betrachtet. Dies drückt sich in der Gleichung des Glattheitsterms wie folgt aus:

$$E_{Smooth}(Z, \mathbf{V}) = \int_{\Omega_0} \mu \cdot S_Z(Z) + \alpha \cdot S_{\mathbf{V}}(\mathbf{V})\, dxdy\ . \tag{3.16}$$

Die wählbaren Parameter $\mu > 0$ und $\alpha > 0$ dienen jeweils der Gewichtung des Einflusses der beiden Terme auf das Energiefunktional

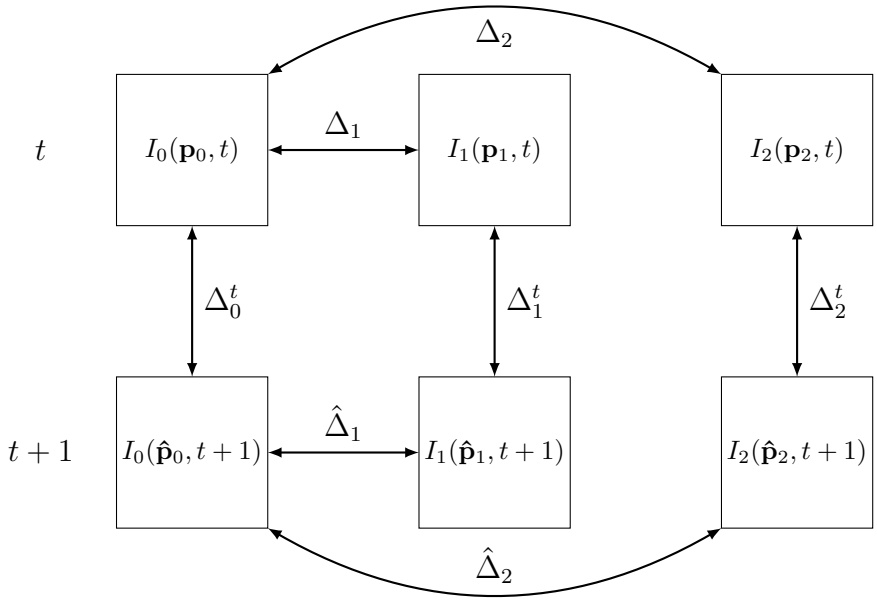


Abbildung 3.3.: Beispielhafter Aufbau des Datenterms mit 3 unterschiedlichen Kameraansichten zu jeweils den Zeitpunkten t und $t+1$. Die Terme Δ_i und $\hat{\Delta}_i$ bedienen die auferlegte Helligkeitswertkonstanzannahme zwischen den gezeigten Bildern zu den jeweiligen Zeitpunkte t und $t+1$. Zwischen diesen Zeitschritten sorgen die Δ_i^t Terme in den jeweiligen Bildsequenzen für die geforderte Helligkeitswertkonstanzannahme.

$E(Z, \mathbf{V})$. Mit isotropen Glattheitstermen 1. Ordnung für S_Z und $S_\mathbf{V}$ ergibt sich jeweils

$$\begin{aligned} S_Z(Z) &= \Psi(|\nabla Z|^2)\ , \\ S_\mathbf{V}(\mathbf{V}) &= \Psi(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\ , \end{aligned} \tag{3.17}$$

mit der gleichen zuvor vorgestellten Kostenfunktion $\Psi(s^2)$. Um eine kompaktere Notation für die folgenden Gleichungen zu ermöglichen, werden die Abkürzungen

$$\begin{aligned} \Delta_i^t &= I_i(\mathbf{p}_i, t) && - I_i(\hat{\mathbf{p}}_i, t+1) \\ \Delta_i &= I_0(\mathbf{p}_0, t) && - I_i(\mathbf{p}_i, t) \\ \hat{\Delta}_i &= I_0(\hat{\mathbf{p}}_0, t+1) && - I_i(\hat{\mathbf{p}}_i, t+1) \end{aligned} \tag{3.18}$$

für die Differenzausdrücke in den Datentermen eingeführt. Abbildung 3.3 verdeutlicht den Aufbau des Datenterms durch die neu eingeführ-

ten Ausdrücke. Die vollständige Formulierung des Energiefunktionals ergibt sich unter Anwendung der neuen Notation wie folgt:

$$\begin{aligned} E(Z,\mathbf{V}) = \int_{\Omega_0} & \sum_{i=0}^{N-1} \Psi(|\Delta_i^t|^2) + \sum_{i=1}^{N-1} \Psi(|\Delta_i|^2) + \sum_{i=1}^{N-1} \Psi(|\hat{\Delta}_i|^2) \\ & + \mu \cdot \Psi(|\nabla Z|^2) \\ & + \alpha \cdot \Psi(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\, dxdy\ . \end{aligned} \tag{3.19}$$

3.3. Euler-Lagrange-Gleichungen

Die Variationsrechnung zeigt, dass die minimierenden Funktionen Z und $\mathbf{V}$ des Energiefunktionals aus Gleichung 3.19 die Euler-Lagrange-Gleichungen erfüllen müssen. Für die Formulierung dieser Gleichungen werden zuerst die benötigten partiellen Ableitungen des Integranden gebildet. Die Gleichungen bezüglich der Variablen v und w werden nicht immer explizit formuliert, da diese analog zu den Gleichungen bezüglich der Variablen u gebildet werden können.

$$\begin{aligned} F_Z &= 2\sum_{i=0}^{N-1} \Psi'(|\Delta_i^t|^2)\Delta_i^t(\Delta_i^t)_Z + 2\sum_{i=1}^{N-1} \Psi'(|\Delta_i|^2)\Delta_i(\Delta_i)_Z \\ &\quad + 2\sum_{i=1}^{N-1} \Psi'(|\hat{\Delta}_i|^2)\hat{\Delta}_i(\hat{\Delta}_i)_Z\ , \\ F_{Z_x} &= 2\mu \cdot \Psi'(|\nabla Z|^2) \cdot Z_x\ , \\ F_{Z_y} &= 2\mu \cdot \Psi'(|\nabla Z|^2) \cdot Z_y\ , \end{aligned} \tag{3.20}$$

$$\begin{aligned} F_u &= 2\sum_{i=0}^{N-1} \Psi'(|\Delta_i^t|^2)\Delta_i^t(\Delta_i^t)_u + 2\sum_{i=1}^{N-1} \Psi'(|\hat{\Delta}_i|^2)\hat{\Delta}_i(\hat{\Delta}_i)_u\ , \\ F_{u_x} &= 2\alpha \cdot \Psi'(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2) \cdot u_x\ , \\ F_{u_y} &= 2\alpha \cdot \Psi'(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2) \cdot u_y\ . \end{aligned} \tag{3.21}$$

Es ist hierbei zu beachten, dass es zwischen den Bildern zum Zeitpunkt t keine Bewegung geben kann und deshalb gilt

$$(\Delta_i)_u = (I_0(\mathbf{p}_0, t) - I_i(\mathbf{p}_i, t))_u = 0\ . \tag{3.22}$$

Der Term konnte somit aus der Gleichung 3.21 herausgekürzt werden. Mithilfe der Gleichung 2.27 aus dem Grundlagenteil und den

oben definierten partiellen Ableitungen lassen sich die Euler-Lagrange-Gleichungen nun wie folgt formulieren:

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^t)^2)\Delta_i^t \cdot (\Delta_i^t)_Z + \sum_{i=1}^{N-1} \Psi'((\Delta_i)^2)\Delta_i \cdot (\Delta_i)_Z \\ &+ \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i)^2)\hat{\Delta}_i \cdot (\hat{\Delta}_i)_Z - \mu \cdot \mathrm{div}(\Psi'(|\nabla Z|^2)\nabla Z) \,, \end{aligned} \tag{3.23}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^t)^2)\Delta_i^t \cdot (\Delta_i^t)_u + \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i)^2)\hat{\Delta}_i \cdot (\hat{\Delta}_i)_u \\ &- \alpha \cdot \mathrm{div}(\Psi'|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\nabla u) \,, \end{aligned} \tag{3.24}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^t)^2)\Delta_i^t \cdot (\Delta_i^t)_v + \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i)^2)\hat{\Delta}_i \cdot (\hat{\Delta}_i)_v \\ &- \alpha \cdot \mathrm{div}(\Psi'(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\nabla v) \,, \end{aligned} \tag{3.25}$$

$$\begin{aligned} = 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^t)^2)\Delta_i^t \cdot (\Delta_i^t)_w + \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i)^2)\hat{\Delta}_i \cdot (\hat{\Delta}_i)_w \\ &- \alpha \cdot \mathrm{div}(\Psi'(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\nabla w) \,, \end{aligned} \tag{3.26}$$

mit den natürlichen Randbedingungen:

$$\mathbf{n}^\top \nabla Z = \mathbf{n}^\top \nabla u = \mathbf{n}^\top \nabla v = \mathbf{n}^\top \nabla w = 0 \,. \tag{3.27}$$

Durch die Verwendung der sub-quadratischen Kostenfunktion $\Psi(s^2)$, der nicht linearisierten Datenterme (vgl. Gleichung 3.14) und der perspektivischen Projektion ist das verwendete Energiefunktional $E(Z, \mathbf{V})$ nichtlinear und nicht konvex. Dies bedeutet, dass die hieraus resultierenden Euler-Lagrange-Gleichungen mehrere Lösungen besitzen, die zu unterschiedlich guten Minimierungen des Energiefunktionals führen. Um trotzdem eine möglichst gute Minimierung des Energiefunktionals zu erreichen, bedient sich das Szenenflussverfahren an der Methode von Brox *et al.* [3] und bettet die Euler-Lagrange-Gleichungen in ein Grob-zu-Fein-Schema mit zwei geschachtelten Fixpunktverfahren ein. Die Methodik für das numerische Lösen des Gleichungssystems und damit für die Minimierung des Energiefunktionals wird im Folgenden genauer beschrieben.

3.4. Minimierung

Die Minimierungsstrategie von Brox *et al.* [3] wird benutzt, um das nichtlineare und nicht konvexe System aus den Euler-Lagrange Gleichungen 3.23 - 3.26 zu lösen. Die Strategie selbst besteht aus mehreren Zwischenschritten, die in den jeweiligen Unterkapiteln dieses Abschnittes erläutert werden. Um über den Rest der Arbeit hinweg eine relativ kompakte Notation der Euler-Lagrange Gleichungen zu ermöglichen, wird zudem die Tensornotation [5, 6] eingeführt.

3.4.1. Fixpunktverfahren

Um die Nichtlinearität in den Variablen $Z, \mathbf{V}$ aufzulösen, wird ein erstes Fixpunktverfahren eingeführt. Dabei steht k als Variable für die Iterationsschritte des Verfahrens. Ein semi-impliziertes Schema in den Datentermen und ein vollständig impliziertes Schema in den Glattheitstermen liefert die nun neu formulierten Euler-Lagrange-Gleichungen:

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^{t,k+1})^2)\Delta_i^{t,k+1} \cdot (\Delta_i^{t,k})_Z \\ &+ \sum_{i=1}^{N-1} \Psi'((\Delta_i^{k+1})^2)\Delta_i^{k+1} \cdot (\Delta_i^{k})_Z \\ &+ \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i^{k+1})^2)\hat{\Delta}_i^{k+1} \cdot (\hat{\Delta}_i^{k})_Z \\ &- \mu \cdot \operatorname{div}(\Psi'(|\nabla Z^{k+1}|^2)\nabla Z^{k+1}) \,, \end{aligned} \tag{3.28}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'((\Delta_i^{t,k+1})^2)\Delta_i^{t,k+1} \cdot (\Delta_i^{t,k})_u \\ &+ \sum_{i=1}^{N-1} \Psi'((\hat{\Delta}_i^{k+1})^2)\hat{\Delta}_i^{k+1} \cdot (\hat{\Delta}_i^{k})_u \\ &- \alpha \cdot \operatorname{div}(\Psi'(|\nabla u^{k+1}|^2 + |\nabla v^{k+1}|^2 + |\nabla w^{k+1}|^2)\nabla u^{k+1}) \,. \end{aligned}$$

Die Gleichungen bezüglich der Variablen v und w können wiederum analog zu der Gleichung für u gebildet werden.

3.4.2. Inkrementelle Berechnung

In einem weiteren Schritt wird die Nichtlinearität der Datenterme beseitigt. Hierfür wird eine inkrementelle Berechnung der Variablen Z und $\mathbf{V}$ eingeführt. Diese setzen sich aus den bisher berechneten Werten und einem unbekannten Inkrement des alten Zeitschrittes k zusammen:

$$\begin{aligned} Z^{k+1} &= Z^k + dZ^k \ , \\ \mathbf{V}^{k+1} &= \mathbf{V}^k + d\mathbf{V}^k \ . \end{aligned} \tag{3.29}$$

Weiter werden mithilfe von Taylorreihen die Terme für $I_i(\mathbf{p}^{k+1}, t)$ und $I_i(\hat{\mathbf{p}}^{k+1}, t)$ linearisiert:

$$\begin{aligned} I_i(\mathbf{p}^{k+1}, t) \quad & \approx I_i(\mathbf{p}^k, t) + (I_i(\mathbf{p}^k, t))_Z \cdot dZ^k \ , \\ I_i(\hat{\mathbf{p}}^{k+1}, t+1) \quad & \approx I_i(\hat{\mathbf{p}}^k, t+1) + (I_i(\hat{\mathbf{p}}^k, t+1))_Z \cdot dZ^k \\ & \quad + (I_i(\hat{\mathbf{p}}^k, t+1))_u \cdot du^k \\ & \quad + (I_i(\hat{\mathbf{p}}^k, t+1))_v \cdot dv^k \\ & \quad + (I_i(\hat{\mathbf{p}}^k, t+1))_w \cdot dw^k \ . \end{aligned} \tag{3.30}$$

Für die Terme Δ^{k+1}, $\hat{\Delta}_i^{k+1}$ und $\Delta_i^{t,k+1}$ ergibt sich durch diese Linearisierung:

$$\begin{aligned} \Delta_i^{t,k+1} \quad & \approx \Delta_i^{t,k} + (\Delta_i^{t,k})_Z \cdot dZ^k \\ & \quad + (\Delta_i^{t,k})_u \cdot du^k + (\Delta_i^{t,k})_v \cdot dv^k + (\Delta_i^{t,k})_w \cdot dw^k \ , \\ \Delta_i^{k+1} \quad & \approx \Delta_i^k + (\Delta_i^k)_Z \cdot dZ^k \ , \\ \hat{\Delta}_i^{k+1} \quad & \approx \hat{\Delta}_i^k + (\hat{\Delta}_i^k)_Z \cdot dZ^k \\ & \quad + (\hat{\Delta}_i^k)_u \cdot du^k + (\hat{\Delta}_i^k)_v \cdot dv^k + (\hat{\Delta}_i^k)_w \cdot dw^k \ . \end{aligned} \tag{3.31}$$

Auf diese Weise werden die Nichtlinearitäten in den Variablen Z und $\mathbf{V}$ und in den Datentermen aufgelöst. Als letztes muss noch der Umgang mit der Nichtlinearität der Funktion $\Psi'(s^2)$ beschrieben werden. Um jedoch weiterhin die Notation für die Euler-Lagrange-Gleichungen kompakt zu halten, wird zuerst die Tensornotation [5, 6] eingeführt.

3.4.3. Tensornotation

Mittels der oben beschriebenen Taylorreihenlinearisierung aus Gleichung 3.31 lassen sich die folgenden Vektoren aufbauen:

$$\begin{aligned}
\mathbf{S}_i^{t,k} &= \left((\Delta_i^{t,k})_Z, (\Delta_i^{t,k})_u, (\Delta_i^{t,k})_v, (\Delta_i^{t,k})_w, \Delta_i^{t,k}\right)^\top , \\
\mathbf{S}_i^{k} &= \left((\Delta_i^{k})_Z, \Delta_i^{k}\right)^\top , \\
\hat{\mathbf{S}}_i^{k} &= \left((\hat{\Delta}_i^{k})_Z, (\hat{\Delta}_i^{k})_u, (\hat{\Delta}_i^{k})_v, (\hat{\Delta}_i^{k})_w, \hat{\Delta}_i^{k}\right)^\top .
\end{aligned} \tag{3.32}$$

Mithilfe dieser Vektoren können nun die folgenden symmetrischen Tensoren gebildet werden:

$$\begin{aligned}
T_i^{t,k} &= \mathbf{S}_i^{t,k}\mathbf{S}_i^{t,k\top} , \\
T_i^{k} &= \mathbf{S}_i^{k}\mathbf{S}_i^{k\top} , \\
\hat{T}_i^{k} &= \hat{\mathbf{S}}_i^{k}\hat{\mathbf{S}}_i^{k\top} .
\end{aligned} \tag{3.33}$$

So ergibt sich z.B. für den Tensor T_i^k:

$$T_i^k = \begin{pmatrix} ((\Delta_i^k)_Z)^2 & (\Delta_i^k)_Z \Delta_i^k \\ (\Delta_i^k)_Z \Delta_i^k & (\Delta_i^k)^2 \end{pmatrix} . \tag{3.34}$$

Aufgrund der perspektivischen Projektion ist die Berechnung der in den Tensoren verwendeten Bildableitungen nicht trivial und wird in Unterabschnitt 3.4.6 beschrieben.

Werden nun die Vektoren

$$\begin{aligned}
\mathbf{dZ}^k &= (dZ^k, 1)^\top , \\
\mathbf{dE}^k &= (dZ^k, du^k, dv^k, dw^k, 1)^\top
\end{aligned} \tag{3.35}$$

definiert, welche die Inkremente der jeweiligen Variablen enthalten, so kann die Euler-Lagrange-Gleichung aus 3.23, unter Berücksichti-

gung der zuvor eingeführten Linearisierung, wie folgt umgeschrieben werden:

$$\begin{aligned} 0 = & \sum_{i=0}^{N-1} \Psi' \left(\mathbf{dE}^{k\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^k \right) \cdot \left([T_i^{t,k}]_1 \cdot \mathbf{dE}^k \right) \\ & + \sum_{i=1}^{N-1} \Psi' \left(\mathbf{dZ}^{k\top} \cdot T_i^k \cdot \mathbf{dZ}^k \right) \cdot \left([T_i^k]_1 \cdot \mathbf{dZ}^k \right) \\ & + \sum_{i=1}^{N-1} \Psi' \left(\mathbf{dE}^{k\top} \cdot \hat{T}_i^k \cdot \mathbf{dE}^k \right) \cdot \left([\hat{T}_i^k]_1 \cdot \mathbf{dE}^k \right) \\ & - \mu \cdot \operatorname{div} \left(\Psi' \left(|\nabla(Z^k + dZ^k)|^2 \right) \cdot \nabla(Z^k + dZ^k) \right) \ , \end{aligned} \tag{3.36}$$

$$\begin{aligned} 0 = & \sum_{i=0}^{N-1} \Psi' \left(\mathbf{dE}^{k\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^k \right) \cdot \left([T_i^{t,k}]_2 \cdot \mathbf{dE}^k \right) \\ & + \sum_{i=1}^{N-1} \Psi' \left(\mathbf{dE}^{k\top} \cdot \hat{T}_i^k \cdot \mathbf{dE}^k \right) \cdot \left([\hat{T}_i^k]_2 \cdot \mathbf{dE}^k \right) \\ & - \alpha \cdot \operatorname{div} \left((\Psi')^k_{Sm,\mathbf{V}} \cdot \nabla(u^k + du^k) \right) \ , \end{aligned} \tag{3.37}$$

mit

$$\begin{aligned} & (\Psi')^k_{Sm,\mathbf{V}} := \\ & \quad \Psi'(|\nabla(u^k + du^k)|^2 + |\nabla(v^k + dv^k)|^2 + |\nabla(w^k + dw^k)|^2) \ . \end{aligned}$$

Die Ausdrücke $[T_i^{t,k}]_j$, $[T_i^k]_j$ und $[\hat{T}_i^k]_j$ in den Gleichungen 3.36 - 3.37 bezeichnen die j-te Zeile des jeweiligen Tensors. Die noch notwendigen Gleichungen bezüglich der Variablen v und w können wieder analog zur Gleichung 3.37 gebildet werden.

Zum Vergleich können die Euler-Lagrange-Gleichungen, die die Taylorreihen benutzen und nicht in der Tensornotation formuliert sind, in Anhang A eingesehen werden.

3.4.4. Verzögerte Nichtlinearitätsmethode

Als weiterer Schritt wird nun die Nichtlinearität der Ψ'-Terme behandelt. Hierfür wird ein zweites, inneres Fixpunktverfahren mit der Iterationsvariablen l für Ausdrücke, die den Term Ψ' enthalten, verwendet. Da die Ψ-Funktion konvex ist und damit genau eine Lösung besitzt, kann das nichtlineare Gleichungssystem als eine Serie von linearen Gleichungssystemen gelöst werden. Bei diesem Verfahren werden die nichtlinearen Ausdrücke in den Daten- und Glattheitstermen mit den Werten des vorherigen Iterationsschrittes ausgewertet.

Aus diesem Grund spricht man von der sogenannten *verzögerten Nichtlinearitätsmethode*. Das vollständige und nun *lineare* Gleichungssystem in den Inkrementen $dZ^{k,l+1}$, $du^{k,l+1}$, $dv^{k,l+1}$ und $dw^{k,l+1}$ ist gegeben durch:

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_1 \cdot \mathbf{dE}^{k,l+1}\right) \\ &+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dZ}^{k,l\top} \cdot T_i^{k} \cdot \mathbf{dZ}^{k,l}\right) \cdot \left([T_i^{k}]_1 \cdot \mathbf{dZ}^{k,l+1}\right) \\ &+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_1 \cdot \mathbf{dE}^{k,l+1}\right) \\ &- \mu \cdot \operatorname{div}\left(\Psi'\left(|\nabla(Z^k + dZ^{k,l})|^2\right) \cdot \nabla(Z^{k,l} + dZ^{k,l+1})\right) , \end{aligned} \tag{3.38}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_2 \cdot \mathbf{dE}^{k,l+1}\right) \\ &+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_2 \cdot \mathbf{dE}^{k,l+1}\right) \\ &- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}} \cdot \nabla(u^k + du^{k,l+1})\right) , \end{aligned} \tag{3.39}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\ &+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\ &- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}} \cdot \nabla(v^k + dv^{k,l+1})\right) , \end{aligned} \tag{3.40}$$

$$\begin{aligned} 0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\ &+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\ &- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}} \cdot \nabla(w^k + dw^{k,l+1})\right) , \end{aligned} \tag{3.41}$$

mit den unveränderten natürlichen Randbedingungen

$$\mathbf{n}^\top \nabla Z = \mathbf{n}^\top \nabla u = \mathbf{n}^\top \nabla v = \mathbf{n}^\top \nabla w = 0 \, . \tag{3.42}$$

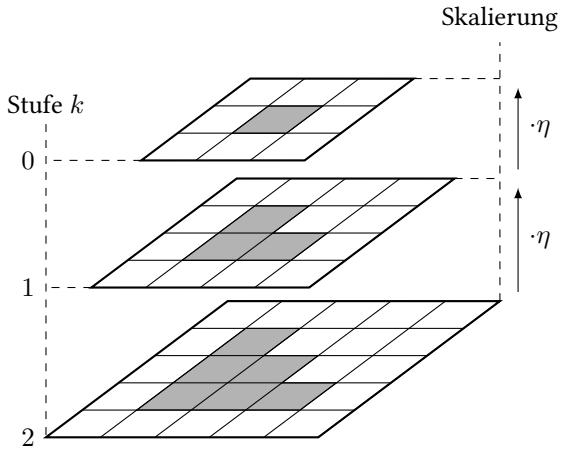


Abbildung 3.4.: Bildpyramide der Grob-zu-Fein-Strategie. Die Bilder werden auf jeder Stufe k jeweils in der Breite und Höhe um den Faktor $\eta \in (0, 1)$ skaliert.

3.4.5. Grob-zu-Fein-Strategie

Der letzte Schritt des Minimierungsverfahrens des Energiefunktionals bettet die bisherige Vorgehensweise in eine Grob-zu-Fein-Strategie ein. Das Ziel dieser Methode ist die Findung eines möglichst guten, lokalen oder sogar des globalen Minimums. Zu diesem Zweck werden die Eingabebilder stufenweise zu immer kleineren Auflösungen hin, jeweils um den Faktor $\eta \in (0, 1)$ skaliert. Abbildung 3.4 zeigt eine durch das Verfahren entstandene Bildpyramide. Das Gleichungssystem aus (3.38) - (3.41) wird zuerst auf der gröbsten Stufe $k = 0$ der Bildpyramide gelöst. Die daraus berechneten Werte für dZ^{k+1} und $\mathbf{dV}^{k+1}$ dienen dann mittels der Gleichung 3.29 als Initialisierung der nächsten Stufe $k + 1$. Das Verfahren wird solange wiederholt, bis man das Gleichungssystem auf der feinsten Auflösungsstufe gelöst hat und damit die endgültigen Ergebnisse für die Tiefe und den Szenenfluss erhält. Die Verkleinerung der Bilder hat einen Glättungseffekt, wodurch Details in den kleiner werdenden Bildern verloren gehen. Dies führt wiederum zu einer Glättung des nicht konvexen Energiefunktionals und hilft somit schlechte lokale Minimas zu vermeiden. Dieser Vorgang ist in Abbildung 3.5 verdeutlicht. Die Reduzierung der Bildgröße bedeutet jedoch auch, dass die verwendeten Kamerakalibierungsmatrizen angepasst werden müssen. Da eine Kalibrierungsmatrix durch den Ausdruck $(\mathbf{K}\ \ \mathbf{0})$ die Transformation von 3D-Kamerakoordinaten zu 2D-Bildpunktkoordinaten beschreibt, müssen die Parameter ebenfalls mit dem Skalierungsfaktor η multipliziert werden. Eine Kamerakali-

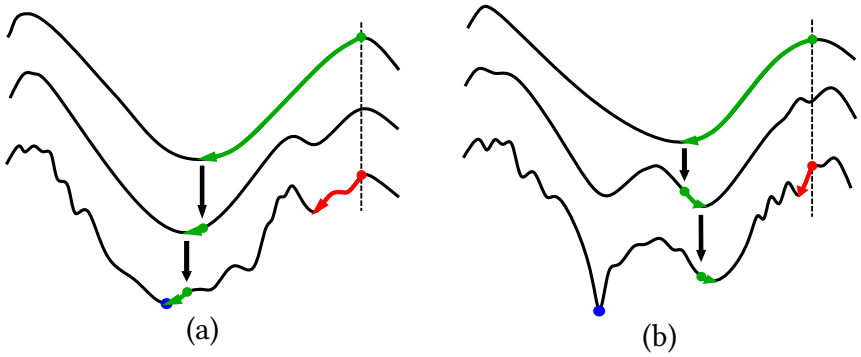


Abbildung 3.5.: Minimierung des Energiefunktionals [5]. Die **rote Linie** beschreibt eine Minimierung, die nur auf der feinsten Ebene arbeitet und hierdurch jeweils in einem schlechtem lokalen Minimum endet. Die **grüne Linie** beschreibt eine Minimierung mittels der Grob-zu-Fein-Strategie.
(a) Globales Minimum gefunden.
(b) Gutes lokales Minimum gefunden.

bierungsmatrix der Stufe $k+1$ kann durch die folgende Gleichung berechnet werden:

$$\mathbf{K}^{k+1} = \begin{pmatrix} \eta & 0 & 0 \\ 0 & \eta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \mathbf{K}^{k} = \begin{pmatrix} \eta \cdot s_x^k & 0 & \eta \cdot o_x^k \\ 0 & \eta \cdot s_y^k & \eta \cdot o_y^k \\ 0 & 0 & 1 \end{pmatrix} . \tag{3.43}$$

Diese Änderung wirkt sich auch auf die vollständige Kameraprojektionsmatrix aus. Deshalb ist zu beachten, dass diese auch entsprechend neu berechnet werden muss. (s. Abschnitt 2.2).

Auf jeder Stufe der Grob-zu-Fein-Strategie wird das SOR-Verfahren [34] zum Lösen des linearen Gleichungssystems aus (3.38) - (3.41) verwendet. Um das Verfahren jedoch überhaupt anwenden zu können, ist es notwendig, Ausdrücke wie $I_i(\mathbf{p_1}, t)$ und deren Ableitungen nach Z, u, v und w, wie sie zum Beispiel in Termen wie $(\hat{\Delta}^k)_Z$ verwendet werden, zu berechnen. Die Berechnungen dieser Größen wird im folgenden Unterabschnitt genauer beschrieben.

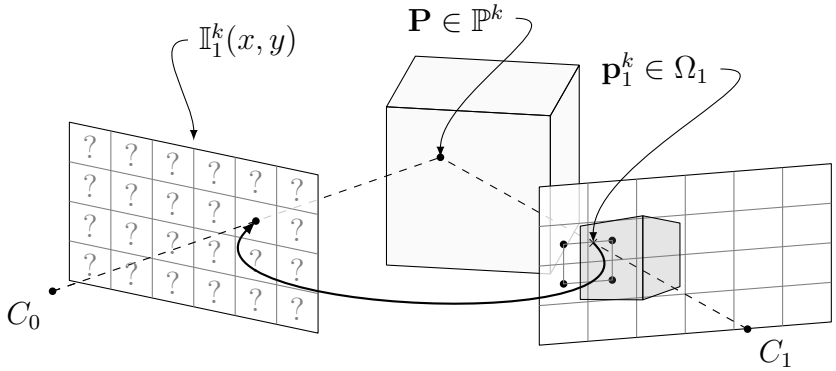


Abbildung 3.6.: Rückprojektion des Bildes $I_1(\mathbf{p}_1^k, t)$ auf die Bildebene der Referenzkamera C_0. Abbildung aus [20].

3.4.6. Berechnung der Rückprojektionen und zugehörigen Ableitungen

Der Ausdruck $I_i(\mathbf{p}_1^k, t)$ wird als Rückprojektion $\mathbb{I}_i^k(x, y)$ des Bildes in die Bildebene der Referenzkamera verstanden. Es gilt daher

$$\mathbb{I}_i^k(x, y) = I_i(\mathbf{p}_1^k, t) \; . \tag{3.44}$$

Hierfür werden die Bildpunkte des Referenzbildes mithilfe der bisher berechneten Tiefe Z^k in die 3D-Szene zurückprojiziert. Anschließend werden die 3D-Punkte wiederum durch die vollständige Kameraprojektionsmatrix M_i^k auf die Bildebene Ω_i projiziert. Der Wert an dieser Stelle des Bildes kann nun dem entsprechenden Bildpunkt in $\mathbb{I}_i^k(x, y)$ zugewiesen werden. Da der projizierte Punkt $\mathbf{p}_i$ (Berechnung s. Gleichung 3.9) jedoch häufig zwischen den Bildpunkten liegt, wird der Wert durch eine bilineare Interpolation zwischen den vier naheliegensten Bildpunkten angenähert. Das Verfahren ist in Abbildung 3.6 noch einmal verdeutlicht. Analog zu dieser Vorgehensweise kann der Ausdruck für $I_i(\hat{\mathbf{p}}_i^k, t + 1)$ berechnet werden. Hier muss jedoch die Verschiebung des 3D-Punktes durch den Szenenfluss $\mathbf{V}$ berücksichtigt werden. Dies schlägt sich in der Berechnung für die Koordinaten des Punktes $\hat{\mathbf{p}}_i^k$ (s. Gleichung 3.11) nieder.

Für die Tensoren $T_i^{t,k}$, T_i^k sowie $\hat{T}_i^k$ werden die partielle Ableitung nach Z für $I_i(\mathbf{p}_i^k, t)$ sowie die partiellen Ableitungen nach Z und $\mathbf{V}$ für den Ausdruck $I_i(\hat{\mathbf{p}}_i^k, t+1)$ benötigt. Die Anwendung der Kettenregel liefert:

$$\begin{aligned}(I_i(\mathbf{p}_i^k, t))_Z &= (\nabla_i I_i(\mathbf{p}_i^k, t))^\top \cdot \partial_Z \mathbf{p}_i^k \\ &= \begin{pmatrix}\partial_{x_i} I_i(\mathbf{p}_i^k, t) & \partial_{y_i} I_i(\mathbf{p}_i^k, t)\end{pmatrix} \cdot \begin{pmatrix}\partial_Z x_i \\ \partial_Z y_i\end{pmatrix} \\ &= \partial_{x_i} I_i(\mathbf{p}_i^k, t) \cdot \partial_Z x_i + \partial_{y_i} I_i(\mathbf{p}_i^k, t) \cdot \partial_Z y_i \ .\end{aligned} \tag{3.45}$$

Mithilfe der Gleichung 3.9 und der Quotientenregel können wiederum die Ausdrücke $\partial_Z x_i$ und $\partial_Z y_i$ bestimmt werden:

$$\begin{aligned}\partial_Z x_i &= \frac{ad - bc}{(c \cdot Z + d)^2} \ , \\ \partial_Z y_i &= \frac{\check{a}d - \check{b}c}{(c \cdot Z + d)^2} \ .\end{aligned} \tag{3.46}$$

Analog hierzu werden nun die partiellen Ableitungen für den Ausdruck $I_i(\hat{\mathbf{p}}_i^k, t+1)$ hergeleitet:

$$\begin{aligned}(I_i(\hat{\mathbf{p}}_i^k, t+1))_Z &= \partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_Z \hat{x}_i \\ &\quad + \partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_Z \hat{y}_i \ , \\ (I_i(\hat{\mathbf{p}}_i^k, t+1))_u &= \partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_u \hat{x}_i \\ &\quad + \partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_u \hat{y}_i \ , \\ (I_i(\hat{\mathbf{p}}_i^k, t+1))_v &= \partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_v \hat{x}_i \\ &\quad + \partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_v \hat{y}_i \ , \\ (I_i(\hat{\mathbf{p}}_i^k, t+1))_w &= \partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_w \hat{x}_i \\ &\quad + \partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1) \cdot \partial_w \hat{y}_i \ .\end{aligned} \tag{3.47}$$

Für die partiellen Ableitungen der Bildpunktkoordinaten werden zunächst die Abkürzungen e, $\check{e}$ und f definiert:

$$\hat{x}_i = \frac{\overbrace{a \cdot Z + M^i_{11} \cdot u + M^i_{12} \cdot v + M^i_{13} \cdot w + b}^{e}}{\underbrace{c \cdot Z + M^i_{31} \cdot u + M^i_{32} \cdot v + M^i_{33} \cdot w + d}_{f}} ,$$

$$\hat{y}_i = \frac{\overbrace{\check{a} \cdot Z + M^i_{21} \cdot u + M^i_{22} \cdot v + M^i_{23} \cdot w + \check{b}}^{\check{e}}}{\underbrace{c \cdot Z + M^i_{31} \cdot u + M^i_{32} \cdot v + M^i_{33} \cdot w + d}_{f}} . \tag{3.48}$$

Hiermit können die benötigten Ableitungen nun wie folgt ausgedrückt werden:

$$\begin{aligned}
\partial_Z \hat{x}_i &= \frac{a \cdot f - c \cdot e}{f^2} , & \partial_Z \hat{y}_i &= \frac{\check{a} \cdot f - c \cdot \check{e}}{f^2} , \\
\partial_u \hat{x}_i &= \frac{M^i_{11} \cdot f - M^i_{31} \cdot e}{f^2} , & \partial_u \hat{y}_i &= \frac{M^i_{21} \cdot f - M^i_{31} \cdot \check{e}}{f^2} , \\
\partial_v \hat{x}_i &= \frac{M^i_{12} \cdot f - M^i_{32} \cdot e}{f^2} , & \partial_v \hat{y}_i &= \frac{M^i_{22} \cdot f - M^i_{32} \cdot \check{e}}{f^2} , \\
\partial_w \hat{x}_i &= \frac{M^i_{13} \cdot f - M^i_{33} \cdot e}{f^2} , & \partial_w \hat{y}_i &= \frac{M^i_{23} \cdot f - M^i_{33} \cdot \check{e}}{f^2} .
\end{aligned} \tag{3.49}$$

Um die Rückprojektion der Bildgradienten $\nabla_i I_i(\mathbf{p}^k_i, t)$ und $\nabla_i I_i(\hat{\mathbf{p}}^k_i, t+1)$ zu berechnen, wird die von Maurer [20] vorgeschlagene und weniger aufwändige Methode gegenüber der Methode von Basha *et al.* [9] verwendet. Hierfür wird der Gradient eines Bildes I_i berechnet und anschließend, nach der am Anfang in diesem Unterabschnitt beschriebenen Methode, in das Referenzbild zurückprojiziert.

3.4.7. Initialisierung

Durch die inkrementelle Berechnung werden die Größen Z^k und $\mathbf{V}^k$ auf jeder Stufe der Grob-zu-Fein-Strategie durch die Ergebnisse der vorherigen Stufe initialisiert (s. Gleichung 3.29). Weiterhin offen bleibt die Frage der Initialisierung von Z^0 und $\mathbf{V}^0$ sowie deren Inkremente dZ^0 und $\mathbf{dV}^0$. Glücklicherweise gestaltet sich die Initialisierung

der Inkremente und des Szenenflusses als trivial, so dass diese mit 0 initialisiert werden können:

$$\begin{aligned} dZ^0 &= 0 \,, \\ \mathbf{dV}^0 &= \left(du^0, dv^0, dw^0\right)^\top = (0,0,0)^\top \,, \\ \mathbf{V}^0 &= \left(u^0, v^0, w^0\right)^\top = (0,0,0)^\top \,. \end{aligned} \tag{3.50}$$

Die Initialisierung $Z^0 = 0$ würde die Rückprojektion der Punkte $\mathbf{p}_0$ und $\hat{\mathbf{p}}_0$ in die 3D-Szene verhindern (s. Gleichung 3.5), da alle 3D-Punkte direkt im Kamerazentrum C_0 liegen würden. Um das Problem zu lösen, wird das Z-Scan-Verfahren aus [20] benutzt. Das Z-Scan-Verfahren tastet einen Z-Wertebereich ab und berechnet die Abweichung des rückprojizierten Bildes zu dem Referenzbild. Der Z-Wert, der mit der kleinsten Abweichung vom Referenzbild verknüpft ist, wird für die Initialisierung von Z^0 verwendet. Da nur eine grobe Initialisierung, die nicht 0 ist, benötigt wird, ist es für das Verfahren ausreichend neben dem Referenzbild nur ein weiteres Bild zu benutzen. Damit die Rückprojektion nicht von $\mathbf{V}$ abhängig ist, muss das zweite Bild, wie das Referenzbild, zum Zeitpunkt t aufgenommen worden sein. Unter Verwendung einer quadratischen Kostenfunktion ergibt sich:

$$E_{\text{Init}}(Z) = \frac{1}{w_1} \int_{\Omega_0} \left(I_0(\mathbf{p}_0, t) - I_i(\mathbf{p}_i, t)\right)^2 \, dxdy \,, \tag{3.51}$$

wobei w_1 die Anzahl der Bildpunkte ist, die auf der Bildebene des Bildes $(I_i(\mathbf{p}_i, t))$ projiziert worden sind.

3.4.8. Diskretisierung

Dieser Abschnitt beschreibt die für ein numerisches Lösungsverfahren notwendige Diskretisierung der Euler-Lagrange-Gleichungen. Da das Verfahren für jeden Bildpunkt im Referenzbild die Tiefe Z^k und die Bewegung $\mathbf{V}^k$ des zugehörigen 3D-Punktes bestimmt, werden diese Größen ebenfalls durch ein rechteckiges Gitter mit der Gittergröße $h_x^k \times h_y^k$ modelliert. Der diskrete Wert an der Stelle (i, j) wird durch eine entsprechende Tiefstellung des Koordinatentupels kenntlich gemacht

(z.B. $Z^k_{i,j}$). Die diskretisierte Version der Euler-Lagrange-Gleichungen aus 3.38 - 3.41 ist dann:

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot T^{t,k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([T^{t,k}_{i\ i,j}]_1 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dZ}^{k,l\top}_{i,j} \cdot T^{k}_{i\ i,j} \cdot \mathbf{dZ}^{k,l}_{i,j}\right) \cdot \left([T^{k}_{i\ i,j}]_1 \cdot \mathbf{dZ}^{k,l+1}_{i,j}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot \hat{T}^{k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([\hat{T}^{k}_{i\ i,j}]_1 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&- \mu \cdot \operatorname{div}\left(\Psi'\left(|\nabla(Z^k_{i,j} + dZ^{k,l}_{i,j})|^2\right) \cdot \nabla(Z^{k,l}_{i,j} + dZ^{k,l+1}_{i,j})\right) ,
\end{aligned} \tag{3.52}$$

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot T^{t,k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([T^{t,k}_{i\ i,j}]_2 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot \hat{T}^{k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([\hat{T}^{k}_{i\ i,j}]_2 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}\ i,j} \cdot \nabla(u^k_{i,j} + du^{k,l+1}_{i,j})\right) ,
\end{aligned} \tag{3.53}$$

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot T^{t,k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([T^{t,k}_{i\ i,j}]_3 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot \hat{T}^{k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([\hat{T}^{k}_{i\ i,j}]_3 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}\ i,j} \cdot \nabla(v^k_{i,j} + dv^{k,l+1}_{i,j})\right) ,
\end{aligned} \tag{3.54}$$

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot T^{t,k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([T^{t,k}_{i\ i,j}]_4 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}_{i,j} \cdot \hat{T}^{k}_{i\ i,j} \cdot \mathbf{dE}^{k,l}_{i,j}\right) \cdot \left([\hat{T}^{k}_{i\ i,j}]_4 \cdot \mathbf{dE}^{k,l+1}_{i,j}\right) \\
&- \alpha \cdot \operatorname{div}\left((\Psi')^{k,l}_{Sm,\mathbf{V}\ i,j} \cdot \nabla(w^k_{i,j} + dw^{k,l+1}_{i,j})\right) .
\end{aligned} \tag{3.55}$$

Bevor die Gleichungen tatsächlich durch ein numerisches Verfahren lösbar sind, muss noch geklärt werden, wie die Ableitungen in den Daten- und die Glattheitsterme selbst diskretisiert werden können.

Für die Bildgradienten in den Datentermen wird ein zentraler Differenzenquotient 4. Ordnung verwendet. Hierdurch ergibt sich für den Gradienten eines Bildes I_i an der Stelle (i, j):

$$\begin{aligned}
\partial_x I^k_{i\ i,j} &= \frac{I^k_{i\ i-2,j} - 8I^k_{i\ i-1,j} + 8I^k_{i\ i+1,j} - I^k_{i\ i+2,j}}{12h^k_x} , \\
\partial_y I^k_{i\ i,j} &= \frac{I^k_{i\ i,j-2} - 8I^k_{i\ i,j-1} + 8I^k_{i\ i,j+1} - I^k_{i\ i,j+2}}{12h^k_y} .
\end{aligned} \tag{3.56}$$

Die Approximierung der diskretisierten Glattheitsterme ergibt sich jeweils zu:

$$\begin{aligned}
&\mu \cdot \operatorname{div}\left(\Psi'\left(|\nabla(Z_{i,j}^{k}+dZ_{i,j}^{k,l})|^2\right)\cdot\nabla(z_{i,j}^{k}+dz_{i,j}^{k,l+1})\right)\\
&\quad\approx \mu \sum_{n\in(x,y)}\sum_{(\tilde{i},\tilde{j})\in\mathcal{N}_n(i,j)} \cdot\kappa_{i,j,\tilde{i},\tilde{j},n}^{k,l}(z_{\tilde{i},\tilde{j}}^{k,l}+dz_{\tilde{i},\tilde{j}}^{k,l+1}-z_{i,j}^{k,l}-dz_{i,j}^{k,l+1}),
\end{aligned} \tag{3.57}$$

$$\begin{aligned}
&\alpha \cdot \operatorname{div}\left((\Psi')_{Sm,\mathbf{V}\ i,j}^{k,l}\cdot\nabla(u_{i,j}^{k}+du_{i,j}^{k,l+1})\right)\\
&\quad\approx \alpha \sum_{n\in(x,y)}\sum_{(\tilde{i},\tilde{j})\in\mathcal{N}_n(i,j)} \cdot\xi_{i,j,\tilde{i},\tilde{j},n}^{k,l}(u_{\tilde{i},\tilde{j}}^{k,l}+du_{\tilde{i},\tilde{j}}^{k,l+1}-u_{i,j}^{k,l}-du_{i,j}^{k,l+1}),
\end{aligned} \tag{3.58}$$

$$\begin{aligned}
&\alpha \cdot \operatorname{div}\left((\Psi')_{Sm,\mathbf{V}\ i,j}^{k,l}\cdot\nabla(v_{i,j}^{k}+dv_{i,j}^{k,l+1})\right)\\
&\quad\approx \alpha \sum_{n\in(x,y)}\sum_{(\tilde{i},\tilde{j})\in\mathcal{N}_n(i,j)} \cdot\xi_{i,j,\tilde{i},\tilde{j},n}^{k,l}(v_{\tilde{i},\tilde{j}}^{k,l}+dv_{\tilde{i},\tilde{j}}^{k,l+1}-v_{i,j}^{k,l}-dv_{i,j}^{k,l+1}),
\end{aligned} \tag{3.59}$$

$$\begin{aligned}
&\alpha \cdot \operatorname{div}\left((\Psi')_{Sm,\mathbf{V}\ i,j}^{k,l}\cdot\nabla(w_{i,j}^{k}+dw_{i,j}^{k,l+1})\right)\\
&\quad\approx \alpha \sum_{n\in(x,y)}\sum_{(\tilde{i},\tilde{j})\in\mathcal{N}_n(i,j)} \cdot\xi_{i,j,\tilde{i},\tilde{j},n}^{k,l}(w_{\tilde{i},\tilde{j}}^{k,l}+dw_{\tilde{i},\tilde{j}}^{k,l+1}-w_{i,j}^{k,l}-dw_{i,j}^{k,l+1}),
\end{aligned} \tag{3.60}$$

mit

$$\kappa_{i,j,\tilde{i},\tilde{j},n}^{k,l}=\frac{\Psi'\left(|\nabla(Z_{\tilde{i},\tilde{j}}^{k}+dZ_{\tilde{i},\tilde{j}}^{k,l})|^2\right)+\Psi'\left(|\nabla(Z_{i,j}^{k}+dZ_{i,j}^{k,l})|^2\right)}{2\cdot h_n^2} \tag{3.61}$$

und

$$\xi_{i,j,\tilde{i},\tilde{j},n}^{k,l}=\frac{(\Psi')_{Sm,\mathbf{V}\ \tilde{i},\tilde{j}}^{k,l}+(\Psi')_{Sm,\mathbf{V}\ i,j}^{k,l}}{2\cdot h_n^2}\,. \tag{3.62}$$

Weiter beschreibt der Ausdruck $\mathcal{N}_n(i,j)$ die Menge der benachbarten Bildpunkte in Richung $n\in(x,y)$. In x-Richung ergeben sich somit die Bildpunkte $(i-1,j)$ und $(i+1,j)$ und in y-Richtung die Punkte $(i,j-1)$ und $(i,j+1)$. Die Herleitung dieser Approximation ist in Anhang B beschrieben und die Matrize in Tabelle B.1 verdeutlicht den Aufbau der Terme, die durch die doppelten Summenzeichen entstehen.

In den Gleichungen 3.61 und 3.62 werden die Gradienten für $Z_{i,j}^{k}$, $dZ_{i,j}^{k,l}$ sowie $\mathbf{V}^k$ und $d\mathbf{V}^{k,l}$ verwendet. Diese werden wieder jeweils mithilfe eines zentralen Differenzenquotienten 4. Ordnung bestimmt:

$$\begin{aligned}
\partial_x Z_{i,j}^{k}&=\frac{Z_{i-2,j}^{k}-8Z_{i-1,j}^{k}+8Z_{i+1,j}^{k}-Z_{i+2,j}^{k}}{12h_x^k}\,,\\
\partial_y Z_{i,j}^{k}&=\frac{Z_{i,j-2}^{k}-8Z_{i,j-1}^{k}+8Z_{i,j+1}^{k}-Z_{i,j+2}^{k}}{12h_y^k}\,.
\end{aligned} \tag{3.63}$$

Die Gradienten bezüglich der verbleibenden Variablen können analog zu den hier gezeigten Gleichungen bestimmt werden.

Die nun linearisierte und diskretisierten Euler-Lagrange-Gleichungen können durch ein numerisches Verfahren gelöst werden. Hierfür wird im Rahmen dieser Arbeit ein SOR-Verfahren [34] verwendet.

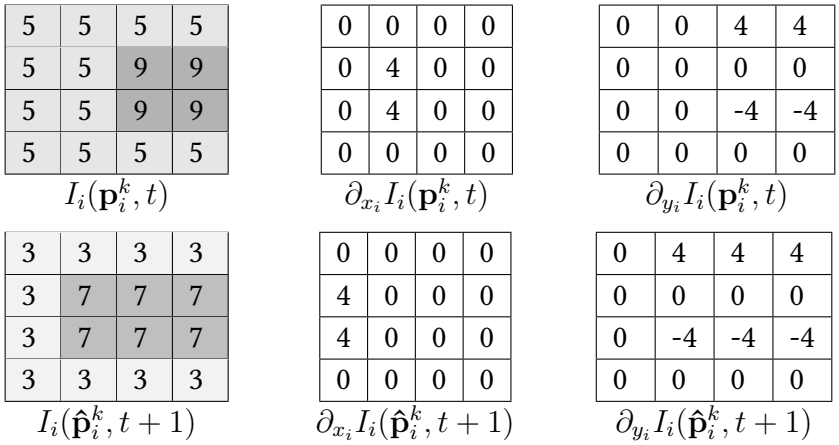


Abbildung 3.7.: Die Gradientenkonstanzannahme betrachtet jeweils die Änderungsrate der Helligkeitswerte in x- und y-Richtung. Die Änderungssrate ist hierbei unabhängig von globalen Beleuchtungsänderungen, so dass eine Zuordnung der Werte zwischen den Bildern weiterhin möglich ist.

3.5. Gradientenkonstanzannahme

Dieser Abschnitt beschäftigt sich mit einer Erweiterung des bisher vorgestellten Szenenflussverfahrens. Zusätzlich zur Helligkeitswertkonstanzannahme wird nun auch noch die Gradientenkonstanzannahme [3, 6] zur Berechnung des Szenenflusses verwendet. Die Gradientenkonstanzannahme ermöglicht dem Verfahren mit Beleuchtungsunterschieden zwischen den einzelnen Bildern, die die bisherige Helligkeitswertkonstanzannahme verletzen, umzugehen. Die Funktionsweise dieser neuen Konstanzannahme ist in Abbildung 3.7 beschrieben. Basha *et al.* [9] verzichteten auf diese Erweiterung, doch Maurer [20] hat gezeigt, dass sein Verfahren zur Stereorekonstruktion von der Gradientenkonstanzannahme profitiert. Da beide Verfahren auf der 3D-Parametrisierung von [9] basieren, erscheint die Gradientenkonstanzannahme auch als sinnvolle Erweiterung des Szenenflussverfahrens.

Die Erweiterung des Datenterms durch die Gradientenkonstanzannahme mit getrennter Robustifizierung [6] liest sich wie folgt:

$$\begin{aligned} E_{Data}(Z, \mathbf{V}) = \int_{\Omega_0} & BC_m(Z, \mathbf{V}) + BC_{s1}(Z) + BC_{s2}(Z, \mathbf{V}) \\ & + \gamma \cdot (GC_m(Z, \mathbf{V}) + GC_{s1}(Z) + GC_{s2}(Z, \mathbf{V}))\, dxdy\ . \end{aligned} \tag{3.64}$$

Dabei wird der Parameter $\gamma \geq 0$ zur Steuerung des Einflusses der Gradientenkonstanzannahmen verwendet. So erhält man für $\gamma = 0$ wieder den Datenterm des ursprünglichen Verfahrens. Im Detail ist die Gradientenkonstanzannahme zwischen den Eingabebildern gegeben durch:

$$\begin{aligned} \nabla I_i(\mathbf{p}_i, t) \quad & - \nabla I_i(\mathbf{p}_i, t+1) \quad = 0\ , \\ \nabla I_0(\mathbf{p}_0, t) \quad & - \nabla I_i(\mathbf{p}_i, t) \quad = 0\ , \\ \nabla I_0(\hat{\mathbf{p}}_0, t+1) \quad & - \nabla I_i(\hat{\mathbf{p}}_i, t+1) \quad = 0\ . \end{aligned} \tag{3.65}$$

Unter der Verwendung einer getrennten Robustifizierung [6] ergibt sich somit das zu minimierende Energiefunktional $E(Z, \mathbf{V})$ mit Gradientenkonstanzannahme zu:

$$\begin{aligned} E(Z, \mathbf{V}) = \int_{\Omega_0} & \sum_{i=0}^{N-1} \Psi(|\Delta_i^t|^2) + \sum_{i=1}^{N-1} \Psi(|\Delta_i|^2) + \sum_{i=1}^{N-1} \Psi(|\hat{\Delta}_i|^2) \\ & + \gamma \cdot \Big(\sum_{i=0}^{N-1} \Psi(|\nabla \Delta_i^t|^2) + \sum_{i=1}^{N-1} \Psi(|\nabla \Delta_i|^2) + \sum_{i=1}^{N-1} \Psi(|\nabla \hat{\Delta}_i|^2) \Big) \\ & + \mu \cdot \Psi(|\nabla Z|^2) + \alpha \cdot \Psi(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)\, dxdy\ . \end{aligned} \tag{3.66}$$

Die Minimierung des Energiefunktionals erfolgt analog zu der Vorgehensweise des ursprünglichen Verfahrens (s. Abschnitt 3.4), so dass im Folgenden nur noch die wichtigsten Gleichungen für das weitere Verständnis aufgeführt werden. Nach Einführung des äußeren Fixpunktverfahrens und der Linearisierung durch die Taylorreihen können die folgenden neuen symmetrischen Tensoren aufgebaut werden:

$$\begin{aligned} T_{\nabla i}^{t,k} &= \mathbf{S}_{\nabla i}^{t,k} \mathbf{S}_{\nabla i}^{t,k\top}\ , \\ T_{\nabla i}^{k} &= \mathbf{S}_{\nabla i}^{k} \mathbf{S}_{\nabla i}^{k\top}\ , \\ \hat{T}_{\nabla i}^{k} &= \hat{\mathbf{S}}_{\nabla i}^{k} \hat{\mathbf{S}}_{\nabla i}^{k\top}\ . \end{aligned} \tag{3.67}$$

Die hierfür verwendeten Vektoren ergeben sich zu:

$$\begin{aligned}
\mathbf{S}_{\nabla i}^{t,k} &= \left((\nabla\Delta_i^{t,k})_Z, (\nabla\Delta_i^{t,k})_u, (\nabla\Delta_i^{t,k})_v, (\nabla\Delta_i^{t,k})_w, \nabla\Delta_i^{t,k}\right)^\top , \\
\mathbf{S}_{\nabla i}^{k} &= \left((\nabla\Delta_i^{k})_Z, \nabla\Delta_i^{k}\right)^\top , \\
\hat{\mathbf{S}}_{\nabla i}^{k} &= \left((\nabla\hat{\Delta}_i^{k})_Z, (\nabla\hat{\Delta}_i^{k})_u, (\nabla\hat{\Delta}_i^{k})_v, (\nabla\hat{\Delta}_i^{k})_w, \nabla\hat{\Delta}_i^{k}\right)^\top .
\end{aligned} \tag{3.68}$$

Hierdurch lassen sich wieder analog zu den Gleichungen (3.38) - (3.41) die linearisierte Version der Euler-Lagrange-Gleichungen formulieren:

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot T_i^{t,k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([T_i^{t,k}]_1\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dZ}^{k,l\top}\cdot T_i^{k}\cdot\mathbf{dZ}^{k,l}\right)\cdot\left([T_i^{k}]_1\cdot\mathbf{dZ}^{k,l+1}\right) \\
&+\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot \hat{T}_i^{k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([\hat{T}_i^{k}]_1\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\gamma\cdot\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot T_{\nabla i}^{t,k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([T_{\nabla i}^{t,k}]_1\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\gamma\cdot\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dZ}^{k,l\top}\cdot T_{\nabla i}^{k}\cdot\mathbf{dZ}^{k,l}\right)\cdot\left([T_{\nabla i}^{k}]_1\cdot\mathbf{dZ}^{k,l+1}\right) \\
&+\gamma\cdot\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot \hat{T}_{\nabla i}^{k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([\hat{T}_{\nabla i}^{k}]_1\cdot\mathbf{dE}^{k,l+1}\right) \\
&-\mu\cdot\mathrm{div}\left(\Psi'\left(|\nabla(Z^k+dZ^{k,l})|^2\right)\cdot\nabla(Z^{k,l}+dZ^{k,l+1})\right) ,
\end{aligned} \tag{3.69}$$

$$\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot T_i^{t,k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([T_i^{t,k}]_2\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot \hat{T}_i^{k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([\hat{T}_i^{k}]_2\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\gamma\cdot\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot T_{\nabla i}^{t,k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([T_{\nabla i}^{t,k}]_2\cdot\mathbf{dE}^{k,l+1}\right) \\
&+\gamma\cdot\sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top}\cdot \hat{T}_{\nabla i}^{k}\cdot\mathbf{dE}^{k,l}\right)\cdot\left([\hat{T}_{\nabla i}^{k}]_2\cdot\mathbf{dE}^{k,l+1}\right) \\
&-\alpha\cdot\mathrm{div}\left((\Psi')_{Sm,\mathbf{V}}^{k,l}\cdot\nabla(u^k+du^{k,l+1})\right) ,
\end{aligned} \tag{3.70}$$

$$
\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \gamma \cdot \sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_{\nabla i}^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_{\nabla i}^{t,k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \gamma \cdot \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_{\nabla i}^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_{\nabla i}^{k}]_3 \cdot \mathbf{dE}^{k,l+1}\right) \\
&- \alpha \cdot \operatorname{div}\left((\Psi')_{Sm,\mathbf{V}}^{k,l} \cdot \nabla(v^k + dv^{k,l+1})\right) ,
\end{aligned}
\tag{3.71}
$$

$$
\begin{aligned}
0 = &\sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_i^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_i^{t,k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_i^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_i^{k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \gamma \cdot \sum_{i=0}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot T_{\nabla i}^{t,k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([T_{\nabla i}^{t,k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\
&+ \gamma \cdot \sum_{i=1}^{N-1} \Psi'\left(\mathbf{dE}^{k,l\top} \cdot \hat{T}_{\nabla i}^{k} \cdot \mathbf{dE}^{k,l}\right) \cdot \left([\hat{T}_{\nabla i}^{k}]_4 \cdot \mathbf{dE}^{k,l+1}\right) \\
&- \alpha \cdot \operatorname{div}\left((\Psi')_{Sm,\mathbf{V}}^{k,l} \cdot \nabla(w^k + dw^{k,l+1})\right) ,
\end{aligned}
\tag{3.72}
$$

mit den natürlichen Randbedingungen

$$
\mathbf{n}^\top \nabla Z = \mathbf{n}^\top \nabla u = \mathbf{n}^\top \nabla v = \mathbf{n}^\top \nabla w = 0 \, . \tag{3.73}
$$

Durch die Verwendung der Gradientenkonstanzannahme werden auch weitere Ableitungen bezüglich der Variablen Z und $\mathbf{V}$ benötigt. Dies soll am Beispiel des 2×2 Tensors $T_{\nabla i}^k$ verdeutlicht werden:

$$
\begin{aligned}
T_{\nabla i}^k = \mathbf{S}_{\nabla i}^k \mathbf{S}_{\nabla i}^{k\top} &= \begin{pmatrix} (\nabla\Delta_i^k)_Z^\top \\ (\nabla\Delta_i^k)^\top \end{pmatrix} \cdot \begin{pmatrix} (\nabla\Delta_i^k)_Z & (\nabla\Delta_i^k) \end{pmatrix} \\
&= \begin{pmatrix} (\nabla\Delta_i^k)_Z^\top (\nabla\Delta_i^k)_Z & (\nabla\Delta_i^k)_Z^\top (\nabla\Delta_i^k) \\ (\nabla\Delta_i^k)_Z^\top (\nabla\Delta_i^k) & (\nabla\Delta_i^k)^\top (\nabla\Delta_i^k) \end{pmatrix} \\
&= \begin{pmatrix} ((\partial_{x_i}\Delta_i^k)_Z)^2 & (\partial_{x_i}\Delta_i^k)_Z (\partial_{x_i}\Delta_i^k) \\ (\partial_{x_i}\Delta_i^k)_Z (\partial_{x_i}\Delta_i^k) & (\partial_{x_i}\Delta_i^k)^2 \end{pmatrix} \\
&+ \begin{pmatrix} ((\partial_{y_i}\Delta_i^k)_Z)^2 & (\partial_{y_i}\Delta_i^k)_Z (\partial_{y_i}\Delta_i^k) \\ (\partial_{y_i}\Delta_i^k)_Z (\partial_{y_i}\Delta_i^k) & (\partial_{y_i}\Delta_i^k)^2 \end{pmatrix} .
\end{aligned}
\tag{3.74}
$$

Aus der Gleichung 3.74 wird ersichtlich, dass Vorschriften für das Berechnen von neuen Ausdrücken wie $\partial_{x_i}(I_i(\mathbf{p}_i^k,t))_Z$ und $\partial_{y_i}(I_i(\mathbf{p}_i^k,t))_Z$ benötigt werden. Die verbleibenden Tensoren $T_{\nabla i}^{t,k}$ und $\hat{T}_{\nabla i}^{k}$ verdeutlichen, dass weitere partielle Ableitungen nach u, v und w für die Ausdrücke $\partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1)$ und $\partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1)$ berechnet werden müssen. Glücklicherweise können die Berechnungsvorschriften analog zu der Verfahrensweise in Gleichung 3.45 bestimmt werden:

$$\begin{aligned}
\partial_{x_i}(I_i(\mathbf{p},t))_Z &= \partial_{x_i}(\nabla_i I_i(\mathbf{p}_i^k,t))^\top \cdot \partial_Z \mathbf{p}_i^k \\
&= \partial_{x_i} \begin{pmatrix} \partial_{x_i} I_i(\mathbf{p}_i^k,t) & \partial_{y_i} I_i(\mathbf{p}_i^k,t) \end{pmatrix} \cdot \begin{pmatrix} \partial_Z x_i \\ \partial_Z y_i \end{pmatrix} \\
&= \partial_{x_i x_i} I_i(\mathbf{p}_i^k,t) \cdot \partial_Z x_i + \partial_{x_i y_i} I_i(\mathbf{p}_i^k,t) \cdot \partial_Z y_i \ , \\
\partial_{y_i}(I_i(\mathbf{p},t))_Z &= \partial_{y_i x_i} I_i(\mathbf{p}_i^k,t) \cdot \partial_Z x_i + \partial_{y_i y_i} I_i(\mathbf{p}_i^k,t) \cdot \partial_Z y_i \ .
\end{aligned} \tag{3.75}$$

Es folgen die partiellen Ableitungen nach Z und u bezüglich der Terme $\partial_{x_i} I_i(\hat{\mathbf{p}}_i^k, t+1)$ und $\partial_{y_i} I_i(\hat{\mathbf{p}}_i^k, t+1)$. Die partiellen Ableitungen nach v und w können trivial durch Symmetrieeigenschaften zwischen den Gleichungen ermittelt werden:

$$\begin{aligned}
\partial_{x_i}(I_i(\hat{\mathbf{p}}_i^k,t+1))_Z &= \partial_{x_i x_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_Z \hat{x}_i \\
&\quad + \partial_{x_i y_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_Z \hat{y}_i \ , \\
\partial_{y_i}(I_i(\hat{\mathbf{p}}_i^k,t+1))_Z &= \partial_{y_i x_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_Z \hat{x}_i \\
&\quad + \partial_{y_i y_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_Z \hat{y}_i \ ,
\end{aligned} \tag{3.76}$$

$$\begin{aligned}
\partial_{x_i}(I_i(\hat{\mathbf{p}}_i^k,t+1))_u &= \partial_{x_i x_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_u \hat{x}_i \\
&\quad + \partial_{x_i y_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_u \hat{y}_i \ , \\
\partial_{y_i}(I_i(\hat{\mathbf{p}}_i^k,t+1))_u &= \partial_{y_i x_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_u \hat{x}_i \\
&\quad + \partial_{y_i y_i} I_i(\hat{\mathbf{p}}_i^k,t+1) \cdot \partial_u \hat{y}_i \ .
\end{aligned} \tag{3.77}$$

4. Evaluation

Dieses Kapitel befasst sich mit der Evaluation des in Kapitel 3 beschriebenen Szenenflussverfahrens. Vor der Präsentation der ausgeführten Experimente und deren Ergebnisse werden zuerst die in dieser Evaluation verwendeten Fehlermaße beschrieben. Die Experimente selbst können in drei Stufen unterteilt werden. Die erste Stufe zeigt die grundlegende Funktionalität der Methode anhand von einfachen Datensätzen. Hiernach soll auf der zweiten Stufe anhand von etwas anspruchsvolleren Datensätzen der Einfluss der Gradientenkonstanzannahme und die Erweiterung des Verfahrens durch zusätzliche Ansichten betrachtet werden. Auf der letzten Stufe werden anspruchsvolle realitätsechte Datensätze verwendet, um zu überprüfen ob geeignete Modellparameter bezüglich der Gewichtung der Daten- und Glattheitsterme statisch vorgegeben werden können. Für die automatisierte Optimierung dieser Modellparameter steht seitens der Universität Stuttgart, Abteilung Intelligente Systeme, ein Programm zur Verfügung, das im Rahmen dieser Evaluationen mit dem Downhill Simplex Algorithmus [23] arbeitet.

4.1. Fehlermaße

Fehlermaße haben den Zweck qualitative Aussagen über ein Verfahren zu ermöglichen. Derartige Aussagen ermöglichen zum einen, geeignete Modellparameter für die jeweiligen Datensätze zu finden und zum anderen, die Ergebnisse des Verfahrens mit Ergebnissen, die durch andere Szenenflussverfahren bestimmt worden sind, zu vergleichen. Für die Berechnung von Fehlermaßen stellen Datensätze für gewöhnlich sogenannte Sollwerte (engl. ground truth) bereit. Diese Sollwerte werden in dieser Evaluation als Z^t für die Tiefenwerte und als $\mathbf{V}^t$ für den Szenenfluss bezeichnet. Die durch das Verfahren berechneten Werte werden weiterhin mit Z und $\mathbf{V}$ bezeichnet. Als erstes Fehlermaß wird die durchschnittliche Winkelabweichung (Average Angular Error) des berechneten Szenenflusses ermittelt. Der AAE ist somit ein Maß für

die Richtungsgenauigkeit der berechneten Bewegung. Die Berechnung des Fehlers erfolgt mittels der folgenden Gleichung:

$$\mathrm{AAE}_{\mathrm{3D}} = \frac{1}{|\Omega_0|} \sum_{(i,j)\in\Omega_0} \arccos\left(\frac{\mathbf{V}^t_{i,j} \cdot \mathbf{V}_{i,j}}{|\mathbf{V}^t_{i,j}| * |\mathbf{V}_{i,j}|}\right) . \tag{4.1}$$

Das zweite in dieser Evaluation verwendete Fehlermaß, das normalisierte quadratische Mittel (NRMS), berechnet die durchschnittliche Abweichung der berechneten 3D-Punkte von ihrer Soll-Position. Da in dieser Arbeit die Position der 3D-Punkte mittels der Kameraparameter als Funktion der Tiefe verstanden wird, gibt dieses Fehlermaß Auskunft über die Qualität der berechneten Tiefenwerte. Der $\mathrm{NRMS}_{\mathbf{P}}$-Fehler wird wie folgt berechnet:

$$\mathrm{NRMS}_{\mathbf{P}} = \frac{\sqrt{\frac{1}{|\Omega_0|}\sum_{(i,j)\in\Omega} |\mathbf{P}_{i,j} - \mathbf{P}^t_{i,j}|^2}}{\max(|\mathbf{P}^t|) - \min(|\mathbf{P}^t|)} . \tag{4.2}$$

Der NRMS-Fehler wird weiter für die Abweichung der berechneten Bewegung $\mathbf{V}$ verwendet und dient somit als das dritte verwendete Fehlermaß. Im Gegensatz zu dem zuvor eingeführten $\mathrm{AAE}_{\mathrm{3D}}$ Fehlermaß, welches nur den Winkel berücksichtigt, gibt der NRMS-Fehler Aufschluss über Abweichungen in den berechneten Werten für u, v und w. Dies ist notwendig, da Vektoren trotz gleicher Orientierung durch unterschiedliche Werte aufgebaut sein können. Die Berechnung dieses Fehler ist gegeben durch

$$\mathrm{NRMS}_{\mathbf{V}} = \frac{\sqrt{\frac{1}{|\Omega_0|}\sum_{(i,j)\in\Omega} |\mathbf{V}_{i,j} - \mathbf{V}^t_{i,j}|^2}}{\max(|\mathbf{V}^t|) - \min(|\mathbf{V}^t|)} . \tag{4.3}$$

Als letztes wird der sogenannte Endpunktfehler (EPE) [13] der Ergebnisse ermittelt. Der EPE berechnet die positionelle Abweichung eines bewegten 2D-Bildpunkts $\hat{\mathbf{p}}_{0,ij}$ von seiner Sollposition $\hat{\mathbf{p}}^t_{0,ij}$ zum Zeitpunkt $t+1$. Hierfür wird die Position des Bildpunkts mittels der Gleichung 3.11 und den berechneten Werten Z und $\mathbf{V}$ auf der eigenen Bildebene ermittelt und mit der Position, die sich durch die Sollwerte Z^t und $\mathbf{V}^t$ ergeben, verglichen. An jedem Bildpunkt wird der EPE mittels der Gleichung

$$\mathrm{EPE}_{ij} = |\hat{\mathbf{p}}_{0,ij}(Z_{ij}, \mathbf{V}_{ij}) - \hat{\mathbf{p}}^t_{0,ij}(Z^t_{ij}, \mathbf{V}^t_{ij})| \tag{4.4}$$

berechnet. Im Falle der *KITTI* Datensätze [21] wird neben der Disparität der optische Fluss u^t_{of}, v^t_{of} zur Überprüfung der Ergebnisse

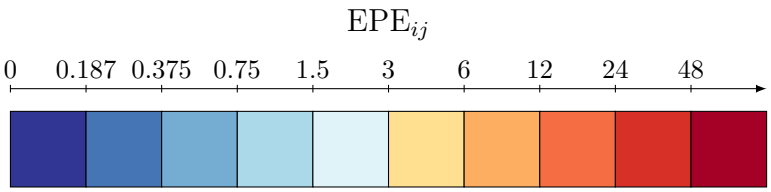


Abbildung 4.1.: Farbkodierung des Fehlerbildes. Abweichungen von $T_{\text{EPE}} \geq 3$ werden als ein fehlerhafter Bildpunkt betrachtet.

bereitgestellt. Die Berechnung des Endpunktfehlers ändert sich damit wie folgt:

$$\begin{aligned} \text{EPE}_{ij} &= |\hat{\mathbf{p}}_{ij} - \mathbf{p}^t_{of,ij}| \\ &= \sqrt{(\hat{x}_{ij} - (i + u^t_{of,ij}))^2 + (\hat{y}_{ij} - (j + v^t_{of,ij}))^2} \, . \end{aligned} \tag{4.5}$$

Das EPE Fehlermaß wird in einem Fehlerbild visualisiert, dessen Farbkodierung in Abbildung 4.1 dargestellt ist. Ist der berechnete Fehlerwert größer als der in dieser Arbeit festgelegte Grenzwert von $T_{\text{EPE}} = 3$, wird von einem fehlerhaften Bildpunkt (engl. **b**ad **p**ixel) gesprochen. Um eine qualitative Aussage über den EPE treffen zu können, wird jedes Testergebnis zusätzlich zu den zuvor eingeführten drei Fehlerwerten einen BP-Wert beinhalten, der die prozentuale Anzahl solcher fehlerhaften Bildpunkten angibt.

4.2. Rahmenbedingungen

Einige der verwendeten Rahmenbedingungen sind für alle der folgenden Versuche identisch und werden aus Gründen der Reproduzierbarkeit hier aufgeführt:

- **Bildrepräsentation**. Für die interne Bildrepräsentation werden Farbbilder durch Bildung des arithmetischen Mittels der RGB-Farbkanäle zu Grauwertbildern konvertiert.
- **Lösen des linearen Gleichungssystems**. Auf jeder Skalierungsstufe wird das sogennante SOR-Verfahren [34] für das Lösen des linearisierten Gleichungssystems aus 3.38 - 3.41 verwendet. Es werden 10 SOR-Iterationen mit einem Relaxationsparameter von $\omega = 1.95$ berechnet.
- **Verzögerte Nichtlinearitätsmethode**. Es werden $l_{\max} = 5$ äußere Iterationsschritte der verzögerten Nichtlinearitätsmethode berechnet.
- **Parameter der Grob-zu-Fein-Strategie**. Der Skalierungsfaktor wird zu $\eta = 0.95$ gewählt und es existiert eine Obergrenze für die maximale Anzahl an möglichen Skalierungsstufen mit $k_{\max} = 200$.

Die damit verbleibenden Parameter für die Gewichtung der einzelnen Terme des Energiefunktionals werden als Modellparameter bezeichnet und haben wesentlichen Einfluss auf die Qualität der Ergebnisse. Diese werden, soweit nicht anders erwähnt, automatisiert durch das verwendete Optimierungsprogramm bestimmt.

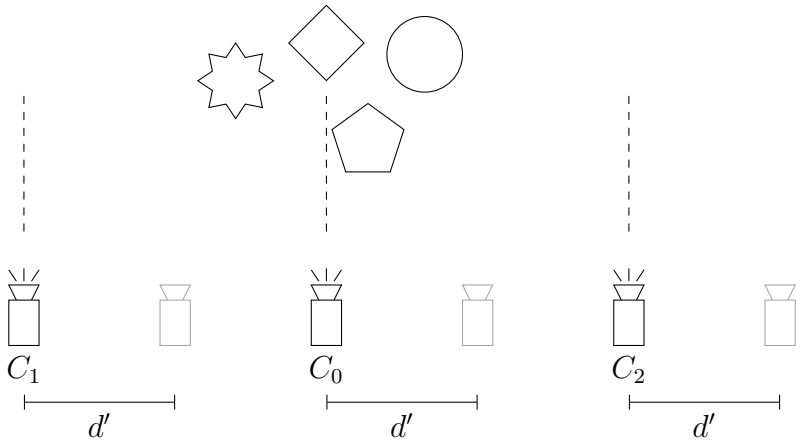
4.3. Homogener Szenenfluss

In einem ersten Versuch wird die grundlegende Funktionalität des Verfahrens überprüft. Hierfür werden die beiden Datensätze *Teddy* und *Cones* aus den Middlebury Stereodatensätzen [26] verwendet. Die Datensätze bestehen jeweils aus 9 rektifizierten Farbbildern mit entsprechenden Sollwerten für die Disparität zwischen den Bildern *im2* und *im6*. Da es sich hierbei um Stereodatensätze handelt, ist die aufgenomme Szene statisch und es existieren keine Sollwerte für den Szenenfluss. Weiter werden keine Kameraparamter, die für das vorgestellte Szenenflussverfahren notwendig sind, zur Verfügung gestellt. Um die Verwendung dieser Datensätze trotzdem zu ermöglichen, wird die Lösung dieser Probleme in der folgenden Aufzählung beschrieben:

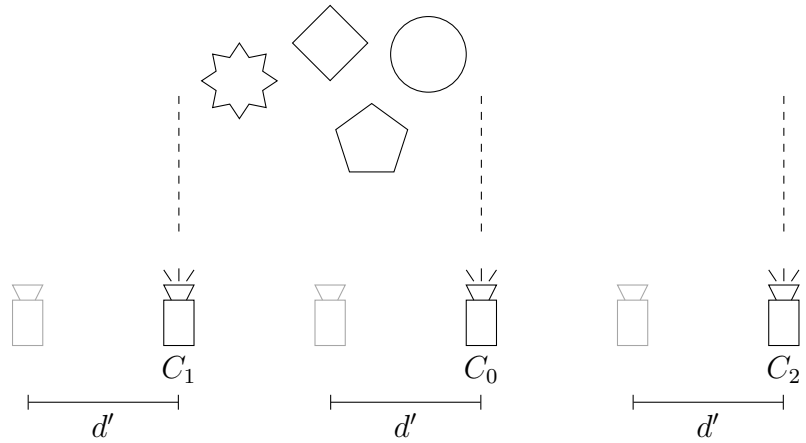
- Die extrinsischen Kameraparameter werden wie folgt bestimmt: die Referenzkamera (*im2*) liegt im Ursprung des Weltkoordinatensystems. Mit dem Wissen, dass die Bilder zu jeweils gleichen Abständen auf der X-Achse des Referenzkamerakoordinatensystems zueinander aufgenommen worden sind [26], wird eine beliebige Distanz, die damit zwischen allen Kameras gleich ist, festgelegt. Der Translationsvektor lässt sich anschließend für jede Kamera trivial bestimmen. Da die Bilder rektifiziert worden sind, existiert keine Rotation zwischen den einzelnen Kamerakoordinatensystemen.
- Für die intrinsischen Kameraparameter o_x und o_y werden die Koordinaten der Bildmitte verwendet. Die Skalierungsfaktoren s_x und s_y können jeweils durch einen beliebigen Wert, der größer Null ist, initialisiert werden. Weiter wird die Brennweite f auf den größeren der beiden Werte s_x und s_y gesetzt. Nun kann mittels der mitgelieferten Disparitätswerte d, den bekannten extrinsischen Parametern und der Formel [33]

$$Z = \frac{f \cdot B}{d} \tag{4.6}$$

  die Tiefe Z an jedem Bildpunkt der Referenzkamera bestimmt werden. Der Parameter B definiert dabei den Abstand der beiden Kameras, für die die Disparität mitgeliefert wurde (hier *im2* und *im6*). Die nun berechnete Tiefe stimmt bis auf einen Skalierungsfaktor mit den in den Bildern tatsächlich auftretenden Werten überein und ist innerhalb des Versuchsaufbaus und dessen Geometrie konsistent. Da das Szenenflussverfahren sich der



(a) Situation zum Zeitpunkt t.



(b) Situation zum Zeitpunkt $t + 1$.

Abbildung 4.2.: Exemplarischer Versuchsaufbau der Stereodatensätze. Die Verwendung unterschiedlicher Ansichten zum Zeitpunkt $t + 1$ simuliert eine homogene Bewegung der Szene nach links um d'.

gleichen Parameter bedient, sollte das Verfahren sich den nun berechneten Sollwerten für die Tiefe annähern.

- Als letztes bleibt zu klären, wie eine Bewegung in einer statischen Szene erzeugt werden kann, welche dann wiederum von dem Szenenflussverfahren berechenbar ist. Hierfür wird jeweils für die Zeitpunkte t und $t + 1$ ein Satz Bilder bestimmt, die untereinander eine homogene Verschiebung der Szenen aufzeigen. Der so entstandene Szenenfluss zwischen den Bildern ist mit dem zuvor festgelegten Abstand zwischen den Kameras berechenbar.

Ein möglicher Versuchsaufbau, der diese Vorgehensweise verdeutlicht, ist in Abbildung 4.2 dargestellt. Aufgrund des Aufbaus der Stereodatensätze kann somit nur jeweils eine homogene Bewegung entlang der X-Achse des Referenzkamerakoordinatensystems simuliert werden. Trotzdem werden die Middlebury Stereo Datensätze von einer Vielzahl an Szenenflussverfahren und deren Evaluation verwendet [33].

Die folgenden zwei Seiten beinhalten die visualisierten Ergebnisse des in dieser Arbeit vorgestellten Szenenflussverfahrens für die, wie oben beschrieben, angepassten Datensätze *Teddy* und *Cones*. Es wird jeweils nach dem bestmöglichen $\mathrm{AAE}_{\mathrm{3D}}$ Fehlerwert optimiert. Weiter werden 4 Kameraansichten zu jeweils 2 Zeitpunkten verwendet. Die Gradientenkonstanzannahme wird für beide Datensätze abgeschaltet ($\gamma = 0$).

Die entsprechenden Fehlerwerte für die abgebildeteten Ergebnisse des *Teddy* und *Cones* Datensatzes sind in der Tabelle 4.1 aufgeführt. Der *Teddy* Datensatz erzielt bessere Ergebnisse für $\mathrm{AAE}_{\mathrm{3D}}$ und BP, wohingegen der *Cones* Datensatz bessere Werte für $\mathrm{NRMS}_{\mathbf{P}}$ und $\mathrm{NRMS}_{\mathbf{V}}$ erzielt.

Datensatz	$\mathrm{NRMS}_{\mathbf{P}}$ [%]	$\mathrm{NRMS}_{\mathbf{V}}$ [%]	$\mathrm{AAE}_{\mathrm{3D}}$	BP [%]
Teddy	7.78	7.78	0.304	4.92
Cones	1.66	1.66	0.747	8.24

Tabelle 4.1.: Fehlermetriken für die Datensätze *Teddy* und *Cones* bei Optimierung nach $\mathrm{AAE}_{\mathrm{3D}}$.

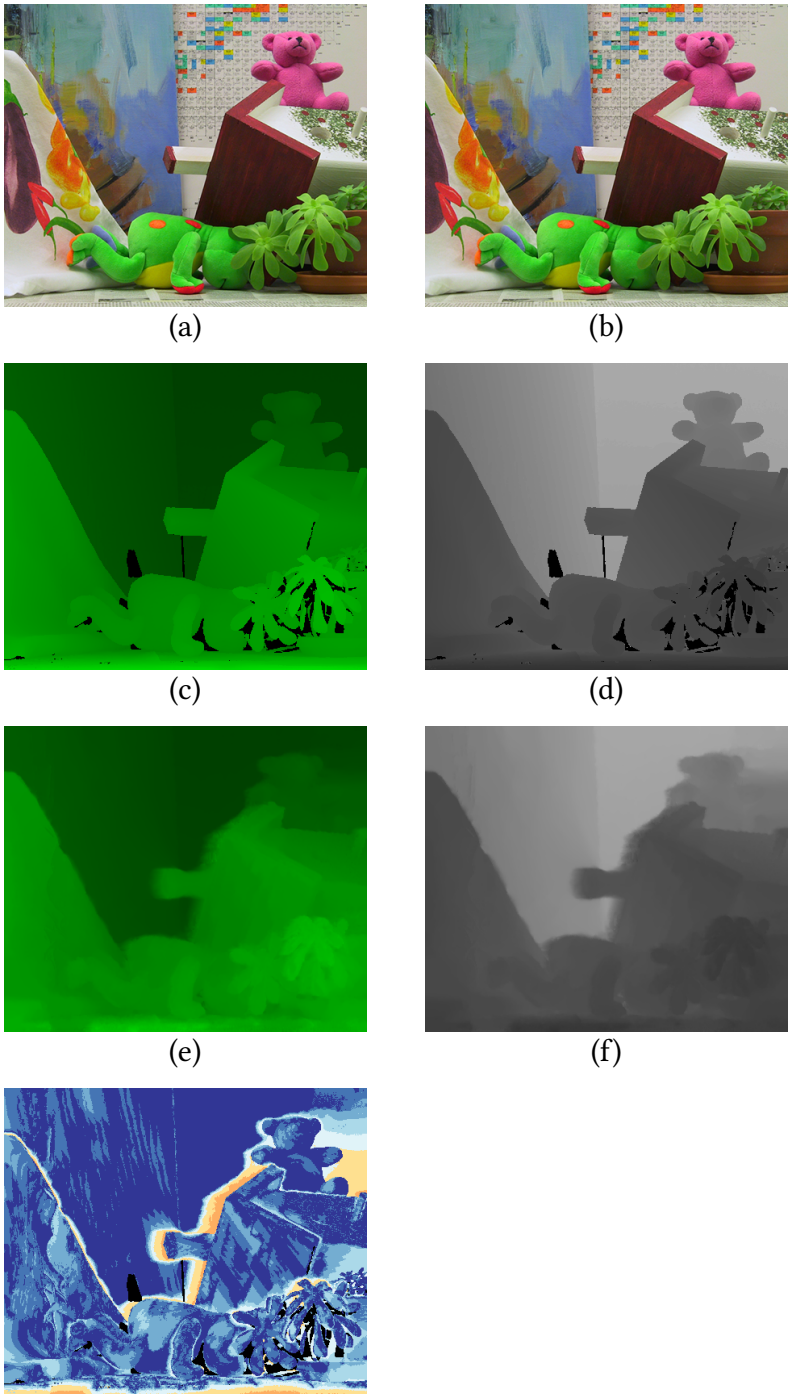


Abbildung 4.3.: Ergebnisse für den *Teddy* Datensatz. (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte Szenenfluss. (d) Sollwerte Tiefe. (e) Durch die Methode berechneter Szenenfluss. (f) Berechnete Tiefe. (g) Fehlerbild.

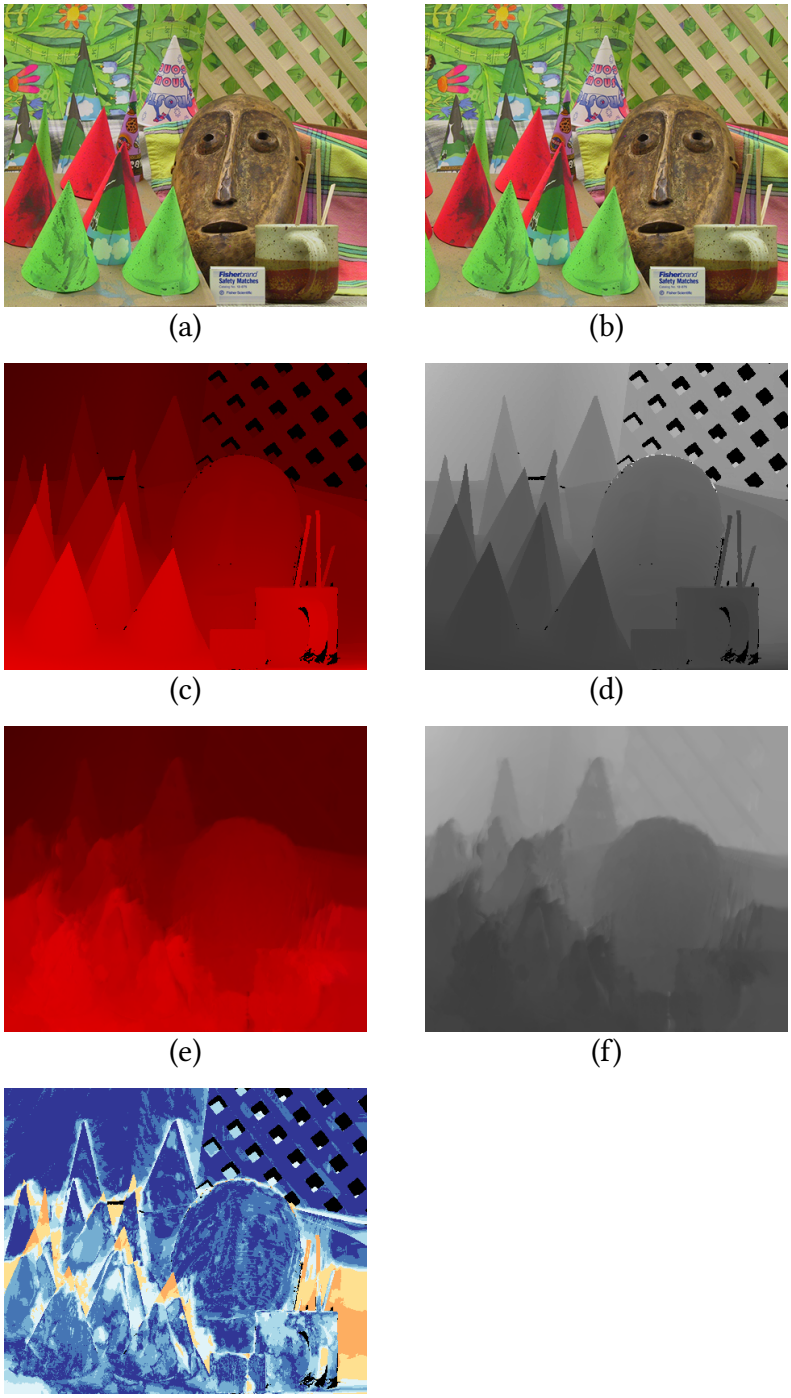


Abbildung 4.4.: Ergebnisse für den *Cones* Datensatz. (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte Szenenfluss. (d) Sollwerte Tiefe. (e) Durch die Methode berechneter Szenenfluss. (f) Berechnete Tiefe. (g) Fehlerbild.

4.4. Gradientenkonstanzannahme

In einem weiteren Versuch soll der Einfluss der Gradientenkonstanzannahme untersucht werden. Zu diesem Zweck wird der Middlebury Stereodatensatz *Art* [25], der Bilder mit unterschiedlichen Beleuchtungsstärken beinhaltet, verwendet. Der Datensatz ist in Abbildung 4.5 dargestellt. Der Stereodatensatz wird, wie zuvor in Abschnitt 4.3 beschrieben, auf die Benutzung in einem Szenenflussverfahren angepasst. Weiter wird für die Bilder des Zeitpunktes $t + 1$ eine höhere Beleuchtungsstärke, die zu einer Verletzung der Helligkeitswertkonstanzannahme führt, verwendet. Durch den Optimierer der Universität Stuttgart, Abteilung Intelligente Systeme, werden zuerst geeignete Gewichte für die Helligkeitswertkonstanzannahme und die Glattheitsterme ermittelt. Anschließend wird die Gewichtung der Gradientenkonstanzannahme γ manuell erhöht. Zudem wird der Optimierer verwendet, um für die zuvor berechneten Gewichte den bestmöglichen γ-Wert zu ermitteln. Hiermit soll der Einfluss der Gradientenkonstanzannahme auf die Testergebnisse sichtbar gemacht werden. Es werden wieder 4 Kameraansichten zu jeweils 2 Zeitpunkten verwendet. Die Ergebnisse dieses Versuchs sind in Abbildung 4.6 visualisiert.

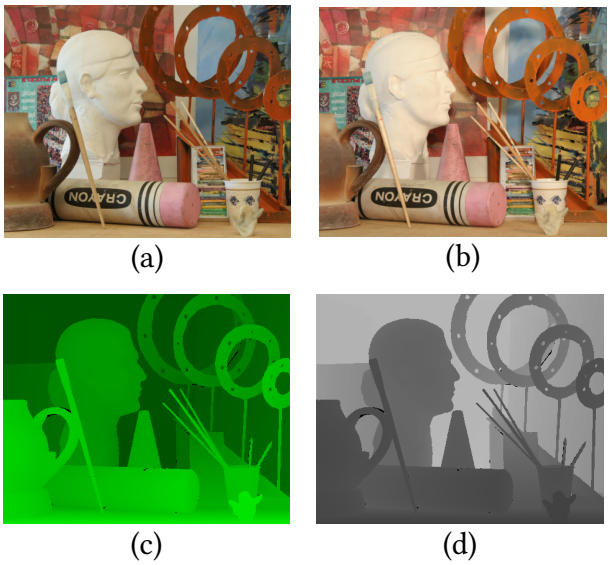
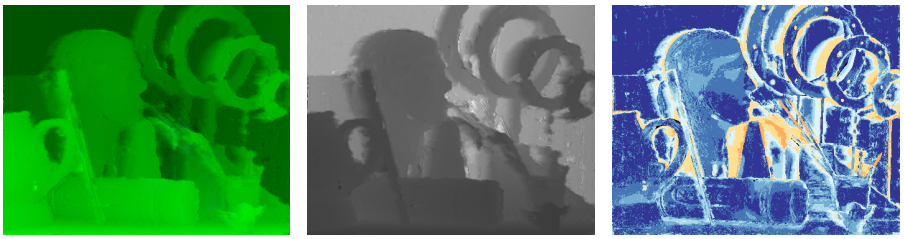


Abbildung 4.5.: *Art Datensatz.* (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte Szenenfluss. (d) Sollwerte Tiefe.

(a)

(b)

(c)



(d)

Abbildung 4.6.: *Art* Datensatz: Einfluss der Gradientenkonstanzannahme. **Reihe** (a) Berechnete Szenenfluss- und Tiefenwerte sowie Fehlerbild ohne Gradientenkonstanzannahme $\gamma = 0$. **Reihe** (b) Ergebnisse für $\gamma = 2$. **Reihe** (c) Ergebnisse für $\gamma = 5$. **Reihe** (d) Ergebnisse für $\gamma \approx 19.11$.

Als letzter Test für den Einfluss der Gradientenkonstanzannahme wird unter Anwendung des Optimierers jeweils das bestmögliche Ergebnis bezüglich der Fehler $\mathrm{AAE}_{\mathrm{3D}}$ und BP ermittelt. Die Ergebnisse sind in Abbildung 4.7 dargestellt.

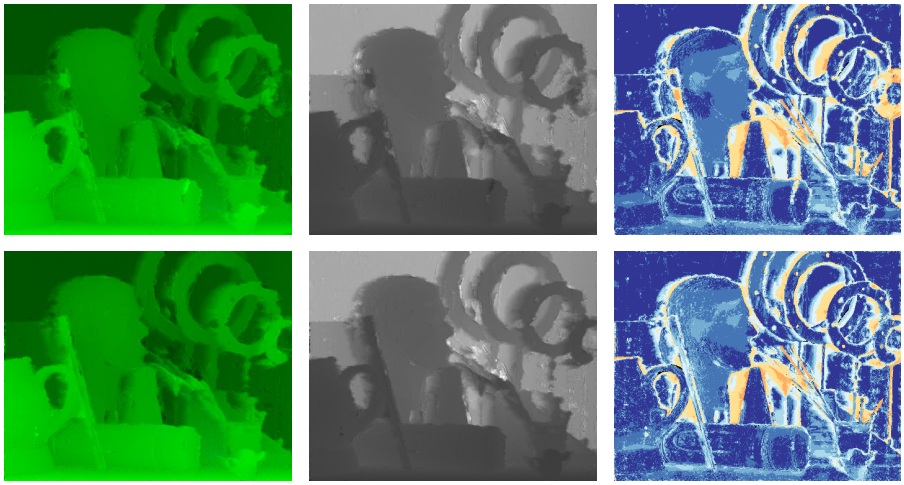


Abbildung 4.7.: *Art* Datensatz - Optimierung mit Gradientenkonstanzannahme. **Reihe oben** Optimierung nach $\mathrm{AAE}_{\mathrm{3D}}$. **Reihe unten** Optimierung nach BP.

Die Fehlerwerte der in diesem Abschnitt beschriebenen Testläufe sind in der folgenden Tabelle kompakt zusammengefasst. Es ist eindeutig erkennbar, dass die Gradientenkonstanzannahme die Ergebnisse bei Helligkeitsunterschieden zwischen den Bildern verbessern kann.

Art Datensatz	$\mathrm{NRMS}_{\mathbf{P}}$ [%]	$\mathrm{NRMS}_{\mathbf{V}}$ [%]	$\mathrm{AAE}_{\mathrm{3D}}$	BP [%]
$\gamma = 0$	14.28	14.28	19.003	**11.36**
$\gamma = 2$	13.81	13.77	11.392	10.37
$\gamma = 5$	13.52	13.44	5.776	9.82
$\gamma \approx 19.11$	13.21	13.07	**1.222**	8.60
Opt. $\mathrm{AAE}_{\mathrm{3D}}$	13.75	13.61	**0.754**	9.74
Opt. BP	13.39	13.26	1.116	**8.45**

Tabelle 4.2.: Zusammenfassung der Fehlermetriken bei den Versuchen in Abbildung 4.6 und 4.7. Nach den **fett** hervorgehoben Fehlerwerten wurde automatisiert optimiert. Die Werte $\gamma = 2$ und $\gamma = 5$ wurden für Testzwecke manuell festgelegt.

4.5. Zusätzliche Kameraansichten

Im einfachsten Fall stehen den Szenenflussverfahren jeweils 2 Bilder zu zwei unterschiedlichen Zeitpunkten zur Verfügung. Durch die verwendete 3D-Parametrisierung in Form einer Punktewolke ist es möglich weitere Ansichten zu verwenden. Deshalb wird im nächsten Experiment der Einfluss von zusätzlichen Kameraperspektiven ermittelt. Hierfür wird der Datensatz *Carpet Ball* [9], der über 5 verschiedene Ansichten zu jeweils 2 Zeitpunkten verfügt, verwendet. Die Szene zeigt einen rotierenden und stark texturierten Ball vor einem ebenfalls rotiertenden und gleich texturierten Teppich. Der Datensatz enthält zudem fertige Kamerakalibrierungsmatrizen für alle Ansichten und Sollwertdaten für die Größen Z und $\mathbf{V}$. Es wird sukzessive mit 2, 3, 4 und schließlich 5 Ansichten des Datensatzes getestet. Weiter wird nach dem $\mathrm{AAE}_{\mathrm{3D}}$ Fehler optimiert und es wird **keine** Gradientenkonstanzannahme verwendet ($\gamma = 0$). Das Referenzbild und die Sollwerte des Datensatz sind in Abbildung 4.8 dargestellt. Die berechneten Ergebnisse sind dagegen in Abbildung 4.9 visualisiert und die entsprechenden Fehlerwerte können in Tabelle 4.3 eingesehen werden. Die Verwendung zusätzlicherer Kameraansichten hat die Ergebnisse teilweise deutlich verbessert. Bei der Verwendung von 5 Ansichten stagnierten die Fehlerwerte für $\mathrm{NRMS}_{\mathbf{P}}$ und $\mathrm{NRMS}_{\mathbf{V}}$, doch $\mathrm{AAE}_{\mathrm{3D}}$ und der BP-Wert konnten noch einmal verbessert werden.

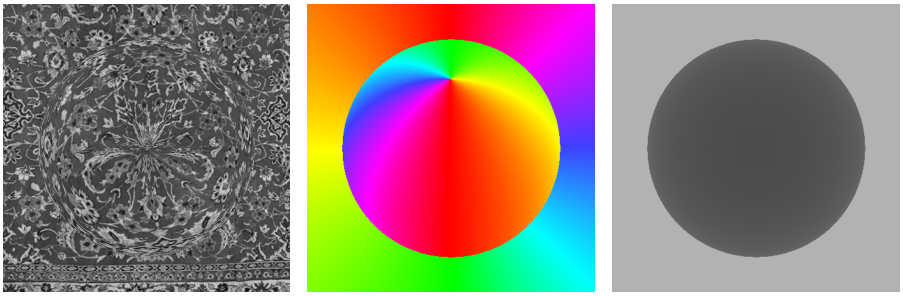


Abbildung 4.8.: *Carpet Ball* Datensatz. **Links** Referenzbild zum Zeitpunkt t. **Mitte** Sollwerte für den Szenenfluss. **Rechts** Sollwerte für die Tiefe.

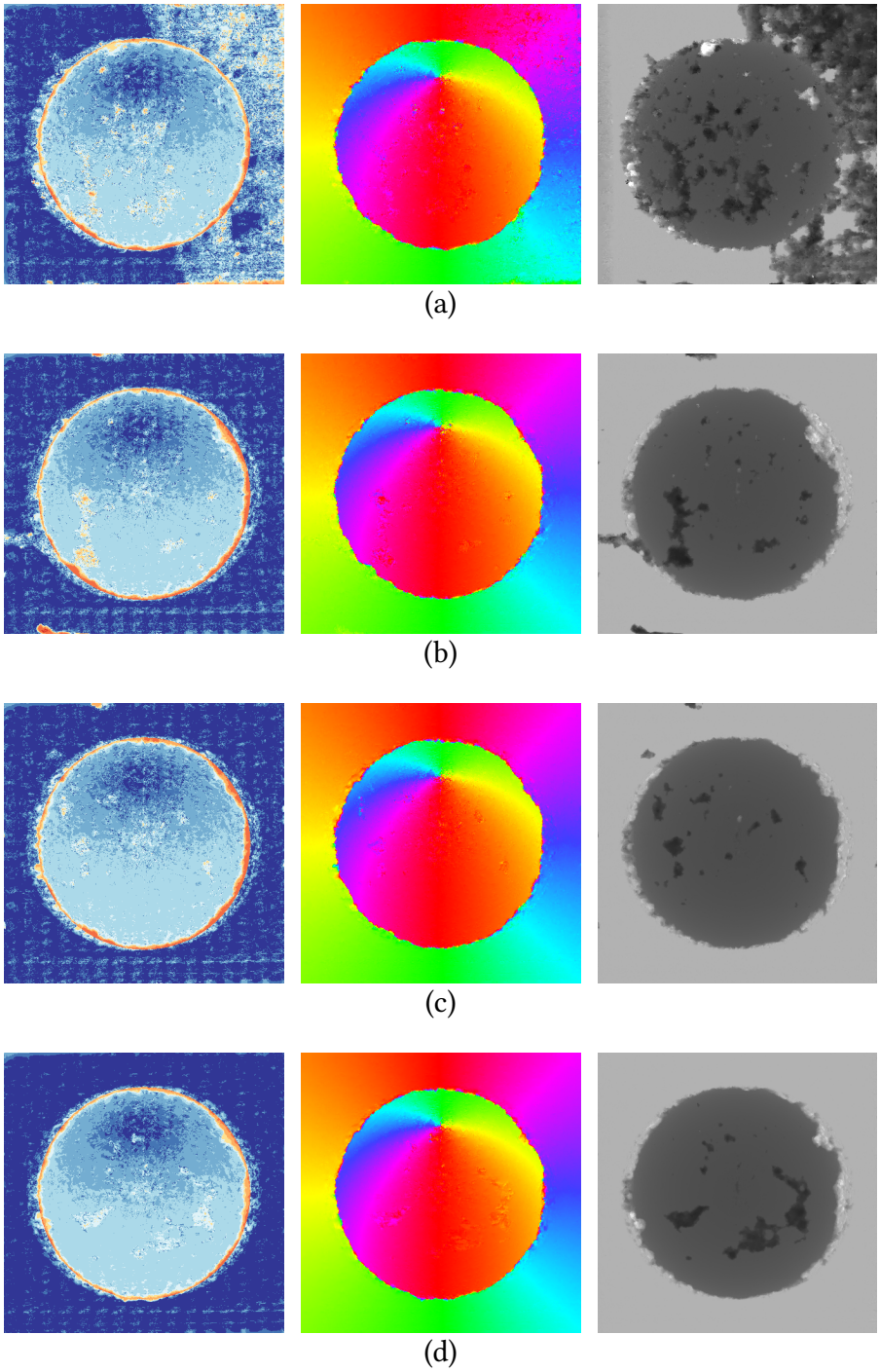


Abbildung 4.9.: Ergebnisse *Carpet Ball* Datensatz. Die Anzahl der verwendeten Kameraansichten variiert in jeder Reihe. **Reihe** (a) 2 Ansichten. **Reihe** (b) 3 Ansichten. **Reihe** (c) 4 Ansichten. **Reihe** (d) 5 Ansichten. Die zugehörigen Fehlermaße sind in der Tabelle 4.3 aufgelistet.

Carpet Ball	$\mathrm{NRMS}_{\mathbf{P}}$ [%]	$\mathrm{NRMS}_{\mathbf{V}}$ [%]	AAE_{3D}	BP %]
(a) 2 Ansichten	37.10	37.10	7.545	5.35
(b) 3 Ansichten	10.90	10.90	6.566	4.11
(c) 4 Ansichten	7.20	7.20	6.422	3.55
(d) 5 Ansichten	7.50	7.50	5.605	3.01

Tabelle 4.3.: Ergebnisse der *Carpet Ball* Testszenarien. Die Verwendung von zusätzlichen Ansichten verbessert die Qualität der Ergebnisse.

4.6. Modellparameter Kalibrierung

Um bestmögliche Ergebnisse zu erhalten, wurden in den bisherigen Versuchen die Modellparameter des Szenenflussverfahrens durch einen Optimierer nahezu perfekt an die jeweilige Szene automatisiert angepasst. In einem letzten Versuch wird deshalb untersucht, ob mithilfe von Trainingsdatensätzen die Modellparameter so kalibriert werden können, dass diese für andere Szenen verwendbar sind. Für die Kalibrierung werden 10 einzelne Szenen gemeinsam, in einer sogenannten Bündeloptimierung, optimiert. In einem weiteren Schritt werden dann die gefundenen Modellparametern an 3 weiteren Szenen getestet. Hierfür wird die anspruchsvolle *KITTI Vision Benchmark Suite* [12, 13], die aus insgesamt 200 verwendbaren realitätsechten Szenen mit vollständig kalibriertem Kameraaufbau und entsprechenden Sollwertdaten besteht, verwendet. Jede Szene wurde von einem fahrenden Auto aus aufgenommen und besteht aus 2 Kameraansichten zu jeweils 2 Zeitpunkten. Die aufgenommen Szenen enthalten viele Herausforderungen für das Szenenflussverfahren. Einerseits verletzen spiegelnde Lackoberflächen und Fensterscheiben die Helligkeitswertkonstanzannahme, andererseits gestaltet sich die Zuordnung von Korrespondenzen auf texturarmen Oberflächen wie z.B. Straßen und Hauswänden als schwierig. Da zudem in realtiätsechten Szenen Schwankungen in der Beleuchtungsstärke zu erwarten sind, wird für die Testszenarien die Gradientenkonstanzannahmen aktiviert ($\gamma > 0$). Weiter wird die *KITTI* spezifische Tiefenvisualisierung, die besser geeignet für Szenen mit weit entfernten Objekten ist, verwendet. Die eingesetzten Datensätze und die Ergebnisse der Bündeloptimierung sind in der Tabelle 4.4 aufgelistet. Ein Vergleich zwischen den Fehlerwerten der Bündeloptimierung und den Fehlerwerten der jeweiligen Einzeloptimierung eines Datensatzes kann im Anhang in der Tabelle C.1 gefunden werden.

KITTI Trainingsdatensatz	$NRMS_{\mathbf{P}}$ [%]	$NRMS_{\mathbf{V}}$ [%]	AAE_{3D}	BP [%]
000002	10.00	10.00	12.349	15.26
000006	6.50	6.50	11.887	49.17
000011	26.14	26.02	9.858	13.40
000022	16.77	16.20	18.358	25.18
000069	6.46	6.53	1.204	35.80
000133	4.44	4.21	8.0717	55.78
000151	5.89	5.85	13.681	7.54
000152	4.76	4.75	7.948	5.12
000154	15.58	15.49	9.753	30.02
000165	9.21	9.18	13.33	11.03

Tabelle 4.4.: Ergebnisse der Bündeloptimierung. Die 10 aufgelisteten Datensätze wurden gemeinsam optimiert. Die hierdurch berechneten Modellparameter werden an weiteren Datensätzen 000054, 000085 und 000168 getestet.

Die durch die Bündeloptimierung gefundenen Modellparameter werden an den *KITTI* Trainingsdatensätzen 000054, 000085 sowie 000168 getestet und mit den jeweiligen Einzeloptimierungen verglichen. Die Ergebnisse sind in Tabelle 4.5 festgehalten. Die entsprechenden Referenzbilder der Datensätze sowie Visualisierungen der Sollwerte und der Ergebnisse sind in den Abbildungen 4.10 - 4.15 auf den folgenden Seiten dargestellt.

KITTI Trainingsdatensatz	$NRMS_{\mathbf{P}}$ [%]	$NRMS_{\mathbf{V}}$ [%]	AAE_{3D}	BP [%]
000054 (B)	3.92	3.94	1.252	14.68
000054 (E)	3.59	3.61	1.145	8.76
000085 (B)	5.74	5.67	7.473	28.20
000085 (E)	6.49	6.42	7.273	20.91
000168 (B)	6.78	6.71	3.005	8.11
000168 (E)	3.43	3.43	1.531	5.13

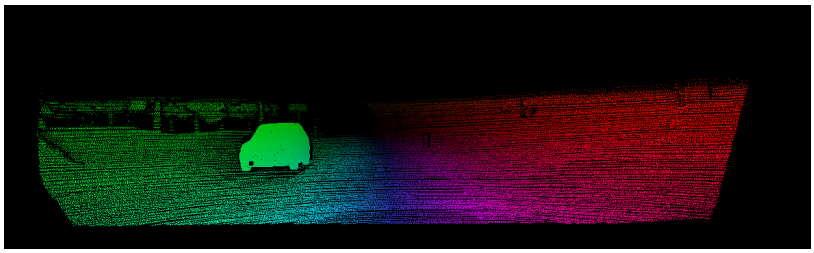
Tabelle 4.5.: Vergleich der Fehlerwerte zwischen den Modellparametern, die durch die Bündeloptimierung (B) ermittelt worden sind und der jeweiligen Einzeloptimierung (E) eines Datensatzes.

Aus der Betrachtung der Ergebnisse wird klar, dass durch die vorab kalibrierten Modellparameter mit Qualitätseinbußen gerechnet werden muss. Dies schlägt sich auch sichtbar in den Visualisierungen der berechneten Werte nieder. Trotzdem werden durch das vorgestellte Szenenflussverfahren größtenteils sinnvolle Werte berechnet, so dass eine Vorabkalibrierung auch für herausforderende Datensätze als möglich erscheint.

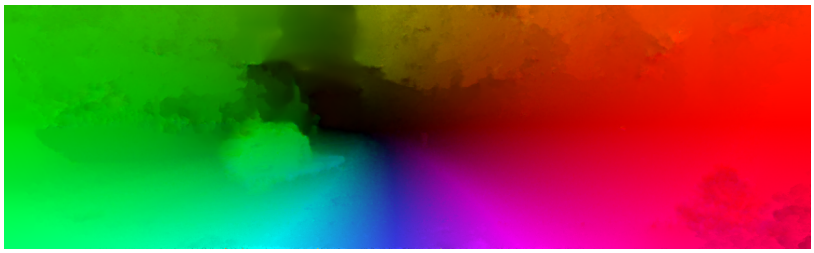
(a)

(b)

(c)



(d)



(e)

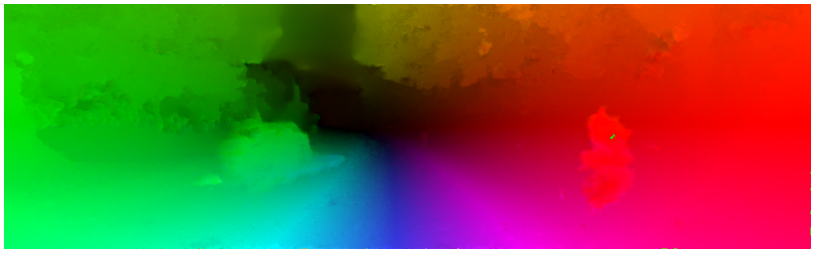
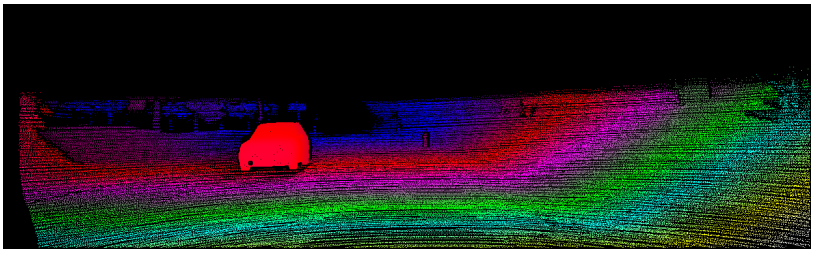
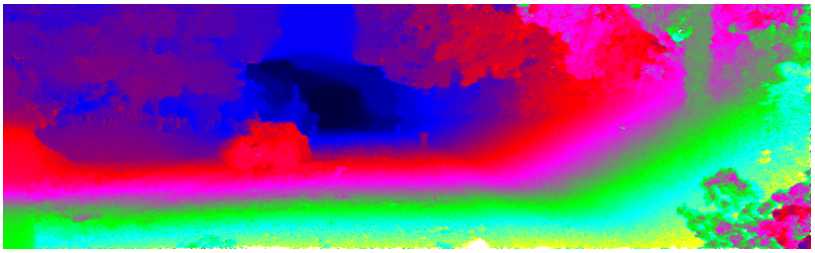


Abbildung 4.10.: *KITTI* Datensatz 000054. (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte des Szenenflusses. (d) Durch Einzeloptimierung berechneter Szenenfluss. (e) Berechneter Szenenfluss bei Verwendung der vorab kalibrierten Modellparameter.

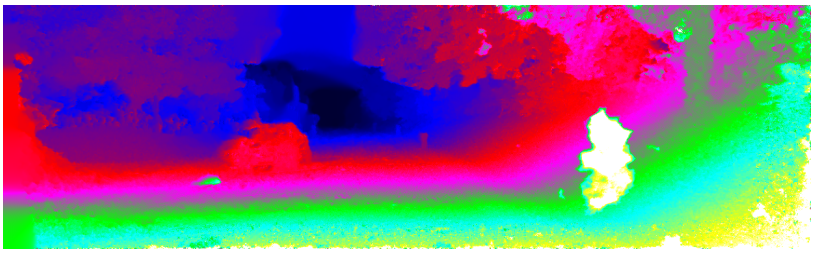
(a)



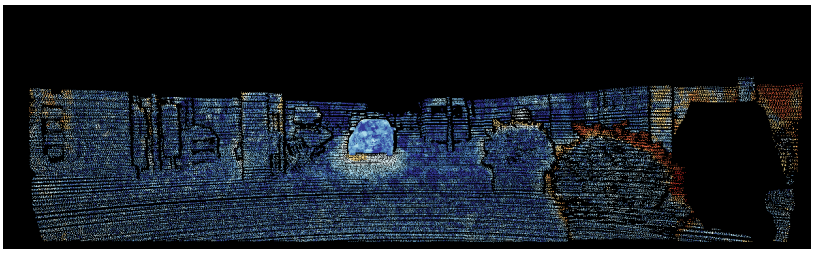
(b)



(c)



(d)



(e)

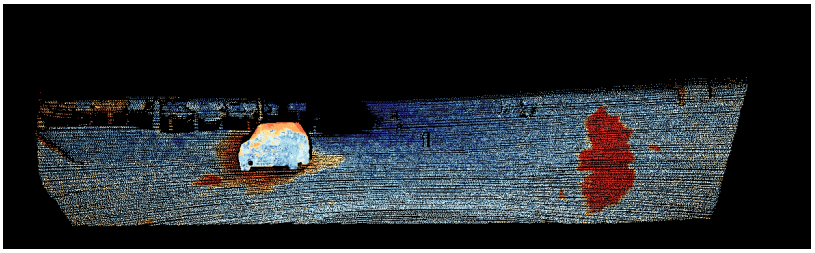
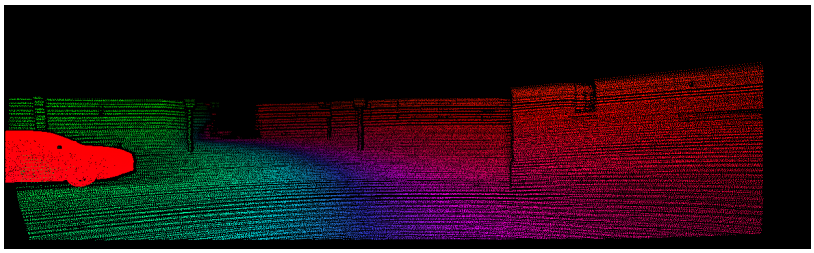


Abbildung 4.11.: *KITTI* Datensatz 000054. (a) Sollwerte der Tiefe. (b) Durch Einzeloptimierung berechnete Tiefenwerte. (c) Berechnete Tiefenwerte bei Verwendung der vorab kalibrierten Modellparameter. (d) Fehlerbild der Einzeloptimierung. (e) Fehlerbild bei Verwendung der vorab kalibrierten Modellparameter.

(a)

(b)

(c)



(d)

(e)

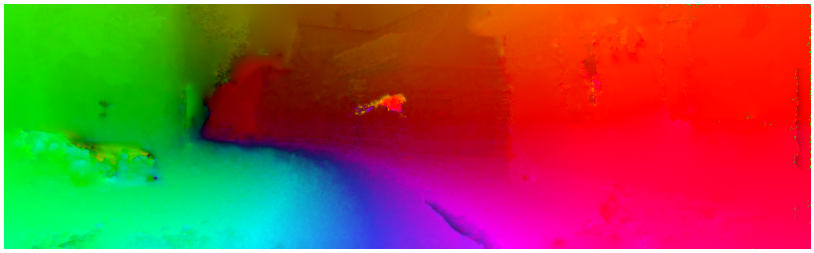


Abbildung 4.12.: *KITTI* Datensatz 000085 (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte des Szenenflusses. (d) Durch Einzeloptimierung berechneter Szenenfluss. (e) Berechneter Szenenfluss bei Verwendung der vorab kalibrierten Modellparameter.

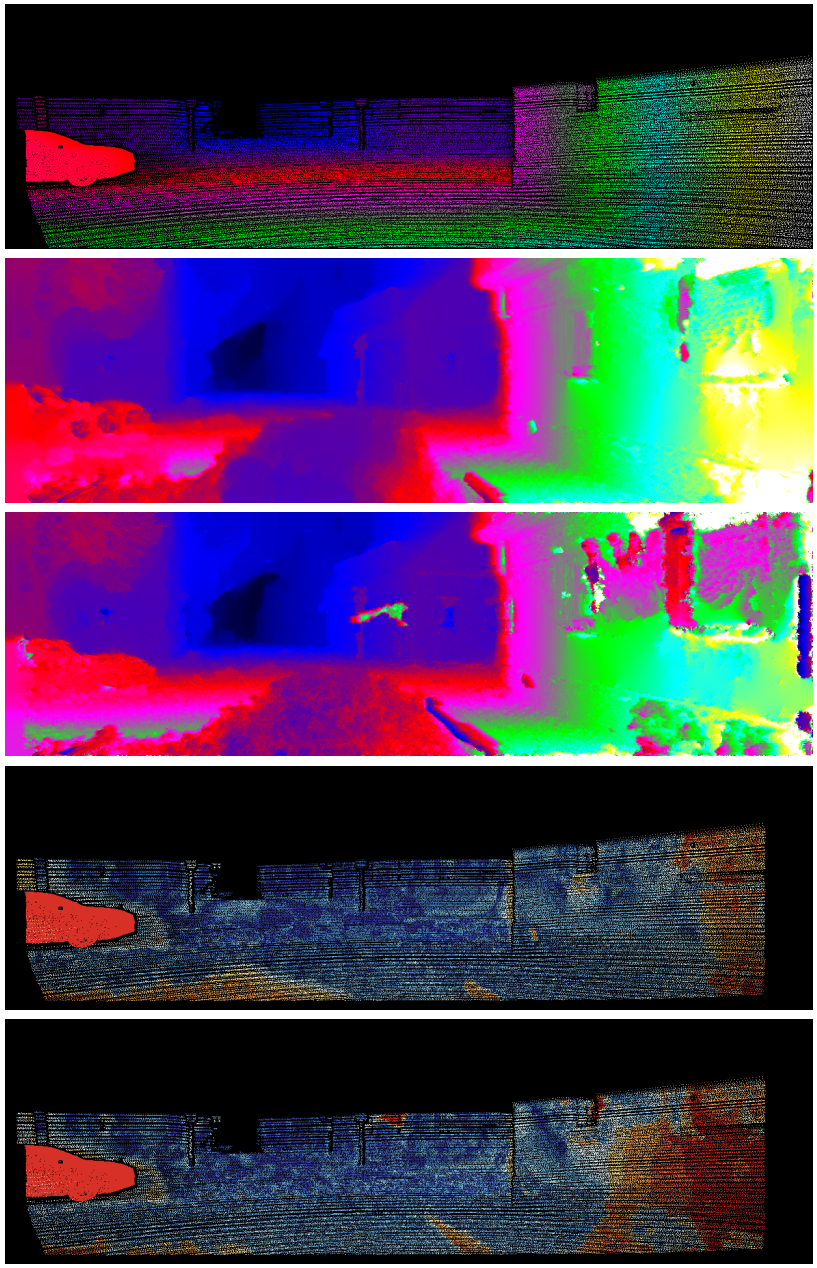
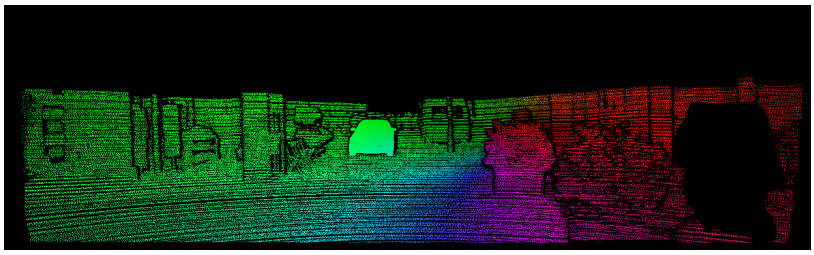


Abbildung 4.13.: *KITTI* Datensatz 000085. (a) Sollwerte der Tiefe. (b) Durch Einzeloptimierung berechnete Tiefenwerte. (c) Berechnete Tiefenwerte bei Verwendung der vorab kalibrierten Modellparameter. (d) Fehlerbild der Einzeloptimierung. (e) Fehlerbild bei Verwendung der vorab kalibrierten Modellparameter.

(a)

(b)

(c)



(d)

(e)

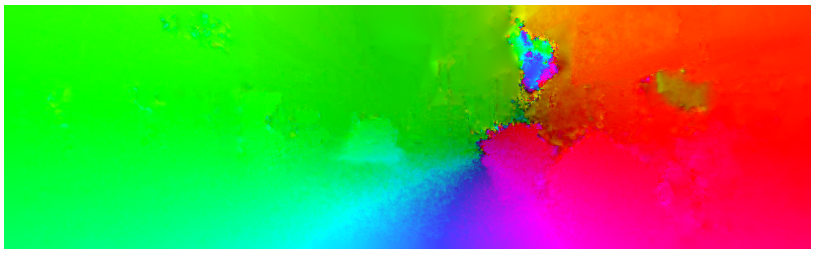
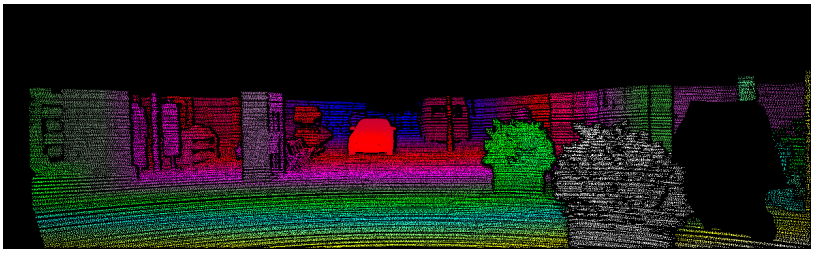
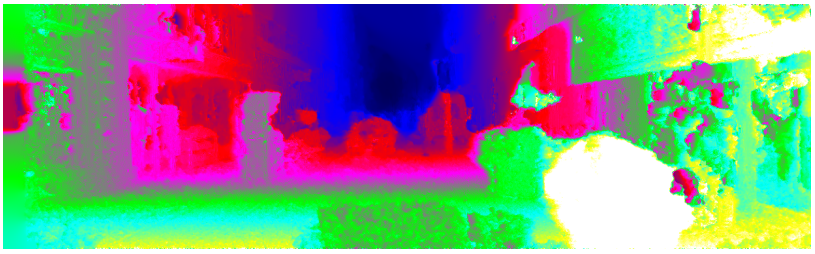


Abbildung 4.14.: *KITTI* Datensatz 000168. (a) Referenzbild zum Zeitpunkt t. (b) Referenzbild zum Zeitpunkt $t + 1$. (c) Sollwerte des Szenenflusses. (d) Durch Einzeloptimierung berechneter Szenenfluss. (e) Berechneter Szenenfluss bei Verwendung der vorab kalibrierten Modellparameter.

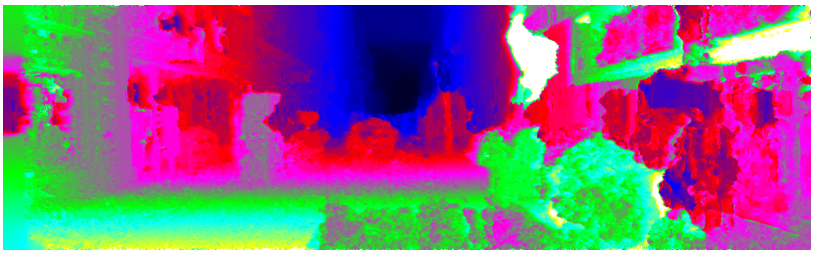
(a)



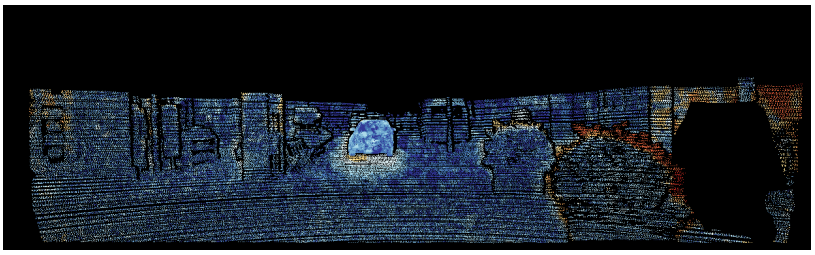
(b)



(c)



(d)



(e)

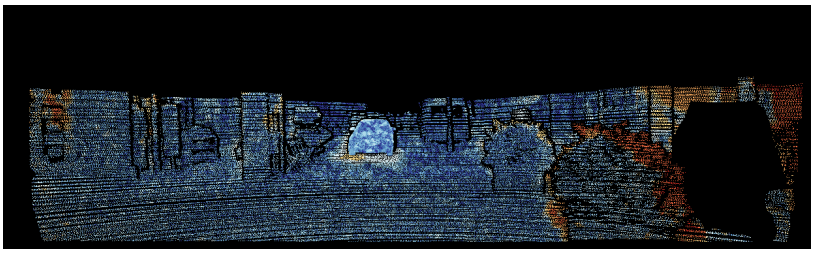


Abbildung 4.15.: *KITTI* Datensatz 000168. (a) Sollwerte der Tiefe. (b) Durch Einzeloptimierung berechnete Tiefenwerte. (c) Berechnete Tiefenwerte bei Verwendung der vorab kalibrierten Modellparameter. (d) Fehlerbild der Einzeloptimierung. (e) Fehlerbild bei Verwendung der vorab kalibrierten Modellparameter.

5. Zusammenfassung und Ausblick

In dieser Arbeit ist ein 3D parametrisierter Variationsansatz zur Berechnung des Szenenflusses, basierend auf der Arbeit von Basha *et al.* [9], präsentiert worden. Auf Basis des Lochkameramodells wurde die verwendete 3D Parametrisierung, welche die Verwendung einer beliebigen Anzahl an Kameraansichten ermöglicht, hergeleitet. Anschließend wurde das verwendete nicht konvexe und nichtlineare Energiefunktional des Variationsansatzes besprochen und dessen Minimierungsverfahren erläutert. Als eine Erweiterung des Verfahrens wurde die Gradientenkonstanzannahme in den Datenterm inkorporiert. Die resultierenden Euler-Lagrange Gleichungen wurden unter Verwendung der kompakten Tensor Notation formuliert.

In einer anschließenden Evaluation wurde das Verfahren durch sukzessiv anspruchsvoller werdende Datensätze erprobt. Neben der grundlegenden Funktionalität wurde der Einfluss zusätzlicher Kameraperspektiven und der Gradientenkonstanzannahme getestet. Während die Verwendung zusätzlicher Kameraansichten zu einer generellen Qualitätssteigerung der Ergebnisse führte, ermöglichte es die Gradientenkonstanzannahme, eventuell auftretende Beleuchtungsänderungen zwischen den Bildern zu berücksichtigen. Durch Verwendung von realitätsechten Datensätzen der *KITTI Benchmark Suite* wurde eine Kalibrierung der in dem Verfahren verwendeten Modellparamter vorgenommen. Es konnte gezeigt werden, dass unter Verwendung der kalibrierten Parameter vernünftige Ergebnisse aus ähnlichen Szenen gewonnen wurden.

Ausblick

Die durchgeführte Evaluation zeigte einige Schwächen des entwickelten Szenenflussverfahrens auf. Vor allem die realitätsechten *KITTI*-Datensätze beinhalten noch viele Schwierigkeiten, die Themen zu-

künftiger Weiterentwicklungen sein könnten. Diese werden in der folgenden Liste zusammengefasst:

- **Behandlung von Verdeckungen**. Für manche Bildpunkte existiert aufgrund von Verdeckungen keine entsprechende Korrespondenz in den weiteren Bildern. Dies führt zu einer Verletzung der Helligkeitswertkonstanzannahme und vermindert so die Qualität der Ergebnisse. Basha *et al.* [9] berechnet solche Verdeckungen und deaktiviert den Datenterm an den entsprechenden Stellen. Dadurch werden die durch Verdeckungen resultierenden Fehler auf ein Minimum reduziert. Von einer ähnliche Verfahrensweise könnte die vorgestellte Methode ebenfalls profitieren.
- **Weiterentwicklung der Glattheitsterme**. Da der Gradient der verwendeten Glattheitsterme auf einer 2D-Ebene berechnet wird, berücksichtigt dieser die mit der Distanz größer werdende Abstände zwischen den zugehörigen 3D-Punkten nicht. Hierdurch kann es zu Problemen bei perspektivisch verzerrten Flächen kommen, wie es z.B. bei vielen Straßen in den *KITTI*-Datensätzen der Fall ist. Ein Glattheitsterm 2. Ordnung könnte für derartige Situationen besser geeignet sein [32].
- **Behandlung von nicht-Lambert'schen Oberflächen**. In den Szenen der *KITTI*-Datensätze verletzen häufig reflektierende Lackoberflächen und Fensterscheiben die Helligkeitswertkonstanzannahme und führen so zu ungenauen oder sogar falschen Berechnungen. Die Verwendung eines komplexeren Reflektionsmodells könnte in einigen realitätsechten Szenen zu erheblichen Verbesserungen führen.
- **Behandlung großer Abstände**. Große Abstände, die durch schnelle Bewegungen zwischen zeitlich aufeinanderfolgenden Bildern hervorgerufen wurden, erschweren die Korrespondenzfindung und können von dem Verfahren nicht immer zuverlässig behandelt werden. Aufgrund der perspektivischen Projektion gilt dies besonders für Objekte, die sich im Bildvordergrund befinden. Um die Situation zu verbessern, bedienen sich Methoden, die den optischen Fluss berechnen, zusammengehörender Bildmerkmale (engl.: Feature Matches) [4, 27]. Eine entsprechende Erweiterung des Szenenflussverfahrens könnte sich als sinnvoll erweisen.

A. Euler-Lagrange-Gleichungen

Die Euler-Lagrange-Gleichung ohne Tensornotation bezüglich der Variablen Z, nach der Taylorreihenlinearisierung aus Gleichung 3.31, ist beschrieben durch:

$$
\begin{aligned}
0 = \sum_{i=0}^{N-1} \Psi'\Big(\Big(\quad & \Delta_i^{t,k} + (\Delta_i^{t,k})_Z \cdot dZ^k + (\Delta_i^{t,k})_u \cdot du^k \\
& + (\Delta_i^{t,k})_v \cdot dv^k + (\Delta_i^{t,k})_w \cdot dw^k\Big)^2\Big) \\
& \cdot \Big(\Delta_i^{t,k} + (\Delta_i^{t,k})_Z \cdot dZ^k + (\Delta_i^{t,k})_u \cdot du^k \\
& + (\Delta_i^{t,k})_v \cdot dv^k + (\Delta_i^{t,k})_w \cdot dw^k\Big) \cdot (\Delta_i^{t,k})_Z \\
+ \sum_{i=1}^{N-1} \Psi'\Big(\Big(\quad & \Delta_i^k + (\Delta_i^k)_Z \cdot dZ^k\Big)^2\Big) \\
& \cdot \Big(\Delta_i^k + (\Delta_i^k)_Z \cdot dZ^k\Big) \cdot (\Delta_i^k)_Z \\
+ \sum_{i=1}^{N-1} \Psi'\Big(\Big(\quad & \hat{\Delta}_i^k + (\hat{\Delta}_i^k)_Z \cdot dZ^k + (\hat{\Delta}_i^k)_u \cdot du^k \\
& + (\hat{\Delta}_i^{t,k})_v \cdot dv^k + (\hat{\Delta}_i^k)_w \cdot dw^k\Big)^2\Big) \\
& \cdot \Big(\hat{\Delta}_i^k + (\hat{\Delta}_i^k)_Z \cdot dZ^k + (\hat{\Delta}_i^k)_u \cdot du^k \\
& + (\hat{\Delta}_i^{t,k})_v \cdot dv^k + (\hat{\Delta}_i^k)_w \cdot dw^k\Big) \cdot (\hat{\Delta}_i^k)_Z \\
- \mu \cdot \mathrm{div}\Big(\quad & \Psi'\Big(|\nabla Z^k + dZ^k|^2\Big) \cdot \nabla(Z^k + dZ^k) \quad \Big) .
\end{aligned}
$$

Die Euler-Lagrange-Gleichung ohne Tensornotation bezüglich der Variablen u, nach der Taylorreihenlinearisierung aus Gleichung 3.31, ist beschrieben durch:

$$
\begin{aligned}
0 = \sum_{i=0}^{N-1} \Psi'\Big(\Big(\quad & \Delta_i^{t,k} + (\Delta_i^{t,k})_Z \cdot dZ^k + (\Delta_i^{t,k})_u \cdot du^k \\
& + (\Delta_i^{t,k})_v \cdot dv^k + (\Delta_i^{t,k})_w \cdot dw^k\Big)^2\Big) \\
& \cdot \Big(\Delta_i^{t,k} + (\Delta_i^{t,k})_Z \cdot dZ^k + (\Delta_i^{t,k})_u \cdot du^k \\
& + (\Delta_i^{t,k})_v \cdot dv^k + (\Delta_i^{t,k})_w \cdot dw^k\Big) \cdot (\Delta_i^{t,k})_u \\
+ \sum_{i=1}^{N-1} \Psi'\Big(\Big(\quad & \hat{\Delta}_i^{k} + (\hat{\Delta}_i^{k})_Z \cdot dZ^k + (\hat{\Delta}_i^{k})_u \cdot du^k \\
& + (\hat{\Delta}_i^{t,k})_v \cdot dv^k + (\hat{\Delta}_i^{k})_w \cdot dw^k\Big)^2\Big) \\
& \cdot \Big(\hat{\Delta}_i^{k} + (\hat{\Delta}_i^{k})_Z \cdot dZ^k + (\hat{\Delta}_i^{k})_u \cdot du^k \\
& + (\hat{\Delta}_i^{t,k})_v \cdot dv^k + (\hat{\Delta}_i^{k})_w \cdot dw^k\Big) \cdot (\hat{\Delta}_i^{k})_u \\
- \alpha \cdot \mathrm{div}\Big(\quad & \Psi'\Big(|\nabla u^k + du^k|^2 + |\nabla v^k + dv^k|^2 + |\nabla w^k + dw^k|^2\Big) \\
& \cdot \nabla(u^k + du^k) \quad \Big) .
\end{aligned}
$$

Die hier fehlenden Euler-Lagrange-Gleichungen bezüglich v und w sind wiederum analog zu der Gleichung für u zu berechnen.

B. Diskretisierung der Glattheitsterme

Disketrisierung des Glattheitsterms 1. Ordnung bezüglich der Variablen u mittels geschachtelter zentraler Differenzenquotienten mit den halben Maschenweiten $\frac{1}{2}h_x$ und $\frac{1}{2}h_y$ liefert:

$$
\begin{aligned}
&\operatorname{div}\Big((\Psi'_{Sm,\mathbf{V}})_{i,j}\cdot\nabla(u_{i,j}+du_{i,j})\Big)\\
&\quad\approx\Big((\Psi'_{Sm,\mathbf{V}})_{i,j}\cdot(u_{i,j}+du_{i,j})_x\Big)_x+\Big((\Psi'_{Sm,\mathbf{V}})_{i,j}\cdot(u_{i,j}+du_{i,j})_y\Big)_y\\
&\quad=\frac{(\Psi'_{Sm,\mathbf{V}})_{i+\frac{1}{2},j}\cdot(u_{i+\frac{1}{2},j}+du_{i+\frac{1}{2},j})_x}{2\cdot(\frac{1}{2}h_x)}\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i-\frac{1}{2},j}\cdot(u_{i-\frac{1}{2},j}+du_{i-\frac{1}{2},j})_x}{2\cdot(\frac{1}{2}h_x)}\\
&\qquad+\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j+\frac{1}{2}}\cdot(u_{i,j+\frac{1}{2}}+du_{i,j+\frac{1}{2}})_y}{2\cdot(\frac{1}{2}h_y)}\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j-\frac{1}{2}}\cdot(u_{i,j-\frac{1}{2}}+du_{i,j-\frac{1}{2}})_y}{2\cdot(\frac{1}{2}h_y)}\\
&\quad=\frac{(\Psi'_{Sm,\mathbf{V}})_{i+\frac{1}{2},j}\cdot\frac{(u_{i+1,j}+du_{i+1,j}-u_{i,j}-du_{i,j})}{2\cdot(\frac{1}{2}h_x)}}{2\cdot(\frac{1}{2}h_x)}\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i-\frac{1}{2},j}\cdot\frac{(u_{i,j}+du_{i,j}-u_{i-1,j}-du_{i-1,j})}{2\cdot(\frac{1}{2}h_x)}}{2\cdot(\frac{1}{2}h_x)}\\
&\qquad+\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j+\frac{1}{2}}\cdot\frac{(u_{i,j+1}+du_{i,j+1}-u_{i,j}-du_{i,j})}{2\cdot(\frac{1}{2}h_y)}}{2\cdot(\frac{1}{2}h_y)}\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j-\frac{1}{2}}\cdot\frac{(u_{i,j}+du_{i,j}-u_{i,j-1}-du_{i,j-1})}{2\cdot(\frac{1}{2}h_y)}}{2\cdot(\frac{1}{2}h_y)}\,.
\end{aligned}
\tag{B.1}
$$

Ausdrücke, die auf den Gitterkanten liegen, wie z.B. $(\Psi'_{Sm,\mathbf{V}})_{i+\frac{1}{2},j}$, werden durch den Mittelwert der benachbarten Felder in der jeweiligen Richtung angenähert:

$$
\begin{aligned}
&\operatorname{div}\left((\Psi'_{Sm,\mathbf{V}})_{i,j}\cdot\nabla(u_{i,j}+du_{i,j})\right)\\
&\quad\approx \frac{\frac{(\Psi'_{Sm,\mathbf{V}})_{i+1,j}+(\Psi'_{Sm,\mathbf{V}})_{i,j}}{2}\cdot\frac{(u_{i+1,j}+du_{i+1,j}-u_{i,j}-du_{i,j})}{h_x}}{h_x}\\
&\qquad-\frac{\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j}+(\Psi'_{Sm,\mathbf{V}})_{i-1,j}}{2}\cdot\frac{(u_{i,j}+du_{i,j}-u_{i-1,j}-du_{i-1,j})}{h_x}}{h_x}\\
&\qquad+\frac{\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j+1}+(\Psi'_{Sm,\mathbf{V}})_{i,j}}{2}\cdot\frac{(u_{i,j+1}+du_{i,j+1}-u_{i,j}-du_{i,j})}{h_y}}{h_y}\\
&\qquad-\frac{\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j}+(\Psi'_{Sm,\mathbf{V}})_{i,j-1}}{2}\cdot\frac{(u_{i,j}+du_{i,j}-u_{i,j-1}-du_{i,j-1})}{h_y}}{h_y}\\
&\quad=\frac{(\Psi'_{Sm,\mathbf{V}})_{i+1,j}+(\Psi'_{Sm,\mathbf{V}})_{i,j}}{2h_x^2}\cdot(u_{i+1,j}+du_{i+1,j}-u_{i,j}-du_{i,j})\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j}+(\Psi'_{Sm,\mathbf{V}})_{i-1,j}}{2h_x^2}\cdot(u_{i,j}+du_{i,j}-u_{i-1,j}-du_{i-1,j})\\
&\qquad+\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j+1}+(\Psi'_{Sm,\mathbf{V}})_{i,j}}{2h_y^2}\cdot(u_{i,j+1}+du_{i,j+1}-u_{i,j}-du_{i,j})\\
&\qquad-\frac{(\Psi'_{Sm,\mathbf{V}})_{i,j}+(\Psi'_{Sm,\mathbf{V}})_{i,j-1}}{2h_y^2}\cdot(u_{i,j}+du_{i,j}-u_{i,j-1}-du_{i,j-1})\,.
\end{aligned}
\tag{B.2}
$$

Die diskretisierten Glattheitsterme bezüglich Z, v und w können analog berechnet werden. Eine Herleitung der Disketrisierung des Z-Glattheitsterms ist in [20] zu finden.

	i - 1	i	i + 1
j-1		$\frac{\alpha}{2 \cdot h_y^2} \left((\Psi'_{Sm})^{k,l}_{i,j-1} + (\Psi'_{Sm})_{i,j} \right)$	
j	$\frac{\alpha}{2 \cdot h_x^2} \left((\Psi'_{Sm} + (\Psi'_{Sm})^{k,l}_{i,j} \right)$	$-\frac{\alpha}{2 \cdot h_x^2} \left((\Psi'_{Sm})^{k,l}_{i-1,j} + 2(\Psi'_{Sm})^{k,l}_{i,j} + (\Psi'_{Sm})^{k,l}_{i+1,j} \right)$ $-\frac{\alpha}{2 \cdot h_y^2} \left((\Psi'_{Sm})^{k,l}_{i,j-1} + 2(\Psi'_{Sm})^{k,l}_{i,j} + (\Psi'_{Sm})^{k,l}_{i,j+1} \right)$	$\frac{\alpha}{2 \cdot h_x^2} \left((\Psi'_{Sm})^{k,l}_{i+1,j} + (\Psi'_{Sm})^{k,l}_{i,j} \right)$
j + 1		$\frac{\alpha}{2 \cdot h_y^2} \left((\Psi'_{Sm})^{k,l}_{i,j+1} + (\Psi'_{Sm})_{i,j} \right)$	

Tabelle B.1.: Matrize für isotropen Glattheitsterm 1. Ordnung. Kann sowohl für $\Psi'_{Sm,\mathbf{V}}$ als auch für $\Psi'_{Sm,Z} = \Psi'(|\nabla(Z + dZ)|^2)$ verwendet werden.

C. Ergebnisse Bündel- und Einzeloptimierung

KITTI Trainingsdatensatz	$\mathrm{NRMS}_{\mathbf{P}}$ [%]	$\mathrm{NRMS}_{\mathbf{V}}$ [%]	AAE_{3D}	BP [%]
000002 (B)	10.00	10.00	12.349	15.26
000002 (E)	8.46	8.42	9.964	11.43
000006 (B)	6.50	6.50	11.887	49.17
000006 (E)	4.99	5.03	10.936	43.14
000011 (B)	26.14	26.02	9.858	13.40
000011 (E)	27.38	27.23	9.153	17.67
000022 (B)	16.77	16.20	18.358	25.18
000022 (E)	26.99	26.63	17.085	31.64
000069 (B)	6.46	6.53	1.204	35.80
000069 (E)	5.34	5.49	1.128	31.44
000133 (B)	4.44	4.21	8.0717	55.78
000133 (E)	7.53	7.34	8.336	28.37
000151 (B)	5.89	5.85	13.681	7.54
000151 (E)	5.45	5.43	13.252	3.54
000152 (B)	4.76	4.75	7.948	5.12
000152 (E)	4.56	4.54	9.199	3.36
000154 (B)	15.58	15.49	9.753	30.02
000154 (E)	6.11	6.11	12.821	15.13
000165 (B)	9.21	9.18	13.33	11.03
000165 (E)	7.56	7.52	9.239	6.52

Tabelle C.1.: Vergleich der berechneten Fehlerwerte zwischen Bündeloptimierung (B) und Einzeloptimierung (E).

Literaturverzeichnis

[1] M. Bertero, T. A. Poggio, V. Torre. „Ill-posed problems in early vision“. In: *Proceedings of the IEEE* 76.8 (1988), S. 869–889 (zitiert auf S. 22).

[2] M. J. Black, P. Anandan. „The robust estimation of multiple motions“. In: *Computer Vision and Image Understanding* 63.1 (1996), S. 75–104 (zitiert auf S. 23).

[3] T. Brox, A. Bruhn, N. Papenberg, J. Weickert. „High accuracy optical flow estimation based on a theory for warping“. In: *European Conference on Computer Vision (ECCV)*. 2004, S. 25–36 (zitiert auf S. 3, 23, 26, 27, 40).

[4] T. Brox, J. Malik. „Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33.3 (2011), S. 500–513 (zitiert auf S. 70).

[5] A. Bruhn. *Variational Optic Flow Computation - Accurate Modelling and Efficient Numerics*. PhD. Thesis. Saarland University. 2006 (zitiert auf S. 2, 27, 28, 33).

[6] A. Bruhn, J. Weickert. „Towards ultimate motion estimation: combining highest accuracy with real-time performance“. In: *IEEE International Conference on Computer Vision (ICCV)*. 2005, S. 749–755 (zitiert auf S. 2, 27, 28, 40, 41).

[7] B. van Brunt. *The Calculus of Variations*. Springer, 2004 (zitiert auf S. 14, 15).

[8] R. Courant, D. Hilbert. *Methods of Mathematical Physics*. Interscience Publishers Inc., 1953 (zitiert auf S. 15).

[9] T. Dekel (Basha), Y. Moses, N. Kiryati. „Multi-view scene flow estimation: A view centered variational approach“. In: *International Journal of Computer Vision (IJCV)* 101.1 (2013), S. 6–21 (zitiert auf S. i, 2, 4, 17, 36, 40, 57, 69, 70).

[10] J. Diebel. *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. 2006 (zitiert auf S. 7, 8).

[11] A. Forsyth. *Calculus of Variations*. Dover Publications Inc., 1960 (zitiert auf S. 14).

[12] A. Geiger, P. Lenz, C. Stiller, R. Urtasun. „Vision meets robotics: the KITTI dataset“. In: *International Journal of Robotics Research (IJRR)* 32.11 (2013), S. 1231–1237 (zitiert auf S. 59).

[13] A. Geiger, P. Lenz, R. Urtasun. „Are we ready for autonomous driving? The KITTI vision benchmark suite“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, S. 3354–3361 (zitiert auf S. 46, 59).

[14] R. I. Hartley, A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004 (zitiert auf S. 5, 7).

[15] E. Herbst, X. Ren, D. Fox. „RGB-D flow: Dense 3-D motion estimation using color and depth“. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013, S. 2276–2282 (zitiert auf S. 4).

[16] F. Huguet, F. Devernay. „A variational method for scene flow estimation from stereo sequences“. In: *IEEE International Conference on Computer Vision (ICCV)*. 2007, S. 1–7 (zitiert auf S. 3, 4).

[17] H. Kielhofer. *Variationsrechnung: Eine Einfuhrung in die Theorie einer unabhangigen Variablen mit Beispielen und Aufgaben*. Vieweg + Teubner, 2010 (zitiert auf S. 14).

[18] P. Lenz, J. Ziegler, A. Geiger, M. Roser. „Sparse scene flow segmentation for moving object detection in urban environments“. In: *IEEE Intelligent Vehicles Symposium (IV)*. 2011, S. 926–932 (zitiert auf S. 1).

[19] P. Liu, M. Reale, L. Yin. „3D Head Pose Estimation Based on Scene Flow and Generic Head Model“. In: *IEEE International Conference on Multimedia and Expo (ICME)*. 2012, S. 794–799 (zitiert auf S. 1).

[20] D. Maurer. „Depth-Driven Variational Methods for Stereo Reconstruction“. Master Thesis. University of Stuttgart, 2014 (zitiert auf S. 4, 15, 18, 21, 34, 36, 37, 40, 74).

[21] M. Menze, A. Geiger. „Object scene flow for autonomous vehicles“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015 (zitiert auf S. 1, 3, 4, 46).

[22] M. Menze, C. Heipke, A. Geiger. „Joint 3D estimation of vehicles and scene flow“. In: *ISPRS Workshop on Image Sequence Analysis (ISA)*. 2015 (zitiert auf S. 1, 4).

[23] J. A. Nelder, R. Mead. „A simplex method for function minimization“. In: *The Computer Journal* 7.4 (1965), S. 308 (zitiert auf S. 45).

[24] J. Richter-Gebert. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*. Springer Publishing Company, Incorporated, 2011 (zitiert auf S. 6).

[25] D. Scharstein, C. Pal. „Learning conditional random fields for stereo“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2007, S. 1–8 (zitiert auf S. 54).

[26] D. Scharstein, R. Szeliski. „High-accuracy stereo depth maps using structured light“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2003, S. 195–202 (zitiert auf S. 49).

[27] M. Stoll, S. Volz, A. Bruhn. „Adaptive integration of feature matches into variational optical flow methods“. In: *Asian Conference on Computer Vision (ACCV)*. 2013, S. 1–14 (zitiert auf S. 70).

[28] D. Stoyanov. „Stereoscopic scene flow for robotic assisted minimally invasive surgery“. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2012, S. 479–486 (zitiert auf S. 1).

[29] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, C. Theobalt. „Joint estimation of motion, structure and geometry from stereo sequences“. In: *European Conference on Computer Vision (ECCV)*. 2010, S. 568–581 (zitiert auf S. 3, 4).

[30] S. Vedula, S. Baker, P. Rander, R. Collins, T. Kanade. „Three-dimensional scene flow“. In: *IEEE International Conference on Computer Vision (ICCV)*. 1999, S. 722–729 (zitiert auf S. 1, 2).

[31] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, D. Cremers. „Efficient dense scene flow from sparse or dense stereo data“. In: *European Conference on Computer Vision (ECCV)*. 2008, S. 739–751 (zitiert auf S. 3, 4).

[32] O. Woodford, P. Torr, I. Reid, A. Fitzgibbon. „Global stereo reconstruction under second-order smoothness priors“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31.12 (2009), S. 2115–2128 (zitiert auf S. 70).

[33] Z. Yan, X. Xiang. „Scene Flow Estimation: A Survey“. In: *Computing Research Repository (CoRR)* abs/1612.02590 (2016) (zitiert auf S. 1, 4, 49, 51).

[34] D. Young. „Iterative methods for solving partial difference equations of elliptic type“. In: *Transactions of the American Mathematical Society* 76.1 (1954), S. 92–111 (zitiert auf S. 33, 40, 48).

[35] A. Zanfir, C. Sminchisescu. „Large Displacement 3D Scene Flow with Occlusion Reasoning“. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, S. 4417–4425 (zitiert auf S. 4).

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.

Stuttgart, den 03. Mai 2017

Steffen Nüssle