Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

RAFT-based Cost Volume Analysis for Rectified Stereo

Jakob Schmid

Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. Andrés Bruhn
Betreuerin:	M. Sc. Azin Jahedi
begonnen am:	31. März, 2021
beendet am:	28. Oktober, 2021

Kurzfassung

Das Stereokorrespondenzproblem ist ein wichtiges Thema im Bereich Computer Vision. Bei diesem Problem wird in einem Stereobildpaar die Verschiebung (Disparität) von Pixelpaaren gesucht, die den gleichen Punkt in einer Szene zeigen. Aus diesem Versatz kann die 3D-Position der aufgenommenen Objekte rekonstruiert werden. Basierend auf dem Recurrent All-Pairs Field Transform (RAFT), einer Deep Learning Methode, die sehr gute Ergebnisse beim optischen Fluss erzielt, wird eine Methode für die Stereokorrespondenz entwickelt. Dafür wird ein Korrelationsvolumen entwickelt, das die Einschränkungen der Epipolargeometrie für Stereobilder berücksichtigt. Für dieses Korrelationsvolumen werden verschiedene Lookup-Strategien getestet. Ein Lookup mit einem statischen Bereich um das Pixel, an dem die Disparität gesucht wird, kann nicht das volle Potenzial der iterativen Aktualisierungen von RAFT ausnutzen. Durch ein Lookup um die geschätzte Position des konjugierten Pixels kann dieses Problem behoben werden. In einem Test mit den Sintel, KITTI 2012 und KITTI 2015 Stereodatensätzen liefert das Stereokorrelationsvolumen mit dieser Lookup-Methode Verbesserungen im durchschnittlichen Endpunktfehler und im D1-Fehler gegenüber dem originalen RAFT Korrelationsvolumen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Abkürzungen

AEE durchschnittlicher Endpunktfehler

CNN Convolutional Neural Network

GRU Gated Recurrent Unit

GwcNet Group-wise Correlation Stereo Network

RAFT Recurrent All-Pairs Field Transform

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

1 Einleitung

Das Stereokorrespondenzproblem ist eines der grundlegenden Probleme im Bereich Computer Vision. Bei diesem Problem ist ein Bildpaar gegeben, das eine Szene zeigt, die aus unterschiedlichen Positionen aufgenommen wurde. Gesucht ist eine Zuordnung von Pixeln zwischen den beiden Bildern. Die Zuordnung zweier Pixel wird normalerweise in Form der Disparität angegeben, dem Versatz der Position der beiden Pixel.

Sind die relative Position und die Kameraparameter bekannt, kann aus dieser Zuordnung die 3D-Position der Punkte rekonstruiert werden. Die Anwendungsbereiche sind unter anderem Fotogrammetrie, Robotik und autonomes Fahren. In der Robotik und beim autonomen Fahren können Tiefeninformationen bei Aufgaben wie der Planung und Navigation helfen.

In der letzten Zeit gab es Fortschritte beim Problem des optischen Flusses. Das Problem des optischen Flusses ist verwandt mit dem Stereokorrespondenzproblem. Gesucht wird wie bei der Stereokorrespondenz eine Zuordnung von Pixeln zwischen zwei Bildern. Anders als bei der Stereokorrespondenz werden die Bilder nicht zur gleichen Zeit aufgenommen. Daher können sich die Objekte in der Szene zwischen der Aufnahme des ersten und zweiten Bildes bewegen wie in einem Video.

Der Recurrent All-Pairs Field Transform (RAFT) [TD20] ist eine neue Methode, die überwachtes Lernen für das Schätzen des optischen Flusses benutzt. Sie liefert gute Ergebnisse und eine starke Generalisierung auf Datensätze, die beim Training nicht verwendet werden.

Ziel dieser Arbeit ist es RAFT für die Schätzung der Disparität von rektifizierten Stereobildpaaren zu modifizieren. Dafür werden neue Korrelationsvolumen entwickelt, die die geometrischen Bedingungen der Stereoaufnahmen berücksichtigen. Diese Korrelationsvolumen werden in RAFT integriert und mit Stereobildern evaluiert.

Verwandte Arbeiten

Das Stereokorrespondenzproblem ist ein Problem, das seit geraumer Zeit untersucht wird. Traditionelle Verfahren basieren auf handgefertigten Funktionen, die die Kosten für das Paaren von Pixeln berechnet. Die Paarungskosten werden minimiert, wodurch sich die Disparität ergibt [MP76, ZW94, KRS96, Hir07].

Neuere Methoden setzten auf künstliche neuronale Netze, um die Disparität zu schätzen. Viele Netzwerkarchitekturen bestehen aus einem Netzwerk, das Merkmale zu kleinen Bildbereichen extrahiert. Mit diesen Merkmalen wird durch Konkatenation ein Kostenvolumen generiert, das für jede mögliche Paarung von Pixeln die zugehörigen Merkmale aus dem Merkmal-Netzwerk enthält. Dieses Korrelationsvolumen wird vom neuronalen Netz mit 3D-Faltungen gefiltert, um die Disparitätsschätzung

zu verbessern [KMD+17, CC18, ZPYT19, ZQY+20]. Das Group-wise Correlation Stereo Network (GwcNet) [GYY+19] benutzt einen ähnlichen Aufbau, teilt aber die Merkmale jedes Pixels in Gruppen ein. Das Kostenvolumen enthält zusätzlich zu den konkatenierten Merkmalen für jede der Gruppen die Korrelation der Merkmale der beiden Pixel. In dieser Arbeit wird die RAFT [TD20] Architektur benutzt. Im Gegensatz zu den vorher genannten Methoden besteht das Kostenvolumen nur aus der Korrelation der Merkmale der beiden Pixel. Konkatenierte Merkmale werden nicht benutzt. Statt den 3D-Faltungen zur Schätzung und Optimierung der Disparität aktualisiert RAFT die Disparität iterativ mit seinem Update-Operator.

Gliederung

Im nächsten Kapitel werden die Themen erklärt, die für das Verständnis dieser Arbeit benötigt werden. Dazu zählen Grundlagen wie die Geometrie von Stereobildaufnahmen und die RAFT Methode. In Kapitel 3 wird die Funktionsweise und Implementierung der Stereokorrelationsvolumen beschrieben. Danach, in Kapitel 4, werden der Trainingsvorgang beschrieben und die Ergebnisse evaluiert. Das Kapitel 5 enthält eine abschließende Zusammenfassung und einen Ausblick.

2 Grundlagen

2.1 Kameramodell

Die Projektion von 3D-Koordinaten auf den Kamerasensor kann sehr komplex sein, besonders wenn die Kamera mehrere Linsen hat. Allerdings lässt sie sich gut mit dem Modell der Lochkamera annähern, wenn die Kamera gut fokussiert ist. Beim Modell der Lochkamera entsteht das Bild durch eine Öffnung ohne räumliche Ausdehnung, sodass alle Sichtstrahlen denselben Punkt passieren.

Abbildung 2.1a zeigt die perspektivische Projektion eines 3D-Szenenpunkts $\boldsymbol{M}$ mit den Koordinaten (x, y, z) auf den 2D-Bildpunkt $\boldsymbol{m}$ mit den Koordinaten (u, v). Der Szenenpunkt wird mit einem Strahl durch den Brennpunkt auf den Schnittpunkt mit der Bildebene projiziert. Mithilfe des Strahlensatzes ergibt sich

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{z} \cdot \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.1}$$

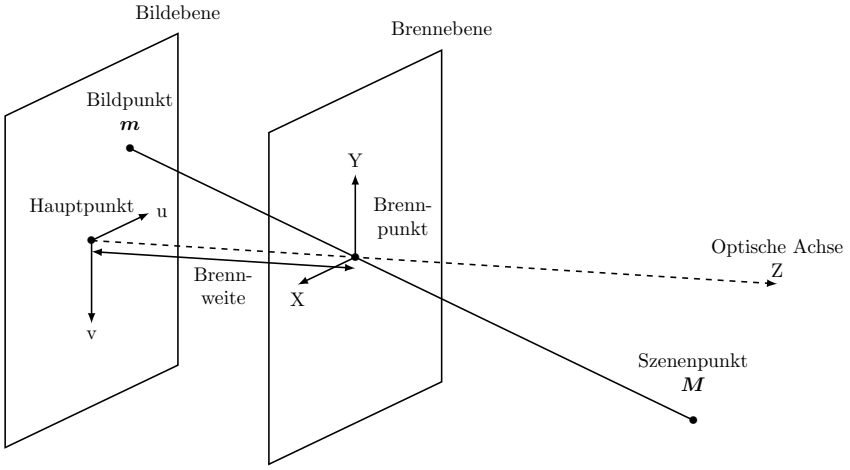
wobei f die Brennweite der Kamera ist. Bilder, die mit solch einer Kamera aufgenommen werden, sind spiegelverkehrt. Um dies zu beheben, sind die Koordinatenachsen der Bildebene auch gespiegelt zu denen des Weltkoordinatensystems.

Eine Vereinfachung des Modells ist in Abbildung 2.1b zu sehen. Gegenüber dem ersten Modell befindet sich hier die Bildebene vor statt hinter der Brennebene. Eine solche Kamera ist physikalisch nicht realisierbar, die entstehende Projektion ist jedoch die Gleiche. Im Gegensatz zum vorherigen Modell ist das Bild gleich wie die ursprüngliche Szene ausgerichtet. Eine Spiegelung ist daher nicht mehr nötig.

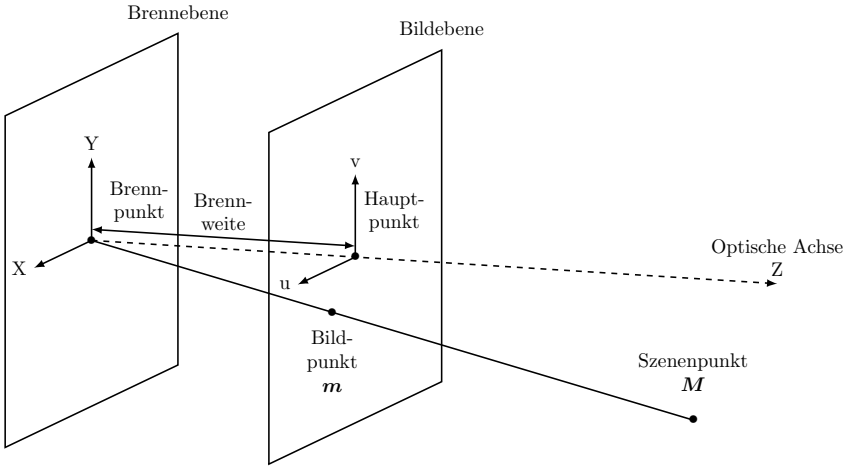
Bei der oben beschriebenen Projektion werden mehrere Szenenpunkte auf denselben Bildpunkt projiziert. Zwei Punkte $\boldsymbol{M_1}$ und $\boldsymbol{M_2}$ werden genau dann auf den gleichen Punkt projiziert, wenn beide auf demselben Sichtstrahl liegen und daher $\boldsymbol{M_1} = \lambda \boldsymbol{M_2}$ gilt. In einem solchen Fall ist nur der vorderste Szenenpunkt im Bild sichtbar und alle Szenenpunkte dahinter sind verdeckt.

2.2 Epipolargeometrie

Durch die perspektivische Projektion bei einer Lochbildkamera ist es nicht möglich aus einem Bild die Tiefe der abgebildeten Objekte zu bestimmen. Mithilfe einer zweiten Kamera, die die Szene aus einer anderen Position aufnimmt, kann durch die Suche nach korrespondierenden Punkten die Tiefe von Bildpunkten berechnet werden. Um einen korrespondierenden Punkt zu finden, ist keine erschöpfende Suche notwendig. Die Epipolargeometrie liefert eine Bedingung, die diese Suche vereinfacht.



(a) Lochkameramodell mit Bildebene hinter der Brennebene



(b) Lochkameramodell mit Bildebene vor der Brennebene

Abbildung 2.1: Eine Projektion mit dem Modell einer Lochkamera. Der 3D-Szenenpunkt wird auf den 2D-Bildpunkt projiziert. Die Projektion aus Abbildung 2.1b stimmt mit der aus Abbildung 2.1a überein, allerdings muss das Bild nicht mehr gespiegelt werden.

In Abbildung 2.2 ist die Aufnahme einer Szene durch zwei Kameras aus unterschiedlichen Perspektiven zu sehen. Ausgehend von einem Bildpunkt in Bild 1 ist es möglich den zugehörigen Sichtstrahl zu bestimmen. Dieser Sichtstrahl enthält alle möglichen Positionen, an denen sich der Szenenpunkt befinden kann. Durch eine Abbildung dieses Sichtstrahls auf Bild 2, ergibt sich die zugehörige Epipolarlinie. Der korrespondierende Punkt kann in Bild 2 nur entlang dieser Linie liegen. Daher kann der Suchraum auf die Epipolarlinie eingegrenzt werden.

Rektifizierte Bilder erleichtern das Finden der Epipolarlinien. Sie werden mit einer parallelen Stereokameraanordnung aufgenommen und etwaige Verzerrungen oder Versätze der beiden Bilder werden entfernt. Die Position, aus der die Bilder aufgenommen sind, unterscheidet sich einzig durch eine

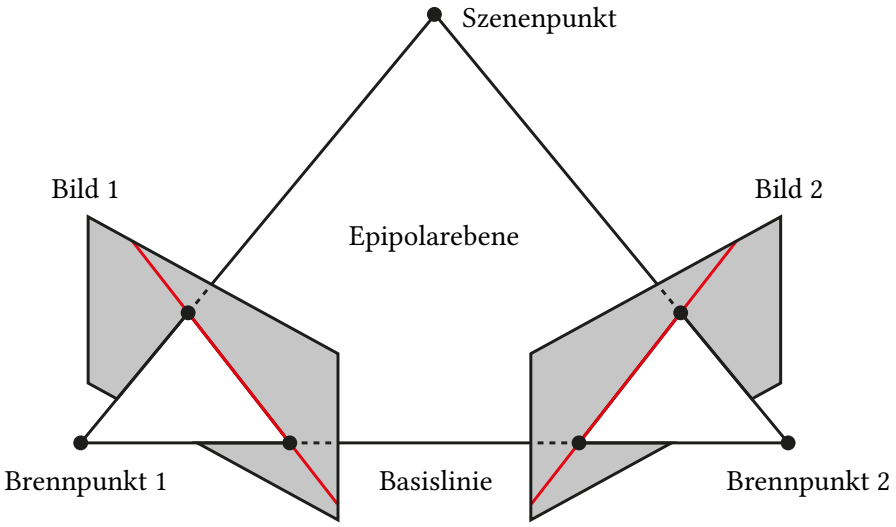


Abbildung 2.2: Aufnahme einer Szene durch zwei Kameras. Die Epipolarebene wird von den beiden Brennpunkten und dem Szenenpunkt aufgespannt. Die Schnittgeraden mit den Bildebenen, hier als rote Linie eingezeichnet, wird als Epipolarlinie bezeichnet.

horizontale Verschiebung, der Basisliniendistanz. Durch diese Anordnung ist für jeden Szenenpunkt die Distanz in y- und z-Richtung zu den beiden Kameras die Gleiche. Daher ist nach Gleichung 2.1 auch die Bildkoordinate der beiden Bilder in y-Richtung gleich. Die Epipolarlinie zu einem Punkt befindet sich daher auf gleicher Höhe wie der Punkt selbst und verläuft parallel zur x-Achse.

2.3 Neuronale Netze

Neuronale Netze werden im Bereich des maschinellen Lernens dazu benutzt, um Funktionen an Daten anzupassen. Sie bestehen aus einer beliebigen Anzahl an Ebenen. Diese Ebenen enthalten Neuronen, die jeweils mit allen Neuronen der vorherigen Ebenen verbunden sind. Aus den Aktivitäten der Neuronen aus der vorherigen Ebene $\boldsymbol{x}$, seinen Gewichten $\boldsymbol{w}$ und dem Bias b wird seine Aktivität berechnet:

$$\text{Aktivität} = \phi(\boldsymbol{w} \cdot \boldsymbol{x} + b). \tag{2.2}$$

Die Aktivierungsfunktion ϕ ist nicht linear und monoton steigend und wird komponentenweise angewendet. Häufig wird dafür die Rectified Linear Unit (ReLU) Funktion $\phi(z) = \max\{0, z\}$ gewählt.

Die Gewichte und der Bias der Neuronen werden nicht im Voraus festgelegt, sondern durch das Training bestimmt. Für das Training wird eine differenzierbare Fehlerfunktion benötigt. Diese vergleicht das Ergebnis des neuronalen Netzes mit dem wahren Ergebnis, der Ground Truth, und berechnet einen Fehlerwert. Dieser Fehler wird durch ein Gradientenverfahren minimiert, indem für Trainingsdaten mit den aktuellen Gewichten und den Bias das Ergebnis des neuronalen Netzes berechnet wird. Mit der Ground Truth werden der Fehler des Ergebnisses berechnet und auf Basis dessen die Gewichte und die Bias angepasst.

Außer den normalen neuronalen Netzen, bei denen die Neuronen jeder Ebene mit allen Neuronen aus der vorherigen Ebene verbunden sind, gibt es weitere Typen von neuronalen Netzen. Convolutional Neural Networks (CNNs) enthalten neben den normalen Ebenen auch Covolutional Layer. Bei diesen ergibt sich die Aktivität eines Neurons nicht aus dem Skalarprodukt von Gewichten und der Aktivität aller Neuronen aus der vorherigen Ebene. Stattdessen ergibt sich die Aktivität durch eine diskrete Faltung eines Kernels mit der vorherigen Ebene. Die Aktivität wird daher nur von den Werten in der Umgebung beeinflusst und alle Neuronen dieser Ebene teilen sich die Gewichte. Außerdem werden Pooling Ebenen benutzt. Sie dienen dem Herunterskalieren der Daten, indem Blöcke an Neuronen durch ihr Maximum oder ihren Durchschnitt zusammengefasst werden.

Ein weiterer Typ von neuronalen Netzen sind die Recurrent Neural Networks (RNNs). Zusätzlich zu den Verbindungen von einer Ebene zur nächsten Ebene können die RNNs auch Rückkopplungen enthalten. Der Ausgang eines Neurons kann auch als Eingang für dieses Neuron, Neuronen aus vorherigen Ebenen oder Neuronen aus der gleichen Ebene dienen. Ein solches neuronales Netz ist die Gated Recurrent Unit (GRU). GRUs sind eine beliebte Option RNNs zu implementieren, da sie durch ihren Aufbau weniger unter Problemen beim Training leiden als naive RNNs. Die GRU berechnet in jedem Schritt aus ihrem verborgenen Zustand aus $\boldsymbol{h}_{t-1}$ und einem Vektor $\boldsymbol{x}_t$ einen neuen verborgenen Zustand $\boldsymbol{h}_t$. Ein Reset-Gate $\boldsymbol{r}_t$ berechnet mit den Gewichten $\boldsymbol{W}_r$ und dem Bias $\boldsymbol{b}_r$ die Wichtigkeit des vorherigen Zustands für die Aktualisierung

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_r). \tag{2.3}$$

Als Aktivierungsfunktion σ wird die Sigmoidfunktion elementweise angewendet. Daher liegen die Werte des Reset-Gates immer zwischen 0 und 1. Mit den Werten des Reset-Gates, dem verborgenen Zustand und dem Eingabevektor wird ein Zustandskandidat $\tilde{\boldsymbol{h}}_t$ berechnet

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_h[\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_h). \tag{2.4}$$

Das komponentenweise Produkt wird durch $\odot$ dargestellt, die Gewichte durch $\boldsymbol{W}_h$ und der Bias durch $\boldsymbol{b}_h$. Ein Update-Gate $\boldsymbol{z}_t$ entscheidet darüber, ob der verborgene Zustand $\boldsymbol{h}_{t-1}$ aktualisiert wird oder gleich bleibt. Mit den Gewichten $\boldsymbol{W}_z$ und dem Bias $\boldsymbol{b}_z$ berechnet sich der Wert des Update-Gates durch

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_z), \tag{2.5}$$

mit der elementweisen Sigmoidfunktion σ. Wie beim Reset-Gate liegen die Werte immer zwischen 0 und 1. Aus dem Wert des Zustandskandidaten und dem Wert des Update-Gates wird der neue verborgene Zustand berechnet

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t. \tag{2.6}$$

Das Symbol $\odot$ bezeichnet wieder das komponentenweise Produkt. Für einen Wert des Update-Gates nahe 1 wird der Zustandskandidat als verborgener Zustand übernommen. Für einen Wert nahe 0 bleibt der vorherige verborgene Zustand bestehen.

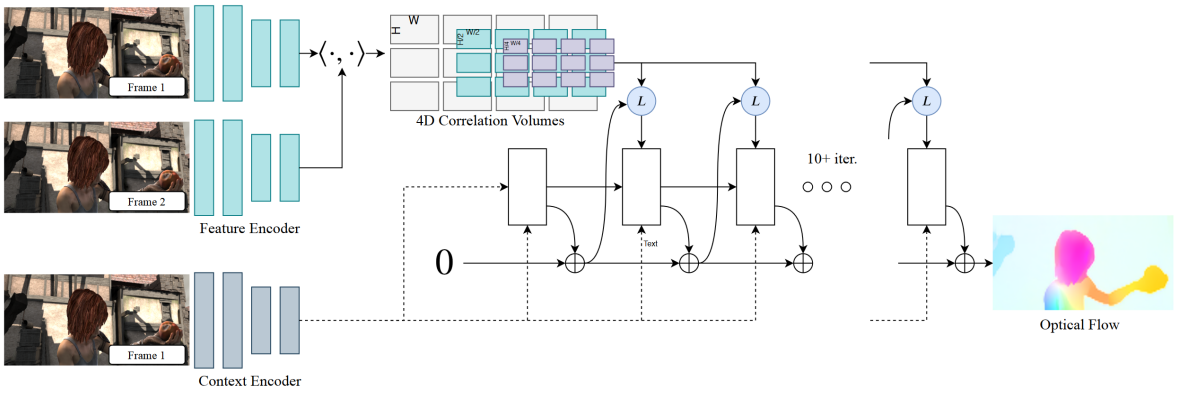


Abbildung 2.3: Der Aufbau von RAFT: Das Merkmal- und Kontext-Netzwerk, das Korrelationsvolumen und der Update-Operator. Das Bild ist aus [TD20].

2.4 RAFT

RAFT [TD20] ist eine Methode, die überwachtes Lernen nutzt, um den optischen Fluss zu approximieren. Sie kann durch gute Resultate und eine starke Generalisierung auf andere Bilder überzeugen.

RAFT besteht aus drei Hauptbestandteilen, dem Merkmal- und Kontext-Netzwerk, dem Korrelationsvolumen und dem Update-Operator. Der Prozess zur Schätzung des optischen Flusses von zwei Bildern wird im Folgenden beschrieben. Gegeben ist ein Bildpaar. Zuerst werden Merkmale, die jeweils kleine Bereiche des Bildes beschreiben, für jedes der Bilder extrahiert. Außerdem werden Informationen über den Kontext der Bereiche aus dem Referenzbild extrahiert. In der zweiten Phase wird die visuelle Ähnlichkeit zwischen allen Pixeln der beiden Bildern berechnet. Dafür wird aus allen Merkmalen zu einem Punkt aus dem ersten Bild und allen Merkmalen zu einem Punkt aus dem zweiten Bild ein Zahlenwert errechnet, der die visuelle Ähnlichkeit misst. Der Update-Operator ruft Werte aus dem Korrelationsvolumen ab und benutzt die Kontext-Merkmale, um die Schätzung des optischen Flusses iterativ zu verbessern. Dieser Aufbau ist in Abbildung 2.3 illustriert.

Merkmal- und Kontext-Netzwerk

Die Merkmale werden mit einem CNN, dem Merkmal-Netzwerk, extrahiert. Dieses Netzwerk bekommt die beiden Bilder als Eingabe. Jedes der Bilder wird in 8×8 Pixel große Blöcke unterteilt. Für jeden dieser Blöcke werden 256 Merkmale extrahiert, die den zugehörigen Bildbereich beschreiben. Dafür wird das Bild von der vollen Auflösung in drei Schritten herunterskaliert. Die Anzahl der Merkmale wird währenddessen erhöht. Die Abbildung 2.4 zeigt die Struktur des Merkmal-Netzwerks.

Auf das erste Bild wird außerdem noch ein zweites neuronales Netz, das Kontext-Netzwerk, angewendet. Das Kontext-Netzwerk ist identisch aufgebaut wie das Merkmal-Netzwerk, hat jedoch eigene Gewichte, wodurch sich die Ausgabe unterscheidet.

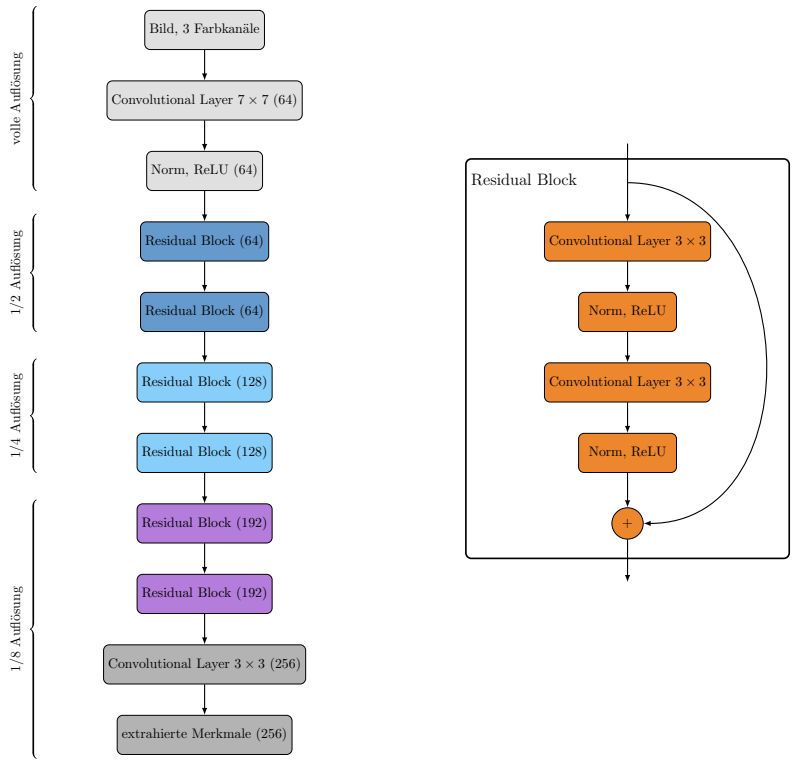


Abbildung 2.4: Die Abbildung zeigt den Aufbau des Merkmal-Netzwerks.

Korrelationsvolumen

Das Korrelationsvolumen speichert die visuelle Ähnlichkeit von Bildpunkten. Aus den Merkmalen der Bilder, die im vorherigen Schritt extrahiert wurden, wird mit dem Skalarprodukt die Ähnlichkeit berechnet. Für zwei Merkmalsvektoren $\boldsymbol{f}$ und $\boldsymbol{g}$ mit den je 256 Merkmalen zu einem Bildpunkt ergibt sich ein Korrelationswert von

$$c = \sum_{k=1}^{256} \boldsymbol{f}_k \cdot \boldsymbol{g}_k. \tag{2.7}$$

Je höher dieser Wert ist, desto ähnlicher sind die Vektoren.

Im Korrelationsvolumen sind Korrelationswerte für die Ähnlichkeit zwischen allen Punkten aus dem ersten Bild und allen Punkten aus dem zweiten Bild gespeichert. Für zwei Bilder mit der Höhe H und der Breite W entsteht ein 4D-Korrelationsvolumen der Größe $H \times W \times H \times W$. Die Korrelation zwischen einem Pixel aus dem ersten Bild an Position (i, j) und einem Pixel aus dem zweiten Bild an Position (k, l) ist im Korrelationsvolumen an der Position (i, j, k, l) gespeichert. Dieses Korrelationsvolumen wird in den letzten beiden Dimensionen mit einem Boxfilter geglättet und herunterskaliert. Dabei werden Filtergrößen von 1, 2, 4 und 8 verwendet. Dadurch entsteht eine Gauß-Pyramide des Korrelationsvolumens.

Für den Update-Operator steht eine Prozedur zur Verfügung, die für einen optischen Fluss die Korrelation zwischen den korrespondierenden Pixeln und den Nachbarn dieser Pixel aus dem Korrelationsvolumen interpoliert. Für jedes Pixel $\boldsymbol{x} = (u, v)^\top$ aus dem ersten Bild und einer Schätzung des optischen Flusses an dessen Position $\boldsymbol{f}(\boldsymbol{x}) = (f^1(u), f^2(v))^\top$ wird ein 9×9 Gitter um das korrespondierende Pixel $\boldsymbol{x}' = (u', v')^\top = (u + f^1(u), v + f^2(v))^\top$ aus Bild 2 gebildet. Auf jeder Ebene der Pyramide wird jeder Punkt des Gitters bilinear interpoliert. Der Gitterabstand wird für die gröberen Ebenen der Pyramide zunehmend größer. Dadurch stehen mit den feinen Ebenen genaue Informationen über die Korrelation nahe des geschätzten optischen Flusses zur Verfügung. Über die groben Ebenen der Pyramide stehen Korrelationswerte für Bildsektoren zur Verfügung, die weit von dem geschätzten optischen Fluss entfernt sind.

Update-Operator

Der Update-Operator ist ein RNN, das iterativ die Vorhersage des optischen Flusses aktualisiert. Zu Beginn wird der optische Fluss mit $f_0 = 0$ initialisiert und in jeder Iteration durch ein Vektorfeld $\Delta \boldsymbol{f}$ angepasst: $\boldsymbol{f}_{i+1} = \boldsymbol{f}_i + \Delta \boldsymbol{f}$.

Der Aufbau des Update-Operators ist in Abbildung 2.5 zu sehen. Kern dieses Update-Operators ist eine GRU. Aus dem vorherigen Schritt, der Korrelation der korrespondierenden Pixel des optischen Flusses, den Kontext-Merkmalen des Kontext-Netzwerks und ihrem Zustand aus dem vorherigen Schritt berechnet die GRU ihren nächsten Zustand. Der optische Fluss und die Korrelation werden jeweils von zwei Convolutional Layern vorverarbeitet. Dadurch werden Merkmale generiert, die es der GRU ermöglichen diese Daten besser zu verarbeiten. Aus dem Zustand der GRU wird mittels zwei Convolutional Layern der optische Fluss geschätzt. Dieser hat 1/8 der Auflösung des Originalbildes durch das Downsampling des Bildes vom Merkmal-Netzwerk in der ersten Phase. Für die Fehlerfunktion des überwachten Lernens und die Evaluation wird der optische Fluss in Originalauflösung benötigt. Deshalb wird der optische Fluss durch Upsampling wieder auf die ursprüngliche Auflösung gebracht. Dies geschieht durch eine konvexe Kombination der Werte der 3×3 Nachbarn auf der groben Auflösung. Die Maske für dieses Upsampling wird von zwei Convolutional Layern generiert.

RAFT [TD20] ist Ende-zu-Ende trainierbar. Daher ist nur eine Fehlerfunktion für das Training der ganzen Methode nötig. Die Fehlerfunktion summiert die L1-Distanzen zwischen dem wahren optischen Fluss $\boldsymbol{f}_g t$ und allen Schätzungen der Methode $\boldsymbol{f}_1$ bis $\boldsymbol{f}_n$. Die Distanzen werden dann mit einem exponentiell steigenden Faktor gewichtet:

$$\mathcal{L} = \sum_{i=1}^{N} \gamma^{n-i} \|\boldsymbol{f}_i - \boldsymbol{f}_g t\|_1, \tag{2.8}$$

mit dem Gewichtungswert $\gamma < 1$.

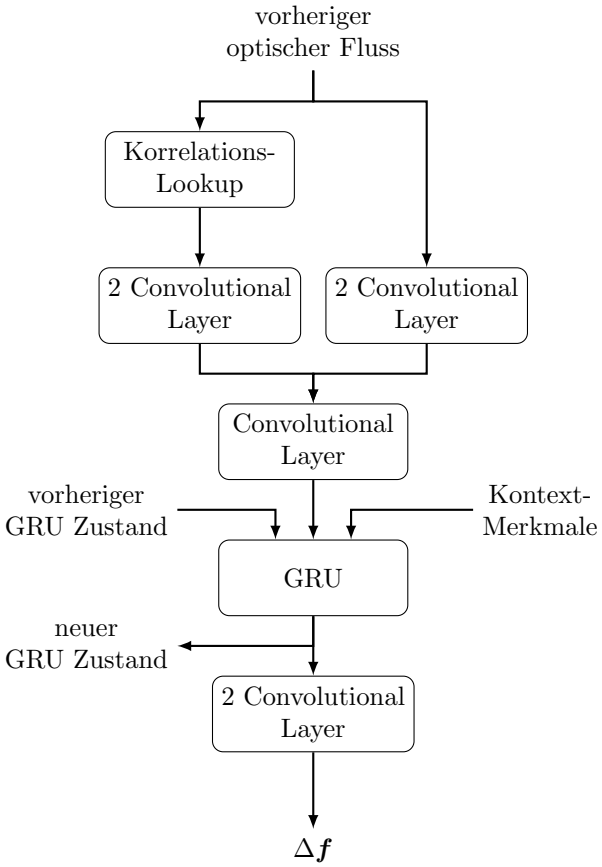


Abbildung 2.5: Aufbau des RAFT Update-Operators. Der optische Fluss und seine Korrelation werden mit mehreren Convolutional Layern vorverarbeitet. Zusammen mit den Kontext-Merkmalen und dem vorherigen GRU Zustand wird der neue GRU Zustand berechnet. Durch zwei Convolutional Layer wird aus dem neuen GRU Zustand das Update des optischen Flusses generiert.

3 Korrelationsvolumen für Stereobildpaare

Im Kapitel 2.4 wurde RAFT eingeführt, eine Methode zur Schätzung des optischen Flusses, die sehr gute Ergebnisse erzielt und eine starke Generalisierung auf verschiedene Datensätze bietet. Wie in Kapitel 2.2 erklärt, gilt gegenüber dem optischen Fluss eine weitere Bedingung bei der Stereokorrespondenz. Die epipolare Bedingung besagt, dass die konjugierten Pixel auf den Epipolarlinien liegen müssen. Das Korrelationsvolumen, das die Ähnlichkeit zwischen zwei Pixeln des Stereobildpaars speichert, kann diese Bedingung widerspiegeln, indem nur Pixelpaare aufgenommen werden, die auf dieser Epipolarlinie liegen. In diesem Kapitel wird ein solches Korrelationsvolumen für RAFT entwickelt und implementiert. Des Weiteren werden verschiedene Lookup-Strategien für dieses Korrelationsvolumen vorgeschlagen.

3.1 Aufbau des Korrelationsvolumens

Das Korrelationsvolumen speichert die visuelle Ähnlichkeit zwischen den Pixeln der beiden Bilder. Während bei RAFT die Ähnlichkeit aller Pixel aus dem ersten Bild zu allen Pixeln aus dem zweiten Bild berechnet wird, ist dies für die Schätzung der Stereodisparität nicht nötig. Da es sich bei den beiden Bildern um ein Stereobildpaar handelt, gilt die epipolare Bedingung aus Kapitel 2.2. Diese besagt, dass die Korrespondenz eines Pixels aus dem ersten Bild nur entlang einer Geraden im zweiten Bild liegen kann, der zugehörigen Epipolarlinie. Im allgemeinen Fall müsste die Fundamentalmatrix für das Stereokamerasystem berechnet werden, um die Epipolarlinie zu einem Pixel im anderen Bild zu finden. Durch die Rektifizierung wird sichergestellt, dass die Epipolarlinien immer horizontal und in beiden Bildern auf der gleichen Höhe verlaufen. Das vereinfacht die Implementierung erheblich. Das Korrelationsvolumen muss daher nur die Korrelation zwischen Pixeln aus den gleichen Pixelzeilen enthalten. Während viele aktuelle Stereokorrespondenzmethoden als Kostenvolumen ein 4D-Konkatenationsvolumen benutzen, um den folgenden Verarbeitungsschritten zu ermöglichen, den Kontext des Bildes zu verstehen [KMD$^+$17, CC18, ZPYT19, ZQY$^+$20], ist dies durch die Architektur von RAFT nicht nötig, da ein extra Kontext-Netzwerk enthalten ist. Stattdessen wird die Ähnlichkeit zwischen Pixeln durch das Skalarprodukt ihrer Merkmale berechnet wie in [LSU16, TD20] vorgeschlagen. Aus den Merkmalen vom Merkmal-Netzwerk $g_\theta(\boldsymbol{I}_1), g_\theta(\boldsymbol{I}_2) \in \mathbb{R}^{H \times W \times D}$ zum Stereobildpaar I_1, I_2 wird ein 3D-Korrelationsvolumen $\boldsymbol{C}(g_\theta(I_1), g_\theta(I_2)) \in \mathbb{R}^{H \times W \times W}$ generiert. Dieses Korrelationsvolumen entsteht durch das Skalarprodukt zwischen allen Merkmalen des ersten Bildes mit den Merkmalen des zweiten Bildes, die sich auf der gleichen Höhe befinden:

```
1 def correlation_volume(feature_map1, feature_map2):
2     feature_map1 = feature_map1.permute(0, 2, 3, 1)
3     batch, height, width, dim_feature = feature_map1.shape
4     feature_map2 = feature_map2.permute(0, 2, 1, 3)
5     corr_volume = torch.matmul(feature_map1, feature_map2)
6     return corr_volume / torch.sqrt(torch.tensor(dim_feature).float())
```

Listing 3.1: Dieses Listing zeigt das Erstellen des Korrelationsvolumens.

$$C_{hij} = \sum_{k=1}^{D} g_\theta(\boldsymbol{I}_1)_{hik} \cdot g_\theta(\boldsymbol{I}_2)_{hjk} \tag{3.1}$$

$$= \Big(g_\theta(\boldsymbol{I}_1)_{hi1}, g_\theta(\boldsymbol{I}_1)_{hi2}, \ldots, g_\theta(\boldsymbol{I}_1)_{hiD}\Big) \Big(g_\theta(\boldsymbol{I}_2)_{hj1}, g_\theta(\boldsymbol{I}_2)_{hj2}, \ldots, g_\theta(\boldsymbol{I}_2)_{hjD}\Big)^\top \tag{3.2}$$

mit der Anzahl der unterschiedlichen Merkmale D. Die Zeile der beiden Bilder wird durch h angegeben. Die Spalte des Referenzbildes wird durch i angegeben, die Spalte des anderen Bildes durch j.

Eine Implementierung in Python mit PyTorch ist in Listing 3.1 zu finden. Die Gleichung 3.2 wird durch eine Matrixmultiplikation implementiert. Die PyTorch Tensoren `feature_map1` und `feature_map2` enthalten alle Merkmale zu einem Batch an Bildpaaren. Die Indizes sind anfangs noch in der Reihenfolge vom Merkmal-Netzwerk - Batch, Nummer des Merkmals, Zeile und Spalte. In Zeile 2 wird die Reihenfolge der Indizes geändert, sodass bei `feature_map1` die Nummer des Merkmals nach hinten geschoben wird. Durch die Permutation ist die Reihenfolge jetzt Batch, Zeile, Spalte und Nummer des Merkmals. In Zeile 4 wird `feature_map2` ähnlich permutiert, nur dass die letzten beiden Dimensionen, die Spalte und die Nummer des Merkmals, vertauscht sind. In der nächsten Zeile wird das Korrelationsvolumen durch das Anwenden der Batch-Matrixmultiplikation `matmul` erstellt. Aufgrund der vorherigen Permutation der Merkmals-Tensoren beschränkt sich die Matrixmultiplikation auf die Spalten und Merkmalnummern. Das Ergebnis ist der Tensor `corr_volume` mit der Form (`batch`, `height`, `width`, `width`). Für das Bildpaar g aus dem Batch ist die Korrelation zwischen den Pixeln (h, i) und (h, j) im Korrelationsvolumen an der Stelle `corr_volume[g,h,i,j]` zu finden. Am Ende erfolgt eine Normalisierung der Korrelationswerte, indem jeder der Werte durch die Wurzel der Anzahl an Merkmalen geteilt wird. Ansonsten würden durch das Skalarprodukt große Werte entstehen, wenn die Anzahl der Merkmale pro Pixel erhöht wird. Im Zuge dieser Arbeit wurde stets eine Anzahl von 256 Merkmalen verwendet, diese Normalisierung ist daher lediglich eine Division durch einen konstanten Faktor.

3.2 Korrelations-Lookup

Für die Schätzung der Disparität benötigt der Update-Operator die Korrelationswerte von den Pixeln, die als korrespondierend infrage kommen. Diese werden dem Update-Operator vom Korrelationsvolumen durch eine Korrelations-Lookup Methode zur Verfügung gestellt. Diese Werte werden dann vom Update-Operator durch mehrere Convolutional Layer aufbereitet, bevor sie von der GRU benutzt werden, um die Disparität anzupassen. Wegen der fixen Eingabegröße der Convolutional Layer muss

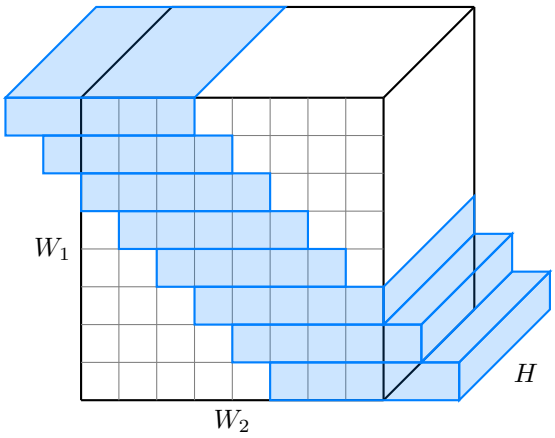


Abbildung 3.1: Eine schematische Darstellung der statischen Nachbarschaft im Korrelationsvolumen. Die Spalte des Pixels im Referenzbild ist entlang W_1 horizontal aufgeführt, die Spalte des Pixels im anderen Bild entlang W_2 vertikal und die Zeile der Pixel in den beiden Bildern entlang der Tiefe H. Die Nachbarschaften um die Pixel des Referenzbildes sind blau markiert.

die Korrelations-Lookup Methode ungeachtet der Größe der Bilder für jeden Punkt der geschätzten Disparität die gleiche Anzahl an Korrelationswerten übergeben. Daher stellt die Korrelations-Lookup Methode für jedes Pixel eine Nachbarschaft mit fester Größe zur Verfügung. Im Folgenden werden verschiedene Varianten zur Implementierung dieser Methode präsentiert.

3.2.1 Statische Nachbarschaft

Die Idee hinter dieser Implementierung ist es, den Suchraum von korrespondierenden Pixeln einzuschränken, indem die maximale Disparität beschränkt wird. Diese Beschränkung führt dazu, dass für ein Pixel aus dem Referenzbild nur die Korrelationen zu Pixeln, deren Abstand kleiner oder gleich der maximalen Disparität ist, betrachtet werden müssen. Für ein Pixel $\boldsymbol{x} = (u, v)$ aus dem Referenzbild und eine maximale Disparität r besteht die Nachbarschaft aus den Korrelationswerten

$$\mathcal{N}_r^{\text{Statisch}}(\boldsymbol{x}) = (C_{v,u,u-r}, \ldots, C_{v,u,u}, \ldots, C_{v,u,u+r}). \tag{3.3}$$

Eine schematische Darstellung dieser Nachbarschaften im Korrelationsvolumen ist in Abbildung 3.1 zu sehen.

Die Implementierung eines solchen Korrelationsvolumens ist in Listing 3.2 zu finden. Ein Korrelationsvolumen wird in RAFT in Form einer Klasse implementiert. Der Konstruktor wird aufgerufen, nachdem die Merkmale der Bildpaare extrahiert wurden. Er bekommt die Merkmale der Bildpaare `feature_map1` und `feature_map2` übergeben. Außerdem wird die Größe der Nachbarschaft in Form des Radius r übergeben.

In Zeile 3 wird zunächst das Korrelationsvolumen erstellt, wie es im vorherigen Kapitel beschrieben wurde. Da die Nachbarschaft unabhängig von der geschätzten Disparität ist, kann sie auch schon im

```
1  class StereoCorrBlockStatic:
2     def __init__(self, feature_map1, feature_map2, radius=60):
3        corr = correlation_volume(feature_map1, feature_map2)
4        batch, h1, w1, w2 = corr.shape
5
6        self.corr_neighborhood = torch.zeros((batch, h1, w1, radius*2+1),
             device=feature_map1.device)
7        corr = F.pad(corr, (radius, radius), 'constant', 0)
8        for x in range(width):
9           self.corr_neighborhood[:, :, x, :] = corr[:, :, x, x:x+2*radius+1]
10
11       self.corr_neighborhood = self.corr_neighborhood.permute(0, 3, 1, 2).contiguous().float()
12
13    def __call__(self, _):
14       return self.corr_neighborhood
```

Listing 3.2: Die Implementierung des Korrelationsvolumens mit statischen Nachbarschaften.

Voraus berechnet und abgespeichert werden. Ein Tensor mit Platz für die Nachbarschaften um jeden Punkt der Referenzbilder des Batches wird in Zeile 6 allokiert und mit Nullen initialisiert.

Für Punkte nahe dem Bildrand sind einige Mitglieder ihrer Nachbarschaft außerhalb des Bildbereichs. Diese Werte werden mit dem Wert 0 aufgefüllt. Das entspricht dem Wert des Skalarprodukts, wenn die Merkmalsvektoren rechtwinklig zueinander sind und daher nicht korrelieren. Um dies zu realisieren, werden an das Korrelationsvolumen links und rechts Nullen angefügt, sodass die Indizes, die normalerweise außerhalb wären, nicht gesondert betrachtet werden müssen.

Dadurch verschieben sich die Indizes von `corr` um den Wert `radius`. Für Spalte `x` des Referenzbilds soll die Nachbarschaft die Werte von Spalte $\texttt{x} - \texttt{radius}$ bis $\texttt{x} + \texttt{radius}$ des zweiten Bildes enthalten. Wegen des Anfügens der Nullen muss auf die Indizes `radius` addiert werden. Daher muss in Zeile 9 die Nachbarschaft von `x` (inkludiert) bis $\texttt{x} + 2 \cdot \texttt{radius} + 1$ (exkludiert) aus `corr` in die Liste der Nachbarschaften `self.corr_neighborhood` kopiert werden.

In Zeile 11 wird die Liste der Nachbarschaften in die vom Update-Operator erwartete Reihenfolge gebracht - Batch, Spalte von Bild 2, Zeile beider Bilder und Spalte von Bild 1. Der Update-Operator ruft die Korrelations-Lookup Methode `__call__` in Zeile 13 auf.

3.2.2 Disparitäts-Nachbarschaft

Ein Nachteil des Korrelations-Lookups mit statischer Nachbarschaft ist, dass im Voraus die maximale Distanz zwischen konjugierten Punkten bekannt sein muss. Wenn der konjugierte Punkt in der ersten Iteration des Update-Operators nicht in der Nachbarschaft enthalten ist, dann wird er aufgrund der festen Position der Nachbarschaft auch in den folgenden Iterationen nicht enthalten sein. Deshalb wird in diesem Kapitel eine alternative Korrelations-Lookup Methode entwickelt, deren Nachbarschaft mithilfe der geschätzten Disparität angepasst wird. Mit der geschätzten Disparität d aus der vorherigen Iteration des Update-Operators wird für ein Pixel $\boldsymbol{x} = (u, v)$ aus dem Referenzbild die Position ausgerechnet, an der sich das korrespondierende Pixel $\boldsymbol{x'}$ im zweiten Bild befindet

$$\boldsymbol{x'} = (u', v) = (u + d(u), v). \tag{3.4}$$

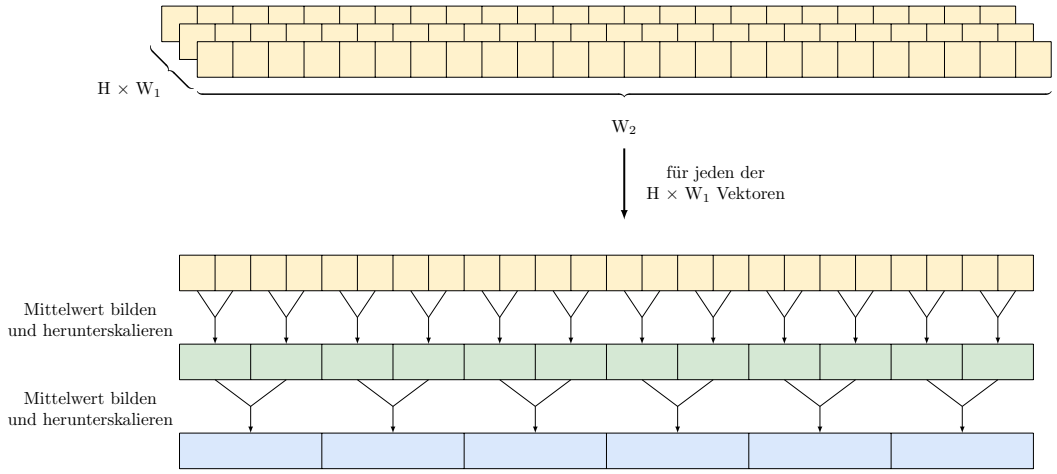


Abbildung 3.2: Beispiel für das Erstellen einer Pyramide vom Korrelationsvolumen. Im hier gezeigten Beispiel wird die Pyramide nur in der W_2 Dimension (x-Achse des konjugierten Pixels) erstellt.

Für den Punkt $\boldsymbol{x}$ wird nun eine Nachbarschaft um den Punkt $\boldsymbol{x'}$ erstellt. Sie enthält die Korrelation zwischen dem Punkt $\boldsymbol{x}$ aus dem Referenzbild und dem Punkt $\boldsymbol{x'}$ sowie alle Punkte mit ganzzahligem Abstand kleiner oder gleich dem Radius r aus dem zweiten Bild

$$\mathcal{N}_r^{\text{Disparität}}(\boldsymbol{x}, \boldsymbol{x'}) = (C_{v,u,u'-r}, \ldots, C_{v,u,u'}, \ldots, C_{v,u,u'+r}). \tag{3.5}$$

Wegen der Addition der Disparität, kann u' nicht ganzzahlig sein. In diesem Fall müssen die Korrelationswerte aus den benachbarten Werten linear interpoliert werden.

Bei dieser Methode des Korrelations-Lookups kann es sinnvoll sein dem Update-Operator zusätzlich Korrelationswerte in gröberen Auflösungen zur Verfügung zu stellen. Dafür wird das Korrelationsvolumen geglättet, indem der Mittelwert von zwei benachbarten Werten gebildet und auf die Hälfte der Größe herunterskaliert wird, sodass eine Gauß-Pyramide entsteht. Dies ist in Abbildung 3.2 illustriert. Auf jeder Stufe dieser Pyramide wird nun eine Nachbarschaft erstellt. Die Anzahl der Punkte in der Nachbarschaft ist für jede Stufe der Pyramide gleich, aber die Schrittweite verdoppelt sich mit jeder Ebene. Wie in Abbildung 3.3 gezeigt, decken die Ebenen mit niedriger Auflösung einen großen Bereich ab, während die Ebenen mit einer hohen Auflösung einen kleinen Bereich abdecken, dafür aber präzisere Korrelationswerte liefern. Dies ermöglicht eine kompakte Beschreibung von großen Nachbarschaften.

In Listing 3.3 ist die Implementierung dieses Korrelations-Lookups zu finden. Dieses enthält zwei Implementierungen bezüglich der Gauß-Pyramide. In der ersten Version wird nur die hinterste Dimension des Korrelationsvolumens herunterskaliert, während die ersten beiden Dimensionen in voller Auflösung zur Verfügung stehen. Der Grund, dies zu tun, ist, dass der Update-Operator die Disparität für ein Bildpaar direkt in voller Auflösung aktualisiert. Durch das Erhaltenbleiben der hohen Auflösung in den ersten beiden Dimensionen stehen genaue Korrelationswerte zu jedem Pixel der Disparität zur Verfügung. So können die Korrelationswerte an Positionen von kleinen Objekten, deren Disparität sich stark von der Disparität ihrer Umgebung unterscheidet, genauer

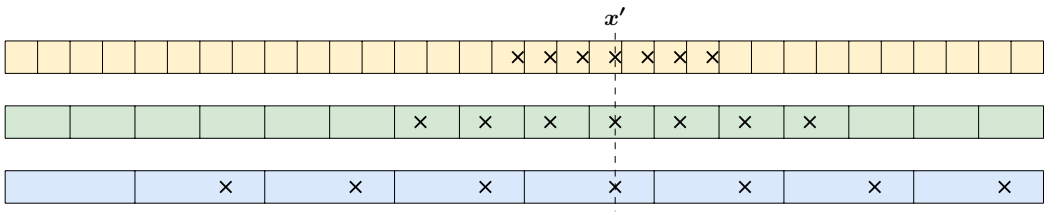


Abbildung 3.3: Zu sehen sind Vektoren aus dem Korrelationsvolumen entlang der letzten Dimension. Darauf sind die Punkte der Nachbarschaft um den konjugierten Punkt x' auf mehreren Ebenen der Pyramide durch das Symbol $\times$ eingezeichnet. Der Korrelationswert dieser Punkte wird aus den beiden benachbarten Werten linear interpoliert.

bestimmt werden. Bei der zweiten Version bleibt die erste Dimension des Korrelationsvolumens in voller Auflösung erhalten und die beiden hinteren Dimensionen werden herunterskaliert. Oft gehören benachbarte Punkte zum selben Objekt und haben daher eine ähnliche Disparität. Daher kann das Bilden des Mittelwertes und das Herunterskalieren der zweiten Dimension für eine höhere Robustheit des Verfahrens sorgen. Die erste Dimension des Korrelationsvolumens entspricht der Höhe und wird in keiner der beiden Versionen herunterskaliert, weil die konjugierten Punkte bei den Stereobildpaaren immer auf derselben Höhe liegen.

Der Parameter `pooling` entscheidet, ob eine 1D Pyramide oder eine 2D Pyramide verwendet wird. Die Version mit einer herunterskalierten Dimension wird in Zeile 14 mit der PyTorch-Funktion `avg_pool1d` implementiert, die Version mit zwei Dimensionen in Zeile 20 durch `avg_pool2d`. Um die Implementierung der Korrelations-Lookup Methode einfacher zu gestalten, werden in der 2D Version die Werte nach dem Herunterskalieren entlang der zweiten Dimension repliziert, sodass die Größe der PyTorch Tensoren auf jeder Ebene der Pyramide übereinstimmt. Dadurch geht zwar die zusätzliche Speichereffizienz der 2D Version verloren, der Speicherverbrauch des Korrelationsvolumens ist dennoch sehr gering verglichen mit dem Gesamtspeicherverbrauch von RAFT. Die Korrelations-Lookup Methode `__call__` bekommt vom Update-Operator die geschätzten Positionen der korrespondierenden Punkte im zweiten Bild x' als `coords` übergeben. Um die Kompatibilität zum Korrelationsvolumen für den optischen Fluss von RAFT beizubehalten, werden hier zwei Werte pro Pixel anstatt einem übergeben. Der erste Wert enthält die geschätzte Disparität, der zweite Wert bleibt unbenutzt. In Zeile 29 bis 31 wird ein Tensor mit den Versatzwerten von $-r$ bis r erstellt. Sie werden in Zeile 37 auf die geschätzten Positionen der korrespondierenden Punkte addiert. Die Korrelationswerte werden mit der Methode `bilinear_sampler` an den zuvor berechneten Positionen linear interpoliert. Da die vorletzte Dimension von `corr` die Größe 1 hat, wird nicht entlang der Höhe interpoliert. Daher interpoliert diese Methode hier nur linear und nicht bilinear. Die Nachbarschaften auf den unterschiedlichen Ebenen der Pyramide werden am Ende aneinandergehängt und an den Update-Operator zurückgegeben.

```
class StereoCorrBlockAdaptive:
   def __init__(self, fmap1, fmap2, num_levels=4, radius=4, pooling="1d"):
      self.num_levels = num_levels
      self.radius = radius
      self.corr_pyramid = []

      corr = StereoCorrBlock.correlation_volume(fmap1, fmap2)
      batch, h1, w1, w2 = corr.shape
      self.corr_pyramid.append(corr.reshape(batch * h1 * w1, 1, 1, w2))

      if pooling == "1d":
         corr = corr.reshape(batch * h1 * w1, 1, w2)
         for i in range(num_levels - 1):
            corr = F.avg_pool1d(corr, 2, stride=2)
            self.corr_pyramid.append(corr.view(batch * h1 * w1, 1, 1, -1))
      else: # 2d pooling
         corr = corr.reshape(batch * h1, w1, w2)
         width_pooled = w1
         for i in range(num_levels - 1):
            corr = F.avg_pool2d(corr, 2, stride=2)
            corr_lvl = corr.repeat_interleave(2**(i+1),dim=1)[:,0:w1,:]
            self.corr_pyramid.append(corr_lvl.reshape(batch * h1 * w1, 1, 1, -1))

   def __call__(self, coords):
      r = self.radius
      coords = coords.permute(0, 2, 3, 1)
      batch, h1, w1, _ = coords.shape

      delta = torch.linspace(-r, r, 2*r+1)
      delta = torch.stack((delta, torch.zeros(2*r+1)), axis=-1)
      delta = delta.view(1, 1, 2*r+1, 2).to(coords.device)

      out_pyramid = []
      for i in range(self.num_levels):
         corr = self.corr_pyramid[i]

         coords_lvl = coords.reshape(batch*h1*w1, 1, 1, 2) / 2**i + delta

         corr = bilinear_sampler(corr, coords_lvl, H=h1//(2**i), W=w1//(2**i))
         out_pyramid.append(corr.view(batch, h1, w1, -1))

      out = torch.cat(out_pyramid, dim=-1)
      return out.permute(0, 3, 1, 2).contiguous().float()
```

Listing 3.3: Die Implementierung des Korrelationsvolumens mit Nachbarschaft um die geschätzte Disparität.

4 Evaluation

In dem vorherigen Kapitel wurde ein Korrelationsvolumen für das Stereokorrespondenzproblem entwickelt. Außerdem wurden verschiedene Möglichkeiten des Korrelations-Lookups für dieses Korrelationsvolumen vorgestellt und implementiert. In diesem Kapitel wird die Leistung der Korrelationsvolumen bezüglich der Schätzung der Disparität getestet und verglichen. Dafür werden Modelle mit dem originalen RAFT Korrelationsvolumen, sowie mit dem Korrelationsvolumen mit statischer Nachbarschaft und dem Korrelationsvolumen mit Disparitäts-Nachbarschaft ohne Pyramide, mit 1D Pyramide und mit 2D Pyramide trainiert. Zunächst werden in diesem Kapitel die verwendeten Datensätze und der Trainingsvorgang thematisiert. Danach werden die trainierten Modelle der verschiedenen Korrelationsvolumen evaluiert.

4.1 Datensätze

Für das Training einer Deep Learning Methode wie RAFT werden große Datensätze benötigt. Im Folgenden werden die für das Training und die Evaluation benutzten Datensätze beschrieben. Zum Trainieren werden die Datensätze Flying Things 3D [MIH+16], Sintel [BWSB12], KITTI 2012 [GLU12] und KITTI 2015 [MG15] benutzt. Evaluiert werden die Modelle mit den Sintel, KITTI 2012 und KITTI 2015 Datensätzen.

Flying Things 3D

Flying Things 3D ist ein großer synthetischer Datensatz für das Training von CNNs. Er besteht aus Alltagsgegenständen, die entlang zufälliger Flugbahnen vor einer texturierten Grundebene durch das Bild fliegen. Der hier genutzte Teil des Datensatzes umfasst 44780 Bildpaare. Die Hälfte dieser Paare entstammt dem Clean Pass und die andere Hälfte dem Final Pass. Der Clean Pass und der Final Pass sind zwei unterschiedliche Renderausführungen, die sich in den im Bild enthaltenen visuellen Effekten unterscheiden. Während der Clean Pass die texturierte Szene mit Ausleuchtung, Licht- und Schatteneffekten enthält, sind beim Final Pass auch Effekte wie Tiefen- und Bewegungsunschärfe oder eine Gammakorrektur enthalten. Der Datensatz enthält für jedes Bildpaar die Ground Truth, die wahre Disparität der beiden Bilder. Flying Things 3D enthält eine Ground Truth mit dem linken Bild als Referenz sowie mit dem rechten Bild als Referenz. Daher bietet der Datensatz insgesamt 89560 Kombinationen aus Bildpaar und Ground Truth.

Sintel

Der Sintel Datensatz besteht aus Szenen eines animierten Kurzfilms. Nachdem er ursprünglich nur für den optischen Fluss verfügbar war, gibt es inzwischen auch eine Version für die Stereokorrespondenz. Zurzeit ist diese als Beta-Version verfügbar. Der Datensatz enthält 1064 Bildpaare aus 23 Szenen. Wie bei Flying Things 3D sind auch bei Sintel ein Clean Pass und ein Final Pass verfügbar. Der Final Pass fügt hier Effekte wie Farbkorrekturen, atmosphärische Effekte, Tiefen- und Bewegungsunschärfe hinzu.

KITTI

Der KITTI Datensatz beinhaltet Straßenaufnahmen eines autonomen Fahrzeugs. Die Szenen stammen von Straßenaufnahmen in der Stadt Karlsruhe, aus ländlichen Gebieten und von der Autobahn. Es existieren zwei Versionen, KITTI 2012 und KITTI 2015, die 194 bzw. 200 Stereobildpaare mit der dazugehörigen Ground Truth zur Verfügung stellen. Zusätzlich wird auch ein Benchmark[1] für die Stereokorrespondenz angeboten. Die Ground Truth wird mithilfe eines Laserscanners erzeugt.

Trainings- und Evaluations-Aufteilung

Die Bilder, die zur Evaluation der verschiedenen Korrelationsvolumen verwendet werden, können nicht für das Training verwendet werden. Daher werden die Datensätze, die sowohl für das Training als auch für die Evaluation benutzt werden, in zwei disjunkte Teile aufgeteilt.

Für KITTI 2012 und KITTI 2015 wird ein Anteil von 80% als Trainingsdaten und die restlichen 20% für die Evaluation benutzt. Konkret werden bei KITTI 2012 die Bilder 0 bis 38 für die Evaluation benutzt und die Bilder 39 bis 193 für das Training. Bei KITTI 2015 sind es die Bilder 0 bis 39 für die Evaluation und die Bilder 40 bis 199 für das Training.

Für Sintel werden die sechs Szenen ambush_2, ambush_6, bamboo_2, cave_4, market_6 und temple_2 für das Training verwendet und die restlichen Szenen für die Evaluation. Diese Aufteilung wird von Zhao *et al.* [ZSD+20] beim Evaluieren ihrer Methode für die Schätzung des optischen Flusses benutzt. Diese Aufteilung führt auch in etwa zu einer Aufteilung von 80% Trainingsbildern und 20% Evaluationsbildern mit 833 Trainings- und 231 Evaluationsbildpaaren.

4.2 Details zur Implementierung

Die Implementierung ist abgesehen von den Korrelationsvolumen weitestgehend gleich zur originalen RAFT [TD20] Methode, die bereits in Kapitel 2.4 erklärt wurde. Im Folgenden werden einige Details der Implementierung genannt, die den Trainingsvorgang betreffen. Wegen der Ähnlichkeit zum

[1] http://www.cvlibs.net/datasets/kitti/eval_stereo.php

originalen RAFT für den optischen Fluss werden dabei dieselben Algorithmen eingesetzt. Diese werden in diesem Kapitel kurz erklärt.

Die Anzahl der Merkmale, die das Merkmal-Netzwerk für jedes 8×8 Feld extrahiert, wird auf 256 festgesetzt. Als Fehlerfunktion wird die exponentiell gewichtete L1-Metrik aus Gleichung 2.8 benutzt.

Als Optimierungsalgorithmus wird AdamW [LH17] eingesetzt. AdamW ist ein Gradientenverfahren, das die Lernrate auf Basis der ersten beiden stochastischen Momente, dem Mittelwert und der nicht zentrierten Varianz, anpasst. Im Gegensatz zu Adam wird bei AdamW der Gewichtsverlust erst nach der Anpassung der Lernrate vorgenommen. Dadurch entstehen Modelle, die besser generalisieren.

Zum Steuern der Lernrate wird die 1cycle Strategie [ST19] angewendet. Ziel dieser Stragegie ist es die Lernrate von einem Anfangswert auf einen maximalen Wert zu erhöhen. Danach wird sie auf einen minimalen Wert abgesenkt bis das Ende des Trainings erreicht ist.

Um sehr große Anpassungen der Gewichte beim Training zu vermeiden, werden die Gradienten auf das Intervall $[-1, 1]$ abgeschnitten. Außerdem wird bei RAFT, wie von Sun *et al.* [SYLK18] vorgeschlagen, der Fehler über die Aktualisierung der Disparität $\Delta \boldsymbol{f}$ zurückgeführt, aber nicht über die vorherige Schätzung $\boldsymbol{f}_i$.

4.3 Trainingsprozess

Nachdem die verwendeten Datensätze und einige Implementierungsdetails beschrieben wurden, wird in diesem Kapitel der Trainingsvorgang erläutert.

Das Training findet in zwei Phasen statt. In der ersten Phase wird für 100 000 Epochen auf dem Flying Things 3D Datensatz trainiert. In Phase zwei wird für weitere 100 000 Epochen auf einer Mischung aus Flying Things 3D und den Trainingsteilen von Sintel und KITTI trainiert. Um keinen der Datensätze zu bevorzugen, sind die Daten aus den Sintel und KITTI Datensätzen mehrfach enthalten, sodass die Anteile gleich groß sind. Beim zufälligen Ziehen von Daten aus dem gemischten Datensatz sind die Wahrscheinlichkeiten ein Bild aus Flying Things 3D, Sintel und KITTI zu bekommen gleich groß. Wegen der langen Trainingszeiten - mehrere Tage für jedes Modell beim Training mit einer Nvidia GeForce RTX 3090 - findet keine weitere Feinabstimmung für die einzelnen Evaluationsdatensätze statt. Die Hyperparameter des Trainingsprozesses sind in Tabelle 4.1 aufgeführt.

4.4 Augmentationen

Augmentationen werden dazu genutzt, um die Schätzungen eines neuronalen Netzes invariant unter einigen Transformationen des Eingabebildes zu machen. Bei der Augmentation werden zufällig räumliche oder farbliche Änderungen am Bild vorgenommen.

Die farblichen Änderungen beinhalten eine Skalierung der Helligkeit, des Kontrasts und der Sättigung um einen zufälligen Faktor zwischen 0,6 und 1,4. Außerdem kann auch der Farbton verändert werden.

	Phase 1	Phase 2
Datensatz	Flying Things 3D	Gemischt
Epochen	100 000	100 000
Batchgröße	6	6
Initiale Lernrate	$1{,}25 \cdot 10^{-4}$	10^{-4}
Gewichtsverlustfaktor	10^{-4}	10^{-5}
γ der Fehlerfunktion Gleichung 2.8	0,8	0,85

Tabelle 4.1: In der Tabelle sind die Trainingsparameter aufgeführt.

Mit einer Wahrscheinlichkeit von 80% werden dieselben farblichen Änderungen an den beiden Bildern vorgenommen. In 20% der Fälle werden unterschiedliche Faktoren verwendet.

Mit einer Wahrscheinlichkeit von 50% werden zusätzlich zu den farblichen Transformationen im zweiten Bild ein oder mehrere viereckige Bereiche einer Größe von bis zu 100×100 Pixeln mit der durchschnittlichen Farbe des Bildes bedeckt. Dadurch wird eine Verdeckung des Bereichs durch andere Objekte simuliert.

Außerdem werden räumliche Augmentationen benutzt. Während die Ground Truth für die bisherigen Transformationen gleich blieb, muss sie für die räumlichen Transformationen mittransformiert werden. In 80% der Fälle wird die Größe des Bildes skaliert. Mit einer Wahrscheinlichkeit von 80% werden dabei unterschiedliche Faktoren in x-Richtung und y-Richtung gewählt, wodurch die Bilder deformiert werden. Die Stärke der Skalierung wird zufällig aus einem Intervall gewählt, das pro Datensatz festgelegt ist. In der ersten Phase des Trainings auf Flying Things 3D liegt der Skalierungsfaktor circa im Bereich 0,75 bis 1,74. In der zweiten Phase werden Faktoren zwischen 0,81 und 1,51 verwendet. Danach wird die Hälfte der Bilder horizontal und ein Zehntel der Bilder vertikal gespiegelt. Zuletzt werden die augmentierten Trainingsbilder auf eine feste Größe beschnitten. Für die erste Trainingsphase beträgt diese Größe 400×720 Pixel und in der zweiten Phase 288×720 Pixel.

4.5 Fehlermetriken

Um die Leistung der trainierten Modelle zu messen, ist eine Metrik notwendig. Die Leistung eines Modells kann durch die von ihm produzierten Disparitäts-Schätzungen gemessen werden. In dieser Arbeit werden zwei Fehlermetriken verwendet, der Endpunktfehler und der D1-Fehler.

Endpunktfehler

Der Endpunktfehler misst den euklidischen Abstand zwischen der geschätzten Disparität $\boldsymbol{d}^e(\boldsymbol{x})$ und der Ground Truth $\boldsymbol{d}^t(\boldsymbol{x})$. Aus diesen Abständen aus einem Bild wird der Durchschnitt berechnet, der durchschnittliche Endpunktfehler (AEE):

$$\text{AEE}(\boldsymbol{d}^e, \boldsymbol{d}^t) = \frac{1}{N} \sum_{\boldsymbol{x}} \|\boldsymbol{d}^e(\boldsymbol{x}) - \boldsymbol{d}^t(\boldsymbol{x})\|_2 \tag{4.1}$$

mit der Gesamtanzahl der Pixel N.

D1-Fehler

Der D1-Fehler wurde von Menze und Geiger [MG15] vorgeschlagen und wird vom KITTI 12 und KITTI 15 Benchmark genutzt. Diese Fehlermetrik gibt den Anteil an fehlerhaften Pixeln in der Disparität an. Als fehlerhaft werden die Pixel eingestuft, an denen die geschätzte Disparität sowohl mehr als 3 Pixel, als auch mehr als 5% von der Ground Truth entfernt ist.

4.6 Auswahl der Parameter der Korrelationsvolumen

Mit dem zuvor beschriebenen Trainingsprozess werden Modelle für die unterschiedlichen Korrelationsvolumen trainiert. Im Folgenden wird die Auswahl der verschiedenen Parameter der Korrelationsvolumen genannt, die trainiert und evaluiert werden.

Als Referenzpunkt dient das originale RAFT Korrelationsvolumen [TD20], das in Kapitel 2.4 beschrieben wurde. Dieses verwendet eine Nachbarschaft mit dem Radius 4 und eine Pyramide mit 4 Ebenen. Weil es für den optischen Fluss konstruiert wurde, nutzt es die epipolare Bedingung nicht aus.

Für das Korrelations-Lookup mit statischer Nachbarschaft wird ein Radius von 61 benutzt. Die Wahl des Radius ist auf Basis der maximalen Disparität geschehen. Die maximale Disparität von Objekten in den Bildpaaren von Sintel beträgt 486,94 Pixel. In den KITTI 2012 und KITTI 2015 Datensätzen ist die maximale Disparität mit 231,61 und 229,96 deutlich geringer. Der Flying Things 3D Datensatz enthält mitunter deutlich größere Disparitäten. Da dieser nicht für die Evaluation verwendet wird, werden diese Disparitäten für die Auswahl des Radius nicht berücksichtigt.

Da die Position der Nachbarschaft bei dieser Korrelations-Lookup Methode in jeder Iteration des Update-Operators gleich ist, wird der Radius so gewählt, dass die maximale Disparität in der Nachbarschaft enthalten ist. Ein Nachbarschaftsradius von 61 führt wegen des Herunterskalierens im Merkmal-Netzwerk zu einer Abdeckung von mindestens $8 \cdot 61 = 488$ Pixeln, genug um die 486,94 Pixel der maximalen Disparität abzudecken. Die Trainingsbilder werden bei der Augmentation auf eine Breite von 720 Pixel zugeschnitten. Die Gesamtbreite der Nachbarschaft mit Radius 61 beträgt $8 \cdot (61 \cdot 2 + 1) = 984$ Pixel. Daher liegt während des Trainings stets ein Teil der Nachbarschaft außerhalb des Bildbereiches. Für Nachbarschaften, deren Zentrum außerhalb der Mitte der Bildzeile liegt, ist trotzdem eines der Enden der Nachbarschaft im Bildbereich. Deshalb sollten auch für die

äußeren Bereiche der Nachbarschaft die Gewichte mit dem Gradientenverfahren gelernt werden können.

Für die Disparitäts-Nachbarschaft auf nur einer Auflösung ohne Pyramide werden verschiedene Radien trainiert und evaluiert. Ein Radius von 60 kann die gesamte Bildbreite der Bilder aus Sintel abdecken. Wie im vorherigen Absatz erklärt, ist dieser Radius so groß, dass die Nachbarschaft den konjugierten Punkt fast immer enthält. Für das Training gilt auch, dass die Nachbarschaft stets teilweise außerhalb des Bildes liegt, das Training aber dennoch funktionieren sollte. Als Vergleich wird ein zweites Modell mit einem Radius von 44 trainiert. Der Radius ist so gewählt, dass die Nachbarschaften kleiner sind als die Breite der Bilder - auch während des Trainings. Wegen der Bildbreite von 720 Pixeln beim Training wird ein Radius von 44 gewählt, da dessen Nachbarschaften eine Gesamtbreite von $8 \cdot (44 \cdot 2 + 1) = 712$ Pixeln haben. Im Vergleich zum Radius von 60 sind bei einem Radius von 44 weniger Punkte von Anfang an in der Nachbarschaft enthalten, dafür ist der Trainingsprozess aber auch weniger bedenklich. Außerdem wird noch ein Modell mit einem Radius von 4 trainiert. Daran lässt sich die Konvergenz sehen, auch wenn die konjugierten Pixel anfangs nicht in der Nachbarschaft enthalten sind.

Bei der Disparitäts-Nachbarschaft mit mehrskaligen Korrelationsvolumen werden Modelle sowohl für die Pyramide mit einer herunterskalierten Dimension erstellt, als auch für die Pyramide mit zwei herunterskalierten Dimensionen. Als Parameter werden für den direkten Vergleich mit dem originalen RAFT Korrelationsvolumen ein Radius von 4 und eine Ebenenanzahl von 4 gewählt. Außerdem werden weitere Modelle mit einem Radius von 8 und 3 Ebenen getestet. Der abgedeckte Bereich beim Lookup ist bei den unterschiedlichen Modellen gleich groß.

4.7 Resultate

Nachdem zuvor die Datensätze, der Trainingsprozess und die Auswahl der getesteten Parameter erklärt wurden, werden in diesem Kapitel die Resultate der Tests auf den Evaluationsdatensätzen vorgestellt.

In Tabelle 4.2 sind die durchschnittlichen Endpunktfehler auf dem Sintel Evaluationsteil zu sehen. Das Korrelationsvolumen mit statischer Nachbarschaft schneidet mit einem Endpunktfehler von 1,042 auf dem Clean Pass und 1,666 auf dem Final Pass am schlechtesten ab. Im Vergleich zum originalen RAFT Korrelationsvolumen ist das eine Verschlechterung um 33% auf dem Clean Pass und um 24% auf dem Final Pass. Bei den Korrelationsvolumen mit Disparitäts-Nachbarschaft ohne Pyramide wird der Endpunktfehler mit zunehmendem Radius niedriger. Am besten ist hier ein Radius von 60 mit einem Fehler von 0,772 und 1,243 auf dem Clean und dem Final Pass. Dies ist eine Verbesserung um 1% und um 7% auf dem Clean bzw. Final Pass gegenüber dem RAFT Korrelationsvolumen. Bei den mehrskaligen Korrelationsvolumen schneidet das Korrelationsvolumen mit 1D Pyramide mit 3 Ebenen und einem Radius von 8 am besten ab. Dieses ist mit einem Fehler von 0,761 und 1,299 auf dem Clean Pass und dem Final Pass jeweils etwa 3% besser als das originale RAFT Korrelationsvolumen.

Für KITTI 2012 und KITTI 2015 sind der Endpunktfehler und der D1-Fehler in Tabelle 4.3 eingetragen. Auf diesen Datensätzen schneidet das Korrelationsvolumen mit statischer Nachbarschaft wieder am schlechtesten ab, wenn auch mit kleinerem Abstand als bei Sintel. Auf diesen Datensätzen

Korrelationsvolumen	Radius	Pyramide	Sintel AEE	
			Clean Pass	Final Pass
Original RAFT	4	4 Ebenen	0,781	1,339
Statische Nachbarschaft	61	-	1,042	1,666
Disparitäts-Nachbarschaft	4	-	0,795	1,375
Disparitäts-Nachbarschaft	44	-	0,777	1,269
Disparitäts-Nachbarschaft	60	-	0,772	**1,243**
Disparitäts-Nachbarschaft	4	4 Ebenen, 1D	0,782	1,279
Disparitäts-Nachbarschaft	4	4 Ebenen, 2D	0,763	1,344
Disparitäts-Nachbarschaft	8	3 Ebenen, 1D	**0,761**	1,299
Disparitäts-Nachbarschaft	8	3 Ebenen, 2D	0,775	1,248

Tabelle 4.2: Endpunktfehler der Korrelationsvolumen auf dem Sintel Evaluationsteil.

Korrelationsvolumen	Radius	Pyramide	KITTI-12		KITTI-15	
			AEE	D1	AEE	D1
Original RAFT	4	4 Ebenen	0,609	1,995	0,787	2,327
Statische Nachbarschaft	61	-	0,689	2,380	0,876	2,945
Disparitäts-Nachbarschaft	4	-	0,652	2,151	0,814	2,467
Disparitäts-Nachbarschaft	44	-	0,605	**1,935**	0,800	2,288
Disparitäts-Nachbarschaft	60	-	0,603	1,936	0,789	**2,280**
Disparitäts-Nachbarschaft	4	4 Ebenen, 1D	0,620	2,028	0,791	2,308
Disparitäts-Nachbarschaft	4	4 Ebenen, 2D	0,618	1,997	**0,773**	2,326
Disparitäts-Nachbarschaft	8	3 Ebenen, 1D	**0,600**	1,943	0,783	2,330
Disparitäts-Nachbarschaft	8	3 Ebenen, 2D	0,623	2,077	0,797	2,382

Tabelle 4.3: Endpunkt- und D1-Fehler der Korrelationsvolumen auf dem KITTI 2012 und KITTI 2015 Evaluationsteil.

schneidet es zwischen 11% und 27% schlechter ab als das RAFT Korrelationsvolumen. Bei KITTI 2012 schneiden die Korrelationsvolumen ohne Pyramide und mit einem Radius von 44 oder 60 und das Korrelationsvolumen mit 1D Pyramide, 3 Ebenen und einem Radius von 8 am besten ab. Diese bringen eine Verbesserung von etwa 1% auf bis zu 0,6 beim Endpunktfehler und etwa 3% beim D1-Fehler auf bis zu 1,935. Bei KITTI 2015 ist einzig das Korrelationsvolumen mit Disparitäts-Nachbarschaft, einem Radius von 4 und 4 2D Pyramidenebenen besser als das originale RAFT Korrelationsvolumen. Die Verbesserungen liegen allerdings nur bei etwa 2% im Endpunktfehler. Das Korrelationsvolumen mit Disparitäts-Nachbarschaft und einem Radius von 60 ohne Pyramide ist zwar mit einem Endpunktfehler von 0,789 um 0,3% schlechter als RAFT, dafür aber beim D1-Fehler mit 2,28 gut 2% besser.

Die Ergebnisse des originalen RAFT Korrelationsvolumens, das die epipolare Bedingung nicht ausnutzt, fallen bezüglich des durchschnittlichen Endpunktfehlers und des D1-Fehlers nur knapp schlechter aus als die des Korrelationsvolumens mit Disparitäts-Nachbarschaft. Bei einer Analyse der Ergebnisse des originalen RAFT Korrelationsvolumens fällt auf, dass die geschätzte Verschiebung in y-Richtung, orthogonal zur Disparitätsrichtung, bei allen Bildern sehr klein ist. Die durchschnittliche Verschiebung in y-Richtung auf den Evaluationsdatensätzen beträgt 0,00118 Pixel. Diese kleine Zahl kann vom Training, das ausschließlich Stereobildpaare und keine Bilder für den optischen Fluss verwendet, verursacht werden. Für alle Bilder des Trainings ist es besser einen y-Wert von 0 zu schätzen als einen anderen Wert.

Beim Betrachten der unterschiedlichen Radien der Disparitäts-Nachbarschaft ohne Pyramide fällt auf, dass der Fehler mit steigendem Radius kleiner wird. Während ein Radius von 4 in allen Tests schlechter abschneidet als das originale RAFT Korrelationsvolumen, liefert ein Radius von 60 mitunter die besten Resultate. Bei den Korrelationsvolumen mit Pyramide zeigt die Version mit 1D Pyramide, 3 Ebenen und einem Radius von 8 sehr gute Ergebnisse. Auf dem Sintel Final Pass liefert die Version mit 2D Pyramide, 3 Ebenen und einem Radius von 8 jedoch bessere Resultate, bei KITTI 2015 die Version mit 2D Pyramide, 4 Ebenen und einem Radius von 4. Das Korrelationsvolumen mit statischer Nachbarschaft schneidet in allen Tests mit Abstand am schlechtesten ab.

4.7.1 Iterationen des Update-Operators

In diesem Kapitel wird der Einfluss der Anzahl der Iterationen des Update-Operators auf den Endpunktfehler analysiert. Abbildung 4.1 enthält Diagramme, die den Endpunktfehler in Abhängigkeit der Anzahl an Iterationen des Update-Operators zeigen. Der Fehler wird nach der ersten und jeder geraden Anzahl an Iterationen gemessen.

Nach der ersten Iteration liegen die Fehler der unterschiedlichen Korrelationsvolumen weit auseinander. Mit steigender Anzahl an Iterationen rücken die Endpunktfehler des originalen RAFT Korrelationvolumens und die Endpunktfehler der Korrelationsvolumen mit Disparitäts-Nachbarschaft Lookup nah zusammen. Das Korrelationsvolumen mit statischer Nachbarschaft beginnt mit einem niedrigen Fehlerwert, fällt aber bei zusätzlichen Iterationen gegenüber den anderen Korrelationsvolumen wegen den deutlich kleineren Verbesserungen ab. Wegen des Korrelations-Lookups, das bei der statischen Nachbarschaft in jeder Iteration die gleichen Werte enthält, können zusätzliche Iterationen weniger gut genutzt werden. Selbst wenn die konjugierten Punkte immer in der Nachbarschaft enthalten sind, gibt diese Lookup Methode keine Auskunft über Korrelationswerte an Subpixelpositionen.

4.7.2 Qualitative Analyse

Die Abbildungen 4.2 bis 4.6 sind Visualisierungen der Disparität und des absoluten Endpunktfehlers für einige Korrelationsvolumen. Visualisiert werden die Ergebnisse des RAFT Korrelationsvolumens, der Disparitäts-Nachbarschaft ohne Pyramide mit einem Radius von 60 und der Disparitäts-Nachbarschaft mit 1D Pyramide, einem Radius von 8 und 3 Ebenen. Diese weisen nach Tabelle 4.2 und 4.3 niedrige

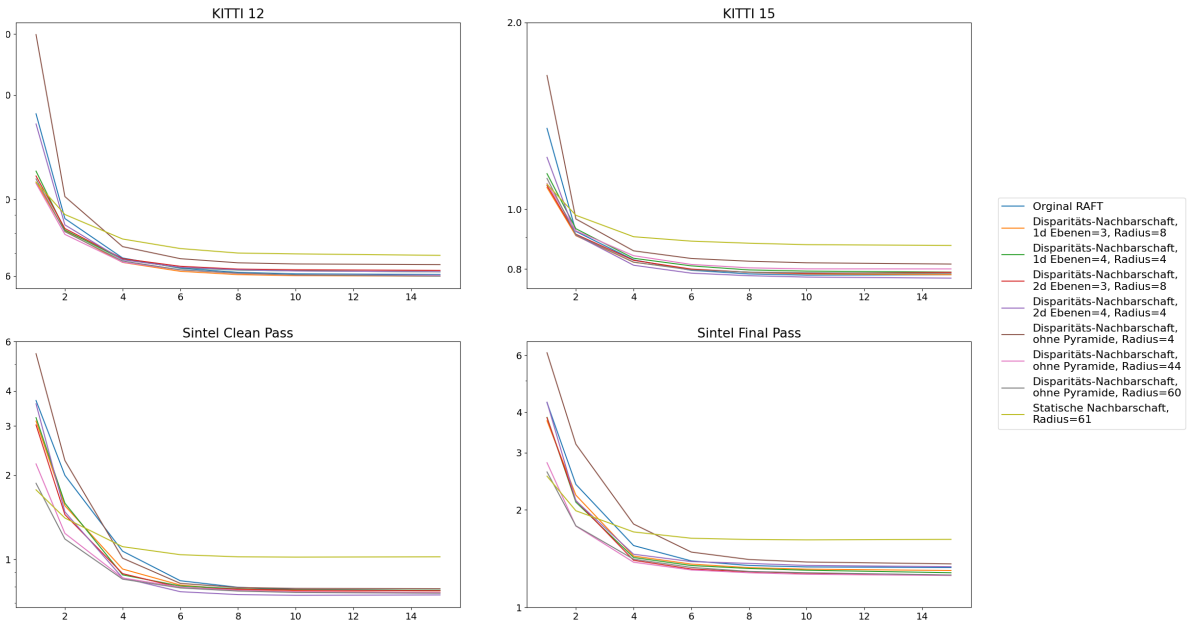


Abbildung 4.1: Der Endpunkfehler der Korrelationsvolumen nach unterschiedlichen Anzahlen an Iterationen des Update-Operators.

Fehlerwerte auf. Außerdem werden auch die Ergebnisse des schlecht abschneidenden Korrelationsvolumens mit statischer Nachbarschaft visualisiert.

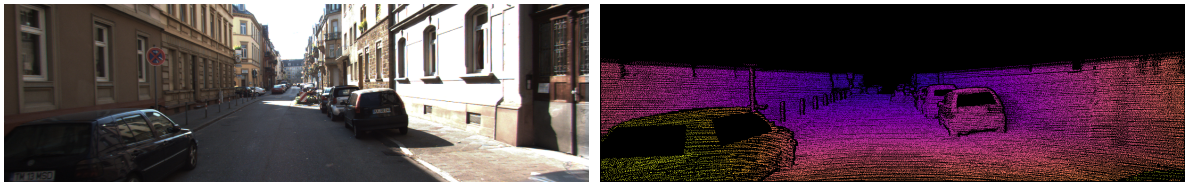
Die Disparität wird auf einer Skala von gelb über violett nach blau visualisiert. Gelbe Punkte haben dabei eine hohe Disparität und eine niedrige Tiefe, blaue Punkte eine niedrige Disparität und eine hohe Tiefe. Der absolute Endpunktfehler wird in Graustufen dargestellt. Fehler von 0 werden in Schwarz dargestellt, Fehler zwischen 0 und 16 in Graustufen und Fehler von mindestens 16 in Weiß. Fehler bei einem Pixel, dessen konjugierter Pixel im zweiten Bild außerhalb des Bildbereiches liegt oder von einem anderen Objekt verdeckt ist, sind rot markiert. Im Sintel Datensatz sind Masken, die diese Pixel markieren, enthalten. In den KITTI Datensätzen werden nur Pixel berücksichtigt, deren konjugierte Pixel im zweiten Bild außerhalb des Bildbereiches liegen. Deshalb werden für KITTI die verdeckten Bereiche approximiert. Bei dieser Approximation wird die Vorwärts- und Rückwärtsdisparität miteinander verglichen. Stimmt die Disparität vom linken zum rechten Bild mit der Disparität des konjugierten Punktes vom rechten zum linken Bild überein, ist der Punkt nicht verdeckt. Stimmen die Disparitäten nicht überein, ist der Punkt vermutlich von einem anderen Objekt verdeckt, das sich zu einer anderen Position bewegt. Um diesen Vorwärts-/Rückwärtstest zu nutzen, wird die Rückwärtsdisparität geschätzt. Dafür wird das auf der Vorwärtsdisparität bezüglich des D1-Fehlers am besten abschneidende Korrelationsvolumen genutzt (siehe Tabelle 4.3). Für KITTI 2012 ist das das Korrelationsvolumen mit einem Radius von 44, für KITTI 2015 das mit einem Radius von 60. Die Ground Truth wird nach dem oben erklärten Verfahren mit der geschätzten Rückwärtsdisparität verglichen. Daraus werden Masken mit den verdeckten Bereichen erstellt, die bei den Visualisierungen benutzt werden.

In den folgenden Visualisierungen sind Beispiele mit den typischen Fehlern zu sehen. In den folgenden Abschnitten werden mögliche Ursachen für diese beschrieben. Die KITTI Datensätze enthalten oftmals sehr schmale Objekte, die Fehler bei der Disparitätsschätzung verursachen. In Abbildung 4.2 befinden sich auf dem linken Gehsteig Pfosten und in Abbildung 4.4 ein Zaun zwischen den Fahrbahnrichtungen. Diese werden zwar in der Disparität angedeutet, aber nicht scharf von der Umgebung getrennt. Beim Betrachten des Endpunktfehlers wird ersichtlich, dass die Disparität dieser Objekte und die Disparität der Objekte in der unmittelbaren Umgebung Fehler enthalten. Ein möglicher Grund für diese Fehler ist die gröbere Auflösung, 1/8 der Auflösung des Bildes, beim Korrelationsvolumen und beim Update-Operator beim Schätzen der Disparität. Wegen der geringen Breite dieser Objekte, oft unter 8 Pixeln, enthält das zugehörige Merkmal oft Teile des Hintergrunds. Für dieses Merkmal, das mehrere Objekte beinhaltet, ist die Disparität schwer bestimmbar, da die verschiedenen Objekte eine unterschiedliche Disparität haben. Abhilfe könnte eine Verringerung der Schrittweite und der Größe der Merkmale schaffen. Dadurch sind weniger Objekte in einem Merkmal enthalten, allerdings erhöht sich im Gegenzug die Rechenzeit, da deutlich mehr Merkmale einander zugeordnet werden müssen.

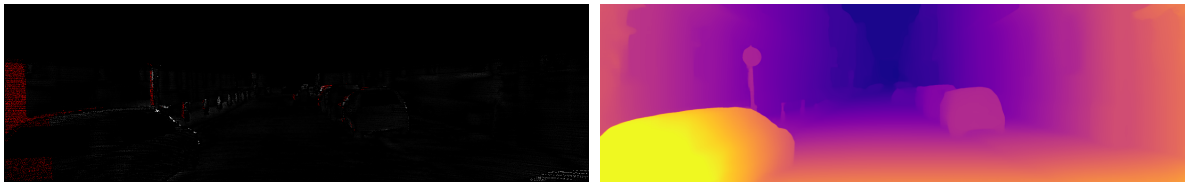
Eine weitere Fehlerregion sind Büsche und Bäume, wie auch in Abbildung 4.3 zu sehen ist. Für diese ist es auf nahe Distanz schwierig die Disparität zu schätzen, da die Büsche und Bäume nicht immer deckend sind. Die unterschiedlichen Tiefen der Objekte an einer Stelle machen es schwer die richtige Disparität zu schätzen.

In den Abbildungen 4.4 und 4.5 sind Fehler im Bereich des Himmels zu sehen. Der KITTI Datensatz enthält keine Ground Truth im Bereich des Himmels. Beim Betrachten der Disparität in Abbildung 4.4 im Bereich des Himmels fällt auf, dass dieser in einem violetten Farbton eingefärbt ist. Da der Himmel sehr weit von der Kamera entfernt liegt, ist die Disparität sehr klein und er müsste in Blau eingefärbt sein. In Abbildung 4.5 treten auch Fehler am Himmel um den fliegenden Vogel herum auf. Diese Fehler könnten durch die kleinen Farbänderungen und fehlenden Ecken oder Kanten im Bereich des Himmels entstehen. Durch diese visuelle Ähnlichkeit aller Punkte des Bereichs ist es schwer den zugehörigen Punkt zu bestimmen. Daher kann die Disparität hier nicht genau bestimmt werden.

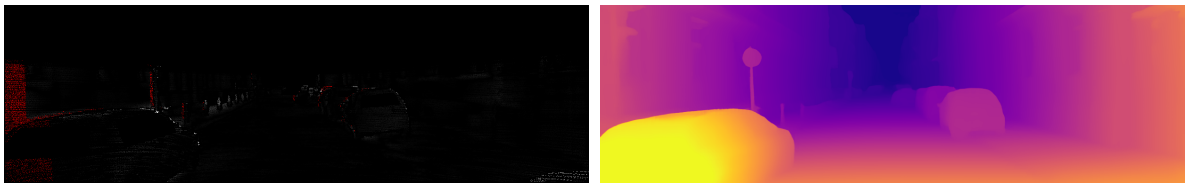
Beim Sintel Final Pass ist der Fehler deutlich größer als beim Clean Pass. Eine Ursache dafür ist die simulierte Bewegungsunschärfe des Final Passes. Wie in Abbildung 4.6 zu sehen ist, lässt diese Objekte, die sich schnell bewegen, in den Bildern unscharf und größer, als sie tatsächlich sind, erscheinen. Dadurch werden die schnellen Objekte, wie der Speer in der Abbildung, zu groß oder gar nicht erfasst. Durch den großen Unterschied in der Disparität von einem sich schnell bewegenden Objekt im Vordergrund und dem Hintergrund kann so der Endpunktfehler des Bildes stark beeinträchtigt werden.



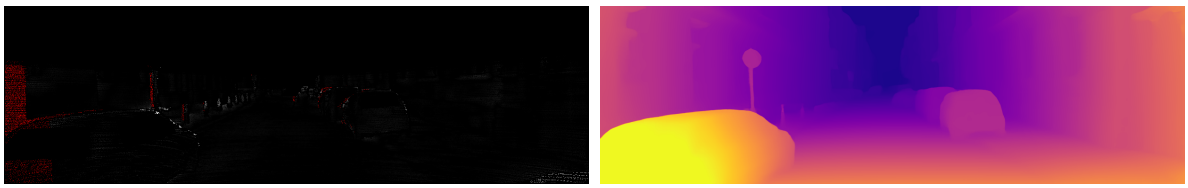
(a) Referenzbild und Ground Truth Disparität



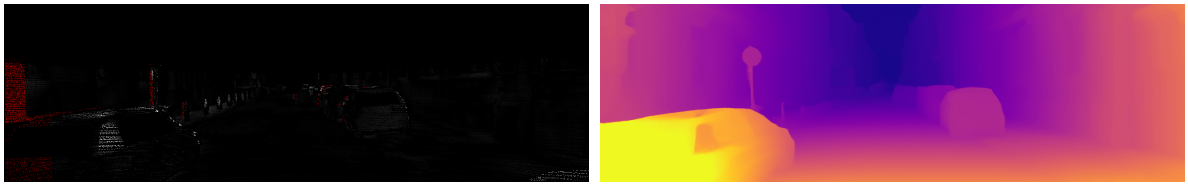
(b) RAFT Korrelationsvolumen



(c) Disparitäts-Nachbarschaft mit 1D Pyramide, 3 Ebenen und einem Radius von 8

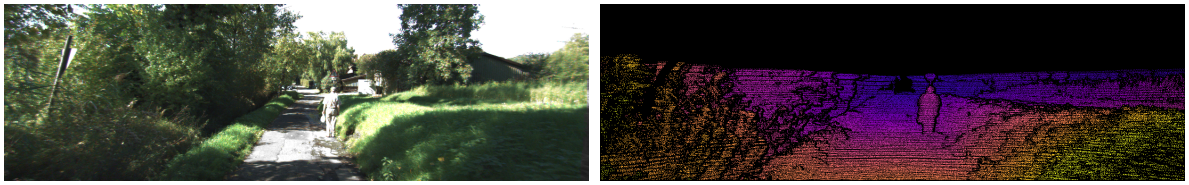


(d) Disparitäts-Nachbarschaft mit einem Radius von 60

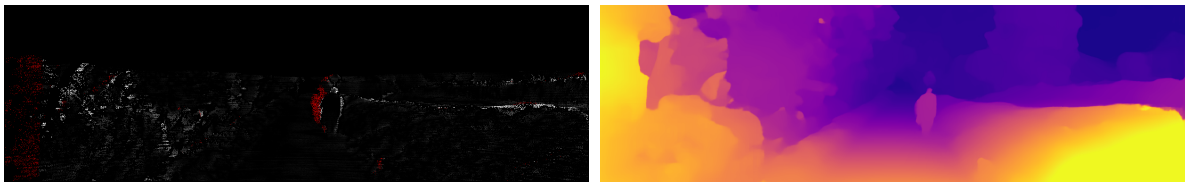


(e) Statische Nachbarschaft mit einem Radius von 61

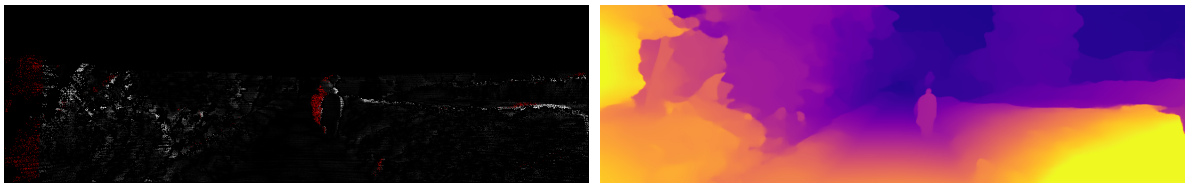
Abbildung 4.2: Visualisierung der Disparität in der linken Spalte und des Endpunktfehlers in der rechten Spalte von KITTI 2012 Bild 15 für unterschiedliche Korrelationsvolumen. Während die Korrelationsvolumen (b) bis (d) sehr ähnliche Disparitäten liefern, hat (e) Probleme bei der Schätzung der Disparität für das Auto unten links.



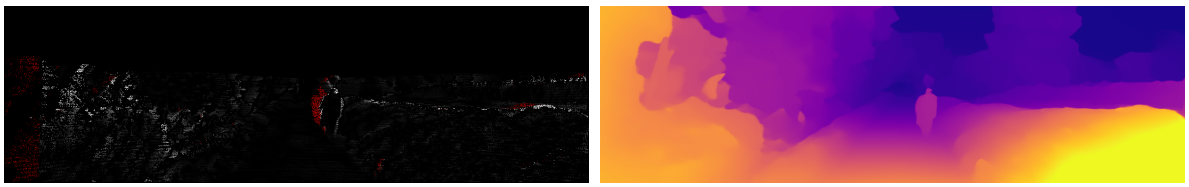
(a) Referenzbild und Ground Truth Disparität



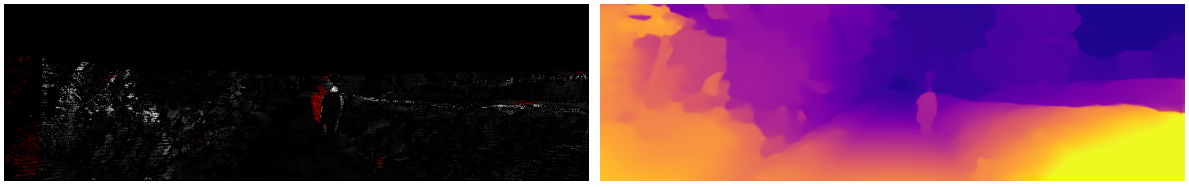
(b) RAFT Korrelationsvolumen



(c) Disparitäts-Nachbarschaft mit 1D Pyramide, 3 Ebenen und einem Radius von 8

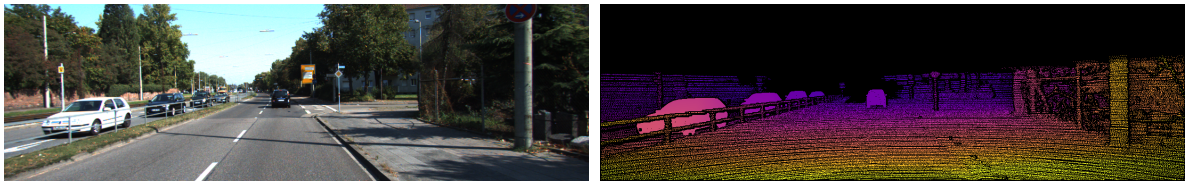


(d) Disparitäts-Nachbarschaft mit einem Radius von 60

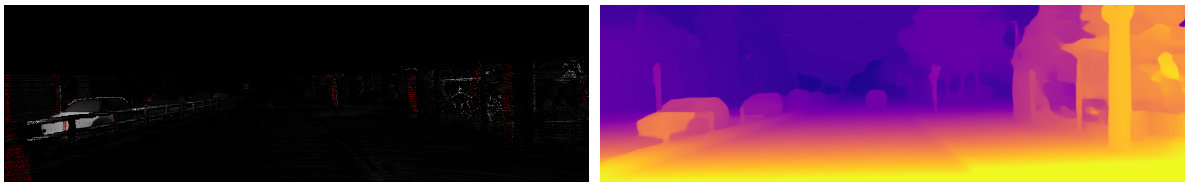


(e) Statische Nachbarschaft mit einem Radius von 61

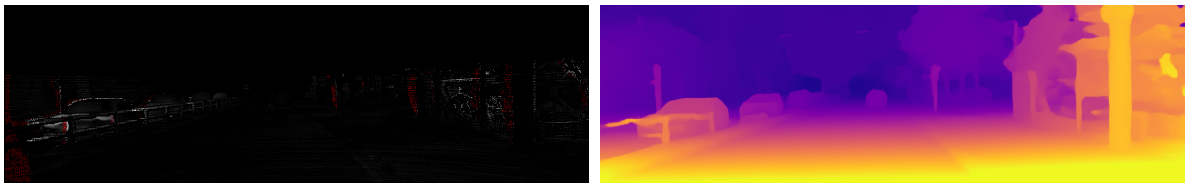
Abbildung 4.3: Visualisierung der Disparität in der linken Spalte und des Endpunktfehlers in der rechten Spalte von KITTI 2012 Bild 36 für unterschiedliche Korrelationsvolumen. Die Schätzung der Disparität bei Büschen und Bäumen ist bei allen Korrelationsvolumen ungenau.



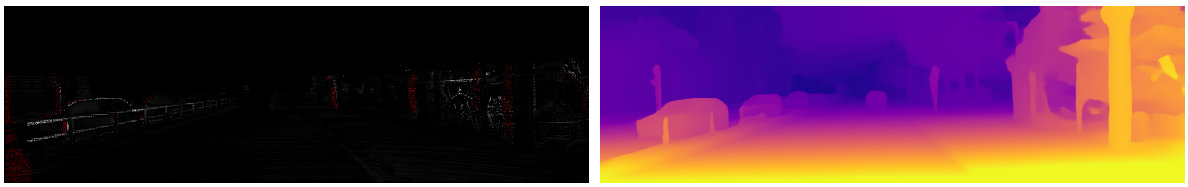
(a) Referenzbild und Ground Truth Disparität



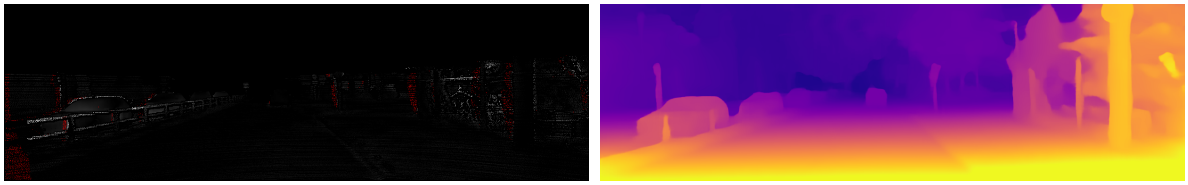
(b) RAFT Korrelationsvolumen



(c) Disparitäts-Nachbarschaft mit 1D Pyramide, 3 Ebenen und einem Radius von 8

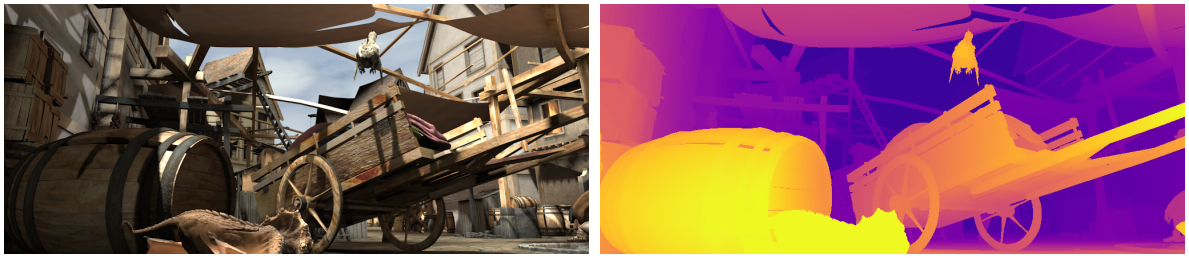


(d) Disparitäts-Nachbarschaft mit einem Radius von 60

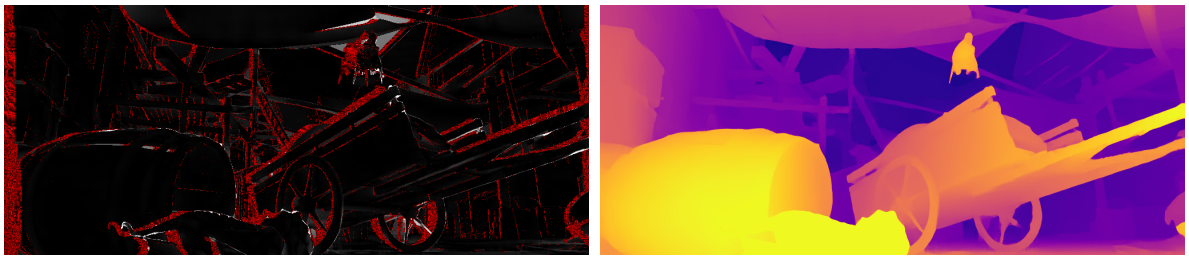


(e) Statische Nachbarschaft mit einem Radius von 61

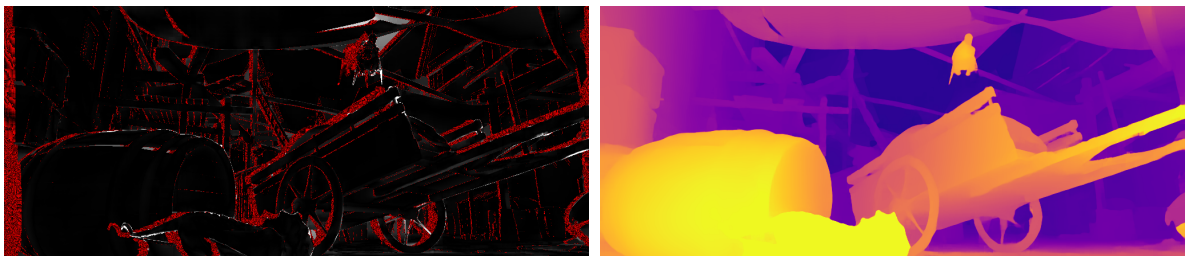
Abbildung 4.4: Visualisierung der Disparität in der linken Spalte und des Endpunktfehlers in der rechten Spalte von KITTI 2015 Bild 25 für unterschiedliche Korrelationsvolumen. Der Zaun, der die beiden Fahrtrichtungen trennt, wird von keinem Modell richtig erkannt.



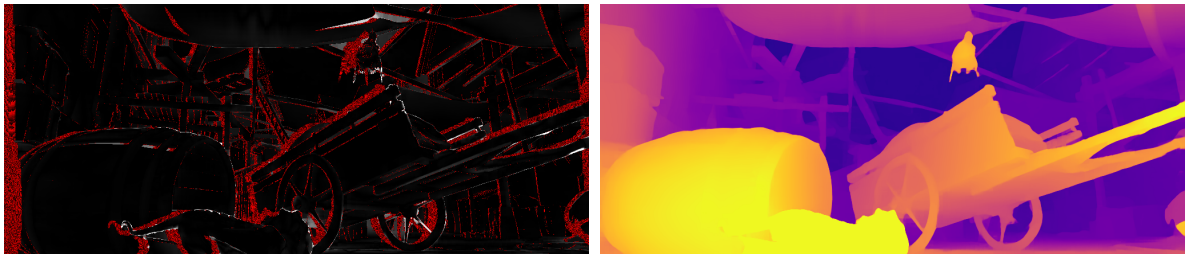
(a) Referenzbild und Ground Truth Disparität



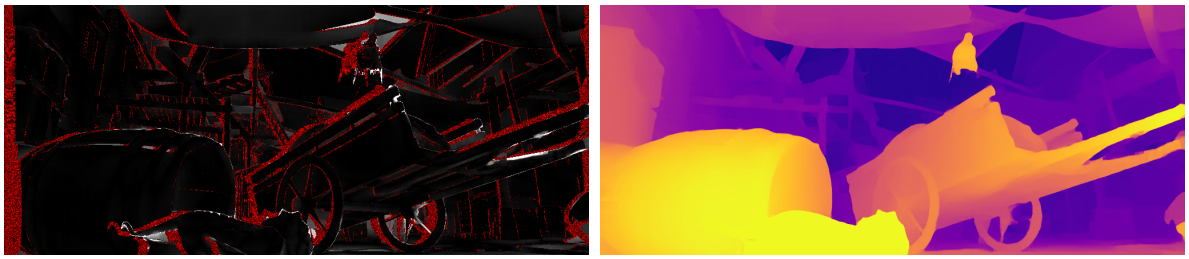
(b) RAFT Korrelationsvolumen



(c) Disparitäts-Nachbarschaft mit 1D Pyramide, 3 Ebenen und einem Radius von 8

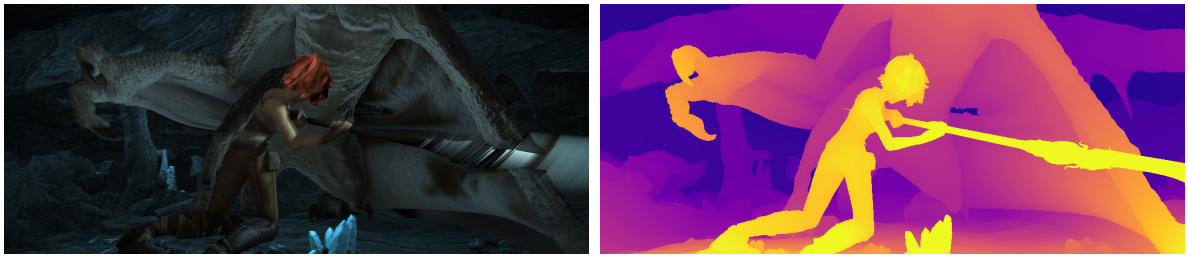


(d) Disparitäts-Nachbarschaft mit einem Radius von 60

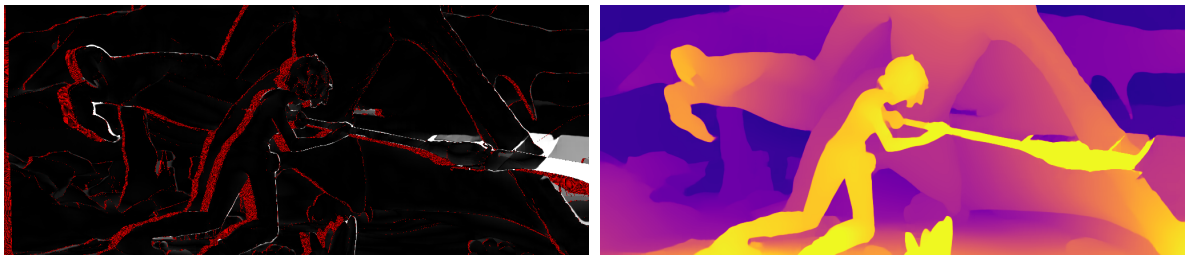


(e) Statische Nachbarschaft mit einem Radius von 61

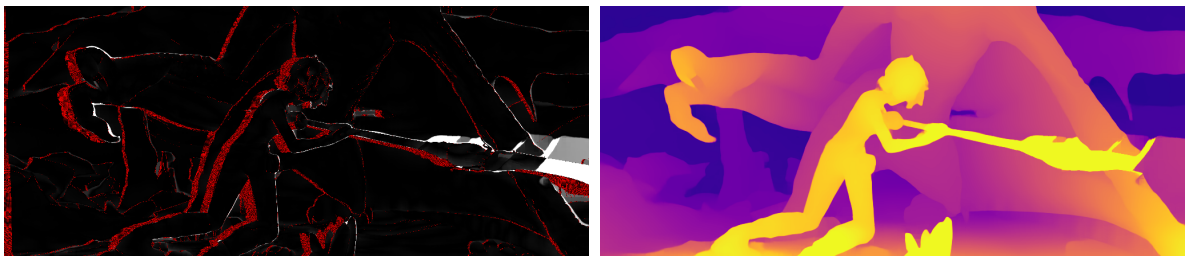
Abbildung 4.5: Visualisierung der Disparität in der linken Spalte und des Endpunktfehlers in der rechten Spalte von dem Clean Pass der Sintel `market_6` Szene Bild 16 für unterschiedliche Korrelationsvolumen. Am Himmel neben dem Vogel, oben im Bild, wird die Disparität nicht korrekt erkannt.



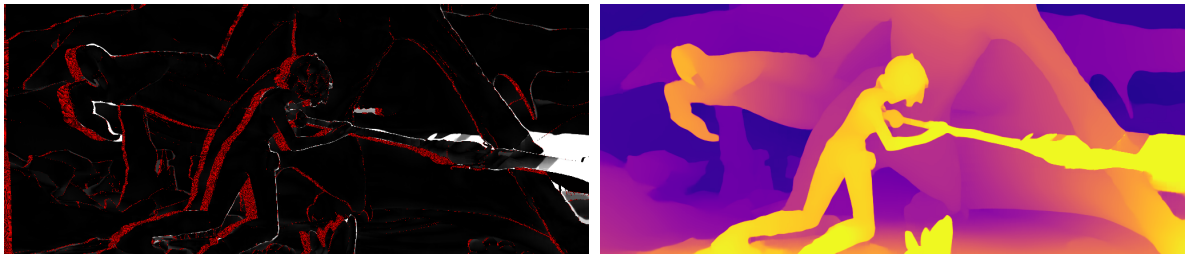
(a) Referenzbild und Ground Truth Disparität



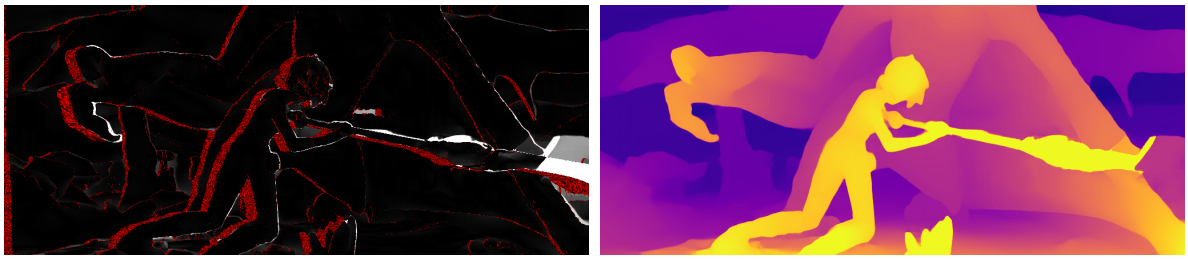
(b) RAFT Korrelationsvolumen



(c) Disparitäts-Nachbarschaft mit 1D Pyramide, 3 Ebenen und einem Radius von 8



(d) Disparitäts-Nachbarschaft mit einem Radius von 60



(e) Statische Nachbarschaft mit einem Radius von 61

Abbildung 4.6: Visualisierung der Disparität in der linken Spalte und des Endpunktfehlers in der rechten Spalte von dem Final Pass der Sintel cave_4 Szene Bild 10 für unterschiedliche Korrelationsvolumen. Durch die Bewegungsunschärfe wird der vordere Teil des Speeres zu groß oder gar nicht erfasst.

5 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es die RAFT Methode zu modifizieren, um das Stereokorrespondenzproblem besser zu lösen. Während die Position des korrespondierenden Pixels beim optischen Fluss beliebig ist, lässt sich der Suchraum bei Stereobildern auf die Epipolarlinie einschränken.

Um sich diese Einschränkung zunutze zu machen, wurden Korrelationsvolumen entwickelt und in die RAFT Pipeline integriert. In diesen Korrelationsvolumen sind nur die Korrelationswerte von Pixeln auf der Epipolarlinie enthalten. Für diese Korrelationsvolumen wurden zwei Lookup-Strategien getestet, die statische Nachbarschaft und die Disparitäts-Nachbarschaft. Die statische Nachbarschaft enthält die Korrelationswerte für Pixel entlang der Epipolarlinie, die höchstens einen im Voraus festgelegten Wert auseinander liegen. Diese Nachbarschaften sind für ein Bildpaar unabhängig von der geschätzten Disparität immer identisch. Bei der Disparitäts-Nachbarschaft sind die Korrelationswerte, die zum Aktualisieren der Disparität benutzt werden, von der bisher geschätzten Disparität abhängig. Die Nachbarschaft enthält die Korrelationswerte entlang der Epipolarlinie um das geschätzte konjugierte Pixel. Außerdem wurden Varianten getestet, die eine Gauß-Pyramide benutzen, um die Korrelationswerte in verschiedenen Auflösungen bereitzustellen.

Die Korrelationsvolumen wurden auf den KITTI 2012, KITTI 2015 und Sintel Datensätzen evaluiert. Dafür wurden Modelle mit verschiedenen Parametern für den Nachbarschaftsradius und die Gauß-Pyramide mit Stereobildpaaren trainiert. Das Korrelationsvolumen mit statischer Nachbarschaft liefert auf allen Datensätzen schlechtere Ergebnisse als das originale RAFT Korrelationsvolumen bezüglich des Endpunktfehlers und des D1-Fehlers. Die Korrelationsvolumen mit Disparitäts-Nachbarschaft erzeugen bessere Ergebnisse als das originale RAFT Korrelationsvolumen bezüglich des Endpunktfehlers und des D1-Fehlers. Die Versionen mit und ohne Gauß-Pyramide erzielen ähnlich gute Resultate. Welches der beiden besser abschneidet, ist von dem Datensatz und der Fehlermetrik abhängig.

Bei der qualitativen Analyse hat sich gezeigt, dass Fehler durch leicht verschobene Objektgrenzen, schmale Objekte, monotone Bereiche und Bewegungsunschärfe entstehen.

In künftigen Arbeiten wäre es interessant den Einfluss der Merkmale auf die Disparität zu untersuchen. Es könnten andere Merkmal-Netzwerke getestet werden, die mit Bildbereichen mit anderen Größen arbeiten, als die bei RAFT benutzten 8×8 Pixel. Außerdem könnten auch Merkmale mit mehreren Größen verwendet werden.

Eventuell wäre es auch interessant die Parameter des Korrelationsvolumens mit Disparitäts-Nachbarschaft ausgiebiger zu testen. Dabei könnten möglicherweise bessere Ergebnisse erzielt werden.

Des Weiteren wäre ein Vergleich der hier entwickelten Methode mit anderen Methoden für das Stereokorrespondenzproblem, die auf dem aktuellen Stand der Technik sind, spannend.

Literaturverzeichnis

[BWSB12] D. J. Butler, J. Wulff, G. B. Stanley, M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), Herausgeber, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, S. 611–625. Springer-Verlag, 2012. (Zitiert auf Seite 29)

[CC18] J.-R. Chang, Y.-S. Chen. Pyramid Stereo Matching Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018. (Zitiert auf den Seiten 12 und 21)

[GLU12] A. Geiger, P. Lenz, R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012. (Zitiert auf Seite 29)

[GYY+19] X. Guo, K. Yang, W. Yang, X. Wang, H. Li. Group-wise correlation stereo network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 3273–3282. 2019. (Zitiert auf Seite 12)

[Hir07] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. (Zitiert auf Seite 11)

[KMD+17] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision*, S. 66–75. 2017. (Zitiert auf den Seiten 12 und 21)

[KRS96] A. Koschan, V. Rodehorst, K. Spiller. Color stereo vision using hierarchical block matching and active color illumination. In *Proceedings of 13th International Conference on Pattern Recognition*, Band 1, S. 835–839. IEEE, 1996. (Zitiert auf Seite 11)

[LH17] I. Loshchilov, F. Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. (Zitiert auf Seite 31)

[LSU16] W. Luo, A. G. Schwing, R. Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 5695–5703. 2016. (Zitiert auf Seite 21)

[MG15] M. Menze, A. Geiger. Object Scene Flow for Autonomous Vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015. (Zitiert auf den Seiten 29 und 33)

[MIH+16] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, T. Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. URL http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16. ArXiv:1512.02134. (Zitiert auf Seite 29)

[MP76] D. Marr, T. Poggio. Cooperative computation of stereo disparity. *Science*, 194(4262):283–287, 1976. (Zitiert auf Seite 11)

[ST19] L. N. Smith, N. Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, Band 11006, S. 1100612. International Society for Optics and Photonics, 2019. (Zitiert auf Seite 31)

[SYLK18] D. Sun, X. Yang, M.-Y. Liu, J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, S. 8934–8943. 2018. (Zitiert auf Seite 31)

[TD20] Z. Teed, J. Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *Proc. European Conference on Computer Vision (ECCV)*, LNCS 12347, S. 402–419. Springer, 2020. (Zitiert auf den Seiten 11, 12, 17, 19, 21, 30 und 33)

[ZPYT19] F. Zhang, V. Prisacariu, R. Yang, P. H. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, S. 185–194. 2019. (Zitiert auf den Seiten 12 und 21)

[ZQY+20] F. Zhang, X. Qi, R. Yang, V. Prisacariu, B. Wah, P. Torr. Domain-invariant stereo matching networks. In *European Conference on Computer Vision*, S. 420–439. Springer, 2020. (Zitiert auf den Seiten 12 und 21)

[ZSD+20] S. Zhao, Y. Sheng, Y. Dong, E. I.-C. Chang, Y. Xu. MaskFlownet: Asymmetric Feature Matching With Learnable Occlusion Mask. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 6277–6286. 2020. doi:10.1109/CVPR42600.2020.00631. (Zitiert auf Seite 30)

[ZW94] R. Zabih, J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *European conference on computer vision*, S. 151–158. Springer, 1994. (Zitiert auf Seite 11)

Erklärung

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

Ort, Datum, Unterschrift