

Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Medieninformatik

Echtzeit Hinderniserkennung für unbemannte Flugsysteme unter Benutzung eines Stereo Kamera Systems

Bachelorarbeit

Hagen Hiller
Geboren am 04.06.1992 in Berlin

Matrikelnummer 110514

1. Gutachter: Prof. Dr. Volker Rodehorst
2. Gutachter: TBA

Datum der Abgabe: 25.01.2016

Erklärung

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

Weimar, den 25.01.2016

.....
Hagen Hiller

Zusammenfassung

 Lorem ipsum dolor sit Amet.

Inhaltsverzeichnis

1 Einführung	3
1.1 Motivation	3
1.2 Setup	4
1.3 Ziel der Arbeit	4
2 State of the Art Algorithmen	6
2.1 Aktiv optische Algorithmen	6
2.2 Passiv optische Algorithmen	7
3 Zugrunde liegende Konzepte und Algorithmen	10
3.1 Epipolareometrie	10
3.2 Kamerakalibrierung und Rektifizierung	11
3.3 Stereo Matching	13
3.3.1 Klassifikation	14
3.3.2 Block Matching	15
3.3.3 Semi Global Block Matching	15
3.4 mvStereoVision Framework	19
4 Entwickelte Hinderniserkennungen	22
4.1 Preprocessing	22
4.2 Subimage Detection	24
4.3 Samplepoint Detection	27
5 Limitierungen und Lösungsansätze	29
5.1 Bestehende Limitierungen und Lösungsansätze	29
5.2 Diskussion	31
6 Evaluation	35
6.1 Testsetup	35
6.2 Evaluierung Subimage Detection	37
6.3 Evaluierung Samplepoint Detection	41
6.4 Weitere Versuche	43

INHALTSVERZEICHNIS

6.4.1	Veränderung der <i>SGBM</i> Block Größe	44
6.5	Diskussion	46
7	Diskussion	48
7.1	Gegenüberstellung beider Algorithmen	48
7.1.1	Performance	48
7.1.2	Robustheit	51
8	Fazit und zukünftige Arbeiten	53
	Literaturverzeichnis	54

Kapitel 1

Einführung

1.1 Motivation

Die Forschung im Bereich der Robotik ist ein immer weiter fortschreitender Prozess. Eine besondere Domäne innerhalb dessen ist die autonome Navigation sowie Steuerung von unbemannten Flugsystemen. Dies findet Anwendung in der Erkundung von unbekannten oder gefahrenrächtigen Gebieten. 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. Einerseits sind alle Objekte welche sich in der unmittelbaren Nähe des Systems befinden eine Gefahrenquelle, andererseits wird im Falle Kamera basierter Navigation von einer Bewegung in Blickrichtung ausgegangen. Dies schließt Objekte welche sich außerhalb des Sichtfeldes, also neben oder hinter der Kamera befinden, für die Erfassung durch die Algorithmen aus. Auch sehr weit entfernte Objekte sind prinzipiell nicht als Hindernis anzusehen, wobei sich die maximale Gefahrendistanz relativ zur aktuellen Geschwindigkeit verhält. 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. Weitere Anwendungsbereiche können sowohl komplexerer als auch einfacherer Natur sein. Prinzipiell ist es möglich die entwickelten Algorithmen im Automobil Bereich zu verwenden um beispielsweise Objekte vor oder Hinter dem

Kraftfahrzeug zu erkennen und die Distanz zu berechnen. Weiterhin ist es vorstellbar grobe kartographische Höhen Klassifikationen vorzunehmen, um Bild basiert eine Höhenkarte geographischer Areale zu erstellen. Das jeweilige Anwendungsszenario richtet sich jedoch dabei nach der verwendeten Hardware.

1.2 Setup

Die aktive Entwicklung der Algorithmen erfolgte im Hinblick auf eine Verwendung dieser durch das von Ascending Technologies [Ascending Technologies, 2015] entwickeltes UAS (Unmanned Aerial System) Pelican (Abbildung 1). Es handelt sich dabei um einen Quattrocopter welcher für den Forschungsbereich entwickelt wurde. Dieser ist mit einem Bordcomputer ausgestattet welche die nötige Leistung für die Entwicklung der Algorithmen bereitstellt (3rd Generation Intel Core i7). Weiterhin wurden zwei MatrixVision BlueFOX mv-MLC200wC Industriekameras [GmbH, 2015] als visuelles System verwendet. Die maximale Auflösung beider Kameras beträgt 752×480 bei 60 möglichen Bildern pro Sekunde, in Abhängigkeit verschiedener Parameter (verwendet Verschlusszeit, aufgenommene Bitrate, u.a.). Für den Echtzeit Aspekt des Systems werden beide Kameras im horizontalen und vertikalen Binning verwendet. Dies reduziert die Anzahl der Bildpunkte in beiden Dimensionen auf 376×240 , wodurch der Berechnungsaufwand verringert und die Aufnahmerate der Kameras auf bis zu 170 Einzelbilder pro Sekunde maximiert wird. Für die Entwicklung der Algorithmen wurde die freie Computer Vision Bibliothek OpenCV [OpenCV, 2015b] verwendet. Diese stellt benötigte algebraische Grundoperationen, sowie bestimmte Algorithmen welche im Rahmen dieser Arbeit genutzt wurden, zur Verfügung.

1.3 Ziel der Arbeit

Mithilfe photogrammetrischer Verfahren ist es möglich die räumliche Tiefe aus zweidimensionalen Daten zu berechnen. Dabei gilt die Verwendung Stereobild basierter Daten als ein einfachste Möglichkeit Disparitäten zwischen korrespondierenden Pixeln zu bestimmen um damit einen räumlichen Eindruck der abgebildeten Szenerie zu erhalten.

Vor diesem Hintergrund werden zunächst einige State of the Art Algorithmen beschrieben wobei dabei zwischen aktiven und passiven optischen Systemen unterschieden wird. Diese Einteilung dient einerseits dafür einen Überblick über bereits bestehende Techniken sowie implementierte System zu erhalten, andererseits um auch die Vor und Nachteile der jeweiligen Technik herauszu-

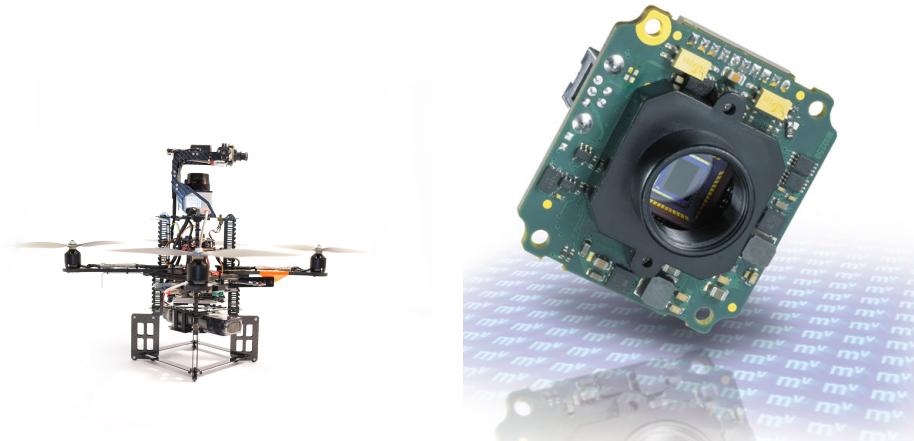


Abbildung 1: AscTec Pelika (links), MatrixVision BlueFOX mv-MLC200wC (rechts)

arbeiten. Kapitel 3 beleuchtet zugrundeliegende Konzepte und Algorithmen, dabei liegt der Fokus primär auf essenziell notwendigen Techniken für die im Rahmen dieser Arbeit entwickelten Systeme. Des Weiteren wird das genutzte Framework sowie dessen Funktionsweise erläutert. Im Anschluss daran beschreibt Kapitel 4 die beiden entwickelten Algorithmen zur Hinderniserkennung detailliert. Anschließend daran erfolgt die Evaluation beider Systeme unter den Gesichtspunkten der Robustheit sowie der Performance jedes einzelnen Algorithmus. Weitere Tests zur zukünftigen Verbesserung der Algorithmen weisen auf welches aktuellen Limitierungen durch einerseits genutzte Konzepte vorliegen. Kapitel 5 erläutert eben diese Limitierungen und gibt Ansätze zur Lösung aus der Fachliteratur sowie eigene Konzepte zur Bewältigung dieser. Die anschliessende Diskussion wertet die im Rahmen der Evaluation in Kapitel 6 erlangten Ergebnisse weitergehend aus und stellt beide Algorithmen aktiv gegenüber. Kapitel 8 zieht ein Résumé aus den Ergebnissen der Arbeit und gibt einen Ausblick auf mögliche zukünftige Arbeiten in diesem Bereich.

Kapitel 2

State of the Art Algorithmen

Eines der wichtigsten und am meisten erforschten Felder 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 Algorithmen 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 Disparity Maps unter Zuhilfenahme stereo-optischer Systeme basiert. Die Kalkulation dieser ist generell sehr rechenaufwändig, liefert aber nach der Auswertung zuverlässige Informationen über die Entfernung sowie Position des aktuellen Objektes.

2.1 Aktiv optische Algorithmen

Aktiv optische Algorithmen beziehen sich auf die aktive Emission von Licht auf die zu rekonstruierende Szene. Dies geschieht indem beispielsweise Muster auf die Szene projiziert werden (Gitter, Streifen, Farben, etc.), mit deren Hilfe es möglich ist, korrespondierende Pixel innerhalb des Bildes zu finden. Jedoch können bei uniformen beziehungsweise einfachen Mustern falsche Bereiche des Musters erkannt werden und somit eine falsche räumliche Repräsentation der Szene. 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. 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 exakte Tie-

fendaten für jedes Einzelbild zu erhalten.

Die von Lee et al. [Lee et al., 2012] verfolgte Methodik bedient sich einer Time of Flight Kamera. Die aufgenommenen Tiefenbilder werden in verschiedene Segmente eingeteilt, welche die verschiedenen Objekte repräsentieren. Sobald etwaige Kanten gefunden werden, werden diese unter dem Ansatz die Objekte so leichter segmentieren zu können entfernt. Anschließend erfolgt eine Tiefenanalyse der nun vorliegenden einzelnen Segmente. Zum Finden der Hindernisse wird nun die Standardabweichung jedes Segmentes berechnet. 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 mit Ihrer Distanz markiert. Ein häufig auftretendes Problem in diesem Ansatz spiegelt der Boden wieder, welcher oft als Hindernis erkannt wird. Ein Mittel zur Unterscheidung ist hierbei ebenfalls die Standardabweichung.

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 Disparity Maps. Dabei werden diese ebenfalls wie bereits in diversen passiven Algorithmen ([Pire et al., 2012], [Kostavelis et al., 2010]) in mehrere horizontale Segmente eingeteilt. Die finale Karte aus 5 Bereichen, von denen 3 als mögliche Richtungen betrachtet werden. Sobald die die minimalste Distanz eines Sektors geringer als 60cm 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.

2.2 Passiv optische Algorithmen

Passiv optische Algorithmen beziehen sich auf die Erfassung dreidimensionaler Informationen aus der Szene ohne eigene Lichtquelle. Dabei sind viele dieser Methoden an das menschliche oder auch tierische Sehen angelehnt. Ein großer Teil der Methodik ist das Prinzip Stereo Vision welches durch die Differenz zweier Bilder derselben Szene einen Eindruck von Tiefe verschafft. Weiterhin kann eine dreidimensionale Rekonstruktion der Umgebung durch Lichteinflüsse, Schattierung 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 Disparity Maps in 3 horizontale Bildbereiche aufgeteilt, welche die möglichen Richtungen des Sy-

stems beschreiben. Für jedes dieser einzelnen Unterbilder wird nun der Durchschnitt der Disparität berechnet, wobei der Bildteil mit dem geringsten Wert auf Hindernisse hinweist, welche sich näher am System befinden. Angesichts der Tatsache, das die Entscheidung darüber, welcher Weg als der sicherste angesehen werden muß, teilweise schwer zu treffen ist, wurde die Threshold Estimation Methode entwickelt. Die Unterteilung der Bilder wird in diesem Algorithmus ebenfalls in drei Regionen vorgenommen. Zunächst werden alle Pixel deren Werte sich über einem vordefinierten Threshold 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 Threshold estimation 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 Thresholds 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.

Einen anderen Ansatz zur Erkennung von Hindernissen geben Richards et al. [Richards et al., 2014] . Der Algorithmus wurde als Hindernisvermeidung für ein UAV entwickelt. Optischer Fluss sowie Feature Tracking bieten dabei die Grundlage für die Erkennung, Vermeidung und Voraussage, welcher Bereich 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 des Objektes 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 Frame verglichen, ob bereits bekannte Punkte innerhalb einer bestimmten Distanz mit denen des aktuellen Frames übereinstimmen. Zur Abschätzung der nächsten Position des Features wird mit Hilfe des Optischen Flusses' der Verschiebungsvektor zwischen beiden Frames 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 die jeweiligen Bilder des 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 Hinder-

nisse) eine stochastische Matrix erstellt welche zur weiteren Planung des Fluges verwendet wird.

Bei der Entwicklung von Algorithmen, welche auf der Erkennung räumlicher Tiefe basieren, bieten stereo-optische Systeme einen großen Vorteil. Durch die Verschiebung 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-Korrespondenz Problemes. Trotzdem 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 et al. [Mori and Scherer, 2013] für die Nutzung eines monokularen Setups. Durch die Verwendung des optischen Fluxes 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 vorhanden 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 et al. setzten bei der Erkennung von Features auf den SURF Algorithmus [Bay et al., 2006], welcher gefundene Features auch nach Veränderung der Größe wieder erkennen kann. 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 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.

Kapitel 3

Zugrunde liegende Konzepte und Algorithmen

Dieses Kapitel beleuchtet dieser Arbeit zugrunde liegende Konzepte und Algorithmen. Zunächst wird das Prinzip der Epipolargeometrie beschrieben welches ein wesentlicher Bestandteil photogrammetrischer Verfahren ist. 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 grundlegend beschrieben, welcher die Grundlage für den im Rahmen dieser Arbeit verwendeten Semi Global Block Matching Algorithmus bildet. Anschließend erfolgt eine Erläuterung des verwendeten Frameworks sowie Details zur Implementierung dessen.

3.1 Epipolargeometrie

Bei der Verwendung der meisten photogrammetrischen Verfahren spielt die Epipolargeometrie eine entscheidende Rolle. Das mathematische Modell beschreibt die geometrische Beziehung zwischen verschiedenen Kamerabildern desselben Objektes, sowie die Beziehung korrespondierender Punkte. Generell gesehen ist die Epipolargeometrie durch das Lochkamera-Modell beschrieben. Dabei liegt jeder Punkt des aufgenommenen Objektes mit dem Projektionszentrum sowie dem Bildpunkt auf einer Geraden. Unter Zuhilfenahme der extrinsischen Orientierung sowie der intrinsischen Parameter der Kamera, ist es möglich den Schnittpunkt zweier Geraden 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.

Abbildung 2 visualisiert diesen Prozess. Gegeben sind die beiden Projektionszentren C_L und C_R der beiden Bildebenen sowie die Bildpunkte P_L und P_R .

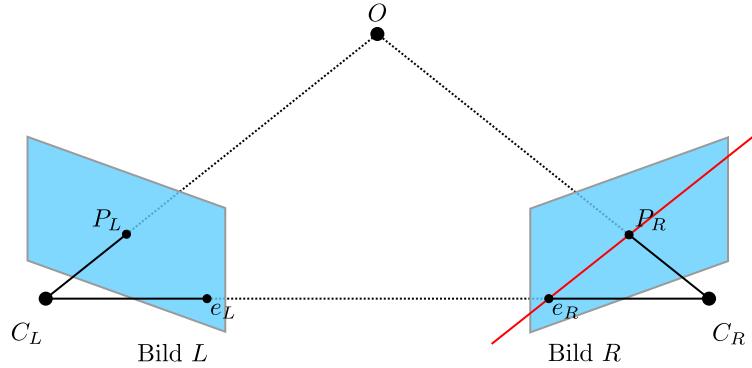


Abbildung 2: Darstellung der Epipolargeometrie.

Der Objektpunkt O bildet im linken Kamerabild auf P_{x_1, y_1} ab. Zunächst ist es nur möglich den Strahl auf dem sich O befindet zu bestimmen. Aufgrund des Wissens der extrinsischen Orientierung der Kameras können mithilfe der Basislinie zwischen den beiden Kameras die beiden Epipole in L und R ermittelt werden. Dabei sind diese durch den Schnittpunkt der Basislinie mit den beiden Bildebenen definiert¹. Somit ist es möglich mithilfe der vorhandenen Epipole e_L, e_R die Epipolarlinie e_l, P_{x_2, y_2} zu bestimmten. Anhand dieser kann im Anschluss die dreidimensionale Position des Objektpunktes O bestimmt werden.

3.2 Kamerakalibrierung und Rektifizierung

Um den Prozess des Stereo Matchings zu vereinfachen und beschleunigen bietet die Rektifizierung der Referenzbilder einen simplen Einstiegspunkt zur Lösung des Stereo-Korrespondenzproblems. Dafür müssen zunächst die intrinsischen und extrinsischen Parameter der Kamera berechnet werden. Dieser Prozess wird als Kalibrierung der Kamera bezeichnet. Dafür werden Bilder eines Kalibrierobjektes aufgenommen. Diese sind meistens einfache Schachbrettmuster mit ungleicher Anzahl an Quadranten, oder auch radiale Objekte mit einem spezifischerem Aufbau. Der Prozess der Kalibrierung zielt darauf ab die „inneren“ Parameter der Kamera zu bestimmen. Dies beinhaltet einerseits die Kameramatrix in welcher Parameter wie der Fokus Punkt(Focal Point), der Bildhauptpunkt(Principal Point) sowie die Brennweite der der Kamera. Wei-

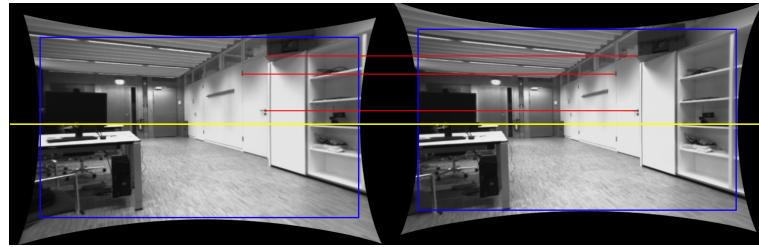
¹Im Falle eines Stereonormalfallen ist ein Schnittpunkt der Basislinie mit den Bildebenen nicht möglich, sodass die Epipole in der Unendlichkeit und parallel zur x-Achse liegen. Dies hat unter anderem den Vorteil, das die Epipolargeometrie bereits bekannt ist, und korrespondierende Bildpunkte nur noch innerhalb einer Pixelreihe gesucht werden müssen.

terhin werden geometrische Verzerrungen des Bildes, parametrisiert und somit nutzbar für die Entzerrung der Bilder in weiteren Schritten. Weiterhin werden bei der Kalibrierung die extrinsischen (äußereren) Parameter der Kamera bestimmt. Zu diesen zählt die Translation (x , y und z) sowie die Rotation der Kamera um die jeweiligen Achsen des Weltkoordinatensystems. 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 Baseline bezeichnet. Diese ist für Prozeduren wie Stereo-Matching unabdingbar. 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 (Objektive, Translation, Rotation).

Folgend aus der Kalibrierung der Kameras beginnt der Prozess der Rektifizierung der Bilder. Dieser kann dabei in zwei Schritte unterteilt werden, zunächst die Berechnung der Region of Interest, anschließend die Rektifizierung jedes aufgenommenen Einzelbildes. Mithilfe der zuvor berechneten intrinsischen und extrinsischen Parameter kann nun die Transformation der beiden Kamerabilder zueinander berechnet werden. Dabei werden die durch die Kalibrierung erhaltenen Objektpunkte unter dem Aspekt der Epipolareometrie paarweise (linkes und rechtes Bild) auf eine Gerade gebracht. Nach der Rektifizierung 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 3 verdeutlicht den Ablauf der Rektifizierung, so ist in Abbildung 3 (c) erkennbar das sich die korrespondierenden Bildpunkte auf einer Epipolarlinie befinden.



(a) Korrespondierende Punkte innerhalb der entzerrten Bilder



(b) Korrespondierende Punkte in den rektifizierten Bildern



(c) Korrespondierende Punkte in den geschnittenen und skalierten rektifizierten Bildern

Abbildung 3: Resultat der Rektifizierung

3.3 Stereo Matching

Das grundlegende Konzept des Stereo Matchings beschreibt das Finden korrespondierender Punkte in zwei simultan aufgenommenen Bildern. Die Position der beiden Kameras ist dabei leicht versetzt, um einen jeweils anderen Blickwinkel auf die Szene zu erhalten. Mithilfe der verschiedenen Perspektiven können Disparitäten (Differenz der Projektion eines Objektes vom linken zum rechten Bild) zwischen korrespondierenden Pixeln berechnet werden. Die dabei erhaltenen Tiefeninformationen eines Objektes sind mit der errechneten Disparität sowie der relativen Position der Kameras und deren intrinsischen Parametern verbunden. Im Laufe dieses Prozesses kommt es zu zwei wesent-

lichen 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.

3.3.1 Klassifikation

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 durch deren lokale Umgebung bestimmt. Der Referenzpunkt ist dabei das Zentrum eines Bereiches in welchem das Matching ausgeführt wird. 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), etc.. Jedoch können aufgrund fehlender Beschränkungen verzerrte 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.

Globale Methoden:

Bei globalen Stereo Matching Algorithmen wird ein globales Modell der zu betrachtenden Szene erstellt sowie eine ebenfalls globale Kostenfunktion definiert, welche minimal gehalten werden soll. Dabei werden im Gegensatz zu lokalen Methoden die Matches in einer Pixelreihe und die des gesamten Bildes miteinander verglichen. Zusätzlich zu den benachbarten Pixeln werden hier ebenfalls die Matches der angrenzenden Pixel betrachtet. Zur Vereinfachung dieses Vorgangs betrachten einige Algorithmen nur die Epipolareometrie, wobei 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.

3.3.2 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 mathematischer Berechnungen (NCC, SAD, etc.) auf Korrespondenz untersucht. Dabei reduziert die Epipolaregeometrie diesen Prozess auf ein eindimensionales Problem indem ein Block im linken Bild mit einem Block im rechten Bild auf der Epipolarlinie des rechten Bildes pixelweise verglichen wird. Das eigentliche Matching erfolgt über die Berechnung eines Blockes mithilfe einer Energiefunktion.

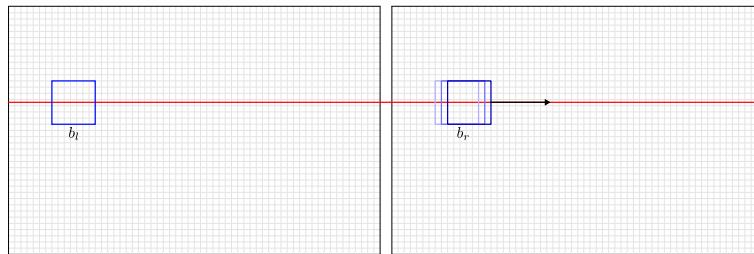


Abbildung 4: Visualisierung des Block Matching Algorithmus

Diesen Prozess veranschaulicht Abbildung 4 . Durch den Fokus auf einzelne Pixelreihen und deren unmittelbare Umgebung ist dieses Verfahren wesentlich schneller, aber auch ungenauer als globale Matching Algorithmen.

3.3.3 Semi Global Block Matching

Das im Rahmen dieser Arbeit verwendete Verfahren zur Berechnung von Disparity Maps ist der in der freien Computer Vision Library implementierte Semi Global Block Matching Algorithmus. Diese leicht abgewandelte Implementierung des Semi Global Matching Algorithmus von Hirschmüller et. al [Hirschmüller, 2005] zeichnet sich sowohl durch seine Genauigkeit, als auch durch seine performante Berechnung aus. Unter Zuhilfenahme dieses Verfahren ist es möglich, Disparity Maps in Echtzeit mit sehr guter Framerate 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 Parameter dessen vorgenommen.

SGM:

Die grundlegende Idee des Semi-Global Matching Algorithmus besteht in dem pixelweisen Matching von Mutual Information ². Die globale zweidimensionale

²Mutual Information(MI), zu deutsch Transinformationen, beschreiben den statistischen

Smoothness Einschränkung wird dabei mithilfe mehrerer eindimensionaler Einschränkungen erzeugt. Ausgangsvoraussetzung dafür sind verschiedene Bilder derselben Szene mit vorhandener und bekannter Epipolargeometrie. Zunächst 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. Die eigentliche Berechnung dieser erfolgt dann mit Hilfe von Mutual Information berechnet.

$$MI_{I_1, I_2} = \sum_p mi_{I_1, I_2}(I_{1p}, I_{2p}) \quad (3.1)$$

$$mi_{I_1, I_2}(i, k) = h_{I_1}(i) + h_{I_2}(k) - h_{I_1, I_2}(i, k) \quad (3.2)$$

Zur Berechnung der MI mithilfe der in 3.1 und 3.2 dargestellten Formeln wird zunächst eine Initial Disparity Map benötigt. Diese wird nach Kim et al's [Zureiki et al., 2008] Ansatz zufällig gewählt, um die Kosten berechnen zu können. Dies ist insbesondere für iterative Verfahren geeignet, da andernfalls die Berechnung verlangsamt werden könnte. Zur Steigerung der Performanz wird die Disparity Map zunächst nur auf halber Auflösung berechnet, wodurch der Rechenaufwand um den Faktor 2·3 reduziert wird. Zur Vermeidung falscher Kostenberechnungen durch auftretende Noise innerhalb des Bildes nimmt der Algorithmus weitere Einschränkungen vor. Dabei werden die benachbarten Disparitäten in die Berechnung 3.3 mit einbezogen.

$$foo = foo \quad (3.3)$$

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 alle Richtungen zur aktuellen Disparität mit einbezogen (siehe Abbildung 5).

Zur Berechnung valider Werte 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. Dabei gilt: "Je mehr Übereinstimmungen in den Kosten, desto wahrscheinlicher ist es, dass D zur Intensität I gehört".

SGBM:

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

Zusammenhang zweier Zufallsgrößen. Genauer gesagt beschreiben sie die Höhe des Informationsgehaltes einer Zufallsvariable aus anderen zufällig gewählten Werten.

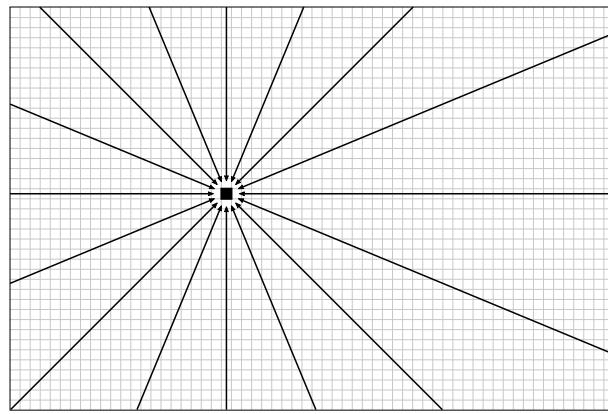


Abbildung 5: Darstellung der verschiedenen Richtungen

von Disparity Maps. Die Grundlage dafür bietet Hirschmüller et al.'s SGM [Hirschmüller, 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 Pre und Post Processing Elemente des *StereoBM* [OpenCV, 2015a] werden verwendet.

Dies erlaubt dem Algorithmus eine schnelle Berechnung der Disparitäten auf einem qualitativ hochwertigen Niveau. Der geringe Verlust an Qualität kann in Anbetracht der Berechnung in Echtzeit vernachlässigt werden.

Abbildung 6 stellt die originalen Bildern 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 sowohl qualitativ hochwertige Berechnung von Tiefenkarten. Kleinere qualitative Einbußen können daher bei steigender Performanz akzeptiert werden.

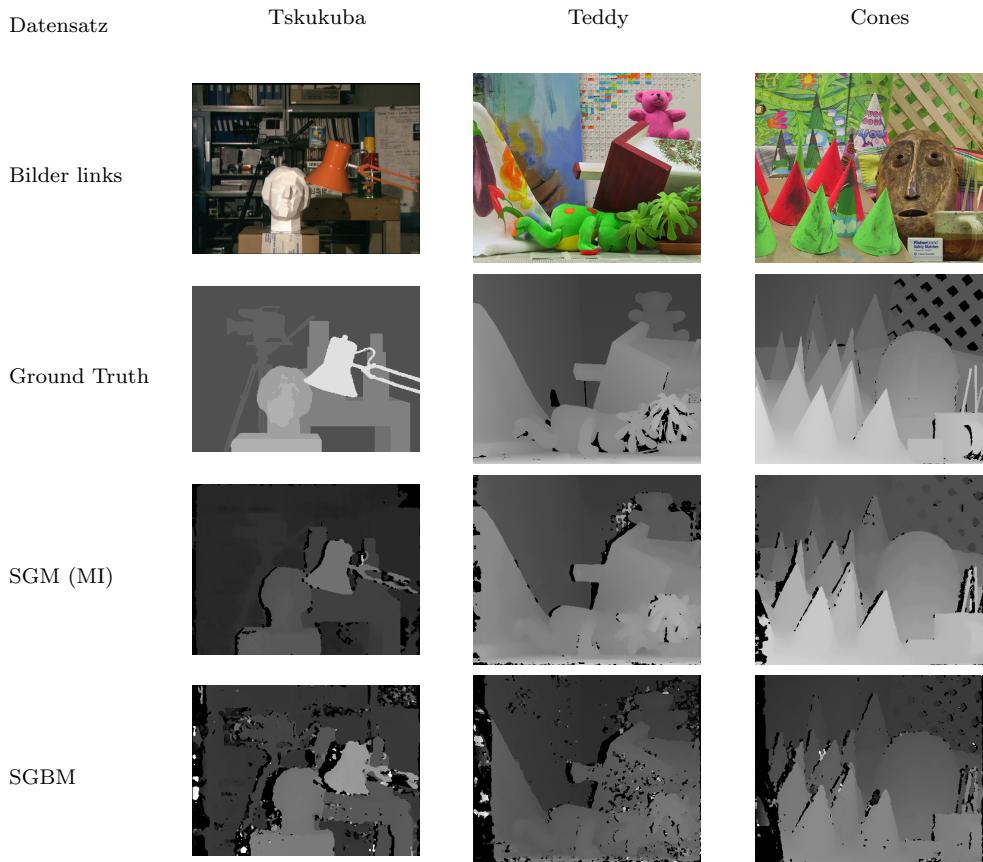


Abbildung 6: Vergleich zwischen Ground Truth Bildern sowie dem SGM(2005) und OpenCV's SGBM

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, dies ist vor allem dann von Nöten wenn die Tiefenberechnung nach unten hin abgegrenzt werden soll. Er beschreibt also die minimale zu erkennende Disparität

- ***numDisparities:***

Mit Hilfe der Number of Disparities *numDisparities* wird die Breite des Matchingbereiches festgelegt. Er definiert wie viele Pixel im linken Bild auf Korrespondenz im rechten Bild analysiert werden sollen.

In Abhängigkeit der Größe des Parameters entsteht bei der Berechnung der Tiefenkarte ein Bereich in welchem keine Informationen enthalten sind. Die damit verbundene weitere Verfahrensweise wird im Abschnitt 4.1 „Preprocessing“ 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 Disparity Map. Dabei gehen jedoch auch Informationen verloren. Ist der Wert niedrig so ist es unter Umständen schwerer homogene Flächen korrekt zu Matchen und es treten mehr nicht-korrespondierende Bereiche auf.

3.4 mvStereoVision Framework

Das verwendete Framework zur Bildaufnahme und Disparity Map Berechnung wurde im Rahmen des Projektes “SLAM for UAV” entwickelt. Dieses bedient sich der von Matrix Vision zur Verfügung gestellte Library [GmbH, 2015] zur Kommunikation mit den Kameras, sowie OpenCV [OpenCV, 2015b] 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 anschliessend mithilfe der durch die Kalibrierung ermittelten Parameter entzerrt und rektifiziert und liegen so zur weiteren Verarbeitung bereit. Die Aufnahme der Bilder erfolgt dabei in separaten Threads um unabhängig von weiteren Berechnungen Bilder aufzunehmen. Die Berechnung der Disparity Map erfolgt ebenfalls in einem eigenen Thread. Sofern eine neue Disparity Map vorliegt wird innerhalb des Hauptprogramms mit Hilfe eines Wahrheitswertes darüber informiert das diese vorliegt. Der gesamte Prozess wird in Abbildung 7 visualisiert.

Distanzberechnung:

Zugrundeliegend bei der Distanzberechnung ist die Reprojektionsmatrix Q welche im Rahmen der Rektifizierung berechnet wird und wie folgt aufgebaut ist.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{-1}{T_x} & \frac{C_x - C'_x}{T_x} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 0 & f \\ 0 & 0 & a & b \end{bmatrix} \quad (3.4)$$

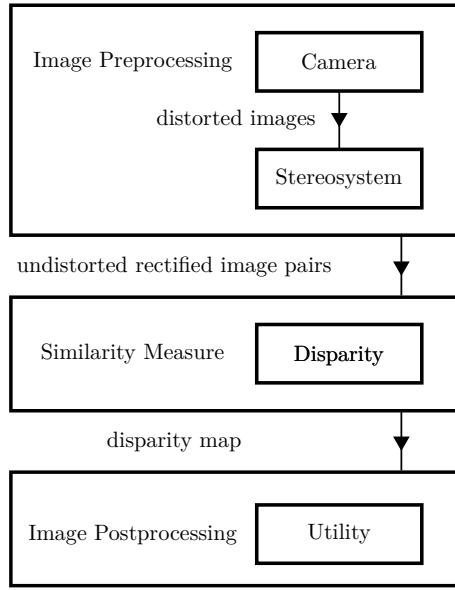


Abbildung 7: Pipeline zur Bildaufnahme, Disparity Map Berechnung sowie weiteren Verarbeitung

Dabei beschreiben $-C_x$ und $-C_y$ die beiden negativen Koordinaten des Bildhauptpunktes der linken Kamera sofern dies als Referenz vorlag, der Parameter f die Brennweite der linken Kamera. T_x ist die horizontale Translation zwischen beiden Kameras. Die in der Q-Matrix enthaltenen Parameter sind mit Ausnahme von C'_x alle die der linken Kamera. Es wird angenommen das sich die beiden Strahlen der Bildhauptpunkte in der Unendlichkeit schneiden woraus sich der Parameter 0 in der unteren rechten Ecke berechnet. Zur Einfacheren Darstellung und Erläuterung werden die Baseline 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 ist in Formel 3.5 abgebildet.

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \cdot \begin{bmatrix} I_x \\ I_y \\ d(x, y) \\ 1 \end{bmatrix} \quad (3.5)$$

$$pointcloud(x, y) = \left[\frac{X}{W}, \frac{Y}{W}, \frac{Z}{W} \right]$$

Die eigentliche Berechnung der einzelnen Parameter ergibt sich somit aus der Multiplikation des transponierten Vektors, welcher sowohl die Koordinaten des

Pixels als auch die Disparität dessen enthält, und der Reprojektions Matrix. Die dabei benötigten Operationen werden im folgenden genauer beleuchtet.

$$\begin{aligned} X &= I_x - C_x \\ Y &= I_y - C_y \\ Z &= f \\ W &= d \cdot a + b \end{aligned} \tag{3.6}$$

Kapitel 4

Entwickelte Hinderniserkennungen

Im Rahmen dieser Arbeit wurden zwei Systeme zur Erkennung von Hindernissen in Echtzeit entwickelt. Diese richten sich nach den in Kapitel 3 erläuterten Algorithmen und Konzepten. Anhand dieser ist es möglich aus den beiden Bildern des Stereo Systems für jeden Frame die Disparity Map zu berechnen, welche im Anschluss daran in mehreren Schritten zunächst so angepasst wird, dass nicht verwertbare Bereiche der Tiefenkarte entfernt 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 Preprocessing

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

Region of Interest der Disparity Map:

Zu Beginn des Algorithmus muss der durch die Baseline verursachte nicht matchbare Bereich entfernt werden. Dies geschieht nach der Wahl der in (3.3.3) beschriebenen Parameter der SGBM. Dieser sogenannte Pixelshift berechnet sich aus der Hälfte der Anzahl der zu berechnenden Disparitäten. Sollte dieser einen ungeraden Wert ergeben so wird er um 1 erhöht. Generell ist der Pixelshift so gewählt das die Dimensionen der Disparity Map durch 8 teilbar sind. Dieser Wert hat sich während der Entwicklung als robuster Wert erwiesen.

Weiterhin dient es zur späteren Verteilung der Samplepoints.

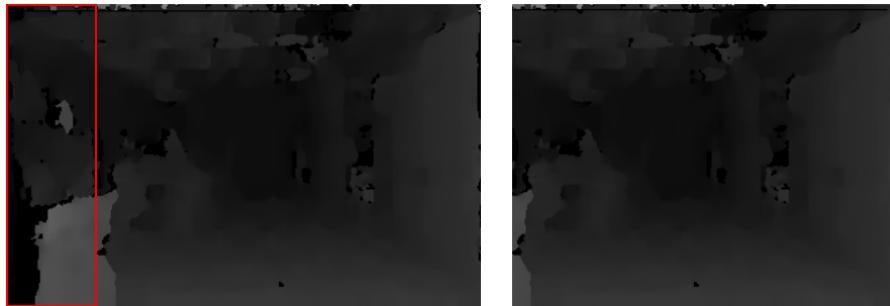


Abbildung 8: Darstellung der Disparity ROI. Der rote Bereich markiert den Pixelshift

Abbildung 8 zeigt den Pixelshift in der finalen Disparity Map. Der *SGBM* berechnet zwar Werte innerhalb des Bereiches welcher nicht zu erkennen ist, 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 während der Erkennung mit den reinen Disparitäten gearbeitet. Da die Disparitäten in Abhängigkeit der Bildgröße sowie der Q-Matrix berechnet werden ergibt kann kein fest definierter Disparitätenwert für eine Distanz bestimmt werden. Für diesen Prozess wird die in Abschnitt 3.4 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 Algorithmen Verwendung bei welcher die metrischen Angaben der Erkennungsreichweite in Disparitäten zurückgerechnet werden. Als Referenz wir dazu die Position des Bildhauptpunktes berechnet welcher sich im Ursprung des Weltkoordinatensystems befindet. 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. Dies ist in beiden Algorithmen derselbe Schritt.

4.2 Subimage Detection

Das grundlegende Funktionsprinzip der *Subimage Detection* ist grob an die von Kostavelis et al. vorgestellten Algorithmen angelehnt. Auch bei diesen werden die berechneten Disparity Maps in Segmente eingeteilt von welchen in jedem Frame der Mittelwert errechnet wird.

Zu Beginn des Algorithmus, nach der vorherigen Bearbeitung der eigentlichen *region of interest*, wird während der Initialisierung die Segmentierung der Disparity Map festgelegt. Dabei wird für jedes Segment ein eigenes Subimage erzeugt. Diese definieren eine *region of interest* (ROI) innerhalb einer Matrix. Subimages selber halten lediglich die Positionsinformationen (obere linke und untere rechte Ecke des definierten Rechtecks), Mittelwert der ROI, die ROI selbst sowie den Mittelpunkt des Rechtecks zur späteren Pointcloud Generierung, wie Abbildung 9 aufzeigt.

Subimage
+tl: Point
+br: Point
+roi_center: Point
+value: float
+Subimage(tl:Point,br:Point): Subimage
+calculateSamplepointValue(d_Map:Mat): float

Abbildung 9: Subimage Klasse

Für die Unterteilung der Disparity Map 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 noch einmal genauer betrachtet wird (siehe Abbildung 10). Um die Hindernisse innerhalb der Segmente zu erkennen wurde auf die Berechnung des Mittelwertes dieser gesetzt. Einerseits, da die Berechnung des

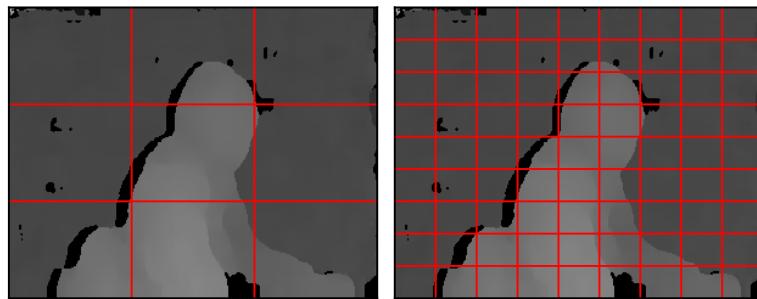


Abbildung 10: Intiale Segmentierung der Disparity Map sowie die weitere Unterteilung zur genaueren Bestimmung der Werte der Subimages

Mittelwertes unter Betrachtung des Echtzeit-Aspektes eine ressourcensparende und schnelle Operation ist, andererseits weil jeder Pixel des Bildes in das Endresultat mit einfließt. Jedoch musste die Berechnung auf das Szenario angepasst werden, da Berechnung des Medians aller Werte zu Verzerrungen geführt hätte. Bei der Kalkulation von Disparity Maps werden Bereiche welche nicht gematcht werden können, sei es aufgrund von fehlenden Informationen im Referenzbild oder homogener Texturen in der Szene, als negativer Wert ausgedrückt. Berechnet man nun den Mittelwert unter Betrachtung positiver und negativer Werte, so entspricht dies zwar dem definierten Term Median, verfälscht allerdings das in diesem Anwendungsbereich erwünschte Ergebnis. 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.

Algorithm 1 Berechnung des Disparity Medians

```

1: procedure CALCMEANDISPARITY(submatrix)
2:   elements_number  $\leftarrow 0$ 
3:   elements_sum  $\leftarrow 0$ 
4:   for value in submatrix do
5:     if value > 0 then
6:       elements_sum  $\leftarrow$  elements_sum + value
7:       elements_number  $\leftarrow$  elements_number + 1
8:     end if
9:   end for
10:  return elements_sum/elements_number
11: end procedure
```

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.

Die eigentliche Hinderniserkennung erfolgt in jedem Frame. Wobei der Term Frame hierbei nicht nach einem von der Kamera aufgenommenen Bild, sondern eine neue Disparity Map definiert. Da die Bildgröße pro Einzelbild stetig ist besteht keine Nötigkeit die einzelnen Segmente erneut zu generieren. Es genügt also die Mittelwerte eines jeden Subimages anhand der neuen Disparity Map zu aktualisieren. Daraufhin wird geprüft ob sich einer oder mehrere der Subimage Mittelwerte innerhalb der Gefahrenzone befinden. Die Gefahrenzone wird zur Initialisierung der Erkennung berechnet. Mit Hilfe der in Abschnitt 4.1 beschriebenen Invertierung der Distanzberechnung müssen zu Beginn die minimale sowie maximale Distanz der Erkennung auf einen Tiefenwert zurückgerechnet werden. Diese Rückrechnung ist ausschlaggebend für die schnelle Erkennung der Tiefe. Im Falle einer weiteren Segmentierung müsste die Berechnung der Distanz als Teil der Weltkoordinate für jedes Subimage einzeln erfolgen, was gerade bei der Samplepoint Detection zum Tragen kommt, sollen die Distanzen für ca. 6000 Punkte (bei voller Auflösung in Abhängigkeit der verwendeten Parameter des SGBM) berechnet werden.

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
```

Die in Algorithmus 2 berechnete Pointcloud ist in diesem Fall ein Matrix mit der Größe der ursprünglichen Tiefenkarte. Dies ist für die Berechnung der

dreidimensionalen Koordinate von Nöten. Mithilfe des Mittelpunkte der Subimages wird für jedes dieser die jeweilige 3D-Koordinate (siehe Abschnitt 3.4) berechnet und an der Position des Mittelwertes in die Punktfolke 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 Pointclouds erkannter Hindernisse einfach analysiert werden um etwaige Vermeidungsstrategien zu entwickeln. Weiterhin ermöglicht das ply Format die Visualisierung jedes Frames in denen Hindernisse erkannt wurden.

4.3 Samplepoint Detection

In Hinblick auf aktive optische Algorithmen zur 3D-Rekonstruktion und damit verbunden 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 Disparity Map betrachtet werden müssen.

Ebenso wie in der in Abschnitt 1 vorgestellten *Subimage Detection* bedient sich die Samplepoint Detection einer vorverarbeiteten Disparity Map zur Hinderniserkennung. Bei der Initialisierung wird die Anzahl der zu berechnenden Samplepoints festgelegt, dabei richtet sich der Wert an der Anzahl der Reihen und Spalten der Disparity Map Matrix um eine äquidistante Verteilung gewährleisten zu können. Der hier verwendete Standart Faktor ist 8 (siehe Formel REF). Dies sorgt unter anderem für eine gleiche Anzahl von Samplepoints im gebinnten und ungebinnten Aufnahmemodus, da die jeweilige Menge relativ zur Bildgröße ist. Auf eine Verteilung der Messpunkte am direkten Rand der Disparity Map wurde bewusst verzichtet, da dieser aufgrund der Rektifizierung eher Fehler aufweist als Bereiche in der Mitte des Bildes.

Ein Samplepoint umfasst im entwickelten System entweder einen Pixel oder einen Zusammenschluss mehrerer Pixel. Die dabei festgelegte Ausgangskoordinate entspricht dabei entweder dem eigentlichen Punkt, oder dem Mittelpunkt einer festgelegten Region of Interest. 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 Samplepoints S mit dem Zentrum C ergibt sich im Falle eines Radius $r = 3$ aus dem Mittelwert des Quadrats Q mit den Eckpunkten

$Q_{tl} = (C_x - r, C_y - r)$ und $Q_{br} = (C_x + r, C_y + r)$. Dies erhöht zwar die Anzahl der zu betrachtenden Pixel welche jedoch, in Abhängigkeit vom gewählten Radius, geringer ist als die Gesamtzahl aller Bildpunkte. Dieser Prozess wird in Abbildung 11 verdeutlicht.

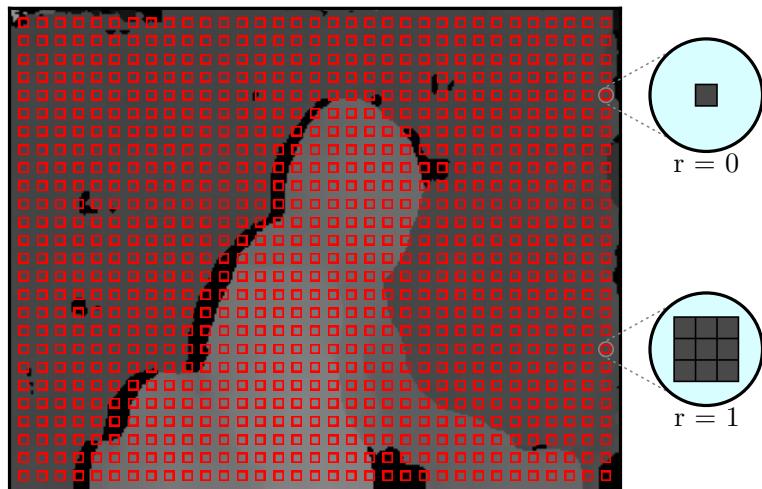


Abbildung 11: Darstellung der Samplepoints mit verschiedenen Radien

Nachdem zunächst alle Samplepoints initialisiert wurden, werden die jeweiligen Werte pro Frame aktualisiert. Die eigentliche Erkennung erfolgt dabei einerseits durch die Erstellung einer Pointcloud welche ebenfalls die selben Dimensionen wie die zugrunde liegende Disparity Map besitzt. Erweiternd dazu lässt sich mithilfe der jeweiligen Punkte das als aktuell am gefährlichsten einzustufende Hindernis sowie dessen Position bestimmen. Dabei werden Samplepoints innerhalb der Gefahrenzone zunächst nur auf Bildebene betrachtet. Zunächst wird die Liste an Samplepoints der errechneten Disparität sortiert. Da generell Hindernisse direkt vor dem UAV als am kritischsten zu betrachten sind, berechnet der Algorithmus den zweidimensionalen Abstand zwischen Bildhauptpunkt und den Samplepoints mit der geringsten Disparität.

Kapitel 5

Limitierungen und Lösungsansätze

In der Erkennung von Hindernissen mithilfe von passiv optischen Systemen können verschiedenste Faktoren der Grund für eine fehlerhafte Erkennung sein. Sei es die Berechnung einer Disparity Map 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.

Das folgende Kapitel beleuchtet noch bestehende Limitierung in der Erstellung der Disparity Map sowie der Erkennung von Hindernissen. Weiterhin werden diese in Abschnitt 5.2 diskutiert. Zudem werden sowohl eigene, als auch literarisch belegte Lösungsansätze präsentiert.

5.1 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 UAV 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. Daher sollte ausgewertet werden ob es sich bei erkannten Hindernissen auch um solche handelt. Ein Ansatz zur Vermeidung solcher Falscherkennungen wäre,

die Objekte insofern zu verfolgen, dass für jeden Frame (in Abhängigkeit der zugrundeliegenden Framerate) überprüft wird, ob im vorherigen Frame bereits ein Objekt gefunden wurde. Erst nachdem dies sichergestellt wurde, wird daraufhin eine Warnung ausgegeben. Zeitgleich wäre es möglich, dass Informationen wie die Eigenbewegung der Drohne sowie die Information, ob sich das erkannte Objekt selber im Raum bewegt, verloren gehen.

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 einerseits einem Pixel an einer weißen Wand kein zugehöriger Pixel zugeordnet werden, andererseits ist es auch möglich, dass falsche Bildpunkte miteinander gematcht werden, welche wiederum eine hohe Disparität zur Folge haben. In diesem Fall wird ein Hindernis erkannt, obwohl sich keines an dieser Position befunden hat. Abbildung 12 zeigt deutlich, in welchen Bereichen der *SGBM* aufgrund fehlender Textur keine Korrespondenzen finden konnte.

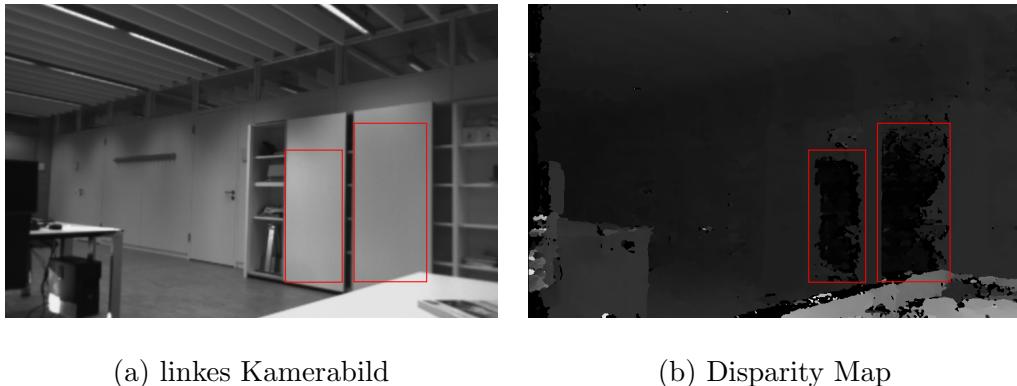


Abbildung 12: Konflikt in der Berechnung der Disparität bei nicht texturierten Flächen. Dabei ist (a) das linke aufgenommene Kamerabild und (b) die dazu gehörige Disparity Map. Die in rot markierten Flächen sind das Resultat homogener Texturen.

Auch spiegelnde Flächen sind eine optische Problemstellung. Aufgrund der Projektion eines anderen Bereiches kommt es zu falschen Informationen innerhalb der Disparity Map. 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 in

diesem Fall eine weiter entfernte Distanz wahrgenommen anstelle der des eigentlichen Objektes. 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 Kameras 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 die Erkennung des Systems auf den Gebrauch in Innenbereichen konzipiert ist, gibt es verschiedenste Konflikte bei der Anwendung in Außenbereichen. Zum einen ist die Verschlusszeit der Kameras 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 Kameras immer gleich sein. Weiterhin ist die Erkennung des Bodens ein zu betrachtender Aspekt. Dieser ist bei aktueller Berechnung ein Teil der zu betrachtenden Hindernisse, was einerseits seine Begründung hat, andererseits auch bei niedrigen Flughöhen zu einer fehlerhaften Erkennung als Hindernis führt.

5.2 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, dass in mindestens zwei aufeinanderfolgenden Frames kein Objekt im selben Subimage/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 würde dabei ein zufälliges Punktmuster auf die Szene projiziert werden, durch welches es dem Matching Algorithmus möglich ist, korrespondierende Punkte innerhalb eines nicht texturierten Bereiches zu finden. Die Zufälligkeit der Punktwolke 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 Aussenbereichen ist es prinzipiell schwer, Hindernisse aufgrund anderer Lichtverhältnisse ausfindig zu machen. Für diesen Anwendungsfall würde sich eine Berechnung homogener Flächen auf Matching Ebene anbieten.

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 Thresholds 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 benachbarten Pixel 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. Zuletzt werden beide Disparity Maps auf ihre Konsistenz geprüft.

Der von Tsin et al. [Tsin et al., 2003] entwickelte Stereo Matching Algorithmus kann sowohl zwischen Reflexion als auch Transparenz unterscheiden. Dabei wird das grundlegende Problem wie folgt beschrieben: Aufgrund des Reflexi-

onsmodells nach Hero (Abbildung 13(a)) welches besagt das sich Einfalls- sowie Ausfallswinkel, gemessen an der Oberflächennormale gleichen, ist es trivial, dass die Position einer Reflexion eines Szenepunktes unabhängig vom aktuellen Sichtfenster ist, da sich die Reflexion 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 13(b)).

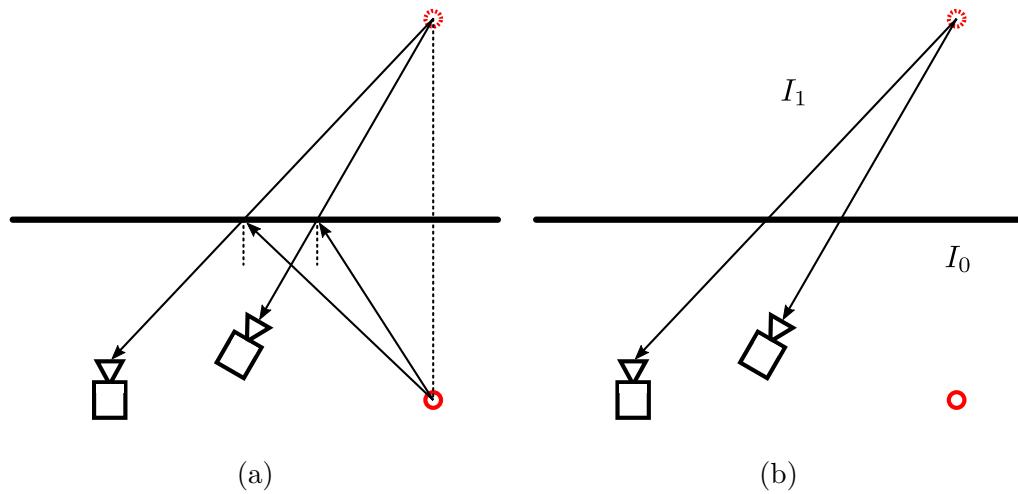


Abbildung 13: (a) beschreibt die physikalische Reflexion nach Hero, (b) das von Tsin et al. verwendete geschichtete Modell

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

Ein weiterer, bereits in Abschnitt 5.1 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, dieses zu umgehen. Dies könnte unter Umständen dazu führen, dass das UAV 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. Sollte sich das UAV unterhalb der zur Erkennung des Bodens nötigen Flughöhe befinden, so wird der Boden nicht weiter betrachtet. Sofern die Flughöhe die Erkennung des Bodens ausschließt wird dieser entfernt.

Kapitel 6

Evaluation

Um die Funktionsweise beider Algorithmen zu belegen wurden beide Methoden einerseits auf ihre Robustheit und andererseits auf ihre Performance getestet. Im folgenden Kapitel wird zunächst das zum Testen verwendete Setup beschrieben. Anschliessend erfolgt die Auswertung der erlangten Testergebnisse. Zuletzt werden weitere Tests durchgeführt welche das System kritischen Situationen testen soll.

6.1 Testsetup

Bei der Durchführung der Tests befinden sich die Kameras statisch im Raum. Die zu erkennenden Hindernisse werden innerhalb und ausserhalb der zu erkennenden Reichweite platziert, wobei die Ausrichtung der Hindernisse teils zufällig, teils bewusst an kritischen Positionen erfolgt, um ein reales Anwendungsszenarien passend zu simulieren. Ein aufgenommenes Testset besteht dabei aus beiden Bildern der Kamera, der normalisierten Disparity Map sowie eine komplette Pointcloud dieser um etwaige Fehler der Algorithmen leichter erkennen zu können, sowie den gelogten Pointclouds der Hinderniserkennung. 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 welchem das System auch aufgrund der hohen Frame rate der Erkennung angewendet werden kann. Eine Erweiterung dessen auf beispielsweise 2.0 Meter wurde nicht durchgeführt, da der Algorithmus auch bei großen Entfernung robuste Werte in der Distanzberechnung liefert [Al-Hallak and Hiller, 2015]. Das dabei erreichte Sichtfeld nach der Anwendung der ROI auf die Disparity Map (siehe 4.1) beträgt 50° auf horizontaler Achse und 38° vertikal. Dies ist in Abbildung 14 visualisiert.

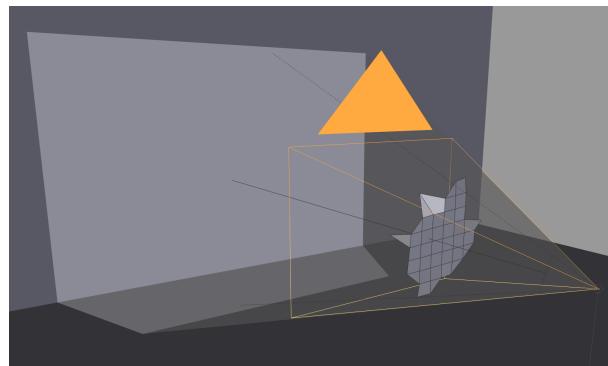


Abbildung 14: Im Bild ist das nach der Rektifizierung sowie Beschniedung der ROI erhaltene Sichtfeld visualisiert. Das Mesh innerhalb des Kamera Frustums repräsentiert die gerenderte Pointcloud eines Hindernisses

Ein aufgenommenes Testset besteht aus jeweils 12 Testbildern. Für jede Methode wurden drei verschiedene Hindernisgrößen getestet, groß, mittel und kleine Hindernisse. Anhand dieser wird ausgewertet welche minimale, maximale sowie mittlere Disparität, und daraus resultierende Distanz erkannt wird.

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. Des Weiteren wird die Zeit für die Hinderniserkennung eines Frames untersucht um eine durchschnittliche Zeit für die Erkennung sowie die daraus resultierende Framerate zu ermitteln. Dies geschieht einerseits durch die Erkennung eines Hindernisses, welches sich über das gesamte Bild ausbreitet, andererseits für nur ein Teilelement jeder Erkennungsmethode (Subimage, Samplepoint).

Zudem wird geprüft inwiefern die Algorithmen mit Limitierungen des *SGBM* umgehen können. Dazu zählen die Erkennung bei spiegelnden, reflektierenden und durchsichtigen Flächen, sowie die Erkennung schwach texturierter Hindernisse.

6.2 Evaluierung Subimage Detection

Hinsichtlich der Robustheit werden beide Algorithmen nach demselben Schema untersucht. Der in Abschnitt 6.1 beschriebene Testablauf erwähnt 3 verschiedene Hindernisgrößen. Tabelle 6.1 zeigt sowohl die Maße der Hindernisse als auch deren Fläche in cm/cm^2 auf.

Hindernis	Radius	Länge	Breite	Fläche
Groß	-	53.5	43.0	2300.5
Mittel	-	16.0	14.5	232.0
Klein	2.5	-	-	19.6

Tabelle 6.1: Maße der verschiedenen genutzten Hindernisse

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. Somit entspricht die gemessene Distanz der mittleren Distanz welche sich aus allen gefundenen Bereichen eines Bildes zusammensetzt. Während der Aufnahme sind die jeweiligen Einzelbilder der linken und rechten Kamera sowie die berechnete Disparity Map sichtbar. Weiterhin ist ein weiterer Anzeigemodus verfügbar welcher die gefundenen Hindernisse innerhalb der Tiefenkarte markiert (siehe Abbildung 15). Weiterhin kann jede erstellte Hindernis-Pointcloud im Nachhinein gerendert werden um einen Überblick über die Ergebnisse des Algorithmus zu erhalten.

Großes Hindernis:

Im ersten Test wurde das große Hindernis gewählt, dabei wurde mithilfe der dargestellten Hindernisse auf der Disparity Map überprüft ob sich die erkannten Hindernisse innerhalb des Gefahrenbereichs befinden. Um auch kritische Bereiche zu testen wurde das Hindernis zudem zu Teilen außerhalb dieser platziert. Aus den 12 aufgenommenen Bildern des Testsets ergaben sich die in Abbildung 16 (a) sichtbaren Ergebnisse.

Wie aus dieser zu erkennen wurde in nahezu allen Frames das Hindernis richtig erkannt. Die berechneten Distanzen stimmen mit den real gemessenen überein. Minimale Abweichungen können in diesem Fall als Messungenauigkeit ignoriert werden, da der Mittelpunkt zwischen der minimalen und maximalen Objektdistanz kein statischer Punkt ist. Es wurden zwar alle Hindernisse erkannt, jedoch nicht der gesamte Bereich den sie umfasst haben. Wie Abbildung 17 zeigt wurden der obere und untere Bereich des Hindernisses im 5. Frame nicht erkannt. Dies könnte einerseits aus der Rotation des Objektes resultieren, an-

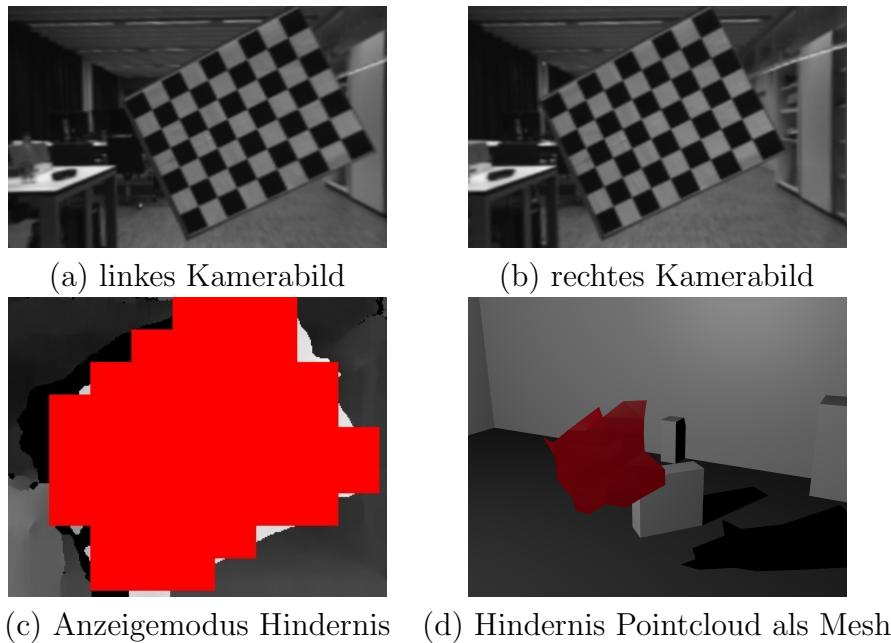


Abbildung 15: Abbildungen (a) - (c) zeigen die sichtbaren Bilder während der Testaufnahme. (c) zeigt eine als Mesh gerenderte Pointcloud

dererseits auch aus einer gewissen Biegung die das physische Objekt mit sich bringt.

Auch im in Abbildung 16 (b) abgebildeten Boxplot lässt sich erkennen, dass der Median beider gemessenen Distanzen, mit geringen Abweichungen, nahezu gleich ist. Auch die minimal und maximal Werte befinden sich, ebenfalls mit vernachlässigbarer Abweichungen auf derselben Gerade.

Mittleres Hindernis:

Auch die Erkennung mittlerer Hindernisse durch den Algorithmus 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. Das getestete Objekt befand sich während der Tests an einem durchsichtigen Plexiglas Stab. Dieser ist teilweise auch als Hindernis erkannt worden. 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. Bei einer Distanz von 120 Zentimetern ist eine Erkennung dieser Hin-

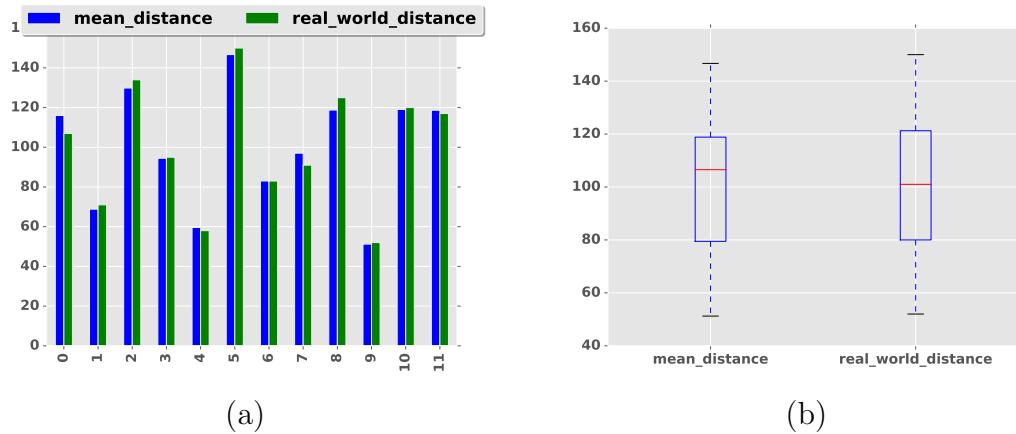


Abbildung 16

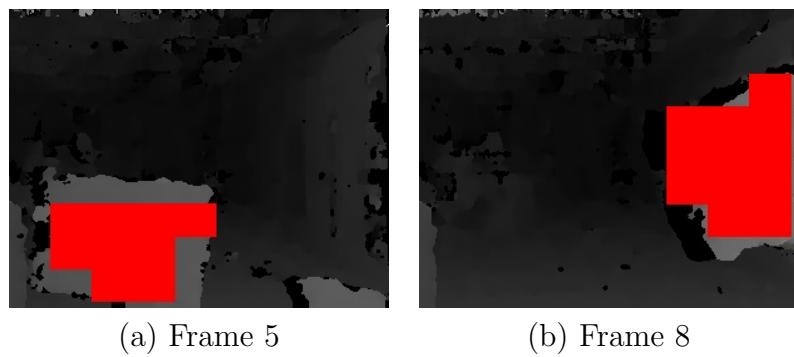


Abbildung 17

dernisgröße in nur 2 Subimages ungewöhnlich. Dies könnte zum Teil aus der verwendeten Blockgröße resultieren.

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. Eben diese Ergebnisse sind auch in Abbildung 18 (sb) sichtbar, das Maximum der realen Distanz resultiert dabei aus dem nicht erkannten Hindernis in Frame 9.

Kleines Hindernis:

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 Medians aufgrund der umliegenden Disparitäten untergeht, so kann auch kein Hindernis erkannt werden. Dies

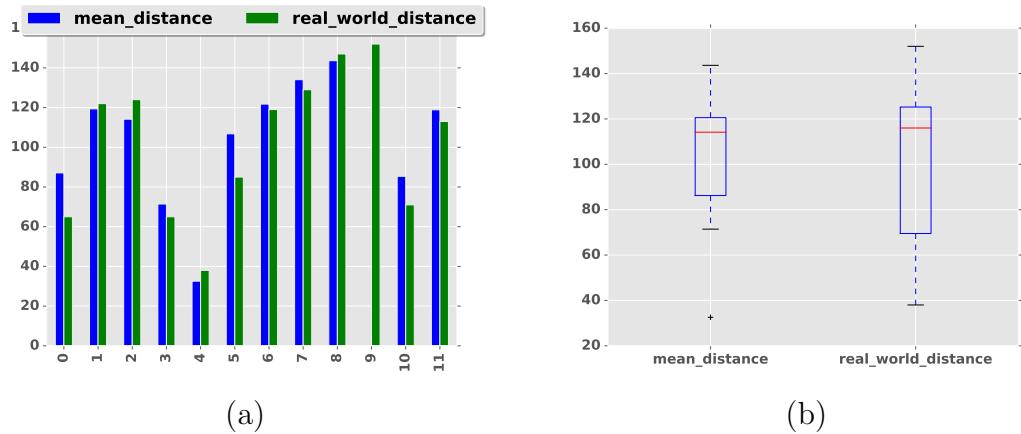


Abbildung 18

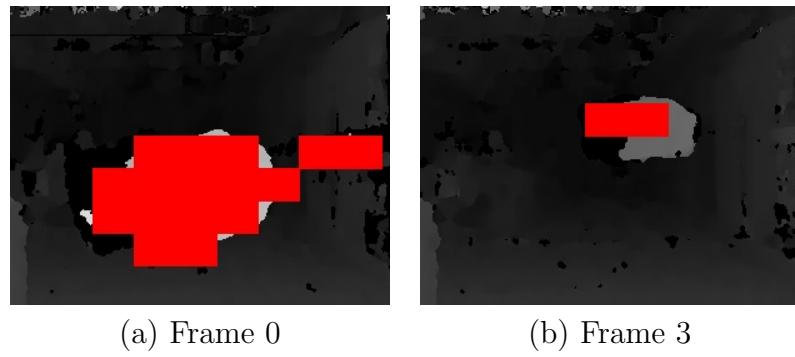


Abbildung 19

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 Median 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 sowie dem auftretenden Schatten welcher durch die Baseline zu begründen ist. Auch der in Abbildung 20 (b) dargestellte Boxplot zeigt, dass signifikant größere Distanzen erkannt wurden als sie in den realen Messungen vorkommen.

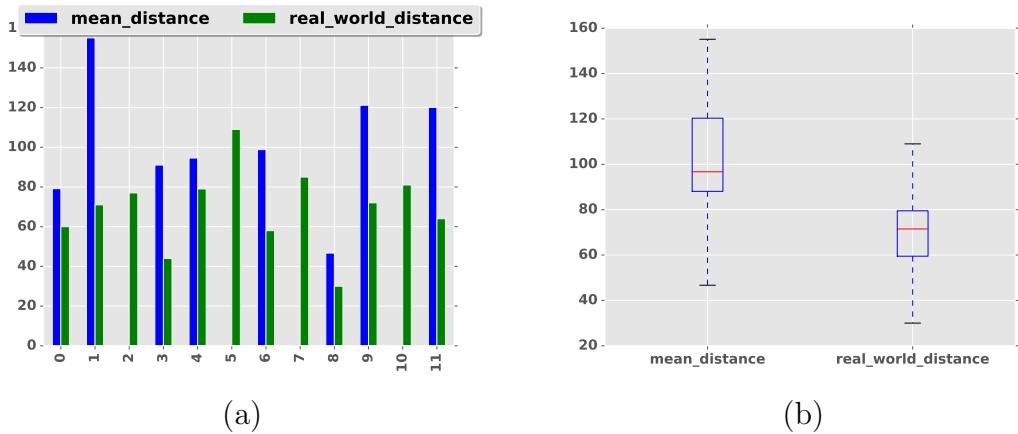


Abbildung 20

6.3 Evaluierung Samplepoint Detection

Durch die wesentlich höhere Anzahl an betrachteten Bereichen innerhalb der Disparity Map ist anzunehmen, dass die Erkennung kleiner Hindernisse eine höhere Erfolgswahrscheinlichkeit verspricht. Im Rahmen dieses Abschnittes wird die Samplepoint Detection getestet. Dies geschieht nach dem in 6.2 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 21 zeigt das jede der 12 Ausrichtungen des Objektes erkannt wurden. Die dabei auftretenden Differenzen zwischen der berechneten sowie gemessenen Distanz sind auf Messungenauigkeiten zurückzuführen. Auch in Grenzbereichen nahe dem Rand des Gefahrenbereichs wurden die Hindernisse erkannt.

Mittleres Hindernis:

Auch die Erkennung mittlerer Hindernisse erfolgte in allen Frames ohne signifikante Probleme (siehe Abbildung 22). Bereiche in denen aufgrund fehlender Textur keine Korrespondenz zugeordnet werden konnte wurden aufgrund der hohen Dichte der Samplepoints trotzdem erkannt. Wie Abbildungen 23 (a) und (b) zeigen wurden nicht gematchte Bereiche auch nicht als Hindernis erkannt, die umliegenden texturierten Bereiche überwiegen diese doch und sorgen daher für eine robuste Detektion.

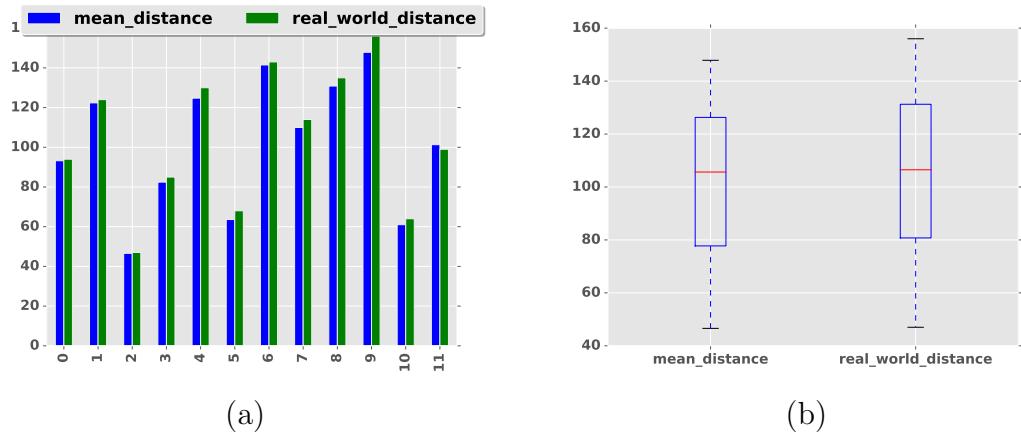


Abbildung 21

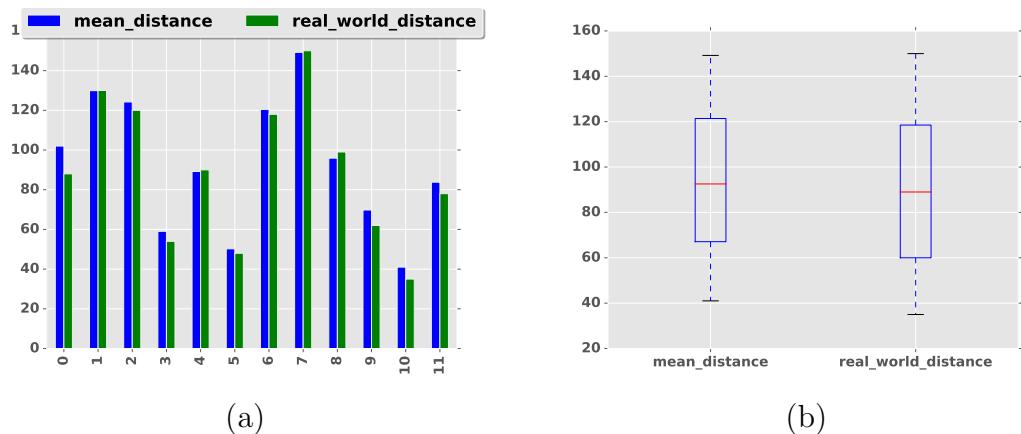


Abbildung 22

Die bereits beschriebene hohe Dichte der Samplepoints und deren geringe Größe hilft auch hier wieder den Stab als Hindernis zu erkennen (siehe Abbildung 23 (b))

Kleines Hindernis:

Die anfangs anfänglich 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. Diagramm 24 (a) belegt diese Aussage. Es wurden alle 12 Hindernisse ohne signifikante Differenz in den Distanzen 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 Dispa-

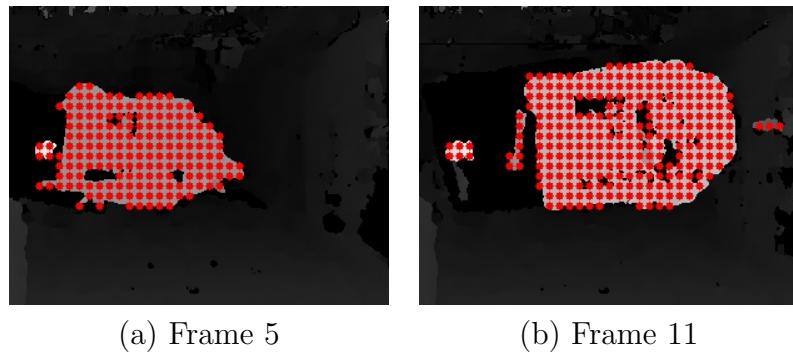


Abbildung 23

riety Map 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.

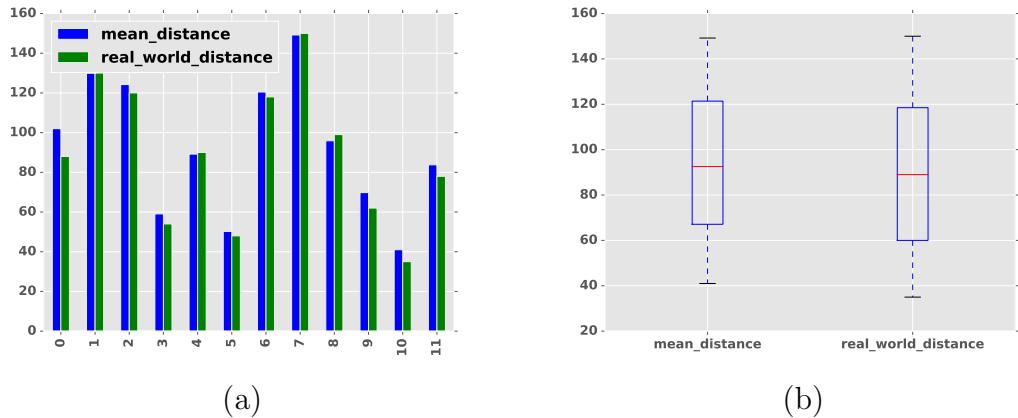


Abbildung 24

Weiterhin ist auch die statistische Auswertung der Distanzen in Boxplot 24 (b) ein Zeichen für die robuste Erkennung kleiner Hindernisse. Lediglich das untere Quartil der berechneten Entfernung weicht um ca. 6 Zentimeter von den realen Werten ab.

6.4 Weitere Versuche

Hinsichtlich der verschiedenen möglichen Faktoren welche eine erfolgreiche Erkennung beeinflussen können wurden im Rahmen der Evaluation beider Algorithmen weitere kleinere Tests durchgeführt. Jene umfassen die Erkennung

kleiner Hindernisse unter Veränderung der Größe des Matching-Blocks. Weiterhin wird die Erkennung kritischer Umgebungen (durchsichtige, reflektierende und homogener Flächen) getestet. Anschließend erfolgt eine Einschätzung welche minimale Größe auf bestimmte von einem Subimage/Samplepoint erkannt werden kann. Das während des Tests genutzte Setup wurde während der Aufnahme nicht bewegt. Das Objekt befand sich für jede Blockgröße an der jeweiligen Distanz. Zur Durchführung der Tests wurde das in Abschnitt 6.2 und 6.3 genutzte Winizige Hindernis verwendet.

6.4.1 Veränderung 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 7.1 statt.

Blockgröße	berechnete Distanz	real gemessene Distanz
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 6.2: Ausgewertete Daten des Subimage Sets

Wie aus Abbildung 6.2 ersichtlich wird hat eine Änderung der Blockgröße

keine Auswirkungen auf die Erkennung des Objektes. Die berechneten Entfernungen werden erheblich von den Pixeln des Hintergrundes beeinflusst, was in den verzerrten berechneten Distanzen resultiert. Dabei auftretende Differenzen zwischen den real gemessenen Entfernungen sowie den berechneten resultieren aus Interferenzen während der Neuberechnung der Disparity Map in jedem Frame.

Samplepoint Detection:

Blockgröße	berechnete Distanz	real gemessene Distanz
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 6.3: 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. Jedoch ist die Differenz der Entfernungen bei 13 Pixeln Blockgröße am geringsten. Während der einzelnen Versuche wurde der Befestigungsstab des Hindernisses ebenfalls erkannt (siehe Abbildung 25).

Dabei lässt sich in allen Bildern welche im Abstand von 100 Zentimetern aufgenommen erkennen, dass das eigentliche Hindernis nicht mehr gefunden wurde, sondern die Befestigung dessen das erkannte Hindernis repräsentiert. Ein Grund für diese „falsch“-Erkennung könnte auch hier die Verwendung des



Abbildung 25: Testset unter Verwendung einer Blockgröße von 13 Pixeln in verschiedenen Distanzen (50 cm bis 150 cm v.l.n.r.)

Mittelwertes, sowie der nicht beachtete Raum zwischen den einzelnen Sample-points 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 Aufnahmeauflösung 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 Disparity Map.

6.5 Diskussion

Die in den Abschnitten 6.2 und 6.3 erlangten Ergebnisse belegen die generelle Funktionsweise 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 stellt. Die Detektion des großen sowie mittleren Hindernisses ergab in allen Versuchen die Ergebnisse mit den niedrigsten Differenzen zwischen berechneten und gemessenen Entfernung.

Das Konzept der Subimage Detection ist in Bezug auf große und mittlere Hindernisse als robust einzustufen, jedoch ist die Berechnung des Medians 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 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.

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. Dies reicht jedoch nicht immer um das eigentliche Hindernis zu erkennen. In vielen Fällen wurde das eigentliche Hindernis nicht mehr direkt erkannt, sondern die Befestigung dessen. Wie bereits in Abschnitt 6.3 angedeutet liegt das Problem dabei in der zugrunde liegenden Disparity Map. 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 vom Blockmatching Algorithmus übergangen oder mit benachbarten Bereichen verbunden wird.

In Kapitel 4 wurden die Abläufe und Funktionsweisen der einzelnen Algorithmen detailliert beleuchtet. Der dabei erläuterte finale Prozess einer Hinderniserkennung (also eines verarbeiteten Disparity Frames) beinhaltet die Erstellung einer Pointcloud bestehend aus den erkannten Hindernissen.

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 weitere Detektion. Auch die Ergebnisse der Samplepoint Detection veränderten sich nicht signifikant. Jegliche Differenz zwischen den erkannten und berechneten Distanzen resultiert daher eher aus der zugrunde liegenden Disparity Map. Trotz dessen erscheinen die Erkennung bei einer Blockgröße von 13 Pixeln als am präzisesten. Auf eine Durchführung der Parameteränderung unter Benutzung des mittleren sowie großen Hindernisses wurde aufgrund der robusten Ergebnisse beider bewusst verzichtet.

Kapitel 7

Diskussion

Im Rahmen des Kapitels „Evaluation“ wurden beide entwickelten Methoden in verschiedenen Versuchen getestet. Die erlangten Ergebnisse beider Algorithmen unterscheiden sich in diversen Bereichen. Es erfolgt somit eine Gegenüberstellung beider Algorithmen in Hinsicht auf die Robustheit der Erkennung sowie deren Performance.

7.1 Gegenüberstellung beider Algorithmen

7.1.1 Performance

Hinsichtlich der Performance der entwickelten Systeme wurden verschiedenen Punkte betrachtet. Die zugrunde liegende Berechnung der Disparity Map 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 ohne Hindernisse

- (c) Geschwindigkeit der detectObstacles Funktion mit einem Hindernis im gesamten Sichtbereich

Zu Beginn ist ein essenzieller Schritt die Geschwindigkeit der Disparity Map Berechnung zu analysieren. Dabei wurden beide Kameras im gebinnten Modus getestet. Die Aufnahmerate der Kameras beträgt dabei 50 Frames pro Sekunde bei einer Verschlusszeit von $10000 \mu\text{s}$. Die unter Veränderung der Blockgröße erhaltenen Parameter sind in Tabelle 7.1 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	Zeit pro Frame	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 7.1: Blockgröße und daraus resultierende Frameraten

Die dabei gemessene Bildwiederholrate von 24 Einzelbildern pro Sekunde ist eine gute Voraussetzung für die Hinderniserkennung. Unter der Annahme das es zu keiner Verlangsamung dieser kommt ist es möglich sich mit Einer Geschwindigkeit von $12 \frac{\text{m}}{\text{s}}$ zu bewegen und für jeden zurückgelegten Meter 2 berechnete Disparity Maps zu erhalten. Eine solche Geschwindigkeit ist bei besagter Framerate nicht die präferierte Geschwindigkeit jedoch potentiell möglich. Zudem in weiteren Schritten mehr Zeit für die Hinderniserkennung sowie die Entwicklung einer Vermeidungsstrategie in Betracht gezogen werden muss.

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 7.2.

Aus dieser wird ersichtlich, dass Szenario 1(a), welches einer echten Anwendung am nächsten kommt, bereits eine Framerate von 173 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, einem Hindernis welches den gesamten Sichtbereich einnimmt ist die kombinierte Framerate nicht geringer als 20.96

Szenario	Zeit pro Frame (Detection)	Detection fps
1(a)	0.0057	173.35
1(b)	0.0067	147.69
1(c)	0.0067	147.69
2(a)	0.0021	469.93
2(b)	0.0001	8759.63
2(c)	0.0042	234.64

Tabelle 7.2: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate

wie die folgende Rechnung aufzeigt. Dabei entsprechen die genutzten Werte denen aus 7.1 mit 13 Pixeln Blockgröße sowie 7.2 1(b).

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

$$t_{frame} = 0.0410 + 0.0067 = 0.0477 \quad (7.1)$$

$$fps = \frac{1}{0.0477} = 20,96$$

Die somit verloren gegangenen Frames 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 Point-cloud wenn die aktualisierten Werte innerhalb der Gefahrenzone liegen. Ohne vorhandene Hindernisse passiert somit nichts.

Die nachfolgende Tabelle (7.3) stellt die Ergebnisse des Versuches für die Samplepoint Detection dar. Auf den ersten Blick sind die erreichten Framerates (mit Ausnahme der Aktualisierung, sowie der Hinderniserkennung) generell niedriger als jene der Subimage Detection.

Szenario	Zeit pro Frame (Detection)	Detection fps
1(a)	0.0074	133.61
1(b)	0.0100	99.33
1(c)	0.0016	606.84
2(a)	0.0015	647.07
2(b)	0.0015	647.07
2(c)	0.0083	120.33

Tabelle 7.3: Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate

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 X. Auch im Echtwelt Szenario finden sich 30 Einzelbilder weniger pro Sekunde. Lediglich das Update sowie die Erkennung ohne Hindernisse erfolgt schneller. Die höhere Geschwindigkeit in der Aktualisierung resultiert aus der geringeren Anzahl an Pixeln welche betrachtet werden müssen. Nach der in 7.1 erläuterten Rechnung ergibt sich bei der Samplepoint Detection eine durchschnittliche Framerate von 19,60 Bildern/s.

7.1.2 Robustheit

Wie die Evaluation bereits aufzeigt ist die Erkennung unterschiedlicher Hindernisgrößen als robust anzusehen. Beide Algorithmen erkennen sowohl große als auch kleine 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. Auch die dahingehend erstellten Punktwolken liefern genaue Positionsinformationen ausgehend von der aktuellen Weltposition der Drohne. Jene liegen im Rahmen dieser Arbeit nicht vor da dies als ein anderer Teil des SLAM Forschungsfeldes anzusehen ist.

Bei der Erkennung 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 kleineren Anzahl an Pixeln. Dadurch sind diese weniger empfänglich für Verzerrungen der berechneten Distanz wie Subimages. Enthält ein einziger Samplepoint zu wenige Daten um als Hindernis angesehen zu werden, so ist die Wahrscheinlichkeit das seine Nachbarn diese Information enthalten bzw. erfassen konnten höher als beispielsweise die benachbarten Subimages. Dies ist auch als der große Vorteil der Samplepoint Detection anzusehen.

Weiterhin ließ sich während der Versuchsdurchführung deutlich erkennen, dass Bewegungen einen wesentlich Bestandteil der Hinderniserkennung darstellt. Wurden die Hindernisse bewegt, konnte gerade im Fall der Subimage Detection festgestellt werden, dass die Erkennung kleiner Hindernisse signifikant besser funktionierte wenn das Objekt Bewegung aufweist. 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 sta-

tischen Szene. Selbiges Phänomen trat auch bei der Samplepoint Detection auf.

Eine Veränderung der *SGBM* Blockgröße hatte keine signifikanten Auswirkungen auf die Robustheit beider Algorithmen. Lediglich die bereits in Abschnitt 6.5 erläuterten Ergebnisse bei 13 Pixeln brachte eine geringe Verbesserung der berechneten Distanz.

Kapitel 8

Fazit und zukünftige Arbeiten

Lorim ipsum dolor sit amet.

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.
- [GmbH, 2015] GmbH, M. V. (2015). MATRIX VISION GmbH - industrial image processing.

- [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.
- [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.
- [Middlebury Univeristy, 2015] Middlebury Univeristy (2015). Middlebury stereo datasets.
- [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, 2015a] OpenCV (2015a). OpenCV - camera calibration and 3d reconstruction.
- [OpenCV, 2015b] OpenCV (2015b). OpenCV - open source computer vision.
- [Pire et al., 2012] Pire, T., de Cristóforis, P., Nitsche, M., and Berlles, J. J. (2012). Stereo vision obstacle avoidance using depth and elevation maps. *IEEE VI RAS Summer School on “Robot Vision and Applications”, Santiago, Chile*.
- [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.