

Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Medieninformatik

Stereo-basierte Echtzeit-Hinderniserkennung für unbemannte Flugsysteme

Bachelorarbeit

Hagen Hiller
Geboren am 04.06.1992 in Berlin

Matrikelnummer 110514

1. Gutachter: Prof. Dr. Volker Rodehorst
2. Gutachter: Junior-Prof. Dr. Florian Echtler

Datum der Abgabe: 22.02.2016

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Weimar, den 22.02.2016

.....
Hagen Hiller

Zusammenfassung

Die Verwendung ferngesteuerter unbemannter Flugsysteme ist längst nicht mehr rein militärischer Natur. Auch in der Zivilgesellschaft werden diese zur Produktion von Filmen, zu Vermessungszwecken oder aber in privaten Bereichen verwendet. Jedoch werden diese privaten Anwendungen zum Teil kritisch beäugt. Trotz dessen ist die autonome Erkundung von unbekannten Gebieten ein weit erforschter Bereich.

Gerade in Arealen ohne verfügbare GPS Verbindung erfolgt die automatische Wegfindung unter Verwendung anderer Sensoren. Die Verwendung einer Stereo-Kamera-Konfiguration ist dabei der einfachste Weg optisch eine dreidimensionale Rekonstruktion der Szene zu erstellen anhand welcher etwaige Hindernisse gesucht werden. Im Rahmen dieser Arbeit wurden zwei Systeme zur Erkennung von Hindernissen in Echtzeit entwickelt. Die gleichzeitige Lokalisierung und Kartographierung (SLAM - *Simultaneous Localization and Mapping*), beispielsweise innerhalb eines einsturzgefährdeten Gebäudes ist somit ein mögliches Anwendungsgebiet. Dabei steht die sichere Navigation des Flugsystems im Vordergrund. Folglich ist gerade der Aspekt der Erkennung in Echtzeit, sowie der Erhalt der Hindernispositionen eine unabdingbare Anforderung. Die zu berechnenden Daten über die Position potentieller Hindernisse müssen dabei effizient zur Entwicklung einer Strategie zur Vermeidung verwendet werden können.

Inhaltsverzeichnis

1 Einführung	3
1.1 Motivation	3
1.2 Hardwarekomponenten	4
1.3 Ziel der Arbeit	5
2 Zugrunde liegende Konzepte und Algorithmen	7
2.1 Epipolareometrie	7
2.2 Kamerakalibrierung und Rektifizierung	9
2.3 Stereo Matching	10
2.3.1 Lokale Methoden	11
2.3.1.1 Block Matching	12
2.3.2 Globale Methoden:	13
2.3.2.1 Semi Global Block Matching	13
2.4 mvStereoVision Framework	17
3 Optische Verfahren zur Hinderniserkennung	20
3.1 Aktiv optische Algorithmen	20
3.2 Passiv optische Algorithmen	21
4 Entwickelte Hinderniserkennungen	24
4.1 Grundlegende Operationen	24
4.2 Subimage Detection	26
4.3 Samplepoint Detection	29
4.4 Applikationsstruktur	31
5 Evaluation	32
5.1 Testsetup	32
5.2 Durchführung und Ergebnisse	35
5.2.1 Robustheit	35
5.2.1.1 Subimage Detection	35
5.2.1.2 Samplepoint Detection	39
5.2.2 Weitere Versuche	41

5.2.2.1	Einfluss der <i>SGBM</i> Block Größe	42
5.2.2.2	Erkennung von reflektierenden und durchsichtigen Bereichen	43
5.2.2.3	Erkennung unter Veränderung des Samplepoint Radius	46
5.3	Diskussion	48
5.3.1	Robustheit	48
5.3.1.1	Subimage Detection	48
5.3.1.2	Samplepoint Detection	50
5.3.2	Einfluss der SGBM Blockgröße	51
5.3.3	Reflektierende und durchsichtige Flächen	52
5.3.4	Veränderung des Subimage Radius	52
5.3.5	Laufzeit	53
5.3.6	Gegenüberstellung	57
6	Limitierungen und Lösungsansätze	60
6.1	Bestehende Limitierungen und Lösungsansätze	60
6.2	Diskussion	62
7	Fazit und zukünftige Arbeiten	66
	Literaturverzeichnis	68

Kapitel 1

Einführung

1.1 Motivation

Die vorliegende Arbeit ist im Forschungsgebiet Robotik angesiedelt und behandelt bildbasierte Verfahren zur Erkennung von Hindernissen für autonome Flugsysteme (UAS¹). Diese Verfahren sind für die Navigation sowie für die autonome Erkundung von unbekannten oder unzugänglichen Gebieten unerlässlich. Dabei müssen nicht nur die physikalischen Eigenschaften der Drohne betrachtet werden sondern auch die Fusion verschiedenster Sensoren.

Ein wichtiges Kriterium in der Entwicklung autonomer Roboter ist die Erkennung von Hindernissen in Echtzeit. Dabei muss jedoch zuerst definiert werden was vom System als potentielles Hindernis erkannt werden soll. Prinzipiell sind alle Objekte welche sich in der unmittelbaren Nähe des Systems befinden eine Gefahrenquelle. Im Fall eines Kamera-basierten Systems mit lediglich einer Hauptblickrichtung, ist die Detektion jedoch beschränkt, so dass eine Einschränkung der zugelassenen Manöver, z.B. auf eine Bewegung lediglich in Blickrichtung der Kamera, erfolgen sollte. Dies schließt eine Kollision mit Objekten außerhalb des Sichtfeldes aus. Auch sehr weit entfernte Objekte sind prinzipiell nicht als Hindernis anzusehen, wobei die maximal zu betrachtende Gefahrendistanz abhängig von der aktuellen Bewegungsgeschwindigkeit angepasst werden muss. Unter Betrachtung dieser Gesichtspunkte wird ein Hindernis innerhalb dieser Arbeit als ein Objekt definiert welches sich innerhalb eines definierten Distanzbereichs und innerhalb des Sichtfeldes der Kamera befindet.

Die hauptsächliche Anwendung des im Rahmen dieser Arbeit entwickelten Systems zielt auf die autonome Navigation von unbemannten Flugsystemen ab.

¹Unmanned Aircraft System

Beide vorgeschlagenen Methoden nutzen Stereo-Bilddaten, um in Echtzeit 3D-Informationen der Umgebung des Systems zu berechnen. Sofern Hindernisse in einem definierten Gefahrenbereich vor der Kamera detektiert werden, alarmiert das System und stellt die 3D-Koordinaten der Hindernisse in Form einer Punktfolge für die Hindernisvermeidung / Flugplanung bereit. Prinzipiell ist es ebenfalls möglich die entwickelten Algorithmen und Methoden im Automobil Bereich zu verwenden um beispielsweise Objekte vor oder hinter dem Kraftfahrzeug zu erkennen und deren Distanz zu ermitteln.

1.2 Hardwarekomponenten

Die aktive Entwicklung der Methoden und Algorithmen erfolgte im Hinblick auf eine Verwendung dieser auf dem von Ascending Technologies [Ascending Technologies, 2015] entwickelten UAS Pelican (Abbildung 1). Dabei handelt es sich dabei um einen Quadrocopter der speziell für Forschungszwecke entwickelt wurde. Er ist mit einem Bordcomputer ausgestattet, der die nötige Leistung für die Entwicklung der Algorithmen bereitstellt (3rd Generation Intel Core i7). Weiterhin wurden zwei MatrixVision BlueFOX mv-MLC200wC Industriekameras [MATRIX VISION, 2015] (Abbildung 2) mit einem Sichtfeld von je 100° als visuelles System verwendet. Die maximale Auflösung beider Kameras beträgt 752×480 bei 60 möglichen Bildern pro Sekunde, in Abhängigkeit verschiedener Parameter (verwendete Verschlusszeit, aufgenommene Bitrate, u. a.). Für den Echtzeit-Aspekt des Systems werden beide Kameras in einem horizontalen und vertikalen Binning-Modus verwendet. Dies halbiert die Anzahl der Bildpunkte in beiden Dimensionen auf 376×240 , indem jeweils 4 Pixel des Sensors zu einem Pixel im ausgehenden Bild zusammengefasst werden. Dadurch verringert sich der Berechnungsaufwand und die Aufnahmerate der Kameras auf bis zu 170 Einzelbilder pro Sekunde maximiert wird, was ebenfalls die gesamte Performance der entwickelten Systeme verbessert. Für die Implementierung der Methoden wurde die freie Computer Vision Bibliothek OpenCV [OpenCV, 2015] verwendet. Diese stellt benötigte algebraische Grundoperationen sowie bestimmte Algorithmen, welche im Rahmen dieser Arbeit genutzt wurden, zur Verfügung.

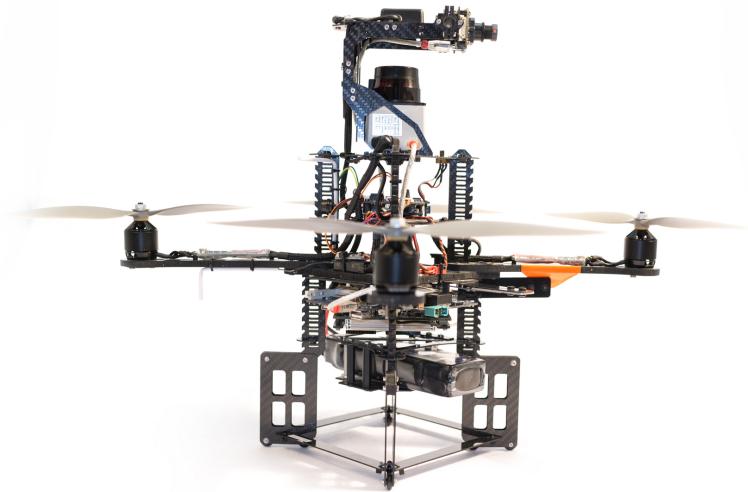


Abbildung 1: AscTec Pelican [Ascending Technologies, 2015]



Abbildung 2: MatrixVision BlueFOX mv-MLC200wC
[MATRIX VISION, 2015]

1.3 Ziel der Arbeit

Basierend auf photogrammetrischen Konzepten ist es möglich räumliche Tiefe aus zweidimensionalen Bilddaten zu berechnen. Generell werden mindestens zwei Bilder derselben Szene aus unterschiedlichen Standpunkten für die Berechnung dreidimensionaler Informationen benötigt. Bei Verwendung einer einzelnen Kamera muss für ein Bildpaar die relative Orientierung zwischen den Kameras geschätzt werden, wobei keine Information über die Skalierung vorliegt. Bei der Verwendung eines Stereosystems ist die relative Orientierung beider Kameras zueinander bereits bekannt, was eine direkte Berechnung metrischer Koordinaten ermöglicht. Da Zuverlässigkeit sowie Genauigkeit vor allem im Kontext der Navigation in Innenräumen sehr wichtig ist wurde in dieser Arbeit auf ein Stereo-Setup gesetzt.

Vor diesem Hintergrund werden in Kapitel 2 der Arbeit zugrundeliegende Algorithmen und Konzepte erläutert. Weiterhin wird das entwickelte Framework zur Bildaufnahme und Vorprozessierung der Bilder grundlegend beschrieben. Anschliessend werden einige State of the Art Methoden der Hinderniserkennung beschrieben wobei dabei zwischen aktiven und passiven optischen Systemen unterschieden wird. Diese Einteilung dient einerseits dafür einen Überblick über bereits bestehende Techniken sowie implementierte Systeme zu erhalten, andererseits um auch die Vor- und Nachteile der jeweiligen Technik herauszuarbeiten. Im Anschluss daran beschreibt Kapitel 4 die beiden entwickelten Methoden zur bildbasierten Hinderniserkennung und deren Implementierung detailliert. Anschließend erfolgt die Evaluation beider Verfahren wobei die Erkennung verschiedener Hindernisgrößen getestet wird. Weitere Tests hinsichtlich zukünftiger Verbesserungen zeigen auf welches aktuellen Limitierungen durch die verwendeten Konzepte vorliegen. Die anschliessende Diskussion wertet die im Rahmen der Evaluation erlangten Ergebnisse weitergehend aus und stellt beide Algorithmen hinsichtlich der Robustheit der Erkennung, sowie der erreichten Performance gegenüber. Kapitel 6 erläutert mögliche Ursachen bestehender Limitierungen und gibt Ansätze zur Lösung aus der Fachliteratur sowie eigene Konzepte zur Bewältigung dieser. Kapitel 7 zieht ein Résumé aus den Ergebnissen der Arbeit und gibt einen Ausblick auf mögliche zukünftige Arbeiten in diesem Bereich.

Kapitel 2

Zugrunde liegende Konzepte und Algorithmen

Dieses Kapitel beleuchtet dieser Arbeit zugrunde liegende Konzepte und Algorithmen. Zunächst wird das Prinzip der Epipolargeometrie beschrieben welches ein wichtiges mathematisches Modell für photogrammetrische Verfahren darstellt. Daraufhin folgt eine Erläuterung des Terms *Stereo Matching* sowie eine Klassifizierung in lokale und globale Algorithmen. Im Anschluss daran wird das Prinzip des *Block Matching* Algorithmus beschrieben, welcher die Grundlage für den im Rahmen dieser Arbeit verwendeten *Semi Global Block Matching* Algorithmus bildet. Anschließend erfolgt eine Erläuterung des im Rahmen dieser Arbeit entwickelten Frameworks sowie Details zur Implementierung dessen.

2.1 Epipolargeometrie

Photogrammetrische Verfahren nutzen oft das Konzept der Epipolargeometrie. Dieses mathematische Modell beschreibt die geometrische Beziehung zwischen verschiedenen Kamerabildern desselben Objektes, sowie die Beziehung korrespondierender Bildpunkte. Generell gesehen ist die der Epipolargeometrie zugrundeliegende Kamera durch das Lochkamera-Modell beschrieben. Dabei befindet sich jeder dreidimensionale Punkt des aufgenommenen Objektes mit dem Projektionszentrum sowie dem entsprechenden Bildpunkt auf einer Geraden. Unter Zuhilfenahme der räumlichen Orientierung, sowie der intrinsischen Parameter der Kamera (beispielsweise die Brennweite, Koordinaten des Bildhauptpunktes) ist es möglich den Schnittpunkt mehrere solcher Raumgeraden zu berechnen um die dreidimensionalen Koordinaten des Objektpunktes zu erhalten. Dabei gilt generell: wenn ein Punkt P im linken Bild gegeben ist, so wird die Suche des korrespondierenden Punktes P' auf die Epipolarlinie des

rechten Bildes reduziert. Die algebraische Repräsentation der relativen Orientierung zweier Bilder ist die Fundamentalmatrix F .

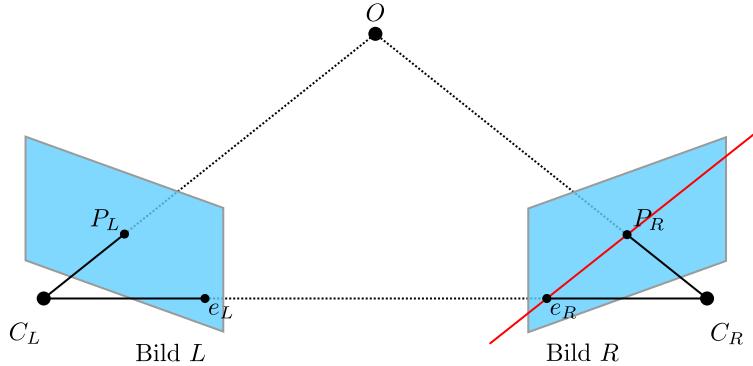


Abbildung 3: Darstellung der Epipolargeometrie.

Abbildung 3 visualisiert diesen Prozess. Gegeben sind die beiden Projektionszentren C_L und C_R zweier Kameras, sowie die Bildpunkte P_L und P_R , welche beide die Position des Objektpunktes O in beiden Bildern beschreiben. Mit der Information über C_L und P_L allein kann keine Information über die räumliche Tiefe des Objektpunkts O abgeleitet werden. Die Projektion aller möglichen Positionen von O entlang des Raumstrahls in das rechte Bild ergibt eine Linie, die so genannte Epipolarlinie. Rechnerisch ergibt sich diese aus der Multiplikation des Bildpunktes mit der Fundamentalmatrix. Betrachtet man den Strahl C_L, O aus Perspektive des rechten Bildes, und verschiebt O entlang dessen, so ergibt sich die Epipolarlinie in R . Somit ist es möglich die Epipolarlinie e_L, P_{x_2, y_2} zu bestimmten. Anhand dieser kann der Suchraum für den gesuchten korrespondierenden Bildpunkt im Rechten Bild auf die Epipolarlinie eingegrenzt werden. Kann der gesuchte korrespondierende Punkt P_R mittels eines *Matching*-Ansatzes gefunden werden, ist es möglich die 3D Position des Objektpunktes zu triangulieren.

Die Epipole sind durch den Schnittpunkt der Basislinie mit den beiden Bildebenen definiert. Im Falle des so genannten Stereonormalfallen ist ein Schnittpunkt der Basislinie mit den Bildebenen nicht möglich, da die Epipole in der Unendlichkeit und parallel zur x-Achse liegen. Da in diesem Fall die Bildebenen in einer Ebene liegen und die Blickrichtungen und Koordinatenachsen der Kamerassysteme parallel ausgerichtet sind, sind alle Epipolarlinien parallel. Des Weiteren beschränkt sich die Suche nach einem korrespondierenden Punkt (x_L, y_L) auf eine Suche entlang der horizontalen Epipolarlinie mit konstanter y-Koordinate ($y_R = y_L$).

2.2 Kamerakalibrierung und Rektifizierung

Um den Prozess der Korrespondenzanalyse zu vereinfachen sind drei verschiedene Schritte nötig:

1. Kalibrierung des Stereo-Kamerasystems
2. Transformation der Bilder zum Stereonormalfall
3. Rektifizierung der Bilder

Zunächst müssen die intrinsischen und extrinsischen Parameter des Stereo-Kamerasystems berechnet werden. Die Ermittlung der intrinsischen Parameter ist dabei der Hauptbestandteil der Kalibrierung. Mit Hilfe weiterer Schritte kann auch die relative Orientierung der beiden Kameras des Systems ermittelt werden. Zu Beginn des Kalibrier-Prozesses werden Bilder eines Kalibrierobjektes aufgenommen. Dies ist meistens ein einfaches Schachbrettmuster mit ungleicher Anzahl an Quadranten, oder auch ein radiales Objekte mit einem spezifischerem Aufbau. Die Dimensionen des Kalibrierobjektes sind dabei bekannt. Der Prozess der Kalibrierung zielt darauf ab die „inneren“ Parameter der Kamera zu bestimmen. Dies beinhaltet einerseits die Kameramatrix in welcher Parameter wie die *Principal Distance* (Abstand vom Projektionszentrum zur Bildebene), der Bildhauptpunkt (*Principal Point*) sowie das Seitenverhältnis der Pixel und der Scherungswinkel der Kamera. Weiterhin werden geometrische Verzerrungen des Bildes, beispielsweise Radiale Verzeichnungen durch die verwendete Optik, parametrisiert und somit nutzbar für die Entzerrung der Bilder in weiteren Schritten. Zudem werden bei der Kalibrierung die extrinsischen (äußereren) Parameter, d.h. die relative Orientierung der beiden Kameras zueinander, bestimmt. Zu diesen zählt die Translation (x , y und z) sowie die Rotation der Kamera um die jeweiligen Achsen des Weltkoordinatensystems (ausgedrückt durch drei Winkel). Ein spezieller Aspekt bei der Kalibrierung von Stereo-Systemen ist die Berechnung der extrinsischen Parameter von einer Kamera zur anderen. Die dabei berechnete Translation einer Kamera zur Referenzkamera wird als Basislinie bezeichnet. Häufig ist die linke Kamera dabei die Referenzkamera. Die Kenntnis über die relative Orientierung von Stereo-Kameras führt dazu, dass man identifizierte korrespondierende Punkte in den Bildpaaren direkt metrisch triangulieren kann. Der Prozess der Parameterfindung ist in der Regel ein Prozess welcher einmalig nach dem Aufbau ausgeführt werden muss und solange währt, wie sich nichts am Aufbau der Kameras ändert (Optik, Translation, Rotation). Anschliessend werden die Bilder mithilfe der erhaltenen Parameter entzerrt.

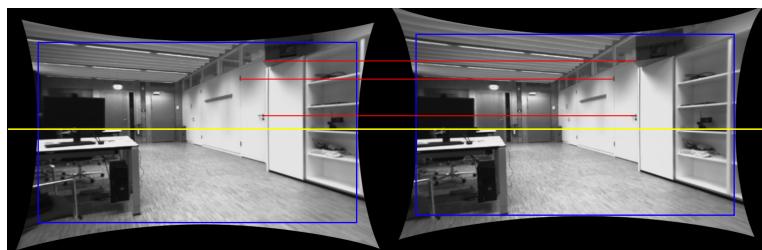
Folgend aus der Kalibrierung der Kameras beginnt der Prozess der Rektifizierung der Bilder. Dieser kann dabei in zwei Schritte unterteilt werden, zunächst die geometrische Enzerrung der Bilder, anschliessend die Berechnung der *region of interest* (ROI) (der in beiden Bildern maximale sichtbare Bereich, sodass der gesamte Bildbereich gefüllt ist (siehe Abbildung 4 (b))) beider Bilder. Mithilfe der zuvor berechneten intrinsischen und extrinsischen Parameter kann nun die Transformation der beiden Kamerabilder in den Stereo-Normalfall erfolgen. Die Rotationen, die bewirken, dass die Epipole auf der x-Achse liegen, werden auf die Bilder angewendet, sodass die Epipole beider Bilder auf der x-Achse liegen. Nach Verschiebung der Epipole entlang der x-Achse ins Unendliche ist Parallelität der Epipolarlinien gewährleistet. Daraus resultieren parallele und orthogonale Blickrichtungen zur identischen Bildebene, somit sind beide Bildebenen Element derselben Ebene. Die finale ROI ergibt sich aus der Konjunktion der einzelnen ROIs. Nach der Rektifizierung, der geometrischen Entzerrung und dem Anwenden der ROIs auf beiden Kamerabildern entspricht die jeweilige Pixelreihe im linken Bild der Pixelreihe mit dem selben Index im rechten Bild. Abbildung 4 verdeutlicht den Ablauf der Rektifizierung, so ist in Abbildung 4 (c) erkennbar das sich die korrespondierenden Bildpunkte auf einer Epipolarlinie befinden.

2.3 Stereo Matching

Mittels *Stereo Matching* Verfahren wird versucht, für jeden Pixel in einem Bild der korrespondierende Punkt in einem zweiten Bild zu finden. Im Stereonormalfall ist die relative Orientierung der beiden Kameras bekannt und konstant. Des Weiteren werden korrespondierende Punkte in diesem Fall als einfache Differenz der x-Koordinaten, so genannte Disparitäten, dargestellt und gespeichert. Mithilfe der verschiedenen Perspektiven können Disparitäten zwischen korrespondierenden Pixeln berechnet werden. Die räumlichen 3D-Koordinaten der aufgenommenen Szene stehen im direkten (funktionalen) Zusammenhang mit der Information über die Disparität, der relativen Orientierung der Kameras sowie der Kamerakalibrierung. Durch die Transformation zum Stereonomalfall vereinfacht sich dieser Zusammenhang enorm. Im Laufe des *Stereo Matching* Prozesses kommt es zu zwei wesentlichen Problemstellungen: die Berechnung der Disparität (Stereo Korrespondenz) sowie die Invertierung der projektiven Geometrie, um dreidimensionale Informationen aus der errechneten Disparität zu erhalten. Sofern eine Lösung beider Probleme vorhanden ist, können diese Informationen durch einfache Triangulierung errechnet werden.



(a) Korrespondierende Punkte innerhalb der entzerrten Bilder.



(b) Korrespondierende Punkte in den stereo-rektifizierten und geometrisch entzerrten Bildern. Die ROIs (blau) kennzeichnen die jeweils maximal möglichen rechteckigen Flächen.



(c) Korrespondierende Punkte in den beschnittenen, skalierten und rektifizierten Bildern.

Abbildung 4: Veranschaulichung des Effekts der Rektifizierung

2.3.1 Lokale Methoden

Zur Berechnung der Disparität in lokalen *Stereo Matching* Algorithmen gilt grundlegendes Prinzip: „Finde Pixel P_2 korrespondierend zu P_1 im Referenzbild“. Dabei wird die Korrelation von P_1 und P_2 unter Einbezug der lokalen Nachbarschaft bestimmt und die maximale Korrelation als Indikator für die Übereinstimmung von Pixeln herangezogen. Für die Korrelation wird jeweils die Bildinformation aus der lokalen Nachbarschaft um die zu testenden Pixel herangezogen, beispielsweise 11×11 Pixel. Geläufige Methoden dafür sind

Sum of Absolute Differences (SAD) (Hirschmüller 2011), *Zero-mean Normalized Cross-Correlation* (ZNCC) (Chen & Medioni, 1999), (Sára, 2002), *Sum of Squared Differences* (SSD) (Cox et al., 1996). Jedoch können aufgrund des recht einfachen Ansatzes, der bei starken perspektivischen Änderungen und untexturierten Bereichen keine sinnvollen Ergebnisse liefert, falsche Berechnungen der räumlichen Tiefe auftreten, da benachbarte Pixel verschiedene Disparitäten aufweisen können (beispielsweise an horizontalen Kanten wie Türrahmen etc.). Strukturell gesehen sind lokale Methoden einfacher gehalten als globale Methoden, wodurch ein hoher Grad an Optimierung in der Implementierung möglich ist.

2.3.1.1 Block Matching

Einen der einfachsten Ansätze zur Berechnung von Korrespondenz zwischen zwei Bildern bietet der *Block Matching* Algorithmus. Bei dieser lokalen Methode werden Blöcke bestimmter Größe mithilfe von Korrelation (Vergleich zweier Signale auf Übereinstimmung) auf Korrespondenz untersucht. Dabei reduziert die Epipolareometrie diesen Prozess auf ein eindimensionales Problem, so dass ein Block im linken Bild mit allen potentiell möglichen Blöcken auf der Epipolarlinie des rechten Bildes verglichen wird. Das eigentliche Matching erfolgt über die Berechnung des gewählten Korrelationswerts eines Blockes mithilfe einer Energiefunktion.

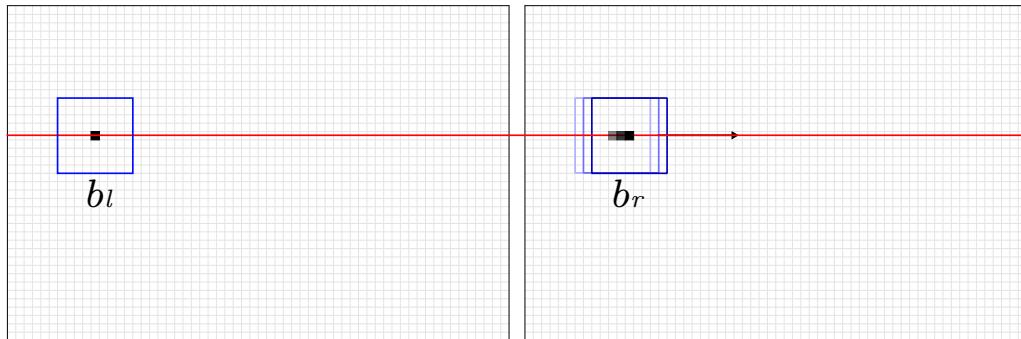


Abbildung 5: Visualisierung des Block Matching Algorithmus

Diesen Prozess veranschaulicht Abbildung 5 . Durch den Fokus auf einzelne Pixelreihen und deren unmittelbare Umgebung ist dieses Verfahren wesentlich schneller, aber auch ungenauer als globale Matching Algorithmen.

2.3.2 Globale Methoden:

Bei globalen *Stereo Matching* Algorithmen wird eine globale Annahme der zu betrachtenden Szene erstellt sowie eine ebenfalls globale Kostenfunktion definiert, welche es zu minimieren gilt. Dabei werden im Gegensatz zu lokalen Methoden die Korrespondenzen in einer Pixelreihe und die des gesamten Bildes miteinander verglichen. Zur Vereinfachung dieses Vorgangs betrachten einige Algorithmen nur die durch die Epipolargeometrie vorgegebenen Bildbereiche, wodurch ein zweidimensionales Problem auf ein eindimensionales reduziert wird. Resultierend daraus liegt die Stärke globaler Methoden in der Bewältigung schwacher Texturen sowie auftretender Okklusionen und unterschiedlicher Lichteinfälle, was, aufgrund der höheren Komplexität in einem höheren Rechen- und Speicheraufwand mündet. Weitere Verbesserungen der Ergebnisse können durch Techniken wie Dynamische Programmierung und *Graph Cut* erreicht werden ([Zureiki et al., 2008]).

2.3.2.1 Semi Global Block Matching

Das im Rahmen dieser Arbeit verwendete Verfahren zur Berechnung von Disparitätenkarten ist der in der freien Computer Vision Bibliothek OpenCV [OpenCV, 2015] implementierte *Semi Global Block Matching* Algorithmus. Diese leicht abgewandelte Implementierung des *Semi Global Matching* (SGM) Algorithmus von Hirschmüller et. al [Hirschmuller, 2005] zeichnet sich sowohl durch seine guten Ergebnisse, als auch durch seine performante Berechnung aus. Durch die Verwendung von Blöcken anstelle einzelner Pixel ist dieser zu Teilen den Globalen Methoden zuzuordnen. Unter Zuhilfenahme dieses Verfahrens ist es möglich, Disparitätenkarten in Echtzeit zu berechnen. Im Folgenden wird zunächst die grundlegende Funktionsweise des SGM erläutert. Im Anschluss daran werden die Unterschiede zum SGBM herausgearbeitet sowie eine kurze Erklärung der wesentlichsten Parameter dessen vorgenommen.

SGM:

Die grundlegende Idee des *Semi-Global Matching* Algorithmus besteht in dem pixelweisen Matching mittels *Mutual Information*¹. Ausgangsvoraussetzung dafür sind verschiedene Bilder derselben Szene mit vorhandener und bekannter Epipolargeometrie. Ein weiteres Kernelement des SGM ist die Anwendung eines globalen zweidimensionalen Glattheitskriteriums, was durch die Kombination mehrerer eindimensionaler Beschränkungen approximiert wird. Zunächst

¹ Mutual Information (MI), zu deutsch Transinformationen, beschreiben den statistischen Zusammenhang zweier Zufallsgrößen. Genauer gesagt beschreiben sie die Höhe des Informationsgehaltes einer Zufallsvariable aus anderen zufällig gewählten Werten.

werden für jeden Pixel P aus der Intensität I_{bp} sowie der vermuteten Korrespondenz I_{mq} auf der Epipolarlinie $q = e_{bm}(pd)$ die Matching Kosten berechnet. Zur Berechnung der MI wird zunächst eine initiale Disparitätenkarte benötigt. Diese wird nach dem Ansatz von Kim et al. [Kim et al., 2003] zufällig gewählt, um die Kosten berechnen zu können. Zur Steigerung der Performance wird die Disparitätenkarte zunächst nur mit halber Auflösung berechnet, wodurch der Rechenaufwand um den Faktor 8 reduziert wird. Zur Vermeidung falscher Kostenberechnungen durch auftretendes Rauschen innerhalb des Bildes werden benachbarte Disparitäten mit in die Berechnungen einbezogen.

Das letzte Problem besteht in der Berechnung der Korrespondenz sowie der daraus resultierenden Disparitäten. Dabei wird nach der Disparität D mit der geringsten berechneten Energiefunktion gesucht. Anstatt nun einfach den minimalsten Pfad der Kosten zu summieren, werden zusätzlich auch andere Richtungen zur aktuellen Disparität mit einbezogen (siehe Abbildung 6).

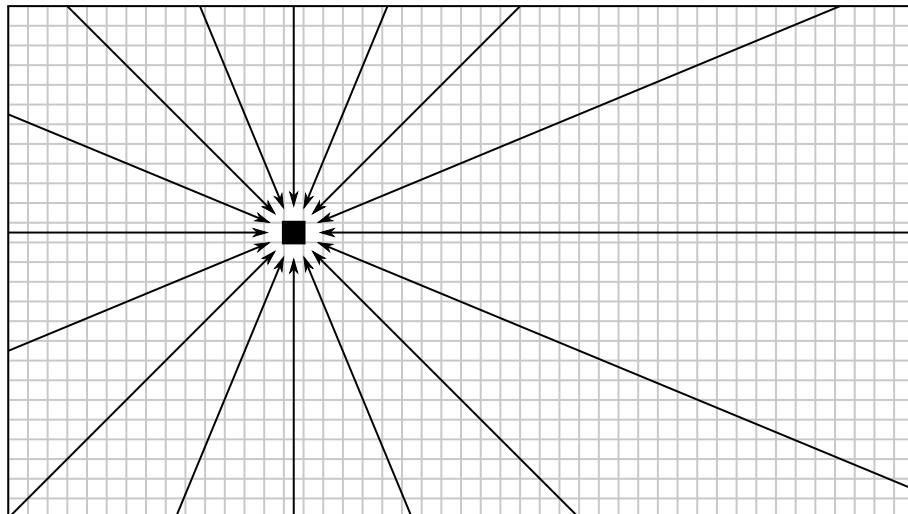


Abbildung 6: Darstellung der verschiedenen betrachteten Richtungen des SGBM

Um das 2D-Glattheitskriterium möglichst gut zu approximieren, sollten dabei mindestens 8 Richtungen vorliegen, eine vermehrte Anzahl an Richtungen, etwa 16, ist dabei vorteilhaft. Die berechnete Disparität ergibt sich aus den minimalen Kosten anhand dieser Pfade.

SGBM:

Der *Semi Global Block Matching* Algorithmus ist ein in der Computer Vision Library OpenCV implementiertes Verfahren zur schnellen Berechnung

von Disparitätenkarten. Die Grundlage dafür bietet Hirschmüller et al.'s SGM [Hirschmuller, 2008] mit den folgenden grundlegenden Änderungen:

- C.1 Statt den originalen 8 bzw. 16 Richtungen werden nur 5 betrachtet.
- C.2 Es werden standartmäßig keine einzelnen Pixel sondern Blöcke verglichen.
- C.3 Anstelle der *Mutual Information* Kostenfunktion wird das von Birchfield et al. vorgestellte *Sub-Pixel Dissimilarity Measurement* Verfahren verwendet [Birchfield and Tomasi, 1998].
- C.4 Verschiedene Schritte der Vor- und Nachprozessierung des *StereoBM* Algorithmus werden verwendet, dazu zählen Filtermethoden wie quatdramatische Interpolation oder Sprenkelfilter.

Dies ermöglicht eine schnelle Berechnung der Disparitäten auf einem qualitativ hochwertigen Niveau. Der geringe Verlust an Qualität, hervorgerufen durch die geringere Anzahl an betrachteten Richtungen, kann in Anbetracht der Berechnung in Echtzeit vernachlässigt werden.

Abbildung 7 stellt die originalen Bilder der Szene ([Middlebury University, 2015]), die mithilfe von strukturiertem Licht aufgenommenen *Ground Truth* Bilder sowie die jeweiligen berechneten Tiefenkarten durch den SGM und den SGBM dar. Aus dieser ist der Qualitätsverlust des SGBM beim Tsukuba- sowie beim Teddy Datensatz zu erkennen.

Trotz dessen sind die Grundformen sowie die berechnete Tiefe klar erkennbar. Grundlegend bietet der SGBM eine qualitativ hochwertige Berechnung von Tiefenkarten. Weiterhin ist die Berechnung sowie die daraus resultierenden Disparitätenkarten stabiler. Bei der Berechnung durch OpenCV's einfachen Block Matching Algorithmus kann es, in Abhängigkeit der zu rekonstruierenden Szene zu einem Flackern zwischen den einzelnen Einzelbildern kommen. Dies würde bedeuten das eventuell keine Informationen zwischen zwei Frames vorliegen was wiederum zu Fehlern in der Erkennung führen würde.

Weiterhin existiert eine Reihe verschiedener Parameter welche die Berechnung der Disparity Map aktiv beeinflussen. Im folgenden werden die zur Initialisierung obligatorischen Parameter in ihrer Funktionsweise grob beschrieben.

- ***minDisparity*:**

Der Parameter *minDisparity* begrenzt den messbaren Tiefenbereich nach oben. Er beschreibt also die minimale zu erkennende Disparität und somit die maximale zu erkennende Distanz.

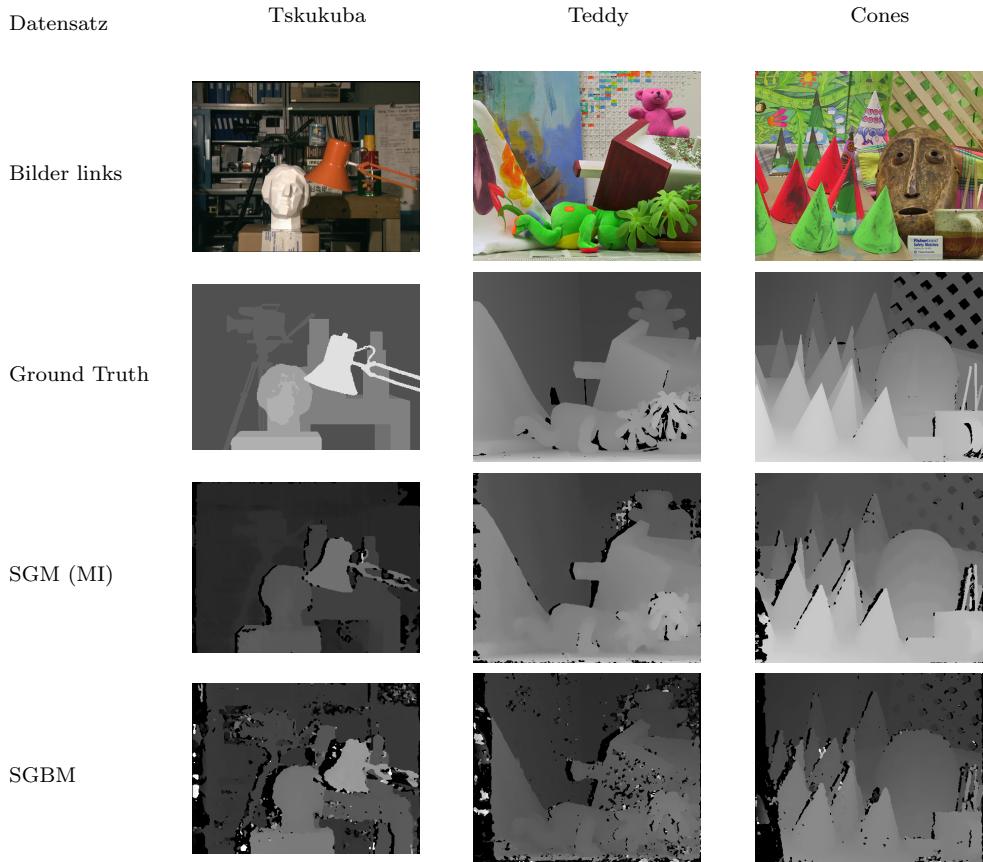


Abbildung 7: Vergleich zwischen Ground Truth Bildern sowie dem SGM(2005) und OpenCV's SGBM

- ***numDisparities:***

Mit Hilfe der *Number of Disparities numDisparities* wird die Breite des Matchingbereiches festgelegt. Geht man davon aus, dass das linke Bild Referenz ist, wird damit festgelegt, wie viele Pixel im rechten Bild auf Korrespondenz untersucht werden sollen. In Abhängigkeit der Größe beider Parameter können in der Tiefenkarte Bereiche entstehen, die keine sinnvolle Information enthalten. Die damit verbundene weitere Verfahrensweise wird im Abschnitt 4.1 näher erläutert.

- ***blockSize:***

Die *blockSize* des SGBM beschreibt die Größe des Matching Blocks in Pixeln. Je größer der Wert gewählt wird, desto weicher wird die resultierende Disparitätenkarte. Dabei gehen jedoch auch Detailinformationen, beispielsweise bezüglich feiner Strukturen in der Szene verloren. Ist

der Wert niedrig so ist es unter Umständen schwerer homogene Flächen korrekt zu Matchen. Zudem können eventuell weniger Korrespondenzen gefunden werden.

2.4 mvStereoVision Framework

Das verwendete *Framework* zur Bildaufnahme und Berechnung der Disparitätenkarten bedient sich der von Matrix Vision zur Verfügung gestellte Bibliothek [MATRIX VISION, 2015] zur Kommunikation mit den Kameras, sowie OpenCV [OpenCV, 2015] zur Verarbeitung der Bilder. Die wesentlichen Funktionen werden im folgenden näher beleuchtet.

Bildaufnahme:

Die Aufnahme der einzelnen Bilder beginnt bei einer Anfrage an die Kamera nach den jeweiligen Rohdaten. Diese werden nun an die *Stereosystem* Klasse weitergegeben, in welcher die Bilder anschliessend mithilfe der durch die Kalibrierung ermittelten Parameter in den Stereonormalfall transformiert, geometrisch entzerrt und beschnitten werden. Damit ist der Abschnitt der Vorprozessierung abgeschlossen, an dessen Ende sich beide Bilder im *Stereopair* Objekt befinden. Die Aufnahme der Bilder erfolgt dabei in separaten Threads um unabhängig von weiteren Berechnungen Bilder aufzunehmen. Anschliessend wird das *Stereopair* Objekt an die Berechnung der Disparitätenkarte in einen separaten Thread übergeben. Sofern eine neue Tiefenkarte vorliegt wird innerhalb des Hauptprogramms mit Hilfe eines Wahrheitswertes darüber informiert. Der gesamte Prozess wird in Abbildung 8 visualisiert.

Distanzberechnung:

Für die Berechnung von 3D-Koordinaten auf der Basis von Bildkoordinaten sowie der inneren und relativen Orientierung des Stereo-Systems wird die Reprojektionsmatrix Q , welche im Rahmen der Rektifizierung berechnet wird benötigt. Diese ist wie folgt aufgebaut:

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{C_x - C'_x}{T_x} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & a & b \end{bmatrix} \quad (2.1)$$

Dabei beschreiben $-C_x$ und $-C_y$ die beiden negativen Koordinaten des Bildhauptpunktes der Referenzkamera, beispielsweise der linken. Der Parameter f beschreibt die Brennweite der linken Kamera. T_x ist die horizontale Translation zwischen beiden Kameras. Die in der Q-Matrix enthaltenen Parameter sind

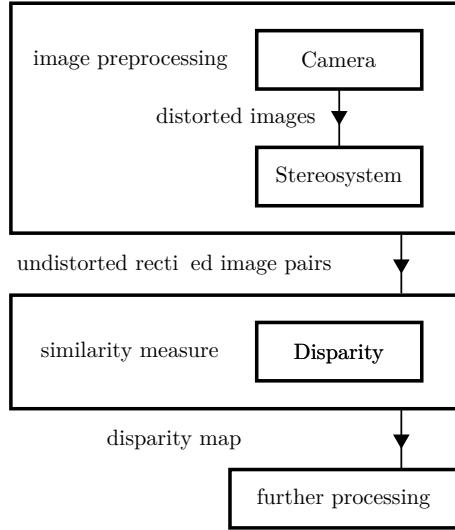


Abbildung 8: Ablauf der Bildaufnahme, Berechnung der Disparitätskarte sowie weiteren Verarbeitung

mit Ausnahme von C'_x alle die der linken Kamera. Es wird angenommen, dass es sich um einen (nach der Rektifizierung vorhandenen) Stereonormalfall handelt und sich die beiden Strahlen der Bildhauptpunkte in der Unendlichkeit schneiden. Dies führt zu einer Annäherung des unteren rechten Parameters in 2.1 an 0. Zur einfacheren Darstellung und Erläuterung werden die Basislinie der Kamera $\frac{-1}{T_x}$ im folgenden als a sowie die Darstellung beider Bildhauptpunkte $\frac{C_x - C'_x}{T_x}$ als b notiert.

Eine Eigenschaft der Reprojektionsmatrix ist die einfache Berechnung von dreidimensionalen Weltkoordinaten. Diese Berechnung erfolgt mittels Gleichung 2.2. Dabei beschreibt der Parameter $d(I_x, I_y)$ den Pixelwert der Disparitätskarte an den Bildkoordinaten I_x, I_y .

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot \begin{bmatrix} I_x \\ I_y \\ d(I_x, I_y) \\ 1 \end{bmatrix} \quad (2.2)$$

$$3d_coordinate = \left[\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right]$$

Die eigentliche Berechnung der einzelnen Parameter ergibt sich somit aus der Multiplikation des Vektors mit der Reprojektionsmatrix. Die dreidimensionalen Koordinaten ergeben sich damit aus:

$$\begin{aligned} X &= I_x - C_x \\ Y &= I_y - C_y \\ Z &= f \\ W &= d \cdot a + b \end{aligned} \tag{2.3}$$

Kapitel 3

Optische Verfahren zur Hinderniserkennung

Ein aktuell erforschtes Feld in der Entwicklung autonomer Systeme ist die Vermeidung von Hindernissen in Echtzeit. Die wesentliche Grundlage dafür besteht in der Erkennung der Hindernisse. Dies geschieht oft mit Hilfe optischer Messtechniken.

Im folgenden Kapitel werden einige State of the Art Verfahren erläutert. Dabei werden zunächst verschiedene aktive optische Systeme näher beschrieben, bei denen die betrachtete Szene aktiv ausgeleuchtet wird. Anschließend erfolgt die Analyse weiterer passiver optischer Algorithmen, welche auf der Berechnung von sogenannten Disparitätenkarten unter Zuhilfenahme Stereo-optischer Systeme basiert. Diese berechneten Tiefenkarten enthalten die horizontale Verschiebung korrespondierender Bereiche in beiden Bildern. Die Kalkulation solcher ist generell sehr rechenaufwändig, liefert aber nach der Auswertung Informationen über die Entfernung sowie Position der abgebildeten Szene. Die dabei erhaltenen Ergebnisse hängen von dem jeweils verwendeten *Stereo-Matching* Algorithmus ab.

3.1 Aktiv optische Algorithmen

Aktiv optische Algorithmen basieren auf einer aktiven Beleuchtung der zu rekonstruierenden Szene. Dies geschieht indem beispielsweise Muster auf die Szene projiziert werden (Gitter, Streifen, Farben, etc.), mit deren Hilfe es möglich ist, das Korrespondenzproblem, also das finden korrespondierender Punkte in zwei Bildern, einfach und robust zu lösen. Uniforme bzw. einfache Muster führen zu Mehrdeutigkeiten, die leicht Fehlzuordnungen herbeiführen können. Ein alternativer Ansatz ist die Projektion zufälliger Muster, um etwaige falsche

Korrespondenzen durch bewusste Zufälligkeit zu vermeiden. Ein weiteres Feld in der Betrachtung aktiver visueller Techniken ist *Time of Flight* (TOF). Hierbei wird die Szene mit einem Lichtimpuls (meist infrarot) ausgeleuchtet und für jeden Pixel die Zeit gemessen, die jener benötigt, um wieder auf dem Sensor aufzutreffen. Damit ist es möglich, in geringer Auflösung dichte Tiefendaten für jedes Einzelbild zu messen.

Die von Lee et al. [Lee et al., 2012] vorgeschlagene Methodik der Hinderniserkennung bedient sich einer TOF Kamera. Die aufgenommenen Tiefenbilder werden basierend auf den Tiefenmessungen segmentiert. Dabei werden mit Hilfe eines Algorithmus zur Kantenerkennung jene herausgefiltert und anschliessend von der Tiefenkarte subtrahiert, um diese in weiteren Schritten als einzelne Objekte betrachten zu können. Anschließend erfolgt eine Tiefenanalyse der nun vorliegenden einzelnen Segmente. Um die gefundenen Segmente als Hindernis klassifizieren zu können wird zunächst die Standardabweichung der Disparität jedes Segments berechnet. Da die Tiefenwerte des Bodens einen hohen Grad der Streuung aufweisen ist die Standardabweichung dessen höher als bei anderen Segmenten. Diese Information dient zur Unterscheidung zwischen Hindernis und Boden. Dabei gelten Hindernisse als sich bewegende oder statische Objekte, welche sich innerhalb eines definierten Gefahrenbereichs befinden. Lee et al. definieren Ihre Gefahrenzone dabei mit 1 – 2 Metern. Die erkannten Hindernisse werden nun anhand der mittleren Distanz des Segmentes markiert.

Bei der Entwicklung eines autonomen Roboters zur Indoor-Überwachung verschiedener Areale bedienen sich Correa et al. [Correa et al., 2012] eines Microsoft Kinect Sensors zur Erstellung von Disparitätenkarten nach dem *Time of Flight* Prinzip. Dabei werden diese in mehrere horizontale Segmente eingeteilt. Die finale Karte besteht aus 5 Bereichen, von denen 3 als mögliche Richtungen betrachtet werden. Sobald die die minimalste Distanz eines Sektors geringer als 60 cm ist wird angenommen, dass sich das System vor einem Hindernis befindet. Ausgehend von der Menge an Sektoren mit gefundenen Hindernissen wird die neue Bewegungsrichtung angepasst.

3.2 Passiv optische Algorithmen

Passiv optische Algorithmen beziehen sich auf die Berechnung dreidimensionaler Informationen aus Bildern ohne eigene Lichtquelle. Dabei sind viele dieser Methoden an das menschliche oder auch tierische Sehen angelehnt. Viele Ansätze nutzen das *Stereo Vision* Prinzip welches durch die Differenz zweier Bilder derselben Szene einen Eindruck von Tiefe verschafft. Weiterhin kann

eine dreidimensionale Rekonstruktion der Umgebung durch die Analyse von Bildern hinsichtlich Lichteinflüssen, Schattierungen oder durch Veränderung des Kamerafokus vorgenommen werden.

Bei der Entwicklung eines autonomen Roboters auf Bodenebene werden nach Kostavelis et al, [Kostavelis et al., 2010] die errechneten Disparitätenkarten in 3 gleich große horizontale Bildbereiche aufgeteilt, welche die möglichen Bewegungsrichtungen des Systems repräsentieren. Für jedes dieser einzelnen Unterbilder wird nun der Durchschnitt der Disparitäten berechnet, wobei der Bildteil mit dem geringsten Wert auf Hindernisse hinweist, welche sich näher am System befinden. Angesichts der Tatsache, dass die Entscheidung darüber, welcher Weg als der sicherste angesehen werden muß, teilweise schwer zu treffen ist, wurde die sogenannte *threshold estimation* Methode entwickelt. Die Unterteilung der Bilder wird hier ebenfalls in drei Regionen vorgenommen. Zunächst werden alle Pixel deren Werte sich über einem vordefinierten Schwellwert befinden markiert und gezählt. Wenn die Anzahl der als Hindernis definierten Pixel über einem ebenfalls vordefinierten Prozentsatz liegen, so wird ein Hindernis als gefunden markiert (in der jeweiligen Region). Die letzte vorgestellte Methode von Kostavelis et al. orientiert sich in ihrer Funktionsweise an der soeben beschriebenen Schwellwertschätzung und erweitert den Algorithmus um die Betrachtung aller 3 Bildteile. Bei der *multi threshold estimation* wird jedes Drittel des Bildes betrachtet, wobei in allen Regionen nach Pixeln innerhalb des Schwellwerts gesucht und diese markiert werden. Anschließend werden die Ergebnisse untereinander verglichen, und das Drittel mit dem geringsten Wert als Hindernis ausgewählt. Sollten die prozentualen Werte aller Drittel größer sein als die gegebene Grenze so wird angenommen das sich das Hindernis sehr nah vor dem System befindet.

Eine weitere Methode zur Hindernisvermeidung für UAS wurde von Richards et al. [Richards et al., 2014] vorgestellt. Optischer Fluss sowie *Feature Tracking* bieten dabei die Grundlage für die Erkennung, Vermeidung und Voraussage, welcher Bereich im Raum der sicherste ist. Dabei gehen die beiden Ausgangstechniken jeweils einer anderen Aufgabe nach. Der Optische Fluss dient zum Erkennen und Verfolgen von Objekten sowie für die Voraussage der Position im nächsten Einzelbild, wohingegen das *Feature Tracking* [Shi and Tomasi, 1994] für die Erkennung von markanten Punkten innerhalb des Objektes genutzt wird. Bei diesem wird in jedem folgenden Einzelbild verglichen, ob bereits bekannte Punkte innerhalb einer bestimmten Distanz mit denen des aktuellen Bildes übereinstimmen. Zur Schätzung zu erwartenden Position der Merkmale im darauffolgenden Bild werden mit Hilfe des Optischen Flusses die Verschiebungsvektoren zwischen beiden Bildern berechnet. Aufgrund dessen, dass der

zugrunde liegende Algorithmus von Lukas-Kanade [Lucas et al., 1981] nur bei kleinen Verschiebungen valide Ergebnisse liefert, wird ein pyramidaler Ansatz [Bouguet, 2001] für das Matching verwendet, bei welchem beide Bilder eines *frames* herunter skaliert werden. Durch die Verbindung dieser beiden Techniken lassen sich Objekte erkennen und verfolgen. Zur Bestimmung der nächstbesten Position für das UAV wird ausgehend von den berechneten Resultaten (gefundene und erkannte Hindernisse) eine stochastische Matrix erstellt welche zur weiteren Planung des Fluges verwendet wird.

Bei der Entwicklung von Algorithmen, welche auf räumlicher Tiefe basieren, bieten stereo-optische Systeme einen großen Vorteil. Durch den Versatz der Kameras auf der Basislinie wird die Szene aus zwei minimal verschiedenen Blickwinkeln aufgenommen. Dies ermöglicht die Berechnung dreidimensionaler Informationen einerseits mithilfe verschiedenster *Feature Tracking* Methoden sowie anschließender Triangulierung oder andererseits unter Zuhilfenahme diverser Algorithmen zur Lösung des Stereo-Korrespondez Problemes. Trotz dessen ist die Benutzung zweier Kameras nicht immer möglich, sei es durch die Limitierung durch das System selber aus Platzgründen oder, im Falle unbemannter Flugsysteme, durch die begrenzte maximale Traglast. Aufgrund letzterer entschieden sich Mori und Scherer [Mori and Scherer, 2013] für die Nutzung eines monokularen Setups. Durch die Verwendung des optischen Flusses ist es auch mit nur einer Kamera möglich, Hindernisse zu erkennen, jedoch fehlt die eigentliche Erkennung von Tiefe. Sofern sich ein Objekt direkt auf das System zu bewegt, kann es nur schwer erkannt werden, da kaum perspektivische Veränderungen beobachtbar sind. Um diese wahrzunehmen, muss der Algorithmus dazu in der Lage sein, die Veränderung der relativen Größe eines Objektes in aufeinander folgenden Bildern abzuschätzen. Mori und Scherer setzten bei der Erkennung von Merkmalen auf den SURF Algorithmus [Bay et al., 2006], welcher gefundene Features auch nach Änderung der Größe wieder erkennen kann (Skaleninvarianz). Im Anschluss daran wird die Veränderung der Größe der benachbarten Umgebung untersucht, um eine Aussage darüber treffen zu können, ob sich das Hindernis auf das System zu bewegt. Jene Features welche nicht skaliert wurden, werden nicht weiter betrachtet. In jedem folgenden Einzelbild wird nun mit Hilfe von *Template Matching*¹ verglichen, wie sich die Skalierung der Umgebung eines Features im Vergleich zum vorherigen Frame verändert hat. Sollte die Distanz eines Features zu nah am System sein, so wird dieses als potentielles Hindernis für die Vermeidung verwendet.

¹ *Template Matching* beschreibt ein Verfahren der digitalen Bildverarbeitung bei dem innerhalb eines Bildes nach einer Vorlage gesucht wird. Dabei ist diese Vorlage entweder eine einzelne Form, oder ein anderes Bild.

Kapitel 4

Entwickelte Hinderniserkennungen

Im Rahmen dieser Arbeit wurden zwei Ansätze zur Erkennung von Hindernissen in Echtzeit entwickelt. Ein Hindernis ist im Rahmen dieser Arbeit als Objekt definiert, welches sich vor dem UAS innerhalb einer variablen Gefahrenzone befindet. Beide Methoden richten sich nach den in Kapitel 2 erläuterten Algorithmen und Konzepten. Anhand dieser ist es möglich aus den beiden Bildern des Stereo Systems für jeden Frame die Disparitätenkarte zu berechnen, welche im Anschluss daran auf zwei verschiedenen Wegen für die Hinderniserkennung ausgewertet werden.

Im folgenden Kapitel werden beide Methoden detailliert beleuchtet. Zu Beginn wird die zugrunde liegende Klassenstruktur beschrieben. In Abschnitt 4.2 die *Subimage Detection* erläutert, wobei auf das grundlegende Konzept sowie den Algorithmus zur Erkennung selber eingegangen wird. Selbiges gilt für die *Samplepoint Detection* in Abschnitt 4.3.

4.1 Grundlegende Operationen

Der Ablauf der Hinderniserkennung ist in beiden entwickelten Methoden gleich. Die dafür benötigen *Preprocessing* Schritte gleichen sich demnach ebenfalls und werden im folgenden erläutert.

ROI der Disparitätenkarte:

Zu Beginn des Algorithmus muss der durch die Translation der Bilder entlang der Basislinie verursachte nicht matchbare Bereich entfernt werden. Dieser entsteht da die Stereobilder aufgrund der Translation entlang der Basislinie nicht exakt dieselben Bereiche der Szene abdecken. Dies geschieht nach der Wahl der

in 2.3.2.1 beschriebenen Parameter der SGBM. Dieser sogenannte *Pixelshift* berechnet sich aus der Anzahl der zu berechnenden Disparitäten (*numDisparities*). Sollte dieser einen ungeraden Wert ergeben so wird er um 1 erhöht. Generell ist der *Pixelshift* so gewählt das die Dimensionen der Disparitätenkarte durch 8 dividierbar sind. Dieser Wert hat sich während der Entwicklung als robuster Wert erwiesen.

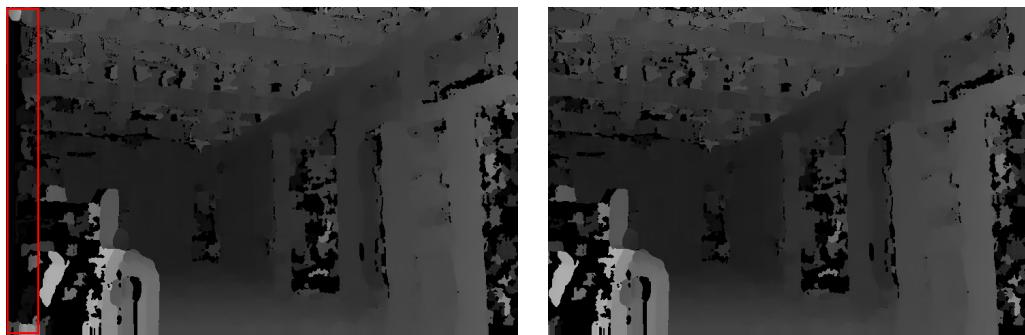


Abbildung 9: Darstellung der Disparity ROI. Der rote Bereich markiert den *Pixelshift*.

Abbildung 9 zeigt den *Pixelshift* in der finalen Disparitätenkarte. Der *SGBM* berechnet zwar Werte innerhalb des Bereiches in dem nur im linken Bild Informationen vorliegen, jedoch geschieht dies in Abhängigkeit der aktuellen Szene. Ist es nicht möglich in diesem Informationen zu berechnen so werden die Disparitäten entweder „geraten“ oder als schwarzer Bereich mit negativen Werten ausgedrückt. Aus Gründen der Performance wird dieser daher entfernt um keine Berechnungen an informationslosen Pixeln durchzuführen.

Berechnung der Disparität aus gegebener Distanz:

Um die Algorithmen ressourcensparend zu gestalten wird bei der Erkennung mit den reinen Disparitäten gearbeitet. Da die Disparitäten in Abhängigkeit der Bildgröße sowie der Q-Matrix berechnet werden, kann im Voraus kein fest definierter Disparitätenwert für eine Distanz bestimmt werden. Für diesen Prozess wird die in Anschnitt 2.4 beschriebene Distanzberechnung invertiert. Mithilfe der in Formel 4.1 dargestellten Berechnung können nun die jeweilige Position sowie die dazugehörige Disparität berechnet werden.

$$\begin{aligned} d &= \frac{f - Z' \cdot b}{Z' \cdot a} \\ I_x &= X' \cdot (d \cdot a + b) + C_x \\ I_y &= Y' \cdot (d \cdot a + b) + C_y \end{aligned} \tag{4.1}$$

Dieser Prozess findet in der späteren Initialisierung der Methoden Verwendung bei welcher die metrischen Angaben der minimalen sowie maximalen Erkennungsreichweite in Disparitäten zurückgerechnet werden. Als Referenz wird dazu die 3D-Koordinate des Bildhauptpunktes berechnet. Diese Vorgehensweise spart in der späteren Hinderniserkennung Ressourcen, da nicht für jedes *Subimage* bzw. jeden *Samplepoint* die Weltkoordinate berechnet werden muss. Stattdessen können die Disparitäten direkt verglichen werden.

4.2 Subimage Detection

Das grundlegende Funktionsprinzip der *Subimage Detection* ist grob an die von Kostavelis et al. vorgestellten Algorithmen angelehnt. Für die Analyse einer Disparitätenkarte wird diese in Segmente unterteilt. In der von Kostavelis et al. vorgeschlagenen Methode erfolgt die Segmentierung in drei gleich große horizontale Bereiche. Diese Aufteilung erweist sich für UAS-Anwendungen als nicht brauchbar, da nur 3 Bewegungsrichtungen betrachtet werden. Aus diesem Grund wird die folgende Segmentierung vorgeschlagen:

Für die Unterteilung der Disparitätenkarte wurden initial nur 9 Segmente vorgesehen wobei jeder Bereich eine der möglichen Flugrichtungen repräsentieren sollte. Aufgrund der Größe der Submatrizen konnte jedoch keine valide Erkennung kleiner Hindernisse gewährleistet werden, da solche in der Berechnung des Mittelwertes untergegangen sind. Aufgrund dessen wurde die Anzahl der Segmente auf 81 erhöht, welches eine wesentlich genauere Erkennung ermöglicht. Zudem ist somit eine weitaus genauere Einteilung der Flugrichtungen möglich, da jede dieser genauer betrachtet wird (siehe Abbildung 10).

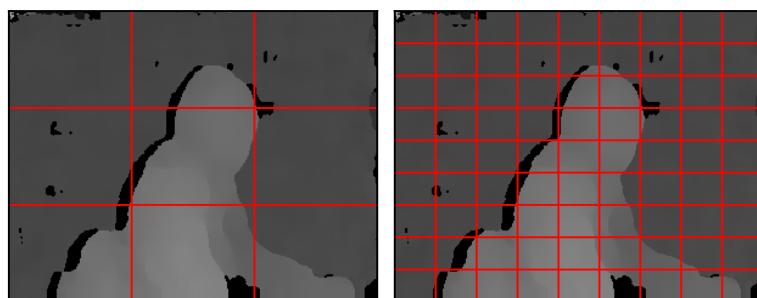


Abbildung 10: Intiale Segmentierung der Disparitätenkarte sowie die weitere Unterteilung zur genaueren Bestimmung der Werte der *Subimages*

Eingangsdaten für die *Subimage Detection* sind die vorprozessierten Disparitätenkarten, die im ersten Schritt segmentiert werden. Dabei wird für jedes Segment eine eigene Objektinstanz der Klasse *Subimage* erzeugt. Diese repräsentiert ein einzelnes Bildsegment, und hält daher die eigene Position im Bild (obere linke und untere rechte Ecke des definierten Rechtecks), den Mittelwert des Segments, sowie den Mittelpunkt des Rechtecks zur späteren *Pointcloud* Generierung. Dies wird in Abbildung 11 dargestellt.

Subimage
+tl: Point
+br: Point
+center: Point
+value: float
+Subimage(in tl:Point,in br:Point): Subimage
+calcSubimageValue(in disparity_map:Mat): void
+draw(inout matrix:Mat): void

Abbildung 11: *Subimage* Klasse

Um die Hindernisse innerhalb der Segmente zu erkennen wurde auf die Berechnung des Mittelwertes dieser gesetzt. Einerseits, da die Berechnung des Mittelwertes unter Betrachtung des Echtzeit-Aspektes eine ressourcensparende und schnelle Operation ist, andererseits weil jeder Pixel des Bildes Einfluss auf das Ergebnis hat wobei der Einfluss von Ausreißern verringert wird. Jedoch musste die Berechnung auf das Szenario angepasst werden. Testergebnisse ergaben das die Berechnung des Mittelwertes aller Werte zu Verzerrungen geführt hätte. Bei der Kalkulation von Disparitätenkarten werden Bereiche welche nicht gematcht werden können, sei es aufgrund von Schatten- oder Überdeckungseffekten in den Referenzbildern oder homogener Texturen in der Szene, als negativer Wert ausgedrückt. Berechnet man nun den Mittelwert unter Betrachtung positiver und negativer Werte, so wird das in diesem Anwendungsbereich erwünschtes Ergebnis verfälscht. Im schlechtesten Fall enthält eine Submatrix mehr negative als positive Werte, was dazu führen kann das Hindernisse vor homogenen Flächen nicht erkannt werden. Daraus ergibt sich die in Algorithmus 1 dargestellte Berechnung des Mittelwertes.

Es werden demnach nur positive Disparitäten betrachtet was auch dazu führt das sich der Gesamtwert aller Werte aus der Menge dieser ergibt. Mit Hilfe dieser Berechnung ist es möglich Hindernisse auch in Bereichen zu erkennen in denen die Mehrzahl der Pixel keine Matches aufweisen.

Algorithm 1 Berechnung des Mittelwertes der Disparitäten

```

1: procedure CALCMEANDISPARITY(submatrix)
2:   elementsnumber  $\leftarrow 0$ 
3:   elementssum  $\leftarrow 0$ 
4:   for value in submatrix do
5:     if value  $> 0$  then
6:       elementssum  $\leftarrow \text{elements}_{\text{sum}} + \text{value}$ 
7:       elementsnumber  $\leftarrow \text{elements}_{\text{number}} + 1$ 
8:     end if
9:   end for
10:  return elementssum/elementsnumber
11: end procedure

```

Die eigentliche Hinderniserkennung erfolgt in jedem Frame. Wobei der Term Frame hierbei nicht ein durch die Kameras aufgenommenes Stereobildpaar, sondern eine daraus berechnete Disparitätenkarte definiert. Da die Bildgröße pro Einzelbild stetig ist besteht keine Notwendigkeit die jeweiligen Segmentobjekte erneut zu generieren. Es genügt also die Mittelwerte eines jeden *Subimages* anhand der neuen Disparitätenkarte zu aktualisieren. Daraufhin wird geprüft ob sich einer oder mehrere der *Subimage* Mittelwerte innerhalb der Gefahrenzone befinden. Die Gefahrenzone wird zur Initialisierung der Erkennung nach der in Abschnitt 4.1 beschriebenen Invertierung der Distanzberechnung berechnet. Diese Rückrechnung ist ausschlaggebend für die schnelle Erkennung der Tiefe, da somit nicht für jeden Pixel einer *Subimage* Instanz die Distanz berechnet werden muss.

Algorithmus 2 beschreibt die Berechnung der *Pointcloud* jedes einzelnen Frames. Mithilfe des Mittelpunktes der *Subimages* wird für jedes dieser die jeweilige 3D-Koordinate (siehe Abschnitt 2.4) berechnet und in die finale Punktwolke geschrieben.

Innerhalb der *Pointcloud* sind nun die Weltkoordinaten jedes *Samplepoints* relativ zur Kameraposition gespeichert. Der Vektor zwischen dem Koordinatenursprung und der Koordinate entspricht dabei der Distanz zum Hindernis. Die Ausgabe der *Pointclouds* erfolgt in Stanfords *Polygon File Format (ply)*. Aufgrund der einfachen Struktur können die einzelnen Punktwolken erkannter Hindernisse einfach analysiert werden um etwaige Vermeidungsstrategien zu entwickeln. Weiterhin ermöglicht das *ply* Format die Visualisierung der 3D-Punkte in jedem Frame in dem Hindernisse erkannt wurden.

Algorithm 2 Ablauf der Hinderniserkennung

```

1: procedure DETECTOBSTACLES(submatrix)
2:   found_points  $\leftarrow 0$ 
3:   found_obstacles  $\leftarrow 0$ 
4:   for value in mean_disparities do
5:     if value  $< range_{min}$  AND value  $> range_{max}$  then
6:       temp_Subimage  $\leftarrow Subimage\_vector[\text{value}]$ 
7:       push temp_Subimage to found_obstacles
8:       calculate coordinate for temp_Subimage
9:       push coordinate to found_points
10:      end if
11:    end for
12:    if size of found_points  $\neq 0$  then
13:      write pointcloud for current found_obstacle container
14:    end if
15:  end procedure

```

4.3 Samplepoint Detection

In Anlehnung an aktive optische Algorithmen zur 3D-Rekonstruktion und damit verbundenen Techniken wie *Laser Scanning* oder *Time of Flight* wurde die *Samplepoint Detection* entwickelt. Das Ziel war dabei eine ressourcensparende Alternative zur *Subimage Detection* zu schaffen indem nicht zwangsläufig alle Pixel der Disparitätenkarte betrachtet werden müssen.

Ebenso wie in der in Abschnitt 4.2 vorgestellten *Subimage Detection* bedient sich die *Samplepoint Detection* einer vorverarbeiteten Disparitätenkarte zur Hinderniserkennung. Bei der Initialisierung wird die Anzahl der zu berechnenden *Samplepoints* festgelegt. Dabei richtet sich der Wert nach der Anzahl der Reihen und Spalten der Matrix der Disparitätenkarte um eine äquidistante Verteilung gewährleisten zu können. Der hier verwendete Standard-Abstand zwischen den einzelnen *Samplepoints* wurde auf 8 Pixel definiert. Die Anzahl dieser ist demnach relativ zur verwendeten Bildgröße. Auf eine Verteilung der Messpunkte am direkten Rand der Disparitätenkarte wurde bewusst verzichtet, da dieser aufgrund der Rektifizierung eher Fehler aufweist als Bereiche in der Mitte des Bildes.

Ein *Samplepoint* (siehe Abbildung 12) umfasst im entwickelten System entweder einen Pixel oder einen Zusammenschluss der lokalen Nachbarschaft. Es werden somit in Abhängigkeit der gewählten Größe der lokalen Nachbarschaft

Samplepoint
+center: Point
+radius: int
+roi: Rect
+value: float
+Samplepoint(in center:Point,in radius:int): Samplepoint
+calcSamplepointValue(in disparity_map:Mat): void
+draw(inout matrix:Mat): void

Abbildung 12: Samplepoint Klasse

um die *Samplepoints* Segmente gebildet. Da ein Pixel gerade bei Bildern mit hoher Auflösung weniger Aussagekraft besitzt als der Pixel inklusive der lokalen Umgebung wurde ein *Samplepoint* auf diese Weise definiert. Der Wert eines einzelnen Messpunktes Sp mit dem Zentrum C ergibt sich im Falle eines Radius $r = 3$ aus dem Mittelwert des Segmentes S mit den Eckpunkten $S_{tl} = (C_x - r, C_y - r)$ und $S_{br} = (C_x + r, C_y + r)$. Dies erhöht zwar die Anzahl der zu betrachtenden Pixel welche jedoch, in Abhängigkeit vom gewählten Radius, geringer ist als die Gesamtzahl aller Bildpunkte. Ist der Radius als 0 definiert so wird lediglich der Pixel des Mittelpunktes betrachtet. Dieser Prozess wird in Abbildung 13 verdeutlicht.

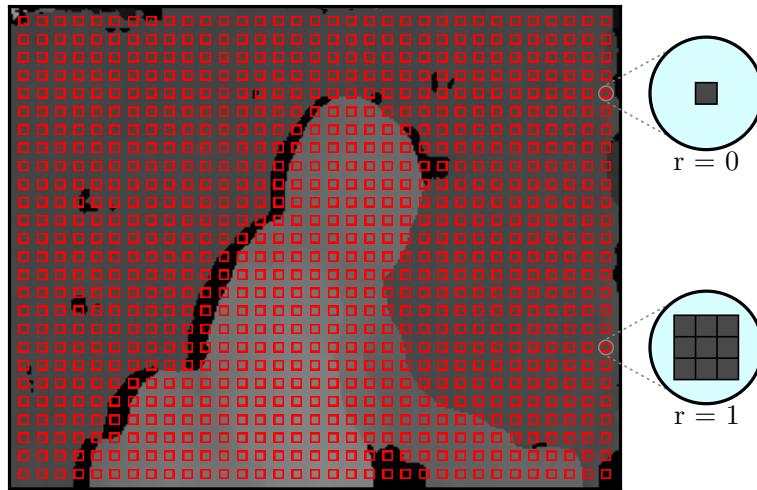


Abbildung 13: Darstellung der Samplepoints mit verschiedenen Radien

Nachdem zunächst alle *Samplepoints* initialisiert wurden, werden die jeweiligen Mittelwerte, nach der in 4.2 beschriebenen Mittelwertsberechnung, pro Frame aktualisiert. Die eigentliche Erkennung erfolgt dabei ebenfalls durch die Erstellung einer *Pointcloud*. Es wird für jedes *Subimage* pro Einzelbild überprüft ob

sich die aktualisierten Werte innerhalb des Gefahrenbereichs befinden. Ist dies der Fall wird für jede Disparität eines *Samplepoints* die jeweiligen dreidimensionale Koordinate berechnet und anschließend in die Punktwolke geschrieben.

4.4 Applikationsstruktur

Die grundlegende Struktur beider im Rahmen dieser Arbeit entwickelten Methoden ist dieselbe. Die abstrakte Klasse *ObstacleDetection* (siehe Abbildung 14 vererbt alle wesentlichen, für die Erkennung auf Basis von Einzelbildern benötigten Methoden an je eine abgeleitete Klasse, welche eine Methode der Hinderniserkennung repräsentiert. Zu diesen zählen die Funktionen *update* und *detectObstacles*. Die Ausführung beider in jedem Frame ist essenziell notwendig um die benötigten Daten innerhalb jedes Einzelbildes vorliegen zu haben. Dabei dient die *update* Funktion zur Aktualisierung der zugrundeliegenden Teilbereiche der Disparitätenkarte. Anschliessend wird in der Funktion *detectObstacles* für jeden Disparitätenwert überprüft ob sich dieser innerhalb der definierten Gefahrenzone befindet. Zusätzlich zu den Funktionen der Basisklasse besitzt jede Methode eine Funktion zur Initialisierung dieser. Bei dieser wird die eigentliche Struktur der *Subimages* und *Samplepoints* erstellt, sowie die aktuelle Gefahrenzone definiert. Im Fall der *Samplepoint Detection* wird ausserdem der Radius um einen *Samplepoint* in Pixeln festgelegt.

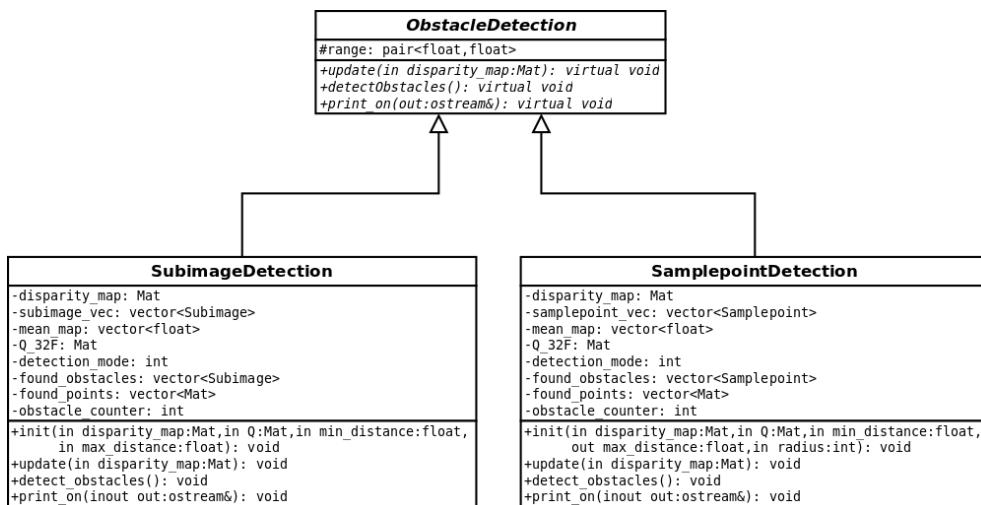


Abbildung 14: Klassenstruktur der Hinderniserkennung

Kapitel 5

Evaluation

Um die Funktionsweise beider Methoden zu belegen wurden diese sowohl auf ihre Robustheit als auch auf ihre Performance getestet. In diesem Kapitel wird zunächst das zum Testen verwendete Setup erläutert. Weiterhin werden die einzelnen durchgeführten Tests sowie die erlangten Ergebnisse beschrieben. Zuletzt erfolgt eine Diskussion der erlangten Werte sowie eine Gegenüberstellung beider Algorithmen.

5.1 Testsetup

Bei der Durchführung der Tests befinden sich die Kameras statisch im Raum. Um sicher zu stellen, dass sich die Position der Kameras während der einzelnen Versuche nicht verändert wurden diese auf einer Eisenplatte mittels eines Laborstativs magnetisch befestigt (siehe Abbildung 15) Die potentiellen Testhindernisse werden innerhalb der zu erkennenden Reichweite platziert, wobei die Ausrichtung der Hindernisse teils zufällig, teils bewusst in kritischen Positionen erfolgt um ein reales Anwendungsszenario zu simulieren. Ein aufgenommenes Testset besteht dabei aus einem Stereobildpaar, der Disparitätenkarte, einer kompletten Punktewolke dieser um etwaige Fehler der Algorithmen leichter erkennen zu können, sowie den einzelnen erkannten Hindernispunktewolken. Weiterhin werden diverse Parameter gespeichert, wie die Anzahl der erkannten Hinderniselemente sowie deren Disparitäten.

Der zu erkennende Bereich wurde auf 0,2 bis 1,5 Meter definiert. Dies entspricht einem Szenario in dem das System auch aufgrund der hohen Framerate der Erkennung angewendet werden kann. Die maximale Geschwindigkeit des Pelican beträgt $16m/s$. Bei einer Framerate von 40 Bildern pro Sekunde ergibt sich eine zurückgelegte Distanz von $16/40 = 0.4m$ je Einzelbild. Da das System für die Verwendung in Innenräumen konzipiert ist, ist auch die maximale Ge-

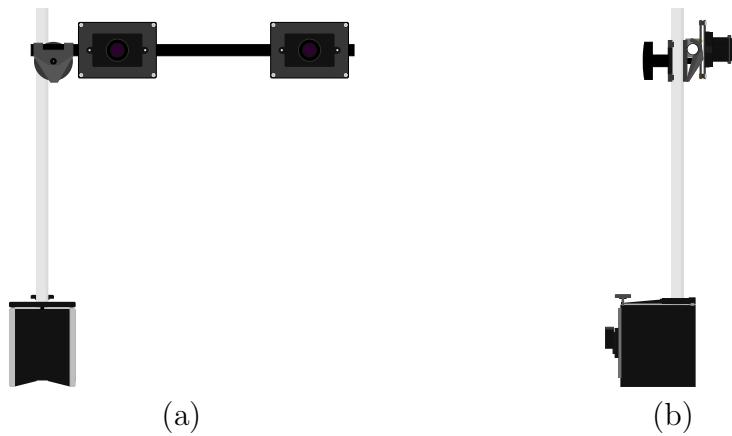


Abbildung 15: Ansicht des Kamera frontal (a) und von der Seite (b).

schwindigkeit deutlich geringer. Bei einer angenommenen Geschwindigkeit von $4m/s$ ergibt sich ein Einzelbild alle 10 cm. Dies reicht für etwaige Manöver zur Vermeidung eines Hindernis aus. Eine Erweiterung dessen auf beispielsweise 2.0 Meter wurde im Rahmen der Tests nicht durchgeführt, da in vorhergehenden Experimenten bereits festgestellt wurde, dass mit dem vorliegenden Setup robust Disparitätenkarten und daraus entsprechend gute Distanzwerte berechnet werden können [Al-Hallak and Hiller, 2015]. Das nach der Anwendung der ROI auf die Disparitätenkarte verbleibende Sichtfeld (siehe 4.1) beträgt 50° auf horizontaler Achse und 38° vertikal. Dabei ist dieses in Abhängigkeit der verwendeten Parameter breiter oder schmäler. Dies ist in Abbildung 16 visualisiert.

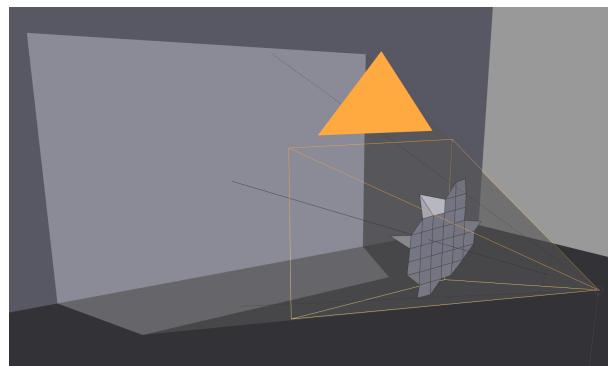


Abbildung 16: Im Bild ist das nach der Rektifizierung sowie Beschneidung der ROI erhaltene Sichtfeld visualisiert. Das Mesh innerhalb des Kamera Frustums repräsentiert die gerenderte Pointcloud eines Hindernisses

Ein aufgenommenes Testset besteht aus jeweils 12 Testbildern. Für jede Me-

thode wurden drei verschiedene Hindernisgrößen getestet, groß, mittel und kleine Hindernisse (dargestellt in Abbildung 17). Anhand dieser wird ausgewertet welche minimale, maximale sowie mittlere Disparität, und daraus resultierende Distanz erkannt wird. Um eine bessere Vorstellung der Größe der verwendeten Hindernisse zu erhalten werden die Dimensionen dieser in Tabelle 5.1 aufgeführt. Ein großer Abstand zwischen Hindernis 2 und 3 ist dabei gewünscht um die Erkennung sehr kleiner Hindernisse zu untersuchen.



Abbildung 17: Verwendete Testhindernisse, die Kantenlänge eines Quadrates des linken Hindernis beträgt 5.5 cm

Hindernis	Radius (cm)	Länge (cm)	Breite (cm)	Fläche (cm ²)
1 (Groß)	-	53.5	43.0	2300.5
2 (Mittel)	-	16.0	14.5	232.0
3 (Klein)	2.5	-	-	19.6

Tabelle 5.1: Maße der verschiedenen genutzten Hindernisse

Weitere Tests beinhalten die Erkennung kleiner Hindernisse unter Veränderung der SGBM Parameter. Dabei wird unter anderem untersucht ob beispielsweise eine verringerte Blockgröße Einfluss auf die Erkennung kleiner Bereiche nimmt. Da die *Samplepoint Detection* den Radius als modifizierbaren Parameter mit sich bringt, wird auch dahingehend untersucht inwiefern die Wahl eines anderen Radius die Ergebnisse der Erkennung beeinflusst. Zudem wird die Funktionsfähigkeit der Methoden bei reflektierenden sowie durchsichtigen Flächen untersucht.

Die aufgenommenen Disparitätenkarten sind in der Darstellung aufgrund der Normalisierung und teilweise sehr hohen Disparitäten (Ausreißer) oft sehr dunkel. Deshalb wurden für Zwecke der Visualisierung der Kontrast einzelner Dis-

paritätenkarten angepasst/optimiert. Weitere Modifikationen der Bilder wurden nicht vorgenommen.

5.2 Durchführung und Ergebnisse

5.2.1 Robustheit

Hinsichtlich der Robustheit werden beide Algorithmen nach demselben Schema untersucht. Während der Tests wurde für jedes aufgenommene Einzelbild die reale Distanz gemessen. Der dabei verwendete Referenzpunkt befand sich in der Mitte des zu erkennenden Hindernisses, da diese nicht zwangsläufig orthogonal zur Bildebene platziert wurden. Die berechnete Distanz entspricht der mittleren Distanz welche sich aus allen erkannten Bereichen eines Bildes zusammensetzt.

Während der Aufnahme sind die jeweiligen Einzelbilder der linken und rechten Kamera sowie die berechnete Disparitätenkarte sichtbar. Zudem ist ein weiterer Anzeigemodus verfügbar welcher die gefundenen Hindernisse innerhalb der Tiefenkarte markiert (siehe Abbildung 18). Weiterhin kann jede erstellte Hindernis-Pointcloud im Nachhinein gerendert werden um einen Überblick über die Ergebnisse des Algorithmus zu erhalten.

5.2.1.1 *Subimage Detection*

Großes Hindernis:

Im ersten Test wurde Hindernis 1 gewählt, dabei wurde mithilfe der dargestellten Hindernisse auf der Disparitätenkarte visualisiert ob sich die Testhindernisse innerhalb des Gefahrenbereichs befinden. Um zu testen, inwieweit sich die Hinderniserkennung auf den tatsächlich definierten Bereich vor dem Kamerasystem beschränkt, wurde das Hindernis zudem zu Teilen außerhalb dieser platziert. Aus den 12 aufgenommenen Bildern des Testsets ergaben sich die in Abbildung 19 (a) sichtbaren Ergebnisse.

Wie aus diesen zu erkennen wurde in nahezu allen Frames das Hindernis richtig erkannt. Die berechneten Distanzen stimmen mit den real gemessenen überein. Minimale Abweichungen im Bereich von 2-6 Zentimetern können in diesem Fall als Messungenauigkeit ignoriert werden, da der Mittelpunkt zwischen der minimalen und maximalen Objektdistanz nicht genau bestimmbar war und die Streckenmessung mittels Laser-Entfernungsmesser oder Maßband aufgrund der fehlenden Information über die genaue Lage des Projektionszentrums der Referenzkamera nicht genauer als die mittlere Abweichung ist. Die durchschnittli-

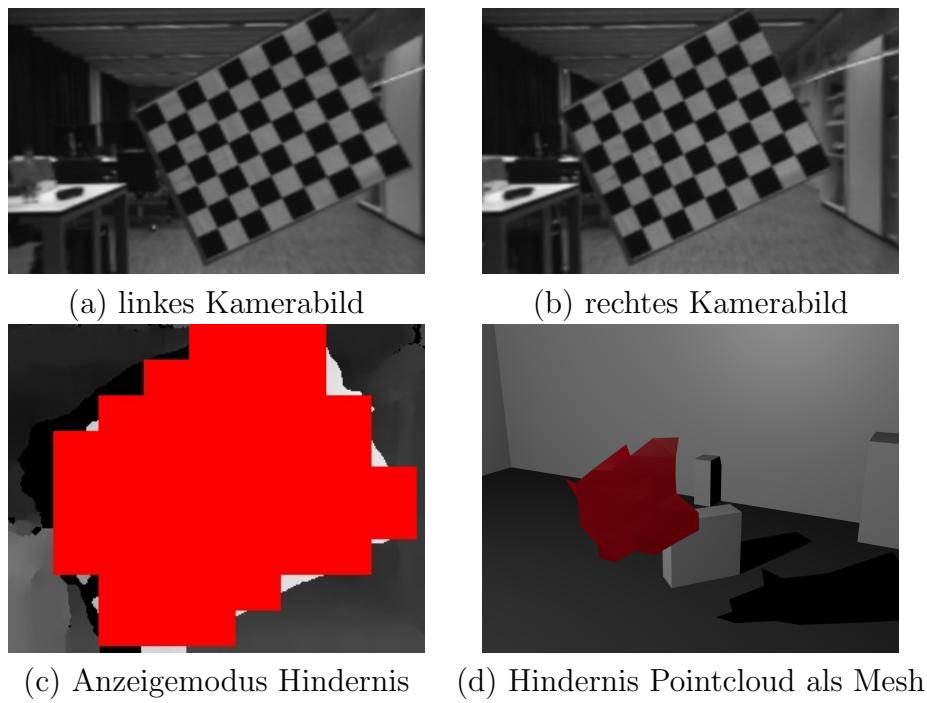


Abbildung 18: Abbildungen (a) - (c) zeigen die sichtbaren Bilder während der Testaufnahme. (d) als Mesh gerenderte Pointcloud

che Abweichung von berechneten zu real gemessenen Werten betrug dabei 0,02 Zentimeter. Die Standardabweichung der Differenzen beträgt 4,0 Zentimeter. Dies deutet auf eine allgemein robuste Erkennung von Hindernis 1 durch die *Subimage Detection* hin.

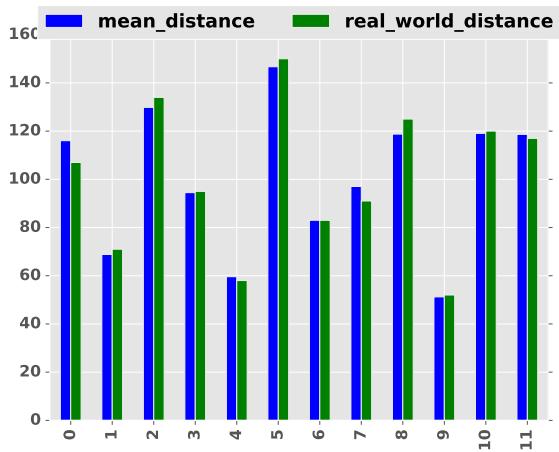


Abbildung 19: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 1 der *Subimage Detection*

Mittleres Hindernis:

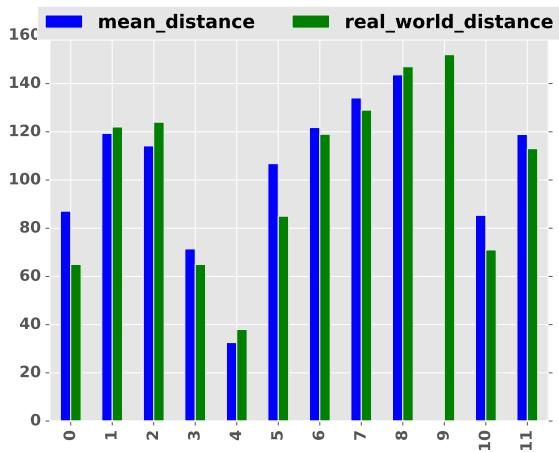


Abbildung 20: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 2 der *Subimage Detection*.

Auch die Erkennung mittlerer Hindernisse durch die Methode war in nahezu allen Fällen erfolgreich. Ein auffälliges Detail im Vergleich zu den Ergebnissen des großen Hindernisses ist die höhere Differenz aus berechneter und gemessener Distanz. Auch die durchschnittliche Abweichung von 7,9 Zentimetern deutet auf eine schlechtere Erkennung der Hindernisse hin. Selbiges lässt sich aus der hohen Standardabweichung von 44,5 Zentimetern erkennen. Das gete-

stete Objekt befand sich während der Tests an einem durchsichtigen Plexiglas Stab welcher teilweise auch als Hindernis erkannt wurde.

Kleines Hindernis:

Die Erkennung von Hindernis 3 durch die *Subimage Detection* erwies sich als am schlechtesten. In vier von 12 durchgeführten Versuchen wurde das Hindernis nicht mehr erkannt. Auch die restlichen erkannten Hindernisse weisen jeweils eine sehr hohe Differenz zwischen berechneter und gemessener Distanz auf (siehe Abbildung 21). Betrachtet man die durchschnittliche Abweichung beider gemessener Distanzen beträgt diese 2,0 cm. Dies deutet zunächst auf keine besonders schlechte Erkennung hin. Die Standardabweichung von 63,9 Zentimetern zeigt jedoch die hohe Fehlerrate der Methode bei Hindernis 3.

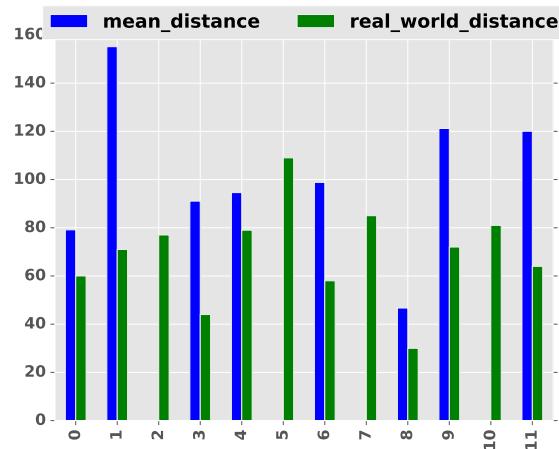


Abbildung 21: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 3 der *Subimage Detection*.

5.2.1.2 *Samplepoint Detection*

Im Rahmen dieses Abschnittes wird die *Samplepoint Detection* getestet. Durch die wesentlich höhere Anzahl an betrachteten Bereichen innerhalb der Disparitätenkarte ist anzunehmen, dass die Erkennung kleiner Hindernisse mit diesem Verfahren eine höhere Erfolgswahrscheinlichkeit verspricht. Dies geschieht nach dem in 5.2.1.1 bereits beschriebenen Schema. Zur Initialisierung der Methode wurde der Radius der *Samplepoints* auf 2 Pixel gesetzt, was einer Gesamtanzahl von 25 Pixeln pro *Samplepoint* entspricht.

Großes Hindernis:

Die Erkennung großer Objekte der *Samplepoint Detection* weist nahezu keine Fehler auf. Abbildung 22 zeigt das jede der 12 Ausrichtungen des Objektes erkannt wurden. Die dabei auftretenden Differenzen zwischen der berechneten sowie gemessenen Distanz sind auf Messungenauigkeiten, durch die Verwendung des Maßbandes sowie Laser-Entfernungsmessers, zurückzuführen. Auch in Grenzbereichen nahe dem Rand des Gefahrenbereichs wurden die Hindernisse erkannt. Mittlere Differenz sowie die Standardabweichung zwischen der gemessenen sowie berechneten Distanz liegen ebenfalls bei einem Wert von 2,5 beziehungsweise 2,7 Zentimetern.

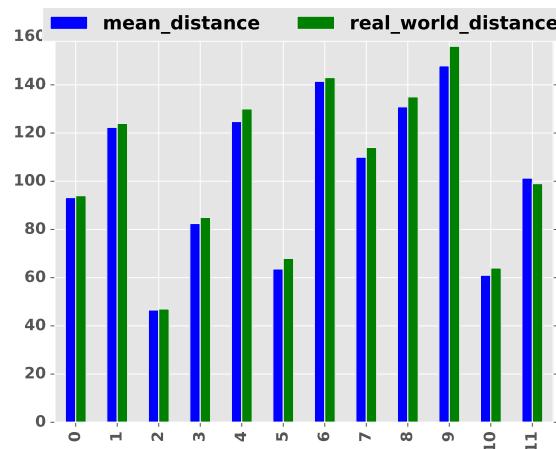


Abbildung 22: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 1 der *Samplepoint Detection*.

Mittleres Hindernis:

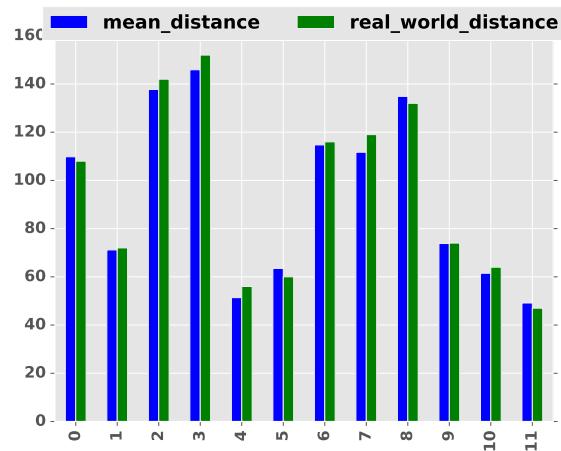


Abbildung 23: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 2 der *Samplepoint Detection*.

Auch die Erkennung mittlerer Hindernisse erfolgte in allen Frames ohne signifikante Probleme (siehe Abbildung 23). Standardabweichung sowie Mittelwert der Differenzen (1,4 und 3,4 Zentimeter) belegen auch hier die sehr gute Erkennung der einzelnen Hindernisse.

Kleines Hindernis:

Die anfangs aufgestellte These, dass sich die Verteilung sowie Größe der *Samplepoints* positiv auf die Erkennung diverser Hindernisse auswirkt wird mit den Ergebnissen des kleinen Hindernisses bestätigt. Abbildung 24 belegt diese Aussage. Es wurden alle 12 Hindernisse ohne signifikant hohe Differenz zwischen berechneten und gemessenen Distanzen erkannt.

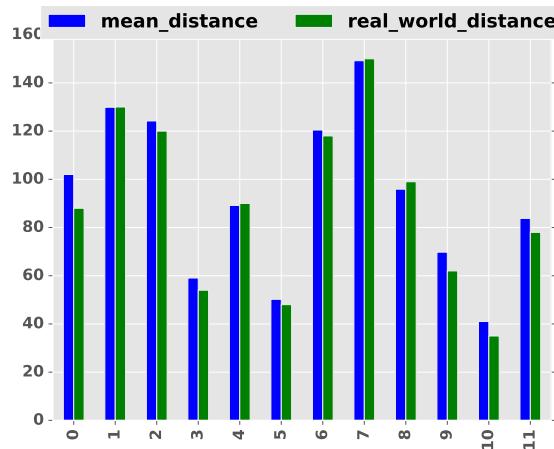


Abbildung 24: Abweichung zwischen real gemessener und berechneter Distanz bei Hindernis 3 der *Subimage Detection*.

Weiterhin ist auch die statistische Auswertung der Differenzen ein Zeichen für die robuste Erkennung kleiner Hindernisse. Die mittlere Differenz zwischen berechneten und gemessene Werten beträgt lediglich 3,5 Zentimeter. Auch die Standardabweichung von 4,5 cm deutet auf eine sehr robuste Erkennung von kleinen Hindernissen hin.

5.2.2 Weitere Versuche

Hinsichtlich der verschiedenen möglichen Faktoren welche eine erfolgreiche Erkennung beeinflussen können wurden im Rahmen der Evaluation beider Algorithmen weitere Tests durchgeführt. Jene umfassen die Erkennung kleiner Hindernisse unter Veränderung der Größe des Matching-Blocks. Weiterhin wird die Erkennung von „schwierigen“ Hindernissen (durchsichtige und reflektierende Flächen) getestet. Anschließend erfolgt ein weiterer Versuch welcher auf die Veränderung des *Samplepoint* Radius und die damit verbundenen Änderungen in der Erkennung abzielt. Das während der Tests genutzte Setup wurde während der Aufnahme nicht verändert. Auch das Objekt befand sich für jede Blockgröße an der selben Position. Zur Durchführung der Tests wurde das in

Abschnitt 5.2.1.1 und 5.2.1.2 genutzte kleine Hindernis verwendet. Auf eine Durchführung der Parameteränderung unter Benutzung des mittleren sowie großen Hindernisses wurde aufgrund der robusten Ergebnisse beider bewusst verzichtet.

5.2.2.1 Einfluss der *SGBM* Block Größe

Subimage Detection:

Um die Auswirkungen der Blockgröße auf die Distanzberechnung zu testen wurden verschiedene Kantenlängen des Blocks getestet. Für jede Größe wurde die Hinderniserkennung aus je 3 Distanzen ausgeführt. Im Folgenden werden zunächst die Daten dargelegt und im Anschluss daran analysiert. Eine Diskussion der erhaltenen Ergebnisse findet in Abschnitt 5.3 statt.

Blockgröße	berechnete Distanz (cm)	real gemessene Distanz (cm)
7	120.6	50
	-	100
	-	150
9	105.8	50
	-	100
	-	150
11	98.9	50
	-	100
	-	150
13	85.2	50
	-	100
	-	150
15	112.1	50
	-	100
	-	150
21	108.6	50
	-	100
	-	150

Tabelle 5.2: Ausgewertete Daten des Subimage Sets

Wie aus Abbildung 5.2 ersichtlich wird hat eine Änderung der Blockgröße keine Auswirkungen auf die Erkennung des Objektes. Die Hindernisse wurden lediglich bei einer Distanz von 50 Zentimetern erkannt, wobei die berechneten

Werte eine starke positive Abweichung aufweisen.

Samplepoint Detection:

Blockgröße	berechnete Distanz (cm)	real gemessene Distanz (cm)
7	63.8	50
	102.2	100
	149.1	150
9	57.3	50
	107.9	100
	153.6	150
11	68.0	50
	107.1	100
	149.1	150
13	51.4	50
	101.4	100
	151.1	150
15	57.3	50
	105.3	100
	149.1	150
21	56.0	50
	101.5	100
	147.3	150

Tabelle 5.3: Ausgewertete Daten des Subimage Sets

Die Änderung der Blockgröße hat auch bei der *Samplepoint Detection* keinen signifikanten Einfluss auf die Erkennung oder Berechnung der Hindernisdistanz. Im Gegensatz zur *Subimage Detection* wurden jedoch alle Distanzen auch korrekt berechnet. Die Differenz zwischen den berechneter und gemessener Distanz ist bei 13 Pixeln Blockgröße am geringsten.

5.2.2.2 Erkennung von reflektierenden und durchsichtigen Bereichen

Die Erkennung von Hindernissen bei reflektierenden sowie durchsichtigen Flächen wurde mit drei verschiedenen Tests untersucht. Zunächst wurde untersucht wie sich das System bei spiegelnden Flächen (etwa Fensterglas) verhält. Anschliessend wurde die Erkennung bei reflektierenden Flächen untersucht sowohl im 90° Winkel, also mit direkter Sicht auf das zu erkennende Hindernis als auch unter diversen anderen Winkeln. Weiterhin wurde untersucht inwiefern sich das System bei einer Kombination beider Problemstellungen verhält. Die

verwendete Methode zur Hinderniserkennung ist in allen folgenden Tests die *Samplepoint Detection* aufgrund der bisherigen besseren Ergebnisse. Die Gefahrenzone wurde aus Platzgründen auf 0, 1 bis 1,0 Meter definiert.

Zur Untersuchung reflektierender Flächen wurde der spiegelnde Bildschirm eines LED-Fernsehers genutzt. Da die Berechnung von Disparitätenkarten auch unter nicht optimalen Lichtverhältnissen robuste Ergebnisse liefert (Abbildung 25) stellt auch die schwarze Grundfarbe des spiegelnden Objektes prinzipiell keinen Konflikt dar. Trotz dessen wurde die Belichtungszeit der Kameras auf 60 - 70 μ s gesetzt, was die erhaltenen Bilder in ihrer Helligkeit erheblich verbessert.

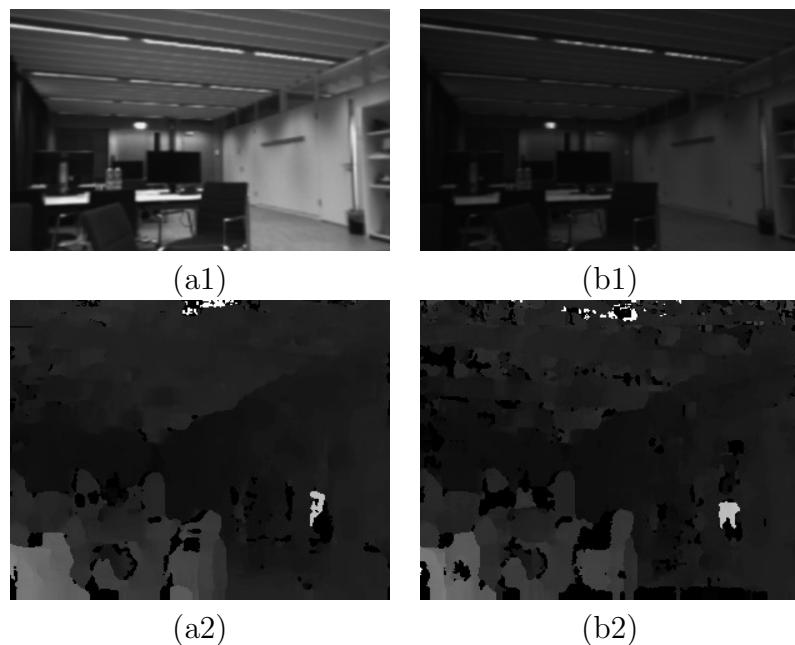


Abbildung 25: Vergleich zwischen einer hellen aufgenommenen Szene in (a1) und (a2) sowie den dazu gehörigen Disparitätenkarten in (a2) und (b2).

Die aufgenommenen Ergebnisse zeigen auf das Hindernisse erkannt wurden, jedoch ist der Ursprung dieser nicht eindeutig zu erkennen. Abbildung 26 stellt die berechneten und aufgenommenen Distanzen gegenüber. Aus diesen sind ebenfalls deutlich die Fehler innerhalb der Berechnung zu erkennen.

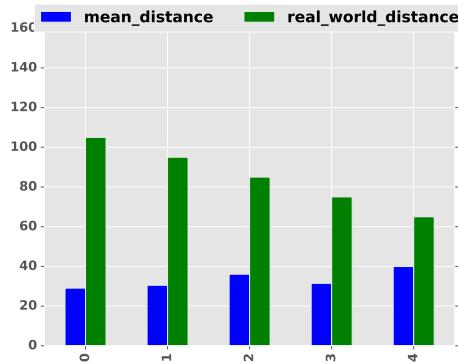


Abbildung 26: Darstellung Abweichung zwischen real gemessener und berechneter Distanz bei reflektierenden Flächen.

Ein weiterer durchgeföhrter Versuch beinhaltet die Veränderung des Winkels zwischen Kamera und reflektierender Ebene. Die ausgehende Vermutung ist die Erkennung dieser aufgrund diverser Lichtreflexionen. Dazu wurden die Bilder aus 3 verschiedenen Winkeln ($45, 22,5$ und $12,25^\circ$) unter Verwendung der Mitte der Basislinie als Rotationszentrum aufgenommen. Dabei repräsentieren die einzelnen Versuche die jeweiligen Winkel in absteigender Reihenfolge. Die gemessenen Distanzen repräsentieren jeweils die Hypotenuse des rechtwinkligen Dreiecks welches durch die Kameras und die beiden Punkte auf der Reflexionsebene gespannt wird. Abbildung 27 stellt die erlangten Ergebnisse dar.

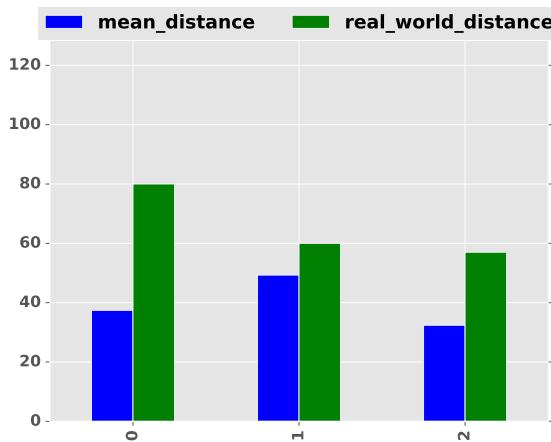


Abbildung 27: Abweichung zwischen real gemessener und berechneter Distanz bei reflektierenden Flächen unter Veränderung Aufnahmewinkels zur reflektierenden Fläche.

Aus diesen wird ersichtlich, dass kein Hindernis valide erkannt werden konnte. Die Differenz zwischen gemessenen und berechneten Distanzen deutet auf Fehler in der Berechnung der Disparitätenkarte aufgrund der Reflexion hin.

Eine Kombination von Reflexion und Durchsichtigkeit brachte ebenfalls keine validen Ergebnisse. In den ersten fünf Versuchen wurden zwar Hindernisse erkannt, jedoch weitaus näher, als sie sich eigentlich befunden haben. Zudem wurden Hindernisse erkannt obwohl sich keines innerhalb der Gefahrenzone befunden hat. Die somit erkannten Hindernisse wirken demnach als Resultat zufälliger Matches. Ab Versuch 5 befand sich das Fenster innerhalb der Gefahrenzone. Abbildung 28 zeigt eben diese Ergebnisse.

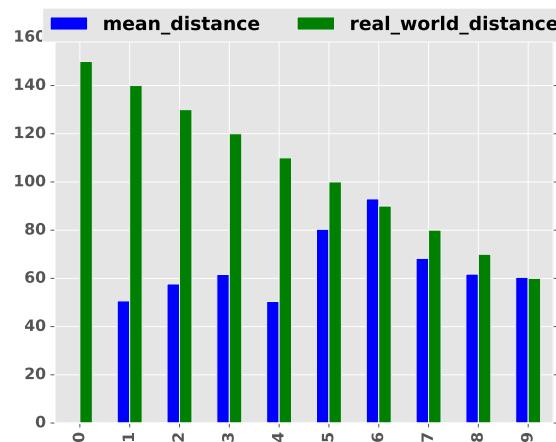


Abbildung 28: Abweichung zwischen real gemessener und berechneter Distanz bei reflektierenden einer Kombination von Reflexion sowie Durchsichtigkeit des zu findenden Hindernis

5.2.2.3 Erkennung unter Veränderung des Samplepoint Radius

Im Gegensatz zu *Subimages* bietet die Verwendung von *Samplepoints* neben der Gefahrenzone die Größe des Radius als veränderlichen Parameter. Im Rahmen der Evaluierung wurde getestet welche Auswirkung die Modifikation dessen auf die Erkennung hat. Dazu wurden die Konfigurationen 0 - 4 (0 indiziert dabei keinen Radius und somit die Verwendung eines einzelnen Pixels) mit jeweils fünf verschiedenen Positionen des kleinen Hindernisses getestet. Hindernis 1 und 2 wurden aufgrund der vorherigen robusten Ergebnisse in diesem Test nicht betrachtet.

Die Veränderung des Samplepoint Radius brachte lediglich bei der Verwendung eines einzelnen Pixels eine wahrzunehmende Veränderung mit sich. Das Hindernis wurde bei größeren Entfernungen zwar besser erkannt, jedoch wurden ebenfalls Bereiche mit Fehlinformationen als Hindernis gedeutet. Dies bestätigt die Verwendung der lokalen Nachbarschaft zur Erweiterung des Suchfeldes als wichtiges Kriterium für die Erkennung.

5.3 Diskussion

Im Rahmen dieses Kapitels wurden beide entwickelten Methoden in verschiedenen Versuchen getestet. Beide Methoden erkennen sowohl große als auch mittlere Hindernisse innerhalb der definierten Gefahrenzone. Die ermittelten Distanzen weisen zwar kleine Ungenauigkeiten auf, jedoch sind diese eher ein Resultat von Messungenauigkeiten sowie der verwendeten Bildgröße. Es erfolgt somit eine Auswertung der erlangten Ergebnisse, sowie eine Gegenüberstellung beider Algorithmen in Hinsicht auf die Robustheit der Erkennung sowie deren Laufzeit.

5.3.1 Robustheit

Die in den Abschnitten 5.2.1.1 und 5.2.1.2 erlangten Ergebnisse belegen die generelle Effektivität und Robustheit beider Algorithmen. Die Grundidee der Methoden gleicht sich in einigen Punkten, ebenso die Resultate. Beide Algorithmen liefern robuste Ergebnisse unter der Voraussetzung einer gewissen Hindernisgröße, wobei das kleine Objekt die größte Anforderung an diese darstellt. Die Detektion des großen sowie mittleren Hindernisses ergab in allen Versuchen die Ergebnisse mit den niedrigsten Differenzen zwischen berechneten und gemessenen Entfernung.

5.3.1.1 Subimage Detection

Das Konzept der *Subimage Detection* ist in Bezug auf große und mittlere Hindernisse als robust einzustufen, jedoch ist die Berechnung des Mittelwertes der Matrizen auch der größte Konflikt. Weist der Hintergrund eines Objektes sehr geringe Disparitäten auf, so besteht die Möglichkeit das der Hintergrund den Mittelwert eines einzelnen *Subimages* insofern beeinflusst, dass das Hindernis (innerhalb dieser Submatrix) nicht mehr erkannt werden kann. Dies ließ sich bei nahezu allen Versuchen und Hindernisgrößen beobachten. Im Fall der großen und mittleren Hindernisse stellt dies keinen maßgeblichen Konflikt dar, da die Fläche der Hindernisse trotzdem für eine Erkennung ausreicht. Ungeachtet dessen stellt gerade dieser Konflikt ein Problem in der Erkennung einzelner kleiner Hindernisse dar.

Bei der Erkennung großer Hindernisse wurden diese zwar alle erkannt, jedoch nicht der gesamte Bereich den sie umfasst haben. Wie Abbildung 29 zeigt wurden der obere und untere Bereich des Hindernisses im 5. Frame nicht erkannt. Dies resultiert daraus, dass nur ein geringerer Anteil der Pixel dieser Segmente das Objekt repräsentieren, so dass der Mittelwert in diesem Fall maßgeblich

durch die sehr kleinen Disparitäten des Hintergrunds verkleinert wird.

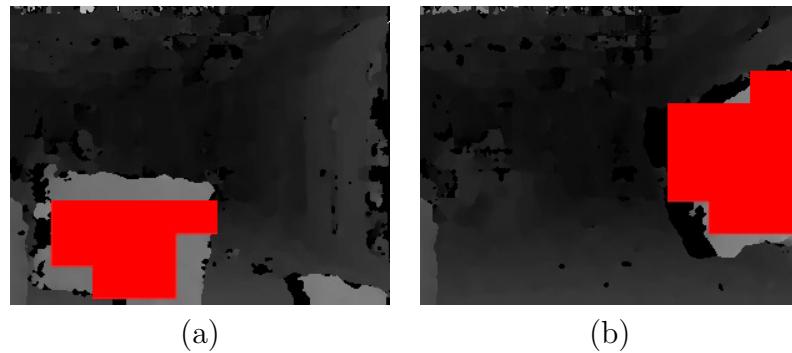


Abbildung 29: Darstellung des in Versuch 5 (a) und Versuch 8 (b) gefundenen Hindernisse, rot markierte Bereiche zeigen ein erkanntes Hindernis innerhalb eines *Subimages*

Die Detektion von Hindernis 2 gelang ebenfalls in allen Versuchen, jedoch zeigen die Werte eine größere Abweichung in den Differenzen von berechneter und gemessener Distanz. Im Fall von Frame 0 umfasst das Objekt insgesamt 17 *Subimages* in denen es erkannt wurde. Als Resultat des Stabes wurden jedoch noch 2 weitere erkannt. Frame 1 weist zwar keine große Abweichung zwischen der berechneten und realen Distanz auf, jedoch wird das Objekt zu Teilen nicht erkannt. Grund dafür ist wiederum der bereits beschriebene weit entfernte Hintergrund. Bei einer Distanz von 120 Zentimetern ist eine Erkennung dieser Hindernisgröße in nur 2 *Subimages* ungewöhnlich. Dies resultiert aus der verwendeten Blockgröße.

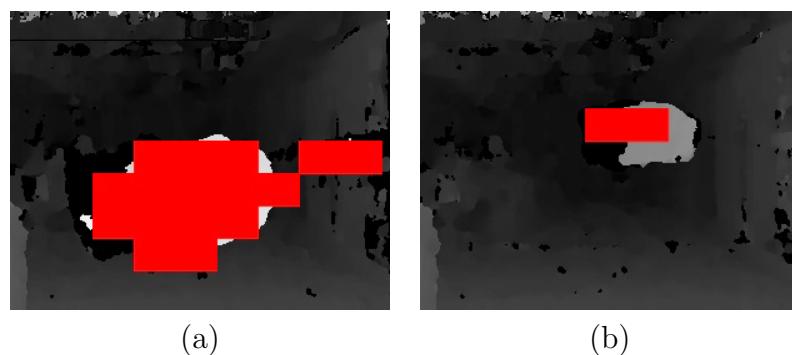


Abbildung 30: Darstellung des in Versuch 0 (a) und Versuch 3 (b) gefundenen Hindernisse, rot markierte Bereiche zeigen ein erkanntes Hindernis innerhalb eines *Subimages*

Selbiges Resultat lässt sich in den Frames 2, 6 und 7 erkennen. In Frame 9 dagegen konnte kein Hindernis erkannt werden, da sich das Objekt an der Grenze der Gefahrenzone befunden hat und demnach die Abweichung der Disparität zu keiner Erkennung führen konnte.

Die Erkennung kleiner Hindernisse gestaltet sich aufgrund diverser Faktoren schwer. Ein limitierender Faktor ist die Fläche. Ist ein einzelnes Hindernis so klein, dass es während der Berechnung des Mittelwertes aufgrund der umliegenden Disparitäten untergeht, so kann auch kein Hindernis erkannt werden. Dies macht sich in vielen der getesteten Bildern bemerkbar. In den Frames 2, 5, 7 und 10 wurde das Hindernis nicht erkannt, da es sich entweder an einer Kreuzung verschiedener *Subimages* befand, oder die verwendete Blockgröße nicht ausreicht, dadurch wird der Mittelwert aller Teilmatrizen nicht signifikant beeinflusst. Auch der Stab wurde in den Frames 3, 6 sowie 8 als zusätzliches Hindernis erkannt. Dies ist ein Resultat der relativ geringen Distanz zu den Kameras. Die hohen Entfernungsdifferenzen resultieren in diesen Fällen wiederum aus der großen Entfernung der Hindernisse zum Hintergrund. Grundlegend ist der hohe Anteil an Hintergrundpixeln im Segment ist in nahezu allen Versuchen der beeinflussende Faktor.

5.3.1.2 Samplepoint Detection

Die Erkennung von Hindernissen mit Hilfe der *Samplepoints* ist ebenfalls eine robuste Methode. Dabei reicht die Anzahl der umfassenden Pixel eines jeden aus um selbst kleine Hindernisse zu erkennen. Der nicht beachtete Bereich zwischen den *Samplepoints* kann entweder 6, 4, 2 oder keinen Pixel betragen, in Abhängigkeit des verwendeten Radius. Der fest gewählte horizontale und vertikale Abstand der *Samplepoints* (8 Pixel) reicht jedoch nicht immer um das Hindernis zu erkennen. In vielen Fällen wurde das Hindernis nicht mehr direkt erkannt, sondern die Befestigung dessen. Das Problem befindet sich dabei bei der zugrunde liegenden Disparitätenkarte. Der hintere Bereich der Szene unterscheidet sich farblich nicht signifikant vor der Farbe des Hindernisses. Zudem ist das Objekt bei einer Entfernung von 150 Zentimetern so klein das es höchstwahrscheinlich durch die verwendete Blockgröße nicht erkannt oder mit benachbarten Bereichen verbunden wird.

Hindernis 1 wurde in allen Versuchen robust erkannt. Die auftretenden Differenzen waren marginal und daher eher Resultat von Messungenauigkeiten. Bereiche in denen aufgrund fehlender Textur keine Korrespondenz zugeordnet werden konnte wurden aufgrund der hohen Dichte der *Samplepoints* trotzdem erkannt. Wie Abbildungen 31 (a) und (b) zeigen wurden nicht gematchte Be-

reiche auch nicht als Hindernis erkannt, die umliegenden texturierten Bereiche überwiegen jedoch und sorgen daher für eine Detektion.

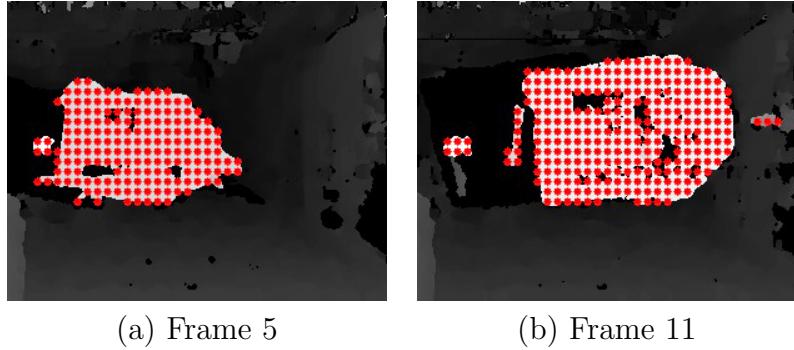


Abbildung 31: Darstellung erkannter Bereiche von Hindernis 2 der *Samplepoint Detection* in Versuch 5 (a) und 11(b)

Hindernis 3 wurde von der *Samplepoint Detection* ebenfalls überraschend gut erkannt. Trotz dessen ist die Detektion dieser bei Entferungen von mehr als 60 Zentimetern stark von der Orientierung des Objektes, dem Hintergrund sowie den verwendeten Parametern zur Berechnung der Disparitätenkarte abhängig. Da die geringe Auflösung der Kameras keine detaillierte Aufnahme der Textur des Hindernisses zulässt ist es möglich das das Objekt aufgrund der Blockgröße des *SGBM* mit dem Hintergrund zusammen gematcht wird.

5.3.2 Einfluss der SGBM Blockgröße

Eine Veränderung der SGBM Blockgröße brachte keine signifikanten Verbesserungen in der Hinderniserkennung mit sich. Im Fall der *Subimage Detection* wurden kleine Hindernisse weder besser noch schlechter erkannt. Ab einer Distanz von mehr als 50 Zentimetern erfolgte keine Detektion. Auch die Ergebnisse der *Samplepoint Detection* veränderten sich nicht maßgeblich im direkten Vergleich zur vorherigen Erkennung des kleinen Hindernisses.

Bei den Ergebnissen der *Subimage Detection* werden die berechneten Distanzen erheblich von den Pixeln des Hintergrundes beeinflusst, was in den verzerrten berechneten Distanzen resultiert. Dabei auftretende Differenzen zwischen den real gemessenen Entferungen sowie den berechneten resultieren aus Interferenzen während der Neuberechnung der Disparitätenkarte in jedem Frame.

In allen durchgeführten Tests der *Samplepoint Detection* lässt ab einem Abstand von 100 Zentimetern erkennen, dass das eigentliche Hindernis nicht mehr gefunden wurde, sondern die Befestigung dessen das erkannte Hindernis

repräsentiert (siehe Abbildung 32). Ein Grund für diese „falsch“-Erkennung könnte auch hier die Verwendung des Mittelwertes, sowie der nicht beachtete Raum zwischen den einzelnen Samplepoints sein. Bei einem Radius von 2 Pixeln beträgt dieser ebenfalls 2 Pixel. Eine Verzerrung der Werte durch den Hintergrund ist in diesem Fall unwahrscheinlich, die Verbindung der Distanz sowie der geringen Aufnahmeflußlung der Kameras führt eher dazu, dass das Hindernis vor dem Hintergrund als Teil dessen wirkt. Die plausibelste Fehlerquelle ist daher am wahrscheinlichsten die zugrunde liegende Disparitätenkarte.



Abbildung 32: Testset unter Verwendung einer Blockgröße von 13 Pixeln in verschiedenen Distanzen (50 cm bis 150 cm v.l.n.r.)

5.3.3 Reflektierende und durchsichtige Flächen

Die Erkennung reflektierender und durchsichtiger Flächen zeigte sich als grundlegend problematisch. Es wurden in nahezu jedem Versuch Hindernisse erkannt, jedoch nur in seltenen Fällen korrekt. Dies kann aus der homogenen Farbgebung der verwendeten Bilder resultieren. Der genaue Ursprung der erkannten Hindernispunkte ist nahezu unmöglich zu identifizieren, da die berechnete Disparitätenkarte keine genaue Lokalisierung von Formen der Szene zulässt. Dies liegt sowohl bei rein spiegelnden Flächen im 90° Winkel, als auch bei einer Veränderung des Aufnahmewinkels feststellen. Bei einer Kombination von reflektierenden und spiegelnden Flächen wurden zumeist nur wenige Punkte erkannt welche nicht das Objekt als ganzes abbilden (siehe Abbildung 33). Die gefundenen Hindernisse sind das Resultat eines reflektierten Lichtpunktes auf dem Glas.

5.3.4 Veränderung des Subimage Radius

Die Anpassung des *Subimage* Radius brachte ebenfalls keine signifikanten Verbesserungen in der Erkennung. Bei einem Radius von 0, also der Verwendung eines einzelnen Pixels wurde Hindernis 3 bei größeren Entfernung zwar besser erkannt, im gleichen Zug aber auch Ausreißer der Disparitätenkarte. Dieser

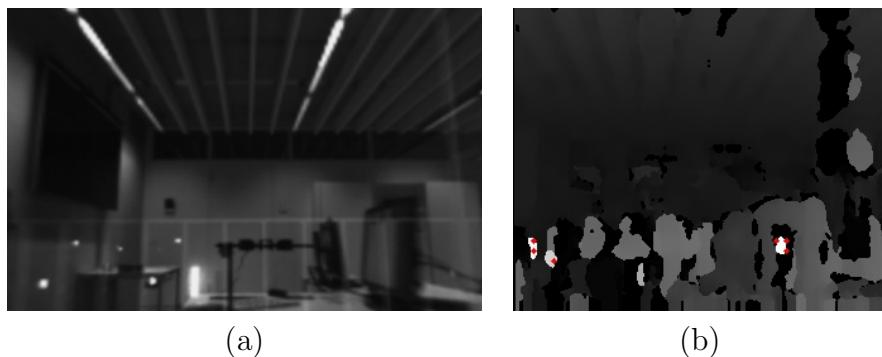


Abbildung 33: Kombination von spiegelnden und durchsichtigen Flächen (a) linkes Referenzbild (b) Berechnete Disparitätenkarte mit markierten Hindernissen

ungewünschte Nebeneffekt kann bei einer realen Anwendung zu Falschentscheidungen von Seiten der Hindernisvermeidung führen und somit im schlimmsten Fall zum Ausfall des UAS.

5.3.5 Laufzeit

Hinsichtlich der Performance der entwickelten Systeme wurden verschiedene Punkte betrachtet. Die zugrunde liegende Berechnung der Disparitätenkarte ist der wohl wichtigste Faktor. Ist dieser Prozess langsam, so ist auch die Performance der Hinderniserkennung eingeschränkt. Weiterhin wird auch die Performance der einzelnen Methoden untersucht. Dabei werden folgende Situationen untersucht:

1. Analyse der gesamten Schleife des Hauptprogramms
 - (a) Hindernisse bewegen sich durch die Gefahrenzone
 - (b) der gesamte Sichtbereich ist mit einem Hindernis gefüllt
 - (c) es befindet sich kein Hindernis in der Gefahrenzone
2. Analyse einzelner Erkennungsschritte
 - (a) Geschwindigkeit der *update* Funktion
 - (b) Geschwindigkeit der *detectObstacles* Funktion mit einem Hindernis im gesamten Sichtbereich
 - (c) Geschwindigkeit der *detectObstacles* Funktion ohne Hindernisse

Zu Beginn ist ein essenzieller Schritt die Geschwindigkeit der Berechnung der Disparitätenkarte zu analysieren. Dabei wurden beide Kameras im gebinnten

Modus getestet. Die Aufnahmerate der Kameras beträgt dabei je nach gewählter Belichtungszeit 55 - 60 Frames pro Sekunde. Die unter Veränderung der Blockgröße erhaltenen Parameter sind in Tabelle 5.4 dargestellt. Aus dieser ist zu erkennen, dass eine Modifikation dieses Parameters nur marginale Änderungen in der Geschwindigkeit auftreten. Die folgenden gemessenen Werte sind die durchschnittliche Framerate aus 1000 Einzelbildern.

Block Größe (Pixel)	Zeit pro Frame (s)	Frames pro Sekunde
7	0.0412	24.21
9	0.0420	23.77
11	0.0413	24.17
13	0.0410	24.36
15	0.0412	24.22
21	0.0413	24.17

Tabelle 5.4: Blockgröße und daraus resultierende Frameraten

Die dabei gemessene Bildwiederholrate von 24 Einzelbildern pro Sekunde ist eine gute Voraussetzung für die Hinderniserkennung. Unter der Annahme, dass es zu keiner Verlangsamung dieser kommt ist es möglich sich mit einer Geschwindigkeit von 12 m/s zu bewegen und für jeden zurückgelegten Meter zwei berechnete Disparitätenkarten zu erhalten. Eine solche Geschwindigkeit ist bei besagter Framerate jedoch nur in Abhängigkeit der Gefahrenzone möglich. Zudem muss berücksichtigt werden, dass die vorhandene Rechenleistung, die in dieser Arbeit ausschließlich für die Hinderniserkennung genutzt wurde, von einem autonomen System auch für weitere Softwaremodule, wie die Selbstpositionierung und Hindernisvermeidung in Anspruch genommen wird.

Die Geschwindigkeit der eigentlichen Hinderniserkennung ist ebenfalls ein wesentlicher Faktor in der Betrachtung der gesamten Performance. Dazu wurden besagte Tests durchgeführt. Die daraus erhaltenen Ergebnisse für die *Subimage Detection* finden sich in Tabelle 5.5.

Aus dieser wird ersichtlich, dass Szenario 1(a), welches einer echten Anwendung am nächsten kommt, bereits eine Framerate von 255 Einzelbildern pro Sekunde aufweist. Die darauf folgenden Tests bestätigen die Annahme, dass keine wesentlich langsamere Framerate aufgrund der Hinderniserkennung zu erwarten ist. Im schlechtesten Fall, wenn ein Hindernis den gesamten Sichtbereich einnimmt, ist die Framerate der gesamten Prozessierung (von Bildaufnahme bis zu den detektierten Hindernissegmenten) nicht geringer als 22.27 Bilder pro Sekunde wie die folgende Rechnung aufzeigt. Dabei entsprechen die genutzten Werte denen aus 5.4 mit 13 Pixeln Blockgröße sowie 5.5 Szenario 1(b).

Szenario	Zeit pro Frame (s)	Detection fps
1(a)	0.0039	255.96
1(b)	0.0072	138.89
1(c)	0.0227	438.80
2(a)	0.0022	451.39
2(b)	0.0047	210.12
2(c)	0.000019	51232.51

Tabelle 5.5: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate der *Subimage Detection*

$$fps = \frac{1}{t_{frame}}$$

$$t_{frame} = 0,0410s + 0,0039s = 0,0449s \quad (5.1)$$

$$fps = \frac{1}{0.0449} = 22,27$$

Die somit verloren gegangenen 3 - 4 Frames in der Disparitätenberechnung sorgen noch immer für eine schnelle Erkennung von Hindernissen. Die hohe Framerate in 2(b) resultiert aus der Funktionsweise der *detectObstacles* Funktion. Jene berechnet nur eine Pointcloud wenn die aktualisierten Werte innerhalb der Gefahrenzone liegen. Ohne vorhandene Hindernisse passiert somit nichts.

Die nachfolgende Tabelle (5.6) stellt die Ergebnisse des Versuches für die *Samplepoint Detection* dar. Auf den ersten Blick sind die erreichten Framerates (mit Ausnahme 1(b), sowie 2(b)) generell niedriger als jene der *Subimage Detection*.

Szenario	Zeit pro Frame (Detection in s)	Detection fps
1(a)	0.00372	268.79
1(b)	0.00821	122.80
1(c)	0.00260	374.97
2(a)	0.06155	641.65
2(b)	0.00650	152.83
2(c)	0.00002	49453.51

Tabelle 5.6: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate der reinen Hinderniserkennung der *Samplepoint Detection*

Gerade die ermittelte Framerate in 1(b) gibt Anlass zu der Annahme, dass die *Samplepoint Detection* mehr Rechenleistung benötigt. Dies resultiert aus der

Anzahl der zu betrachtenden Objekte. Im Vergleich zur *Subimage Detection* werden nicht nur 81 Bereiche betrachtet, sondern , je nach verwendeter Bildgröße 5640 (volle Auflösung) oder 1410 (horizontales und vertikales Binning), wobei sich die Anzahl in Abhängigkeit der Parameter verringert. Auch im Echtwelt Szenario finden sich 13 Einzelbilder weniger pro Sekunde. Lediglich das Update erfolgt schneller. Die höhere Geschwindigkeit in der Aktualisierung resultiert aus der geringeren Anzahl an Pixeln welche betrachtet werden müssen. Nach der in 5.1 erläuterten Rechnung ergibt sich bei der *Samplepoint Detection* eine durchschnittliche Framerate von 22,37 Bildern/s. Anhand der berechneten Werte ist nun deutlich zu erkennen, dass sich die Gesamtperformance beider Methoden bis auf eine marginale Abweichung von 0.1 Einzelbildern pro Sekunde gleicht.

Zur Steigerung der Performance wurden in einem weiteren Test die initialen Bilder zur Berechnung der Disparitätenkarte um den Faktor 1/2 skaliert, wobei alle restlichen verwendeten Parameter unverändert blieben. Die dadurch erhaltenen Ergebnisse sind in den Tabellen 5.7 und 5.8 abgebildet.

Szenario	Zeit pro Frame (Detection in s)	Detection fps
1(a)	0.0022	440.33
1(b)	0.0026	381.62
1(c)	0.0007	1318.17
2(a)	0.0007	1328.21
2(b)	0.0019	515.10
2(c)	0.000006	174916.76

Tabelle 5.7: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate der reinen Hinderniserkennung mit skalierten Ausgangsbildern der *Subimage Detection*

Szenario	Zeit pro Frame (Detection in s)	Detection fps
1(a)	0.0015	634.28
1(b)	0.0023	418.96
1(c)	0.0016	612.04
2(a)	0.0004	2132.13
2(b)	0.0024	403.07
2(c)	0.000006	160745,86

Tabelle 5.8: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate der reinen Hinderniserkennung mit skalierten Ausgangsbildern der *Samplepoint Detection*

Somit kann aus diesen geschlossen werden, dass eine verringerte Größe der Ausgangsbilder die Gesamtframerate immens steigert. Dies resultiert hauptsächlich aus der Berechnung der zugrundeliegenden Disparitätenkarte. Diese ist in dieser Konfiguration auf durchschnittlich 94 Einzelbilder pro Sekunde angestiegen. Jedoch ist dieser Wert nur In Abhängigkeit der Wiederholrate der Aufnahme zu erreichen. Im horizontalen sowie vertikalen Binning Modus liegt diese durchschnittlich bei 60 – 66 Bildern pro Sekunde. Für eine Erkennung in Szenario 1(c) ergeben sich demnach die folgenden möglichen Werte bei beiden Methoden:

$$\begin{array}{ll} \text{Subimage Detection:} & 75,53 \text{ fps} \\ \text{Samplepoint Detection:} & 77,28 \text{ fps} \end{array}$$

5.3.6 Gegenüberstellung

Resultierend aus den durchgeführten Versuchen werden beide vorgeschlagenen Methoden im folgenden gegenübergestellt. Dabei wird insbesondere auf die finale Punktwolke sowie die Erkennung kleiner Hindernisse eingegangen. Des Weiteren wird diskutiert ob eine Eigenbewegung bzw. sich bewegende Objekte in der Szene einen Einfluss auf die Hinderniserkennung haben.

Bei der Erkennung sehr kleiner Hindernisse ist jedoch zu erkennen, dass die entwickelte *Samplepoint Detection* wesentlich robustere Ergebnisse liefert als die *Subimage Detection*. Dies resultiert vornehmlich aus der signifikant geringeren Anzahl an Pixeln. Dadurch sind die verwendeten *Samplepoints* weniger empfänglich für Verzerrungen, beispielsweise durch einen weit entfernten Hintergrund, als *Subimages*. Enthält ein einziger *Samplepoint* zu wenige Daten um als Hindernis angesehen zu werden, so ist die Wahrscheinlichkeit, dass seine

Nachbarn diese Information enthalten bzw. erfassen konnten höher als beispielsweise bei benachbarten *Subimages*. Dies ist auch als der große Vorteil der *Samplepoint Detection* anzusehen.

Auch die berechneten Punktwolken unterscheiden sich sowohl in ihrer Form als auch der Anzahl an Punkten. Tabelle 5.9 stellt dabei zwei gerenderte Pointclouds desselben Objektes dar.

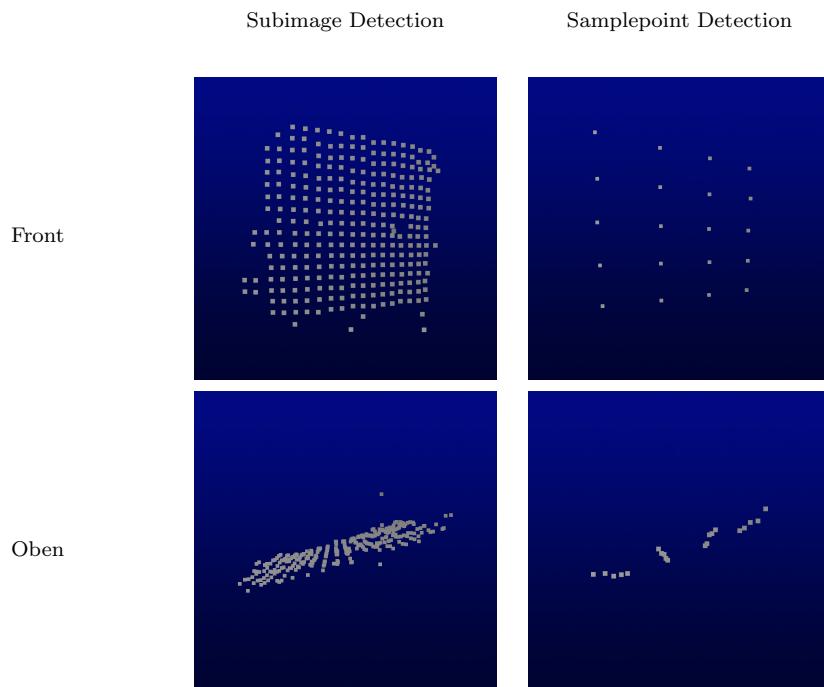


Tabelle 5.9: Gespeicherte Hindernis Punktwolke desselben Objektes beider Methoden.

Prinzipiell ist die Punktwolke der *Samplepoint Detection* aufgrund der Verwendung von signifikant mehr Messpunkten wesentlich detaillierter. Dies kann beispielsweise bei einer dreidimensionalen Kartographierung von Vorteil sein. Jedoch wäre es auch möglich, das aufgrund der kleinen Fläche mehr Ausreißer erkannt werden. Dies konnte zwar im Rahmen der Tests nur bei spiegelnden und durchsichtigen Flächen beobachtet werden, ist jedoch eine prinzipiell mögliche Fehlerquelle bei der späteren Hindernisvermeidung.

Weiterhin ließ sich während der Versuchsdurchführung deutlich erkennen, dass Bewegungen einen großen Einfluss auf die Hinderniserkennung haben. Wurden die Hindernisse bewegt, konnte gerade im Fall der *Subimage Detection*

festgestellt werden, dass die Erkennung kleiner Hindernisse signifikant besser funktionierte. Dadurch eliminieren sich bereits beschriebene Konfliktfälle in denen sich das zu erkennende Hindernis an der Kreuzung mehrerer *Subimages* befand. Die Bewegung sorgte in diesem Fall dafür das die Hindernisse in mehr Frames erkannt wurden als in einer statischen Szene. Selbiges Phänomen trat auch bei der *Samplepoint Detection* auf.

Kapitel 6

Limitierungen und Lösungsansätze

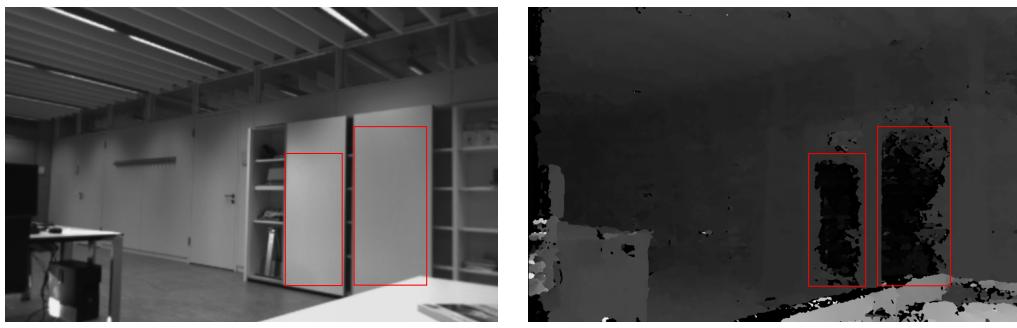
In der Erkennung von Hindernissen mithilfe von passiv optischen Systemen können verschiedene Faktoren der Grund für eine fehlerhafte Erkennung sein. Sei es die Berechnung einer Disparitätenkarte von Bereichen mit einer Vielzahl homogener oder reflektierender Flächen oder die Veränderung der Lichtverhältnisse in einem der beiden Kamerabilder. In diesem Kapitel werden einige der bestehenden, sowie einige potentiell mögliche Limitierungen erläutert sowie dazugehörige Lösungsansätze entwickelt. Weiterhin werden diese in Abschnitt 6.1 diskutiert. Zudem werden sowohl eigene, als auch literarisch belegte Lösungsansätze präsentiert

6.1 Bestehende Limitierungen und Lösungsansätze

Die Validierung der erkannten Hindernisse ist ein kompliziertes Problem in der autonomen Hinderniserkennung. Durch etwaige äußere Einflüsse wie die Veränderung der Lichtverhältnisse kann die Berechnung der Tiefenkarte fehlerhaft sein. Dies kann im Fall eines autonomen Flugs dazu führen, dass das UAS ein Hindernis innerhalb eines Korridors erkennt und aufgrund dessen versucht diesem imaginären Hindernis auszuweichen. Gerade in engen Umgebungen ohne viel verfügbaren Platz kann dies zum totalen Systemausfall führen. Selbiges gilt für nicht erkannte Hindernisse. Daher sollte ausgewertet werden ob die erkannten Hindernisse Resultat fehlerhafter Disparitätenkarten sind. Ein Ansatz zur Vermeidung solcher Falscherkennungen wäre, die Objekte insofern zu verfolgen, dass für jeden Frame (in Abhängigkeit der zugrundeliegenden Fra-

merate) überprüft wird, ob im vorherigen Frame bereits ein Objekt gefunden wurde. Erst nachdem dies sichergestellt wurde, wird daraufhin eine Warnung ausgegeben. Eine Möglichkeit der Vorhersage der Position von bereits detektierten Hindernissen in früheren Frames ist durch den Einbezug des Wissens über die Eigenbewegung der Drohne möglich.

Eine andere Problematik stellt die Erkennung von homogenen, spiegelnden sowie durchsichtigen Flächen dar. Aufgrund fehlender Texturen sowie vorhandener Pixeldifferenzen kann die Korrespondenz zweier Pixel unter Umständen nicht bestimmt werden. Diese homogenen Bereiche haben demnach fehlerhafte Informationen zur Folge, so kann einem Pixel an einer weißen Wand kein zugehöriger Pixel zugeordnet werden. Außerdem besteht die Möglichkeit das falsche Bildpunkte miteinander gematcht werden welche wiederum falsche Disparitäten zur Folge haben. Abbildung 34 zeigt deutlich, in welchen Bereichen der SGBM aufgrund fehlender Textur keine Korrespondenzen finden konnte.



(a) linkes Kamerabild

(b) Disparity Map

Abbildung 34: Konflikt in der Berechnung der Disparität bei nicht texturierten Flächen. Dabei ist (a) das linke aufgenommene Kamerabild und (b) die dazu gehörige Disparitätenkarte. Die in rot markierten Flächen sind das Resultat homogener Texturen.

Auch spiegelnde Flächen sind eine optische Problemstellung. Aufgrund der Projektion eines anderen Bereiches kommt es zu falschen Informationen innerhalb der Disparitätenkarte. Hindernisse können zwar erkannt werden sofern sich die spiegelnde Fläche in einer günstigen Position befindet. Dies ist der Fall, wenn beide Kameras exakt den selben Bereich erfassen. Jedoch wird auch dabei anstelle des eigentlichen Objektes eine weiter entfernte Distanz wahrgenommen und somit ein Hindernis, welches sich an einer gänzlich anderen Stelle befindet. Ist dies nicht der Fall so kann auch im Fall spiegelnder Flächen keine

Disparität und folglich keine Tiefe wahrgenommen werden.

Eine weitere Fehlerquelle sind durchsichtige Bereiche. Diese vereinen zum einen Fehlerquellen, welche auch bei spiegelnden Arealen auftreten, zum anderen werden Objekte, die sich hinter einer Glasscheibe befinden zwar erkannt (sofern keine Reflexion vorliegt), jedoch das Glas als eigentliches Hindernis nicht. Weiterhin besteht die Möglichkeit, dass Reflexionen aus Perspektive der beiden Kameräne betrachtet einen anderen Tiefeneindruck vermitteln. Objekte werden so entweder weiter entfernt oder auch in kürzerer Distanz erkannt als sie sich wirklich befinden.

Aufgrund des Aspektes, dass das System für den Gebrauch in Innenbereichen konzipiert ist, gibt es verschiedenste Konflikte bei der Anwendung in Außenbereichen. Zum einen ist die Verschlusszeit der Kameräne fixiert. Die Verwendung einer automatischen Belichtungszeit ist nur für jede Kamera einzeln möglich. Um mit verschiedenen Lichtverhältnissen umgehen zu können, ohne dabei die Möglichkeit der Korrespondenzanalyse zu verlieren, muss die Verschlusszeit daher bei beiden Kameräne immer gleich sein. Weiterhin ist die Erkennung des Bodens ein zu betrachtender Aspekt. Mit den vorgeschlagenen Methoden wird der Boden aktuell als Hindernis erkannt, was einerseits seine Berechtigung hat, andererseits auch bei niedrigen Flughöhen zu einer fehlerhaften Erkennung als Hindernis führt.

Die Erkennung sehr kleiner Hindernisse ist aufgrund verschiedenster Faktoren deutlich eingeschränkt. Ist das Hindernis so klein, dass es nur Teile eines *Subimages* oder *Samplepoints* ausfüllt, so ist die Erkennung dessen sehr stark vom Hintergrund abhängig. Dies resultiert aus der Berechnung des Mittelwertes zur Erkennung der Hindernisse. Ist der Hintergrund sehr weit entfernt beeinflusst dieser die Summe aller Pixel eines *Subimages/Samplepoints* insofern, dass die gemittelte Disparität nicht mehr innerhalb der Gefahrenzone liegt.

6.2 Diskussion

Ein Ansatz, erkannte Hindernisse zu validieren ist, sich auf die letzte bekannte Position des Objektes zu berufen und im nächsten Einzelbild zu überprüfen, ob es sich noch immer an derselben Position befindet. Dabei würde für jedes *Subimage*/ jeden *Samplepoint* ausgewertet werden, ob sich das Hindernis noch innerhalb dessen befindet. Jedoch schränkt dies die Erkennung bewegter Objekte stark ein. Ist die Geschwindigkeit eines Hindernisses so hoch, das in mindestens zwei aufeinanderfolgenden Frames kein Objekt im selben *Subi-*

mage/Samplepoint erkannt werden kann so würde das System kein Hindernis erkennen können. Diese Methode ist daher prinzipiell mögliche, jedoch stark von der Framerate sowie der eigenen bzw. Bewegung des Objektes abhängig. Grundlegend ist das System aufgrund der Beschaffenheit der *Subimages* für die *Subimage Detection* geeignet, da die Größe der *Subimages* eine solche Validierung zulassen würde. Im Falle der *Samplepoint Detection* müssten auch die unmittelbar benachbarten Messpunkte mit einbezogen werden sodass die Bewegung auch verfolgt werden kann. Eine Kombination von Feature Tracking wäre dabei hilfreich, jedoch sehr rechenaufwändig.

Um auch die Erkennung homogener Flächen gewährleisten zu können, besteht die Möglichkeit, die Szene mithilfe externer Lichtquellen auszuleuchten um eine Texturierung zu erzwingen. Mit Hilfe eines Lasers könnte beispielsweise ein zufälliges Linienmuster auf die Szene projiziert werden, durch welches es dem Matching Algorithmus möglich ist, korrespondierende Punkte innerhalb eines nicht texturierten Bereiches zu finden. Die Zufälligkeit des Musters ist dabei ein wichtiges Kriterium, da die Gleichmäßigkeit eines Rasters zu falschen Korrespondenzen führen könnte. Die Anwendung dieses Verfahrens ist jedoch hauptsächlich in Innenbereichen möglich, da die zu erkennenden Entfernung unter Betrachtung der Einzelbild-Auflösung eine Detektion dieser zulassen würden.

In vielen Fällen sind gefundene Stereo Korrespondenzen nicht eindeutig, so beschreiben Geiger et al. [Geiger et al., 2011] die Berechnung solcher im ELAS (*Efficient Large-scale Stereo*) Algorithmus. Zu Beginn des Matching Verfahrens wird eine karge Menge an Hilfspunkten berechnet. Diese sind dabei als Pixel definiert, welche aufgrund der gegebenen Textur oder ihrer Einzigartigkeit robust gematcht werden können. Dabei werden solche „Support Points“ als eine Konkatenation der Koordinaten des Pixels (x_m, y_m) sowie der zugehörigen Disparität d_m definiert. Die Verteilung der Hilfspunkte auf den Referenzbildern erfolgt in gleichem Abstand zueinander. Unklare Matches werden währenddessen anhand eines Schwellwertes aussortiert. Anschliessend wird unter Zuhilfenahme der erstellten Hilfspunkte nach sogenannten *Observations* gesucht, welche aus den Bildkoordinaten (x_n, y_n) sowie einem Feature Vector f_n bestehen. Der Feature Vektor ist dabei entweder die Intensität des Pixels oder ein berechneter Deskriptor, der aus den Intensitäten der lokalen Nachbarschaft des Pixels berechnet wird. Mithilfe dieser beiden Informationen wird ein Gitternetz generiert. Die eigentliche Berechnung der Disparität erfolgt mit Hilfe der *Observations* auf deren jeweiliger Epipolarlinie unter Benutzung des *Maximum a posteriori* (MAP) Verfahrens und wird für jedes Bild separat angewandt.

Der von Tsin et al. [Tsin et al., 2003] entwickelte *Stereo Matching* Algorith-

mus kann zwischen Reflexion und Transparenz unterscheiden. Grundlage ist das Reflexionsmodell nach Hero (Abbildung 35), wonach der Einfallswinkel eines reflektierten Lichtstrahls (gemessen bezüglich der Oberflächennormale) gleich dem Ausfallwinkel ist. Somit ist die Position der Reflexion eines Szene-punktes unabhängig vom aktuellen Sichtfenster, das sich diese an einer festen Position hinter dem Reflektor befindet. Daher kann die Abhängigkeit eines Szenepunktes von seiner Reflexion ignoriert werden. Tsin et al. beschreiben ein geschichtetes Modell in dem das reflektierende Objekt als vordere Ebene I_0 und die Reflexion selber als hintere Ebene I_1 dargestellt ist (Abbildung ??(b)).

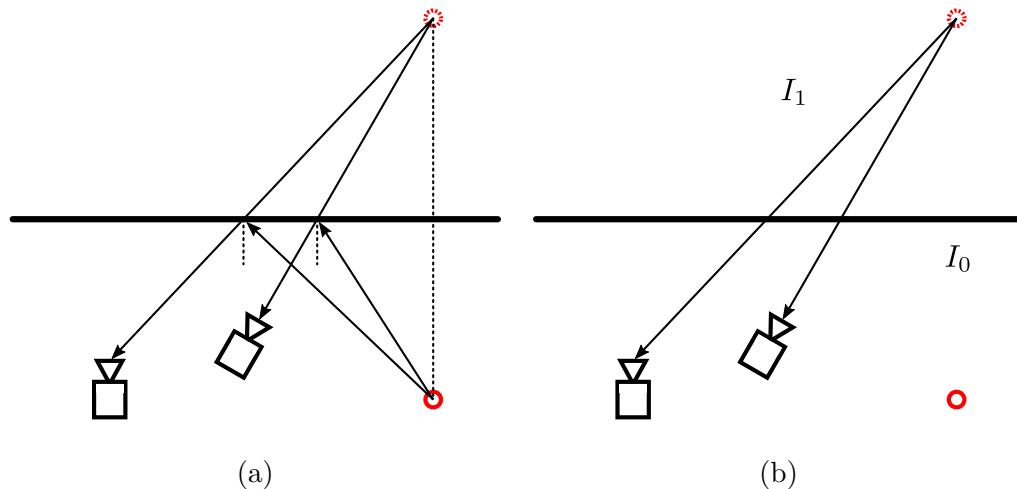


Abbildung 35: (a) beschreibt die physikalische Reflexion nach Hero,
 (b) das von Tsin et al. verwendete geschichtete Modell

Die aufgenommenen Bilder sind demnach eine Zusammensetzung beider Ebenen. Selbiges gilt für nicht planare Flächen sowie Durchsichtigkeit solange Translation und Rotation zwischen den einzelnen Bildern gering sind. Während des Algorithmus werden die relativen Bewegungen der Ebenen zwischen aufeinanderfolgenden Frames anhand der verschiedenen Tiefen analysiert.

Ein weiterer, bereits in Abschnitt 6.1 angeschnittener Punkt ist die Erkennung des Bodens als Hindernis. Dabei stellt sich die Frage, ob eine Erkennung als Hindernis gewünscht ist oder nicht. Im Falle einer geringen Flughöhe wird der Boden ebenfalls als Hindernis erkannt. Dies kann einerseits eine Absicherung dafür sein, dass die durch interne Sensoren ermittelte Flughöhe korrekt ist, andererseits kann es dazu führen, dass der Algorithmus zur Vermeidung von Hindernissen versucht, diesen zu umgehen. Dies könnte unter Umständen da-

zu führen, dass das UAS eine Flughöhe annimmt, welche über der durch die Umgebung gegebenen maximalen Flughöhe liegt. Eine mögliche Lösung wäre die Erkennung des Bodens in Abhängigkeit der aktuellen Höhe zu gestalten. Befindet sich das UAS in einer Höhe in welcher der Boden als Hindernis erkannt wird, so wird dieser nicht weiter betrachtet.

Kapitel 7

Fazit und zukünftige Arbeiten

Im Rahmen dieser Arbeit wurden zwei Methoden zur Erkennung von Hindernissen für unbemannte Flugsysteme in Echtzeit entwickelt. Die grundlegende Funktionsweise beider Methoden unterscheidet sich in der Verwendung ihrer zugrundeliegenden Datenstruktur sowie der Verteilung dieser verwendeten Segmente. Trotz Dessen ist im Lauf der Entwicklung ein deutlicher Unterschied in der Robustheit sowie der Performance zu erkennen gewesen. In zukünftigen Arbeiten sollten beide Methoden in UAS Flugmanövern weiter getestet werden.

Die während der Evaluation erhaltenen Daten deuten auf eine robuste Funktionsweise beider Methoden hin. Die getestete Performance ist, gerade bei zusätzlich skalierten Bildern so gut, dass prinzipiell auch Echtzeitanwendungen bei hoher Geschwindigkeit denkbar sind. Weiterhin erlaubt gerade die hohe Performance zusätzliche Berechnungen zur Verbesserung der Hinderniserkennung.

Zur Verbesserung der Ergebnisse ist zudem die Anwendung alternativer oder die Entwicklung neuer Algorithmen zur Berechnung der Disparitätenkarte denkbar. Dieser sollte sowohl schnelle als auch robuste Ergebnisse liefern, wobei die visuelle Qualität nicht die oberste Priorität ist. Eine Parallelisierung dessen ist dabei zur Verbesserung der Performance zu erwägen.

Weiterhin ist die Integration des entwickelten Systems in das *Robot Operating System* (ROS) zur weitergehenden Verwendung durch SLAM (Simultaneous Localization and Mapping) Algorithmen ein wichtiger Bestandteil zukünftiger Arbeiten. Damit verbunden ist gleichzeitig die Entwicklung eines Systems zur Hindernisvermeidung unter Verwendung der erstellten Punktwolken zur Planung der aktuellen Flugroute. In Kombination mit der aktuellen Position

KAPITEL 7. FAZIT UND ZUKÜNFTIGE ARBEITEN

sowie Rotation der Drohne im Raum ist die darauf folgende Kartographierung gefundener Hindernisse möglich.

Ein weiterer Punkt für zukünftige Arbeiten ist die Verbesserung der Hinderniskennung im Hinblick auf die in Kapitel 6 beschriebenen Limitierungen. Die Lösung dieser ist generell ein wichtiges Kriterium um die Erkennung signifikant zu verbessern. Dabei sollte besonderes Augenmerk auf die Erkennung von homogenen bzw. schwach Texturierten Bereichen gelegt werden. Diese sorgen aktuell für eine Falscherkennung aufgrund von Konflikten innerhalb der Berechnung der Disparitätenkarte. Weiterhin sollten Techniken zur Validierung der gefundenen Hindernisse entwickelt werden.

Literaturverzeichnis

- [Al-Hallak and Hiller, 2015] Al-Hallak, M. and Hiller, H. (2015). Fast depth map estimation from stereo camera systems.
- [Ascending Technologies, 2015] Ascending Technologies (2015).
- [Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer.
- [Birchfield and Tomasi, 1998] Birchfield, S. and Tomasi, C. (1998). A pixel dissimilarity measure that is insensitive to image sampling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(4):401–406.
- [Bouguet, 2001] Bouguet, J.-Y. (2001). Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10.
- [Bradski and Kaehler, 2008] Bradski, D. G. R. and Kaehler, A. (2008). *Learning OpenCV, 1st Edition*. O'Reilly Media, Inc., first edition.
- [Correa et al., 2012] Correa, D. S. O., Sciotti, D. F., Prado, M. G., Sales, D. O., Wolf, D. F., and Osório, F. S. (2012). Mobile robots navigation in indoor environments using kinect sensor. In *Critical Embedded Systems (CBSEC), 2012 Second Brazilian Conference on*, pages 36–41. IEEE.
- [Cyganek and Siebert, 2011] Cyganek, B. and Siebert, J. P. (2011). *An introduction to 3D computer vision techniques and algorithms*. John Wiley & Sons.
- [Geiger et al., 2011] Geiger, A., Roser, M., and Urtasun, R. (2011). Efficient large-scale stereo matching. In *Computer Vision–ACCV 2010*, pages 25–38. Springer.
- [Hirschmuller, 2005] Hirschmuller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. 2:807–814 vol. 2.

- [Hirschmuller, 2008] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):328–341.
- [Kim et al., 2003] Kim, J., Kolmogorov, V., and Zabih, R. (2003). Visual correspondence using energy minimization and mutual information. pages 1033–1040.
- [Kostavelis et al., 2010] Kostavelis, I., Nalpantidis, L., and Gasteratos, A. (2010). Comparative presentation of real-time obstacle avoidance algorithms using solely stereo vision. In *IARP/EURON International Workshop on Robotics for risky interventions and Environmental Surveillance-Maintenance, Sheffield, UK*.
- [Lee et al., 2012] Lee, C.-H., Su, Y.-C., and Chen, L.-G. (2012). An intelligent depth-based obstacle detection system for visually-impaired aid applications. In *Image Analysis for Multimedia Interactive Services (WIAMIS), 2012 13th International Workshop on*, pages 1–4. IEEE.
- [Lucas et al., 1981] Lucas, B. D., Kanade, T., et al. (1981). An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679.
- [MATRIX VISION, 2015] MATRIX VISION (2015). MATRIX VISION GmbH - industrial image processing. <https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html>.
- [Middlebury Univeristy, 2015] Middlebury Univeristy (2015). Middlebury stereo datasets. <http://vision.middlebury.edu/stereo/data/>.
- [Mori and Scherer, 2013] Mori, T. and Scherer, S. (2013). First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1750–1757. IEEE.
- [OpenCV, 2015] OpenCV (2015). OpenCV - open source computer vision. <http://opencv.org/>.
- [Richards et al., 2014] Richards, B., Gan, M., Dayton, J., Quintana, J., Liu, J., and Enriquez, M. (2014). Obstacle avoidance system for uavs using computer vision.
- [Shi and Tomasi, 1994] Shi, J. and Tomasi, C. (1994). Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE.

- [Tsin et al., 2003] Tsin, Y., Kang, S. B., and Szeliski, R. (2003). Stereo matching with reflections and translucency. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–702. IEEE.
- [Zureiki et al., 2008] Zureiki, A., Devy, M., and Chatila, R. (2008). *Stereo Matching and Graph Cuts*. INTECH Open Access Publisher.