Gewinnung eines Geländeprofils mittels Sensordaten und Bewegungsplanung

Bachelorarbeit

Institut für Robotik der Fakultät für Informatik
an der
Hochschule Mannheim

hochschule mannheim

vorgelegt von:

Claude Heischbourg

Datum: 21. März 2007
Betreuer: Dr. Prof. Thomas Ihme
Zweitkorrektor: Dr. Prof. Elena Fimmel

Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit hat in dieser oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegen.

Mannheim, den 21. März 2007

Heischbourg Claude

Danksagung

An dieser Stelle möchte ich mich bei Herrn Thomas Ihme bedanken für die Unterstützung und Betreuung während der Bachelorarbeit.

Dank gebührt auch meiner Familie für die moralische und finanzielle Unterstützung während meines Studiums.

Heischbourg Claude Mannheim, den 21. März 2007

Zusammenfassung:

Um für sechsbeinige Roboter bessere Laufbewegungen zu erreichen, hat André Herms einen Laufplanungsalgorithmus implementiert, der die Fußauftrittspunkte plant. Dazu verwendet er Geländeinformationen die in Form von einer Höhenkarte vorliegen. Um diese Höhenkarte der realen Welt zu berechnen werden Sensorinformationen benötigt welche mit Hilfe eines Stereokamerakopfes, der am Roboter befestigt ist, gewonnen werden. Durch Anwendung von Stereobildverarbeitungsalgorithmen auf diese Informationen in Form von Stereobildpaaren kann man eine Tiefenkarte erstellen. In dieser Arbeit werden geeignete Stereobildverarbeitungsalgorithmen vorgestellt und miteinander nach Qualität und Rechenzeit verglichen. Das Graph Cut Verfahren sowie das Blockmatching Verfahren werden in einer engeren Auswahl auf verschiedene Testszenarien angewendet und auf Echtzeitfähigkeit untersucht. Dabei kommt man zum Ergebnis dass der Blockmatching Stereobild-verarbeitungsalgorithmus am geeignetsten für die Erstellung des Geländeprofils für sechsbeinige mobile autonome Roboter ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung:

Autonome Mobile Roboter, das heißt Roboter die sich in ihrer Umgebung selbstständig bewegen und agieren können, gewinnen in der heutigen Zeit immer mehr an Bedeutung. Das liegt vor allem daran, dass sie in Umgebungen eingesetzt werden können, in denen der Mensch schlechten oder gar keinen Zugang hat. Beispiele hierfür sind Roboter, die bei Bergungsarbeiten nach Erdbebenkatastrophen eingesetzt werden, oder Roboter die nach Kernkraftunglücken in das verseuchte Gebiet vordringen können uwm. Aber auch im alltäglichen Gebrauch kommen mobile Roboter immer mehr auf. So gibt es zum Beispiel schon Roboter die den Rasen mähen [11] oder in einigen Krankenhäusern rollen autonome Wagen durch die Gänge um Transportaufgaben zu übernehmen (siehe Abbildung 1) [15].

Abbildung 1: Transportroboter

Um ihre Aufgaben wahrnehmen zu können, müssen Mobile Roboter die Umgebung, sowie die Position an der sie sich befinden, kennen. Die benötigten Informationen erhalten sie hierbei abgeleitet durch einzelne Sensoren oder durch mehrere Sensoren (Sensorfusion). Man unterscheidet zwischen aktiven Sensoren (Laserscanner, Ultraschall) und passiven Sensoren (Kraftsensoren, Kamera).

Ferner gibt es verschiedene Arten, wie ein Mobiler Roboter sich auf dem Land fortbewegen kann. Neben der rad- und kettengetriebenen Fortbewegung, dem Kriechen oder Schlängeln usw, kann der Roboter sich auch schreitend mit Hilfe von Beinen fortbewegen. Als besonders vorteilhaft hat sich dabei die schreitende Fortbewegung mit 6 Beinen herausgestellt, da bei dieser das Balancieren entfallen kann, solange mindestens 3 Beine festen Halt haben [28]. Bei mehr Beinen steigt die Komplexität der Koordination rapide an. Außerdem ermöglicht sie das Manövrieren in sehr zerklüftetem Gelände, wo ein Fahrzeug mit Rädern zum Beispiel stecken bleiben würde. Schlussendlich ist diese Art der Fortbewegung viel weniger belastend für den Untergrund, und eine Arbeitsmaschine mit sechs Beinen würde zum Beispiel bei Waldarbeiten viel weniger Schäden an der Umwelt hinterlassen als ein rad- oder kettengetriebenes Arbeitsgerät (siehe Abbildung 2).

Abbildung 2: Waldroboter Plusjack von der finnischen Firma Plustech Oy Ltd

1.1 Motivation

Am Institut für Robotik an der Hochschule Mannheim werden Forschungsarbeiten zum Thema Technische Operationen mit Laufrobotern durchgeführt. Zu diesem Zweck steht ein Laufroboter LAURON der Version IVb zur Verfügung.

Ursprünglich wurde dieser Roboter vom Forschungszentrum für Informatik an der Universität Karlsruhe entwickelt wobei LAURON für **LAU**fender **RO**boter **N**euronal gesteuert steht, auch wenn die Steuerung schon längst nicht mehr neuronal ist.

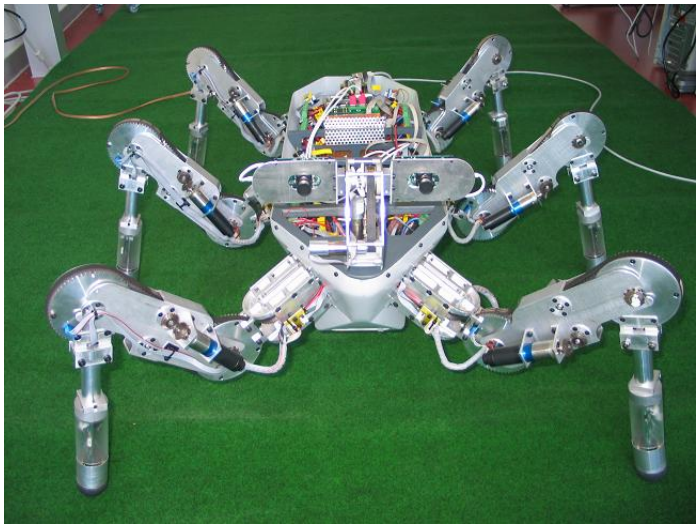

Abbildung 3: LAURON-Roboter

In vorhergehenden Arbeiten wurden Laufalgorithmen entwickelt, die es dem LAURON Roboter ermöglichen sollen, sich selbstständig fortzubewegen. Die verwendeten Algorithmen zur Beinsteuerung sind bisher meist reaktiv [8] [9], d.h. sie versuchen zunächst, einen Schritt auszuführen und reagieren dann auf die vorgefundene Situation (Hindernis, Loch). Diese Strategie führt dann zu Problemen, wenn die Reaktion auf solche Situationen die eigentliche Fortbewegung dominiert. Um dieses Problem zu verbessern, wurde ein Laufplaner, zunächst

in einer eigens entwickelten Simulationsumgebung, realisiert. Wie sich zeigte kann das Laufverhalten dadurch positiv beeinflusst werden. [10]

In der Arbeit von Uli Ruffler [24] wurde der Laufplanungsalgorithmus verbessert, indem er die Fußauftrittspunkte des Roboters inkrementell generieren lässt. Dies ist nötig, da eine weiter entfernte Geländeinformation unsicherer ist als eine nahe liegende, und somit eventuell Fehler enthält, die eine Neuplanung im Verlauf des Schreitens zum Ziel erzwingen. Außerdem sollte der Laufplanungsalgorithmus während des Schreitens arbeiten, um einen flüssigen Lauf zu ermöglichen.

Damit der Algorithmus die Positionen der Fußauftrittspunkte während des Schreitens bestimmen kann, muss dieser das Gelände und die Position des Roboters in diesem kennen. In der Simulationsumgebung bereitet dies keine Schwierigkeiten, da das Gelände bereits als Höhenkarte vorliegt. In der Realität muss LAURON seine Umgebung mittels Sensoren selbst erkunden um seine Position zu bestimmen. Zu diesem Zweck ist der Roboter mit zwei Kameras ausgestattet, die es ihm ermöglichen sollen räumlich zu sehen. Entsprechende Stereobildverarbeitungsalgorithmen gibt es bereits [1] [17] [18].

Um dem langfristigen Ziel eines selbsterkundenden Laufroboter näher zu kommen, sollen die Ergebnisse der beiden Arbeitsbereiche miteinander verknüpft werden. Dazu bedarf es allerdings noch einiger Vorarbeiten, Verbesserungen und Untersuchungen.

1.2 Aufgabenstellung

In der Bildverarbeitung gibt es eine ganze Reihe von Verfahren, die sich verschiedener Ansätze bedienen, um mit Hilfe eines Stereokamerasystems räumlich sehen zu können. Sie ermöglichen es so genannte Tiefenkarten zu erstellen. Also Bilder in denen jeder einzelne Pixel einen bestimmten Tiefenwert hat. In dieser Arbeit sollen einige dieser Ansätze vorgestellt und diskutiert werden. Es handelt sich dabei aber nur um so genannte passive Verfahren, das heißt es muss mit dem vorhandenen Licht ausgekommen werden und es wird nicht extra ein Licht in den Raum projiziert um diesen damit besser erfassen zu können (c.f. Lasertriangulation [24]). Außerdem soll durch praktische Anwendung dieser Verfahren auf verschiedenen Testszenarien herausgefunden werden, welcher dieser Ansätze am besten für das LAURON Robotersystem geeignet ist. Dafür sollen Metriken aufgestellt werden um die einzelnen Stereobildverarbeitungsalgorithmen auf die Qualität ihrer Ergebnisse zu überprüfen. Ferner soll herausgefunden werden ob die Laufzeiten der Verfahren den Echtzeitanforderungen eines autonomen mobilen Robotersystems Genüge tun.

1.3 Ergebnisse der Arbeit

Als Ergebnis dieser Arbeit kommt heraus, dass sich aus einer ganzen Reihe von Verfahren, der so genannte Anytime-Blockmatching Stereobildverarbeitungsalgorithmus als am geeignetsten für das LAURON Robotersystem herausgestellt hat. Er erfüllt die erforderten Kriterien am Besten. Sprich erst einmal die Echtzeitfähigkeit des Verfahrens. In unseren Testszenarien lagen erste Ergebnisse bei einer Auflösung von 640×480 schon nach 0,4 Sekunden vor. Diese Ergebnisse sind zwar noch von minderer Qualität bzw Auflösung. Nach 1,23 Sekunden liegen aber schon Ergebnisse mit einer sehr guter Qualität vor. Das heißt da sich der LAURON Roboter sehr langsam bewegt reicht ein Bild alle 1,23 Sekunden locker aus um die Echtzeitanforderung zu erfüllen.

Das zweite Kriterium ist die, dass die Tiefenkarten möglichst dicht sein sollen, das heißt es sollen keine „Lücken“ darin enthalten sein. Zwar liefern die Szenarien die in einer Simulationsumgebung erstellt wurden, also von den Lichtverhältnissen her perfekt sind fast lückenlose Ergebnisse, bei den mit den Kameras vom LAURON Roboter selbst aufgenommen Bildern gibt es allerdings Probleme in verschiedenen Bereichen., vor allem bei texturarmen Regionen. Deshalb ist es empfehlenswert beim LAURON Robotersystem die Stereobildverarbeitung noch mit anderen Verfahren und Sensoren zu kombinieren. Außerdem könnte man bei der Laufplanung reaktive und statische Verfahren kombinieren. Dies würde man hinkriegen indem man das aus der Stereobildverarbeitung gewonnene Weltmodell für die Laufplanung verwendet. Wenn während der geplanten Fortbewegung allerdings nun ein übersehenes Hindernis auftritt, so kann ein reaktives Verfahren eingesetzt werden.

1.4 Aufbau der Arbeit

In Kapitel 2 werden einige Grundlagen erläutert die man für das Verstehen der Arbeit braucht. Dazu wird in Kapitel 2.1 zuerst auf den Aufbau des Laufroboter LAURON eingegangen und seine technischen Details werden vorgestellt. In Kapitel 2.2 wird dann die Stereobild-verarbeitung vorgestellt und einige wichtige Begriffe aus dem Bereich erklärt. Anschließend wird in Kapitel 2.3 auf die Laufplanung eines LAURON Roboters eingegangen, bevor in Kapitel 2.4 eine Simulationsumgebung mit darin implementiertem Laufplanungsalgorithmus für LAURON Roboter vorgestellt wird.

In Kapitel 3 werden einige verwandte Arbeiten zu Thema dieser Arbeit vorgestellt bevor man in Kapitel 4 zum Hauptteil übergeht. Hier wird in Kapitel 4.1 die Struktur der Steuerungssoftware des LAURON Roboters vorgestellt, also in welche Module sie aufgeteilt ist, und das für unsere Arbeit relevante Modul wird daraus extrahiert. In Kapitel 4.2 werden dann einige für unser Problem geeignete Stereobildverarbeitungsalgorithmen aufgelistet und durch Untersuchung verschiedener Kriterien werden die zwei geeignetsten in Kapitel 4.3 von ihnen ausgewählt. Diese beiden ausgewählten Verfahren, nämlich das Blockmatching Verfahren sowie das Graph Cut Verfahren, werden anschließend in den Kapiteln 5 und 6 genauer vorgestellt.

In Kapitel 7.1 wird dann in einzelnen Schritten das Erstellen von Testszenarien vorgestellt und auf was man dabei achten muss. In Kapitel 7.2 geht man auf Methoden ein die man benutzen kann um die Qualität der Stereobildverarbeitungsalgorithmen zu messen. In Kapitel 7.3 wird zum Schluss noch kurz auf die Implementierung eingegangen.

In Kapitel 8 wird dann die Auswertung vorgenommen. Zuerst werden in Kapitel 8.1 die einzelnen Szenarien vorgestellt bevor in Kapitel 8.2 die Ergebnisse des Blockmatching Verfahren und in Kapitel 8.3 die Ergebnisse des Graph Cut Verfahren, angewendet auf die verschiedenen Szenarien, vorgestellt werden. In Kapitel 8.4 wird dann der Erfassungsbereich der Stereobildverarbeitung diskutiert. Anschließend werden in Kapitel 8.5 das Graph Cut und das Blockmatching Verfahren miteinander verglichen und ein Fazit gezogen welches für das LAURON Roboter System am besten geeignet ist.

Zum Schluss wird in Kapitel 9 dann noch eine Zusammenfassung sowie ein Ausblick gegeben.

2. Grundlagen

In diesem Kapitel werden die Grundlagen, die man zum Verständnis der Arbeit braucht, erläutert. Dabei wird zuerst auf den Aufbau des LAURON Roboters eingegangen, also der Roboter der später einmal mittels seines Stereokamerasystem seine Umgebung selbst erkunden soll, und seine technischen Details werden vorgestellt. Anschließend wird das Konzept der Stereobildverarbeitung vorgestellt. Schlussendlich wird noch kurz auf die Laufplanung eingegangen und ein Laufplanungsalgorithmus vorgestellt der bereits in einer Simulationsumgebung läuft.

2.1 Der Laufroboter Lauron

Ursprünglich wurde der LAURON Roboter (siehe Abbildung 3) vom Forschungszentrum für Informatik an der Universität Karlsruhe entwickelt wobei LAURON für **LAU**fender **RO**boter **N**euronal gesteuert steht, auch wenn die Steuerung schon längst nicht mehr neuronal ist. Er wurde gebaut um Laufalgorithmen für unebenes Gelände zu erforschen. Er besteht aus einem Körper, in dem die Steuerungselektronik untergebracht ist, einem Kopf der drehbar auf den Körper montiert wurde und sechs identischen Beinen. Als biologisches Vorbild für den Roboter wurde die indische Stabheuschrecke genommen, dessen Gangart bereits sehr gut erforscht ist. Hinterleib und Segmentierung des Körpers wurde von seinem biologischen Vorbild nicht übernommen, da diese nur unwesentlich zum Laufverhalten beitragen. Jedes Bein hat drei Freiheitsgrade und somit einen Freiheitsgrad weniger als das Insekt (siehe Abbildung 4). Da die Stabheuschrecke diesen Freiheitsgrad nicht oder nur selten zum Laufen benutzt, wurde dieser bei der Maschine zur Reduktion der Komplexität der Beinkoordination vernachlässigt. Damit kommt der Roboter auf 18 Beinfreiheitsgrade (sechs Beine mit je drei Freiheitsgraden).

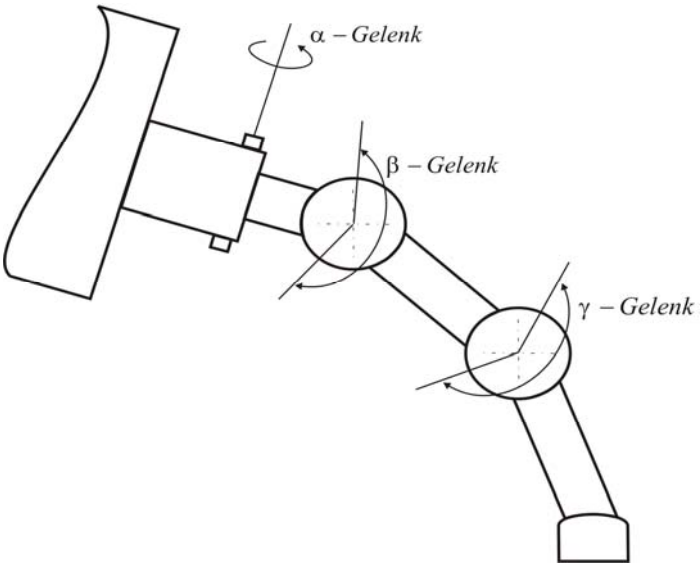

Abbildung 4: Bein eines Lauron Roboters

Jedes Bein verfügt über Kraftsensoren im letzten Glied, mit denen der Roboter entscheiden kann, ob das Bein den Untergrund berührt und wie viel Kraft das entsprechende Bein bei seiner Stützfunktion aufnimmt. Die Gelenke sind mit Winkelsensoren ausgestattet. Um Sensorsignale auswerten und Stellgrößen anlegen zu können, verarbeitet ein Digitaler-Signal-

Prozessor (DSP) je Bein (und einer für den Kopf) die Ein- und Ausgangssignale. Diese DSPs sind mittels CAN-Bus vernetzt und an den Hauptrechner, einen gewöhnlichen Industrie-PC (PC104), angeschlossen. Der Hauptrechner übernimmt die gesamte Laufsteuerung und wird mit einem Linux (Debian) System betrieben. Durch die Verwendung eines RT-Kernels erhält das Betriebssystem Echtzeitfähigkeiten.

Der drehbare Kopf des Roboters beherbergt zwei Digitalkameras, die per IEEE 1394 an den PC104 angeschlossen sind. Um die Blickrichtung zu ändern ist nicht nur der Kopf schwenkbar, sondern zusätzlich ist es möglich die Kameras zu neigen. [24] [28] Zum Abschluss werden in Abbildung 5 die technischen Details des LAURON Roboters tabellarisch aufgelistet. Wichtig für unsere Arbeit (bzgl Echtzeitfähigkeitskriterium) ist hier seine maximale theoretische Geschwindigkeit. Diese beträgt „nur“ 0,5 km/h. Das heißt der Roboter bewegt sich sehr langsam.

Basis Parameter	
Grundform	Stabheuschreckenartig
Anzahl der Beine	6
Aktive Bein-Freiheitsgrade (gesamt)	6*3 = 18
Max. theoretische Geschwindigkeit ca.	0,5 km/h
Gewicht (Grundkonfiguration)	19 kg
Nutzlast ca.	10 kg
Körper	
Grundkörperabmessungen	
Höhe ca	19 cm
Seitenlänge ca	72 cm
Breite ca	24 cm
Sensoren	
Neigungssensoren	1
Beine	
Aktive Freiheitsgrade	3
Arbeitsbereiche der Gelenke	
α - Gelenk	-60° .. + 60°
β - Gelenk	-30° .. + 60°
γ - Gelenk	-30° .. + 60°
Antriebe	Gleichstrommotoren (24V)
Abmessungen eines Beines	
Gesamt	64,5 cm
A-Segment	7 cm
B-Segment	7,5 cm
C-Segment	20 cm
D-Segment	30 cm
Masse eines Beines ca.	2,2 kg
Sensoren	
Gelenkwinkelsensoren	3 optische Inkrementalsensoren pro Beingelenk
Fußkraftsensoren	3 orthogonale Komponenten im Unterschenkel
Motorstromsensoren	3 pro Gelenkmotor
Kopf	
Aktive Freiheitsgrade	2
Sensoren	
Kameras	2 am Kopf angebrachte IEEE 1394 Stereokameras

Tabelle 1: Technische Daten des LAURON Roboters

2.2 Stereobildverarbeitung

Um Informationen über das Gelände zu gewinnen in dem sich der LAURON Roboter befindet, stehen ihm zwei digitale Videokameras zur Verfügung. Bei Optischen Systemen unterscheidet man zwischen zwei Arten: aktive und passive Systeme. Aktive Systeme senden ein strukturiertes Licht aus, das als Messsignal ausgewertet wird (z.B Laserentfernungsmessung nach dem Triangulationsprinzip, siehe [24]). Passive Systeme hingegen, begnügen sich mit dem bereits vorhandenen Licht, und stellen die Lichtintensitäts- oder Farbinformationen der Umgebung in einem zweidimensionalen Bild fest. In dieser Arbeit wird sich vornehmlich mit passiven Systemen beschäftigt.

Um mit den von den Kameras aufgenommenen Informationen das Räumliche im Bild wahrnehmen zu können, wird das Verfahren der Stereobildverarbeitung darauf angewendet. Der Begriff Stereobildverarbeitung definiert sich aus den Wörtern Stereo und Bildverarbeitung. Die Bildverarbeitung ist die Auswertung und Aufbereitung von Bildsignalen und Bildsignalfolgen. Bildsignale stellen hier zweidimensionale Funktionen des Ortes und der Zeit dar, denen eine spezifischen physikalischen Größe zugeordnet ist. Das zweite Wort Stereo – Zwei – beschreibt in Anlehnung an das menschliche Sehen die Grundidee der Stereobildverarbeitung. Sie lautet wie folgt:

> *Wird eine Szene im dreidimensionalen Raum aus zwei unterschiedlichen Blickrichtungen betrachtet, ist die Bestimmung von Tiefeninformationen der betrachteten Objekte auf Grund geometrischer Beziehungen möglich.*[16]

Werden beide Begriffe verknüpft, wird ein Prozess beschrieben, bei dem ausgehend von zwei Bildern, welche die gleiche Szene von verschiedenen Standorten zeigen, Tiefeninformationen der sichtbaren Objektoberflächen bestimmt werden können.

Die Berechnung von Tiefeninformationen ist aber nur der letzte Schritt eines Stereobildverarbeitungssystems. Es besteht im allgemeinen aus vier Arbeitsschritten. Die Arbeitsschritte im einzelnen sind:

(1) Bildaufnahme
(2) Vorverarbeitung des Bildpaares
(3) Korrespondenzanalyse
(4) Tiefenwertbestimmung

2.2.1 Bildaufnahme

In den meisten Fällen findet die Bildaufnahme unter Zuhilfenahme von zwei Kameras statt, damit eine Szene zur gleichen Zeit aufgenommen werden kann. Das ist von entscheidender Bedeutung, denn dynamische Objekte in der Szene müssen sich bei beiden Bildern des Stereobildpaares an der gleichen Weltposition befinden, um eine korrekte Tiefenberechnung zu gewährleisten.

Eine bewährte Kameraanordnung für die Stereobildverarbeitung ist die kanonische Konfiguration oder in anderen Worten die achsenparallele Stereogeometrie. Bei diesem Aufbau sind beide Kameras C_l und C_r so in einer Ebene angeordnet, dass die Grundlinien ihrer Bilder horizontal und in derselben Höhe liegen. Des Weiteren gilt für die optischen Achsen der Kameras $\vec{c}_l$ und $\vec{c}_r$, dass sie senkrecht zu dieser Ebene stehen. Daraus ergibt sich außerdem:

$$\vec{c}_l \parallel \vec{c}_r \qquad (1)$$

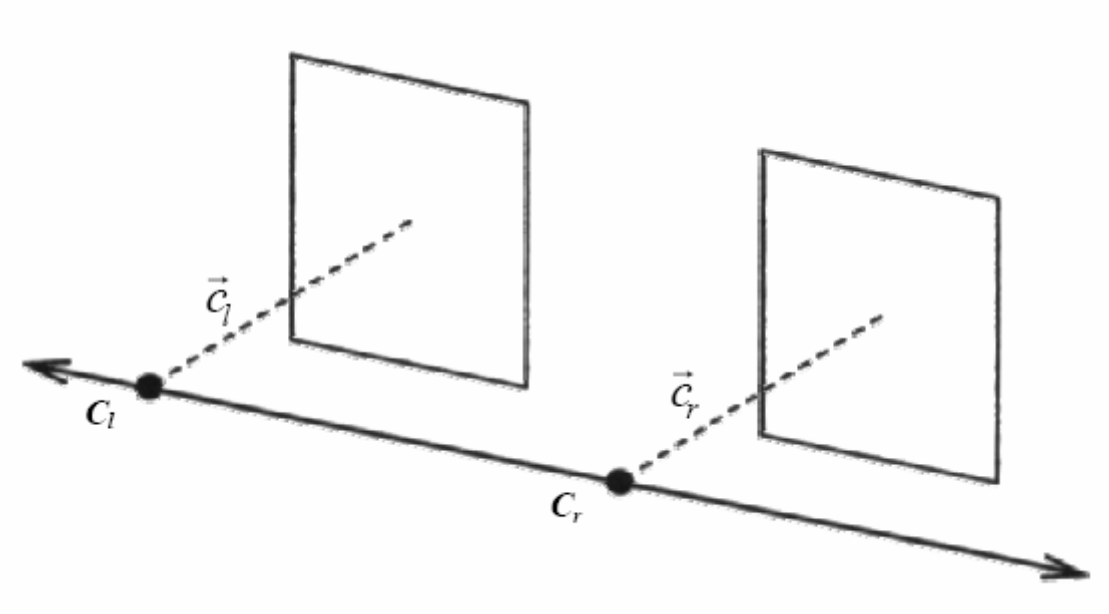

Abbildung 5: Kanonische Kamerakonfiguration für Stereovision

Die kanonische Kamerakonfiguration hat einige positive Effekte, die sie für den Einsatz mit mobilen Robotern prädestinieren. Das sind, bedingt durch den einfachen Aufbau, die Suche nach Gemeinsamkeiten in den Stereobildpaaren und die anschließende Tiefenrekonstruktion.

2.2.2 Vorverarbeitung des Bildpaares

Nach der Bildaufnahme folgt im Allgemeinen eine Vorverarbeitung der Bilder. Sie ist bei der Stereobildverarbeitung von besonderer Wichtigkeit. Zum einen müssen die durch die Bildaufnahme durch fehlerhafte Sensorelemente entstandenen Fehler korrigiert werden, zum anderen das in dem Bildpaar vorhandene hochfrequente Rauschen herausgefiltert werden. Das kann zum Beispiel durch elektrische Störfelder im Rechner bei der Bilddigitalisierung auftreten.

Durch die Anwendung von linearen verschiebungsinvarianten Filtern werden diese Probleme aber weitestgehend behoben, sodass eine Korrespondenzanalyse möglich ist. Durch diese Filter werden die Bilder unschärfer und damit insgesamt störungsfreier und rauschärmer.

Optional muss der Einsatz von Methoden zur Kalibrierung vorgesehen werden, die sozusagen Linsenverzerrungen und daraus entstehende Abbildungsfehler korrigieren. In dieser Arbeit fällt dieses Problem jedoch nicht an, da die Kameraverzerrungen so minimal sind, dass sie nicht ins Gewicht fallen.

2.2.3 Korrespondenzanalyse

Auf die Vorverarbeitung folgt die Korrespondenzanalyse. Sie stellt die Suche nach Gemeinsamkeiten zwischen den Bildern dar. Das ist der wichtigste Operationsschritt in der Stereobildverarbeitung, denn die korrespondierenden Pixel in den Bildern beschreiben dieselben Punkte im Raum. Probleme die bei der Korrespondenzanalyse auftreten können zweierlei Natur sein.

Zum einen gibt es Bereiche, in welchen es grundsätzlich keine Lösung zur Korrespondenzanalyse gibt. Aufgrund der unterschiedlichen Perspektive haben beide Kameras verschiedene Sichtbereiche, in welchem im jeweils anderen Bild kein entsprechendes Bildmerkmal gefunden werden kann. Außerdem existieren aufgrund der Tiefenstruktur einer Szene verdeckte Bereiche, die nur von einer Kamera gesehen werden.

Zum anderen können bestimmte Bildmuster zu Schwierigkeiten bei der Korrespondenzanalyse führen. Beispiele hierfür sind schwach texturierte Bildregionen oder periodische Strukturen (siehe Abbildung 6).

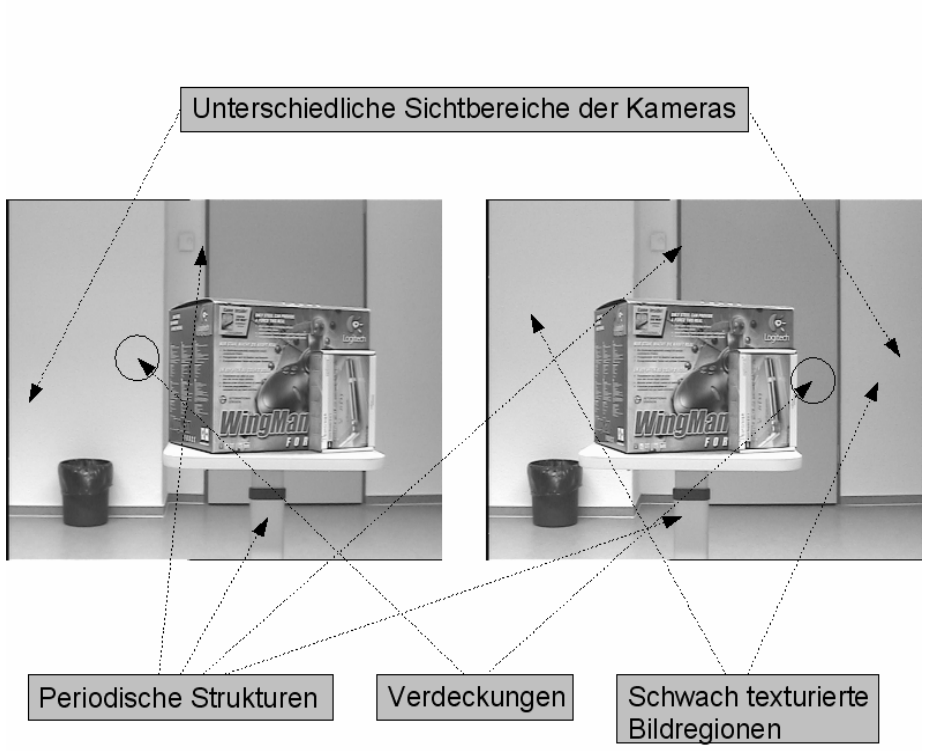

Abbildung 6: Problembereiche bei der Korrespondenzanalyse

Werden korrespondierende Pixel gefunden, kann mittels Anwendung des Strahlensatzes unter Benutzung von Kameraparametern (Brennweite, Größe des Aufnahmechips) und der Kamerakonfiguration (Abstand zwischen beiden Kameras, gegenseitige Orientierung) die Entfernung des Punktes, der in beiden Bildern als gleich markiert wurde, bestimmt werden.

Bevor Gemeinsamkeiten aber gesucht werden können, muss definiert werden, wann zwei Punkte des Bildpaares gleich aussehen, also gleich sind. In der Stereobildverarbeitung wird hier für die Korrespondenzanalyse die Annahme getroffen, dass korrespondierende Pixel einen ähnlichen Intensitätswert besitzen.

Ansätze zur Korrespondenzanalyse lassen sich prinzipiell in zwei unterschiedliche Klassen einteilen: lokale und globale Verfahren [3].

Lokale Verfahren

Lokale Verfahren betrachten lediglich die nähere Umgebung von in Frage kommenden Pixel im Suchbild, um das darunter am Besten zum aktuellen Pixel im Referenzbild korrespondierende auszuwählen. Beispiele für lokale Verfahren sind *merkmalbasierte Verfahren*, *gradientenbasierte Verfahren* oder das *Blockmatching Verfahren*.

Globale Verfahren

Globale Verfahren versuchen eine optimale Zuordnung der Pixel im Referenz- und Suchbild durch Minimierung einer globalen Kostenfunktion zu erreichen. Es wird im wesentlichen hierbei versucht eine globale Energie der Form

$$E(\mathrm{d}) = E_{data}(\mathrm{d}) + E_{constraints}(\mathrm{d}) \qquad (2)$$

zu minimieren. Wobei E_{data} wie im lokalen Fall die Güte des pixelweisen Matchings bezeichnet und über $E_{constraints}$ zusätzliche nicht lokale Eigenschaften, wie zum Beispiel Glattheit oder Eindeutigkeit, mit eingebunden werden können. Die Lösung einer solchen Minimierung über den gesamten Bildbereich ist aber mit erheblichem Rechenaufwand verbunden, weshalb der Suchraum häufig auf Scanlinien (Epipolarlinien) beschränkt wird. Beispiele für solche Verfahren sind *Dynamisches Programmieren* und *Graph Cut Verfahren*.

2.2.4 Tiefenberechnung

Der letzte Schritt des Stereobildverarbeitungsverfahrens ist die Bestimmung der Entfernung jedes einzelnen Punktes zu der Kamera, für den zwei Pixel in einem Bildpaar bei der Korrespondenzanalyse als gleich identifiziert werden konnten. Dieses Problem ist unter Annahme einer kanonischen Kamerakonfiguration, wie sie hier gemacht wurde, ohne Probleme durchführbar. Schematisch ist die Vorgehensweise in Abbildung 7 dargestellt.

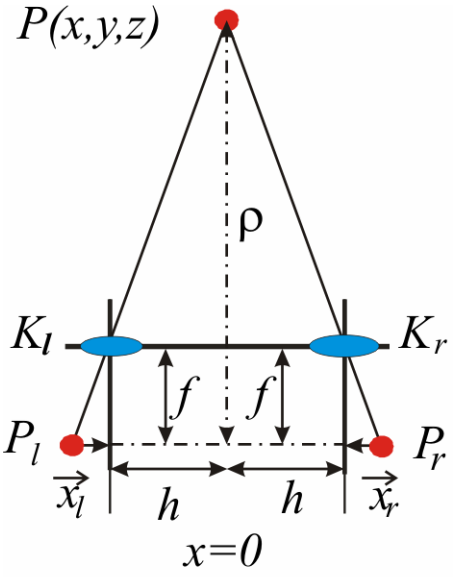

Abbildung 7: Geometrie der Stereovision mit kanonischer Kamerakonfiguration

Dabei stehen K_l und K_r für die beiden Kameras, die in einer Distanz von 2h voneinander in kanonischer Konfiguration aufgebaut sind. In beiden entsprechenden Bildern wurden die Punkte P_l und P_r als gleich identifiziert. Sie sind die Projektionen des Weltpunktes P mit den Koordinaten (x ,y ,z) auf die Bildebene. Das Koordinatensystem ist folgendermaßen definiert: die z-Achse repräsentiert die Entfernung von den Kameras. An den Kameras gilt somit z = 0.

Die horizontale Entfernung wird durch die x-Achse definiert, x = 0 ist die Position in der Mitte zwischen den beiden Kameras. Die Brennweite f legt den Abstand der beiden Brennpunkte zu ihren Hauptebenen fest und wird für beide Kameras als identisch vorausgesetzt (siehe Abbildung 8) [16].

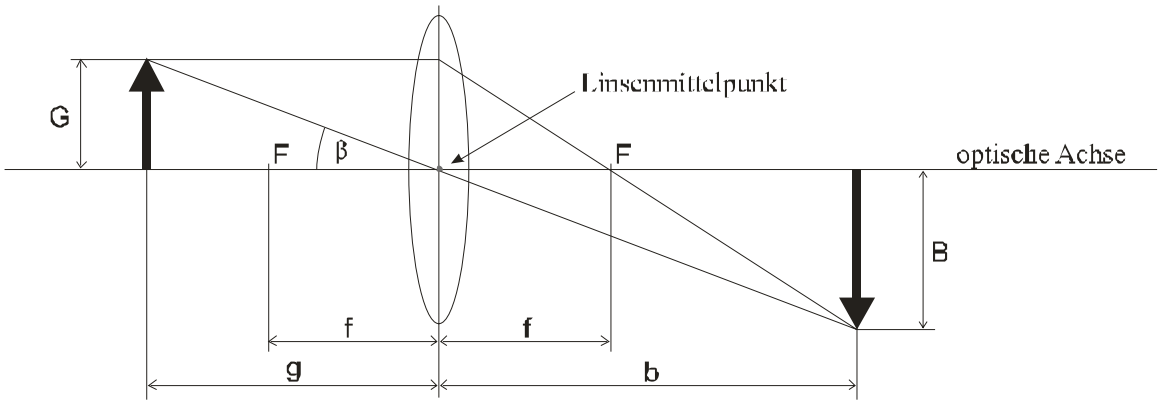

Abbildung 8: Optische Abbildung

Nun lassen sich folgende geometrische Regeln aufstellen:

$$\delta = | x_l - x_r | > 0 , \qquad (3)$$

da zwischen x_l und x_r eine Disparität besteht. Dabei handelt es sich bei δ um die Einheit Pixel.

Für die linke Seite von Abbildung 7 hat man laut Strahlensatz, wobei *x* für die Verschiebung des Punktes P nach links oder rechts steht (in Abbildung 7 ist $x = 0$):

$$\frac{x_l}{f} = - \frac{h + x}{\rho} \tag{4}$$

Folglich hat man für die rechte Seite laut Strahlensatz:

$$\frac{x_r}{f} = \frac{h - x}{\rho} \tag{5}$$

Löst man die zwei vorhergehenden Formeln nach $-x$ auf, so erhält man:

$$\frac{x_l}{f} \cdot \rho = \frac{x_r}{f} \cdot \rho - 2h \tag{6}$$

Nach ρ umgestellt und mit $2h = L$ ergibt sich:

$$\rho = \frac{Lf}{\delta \cdot d_u} \tag{7}$$

Letztendlich sind die Koordinaten des beobachteten Punktes P also nur von der Disparität $x_l - x_r$ bzw δ abhängig. d_u bezeichnet hierbei die Breite eines CCD-Elementes und stellt den Skalierungsfaktor dar, der die Disparität in Pixel in die Einheit *mm* umrechnet. Die Tiefeninformation ρ steht im umgekehrt proportionalen Zusammenhang mit der Disparität.

2.3 Laufplanung

Um die Fortbewegung eines Roboters zu implementieren, hat man sich an der Natur angelehnt und die Gangart verschiedener Tiere untersucht und studiert. Sehr gut untersucht wurde dabei die Gangart der indischen Stabheuschrecke. Wie bei allen Insekten ist auch ihr Körper dreigeteilt: nämlich in Kopf (Caput), Brust (Thorax) und Hinterleib (Abdomen). Der Thorax ist dabei auch wiederum dreigeteilt und an jedem Teil befindet sich jeweils ein Beinpaar. Jedes Bein hat drei Hauptsegmente (Coxa [Hüfte], Femur [Oberschenkel], Tibia [Unterschenkel]) und drei Hauptgelenke (Subcoxal [α], Coxa-Trochanter [β], Femur-Tibia [γ]) die ungefähr in einer Ebene liegen, wie in der folgenden Abbildung dargestellt.

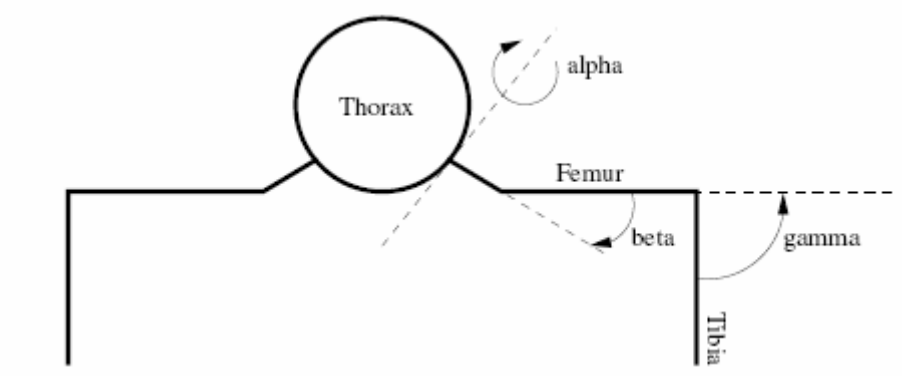

Abbildung 9: Beinsegmente und Hauptbewegungsachsen im vertikalen Schnitt

Die Gangart die sich als am stabilsten herausgestellt hat, ist der Dreibeingang. Um diese zu reproduzieren werden statische Verfahren eingesetzt. Beim LAURON Roboter wurde hierzu ein Verfahren gewählt in dem die Punkte für die Übergänge von der Schwingphase in die Stemmphase und von der Stemmphase in die Schwingphase in Abhängigkeit der benachbarten Beine definiert werden. Dazu wurden sechs einfache Regeln definiert, die in Form eines zellulären Automaten implementiert wurden. [5]

Die statischen Verfahren wurden für ein ganz bestimmtes Gelände entwickelt und können auch nur in diesem verwendet werden. So kann zum Beispiel ein Roboter der für ebenes Terrain programmiert wurde keine Treppen steigen und umgekehrt. Dies hat zur Folge, dass ein tiefes Schlagloch im Boden, auf sonst flachem Terrain unter Umständen zu einem unüberwindbaren Hindernis werden kann.

Abhilfe schaffen hier die reaktiven Verfahren. Diese versuchen mittels Kraftsensoren festzustellen wann (zu früh oder zu spät) ein Bein in die Stemmphase eintritt, oder ein Bein in seiner Bewegung blockiert wird, oder mittels Tastsensoren festzustellen ob ein Kontakt mit den Untergrund bzw dem Hindernis vorliegt. Tritt ein solcher Fall ein, wird durch „Tasten“ eine geeignete Auftrittsstelle in der direkten Umgebung gesucht, um danach wieder in die statische Bewegung überzugehen.

In stark zerklüftetem Gelände hat dieses Verfahren aber seine Grenzen. Hat der Boden zum Beispiel zu viele Unebenheiten, überlagert das reaktive Verhalten die eigentliche Fortbewegung und es kann dazu kommen dass der Roboter sozusagen fest hängt. Es hat sich gezeigt, dass mit der Kenntnis des Geländes, eine Laufplanung das Schreiten verbessern kann.

Aufgabe der Laufplanung

Die Laufplanung soll im Vorfeld, also vor dem Eintreten der Störung, geeignete Fußauftrittspunkte finden. Anders als bei reaktiven Verfahren muss auf diese Weise die Fortbewegung nicht unterbrochen werden.

Ein „Nicht-Unterbrechen“ bedeutet aber auch, dass die Laufplanung während des Schreitens berechnet werden muss. Dies stellt eine Echtzeitanforderung dar. Im Gegensatz Pfadplanung ist die Laufplanung nur für die nähere Umgebung, das heißt für die nächsten paar Schritte verantwortlich.

Aufgabe der Routen oder Pfadplanung

Das Suchen nach einem möglichen Weg oder Wegen um ein Hindernis herum fällt nicht in den Bereich der Laufplanungsalgorithmen, sondern in den der Routen- oder auch Pfadplanung. Es ist wichtig dies zu unterscheiden, da bei der Routenplanung von den Sensordaten eine deutlich weitere „Sichtweite“ gefordert wird als bei der Laufplanung. [24]

2.4 Simulationsumgebung zur Laufplanung

In diesem Kapitel wird die Simulationsumgebung vorgestellt in der man bereits eine LAURON Simulation erstellt hat, um den Laufplaner zu testen und Simulationsergebnisse und Sensorinformationen zu visualisieren. Sie bietet die Möglichkeit, verschiedene Szenarien unter reproduzierbaren Bedingungen zu testen [10]. Die simulierten Kameras erlauben es, Stereobildpaare von der Umgebung und Vergleichsinformationen zur Bewertung der Resultate der Stereobildverarbeitung zu gewinnen. Sie wird im Folgenden kurz vorgestellt.

2.4.1 Simulationsumgebung

Die Simulationsumgebung ist in der objektorientierten Programmiersprache C++ geschrieben. Zur Darstellung wird das 3D-Grafik Entwicklungs-Toolkit von SGI Open Inventor benutzt [13]. Es ist auf OpenGL aufgesetzt und ebenfalls in C++ implementiert. Openinventor bietet eine komplette graphische Benutzeroberfläche, die grundlegende Manipulationen am Darstellungsobjekt zulässt, wie zum Beispiel das Vergrößern oder Verkleinern von Objekten, Kameraflug, Verschieben von Objekten usw.

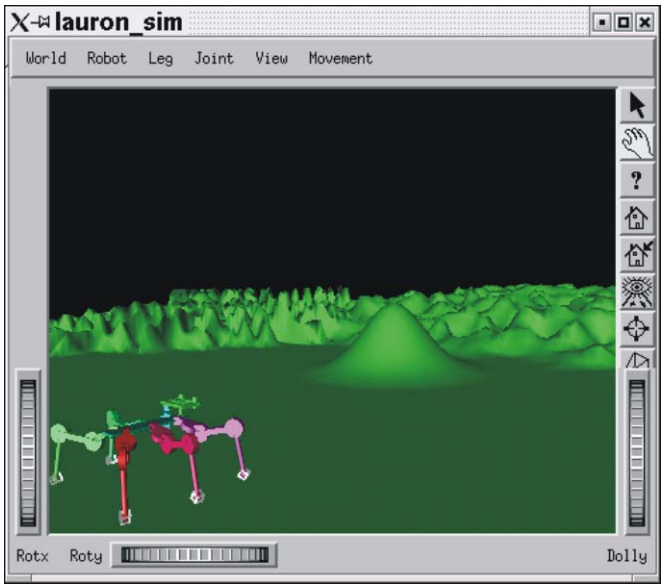

Abbildung 10: Simulationsumgebung

Der Aufbau der LAURON Simulation wird im Folgenden beschrieben. In der *main* – Funktion werden die Objekte *World*, *Movement* und *Terrain* angelegt. Die Klassen *Eye* und

Robot hängen von der Klasse *World* ab, während *Head* und *Leg* wiederum von der Kompositionsklasse *Robot* abhängen (siehe Abbildung 11).

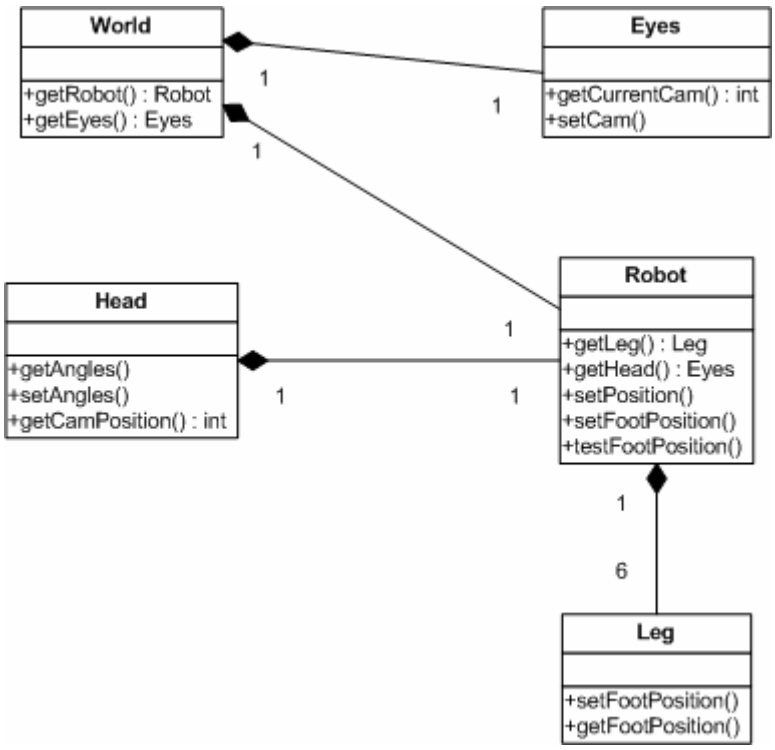

Abbildung 11: Auszug aus dem UML Klassendiagramm der Simulationsumgebung

Die Klasse *Eye* stellt dabei die in Open Inventor definierten Kameras dar, die einen Blick auf die Szene ermöglichen. Durch Umschalten des Viewports kann man ein Stereokamerasystem simulieren und Schnappschüsse von der 3D Welt machen um Stereobildpaare zu erhalten. Dazu wird direkt mittels openGL auf das generierte Bild zugegriffen, und die so gewonnen Daten werden in eine Datei gespeichert.

Da die generierten Bilder anders als tatsächliche Aufnahmen keine Textur erkennen lassen, und ohne Texturen auf den Bildern die Stereobildverarbeitungsalgorithmen nix erkennen können, muss ein Texturmapping implementiert werden. Open Inventor stellt hierzu die Klasse SoTexture2 zur Verfügung, die eine Textur aus einem JPEG Bild laden kann. Um mit den Stereobildverarbeitungsalgorithmen möglichst gute Resultate zu erzielen, wählt man eine Textur mit möglichst hoher Auflösung. Eine hohe Auflösung hat zur Folge, dass sich die Textur nicht zu häufig wiederholt, somit sinkt die Wahrscheinlichkeit, dass der Roboter mehrmals die gleiche Oberfläche sieht, bzw von periodischen Strukturen.

2.4.2 Laufplanungsalgorithmus

André Herms entwickelte im Laufe seiner Arbeit [10] einen Laufplanungsalgorithmus in der eben vorgestellten Simulationsumgebung. Da kein exaktes Lösungsverfahren bekannt war suchte er eine geeignete Heuristik. Es wurden eine Auswahl von allgemeinen Heuristiken auf ihre Eignung untersucht. Dabei stellte sich heraus das die Random Sampling Methode am geeignetesten dafür ist, da der Algorithmus sehr einfach ist und auch geringe Rechenzeiten hat.

Die Grundidee von Random Sampling ist ziemlich simpel und auch leicht umzusetzen: Es werden große Mengen von zufälligen Lösungen generiert. Das entsprechend der Bewertungsfunktion beste Ergebnis wird übernommen.

Auf den Laufplaner des LAURON Roboters angewendet heißt das, dass man zufällig Fußauftrittspunkte und Körperpositionen generiert, wobei man bereits beim Erzeugen darauf achtet, dass nur gültige Konfigurationen erzeugt werden. Das heißt:

- es werden keine Füße angehoben die ein Kippen des Roboters zur Folge hätten
- es werden keine Punkte außerhalb der physikalischen Reichweite der Beine angesteuert
- es wird kein Untergrund mit zu starkem Gefälle betreten bzw der instabil ist
- es dürfen keine Füße bzw Beine zusammenstoßen
- es dürfen keine Geschwindigkeiten für die Fußbewegung angenommen werden, die die Beinregler nicht leisten können

Der Algorithmus legt zunächst einen Pfad zum Ziel fest. Er erzeugt dazu eine zufällige Anzahl zufälliger Körperpositionen, welche in eine Liste eingetragen werden. In Zukunft sind nur noch diese Körperpositionen zulässig. Jetzt wird für die aktuelle Fußstellung der Bereich ermittelt, in dem sich der Körpermittelpunkt befinden kann. Dann wird zufällig ein Fuß ausgewählt, der angehoben bzw abgesetzt wird. Für das Absetzen wird dabei zufällig eine neue gültige Fußposition bestimmt. Danach wird eine neue Körperposition innerhalb des zuvor bestimmten Bereiches festgelegt. Zu den einzelnen Positionen werden anschließend minimale zulässige Werte für Dauer und Startzeit berechnet. Weitere Details zu diesem Thema gibt es in den Arbeiten von André Herms [10] und Uli Ruffler [23].

Durch Implementierung eines Stereobildverarbeitungsalgorithmus und somit verbundener Erstellung eines Weltmodells, kann man den Laufplanungsalgorithmus in der Hinsicht verbessern, dass man die Fussauftrittspunkte im Vorfeld als nicht gültig markiert, welche auf Kollisionskurs mit einem unüberwindbaren Hindernis stehen.

3. Verwandte Arbeiten

Die Gewinnung von Geländeprofilen mittels Stereobildverarbeitung wurde in der Wissenschaft schon häufiger untersucht und es wurden schon vielerlei Algorithmen dafür entwickelt. Allerdings gibt es wenige Arbeiten in denen die verschiedenen Algorithmen klassifiziert wurden und verglichen wurden um ihre Performanz zu untersuchen. Hier setzt die Arbeiten von Scharstein und Szeliski 2001 [25] und von Brown [3] auf, die durch verschiedene Experimente versucht die Qualität der Ergebnisse und die Rechenzeit der einzelnen Verfahren zu bestimmen um dann eine Klassifizierung aufzustellen.

Es wurden auch bereits Methoden entwickelt um das Verfahren zu beschleunigen. Denn je genauer die Umgebung mit ihren Objekten bekannt ist, desto schneller wird die Bildverarbeitung bzw Stereobildverarbeitung weil man den Suchbereich zum Beispiel einschränken kann, und desto genauer werden die Ergebnisse. So kann man durch den Vergleich zwischen einem vom Kamerakopf aufgenommenen Objekt, mit bereits in einer Datenbank abgelegtem bekanntem Objekt, dieses Objekt erkennen. [2]

Bei mobilen autonomen Robotersystemen kommt dieser Ansatz jedoch nicht in Frage da keinerlei Informationen über die Umgebung vorliegen. Das heißt man muss mit sehr generischen Ansätzen arbeiten, was die Problematik komplizierter werden lässt. Ein Ansatz wäre deshalb, mehrere Verfahren zu kombinieren, je nachdem, welche Objekte im Bild sind und je nachdem wie die äußeren Lichtverhältnisse sind. So stellen Matthies, Kelly, Litwin und Tharp [20] in ihrer Arbeit ein eine Methode vor, mit der unbemannte autonome Fahrzeuge Hindernisse erkennen können, sei es bei Tag, Nacht oder anderweitigen schlechten Sichtverhältnissen. Zum Einsatz kommt dabei ein echtzeitfähiges Stereobildverarbeitungs-system, das sogenannte JPL (Jet Propulsion Laboratory) Stereo Vision System welches einen Algorithmus beschreibt der aus verschiedenen Filterungen, Disparitätserrechnung und Triangulationen besteht. Die Stereobilder werden bei Tag vom Kamerasystem in einer Auflösung von 256×45 aufgenommen und erreichen eine Framerate von 0,6 Bildern in der Sekunde. Bei Nacht kommen FLIR Kameras [13] zum Einsatz, das heißt spezielle Infrarot und Wärmebildkameras. Hier werden die Bilder in einer Auflösung von 128×120 aufgenommen. Mit Hilfe der Infrarot Kameras ist es sogar möglich Vegetationshindernisse, wie zum Beispiel einen Busch von nicht Vegetationshindernissen, wie zum Beispiel einer Schlucht zu unterscheiden.

Auch im Bereich der Umweltmodellierung wurde schon geforscht [7]. Lutz Frommberger stellt in seiner Arbeit einen Entwurf einer Umweltmodellierung vor, um die Navigation von Laufmaschinen, wie zum Beispiel dem Laufroboter LAURON, im Gelände zu ermöglichen. Es wurde dabei ein Verfahren entwickelt, mit dem es möglich ist, sensorisches Wissen über die Umgebung eines Roboters in geeigneter Form aufzunehmen und zu speichern, so dass eine Auswertung möglich ist, die das Laufverhalten der Maschine deutlich verbessert. Das vorgestellte Modell verwaltet Kartierungen der Umwelt in verschiedenen Abstraktionsstufen, indem es aus einer scrollenden 3D-Urkarte globale 2D-Karten und lokale Abbildungen der Fußumgebungen erzeugt und Detailinformationen vor allem für das unmittelbare Umfeld der Maschine speichert. Damit kann für jede aus dem Weltmodell erwachsende Aufgabe eine dafür geeignete Karte bereit gestellt werden.

4. Stereobildverfahren zur Umgebungsmodellierung

In diesem Kapitel werden erst einmal die einzelnen Module der Steuerungssoftware des LAURON Systems vorgestellt und eine Abstraktion auf das hier in der Arbeit zu behandelnde Thema, den Stereobildverarbeitungsalgorithmen gemacht. Es werden einige geeignete solcher Verfahren vorgestellt, und die zwei vielversprechendsten daraus extrahiert um sie genauer zu untersuchen.

4.1 Problemexposition

Um das langfristige Ziel, einen selbsterkundenden LAURON Roboter zu realisieren, hat man die Steuerungssoftware in verschiedene Module aufgeteilt, die jeweils eine Teilaufgabe übernehmen. Schematisch ist die Hierarchie der Steuerungssoftware in Abbildung 12 dargestellt.

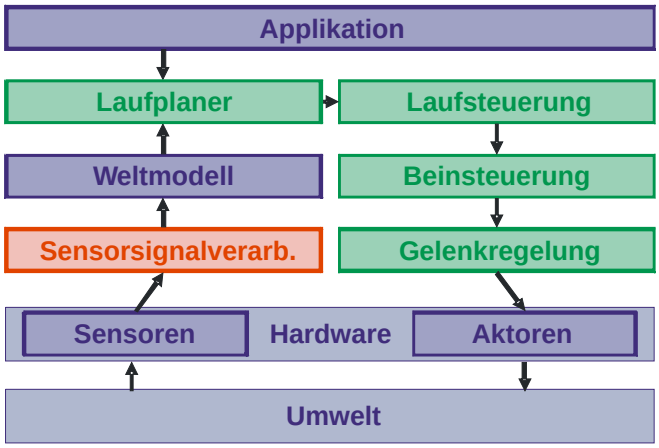

Abbildung 12: Systemübersicht

Gelenkreglung

Der Gelenkregler (engl: joint control) dient der Steuerung der Gelenke der Roboterbeine. Als Eingabe wird dabei der gewünschte Winkel vorgegeben.

Beinsteuerung

Der Beinregler (engl: leg control) dient der Steuerung eines kompletten Beines. Dabei unterscheidet man zwischen dem Stützmodus, bei dem das Bein sich auf dem Boden befindet und den Roboter stützt, und dem Hebemodus bei dem das Bein vorwärts gesetzt wird indem es sich vom Untergrund abhebt um sich zu den vorgegebenen Koordinaten hinzubewegen.

Laufsteuerung

Die Laufsteuerung (engl: gait control) hat die Aufgabe alle Beine zu koordinieren. Durch eine geeignete Abfolge von Beinbewegungen wird das Laufen des Roboters ermöglicht.

Weltmodell

Das Weltmodell (engl: world model) stellt die Umgebung des Roboters in Form eines Modells dar. Es enthält Hindernisse und Höheninformationen des Untergrundes. Dieses Weltmodell wird auf Grundlage der Sensorinformationen durch das System erstellt und ermöglicht eine Planung von Bewegungen.

Applikation

Die Applikation (engl: application) legt fest, wohin sich der Roboter bewegen soll. Dabei ist es unerheblich ob ein autonomes System aufgesetzt wird, das aktiv die Umgebung erkundigt oder der Mensch selbst die Steuerung übernimmt. Es wird sozusagen nur eine Menge von Zielpunkten vorgegeben, für deren Erreichen die unteren Schichten verantwortlich sind.

Laufplaner

Der Laufplaner (engl: gait planner) bezieht Informationen aus dem Weltmodell, und berechnet damit im Vorfeld geeignete Fußauftrittspunkte. Das heißt Fußauftrittspunkte, die gewisse Kriterien erfüllen wie die Stabilität des Roboters oder ein möglichst frühes Erreichen des Zielpunktes. Außerdem soll die Fortbewegung nicht unterbrochen werden, das heißt neue Fußauftrittspunkte müssen während der Laufzeit berechnet werden, was eine Echtzeitanforderung darstellt. Die Laufplanung (siehe Kapitel 2.4.2) wurde in den Arbeiten von André Herms [10] und Uli Ruffler [24] realisiert und verbessert.

Sensorsignalverarbeitung

Die Sensorverarbeitung (engl: sensor processing) ist schließlich das Thema dieser Arbeit. Sie dient der Aufbereitung der Sensordaten. In diesem Fall handelt es sich um die Daten eines Stereokamerakopfes in Form von Bildern die von Interesse sind. Die beiden Kameras liefern zeitgleich zwei Bilder, die in der Horizontale leicht versetzt sind, und mit deren Hilfe man durch Stereobildverarbeitungsalgorithmen Tiefeninformationen für jeden einzelnen Pixel extrahieren kann.

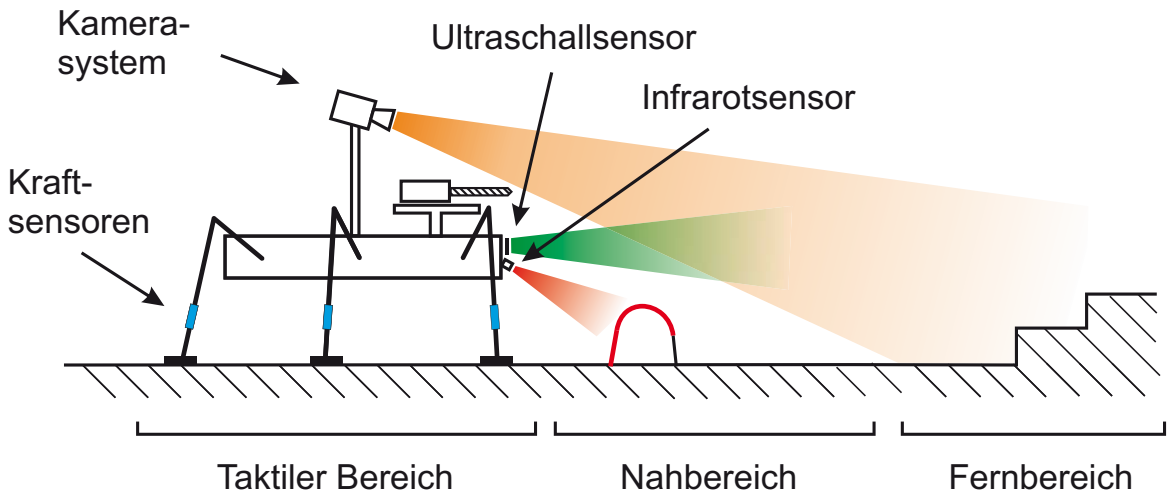

Abbildung 13: Sensoren am Roboter

Es werden in dieser Arbeit dabei verschiedene Verfahren der Stereobildverarbeitung theoretisch sowie auch praktisch untersucht, durch Anwendung der Verfahren auf verschiede Szenarien, unter anderem aus der realen Welt. Wir benutzen dabei nur passive Systeme, das heißt dass man sich mit dem bereits vorhandenen Licht begnügt und nicht extra ein Licht aussendet, um die Lichtintensitäts- oder Farbinformationen der Umgebung in einem zweidimensionalen Bild festzustellen. Dabei werden Szenarien aus dem Nahbereich wie auch aus dem Fernbereich untersucht (siehe Abbildung 13).

4.2 Untersuchung geeigneter Stereobildverarbeitungsalgorithmen

In diesem Kapitel geht es darum verschiedene Stereobildverarbeitungsverfahren zur Erstellung von Korrespondenzen vorzustellen, etwaige Vor- und Nachteile zu diskutieren um anschließend diejenigen zu extrahieren, welche für unser System am geeignetesten erscheinen. Man unterscheidet dabei laut Brown [3] zwischen lokalen und globalen Verfahren.

4.2.1 Merkmalbasierte Verfahren

Bei merkmalbasierten Verfahren, einem lokalen Verfahren, werden nur ausgewählte Bildmerkmale hinsichtlich ihrer Korrespondenz untersucht. Deshalb ist zuerst eine Vorverarbeitungsstufe notwendig, um aus den beiden Stereobildern die gewünschten Bildmerkmale zu extrahieren. Der zusätzliche Rechenaufwand durch die Merkmalsanalyse kann in der nachfolgenden Korrespondenzanalyse unter Umständen wieder aufgehoben werden. Diese Merkmale können verschiedener Natur sein wie z.B Kantenpunkte oder andere Punktmerkmale, Linien, Konturen oder sogar abstrakte Objekte.

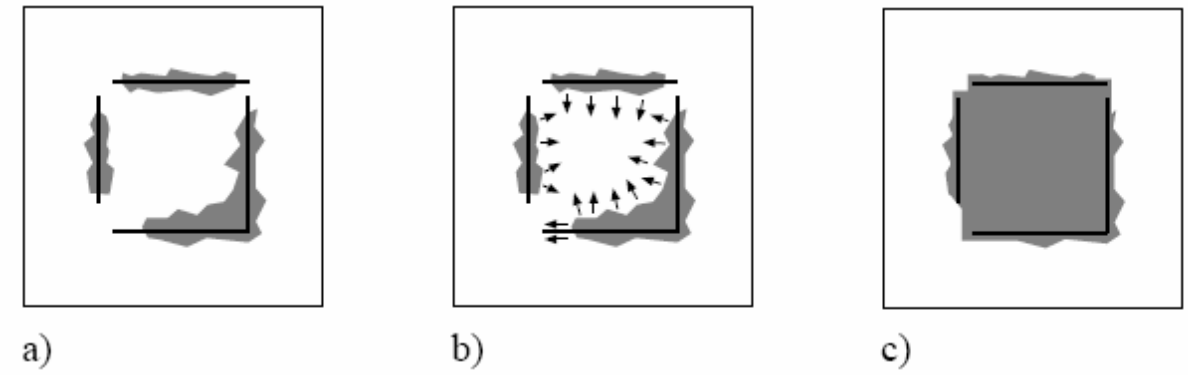

Abbildung 14: Merkmalbasiertes Verfahren

Abbildung 14 zeigt eine Anwendung für dieses Verfahren. In a) werden mit Hilfe des Canny Kantendetektors die Kanten bestimmt und die initialen Disparitätswerte. Zusätzlich kann man dann mit relativ hohem Aufwand die Disparitätswerte welche mittels eines Flutalgorithmus expandieren (siehe b) um schlussendlich das Ergebnis c) zu erhalten.

Allerdings ist die Dichte des Tiefenbildes stark begrenzt. Da aber viele aktuelle Applikationen sehr dichte Tiefenkarten brauchen und andere Verfahren wie zum Beispiel das Blockmatching robuster und effizienter geworden sind hat man im letzten Jahrzehnt das Interesse an ihnen verloren. [3] [27]

4.2.2 Gradientenbasierte Verfahren

Gradientenbasierte Verfahren gehören zur Klasse der lokalen Verfahren. Sie versuchen lokale Disparitäten zwischen Bildpaaren mittels Differenzialgleichungen über Bewegung und Bildhelligkeit zu beschreiben. Dabei wird die Annahme gemacht dass die Bildhelligkeit eines Pixels zwischen den beiden Sichten gleich ist. Im Anschluss wird die horizontale Translation vom Pixel des einen Bildes zu dem anderen mit Hilfe einer einfachen Differenzialgleichung berechnet:

$$(\nabla_x E)v + E_t = 0\,, \qquad (8)$$

wobei $\nabla_x E$ den Horizontalteil des Bildgradienten, E_t das Zeitdifferenzial und v die Translation zwischen den beiden Bildern bezeichnet.

Gradientenbasierte Verfahren liefern zwar hinreichend dichte Tiefeninformationen, haben aber in der Regel einen recht hohen Aufwand. [3] [26]

4.2.3 Blockmatching Verfahren

Das einfachste Bildmerkmal zur Korrespondenzanalyse ist ein Bildpunkt, der sich wie bereits gesehen im Allgemeinen durch seine Intensität und bei Farbbildern zusätzlich durch seine Farbwerte auszeichnet. Wesentlich aussagekräftiger ist jedoch die Bildstruktur in der Umgebung dieses Bildpunktes, deshalb wird nicht der einzelne Bildpunkt, sondern ein Fenster um diesen Bildpunkt herum, ein so genannter Block, bei der Korrespondenzanalyse betrachtet. Diese Vorgehensweise wird Blockmatching genannt. Es handelt sich dabei um ein Verfahren aus der Klasse der lokalen Verfahren.

Laut Brown ist das Blockmatching ein sehr robustes und effizientes Verfahren das sehr dichte Disparitätskarten liefert. [3]

4.2.4 Dynamisches Programmieren

Das Dynamische Programmieren gehört zu der Klasse der globalen Verfahren. Es bezeichnet nicht die besondere Art der Entwicklung eines Computerprogramms, sondern vielmehr eine mathematische Herangehensweise, mittels definierter Regeln ein komplexes Zuordnungsproblem zu lösen. Man versucht dabei ein komplexes Problem in weniger komplexe Teilprobleme zu zerlegen. Auf das Problem der Korrespondenzanalyse angewendet bedeutet dies Folgendes. Zuerst wird das Ähnlichkeitsmaß für alle paarweisen Korrespondenzen zwischen den Bildpunkten auf korrespondierenden Zeilen in den beiden Stereoansichten berechnet. Dabei wird von einem eingeschränkten Disparitätsbereich ausgegangen. Anschließend erfolgt die Abarbeitung der Ähnlichkeitsfunktion und die Auswahl der tatsächlichen Korrespondenzen, indem der Pfad innerhalb der zweidimensionalen Ähnlichkeitsfunktion mit der größten Ähnlichkeit gesucht wird.

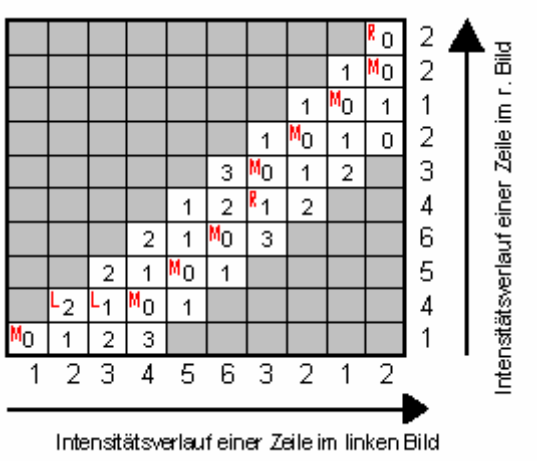

Abbildung 15: Ähnlichkeitsmatrix bei der Dynamischen Programmierung

In Abbildung 15 ist für die Intensitätsverläufe zweier korrespondierender Bildzeilen die Ähnlichkeitsmatrix angegeben, wobei als Ähnlichkeitskriterium die absolute Differenz gewählt wurde.

Nachdem für jeden Bildpunkt in der linken Zeile die Ähnlichkeit zu Bildpunkten in der rechten Zeile berechnet wurde, geht man bei der dynamischen Programmierung schrittweise durch diese Ähnlichkeitsmatrix und sucht den optimalen Pfad unter der Annahme, dass die Teillösungen für jeden Bildpunkt in der linken Zeile optimal sind. Somit können entsprechende Zuweisungen bestimmt werden, die mit **M** bezeichnet sind. Zuweisungen, die aufgrund von bestehenden Korrespondenzen in der anderen Zeile nicht mehr möglich sind, werden als Verdeckungen entweder von links (**L**) oder von rechts (**R**) markiert.

Die Korrespondenzanalyse mittels Dynamischer Programmierung liefert relativ robuste Ergebnisse, es ist jedoch aufgrund des hohen Rechenaufwandes durch die große Anzahl der Ähnlichkeitsberechnungen nur begrenzt und vor allem in Offline Anwendungen einsetzbar. Außerdem kann es bei der Herstellung der Konsistenz zwischen den einzelnen Zeilen oder Scanlinien zu Problemen kommen, so dass es im Ergebnis häufig zu streifenförmigen Fehlern kommt. [27]

4.2.5 Graph Cut Verfahren

Im Gegensatz zum Dynamischen Programmieren das die Korrespondenzen nur zeilenweise sucht, bietet das Graph Cut Verfahren die Möglichkeit auch die 3.te Dimension mit einzubeziehen. Es werden also Nachbarschaften sowohl entlang als auch quer zu den Bildzeilen mit einbezogen. Somit werden Streifenmuster vermieden.

Dieses Verfahren das zu der Klasse der globalen Verfahren gehört, ist nach Brown schneller als Dynamisches Programmieren und liefert außerdem bessere Ergebnisse.

4.3 Auswahl geeigneter Verfahren

Der LAURON Roboter bezieht bekanntlich geeignete Fußauftrittspunkte aus dem Weltmodell. Dieses wird während der Laufzeit mit Hilfe der aus den Stereobildverarbeitungsalgorithmen gewonnenen Tiefenkarten erstellt. Damit dieses reibungslos funktioniert müssen die Stereobildverfahren bestimmte Kriterien erfüllen:

- Dichte Tiefenkarten
- Echtzeitfähigkeit

Damit ein Weltmodell ohne „Lücken“ erstellt werden kann müssen die von den Stereobildverarbeitungsalgorithmen gelieferten Tiefenkarten möglichst dicht sein, das heißt es sollten keine größeren Gebiete in der Karte sein, wo keine Tiefeninformationen bzw falsche Informationen gefunden wurden. Es wäre zum Beispiel schlecht wenn Objekte oder Teile davon einfach „übersehen“ werden. Merkmalbasierte Verfahren erfüllen dieses Kriterium nicht und scheiden somit aus.

Damit die Berechnung des Weltmodells und somit geeigneter Fußauftrittspunkte während der Laufzeit geschehen kann, darf die Rechenzeit der Stereobildverarbeitungsalgorithmus nicht zu hoch sein. Wir legen sie auf maximal 5 Sekunden fest, da sich der LAURON Roboter sehr langsam bewegt. Gradientenbasierte Verfahren, sowie Dynamisches Programmieren erfüllen dieses Kriterium nicht da sie sehr hohe Rechenzeiten haben.

Übrig bleiben also das Blockmatching Verfahren und das Graph Cut Verfahren welche als geeignete Verfahren in Frage kommen könnten.

5. Blockmatching Verfahren

In diesem Kapitel wird das Blockmatching Verfahren vorgestellt und auf Tauglichkeit für unser LAURON-Roboter System untersucht. Dabei wird zuerst die Funktionsweise des allgemeinen Blockmatchings an einem Stereobildpaar erklärt, bevor man zeigt wie man die erhaltenen Ergebnisse verbessern kann. Anschließend werden Methoden zur Beschleunigung des Verfahrens vorgestellt.

5.1 Allgemeines Blockmatching

Als sehr robustes und sehr effizientes lokales Verfahren hat sich das Blockmatching herausgestellt [3]. Dieser Algorithmus hat seinen Namen auf Grund seiner Lösungsstrategie bekommen, welche die Definition von gleichen Pixel zwischen den Bildern auf die Gleichheit von Blöcken ausdehnt. Das Verfahren ist also nicht nur pixelbasiert, sondern regionalbasiert, da die Annahme, dass ein Pixel auf Basis seines Intensitätswertes eindeutig zugeordnet werden kann, als nicht ausreichend bewertet wird. Die Ursache ist, dass normalerweise eine große Anzahl identischer Intensitätswerte in einem Fenster, auch Block genannt, zusammengefasst werden, um eine höhere Eindeutigkeit zu erhalten.

Ein zentrales Problem bei der Korrespondenzanalyse mittels Blockmatching ist die geeignete Wahl der Fenstergröße. Um eine zuverlässige Bestimmung der Disparität zu erreichen, ist eine große Intensitätsvariation und damit ein genügend großes Fenster notwendig. Auf der anderen Seite kann dieses große Fenster Regionen erfassen, in welchen die Tiefe der Szenenpunkte variiert. Dies kann wegen perspektivischer Verzerrungen zu Fehlern in der Korrespondenzanalyse führen. Wird das Fenster zu klein gewählt, so kann zwar der Einfluss durch perspektivische Verzerrungen vermindert werden, allerdings reduziert sich damit auch die Intensitätsvariation und es verschlechtert sich die Disparitätsbestimmung. Durch Experimente wurde ermittelt dass eine Blockgröße um 8×8 Pixel die besten Resultate liefert.

Die Zuordnung zwischen den Pixeln erfolgt anschließend anhand eines Ähnlichkeitsmaßes zwischen den Intensitätswerten der Fenster. Man unterscheidet dabei zwischen parametrischen Ähnlichkeitsmassen, zu denen der Mittlere absolute Fehler, die normierte Kreuzkorrelation und auch der später verwendete mittlere quadratische Fehler MSE gehören und nicht-parametrischen Ähnlichkeitsmassen. Letztere vergleichen nicht die Intensitäten selbst zwischen zwei Blöcken, sondern die relative Anordnung der Bildpunktintensitäten zueinander. Dies bedeutet dass vor einem Vergleich zuerst eine nicht-parametrische Transformation durchgeführt wird. Zu diesen Transformationen zum Beispiel die Rank-Transformation und die Census-Transformation. Genaueres hierzu kann in [24] nachgelesen werden.

Das Blockmatching Verfahren arbeitet im Detail nach folgendem Muster. Als erster Verarbeitungsschritt wird ein Bild des Bildpaares (zb das linke) in eine konstante Anzahl von gleich großen Blöcken aufgeteilt. Die Suche nach einem korrespondierenden Block im rechten Bild wird nur für die festgelegten Blöcke des linken Bildes durchgeführt. Gesucht werden muss im rechten Bild, dank der kanonischen Kamerakonfiguration, nur in horizontaler Richtung, das heißt nur in einer Zeile (siehe Abbildung 16). Als Maß für die Ähnlichkeit zweier Blöcke wird hier der mittlere quadratische Fehler MSE (mean square error) zwischen den Intensitätswerten der Pixel innerhalb der entsprechenden Blöcke verwendet.

Die Intensitätsfunktion des linken bzw. rechten Bildes werden mit E_l bzw. E_r bezeichnet. Das Ähnlichkeitsmaß ist für einen Offset Δ, der die Differenz $(x_l - x_r)$ zwischen den Spaltenpositionen im rechten und im linken Bild angibt, um eine Blockgröße von n×m Pixeln durch:

$$\text{MSE}(x, y, \Delta) = \frac{1}{n \cdot m} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} | E_l(x + i, y + j) - E_r(x + i + \Delta, y + j) |^2 \qquad (9)$$

definiert.

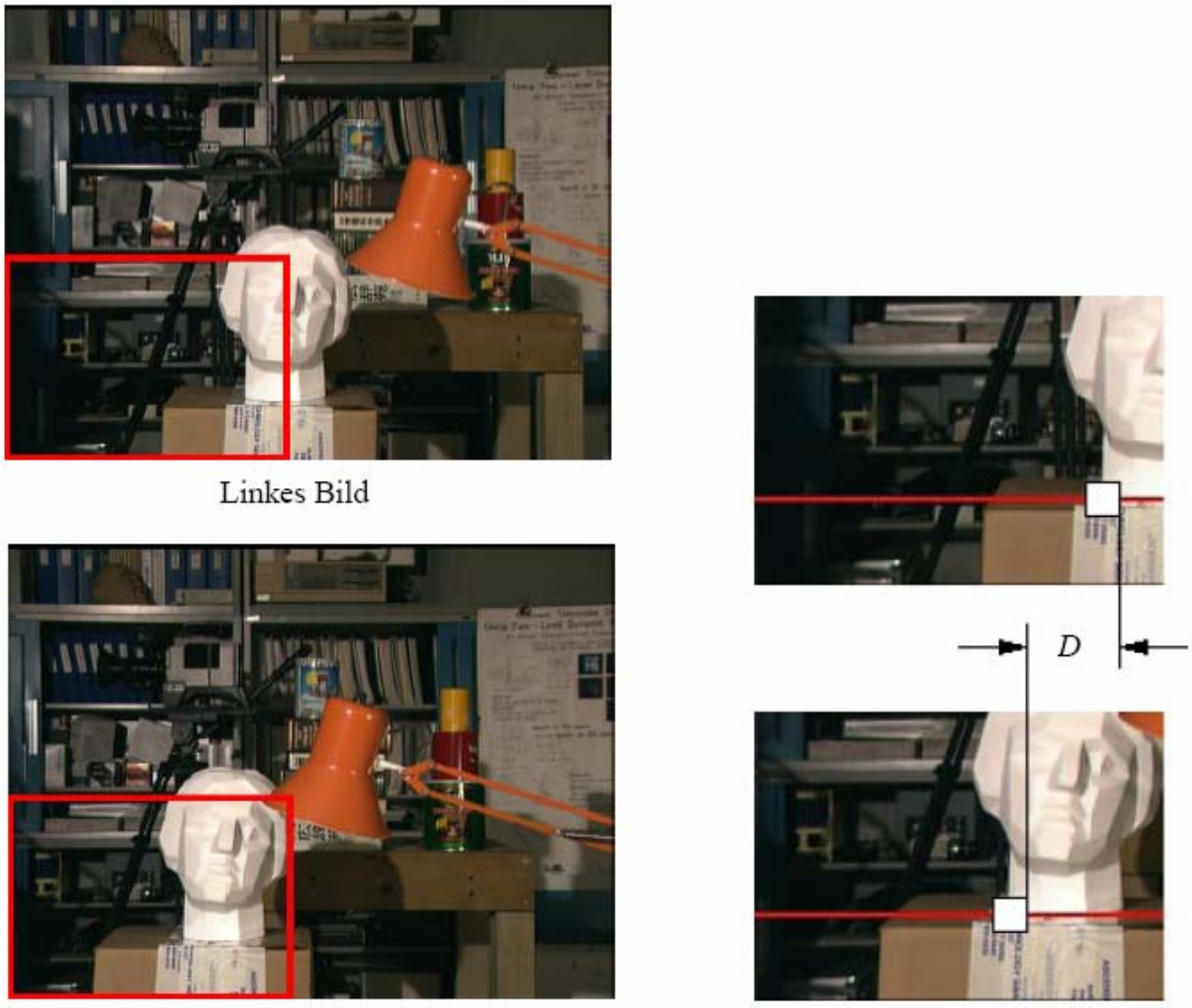

Abbildung 16: Blockmatching Verfahren

(x,y) bezeichnet hier jeweils die linke obere Ecke eines Blockes im linken Bild. Die Disparität D zwischen den Blöcken ist definiert durch den Abstand zwischen den Positionen der Blöcke, die die minimale Abweichung aufweisen.

Der Suchbereich in horizontaler Richtung im rechten Bild wird zusätzlich durch ein Disparitätslimit $d_{\max}$ beschränkt. $d_{\max}$ ergibt sich direkt aus der definierten Einsatzumgebung des Verfahrens. Je dichter Objekte der beobachteten Szene am Kamerasystem liegen, desto

größer ist die Disparität der zu dem Objekt gehörenden Pixel. Durch die Festlegung einer Mindestdistanz lässt sich $d_{\max}$ also leicht ermitteln.

Innerhalb des Suchbereiches (zwischen 0 und $d_{\max}$) wird der n×m-große Block punktweise verschoben. Der Verschiebungswert Δ für den die MSE-Funktion ihr Minimum annimmt, bestimmt den so genannten Blockdisparitätswert D. Ein Disparitätswert ist ein Vektor, welcher die Änderung der Lage eines Blockes (später eines Bildpunktes) zwischen den Bildern des Bildpaares beschreibt. Er ist nur dann eindeutig bestimmt, wenn die MSE-Funktion im Suchbereich ein eindeutiges Minimum besitzt. In den Fällen, in denen kein eindeutiges Minimum existiert, wird ein zusätzliches Entscheidungskriterium verwendet.

Unter der Annahme, dass sich die Disparitätswerte benachbarter Blöcke nur geringfügig unterscheiden, werden alle Disparitätswerte, für die die MSE-Funktion ein Minimum annimmt, mit dem Wert des benachbarten Blocks verglichen. Ausgewählt wird die Disparität mit dem geringsten Unterschied zu der Disparität des Nachbarblockes. Das Ergebnis der Anwendung des Blockmatchingverfahrens ist eine Disparitätsmatrix, in der jeweils Blöcke fester Größe einen identischen Wert besitzen [1].

5.2 Verfeinerung der Ergebnisse

Das Ergebnis der in Kapitel 5.1 aufgestellten Disparitätsmatrix lässt sich unter Verwendung eines Pixelselektionsverfahrens weiter verfeinern, sodass für jedes Pixel ein Disparitätswert bestimmt werden kann. Ein Verfahren hierfür ist zum Beispiel das Verfahren von T. Reuter [22] [1]. Dieses setzt sich aus 3 Verarbeitungsschritten zusammen:

1. Anwendung des Medianoperators auf die Disparitätswerte der Blöcke (3×3 Block) um einzelne Ausreißer in den Disparitätswerten zu eliminieren

2. Pixelselektion: hierbei wird die Disparität für jeden Pixel (x',y') eines Blockes unter Verwendung der Disparitätswerte dieses und der benachbarten Blöcke bestimmt. Dabei geht man folgendermaßen vor: man bildet die Differenzen DIFF(k) zwischen dem Intensitätswert des linken Bildes an der Position (x',y') und den Intensitätswerten des rechten Bildes an den Positionen (x' + D(k) ,y') für alle Disparitäten D(k) mit (1<k<9) aus der 3×3-Blockumgebung (siehe Abbildung 14):

$$\mathrm{DIFF}(k) = |E_r(x', y') - E_l(x' + D(k), y')| \qquad \text{mit } k=1,...,9. \qquad (10)$$

Der Disparitätswert DISP(x',y') ist definiert durch den Wert D(k), für den die Betragsdifferenz DIFF(k) ihr Minimum annimmt.

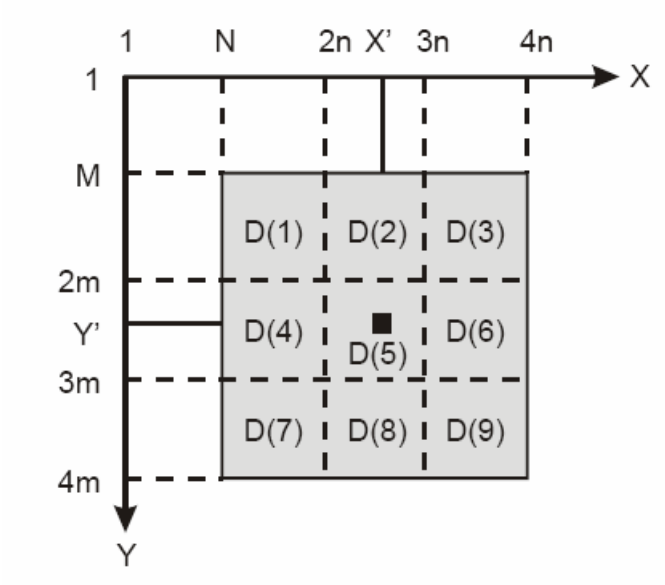

Abbildung 17: Pixelselektion

3. Anwendung des Medianoperators auf die Disparitätswerte, die für jedes einzelne Pixel bestimmt wurden.

Zum Schluss kommt dann eine Disparitätsmatrix heraus mit derselben Größe wie das Originalbild (siehe Abbildung 19).

Zusammenfassend hat man also folgenden Algorithmus (siehe Abbildung 18) [1]

```
begin
        unterteile linkes Bild in Blöcke der Größe n x m;
        for (jeden Block im linken Bild) do begin            {BM}
                initialisiere min und D;
                setze (x,y) gemäß linker oberer Ecke des aktuellen Blockes;
                for Δ := 1 to d_max do begin
                        berechne MSE (x, y, Δ);
                        if MSE (x, y, Δ) = min then
                                min := MSE (x, y, Δ);          D := Δ
                        else
                                if  MSE (x, y, Δ) = min then
                                        D := Disparität mit geringstem Unterschied zum
                                        Disparitätswert des Nachbarblockes
                                end {if}
                        end {if};
                        speichere Blockdisparitätswert
                end {for}
        end {for};
        filtere Matrix DISPARITÄT mit Medianoperator;
        for (jedes Pixel im linken Bild) do begin            {Pixelselektion}
                for k:=1 to 9 do begin berechne DIFF(k);
                        bestimme Wert D für den DIFF(k) minimal ist;
                        DISPARITÄT(Pixelposition) := D
                end {for}
        end {for};
        filtere Matrix DISPARITÄT mit Medianoperator
end
```

Abbildung 18: Block Matching Algorithmus mit anschließender Pixelselektion

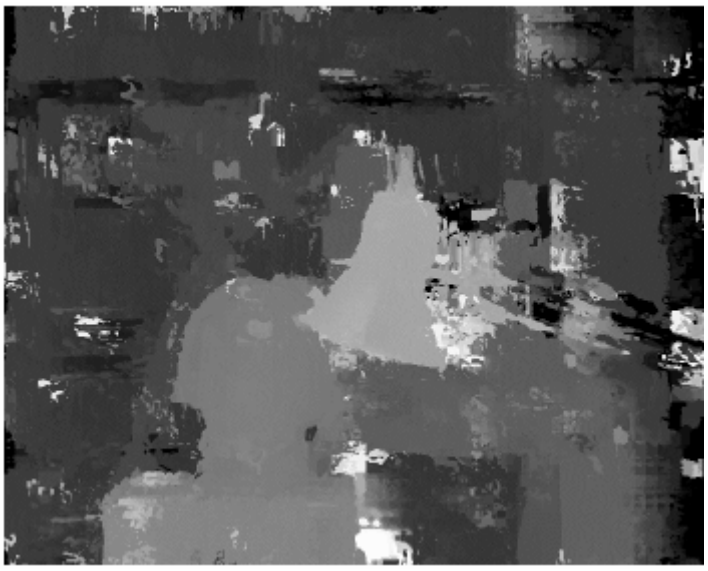

Abbildung 19: Disparitätsmatrix

Bei der Disparitätsmatrix ist der Betrag der Länge eines Disparitätsvektors in Graustufen abgebildet. Helle Grauwerte stellen längere Vektoren, dunkle Grauwerte kurze Vektoren dar. Das heißt je heller der Wert desto kürzer die Distanz vom Kamerasystem aus gesehen (siehe Abbildung 19).

5.3 Anytime Blockmatching

Anytime Algorithmen, welche eine Rolle im Zusammenhang mit zeitbeschränkten Verfahren spielen, müssen per Definition folgende Kriterien erfüllen [24]:

- Anytime Algorithmen können mit vernachlässigbarem Verwaltungsaufwand jederzeit unterbrochen und wieder aufgenommen werden
- Anytime Algorithmen können jederzeit ein Ergebnis zurückliefern
- Die Qualität des Ergebnisses steigt mit zunehmender Rechenzeit

Das in Abbildung 18 vorgestellte Blockmatching Verfahren muss, um ein Ergebnis zu liefern, einmal über das komplette Bild laufen. Das heißt, wird das Verfahren vorzeitig unterbrochen so steht kein Gesamtergebnis zur Verfügung und entspricht somit nicht den Anytime-Anforderungen.

Eine Lösung zu diesem Problem ist das Konzept der Gaußpyramide. Das heißt, dass ein Bildpaar zuerst auf eine geringere Auflösung heruntergerechnet wird. Dies geschieht indem man einen Tiefpassfilter, wie zum Beispiel den Binomialfilter, auf das Originalbild anwendet. Dadurch wird die Information im Bild reduziert und es entstehen so genannte Redundanzen zwischen den Pixeln, die es erlauben, genau diese Pixel ohne weiteren Informationsverlust zusammenzufassen. Die Anwendung eines 5×5 Binomialfilters mit anschließender Halbierung von Bildbreite und Bildhöhe, verkleinert die Bildgröße effektiv um Faktor 4. Der in dieser Arbeit verwendete Blockmatching Algorithmus von Richard Bade [1] benutzt eine Gauss Pyramide die sich aus 4 Ebenen zusammenstellt. Die Hinzunahme weiterer Ebenen macht wenig Sinn, da die Informationen bei mehr als 4 Ebenen schon so weit reduziert sind, dass während der Korrespondenzanalyse eine falsche Zuordnung der Blöcke sehr wahrscheinlich ist.

Das so skalierte Bildpaar (um den Faktor 64) kann der Blockmatching Algorithmus sehr schnell lösen, so dass bereits binnen kürzester Zeit ein Ergebnis vorliegt. Dieses hat natürlich nicht dieselbe Qualität, aber erfüllt die Anytime-Anforderungen. Nun werden nacheinander die darunterliegenden Ebenen (siehe Abbildung 20) durch das Verfahren bearbeitet bis hin zum Originalbild (hier Ebene1). Dabei wird die Disparitätsmatrix der aktuellen Stufe mit den Werten der Disparitätsmatrix der darunterliegenden, bereits berechneten Stufe initialisiert. Auf diese Weise verbessert sich die Qualität des Ergebnisses mit der Laufzeit und ist jederzeit abbrechbar.

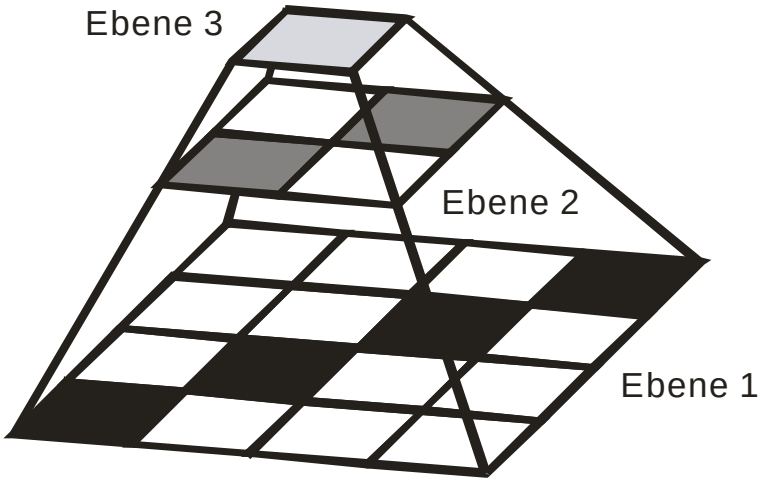

Abbildung 20: Schema einer 3-stufigen Gausspyramide

Um die Suche nach korrespondierenden Pixeln möglichst effizient zu gestalten ist es sinnvoll, den Suchraum so eng wie möglich zu gestalten. Wegen der achsenparallelen Kamerakonfiguration kann man den Suchraum auf horizontale Linien beschränken. Wie in Kapitel 5.1 bereits gesehen, kann der Suchbereich einschränkt werden indem man ein Disparitätsmaximum $d_{\max}$ einführt. Ein maximaler Disparitätswert entspricht einem Mindestabstand zwischen dem Kamerasystem und einem Objekt. Mittels der Formel (7) kann man diesen Mindestabstand ausrechnen. Bei der Halbierung von Breite und Höhe bei der Gausspyramide, halbiert sich $d_{\max}$ ebenfalls.

Außerdem ermöglicht die Initialisierung der aktuellen Disparitätsmatrix mittels der Disparitätsmatrix der vorherigen Stufe der Gausspyramide, die Suche nach einem korrespondierenden Block einzuschränken. Das heißt anstatt den Suchraum eines Blocks pixelweise von rechts nach links zu durchlaufen, kann man zuerst in der näheren Umgebung der zu erwartenden Lösung suchen. Nur wenn man da nichts findet wird in entfernteren Gegenden gesucht.

Das Ergebnis des Anytime Blockmatching Verfahrens in verschiedenen Auflösungsstufen kann in Abbildung 21 betrachtet werden:

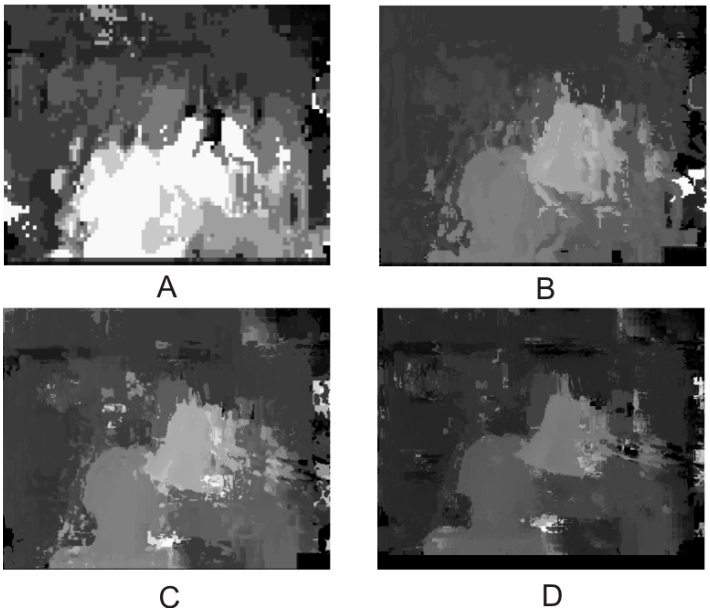

Abbildung 21: Disparitätsmatrix in verschiedenen Auflösungsstufen

6. Graph Cut Verfahren

Da man für die Erstellung eines Weltmodells, welches für die Berechnung der Fussauftrittspunkte dienen soll, dichte Tiefenkarten braucht, soll in diesem Kapitel gezeigt werden wie man diese mit Hilfe des Graph Cut Verfahrens berechnet. Ferner soll untersucht werden ob dieser Ansatz schnell genug ist um Echtzeitfähigkeit zu erfüllen.

Das Graph Cut Verfahren versucht eine optimale Zuordnung der Pixel im Referenz- und Suchbild durch Minimierung einer globalen Kostenfunktion zu erreichen. Dabei versucht es, im Gegensatz zum Dynamischen Programmieren, das die Korrespondenzen nur zeilenweise sucht, auch die 3.te Dimension mit einzubeziehen. Es werden also Nachbarschaften sowohl entlang als auch quer zu den Bildzeilen mit einbezogen. Dieses geschieht durch den Aufbau eines Graphen, dessen Kosten man versucht zu minimieren. Diese Kosten sind in Form von Energien dargestellt.

Es werden jetzt einige grundlegende Begriffe vorgestellt.

6.1 Definitionen aus der Graphentheorie

Ein **Graph** besteht aus einer Menge von Knoten v und einer Menge von Kanten ε. Der hier verwendete Graph besitzt gerichtete Kanten, sie sind daher geordnete Paare von Knoten:

$$e \in \varepsilon, \quad e = (v_i, v_j), \quad v_i, v_j \in v \tag{11}$$

Eine Kante *e* hat eine **Kapazität** $c(e) \in \mathrm{N}_0^+$.

Sei *G* ein Graph mit Knoten $v = \{v_1,...,v_n,s,t\}$ und Kanten $\varepsilon = \{e_1,...,e_m\}$ mit Kapazitäten $c_1,...,c_m$. Ein s-t-**Schnitt** ist eine Partitionierung der Knotenmenge v in zwei Teilmengen *S* und *T*, so dass $s \in S$ und $t \in T$.

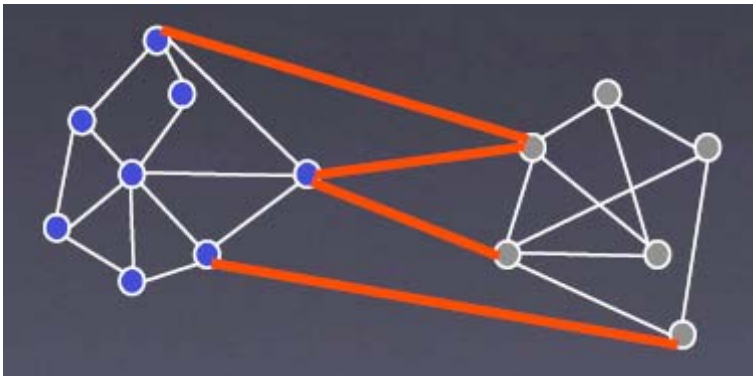

Abbildung 22: Graph Cut

Die **Kosten** $c_{cut}(S, T)$ eines s-t-Schnittes berechnen sich als die Summe der Kapazitäten aller Kanten welche S von T trennen.

$$c_{cut}(S, T) = \sum_{u \in S, v \in T, (u,v) \in \varepsilon} c(u,v). \qquad (12)$$

Der minimale s-t-Schnitt ist der Schnitt mit den geringsten Kosten.
Es sei mit ε_j^+ die Menge der ausgehenden Kanten von v_j bezeichnet und mit ε_j^- die Menge der eingehenden Kanten. Der Knoten s wird als **Quelle** und t als **Senke** bezeichnet. Dann ist der ganzzahlige **Fluss** f wie folgt definiert: Für jede Kante $e_j \in \varepsilon$ gilt:

$$0 \leq f(e_j) \leq c_j, \qquad (13)$$

und für jeden Knoten v' gilt:

$$\sum_i f(v', v_i) = \sum_j f(v_j, v') \text{ mit } (v', v_i) \in \varepsilon'^+ \text{ und } (v_j, v') \in \varepsilon'^-. \quad (14)$$

Des weiteren gilt für die Quelle s: $\sum_i f(s, v_i) \geq \sum_j f(v_j, s)$ und für die Senke t:
$\sum_i f(t, v_i) \leq \sum_j f(v_j, t)$. Der **Wert** des Flusses F auf G ist:

$$F = \sum_i f(s, v_i) - \sum_j f(v_j, s) \text{ mit } (s, v_i) \in \varepsilon_s^+ \text{ und } (v_j, s) \in \varepsilon_s^- \qquad (15)$$

6.2 Energieminimierung

Durch Energieminimierung wird eine möglichst gute Tiefenkarte erstellt. Dabei wird eine Hypothese über den Disparitätswert aufgestellt. Dann werden für mehrere Hypothesen Fehlerenergien aufgestellt. Durch die Minimierung der Fehlerenergie werden diejenigen Hypothesen bestimmt, welche mit den vorhandenen Daten und Modellannahmen am besten übereinstimmen.

6.2.1 Berechnung der Gesamtfehlerenergie

Die Gesamtfehlerenergie setzt sich aus zwei Teilen zusammen: dem Datenterm und dem Glattheitsterm.

Der **Datenterm** E_{data} setzt die Annahme der Photokonsistenz um. Zu einem Pixel wird eine Disparitätshypothese aufgestellt und so ein korrespondierendes Pixel bestimmt. Mit Hilfe eines Ähnlichkeitsmaßes (wie zum Beispiel das MSE aus Kapitel 5.1) wird die Ähnlichkeit in Helligkeit und Farbe zweier Pixel bestimmt. Bilden sie den gleichen Szenepunkt ab, so müssen sie wegen der Photokonsistenz der Szene gleiche Pixelwerte besitzen.

Der **Glattheitsterm** E_{smooth} setzt die Annahme der stückweisen Glattheit um. Dabei wird der Tiefenunterschied benachbarter Pixel als Bewertungskriterium für die Glattheit genommen. Nachbarschaft kann dabei beliebig definiert werden, wird jedoch meistens als direkte Vierernachbarschaft gewählt. Ein Tiefenunterschied von 0 bedeutet perfekte Glattheit, die zugehörige Fehlerenergie muss als 0 sein und mit wachsendem Tiefenunterschied ansteigen. Mit dieser Modellierung entstehen jedoch an Objektkanten der Szene Probleme: Dort sind große Tiefensprünge zu erwarten und sollen auch in der Tiefenberechnung nicht als Fehler bewertet werden. Der Glattheitsterm muss diesem Umstand Rechnung tragen. Dies wird durch die Einführung eines Schwellwertes erreicht. Übersteigt die Tiefendifferenz den Schwellwert, so steigt die Fehlerenergie nicht weiter an oder vermindert sich sogar.

6.2.2 Wahl der Energiefunktionen

Kolmogorov und Zabih schlagen in [19] die Benutzung von Energietermen von binären Variablen vor weil diese mittels Graph-Cut Verfahren schnell minimierbar sind. Die binären Variablen gestatten die Minimierung für nur zwei Hypothesen. Durch die Zerlegung des Gesamtproblems in Teilprobleme, welche jeweils nur zwei Hypothesen verwenden, wird diese Einschränkung umgangen.

Die Aufstellung der Energieterme erfolgen für alle Pixel eines Bildes und die Gesamtenergiefunktion *E* kann folgendermaßen geschrieben werden:

$$E = E_{data}(x_i) + E_{smooth}(x_i, x_j) \qquad (16)$$

Jede binäre Variable x_i kann die Werte 0 oder 1 annehmen, wir schreiben entsprechend $E(0)$ oder $E(1)$. Es ist allerdings nicht jede beliebige Energiefunktion mittels Graph-Cut Verfahren minimierbar, sondern es gibt eine Bedingung für die Graphrepräsentierbarkeit von Energiefunktionen auf binären Variablen die laut Kolmogorov und Zabih folgendermaßen lautet:

Therorem 1 (Kolmogorov, Zabih) *Sei E eine Funktion von n binären Variablen, die als Summe von Funktionen mit höchstens zwei Variablen geschrieben werden kann:*

$$E(x_1, ..., x_n) = \sum_i E^i(x_i) + \sum_{i<j} E^{i,j}(x_i, x_j) \qquad (17)$$

Dann ist E graphrepräsentierbar, wenn jeder Term E^{ij} die folgende Ungleichung erfüllt:

$$E^{ij}(0,0) + E^{ij}(1,1) \leq E^{ij}(0,1) + E^{ij}(1,0) \qquad (18)$$

Den Beweis für das Theorem 1 findet man in [21].

6.2.3 Aufstellen der Energieterme

Der **Datenterm** verwendet die maximumbegrenzte, quadrierte Intensitätsdifferenz der korrespondierenden Pixel. Der Term lautet:

$$E_{data}(x) = \min\{0, (Intensity(x) - Intensity(M(x)))^2 - K_{data}\} \qquad (19)$$

Darin ist K_{data} eine Konstante und $M(x)$ eine Funktion, welche zu Pixel x korrespondierenden Partnerpixel angibt. x und $M(x)$ müssen natürlich aus unterschiedlichen Kamerabildern stammen.

Der **Glattheitsterm** kommt in zwei verschiedenen Varianten zum Einsatz.

Die **Rampenfunktion** beschreibt eine Tiefendifferenz zwischen Nachbarpixeln und legt fest ab wann die Existenz einer Objektkante angenommen wird. Bis zu dieser Grenze steigt die Funktion linear an, darüber verläuft sie konstant:

$$E_{smooth}^{ramp}(x_k, x_l) = \min(|l_k - l_l|, K_{smooth}) \qquad (20)$$

Dabei ist K_{smooth} eine Konstante.

Das **Intensitätsmodell** basiert auf der aktuellen Konfiguration und zusätzlich auf den Pixelintensitäten. Bei gleichen Disparitätswerten ergibt sich eine Fehlerenergie von 0. Bei unterschiedlicher Tiefe wird die Differenz der Pixelintensitäten betrachtet. Ist sie geringer als ein Schwellwert wird angenommen, die Pixel gehörten demselben Objekt an und sollten folglich den gleichen Tiefenwert besitzen. Es wird daher eine hohe Fehlerenergie erzeugt. Andernfalls werden die Pixel als zu unterschiedlichen Objekten gehörig betrachtet und eine geringere Fehlerenergie zugeordnet:

$$E_{smooth}^{\text{int}ensity}(x_k, x_l) = \begin{cases} 0 & wenn \quad l_k = l_l \\ U(x_k, x_l) & sonst \end{cases} \qquad (21)$$

$$\text{wobei } U(x_k, x_l) = \begin{cases} 2\lambda & wenn \quad \Delta Intensity \le 4 \\ \lambda & sonst \end{cases}$$

Dabei ist λ eine Konstante.

6.3 Das Min Cut / Max Flow Verfahren

Nach dem Theorem von Ford und Fulkerson [6] ist die Bestimmung eines maximalen Flusses auf einem Graphen äquivalent zur Bestimmung des minimalen Schnittes. Die Bestimmung des maximalen Flusses ist mit effizienten Algorithmen lösbar und wird deshalb verwendet. Der maximale Fluss $F_{\max}$ auf G ist ein Fluss, dessen Wert nicht mehr erhöht werden kann. Um diesen Sachverhalt besser beschreiben zu können, sei zunächst ein **f-augmentierender Pfad** P definiert. Ein Pfad von v_1 nach v_{k+1} ist eine Menge von miteinander verbundenen Knoten

und Kanten $\{v_1, e_1, v_2, ..., e_k, v_{k+1}\}$. Er ist f-augmentierend, wenn für jede Kante e_j des Pfades gilt:

$$f(e_j) < c_j \text{ falls } e_j = (v_j, v_{j+1})$$
$$f(e_j) > 0 \text{ falls } e_j = (v_{j+1}, v_j) \qquad (22)$$

Der Wert des Flusses *F* auf *G* kann erhöht werden, solange ein f-augmentierender *s*-*t* Pfad auf *G* existiert. Ist der Fluss $F_{\max}$ gefunden, so gibt es auf jedem *s*-*t* Pfad eine Kante, auf welcher der Fluss nicht mehr erhöht (sofern sie in Flussrichtung verläuft) oder vermindert werden kann (sofern sie entgegen der Flussrichtung verläuft). Die Menge dieser Kanten repräsentiert einen Schnitt minimaler Kapazität *G*.

Ein **Max-Flow-Algorithmus** wird nun im folgenden vorgestellt:
Der Algorithmus durchläuft drei Phasen solange, bis ein Abbruchkriterium erreicht ist: die Wachstumsphase, die Verstärkungsphase und die Adoptionsphase.

1. **Wachstumsphase:** Der Suchbaum *S* wächst, bis die Senke *T* erreicht ist und somit ein f-augmentierender *s*-*t* Pfad gefunden wurde.

2. **Verstärkungsphase:** Der Fluss auf dem gefundenen Pfad wird maximal verstärkt. Dadurch zerfällt *S* in einen Wald.

3. **Adoptionsphase:** Der Wald wird wieder zu einem Baum zusammengefügt

Zunächst wird ein Suchbaum *S* aufgebaut, dessen Wurzel die Quelle *s* ist. Er dient der Bestimmung f-augmentierender *s*-*t* Pfade. Alle Kanten in *S* vom jeweiligen Vaterknoten zu seinem Kindknoten sind ungesättigt. Alle anderen Knoten gehören der Menge *T* der freien Knoten an. Die Knoten in *S* sind unterteilt: Solche, die keine ungesättigten Kanten zu freien Knoten besitzen, heißen passiv. Alle anderen können noch Kindknoten aus der Menge freier Knoten bekommen, sie werden als aktiv bezeichnet. In der Wachstumsphase wächst der Suchbaum, die aktiven Knoten bekommen neue Kindknoten, welche dann aktive Knoten von *S* werden. Sobald $t \in S$ ist, endet die Wachstumsphase.

In der Verstärkungsphase wird der Fluss auf dem gefundenen Pfad maximal erhöht. An den dadurch gesättigten Kanten zerfällt *S*. Die jeweiligen Kindknoten dieser Kanten sind nun Wurzelknoten von Teilbäumen.

In der Adoptionsphase wird für jeden dieser neuen Wurzelknoten ein gültiger Vaterknoten gesucht. Wird keiner gefunden, wird der Wurzelknoten der Menge *T* zugeteilt, seine Kindknoten werden neue Wurzelknoten. Die Adoptionsphase endet, wenn alle Teilbäume wieder zusammengefügt sind. Nach dem Abschluss der Adoptionsphase beginnt die nächste Wachstumsphase. Der Algorithmus terminiert, sobald es während einer Wachstumsphase keine aktiven Knoten mehr gibt, bevor die Senke gefunden wurde.

6.4 Minimierbarkeit mittels Graph Cut

Laut Kolmogorov und Zabih [19] gilt die folgende Definition:

Definition 1 *Eine Funktion E von n binären Variablen heißt graphrepräsentierbar wenn es einen Graphen* $G = (v, \varepsilon)$ *gibt mit Quelle s und Senke t und einer Knotenuntermenge* $v_0 = \{v_1,...v_k\} \subset v \setminus \{s,t\}$, *so dass für jede Konfiguration* $x_1,...,x_n$ *der Wert* $E(x_1,...,x_n)$ *gleich einer Konstanten plus den Kosten c des minimalen s-t-Schnittes von G ist, bei dem* $v_i \in S$ *wenn* $x_i = 1$. *E ist exakt repräsentierbar von G und* v_0, *wenn die Konstante 0 ist.*

Diese Definition stellt sicher, dass das Minimum einer graphrepräsentierbaren Funktion *E* durch die Bestimmung eines minimalen *s*-*t*-Schnittes des *E* repräsentierenden Graphen gefunden werden kann.

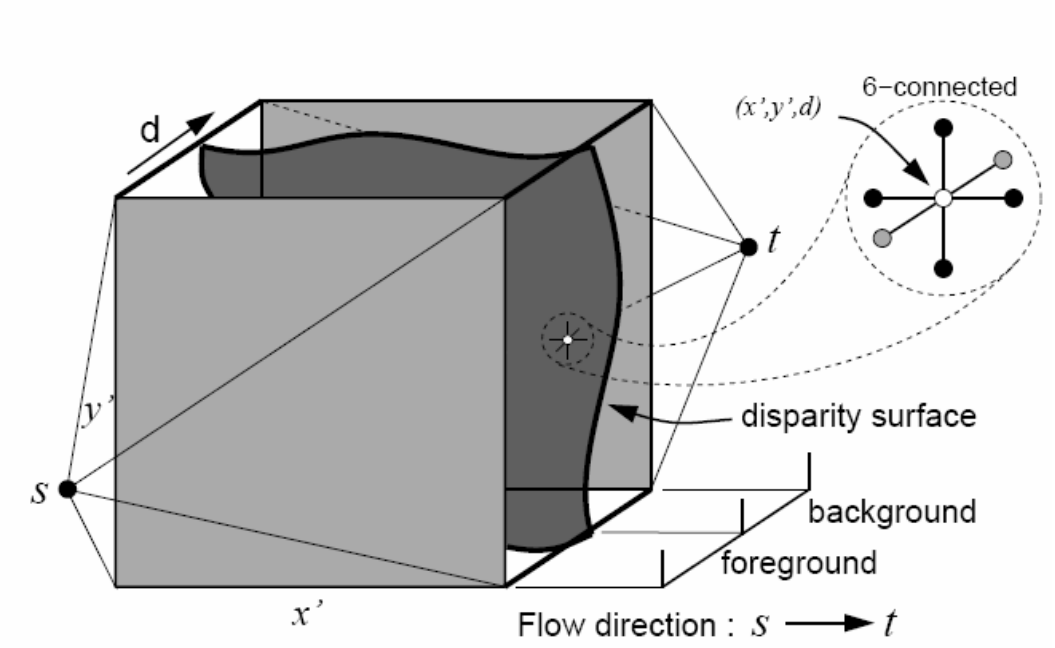

Abbildung 23: Maximaler Fluss / Minimaler Schnitt Problem

Abbildung 23 [23] zeigt die Korrespondenzsuche als ein Maximaler Fluss Problem. „Foreground“ und „Background“ entsprechen dem nahesten und entferntesten Tiefenlabel. Die „disparity surface“ beschreibt den Schnitt.

6.5 Graphaufbau

In den vorhergehenden Kapiteln wurden die Gesamtfehlerenergie und ihre Minimierbarkeit mittels Graph-Cut Verfahren gezeigt. Außerdem wurde ein Max-Flow-Algorithmus zur Graphminimierung angegeben. Diese Ergebnisse werden nun zusammengeführt, indem ein konkreter Graphaufbau beschrieben und die Berechnung der Tiefenkarte aus mehreren Einzelschritten, den sogenannten α -Expansion-Moves, gezeigt wird.

6.5.1 α -Expansion-Move

Die Berechnung einer Tiefenkarte bedeutet die Zuordnung eines Tiefenlabels *l* zu jedem Pixel eines Bildes. Die Tiefenlabel werden dabei wie im Kapitel Grundlagen gezeigt aus den Disparitätswerten berechnet. Also jedem Tiefenlabel kann sozusagen ein Disparitätswert zugeordnet werden.

Die Zuordnung der Tiefenlabel zu den Pixeln wird bestimmt indem man durch das Aufstellen aller möglichen Tiefenhypothesen für jedes Pixel die beste Hypothese auswählt. Werden in einer Energiefunktion die Fehlerenergien der Hypothesen zusammengefasst, so liefert die Minimierung der Energiefunktion die besten Hypothesen und damit die Tiefenkarte. Hier benutzt man Energiefunktionen von binären Variablen x_i. Damit können maximal zwei Hypothesen kodiert werden. Und zwar:

- Das Pixel j behält seinen bisherigen Tiefenlabel l_j bei, dann ist $x_j = 0$.
- Dem Pixel j wird ein neuer Tiefenlabel α zugeordnet, dann ist $x_j = 1$.

Durch die Graph-Cut-Minimierung wird der optimale Wert von x_j bestimmt. Eine Zugehörigkeit von x_j zum Quellen- oder Senkensegment zeigt den Ergebniswert der Minimierung:

- x_j wird dem Quellensegment *S* zugeordnet für $x_j = 0$.
- x_j wird dem Senkensegment *T* zugeordnet für $x_j = 1$.

6.5.2 Teilgraph eines Pixelpaares

Der Teilgraph für ein Paar benachbarter Pixel mit zwei Knoten v_k, v_l, Quelle *s*, Senke *t* und drei Kanten aus ihrer Vereinigung ist wie folgt definiert:

$$e_1 = (v_k, v_l) \quad \text{und} \quad c(e_1) = E(0,1) + E(1,0) - E(0,0) - E(1,1)$$

$$\begin{aligned} e_2 &= (s, v_k) && \text{wenn } E(1,0) > E(0,0), && \text{mit } c(e_2) = E(1,0) - E(0,0) \\ e_2 &= (v_k, t) && \text{wenn } E(1,0) \le E(0,0), && \text{mit } c(e_2) = E(0,0) - E(1,0) \end{aligned}$$

$$\begin{aligned} e_3 &= (v_l, t) && \text{wenn } E(1,0) > E(1,1), && \text{mit } c(e_3) = E(1,0) - E(1,1) \\ e_3 &= (s, v_l) && \text{wenn } E(1,0) \le E(1,1), && \text{mit } c(e_3) = E(1,1) - E(1,0) \end{aligned}$$

Der Beweis kann in [21] nachgelesen werden.

6.5.3 Aufbau des Gesamtgraphen

Die Teilgraphen aller Pixelpaare eines Bildes ergeben in ihrer Vereinigung den Gesamtgraphen des Bildes. Für die beiden Eingabebilder wird jeweils ein solcher Graph berechnet. Sie werden gemeinsam als ein Graph minimiert. Die Minimierung der Bildgraphen komplettiert einen α-Expansion-Move. Ein Lauf von α-Expansion-Moves über alle möglichen Tiefenlabel nennt man Zyklus. Das Verfahren ist beendet, sobald innerhalb eines Zyklus kein Pixel mehr seine Konfiguration ändert. Damit ist die Tiefenkarte fertig berechnet.

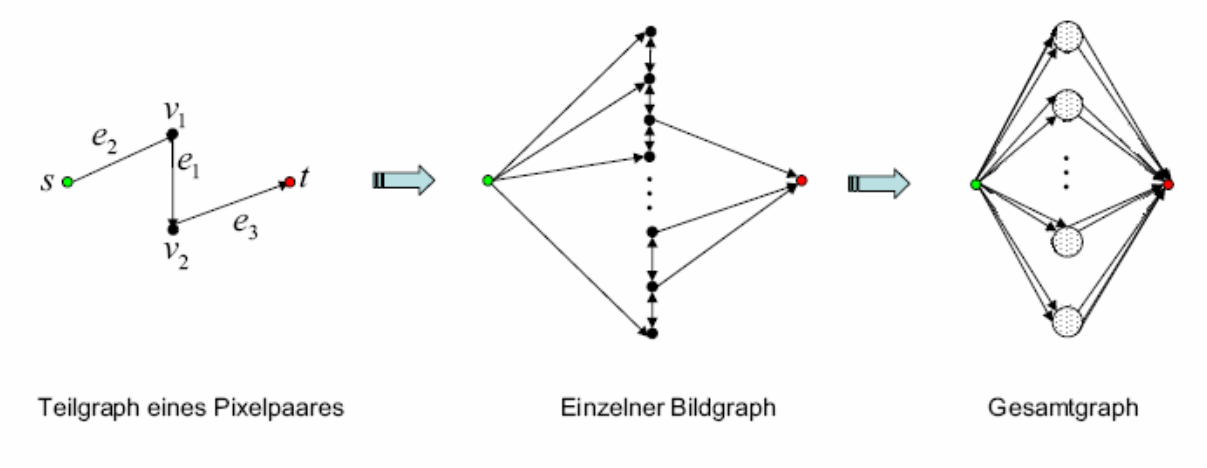

Abbildung 24: Zusammensetzung von Teilgraphen zu einem Gesamtgraph

6.6 Laufzeit

Die Laufzeit des Graph Cut Verfahren setzt sich zusammen aus der Laufzeit des Max-Flow Algorithmus und der Erstellung des Gesamtgraphen.
Seien C die Kosten eines minimalen Schnittes, n die Anzahl der Knoten, m die Anzahl der Kanten. Es ergibt sich also durch den Max-Flow Algorithmus eine Komplexität von $O(mn^2 \mid C \mid)$. Da nur maximal 2 Kanten von einem Knoten ausgehen können ergibt sich $O(n^4 \mid C \mid)$.

Sei l die Anzahl der Tiefenlabel. Da der Gesamtgraph maximal l Tiefenlabel haben kann ist die Gesamtkomplexität also $O(n^4 \mid C \mid l)$.

Wie schon beim Anytime Blockmatching, ist es auch hier möglich das Verfahren zu beschleunigen indem man den Gausspyramiden Ansatz benutzt. Da jeder Pixel im Bild sozusagen einen Knoten repräsentiert, verringert sich die Laufzeit bei geringerer Auflösung.

7. Testumgebung

In diesem Kapitel wird vorgestellt wie man die Qualität der Stereobildverarbeitungs-algorithmen testen und bewerten kann, und in welcher Umgebung dies hier geschieht.

7.1 Erstellung von Testszenarien

In diesem Kapitel wird beschrieben wie man mit Hilfe eines Stereokamerasystems Bilder erstellen kann, welche als Eingabe für die Stereobildverarbeitungsalgorithmen dienen.

7.1.1 Versuchsaufbau

Der Versuchsaufbau wurde, wie in Abbildung 25 zu sehen, mit 2 Phytec Firewire Cam-002 realisiert [14]. Dabei handelt es sich um Kameras mit einer maximalen Auflösung von 640x 480 Pixeln bei einer Framerate von bis zu 30 Bildern/sec. Die Pixelgrösse horizontal sowie vertikal ist 5,6 µm. Die Brennweite der verwendeten Objektive beträgt 8 mm.

Abbildung 25: Versuchsaufbau mit 2 Phytec Firewire Cam-002

Dabei wurden die beiden Kameras auf Aluminium Platten festgeschraubt, die wiederum an einer Stange befestigt sind. Durch hin und herschieben der Platten auf der Stange kann man den Abstand zwischen den beiden Kameras einstellen (Minimum 16 cm, Maximum 100 cm). Es lassen sich außerdem Winkel für die Kameras einstellen (-45° bis +45°). In dieser Arbeit wurde der Abstand auf 16 cm und die Winkel auf 0° festgelegt.

7.1.2 Bildaufnahme

Die Bildaufnahme wurde mit Hilfe der Acquisition Tool Box von Matlab 7.1 gemacht. Hierbei wurde ein Programm geschrieben das ein Objekt der linken sowie ein Objekt der rechten Kamera erzeugt, von denen dann nacheinander eine Schnappschuss gemacht wird, die schlussendlich im PPM oder im PGM Format, je nachdem welche Parameter man setzt, abgespeichert werden.

Beim PPM (engl. **P**ortable **P**ix**M**ap) handelt es sich um ein Dateiformat zur Speicherung von Bilddaten, das in den achtziger Jahren von Sun Microsystems entwickelt wurde.

Name	**P**ortable **P**ix**M**ap
Abkürzung	PPM
Dateiendung	.ppm
Hersteller	Sun Microsystems
Farbraum	RGB
Farbtiefe	24 Bit, 16,7 Mio Farben (binär)
Kompression	keine

Tabelle 2: PPM Daten

Beim PGM (engl. **P**ortable **G**ray**M**ap) handelt es sich um ein proprietäres Dateiformat zur Speicherung von Bilddaten, das in den achtziger Jahren von Sun Microsystems entwickelt wurde.

Name	**P**ortable **G**ray**M**ap
Abkürzung	PGM
Dateiendung	.pgm
Hersteller	Sun Microsystems
Farbraum	Graustufen
Farbtiefe	8 Bit, 256 Graustufen (binär)
Kompression	keine

Tabelle 3: PGM Daten

7.1.3 Bildvorverarbeitung

Nach der Bildaufnahme kann es bei der Übertragung des Bildes in den Rechner zu Störungen kommen, die die Qualität des Bildes beeinflussen. In diesem Kapitel werden diese kritischen Stellen vorgestellt und einige Lösungen dazu vorgeschlagen.

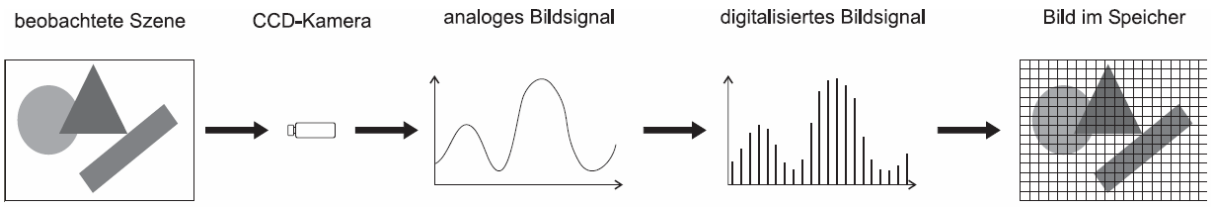

Abbildung 26: Bildaufnahme und Digitalisierungsweg

Die erste Störstelle ist der CCD Chip der Kamera. Hier können Pixelfehler im Bild entstehen, durch das Auftreten einzelner fehlerhafter Sensorelemente. Auch produzieren CCD Sensoren bei geringer Beleuchtungsstärke ein Farbrauschen.

Eine weitere Störstelle ist die analoge Übertragung des Bildsignals von der Kamera zum Rechner. Hier kann die Bildqualität durch den Leitungswiderstand oder elektromagnetische Strahlung beeinflusst werden.

Eine dritte Störung tritt bei der Digitalisierung der Bilder in der Matlab Acquisition Toolbox auf. Dazu gehören z.B Quantisierungsfehler. Diese Störungen stellen in der Regel ein hochfrequentes Rauschen dar, das heißt die Intensitätswerte der Pixel sind verändert [1].

Verwendete Filter

Damit die Stereobildverarbeitung durch fehlerhafte Pixel nicht negativ beeinflusst wird, versucht man diese Pixelfehler durch verschiedene Filter zu eliminieren. Für das hochfrequente Bildrauschen benutzt man in der Regel lineare verschiebungsinvariante Filter, für einzelne Pixelausreißer hingegen nichtlineare Filter, wie zum Beispiel den Medianfilter.

Medianfilter

Der Medianfilter ist ein nichtlinearer Filter der zur Klasse der Rangordnungsfilter gehört. Beim Einsatz in der Bildverarbeitung werden die Grauwerte der Pixel innerhalb einer definierten Umgebung eines Pixels im Zentrum dieser Umgebung nach ihrer Größe sortiert, wobei der Wert des zu ersetzenden Pixels in diese Berechnung mit einfließt. Der mittlere Wert der sortierten Liste wird zurückgegeben und der Wert des zentralen Pixels wird durch ihn ersetzt (siehe Abbildung 27) [1] [27].

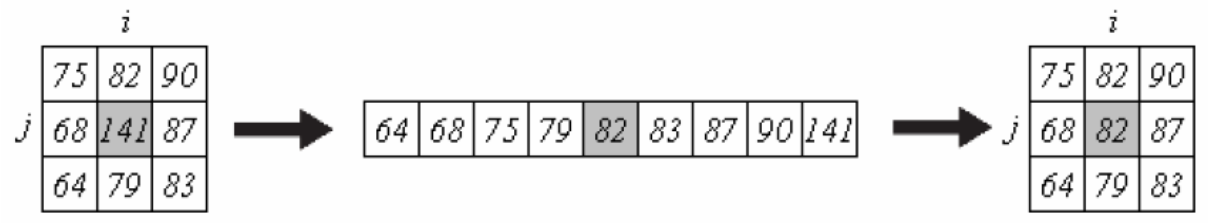

Abbildung 27: Medianfilter

Aus seiner Funktionsweise lässt sich schließen dass ein Medianfilter also gut für die Korrektur von Ausreißern bei den Pixeln, sei es durch Übertragungsfehler oder defekte Pixel, geeignet ist.

Abbildung 28: Median-Bildfilterung eines Bildes mit weißen Störpixeln

Die Anzahl der Operationen, die sowohl für eine Faltung, als auch für den Medianfilter auszuführen sind, ist abhängig von der Größe der Filtermatrix.

Lineare verschiebungsinvariante Filter

Zweidimensionale lineare Filter werden durch eine Filtermaske $g(u,v)$ der Größe $M \times N$ beschrieben. Aus Symmetriegründen werden in der Regel Filtermasken ungerader Maskengröße bevorzugt. Die Filterung eines Bildes ergibt sich durch Faltung des Bildes mit der Filtermaske. Eine Faltung ist dabei eine Operation, die eine Bildbereich auf einen Pixel abbildet. Das Bild sowie der Faltungskern sind als zweidimensionale Matrix zu betrachten (siehe Abbildung 29). Die zweidimensionale Faltung lautet [27]:

$$y(u,v) = \sum_{m=-M/2}^{m=M/2} \sum_{n=-M/2}^{n=N/2} x(u-m, v-n) \cdot g(m,n) \qquad (23)$$

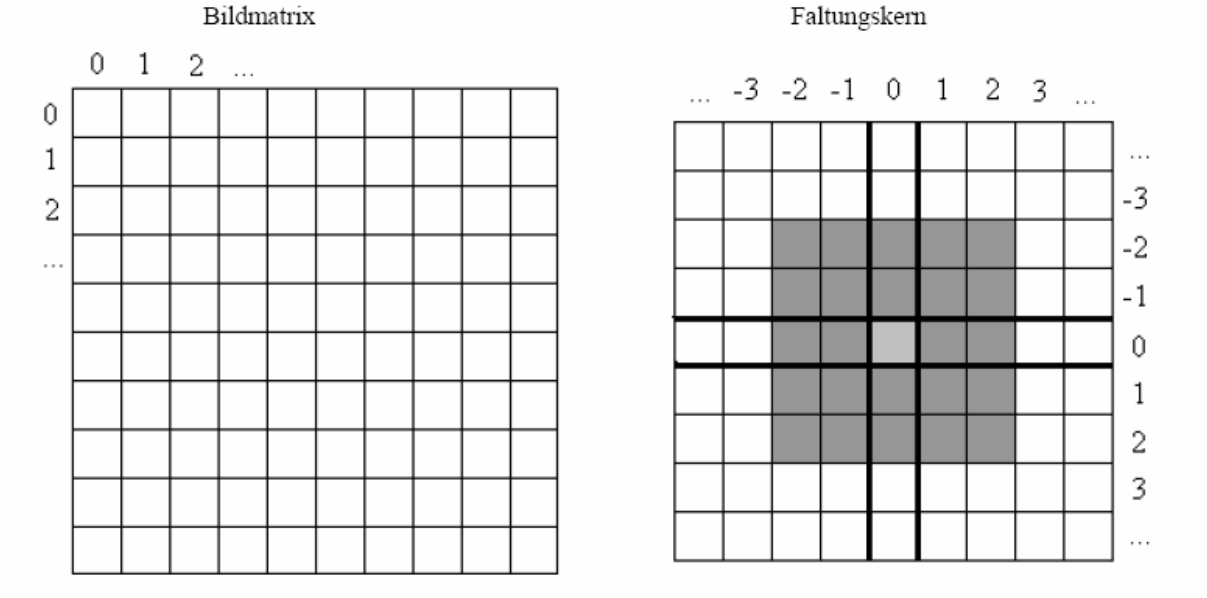

Abbildung 29: Bildmatrix und Faltungskern

Dabei wird der Faltungskern mit dem Zentrum über das zu verändernde Bildpixel gelegt, welches dann entsprechend der Filterwerte neu berechnet wird. Jedes Element der Matrix des Faltungskerns ist dabei ein Gewicht, das darstellt wie stark der jeweilige Bildpixel an dieser Position, das Zielpixel im Zentrum beeinflusst. Der Faltungskern wird für die Faltung des ganzen Bildes sukzessive Pixel für Pixel über das gesamte Bild geschoben.

Ein typischer Filter aus dieser Klasse ist der Binomialfilter. Es handelt sich dabei um einen sogenannten Tiefpassfilter oder Glättungsfilter der dazu eingesetzt wird das hochfrequente Rauschen im Bild zu verringern. Die Binomialfilter der Größe 3×3 bzw 5×5 sehen folgendermaßen aus:

$$g(u,v) = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} / 16 \qquad \text{bzw.} \qquad g(u,v) = \begin{pmatrix} 01 & 04 & 06 & 04 & 01 \\ 04 & 16 & 24 & 16 & 04 \\ 06 & 24 & 36 & 24 & 06 \\ 04 & 16 & 24 & 16 & 04 \\ 01 & 04 & 06 & 04 & 01 \end{pmatrix} / 256$$

7.2 Qualitative Auswertung der Ergebnisse

Um die Performanz von Stereobildverarbeitungsalgorithmen zu erfassen gibt es verschiedene Qualitätsmetriken die man benutzten kann. Zum einen ist das der Faktor Zeit, also wie lange ein Algorithmus braucht (Rechenzeit) um aus einem Bildpaar eine Disparitätskarte zu erstellen, zum anderen sind das optische Bewertungskriterien oder Vergleiche.

7.2.1 Optische Bewertung

Eine erste Bewertung kann mit Hilfe des menschlichen optischen Eindrucks gewonnen werden. Folgende Kriterien sind dabei zu beachten:

- Anzahl der punktförmigen, starken Abweicher, den sogenannten Ausreissern
- Erscheinen die Flächen homogen?
- Sind an den Objektkanten Haloeffekte feststellbar?
- Wie genau sind die Objektkanten abgebildet?

Dieser Ansatz leistet zwar zunächst gute Dienste, er ist jedoch subjektiv geprägt und nicht sehr exakt.

7.2.2 Ground-Truth Methode

Bei verschiedenen Stereobildpaaren sind die wahren Tiefenwerte bzw Disparitäten bereits bekannt und in Form von einer Disparitätskarte vorhanden (zum Beispiel beim Stereobildpaar der Universität von Tsukuba: siehe Abbildung 30).

Abbildung 30: Ground-Truth Karte

So kann man mit Hilfe der wahren Disparitätskarte folgende Qualitätsmasse errechnen [24]:

1. RMS (root-mean-squared) error (in Disparitätsunitäten gemessen) zwischen der vom Stereobildverarbeitungsalgorithmus errechneten Disparitätskarte $d_C(x,y)$ und der wahren Disparitätskarte $d_T(x,y)$

$$R = \left(\frac{1}{N} \sum_{(x,y)} | d_C(x,y) - d_T(x,y) |^2 \right)^{\frac{1}{2}} , \qquad (24)$$

   wobei N für die Gesamtzahl der Pixel steht.

2. Prozent der Pixel die nicht übereinstimmen (bad matching pixels)

$$B = \frac{1}{N} \sum_{(x,y)} \left(|d_C(x,y) - d_T(x,y)| > \delta_d \right) , \qquad (25)$$

wobei δ_d für die Disparitätsfehler Toleranz steht. Ein guter Wert hierfür wäre 1.
N steht für die Gesamtzahl der Pixel.

Man kann zusätzlich zu den Statistiken die man über das ganze Bild berechnen kann, sich auf einzelne Regionen im Bild beschränken. Hier gibt es drei interessante Gebiete, nämlich die texturlosen Regionen, die überdeckten Regionen und die tiefendiskonuitäten Regionen. (siehe Abbildung 31: Das erste Bild zeigt texturfreie Regionen (weiss) und überdeckte Regionen (schwarz) ; das zweite Bild überdeckte Regionen (schwarz), tiefendiskontinuierliche Regionen (weiss))

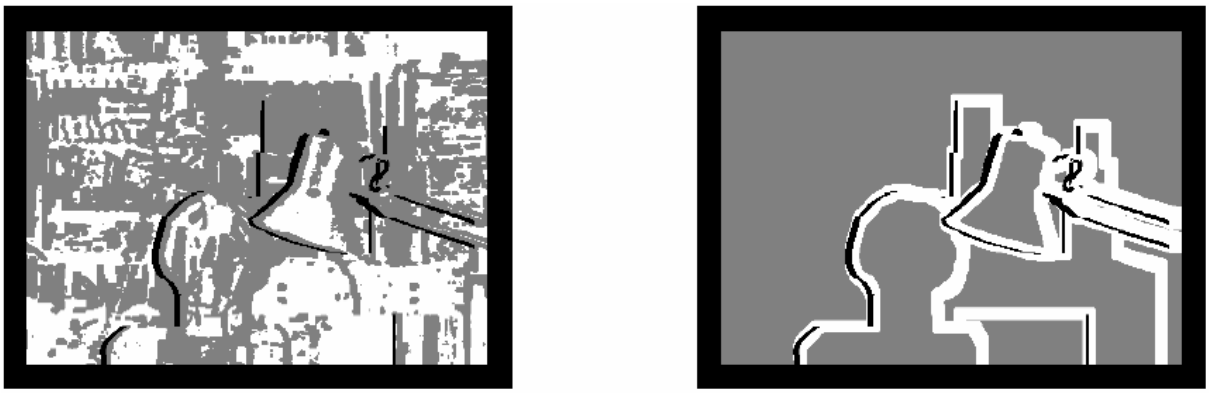

Abbildung 31: texturfreie, überdeckte, tiefendiskontinuierliche Regionen

7.3 Implementierung

Die Implementierungen der beiden Stereobildverarbeitungsalgorithmen Blockmatching und Graph Cut wurden in C++ durchgeführt [1] [17] und ensprechend der Testumgebung angepasst. Objektorientierte Konzepte wurden auf Grund der dadurch einfacheren Modularisierung der Komponenten in Hinblick auf eine spätere Weiterentwicklung verwendet. Da der Einsatz von new und delete Operatoren auf Grund fehlenden Speichermanagements in Echtzeitsystemen nicht erfolgen kann, fanden alle Speicheranforderungen beim Anytime Blockmatching Verfahren zum Zeitpunkt des Programmstarts durch Templateklassen statt.

7.3.1 Klasse Evaluator

Die Klasse Evaluator dient dazu die Ground Truth Methode praktisch anzuwenden. Als Eingabewerte werden zwei PGM Bilder, einmal das Ground Truth Bild und einmal das zu überprüfende Bild, in Form eines Image-Objektes übergeben. Diese werden dann pixelweise miteinander auf Gleichheit verglichen mit einer gewissen Toleranz. Als Ausgabe bekommt man die Anzahl der schlechten Pixel in % bzw den RMS (siehe Formel 24 und 25).

7.3.2 Klasse Timer

Zur Messung der Rechenzeiten der Algorithmen wurde eine Klasse Timer implementiert die die Methoden start() zum Starten des Timers, stop() zum Stoppen des Timers, check() zur Ausgabe der aktuellen Zeit, restart() zum Restarten zur Verfügung stellt.

8. Ergebnisse und Messungen

Zur Evaluierung der Qualität und Rechenzeit des Blockmatching und Graph Cut Verfahren wurden Experimente mit verschiedenen Testszenarien durchgeführt, die nun im Anschluss vorgestellt und diskutiert werden. Als Betriebssystem wurde Debian Linux benutzt und als Rechner stand ein Intel Pentium 4 mit 2,4 GHz und 512 MB DDR-RAM zur Verfügung. Als C++ Compiler wurde Gcc Version 2.4.1 benutzt.

8.1 Vorstellung der Szenarien

Die Stereobildverarbeitungsalgorithmen wurden auf eine ganze Reihe von Szenarien angewendet:

8.1.1 Stereobildpaar der Universität von Tsukuba

Das „Head and Lamp“ – Szenario der University of Tsukuba [25] hat eine maximale Disparität von 108, eine Auflösung von 720×576 und liegt im PGM Format vor (siehe Abbildung 32). Das Bild ist auch in einer Auflösung von 384×288 verfügbar mit einer maximalen Disparität von 16. Für diese Auflösung gibt es auch Ground-Truth-Informationen.

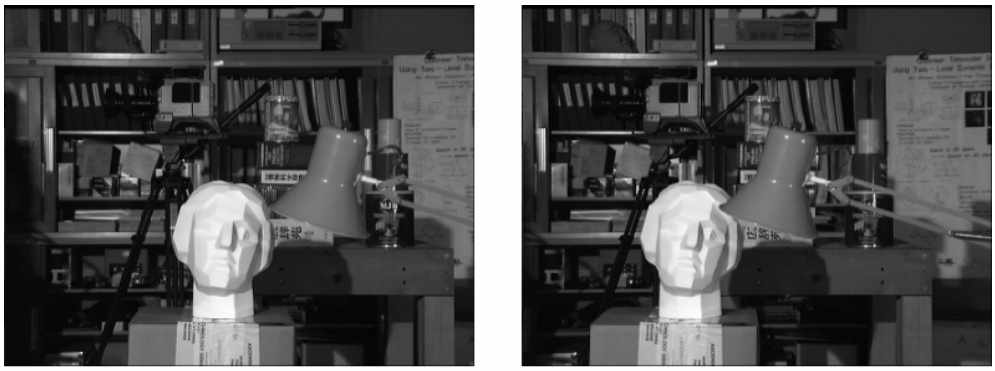

Abbildung 32: Stereobildpaar der Universität von Tsukuba

8.1.2 Stereobildpaar Hindernis

Das Stereobildpaar Hindernis wurde in der Simulationsumgebung für den LAURON Roboter aufgenommen. Dabei wurde ein Weltmodell mit einem Hindernis geladen und eine Textur drübergelegt. Anschließend wurde auf die am LAURON Roboter installierten Kameras umgeschaltet und für jede der zwei Kameras ein Schnappschuss gemacht (siehe Abbildung 33). Die maximale Disparität für dieses Stereobildpaar ist 100 und die Auflösung ist 628×582.

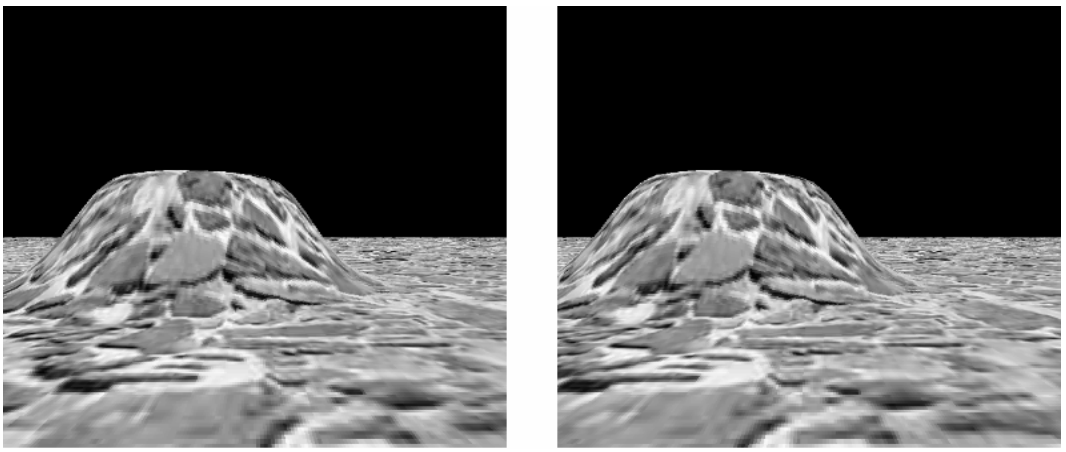

Abbildung 33: Stereobildpaar Hindernis aus der Simulationsumgebung

Es sind auch Ground Truth Daten verfügbar, welche man auch in der Simulationsumgebung erzeugen kann (siehe Abbildung 34).

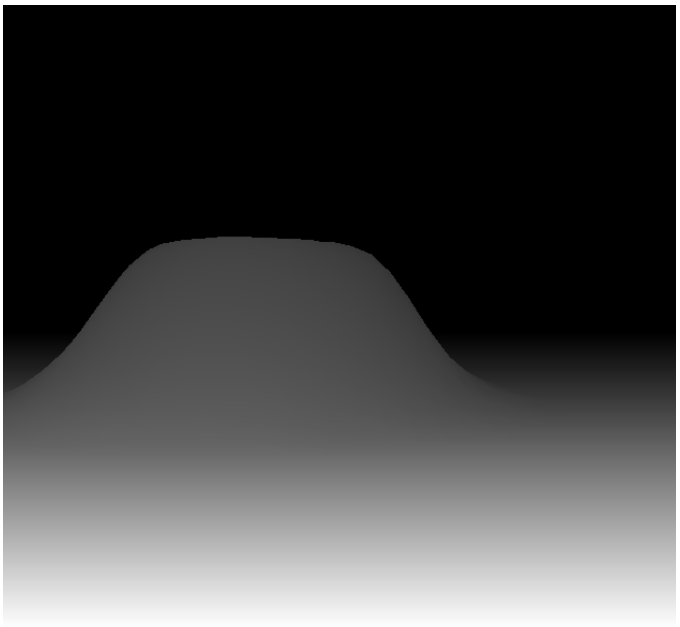

Abbildung 34: Ground Truth Karte des Stereobildpaares Hindernis

8.1.3 Stereobildpaar zerklüftetes Gelände

Das Stereobildpaar zerklüftetes Gelände wurde ebenfalls wie das Stereobildpaar Hindernis in der Simulationsumgebung gemacht. Auch hier ist die maximale Disparität 100 und die Auflösung 628×582.

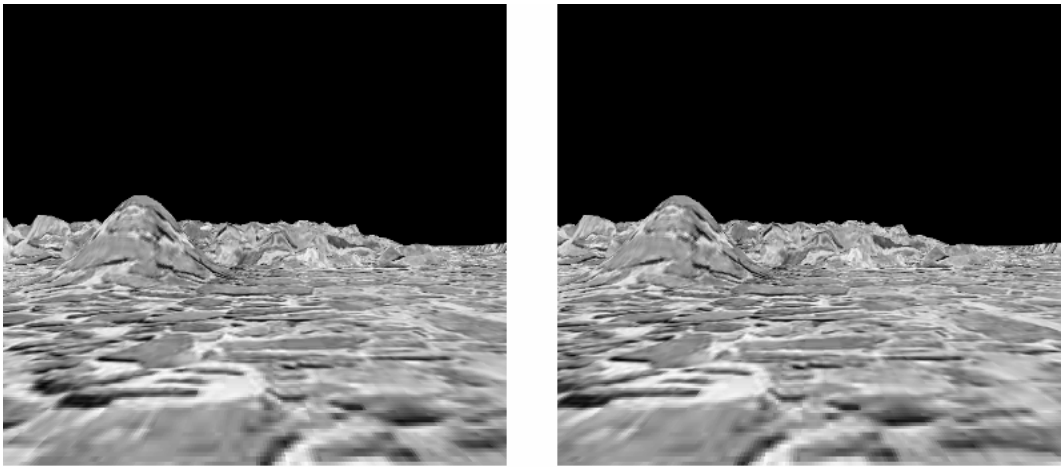

Abbildung 35: Stereobildpaar zerklüftetes Gelände aus der Simulationsumgebung

Es sind auch Ground Truth Daten verfügbar (siehe Abbildung 36).

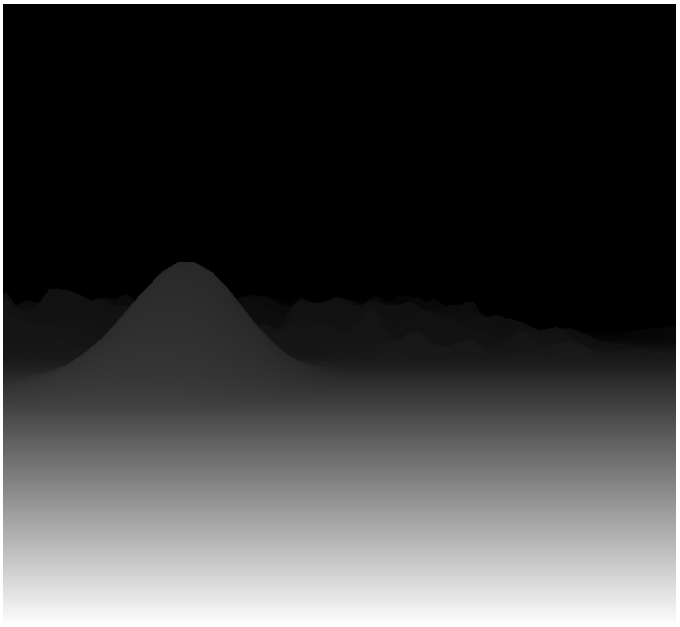

Abbildung 36: Ground Truth Karte des Stereobildpaares Gelände

8.1.4 Stereobildpaar Box

Das Stereobildpaar Box wurde von Richard Bade [1] aufgenommen. Sie liegen im PGM Format in einer Auflösung von 720×576 vor. Er benutzt hierfür 2 Kameras VCAM-600 der

Firma Phytec die in einem Abstand von 16 cm voneinander aufgebaut wurden in einem Winkel von 0°. Die Brennweite beträgt 8 mm und die Pixelgrösse horizontal ist 6,78µm und vertikal 6,32µm. Der Abstand zwischen dem Kamerasystem und der Kiste beträgt 170 cm.

Abbildung 37: Stereobildpaar Box

8.1.5 Stereobildpaar Objekte

Das Stereobildpaar Objekte wurde mit dem hier in der Arbeit beschriebenen Versuchsaufbau aufgenommen in einer Auflösung von 640×480. Der Abstand zwischen den Kameras beträgt 8 cm.

Abbildung 38: Stereobildpaar Objekte

8.1.6 Stereobildpaar Tonne

Das Stereobildpaar Tonne wurde mit dem hier in der Arbeit beschriebenen Versuchsaufbau aufgenommen in einer Auflösung von 640×480. Die Tonne ist dabei 173 cm von dem Stereokamerasystem entfernt aufgebaut. Die Kameras haben einen Abstand von 16 cm.

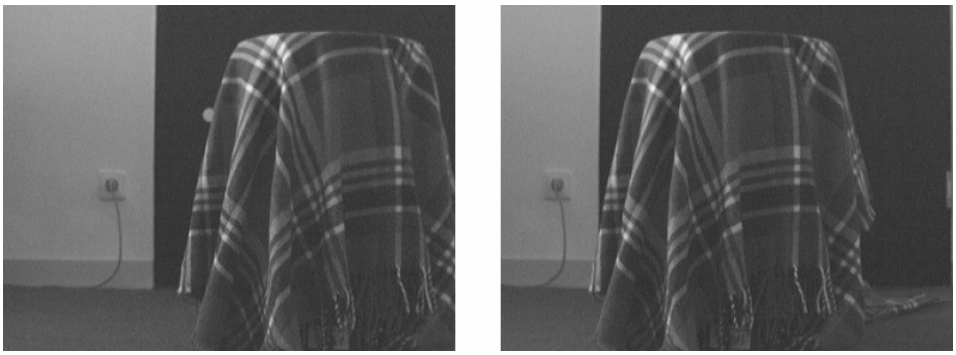

Abbildung 39: Stereobildpaar Tonne

8.1.7 Stereobildpaar Treppe

Das Stereobildpaar Treppe wurde mit dem hier in der Arbeit beschriebenen Versuchsaufbau aufgenommen in einer Auflösung von 640×480. Dabei ist die erste Stufe der Treppe 368 cm vom Stereokamerasystem entfernt. Außerdem hat jede Stufe eine Breite von 30cm. Die Kameras haben einen Abstand von 16 cm.

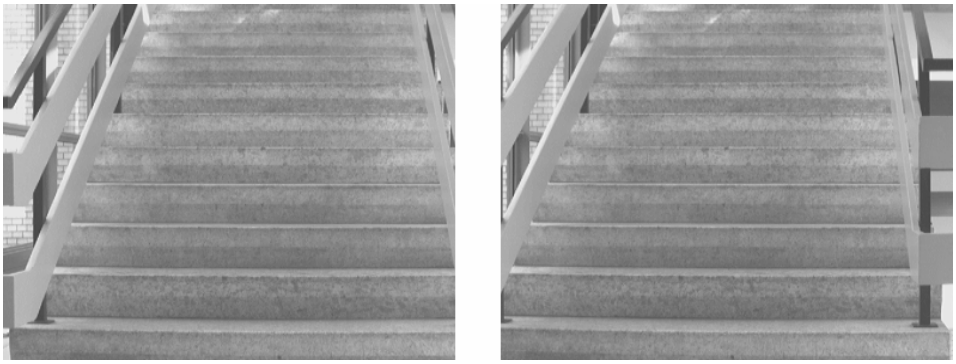

Abbildung 40: Stereobildpaar Treppe

8.2 Anwendung des Blockmatching Verfahrens

In diesem Kapitel wird das Blockmatching Verfahren praktisch auf die verschiedenen Szenarien angewendet. Dazu muss man aber erstmal festlegen wie man die Parameter einstellt.

8.2.1 Festlegen der Parameter

Das Blockmatching Verfahren besitzt folgende Parameter:

- Blockgröße B
- Suchbereich S

Laut Literatur [16] sollen Blockgrößen um 8×8 die besten Ergebnisse liefern. Um dies zu überprüfen wurden verschiedene Größen auf das Tsukuba Bildpaar angewendet. S wurde auf 160 festgelegt. Das Ergebnis ist in Abbildung 40 zu sehen:

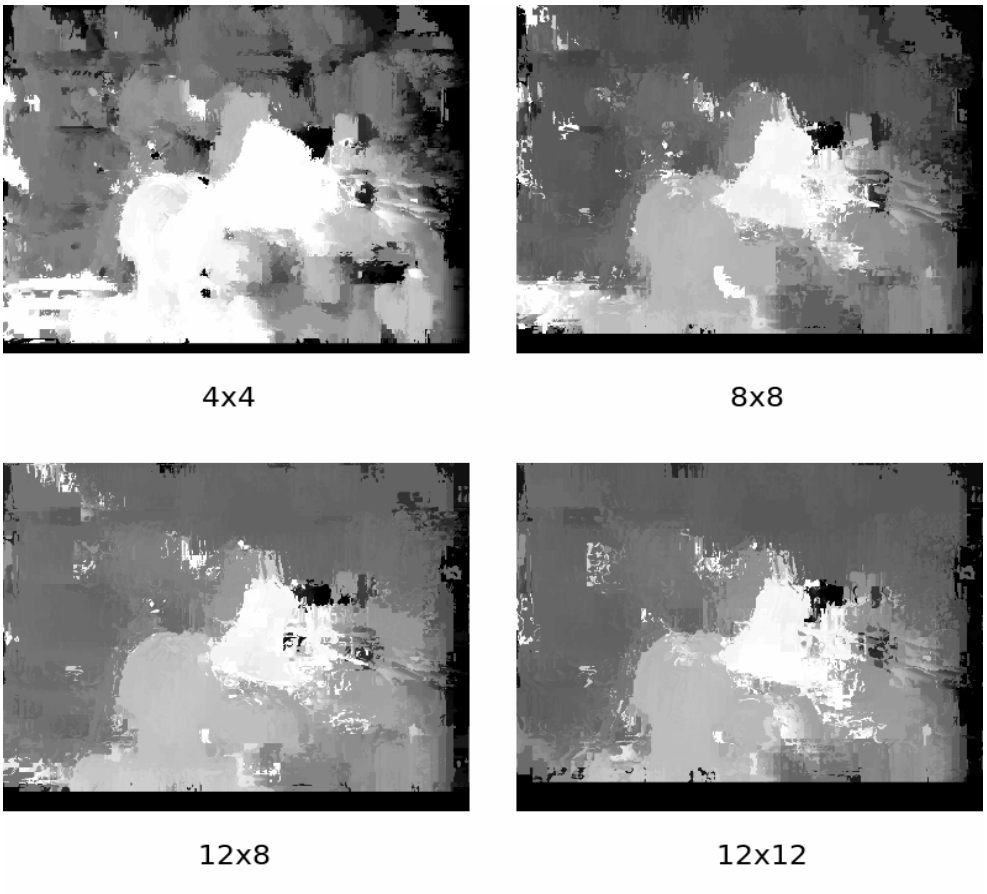

Abbildung 41: Disparitätsmatrizen durch verschiedene Blockgrößen erzeugt

In Abbildung 40 ist deutlich zu erkennen, dass die Qualität des Anytime-Blockmatching Verfahrens mittels eines 4×4 Blockes am geringsten ist. Es sind viele Fehlzuordnungen bei der Korrespondenz in Form von schwarzen oder weißen Flecken zu sehen. Bei den anderen 3 Arten von Blockgrößen sind nur geringe Unterschiede zu erkennen. Im 8×8 Bild sieht man dennoch etwas mehr Fehler und im 12×12 sind die Strukturen etwas ungenauer. Deshalb wird das Blockmatching Verfahren mit einem 12×8 Block für am besten empfunden und alle zukünftigen Experimente werden mit dieser Blockgröße gemacht. Es wurde damit das gleiche Ergebnis gefunden wie bei Richard Bade [1].

8.2.2 Anwendung auf die Szenarien

Für einzelnen Szenarien werden nun die erhaltenen Disparitätsmatrizen sowie die Rechenzeit vorgestellt und analysiert.

Tsukuba Stereobildpaar

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,55
Nach der ersten Blockmatching Stufe	0,56
Nach der zweiten Blockmatching Stufe	0,59
Nach der dritten Blockmatching Stufe	0,72
Nach der vierten Blockmatching Stufe	1,37
Nach der Pixelselektion	2,16

Tabelle 4: Laufzeiten des BM Verfahrens für Tsukuba Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

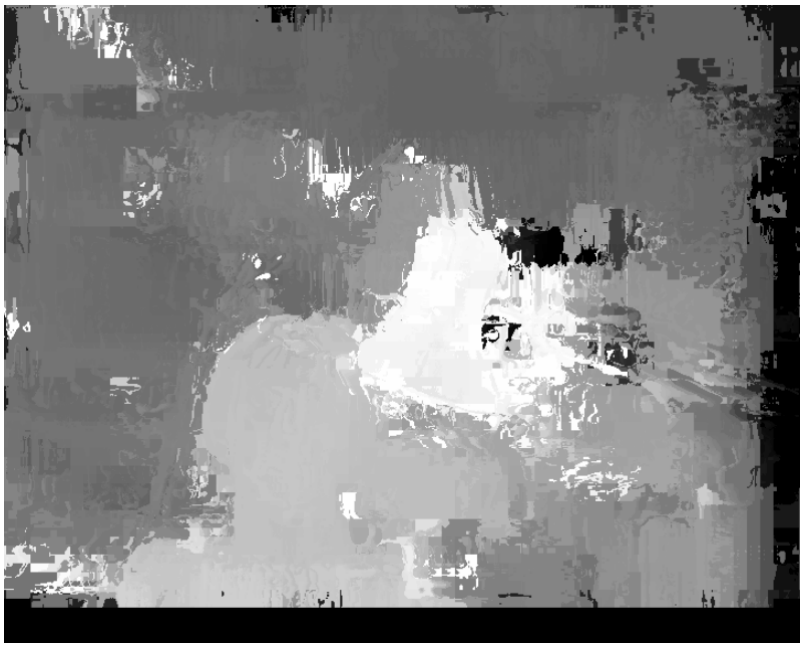

Abbildung 42: Disparitätsmatrix des Tsukuba Bildpaares per BM Verfahren

Die einzelnen Objekte Lampe, Kopfstatue, Kamera, Hintergrund wurden erkannt, allerdings sind die Konturen ungenau.

Stereobildpaar Hindernis

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,47
Nach der ersten Blockmatching Stufe	0,47
Nach der zweiten Blockmatching Stufe	0,49
Nach der dritten Blockmatching Stufe	0,52
Nach der vierten Blockmatching Stufe	0,71
Nach der Pixelselektion	1,41

Tabelle 5: Laufzeiten des BM Verfahrens für Hindernis Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

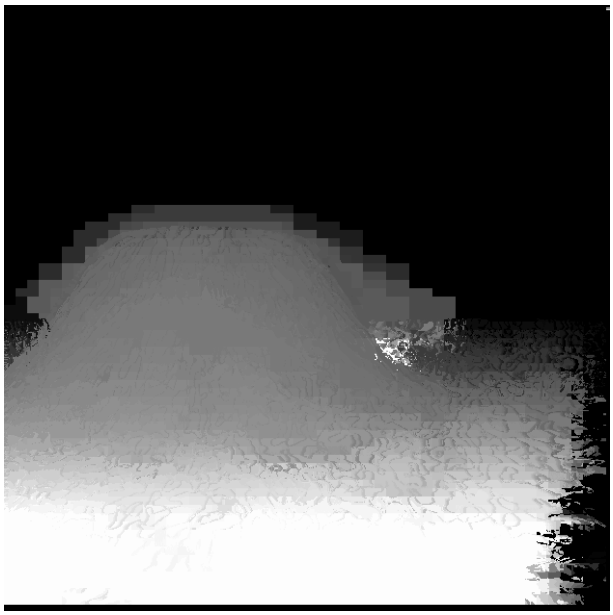

Abbildung 43: Disparitätsmatrix des Hügel Bildpaars mittels BM Verfahren

Laut optischer Bewertung wurden der Boden sowie der Hügel sehr gut erkannt, allerdings gibt es Probleme bei der Erkennung der Kanten beim Hügel. Mit der Ground Truth Methode kommt man bei einer Fehlertoleranz von 4% auf 50,4% falsche Pixel, bei einer Fehlertoleranz von 10% auf 45,62%.

Stereobildpaar zerklüftetes Gelände

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,47
Nach der ersten Blockmatching Stufe	0,47
Nach der zweiten Blockmatching Stufe	0,49
Nach der dritten Blockmatching Stufe	0,52
Nach der vierten Blockmatching Stufe	0,69
Nach der Pixelselektion	1,40

Tabelle 6: Laufzeiten des BM Verfahrens für Gelände Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

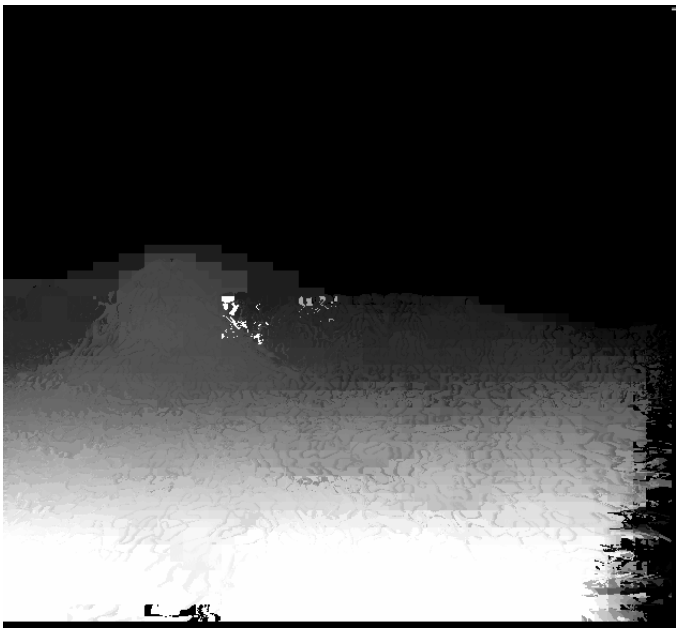

Abbildung 44: Disparitätsmatrix vom Bildpaar Gelände mittels BM Verfahren

Wie beim Stereobildpaar Hindernis sind auch hier die Objektkanten sehr ungenau dargestellt. Die Ground Truth Methode ergibt bei einer Fehlertoleranz von 4% genau 49,93% schlechte Pixel, bei einer Fehlertoleranz von 10% immerhin noch 35,34% fehlerhafte Pixel.

Stereobildpaar Box

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,55
Nach der ersten Blockmatching Stufe	0,56
Nach der zweiten Blockmatching Stufe	0,58
Nach der dritten Blockmatching Stufe	0,67
Nach der vierten Blockmatching Stufe	1,19
Nach der Pixelselektion	1,97

Tabelle 7: Laufzeiten des BM Verfahrens für Box Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

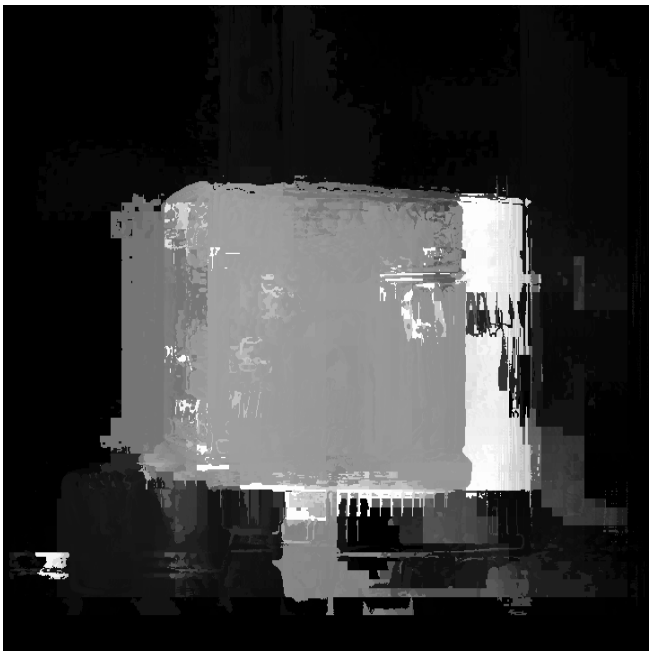

Abbildung 45: Disparitätsmatrix des Bildpaares Box mittels BM Verfahren

Die Kiste sowie Hintergrund wurden erkannt. Probleme gibt es hier bei den Überdeckungen sowie auch beim Boden, der gar nicht erkannt wurde. Wenn man das Bild in 12×8 Blöcke aufteilt findet der Blockmatching Algorithmus für Block 25/50 also einen Block links unten an der Kiste einen Disparitätswert von 107. Umgerechnet mit Formel (7) erhält man damit eine Entfernung von 1,76 m. Der real gemessene Abstand betrug 1,70 m.

Stereobildpaar Objekte

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,41
Nach der ersten Blockmatching Stufe	0,41
Nach der zweiten Blockmatching Stufe	0,43
Nach der dritten Blockmatching Stufe	0,50
Nach der vierten Blockmatching Stufe	1,01
Nach der Pixelselektion	1,60

Tabelle 8: Laufzeiten des BM Verfahrens für Objekte Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

Abbildung 46: Disparitätsmatrix des Bildpaares Objekte mittles BM Verfahren

Die beiden Kisten im Vordergrund sowie die Tonne mit der Decke drüber wurden relativ gut erkannt. Probleme gibt es allerdings bei der Genauigkeit der Objektkanten sowie beim Erkennen des Hintergrunds. Letzteres liegt vor allem da dran dass die Tür keine Texturen aufzuweisen hat und die Korrespondenzanalyse sich bei texturarmen Regionen sich als schwierig erweist.

Stereobildpaar Tonne

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 160.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,39
Nach der ersten Blockmatching Stufe	0,40
Nach der zweiten Blockmatching Stufe	0,41
Nach der dritten Blockmatching Stufe	0,45
Nach der vierten Blockmatching Stufe	0,64
Nach der Pixelselektion	1,23

Tabelle 9: Laufzeiten des BM Verfahrens für Tonne Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

Abbildung 47: Disparitätsmatrix des Bildpaares Tonne mittels BM Verfahren

Die Tonne mit der Decke im Vordergrund wurde zwar erkannt allerdings gibt es wiederum Probleme, wie schon beim Stereobildpaar Objekte, mit der Erkennung des texturarmen Hintergrunds. Für einen Block in der Mitte der Tonne wurde eine Disparität von 137 errechnet. Umgerechnet mit Formel (7) ergibt dies einen Abstand von 1,67m. In der Realität wurde ein Abstand von 1,73m gemessen.

Stereobildpaar Treppe

Die Blockgröße wurde auf 12×8 eingestellt und der Suchbereich auf 100.

Als Laufzeiten wurde ermittelt:

Haltepunkt	Laufzeit bis zum Haltepunkt in Sekunden
Nach der Vorverarbeitung	0,39
Nach der ersten Blockmatching Stufe	0,40
Nach der zweiten Blockmatching Stufe	0,42
Nach der dritten Blockmatching Stufe	0,47
Nach der vierten Blockmatching Stufe	0,77
Nach der Pixelselektion	1,23

Tabelle 10: Laufzeiten des BM Verfahrens für Treppe Bildpaar

Als Ergebnis für die Disparitätsmatrix kam heraus:

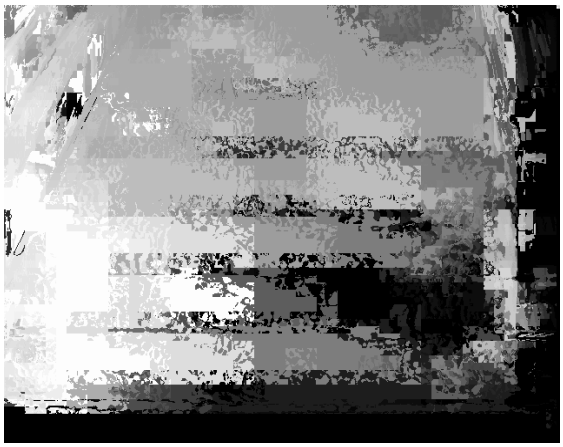

Abbildung 48: Disparitätsmatrix des Bildpaares Treppe mittels BM Verfahren

Hier gibt es massive Probleme bei der Korrespondenzanalyse. Auf der linken Seite des Bildes sind die Disparitäten zwar noch einigermaßen korrekt, allerdings ist dies auf der rechten Seite nimmer der Fall.

8.2.3 Laufzeiten

Als Referenzlaufzeit wurden die Laufzeiten des Tsukuba Stereobildpaares benommen. Die zeitliche Verteilung sieht folgendermaßen aus:

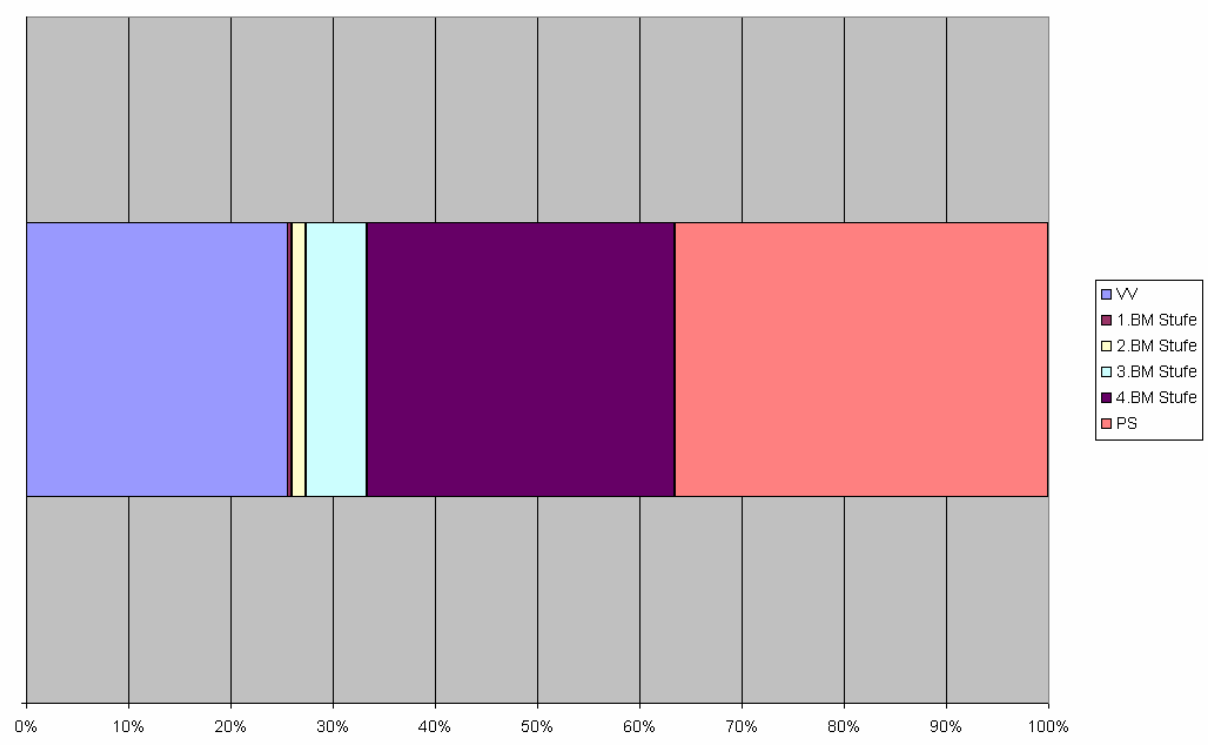

Abbildung 49: Zeitliche Verteilung des Anytime Blockmatching Algorithmus

Nach der Vorverarbeitung (VV) erfolgt die Berechnung der gröbsten Auflösungsstufe. Nach dieser liegt ein erstes Ergebnis vor, sprich nach 0,56 Sekunden. In den nächsten Stufen wird das Ergebnis dann verbessert bis hin zur Nachbereitung in Form von der Pixelselektion. Wird die Pixelselektion weggelassen ist die Auflösung der Disparitätsmatrix sehr grob und ist für den praktischen Einsatz nicht zu empfehlen.

In Abbildung 50 ist die Laufzeit des Blockmatching Verfahrens bezüglich des Suchbereichs N dargestellt in verschiedenen Auflösungsstufen. Benutzt dazu wurde das Tsukuba Bildpaar. Man kann erkennen dass sich die Laufzeit in etwa linear mit der Größe des Suchbereichs erhöht.

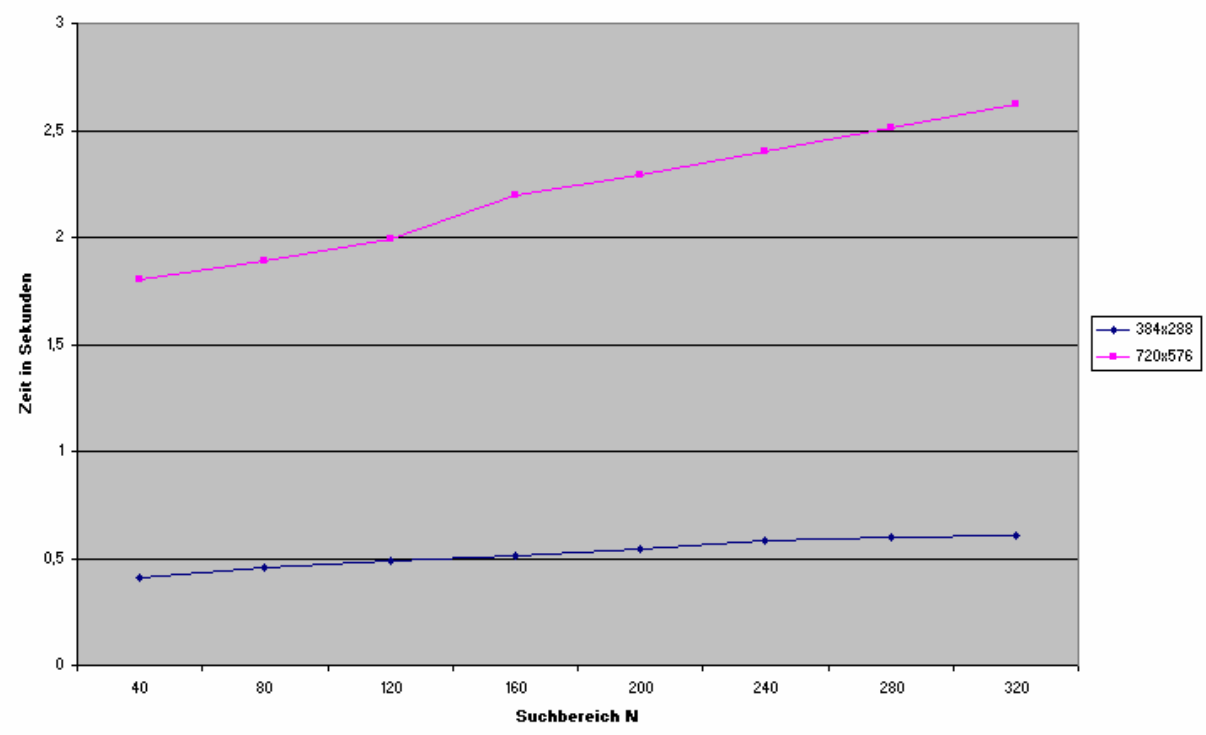

Abbildung 50: Laufzeit des Blockmatching Verfahrens bzgl des Suchbereichs

8.3 Anwendung des Graph Cut Verfahrens

In diesem Kapitel wird das Graph Cut Verfahren praktisch auf die verschiedenen Szenarien angewendet. Dazu muss man aber erstmal festlegen wie man die Parameter einstellt.

8.3.1 Festlegen der Parameter

Das Graph Cut Verfahren besitzt folgende Parameter:

- Begrenzung der Rampenfunktion K_{smooth}
- Begrenzung des Datenterms K_{data}
- Gewichtung des Glattheitsterms λ
- Begrenzung des Suchbereichs S

Die Begrenzung der Rampenfunktion K_{smooth} definiert die Tiefendifferenz, ab der die Existenz einer Objektkante angenommen wird. Der Einfluss von K_{smooth} ist sehr gering. Laut Kolmogorov [17] ist der Standardwert hierfür $K_{smooth} = 5$. Im Folge der Arbeit wird immer dieser Wert angenommen.

Um einen geeigneten Wert für den Datenterm K_{data} zu finden wurden eine Testreihe auf dem Tsukuba Stereobildpaar durchgeführt. Als Parameter wurden $\lambda = 2$ eingegeben und der Suchbereich beträgt 108 Pixel. Das Ergebnis ist in folgender Abbildung zu sehen:

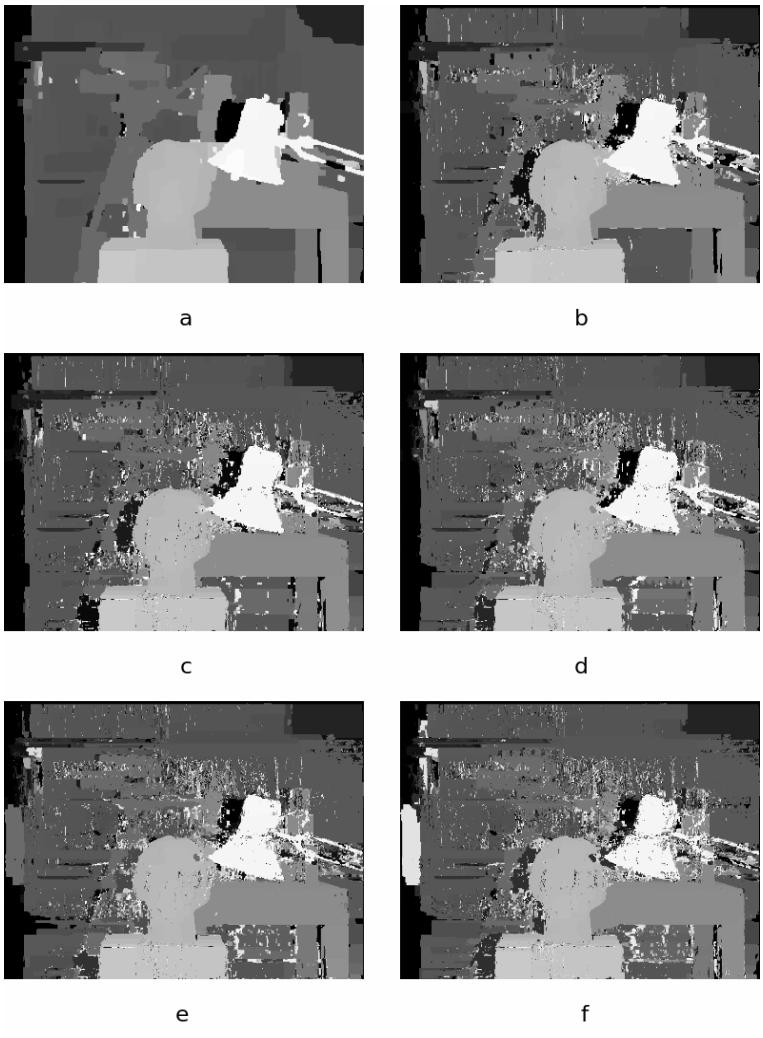

Abbildung 51: Testreihe mit variablem Datenterm und festem Glattheitsterm

Von Bild a bis f wurden folgende Werte für K_{data} gewählt: a: 5, b:15, c:30, d:45, e:60, f:90. Deutlich zu sehen ist dass mit der Zunahme von K_{data} die Genauigkeit des Bildes zunimmt, also mehr Details dargestellt werden. Allerdings nimmt die Menge der fehlerhaften Pixel zu. Ein guter Kompromiss findet man bei $K_{data} = 45$.

In der nächsten Testreihe (siehe Abbildung 52) wird der Parameter λ als Variable angenommen und K_{data} wird auf 45 festgesetzt. Die restlichen Parameter bleiben gleich.

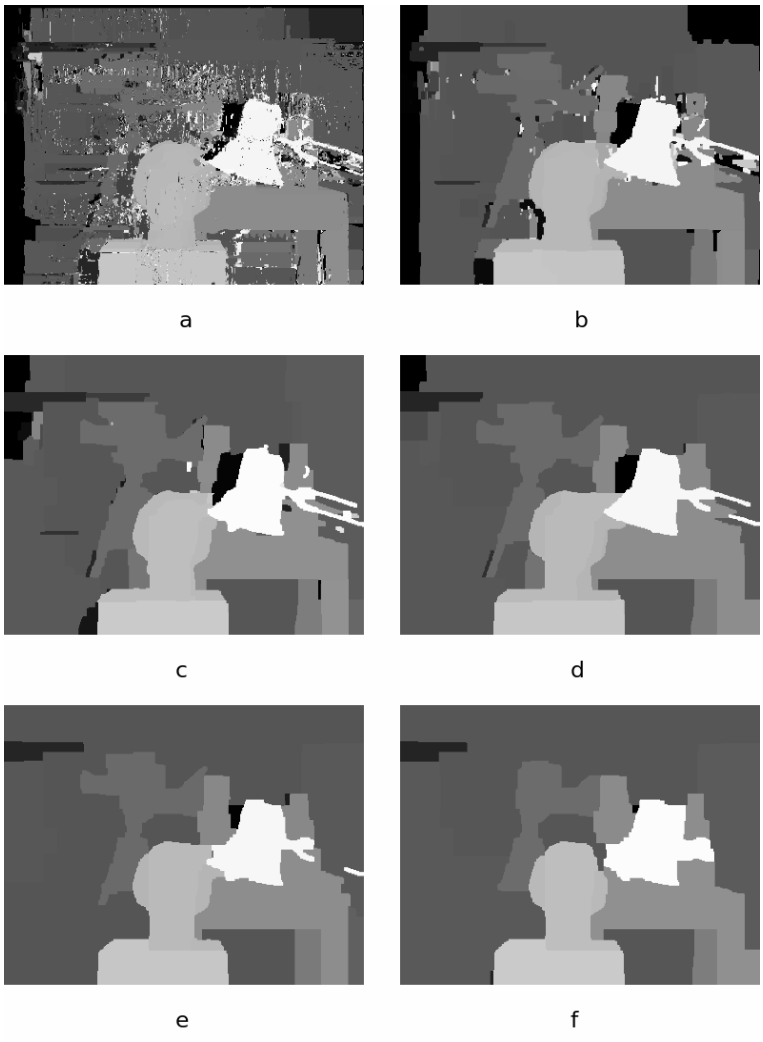

Abbildung 52: Testreihe mit festen Datenterm und Variablem Glattheitsterm

Von Bild a bis f wurden folgende Werte für λ gewählt: a: 2, b:15, c:30, d:45, e:60, f:90. Zu sehen ist dass bei Zunahme von λ die Disparitätskarte immer glatter wird. Zum Schluss regelrecht so glatt dass ganze Teile von Objekten wie zum Beispiel der Lampenhalter oder ein Teil der Kamera verschwinden. Das Übersehen von Hindernissen darf bei einem Robotersystem natürlich nicht passieren weshalb wir zum Schluss kommen dass ein Wert von λ zwischen 2 und 15 am geeignetsten ist. Genauere Untersuchungen haben ergeben dass $\lambda =$ 8 dafür in Frage kommt. Mit Hilfe der Ground Truth Methode bei einer Toleranz von 4% wurde nämlich eine Prozentzahl von fehlerhaften Pixeln von 5,84 % gefunden bei $K_{data} = 45$, $\lambda = 8$. Bei steigendem λ werden die Resultate der Ground Truth Methode wegen dem Schließen der „Lücken“ zwar noch etwas besser, allerdings verschwinden dabei Teile von Objekten weswegen wir bei $\lambda = 8$ bleiben. (siehe Abbildung 53 Bild c).

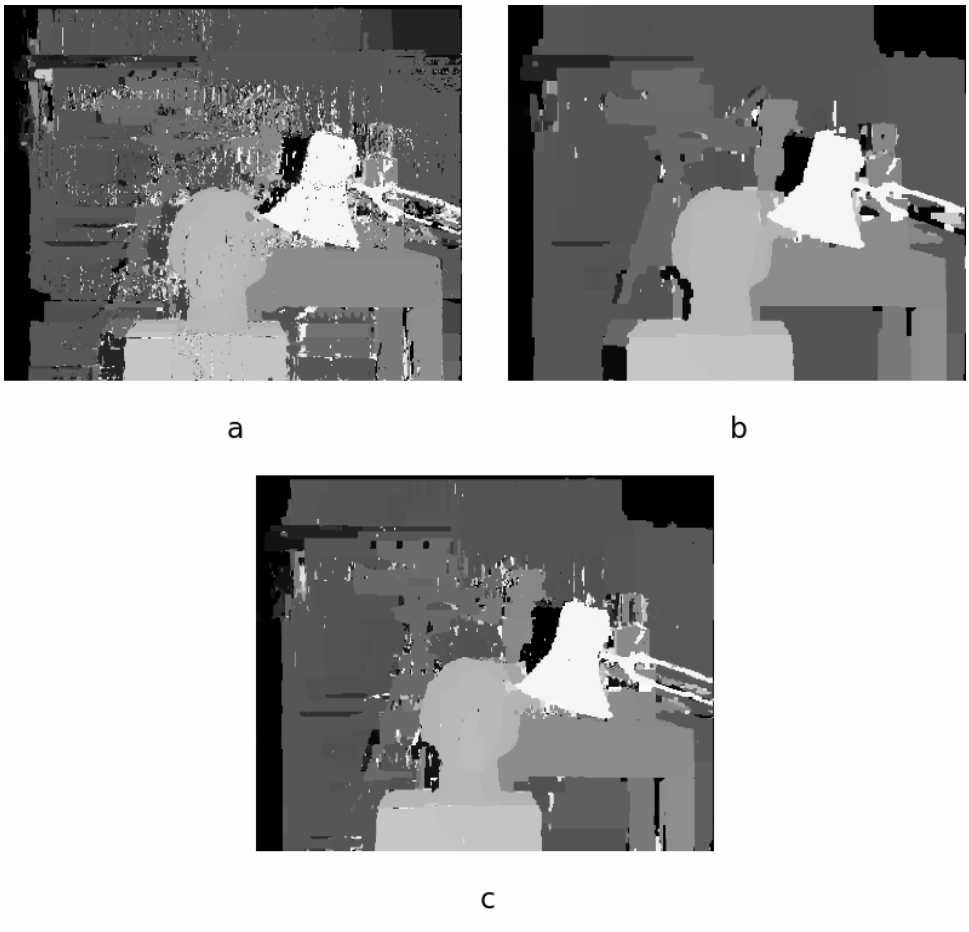

Abbildung 53: Testreihe mit Datenterm = 45 und Glattheitsterm = 2 / 8 /15

8.3.2 Anwendung auf die Szenarien

Für einzelnen Szenarien werden nun die Ergebnisse in Form von Disparitätsmatrizen sowie die Rechenzeit vorgestellt und analysiert.

Tsukuba Stereobildpaar

Als Parameter wurden $K_{smooth} = 5$, $K_{data} = 45$, $\lambda = 8$, $S = 108$ gewählt. Die Rechenzeit betrug 565,32 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

Abbildung 54: Disparitätsmatrix des Tsukuba Stereobildpaars per GC Verfahren

Die Objekte Lampe, Kopf, Kamera, Tisch und Hintergrund wurden erkannt. Die Konturen sind relativ exakt, allerdings gibt es doch einige Lücken bzw Pixelfehler.

Stereobildpaar Hindernis

Als Parameter wurden $K_{smooth} = 5$, $K_{data} = 45$, $\lambda = 8$, $S = 100$ gewählt. Die Rechenzeit betrug 222,77 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

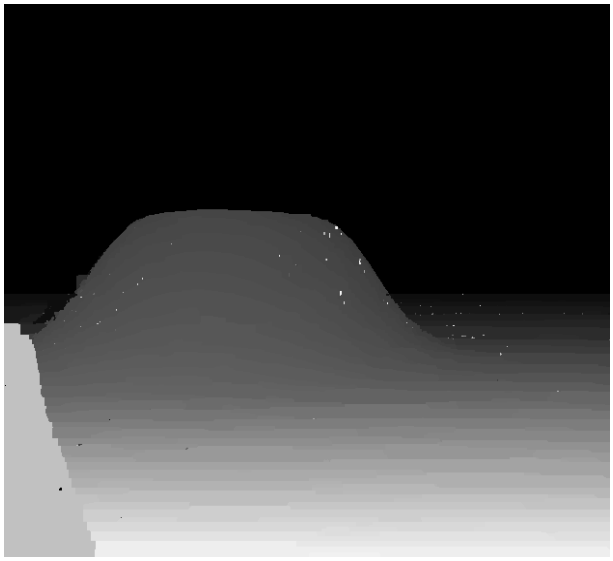

Abbildung 55: Disparitätsmatrix des Hindernis Bildpaares per GC Verfahren

Bis auf einige Ausreißer wurde das Stereobildpaar Hindernis fast perfekt erkannt. Auch die Ground Truth Methode ergibt gute Ergebniss. Bei einer Fehlertoleranz von 4% hat man 5,98% schlechte Pixel, bei einer Fehlertoleranz von 10% hat man 2,58% fehlerhafte Pixel.

Stereobildpaar zerklüftetes Gelände

Als Parameter wurden $K_{smooth} = 5,\ K_{data} = 45,\ \lambda = 8, S = 100$ gewählt. Die Rechenzeit betrug 236,28 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

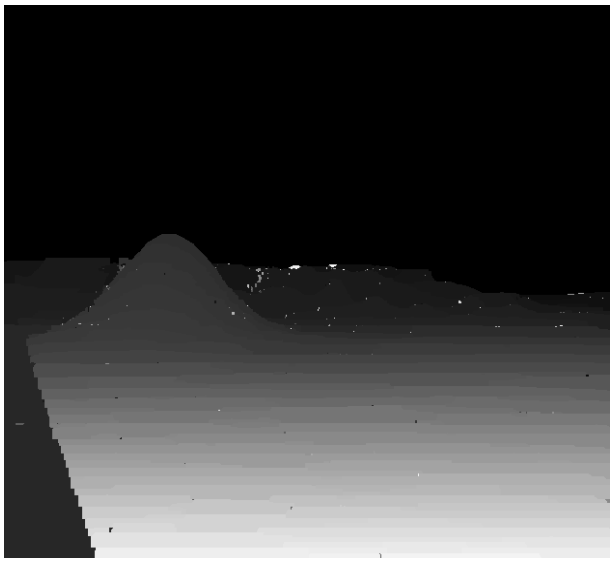

Abbildung 56: Disparitätsmatrix des Stereobildpaars Gelände mittels GC Verfahren

Auch hier wurden, wie beim Stereobildpaar Hindernis, bis auf einige Ausreißer die Disparitäten sehr gut erkannt. Die Ground Truth Methode ergibt 8,34% fehlerhafte Pixel bei 4% Toleranz. Bei 10% Toleranz hat man 3,56% fehlerhafte Pixel.

Stereobildpaar Box

Als Parameter wurden $K_{smooth} = 5,\ K_{data} = 45,\ \lambda = 8, S = 108$ gewählt. Die Rechenzeit betrug 565,32 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

Abbildung 57: Disparitätsmatrix des Stereobildpaars Box mittels GC Verfahren

Die Kiste wurde relativ gut erkannt. Ihre Kanten sind relativ genau abgebildet. Allerdings gibt es einige Probleme auf der linken Seite der Kiste mit dem texturarmen Hintergrund.

Stereobildpaar Objekte

Als Parameter wurden $K_{smooth} = 5,\ K_{data} = 45,\ \lambda = 8, S = 100$ gewählt. Die Rechenzeit betrug 524,79 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

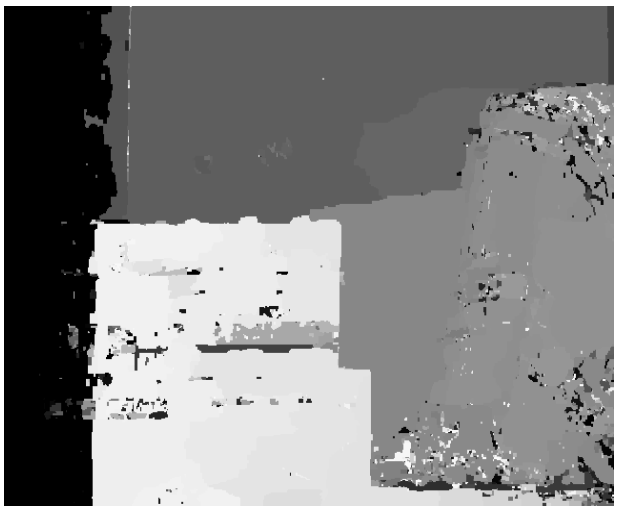

Abbildung 58: Disparitätsmatrix des Stereobildpaars Objekte mittels GC Verfahren

Die beiden Kisten im Vordergrund sowie die Tonne wurden gut erkannt. Die Kanten sind relativ genau. Probleme gibt es allerdings bei der Erkennung des Hintergrundes.

Stereobildpaar Tonne

Als Parameter wurden $K_{smooth} = 5,\ K_{data} = 45,\ \lambda = 8, S = 140$ gewählt. Die Rechenzeit betrug 524,79 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

Abbildung 59: Disparitätsmatrix des Stereobildpaars Tonne mittels GC Verfahren

Die Tonne wurde erkannt, die Konturen sind relativ genau allerdings gibt es große Probleme beim Erkennen des Hintergrundes.

Stereobildpaar Treppe

Als Parameter wurden $K_{smooth} = 5$, $K_{data} = 45$, $\lambda = 8$, $S = 70$ gewählt. Die Rechenzeit betrug 217,26 Sekunden. Als Ergebnis kam folgende Disparitätsmatrix heraus:

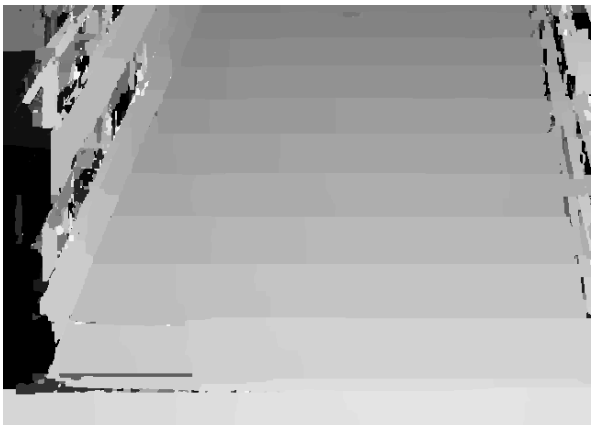

Abbildung 60: Disparitätsmatrix des Stereobildpaars Treppe mittels GC Verfahren

Die Treppe, sowie die einzelnen Stufen wurde sehr gut erkannt. Die Kanten sind relativ genau.

8.3.3 Laufzeiten

Um die Laufzeiten zu ermitteln wurde das Tsukuba Stereobildpaar verwendet. Als Parameter wurde $K_{smooth} = 5$, $K_{data} = 45$, $\lambda = 8$ und S als Variable gewählt. Wie schon in der Theorie angenommen, bestätigt sich auch in der Praxis dass die Laufzeit sich linear zum Suchbereich S, bzw der Anzahl der Disparitätslabel erhöht. In Abbildung 61 ist die Laufzeit in Sekunden für das Tuskuba Bildpaar in zwei verschiedenen Auflösungen dargestellt. Da beim Stereo Graphcut Verfahren die Anzahl der Knoten identisch ist mit der Anzahl der Pixel des Eingabebildes erhöht sich die Laufzeit mit zunehmender Auflösung entsprechend.

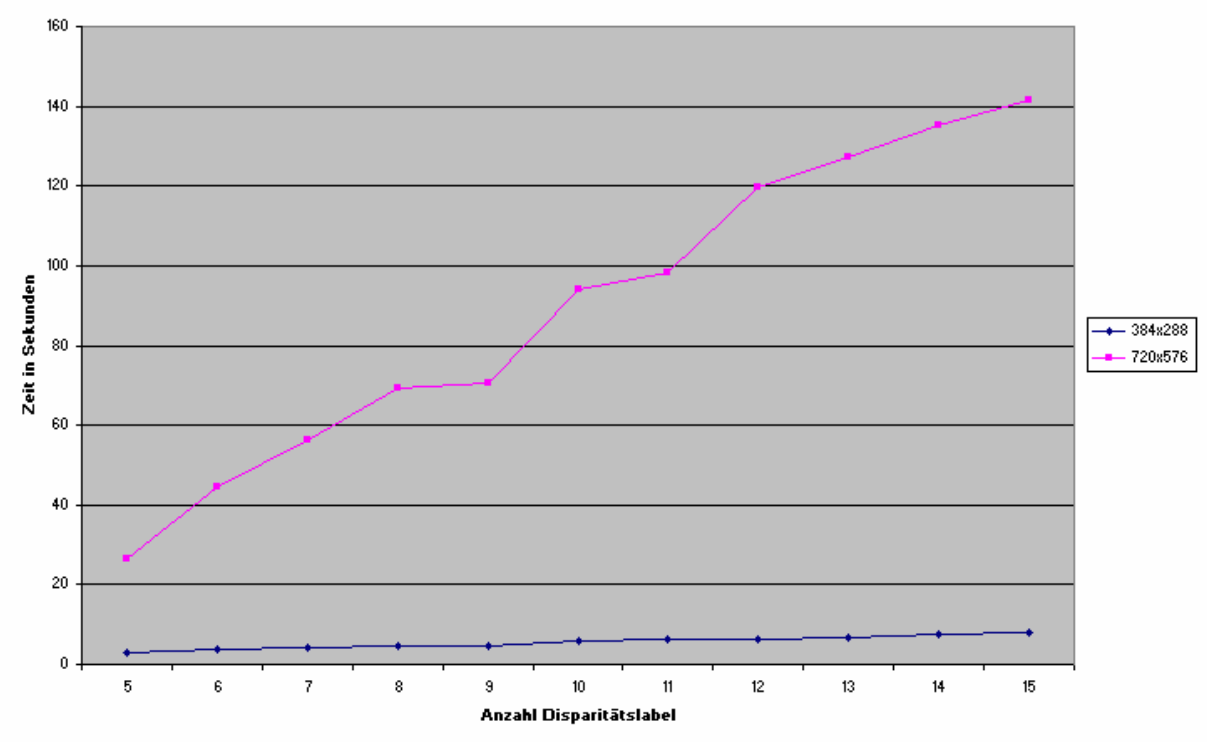

Abbildung 61: Laufzeit des Graphcut Verfahrens bzgl des Suchbereichs

8.4 Erfassungsbereich unseres Versuchaufbaus

Bei unserem Versuchaufbau haben wir, wie bereits in Kapitel 8.1 erwähnt ein Stereokamerasystem mit folgenden Parametern gewählt:

- L = 16 cm
- d_u = 5,6 µm
- f = 8 mm

Mit Hilfe der Formel (7): $\rho = \frac{Lf}{\delta \cdot d_u}$ lässt sich nun die Tiefe aus den verschiedenen Disparitätswerten berechnen. Das Ergebnis mit Disparitätslimit $d_{max} = 324$ (das heißt ein Bild von 640×480 wäre um etwa 50% überlappend) ist in der folgenden Abbildung zu sehen:

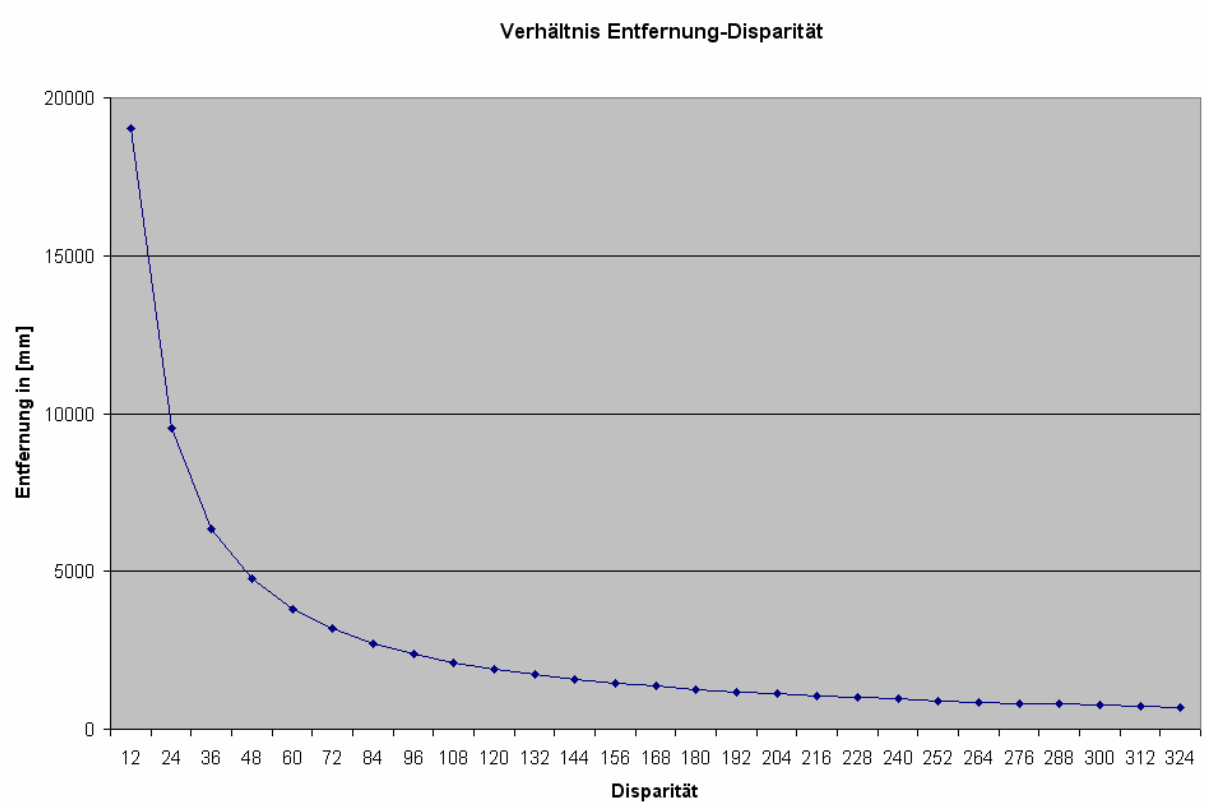

Abbildung 62: Verhältnis Entfernung zur Disparität

Insbesondere für das Blockmatching Verfahren, wo wir die Blockgrösse für unsere Versuche ja bekanntlich auf 12 Pixel in der Horizontale festgelegt haben, gelten folgende Eigenschaften: In der Theorie kann man bei einer Disparität von 12 Pixeln maximal eine Tiefe von ca 19 m erfassen . In der Praxis ist aber diese Entfernung schlecht zu erkennen, da diese minimale Disparität von Rauschen kaum zu unterscheiden ist. Des weiteren ist die Auflösung bei mehr als 3-4 m relativ grob, was auch zu einer Verschlechterung der Resultate führt. Das Disparitätslimit $d_{\max} = 324$ beschränkt die minimale Tiefe auf 0,7 m. Ein guter Arbeitsbereich liegt also zwischen 0,7 m und 3,5 m.

Durch eine Vergrößerung des Abstandes L zwischen den beiden Kameras, kann man den Arbeitsbereich in eine weitere Entfernung verschieben.

Bei dem Graph Cut Verfahren, welches pixelbasiert ist, kriegt man auch bessere Resultate außerhalb dieses Arbeitsbereiches, sprich wenn die Entfernung vom Objekt zu der Kamera größer als 3,5 m ist.

8.5 Vergleiche und Fazit

In diesem Kapitel werden Vergleiche von den Ergebnissen von dem Blockmatching Verfahren und dem Graph Cut Verfahren gezogen bevor abgewogen wird welches der beiden Verfahren für das LAURON Roboter System am besten geeignet ist.

8.5.1 Vergleiche

Die beiden Verfahren werden nun nach verschiedenen Kriterien verglichen:

Dichte der Tiefeninformationen

Bei der Dichte der Tiefeninformationen nehmen sich die beiden Verfahren nicht viel. Probleme gibt es bei beiden Verfahren bei der Zuordnung von texturarmen Regionen (siehe zum Beispiel den Hintergrund bei Stereobildpaar Tonne). Außerdem ist zu vermerken dass bei den „perfekten" Bildern aus der Simulationsumgebung weniger Korrespondenz-zuordnungsfehler zu vermerken sind als bei den Bildern aus der realen Welt. Dies ist unter anderem auch auf äußere Einflüsse zurückzuführen wie Sonneneinstrahlung und Reflektion von Licht.

Genauigkeit der Objektkanten

Bei der Genauigkeit der Objektkanten sind deutliche Vorteile beim Graph Cut Verfahren zu erkennen, da es sich hier um ein pixelbasiertes Verfahren handelt. Beim Blockmatching wo es sich um ein fensterbasiertes Verfahren handelt sind die Kanten oft etwas „verflossen". Deutlich zu sehen ist dies beim Stereobildpaar Hindernis.

Laufzeit

Bei der Laufzeit ist das Anytime Blockmatching Verfahren deutlich schneller als das Graph Cut Verfahren. So hat zum Beispiel das Graph Cut Verfahren beim Tsukuba Bildpaar bei einem Suchbereich von 160 Pixeln von eine Laufzeit von 565,32 Sekunden wobei das Blockmatching Verfahren nur 2,14 Sekunden braucht. Da die Echtzeitanforderung für den LAURON Roboter bei ungefähr 1 Bild pro alle 5 sec liegt erfüllt das Blockmatching Verfahren dieses Kriterium. Bei sehr kleinen maximalen Disparitäten und geringen Auflösungen bzw durch Implementierung des Pyramideverfahren, kann man das Graph Cut Verfahren zwar beschleunigen, aber Echtzeitanforderungen werden damit nicht erreicht.

Größe des Arbeitsbereichs

Die Größe des Arbeitsbereichs beim Blockmatching Verfahren beträgt bei unserem Versuchsaufbau 3,5 – 0,7 = 2,8 m. Beim Graph Cut Verfahren hingegen kann man auch Objekte aus weiterer Entfernung (19 m und mehr) noch sehr gut erkennen, wie man am Stereobildpaar Treppe sieht wo die höchste Stufe welche 6,4 m entfernt ist auch noch deutlich zu erkennen ist.

Ground Truth

Mit Hilfe der Ground Truth Methode stellt man fest dass die Ergebnisse des Graph Cut Verfahrens um einiges besser sind als die der Blockmatching Methode (siehe Ergebnisse beim Stereobildpaar Hindernis und Stereobildpaar Gelände). So werden beim Stereobildpaar Hindernis bei einer Fehlertoleranz von 4% beim Graph Cut Verfahren 5,98% fehlerhafte Pixel gefunden, beim Blockmatching Verfahren über 50%. Die Ergebnisse beim Blockmatching Verfahren werden erst besser wenn man die Toleranz erhöht.

Tabellarisch zusammengefasst haben die beiden Verfahren im Vergleich also folgende Vor (+) bzw Nachteile (-) :

Kriterium	Blockmatching	Graph Cut
Dichte der Tiefeninformationen	+	+
Genauigkeit der Objektkanten	-	+
Laufzeit	+	-
Größe des Arbeitsbereichs	-	+
Ground Truth	-	+

Tabelle 11: Vorteile / Nachteile des BM bzw GC Verfahren im Vergleich

8.5.2 Fazit

Die Ergebnisse des Graph Cut Verfahren sind zwar von sehr guter Qualität, allerdings verhindert ein essenzielles Problem die Auswahl dieses Verfahrens für das LAURON Roboter System: wie in den Kapiteln 9.3.3 und 9.5.1 gezeigt wurde ist die Laufzeit bzw Rechenzeit des Verfahrens einfach zu hoch um das Kriterium der Echtzeitfähigkeit zu erfüllen (das könnte sich allerdings in Zukunft bei zunehmender Rechenleistung ändern).

Wir kommen also zum Fazit dass das Blockmatching Verfahren für das LAURON Roboter System von den vorgestellten Verfahren am besten geeignet ist. Die Dichte der Tiefeninformationen ist relativ hoch. Die Objektkanten werden zwar nicht genau erkannt, allerdings reichen die Informationen aus um eine grobes Weltmodell zu erschaffen. Die Laufzeit ist absolut echtzeitfähig, selbst mit der höchsten Auflösungsstufe. Da der LAURON Roboter sich nur sehr langsam bewegt, reichen 2-3 Sekunden Rechenzeit absolut aus. Den Fakt des eingeschränkten Arbeitsbereiches muss man allerdings beim Erstellen des Weltmodells mit berücksichtigen, wie auch die Tatsache dass die Ergebnisse nur mit einer relativ hohen Toleranz einigermaßen korrekt sind.

9. Zusammenfassung und Ausblick

Um für sechsbeinige Roboter bessere Laufbewegungen zu erreichen, hat André Herms einen Laufplanungsalgorithmus implementiert, der die Fußauftrittspunkte plant.. Dazu verwendet er Geländeinformationen die in Form von einer Höhenkarte vorliegen. Um diese Höhenkarte der realen Welt, also ein so genanntes Weltmodell zu berechnen, werden Sensorinformationen benötigt welche mit Hilfe eines Stereokamerakopfes, der am Roboter befestigt ist gewonnen werden. Durch Anwendung von Stereobildverarbeitungsalgorithmen auf diese Informationen in Form von Stereobildpaaren kann man eine Tiefenkarte erstellen. In dieser Arbeit werden geeignete Stereobildverarbeitungsalgorithmen vorgestellt und miteinander nach Qualität und Rechenzeit verglichen. Aus einer Auswahl geeigneter Bildverarbeitungsalgorithmen, die da wären Merkmalbasierte Verfahren, Gradientenbasierte Verfahren, Blockmatching Verfahren, Dynamisches Programmieren sowie Graph Cut Verfahren wurde, eine nach den Kriterien von Rechenzeit und Dichte der gewonnen Tiefeninformationen, engere Auswahl getroffen: Das merkmalbasierte Verfahren schied wegen mangelnder Dichte, das Dynamische Programmieren sowie die Gradientenbasierte Verfahren wegen zu hohem Rechenaufwand aus. Übrig blieben das Graph Cut Verfahren sowie das Blockmatching Verfahren, welche auf verschiedene Testszenarien angewendet wurden und auf Echtzeitfähigkeit untersucht wurden. Außerdem wurde die Dichte der Tiefeninformationen, die Genauigkeit der Objektkanten sowie die Größe des Arbeitsbereichs untersucht. Auch eine Messung der Anzahl der fehlerhaften Pixel mit Hilfe der Ground Truth Methode wurde durchgeführt. Dabei kommt man zum Ergebnis dass der Anytime-Blockmatching Stereobildverarbeitungsalgorithmus am geeignetsten für die Erstellung des Weltmodells ist, weil er das Kriterium der Echtzeitfähigkeit erfüllt und sehr dichte Tiefenkarten erstellt. Allerdings sind die Ergebnisse nicht exakt genug um ein perfektes Weltmodell zu liefern. So werden texturarme Regionen zum Beispiel sehr schlecht erkannt. Auch die Kanten der Objekte werden nur „grob“ erkannt und man muss eine gewisse Fehlertoleranz mit einberechnen. Als Lösung könnte man bei der Laufplanung reaktive und statische Verfahren kombinieren. Dies würde man hinkriegen indem man das aus der Stereobildverarbeitung gewonnene Weltmodell für das planende Laufen verwendet. Wenn während der geplanten Fortbewegung allerdings nun ein übersehenes Hindernis auftritt, so kann ein reaktives Verfahren eingesetzt werden.

Literaturverzeichnis

[1] Bade, Richard. *"Modifikation einer geeigneten Stereobildverarbeitungsmethode für die Anwendung in einer Echtzeitumgebung", Diplomarbeit Universität Magdeburg 2003.*

[2] Becker Mark, Kefalea Efthimia, Mael Eric, Von der Malsburg Christoph, Pagel Mike, Triesch Jochen, Vorbrüggen Jan C., Würtz Rolf P., Zadel Stefan: *"GripSee: A Gesture-controlled Robot for Object Perception an Manipulation", Autonomous Robots 1999, pages 203-221.*

[3] Brown, Myron Z.; Burschka, D.; Hager G.D. *"Advances in Computational Stereo". In: IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.25, No.8, August 2003, pages 993-1008.*

[4] Büker, Ulrich. *"Aktive Szenenauswertung und Objekterkennung". Universität Paderborn 2001.*
http://ubdata.uni-paderborn.de/ediss/14/2001/bueker/habil.pdf

[5] Cruse, H., Bartling Ch., Dreifert, M., Schmitz, J., Brunn D.E., Dean J., Kindermann T. *"Walking: A Complex Behavior Controlled by Simple Networks", In: Adaptive Behavior Vol.3, No.4, 1995, pages 385-418.*

[6] Ford, L; Fulkerson, D. *"Flows in Networks". Princeton University Press, 1962.*

[7] Frommberger, Lutz. *"3-D Umweltmodellierung zur Navigationsunterstützung von Laufmaschinen im Gelände", Diplomarbeit Universität Karlsruhe 2002.*

[8] Gassmann, Bernd. *"Erweiterung einer modularen Laufmaschinensteuerung für unstrukturiertes Gelände." Diplomarbeit Universität Karlsruhe September 2000.*

[9] Guddat, Martin. *"Autonome, adaptive Bewegungskoordination von Gehmaschinen in komplexer Umgebung", Düsseldorf: Vdi-Verlag 2003,* ISBN 3-18-399508-1.

[10] Herms, André. *"Entwicklung eines verteilten Laufplaners basierend auf heuristischen Optimierungsverfahren", Diplomarbeit Universität Magdeburg 2004.*

[11] Homepage - RoboMow, der Mähroboter von Friendly Robotics.
http://www.rasenmaeher-roboter.de/

[12] Homepage - FLIR, The World Leader in infrared technology.
http://www.flir.com

[13] Homepage - SGI Open Inventor.
http://oss.sgi.com/projects/inventor

[14] Homepage - Phytec Messtechnik Gmbh.
http://www.phytec.de/phytec/firewire-kameras/firewire-cam-002.html

[15] Homepage – MLR System Gmbh, Materialfluß- und Logistiksysteme.
http://www.mlr.de/pdf/Datenbl%C3%A4tter/DB_AWT.pdf

[16] Klette, R.; Schlüns, K. *"Computer Vision – Räumliche Information aus digitalen Bildern"; Vieweg Technik Verlag; 1996*

[17] Kolmogorov, Vladimir and Ramin, Zabih. *"Multi-Camera scene Reconstruction via Graph Cuts" In: European Conference on Computer Vision, May 2002, pages 82-96.*

[18] Kolmogorov, Vladimir and Ramin, Zabih. *"Computing Visual Correspondence with Occlusions using Graph Cuts" In: International Conference on Computer Vision, July 2001, pages 508-515.*

[19] Kolmogorov Valdimir, Zabih Ramin. *"What energy functions can be minimized via graph cuts " In*: *European Conference on Computer Vision 2002, pages 65-81.*

[20] L. Matthies, Alonzo Kelly, T. Litwin and G. Tharp, *"Obstacle Detection for Unmanned Ground Vehicles: A Progress Report", In: 7th International Symposium on Robotics Research,Oct 1995 , pages 66-71.*

[21] Nimser, Michael. *"Dichte Tiefenschätzung aus mehreren Kameras mittels Graph Cut Algorithmen", Diplomarbeit Universität Kiel 2003.*

[22] Reuter, T. *"HDTV standards conversion". Proc. IEEE-ASSP & EURASIP 5th Workshop on Multidimensional Signal Processing, Nordwijkerhouth, Niederlande, 1987.*

[23] Roy S., Cox I.J., *"A Maximum-Flow Formulation of the N-Camera Correspondence Problem", In: International Conference Computer Vision 1998, pages 492-499.*

[24] Ruffler, Uli. *"Laufplanung basierend auf realitätsnahen Umgebungsdaten für einen sechsbeinigen Laufroboter", Diplomarbeit Hochschule Mannheim 2006.*

[25] Scharstein, Daniel; Szeliski, R. *"A taxonomy and evaluation of dense two-frame stereo correspondence algorithms". In: IJCV 47(1/2/3):7-42, April-June 2002.*

[26] Schmalz, Steffen. *"Entwurf zur Evaluierung von Strategien zur 2D/3D-Objekterkennung in aktiven Sehsystemen".*
http://deposit.ddb.de/cgi-bin/dokserv?idn=961994681&dok_var=d1&dok_ext=pdf&filename=961994681.pdf

[27] Schreer, O. *"Stereoanalyse und Bildsynthese", Springer Verlag, 2005***,** ISBN 3-540-23439-X.

[28] Wetzelsberger, Kai. *"Konzeption und Implementierung von sensorbasierten Bewegungsalgorithmen für einen Laufroboter", Diplomarbeit Hochschule Mannheim 2005.*