Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterthesis

Vida Ostadzadeh

Entwicklung eines Prototyps zur steuerungsbasierten Kollisionsvermeidung von komplexen 3D Objekten

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Vida Ostadzadeh

Entwicklung eines Prototyps zur steuerungsbasierten Kollisionsvermeidung von komplexen 3D Objekten

Masterthesis eingereicht im Rahmen der Masterprüfung
im Masterstudiengang Automatisierung
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Robert Heß
Zweitgutachter : Prof. Dr. rer. nat. Annabella Rauscher-Scheibe

Abgegeben am 4. Januar 2016

Vida Ostadzadeh

Thema der Masterthesis

Entwicklung eines Prototyps zur steuerungsbasierten Kollisionsvermeidung von komplexen 3D Objekten

Stichworte

Computertomografie, 3D-Objeket-Repräsentation, Bounding Volume, Oriented Bounding Box, Oriented Bounding Box Hierarchie, Minkowski Summe, Speicherprogrammierbare Steuerungen, Automation Studio

Kurzzusammenfassung

Die vorliegende Arbeit beschreibt den Entwicklungsprozess eines Prototyps zur steuerungsbasierten Kollisionserkennung von komplexen 3D Objekten in dynamischen Umgebungen mit Hilfe von 3D Objekt-Repräsentation.

Vida Ostadzadeh

Title of the paper

Development of a prototype to the control-based collision detection of complex 3D objects

Keywords

X-ray computed tomography, 3D-Objcet-Representation, Bounding Volume, Oriented Bounding Box, Oriented Bounding Box Hierarchy, Minkowski Sum, Programmable Logic Controller, Automation Studio

Abstract

The present work describes the process of development of a prototype to the control-based collision recognition of complex 3D objects in dynamic environments by 3D object representation.

Danksagung

Zuallererst möchte ich mich ganz herzlich bei Prof. Dr. Ing. Robert Heß und Prof. Dr. Ing. Annabella Rauscher-Scheibe für ihre freundliche und kompetente Unterstützung bedanken.

Ein besonderer Dank gilt meinen Betreuern bei der Firma YXLON International GmbH, Herrn Felix Woelk und Herrn Jan Spalding, die mich während dieser Bachelorarbeit fachlich und organisatorisch unterstützt haben.

Eine kollegiale Unterstützung erhielt ich von Herrn Robert Lewis und Lars Holzberger , den ein spezieller Dank dafür zukommt.

Weiterhin möchte ich aus der Firma YXLON für Hilfe und Unterstützung noch folgenden Personen danken:
Christian Züdel, Ditmeier Ewert, Carina Wüst.

Für Ihre Unterstützung danke ich recht herzlich Frau Nadiene Gohert, Herrn Dominique Dauch und Frau Ines Hilbert.

Ein besonderer Dank geht an meine Freundin Frau Christina Hinderer, die gewissenhaft die Lesekorrektur der Masterarbeit übernommen hat.

Abkürzungsverzeichnis

AABB Axis-aligned Bounding Boxes

B&R Bernecker + Rainer Industrie-Elektronik Ges.m.b.H.

BV Bounding Volume

BVH Bounding Volume Hierarchy

CAD Computer-Aided Design. Deutsch: Rechnerunterstütztes Konstruieren

CT Computertomografie

DOD Detector-Object-Distance

DOP Discrete Oriented Polytopes

FF Fine Focus

FOD Focus-Object-Distance

GUI Graphical User Interface

GUIDE Graphical User Interface Design Environment

HMI Human-Machine Interface. Deutsch: Mensch-Maschine-Schnittstelle

k-DOP k-Discrete Oriented Polytopes

OBB Oriented Bounding Boxes

OMG Object Management Group

RAPID Robust and Accurate Polygon Interference Detection

SPS Speicherprogrammierbare Steuerung. Englisch: PLC Programmable Logic Controller

STL STereoLithography

svd Singular value decomposition

UDP User Datagram Protocol

UML Unified Modeling Language

VIIDA Very Intelligent Interference Detection Algorithm

Inhaltsverzeichnis

1. Einführung

1.1. Motivation

Für die Erhöhung der Sicherheitsleistung der Bauteile in industriellen Anwendungen hat sich das zerstörungsfreie Prüfverfahren der Computertomografie (CT) als Standardverfahren etabliert.
Die zunehmende Nachfrage für industrielle Computertomografie stellt hohe Anforderungen an die Technologie der Prüfsysteme und an deren Hersteller. Um diesen Anforderungen gerecht zu werden, setzen Unternehmen leistungsfähige Rechner ein, um die Darstellung und Analyse immer größerer Mengen an Daten zu ermöglichen und damit die Präzision der Prüfsysteme laufend zu erhöhen. Darüber hinaus werden für die Automatisierung der Prüfsysteme speicherprogrammierbare Steuerungen SPS[1] eingesetzt.
Die Bandbreite der modernen CT-Systeme auf dem Markt ist heutzutage bemerkenswert. Sie reicht vom Groß-Scanner für Objekte mit großen Dimensionen wie z.B. Flugzeugteilen, bis zu μCT-Scannern mit einer Auflösung von bis zu 1μm für die zerstörungsfreie Untersuchung aus dem mikroelektronischen Anwendungsbereich.
Insbesondere im mikroelektronischen Bereich stellt die bestmögliche Auflösung bzw. Vergrößerung kleinster Objekte ein entscheidendes Kriterium für die nutzbringende Anwendung von CT-Scannern dar. Die Vergrößerung und die Auflösung errechnen sich maßgeblich über geometrische Größen. Die Direktvergrößerung des Prüfobjekts ergibt sich aus dem Strahlensatz. Wie in der Abbildung 1.1 zu erkennen ist, ist das Auflösungsvermögen des CTs von der Position des Prüfobjekts, der Röntgenquelle und des Detektors abhängig.

Für eine optimale Auflösung muss die Röntgenröhre also so nah wie möglich an das Prüfobjekt positioniert werden. Das Befahren der Anlagenachsen ist unter diesen Bedingungen nicht möglich, ohne einen sicheren Schutz der Objekte vor einer Kollision mit der Röntgenröhre bzw. mit dem Detektor zu gewährleisten. Kollisionserkennung in einer CT-Anlage ist daher ein denkwürdiges Thema bei der YXLON International GmbH, die sich auf die Herstellung industrieller CT-Systeme spezialisiert hat, und in deren Zusammenarbeit diese Masterarbeit verfasst wurde.

Meine Masterarbeit zeigt, wie sich eine automatisierte, echtzeitfähige Kollisionsvermeidung für eine industrielle CT-Anlage realisieren lässt.

[1] SPS: ist ein Gerät für die Steuerung einer Maschine. Englisch: PLC

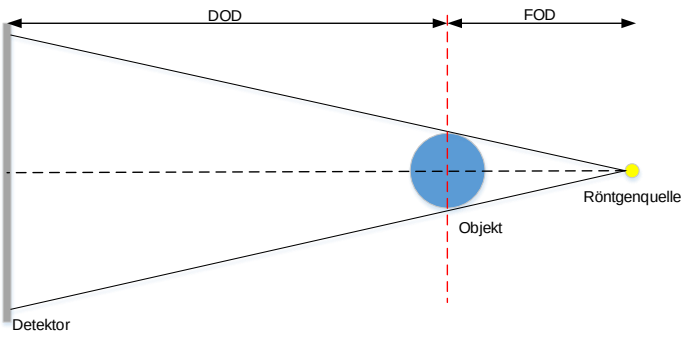


Abbildung 1.1.: 3D CT-Aufnahmegeometrie. Dabei bezeichnet FOD die Fokus-Objekt-Distanz und DOD den Detektor-Objekt-Distanz.

1.2. Aufgabenstellung

Die Firma YXLON International GmbH - im Folgenden nur YXLON genannt - produziert automatisierte Röntgensysteme zur Analyse von Produktionsteilen für industrielle Anwendungen.
Für die Bewegung des Objektes in der CT-Anlage während des Röntgenvorgangs kommt in den YXLON-Systemen eine SPS zum Einsatz. Diese verfügt über einen rudimentären Kollisionsschutz, um die Maschinenteile (Röhre, Objektträger und Detektor) und das Prüfobjekt vor Beschädigungen zu schützen.
Die dazu notwendigen Berechnungen werden derzeit noch auf der SPS durchgeführt und arbeiten mit einer einfachen Abstraktion der Kollisionskörper als grobe Quader.
Eine weitere Kollisionsschutzlösung ist in der PC Steuerungssoftware realisiert. Dazu werden zum einen die CAD-Zeichnungen der eingesetzten Systembestandteile (Röntgenröhre, Werkstückaufspannung und Detektor) vom PC interpretiert, zum anderen wird per Kamera eine Punktewolke des Prüfobjektes erfasst und vom PC interpretiert. Dieses System hat sich aber in der Praxis als nicht echtzeitfähige erwiesen und Kollision zugelassen.
Um die Echtzeitfähigkeit der YXLON-Systeme zu erhöhen, ist ein neuer Kollisionsvermeidungsalgorithmus erforderlich. Diesen zu entwickeln und auf der SPS zu realisieren ist Ziel dieser Masterarbeit.

1.3. Darstellung der Vorgehensweise

Wie bei allen wissenschaftlichen Arbeiten ist eine gründliche Literatur-Recherche ein essentieller Bestandteil auch dieser Arbeit. Es wurde eine Kombination von zwei Methoden der Literatur-Recherche, die sogenannte Systematische Recherche und das Schneeballsystem, durchgeführt. Unter Berücksichtigung der zeitlichen und fachlichen Aspekte wurde diese Recherche auf folgende Bereiche begrenzt:

- Computergrafik
- Robotik

Bei der Suche nach einem geeigneten Kollisionserkennungsalgorithmus wurden am Anfang die Softwareanforderungen und die Hardwareeinschränkungen außer Acht gelassen und der Fokus nur auf den Algorithmus gerichtet. Ziel ist durch einen modularen Aufbau der Software-Komponenten die Hardwareeinschränkung - Vorrangig die eingeschränkte Rechenleistung der SPS [2] - zu kompensieren.

[2] Vorrangig die eingeschränkte Rechenleistung der SPS. Siehe Unterkapitel 2.3

2. Hintergrund und technische Grundlagen

Dieses Kapitel enthält eine kurze Vorstellung der Unternehmen, die für diese Masterarbeit relevant sind. Außerdem nennt es die Produkte der Firma YXLON, in deren Auftrag diese Masterarbeit erstellt wurde. Darüber hinaus werden die Firma Bernecker + Rainer Industrie-Elektronik Ges.m.b.H., kurz B&R, genannt und ihre für diese Arbeit relevanten Hardware- und Softwareprodukte vorgestellt. Die Firma B&R ist der Lieferant von Steuerungskomponenten für die Firma YXLON.
Des Weiteres enthält dieses Kapitel eine genauere Beschreibung des letzten Entwicklungsstandes der aktuellen Lösungsansätze zur Kollisionsvermeidung bei YXLON.

2.1. Vorstellung der YXLON International GmbH

YXLON ist der weltweit führende Technologiekonzern für Röntgensysteme zur industriellen Anwendung.

Sie entwickelt und fertigt Röntgenprüfsysteme für die unterschiedlichsten Anwendungen und Branchen. Ihre Wurzeln reichen bis zu Carl Heinrich Florenz Müller, dem Hersteller der ersten Röntgenröhre, und zu Wilhelm Conrad Röntgen, dem Entdecker der Röntgenstrahlen im Jahr 1895 zurück. Gegründet wurde YXLON 1998 (YXLON, 2014).
YXLON gehört seit Januar 2007 zur COMET Holding AG (Schweiz). Weltweit beschäftigt YXLON mit ihrem Hauptsitz in Hamburg rund 350 Mitarbeiter.
Aufgrund der hohen Anforderungen an die Laufsicherheit und Qualität der Produkte steht die Verwendung neuester Technologien im Vordergrund.

2.1.1. Kurzvorstellung der Produkte

YXLON stellt hochwertige Röntgenkomponenten und Röntgensysteme für die industrielle Röntgenprüfung her. Sie bietet ganze Systemlösungen inklusive Software und Service an.

Es werden unter anderem folgende Produkte von YXLON hergestellt:

- Röntgensysteme zur Durchleuchtungs- und CT-Prüfung,

- Computertomografiesysteme zur Detektion von Porosität bei Gießereiprodukten,
- Software zur Bearbeitung von Röntgenbildern,
- Röntgensysteme zur Prüfung von Leichtmetallrädern und zur Reifenprüfung.

Die im Rahmen dieser Masterarbeit entwickelten Algorithmen und Modelle beziehen sich in der ersten Linie auf die industriellen Computertomographen von YXLON - speziell auf die Modelle YXLON FF20 CT und YXLON FF35 CT.

Abbildung 2.1.: Computertomografiesystem YXLON FF20 CT
(YXLON, 2015d)

Das YXLON FF20 CT System bietet die besten Prüfergebnisse mit höchster Auflösung. Es ermöglicht daher zum Beispiel die Prüfung von kleinsten Elektronikbauteilen. Grundsätzlich wurde das YXLON FF20 CT für die Anwendung in Forschung und Entwicklung, zur Fehleranalyse und Prozessüberwachung und zur CT-Metrologie[1] entwickelt. Das System ist mit einer 190 kV FeinFokus-Transmissionsröhre mit Wasserkühlung ausgerüstet und bietet eine 3D-Darstellung von kleinsten - bis 1 mm - Strukturen (YXLON, 2015d).

Das YXLON FF35 CT System besitzt eine etwas geringere Auflösung und wurde zur Prüfung größerer Objekte, vorrangig in der Automobil-, Elektronik- und Luftfahrtindustrie, sowie für die Medizintechnik entwickelt. Das YXLON FF35 CT besitzt ebenfalls eine 190 kV FeinFocus-Transmissionsröhre und ist zusätzlich mit einer leistungsstarken FeinFocus-Direktstrahlröhre ausgestattet.

[1] Metrologie ist die Wissenschaft und Technik des Messens und ihre Anwendung.

2.2. Vorstellung der Firma B&R

Die Firma B&R Automation, im Folgenden nur B&R genannt, ist ein Hersteller von Speicherprogrammierbaren Steuerungen (SPS) und liefert auch an die Firma YXLON.

Das Unternehmen B&R bietet zahlreiche Produkte im Bereich Industrielle Automatisierungstechnik an. Dazu gehören z.B. SPS, Industrie-PCs, Planetengetriebe, Antriebstechnik und Automatisierungssoftware.
Die Automatisierungssoftware Automation Studio wird eingesetzt, um Industrieanlagen auf Basis des Windows-Betriebssystems zu automatisieren. Die Kommunikation mit den Komponenten erfolgt über ein schnelles Ethernet, "POWERLINK". Die POWERLINK-Technologie erfüllt die Echtzeitanforderungen. Bei YXLON kommen die PC-basierten Steuerungen sowie I/O-Komponenten von B&R zum Einsatz.

2.3. Stand der Technik YXLON

2.3.1. Systemarchitektur der FF-Anlagen

Die Computertomografiesysteme von YXLON werden mit einem Industrie-PC für Visualisierung (HMI), sowie einer oder mehreren Steuerungen (PLC) automatisiert. Die folgende Abbildung stellt die Systemarchitektur der hier betrachteten FeinFocus-Anlage dar.

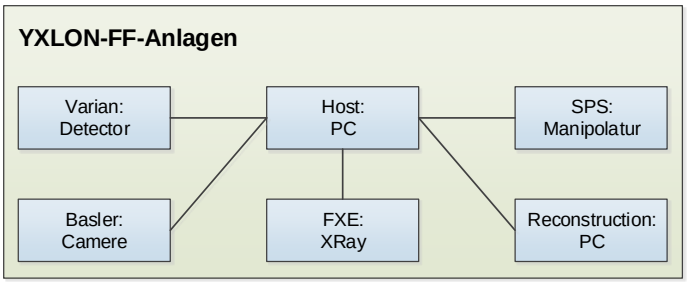
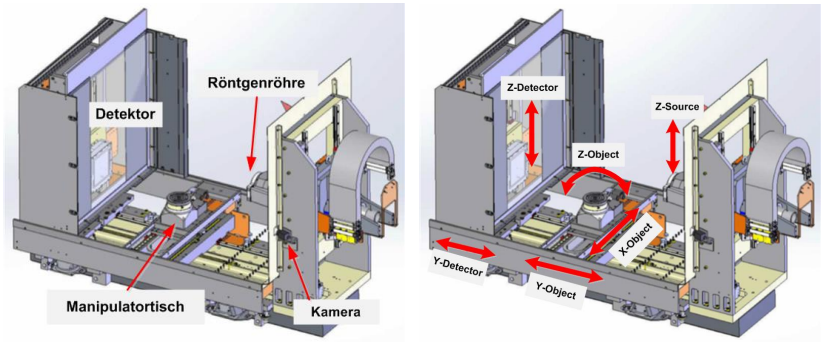


Abbildung 2.2.: Systemarchitektur der FF-Anlagen. Vgl. (YXLON, 2015c)

Der Host-PC bildet die zentrale Einheit des Gesamtsystems. Er ist für das Zusammenspiel der Komponenten zuständig. Der Varian-Detector ist das Bildgebende System, welches einen Liveröntgen-Stream erzeugt. Die Basler-Kamera ist die Kabinenkamera und für den Kollisionsschutz[2] verantwortlich. Die Anzahl der Kameras ist variabel. Die FXE-Xray (die Röhre) ist für

[2] Nähere Informationen dazu finden sich in Abschnitt 2.3.3.

die Röntgensteuerung der Röhre zuständig. Der Reconstruction PC ist auf die Berechnung von CT-Volumen-Daten spezialisiert. Der SPS-Manipulator ist die speicherprogrammierbare Steuerung der Anlage. Die Abbildung 2.3 zeigt das Innere der Anlage.



(a) Systemkomponenten der FF20-CT (b) Bewegungsrichtungen der Achsen

Abbildung 2.3.: Innenraum der Anlage. Vgl. (YXLON, 2015a)

2.3.2. Steuerung der FF-Anlagen

YXLON verwendet als Steuerung für ihre FF-Anlagen die echtzeitfähige Zentraleinheit X20CP1585 der bereits erwähnten Firma B&R.

Die X20CP1585 besitzt einen hochleistungsfähigen ATOM Prozessor mit 1 GHz Taktrate und 256 MByte DDR2-SDRAM Arbeitsspeicher. In ihrer Basisausstattung verfügt sie über USB, Ethernet, POWERLINK[1] und eine auswechselbare CompactFlash Speicherkarte. Trotz der Gigabit-fähigen Standard-Ethernet Schnittstelle stellt die Zentraleinheit für zusätzliche Echtzeitnetzwerkperformance die OnBoard POWERLINK Schnittstelle zur Verfügung. Eine der wichtigsten Eigenschaften der Steuerungen ist ihr modularer Aufbau, d.h., dass man sie je nach Anwendungsfall und Aufgabenstellung mit anderen Zusatzkomponenten erweitern kann. Dadurch lassen sich flexibel unterschiedliche Bus- und Netzwerksysteme in das X20 System integrieren. Die folgende Abbildung zeigt eine X20CP1585 Steuerung in ihrem Grundmodell.

[1] POWERLINK ist ein Echtzeit-Ethernet, welches ursprünglich von der Firma B&R entwickelt wurde.

Abbildung 2.4.: Zentraleinheit X20CP1585 (YXLON, 2015b)

2.3.3. Kollisionsvermeidung

Um eine möglichst hochauflösende Abbildung der Prüfobjekte zu erreichen, müssen Objekt und Röntgenröhre einen möglichst geringen Abstand haben. Um dies auch für unregelmäßig bzw. asymmetrisch geformte Objekte zu gewährleisten, sind bei den FF-Systemen von YXLON sowohl die Prüfobjekte, als auch die Detektor-Röhren beweglich. Eine große Herausforderung stellt dabei der Kollisionsschutz dar.

Der aktuelle Lösungsansatz von YXLON für Kollisionsschutz besteht aus zwei Einheiten.

Ein Teil wird zur Laufzeit in der SPS berechnet und schützt die Maschinenteile (Röntgenröhre, Manipulatortisch, Werkstückaufspannung, Detektor) vor Kollision miteinander. Hierbei werden bei der Realisierung die Kollisionskörper als grobe Quader abstrahiert und dargestellt. Dadurch wird eine schnelle Funktionalität erreicht. Dieses Kollisionsschutzmodul ist immer aktiv.

Der andere Teil des Kollisionsschutzes wird in der PC Steuerungssoftware realisiert. Dazu werden zum einen CAD-Zeichnungen der eingesetzten Maschinenbestandteile vom PC interpretiert, zum anderen wird per Visioncamera (Basler Kamera) eine Punktewolke des Prüfobjektes erfasst und vom PC interpretiert. Dieser Teil funktioniert nicht in Echtzeit.

2.3.4. Kommunikation zwischen beiden Modulen

Im Folgenden wird das Zusammenspiel beider Kollisionsschutzmodule durch ein UML-Diagramm[3] für den aktuellen Lösungsansatz von YXLON dargestellt.

[3] UML: Die Abkürzung steht für Unified Modeling Language. UML ist eine Sammlung grafischer Notationen zur Beschreibung, Visualisierung, Entwicklung und Dokumentierung von Softwaresystemen. Diese Notation wird von der Object Management Group (OMG) kontrolliert. UML bietet durch ihre Diagrammtypen die Möglichkeit an, ein System aus mehreren Blickwinkeln zu untersuchen. Die aktuelle Version ist UML2 und erfasst 13 offizielle Diagrammtypen. Für diese Masterarbeit wurden die UML-Diagramme als Werkzeug in der Modellierungs-, Analyse- und Entwicklungsphase eingesetzt. Das Visualisierungsprogramm Microsoft Visio 2010 dient als Tool.

Die Wunschpostion wird über User Interface an das Crash Protection Modul (im PC) gegeben. Bevor diese Postion an die SPS abgesendet wird, wird sie auf Kollisionsfreiheit geprüft. Um die Kollisionsfreiheit überprüfen zu können, wird die Fahrt simuliert. In der folgende Abbildung wird diese Kommunikation dargestellt.

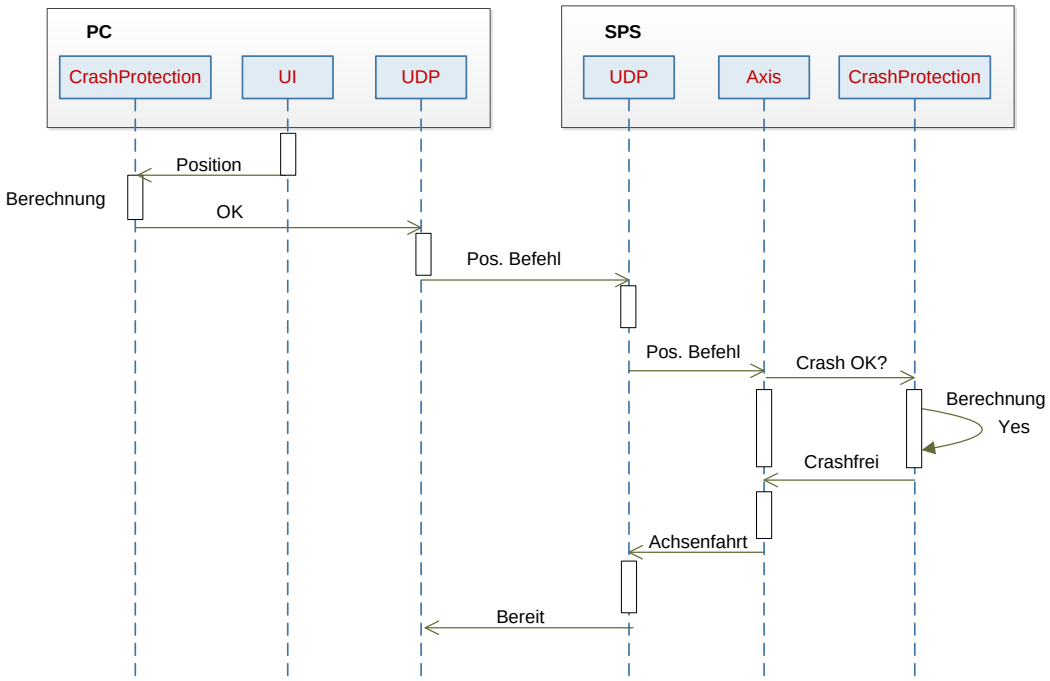


Abbildung 2.5.: Zusammenspiel beider Kollisionsschutzmodule. Hier für den Fall Kollisionsfrei.

Das CrashProtection-Modul im PC bekommt die aktuelle, sowie die Zielposition übergeben. Der PC hat Polygonnetz-Modelle der Röhre, des Detektors und des Prüfteils. In einer Anzahl von Positionsinkrementen (abhängig von der gewünschten Positioniergenauigkeit) wird der kleinste Abstand zwischen Detektor, Objekt und Röhre ermittelt. Wird dieser Wert kleiner als der definierte Mindestabstand, wird die letzte gültige Position an den PC zurückgegeben. Zur Bestimmung des kleinsten Abstands zwischen Röhre, Objekt und Detektor wird die Bullet Physik-Engine - eine Open Source Software für 3D-Kollisionserkennung - verwendet.

Wie erwähnt, hat sich diese Lösung als nicht Kollisionssicher erwiesen, da der - nicht echtzeitfähige - PC in diesem Aufbau Echtzeitberechnungen durchführen muss. Aus dieser Erfahrung entstand der Bedarf nach einer echtzeitfähige Lösung.

3. Grundlagen der Kollisionsvermeidung

In den meisten Fällen wird in der Kollisionsvermeidung auf die selben Grundelemente zurückgegriffen, wie in der Kollisionserkennung[1]. Die Aufgabe eines Kollisionserkennungssystems ist, für eine vorgegebene Anzahl von Objekten alle ihre Berührungen oder Annäherungen zu bestimmen und realisieren.

Dieses Kapitel beschreibt die mathematischen Grundlagen der Kollisionserkennung, die zum Verständnis dieser Arbeit relevant sind. Es zu beachten, dass die Kollisionserkennung in 3D aus sehr komplexen Schritten besteht, daher werden hier nicht alle Aspekte detailliert beschrieben und diskutiert. Für die genaueren Ausführungen wird auf (Ericson, 2005) verwiesen.

3.1. Kollisionsvermeidung Allgemein

Von der Computergrafik und Robotik ausgehend, ist Kollisionserkennung von komplexen 3D Objekten auch in der Automatisierungstechnik auch von beachtlicher Bedeutung.
Viele der Kollisionserkennungsansätze basieren auf dem Prinzip "Teile und Herrsche". Hierbei wird das anfängliche Problem so lange rekursiv in kleinere und einfachere Unterprobleme aufgeteilt, bis diese lösbar sind. Im Anschluss wird aus diesen Unterlösungen eine Lösung für das anfängliche Problem entwickelt.
Laut (Jimenez u. a., 2001) existieren grundsätzlich zwei Kategorien von Kollisionserkennungsansätzen: der geometrische und der algebraische Ansatz.
Die Anwendung der Methoden aus der Algorithmischen Geometrie setzt voraus, dass präzise Informationen über die Geometrie der zu kollidierenden Objekte vorliegen. Grundsätzlich geht es dabei darum, möglichst schnell zu erkennen, ob sich zwei oder mehr Objekte innerhalb einer definierten Szene überschneiden. Dies ließe sich bspw. durch regelmäßige Berechnung der kleinsten Abstände zwischen den Objekten bzw. den Grundelementen realisieren.
Die algebraischen Ansätze basieren auf der Parametrisierung der Trajektorien[2]. Dabei kann die Kollision instant analytisch bestimmt werden, wenn die Trajektorien des Objektes als Funktion der Zeit dargestellt werden können. Da dieser Ansatz in dieser Arbeit nicht angewendet wird, wird sie hier nicht weiter diskutiert. Eine detaillierte Erklärung findet man bei (Jimenez

[1] Die Kollisionsvermeidung macht sich die Kollisionserkennung zunutze, um zwei oder beliebig viele Objekte so nah wie möglich anzunähern, ohne sie kollidieren zu lassen.
[2] Ein Trajektorie ist der Bewegungspfad oder die Bahnkurve eines Objektes.

u. a., 2001) sowie bei (Canny, 1986).
Da das Bestimmen der geometrischen Eigenschaften der Objekte vergleichsweise simpel ist, wird der geometrische Ansatz am häufigsten angewendet.

Die naive Lösung zur Bestimmung der Kollision zwischen 3D Objekten mit vielen Polygonen[3] besteht darin, jedes Objekt und zwar jedes Polygon mit jedem anderen zu testen. Diese direkten Überlappungstests sind sehr rechen- und daher zeitaufwendig. Der Aufwand ergibt sich folgendermaßen, wobei n die Anzahl der Polygone bezeichnet, die zur Darstellung eines Objektes benötigt werden:

$$AnzahlderRechenoperationen = n.(n-1)/2 \tag{3.1}$$

Da dies - gerade bei komplexeren Objekten - schnell zu Performance-Problemen führt, ist die Reduzierung der Überlappungstests an dieser Stelle sehr bedeutend. Eine Methode zur Reduzierung der Anzahl der Überlappungstests ist die Aufteilung des Kollisionserkennungsprozesses in zwei Kollisionserkennungsprozessesaufteilungsphasen .

3.2. Zweiphasige Kollisionserkennung

Bei den meisten Algorithmen wird die Kollisionserkennung in zwei Phasen durchgeführt.
In der ersten Phase, der sogenannten Broad-Phase (Deutsch: Weiten Phase) wird überprüft, welche Objekte sich eigentlich schneiden können. Dazu werden die Bounding Volumes/ Boxes (Deutsch: Hüllkörper) verwendet. Diese sind im nächsten Unterkapitel noch ausführlicher beschrieben.
In der zweiten Phase, der Narrow-Phase (Deutsch: Nahe Phase) werden die komplexen Körper innerhalb der Bounding Volumes (das können Polytope, Quader oder Hüllkugeln sein) auf mögliche Schnittpunkte überprüft. Hierfür wird die Bounding Volume Hierarchy (BVH) verwendet, auf die ebenfalls weiter unten noch eingegangen wird.

3.3. Hüllkörper (Bounding Volume)

Die Repräsentation der 3D Objekte ist für die Kollisionserkennung sehr entscheidend. Eine präzise Objektrepräsentation führt zu einer exakten Kollisionserkennung. Die zweite, ebenso wichtige Anforderung verlangt aber eine endliche Anzahl von Rechenzyklen, um die Anwendung echtzeitfähig zu machen. Die Objektrepräsentation muss also sowohl präzise, als auch effizient strukturiert sein.

[3] Polygondarstellung ist eine der Darstellungsmöglichkeiten von 3D Objekten in der Computergrafik.

Wie bereits erwähnt, ist eine naiver Überlappungstest zwischen 3D Objekten zu rechenaufwendig für ein Echtzeitsystem mit begrenzten Zykluszeiten. Um die Zahl der notwendigen Überlappungstests zu verringern, werden die Objekte daher mit hierarchisch geschachtelten Bounding Volumes (BVs)[4] repräsentiert, die eine Bounding Volume Hierarchy (BVH) bilden. Die BVs umschließen ein Objekt vollständig und ermöglichen somit die Überlappungstests für einfache Geometrien durchzuführen. Die Idee besteht darin, dass die Objekte sich nur dann überlappen können, wenn auch deren BVs sich überlappen (Ericson, 2005).
Es existieren laut (Ericson, 2005) unter anderem folgende Arten von Bounding Volumes:

- Hüllkugeln (engl. Spheres),
- Achsparalleler Quader (engl. Axis-aligned Bounding Boxes, kurz: AABB),
- Gerichtete Quader (engl. Oriented Bounding Boxes, kurz: OBB),
- Diskrete orientierte Polytope (engl. Discrete Oriented Polytopes, kurz: DOP).

Im folgenden Kapitel werden diese BVs kurz vorgestellt.

3.3.1. Hüllkugeln

Hüllkugeln sind kugelförmige Hüllkörper, die das Objekt oder dessen Punktewolke umschließen. Die mathematische Beschreibung dieses Hüllkörpers ist sehr einfach. Für Hüllkugeln werden ausschließlich der Kugelmittelpunkt und der Kugelradius gespeichert. Sie sind rotationsinvariant.

3.3.2. Achsparalleler Quader

Axis-Aligned Bounding Boxes (AABB) sind rechteckige Hüllkörper, die senkrecht zum Weltkoordinatensystem ausgerichtet sind. Sie sind die einfachste Form eines Bounding Volumes. Für die Erstellung der AABBs, beispielweise in 2D, werden durch einen Minimum/Maximum-Algorithmus der linke oberste und rechte unterste Punkt gesucht. Diese beiden Punkte stellen die rechteckige Bounding Box dar, die das Objekt umgibt. Der Speicherbedarf für AABBs ist gering und der Überlappungstest ist relativ einfach. Der Nachteil besteht darin, dass sie nicht rotationsinvariant sind.

[4] Ein Bounding Volume ist ein einfacher geometrischer Körper, der ein komplexes dreidimensionales Objekt oder einen komplexen Körper umschließt (Bender und Brill, 2006).

3.3.3. Gerichtete Quader

Oriented Bounding Boxes (OBB) sind den AABBs ähnlich. Der Hauptunterschied liegt darin, dass die Quader an den Achsen des Objektes ausgerichtet sind. Demzufolge bietet dieser Hüllkörper eine präzisere Approximation der Geometrie des Objektes als die oben genannten Hüllkörper. Die optimale Approximation des Objektes ist allerdings sehr aufwendig. Der große Vorteil von OBBs besteht darin, dass sie rotationsinvariant sind.

3.3.4. k-Diskrete orientierte Polytope

Die k-Diskreten orientierten Polytope (k-DOP) können als ein verallgemeinerter Fall des achsparallelen Quaders (der OBBs) betrachtet werden, wobei k die Anzahl der Flächen des Polytops repräsentiert. OBBs entsprechen demnach einem k-DOP mit k = 6. Die Objektrepräsentation mit k-DOPs erlaubt Polytope mit beliebig vielen (k) Beschränkungsflächen. Dadurch können die Objekte exakter eingeschlossen werden. Je größer k gewählt wird, desto besser passt sich der Hüllkörper an das Objekt an. Der Nachteil der k-DOPs liegt darin, dass sie bei Rotation neu berechnet werden müssen (Klosowski u. a., 1998).

Die folgende Abbildung 3.1 zeigt die verschiedenen Bounding Volumes in zweidimensionaler Darstellung.

Abbildung 3.1.: Bounding Volume in zweidimensionaler Darstellung
vgl. (Ericson, 2005, S. 77)

3.3.5. Vergleich wichtiger Bounding Volumes

Die folgende Tabelle stellt die wesentlichen Vor- und Nachteile der Hüllkörper dar.

BVs	Hülleffizienz	Kollisionsberech.	Neuberech. Rotation	Erstellung	Speicher
Kreis	sehr schlecht	sehr einfach	Nein	mittel	gering
AABB	schlecht	einfach	Ja	einfach	gering
OBB	gut	mittel	Nein	schwer	hoch
DOP	sehr gut	mittel	Ja	mittel	sehr hoch

Tabelle 3.1.: Vergleich wichtiger Bounding Volumes

Man kann erkennen, dass jeder der vorgestellten Hüllkörper Vor- und Nachteile besitzt. Je nach Szenario eignet sich ein Hüllkörper besser als der andere. Beispielweise in zweidimensionalen Szenarien bietet sich der achsorientierte Quader (AABB) an, da er hier besonders leicht zu erstellen ist und sich die Algorithmen zur Neuberechnung nach Rotation des Objekts stark vereinfachen. Die Hüllkugel bietet sich für Objekte an, deren Bestandteile weit voneinander entfernt sind. Für Anwendungsfälle, bei denen keine Einschränkungen wegen des Speicheraufwands vorhanden sind, eignen sich die k-DOPs sehr gut.

3.4. Hüllkörper-Hierarchie (Bounding Volume Hierarchy)

Wie bereits erwähnt, kommt die Bounding Volume Hierarchy (BVHs) in der zweiten Phase der Kollisionserkennung zum Tragen, um den Prozess zu beschleunigen. In der Hüllkörper-Hierarchie werden die Objekte, bzw. die das Objekt repräsentierenden Unter-Hüllkörper, in einer Baumstruktur (Hierarchie von Knoten) organisiert. Dabei stellt jeder Zweig (Teilbaum) einen zusammenhängenden Teil des Objekts dar. Wird nun bei der Kollisionsprüfung mit Hilfe des Traversierungsalgorithmus an einem beliebigen Knoten der BVH eine Kollision erkannt, so kann die Traversierung frühzeitig abgebrochen werden. Die minimale mögliche Distanz zwischen den Objekten wurde erreicht.

Mit Hilfe von Hierarchischen Hüllkörpern kann zwar die Performance verbessert werden, jedoch werden die Tests immer noch paarweise durchgeführt. Das bedeutet, der Aufwand wird nur um einen konstanten Faktor verringert.

3.4.1. Erstellen der Bounding Volume Hierarchie

Zum Verständnis der BVH sei vorweg erwähnt, dass vor der Kollisionserkennung die Erfassung des 3D-Objektes mit einer Kamera stattfindet. Diese bereitet die Lage und Form des Objektes

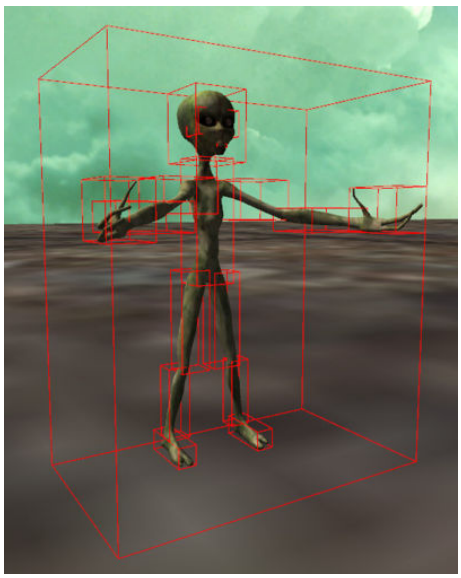


Abbildung 3.2.: Hüllkörper-Hierarchie eines Modells aus Bounding Volume
Quelle: (Freiburg, 2015)

im Raum als Punktewolke auf. Diese Punktewolke ist jedoch ungeeignet für eine performante Kollisionserkennung. Hier kommt die Bounding Volumes Hierarchie ins Spiel, die auf Grundlage der Punktewolke das Objekt in einfachen geometrischen Formen nachbildet.

Diese Bounding Volume Hierarchien werden nach zwei Ordnungsprinzipien aufgestellt:

- Top Down
- Bottom Up

Top Down (am verbreitetsten):
Beim Top Down-Ansatz wird zunächst ein Hüllkörper gebildet, der das gesamte 3D-Modell, also die gesamte Punktewolke umfasst. In die Baumstruktur übersetzt entspricht dieser (größte) Hüllkörper der Wurzel des Baums. Im nächsten Schritt wird die Punktemenge in zwei oder mehr Teile aufgeteilt und wiederholt ein Hüllkörper um jede dieser Teilmengen gebildet. Dieser Vorgang wird rekursiv solange ausgeführt, bis entweder in jeder Menge nur noch eine bestimmte Anzahl von Punkten enthalten ist, oder eine zuvor festgelegte maximale Tiefe erreicht wurde. Da diese Hierarchie schnell und leicht erstellt werden kann, wird diese Methode häufiger benutzt (Gottschalk, 2000).

Es ist zu beachten, dass die Regel, nach der die Objekte aufgeteilt werden, einen großen

Einfluss auf die Qualität der Hierarchie hat (Klosowski u. a., 1998).
Dies wird im Kapitel 5 Realisierung noch ausführlicher diskutiert.

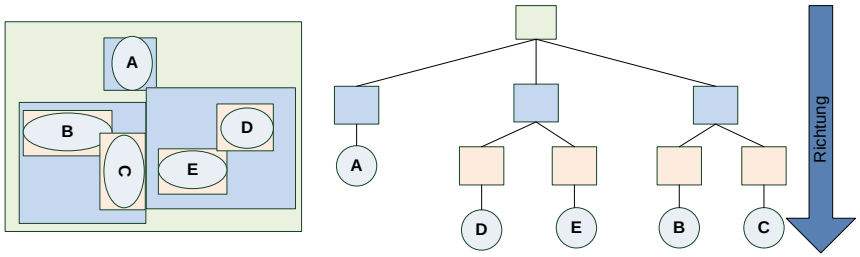


Abbildung 3.3.: Top Down-Hierarchie

Bottom Up:
Beim Bottom Up-Ansatz beginnt man mit den Primitiven, und baut durch Verschmelzung benachbarter Knoten ihre Vorgänger auf. Dies wird solange gemacht, bis nur noch ein Hüllkörper übrigbleibt, die Wurzel des Hierarchiebaumes (Gottschalk, 2000). Sie bietet den Vorteil, dass man bei einer definierten Hüllkörpergröße beginnt und die Rekursion einen klar definierten Endpunkt hat, wenn die Bedingung erfüllt ist, dass sich die ganze Punktemenge innerhalb des Hüllkörpers befindet. Diese Strategie benötigt jedoch die meiste Rechenzeit und wird daher nur in Fällen ohne Rechenzeitbegrenzung angewendet.

3.4.2. Kollisionstest zwischen zwei Hüllkörper-Hierarchien

Die Hüllkörper-Hierarchien stellen im Allgemein zwei bewegliche Objekte dar, die auf Kollision geprüft werden sollen.

Grundsätzlich existieren drei Ansätze für diese Überprüfung:

- Gleichzeitig in der Hierarchie A und in der Hierarchie B eine Ebene absteigen.
- Zuerst vollständig in der Hierarchie, die den größeren Hüllkörper besitzt, absteigen. Danach in der Hierarchie mit dem kleineren Hüllkörper.
- Kontinuierlich die Hierarchie mit dem größeren Hüllkörper bestimmen und dieser Hierarchie eine Ebene absteigen.

Die erste Methode ist relativ einfach zu implementieren und man kann schnell in der Hierarchie absteigen. Da kein Hierarchie-Größenvergleich erforderlich ist, wird diese Methode bei sehr unterschiedlichen Größen von A und B wesentlich länger dauern. Bei der zweiten und dritten Methoden muss zuerst die Definition des größeren Hüllkörper festgelegt werden.

Auch für den Vergleich zwischen zwei Hüllkörper-Hierarchien gibt es also keine allgemeingültige Lösung. Je nach Szenarien eignet sich jeweils eine andere Strategie besser. In (Ericson, 2005) werden die Vor- und Nachteile der verschiedenen Strategien ausführlicher beschrieben. Die Abbildung 3.4 zeigt den gegenseitigen Test zweier Hierarchien, die als Binärbäume gespeichert sind. Zuerst werden die beiden Wurzeln gegeneinander getestet (1). Wenn diese sich überlappen, erfolgt eine Überprüfung der Kinder. Dabei wird in den Zweig abgestiegen, der das größere Bounding Volume hat. Im abgebildeten Beispiel wird beim 9. Vergleich eine

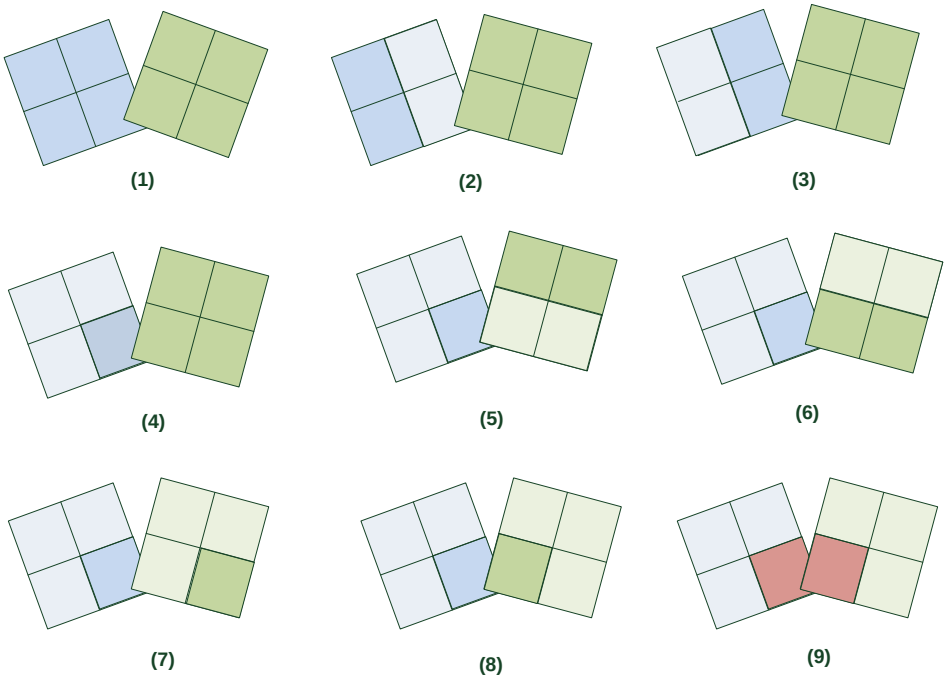


Abbildung 3.4.: Vergleich zwischen zwei Hüllkörper-Hierarchien
Vgl.(Mezger, 2001, S. 38)

Kollision festgestellt.

Die am weitesten verbreitete Methode ist die dritte Methode, bei der zuerst in der Hierarchie mit dem größeren Hüllkörper abgestiegen wird. Diese Strategie wird von Autoren wie (Gottschalk, 2000) und (Bergen, 1998) favorisiert.

3.5. Überlappungstest Hüllkörper

Nach der anstrakten Darstellung der Kollisionstest der Überführung des komplexen Objektes in eine Menge von einfachen geometrischen Hüllkörpern stellt sich nun das Problem, wie sich die Hüllkörper zweier Objekte auf Überlappung überprüfen lassen. Dabei gilt, dass ein beliebiges Objekt sich dann mit dem zweiten überlappt, wenn sich auch nur ein Hüllkörper aus der ersten Menge mit einem Hüllkörper des zweiten Objektes überlappt.
Die einfachste Methode, um herauszufinden, Ob sich zwei Polyeder A und B überlappen bzw. überschneiden, ist die Überprüfung jeder Kante von A mit jeder Fläche von B und umgekehrt. Dies ist jedoch sogar für Polyeder mit wenigen Ecken sehr aufwendig und teuer. Beispielsweise für zwei Quader müssten zweimal 12 Kanten gegen 6 Flächen überprüft werden.
Eine effizientere Lösung bietet das Separating Axis Theorem an (Ericson, 2005).

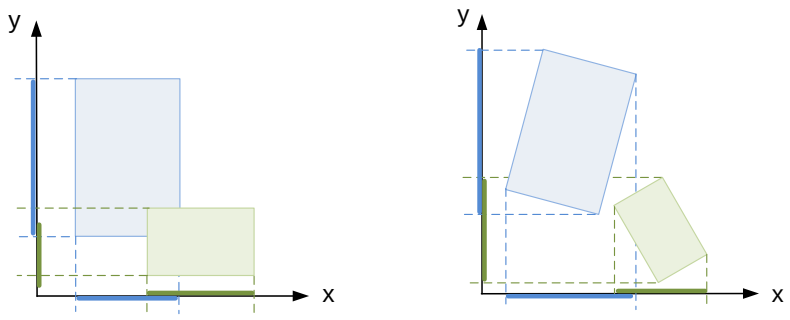
3.5.1. Seprating Axis Theorem

Nach dem Separating Axis Theorem kann eine Kollision zwischen zwei Polyeder dann ausgeschlossen werden, wenn mindestens eine separierende Achse für diese Polyeder existiert. Für zwei gegebene Polyeder A und B existiert dann eine separierende Achse, wenn die Projektionen der beiden Objekte auf diese Achsen sich nicht überlappen (Gottschalk u. a., 1996b). Einen Spezialfall des Separating Axis Theorem stellt der Überlappungstest zweier Achs-paralleler Quader in 3D-Raum dar.

3.5.2. Überlappungstest für Achsparallele Quader

Zwei Achsparallele Quader überlappen sich nur dann, wenn alle ihre Projektionen auf die Koordinatenachsen x,y und z sich überlappen. Das bedeutet wenn es mindestens eine Koordinatenachse gibt, auf der die Projektionen der Achsparallelen Quader sich nicht schneiden, kann die Kollision ausgeschlossen werden.

Die Abbildung 3.5(a) stellt die separierende Achse der Achsparallelen Quader dar.



(a) Achsparalleler Quader Überlappungstest (b) Gerichtete Quader Überlappungstest

Abbildung 3.5.: Überlappungstest Hüllkörper, Vgl. (Wittmann, 2011, S. 18)

3.5.3. Überlappungstest für Gerichtete Quader

Beim Gerichteten Quader ist die Prüfung auf Überschneidung nicht so trivial wie beim Achsparallelen Quader. Wie in der Abbildung 3.5(b) zu erkennen ist, überlappen sich die beiden Objekte nicht, obwohl sich ihre Projektionen auf Koordinatenachsen (x- und y-Achse) überlappen. Demzufolge muss eine Achse im Raum gesucht werden, die separierende Achse.
Laut (Gottschalk, 2000) kann gezeigt werden, dass zwei konvexe Polyeder, die sich nicht überschneiden, immer durch eine Ebene separiert werden können, die entweder parallel zu einer Fläche des Polyeders oder parallel zu einer Kante von jedem Polyeder läuft. Die Abbildung 3.5(b) stellt diese Achse dar.

Um diese Achse zu finden, werden die Mittelpunkte und die aufgespannten Radien der Bounding Boxes auf eine der zu testenden Achsen projiziert (Abbildung 3.6). Ist die Entfernung der Mittelpunkte größer als die Summe der Radien, so überlappen sich die Intervalle und damit die Objekte nicht. Diese gefundene Achse ist die separierende Achse, deren Berechnung durch folgende Formel erfolgt:

$$|\vec{T}.\vec{L}| > \sum_i |a_i \vec{A^i}.\vec{L}| + \sum_i |b_i \vec{B^i}.\vec{L}| \tag{3.2}$$

Ist diese Bedingung erfüllt, so separiert L die beiden getesteten Boxen und diese überlappen sich folglich nicht.
Für Rechteckige bzw. Quaderförmige Hüllkörper beträgt die Anzahl der Überprüfungen im Zweidimensionalen 9 und im Dreidimensionalen 15.

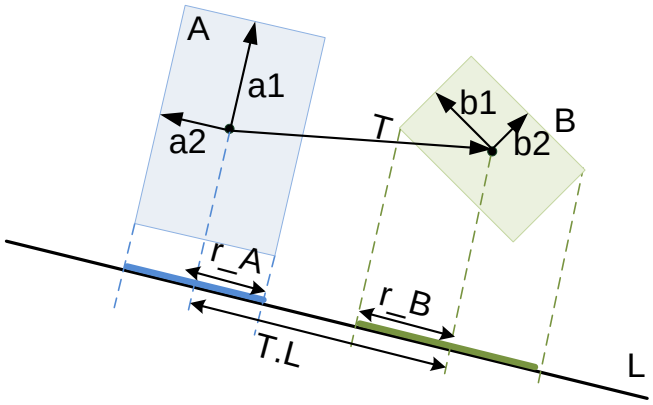


Abbildung 3.6.: OBB separierende Achse, Vgl. (Ericson, 2005, S. 102)

3.6. Abstandsmessung Hüllkörper

Die meisten Kollisionserkennungs-Algorithmen in der Computergrafik haben die Aufgabe festzustellen, ob sich zwei Objekte überschneiden oder nicht. Das bedeutet, der Algorithmus beschreibt die Suche nach der Antwort JA/ NEIN. Für die Aufgabe dieser Masterarbeit hingegen spielt die Abstandsberechnung eine beträchtliche Rolle. Es müssen regelmäßig die kürzesten Abstände zwischen den Objekten in der Anlage berechnet werden, um zu prüfen, ob sich diese Abstände noch ohne Gefahr der Kollision verringern lassen. Diese Berechnungen sollen schnell und robust durchgeführt werden. Um die Effektivität der Kollisionsberechnungen zu erhöhen, werden die Abstandsberechnungen zwischen grundlegenden Elementen wie Punkten, Geraden oder Strecken durchgeführt.
Die zwei wichtigsten Algorithmen zur Abstandsberechnung werden hier vorgestellt. Weitere Algorithmen zur Abstandsberechnung, deren Herleitung sowie Hinweise zur robusten Implementierung sind in (Ericson, 2005) zu finden.

3.6.1. Minkowski-Summe

Die Minkowski-Summe bietet eine einfache Methode zur Kollisionserkennung und zur Abstandsberechnung von zwei Objekten an.

Definition 1: *Minkowski-Summe*
Seien $A, B \in R^n$ Punktmengen und a, b Ortsvektoren zu den korrespondierenden Punkten in den Mengen A und B, ist die Minkowski-Summe $A \oplus B$ definiert als (Ericson, 2005, S. 70):

$$A \oplus B = \{a + b | a \in A, b \in B\} \tag{3.3}$$

Somit enthält die Minkowski-Summe die resultierende Summen-Menge aller Elemente aus A und aller Elemente aus B.

Definition 2: *Minkowski-Differenz*
Analog zur Minkowski-Summe wird die Minkowski-Differenz von zwei Punktmengen als Subtraktion zweier Vektoren a und b definiert (Ericson, 2005, S. 71).

$$A \ominus B = \{a - b | a \in A, b \in B\} \tag{3.4}$$

Minkowski-Differenz und Kollisionserkennung

Definition 3: *Kollision*
Wenn A und B zwei Polygone sind und deren Durchschnittsmenge ungleich der leeren Menge ist, dann ist der Ursprung des Koordinatensystems in der Minkowski-Differenz enthalten- die beiden Polygone A und B kollidieren (Ericson, 2005, S. 71).

$$A \cap B \neq \emptyset \equiv 0 \in A \ominus B \tag{3.5}$$

Minkowski-Differenz und Abstandsberechnungen
Die Minkowski-Differenz kann ebenfalls dazu verwendet werden, um den minimalen Abstand zwischen zwei Objekten zu bestimmen.

Definition 4: *Distanz*
Der minimale Abstand zwischen zwei Polygonen A und B ist äquivalent zu dem Abstand der Minkowski-Differenz $A \ominus B$ zum Ursprung des Koordinatensystems (Ericson, 2005, S. 400).

$$d(A, B) = min\{\|a - b\| : a \in A, b \in B\} = min\{\|c\| : c \in A \ominus B\} \tag{3.6}$$

Der Vorteil der Minkowski-Differenz liegt darin, dass das Ursprungsproblem der Berechnung des minimalen Abstands zwischen zwei Objekten sich auf die Bestimmung des minimalen Abstands zwischen der Minkowski-Differenzmenge und dem Ursprung des Koordinatensystems

reduziert. Es ist zu beachten, dass beide Objekte im gleichen Koordinatensystem definiet sein müssen.
Um die Effizienz des Algorithmus zu erhöhen und die Begrenzung der Minkowski-Differenz zu bestimmen, wird statt der Minkowski-Differenzmenge deren konvexe Hülle benutzt. Die Komplexität der Minkowski-Differenz ist von der Berechnung der konvexe Hülle abhängig. Infolgedessen wurde im Rahmen dieser Masterarbeit eine gründliche Recherche bezüglich der konvexen Hülle durchgeführt, deren Ergebnisse hier kurz vorgestellt werden.

3.6.2. Konvexe Hülle

Eine Menge ist konvex, wenn die Verbindungsstrecke zwischen je zwei Punkten dieser Menge auch vollständig im durch diese Menge aufgespannten Raum liegt. Mit anderen Worten, die konvexe Hülle einer Menge ist ein konvexes Polygon, das die Ausgangsmenge enthält Barber u. a. (December 1996).

Beschreibung der konvexen Hülle in 2D:

$$C = \{x | x = \lambda.A + (1 - \lambda).B, 0 \leq \lambda \leq 1\} \tag{3.7}$$

Beschreibung der konvexen Hülle in 3D:

$$C = \{x | x = \lambda_1.A + \lambda_2.B + \lambda_3.C, \lambda_i \geq 0, \sum \lambda_i = 1\} \tag{3.8}$$

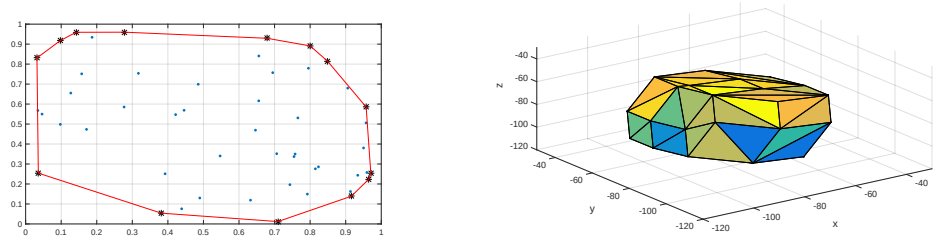
Allgemeine Beschreibung der konvexen Hülle :

$$C = \{x | x = \sum \lambda_i.A_i, \lambda_i \geq 0, , \lambda_i \geq 0, \sum \lambda_i = 1\} \tag{3.9}$$

Die folgende Abbildung zeigt Beispiele für die konvexe Hülle einer endlichen Punktmenge.

In der Literatur finden sich zahlreiche Algorithmen zur Ermittlung der konvexen Hülle in 2D. Soll die konvexe Hülle - wie für den hier untersuchten Anwendungsfall - im Dreidimensionalen Raum ermittelt werden, verringert sich die Auswahl der möglichen Algorithmen auf die Folgenden drei:

- Gift Wrapping-Algorithmus,
- QuickHull-Algorithmus,
- Divide-and-Conquer-Verfahren.



(a) Konvexe Hülle 2D

(b) Konvexe Hülle 3D

Abbildung 3.7.: Konvexe Hülle einer Punktmenge

Für den hier untersuchten Anwendungsfall in einem Echtzeit-System ist die Laufzeit das wichtigste Kriterium eines Algorithmus zur Bestimmung der konvexen Hülle. Für die drei genannten Algorithmen ergeben sich die folgenden Laufzeiten:

Algorithmus	**Laufzeit**	**Worst Case**
Gift Wrapping-Algorithmus	$O(n.h)$	$O(n^2)$
QuickHull-Algorithmus	$O(n.\log n)$	$O(n^2)$
Divide-and-Conquer-Verfahren	$O(n.\log n)$	$O(n^2)$

Tabelle 3.2.: Vergleich wichtiger Algorithmen zur Berechnung der konvexen Hülle

Anmerkung : h ist die Anzahl der Punkte auf dem Rand der Hülle.

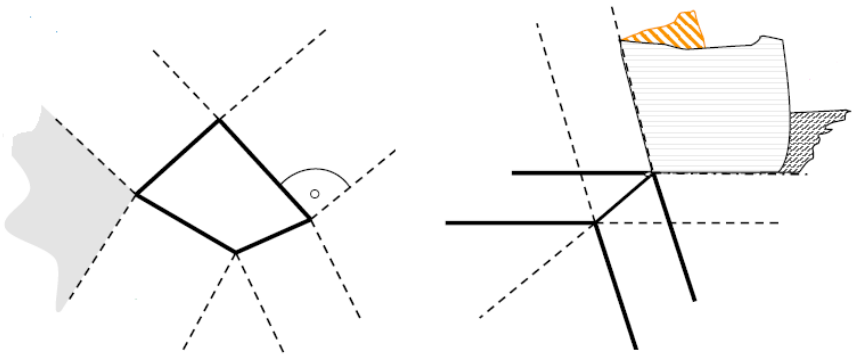
Da die Entwicklung der Kollisionsvermeidung für diese Arbeit zuerst mit Matlab erfolgen sollte, und Matlab eine mathematische Bibliothek für den QuickHull-Algorithmus anbietet, fiel die Wahl auf diesen Algorithmus.

Eine detaillierte Beschreibung der Eigenschaften und der Ideen der einzelnen Algorithmen bietet (Ericson, 2005).

3.6.3. Voronoi Region

Neben der Minkowski-Summe existiert die Voronoi-Region zur Bestimmung des kleinsten Abstands zwischen zwei Objekten. Dieser Algorithmus sterbt an, das Problem der Berechnung des minimalen Abstands zwischen zwei Objekten auf die Bestimmung der nächstgelegenen Punkte der beiden Objekte zu reduziert.
Das Voronoi-Diagramm für eine Punktmenge $P = \{p_1, ..., p_n\}$ ist die Einteilung einer Ebene

in n disjunkte Gebiete, die sogenannte Voronoi-Region (Klein, 1997). Das heißt, die Voronoi Region eines Punktes p ist die Menge aller Punkte, die näher zu p_i als zu jedem anderen Punkt p_j sind (Ericson, 2005, S. 69). Somit wird jede Kante der Voronoi Region des Punktes durch dem nächsten Nachbarn dieses Punktes definiert. Die Abbildung 3.8 stellt die Voronoi Region zu einer Kante und zu einer Ecke in 2D und 3D dar.



(a) Voronoi-Region in 2D (b) Voronoi-Region in 3D

Abbildung 3.8.: Voronoi-Region zu einer Kante, Quelle: (Zachmann, 2011, S. 11)

Definition 5: *Feature*
Das Feature P^f eines Polyeders P bezeichnet die Ecken, Kanten oder Polygone des Polyeders.

Definition 6: *Closest Feature*
Seien P und Q zwei Polyeder und f^P und f^Q zwei Features von P bzw. Q. Weiterhin seien p und q Punkte auf f^P bzw. f^Q, die den kleinsten Abstand von P und Q realisieren, das heißt:

$$d(P,Q) = d(f^P, f^Q) = \|p - q\| \tag{3.10}$$

Dann heißen f^P und f^Q "Closest Feature".

Definition 6: *Kollision*
Wenn der Punkt p in der Voronoi Region von Q und der Punkt q in der Voronoi Region von P liegt, dann kollidieren die Objekte P und Q. Und die Punkte p und q sind zwei Punkte, die am dichtesten aneinander liegen.

In der Abbildung 3.9 werden diese Punkte dargestellt.

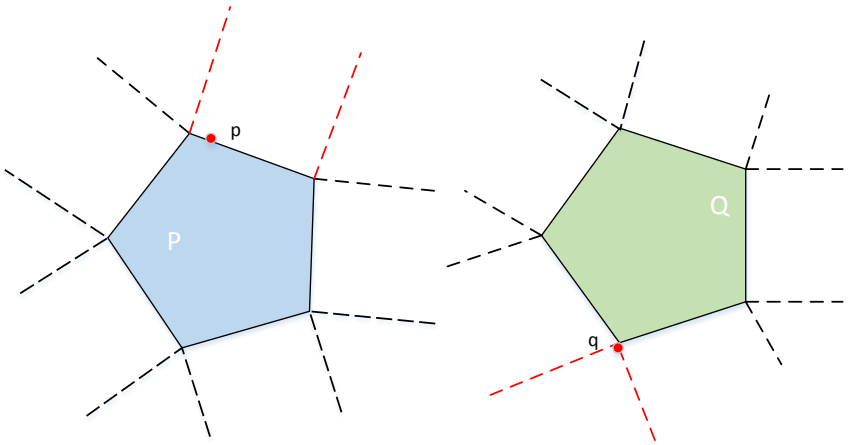


Abbildung 3.9.: Kollision zweier Objekte in 2D

4. Design

Dieses Kapitel beschreibt die Schritte der Auswahl des Kollisionserkennungsalgorithmus für die gestellte Aufgabe. Des Weiteren enthält es eine Beschreibung der Softwarearchitektur, die bei der Kollisionserkennung zum Einsatz kommen soll.

4.1. Systemarchitektur des Kollisionserkennungssystems

Wie in der Abbildung 2.2 zu erkennen ist, besitzt jede YXLON-CT-Anlage einen Rekonstruktions-PC und eine Steuerung. Die Kommunikation zwischen den beiden Komponenten geschieht über das User Datagram Protocol, kurz UDP[1]. Der Rekonstruktions-PC stellt mit Hilfe der Basler-Kamera ein Bild von dem Prüfobjekt dar. Die Informationen über die Geometrie des aktuellen Prüfobjekts werden in einer Log-Datei gespeichert, welche detailliert im Kapitel 5 Realisierung dargestellt wird. Des Weiteren verfügt der Rekonstruktions-PC über die Polygonnetz-Modelle der Röhre und des Detektors in Form von STL-Datei [2].
Die Entwicklung des Kollisionserkennungssystems erfolgt auf Basis dieser Systemarchitektur der Anlagen. Der entscheidende Punkt ist, dass diese Architektur es ermöglicht, dass der PC die Offline-Berechnungen übernimmt, und dadurch die Rechenlast der Steuerung deutlich reduziert.

Mit den Ausführungen oben sind die wesentlichen Umgebungsbedingungen beschrieben, innerhalb derer das Problem der Kollisionsvermeidung gelöst werden muss.

4.2. Herleitung der Algorithmus-Anforderungen

Im Folgenden werden die Anforderungen an den Kollisionserkennungsalgorithmus hergeleitet. Auf der Grundlage des derzeit bei XYLON eingesetzten Systems wurde folgende Spezifikation für den zu entwickelnden Algorithmus aufgestellt:

[1] User Datagram Protocol ist ein verbindungsloses Übertragungsprotokoll zum Übermitteln von Datenpaketen von A nach B.

[2] Die Abkürzung STL steht für STereoLithography. Sie beschreibt 3D-Objekte mittels zusammengesetzter Dreiecke ihrer Oberflächen.

- Der Algorithmus soll auf einer B&R Steuerung von Typ X20CP1585 implementierbar sein.
- Die Rechenoperationen müssen aus einer Folge von elementaren Rechenschritten bestehen, um auf einer Steuerung ausführbar zu sein.
- Der Algorithmus muss echtzeitfähig[3] sein.
- Der Algorithmus soll modular aufgebaut werden.
- So viele Berechnungen wie möglich müssen offline durchgeführt werden.
- Die Anwendung externer Bibliotheken z.B. für die komplexe mathematische Operationen ist nicht möglich.

Von den Objekten sind darüber hinaus folgende Eigenschaften bekannt:

- Die Objekte haben Rotations- und Translationsbewegung.
- Alle Kollisionskörper bewegen sich gleichzeitig.
- Das System behandelt nur Festkörper und keine deformierbaren Körper.

Gemäß dieser Spezifikationen wurde ein Kollisionserkennungsalgorithmus entwickelt, der sich aus den drei großen Modulen

- Objekt-Repräsentatio,
- Hüllkörper-Hierarchie,
- Abstandsmessung

zusammensetzt.

Im Folgenden wird das jeweilige Design dieser Module und die Überlegungen, die zur Auswahl eines bestimmten Ansatzes geführt haben, kurz umrissen.

[3] Unter Echtzeit versteht man die genaue Bestimmung der maximalen Zeit, bis ein System auf ein Ereignis reagiert.

4.3. Auswahl des Kollisionserkennungsalgorithmus

4.3.1. OBB für Objekt-Repräsentation

Nach eingehender Literatur-Recherche zeigt sich, dass für die Repräsentation der Objekte in der vorliegenden Aufgabenstellung der Gerichtete Quader (Oriented Bounding Boxes, OBB) die am besten geeignete Objekt-Repräsentation darstellt.
Beim Gerichteten Quader ist zum einen die Hülleffizienz sehr gut, d.h., das Objekt kann mit wenig Aufwand ausreichend genau beschrieben werden. Zum anderen ist beim OBB-Modell keine Neuberechnung bei der Rotation erforderlich, da die Quader bereits bei ihrer initialen Berechnung rotationssymmetrisch angelegt werden. Diese beiden Eigenschaften werden hier ausgenutzt. Durch eine effiziente Objekt-Repräsentation wird eine präzise Abstandsmessung erreicht, und durch die Vermeidung der Neuberechnung bei der Rotation wird ein enormer Gewinn an Laufzeit erzielt.

4.3.2. Top-Down Hierarchie

Prinzipiell wird durch die Anwendung der Hüllkörper-Hierarchie der extrem hohe Rechenaufwand des Kollisionstests reduziert. Wie bereits im Unterkapitel 3.4.2 Kollisionstest zwischen zwei Hüllkörper-Hierarchien erwähnt, ist die weitverbreitetste Methode für den Hierarchie-Aufbau die Top-Down-Strategie. Zum einen ist sie sehr viel schneller als die Bottom-Up-Strategie bei der Berechnung der Kollisionstests, zum anderen ist schon die Erstellung der Hüllkörper-Hierarchie mit weniger Aufwand verbunden und somit die Implementierung unkomplizierter.
Aus diesen Gründen fiel die Entscheidung bei der Suche nach einem schnellen und robusten Ansatz für den Aufbau der Hüllkörper-Hierarchie auf die Top-Down-Strategie.

4.3.3. Abstandsmessung

Das wichtigste Auswahlkriterium für die Wahl des Verfahrens zur Abstandsmessung stellt die mathematische Komplexität dar. Entsprechend der vorliegenden Spezifikation soll die Abstandsmessung auf einer SPS durchführbar sein. Das bedeutet für die Auswahl des Algorithmus, dass die Rechenoperationen aus einer Folge von elementaren Rechenschritten bestehen müssen. Hier zeigte sich bei der Implementierung der Abstandsmessung, dass Voroni-Region-Ansatz zu komplex für die SPS ist.
Die Wahl fiel entsprechend auf die Minkowski-Summe, die zwar gegenüber des Voronoi-Algorithmus eine geringfügig höhere Anzahl an Berechnungsschritten benötigt, die sich aber mit elementaren Rechenoperationen durchführen lässt.

4.3.4. Konvexe Hülle

Wie schon in Abschnitt 3.6.2 Konvexe Hülle beschrieben wurde, habe alle konvexe Hülle Algorithmen zur Berechnung der konvexen Hülle, die für die Aufgabe dieser Arbeit geeignet wären, nahezu die gleiche Laufzeit. Demzufolge wurde das Auswahlkriterium auf der Häufigkeit der Anwendung der Methode gelegt. Der QuickHull-Algorithmus ist die am meistens benutze Methode und auch in Matlab verfügbar.

4.4. Architektur des Kollisionserkennungssystems

Nach der Auswahl des Kollisionserkennungsalgorithmus wurde eine der Spezifikation entsprechende Systemarchitektur entwickelt, welche in der Abbildung 4.1 dargestellt wird. Sie zeigt, dass der Algorithmus zur Kollisionserkennung aus zwei großen Teilprozessen besteht.
Der erste Teilprozess setzt sich aus initialen Berechnungen zusammen, die vor jedem Röntgenvorgang einmalig die Approximation der in der Szene beteiligten Objekte behandeln. Diese Berechnungen finden nicht während des eigentlichen Röntgenvorgangs statt und können daher auf dem Rekonstruktion-PC als Offline-Prozess (nicht Echtzeit-abhängig) durchgeführt werden.
Der zweite Teilprozess wird direkt auf der SPS berechnet und ermittelt zur Laufzeit den minimalen Abstand zwischen den zwei Objekten. Der zweite Teilprozess nutzt die Datenstruktur, die vom Rekonstruktions-PC auf die SPS übertragen wird und die erstellten Hierarchien der Objekte enthält. Die Datenstruktur besteht aus einer einfachen Text-Datei und wird nur einmal am Anfang des Röntgenvorgangs geladen.

(Neu!) Für die Entwicklung des Offline-Teilprozesses wurde Matlab als Entwicklungsumgebung verwendet. Erfahrungsgemäß ist Matlab für die Lösung mathematischer und technischer Probleme und für die Visualisierung von 3D-Objekten sehr geeignet. Weiterhin wurde beschlossen, dass die Entwicklung der Abstandsmessung (Online-Teilprozess) auch zuerst auf Matlab erfolgt und im Anschluss auf die SPS übertragen wird. Denn auf der SPS hat man keine Möglichkeit z.B. die Visualisierung der 3D-Objekte durchzuführen.

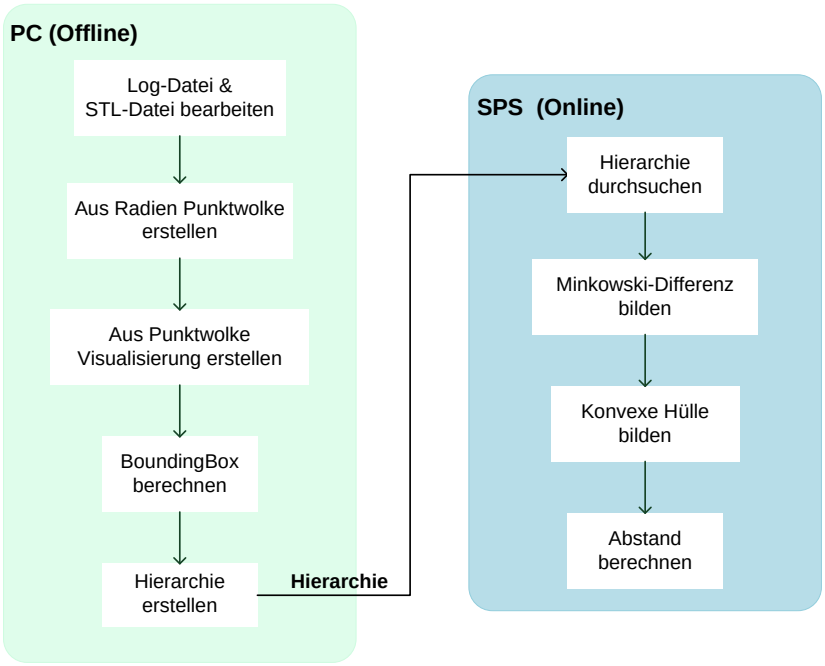


Abbildung 4.1.: Systemarchitektur des Kollisionserkennungssystems

5. Realisierung

Auf die Auswahl des geeigneten Algorithmus, der geeigneten Systemarchitektur und Modularisierung folgt in diesem Kapitel die Darstellung der Implementierung der Kollisionsvermeidung für die YXLON CT-Anlage.

5.1. Datenquelle

Wie bereits im Unterkapitel 4.1 Systemarchitektur des Kollisionserkennungssystems erwähnt, stellt der Rekonstruktions-PC mit Hilfe einer Basler-Kamera ein rotationssymmetrisches Bild des Prüfobjekts bereit. Die Informationen zum Bild werden in einer Log-Datei gespeichert und dem Kollisionserkennungssystem zur Verfügung gestellt. Des Weiteren stellt dieser PC die Geometriedaten der Röhre und des Detektors zur Verfügung. Zusammen bilden diese Geometrie- und Log-Daten die Quelldaten für das entwickelte Kollisionserkennungssystem.

5.2. Aufbereitung der Quelldaten

Bevor die o.g. Rohdaten vom Kollsionserkennungssystem genutzt werden können, müssen diese zunächst in eine Form gebracht werden, die für die Echtzeitumgebung besser geeignet ist. So werden die Prüfobjekte in der Log-Datei als übereinander gestapelte kreisförmige Scheiben repräsentiert, die sich allein mit Informationen über die Höhe und den maximalen Objektradius des Prüfobjektes beschreiben lassen[1]. Diese Darstellung ist zweckmäßig, aber als Objekt-Repräsentation im Kollsionserkennungssystem ungeeignet. Dieses benötigt zur Bildung der Objekt-Hierarchie und zur Distanzmessung mittels der Minkowski-Differenz eine Repräsentation der Oberfläche (nicht des Volumens) des Prüfobjekts im Raum.
Dazu wird aus der Höhe (Z-Ausdehnung) und den Radien der Log-Datei eine Punktwolke extrahiert, welche die rotationssymmetrische Oberfläche des Objektes darstellt. Die Punktemenge wird in einem nächsten Schritt dann so lange verringert, bis die Oberfläche durch ein Minimum

[1] Mit dem Objektradius ist der Radius gemeint, den das Prüfobjekt ausgehend von dem Rotationszentrum bei einer Rotation maximal einnehmen kann. Dieser Radius ist je nach Messhöhe variabel.

von 3D-Koordinaten beschrieben werden kann. Um die so erzeugten Daten effizient verarbeiten und visualisieren zu können, wurde als Datentype ein Mehrdimensionales Array ($nx3$ Matrix) verwendet. Dabei stehen in jeder Zeile die Eckpunkte der zu bearbeitenden 3D-Geometrie des Prüfobjekts (Dargestellt als Punkte im Raum mit(X, Y, Z)-Koordinaten).

Die Abbildung 5.1 zeigt die drei Schritte zur Bearbeitung der Log-Datei, um diese für den Anwendungsfall anzupassen.

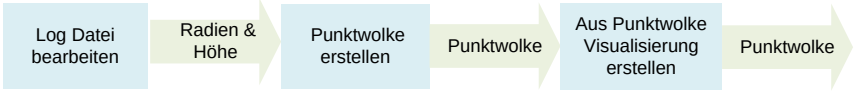


Abbildung 5.1.: Berechnungsschritte zur Bearbeitung der Log-Datei

Ähnlich wird mit den geometrischen STL-Dateien von Röhre und Detektor verfahren.
Nach dieser - auf dem leistungsfähigeren Rekonstruktions-PC durchgeführten - Aufbereitung der Rohdaten stehen die für das zweite Modul bereit. Dieses realisiert das die Repräsentation der Kollisionsobjekte als quaderförmige Hüllkörper.

5.3. Oriented Bounding Boxes-Berechnung

Wie in Unterkapitel 3.3 Hüllkörper (Bounding Volume) beschrieben, ist die Berechnung eines OBBs[2] für ein Objekt bzw. für eine Punktwolke nicht trivial. Die OBBs müssen nach dem zu umhüllenden Objekt ausgerichtet sein. Das heißt, wenn dieses bspw. ein Stift ist, muss eine der OBB-Achsen nach dessen längster Seite ausgerichtet werden. Laut (Gottschalk u. a., 1996b) wird dies erreicht, indem man die statistische Verteilung der Geometrie des zu umhüllenden Objekts untersucht. Dazu müssen die drei folgenden Parameter berechnet werden:

- Mittelwert,
- Orientierung,
- Ausdehnung.

Entsprechend wird der Mittelpunkt und die Kovarianzmatrix[3] der Punktwolke berechnet, um deren Symmetrieachse, Form und Gestalt zur approximieren. Die Kovarianzmatrix wird auch zur Bestimmung der Orientierung und der Ausdehnung der OBBs verwendet.

[2] Oriented Bounding Box - im Folgenden wird hier nur noch die Abkürzung OBB verwendet.
[3] Der Mittelpunkt und die Kovarianzmatrix sind ausschließlich die Verallgemeinerungen der normalen Glockenkurve für N-Dimensionen.

Damit gilt, wenn P_i die Eckpunkte der zu bearbeitenden 3D-Geometrie sind:

$$Punktwolke = P_i = \begin{pmatrix} X_i & Y_i & Z_i \end{pmatrix}$$

Mit ihrem Mittelpunkt:

$$\bar{P} = \frac{1}{N} \sum_{i=0}^{N} P_i$$

Dann lässt sich die Punktwolke in den Mittelpunkt verschieben:

$$P_{i-New} = P_i - \bar{P}$$

Um danach die Kovarianzmatrix zu berechnen:

$$C(P_{i-New}) = \sum_{i=0}^{N} (P_i - P_{i-New}).(P_i - P_{i-New})^T$$

Die Kovarianzmatrix enthält die Eigenwerte der Punktewolke mit ihren zugehörigen Eigenvektoren. Die Seitenflächen der OBBs verlaufen per Definition parallel zu den Eigenvektoren. Deswegen müssten für die Identifikation der OBB-Parameter abschließend nur noch die Eigenvektoren der Kovarianzmatrix berechnet werden. Aber eine bessere Möglichkeit bietet die Methode der Singulärwertzerlegung. Grundsätzlich können die Singulärwerte interpretiert werden wie die Eigenwerte symmetrischer Matrizen. Jedoch liegt die Stärke dieser Methode zum einen darin, dass sie nicht auf quadratische Matrizen beschränkt ist.

Diesem Ansatz folgend, wurde mit Hilfe der Singulärwertzerlegung die Kovarianzmatrix als Produkt dreier Matrizen dargestellt. Hierfür wurde die Matlab-Funktion $[U, S, V] = svd(C)$ (svd: Singular value decomposition) benutzt. Dann wird die verschobene Punktewolke in ihrem Koordinatensystem rotiert, so dass die Rotationsachse der Punktewolke auf der Y-Achse des Koordinatensytems zu liegen kommt.

$$P_{rotate} = (U * P'_{i-New})$$

Durch eine Min-Max-Suche werden die Eckpunkte des Quaders bestimmt, welcher diese Punktewolke umhüllt. Anschließend werden die Koordinaten der Punktewolke und des Quaders in das Weltkoordinatensystem (welches auch das Koordinatensystem der CT-Systems ist) übertragen. Der so entstandene Gerichtete Quader umhüllt nun die das Prüfobjekt repräsentierende Punktwolke.

Alle beschriebenen Berechnungen werden durch ein Software-Modul durchgeführt. Die entwickelte Funktion ***orientedBox3D()*** bekommt als Eingangsdaten die Punktwolke und berechnet den dazugehörigen OBB. Die Abbildung 5.2 zeigt zwei Beispiele von Gerichteten Quadern um zwei Punktwolken.

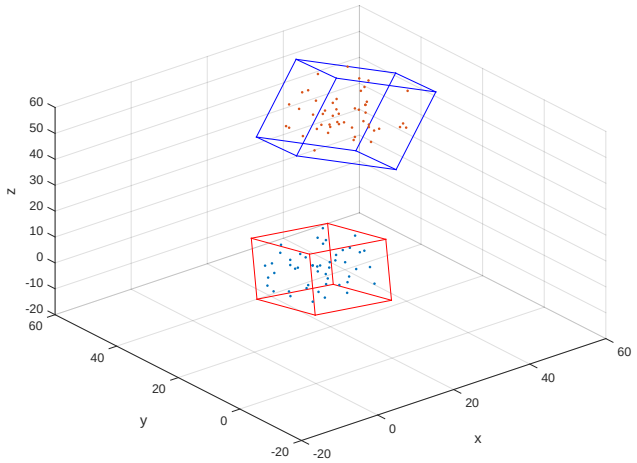


Abbildung 5.2.: OBB zweier Punktwolken

5.4. Aufbau der OBB-Hierarchie

Grundsätzlich ist es selten optimal, ein 3D-Objekt mit einem einzelnen Hüllkörper zu umschließen. Optimierung lässt sich erzielen, wenn das Objekt aus mehreren kleineren Hüllkörpern (OBBs) zusammengesetzt, und die Repräsentation hierarchisch aufgebaut wird. Wie schon erwähnt, wurde die Hierarchie nach der Top Down-Methode erstellt. Das bedeutet, die Wurzel des Hierarchiebaums besteht aus einen OBB, der alle seine Kinder und das gesamte 3D-Objekt umschließt. Alle untergeordneten OBBs lassen sich mit dem oben beschriebenen Verfahren berechnen und ausrichten.

5.4.1. Konstruktion der Hierarchie

Die Qualität der OBB-Hierarchie und demzufolge die Qualität der Objekt-Repräsentation hängt entscheidend von der Regel ab, nach der die Punktwolken der Objekte aufgeteilt werden.

Um die optimale OBB-Hierarchie anzuwenden, wurde daher eine gründliche Untersuchung bezüglich der geeigneten Aufteilung der Punktwolke bzw. der OBBs durchgeführt.
Es hat sich dabei herausgestellt, dass die beste Methode zur Aufteilung der OBBs das Halbieren der OBBs entlang ihrer längsten Achse ist. Auf diese Weise generierte OBBs zeigten die beste Hülleffizienz. Nach dieser Strategie wurde die Aufteilung der OBBs daher fortan durchgeführt.

Ein Beispiel stellt die Abbildung 5.3 dar.

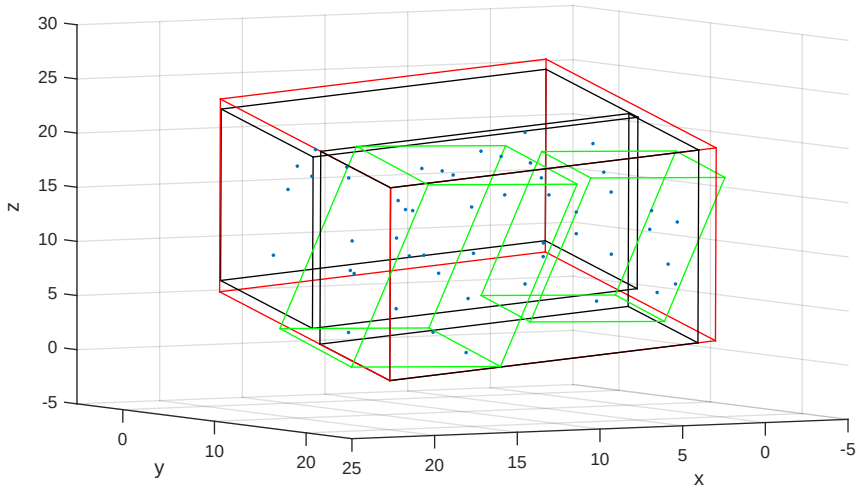


Abbildung 5.3.: OBB-Hierarchie

Der in rot dargestellte OBB ist die Wurzel der Hierarchie. Dieser wird dann in zwei weitere OBBs aufgeteilt, welche in schwarz dargestellt sind. Jeder schwarze OBB wird in zwei weitere OBBs aufgeteilt, die in grün dargestellt sind[4]. Dieser Vorgang wird solange wiederholt, bis entweder die Anzahl der OBB-Punkte in der untersten Ebene der Hierarchie kleiner als vier ist oder das Volumen der berechneten OBBs ein gewünschtes prozentuales Verhältnis zum Volumen des Wurzel-OBBs besitzt. Dadurch ist es möglich, die Präzision der Objektrepräsentation beliebig zu beeinflussen. Die Funktion ***obbSplit3D()*** implementiert die Aufteilung der OBBs.

Da die Konstruktion der OBB-Hierarchie für komplexe 3D-Objekte ein recht aufwendiger Prozess ist, ist es sinnvoll, diesen Prozess nur einmal auszuführen und das Ergebnis zu speichern. Für die Speicherung der OBB-Hierarchie wurde die Matlab-Funktion ***tree()*** benutzt. Diese ist

[4] Um eine bessere Visualisierung zu erzielen, wurden für dieses Beispiel nur die Kinder von einem der beiden schwarzen OBBs dargestellt.

eine hierarchische Datenstruktur, bei der jeder Knoten genau einen Vater und zwei Kindern hat. Die Abbildung 5.4 zeigt die Visualisierung einer Hierarchie für eine Punktwolke mit 50 Zufallspunkten im 3D-Raum.

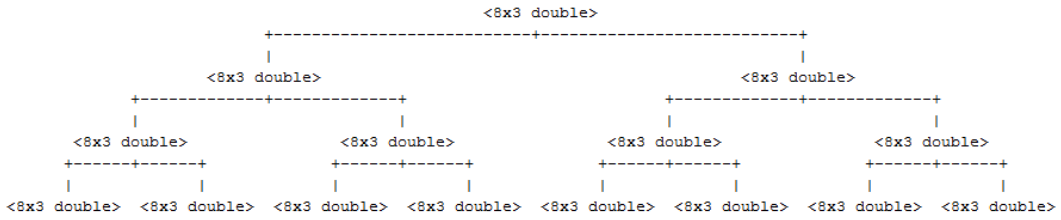


Abbildung 5.4.: Hierarchische Baumstruktur

An jedem Konten wurden die Eckpunkte der von der Funktion obbSplit3D() berechneten OBBs hinzugefügt, welche das Datenformat einer $8x3 - Matrix$ haben. Der Prozess der OBB-Erstellung bis zur Konstruktion der OBB-Hierarchie wurde durch den rekursiven Aufruf der Funktion ***creatTree3D()*** ausgeführt. Eine unkomplizierte Möglichkeit, eine Hierarchie aufzubauen, ist das Rekursive Zusammenfassen. Dieser Algorithmus liefert zu einer gegebenen Szene - bestehend aus einer Punktwolke - einen Hierarchiebaum zurück. Anschließend werden diese Daten inklusive der Informationen über den Hierarchie-Aufbau in eine Text-Datei geschrieben, die später auf die SPS übertragen wird. In der Abbildung 5.5 wird ein kleiner Ausschnitt von der Text-Datei dargestellt. Jede Zeile entspricht einem Knoten, dargestellt durch die acht OBB-Koordinaten, die Kinder-ID des jeweiligen Vaters, sowie den dazugehörige Index der Hierarchieebene.

```
53.1829 47.7226 26.5911 … 54.5289 48.4963 27.4149 2 3 1
41.1888 35.7285 26.5911 … 51.1674 48.5585 27.4771 4 5 2
52.8654 49.2499 37.7489 … 48.7935 45.5102 31.5513 10 11 2
25.7322 24.5144 43.3360 … 51.7688 46.0087 43.5846 6 7 3
28.3096 26.6653 36.9058 … 51.5971 47.5162 44.2429 8 9 3
34.4402 42.2292 38.8880 … 42.1253 50.2584 48.1412 -1 -1 3
```

Abbildung 5.5.: : Ausschnitt aus der Text-Datei zur Verwaltung der OBB-Hierarchie. Die in Rot dargestellten Zahlen sind die IDs der Kinder, die dieser Knoten hat. Die grüne Ziffer bezeichnet die Hierarchieebene $(0 \ldots n)$.

5.4.2. Verifizierung der OBB-Hierarchie

Um sicherzustellen, dass die durch das oben beschriebene Modul konstruierte OBB-Hierarchie die Anforderung in der Praxis erfüllt, wurde eine Verifizierung des Programmcodes anhand synthetischer Testdaten vorgenommen. So wurden bestimmte Testszenarien generiert und überprüft, ob die erstellte OBB-Hierarchie der vorgegebenen Spezifikation entspricht. Beispielweise, ob die Aufteilung der OBBs an der richtigen Stelle erfolgt. In der Abbildung 5.6 wird einer dieser Tests dargestellt - der abgebildete Test visualisiert die Aufteilung der OBBs. Es ist zu erkennen, dass sowohl die Aufteilung, als auch Ausrichtung der OBBs mit den Erwartungen übereinstimmen.

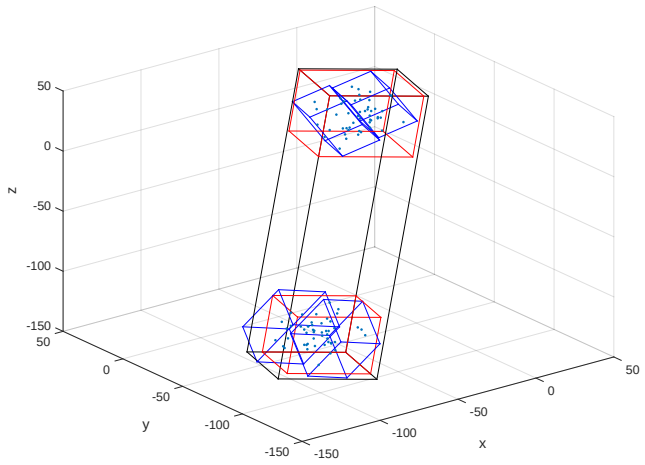


Abbildung 5.6.: Synthetische Testdaten für die Verifizierung der OBB-Hierarchie

Nach der erfolgreichen Implementierung der OBB-Hierarchie wurden die ersten Tests bezüglich des Speicherbedarfs der SPS zur Speicherung der Objekt-Repräsentationen durchgeführt, deren Ergebnisse in Kapitel 6 Exemplarische Implementierung und Testergebnisse näher erläutert werden.

Nachdem nun das zweite Modul der Kollisionsvermeidung - die Erzeugung der OBB-Hierarchien - einsatzbereit ist , soll im Folgenden erörtert werden, wie die Abstandsmessung zwischen zwei OBB-Hierarchien bzw. Objekten in Echtzeit möglichst effizient durchgeführt werden kann.

5.5. Abstandsmessung mit der Minkowski-Differenz

Die beiden ersten Module zur Objekt-Repräsentation und zur Erzeugung der OBB-Hierarchien wurden auf dem PC unter Verwendung von Matlab realisiert. Auch zur Laufzeit können die Berechnungen dieser Module auf dem leistungsstarken PC durchgeführt werden. Dagegen muss das dritte Modul, die Abstandsmessung zwischen den OBB-Hierarchien in Echtzeit, auf einer SPS lauffähig sein, was an den Algorithmus ungleich härtere Anforderungen stellt.

Der hier vorgestellte Prototyp des Abstandsmessungs-Moduls wurde zunächst ebenfalls in Matlab entwickelt und getestet. Dabei wurde aber berücksichtigt, dass die angewendeten Funktionen und Datenstrukturen auf einer SPS implementiert werden können.
Aus zeitlichen Gründen konnte die abschließende Übertragung des entwickelten Prototypen auf eine Test-SPS leider nicht mehr realisiert werden.

5.5.1. Traversierungs-Verfahren

Im Unterkapitel 3.4.2 Kollisionstest zwischen zwei Hüllkörper-Hierarchien wurden verschiedene Methoden zur Tiefensuche (Traversierung) in der OBB-Hierarchie vorgestellt. Von den vorgestellten und in der Literatur genannten Methoden kann jedoch keine hier direkt genommen werden, denn diese beantworten lediglich die Frage, ob zwei Objekte (bzw. zwei OBBs) kollidieren oder nicht. Für die Aufgabenstellung dieser Arbeit ist aber der minimale realisierbare Abstand der Objekte von Bedeutung, die Kollisionserkennung ist nachgeordnet. Aufgrund dieser Lücke in der Literatur und Methodik wurde eine neue Strategie zur Abstandsbestimmung entwickelt und implementiert, die auf existierenden Traversierungsstrategien aufbaut. Im Folgenden wird zunächst das angewendete Traversierungs-Verfahren vorgestellt. Dann wird die Realiserung der Abstandsmessung im Traversierungs-Vorgang erläutert.

Bei der Traversierung zweier Hierarchie-Bäume geht es grundsätzlich um die effiziente Untersuchung der Datenstrukturen auf bestimmte Kriterien. Im vorliegenden Fall sollen zwei Objekte im Raum auf die Lage ihres minimalen Abstands untersucht werden.
Dazu wird der erste Messvorgang zwischen den Wurzel-OBBs der beiden 3D-Objekte durchgeführt. Danach folgt die Messung des Abstands zwischen dem Wurzel-OBB von Objekt A (als Referenz-OBB) und der gesamten OBB-Hierarchie des Objekts B. Dieser Prozess läuft bis zur untersten Ebene der Hierarchie B. Am Ende dieses Prozesses ist der OBB von Hierarchie B ermittelt, der den kleinsten Abstand zu dem Referenz-OBB von Hierarchie A hat. Der gleiche Prozess wird dann mit vertauschten Rollen von Hierarchie A und B erneut durchgeführt.

Die Abbildung 5.7 stellt diese Tiefensuche schematisch dar. Wie bereits beschrieben, entspricht jeder Knoten der Hierarchien einem OBB, der weitere OBBs enthält. Bei der Tiefensuche, oder Traversierung, wird zuerst der Anstand zwischen A1 und B1 berechnet. Danach

folgen die Paarungen zwischen A1 und B2 und A1 und B3. Für die Effizienz des Traversierungs-Algorithmus ist es wichtig, mit möglichst wenigen Knoten-Paarungen zum Endergebnis zu kommen. Nach der ersten Paarung stehen nun aber zwei Hierarchiezweige zum Abstieg in der Hierarchie B zur Auswahl. Hier ist der Algorithmus am effizientesten, der in den Zweig absteigt, dessen OBB den kleineren Abstand zum Referenz-OBB A1 hat.
Auf diese Art wird die Suche bis zur untersten Ebene der Hierarchie B durchgeführt, bis am Ende der OBB aus dem Objekt B benannt wird, der den kleinsten Abstand zu A hat (B4 in der Abbildung 5.7). Um sicher zu stellen, dass dies der Fall ist, und nicht beispielsweise B5 zurückgegeben wird, soll die Tiefensuche jetzt noch einmal durchgeführt werden. Diesmal ist der Referenz-OBB der kleinste OBB, der von der vorherigen Suche ermittelt wurde. Dieser Prozess wird solange ausgeführt, bis entweder eine festgelegte Anzahl Iterationen überschritten wird oder der berechneten OBB des Prozesses gleich dem berechneten OBB im vorherigen Prozess ist, anders gesagt, bis die Traversierung konvergiert.

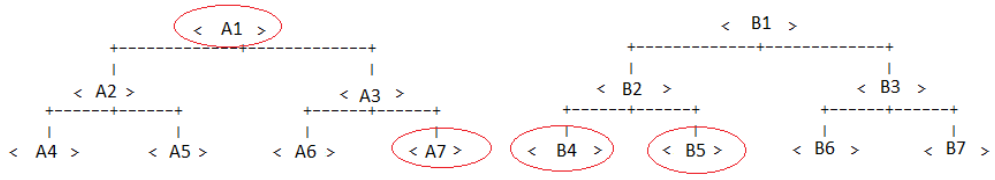


Abbildung 5.7.: Hierarchische Baumstruktur

Der Aufwand für das Traversieren der generierten OBB-Hierarchie nach der entwickelten Methode ist im Gegensatz zu der linearen Methode logarithmisch. Dadurch verbessert sich die Performance erheblich.

Die folgende Abbildung zeigt einen Ausschnitt der Implementierung der Tiefensuche.

```
function [Dist,  kleinstBox, shortestPoint] = getDistanceTwoOBB(A, ObjB)
% Funktion zum Traversieren der OBB-Hierarchie
% Input: Referenz-OBB von Objekt A und die gesamte OBB-Hierarchie von Objekt B
% Output: min Abstand, der kleinste OBB und die Koordinaten des kleinsten OBBs

aktEbene = ObjB (1,:);
shortestPoint = [0 0 0];

while (aktEbene(25)~=-1)
    %Zeilen der nächsten Kinder in der Matrix
    Zeile_Child1 = aktEbene(25);
    Zeile_Child2 = aktEbene(26);

    %nächste Kinder in der Matrix
    Obj_Child1 = ObjB(Zeile_Child1,:);
    Obj_Child2 = ObjB(Zeile_Child2,:);

    %Entfernung beider Kinder zu A bestimmen
    [Dist_Child1, shortestPoint1] = MinkDiff_ConHull(Obj_Child1(1:24), A);
    [Dist_Child2, shortestPoint2] = MinkDiff_ConHull(Obj_Child2(1:24), A);

    %Entscheiden, in welchen Zweig abgestiegen wird
    if (Dist_Child1 <= Dist_Child2)
        aktEbene = ObjB (Zeile_Child1,:);
        aktDistance = Dist_Child1;
        shortestPoint = shortestPoint1;
    else
        aktEbene = ObjB (Zeile_Child2,:);
        aktDistance = Dist_Child2;
        shortestPoint = shortestPoint2;
    end
end

Dist = aktDistance;
kleinstBox = aktEbene;
end
```

getDistanceTwoOBB.m

Die oben erwähnte Abstandsmessung beruht wesentlich auf der Minkowski-Differenz, genauer, ihrer konvexen Hülle. Da die Berechnung der konvexen Hülle im Raum eine der komplizierten Teilaufgaben dieser Masterarbeit war, wird diese hier zuerst näher beschrieben. Anschließend wird das Verfahren zur Abstandsmessung anhand der konvexen Hülle erklärt.

5.5.2. Konvexe Hülle

Laut Definition 4 im Unterkapitel 3.6.1 Minkowski-Summe ist der minimale Abstand zwischen zwei Polygonen A und B äquivalent zum Abstand der Minkowski-Differenz $A \ominus B$ vom Ursprung des Koordinatensystems. Besser gesagt, entspricht der minimale Abstand zwischen A und B dem Abstand der konvexen Hülle der Minkowski-Differenz von A und B vom Ursprung des Koordinatensystems.

QuickHull-Algorithmus 2D

Wie bereits erwähnt, wurde für die Ermittelung der konvexen Hülle der Minkowski-Differenz der QuickHull-Algorithmus angewendet.
Dieser Algorithmus basiert auf dem Prinzip "Teile und Herrsche". Er geht davon aus, dass die Punkte innerhalb einer Hülle nicht selber zur konvexen Hülle gehören können. In 2D besteht das Verfahren aus folgenden Schritten:

1. Die Punktmenge in zwei Teile aufteilen, indem durch die beiden Extremalpunkte eine Gerade gezogen wird.
2. Den Punkt mit maximaler Distanz zur Geraden bestimmen. Dadurch entsteht ein Dreieck.
3. Die Punktmenge innerhalb des Dreiecks wird für die weitere Berechnung nicht mehr beachtet.
4. Ausgehend von den Außenlinien wird diese Dreiecksbildung erneut durchführt, bis die Abbruchbedingung (z.B. Anzahl der Punkte kleiner als drei) erfühlt ist.

Die verbliebene Punktmenge nach der letzten Rekursion bildet die konvexe Hülle.

Das implementierte Programm für QuickHull2D befindet sich auf der beigefügten CD.

QuickHull-Algorithmus 3D

Der QuickHull-Algorithmus für dreidimensionale Punktmengen ist wesentlich komplizierter als in 2D. Das Flussdiagramm in der Abbildung 5.8 zeigt den Programmablauf des implementierten Algorithmus.

Zuerst wird mit Hilfe der Funktion ***getTriangle()*** ein Dreieck mit den drei Extrempunkten der Punktmenge generiert. Anschließend wird durch die Funktion ***splitPoints()*** die Punktwolke entlang der Dreiecksebene in zwei Untermengen aufgeteilt. Für jede Untermenge wird dann eine Pyramide errechnet. Diese bestehen aus der Dreiecksebene und einem vierten Punkt,

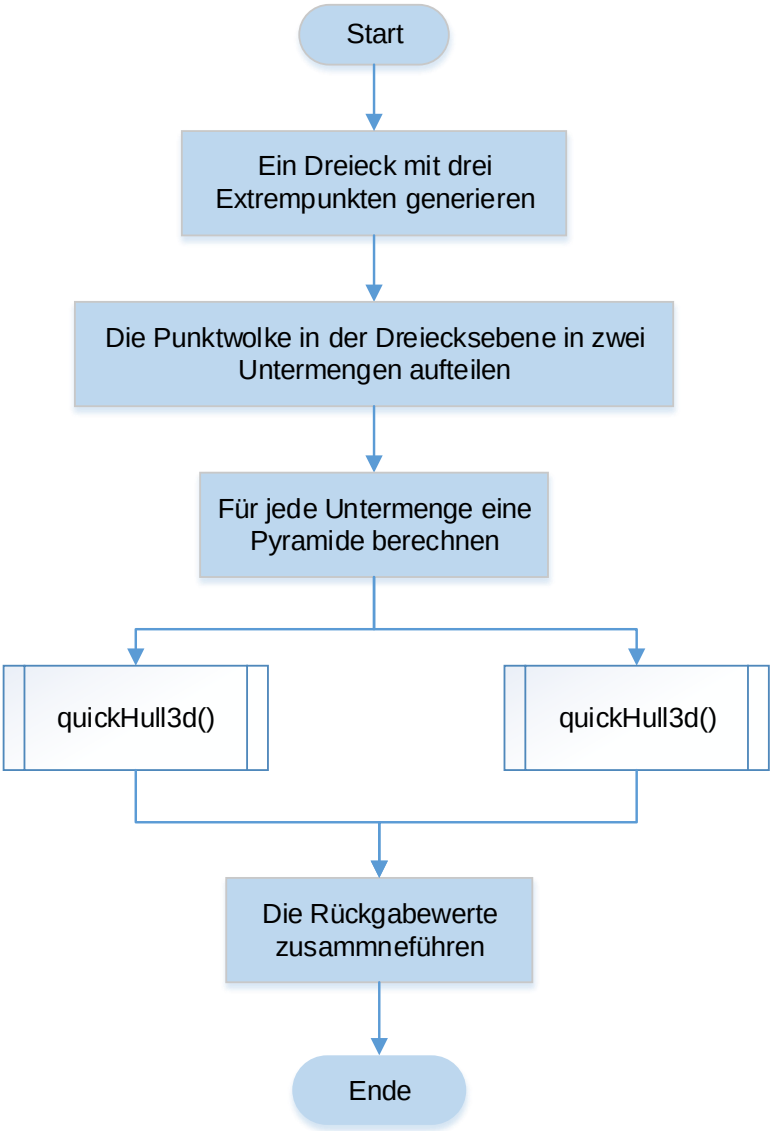


Abbildung 5.8.: QuickHull-Algorithmus zur Berechnung der konvexe Hülle in 3D

der zur Dreiecksebene den größten Abstand hat. Beide Pyramiden werden dann der Funktion ***quickHull3d()*** mit der entsprechenden Untermenge übergeben.

In der Funktion *quickHull3d()* werden diese Schritte jetzt mit der entsprechenden Untermenge der Punktwolke entlang des drei seitlichen Dreiecke der Pyramide wiederholt. Zunächst wird dazu ein Dreieck aus den Punkten 1, 2 und 4 der Pyramide erstellt. Anschließend wird die Punktmenge wieder anhand dieser Dreiecksebene aufgeteilt. Dann wird mit Hilfe der Funktion ***checkUpOrDown()*** geprüft, mit welcher Punktmenge fortgefahren wird. Das Kriterium dabei ist, ob der 4. Punkt der Pyramide ober- oder unterhalb der Dreiecksebene liegt. Liegt er unterhalb der Ebene, wird die obere Teilmenge verwenden, sonst die Untere. Die entsprechende Punktmenge wird in ***newPointCloud*** gespeichert. Wenn diese nicht leer ist, wird eine neue Pyramide daraus generiert und anschließend die Funktion *quickHull3d()* wieder rekursiv aufgerufen. Sollte ***pointPyramid_new*** keine Punkte mehr enthalten, wird das Ganze mit der nächsten Pyramidenseite wiederholt. Sind alle drei Pyramidenseiten abgearbeitet, kehrt die Funktion eine Rekursionsebene zurück. Am Ende der Rekursion beschreiben die verbliebenen Punkte die konvexe Hülle der Punktmenge.
Da bei der Entwicklung des Moduls zur Anstandsmessung die Überprüfung der Minkowski-Differenz-Methode in Vordergrund stand, wurde hier auf eine eigene Implementierung verzichtet und stattdessen auf die Matlab-Implementierung des QuickHull3D-Algorithmus zurückgegriffen.

Die Abbildung 5.9 zeigt das Ergebnis dieser Implementierung für eine Punktwolke aus 50 Zufallswerten.

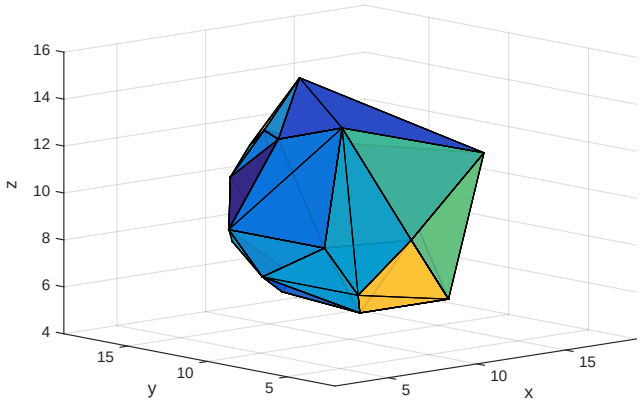
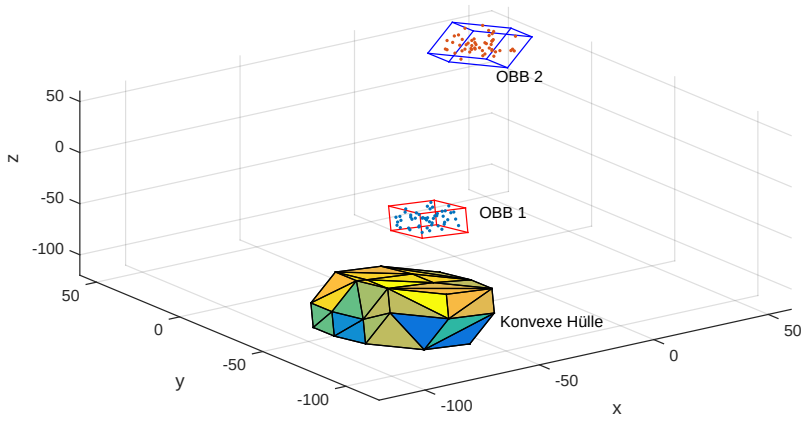


Abbildung 5.9.: Die berechnete konvexe Hülle aus 50 Zufallspunkten

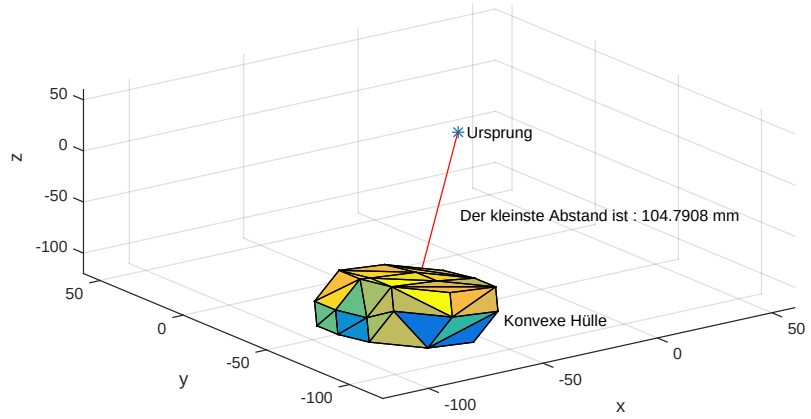
5.5.3. Abstandsberechnung mit der Minkowski-Differenz

Wie schon beschrieben, ist der Abstand zwischen zwei OBBs äquivalent zum Abstand der konvexen Hülle ihrer Minkowski-Differenz zum Ursprung des Koordinatensystems. Im Folgenden wird erläutert, wie diese Abstandsberechnung realisiert wurde.
Die nach dem beschriebenen Vorgehen berechnete konvexe Hülle der Minkowski-Differenz besteht aus Dreiecknetzen, deren Eckpunkte als Matrix gespeichert wurden. Für die Abstandsmessung zwischen dieser Matrix und dem Nullpunkt wurde das Modul ***DisHullToOrigin()*** implementiert. Diese bestimmt für jedes der Dreiecke über den Normalenvektor dessen Abstand zum Ursprung. Für den Fall, dass dabei der Fußpunkt des Normalenvektors außerhalb des Dreiecks liegt, wird der Punkt des betreffenden Dreiecks, der dem Ursprung am nächsten liegt, als kürzester Abstand zurückgegeben.

Die folgende Abbildung zeigt beispielhaft die Bestimmung des kürzesten Abstands zwischen zwei OBBs anhand der Minkowski-Differenz.



(a) Nach der Berechnung der Minkowski-Differenz von OBB1 und OBB2 wird deren konvexe Hülle berechnet



(b) Der minimalen Abstand zwischen OBB 1 und OBB 2 ist äquivalent zu dem Abstand der konvexen Hülle und dem Ursprung des Koordinatensystem.

Abbildung 5.10.: Abstandsmessung mit Minkowski-Differenz

6. Exemplarische Implementierung und Testergebnisse

Um zu beweisen, dass das entwickelte Kollisionsvermeidungssystem technisch realisierbar ist, wurden entsprechenden Programmmodule zunächst in Matlab implementiert und über eine interaktive Benutzeroberfläche getestet. Im Folgenden wird das interaktive Test-Programm kurz vorgestellt. Anschließend werden zwei Ansätze zum Erstellen der OBB-Hierarchie verglichen und die Testergebnisse diskutiert.

6.1. Simulations- und Testkonsole

Um die Funktionsfähigkeit der entwickelten Module und deren Algorithmen während der Entwicklung testen zu können, und um bestimmte Testszenarien zu visualisieren, wurde eine Simulations-Anwendung entwickelt, die über einer grafische Benutzeroberfläche mit Hilfe von Matlab GUIDE[1] zu bedienen ist. Sie ermöglicht die interaktive Eingabe von Test-Objektdaten, die dann grafisch dargestellt und analysiert werden können. Mit Hilfe der Anwendung ließ sich die Funktionsfähigkeit des Prototyps verifizieren, obwohl die Implementierung des dritten Moduls, die Abstandsmessung zwischen den Objekten, nicht mehr aus Zeitgründen auf der SPS Hardware realisiert werden konnte.

Die interaktive Testanwendung wird in der Abbildung 6.1 dargestellt.

Sie ermöglicht, über die Symbolleiste *Open* ein Prüfobjekt auszuwählen. Über die Edit-Felder können beliebige Wunschpositionen und Rotationen des Objektes eingegeben werden.
Dieses Verhalten ist der realen Hardware nachgebildet, die ein Objekt ans eine gewünschte Position fahren kann. Ist die Position eingegeben, wird der kleinste Abstand zwischen zwei Objekten berechnet und in der Konsole ausgegeben.

Des Weiteren werden die folgenden Parameter ausgegeben, die vor allen für den Testbetrieb von Interesse sind:

[1] Die Ankürzung GUIDE steht für Graphical User Interface Design Environment. GUIDE ist ein von Matlab verwendetes Tool zum Erzeugen und Verwalten der GUI-Oberflächen.

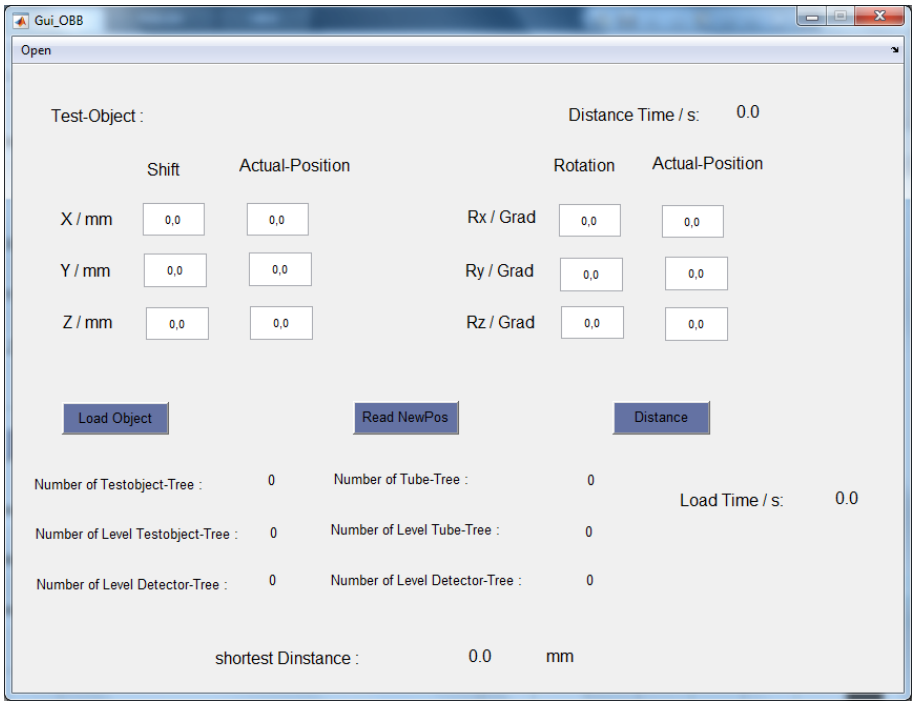


Abbildung 6.1.: Die Testanwendung

- Dauer der Abstandsberechnung,
- Dauer des Ladevorgangs für ein gewähltes Prüfobjekt,
- Aktuelle Positionen (die Position vor der Verschiebung bzw. Rotation),
- Anzahl der Knoten der OBB-Hierarchie,
- Anzahl der erzeugten OBB-Hierarchie.

6.1.1. Transformationen

Um die Lage der Testobjekte im Raum zu verändern, wurden im Prototyp Transformationsmatrizen benutzt. Wie in der Computergrafik verbreitet, wurden die Berechnungen im homogenen Koordinatensystem durchgeführt. Dieses ist eine Erweiterung des üblichen kartesischen Koordinatensystems. Dabei wurden die Eckpunkte des Objektes durch einen Spaltenvektor aus 4 Komponenten beschrieben $P = (P_x, P_y, P_z, 1)^T$. Die notwendigen Transformationen (Translation, Rotation) sind wiederum durch Matrizen realisiert, welche im dreidimensionalen Raum eine $4x4 - Matirx$ benötigen.

Translation

$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Wobei t_x , t_y und t_z jeweils die Verschiebung in der X-, Y- und Z-Richtung sind, die von dem Benutzer der Testanwendung eingegeben werden.

Die Rotationsmatrizen für die Rotation um die drei Koordinatenachsen (X, Y, Z) sehen wie folgt aus:

Rotation um die X-Achse

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) & 0 \\ 0 & \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die Y-Achse

$$R_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation um die Z-Achse

$$R_z(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Koordinaten des transformierten Objektes ergeben sich aus der Multiplikation der Objektkoordinaten mit der Transformationsmatrix:

$$Trans_M = T * R_x(\theta) * R_y(\theta) * R_z(\theta)$$

$$P_{New} = Trans_M * P$$

Die Funktion ***ObjectTransforamtion3D()*** wurde für die Realisierung dieses Verfahrens entworfen.

6.2. Vergleich zweier Hierarchie-Bibliotheken

Es existieren verschiedene kostenlose und quelloffene Kollisions-Bibliotheken, die zu Forschungszwecken verwendet werden können. Eine davon ist die RAPID-Bibliothek[2]. Der RAPID-Algorithmus kann mit unstrukturierten Mengen von Polygonen ohne topologische Einschränkungen arbeiten. Über diesen wird eine Bottom-Up-Hierarchie erstellt. Leider bietet RAPID keine Möglichkeit für die Abstandsberechnung an. Nichtsdestotrotz wurde im Rahmen dieser Arbeit zu Untersuchungszwecken diese Bibliothek zur Erstellung der OBB-Hierarchie verwendet, um einen direkten Vergleich mit dem speziell für diese Masterarbeit entwickelten ***VIIDA-Algorithmus*** anstellen zu können.
Die RAPID-Bibliothek wurde von `http://gamma.cs.unc.edu/OBB/` heruntergeladen und ist auch auf der beigefügten CD gespeichert (Gottschalk u. a., 1996a).

Es wurden drei Testobjekte (ein Kugelschreiber mit seinem Objekthalter, ein Karbon-Stab ebenfalls mit seinem Objekthalter, und die Röntgenröhre) untersucht und deren OBB-Hierarchie mit beiden Anwendungsprogrammen generiert. In der folgenden Tabelle wird die Anzahl der erzeugten OBBs für den jeweiligen Algorithmus verglichen.

Testobjekt	RAPID-Bibliothek	VIIDA-Algorithmus
Kugelschreiber	1973	439
Karbon-Stab	1686	375
Röhre	1595	355

Tabelle 6.1.: Vergleich Anzahl der erstellten OBBs

Es ist zu erkennen, dass die Anzahl der von der RAPID-Bibliothek erstellten OBBs etwa dem Fünffache der von VIIDA[3] generierten OBBs entspricht. Dies bedeutet, der Vorteil der RAPID-Bibliothek liegt in einer größeren Genauigkeit der Objektrepräsentation. Der Nachteil aber ist,

[2] Die Abkürzung RAPID steht für Robust and Accurate Polygon Interference Detection.
[3] VIIDA steht für Very Intelligent Interference Detection Algorithm.

dass die RAPID-Bibliothek keine Möglichkeit anbietet, die Genauigkeit der Objektrepräsentation anzupassen. Dies ist aber bei dem im Rahmen dieser Masterarbeit entwickelten Anwendungsprogramm möglich. Dadurch hat man die Möglichkeit in der Testphase auf der SPS verschiedene Präzisionsstufen der Objektrepräsentation zu testen und deren Laufzeitverhalten zu analysieren. Diese ermöglicht die Optimierung der Objektrepräsentation auf die beschränkte Rechenleistubg der SPS. Ein weiterer Vorteil des entwickelten Programms ist der geringe Speicherplatz, den die OBB-Hierarchie auf der SPS benötigt. Darauf wird in Abschnitt 6.3.3 Speicherbedarf SPS eingegangen.

6.3. Tests

Im Rahmen dieser Masterarbeit wurden sowohl während als auch nach der Entwicklungsphase zahlreiche Tests und Validierungen durchgeführt. Diese Tests und Simulationen sollten zum Beispiel sicherstellen, dass die der Kollisionserkennung zugrundeliegende Visualisierung korrekt arbeitet und die Abstandsmessung korrekte Ergebnisse liefert. Für die Umsetzung der Echtzeitberechnung auf der SPS musste geprüft werden, ob der vorgesehene Lösungsansatz mit der beschränkten Speicherkapazität auf der SPS kompatibel ist.

6.3.1. Visualisierung des Systems

Wie schon beschrieben, erfolgt die Visualisierung der Kollisionsvermeidung durch die Matlab gestützte Testanwendung. Für die Darstellung der Röhre wurde die von YXLON zur Verfügung gestellte STL-Datei aufbereitet und mit einem 3D Plot in Matlab visualisiert. Das Prüfobjekt wurde basierend auf den Daten der Basler-Kamera ebenfalls in Matlab visualisiert. Um sicher zu stellen, dass die in Matlab visualisierten Objekte eine korrekte Darstellung aufweisen, wurde die Darstellung mit der 3D-Grafiksoftware *Blender*[4] verglichen. Diese kann STL-Daten direkt verarbeiten und darstellen.
Die Abbildung 6.2 zeigt den inneren Raum der YXLON FF20-Anlage. Auf der rechten Seite befindet sich der Röntgenröhre und links das Prüfobjekt, welches in diesem Fall aus einem Kugelschreiber und dessen Objekthalter besteht.

Beim Vergleich der beiden Darstellungen zeigt sich, dass Matlab und Blender die Daten identisch interpretieren. Es darf also von einer korrekten Vorverarbeitung der Rohdaten durch den Prototypen ausgegangen werden.

[4] Blender ist eine Software zur Modellierung von dreidimensionalen Körpern. Sie kann STL-Dateien verarbeiten.

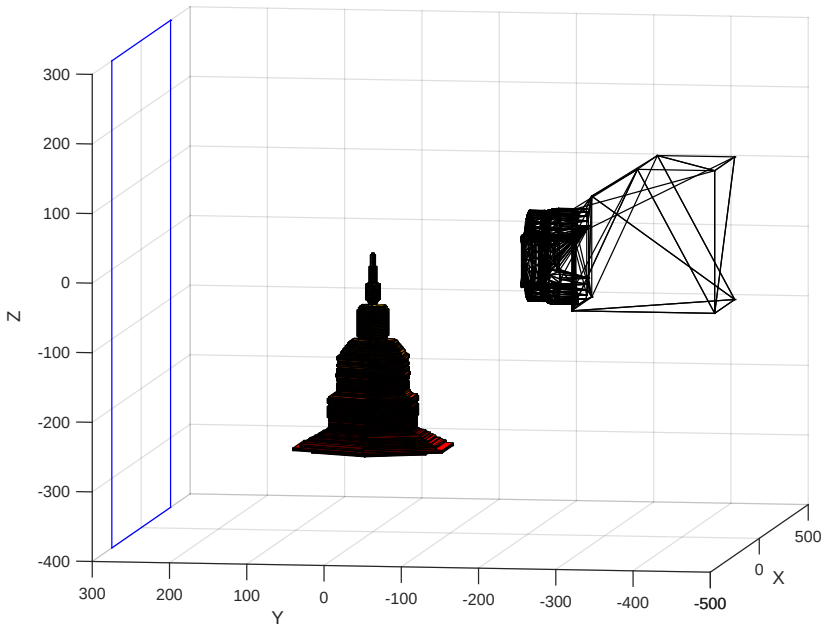


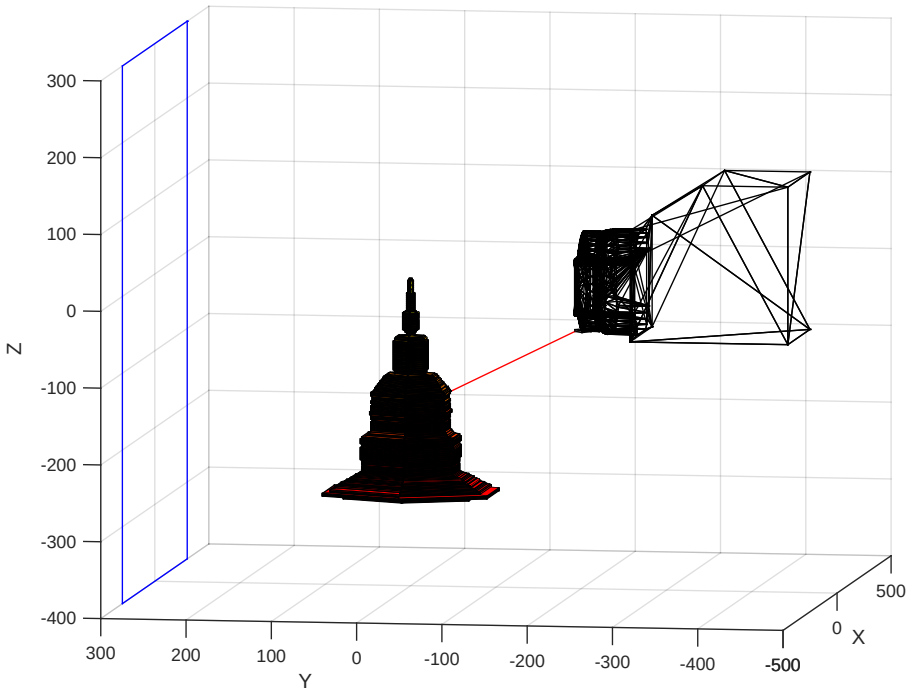
Abbildung 6.2.: Eine Beispielszene für das Kollisionsvermeidungssystem. Es stellt den inneren Raum der YXLON FF20-Anlage dar.

6.3.2. Abstandsmessung

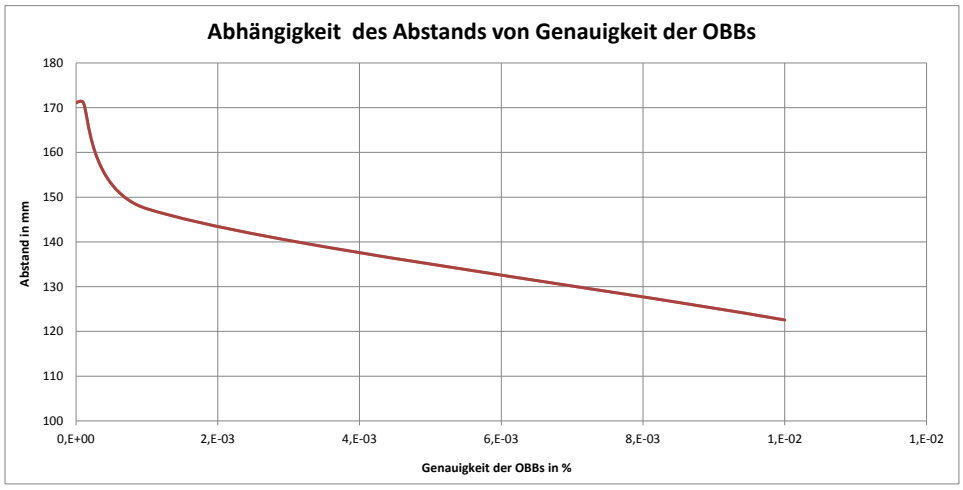
Wie bereits beschrieben, wurde die Abstandsmessung mit Hilfe der Minkowski-Differenz realisiert. In der Testanwendung wird nach der Berechnung des kleinsten Abstands zwischen zwei Objekten - z.B. Röhre und Prüfobjekt - diese Entfernung entsprechend durch eine rote Linie visualisiert, wie auch die Abbildung 6.3(a) zeigt. Bei der Abstandsmessung wurde hier nur das Prüfobjekt und die Röntgenröhre betrachtet.

Um zu untersuchen, wie sich der kleinste Abstand zwischen zwei Objekten in Abhängigkeit von der Genauigkeit der generierten OBB-Hierarchie ändert, wurde für die verschiedenen Präzisionslevel der kleinste Abstand gemessen und über dem Genauigkeitsgrad aufgetragen. Die Ergebnisse sind in Abbildung 6.3(b) dargestellt.

Wie erwartet, wird der ermittelte Abstand der Objekte ungenauer (d.h., sie sind scheinbar nähre bei einander), wenn die Präzision der generierten OBB-Hierarchie für diese Objekte abnimmt.



(a) Messen des kleinsten Abstand zwischen dem Prüfobjekt und der Röhre



(b) Der Abstand zwischen zwei Objekten wird ungenauer (nimmt ab), wenn die Genauigkeit der generierten OBB-Hierarchie für diese Objekte abnimmt.

Abbildung 6.3.: Abstandsmessung

6.3.3. Speicherbedarf SPS

Des Weiteren gilt es zu klären, ob die berechneten Datensätze zur Objektrepräsentation bezüglich der Speicherkapazität für die SPS geeignet sind. Wie bereits in Abschnitt 2.3.2 Steuerung der FF-Anlagen erwähnt, besitzt die Steuerung X20CP1585, welche in der YXLON CT-Anlage zum Einsatz kommt, einen 256 MByte DDR2-SDRAM Arbeitsspeicher. Für jeden OBB werden neben seinen 8 Eckpunkten noch zwei IDs von seinen Kindern und seine Hierarchie-Ebene in einer Textdatei gespeichert. Das bedeutet, wenn ein Objekt in seiner Höchstauflösung durch 439 OBBs dargestellt wird (wie der Kugelschreiber mit seinem Objekthalter), benötigt er folgenden Speicherplatz:

$$Speicherbedarf = OBB + Kinder + Hierarchie - ID$$

Also:

$$Speicherbedarf = 439 * 8 * 3 * 4Byte + 439 * 2 * 4Byte + 439 * 1 * 4Byte = 0.048MB$$

Dieser Bedarf an Speicherplatz ist technisch akzeptabel und ein Erfolgsergebnis für das entwickelte Kollisionsvermeidungssystem.

7. Zusammenfassung

Die Aufgabenstellung dieser Masterarbeit bestand in der Entwicklung eines Prototyps zur steuerungsbasierten Kollisionsvermeidung von komplexen 3D Objekten für die CT-Anlagen der Firma YXLON.

Zuerst wurde bei einer Literaturrecherche eine systematische Auswahl der geeigneten Fachliteratur getroffen. Danach wurden durch die Analyse des vorhandenen Kollisionsvermeidungssystems bei YXLON und basierend auf der Fachliteratur die technischen Anforderungen an den Kollisionsvermeidungs-Algorithmus hergeleitet.

Gemäß der getroffenen Spezifikation wurde dann ein verlässliches und robustes Kollisionsvermeidungssystem für CT-Anlagen in der Entwicklungsumgebung Matlab entwickelt. Dieses setzt sich aus drei Modulen zusammen: zwei Module, die für die 3D Objekt-Repräsentation verantwortlich sind und ein Modul zur Abstandsbestimmung zwischen den Kollisionsobjekten in Echtzeit. Während die Objekt-Repräsentation Offline berechnet wird, erfolgt die Abstandsbestimmung in Echtzeit auf der SPS. Durch den modularen Software-Aufbau ist es einfach möglich, das System mit weiteren Funktionen zu erweitern.

Für eine präzise Repräsentation der 3D Objekte wurde mit einer Hüllkörper-Hierarchie nach dem Top Down-Verfahren gearbeitet, welche die komplexen Objekte als Schachtelung von einfachen Geometrischen Körpern vereinfacht. Die Wahl des zur Repräsentation verwendeten Hüllkörpers fiel auf den Gerichteten Quader (Oriented Bounding Boxes, OBB). OBBs bieten die beste Hülleffizienz zur Lösung der gestellten Aufgaben. Auch die geforderte Abstandsmessung lässt sich mit Hilfe von OBBs effizient realisieren. Diese OBB-Hierarchie wird offline erzeugt und als Matrix in einer Textdatei gespeichert, die auf die SPS übertragen wird. Auf Grundlagen dieser Daten wird die Echtzeit-Kollisionserkennung und -schutz während des Röntgenvorgangs berechnet. Als Erweiterung gegenüber handelsüblichen Kollsionsschutz-Anwendungen erlaubt es die hier entwickelte Anwendung dem Benutzer, die Genauigkeit der 3D Objekt-Repräsentation zu konfigurieren und an die technischen Erfordernisse anzupassen.

Basierend auf den vorgestellten Verfahren zur Objekt-Repräsentation wurde eine schnelle und robuste Abstandsmessung implementiert, die mit Hilfe der Minkowski-Differenz zur Laufzeit den kürzesten Abstand zwischen den Kollisionsobjekten berechnet. Dabei wurden die in der Literatur bekannten Methoden der Hierarchie-Traversierung erweitert, damit sie der Anforderung der strikten Kollisionsvermeidung gerecht werden und nicht, wie bekannte Algorithmen, lediglich eine Kollision detektieren.

Das komplette Kollisionsvermeidungssystem wurde mit Hilfe von Matlab simuliert, um zu beweisen, dass der Kollisionsvermeidungs-Algorithmus technisch realisierbar ist.
Aufgrund der unvorhersehbaren Komplexität der gestellte Aufgabe dieser Masterarbeit wurde die Implementierung auf der SPS nicht mehr durchgeführt. Dennoch wurde das Anwendungsprogramm so realisiert, dass es durch die Verwendung primitiver Rechenoperationen im Programm für die SPS technisch realisierbar ist. Die relevanten mathematischen Funktionen für die Abstandsmessung wie z.B. Matrizen-Rechnung wurden auch für SPS implementiert und getestet. Alle Testergebnisse haben die Erwartungen erfüllt.

7.1. Fazit

Diese Masterarbeit soll der YXLON International GmbH einen Überblick über die möglichen Kollisionsvermeidungs-Algorithmen für komplexe 3D Objekte verschaffen, um anstelle der aktuellen, nicht steuerungsbasierten Lösung, ein echtzeitfähiges System zur Kollisionsvermeidung anbieten zu können. Mit der vorliegenden Arbeit ist es gelungen, den Prototypen eines verlässlichen und robusten Kollisionsvermeidungssystems zu entwickeln, zu implementieren und in Matlab zu simulieren.

7.2. Ausblick

Im Rahmen dieser Masterarbeit konnten zwei der drei spezifizierten Anwendungsmodule in ihrer Anwendungsumgebung realisiert werden. Das dritte Modul, welches die Abstandsberechnung der Kollisionsobjekte in Echtzeit auf der SPS durchführen soll, wurde bereits als Prototyp in der Entwicklungsumgebung von Matlab implementiert und die Anwendbarkeit der Minkowski-Differenz zur Abstandsberechnung getestet. Die technische Implementierung des lauffähigen Algorithmus auf der SPS wäre eine logische Fortführung dieser Masterarbeit.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Barber u. a. December 1996] BARBER, C. B. ; DOBKIN, DAVID P. ; HUHDANPAA, HANNU: The Quickhull Algorithm for Convex Hulls. (December 1996)

[Bender und Brill 2006] BENDER, Michael ; BRILL, Manfred: *Computergrafik: ein anwendungsorientiertes Lehrbuch*. München Verlag, 2006

[Bergen 1998] BERGEN, Gino Van D.: Effcient Collision Detection of Complex Deformable Models using AABB Trees. (1998)

[Canny 1986] CANNY, J.F.: Collision Detection for Moving Polyhedra. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 2* (1986), Nr. 29

[Ericson 2005] ERICSON, Christer: *Real-Time Collision Detection*. Morgan Kaufmann, 2005. – ISBN 1-55860-732-3

[Freiburg 2015] FREIBURG, Universität: *Bounding Volume*. Universität Freiburg. 2015. – URL https://sopra.informatik.uni-freiburg.de/soprawiki/Bounding_Volume

[Gottschalk 2000] GOTTSCHALK, Stefan: *Collision queries using oriented bounding boxes*. 2000. – URL http://gamma.cs.unc.edu/users/gottschalk/main.pdf

[Gottschalk u. a. 1996a] GOTTSCHALK, Stefan ; LIN, Ming ; MANOCH, Dinesch: *RAPID-Library*. 1996. – URL http://gamma.cs.unc.edu/OBB/

[Gottschalk u. a. 1996b] GOTTSCHALK, Stefan ; LIN, Ming ; MANOCHA, Dinesch: OBB-Tree: A Hierarchical Structrue for Rapid Interference Detection. In: *University fo North Carolina* (1996), Nr. 27

[Jimenez u. a. 2001] JIMENEZ, P. ; THOMAS, F. ; TORRAS, C.: 3D Collision Detection: A Survey. (2001), Nr. 29

[Joswig und Theobald 2008] JOSWIG, Michael ; THEOBALD, Thorsten: *Algorithmische Geometrie, Polyedrische und algebraische Methoden*. Vieweg & Sohn Verlag, 2008. – ISBN 978-3-8348-0281-1

[Klein 1997] KLEIN, Rolf: *Algorithmische Geometrie*. Addison Wesely Longman Verlag, 1997. – ISBN 3-8273-1111-x

[Klosowski u. a. 1998] KLOSOWSKI, J. T. ; HELD, M. ; MITCHELL, J. S. B. ; SOWIZRAL, H. ; ZIKAN., K.: Effcient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. In: *IEEE Transactions on Visualization and Computer Graphics, 4(1):21-36* (1998), Nr. 21

[Li 1992] LI, Wei: *Grafische Simulation und Kollisionvermeidung von Robotern*. Morgan Kaufmann, 1992. – ISBN 3-528-06494-3

[Mezger 2001] MEZGER, Johannes: *Effziente Kollisionsdetektion in der Simulation von Textilien*. Wilhelm-Schickard-Institut fü Informatik Graphisch-Interaktive Systeme, Universität Tübingen. 2001. – URL http://gris.uni-tuebingen.de/fileadmin/user_upload/Paper/Mezger-2001-Effiziente.pdf

[Wittmann 2011] WITTMANN, Stefan: Verfahren zur Simulation und Analyse der Auswirkungen toleranzbedingter Bauteilabweichungen. In: *Der Technischen Fakultät der Universität Erlangen Nürnberg zur Erlangung des Gradese,* (2011). – URL https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/1641.

[YXLON 2014] YXLON: *Unternehmensprofil*. 2014. – URL http://www.yxlon.de/Resources/About-Us/Unternehmensprofil_de_2014.pdf

[YXLON 2015a] YXLON: *Bedienungsanleitung, YXLON FF20 CT*. YXLON International GmbH. 2015. – URL http://www.yxlon.de/Produkte/CT-Systeme/FF20-CT

[YXLON 2015b] YXLON: *B&R X20CP1585*. B&R Automation. 2015. – URL http://www.br-automation.com/smc/63a5bdb1f3800de4493804a6f5c91cf806a68de4.jpg

[YXLON 2015c] YXLON: *Nexis Architektur Überblick,*. YXLON International GmbH. 2015

[YXLON 2015d] YXLON: *YXLON International GmbH, Produkte*. YXLON International GmbH. 2015. – URL http://www.yxlon.de/Produkte/CT-Systeme/FF20-CT

[Zachmann 2011] ZACHMANN, G.: *Virtuelle Realität Kollisionsdetektion*. Clausthal University, Germany. 2011. – URL cg.in.tu-clausthal.de

A. Hilfsmittel

Inhalt der CD

In der beigefügten CD sind folgende Ordner und Dateien enthalten.

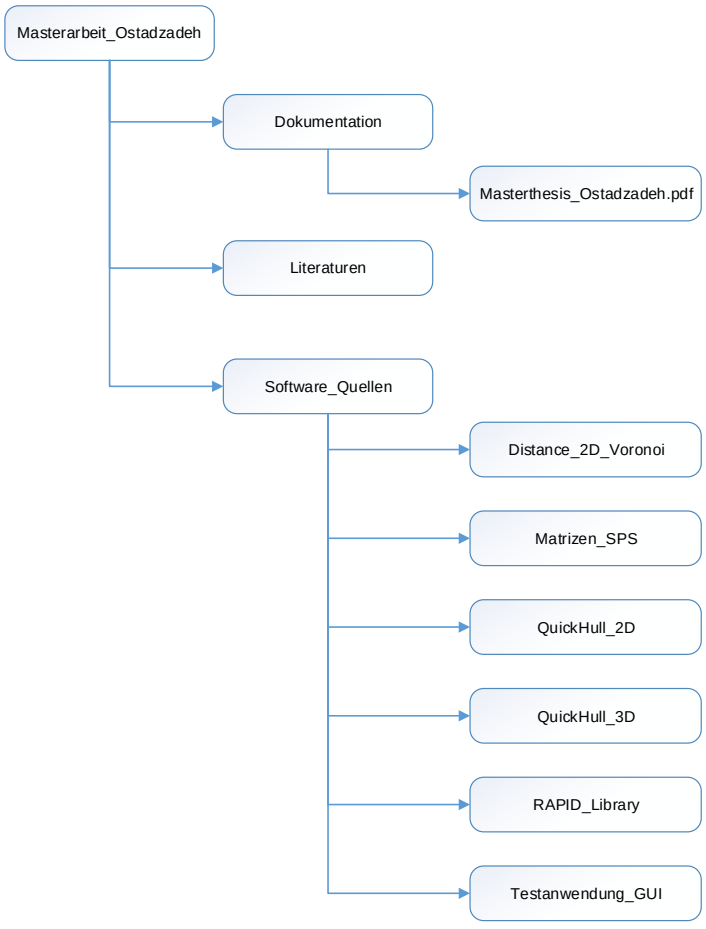


Abbildung A.1.: Ordnerstruktur der beiliegenden CD

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 4. Januar 2016	
Ort, Datum	Unterschrift