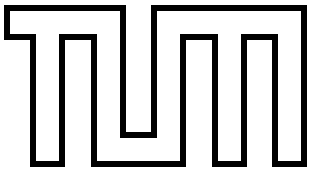


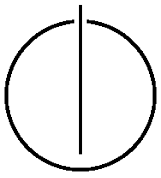
FAKULTÄT FÜR INFORMATIK

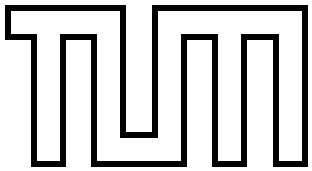
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Aktive Arbeitsraumüberwachung für die Mensch-Roboter-Kooperation mittels Tiefenbild-Kameras

Robert Maier





FAKULTÄT FÜR INFORMATIK

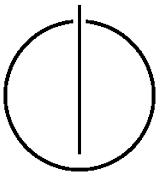
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Bachelorarbeit in Informatik

Aktive Arbeitsraumüberwachung für die Mensch-Roboter-Kooperation mittels Tiefenbild-Kameras

Dynamic Workspace-Surveillance in a Human-Robot Collaboration Scenario using Depth Cameras

Bearbeiter:	Robert Maier
Aufgabensteller:	Prof. Dr.-Ing. Alois Knoll
Betreuer:	Dipl.-Ing. Claus Lenz
Abgabedatum:	14. Oktober 2010



Ich versichere, dass ich diese Bachelorarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 14. Oktober 2010 Robert Maier

Danksagungen

Die Anfertigung dieser Arbeit wäre nicht ohne meinen Betreuer Claus Lenz vom *Lehrstuhl für Robotik und eingebettete Systeme* möglich gewesen, weshalb ich ihm recht herzlich für die ständige Unterstützung bei der Anfertigung dieser Bachelorarbeit danken möchte.

Für ihre Hilfe bei praktischen Problemen am JAHIR-Set-Up gilt mein Dank auch Helmuth Radrich vom *Lehrstuhl für Robotik und eingebettete Systeme* und Wolfgang Rösel vom *Institut für Werkzeugmaschinen und Betriebswissenschaften*.

Dies alles wäre auch nicht ohne die liebevolle Unterstützung meiner Freundin Katharina möglich gewesen, die jederzeit ein offenes Ohr für auftretende Probleme hatte und mich auf jede mögliche Art und Weise unterstützte. Vielen Dank. Außerdem möchte ich mich bei ihrer Familie bedanken, dass sie sich so viel Zeit für das Korrekturlesen dieser Arbeit genommen haben und mir somit wertvolles Feedback gegeben haben.

Zuletzt geht eine großer Dank an meine Familie, die es mir selbst in den arbeitsreichen Zeiten jederzeit erlaubt hat, dass ich mir meinen Traum des Informatik-Studiums ermöglichen kann.

Zusammenfassung

Industrieroboter und Menschen werden bisher strikt getrennt eingesetzt, da der Arbeitsraum der Roboter aufgrund fehlender Sensorüberwachung meist statisch ist und ein direkter Kontakt des Roboters mit dem Menschen schwerwiegende gesundheitliche Folgen haben kann. Durch eine Kooperation von Menschen und Industrierobotern, die den menschlichen Arbeiter unterstützen, kann jeder seine Fähigkeiten zur Bewältigung einer Aufgabe gewinnbringend einsetzen. In einem solchen Kooperations-Szenario liegt ein besonderer Augenmerk auf der Gewährleistung der Sicherheit des Menschen.

In der hybriden Montagestation *JAHIR (Joint Action for Humans and Industrial Robots)* erfolgt eine direkte Kooperation eines menschlichen Arbeiters mit einem Industrieroboter. In einer verteilten, modularen und echtzeitfähigen Systemarchitektur sind Komponenten zur Verarbeitung von Sensordaten integriert, mit deren Hilfe ein Algorithmus zur Kollisionsvermeidung die Sicherheit des Arbeiters garantiert.

Im Rahmen dieser Thesis wird eine Komponente für JAHIR zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kameras entwickelt, die als zusätzliche Maßnahme zur Erhöhung der Sicherheit des Menschen dient. Damit können vorab unbekannte Objekte im gemeinsamen Arbeitsraum zur Laufzeit wahrgenommen werden, so dass eine entsprechende Reaktion des Roboters erfolgen kann.

Der grundlegende Aufbau von JAHIR wird für eine Einordnung der erstellten Komponente in das JAHIR-Gesamtsystem erläutert. Zum Ermitteln von dreidimensionalen Informationen über den Arbeitsraum wird eine geeignete Tiefenbild-Kamera ausgewählt, montiert und mit Hilfe einer CCD-Kamera kalibriert. Mit Hilfe des Umgebungsmodells des Roboters findet eine Bereinigung der Sensordaten statt. Zur Veranschaulichung wird die interne Szenenrepräsentation in einer Visualisierung angezeigt. Ein Veröffentlichen der verarbeiteten Sensordaten über das verteilte System bewirkt Reaktionen des Algorithmus zur Kollisionsvermeidung auf dynamische Objekte im Arbeitsraum. Im Verlauf dieser Arbeit werden zu jedem Sachverhalt sowohl theoretische Grundlagen als auch praktische Umsetzung beschrieben.

Den Abschluss dieser Arbeit stellen eine Evaluation der erstellten Komponente sowie ein Ausblick auf mögliche Erweiterungen und Verbesserungsmöglichkeiten dar.

Abstract

In current applications industrial robots and humans are strictly separated due to a lack of sensor devices that surveil the robot workspace. Because of that collisions of humans with a robot may have severe consequences for the human worker.

In a human-robot collaboration scenario an industrial robot assists the human worker to accomplish a given task in a way that each member contributes his given capabilities. A very important aspect is to guarantee the safety of the human.

JAHIR (Joint Action for Humans and Industrial Robots) is a hybride assembly station enabling a direct collaboration between humans and an industrial robot. A modular distributed system architecture integrates components that process sensor information. In order to ensure the worker's safety, these components provide the integrated collision avoidance algorithm with that processed data.

The purpose of this thesis is to develop a component for JAHIR that provides dynamic workspace surveillance using depth cameras. This additional measure helps to improve the safety of the human worker. Furthermore, a detection of unknown obstacles in the shared workspace and an appropriate reaction of the robot is possible.

First, an overview of the basic structure of JAHIR is given. To get three-dimensional data of the environment an appropriate depth camera is selected, installed and calibrated using a CCD-camera. The sensor information is reduced by testing it against the internal environment model. A visualization component illustrates the internal scene representation. Reactions of the collision avoidance algorithm on dynamic objects in the workspace are achieved by publishing the sensor data over the distributed network. This thesis presents both, the theoretical basics and their practical implementation.

Finally, a conclusion with an evaluation of the developed component and an outlook on possible extensions and enhancements is given.

Inhaltsverzeichnis

1 Einleitung und Motivation

Ende des Jahres 2008 waren weltweit mehr als eine Million Industrieroboter im Einsatz, von denen jedoch die meisten aus Sicherheitsgründen in einem zeitlich und räumlich vom Menschen getrennten Arbeitsraum eingesetzt sind. Dabei werden die Roboter aus der Ferne vom Menschen gesteuert oder führen offline programmierte statische Aufgaben durch. Da in den meisten gegenwärtigen Anwendungen von Industrierobotern der Arbeitsraum aufgrund fehlender Sensorüberwachung statisch ist, können bei nicht vorab geplanten Veränderungen der Roboterumgebung Kollisionen auftreten. Solch eine Umgebungsveränderung kann beispielsweise ein Mensch sein, der in die Arbeitszelle eintritt und die Traktorie des Roboters kreuzt, was schwerwiegende Folgen für den Menschen nach sich ziehen kann.

In den Produktionslinien der Automobilindustrie werden beispielsweise die Montageschritte ausschließlich von Robotern ohne manuelle Eingriffe durch menschliche Arbeiter durchgeführt. Auf der anderen Seite wiederum fehlt bei der manuellen Fließbandproduktion eine unterstützende Integration von Robotern an der Seite von Menschen. Durch diese bisherige strikte Trennung bleibt das Potential der gemeinsamen Teamarbeit zwischen Mensch und Roboter unberücksichtigt, in der jeder seine Fähigkeiten zur Bewältigung einer Aufgabe gewinnbringend einsetzt.

Im Gegensatz zu diesen herkömmlichen Szenarien mit räumlich und zeitlich strikt von Menschen getrennten Robotern, erfolgt im Projekt *Joint Action for Humans and Industrial Robots (JAHIR)* [1] [2] eine direkte Kooperation von Menschen mit Industrierobotern. Insbesondere werden dabei Industrieroboter als unterstützende Maßnahmen in bisher vom Menschen dominierte Bereiche von Produktionsprozessen integriert, was zu einer Erhöhung der Produktivität sowie einer besseren Ergonomie für den menschlichen Arbeiter führt.

JAHIR ist im Rahmen des Exzellenzclusters *CoTeSys (Cognition for Technical Systems)* [3] angesiedelt, der sich mit der Integration von kognitiven Fähigkeiten in technische Systeme (wie Roboter und Fabriken) befasst. Kognitive Systeme sind informationsverarbeitende Systeme, die mit Sensoren und Aktuatoren ausgestattet und in physische Systeme integriert sind, die mit der Außerwelt interagieren. Das Hinzufügen von kognitiven Fähigkeiten führt zu stabileren, flexibleren, adaptiveren und performanteren Systemen. Ein kognitives System verfolgt ein langfristiges Ziel, wie die Herstellung eines Produkts. Dabei erfasst es seine Umgebung mit Sensoren, verarbeitet diese Sensordaten und reagiert wissensbasiert auf geeignete Weise auf Situationen, wodurch die Wahrnehmungs-Kette „Perception“, „Cognition“, „Action“ geschlossen wird. Die *Kognitive Fabrik* im Rahmen des CoTeSys-Clusters stellt einen Demonstrator eines solchen kognitiven Systems dar und hat die Unterstützung einer effektiveren Herstellung von Produkten zum Ziel. Die Kognitive Fabrik besteht aus einer automatischen Montagestation *Cognitive Machine Shop (CogMaSh)*, einer rein von Menschen dominierten Montage *Adaptive Cognitive Interaction in Production Environments (ACIPE)* und einer hybriden Montage JAHIR. [4] [5] [6]

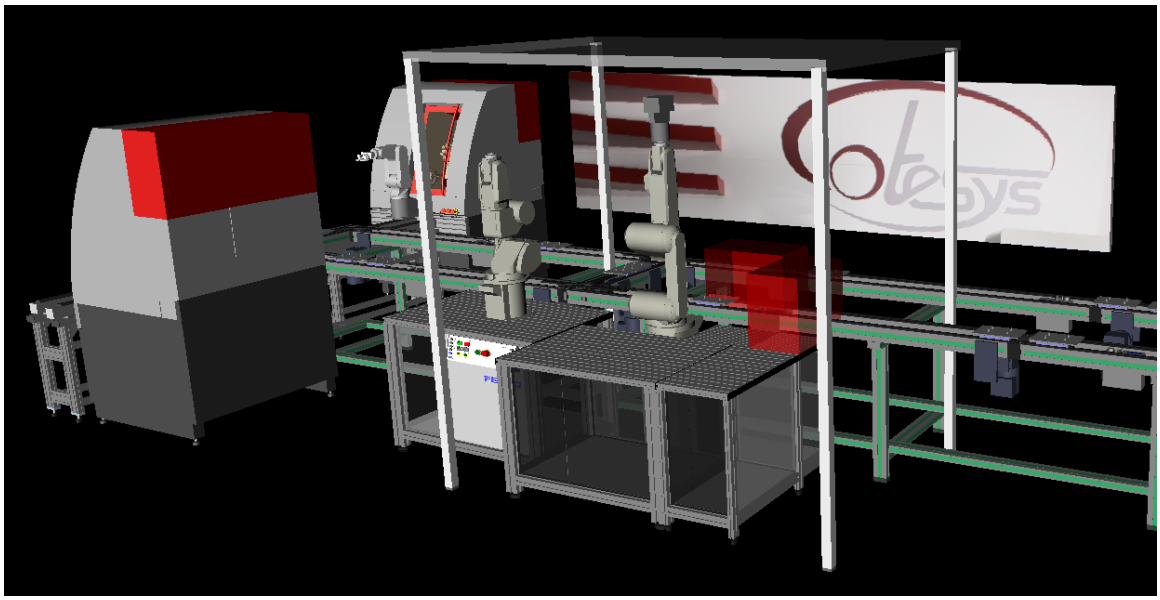


Abbildung 1.1: 3D-Modell der Stationen *CogMaSh* und *JAHIR* der Kognitiven Fabrik.

Aufgrund der direkten Kooperation von Menschen mit einem Roboter in JAHIR ist es zwingend notwendig, dem Roboter unbekannte Objekte im gemeinsamen Arbeitsraum zu detektieren und eine entsprechende Reaktion des Roboters zu bewirken. Dabei liegt ein besonderer Augenmerk auf der Gewährleistung der Sicherheit des menschlichen Arbeiters.

Das Ziel der vorliegenden Arbeit ist die Entwicklung und Integration einer zusätzlichen Komponente in das JAHIR-Projekt, mit deren Hilfe der gemeinsame Arbeitsraum von menschlichem Arbeiter und Roboter mittels einer Tiefenbild-Kamera überwacht wird. Dabei können mit Hilfe der dreidimensionalen Tiefendaten der Kamera unbekannte Objekte wahrgenommen werden, so dass anschließend Maßnahmen für eine schnelle Kollisionserkennung und -vermeidung eingeleitet werden können. Dies soll eine signifikante Erhöhung der Sicherheit des menschlichen Arbeiters gewährleisten.

Diese Arbeit ist in sechs Kapitel gegliedert: Kapitel 2 gibt einen Überblick über JAHIR und geht auf die Grundlagen der für die Realisierung der Komponente verwendeten Konzepte ein.

Kapitel 3 widmet sich Theorie und Praxis der Kalibrierung einer PMD-Kamera.

Kapitel 4 geht auf die praktische Implementierung einer Komponente zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kamera ein.

In Kapitel 5 findet eine Evaluation der erstellten Komponente hinsichtlich Geschwindigkeit und Genauigkeit statt.

Kapitel 6 stellt einen Abschluss dieser Arbeit dar und gibt nach einer Zusammenfassung einen Ausblick auf mögliche Erweiterungen und Verbesserungsmöglichkeiten des gewählten Ansatzes.

2 Grundlagen

Vor der Implementierung einer Komponente zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kameras für JAHIR werden die dieser Arbeit zugrunde liegenden Konzepte dargelegt.

Zuerst wird das erarbeitete Gesamtkonzept der zu erstellenden Komponente erläutert, bevor ein ausführlicher Überblick über die JAHIR-Plattform gegeben wird, der für eine Einordnung der zu Komponente in das Gesamtsystem notwendig ist. Dem schließen sich eine Erläuterung der unterschiedlichen Verfahren zum Gewinnen von Tiefeninformationen sowie die Auswahl eines geeigneten Verfahrens an. Das Grundprinzip von Bounding-Box-Tests bei der Verarbeitung der Tiefenbild-Informationen wird ebenso behandelt wie die theoretischen Grundlagen von Clustering-Verfahren. Ferner wird auf vergleichbare Ansätze sowie deren Vor- und Nachteile eingegangen.

2.1 Gesamtkonzept

Abbildung 2.1: Gesamtkonzept einer Komponente zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kameras.

Für die Entwicklung der zu erstellenden Komponente für das JAHIR-Projekt wurde das in Abbildung 2.1 dargestellte Gesamtkonzept entwickelt, das die in dieser Arbeit abzudeckenden Teilbereiche festlegt. Für eine genauere Beschreibung des Gesamtkonzepts werden die einzelnen Teilbereiche wie folgt näher beschrieben:

Umgebungsmodell. Das JAHIR-Gesamtsystem und somit auch der Roboter haben eine interne Repräsentation der Roboter-Umgebung. Dabei unterscheidet man zwischen der statischen Umgebung, zu der die unbeweglichen Teile der Arbeitszelle wie der Arbeitstisch und der Roboter zählen, und der dynamischen Umgebung, die vorab unbekannte Objekte enthält. Eine umfassende Kenntnis der Umgebung ist für eine Überwachung des Arbeitsraums des Roboters unerlässlich.

Tiefenbild-Kamera. Um beliebige Objekte und Hindernisse im Arbeitsraum erkennen zu können, muss die Umgebung mit Hilfe einer Tiefenbild-Kamera überwacht werden.

In diesem Zusammenhang sind eine genaue Kamera-Kalibrierung sowie eine echtzeitfähige und zuverlässige Kommunikation mit der Tiefenbild-Kamera zum Laden der dreidimensionalen Tiefeninformationen notwendig.

Verarbeiten der Kameradaten. Nach dem Laden der Sensordaten sollen diese Informationen mit Hilfe des Wissens über die Umgebung bereinigt werden, so dass die zu vorab bekannten Objekten (z.B. Arbeitstisch) gehörenden Daten bereits ausgeschlossen werden. Auch Objekte, die sich nicht in der Reichweite des Roboters befinden, können entfernt werden. Um aus den einzelnen Punkten der Punktwolke einer Tiefenbild-Kamera zusammenhängende Objekte zu ermitteln, können mit Hilfe der Clusteranalyse Punkthäufungen berechnet und zu größeren Objekten zusammengefasst werden.

Publishen. Da im JAHIR-Projekt die einzelnen Komponenten über ein verteiltes System miteinander kommunizieren, werden in einem weiteren Schritt die gefundenen Cluster über das verteilte System an andere Komponenten weitergereicht. Dadurch können auch wichtige sicherheitskritische Algorithmen in anderen Komponenten (z.B. Kollisionsvermeidung) auf die gefundenen Objekte geeignet reagieren.

Visualisierung. Zur grafischen Veranschaulichung sollen in einer auf dem Umgebungsmodell basierenden Visualisierung die dreidimensionalen Daten der Tiefenbild-Kamera sowie die gefundenen Hindernisse angezeigt werden.

Im weiteren Verlauf dieser Arbeit dient dieses Gesamtkonzept als Basis für die Implementierung der Komponente und ihre theoretischen Grundlagen.

2.2 JAHIR: Joint Action for Humans and Industrial Robots

JAHIR ist eine hybride Montagestation, in der eine Kooperation zwischen einem menschlichen Arbeiter und einem Industrieroboter stattfindet.

Der installierte Industrieroboter kann neue hybride Montageaufgaben, die von Mensch und Roboter kooperativ ausgeführt werden sollen, durch sprachbasierte Instruktionen des Arbeiters erlernen. Dabei können sowohl menschliche Fähigkeiten als auch die Fähigkeiten des Roboters in die erlernte Aufgabe integriert werden. Für eine intuitive Interaktion mit dem System gibt es verschiedene Eingabemöglichkeiten.

Eine effiziente Zusammenarbeit von Mensch und Maschine wird durch ein antizipatives Verhalten des Robotersystems erreicht, das zur Garantierung der Sicherheit des menschlichen Arbeiters mit reaktivem Verhalten kombiniert ist. Als reaktive Komponente ist ein Kollisionsvermeidungs-Algorithmus integriert, der die Daten von unterschiedlichen Sensorgeräten, die die Arbeitsraumüberwachung in diesem Szenario gewährleisten, verarbeitet.

Im Folgenden wird der Aufbau von JAHIR in physikalischer Hinsicht und im Hinblick auf die verwendete Softwarearchitektur beschrieben. [5] [6]

Abbildung 2.2: Mensch-Roboter-Kooperation in JAHIR. [6]

2.2.1 Aufbau des Set-Up

Die JAHIR-Arbeitszelle ist in semantischer und in räumlicher Hinsicht in das Gesamtszenario der Kognitiven Fabrik zwischen voll automatisierter und manueller Montage eingebettet. Das JAHIR-Set-Up ist in Abbildung 2.3 zu sehen, wobei die nachfolgend aufgeführten Bestandteile für eine eindeutige Zuordnung mit Nummern versehen sind. Um einen schnellen und flexiblen Austausch von benötigten und verarbeiteten Teilen zu gewährleisten, ist JAHIR über ein Fließband (1) direkt an die beiden anderen Stationen *CogMaSh* und *ACIPE* angebunden.

JAHIR selbst besteht aus einer 0,7 m x 1,4 m großen Werkbank (2), auf die der davor stehende Mensch und der hinter der Werkbank platzierte Roboter gemeinsam zugreifen können. Als Roboter ist ein Industrieroboter *Mitsubishi RV-6SL* (3) installiert, der Objekte mit einem maximalen Gewicht von sechs Kilogramm heben kann, sechs Freiheitsgrade und einen Bewegungsradius von 0,902 m besitzt. Dieser Roboter dient als Assistent für den menschlichen Arbeiter und kann auch auf das Fließband zugreifen.

Da der Werkzeugs-Mittelpunkt mit einem 6-Achsen-Kraft-Momenten-Sensor und einer Werkzeug-Wechseleinheit ausgestattet ist, kann zwischen unterschiedlichen Arten von Endeffektoren gewechselt werden. Der Roboter kann dabei das für den anstehenden Arbeitsschritt geeignete Werkzeug autonom wählen. Die verschiedenen Endeffektoren wie ein Zwei-Finger-Parallelgreifer oder eine Bohrmaschine sind an zwei Austauschstationen (4) mit jeweils drei Werkzeug-Ports abgelegt.

Oberhalb der Werkbank ist ein Tisch-Projektor (5) montiert, der Informationen wie Montageanweisungen oder Interaktionsfelder in das Sichtfeld des Arbeiters auf den Arbeitstisch projiziert.

Zur Wahrnehmung der Umgebung und zur Überwachung des gemeinsamen Arbeitsraums sind verschiedene Arten von Sensor- und Eingabegeräten im Set-Up installiert, so

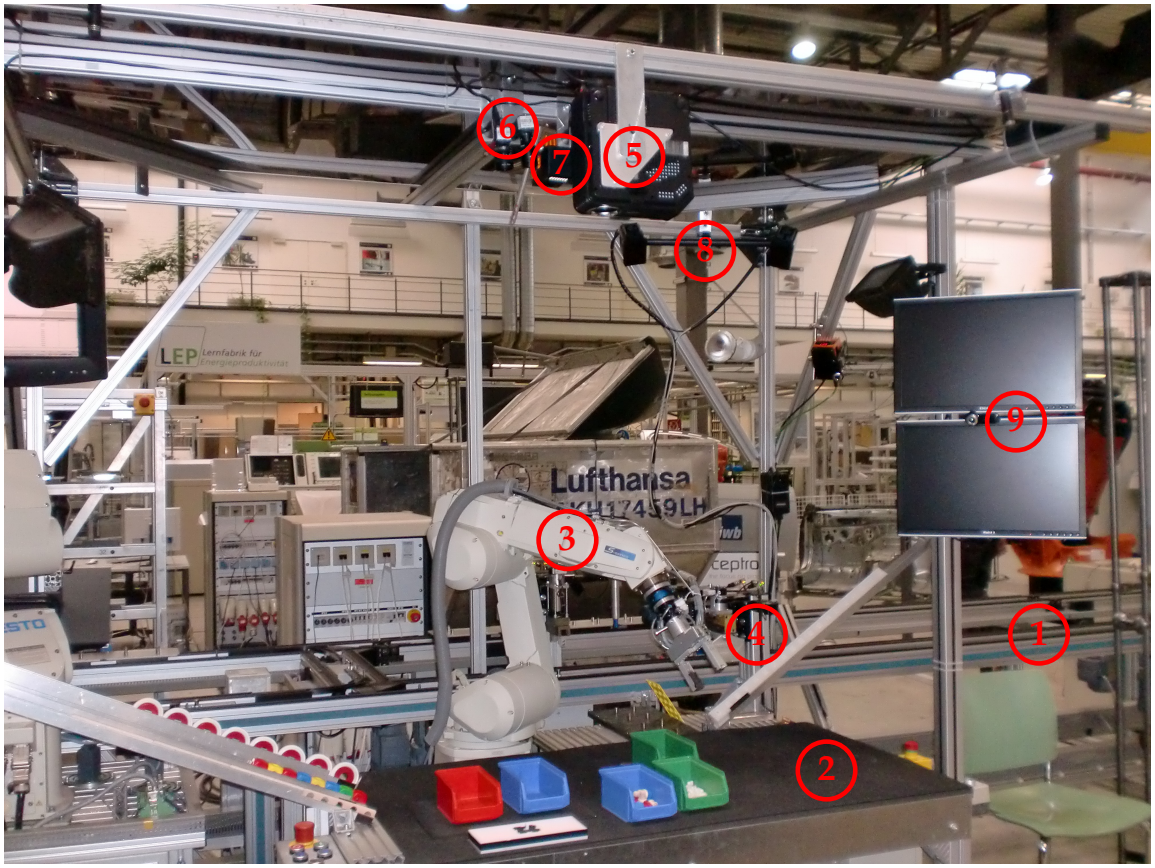


Abbildung 2.3: Aufbau des JAHIR-Setup.

dass das Verhalten des Arbeiters antizipiert oder erkannt werden kann und geeignete Reaktionen auf gefährliche oder unvorhersehbare Situationen eingeleitet werden können.

Über der Werkbank sind an einem Käfig, der den Arbeitsraum umgibt, nach unten ausgerichtete Kameras (6) montiert, die eine globale Sicht auf den Arbeitsraum bieten. Diese Kameras dienen zur Überwachung von Aktionen auf dem Arbeitstisch und zur Lokalisierung von auf dem Tisch liegenden Objekten. Die Daten der Kameras können mit Hilfe einer zusätzlichen PMD-Kamera (7) in Echtzeit mit Tiefeninformationen angereichert werden, so dass komplexere Aufgaben der Bildsegmentierung wie bildbasiertes Hand-Tracking oder Objekterkennung ermöglicht werden.

Eine dieser Kameras ist oben in der Mitte des Arbeitstisches befestigt und dient zur Gewinnung von visuellen Informationen, mit deren Hilfe bekannte Objekte und deren Position innerhalb der Arbeitsumgebung ermittelt werden können. Zwei zueinander kalibrierte Kameras sind von links oben bzw. rechts oben in den Arbeitsraum gerichtet, wodurch beispielsweise eine nicht-invasive Positionsermittlung in Welt-Koordinaten ermöglicht wird. Eine weitere Kamera ist schließlich von vorne auf den Kopf des Arbeiters gerichtet und dient zur Wahrnehmung des Aufmerksamkeitsfokus des Arbeiters. Diese Kamera kann jedoch auch zur Erkennung von verschiedenen Arbeitern und einer automatischen Zuweisung von Profilen benutzt werden, so dass ein Neuling intensivere Unterstützung im

Umgang mit dem System erhält oder ein Linkshänder automatisch eine andere Übergabeposition verwenden kann.

Der Aufmerksamkeitsbereich des Arbeiters kann auch über eine auf dem Kopf getragene Tracking-Brille erfasst werden, die Informationen über die Blickrichtung des Arbeiters liefert.

Ein P5-Datenhandschuh zur Erkennung von Gesten liefert Koordinaten der Hand des Arbeiters sowie Informationen über die Beugungen der Finger, wobei dieses Verfahren die Nachteile einer geringen Reichweite und einer geringen Genauigkeit aufweist. Ein größerer Arbeitsbereich und eine höhere Genauigkeit für eine zuverlässigere Ortung der Handposition, die bespielsweise bei der Übergabe von Teilen besonders wichtig ist, kann mit Hilfe eines Stereo-Infrarot-basierten Marker-Tracking-Systems (8) erreicht werden, wobei die Messung mit Infrarot sehr empfindlich gegenüber Sonnenlicht ist.

Die Standposition des Arbeiters vor dem Tisch kann mit gewichtsempfindlichen Matten und einem unter dem Tisch angebrachten Laserscanner ermittelt werden.

Mit einem Desktop-Mikrofon (9) können Äußerungen des Arbeiters aufgenommen und zur Spracheingabe verwendet werden, so dass ein natürlicher Umgang mit dem System gewährleistet ist. [5] [6]

2.2.2 Systemarchitektur

Das JAHIR-Projekt besteht aus einer verteilten echtzeitfähigen Softwarearchitektur, die in [5] und [6] beschrieben ist. Sie ist in mehrere Module aufgeteilt, die in die Bereiche Wahrnehmung (Sensoren), Kognition (Wissensdatenbank, Verarbeitungseinheiten und Zustandsmaschine) und Aktuator-Module (Robotersteuerung und Benutzerschnittstellen) eingeteilt werden können. Zu diesen Modulen gehört auch Software zur Kontrolle der standardisierten industriellen Hardwarekomponenten (Sensoren und Aktuatoren) in der JAHIR-Arbeitszelle, die im vorhergehenden Abschnitt dargestellt wurden. Ein Zusammenschalten der Module erfolgt mit Hilfe des Knowledge-based System-Controllers. Mit diesem modularen Aufbau des Systems und der damit verbundenen leichten Integration von unterschiedlichen Komponenten geht eine signifikante Reduzierung der Komplexität des Systems einher. Diese Modularität erlaubt eine problemlose und flexible Integration von Erweiterungen, die häufig Ergebnisse von anderen Projekten sind, in den JAHIR-Demonstrator, wodurch auch die Fähigkeiten des Gesamtsystems wachsen.

Da die online-Bewegungskontrolle des Roboters in der hybriden Arbeitszelle Befehle im 7 ms-Takt benötigt und einige Module (z.B. für Bildverarbeitung) hohen Berechnungsaufwand benötigen, wurden die Systemmodule auf mehrere Computer verteilt. Die Kommunikation zwischen den einzelnen Modulen wurde mit der Middleware *Zeroc Ice (Internet Communication Engine)* [7] für verteilte Verarbeitung und asynchronen Nachrichtenaustausch unter Berücksichtigung von Topics of interest realisiert.

Das System hat auch Zugriff auf eine integrierte Echtzeit-Datenbank, die auf einem Ringpuffer basiert und mehreren Modulen einen nicht blockierenden Zugriff auf die gleichen Daten durch Shared Memory ermöglicht. Einerseits dient diese Wissensdatenbank zur lokalen Pufferung von Sensorinformationen und zur Aufzeichnung von Sensordaten in Echtzeit. Eine solche Aufzeichnung von großen Mengen von Sensordaten hat den Vorteil, dass die Daten für eine erneute Wiedergabe, zur Simulation oder für eine nachträgliche Analyse wiederverwendet werden können. Außerdem können die aufgezeichneten Daten

zur Evaluation von Fremdsystemen in einer fremden Umgebung oder für Leistungsvergleiche von verschiedenen Systemen unter gleichen Bedingungen herangezogen werden. Andererseits können in der Wissensdatenbank Informationen über den gegenwärtigen Kontext gespeichert werden. Dazu zählen ein Montageplan der aktuellen Aufgabe, die verfügbaren Werkzeuge und die benutzbaren Objekte auf dem Arbeitstisch inklusive deren Eigenschaften (Farbe, Position, etc.). Mit dem Wissen über die Aufgabe und Umgebung in Kombination mit gesammelten Sensordaten kann das System den aktuellen Kontext erkennen, mögliche nächste Schritte antizipieren und bereits vorab die nächsten Schritte einleiten.

Zentrale Bestandteile der Systemarchitektur sind die in Abbildung 2.4 auf der linken Seite dargestellten Verarbeitungs-Module, auf die nachfolgend näher eingegangen wird.

Multimodale Ein- und Ausgabe. Um die Kommunikation mit dem System für den Arbeiter intuitiver und ergonomischer zu machen, besitzt es verschiedene Eingabemöglichkeiten. Damit können auch Arbeiter ohne Erfahrung im Programmieren von Robotern dem System neue Aufgaben online und in Echtzeit beibringen, während gewöhnlicherweise die meist zeitintensive Programmierung von Produktionssequenzen nur von erfahrenem und somit teurem Personal möglich ist.

- Eine Sprecher-unabhängige Spracherkennung in Deutsch oder Englisch ermöglicht die Eingabe gesprochener Anweisungen per Desktop-Mikrofon für eine natürlichere und intuitivere Kommunikation zwischen Mensch und Roboter. Zur Verbesserung der Erkennungsrate ist die Sprachgrammatik auf den aktuellen Aufgabenkontext reduziert, wobei jedes Modul seine eigene Grammatik hat, die dem wissensbasierten System-Controller zur Verfügung steht.
- Eine Tracking-Brille ermittelt den Aufmerksamkeitsfokus des Arbeiters im Welt-Koordinatensystem (KoSy), indem sie mit zwei Kameras die Pupillen verfolgt und mit einer weiteren Kamera die Szene beobachtet. Die Tracking-Brille erlaubt eine höhere Präzision als entferntes Augentracking, jedoch ist sie invasiver und nicht so komfortabel.
- Manuelle Eingaben können über Hardware-Tasten sowie über von oben auf die Werkbank projizierte sensitive Felder (Soft-Buttons) getätigt werden. Diese Soft-Buttons können in Inhalt und Position zur Laufzeit angepasst werden, weshalb sie eine sehr ergonomische und flexible Interaktionsmethode darstellen. Die Position der Hand des Arbeiters wird mit Hilfe von Segmentierung ermittelt. In Kombination mit der Position der Soft-Buttons erzeugt ein weiteres Modul Events, wenn eine Hand für einen bestimmten Zeitraum über einem senstiven Feld bleibt.
- Um dem Arbeiter ein Audio-Feedback zu geben, generiert das System mittels Sprachsynthese (TTS) Informationen über die nächsten Schritte im Herstellungsprozess oder über Fehler. Als visuelles Feedback werden Montageanweisungen und Soft-Buttons direkt auf die Werkbank projiziert.

Gripper Manager. Der Gripper Manager stellt die Schnittstelle zum Zugriff auf die verschiedenen Werkzeuge des Roboters dar. Der Roboter weiß anhand eines kodierten Musters zur Identifikation jederzeit, welches Werkzeug er in der Hand hält und passt

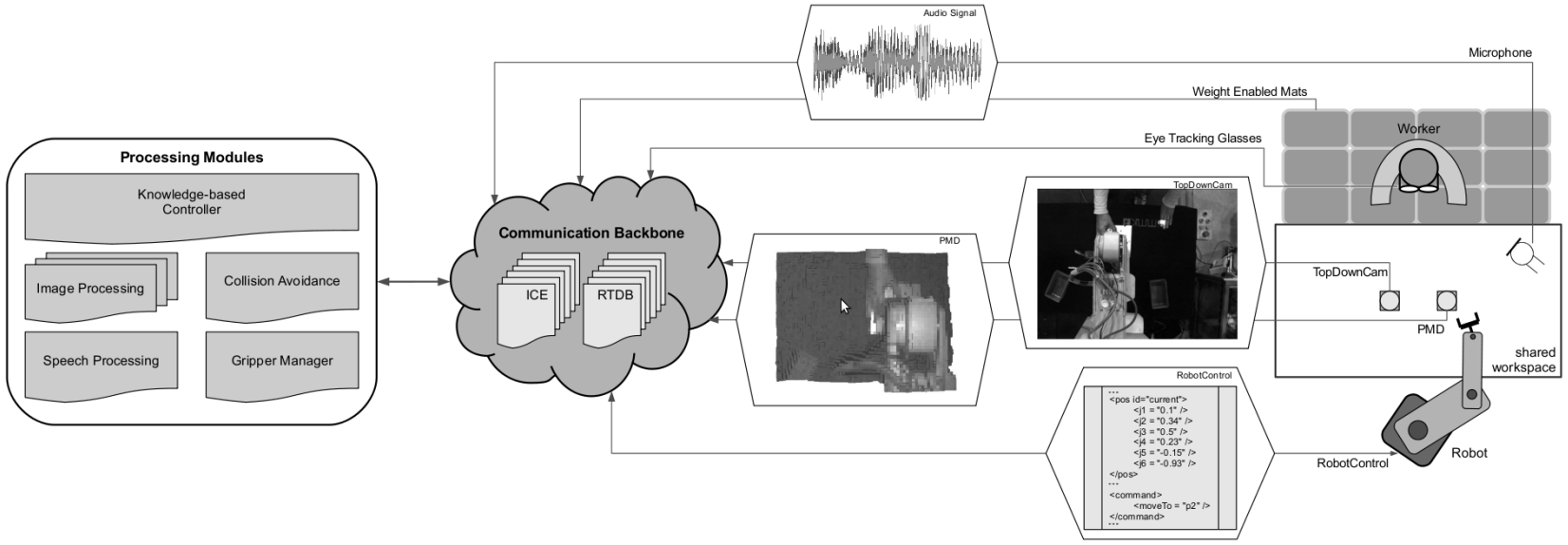


Abbildung 2.4: Schematischer Überblick über die JAHIR-Systemarchitektur bestehend aus Verarbeitungs-Modulen, Kommunikations-Backbone, Sensoren/Aktuatoren und gemeinsamen Arbeitsraum. [6]

automatisch die kinematischen Parameter (Werkzeuge-Länge und -Mittelpunkt) an. Auch die Belegung der Werkzeug-Ports ist bekannt. Über den mit dem Gripper Manager verbundenen Knowledge-based System Controller kann autonom ein Werkzeugwechsel durchgeführt werden, wenn eine bestimmte Fähigkeit angefordert wird.

Überwachung des gemeinsamen Arbeitsraums. Bei einer Kooperation von Menschen und Robotern in einem gemeinsamen Arbeitsraum können sicherheitskritische Situationen auftreten, die insbesondere bei Standard-Industrierobotern fatale Folgen für den Menschen haben können. Zur Erhöhung der Sicherheit wird in JAHIR eine redundante Komponente zur Überwachung des gemeinsamen Arbeitsraums verwendet.

Zur Erkennung des Menschen können verschiedene Arten von Sensoren verwendet werden, deren Informationen von zugehörigen Modulen ausgewertet und aufbereitet werden. Eine Matrix von gewichtssensiblen Matten auf dem Boden kann Kontakt mit den Füßen des Menschen erkennen, um eine grobe Position des Arbeiters zu bekommen. Die Erkennung und Lokalisierung von Armen und Händen des Arbeiters erfolgt mit Hilfe einer PMD-Kamera.

Collision Avoidance Controller. In einem gemeinsamen Arbeitsraum müssen Kollisionen sowohl für statische Objekte (z.B. Werkbank) als auch für dynamische (z.B. bewegliche Hindernisse) vermieden werden. Im Collision Avoidance Controller, der ein Bestandteil des Robot-Controller ist, können Kollisionen auf reaktive Weise mit Hilfe eines dynamischen 3D-Modells vermieden werden. Somit ist dieses Modul die zentrale Komponente zur Garantierung der Sicherheit des Arbeiters. Die interne 3D-Repräsentation der Roboter-Umgebung passt sich flexibel an von Eingabesensoren ermittelte und weitergeleitete Änderungen der Umgebung an. Dies erlaubt ein einfaches Hinzufügen neuer Sensormodule ohne Änderungen am Robot-Controller, wie in Abbildung 2.5 zu erkennen ist.

Bei einer konkreten Gefahr für den Menschen wie einem drohenden direkten Kontakt mit dem Roboter, kann dieser über einen direkten Anschluss mit dem Notaus gestoppt werden. Bei einer möglichen Kollision wird die geplante Bewegung des Roboters mit einer Vermeidungsbewegung so umgeplant, dass er seine Aufgabe ohne Auftreten einer Kollision ausführen kann. Nähere Details zur Kollisionsvermeidung in JAHIR sind in [8] zu finden.

Knowledge-based System Controller. Der Knowledge-based System Controller kümmert sich um die komplexen Prozesse innerhalb einer hybriden Montageaufgabe, wobei eine Modifikation seines internen Wissens (Fakten und Regeln) auch zur Laufzeit möglich ist.

Fakten sind Informationen über die auftretenden Ereignisse, die Umgebung und die Fähigkeiten der verbundenen Module und werden bei Änderungen stets auf Übereinstimmung mit vorhandenen Regeln geprüft. Regeln, die auch komplex sein können, beschreiben den Arbeitsfluss innerhalb des Systems. Eine detaillierte Beschreibung des Knowledge-based System Controllers findet man in [6].

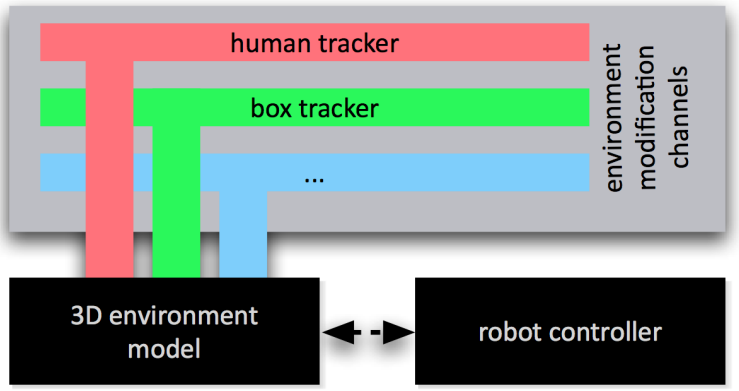


Abbildung 2.5: Den Arbeitsraum überwachende Module nehmen Veränderungen der Umgebung wahr und leiten sie an das 3D-Umgebungsmodell des Kollisionsvermeidungs-Moduls im Robot-Controller weiter. [6]

2.3 Verfahren zum Gewinnen von Tiefeninformationen

Um bekannte und unbekannte Objekte im Arbeitsraum wahrnehmen und verarbeiten zu können, ist eine genaue dreidimensionale Beschreibung dieser Objekte im Raum unerlässlich. Während bei herkömmlichen CCD-Kameras sämtliche Tiefeninformationen verloren gehen, kann mit speziellen Verfahren die „dritte Dimension“ der Umgebung zurückgewonnen werden.

Nachfolgend werden verschiedene Verfahren zum Gewinnen von Tiefeninformationen sowie deren Vor- und Nachteile ausführlich dargestellt.

2.3.1 Stereo Vision

Ein erstes solches Verfahren stellt das Prinzip der Stereo Vision dar, das auch als Triangulation bekannt ist. Darunter versteht man den in Abbildung 2.6 dargestellten Prozess der visuellen Wahrnehmung von Tiefe anhand von Bildern von mindestens zwei zueinander versetzten Kameras. Solche Triangulationsverfahren verwenden in der Regel zwei oder mehrere zueinander kalibrierte Standard-CCD-Kameras und entsprechen dadurch prinzipiell der Rekonstruktion von Tiefeninformationen mit dem menschlichen Auge. Stereo Vision basiert in der Praxis darauf, dass charakteristische Features von Objekten aus der Szene extrahiert werden und deren Position innerhalb der Bilder der verschiedenen Kameras verglichen wird.

Vorteilhaft an diesem Verfahren ist die günstige Verfügbarkeit von Standard-CCD-Kameras sowie die Fülle von Informationen, die aus deren detailreichen Bildern extrahiert werden können.

Bei kontrastlosen Szenen oder bei einfarbigen Flächen ohne Muster ist jedoch ein Detektieren der Features und somit ein Matching der zu kombinierenden Bilder kaum möglich. Ein solches Matching kann zusätzlich dadurch erschwert werden, dass Features nur in einem der Bilder zu sehen sind, wobei man auch von Okklusion spricht. Die Extraktion von 3D-Informationen aus den gemessenen Daten benötigt überdies einen sehr hohen Rechenaufwand. Bedingt durch diesen hohen Rechenaufwand ist das Ermitteln von Tiefeninfor-

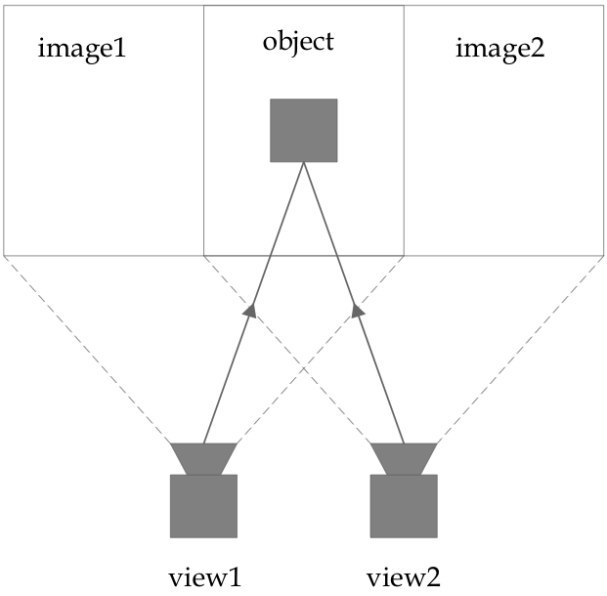


Abbildung 2.6: Funktionsweise eines Stereo Vision Systems [9].

mationen in für Echtzeit-Anwendungen akzeptablen Frameraten bisher nicht möglich. Für eine hohe Tiefenauflösung ist überdies eine große Triangulationsbasis notwendig, was nur mit großen Kamera-Systemen erreicht werden kann. Durch die damit verbundenen Hardwareanforderungen sind solche Systeme in ihrer Gesamtheit auch sehr teuer. Aufgrund der überwiegenden Nachteile ist somit von einer Anwendung von Stereo Vision in diesem Applikationsszenario abzusehen. [9] [10]

2.3.2 3D-Laserscanner

Im Vergleich zu Stereo Vision lassen sich mit dem Time-of-Flight-Prinzip (ToF) praktikablere Methoden zur Abstandsmessung durchführen. Zur Veranschaulichung können ToF-Messungen als optisches Analogon zum Ultraschall-System von Fledermäusen gesehen werden.

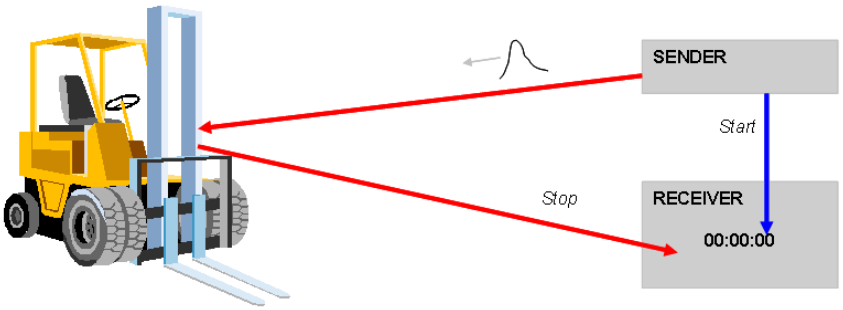


Abbildung 2.7: Funktionsweise von Time-of-Flight-Kameras [11].

Wie in Abbildung 2.7 zu sehen bestehen ToF-Systeme im Allgemeinen aus einem optischen Transmitter und einem optischen Empfänger. Dabei wird ein Lichtimpuls von der Sendeeinheit ausgesendet und mit Hilfe der Round-trip-time (RTT) der Abstand des Ziels ermittelt. Die RTT ist die Zeitdauer, die der Impuls vom Sender zum Ziel und wieder zurück zum Empfänger benötigt. Unter Kenntnis der Lichtgeschwindigkeit ist die Berechnung des Abstands leicht möglich. Aufwändig bei diesem Prinzip ist jedoch, dass bei einer Abstandsgenauigkeit im Millimeter-Bereich der Empfänger mit einer Genauigkeit im pico-Sekunden-Bereich zwischen Start und Stop messen können muss, da Licht für 1 mm nur 6,6 ps benötigt.

Laserscanner basieren auf dem ToF-Prinzip und bestehen aus einem Punkt-förmigen Empfänger und einem Sender, der zum Gewinnen von Tiefeninformationen einen einzelnen modulierten Laserstrahl zum Scannen der Szene aussendet. Um die gesamte Szene abzutasten, wird der Laserstrahl durch einen mechanisch rotierenden Spiegel von einer Position zur nächsten umgelenkt. Beim einem 2D-Laserscanner kann der Spiegel nur um eine Achse rotiert werden, während bei 3D-Laserscannern eine Rotation um zwei Achsen zur Abtastung einer gesamten Szene möglich ist. Der Strahl wird an Objekt-Oberflächen reflektiert, wodurch der Abstand zum Punkt auf der Objekt-Oberfläche mit Hilfe des Laufzeitverfahren berechnet werden kann. In Kombination mit der aktuellen Rotationsposition des Spiegels können die Auftrittspunkte in Polarkoordinaten repräsentiert werden.

Laserscanner haben im Gegensatz zu Triangulationsverfahren den Vorteil, dass sie auch Abstandsinformationen zu einfarbigen musterlosen Flächen liefern. Als nachteilig erweist sich jedoch, dass sie teuer, verhältnismäßig groß, vibrationsempfindlich und nur eingeschränkt echtzeitfähig sind. So benötigt ein Laserscanner in der Praxis laut [12] bei einem Scan-Winkel von 150° x 90° mindestens vier Sekunden für ein komplettes 3D-Bild seiner Umgebung. Dies entspricht einer Framerate von 0,25 fps und bestätigt somit die bedingte Echtzeitfähigkeit, weshalb ein 3D-Laserscanner für die in dieser Arbeit vorgestellte Anwendung ebenfalls nicht in Frage kommt. [10] [11] [12]

2.3.3 PMD-Kameras

PMD-Kameras (Photonic Mixer Device) stellen eine neue Generation von ToF-Kameras dar und kombinieren sowohl Intensitätswerte als auch Tiefeninformation in jedem Pixel des Sensors. Man spricht dabei von so genannten „smart pixels“, während herkömmliche Bildverarbeitungssysteme wie CCD-Sensoren nur die Intensität jedes Pixels ohne Tiefeninformationen wahrnehmen können.

Eine PMD-Kamera besteht aus mehreren Komponenten. Der PMD-Chip mit seiner peripheren Elektronik ist die Schlüsselkomponente des Systems, da er die von herkömmlichen (2D-) Kameras bekannte laterale Auflösung bestimmt und jeden Pixel mit Tiefeninformationen über den zugehörigen Punkt in der Objektebene versieht. Als Beleuchtungsquelle werden bei PMDs meist (Infrarot-) LEDs verwendet, da bei Laser-Dioden wesentlich mehr Achtung auf sicherheitskritische Aspekte gelegt werden muss. Durch die verwendete Beleuchtungsquelle wird das Sichtfeld der Kamera bestimmt. Die Empfängeroptik macht analog zu 2D-Kameras ein Bild von der Szene, wobei jedoch zusätzlich zur Beleuchtung auch die Laufzeit des ausgesendeten Lichts an einem Bildpunkt festgestellt wird.

Abbildung 2.8 zeigt grob das Funktionsprinzip einer PMD-Kamera. Moduliertes Licht (anstelle eines einzelnen Laserstrahls) beleuchtet die gesamte Szene, die wiederum mit ei-

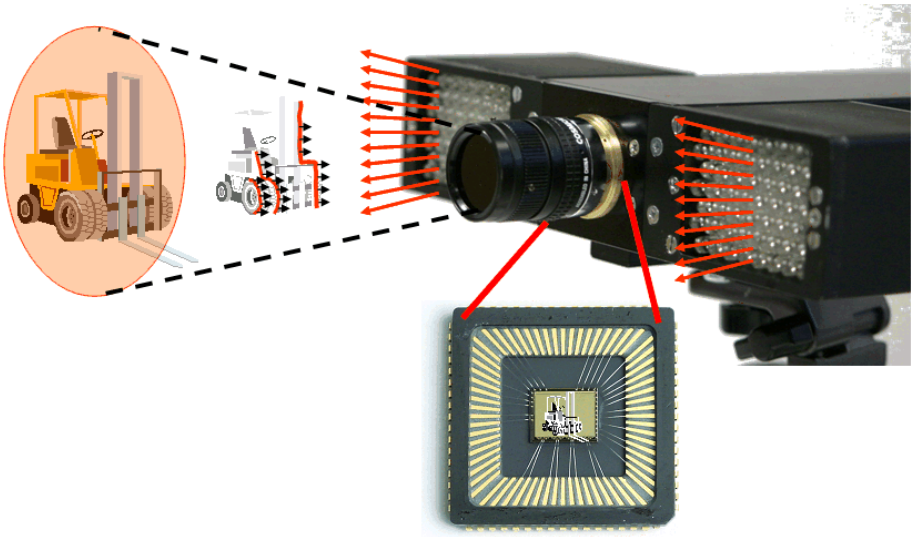


Abbildung 2.8: Funktionsweise von PMD-Kameras [11].

ner intelligenten Anordnung von Pixeln beobachtet wird. Jeder einzelne Pixel in der PMD-Kamera kann die Laufzeit des ausgesendeten modulierten Lichts messen, was typischerweise mit kontinuierlicher Modulation und Messung der Phasenverschiebung erfolgt. Mit dieser gemessenen Laufzeit kann schließlich dem ToF-Prinzip entsprechend der Abstand des Punkts in der Objektebene festgestellt werden.

Da die gesamte Szene gleichzeitig mit moduliertem Licht beleuchtet wird und die Laufzeit für jeden PMD-Pixel gleichzeitig gemessen werden kann, ist eine sehr vorteilhafte schnelle dreidimensionale Bildgebung einer Szene realisierbar, was bei einem Laserscanner aufgrund des mechanischen Scannens nicht möglich ist. Dies stellt den großen Vorteil gegenüber den anderen vorgestellten Systemen dar, da mit diesem Verfahren ein echtzeitfähiges Gewinnen von Tiefeninformationen mit bis zu 25 fps erreicht werden kann. Durch den im Vergleich zu Scanner-Systemen mechanisch relativ einfachen Aufbau und den von der Kamera selbst tragbaren Berechnungsaufwand sind eine preiswerte Herstellung und somit geringe Einkaufspreise bei den neuen industriellen PMD-Kameras möglich. Durch das direkte Zurückliefern von Tiefenwerten je Pixel aus der PMD-Kamera muss auch kein aufwändiges Korrespondenz-Problem mehr gelöst werden. Auch die Problematik von einfarbigen Flächen ohne Muster stellt auf Grund des ToF-Prinzips keine Hürden für PMD-Kameras dar, allerdings kann bei dunklen Flächen ein starkes Rauschen der Abstandswerte vorliegen. [11]

Zur Auswahl des am besten geeigneten Verfahrens zum Gewinnen von Tiefeninformationen wurde eine Nutzwertanalyse erstellt, die in Anhang A.3 zu sehen ist. Aus dem Ergebnis der Nutzwertanalyse ist abzuleiten, dass PMD-Kameras auf Grund ihrer Echtzeitfähigkeit und der relativ günstigen und einfachen Integrationsmöglichkeit die optimale Wahl für die in dieser Arbeit zu implementierende Arbeitsraumüberwachung darstellen.

2.4 Bounding-Box-Test

Um unbekannte Objekte in der Umgebung als Hindernisse erkennen zu können ist es notwendig, die von der Tiefenbild-Kamera zurückgelieferten dreidimensionalen Daten gegen bereits bekannte Objekte in der Umgebung zu testen. Mit Hilfe von so genannten Bounding-Box-Tests werden die Daten bereinigt, wodurch nur noch Daten von bisher unbekannten Objekten übrig bleiben, die für die weitere Verarbeitung relevant sind.

Als bekannte Objekte können Objekte innerhalb eines vorab modellierten statischen 3D-Umgebungsmodells herangezogen werden, das den Arbeitstisch sowie den Roboterarm beinhaltet. Einzelne Teilmodelle des Umgebungsmodells, zum Beispiel ein Gelenk des Roboters, wurden in ihrem eigenen lokalen KoSy modelliert, in dem das Bauteil relativ zum Ursprung dieses KoSy ausgerichtet ist. Für die korrekte Positionierung innerhalb der Umgebung werden diese lokalen Koordinatensysteme beim initialen Laden der Szene und bei Bewegungen des Roboterarms entsprechend an die Positionen innerhalb des globalen Welt-KoSy transformiert. Bei den Transformationen verändern sich die Lage und die Ausrichtung des Bauteils innerhalb seines lokalen KoSy jedoch nicht.

Jedes Bauteil wird von einer Quader-förmigen Bounding-Box umgeben, die beim Laden der Szene sowie bei Transformationen automatisch mit berechnet wird. Diese Bounding-Boxen sind entlang der Achsen des lokalen KoSy des Bauteils ausgerichtet, weshalb man von so genannten „axis aligned bounding boxes“ spricht. Eindeutig gekennzeichnet werden können solche Bounding-Boxen durch die Eckpunkte min und max auf beiden Seiten der Quader-Diagonalen. Der Vorteil von solchen Bounding-Boxen ist eine einfache und schnelle Berechnung von Überschneidungen, da nur Tests gegen ein geometrisch einfaches Objekt statt gegen ein komplexes Objekt nötig sind.

Die Tiefenbild-Kamera liefert eine dreidimensionale Punktwolke, die in Koordinaten des globalen Welt-KoSy ausgedrückt ist. Da die einzelnen Punkte der Punktwolke und die Bounding-Boxen des Umgebungsmodells sich in unterschiedlichen Koordinatensystemen befinden, ist hier ein einfacher Bounding-Box-Test nicht möglich, so dass die zu vergleichenden Objekte und Punkte erst transformiert werden müssen.

Die aktuelle Transformation $T_{BoundingBox}$ des lokalen KoSy einer Bounding-Box ist bekannt, so dass die einzelnen Punkte p der Punktwolke analog zu Abbildung 2.9 unter Anwendung der inversen Transformation $T^{-1}_{BoundingBox}$ zurück in den Ursprungsframe auf p' abgebildet werden können. Nun kann wegen des vergleichbaren Koordinatensystems eine einfache Überprüfung auf Überschneidungen stattfinden, wobei für $p' > min \wedge p' < max$ ein Punkt innerhalb der Bounding-Box liegt.

2.5 Clusteranalyse

Da selbst nach dem Bounding-Box-Test immer noch eine große Anzahl von dreidimensionalen Punkten der Tiefenbild-Kamera die unbekannten Objekte in der Umgebung beschreiben, soll ein weiteres Zusammenfassen von Punkten erfolgen. Dies ist insofern wichtig, da so die Punkte effizient veröffentlicht und vom Kollisionsvermeidungs-Algorithmus ausgewertet werden können.

Eine Möglichkeit zum Gruppieren von Punkten im Raum stellt Clustering dar, unter dem man das Finden von Häufigkeitsregionen im Merkmalsraum versteht. Unter der An-

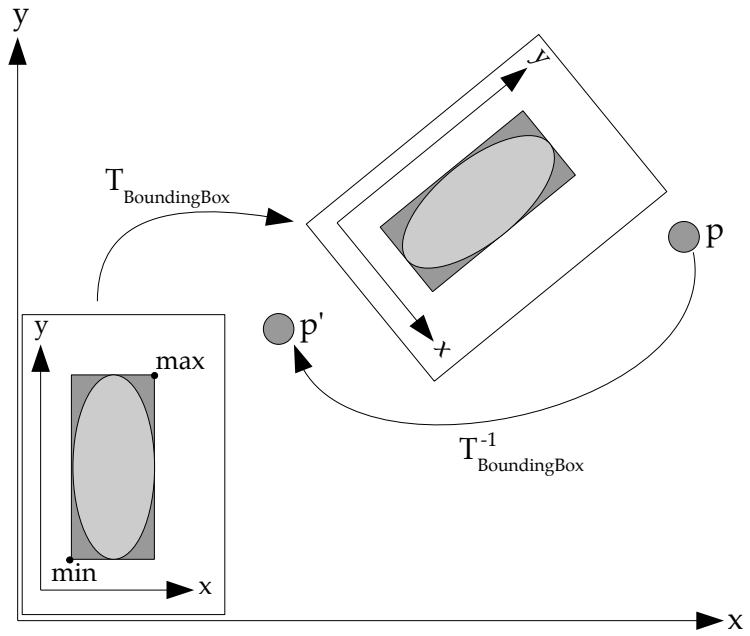


Abbildung 2.9: Inverse Transformation eines Punkts p nach p' beim Bounding-Box-Test.

nahme, dass es eine bekannte Anzahl von Klassen C gibt, in die der Merkmalsraum eingeteilt werden kann, findet ein Clustering-Verfahren die C wahrscheinlichsten Häufungspunkte und ordnet jedem Cluster die zugehörigen Elemente im Merkmalsraum zu. Hierbei ist der Abstand zwischen Klassen größer als der Abstand von Elementen innerhalb einer Klasse.

Clustering-Verfahren. Clustering-Verfahren lassen sich in hierarchische und partitionierende Verfahren einteilen. Kennzeichnend für hierarchisches Clustering ist entweder eine Gruppierung von Merkmalsvektoren von Segmenten nach einer Top-Down-Strategie durch fortgesetzte Zerlegung in Cluster oder ein Erzeugen von Clustern nach einer Bottom-Up-Strategie durch fortgesetzte Verschmelzung. Clustering durch Agglomeration verfolgt eine Bottom-Up-Strategie und stellt ein Beispiel für ein solches hierarchisches Clustering-Verfahren dar.

Im Gegensatz zu hierarchischen Verfahren wird der Merkmalsraum bei partitionierenden Clustering-Verfahren in eine vorab gegebene Anzahl von Clustern zerlegt. Das Grundprinzip besteht aus einer initialialen Zuordnung aller Stichproben zu Clustern und der Veränderung der Zuordnung der einzelnen Stichproben mit dem Ziel der Optimierung einer Zielfunktion. Da ein solcher Algorithmus auch für die in dieser Arbeit angestrebte Gruppierung von Punkten innerhalb einer Punktwolke geeignet ist, wird im Folgenden der K-Means-Algorithmus als geläufigster Vertreter von partitionierendem Clustering näher erläutert.

K-Means-Algorithmus. Beim K-Means-Algorithmus steht K für die gegebene Anzahl der Cluster C, denen die Segmente $S = \{s_0, \ldots, s_{N-1}\}$ des Merkmalsraums unter ständiger Optimierung zugeordnet werden. Dieser Algorithmus ist ein iterati-

ves Verfahren zum Finden von optimalen Cluster-Zentren und einer Zuordnung von Segmenten zu Clustern. Als Zielfunktion ist der durchschnittliche Abstand aller Elemente eines Clusters zu ihrem Cluster-Zentrum zu optimieren, was wegen der Cluster-Zuordnung aller Elemente und der Orte der Cluster-Zentren als Variablen der Zielfunktion schwierig ist. Durch den hochdimensionalen Definitionsbereich der Zielfunktion kommt eine vollständig Suche nicht in Frage, weshalb ein iteratives Verfahren verwendet wird. Dieses Iterationsverfahren setzt sich aus einem Initialisierungsschritt, einem Minimierungsschritt für die Zuordnung von Elementen aus S zu Clustern und einem Aktualisierungsschritt für die Orte der Cluster-Zentren zusammen.

In einem Initialisierungsschritt werden C Elemente aus S zufällig ausgewählt und als Cluster-Zentren $c_0, \ldots, c_{C-1}$ deklariert.

Im anschließenden Minimierungsschritt erfolgt eine Zuordnung der verbleibenden $N - C$ Elemente aus S zum am nächsten gelegenen Cluster. Da jedoch nach der Zuordnung das Cluster-Zentrum nicht mehr unbedingt im Zentrum des Clusters liegt, wird nach jeder einzelnen Zuordnung der Aktualisierungsschritt ausgeführt, der den Ort des Cluster-Zentrums optimiert. Der neue Ort des Cluster-Zentrums wird der durchschnittliche Ort aller zum Cluster gehörenden Elemente im Merkmalsraum. Die Cluster-Zentren wandern somit während der Zuordnung. Nun erfolgt eine Neuzuordnung aller Elemente zu den nach in der vorherigen Iteration gefundenen Cluster-Zentren. Minimierungs- und Aktualisierungsschritt werden so lange wiederholt, bis sich keine Änderungen mehr ergeben (Zielfunktion optimal) oder die Verbesserung des Werts der Zielfunktion unter ein vorgegebenes Minimum fällt.

Bei dieser Strategie gibt es jedoch keine Garantie, dass das globale Minimum der Zielfunktion gefunden wird. Das Verfahren terminiert zwar in einem lokalen Minimum, das jedoch nur bei einer günstig gewählten Anfangsbedingung auch das globale Minimum der Zielfunktion ist. Die Wahrscheinlichkeit eines nahe an der optimalen Lösung liegenden Endergebnisses kann mit Hilfe von unterschiedlichen Initialisierungen erhöht werden. [13]

Quickhull-Algorithmus. Eine andere Möglichkeit zum Zusammenfassen einer Menge von Punkten zu größeren Objekten stellt der Quickhull-Algorithmus dar, mit dem die konvexe Hülle einer Punktwolke berechnet werden kann. Während der K-Means-Algorithmus als Ergebnis genauso viele Cluster liefert wie in der Eingabe festgelegt wurden, wird mit dem Quickhull-Algorithmus ein einziges Objekt ermittelt, dessen Form durch einen geschlossenen Polygonzug über alle äußeren Punkte der Punktmenge beschrieben ist. Der Algorithmus selbst wird nachfolgend im zweidimensionalen Raum eingeführt und auf den dreidimensionalen Raum erweitert.

Zuerst werden im zweidimensionalen Fall die „axis aligned bounding box“ zur Punktwolke ermittelt und die auf jeder Seite der Bounding-Box liegenden Punkte als Extrema markiert (Abb. [14] links oben). Diese Punkte liegen als Extrema bereits auf der konvexen Hülle und werden somit zu einer ersten viereckigen Approximation der Hülle verbunden. Alle inneren Punkte können nicht auf der konvexen Hülle liegen und werden somit entfernt (Abb. [14] rechts oben). Diese erste Approximation

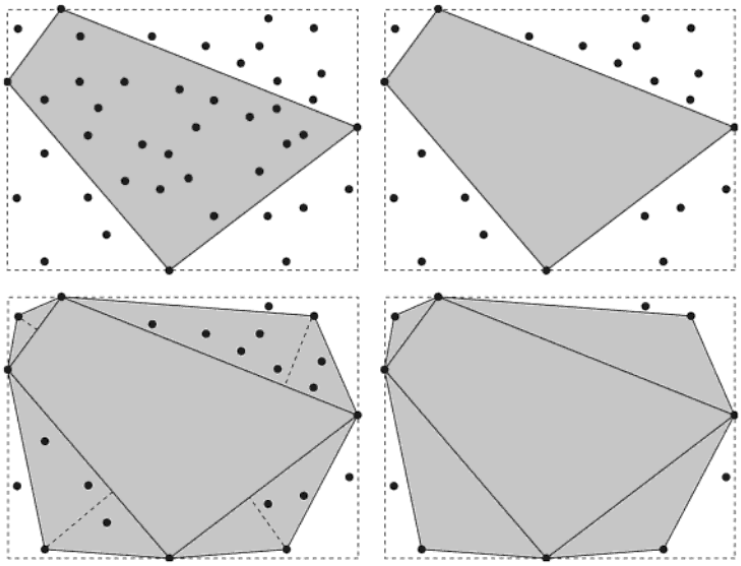


Abbildung 2.10: Ablauf des Quickhull-Algorithmus im zweidimensionalen Raum [14].

wird verfeinert, indem mit Hilfe des Lots zu jeder Kante der am weitesten entfernte außen liegende Punkt berechnet wird. Der gefundene Punkt wird nun zwischen die beiden Endpunkte der Kante in der konvexen Hülle eingefügt (Abb. [14] links unten). Werden diese drei Punkte miteinander verbunden, so ergibt sich ein Dreieck, deren innere Punkte wiederum nicht auf der konvexen Hülle liegen und somit verworfen werden (Abb. [14] rechts unten).

Nun findet eine rekursive Wiederholung dieser Prozedur für jede neue Kante der Hülle statt, bis sich keine Punkte mehr außerhalb befinden und somit die konvexe Hülle berechnet wurde.

Für eine robuste Implementierung dieses Algorithmus muss darauf geachtet werden, dass sich bei der initialen Approximation auch ein Dreieck oder eine Linie ergeben kann, falls ein Eckpunkt beispielsweise auf zwei Seiten ein Extremum ist. Somit muss die Verfeinerung der ersten Approximation auch mit einer beliebigen Anzahl von Kanten zurecht kommen. Außerdem können gleichzeitig mehrere Punkte von einer zu verfeinernden Kante am weitesten entfernt sein, wobei hier ein Punkt als Extremum ausgewählt werden kann.

Mit wenigen Modifikationen lässt sich der bisher für 2D beschriebene Quickhull-Algorithmus auch auf 3D erweitern. Die dreidimensionale „axis aligned bounding box“ besteht aus sechs Kanten, so dass auch die initiale Approximation sich aus bis zu sechs Extrem-Punkten zusammensetzt. Bei der anschließenden Verfeinerung werden nicht die Punkte gewählt, die am weitesten von den Kanten eines Polygons entfernt sind, sondern die Punkte, deren Abstände zu den Flächen eines Polyeders maximal sind. Statt dem Aufbrechen der Polygon-Kanten werden dann die Polyeder-Flächen in zwei oder mehrere Flächen aufgebrochen, wann immer ein Punkt in die konvexe Hülle eingefügt wird. [14] [15]

2.6 Vergleichbare Ansätze

Als Abschluss des Grundlagen-Kapitels soll auf vergleichbare Ansätze eingegangen werden, in denen es um ähnliche Szenarien der Arbeitsraumüberwachung in einer Mensch-Roboter-Kooperation geht. Einige dieser Ansätze sind auch in [4] aufgeführt.

In [16] und [17] stellen Stereo-Systeme die Grundlage für die Arbeitsraumüberwachung dar. [16] extrahiert Informationen über den Menschen basierend auf der Hautfarbe und verwendet die so ermittelte Lage des Menschen zur Einschränkung der Roboter-Geschwindigkeit. Dabei ist fragwürdig, ob auch andere Hindernisse problemlos erkannt werden. Außerdem ist das Erreichen von Echtzeitfähigkeit äußerst problematisch. In [17] sind die Sicherheitszonen des Roboters statisch, so dass seine Bewegungen bei der Kollisionserkennung nicht mit einbezogen werden.

[18] verwendet einen einzelnen Laserscanner, mit dem eine Ebene über dem Boden abgescannt wird. Alle entdeckten Hindernisse werden als stehende Menschen interpretiert und durch Zylinder approximiert, anhand deren Distanz die Geschwindigkeit des Roboters angepasst wird. Jedoch werden andere Hindernisse oder Menschen mit gebeugtem Oberkörper nicht korrekt berücksichtigt.

Der konzeptuell dieser Arbeit am nächsten gelegene Ansatz stellt [4] dar. Im darin vorgestellten echtzeitfähigen Szenario werden mehrere PMD-Kameras, die den gemeinsamen Arbeitsraum überwachen, für eine online-Kollisionserkennung verwendet. Die Tiefenbilder werden mit Hilfe eines geometrischen 3D-Umgebungsmodells bereinigt, so dass Hindernisse identifiziert werden können. Die Geschwindigkeit des Roboters wird durch seinen minimalen Abstand zu einem Hindernis beschränkt, so dass er stets vor einer drohenden Kollision angehalten werden kann. Die gesamte Verarbeitung und Aufbereitung der Sensordaten erfolgt hierbei an einem Master-Rechner, während sich Slave-Rechner nur um das Auslesen und Weiterleiten der PMD-Kameras kümmern. Dieses Szenario hat somit im Vergleich zum verteilten Ansatz von JAHIR den Nachteil, dass das Gesamtsystem nicht flexibel um weitere Komponenten erweiterbar ist, so dass auch eine Aufteilung des Berechnungsaufwands nicht erfolgen kann. In JAHIR können durch das verteilte System überdies auch mühelos die Daten von anderen Sensorgeräten mit in den Algorithmus zur Kollisionsvermeidung einbezogen werden, so dass eine umfangreichere Überwachung des Arbeitsraums möglich ist.

In [19] erfolgt eine Überwachung des gemeinsamen Arbeitsraum mit einer PMD-Kamera, jedoch handelt es sich auch hier um einen nicht verteilten Ansatz mit den damit verbundenen Nachteilen.

3 Kalibrierung der PMD-Kamera

Einen wichtigen Bestandteil dieser Arbeit stellt die zu integrierende Tiefenbild-Kamera dar, die sich um das echtzeitfähige und zuverlässige Gewinnen von dreidimensionalen Informationen über die Roboter-Umgebung kümmert. Wie in Abschnitt 2.3 dargestellt haben PMD-Kameras Vorteile gegenüber den anderen erläuterten Verfahren, so dass eine PMD-Kamera zur Arbeitsraumüberwachung in JAHIR integriert werden soll. Um die von der Tiefenbild-Kamera zurückgelieferten Informationen als Koordinaten im Welt-KoSy der Szene darstellen zu können und um eine bestimmte Genauigkeit zu erreichen ist eine genaue Kalibrierung der PMD-Kamera unumgänglich.

Ziel der Kalibrierung einer Kamera im Allgemeinen ist das Ermitteln von Kamera-spezifischen Besonderheiten (intrinsische Kalibrierung), damit das Bild unter Kenntnis dieser Besonderheiten korrigiert werden kann, sowie die Berechnung der Kameraposition und -orientierung innerhalb eines globalen Welt-KoSy (extrinsische Kalibrierung).

Da die laterale Auflösung der PMD-Kamera *ifm efector O3D200* für eine exakte extrinsische Kalibrierung zu gering ist, muss die Kalibrierung der PMD-Kamera unter Zuhilfenahme einer CCD-Kamera, im Set-Up vom Typ *Logitech QuickCam Pro 9000*, erfolgen. Diese CCD-Kamera kann extrinsisch kalibriert werden, so dass in einer anschließenden weiteren Kalibrierung nur noch die relative Position und Orientierung der PMD-Kamera zur Webcam ermittelt werden muss. Für eine fixe relative Positionierung von PMD-Kamera und CCD-Kamera werden beide Kameras in einem Sensorkopf zusammen montiert.

In diesem Kapitel werden zuerst die technischen Eigenschaften der verwendeten PMD-Kamera dargestellt, bevor näher auf die theoretischen Grundlagen der Kamera-Kalibrierung eingegangen wird. Anschließend werden Aufbau und Positionierung des für die Kalibrierung nötigen Sensorkopfes sowie der eigentliche Kalibrierungsvorgang erläutert.

3.1 Tiefenbild-Kamera ifm efector O3D200

Die Tiefenbild-Kamera stellt wie bereits erwähnt die zentrale Komponente zum Gewinnen von Informationen für die Arbeitsraumüberwachung in dieser Arbeit dar. Die PMD-Kamera *ifm efector O3D200* ist eine der ersten verfügbaren industriellen PMD-Kameras zu einem verhältnismäßig günstigen Preis. Sie bietet laut Datenblatt [20] eine Auflösung von 64 x 50 Pixeln und kann Frameraten von bis zu 25 fps erreichen. Über einen vorhandenen Ethernet-Anschluss kann die Kamera einfach in das verteilte JAHIR-System integriert werden. Die Beleuchtung der zu überwachenden Szene erfolgt mit Infrarot-LEDs, mit deren Hilfe eine Abstandsmessung bis zu einer maximalen Reichweite von 6,5 Metern realisiert werden kann.

Mit Hilfe von XML-RPC kann eine Plattform-unabhängige Kommunikation mit der PMD-Kamera erfolgen. Der Zugriff auf die Kamera kann jedoch mit einer im Lehrstuhl-internen Computer-Vision-Framework vorhandenen Wrapper-Klasse erfolgen, in der die

umfangreiche Kommunikation bereits gekapselt ist. Mit dieser Klasse ist ein Auslesen von Grauwert-Bildern, die vor allem für die Kalibrierung mittels Schachbrettmustern benötigt werden, und von dreidimensionalen Punktwolken, die als z-Koordinaten die Abstandsinformationen je Pixel enthalten, möglich.

3.2 Theoretische Grundlagen

Wie bereits erwähnt kann die verwendete PMD-Kamera aufgrund ihrer geringen Auflösung nur unter Zuhilfenahme einer zusätzlichen CCD-Kamera kalibriert werden.

Grundsätzlich stellt eine Kamera eine Abbildung zwischen der 3D-Welt und einem 2D-Bild dar, wobei ein 2D-Punkt als $m = [u, v]^T$ und ein 3D-Punkt als $M = [X, Y, Z]^T$ beschrieben werden kann. Die um 1 als letztes Element erweiterten Vektoren im homogenen KoSy sind $\tilde{m} = [u, v, 1]^T$ und $\tilde{M} = [X, Y, Z, 1]^T$.

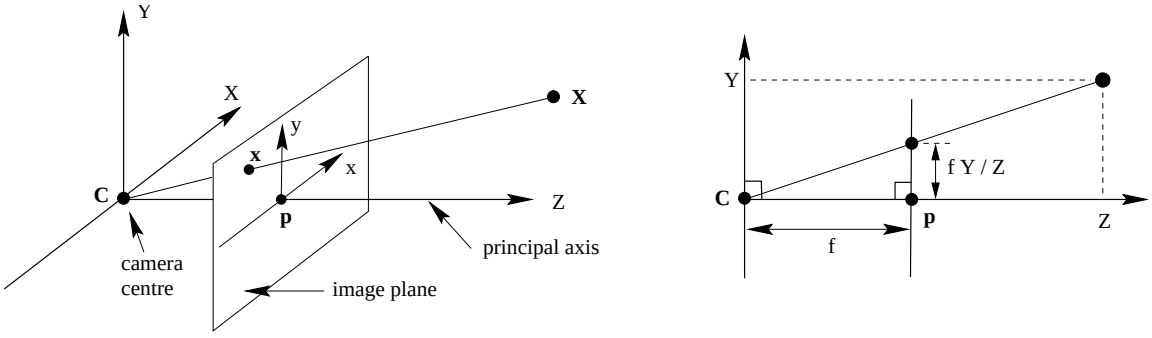


Abbildung 3.1: Geometrie einer Lochkamera mit Kamera-Zentrum C, principal point p und fokaler Länge f [21].

Beim in Abbildung 3.1 dargestellten Lochkamera-Modell, das als einfachstes Kamera-Modell gilt und somit als Grundlage dient, ist die Beziehung zwischen einem 3D-Punkt $\tilde{M}$ und seiner Bildprojektion $\tilde{m}$ durch die Gleichung

$$s\,\tilde{m} = A\,T\tilde{M} \tag{3.1}$$

gegeben. Hierbei stellen s einen beliebigen Skalierungsfaktor, A die intrinsische Kamera-Matrix und T die extrinsischen Parameter dar.

Intrinsische Kalibrierung. Ziel der intrinsischen Kalibrierung ist die Berechnung der intrinsischen Kamera-Matrix (oder Kalibrierungsmatrix) A einer Kamera, die die charakteristischen Eigenschaften des zugehörigen Kamera-Modells beschreibt:

$$A = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

f_x und f_y sind die Skalierungsfaktoren in x- und y-Achse der Bildebene und beschreiben die Zentralprojektion von 3D-Welt- in 2D-Bild-KoSy, wobei analog zu Abbildung 3.1 $f_x = fX/Z$ und $f_y = fY/Z$ gilt. p_x und p_y sind die Koordinaten des principal points p in der Bildebene.

Durch Ergänzen der z-Achse zur x- und y-Achse der Bildebene erhält man das so genannte Kamera-KoSy, wobei die z-Achse gemäß Abbildung 3.1 entlang der principal axis aus der Bildebene heraus gerichtet ist.

Extrinsische Kalibrierung. Im Allgemeinen werden Punkte in Koordinaten des euklidischen Welt-KoSy ausgedrückt, das mit dem Kamera-KoSy über eine Rotation R und eine Translation t in Relation steht. Die extrinsische Kalibrierung dient zur Berechnung von R und t, so dass die Position der Kamera innerhalb des Welt-KoSy ermittelt werden kann. Die 3x3 Rotationsmatrix R und der dreielementige Translationsvektor t ergeben zusammen die Transformation T aus Gleichung 3.1:

$$T = [R \quad t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \tag{3.3}$$

Gleichungen 3.2 und 3.3 eingesetzt in Gleichung 3.1 ergibt:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A\,T \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Da das Kalibrieren der PMD-Kamera aus praktischen Gründen mit Hilfe der Logitech-Kamera erfolgen muss, setzt sich die Transformation $T = T_{worldToPMD}$ vom Welt-KoSy zum KoSy der PMD-Kamera aus einer Kombination der Transformation vom Welt-KoSy zum KoSy der Logitech-Kamera und einer anschließenden Transformation von der Logitech-Kamera zur PMD-Kamera zusammen:

$$T = T_{worldToPMD} = T_{worldToLogitech} * T_{LogitechToPMD} \tag{3.5}$$

Veranschaulicht werden kann diese Abfolge von Transformationen anhand von Abbildung 3.2. [21] [22]

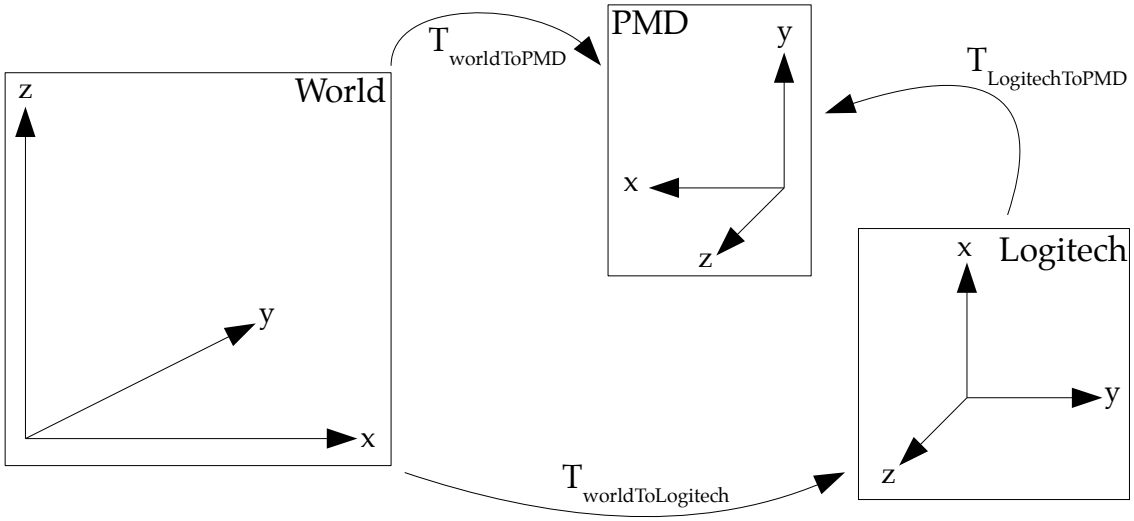


Abbildung 3.2: Abfolge von Transformationen zum Gewinnen der Transformationsmatrix $T_{worldToPMD}$.

3.3 Aufbau und Positionierung des Sensorkopfes

Der Sensorkopf für die Kalibrierung setzt sich aus der PMD-Kamera ifm efector O3D200 und der CCD-Kamera Logitech QuickCam Pro 9000 zusammen, wobei die Logitech-Kamera seitlich an der PMD-Kamera angebracht wurde (siehe Abbildung 3.3). Die unterschiedliche Orientierung und Neigung der beiden Kameras zueinander stellt in der Praxis kein Problem dar, da dies in der in Gleichung 3.5 dargestellten Gesamt-Transformation als $T_{LogitechToPMD}$ berücksichtigt wird.

Abbildung 3.3: Sensorkopf bestehend aus *ifm efector O3D200* und *Logitech QuickCam Pro 9000*.

Bei der Positionierung der PMD-Kamera bzw. des Sensorkopfes ist darauf zu achten, dass das Sichtfeld der Kamera den nahezu kompletten Arbeitsraum beinhaltet. Eine zur PMD-Kamera gehörende Windows-Software ermöglicht ein problemloses Auslesen der Grauwert-Bilder, so dass man wie in Abbildung 3.4 zu sehen den von der Kamera erfassten Arbeitsbereich leicht überprüfen kann. Als optimale Montageposition hat sich eine Position an der hinteren oberen Ecke auf der rechten Seite des Käfigs herausgestellt, zumal so der Hauptarbeitsbereich des menschlichen Arbeiters, der sich rechts neben dem Roboter befindet, meist ohne Okklusionen überwacht werden kann.

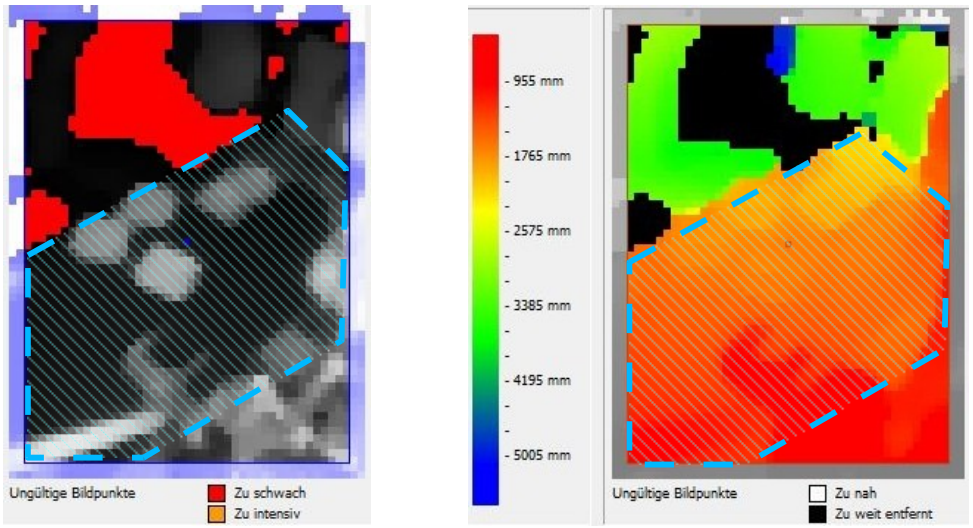


Abbildung 3.4: Überprüfen des einsehbaren Arbeitsbereichs mit der Windows-Software zum Auslesen der PMD-Kamera. Zur Veranschaulichung wurde der Arbeitstisch auf dem Intensitätsbild (links) und auf dem Distanzbild (rechts) blau markiert.

3.4 Kalibrierungsvorgang

Nach der festen Montage des Sensorkopfes kann der eigentliche Kalibrierungsvorgang erfolgen, dessen Ziel eine Abschätzung der intrinsischen und extrinsischen Kamera-Parameter ist. Eine solche Abschätzung geschieht in der Praxis meist unter Zuhilfenahme eines bekannten Schachbrettmusters. Durch mehrere möglichst unterschiedliche Ansichten dieses Musters können die gesuchten Parameter unter Zuhilfenahme von 2D-3D-Korrespondenzen ermittelt werden.

In der Praxis läuft die Kalibrierung der verwendeten PMD-Kamera mit Hilfe der CCD-Kamera wie folgt ab:

- Gleichzeitige Aufnahme von Bildern des Schachbrettmusters mit beiden Kameras.
- Berechnung der intrinsischen Parameter der beiden Kameras.
- Ermitteln der relativen Position der PMD-Kamera zur CCD-Kamera mittels Stereo-Kalibrierung.

- Berechnung der extrinsischen Parameter der CCD-Kamera.

Für die Kamera-Kalibrierung gibt es frei verfügbare und ausgereifte Software, die Schritte der Kalibrierung automatisiert und wesentlich erleichtert. Eine weit verbreitete Software zur Kalibrierung von Kameras stellt die *Camera Calibration Toolbox for MATLAB (CCTfM)* [23] dar, die zusätzliche Funktionen zur intrinsischen, extrinsischen und Stereo-Kalibrierung in MATLAB integriert.

Kalibrierungs-Bilder. Der erste Schritt bei der Kamera-Kalibrierung ist das Aufnehmen von Kalibrierungs-Bildern, auf denen ein Schachbrettmuster zu sehen ist.

Bei der Wahl eines geeigneten Schachbrettmusters ist darauf zu achten, dass die einzelnen Felder des Musters genügend groß sind, damit das Schachbrett auch in den gering aufgelösten Bildern der PMD-Kamera gut genug erkennbar ist. Das schließlich verwendete Schachbrett hat 5 x 3 Felder bei einer Feldgröße von 6,0 x 6,0 cm.

Für gleichzeitige Aufnahmen von beiden Kameras, die für die Stereo-Kalibrierung zwingend notwendig sind, wurde im Rahmen dieser Arbeit ein Tool erstellt, das sowohl aus der PMD-Kamera als auch aus der Logitech-Kamera zeitgleich in festen Zeitabständen Grauwert-Bilder aufnimmt, ausliest und als Bilddateien abspeichert. Dabei sollte das Schachbrettmuster auf beiden Bildern komplett zu sehen sein. Für eine stabile und gute Kalibrierung sind in der Praxis mindestens 20 Bilderpaare nötig, auf denen das Schachbrettmuster in möglichst vielen Variationen hinsichtlich Abstand und Rotationswinkel zu den Kameras zu sehen ist.

In Abbildung 3.5 sind beispielhaft 6 der 41 aufgenommenen Bilderpaare zu sehen.

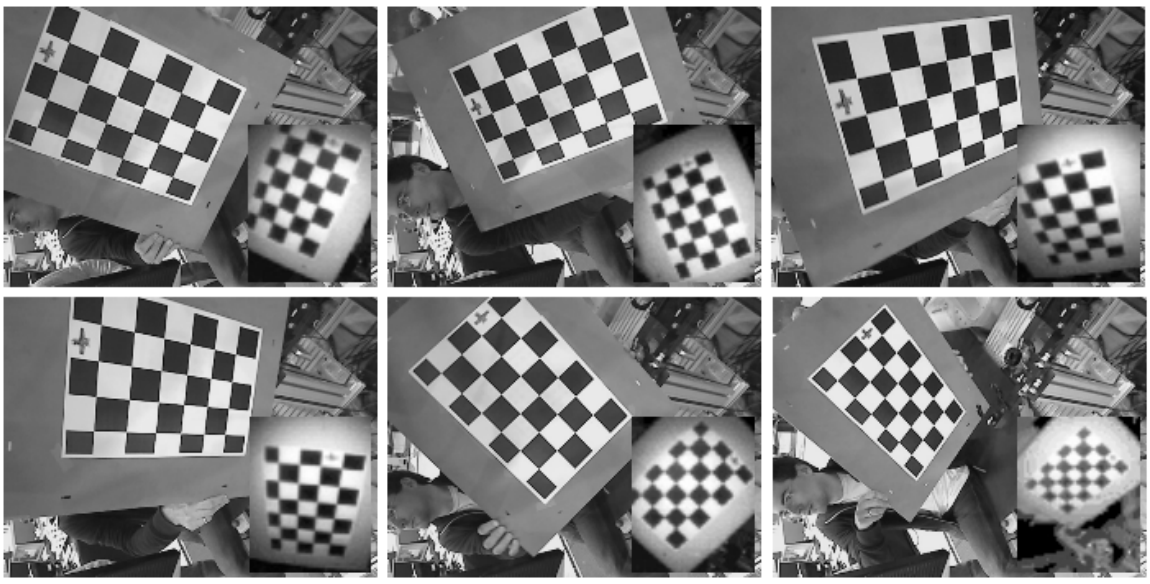


Abbildung 3.5: 6 der 41 aufgenommenen Bilderpaare für die Kalibrierung. Rechts unten innerhalb des CCD-Bildes befindet sich jeweils das zugehörige PMD-Bild.

Intrinsische Kalibrierung. Die aufgenommenen Bilder werden nun für die intrinsische Kalibrierung mittels CCTfM verwendet, die entsprechend der Anleitung unter [23]

durchgeführt werden kann. Dabei werden nach dem Markieren der Ecken des Schachbrettmusters in den Bildern der CCD-Kamera und dem Ausführen der Kalibrierung die intrinsischen Kalibrierungs-Parameter angezeigt. Anschließend werden auf die gleiche Weise auch die intrinsischen Parameter der PMD-Kamera ermittelt.

Stereo-Kalibrierung. Basierend auf den Ergebnissen der intrinsischen Kalibrierung der beiden Kameras erfolgt die Stereo-Kalibrierung analog zu den Anweisungen unter [24]. Als Ergebnis erhält man die Transformation $T_{LogitechToPMD}$ aus Gleichung 3.5. Diese Transformation besteht aus der Translation der PMD-Kamera ausgehend von der Logitech-Kamera und aus der zugehörigen Rotation in Form von Rodrigues-Parametern, die jedoch direkt in eine 3x3-Rotationsmatrix umgerechnet werden können. Abbildung 3.6 visualisiert das Ergebnis der Stereo-Kalibrierung in dreidimensionaler Form.

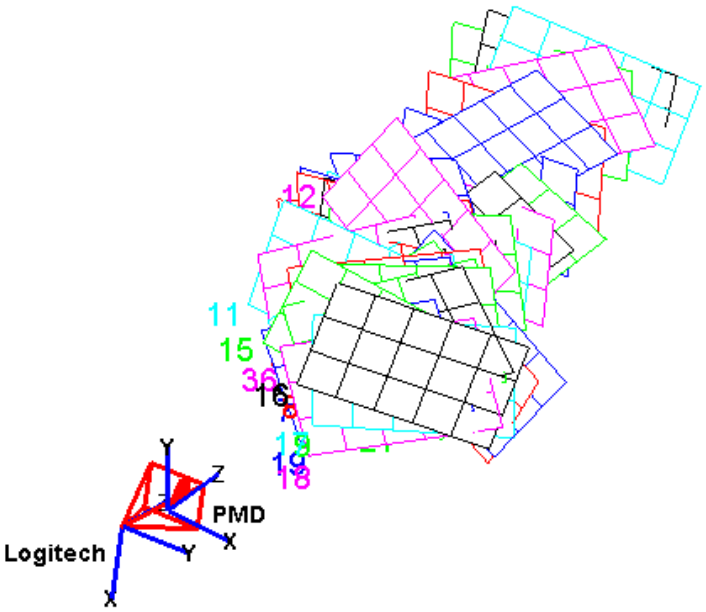


Abbildung 3.6: 3D-Darstellung des Ergebnisses der Stereo-Kalibrierung mit der *Camera Calibration Toolbox for MATLAB*.

Extrinsische Kalibrierung. Um die Transformation $T_{worldToPMD}$ aus Gleichung 3.5 zu erhalten, wird noch die Transformation der Logitech-Kamera $T_{worldToLogitech}$ ausgehend vom Welt-KoSy benötigt. Dafür ist ein Bild der Logitech-Kamera nötig, auf der ein möglichst großes Schachbrettmuster am Ursprung des Welt-KoSy ausgerichtet ist. Anschließend müssen in der CCTfM die Ecken des Schachbrettmusters wie in Abbildung 3.7 markiert werden, wobei mit dem Punkt des Schachbrettmusters begonnen werden muss, der mit dem Ursprungspunkt des Welt-KoSy übereinstimmt. Als Ergebnis erhält man die Transformation $T_{worldToLogitech}$, die wiederum aus dem Translationsvektor und der Rotation in Rodrigues-Parametern besteht.

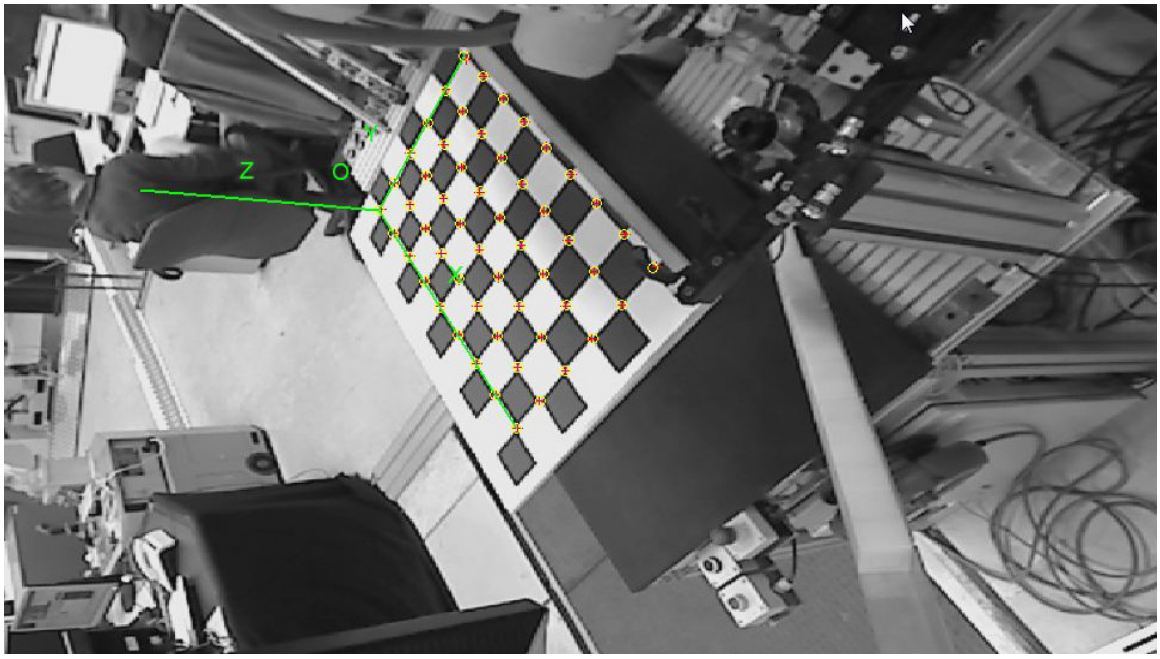


Abbildung 3.7: Ergebnis der extrinsischen Kalibrierung mit der *Camera Calibration Toolbox for MATLAB*. Auf dem 8x5-Schachbrettmuster mit einer Feldgröße von 10,0 x 10,0 cm sind bereits die Achsen des Welt-KoSy aufgetragen.

4 Aktive Arbeitsraumüberwachung mittels Tiefenbild-Kameras

Dieses Kapitel befasst sich mit der Implementierung einer Komponente zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kamera in JAHIR. Als Ausgangsbasis für die praktische Umsetzung dienen das in Abschnitt 2.1 dargelegte Gesamtkonzept sowie die theoretischen Grundlagen aus Kapitel 2. Die verwendete PMD-Kamera wurde bereits im vorherigen Abschnitt positioniert und kalibriert.

4.1 Systemarchitektur

Unter Kenntnis des Aufbaus und der Softwarearchitektur von JAHIR kann das Gesamtkonzept in eine konkrete Systemarchitektur für die zu erstellende Komponente überführt werden.

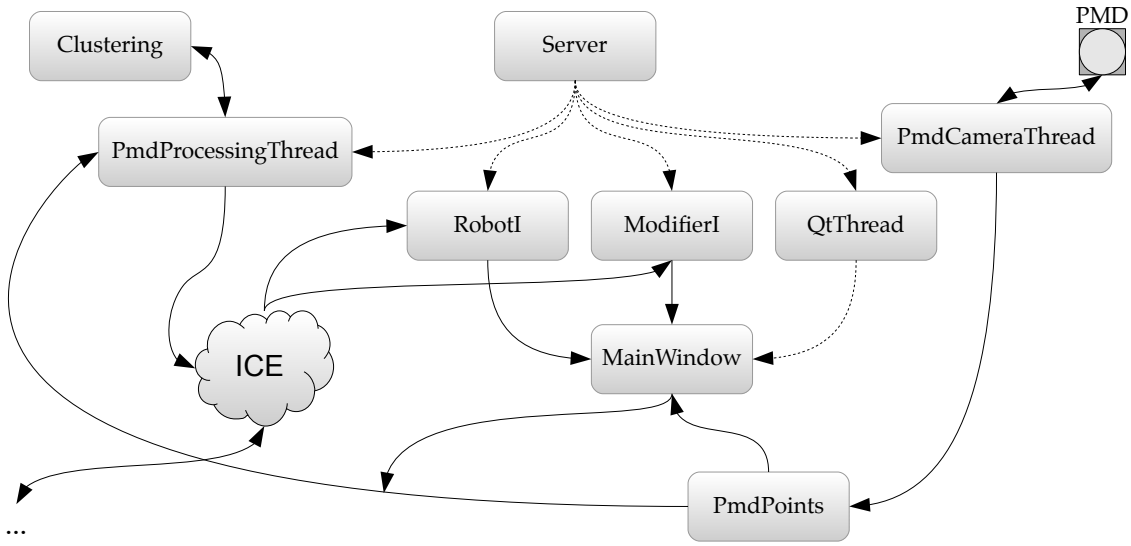


Abbildung 4.1: Systemarchitektur der erstellten Komponente.

Die in Abbildung 4.1 visualisierte Systemarchitektur besteht aus den im Folgenden näher beschriebenen Klassen, deren Aufbau detailliert im Klassendiagramm in Anhang A.4 zu sehen ist:

Server. Die Server-Klasse stellt die zentrale Klasse der erstellten Komponente dar. Darin wird das statische Umgebungsmodell (Szene) geladen sowie die Kommunikation mit dem Ice-Framework initialisiert. Ferner erfolgt hier das Starten des PmdPro-

cessingThread, PmdCameraThread und QtThread sowie das Erstellen eines RobotI- und eines ModifierI-Objekts.

PmdPoints. Die Klasse PmdPoints regelt den Zugriff auf die aus der PMD-Kamera ausgelesene Punktwolke und sorgt dafür, dass konkurrierende Objekte nicht gleichzeitig auf die Punkte zugreifen dürfen. Ferner enthält diese Klasse eine Methode zum Bounding-Box-Test der enthaltenen Punkte.

PmdCameraThread. Im PmdCameraThread erfolgt die Kommunikation mit der installierten PMD-Kamera. Dabei werden kontinuierlich Punktwolken aus der Kamera ausgelesen und an ein globales PmdPoints-Objekt übergeben.

QtThread. Innerhalb eines eigenen Threads werden hier die MainWindow-Klasse sowie die damit verbundenen Signal/Slot-Methoden initialisiert und instanziert.

MainWindow. Das MainWindow stellt das Hauptfenster der Komponente dar und dient zur Visualisierung der Szene. Darin erfolgt eine Anzeige des statischen Umgebungsmodells und ein Einzeichnen von dynamisch veröffentlichten Objekten der Umgebung. Außerdem wird in Abständen von 60 ms die dreidimensionale Punktwolke aus PmdPoints, die im PmdCameraThread ausgelesen wird, stets neu eingezeichnet.

ModifierI und RobotI. Diese beiden Klassen kümmern sich um die über das Ice-Framework gepublishten Objekte und Ereignisse. Über ModifierI werden dynamische Objekte der Szene hinzugefügt, geändert oder gelöscht. Bewegungen oder Änderungen des Roboters werden über RobotI registriert und verarbeitet. Mit diesen beiden Klassen empfangene Veränderungen werden auch unverzüglich in der Visualisierung umgesetzt. Um eine Kommunikation mit dem MainWindow über den Signal-/Slot-Mechanismus von Qt zu erreichen, sind ModifierI und RobotI von *ModifierObject* bzw. *RobotObject* abgeleitet, in denen jeweils die zugehörigen Signal-Methoden deklariert sind.

PmdProcessingThread. Das Verarbeiten der Daten der Tiefenbild-Kamera erfolgt im PmdProcessingThread. Nach der Übergabe von Bounding-Boxen von bekannten Objekten aus der Szene an die PmdPoints-Klasse werden die Punkte, die sich im Arbeitsraum befinden und zu keinem bekannten Objekt gehören, mit der Clustering-Klasse zu Clustern gruppiert. Abschließend erfolgt hier eine Veröffentlichung der gefundenen Cluster über Ice, so dass der Algorithmus zur Kollisionsvermeidung geeignet reagieren kann. Durch dieses Publishen werden die Cluster auch automatisch von ModifierI selbst empfangen und in die Visualisierung eingezeichnet.

Clustering. In der Clustering-Klasse werden die Punkte einer übergebenen Punktwolke einer festen Anzahl von Clustern zugeordnet. Dabei werden auch die Mittelpunkte und die auf der Standardabweichung basierenden Größen der Cluster berechnet. Außerdem ist die Funktionalität zur Berechnung der konvexen Hülle einer Punktwolke mit Hilfe des Quickhull-Algorithmus enthalten.

4.2 Szenenrepräsentation

Um eine Unterscheidung zu treffen, ob die aus der PMD-Kamera ausgelesenen Punkte zu vorab bekannten oder zu unbekannten Objekten gehören, ist ein internes Umgebungsmodell innerhalb der erstellten Komponente notwendig.

Grundsätzlich besteht diese interne Repräsentation der Szene aus mehreren Models, die wiederum beliebig viele Bodies enthalten können. Innerhalb eines Body, der einen eigens transformierbaren und separaten Teil eines Models darstellt, können beliebig viele geometrische Formen (sog. Shapes) wie Kugeln oder Quader-förmige Boxen sein. Beispielsweise ist der Roboter so modelliert, dass er selbst das übergeordnete Model ist und die einzeln beweglichen Gelenke die Bodies sind, die wiederum aus mehreren primitiven Shapes bestehen. Für die interne Szenenrepräsentation wurde die Klasse *rl::sg::so::Scene* aus der *Robotics Library* verwendet.

Innerhalb des Umgebungsmodells kann eine Aufteilung in statische und dynamische Umgebung erfolgen, auf die im Folgenden näher eingegangen wird.

4.2.1 Statische Umgebung

Das statische Umgebungsmodell basiert auf einer offline mit CAD modellierten VRML-Datei und enthält feste Objekte, die nicht gelöscht oder hinzugefügt werden können und sich nicht oder nur geringfügig zur Laufzeit verändern. Abbildung 4.2 zeigt das statische Umgebungsmodell des Set-Ups bestehend aus dem Arbeitstisch und dem Roboterarm. In der erstellten Komponente wird diese VRML-Datei beim Programmstart geladen und bereits in der Visualisierung angezeigt.

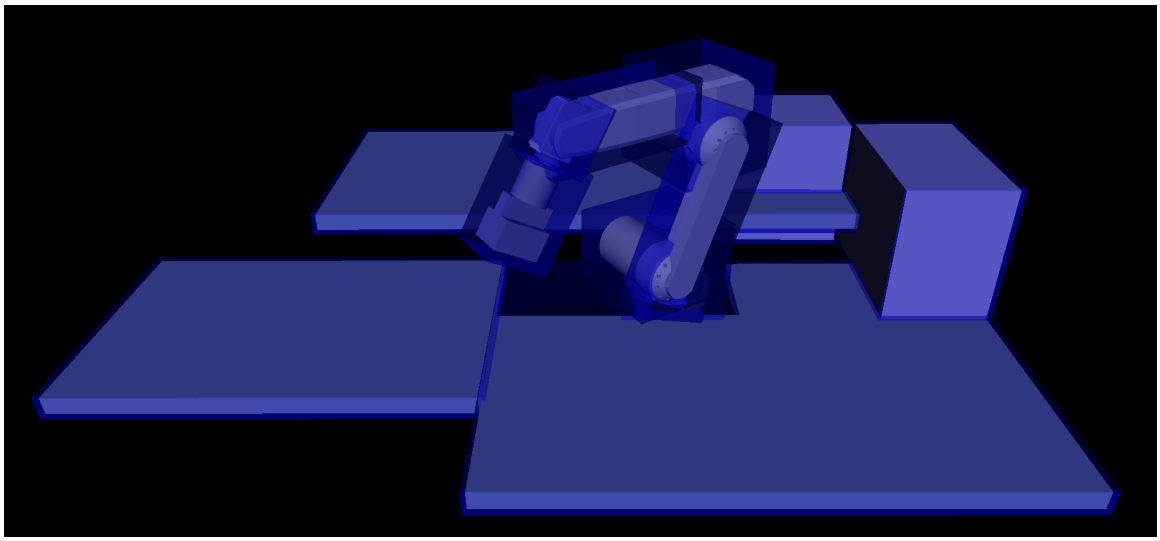


Abbildung 4.2: Statisches Umgebungsmodell von JAHIR.

Der Hauptzweck des statischen Umgebungsmodells besteht darin, dass damit eine Bereinigung der Sensordaten durch Bounding-Boxen-Tests erfolgen kann, so dass Punkte von bekannten Objekten herausfallen. Da für diese Bereinigung nur noch die Bounding-Boxen der Objekte interessant sind, wird ein neues Model in der Szene erstellt, zu dem die

Bounding-Boxen der einzelnen Bestandteile der statischen Umgebung hinzugefügt werden. Diese Bounding-Boxen sind in Abbildung 4.2 als transparente blaue Boxen eingezeichnet und im Vergleich zu den enthaltenen Teilen etwas vergrößert, damit eine robustere Zuordnung von PMD-Punkten zu bekannten Objekten erreicht wird.

Mit Vorsicht ist das Modell des Roboterarms zu behandeln, da sich Lage und Orientierung des Arms durch eine Bewegung des Roboters zur Laufzeit verändern können. Den Umgang mit dynamischen Veränderungen der Szene beinhaltet der nachfolgende Abschnitt.

4.2.2 Dynamische Umgebung

Dynamische Veränderungen der Umgebung können mit Hilfe bereits vorhandener Komponenten zur Arbeitsraumüberwachung, die in Abschnitt 2.2 erläutert wurden, entdeckt werden. Von diesen Komponenten ermittelte Hindernisse in der Umgebung und Veränderungen des Roboters werden über das Ice-Framework veröffentlicht, so dass alle restlichen Komponenten geeignete Reaktionen einleiten können.

In der im Rahmen dieser Arbeit erstellten Komponente wird mit den Klassen *ModifierI* und *RobotI* auf zur Laufzeit über Ice veröffentlichte Änderungen reagiert. Den Roboter betreffende Änderungen werden unter dem Ice-Topic „Robot“, dynamische Änderungen in der Szene werden unter dem Topic „Scene“ veröffentlicht.

Algorithmus 1 zeigt den Ablauf, wie in der Klasse *Server* die Kommunikation von *ModifierI* und *RobotI* mit Ice hergestellt wird, damit auf über Ice veröffentlichte Änderungen reagiert werden kann:

Algorithmus 1 Abonnieren von Ice-Topics mit *ModifierI* und *RobotI*.

```
create robotI
create modifierI
retrieve sceneTopic
if ∄ sceneTopic then
   create sceneTopic
end if
retrieve robotTopic
if ∄ robotTopic then
   create robotTopic
end if
subscribe modifierI to sceneTopic
subscribe robotI to robotTopic
```

Nach dem Abonnieren des „Robot“-Ice-Topics wird bei Veröffentlichungen von anderen Komponenten unter diesem Topic die entsprechende Methode in der *RobotI*-Klasse aufgerufen. So können beim Ausführen der Methoden *updateJoints()* und *updatePose()* zusätzlich zur Transformation der Gelenke des Roboters gleichzeitig die zugehörigen Bounding-Boxen mit transformiert werden, so dass sie auch nach Bewegungen des Roboters wieder mit der realen Position des Roboters übereinstimmen. Die Methode *changeTool()* bewirkt ein Auswechseln des aktiven Werkzeugs des Roboters.

Änderungen über das „Scene"-Topic werden direkt an die entsprechenden Methoden der Klasse *ModifierI* weitergereicht, mit denen dynamisch vorab unbekannte Objekte zum Umgebungsmodell hinzugefügt werden können. Die Methoden *addModel()* und *addBody()* kümmern sich um das Einfügen von Models und Bodies in die Szene, während sich mit *addBox()*, *addCylinder()* und *addSphere()* Shapes zu einem Body hinzufügen lassen. Über die Methoden *update()* und *remove()* lassen sich Bodies und komplette Models hinsichtlich Position und Orientierung anpassen oder gänzlich aus der Szene entfernen.

Da über das „Scene"-Topic veröffentlichte Objekte in der Regel a priori unbekannt sind und somit nicht zum statischen Umgebungsmodell gehören, werden sie auch nicht in den Bounding-Box-Test zum Bereinigen der Sensordaten mit einbezogen.

4.3 Auswertung der Sensordaten der PMD-Kamera

Der zentrale Verarbeitungsschritt der entwickelten Komponente ist die Auswertung der dreidimensionalen Daten der PMD-Kamera, wobei auf das im vorherigen Abschnitt erläuterte Umgebungsmodell zurückgegriffen wird.

Die dreidimensionalen Daten der PMD-Kamera werden in einem eigenen Thread *PmdCameraThread* analog zu Algorithmus 2 ausgelesen und über die Methode *setPointCloud()* in der Klasse *PmdPoints* gespeichert, die einen geschützten Zugriff auf die Daten gewährleistet.

Algorithmus 2 Wiederholtes Auslesen der PMD-Kamera.

```
open pmdCamera
init pmdCamera
while running do
  pointCloud ← capture next from pmdCamera
  init pmdPoints with pointCloud
end while
close pmdCamera
```

Der Ablauf des Aufbereitens der Sensordaten innerhalb eines Verarbeitungszyklus ist in Abbildung 4.3 veranschaulicht. Nach dem Auslesen der Tiefendaten aus der PMD-Kamera (1) können in einem weiteren Verarbeitungsschritt die von der Klasse *PmdPoints* bereitgestellten Punkte ausgeschlossen werden, die sich außerhalb des gemeinsamen Arbeitsraums von Mensch und Roboter befinden (2). Weitere Bounding-Box-Tests bereinigen die Punkte, die zu bereits bekannten Objekten gehören (3). Um die in das Umgebungsmodell einzuspeisende Datenmenge zu reduzieren, werden die übrig gebliebenen Punkte zu Clustern (bzw. zu einer konvexen Hülle) zusammengefasst (4) und veröffentlicht (5).

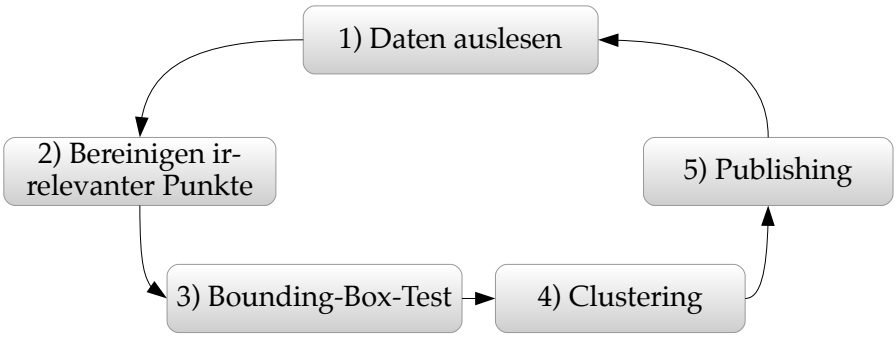


Abbildung 4.3: Ablauf eines Verarbeitungszyklus bei der Auswertung der Sensordaten.

4.3.1 Bereinigen der Sensordaten

Ziel der Bereinigung der Sensordaten ist das Aussortieren von für die Arbeitsraumüberwachung irrelevanten Punkten. Dies sind genau die Punkte, die zu bereits bekannten Objekten gehören und somit keine unbekannten Hindernisse darstellen. Damit einhergehend ist eine Reduzierung der zu verarbeitenden Datenmenge für das nachfolgende Clustering.

Beim Hinzufügen der Bounding-Boxen der statischen Umgebung in einem eigenen Model beim Programmstart werden auch die Eckpunkte des Arbeitsraums ermittelt. Alle Punkte links und rechts außerhalb dieser Eckpunkte können in einem ersten Schritt entfernt werden, da der Roboter eine geringere Reichweite aufweist. Ferner kann der Roboterarm keine Objekte erreichen, die sich mehr als 50 cm vor dem Tisch befinden, so dass auch Punkte jenseits dieser Distanz bereinigt werden können.

Die restlichen Punkte befinden sich innerhalb des gemeinsamen Arbeitsraums, können jedoch zu bekannten Objekten der statischen Umgebung gehören. Um Punkte, die der statischen Umgebung zuzurechnen sind, ebenfalls zu entfernen, werden die Punkte der PMD-Punktwolke gegen die Bounding-Boxen aller Teilobjekte des Arbeitstisches und des transformierten Roboters getestet. Die verbleibenden Punkte sind unbekannten Hindernissen zuzurechnen und werden somit weiterverarbeitet.

Algorithmus 3 zeigt, wie in der Klasse *PmdProcessingThread* eine Verarbeitung der Sensordaten erfolgt:

Algorithmus 3 Verarbeiten der Sensordaten in *PmdProcessingThread*.

```
boundingBoxModel ← get bounding-box model from scene
while running do
    unknownPoints ← test points in pmdPoints against boundingBoxModel
    clusters ← find clusters in unknownPoints
    publish found clusters
end while
```

Das Bereinigen der Tiefeninformationen inklusive dem auf Abschnitt 2.4 basierenden Bounding-Box-Test gegen das Bounding-Box-Model geschieht in der Methode *testPoints()* der Klasse *PmdPoints* nach folgendem Schema:

Algorithmus 4 Bereinigen der Sensordaten in der Methode *testPoints()*.

```
points ← get pointcloud from pmdPoints
clear knownPoints
clear unknownPoints
for all body b ∈ boundingBoxModel do
  for all point p ∈ points do
    if p outside of workspace then
      add p to knownPoints
    else if p inside bounding-box of b then
      add p to knownPoints
    end if
  end for
end for
for all point p ∈ points do
  if p ∉ knownPoints then
    add p to unknownPoints
  end if
end for
```

Über die Methode *getUnknownPoints()* der Klasse *PmdPoints* kann anschließend auf alle Punkte, die nach der Bereinigung übrig geblieben sind und somit unbekannten Objekten zuzurechnen sind, zugegriffen werden.

Mit Hilfe von Abbildung 4.4, in der ein vor dem Tisch stehender Arbeiter zum Roboterarm greift, kann das Bereinigen der PMD-Punktwolke veranschaulicht werden. Die Punkte links, rechts, unter und 50 cm vor dem Arbeitstisch werden als bekannt gesehen und somit grün angezeigt. Weitere Punkte, die in den Bereich der Bounding-Boxen fallen, werden ebenfalls grün markiert. Die restlichen Punkte sind unbekannten Objekten zuzuordnen und somit in roter Farbe.

4.3.2 Zusätzliche Filterung der Sensordaten durch Clustering

Da das Veröffentlichen eines Objektes über Ice, beispielsweise in Form einer Box, einen gewissen Overhead mit sich bringt, ist es unvorteilhaft, eine größere Anzahl von Punkten als einzelne Objekte über die vorhandenen Ice-Interfaces zu veröffentlichen. Durch Clustering können Punkthäufungen zu Objekten zusammengefasst werden, so dass die zu veröffentlichende Datenmenge erheblich reduziert werden kann.

K-Means-Algorithmus. Der zentrale Ablauf der Cluster-Berechnung, das Auslesen der ermittelten Cluster-Zentren und der zu einem Cluster gehörenden Punkte bei der Verarbeitung der Sensordaten in Klasse *PmdProcessingThread* ist in Algorithmus 5 dargestellt.

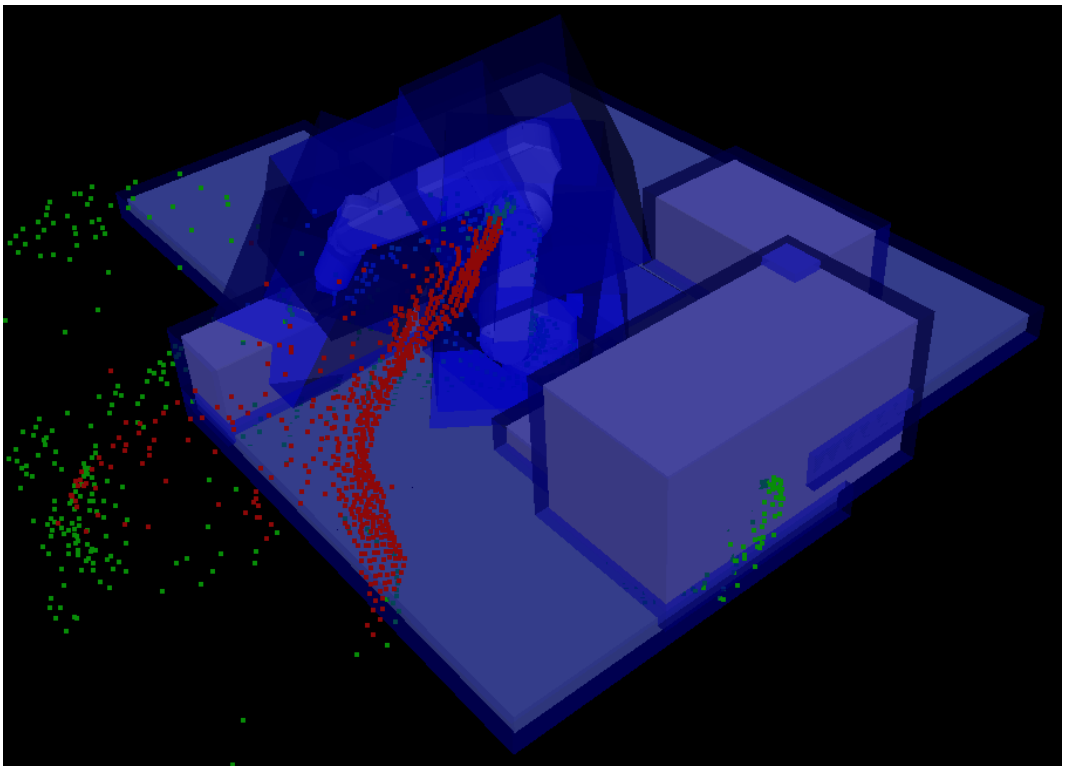


Abbildung 4.4: Visualisierung von zu bekannten (grün) und unbekannten (rot) Objekten gehörenden Punkten.

Algorithmus 5 Cluster-Berechnung, -Auslesen und -Publishen bei der Verarbeitung der Sensordaten im *PmdProcessingThread*.

```
init kmeans with number of clusters (100)
unknownPoints ← get unknown points from pmdPoints after tests
foundClusters ← find clusters in unknownPoints with k-means
for all cluster cl ∈ foundClusters do
  if cl has points then
    clCenter ← get center of cl
    clPoints ← get points of cl
    clStdDevX ← get x standard deviation of cl
    clStdDevY ← get y standard deviation of cl
    clStdDevZ ← get z standard deviation of cl
    clPosition ← clCenter
    clSize ← set size (w = clStdDevX * 2, d = clStdDevY * 2, h = clStdDevZ * 2)
    publish cluster at position clPosition with size clSize
  end if
end for
```

Dem Algorithmus 5 ist auch zu entnehmen, dass die Standardabweichung, die aussagt, wie sehr die Punkte eines Clusters im Durchschnitt vom Cluster-Zentrum ab-

weichen, als Cluster-Größe verwendet wird.

Die eigentliche Cluster-Berechnung basiert auf dem im Grundlagenteil (Abschnitt 2.5) vorgestellten K-Means-Algorithmus und findet in der Methode *findClusters()* der Klasse *Clustering* statt. Dabei wird auf die bereits vorhandene K-Means-Implementierung *cvKmeans2()* aus der OpenCV-Bibliothek zurückgegriffen. Der in Algorithmus 6 dargestellte Ablauf der Methode *findClusters()* zeigt die Berechnung der Cluster, Cluster-Zentren und Cluster-Punkte.

Algorithmus 6 Cluster-Berechnung in der Methode *findClusters()*.

```
unknownPoints ← get unknown points from pmdPoints after tests
clear kmClusters
clear kmClCenters
kmClusters, kmClCenters ← call cvKmeans2 (using unknownPoints, num of clusters)
for all cluster cl ∈ kmClusters do
  for all point p ∈ unknownPoints do
    if p belongs to cl) then
      add p to clusterPoints of cl
    end if
  end for
end for
```

Quickhull-Algorithmus. In der Praxis (vgl. Kapitel 5) wurde festgestellt, dass das vorgestellte Vorgehen der Cluster-Berechnung mit K-Means gravierende Nachteile aufweist, so dass eine alternative Zusammenfassung von Punkten mit Hilfe des in Abschnitt 2.5 eingeführten Quickhull-Algorithmus integriert wurde. Für diesen Algorithmus ist eine robuste Implementierung in Form des *Qhull Code* [25] verfügbar.

Algorithmus 7 zeigt im Groben, wie in der Methode *qhull()* der Klasse *Clustering* die Punkte zu einem IndexedFaceSet verarbeitet und an einen Node zur Visualisierung weitergegeben werden:

Algorithmus 7 Zusammenfassung einer Punktwolke zu einer konvexen Hülle mit Qhull.

```
remove children from qhull visualization node
unknownPoints ← get unknown points from pmdPoints after tests
vertices, facets ← call qhull (using unknownPoints)
coordinates ← process vertices
indexedFaceSet ← process facets
add coordinates to qhull visualization node
add indexedFaceSet to qhull visualization node
```

Die visualisierten Ergebnisse von K-Means und Quickhull sind in den Abbildungen 4.5 und 4.6 im nachfolgenden Abschnitt 4.5 veranschaulicht und geben jeweils gut die Grundidee des zugrunde liegenden Algorithmus wieder.

4.4 Wiedereinspeisen der gefilterten Messwerte in das Umgebungsmodell

Nachdem die Punkte der PMD-Kamera bereinigt und im vorherigen Abschnitt zu größeren Objekten zusammengefasst worden sind, können sie nun über das Ice-Framework veröffentlicht werden.

Die gefundenen Cluster bzw. die gefundene konvexe Hülle stellen dynamische Objekte in der Umgebung dar und sollen demzufolge mit Hilfe eines Ice-Proxy-Objekts unter dem Ice-Topic „Scene“ veröffentlicht werden.

Beim K-Means-Verfahren wird der Szene in der Klasse *PmdProcessingThread* im Rahmen des Publishings zuerst ein neues Model und darin ein Body hinzugefügt. In diesem Body werden anschließend alle Cluster in Form von Boxen eingefügt. Die Position einer Box wird wie erwähnt durch das jeweilige Cluster-Zentrum festgelegt, während die Größe einer Box anhand der Standardabweichung des Clusters in x-, y- und z-Richtung ermittelt wird.

Bei der auf Quickhull basierenden Alternative zum K-Means-Clustering ist für die Veröffentlichung der ermittelten Objekte eine Erweiterung der Modifier-Ice-Interfaces nötig, da bisher noch keine Methode zum Publishen von konvexen Hüllen vorhanden war.

Andere Komponenten von JAHIR, die ebenfalls die über den „Scene“-Topic von Ice veröffentlichten Änderungen empfangen, können nun auf die veröffentlichten Objekte reagieren. Beispielsweise kann nun die Komponente zur Visualisierung die empfangenen Objekte einzeichnen, um eine aktuelle Darstellung der Umgebung zu erhalten. Der Algorithmus zur Kollisionsvermeidung behandelt diese dynamisch hinzugefügten Objekte als Hindernisse, so dass er die Bewegungsplanung des Roboterarms gegebenenfalls anpassen oder bei einer unmittelbar bevorstehenden Kollision den Roboter stoppen kann.

4.5 Visualisierung

Den letzten noch ausstehenden Punkt des Gesamtkonzepts stellt die Visualisierung dar, die mit der Klasse *MainWindow* erfolgt. Darin wird zu jeder Zeit das interne 3D-Umgebungsmodell des Systems für den Benutzer veranschaulicht.

Dabei erledigt die aus dem *SoQt*-Framework stammende Klasse *SoQtExaminerViewer* die praktische Anzeige der Szene. Über Slot-Methoden wie beispielsweise *updateRobot()* und *updateBody* werden von *ModifierI* bzw. *RobotI* weitergereichte Umgebungsveränderungen auf die Darstellung übertragen.

Nach dem Laden der Szene beim Öffnen des *MainWindow* wird die Methode *addBoundingBoxes()* zur Generierung des Models für die Bounding-Boxen aufgerufen. Gegen dieses Model werden dann die in Abschnitt 4.3.1 erläuterten Bounding-Box-Tests durchgeführt.

In zeitlichen Abständen von 60 ms wird mit Hilfe eines Timers die Methode *updatePmdPoints()* aufgerufen. Darin werden die zuletzt aus der PMD-Kamera ausgelesenen Punkte angezeigt, wobei die zu bekannten Objekten gehörenden Punkte in grüner Farbe und die zu unbekannten Hindernissen gehörenden Punkte in roter Farbe eingezeichnet werden.

Die Abbildungen 4.5 und 4.6 zeigen jeweils eine Szene in der Visualisierung, in der ein Arbeiter vor dem Tisch steht und nach dem Roboter greift, wobei die Gruppierung von Punkten mit unterschiedlichen Verfahren erfolgt.

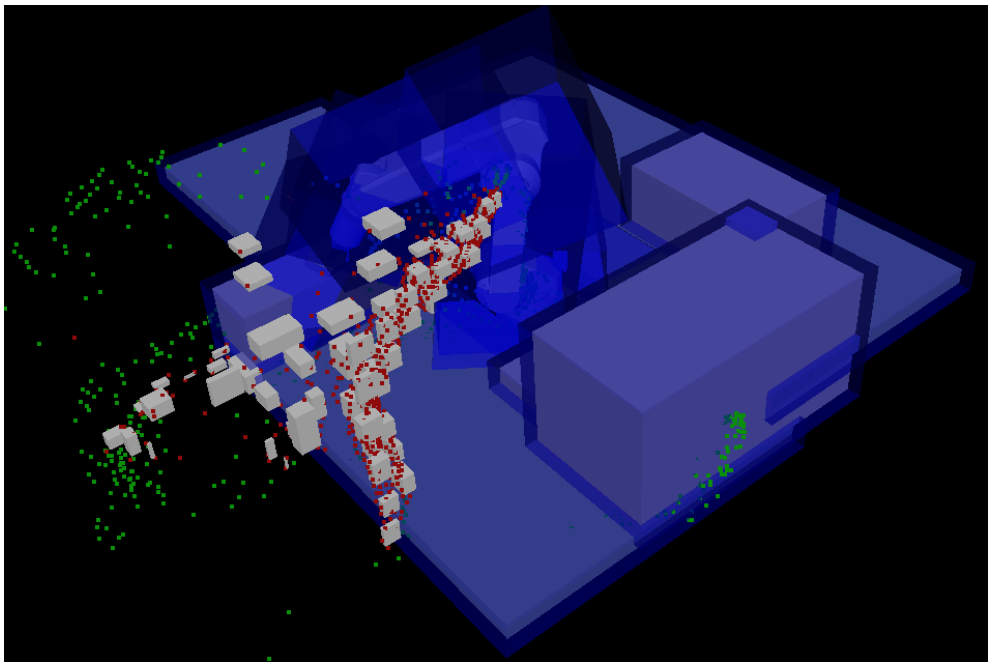


Abbildung 4.5: Visualisierung mit K-Means-Clustering.

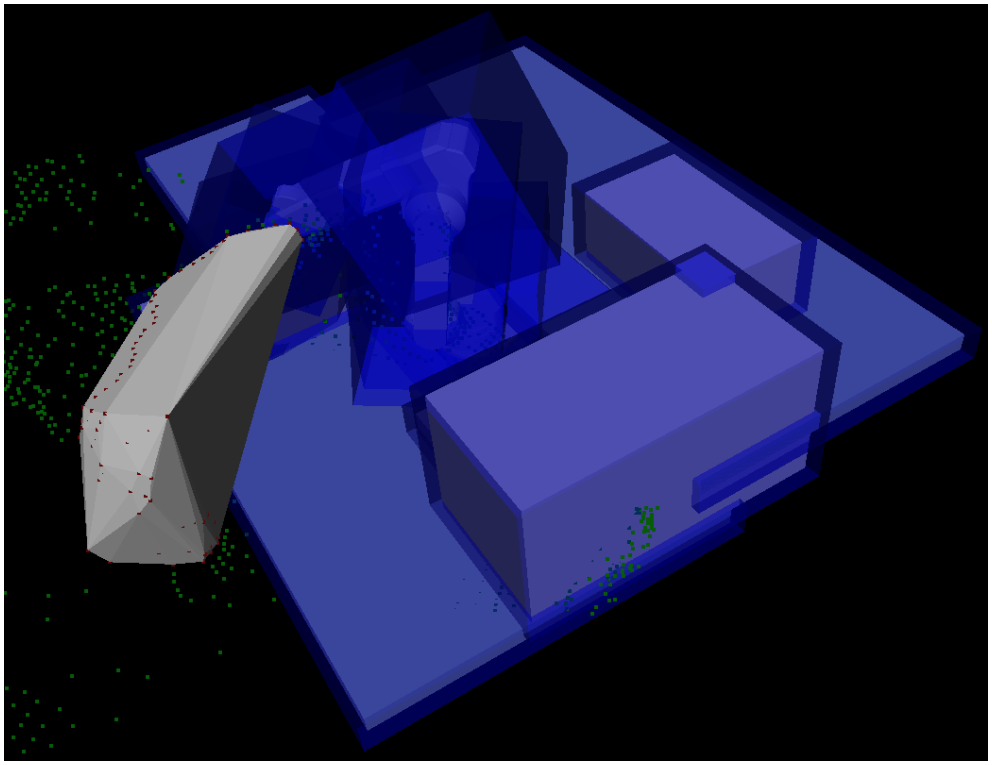


Abbildung 4.6: Visualisierung mit Quickhull.

5 Evaluation

In diesem Kapitel soll eine ausführliche Evaluation der im Rahmen dieser Arbeit erstellten Komponente erfolgen, wobei ein besonderer Fokus auf Performanz und Genauigkeit gelegt wird.

5.1 Genauigkeit

Zuerst sollen die Genauigkeit des Systems betreffende Aspekte näher betrachtet werden.

Kalibrierung. Wie in Abschnitt 3 beschrieben muss die Kalibrierung einer PMD-Kamera mit Hilfe einer zusätzlichen CCD-Kamera erfolgen. Während die extrinsische Kalibrierung der CCD-Kamera wegen der hohen Auflösung ein hohes Maß an Genauigkeit aufweist, ist die mit der Stereo-Kalibrierung berechnete Transformation von der PMD- zur Logitech-Kamera wegen der geringen Auflösung der PMD-Kamera mit einem verhältnismäßig großen Fehler behaftet. Dabei weicht vor allem die ermittelte Rotation stark von den realen Werten ab, so dass eine manuelle Korrektur nötig ist.

Abbildung 5.1 veranschaulicht die Abweichung der von der Kalibrierung gelieferten extrinsischen Parameter im Vergleich zu den optimalen Parametern, die durch manuelle Anpassungen gefunden wurden.

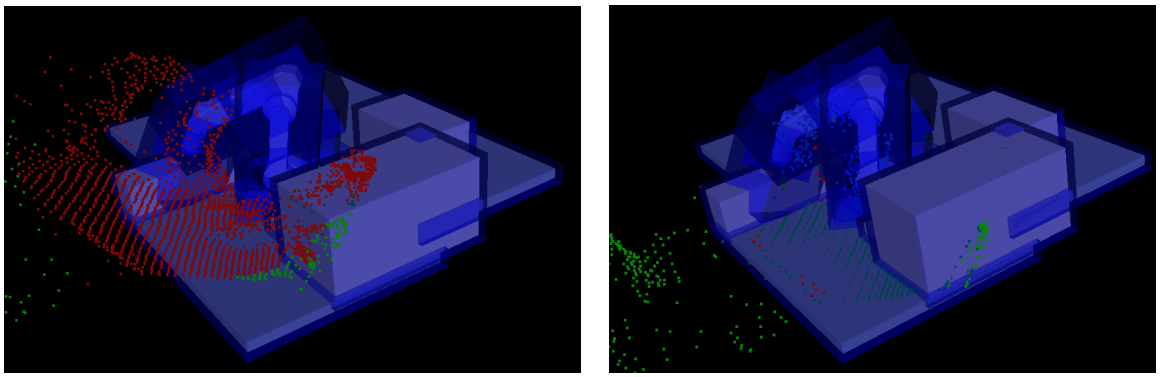


Abbildung 5.1: Abweichung der von der Kalibrierung gelieferten extrinsischen Parametern im Vergleich zu den manuell angepassten optimalen Parametern.

Rauschen in den Tiefendaten. Ein bekanntes Problem bei der Verwendung von PMD-Kameras stellt das Rauschen der erhaltenen Tiefendaten dar, das in nicht genügend reflektierenden Objekten und einer dem Funktionsprinzip zugrunde liegenden Phasenambiguität begründet ist. Dieser Sachverhalt ist in Abbildung 5.2 dargestellt und

kann in [26] lediglich mit Hilfe eines aufwändigen Stereo-Systems ausgeglichen werden.

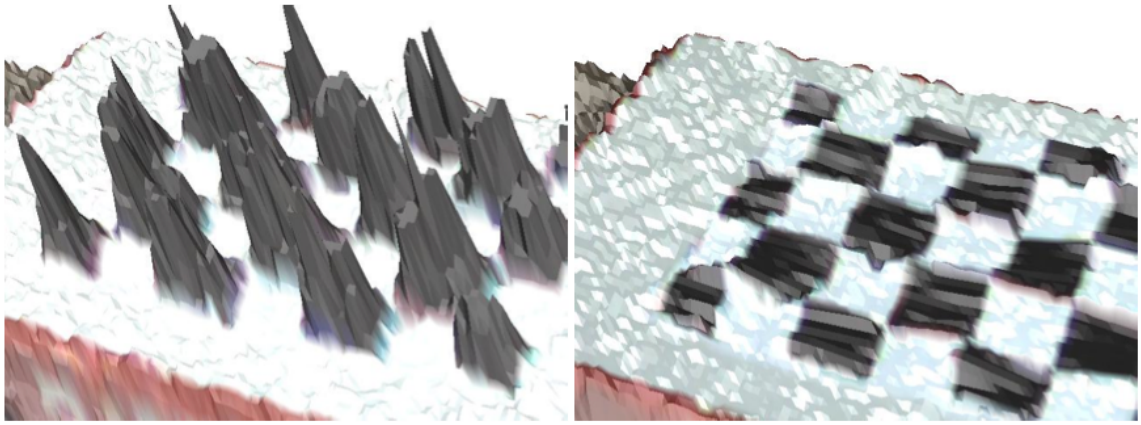


Abbildung 5.2: Rauschen in den PMD-Tiefenwerten bei dunklen Oberflächen. Eine Interpolation mit Hilfe eines zusätzlichen Stereo-Systems (rechts) kann das Ergebnis verbessern. [26]

In der Praxis bewirkt die schwarze Oberfläche des Arbeitstisches ein sehr starkes Rauschen, so dass eine korrekte Zuordnung von zum Tisch gehörenden Punkten nur mit einem sehr großen und wenig praktikablen Sicherheitsabstand realisierbar ist.

Dieses Rauschen lässt sich jedoch stark reduzieren, indem man ein weißes Tischtuch auf der Tischoberfläche ausbreitet. Abbildungen 5.3 und 5.4 zeigen, wie signifikant sich ein auf den Tisch gelegtes weißes Tuch auf die Tiefenwerte in der Praxis auswirkt.

Jedoch produzieren auch reflektierende Gegenstände wie Metalloberflächen fehlerhafte Tiefenwerte, so dass ein bedachter Umgang mit den erhaltenen Werten notwendig ist. Selbst die dunkle Oberfläche der außerhalb des gemeinsamen Arbeitsraums vor dem Tisch liegenden gewichtsempfindlichen Matten lässt die zugehörigen Punkte in den gemeinsamen Arbeitsraum wandern.

Bounding-Boxen der statischen Umgebung. Die Bounding-Box-Tests aus Abschnitt 4.3.1 erlauben eine Reduzierung der Sensordaten durch Bereinigen von Punkten, die der statischen Umgebung zuzurechnen sind.

Aufgrund des bereits dargelegten Rauschens der PMD-Tiefenwerte müssen die Bounding-Boxen des statischen Umgegungsmodells etwas vergrößert werden, so dass beispielsweise zum Roboter gehörende Punkte auch mit hoher Sicherheit dem Roboter-Modell zugeordnet werden.

Da das in Abschnitt 4.2.1 verwendete vereinfachte statische Umgebungsmodell nicht alle statischen Objekte des realen Set-Ups enthält, wurden manuelle Ergänzungen in der VRML-Datei vorgenommen, die auch in Abbildung 5.4 zu sehen sind (vgl. Abb. 4.2).

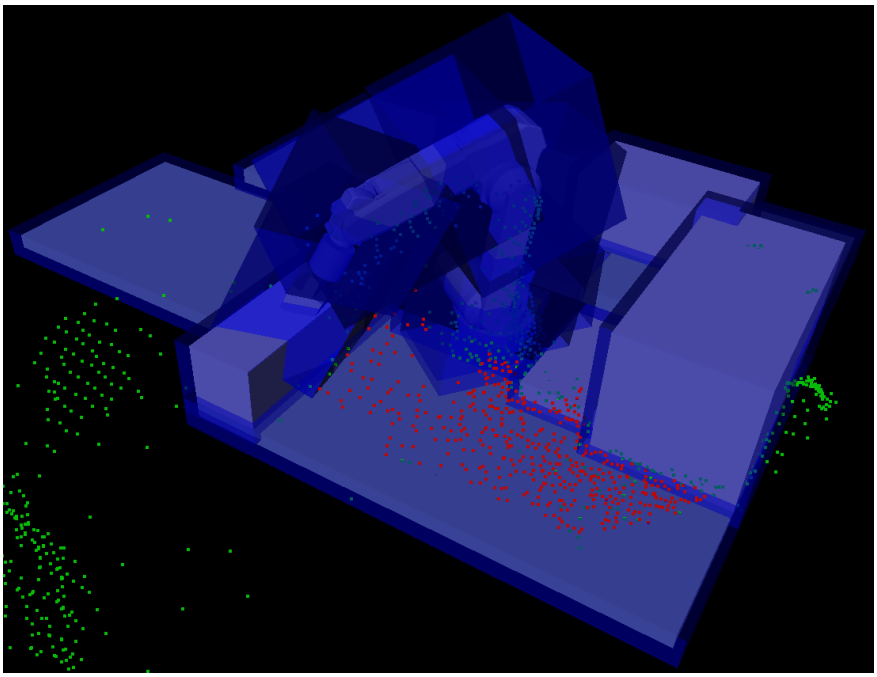


Abbildung 5.3: Signifikanter Fehler in den Tiefenwerten der zum Tisch gehörenden Punkte ohne weißes Tischtuch.

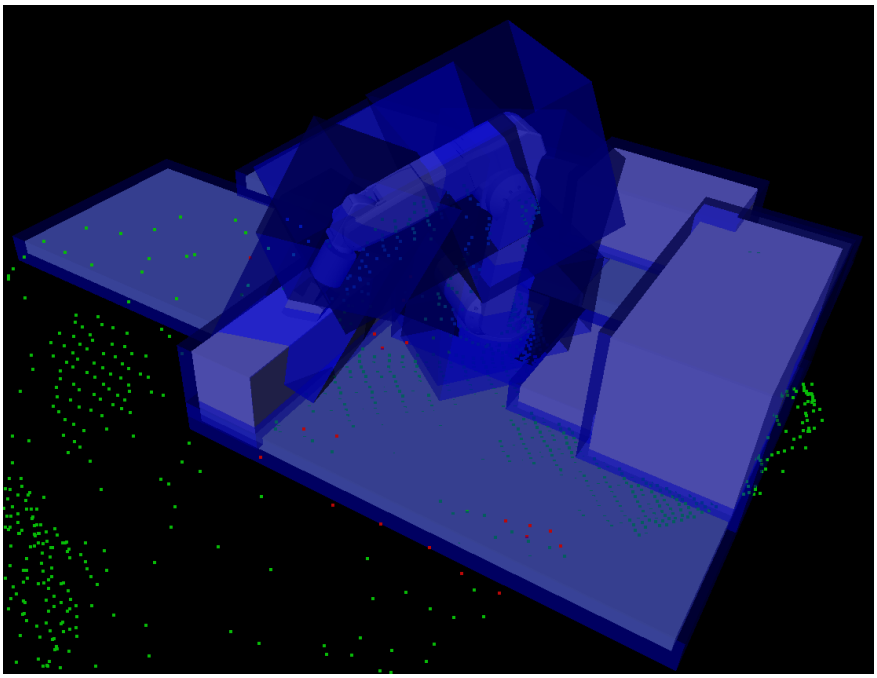


Abbildung 5.4: Wenig Rauschen in den PMD-Punkten bei Verwendung eines weißen Tischtuchs.

Die optimale Größe des Sicherheitsabstands für Bounding-Boxen des statischen Umgebungsmodells kann durch Sichtprüfung mit Hilfe der Visualisierung gefunden werden, da bekannten Objekten zugeordnete Punkte grün und unbekannte Punkte rot angezeigt werden. Somit gilt es die Anzahl der roten Punkte zu minimieren, wenn sich keine dynamischen Hindernisse im Arbeitsraum befinden. Wählt man den Sicherheitsabstand jedoch zu groß, so werden möglicherweise Punkte von unbekannten Hindernissen fehlerhaft einem statischen Objekt zugeordnet.

In der Praxis hat sich für den Arbeitstisch ein optimaler zusätzlicher Sicherheitsabstand von 2,5 cm herausgestellt, der alle dem Tisch zuzuordnenden Punkte beinhaltet. Als problematisch anzusehen ist hierbei, dass auf dem Tisch liegende Gegenstände (Boxen etc.) teilweise komplett innerhalb der erweiterten Bounding-Boxen liegen und somit nicht von der Kollisionsvermeidung berücksichtigt werden.

Für den Roboterarm wurde ein wesentlich größerer Sicherheitsabstand von 12 cm gewählt, der wegen abstehenden Kabeln sowie wegen möglichen schnellen Bewegungen des Arms nötig ist. Alle Sicherheitsabständen sind in Abbildung 5.4 bereits eingerechnet.

Clustering. Das Finden von Punkthäufungen innerhalb der bereinigten Sensordaten dient zur Reduzierung der an andere Komponenten weiterzugebenden Daten.

Beim verwendeten K-Means-Clustering-Verfahren wird die Anzahl von Clustern fest vorgegeben. Wählt man eine zu hohe Anzahl von Clustern, so nimmt der Effekt der Reduzierung der Datenmenge zunehmend ab. Vermindert man die Anzahl der Cluster, so nimmt in der Regel der durchschnittliche Abstand der Punkte eines Clusters zum zugehörigen Cluster-Zentrum zu, so dass sich auch die Cluster-Größe vergrößert. Bei einer Clusterzahl von 100 Clustern ist die Datenmenge noch klein genug für eine effiziente Weitergabe und die Cluster passen sich auch noch genügend gut an die Form der Punktwolke an.

Die Verwendung der Standardabweichung als Clustergröße erweist sich in der Praxis jedoch als problematisch, da es damit zwangsläufig Punkte gibt, die außerhalb der zur Repräsentation der Hindernisse gewählten Boxen liegen. In Abbildung 4.5 sind einige Punkte zu sehen, die sich außerhalb von Clustern befinden. Dieser Sachverhalt kann negative Auswirkungen auf die von der Kollisionsvermeidung garantierte Sicherheit haben, da dabei nur auf die Cluster und nicht auf die dazwischen liegenden Punkte geprüft wird. Ferner leidet die Effizienz der Kollisionsvermeidung unter der gewählten Anzahl von Objekten, gegen die stets geprüft werden muss.

Beim Quickhull-Algorithmus hingegen wird nur ein einzelnes Objekt ermittelt, was eine effizientere Kollisionsvermeidung ermöglicht. Die dabei berechnete konvexe Hülle, die in der Praxis eine aufwändigere Visualisierung erforderlich macht, enthält im Inneren auch ausnahmslos alle Punkte von unbekannten Objekten. Für eine höhere Sicherheit ist somit der Quickhull-Algorithmus die bessere Wahl. Bei konkaven Hindernissen kann es jedoch passieren, dass ein wesentlich größerer Raum von der Hülle eingenommen wird, als das eigentliche Objekt beansprucht. Für eine optimale Anpassung an die äußere Form von konkaven Objekten besteht ein Optimierungspotential in einer Kombination von K-Means-Clustering mit dem Quickhull-Algorithmus, bei der zuerst Cluster berechnet und anschließend deren äußere Hülle

ermittelt wird. Ein solches Verfahren hat jedoch negative Auswirkungen auf die zu veröffentlichende Datenmenge.

5.2 Performanz

Das zur Evaluation der Performanz verwendete System ist ein Core2 Duo 2,13 GHz mit 2 GB RAM, integrierter Intel Q965 Chipsatz-Grafik und einer 160 GB IDE-Festplatte. Nachfolgend soll die Geschwindigkeit der Komponente hinsichtlich unterschiedlicher Aspekte ausgewertet werden.

Maximalgeschwindigkeit des Roboters. Die maximale Geschwindigkeit eines Industrieroboter in einer Mensch-Roboter-Kooperation ist durch den Industriestandard *EN ISO 10218-1: 2006* auf 250 mm/s beschränkt, wenn Kraft und Leistung des Roboters nicht bereits vom Systemdesign her genügend beschränkt sind. Im JAHIR-Szenario bewegt sich der Roboter jedoch stets mit geringeren Geschwindigkeiten.

Ausleserate der PMD-Kamera. Um festzustellen, welche Framerate für die Komponente in Kombination mit der verwendeten PMD-Kamera maximal erreichbar ist, wird ausgewertet, wie viele Bilder maximal pro Sekunde aus der PMD-Kamera ausgelesen werden können. Dabei wurden jeweils eine feste Anzahl von Tiefenbildern geladen und die dafür benötigten Zeitdauern gemessen.

Anzahl Frames	Dauer (Sekunden)	Geschwindigkeit (fps)
1000	70,681	14,148
1000	70,260	14,233
2000	140,820	14,203
2000	140,237	14,262

Tabelle 5.1: Ausleserate der PMD-Kamera.

Aus den Messergebnissen in Tabelle 5.1 folgt, dass mit einer durchschnittlichen Ausleserate von ca. 14 fps zu rechnen ist. Dieser Wert liegt wesentlich unterhalb der im Datenblatt der PMD-Kamera angegebenen 25 fps, was auf die auf Ethernet basierende Kommunikation mit der Kamera oder auf Verzögerungen durch die vorhandenen Netzwerk-Komponenten zurückzuführen sein könnte.

Verarbeitungsrate der Komponente. Die Verarbeitungsrate misst die Anzahl der in einer bestimmten Zeitspanne verarbeiteten Frames und drückt die eigentliche Geschwindigkeit der erstellten Komponente aus. Im Allgemeinen kann man sagen, je höher die gemessene Framerate ist, umso häufiger kann der Roboter von der Komponente angesprochen und gegebenenfalls korrigiert werden.

Zum Messen der Framerate werden die Laufzeit des Programms und die Anzahl der in dieser Zeitspanne durchgeführten Verarbeitungszyklen, die in Abschnitt 4.3 beschrieben sind, protokolliert. Beim K-Means-Clustering werden außerdem die Gesamtzahl der unbekannten Punkte sowie die Anzahl der berechneten Cluster über

die gesamte Laufzeit hinweg berechnet. Das Ergebnis der Geschwindigkeitsmessung bei Verwendung des K-Means-Algorithmus ist in Tabelle 5.2 aufgelistet, während das Ergebnis für den Quickhull-Algorithmus in Tabelle 5.3 zu sehen ist. Dabei wird auch für jeden Messlauf protokolliert, ob sich ein aktiver Arbeiter im Arbeitsraum befand.

Frames	Dauer (s)	Geschwindigkeit (fps)	Ø Punkte je Frame	Ø Cluster je Frame	Ø Punkte je Cluster	Arbeiter aktiv
752	87,9868	8,54673	437,21941	52,81516	8,27829	ja
535	61,4284	8,70933	455,39626	46,09159	9,88025	ja
728	79,0908	9,20461	357,65110	45,12363	7,92603	ja
526	41,9521	12,5381	12,081749	9,39734	1,28566	nein
892	68,3379	13,0528	14,283632	11,06726	1,29062	nein
731	54,4895	13,4154	9,5827633	7,93707	1,20734	nein
760	56,5604	13,4370	13,084211	10,04211	1,30294	nein

Tabelle 5.2: Geschwindigkeit der Komponente bei Verwendung von K-Means-Clustering.

Frames	Dauer (s)	Geschwindigkeit (fps)	Ø Punkte je Frame	Arbeiter aktiv
396	42,0565	9,41589	351,14899	ja
527	50,9061	10,3524	386,17078	ja
401	36,1892	11,0806	315,20449	ja
630	51,0131	12,3498	9,55556	nein
773	60,0230	12,8784	11,47477	nein
711	54,0807	13,1470	14,99156	nein
1018	75,8783	13,4162	11,38212	nein
651	47,8289	13,6110	9,98464	nein

Tabelle 5.3: Geschwindigkeit der Komponente bei Verwendung des Quickhull-Algorithmus.

Vergleicht man die beiden Verfahren hinsichtlich der Frameraten, so sind auf den ersten Blick keine großen Unterschiede auszumachen.

Aus den beiden Tabellen kann jedoch eindeutig geschlossen werden, dass die Geschwindigkeit der Komponente bzw. die Ausführungsdauer eines einzelnen Verarbeitungszyklus direkt von der Anzahl der unbekannten Punkte im Arbeitsraum abhängt, die nach dem in Abschnitt 4.3.1 beschriebenen Bereinigen übrig bleiben. Somit ist die Verarbeitungsrate auch vom zusätzlichen Sicherheitsabstand der Bounding-Boxen abhängig, da bei größerem Sicherheitsabstand um die Objekte der statischen Umgebung weniger unbekannte Punkte übrig bleiben.

Bewegt sich ein Arbeiter im überwachten Arbeitsraum, so werden laut Messungen mindestens 300 unbekannte Punkte je Frame verarbeitet, während im leeren Arbeits-

raum durchschnittlich unter 15 unbekannte Punkte pro Frame zur Weiterverarbeitung übrig bleiben. Wenn nur wenige Punkte weiter verarbeitet werden müssen, dann sind Frameraten von über 13 fps machbar, so dass die bereits evaluierte maximale Ausleserate der PMD-Kamera von knapp über 14 Sekunden nahezu ausgereizt wird.

Mit ansteigender Anzahl von unbekannten Punkten im gemeinsamen Arbeitsraum sinkt jedoch die Framerate zunehmend. Um eine realistische Abschätzung für die durchschnittlich zu erwartende Anzahl von unbekannten Punkten und somit auch zuverlässige Aussagen über die realen Frameraten zu bekommen, sind allerdings tiefer gehende Evaluationen verschiedener realer Anwendungsszenarien von Nöten.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde die Entwicklung und Integration einer echtzeitfähigen Komponente zur aktiven Arbeitsraumüberwachung mittels Tiefenbild-Kamera im Rahmen des JAHIR-Projekts beschrieben, mit deren Hilfe vorab unbekannte Hindernisse in der Umgebung detektiert werden können. Eine einfache und problemlose Integration der Komponente in das vorhandene System wurde durch das Ice-Framework für verteilte Anwendungen ermöglicht, so dass nun eine weitere Maßnahme zur Überwachung der Roboterumgebung in diesem Szenario vorhanden ist.

Die zentrale Komponente ist dabei die unter verschiedenen Systemen von Tiefenbild-Kameras ausgewählte PMD-Kamera, die mit einer CCD-Kamera zu einem Sensorkopf kombiniert wurde. Dieser wurde so montiert, dass der Haupt-Arbeitsbereich rechts des Roboters abgedeckt ist. Mit Hilfe der CCD-Kamera wurde die PMD-Kamera extrinsisch kalibriert.

Es wurde ein Verarbeitungszyklus entwickelt, der die Sensordaten auswertet und in gefilterter Form an andere Komponenten weiterreicht. Dabei erfolgt nach dem Auslesen von Tiefendaten aus der PMD-Kamera eine Bereinigung mit Hilfe des statischen Umgebungsmodells, so dass nur noch Punkte übrig sind, die unbekannten Hindernissen in der Umgebung zuzuordnen sind.

Um diese Punkte zu Objekten zusammenzufassen, wurde ein Clustering mit K-Means integriert. Aufgrund der praktischen Schwächen dieses Verfahrens wird nun mit dem Quickhull-Algorithmus die komplexe Hülle einer Punktwolke berechnet, was ein einziges polyedrisches Objekt liefert.

Das Ergebnis des Clusterings wird über das Ice-Framework an die anderen Komponenten von JAHIR veröffentlicht, so dass die ermittelten Hindernisse auch in der Kollisionsvermeidung berücksichtigt werden. Zur Veranschaulichung wird die interne Szenenrepräsentation mit ihren statischen und dynamischen Objekten stets in Form einer dreidimensionalen Visualisierung veranschaulicht.

Während die gesetzten Ziele dieser Arbeit erreicht wurden, so bestehen noch zukünftige Erweiterungsmöglichkeiten und zu bewältigende Herausforderungen.

Da bisher nur der Bereich des Arbeitsraums überwacht wird, der rechts vom Roboter ist, kann mit Hilfe einer weiteren PMD-Kamera der überwachte Bereich vergrößert werden. Damit geht auch eine Erhöhung der Sicherheit des menschlichen Arbeiters einher, da somit das Problem von Okklusionen eingegrenzt wird. Der in dieser Arbeit gewählte Ansatz lässt ein einfaches Einbinden von weiteren Tiefenbild-Kameras in die vorhandene Systemarchitektur zu.

Durch eine Kombination der Tiefendaten der PMD-Kamera mit den Daten einer weiteren PMD-Kamera ist auch eine Stabilisierung der bisweilen sehr fehlerbehafteten Tiefeninformationen möglich.

Führt man Bilder einer CCD-Kamera mit den erhaltenen Tiefendaten zusammen, so

können Objekte nicht nur detektiert, sondern anhand vordefinierter Modelle auch erkannt werden. Anwendungsgebiete einer solchen Fusion von Sensordaten sind beispielsweise das Tracken des gesamten Körpers des Arbeiters oder eine Gestenerkennung der Hand.

Um den äußerst problematischen Aspekt der fehlerbehafteten Tiefenwerte in den Griff zu bekommen, ist eine technologische Weiterentwicklung der PMD-Kameras notwendig. Zwar kann momentan problemlos eine grobe Erkennung von Objekten im Raum erfolgen, jedoch sind für eine genauere Lokalisierung von Objekten in einem industriellen Szenario wie JAHIR, in denen durch unterschiedliche Arten und Farben von Oberflächen komplexe Farb- und Lichtverhältnisse vorherrschen, technische Verbesserungen unumgänglich. Für eine genauere Kalibrierung werden überdies PMD-Kameras mit einer höheren Auflösung benötigt.

A Anhang

A.1 Abkürzungsverzeichnis

ACIPE Adaptive Cognitive Interaction in Production Environments

CAD Computer-aided Design

CCD Charge-coupled Device

CCTfM Camera Calibration Toolbox for MATLAB

CogMaSh Cognitive Machine Shop

CoTeSys Cognition for Technical Systems

fps Frames per Second

JAHIR Joint Action for Humans and Industrial Robots

KoSy Koordinatensystem

LED Light Emitting Diode

PMD Photonic Mixer Device

RTT Round-trip-time

ToF Time-of-Flight

VRML Virtual Reality Modeling Language

XML-RPC Extensible Markup Language Remote Procedure Call

A.2 Frameworks

Zur Implementierung der Komponente wurden Frameworks verwendet, die zum Teil bereits in anderen Komponenten von JAHIR benutzt wurden, da somit bereits vorhandener Code wiederverwendet bzw. als Beispiel herangezogen werden konnte. Nachfolgend sind die wichtigsten Frameworks, die im Rahmen dieser Arbeit eingesetzt wurden, aufgeführt.

ZeroC Ice: Die *Internet Communications Engine (Ice)* von ZeroC [7] ist eine objektorientierte Middleware, die das Erstellen von verteilten Anwendungen wesentlich erleichtert, indem es für den Programmierer alle Interaktionen mit unteren Ebenen der Netzwerk-Programmierschnittstellen übernimmt. Dadurch kann man sich auf die

Programmierung der Anwendungslogik konzentrieren. Ferner wird eine mühelose Kommunikation von auf unterschiedlichen Plattformen laufenden und in unterschiedlichen Programmiersprachen geschriebenen Anwendungen ermöglicht.

Coin3D und SoQT: *Coin3D* [27] ist eine objektorientierte 3D-Grafik-API für eine erleichterte OpenGL-Programmierung auf höherer Ebene. Mit Hilfe von *SoQt* [28] können Coin3D-Visualisierungen direkt in mit *Qt* [29] erstellte Benutzeroberflächen integriert werden.

Robotics Library: Die am *Lehrstuhl für Robotik und eingebettete Systeme* der TU München entwickelte *Robitics Library* [30] stellt ein Open Source-Framework zur Entwicklung von Robotik-Anwendungen dar und ist ein wesentlicher Bestandteil vieler Komponenten von JAHIR. Diese Bibliothek stellt auf C++-Standards basierende kompatible Datenstrukturen, Implementierungen von Standard-Algorithmen (Winkel-Konvertierung, Kinematiken, Bewegungsplanung, etc.) sowie Abstraktionen von geläufiger Roboter-Hardware zur Verfügung.

CF-Framework: Das *CF-Framework* wurde ebenso wie die Robotics Library am *Lehrstuhl für Robotik und eingebettete Systeme* der TU München entwickelt und hat Computer Vision als Hauptanwendungsgebiet. Mit dieser Bibliothek ist beispielsweise auch ein komfortabler Zugriff auf verschiedene Arten von Kameras (PMD-Kameras, Webcams, etc.) möglich.

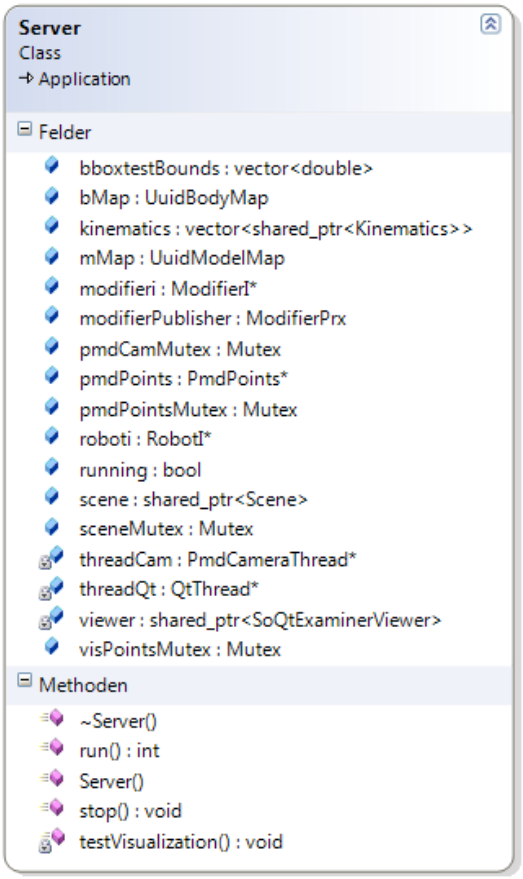
OpenCV: Die weit verbreitete Bibliothek *Open Source Computer Vision Library (OpenCV)* [31] stellt eine Vielzahl von Algorithmen für die Bildverarbeitung und maschinelles Lernen bereit. Dazu zählen unter anderem Algorithmen zur Bildtransformation, Segmentierung, Filter (z.B. Sobel, Canny), Clustering-Verfahren (z.B. K-Means-Algorithmus) sowie Funktionen für die Kamera-Kalibrierung.

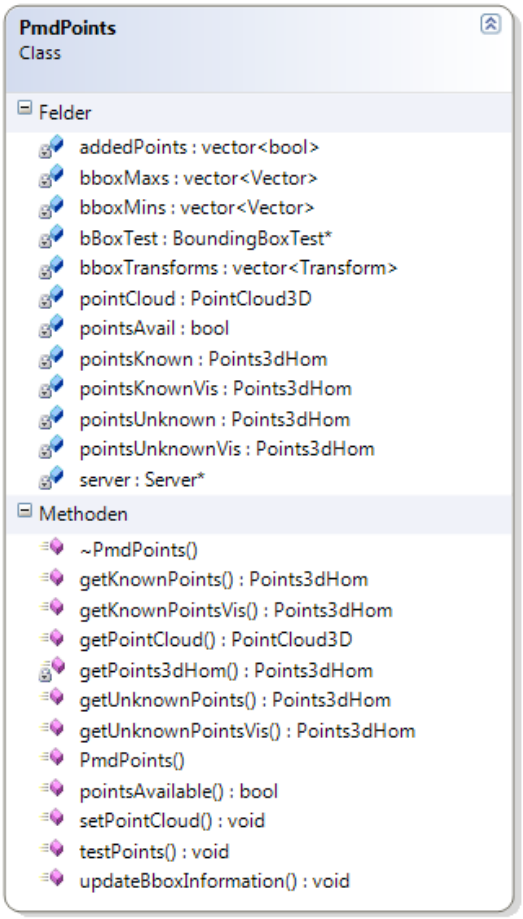
A.3 Nutzwertanalyse von Tiefenbild-Kameras

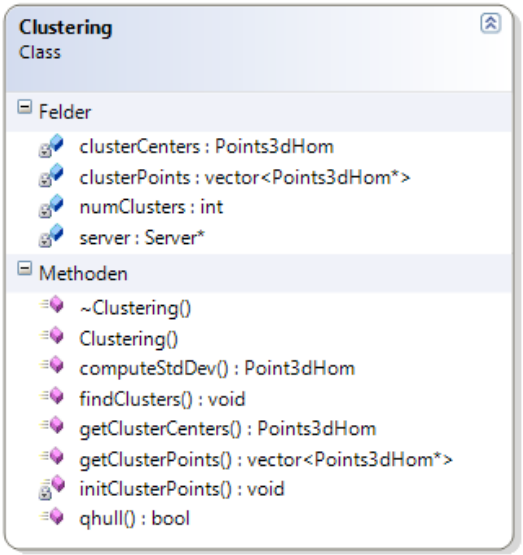
Ziel	Echtzeit-fähigkeit	Integrations-aufwand	Erfassung beliebiger Objekte	Kosten	
Gewichtung	0,5	0,15	0,2	0,15	Nutzwert
Stereo Vision	0	1	0	1	0,3
3D-Laserscanner	0	0	1	0	0,2
PMD-Kamera	1	1	1	0,5	0,925

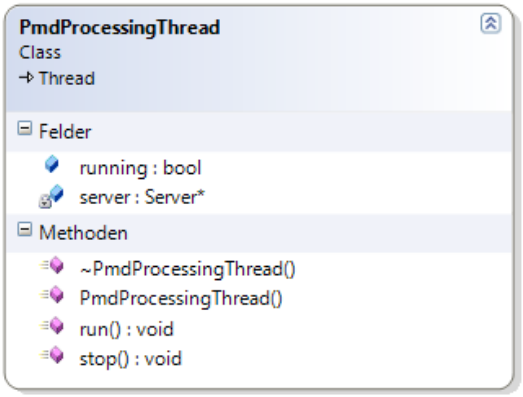
Tabelle A.1: Nutzwertanalyse von Tiefenbild-Kameras.

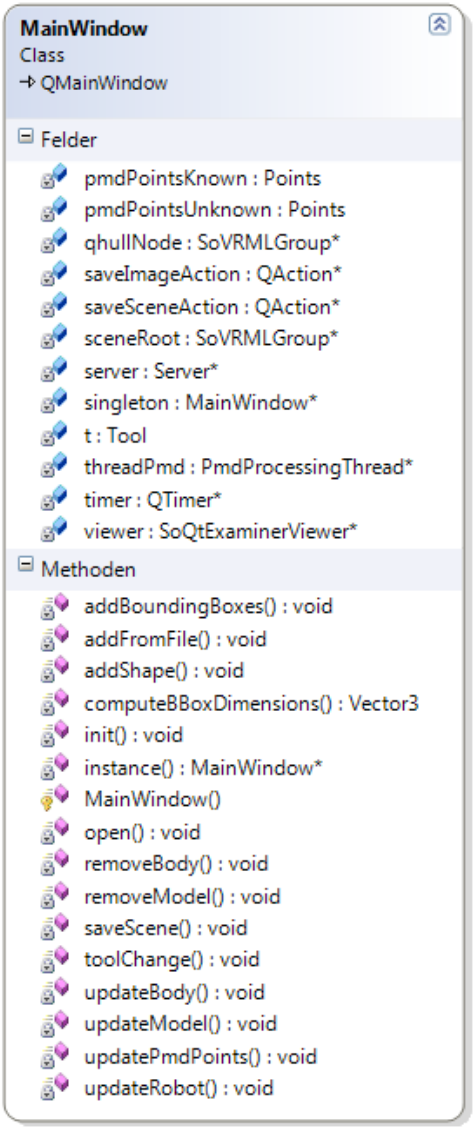
A.4 UML-Klassendiagramme

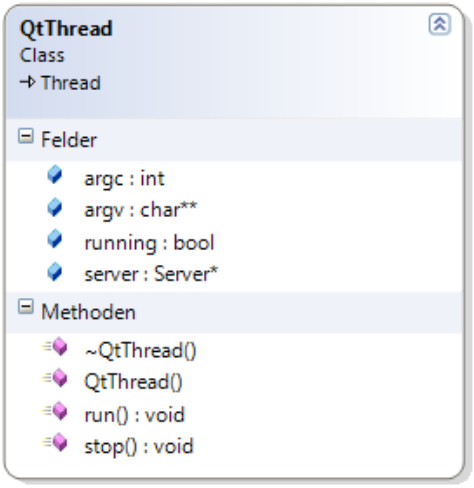


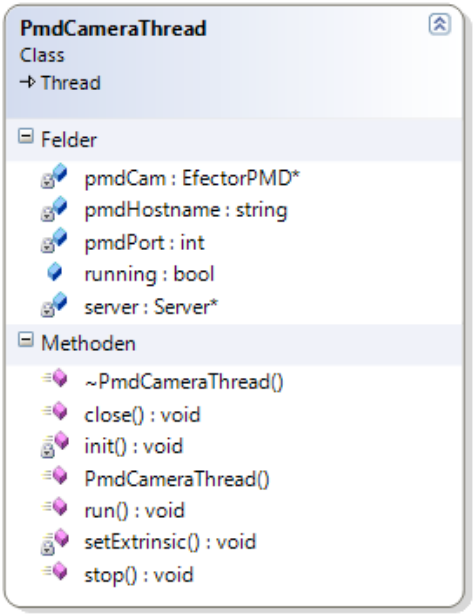


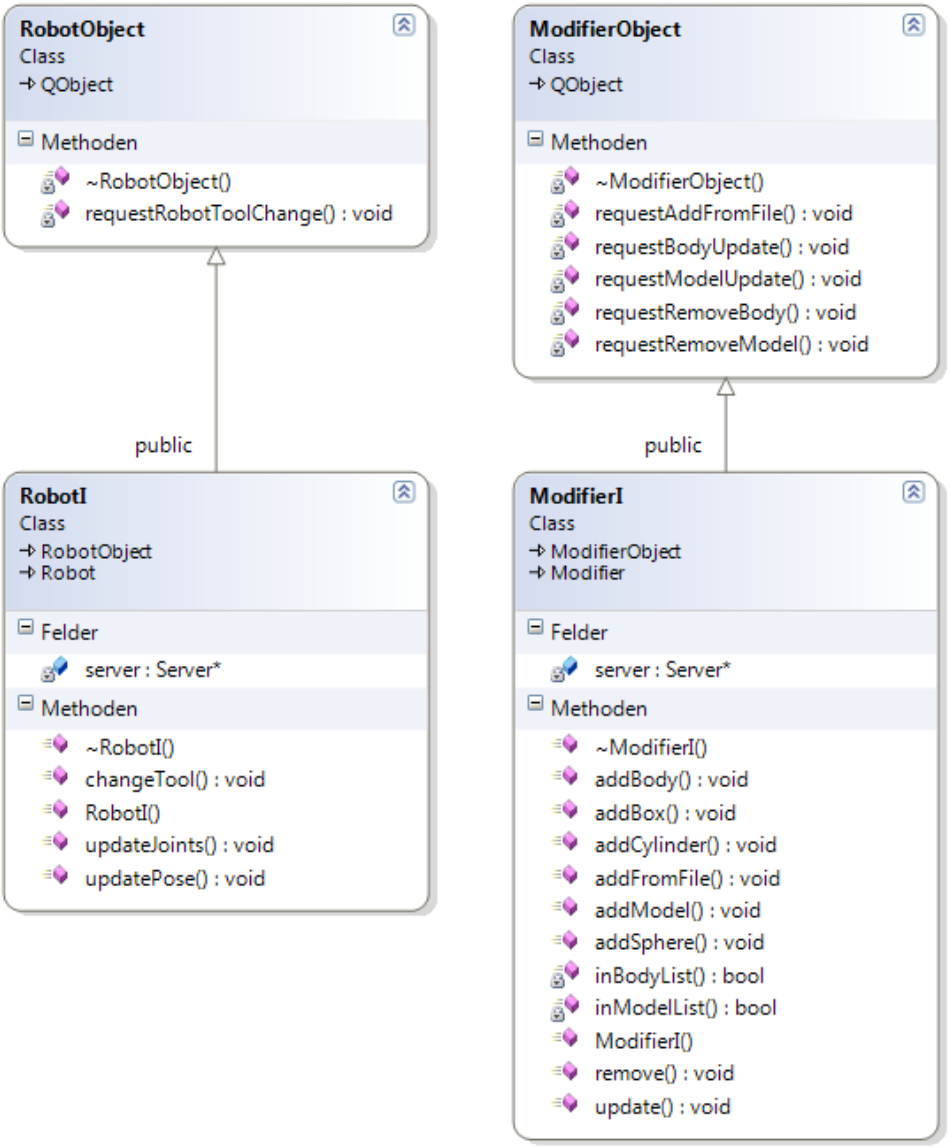





MainWindow
Class
QMainWindow
Felder
pmdPointsKnown : Points
pmdPointsUnknown : Points
qhullNode : SoVRMLGroup*
saveImageAction : QAction*
saveSceneAction : QAction*
sceneRoot : SoVRMLGroup*
server : Server*
singleton : MainWindow*
t : Tool
threadPmd : PmdProcessingThread*
timer : QTimer*
viewer : SoQtExaminerViewer*
Methoden
addBoundingBoxes() : void
addFromFile() : void
addShape() : void
computeBBoxDimensions() : Vector3
init() : void
instance() : MainWindow*
MainWindow()
open() : void
removeBody() : void
removeModel() : void
saveScene() : void
toolChange() : void
updateBody() : void
updateModel() : void
updatePmdPoints() : void
updateRobot() : void


QtThread
Class
Thread
Felder
argc : int
argv : char**
running : bool
server : Server*
Methoden
~QtThread()
QtThread()
run() : void
stop() : void


PmdCameraThread
Class
Thread
Felder
pmdCam : EfectorPMD*
pmdHostname : string
pmdPort : int
running : bool
server : Server*
Methoden
~PmdCameraThread()
close() : void
init() : void
PmdCameraThread()
run() : void
setExtrinsic() : void
stop() : void


RobotObject
Class
QObject
Methoden
~RobotObject()
requestRobotToolChange() : void
ModifierObject
Class
QObject
Methoden
~ModifierObject()
requestAddFromFile() : void
requestBodyUpdate() : void
requestModelUpdate() : void
requestRemoveBody() : void
requestRemoveModel() : void
public
public
RobotI
Class
RobotObject
Robot
Felder
server : Server*
Methoden
~RobotI()
changeTool() : void
RobotI()
updateJoints() : void
updatePose() : void
ModifierI
Class
ModifierObject
Modifier
Felder
server : Server*
Methoden
~ModifierI()
addBody() : void
addBox() : void
addCylinder() : void
addFromFile() : void
addModel() : void
addSphere() : void
inBodyList() : bool
inModelList() : bool
ModifierI()
remove() : void
update() : void

Literaturverzeichnis

[1] *JAHIR - Joint-Action for Humans and Industrial Robots*. http://www6.in.tum.de/Main/ResearchJahir, 2010

[2] *JAHIR - Joint Action for Humans and Industrial Robots*. http://www.iwb.tum.de/JAHIR___Joint_Action_for_Humans_and_Industrial_Robots.html, 2010

[3] *CoTeSys - Cognition for Technical Systems*. http://www.cotesys.org/, 2010

[4] FISCHER, M. ; HENRICH, D.: 3D collision detection for industrial robots and unknown obstacles using multiple depth images. In: *Advances in Robotics Research* (2009), S. 111–122

[5] LENZ, C. ; NAIR, S. ; RICKERT, M. ; KNOLL, A. ; RÖSEL, W. ; GAST, J. ; WALLHOFF, F.: Joint-action for humans and industrial robots for assembly tasks. In: *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication*, 2008, S. 130–135

[6] WALLHOFF, F. ; BLUME, J. ; BANNAT, A. ; RÖSEL, W. ; LENZ, C. ; KNOLL, A.: A skill-based approach towards hybrid assembly. In: *Advanced Engineering Informatics* 24 (2010), Nr. 3, S. 329–339

[7] *ZeroC Ice*. http://www.zeroc.com/, 2010

[8] GIULIANI, M. ; LENZ, C. ; MÜLLER, T. ; RICKERT, M. ; KNOLL, A.: Design Principles for Safety in Human-Robot Interaction. In: *International Journal of Social Robotics* (2010), S. 1–22

[9] HUSSMANN, S. ; RINGBECK, T. ; HAGEBEUKER, B.: A Performance Review of 3D TOF Vision Systems in Comparison to Stereo Vision Systems. In: *Stereo Vision* (2008)

[10] LANGE, R. ; SEITZ, P.: Solid-state time-of-flight range camera. In: *IEEE Journal of Quantum Electronics* 37 (2001), Nr. 3, S. 390–397

[11] RINGBECK, T. ; HAGEBEUKER, B.: A 3D Time of flight camera for object detection. In: *Optical 3-D Measurement Techniques* (2007)

[12] PRUSAK, A. ; ROTH, H. ; SCHWARTE, R. u. a.: Application of 3D-PMD Video Cameras for Tasks in the Autonomous Mobile Robotics. In: *16th IFAC World Congress, Prague, Czech Republic, from July* Bd. 4, 2005

[13] TÖNNIES, K.D.: *Grundlagen der Bildverarbeitung*. Pearson Studium, 2005

[14] ERICSON, C.: *Real-time collision detection*. Morgan Kaufmann, 2005. – 66–69 S.

[15] BARBER, C.B. ; DOBKIN, D.P. ; HUHDANPAA, H.: The Quickhull Algorithm for Convex Hulls. In: *ACM Transactions on Mathematical Software (TOMS)* 22 (1996), Nr. 4, S. 469–483

[16] THIEMERMANN, S.: Direkte Mensch-Roboter-Kooperation in der Kleinteilmontage mit einem SCARA-Roboter. In: *IPA-IAO-Bericht* (2005)

[17] PILZ GMBH & CO. KG: *Patent DE 10 2006 057 605 A1: Verfahren und Vorrichtung zum Überwachen eines dreidimensionalen Raumbereichs*. 2006

[18] SOM, F.: Sichere Steuerungstechnik für den OTS-Einsatz von Robotern. In: *IPA* (2005)

[19] WINKLER, B.: Safe Space Sharing Human-Robot Cooperation Using a 3D Time-of-Flight Camera. In: *International Robots and Vision Show* (2007)

[20] *ifm efector O3D200: Datenblatt*. `http://www.ifm-electronic.com/obj/2_O3D200_d_09_n.pdf`, 2010

[21] HARTLEY, R. ; ZISSERMAN, A.: *Multiple view geometry in computer vision*. Cambridge University Press, 2003. – 153–157 S.

[22] ZHANG, Z.: A flexible new technique for camera calibration. In: *IEEE Transactions on pattern analysis and machine intelligence* 22 (2000), Nr. 11, S. 1330–1334

[23] *Camera Calibration Toolbox for Matlab*. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2010

[24] *Camera Calibration Toolbox for Matlab: Calibrating a stereo system, stereo image rectification and 3D stereo triangulation*. `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example5.html`, 2010

[25] *Qhull Code*. `http://www.qhull.org/`, 2010

[26] HAHNE, U. ; ALEXA, M.: Depth Imaging by Combining Time-of-Flight and On-Demand Stereo. In: *Dynamic 3D Imaging* (2009), S. 70–83

[27] *Coin3D*. `http://www.coin3d.org/`, 2010

[28] *SoQt*. `http://doc.coin3d.org/SoQt/`, 2010

[29] *Qt - Cross-platform application and UI framework*. `http://qt.nokia.com/`, 2010

[30] *Robotics Library*. `http://sourceforge.net/apps/mediawiki/roblib/`, 2010

[31] *OpenCV*. `http://opencv.willowgarage.com/wiki/`, 2010