HAW HAMBURG

BACHELOR THESIS
Tom Rathjens

Neuronale Tiefenbildrekonstruktion anhand von Stereobildpaaren

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Tom Rathjens

Neuronale Tiefenbildrekonstruktion anhand von Stereobildpaaren

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Technische Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter: Prof. Dr. Philipp Jenke

Eingereicht am: 08. Juni 2022

Tom Rathjens

Thema der Arbeit

Neuronale Tiefenbildrekonstruktion anhand von Stereobildpaaren

Stichworte

Tiefenbilder, Stereobilder, Neuronale Netze, Faltungsnetzwerke

Kurzzusammenfassung

Tiefeninformationen aus Bildern zu ermitteln ist für den Menschen eine einfache und unterbewusste Aufgabe. Mit seinen Augen und seinen Erfahrungen erhält er einen räumlichen Eindruck von Bildern. Im Bereich des Computer Visions wurde die Aufgabe der Tiefenbildgenerierung aus Stereobildpaaren traditionell mit rein mathematischen Modellen und Algorithmen gelöst. Durch die Entwicklung neuronaler Netze ergeben sich neue Lösungsmöglichkeiten, die an das menschliche Sehen angelehnt sind. Neuronale Netze können trainiert und so Erfahrungen generiert werden. Diese Erfahrungen können für die Extraktion von Tiefeninformationen nach dem Vorbild des Menschen genutzt werden. Aus diesen Tiefeninformationen können Tiefenbilder mit einem räumlichen Eindruck der Ausgangsbilder generiert werden. In den vergangenen Jahren wurden überwiegend Verfahren mit mehreren kombinierten neuronalen Netzen verwendet. In dieser Arbeit wird ein neuer Ansatz namens SS-CNN mit einem einzigen neuronalen Netz untersucht. Für die Untersuchung wird diese neue Architektur vorgestellt und eine Einordnung der erzielten Ergebnisse im Vergleich zu anderen Verfahren vorgenommen. Zusätzlich wird der Einfluss und die Möglichkeiten der Nachbearbeitung der Ergebnisse des neuronalen Netzes untersucht.

Tom Rathjens

Title of Thesis

Neural depth image reconstruction with stereo images

Keywords

Depth Images, Stereo Images, Neural Networks, Convolutional Neural Networks

Abstract

Determining depth information from images is a simple and subconscious task for humans. With his eyes and his experiences he gets a spatial impression of images. In the field of computer vision, the task of depth image generation from stereo image pairs is traditionally solved using purely mathematical models and algorithms. The development of neural networks opens up new solution possibilities that are based on human vision. Neural networks can be trained to generate experience. These experiences can be used to extract depth information along human lines. From this depth information, depth images with a spatial impression of the original images can be generated. In recent years, methods with multiple combined neural networks have been predominantly used. In this paper, a new approach called SS-CNN with a single neural network is investigated. For the investigation, this new architecture is presented and a classification of the obtained results in comparison to other methods is made. In addition, the influence and the possibilities of the post-processing is investigated.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Zeit herrscht ein steigender Bedarf an autonomen Systemen. Roboter und Autos sollen sich möglichst autonom fortbewegen oder Aufgaben lösen. Ein Roboter, der sich durch einen Raum voller Hindernisse bewegt oder ein Auto, welches sich im Straßenverkehr fortbewegt, muss seine Umgebung auswerten und dementsprechend agieren können. In diesem Zusammenhang spielen Kameras eine große Rolle im Sammeln von Umgebungsdaten. Aus den gesammelten zweidimensionalen Informationen müssen dreidimensionale Tiefeninformationen in Form von Tiefenbildern generiert werden. Diese Tiefenbilder ermöglichen eine Navigation und/oder räumliche Auswertung der Umgebung.

In der Vergangenheit wurden solche Problemstellungen im Computer Vision mit rein mathematischen Modellen und fest definierten Algorithmen gelöst. Durch die Entwicklung neuronaler Netze haben sich neue Möglichkeiten in der Auswertung solcher Informationen ergeben. Angelehnt an das menschliche Stereosehen in Verbindung mit Erfahrungen können neuronale Netze konstruiert werden, die an Stelle der fest definierten Algorithmen treten. Diese neuronalen Netze verarbeiten die gewonnen zweidimensionalen Informationen anhand von gelernter Erfahrung und ermöglichen die Generierung der notwendigen räumlichen Informationen.

1.2 Zielsetzung

Im Rahmen dieser Arbeit wird ein neuer Ansatz zur Erstellung von Tiefenbildern aus Sterobildpaaren mittels eines neuronalen Netzes untersucht. Im Gegensatz zur üblicherweise genutzten siamesischen Netzwerkstruktur mit zwei oder drei neuronalen Netzen und nachgelagertem Postprocessing, verwendet dieser Ansatz ein einziges neuronales Netz mit

nachgelagertem Postprocessing. Dieser Ansatz verspricht durch seine einfache Struktur eine Reduktion des benötigten Speicherplatzes und ermöglicht die Ausführung auf einem einfachen Computer.

Inhalt dieser Arbeit soll die Umsetzung und Modifikation der vorgestellten Systemarchitektur der Veröffentlichung „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] sein. Im Zuge einer Analyse der Ergebnisse sollen die erzielten Ergebnisse im Vergleich zur Referenz [29] und zu einer Auswahl von State of the Art Verfahren gestellt und eine Einschätzung vorgenommen werden. Im Rahmen des Vergleichs soll die These der Veröffentlichung [29] beleuchtet werden, dass die erzielten Ergebnisse vergleichbar zu den Ergebnissen der State of the Art Verfahren sind. Weiterführend soll der Einfluss des Postprocessings auf die Ergebnisqualität untersucht werden.

1.3 Aufbau der Arbeit

Die Arbeit ist in die folgenden Kapitel gegliedert:

In **Kapitel 2** werden die Grundlagen und das theoretische Wissen über neuronale Netze und ihrer Bestandteile in Hinblick auf die durchgeführte Arbeit vermittelt.

In **Kapitel 3** wird der Aufbau und die Funktionsweise der erstellten Systemarchitektur dargelegt.

In **Kapitel 4** wird die allgemeine Umsetzung der vorgestellten Systemarchitektur vorgestellt. Dazu gehören die Generierung von Trainingsdaten, die technische Umsetzung sowie das Training des neuronalen Netzes und die Tiefenbildgenerierung.

In **Kapitel 5** werden die allgemeine Ergebnisse des Systems dargestellt, eine Einordnung zur Referenz und anderen State of the Art Verfahren vorgenommen und der Einfluss des Postprocessings auf die Ergebnisse untersucht.

In **Kapitel 6** wird die Arbeit zusammengefasst, ein Fazit gezogen und ein Ausblick für weiterführende Untersuchungen gegeben.

2 Grundlagen

2.1 Menschliches Sehen

Die Augen erlauben es dem Menschen seine Umgebung wahrzunehmen und mit ihr zu interagieren. Dabei erstellt der Mensch automatisch und unbewusst aus seiner zweidimensionalen Sicht eine dreidimensionale Wahrnehmung.

Diese Tiefensicht wird im visuellen Cortex mit Hilfe von Stereopsis (Stereosehen) und monokularen Tiefenkriterien (Monocular Cues) gebildet [30]. Bei der Stereopsis werden die Tiefeninformationen aus dem Positionsunterschied eines Objektes auf den zwei Retinas bestimmt (binokulare Disparität). Je größer die binokulare Disparität ist, desto dichter befindet sich ein Objekt am Betrachter [25].

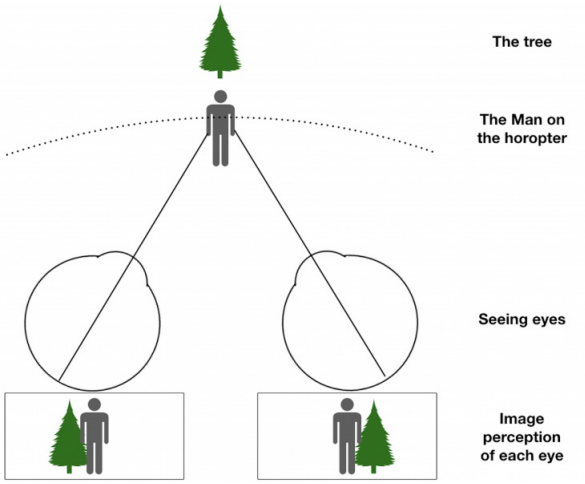
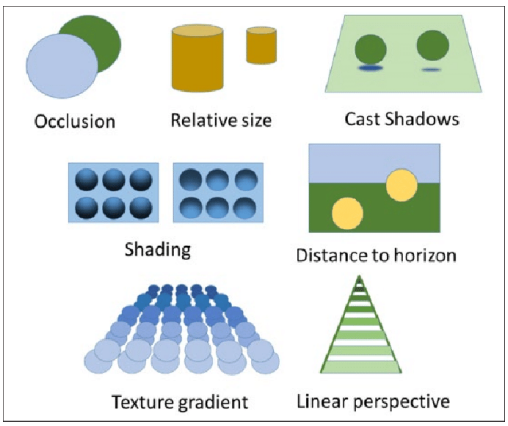
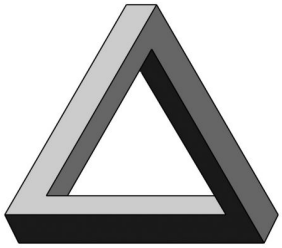


Abbildung 2.1: Stereosehen mittels binokulare Disparität [28]. Beim Betrachten einer Person, die vor einem Baum steht, nimmt jedes Auge die Position der Person zum Baum unterschiedlich wahr. Dieser Positionsunterschied wird als binokulare Disparität bezeichnet und ermöglicht Stereosehen.

Zusätzlich zur Stereopsis werden monokulare Tiefenkriterien hinzugezogen, wie Texturvariationen und Texturänderungen, Verdeckung (Occlusion), bekannte Größen von Objekten und weitere Kriterien. Einige Beispiele können der Abbildung 2.2a entnommen werden. Viele dieser Tiefenkriterien beruhen auf Erfahrungen und ermöglichen, bei Verlust oder Einschränkung eines Auges, weiterhin Tiefensicht. Beispielsweise erinnert sich ein Mensch daran, dass eine bestimmte Form ein Haus verkörpert und kann aufgrund dessen seine Größe abschätzen. Diese Erinnerung wird in zukünftigen Situationen hinzugezogen und hilft bei der Tiefeneinschätzung [25]. Erinnerungen können sich auch negativ auf die Tiefenwahrnehmung auswirken. Optische Täuschungen suggerieren ein anderes Bild der Realität und manipulieren die räumliche Wahrnehmung (siehe Abbildung 2.2b).



(a)



(b)

Abbildung 2.2: Beispiele für monokulare Tiefenkriterien (2.2a) [5] und ein Beispiel für optische Täuschungen (2.2b [2])

2.2 Computer Vision

Computer Vision ist ein interdisziplinäres Wissenschaftsgebiet an der Schnittstelle zwischen Informatik und anderen Naturwissenschaften. Es kann in drei große Teilbereiche unterteilt werden [24]. Das erste Teilgebiet ist die Bildaufnahme. Zur Bildaufnahme gehört die Technik von Sensoren und Kameras sowie die Speicherung der Aufnahmen. Die weiteren Teilgebiete Bildverarbeitung und Bildanalyse gehen fließend ineinander über

und können in die Segmente Bildvorverarbeitung, Detektion von elementaren Merkmalen, Segmentierung und in die Identifikation von Objekten aufgeteilt werden. In der Bildvorverarbeitung werden Aufnahmen bereits in der Kamera aufbereitet. Es werden beispielsweise die Helligkeit oder der Kontrast, für eine bessere Wiedergabe oder bessere Analysierbarkeit, optimiert. Bei der Detektion von elementaren Merkmalen geht es um die Lokalisierung von markanten Bereichen in einem Bild. Diese markanten Bereiche können zum Beispiel Kantenübergänge oder Eckpunkte sein. In der Segmentierung werden zusammengehörige Bereiche in einem Bild identifiziert. Die Zusammengehörigkeit wird dabei über die Ähnlichkeit der Pixel definiert, wobei die Definition der Ähnlichkeit abhängig vom Anwendungsfall sein kann. Ein typisches Ähnlichkeitsmerkmal stellt zum Beispiel der Farb- oder Grauwert eines Pixels dar. In der Identifikation von Objekten geht es um die Lokalisation und Identifikation von Objekten in Bildern. Beispielsweise der Erkennung eines Verkehrszeichen in ein Straßenszene.

2.3 Maschinelles Lernen - Überwachtes Lernen

Im Allgemeinen wird bei einer Maschine von einem Lernprozess gesprochen, wenn diese, als Reaktion auf ein Ereignis, ihr Programm, ihre Struktur oder ihre Parameter so anpasst, dass die zukünftig zu erwartende Leistung verbessert wird [23].

Beim maschinellen Lernen soll Wissen aus Erfahrung generiert werden, indem Lernalgorithmen aus Beispielen ein komplexes Modell entwickeln, welches die explizite Programmierung ersetzt [10]. Das entwickelte Modell soll anschließend auf unbekannte Daten derselben Art angewendet werden können. Diese Art der Problemlösung biete sich immer dann an, wenn der zu realisierende Prozess zu komplex für eine analytische Beschreibung ist und gleichzeitig genügend Beispieldaten vorhanden sind.

Ein Beispielproblem, in welchem sich maschinelles Lernen bewährt hat, ist die Klassifikation. Der häufig verwendete MNIST-Datensatz enthält Bilder von 70000 handgeschriebenen Ziffern (Beispiele in Abb. 2.3) [18].

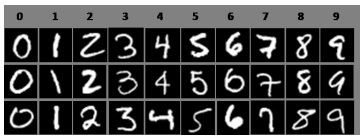


Abbildung 2.3: Beispielziffern aus dem MNIST-Datensatz [31].

Die in Abbildung 2.3 dargestellten Beispielziffern verdeutlichen wie unterschiedlich die einzelnen Ziffern geschrieben werden können. Zusätzlich weisen einige dieser Ziffern Ähnlichkeiten zueinander auf und erschweren die Klassifikation. Dieses Problem mit allgemeingültigen Regeln und expliziter Programmierung zu lösen ist schwierig. Für die Lösung dieses Problems bietet sich daher ein lernendes Verfahren an.

Beim maschinellen Lernen wird im Allgemeinen zwischen dem überwachtem Lernen, dem unüberwachtem Lernen und dem bestärkendem Lernen unterschieden [23]. In dieser Arbeit findet nur das überwachte Lernen Anwendung.

Überwachtes Lernen

Überwachtes Lernen ist die Optimierung eines Modells anhand bekannter Input- und dazugehöriger Outputdaten. Ziel dieser Modelloptimierung ist die möglichst genaue Abbildung der Trainingsdaten, ohne diese auswendig zu lernen (Overfitting). Stattdessen soll das Modell so verallgemeinert sein, dass auch unbekannte Daten möglichst korrekt zugeordnet werden [11].

Das überwachte Lernen kann in die Teilbereiche Klassifikation und Regression unterteilt werden. Bei der Klassifikation wird einem Input eine passende Klasse zugeordnet. Ein Beispiel für die Klassifikation ist die bereits erwähnte Klassifikation von MNIST Ziffern. Bei der Klassifikation von MNIST Ziffern wird einer handgeschriebenen Ziffer der passende Ziffernwert zugeordnet. Bei der Regression soll die Beziehung einzelner Variablen zueinander analysiert werden und wie sie sich wechselseitig beeinflussen. Beispielsweise wird in der prädiktiven Analytik Regression von historischen Daten zur Vorhersage von Trends und Kursentwicklungen verwendet.

2.4 Künstliche neuronale Netze

Die künstlichen neuronalen Netze sind ein Teilgebiet des maschinellen Lernens. Die Idee beruht auf der Nachahmung des aus Neuronen und Synapsen bestehenden menschlichen Gehirns [7]. Künstliche neuronale Netze besitzen eine Graphen-ähnliche-Struktur, welche aus über Gewichte verknüpfte Schichten (Layer) von Neuronen besteht. Die Abbildung 2.4 zeigt beispielhaft die Struktur eines künstlichen neuronalen Netzes. Bei dem hier gezeigten Beispiel handelt es sich um ein Fully Connected Feed Forward Network, bei

dem alle Neuronen eines Layers mit allen Neuronen des nachfolgenden Layers verbunden sind. Der erste Layer wird als Input-Layer bezeichnet und sorgt für die Aufnahme der Input-Daten. Am Ende des Netzes befindet sich der Output-Layer, der die finale Antwort berechnet. Zwischen diesen beiden Schichten können sich eine Vielzahl von Hidden-Layern befinden.

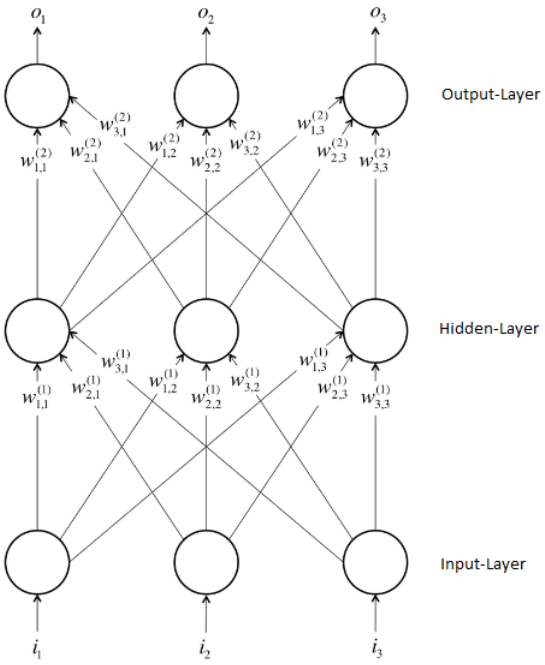


Abbildung 2.4: Beispiel eines neuronalen Netzes (in Anlehnung an [7]).

Der Grundbaustein eines Layers ist das Neuron. Ein Neuron ordnet n skalaren Eingangswerten $x_1 ... x_n$ einen skalaren Ausgangswert y nach der Formel 2.1 zu.

$$y = \varphi(\sum_{i=1}^{n} x_i \cdot w_i + b) \tag{2.1}$$

Die Parameter $w_1 ... w_n$ werden als Gewichte und b als Bias bezeichnet. Diese stellen die trainierbaren Parameter dar. $\varphi(\cdot)$ ist die Aktivierungsfunktion (siehe dazu Abschnitt 2.5.3).

Wird diese formale Beschreibung eines Neurons auf einen Layer übertragen, so können die Inputwerte als $\vec{x} = x_1 ... x_n$ und die Outputwerte als $\vec{y} = y_1 ... y_n$ dargestellt werden.

Die Gewichte w werden in der Gewichtsmatrix W und die Biaswerte im Spaltenvektor $\vec{b}$ zusammengefasst. Daraus ergibt sich folgende Formel für einen Layer:

$$\vec{y} = \varphi(W\vec{x} + \vec{b}) \tag{2.2}$$

Als Hyperparameter werden die Parameter bezeichnet, die die Netzwerkstruktur definieren. Dazu zählen die Anzahl der Layer, die Anzahl der Neuronen pro Layer sowie die Aktivierungsfunktion.

2.4.1 Convolutional Neural Networks (CNN)

In Computern werden Bilder als Tensoren der Form $W \cdot H \cdot C$ dargestellt. Dabei ist W die Breite des Bildes (Width), H die Höhe (Height) und C die Anzahl der Farbkanäle. Soll ein Bild mittels eines vollvernetzten künstlichen neuronalen Netzes analysiert werden, ergibt sich daraus selbst bei einem kleinen Netzwerk eine große Anzahl von Parametern. Soll beispielsweise ein Farbbild der Größe $200x200$ Pixeln analysiert werden, ergibt sich daraus ein Inputvektor mit einer Größe von $200 \cdot 200 \cdot 3 = 120000$ [7]. Wird dieser Inputvektor von einem Hidden-Layer mit 200 Neuronen verarbeitet, ergeben sich daraus $120000 \cdot 200 = 24000000$ Gewichte und 200 Biaswerte die trainiert werden müssen. Mit steigender Bildgröße und Anzahl an Hidden-Layern wird die Parameteranzahl immer unpraktikabler und begünstigt Overfitting beim Training. In Abbildung 2.5 wird dieser Sachverhalt der Parameterzunahme anhand eines Beispiels visualisiert. Bei einer Inputbildvergrößerung von 2x2 auf 4x4 Pixeln erhöht sich die Anzahl Verbindungen und damit auch die der Parameter zum nächsten Layer deutlich.

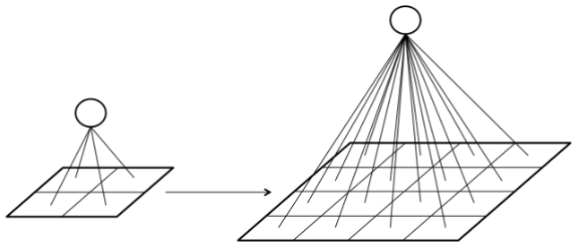


Abbildung 2.5: Skalierungsproblem eines künstlichen neuronalen Netzes bei der Analyse von Bildern [7].

Im Bereich der Verarbeitung von Daten mit räumlichen Relationen, zum Beispiel Bildern, haben sich Convolutional Neural Networks (CNNs, Faltungsnetzwerke) als effiziente und

erfolgreiche Modell erwiesen [12]. Bei CNNs wird Faltung (Convolution) zwischen den Inputdaten, zum Beispiel einem Bild, und kleine Filtern angewendet, wodurch die Anzahl an Parametern sinkt. Durch die räumliche Relation in den Inputdaten kann das Netzwerk sich die Gewichte teilen. In Abschnitt 2.5 werden die einzelnen Bestandteile eines Faltungsnetzwerkes und deren Funktionsweise genauer erläutert.

2.4.2 Training neuronaler Netze

Ziel des Trainings eines neuronalen Netzes ist die Leistungssteigerung des Modells. Diese Leistungssteigerung wird durch Adaption der lernbaren Parameter, Gewichte und Biases, erreicht. Für die Verbesserung der lernbaren Parameter werden sogenannte Optimierer (Optimizer) verwendet, die die Verlustfunktion (Loss Function) minimieren. Mittels der Verlustfunktion wird beim Training der Unterschied zwischen der Vorhersage und dem zu erwartenden Output (Ground Truth) ermittelt [12]. Im folgenden Abschnitt sollen die in dieser Arbeit verwendeten Optimierer vorgestellt werden.

Gradientenabstieg

Das Gradientenabstiegsverfahren (Gradient Descent) ist ein Gradienten basierendes, iteratives Verfahren zur Lokalisierung des Minimums einer Funktion $f(x)$ [11] [12]. Es beruht auf der iterativen Anpassung eines Punktes x entlang der negativen Gradientenrichtung $-\nabla f(x)$ und somit in Richtung eines Funktionsminimums. Bezogen auf den Verwendungszweck in neuronalen Netzen geht es um die Optimierung der Gewichte w [6]. Beim Gradientenabstieg muss zunächst der gesamte Trainingsdatensatz durchlaufen werden, bevor eine Anpassung der Gewichte vorgenommen werden kann. Der Gradientenabstieg ist definiert als

$$w_{i+1} = w_i - \frac{\gamma}{n} \sum_{i=1}^{n} \nabla f(w_i) \tag{2.3}$$

mit w_i als aktuell evaluiertes Gewicht, n als Anzahl der Trainingsdatensätze, $\nabla f(w)$ als Gradient und γ als Schrittlänge bzw. Lernrate (Learning rate).

Stochastischer Gradientenabstieg

Der stochastische Gradientenabstieg (Stochastic gradient descent, SGD) ist eine vereinfachte bzw. verschlankte Form des Gradientenabstiegs. Im Vergleich zum Gradientenabstieg wird der Gradient nicht genau und somit in Abhängigkeit aller Datensätze berechnet, sondern es wird eine Schätzung auf Basis eines zufällig ausgewählten Beispiels vorgenommen [6]. Daraus ergibt sich folgende Definition für den stochastischen Gradientenabstieg:

$$w_{i+1} = w_i - \gamma \nabla f(w_i) \tag{2.4}$$

Mit den Parametern: w_i als aktuell evaluiertes Gewicht, $\nabla f(w)$ als Gradient und γ als Schrittlänge bzw. Lernrate (Learning rate).

Mini-Batch-Gradientenabstieg

Eine Mischung aus Gradientenabstieg und stochastischem Gradientenabstieg ist der Mini-Batch-Gradientenabstieg (Minibatch Stochastic Gradient Descent, MBSGD). Im Mini-Batch-Gradientenabstieg wird der Trainingsdatensatz in sogenannte Mini-Batches der Größe n geteilt. Die Optimierung der Gewichte erfolgt nach jeder Mini-Batch. Die formale Definiton des Mini-Batch-Gradientenabstieg ist analog zum Gradientenabstieg in Formel 2.3 mit dem Unterschied, dass n die Anzahl der Datensätze in einer Mini-Batch darstellt. Eine Erweiterung des Mini-Batch-Gradientenabstiegs ist der Adam Optimierer [15]. Bei Adam wurde der Mini-Batch-Gradientenabstieg um eine adaptive Lernrate und einen Momentumterm erweitert, wodurch die lokalen Minima schneller gefunden und Plateaus schneller überwunden werden können.

2.5 Netzwerkbestandteile

Der Aufbau und damit die enthaltenen Komponenten eines neuronalen Netzes variieren je nach Anwendung. In diesem Abschnitt sollen die in dieser Arbeit verwendeten Netzwerkkomponenten des neuronalen Netzes vorgestellt werden.

2.5.1 Faltungs-Layer

Der Faltungs-Layer (Convolutional Layer) ist der primäre, lernbare Layer in einem Faltungsnetzwerk. Jeder Layer besteht aus einer Anzahl von lernbaren Filtern, auch Kernels genannt, die die Gewichte w und den Biasterm b enthalten. Die Filter lernen beim Training Eigenschaften (Features), wie zum Beispiel Kanten, zu erkennen. Ein Filter ist definiert durch seine Größe FxF (z. B. 3x3 oder 5x5), seine Schrittgröße (Stride) und seinen durch Faltung erzeugten Output (Feature Map). Bei der Faltung wandert der Filter mit seiner Schrittgröße über den Input und erzeugt dabei den Output. In Abbildung 2.6 wird der Ablauf einer Faltung beispielhaft dargestellt.

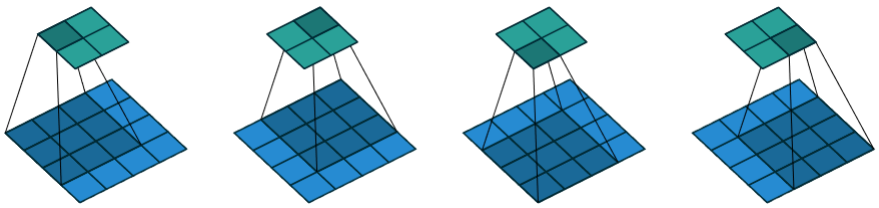


Abbildung 2.6: Beispiel einer Faltung. Ein Input mit einer Größe von $4x4$ wird mittels eines $3x3$ Filters mit Schrittgröße von Eins gefaltet [9].

Da Bilddaten zusätzlich zu ihrer Größe $W \cdot H$ (Breite und Höhe) noch eine weitere Dimension für die Farb- oder Grauwerte aufweisen, muss die Faltung im Volumenraum durchgeführt werden. Dazu werden die Filter ebenfalls als Volumenfilter angelegt. In der nachfolgenden Abbildung 2.7 wird der Vorgang der Faltung eines RGB-Farbbildes mit einem Volumenfilter dargestellt.

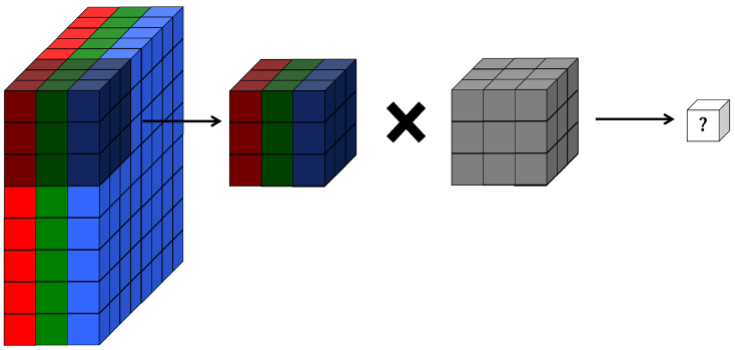


Abbildung 2.7: Beispiel einer Faltung eines RGB-Farbbildes [7].

Wird die Darstellung in Abbildung 2.7 auf den gesamten Faltungs-Layer übertragen, so ergibt sich die Abbildung 2.8. In der Abbildung 2.8 werden mehreren Filter zu einem Faltungslayer zusammengeschlossen und diese erzeugen das Outputvolumen (Feature Map).

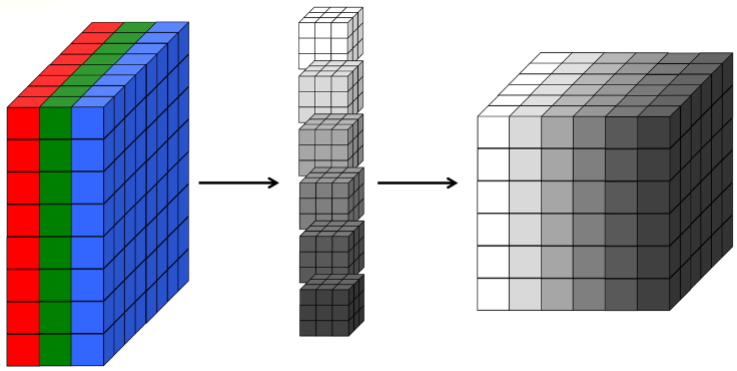


Abbildung 2.8: Beispiel eines Faltungslayers [7].

Wie in den vorangegangen Abbildungen 2.6, 2.7 und 2.8 zu erkennen ist, verkleinert sich die Outputgröße im Vergleich zur Inputgröße bei der Faltung. Das sogenannte Zero-Padding ist eine Maßnahme mit der der Output auf die gewünschte Größe gebracht werden kann. Beim Zero-Padding wird der Input mit einem Rand von Nullen erweitert, sodass nach der Faltung die gewünschte Größe erreicht wird (siehe Abbildung 2.9).

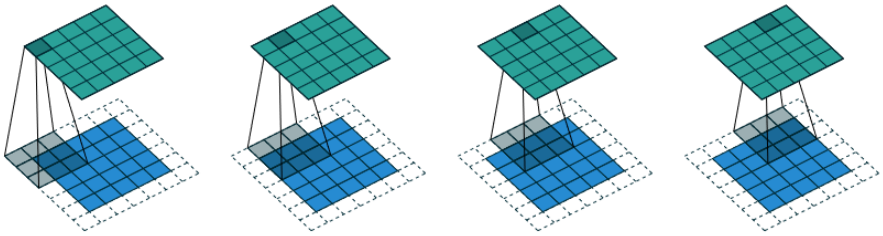


Abbildung 2.9: Beispiel einer Faltung mit Zero-Padding [9].

2.5.2 Dense-Layer

In einem Dense-Layer (Fully-connected Layer, voll vernetzter Layer) sind alle Neuronen eines Layers mit allen Neuronen des nachfolgenden Layers verbunden (siehe Abbildung 2.4). In vielen Modellen des maschinellen Lernens sind die letzten paar Layer Dense-Layer. Sie sammeln die in den vorherigen Layern extrahierten Informationen (Features)

und generieren daraus den finalen Output des Netzes. Der Input für einen Dense-Layer ist ein Feature Vektor.

In Faltungsnetzwerken muss der mehrdimensionale Output eines Faltungs-Layers in einen eindimensionalen Vektor überführt werden. Dieser Vorgang wird als Flatten bezeichnet.

2.5.3 Aktivierungsfunktionen

Mittels der Aktivierungsfunktion wird entschieden, ob ein Neuron aktiviert werden soll oder nicht. Genauer bedeutet dies, dass auf Basis des Outputs eines Neurons entschieden wird, ob der Wert wichtig für die Vorhersage des Netzes ist und ob dieser bei der Optimierung des Netzes berücksichtigt werden soll. Durch die Aktivierungsfunktion wird Nichtlinearität der sonst linearen Netzwerkstruktur sichergestellt [11].

Im Nachfolgenden sollen die beiden in dieser Arbeit verwendeten Aktivierungsfunktionen vorgestellt werden.

Sigmoid

Bei der Sigmoid-Aktivierungsfunktion wird für einen Input z ein Output $f(z)$ im Bereich zwischen 0 und 1 berechnet (siehe Abbildung 2.10). Ein Output von $f(z) = 0$ bedeutet keine Aktivität und $f(z) = 1$ ist die maximal mögliche Aktivierungsrate. Für Zwischenwerte nimmt die Funktion eine S-Form an [7].

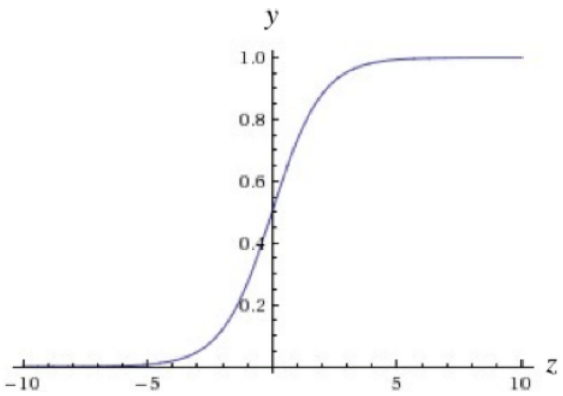


Abbildung 2.10: Darstellung der Sigmoid-Aktivierungsfunktion [7].

Die Sigmoid-Aktivierungsfunktion ist nichtlinear, differenzierbar, monoton und hat einen festen Ergebniswertebereich. Die formale Definition der Sigmoid-Aktivierungsfunktion ist in Gleichung 2.5 dargestellt.

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

Die nichtlineare Sigmoid-Aktivierungsfunktion weist zwei Hauptnachteile auf. Der erste Nachteil ist das mögliche Verschwinden von Gradienten (Vanishing Gradient). Die Sigmoid-Aktivierungsfunktion sättigt die Aktivierung von Neuronen nahe der Bereiche von $f(z) = 0$ oder $f(z) = 1$. Dieser Sachverhalt führt dazu, dass die Gewichte dieser Neuronen beim Training nicht mehr aktualisiert und so der Lernprozess gehemmt wird. Der zweite Nachteil ist, dass die Sigmoid-Aktivierungsfunktion nicht null-zentriert ist. Dies führt zu Problemen bei der Backpropagation, denn nachfolgende Layer erhalten nur nicht null-zentrierte Daten, was Einfluss auf die Gradientenoptimierung hat. Daraus resultieren nur positive oder nur negative Gradienten der Gewichte, was zu ungewollten Oszillationen in der Backpropagation und der Gewichtsaktualsierung führt.

Eine Alternative zur Sigmoid-Aktivierungsfunktion ist der Tangens hyperbolicus (tanh). Im Vergleich zur Sigmoid-Aktivierungsfunktion werden die Daten null-zentriert weitergegeben und ungewollte Oszillation während der Gewichtsaktualsierung wird vermieden. Das Problem des möglicherweise verschwindenden Gradienten ist weiterhin vorhanden.

Rectified Linear Unit

Die Rectified Linear Unit Aktivierungsfunktion (ReLU) ist eine rechnerisch einfache Aktivierungsfunktion, die im Vergleich zu sigmoiden Aktivierungsfunktionen keine Sättigung aufweist und ein schnelleres Training ermöglicht [17]. Für einen Input z ist sie definiert als

$$f(z) = max(0, z). \tag{2.6}$$

Die Relu-Aktivierungsfunktion führt eine Schwellwertoperation aus, die nur für positive Werte aktiviert wird (siehe Abbildung 2.11).

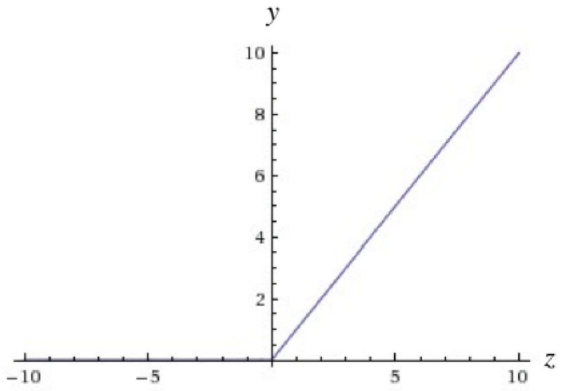


Abbildung 2.11: Darstellung der Relu-Aktivierungsfunktion [7].

Im Gegensatz zu den anderen nichtlinearen Aktivierungsfunktionen, wie zum Beispiel Sigmoid, hat die Relu-Aktivierungsfunktion einen konstanten Gradienten für Werte von $z \geq 0$. Dadurch wird für ansteigende Werte von z das Verschwinden von Gradienten eliminiert. Für negative Werte wird die Relu-Aktivierungsfunktion nicht aktiviert, was in einem Null-Output und einem Null-Gradienten bei der Backpropagation resultiert. Neuronen, auf die dies zutrifft, können als „tot“ bezeichnet werden. Sie werden nicht länger beim Training aktualisiert und sind dadurch nicht mehr Teil des Netzes.

2.5.4 Trainingsverbesserungskomponenten

In diesem Abschnitt werden Layer vorgestellt, die die Performanz und Konvergenz des Netzes verbessern.

Batchnormalisierung

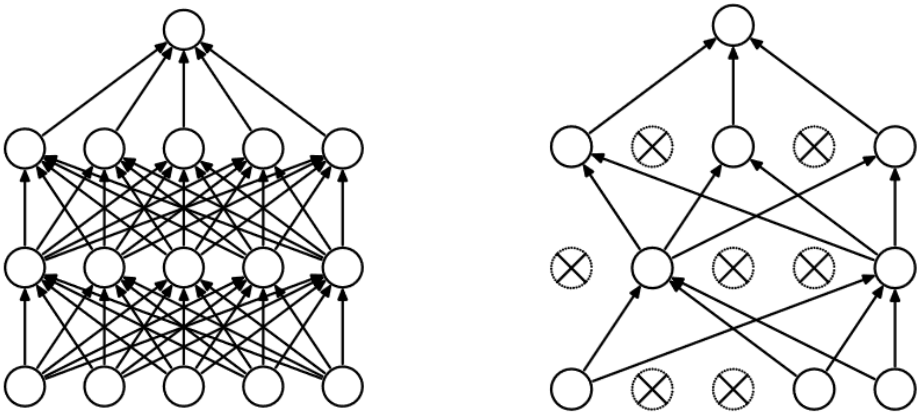
Die Batchnormalisierung (Batch Normalization) wurde in 2015 von Ioffe et al. [14] vorgestellt und ist ein Verfahren zur Verbesserung der Trainingsleistung eines Netzes. Es erhöht die Konvergenzrate, ermöglicht schnelleres Lernen, verhindert die Sättigung von Aktivierungsfunktionen und erübrigt die Notwendigkeit von Dropout-Layern [7][14].

Beim Training von nicht normalisierten Inputdaten verändert sich die Verteilung nach jeder Mini-Batch. Dieser Sachverhalt resultiert in einer Verschiebung der Kovariaten. Durch diese Verschiebung wird der Trainingsprozess erschwert, weil für jede Mini-Batch

die Parameter die Verschiebung kompensieren müssen. Aus diesem Grund muss eine kleine Lernrate gewählt und bei der Initialisierung der Gewichte aufgepasst werden, damit das Netzwerk bei Training nicht divergiert. Die Idee hinter der Batchnormalisierung ist die Verringerung der Verschiebung der Kovariaten zwischen den Mini-Batches durch Normalisierung. Die Normalisierung in der Batchnormalisierung wird durch die Subtraktion des Mittelwertes und der anschließenden Teilung durch die Standardabweichung für jede Mini-Batch erreicht [7].

Dropout

Dropout ist eine Technik zur Vermeidung von Overfitting beim Training von neuronalen Netzen [26]. Die Idee hinter Dropout ist es, einzelne Neuronen mit all ihren ein- und ausgehenden Verbindungen temporär aus dem Netz zu entfernen bzw. zu deaktivieren (siehe Abbildung 2.12). Die Auswahl, welche Neuronen temporär deaktiviert werden sollen, wird zufällig getroffen. Die Anzahl, der zu deaktivierenden Neuronen, kann über einen Parameter definiert werden.



(a) Normales neuronales Netz (b) Neuronales Netz mit Dropout

Abbildung 2.12: Vergleich eines neuronalen Netzes mit und ohne Dropout [26].

3 Systemarchitektur

In diesem Kapitel wird die verwendete Systemarchitektur vorgestellt. Der in dieser Arbeit verwendete Ansatz ist ein „patch based“ Ansatz. Dies bedeutet, dass das zu analysierende Bild in kleinere Teilbilder (Patches) geteilt und diese nacheinander im neuronalen Netz analysiert werden. Wie in den meisten „patch based“ Ansätzen besteht auch dieser Ansatz aus zwei Hauptbestandteilen [29]. Der erste Teil ist die initiale Kostengenerierung (Cost Volume Generation) mittels des neuronalen Netzes für die Berechnung der Disparity Map. Im zweiten Schritt werden die Ergebnisse der initialen Kostengenerierung weiter aufgearbeitet und der finale Output generiert. Dieser Schritt wird auch als Nachbearbeitung (Postprocessing) bezeichnet.

In den nachfolgenden Unterkapiteln werden die verwendete Netzwerkstruktur und die angewendeten Schritte des Postprocessing vorgestellt.

3.1 Netzwerkstruktur

In den meisten Anwendungsfällen von neuronalen Netzen für das „Patch Based Stereo Matching“ wird eine siamesische Netzwerkstruktur gewählt. Bei dieser Strukturform bearbeiten zwei identische neuronale Netze mit gleichen Gewichten parallel zwei verschiedene Inputs und erzeugen vergleichbare Outputs [29]. Diese Outputs werden in ein weiteres neuronales Netz überführt und dort die Übereinstimmung berechnet. In der Abbildung 3.1 wird eine solche Netzwerkstruktur beispielhaft dargestellt.

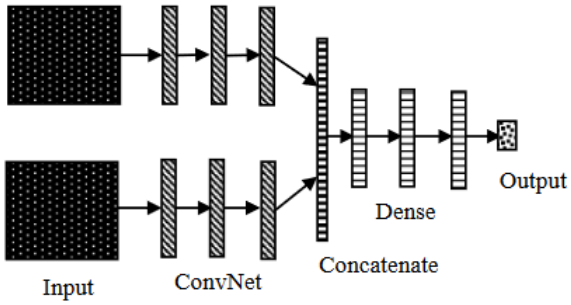


Abbildung 3.1: Beispiel einer siamesischen Netzwerkstruktur [29].

In dieser Arbeit wird ein anderer neuer Ansatz gewählt, der als „Stacked Stereo CNN“ oder kurz als „SS-CNN“ bezeichnet wird [29]. Bei dieser Netzwerkstruktur wird nur ein einziges neuronales Netz verwendet. Als Input dienen zwei Patches, die entlang der Farbinformationen aufeinander gestapelt (Stacked) werden. Auf Basis dieses gestapelten Patches wird die Ähnlichkeit (Similarity Score) berechnet. Die verwendete Netzwerkstruktur ist in Abbildung 3.2 dargestellt. Sie besteht aus vier Faltungs-Layern mit jeweils nachfolgender Batchnormalisierung, einem Flatten-Layer, einem Dense-Layer und zuletzt dem Output-Layer.

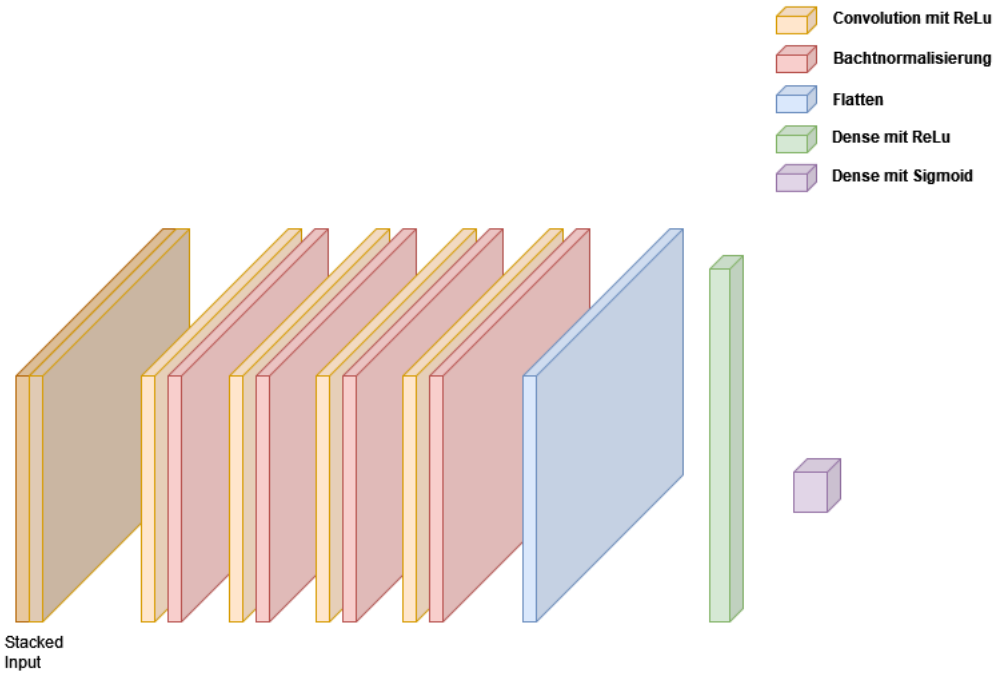


Abbildung 3.2: Architektur des SS-CNN.

Die vier Faltungs-Layer haben alle einen $3x3$ Kernel und ReLu-Aktivierungsfunktion. Die Anzahl der Filter beträgt im ersten Faltungs-Layer 8, im Zweiten 16, im Dritten 32 und im Vierten 64. Der Output des letzten Faltungs-Layers wird in den Flatten-Layer überführt und anschließend in einen vollvernetzten Layer mit 64 Neuronen und Relu-Aktivierungsfunktion geleitet. Der letzte Layer ist der Output-Layer mit einem Neuron und Sigmoid-Aktivierungsfunktion. Die Zusammenfassung des Modells und seiner Parameter ist in Abbildung 3.3 dargestellt.

Layer	Output Shape	No. Parameters
Conv2D	(11, 11, 8)	440
Batch Normalization	(11, 11, 8)	32
Conv2D	(11, 11, 16)	1168
Batch Normalization	(11, 11, 16)	64
Conv2D	(11, 11, 32)	4640
Batch Normalization	(11, 11, 32)	128
Conv2D	(11, 11, 64)	18496
Batch Normalization	(11, 11, 64)	256
Flatten	7744	0
Dense	64	495680
Dense	1	65
Total Parameters		520969
Trainable Parameters		520729
Non-trainable Parameters		240
Model-Type		Sequential

Abbildung 3.3: Modellzusammenfassung des SS-CNN.

Der Input in das Netz ist ein Array der Form $W \cdot W \cdot C$ mit $W \cdot W$ als Patchgröße und C als Gesamtanzahl der Farbkanäle. Die Gesamtanzahl der Farbkanäle ergibt sich aus der Addition der Farbkanäle der beiden Inputpatches. Bei Graubildern ist die Gesamtanzahl 2 (1 für jede Inputpatch) und bei Farbbildern 6 (3 für jede Inputpatch).

Als Ergebnis liefert das SS-CNN ein Array der Form $H \cdot W \cdot D$ mit H und W als Höhe und Breite des Gesamtbildes und D als Disparity Range. Es enthält für jeden Pixel im Gesamtbild die Similarity Scores für jede mögliche Disparity. Für die weiterführenden Aufarbeitung im Postprocessing müssen die Similarity Scores in Matching Costs überführt werden. Dazu werden die Similarity Scores mit minus Eins multipliziert [32].

3.2 Postprocessing

Im Postprocessing werden die Ergebnisse des neuronalen Netzes weiter aufgearbeitet und die finale Disparity Map generiert. Die Schritte des Postprocessings sollen die negativen Einflüsse von texturarmen Bereichen, Occlusion, sich wiederholenden Mustern, Reflektion und anderen negativen Einflüssen minimieren [29].

Das Postprocessing besteht aus vier Schritten. Im ersten Schritt, dem Semi-Global Matching, werden die Matching Costs des neuronalen Netzes in Zusammenhang zu ihren jeweiligen Umgebungen gebracht und dementsprechend angepasst. Im nachfolgenden Schritt werden die Werte mit den geringsten Matching Costs für jeden Pixel ausgewählt und aus ihnen die initiale Disparity Map generiert. Diese beiden Schritte werden für beide Bilder angewendet. Die beiden initialen Disparity Maps für das linke und das rechte Bild werden im nachfolgenden Schritt miteinander verglichen. Punkte mit einer zu großen Disparitätendifferenz zwischen linkem und rechtem Bild werden verworfen. Im letzten Schritt werden diese verworfenen Punkte versucht in Abhängigkeit ihrer zugehörigen Nachbarn zu füllen.

3.2.1 Semi-Global Matching

Der erste Schritt des Postprocessings ist das Semi-Global Matching [13]. Die hier verwendete Variante und die dazugehörigen Parameter beruhen auf der adaptierten Variante von Zbontar und LeCun aus 2016 [32].

Außer an den Rändern variiert die Disparität in einem Bildbereich nur wenig und gleichmäßig [29]. Diese Gleichmäßigkeit (Smoothness Constraint) wird beim Semi-Global Matching dazu verwendet, eine Energiefunktion $E(D)$ in Abhängigkeit des Disparitätenbildes (Disparity Image) D aufzustellen. Angelehnt an Zbontar und LeCun [32] ergibt sich daraus der nachfolgende Term 3.1 für die Energiefunktion $E(D)$.

$$E(D) = \sum_{p} \left(C_{CNN}(p, D(p)) \; + \sum_{p \in N_p} P_1 \cdot 1\{|D(p) - D(q)| = 1\} \right. \\ \left. + \sum_{p \in N_p} P_2 \cdot 1\{|D(p) - D(q)| > 1\} \right) \quad (3.1)$$

In der Energiefunktion ist p der aktuelle Pixel, N_p die Nachbarschaft des Pixels p, q ein Pixel aus der Nachbarschaft N_p, $D(p)$ die Disparität von p, $D(q)$ die Disparität von q, C_{CNN} die Matching Costs, P_1 und P_2 sind „Strafparameter“ zum „bestrafen“ von großen Disparitätensprüngen und $1\{\cdot\}$ ist die Indikatorfunktion. Der erste Term sind die berechnete Matching Cost des neuronalen Netzes. Im zweiten Term wird eine „Strafe“ P_1 den Matching Costs hinzugefügt, wenn der Disparitätenunterschied zum Nachbarn genau 1 ist. Im dritten Term wird eine größere „Strafe“ P_2 hinzugefügt, wenn der Disparitätenunterschied zum Nachbarn größer als 1 ist.

Zur Optimierung der Matching Costs gilt es die Energiefunktion $E(D)$ (Formel 3.1) entlang der beiden horizontalen und der beiden vertikalen Richtungen zu minimieren. Anschließend wird aus den Ergebnissen der Durchschnitt gebildet, um das finale Ergebnis zu erhalten. Zur Minimierung der Energiefunktion $E(D)$ entlang einer Richtung r wird die Matching Cost $C_r(p,d)$ nach folgender Formel berechnet:

$$C_r(p,d) = C_{CNN}(p,d) - \min_k C_r(p-r,k) + min\Big\{C_r(p-r,d), C_r(p-r,d-1) + P_1, \\ C_r(p-r,d+1) + P_1, \min_k C_r(p-r,k) + P_2\Big\}. \quad (3.2)$$

Dabei ist $C_r(p,d)$ die Matching Cost für den Pixel p und der Disparität d entlang der Richtung r, C_{CNN} die Matching Costs aus dem neuronalen Netz, $p-r$ ist die vorherige Pixelposition entlang der Richtung r, k sind alle Matching Costs für alle Disparitäten eines Pixels und P_1 und P_2 sind die „Strafparameter“.

Die „Strafparameter“ P_1 und P_2 werden auf Basis des Bildgradienten gesetzt. Dazu wird der Intensitätsunterschied I_{Diff} zwischen benachbarten Pixeln entlang der Richtung r mittels der Formel 3.3 berechnet. Anschließend werden P_1 und P_2 nach der in Tabelle 3.1 gezeigten Zuordnungsvorschrift belegt.

$$I_{Diff} = |I(p) - I(p-r)| \quad (3.3)$$

Dabei ist I_{Diff} der Intensitätsunterschied, $I(p)$ die Intensität eines Pixels p und $p-r$ ist die vorherige Pixelposition.

Bedingung	$\boldsymbol{P_1 =}$	$\boldsymbol{P_2 =}$
$I_{Diff} < sgm\_D$	sgm_P1	sgm_P2
$I_{Diff} \geq sgm\_D$	sgm_P1/sgm_Q2	sgm_P2/sgm_Q2
$Sonst$	sgm_P1/sgm_Q1	sgm_P2/sgm_Q1

Tabelle 3.1: Zuordnungsvorschrift der „Strafparameter“ P_1 und P_2

Die verwendeten Hyperparameter sgm_P1, sgm_P2, sgm_Q1, sgm_Q2, sgm_V und sgm_D sind von Zbontar und LeCun [32] adaptiert und in den Anlagen A.1 dargestellt. Der „Strafparameter“ P_1 wird für die vertikalen Richtungen um den Faktor sgm_V vermindert, da in diesen Richtungen öfter kleine Disparitätsschwankungen auftreten [32].

Die finalen Kosten des Semi-Global Matching C_{SGM} berechnen sich aus dem Durchschnitt der vier berechneten Richtungen mit der Formel:

$$C_{SGM}(p,d) = \frac{1}{4} \cdot \sum_{r} C_r(p,d). \tag{3.4}$$

3.2.2 Initiale Disparity Map

Zur Generierung der initialen Disparity Map wird die Disparität mit den geringsten Matching Costs für jeden Pixel ermittelt und daraus die initiale Disparity Map D erzeugt. Dazu wird die Formel 3.5 verwendet, bei der über den *argmin* der Index des Eintrages mit den geringsten Matching Costs zurückgegeben wird. Der Index des Eintrags entspricht der Disparität. Die so ermittelten Werte ergeben die initiale Disparity Map D.

$$D(p) = \underset{d}{argmin}(C_{SGM}(p,d)) \tag{3.5}$$

3.2.3 Left Right Consistency Check

Beim Left Right Consistency Check werden die initialen Disparity Maps des linken und rechten Bildes miteinander auf Gleichheit verglichen. Pixel mit einem Disparitätenunterschied größer 1 werden als ungleich klassifiziert.

Der Disparitätenunterschied D_{diff} eines Pixels ergibt sich aus

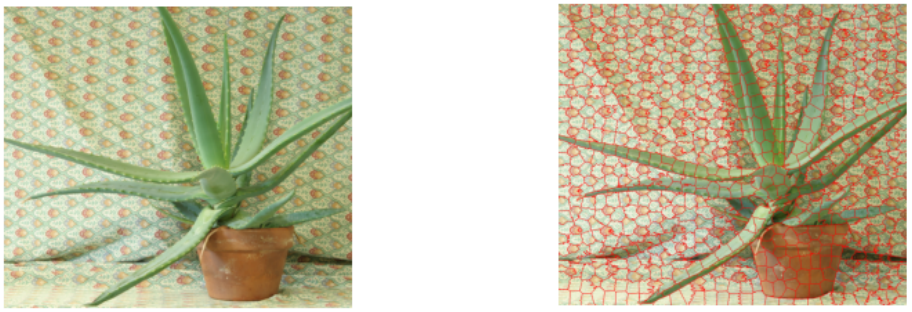
$$D_{diff} = |D_L(x, y) - D_R(x - d_l, y)|$$

mit $D_L(x, y)$ als Disparität der linken Disparity Map für den Pixel $p(x, y)$ und $D_r(x - d_L, y)$ als Disparität der rechten Disparity Map verschoben um die Disparität der linken Disparity Map d_l.

Für als ungleich klassifizierte Pixel wird der Disparitätenwert verworfen beziehungsweise durch einen negativen Wert ersetzt. Pixel mit einem negativen Wert werden als Löcher (Holes) bezeichnet. Im nachfolgenden Schritt dem Hole Filling Scheme (Abschnitt 3.2.4) werden diese Löcher gefüllt.

3.2.4 Hole Filling Scheme

Der letzte Schritt des Postprocessings ist das Hole Filling Scheme. Beim Hole Filling Scheme sollen die noch vorhandenen Löcher (Holes) in der Disparity Map gefüllt werden. Der hier verwendete Ansatz besteht aus drei Schritten. Im ersten Schritt wird das Ausgangsbild durch Segmentierung mittels des SLIC-Algorithmus in ein Superpixel-Bild überführt. Ein Superpixel ist eine kleine Region innerhalb des Bildes, in dem alle Pixel gleiche Bildeigenschaften wie zum Beispiel Farbe oder Textur aufweisen [29]. Der SLIC-Algorithmus ist eine Adaption des k-Means-Algorithmus [4]. Er erstellt Superpixel basierend auf den Farbeigenschaften des Bildes und der gewünschten Anzahl. In der nachfolgenden Abbildung 3.4 ist das Ausgangsbild dem segmentierten Bild gegenübergestellt. Im segmentierten Bild wurden die Grenzen zwischen den einzelnen Segmenten in rot markiert.



(a) Ausgangsbild [1] (b) SLIC-Segmentierung (in Anlehnung an [1])

Abbildung 3.4: Beispiel einer Segmentierung mit dem SLIC-Algorithmus.

Im zweiten Schritt wird reihenweise über die gesamte Disparity Map iteriert und nach Löchern gesucht. Wenn ein Loch gefunden wird, wird ein zentrierter $41x41$ Pixelausschnitt um das gefunden Loch herum betrachtet. Alle Pixel in diesem Ausschnitt, die sich im gleichen Superpixel wie das Loch befinden und selbst kein Loch sind, werden gesammelt und ihr Median gebildet. Dem aktuell betrachtetem Loch wird dieser Median-Wert zugeordnet. Im letzten Schritt wird die Disparity Map mit einem Medianfilter geglättet und so die finale Disparity Map erzeugt. Die besten Ergebnisse hinsichtlich der Fehlerquoten werden mit großen Medianfiltern (z. B. $21x21$ Pixel) erreicht. Diese führen im Gegenzug zu einer starken Glättung des finalen Bildes, wodurch sehr feine Strukturen verloren gehen können. Zur Beibehaltung dieser Strukturen bietet sich ein kleinerer Medianfilter mit einer Größe bis $11x11$ Pixel an. In dieser Arbeit wurde ein $11x11$ großer Medianfilter verwendet.

4 Umsetzung

In diesem Kapitel wird die allgemeine Umsetzung des Systems erläutert. Dafür soll zunächst auf die Generierung der Trainingsdaten eingegangen werden. Anschließend wird die technische Umsetzung, sowie das Training und der Gesamtablauf vorgestellt.

4.1 Trainingsdaten

Als Grundlage der Trainingsdaten dient der Middlebury Stereo-Datensatz 2006 [1]. Dieser Datensatz enthält 21 Stereobildpaare in verschiedenen Auflösungen mit dazugehörigen Disparity Maps. Ein Beispiel ist in Abbildung 4.1 dargestellt.

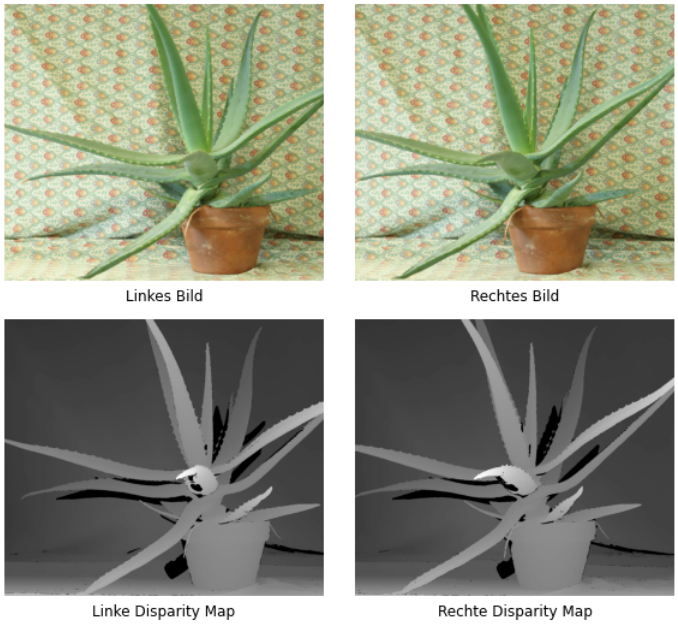
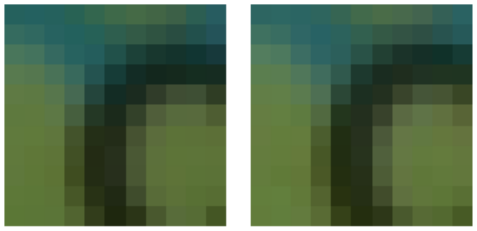


Abbildung 4.1: Beispiel eines Middlebury Stereobildpaares mit dazugehörigen Disparity Maps [1].

Aus den 21 Stereobildpaaren in Vollauflösung wurde ein gelabelter Datensatz mit 728700 gestapelten Patchpaaren erstellt. Ein gestapeltes Patchpaar ist die Zusammenführung zweier $11x11$ großen Patches entlang der Farbkanäle. Daraus ergibt sich eine Größe von $11 \cdot 11 \cdot 6$ für jedes gestapelte Patchpaar. Zu jedem Patchpaar gehört ein Label, welches definiert, ob es sich um ein gleiches oder ungleiches Patchpaar handelt. Pro Stereobildpaar wurden je 17350 gleiche und ungleiche Patchpaare erstellt (Beispiel in Abbildung 4.2).



(a) Gleiches Patchpaar

(b) Ungleiches Patchpaar

Abbildung 4.2: Beispiel eines gleichen und ungleichen Patchpaares. In 4.2a ist ein gleiches Patchpaar und in 4.2b ein ungleiches Patchpaar dargestellt.

Für die Erstellung eines gleichen Patchpaares wird eine zufällige Pixelposition $p_l(x, y)$ im linken Bild gewählt. Um diese Pixelposition p_l wird eine zentrierte $11x11$ Pixel große Patch gebildet. Dies ist die Patch des linken Bildes. Für die dazugehörige rechte Patch wird die x-Koordinate um die Disparität d_l verschoben $(p_r(x - d_l, y))$ und ebenfalls eine zentrierte $11x11$ Pixel große Patch erstellt.

Für die zufällig ausgewählte Pixelposition p_l muss eine bekannte Disparität d_l vorliegen $(d_l(p_l) \neq 0)$ und die zugehörige Position im rechten Bild muss innerhalb des Bildes liegen $(x - d_l \geq 0)$. Die zweite Bedingung ist wichtig, da es Pixel im linken Bild gibt, die zwar eine bekannte Disparität haben $(d_l(p_l) \neq 0)$, aber die Partnerposition im rechten Bild nicht vorhanden ist. Ohne diese Bedingung könnte es zu fehlerhaften Patchpaaren kommen. Diese fehlerhaften Patchpaare würden aus einer komplett schwarzen rechten

Patch und einer normalen linken Patch mit Struktur und Farbe bestehen. Dies trifft auf die Pixel dicht am linken Rand des linken Bildes zu.

Für ein ungleiches Patchpaar wird ebenfalls eine zufällige Pixelposition $p_l(x, y)$ im linken Bild gewählt und eine zentrierte $11x11$ Pixel große Patch erstellt. Für die Partnerpatch wird die Verschiebung z der x-Koordinate zufällig zwischen 1 und der maximalen Disparität des Bildes gewählt. Dabei gilt, dass z nicht im Bereich von ± 3 um die wahre Disparität d_l liegen darf.

4.2 Technische Umsetzung

Die Implementierung des Projektes erfolgt in Python 3 [27]. Python hat sich in den Bereichen des maschinellen Lernens und der Datenanalyse durch seine breite Auswahl an Bibliotheken und deren Zusammenspiel etabliert.

Die Erstellung und das Training des neuronalen Netzes erfolgt in Keras [8] mit einem Tensorflow [3] Backend. Dieses Zusammenspiel ermöglicht effizientes Rechnen und die Nutzung der Grafikkarte (GPU). Die Nutzung der Grafikkarte beschleunigt das Training und die Auswertung des neuronalen Netzes erheblich.

Als Laufzeitumgebung wird das JupyterHub [16] der HAW Hamburg mit 4 CPUs, 16 GB RAM und einer NVIDIA Tesla V100 GPU mit 16 GB VRAM verwendet.

Die verwendeten Bibliotheken sind im Anhang A.2 aufgelistet.

4.3 Training des neuronalen Netzes

Das Training des neuronalen Netzes aus Abschnitt 3.1 erfolgt mittels Trainingsdaten, die auf Basis des in Abschnitt 4.1 dargestellten Verfahrens der Trainingsdatengenerierung erzeugt wurden. Der Trainingsdatensatz wird vor dem Training normalisiert. Für die Normalisierung wird pro Farbkanal zunächst der Durchschnitt des Farbkanals subtrahiert und anschließend durch die Varianz des Kanals dividiert. Der normalisierte Trainingsdatensatz wird anschließend durchgemischt und in 80% Trainingsdaten und 20% Validierungsdaten geteilt. Die Durchmischung soll sicherstellen, dass von jedem Ausgangsbild Patches sowohl im Trainings- als auch Validierungsdatensatz vorhanden sind.

Als Hyperparameter des Trainings wird eine Batchgröße von 16, eine Laufzeit von 50 Epochen und der Adam Optimierer verwendet. Das trainierte neuronale Netz erreicht eine Endgenauigkeit (Accuracy) von 0,9269 und während des Trainings ein Maximum von 0,9276. Der Verlauf der Genauigkeit des Trainings ist in Abbildung 4.3 dargestellt. Eine Erhöhung der Epochen führt zu keiner weiteren Verbesserung der Genauigkeit.

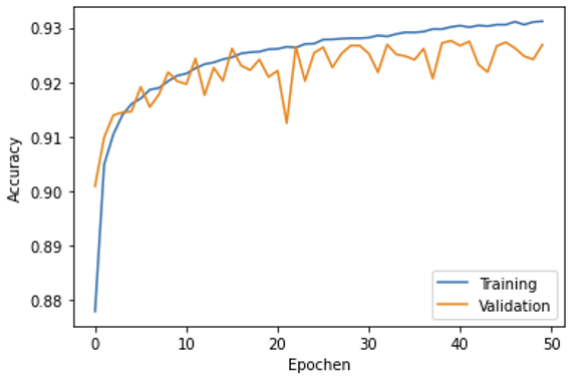


Abbildung 4.3: Verlauf der Genauigkeit (Accuracy) beim Training des neuronalen Netzes

4.4 Tiefenbildgenerierung

Für die Erstellung des gewünschten Tiefenbildes wird das Stereobildpaar mit der in Kapitel 3 vorgestellten Architektur analysiert.

Im ersten Schritt wird über die beiden Stereobilder iteriert und für jede Pixelposition die Matching Costs für alle möglichen Disparitäten ermittelt. Die Matching Costs werden im Semi-Global Matching weiter aufgearbeitet und anschließend die initialen Disparity Maps erstellt. Die initialen Disparity Maps werden im Left Right Consistency Check zusammengeführt und auf inkonsistente Stelle verglichen. Im nachfolgendem Hole Filling Scheme werden die ermittelten inkonsistente Stellen aufgearbeitet und die finale Disparity Map generiert. Die finale Disparity Map wird in das Tiefenbild überführt. Der Ablauf der Tiefenbildgenerierung ist schematisch in der Abbildung 4.4 dargestellt.

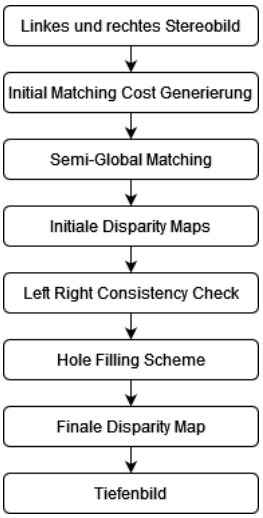


Abbildung 4.4: Schematischer Ablauf der Tiefenbildgenerierung

5 Evaluation

In diesem Kapitel werden die Ergebnisse der umgesetzten Systemarchitektur aus Kapitel 3 aufgezeigt und analysiert. Dafür werden die erzielten Ergebnisse in verschieden Stadien der Aufarbeitung dargestellt und eine Einordnung der Ergebnisse vorgenommen. Des Weiteren wird der Einfluss des Postprocessings auf die Ergebnisse analysiert und in fern Postprocessing für die entwickelte Systemarchitektur notwendig ist.

5.1 Ergebnisse des Systems

In diesem Abschnitt werden die Ergebnisse des neuronalen Netzes und der weiterführenden Aufarbeitung mittels des Postprocessing dargestellt.

Eine Qualitätseinschätzung bzw. Fehlerbewertung wird mittels Vergleich gegen die originalen Disparity Maps des Middlebury Stereo-Datensatzes 2006 erstellt. Sowohl im Middlebury Stereo Evaluation Ranking [1] als auch in verschiedenen Veröffentlichungen werden jeweils unterschiedliche Bildbereiche für die Fehlerbewertung untersucht. Für eine bessere Vergleichbarkeit wird die Fehlerbewertung für die beiden am häufigsten verwendeten Bildbereiche durchgeführt. Der erste Bildbereich ist das Gesamtbild mit allen bekannten Disparitäten der originalen Disparity Map. Dieser wird im Folgenden als Bildbereich 1 bezeichnet. Als Bildbereich 2 wird der Teil des Gesamtbildes bezeichnet, der nicht von Occlusion betroffen ist und für den eine bekannte Disparität vorliegt. Dieser Bereich wird durch Vergleich der beiden originalen Disparity Maps bestimmt.

Die Fehlerbewertung der Bildbereiche wird jeweils auf Basis des 1- und 2-Pixelfehlers erstellt. Dies bedeutet, dass ein Pixel als fehlerhaft ermittelt gilt, wenn der Disparitätsunterschied zum Original mehr als 1 bzw. 2 beträgt. Pixel mit einer unbekannten Disparität in der originalen Disparity Map ($d(p) = 0$) werden nicht in die Fehlerbewertung mit einbezogen. Diese Bereiche sind in den originalen Disparity Maps in schwarz

und in den Abbildungen der Bereiche 1 & 2 in Dunkelgrau dargestellt. Fehlerhafte Pixel bzw. Pixelbereiche sind in den Auswertungsbildern der Pixelfehler in rot markiert.

Im nachfolgenden Teil werden die Ergebnisse für 7 der 21 Stereobildpaare des Middlebury Stereo-Datensatzes 2006 in verschiedene Phasen der Analyse dargestellt. Die Auswahl wurde auf Basis der Bildtypen und der Anforderungen an das System getroffen. Unter den ausgewählten Bildern befinden sich unter anderem Bilder mit texturarmen Regionen (z. B. Abbildung 5.6), mit feinen Strukturen (z. B. Abbildung 5.1), mit gleichbleibenden texturreichen Regionen (z. B. Abbildung 5.3), mit vielen Textur- und Objektübergängen (z. B. Abbildung 5.5 und 5.7) und Mischformen (z. B. Abbildung 5.2). Als Namen dienen die Bezeichnungen des Middlebury Stereo-Datensatzes.

Als Erstes wird jeweils das linke originale Stereobild mit dazugehöriger Disparity Map als Ausgangs- bzw. Zielbild dargestellt. Darauf folgen die „rohen“ Ergebnisse des neuronalen Netzes für das linke und rechte Bild. Diese Ergebnisse wurden nicht weiter aufgearbeitet, sondern mittels der Maximumfunktion ermittelt. Als nächstes werden die initialen Disparity Maps nach dem Semi-Global Matching mit anschließender initialer Disparity Map Generierung dargestellt. In der nachfolgenden Zeile befinden sich das Ergebnisse des Left Right Consistency Checks und die finale Disparity Map nach dem Hole Filling Scheme. Die letzten beiden Zeilen zeigen die Auswertungsbilder mit markierten Fehlerregionen für beide Bildbereiche und beide Pixelfehler.

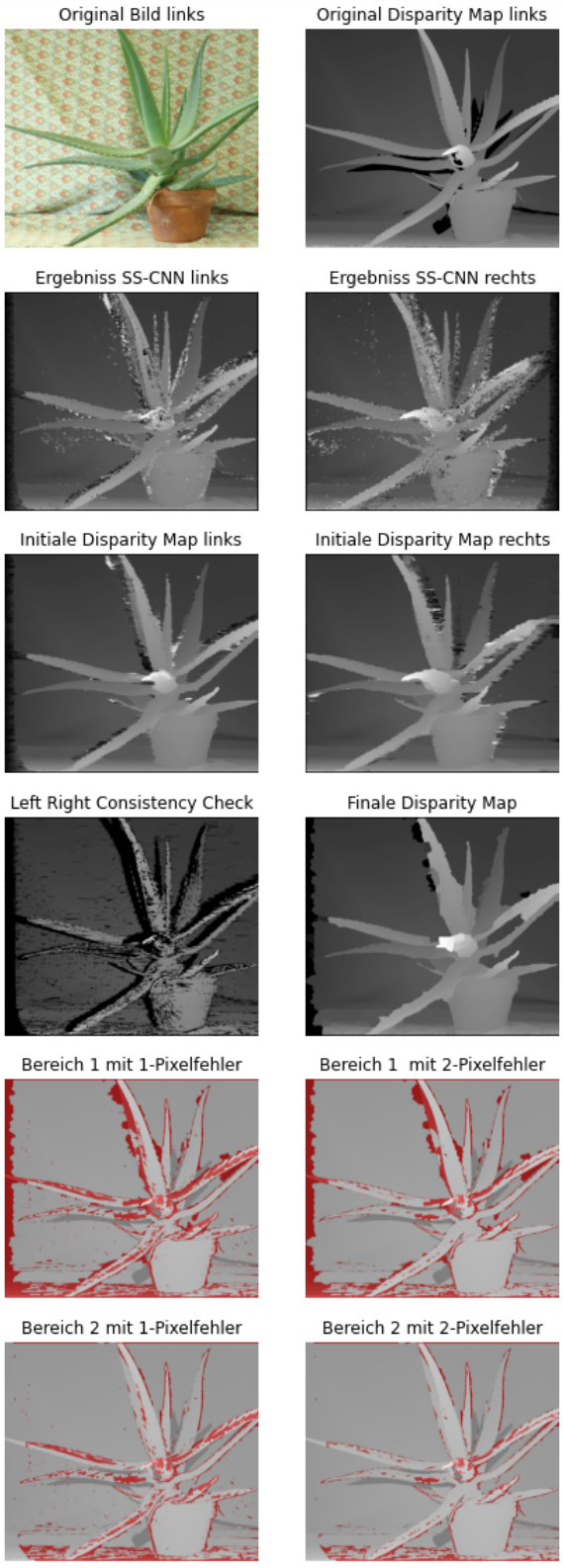


Abbildung 5.1: Ergebnisse für das Stereobild: Aloe

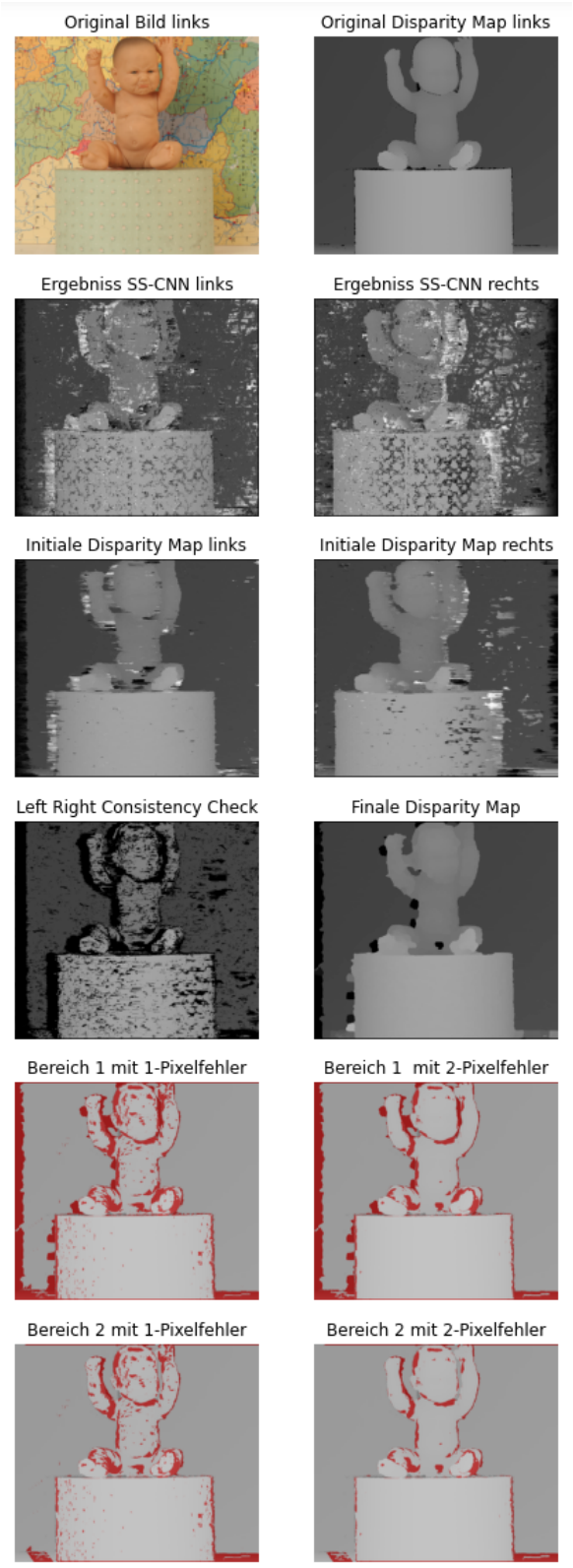


Abbildung 5.2: Ergebnisse für das Stereobild: Baby1

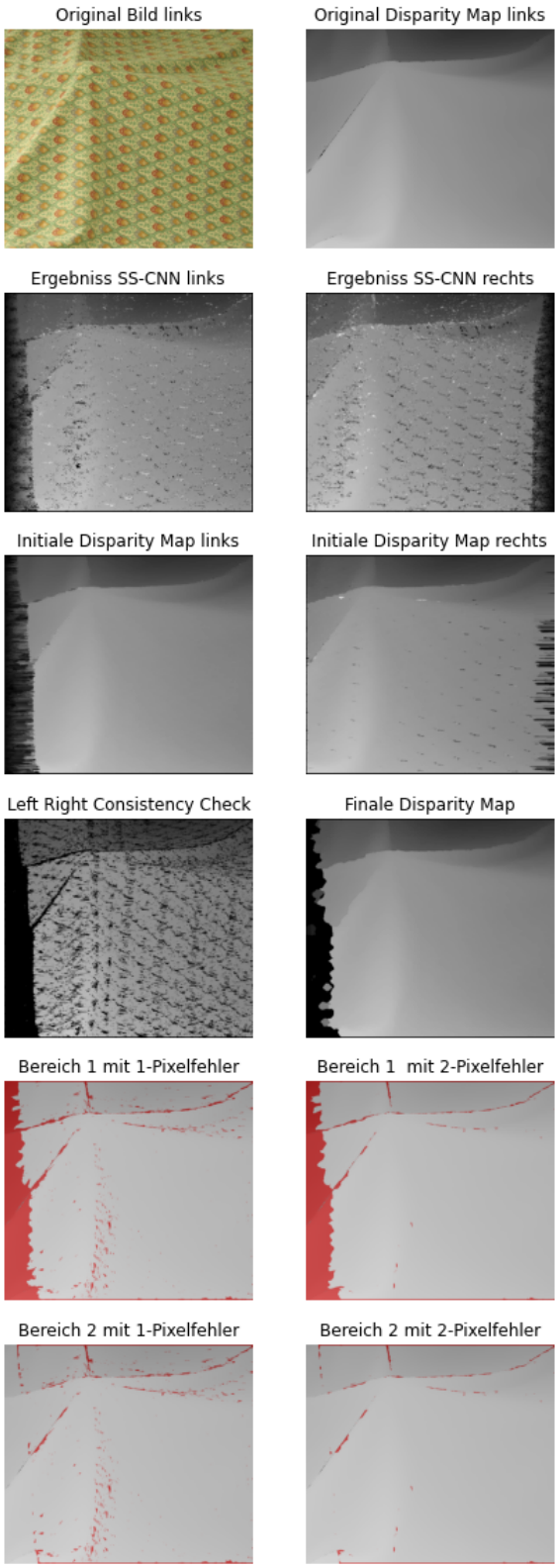


Abbildung 5.3: Ergebnisse für das Stereobild: Cloth1

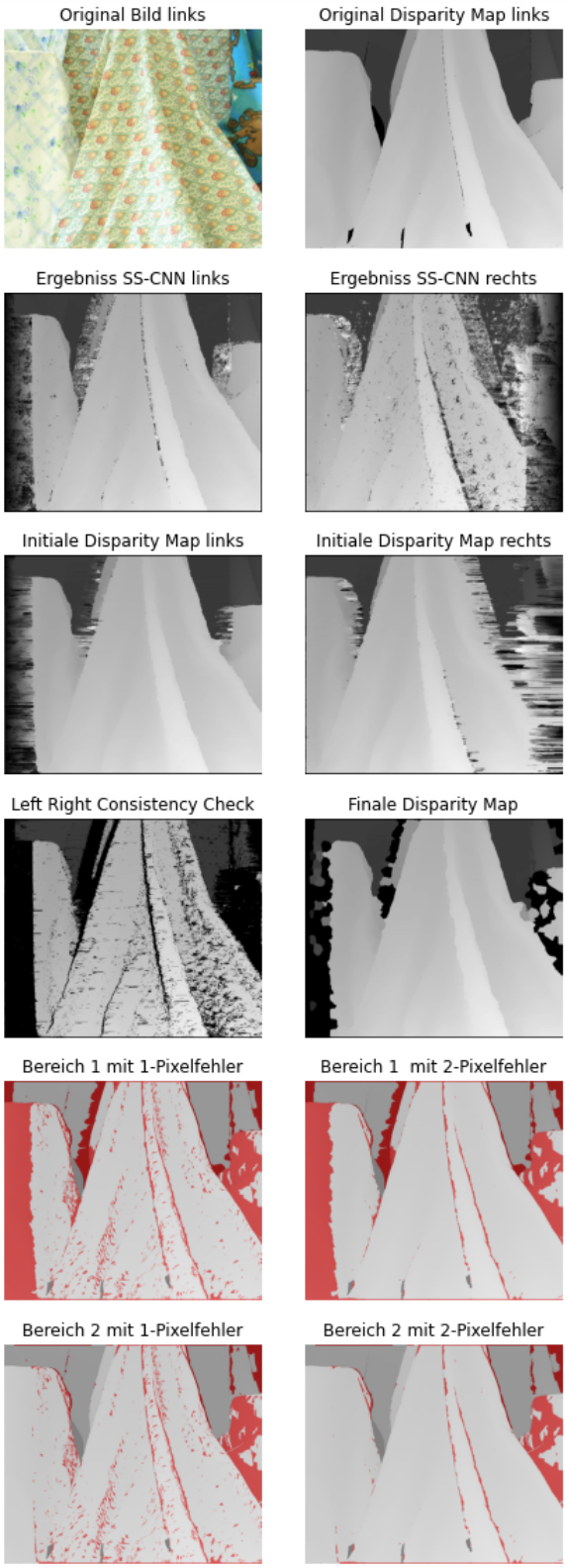


Abbildung 5.4: Ergebnisse für das Stereobild: Cloth4

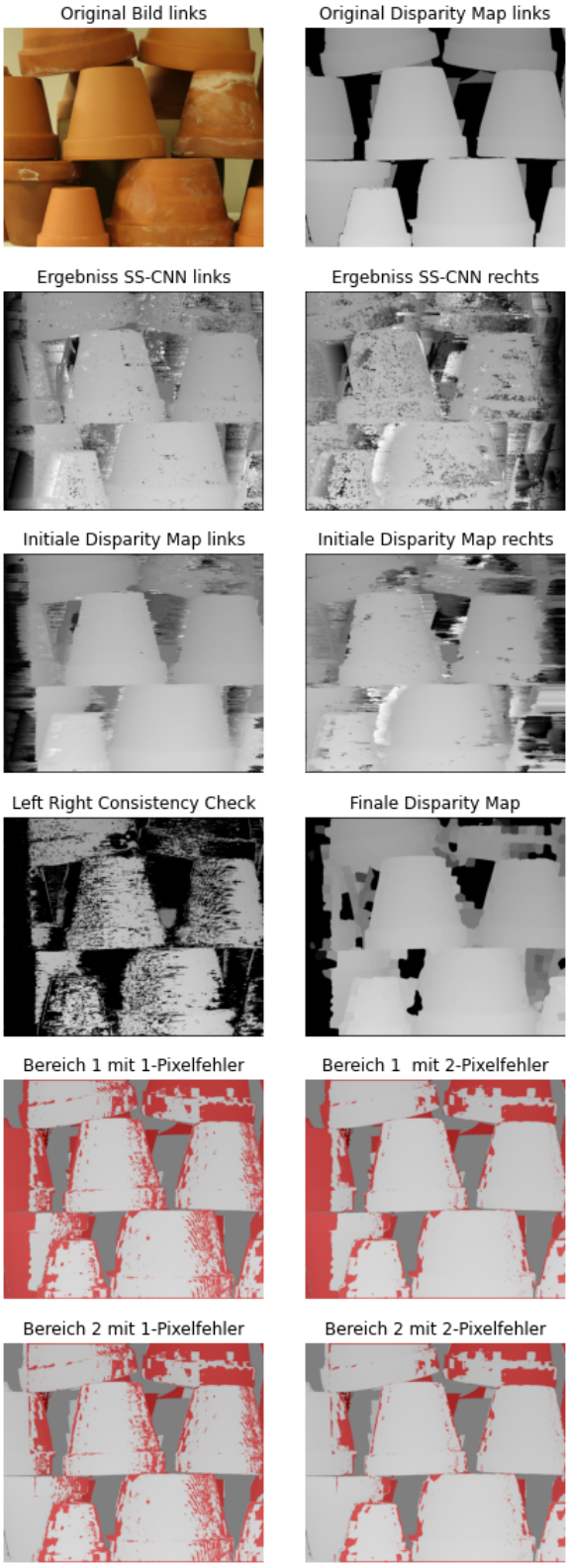


Abbildung 5.5: Ergebnisse für das Stereobild: Flowerpots

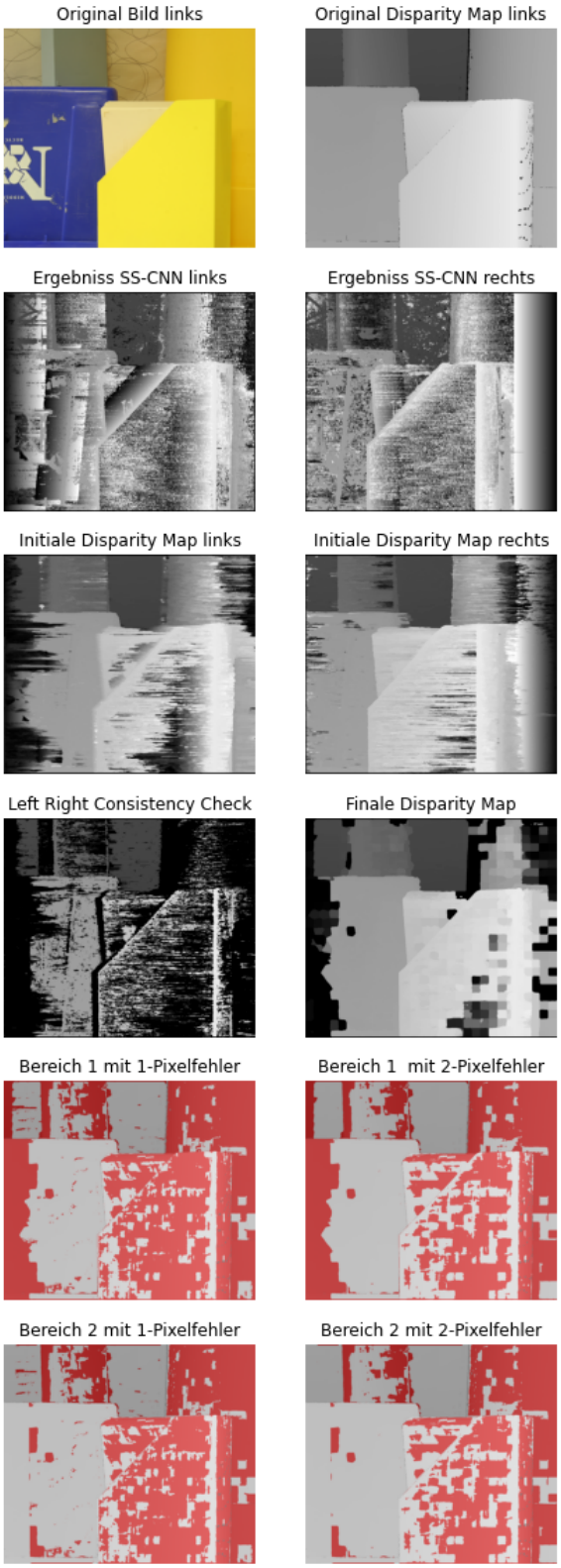


Abbildung 5.6: Ergebnisse für das Stereobild: Plastic

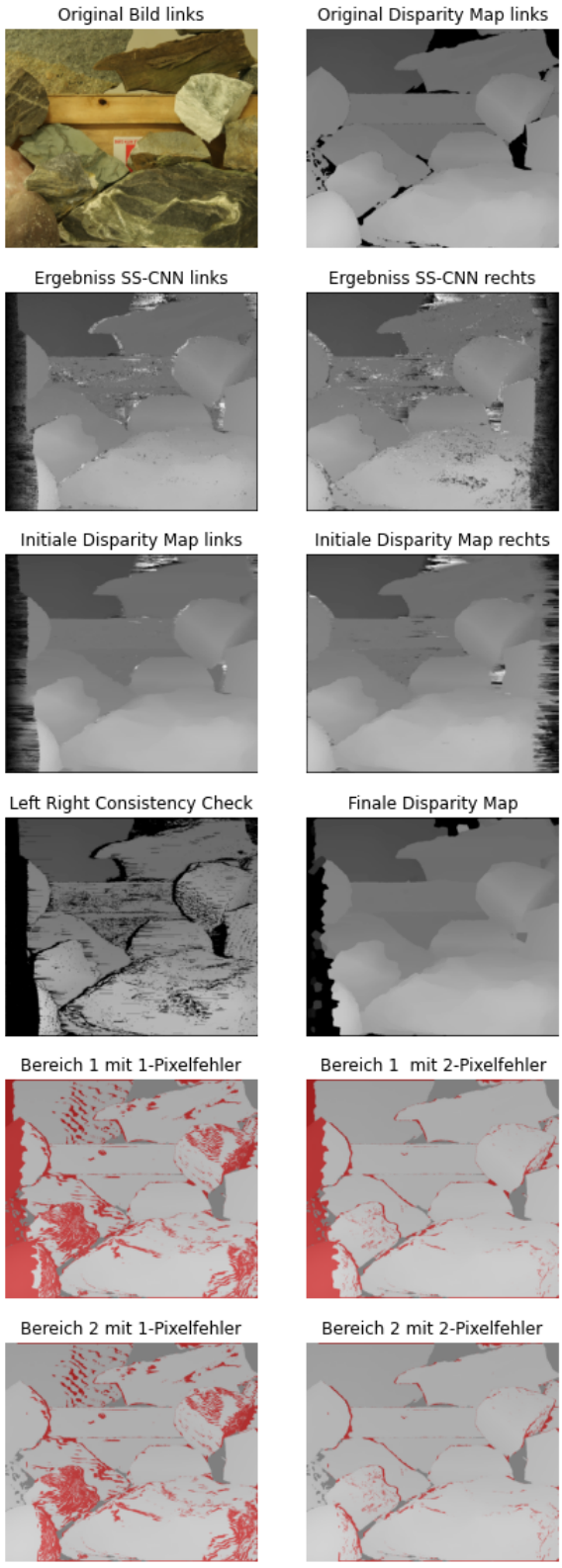


Abbildung 5.7: Ergebnisse für das Stereobild: Rocks1

Die finalen Fehlerraten der dargestellten Stereobilder sind in der nachfolgenden Tabelle 5.1 dargestellt. Die dargestellten Fehlerraten in Prozent spiegeln die rot markierten Bereiche der vorangegangen Abbildungen wieder.

Bildname	**Bildbereich 1**		**Bildbereich 2**	
	1-Pixelfehler	**2-Pixelfehler**	**1-Pixelfehler**	**2-Pixelfehler**
Aloe	17,26 %	13,35 %	11,41 %	7,37 %
Baby1	12,92 %	10,16 %	8,02 %	5,07 %
Cloth1	11,25 %	9,83 %	2,71 %	1,51 %
Cloth4	23,38 %	19,08 %	12,53 %	8,02 %
Flowerpots	36,92 %	25,68 %	29,45 %	16,88 %
Plastic	54,76 %	47,01 %	52,37 %	43,71 %
Rocks1	22,92 %	11,57 %	16,34 %	4,00 %

Tabelle 5.1: Pixelfehlerraten der analysierten Stereobilder

5.2 Einordnung der Ergebnisse

Für eine Einordnung des Verfahrens und seiner Ergebnisse wird ein Vergleich zu anderen Verfahren durchgeführt.

Im ersten Schritt wird ein Vergleich zur Veröffentlichung „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] vorgenommen. Das in dieser Arbeit in Kapitel 3 entwickelte System stellt eine modifizierte Variante dieser Veröffentlichung dar. Eine Gegenüberstellung der Ergebnisse ist für Bildbereich 1 (Gesamtbild) in Tabelle 5.2 und für Bildbereich 2 (Gesamtbild ohne Occlusion) in Tabelle 5.3 dargestellt.

Beim Vergleich der Ergebnisse des Bildbereich 1 zeigt sich, dass sich die Fehlerraten im gleichen Bereich bewegen und sich um wenige Prozent unterscheiden. Als Ausnahme kann das Bild Cloth4 bezeichnet werden. Für dieses Bild liegt eine deutlich höhere Abweichung im 1-Pixelfehler vor. Dieser Unterschied vermindert sich bei der Betrachtung des 2-Pixelfehlers. Des Weiteren zeigt sich, dass bei dem in dieser Arbeit entwickelten System eine deutlichere Verringerung des Fehlers im 2-Pixelfehlerbereich vorliegt.

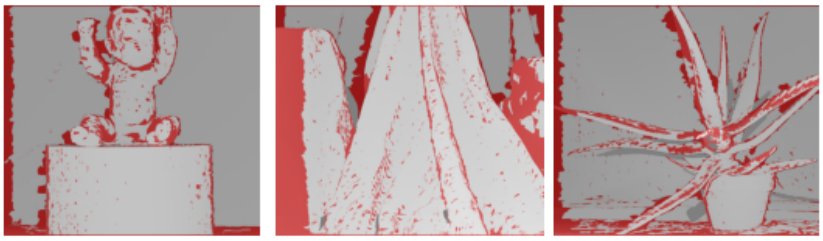
Bildname	Veröffentlichung		Entwickelte Systemarchitektur	
	1-Pixelfehler	2-Pixelfehler	1-Pixelfehler	2-Pixelfehler
Aloe	18,3 %	17,2 %	17,26 %	13,35 %
Baby1	11,6 %	10,3 %	12,92 %	10,16 %
Cloth4	17,6 %	16,9 %	23,38 %	19,08 %

Tabelle 5.2: Pixelfehlerraten für Bildbereich 1 im Vergleich zwischen „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] und der entwickelten Systemarchitektur

Die aus dem Vergleich der Fehlerraten gezogenen Schlüsse für Bildbereich 1 lassen sich durch Betrachtung der Fehlerbilder in Abbildung 5.8 belegen und weitere Unterschiede erkennen. Der Vergleich der Fehlerbilder des Systems und der Veröffentlichung weisen ähnliche Fehlerbereiche auf. Diese zentrieren sich sich größtenteils an Objektkanten, z. B. Übergang eines Blattes zum Hintergrund, und am linken Rand der Bilder. Ein allgemeiner Unterschied kann in den Bodenbereichen herausgestellt werden. Diese sind sowohl im 1-Pixelfehlerbereich als auch im 2-Pixelfehlerbereich in der entwickelten Systemarchitektur durch größere fehlerhafte Bereiche gekennzeichnet. Besonders fällt diese Tatsache in den Bodenbereichen der Aloe Bilder in Abbildung 5.8 auf. Die herausgestellte Differenz der Fehlerquote der Ergebnisse für Cloth4 befinden sich vor allem am rechten Rand. An dieser Stelle befindet sich im originalen Bild ein Übergang zwischen zwei verschiedenen Deckenmustern (siehe „Original Bild links“ in Abbildung 5.4). Dieser Übergang und das neue Muster wurden in der erstellen Systemarchitektur nur teilweise erkannt. Dies bezieht sich vor allem auf die rechte initiale Disparity Map (siehe Abbildung 5.4). In der linken initialen Disparity Map wurde diese Fläche gut erkannt. Die deutlichere Verbesserung der Fehlerquote bei der Betrachtung des 2-Pixelfehlers kann auf das Rauschen (Salt and Pepper Noise) im 1-Pixelfehlerbild zurückgeführt werden. Dieses Rauschen ist in der Ergebnissen der Veröffentlichung nicht vorhanden.

Allgemein kann auf den Fehlerbildern der Veröffentlichung erkannt werden, dass ebenfalls Pixel mit einer unbekannten Disparität im Ausgangsbild mit in die Berechnung und Markierung der Fehler eingeflossen sind. Diese wurden in den Ergebnissen des Systems in einem dunklen Grau hinterlegt und nicht mit in die Fehlerauswertung einbezogen, da keine Vergleichswerte in den originalen Disparity Maps vorliegen. Diese Bereich sind vor allem in den Aloe Bildern sichtbar. In den anderen Bildern gibt es nur wenige und kleine Bereiche. Für den Vergleich der Ergebnisse ergeben sich daraus leichte Verschiebungen der

ermittelten Fehlerwerte. Je nach Betrachtungsweise wäre der Wert der Veröffentlichung niedriger oder der des Systems höher. Die Anpassung würde vor allem das Aloe Bild betreffen, da dort ein Anteil von 3,45 % der Gesamtpixel eine unbekannte Disparität aufweisen. Durch diese Verschiebung näheren sich die Ergebnisse der beiden Verfahren für das Aloe Bild weiter an. Bei den beiden anderen Bildern liegen die Werte bei 0,65 % für Baby1 und bei 0,83 % für Cloth4. Für diese beiden Bilder würden sich die Ergebnisse leicht in Richtung der Veröffentlichung verschieben.



(a) Fehlerbild des Systems: 1-Pixelfehler

(b) Fehlerbild der Veröffentlichung: 1-Pixelfehler

(c) Fehlerbild des Systems: 2-Pixelfehler

Abbildung 5.8: Vergleich der Fehlerbilder des „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] und der entwickelten Systemarchitektur für Bildbereich 1. Bildnamen von links nach Rechts: Baby1, Cloth4 und Aloe

Für Bildbereich 2, Gesamtbild ohne Occlusion, ergeben sich beim Vergleich ähnliche Ergebnisse wie bei Bildbereich 1 (siehe Bildbereich 2 Ergebnisse in Tabelle 5.3). Die Werte für Baby1 und Cloth4 sind sehr ähnlich und haben sich ungefähr um den gleichen Wert

verschoben. Dies ist darauf zurückzuführen, dass beide Verfahren die Hauptfehlerquelle am linken Bildrand haben. Dieser Bereich ist im rechten Bild nicht zu sehen und entfällt dadurch bei der Betrachtung ohne Occlusion. Bei Cloth4 erklären sich die höheren Fehlerwerte aus dem fehlerhaften Bereich am rechten Rand. Diese Bildregion bleibt in Bildbereich 2 erhalten und erklärt die im Vergleich hohen Fehlerwerte von 12,53 % und 8,02 %. Bei der Aloe ist zu vermuten, dass die Diskrepanz auf der Berechnung mit unbekannten Disparitäten beruht und die Ergebnisse eigentlich nahezu identisch sind (genaue Erläuterung befinden sich im Teil von Bildbereich 1). Dieser Sachverhalt kann allerdings nicht genau geklärt werden, da die Veröffentlichung keine Fehlerbilder für diesen Bildbereich aufweist.

Bildname	**Veröffentlichung**		**Entwickelte Systemarchitektur**	
	1-Pixelfehler	**2-Pixelfehler**	**1-Pixelfehler**	**2-Pixelfehler**
Aloe	14,6 %	13,2 %	11,41 %	7,37 %
Baby1	6,0 %	4,6 %	8,02 %	5,07 %
Cloth4	4,9 %	4,0 %	12,53 %	8,02 %

Tabelle 5.3: Pixelfehlerraten für Bildbereich 2 im Vergleich zwischen „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] und der entwickelten Systemarchitektur

Der Vergleich der beiden Verfahren zeigt, dass beide Verfahren ähnliche Ergebnisse erzielen. Dieser Umstand war zu erwarten, da die in dieser Arbeit vorgestellt Architektur auf der Architektur der Veröffentlichung [29] beruht. Beide Verfahren zeigen ähnlich Schwachstellen an den Übergängen von Strukturen und am linken Bildrand. Die ermittelte Diskrepanz für das Cloth4 Bild beruht auf den schlechteren Ergebnissen des SS-CNN für das rechte Stereobild, wodurch eine große Region von Löchern im Left Right Consistency Check generiert wird (siehe „Initiale Disparity Map rechts“ in Abbildung 5.4). Diese große Region kann im Hole Filling Scheme nicht bzw. nur teilweise gefüllt werden, da dort kaum Werte für die Medianberechnung der Masken vorliegen. Durch einen größeren Testbildumfang könnte analysiert werden ob dies ein generelles Problem der umgesetzten Architektur ist oder ob es sich um einen zufälligen Fehler handelt. Ebenfalls könnte so analysiert werden, ob die Architektur der Veröffentlichung ähnliche Probleme bei anderen Bildern und Bildtypen aufwirft. Die Veröffentlichung stellt hierfür keiner weiteren Daten zur Verfügung.

Für eine weiterführende Einordnung des entwickelten Systems wird dieses gegen andere State of the Art Systeme bzw. Verfahren verglichen. Die Literaturrecherche hat ergeben, dass vielfach nicht neuronale Ansätze für die Analyse des Middlebury Stereo-Datensatzes 2006 gewählt werden und diese die besten Ergebnisse für diesen Datensatz erzielen. Um Vergleichswerte zu einem anderen Verfahren mit einer neuronalen Netzkomponente durchführen zu haben, wurde das MC-CNN von Zbontar und LeCun [34] in den Vergleich aufgenommen, obwohl die Endergebnisse im Durchschnitt schlechter als die der besten Verfahren sind. Das MC-CNN Verfahren stellt nach der durchgeführten Literaturrecherche einen der besten neuronalen Ansätze für diesen Datensatz dar. Die Fehlerwerte des MC-CNN wurde aus der Veröffentlichung „Edge-preserving guided filtering based cost aggregation for stereo matching“ [33] übernommen, da die originale Veröffentlichung des MC-CNN [34] keine Einzelfehlerwerte für die Bilder liefert. Des weiteren wurden die drei besten Verfahren der Literaturrecherche in den Vergleich aufgenommen. Bei den drei besten Verfahren handelt es sich um das EnSoft3D Verfahren aus der Veröffentlichung „Enhanced Soft 3D Reconstruction Method with an Iterative Matching Cost Update Using Object Surface Consensus“ [19], dem MFMSR Verfahren aus der Veröffentlichung „Stereo matching based on multi-scale fusion and multi-type support regions“ [20] und dem PMSC Verfahren aus der Veröffentlichung „PMSC: PatchMatch-Based Superpixel Cut for Accurate Stereo Matching“ [21]. Diese drei Verfahren erzielen über den Gesamtdatensatz der 21 Stereobildpaare ähnliche Fehlerwerte. Im Detail unterscheiden sich die Ergebnisse leicht je Bildtyp. In der nachfolgenden Tabelle 5.4 sind die Fehlerquoten der einzelnen Verfahren für den Bildbereich 2 (Gesamtbild ohne Occlusion) für den 1-Pixelfehler dargestellt. Für Zellen ohne Prozentwert konnten keine Fehlerwerte ermittelt werden.

Die in Tabelle 5.4 dargestellten Fehlerwerte zeigen, dass die entwickelte Architektur nicht mit den State of the Art Verfahren mithalten kann. Gerade bei Bildtypen mit vielen texturarmen Regionen, wie z.B. dem Bild Plastic (siehe Abbildung 5.6), liegt ein sehr große Diskrepanz vor. Die entwickelte Architektur zeigt dort schwächen. Aus den finalen Fehlerwerten der Vergleichsverfahren lässt sich ableiten, dass diese Bildtypen generell etwas schwieriger einzuordnen sind, da für diese Bilder im Vergleich zum Durchschnitt eine doppelt bis dreifach so große Fehlerquote vorliegt. Am meisten Potential für annähernd identische Werte kann in Bild Cloth1 gesehen werden. Dort liegt der ermittelte Wert des Systems immer noch deutlich über den anderen Ergebnissen jedoch in einem ähnlichen Bereich. Bei der Betrachtung des 2-Pixelfehlers für Cloth1 des entwickelten Systems näheren sich die Ergebnisse noch deutlicher an die 1-Pixelfehler der anderen Verfahren an.

Bildname	Eigenes System	MC-CNN	EnSoft3D	MFMSR	PMSC
Aloe	11,41 %	4,75 %	2,98 %	4,15 %	3,06 %
Baby1	8,02 %	3,14 %	1,49 %	1,40 %	1,98 %
Cloth1	2,71 %	0,85 %	0,34 %	0,71 %	0,60 %
Cloth4	12,53 %	1,27 %	0,86 %	0,79 %	1,87 %
Flowerpots	29,45 %	8,29 %	2,59 %	1,81 %	2,49 %
Plastic	52,37 %	-	-	3,15 %	4,40 %
Rocks1	16,34 %	2,21 %	1,04 %	1,28 %	1,80 %

Tabelle 5.4: Fehlerquotenvergleich des 1-Pixelfehlers für Bildbereich 2

Mit 1,51 % (siehe Tabelle 5.1) befindet sich das Ergebnis im Bereich von 1 % Unterschied zu den Vergleichsverfahren. Interessant wäre hier ein Vergleich zu den 2-Pixelfehlern der Vergleichsverfahren und wie sich dort die Fehlerquoten verändern. Diese Daten stehen jedoch nicht zur Verfügung.

Für das allgemeinen Erreichen ähnlicher Fehlerwerte müssen die vorhanden Probleme minimiert werden. Zu den Problemen des System gehören Fehler in einfarbigen texturarmen Regionen (siehe z. B. Abbildung 5.6 Plastic), Fehler in Übergänge zwischen verschiedenen Texturen (siehe z. B. Abbildung 5.1 Aloe), Fehler in Bildrandbereiche (siehe z. B. Bodenbereich in Abbildung 5.2 Baby1) und Qualitätsunterschiede zwischen den Disparity Maps des linken und rechten Bildes (siehe z. B. Abbildung 5.2 Baby1). Zur Behebung dieser Problematiken müssen weitere Schritte ins Postprocessing aufgenommen und vorhandene Schritte überarbeitet werden. Beispielsweise könnte durch zusätzliche Interpolation im Hole Filling Scheme lückenhafte Regionen gefüllt werden, wodurch bessere Ergebnisse in texturarmen und verdeckten Regionen erzielt werden können. Interpolation wird unter anderem im MC-CNN [34] zur Konfliktminimierung zwischen den Disparity Maps des linken und rechten Bildes verwendet. Durch eine zusätzliche Kantendetektion, beispielsweise mit dem Sobel-Operator [24] oder dem Canny-Operator [24], könnten die Machting Costs für eine Disparität auf Basis von Übergängen verfeinert werden. Durch diese Maßnahme könnten die Fehler in Übergängen zwischen verschiedenen Texturen oder Regionen verbessert werden. Die Probleme an den Bildrandbereichen könnten durch explizierte Trainingsdaten für diese Fälle verbessert werden. Trainingsdatenbilder für dieses Problem würden aus teilweise schwarzen Bereichen gepaart mit Farbbereichen bestehen,

z. B. würde die obere Hälfte des Bildes aus Textur und Farbe bestehen und die untere Hälfe wäre komplett schwarz. Die allgemeine Diskrepanz zwischen der linken und rechten Disparity Map des neuronalen Netzes und die allgemeinen erzielten Ergebnisse des Verfahrens könnten durch Training mit weiteren Trainingsdaten vermindert werden.

Im Gegensatz zu der Einschätzung der Veröffentlichung „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] können die erzielten Ergebnisse nicht als vergleichbar zu den Ergebnisse der State of the Art Verfahren eingeschätzt werden. Die Ergebnisse der Veröffentlichung weisen ähnliche Fehlerregionen wie die entwickelte Systemarchitektur auf. Zum Erreichen vergleichbarer Fehlerwerte zu den State of the Art Verfahren müssen die dargelegten Probleme des Verfahrens verbessert werden. Allgemein zeigt sich das Potential des entwickelten Systems am Beispielbild Cloth1. Für dieses Bild werden ähnlich niedrige Fehlerraten ermittelt.

Bezogen auf mögliche Anwendungsfälle kann durchaus eine anderen Einschätzung getroffen werden. Bei allen Bildern zeigt sich, dass das entwickelte System visuell betrachtet die Hauptbestandteile der Bilder ermitteln kann. Beispielsweise ist die Pflanze auf dem Bild Aloe oder die Strukturverläufe der beiden Bilder Cloth1 und Cloth4 deutlich zu erkennen. Vor dem Hintergrund einer Roboternavigation oder der Navigation eines Autos sind einzelne Pixelfehler in Regionen oder an Ränder von Regionen als nicht entscheidend einzuordnen. Für solche Anwendungsfälle ist die allgemeine Erkennung von Strukturen wichtiger als einzelne Pixelfehler innerhalb oder an Rändern von Strukturen. Für Anwendungsfällen, die eine hohe Genauigkeit erfordern, kann das vorgestellte Verfahren keine gute Werte liefern.

5.3 Einfluss des Postprocessings auf die Ergebnisse

In diesem Abschnitt wird der Einfluss des Posprocessings auf die erzielten Ergebnisse analysiert und in Vergleich zu den „rohen“ Ergebnissen des neuronalen Netzes (SS-CNN) gesetzt. Auf Basis des Vergleichs wird eine Einschätzung über die Notwendigkeit des Postprocessings vorgenommen.

In der Tabelle 5.5 sind die Fehlerquoten für das linke Bild in verschiedenen Phasen des Systems dargestellt. Für eine bessere Vergleichbarkeit der Ergebnisse wird der Bildbereich 2 (ohne Occlusion) betrachtet, da nicht in beiden Bildern vorhandene Bereiche nicht vom

neuronalen Netz bestimmt werden können und das entwickelte System allgemein ein Problem in diesen Bereichen aufweist. Als Phasenergebnisse dienen die erzielten Fehlerwerte direkt nach dem neuronalen Netz ohne weitere Aufarbeitung (SS-CNN), die Ergebnisse nach dem Semi-Global Matching (Initial Disparity Map) und die Ergebnisse nach dem Hole Filling Scheme (Final Disparity Map). Die Ergebnisse nach dem Left Right Consistency Check werden nicht gesondert betrachtet, da dieser Schritt die Vorstufe für das Hole Filling Scheme ist und zu keiner Verbesserung der Ergebnisse führen kann.

Die Ergebnisse aus Tabelle 5.5 zeigen unterschiedliche Auswirkungen auf die erzielten Fehlerquoten je Bildtyp. Bei den Bildern Baby1, Cloth1 und Plastic zeigt sich eine deutliche Verminderung der Fehlerquoten in beiden Fehlerbereichen mit fortgeschrittener Nachbearbeitung. Die Bilder Baby1 und Cloth1 weisen dabei die deutlichsten Verbesserungen auf. Bei Baby1 vermindert sich die Fehlerquote um 72,19 % und bei Cloth1 um 80,79 % über den gesamten Vorgang der Nachbearbeitung. Bei Plastic liegt ein weniger stark ausgeprägt Verbesserung im Postprocessing vor. Hier zeigt sich die in der Einordnung (Abschnitt 5.2) dargestellte Schwäche in texturarmen Regionen. Das Postprocessing ist nicht in der Lage die texturarmen Regionen mittels Left Right Consistency Check und Hole Filling Scheme vollständig zu füllen. Des Weiteren ergeben sich leichte Disparitätsschwankungen innerhalb der gefüllten texturarmen Regionen aus dem Hole Filling Scheme mittels Superpixeloptimierung. Durch die viele Löcher in diesen Regionen ergeben sich bei den Medianberechnungen innerhalb eines Superpixels mit der Zeit leichte Verschiebungen. Diese Disparitätsschwankungen sind durch Grautonschwankungen in Abbildung 5.9 im rot markierten Kasten zu erkennen.

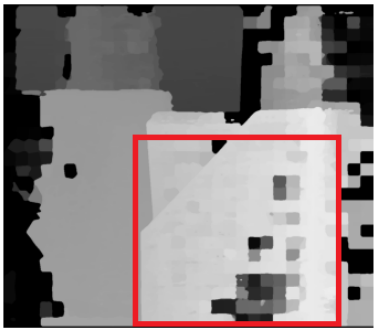


Abbildung 5.9: Problem in texturarmen Regionen am Beispiel Plastic

In die gleiche Kategorie wie Baby1, Cloth1 und Plastic können auch die Bilder Cloth4 und Flowerpots eingeordnet werden. Die dargestellten Fehlerwerte zeigen eigentlich gegenläufiges Verhalten zu den Fehlerwerten der anderen Bilder. Diese Beobachtung beruht

Bildname	SS-CNN		Initial		Final	
	≤ 1	≤ 2	≤ 1	≤ 2	≤ 1	≤ 2
Aloe	8,91 %	6,83 %	11,36 %	6,64 %	11,41 %	7,37 %
Baby1	28,84 %	25,16 %	14,74 %	9,85 %	8,02 %	5,07 %
Cloth1	14,11 %	10,27 %	4,08 %	1,78 %	2,71%	1,51 %
Cloth4	3,15 %	2,47 %	9,31 %	3,78 %	12,53%	8,02 %
Flowerpots	26,61 %	22,69 %	29,95 %	16,38 %	29,45 %	16,88 %
Plastic	80,23 %	77,18 %	60,98 %	53,48 %	52,37 %	43,71 %
Rocks1	7,47 %	5,19 %	17,01 %	4,42 %	16,34 %	4,00 %

Tabelle 5.5: Fehlerquoten für das linke Bild in verschiedenen Phasen der Aufarbeitung. Werte nach dem SS-CNN, nach der Initialen Disparity Map Generierung und die Finale Disparity Map für die Pixelfehler ≤ 1 und ≤ 2 im Bereich 2 (ohne Occlusion).

allerdings nur auf der Betrachtung der linken Disparity Maps nach dem SS-CNN und der Initialen Disparity Map Generierung und nicht auf der Betrachtung beider Disparity Maps. Für die Bilder Cloth4 und Flowerpots ergeben sich für den 1-Pixelfehler der rechten Disparity Maps nach dem SS-CNN ein Fehlerwert von 18,52 % für Cloth4 und 52,19 % für Flowerpots. Nach der Initialen Disparity Map Generierung ein Wert von 14,92 % für Cloth4 und 40,74 % für Flowerpots. Für das nachfolgende Postprocessing werden beide Disparity Maps zusammengeführt, wodurch ein höherer Gesamtfehler entsteht. Aus diesem höheren Gesamtfehler wird die finale Disparity Map generiert, die eine Fehlerreduktion aufweist. Auf Grund dieser Tatsache täuschen die Ergebnisse der Tabelle 5.5 für Cloth4 und Flowerpots. In der Abbildung 5.10 ist die Entwicklung des Fehlerbildes für Flowerpots als Beispiel dargestellt. Sie veranschaulicht die Unterschiede in den Fehlerbereichen zwischen der linken und rechten Disparity Map. Das aufgezeigte Phänomen des deutlichen Disparitätsunterschied zwischen den Disparity Maps nach dem SS-CNN und der initialen Disparity Map Generierung kann auch gedreht vorkommen, sodass die linke Disparity Map die schlechteren Fehlerwerte aufweist. Aus diesem Grund ist eine beidseitige Betrachtung der Disparity Maps notwendig.

Die beiden Bilder Aloe und Rocks1 zeigen hinsichtlich der Fehlerquoten ein anderes Verhalten als die anderen Bilder. Bei beiden Bildern steigt der 1-Pixelfehler im Übergang vom SS-CNN zur initialen Disparity Map an, während der 2-Pixelfehler sinkt. Im Rocks1

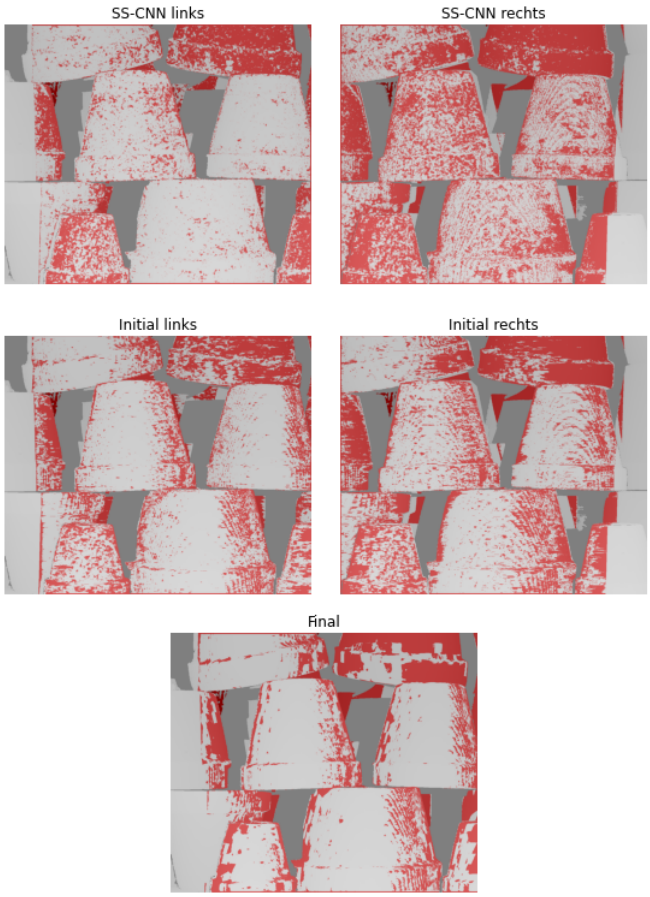


Abbildung 5.10: Fehlerbereiche für Flowerpots in verschiedenen Phasen der Aufarbeitung

Bild sinken der 1- und 2-Pixelfehler im letzten Schritt der Aufarbeitung weiter. Im Gegensatz dazu steigen die Werte des Aloe Bildes leicht an.

Der allgemeine Anstieg beider Bilder beim ersten Schritt des Postprocessings (Initiale Disparity Map Generierung) ist auf die Bestimmung der Ähnlichkeit (Similarity Score) durch das neuronale Netz und der Aufarbeitung mittels Semi-Global Matching zurückzuführen. Bei der Analyse und der Bestimmung der Ähnlichkeit durch das neuronale Netz gibt es Pixelpositionen für die keine exakte Pixelposition im Partnerbild vorhanden ist. Für diese Postionen liegt der wahre Wert der Verschiebung zwischen zwei Pixeln, sodass keine Pixelposition genau passt. Bestimmt nun das neuronale Netz den im Vergleich zur originalen Disparity Map anderen Wert, ergibt sich automatisch eine Abweichung von einem Pixel. Dieses Problem verstärkt sich in Regionen mit gleichartigen Strukturen und Farben. In diesen Regionen kann es dazu kommen, dass das neuronale Netz einer weiter

entfernen Pixelposition die höchste Ähnlichkeit gibt. Beim Semi-Global Matching werden die ermittelten Disparitäten mittels Energiefunktionsminimierung (genaue Erläuterung siehe Abschnitt 3.2.1) in Abhängigkeit ihrer Nachbarn aufgearbeitet. In Bildbereichen mit einer hohen Dichte an Pixeln mit einem Disparitätsunterschied von zwei wird im Semi-Global Matching in Richtung dieser Werte optimiert. Durch diese Optimierung verschieben sich die Ergebnisse des Semi-Global Matchings in diese Richtung und der 1-Pixelfehler steigt, während der 2-Pixelfehler sinkt. Die weitere Verbesserung zwischen der Initialen und Finalen Disparity Map bei Rocks1 ist auf das Füllen von vorhanden Löchern und dem Entfernen von Rauschen (Salt and Pepper Noise) mittels Medianfilter zurückzuführen.

Allgemein bietet das Postprocessing das Potential einer großen Ergebnisverbesserung von bis zu 80 % im Vergleich zu den Ausgangswerten des SS-CNN. Gleichzeitig zeigt sich eine Abhängigkeit zwischen der Qualität der Ergebnisse des neuronalen Netzes und der Qualität und des Verbesserungspotential des Postprocessings besteht. Werden im neuronalen Netz keine gute oder verschobene Ergebnisse erzielt, optimiert das Postprocessing entlang dieser ermittelten Werte und die Ergebnisverbesserung ist gehemmt oder rückläufig. Für eine optimale Ergebnisverbesserung mittels Postprocessing muss das neuronale Netz konsistente Ergebnisse für das linke und rechte Bild liefern. Verbesserungspotential besteht vor allem in der Aufarbeitung texturarmer Regionen. In diesen Regionen müssen Löcher besser gefüllt und einheitlichere Werte ermittelt werden.

Auf Basis der genannten und dargestellten Argumente ist das Postprocessing als notwendig für die Generierung bestmöglicher Ergebnisse einzuschätzen und bietet vor allem Vorteile. Für Anwendungsfälle, in denen nicht die bestmögliche Fehlerquote erreicht werden muss, sondern es um das grundsätzliche Erkennen von beispielsweise Hindernissen geht, kann durchaus auf das Postprocessing oder Teile des Postprocessings verzichtet werden.

6 Fazit

6.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein neuer Ansatz zur Bestimmung von Tiefenbildern aus Stereobildpaaren mittels eines neuronalen Netzes umgesetzt und untersucht. Im Gegensatz zu den üblichen neuronalen Ansätzen verwendet dieser Ansatz keine siamesische Netzwerkstruktur. Bei einer siamesische Netzwerkstruktur wird jedes der beiden Stereobilder in einem eigenen neuronalen Netz analysiert, die Ergebnisse zusammengeführt und diese weiter aufgearbeitet. In der umgesetzt Architektur wird das Stereobildpaar aufeinander gestapelt und von einem einzigen neuronalen Netz mit nachgelagertem Postprocessing analysiert.

Der Vergleich mit den Ergebnisse der Referenzarchitektur [29] zeigt, dass die umgesetzt und modifizierte Variante ähnliche Ergebnisse für die Bestimmung von Tiefenbildern erreicht. Im Vergleich zu den führenden Verfahren des untersuchten Middlebury Stereo-Datensatz 2006 zeigen sich größere Abweichungen. Zur Generierung ähnlicher Ergebnisse müssen die vorhandenen Schwächen bei der Bestimmung in texturarmen Regionen, an den Übergängen von Texturen, an den Bildrandbereichen und die Diskrepanzen zwischen den generierten Disparity Maps des neuronalen Netzes verbessert werden. Erfolgt die Betrachtung der Ergebnisse in Bezug zu möglichen Anwendungsfällen zeigt sich, dass die Hauptstrukturen in allen Bildern ermittelt werden konnten. In Anwendungsfällen, die keine bestmöglichen Ergebnisse erfordern, kann die umgesetzte Architektur bereits gute Ergebnisse liefern.

Bei der Untersuchung des Einflusses des Postprocessings auf die Ergebnisse des Verfahrens hat sich gezeigt, dass eine Verbesserung von bis zu 80 % im Vergleich zu den Ausgangswerten des neuronalen Netzes erreicht werden können. Gleichzeitig wurde eine Relation zwischen der Qualität der Ergebnisse des neuronalen Netzes und der Ergebnisse

nach dem Postprocessing ermittelt. Die durch das neuronale Netz ermittelten Ergebnisse definieren, in welchen Rahmen sich die Ergebnisse des Postprocessings verbessern oder verschlechtern können. Diese Relation betont die Wichtigkeit der Ergebnisse des neuronalen Netzes. Diese bilden das Fundament der Ergebnisqualität des Verfahrens.

In Hinblick auf die Zielsetzung in Abschnitt 1.2 konnte die in der Veröffentlichung „Patch Based Stereo Matching Using Convolutional Neural Network“ [29] vorgestellte Architektur umgesetzt und modifiziert werden. Die erzielten Ergebnisse der umgesetzten und modifizierten Variante decken sich mit denen der Veröffentlichung. Im Gegensatz zur Einschätzung der Veröffentlichung [29] können die erzielten Ergebnisse nicht als vergleichbar zu den State of the Art Verfahren eingeschätzt werden. Dennoch zeigt die Architektur Potential für Ergebnisverbesserungen oder in der differenzierten Betrachtung je Anwendungsfall.

6.2 Ausblick

Im Folgenden soll ein Ausblick zu möglichen Modifikationen und Erweiterungen dieser Arbeit gegeben werden.

Erweiterung der Daten- und Evaluationsbasis

Im Rahmen dieser Arbeit wurde das gesamte Training und die Auswertung auf dem Middlbury Stereo-Datensatz 2006 ausgeführt. Dies hatte den Hintergrund, dass in der Referenzveröffentlichung nur für diesen Datensatz Vergleichswerte vorliegen. Dieser Umstand ermöglicht eine Erweiterung der Datenbasis für Training und Evaluation. Von den Middlebury Stereo-Datensätzen liegen noch fünf weitere Datensätze vor, wovon zwei neuer als der Stereo-Datensatz 2006 sind. In den neueren Stereo-Datensätzen werden neue Bildtypen eingeführt, die räumlichere Szenen wie Klassenräume oder Wohnzimmer darstellen.

Des Weiteren ergibt sich im Rahmen der Datenbasis die Erweiterung auf die vielfach genutzten KITTI Stereo-Datensätze [22], welcher Stereobilder von verschiedenen Straßenszenen beinhalten. Durch diese Erweiterung könnte eine weitere Verallgemeinerung des entwickelten Modells erreicht werden.

Sowohl für die neueren Middlebury Stereo-Datensätze als auch die KITTI Stereo-Datensätze gibt es aktuelle Evaluationsranglisten, die eine genauere Einordnung des Verfahrens ermöglichen.

Optimierung des Verfahrens

Die Ergebnisanalyse des Verfahrens hat gezeigt, dass die ermittelten Ergebnisse schlechter als die der State of the Art Verfahren sind. In weiterführender Arbeit können die aufgeführten Schwächen in texturarmen Regionen, an Übergängen von Texturen und an den Bildrandbereichen gezielt minimiert werden. Die Optimierung sollte dafür sowohl im Bereich des neuronalen Netzes als auch im Postprocessing angesetzt werden.

Im Bereich des neuronalen Netzes bieten sich vor allem Möglichkeiten in der bereits dargelegten Erweiterung der Datenbasis und in der weiteren Modifikation der Netzstruktur und seiner Parameter. Eine weiterführende Merkmalextraktion (Feature extraction), z. B. durch zusätzliche Faltungs-Layer, könnten zu weiteren Verbesserung führen.

Hinsichtlich des Postprocessings könnten weitere Schritte, wie Interpolation oder Kantendetektion, aufgenommen werden. Zusätzlich sollten die Auswirkungen der verwendeten Hyperparameter in den einzelnen Schritten des Postprocessings genauer analysiert und angepasst werden.

Zusätzlich zu den Optimierungen des neuronalen Netzes und des Postprocessings können Optimierungen im Bereich der technischen Umsetzung vorgenommen werden. Diese Optimierungen sollen die Laufzeiten der Auswertung eines Stereobildpaares verringern. Beispiele für Ansätze zur Laufzeitoptimierung sind die Minimierung der Anwendung von Schleifen oder das Auswerten größerer Datenblöcke auf einmal.

Auswirkungen von Auflösungsänderungen

Erste Versuche im Bereich der Auflösungsänderungen der Ausgangsbilder haben gezeigt, dass durch die Halbierung der Auflösung der Stereobilder ein Geschwindigkeitsgewinn von ungefähr 70 % im Vergleich zur Vollauflösung vorliegt. In weiterführenden Schritten müssen die erzielten Ergebnisse für die halbierte Auflösung untersucht und die Auswirkungen, der für die Vollauflösung herausgestellten Probleme, untersucht werden. Weiterführend kann die Untersuchung auf weitere Auflösungsformate erweitert werden.

Literaturverzeichnis

[1] *Middlebury Stereo Datasets*. https://vision.middlebury.edu/stereo/. – [Online; zuletzt besucht am 17.05.2022]

[2] *Optische Täuschungen – wenn das Gehirn uns einen Streich spielt*. https://www.blickcheck.de/auge/funktion/optische-taeuschungen. – [Online; zuletzt besucht am 30.04.2022]

[3] Abadi, Martín u. a.: *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. – URL https://www.tensorflow.org/. – Software available from tensorflow.org

[4] Achanta, Radhakrishna ; Shaji, Appu ; Smith, Kevin ; Lucchi, Aurelien ; Fua, Pascal ; Süsstrunk, Sabine: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (2012), Nr. 11, S. 2274–2282

[5] Bogdanova, Rositsa ; Boulanger, Pierre ; Zheng, Bin: Depth Perception of Surgeons in Minimally Invasive Surgery. In: *Surgical Innovation* 23 (2016), 03

[6] Bottou, Léon: Large-Scale Machine Learning with Stochastic Gradient Descent. In: *Proc. of COMPSTAT* (2010), 01. ISBN 978-3-7908-2603-6

[7] Buduma, Nikhil ; Locascio, Nicholas: *Fundamentals of deep learning*. O´Reilly, 2017. – xii, 283 Seiten S. – ISBN 978-1-4919-2561-4

[8] Chollet, François u. a.: *Keras*. https://keras.io. 2015

[9] Dumoulin, Vincent ; Visin, Francesco: *A guide to convolution arithmetic for deep learning*. 2016. – URL https://arxiv.org/abs/1603.07285

[10] Döbel, Inga ; Leis, Miriam ; Vogelsang, Manuel M. ; Neustroev, Dmitry ; Petzka, Henning ; Riemer, Annamaria ; Rüping, Stefan ; Voss, Angelika ; Wegele, Martin ; Welz, Juliane: Maschinelles Lernen: Kompetenzen, Forschung, Anwendung / Fraunhofer-Gesellschaft. 2018. – Forschungsbericht

[11] Ertel, W.: *Grundkurs Künstliche Intelligenz: Eine praxisorientierte Einführung*. Springer Fachmedien Wiesbaden, 2016 (Computational Intelligence). – ISBN 9783658135492

[12] Goodfellow, Ian ; Bengio, Yoshua ; Courville, Aaron: *Deep Learning*. MIT Press, 2016. – http://www.deeplearningbook.org

[13] Hirschmuller, Heiko: Stereo Processing by Semiglobal Matching and Mutual Information. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30 (2008), Nr. 2, S. 328–341

[14] Ioffe, Sergey ; Szegedy, Christian: *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. – URL https://arxiv.org/abs/1502.03167

[15] Kingma, Diederik ; Ba, Jimmy: Adam: A Method for Stochastic Optimization. In: *International Conference on Learning Representations* (2014), 12

[16] Kluyver, Thomas ; Ragan-Kelley, Benjamin ; Pérez, Fernando ; Granger, Brian ; Bussonnier, Matthias ; Frederic, Jonathan ; Kelley, Kyle ; Hamrick, Jessica ; Grout, Jason ; Corlay, Sylvain ; Ivanov, Paul ; Avila, Damián ; Abdalla, Safia ; Willing, Carol: Jupyter Notebooks – a publishing format for reproducible computational workflows. In: Loizides, F. (Hrsg.) ; Schmidt, B. (Hrsg.): *Positioning and Power in Academic Publishing: Players, Agents and Agendas* IOS Press (Veranst.), 2016, S. 87 – 90

[17] Krizhevsky, Alex ; Sutskever, Ilya ; Hinton, Geoffrey E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F. (Hrsg.) ; Burges, C.J. (Hrsg.) ; Bottou, L. (Hrsg.) ; Weinberger, K.Q. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 25, Curran Associates, Inc., 2012. – URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[18] LeCun, Yann: *The MNIST Database of handwritten digits*. http://yann.lecun.com/exdb/mnist/. – [Online; zuletzt besucht am 30.04.2022]

[19] Lee, Min-Jae ; Um, Gi-Mun ; Yun, Joungil ; Cheong, Won-Sik ; Park, Soon-Yong: Enhanced Soft 3D Reconstruction Method with an Iterative Matching Cost Update Using Object Surface Consensus. In: *Sensors* 21 (2021), Nr. 19. – URL https://www.mdpi.com/1424-8220/21/19/6680. – ISSN 1424-8220

[20] Li, Haibin ; Gao, Yakun ; Huang, Ziyue ; Zhang, Yakun: Stereo matching based on multi-scale fusion and multi-type support regions. In: *J. Opt. Soc. Am. A* 36 (2019), Sep, Nr. 9, S. 1523–1533. – URL http://opg.optica.org/josaa/abstract.cfm?URI=josaa-36-9-1523

[21] Li, Lincheng ; Zhang, Shunli ; Yu, Xin ; Zhang, Li: PMSC: PatchMatch-Based Superpixel Cut for Accurate Stereo Matching. In: *IEEE Transactions on Circuits and Systems for Video Technology* 28 (2018), Nr. 3, S. 679–692

[22] Menze, Moritz ; Heipke, Christian ; Geiger, Andreas: Object Scene Flow. In: *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)* (2018)

[23] Nilsson, Nils J.: Introduction to Machine Learning, 1996

[24] Priese, L.: *Computer Vision: Einführung in die Verarbeitung und Analyse digitaler Bilder*. Springer Berlin Heidelberg, 2015 (eXamen.press). – URL https://books.google.de/books?id=MgycCgAAQBAJ. – ISBN 9783662451298

[25] Saxena, Ashutosh ; Schulte, Jamie ; Ng, Andrew: Depth Estimation Using Monocular and Stereo Cues., 01 2007, S. 2197–2203

[26] Srivastava, Nitish ; Hinton, Geoffrey ; Krizhevsky, Alex ; Sutskever, Ilya ; Salakhutdinov, Ruslan: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In: *Journal of Machine Learning Research* 15 (2014), Nr. 56, S. 1929–1958. – URL http://jmlr.org/papers/v15/srivastava14a.html

[27] Van Rossum, Guido ; Drake, Fred L.: *Python 3 Reference Manual*. Scotts Valley, CA : CreateSpace, 2009. – ISBN 1441412697

[28] Vannadil, Harikrishnan: *Stereopsis and Tests for Stereopsis*. https://eyewiki.aao.org/Stereopsis_and_Tests_for_Stereopsis. – [Online; zuletzt besucht am 30.04.2022]

[29] Verma, Rachna ; Verma, Arvind: PATCH BASED STEREO MATCHING USING CONVOLUTIONAL NEURAL NETWORK. In: *ICTACT Journal on Image and Video Processing* 11 (2021), 03, S. 0976–9102

[30] Welchman, Andrew E.: The Human Brain in Depth: How We See in 3D. In: *Annual Review of Vision Science* 2 (2016), Nr. 1, S. 345–376. – URL https://doi.org/10.1146/annurev-vision-111815-114605

[31] Yahya, Zakia ; Hassan, Muhammad ; Younis, Shahzad: Probabilistic Analysis of Targeted Attacks Using Transform-Domain Adversarial Examples. In: *IEEE Access* PP (2020), 02, S. 1–1

[32] Zbontar, Jure ; LeCun, Yann: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. In: *CoRR* abs/1510.05970 (2015). – URL http://arxiv.org/abs/1510.05970

[33] Zhu, Shiqiang ; Wang, Zhi ; Zhang, Xuequn ; Li, Yuehua: Edge-preserving guided filtering based cost aggregation for stereo matching. In: *Journal of Visual Communication and Image Representation* 39 (2016), S. 107–119. – URL https://www.sciencedirect.com/science/article/pii/S1047320316300797. – ISSN 1047-3203

[34] Žbontar, Jure ; LeCun, Yann: Computing the stereo matching cost with a convolutional neural network. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, S. 1592–1599

A Anhang

A.1 Hyperparameter des Semi Global Matching

Hyperparameter	Wert
sgm_P1	2,3
sgm_P2	55,9
sgm_Q1	4
sgm_Q2	8
sgm_V	1,5
sgm_D	0,08

Abbildung A.1: Hyperparameter des Semi Global Matching nach Zbontar und LeCun [32]

A.2 Verwendete Python Bibliotheken

- Keras 2.8.0
- Tensorflow 2.8.0
- NumPy 1.21.5
- PIL 7.2.0
- Matplotlib 3.2.2
- Scikit-image 0.16.2
- SciPy 1.8.0
- OpenCV 4.5.5

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Neuronale Tiefenbildrekonstruktion anhand von Stereobildpaaren

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original