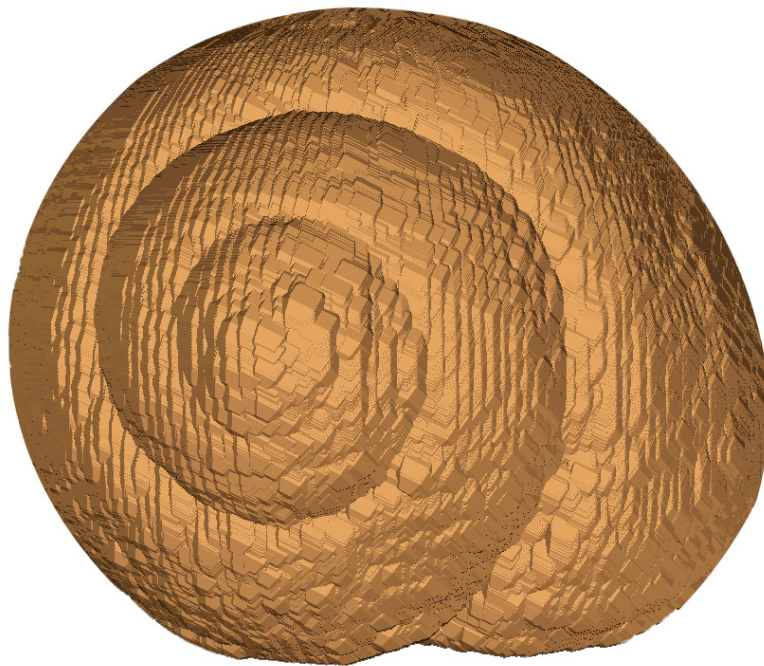


# Technische Realisierung eines 3D-Scanners auf Basis der Lichtschnitt-Technik

Diplomarbeit von Pascal Reolon  
98620107  
Zürich

Betreut von  
Thomas Hübner

Montag, 22. Mai 2006



# Zusammenfassung

Laserscanner für die Erfassung der dreidimensionalen Struktur von Objekten konnten sich bis anhin nicht im Massenmarkt durchsetzen. Da dies vor allem in den hohen Kosten dieser Produkte begründet ist, soll aufgezeigt werden, wie mit vergleichsweise günstigen Komponenten ein 3D-Scanner gebaut werden kann. Neben dem Aufbau der Hardware wird beschrieben über welche frei verfügbaren Softwarelösungen eine Applikation zur Steuerung, Auswertung und Darstellung implementiert werden kann.

# Abstract

Laserscanners for threedimensional shape acquisition have until now not become a mass-market product. Since this is due to the high price level of these devices, the implementation of a 3D-Scanner consisting of comparatively low-priced components will be presented. Besides of the hardware assembly, the generation of the software for control, analysis and display based on free-to use components is shown.

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>5</b>
1.1	Motivation und Ziele . . . . .	5
1.2	Gliederung . . . . .	5
<b>2</b>	<b>Technologien und Verfahren</b>	<b>6</b>
2.1	Definition . . . . .	6
2.2	Konzepte zur räumlichen Abtastung . . . . .	6
2.2.1	Oberflächenabtastung . . . . .	6
2.2.2	Durchleuchtung (Computertomographie) . . . . .	6
<b>3</b>	<b>Bildaquisition</b>	<b>8</b>
3.1	Vorbedingungen . . . . .	8
3.2	Technologien . . . . .	8
3.2.1	Java Media Framework . . . . .	8
3.2.2	QuickTime for Java . . . . .	9
3.2.3	Twain . . . . .	9
3.3	Abwägung . . . . .	10
3.4	Implementation . . . . .	10
3.5	Einschränkungen . . . . .	11
3.6	Bildverarbeitung . . . . .	12
3.6.1	Anforderungen . . . . .	12
3.6.2	Java Advanced Imaging . . . . .	12
<b>4</b>	<b>Bilderkennung</b>	<b>13</b>
4.1	Anforderungen . . . . .	13
4.2	Ansätze . . . . .	13
4.2.1	Kantenerkennung . . . . .	13
4.2.2	Schwellwerte . . . . .	13
4.2.3	Optische Filter . . . . .	14
4.3	Lösung . . . . .	15
4.3.1	Filterung . . . . .	15
4.3.2	Erkennung . . . . .	15
4.3.3	Optimierende Bildoperationen . . . . .	16
4.3.4	Verfeinerung (Sub-Pixel Genauigkeit) . . . . .	17
4.4	Bewertung . . . . .	18
<b>5</b>	<b>Positionsermittlung</b>	<b>19</b>
5.1	Ansätze . . . . .	19
5.1.1	Direkte Kontrolle . . . . .	19
5.1.2	Zeitmessung . . . . .	19
5.1.3	Markierungen . . . . .	19
5.2	Lösung . . . . .	20
5.2.1	Genauigkeit . . . . .	22
5.2.2	Kantendetektor . . . . .	22

5.2.3	Winkelberechnung . . . . .	22
5.2.4	Zuverlässigkeit . . . . .	22
5.2.5	Bestimmung der Drehgeschwindigkeit . . . . .	23
<b>6</b>	<b>Aufbau und Kalibrierung</b>	<b>25</b>
6.1	Anforderungen . . . . .	25
6.2	Umsetzung . . . . .	25
6.2.1	Ausrichtung der Komponenten . . . . .	25
<b>7</b>	<b>Auswertung</b>	<b>27</b>
7.1	Distanzwerte . . . . .	27
7.2	Koordinaten . . . . .	28
7.2.1	Abszisse und Applikate . . . . .	29
7.2.2	Ordinate . . . . .	30
7.3	Erweiterung . . . . .	32
7.4	Ausgabe . . . . .	32
<b>8</b>	<b>Präzision</b>	<b>33</b>
8.1	Auflösung . . . . .	33
8.2	Optische Verzerrung . . . . .	34
<b>9</b>	<b>Benutzeroberfläche</b>	<b>35</b>
<b>10</b>	<b>Vergleichbare Ansätze</b>	<b>38</b>
10.1	Ein interaktiver Laserscanner . . . . .	38
10.2	Einsatz von strukturiertem Licht . . . . .	38
<b>11</b>	<b>Resultate</b>	<b>39</b>
11.1	Geometrische Formen . . . . .	39
11.2	Komplexe Formen . . . . .	39
<b>12</b>	<b>Zusammenfassung</b>	<b>41</b>
12.1	Stand der Arbeiten . . . . .	41
12.2	Weiterentwicklung . . . . .	41
12.3	Ungelöste Fragen . . . . .	41
<b>A</b>	<b>Installation</b>	<b>44</b>
A.1	Kamera . . . . .	44
A.2	Java Runtime . . . . .	44
A.3	Java Advanced Imaging . . . . .	44
A.4	Java Media Framework . . . . .	44
A.5	Java to OpenGL . . . . .	45
A.6	Dateien . . . . .	45

# 1 Einleitung

Die vorliegende Diplomarbeit befasst sich mit der technischen Realisierung eines 3D-Scanners auf Basis der Lichtschnitttechnik und wurde am *Visualization and Multimedia Lab* des Instituts für Informatik der Universität Zürich durchgeführt.

## 1.1 Motivation und Ziele

Obwohl funktionierende 3D-Scanner Systeme kommerziell erhältlich sind, konnten sie sich auf Grund des relativ hohen Preisniveaus nicht im Massenmarkt durchsetzen. Es erscheint deshalb interessant, ein solches Gerät mit günstigen Komponenten zu konstruieren. Das Resultat soll in Bezug auf die Genauigkeit aber durchaus mit teureren Lösungen vergleichbar sein. Ein besonderes Augenmerk liegt deshalb auf der Software, welche die Nachteile der günstigen Komponenten kompensieren soll. Die resultierende Applikation wird über die optische Triangulation die Distanzwerte einer Szene ermitteln und daraus die Punktwolke berechnen. Dieses Modell wird danach am Bildschirm angezeigt und die Rohdaten können exportiert werden für die Weiterverarbeitung in anderen Programmen. Zur Verfügung stehen dazu eine Webcam, ein Laser und ein Drehteller.

## 1.2 Gliederung

**Kapitel 1** Einleitung

**Kapitel 2** Überblick über existierende Technologien und Verfahren

**Kapitel 3** Softwarekomponenten zur Bildakquisition und Bildverarbeitung

**Kapitel 4** Diskussion verschiedener Methoden zur Segmentierung

**Kapitel 5** Verschiedene Ansätze zur Positionsermittlung

**Kapitel 6** Aufbau des Systems und Regeln zur Kalibrierung

**Kapitel 7** Distanzberechnungen mittels optischer Triangulation, Auswertung

**Kapitel 8** Faktoren, welche die erzielbare Präzision beeinflussen

**Kapitel 9** Grafische Benutzeroberfläche

**Kapitel 10** Vergleichbare Konzepte

**Kapitel 11** Präsentation der Scanresultate

**Kapitel 12** Zusammenfassung und Ausblick

## 2 Technologien und Verfahren

### 2.1 Definition

Ein 3D-Scanner digitalisiert dreidimensionale Körper. Als Resultat entsteht eine diskrete Menge von Abtastpunkten, eine Punktwolke. Diese kann danach zur Vermessung des Objektes oder zur Oberflächenrekonstruktion verwendet werden. Im Gegensatz zu 2D-Scannern wird nicht die optische Oberflächenbeschaffenheit, sondern in erster Linie die räumliche Form eines Objektes abgetastet. Als Resultat entsteht ein Modell des Objektes mit Koordinaten für jeden mit der erzielbaren Auflösung erfassbaren Punkt.

### 2.2 Konzepte zur räumlichen Abtastung

#### 2.2.1 Oberflächenabtastung

Die Oberflächenabtastung kann grundsätzlich mit drei verschiedenen Methoden erfolgen, der Triangulation, der Laufzeitmessung und der Differenzmessung.

##### Optische Triangulation

Ein Lichtstrahl wird vom Objekt so reflektiert, dass unter Zugrundelegung geometrischer Gesetzmässigkeiten die Variation des Abstandes des Objekts zu einer Lageänderung des reflektierten Lichtstrahls auf einem positionsempfindlichen Detektor führt. Aus der Lageänderung des Strahls kann die Entfernung bestimmt werden.[Bos05] Dieses Prinzip kann auch beim Streifen-Projektions-Verfahren verwendet werden. Die Erweiterung besteht darin, dass anstatt eines Lichtstreifens mit einem Projektor ein Muster auf das Objekt projiziert wird.

##### Laufzeitmessung

Ein Sender emittiert kurze Impulse oder modellierte Signale. Das vom Objekt reflektierte Licht wird von einem Empfänger aufgenommen. Auf Grund der Laufzeit des Lichtes kann durch eine Zeitmessung der Objektabstand bestimmt werden. [Bos05] Die selbe Methodik kann sowohl mit Licht als auch mit Schall arbeiten. Es ist dabei zu bedenken, dass die Ausbreitungsgeschwindigkeit des Lichtes sehr hoch ist und nur hochentwickelte Geräte als Sender und Empfänger in Frage kommen..

##### Differenzmessung

Mit dem Phasendifferenz-Verfahren wird über die Phasen-Verschiebung zwischen gesendetem und reflektiertem Laserlicht, das auf eine Sinuswelle aufmoduliert wird, die Laufzeit und damit die Entfernung bestimmt. Es lassen sich fast beliebig grosse Entfernungen überbrücken. [Sie01] Zusätzlich zu den Geometrie-Daten liefert das Verfahren auch schwarzweiss Bilder, ist aber völlig unabhängig von den herrschenden Lichtverhältnissen.

#### 2.2.2 Durchleuchtung (Computertomographie)

Im Gegensatz zu den Scanverfahren mit optischen Signalen, verwendet die Computertomographie Röntgenstrahlen (Spiralcomputertomographie) oder Magnetfelder (Magnetresonanztomographie). Bei der Spiralcomputertomographie werden von einer Kreisbahn aus mehrere Röntgenaufnahmen eines

Körpers gemacht. Dabei wird die vom Körper absorbierte Strahlung in Differenz zur Ausgangsstrahlung gemessen. Aus den Intensitätsdaten kann danach das dreidimensionale Modell erstellt werden. Da dieses Verfahren den Nachteil der Strahlenexposition mit sich bringt, wird heute vermehrt die Kernspintomographie eingesetzt. Diese arbeitet mit der physikalischen Eigenschaft, dass die rotierenden Elektronen und Protonen der Wasserstoffatome im Zellkern ein Magnetfeld erzeugen und sich dadurch mit einem externen Magnetfeld ausrichten lassen. Wird auf die ausgerichteten Atome ein kurzer, genau definierter elektromagnetischer Impuls gerichtet, kippen sie kurz aus der Rotationsachse. Während der Rückkehr zum ausgerichteten Zustand senden die Atome Resonanzsignale aus, welche von den selben Magnetspulen erfasst werden. Daraus wird danach das Modell des Objektes rekonstruiert, wobei auch Informationen über die Materialbeschaffenheit gewonnen werden. Eine detaillierte Ausarbeitung zu den physikalischen Grundlagen findet sich in [Tsc96]. Die Computertomographie liefert somit nicht nur die Kontur eines Objektes sondern ist auch in der Lage, zertsörungsfrei und nicht invasiv das Innenleben eines Körpers zu erfassen.



## 3 Bildaquisition

In diesem Kapitel erfolgt eine Auflistung der zur Anbindung der Kamera an den Rechner zur Verfügung stehenden Technologien.

### 3.1 Vorbedingungen

Als Basis für die Evaluation müssen zuerst die von den Konzepten zu erfüllenden Vorbedingungen definiert werden. Aus den lizenzrechtlichen und technischen Anforderungen an das Projekt leiten sich folgende Punkte ab:

- *Lizenzkosten*: Keine Lizenzkosten
- *Schnittstellen*: Einbindung über Java und Eigenständigkeit
- *Funktionalität*: Folgende Anforderungen an die Funktionalität werden definiert:
  - Unterstützung von Multimedia Geräten über die Schnittstellen USB und Firewire
  - Unterstützung einer Auflösung von minimal 640×480 Bildpunkten
  - Anzeige von Video mit 30 Bildern pro Sekunde in Echtzeit
  - Komprimierungsfreie und effiziente Akquisition von Einzelbildern
- *Stabilität*: Robuste Architektur
- *Kompatibilität*: Verfügbarkeit für verschiedene Plattformen (wünschenswert)
- *Komplexität*: Einfache Handhabung trotz hohem Funktionsumfang
- *Dokumentation*: Qualität, Umfang und Zugänglichkeit

Als mögliche Alternativen ergeben sich dadurch folgende Produkte:

- Java Media Framework von Sun Microsystems
- QuickTime for Java von Apple
- Twain von der Twain Working Group

Im den nächsten zwei Abschnitten werden die Eigenschaften dieser Lösungen beschrieben und auf deren Übereinstimmung mit den gestellten Anforderungen geprüft.

### 3.2 Technologien

#### 3.2.1 Java Media Framework

Sun Microsystems stellt mit Java Media Framework (nachfolgend *JMF* genannt) eine Architektur zur Aufzeichnung, Verarbeitung und Anzeige von zeitbasierten Medieninhalten bereit. Dieses Paket steht kostenlos unter der Sun Public License zur Verfügung.

Es wird eine Vielzahl von geläufigen Audio- und Videoformaten unterstützt. Als Datenquelle dient eine Kamera für die direkte Aufnahme oder ein Speichermedium für die Wiedergabe Mediendateien.

Die Datenquelle wird vom *player* interpretiert und im *outputdevice* wiedergegeben [Van02]. Neben der lokalen Verarbeitung und Wiedergabe unterstützt *JMF* auch die Ausstrahlung von Medien über Netzwerke mittels *Real-time Transport Protocol (RTP)*.

Der grosse Vorteil von *JMF* ist, dass die meisten wichtigen Komponenten wie Effekte, Codecs, Wiedergabegeräte und Einzelbildzugriff bereits implementiert und frei zugänglich sind. Somit muss der Programmierer nur die benötigten Komponenten zusammenfügen.

Bei *JMF* handelt es sich um eine pure Java-Umgebung, deren Leistung mittels nativer Bibliotheken gesteigert wird. Somit ist das Paket aber nicht vollständig unabhängig vom Betriebssystem. Komplette Pakete stehen für Windows, Linux und Solaris zur Verfügung. Zusätzlich ist eine plattformübergreifende Version verfügbar, welche nur über die Java-Bibliotheken verfügt und gegebenenfalls unter Apple verwendet werden kann. Diese Möglichkeit wurde im Rahmen dieses Projektes nicht vollständig getestet, es ist jedoch anzunehmen, dass die erzielbare Funktionalität und Leistung dieser Variante nicht mit nativer Implementation vergleichbar ist.

*JMF* verfügt auch über eine eigenständige Applikation *JMStudio*<sup>1</sup>, welche es ermöglicht Inhalte anzuzeigen, aufzunehmen und über Netzwerke zu verbreiten. Ebenso werden mit *JMF Registry* und *JMF Customizer* zwei Werkzeuge zur Registrierung von Multimediageräten und Anpassung von Parametern bereitgestellt. Die Funktionalität und Qualität der Anbindung kann somit schon vor der Integration überprüft werden.

Lizenzrechtlich untersteht *JMF* der Sun Public License[Spl06] und ist kostenlos erhältlich.

### 3.2.2 QuickTime for Java

Als Alternative zu *JMF* bietet sich *QuickTime for Java* an. Die von Apple vorangetriebene Technologie bietet die Integration der nativen QuickTime Implementation über das QuickTime API. Neben Java werden auch die meisten aktuellen höhere Programmiersprachen unterstützt, wobei die Einbindung über C und C++ am weitesten verbreitet ist.

Zur Verwendung unter Java benötigt man das *QuickTime for Java SDK* und eine installierte Instanz von QuickTime. Es ist insbesondere zu erwähnen, dass die frei erhältliche Version für Windows keine eigenständige Implementation zur Anbindung einer Kamera enthält. Dies ist nur mit dem kostenpflichtigen *QuickTime Pro* möglich.

QuickTime ist eine komplette Multimedia-Architektur mit einer Vielzahl von Funktionen die weit mehr abdecken als für dieses Projekt vorausgesetzt<sup>2</sup>. *QuickTime for Java* steht für Windows und Apple zur Verfügung und ist gemäss Herstellerangaben kompatibel ab Java 1.4.1.<sup>3</sup>

### 3.2.3 Twain

In der TWAIN Working Group<sup>4</sup> haben sich namhafte internationale Unternehmen zusammengeschlossen um eine Schnittstelle zu Scannern und Digitalkameras anzubieten. Eine Bildaneignung über die Twain ist unter Windows oder Apple grundsätzlich schon nach der Installation der Kamera mit dem Betriebssystem eigenen Treiber und einem Bildverarbeitungsprogramm möglich. Um Twain mit Java zu benutzen, ist jedoch eine Java Schnittstelle nötig. Dafür sind sowohl freie wie auch kostenpflichtige Schnittstellenlösungen verfügbar. Eine ausführliche Betrachtung der Bildaneignung (nicht nur) mit Twain bietet [Dav05]. Die vom Autor gezeigten Implementationen sind leicht verständlich, die Tatsache dass eine Bildaneignung über Twain aber länger als eine Sekunde dauert, verletzt die zuvor gestellte Bedingung der Effizienz derart, dass an dieser Stelle nicht weiter auf diese Technologie eingegangen wird.

---

<sup>1</sup>Entspricht einer Referenzimplementation von JMF

<sup>2</sup>vgl. <http://developer.apple.com/documentation/QuickTime/RM/Fundamentals/QTOve>, zuletzt besucht am 24. April 2006

<sup>3</sup>vgl. <http://developer.apple.com/quicktime/qtjava/> zuletzt besucht am 20. April 2006

<sup>4</sup><http://www.twain.org>

### 3.3 Abwägung

Zur Beurteilung wird die Erfüllung der gestellten Kriterien durch die drei zur Verfügung stehenden Technologien jeweils einander gegenübergestellt.

<i>Kriterium</i>	<i>JMF</i>	<i>QTJava</i>	<i>Twain</i>
<i>Lizenz Implementation</i>	frei verfügbar	teilweise frei verfügbar <sup>5</sup>	nicht verfügbar
<i>Lizenz Schnittstelle</i>	frei verfügbar	frei verfügbar	frei verfügbar
<i>Funktionalität</i>	ausreichend	hoch	tief
<i>Stabilität</i>	stabil	nicht getestet	nicht getestet
<i>Kompatibilität</i>	Windows, Linux, Solaris	Windows, Apple	Windows, Apple
<i>Komplexität</i>	mittel	hoch	tief
<i>Dokumentation</i>	hoch	hoch	hoch
<i>Resultat</i>	+	+	-

Tabelle 1: Abwägung der Produkteigenschaften

Die in Abbildung 3.3 dargelegten Produkteigenschaften sprechen klar für die Verwendung von entweder *JMF* oder *QuickTime*. Die Entscheidung zwischen diesen beiden Technologien ist hingegen schwieriger. Da Windows von beiden voll unterstützt wird, lässt sich dadurch keine Präferenz ableiten. Sollte eine Kompatibilität mit Apple zwingend gefordert sein, würde die Wahl sicherlich auf *QuickTime* fallen. Die Tatsache dass *Sun Microsystems* als Hersteller der Programmiersprache und der Entwicklungsumgebung auch ein passendes Multimedia Framework bereitstellt, lässt den Entwickler aber dazu tendieren, alle Komponenten aus einer Hand zu beziehen und *JMF* zu verwenden. Zudem ist zu erwähnen, dass die Konzepte von *JMF* leicht verständlich und gut dokumentiert sind, wogegen Dokumentation und Implementationsbeispiele zu *QuickTime* spärlich und teilweise nur kostenpflichtig erhältlich sind. Die Applikation ist jedenfalls so konzipiert, dass die Bildakquisitionskomponente zu einem späteren Zeitpunkt problemlos ausgetauscht werden kann.

### 3.4 Implementation

Die Vielfalt an Beispielen zur Erstellung einer *JMF* Applikation ist gross. Als Beispiele sei [Dav05] angefügt, wobei die konkrete Implementierung jeweils abhängig ist von der Version von *JMF*. Übereinstimmend lässt sich aber eine grundsätzliche Architektur zum Aufbau der Videoakquisition ableiten. Es wird folgender Ablauf vorgeschlagen:

1. Erkennung der an das System angeschlossenen Multimediageräte. Diese werden von System in einer Geräteliste verwaltet.
2. Für das aus der Liste ausgewählte Bilderfassungsgerät wird ein *medialocator* erstellt. Er beinhaltet die Adresse des Gerätes auf dem System sowie Informationen über unterstützte Medienformate.
3. Erstellung eines Players mit dem *medialocator* als Quelle
4. Starten des Players
5. Zuweisung des gewünschten Ausgabeformates an den Player.
6. Hinzufügen der visuellen Komponente zur grafischen Benutzerschnittstelle

Schliesslich benötigt man eine Methode zur Aneignung des aktuellen Einzelbildes

1. *FrameGrabbingControl* des Players holen

<sup>5</sup>Kostenpflichtige Version von *QuickTime Pro* nötig zur Bild- und Videoakquisition

2. Bild aneignen und Bilddaten in einem Puffer zwischenspeichern
3. Umwandlung des Puffers in ein Bild
4. Rückgabe des Bildes

### 3.5 Einschränkungen

Die Akquisition der Bilder (Rahmen aus dem Video-Stream) erfolgt über JMF-Funktionen. Es kann grundsätzlich zu jedem beliebigen Zeitpunkt auf den aktuellen Rahmen zugegriffen werden. In der Praxis hat sich jedoch gezeigt, dass sporadisch Verzögerungen auftreten können, welche es verunmöglichen eine kontinuierliche Akquisition der Rahmen in festen Abständen vorzunehmen. Diese Problematik wird ausführlich von [Van02] beschrieben. Beim dort betrachteten System handelt es sich zwar um Linux, jedoch wird auf Grund der Betriebssystemunabhängigkeit von Java und Firewire (IEEE1394) angenommen, dass die Verzögerungen auf die Aktivität der Java Garbage Collection in Verbindung mit eingeschränkten System-Ressourcen zurückzuführen sind.

Bei der für dieses Projekt verwendeten Hard- und Software ergibt sich für die Zeitspanne zwischen zwei Rahmen ein Idealwert 16 Millisekunden. Die sporadische Verzögerung beträgt ein Vielfaches und tritt gelegentlich auf.

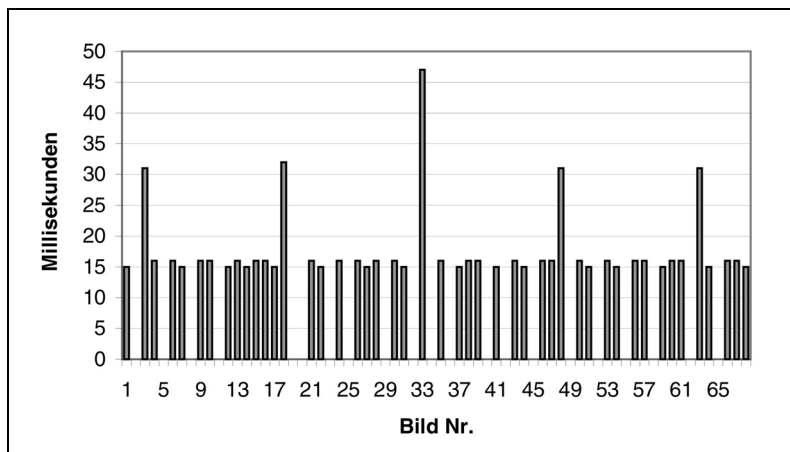


Abbildung 1: Dauer der Bildakquisition mit JMF

Bei der Abwägung der Methoden zur gleichmässigen Abtastung des sich drehenden Objektes im Kapitel 5 ist dieser Umstand deshalb unbedingt zu berücksichtigen. Falls ein Bild auf die Festplatte geschrieben wird, ist eine Multithreading-Strategie zu verwenden, um sofort wieder Ressourcen für die Akquisition des nächsten Rahmens bereitzustellen.

Abschliessend kann festgehalten werden, dass die Bildakquisition mit *JMF* stabil und effizient ist. Die Installation und Integration in eine Java-Applikation ist problemlos, auch wenn zur Leistungssteigerung systemspezifische native Treiber verwendet werden. Die Unterstützung von generischen Multimediageräten erlaubt die Verwendung von unterschiedlichster Hardware. Trotz der erwähnten Einschränkung kann *JMF* für die Lösung der in diesem Projekt an die Bildakquisition gestellten Aufgaben als sehr geeignet bezeichnet werden.

## 3.6 Bildverarbeitung

### 3.6.1 Anforderungen

Die kontinuierliche Aufnahme von Bildern erfordert auch von der Bildverarbeitung ein hohes Mass an Effizienz. Es sind folgende Voraussetzungen zu erfüllen:

- Interne Repräsentation und Anzeige der Bilddaten
- Traversierung der RGB-Matrix und wahlfreier Zugriff auf Pixel
- Erstellung von Bildern aus einer RGB-Matrix
- Verfügbarkeit und Flexibilität von Bildmanipulatoren
- Persistente, unkomprimierte Speicherung von Bildern

Abgesehen von der Verfügbarkeit von Bildmanipulatoren wäre Java2D als Standardbibliothek ausreichend. Zur Ermöglichung von verschiedenen Ansätzen der Segmentation, wobei unter Umständen der vielseitige Einsatz von Bildmanipulatoren nötig ist, wird jedoch die weit mächtigere Komponente *Java Advanced Imaging* bevorzugt.

### 3.6.2 Java Advanced Imaging

JAI stellt mit leicht verständlichen Konzepten objektorientierte Schnittstellen zur Bildbearbeitung bereit. Die Technologie wurde bereits in einer Vielzahl von kommerziellen und wissenschaftlichen Projekten erfolgreich eingesetzt<sup>6</sup>.

#### Bilddarstellung

Zur Darstellung von Bildern in der grafischen Benutzerschnittstelle wurde die Klasse `ImageDisplay` von *Sun Microsystems* eingebunden. Sie ermöglicht die effiziente Darstellung von Objekten des Typs `PlanarImage`, welche in JAI standardmässig zur Repräsentation von Bildern verwendet wird. Konzepte wie Doppelpufferung und Ausnahmebehandlung sind darin bereits vorhanden. Diese Klasse untersteht der *Sun Public License*[Spl06] und wird ohne Veränderungen übernommen.

#### Bildspeicherung

Auf Grund der grossen Datenmenge ist eine Zwischenspeicherung auf der Festplatte unumgänglich. JAI stellt eine umfangreiche Sammlung von Formaten und Codecs zur Bildspeicherung zur Verfügung. Für dieses Projekt wurde lediglich die Speicherung als unkomprimiertes TIFF verwendet. Dazu wird die auf einem Vorschlag des Herstellers beruhende Klasse *ImageCreator* verwendet.

---

<sup>6</sup><http://java.sun.com/products/java-media/jai>, zuletzt besucht am 10.01.2006

# 4 Bilderkennung

## 4.1 Anforderungen

In jedem Einzelbild soll eine Segmentation des Laserstreifens vorgenommen, und falls vorhanden für jede Zeile des Bildes der Spaltenindex des Laserstreifens ermittelt werden. Die Korrektheit des Verfahrens muss sowohl bei Beleuchtung mit Tageslicht als auch bei Kunstlicht oder in völliger Dunkelheit sichergestellt werden.

## 4.2 Ansätze

Um den Inhalt der Bildmatrix zu verarbeiten, wird eine zweidimensionale Ganzzahlliste erstellt, welche mit einem Iterator durchlaufen werden kann. Zuvor muss das Bild dahingehend verändert werden, dass nur noch die für die Segmentation relevante Information enthalten ist, also während der Traversierung der Bildmatrix alle in Frage kommenden Bildpunkte eindeutig erkannt werden können.

### 4.2.1 Kantenerkennung

Übergänge in markanten Bereichen eines Bildes können mit einem Kantenerkennungsfilter bestimmt werden.[Wi106] Dabei wird aus dem Originalbild mittels Faltung ein Gradientenbild erzeugt. Die grössten Intensitäten liegen nach der Operation dort, wo sich im Original die Helligkeit am stärksten ändert. Für die meisten Objektformen wird der Laserstreifen in vorwiegend horizontaler Richtung verlaufen. Deshalb ist ein horizontaler Sobel-Operator geeignet.

1	0	-1
2	0	-2
1	0	-1

Tabelle 2: Kantenfilter-Operator nach Sobel

Wird auf dem entstehenden Gradientenbild zusätzlich eine Schwellwert Funktion (siehe 4.2.2) angewandt, lassen sich die zwei den Laserstreifen begrenzenden Kanten leicht erkennen. Unter normaler Beleuchtung ergibt sich jedoch das Problem, dass auch alle anderen Kanten im Bild markiert werden. In einem komplett abgedunkeltem Raum besteht diese Problematik zwar nicht, ohne eine genaue Abstimmung der Beleuchtung und der Sensorempfindlichkeit wird aber eine Segmentierung sehr störungsanfällig. Die Kantenerkennung ist daher für eine umfassende Lösung des Segmentationsproblems nur beschränkt tauglich.

### 4.2.2 Schwellwerte

Ein weiterer Ansatz beruht auf der Annahme, dass der Laserstreifen im Allgemeinen das hellste Element des Bildes darstellt. In dunklen Umgebungen oder bei Kunstlicht ist das zutreffend. Somit kann dort mit einem Schwellwertverfahren[Wi206] der Laserstreifen aus dem Bild gefiltert werden. Allen Bildpunkten die über oder unter dem Schwellwert liegen, wird ein Maximal- respektive Minimalwert zugewiesen. Durch diese Binarisierung geht jedoch jegliche Information über die Intensitätsverteilung innerhalb des Laserstreifens verloren. Die Problematik bei dieser Methode besteht zusätzlich darin, dass weisses Umgebungslicht stark störend wirkt. Bildpunkte mit Beleuchtung durch Umgebungslicht haben in allen Bändern hohe Werte. Wird der Schwellwert nur auf das Rot-Band angewandt,

entstehen zwar bessere Resultate, den Laserstreifen von anderen hellen Partien zu unterscheiden, ist aber mit der Schwellwertmethode nicht unter allen Lichtbedingungen möglich. Die Bestimmung des Schwellwertes ist zudem für die jeweiligen veränderten Lichtverhältnisse neu vorzunehmen.

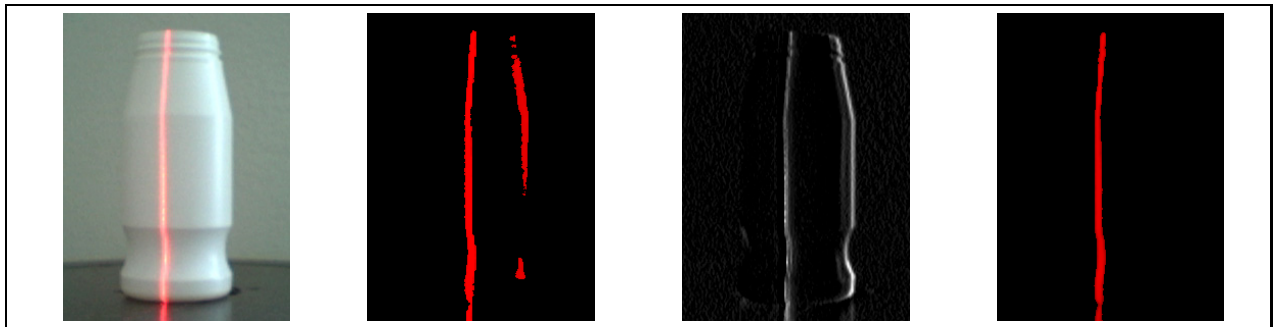


Abbildung 2: Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Tageslicht

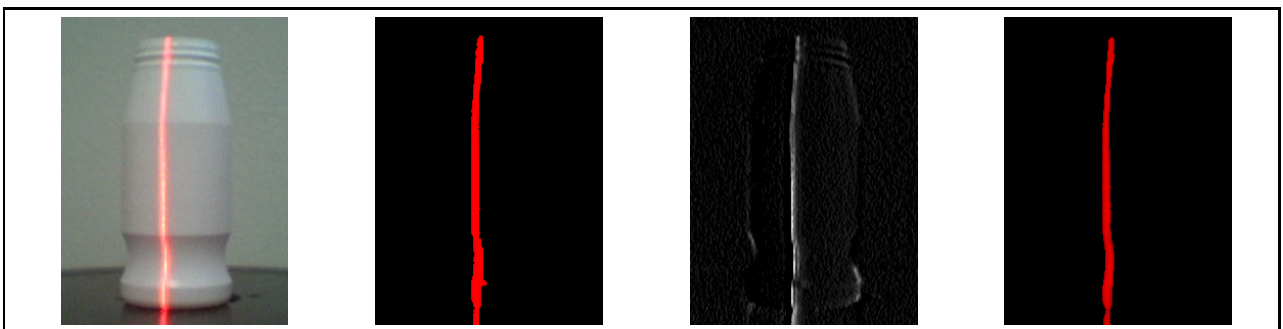


Abbildung 3: Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Kunstlicht

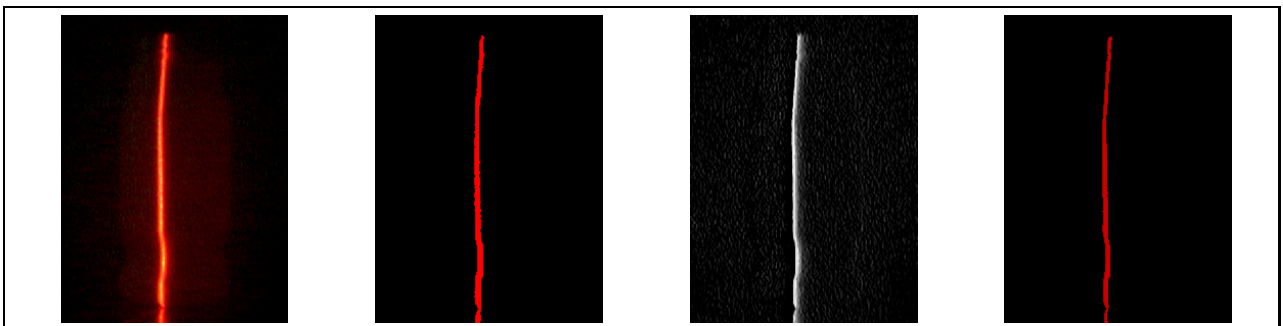


Abbildung 4: Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Dunkelheit

### 4.2.3 Optische Filter

Anstatt die Segmentation softwareseitig vorzunehmen, ist es auch möglich einen optischen Filter vorzuschalten. Dieser statische, auf die Wellenlänge des Lasers zugeschnittene Filter würde nur das reflektierte Laserlicht zum Sensor durchlassen. Ein solches Verfahren würde die Segmentation stark vereinfachen, jedoch auch die Flexibilität einschränken und zusätzliche Kosten verursachen. Wie schon in der Einleitung dargelegt, ist es aber das Ziel, durch die entwickelte Software die Kosten für die Komponenten tief zu halten. Deshalb wird auf den Einsatz eines optischen Filters verzichtet und die Segmentation wie unter 4.3.1 aufgezeigt, rein algorithmisch vorgenommen. Nachfolgende Projekte erhalten so zudem die Möglichkeit, aus den Einzelbildern weitere Information zur Beschaffenheit des Objektes zu gewinnen und in der Oberflächenrekonstruktion einzusetzen.

## 4.3 Lösung

Nach der Darlegung von verschiedenen Ansätzen soll nun die detaillierte Beschreibung des zum Ziel führenden Segmentationsprozesses erfolgen.

### 4.3.1 Filterung

Die Lösung zum Segmentierungsproblem liegt in der Umwandlung des RGB- in einen HSL-Farbraum, denn die Farbe des Laserstreifens kann unter der Voraussetzung eines korrekten Weissabgleichs als bekannt angenommen werden. Sie liegt auf einer von  $0^\circ$  bis  $360^\circ$  reichenden Farbskala entweder zwischen  $0^\circ$  und  $60^\circ$  oder zwischen  $240^\circ$  und  $360^\circ$ . Im Allgemeinen ist es ausreichend, nur Bereiche im mittleren und oberen Sättigungs- und Helligkeitsspektrum zu betrachten. Mit der Festlegung eines unteren Grenzwertes für die Sättigung gelingt es jedoch, gegen weiss tendierende Bereiche zu eliminieren. Bei Objekten mit ähnlicher Farbkomponente wie der Laserstreifen und bei starkem Einfluss von Umgebungslicht muss unter Umständen auch der Grenzwert für die Helligkeit variiert werden um störende Bereiche auszublenden. Als Grundeinstellung werden folgende Parameter vorgeschlagen, sie liefern bei normaler Raumbelichtung gute Resultate:

Komponente	Bereich
Farbe [°]	0-60 und 240-360
Sättigung [%]	50-100
Helligkeit[%]	10-100

Tabelle 3: Filterungskriterien im HSL Modell

Ist  $R$  der Rotwert des resultierenden einbandigen Bildes,  $H$  der Farbwert,  $S$  die Sättigung und  $L$  die Helligkeit eines Bildpunktes im nach HSL gewandelten Bild, dann gilt für Punkte welche die Kriterien

$$H_{min} < H < H_{max} \text{ und } S_{min} < S < S_{max} \text{ und } L_{min} < L < L_{max} \quad (1)$$

erfüllen

$$R = 127 \times S + 127 \times L. \quad (2)$$

Für alle anderen Punkte gilt

$$R = 0. \quad (3)$$

Der Wert  $R$  der Bildpunkte ergibt sich je zur Hälfte aus der mit dem halben Maximalwert multiplizierten Sättigungs- und Helligkeitskomponente. Dadurch wird erreicht, dass die für den Laserstreifen charakteristische Intensitätsverteilung erhalten bleibt. Das ist für die nachfolgende Analyse dieses Zwischenresultates unbedingt erforderlich.

### 4.3.2 Erkennung

Das in 4.3.1 generierte Bild wird nun zeilenweise durchlaufen. Zuerst ist es notwendig, den Maximalwert für jede Zeile zu bestimmen.

```
int[Image.height] rowmax;
int tempmax;
for(i=0;i<Image.height;i++){
  for(j=0;j<Image.width;j++){
    if(ImageData[i][j]>tempmax){ //Rotes Band des Bildpunktes hat höhere
      tempmax=ImageData[i][j]; //Wertigkeit als bisheriges Zeilenmaximum
    }
  }
  rowmax[i]=tempmax;
  tempmax=0;
}
```



Mit der Kenntnis der Zeilenmaxima wird das Bild nun ein zweites Mal durchlaufen. Dabei geht es darum, die Anzahl der Bildpunkte mit Maximalwert zu erkennen. Gibt es mehr als ein Maximum, wird die Lage des Laserstreifens durch deren Durchschnitt bestimmt. Wenn es nur ein Maximum gibt, werden die links und rechts davon liegenden Nachbarn berücksichtigt und unter diesen drei Bildpunkten die Sub-Pixel Genauigkeit berechnet<sup>1</sup>.

```
int maxcount
int[Image.width] maxvalues
int leftneighbour, pixel, rightneighbour
float[Image.height] result
for(i=0;i<Image.height;i++){
maxcount=0
  for(j=0;j<Image.width;j++){
    if(ImageData[i][j]=rowmax[i]&&rowmax[i]>0){
      maxvalues[j]=ImageData[i][j] //Speichern des Wertes auf Spaltenindex
      maxcount++
    }
  }
  if{hasOnlyOneValue(maxvalues)}{
    leftneighbour=ImageData[i][getIndexOfMaximum(maxvalues)-1]
    maxpixel=ImageData[i][getIndexOfMaximum(maxvalues)]
    rightneighbour=ImageData[i][getIndexOfMaximum(maxvalues)+1]
    result[i]=SubPixelOffset(leftneighbour, pixel, rightneighbour)
                                     //Sub-Pixel Offset
  }else{
    result[i]=getAverageColumnValueOfMaximas(maxvalues)
                                     //Mittelwert der Maximal-Pixel Indizes
  }
  removeAll(maxvalues)
}
```

Die resultierende Liste `result` beschreibt nun den segmentierten Laserstreifen. Daraus wird anschliessend die Distanz zur Referenzebene bestimmt. Diese Berechnung findet mit den Formeln aus Abschnitt 7.1 statt. Mit der Umrechnung zu kartesischen Koordinaten wie in Abschnitt 7.2 beschrieben ist die Auswertung abgeschlossen.

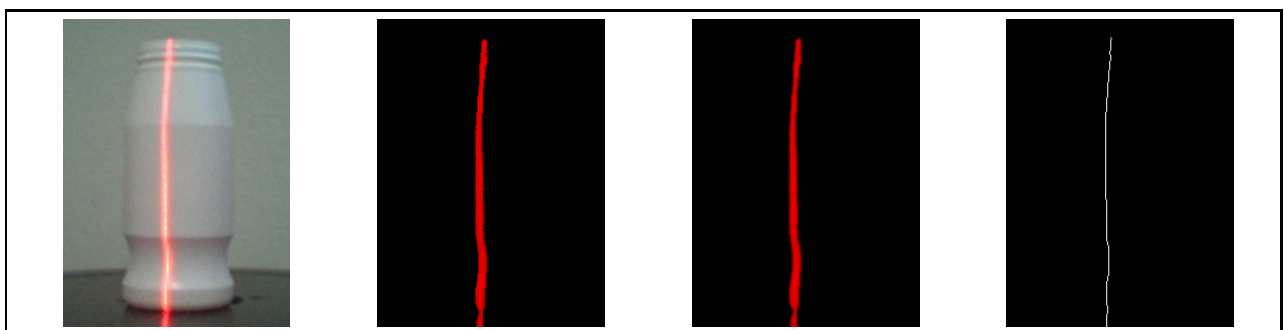


Abbildung 5: Originalbild, HSL-Filterung, Medianfilter und Resultat der Segmentation (bei Kunstlicht)

### 4.3.3 Optimierende Bildoperationen

Um das Rauschen des Bildes zu minimieren, sollte vor der Analyse ein Weichzeichnungsfilter und nach Filterung auf dem HSL Farbmodell ein Medianfilter angewendet werden.

<sup>1</sup>(siehe Abschnitt 4.3.4)

## Weichzeichungsfilter

Der Weichzeichungsoperator besteht aus einer 5x5-Matrix mit diskreter Annäherung an die Gauss'sche Wahrscheinlichkeitsverteilung. [Fis94] Es werden alle Bänder des Originalbildes verarbeitet.

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Tabelle 4: Weichzeichungsoperator mit Gauss-Verteilung

## Medianfilter

Nach der Berechnung der Intensitäten wird auf dem roten Band ein Medianfilter eingesetzt. Damit soll die Erkennung von kontinuierlichen Bereichen innerhalb des Laserstreifens verbessert werden. Die Grösse der Matrix beträgt 3x3. Eine Herleitung findet sich bei [Fis94].

### 4.3.4 Verfeinerung (Sub-Pixel Genauigkeit)

Da der digitale Bildsensor die Szene nur in diskreten Schritten auflösen kann, ist es sinnvoll über eine Analyse der Intensitätsverteilung des Laserstreifens zu einer Methode mit genauerer Abschätzung der Lage des Intensitätsmaximums zu gelangen. In [For04] wird aufgezeigt, dass durch elektrisches- und quantisationsbedingtes Rauschen oder die für Laserlicht typische Sprenkelung - abhängig vom beleuchteten Material - Streulicht entsteht. Die Verteilung dieses Streulichtes kann mit verschiedenen Methoden abgeschätzt werden. Deren Standardabweichung ist stark abhängig vom Rauschabstand, als universell einsetzbar hat sich dabei aber die Annahme der Wahrscheinlichkeitsverteilung nach Gauss erwiesen. Das nachfolgende Diagramm zeigt die Intensitätsverteilung nach der HSL-Filterung für einen Ausschnitt aus dem Laserstreifen.

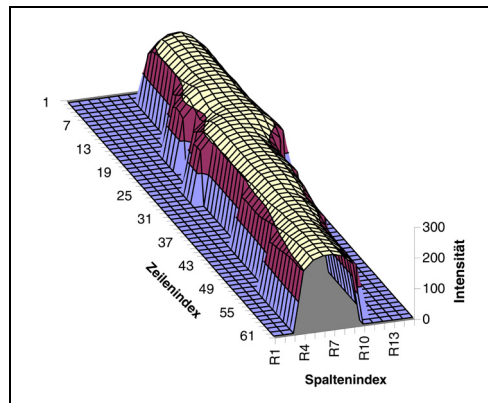


Abbildung 6: Intensitätsverteilung nach der Filterung auf dem HSL-Farbmodell

Zur Berechnung der Sub-Pixel Genauigkeit ergibt sich dementsprechend

$$\delta = \frac{1}{2} \times \frac{\ln(a) - \ln(c)}{\ln(a) + \ln(c) - 2 \times \ln(b)} \quad (4)$$

wobei  $a$  und  $c$  die benachbarten Bildpunkte von  $b$  sind.

## 4.4 Bewertung

Das vorgestellte Konzept der Umwandlung des Bildes von RGB nach HSL und der Filterung mittels empirisch ermittelten Parametern liefert unter allen betrachteten Beleuchtungsverhältnissen akzeptable Resultate. In der Implementation wird es dem Benutzer zudem ermöglicht, die Filterparameter den Umgebungsbedingungen anzupassen. Diese Flexibilität ist auch im Hinblick auf die verwendete Kamera und deren Einstellung notwendig. Auf dieser Grundlage kann die Ermittlung der Spaltenindizes effizient durchgeführt und unter der Annahme einer typischen Intensitätsverteilung mit Sub-Pixel Genauigkeit präzisiert werden.

# 5 Positionsermittlung

Während sich das Objekt auf dem Drehteller dreht, sollen von der Kamera in konstanten Abständen Bilder gemacht werden. Zur späteren Auswertung muss von jedem Bild bekannt sein, bei welchem Drehwinkel es aufgenommen wurde. Deshalb muss eine Konzept erarbeitet werden, das ermöglicht, entweder zu beliebigen Zeitpunkten Bilder aufzunehmen und nachträglich den Drehwinkel zu bestimmen, oder die Bilder in vordefinierten, konstanten Abständen auszulösen. Die in 3.5 beschriebene Verzögerung der Bildakquisition stellt dabei eine zu berücksichtigende Problematik dar.

## 5.1 Ansätze

### 5.1.1 Direkte Kontrolle

Grundsätzlich ist es denkbar, einen Drehteller zu verwenden, der von der Applikation kontrolliert wird. Dann kann der Teller jeweils um einen bestimmten Winkel gedreht und dann ein Bild gemacht werden. Die Applikation führt den nächsten Schritt erst aus, wenn der vorherige komplett abgeschlossen, also die Bildakquisition durch die Kamera abgeschlossen und das Bild gespeichert ist. Diese Methode setzt jedoch voraus, dass die Applikation über eine Schnittstelle mit dem Drehteller kommuniziert. Damit hängt das gesamte Konzept davon ab, dass ein potentiell teurer, steuerbarer Drehteller zum Einsatz kommt. Zudem erhöht die Schnittstellenkommunikation die Komplexität der Applikation. Im Hinblick auf eine möglichst kostengünstige und unabhängige Implementation ist dieser Ansatz ungünstig. Die einzige Anforderung an den Drehteller soll nämlich darin bestehen, dass er sich mit konstanter Geschwindigkeit dreht.

### 5.1.2 Zeitmessung

Durch die Kenntnis der Drehgeschwindigkeit wäre es leicht möglich in einem dem Drehwinkel-Inkrement entsprechenden Intervall Bilder zu machen. Mit einer Schnittstelle zum Drehteller und Zugriff auf dessen Parameter oder dem später in Abschnitt 5.2.5 erläuterten Vorgehen zur Drehgeschwindigkeitsmessung wäre dies möglich. Die Steuerung erscheint dabei aber nur auf den ersten Blick trivial, denn der Applikation selbst bleibt es verborgen, zu welchem Zeitpunkt das Bild aufgenommen wurde. Es sind lediglich die Zeitpunkte vor der Anweisung und nach erfolgreicher Bildakquisition bekannt. Wie unter 3.5 dargelegt, ist die Zeitspanne zwischen Aufruf und Ausführung der Bildakquisition aber als volatil zu bezeichnen.

### 5.1.3 Markierungen

Um unabhängig von Drehgeschwindigkeit und Kommunikationsmöglichkeiten zu sein, soll anhand von Markierungen die Position bestimmt werden. An der Stirnseite der Platte des Drehtellers wird eine aus Papier gefertigte Markierung angebracht. Sie deckt den vollen Umfang ab und ist in vertikale, regelmässige Streifen unterteilt. Die Marken (Streifen) sind abwechslungsweise schwarz und weiss. Da die Laserebene senkrecht zur Plattenebene projiziert wird, ist auf der Stirnseite der Platte ein Laserstreifen sichtbar. Durch die absorbierende Eigenschaft der schwarzen Farbe, ist dies aber nur auf einer weissen Marke möglich. Die Kombination der unter 4.3 aufgezeigten Segmentation mit einem Zustandsautomaten ermöglicht so die grobe Positionsermittlung des Drehtellers. Im Detail treten dabei aber auch einige zusätzliche Schwierigkeiten auf, welche mit zugehörigen Lösungen im nächsten Abschnitt beschrieben werden.

## 5.2 Lösung

Die Akquisition der aktuellen Rahmen geschieht kontinuierlich mit dem schnellst möglichen Intervall<sup>1</sup>. In Abschnitt 4.3 wurde beschrieben wie sich der Laserstreifen aus dem Bild filtern lässt. Im Unterschied zur Segmentation ist nun jedoch nicht die Lage des Laserstreifens gesucht. Es muss lediglich geklärt werden, ob der Laserstreifen überhaupt sichtbar ist, also auf einer weissen Marke liegt. Dazu wird die Filterung mittels HSL-Modell wie bei der Segmentation vorgenommen. Die Lage des Laserstreifens auf der Stirnseite des Drehtellers wird dazu vorab vom Benutzer in dem von der Applikation bereitgestellten Testbild<sup>2</sup> mit einem Rechteck markiert<sup>3</sup>. Die Abmessungen dieser Auswahl sollen ungefähr der Breite des Laserstreifens und der halben Höhe des Drehtellers im Bild entsprechen<sup>4</sup>. In jedem von der Kamera aquirierten Rahmen wird nun dieser Bereich darauf hin untersucht, wie viele der Bildpunkte der Bedingung der Filterfunktion genügen. Dabei kann eine Toleranz erlaubt werden, die Verunreinigungen, Unschärfe und Bildrauschen Rechnung trägt. Um die Anzahl Bildpunkte zu ermitteln, wird vor jedem Scanvorgang während einer Kalibrierungsphase die maximal mögliche Anzahl Bildpunkte bestimmt und ein darauf basierender Toleranzwert festgelegt.

### Kalibrierungsfunktion

```
int pixels //Anzal der Bildpunkte welche
           //Kriterim erfüllen
int xstart, xend, ystart, yend //Lage des zu analysierenden
                               //Bildausschnittes
function calibrateNumberOfPixels(image){
    long starttime=now
    int [] [] [] imageData=getRGBImageMatrix(image)
    double [] [] [] hslData=getHSLData(imageData, xstart, xend, ystart, yend)
                               //Nur den zu analysierenden Bereich
                               //nach HSL konvertieren
    while(now<starttime+5000){ //Kalibration dauert 5 Sekunden
        for(int i=0;i<yend-ystart;i++){ //Bildausschnitt in HSL traversieren
            for(int j=0;j<xend-xstart;j++){
                if(fulfillsHSLCriteria(hslData[i][j])){
                    pixels++ //Bildpunkt erfüllt Filterkriterien
                }
            }
        }
        pixels=round(pixels*0.9) //Abweichung von 10 Prozent zulassen
    }
    return pixels
}
```

### Analysefunktion

```
function analyseNumberOfPixels(image){
    int foundpixels
    int [] [] [] imageData=getRGBImageMatrix(image)
    double [] [] [] hslData=getHSLData(imageData, xstart, xend, ystart, yend)
    for(int i=0;i<yend-ystart;i++){
        for(int j=0;j<xend-xstart;j++){
```

---

<sup>1</sup>siehe Abschnitt 3.5

<sup>2</sup>siehe Abbildung 19

<sup>3</sup>Nur bei Änderung der Aufstellungsgeometrie notwendig

<sup>4</sup>Beispielsweise 3×10 Bildpunkte

```

        if(fulfillsHSLCriteria(hslData[i][j]){
            foundpixels++
        }
    }
}
if(!foundpixels<pixels){
    return true
}else{
    return false
}
}

```

Während des Scanvorganges wird nun, bis die (von der Einteilung der Markierungen abhängige) benötigte Anzahl Bilder erreicht ist, folgende Funktion ausgeführt.

### Scanfunktion

```

int imagecount=0
int angle=0
int totalimages=180 //Für 1 Grad Markierungen
Image image
function scan{
    while(imagecount<totalimages){
        image=getImage()
        if(statusChanged(analyseNumberOfPixels(image)){
            storeImage(image, angle)
            imagecount++
            angle=angle+360/totalimages
        }
    }
}

```

Der von `analyseNumberOfPixels` zurückgegebene bool'sche Zustand ob der aktuelle Bereich dem Kriterium für eine weisse Marke entspricht, wird also vom Zustandsautomaten verarbeitet. Wenn dieser feststellt, dass soeben ein Übergang von einer schwarzen zu einer weissen Marke erfolgte, wird das aktuelle Bild gespeichert.

### Zustandsautomat

```

boolean state=false
function boolean statusChanged{
    if(newstate=state){ //Kein Übergang
        return false
    }else{
        if(newstate=true && state=false){ //Übergang schwarz -> weiss
            state=newstate
            return true
        }else{ //Übergang weiss -> schwarz
            state=false
            return false
        }
    }
}
}

```

### 5.2.1 Genauigkeit

Der vorgeschlagene Mechanismus erlaubt es also, bei jeder weissen Marke ein Bild zu speichern. Als notwendige Zusatzinformation wird dem Dateinamen<sup>5</sup> die sich aus dem Bildzähler ergebende Gradzahl angehängt. Die Position des Drehtellers ist damit aber nicht genauer als die Einheit der Markierung feststellbar. Zur Präzisierung der Lage innerhalb der weissen Marke muss somit noch ein weiteres Merkmal analysiert werden. Dies geschieht mittels Kantendetektion. Dazu markiert der Benutzer, wie schon für die Lage des Laserstreifens, einen zusätzlichen Bereich auf der Stirnseite des Drehtellers. Dieser wird dazu in eine Position gebracht in welcher der Laserstreifen genau im Zentrum einer weissen Markierung liegt. Nun wird (an einer beliebigen Stelle) eine Kante zwischen zwei Marken ausgewählt. Der Bereich durch dessen Mitte die Kante verlaufen soll, muss dabei genau der Markenbreite entsprechen. Bei der Selektion wird der Benutzer von der Applikation unterstützt, welche die genaue Positionierung selbst vornimmt. Vor der Speicherung des Bildes wird jeweils eine Kantendetektion in diesem Bereich vorgenommen. Aus der Lage der Kante und aus der Breite des Bereiches ergibt sich dann die relative Lage des Laserstreifens innerhalb einer Markierung.

### 5.2.2 Kantendetektor

```
int xstart, xend, ystart, yend //Lage Bildausschnitt für Kantenerkennung
function int analyseEdge(image){
    int [] [] [] imageData=getRGBMatrix(image)
    double [] [] [] hslData=getHSLData(imageData,
        xstart, xend, round((yend-ystart)/2))
        //Es wird nur eine Zeile in der Mitte des
        Bereiches analysiert
    hslData=applyEdgeDetection(hslData) //Kantendetektionsfilter anwenden
    for(int i=0;i<xend-xstart;i++){
    if(hslData[i][round((yend-ystart)/2)][1]<0.2
    &&hslData[i][round((yend-ystart)/2)][2]>0.8){
        return i //Kante erkennen
    }
    }
}
```

Mit der Kenntnis der Lage der Kante lässt sich somit in der Scanfunktion der Drehwinkel präzise berechnen.

### 5.2.3 Winkelberechnung

```
double edgelocation=(analyseEdge(image)/(xend-xstart)) //Lage im Ausschnitt
angle=angle + 0.5 - edgelocation
```

Diese Berechnung liefert nur für die eine Drehrichtung die effektiven, für die andere aber die um 1° verschobene Positionen, was für das Endresultat aber unerheblich ist.

### 5.2.4 Zuverlässigkeit

Bei den bisherigen Betrachtungen zur Positionsbestimmung wurde immer davon ausgegangen, dass das System fähig ist, mindestens ein Bild von jeder Marke zu liefern. Wie unter 3.5 beschrieben, ist leider die Bildakquisition nicht mit konstanten Raten möglich. Bei Drehgeschwindigkeiten von mehr als 3° pro Sekunde führt dies dazu, dass beim Scanvorgang Marken ausgelassen werden. Dadurch wird das Endresultat nicht nur durch fehlende Bereiche und Überschneidungen, sondern auch durch falsche Berechnungen in der Auswertung unbrauchbar. Da dieses Verhalten nicht behoben werden kann, muss eine Lösung gefunden werden, die es wenigstens ermöglicht, das Auslassen einer Marke festzustellen.

---

<sup>5</sup>Zeitstempel

Das folgende Bild wird dann automatisch wieder mit dem dafür gültigen Drehwinkel gespeichert. Als einziges Artefakt bleibt eine Lücke im Modell. Je nach Objektform und Verwendungszweck wird das zwar störend wirken, aber trotzdem für alle errechneten Punkte korrekte Resultate liefern.

### 5.2.5 Bestimmung der Drehgeschwindigkeit

Um abzuschätzen, ob eine Markierung ausgelassen wurde, ist zuerst die Zeitspanne zwischen zwei aufeinander folgenden Bildern zu bestimmen. Dies geschieht während der Kalibrierungsphase. In einem Testlauf wird eine Anzahl von Bildern mit dem eben beschriebenen Scanmechanismus gemacht. Dabei werden die Intervalle von unterschiedlicher Länge sein, unter Umständen wird sogar hier schon eine Verzögerung auftreten und eine Marke ausgelassen. Der Mittelwert dieser Intervalle kann als ausreichend genau bezeichnet werden, jedoch müssen vor der Mittelwertbildung die eventuell vorhandenen langen Abschnitte einer Auslassung entfernt werden. Ist die durchschnittliche Zeitspanne zwischen zwei Marken schliesslich bekannt, wird das System während dem Scanvorgang ständig die verstrichene Zeit seit dem letzten Bild damit vergleichen. Ist der Abstand um ein Vielfaches grösser, kann die Anzahl der ausgelassenen Marken festgestellt werden.

```
int testimages=20
int i=0
long[] intervals
long lastcapturetime
Image image
function measureIntervals{
    while(i<testimages){
        image=getImage()
        if(statusChanged(analyseNumberOfPixels(image))){ //Wie beim Scanvorgang
            if(i>0){
                intervals[i]=now-lastcapturetime
                //Berechnung und Speicherung des Intervalls
            }
            i++
            lastcapturetime=now
        }
    }
}

avgtimebetweenmarks=getAverage(removePeaks(intervals))
```

Während dem eigentlichen Scanvorgang wird nun die selbe Zeitmessung vorgenommen. Maximal können zwischen zwei weissen Marken eineinhalb Intervalle liegen, wenn die letzte Marke am Anfang und diese am Ende „erwischt“ wurde. Ist dies nicht der Fall und das Intervall ist länger, wird der Drehwinkel dementsprechend angepasst und auch der Bildzähler wird um die Anzahl der ausgelassenen Bilder inkrementiert. Analog dazu kann auch eine Auslassung von mehreren Marken entdeckt werden, die maximal zulässige Dauer erhöht sich dann je Auslassung um ein Intervall.



Die Abbildung 7 zeigt die Intervalle zwischen zwei Bildern bei einer Rotationsgeschwindigkeit von  $10^\circ$  pro Sekunde. Deutlich sichtbar sind die Auslassungen von jeweils einer Marke. Der der Durchschnitt liegt bei 202 Millisekunden.

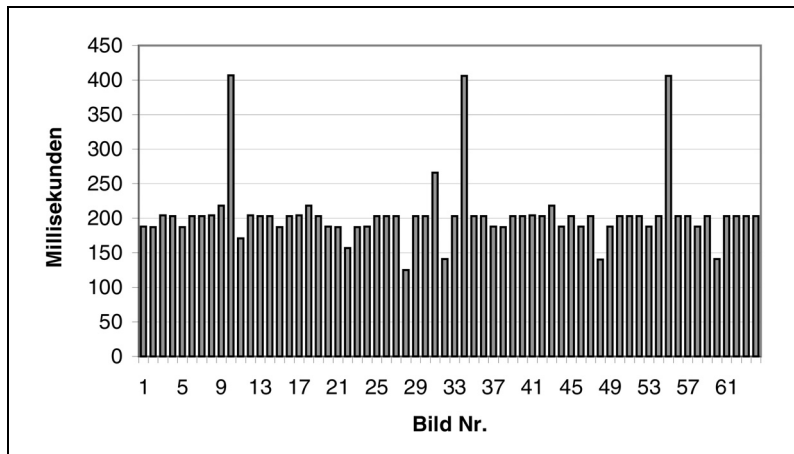


Abbildung 7: Zeitspanne zwischen zwei Bildern (mit drei Auslassungen)

# 6 Aufbau und Kalibrierung

In diesem Kapitel wird der Aufbau des Gerätes beschrieben. Eine detailliertere Abhandlung über Vor- und Nachteile der Aufstellungsgeometrie wird in Kapitel 8 vorgenommen.

## 6.1 Anforderungen

Das gewählte Verfahren der optischen Triangulation bedingt, dass Laser und Kamera (nachfolgend *Scanner*) rechtwinklig zur Laserebene fest miteinander verbunden sind. Das gewählte Positionsermittlungs-Verfahren aus Kapitel 5 erfordert seinerseits eine feste Kopplung des Scanners an den Drehteller. Dabei soll aber nachträglich jederzeit eine Abänderung (z.B. für Nachfolgeprojekte) möglich sein. Natürlich gilt es auch hier kostengünstige Lösungen zu finden.

## 6.2 Umsetzung

Wie in [KKS96] vorgeschlagen, wird der Scanner so am Drehteller befestigt, dass die Laserebene durch das Zentrum des Drehtellers verläuft. In Abbildung 9 ist diese Anordnung der Komponenten ersichtlich. Zwei Aluminium-Vierkantrohre sind an der Grundplatte des Drehtellers festgeschraubt und tragen Laser und Kamera, welche auf einer rechtwinklig dazu angebrachten Leiste montiert sind. Die Befestigung des Lasers geschieht mit der mitgelieferten Halterung und ist problemlos. Für die Kamera musste speziell eine Halterung konstruiert werden, da die vom Hersteller mitgelieferten Standfüsse zu instabil und nicht genügend flexibel in Bezug auf die Aufstellung waren.



Abbildung 8: Montage von Laser und Kamera mit Standfüssen

Durch die Konstruktion nach dem Baukasten-Prinzip ist es leicht möglich, die Aufstellungsgeometrie umzustellen. Bei guter Verschraubung ist dennoch eine ausreichende Fixierung möglich.

### 6.2.1 Ausrichtung der Komponenten

Bei dem implementierten Konzept handelt es sich nicht um ein kalibrierungsfreies Verfahren. Es ist notwendig, die Ausrichtung der einzelnen Komponenten exakt zu kennen, respektive die Komponenten nach genauen Vorgaben auszurichten.



Abbildung 9: Laserscanner, bestehend aus Drehteller, Laser (u. links), Kamera (u. rechts) und Steuerungseinheit mit Benutzerschnittstelle (rechts)

### Kamera

Die Kamera muss horizontal auf die Szene blicken. Die Ausrichtung ist manuell vorzunehmen. Seitlich ist die Kamera so auszurichten, dass das Zentrum des Drehtellers im Bild der Spalte  $M/2$  entspricht. Dazu wird in der Applikation im Modul „Adjust“<sup>1</sup>

# 7 Auswertung

Die Berechnungen zur Bestimmung der Distanzwerte und die dazugehörige Abbildung 10 unter 7.1 stammen, mit gekennzeichneten Ausnahmen, aus [KKS96]. Alle anderen Formeln in diesem Kapitel ab Abschnitt 7.2 sind vom mir selbst entwickelt.

## 7.1 Distanzwerte

Mit dem Verfahren der optischen Triangulation werden aus den mit der Kamera erfassten Bildern die Distanzwerte berechnet. Das Kamerabild liegt in einer Matrix von  $M \times N$  Bildpunkten vor.  $k = 0$  repräsentiert dabei die im Bild am weitesten links gelegene Spalte,  $k = M - 1$  den am weitesten rechts gelegene.

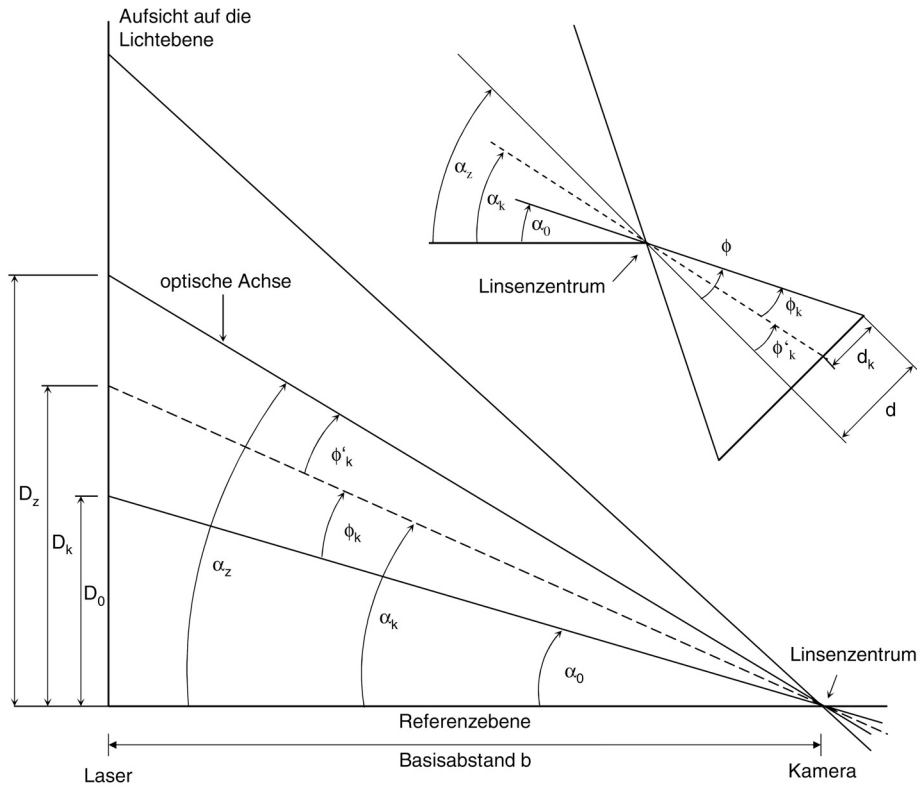


Abbildung 10: Berechnung der Distanzwerte mittels Spaltenindex  $d_k$

Als erster Schritt müssen die beiden Winkel  $\alpha_0$  und  $\alpha_z$  bestimmt werden.  $D_z$  entspricht dabei dem Abstand der Referenzebene zum Mittelpunkt des Drehtellers und  $D_0$  ist der Abstand von der Referenzebene bis zum in der Spalte  $k = 0$  liegenden Objekt. Mit  $b$  wird die als bekannt angenommene Triangulationsbasis, die Distanz zwischen Laser und Kamera, bezeichnet.

$$\alpha_z = \arctan\left(\frac{D_z}{b}\right) \quad (5)$$

$$\alpha_0 = \arctan\left(\frac{D_0}{b}\right) \quad (6)$$

Daraus lässt sich leicht der für die weiteren Berechnungen benötigte halbe Durchmesser  $d$  des Sensors berechnen

$$d = f \times \tan(\alpha_z - \alpha_0), \quad (7)$$

wobei  $f$  die Brennweite der Kamera ist. Bei einer Digitalkamera entspricht die Brennweite theoretisch dem Abstand der Objektiv-Hauptebene zur Sensorebene. Da bewegliche Linsensysteme diesen Abstand aber während des Betriebes verändern, ist die Bestimmung einer festen Brennweite nur schwer möglich. In den folgenden Ausführungen wird  $f$  mit der vom Hersteller angegebenen Länge angenommen. Da  $f$  in der resultierenden Formel nicht mehr enthalten sein wird, ist die Genauigkeit dieser Angabe unbedeutend.

Für ein  $M$ -spaltiges Bild beträgt das Abstandsinkrement  $d_k$  zwischen zwei Spalten

$$d_k = k \times \frac{d}{M/2} = \frac{2kd}{m}. \quad (8)$$

Um den Winkel  $\alpha_k$  zwischen der Projektion des Laserstreifens und der Referenzebene zu bestimmen, verwendet man

$$\alpha_k = \alpha_z - \phi'_k \quad \text{bzw.} \quad \alpha_k = \alpha_z + \phi''_k \quad (9)$$

mit

$$\tan(\phi'_k) = \frac{d - d_k}{f} \quad \text{bzw.} \quad \tan(\phi''_k) = \frac{d_k - d}{f}. \quad (10)$$

Somit ergibt sich aus 8 und 10

$$\tan(\phi'_k) = \frac{d(M - 2 \times k)}{M \times f} \quad \text{bzw.} \quad \tan(\phi''_k) = \frac{d(2 \times k - M)}{M \times f} \quad (11)$$

für Bildpunkte jeweils auf der einen  $0 \leq k \leq M/2$  oder der anderen Seite der optischen Achse  $M/2 < k \leq M - 1$ .

Zur Berechnung des Abstandes  $D_k$  wird der Winkel  $\alpha_k$  von 9 benötigt. Es gilt somit

$$D_k = b \times \tan(\alpha_k). \quad (12)$$

An dieser Stelle wird in [KKS96] ein Fehler gemacht. Es wird fälschlicher Weise  $\phi_k$  an Stelle von  $\alpha_k$  eingesetzt.

## 7.2 Koordinaten

Um das Resultat in ein räumliches Modell zu überführen, müssen für jeden Punkt aus dem errechneten Distanzwert  $D_k$ , dem Zeilenindex im Bild  $j$  und der Position des Drehtellers Winkel  $\beta$  die kartesischen Koordinaten errechnet werden. Als Ursprung wird dabei das Rotationszentrum festgelegt. Die Herleitung der Berechnung erfolgt über trigonometrische Grundlagen und die Beobachtung der Zustandsänderung des Systems während des Scanvorgangs.

### 7.2.1 Abszisse und Applikate

Zur Ermittlung der Abszisse  $x$  und der Applikate  $z$  betrachten wir das System von oben. Der Drehteller wird in die Quadranten  $Q_1$  bis  $Q_4$  unterteilt, welche den Drehwinkel klassifizieren. Der Winkel  $\gamma$  wird nur zur gedanklichen Unterstützung verwendet und in der Berechnung durch den bekannten Drehwinkel  $\beta$  ersetzt. Als erstes betrachten wir in Abbildung 11  $Q_1$ , danach wird das Modell auf die anderen Quadranten ausgeweitet.

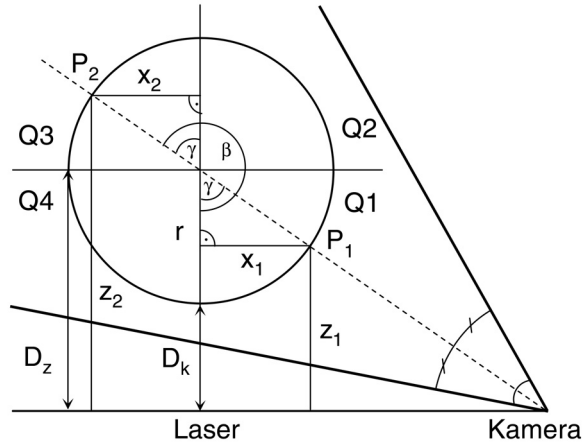


Abbildung 11: Modell zur Berechnung der kartesischen Koordinaten

Die Distanz  $r$  vom Rotationszentrum zum gemessenen Punkt ist überall bestimmt mit

$$r = |D_z - D_k|. \quad (13)$$

Für  $Q_1$  gilt die Bedingung  $0 \leq \beta < 90$ . Zur Berechnung des Abszissenwertes in  $Q_1$  gilt

$$\gamma = \beta. \quad (14)$$

Für Punkte im vorderen Bereich des Drehtellers ( $D_k < D_z$ ) ist somit

$$x = r \times \sin(\beta), \quad (15)$$

für Punkte im hinteren Bereich gilt

$$x = -r \times \sin(\beta). \quad (16)$$

In der schematischen Darstellung wird das durch  $P_1$  und  $P_2$  illustriert. Anstatt der Negation könnte der theoretische Drehwinkel  $\gamma$  für  $P_2$  auch mit  $\beta + 180$  ersetzt werden<sup>1</sup>. Im Hinblick auf die Verwendung in einem Algorithmus ist die fallweise Negation aber zu bevorzugen.

Um von  $Q_1$  zu den anderen Quadranten zu gelangen, soll nun zuerst der jeweils für die Überlegung relevante Winkel  $\gamma$  bestimmt werden.

Aus 15 und 16 ergibt sich für  $Q_1$  und  $Q_2$  im vorderen Bereich  $D_k < D_z$ :

$$x = r \times \sin(\beta) \quad (17)$$

und im hinteren Bereich

$$x = -r \times \sin(\beta). \quad (18)$$

<sup>1</sup> $\sin(\alpha) = -\sin(\alpha + 180)$

$Q$	<i>Bedingung</i>	$\gamma$	$\sin(\gamma)$
1	$0 \leq \beta < 90$	$\beta$	$\sin(\beta)$
2	$90 \leq \beta < 180$	$180 - \beta$	$\sin(\beta)$
3	$180 \leq \beta < 270$	$\beta - 180$	$-\sin(\beta)$
4	$270 \leq \beta < 360$	$360 - \beta$	$-\sin(\beta)$

Tabelle 5:  $\sin(\gamma)$  in Abhängigkeit vom Drehwinkel  $\beta$

Für  $Q_3$  und  $Q_4$  gilt dann genau die Umkehrung, also für  $D_k < D_z$

$$x = -r \times \sin(\beta) \quad (19)$$

und

$$x = r \times \sin(\beta) \quad (20)$$

für  $D_k \geq D_z$ . Bei der Implementation ist darauf zu achten, dass der Fall  $\beta \geq 360$  wieder mit der Regel für  $Q_1$  behandelt wird.

Aus der Kenntnis des Abszissenwertes kann jetzt leicht der Wert für die Applikate errechnet werden. Betrachtet man wiederum das rechtwinklige Dreieck mit Hypotenuse  $r$ , dann entspricht  $z$  der Ankathete von  $\gamma$ . Deren Länge ist für alle Quadranten bestimmt mit

$$z = \sqrt{r^2 - x^2}. \quad (21)$$

Für die korrekte Lage ist lediglich das Vorzeichen in Abhängigkeit von  $\beta$  zu ermitteln. Es gilt in  $Q_1$  und  $Q_4$  für Punkte im vorderen Bereich  $D_k < D_z$

$$z = -\sqrt{r^2 - x^2} \quad (22)$$

und im hinteren Bereich

$$z = \sqrt{r^2 - x^2}. \quad (23)$$

Für  $Q_2$  und  $Q_3$  gilt dementsprechend genau das Umgekehrte.

## 7.2.2 Ordinate

Als Vorbedingung wird festgelegt, dass das Linsenzentrum der Kamera höher liegt als der Drehteller und die Kamera in horizontaler Richtung auf die Szene blickt.

Zur Bestimmung der Ordinate wird nun, wie unter 4.3 beschrieben, der Zeilenindex  $j$  jedes Punktes ermittelt. Um von diesem Index zu einer Masseinheit zu gelangen, muss er mit einem Umrechnungsfaktor  $c$  multipliziert werden. Die Bestimmung dieses Faktors geschieht empirisch aus dem Testbild eines Objektes mit bekannten Dimensionen, wobei die sichtbare Seite des Objektes auf dem Koordinatenursprung, also auf dem Zentrum des Drehtellers liegen muss. Mit einem Bildbearbeitungsprogramm kann jetzt leicht das Verhältnis von Bildpunkten zur Masseinheit bestimmt werden.

Da die Szene von der Kamera perspektivisch erfasst wird und nicht analog zu einem Flachbettscanner jeder Punkt aus einer Normalposition separat abgetastet wird, entsteht eine optische Verzerrung. Verdeutlicht wird dies, wenn man gedanklich die Sensorebene parallel verschiebt, so dass sie durch den Mittelpunkt des Drehtellers verläuft. Nun wird ersichtlich, dass die Ordinatenwerte nur für die Punkte mit  $D_{kc} = D_{zc}$  korrekt errechnet werden können. Für alle anderen Punkte entsteht in Abhängigkeit von  $D_{kc}$  ein um  $\Delta y$  von der Wirklichkeit abweichendes Resultat. Punkte unter der Bildmitte die nahe bei der Kamera liegen, würden zu weit unten abgebildet, Punkte darüber zu weit oben. Diese Tatsache muss somit in die Formel zur Bestimmung der Ordinatenwerte einfließen, welche ich nachfolgend unter Betrachtung der Abbildungen 12 und 13 herleite.

Zuerst muss erwähnt werden, dass für die Berechnungen zur Ordinate die Distanzen zur Kamera  $D_{kc}$  und  $D_{zc}$  massgeblich sind und nicht wie unter 7.1 und 7.2.1 der Abstand zur Referenzebene.  $D_{kc}$  und  $D_{zc}$  sind immer bekannt mit

$$D_{kc} = \sqrt{D_k^2 + b^2} \quad \text{bzw.} \quad D_{zc} = \sqrt{D_z^2 + b^2} \quad (24)$$

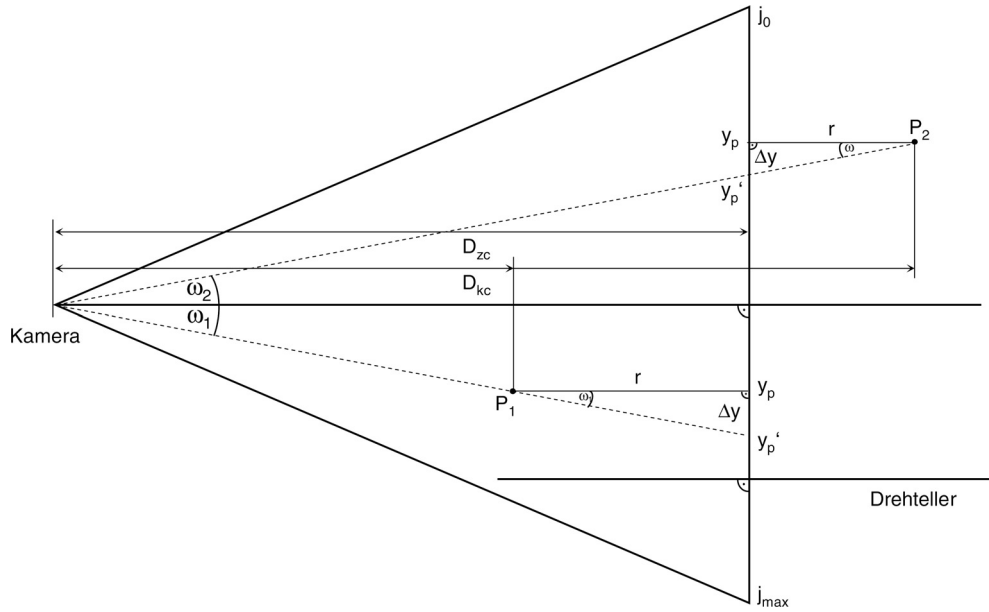


Abbildung 12: Seitliche Ansicht: Vertikale Verzerrung in Abhängigkeit vom Abstand zur Kamera

Durch Multiplikation des Zeilenindex  $j$  eines Punktes  $P$  mit dem Faktor  $c$  erhalten wir den Wert  $y'_p$ .

$$y'_p = j_k \times c \quad (25)$$

Es errechnet sich der Winkel  $\omega$  für jeden Punkt mit

$$\omega = \arctan\left(\frac{\frac{j_{max}-j_0}{2} \times c - y'_p}{D_{zc}}\right) \quad (26)$$

Der orthogonale Abstand des Punktes  $P$  von der durch den Mittelpunkt verschobenen Projektionsebene beträgt somit

$$r = |D_{zc} - D_{kc}|, \quad (27)$$

wobei  $D_{kc}$  die für den Punkt berechnete Distanz zur Kamera ist. Auf das Betragszeichen muss im Folgenden verzichtet werden, da das Vorzeichen zur Bestimmung der Lage vor oder hinter dem Drehtellerzentrums verwendet wird. Da  $\omega$  bekannt ist, ergibt sich für die Abweichung  $\Delta y$  des ermittelten  $y$ -Wertes  $y'_p$  von der Realität

$$\Delta y = \tan(\omega) \times r. \quad (28)$$

Aus 26 und 28 folgt somit für den korrigierten Wert der Ordinate:

$$y = y'_p + \frac{\frac{j_{max}-j_0}{2} \times c - y'_p \times r}{D_{zk}} \quad (29)$$

Diese Korrektur der Ordinate entspricht somit einer Normalprojektion des Bildpunktes auf die durch  $D_{zc}$  verlaufende, parallel verschobene Projektionsebene.

Da der Zeilenindex üblicherweise bei 0 beginnt und bei  $k_{max}$  endet, wird schliesslich der errechnete, korrigierte Höhenwert  $y$  von  $j_{max} \times c$  subtrahiert. Dadurch erhält man den Höhenwert relativ zum unteren Bildrand. Um die Lage des Punktes in Bezug auf den Drehteller zu erhalten, kann der Zeilenindex des Drehtellerzentrums ( $j_z$ ) ermittelt und danach mit dem Faktor  $c$  quantifiziert werden. Die Höhe des Punktes über dem Drehteller ergibt sich somit aus

$$y = y - (j_{max} - j_z) \times c. \quad (30)$$



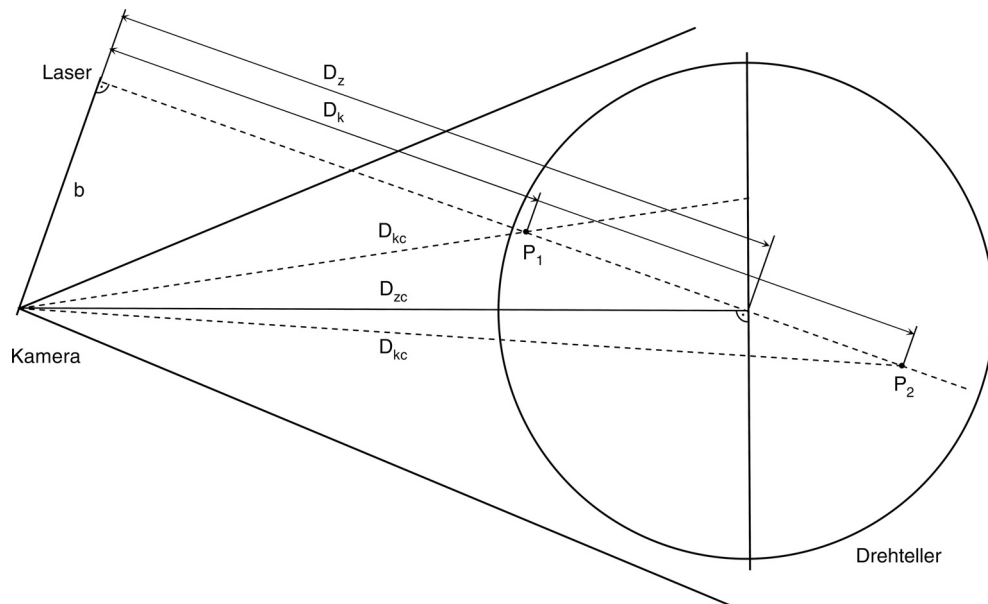


Abbildung 13: Grundriss: Bestimmung des Abstandes zur Kamera aus der gemessenen Distanz um Laser

Es ist anzufügen, dass die optische Verzerrung nur bei Objekten mit grosser Breiten- respektive Tiefenausdehnung ins Gewicht fällt. Bei Objekten mit annähernd zylindrischer Form, die senkrecht auf dem Drehteller platziert werden, ist sie gering.

### 7.3 Erweiterung

In diesem Kapitel wurde immer vorausgesetzt, dass die Kamera in horizontaler Richtung auf die Szene blickt. Dadurch stehen als Grundlage für die Berechnung immer rechtwinklige Dreiecke zur Verfügung. Soll nun das Modell auf beliebige Aufsichtswinkel der Kamera auf den Drehteller ausgedehnt werden, ist es nötig alle Relationen die auf dem *Satz des Pythagoras* beruhen dahingehend zu modifizieren, dass sie dessen Verallgemeinerung, dem *Cosinussatz*, genügen. Als zusätzliche Variable würde in diesem Fall der Aufsichtswinkel benötigt. Dieser müsste, wie auch die horizontale Ausrichtung, manuell eingestellt und vermessen werden. Auf Grund der Tatsache, dass die horizontale Sicht auf die Szene die grösste Flexibilität in Bezug auf die Form der erfassbaren Objekte bietet, wird in dieser Ausführung auf diese Verallgemeinerung verzichtet.

### 7.4 Ausgabe

Die Ausgabe der resultierenden Punktwolke geschieht in eine Textdatei. Darin sind mit Komma getrennt die kartesischen Koordinaten  $x$ ,  $y$  und  $z$  für jeden Punkt in einer Zeile enthalten. Dabei ist das Zentrum des Drehtellers der Ursprung des Koordinatensystems und eine Masseinheit entspricht einem Millimeter. Das Textformat wurde deshalb gewählt, weil im Rahmen dieses Projektes nicht abgeklärt wurde, mit welcher Applikation und Formaten das Resultat nachher verarbeitet werden soll und damit eine Weiterverarbeitung mit beliebigen Programmen ermöglicht wird.

## 8 Präzision

Nach den Berechnungen zur Auswertung soll nun eine qualitative Bewertung der erzielbaren Präzision vorgenommen werden. Es ist festzuhalten, dass die Genauigkeit entscheidend von der Messgenauigkeit der Kalibrierung unter 6.2.1 abhängt. Vor allem bei komplexen Objekten mit Positionen vor und hinter dem Drehtellerzentrum, oder bei nicht zentrisch platzierten Objekten schlagen sich Messfehler in Lücken oder Überschneidungen der Punktwolke nieder.

### 8.1 Auflösung

Die Auflösung der Kamera von  $640 \times 480$  Bildpunkten ist ausreichend. Um von der Auflösung bestmöglich zu profitieren, muss die Kamera, unter der Restriktion der Sichtbarkeit der Drehteller-Stirnseite und der horizontalen Betrachtungsrichtung, so nahe wie möglich positioniert werden. Dadurch ergibt sich aber auch eine Einschränkung in der Objektgrösse, die dann bei der für dieses Projekt verwendeten Aufstellungsgeometrie maximal 20 cm in der Breite und 13 cm in der Höhe betragen darf. Von der Basis  $b$  hängt die Fähigkeit des Systems ab, Vertiefungen des Objektes zu erfassen. Die Kameraposition nahe beim Laser ist dafür optimal, da sie fast parallel zur Laserebene blickt und so Einsicht in Vertiefungen erhält. Dementsprechend ist eine grosse Basis für komplexe Objekte zu vermeiden. Bei einfachen Objekten wie Quadern, Zylindern, Kugeln etc. hat eine grosse Basis aber auch einen entscheidenden Vorteil. Durch die diskrete Abtastung mit einem digitalen Sensor wird die Abbildung des Laserstreifens für alle nicht planen *und* senkrechten Oberflächen einen „Treppeneffekt“ aufweisen. Die limitierte Auflösung des Bildsensors hat somit zur Folge, dass nur grobe Abstufungen in der Distanzmessung möglich sind, was in der resultierenden Punktwolke als wellige Oberfläche erkennbar wird. Die Abbildung 8.1 zeigt dreimal das gleiche Objekt, mit drei verschiedenen Basisdistanzen aufgenommen. Während die Kugeloberfläche im ersten Modell relativ glatt rekonstruiert werden könnte, ist im letzten Bild eine deutliche Unregelmässigkeit zu erkennen.

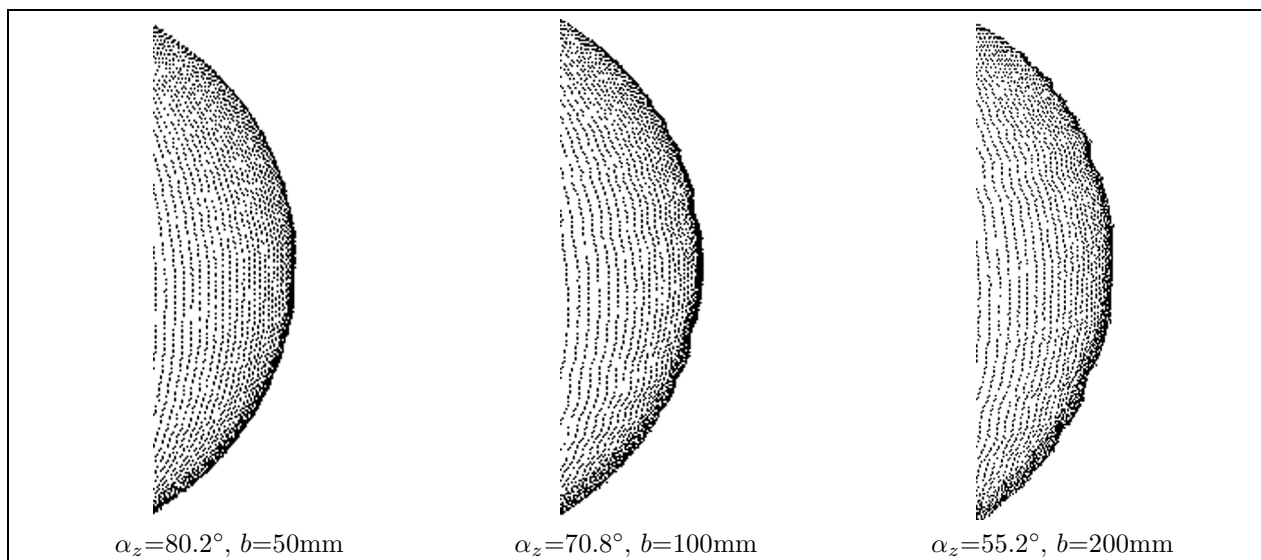


Abbildung 14: Artefakte durch diskrete Abtastung in Abhängigkeit von der Basisdistanz  $b$

Je nach Form und Oberflächenstruktur des Objektes ergibt sich somit eine entsprechende ideale Basis. Die Abwägung zwischen Genauigkeit und Erfassbarkeit (Vertiefungen) bleibt dem Anwender überlassen.

## 8.2 Optische Verzerrung

In Abschnitt 7.2.2 wurde ein Verfahren vorgestellt, das die Umrechnung des perspektivischen Bildes in eine normalisierte Darstellung ermöglicht. Die damit korrigierten Daten aus der Segmentation liefern gute Ergebnisse. Nebst dieser Notwendigkeit wurde jedoch aus Zeitgründen und mangelnden Kenntnissen in C++ auf eine Kamerakalibration gemäss [Tsa87] unter Verwendung der Software von [Wil06] verzichtet.

## 9 Benutzeroberfläche

Die Benutzeroberfläche besteht aus fünf Komponenten:

**Application Control:** Steuerungsfunktionen über ein Menu

**Laserscanner Capture:** Anzeige des *JMF*-Videobildes, Voransicht für die Segmentation

**Adjust:** Grundlegende Einstellungen zu Filtern und Positionierung

**Display Control:** Steuerung der 3D-Anzeige

**Laserscanner 3D Output:** Anzeige der dreidimensionalen Modelle mit *Java OpenGL*

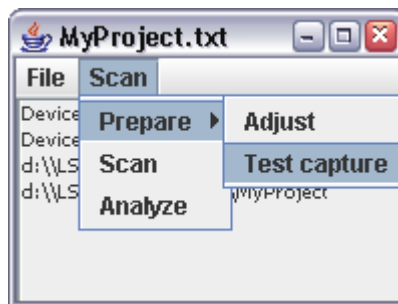


Abbildung 15: Fenster mit Video- und Segmentationsvorschau

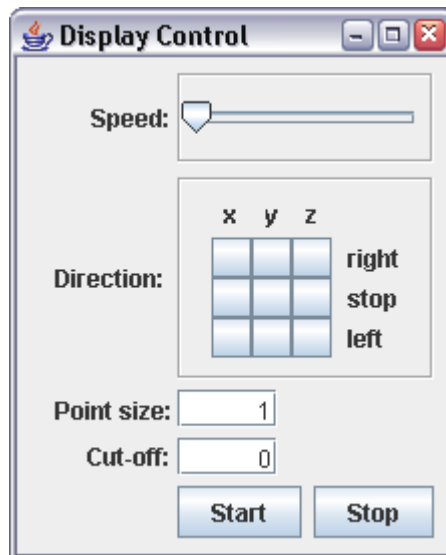


Abbildung 16: Steuerung der 3D-Ansicht

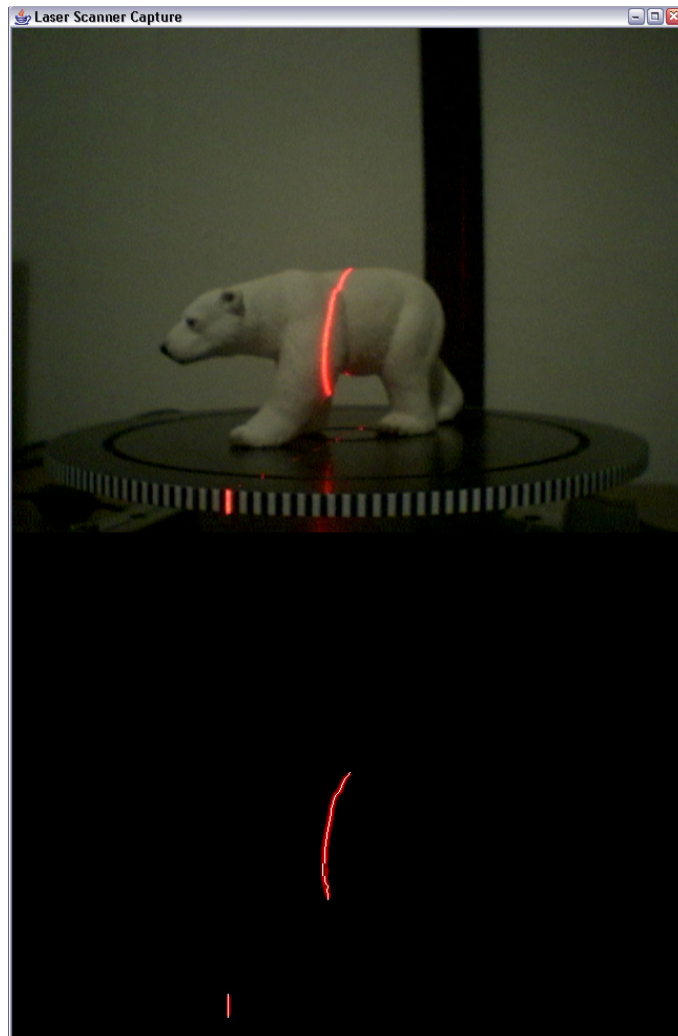


Abbildung 17: Fenster mit JMF-Video und Segmentationsvorschau

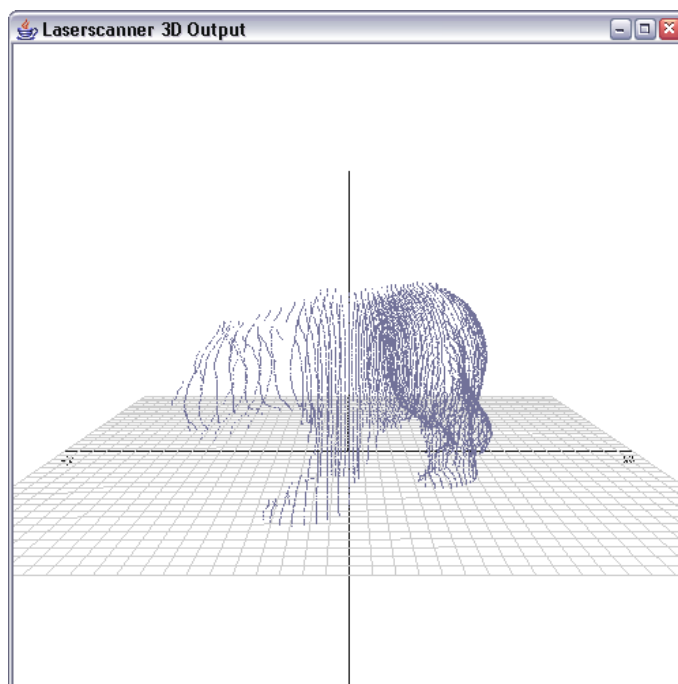


Abbildung 18: Java OpenGL Implementation zur Betrachtung der Punktwolken

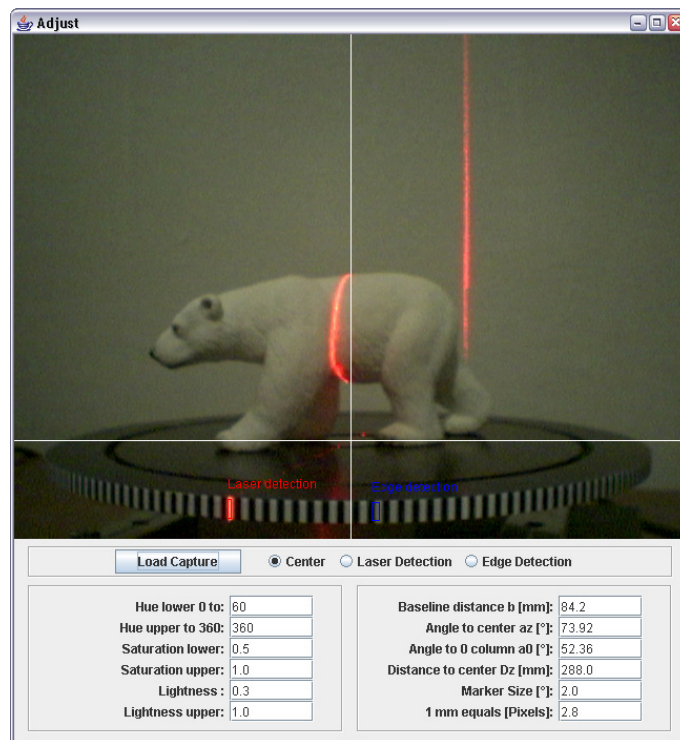


Abbildung 19: Verwalten von Applikationseinstellungen

## 10 Vergleichbare Ansätze

Die Vielfalt an kommerziellen Produkten ist sehr gross, jedoch lassen sich diese nur schon wegen der Preisdifferenz nicht mit dem vorgestellten Konzept vergleichen. Deshalb soll hier nur zwei „low-cost“ Ansätze angeführt werden.

### 10.1 Ein interaktiver Laserscanner

In [Tak03] wird ein kostengünstiger 3D-Scanner vorgestellt, der auf einer Digitalkamera und einem Laser basiert. Der Laser projiziert einen Punkt und kann von Hand über das Objekt bewegt werden. Immer sichtbar sein müssen aber drei LED Marken auf dem Gerät. Mittels Mustererkennung wird die Position der Marken und des Laserpunktes im Bild ausgelesen und über eine Triangulation die Koordinaten des Schnittpunktes von Laserstrahl und Objektoberfläche berechnet. Dabei kann jedoch das Objekt nur auf der Kamera zugewandten Seite abgetastet werden. Der Benutzer hat die Möglichkeit, interessierende Bereiche genauer abzutasten und entscheidet schliesslich selbst über die Auflösung des Scans. Die Genauigkeit ist abhängig von der Erfassung der Leuchtdioden und dem Laserpunkt.

### 10.2 Einsatz von strukturiertem Licht

Mittels eines handelsüblichen Multimediaprojektors und einer seitlich dazu angeordneten Digitalkamera wird in [Roc01] das Streifen-Projektions-Verfahren umgesetzt. Der Projektor beleuchtet dabei die Szene mit einem Muster, welches von starken Farbübergängen geprägt ist. Deren Lage auf dem Objekt kann mittels Kantendetektion ermittelt und daraus über die optische Triangulation die Distanz zum Projektor gemessen werden. Diese Methode hat den Vorteil, dass die Auflösung dynamisch angepasst werden kann und pro Kamerabild gleich mehrere Streifen abgetastet werden können.

# 11 Resultate

Abschliessend sollen einige Resultate präsentiert werden. Im Anzeigemodul der Applikation können nur Punkte ausgegeben werden. Zur besseren Visualisierung bietet sich eine Vielzahl von Programmen an. Um die Punktwolke mit einem *Glyph* Filter zu visualisieren, habe ich *Paraview*<sup>1</sup> von *Kitware*<sup>2</sup> verwendet.

Für das Resultat von entscheidender Bedeutung ist, neben den Lichteinflüssen und der Scangeschwindigkeit, die Platzierung des zu scannenden Objektes. Für einfache Objekte empfiehlt sich eine zentrische Lage. Dadurch wird die Oberfläche gleichmässig abgetastet. Bei komplexeren Objekten führt dies aber dazu, dass Bereiche mit grösserem Abstand zum Zentrum weniger genau aufgelöst werden können.

## 11.1 Geometrische Formen

Anhand eines Lebensmittelbehälters soll die Anwendbarkeit des Verfahrens auf technische Objekte mit geometrischen Formen aufgezeigt werden. Hier würden schon kleinste Ungenauigkeiten sichtbar. Das Objekt ist wegen seiner zylindrischen Form idealerweise im Zentrum zu platzieren. Die perfekte Rundung bestätigt die Korrektheit der Formeln aus Kapitel 7. Kleinere Ausreisser sind aber wegen der beschränkt verfügbaren Auflösung enthalten. Zudem tritt auch die im vorherigen Kapitel beschriebene Treppenbildung auf.

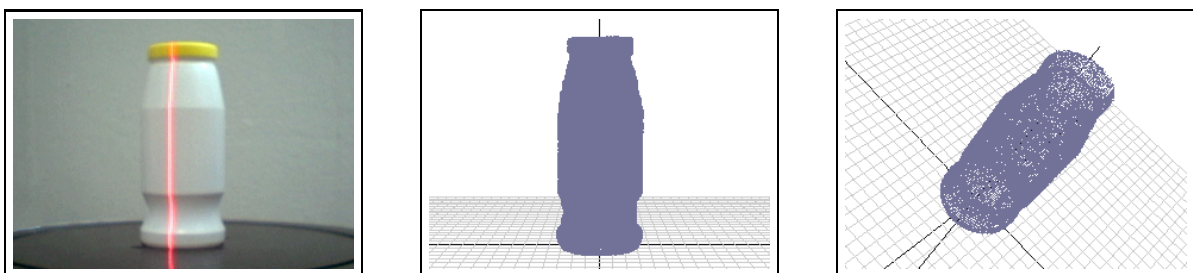


Abbildung 20: Original, Modelle von vorne und seitlich diagonal

## 11.2 Komplexe Formen

Als Beispiel für komplexere Objekte soll eine Elefanten-Figur aus Kunststoff gezeigt werden. Wegen der starken Lichtabsorbtion des Materials ist, um ein besseres Resultat zu erzielen, die Figur zuvor mit weisser Farbe bemalt worden. In Abbildung 21 steht die Figur zuerst tangential auf dem Drehteller. Um die Vorzüge dieser Aufstellung zu verdeutlichen, ist im Modell rechts das selbe Objekt im Zentrum platziert eingescannt. Je nach Verwendungszweck werden die Vorteile aber bei der einen oder anderen Variante liegen. Zur Illustration folgt in Abbildung 22 die selbe Figur, visualisiert mit *Glyph*.

---

<sup>1</sup><http://www.kitware.org>

<sup>2</sup><http://www.paraview.org>



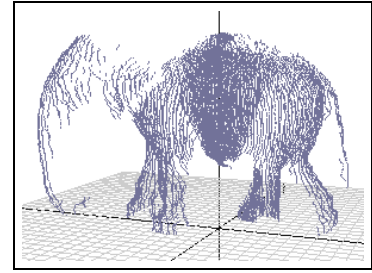
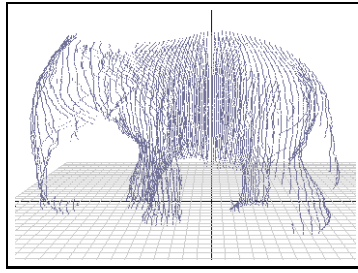
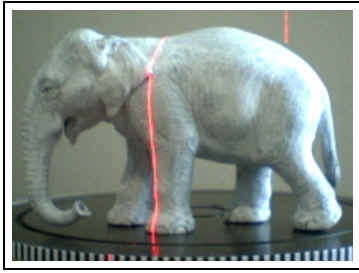


Abbildung 21: Original, Modelle azentrisch und zentrisch eingescannt

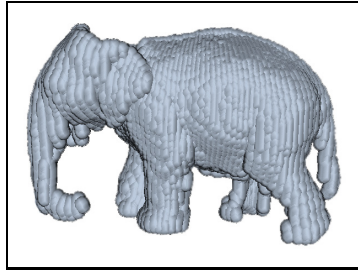


Abbildung 22: Visualisierung mit *Glyph-Sphere*

Schliesslich eine Figur die sich als gutes Testobjekt bewiesen hat. Eine Tonfigur, der Hase, ist wegen seiner Materialbeschaffenheit (wenig Absorbtion, geringe Reflektion) hervorragend geeignet.

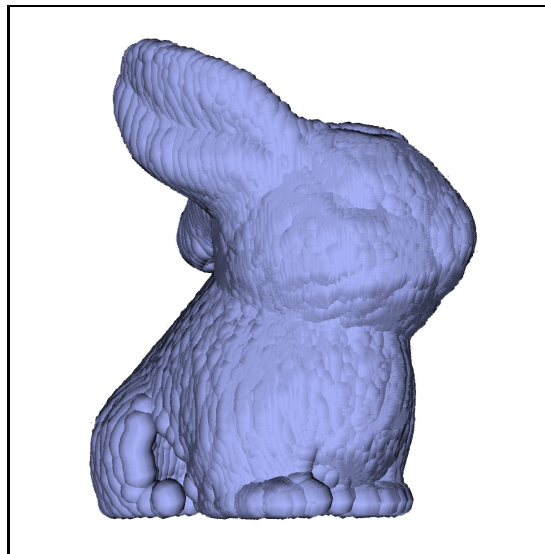


Abbildung 23: Tonfigur visualisiert mit *Glyph-Sphere*

# 12 Zusammenfassung

## 12.1 Stand der Arbeiten

Mit dem vorgestellten Projekt wird aufgezeigt, wie man mit der Technik der optischen Triangulation aus günstigen und leicht zu beschaffenden Komponenten einen 3D-Laserscanner bauen kann. Der Zugriff auf Bilder und Video einer Webcam mit Java funktioniert stabil und effizient. Die Erkennung des Laserstreifens im Bild kann als gut bezeichnet werden, ist jedoch bei gewissen Beleuchtungsszenarien nicht frei von Störungen. Um konstant Bilder von dem sich drehenden Objekt zu machen, wurde eine Methode zur Positionsermittlung vorgeschlagen, welche die Eigenschaften des Laserstrahls auf unterschiedliche gefärbten Marken berücksichtigt und zudem eine Feinkorrektur mittels Kantendetektion vornimmt. Die Ausgabe der Resultate geschieht in eine Textdatei und besteht nur aus den Koordinaten der Punkte. Die dadurch definierte Punktwolke kann mit dem programmeigenen Betrachter im dreidimensionalen Raum angezeigt und gedreht werden. Eine verbesserte Darstellung der Resultate mittels Implementation von *Nearest Neighbourhood* und *Meshing* ist sehr zeitaufwändig und anspruchsvoll. Basierend auf der Tatsache dass sowohl kommerzielle als auch frei verfügbare Produkte diese Berechnungen auf dem Resultat ausführen können, wurde darauf verzichtet.

## 12.2 Weiterentwicklung

Da es sich bei dem verwendeten Drehteller nicht um ein Standardprodukt handelt, ist eine Loslösung davon anzustreben. Die vorgestellte Erkennung der Marken kann problemlos für einen anderen, unter Umständen sogar selbst konstruierten Drehteller angepasst werden. Die Voraussetzung dafür ist nur eine gleichförmige Rotation. Zur Verbesserung der Ergebnisse in Bezug auf Abtastungslücken, wird vorgeschlagen die Ermittlung der Kameraposition (und in Abhängigkeit davon die Position des Lasers) durch eine räumliche Erkennung von Marken zu ersetzen. Dadurch könnte das Objekt von beliebigen Seiten manuell abgetastet werden.

## 12.3 Ungelöste Fragen

Die im Abschnitt 3.5 beschriebenen Probleme bei der Aneignung von Bildern mit einem konstantem Zeitintervall sind im Rahmen dieser Arbeit nicht bis ins Detail analysiert worden. Mit einer Verbesserung des Datendurchsatzes könnten auch andere Konzepte wie eine zeitbasierte Abtastung und höhere Geschwindigkeiten erreicht werden.

# Literaturverzeichnis

- [Van02] Yves Vandewoude et al. *A Java-interface to digital cameras*. Department of Computer Science, KULeuven, Celestijnenlaan 200A, Leuven, Belgium, Februar 2002
- [Roc01] C.Rocchini et al. *A low cost 3D scanner based on structured light*. Instituto di Sienza e Tecnologie dell'Informazione (ISTI), Consiglio Nazionale delle Richerche, C.N.R., Pisa, Italia, 2001
- [Tsa87] R. Tsai. *A versatile camera calibration technique for high accuracy 3D machine vision metrology using offthe-shelf TV cameras and lenses*. IEEE Journal of Robotics and Automation RA-3, 1987, Nummer 4.
- [Tsc96] René Tschirley. *Bildgebende Verfahren in der Medizin*. <http://cg.cs.tu-berlin.de/~pooh/uni/car/html/car.html>, Institut für Computer Graphics, Fachbereich Informatik, Technische Universität Berlin, 16. November 1996
- [Rod01] Lawrence H. Rodrigues. *Building imaging applications with Java technology*. Addison-Wesley, 2001
- [Cur99] Brian Curless. *From range scans to 3D models*. Computer Graphics, November 1999
- [Sie01] Siegmur Geiselberger. *Gebäude-Bestandesaufnahmen bei Baudenkmälern mit Laser Scannern*. <http://www.dombauwien.at/cit/pdf/bestandsaufnahme-dome-laser-scannen.pdf>, CAD-Stelle Bayern, Bayerische Staatsbauverwaltung - Hochbau, 18.07.2001
- [Fis94] Bob Fisher et al. *Hyper media image processing reference*. [http://www.cee.hw.ac.uk/hipr/html/hipr\\_top.html#reference](http://www.cee.hw.ac.uk/hipr/html/hipr_top.html#reference), Department of Artificial Intelligence University of Edinburgh, UK, 1994, Kapitel Digital Filters
- [Dav05] Andrew Davidson. *Java programming for games*. Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, P.O.Box 2 kohong, Hatyai, Songkhla, Thailand, 2005, Kapitel 28.7
- [Wi106] Wikipedia. *Kantendetektion*. <http://de.wikipedia.org/wiki/Kantenerkennung>, zuletzt besucht am 03.03.2006
- [Bos05] Stefan Bosse. *Laseroptische Time-Of-Flight Abstandsmessung*. BSSLAB Dr. Stefan Bosse, Research and Development Laboratory, Vohnen Strasse 86, 28201 Brehmen, Deutschland
- [For04] Josep Forest et al. *Laser stripe peak detector for 3D scanners. A FIR filter approach*. University of Girona, 2004.
- [Tak03] Masahiro Takatsuka et al. *Low-cost interactive active range finder*. Machine Vision and Applications, Springer Verlag, 18. Juni 2003
- [KKS96] Reinhard Klette et al. *Räumliche Information aus digitalen Bildern*. Technische Universität Berlin, 1996, Seiten 346-351
- [Wi206] Wikipedia. *Schwellwertfunktion*. <http://de.wikipedia.org/wiki/Schwellwert-Verfahren>. zuletzt besucht am 03.03.2006

[Spl06] Open Source Initiative *Sun public license*. <http://www.opensource.org/licenses/sunpublic.php>, Zulezt besucht am 2. Main 2006

[Wil06] Reg Willson. *Tsai's camera model*. <http://www.cs.cmu.edu/~rgw/TsaiDesc.html>, zuletzt besucht am 30.03.2006, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA

# A Installation

In diesem Anhang werden die zum Betrieb der Applikation erforderlichen Programmkomponenten von Drittanbietern beschrieben und deren Installation erläutert. Die Installation wurde auf einem Rechner mit *Microsoft Windows XP* durchgeführt.

## A.1 Kamera

Die *iSight* Webcam von *Apple* wird über die Firewire Schnittstelle mit dem Rechner verbunden. Eine Treiberinstallation ist dazu nicht nötig. Unter Windows ist die Kamera als generisches Bilderfassungsgesät ansprechbar. Eine Registrierung als eigenständiges Gesät schlägt in jedem Fall fehl, da keine Treiber verfügbar sind. Die Kamera wird in der Systemsteuerung als unbekanntes Gesät angezeigt. Dieser Eintrag kann gelöscht, respektive die Kamera deinstalliert werden. Da von *Apple* keine Software für Windows angeboten wird, ist eine Konfiguration der Kamera mit einem Standardprogramm wie *NetMeeting* oder *MovieMaker* nötig. Dort lassen sich unter in der Videokonfiguration die Parameter für Helligkeit und Autofokus einstellen. Standardmässig stellt Windows das Bild der Kamera mit 320×240 Bildpunkten dar. Um die volle Auflösung von 640×480 Bildpunkten zu erreichen, ist es nötig den Treiber eines Drittherstellers zu verwenden. Für das vorliegende Projekt wurde zu Testzwecken zusätzlich eine *Phillips ToUcam II* verwendet. Deren Treibersoftware hat das Betriebssystem aktualisiert, so dass die volle Auflösung nachher auch für die *iSight* zur Verfügung stand.

## A.2 Java Runtime

Die Applikation wurde mit Standardbibliotheken aus *J2SE Development Kit 5.0 Update 6* entwickelt. Um die Anwendung auf einem Rechner zu verwenden, ist lediglich *Java Runtime Environment 5.0 Update 6* nötig.

## A.3 Java Advanced Imaging

Obwohl Java Advanced Imaging als Installationspaket verteilt wird, handelt es sich dabei nur um Java Bibliotheken. Sie müssen in den Klassenpfad der Java Virtual Machine kopiert werden.

## A.4 Java Media Framework

Zur Einbindung der Webcam wird *Java Media framework 2.1.1e* verwendet. Es handelt sich dabei um Bibliotheken in Java als auch in nativem Maschinencode. Deshalb steht JMF in einem Paket mit Installationsroutine zur Verfügung. JMF wird in einen eigenen Programmordner installiert. Grundsätzlich kann im Klassenpfad auf dieses Verzeichnis verwiesen werden um die Bibliotheken verfügbar zu machen. Alternativ können die *.jar* Dateien auch in den existierenden Klassenpfad der VM<sup>1</sup> kopiert werden. Nach der Installation muss die Webcam mit dem JMF Werkzeug *JMF Registry* registriert werden. Mit den von JMF zur Verfügung gestellten Steuerungselementen kann auch eine Auflösung gewählt werden die dann als Voreinstellung standardmässig verfügbar ist.

---

<sup>1</sup>Findet sich unter `../lib/ext/`

## A.5 Java to OpenGL

Von *Java to OpenGL* (auch *JOGL* genannt) gibt es viele unterschiedliche Versionen. Für dieses Projekt wurde die Version vom November 2004 verwendet. *JOGL* besteht aus drei Dateien, zwei Windows Bibliotheken und einer Java Bibliothek. Sie sind in die entsprechenden Verzeichnisse von JRE zu kopieren. Darstellungsqualität und -geschwindigkeit sind abhängig von der installierten Grafikkarte.

## A.6 Dateien

Auf der abgegebenen CD-ROM befinden sich neben den Quelldateien und den kompilierten Klassen alle Softwarekomponenten von Drittherstellern und die Javadoc-Dokumentation sowie eine Installationsanleitung.

# Abkürzungsverzeichnis

$\alpha_0$	Winkel der Spalte $k = 0$
$\alpha_k$	Winkel des Laserstreifens
$\alpha_z$	Winkel der Spalte $k = M/2$
$\beta$	Drehwinkel
$\delta$	Sub-Pixel Verschiebung
$\omega$	Winkel Bildmitte zu Bildpunkt, in horizontaler Richtung
$\phi'_k$	Winkel zwischen optischer Achse und Laserstreifen
$\phi_k$	Winkel zwischen Spalte $k = 0$ und Laserstreifen
$b$	Basisdistanz zwischen Laser und Kamera
$c$	Umrechnungsfaktor von Bildpunkt zur Masseinheit
$d$	Halber Durchmesser des CCD-Sensors
$D_0$	Distanz zum in der Spalte $k = 0$ gelegenen Objekt
$d_k$	Abstandsinkrement zwischen zwei Spalten
$D_z$	Distanz der Referenzebene zum Zentrum des Drehtellers
$D_{kc}$	Distanz Laserstreifen zur Kamera
$D_{zc}$	Distanz Drehtellerzentrum zur Kamera
$f$	(Theoretische) Brennweite der Kamera
$j$	Zeilenindex im Bild
$k$	Spaltenindex im Bild
$M$	Anzahl Spalten des Bildes
$N$	Anzahl Zeilen des Bildes
$Q_x$	Quadranten des Drehtellers
$r$	Distanz des Bildpunktes zum Zentrum des Drehtellers
$x$	Abbildungsverzerrung
$x$	Abszissenwert im Koordinatensystem
$x$	Applikatenwert im Koordinatensystem
$y'_p$	Mit Abbildungsverzerrung ermittelte Lage in ordinal Richtung

# Abbildungsverzeichnis

1	Dauer der Bildakquisition mit JMF . . . . .	11
2	Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Tageslicht . . . . .	14
3	Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Kunstlicht . . . . .	14
4	Originalbild, Schwellwert, Kantendetektion und HSL-Filterung bei Dunkelheit . . . . .	14
5	Originalbild, HSL-Filterung, Medianfilter und Resultat der Segmentation (bei Kunstlicht) . . . . .	17
6	Intensitätsverteilung nach der Filterung auf dem HSL-Farbmodell . . . . .	17
7	Zeitspanne zwischen zwei Bildern (mit drei Auslassungen) . . . . .	24
8	Montage von Laser und Kamera mit Standfüßen . . . . .	25
9	Laserscanner, bestehend aus Drehteller, Laser (u. links), Kamera (u. rechts) und Steuerungseinheit mit Benutzerschnittstelle (rechts) . . . . .	26
10	Berechnung der Distanzwerte mittels Spaltenindex $d_k$ . . . . .	27
11	Modell zur Berechnung der kartesischen Koordinaten . . . . .	29
12	Seitliche Ansicht: Vertikale Verzerrung in Abhängigkeit vom Abstand zur Kamera . . . . .	31
13	Grundriss: Bestimmung des Abstandes zur Kamera aus der gemessenen Distanz um Laser . . . . .	31
14	Artefakte durch diskrete Abtastung in Abhängigkeit von der Basisdistanz $b$ . . . . .	33
15	Fenster mit Video- und Segmentationsvorschau . . . . .	35
16	Steuerung der 3D-Ansicht . . . . .	35
17	Fenster mit JMF-Video und Segmentationsvorschau . . . . .	36
18	Java OpenGL Implementation zur Betrachtung der Punktwolken . . . . .	36
19	Verwalten von Applikationseinstellungen . . . . .	37
20	Original, Modelle von vorne und seitlich diagonal . . . . .	39
21	Original, Modelle azentrisch und zentrisch eingescannt . . . . .	40
22	Visualisierung mit <i>Glyph-Sphere</i> . . . . .	40
23	Tonfigur visualisiert mit <i>Glyph-Sphere</i> . . . . .	40



# Tabellenverzeichnis

1	Abwägung der Produkteigenschaften . . . . .	10
2	Kantenfilter-Operator nach Sobel . . . . .	13
3	Filterungskriterien im HSL Modell . . . . .	15
4	Weichzeichungsoperator mit Gauss-Verteilung . . . . .	17
5	$\sin(\gamma)$ in Abhängigkeit vom Drehwinkel $\beta$ . . . . .	30