

Bauhaus-Universität Weimar  
Fakultät Medien  
Studiengang Medieninformatik

# Stereo-basierte Echtzeit-Hinderniserkennung für unbemannte Flugsysteme

## Bachelorarbeit

Hagen Hiller  
Geboren am 04.06.1992 in Berlin

Matrikelnummer 110514

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

Datum der Abgabe: 22.02.2016

### **Erklärung**

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

Weimar, den 22.02.2016

.....  
Hagen Hiller

## **Zusammenfassung**

Die Verwendung ferngesteuerter unbemannter Flugsysteme ist längst nicht mehr rein militärischer Natur. Auch in der Zivilgesellschaft werden diese zur Produktion von Filmen, zu Vermessungszwecken oder aber in privaten Bereichen verwendet. Jedoch werden diese privaten Anwendungen seit der zum Teil kritisch beäugt. Eine Assoziation von Drohnen mit dem Militär ist demnach noch immer gegeben. Gerade mediale Berichte bestärken diese Verbindung. Trotz dessen ist die autonome Erkundung von unbekannten Gebieten ein weit erforschter Bereich.

Die Verwendung einer Stereo-Kamera-Konfiguration ist dabei der einfachste Weg optische eine dreidimensionale Rekonstruktion der Szene zu erstellen anhand welcher etwaige Hindernisse gesucht werden. Im Rahmen dieser Arbeit wurden zwei Systeme zur Erkennung von Hindernissen in Echtzeit entwickelt. Dies ist gerade bei einer gleichzeitigen Lokalisierung und Kartographierung von Nöten, um die sichere Navigation des Flugsystems zu gewähren. Folglich ist gerade der Aspekt der Erkennung in Echtzeit, sowie der Erhalt der Hindernispositionen eine unabdingbare Anforderung. Die zu erhaltenen Daten müssen dabei effizient zur Entwicklung einer Strategie zur Vermeidung verwendet werden können.

# Inhaltsverzeichnis

<b>1 Einführung</b>	<b>3</b>
1.1 Motivation . . . . .	3
1.2 Hardwarekomponenten . . . . .	4
1.3 Ziel der Arbeit . . . . .	4
<b>2 Zugrunde liegende Konzepte und Algorithmen</b>	<b>6</b>
2.1 Epipolareometrie . . . . .	6
2.2 Kamerakalibrierung und Rektifizierung . . . . .	8
2.3 Stereo Matching . . . . .	10
2.3.1 Lokale Methoden . . . . .	10
2.3.2 Globale Methoden: . . . . .	10
2.3.3 Block Matching . . . . .	11
2.3.4 Semi Global Block Matching . . . . .	11
2.4 mvStereoVision Framework . . . . .	15
<b>3 Optische Verfahren zur Hinderniserkennung</b>	<b>18</b>
3.1 Aktiv optische Algorithmen . . . . .	18
3.2 Passiv optische Algorithmen . . . . .	19
<b>4 Entwickelte Hinderniserkennungen</b>	<b>22</b>
4.1 Applikationsstruktur . . . . .	22
4.2 Grundlegende Operationen . . . . .	23
4.3 Subimage Detection . . . . .	25
4.4 Samplepoint Detection . . . . .	28
<b>5 Evaluation</b>	<b>31</b>
5.1 Testsetup . . . . .	31
5.2 Evaluierung Subimage Detection . . . . .	33
5.3 Evaluierung Samplepoint Detection . . . . .	37
5.4 Weitere Versuche . . . . .	40
5.4.1 Erkennung unter Veränderung der <i>SGBM</i> Block Größe .	40

## **INHALTSVERZEICHNIS**

---

5.4.2	Erkennung von reflektierenden und durchsichtigen Be- reichen . . . . .	43
5.4.3	Erkennung unter Veränderung des Samplepoint Radius .	45
5.5	Diskussion . . . . .	46
<b>6</b>	<b>Limitierungen und Lösungsansätze</b>	<b>49</b>
6.1	Bestehende Limitierungen und Lösungsansätze . . . . .	49
6.2	Diskussion . . . . .	51
<b>7</b>	<b>Diskussion</b>	<b>55</b>
7.1	Gegenüberstellung beider Algorithmen . . . . .	55
7.1.1	Performance . . . . .	55
7.1.2	Robustheit . . . . .	59
<b>8</b>	<b>Fazit und zukünftige Arbeiten</b>	<b>61</b>
	<b>Literaturverzeichnis</b>	<b>63</b>

# Kapitel 1

## Einführung

### 1.1 Motivation

Die vorliegende Arbeit ist im Forschungsgebiet Robotik angesiedelt und behandelt bildbasierte Verfahren zur Erkennung von Hindernissen für autonome Flugsysteme (UAV<sup>1</sup>). Diese Verfahren sind für die Navigation sowie für die autonome Erkundung von unbekannten oder unzugänglichen Gebieten unerlässlich. Dabei müssen nicht nur die physikalischen Eigenschaften der Drohne betrachtet werden sondern auch die Fusion verschiedenster Sensoren.

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

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

---

<sup>1</sup>Unmanned Aircraft System

Weitere Anwendungsbereiche können sowohl komplexerer als auch einfacherer Natur sein. Prinzipiell ist es möglich die entwickelten Algorithmen und Methoden im Automobil Bereich zu verwenden um beispielsweise Objekte vor oder hinter dem Kraftfahrzeug zu erkennen und deren Distanz zu ermitteln. Weiterhin ist es möglich die aufgenommenen Punktwolken im Nachhinein zur groben dreidimensionalen Rekonstruktion der Umgebung zu verwenden.

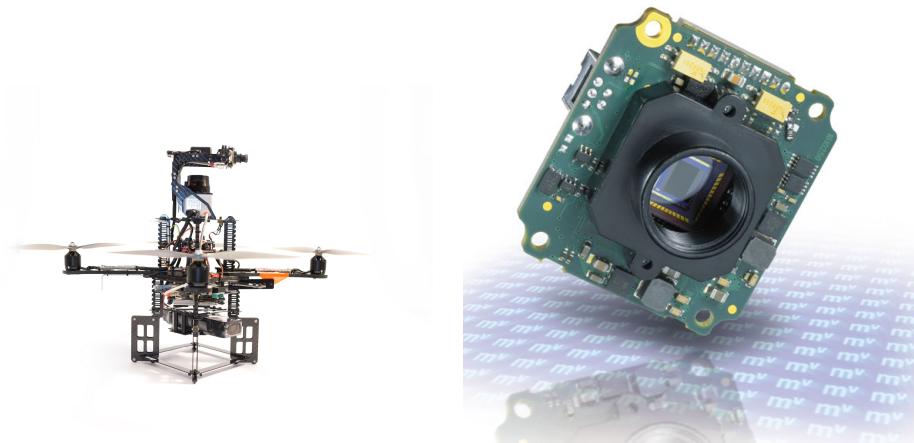
## 1.2 Hardwarekomponenten

Die aktive Entwicklung der Methoden und Algorithmen erfolgte im Hinblick auf eine Verwendung dieser durch das von Ascending Technologies [Ascending Technologies, 2015] entwickelte UAS Pelican (Abbildung 1). Dabei handelt es sich dabei um einen Quadrocopter der speziell für Forschungszwecke entwickelt wurde. Er ist mit einem Bordcomputer ausgestattet, der die nötige Leistung für die Entwicklung der Algorithmen bereitstellt (3rd Generation Intel Core i7). Weiterhin wurden zwei MatrixVision BlueFOX mv-MLC200wC Industriekameras [GmbH, 2015] mit einem Sichtfeld von je 100° als visuelles System verwendet. Die maximale Auflösung beider Kameras beträgt  $752 \times 480$  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 in einem horizontalen und vertikalen Binning-Modus verwendet. Dies halbiert die Anzahl der Bildpunkte in beiden Dimensionen auf  $376 \times 240$ , wodurch der Berechnungsaufwand verringert und die Aufnahmerate der Kameras auf bis zu 170 Einzelbilder pro Sekunde maximiert wird.

Für die Implementierung der Methoden wurde die freie Computer Vision Bibliothek OpenCV [OpenCV, 2015] verwendet. Diese stellt benötigte algebraische Grundoperationen sowie bestimmte Algorithmen, welche im Rahmen dieser Arbeit genutzt wurden, zur Verfügung.

## 1.3 Ziel der Arbeit

Basierend auf photogrammetrischen Konzepten ist es möglich räumliche Tiefe aus zweidimensionalen Bilddaten zu berechnen. Generell werden mindestens zwei Bilder aus unterschiedlichen Standpunkten für die Berechnung dreidimensionaler Informationen benötigt. Das daraus resultierende Problem welches bei der Benutzung einer Kamera entsteht ist die oft ungenau Schätzung der relativen Orientierung sowie die damit einher gehende Skalierung. Bei der Verwendung eines Stereosystems ist die relative Orientierung beider Kameras zueinander bereits bekannt, was eine direkte Berechnung metrischer Koordinaten ermöglicht. Da Zuverlässigkeit sowie Genauigkeit vor allem im Kontext



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

der Navigation in Innenräumen sehr wichtig ist wurde in dieser Arbeit auf ein Stereo-Setup gesetzt.

Vor diesem Hintergrund werden in Kapitel 2 der Arbeit zugrundeliegende Algorithmen und Konzepte erläutert. Weiterhin wird das entwickelte Framework zur Bildaufnahme und Vorprozessierung der Bilder grundlegend beschrieben. Anschliessend werden einige State of the Art Methoden der Hinderniserkennung beschrieben wobei dabei zwischen aktiven und passiven optischen Systemen unterschieden wird. Diese Einteilung dient einerseits dafür einen Überblick über bereits bestehende Techniken sowie implementierte Systeme zu erhalten, andererseits um auch die Vor- und Nachteile der jeweiligen Technik herauszuarbeiten. Im Anschluss daran beschreibt Kapitel 4 die beiden entwickelten Methoden zur bildbasierten Hinderniserkennung und deren Implementierung detailliert. Anschließend erfolgt die Evaluation beider Verfahren wobei die Erkennung verschiedener Hindernisgrößen getestet wird. Weitere Tests zur zukünftigen Verbesserung der Algorithmen weisen auf welches aktuellen Limitierungen durch die verwendeten Konzepte vorliegen. Kapitel 6 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 5 erlangten Ergebnisse weitergehend aus und stellt beide Algorithmen hinsichtlich der Robustheit der Erkennung, sowie der erreichten Performance aus. Kapitel 8 zieht ein Résumé aus den Ergebnissen der Arbeit und gibt einen Ausblick auf mögliche zukünftige Arbeiten in diesem Bereich.

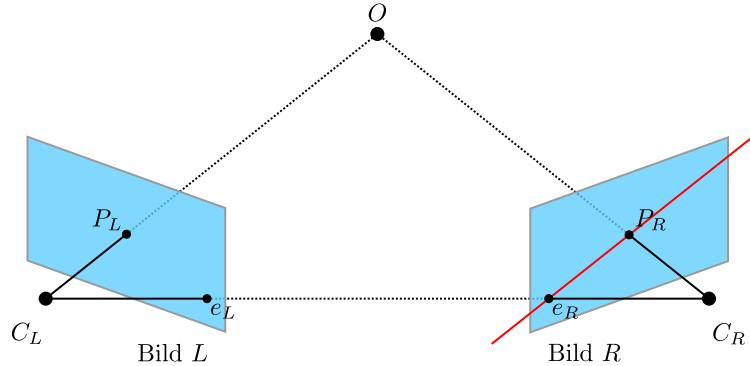
# Kapitel 2

## Zugrunde liegende Konzepte und Algorithmen

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

### 2.1 Epipolargeometrie

Photogrammetrische Verfahren basieren oft auf dem Konzept der Epipolargeometrie. Dieses 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 entsprechenden Bildpunkt auf einer Geraden. Unter Zuhilfenahme der räumlichen Orientierung, sowie der intrinsischen Parameter der Kamera (Brennweite, Koordinaten des Bildhauptpunktes) ist es möglich den Schnittpunkt mehrere solcher Raumgeraden zu berechnen um die dreidimensionalen Koordinaten des Objektpunktes zu erhalten. Dabei gilt generell, wenn ein Punkt  $P$  im linken Bild gegeben ist, so wird die Suche des korrespondierenden Punktes  $P'$  auf die Epipolarlinie des rechten Bildes reduziert. Die algebraische Repräsentation dessen ist die Fundamentalmatrix  $F$ .



**Abbildung 2:** Darstellung der Epipolargeometrie.

Abbildung 2 visualisiert diesen Prozess. Gegeben sind die beiden Projektionszentren  $C_L$  und  $C_R$  der beiden Bildebenen, welche im Zuge der Kalibrierung mitbestimmt werden, sowie die Bildpunkte  $P_L$  und  $P_R$ , welche beide die Position des Objektpunktes  $O$  in beiden Bildern beschreiben. Zunächst ist es nur möglich den Strahl auf dem sich  $O$  befindet zu bestimmen. Aufgrund des Wissens der relativen Orientierung der Kameras können mithilfe der Basislinie zwischen den beiden Kameras die Epipole in  $L$  und  $R$  ermittelt werden. Rechnerisch ergibt sich die Epipolarlinie aus der Multiplikation des Bildpunktes mit der Fundamentalmatrix. Betrachtet man den Strahl  $C_L, O$  aus Perspektive des rechten Bildes, und verschiebt  $O$  entlang dessen, so ergibt sich die Epipolarlinie in  $R$ . Somit ist es möglich mithilfe der vorhandenen Epipole  $e_L, e_R$  die Epipolarlinie  $e_l, P_{x_2,y_2}$  zu bestimmten. Anhand dieser kann der Suchraum für den gesuchten korrespondierenden Bildpunkt im Rechten Bild auf die Epipolarlinie eingegrenzt werden. Kann der Punkt mittels eines *Matching*-Ansatzes gefunden werden, ist es möglich die 3D Position des Objektpunkts zu triangulieren.

Generell sind die Epipole durch den Schnittpunkt der Basislinie mit den beiden Bildebenen definiert. Im Falle eines Stereonormalfallen jedoch 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, dass die Epipolargeometrie bereits bekannt ist, und korrespondierende Bildpunkte nur noch innerhalb einer Pixelreihe gesucht werden müssen.

## 2.2 Kamerakalibrierung und Rektifizierung

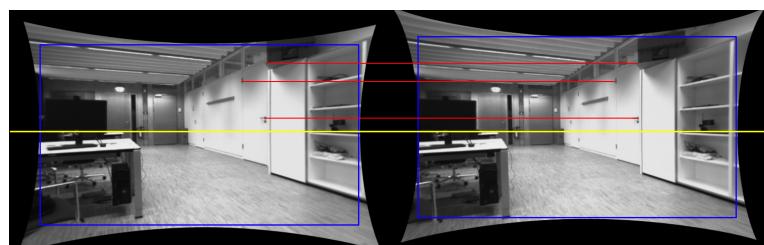
Um den Prozess der Korrespondenzanalyse zu vereinfachen und beschleunigen bietet die Rektifizierung der Referenzbilder einen simplen Einstiegspunkt zur Lösung des Stereo-Korrespondenzproblems. Dafür müssen zunächst die intrinsischen und extrinsischen Parameter der Kamera berechnet werden. Die Ermittlung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

Folgend aus der Kalibrierung der Kameras beginnt der Prozess der Rektifizierung der Bilder. Dieser kann dabei in zwei Schritte unterteilt werden, zunächst die Rektifizierung der Bilder, anschliessend die Berechnung der ROI. 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 Epipolargeometrie paarweise rotiert, sodass die Epipole beider Bilder auf der  $x$ -Achse liegen. Nach Verschiebung der Epipole entlang der  $x$ -Achse ins

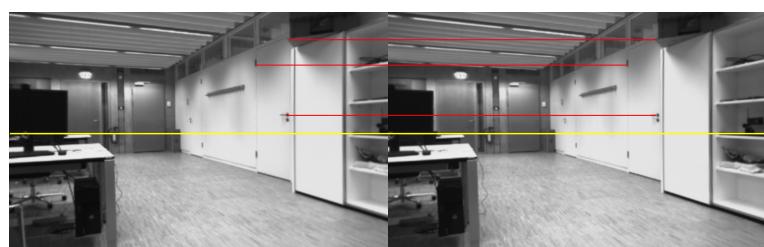
Unendliche ist Parallelität der Epipolarlinien gewährleistet. Daraus resultieren parallele und orthogonale Blickrichtungen zur identischen Bildebene, somit sind beide Bildebene Element derselben Ebene. Die finale ROI ergibt sich aus der Konjunktion der einzelnen ROI's. Nach der Rektifizierung und dem 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.  
Die ROIs (blau) kennzeichnen die jeweils maximal möglichen rechteckigen Flächen.



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

**Abbildung 3:** Ablauf der Rektifizierung

## 2.3 Stereo Matching

Das grundlegende Konzept des *Stereo Matchings* beschreibt das Finden korrespondierender Punkte in zwei 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 zwischen korrespondierenden Pixeln berechnet werden. Die räumlichen 3D-Koordinaten der aufgenommenen Szene stehen im direkten (funktionalen) Zusammenhang mit der Information über Disparität, der relativen Orientierung der Kameras sowie der Kamerakalibrierung. Durch die Transformation zum Stereonormalfall vereinfacht sich dieser Zusammenhang enorm. Im Laufe dieses Prozesses kommt es zu zwei wesentlichen Problemstellungen: die Berechnung der Disparität (Stereo Korrespondenz) sowie die Invertierung der projektiven Geometrie, um dreidimensionale Informationen aus der errechneten Disparität zu erhalten. Sofern eine Lösung beider Probleme vorhanden ist, können diese Informationen durch einfache Triangulierung errechnet werden.

### 2.3.1 Lokale Methoden

Zur Berechnung der Disparität in lokalen *Stereo Matching* Algorithmen gilt grundlegendes Prinzip: „Finde Pixel  $P_2$  korrespondierend zu  $P_1$  im Referenzbild“. Dabei wird die Korrelation von  $P_1$  und  $P_2$  unter Einbezug der lokalen Nachbarschaft bestimmt und die maximale Korrelation als Indikator für die Übereinstimmung von Pixeln herangezogen. 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). Jedoch können aufgrund des recht einfachen Ansatzes, der bei starken perspektivischen Änderungen und untexturierten Bereichen keine sinnvollen Ergebnisse liefert, falsche Berechnungen der räumlichen Tiefe auftreten, da benachbarte Pixel verschiedene Disparitäten aufweisen können (beispielsweise an horizontalen Kanten wie Türrahmen etc.). Strukturell gesehen sind lokale Methoden einfacher gehalten als globale Methoden, wodurch ein hoher Grad an Optimierung in der Implementierung möglich ist.

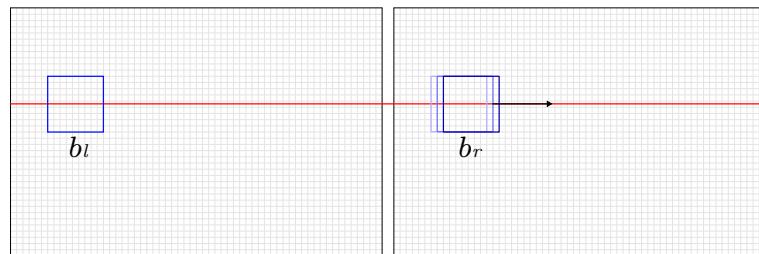
### 2.3.2 Globale Methoden:

Bei globalen *Stereo Matching* Algorithmen wird ein globales Modell der zu betrachtenden Szene erstellt sowie eine ebenfalls globale Kostenfunktion definiert, welche es zu minimieren gilt. Dabei werden im Gegensatz zu lokalen

Methoden die Korrespondenzen in einer Pixelreihe und die des gesamten Bildes miteinander verglichen. Zur Vereinfachung dieses Vorgangs betrachten einige Algorithmen nur die durch die Epipolargeometrie vorgegebenen Bildbereiche, wodurch ein zweidimensionales Problem auf ein eindimensionales reduziert wird. Resultierend daraus liegt die Stärke globaler Methoden in der Bewältigung schwacher Texturen sowie auftretender Okklusionen und unterschiedlicher Lichteinfälle, was, aufgrund der höheren Komplexität in einem höheren Rechen- und Speicheraufwand mündet. Weitere Verbesserungen der Ergebnisse können durch Techniken wie Dynamische Programmierung und Graph Cut erreicht werden.

### 2.3.3 Block Matching

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



**Abbildung 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.

### 2.3.4 Semi Global Block Matching

Das im Rahmen dieser Arbeit verwendete Verfahren zur Berechnung von Disparity Maps ist der in der freien Computer Vision Bibliothek OpenCV [OpenCV, 2015]

implementierte *Semi Global Block Matching* Algorithmus. Diese leicht abgewandelte Implementierung des *Semi Global Matching* (SGM) Algorithmus von Hirschmüller et. al [Hirschmüller, 2005] zeichnet sich sowohl durch seine Genauigkeit, als auch durch seine performante Berechnung aus. Unter Zuhilfenahme dieses Verfahren ist es möglich, Disparitätenkarten in Echtzeit zu berechnen. Im Folgenden wird zunächst die grundlegende Funktionsweise des SGM erläutert. Im Anschluss daran werden die Unterschiede zum SGBM herausgearbeitet sowie eine kurze Erklärung der wesentlichsten Parameter dessen vorgenommen.

### **SGM:**

Die grundlegende Idee des *Semi-Global Matching* Algorithmus besteht in dem pixelweisen Matching von *Mutual Information*<sup>1</sup>. Ausgangsvoraussetzung dafür sind verschiedene Bilder derselben Szene mit vorhandener und bekannter Epipolargeometrie. Ein weiteres Kernelement des *SGM* ist die Anwendung eines globalen zweidimensionalen Glattheitskriteriums, was durch die Kombination mehrerer eindimensionaler Beschränkungen approximiert wird. Zunächst werden für jeden Pixel  $P$  aus der Intensität  $I_{bp}$  sowie der vermuteten Korrespondenz  $I_{mq}$  auf der Epipolarlinie  $q = e_{bm}(pd)$  die Matching Kosten berechnet. Zur Berechnung der MI wird zunächst eine initiale Disparitätenkarte benötigt. Diese wird nach dem Ansatz von Kim et al [Lee et al., 2012] zufällig gewählt, um die Kosten berechnen zu können. Zur Steigerung der Performance wird die Disparitätenkarte zunächst nur mit halber Auflösung berechnet, wodurch der Rechenaufwand um den Faktor 8 reduziert wird. Zur Vermeidung falscher Kostenberechnungen durch auftretendes Rauschen innerhalb des Bildes werden benachbarte Disparitäten mit in die Berechnungen einbezogen.

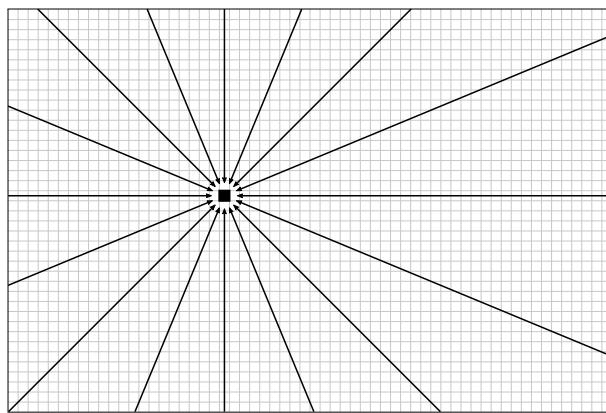
Das letzte Problem besteht in der Berechnung der Korrespondenz sowie der daraus resultierenden Disparitäten. Dabei wird nach der Disparität  $D$  mit der geringsten berechneten Energiefunktion gesucht. Anstatt nun einfach den minimalsten Pfad der Kosten zu summieren, werden zusätzlich auch andere Richtungen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

### **SGBM:**

Der *Semi Global Block Matching* Algorithmus ist ein in der Computer Vi-

---

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



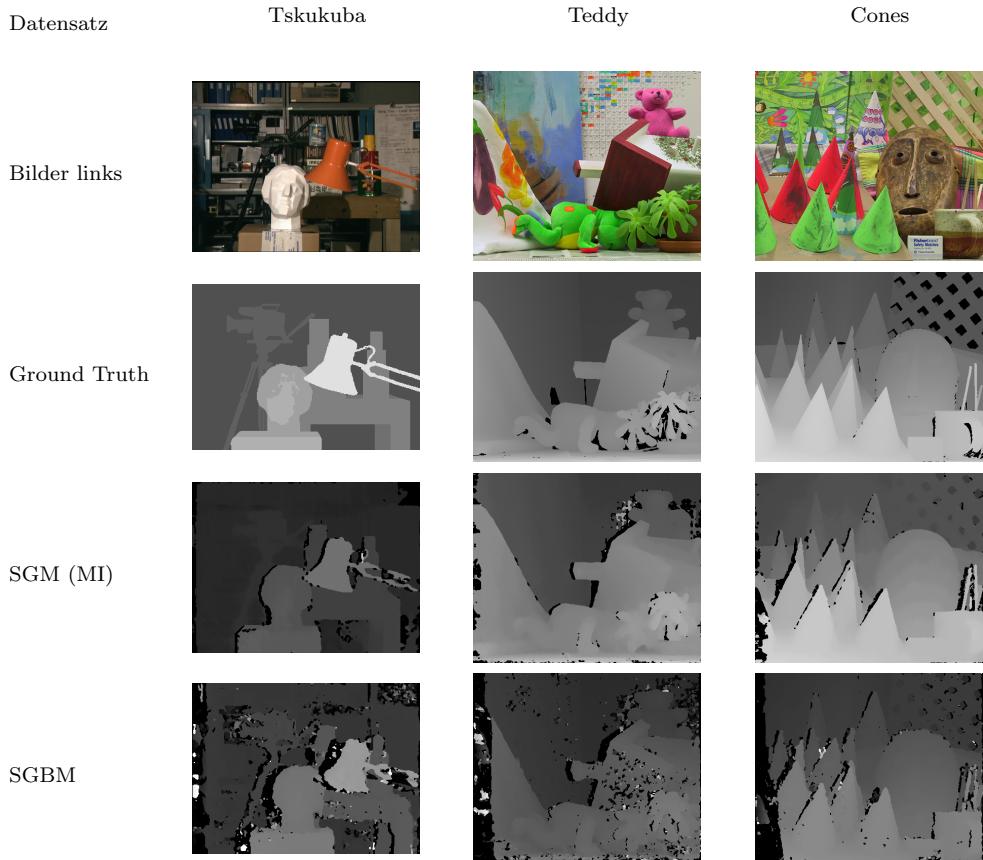
**Abbildung 5:** Darstellung der verschiedenen Richtungen

sion Library OpenCV implementiertes Verfahren zur schnellen Berechnung von Disparitätenkarten. 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 Verschiedene Schritte der Vor- und Nachprozessierung des *StereoBM* Algorithmus werden verwendet, dazu zählen Filtermethoden wie quatdra-tische Interpolation oder Sprenkelfilter.

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

Abbildung 6 stellt die originalen Bildern der Szene ([Middlebury Univeristy, 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.



**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 4.2 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.

## 2.4 mvStereoVision Framework

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

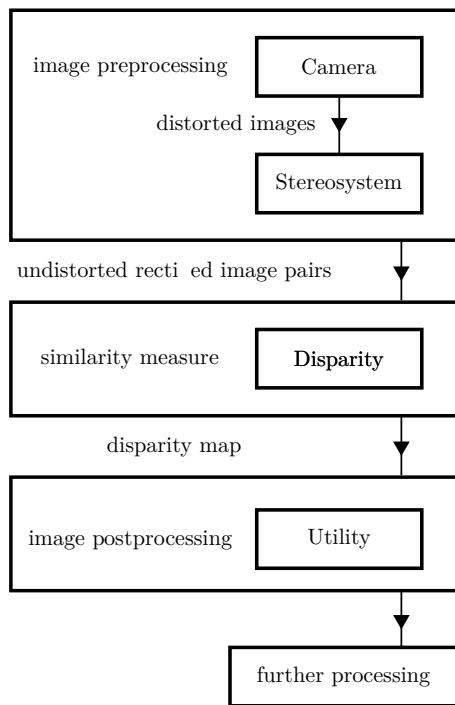
### Bildaufnahme:

Die Aufnahme der einzelnen Bilder beginnt bei einer Anfrage an die Kamera nach den jeweiligen Rohdaten. Diese werden 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 Disparitätenkarte erfolgt ebenfalls in einem eigenen Thread. Sofern eine neue Tiefenkarte 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.

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



**Abbildung 7:** Ablauf der Bildaufnahme, Berechnung der Disparitätskarte sowie weiteren Verarbeitung

Dabei beschreiben  $-C_x$  und  $-C_y$  die beiden negativen Koordinaten des Bildhauptpunktes der Referenzkamera, beispielsweise der linken. 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 es sich um einen (nach der Rektifizierung vorhandenen) Stereonormalfall handelt und sich die beiden Strahlen der Bildhauptpunkte in der Unendlichkeit schneiden. Dies führt zu einer Annäherung des unteren rechten Parameters an 0. Zur Einfacheren Darstellung und Erläuterung werden die Basislinie der Kamera  $\frac{-1}{T_x}$  im folgenden als  $a$  sowie die Darstellung beider Bildhauptpunkte  $\frac{C_x - C'_x}{T_x}$  als  $b$  notiert.

Eine Eigenschaft der Reprojektionsmatrix ist die einfache Berechnung von dreidimensionalen Weltkoordinaten. Diese Berechnung erfolgt mittels Gleichung 2.2.

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

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

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, mit der Reprojektionsmatrix. Die dreidimensionalen Koordinaten ergeben sich damit aus:

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

# Kapitel 3

## Optische Verfahren zur Hinderniserkennung

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

### 3.1 Aktiv optische Algorithmen

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

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

Die von Lee et al. [Lee et al., 2012] vorgeschlagene Methodik der Hinderniserkennung bedient sich einer TOF Kamera. Die aufgenommenen Tiefenbilder werden basierend auf den Tiefenmessungen segmentiert. Dabei werden mit Hilfe eines Algorithmus zur Kantenerkennung jene herausgefiltert und anschliessend von der Tiefenkarte subtrahiert. Anschließend erfolgt eine Tiefenanalyse der nun vorliegenden einzelnen Segmente. Um die gefundenen Segmente als Hindernis klassifizieren zu können wird zunächst die Standardabweichung jedes Segments berechnet. Da die Tiefenwerte des Bodens einen hohen Grad der Streuung aufweisen ist die Standardabweichung dessen höher als bei anderen Segmenten. Diese Information dient zur Unterscheidung zwischen Hindernis und Boden. Dabei gelten Hindernisse als sich bewegende oder statische Objekte, welche sich innerhalb eines definierten Gefahrenbereichs befinden. Lee et al. definieren Ihre Gefahrenzone dabei mit 1 – 2 Metern. Die erkannten Hindernisse werden nun anhand der mittleren Distanz 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 Disparitätenkarten. 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 besteht aus 5 Bereichen, von denen 3 als mögliche Richtungen betrachtet werden. Sobald die die minimalste Distanz eines Sektors geringer als 60 cm ist wird angenommen, dass sich das System vor einem Hindernis befindet. Ausgehend von der Menge an Sektoren mit gefundenen Hindernissen wird die neue Bewegungsrichtung angepasst.

## 3.2 Passiv optische Algorithmen

Passiv optische Algorithmen beziehen sich auf die Berechnung dreidimensionaler Informationen aus Bildern ohne eigene Lichtquelle. Dabei sind viele dieser

Methoden an das menschliche oder auch tierische Sehen angelehnt. 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 die Analyse von Bildern hinsichtlich Lichteinflüssen, Schattierungen oder durch Veränderung des Kamerafokus vorgenommen werden.

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

Eine weitere Methode zur Hindernisvermeidung für UAVs wurde von Richards et al. [Richards et al., 2014] vorgestellt. Optischer Fluss sowie *Feature Tracking* bieten dabei die Grundlage für die Erkennung, Vermeidung und Voraussage, welcher Bereich im Raum der sicherste ist. Dabei gehen die beiden Ausgangstechniken jeweils einer anderen Aufgabe nach. Der Optische Fluss dient zum Erkennen und Verfolgen von Objekten sowie für die Voraussage der Position im nächsten Einzelbild, wohingegen das *Feature Tracking* [Shi and Tomasi, 1994] für die Erkennung von markanten Punkten innerhalb des Objektes genutzt wird. Bei diesem wird in jedem folgenden Einzelbild verglichen, ob bereits bekannte Punkte innerhalb einer bestimmten Distanz mit denen des aktuellen

Bildes übereinstimmen. Zur Schätzung zu erwartenden Position der Merkmale im darauffolgenden Bild werden mit Hilfe des Optische Fluss' die Verschiebungsvektoren zwischen beiden Bildern berechnet. Aufgrund dessen, dass der zugrunde liegende Algorithmus von Lukas-Kanade [Lucas et al., 1981] nur bei kleinen Verschiebungen valide Ergebnisse liefert, wird ein pyramidaler Ansatz [Bouguet, 2001] für das Matching verwendet, bei welchem beide Bilder eines *frames* herunter skaliert werden. Durch die Verbindung dieser beiden Techniken lassen sich Objekte erkennen und verfolgen. Zur Bestimmung der nächstbesten Position für das UAV wird ausgehend von den berechneten Resultaten (gefundene und erkannte Hindernisse) eine stochastische Matrix erstellt welche zur weiteren Planung des Fluges verwendet wird.

Bei der Entwicklung von Algorithmen, welche auf 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 Problems.

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 und Scherer [Mori and Scherer, 2013] für die Nutzung eines monokularen Setups. Durch die Verwendung des optischen Flusses ist es auch mit nur einer Kamera möglich, Hindernisse zu erkennen, jedoch fehlt die eigentliche Erkennung von Tiefe. Sofern sich ein Objekt direkt auf das System zu bewegt, kann es nur schwer erkannt werden, da kaum perspektivische Veränderungen 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 und Scherer setzten bei der Erkennung von Merkmalen auf den SURF Algorithmus [Bay et al., 2006], welcher gefundene Features auch nach Skaleninvarianz 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 4

## Entwickelte Hinderniserkennungen

Im Rahmen dieser Arbeit wurden zwei Ansätze zur Erkennung von Hindernissen in Echtzeit entwickelt. Ein Hindernis ist im Rahmen dieser Arbeit als Objekt definiert, welches sich vor dem UAV innerhalb einer variablen Gefahrenzone befindet. Beide Methoden richten sich nach den in Kapitel 2 erläuterten Algorithmen und Konzepten. Anhand dieser ist es möglich aus den beiden Bildern des Stereo Systems für jeden Frame die Disparitätenkarte zu berechnen, welche im Anschluss daran in mehreren Schritten zunächst so angepasst wird, dass nicht verwertbare Bereiche dieser entfernt werden.

Im folgenden Kapitel werden beide Methoden detailliert beleuchtet. Zu Beginn wird die zugrunde liegende Klassenstruktur beschrieben. In Abschnitt 4.3 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 4.4.

### 4.1 Applikationsstruktur

Die grundlegende Struktur beider im Rahmen dieser Arbeit entwickelten Methoden ist dieselbe. Die abstrakte Klasse *ObstacleDetection* (siehe Abbildung 8 vererbt alle wesentlichen, für die Erkennung auf Basis von Einzelbildern benötigten Methoden an je eine abgeleitete Klasse, welche eine Methode der Hinderniserkennung repräsentiert. Zu diesen zählen die Funktionen *update* und *detectObstacles*. Die Ausführung beider in jedem Frame ist essenziell notwendig um die benötigten Daten innerhalb jedes Einzelbildes vorliegen zu haben. Dabei dient die *update* Funktion zur Aktualisierung der zugrundeliegenden Teilbereiche der Disparitätenkarte. Anschliessend wird in der Funktion *detec-*

*tObstacles* für jeden Wert überprüft ob sich dieser innerhalb der definierten Gefahrenzone befindet. Zusätzlich zu den Funktionen der Basisklasse besitzt jede Methode eine Funktion zur Initialisierung dieser. Bei dieser wird die eigentliche Struktur der *Subimages* und *Samplepoints* erstellt, sowie die aktuelle Gefahrenzone definiert. Im Fall der *Samplepoint Detection* wird außerdem der Radius dieser festgelegt.

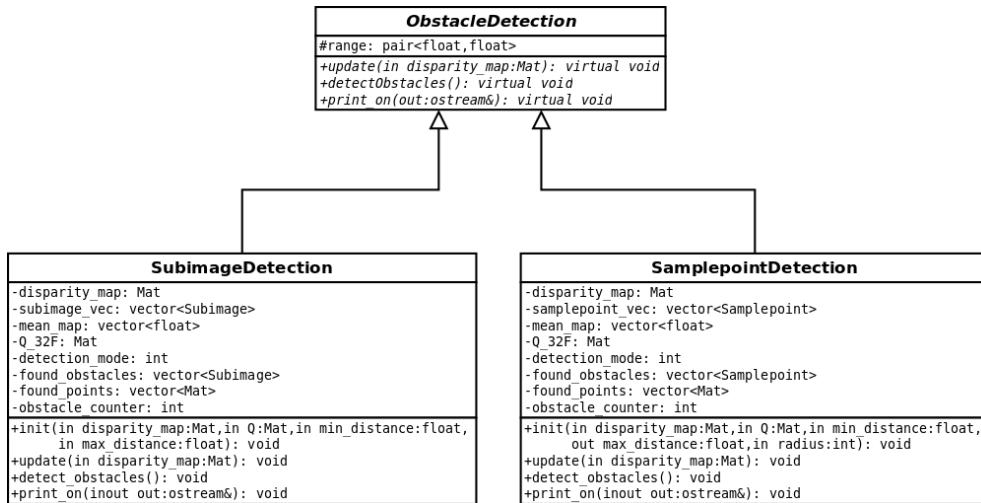


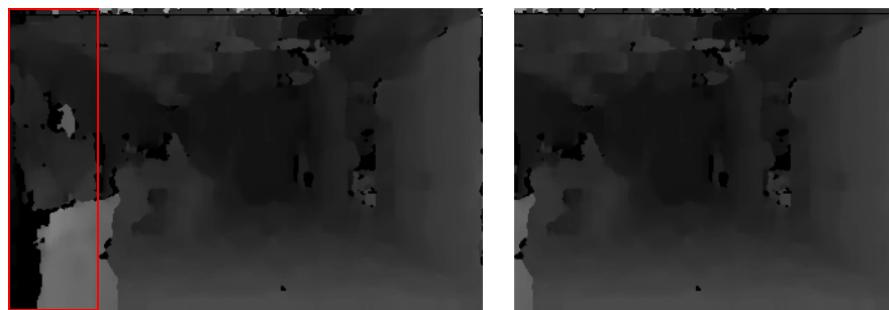
Abbildung 8: Klassenstruktur der Hinderniserkennung

## 4.2 Grundlegende Operationen

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

### ROI der Disparitätenkarte:

Zu Beginn des Algorithmus muss der durch die Translation der Bilder auf der Basislinie verursachte nicht matchbare Bereich entfernt werden. Dies geschieht nach der Wahl der in 2.3.4 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 Disparitätenkarte durch 8 dividierbar sind. Dieser Wert hat sich während der Entwicklung als robuster Wert erwiesen.



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

Abbildung 9 zeigt den *Pixelshift* in der finalen Disparitätenkarte. Der *SGBM* berechnet zwar Werte innerhalb des Bereiches 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:

Dieser Prozess findet in der späteren Initialisierung der Algorithmen Verwendung bei welcher die metrischen Angaben der Erkennungsreichweite in Disparitäten zurückgerechnet werden. Als Referenz wird 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.

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

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

### 4.3 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 Disparitätenkarten in Segmente eingeteilt von welchen in jedem Einzelbild der Mittelwert errechnet wird.

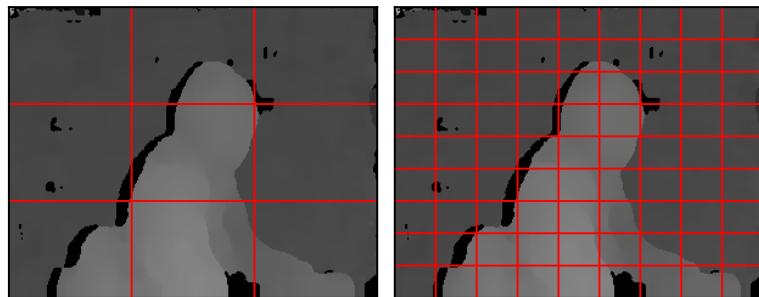
Zu Beginn des Algorithmus, nach der vorherigen Bearbeitung der eigentlichen ROI, wird während der Initialisierung die Segmentierung der Disparitätenkarte festgelegt. Dabei wird für jedes Segment ein eigenes *Subimage* erzeugt. Diese definieren eine weitere ROI innerhalb der Bild-Matrix. *Subimages* selber 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 10 aufzeigt.

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

**Abbildung 10:** Subimage Klasse

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

da jede dieser genauer betrachtet wird (siehe Abbildung 11).



**Abbildung 11:** Intiale Segmentierung der Disparitätenkarte sowie die weitere Unterteilung zur genaueren Bestimmung der Werte der *Subimages*

Um die Hindernisse innerhalb der Segmente zu erkennen wurde auf die Berechnung des Mittelwertes dieser gesetzt. Einerseits, da die Berechnung des Mittelwertes unter Betrachtung des Echtzeit-Aspektes eine ressourcensparende und schnelle Operation ist, andererseits weil jeder Pixel des Bildes 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 Disparitätenkarten 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.

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 Disparitätenkarte definiert. Da die Bildgröße pro Einzelbild

**Algorithm 1** Berechnung des Disparity Medians

---

```

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

```

---

stetig ist besteht keine Nötigkeit die einzelnen Segmente erneut zu generieren. Es genügt also die Mittelwerte eines jeden *Subimages* anhand der neuen Disparitätenkarte zu aktualisieren. Daraufhin wird geprüft ob sich einer oder mehrere der Subimage Mittelwerte innerhalb der Gefahrenzone befinden. Die Gefahrenzone wird zur Initialisierung der Erkennung berechnet. Mit Hilfe der in Abschnitt 4.2 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.

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 Mittelpunktes der *Subimages* wird für jedes dieser die jeweilige 3D-Koordinate (siehe Abschnitt 2.4) berechnet und an der Position des Mittelwertes in die Punktwolke geschrieben.

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

---

**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  $<$  rangemin AND value  $>$  rangemax then
6:       temp_Subimage  $\leftarrow$  Subimage_vector[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

```

---

Frames in denen Hindernisse erkannt wurden.

## 4.4 Samplepoint Detection

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

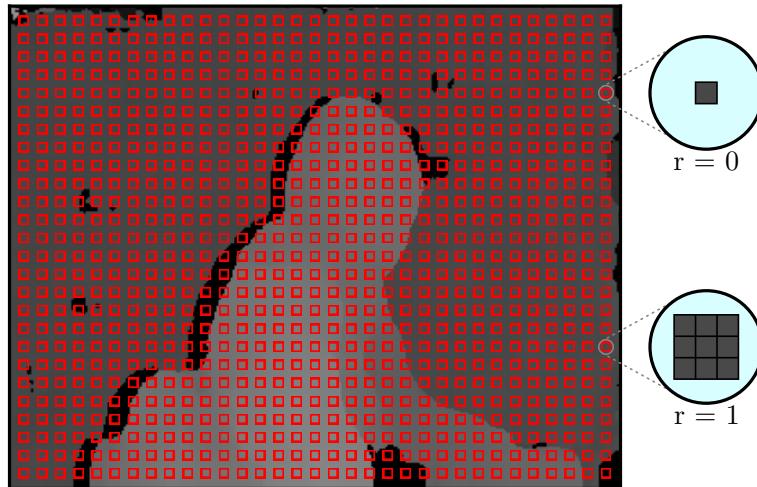
Ebenso wie in der in Abschnitt 1 vorgestellten *Subimage Detection* bedient sich die *Samplepoint Detection* einer vorverarbeiteten Disparitätenkarte zur Hinderniserkennung. Bei der Initialisierung wird die Anzahl der zu berechnenden *Samplepoints* festgelegt. Dabei richtet sich der Wert an der Anzahl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

Ein *Samplepoint* (siehe Abbildung 12) umfasst im entwickelten System entwe-

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

**Abbildung 12:** Samplepoint Klasse

der einen Pixel oder einen Zusammenschluss mehrerer Pixel. Die dabei festgelegte Ausgangskoordinate entspricht dabei entweder dem eigentlichen Punkt, oder dem Mittelpunkt einer festgelegten ROI. Da ein Pixel gerade bei Bildern mit hoher Auflösung weniger Aussagekraft besitzt als der Pixel inklusive der lokalen Umgebung wurde ein *Samplepoint* auf diese Weise definiert. Der Wert eines einzelnen Messpunktes  $S$  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 13 verdeutlicht.



**Abbildung 13:** Darstellung der Samplepoints mit verschiedenen Radien

Nachdem zunächst alle *Samplepoints* initialisiert wurden, werden die jeweiligen Werte pro Frame aktualisiert. Die eigentliche Erkennung erfolgt dabei durch die Erstellung einer Pointcloud. Es wird für jedes *Subimage* pro Einzelbild

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

# Kapitel 5

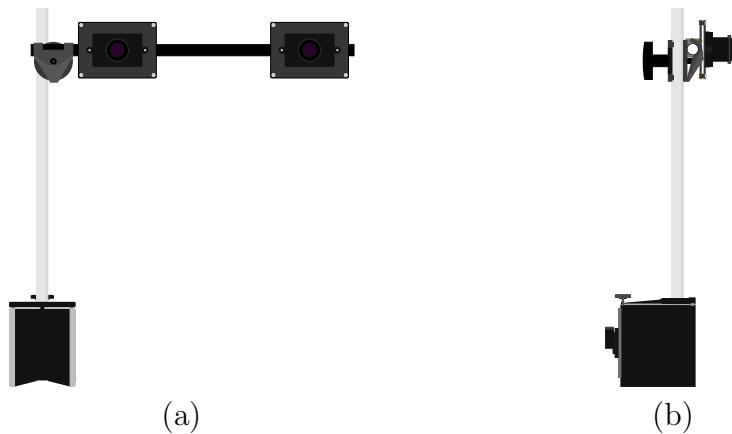
## Evaluation

Um die Funktionsweise beider Methoden zu belegen wurden diese sowohl auf ihre Robustheit als auch 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.

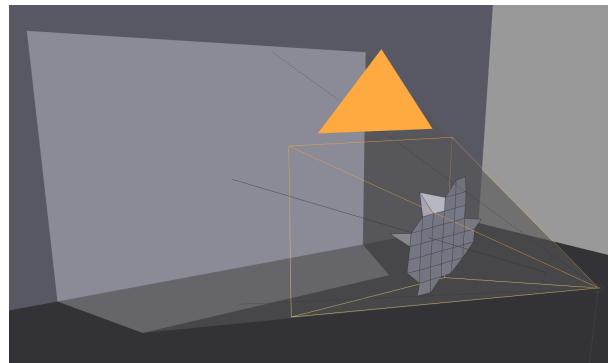
### 5.1 Testsetup

Bei der Durchführung der Tests befinden sich die Kameras statisch im Raum. Um sicher zu stellen, dass sich die Position der Kameras während der einzelnen Versuche nicht verändert wurden diese auf einer Eisenplatte mittels eines Laborstativs magnetisch befestigt (siehe Abbildung 14) 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 Disparitätenkarte sowie eine komplett Punkt-wolke dieser um etwaige Fehler der Algorithmen leichter erkennen zu können, sowie den einzelnen Punktwolken 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 nach der Anwendung der ROI auf die Dispa-



**Abbildung 14:** Ansicht des Kamera frontal (a) und von der Seite (b).



**Abbildung 15:** Im Bild ist das nach der Rektifizierung sowie Beschneidung der ROI erhaltene Sichtfeld visualisiert. Das Mesh innerhalb des Kamera Frustums repräsentiert die gerenderte Pointcloud eines Hindernisses

ritätenkarte erhaltene Sichtfeld (siehe 4.2) beträgt  $50^\circ$  auf horizontaler Achse und  $38^\circ$  vertikal. Dies ist in Abbildung 15 visualisiert.

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 an-

deren 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 durch die Erkennung eines Hindernisses, welches sich über das gesamte Bild ausbreitet, sowie 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 reflektierenden und durchsichtigen Flächen.

Die im Rahmen der Tests aufgenommenen Bilder wurden in einzelnen Fällen in Hinsicht auf bessere Sichtbarkeit bearbeitet. Die aufgenommenen Disparitätenkarten sind aufgrund der Normalisierung und hohen Distanzen oft sehr dunkel. Deshalb wurden die Helligkeit dieser verändert. Weiter Modifikationen der Bilder wurden nicht vorgenommen.

## 5.2 Evaluierung Subimage Detection

Hinsichtlich der Robustheit werden beide Algorithmen nach demselben Schema untersucht. Der in Abschnitt 5.1 beschriebene Testablauf erwähnt 3 verschiedene Hindernisgrößen. Tabelle 5.1 zeigt sowohl die Maße der Hindernisse als auch deren Fläche in  $cm/cm^2$  auf. Weiterhin werden die verwendeten Hindernisse in Abbildung 16 dargestellt.

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

**Tabelle 5.1:** Maße der verschiedenen genutzten Hindernisse

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 Disparitätenkarte sichtbar. Zudem ist ein weiterer Anzeigemodus verfügbar welcher die gefundenen Hindernisse innerhalb der Tiefenkarte markiert (siehe Abbildung 17). Weiterhin kann jede erstellte Hindernis-Pointcloud im Nachhinein gerendert werden um einen Überblick über die Ergebnisse des Algorithmus zu erhalten.



**Abbildung 16:** Verwendete Test Hindernisse

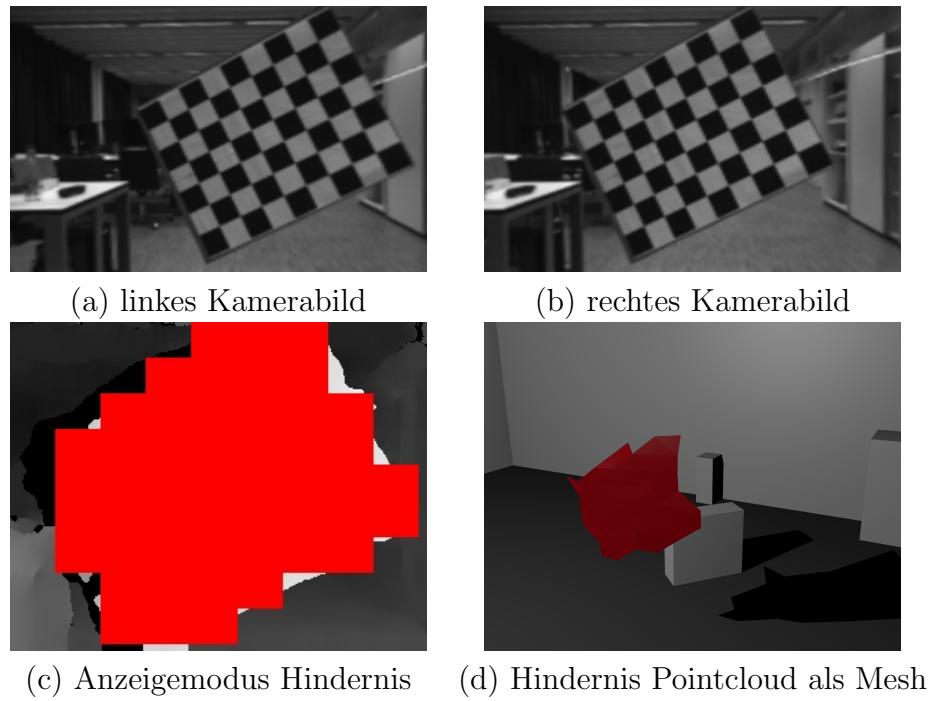
### **Großes Hindernis:**

Im ersten Test wurde das große Hindernis gewählt, dabei wurde mithilfe der dargestellten Hindernisse auf der Disparitätenkarte 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 18 (a) sichtbaren Ergebnisse.

Wie aus diesen zu erkennen wurde in nahezu allen Frames das Hindernis richtig erkannt. Die berechneten Distanzen stimmen mit den real gemessenen überein. Minimale Abweichungen können in diesem Fall als Messungenauigkeit ignoriert werden, da der Mittelpunkt zwischen der minimalen und maximalen Objektdistanz kein statischer Punkt ist. Zwar wurden alle Hindernisse erkannt, jedoch nicht der gesamte Bereich den sie umfasst haben. Wie Abbildung 19 zeigt wurden der obere und untere Bereich des Hindernisses im 5. Frame nicht erkannt. Dies könnte einerseits aus der Rotation des Objektes resultieren, andererseits auch aus einer gewissen Biegung die das physische Objekt mit sich bringt. Auch im in Abbildung 18 (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 vernachlässigbaren 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



**Abbildung 17:** Abbildungen (a) - (c) zeigen die sichtbaren Bilder während der Testaufnahme. (c) zeigt eine als Mesh gerenderte Pointcloud

und gemessener Distanz. Das getestete Objekt befand sich während der Tests an einem durchsichtigen Plexiglas Stab welcher teilweise auch als Hindernis erkannt wurde. 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 Hindernisgröße 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 20(b) sichtbar, das Maximum der realen Distanz resultiert dabei aus dem nicht erkannten Hindernis in Frame 9.

#### Kleines Hindernis:

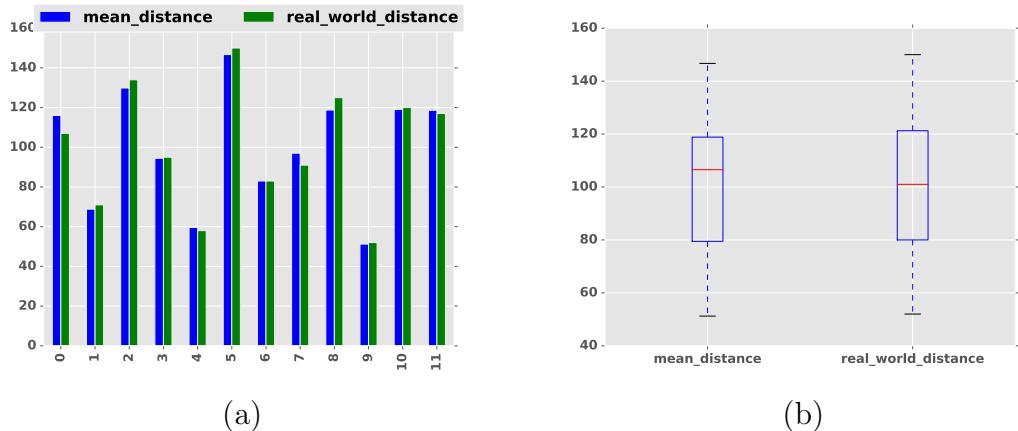


Abbildung 18

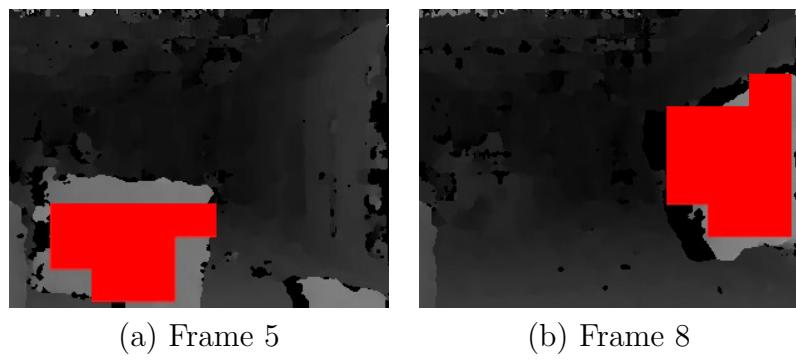


Abbildung 19

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 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 22 (b) dargestellte Boxplot zeigt, das signifikant größere Distanzen erkannt wurden als sie in den realen Messungen vorkommen.

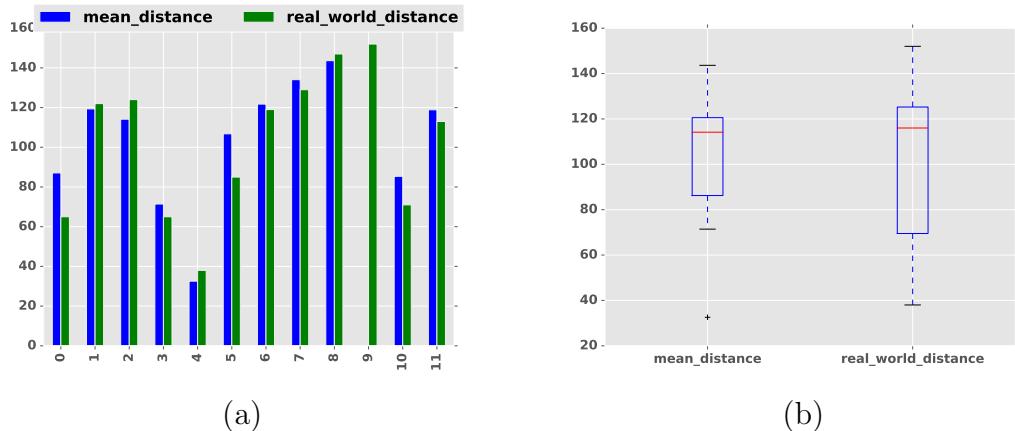


Abbildung 20

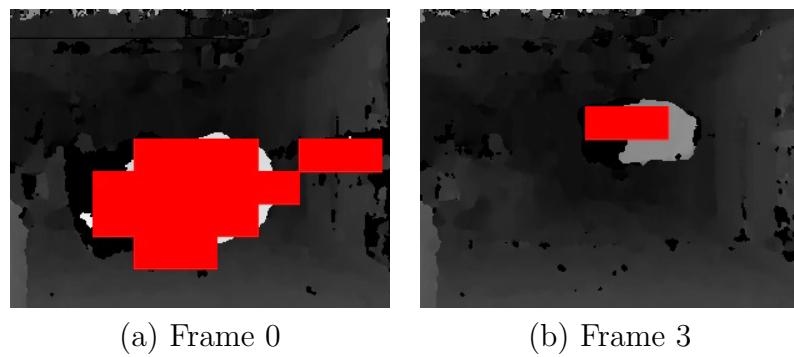


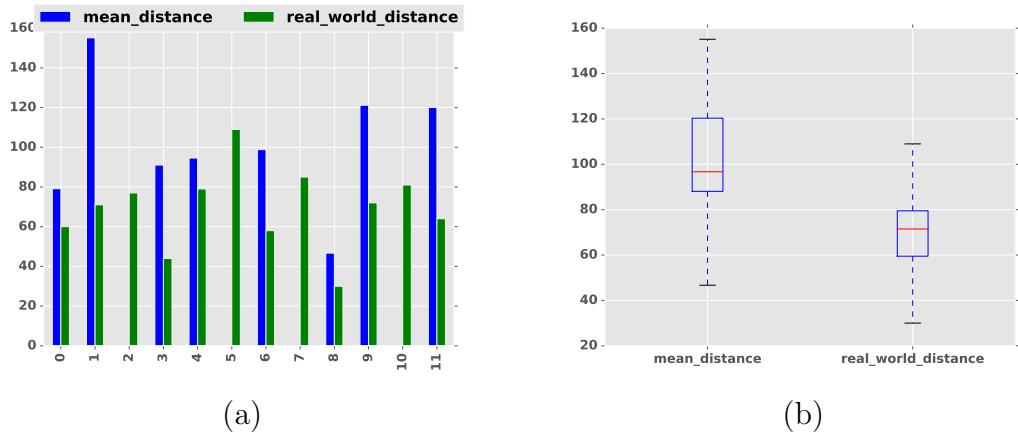
Abbildung 21

### 5.3 Evaluierung Samplepoint Detection

Durch die wesentlich höhere Anzahl an betrachteten Bereichen innerhalb der Disparitätenkarte 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 5.2 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.

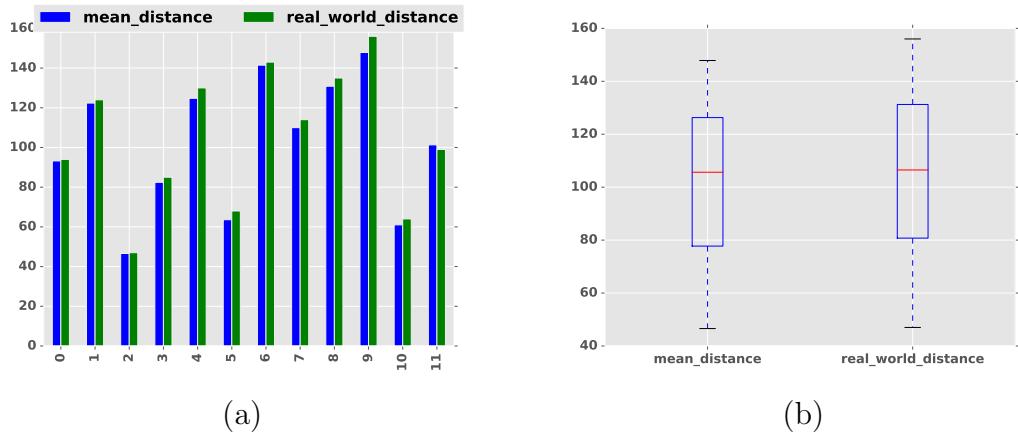
#### Großes Hindernis:

Die Erkennung großer Objekte der Samplepoint Detection weist nahezu keine Fehler auf. Abbildung 23 zeigt das jede der 12 Ausrichtungen des Objektes erkannt wurden. Die dabei auftretenden Differenzen zwischen der berechneten sowie gemessenen Distanz sind auf Messungenauigkeiten zurückzuführen.



**Abbildung 22**

Auch in Grenzbereichen nahe dem Rand des Gefahrenbereichs wurden die Hindernisse erkannt.



**Abbildung 23**

### Mittleres Hindernis:

Auch die Erkennung mittlerer Hindernisse erfolgte in allen Frames ohne signifikante Probleme (siehe Abbildung 24). Bereiche in denen aufgrund fehlender Textur keine Korrespondenz zugeordnet werden konnte wurden aufgrund der hohen Dichte der Samplepoints trotzdem erkannt. Wie Abbildungen 25 (a) und (b) zeigen wurden nicht gematchte Bereiche auch nicht als Hindernis erkannt, die umliegenden texturierten Bereiche überwiegen jedoch und sorgen daher für eine robuste Detektion.

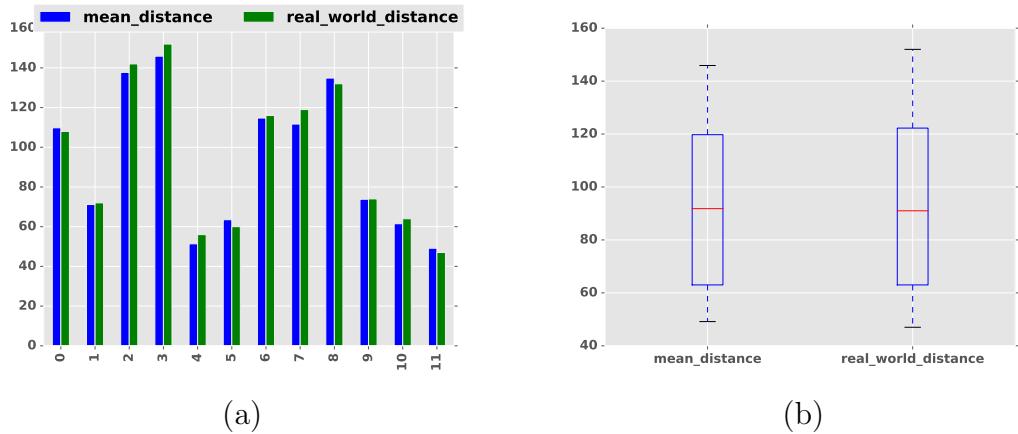


Abbildung 24

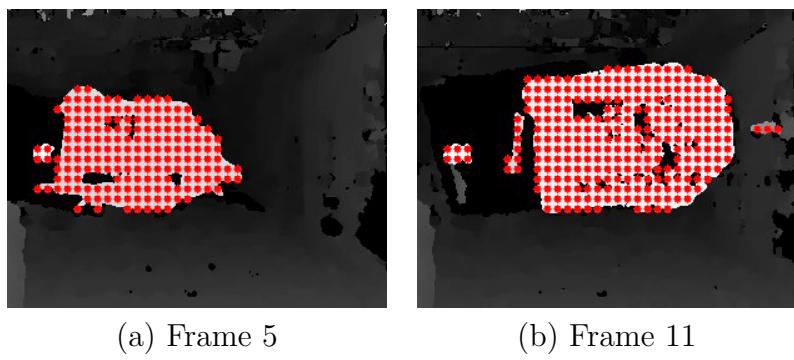


Abbildung 25

Die bereits beschriebene hohe Dichte der Samplepoints und deren geringe Größe hilft auch hier wieder den Stab als Hindernis zu erkennen (siehe Abbildung 25 (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 26 (a) belegt diese Aussage. Es wurden alle 12 Hindernisse ohne signifikante Differenz in den Distanzen erkannt. Trotz dessen ist die Detektion dieser bei Entfernung von mehr als 60 Zentimetern stark von der Orientierung des Objektes, dem Hintergrund sowie den verwendeten Parametern zur Berechnung der Disparitätenkarte abhängig. Da die geringe Auflösung der Kameras keine detaillierte Aufnahme der Textur des Hindernisses zulässt ist es möglich das das Objekt aufgrund der Blockgröße des *SGBM* mit dem Hintergrund zusammen

gematcht wird.

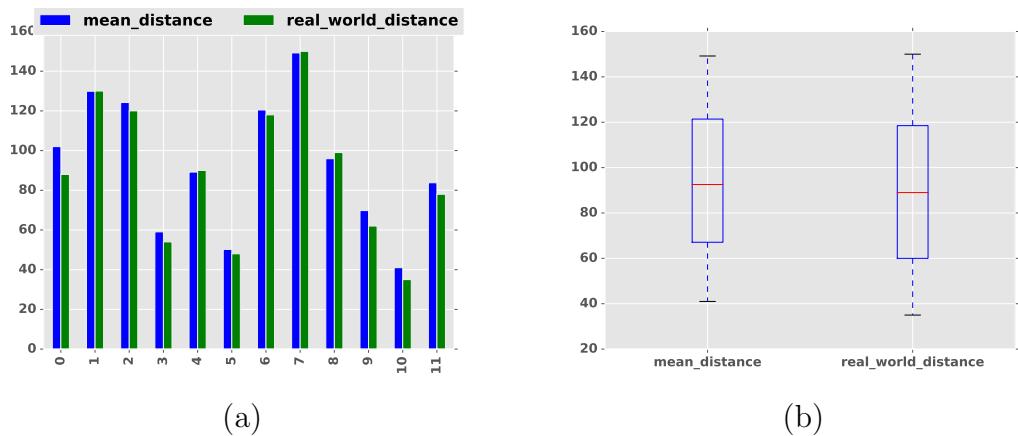


Abbildung 26

Weiterhin ist auch die statistische Auswertung der Distanzen in Boxplot 26 (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.

## 5.4 Weitere Versuche

Hinsichtlich der verschiedenen möglichen Faktoren welche eine erfolgreiche Erkennung beeinflussen können wurden im Rahmen der Evaluation beider Algorithmen weitere Tests durchgeführt. Jene umfassen die Erkennung kleiner Hindernisse unter Veränderung der Größe des Matching-Blocks. Weiterhin wird die Erkennung kritischer Umgebungen (durchsichtige und reflektierende Flächen) getestet. Anschließend erfolgt ein weiter Versuch welcher auf die Veränderung des Samplepoint Radius und die damit verbundenen Änderungen in der Erkennung abzielt. Das während des Tests genutzte Setup wurde während der Aufnahme nicht verändert. Auch das Objekt befand sich für jede Blockgröße an der jeweiligen Distanz. Zur Durchführung der Tests wurde das in Abschnitt 5.2 und 5.3 genutzte kleine Hindernis verwendet.

### 5.4.1 Erkennung unter 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 5.2:** Ausgewertete Daten des Subimage Sets

Wie aus Abbildung 5.2 ersichtlich wird hat eine Änderung der Blockgröße keine Auswirkungen auf die Erkennung des Objektes. Die berechneten Entfernung werden erheblich von den Pixeln des Hintergrundes beeinflusst, was in den verzerrten berechneten Distanzen resultiert. Dabei auftretende Differenzen zwischen den real gemessenen Entfernung sowie den berechneten resultieren aus Interferenzen während der Neuberechnung der Disparitätenkarte in jedem Frame.

#### Samplepoint Detection:

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 Entfernung bei 13 Pixeln Blockgröße am

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 5.3:** Ausgewertete Daten des Subimage Sets

geringsten. Während der einzelnen Versuche wurde der Befestigungsstab des Hindernisses ebenfalls erkannt (siehe Abbildung 27).



**Abbildung 27:** Testset unter Verwendung einer Blockgröße von 13 Pixeln in verschiedenen Distanzen (50 cm bis 150 cm v.l.n.r.)

Dabei lässt sich in allen Bildern welche im Abstand von 100 Zentimetern aufgenommenen 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 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 Disparitätenkarte.

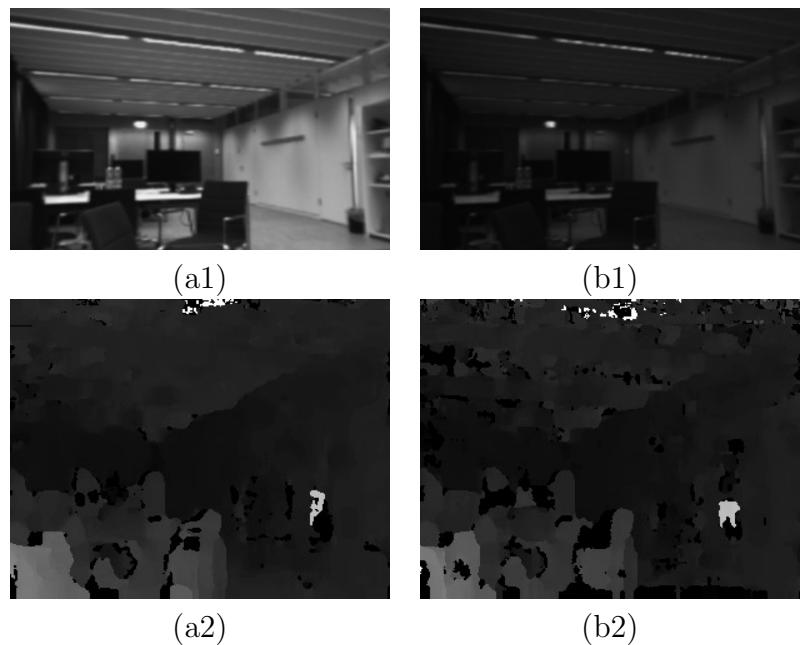
#### 5.4.2 Erkennung von reflektierenden und durchsichtigen Bereichen

Die Erkennung von Hindernissen bei reflektierenden sowie durchsichtigen Flächen wurde mit drei verschiedenen Tests untersucht. Zunächst wurde untersucht wie sich das System bei spiegelnden Flächen (etwa Fensterglas) verhält. Anschliessend wurde die Erkennung bei reflektierenden Flächen untersucht sowohl im  $90^\circ$  Winkel, also mit direkter Sicht auf das zu erkennende Hindernis als auch unter diversen anderen Winkeln. Weiterhin wurde untersucht inwiefern sich das System bei einer Kombination beider Problemstellungen verhält. Die verwendete Methode zur Hinderniserkennung ist in allen folgenden Tests die Samplepoint Detection. Die Gefahrenzone wurde auf 0,1 bis 1,0 Meter definiert.

Zur Untersuchung reflektierender Flächen wurde der spiegelnde Bildschirm eines Fernsehers genutzt. Da die Berechnung von Disparitätenkarten auch unter nicht optimalen Lichtverhältnissen robuste Ergebnisse liefert (Abbildung 28) stellt auch die Schwarze Grundfarbe des spiegelnden Objektes kein Konflikt dar. Trotz dessen wurde die Belichtungszeit der Kameras auf 60 - 70 ms gesetzt, was zwar die Performance beeinflusst jedoch die erhaltenen Bilder erheblich verbessert.

Die aufgenommenen Ergebnisse zeigen auf das Hindernisse erkannt wurden, jedoch ist der Ursprung dieser nicht eindeutig zu erkennen. Abbildung 29 stellt die berechneten und aufgenommenen Distanzen gegenüber. Aus diesen sind ebenfalls deutlich die Fehler innerhalb der Berechnung zu erkennen.

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



**Abbildung 28:** Vergleich zwischen einer hellen aufgenommenen Szene in (a1) und (a2) sowie den dazu gehörigen Disparitätenkarten in (a2) und (b2).

Reflexionsebene gespannt wird. Abbildung 30 stellt die erlangten Ergebnisse dar.

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

Eine Kombination von Reflexion und Durchsichtigkeit brachte ebenfalls keine validen Ergebnisse. In den ersten fünf Versuchen wurden zwar Hindernisse erkannt, jedoch weitaus näher, als sie sich eigentlich befunden haben. Zudem wurden Hindernisse erkannt obwohl sich keines innerhalb der Gefahrenzone befunden hat. Ab Versuch 5 befand sich das Fenster innerhalb der Gefahrenzone. Die in Abbildung REF abgebildeten Disparitätenkarten zeigen jedoch das zumeist nur wenige Punkte erkannt wurden welche nicht das Objekt als ganzes abbilden. Die gefundenen Hindernisse sind das Resultat eines reflektierten Lichts auf dem Glas. Abbildung 31 zeigt eben diese Ergebnisse.

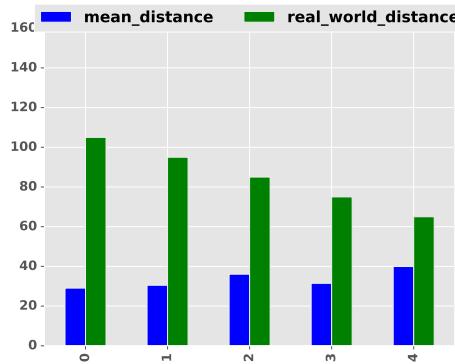


Abbildung 29

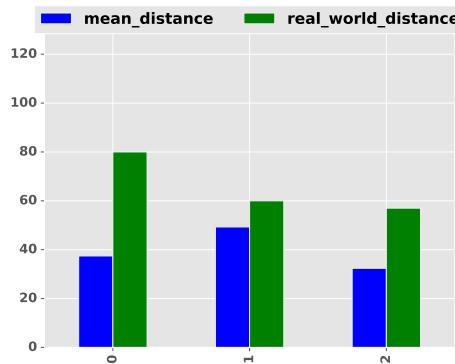


Abbildung 30

### 5.4.3 Erkennung unter Veränderung des Samplepoint Radius

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

Die Veränderung des Samplepoint Radius brachte lediglich bei der Verwendung eines einzelnen Pixels eine wahrzunehmende Veränderung mit sich. Das Hindernis wurde bei größeren Entfernungen zwar besser erkannt, jedoch wurden

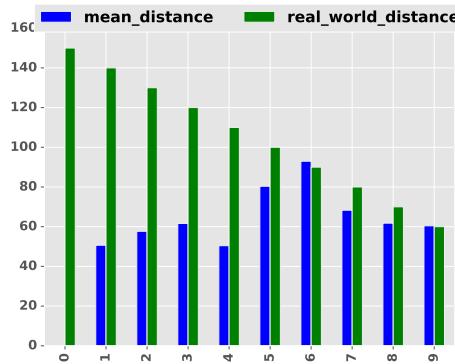


Abbildung 31

ebenfalls Bereiche mit Fehlinformationen als Hindernis gedeutet.

## 5.5 Diskussion

Die in den Abschnitten 5.2 und 5.3 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 Mittelwertes der Matrizen auch der größte Konflikt. Weist der Hintergrund eines Objektes sehr geringe Disparitäten auf, so besteht die Möglichkeit das der Hintergrund den Mittelwert eines einzelnen Subimages insofern beeinflusst, dass das Hindernis (innerhalb dieser Submatrix) nicht mehr erkannt werden kann. Dies ließ sich bei nahezu allen Versuchen und Hindernisgrößen beobachten. Im Fall der großen und mittleren Hindernisse stellt dies keinen 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 5.3 angedeutet liegt das Problem dabei in der zugrunde liegenden Disparitätenkarte. Der hintere Bereich der Szene unterscheidet sich farblich nicht signifikant vor der Farbe des Hindernisses. Zudem ist das Objekt bei einer Entfernung von 150 Zentimetern so klein das es höchstwahrscheinlich durch die verwendete Blockgröße nicht erkannt oder mit benachbarten Bereichen verbunden wird.

In Kapitel 4 wurden die Abläufe und Funktionsweisen der einzelnen Algorithmen detailliert beschrieben. 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 maßgeblich. Jegliche Differenz zwischen den erkannten und berechneten Distanzen resultiert daher eher aus der zugrunde liegenden Disparitätenkarte. 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.

Die Erkennung reflektierender und durchsichtiger Flächen führte zu gewissen Konflikten. Hindernisse wurden in nahezu jedem Versuch erkannt, jedoch nur in seltenen Fällen korrekt. Dies kann aus der homogenen Textur der verwendeten Bilder resultieren, welches jedoch unwahrscheinlich ist (siehe Abbildung 28). Der genaue Ursprung der erkannten Hindernispunkte ist nahezu unmöglich zu identifizieren, da die berechnete Disparitätenkarte keine genaue Lokalisierung von Formen der Szene zulässt.

Die Anpassung des Subimage Radius brachte ebenfalls keine signifikanten Verbesserungen in der Erkennung. Zwar wurde Hindernis 3 bei größeren Entfernungen besser erkannt, jedoch ging damit auch die Erkennung von Bereichen welche falsche Werte enthalten als Hindernis einher. Dieser ungewünschte Nebeneffekt kann bei einer realen Anwendung zu Falschentscheidungen von Seiten

der Hindernisvermeidung führen und somit im schlimmsten Fall zum Ausfall des UAV.

# Kapitel 6

## 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.

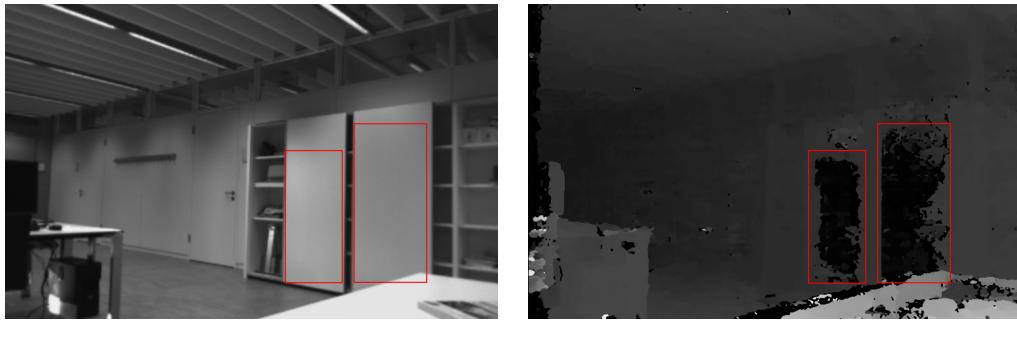
Das folgende Kapitel beleuchtet noch bestehende Limitierung in der Erstellung der Disparity Map sowie der Erkennung von Hindernissen. Weiterhin werden diese in Abschnitt 6.1 diskutiert. Zudem werden in 6.2 sowohl eigene, als auch literarisch belegte Lösungsansätze präsentiert.

### 6.1 Bestehende Limitierungen und Lösungsansätze

Die Validierung der erkannten Hindernisse ist ein kompliziertes Problem in der autonomen Hinderniserkennung. Durch etwaige äußere Einflüsse wie die Veränderung der Lichtverhältnisse kann die Berechnung der Tiefenkarte fehlerhaft sein. Dies kann im Fall eines autonomen Flugs dazu führen, dass das 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 fehlerhafte Disparitäten 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 mit einbezogen werden, und somit dieses Konzept unbrauchbar werden lassen.

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 falsche Disparitäten zur Folge haben. In diesem Fall wird ein Hindernis erkannt, obwohl sich keines an dieser Position befunden hat. Abbildung 32 zeigt deutlich, in welchen Bereichen der *SGBM* aufgrund fehlender Textur keine Korrespondenzen finden konnte.



**Abbildung 32:** 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 Be-rechtigung hat, andererseits auch bei niedrigen Flughöhen zu einer fehlerhaften Erkennung als Hindernis führt.

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

## 6.2 Diskussion

Ein Ansatz, erkannte Hindernisse zu validieren ist, sich auf die letzte bekannte Position des Objektes zu berufen und im nächsten Einzelbild zu überprüfen, ob es sich noch immer an derselben Position befindet. Dabei würde für jedes Subimage/ jeden Samplepoint ausgewertet werden, ob sich das Hindernis

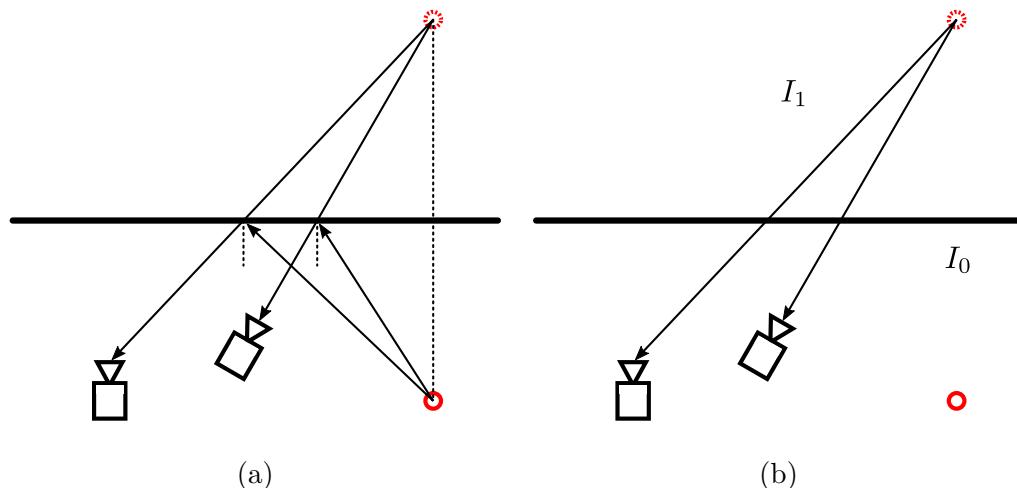
noch innerhalb dessen befindet. Jedoch schränkt dies die Erkennung bewegter Objekte stark ein. Ist die Geschwindigkeit eines Hindernisses so hoch, das in mindestens zwei aufeinanderfolgenden Frames kein Objekt im selben Subimage/Samplepoint erkannt werden kann so würde das System kein Hindernis erkennen 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 Reflexionsmodells nach Hero (Abbildung 33(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 33(b)).



**Abbildung 33:** (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 6.1 angeschnittener Punkt ist die Erkennung des Bodens als Hindernis. Dabei stellt sich die Frage, ob eine Erkennung als Hindernis gewünscht ist oder nicht. Im Falle einer geringen Flughöhe wird der Boden ebenfalls als Hindernis erkannt. Dies kann einerseits eine Absicherung dafür sein, dass die durch interne Sensoren ermittelte Flughöhe korrekt ist, andererseits kann es dazu führen, dass der Algorithmus zur Vermeidung von Hindernissen versucht, 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 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 Disparitätenkar-  
te 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 Berechnung der Disparitätenkarte 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  $12 \frac{m}{s}$  zu bewegen und für jeden zurückgelegten Meter 2 berechnete Disparitätenkarten 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 der *Subimage Detection*

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 der *Samplepoint Detection*

Gerade die ermittelte Framerate in 1(b) gibt Anlass zu der Annahme, dass die Samplepoint Detection mehr Rechenleistung benötigt. Dies resultiert aus der Anzahl der zu betrachtenden Objekte. Im Vergleich zur Subimage Detection werden nicht nur 81 Bereiche betrachtet, sondern 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 19,6 Bildern/s.

Zur Steigerung der Performance wurden in einem weiteren Test die initialen Bilder zur Berechnung der Disparitätenkarte um den Faktor 1/2 skaliert, wobei alle restlichen verwendeten Parameter unverändert waren. Die dadurch erhaltenen Ergebnisse sind in den Tabellen 7.4 und 7.5 abgebildet.

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

**Tabelle 7.4:** Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate mit skalierten Ausgangsbildern der *Subimage Detection*

Szenario	Zeit pro Frame (Detection)	Detecion fps
1(a)	0.0015	634.28
1(b)	0.0023	418.96
1(c)	0.0016	612.04
2(a)	0.0004	2132.13
2(b)	0.0015	666.52
2(c)	0.0024	403.07

**Tabelle 7.5:** Gemessene Einzelbilder pro Sekunde sowie Gesamtframe-rate mit skalierten Ausgangsbildern der *Samplepoint Detection*

Aus diesen ist deutlich ersichtlich, dass eine verringerte Größe der Ausgangsbilder die Framerate immens steigert. Dies resultiert hauptsächlich aus der

Berechnung der zugrundeliegenden Disparitätenkarte. Diese ist in dieser Konfiguration auf durchschnittlich 94 Einzelbilder pro Sekunde angestiegen. Dieser Wert ist jedoch nur in Abhängigkeit der Wiederholrate der Aufnahme zu erreichen. Im horizontalen sowie vertikalen Binning Modus liegt diese durchschnittlich bei 60 – 66 Bilder pro Sekunde. Für eine Erkennung in Szenario 1(c) ergeben sich demnach die folgenden möglichen Werte bei beiden Methoden:

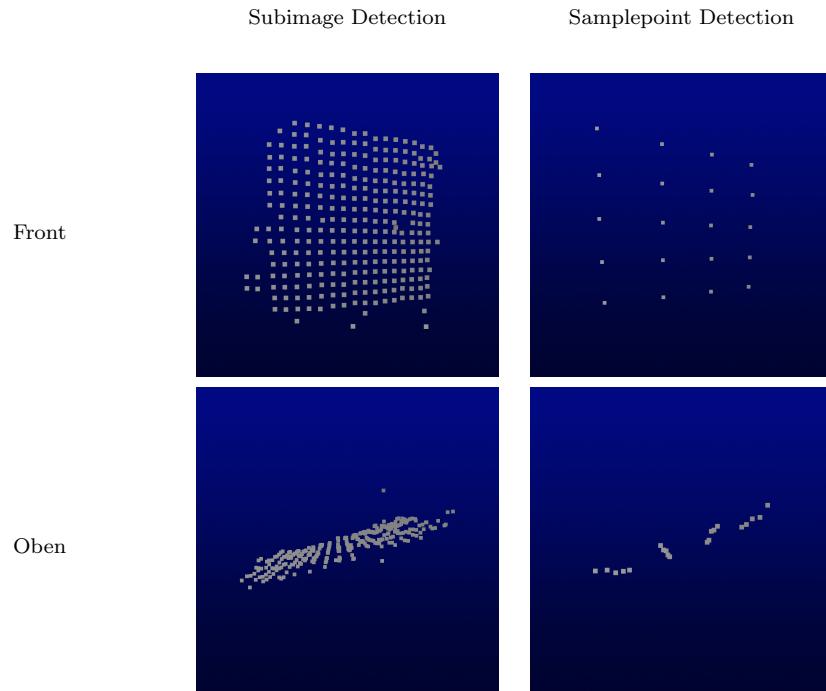
Subimage Detection:	75,53
Samplepoint Detection:	77,28

### 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 darstellen. 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-



**Tabelle 7.6:** Gespeicherte Hindernis Punktfolke desselben Objektes beider Methoden.

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 5.5 erläuterten Ergebnisse bei 13 Pixeln brachte eine geringe Verbesserung der berechneten Distanz.

# Kapitel 8

## Fazit und zukünftige Arbeiten

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

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

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

Weiterhin ist die Integration des entwickelten Systems in einen ROS Node zur weitergehenden Verwendung durch SLAM Algorithmen ein wichtiger Bestandteil zukünftiger Arbeiten. Damit verbunden ist gleichzeitig die Entwicklung eines Systems zur Hindernisvermeidung unter Verwendung der erstellten Punktwolken. In Kombination mit der aktuellen Position sowie Rotation der Drohne im Raum ist die darauf folgende Kartographierung gefundener Hindernisse möglich.

## *KAPITEL 8. FAZIT UND ZUKÜNFTIGE ARBEITEN*

---

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

# Literaturverzeichnis

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