Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Max Rostock

Konzept und Entwicklung einer Mikrocontroller-basierten Time-of-Flight-Sensorik zur autonomen Navigation im Nahbereich

Fakultät Technik und Informatik
Department Fahrzeugtechnik und Flugzeugbau

Faculty of Engineering and Computer Science
Department of Automotive and Aeronautical Engineering

Max Rostock

Konzept und Entwicklung einer Mikrocontroller-basierten Time-of-Flight-Sensorik zur autonomen Navigation im Nahbereich

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Mechatronik
am Department Fahrzeugtechnik und Flugzeugbau
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Erstprüfer: Prof. Dr.-Ing. Lutz Leutelt
Zweitprüfer: Prof. Dr. Pawel Buczek

Abgabedatum: 01.04.2020

Zusammenfassung

Max Rostock

Thema der Bachelorthesis

Konzept und Entwicklung einer Mikrocontroller-basierten Time-of-Flight-Sensorik zur autonomen Navigation im Nahbereich

Stichworte

Time-of-Flight, Sensorik, Mikrocontroller, autonome Navigation, Matlab, 3D-Druck, elektronischer Blindenhund

Kurzzusammenfassung

Diese Arbeit umfasst die Auswahl, Inbetriebnahme und Evaluation einer Time-of-Flight-Kamera zur autonomen Navigation eines „elektronischen Blindenhundes“.
Mit Hilfe der Time-of-Flight-Kamera soll der „elektronische Blindenhund“ die Umgebung und potenzielle Hindernisse erfassen, sodass der Anwender sicher geführt werden kann. Hierfür wird Mikrocontroller-Software zum Auslesen der Sensordaten entwickelt und es werden diverse Messungen durchgeführt. Des Weiteren wird Software zur Visualisierung der Daten entwickelt und es werden Bauteile zur Integration des Sensors in einen Prototyp konstruiert und gefertigt.

Max Rostock

Title of the paper

Concept and development of a microcontroller-based time-of-flight sensor technology for autonomous close-range navigation.

Keywords

time-of-flight, sensor technology, microcontroller, autonomous navigation, Matlab, 3D printing, electronic guide dog

Abstract

This thesis includes the selection, installation and evaluation of a time-of-flight camera for autonomous navigation of an “electronic guide dog”. With the time-of-flight camera the “electronic guide dog” should capture its surroundings and detect potential obstacles so that the user can be guided safely.
Therefore, microcontroller software for reading sensor data is developed and various measurements are performed.
Furthermore, software for data visualization is developed and mechanical components for integration of the sensor in a prototype are designed and manufactured.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

CCD charge-coupled device

CCS Code Composer Studio

CRC cyclic redundancy check

cwTOF continuous wave modulated Time-of-Flight

EGD Eletronic Guide Dog

FoV Field-of-View

NFOV Narrow-Field-of-View

pTOF pulse modulated Time-of-Flight

SLAM simultaneous localization and mapping

ToF Time-of-Flight

UART Universal Asynchronous Receiver Transmitter

WFOV Wide-Field-of-View

1 Einleitung

1.1 Motivation

Sich selbstständig und ohne fremde Hilfe im Alltag zu bewegen birgt für blinde und sehbehinderte Menschen, besonders in Großstädten, oft erhebliche Schwierigkeiten und Gefahren.

Blindenführhunde sind ein bewährtes Mittel, um hierfür Abhilfe zu schaffen. Allerdings ist die Ausbildung dieser mit enormem Aufwand und dementsprechend hohen Kosten von etwa 20.000 bis 30.000 Euro verbunden, wenngleich diese in Deutschland unter Umständen von der jeweiligen Krankenkasse getragen werden können [31].

Des Weiteren kommt es für manchen blinde und sehbehinderte Menschen aus unterschiedlichen persönlichen Gründen, wie Allergien oder der mangelnden Bereitschaft einen Hund im Haushalt zu halten, nicht in Frage einen Blindenführhund in Anspruch zu nehmen. So besitzen in Deutschland nur etwa ein bis zwei Prozent der Blinden einen Blindenführhund [31].

In Zukunft könnten vermehrt technische Lösungen eingesetzt werden, um blinden und sehbehinderten Menschen die Orientierung und das Vermeiden von Unfällen zu erleichtern und ihnen somit ein großes Stück Mobilität und Selbstständigkeit zu ermöglichen. Ein solcher Eletronic Guide Dog (EGD) muss hohe Anforderungen an Zuverlässigkeit, Robustheit und Ausfallsicherheit erfüllen. Außerdem muss er kompakt und leicht zu transportieren und dennoch mechanisch stabil sein. Des Weiteren muss ein solches System seine Umgebung zuverlässig erfassen, Hindernisse zuverlässig erkennen und dem Anwender anschließend einen sicheren Weg vorgeben können. Der technische Fortschritt in Bereichen wie Sensorik, autonomes Fahren und Robot-Vision/Bildverarbeitung brachte zahlreiche Konzepte hervor, welche in einem elektronischen Blindenhilfsmittel Anwendung finden könnten.

Die technische Lösung zur Erfassung der Umgebung und der Erkennung von Hindernissen darf hierbei unter anderem in der Genauigkeit und Wiederholbarkeit den Fähigkeiten eines gut ausgebildeten Blindenführhundes in nichts nachstehen. Zu ebendieser Erfassung von Umgebung und Hindernissen sollen die Ergebnisse dieser Arbeit beitragen.

1.2 Vorgehensweise

Im Verlauf dieser Arbeit wird versucht ein Konzept zur Erfüllung der in Kapitel 3 aufgelisteten Anforderungen zu entwickeln. Dabei wird erläutert, welche Technologien hierfür geeignet sein könnten, deren Grundlagen erklärt und begründet warum die Entscheidung auf die letztendlich verwendete Hardware fiel. Es wird im Detail auf die verwendete Hardware eingegangen, sowie deren Vor- und Nachteile herausgearbeitet.

Weiterhin wird die Implementierung der einzelnen Funktionalitäten beschrieben. Hierbei wird auf die mechanische, elektrische und softwaretechnische Integration des Teilsystems in das Gesamtsystem eingegangen. Dies umfasst die Verkabelung der Hardware, die Entwicklung von Software zum Auslesen und Visualisieren von Sensorsignalen, sowie die Entwicklung und Fertigung von mechanischen Bauteilen zur Montage der Hardware am bestehenden Prototyp.

Es wird das entwickelte Konzept evaluiert, sowie Messergebnisse präsentiert und ausgewertet.

Schlussendlich wird ein Fazit gezogen, in dem auch auf die Grenzen der bisherigen Implementierung eingegangen und ein Ausblick auf mögliche zukünftige Verbesserungen vorgenommen wird.

Im Gegensatz zum Fließtext dieser Arbeit sind Source-Code, inklusive Kommentar sowie Text in Diagrammen zur Software in englischer Sprache verfasst, da dies in vielen Fällen gängige Praxis ist und einer größeren Gruppe potenzieller Entwickler das Verständnis erleichtert.

2 Grundlagen

Es existieren diverse Verfahren, mit denen Sensoren die Entfernungen zu Objekten in ihrer Umgebung bestimmen können. Einige dieser Verfahren werden in diesem Kapitel aufgeführt und kurz erläutert. Im Vorfeld wurde bereits entschieden, dass Ultraschall aufgrund diverser Nachteile vorerst nicht als primär zu verwendendes Sensorsystem für diese Arbeit in Betracht gezogen wird. Der Fokus lag in den Betrachtungen auf optischen oder radarbasierten Sensoren.

2.1 Time-of-Flight-Technologie

Time-of-Flight (ToF)-Kameras sind eine Form von 3D-Kameras, welche im Gegensatz zu herkömmlichen 2D-Kameras auch Tiefeninformation ermitteln können. Es wird dafür das sogenannte Laufzeitverfahren verwendet, bei dem die betrachtete Szene beleuchtet wird und aus der zeitlichen Verzögerung, durch das Zurücklegen der Strecke von der Kamera zum Objekt und zurück, die jeweilige Distanz für jeden einzelnen Bildpunkt bestimmt wird [30].

Aufgrund der hohen Geschwindigkeit von Licht sind besonders schnelle Schaltungen notwendig um das Time-of-Flight-Verfahren implementieren zu können [7]. Es handelt sich daher um eine relativ junge Technologie, die durch die Entwicklung schneller Halbleiterbauteile möglich geworden ist. So begannen Produkte, die das Time-of-Flight-Verfahren in zivilen Anwendungen implementieren, um das Jahr 2000 auf den Markt zu kommen.

Es kann zwischen zwei Arten des Time-of-Flight-Verfahrens unterschieden werden: pulse modulated Time-of-Flight (pTOF) und continuous wave modulated Time-of-Flight (cwTOF) [7]. Beide Varianten besitzen einen Sender, der die Szene beleuchtet und einen Empfänger, der das reflektierte Licht detektiert. Beim pTOF-Verfahren wird ein einzelner Lichtpuls ausgesendet und die Zeit bis der reflektierte Lichtpuls den Empfänger erreicht gemessen. Im Gegensatz dazu wird beim cwTOF-Verfahren ein kontinuierliches

Signal ausgesendet und mit Hilfe der Phasenverschiebung zwischen ausgesendetem und empfangenem Lichtsignal die Entfernung bestimmt.

In [4] werden unter anderem folgende Einflussfaktoren bei der Messung mit einer ToF-Kamera genannt:

- Mehrfach-Reflexionen: Das reflektierte Licht kann unter Umständen auf mehreren Wegen zurück zum Empfänger gelangen. Dies tritt zum Beispiel auf, wenn das Licht auf eine Ecke im Raum trifft. Dieser Effekt kann zu Problemen in der Messung führen.
- Streulicht: Streulicht tritt auf wenn sich helle Oberflächen in der Nähe befinden. Bei der Helligkeitsmessung einer Kamera führt Streulicht zu verringerten Kontrasten.
- Umgebungslicht: Während künstliches Licht mit einem geringen Spektrum unproblematisch für die ToF-Messung sein sollte, kann Sonnenlicht aufgrund seines sehr großen Spektrums durchaus einen negativen Einfluss auf die Messung haben. Auch ein optischer Bandpassfilter wird also einen Teil des Sonnenlichts durchlassen. Problematisch ist besonders Umgebungslicht nahe des Infrarotbereichs, in dem die ToF-Kamera arbeitet.
- Hohe Reflexion und Transparenz: Die Entfernungen von spiegelnden und transparente Oberflächen sind meist nicht sicher zu bestimmen.
- Helle, diffus reflektierende Oberflächen eignen sich besonders gut zur Entfernungsbestimmung.

2.2 CCD-Sensoren

Als Empfänger in ToF-Kameras können sogenannte charge-coupled device (CCD)-Sensoren zum Einsatz kommen.

CCD-Sensoren sind Halbleiterbauelemente, welche im Kern aus einer Anordung von Photodioden bestehen [29]. Das Eintreffen von Licht sorgt, gemäß dem inneren photoelektrischen Effekt, für das Entstehen von Elektronen-Loch-Paaren. Eine angelegte Spannung bewirkt eine Ladungstrennung. Die vorhandene Ladung wird zwischengespeichert und ihre Menge beim Auslesen des Sensors bestimmt. Aus der bestimmten Ladungsmenge kann auf die Menge des zuvor eingefallenen Lichts geschlossen werden.

2.3 Radar-Sensoren

Radarsensoren bestimmen die Distanz zu einem Objekt ebenfalls über ein Laufzeitverfahren. Hierfür werden elektromagnetische Wellen versendet und deren Reflexion empfangen [5]. Es kann außerdem über den Dopplereffekt die Geschwindigkeit eines Objekts bestimmt werden [33]. Radarsensoren können bei schlechten Sichtverhältnissen einen Vorteil gegenüber optischen Systemen haben.

2.4 Vergleich der Verfahren

Als Vorteil der Laufzeitmessung mit Licht ist, im Vergleich zum anfangs erwähnten Ultraschall, die hohe Lichtgeschwindigkeit zu nennen. Daraus resultierend lassen sich hohe Bildraten erreichen, wodurch besonders dynamische Szenen besser erfasst werden können. Allerdings verfügen Radarsensoren, durch die Verwendung von elektromagnetischen Wellen, ebenfalls über den Vorteil der hohen Geschwindigkeit. Außerdem sind diese weniger anfällig gegenüber schlechten Sichtverhältnissen, als es optische Verfahren wie ToF-Kameras sind. Ein großer Vorteil einer ToF-Kamera ist, dass ein detailliertes Bild der betrachteten Szene erstellt wird. Es kann nicht nur auf das Vorhandensein von Hindernissen hingewiesen werden, sondern zusätzlich können die Konturen der Hindernisse erfasst werden. Es müsste also ein Radarsensor gefunden werden, der ebenfalls solch ein detailliertes Bild seiner Umgebung erzeugen kann, um einer ToF-Kamera in der Anwendung in einem Blindenhilfsmittel überlegen zu sein.

3 Anforderungen

Das Thema dieser Bachelorarbeit entstand im Rahmen des Vorhabens an der HAW Hamburg einen elektronischen Blindenführhund zu entwickeln. In der Aufgabenbeschreibung wurden folgende Anforderungen an die zu erarbeitende Lösung gestellt:

- Erfassung von Hindernissen im Abstand von 30 cm bis zu mind. 2 m (idealerweise bis zu 3 m).
- Absicherung eines Bereichs in der Höhe von 2 m und der Breite 1,5 m vor dem EGD.
- Dieser Bereich soll keine signifikanten Lücken haben, sodass typische Hindernisse wie Pfähle, Zäune, vorstehende Objekte nicht übersehen werden.
- Die Auflösung soll $< 20°$ in Azimut und Elevation betragen.
- Der Sensor soll auf einer Fläche von max. $200\,\mathrm{cm}^2$ untergebracht werden können.
- Es dürfen mehrere Einzelsensoren zu einem Array kombiniert werden.
- Die Tiefenbildszene, die sich aus den Einzelsensorsignalen ergibt, soll mit einem Mikrocontroller auswertbar sein.
- Der Sensor soll für den Innen- und Außenbereich, auch bei Sonnenlicht, geeignet sein.
- Der Sensor soll mit Hilfe eines Adapters, der im 3D-Druckverfahren entwickelt werden kann, auf dem EGD fest verbunden werden.
- Es sollen die Abstandsinformationen mit einem Neigungssensor kombiniert werden bzw. kombinierbar sein, um Einflüsse bspw. des Bodens zu eliminieren.
- Die Auswertung des Sensors soll „embedded“ erfolgen oder zumindest möglich sein, d.h. nicht auf einem PC sondern einem Mikrocontroller.

- Optional: es soll eine Strategie gefunden werden, wie die Tiefenbildszene interpretiert werden kann und Ausweichstrategien, die an die Steuerung des Motors gegeben werden, entwickelt werden.
- Kosten: das Evaluationssystem soll unter 1000 Euro bleiben.
- Es soll nach Möglichkeit ein Evaluationssystem zum schnellen Test (kurze Lieferzeit) zur Verfügung stehen.
- Es soll ausreichend Unterstützung geben (z.B. Schaltpläne und Gerberfiles bzw. Layoutempfehlungen) um in einem späteren Schritt eine maßgeschneiderte Lösung entwickeln zu können.
- Evaluation:
  - Wie groß müssen Objekte mindestens sein, damit sie erkannt werden können?
  - Welchen Einfluss hat das Material, die Entfernung, die Form und Orientierung des Objektes?
  - Welchen Einfluss haben Umgebungsbedingungen wie das Sonnenlicht?
  - Welchen Einfluss haben Wände und Fußboden auf das Ergebnis?
  - Optional: lassen sich Gefährdungen, wie niedergehende Treppen oder Bordsteine erkennen?

Zunächst sollte bei der Sensorauswahl (siehe Kapitel 4.1) der Erfassungsbereich, sowie Schnittstellen zur Auswertung für die infrage kommenden Sensoren in Erfahrung gebracht werden. Diese sind oft direkt auf der Internetseite eines Herstellers oder Lieferanten aufgeführt.

Auch der Preis, die Verfügbarkeit und die Größe sind auf solch einer Internetseite in der Regel direkt ersichtlich.

Wird ein ausreichender Erfassungsbereich sowie eine Schnittstelle, welche leicht mit einem Mikrocontroller angesprochen werden kann (UART, I2C, etc.) angegeben und ist der Sensor preislich im Budget, lieferbar und kompakt genug, sollte im nächsten Schritt das Datenblatt betrachtet werden.

Hierbei sollten detailliertere Infos zu Auflösung, Genauigkeit, Empfindlichkeit gegenüber Sonnenlicht und weitere potenzielle Messprobleme in Erfahrung gebracht und mit den oben aufgeführten Anforderungen abgeglichen werden.

4 Konzept

4.1 Mögliche technische Lösungen

Um die aufgeführten Anforderungen zu erfüllen, wurden verschiedene Sensoren in Erwägung gezogen.

Es waren sowohl radarbasierte als auch andere optische Verfahren im Gespräch. Einzelne optische Sensoren hätten eigenhändig zu einem größeren Sensor-Array kombiniert werden können, was kostengünstig gewesen wäre und eine maßgeschneiderte Anordnung ermöglicht hätte. Damit hätte allerdings der reine Bau des Sensorarrays lange Zeit in Anspruch genommen und die Möglichkeiten zur Auswertung und Verarbeitung der Sensordaten eingeschränkt.

Die Lösung einen fertigen Sensor zu verwenden, diesen ausgiebig zu testen und Software zur Auswertung und Verarbeitung der Daten zu implementieren, schien für die gewünschten Anforderungen zielführender. Eine ToF-Kamera bietet im Gegensatz zu diversen anderen Sensorsystemen den Vorteil, ein detailliertes Bild der Umgebung zu liefern, sodass in Zukunft Verfahren wie Objektklassifizierung möglich wären. Von besonderer Bedeutung bei der Wahl des Sensors war auch die zu erwartende Lieferzeit, da ein enger Zeitplan vorgesehen war.

Die untersuchten Radarsensoren, hatten zum Teil große Erfassungsbereiche. Allerdings hatten diese den Nachteil, dass kein so detailliertes Bild der Umgebung erzeugt werden kann, wie es bei einer ToF-Kamera möglich ist. Daher wurde die Entscheidung getroffen, eine ToF-Kamera auszuwählen.

In die engere Wahl kam die ToF-Kamera mit der Bezeichnung AD-96TOF1-EBZ der Firma Linear Technology/Analog Devices [1]. Diese verfügt über eine Auflösung von 640 x 480 Pixel, ein Field-of-View (FoV) von 90° x 69,2°, eine Genauigkeit $< 2\%$ und diverse Reichweite-Modi, mit denen die geforderte Reichweite von bis zu 2 bzw. 3 m auf jeden

Fall erreichbar wäre [2] . Des Weiteren weist der Hersteller auf ein Software-Development-Kit (SDK) mit unter anderem Matlab- und Python-Wrappern und die Kompatibilität mit einem Raspberry Pi hin.

Der Einsatz dieser Kamera scheiterte letztendlich an der zu erwartenden Lieferzeit.

Die letztendlich verwendete TOF>cam 635 der Firma espros photonics corproration [12] verfügt laut Datenblatt über ein FoV von 50° x 19° und einen Erfassungsbereich von 0,1 m bis 7,5 m im Wide-Field-of-View (WFOV)-Betrieb, sowie ein FoV von 5° x 5° und einen Erfassungsbereich von 1 m bis 15 m im Narrow-Field-of-View (NFOV)-Betrieb [14]. Die Kamera ist für Umgebungs- und Sonnenlicht bis 100 kLux geeignet und kann mit Hilfe der seriellen Schnittstelle UART, bei einer Bitrate von 10 Mbit/s, ausgelesen werden.

Es kann bereits an dieser Stelle festgestellt werden, dass in der Anwendung als Hilfsmittel für blinde und sehbehinderte Menschen mehrere solcher Kameras zum Einsatz kommen sollten, um einen größeren Sichtbereich abzudecken.

Mit einer kurzen Lieferzeit schien dieses Modell als die beste Option für die Erfüllung der gegebenen Anforderungen.

4.2 Verfügbare Mittel und Werkzeuge

Zur Beschaffung von Sensoren für den EGD stehen finanzielle Mittel, in dem in den Anforderungen beschriebenen Rahmen, zur Verfügung, sodass die in dieser Arbeit ausgewählte ToF-Kamera erworben werden konnte.

Außerdem stehen Werkzeuge wie Zangen und Schraubendreher sowie Oszilloskope und diverse Kabel und elektronische Bauelemente zur Verfügung. Im Makerspace der HAW können außerdem Lötstationen benutzt werden. Es bestehen des Weiteren mehrere Möglichkeiten mechanische Bauteile für den EGD mit einem 3D-Drucker zu fertigen.

4.3 Verwendete Hardware

4.3.1 TOF>cam 635

Die in dieser Arbeit verwendeten ToF-Kamera basiert auf einem integrierten Schaltkreis mit der Bezeichnung epc635 3D-TOF-imager der espros photonics corporation [14]. Bei diesem Chip wird eine Form des cwTOF mit einem Rechtecksignal und einem Tastgrad von 50 % und als Empfänger ein CCD-Pixel-Array bestehend aus 160 x 60 Pixel verwendet [13].

Abb. 4.1 zeigt die TOF>cam 635, sowie die FoVs und Reichweiten für die jeweiligen Betriebsmodi.

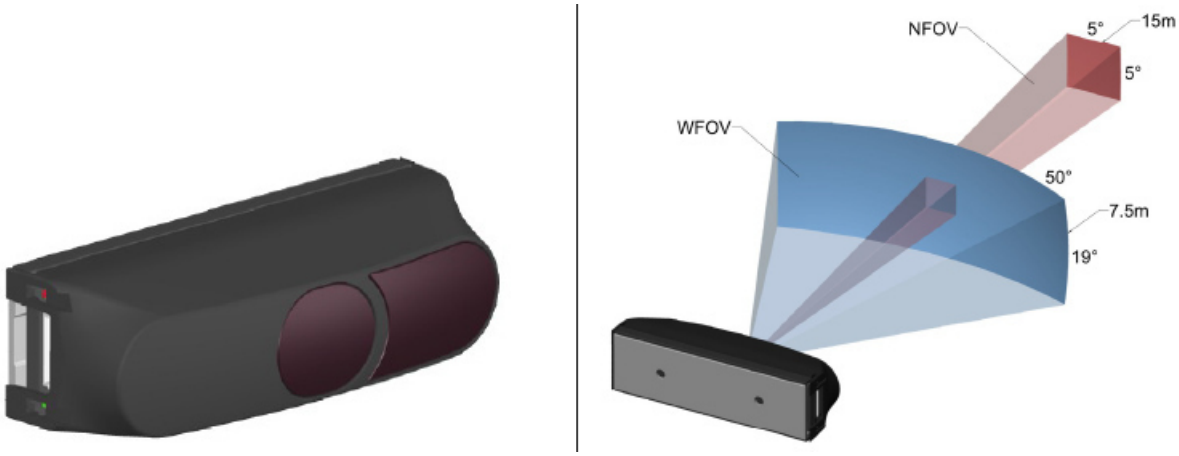

Abbildung 4.1: TOF>cam 635 Field-of-Views [14]

Die höhere Reichweite des NFOV von bis zu 15 m ist für die Erfüllung der Anforderungen nicht erforderlich, während das dadurch verringerte FoV sowie die mangelnde Verwendbarkeit im Bereich von 0 bis 1 m erhebliche Nachteile darstellen. Folglich ist für die Anwendung in dieser Arbeit nur der WFOV-Betrieb von Interesse.

Es existiert ein USB-Adapter mit dem die Kommunikation zwischen PC und ToF-Kamera ermöglicht wird. Die Firma espros photonics corporation stellt eine GUI basierend auf der Bibliothek Qt [27] zur Verfügung [11]. So kann die Kamera leicht für erste Messungen in Betrieb genommen und getestet werden. In der späteren Anwendung muss die Kamera allerdings direkt mit dem Mikrocontroller kommunizieren, sodass einer der UARTs passend zum Kommunikationsprotokoll der Kamera konfiguriert werden muss.

Mögliche Probleme bei der Aufnahme von reflektierenden Oberflächen, besonders auf kurze Distanzen, sind unter anderem die Pixelsättigung (pixel saturation) und der ADC-Überlauf (ADC overflow) [14]. Diese werden im Datenblatt der TOF>cam 635 erwähnt, allerdings nicht konkret erläutert, sind aber wohl auf eine zu große eingefallene Lichtmenge zurückzuführen (Überbelichtung). Kamerapixel, bei denen eins dieser Phänomene auftritt, können nicht zur Entfernungsbestimmung verwendet werden.

Ein besonders wichtiger Parameter für die Verwendung einer TOF>cam 635 ist die sogenannten integration time [14]. Die integration time ist vergleichbar mit der Belichtungszeit einer 2D-Kamera. Bei einer höheren integration time können auch weiter entfernter Objekte erfasst werden. Außerdem ist bei einer dunklen Umgebung eine höhere integration time erforderlich. Auf der anderen Seite wird bei Messungen mit höherer integration time der Einfluss des Umgebungslicht erhöht. Vor allem tritt bei hoher integration time öfter die Pixelsättigung auf, was eine Entfernungsmessung verhindert.

4.3.2 TM4C1294NCPDT Mikrocontroller

Zur Auswertung der Kameradaten wird der Mikrocontroller TM4C1294NCPDT im EK-TM4C1294XL Evaluation Kit der Firma Texas Instruments verwendet [25]. Dieser verfügt über eine 120-Mhz CPU und acht UARTs, welche im High-Speed-Betrieb auf Bitraten bis zu 15 Mbit/s konfiguriert werden können [21].

4.3.3 Kommunikationsschnittstelle

Tabelle 4.1 zeigt die Pinbelegung der ToF-Kamera. Verwendet wird ein Kabel mit Stecker vom Typ JST SM10B-SRSS-TB [14].

Die Anschlüsse IN, OUT1 und OUT2 werden vorerst nicht benötigt.

No.	Name	Function	Comments
1	VDD	VDD: +5V	Stable and free of noise power supply for the imager section. Decouple from pin 8/10.
2	GND	Negative supply terminal	Short with pin 9.
3	PIN3	OUT1	Open-drain output, refer to Chapter 11.1, SET_OUTPUT [0x51] and Figure 5.
4	PIN4	OUT2	
5	PIN5	IN	Digital input, refer to Chapter 11.2, GET_INPUT [0x52] .
6	UART_TX	Data output Tx	Data interface, refer to Chapter 7.
7	UART_RX	Data input Rx	
8	VDDLED	VDD_{LED}: +5V	Supply pin for illumination circuitry. Short with pin 10.
9	GND	Negative supply terminal	Short with pin 2.
10	VDDLED	VDD_{LED}: +5V	Supply pin for illumination circuitry. Short with pin 8.

Tabelle 4.1: Pinbelegung der ToF-Kamera [14]

Die Kommunikation mit der ToF-Kamera funktioniert über eine UART-Schnittstelle [14].

Universal Asynchronous Receiver Transmitter (UART) ist eine serielle Schnittstelle welche Daten in einem festgelegten Rahmen übertägt. Der Datenrahmen besteht aus einem Startbit, fünf bis neun Datenbits, einem optionalen Parity-Bit zur Erkennung von Übertragungsfehlern, sowie einem bis zwei Stoppbits [34].

UART-Konfiguration und UART-Rahmenformat der ToF-Kamera werden in Tabelle 4.2 und Abb. 4.2 dargestellt.

Parameter	Value	Unit	Comment
Baud rate	10	Mbit/s	1 bit = 0.1 µs
Start bits	1	Bit	low active
Data	8	Bit	
Stop bits	1	Bit	high active
Parity	No		
Voltage level LVTTL	3.3	V	

Tabelle 4.2: UART-Kofiguration der ToF-Kamera [14]

Abbildung 4.2: UART-Rahmenformat der ToF-Kamera [14]

Die Kommunikation findet nach dem Master-Slave-Prinzip statt [14]. Die ToF-Kamera fungiert hier als der Slave, der einen Befehl vom Master (in der zu entwickelnden Anwendung der Mikrocontroller) entgegennimmt und daraufhin eine Antwort sendet.

Abb.4.3 visualisiert eine Kommando- und Antwortsequenz.

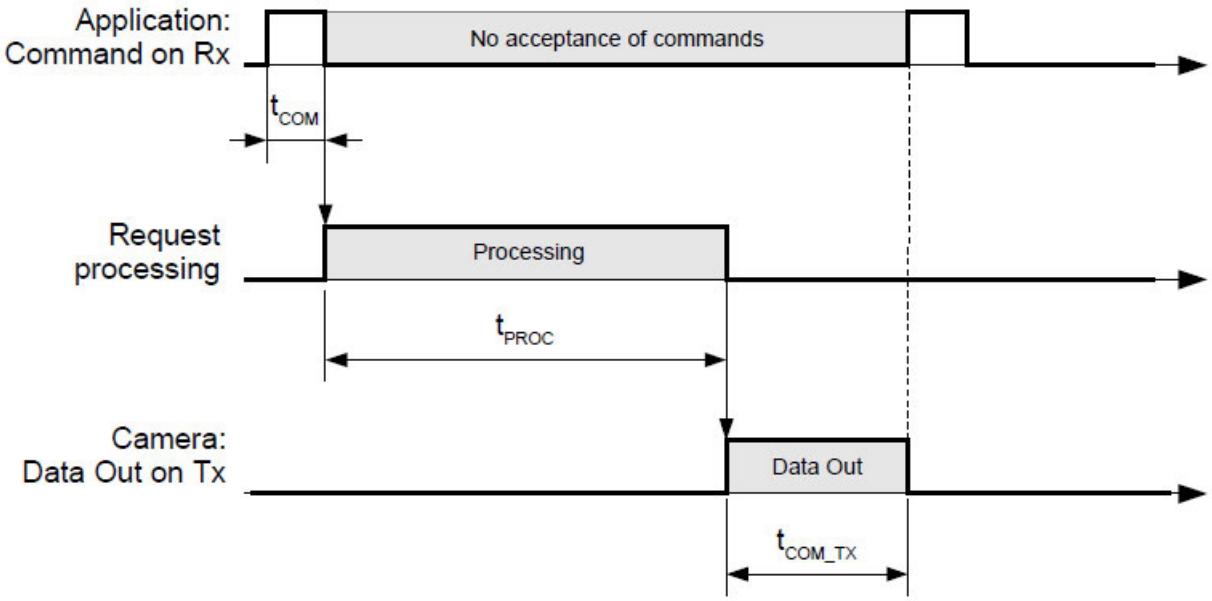

Abbildung 4.3: Kommando- und Antwortsequenz der ToF-Kamera [14]

Zur kontinuierliche Übertragung von Daten kann der Streaming-Mode verwendet werden [14]. Der zugehörige Ablauf wird in Abb 4.4 dargestellt.

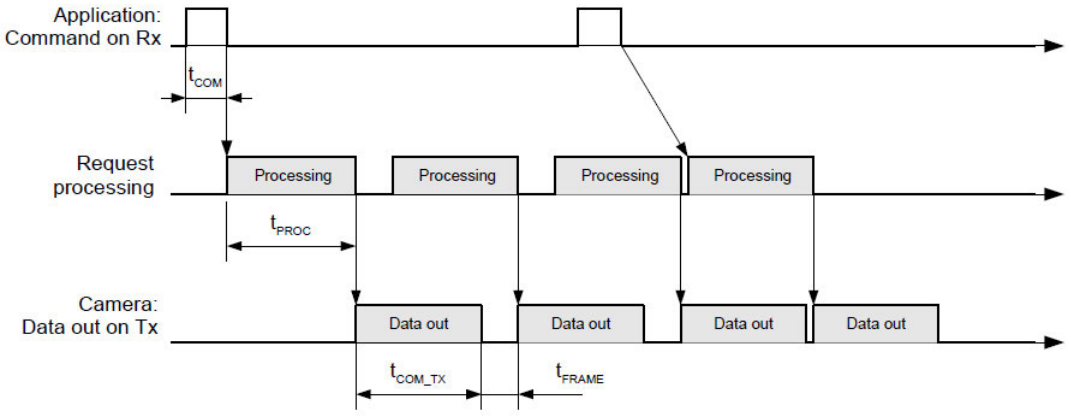

Abbildung 4.4: Streaming Mode [14]

Ein Kommando an die ToF-Kamera wird im Format des in Abb. 4.5 dargestellten Datenpakets übermittelt.

Abbildung 4.5: Kommandoformat der ToF-Kamera [14]

Um Übertragungsfehler erkennen zu können, kommt eine zyklische Redundanzprüfung, auch cyclic redundancy check (CRC) genannt, zum Einsatz [14]. In die Berechnung der CRC-checksumme gehen alle Bytes des Datenpakets bis auf die CRC-Bytes selbst ein. In [14] wird Beispielcode zur CRC-Berechnung bereitgestellt.

Die ToF-Kamera antwortet auf ein Kommando im Format des in Abb. 4.6 dargestellten Datenpakets.

Abbildung 4.6: Antwortformat der ToF-Kamera [14]

Die meisten verwendbaren Kommandos unterteilen sich in GET-commands und SET-commands [14]. Darüber hinaus stehen einige weitere speziellere Kommandos zur Verfügung. Messdaten werden in erster Linie mit Hilfe des Kommandos GET_DIST angefordert. Mit diesem Kommando können unter anderem Einzelbilder oder, im zuvor erwähnten Streaming-Mode, kontinuierlich Bilder aufgenommen werden.

Die Datenbytes der Antwort bestehen, bei erfolgreicher Kommunikation, aus einem Header aus 80 Byte der diverse Informationen enthält, sowie zwei Byte pro Pixel des aufgenommenen Bildes [14]. Dazu kommen die start-, type-, length- und CRC-Bytes aus dem zuvor dargestellten Antwortformat. Bei Verwendung des kompletten Pixel-Arrays von 160 x 60 Pixel ergeben sich somit

$$1Byte(Start)+2Byte(Length)+80Byte(Header)+2\cdot(160\cdot60)Byte+4Byte(CRC) = 19288Byte$$

für jedes Einzelbild der ToF-Kamera.

Die Distanzmessung ergibt für jedes Pixel einen 16-Bit-Wert, welcher wie erwähnt in zwei einzelnen Byte übertragen wird [14]. Die beiden MSBs geben hierbei Auskunft über die Amplitude des empfangenen Lichts und damit über die Verlässlichkeit der Messung. Die restlichen 14 Bit stellen im Dezimalformat die gemessene Distanz in mm dar. Im WFOV-Betrieb also einen Zahlenwert zwischen 0 und 7500, sofern die Messung ein verwertbares Ergebnis für den betreffenden Pixel ergeben hat. Andernfalls wird ein Wert zwischen 16001 und 16008 übertragen, welcher für ein aufgetretenes Problem in der Messung steht und dessen Bedeutung dem Datenblatt zu entnehmen ist.

4.4 Auslesen der ToF-Kamera mittels Mikrocontroller

Zum Auslesen der ToF-Kamera mit Hilfe des Mikrocontrollers muss eine UART-Verbindung zwischen Mikrocontroller und ToF-Kamera aufgebaut werden [14]. Mit den Informationen aus dem Datenblatt kann die Schnittstelle der ToF-Kamera angesprochen, Kommandos versendet und Daten empfangen werden.

Das zu entwickelnde Mikrocontroller-Programm soll also in der Lage sein, Kommandos zum Einstellen der benötigten Parameter sowie Kommandos zum Aufnehmen von Kamerabildern zu versenden und die Antwort der ToF-Kamera korrekt auszulesen.

Die Aufnahme von Kamerabildern, vorerst ohne Interpretation der Ergebnisse, wird grob in Abb. 4.7 dargestellt. Für Details zur Realisierung siehe Abschnitt 5.

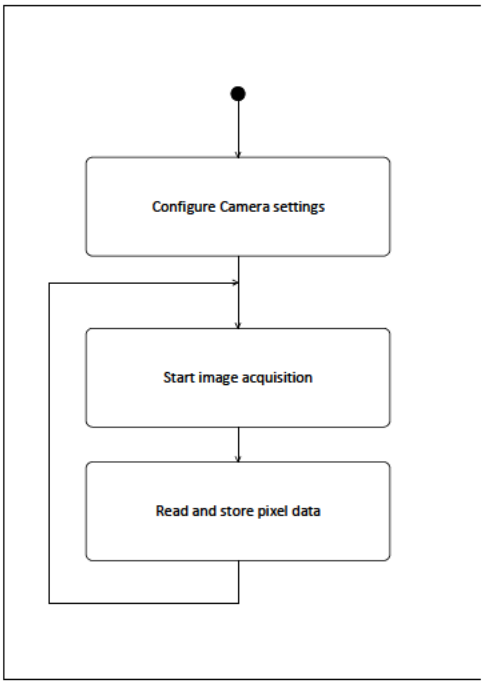

Abbildung 4.7: Allgemeiner Ablauf der Kommunikation zwischen ToF-Kamera und Mikrocontroller

4.5 Navigationsalgorithmen

Um dem Anwender des EGD einen Weg vorzugeben, können Navigationsalgorithmen, wie sie bei mobilen Robotersystemen zum Einsatz kommen, verwendet werden.

Navigationsalgorithmen für autonome Roboter lassen sich unterteilen in solche mit Verwendung einer Karte und solche ohne Karte [6].

Für die Anwendung als Hilfsmittel für blinde und sehbehinderte Menschen könnten beide Arten von Navigationsalgorithmen von Interesse sein. Sofern das Gesamtsystem nur über Sensoren zur Erfassung des Nahbereichs, wie die verwendete ToF-Kamera verfügt, scheint der Anwendungsfall ohne Karte zunächst naheliegender. Ein Staubsaugerroboter, der auf begrenzten Raum agiert, könnte diesen im ersten Schritt komplett abfahren und so eine Umgebungskarte erstellen. Der EGD allerdings könnte sich potentiell überall auf

der Welt bewegen und von vornherein auf eine detaillierte Karte des Terrains, inklusive dynamischer Hindernisse zurückgreifen zu können, scheint kaum realisierbar. Dennoch gibt es Lösungsansätze für solch einen Fall, in welchem Umgebungskarten zum Einsatz kommen sollen.

4.5.1 Kartenbasierte Navigation

Ein erster Ansatz zur Verwendung einer kartenbasierten Navigation soll anhand von Abb. 4.8 und Abb. 4.9 erläutert werden.

Abb. 4.8 zeigt ein Beispiel eines Wegplanungsproblems. Die Szene wird hierbei von oben betrachtet. Die blauen Objekte, jeweils durch einen Buchstaben gekennzeichnet, stellen hierbei Hindernisse dar. Aufgabe des EGD ist es nun einen Weg vom Start zum Ziel zu finden.

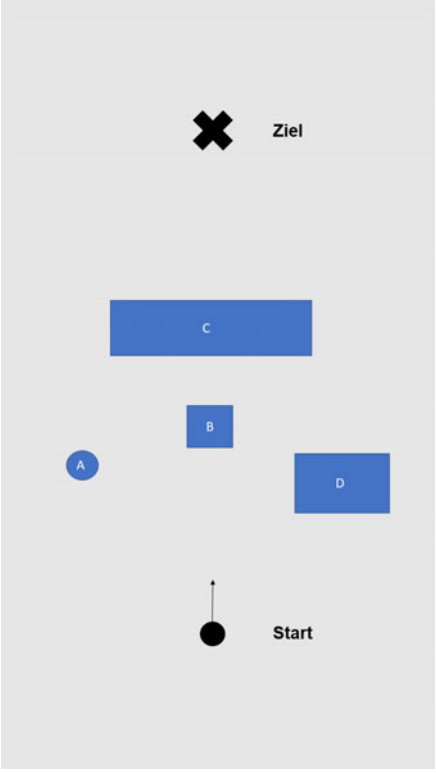

Abbildung 4.8: Beispiel eines Wegplanungsproblems mit Hindernissen

Es wird hierfür angenommen, dass mit Hilfe der ToF-Kamera alle vorhandenen Hindernisse und deren Entfernungen vom EGD korrekt bestimmt werden können. Durch Verwendung der ToF-Kamera alleine besteht allerdings im ersten Schritt keine Möglichkeit zu erfassen, was sich direkt hinter einem Hindernis befindet. Ein erkanntes Hindernis könnte also zunächst theoretisch unendlich tief sein. Befindet sich der EGD noch ein Stück weit von der Szene entfernt, wäre zunächst unklar, ob überhaupt Zwischenräume zwischen den Hindernisse existieren, oder ob es sich um ein großes Hindernis handelt, welches komplett von außen umgangen werden muss.

Durch den strahlenförmigen Erfassungsbereich, könnte bei einer kürzeren Distanz möglicherweise bereits erkannt werden, dass sich zwischen Hindernis A und Hindernis C eine Lücke befindet, durch die hindurch navigiert werden könnte.

Abb. 4.9 soll näherungsweise skizzieren, wie der EGD die Szene mit Hilfe der ToF-Kamera interpretieren könnte. Dabei stellen jeweils blaue und rote Bereiche zusammen die mögliche Form des Hindernisses dar.

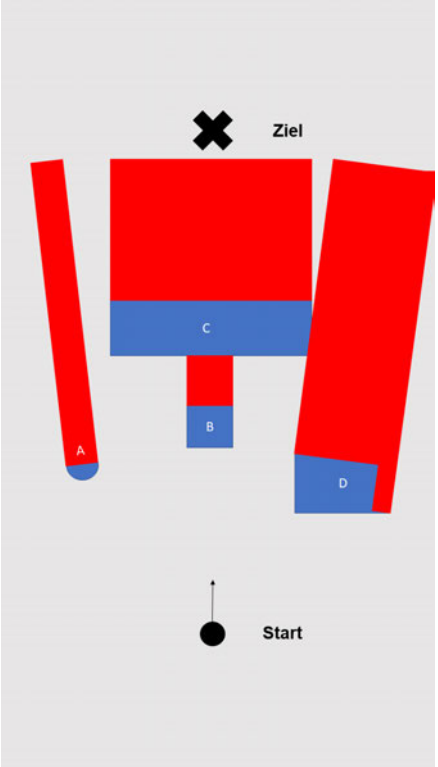

Abbildung 4.9: Hindernisse aus Sicht des EGD

Wird auf die Lücke zwischen Hindernis A und Hindernis C zugesteuert, würde aus der sich ändernden Perspektive vermutlich schnell klar werden, wie Hindernis A tatsächlich geformt ist.

Nach dieser Idee könnte also die betrachtete Szene als eine Art Teilkarte der Umgebung interpretiert werden, auf die ein kartenbasierter Navigationsalgorithmus, eventuell in angepasster Form, angewandt wird und welche dynamisch aktualisiert wird.

Die zeitgleiche Kartierung und Lokalisierung durch Roboter stellt unter der Bezeichnung simultaneous localization and mapping (SLAM) ein großes Forschungsgebiet der Robotik, mit einer Vielzahl von Lösungsansätzen, dar [17].

Eine tiefer gehende Befassung mit SLAM schien für diese Arbeit zu umfangreich. Es wird empfohlen in zukünftigen Arbeiten Literatur zu diesem Thema zu Rate zu ziehen.

Pfadplanung

Steht eine Umgebungskarte zur Verfügung, kann in dieser Karte ein Graph erzeugt werden [17], in welchem durch Anwendung von üblichen Graphsuch-Algorithmen wie dem Dijkstra-Algorithmus [32] ein Pfad zwischen Start- und Zielknoten berechnet wird. Zwei solcher Verfahren zur Erzeugung eines Graphen in einer vorhandenen Karte werden nachfolgend erläutert.

Sichtbarkeitsgraphen

Bei Erzeugung eines Sichtbarkeitsgraphen werden alle vorhandenen Objekte als Polygone interpretiert [6]. Dabei wird jede Ecken eines Polygons mit dem Start- und dem Zielknoten sowie mit allen Ecken aller anderen Polygone verbunden, sofern dadurch kein Polygon geschnitten wird. Dies wird in Abb. 4.10 visualisiert, wobei die dunkelgrau dargestellten Polygone durch einen hellgraue dargestellten Rand erweitert wurden, um die Ausdehnung des Roboters zu berücksichtigen, da dieser andernfalls kollidieren würde. Die Breite des Rands muss mindestens dem halben Durchmesser des Roboters entsprechen.

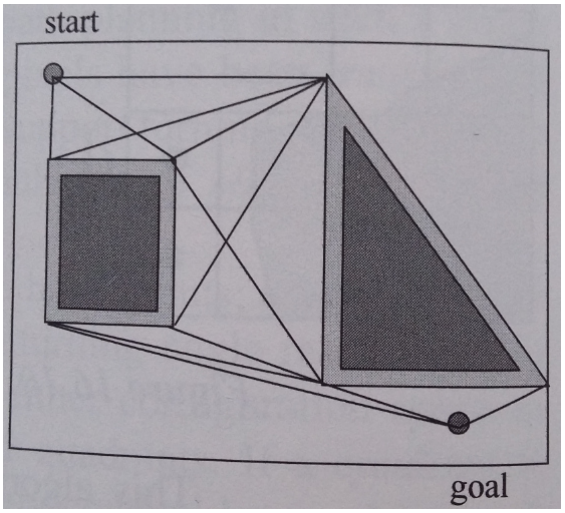

Abbildung 4.10: Beispiel eines Sichtbarkeitsgraphen [6]

Voronoi-Diagramme

Mit Hilfe eines Voronoi-Diagramms kann ein Weg gefunden werden, welcher möglichst kurz ist, dabei aber gleichzeitig maximal weit entfernt von allen Hindernissen entlangführt [17]. Hierfür werden alle Punkte im befahrbaren Raum, welche den maximal möglichen Abstand zwischen allen umliegenden Hindernissen aufweisen, durch Linien verbunden. Daraufhin werden Knoten an den Kreuzungspunkten, den Endpunkten und an willkürlich gewählten Punkten direkt auf den Linien platziert. Werden dann der Start- und der Zielknoten jeweils mit einer kürzestmöglichen Linie mit den Linien des Voronoi-Diagramms verbunden, steht nun ein Graph zur Verfügung, in dem der kürzesten Weg gesucht werden kann.

In Abb. 4.11 wird ein Voronoi-Graph für eine beispielhafte Umgebung dargestellt.

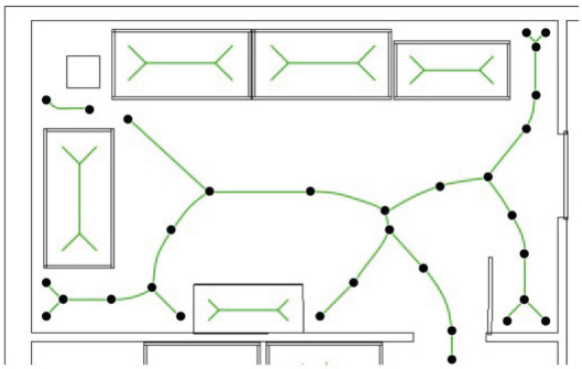

Abbildung 4.11: Beispiel eines Voronoi-Graphen [17]

In einer sehr offenen Umgebung, kann dieses Vorgehen zu Problemen bei der Orientierung führen, da der Weg in so einem Fall sehr weit weg von allen umliegenden Objekten und Wänden führt, sodass die verwendeten unter Umständen Sensoren nichts mehr erfassen können [17]. Um hierfür Abhilfe zu schaffen, kann ein Maximalabstand zu umliegenden Objekten eingeführt werden, der bei der Wegplanung nicht überschritten wird.

4.5.2 Reaktive Navigation ohne Karte

Gegenüber der kartenbasierten Navigation steht die reaktive, auf äußere Reize, also auf Sensorsignale reagierenden Navigation [17]. Nachfolgende werden ausgewählte Algorithmen der reaktiven Navigation erläutert.

Wandering Standpoint Algorithm

Beim Wandering Standpoint Algorithm bewegt das autonome System sich solange geradeaus, bis ein Hindernis im Weg erkannt wird [20]. Tritt dieser Fall ein, wird eine Drehung eingeleitet, um das Hindernis zu umfahren. Das autonome System dreht dabei entweder nach links oder rechts, je nachdem in welcher Richtung der erforderliche Winkel kleiner ist und fährt daraufhin am Hindernis vorbei. Nachteil dieser Vorgehensweise ist, dass Fälle auftreten können, in denen in einer endlosen Sequenz das Ziel niemals erreicht wird. Falls das autonome System sich während der Durchführung einmal komplett um sich selbst gedreht hat ohne dem Ziel näher zu kommen, muss anschließend statt um den

kleineren, um den größere Winkel gedreht werden. Hierfür ist es notwendig eine komplette Umdrehung erkennen zu können und außerdem den aktuellen Abstand zum Ziel zu kennen.

Bug-Algorithmen

Die Bug-Algorithmen sind eine Reihe von Algorithmen, welche in einem autonomen System zur Hindernisvermeidung eingesetzt werden können, sofern dem System bekannt ist in welcher Richtung sein Ziel liegt [17]. Beim Bug1-Algorithmus, dem simpelsten Fall eines Bug-Algorithmus, fährt das autonome System auf sein Ziel zu, bis ein Hindernis erreicht wird, fährt vollständig um das gesamte Hindernis herum und daraufhin zum dem Punkt der abgefahrenen Kontur, welcher die kürzeste Distanz zum Ziel aufweist. Von dort aus wird das Ziel angesteuert. Falls das Erreichen des Ziels möglich ist, wird so immer ein weg gefunden. Allerdings ist dieses Vorgehen offensichtlich sehr ineffizient, da das komplette Hindernis umfahren wird.

Ein Blindenhilfsmittel nach dem Bug1-Algorithmus agieren zu lassen ist für den Anwender wohl kaum zumutbar.

Eine abgewandelte Form stellt der Bug3-Algorithmus dar, bei dem die Umrundung des Hindernisses beendet wird, sobald freie Sicht auf das Ziel erreicht wurde [17]. Hierfür ist es erforderlich, dass die gefahrene Strecke gespeichert wird, da ansonsten eine Endlosschleife erreicht werden kann, falls der bereits befahrene Weg geschnitten wird. In [6] wird angemerkt, dass dieser Algorithmus aufgrund von üblicherweise auftretenden Ungenauigkeiten in der Positions- und Distanzbestimmung meist nicht realisierbar ist.

4.5.3 Fazit zur Navigation

Am erfolgversprechendsten scheint eine umfangreichere Beschäftigung mit dem Forschungsgebiet des SLAM zu sein, um eine Umgebungskarte verwenden zu können, in welcher mit einem erzeugten Graphen nach einem Pfad zwischen den Hindernissen gesucht werden kann.

Bei reaktiven Navigationsalgorithmen werden, aufgrund der geringen verfügbaren Informationen, in der Regel keine optimalen Wege gefunden [17].

5 Implementierung

5.1 Programmierung des Mikrocontrollers

Zur Programmierung des TM4C1294NCPDT stehen zwei Programmiermodelle zur Verfügung: das Direct Register Access Model, bei dem direkt in die Register des Mikrocontrollers geschrieben wird und das Software Driver Model, bei dem die bereitgestellte TivaWare-API mit bereits implementierten und vom Anwender nutzbaren Funktionen verwendet wird [23]. Es besteht weiterhin die Möglichkeit diese beiden Modelle zu kombinieren, falls der Entwickler dies als hilfreich oder notwendig erachtet.

Im Rahmen dieser Arbeit wird bevorzugt das Software Driver Model verwendet, da es eine schnelle Implementierung von Lösungen für Standard-Probleme, wie der Konfiguration des Systemtakts, ermöglicht. Es sorgt außerdem für einen bessere Lesbarkeit des Codes und ist weniger anfällig für Flüchtigkeitsfehler, die beim Setzen und Löschen einzelner Bits, nach dem Direct Register Access Model, auftreten können.

Als Entwicklungsumgebung kommt Code Composer Studio (CCS) in der Version: 9.3.0.00012 zum Einsatz [24].

5.1.1 UART-Konfiguration

Um mit der ToF-Kamera zu kommunizieren muss einer der UARTs auf eine Bitrate von 10 Mbit/s eingestellt werden [14]. Hier verwendet wird der UART6 des Mikrocontrollers. Zunächst wird der Systemtakt auf 120 MHz eingestellt:

```
//Configure UART6 for Communication with ToF-Kamera
//set clock frequency 120 MHz
sysClk = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                                 SYSCTL_OSC_MAIN |
                                 SYSCTL_USE_PLL |
                                 SYSCTL_CFG_VCO_480), 120000000);
```

Die Konfiguration des UART6 wird in eine separate Funktion ausgelagert, in welcher die UART-Funktionen der TivaWare-API [23] aufgerufen werden:

```
void configUART6(uint32_t sysClk, uint32_t bitrate, uint32_t dataLength,
    uint32_t stopBits, uint32_t parity){
    //Enable PORTP for UART6
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART6);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOP);

    GPIOPinConfigure(GPIO_PP0_U6RX);
    GPIOPinConfigure(GPIO_PP1_U6TX);
    ROM_GPIOPinTypeUART(GPIO_PORTP_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    //configure UART6
    ROM_UARTConfigSetExpClk(UART6_BASE, sysClk, bitrate,
                            (dataLength | stopBits |
                             parity));
}
```

Verwendet wird die configUART6-Funktion mit den erforderlichen Parametern wie folgt:

```
    //configure UART6 with Bitrate 10 Mbit/s, 8 Data Bits, 1 Stop Bit, Parity
         None
    configUART6(sysClk, 10000000, UART_CONFIG_WLEN_8, UART_CONFIG_STOP_ONE,
        UART_CONFIG_PAR_NONE);
```

Ein ToF-Kamera-Kommando wird in einem Array gespeichert, wobei jedes Element aus einem Byte besteht, da die Kommandos Byte für Byte versendet werden. Hier enthält das Array GET_DIST das gleichnamige Kommando. Die letzte vier Byte werden mit den berechneten CRC-Werten gefüllt:

```
        //GET_DIST single measurement, 0xF5: start, 0x20: GET_DIST, 0x00:
            single measurement
    uint8_t GET_DIST[COMMANDLENGTH] = {0xF5, 0x20 , 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    //calculate CRC-checksum
    checksum = calcCrc32_32(GET_DIST, 10);

    //store checksum-bytes in command-array
    for(j = 0; j < 4; j++){
        GET_DIST[j+10] = (uint8_t)((checksum>>(8*j)) & 0x000000FF);
    }
```

Mit der Funktion UARTsend() wird das Kommando an die ToF-Kamera versendet:

```
    //send command to get image
    UARTsend(UART6_BASE, GET_DIST, 14);
```

Die Funktion UARTCharGet() liest ein Byte aus dem UART-FIFO ein:

```
        receivedByte = UARTCharGet(UART6_BASE);
```

Um ein komplettes Bild der ToF-Kamera zu empfangen wird die Funktion UARTCharGet() in einer for-Schleife 19288 mal hintereinander aufgerufen. Das zugehörige CCS-Projekt trägt die Bezeichnung "TOFcamera". Zum CCS-Projekt sei zu erwähnen, dass die Stackgröße in den "Properties" des Projektes angepasst wurde, um Stacküberläufe zu vermeiden, da das Array mit den Pixelwerten in der Main-Funktion deklariert wurde.

5.1.2 Empfang eines Einzelbildes

Zunächst wurde ein Kommando zum Aufnehmen eines Einzelbildes an die ToF-Kamera gesendet. Die empfangenen Daten wurden mit Hilfe eines 3D-Scatterplots in Matlab visualisiert. Es wurde ein Maximalwert von 3000 festgelegt auf den alle höheren Werte herunter gesetzt wurden, da größere Abstände im Rahmen der Anforderungen nicht von Interesse sind und so außerdem fehlerhafte Pixel (Werte 16001-16008) ignoriert werden.

Bei der integration time muss ein Kompromiss gefunden werden, sodass die Szene möglichst genau erfasst wird, Rauschen und Pixelsättigung aber minimiert werden.

Abb. 5.1 zeigt den Scatterplot der empfangenen Einzelbilddaten aus der Aufnahme einer Kaffetasse im Abstand von ca. 0.5 m bei einer integration time von 35 μs. Die Form einer Kaffeetasse ist deutlich erkennbar, unter ihr ist weiterhin der Tisch auf dem die Tasse stand zu erkennen.

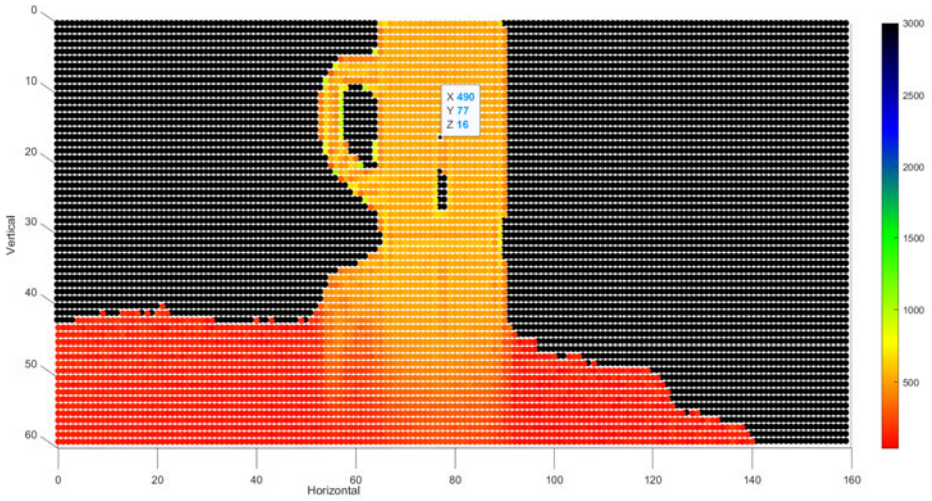

Abbildung 5.1: Kaffeetasse im Abstand von 0.5 m, 35 μs

Die Visualisierung entspricht dem, aufgrund von im Vorfeld durchgeführten Tests mit GUI und USB-Adapter, erwarteten Ergebnis. Aufgrund dieser Aufnahme kann die Implementierung der Kommunikation von Mikrocontroller und ToF-Kamera als erfolgreich betrachtet werden. Der schwarze Bereich auf der Tasse entstand höchstwahrscheinlich aufgrund von ADC-Überläufen und/oder Pixelsättigungen, welche in der hier durchgeführten Auswertung ignoriert wurden.

5.1.3 Livedaten in Matlab via COM-port

Zur Verifizierung der mit dem Mikrocontroller empfangenen Daten, wurde eine serielle Kommunikation über einen COM-Port zwischen PC und Mikrocontroller in Betrieb genommen.

In der Default-Konfiguration der Jumper JP4 und JP5 auf dem Entwicklungsboard ist der UART0 des Mikrocontrollers mit dem virtuellen COM-Port verbunden [22]. Es können also die üblichen Funktionen zur UART-Kommunikation des UART0 verwendet werden um über den COM-Port zu kommunizieren.

Im Matlab-Skript "ToF.m" wird eine Verbindung zum verwendeten COM-Port aufgebaut. Die von der ToF-Kamera empfangenen Daten werden nach Aufnahme eines einzelnen Bildes über den UART0 versendet. Eine Callback-Funktion in "readToFdata.m" wird

aufgerufen, sobald Daten über den COM-Port vefügbar sind. In "readToFdata.m" werden die Daten eingelesen und an die Funktion "plotToFdata.m" übergeben, welche diese zur Visualisierung aufbereitet und mittels 3D-scatterplot darstellt. Verwendet wurde die Matlab-Version R2019b [26].

In dieser Konfiguration wird weiterhin für jedes Bild ein Kommando zur Aufnahme eines Einzelbildes an die ToF-Kamera gesendet, da die Verwendung des Streaming-Modes im Zusammenhang mit der seriellen Kommunikation zum PC offenbar Probleme in der Synchronisation zwischen Mikrocontroller und Matlab-Skript verursacht, wodurch die Messdaten nicht in korrekter Form im Matlab workspace ankommen.

Messungen mit Hilfe der Timer des Mikrocontrollers lassen darauf schließen, dass im Streaming-Mode etwa 44 Bilder pro Sekunde aufgenommen werden können. Bei Messungen mit unterschiedlicher integration time variiert die Bildrate leicht.

Bei kontinuierlicher Aufnahme von Einzelbildern, ergab sich in den Messungen etwa eine Reduzierung der Bildrate auf ca. 20 Bilder pro Sekunde. In der letztendlichen Anwendung, ohne Kommunikation zum PC, ist also der Streaming-Mode aufgrund der höheren Geschwindigkeit zu bevorzugen.

Abb. 5.2 stellt den Ablauf des Mikrocontrollerprogramms zum Auslesen der ToF-Kamera dar.

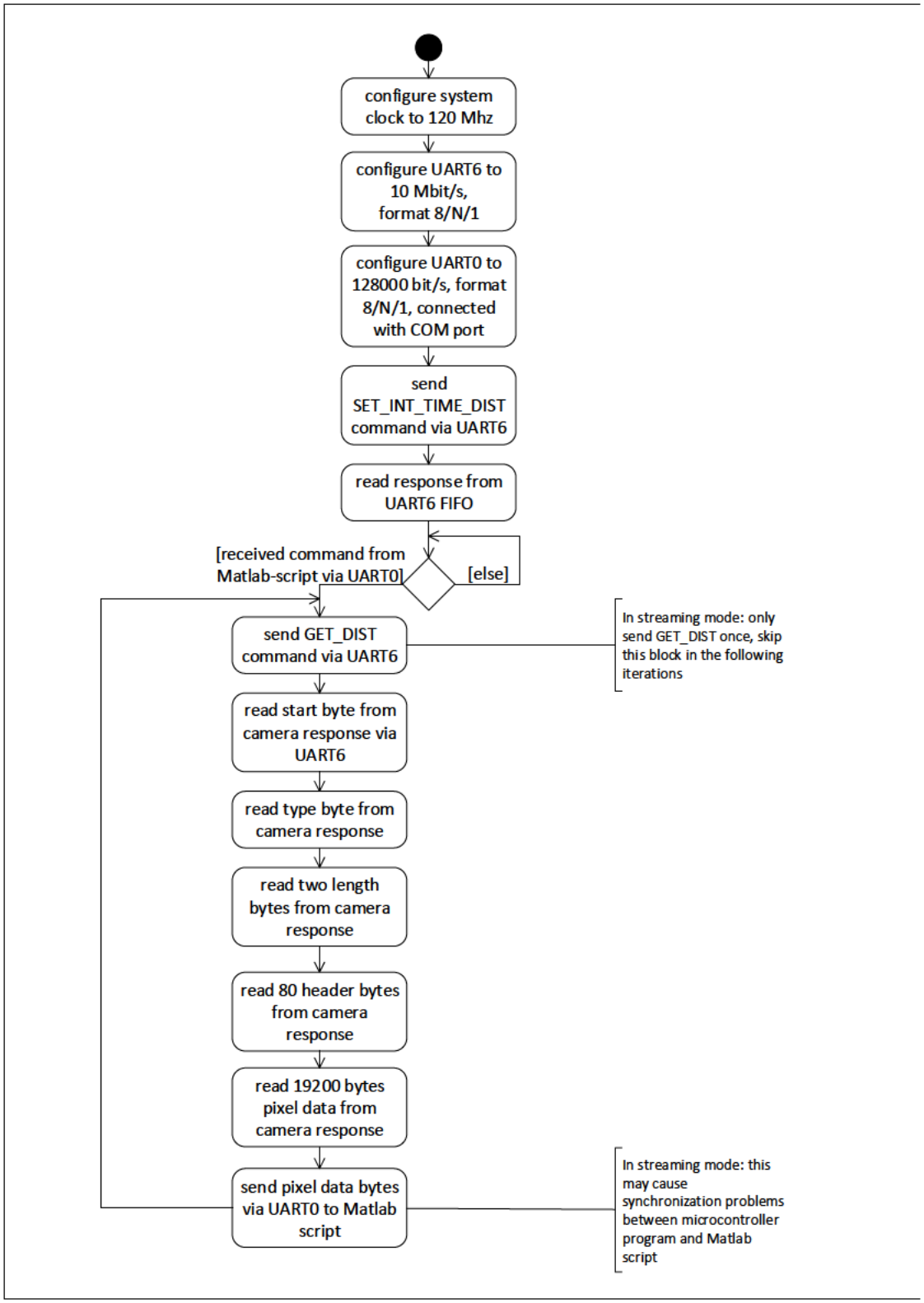

Abbildung 5.2: Ablauf des Mikrocontrollerprogramms

Der generelle Ablauf der Kommunikation mit dem Mikrocontroller sowie der Verarbeitung der Sensordaten in Matlab wird in Abb. 5.3 dargestellt. Wie erwähnt wurden einige der Schritte in Funktionen ausgelagert, verteilt auf zwei zusätzliche Dateien.

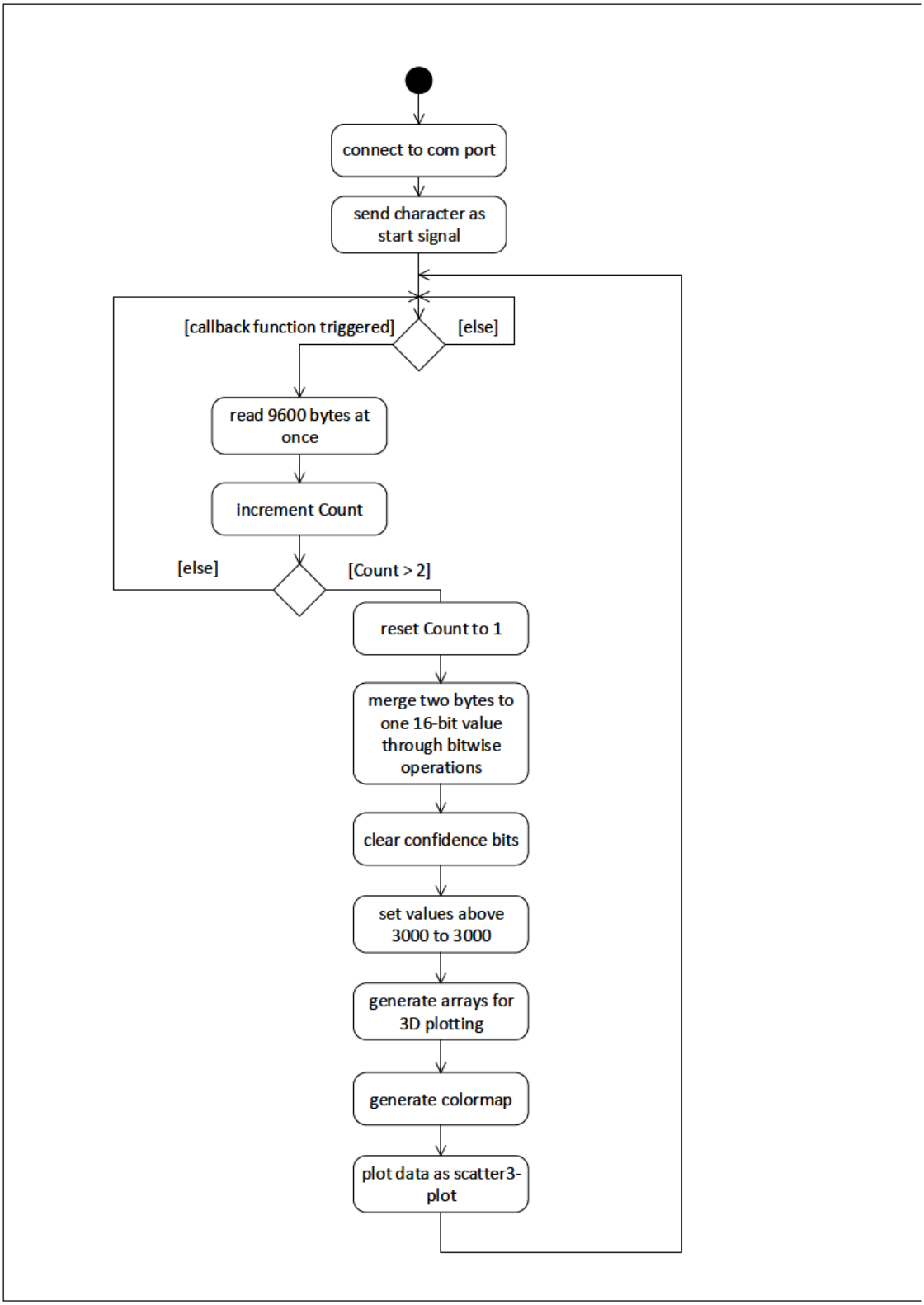

Abbildung 5.3: Ablauf des Matlab-Programms

Des Weiteren werden im CCS-Projekt mit der Bezeichnung "TOFcameraStreaming" im Streaming-Mode zehn Bilder hintereinander aufgenommen und gespeichert. Diese können im Debug-Modus von CCS aus dem Speicher heruntergeladen und mit dem Matlabskript "ToF_movingObject.m" zu einem kurzen Video zusammengefügt werden, um Bewegungen erfassen zu können. Zum Herunterladen aus dem Speicher des Mikrocontrollers, muss ein Breakpoint gesetzt werden, welcher erreicht wird, sobald die zehn Bilder fertig empfangen wurden.

5.2 Mechanische Integration in den Prototyp

Zur mechanischen Integration des Sensorsystems in den bestehenden Prototyp des EGD wurde ein Adapter konstruiert und mittels 3D-Druckverfahren hergestellt.

Es war bereits abzusehen, dass die Kamera nicht den kompletten erforderlichen Erfassungsbereich abdecken kann. Langfristig müssen hierfür mehrere ToF-Kameras zum Einsatz kommen.

Zunächst muss also ein Teilbereich des gewünschten Erfassungsbereichs ausgewählt werden, der dann durch die erste Kamera abgedeckt wird. Möglich wäre eine Positionierung, bei der die Kamera etwa auf Hüfthöhe des Anwenders geradeaus zeigt und so große Hindernisse direkt vor dem Anwender erfasst.

Eine andere Option wäre, zunächst den Bereich in Bodennähe abzudecken, da zu erwarten ist, dass ein großer Teil der potentiellen Hindernisse am Boden steht. Dies könnte mit der Beobachtung des Bodens kombiniert werden, um Treppenabgänge erkennen zu können. Diese Argumente führten zum Entschluss eine Positionierung am EGD vorgesehen, mit der Objekte in Bodennähe sowie der Boden erfasst werden kann.

Experimentell konnte mit einer Kamera eine Konfiguration gefunden werden, in der zumindest ein gewisser Bereich des Bodens vor dem EGD, zeitgleich mit Hindernissen in etwa 1,30 m Abstand vom EGD erfasst werden konnte. Die ToF-Kamera muss hierfür in einem gewissen Winkel zum Boden geneigt werden. Diesen erforderlichen Vertikalwinkel ohne eine vorhandene Halterung am EGD-Prototyp zu bestimmen, stellte sich als schwierig heraus.

Der erforderliche Vertikalwinkel hängt auch von der Höhe in der der Griff des EGD gehalten wird und damit von der Größe, Armlänge sowie der Haltung des Anwenders ab.

Selbst wenn zunächst exemplarisch eine feste Körpergröße, bzw. Armlänge und Körperhaltung angenommen wird und die Kamera nach diesen Vorgaben orientiert werden soll, ist es umständlich eine fixe Konfiguration zu finden, in der sowohl Boden als auch Hindernisse am Boden gut zu erkennen wären. Eine leichte Verkippung hätte hierbei einen zu großen Einfluss. Daher schien es sinnvoll den Adapter so zu konstruieren, dass der Vertikalwinkel einstellbar gehalten wird. Dadurch kann die Einstellung des Vertikalwinkels zunächst von Hand deutlich leichter durchgeführt und angepasst werden, als wenn dies direkt und unveränderbar bei der Konstruktion des Adapters erfolgen würde. Außerdem könnte so im Rahmen zukünftiger Arbeiten eine automatische, dynamische Anpassung mittels Gyro-Sensoren und Motoren erfolgen. Aus diesen Gründen wurde beschlossen den Adapter aus mehreren Einzelteilen zu fertigen, von denen die beiden Hauptkomponenten beweglich und feststellbar miteinander verbunden werden.

Um den Einstellbereich für den Vertikalwinkel nicht unnötig konstruktiv einzuschränken, wurde der Adapter so konstruiert, dass die ToF-Kamera etwas aus dem EGD-Prototyp herausragt. So wird auch gewährleistet, dass nicht ein Teil des EGD-Prototyp im Kamerabild auftaucht und so den Erfassungsbereich verkleinert.

Am Prototypen des EGD waren zwei Löcher mit Innengewinde vorhanden, die bisher nicht benötigt wurden und daher zur Befestigung des Adapters verwendet werden.

Da das FoV der ToF-Kamera sich in der vertikalen Richtung mit 19° als etwas klein erwies, um eine stabile Konfiguration zu finden, in der Boden und der Bereich in Bodennähe zeitgleich erfasst werden, wurde bereits an dieser Stelle ein Platz für eine zweite Kamera vorgesehen, deren FoV sich möglichst nahtlos an das der ersten Kamera anschließt. Der Winkel zwischen den beiden Kameras, welche am Adapter befestigt werden können, wurde so festgelegt, dass die FoVs sich nicht schneiden, aber möglichst nah beieinander liegen. Dieser Winkel wurde experimentell mit Hilfe der CAD-Modelle und deren Darstellung der FoVs bestimmt. Die von der Firma espros photonics corporation zur Verfügung gestellten CAD-Modelle der TOF>cam 635 [10], enthalten wie erwähnt eine Darstellung der FoVs, wie in Abb. 5.5 erkennbar ist. Zur Bestimmung des erforderlichen Winkels zwischen den beiden vorgesehenen ToF-Kameras, wurden die Ränder der dargestellten WFOVs auf eine Länge von 3 m verlängert und experimentell bestimmt, bei welchem Winkel die Ränder der beiden WFOVs nah beieinander liegen ohne sich zu schneiden. Der Platz für die zweite ToF-Kamera ist um 18,5 Grad nach hinten geneigt. Bei Verfügbarkeit einer zweiten Kamera muss diese Einstellung in der Praxis getestet werden.

Der Bereich, in dem der Adapter angeschraubt werden soll, verläuft nach innen abgerundet. Zunächst bot es sich an, Rundungen am Adapter so abzustimmen, dass sie sich exakt in die des EGD-Prototypen einfügen. Aufgrund der Geometrie des EGD-Prototypens war es allerdings zum Teil schwierig, mit einem Messschieber genaue Maße zu nehmen. Besonders die Radien der Rundungen in dem Bereich, in dem der Adapter festgeschraubt werden sollte, waren schwer zu bestimmen. Daher wurde der hintere Teil des Adapters, welcher am EGD-Prototypen verschraubt ist, vergleichsweise dünn konstruiert, da er so leichter einzupassen war und hauptsächlich die Position der beiden Löcher korrekt bestimmt werden musste. Zusätzlich konnte so auch Material eingespart und damit die Druckzeit des Einzelteils reduziert werden. Bei einem Konstruktionsfehler, welcher erst beim festschrauben aufgefallen wäre, wäre der Zeitverlust so geringer.

Eine Krafteinwirkung in vertikaler Richtung am Adapter könnte allerdings im hinteren Teil des Adapters, welcher am EGD-Prototypen verschraubt ist, ein Biegemoment erzeugen, was für die dünnen Konstruktion an dieser Stelle problematisch wäre. Um den montierten Adapter gegen ein mögliches Einwirken von Kräften in vertikaler Richtung zu stabilisieren, wird daher zusätzlich eine Stange mit Gewinde am unteren Ende (halterungsstange.prt), sowie eine Platte (halterungsplatte.prt) verwendet, welche ebenfalls mit einer gedruckten Mutter (mutter.prt) gegeneinander geschraubt werden. Vertikale Kräfte, welche auf den Adapter wirken, werden so zu einem großen Teil auf das Gehäuse des EGD-Prototypen übertragen und entlasten damit die Schraubverbindung. Bei dieser Konstruktion wurde darauf geachtet, dass das mit der ToF-Kamera verschraubte Bauteil in der Baugruppe seine Bewegungsfreiheit behält.

Die vorhandene Konfiguration soll wie erwähnt ermöglichen, dass der Boden sowie Hindernisse in Bodennähe erkannt werden können. Hierdurch sollen unter anderem Treppenstufen detektiert werden. Außerdem kann so bereits eine große Anzahl Hindernisse, nämlich solche die auf dem Boden stehen, erkannt werden. Gegen hervorstehende oder von oben hängende Objekte muss der Anwender mit weiteren Sensoren abgesichert werden.

Es wurden die benötigten Anschlussmaße am Prototypen genommen und passende Bauteile konstruiert. Für den 3D-Druckprozess ist es erforderlich, die erstellten Bauteile als Stereolithografie-Dateien (.stl) zu speichern. Dabei wird das Bauteil in eine Vielzahl von Dreiecken aufgeteilten, deren Anzahl einstellbar ist und die Genauigkeit der zuvor festgelegten Maße beeinflusst. Die .stl-Datei wird anschließend in einer sogenannten Slicer-Software geöffnet, diverse Druckeinstellungen vorgenommen und daraus eine GCODE-

Datei (.gcode) erzeugt, welche die letzendlichen Anweisungen für den 3D-Drucker in einer für Menschen prinzipiell lesbaren Form enthält. Die .gcode-Datei wird vom 3D-Drucker von einer SD-Karte gelesen.

Als CAD-Software kam PTC Creo Parametric 3.0 M100 zum Einsatz [19]. Als Slicer-Software wurde Ultimaker Cura 3.2.1 verwendet [28]. Gedruckt wurden alle im Rahmen dieser Arbeit 3D-gedruckten Objekte und Bauteile auf einem Anycubic I3 Mega, des chinesischen Herstellers Anycubic [3]. Als Material kam jeweils schwarzes PLA, ebenfalls hergestellt von Anycubic, zum Einsatz.

Der mechanische Adapter besteht aus insgesamt sieben Einzelteilen. Das Bauteil in der Datei "tof_adapter.prt" wird direkt am EGD verschraubt. Am Bauteil in der Datei "kamerahalter.prt" wird die ToF-Kamera angeschraubt. Außerdem ist ein Platz für eine zweite Kamera oberhalb der ersten vorgesehen. Diese beiden Bauteile werden mit einer ebenfalls gedruckten Schraube (schraube.prt) sowie einer gedruckten Mutter (mutter.prt) zusammengefügt. Dies gewährleistet, dass der Vertikalwinkel der Kamera angepasst werden kann.

Abb. 5.4 zeigt die Baugruppe des Adapters in Creo Parametric.

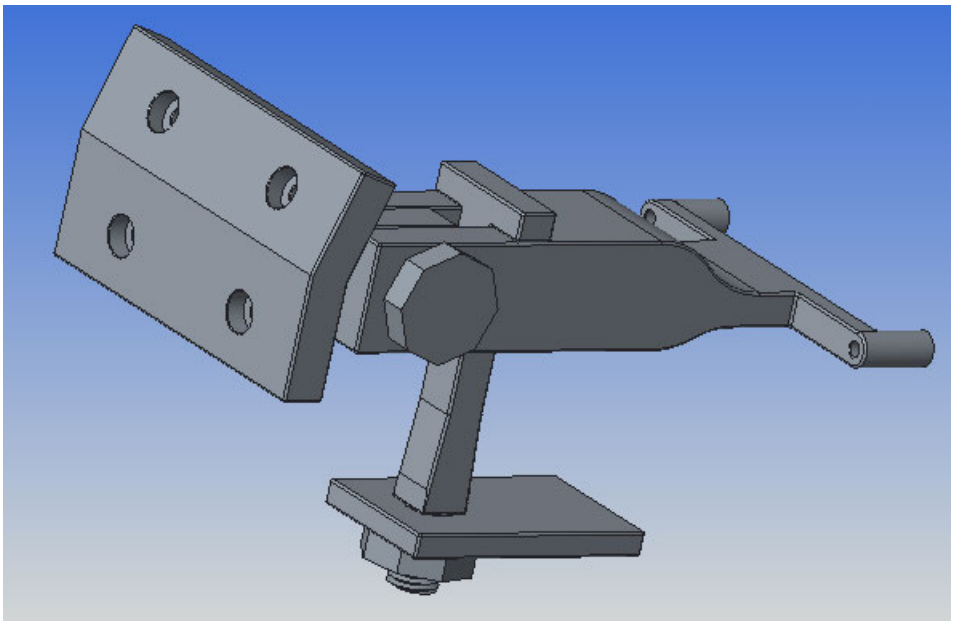

Abbildung 5.4: Baugruppe mechanischer Adapter

In Abb. 5.5 ist die Baugruppe des Adapters mit zwei Modellen der TOF>cam 635 inklusive einer Visualisierung der FoVs in Creo Parametric dargestellt.

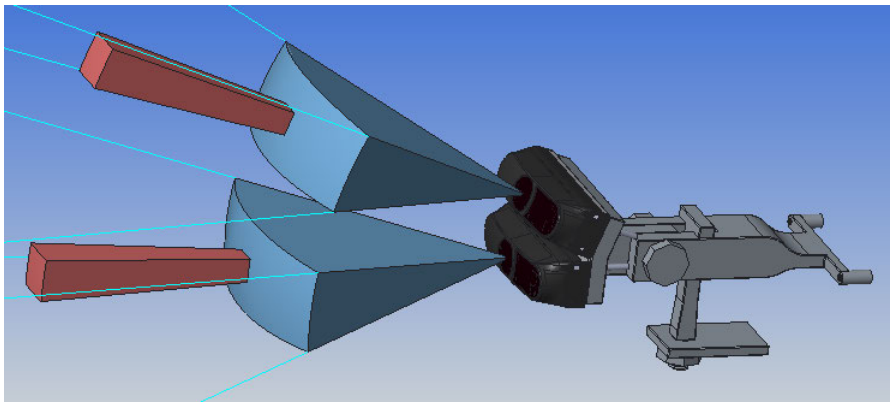

Abbildung 5.5: Baugruppe mechanischer Adapter mit Kameras

Abb. 5.6 zeigt den Prototyp des EGD mit angeschraubter ToF-Kamera.

Abbildung 5.6: EGD mit Kamera

Abb. 5.7 zeigt den Adapter am EGD-Prototyp ohne Kamera und Kamerahalter.

Abbildung 5.7: Mechanischer Adapter ohne Kamera und Kamerahalter

Die Abb. 5.8 und 5.9 zeigen den Adapter mit Kamera am EGD-Prototyp.

Abbildung 5.8: Mechanischer Adapter von vorne

Abbildung 5.9: Mechanischer Adapter von der Seite

6 Evaluation

6.1 Messungen mit USB-Adapter

Zunächst wurde die ToF-Kamera mit Hilfe des USB-Adapters mit Messobjekten unterschiedlicher Farben, Formen, Größen und Orientierungen getestet. Verwendet wurden hierbei vier schwarze, 3D-gedruckte Messobjekte, sowie diverse Platten (50 mm x 50 mm x 2 mm) [16] aus unterschiedlichen Materialien aus dem Objektsortiment der Firma Festo Didactic GmbH & Co. KG [15].

Die vier 3D-gedruckten Messobjekt werden im folgenden als Messobjekt 1-4 bezeichnet:

- Messobjekt 1: quadratischer Querschnitt, Breite 10 mm, Höhe 33 mm
- Messobjekt 2: quadratischer Querschnitt, Breite 5 mm, Höhe 33 mm
- Messobjekt 3: runder Querschnitt, Durchmesser 10 mm, Höhe 33 mm
- Messobjekt 4: runder Querschnitt, Durchmesser 5 mm, Höhe 33 mm

In den jeweiligen Abbildungen ist auf der linken Seite das Bild der Szene dargestellt und auf der rechten Seite das Ergebnis des pixel scopes. Zur Verwendung des pixel scopes werden einzelne Pixel im Bild markiert, zu erkennen an der waagerechten, weißen Linie und deren gemessene Entfernungswerte dargestellt. Da die Kamera bei den Messungen auf einem Tisch lag, ist der farbliche Bereich, welcher sich in etwa über die untere Hälfte des Bildes erstreckt auf den Tisch zurückzuführen.

Die Dokumentation enthält keine Legende der Farbgebung, diese kann mit Hilfe des Source-Codes der GUI interpretiert werden. Eine violette Färbung wie am unteren Rand von Abb. 6.1 deutet auf eine Pixelsättigung hin, während eine Färbung in einem Magenta-Ton einen ADC-Überlauf anzeigt. Diese Pixel können also nicht zur Entfernungsbestimmung betrachtet werden.

Exakte Werte konnten aus dem pixel scope nicht ohne Weiteres ausgelesen werden, sodass diese eher grob aus dem Plot abgelesen werden. Exakte Werte werden in späteren Betrachtungen mit Hilfe des Mikrocontrollers ausgelesen.

In der Bildunterschrift ist jeweils auch die verwendete integration time aufgeführt.

Abb. 6.1 zeigt die Aufnahme von Messobjekt 1 im Abstand von 0,5 m, aufgenommen mit einer integration time von 105 μs.

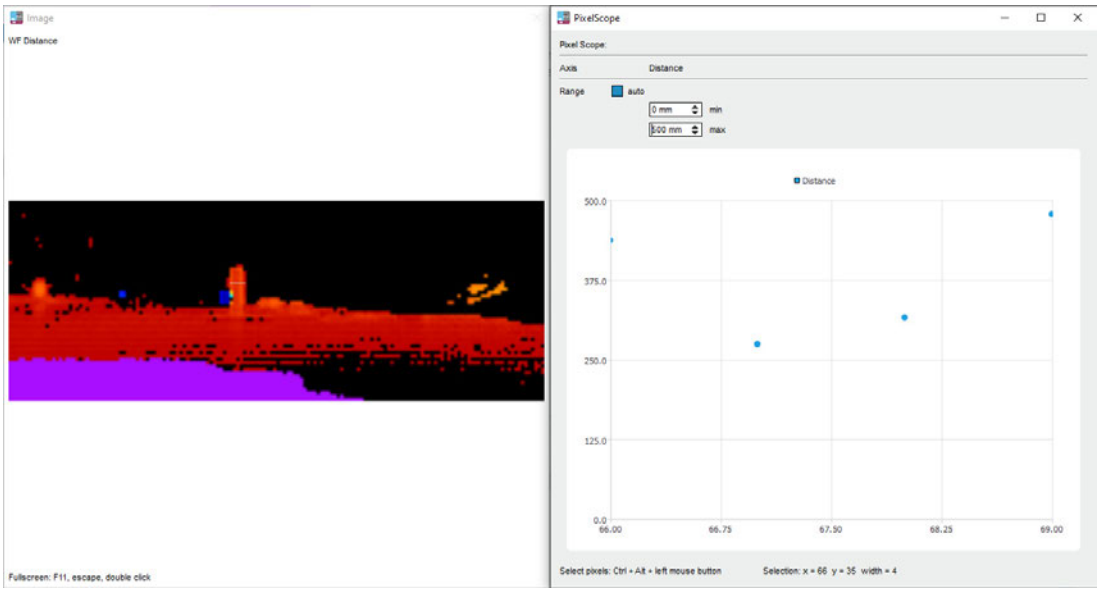

Abbildung 6.1: Messobjekt 1, 0.5 m, 105 μs

Die gemessenen Abstandswerte bewegen sich in etwa zwischen 275 mm und 480 mm.

Abb. 6.2 zeigt die Aufnahme von Messobjekt 2 im Abstand von 0,5 m, aufgenommen mit einer integration time von 105 μs.

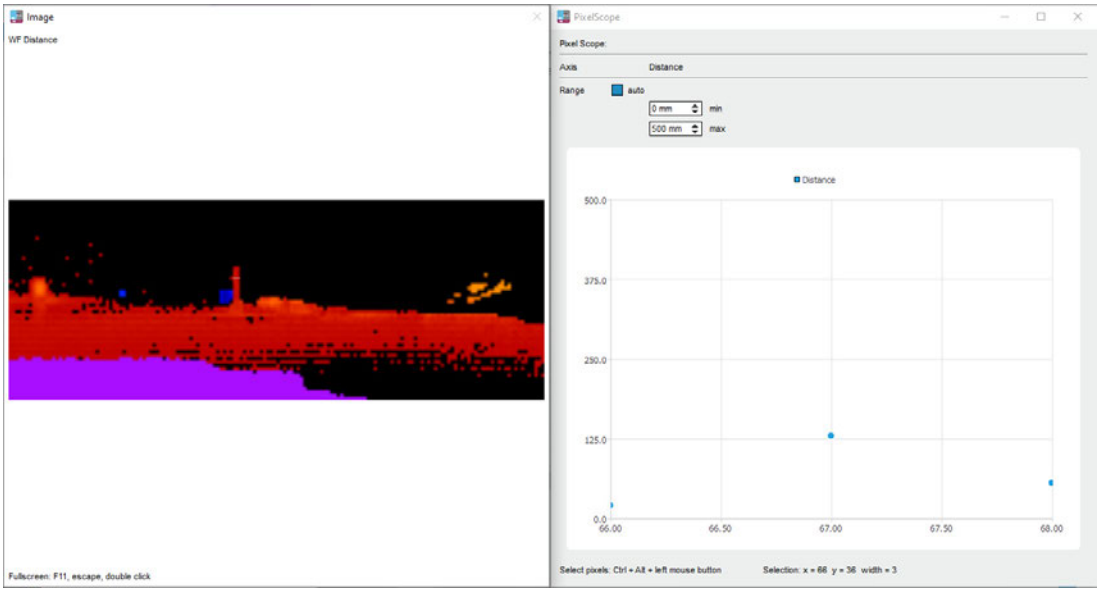

Abbildung 6.2: Messobjekt 2, 0.5 m, 105 μs

Die gemessenen Abstandswerte bewegen sich in etwa zwischen 25 mm und 130 mm.

Abb. 6.3 zeigt die Aufnahme von Messobjekt 3 im Abstand von 0,5 m, aufgenommen mit einer integration time von 105 μs.

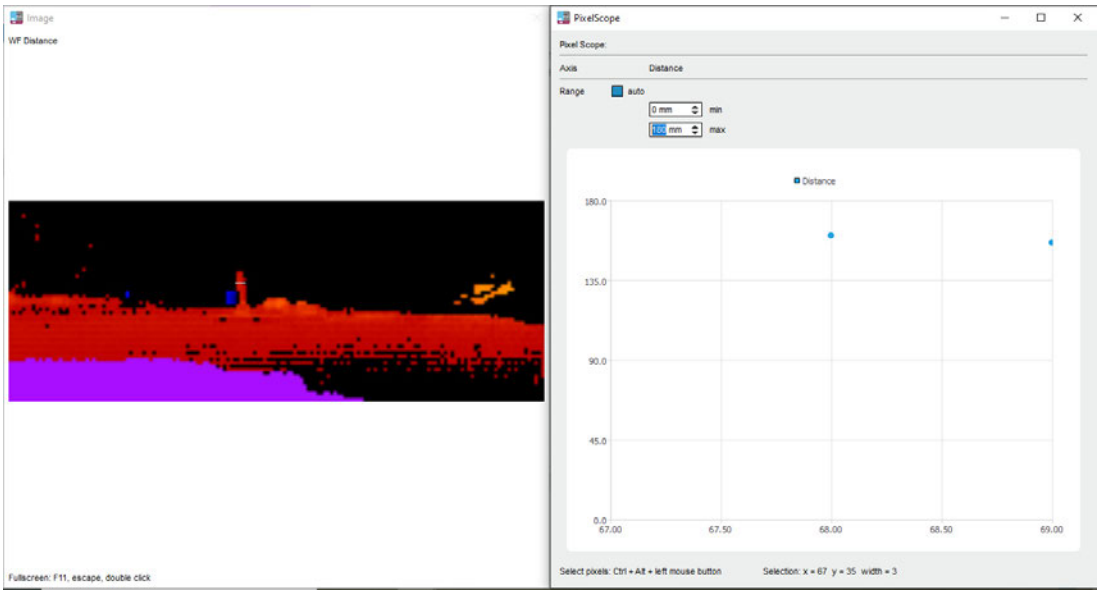

Abbildung 6.3: Messobjekt 3, 0.5 m, 105 μs

Die gemessenen Abstandswerte bewegen sich in etwa im Bereich von 160 mm.

Abb. 6.4 zeigt die Aufnahme von Messobjekt 4 im Abstand von 0,5 m, aufgenommen mit einer integration time von 105 μs.

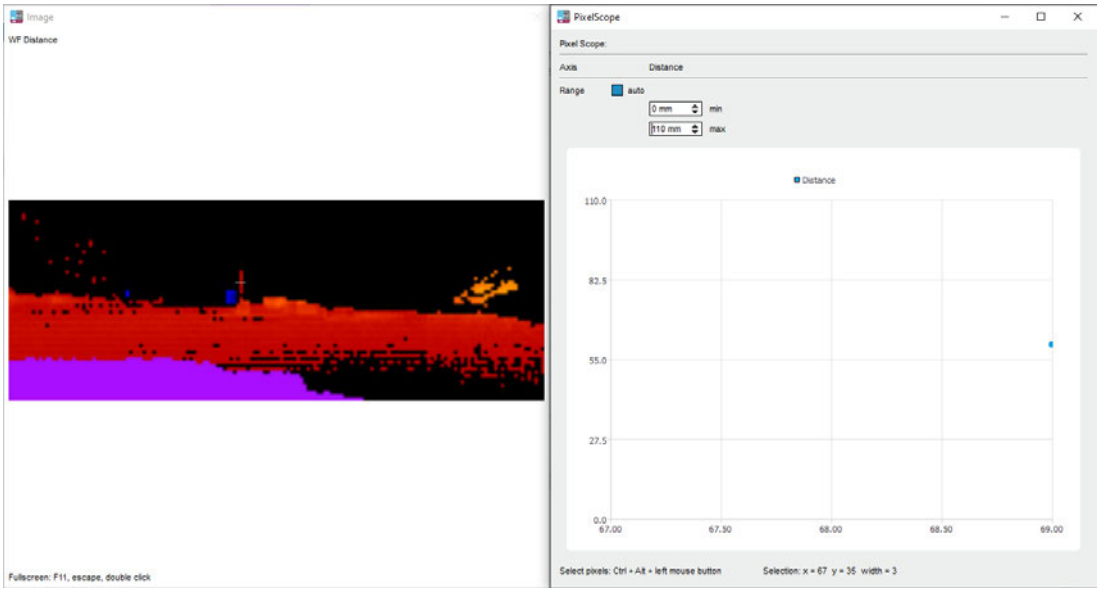

Abbildung 6.4: Messobjekt 4, 0.5 m, 105 μs

Die gemessenen Abstandswerte bewegen sich in etwa im Bereich von 60 mm.

Diese vier Messungen weisen zum Teil sehr große relative Fehler auf. Es muss allerdings erwähnt werden, dass die verwendeten Objekte sehr klein sind. Derart kleine Objekte auf kurze Distanzen zu detektieren ist keine typische Problemstellung in der Anwendung als Hilfsmittel für blinde und sehbehinderte Menschen. Hierfür sind eher größere Objekte, in der Größenordnung von Pfählen oder ähnlichem, relevant. Die zuvor dargestellten Messungen sind also eher Extremfälle, mit deren Hilfe bestimmt werden sollte, wie groß Objekte überhaupt sein müssen um detektiert werden zu können. Es ist also positiv zu bemerken, dass Objekt mit solch einer kleinen Größe, zumindest im Abstand von 0,5 m zuverlässig erkannt werden können, wenngleich deren Abstandsbestimmung sich als sehr ungenau erwiesen hat.

Um den Fall eines Laternenpfahls oder vergleichbarer Hindernisse zu testen wurde eine Papprolle aus einer handelsüblichen Küchentücher-Rolle mit einem Durchmesser von 44 mm als Testobjekt verwendet. Dieses verfügt zwar nicht über ein im Straßenverkehr üblicherweise verwendetes Material, allerdings über eine durchaus relevante Form.

Abb. 6.5 zeigt die Aufnahme der genannten Rolle im Abstand von 0,5 m, aufgenommen mit einer integration time von 105 μs.

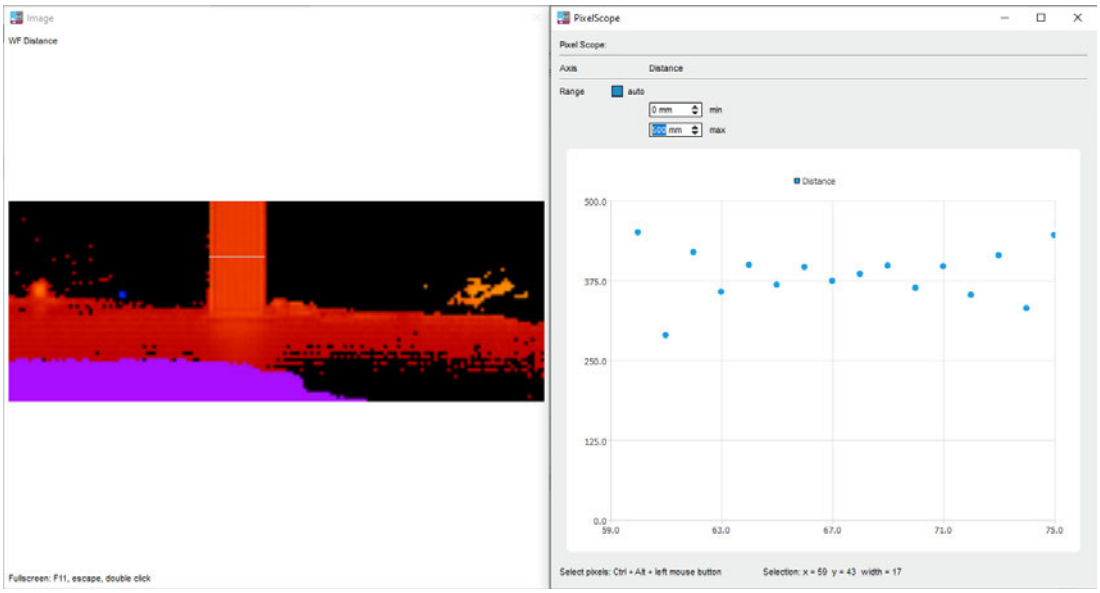

Abbildung 6.5: Papprolle, 0.5 m, 105 μs

Wie bei einem runden Objekt zu erwarten, gehen die Messwerte an den Randbereichen etwas auseinander. Im Zentrum liegen die Abstandswerte in etwa zwischen 375 mm und 390 mm.

Abb. 6.6 zeigt die Aufnahme der genannten Rolle im Abstand von 1 m aufgenommen mit einer integration time von 105 μs.

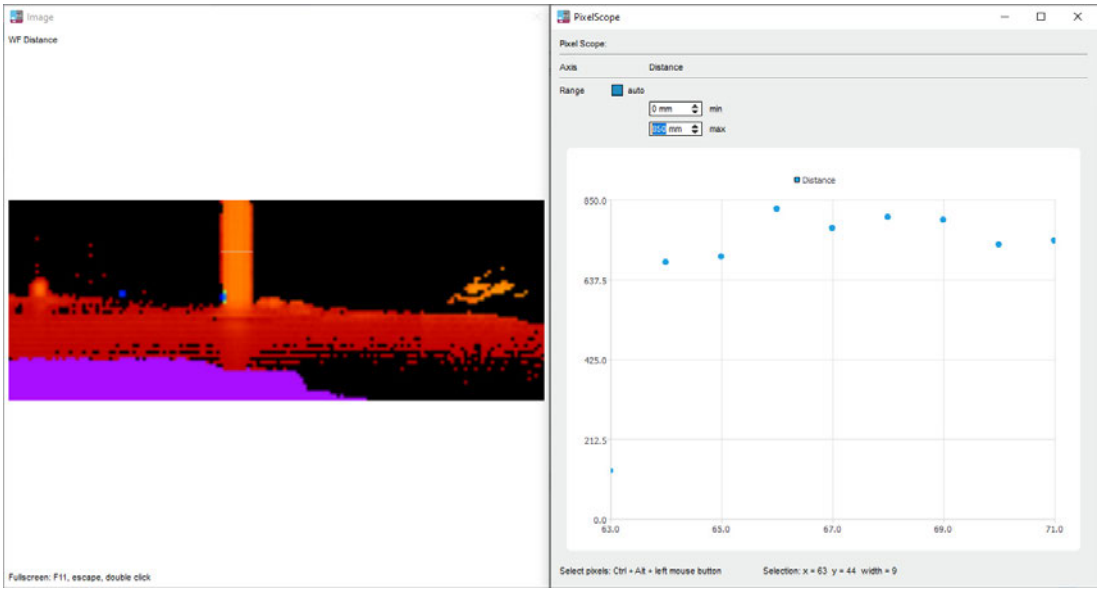

Abbildung 6.6: Papprolle, 1 m, 105 μs

Die gemessenen Werte liegen etwa zwischen 650 mm und 845 mm.

6.2 Messungen mit Mikrocontroller und Matlab

In den nachfolgenden Betrachtungen wird die ToF-Kamera mit Hilfe des verwendeten Mikrocontrollers ausgewertet und die Daten per serieller Schnittstelle an eine PC übertragen. Dort werden die Daten aufbereitet und visualisiert. Einige der vorherigen Messungen werden auf diese Art wiederholt. Bei den folgenden Messungen wurde exemplarisch jeweils ein Pixel angewählt, der sich in etwa mittig auf dem aufgenommenen Objekt befindet und dessen konkreter Abstandswert angezeigt wird. Hierbei ist zu beachten, dass je nach Form und Orientierung die umliegenden Pixel abweichende Werte enthalten können. In der gewählten Darstellung waren allerdings mehrere zeitgleich angezeigte Pixelwerte schlecht zu erkennen.

Das verwendete Matlab-Skript stellt die Kamerabilder als 3D-Scatterplot dar. Für die folgenden Abbildungen 6.7, 6.8, 6.9 und 6.10, der Messobjekte 1-4, wurden die 3D-Plots frontal betrachtet um mit Hilfe der Farblegende die Entfernungen besser einschätzen zu können.

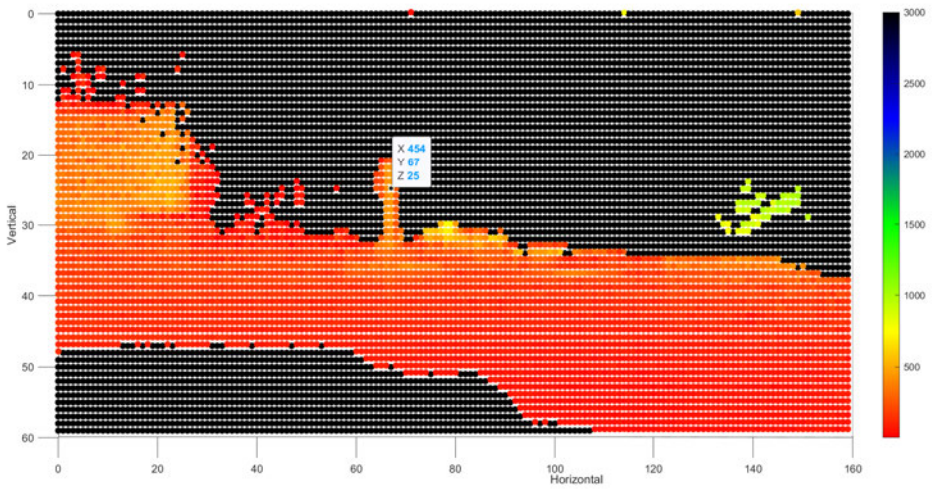

Abbildung 6.7: Messobjekt 1, 0.5 m, 105 μs, Matlab-Plot

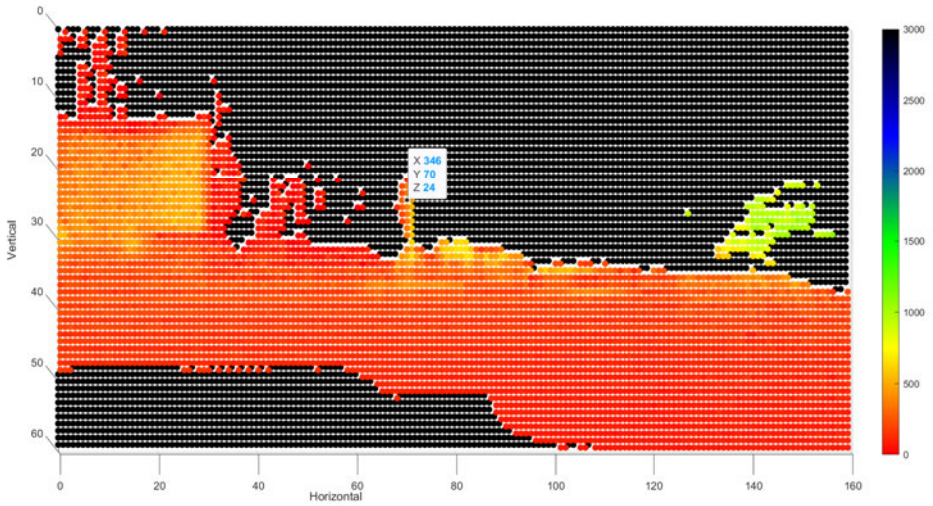

Abbildung 6.8: Messobjekt 2, 0.5 m, 105 μs, Matlab-Plot

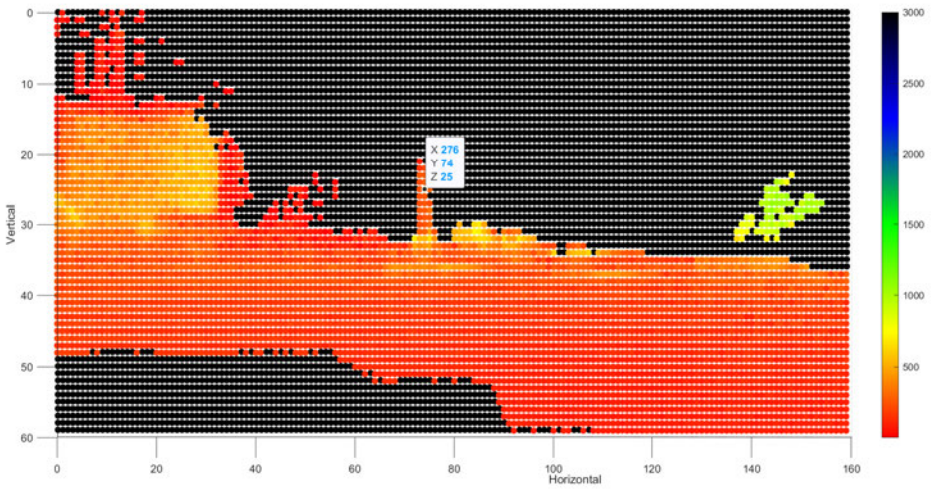

Abbildung 6.9: Messobjekt 3, 0.5 m, 105 μs, Matlab-Plot

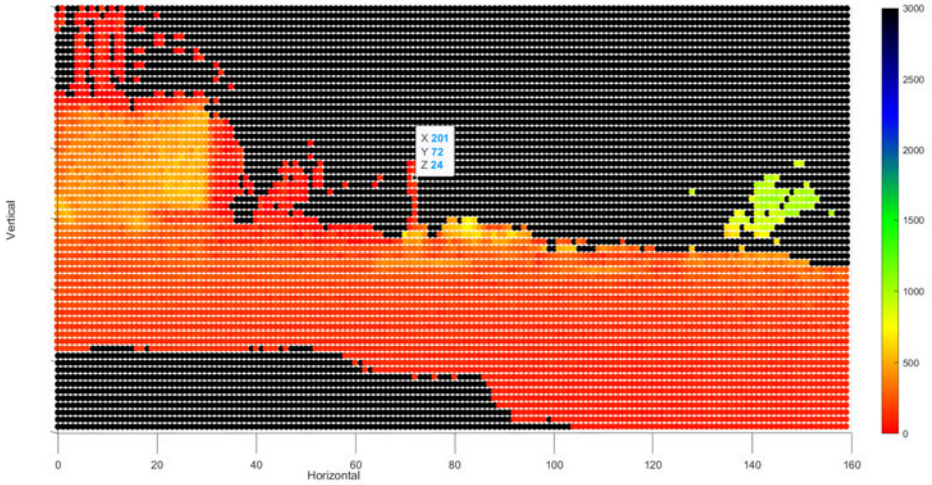

Abbildung 6.10: Messobjekt 4, 0.5 m, 105 μs, Matlab-Plot

Die Abb. 6.11, 6.12, 6.13 und 6.14 zeigen die Messergebnisse unterschiedlicher Kunststoffplatten aus dem Festo-Objektsortiment im Abstand von 0,5 m. Aufgrund der hohen direkten Reflektion der Objekte, muss auf die kurze Distanz von 0,5 m eine kleine integration time gewählt werden, um die Pixelsättigung und ADC-Überläufe zu minimieren.

Bis auf den transparenten Kunststoff wurden alle Kunststoffplatten mit einer integration time von 10 μs aufgenommen.

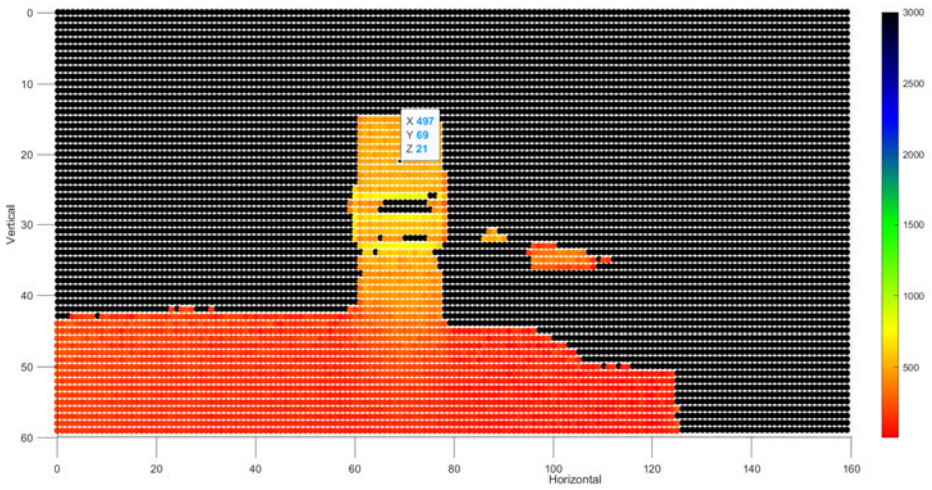

Abbildung 6.11: Kunststoff blau, 0.5 m, 10 μs

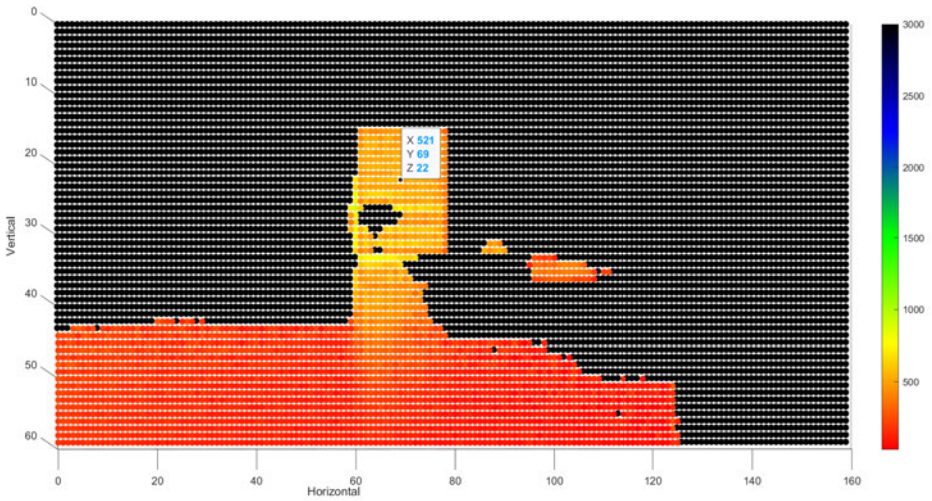

Abbildung 6.12: Kunststoff rot, 0.5 m, 10 μs

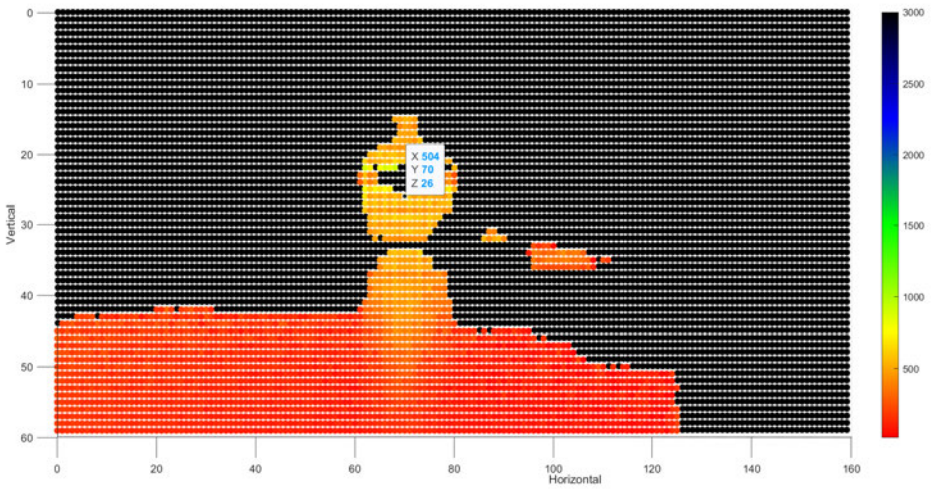

Abbildung 6.13: Kunststoff schwarz, 0.5 m, 10 μs

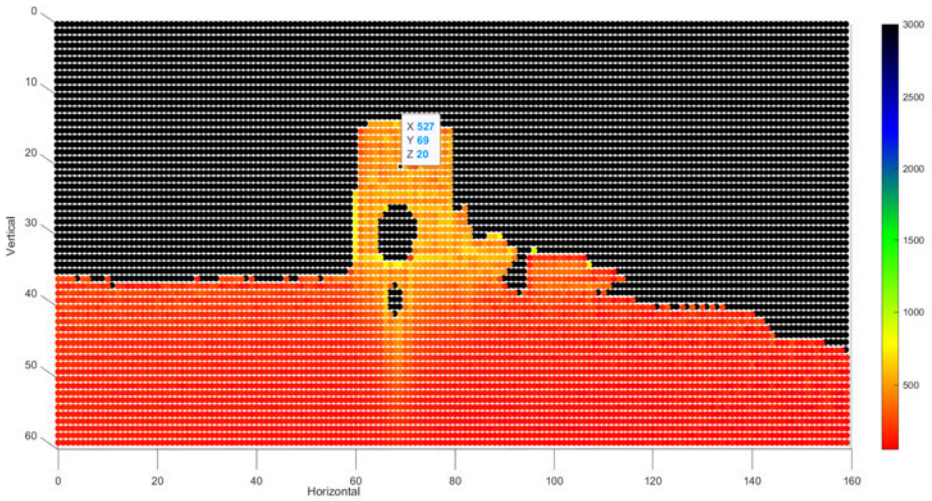

Abbildung 6.14: Kunststoff transparent, 0.5 m, 35 μs

Schwierigkeiten zeigten sich bei der Positionierung der Messobjekte, da die Orientierung große Auswirkungen auf die gemessenen Ergebnisse hat. Beim transparenten Kunststoff wurden eine integration time von 35 μs gewählt, da dieses Objekt ansonsten aufgrund der optischen Durchlässigkeit bei niedrigeren Werten der integration time kaum zu erkennen war. Anhand der vorliegenden Ergebnisse lassen sich nur teilweise Rückschlüsse auf den Einfluss unterschiedlicher Farben ziehen. Die gemessenen Abstandswerte liegen

bei allen Messungen um wenige Zentimeter Abweichung beieinander. Dies legt zunächst die Vermutung nahe, dass die unterschiedlichen Farben keinen signifikanten Einfluss auf die Messungen haben. Auffällig ist die weniger klare Kontur des schwarzen Messobjekts, allerdings muss hier auch die erwähnte hohe Anfälligkeit gegenüber abweichender Orientierung berücksichtigt werden. Aufgrund der begrenzten Möglichkeiten die Messobjekte präzise auszurichten, lassen sich die Ergebnisse also nur in gewissen Grenzen verlässlich miteinander Vergleichen.

Es zeigte sich während der Messungen, dass stark direkt reflektierenden Objekte, je nach Entfernung, bereits bei wenigen Grad Verdrehung um die vertikale Achse komplett aus dem aufgenommenen Bild verschwinden.

Die durchgeführten Messungen mit dem transparenten Kunststoff sprechen für die naheliegende Vermutung, dass transparente Objekte ein Problem für die Messungen mit der ToF-Kamera darstellen können.

Messungen an Fensterscheiben zeigten, dass aufgrund der hohen Reflektion bei direkter Belichtung, leicht ADC-Überläufe und Pixelsättigung in der Mitte des Bildes auftreten, an den Randbereichen allerdings ein Großteil des Lichtes weg reflektiert wird. Fensterscheiben sind also aus einem Bild der ToF-Kamera nicht ohne weiteres als solche zu erkennen. Eine Kombination z.B mit einem Ultraschallsensor würde hier voraussichtlich das Problem der Randbereiche nicht lösen, da auch der Schall bei zu großen Winkeln weg reflektiert würde. Allerdings könnte das Problem der ADC-Überläufe und Pixelsättigungen in den frontal beleuchteten Bereich durch Ultraschallsensoren ausgeglichen werden. Gleiches gilt für das Problem der optischen Durchlässigkeit, da der Schall, im Gegensatz zum Licht, ein transparentes Objekt nicht nennenswert durchdringen dürfte.

Die Abb. 6.15, 6.16 und 6.17 zeigen eine Holzplatte aus dem Festo-Objektsortiment in verschiedenen Abständen, bei einer integration time von 55 μs.

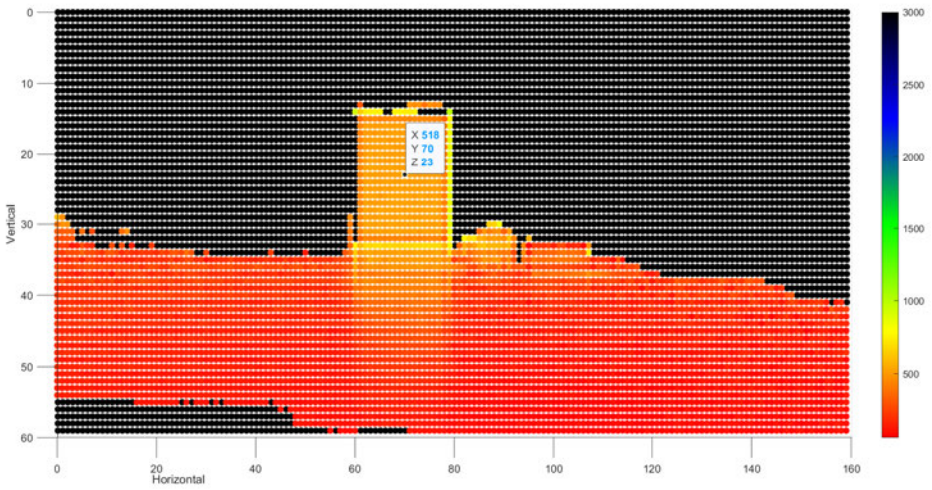

Abbildung 6.15: Holz, 0.5 m, 55 μs

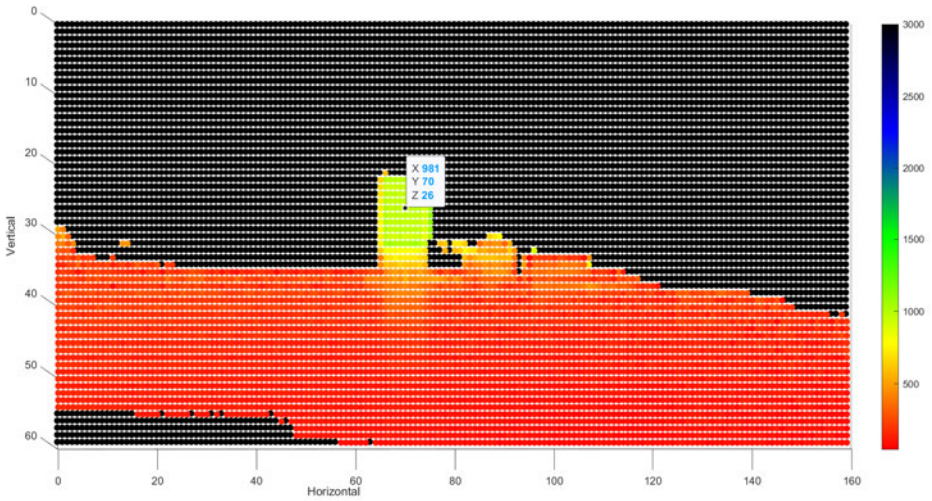

Abbildung 6.16: Holz, 1 m, 55 μs

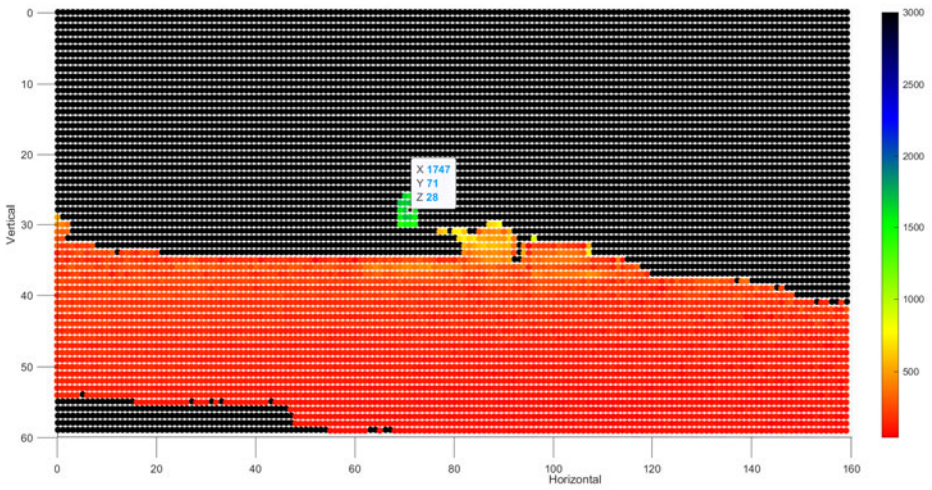

Abbildung 6.17: Holz, 2 m, 55 μs

Aufgrund der Oberflächenbeschaffenheit der Holzplatte, handelt es sich hier eher um diffuse Reflektion. Dadurch treten weniger ADC-Überläufe und Pixelsättigungen auf und das Licht wird bei kleineren Verdrehungen des Messobjekts weniger stark weg reflektiert, sodass hier zuverlässigere Messergebnisse erreicht werden können.

Die Papprolle kam, wie die Messobjekte 1-4, noch ein zweites Mal als Messobjekt zum Einsatz. Abb. 6.18, Abb. 6.19 und Abb. 6.20 zeigen die aufgenommenen Bilder als Matlab-Plot.

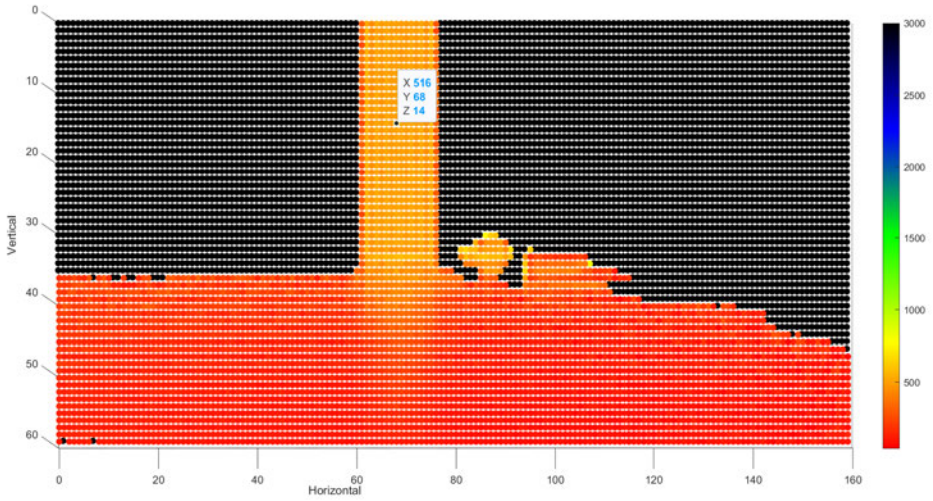

Abbildung 6.18: Papprolle, 0.5 m, 35 μs, Matlab-Plot

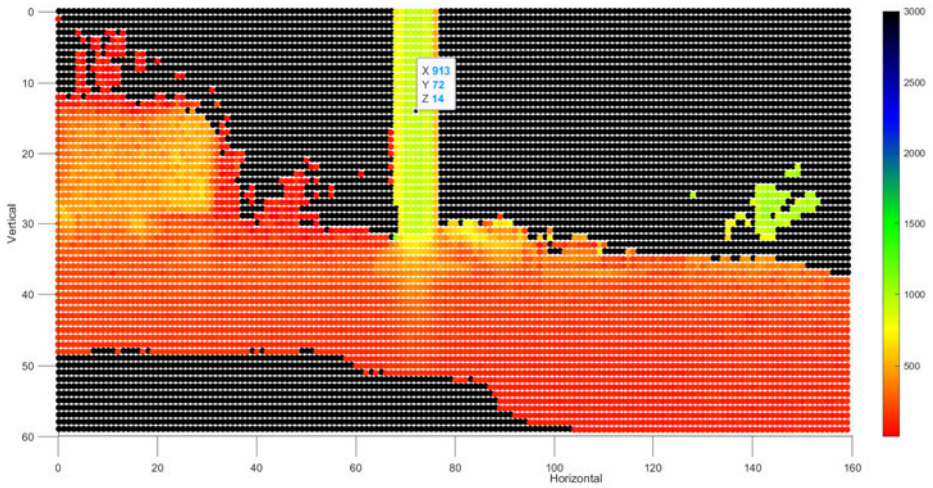

Abbildung 6.19: Papprolle, 1 m, 105 μs, Matlab-Plot

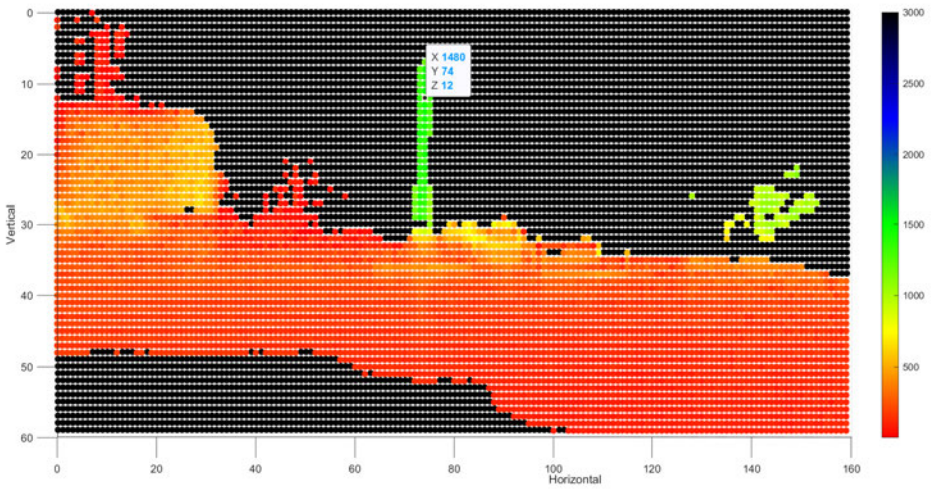

Abbildung 6.20: Papprolle, 2 m, 105 μs, Matlab-Plot

Im Abstand von 0,5 m und 1 m scheinen die Entfernungswerte nah an der tatsächlichen Entfernung zu liegen. Beim Abstand von 2 m ergeben sich vergleichsweise große Abweichungen. Die runde Form spielt hier vermutlich eine wichtige Rolle und begünstigt, dass das Objekt bei größeren Abständen in der Darstellung deutlich verschwimmt.

Zuletzt wird in Abb. 6.21 eine Aufnahme einer Person mit schwarz bekleidetem Oberkörper im Abstand von 2 m und mit einer integration time von 105 μs dargestellt. Diesmal wurden exemplarisch 14 Pixel der Aufnahme ausgewählt und deren Entfernungswerte angezeigt. Dabei wurden bewusst Pixel mit unterschiedlicher Färbung gewählt, um in etwa abzuschätzen in welchem Wertebereich die bestimmten Entfernungen sich bewegen.

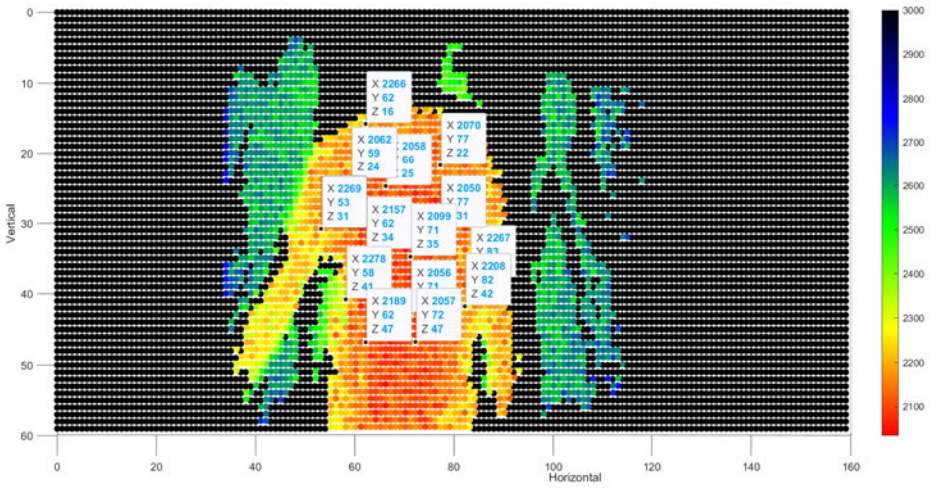

Abbildung 6.21: Person, 2 m, 105 μs

Die hier bestimmten Abstandswerte liegen zwischen 2,05 m und 2,278 m. Der gemessene Abstand ist bei diesen betrachteten Werten also bis zu etwa 28 cm größer als erwartet. Die größeren Abweichungen scheinen hier vor allem in den Randbereichen aufzutreten.

An dieser Stelle wurden wie erwähnt nur einzelne Pixelwerte betrachtet. In Zukunft könnten nah beieinanderliegende Pixel zusammengefasst werden, sodass Objekte als Einheit erfasst werden und eine Aussage darüber, wie weit das betreffende Objekt vermutlich tatsächlich entfernt ist, getroffen werden kann. Die Zusammenfassung von zusammengehörigen Bildbereichen wird in der Bildverarbeitung als Segmentierung bezeichnet [18].

6.3 Evaluation im Hinblick auf die Anforderungen

Im Folgenden wird auf die in der Aufgabenstellung formulierten Anforderungen eingegangen und deren Erfüllung durch das erarbeitete System untersucht.

- Erfassung von Hindernissen im Abstand von 30 cm bis zu mind. 2 m (idealerweise bis zu 3 m).

  → Laut Datenblatt liegt der Erfassungsbereich im WFOV-Betrieb bei 0,1 m bis 7,5 m [14]. Bei stark reflektierender Oberfläche eines Hindernisses kann ein geringer Messabstand zu ADC-Überläufen und/oder Pixelsättigungen führen, diese

können aber unter Umständen durch Anpassung der integration time verhindert werden. Die Reichweite der ToF-Kamera deckt die gewünschten Entfernungen ab. Je nach Einstellungen und Hindernis können sogar Hindernisse deutlich über 3 m Entfernung erfasst werden.

Alle verwendeten Messobjekte konnten in den durchgeführten Versuchen erfolgreich erfasst werden.

- Absicherung eines Bereichs in der Höhe von 2 m und der Breite 1,5 m vor dem EGD.

  → Die Absicherung des kompletten geforderten Bereichs kann nicht durch eine einzelne ToF-Kamera des verwendeten Typs realisiert werden. In der implementierten Konfiguration können nur Hindernisse in Bodennähe erkannt werden. Durch Verwendung mehrerer ToF-Kameras kann allerdings die Höhe des erfassbaren Bereichs, je nach Anzahl der Kameras beliebig vergrößert werden. Der begrenzende Faktor wäre hierbei das verfügbare Budget.

  In der Breite variiert der erfassbare Bereich mit der Entfernung zum Objekt. Mit Hilfe der in Abb. 6.22 dargestellten, nicht maßstäblichen Skizze soll das Verhältnis zwischen Entfernung und Breite des Erfassungsbereiches noch einmal verdeutlicht werden.

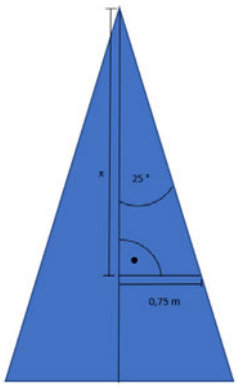

Abbildung 6.22: FoV von oben

Halbiert man das horizontale FoV, bei einem horizontalen Winkel von 50° und einer Breite von 1,5 m, ergeben sich die eingezeichneten Werte.

Der Abstand x, in welchem der Erfassungsbereich eine Breite von 1,5 m aufweist, lässt sich so zu etwa

$x = \frac{0.75m}{\tan 25°} \approx 1.61m$

berechnen.

In kürzeren Abständen ist die Breite entsprechend geringer.

- Dieser Bereich soll keine signifikanten Lücken haben, sodass typische Hindernisse wie Pfähle, Zäune, vorstehende Objekte nicht übersehen werden.

  → Formen von typischen Hindernissen konnten in den Messungen detektiert werden. Signifikante Lücken wurden hierbei nicht entdeckt.

- Die Auflösung soll $< 20°$ in Azimut und Elevation betragen.

  → Es wurden keine quantitativen Messungen oder Berechnungen zu diesem Punkt durchgeführt, die Auflösung von 160 x 60 Pixel sollte allerdings hoch genug sein.

- Der Sensor soll auf einer Fläche von max. $200\,\mathrm{cm}^2$ untergebracht werden können.

  → Die ToF-Kamera selbst kann auf deutlich kleinerer Fläche untergebracht werden. Je nach Anordnung der restlichen Hardware sollte diese Anforderung durchaus einzuhalten sein.

- Es dürfen mehrere Einzelsensoren zu einem Array kombiniert werden.

  → Mit der Verwendung einer vollständigen ToF-Kamera ist diese Anforderung hinfällig.

- Die Tiefenbildszene, die sich aus den Einzelsensorsignalen ergibt, soll mit einem Mikrocontroller auswertbar sein.

  → Die Tiefenbildszene ist mit dem Mikrocontroller auswertbar. Dies wurde erfolgreich implementiert und getestet.

- Der Sensor soll für den Innen und Außenbereich, auch bei Sonnenlicht, geeignet sein.

  → Messungen im Außenbereich wurden nicht durchgeführt. Der Einfluss des Umgebungslichts steigt mit der eingestellten integration time [14]. Bei Messungen im Innenraum hatte das einfallende Sonnenlicht keinen schwerwiegenden Einfluss.

- Der Sensor soll mit Hilfe eines Adapters, der im 3D-Druckverfahren entwickelt werden kann, auf dem EGD fest verbunden werden.

  → Ein mechanischer Adapter wurde erfolgreich entwickelt, gefertigt und am EGD verschraubt.

- Es sollen die Abstandsinformationen mit einem Neigungssensor kombiniert werden bzw. kombinierbar sein, um Einflüsse bspw. des Bodens zu eliminieren.

  → Ein Neigungssensor ist im Rahmen dieser Arbeit nicht verwendet worden, kann aber durchaus in späteren Implementierung hinzugefügt werden.

- Die Auswertung des Sensors soll „embedded“ erfolgen oder zumindest möglich sein, d.h. nicht auf einem PC sondern einem Mikrocontroller.

  → Die Sensorauswertung durch einen Mikrocontroller ist möglich und wurde erfolgreich in Betrieb genommen.

- Optional: es soll eine Strategie gefunden werden, wie die Tiefenbildszene interpretiert werden kann und Ausweichstrategien, die an die Steuerung des Motors gegeben werden, entwickelt werden.

  → Ansätze zum Ausweichverhalten wurden theoretisch betrachtet und verglichen. Implementiert werden konnte allerdings noch keiner der erwähnten Algorithmen.

- Kosten: das Evaluationssystem soll unter 1000 Euro bleiben.

  → Die bisherigen Kosten bleiben deutlich unter der Grenze von 1000 Euro. Der Preis der ToF-Kamera liegt bei 180,52 Euro(inklusive MwSt.) [8], der Preis für den verwendeten USB-Adapter bei 163,99 Euro (inklusive MwSt.) [9]. Da kein weiterer USB-Adapter benötigt wird, sollte es auf jeden Fall möglich sein drei weitere ToF-Kameras zu erwerben, ohne das Budget von 1000 Euro zu überschreiten. Es muss in der Praxis untersucht werden, ob sich eine Anordnung finden lässt, in der vier Kameras ausreichen um den gewünschten Bereich abzudecken. Da in der bisherigen Anordnung schon zwei Kameras für den Bereich Boden und Bodennähe vorgesehen sind, sollte überprüft werden ob zu der bisher vorhandenen auch vier oder fünf weitere ToF-Kameras erworben werden können. Des Weiteren sollte ein passender Akku und eventuell Gyro- und Ultraschallsensoren erworben werden, um das bestehende System zu ergänzen.

- Es soll nach Möglichkeit ein Evaluationssystem zum schnellen Test (kurze Lieferzeit) zur Verfügung stehen.

  → Der verwendete USB-Adapter bietet die Möglichkeit zur schnellen Evaluation und konnte mit der ToF-Kamera geliefert werden.

- Es soll ausreichend Unterstützung geben (z.B. Schaltpläne und Gerberfiles bzw. Layoutempfehlungen) um in einem späteren Schritt eine maßgeschneiderte Lösung entwickeln zu können.

  → Die vorhandene Unterstützung ist ausreichend, um die ToF-Kamera in ein maßgeschneidertes System zu integrieren.

- Evaluation:

  – Wie groß müssen Objekte mindestens sein, damit sie erkannt werden können?

    → Die vorangegangenen Messungen zeigen, dass, zumindest bei kürzeren Abständen, selbst kleine Objekte wie die genannten Messobjekte 1-4 erkannt werden können.

  – Welchen Einfluss hat das Material, die Entfernung, die Form und Orientierung des Objektes?

    → Besonders die Oberflächenbeschaffenheit hat einen Einfluss auf die Messung. Stark direkt reflektierende Oberflächen führen zu Problemen mit ADC-Überläufen und Pixelsättigungen. Außerdem werden bei solchen Objekten schon bei leichten Verdrehungen große Teile des Lichts weg reflektiert. Die als Messobjekt verwendete Holzplatte, welche durch ihre rauere Oberfläche eher diffus reflektiert, war dagegen auch bei Verdrehung um größere Winkel noch zu erkennen, wenngleich sich die Breite des auf dem Bild zu erkennenden Objekts entsprechend verringert.

  – Welchen Einfluss haben Umgebungsbedingungen wie das Sonnenlicht?

    → In den bisherigen Messungen wurden keine schwerwiegenden Probleme durch Sonnenlicht bemerkt. Der Einfluss des Umgebungslichts steigt aber wie erwähnt mit der integration time [14]. Direkte Beleuchtung des Pixelarrays der ToF-Kamera mit der Taschenlampe eines Smartphones zeigte experimentell keinen signifikanten Einfluss auf das Bild. Allerdings ist das Spektrum von Sonnenlicht in der Regel sehr viel größer als das von künstlichem Licht, sodass

es wahrscheinlich ist, dass auch ein Teil des Sonnenlichts den Empfänger der ToF-Kamera beeinflusst [4].

- Welchen Einfluss haben Wände und Fußboden auf das Ergebnis?

  → In den durchgeführten Messungen wurden Wände und Fußboden in der Regel wie andere Objekte auch erkannt, sofern sie im Erfassungsbereich der ToF-Kamera lagen.

- Optional: lassen sich Gefährdungen, wie niedergehende Treppen oder Bordsteine erkennen?

  → Je nach Orientierung der Kamera werden solche Gefährdungen durchaus erkannt. Aufgrund des kleinen FoVs in der vertikalen Dimension muss allerdings eine Kamera deutlich zum Boden hin geneigt werden um den Boden zu beobachten. Um einen Treppenabgang als solchen zu erkenne, sollten Klassifizierungsverfahren aus dem Bereich der Bildverarbeitung untersucht werden.

7 Fazit und Ausblick

In dieser Arbeit sollte ein Sensorsystem für ein Blindenhilfsmittel ausgewählt und in Betrieb genommen werden, mit dessen Hilfe potentielle Hindernisse im Weg einer blinden oder sehbehinderten Person erkannt werden können, sodass diese Person sicher navigiert werden kann. Es konnte ein geeigneter Sensor, die TOF>cam 635, ausgewählt und beschafft werden. Außerdem konnte ein passender Mikrocontroller zum Auslesen der Sensordaten ausgewählt werden.

Mögliche Fallstricke bei der Inbetriebnahme des Sensorsystems, welche sich im Verlauf der Arbeit an dieser Bachelorthesis zeigten, konnten beseitigt werden. Die verwendete ToF-Kamera konnte erfolgreich in Betrieb genommen und mit Hilfe der entwickelten Mikrocontroller-Software ausgelesen werden. Das Mikrocontroller-Programm kann über das vorgegebene Kommunikationsprotokoll der ToF-Kamera erfolgreich mit dieser kommunizieren, d.h Kommandos an die ToF-Kamera versenden und die Antwort der ToF-Kamera einlesen. Des Weiteren können die Messdaten vom Mikrocontroller über einen seriellen COM-Port an einen PC oder Laptop gesendet und über die erarbeiteten Matlab-Skripte empfangen, visualisiert und überprüft werden.

In diversen Messungen erwies die TOF>cam 635 sich als durchaus vielversprechend, im Hinblick auf die Erfüllung der meisten der festgelegten Anforderungen.

Es konnte ein mechanischer Adapter entwickelt und gefertigt werden, welcher es ermöglicht die ToF-Kamera am EGD-Prototyp zu positionieren und zu befestigen. Außerdem steht bereits die Möglichkeit zur Verfügung eine zweite Kamera anzubringen. Hierbei muss in der Praxis getestet werden ob zwei Kameras in dieser Anordnung sinnvoll parallel genutzt werden können, ohne dass die Lichtstrahlen sich gegenseitig behindern oder die Lücke zwischen den beiden FoVs zu groß ist. Für weitere Kameras zur Sicherung des Kopfes, Oberkörpers etc. eines Anwenders müssen neue Adapter entwickelt und am EGD untergebracht werden. In der im Rahmen dieser Arbeit entwickelten Konfiguration können nur Hindernisse in Bodennähe und ein Teil des Bodens beobachtet werden.

Es sollte außerdem geprüft werden wie das Gesamtsystem um weitere Sensorkonzepte, wie Ultraschall ergänzt werden kann, um einige Nachteile der optischen Sensorik auszugleichen. Transparent Objekte, welche eine zu große Menge des einfallenden Lichts durchlassen, könnten durch Ultraschall vermutlich besser erkannt werden. Auch auf kurze Distanzen, besonders bei stark reflektierenden Oberflächen könnte ein Ultraschallsensor die Bereiche der Szene, welche aufgrund von ADC-Überläufen und Pixelsättigung mit der Kamera nicht erkannt werden können, abdecken. Gyrosensoren, kombiniert mit Motoren, könnten zukünftig eingesetzt werden, um den vertikalen Winkel der Kameras dynamisch anzupassen und so Größenunterschiede der potentiellen Anwender auszugleichen und um die betrachtete Szene im dreidimensionalen Raum korrekt zu interpretieren.

Bei der Interpretation der Messdaten und den resultierenden Kommandos zur Steuerung der Motoren des EGD muss in Zukunft noch viel Arbeit investiert werden. Navigationsalgorithmen wurden bisher nur theoretisch betrachtet und noch nicht implementiert. Zukünftige Arbeiten können hierbei auf die bisherigen Ergebnisse zurückgreifen und sich daher gezielt auf die Implementierung von Algorithmen konzentrieren. Aus dem Bereich der industriellen Bildverarbeitung stehen vielfältige Möglichkeiten zur Filterung und Verarbeitung von Sensordaten zur Verfügung, die im Rahmen dieser Arbeit nicht implementiert werden konnten. Zunächst könnten Rauschfilter zum Einsatz kommen um rauschärmere Messdaten zu erhalten. Des Weiteren können Pixelgruppen zusammengefasst (Segmentierung [18]) und diese als unterschiedliche Objekte klassifiziert werden.

Hierbei sollten in Zukunft auch die confidence bits der Pixeldaten betrachtet werden. Die confidence bits geben Auskunft darüber, wie zuverlässig der Messwert eines Pixels ist und könnten daher verwendete werden, um verlässliche von weniger verlässlichen Werten zu unterschieiden und so ein zuverlässigeres Gesamtbild der Szene zu bekommen [14]. Es sollten auch Möglichkeiten zur dynamischen Anpassung der integration time untersucht werden, um z.B. auf unterschiedliche Lichtverhältnisse zu reagieren.

Ansätze aus dem Machine Learning, wie die Klassifizierung von Objekten könnten in Zukunft untersucht werden. Auch aus dem Bereich der Routenplanung für mobile Roboter bieten sich zahlreiche Konzepte, welche in zukünftigen Arbeiten untersucht und an den EGD angepasst werden können.

Die entwickelte Mikrocontroller-Software ist bisher so konzipiert, dass beim Warten auf UART-Zeichen die CPU blockiert wird, da bisher nur das zuverlässige Empfangen der Bilddaten Priorität hatte. Wenn auf dem Mikrocontroller weitere Prozesse implemen-

tiert werden, sollte in Betracht gezogen werden, die UART-Kommunikation Interrupt-gesteuert zu verwenden.

Das in dieser Arbeit erarbeitete System ermöglicht zukünftigen Entwicklern eine modulare Weiterentwicklung des EGDs. Algorithmen können in der Praxis getestet werden, da eine funktionierende Basis zur Verfügung steht. Durch die in dieser Arbeit geschaffenen Grundlagen sollte zukünftigen Entwicklern ein erleichterter Einstieg in die Materie möglich sein.

Literaturverzeichnis

[1] Analog Devices: *AD-96TOF1-EBZ Analog Devices Website. [Online].* – URL https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/ad-96tof1-ebz.html#eb-overview. – Letzter Zugriff am 26.03.2020

[2] Analog Devices: *AD-96TOF1-EBZ User Guide. 2020. [Online].* – URL https://wiki.analog.com/resources/eval/user-guides/ad-96tof1-ebz. – Letzter Zugriff am 21.03.2020

[3] Anycubic: *Anycubic I3 Mega Website. [Online].* – URL https://www.anycubic.com/products/anycubic-i3-mega. – Letzter Zugriff am 26.03.2020

[4] Basler AG: *WHITE PAPER Hochauflösende Lichtlaufzeit-Kamera (Time-of-Flight-Kamera).* 2019. – URL https://www.baslerweb.com/de/vertrieb-support/downloads/downloads-dokumente/time-of-flight-kameras/. – Letzter Zugriff 16.02.2020

[5] Baumer GmbH: *Funktionsweise und Technologie von Radarsensoren. [Online].* – URL https://www.baumer.com/de/de/service-support/know-how/funktionsweise/funktionsweise-und-technologie-von-radarsensoren/a/Know-how_Function_Radar-sensors. – Letzter Zugriff am 26.03.2020

[6] Bräunl, Thomas: *Embedded Robotics.* Springer-Verlag Berlin Heidelberg, 2008. – ISBN 9783540705338

[7] De Coi, Beat: *3D-TOF A guideline to 3D-TOF sensors that work.* espros photonics corporation, 2018. – ISBN 9783033070967

[8] Digi-Key: *Digi-Key Versandhändler, TOF>CAM 635. [Online].* – URL https://www.digikey.de/product-detail/de/espros-photonics-ag/TOF-CAM-635/2199-TOF-CAM635-ND/10516857. – Letzter Zugriff am 30.03.2020

[9] Digi-Key: *Digi-Key Versandhändler, TOF>CAM 635 USB ADAPTER KIT. [Online].* – URL https://www.digikey.de/product-detail/de/espros-photonics-ag/TOF-CAM-635-USB-ADAPTER-KIT/2199-TOF-CAM635USBADAPTERKIT-ND/10516868. – Letzter Zugriff am 30.03.2020

[10] espros photonics corporation: *epc Mechanical models Downloadseite. [Online].* – URL https://www.espros.com/downloads/02_Cameras_and_Modules/Mechanical_models/. – Letzter Zugriff am 30.03.2020

[11] espros photonics corporation: *epc Software-Packages Downloadseite. [Online].* – URL https://www.espros.com/downloads/02_Cameras_and_Modules/Software-Packages/. – Letzter Zugriff am 26.03.2020

[12] espros photonics corporation: *TOF>cam 635 epc Website. [Online].* – URL https://www.espros.com/photonics/tofcam635/. – Letzter Zugriff am 26.03.2020

[13] espros photonics corporation: *DATASHEET - epc635 3D TOF imager 160 x 60 pixel V2.11.* 2019. – URL https://www.espros.com/downloads/01_Chips/Datasheet_epc635.pdf. – Letzter Zugriff am 06.02.2020

[14] espros photonics corporation: *TOF>cam 635 Installation and Operation Manual.* 2019. – URL https://www.espros.com/downloads/02_Cameras_and_Modules/Installation_and_Operation_Manual_TOFCAM635.pdf. – Letzter Zugriff am 06.01.2020

[15] Festo Didactic: *Festo Didactic Website. [Online].* – URL https://www.festo-didactic.com/de-de/. – Letzter Zugriff am 29.03.2020

[16] Festo Didactic GmbH & Co. KG: *Festo Objektsortiment.* 2009. – URL https://www.haw-hamburg.de/ti-ie/labore/grundlagen-elektrotechnik/download/datenblaetter.html. – Letzter Zugriff am 29.03.2020

[17] Hertzberg, Joachim ; Lingemann, Kai ; Nüchter, Andreas: *Mobile Roboter Eine Einführung aus Sicht der Informatik (ebook)*. Springer-Verlag Berlin Heidelberg, 2012. – ISBN 9783642017261

[18] Meisel, Prof. Dr.-Ing. A.: *Vorlesungsskript Robot Vision, HAW Hamburg*. 2017

[19] PTC: *Creo Parametric 3D-Modellierungssoftware PTC Website. [Online]*. – URL https://www.ptc.com/de/products/cad/creo/parametric. – Letzter Zugriff am 26.03.2020

[20] Puttkamer, Prof. Dr. E. von: *Vorlesungsskript Autonome Mobile Roboter, Wintersemester 1999/2000, University of Kaiserslautern, Department of Computer Science*. – URL http://ag-vp-www.informatik.uni-kl.de/Papers/skriptamr/contents.html. – Letzter Zugriff am 21.03.2020

[21] Texas Instruments: *Tiva^{TM} TM4C1294NCPDT Microcontroller DATA SHEET*. 2015. – URL http://www.ti.com/lit/ds/symlink/tm4c1294ncpdt.pdf. – Letzter Zugriff am 06.01.2020

[22] Texas Instruments: *Tiva^{TM} C Series TM4C1294 Connected LaunchPad Evaluation Kit EK-TM4C1294XL User's Guide*. 2016. – URL http://www.ti.com/lit/ug/spmu365c/spmu365c.pdf. – Letzter Zugriff am 19.03.2020

[23] Texas Instruments: *TivaWare^{TM} Peripheral Driver Library User's Guide*. 2016. – URL https://www.ti.com/lit/ug/spmu298d/spmu298d.pdf. – Letzter Zugriff am 16.01.2020

[24] Texas Instruments Incorporated: *CCS Texas Instruments Website. [Online]*. – URL http://www.ti.com/tool/CCSTUDIO. – Letzter Zugriff am 26.03.2020

[25] Texas Instruments Incorporated: *EK-TM4C1294XL Texas Instruments Website. [Online]*. – URL http://www.ti.com/tool/EK-TM4C1294XL. – Letzter Zugriff am 26.03.2020

[26] The MathWorks, Inc.: *MathWorks Matlab Website. [Online]*. – URL https://www.mathworks.com/products/matlab.html. – Letzter Zugriff am 31.03.2020

[27] The Qt Company: *Qt Website. [Online]*. – URL https://www.qt.io/. – Letzter Zugriff am 30.03.2020

[28] Ultimaker: *Ultimaker Cura Website. [Online].* – URL https://ultimaker.com/de/software/ultimaker-cura. – Letzter Zugriff am 26.03.2020

[29] Wikipedia: *CCD-Sensor — Wikipedia, Die freie Enzyklopädie.* 2019. – URL https://de.wikipedia.org/w/index.php?title=CCD-Sensor&oldid=194865912. – [Online; Stand 6. Februar 2020]

[30] Wikipedia: *TOF-Kamera — Wikipedia, Die freie Enzyklopädie.* 2019. – URL https://de.wikipedia.org/w/index.php?title=TOF-Kamera&oldid=189220682. – [Online; Stand 6. Februar 2020]

[31] Wikipedia: *Blindenhund — Wikipedia, Die freie Enzyklopädie.* 2020. – URL https://de.wikipedia.org/w/index.php?title=Blindenhund&oldid=196295195. – [Online; Stand 6. Februar 2020]

[32] Wikipedia: *Dijkstra-Algorithmus — Wikipedia, Die freie Enzyklopädie.* 2020. – URL https://de.wikipedia.org/w/index.php?title=Dijkstra-Algorithmus&oldid=197575866. – [Online; Stand 30. März 2020]

[33] Wikipedia: *Radar — Wikipedia, Die freie Enzyklopädie.* 2020. – URL https://de.wikipedia.org/w/index.php?title=Radar&oldid=196139774. – [Online; Stand 30. März 2020]

[34] Wikipedia: *Universal Asynchronous Receiver Transmitter — Wikipedia, Die freie Enzyklopädie.* 2020. – URL https://de.wikipedia.org/w/index.php?title=Universal_Asynchronous_Receiver_Transmitter&oldid=195873295. – [Online; Stand 7. Februar 2020]

A Anhang

A.1 Inhalt der CD

- Bachelorthesis
- Mikrocontroller-Software als CCS-Projekte
- Matlab-Dateien
- Datenblätter von ToF-Kamera und Mikrocontroller
- CAD-Dateien

A.2 Quellcode

Nachfolgend sind die Quellcode-Dateien, welche am meisten zum Verständnis dieser Arbeit beitragen dürften, angefügt. Eine vollständige Sammlung aller verwendeten Dateien befindet sich auf der CD.

A.2.1 CCS-Projekt TOFcam

main.c

```
#include <inc/tm4c1294ncpdt.h>
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
```

```
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/timer.h"

#include "tivaUART.h"
#include "crc.h"

#define COMMANDLENGTH 14
#define ROWS 60
#define COLUMNS 160
#define RESPONSE_HEADER_SIZE 80

uint32_t sysClk;

int main(void){
    //Configure UART6 for Communication with ToF-Kamera
    //set clock frequency 120 MHz
    sysClk = MAP_SysCtlClockFreqSet((SYSCTL_XTAL_25MHZ |
                                     SYSCTL_OSC_MAIN |
                                     SYSCTL_USE_PLL |
                                     SYSCTL_CFG_VCO_480), 120000000);

    //configure UART6 with Bitrate 10 Mbit/s, 8 Data Bits, 1 Stop Bit, Parity
        None
    configUART6(sysClk, 10000000, UART_CONFIG_WLEN_8, UART_CONFIG_STOP_ONE,
        UART_CONFIG_PAR_NONE);

    //Communication via virtual COM port
    //configure UART0 with Bitrate 128000 bit/s, 8 Data Bits, 1 Stop Bit,
        Parity None
    configUART0(sysClk, 128000, UART_CONFIG_WLEN_8, UART_CONFIG_STOP_ONE,
        UART_CONFIG_PAR_NONE);

    //set integration time to 35 us: 0x23, 105 us: 0x69
    uint8_t SET_INT_TIME_DIST[COMMANDLENGTH] = {0xF5, 0x00, 0x00, 0x69, 0x00,
         0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    //calculate CRC-checksum
    uint32_t checksum = calcCrc32_32(SET_INT_TIME_DIST, 10);

    //store checksum-bytes in command-array
    int j = 0;
    for(j = 0; j < 4; j++){
```

```
        SET_INT_TIME_DIST[j+10] = (uint8_t)((checksum>>(8*j)) & 0x000000FF);
    }

    //GET_DIST single measurement, 0xF5: start, 0x20: GET_DIST, 0x00: single
        measurement
    uint8_t GET_DIST[COMMANDLENGTH] = {0xF5, 0x20 , 0x00, 0x00, 0x00, 0x00, 0
        x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};

    //calculate CRC-checksum
    checksum = calcCrc32_32(GET_DIST, 10);

    //store checksum-bytes in command-array
    for(j = 0; j < 4; j++){
        GET_DIST[j+10] = (uint8_t)((checksum>>(8*j)) & 0x000000FF);
    }

    //variables to store sensordata
    uint8_t receivedByte = 0;
    //data is read byte per byte
    uint8_t pixelData[ROWS*COLUMNS*2] = {0};

    //variables to store start, length and CRC bytes from ToF-response
    uint8_t start = 0;
    uint8_t type = 0;
    uint8_t length[2] = {0};
    uint8_t crc[4] = {0};

    //array to store response to SET command
    uint8_t set_response[8] = {0};

    //transmit SET command
    UARTsend(UART6_BASE, SET_INT_TIME_DIST, 14);
    for(j = 0; j < 8; j++){
        set_response[j] = UARTCharGet(UART6_BASE);
    }

    //wait for character from MATLAB script
    while(!UARTCharsAvail(UART0_BASE));

    //check for correct character
    while('s' != UARTCharGet(UART0_BASE));

    //empty UART0 FIFO
    while(UARTCharsAvail(UART0_BASE)){
```

```
        UARTCharGet(UART0_BASE);
    }

    //empty UART6 FIFO
    while(UARTCharsAvail(UART6_BASE)){
        UARTCharGet(UART6_BASE);
    }

    int imageCounter = 0;

    //send command to get image
    UARTsend(UART6_BASE, GET_DIST, 14);

    int i = 0;

    /*
    //Measure time for image acquisition...
    double time = 0;

    //configure Timer0
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    ROM_TimerConfigure(TIMER0_BASE, TIMER_CFG_ONE_SHOT_UP);
    ROM_TimerLoadSet(TIMER0_BASE, TIMER_A, 0xFFFFFFFF);
    ROM_TimerControlStall(TIMER0_BASE, TIMER_A, true);

    ROM_TimerEnable(TIMER0_BASE, TIMER_A);*/
    while(1){
        //response format: 1 byte start, 1 byte type, 2 bytes length n, n
            bytes data, 4 bytes CRC
        //for GET_DIST 80 byte header -> 160 x 60 pixel data + 80 byte header
             + 8 bytes

        for(i = 0; i < ROWS*COLUMNS*2+RESPONSE_HEADER_SIZE+8; i++){
            receivedByte = UARTCharGet(UART6_BASE);

            //store start byte
            if(0 == i){
                start = receivedByte;
            }

            //store type byte
            else if(1 == i){
                type = receivedByte;
            }
```

```
            //store length bytes
            else if(2 == i){
                length[1] = receivedByte;
            }

            else if(3 == i){
                length[0] = receivedByte;
            }

            //store CRC values
            else if(i >= ROWS*COLUMNS*2+RESPONSE_HEADER_SIZE+4){
                crc[i-(ROWS*COLUMNS*2+RESPONSE_HEADER_SIZE+4)] = receivedByte
                    ;
            }

            //store pixel data
            else if(i >= RESPONSE_HEADER_SIZE+4) {
                pixelData[i-(RESPONSE_HEADER_SIZE+4)] = receivedByte;
            }
        }
        imageCounter++;

        /*if(imageCounter >= 500){
            //measure required time
            time = (double)ROM_TimerValueGet(TIMER0_BASE, TIMER_A)/(double)
                sysClk;
        }*/

        //transmit pixelData to MATLAB script
        UARTsend(UART0_BASE, pixelData, ROWS*COLUMNS*2);


        //get new image, do not send this when operating in streaming mode!
        UARTsend(UART6_BASE, GET_DIST, 14);
    }
}
```

tivaUart.c

```
/*
 * tivaUART.c
 *
 *  Created on: 17.01.2020
```

```
 *      Author: max_r
 */

#include <tivaUART.h>

void configUART6(uint32_t sysClk, uint32_t bitrate, uint32_t dataLength,
    uint32_t stopBits, uint32_t parity){
    //Enable PORTP for UART6
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART6);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOP);

    GPIOPinConfigure(GPIO_PP0_U6RX);
    GPIOPinConfigure(GPIO_PP1_U6TX);
    ROM_GPIOPinTypeUART(GPIO_PORTP_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    //configure UART6
    ROM_UARTConfigSetExpClk(UART6_BASE, sysClk, bitrate,
                            (dataLength | stopBits |
                             parity));

    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_UART6));
}

void configUART0(uint32_t sysClk, uint32_t bitrate, uint32_t dataLength,
    uint32_t stopBits, uint32_t parity){
    //Enable PORTA for UART0
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    ROM_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);

    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    ROM_GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

    //configure UART0
    ROM_UARTConfigSetExpClk(UART0_BASE, sysClk, bitrate,
                            (dataLength | stopBits |
                             parity));

    while(!SysCtlPeripheralReady(SYSCTL_PERIPH_UART0));
}

void UARTsend(uint32_t uartBase, const uint8_t *pui8Buffer, uint32_t
    ui32Count) {
    while(ui32Count--){
```

```
        ROM_UARTCharPut(uartBase, *pui8Buffer++);
    }
}
```

```
/*
 * crc.c
 *
 *  Created on: 05.02.2020
 *
 *      CRC-calculation function for the TOF>cam 635 taken from
 *      Installation_and_Operation_Manual_TOF>cam_635-V0.21
 *              URL: https://www.espros.com/downloads/02_Cameras_and_Modules/
    Installation_and_Operation_Manual_TOFCAM635.pdf
 *
 */

#include "crc.h"

uint32_t polynom = 0x04C11DB7;
uint32_t xorValue = 0x00000000;
uint32_t initValue = 0xFFFFFFFF;

uint32_t calcCrc32_32(const uint8_t *data, const uint32_t size){
    uint32_t crc = initValue;
    int i;
    for(i = 0; i < size; i++){
        crc = calcCrc32Uint32(crc, data[i]);
    }

        return crc ^ xorValue;
}

uint32_t calcCrc32Uint32(uint32_t crc, uint32_t data){
    int i;
    crc = crc ^ data;
    for(i=0; i<32; i++){
        if (crc & 0x80000000){
            crc = (crc << 1) ^ polynom;
        }
        else{
            crc = (crc << 1);
        }
    }
```

```
    return(crc);
}
```

A.2.2 Matlab-Code

ToF.m

```
%% Evaluation of ToF-Sensordata
%%
close all;
clear all;
clc;

%% Serial communication
% list available serial ports
serialportlist("available")'

%connect and configure com port
serialPort = serialport("COM6", 128000);
set(serialPort, 'DataBits', 8, 'StopBits', 1, 'ByteOrder', 'little-endian');
configureTerminator(serialPort, "LF");

serialPort.UserData = struct("Data",[],"Count",1);

writeline(serialPort, "s");
configureCallback(serialPort, "terminator", readToFdata);
```

readToFdata.m

```
function readToFdata(src, ~)
columns = 160;
rows = 60;

tic

%read half of the values at once
data = read(src, columns*rows, 'uint8');
count = src.UserData.Count;
src.UserData.Data((1+(count-1)*columns*rows):(count*columns*rows)) = data;

src.UserData.Count = src.UserData.Count + 1;

if src.UserData.Count > 2
```

```
    pixelData = src.UserData.Data;
    plotToFdata(pixelData);

    % turn off callback function
    %configureCallback(src, "off");

    src.UserData.Count = 1;
end
end
```

plotToFdata.m

```
function plotToFdata(pixelData)
    columns = 160;
    rows = 60;
    image = zeros(1, columns*rows);

    % merge to bytes two one 16-bit pixel data value
    for i = 1:columns*rows
        image(i) = bitor(pixelData(2*(i-1)+1), bitshift(pixelData(2*i), 8));

        % clear confidence bits, todo: store somewhere to evaluate
        image(i) = bitset(image(i), 16, 0);
        image(i) = bitset(image(i), 15, 0);
    end

    %ignore values above 3000
    for i = 1:length(image)
        if image(i) > 3000
            image(i) = 3000;
        end
    end

    y = repmat([0:columns-1], 1, rows); %horizontal axis
    z = zeros(1, columns); %vertical axis

    for i = 1:rows-1
        a = i*ones(1, columns);
        z = cat(2, z, a);
    end

    %configure colormap
    red = [1 0 0];
    yellow = [1 1 0];
    green = [0 1 0];
```

```
    blue = [0 0 1];
    black = [0 0 0];

    len = 100;
    redToYellow = [linspace(red(1), yellow(1), len)' linspace(red(2), yellow
        (2), len)' linspace(red(3), yellow(3), len)'];
    yellowToGreen = [linspace(yellow(1), green(1), len)' linspace(yellow(2),
        green(2), len)' linspace(yellow(3), green(3), len)'];
    greenToBlue = [linspace(green(1), blue(1), len)' linspace(green(2), blue
        (2), len)' linspace(green(3), blue(3), len)'];
    blueToBlack = [linspace(blue(1), black(1), len)' linspace(blue(2), black
        (2), len)' linspace(blue(3), black(3), len)'];

    redToGreen = cat(1, redToYellow, yellowToGreen);
    redToBlue = cat(1, redToGreen, greenToBlue);
    redToBlack = cat(1, redToBlue, blueToBlack);

    % plot data
    scatter3(image, y, z, [], image, 'filled');

    % configure plot
    colormap(redToBlack);
    colorbar;
    set(gca,'ydir','reverse', 'zdir', 'reverse'); %coordinates from top left
        to bottom right
    ylabel('Horizontal');
    zlabel('Vertical');
    xlabel('Distance');

    toc
end
```

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Dieses Blatt, mit der folgenden Erklärung, ist nach Fertigstellung der Abschlussarbeit durch den Studierenden auszufüllen und jeweils mit Originalunterschrift als letztes Blatt in das Prüfungsexemplar der Abschlussarbeit einzubinden.
Eine unrichtig abgegebene Erklärung kann -auch nachträglich- zur Ungültigkeit des Studienabschlusses führen.

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: Rostock

Vorname: Max

dass ich die vorliegende Bachelorarbeit bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Konzept und Entwicklung einer Mikrocontroller-basierten Time-of-Flight-Sensorik zur autonomen Navigation im Nahbereich

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der -bitte auswählen- ist erfolgt durch:

Hamburg	31.03.2020	[redacted]
Ort	Datum	Unterschrift im Original