Lizenz

Zitieren

CHRISTOPH DREXLER:

Beleuchtungsinvariante und rauschintensive Disparitätskartenberechnung.

Masterarbeit für den Studiengang „Master of Science in Praktischer Informatik“.

Fernuniversität in Hagen, Sept. 2012. Online: http://www.drexler.eu/masterarbeit

Corrigenda

S. 39: Die Abb. 6 verdeckt die Fußnoten 110 bis 112. Diese lauten:

110 Vgl. z. B. [Won2011].
111 Vgl. [Gales2010].
112 So lautet die Abkürzung in der Middlebury-Evaluationstabelle; im entsprechenden Artikel wird hingegen die Abkürzung LSTDP verwendet, vgl. [Deng2006].

S. 53: In der Formel zur Subpixel-Interpolation wurden versehentlich in der Druckversion das Minus- und das Pluszeichen vertauscht. Die korrekte (in der Referenzimplementierung auch so verwendete) Formel lautet:

$$d_{float}(p) = d_{WTA}(p) \; + \; \frac{S(p, d_{WTA} - 1) - S(p, d_{WTA} + 1)}{2(S(p, d_{WTA} - 1) - 2S(p, d_{WTA}) + S(p, d_{WTA} + 1))}$$

Außerdem fehlt an dieser Stelle der Verweis auf: [Ernst2008], S. 232.

Beleuchtungsinvariante und rauschintensive Disparitätskartenberechnung

Masterarbeit
für den Studiengang
„Master of Science in Praktischer Informatik“

Eingereicht

im September 2012

an der Fernuniversität in Hagen

Autor:

Dr. Christoph Drexler, Franz-Baumann-Weg 7, 6020 Innsbruck, Österreich

Betreuung:

Prof. Dr. Gabriele Peters

Dr. Klaus Häming

Lehrgebiet Mensch-Computer-Interaktion der Fernuniversität in Hagen

Inhaltsverzeichnis

1. Aufgabenstellung

Die Aufgabenstellung dieser Arbeit verdankt sich einem Vorschlag durch Frau Prof. Dr. Gabriele Peters und Herrn Dr. Klaus Häming am Lehrgebiet „Mensch-Computer-Interaktion“ der Fernuniversität Hagen und steht in Zusammenhang mit einem Forschungsschwerpunkt zur „Freihanderfassung von 3D-Objekten“, der von den beiden Genannten bereits seit mehreren Jahren verfolgt wird[1]. Während in anderen gängigen Ansätzen eine große Zahl von Fotos mit genau kalibrierten Kameraeinstellungen erstellt wird, um daraus die dreidimensionale Struktur eines Objektes zu erfassen[2], besteht die Besonderheit des von Peters und Häming verfolgten Ansatzes darin, dass nur wenige Fotos für diese Aufgabe herangezogen werden, die noch dazu Freihand, d. h. ohne Kalibrierung, aufgenommen werden. Es geht also darum, bei der 3D-Erfassung auf teures und u. U. kompliziert zu bedienendes Spezialequipment verzichten zu können und stattdessen mit einer einfachen handelsüblichen Digitalfotokamera und einem einfachen Notebook o. Ä. das Auslangen zu finden.

Die auf dem Hintergrund dieses Forschungsinteresses konzipierte Themenstellung für die vorliegende Arbeit wurde mir erstmals von Herrn Klaus Häming Mitte Dezember 2012 in einem längeren Telefonat vorgestellt und später aufgrund meiner Nachfragen in zahlreichen E-Mails präzisiert.

Als **Anforderung** wurde formuliert, dass ein eigener Stereoalgorithmus erstellt werden sollte, der (1) *schnell* und (2) *beleuchtungsunabhängig* ist und (3) auch mit *weniger texturierten* Bildern zurechtkommt. Was unter einem Stereoalgorithmus zu verstehen ist, wird in Kapitel 2 dieser Arbeit genauer erläutert.

Bezüglich des dafür nötigen **Datenmaterials** (Stereo-Bildpaare mit Ground-Truth-Daten[3]) sollte ich einerseits auf die Middlebury-Datensätze[4] zurückgreifen, andererseits sollte ich spe-

---

1 Vgl. http://www.inf.fh-dortmund.de/personen/professoren/peters/pages/research/Freehand/Freehand.html (Abrufdatum: 2012-08-28). An der Fernuniversität gibt es bislang noch keine eigene Homepage für das Lehrgebiet, weshalb hier auf die Beschreibung des Forschungsschwerpunkts auf der Homepage der Fachhochschule in Dortmund verwiesen wird.

2 Darüber hinaus gibt es bekanntermaßen noch Ansätze, die nicht oder nur teilweise auf fotografische Aufnahmen setzen, sondern andere Verfahren anwenden, wie z.B. die Erfassung einer speziell dafür auf das zu erfassende Objekt projizierten Infrarot-Punktwolke wie bei der Kinect.

3 Als „Ground Truth“ bezeichnet man in der einschlägigen Fachliteratur Bilddateien, in denen die korrekten Tiefeninformationen als sog. Disparitätswerte (dieser Begriff wird in Kapitel 2 genauer erläutert) gespeichert sind, die auf anderem Weg als durch einen Stereoalgorithmus (z. B. mit Hilfe eines 3D-Scanners) ermittelt wurden. Die durch den jeweiligen Stereoalgorithmus errechneten Disparitätskarten werden dann mit dieser „Ground Truth“ verglichen, um die Qualität des Algorithmus zu überprüfen und dessen Fehlerrate zu ermitteln.

4 Vgl. http://vision.middlebury.edu/stereo/data/.

ziell für das o. g. Anforderungsprofil erstellte Datensätze durch das Lehrgebiet zur Verfügung gestellt bekommen.

Als **Vorgangsweise** wurde mir empfohlen, mir zuerst einen Überblick über den State-of-the-Art zu verschaffen, dabei Algorithmen zu identifizieren, die bei der Umsetzung der Anforderung hilfreich sind, und mir dann einen eigenen Stereoalgorithmus zu überlegen, der nützliche Ideen kombiniert und eventuell eigene einbringt, diesen umzusetzen und mit Hilfe der Middlebury-Daten zu testen und einzuordnen. Als zusätzliche Anforderung wurde genannt, dass die Umsetzung in der Programmiersprache C/C++ erfolgen und jedenfalls auch unter Linux lauffähig sein sollte.[5]

Die genannten Anforderungsaspekte (schnell, beleuchtungsunabhängig, auch für wenig texturierte Bilder geeignet) geben zwar die ungefähre Stoßrichtung vor, bedürfen aber einer Präzisierung (die teilweise aufgrund meiner Rückfragen durch Herrn Häming erfolgte und sich zum anderen Teil aufgrund des mir zur Verfügung gestellten Datenmaterials ergab). Um die Besonderheiten der Aufgabenstellung deutlicher zu machen, werde ich im Folgenden auf die im Anforderungsprofil genannten Aspekte etwas genauer eingehen.

1.1. Schnelligkeit

Die bereits bestehenden Ansätze zu Stereoalgorithmen unterscheiden sich in Bezug auf ihre Rechengeschwindigkeit beträchtlich: Während einige Stereoalgorithmen unter Zuhilfenahme des Grafikprozessors mehrere Disparitätskarten pro Sekunde berechnen können[6], haben andere eine Laufzeit im zweistelligen Sekundenbereich[7], wiederum andere (sog. globale Algorithmen) benötigen für diese Aufgabe eine zweistellige Anzahl an Minuten.

Auf meine Nachfrage, wie Herr Häming „schnell“ genauer definieren würde, erhielt ich zur Antwort, dass – im Hinblick auf die am Lehrgebiet entwickelte Demo-Anwendung zur Freihanderfassung von 3D-Modellen – 30 Sekunden pro Bildpaar bei einer Auflösung von 800 x 600 Pixeln „schon hart an der Schmerzgrenze“ wären[8]. Da die mir vom Lehrgebiet zur Verfügung gestellten Datensätze (s. Kap. 2) nicht nur eine höhere Auflösung als die Middlebury-Datensätze aufweisen (die in der Middlebury-Evaluation einbezogenen und von vielen

5 Diese Formulierung der Aufgabenstellung entstammt weitgehend einer E-Mail, die ich von Herrn Klaus Häming am 14. Dezember 2011 erhalten habe.

6 Zu den schnellsten gehört der in der Middlebury-Evaluation als „CostFilter“ geführte Algorithmus, für den im dazugehörigen Artikel eine Laufzeit von 65 ms für die Teddy- und Cones-Bildpaare der Middlebury-Datensätze angegeben wird: [Hosni2011], S. 5; [Rhemann2011], S. 3022; die beiden Artikel beschreiben denselben Algorithmus.

7 Der in der Middlebury-Evaluation derzeit an erster Stelle gereihte „ADCensus“-Algorithmus benötigt in seiner GPU-Implementierung auch weniger als 100 ms, für die CPU-Implementierung werden allerdings ca. 15 s als Laufzeit für die Teddy- und Cones-Bildpaare angegeben.

8 Vgl. E-Mail vom 23. Jänner 2012.

Autoren getesteten Teddy- und Cones-Bildpaare haben 450 x 375 Pixel, die vom Lehrgebiet zur Verfügung gestellten hingegen 800 x 600 Pixel), sondern teilweise auch eine deutlich breitere Streuung der in den Bildern auftretenden Disparitätswerte („Teddy“ und „Cones“: Disparitätswerte von 0 bis 59; Datensätze des Lehrgebiets: Werte von -127,28 bis 124.22 beim Bildpaar „Clown Sokrates“), folgte daraus, dass der Aspekt der Schnelligkeit eine besondere Rolle bei der Bestimmung der in Frage vorkommenden Algorithmus-Bestandteile spielen musste.

Bei der Sichtung der einschlägigen Fachliteratur lag es daher nahe, jene Artikel bevorzugt zu berücksichtigen, die eine möglichst schnelle Berechnung – auch ohne Unterstützung durch den Grafikprozessor – in Aussicht stellen[9]. Wenn die Berechnung auch bei den höher aufgelösten Datensätzen mit größerer Disparitätsbandbreite in maximal 30 Sekunden erfolgen soll, muss die Laufzeit für die Teddy- und Cones-Bildpaare (die in vielen Artikeln angegeben wird) wohl ihm einstelligen Sekundenbereich liegen, um den gestellten Anforderungen an die Schnelligkeit zu genügen.

1.2. Beleuchtungsunabhängigkeit

Die Datensätze, die in der Middlebury-Evaluation[10] für den Qualitätsvergleich verschiedener Stereoalgorithmen verwendet werden, sind unter sehr stark kontrollierten Bedingungen erstellt worden. Dazu gehört auch, dass die Beleuchtungsbedingungen möglichst stabil gehalten wurden.

Bei der am Lehrgebiet Mensch-Computer-Interaktion der Fernuniversität in Hagen erforschten Freihanderfassung von 3D-Modellen werden Fotos verwendet, die mit einer handelsüblichen einfachen Digitalkamera ohne Kalibrierung erstellt wurden. Dabei ist es sehr leicht möglich, dass sich die Beleuchtungsbedingungen des ersten und zweiten Bildes unterscheiden, beispielsweise aufgrund des unterschiedlichen Lichteinfallswinkels, oder weil der/die Fotograf/in oder eine vorbeigehende Person einen Schatten wirft, oder auch, weil der automatische Blitz der Kamera einmal auslöst und beim zweiten Mal nicht.

Während die Beleuchtungsunabhängigkeit in der Middlebury-Evaluation also keine relevante Rolle spielt, ist sie für die Aufgabenstellung dieser Arbeit von großer Bedeutung. Das

9 Das heißt nicht, dass nur die Artikel zu besonders schnellen Algorithmen einbezogen worden wären, sondern lediglich, dass ich diese genauer studiert habe und mich eingehender mit dem Funktionsprinzip dieser Algorithmen beschäftigt habe. Bei Artikeln, die keine Angabe zur Laufzeit machen, bin ich davon ausgegangen, dass sie nicht den gestellten Anforderungen an die Schnelligkeit entsprechen, weil unwahrscheinlich ist, dass die Autorinnen und Autoren dieser Artikel es verabsäumt hätten, auf einen derartigen besonderen Vorzug ihres Algorithmus hinzuweisen.

10 Vgl. http://vision.middlebury.edu/stereo/eval/.

hat zur Folge, dass nicht von vornherein gesagt werden kann, ob die in der Middlebury-Evaluation an vorderster Stelle gereihten Algorithmen auch unter schlechteren Beleuchtungsbedingungen ähnlich gut reüssieren könnten. Bei der Sichtung der Fachartikel galt es daher, sich nicht so sehr an den Messzahlen der Middlebury-Evaluation zu orientieren, sondern speziell nach Verfahren Ausschau zu halten, die auch eine gute Performance unter schwierigeren Beleuchtungsbedingungen versprechen.

Glücklicherweise gibt es bereits einige Evaluationsstudien, die speziell auf die Beleuchtungsunabhängigkeit verschiedener Kostenfunktionen eingehen[11], sodass in diesem Punkt bereits auf fundierte Forschungen zurückgegriffen werden kann.

1.3. Eignung für wenig texturierte Bilder

Bei der Frage, inwieweit sich bestimmte Kostenfunktionen oder Algorithmen für das Stereo Matching von Bildern mit geringer Texturierung eignen, können leider keine einschlägigen Evaluationsstudien herangezogen werden – es finden sich lediglich dort und da in Artikeln einige verstreute Hinweise, wie sich der jeweilige Algorithmus in schlecht texturierten Bildteilen schlägt und wie die Performance in diesen Bildregionen vielleicht verbessert werden könnte.

In Bezug auf diese Anforderung blieb daher nicht viel übrig, als die wenigen vorhandenen Hinweise aufzugreifen und zu beobachten, wie sich der in dieser Arbeit vorgeschlagene Algorithmus in Bildregionen mit geringer Texturierung bewährt.

Eine noch genauere Beschreibung der Aufgabenstellung der vorliegenden Arbeit ist nicht möglich, ohne auf entsprechende Fachbegriffe zurückzugreifen. Daher sollen im Folgenden das Grundprinzip und die Vorgangsweise des maschinellen Stereo-Sehens vorgestellt werden (Kapitel 2). Im Anschluss daran wird das Datenmaterial vorgestellt, das bei der Erstellung eines eigenen den genannten Anforderungen möglichst gut entsprechenden Algorithmus verwendet werden sollte (Kap. 3). Dabei wird sich herausstellen, dass das zur Verfügung gestellte Datenmaterial weitere, in der obigen Aufzählung noch nicht genannte Anforderungen mit sich bringt. Im darauf folgenden Kapitel (Kap. 4) werden die wichtigsten Ergebnisse der Lektüre einschlägiger Fachartikel zusammengefasst, bevor in Kapitel 5 der Algorithmus vorgestellt wird, den ich zur Lösung der gestellten Aufgabe vorschlage. Das abschließende Kapitel wird zusammenfassen, welche Fragen im Rahmen der für diese Masterarbeit recht knapp bemessenen Zeit offen bleiben mussten und in welche Richtungen der hier vorgeschlagene Algorithmus weiter entwickelt werden könnte.

11 Vgl. [Hirschmüller2009]; [Hermann2011].

2. Stereo Vision und Stereo Matching

Ein bereits seit Längerem intensiv erforschtes Verfahren, um aus zwei aufgenommenen Bildern desselben Objekts Rückschlüsse auf die dreidimensionale Struktur dieses Objekts zu gewinnen, wird üblicherweise als *Stereo Vision*, *Stereo Matching* oder *Stereo Correspondance* bezeichnet[12]. Dabei orientiert man sich am grundlegenden Funktionsprinzip der menschlichen visuellen Raumwahrnehmung: Ähnlich wie wir aus der Differenz der vom rechten und vom linken Auge wahrgenommenen visuellen Sinneseindrücke darauf schließen, wie nahe oder weit entfernt sich die Gegenstände befinden, die sich im Sichtfeld unserer Augen befinden, versucht man beim maschinellen Stereo-Sehen aus der Differenz zweier oder mehrerer Fotografien die Tiefeninformationen zu den auf den Fotografien sichtbaren Objekten zu extrahieren. Man kann sich das Grundprinzip der Herangehensweise sehr leicht vor Augen führen (im wörtlichen Sinn!), wenn man ein Objekt, z. B. die eigene Hand oder einen Bleistift in ca. 10–20 cm Entfernung vor die eigenen Augen hält und dann abwechselnd das linke und rechte Auge schließt: Während das relativ nahe Objekt vom rechten Auge ganz links und vom linken Auge weit rechts gesehen wird, werden weiter entfernte Gegenstände von den Augen wesentlich ähnlicher wahrgenommen. Dass unser Gehirn diese Differenz zwischen den Sinneseindrücken des rechten und des linken Auges nutzt, um die Entfernung von Gegenständen abzuschätzen, ist seit Langem bekannt und erforscht.

Dasselbe Prinzip lässt sich auch für die Umrechnung fotografischer Aufnahmen in dreidimensionale Modelle nutzen. Richard Szeliski definiert das Verfahren folgendermaßen:

> Stereo Matching is the process of taking two or more images and estimating a 3D model of the scene by finding matching pixels in the images and converting their 2D positions into 3D depths.[13]

12 Die drei genannten Begriffe werden in der einschlägigen Literatur weitgehend synonym gebraucht. Richard Szeliski überschreibt beispielsweise das entsprechende Kapitel in seinem erschienenen Standardwerk „Computer Vision“ mit dem Begriff „Stereo Correspondance“, verwendet aber innerhalb des Kapitels hauptsächlich den Begriff „Stereo Matching“. Wenn man es genau nehmen will, kann man zwischen den beiden Begriffen *Stereo Vision* und *Stereo Matching* noch weiter differenzieren, indem man unter Stereo Matching lediglich jenen Teilschritt versteht, bei dem die Differenz der beiden Bilder in Form von sog. Disparitätskarten (siehe dazu die Erläuterung an späterer Stelle dieser Arbeit) berechnet wird, während man unter Stereo Vision den gesamten Vorgang fasst, zu dem man außer dem eigentlichen Stereo Matching noch die u. U. vor der Berechnung der Disparitätskarten nötige Rektifizierung und weiters die Umrechnung der Disparitätskarten in Tiefeninformationen und eventuell noch die Texturierung des berechneten 3D-Modells zählen kann.

13 [Szeliski2011], S. 469.

Zumeist wird der Gesamtvorgang dabei in die folgenden Teilschritte zerlegt[14]: Zuerst werden durch eine sog. *Rektifizierung* (1) möglichst gute Voraussetzungen für den zweiten Schritt, das *Stereo Matching* im engeren Sinn (2) geschaffen, bei dem Pixel aus dem linken und rechten Bild einander zugeordnet werden. Als Ergebnis dieser Zuordnung werden sog. Disparitätskarten erstellt, die die Differenz der beiden Bilder kodieren. Aus diesen Differenzinformationen können im nächsten Schritt auf relativ einfache Weise *Tiefeninformationen* (3) gewonnen werden, die zur *Konstruktion des dreidimensionalen Modells* (4) der abgebildeten Szene herangezogen werden. Durch die *Texturierung* (5) werden diesem 3D-Modell Farbinformationen hinzugefügt, die dazu beitragen, einen realistischeren Eindruck der Bildszene zu erzeugen.[15]

Da es andernorts sehr gute ausführliche Darstellungen dieser Vorgangsweise gibt[16], soll an dieser Stelle auf genauere Erläuterungen zu den einzelnen genannten Schritten weitgehend verzichtet werden. Lediglich die beiden ersten Teilschritte werden im Folgenden etwas genauer beleuchtet: das *Stereo Matching*, weil sich die vorliegende Arbeit genau damit beschäftigen wird, und die *Rektifizierung*, weil sie nicht nur das Stereo Matching enorm erleichtert, sondern darüber hinaus einige Nebenprodukte (die sog. *Features*) erzeugt, die im Algorithmus, der in dieser Arbeit vorgestellt werden soll, eine ausgesprochen wichtige Rolle spielen.

2.1. Rektifizierung

Obwohl unser Gehirn die Aufgabe, Bildeindrücke des rechten Auges mit denen des linken Auges in Verbindung zu bringen, in jedem Augenblick scheinbar mühelos erledigt[17], zeigen die inzwischen bereits mehrere Jahrzehnte andauernden Bemühungen, diesen Vorgang durch möglichst effizienten Algorithmen in Soft- und Hardware nachzubilden, wie schwierig dieses Unterfangen eigentlich ist. Selbst ein für aktuelle Digitalkameratechnik relativ gering aufgelöstes Foto mit 800x600 Pixeln erlaubt theoretisch bereits eine unvorstellbar hohe Zahl an gegenseitigen Zuordnungen. Dazu kommt noch, dass es in fast jedem Bildpaar Bildbereiche gibt, die im jeweils anderen Bild gar nicht zu sehen sind, weil sie dort durch näher gelegene Gegenstände verdeckt werden (so genannte Verdeckungen oder „Occlusions“), d. h.

14 Die einzelnen Teilschritte werden hier nur kurz überblicksartig vorgestellt und später – soweit für den Gesamtzusammenhang der vorliegenden Arbeit nötig – näher erläutert.

15 Ich verzichte hier auf eine Diskussion von Detailfragen, die für den Zusammenhang dieser Arbeit nicht relevant sind, z. B. die genaue Abgrenzung der genannten Teilschritte.

16 Vgl. z. B. [Szeliski2011], S. 467–503.

17 Die Schwierigkeit dieser Aufgabe wird den meisten von uns nur in manchen Ausnahmesituationen bewusst, wenn wir – z. B. unter Alkoholeinfluss oder aufgrund von Kreislaufproblemen – es plötzlich einmal doch nicht schaffen, die beiden Bilder in Einklang zu bringen und dann „doppelt sehen“.

dass manche Bildpunkte nur in einem der beiden Bilder sichtbar sind und gar nicht zu einem Bildpunkt im anderen Bild in Beziehung gesetzt werden *können*. Jegliche Möglichkeit, die unvorstellbar große Anzahl an Möglichkeiten einzuschränken, kann daher nur willkommen sein.

Für eine sehr grundlegende Einschränkungsmöglichkeit kann man wiederum bei den menschlichen Sehgewohnheiten Anleihe nehmen: Anders als manche Tiere, die ihre Augen unabhängig voneinander bewegen (können), richten wir unseren Blick normalerweise fast parallel aus[18]. Das bringt für das räumliche Sehen manche Vorteile: (1) Die Bilder des linken und rechten Auges unterscheiden sich zwar, zeigen aber im Großen und Ganzen doch einen recht ähnlichen Ausschnitt der uns umgebenden Wirklichkeit, sodass es überhaupt möglich ist, diese beiden Bilder in Verbindung zu bringen und aus den Differenzinformationen auf die Entfernungen zu schließen. (2) Das Zusammendenken der beiden Bilder (das „Matching“) wird durch deren relativ große Ähnlichkeit deutlich erleichtert. (3) Während wir uns zwar nicht unbedingt darauf verlassen können, dass ein Gegenstand, der sich im einen Bild rechts befindet, auch im anderen Bild rechts zu sehen ist (was z. B. bei nahe vor die Nase gehaltenen Objekten nicht der Fall ist), ist ein bestimmtes Objektmerkmal (beispielsweise die Spitze eines Berggipfels) immer in beiden Bildern auf derselben Höhe zu finden.

Wenn man sich diese Besonderheiten für das maschinelle Stereo-Sehen zunutze machen könnte, wäre schon ein sehr großer Schritt in Richtung einer Vereinfachung der überaus komplexen Aufgabe geschafft. Viele Informatiker verwenden daher für ihre Suche nach effizienten Algorithmen für das Stereo Matching Fotografien, die unter streng kontrollierten Bedingungen von in relativ geringem Abstand exakt parallel montierten Kameras bzw. von derselben parallel verschobenen und genau kalibrierten Kamera aufgenommen wurden[19]. Die von sehr vielen Forscherinnen und Forschern verwendeten „Middlebury Stereo Datasets“[20] sind renommierte Beispiele dieser Kategorie.

Allerdings ist es selbst bei so stark kontrollierten Bedingungen kaum möglich, die Kamera(s) so genau zu kalibrieren, dass die Bildpixel in der zweiten Aufnahme exakt auf derselben Höhe abgebildet werden wie in der ersten Fotografie. Insbesondere bei etwas höheren Auflösungen (wie sie bei heutigen Digitalkameras allgemein üblich sind), ist ein leichtes „Verrutschen“ nicht der Ausnahme-, sondern der Normalfall.

---

18 Ich verzichte hier auf die Diskussion von Sonderfällen, z. B. das Schielen, sei es nun krankhaft oder bewusst herbeigeführt.

19 Man spricht in diesem Zusammenhang von einer Off-Axis-Konstellation. Ob die Fotos von zwei identischen Kameras simultan oder von derselben Kamera hintereinander aufgenommen wurden, ist für unseren Zusammenhang nicht weiter von Belang.

20 Vgl. http://vision.middlebury.edu/stereo/data/ (Abrufdatum: 2012-08-29).

Sogar bei solchen unter Laborbedingungen aufgenommenen Fotos ist daher ein entsprechender Korrekturschritt – die so genannten *Rektifizierung* – notwendig. Dabei werden bestimmte eindeutige Bildmerkmale (besonders eignen sich dafür beispielsweise Ecken) in beiden Bildern identifiziert. Aufgrund dieser eindeutig zugeordneten Bildpunktpaare werden aus den ursprünglich aufgenommenen Bildern mithilfe geometrischer Verfahren[21] zwei neue berechnet, die so aufgebaut sind, dass die korrespondierenden Bildpixel in den beiden Bildern – sofern es zu einem Bildpixel überhaupt ein korrespondierendes Pixel im zweiten Bild gibt, was bei Verdeckungen nicht der Fall ist – auf jeden Fall auf derselben Bildzeile liegen. Das Beispiel in Abb. 1 verdeutlicht[22], dass rektifizierte Bilder, die aus Bildern berechnet wurden, die von einer kalibrierten Kamera in paralleler Blickrichtung aufgenommen wurden, sich kaum von den ursprünglichen Fotos unterscheiden, während Abb. 2 zeigt[23], dass die auf diese Weise errechneten Bilder u. U. stark verzerrt aussehen können, wenn sie nicht aus parallel

Abbildung 1: Linkes und rechtes (bereits rektifiziertes) Bild des sog. Teddy-Bildpaars aus den Middlebury Datasets

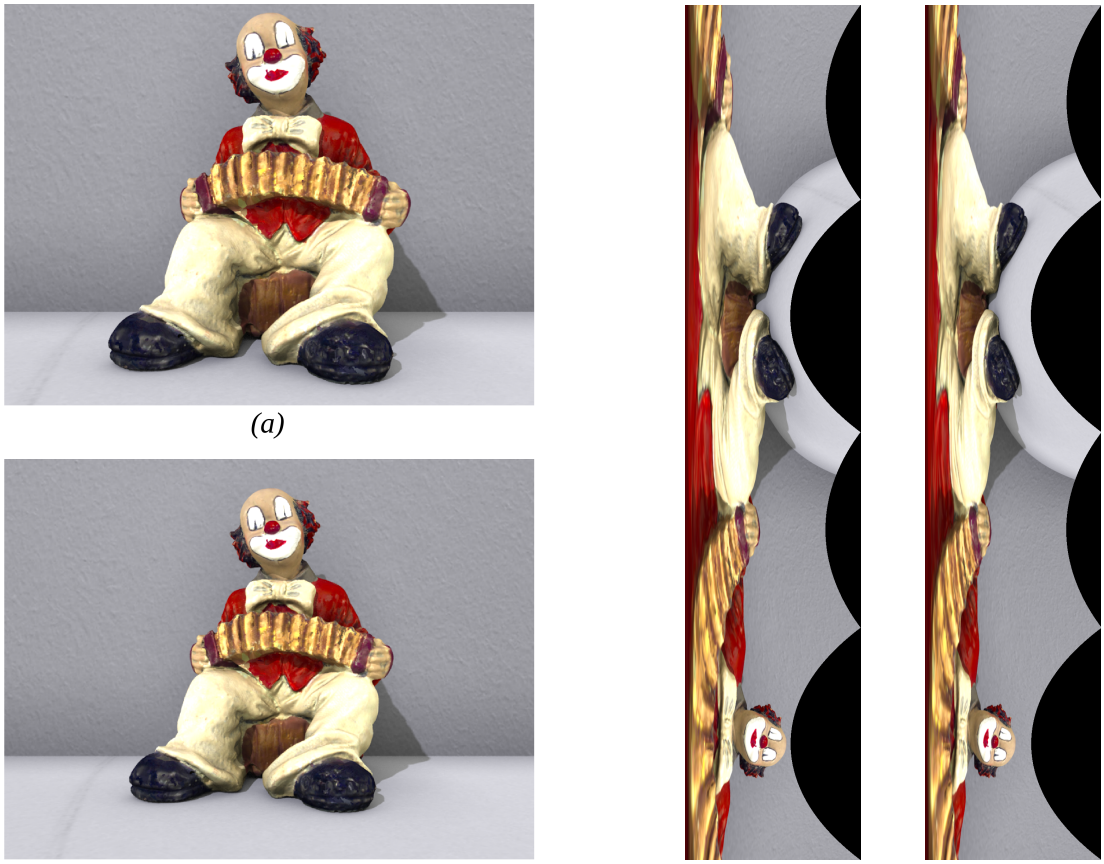
versetzten Kamerapositionen erstellt wurden.

Auch wenn die rektifizierten Bilder also zuweilen stark verzerrt sind, bringen sie den unschlagbaren Vorteil mit sich, dass die Suche nach den korrespondierenden Pixeln nun nicht mehr zweidimensional im ganzen Bild stattfinden muss, sondern eindimensional auf die einzelnen Bildpunktzeilen beschränkt werden kann – auf diese Weise bleibt nur mehr ein Bruchteil an Möglichkeiten für das Stereo Matching übrig.

21 Es ist an dieser Stelle für das weitere Verständnis der Arbeit m. E. nicht nötig, die Prinzipien der Epipolargeometrie genauer darzustellen.

22 Das sog. Teddy-Bildpaar entstammt den sog. Middlebury-Datasets aus dem Jahr 2003, die von Daniel Scharstein, Alexander Vandenberg-Rodes, and Richard Szeliski für Forschungszwecke erstellt und auf der Seite http://vision.middlebury.edu/stereo/data/scenes2003/ zur Verfügung gestellt wurden. Nähere Hinweise zur Technik, mit der diese Bilder erstellt wurden, finden sich in: [Scharstein2003].

23 Das Beispiel stammt aus den Datensätzen, die von Klaus Häming speziell erstellt wurden und die an späterer Stelle in dieser Arbeit noch genauer vorgestellt werden.



(a) *(b)* *(c)* *(d)*

Abbildung 2: Nicht rektifiziertes (a) erstes und (b) zweites sowie rektifiziertes (c) erstes und (d) zweites Bild des sog. Clown-in-Front-Bilderpaars; die starke Verzerrung resultiert in diesem Fall daraus, dass das zweite Bild nicht parallel neben, sondern aus einer Kameraposition vor jener des ersten Bilds aufgenommen wurde.

Da die Aufgabenstellung dieser Arbeit keine Rektifizierung beinhaltet, sondern diese lediglich voraussetzt, ist es an dieser Stelle nicht nötig, auf die mathematischen Details der Rektifizierung einzugehen. Für das Verständnis des in dieser Arbeit vorgestellten Algorithmus ist jedoch essentiell, dass für die Rektifizierung üblicherweise besonders auffällige Bildmerkmale bestimmt werden, bei denen eine gegenseitige Zuordnung in den beiden Bildern mit hoher Wahrscheinlichkeit möglich ist. Je nach verwendetem Rektifizierungsalgorithmus sind zumindest sieben oder acht Featurepaare nötig. Aufgrund der relativ geringen benötigten Anzahl an Features ist es möglich, diese „von Hand“ zu bestimmen, es gibt aber auch Verfahren, diese automatisch zu ermitteln – beispielsweise enthält die freie Programmbibliothek OpenCV eine Reihe entsprechender Algorithmen für das „Feature Matching“[24].

Da es bei der am Lehrgebiet Mensch-Computer-Interaktion der Fernuniversität in Hagen erforschten Freihanderfassung von 3D-Modellen gerade darum geht, den Vorgang möglichst zu vereinfachen, werden in der am Lehrgebiet entwickelten Demo-Anwendung die für die

24 Vgl. http://docs.opencv.org/modules/features2d/doc/features2d.html (Abrufdatum: 2012-08-30).

Rektifizierung nötigen Features nicht „von Hand“, sondern automatisch bestimmt. Bei diesem Vorgang werden jedoch nicht nur die unbedingt nötigen sieben Featurepaare ermittelt, sondern sehr viel mehr, es handelt sich dabei üblicherweise um eine Anzahl an Bildpunktpaaren im dreistelligen Bereich – allerdings variiert diese Zahl stark je nach Art der aufgenommenen Szene.

Im Zusammenhang mit der Rektifizierung werden aus dieser relativ großen Zahl die für die Rektifizierung geeignetsten ausgewählt, die übrigen werden verworfen. Im Stereo-Matching-Algorithmus, der in dieser Arbeit vorgeschlagen wird, werden diese „Abfallprodukte“ jedoch wiederverwendet. Da die Disparität (also die zwischen dem linken und rechten Bild bestehende horizontale Differenz) für diese Bildpunktpaare bekannt ist bzw. leicht bestimmt werden kann[25], können sie als Ausgangspunkte für die Bestimmung des Disparitätssuchbereichs benutzt werden[26].

Zur Erinnerung: Sinn der Rektifizierung ist es, die ansonsten nötige 2D-Suche auf eine wesentlich weniger komplexe eindimensionale Suche einzuschränken. Damit sind wir am Kern des ganzen Verfahrens angelangt: dem „Stereo Matching“ im engeren Sinn.

2.2. Stereo Matching

Beim eigentlichen Stereo Matching (im weiteren Sinn wird der Begriff ja auch für das gesamte Verfahren inklusive der vorbereitenden und nachfolgenden Teilschritte verwendet) geht es darum, Bildpunkte des einen Bildes und Bildpunkte des zweiten Bildes einander zuzuordnen. Die dabei übliche Vorgangsweise soll im Folgenden beschrieben werden. Um das Grundprinzip zu verdeutlichen, wird dabei vorerst von einer Off-Axis-Konstellation ausgegangen, also davon, dass die Kamera für die zweite Aufnahme seitlich in rechtem Winkel zur Blickrichtung der Kamera verschoben wurde,

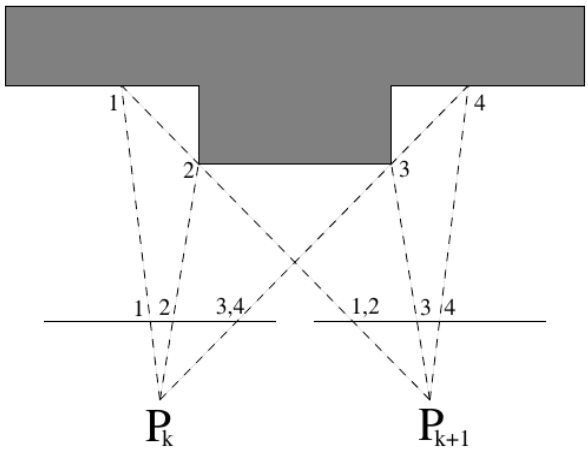


Abbildung 3: Sichtbarkeit von Pixeln bei parallelen Kameraachsen

25 Leider bringt die automatische Bestimmung der Features nicht nur korrekte, sondern auch falsch zugeordnete Bildpunktpaare hervor. Um die Aufgabenstellung dieser Arbeit nicht über Gebühr zu verkomplizieren, werden in dieser Arbeit jedoch – auf Vorschlag von Herrn Häming, mit dem ich im Erstellungsprozess dieser Arbeit in regem Austausch stand – ausschließlich überprüfte und verifizierte Bildpunktpaare herangezogen.

26 Das Verfahren wird später in dieser Arbeit noch genauer beschrieben.

und zwar so, dass die beiden Aufnahmen mit paralleler Ausrichtung der Kameraachse erstellt wurden (vgl. Abb. 3[27]).

Bei einer solchen Konfiguration können folgende Möglichkeiten unterschieden werden: (1) Das zu einem Bildpunkt des linken Fotos gehörige Pixel des rechten Bildes befindet sich an derselben Position[28] – was de facto jedoch nur bei geringem Kameraabstand und sehr weit entfernten Gegenständen möglich ist. (2) Das rechte Pixel befindet sich – je nach Entfernung des Gegenstands – unterschiedlich weit links vom zugehörigen linken Bildpunkt (wie dies bei den Punkten 2 und 3 in Abb. 3 der Fall ist). (3) Ein im linken Bild sichtbarer Punkt ist im rechten verdeckt (was in Abb. 3 auf alle Bildpunkte zwischen den Punkten 1 und 2 zutrifft).

Abbildung 4: Linkes Basisbild und zugehörige Disparitätskarte des Teddy-Bildpaars aus den Middlebury Stereo Datasets

Die Differenz zwischen den zugeordneten Bildpunkten des linken und rechten Bildes wird als „Disparität“ bezeichnet. Wenn die Disparitäten aller Bildpunkte bestimmt wurden (üblicherweise werden auf jeden Fall die Disparitäten für das linke Bild berechnet, in manchen Algorithmen auch jene für das rechte Bild), kann eine sog. „Disparitätskarte“ erstellt werden, bei dem an Stelle jedes Bildpunkts des Basisbilds (meist des linken Fotos) dessen Disparität zum Referenzbild (meist zum rechten Bild)[29] eingetragen wird. Nach einer entsprechenden

27 Die Grafik entstammt: [Heyden2000], S. 93.

28 Theoretisch gesehen ist eine exakte Übereinstimmung der Pixelposition eigentlich nur bei unendlich weit entfernten Objekten möglich, da der Pixelabstand (man verwendet dafür auch den Terminus technicus „Disparität“) jedoch in vielen Stereoalgorithmen ganzzahlig bestimmt wird, ist auch bei weit entfernten Objekten (wenn die exakte Disparität kleiner als eine halbe Pixelbreite wäre) ein Nullabstand möglich.

29 Die in einigen Artikeln auffindbare Verwendung der Terminologie „Basisbild/Referenzbild“ ist exakter als die Terminologie „linkes/rechtes Bild“, weil sich die meisten Algorithmen auf die Berechnung *einer* Disparitätskarte (für das Basisbild) konzentrieren. Sofern überhaupt eine zweite Disparitätskarte (für das Referenzbild) berechnet wird (was oft, aber nicht immer der Fall ist), dient dies nur der Überprüfung mittels des sog. „Left-Right-Consistency Check“. Die Disparitätsverfeinerung wird dann meist nur für die Disparitätskarte des Basisbilds durchgeführt. Was in den Algorithmen als Basisbild verwendet wird, hängt davon ab, für welches der beiden Bilder eine „Ground Truth“ zur Kontrolle vorliegt. Bei den Middlebury-Datensätzen dient das linke Bild als Basisbild und das rechte als Referenzbild, bei den vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellten Datensätzen ist es hingegen genau umgekehrt.

Skalierung der dadurch erhaltenen Werte können auf diese Weise Graustufenbilder erzeugt werden, die Auskunft über die relative Entfernung der abgebildeten Gegenstände geben (s. Abb. 4[30]).

Da die Disparitäten in der linken Disparitätskarte bei der Off-Axis-Konstellation lediglich kleiner oder gleich Null sein können und die Disparitäten in der rechten Disparitätskarte immer größer oder gleich Null sind, sodass in keiner Disparitätskarte positive und negative Werte gleichzeitig vorkommen, hat es sich eingebürgert, in den Disparitätskarten die Absolutwerte der jeweiligen Disparitäten zu verwenden:

> The correspondence between a pixel *(x, y)* in reference image *r* and a pixel *(x', y')* in matching image *m* is then given by
> $x' = x + sd(x, y), y' = y,$ (1)
> where $s = \pm 1$ is a sign chosen so that disparities are always positive.[31]

Das ist auch der Grund, warum die weiter vorne liegenden Objekte in Abb. 4 heller erscheinen. Würden nicht die Absolutwerte verwendet, sondern die eigentlichen Disparitäten, dann würden die vorne liegenden Objekte in den rechten Disparitätskarten ebenfalls hell erscheinen, in den linken Disparitätskarten jedoch dunkel (und die weiter hinten liegenden Objekte heller).

Zur Bestimmung der Disparitäten (der Kernaufgabe des Stereo Matching) sind in den letzten drei Jahrzehnten eine schier unüberschaubare Anzahl verschiedenster Algorithmen entwickelt worden. Manche davon benötigen (meist unter Zuhilfenahme der größeren Rechenleistung des Grafikprozessors) nur Sekundenbruchteile, andere mehrere oder sogar viele Minuten.

2.2.1. Funktionsprinzip lokaler Algorithmen

Die schnellsten Algorithmen sind meistens *lokale* Algorithmen, die zur Bestimmung der Disparität eine begrenzte Bildregion (die „Fenster“ genannt wird) rund um das jeweilige Pixel heranziehen. Dabei werden in einem ersten Schritt meist die *Kosten* für ein bestimmtes Matching berechnet[32]. Eine sehr einfache Kostenberechnung besteht beispielsweise darin, die

30 Das Basisbild kann auf der Seite http://vision.middlebury.edu/stereo/data/scenes2003/ bezogen werden. Die Disparitätskarte entstammt dem Artikel: [Mei2011], S. 8. Auch auf dieser – im Vergleich zu anderen – sehr guten Disparitätskarte sind einige Fehler zu erkennen, wie z. B. links vom Teddybär und am unteren Bildrand.

31 [Scharstein2002], S. 9 (bzw. S. 3 in der Online-Version). Die meisten jüngeren Arbeiten zum Stereo Matching folgen dieser von Scharstein und Szeliski vorgeschlagenen Konvention.

32 Die Darstellung der einzelnen Schritte lokaler und globaler Stereoalgorithmen orientiert sich an der Beschreibung in: [Scharstein2002]. Dabei handelt es sich nur um eine relativ grobe schematische Darstellung; in den konkreten Algorithmen können diese Schritte weitere Teilschritte aufweisen oder auch ineinander übergehen.

absolute Differenz der beiden Farbintensitätswerte als Kosten zu verwenden[33]: Haben beide Pixel dieselben oder nur sehr wenig verschiedene Intensitätswerte, dann wäre ein Matching dieser Pixel entsprechend „billig“, unterscheiden sich die Intensitätswerte hingegen stark, dann wäre eine entsprechende Zuordnung „teuer“.

Die jeweiligen Kosten werden in einem so genannten „Disparity Space Image“[34] abgelegt: In einer dreidimensionalen Matrix wird an der Stelle (x, y, d) der Kostenwert für das Matching des im Basisbild (meist dem linken Bild) liegenden Pixels *(x, y)* mit dem im Referenzbild (meist dem rechten Bild) liegenden Pixel *(x', y')* abgelegt, wobei $y' = y$ (die Bilder sind ja bereits rektifiziert) und $x' = x - d$ oder $x' = x + d$, je nachdem, ob es sich um die linke oder die rechte Disparitätskarte handelt.

In einem zweiten Schritt nutzt man die Tatsache aus, dass nahe beieinander liegende Pixel oft auch ähnliche Disparitäten aufweisen. Daher werden bei der *Kostenaggregation* die Matching-Kosten innerhalb der das Pixel umgebenden Bildregion aufaddiert. Am einfachsten geschieht das dadurch, dass die Kosten innerhalb eines Fensters fixer Größe (beispielsweise 9 x 9 Pixel) zusammenzählt. Allerdings versagt diese sehr einfache Form der Kostenaggregation bei bestimmten Konstellationen, insbesondere an den so genannten „Diskontinuitäten“, d.h. an jenen Stellen, an denen sich die Disparitäten nicht kontinuierlich, sondern plötzlich verändern (wie das an den Rändern der Vordergrundobjekte der Fall ist). In diesen Fällen können die Resultate deutlich verbessert werden, wenn die Größe des Fensters auf „intelligentere“ Weise bestimmt wird (was leider zumeist auch einen deutlich höheren Berechnungsaufwand nach sich zieht). Beispielsweise werden bei der „Adaptive Support-Weight Aggregation“ die Kosten jener Pixel, die näher liegen und farblich ähnlicher sind, stärker berücksichtigt als die Kosten jener Pixel, die weiter entfernt liegen und größere farbliche Unterschiede aufweisen[35]. Das Ergebnis der Kostenaggregation wird entweder in jenem „Disparity Space Image“ abgelegt, in dem zuvor die noch nicht aggregierten Kosten gespeichert waren, oder in einem neu angelegten zweiten „Disparity Space Image“, falls die ursprünglichen Kostenwerte im weiteren Verlauf des Algorithmus noch benötigt werden.

Die eigentliche *Disparitätsberechnung* im dritten Schritt erfolgt bei den lokalen Algorithmen meist nach einem sehr einfachen Prinzip: „Winner takes all“ (WTA), d. h. jenes

33 Bei farbigen RGB-Bildern gibt es unterschiedliche Möglichkeiten, die Intensitätsdifferenz zu bestimmen: Am ehesten wird man wohl entweder die Differenz der aufaddierten roten, grünen und blauen Intensitätswerte oder aber das Maximum der für jede Farbe extra bestimmten Intensitätsdifferenzen verwenden.

34 Mir ist keine Übersetzung dieses Terminus technicus bekannt und ich kann mir auch nicht so recht vorstellen, wie man den Begriff übersetzen könnte, ohne dass das Ergebnis lächerlich wirkt. Ich verwende daher im Folgenden nicht irgendeine unschöne Eindeutschung wie z. B. „Disparitätsraumbild“, sondern den in der Fachliteratur gebräuchlichen englischen Terminus.

35 Vgl. [Yoon2005]; vgl. auch [Hosni2009].

Matching, das nach der Kostenaggregation die geringsten aufsummierten Kosten benötigt, wird als das „korrekte“ Matching angenommen.

In einem abschließenden vierten Schritt werden in vielen Algorithmen die Ergebnisse dieser Disparitätsbestimmung bereinigt und verfeinert: So können beispielsweise verdeckte Bildbereiche aufgespürt und die Disparitätswerte an diesen Stellen durch speziell darauf abgestimmte Verfahren geschätzt werden; oder die ganzzahligen Disparitätswerte werden durch Interpolation auf Subpixel-Genauigkeit umgerechnet; auch die Beseitigung von „Ausreißern“ beispielsweise durch einen Median-Filter kann zur Verbesserung der Disparitätskarte beitragen.

Der größte Vorteil lokaler Algorithmen liegt vor allem in ihrer Schnelligkeit: Da sie zur Disparitätsberechnung nur einen begrenzten Bildbereich heranziehen, hält sich ihre Komplexität (und dem entsprechend auch ihre Rechenzeit) in Grenzen: Lokale Algorithmen benötigen oft nur einige Sekunden, um eine Disparitätskarte zu berechnen, einige sind sogar in der Lage (meist unter Zuhilfenahme des Grafikprozessors), mehrere Disparitätskarten pro Sekunde zu schaffen[36].

2.2.2. Globale Algorithmen

Die sog. *globalen* Stereoalgorithmen stehen in gewissem Sinn am anderen Ende der Skala: Während die lokalen Algorithmen einen eng begrenzten Bereich zur Berechnung heranziehen, beziehen die globalen Verfahren das ganze Bild in die Berechnung jedes einzelnen Pixels ein.

Dazu wird meist eine globale Energiefunktion definiert und dann eine Disparitätsfunktion $d(x,y)$ gesucht, die die globale Energie minimiert:

$$E(d) = E_{data}(d) + \lambda E_{smooth}(d)$$

Der Term $E_{data}(d)$ bemisst, wie gut die gesuchte Disparitätsfunktion zum jeweiligen Bildpaar passt. Wenn Kostenfunktion[37] als $C(x,y,d)$ formuliert wird, dann kann der entsprechende Term der Energiefunktion folgendermaßen dargestellt werden:

36 Selbstverständlich ist die Zeitdauer sehr stark von Randfaktoren abhängig, wie z. B. von der Größe der verwendeten Bilddateien und von der Rechenleistung der verwendeten Prozessoren. Die in manchen Fachartikeln angegebenen Zeiten beziehen sich meist auf die Berechnung eines linken und eines rechten Disparitätsbilds der Middlebury Stereo Datasets (450 x 375 Pixel) auf einem handelsüblichen PC der gehobenen Mittelklasse.

37 Dabei ist es für das Funktionsprinzip globaler Algorithmen irrelevant, ob es sich um die ursprünglichen Kosten oder um bereits aggregierte Kosten handelt. Manche globale Algorithmen aggregieren die Kosten vor der globalen Disparitätsberechnung, andere verzichten darauf. Insgesamt kann man jedenfalls festhalten, dass der für lokale Algorithmen zentrale Schritt der Kostenaggregation in den globalen Algorithmen eine untergeordnete Rolle spielt, während im Kontrast dazu die in lokalen Algorithmen meist durch eine simple WTA-Strategie gelöste Disparitätsberechnung in den globalen Algorithmen höchst aufwändig gestaltet wird.

$$E_{data}(d)=\sum_{(x,y)} C(x,y,d(x,y))$$

Der zweite Teil der Energiefunktion kodiert die Annahmen, die vom Algorithmus bzgl. der „Glattheit“[38] der untersuchten Szene macht. Dies kann beispielsweise dadurch gelöst werden, dass die Differenz der Disparitäten benachbarter Pixel gemessen werden, sodass größere Disparitätssprünge zu einer stärkeren Erhöhung der globalen Energie führen:

$$E_{smooth}(d)=\sum_{(x,y)} \rho(d(x,y)-d(x+1,y))+\rho(d(x,y)-d(x,y+1))$$

Die Funktion ρ wäre in diesem Fall irgend eine mit der Disparitätsdifferenz monoton steigende Funktion. In diese Glattheitsfunktion können auch die Farbwerte der beteiligten Pixel einbezogen werden, sodass Disparitätssprünge weniger stark „bestraft“ werden, wenn sich zugleich die Farbwerte stark ändern (was oft der Fall ist, wenn der Hintergrund anders gefärbt ist als ein Objekt im Vordergrund).

Der zentrale Teil der Disparitätsberechnung besteht nun darin, eine Disparitätsfunktion $d(x,y)$ zu finden, die die globale Energie in der Funktion $E(d)$ minimiert. Dazu wurden eine ganze Reihe von Algorithmen entwickelt und erprobt, die alle mehr oder weniger gemeinsam haben, dass sie vergleichsweise aufwändig sind, was wenig verwundert, wenn man bedenkt, dass jede Änderung eines Disparitätswerts an einem beliebigen Pixel des Bildes das Gesamtergebnis beeinflusst.

Selbstverständlich können auch bei globalen Algorithmen auf die eigentliche Disparitätsberechnung noch weitere Schritte folgen, die der Bereinigung und Verfeinerung der Ergebnisse dienen.

Aufgrund der höheren Komplexität benötigen globale Algorithmen meist deutlich mehr Rechenzeit als lokale Algorithmen, d. h. auch bei den relativ gering aufgelösten Bildern der „Middlebury Stereo Datasets“ sind meist mehrere Minuten für die Berechnung eines Bildpaares nötig. Während dieser Mehraufwand bis vor einigen Jahren noch mit einem deutlich besseren Ergebnis belohnt wurde (die Tabelle der „Middlebury Stereo Evaluation“[39], in der eine Vielzahl an Stereoalgorithmen nach der Qualität ihrer Ergebnisse gereiht werden, wurde jahrlang von den globalen Algorithmen angeführt), haben die lokalen Algorithmen inzwischen die globalen nicht nur eingeholt, sondern teilweise sogar überholt: In der „Middlebury Stereo

38 Leider klingt das deutsche Wort „Glattheit“ sehr viel weniger elegant als das englische „smoothness“. Die gemeinte Sache wird jedoch auch im deutschen Begriff Glattheit einigermaßen passabel eingefangen: Es geht ja darum, dass sich die Disparitäten in einer aufgenommenen Szene nur selten plötzlich stark ändern, während sie sehr oft gleich bleiben oder sich nur geringfügig ändern, dass also die meisten Szenen mehr „glatte“ Flächen als grobe Brüche aufweisen.

39 Vgl. http://vision.middlebury.edu/stereo/eval/.

Evaluation“ finden sich inzwischen schon eine ganze Reihe lokaler Algorithmen in den vorderen Rängen.

2.2.3. Dynamische Programmierung, Scanline Optimization

Eine weitere Klasse von Algorithmen kann in gewisser Weise zwischen den lokalen und globalen Algorithmen angesiedelt werden: Bei Verfahren, die mit den Begriffen „Dynamische Programmierung“ und „Scanline Optimization“ zusammengefasst werden, findet die Optimierung der Energiefunktion nicht global über das gesamte Bild hinweg statt, sondern nur über einen Teil des Bildes – oft wird beispielsweise das globale Energieminimum für eine einzelne Bildpunktzeile gesucht. Das an späterer Stelle näher vorgestellte „semi-globale Matching“ kann als eine spezielle Ausprägung dieser Strategie verstanden werden, daher ist eine ausführlichere Darstellung dieses Verfahrens an dieser Stelle nicht nötig.

Darüber hinaus gibt es selbstverständlich auch noch Ansätze, die nicht ohne Weiteres in die hier vorgestellte (und von Daniel Scharstein und Richard Szeliski entlehnte) Systematik eingepasst werden können, beispielsweise die (bei Scharstein/Szeliski auch kurz angeschnittenen) kooperativen Algorithmen[40] oder jene Verfahren, die bei einer Segmentierung des Bildes ansetzen. Für eine grundsätzliche Orientierung bzgl. der in dieser Arbeit verfolgten Aufgabenstellung sollte die genannte Einteilung jedoch ausreichen.

40 Vgl. [Scharstein2002], S. 13 (bzw. S. 5 in der Online-Version).

3. Verwendetes Datenmaterial

Über die o. g. explizit gestellten Anforderungen ergeben sich einige zusätzliche Notwendigkeiten aus dem Datenmaterial, das mir vom Lehrgebiet Mensch-Computer-Interaktion für den Zweck dieser Arbeit zur Verfügung gestellt wurde. Ich werde dieses Datenmaterial daher im Folgenden kurz vorstellen und dann auf dessen Auswirkungen auf die Aufgabenstellung dieser Arbeit hin befragen.

3.1. Middlebury Stereo Datasets

Die Middlebury-Datensätze bestehen aus mehreren Teilen, deren Bildpaare jeweils unterschiedliche Eigenschaften aufweisen.

Die „2001 Datensätze“[41] beinhalten sechs Szenen. Zu jeder dieser Szenen sind 9 aus paralleler Kameraposition aufgenommene Fotos (nummeriert von 0 bis 9) verfügbar sowie die Disparitätskarten (mit Subpixel-Genauigkeit) für die Fotos in Kameraposition 2 und 7. Die Bilder weisen eine Auflösung von ca. 430 x 380 Pixeln (also ca. 165.000 Bildpunkten) auf. Alle Bilder bestehen aus mehreren unterschiedlich texturierten Ebenen. Zu Evaluationszwecken existieren darüber hinaus spezielle Masken (d. h. spezielle Schwarz-Weiß- oder Graustufen-Bilder), die es ermöglichen, die Evaluation einer berechneten Disparitätskarte auf die nichtverdeckten Pixel oder auf die Diskontinuitäten zu beschränken[42]. Die bekannteste der sechs Szenen der „2001 Datensätze“ ist die Szene „Venus“ (vgl. Anhang 1, Abb. 12), weil das Venus-Bildpaar eines von vier Bildpaaren ist, die für die Reihung in der „Middlebury Stereo Evaluation“ herangezogen werden.

Die „2003 Datensätze“[43] beinhalten die beiden Szenen „Teddy“ (vgl. Anhang 1, Abb. 13) und „Cones“ (vgl. Anhang 1, Abb. 14). Auch von diesen Szenen wurden jeweils 9 Bilder mit paralleler Kameraausrichtung bei konstanten Lichtverhältnissen aufgenommen und zu je zweien davon eine Disparitätskarte in Subpixelgenauigkeit erstellt[44]. Die volle Auflösung der

41 Vgl. http://vision.middlebury.edu/stereo/data/scenes2001/.

42 Da die genannten Masken nicht ganz leicht aufzufinden sind, gebe ich hier die genauen URLs an:
http://vision.middlebury.edu/stereo/eval/newEval/venus/all.png,
http://vision.middlebury.edu/stereo/eval/newEval/venus/nonocc.png,
http://vision.middlebury.edu/stereo/eval/newEval/venus/disc.png.
Abgesehen von diesen Direktlinks kommt man an diese Bilder am einfachsten, wenn man am oberen Ende der Middlebury-Evaluationstabelle im Bereich „Venus“ auf den Hyperlinks „nonocc“, „all“ und „disc“ folgt.

43 Vgl. http://vision.middlebury.edu/stereo/data/scenes2003/.

44 Die Technik, mit der die Disparitätskarten erstellt wurden, unterscheidet sich von jener der 2001 Datensätze und ist dokumentiert in: [Scharstein2003]. Die schwarzen Pixel in den Disparitätskarten bedeuten in diesem Fall, dass die Disparität nicht bestimmt werden konnte, was bei dem angewendeten Verfahren in einigen Bildbereichen der Fall war.

Bilder beträgt 1800 x 1500 Bildpunkte, darüber hinaus werden die Bilder und Disparitätskarten auch in halber (900 x 750) und Viertel-Auflösung (450 x 375 Pixel) zur Verfügung gestellt. In so gut wie allen Veröffentlichungen werden – wie auch in der Middlebury-Evaluationstabelle – die Bilder in der geringsten der genannten Auflösungen, also mit 168.750 Bildpunkten, verwendet. Für die verdeckten und für die untexturierten Bildregionen sowie für die Diskontinuitäten existieren spezielle Evaluationsmasken. Die beiden Bildpaare „Cones“ und „Teddy“ werden ebenfalls in der Middlebury Evaluation berücksichtigt und sind die anspruchsvollsten Bildszenen, die in diese Evaluation einfließen, was sich in deutlich höheren Fehlerraten der entsprechenden Disparitätskarten niederschlägt (ca. 5–6 % Fehlerpixel bei den besten Algorithmen).

Die „2005 Datensätze“[45] beinhalten neun Szenen, die aus je sieben parallelen Kamerapositionen aufgenommen wurden. Die volle Auflösung der aufgenommenen Bilder schwankt zwischen 1330 x 1110 und 1390 x 1110 Pixeln. Außer der vollen Auflösung sind die Bilder auch in halber und in Drittel-Auflösung verfügbar. Die Disparitätskarten wurden in diesem Fall nur für sechs von den neun aufgenommenen Szenen freigegeben. Die übrigen Disparitätskarten sollen vielleicht später einmal veröffentlicht werden. Das Besondere an diesen Datensätzen ist, dass jede dieser Szenen aus jeder Kameraposition in je drei verschiedenen Beleuchtungssituationen und mit je drei verschiedenen Belichtungszeiten aufgenommen wurde. In jeder Kameraposition wurden also neun Fotos aufgenommen, die sich bezüglich des Schattenwurfs und der Helligkeit stark voneinander unterscheiden. Daher eignen sich diese Datensätze auch dazu, die Beleuchtungsabhängigkeit eines Algorithmus zu untersuchen. Leider wird das nur in wenigen Veröffentlichungen gemacht, sodass Vergleichsdaten weitgehend fehlen. Masken, die es ermöglichen würden, speziell die Qualität einer Disparitätskarte in den nicht-verdeckten Bereichen oder in texturarmen Regionen zu messen, werden allerdings für diese Datensätze nicht bereitgestellt.

Die „2006 Datensätze“[46] sind genau gleich aufgebaut wie jene von 2005. Wiederum wurden die Szenen in je drei Beleuchtungssituationen mit je drei Belichtungszeiten aufgenommen und sind außer in der vollen auch in der halben und einer Drittel-Auflösung verfügbar. Die volle Auflösung variiert bei den 21 Szenen dieser Datensätze von 1240 x 1110 bis 1396 x 1110 Pixel. Einige der Bildpaare können als ausgesprochen texturarm bezeichnet werden (z. B. die Szene „Plastic“), allerdings fehlen wiederum Masken, die eine spezielle Evaluation der wenig texturierten Bildregionen erlauben würden.

45 Vgl. http://vision.middlebury.edu/stereo/data/scenes2005/.
46 Vgl. http://vision.middlebury.edu/stereo/data/scenes2006/.

3.2. Vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellte Datensätze

Da Herr Häming, der bei der Erstellung dieser Arbeit mein erster Ansprechpartner für alle auftauchenden Fragen war, der Ansicht war, dass die Middlebury-Datensätze nicht alle Aspekte, die für die am Lehrgebiet erforschte Freihanderfassung von 3D-Modellen relevant sind, ausreichend abdecken, wurden mir von ihm einige speziell auf diese Anforderungen abgestimmte Datensätze zur Verfügung gestellt[47].

3.2.1. Erstellungsverfahren

Es handelt sich dabei um fünf verschiedene Szenen, die in einer völlig anderen Technik als die Middlebury-Datensätze erstellt wurden. Während bei den Middlebury-Datensätzen reale Szenen unter Laborbedingungen fotografiert und dann die Aufnahmen nachbearbeitet wurden, hat Herr Häming einerseits bereits vorhandene und andererseits mit einem 3D-Scanner erstellte 3D-Modelle mit Hilfe des frei verfügbaren 3D-Modellierungswerkzeugs Blender[48] texturiert, mit unterschiedlichen Beleuchtungen versehen und anschließend mithilfe des ebenfalls frei verfügbaren Raytracers YafaRay[49] aus je zwei verschiedenen virtuellen Kamerapositionen gerendert. So entstanden von jeder Szene zwei unterschiedlich beleuchtete linke („links_licht1.png“ und „links_licht2.png“) und ein rechtes („rechts_licht1.png“) Bild. Außerdem wurden für die beiden Aufnahmepositionen Tiefenkarten („depth_left.exr“ und „depth_right.exr“) erstellt, die in einem späteren Schritt in Disparitätskarten umgerechnet werden konnten.

Mithilfe eines von Herrn Häming selbst erstellten Hilfsprogramms namens „depth4disp“[50] wurden die so erhaltenen Bilder – sofern es sich nicht ohnehin schon um eine Off-Axis-Konstellation handelte – unter Verwendung der beim Rendern benutzten Kamerapositionen und -parameter rektifiziert („rect_links_licht1.png“, „rect_links_licht2.png“, „rect_rechts_licht1.png“) und entsprechende rektifizierte Disparitäts- und Okklusionskarten („rightRectDisp_offset_*_scale_*.exr“ bzw. „rightRectDisp_offset_*_scale_*.exr“ sowie

47 Ich möchte mich an dieser Stelle für das außerordentlich große Engagement bedanken, das Herr Klaus Häming bei der Begleitung dieser Arbeit an den Tag gelegt hat. Das wurde nicht nur bei den hier vorgestellten extra für die Zwecke dieser Arbeit erstellten Datensätzen sichtbar, sondern schlug sich auch in einem sehr regen E-Mail-Austausch nieder, in dem mich Herr Häming in vielen einzelnen Fachfragen beriet und mir Rückmeldungen gab.

48 Vgl. http://www.blender.org/.

49 Vgl. http://yafaray.org/.

50 Die verwendeten Hilfsprogramme stehen unter einer „Creative Commons Attribution-Noncommercial-Share Alike 3.0 Germany License“. Ob sie irgendwo auch allgemein zugänglich veröffentlicht wurden, ist mir nicht bekannt, ich gehe aber davon aus, dass sie etwaigen Interessent/inn/en auf Anfrage vom Lehrgebiet Mensch-Computer-Interaktion entsprechend zur Verfügung gestellt werden.

„rightRectDispOcc_offset_*_scale_*.exr“ sowie „rightRectDispOcc_offset_*_scale_*.exr“) berechnet.

Bildrauschen

Die dadurch erhaltenen Bilddateien wurden anschließend mit einem weiteren Hilfsprogramm namens „noise“ künstlich mit verschiedenen Sorten von Bildrauschen versehen. Das Bildrauschen, das den Bildern hinzugefügt wurde, orientiert sich an verschiedenen Rauschsorten, die bei der Aufnahme von Fotos mithilfe von Digitalkameras auftreten[51]: (1) Das sog. „Schrotrauschen“ (engl. „Photon Shot Noise“) ist darauf zurückzuführen, dass die Anzahl der Photonen, die in einem bestimmten Zeitraum auf einem Kamerasensor eintreffen, auch bei gleichmäßiger Beleuchtung zufallsverteilt variiert. (2) Das „Ausleserauschen“ (engl. „Amplifier Noise“ bzw. „Gaussian Noise“) entsteht durch kleine Spannungs- und Temperaturschwankungen, die zu kleinen Lesefehlern der am Auslesevorgang des Digitalbildes beteiligten elektronischen Bauteile führen. (3) Durch Übertragungsfehler kann in Bildern auch das sog. „Impulsrauschen“ (engl. „Salt-and-Pepper Noise“) auftreten, das sich durch helle Pixel in dunklen Regionen und dunkle Pixel in hellen Regionen (daher der englische Name) auszeichnet. (4) Die zur Reduktion des Speicherbedarfs oft eingesetzte JPEG-Komprimierung führt zu mehr oder weniger starken Kompressionsartefakten, die die Qualität der Bilddaten reduzieren.

Da bei zwei der Rauschsorten je zwei verschiedene Rauschstufen verwendet wurden (Ausleserauschen mit den σ-Werten 1 und 2; JPEG-Kompression mit den Qualitätsstufen 100 % und 85 %) und das Schrotrauschen (das auch durch Kameras mit höherer technischer Qualität nicht beseitigt werden kann) mit jedem der anderen Rauschtypen kombiniert wurde, entstanden auf diese Weise von jedem der drei Basisbilder und (sofern eine Rektifizierung überhaupt stattgefunden hatte) der drei rektifizierten Bilder je 12 unterschiedlich verrauschte Varianten, deren Nomenklatur anhand eines linken Basisbilds erläutert werden soll:

links_licht1_s0.0000_g0.0000_i0.0000.png	unverrauscht
links_licht1_s1.0000_g0.0000_i0.0000.png	Schrotrauschen
links_licht1_s0.0000_g1.0000_i0.0000.png	Ausleserauschen, normalverteilt mit σ = 1
links_licht1_s0.0000_g2.0000_i0.0000.png	Ausleserauschen, normalverteilt mit σ = 2
links_licht1_s0.0000_g0.0000_i0.1000.png	Impulsrauschen, Cauchy-verteilt
links_licht1_s0.0000_g0.0000_i0.0000_j100.jpg	JPEG-komprimiert, Qualitätsstufe 100 %

51 Vgl. zum Folgenden: http://theory.uchicago.edu/~ejm/pix/20d/tests/noise/; http://en.wikipedia.org/wiki/Image_noise; http://en.wikipedia.org/wiki/Shot_noise; http://en.wikipedia.org/wiki/Salt_and_pepper_noise; http://de.wikipedia.org/wiki/Bildrauschen; http://de.wikipedia.org/wiki/Schrotrauschen; http://www.filmscanner.info/Bildrauschen.html;

links_licht1_s0.0000_g0.0000_i0.0000_j85.jpg	JPEG-komprimiert, Qualitätsstufe 85 %
links_licht1_s1.0000_g1.0000_i0.0000.png	Schrotrauschen kombiniert mit Auslauserauschen
links_licht1_s1.0000_g2.0000_i0.0000.png	Schrotrauschen kombiniert mit Auslauserauschen
links_licht1_s1.0000_g0.0000_i0.1000.png	Schrotrauschen kombiniert mit Impulsrauschen
links_licht1_s1.0000_g0.0000_i0.0000_j100.jpg	Schrotrauschen und JPEG-komprimiert
links_licht1_s1.0000_g0.0000_i0.0000_j85.jpg	Schrotrauschen und JPEG-komprimiert

3.2.2. Charakteristik der fünf Szenen

Die auf diese Weise erhaltenen Szenen greifen unterschiedliche Probleme auf, die bei der Freihanderfassung von 3D-Objekten auftauchen können[52].

Szene 1: „Sokrates Off-Axis“ (bzw. sokrates_off_axis)

In der ersten Szene (vgl. Anhang 1, Abb. 15[53]) ist eine am Lehrgebiet Mensch-Computer-Interaktion mittels eines 3D-Scanners eingescannte Sokrates-Figur in der Mitte des Bildes platziert. Die Figur selbst nimmt weniger als ein Drittel der Bildfläche in Anspruch, ein großer Teil der Bilder wird also von der wenig texturierten Tischplatte und der Wand hinter der Figur in Anspruch genommen. Die beiden Kameras sind in klassischer seitlich versetzter Position mit paralleler Blickrichtung positioniert. Die zweite Beleuchtungsvariante wirft den Schatten in eine andere Richtung und zeichnet sich durch „wärmere“ Farben aus.

Szene 2: „Raum Off-Axis“ (bzw. raum_off_axis)

Die zweite von parallelen Kamerapositionen aus aufgenommene Szene (vgl. Anhang 1, Abb. 16) wurde von Klaus Häming mit Hilfe frei verfügbarer 3D-Modelle[54] und Texturen[55] erstellt. Der gesamte Raum samt der darin befindlichen Objekte ist stark texturiert, allerdings unterscheiden sich die beiden Beleuchtungsvarianten stark: Während der Raum in der ersten Variante hell erleuchtet ist und sich das Licht in einem Flachbildfernseher spiegelt, kommt das Licht in der zweiten Variante von der gegenüber liegenden Seite und ist sehr viel gedämpfter.

52 Die nachfolgende Darstellung der fünf Szenen orientiert sich an der von Herrn Häming mit dem Datenmaterial mitgelieferten Dokumentation und ergänzt sie um eigene Beobachtungen.

53 Die weißen Pixel im Ground-Truth-Bild markieren verdeckte Bildbereiche.

54 Die verwendeten 3D-Modelle stammen – samt den dazu gehörigen Texturen – von der Seite http://resources.blogscopia.com/ und werden dort von Carlos Folch unter einer Creative-Commons-Lizenz zur Verfügung gestellt, die die Verwendung und Anpassung der Modelle sowohl in kommerziellen als auch in nicht-kommerziellen Projekten erlaubt, vgl. http://resources.blogscopia.com/license-2/.

55 Weitere Texturen wurden von der Seite http://www.polygonblog.com/seamless-texture/ bezogen. Der Autor namens Antti Lehtinen schreibt dazu: „You can use them freely in both personal and commercial projects but you are not allowed to sell them.“

Szene 3: „Clown-Sokrates“ (bzw. clown_sokrates)

In der dritten Szene (vgl. Anhang 1, Abb. 17), die neben dem in Szene 1 verwendeten Sokrates-Modell ein weiteres, mit Hilfe eines 3D-Scanners gewonnenes Modell, einen Gildenclown, zeigt, wird ein/e ungeschickte/r Fotograf/in simuliert: Die zweite Kameraposition ist von der ersten weiter entfernt als sonst, die beiden Kameraachsen überschneiden sich, und die Kamera wurde gegenüber der ersten Position ein wenig um die Kameraachse gedreht (was dann passieren würde, wenn die Kamera bei einer Aufnahme schief gehalten würde). In der ersten Beleuchtungsvariante fällt außerdem ein Schatten auf die Szene – in realen Situationen könnte es sich beispielsweise um den Schatten des Fotografen oder einer daneben stehenden Person handeln. Durch die ungewöhnlichen Kamerapositionen weist das Disparitätsbild relativ große Verdeckungsbereiche auf, außerdem variieren die Disparitäten in diesem Bild besonders stark: von -127,28 bis 124,22.

Szene 4: „Raum Poltergeist“ (bzw. raum_poltergeist)

Auch in dieser Szene (vgl. Anhang 1, Abb. 18) kreuzen sich die Kameraachsen. Die Beleuchtungsvarianten unterscheiden sich nur bezüglich ihrer Helligkeit, allerdings wurden als zusätzliche Schwierigkeit in der zweiten Variante zwei kleinere Objekte (Obstschale, linke Schale auf dem Regalboden im Hintergrund) etwas verrückt. Eine solche Änderung der Szenerie kann bei der Freihanderfassung von 3D-Modellen aufgrund der zeitlichen Verschiebung der beiden Aufnahmen entstehen, beispielsweise, wenn beim Fotografieren eines Gebäudes eine Passantin durchs Bild läuft. Außerdem variieren aufgrund der sich mitten in der Szene überschneidenden Kameraachsen die aus dem Tiefenbild errechneten Disparitäten stark, nämlich von -55,84 bis 117,44.

Szene 5: „Clown-in-front“ (bzw. clown_in_front)

In der letzten der fünf Szenen (vgl. Anhang 1, Abb. 19) befindet sich die „rechte“ Kameraposition hinter der „linken“ (wodurch die Bezeichnungen „rechts“ und „links“ in diesem Fall eigentlich nicht passen). Dadurch sind die rektifizierten Bilder sehr stark verzerrt. Die beiden Beleuchtungsvarianten unterscheiden sich in der Beleuchtungsrichtung und im Farbton (bläulich vs. gelblich).

3.2.3. Anforderungen, die sich aus der Charakteristik des Datenmaterials ergeben

Bei genauerem Hinsehen unterscheiden sich die soeben vorgestellten Bilddaten nicht nur in der Erstellungstechnik, sondern auch in einigen recht grundsätzlichen Aspekten von den Middlebury-Datensätzen, auf die in den meisten einschlägigen Fachartikeln rekurriert wird.

Diese Unterschiede haben einerseits eine spezifische Auswirkung auf die im Einleitungskapitel genannten Anforderungen an den zu erstellenden Stereoalgorithmus, andererseits bringen sie aber auch zusätzliche Erfordernisse mit sich, die über die bereits zu Beginn definierten Anforderungen hinausgehen.

Präzisierung der definierten Anforderungen aufgrund des Bilddatenmaterials

Dass das zur Verfügung gestellte Datenmaterial hohe Ansprüche an die geforderte **Schnelligkeit** des Stereoalgorithmus stellt, wurde bereits angeschnitten und soll hier nur noch etwas erläutert werden. Für eine grobe Abschätzung kann davon ausgegangen werden, dass die Disparitätsberechnung in etwa die Komplexität *O(W H D)* aufweist, wobei mit W die Bildbreite in Pixeln (von engl. „Width"), mit H die Höhe und mit D die Disparitätsspannweite gemeint ist. Während also beim Middlebury-Datensatz „Teddy" 60 Disparitätsstufen für 450 x 375 Pixel berechnet werden müssen, sind es bei „Clown-Sokrates" 254 Disparitätsstufen für ca. 800 x 600 Pixel[56], also ist für „Clown-Sokrates" in einer Grobabschätzung eine ca. 12 Mal so lange Laufzeit anzunehmen wie für das „Teddy"-Bildpaar. Wenn – wie auf meine Rückfrage von Herrn Häming präzisiert – die Laufzeit bei einer Bildgröße von 800 x 600 Pixeln – 30 Sekunden nicht überschreiten sollte, dann musste nach Algorithmen Ausschau gehalten werden, die beim „Teddy"-Bildpaar[57] eine Laufzeit im niedrigen einstelligen Sekundenbereich aufweisen.

In Bezug auf die **Beleuchtungsunabhängigkeit** kann festgehalten werden, dass das Datenmaterial sehr unterschiedliche Ausprägungen von Beleuchtungsvariationen abdeckt: unterschiedliche Beleuchtungsrichtungen, Differenzen in der Beleuchtungsstärke, der Farbtemperatur, Spiegelungen, verschiedene Schattenwürfe. Es ist also nicht möglich, sich auf eine bestimmte Form radiometrischer Veränderungen zu beschränken.

Was die **Eignung für wenig texturierte Bildbereiche** betrifft, so fällt auf, dass die zur Verfügung gestellten Datensätze keine besondere Häufung an untexturierten Bildregionen aufweisen. Daraus darf man schließen, dass dieser Aspekt wohl berücksichtigt werden muss, jedoch nicht unbedingt in wesentlich höherem Maß, als dies auch die Algorithmen tun müssen, die an den Middlebury-Datensätzen erprobt wurden bzw. werden.

---

56 Da sich die Pixelzahl in den rektifizierten Bildern von jener im Originalbild unterscheidet, stimmen die Zahlen nicht genau. Die volle Punktanzahl der rektifizierten Bilder zu verwenden (891 x 756 bei „Clown Sokrates"), wäre auch nicht korrekt, weil die rektifizierten Bilder ja auch an den Rändern Pixel enthalten, die gar nicht gültig sind. Da es hier aber nur um eine grobe Abschätzung der geforderten Schnelligkeit geht, ist es an dieser Stelle ausreichend, dass die ungefähre Größenordnung der Zahlen stimmt.

57 Sofern in den einschlägigen Fachartikeln überhaupt eine Laufzeit angegeben wird, geschieht dies oft in Bezug auf die vier Bildpaare, die in der Middlebury-Evaluation berücksichtigt werden. Daher eignet sich die Laufzeit für das „Teddy"-Bildpaar ganz gut als Orientierungspunkt.

Auswirkungen und zusätzliche Anforderungen, die sich aus der Eigenart des Datenmaterials ergeben

Erst allmählich wurde mir bei meiner Suche nach einem geeigneten, den genannten Anforderungen entsprechenden Stereoalgorithmus bewusst, dass das zur Verfügung gestellte einige weitere Auswirkungen mit sich bringt, deren Tragweite nicht unbedingt sofort sichtbar ist.

Positive und negative Disparitäten. Dass sich die Kameraachsen in zwei der Szenen kreuzen, bringt mit sich, dass in ein und derselben Disparitätskarte sowohl positive als auch negative Disparitäten auftauchen können. Es ist daher nicht mehr möglich, sowohl für die rechten als auch für die linken Disparitätskarten die Absolutwerte zu verwenden, wie es von Scharstein und Szeliski vorgeschlagen wurde[58] und zumindest seither[59] in den allermeisten Veröffentlichungen gehandhabt wird. Stattdessen muss das „Disparity Space Image“ ebenso wie die Disparitätskarten sowohl positive als auch negative Disparitätswerte zulassen. Das scheint zwar keine große Änderung zu sein (und ist auch programmiertechnisch einfach umzusetzen), hat aber doch einige Auswirkungen, derer man sich bewusst sein muss: Da ein positiver Disparitätswert $+d$ im einen Bild einem negativen Disparitätswert $-d$ im anderen Bild entspricht, ist es nun nicht mehr so, dass in allen Disparitätskarten Objekte im Vordergrund hell und der Hintergrund dunkel erscheinen; vielmehr erscheinen die Vordergrundobjekte im linken Disparitätsbild dunkel und im rechten hell. Auf den ersten Blick erscheint daher das linke Disparitätsbild wie das Negativ des rechten.

Unbekannte Disparitätsbandbreite. Die in den fünf Szenen sehr stark unterschiedliche Bandbreite an Disparitätsstufen (-125 bis 128 im maximalen und 31 bis 60 im minimalen Fall) erinnert daran, dass bei einer nicht-kalibrierten Freihanderfassung von 3D-Modellen (in dessen Zusammenhang ja der vorzuschlagende Algorithmus Anwendung finden soll) die Bandbreite der in den Bildern vorkommenden Disparitäten vorerst unbekannt ist. Während die im Hinblick auf die Middlebury-Datensätze entwickelten Stereoalgorithmen (und das sind die allermeisten) also einfach von einer vorgegebenen, relativ begrenzten Disparitätsbandbreite (im Fall des „Teddy“-Bildpaars beispielsweise die Werte von 0 bis 59) ausgehen können[60], muss es im Zusammenhang mit freihändig erstellten Bildpaaren irgendein Verfahren geben, mit dem festgelegt wird, für welche Disparitätswerte überhaupt eine Kostenberechnung und -aggregation stattfinden soll. Dieser Schritt, der in den meisten Artikeln zum Stereo

58 Vgl. [Scharstein2002]. S. 9 (bzw. S. 3 in der Online-Version).

59 Wie verbreitet die Verwendung von Absolutwerten vor der bahnbrechenden Evaluationsstudie von Scharstein/Szeliski war, habe ich nicht weiter recherchiert und ist hier auch irrelevant.

60 Die meisten einschlägigen Fachartikel gehen auf die Frage der Disparitätsbandbreite überhaupt nicht ein, daher kann man wohl davon ausgehen, dass die Disparitätsberechnung sich auf einen fix vorgegebenen Disparitätsbereich bezieht.

Matching überhaupt nicht diskutiert wird, ist von großer Bedeutung: Wird die Bandbreite zu groß gewählt, werden jede Menge unnötiger Berechnungen durchgeführt; die dafür nötige Rechenzeit sollte aber sinnvoller für die Verbesserung des Ergebnisses verwendet werden. Wird die Bandbreite hingegen zu eng oder falsch gewählt, dann werden u. U. die korrekten Disparitätsstufen gar nicht in die Berechnung mit einbezogen – die Disparitätswerte, die für Pixel berechnet wurden, deren korrekte Disparität sich nicht in der gewählten Bandbreite befindet, sind dann trotz des Berechnungsaufwands auf jeden Fall falsch.

Ungültige Bildbereiche. Dass drei der fünf vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellten Szenen keine Off-Axis-Konstellation (also Aufnahme der beiden Bilder aus geringem Kameraabstand mit paralleler Kameraachse) verwenden, hat – worauf ja bereits hingewiesen wurde – auch noch zur Folge, dass die rektifizierten Bilddateien Bereiche aufweisen, die eigentlich als ungültig bezeichnet werden müssten. Da die rektifizierten Bilddaten in diesem Fall keine rechteckige Form aufweisen, gängige Dateiformate aber von rechteckigen Bildern ausgehen, bleibt nichts anderes übrig, als um die eigentlichen Bilddaten ein Rechteck zu legen, das groß genug ist, um alle diese Bilddaten zu umschließen. Die nicht zu den eigentlichen Bilddaten gehörigen Regionen dieses Rechtecks sind in den rektifizierten Bildern als schwarze Flächen an den Rändern erkennbar (sehr gut sichtbar in den Abb. 18 und 19). Diese ungültigen Bildbereiche werfen unangenehme Fragen auf: (a) Sowohl einige Kostenfunktionen (wie die an späterer Stelle vorgestellte Hamming-Distanz der Census-Transformation) als auch viele Varianten der Kostenaggregation (die ja bei lokalen Algorithmen eine zentrale Rolle spielt) benötigen die Pixelwerte in einem fixen Fenster um jedes Pixel, dessen Disparität berechnet werden soll. Wie soll in diesem Fall vorgegangen werden, wenn ein Teil dieses Fensters aus ungültigen Pixeln besteht? (b) Wenn auch für diese ungültigen Pixel die Berechnungen durchgeführt würden wie für die gültigen, würde nicht nur die Berechnungsdauer unnütz verlängert, sondern die falsch berechneten Disparitätswerte (an ungültigen Pixeln kann es ja nur falsch berechnete Werte geben!) könnten – aufgrund der Kostenaggregation – im schlimmsten Fall sogar in die gültigen Bildbereiche quasi verschleppt werden und so das Gesamtergebnis negativ beeinflussen. Wie kann man also vermeiden, dass die Berechnungen auch für diese ungültigen Pixel durchgeführt werden?

3.2.4. Konsequenzen für die vorliegende Arbeit

Angesichts dieser sich durch das Datenmaterial ergebenden Anforderungen stellte sich im Erstellungsprozess dieser Arbeit die Frage, wie damit umgegangen werden sollte. Während manche der genannten Herausforderungen (z. B. bzgl. der geforderten Schnelligkeit) ohnehin

unumgänglich gewesen wären, wäre es bei anderen (Abschätzung der Disparitätsbandbreite; Eignung für Aufnahmen, bei denen sich die Kameraachsen überkreuzen) u. U. auch möglich gewesen, darauf hinzuweisen bzw. sich darauf zu berufen, dass diese Anforderungen in der ursprünglichen Themenstellung nicht genannt worden waren, und in der Folge die Suche nach einer effektiven Lösung des Problems zu vermeiden (also z. B. für jede der fünf Szenen einfach eine entsprechende fixe Disparitätsbandbreite zu verwenden).

Im Hinblick auf die am Lehrgebiet entwickelte Demo-Anwendung zur Freihanderfassung von 3D-Modellen erschien es mir jedoch sinnvoller, den genannten Herausforderungen nicht auszuweichen, sondern mich ihnen zu stellen. Ich habe daher – teilweise nach Rücksprache mit Herrn Häming, teils aber auch in Eigenregie – viel Zeit und Energie in einige der o. g. Probleme investiert und meine, dass es mir gelungen ist, insbesondere in Bezug auf eine sehr flexible Abschätzung des Disparitätssuchbereichs neue, innovative Wege zu gehen.

Freilich hat diese Vorgangsweise auch ganz wesentlich dazu beigetragen, dass es mir aufgrund der von der Prüfungsordnung eng begrenzten Bearbeitungszeit bei einigen Aspekten des hier vorgestellten Algorithmus nicht mehr möglich war, das vorhandene Potenzial an Verfeinerungen und Verbesserungen auch noch programmiertechnisch umzusetzen. Auch für eine abschließende Evaluation des Algorithmus[61] blieb nicht mehr so viel Zeit, wie ich mir gewünscht hätte. Ich werde daher in einem Schlusskapitel auf diese offen gebliebenen Verbesserungsmöglichkeiten hinweisen und hoffe, dass die gewählte Vorgangsweise – trotz der dadurch entstandenen Lücken – wegen des Innovationspotenzials des vorgestellten Verfahrens zur Abschätzung des Disparitätssuchraums auch für andere als gerechtfertigt erscheint.

61 Während der Erstellung des Algorithmus habe ich eigentlich ständig Teile desselben evaluiert, beispielsweise verschiedene Kostenfunktionen ausprobiert und verglichen. Soweit ich diese Zwischenevaluationen ausreichend dokumentiert habe, werden sie in die nachfolgende Darstellung einfließen.

4. Lektüreergebnisse der einschlägigen Fachliteratur

Da das Stereo Matching inzwischen seit mehreren Jahrzehnten intensiv beforscht wird und mittlerweile eine beträchtliche Anzahl an einschlägigen Artikeln vorliegt, ist eine gründliche Berücksichtigung der vorliegenden Fachliteratur unerlässlich. Ein besonderes Augenmerk wird dabei sinnvoller auf Evaluationsstudien liegen, die Aspekte der verschiedenen Ansätze vergleichen, insbesondere, wenn sie sich auf Aspekte der in dieser Arbeit gestellten Anforderungen beziehen (Abschnitt 4.1). Darüber hinaus gilt es aber auch, in den Beschreibungen der zum Stereo Matching vorgeschlagenen Verfahren nach Ideen Ausschau zu halten, die bei der Bewältigung der gestellten Aufgabe hilfreich sein könnten (Abschnitt 4.2)

4.1. Evaluationsstudien

4.1.1. „A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms“

Die wohl berühmteste einschlägige Evaluationsstudie wurde von Daniel Scharstein und Richard Szeliski 2002 veröffentlicht[62]. Der Einfluss dieses Artikels auf die alle danach geschriebenen kann wohl kaum überschätzt werden: Man kann nicht nur feststellen, dass sich so gut wie alle neueren Veröffentlichungen auf die eine oder andere Weise auf die o. g. Evaluationsstudie beziehen, sondern auch, dass die vorgeschlagene Repräsentation der Zwischenergebnisse der Berechnung in einem „Disparity Space Image“ und die Verwendung der Absolutwerte für die Disparitäten (vgl. Abschnitt 2.2), die Taxonomie (1. Kostenberechnung; 2. Kostenaggregation; 3. Disparitätsberechnung; 4. Disparitätsverfeinerung) sowie die Kategorisierung der Verfahren (lokale und globale Algorithmen, dynamische Programmierung und kooperative Algorithmen) zum Allgemeingut geworden sind. Vor allem aber messen die meisten Autorinnen und Autoren ihre vorgestellten Verfahren an den in der parallel zum genannten Artikel im Internet veröffentlichten Evaluationstabelle[63], die inzwischen bereits in der zweiten Version vorliegt und (Stand: September 2012) mehr als 130 Algorithmen enthält.

So einflussreich und bahnbrechend diese Studie auch sein mag – für die Zwecke dieser Arbeit ist sie nur von begrenzter Bedeutung: (1) Dass die Repräsentation der Disparität als Absolutwert versagt, wenn sich die Kameraachsen der beiden Bilder überschneiden, wurde bereits diskutiert (vgl. Abschnitt 3.2.3). Auch die Repräsentation in einem fixen dreidimensio-

62 Vgl. [Scharstein2002].
63 Vgl. http://vision.middlebury.edu/stereo/eval/.

nalen „Disparity Space Image“ hat sich bei der Erstellung des Stereoalgorithmus in dieser Arbeit als problematische Einschränkung erwiesen (vgl. dazu Kap. 5). (2) Die bei der Themenstellung für diese Arbeit ausdrücklich genannten (Schnelligkeit, Beleuchtungsunabhängigkeit, Eingung für schwach texturierte Szenen) und die im zur Verfügung gestellten Datenmaterial implizierten (Abschätzung des vorerst unbekannten Disparitätsbereichs, Umgang mit ungültigen Bildbereichen) Anforderungen spielen in der Middlebury-Evaluation eine untergeordnete oder gar keine Rolle[64].

Das hat leider auch zur Folge, dass bei der Lektüre der Fachartikel der Eindruck entsteht, dass die meisten entwickelten Verfahren sich stark daran orientieren, einen guten Platz in der Middlebury-Evaluationstabelle zu erreichen und folgerichtig die genannten für diese Arbeit zentralen Anforderungen wenig Berücksichtigung finden.

Das zeigt sich recht gut an einem einfachen Beispiel: Der mit „ADCensus“ bezeichnete Top-Performer der Middlebury-Evaluationstabelle verwendet (wie der Name schon andeutet) als Kostenfunktion eine Kombination zweier Kostenfunktionen: „AD“ steht für „absolute Differenzen“ (womit die Differenz zwischen den Intensitätswerten der beiden beteiligten Pixel gemeint ist), „Census“ für die Hamming-Distanz der Census-Transformation der beiden Pixel (für eine genauere Beschreibung vgl. die nachfolgenden Abschnitte). Eine etwas abgewandelte eigene Implementierung dieser Kostenfunktion zeigte zwar bessere Ergebnisse bei den unverrauschten und gleich beleuchteten Bildpaaren, aber schlechtere Ergebnisse bei ungleicher Beleuchtung der beiden Fotos und vor allem bei stärkerem Rauschen.

Ohne die Bedeutung der Evaluationsstudie und der dazu gehörigen Evaluationstabelle von Scharstein und Szeliski schmälern zu wollen, muss daher festgehalten werden, dass es nicht sinnvoll wäre, sich einseitig daran zu orientieren[65].

4.1.2. Kostenberechnung

Die Berechnung der Kosten für das mögliche Matching zweier Pixel steht am Anfang aller Stereoalgorithmen und spielt zweifellos eine besonders wichtige Rolle, weil sich Schwächen, die sich ein Verfahren am Anfang erlaubt, später nur schwer wieder ausmerzen lassen. Es

64 Allenfalls könnte man noch etwas über die Eignung für wenig texturierte Bildpaare herausfinden, wenn man die Performance der Algorithmen bei den verschiedenen (unterschiedlich stark texturierten) Bildpaaren vergleicht. Allerdings ist auch dann noch immer nicht gesichert, dass ein etwaiges besseres Abschneiden eines Verfahrens bei einem weniger texturierten Bild tatsächlich auf dessen bessere Eignung für wenig texturierte Bilder zurückgeht, weil sich die Bildpaare ja nicht nur in diesem einen Punkt voneinander unterscheiden und das bessere Abschneiden auch auf eine andere Eigenschaft des jeweiligen Bildpaares zurückzuführen sein könnte. Außerdem wurde ja bereits besprochen, dass keines der in der Evaluationstabelle berücksichtigten Bildpaare als ausgesprochen wenig texturiert bezeichnet werden kann.

65 Nicht unerwähnt soll bleiben, dass die Evaluationstabelle unter http://vision.middlebury.edu/stereo/eval/ den sehr angenehmen Nebeneffekt hat, dass sie eine große Anzahl einschlägiger Fachartikel verlinkt und dadurch die Literatursuche für die vorliegende Arbeit deutlich vereinfacht hat.

verwundert daher wenig, dass sich eine ganze Reihe von Studien mit verschiedenen Aspekten der Kostenberechnung beschäftigen.

Michael Bleyer und Sylvie Chambon haben sich (teilweise mit noch weiteren Autorinnen) in zwei Artikeln mit der Frage auseinandergesetzt, welche Rolle Farbinformationen bei Stereo Matching spielen bzw. spielen sollen[66]. Während sie im ersten der beiden Artikel noch zum Schluss kommen, dass die Verwendung von Farbinformationen messbar zu besseren Ergebnissen führt[67], kommen sie in der zweiten Studie, in der sie Kostenfunktionen mit einbezogen haben, die sich in einer anderen Studie (s. u.) als relativ resistent gegenüber Beleuchtungsvarianzen gezeigt haben, zum Resultat, dass die Verwendung von Farbinformationen u. U. sogar kontraproduktiv sei:

> The results speak a clear language. ZNCC and Census have the same effect as color matching, i.e. they reduce the error in radiometric distorted regions. However, ZNCC and Census are considerably more effective in this respect. In addition, both measures are even capable of reducing the errors in radiometric clean regions, which leads to the over-all best performance.[68]

Konsequenterweise empfehlen Bleyer und Chambon am Ende ihres Artikels, auf die Verwendung von Farbinformationen beim Stereo Matching zu verzichten.

Die Evaluationsstudie, auf die sich Bleyer und Chambon bezüglich der radiometrisch unempflindlichen Kostenfunktionen beziehen und die selbstverständlich auch für diese vorliegende Arbeit von zentraler Bedeutung ist, stammt von Heiko Hirschmüller und Daniel Scharstein. Die beiden Autoren haben in dieser Studie 15 verschiedene Kostenfunktionen daraufhin untersucht, wie sehr sie sich bei Bildpaaren mit radiometrischen Differenzen (Belichtungsdauer, Vignettierung, wechselnde Lichtverhältnisse, Bildrauschen) bewähren. Dazu wurden die Kostenfunktionen – soweit das möglich war[69] – je einmal mit einem lokalen Algorithmus (Kostenaggregation mit 9 x 9 Pixel großen Fenstern), einem globalen Verfahren (Graph Cuts) und einer Methode, die zwischen diesen beiden Polen liegt (semi-globales Matching) verwendet und die Ergebnisse verglichen.

Am Ende der Studie heben die Autoren vier Kostenfunktionen besonders hervor: „Bilateral Background Subtraction“ (abgekürzt als „BilSub“) erwies sich sehr gut, wenn die radiometrischen Differenzen nicht zu groß wurden; „Zero-mean Normalized Cross-Correlation“

66 Vgl. [Bleyer2008a]; [Bleyer2010a].

67 Da sich die Studie auf globale Algorithmen bezieht, müsste man eigentlich erst überprüfen, ob das Ergebnis auch für lokale Algorithmen gilt; angesichts des Resultats der zweiten Studie erübrigt sich eine solche Überprüfung aber ohnehin.

68 [Bleyer2010a], S. 7. Als weiteren Grund, warum Farbinformationen doch keine so große Bedeutung haben wie im ersten Artikel angenommen, führen die beiden an, dass die Farben von heutigen Kameras weniger robust aufgenommen werden als die Intensitätsinformationen, vgl. ebd., S. 8.

69 Vier der Kostenfunktionen lassen sich nur mit dem lokalen Algorithmus sinnvoll verwenden.

(„ZNCC“) erwies sich bei größeren radiometrischen Unterschieden als robuster, allerdings hat es von allen getesteten Funktionen die größte Fehlerrate an den Diskontinuitäten; „Census“ (genauer: die Hamming-Distanz der Census-Transformation) zeigte sich bei allen Sorten radiometrischer Differenzen als sehr gut, ausgenommen bei starkem Bildrauschen, gegenüber „BilSub“ und „ZNCC“ brachte „Census“ fast durchwegs die besseren Ergebnisse hervor; „Hierarchical Mutual Information“ („HMI“) brachte schließlich bei starkem Bildrauschen die besseren Ergebnisse hervor, zeigte allerdings Probleme bei lokalen Beleuchtungsdifferenzen, also bei unterschiedlicher Beleuchtungsrichtung oder Vignettierung[70]. Die Vorrangstellung der „Census“-Kostenfunktion hat Hirschmüller übrigens offenbar so sehr beeindruckt, dass er in dem von ihm selbst entwickelten Verfahren, dem „semi-globalen Matching“ (SGM) inzwischen die jahrelang verwendete Kostenfunktion HMI durch Census ersetzt hat[71].

Daniel Neilson und Yee-Hong Yang gehen in zwei Studien[72] an die Frage nach der „besten“ Kostenfunktion mit einer Methodik heran, die jener von Hirschmüller/Scharstein zwar in ihrem Grundgerüst ähnelt, sich in den Details aber stark unterscheidet: Sie zerlegen gängige Kostenfunktionen in ihre Bestandteile (Berechnung der Kosten für jeden Farbkanal getrennt; Kombination der drei Farbkanäle; räumliche Aggregation innerhalb eines fixen Fensters[73]), wenden diese Kostenfunktion in der ersten Studie mit einem Verfahren (Hierarchical Belief Propagation) und in der zweiten mit vier verschiedenen Algorithmen (zwei Versionen des semi-globalen Matchings und zwei Versionen der „Hierarchical Belief Propagation“) an, um die Disparitätskarten für eine sehr große Zahl an Bildpaaren[74] zu berechnen.

---

70 Übrigens untersuchten Hirschmüller/Scharstein auch den Einfluss der Verwendung von Farbinformationen und kamen dabei zum Schluss: „We also investigated the potential benefit of using color information, which appears to be rather small, and in some cases color is even detrimental. This is clearly an important topic for future research.“ Diese dem ersten Ergebnis von Bleyer/Chambon (vgl. [Bleyer2008a]) wiedersprechende Ergebnis nahmen die beiden Letztgenannten zum Anlass, ihr erstes Ergebnis zu überprüfen und zu revidieren, vgl. [Bleyer2010a].

71 Während Hirschmüller in seinem letzten ausführlichen Artikel noch HMI als Kostenfunktion beschreibt (vgl. [Hirschmüller2008]), gibt er in einer Studienunterlage zur „photogrammetrischen Woche“ – nach einer ausführlichen Diskussion der Vor- und Nachteile von Census und HMI – an, dass er nun Census im Zusammenhang mit SGM verwende, vgl. [Hirschmüller2011], S. 174f.

72 Vgl. [Neilson2008]; [Neilson2011].

73 Neilson/Yang rechnen die Kostenaggregation innerhalb eines fixen Fensters noch zur Berechnung der pixelweisen Kosten – dieser Schritt wird sonst meist bereits zur Kostenaggregation gerechnet.

74 Im ersten der beiden Artikel ist angeführt, dass der Algorithmus 11,76 Millionen Mal (!) laufen musste: 1944 Kostenfunktionen wurden mit je 12 Parametereinstellungen an 252 Bildpaaren erprobt. Neben 162 Bildpaaren der Middlebury Datasets wurden auch 90 selbst produzierte verwendet, deren Erstellung übrigens in manchen Aspekten (Erstellung der Bilder mit Hilfe eines Raytracers, nachträgliches Hinzufügen von Bildrauschen) der Vorgangsweise Klaus Hämings entspricht. Die Berechnung hätte auf einem einzelnen Xeon-Prozessor ca. 4,8 Jahre benötigt, der verwendete Cluster schaffte es in zehn Wochen, vgl. [Neilson2008], S. 6. In der zweiten Studie („nur“ 272 Kostenfunktionen und 171 Bildpaare, aber vier verschiedene Stereoalgorithmen und ein wesentlich komplizierteres Verfahren zur Ermittlung der optimalen Parameter), wären dann schon 1,5 Dekaden an Computerrechenzeit erforderlich gewesen – wie lange der Cluster tatsächlich brauchte, ist hier nicht angegeben.

Allerdings sind die Ergebnisse – trotz des immensen Datenmaterials, auf dem sie basieren – für die Zwecke dieser Arbeit wenig brauchbar, weil weder Census noch HMI – die beiden Kostenfunktionen, die in Hirschmüllers und Scharsteins einschlägiger Evaluationsstudie als die für Beleuchtungsschwankungen resistentesten hervorgegangen sind – in der Studie von Neilson/Yang keinerlei berücksichtigung fanden[75]. Lediglich einige Detailergebnisse[76] könnten auch für diese Arbeit von Interesse sein: So kommen die Autoren beispielsweise zum Schluss, dass eine Trunkierung der Kostenfunktion deren Anfälligkeit gegenüber Bildrauschen reduzieren kann[77]; auch die Beobachtung, dass ein größerer Radius bei der Kostenaggregation nicht unbedingt zu besseren Ergebnissen führt, sondern ab einem bestimmten Grenzwert (der je nach Kostenfunktion variiert) sogar das Gegenteil bewirken kann, könnte auch für die vorliegende Arbeit von Bedeutung sein.

Zhu et al. vergleichen die Performance von Mutual Information (MI), Census und einer aus MI und Census kombinierten Kostenfunktion bei Luftbildern und Satellitenaufnahmen, die – im Unterschied zu den Middlebury-Bildpaaren – aus großer Distanz und aus relativ großem Abstand erstellt worden sind. Als Rahmenverfahren für den Vergleich wird das semi-globale Matching (SGM) verwendet. Wie schon in der Studie von Hirschmüller/Szeliski wird auch in dieser Studie festgestellt, dass MI (bzw. HMI) in Bildern mit geringen radiometrischen Unterschieden die etwas besseren Ergebnisse zeigt, während Census in Bildern mit lokalen Beleuchtungsschwankungen deutlich besser abschneidet. Darüber hinaus wird festgehalten, dass MI die Kanten an den Diskontinuitäten besser erhält und weniger verrauschte Disparitätskarten erzeugt, Census sich aber bei größerem Kameraabstand besser bewährt. Die als dritte Möglichkeit vorgeschlagene und getestete gewichtete Summe von MI und Census vereinigt die Vorteile der beiden einzelnen Funktionen und übertrifft sie bei Bildpaaren, die aus großer Entfernung aufgenommen wurden[78].

Simon Hermann et al. untersuchen drei Kostenfunktionen: Census, die Differenz der Gradienten und die in einem 3 x 3 Pixel großen Fenster ermittelte Summe der absoluten Differenzen in ihrer ursprünglichen („SAD“) sowie in einer speziellen Abwandlung (nicht auf das ur-

75 Etwas überspitzt formuliert könnte man die These aufstellen, dass Neilson/Yang mit sehr hohem rechnerischem Aufwand die Effizienz verschiedener Varianten von Kostenfunktionen erforscht haben, die in den letzten Jahren ohnehin bereits weitgehend von anderen Kostenfunktionen – die in der rechenintensiven Studie nur zum geringen Teil Berücksichtigung fanden – abgelöst worden sind.

76 Ich beziehe mich im Folgenden nur auf Ergebnisse der zweiten Studie, einerseits, weil ich davon ausgehe, dass die Erkenntnisse der ersten Untersuchung in die zweite eingeflossen sind, und andererseits, weil die zweite Studie auch deutlichere Aussagen zu den Ergebnissen macht.

77 Allerdings müsste erst untersucht werden, ob diese Beobachtung auch auf die Census-Kostenfunktion übertragbar ist. Immerhin gibt es bei einem – oft verwendeten – Census-Fenster von 9x7 Pixeln ohnehin „nur“ 62 verschiedene Kostenstufen – im Vergleich zu beispielsweise 765 bei AD; man müsste also überprüfen, ob eine Trunkierung bei Census nicht zu negativen Nebeneffekten führen würde, die die positiven Effekte zunichte machen.

78 Vgl. [Zhu2011], S. 165f.

sprüngliche Bilderpaar, sondern auf deren sog. „Texturkomponente“ angewendet, „RSAD“)[79]. Der Performance-Vergleich wird auch in dieser Studie mittels semi-globalem Matching (SGM) durchgeführt. Als Ergebnis wird festgehalten, dass sich Census, Gradientendifferenz und RSAD im Großen und Ganzen ähnlich gut schlagen. Das wird darauf zurückgeführt, dass sich alle drei genannten Kostenfunktionen letztlich auf den Gradienten zurückführen lassen:

> The census function describes the gradient in a rough sense, which makes it robust to noise. The gradient and RSAD function adds intensity information on a relative scale to the cost, which adds more descriptive information to the cost. However, this makes those functions more affected by noise than the census.[80]

Auch wenn die Performance dieser drei Kostenfunktionen nicht in jedem Bilderpaar gleich ist, so wird also in der Gesamtbetrachtung Census attestiert, insgesamt etwas bessere Ergebnisse zu produzieren, insbesondere bei verrauschten Bildern.

4.1.3. Kostenaggregation

Die o g. Studie von Neilson/Yang markiert bereits den Übergang zu den Studien zur Kostenaggregation, weil darin auch bereits einige Formen der Kostenaggregation eingeflossen sind. So kommen Neilson und Yang u. a. zum Schluss, dass eine lokal begrenzte Kostenaggregation die Wahrscheinlichkeit für ein gutes Ergebnis des Stereoalgorithmus erhöht; außerdem erweist sich von den fünf geprüften Methoden der Kostenaggregation[81] die als „Adaptive Support Weight“ genannte Methode als die beste[82].

Diese Ergebnisse überraschen allerdings kaum, weil auch schon frühere Evaluationsstudien zur Kostenaggregation zu ähnlichen Ergebnissen gekommen sind. Wang et al. vergleichen ebenfalls fünf verschiedene Aggregationsmethoden[83], und zwar solche, die sich für die Berechnung mittels Grafikprozessor optimieren lassen („square-window“, „shiftable-window“, „oriented-rod“, „boundary-guided“, „adaptive-weight“). Auch in dieser Studie schneidet „Adaptive Weight“ (in einer für die Berechnung mittels GPU vereinfachten Variante) am besten ab, erfordert aber auch die größte Rechenkomplexität aller getesteten Verfahren.

Gong et al. vergleichen ebenfalls die Performance gängiger Kostenaggregationsverfahren bei deren Berechnung mithilfe des Grafikprozessors. Neben den fünf von Wang et al.

79 Vgl. [Hermann2011].

80 [Hermann2011], S.

81 Die fünf getesteten Varianten sind: (1) Mittelwert der Pixel in einem fixen Fenster; (2) Gewichtung nach der räumlichen Distanz; (3) Gewichtung nach der farblichen Distanz; (4) sog. „bilaterale“ Gewichtung sowohl nach räumlicher als auch nach farblicher Distanz; (5) „Adaptive Support Weight“, d. h. Gewichtung nach räumlicher und farblicher Distanz sowohl beim Pixel des Basisbilds als auch beim für das Matching vorgesehenen Pixel des Referenzbilds.

82 Auch hier bleiben neuere Ansätze, die die beträchtliche Rechenzeit, die „Adaptive Support Weight“ benötigt, zu reduzieren versuchen, leider unberücksichtigt.

83 Vgl. [Wang2006].

untersuchten Methoden[84] wird auch noch eine als „adaptive-window“ bezeichnete Variante berücksichtigt. Als ein wichtiges Ergebnis halten sie fest, dass die Kostenaggregationsverfahren bessere Ergebnisse hervorbringen, wenn als Kostenfunktion nicht die absolute Differenz in ihrer ursprünglichen Form verwendet wird, sondern eine Trunkierung (also eine Begrenzung der maximalen Kosten) durchgeführt wird[85]. Wie schon in der zuvor genannten Studie wird auch in dieser die Adaptive-Weight-Aggregation als die qualitativ beste identifiziert, aber auch die relativ lange Rechenzeit bemängelt.

Tombari et al. vergleichen 14 verschiedene Methoden der Kostenaggregation. Auch sie kommen zum Ergebnis, dass verschiedene von ihnen getesteten Varianten des Adaptive-Weight-Verfahrens die besten Ergebnisse hervorbringen, aber auch vergleichsweise lange Laufzeiten erfordern.

4.1.4. Maße für die Zuverlässigkeit der Ergebnisse

Eine neuere Evaluationsstudie untersucht verschiedene Messmethoden für die Zuverlässigkeit der durch einen Stereoalgorithmus erhaltenen Ergebnisse[86]. Xiaoyan Hu und Philippos Mordohai vergleichen 17 Verfahren, deren Zweck es ist, die Wahrscheinlichkeit einzuschätzen, dass ein bestimmter Disparitätswert korrekt ist. Ein solches Zuverlässigkeitsmaß kann als gut gelten, wenn es den verdeckten Pixeln und den in einer Disparitätskarte falsch zugeordneten Pixeln einen niedrigen Zuverlässigkeitswert zuweist, korrekte Disparitätswerte hingegen mit hohen Zuverlässigkeitswerten versieht.

Die Ergebnisse dieser Studie sind vielfältig, daher greife ich nur jene Aspekte heraus, die für den Zusammenhang dieser Arbeit von Bedeutung sein könnten. Die Studie bestätigt, dass „Matching Score Measure (MSM)“ (bei dem ein Pixel ganz einfach mit einer umso höheren Zuverlässigkeit gekennzeichnet wird, je geringer die Kosten für die gewählte Disparität sind) und „Left Right Consistency (LRC)“ (bei dem je eine Disparitätskarte für beide Bilder des Bildpaars berechnet und dann gemessen wird, wie gut die jeweils berechneten Disparitäten zueinander passen) sich sehr gut eignen, um Verdeckungen aufzufinden.

Bei der Vorhersage, ob ein Disparitätswert korrekt ist, erweist sich eine von Hu und Mordohai neu vorgeschlagene Messmethode namens „Left Right Difference (LRD)“ als die beste.

---

84 Genau genommen stimmen auch diese fünf Aggregationsmethoden bei den beiden Studien nicht in allen Details überein. Das ist aber hier nicht weiter von Bedeutung.

85 Allerdings wird auch hier nicht weiter untersucht, ob sich diese Beobachtung auch auf andere Kostenfunktionen, wie z. B. Census übertragen ließe.

86 Vgl. [Hu2012].

4.2. Anregungen aufgrund der vorgeschlagenen Stereoalgorithmen

Abgesehen von den präsentierten Evaluationsstudien gibt es noch eine Unzahl an Beschreibungen einzelner vorgeschlagener Stereoalgorithmen, die wertvolle Hinweise enthalten, wie die für diese Arbeit relevanten Anforderungen erfüllt werden könnten. Es ist unmöglich, auch nur annähernd einen Überblick über diese Fülle an mehr oder weniger innovativen Ideen zu geben. Im Folgenden werde ich daher nur jene Ansätze beschreiben, die eine besondere Bedeutung für den in dieser Arbeit entwickelten Stereoalgorithmus haben.

4.2.1. Kostenberechnung

Auffällig ist, dass die Popularität der Census-Kostenfunktion in jüngerer Zeit – wohl unter dem Einfluss der Evaluationsstudie von Hirschmüller/Scharstein[87] – enorm gestiegen ist. Viele der in der Middlebury-Evaluation besonders gut abschneidenden Algorithmen verwenden allerdings keine reine Census-Funktion, sondern eine Kombination mit einer zweiten Kostenfunktion, meistens mit einer der Varianten von AD („absolute Differenz“).

Der ADCensus-Algorithmus[88] verwendet beispielsweise für die Kombination der beiden Teile die folgende Funktion[89]:

$$C(p,d)=\left(1-\exp\left(-\frac{C_{census}(p,d)}{\lambda_{census}}\right)\right)+\left(1-\exp\left(-\frac{C_{AD}(p,d)}{\lambda_{AD}}\right)\right)$$

Die beiden Werte λ_{AD} und λ_{census} ermöglichen in dieser Formel auf einfache Weise, Ausreißer zu kontrollieren. Eine solche Kombination kann offenbar die Qualität der Ergebnisse deutlich verbessern[90], zwingt allerdings dazu, auch die weiteren Berechnungen in Gleitkommaarithmetik durchzuführen, sodass nicht nur die Kostenberechnung (im engeren Sinn) relativ rechenintensiv ist, sondern auch die nachfolgenden Schritte, v. a. die Kostenaggregation[91].

87 Vgl. [Hirschmüller2009].

88 Vgl. [Mei2011], S. 2. Eine andere Verbindungsvariante findet sich beispielsweise in [Çığla2012], S. 3.

89 Die beiden Teilfunktionen C_{AD} und C_{census} werden im nachfolgenden Kapitel genauer erklärt.

90 Die Autoren geben an, dass die Prozentzahl der nicht-verdeckten Fehler sich um 1,96 % (Tsukuba), 0,4 % (Venus), 1,36 % (Teddy) und 1,52 % verringert.

91 Die Autoren resümieren zwar optimistisch: „And this improvement comes at low additional computation cost.“ Dieses Resümee stimmt aber nur, wenn man isoliert die Rechenzeit für die Kostenfunktion betrachtet und davon ausgeht, dass die nachfolgenden Schritte in jedem Fall auch in Gleitkommaarithmetik durchzuführen wären. Wenn man jedoch auch noch die Kostenaggregation in den Vergleich mit hineinnimmt, die im einen Fall in Gleitkommaarithmetik erfolgen muss, im anderen Fall jedoch nur Ganzzahlarithmetik erfordert, ist der Unterschied beträchtlich. Leider habe ich meine eigenen diesbezüglichen Vergleichsversuche nicht gut genug dokumentiert, um dafür Zahlen angeben zu können – der Unterschied war jedoch deutlich genug, dass ich die Variante, die Kostenaggregation in Gleitkommaarithmetik durchzuführen, verwarf.

Allerdings gibt es auch Algorithmen, die zwei Kostenfunktionen in Ganzzahlarithmetik verbinden, wie z. B. der RDP-Algorithmus:

$$C(p,d)=min(C_{AD}(p,d),\lambda_{AD})+min(C_{census}(p,d),\lambda_{census})$$

Auch hier dienen die beiden Werte λ_{AD} und λ_{census} der Beseitigung von Ausreißern, allerdings handelt es sich in diesem Fall um fixe Schwellenwerte. Bei einer solchen Verbindung kann im weiteren Verlauf des Algorithmus Ganzzahlarithmetik verwendet werden.

Der Vorteil einer solchen Kombination liegt vor allem darin, dass die Stärken der einen Teilfunktion in der Lage sind, die Schwächen der jeweils anderen Teilfunktion auszugleichen. Während AD Schwierigkeiten mit texturarmen Bildregionen hat, versagt Census mitunter bei sich wiederholenden Bildstrukturen, wie sie z. B. bei Wandtapeten auftreten können. Die Kombination der beiden Kostenfunktionen zeigt in beiden Problembereichen gute Ergebnisse[92]:

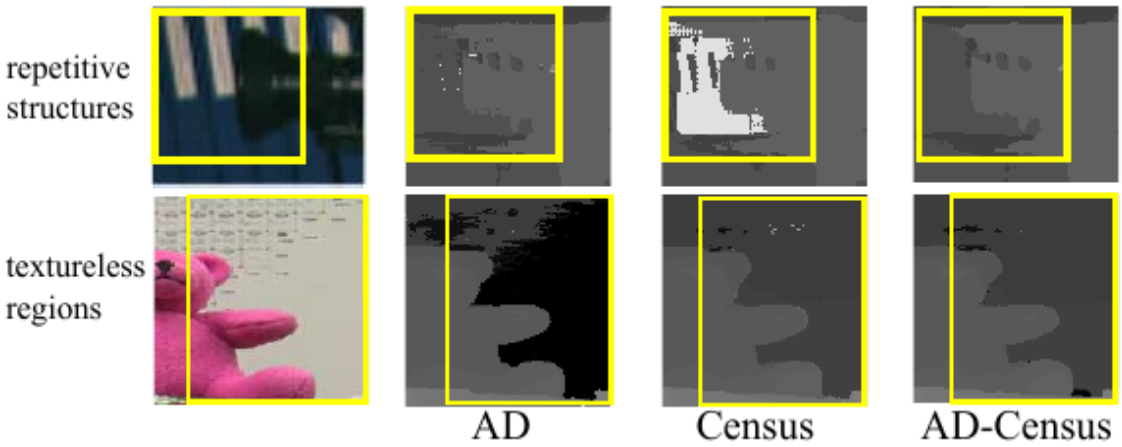


Abbildung 5: Vergrößerungen aus dem Tsukuba- und dem Teddy-Bildpaar

4.2.2. Kostenaggregation bzw. Rahmenalgorithmus

Wie bereits ausführlich besprochen, stellt die in der Themenstellung dieser Arbeit geforderte Schnelligkeit des Algorithmus eine hohe Hürde dar. Bei den lokalen Algorithmen ist die Kostenaggregation meistens der rechenintensivste Teilschritt[93], daher kommt diesem Teil in Bezug auf die Gesamtgeschwindigkeit des Algorithmus eine besondere Bedeutung zu[94]. Da

92 Die Grafik ist entnommen aus: [Mei2011], S. 2.

93 Im ADCensus-Artikel werden für die Rechenzeit der vier Teilschritte die folgenden Prozentzahlen angegeben: Kosteninitialisierung 1 %, Kostenaggregation 70 %, Scanline-Optimierung 28 %, Disparitätsverfeinerung 1 %. Diese Prozentzahlen beziehen sich zwar auf die mithilfe der GPU durchgeführte Berechnung, geben aber dennoch einen Eindruck von der dominierenden Bedeutung der Kostenaggregation für die gesamte Rechenzeit.

94 Ich bin mir durchaus dessen bewusst, dass die Gesamtzeit eines Algorithmus nur sehr begrenzt Rückschlüsse auf die Rechenzeit der Kostenaggregation zulässt. Da in den meisten Artikeln jedoch – wenn überhaupt Angaben zur Rechenzeit gemacht werden – keine Aufschlüsselung bezüglich der Teilschritte veröffentlicht wird, gehe ich davon aus, dass die Gesamtrechenzeit angesichts der typischen Dominanz der Kosten-

die Aufgabenstellung nicht vorsieht, diese Schnelligkeit durch die Berechnung mittels des Grafikprozessors zu erreichen, lag es nahe, bei der Auswahl einer grundlegenden Vorgangsweise eine Vorentscheidung dahingehend zu treffen, inwieweit bestimmte Methoden in der Lage sein könnten, die geforderte Schnelligkeit zu erreichen. Im Folgenden wird die prinzipielle Funktionsweise dieser ausreichend schnellen Algorithmen in Kürze vorgestellt, um einschätzen zu können, wie gut die jeweiligen Ansätze den anderen in der Themenstellung dieser Arbeit genannten Anforderungen entsprechen könnten. Da bei vielen schnell arbeitenden Algorithmen die Kostenaggregation an zentraler Stelle steht, wird die Gesamtperformance eines Algorithmus (und damit die Frage, ob sich dieser Algorithmus als Rahmenkonzept eignet) in diesem Abschnitt untersucht, und zwar auch dann, wenn ein Algorithmus nicht dem Schema Kostenberechnung – Kostenaggregation – Disparitätsauswahl – Disparitätsverfeinerung folgt[95].

Als Ausgangspunkt dieser Untersuchung eignet sich die im Internet veröffentlichte Tabelle der Middlebury-Evaluation[96]. Da in der Middlebury-Evaluation zu den Stereoalgorithmen (anders als in der entsprechenden Evaluation der „Multi-View Stereo"-Algorithmen[97]) kein Geschwindigkeitsvergleich vorgesehen ist, konnten in den Vergleich lediglich jene Ansätze einbezogen werden, deren dazugehörige Fachartikel bzw. „Conference papers" in irgendeiner Weise verfügbar waren: entweder, weil sie direkt auf der Middlebury-Evaluationsseite verlinkt sind, oder weil sie mithilfe der Literaturangabe über die Universitätsbibliothek der Fernuniversität in Hagen ohne Zusatzkosten[98] besorgt werden konnten, oder weil sie mittels einer der Internet-Recherche auffindbar waren[99]. Algorithmen, deren durchschnittliche Prozentzahl der Fehlerpixel 8 % übersteigt, wurden nicht berücksichtigt, weil man mit gutem Grund da-

---

aggregation bei den lokalen Algorithmen einen guten Näherungswert für den Rechenaufwand der Kostenaggregation liefert.

95 Während das semi-globale Matching, obwohl man es am ehesten zu den Algorithmen rechnen kann, die auf dynamische Programmierung aufbauen, viele Ähnlichkeiten zu lokalen Algorithmen aufweist, lässt sich der in diesem Abschnitt kurz beschriebene SegTreeDP-Algorithmus nur schwer in das übliche Vier-Stufen-Schema integrieren. Da in diesem Abschnitt jedoch der Aspekt der Schnelligkeit besonders berücksichtigt wird, wird auch SegTreeDP hier diskutiert.

96 Vgl. http://vision.middlebury.edu/stereo/eval/.

97 Vgl. http://vision.middlebury.edu/mview/eval/.

98 Ausgenommen sind natürlich die auch sonst üblichen Kosten wie Internet- und Postgebühren. Viele der Artikel könnten zwar auch einzeln käuflich erworben, allerdings werden dafür oft ca. 30 US$ verlangt, was bei der Fülle der Artikel nur schwer finanzierbar gewesen wäre und angesichts der großen Fülle an frei verfügbarer Literatur auch nicht notwendig erscheint.

99 Einige Artikel konnten beispielsweise über einen Link auf der Publikationsliste eines Autors bzw. einer Autorin aufgefunden werden. Da die Recherche im Spätwinter 2012 stattgefunden hat, ist es denkbar, dass inzwischen einige der jüngeren Artikel verfügbar gemacht wurden; da aber eine entsprechende Berücksichtigung im Rahmen dieser Arbeit ohnehin nicht mehr möglich wäre, wurde auf eine erneute Recherche verzichtet.

von ausgehen kann, dass diese Verfahren im Vergleich zu den anderen deutliche Qualitätsmängel aufweisen[100].

Unsere Abschätzung der geforderten Schnelligkeit in Abschnitt 3.2.3 hat ergeben, dass ein Algorithmus, der der in der Themenstellung geforderten Schnelligkeit entspricht, beim Teddy-Bildpaar der Middlebury-Datensätze vermutlich eine Laufzeit im einstelligen Sekundenbereich aufweisen müsste. Die im Anhang dieser Arbeit präsentierte Tabelle zeigt, dass nur wenige Algorithmen dieser Anforderung entsprechen.

Bei den Algorithmen, deren angegebene Geschwindigkeit in Aussicht stellt, dass der Algorithmus auch für die Zwecke dieser Arbeit schnell genug sein könnte (in der tabellarischen Übersicht grün unterlegt), ergibt sich folgendes Bild:

„Reliable Disparity Propagation“ (RDP) bringt zwar in der Middlebury-Evaluation sehr gute Ergebnisse hervor, im dazu gehörigen Fachartikel werden allerdings einige Einschränkungen genannt, die gerade für die in dieser Arbeit maßgeblichen Anforderungen von Bedeutung sind: In diesem Verfahren werden auf jeder Bildpunktzeile Liniensegmente gebildet, sodass die Disparitätswerte besonders sicher geltender „seed pixels“ innerhalb dieser Segmente auf Bildpunkte übertragen werden können, deren berechnete Disparitätswerte als weniger zuverlässig gelten. Das funktioniert bei den unter streng kontrollierten Bedingungen aufgenommenen Middlebury-Bildpaaren sehr gut, könnte aber bei stärkerem Bildrauschen oder Beleuchtungsschwankungen zu wesentlich schlechteren Ergebnissen führen[101]. Außerdem ermöglicht dieses Verfahren keine Berechnung der Disparitätswerte in Subpixel-Genauigkeit, sodass es wenig verwundert, dass das Verfahren in der Middlebury-Evaluationsstudie wesentlich schlechter abschneidet, wenn der Fehlertoleranzwert für die Evaluation herabgesetzt wird[102].

Das mit dem Akronym „InfoPermeable“ abgekürzte Verfahren[103] verspricht sehr gute Ergebnisse in sehr kurzer Rechenzeit. Bei diesem Verfahren wird die Kostenaggregation zweidimensional in waagrechten und senkrechten Pixellinien durchgeführt. Für jeden Pixelübergang in waagrechter und senkrechter Richtung wird ein Durchlässigkeitswert berechnet, der bestimmt, in welchem Ausmaß die bis zu dieser Stelle aggregierten Kosten zu den Kosten

100 Derzeit (Stand September 2012) wird die Tabelle von einem Algorithmus angeführt, der nur knapp vier Prozent Fehlerpixel hervorbringt.

101 Vgl. [Sun2011], S. 5. Insbesondere Salt-and-Pepper-Noise und Photon-Shot-Noise würden wohl bedeutende Probleme für den Algorithmus aufwerfen.

102 In der Standardeinstellung liegt der Fehlertoleranzwert bei 1, d. h. Disparitätswerte werden als korrekt angesehen, wenn sie höchstens um eine Stufe vom Disparitätswert der „Ground Truth“ abweichen. Während RDP bei dieser Standardeinstellung derzeit (Stand: 2012-09-07) an sechster Stelle liegt, rutscht das Verfahren bei einem Fehlertoleranzwert von 0,5 auf die 36. Stelle ab.

103 Vgl. [Çığla2011]; [Çığla2012].

des nächsten Bildpunktes hinzugefügt werden: Je ähnlicher sich die Farbwerte der benachbarten Pixel sind, desto mehr werden die bis dahin aggregierten Kosten auf das nächste Pixel übertragen. Dasselbe Verfahren kann mit nur wenigen Abwandlungen verwendet werden, um die verdeckten Bildregionen mit sinnvollen Pixelwerten zu versehen. Dass dieses eigentlich recht einfache Verfahren in sehr kurzer Zeit zu sehr guten Ergebnissen führt[104], ist durchaus beeindruckend, allerdings könnte sich das Verfahren auch als anfällig für Bildrauschen erweisen, weil die Durchlässigkeitswerte aufgrund der Farbähnlichkeit der benachbarten Pixel berechnet werden[105].

Das als „semi-globales Matching“ (meist abgekürzt als: SGM) bezeichnete Verfahren[106] zeichnet sich ebenfalls durch sehr kurze Laufzeiten aus[107]. Wie gut SGM (im Vergleich mit anderen Verfahren) mit Bildrauschen zurechtkommt, ist vom theoretischen Standpunkt her schwer einzuschätzen, allerdings spricht manches dafür, dass SGM mit einem gewissen Ausmaß an Bildrauschen ganz gut zurechtkommen könnte, denn ein Disparitätswechsel wird bei der Kostenaggregation mit zusätzlichen Kosten sanktioniert, sodass die Wahrscheinlichkeit, dass einzelne Pixel mit falscher Farbintensität nicht gleich zu einem Disparitätswechsel führen, hoch ist[108]. Da das Verfahren auch bzgl. seiner Verwendung in Fahrassistenzsystemen intensiv beforscht wird[109], kann als Hinweis gewertet werden, dass es bezüglich der in dieser Arbeit geforderten Anforderungen gut abschneiden könnte, weil Fahrassistenzsysteme schnell sein, mit schwierigen Beleuchtungsverhältnissen umgehen und mit einem gewissen Maß an Bildrauschen zurechtkommen müssen. Die Tatsache, dass das Verfahren in diversen Evaluationsstudien verwendet wurde (vgl. Abschnitt 4.1) und auch Forscher/innen, die nicht ursprünglich an der Konzeption des Algorithmus beteiligt waren, sich um dessen Weiterent-

104 Die Qualität der Ergebnisse relativiert sich allerdings etwas, wenn man bedenkt, dass das Verfahren vom 21. Rang auf den 73. abrutscht (Stand: 2012-09-07), wenn man den Schwellenwert für Fehlerpixel von 1 auf 0,5 herabsetzt. Allerdings wäre es bei diesem Verfahren – anders als bei RDP – prinzipiell möglich, die Disparitätswerte durch das oft übliche Verfahren mittels einer quadratischen Kurve durch die Nachbardisparitäten in Subpixel-Genauigkeit zu berechnen – was aber offenbar für die Testdaten nicht gemacht wurde.

105 Auf älteren Systemen – wie dem von mir verwendeten – ist außerdem zu berücksichtigen, dass der Speicherbedarf des Verfahrens relativ hoch ist, weil das Verfahren Gleitkommaarithmetik benötigt; das könnte u. U. bei den vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellten Bilddaten zu Speicherplatzproblemen führen.

106 Dieses Verfahren scheint in der Middlebury-Evaluation unter zwei Akronymen auf: „SemiGlob“ bezeichnet das Verfahren in seiner ursprünglichen Variante, vgl. [Hirschmüller2005], „C-SemiGlob“ eine zweite, verbesserte Variante, vgl. [Hirschmüller2006]. Ein zwei Jahre später erschienener Artikel ergänzt das Verfahren v. a. um Vorschläge, wie man das Verfahren auf besonders große Bilder anwenden könnte, und um eine ausführlichere Evaluation, vgl. [Hirschmüller2008].

107 Bei einer Fehlertoleranz von 1 liegt „C-SemiGlob“ an vierzigster Stelle, bei einem Schwellenwert von 0,5 sogar an zehnter Stelle der Middlebury-Evaluation (Stand: 2012-09-07).

108 Um das stichhaltig begründen zu können, müsste erst einmal das Verfahren genau vorgestellt werden – dies soll aber erst im nächsten Kapitel gemacht werden.

109 Vgl. die einschlägigen Artikel, die aus dem .enpeda.-Projekt der Universität Auckland hervorgegangen sind: [Hermann2009], [Hermann2011], [Hermann2012].

wicklung annehmen[110], zeigt, dass dem semi-globalen Matching in der Fachwelt ein hoher Wert beigemessen wird.

Der mit dem Akronym „RandomVote“ abgekürzte Algorithmus[111] verwendet unterschiedliche Farbsegmentierungen des Basisbilds, um für jedes Pixel mehrere Disparitätshypothesen aufzustellen, von denen dann mittels einer sog. „Dichtheitsfunktion“ die wahrscheinlichste ausgewählt wird. Da sich die Farbwerte einzelner Pixel bei unterschiedlicher Beleuchtung zum Teil sehr stark verändern, ist zu vermuten, dass dieser Algorithmus bei größeren radiometrischen Differenzen Schwierigkeiten bekommt. Ob diese vermutliche Schwäche andererseits durch bessere Ergebnisse in texturarmen Regionen ausgeglichen werden könnte, ist schwer einzuschätzen.

Das mit dem Akronym „SegTreeDP“[112] abgekürzte Verfahren arbeitet mit dynamischer Programmierung, allerdings werden – anders als bei den allermeisten Verfahren – eigentlich nicht einzelne Bildpunkte einander zugeordnet, sondern einzelne Liniensegmente. Der Algorithmus arbeitet extrem schnell (60–70 ms für ein Bildpaar mit 320 x 240 Bildpunkten), allerdings finden sich im Artikel keinerlei Hinweise darauf, wie sich der Algorithmus bei Beleuchtungsdifferenzen oder Bildrauschen bewährt. Aufgrund der eigenwilligen Arbeitsweise wäre eine Kombination dieses Algorithmus mit Elementen anderer Algorithmen eher schwierig.

4.2.3. Disparitätsberechnung

Die Disparitätsauswahl bei lokalen Stereoalgorithmen mittels des simplen Prinzips „Winner takes all“ (oft abgekürzt als „WTA“) erlaubt vorerst nur eine ganzzahlige Disparitätsbestimmung. Aufgrund der in der digitalen Fotografie nötigen Pixelrasterung würde der theoretisch korrekte Disparitätswert aber oft zwischen zwei Ganzzahlen liegen.

In manchen Algorithmen wird daher versucht, Subpixel-Genauigkeit zu erreichen. Dabei werden meistens die Kosten der Disparitätsnachbarn der ausgewählten Disparität verwendet, um eine Kurve durch diese Kostenpunkte zu interpolieren. Eine oft angewendete Variante mit

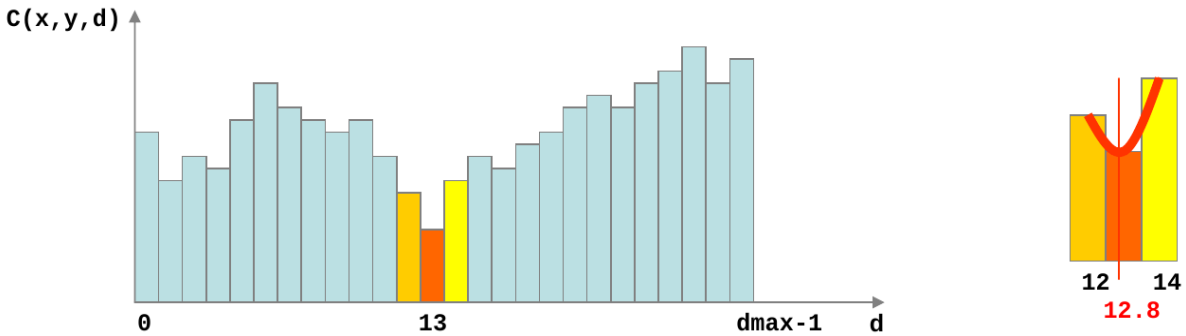


Abbildung 6: Subpixel-Interpolation mittels Parabel

Abkürzung LSTDP verwendet, vgl. [Deng2006].

relativ geringen Berechnungskosten nutzt zu diesem Zweck einfach eine Parabel (vgl. Abb. 6)[113].

Während die Subpixel-Interpolation relativ wenig zusätzliche Rechenzeit erfordert, sind andere Methoden (wie z. B. die Berechnung der Disparitätswerte für höher aufgelöste Bilder, die mittels bikubischer Bildpunktinterpolation erzeugt werden) wesentlich zeitaufwändiger.

4.2.4. Disparitätsverfeinerung

Ein sehr häufig angewendetes Verfahren, um falsch zugeordnete Pixel aufzuspüren, ist der Left-right-Consistency Check. Dabei wird meist eine zweite Disparitätskarte für das Referenzbild berechnet, um dann zu überprüfen, ob die im linken und im rechten Bild jeweils zugeordneten Bildpunkte zusammen passen. Wenn sich zwei Bildpunkte gegenseitig referenzieren, gilt die Disparität als korrekt, ansonsten als inkorrekt. Um kleine Abweichungen, die ja auch darauf zurückzuführen sein können, dass der „korrekte“ Disparitätswert zwischen zwei Ganzzahlwerten liegt, zu tolerieren, bleiben üblicherweise kleinere Fehler unberücksichtigt. Oft wird dabei ein Unterschied von maximal einer Disparitätsstufe toleriert.

Allerdings ist es nicht unbedingt nötig, für die zweite Disparitätskarte eine neuerliche Kostenaggregation durchzuführen; es ist auch möglich, sie aus demselben „Disparity Space Image“ zu berechnen. Während die Disparitätskarte des Basisbilds mittels der Formel

$$D_B(p) = argmin_d S(p,d)$$

berechnet wird, kommt bei der Disparitätskarte des Referenzbilds die Formel

$$D_R(p) = argmin_d S(p+d,-d)$$

zur Anwendung[114]. Eine eigenständige Berechnung der zweiten Disparitätskarte mithilfe eines zweiten „Disparity Space Image“ erhöht zwar die Treffsicherheit, aber auch den Zeit- und Speicherbedarf erheblich.

Sofern eine „dichte“ Disparitätskarte gewünscht wird (was für einen Eintrag in der Middlebury-Evaluationstabelle obligatorisch ist), werden daraufhin die als ungültig markierten Disparitätswerte von den bereits berechneten abgeleitet, wobei sich die Methoden darin unterscheiden, wie sie die dafür in Frage kommenden als korrekt markierten Disparitätswerte identifizieren. In einer Reihe von Algorithmen wird dabei zwischen verschiedenen Typen ungültiger Pixel unterschieden: Vermutlich verdeckte Bildpunkte und Bildpunkte in der Nähe von

113 Die Grafik ist entnommen aus: [Mattoccia2012], Folie Nr. 164.

114 Die hier wiedergegebene Formel entspricht im Wesentlichen der Formel in [Ernst2008], S. 232, allerdings muss das Vorzeichen des Disparitätswerts abgeändert werden, weil die vorliegende Arbeit zwar von rektifizierten Bildern, aber nicht von einer Off-Axis-Konstellation ausgeht, sodass in derselben Disparitätskarte positive und negative Disparitätswerte vorkommen und daher nicht die Absolutwerte verwendet werden können.

Diskontinuitäten werden dabei durch den niedrigsten in der „Nähe“ liegenden korrekten Disparitätswert ersetzt (wobei die „Nähe“ meist aufgrund einer Kombination aus Distanz und Farbähnlichkeit bestimmt wird), weil verdeckte Pixel im Normalfall zum Hintergrund des Bildes gehören; die Werte aller anderen ungültigen Pixel werden mithilfe der nächstliegenden korrekten Disparitätswerte interpoliert.

Zu beachten ist freilich, dass die für die verdeckten Bildbereiche geschilderte Strategie in dieser Weise nur bei einer Off-Axis-Konstellation funktioniert, weil nur bei dieser im „Disparity Space Image“ ausschließlich Absolutwerte der Disparitäten verwendet werden können, sodass der Hintergrund in jedem Fall niedrigere Disparitätswerte besitzt. Wenn allerdings – wie in dieser Arbeit – davon ausgegangen werden muss, dass sowohl positive als auch negative Disparitätswerte vorkommen, können die Disparitätswerte nicht mehr generell mittels der Absolutwerte dargestellt werden. In diesem Fall besitzen die Hintergrundpixel im linken Disparitätsbild hohe Werte (im Spezialfall einer Off-Axis-Konstellation wäre der *höchste* theoretisch mögliche Disparitätswert der Wert null), jene im rechten hingegen niedrige Werte (dieses Mal wäre bei einer Off-Axis-Konstellation der *niedrigste* mögliche Wert null).

Allerdings ist eine solche Abschätzung bzw. Interpolation der als unkorrekt markierten Bildpunkte nur nötig, wenn man eine dichte Disparitätskarte benötigt, also eine, in der jedem Pixel ein Disparitätswert zugeordnet wird[115].

Der im Rahmen dieser Masterarbeit entwickelte Algorithmus baut auf den Erkenntnissen auf, die die Lektüre der in diesem Kapitel vorgestellten Artikel (und noch einer ganzen Reihe weiterer, die hier aus Platzgründen nicht berücksichtigt werden konnten) gebracht hat. Nach der Definition der Aufgabenstellung (Kap. 1), der Einführung in das Forschungsfeld (Kap. 2), der Vorstellung des zur Verfügung stehenden Datenmaterials (Kap. 3) und der Diskussion wichtiger bereits zu dieser Thematik geleisteter Vorarbeiten (Kap. 4) ist nun der Weg dafür geebnet, den im Rahmen dieser Arbeit entwickelten Algorithmus vorzustellen, der den gestellten Anforderungen zu entsprechen versucht.

115 Wie mir Herr Häming vom Lehrgebiet Mensch-Computer-Interaktion auf meine Nachfrage hin mitteilte, ist eine dichte Disparitätskartenberechnung im Rahmen dieser Arbeit nicht erforderlich.

5. Algorithmus

Die meisten Artikel, die zum Stereo Matching veröffentlicht wurden, kümmern sich nicht um die Frage, wie der Disparitätssuchbereich zu bestimmen ist. Vielmehr führen sie ihre Berechnungen für eine fixe, im Vorhinein festgelegte Anzahl an Disparitäten durch, die die im Vorhinein bekannten minimalen und maximalen Disparitätswerte umfasst.

Eine solche Herangehensweise ist für die am Lehrgebiet Mensch-Computer-Interaktion entwickelte Demo-Anwendung zur Freihanderfassung von 3D-Objekten jedoch nicht praktikabel. Vielmehr ist – wie an den entsprechenden für die vorliegende Arbeit zur Verfügung gestellten Datensätzen ersichtlich – damit zu rechnen, dass die für das Stereo Matching zur Verfügung stehenden Bilder ganz unterschiedliche Disparitätswerte aufweisen. Diese Disparitätswerte sind einerseits nicht von vornherein bekannt und können andererseits eine sehr große Bandbreite umfassen, beispielsweise, wenn die beiden Bilder aus ungünstigen Winkeln oder aus sehr großer Nähe aufgenommen wurden.

Obwohl diese Anforderung nicht ausdrücklich in der Themenstellung der Arbeit genannt war, setzt der hier vorgeschlagene und parallel zu dieser Arbeit auch beispielhaft implementierte Algorithmus daher bereits einen Schritt früher als die meisten in der Fachliteratur beschriebenen Verfahren an: bei der Abschätzung und Einschränkung des Disparitätssuchraums.

5.1. Abschätzung des Disparitätssuchraums

Den Disparitätssuchraum möglichst passend abzuschätzen, ist von hoher Relevanz für den Speicher- und Zeitbedarf des Stereo Matching: Es leuchtet unmittelbar ein, dass Speicher- und Platzbedarf entsprechend steigen, wenn die Kosten für eine größere Anzahl an Disparitätsstufen berechnet und aggregiert werden müssen[116].

Eine Möglichkeit, den Suchraum einzuschränken, wäre, die beiden Bilder auf eine kleinere Auflösung herunterzurechnen, die Berechnung für alle möglichen Disparitätsstufen (oder für einen bestimmten Teil davon) dieser niedrigeren Auflösungsstufe durchzuführen und dann den Disparitätsbereich, den die gefundenen Disparitätswerte einnehmen, auf die höhere Auflösung

116 Bei einem Vergleichstest benötigte eine Variante des hier vorgestellten Algorithmus beim Middlebury-Datensatz „Teddy“ 5,76 Sekunden, wenn die üblichen 60 Disparitätsstufen berechnet wurden, aber mehr als 62 Sekunden, wenn die Berechnung für die maximal möglichen 899 Disparitätsstufen (von -449 bis 449) durchgeführt wurde.

hochzurechnen, um dort die Berechnung nur mehr für den auf diese Weise eingeschränkten Disparitätsbereich, dieses Mal allerdings in der vollen Auflösung, durchzuführen.

Ein solches Down- und Upscaling wird in der Fachliteratur mehrfach beschrieben, allerdings wird dort meistens „nur“ eine Verringerung des Speicherbedarfs und/oder der Berechnungskomplexität angestrebt, während es in vielen Fällen nicht darum geht, die minimale und maximale Disparitätsstufe festzulegen, weil diese meist als bekannt angenommen wird[117].

Der hier vorgeschlagene Algorithmus beschreitet allerdings einen anderen Weg. Ausgangspunkt ist die Beobachtung, dass die Themenstellung dieser Arbeit von rektifizierten Bildpaaren ausgeht[118]. Wie bereits beschrieben (vgl. Abschnitt 2.1), sind für die Rektifizierung nur 7–8 Featurepaare nötig, allerdings werden beim Feature Matching üblicherweise sehr viel mehr Featurepaare gefunden. Die Anzahl der gefundenen und einander zugeordneten Features variiert sehr stark, liegt aber – je nach fotografierter Szene – oft im dreistelligen Bereich.

Im hier vorgestellten Verfahren werden diese Features verwendet, um den Disparitätssuchraum des Bildes abzuschätzen. Ziel dieser Abschätzung ist es, nicht nur die minimale und maximale Disparitätsstufe des Bildes zu ermitteln, sondern den Suchraum auch lokal einzuschränken. Das sei an einem Beispiel erläutert: Der Datensatz „Raum Poltergeist“ des Lehrgebiets Mensch-Computer-Interaktion (vgl. Abb. 18, S. 71) weist Disparitätwerte von –55,8 bis +117,4 auf. Selbst wenn man den dementsprechenden Disparitätssuchraum (von –56 bis +119[119]) exakt abschätzen könnte, müssten 175 Disparitätswerte berücksichtigt werden. Allerdings weist das Bild diese große Disparitätsbandbreite gar nicht im gesamten Bild auf: Wie in der „Ground Truth“ dieses Datensatzes an der Helligkeit der Bildpunkte leicht zu erkennen ist, überwiegen in der linken unteren Bildregion die niedrigen Disparitätswerte, während der Rest des Bildes von hohen Disparitätswerten dominiert wird. Sowohl für den Speicherbedarf

117 Hirschmüller verwendet das Down- und Upscaling lediglich für die effektive Berechnung der Kostenfunktion HMI, nicht jedoch für die Kostenaggregation, vgl. [Hirschmüller2008]. Anderen Autor/inn/en geht es um die Verringerung des Speicherbedarfs und eine effiziente Implementierung mithilfe des Grafikprozessors, vgl. [Won2011].

118 Eine alternative Vorgangsweise wäre, die unrektifizierten Bilder zum Matching zu verwenden, was allerdings nur wenige Algorithmen erlauben. Das semi-globale Matching kann auch mit unrektifizierten Bildern durchgeführt werden, allerdings ist dafür die Kenntnis der Epipolargeometrie des Bildes erforderlich, sodass auch in diesem Fall ein Feature Matching nötig wäre. Eine Verallgemeinerung des hier vorgestellten Verfahrens auf unrektifizierte Bilder könnte daher möglich sein, ist aber nicht Gegenstand der vorliegenden Arbeit.

119 Im Rahmen dieser Arbeit wird – wie auch in der parallel erstellten Referenzimplementierung des Algorithmus – jeweils der untere Wert eines Bereichs inklusiv, der obere Wert hingegen exklusiv angegeben, d. h. bei einem Disparitätssuchraum von –56 bis +119 ist die minimale Stufe, für die eine Berechnung durchgeführt wird, 115 und die maximale berechnete Stufe +118. Das geschieht einerseits deshalb, weil diese Vorgangsweise jener in der freien Programmbibliothek OpenCV entspricht, die in der Referenzimplementierung intensiv genutzt wird. Nebenbei hat diese Verwendungsweise den Vorteil, dass sich die Anzahl der zu berechnenden Disparitätsstufen durch einfache Subtraktion der unteren Grenze von der oberen Grenze ermitteln lässt, während sonst immer die Nulldisparität extra dazu gerechnet werden müsste.

als auch für den Rechenaufwand wäre sehr viel gewonnen, wenn man nicht für das gesamte Bild alle 175 Stufen berechnen müsste, sondern in jeder Bildregion nur jene Stufen, die in dieser Bildregion auch vorkommen. Allerdings birgt diese Vorgangsweise auch ein großes Risiko: Wird der Disparitätssuchraum nämlich zu stark eingeschränkt, sodass der korrekte Disparitätswert nicht mehr innerhalb des festgelegten Suchraums liegt, dann wird eine Ermittlung der korrekten Disparität verunmöglicht. Im Zweifelsfall ist es daher wohl sinnvoller, den Suchraum etwas zu groß festzulegen, als ihn zu sehr einzuschränken.

Das dazu hier vorgeschlagene Verfahren geht folgendermaßen vor: In einem ersten Schritt werden zu jedem Feature fünf Nachbarfeatures gesucht. Als Kriterium für die „Nähe“ der Nachbarschaft wird die maximale Farbdistanz zwischen je zwei Pixeln, die auf einer Linie zwischen den beiden Features liegen, verwendet, wobei die Farbdistanz zweier Punkte, die auf der Linie zwischen den Features f, f' liegen, bestimmt wird als:

$$D(p,q)=max(|r_p-r_q|,|g_p-g_q|,|b_p-b_q|)\,,$$

wobei es sich bei r_p, r_q, g_p, g_q, b_p, b_q um die Farbwerte der jeweiligen Bildpunkte handelt. Diese Vorgangsweise soll sicherstellen, dass Punkte, die zur selben Fläche gehören, gegenüber Punkten, die an verschiedenen Seiten von Diskontinuitäten liegen, bevorzugt werden.

Anschließend werden die initialen Disparitätswerte für den Suchraum festgelegt. Dazu wird je eine der Bildgröße entsprechende Matrix für die untere und die obere Grenze des Suchraums angelegt. An den Positionen, die den Features entsprechen, werden die entsprechenden Disparitäten eingetragen, die Disparitätswerte zwischen je zwei benachbarten Features werden linear interpoliert. Wenn also ein Featurepaar den Disparitätswert –30 aufweist wird für die Position dieses Features der Disparitätssuchbereich [–30, –29] festgelegt. Hat nun ein (nach dem Kriterium der Farbdistanz, s. o.) benachbartes Feature den Disparitätssuchbereich [+15, +16], dann bekommen alle Bildpunkte, die auf einer Linie zwischen den benachbarten Features liegen, einen jeweiligen Disparitätssuchbereiche [d, d+1] zugeordnet, wobei gilt: –30 <=d < +15. Dass als Kriterium für die „Nachbarschaft“ zweier Pixel die Farbdistanz (und nicht die räumliche Distanz) verwendet wird, hat den Vorteil, dass auch die Disparitätswerte zwischen räumlich weiter voneinander entfernten Pixeln verbunden werden und daher eine größere Anzahl an initialen Disparitätsvermutungen zustande kommt.

Da Features – abhängig vom verwendeten Algorithmus – häufig an Ecken ermittelt werden, werden sie zwar oft auch an Vordergrundobjekten gefunden, die Wahrscheinlichkeit, dass eines der Featurepaare eine Position im vorderen Bildbereich erfasst, ist daher relativ hoch; zugleich ist aber die Wahrscheinlichkeit gering, dass der Bildpunkt, der zum vordersten Punkt

der abgebildeten Szene gehört (also der Bildpunkt mit der höchsten oder niedrigsten Disparitätsstufe des Bildes), gleichzeitig als Feature erfasst wurde. Deshalb wird im nächsten Schritt die Anzahl der Disparitätsstufen erhöht, indem die Untergrenze der bereits abgeschätzten Suchbereiche um einen fixen Wert erniedrigt wird, die Obergrenze um denselben Wert erhöht.

Im nächsten Schritt werden die auf die beschriebene Weise ermittelten Disparitätssuchbereiche auf benachbarte Pixel übertragen. Dabei ist es wünschenswert, dass der Disparitätssuchbereich so lange von einem Pixel auf ein jeweils benachbartes Pixel übertragen wird, solange die beiden Pixel ungefähr die gleiche Disparität aufweisen. Da lokale Algorithmen darauf setzen, bei der Kostenaggregation möglichst nur die Kosten jener benachbarten Pixel aufzuaddieren, die derselben Disparitätsstufe angehören, liegt es nahe, bei einem dieser Algorithmen Anleihen zu nehmen.

Der in der Middlebury-Evaluation mit dem Akronym „ADCensus“ abgekürzte Stereoalgorithmus aggregiert die Kosten mittels eines Verfahrens, das sich „Cross-based Aggregation“ nennt[120]. Die Idee dieses Verfahrens ist, bei der Kostenaggregation für jeden Bildpunkt eine flexible Unterstützungsregion heranzuziehen, die im Umfeld des Pixels nur jene Nachbarpixel umfasst, die derselben Disparitätsstufe angehören.

Dazu wird zu jedem Pixel aufgrund bestimmter Kriterien ein linker, rechter, oberer und unterer Seitenarm festgelegt (vgl. Abb. 7). Demnach wird ein Pixel p_l dann zum linken Seitenarm des Pixels p gerechnet, solange gilt:

1. $D_c(p_l, p) < \tau_1$ *und* $D_c(p_l, p_l+(1,0)) < \tau_1$
2. $D_s(p_l, p) < L_1$
3. $D_c(p_l, p) < \tau_2$, *wenn* $L_2 < D_s(p_l, p) < L_1$.

Dabei bezeichnet $D_c(p_l, p)$ die Farbdistanz der beiden Pixel p und p_l, die auf dieselbe Weise wie bei den Features berechnet wird, also als Maximalwert der Intensitätsdifferenz der drei Farbkanäle; $D_s(p_l, p)$ bezeichnet die räumliche Distanz der beiden Pixel; $p_l+(1,0)$ steht für das Vorgängerpixel von p_l. Die erste Regel soll dafür sorgen, dass der Unterstützungsarm nicht über Kanten im Bild hinwegläuft. Die maximale Länge des Unterstützungsarms wird durch die zweite Regel begrenzt. Schließlich legt die dritte Regel für Pixel in größerer Distanz zum Punkt p strengere Regeln an als für die näher liegenden Pixel, sodass sich der Arm einerseits in texturlosen Bildregionen weit ausbreiten kann, ohne dass das Risiko steigt, dass der Arm über Kanten hinweg läuft.

120 Dieses Verfahren wurde ursprünglich von Zhang et al. entwickelt und von Mei et al. im Zusammenhang mit dem ADCensus-Algorithmus weiterentwickelt, vgl. [Mei2011], S. 3.

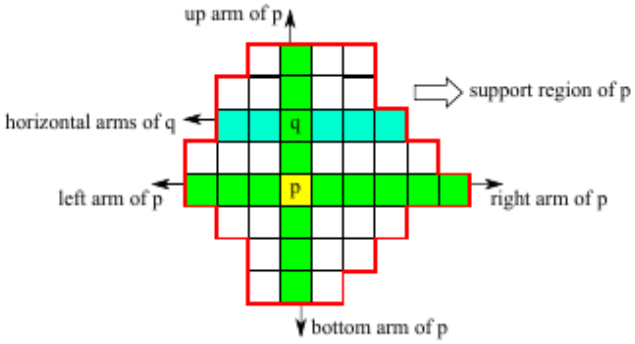
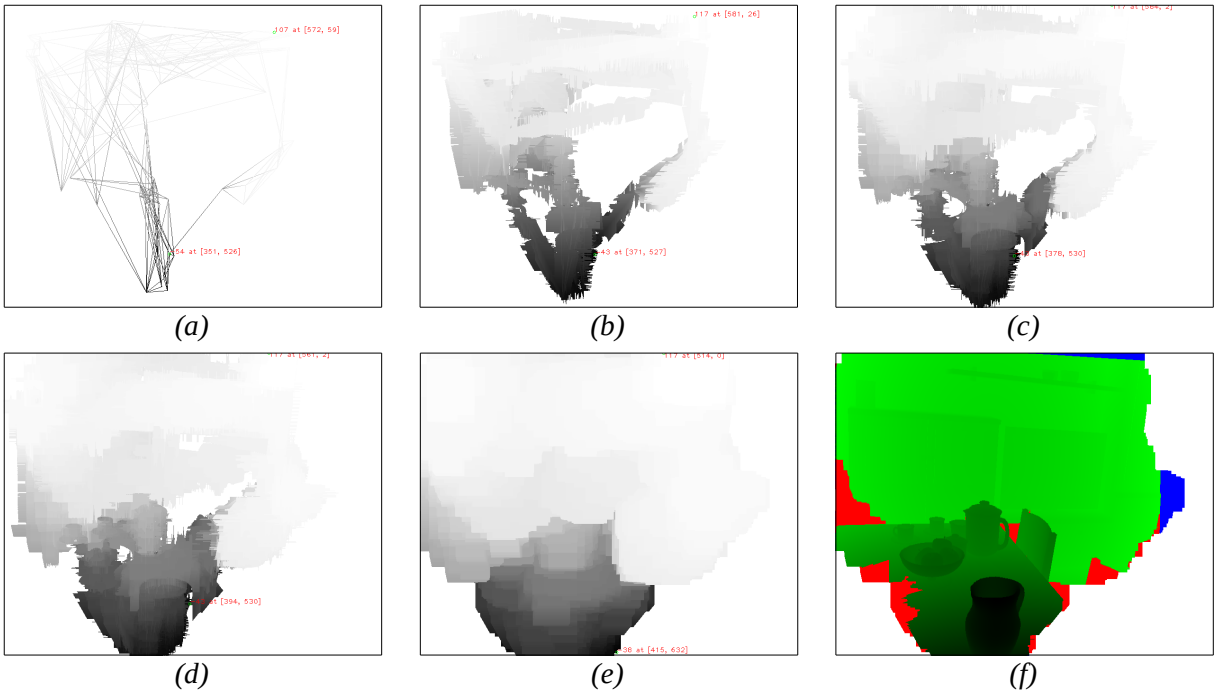


Abbildung 7: Cross-based Aggregation: Die Unterstützungsregion des Pixels p besteht aus den benachbarten Pixeln q und deren rechtwinklig zur Verbindungslinie p–q liegenden Seitenarmen.

Während die „Cross-based Aggregation“ die Unterstützungsregion nutzt, um die Kosten horizontal und vertikal aufzuaddieren, verwendet der hier vorgeschlagene Algorithmus die Unterstützungsregion, um einen bereits bekannten Disparitätssuchraum auf umliegende Pixel zu verteilen. Hat also das Pixel p den Disparitätssuchraum [–10, +11] und q den Suchraum [+5, +26], dann wird der Suchraum von q auf [–10, +26] erweitert. Diese Weiterverteilung der Suchraumgrenzen findet zuerst entlang der horizontalen Unterstützungsarme statt; die bereits neu bestimmten Suchräume werden dann entlang der vertikalen Arme weiterverteilt.



(a) *(b)* *(c)*

(d) *(e)* *(f)*

Abbildung 8: Schrittweise Einschätzung des Disparitätssuchraums anhand des Bildpaars „Raum Poltergeist“. Die Bilder (a)–(e) zeigen die obere Grenze des bis zum jeweiligen Schritt festgelegten Suchraums: (a) Initialisierung mittels Interpolation zwischen benachbarten Features;
(b)–(d) Verteilung der Suchräume mittels der Unterstützungsregionen der „Cross-based Aggregation“; (e) Verteilung der Suchräume um eine fixe Distanz in allen Hauptrichtungen. Das letzte Bild (f) zeigt, inwieweit die auf diese Weise festgelegten Disparitätssuchräume auch die in der „Ground Truth“ auffindbare korrekte Disparität umfassen: In grünen Bildregionen befindet sich der korrekte Disparitätwert innerhalb des festgelegten Suchraums; rot sind Bildregionen eingefärbt, die die korrekte Disparität nicht beinhalten; blaue Regionen kennzeichnen verdeckte Bildbereiche.

Um bereits in der ersten Berechnungsrunde möglichst viele Pixel berechnen zu können, wird dieser Schritt drei Mal durchgeführt.

Auf diese Weise entsteht – abhängig von der Anzahl der zur Verfügung stehenden Features sowie von den Eigenschaften der Szene und der Fotografien – eine zusammenhängende Teilmenge an Pixeln, für die eine erste Einschätzung bzgl. des Disparitätssuchraums getroffen wurde. Diese Teilmenge des Gesamtbildes weist allerdings an den Rändern viele Fransen auf, was sich für die nachfolgende Kostenaggregation nachteilig auswirkt. Deshalb werden in einem abschließenden Schritt die Disparitätssuchräume noch einmal auf ihre Nachbarpixel verteilt, dieses Mal jedoch nicht nach dem Prinzip der „Cross-based Aggregation“, sondern jeweils um einen fixen Pixelwert in alle vier Hauptrichtungen, wiederum zuerst horizontal und dann vertikal. Abbildung 8 zeigt beispielhaft, dass auf diese Weise eine ganz gute Einschätzung des Disparitätssuchraums zustande kommen kann, allerdings auch, dass die korrekte Einschätzung vor allem an den Rändern (links unten und rechts unten) und bei Diskontinuitäten (hinter dem Griff der Teekanne) schwierig ist.

Der weitere Verlauf sieht nun vor, dass für jene Pixel, für die bereits ein Suchraum festgelegt worden ist, eine Disparitätsberechnung stattfindet (vgl. 5.2–5.5). Die dabei gefundenen Disparitäten werden verwendet, um die nächste Runde der Suchraumbestimmung damit zu initialisieren (vgl. 5.4).

5.2. Kostenfunktion

Da die Fähigkeit des zu entwickelnden Algorithmus, mit schwierigen Beleuchtungssituationen umzugehen, zu den ausdrücklich genannten Anforderungen dieser Abschlussarbeit gehört, liegt es nahe, die als „Census“ bekannte Kostenfunktion zu verwenden, weil sich in Vergleichstests relativ klar gezeigt hat, dass Census besonders gut mit radiometrischen Differenzen zurechtkommt.

Die Berechnung geht dabei folgendermaßen vonstatten: In einem ersten Schritt werden für alle Pixel des Basis- und des Referenzbilds die Census-Transformationen berechnet. Dabei wird das Pixel mit jedem anderen Pixel innerhalb eines bestimmten fixen Fensters verglichen und eine Bitfolge nach folgendem Muster erstellt: Wenn die Intensität des Referenzpixels geringer ist als jene des Basispixels, wird das entsprechende Bit auf 1 gesetzt, ansonsten auf 0. Bei einer (in vielen Algorithmen verwendeten) Fenstergröße von 9 x 7 Pixeln ergibt sich so eine 62 Bits lange Folge, weil auf einen Vergleich des Basispixels mit sich selbst verzichtet werden kann.

Die Kosten, um einen Bildpunkt des Basisbilds einem Pixel des Referenzbilds zuzuordnen, werden dann als Hamming-Distanz der jeweiligen Census-Transformationen berechnet, d. h. es wird abgezählt, wie viele Bits der beiden Bitfolgen unterschiedlich sind. Da in den Census-Transformationen Strukturinformationen der Pixel festgehalten werden, erfasst die Hamming-Distanz vor allem, wie weit die Farb*struktur*, die die beiden Pixel umgibt, übereinstimmt. Dass diese Kostenfunktion trotz ihrer relativ geringen Differenzierung (bei einem 9 x 7 großen Fenster gibt es nur 62 voneinander verschiedene Kostenwerte, bei einem kleineren Fenster noch weniger) vor allem bei radiometrischen Differenzen gut abschneidet, liegt daran, dass sich bei unterschiedlichem Lichteinfall die absoluten Werte der Bildpixel wesentlich häufiger ändern als die Relation der Intensitätswerte benachbarter Pixel zueinander.

Wie bereits im letzten Kapitel beschrieben, wird die Census-Kostenfunktion in neueren Veröffentlichungen oft mit einer anderen, insbesondere mit AD verbunden. Diese Verbindung trägt zwar offenbar zu sehr guten Ergebnissen in der Middlebury-Evaluation bei, allerdings ist mir keine Studie bekannt, die überprüft hätte, wie gut sich diese kombinierte Kostenfunktion bei radiometrischen Differenzen und Bildrauschen bewährt. Es lag daher nahe, bei der Erarbeitung des vorliegenden Algorithmus als Alternative zu Census eine Variante ADCensus zu implementieren und diese mit der „reinen“ Census-Funktion zu vergleichen. Wie aus Abbildung 9 ersichtlich, zeichnet der Vergleich ein differenziertes Bild: Während ADCensus bei jenen Bildern, die mit der gleichen Lichtquelle erstellt wurden, tatsächlich bessere Ergebnisse

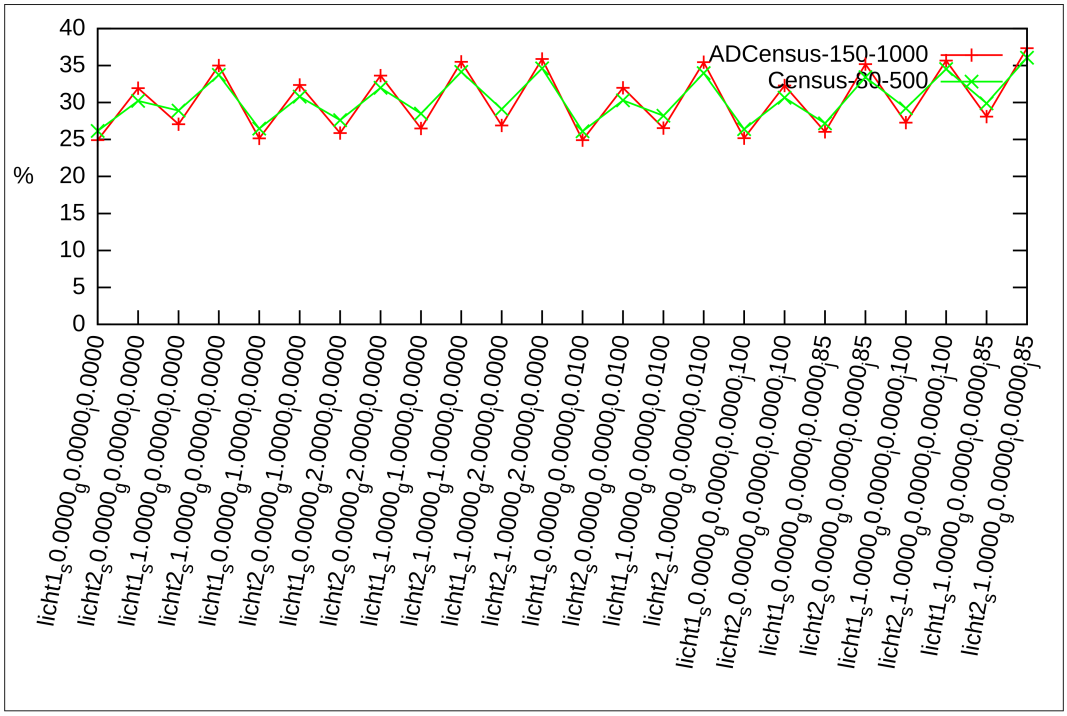


Abbildung 9: Vergleich der Kostenfunktionen Census und ADCensus bzgl. der Anzahl der Fehlerpixel beim Bildpaar „Sokrates Off-Axis“: Bei Bildpaaren mit gleichmäßiger Beleuchtung schneidet ADCensus besser ab, bei wechselnden Lichtverhältnissen hingegen Census.

hervorbringt, kehrt sich das Bild bei wechselnden Lichtverhältnissen um; in diesem Fall schneidet Census durchwegs besser ab.

Ein ähnliches Bild ergibt sich, wenn man Census mit verschiedenen Census-Varianten vergleicht. Bei der als „Modified Census Transform“ (MCT) bekannten Variante[121] werden die Intensitätswerte der Pixel im Fenster nicht mit der Intensität des Mittelpunkts verglichen, sondern mit dem Durchschnittswert aller Intensitätswerte im Fenster, um die sehr große Abhängigkeit des Ergebnisses vom Mittelpunkt zu reduzieren. Eine weitere Variante namens „Sparse Census“ sieht vor, nicht jedes Pixel des Fensters zu vergleichen, sondern nur die Pixel jeder zweiten Pixelzeile und jeder zweiten Pixelspalte zu verwenden, sodass sich mit einem 15 x 15 Pixel großen Fenster ein 64 Bits langes Bitmuster bilden lässt[122]. Da sich beide Varianten mit relativ geringem zusätzlichem Programmieraufwand implementieren lassen, konnte Census auch mit diesen Alternativen verglichen werden. Auch hier schneidet Census bei den Bildern mit größeren Beleuchtungsdifferenzen durchwegs besser ab (vgl. Abb. 10)[123].

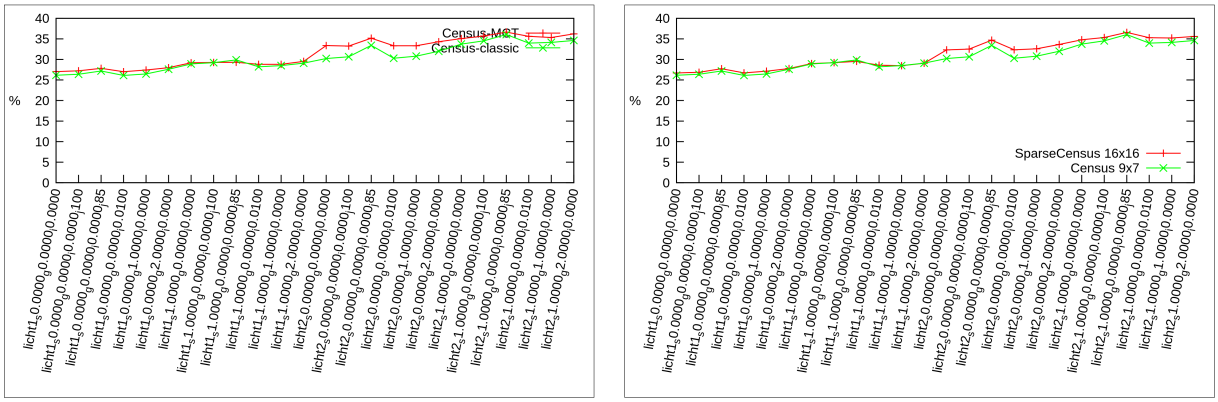


Abbildung 10: Vergleich der Kostenfunktionen bzgl. der Anzahl der Fehlerpixel beim Bildpaar „Raum Off-Axis“: Census schneidet sowohl gegenüber MCT als auch gegenüber „Sparse Census“ bei größeren radiometrischen Differenzen besser ab. Zu beachten ist, dass in diesen Grafiken die Bildvarianten anders angeordnet sind als in Abb. 9: Alle Bildvarianten mit gleicher Beleuchtung finden sich in der linken Hälfte der Grafik, alle Varianten mit unterschiedlicher Beleuchtung in der rechten Hälfte.

Auch mit einigen anderen Versuchen (beispielsweise, indem man für die Census-Transformation nicht die Intensitätswerte verwendet, sondern die Gradienten) konnte keine Verbesserung der Ergebnisse erzielt werden. ADCensus wäre also lediglich bei stabilen

121 Vgl. [Puxbaum2010]; [Irijanti2011].

122 Vgl. [Humenberger2010b]. Meiner Ansicht nach ist der dort durchgeführte Vergleich zwischen Census und „Sparse Census“ nicht fair, weil dafür gleich große Fenster verwendet werden. Dass der Rechenaufwand für „Sparse Census“ wesentlich geringer ist, ist wenig verwunderlich, weil ja nur jedes vierte Pixel im Fenster mit dem zentralen Pixel verglichen werden muss; allerdings ist dieser Gewinn sehr fragwürdig, wenn sich mit demselben geringeren Rechenaufwand ein besseres Ergebnis als mit „Sparse Census“ erzielen lässt. Für einen korrekten Vergleich müssen daher m. E. unterschiedliche Fenstergrößen für Census und „Sparse Census“ verwendet werden, die ungefähr den gleichen Rechenaufwand haben.

123 Das hat mich übrigens etwas überrascht, weil ich damit gerechnet hätte, dass MCT bei verrauschten Bildern besser abschneiden würde, weil die klassische Census-Funktion ja u. U. völlig fehlschlagen kann, wenn das zentrale Pixel im einen Bild, beispielsweise durch Schrotrauschen, einen stark verfälschten Intensitätswert besitzt. Allerdings schnitt MCT bei meinem Vergleich auch bei fast allen verrauschten Bildern (mit Ausnahme der JPEG-Bilder) schlechter ab als Census.

Beleuchtungsverhältnissen die bessere Wahl (was das gute Abschneiden in der Middlebury-Evaluation erklärt); soll ein Algorithmus – wie für diese Arbeit gefordert – jedoch mit schwierigeren Lichtverhältnissen zurecht kommen, so ist die Census-Funktion in ihrer ursprünglichen Form wohl die bessere Alternative.

Interessanterweise verbessert allerdings eine Begrenzung des maximalen Kostenwerts die Ergebnisse, deshalb wird der mittels einer Census-Transformation mit einer Fenstergröße von 9 x 7 Pixeln ermittelte Kostenwert in der Referenzimplementierung des Algorithmus mit einem Schwellenwert von 30 trunkiert.

5.3. Semi-globales Matching

Als nächsten Schritt sieht der hier vorgeschlagene Algorithmus eine Kostenaggregation mittels des semi-globalen Matchings vor. In diesem Verfahren werden die Kosten nicht wie bei den meisten Aggregationsverfahren innerhalb eines (mehr oder weniger) flexiblen oder fixen Fensters rund um das Pixel aufgerechnet, sondern entlang mehrerer Scanlinien (vgl. Abb 11).

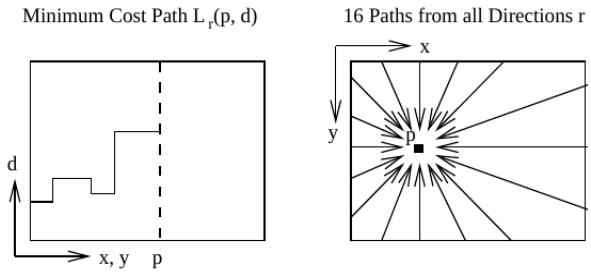


Abbildung 11: Semi-globales Matching: Suche nach dem optimalen Pfad durch das Disparity Space Image mittels Kostenaggregation entlang mehrerer Scanlinien.

Dieses Verfahren ähnelt sehr stark der „dynamischen Programmierung“, bei der es darum geht, den optimalen (d. h. mit den geringsten Kosten verbundenen) Pfad entlang der Bildzeilen in den rektifizierten Bildern zu finden. Das semi-globale Matching sucht diesen optimalen Pfad nicht nur entlang eines Pfades, sondern anhand mehrerer (meist 8 oder 16) Pfade.

Abgesehen von der Pfadanzahl unterscheidet sich das semi-globale Matching von der „dynamischen Programmierung“ jedoch auch noch darin, dass sie das „Ordering Constraint“ aufgibt: Bei der „dynamischen Programmierung“ wird nämlich üblicherweise verlangt, dass sich die Reihenfolge der Pixel im linken und rechten Bild nicht ändern kann. Pixel des einen Bildes können zwar im anderen verdeckt sein, aber wenn ein Pixel im einen Bild links von einem weiteren Pixel liegt, dann darf es nach dem Ordering Constraint nicht so sein, dass das Referenzpixel des ersten Bildpunkts im Referenzbild rechts vom Referenzpixel des zweiten Bildpunkt liegt. Da diese Annahme oft (wenn auch nicht immer) zutrifft, trägt sie in den verschiedenen Varianten der „dynamischen Programmierung“ dazu bei, die Ergebnisse zu

verbessern. Beim semi-globalen Matching, das die Kosten dagegen nicht nur entlang der Epipolarlinien aggregiert, lässt sich diese Einschränkung jedoch nicht aufrecht erhalten[124].

Die Kostenaggregation entlang der Pfade erfolgt nach folgender rekursiver Berechnungsformel:

$$L_r(p,d) = C(p,d) + \min\left(L_r(p-r,d),\quad L_r(p-r,d-1)+P_1,\quad L_r(p-r)+P_1,\quad \min_i L_r(p-r,i)+P_2 \right) - \min_k L_r(p-r,k)$$

$C(p,d)$ bezeichnet in dieser Formel die Kosten des Pixels p für die Disparitätsstufe d; $p-r$ ist das Vorgängerpixel von p auf der Scanlinie r; und der Term in der letzten Zeile der Formel sorgt dafür, dass die Kosten entlang einer Scanlinie nicht über den Wert $C_{max}+P_2$ hinaus ansteigen können, sodass eine effiziente Speicherung der aggregierten Kosten in einer 16-Bit-Ganzzahl möglich ist. Wenn Teile der angegebenen Formel nicht berechenbar sind, werden sie einfach weggelassen: Wenn es kein Vorgängerpixel gibt, werden nur die Kosten des Pixels aufaddiert; existiert die Disparitätsstufe $d-1$ oder $d+1$ nicht, dann wird sie auch nicht berücksichtigt. P_1 und P_2 („P“ steht für „penalty“) werden als also „Strafe“ den Kosten hinzugezählt, wenn sich die Disparität geringfügig (eine Disparitätsstufe) oder stärker ändert. Da Diskontinuitäten oft mit einer plötzlichen Änderung der Farbintensität einhergehen, empfiehlt Hirschmüller, die Strafkosten flexibel zu berechnen[125]:

$$P_2 = \frac{P_2'}{|I_p - I_{p-r}|}$$

Der Wert P_2' ist hierbei fix vorgegeben; bei I_p und I_{p-r} handelt es sich um die Intensitätswerte des Pixels p und seines Vorgängers. Falls der Wert P_2 durch diese Berechnung zu klein wird, muss er noch einmal nach oben korrigiert werden[126]:

$$P_2 = P_1+1, \text{ falls } \frac{P_2'}{|I_p - I_{p-r}|} \leq P_1$$

124 Hirschmüller merkt daher zu Recht an, dass der Ansatz trotz seiner Verwandtschaft zur „dynamischen Programmierung“ eher der sog. „Scanline Optimization“ ähnelt als jenem der klassischen „dynamischen Programmierung“, vgl. [Hirschmüller2008], S. 4 der Online-Version.

125 Seltsamerweise scheint es niemanden zu stören, dass die Formel nicht berechenbar ist, wenn der Intensitätswert des Pixels identisch ist mit jenem des Vorgängerpixels. Da das meinen Computer aber sehr wohl stört, setze ich in diesem Fall $P_2 = P_2'$. Übrigens verzichtet Hirschmüller in [Hirschmüller2008] wieder auf die von ihm vorgeschlagene flexible Berechnung, während sie in neueren Artikeln anderer Autor/inn/en weiterhin – teilweise abgewandelt – angewandt wird.

126 Diese Vorgangsweise findet sich in: [Hermann2009], S. 635.

Die auf diese Weise ermittelten Pfadkosten werden dann einfach aufaddiert[127]:

$$S(p,d) = \sum_r L_r(p,d)$$

Für die Implementierung werden also zwei „Disparity Space Images“ benötigt, eines, in dem die Kosten gespeichert werden (die ja für jede Pfadrichtung wieder benötigt werden), und ein weiteres, in dem die aggregierten Kosten abgelegt werden[128]. Hirschmüller empfiehlt, die Aggregation zumindest in 8, besser noch in 16 Richtungen durchzuführen. Andere Autor/inn/en berichten jedoch, dass der Qualitätsgewinn der Aggregation auf 8 Pfaden gegenüber 8 Richtungen so gering sei, dass sich der Mehraufwand nicht lohne[129]. Daher wird auch für den hier vorgestellten Algorithmus vorgeschlagen, sich auf die Aggregation in 8 Richtungen zu beschränken.

5.4. Disparitätsberechnung und erneute Abschätzung des Disparitätssuchraums

Die Disparitätsauswahl wird beim semi-globalen Matching wie bei den lokalen Algorithmen mittels des üblichen „Winner-takes-all“-Prinzips durchgeführt.

Wenn auch noch die Disparitäten für einen Teil des restlichen Bilds berechnet werden sollen, können die bisher ermittelten Disparitätwerte herangezogen werden, um den Disparitätssuchraum für weitere Pixel einzuschätzen. Dazu wird in einem ersten Schritt der Disparitätswert der bereits berechneten Pixel auf $[\, d_{WTA} \,,\, d_{WTA} + 1]$ zurückgesetzt.

Die berechneten Disparitäten sind allerdings nicht in allen Fällen die korrekten Disparitäten. Das ist insbesondere dort problematisch, wo der zuvor festgelegte Disparitätssuchraum die korrekte Disparitätsstufe nicht umfasst. Das kann beispielsweise dann passieren, wenn sich bei einer gleichmäßigen schiefen Fläche (wie zum Beispiel bei der Tischplatte im „Raum Off-Axis“ und im „Raum Poltergeist“) die Disparität allmählich ändert, die Farbwerte aber relativ gleich bleiben, weil ja die Disparitätssuchräume nur rechtwinklig zur Kameraachse verteilt werden[130]. Wenn die ausgewählte Disparität jene am äußersten unteren oder oberen

127 Das „S“ steht übrigens für „smooth“, weil die Kostenaggregation die Disparitätskarte glättet.

128 Der hohe Speicherbedarf des semi-globalen Matchings wird in der Fachliteratur immer wieder beklagt; manche Abwandlungen des Algorithmus wurden ausdrücklich mit der Absicht entwickelt, den Speicherbedarf zu verringern, vgl. [Won2011].

129 Vgl. [Humenberger2010a], S. 4. Der ADCensus-Algorithmus aggregiert sogar nur in 4 Richtungen, vgl. [Mei2011], S. 4.

130 Dass die lokalen Algorithmen bei der Kostenaggregation so genannte „frontoparallele“ Ebenen einseitig bevorzugen, ist ja ein bekanntes und oft diskutiertes Phänomen. M. E. ist gerade eine der Stärken des semi-globalen Matchings, dass es geneigte Flächen nicht mehr oder weniger gleich behandelt wie Diskontinuitäten, sondern bei einer geringfügigen Änderung der Disparität nur geringe Strafkosten aufaddiert. Auch wenn hier keine Kosten aggregiert werden, sondern die Unterstützungsregionen der „Cross-based Aggregation“ eingesetzt werden, um Disparitätssuchräume zu verteilen, zeigt sich hier trotzdem ein ähnliches Problem.

Rand des Disparitätssuchbereichs ist, könnte das ein Hinweis darauf sein, dass genau dieser Fall eintritt. Deshalb werden diese Pixel und die in einem bestimmten Umkreis um sie herum liegenden Bildpunkte als „noch nicht berechnet“ markiert, damit sie in der nächsten Runde noch einmal neu berechnet werden, in der Hoffnung, dass diese Berechnung (bei welcher der in dieser Runde ermittelte Disparitätswert nun in der Mitte des Suchraums liegt) dann zuverlässiger ist[131].

Anschließend werden die in Abschnitt 5.1 beschriebenen Schritte (Ausweitung des Suchbereichs nach oben und unten um eine bestimmte Disparitätszahl, dreimalige Verteilung des Suchraums auf die Unterstützungsregionen der „Cross-based Aggregation“, Verteilung in horizontaler und vertikaler Richtung in einem fixen Abstand) erneut durchgeführt. Die Berechnung muss in der nächsten Runde nur mehr für die als unsicher markierten und für die neu hinzugekommenen Pixel durchgeführt werden. Die beschriebene Abfolge (Einschätzung des Disparitätssuchraums, Berechnung neu hinzugekommener Pixel, Disparitätsauswahl, neuerliche Initialisierung des Disparitätssuchraums mittels der bereits berechneten Disparitätswerte wird so lange wiederholt, bis ein bestimmtes Abbruchkriterium erreicht wird. In der Referenzimplementierung wird der Vorgang wiederholt, bis entweder für zumindest 80 % der Pixel ein Disparitätswert berechnet wurde oder fünf Berechnungsrunden durchgeführt wurden.

5.5. Disparitätsverfeinerung

Falls kein neuerlicher Berechnungsdurchgang mehr notwendig erscheint, werden einige abschließende Schritte durchgeführt. Subpixel-Genauigkeit kann mittels der in Abschnitt 4.2.3 beschriebenen Interpolation durch eine quadratische Kurve (Parabel) erreicht werden. Als Formel dafür eignet sich:

$$d_{float}(p) = d_{WTA}(p) \; + \; \frac{S(p, d_{WTA}-1) - S(p, d_{WTA}+1)}{2\left(S(p, d_{WTA}-1) + 2\,S(p, d_{WTA}) - S(p, d_{WTA}+1)\right)}$$

Nach dem in Abschnitt 4.2.4 vorgestellten Verfahren kann eine Disparitätskarte für das Referenzbild bestimmt werden, ohne eine neuerliche Kostenberechnung und -aggregation durchführen zu müssen. Diese zweite Disparitätskarte kann für einen „Left-right-Consistency Check“ herangezogen werden.

Das semi-globale Matching sieht an dieser Stelle – wie auch die meisten anderen Algorithmen – noch weitere Schritte der Disparitätsverfeinerung vor. Dabei werden die im Left-right-Consistency Check als unkorrekt markierten Pixel in verdeckte und falsch berechnete

131 Weiter gehende Möglichkeiten, unsichere Pixel zu erkennen, werden im nächsten Kapitel diskutiert.

Pixel eingeteilt. Während erstere mit einer Disparitätsstufe eines „sicheren“ Pixels aus der Nachbarschaft, das vermutlich zum Hintergrund gehört, versehen werden, wird die Disparität der ersteren mithilfe umliegender „sicherer“ Disparitätswerte interpoliert.

Da die Aufgabenstellung dieser Arbeit keine „dichte“ Disparitätskarte erfordert, kann auf diese abschließenden Schritte jedoch zugunsten einer kürzeren Laufzeit verzichtet werden. Stattdessen können die Pixel, die den Left-right-Consistency Check nicht bestehen, einfach als ungültig markiert werden. Bezüglich der übrigen Pixel können die für den ausgewählten Disparitätswert aggregierten Kosten darüber Auskunft geben, wie groß die Wahrscheinlichkeit ist, dass der gewählte Disparitätswert korrekt ist.

5.6. Evaluation

Der vorgeschlagene Algorithmus wurde nicht zuerst auf dem Papier entworfen und erst danach implementiert und getestet. Vielmehr wurden Erfolg versprechende Ideen, die in der Fachliteratur aufzufinden waren, implementiert, zum Teil mit Alternativen verglichen und dann wieder verworfen oder beibehalten[132]. In diesem Sinn fanden während der Konzeption des Algorithmus immer wieder auch Phasen der Zwischenevaluation (z. B. Vergleich verschiedener Kostenfunktionen) statt, deren Ergebnisse zwar in die weitere Konzeption des Algorithmus einflossen, aber hier nicht in allen Details präsentiert werden können.

An dieser Stelle soll jedoch über diese Zwischenergebnisse hinaus die Frage gestellt werden, welche Ergebnisse der Algorithmus in seiner hier vorgestellten Version hervorbringt. Im Folgenden wird zuerst vorgestellt und diskutiert, wie gut es dem Algorithmus gelingt, den Disparitätssuchbereich einzuschätzen, und dann, wie es um die Qualität der berechneten Disparitätskarten bestellt ist.

5.6.1. Einschätzung und Einschränkung des Disparitätssuchbereichs

Zur Evaluation der Einschätzung des Disparitätssuchbereichs wurden nur Featurepaare verwendet, deren korrekte Zuordnung zuvor überprüft wurde[133]. Dazu wurde folgendermaßen

132 Beispielsweise wurde die „Cross-based Aggregation“ ursprünglich nicht für die Suchraumeinschätzung implementiert, sondern als Methode der Kostenaggregation. Allerdings stellte sich heraus, dass die Aneinanderreihung von „Cross-based Aggregation“ und semi-globalem Matching, wie sie der ADCensus-Algorithmus vorsieht (vgl. [Mei2011]), zu zeitaufwändig ist, bei einem direkten Vergleich der beiden Aggregationsmethoden das semi-globale Matching jedoch die besseren Ergebnisse bei annähernd gleicher Rechendauer hervorbrachte, deshalb wurde die „Cross-based Aggregation“ verworfen. Als ich dann nach einer halbwegs zuverlässigen Möglichkeit suchte, mit Hilfe der von den Features bekannten Disparitäten den Disparitätssuchraum anderer Pixel einzuschätzen, erschienen mir die Unterstützungsregionen der „Cross-based Aggregation“ ein gangbarer Weg.

133 Diese Vorgangsweise wurde von Herrn Häming in seinem E-Mail vom 11. Mai 2012 vorgeschlagen, um nicht auch noch zusätzlich Probleme lösen zu müssen, die unmittelbar mit dem Feature-Matching zusammenhängen.

vorgegangen: Mittels eines kleinen Hilfsprogramms[134] wurden – unter Nutzung der freien Programmbibliothek OpenCV – Features gesucht und einander zugeordnet. Eine von Klaus Häming vom Lehrgebiet Mensch-Computer-Interaktion programmierte Erweiterung des Rektifizierungswerkzeugs „depth4disp“ ermöglichte, die gefundenen Features zu rektifizieren. Ein weiteres Hilfsprogramm[135] siebt aus den gefundenen und rektifizierten Featurepaaren jene aus, deren Disparität stärker als ein Schwellenwert (1,0) von der in der „Ground Truth“ abgelegten korrekten Disparität abweicht. Zur Initialisierung der Abschätzung des Disparitätssuchraums wurden dann diese rektifizierten und validierten Featurepaare herangezogen.

In Anhang 3 zeigen einige Bilder, welche Ergebnisse die Abschätzung des Disparitätssuchraums bei den vom Lehrgebiet Mensch-Computer-Interaktion sowie bei den Bildpaaren „Venus“, „Teddy“ und „Cones“ der Middlebury-Datensätze hervorbringen. Der Algorithmus wurde jeweils abgebrochen, wenn für zumindest 80 % eine Festlegung des Suchraums getroffen worden war, spätestens jedoch nach 5 Suchdurchgängen.

Das Ergebnis fällt differenziert aus: Wenn viele Featurepaare zur Verfügung stehen, gelingt die Einschätzung recht gut. In vielen Fällen („Raum Off-Axis“, „Clown-in-front“, „Venus“, „Teddy“, „Cones“) werden schon im ersten Durchgang mehr als 80 % der Features mit Disparitätssuchräumen versehen, einige Bildpaare benötigen zwei („Sokrates Off-Axis“, „Raum Poltergeist“) oder vier („Clown-Sokrates“) Durchgänge.

Die festgelegten Disparitätssuchräume sind innerhalb der konvexen Hülle der verwendeten Featurepaare fast immer korrekt. Darüber hinaus gelingt die Einschätzung in vielen Fällen auch sehr gut, in einigen Fällen versagt sie jedoch. Auf zusammenhängende frontoparallele oder nur wenig geneigte Flächen wird der Disparitätssuchraum korrekt übertragen, bei schiefen Flächen (sichtbar in Abb. 21b an der rechten vorderen Tischecke) und an Diskontinuitäten (s. Abb. 20b) versagt der Algorithmus. Während der Disparitätssuchraum bei schiefen Flächen oft im nächsten Durchgang richtig eingeschätzt wird (gut sichtbar beim „Raum Poltergeist“ an der Tischfläche und an der linken Seite der Wandfläche, wenn man Abb. 23b mit 23c vergleicht), misslingt die korrekte Einschätzung bei Disparitäten oft auch in den nächsten Durchgängen (vgl. Abb. 20bc).

Sofern es in der Szene eine oder zwei zentrale Figuren gibt[136] („Sokrates Off-Axis“, „Clown-Sokrates“, „Clown-in-front“), werden zumindest wesentliche Teile dieser Figur(en) erfasst, am schlechtesten beim Bildpaar „Clown-Sokrates“, was zweifellos mit der sehr ge-

134 Die entsprechende Programmdatei in der Referenzimplementierung trägt den Namen „FeatureMatcher.cpp“.
135 Das Programm ist in der Datei „FeatureValidator.cpp“ abgelegt.
136 Wie gut sich der Algorithmus in diesem Fall schlägt, scheint mir im Zusammenhang mit der Freihanderfassung von 3D-Objekten besonders wichtig zu sein.

ringen Anzahl an Featurepaaren (4 Paare) zusammenhängt. Zu beachten ist einerseits, dass bei einer so geringen Zahl an korrekt erkannten Featurepaaren auch keine automatisch ablaufende Rektifizierung möglich wäre, und andererseits, dass am Ende der Disparitätsberechnung zumindest sehr viele der aufgrund der falschen Einschätzung des Disparitätssuchraums falsch berechneten Disparitäten auch als nicht vertrauenswürdig erkannt und entsprechend ausgesiebt werden.

Neben der möglichst passenden *Einschätzung* des Disparitätssuchraums ist aber auch noch der Aspekt der möglichst effizienten *Einschränkung* desselben von Bedeutung. Um das zu bemessen, kann man die Anzahl der Disparitätswerte, die nach Durchführung zu berechnen sind, mit der Anzahl der Werte vergleichen, die zu berechnen wären, wenn man die minimale und maximale Disparitätsstufe des Gesamtbildes kennen würde (wie dies bei den Middlebury-Datensätzen ja üblicherweise geschieht).

Bei dieser Berechnungsweise ergeben sich die folgenden Prozentzahlen nach der ersten Suchraumeinschätzung: Venus 133,8 %; Teddy 53,8 %; Cones 57,3 %; Sokrates Off-Axis 91,2 %; Raum Off-Axis 63,2 %; Clown-Sokrates 17,5 %; Raum Poltergeist 38,1 %; Clown-in-front 41,4 %. Die Aufgabe der Suchraumeinschränkung (die auch der Reduktion des Rechenaufwands dient) erledigt der vorgeschlagene Algorithmus also sehr gut. Nur beim Bildpaar „Venus“, das besonders wenige Disparitätsstufen aufweist (20 Stufen) steigt der Rechenaufwand um ein Drittel. Aber insbesondere bei den höher aufgelösten Bildern mit besonders vielen Disparitätsstufen führt der Algorithmus zu einer deutlichen Verringerung des Rechenaufwands.

5.6.2. Qualität der Disparitätskarten

Wie es um die Qualität der berechneten Disparitätskarten bestellt ist, wird in den meisten Fachartikeln an den Kriterien der Middlebury-Evaluation gemessen. Dieser Vergleich ist im vorliegenden Fall nicht ganz fair, weil der Algorithmus mit anderen Designzielen erstellt wurde. Wegen der spezifischen Anforderungen dieser Arbeit wurden ja teilweise Entscheidungen getroffen, die die Qualität der Ergebnisse bezüglich der Middlebury-Datensätze zugunsten anderer Vorteile (Beleuchtungsunabhängigkeit, Geschwindigkeit) verschlechtern (Verwendung der Census-Kostenfunktion statt einer Verbindung von AD und Census, vgl. Abschnitt 5.2; Aussieben von Pixeln, die den „Left-right-Consistency Check“ nicht bestehen, statt deren Ersetzung oder Interpolation durch die Disparitäten der Nachbarpixel, vgl. 5.5).

Trotzdem ermöglicht ein Vergleich mit der Middlebury-Evaluation eine ungefähre Einschätzung der Ergebnisqualität. Dazu ist es in der Referenzimplementierung möglich, den

Algorithmus zur Einschätzung des Disparitätssuchraums abzuschalten und stattdessen einen fixen Disparitätsbereich für alle Pixel berechnen zu lassen. Für die Ergebnisse dieses Durchlaufs wird die Prozentzahl der Fehlerpixel berechnet: einmal für „alle“ Pixel[137]; einmal für die nicht-verdeckten Pixel; und einmal für die in der Nähe der Diskontinuitäten liegenden Pixel.

Die Ergebnisse finden sich im Anhang 4. Der Algorithmus liegt in der Middlebury-Evaluationstabelle beim Schwellenwert 1,0 an der 102. Stelle und beim Schwellenwert 0,5 an der 61. Stelle von insgesamt 135 Algorithmen. Angesichts der Tatsache, dass auf eine Disparitätsverfeinerung (abgesehen von der Subpixelberechnung mittels Parabel-Interpolation) weitgehend verzichtet wurde und keinerlei Optimierung im Hinblick auf die Erfordernisse der Middlebury-Datensätze stattfand, ist das zwar kein glänzendes, aber doch ein passables Ergebnis.

Bei den vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellten Datensätzen kommt der Algorithmus mit der Szene „Clown-in-front“ am besten zurecht, hat aber mit der Szene „Raum Poltergeist“ erhebliche Schwierigkeiten. Bei den deutlich höheren Fehlerpixelzahlen ist allerdings auch zu bedenken, dass die Bilder eine wesentlich höhere Auflösung und zum Teil sehr viel mehr Disparitätsstufen aufweisen als die Middlebury-Datensätze.

Von größerem Interesse – allerdings auch noch schwerer einschätzbar – ist die Frage, wie sich der Algorithmus in Bezug auf die in der Themenstellung genannten Anforderungen verhält.

Schnelligkeit

Die Laufzeit des Algorithmus wurde auf einem Laptop mit einem „AMD Turion 64 X2 Mobile Technology TL-60“ Prozessor mit 2 GB RAM getestet. Der Algorithmus war dabei so eingestellt, dass er abbrach, sobald zumindest 80 % der im rektifizierten Basisbild gültigen Bildpixel berechnet worden waren, spätestens jedoch nach 5 Berechnungsrunden. Die Laufzeit auf dem Referenzsystem betrug mit diesen Einstellungen: Venus 5,8 Sekunden; Teddy 4,24 Sekunden; Cones 5,11 Sekunden; Sokrates Off-Axis 12,35 Sekunden (2 Berechnungsrunden); 11,31 Sekunden; Clown-Sokrates 25,76 Sekunden (4 Berechnungsrunden); Raum Poltergeist 23,76 Sekunden (2 Berechnungsrunden); Clown-in-front 23, 72 Sekunden. Selbst auf dem nicht sehr schnellen Referenzsystem blieb der Algorithmus also bei allen Bildpaaren

137 Ich schreibe hier „alle“ bewusst in Anführungszeichen, weil in der Middlebury-Evaluation auch bei der Berechnung „aller“ Pixel ein Teil des Bildes mittels einer Maske ausgeblendet wird: Teilweise werden die Randpixel ausgeblendet, teilweise Pixel, für die bei der Erstellung der „Ground Truth“ die korrekte Disparität nicht ermittelt werden konnte.

deutlich unter dem als „Schmerzgrenze“ genannten Zeitlimit von 30 Sekunden[138]. Die Referenzimplementierung nutzt dabei keinerlei Beschleunigung durch den Grafikprozessor.

Beleuchtungsunabhängigkeit und Rauschunabhängigkeit

Die im Anhang 6 präsentierten Übersichtsgrafiken zeigen, dass bei schwierigeren Beleuchtungsverhältnissen und stärkerem Bildrauschen zwar mehr Pixel vom Algorithmus als nicht vertrauenswürdig gekennzeichnet werden (grüne Linie), allerdings sowohl die Anzahl der Bildpunkte, für die überhaupt eine Berechnung durchgeführt wird (rote Linie), als auch die Anzahl der Pixel, deren Disparität falsch berechnet wurde, obwohl sie als vertrauenswürdig eingestuft wurden (blaue Linie), relativ stabil bleibt. Die ausgesprochen schlechten Werte bei der Szene „Clown-Sokrates“ gehen übrigens – wie ein Vergleich mit den Fehlerwerten in Anhang 4, die mit einem vorgegebenen fixen Disparitätssuchraum erstellt wurden, zeigt – vor allem auf die sehr schlechte Einschätzung des Disparitätssuchraums aufgrund der zu geringen Anzahl an Featurepaaren zurück. Interessant ist auch, dass der unterschiedlich beleuchtete „Sokrates Off-Axis“ dem Algorithmus wesentlich weniger zu schaffen macht, als der von verschiedenen Seiten beleuchtete „Raum Off-Axis“.

Bezüglich der Resistenz gegen das Bildrauschen verhält es sich genau umgekehrt: Das Bildrauschen wirkt sich beim „Sokrates Off-Axis“ negativer aus als beim „Raum Off-Axis“, wohl aufgrund der recht großen wenig texturierten Wandfläche hinter der Sokrates-Statue. Ansonsten kommt der Algorithmus allerdings im Allgemeinen relativ gut mit verschiedenen Formen des Rauschens zurecht. Eine Ausnahme bilden hier die mit Qualitätsstufe 85 % (Voreinstellung vieler Digitalkameras) gespeicherten JPEG-Bilder: Dadurch lässt sich der Algorithmus wesentlich stärker negativ beeinflussen als durch die verschiedenen Formen des Bildrauschens im engeren Sinn.

Eignung für wenig texturierte Objekte

Die Eignung für wenig texturierte Objekte wurde bei der Konzeption des Algorithmus berücksichtigt[139], kann aber nur partiell beurteilt werden.

Bei der texturarmen Stelle im rechten oberen Bildbereich der „Teddy“-Szene (zwischen dem – von der Betrachterin aus gesehen – rechten Arm und rechten Bein des Teddybärs), auf die manche Algorithmen extra eingehen, werden die Disparitäten der betreffenden Pixel vom

138 Eine frühere Version des Algorithmus hat Herr Häming einmal auf seinem wesentlich besser ausgestatteten PC zu Vergleichszwecken laufen lassen – dabei benötigte das Programm im Vergleich zur Laufzeit auf meinem Laptop nur ungefähr die halbe Rechenzeit.

139 Das semi-globale Matching wurde nicht zuletzt deswegen ausgwählt, weil es vom theoretischen Standpunkt sowohl mit einzelnen durch Bildrauschen verursachten Ausreißern als auch mit kleineren texturarmen Bildregionen gut zurechtkommen sollte.

Algorithmus als unsicher erkannt und entsprechend verworfen. Das erscheint angesichts der Tatsache, dass laut Vorgabe keine „dichte“ (also alle Pixel berücksichtigende) Disparitätskarte berechnet werden muss, als akzeptabel. Wenn man die Dichte der Ergebnisse auch an texturarmen Stellen erhöhen möchte, führt letztlich kein Weg an einer Disparitätsverfeinerung herum, die die unsicheren Pixel mit Disparitätswerten aus ihrem Umfeld versieht – dafür gibt es ja in der Fachliteratur eine Fülle an Vorschlägen.

Eine Durchsicht der Ergebnisse zeigt jedoch, dass bei den vom Lehrgebiet Mensch-Computer-Interaktion zur Verfügung gestellten Datensätzen nicht so sehr Probleme in texturarmen Bildregionen eine Rolle spielen als vielmehr sehr schräge Flächen (Tischfläche) und Spiegelungen (Fernseher, Glasvitrinen). Sofern diese Datensätze für das am Lehrgebiet relevante Anwendungsfeld repräsentativ sind, wäre es daher unter Umständen wichtiger, Lösungen für diese Problembereiche zu finden.

6. Offen bleibende Fragen

Wie bei wissenschaftlichen Arbeiten oft der Fall, so ist es auch bei dieser Untersuchung: Zum Schluss bleiben mehr Fragen als Antworten. Der begrenzte Umfang dieser Arbeit erlaubt nicht mehr, alle Fragen, die sich im Erstellungsprozess ergeben haben, auszuführen. Die wichtigsten sollen jedoch im Folgenden zumindest kurz umrissen werden.

6.1. Fragen im Zusammenhang mit der Bestimmung des Disparitätssuchraums

Bei der Erarbeitung des vorgestellten Algorithmus wurde viel Zeit und Energie in die Erstellung eines Algorithmus zur Einschätzung und Einschränkung des Disparitätssuchraums investiert. In diesem Zusammenhang könnte man beispielsweise die Frage stellen, ob es nicht noch bessere (d. h. vielleicht weniger aufwändige, aber trotzdem zielführende) Methoden gäbe, die „Nachbarschaft“ zweier Features zu bestimmen. Noch wichtiger wäre allerdings die Frage, wie man die Ergebnisse der Suchraumeinschätzung, insbesondere an den Diskontinuitäten, verbessern könnte.

Sehr interessant könnte es in diesem Zusammenhang auch sein, das hier vorgestellte Verfahren mit der jüngst von Hermann und Klette beschriebenen Vorgangsweise, die mit einem Down- und Upscaling der Bilder arbeitet, zu vergleichen. Darüber hinaus könnten auch noch andere Algorithmen zur Bestimmung des Disparitätssuchraums genauer untersucht werden, die für die vorliegende Arbeit nicht mehr berücksichtigt werden konnten[140].

6.2. Fragen im Zusammenhang mit der Disparitätsberechnung

Im Zusammenhang mit der eigentlichen Disparitätsberechnung wäre vor allem die Einbeziehung von Algorithmen wünschenswert, die während des Schreibens dieser Arbeit neu veröffentlicht wurden[141].

Was die Disparitätsverfeinerung betrifft, könnte man, wenn gewünscht, sicherlich mit vorhandenen Methoden der Disparitätsergänzung und -interpolation eine dichteres Ergebnis bei den Disparitätskarten erreichen. Zur Frage des Umgangs mit sehr schrägen Flächen gibt es dort und da bereits weitergehende Überlegungen. Aber für das Problem, wie man beim Stereo Matching mit spiegelnden Flächen zurechtkommt, ist – soweit ich es sehen kann – noch weit

140 Vgl. Jan Cech, Growing Correspondence Seeds : A Fast Stereo Matching of Large Images. Online: http://cmp.felk.cvut.cz/~cechj/GCS/ (Abrufdatum: 2012-09-16).

141 Vgl. z. B. den in der Middlebury-Evaluation unter dem Akronym „CrossLMF“ geführten Algorithmus.

und breit keine Lösung in Sicht, entsprechende Forschungsbemühungen wären daher m. E. dringend vonnöten.

Beim Lesen der einschlägigen Fachartikel fiel mir häufig auf, dass oft darauf verzichtet wird, auf Grenzfälle einzugehen. So ist es beispielsweise nicht ohne Weiteres möglich, auf korrekte Weise die Hamming-Distanz zwischen einem Pixel am Bildrand und einem Pixel in der Bildmitte zu bestimmen, weil das Fenster, das zur Census-Bildung herangezogen wird, am linken und am rechten Bildrand ja nicht vollständig ist. Werden die Census-Bits der nicht-sichtbaren Pixel einfach fix auf 0 gesetzt, dann kann das Matching mit dem zweiten Pixel u. U. einen viel zu niedrigen oder aber einen viel zu hohen Wert ergeben. (Bei einem 9 x 7 großen Fenster kann der Wert immerhin um +/- 28 Punkte daneben liegen.) Zieht man hingegen nur jene Census-Bits heran, die in den Census-Bitmustern beider Pixel berücksichtigt werden konnten, dann bewegen sich die „Kosten“ für dieses Matching bei der genannten Fenstergröße nur mehr im Bereich von 0–35 statt 0–63, was ein Matching mit den Randpixeln tendeziell bevorzugen würde.

Noch vertrackter wird die Angelegenheit, wenn es um die Kostenaggregation geht. Für Pixel in der äußersten linken Bildpixelspalte können ja nur Matching-Kosten für Disparitätsstufen berechnet werden, die größer oder gleich Null sind, in der nächsten Spalten für die Stufen größer oder gleich –1, dann für größer oder gleich –2 usw., d. h. in jeder Pixelspalte kommt eine neue Disparitätsstufe hinzu. Bei der Kostenaggregation auf dem Pfad, der von links nach rechts geht, können daher in der dritten Spalte von links für die Disparitäten ab Null bereits je zwei Kostenwerte aufaddiert werden, für die Disparität –1 nur ein Vorgängerwert, und für die Disparität kann erst hier mit der Aggregation begonnen werden. Wie soll also die Kostenaggregation auf jenen Disparitätsstufen vonstatten gehen, deren Vorgängerkosten unbekannt sind? Die Kostenaggregation ab dem ersten bekannten Wert neu zu beginnen, verschafft diesen Disparitäten gegenüber den anderen, die bereits Kosten ihrer Vorgängerpixel aufaddiert bekommen haben, einen Vorteil; Wenn man hingegen jene Teile der in Abschnitt 5.3 für das semi-globale Matching genannten Formel weglässt, die nicht berechenbar sind, würde man die neu hinzukommenden Disparitätsstufen benachteiligen, weil sie zumindest mit den Strafkosten für den Wechsel einer Disparitätsstufe – zusätzlich zu den von dieser Disparitätsstufe übernommenen Vorgängerkosten – belastet würden.

Dieses Problem verschärft sich, wenn man, wie in der vorliegenden Arbeit vorgeschlagen, den Disparitätssuchraum nicht einheitlich für das gesamte Bild festlegt, sondern innerhalb des Bildes lokal variiert.

6.3. Fragen, die sich aus dem Zusammenspiel von Einschätzung des Disparitätssuchraums und Disparitätsberechnung ergeben

Wenn sich der Disparitätssuchraum innerhalb des Bildes ändert, kann das beschriebene Problem, das sonst nur an den Rändern auftritt, an jeder Stelle des Bildes auftreten, nämlich immer dann, wenn sich der Disparitätssuchraum von einem Bild zum Nachbarbild ändert. In den Testdaten ist das Problem dort sichtbar, wo die Einschätzung des Disparitätssuchraums im ersten Durchgang misslingt (z. B. beim Bildpaar „Raum Poltergeist“ auf der linken vorderen Tischfläche), sodass die korrekte Disparitätsstufe nicht innerhalb des festgelegten Disparitätssuchraums liegt. Obwohl der Disparitätssuchraum in den benachbarten Regionen in der nächsten Runde oft richtig eingeschätzt wird, misslingt nicht selten auch die Berechnung der Nachbarregionen im nächsten Schritt, obwohl deren Disparitätssuchraum korrekt eingeschätzt worden wäre. Da die neu hinzugekommenen Disparitätsstufen bei ihren Vorgängerwerten keine gültigen Kostenwerte vorfinden, werden ihnen fälschlicherweise Strafkosten für Disparitätssprünge hinzuaddiert, die sie gegenüber anderen Disparitätsstufen benachteiligen[142].

Zwei mögliche Vorgangsweisen, um diesem Problem zu begegnen, die aus Zeitgründen nicht mehr implementiert werden konnten, sollen hier wenigstens in ihren Grundzügen beschrieben werden. Die erste Möglichkeit wäre, dass man bei den „gültigen“ Disparitätsstufen des Vorgängerpixels berechnet, wieviel zusätzliche Kosten den Disparitätsstufen im Durchschnitt aufaddiert wurde, und diese durchschnittlich aufaddierten Kosten auch jener Disparitätsstufe aufaddiert, deren Kosten beim Vorgängerpixel entweder nicht berechenbar waren (Randpixel) oder nicht berechnet wurden (Einschränkung des Disparitätssuchraums).

Eine zweite Möglichkeit wäre, dass man die Pixel, für die bereits eine Kostenaggregation durchgeführt wurde, in der nächsten Berechnungsrunde noch einmal in die Berechnung mit einbezieht, allerdings mit einem sehr viel kleineren Disparitätssuchraum, der sich in einem festen Rahmen rund um die bereits berechneten Disparitätsstufen bewegt (also z. B. gefundene Disparitätsstufe +/-10 Stufen). Das würde zwar den Rechenaufwand erhöhen, aber eine zuverlässigere Kostenaggregation in der nächsten Berechnungsrunde ermöglichen. Als Kompensation für den zusätzlichen Rechenaufwand könnte man versuchen die Kostenaggregation in der ersten Berechnungsrunde auf vier Pfadrichtungen (statt acht) zu beschränken.

Zu tun (d. h. im wissenschaftlichen Zusammenhang v. a.: zu forschen) gäbe es also noch genug. Die vorliegende und hiermit vorgelegte Arbeit versteht sich in diesem Sinne als Zwischenergebnis und als Anregung zum Weiterdenken und Weiterforschen.

142 Dieser Effekt ist übrigens an den linken Bildrändern der für diese Arbeit erstellten Disparitätskarten sehr gut als Farbverlauf sichtbar.

7. Nachwort

Ich möchte diese Arbeit nicht beschließen, ohne mich bei verschiedenen Menschen zu bedanken, ohne die es die Arbeit der vorliegenden Form nicht geben würde.

Frau Prof. Dr. Gabriele Peters und Herrn Klaus Häming danke ich, dass sie mich mir ein Thema vorgeschlagen und dessen Bearbeitung zugetraut haben, das mich zunehmend in den Bann gezogen hat, vor allem aber, dass sie auch für eine konstante und kompetente Begleitung der Arbeit Sorge getragen haben. Von Frau Peters erhielt ich auf meine Fragen immer informative und oft auch humorvolle Antworten. Die E-Mails, die Herr Häming und ich über den Äther geschickt haben, um uns über Fachfragen auszutauschen, würden wohl viele Seiten füllen. Darüber hinaus legte Herr Häming auch tatkräftig Hand an, wenn es darum ging, Test-Datensätze zu generieren, sein Rektifizierungswerkzeug zu erweitern, damit auch die Featurepaare rektifiziert werden konnten usw. usf. Viele seiner Anregungen und Rückmeldungen sind in der einen oder anderen Weise in diese Arbeit eingeflossen, ohne dass ich das an jeder einzelnen Stelle deutlich machen hätte können.

Ein riesengroßes Danke möchte ich aber vor allem auch meiner Familie aussprechen. Meine Kinder Lea, Debora, Jona und Tobias mussten in den letzten Monaten viel zu oft von mir hören, dass der Papa leider schon wieder keine Zeit hat, weil er wieder hinterm (vorm?) Computer hockt. Sie haben das mit großer Geduld ertragen, obwohl es ihnen nicht immer leicht fiel. Danke dafür!

Der Dank, den ich meiner Frau Christine schulde, lässt sich in wenigen Worten gar nicht ausdrücken. Sie hat mir nicht nur immer wieder den Rücken frei gehalten, sondern mich auch ertragen, wenn ich – meinen Ideen und Konzepten nachsinnend – wieder einmal nur halb „bei der Sache war“. Auch beim Korrekturlesen hat sie sich tapfer durch die ihr fremde Materie durchgekämpft. Danke!

Literaturverzeichnis

BANNO, ATSUHIKO ; IKEUCHI, KATSUSHI: Disparity Map Refinement and 3D Surface Smoothing via Directed Anisotropic Diffusion. In: *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2009. Online: http://www.cvl.iis.u-tokyo.ac.jp/~vanno/Papers/iccv2009w_244.pdf (Abrufdatum: 2012-09-16). [Banno2009]

BHUSNURMATH, ARVIND ; TAYLOR, CAMILLO J.: Solving Stereo Matching Problems using Interior Point Methods. In: *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2008. Online: http://www.cis.upenn.edu/~cjtaylor/PUBLICATIONS/pdfs/Bhusnurmath3DPVT08.pdf (Abrufdatum: 2012-09-06). [Bhusnurmath2008]

BLEYER, MICHAEL ; CHAMBON, SYLVIE: Does Color Really Help in Dense Stereo Matching?. In: *International Symposium 3D Data Processing, Visualization and Transmission (3DPVT)*, 2010. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/ColorEval_3DPVT2010.pdf (Abrufdatum: 2012-09-17). [Bleyer2010a]

BLEYER, MICHAEL ; CHAMBON, SYLVIE ; POPPE, UTA ET AL.: Evaluation of Different Methods for Using Colour Information in Global Stereo Matching. In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVII, Part 3A*, 2008, S. 63-68. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/Bleyer_ColorEval_ISPRS2008.pdf (Abrufdatum: 2012-09-17). [Bleyer2008a]

BLEYER, MICHAEL ; GELAUTZ, MARGRIT: A Layered Stereo Algorithm Using Image Segmentation and Global Visibility Constraints. In: *International Conference on Image Processing (ICIP)*, 2004. Online: http://www.ims.tuwien.ac.at/media/documents/publications/Bleyer_Layered_Stereo_Global_Vis_ICIP2004.pdf (Abrufdatum: 2012-09-17). [Bleyer2004]

BLEYER, MICHAEL ; GELAUTZ, MARGRIT ; ROTHER, CARSTEN ET AL.: A Stereo Approach that Handles the Matting Problem via Image Warping. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/StereoMatting_CVPR2009.pdf (Abrufdatum: 2012-08-30). [Bleyer2009]

BLEYER, MICHAEL ; RHEMANN, CHRISTOPH ; ROTHER, CARSTEN: PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In: *Proceedings of the British Machine Vision Conference*, 2011, S. 14.1-14.11. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/PatchMatchStereo_BMVC2011.pdf (Abrufdatum: 2012-08-30). [Bleyer2011a]

BLEYER, MICHAEL ; ROTHER, CARSTEN ; KOHLI, PUSHMEET: Surface Stereo with Soft Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/SurfaceStereo_CVPR2010.pdf (Abrufdatum: 2012-09-17). [Bleyer2010b]

BLEYER, MICHAEL ; ROTHER, CARSTEN ; KOHLI, PUSHMEET ET AL.: Object Stereo — Joint Stereo Matching and Object Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/ObjectStereo_CVPR2011.pdf (Abrufdatum: 2012-08-30). [Bleyer2011b]

BROCKERS, ROLAND: Cooperative Stereo Matching with Color-Based Adaptive Local Support. In: JIANG, XIAOYI et al. (Hrsg.): *Computer Analysis of Images and Patterns*. Berlin ; Heidelberg : Springer, 2009 (Lecture Notes on Computer Science ; 5702), S. 1019-1027. Online: http://www-robotics.jpl.nasa.gov/publications/Roland_Brockers/caip2009_final_edited2.pdf (Abrufdatum: 2012-09-06). [Brockers2009]

ÇIĞLA, CEVAHIR ; ALATAN, A. AYDIN: Efficient Edge-Preserving Stereo Matching. In: *ICCV Workshop on Live Dense Reconstruction from Moving Cameras*, 2011. Online: http://cevahircigla.com/files/Efficient%20Edge-Preserving%20Stereo%20Matching.pdf (Abrufdatum: 2012-02-16). [Çığla2011]

ÇIĞLA, CEVAHIR ; ALATAN, A. AYDIN: *Edge-Aware Stereo Matching with O(1) Complexity*. Online: http://cevahircigla.com/files/Edge-Aware%20Stereo%20MAtching%20with%20O1%20Complexity.pdf (Abrufdatum: 2012-08-30). [Çığla2012]

DE-MAEZTU, LEONARDO ; MATTOCCIA, STEFANO ; VILLANUEVA, ARANTXA ET AL.: Efficient Aggregation via Iterative Block-based Adapting Support-weights. In: *International Conference on 3D Imaging (IC3D)*, 2011. Online: http://www.vision.deis.unibo.it/smatt/Papers/IC3D2011/IC3D_iFBS_2011.pdf (Abrufdatum: 2012-09-05). [De-Maeztu2011a]

DE-MAEZTU, LEONARDO ; MATTOCCIA, STEFANO ; VILLANUEVA, ARANTXA ET AL.: Linear stereo matching. In: *International Conference on Computer Vision (ICCV)*, 2011. Online: http://www.vision.deis.unibo.it/smatt/Papers/ICCV2011/ICCV2011_LinearStereo.pdf (Abrufdatum: 2012-09-17). [De-Maeztu2011b]

DENG, YI ; LIN, XUEYIN: A Fast Line Segment Based Dense Stereo Algorithm Using Tree Dynamic Programming. In: LEONARDIS, ALEŠ et al. (Hrsg.): *Computer Vision - ECCV 2006, Part III*. Berlin ; Heidelberg : Springer, 2006 (Lecture Notes on Computer Science ; 3953), S. 201-212. Online: http://dx.doi.org/10.1007/11744078_16 (Abrufdatum: 2012-09-17). [Deng2006]

ERNST, INES ; HIRSCHMÜLLER, HEIKO: Mutual Information Based Semi-Global Stereo Matching on the GPU. In: BEBIS, GEORGE et al. (Hrsg.): *Advances in Visual Computing : 4th International Symposium, ISVC 2008. Proceedings, Part I*. Berlin ; Heidelberg : Springer, 2008 (Lecture Notes on Computer Science ; 5358), . Online: http://dx.doi.org/10.1007/978-3-540-89639-5_22 (Abrufdatum: 2012-09-12). [Ernst2008]

GALES, GUILLAUME ; CROUZIL, ALAIN ; CHAMBON, SYLVIE: A Region-Based Randomized Voting Scheme for Stereo Matching. In: BEBIS, GEORGE et al. (Hrsg.): *Advances in Visual Computing*. Berlin ; Heidelberg : Springer, 2010 (Lecture Notes in Computer Science ; 6454), S. 182-191. Online: http://dx.doi.org/10.1007/978-3-642-17274-8_18 (Abrufdatum: 2012-08-30). [Gales2010]

HERMANN, SIMON ; KLETTE, REINHARD: Evaluation of a New Coarse-to-Fine Strategy for Fast Semi-Global Stereo Matching. In: HO, YO-SUNG (Hrsg.): *Advances in Image and Video Technology*. Berlin ; Heidelberg : Springer, 2012 (Lecture Notes in Computer Science ; 7087), S. 395-406. Online: http://dx.doi.org/10.1007/978-3-642-25367-6_35. (Abrufdatum: 2012-08-30). [Hermann2012]

HERMANN, SIMON ; KLETTE, REINHARD ; DESTEFANIS, EDUARDO: Inclusion of a Second-Order Prior into Semi-Global Matching. In: WADA, T. et al. (Hrsg.): *PSIVT 2009*
. Berlin ; Heidelberg : Springer, 2009 (Lecture Notes in Computer Science ; 5414), S. 633–644. Online: http://www.mi.auckland.ac.nz/tech-reports/MItech-TR-16.pdf (Abrufdatum: 2012-08-30). [Hermann2009]

HERMANN, SIMON ; MORALES, SANDINO ; VAUDREY, TOBI ET AL.: Illumination Invariant Cost Functions in Semi-Global Matching. In: KOCH, REINHARD et al. (Hrsg.): *Computer Vision – ACCV 2010 Workshops*. Berlin ; Heidelberg : Springer, 2011 (Lecture Notes in Computer Science ; 6469), S. 245-254. Online: http://dx.doi.org/10.1007/978-3-642-22819-3_25 (Abrufdatum: 2012-09-17). [Hermann2011]

HEYDEN, ANDERS ; POLLEFEYS, MARC: Multiple View Geometry. In: MEDIONI, GERARD ; KANG, SING BING (Hrsg.): *Emerging Topics in Computer Vision*. Upper Saddle River, NJ, USA : Prentice Hall PTR, 2004. S. 45-107. [Heyden2000]

HIRSCHMÜLLER, HEIKO: Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. Online: http://www.dlr.de/rm/en/PortalData/3/Resources//papers/modeler/cvpr05hh.pdf (Abrufdatum: 2012-09-06). [Hirschmüller2005]

HIRSCHMÜLLER, HEIKO: Stereo Vision in Structured Environments by Consistent Semi-Global Matching. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. Online: http://www.robotic.de/fileadmin/robotic/hirschmu/cvpr06hh.pdf (Abrufdatum: 2012-09-17). [Hirschmüller2006]

HIRSCHMÜLLER, HEIKO: Stereo Processing by Semi-Global Matching and Mutual Information. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 30 (2008), S. 328-341. Online: http://elib.dlr.de/55367/1/Stereo_Processing-Hirschm%C3%BCller.pdf (Abrufdatum: 2012-08-30). [Hirschmüller2008]

HIRSCHMÜLLER, HEIKO: Semi-Global Matching – Motivation, Developments and Applications. In: FRITSCH, DIETER (Hrsg.): *Photogrammetric Week '11*. Berlin ; Offenbach : Wichmann, 2011, . Online: http://www.ifp.uni-stuttgart.de/publications/phowo11/180Hirschmueller.pdf (Abrufdatum: 2012-09-04).

HIRSCHMÜLLER, HEIKO ; SCHARSTEIN, DANIEL: Evaluation of Stereo Matching Costs on Images with Radiometric Differences. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31 (2009), S. 1582-1599. Online: http://www.cs.middlebury.edu/~schar/papers/evalcosts-pami08.pdf (Abrufdatum: 2012-09-17). [Hirschmüller2009]

HOSNI, ASMAA ; BLEYER, MICHAEL ; GELAUTZ, MARGRIT ET AL.: Local Stereo Matching Using Geodesic Support Weights. In: *International Conference on Image Processing (ICIP)*, 2009. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/GeoSup_ICIP2009.pdf (Abrufdatum: 2012-09-17). [Hosni2009]

HOSNI, ASMAA ; BLEYER, MICHAEL ; RHEMANN, CHRISTOPH ET AL.: Real-time Local Stereo Matching Using Guided Image Filtering. In: *Hot3D 2011, the 2nd IEEE International Workshop on Hot Topics in 3D, in conjunction with ICME 2011*, 2011. Online: http://www.ims.tuwien.ac.at/research/stereo_matching/papers/RealTimeStereo_Hot3D2011.pdf (Abrufdatum: 2012-09-17). [Hosni2011]

HU, XIAOYAN ; MORDOHAI, PHILIPPOS: A Quantitative Evaluation of Confidence Measures for Stereo Vision. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* Preprint (2012-01-30), S. 1-14. Online: http://dx.doi.org/10.1109/TPAMI.2012.46 (Abrufdatum: 2012-09-17). [Hu2012]

HUMENBERGER, MARTIN ; ENGELKE, TOBIAS ; KUBINGER, WILFRIED: A Census-Based Stereo Vision Algorithm Using Modified Semi-Global Matching and Plane Fitting to Improve Matching Quality. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. Online: http://robots-at-home.acin.tuwien.ac.at/publications/conferences/PID1250463.pdf (Abrufdatum: 2012-09-17). [Humenberger2010a]

HUMENBERGER, MARTIN ; ZINNER, CHRISTIAN ; WEBER, MICHAEL ET AL.: A fast stereo matching algorithm suitable for embedded real-time systems. In: *Computer Vision and Image Understanding* 114 (2010), S. 1180-1202. [Humenberger2010b]

IRIJANTI, E. ; NAYAN, M. Y. ; YUSOFF, M. Z.: *Local Stereo Matching Algorithm Using Small-Color Census and Sparse Adaptive Support Weight*. Online: http://eprints.utp.edu.my/7207/2/PID2063845.pdf (Abrufdatum: 2012-09-17). [Irijanti2011]

KLAUS, ANDREAS ; SORMANN, MARIO ; KARNER, KONRAD: Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In: *International Conference on Pattern Recognition (ICPR)*, 2006. Online: http://old.vrvis.at/2d3d/technology/stereomatching/images/segment-based-stereomatching.pdf (Abrufdatum: 2012-09-17). [Klaus2006]

LARSEN, E. SCOTT ; MORDOHAI, PHILIPPOS ; POLLEFEYS, MARC ET AL.: Temporally Consistent Reconstruction from Multiple Video Streams Using Enhanced Belief Propagation. In: *International Conference on Computer Vision (ICCV)*, 2007. Online: http://www.inf.ethz.ch/personal/pomarc/pubs/LarsenICCV07 (Abrufdatum: 2012-09-06). [Larsen2007]

MATTOCCIA, STEFANO: A locally global approach to stereo correspondence. In: *International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2009. Online: http://www.vision.deis.unibo.it/smatt/Papers/3DIM2009/A%20locally%20global%20approach%20to%20stereo%20correspondence.pdf (Abrufdatum: 2012-08-30). [Mattoccia2009]

MATTOCCIA, STEFANO: *Stereo Vision: Algorithms and Applications*. Präsentationsfolien zu einem Seminar an der Universität Bologna. Online: http://www.vision.deis.unibo.it/smatt/Seminars/StereoVision.pdf (Abrufdatum: 2012-09-12). [Mattoccia2012]

MATTOCCIA, STEFANO ; GIARDINO, SIMONE ; GAMBINI, ANDREA: Accurate and Efficient Cost Aggregation Strategy for Stereo Correspondence Based on Approximated Joint Bilateral Filtering. In: ZHA, HONGBIN et al. (Hrsg.): *Computer Vision - ACCV 2009, Part II*. Berlin ; London : Springer, 2010 (Lecture Notes on Computer Science ; 5995), S. 371-380. Online: http://www.vision.deis.unibo.it/smatt/Papers/ACCV2009/Accurate%20and%20efficient%20cost%20aggregation%20strategy%20for%20stereo%20correspondence%20based%20on%20approximated%20joint%20bilateral%20filtering.pdf (Abrufdatum: 2012-09-06). [Mattoccia2010]

MATTOCCIA, STEFANO ; TOMBARI, FEDERICO ; DI STEFANO, LUIGI: Stereo Vision Enabling Precise Border Localization Within a Scanline Optimization Framework. In: YAGI, YASUSHI et al. (Hrsg.): *Computer Vision – ACCV 2007*. Berlin ; Heidelberg : Springer, 2007 (Lecture Notes in Computer Science ; 4844), S. 517-527. Online: http://www.vision.deis.unibo.it/smatt/Papers/ACCV2007/Stereo%20Vision%20Enabling%20Precise%20Border%20Localization%20Within%20a%20Scanline%20Optimization%20Framework.pdf (Abrufdatum: 2012-08-30). [Mattoccia2007]

MEI, XING ; SUN, XUN ; ZHOU, MINGCAI ET AL.: On Building an Accurate Stereo Matching System on Graphics Hardware. In: *GPUCV'11: ICCV Workshop on GPU in Computer Vision Applications*, 2011. Online: http://xing-mei.net/resource/pdf/adcensus.pdf (Abrufdatum: 2012-09-17). [Mei2011]

MIN, DONGBO ; LU, JIANGBO ; DO, MINH N.: A Revisit to Cost Aggregation in Stereo Matching: How Far Can We Reduce Its Computational Redundancy?. In: *International Conference on Computer Vision (ICCV)*, 2011. Online: http://diml.yonsei.ac.kr/~forevertin/conference/2011-ICCV-Min.pdf (Abrufdatum: 2012-08-30). [Min2011]

MIN, DONGBO ; SOHN, KWANGHOON: Cost Aggregation and Occlusion Handling With WLS in Stereo Matching. In: *IEEE Transactions on Image Processing* 17 (2008), S. 1431-1442. Online: http://diml.yonsei.ac.kr/~forevertin/IEEE_TIP_Stereo_2008.pdf (Abrufdatum: 2012-09-05). [Min2008]

NEILSON, DANIEL ; YANG, YEE-HONG: Evaluation of Constructable Match Cost Measures for Stereo Correspondence Using Cluster Ranking. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Online: http://mplab.ucsd.edu/wp-content/uploads/CVPR2008/Conference/data/papers/352.pdf (Abrufdatum: 2012-09-17). [Neilson2008]

NEILSON, DANIEL ; YANG, YEE-HONG: A Component-Wise Analysis of Constructible Match Cost Functions for Global Stereopsis. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33 (2011), S. 2147-2159. Online: http://dx.doi.org/10.1109/TPAMI.2011.67 (Abrufdatum: 2012-09-17). [Neilson2011]

PSOTA, ERIC T ; KOWALCZUK, JĘDRZEJ ; CARLSON, JAY ET AL.: A Local Iterative Refinement Method for Adaptive Support-Weight Stereo Matching. In: *International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, 2011. Online: http://mc2.unl.edu/publications/IPCV2011Psota.pdf (Abrufdatum: 2012-04-20). [Psota2011]

PUXBAUM, PHILIPP ; AMBROSCH, KRISTIAN: Gradient-Based Modified Census Transform for Optical Flow. In: BEBIS, GEORGE et al. (Hrsg.): *Advances in Visual Computing*. Berlin ; Heidelberg : Springer, 2010 (Lecture Notes in Computer Science ; 6453), S. 437-448. Online: http://dx.doi.org/10.1007/978-3-642-17289-2_42 (Abrufdatum: 2012-08-30). [Puxbaum2010]

RHEMANN, CHRISTOPH ; HOSNI, ASMAA ; BLEYER, MICHAEL ET AL.: Fast Cost-Volume Filtering for Visual Correspondence and Beyond. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. Online: http://www.ims.tuwien.ac.at/media/documents/publications/0276.pdf (Abrufdatum: 2012-09-17). [Rhemann2011]

SCHARSTEIN, DANIEL ; SZELISKI, RICHARD: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. In: *International Journal of Computer Vision (IJCV)* 47 (2002), S. 7–42. Online: http://cat.middlebury.edu/stereo/taxonomy-IJCV.pdf (Abrufdatum: 2012-09-17). [Scharstein2002]

SCHARSTEIN, DANIEL ; SZELISKI, RICHARD: High-Accuracy Stereo Depth Maps Using Structured Light. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. Online: http://community.middlebury.edu/~schar/papers/structlight/structlight.pdf (Abrufdatum: 2012-08-30). [Scharstein2003]

STANKIEWICZ, OLGIERD ; WEGNER, KRZYSZTOF: *Depth Map Estimation Software version 3*. MPEG/M15540, 2008. Online: http://www.multimedia.edu.pl/publications/files/m15540.pdf (Abrufdatum: 2012-09-06). [Stankiewicz2008]

SUN, JIAN ; LI, YIN ; KANG, SING BING ET AL.: Symmetric Stereo Matching for Occlusion Handling. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. Online: http://research.microsoft.com/~jiansun/papers/SymmeticStereo_cvpr05.pdf (Abrufdatum: 2012-09-17). [Sun2005]

SUN, XUN ; MEI, XING ; JIAO, SHAOHUI ET AL.: Stereo Matching with Reliable Disparity Propagation. In: *3D Imaging Modeling Processing Visualization Transmission (3DIMPVT)*, 2011. Online: http://xunsun3d.com/RDP_3DIMPVT2011.pdf (Abrufdatum: 2012-09-17). [Sun2011]

SZELISKI, RICHARD: *Computer Vision : Algorithms and Applications*. London : Springer, 2011. Eine PDF-Version dieses Buches, die voll inhaltlich, nicht jedoch bezüglich der Paginierung mit der gedruckten Version übereinstimmt, kann unter http://szeliski.org/Book/ kostenlos für den persönlichen Gebrauch heruntergeladen werden. Die Seitenangaben in dieser Arbeit beziehen sich auf den „letzten Entwurf“ dieses Buchs vom 3. September 2010: http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf (Abrufdatum: 2012-08-30). [Szeliski2011]

TAGUCHI, YUICHI ; WILBURN, BENNETT ; ZITNICK, C. LAWRENCE: Stereo Reconstruction with Mixed Pixels Using Adaptive Over-Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Online: http://www.hc.ic.i.u-tokyo.ac.jp/~yuichi/pub/StereoMixedPixels_CVPR2008.pdf (Abrufdatum: 2012-09-17). [Taguchi2008]

TOMBARI, FEDERICO ; MATTOCCIA, STEFANO ; DI STEFANO, LUIGI: Segmentation-Based Adaptive Support for Accurate Stereo Correspondence. In: *IEEE Pacific-Rim Symposium on Image and Video Technology (PSIVT)*, 2007. Online: http://vision.deis.unibo.it/fede/papers/psivt07.pdf (Abrufdatum: 2012-09-05). [Tombari2007]

WANG, LIANG ; GONG, MINGWEI ; GONG, MINGLUN ET AL.: How Far Can We Go with Local Optimization in Real-Time Stereo Matching : A performance study on different cost aggregation approaches. In: *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2006. Online: http://vis.uky.edu/~wangl/Research/Publication/2006/evaluate_aggregation_3dpvt06.pdf (Abrufdatum: 2012-09-17). [Wang2006]

WANG, LIANG ; YANG, RUIGANG: Global Stereo Matching Leveraged by Sparse Ground Control Points. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. Online: http://vis.uky.edu/~wangl/Research/Publication/2011/gcp_stereo_cvpr11.pdf (Abrufdatum: 2012-09-17). [Wang2011]

WANG, ZENG-FU ; ZHENG, ZHI-GANG: A Region Based Stereo Matching Algorithm Using Cooperative Optimization. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Online: http://vision.middlebury.edu/stereo/eval/papers/CORegion.pdf (Abrufdatum: 2012-09-17). [Wang2008]

WON, KWANG HEE ; JUNG, SOON KI: hSGM: Hierarchical Pyramid Based Stereo Matching Algorithm. In: BLANC-TALON, JACQUES et al. (Hrsg.): *Advances Concepts for Intelligent Vision Systems*. Berlin ; Heidelberg : Springer, 2011 (Lecture Notes in Computer Science ; 6915), S. 693-701. Online: http://dx.doi.org/10.1007/978-3-642-23687-7_62 (Abrufdatum: 2012-08-30). [Won2011]

WOODFORD, O. J. ; TORR, P. H. S. ; REID, I. D. ET AL.: Global Stereo Reconstruction under Second Order Smoothness Priors. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. Online: http://www.robots.ox.ac.uk/~ojw/2op/Woodford08.pdf (Abrufdatum: 2012-09-06). [Woodford2008]

XU, LI ; JIA, JIAYA: Stereo Matching: An Outlier Confidence Approach. In: *European Conference on Computer Vision (ECCV)*, 2008. Online: http://www.cse.cuhk.edu.hk/~leojia/all_final_papers/stereo_eccv08.pdf (Abrufdatum: 2012-09-17). [Xu2008]

YANG, QINGXIONG ; ENGELS, CHRIS ; AKBARZADEH, AMIR: Near Real-time Stereo for Weakly-Textured Scenes. In: *British Machine Vision Conference (BMVC)*, 2008. Online: http://vision.ai.uiuc.edu/~qyang6/publications/bmvc-08-qingxiong-yang.pdf (Abrufdatum: 2012-04-18). [Yang2008]

YANG, QINGXIONG ; WANG, LIANG ; YANG, RUIGANG ET AL.: Real-time Global Stereo Matching Using Hierarchical Belief Propagation. In: *British Machine Vision Conference (BMVC)*, 2006. Online: http://vis.uky.edu/%7Ewangl/Research/Publication/2006/realtimebp_bmvc2006.pdf (Abrufdatum: 2012-09-17). [Yang2006]

YANG, QINGXIONG ; WANG, LIANG ; YANG, RUIGANG ET AL.: Stereo Matching with Color-Weighted Correlation, Hierarchical Belief Propagation, and Occlusion Handling. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 31 (2009), S. 492-504. Online: http://vision.ai.uiuc.edu/~qyang6/publications/pami-08-qingxiong-yang.pdf (Abrufdatum: 2012-03-26). [Yang2009]

YANG, QINGXIONG ; YANG, RUIGANG ; DAVIS, JAMES ET AL.: Spatial-Depth Super Resolution for Range Images. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. Online: http://vis.uky.edu/~liiton/publications/cvpr-07-qingxiong-yang.pdf (Abrufdatum:). [Yang2007]

YOON, KUK-JIN ; KWEON, IN-SO: Locally Adaptive Support-Weight Approach for Visual Correspondence Search. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. Online: http://rcv.kaist.ac.kr/~kjyoon/CVPR05_stereo.pdf (Abrufdatum: 2012-08-30). [Yoon2005]

YOON, KUK-JIN ; KWEON, IN-SO: Adaptive Support-Weight Approach for Correspondence Search. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 28 (2006), S. 650-656. Online: http://rcv.kaist.ac.kr/~kjyoon/PAMI2006_stereo.pdf (Abrufdatum: 2012-09-06). [Yoon2006]

ZHU, KE ; D'ANGELO, PABLO ; BUTENUTH, MATTHIAS: A Performance Study on Different Stereo Matching Costs Using Airborne Image Sequences and Satellite Images. In: STILLA, UWE et al. (Hrsg.): *Photogrammetric Image Analysis*. Berlin ; Heidelberg : Springer, 2011 (Lecture Notes in Computer Science ; 6952), S. 159-170. Online: http://dx.doi.org/10.1007/978-3-642-24393-6_14 (Abrufdatum: 2012-08-30). [Zhu2011]

ZITNICK, C. LAWRENCE ; KANG, SING BING: Stereo for Image-Based Rendering using Image Over-Segmentation. In: *International Journal of Computer Vision (IJCV)* 75 (2007), S. 49-65. Online: http://research.microsoft.com/~larryz/ZitnickKangIJCV07.pdf (Abrufdatum: 2012-09-17). [Zitnick2007]

Anhang 1: Bildmaterial

Abbildung 12: Linkes und rechtes Bild sowie „Ground Truth“ zum linken Bild des „Venus“-Bildpaars

Abbildung 13: Linkes und rechtes Bild sowie „Ground Truth“ zum linken Bild des „Teddy“-Bildpaars

Abbildung 14: Linkes und rechtes Bild sowie „Ground Truth“ zum linken Bild des „Cones“-Bildpaars

Abbildung 15: Linkes und rechtes Bild in der ersten Beleuchtungsvariante, linkes Bild in der zweiten Beleuchtungsvariante sowie rechte „Ground Truth“ der Szene „Sokrates Off-Axis“

Abbildung 16: Linkes und rechtes Bild in der ersten Beleuchtungsvariante, linkes Bild in der zweiten Beleuchtungsvariante sowie rechte „Ground Truth“ der Szene „Raum Off-Axis“

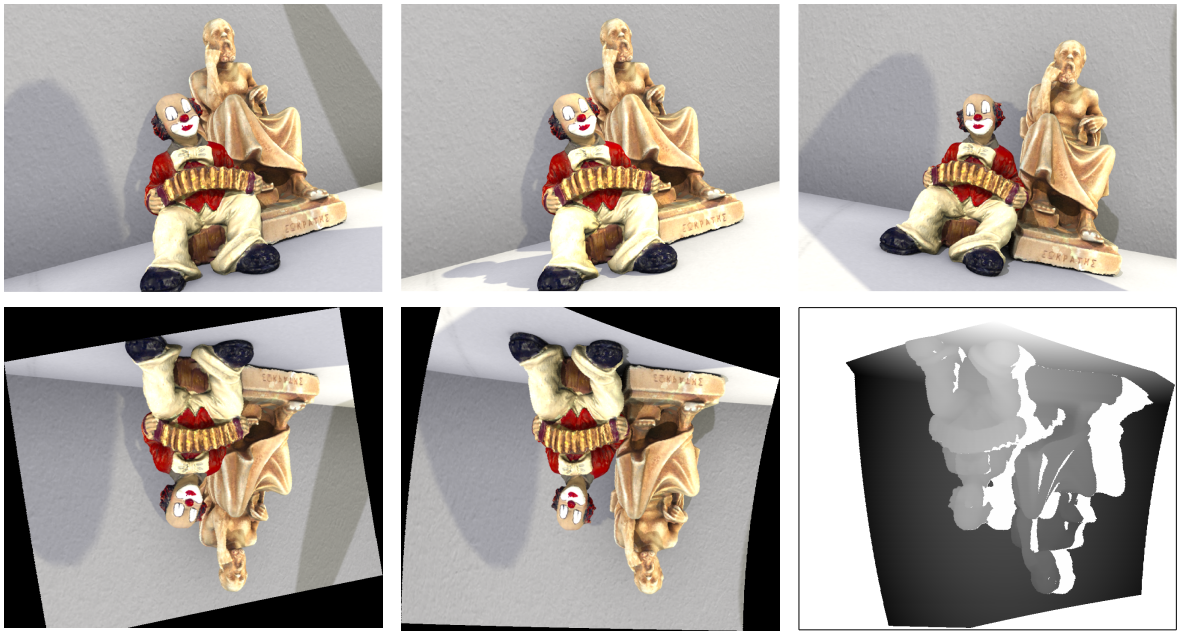


Abbildung 17: Linkes Bild in der ersten und zweiten Beleuchtungsvariante, rechtes Bild, rektifiziertes linkes und rechtes Bild sowie rechte „Ground Truth“ der Szene „Clown-Sokrates“

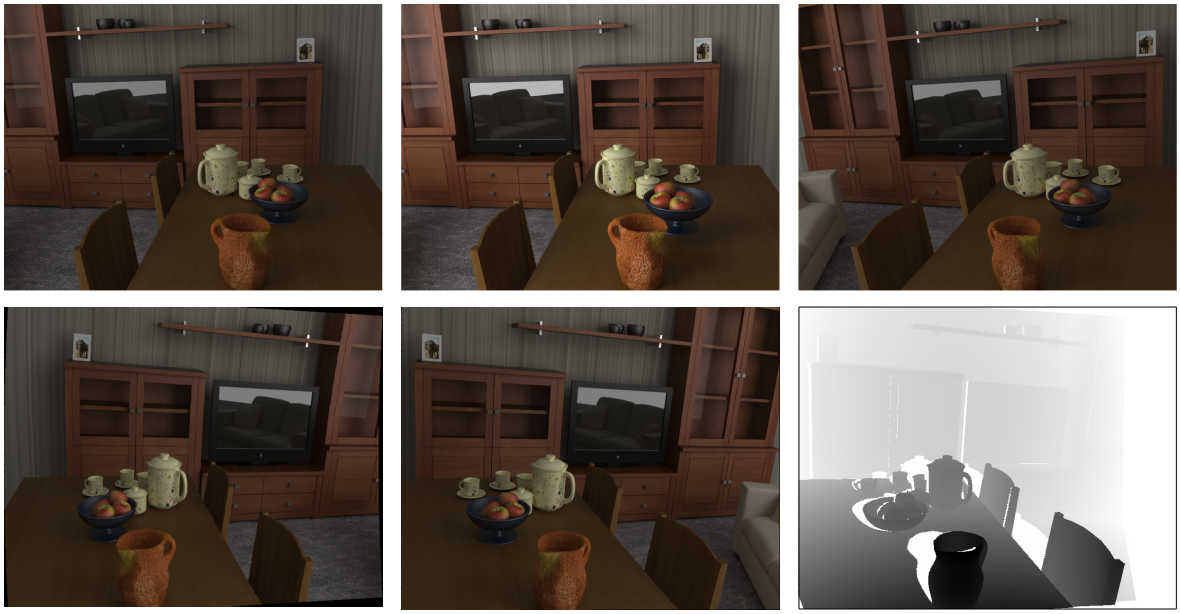


Abbildung 18: Linkes Bild in der ersten und zweiten Beleuchtungsvariante, rechtes Bild, rektifiziertes linkes und rechtes Bild sowie rechte „Ground Truth“ der Szene „Raum Poltergeist“

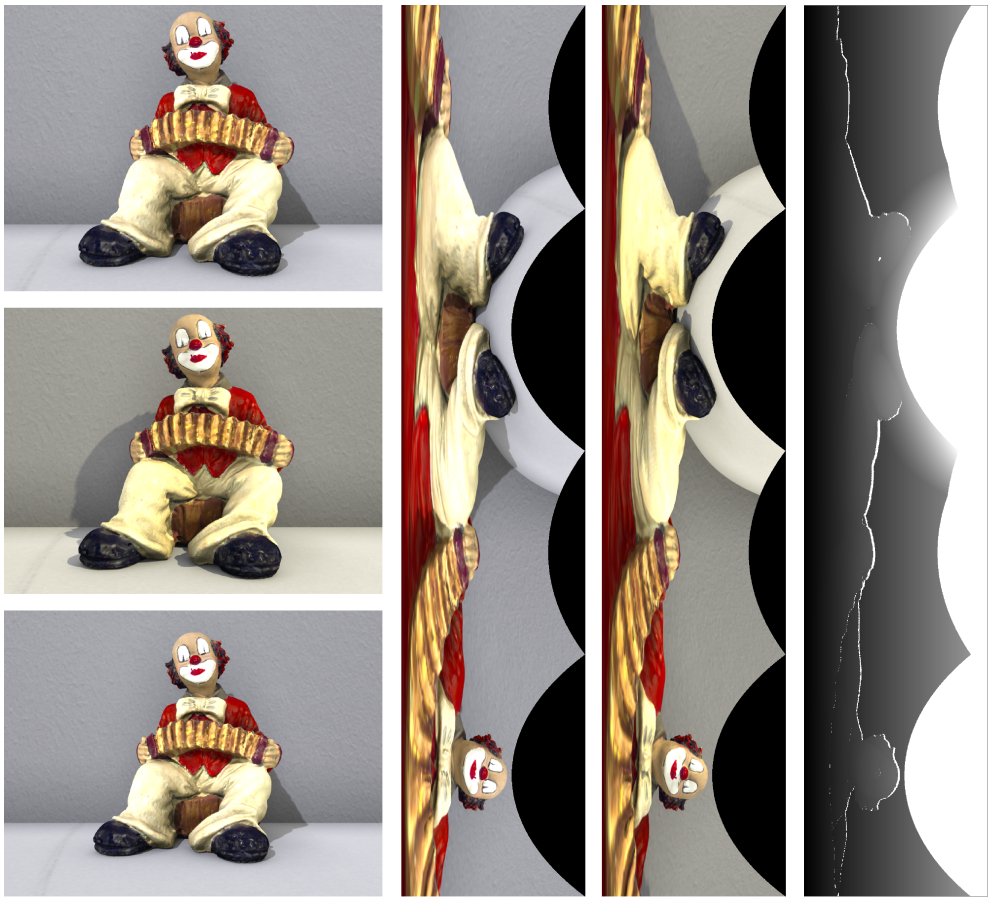


Abbildung 19: „Linkes“ Bild in der ersten und zweiten Beleuchtungsvariante, „rechtes“ Bild, rektifiziertes „linkes“ und „rechtes“ Bild sowie „ground truth“ zum „rechten“ Bild der Szene „Clown-in-front“

Anhang 2: Angegebene Laufzeit der Algorithmen in der Middlebury-Evaluation

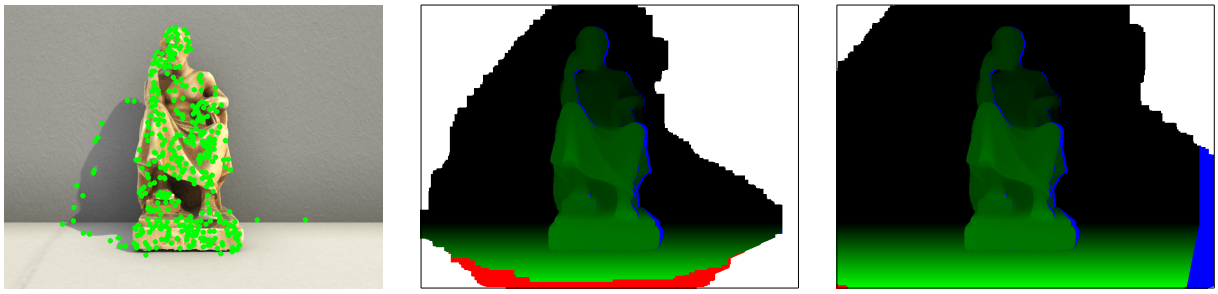
Die Laufzeit wird in der Tabelle so angegeben wie in der Studie, d. h. es findet keine Form der Normalisierung (die beispielsweise unterschiedliche Prozessortypen berücksichtigen müsste) statt. Die Angaben können daher nur bedingt miteinander verglichen werden, geben aber eine erste Orientierung bezüglich der Laufzeit der Algorithmen. die Farbcodierung gibt – nach dem Ampelsystem – Auskunft darüber, inwieweit die jeweilige Laufzeit für die für diese Arbeit relevanten Anforderungen ausreichend sein könnte[143].

Algorithmus	**Literaturverweis**	**% Fehlerpixel**	**Zeitangabe**
ADCensus [94]	[Mei2011]	3.97	2,5s (Tsukuba) 15s (Teddy, Cones); <0,1s (GPU)
SurfaceStereo [79]	[Bleyer2010b]	4.06	ca. 1h (!)
DoubleBP [35]	[Yang2009]	4.19	ca. 1min (Tsukuba)
AdaptingBP [17]	[Klaus2006]	4.23	14-25s (Tsukuba, Venus, Teddy, Cones)
SubPixDoubleBP [30]	[Yang2007]	4.39	keine Angabe
CoopRegion [41]	[Wang2008]	4.41	20s (Tsukuba)
ObjectStereo [98]	[Bleyer2011b]	4.46	20min
GC+SegmBorder [57]		4.52	nicht verfügbar
RDP [102]	[Sun2011]	4.57	2,5s (Tsukuba), 3,8s (Venus), 8,7/8,6s (Teddy, Cones)
PatchMatch [112]	[Bleyer2011a]	4.59	ca. 1min
OutlierConf [42]	[Xu2008]	4.60	keine Angabe
RVbased [116]		4.88	nicht verfügbar
WarpMat [55]	[Bleyer2009]	4.98	keine Angabe
Undr+OvrSeg [48]		5.39	nicht verfügbar
Segm+visib [4]	[Bleyer2004]	5.40	keine Angabe
BSM [117]		5.42	nicht verfügbar
GeoDif [103]		5.49	nicht verfügbar
InfoPermeable [109]	[Çığla2011], [Çığla2012]	5.51	<1s
CostFilter [95]	[Hosni2011], [Rhemann2011]	5.55	65ms – GPU-based
AdaptOvrSegBP [33]	[Taguchi2008]	5.59	90s (Tsukuba), 20min (Cones)
GlobalGCP [104]	[Wang2011]	5.60	130s (Teddy)
P-LinearS [99]	[De-Maeztu2011b]	5.68	33s (Tsukuba), 144s (Teddy)
FeatureGC [107]		5.72	nicht verfügbar
ConfSuppWin [113]		5.75	nicht verfügbar
CurveletSupWgt [73]		5.75	nicht verfügbar

143 Da die Erhebung im Februar 2012 durchgeführt wurde, fehlen Beiträge, die erst danach auf der Middlebury-Evaluationsseite veröffentlicht wurden. Die Reihung erfolgt hier – anders als in der Middlebury-Tabelle – nach der Prozentzahl der Fehlerpixel, weil sich diese nicht ändert, während sich der in der Middlebury-Tabelle verwendet durchschnittliche Rang jederzeit ändern kann (und sich auch ständig ändert), wenn der Tabelle ein Algorithmus hinzugefügt wird.

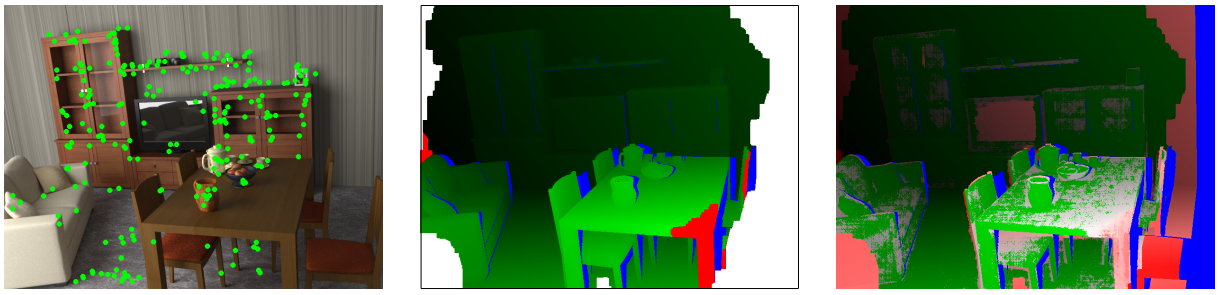
Algorithmus	**Literaturverweis**	**% Fehlerpixel**	**Zeitangabe**
C-SemiGlob [19]	[Hirschmüller2006]	5.76	„a few seconds“ (2006); 1-2s (2008)
ASSM [97]		5.77	nicht verfügbar
PlaneFitBP [32]	[Yang2008]	5.78	1s (512x384) – GPU-based
GeoSup [64]	[Hosni2009]	5.80	ca. 1min
SymBP+occ [7]	[Sun2005]	5.92	45s (Tsukuba)
SO+borders [29]	[Mattoccia2007]	6.03	„some minutes“
MultiResGC [49]		6.04	nicht verfügbar
iFBS [115]	[De-Maeztu2011a]	6.05	18,2s (Teddy)
IterAdaptWgt [105]	[Psota2011]	6.08	keine Angabe
AdaptDispCalib [36]		6.10	nicht verfügbar
OverSegmBP [26]	[Zitnick2007]	6.11	50s
DistinctSM [27]		6.14	nicht verfügbar
CostAggr+occ [39]	[Min2008]	6.20	9,8s (320x240); 42,5s (640x480)
RTAdaptWgt [114]		6.20	nicht verfügbar
VSW [108]		6.29	nicht verfügbar
LocallyConsist [69]	[Mattoccia2009]	6.33	13s (Tsukuba); 37s (Teddy) – „unoptimized“
MVSegBP [66]		6.34	nicht verfügbar
SegmentSupport [28]	[Tombari2007]	6.44	keine Angabe
RandomVote [89]	[Gales2010]	6.53	1s
GradAdaptWgt [60]		6.55	nicht verfügbar
RT-ColorAW [106]		6.55	nicht verfügbar
RegionTreeDP [18]		6.56	nicht verfügbar
PUTv3 [63]	[Stankiewicz2008]	6.64	keine Angabe
AdaptWeight [12]	[Yoon2005], [Yoon2006]	6.67	ca. 1min (Tsukuba)
EnhancedBP [24]	[Larsen2007]	6.69	4min (1024x768)
SegTreeDP [22]	[Deng2006]	6.82	<1s
MultiCue [51]		6.89	nicht verfügbar
ImproveSubPix [25]		6.90	nicht verfügbar
InteriorPtLP [34]	[Bhusnurmath2008]	7.26	9min
BP+DirectedDiff [61]	[Banno2009]	7.29	keine Angabe
FastBilateral [68]	[Mattoccia2010]	7.31	32s (Teddy)
HistoAggr [111]	[Min2011]	7.33	<=6s (Tsukuba); <=30s (Teddy)
SemiGlob [6]	[Hirschmüller2005]	7.50	1,3s (Teddy)
BPcompressed [56]		7.53	nicht verfügbar
VariableCross [44]		7.60	nicht verfügbar
RealtimeBFV [65]		7.65	nicht verfügbar
CostRelaxAW [59]	[Brockers2009]	7.66	20s bzw. 11,5s (Tsukuba)
RealtimeBP [21]	[Yang2006]	7.69	3,6-7,7s (Tsukuba); real-time (GPU-based)
2OP+occ [37]	[Woodford2008]	7.75	keine Angabe
RealTimeABW [81]		7.90	nicht verfügbar

Anhang 3: Automatische Abschätzung des Disparitätssuchraums



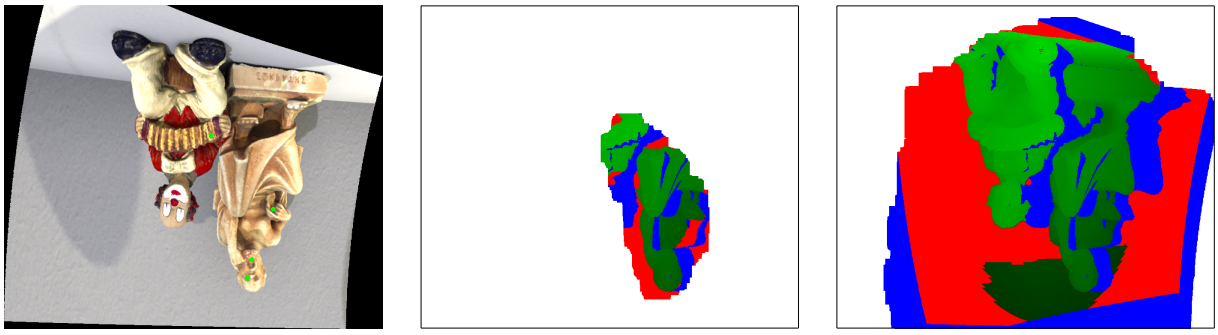
(a) (b) (c)

Abbildung 20: Abschätzung des Disparitätssuchraums bei der Szene „Sokrates Off-Axis“: (a) zur Initialisierung verwendete Features; (b)–(c) gefundene Disparitätssuchräume in der ersten und zweiten Runde; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte Bildregionen.



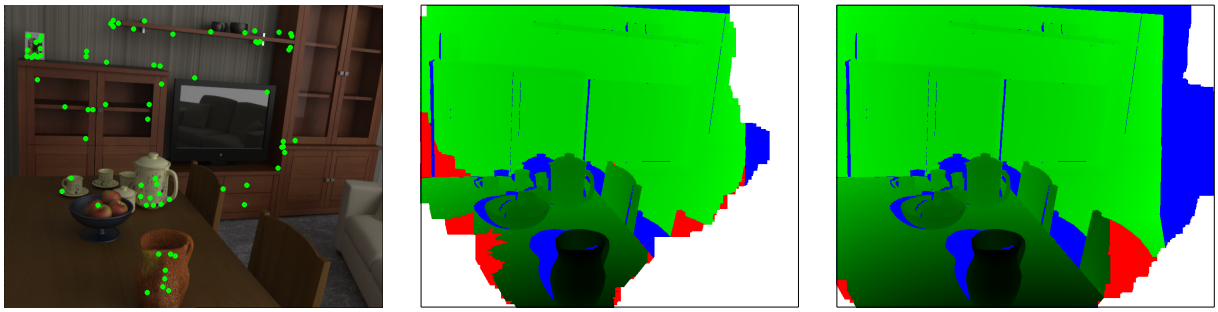
(a) (b) (c)

Abbildung 21: Abschätzung des Disparitätssuchraums bei der Szene „Raum Off-Axis“: (a) zur Initialisierung verwendete Features; (b) gefundene Disparitätssuchräume; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte Bildregionen; (c) Abweichungen der gerundenen Disparitäten gegenüber den korrekten Disparitäten: je kräftiger der Rotton, desto größer ist die Abweichung; gut zu sehen ist, dass die Disparitätssuche nicht erfolgreich sein kann, wo der Disparitätssuchraum falsch oder gar nicht eingeschätzt wurde.



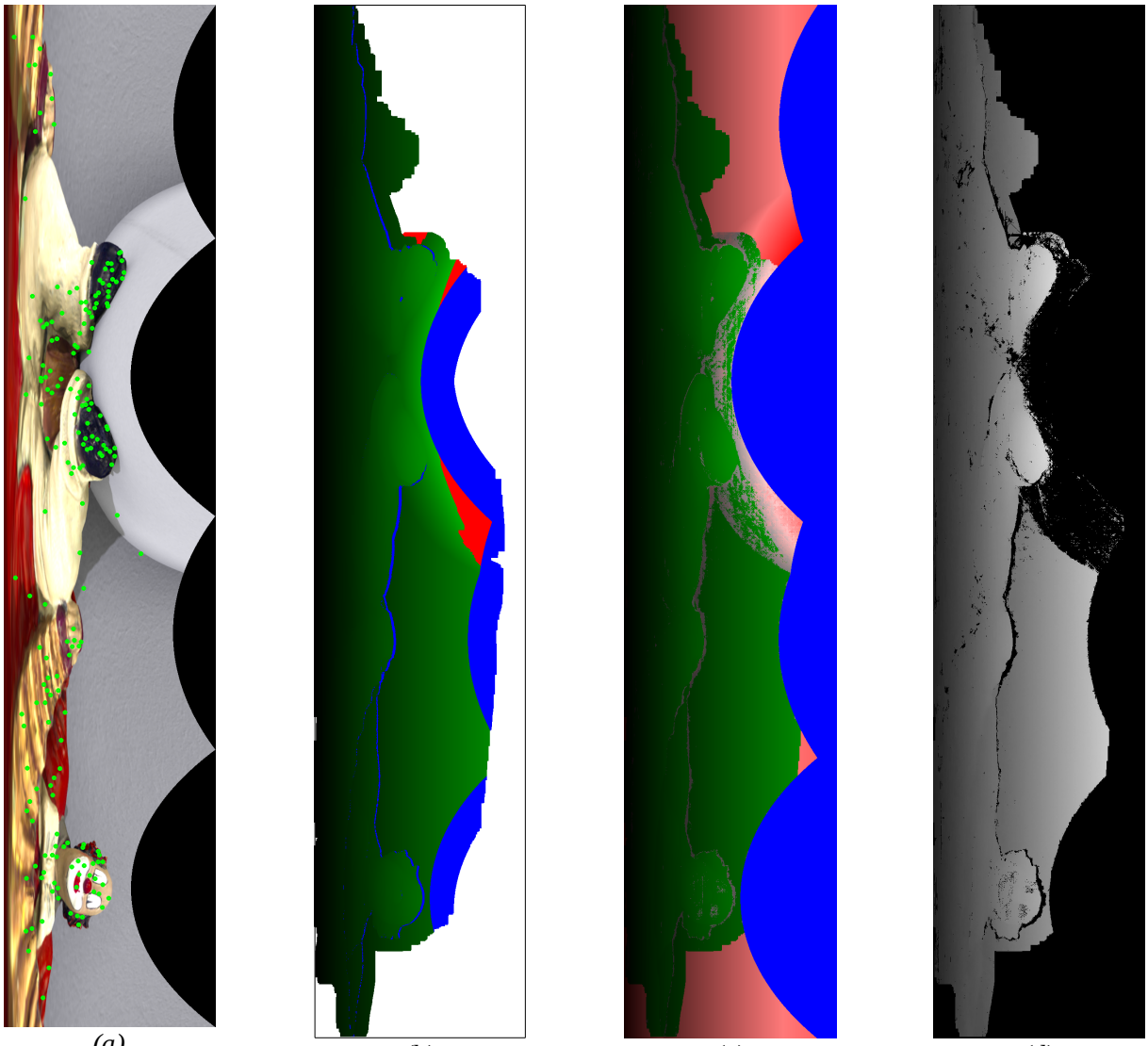
(a) (b) (c)

Abbildung 22: Abschätzung des Disparitätssuchraums bei der Szene „Clown-Sokrates“: (a) zur Initialisierung verwendete Features (4 Features: an Auge, Bart und Hand der Sokrates-Figur und an der Ziehharmonika); (b)–(c) gefundene Disparitätssuchräume in der ersten und dritten Runde; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte Bildregionen. Deutlich zu sehen ist, dass die Suchraumeinschätzung aufgrund der sehr niedrigen Zahl der Features an den Diskontinuitäten versagt; zu beachten ist allerdings, dass in diesem Fall auch eine automatische Rektifizierung der Bilder unmöglich wäre.



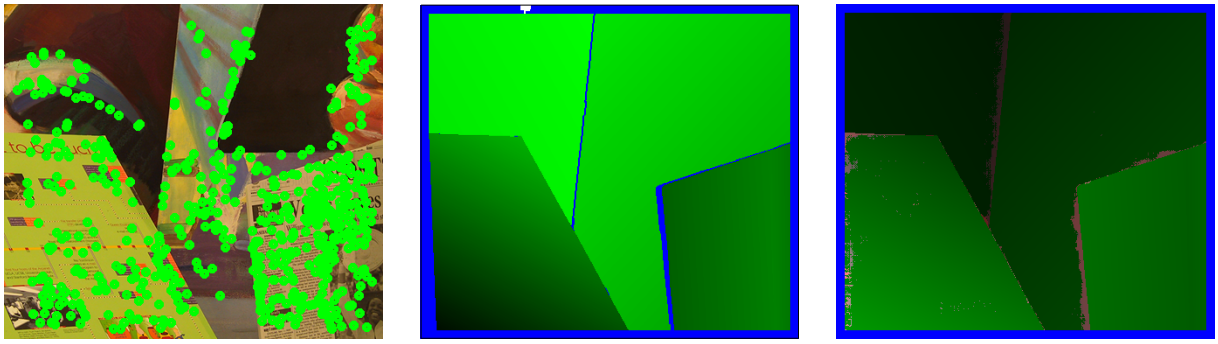
(a) (b) (c)

Abbildung 23: Abschätzung des Disparitätssuchraums bei der Szene „Raum Poltergeist“: (a) zur Initialisierung verwendete Features; (b)–(c) gefundene Disparitätssuchräume in der ersten und zweiten Runde; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte und ungültige Bildregionen. Schwierigkeiten hat der Algorithmus hier v. a. mit kontrastarmen, weil schattigen Bildbereichen.



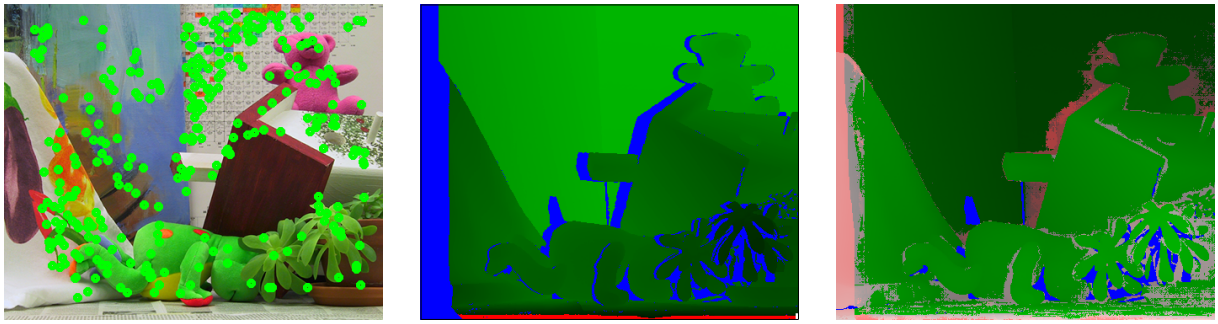
(a) (b) (c) (d)

Abbildung 24: Abschätzung des Disparitätssuchraums bei der Szene „Clown-in-front“: (a) zur Initialisierung verwendete Features; (b) gefundene Disparitätssuchräume; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte und ungültige Bildregionen; (c) Abweichungen der gerundenen Disparitäten gegenüber den korrekten Disparitäten: je kräftiger der Rotton, desto größer ist die Abweichung; (d) nicht-dichte Disparitätskarte mit den nur als korrekt vermuteten Disparitätswerten.



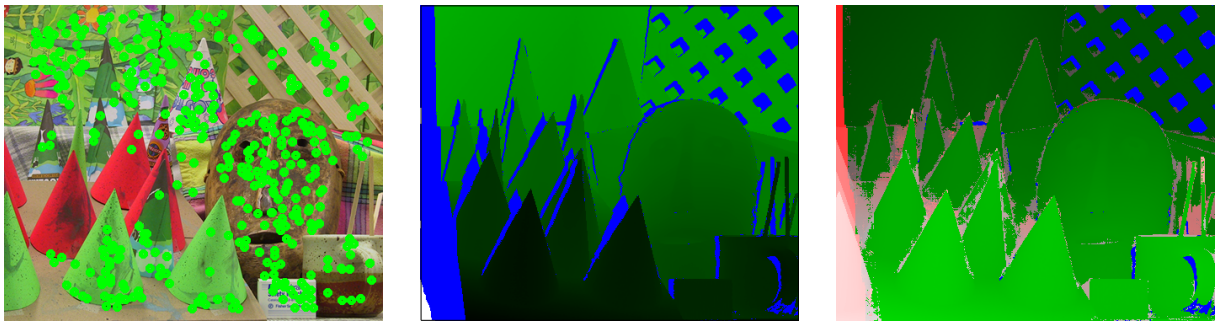
(a) (b) (c)

Abbildung 25: Abschätzung des Disparitätssuchraums bei der Szene „Venus“: (a) zur Initialisierung verwendete Features; (b) gefundene Disparitätssuchräume; blaue Bildbereiche kennzeichnen verdeckte Bildregionen; (c) Abweichungen der gerundenen Disparitäten gegenüber den korrekten Disparitäten: je kräftiger der Rotton, desto größer ist die Abweichung.



(a) (b) (c)

Abbildung 26: Abschätzung des Disparitätssuchraums bei der Szene „Teddy“: (a) zur Initialisierung verwendete Features; (b) gefundene Disparitätssuchräume; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte Bildregionen; (c) Abweichungen der gerundenen Disparitäten gegenüber den korrekten Disparitäten: je kräftiger der Rotton, desto größer ist die Abweichung.



(a) (b) (c)

Abbildung 27: Abschätzung des Disparitätssuchraums bei der Szene „Cones“: (a) zur Initialisierung verwendete Features; (b) gefundene Disparitätssuchräume; in den rot markierten Bildbereichen liegt die korrekte Disparität außerhalb des vermuteten Disparitätssuchraums; blaue Bildbereiche kennzeichnen verdeckte Bildregionen; (c) Abweichungen der gerundenen Disparitäten gegenüber den korrekten Disparitäten: je kräftiger der Rotton, desto größer ist die Abweichung.

Anhang 4: Qualität der Ergebnisse

Schwellenwert: 1,0	**nonocc**[144]	**„all“**	**disc**	**trusted**
Venus	1,40 %	2,47 %	15,09 %	1,37 %
Teddy	8,37 %	16,37 %	20,42 %	7,54 %
Cones	4,29 %	12,31 %	12,68 %	4,90 %
Sokrates Off-Axis	2,45 %	2,71 %		1,07 %
Raum Off-Axis	20,93 %	23,05 %		12,19 %
Clown-Sokrates	19,96 %	28,50 %		7,29 %
Raum Poltergeist	32,51 %	35,05 %		15,48 %
Clown-in-front	5,32 %	4,60 %		1,56 %

Schwellenwert: 0,5	**nonocc**	**„all“**	**disc**	**trusted**
Venus	5,25 %	6,51 %	20,43 %	5,10 %
Teddy	14,76 %	23,12 %	33,00 %	13,12 %
Cones	7,50 %	15,91 %	19,90 %	7,58 %

Prozentwerte laut Middlebury-Evaluation[145]:

Schwellenwert: 1,0	**nonocc**	**„all“**	**disc**
Venus	1,31 %	2,37 %	14,8 %
Teddy	7,78 %	15,4 %	19,1 %
Cones	4,09 %	12,0 %	12,2 %

Schwellenwert: 0,5	**nonocc**	**„all“**	**disc**
Venus	3,92 %	5,16 %	18,4 %
Teddy	11,7 %	20,2 %	27,9 %
Cones	5,77 %	14,1 %	16,5 %

144 Die Abkürzungen bedeuten: „nonocc“: nichtverdeckte Pixel; „all“: alle gültigen Pixel; „disc“: Pixel in der Nähe der Diskontinuitäten; „trusted“: durch den Algorithmus als korrekt markierte Pixel. Die Prozentzahl gibt an, wieviele Pixel der genannten Gesamtmenge einen Disparitätswert aufweisen, der den Schwellenwert überschreitet. Die Datensätze des Lehrgebiets Mensch-Computer-Interaktion enthalten keine Masken für die Pixel in der Nähe der Disparitäten.

145 Leider kann ich nicht nachvollziehen, warum die von der Middlebury-Evaluationsseite ermittelten Prozentwerte niedriger sind als die von meiner Referenzimplementierung berechneten. Möglicherweise hängt der Unterschied damit zusammen, dass die Middlebury-Evaluation Werte im Wertebereich [0, 255] skaliert, während meine Referenzimplementierung die Disparitäten per Interpolation in Gleitkommaarithmetik berechnet. Vielleicht wird aber auch in der Middlebury-Evaluation die Gesamtheit der Pixel anders bestimmt. Da die nackten Zahlen sowieso nur begrenzte Aussagekraft haben und eine gute Platzierung in der Middlebury-Evaluation kein primäres Ziel dieser Arbeit ist, bin ich diesem merkwürdigen Umstand nicht weiter nachgegangen.

Anhang 5: Disparitätskarten

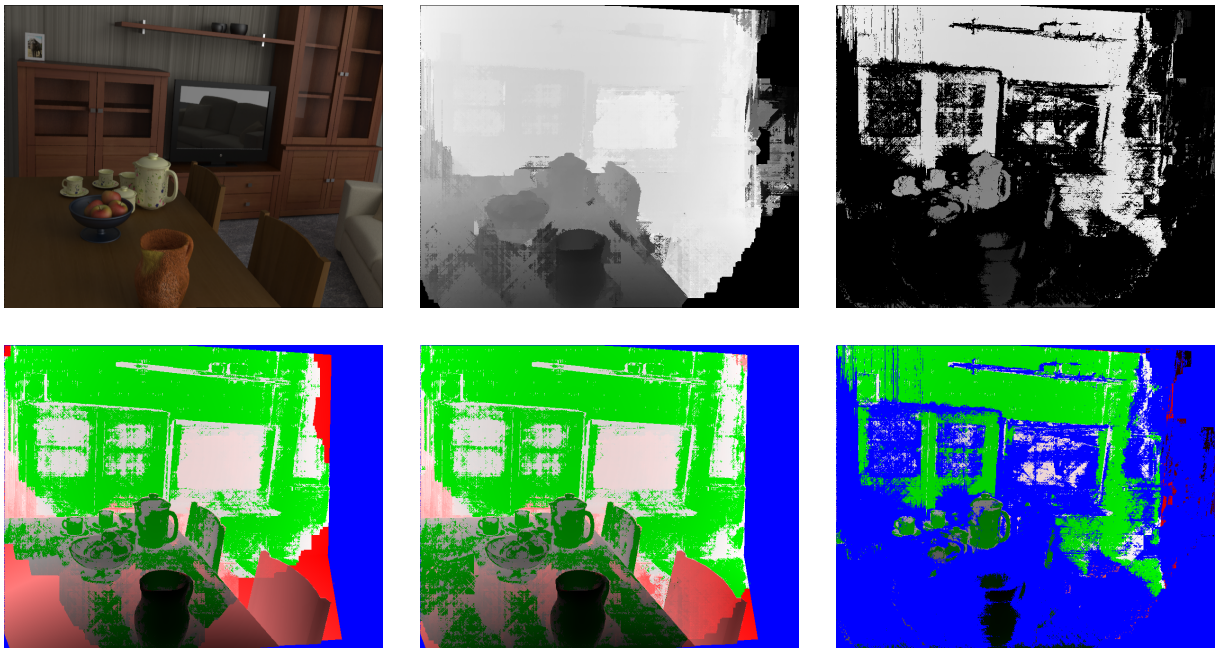


Abbildung 28: Disparitätskarten der Szene „Raum Poltergeist“: Basisbild, Disparitätskarte aller Pixel und Disparitätskarte der als vertrauenswürdig eingestuften Pixel (oben); Differenz zur „Ground Truth“ aller Pixel nach dem ersten Durchgang und nach dem zweiten Durchgang der Disparitätsberechnung (unten links und in der Mitte); Differenz der als vertrauenswürdig eingestuften Pixel zur „Ground Truth“ (unten rechts); grüne Bildbereiche bedeuten, dass sich die berechnete Disparität innerhalb der Toleranzgrenze (1,0) befindet; rote Bildbereiche kennzeichnen falsch berechnete Disparitätswerte, wobei der Grad der Rotfärbung über das Ausmaß der Differenz Auskunft gibt; blaue Bildbereiche werden nicht in die Evaluation einbezogen (Pixel ohne Entsprechung im Basisbild; Bildpunkte, deren Entsprechung außerhalb des Referenzbilds liegen würde; als unsicher markierte Pixel).

Abbildung 29: Disparitätskarten der Szene „Teddy“: Disparitätskarte aller Pixel und Disparitätskarte der als vertrauenswürdig eingestuften Pixel und Differenz zur „Ground Truth“; zur Farbcodierung siehe Abb. 28.

Anhang 6: Anzahl der Fehlerpixel bei verschiedenen Beleuchtungssituationen und Bildrauschen

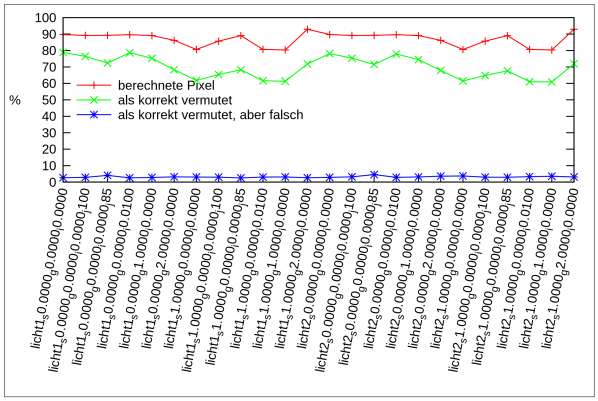


Abbildung 30: Fehlerpixelwerte der Szene „Sokrates Off-Axis“

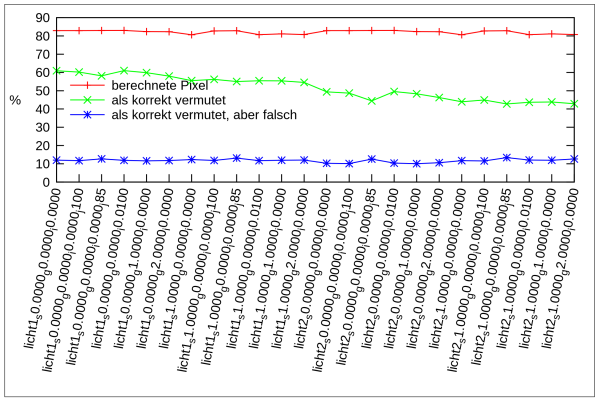


Abbildung 31: Fehlerpixelwerte der Szene „Raum Off-Axis“

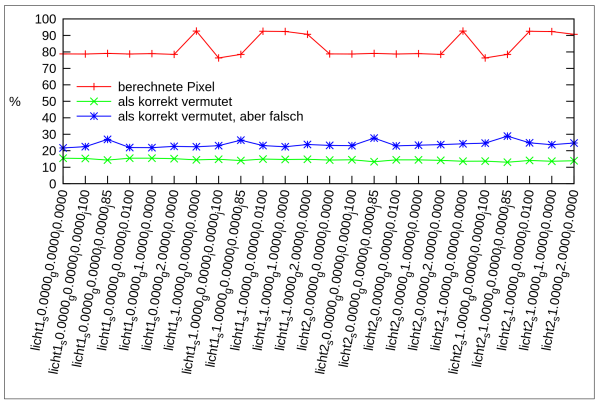


Abbildung 32: Fehlerpixelwerte der Szene „Clown-Sokrates“

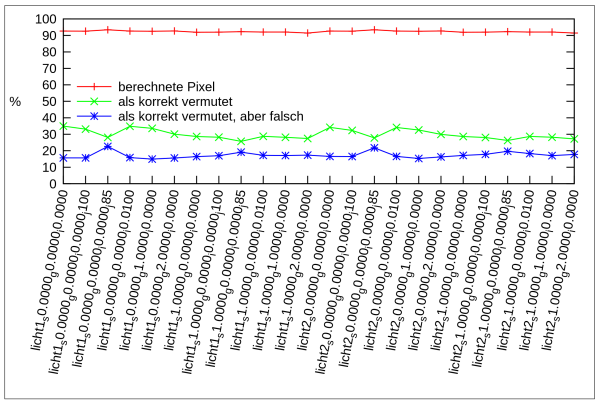


Abbildung 33: Fehlerpixelwerte der Szene „Raum Poltergeist“

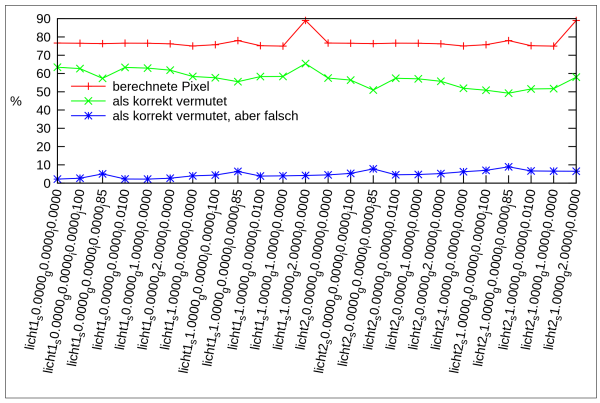


Abbildung 34: Fehlerpixelwerte der Szene „Clown-in-front“