INAUGURAL - DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich - Mathematischen
Gesamtfakultät
der Ruprecht - Karls - Universität
Heidelberg

vorgelegt von

Dipl. Math., Dipl. Phys. Manuel Glas
aus Stuttgart - Bad Cannstatt

Tag der mündlichen Prüfung: 27.06.2012

Methoden zur sechsdimensionalen Objektlageerkennung aus Tiefenbildern

Gutachter:	**Prof. Dr. Bernd Jähne**
	Prof. Dr. Reinhard Männer

Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde ein *neues Software-System zur Objektlageerkennung aus Tiefenbildern* unter teilweiser Hinzunahme bekannter Einzelalgorithmen komplett implementiert. Es umfasst die Bildaufnahme und Tiefenbildrekonstruktion aus den Daten einer Stereokamera mit Musterprojektion, die Segmentierung der Objektpunkte vom Hintergrund sowie die Grob- und Feinlageerkennung.

Ein zusätzlich integriertes *Modelltraining* erlaubt es, alle für die Lageerkennung eines Bauteiletyps notwendigen Grunddaten *automatisch* und *innerhalb kurzer Zeit* zu erfassen.

Die *Evaluation* des implementierten Gesamtsystem anhand eines für die Produktion bei der Robert Bosch GmbH repräsentativen Bauteilespektrums zeigte, dass mehr als 40 % der Bauteiletypen mit der erforderlichen Genauigkeit erkannt und damit im üblichen Anwendungsfall von einem Montageroboter zuverlässig gegriffen werden können. Die Taktzeitvorgabe von durchschnittlich 0,5 Sekunden pro Objektlage-Erkennungsvorgang wird ohne Hardwarebeschleunigung eingehalten.

Mit dem Ziel, das Spektrum der erkennbaren Bauteiletypen zu erweitern, wurden die verbliebenen Hauptursachen für Fehl-Erkennungen herausgearbeitet. An erster Stelle steht demnach eine unzureichende Tiefenbildrekonstruktion, die insbesondere im Falle von Glanzpunkten auf der Oberfläche der beleuchteten Objekte auftreten kann.

Im abschließenden Kapitel wird dargestellt, dass *Lichtfelder* ein zielführendes Konzept zur Tiefenbildrekonstruktion bieten, das gegenüber Glanzflecken robust ist und darüber hinaus durch Übertragung der in der Stereotiefenbildrekonstruktion angewandten *Methode der Musterprojektion* wesentlich verbessert und weiter ausgebaut werden kann.

Abstract

This thesis describes a *new software system for object position recognition from depth images*. It was completely implemented during the course of this work partially taking proven algorithms into account. The system comprises image acquisition and depth image reconstruction from data taken by a stereo camera with pattern projection, segmentation of object points from background and finally course and fine position recognition.

In addition a *fast model training* has been integrated which provides an *automatic* acquisition of all data needed for an arbitrary object type.

Due to the *evaluation* of the system on a component spectrum representative for assembly lines at Robert Bosch GmbH shows more than 40 % of the object types being recognized precisely enough to be reliably picked up by a robot. Cycle time requirements of 0.5 seconds have been met without hardware acceleration.

Primary causes of deviation in position accuracy were elaborated in order to broaden the spectrum of detectable components. In particular, insufficient depth image reconstruction causes position deviations, mostly due to highlighted surface points on the illuminated object.

The concept of *light fields* is introduced which not only avoids any impact of highlighted surface points but also can be improved substantially by *pattern projection*.

Inhaltsverzeichnis

1. Einleitung

1.1 Motivation

Wurde der Begriff *Roboter* zu Beginn des 20. Jahrhunderts noch als Fantasiekonstrukt von Romanautoren (Asimov, Čapek) geprägt, so zogen sie ab Mitte des 20. Jahrhunderts in immer mehr Bereiche unseres Lebens ein. Heute gibt es nach Angaben des IFR (International Federation of Robotics) Statistical Department mehr als 18 Millionen Roboter. Den Hauptanteil bildet mit 17 Millionen die Servicerobotik (Waschmaschinen etc.), aber auch die Industrierobotik unterliegt mit ca. 150000 in 2011 verkauften Robotern (IFR 2011) und ca. 1,2 Millionen insgesamt eingesetzten Robotern stetigem Wachstum.

Der Trend in der industriellen Robotik geht durch immer kürzer werdende Produktlebenszeiten weg von starren Abläufen und hin zu flexiblerem Einsatz. Hierbei ist das Greifen von Objekten eines der Kerneinsatzgebiete. Um auf wechselnde Objekte und variierende Objektpositionen reagieren zu können, werden Roboter mit zunehmend intelligenter Sensorik ausgestattet und die Algorithmik zur Verarbeitung der Sensordaten - das *Maschinelle Sehen* - gewinnt an Bedeutung.

1.2 Zielsetzung

Ziel der Arbeit ist die Objektlagebestimmung von dreidimensionalen Objekten in allen 6 Freiheitsgraden (3 translatorische, 3 rotatorische) bezogen auf ein Koordinatensystem im Raum. Ausgangspunkt hierfür sind 2,5D-Tiefenbilder, aus denen in einer Vorverarbeitungsphase ein *Modell* erstellt wird, das sich in Solllage innerhalb des Koordinatensystems befindet. Dieses umfasst eine Punktewolkendarstellung des Objektes und je nach Verfahren zusätzliche 3D-Merkmale. Aufgabe ist dann, eine Zuordnung zwischen Modellpunkten bzw. -merkmalen und Punkten bzw. Merkmalen einer aktuellen 2,5D-Aufnahme des Objektes zu finden. Ist die Zuordnung so gewählt, dass die Kombination der Merkmale im Modell und der Aufnahme bestmöglich übereinstimmen, so wurde das Modell mit der Aufnahme *gematcht*. Ist dieser Vorgang, das sogenannte *Matching*, abgeschlossen, soll die Objektlage so genau bestimmt worden sein, dass das Objekt von einem Roboter gegriffen werden kann.

Das Verfahren der 3D-Objektlagebestimmung aus Tiefenbildern findet in leicht abgeänderter Form zahlreiche Anwendungen. Ebenso umfangreich ist die Fülle der Vorschläge in der Literatur [KSA87, Hor87, TR04], das Problem zu lösen. Es kristallisieren sich im Wesentlichen drei Schritte heraus [TM06]. Erstens die *Segmentierung*, d.h. das Trennen

von Objekt und Hintergrundpunkten als Grenze zwischen Vorverarbeitung und Bildverarbeitung. Zweitens die *Groblageerkennung*, bei der die Herausforderung darin besteht, eine Euklidsche Transformation zu finden, die das Objekt grob mit dem Modell in Deckung bringt. Meist möchte man damit einen „guten“ Startwert für die folgende *Feinlageerkennung* finden, welche den dritten Schritt darstellt. Darin wird dann lokal das Funktional f aus dem Optimierungsproblem aus Kapitel 2.3 minimiert und davon ausgegangen, dass Modell und Punktewolke schon grob übereinstimmen.

Abbildung 1.1 zeigt eine Gliederung der zur 3D-Objektlagebestimmung notwendigen Arbeitsschritte.

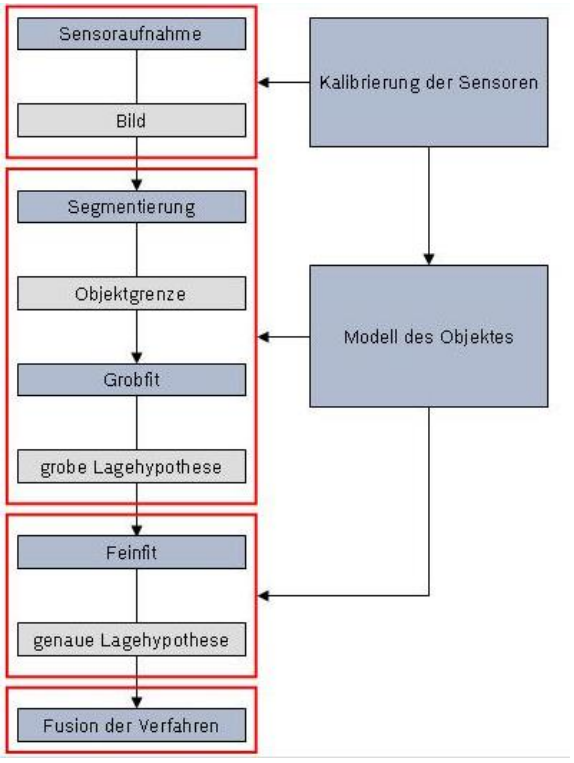


Abbildung 1.1: Übersicht über die für die 3D-Objektlagebestimmung notwendigen Arbeitsschritte

1.3 Eigener Beitrag

Einige der im Rahmen dieser Arbeit eingesetzten Algorithmen gibt es bereits. So wurde das Tiefenbildrekonstruktionsverfahren in ähnlicher Weise von [ZL02] vorgeschlagen, das verwendete Segmentierungverfahren (Shape Based Matching) wurde aus der Bildverarbeitungsbibliothek Halcon [Gmb80] benutzt und die Feinlageerkennung wurde mittels einer Implementierung (Aqsense) des ICP-Algorithmus von Chen [YC91] eingebunden.

Neu ist jedoch diese Algorithmen zu einem Gesamtsystem zusammenzufassen und eine Implementierung dieses Gesamtalgorithmus zu schaffen. Insbesondere wurden die oben genannten Algorithmen durch eine Projektorkalibrierung (im Rahmen einer von mir mitbetreuten Diplomarbeit [Dos09]) und durch eine Groblageerkennung mittels Hauptachsentransformation (eigener Beitrag) ergänzt.

Im industriellen Umfeld ist es Stand der Technik Objektlageerkennungen durchzuführen, um Bauteile beispeilsweise von einem Förderband abzugreifen. Hierbei wird für eine spezifi-

sche Aufgabenstellung üblicherweise ein genau darauf zugeschnittender Objekterkennungsalgorithmus entwickelt. Andererseits gibt es im Bereich der Servicerobotik Umsetzungen von Bilderkennungsverfahren, die ein größeres Spektrum von Objekten abdecken sollen [For09], [AMAD11]. Diese beruhen jedoch meist auf Texturerkennungsverfahren.

Im Rahmen der vorliegenden Arbeit soll ein Algorithmus entwickelt, implementiert und evaluiert werden, der (nahezu) *automatisch* auf ein beliebig geformtes Objekt trainiert werden kann, ohne dass das Objekt eine Eigentextur haben muss.

Des Weiteren werden Algorithmen oft, wenn überhaupt, nur auf Spielszenen evaluiert und auch nur der jeweils kleine Teilalgorithmus. Im Rahmen der vorliegenden Arbeit wird der *Gesamtalgorithmus* einschließlich automatischem Training auf einem *repräsentativen Bauteilespektrum* der Industrie evaluiert. Es wird gezeigt, dass dies zu Erkenntnissen bezüglich der breiten Anwendbarkeit und der möglichen Hauptfehlerursachen führt. Konkret wird eine Abdeckung des Bauteilespektrums von 40% erreicht und als Hauptfehlerursache eine mangelhafte Tiefenbildrekonstruktion wegen Glanzflecken ermittelt.

Aus den Hauptfehlerursachen wird ein Verbesserungsvorschlag hergeleitet. Ein Ausweg bietet nämlich das Gebiet der *Lichtfelder*. Dieses ist vor allem in der Computergrafik erforscht, findet aber mit neueren Entwicklungen auch in der Bildverarbeitung Anwendung. Die für die Objektlageerkennung erforderlichen Tiefenbilder können - neben der in der Stereobildverarbeitung üblichen Methode - auch aus Lichtfeldern berechnet werden [RZ07], [SW11]. In dieser Arbeit wird nun gezeigt, wie die in der Stereobildverarbeitung vorteilhaft eingesetzte Musterprojektion auch zur Verbesserung der Tiefenbildrekonstruktion aus Lichtfeldern eingesetzt werden kann und dass die Tiefenbildrekonstruktion aus Lichtfeldern auch an Bauteilstellen, auf denen sich ein Glanzfleck befindet, möglich ist.

1.4 Aufbau der Arbeit

In Kapitel 2 wird zunächst in das Thema der Objektlageerkennung aus Tiefenbildern eingeführt und ein Überblick über den Stand der Technik in verschiedenen Aspekten dieses Themas gegeben. Dies sind neben den Einzelschritten der Objektlageerkennung (Bildaufnahme, Segmentierung, Groblageerkennung, Feinlageerkennung) die Kalibrierung von Sensorsystemen und die Strukturierung ungeordneter Punktewolken.

Das darauffolgende Kapitel 3 beschäftigt sich mit der konkreten Auswahl und Umsetzung des Algorithmus, der in der vorliegenden Arbeit implementiert wurde. Kapitel 4 beschäftigt sich dann mit der Beschreibung der mit dieser Implementierung durchgeführten Experimente auf dem vorgegebenen Bauteilespektrum und deren Auswertung.

Die Schlussfolgerungen dieser Auswertung führen auf ein erst in letzter Zeit in der Bildverarbeitung aufkommendes Gebiet - das Gebiet der Lichtfelder, welche in Kapitel 5 erklärt werden. Ferner wurden zur eingehenderen Betrachtung erste Experimente durchgeführt, die auch in diesem Kapitel beschrieben und deren Ergebnisse ausgewertet werden.

Das letzte Kapitel 6 resümiert die Arbeit und gibt einen Ausblick auf mögliche Folgearbeiten.

2. Stand der Technik: Objektlageerkennung

2.1 Motivation

Die Objektlageerkennung ist ein Teil der Bildverarbeitung, welche Methoden aus Physik, Mathematik und Informatik benötigt. Physikalisch ist hierbei v.a. die Bildaufnahmetechnik, also die Umwandlung der zu „sehenden“ Szene in Daten, die von einem Algorithmus verarbeitet werden können. Mathematisch sind die Algorithmen, die aus den Eingangsdaten und Modellwissen eine Objektlage berechnen. Diese Aufgabe ist höchst rechenintensiv und benötigt um effizient gelöst werden zu können die Parallelisierungs- und Umsetzungsmethoden der Informatik.

In diesem Kapitel sollen alle Grundlagen für die vorliegende Arbeit erklärt werden. Da es sich um Grundlagen aus drei unterschiedlichen Gebieten handelt, werden die Teilgebiete auch leicht unterschiedlich beleuchtet, je nach später benötigtem Aspekt. Der einschlägig erfahrene Leser fühle sich frei, einen Teil oder auch alle Abschnitte dieses Kapitels zu überspringen und erst bei Bedarf auf die hier vorgestellten Definitionen und Erklärungen zurückzugreifen.

Der große Teil der hier vorgestellten Sachverhalte ist der Literatur des jeweiligen Arbeitsgebietes entnommen. Die Hauptquellen werden am Anfang jedes Abschnitts genannt. Sind spezielle Definitionen aus anderen Quellen entnommen, so wird direkt an der entsprechenden Passage darauf verwiesen.

2.2 Bildaufnahmemethoden in 2,5D

In diesem Abschnitt sollen Methoden zur Tiefenbildaufnahme behandelt werden. Hierbei handelt es sich nicht um eine echte 3D Bildaufnahme, also nicht um Voxel (dem 3D-Äquivalent zu den zweidimensionalen Pixeln). Vielmehr wird in jedem Bildpunkt zusätzlich der Abstand des zu sehenden Objekts zur Kamera ermittelt, was als 2,5D bezeichnet wird. Die hier behandelten Grundlagen können in Bildverarbeitungsbüchern vertieft nachgelesen werden. Mir diente [Jäh05] als Hauptquelle.

Für die Tiefenrekonstruktion gibt es fünf Grundprinzipien [Jäh05]: *Triangulation*, *Laufzeit*, *Phase (Interferometrie)*, *Kohärenz* und *Gestalt aus Schattierung (shape from shading)*.

In der Interferometrie und bei der Tiefenrekonstruktion aus Kohärenz werden Distanzen in Bruchteilen bzw. einigen wenigen Wellenlängen (Koheränzlänge) des verwendeten Lichts gemessen. Sichtbares Licht hat Wellenlängen zwischen 400 nm und 700 nm, weshalb diese Rekonstruktionsmethoden für die hier betrachteten makroskopischen Objekte mit, gemessen in dieser Skala, ebenso makroskopischen Oberflächensprüngen, nicht anwendbar sind.

Die übrigen Verfahren sowie typische Realisierungen werden in den folgenden Abschnitten vorgestellt und im Anschluss für unsere Aufgabenstellung bewertet.

2.2.1 Tiefe aus Triangulation

In diesem Abschnitt werden Verfahren zusammengefasst, die die Tiefe daraus berechnen, dass ein Objektpunkt aus zwei Richtungen sensiert wird. Mit bekannter Lage der beiden Blickwinkel zueinander, kann die Objektlage berechnet werden.

Stereoskopie

Betrachtet man ein Objekt aus zwei verschiedenen Blickwinkeln, so erscheint es in den Bildern jeweils an einem anderen Ort. Wenn die metrische Lage dieser Blickwinkel zueinander (extrinsische Kalibrierung), sowie das genaue Abbildungsverhalten der Kamera (intrinsische Kalibrierung) bekannt ist, und das Abbild eines Objektpunktes in einem Bild dem entsprechenden Abbild im anderen Bild zugeordnet werden kann (Korrespondenz), so kann daraus die Lage des Objektes berechnet werden. Diese Methode heißt *Stereoskopie*, der Abstand zwischen den beiden Blickwinkeln *stereoskopische Basis* b, die Differenz zwischen den Positionen im jeweiligen Bild *Disparität* d (siehe Abb. 2.1). Es gilt die einfache Formel [Jäh05]:

$$d = x_r - x_l = f' \frac{X + b/2}{Z} - f' \frac{X - b/2}{Z} = b \frac{f'}{Z}$$

Hierbei entspricht f' der Fokuslänge, bei unendlich weit entfernten Objekten.

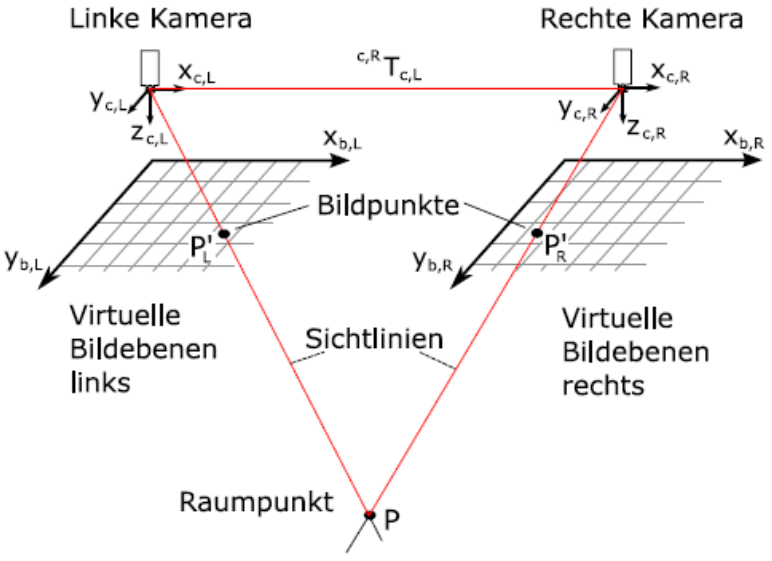


Abbildung 2.1: Stereoaufbau im Stereonormalfall (aus [Gmb80])

Üblicherweise werden, um das Korrespondenzproblem zu lösen, Bildregionen verglichen. Die betrachtete Szene muss jedoch entlang der stereoskopischen Basis eine nicht wiederkehrende Struktur aufweisen, damit die Zuordnung eindeutig lösbar ist. Beispielsweise kann in der Stereoaufnahme einer homogenen Fläche keine Tiefe gemessen werden.

Die Berechung kann dadurch vereinfacht werden, dass die Kamerazeilen parallel zur stereoskopischen Achse (s.o.) und die optischen Achsen der Kameras parallel ausgerichtet werden, sowie die Verzeichnung der Kameras herausgerechnet wird, so dass gerade Objektkanten auch im Bild gerade erscheinen. Man nennt diesen Vorgang *Rektifizierung* und diese Konfiguration *Stereonormalfall*. Korrespondenzen müssen dann nur noch in der gleichen Zeile der Kamerabilder gesucht werden.

Aktive Beleuchtung

Einer der Blickwinkel kann durch eine aktive Beleuchtung ersetzt werden. Dann unterscheidet man je nach Art der Lichtquelle *Lasertriangulation* und *Musterprojektion* (vgl. Abschnitt 2.2.3).

Bei Einsatz eines Lasers wird das Bild entweder mit einem Punkt oder einer Laserlinie abgetastet und aus dem Bild der Linie eine Tiefenkarte erstellt. Die Szene muss hierfür i.A. statisch sein. Probleme können bei stark glänzenden oder konkaven Oberflächen auftreten, da die Laserlinie dann gar nicht mehr bzw. durch Doppelreflexion an falscher Stelle im Kamerabild erscheint. Lasersysteme sind kommerziell erhältlich, kosten mehr als 1000 Euro und haben typische Abtastzeiten von wenigen Sekunden.

Bei Musterprojektionsverfahren unterscheidet man *One-Shot-Verfahren* und solche, die *Sequenzen* projezieren. Das Prinzip ist, dass die Szene flächendeckend mit einem eindeutigen Muster beleuchtet wird. Je nach Ort, an dem das Muster im Bild gefunden wird, kann die Tiefe des jeweiligen Szenenpunktes trianguliert werden.

Eine Klassifikation möglicher Musterarten bietet Salvi in dem Paper [JB97]. Hierin unterscheidet er neben den Anforderungen an die zeitliche Entwicklung der Szene (statisch oder dynamisch, also One-Shot-Verfahren oder nicht) die Art des projizierten Lichts (binär, Graustufen oder farbig) sowie die Abhängigkeit von Tiefensprüngen auf der Objektoberfläche. Wird ein periodisches Muster verwendet, limitiert dies die Tiefendiskontinuitäten auf der Objektoberfläche im Gegensatz zu absoluten Mustern, die beliebige Oberflächen zulassen.

Tiefe aus Fokussierung

Bei diesen Methoden wird Tiefeninformation über Bildserien, die unterschiedlich fokussiert sind, gewonnen. Das Prinzip beruht darauf, dass ein Objekt in der Fokalebene scharf abgebildet wird und die Unschärfe mit dem Abstand zur Fokalebene quadratisch zunimmt. Mathematisch ausgedrückt heißt das, die 3D-PSF (Point Spread Function) hat in der Fokalebene ein ausgeprägtes Maximum. Man kann also in einer Bildserie für jeden Objektpunkt das Bild ermitteln, an dem er den stärksten lokalen Kontrast hat und ihm diese Tiefe zuweisen. Dies ist natürlich nur dann möglich, wenn das betrachtete Objekt sehr fein texturiert ist. Vgl. hierzu [Jäh05].

2.2.2 Tiefe aus Laufzeit

Einen anderen Ansatz bietet die Rekonstruktion des Abstandes aus der *Laufzeit*. Hierbei macht man sich zu Nutze, dass die Lichtgeschwindigkeit eine Konstante ist. Sendet man also ein kodiertes Lichtpaket auf eine Oberfläche und empfängt die Reflektion wieder, kann man aus der Zeitverzögerung den zurückgelegten Weg errechnen.

Eine kommerziell erhältliche Technologie hierfür stellen die Laufzeitkameras (Time Of Flight) dar. Hierbei sind sowohl Sende- als auch Empfangseinheiten flächenhaft angeordnet, so dass ein flächenhaftes Bild in einem Schuss errechnet werden kann. Die Technik ist hinsichtlich möglichen Abstandsmessbereich eingeschränkt wegen der hohen Geschwindigkeit von Licht und dem Abflachen der reflektierten Intensität mit der Entfernung. Probleme ergeben sich vor allem durch Fremdlicht (keine Rekonstruktion) und Mehrfachreflektionen (falsche Rekonstruktion).

2.2.3 Musterrekonstruktion

Wie sich im Rahmen dieser Arbeit herausgestellt hat, bietet der Ansatz der Kalibrierung von Kameras gegen einen Musterprojektor viele Vorteile und wurde deshalb als Bestandteil unseres Systems gewählt. Deshalb soll in diesem Abschnitt auf den Stand der Technik der Anwendung dieses Verfahrens zur Oberflächenvermessung eingegangen werden.

Gegliedert ist dieser Abschnitt in vier Teile: Erstens die Art der Musterzuordnung. Zweitens die Musterextraktion im Bild (insbes. Farbsegmentierung) und drittens die daraus mögliche Rekonstruktion von Raumpunkten. Damit sind die Schritte beschrieben, die in einem Objekterkennungszyklus durchlaufen werden. Der vierte Teil beschäftigt sich mit der Projektorkalibrierung. Die konkrete Umsetzung der jeweiligen Schritte in unserem System ist im nächsten Kapitel im Abschnitt 3.3 erörtert. Die hier vorgestellten Ergebnisse wurden im Rahmen einer Diplomarbeit von S. Dose in der Abteilung CR/APA2 der Robert Bosch GmbH erarbeitet [Dos09]. Im folgenden Abschnitt werden die wichtigsten Ergebnisse vorgestellt, soweit sie zum Verständnis dieser Arbeit notwendig sind.

2.2.3.1 Art der Musterzuordnung

Im Rahmen dieser Arbeit wurde eine Abfolge von Farbstreifenmuster kodiert. Hierbei stellt sich das Korrespondenzproblem analog zur Stereorekonstruktion bei der Oberflächenvermessung mittels Projektor, d.h. welcher Stelle der Mustersequenz der im Bild betrachtete Abschnitt zuzuordnen ist. Hierfür gibt es in der Literatur zwei Gruppen von Verfahren, die *Dekodierverfahren*, die versuchen, kurze Bildabschnitte im Muster zuzuordnen, und die zweite Gruppe von Verfahren, die das Problem als ein *Kostenminimierungsproblem* formulieren und dieses mittels *Dynamischer Programmierung* lösen.

Dekodierverfahren

Ein typisches Beispiel eines Dekodierverfahrens ist von Forster [For06] beschrieben worden. Um die Suche nach Korrespondenzpunktpaaren zwischen Kamera und Projektor zu erleichtern, werden Kamerabild und Muster zunächst zueinander rektifiziert, d.h. die beiden Objektive werden entzerrt, sowie die Epipolarlinien (Bildzeilen) horizontal gestellt. Damit sind korrespondierende Punkte stets in der gleichen Zeile zu suchen.

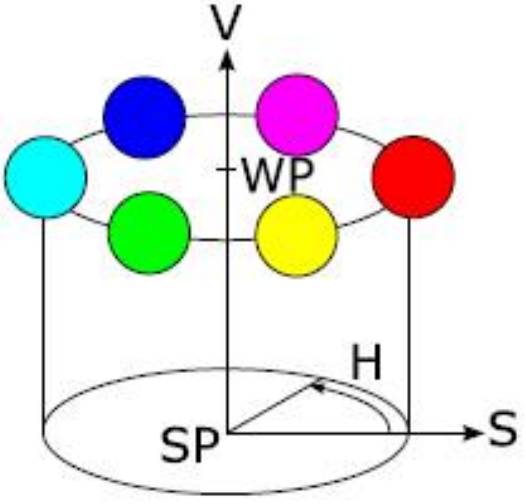


Abbildung 2.2: Schema des HSV-Farbraumes (**H**ue (Farbton), **S**aturation (Farbsättigung), **V**alue (Helligkeit))

Im Bild werden die Farbstreifen segmentiert, d.h. ermittelt, an welcher Stelle im Bild sich welcher Farbstreifen befindet. Diese Sequenz wird mit dem aufprojizierten Muster verglichen. Ab drei aufeinanderfolgenden Streifen ist eine Sequenz eindeutig.

Vorteil dieses Verfahrens ist die geringe Rechendauer. Nach [For06] erreichte sein Algorithmus bei einer Kameraauflösung von 780 x 580 Pixeln auf einem 3,2 GHz PC abhängig von der Menge der zu berechnenden Distanzwerte zwischen 18 (ca. 192 000 Werte) und 23 (ca. 83 000 Werte) Bilder pro Sekunde. Ferner erreicht er mit 0,01 mm - 0,04 mm Standardabweichung eine hohe Genauigkeit.

Nachteil ist die starke Abhängigkeit von einer guten Farbsegmentierung. Sequenzstücke sind erst ab einer Länge in Höhe der Ordnung der gewählten de Bruijn-Sequenz verwendbar, womit lückenhafte Segmentierungen zu sehr lückenhaften Rekonstruktionen führen. Inbesondere können solche Segmentierungslücken durch farbige Objekte oder Sprünge in der Objektoberfläche entstehen, womit Oberflächenglattheit eine weitere Voraussetzung ist.

Erweitert wurde das Verfahren in [For05] auf ein Stereokamerasystem mit Projektor, womit die Robustheit durch die Redundanz verbessert werden konnte, sich die Rekonstruktionszeit jedoch auf 1-2 Bilder pro Sekunde (bei oben genannter Hardware) reduzierte.

Kostenminimierungsverfahren mit Dynamischer Programmierung

Bei diesem Typ von Ansätzen, wird die Korrespondenzpunktsuche in ein Kostenminimierungsproblem überführt, indem in einer Tabelle z.B. vertikal alle Pixel des Bildes und horizontal alle Pixel des projizierten Musters aufgetragen werden. Jede Zelle der Tabelle entspricht dabei einer Pixelkorrespondenz, ein Pfad durch die Tabelle einer möglichen Lösung des Problems. Man startet links unten. Von jedem Punkt aus gibt es nur drei mögliche Pfade: schräg nach rechts oben - in Muster und Bild werden die nächsten Pixel einander zugeordnet; nach oben - das Pixel im Bild hat kein korrespondierendes Pixel im Muster. Es wird das nächste Pixel im Bild betrachtet, das Pixel des Musters wird beibehalten; nach rechts - wie nach oben nur mit vertauschten Rollen von Bild und Muster.

Bewegungen vertikal oder horizontal entsprechen also Sprüngen im Bild respektive Muster und werden daher mit zusätzlichen Kosten belegt. Als minimal gilt der Weg mit den geringsten Gesamtkosten. Man berechnet ihn Korrespondenz für Korrespondenz, d.h. man hangelt sich von Pixel zu Pixel und addiert zu den bisherigen Gesamtkosten nur die Kosten des aktuellen Schrittes. Diese Methode heißt *Dynamische Programmierung* und wurde in den 1940er Jahren vom amerikanischen Mathematiker Richard Bellman auf dem Gebiet der Regelungstechnik eingeführt [Bel57]. Mathematisch wichtig zur Anwendung dieses Prinzip ist es, dass das Problem tatsächlich in unabhängige Teilprobleme zerlegt werden kann, so dass sich die optimale Lösung aus optimalen Teillösungen zusammensetzt (*Optimalitätsprinzip von Bellman*). Dies stellt in unserem Fall eine Monotonie-Voraussetzung an die Oberfläche, d.h. auf dem Objekt weiter rechts liegende Merkmale müssen auch weiter rechts abgebildet werden (Die Monotonie ist z.B. auf dem Henkel der Tasse in Abbildung 2.3 verletzt).

Vorteil dieses Ansatzes ist, dass korrespondierende Merkmale (Pixel) auch gefunden werden, wenn sie nur ähnlich sind, so lange sie lokal optimal sind. Wählt man alle Pixel als Merkmal, können dichte Rekonstruktionen entstehen. Nachteil ist neben der hohen Rechenzeit die Fehlrekonstruktion, wenn die Monotoniebedingung nicht erfüllt ist (siehe Abbildung 2.3 Tasse).

Dies kann durch Rekursion (Entfernen der nicht gefunden Merkmale aus der Tabelle und Neuberechnung - vgl. [ZL02]) verbessert werden, bremst jedoch die Berechnung noch weiter. Bei einer Auflösung von 864x576 benötigte [ZL02] ca. 1 Minute auf Pentium III mit

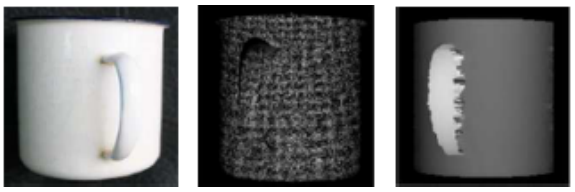


Abbildung 2.3: Rekonstruktion einer Tasse. l: Originalbild, m: Objekt mit projiziertem Muster, r: Tiefenkarte (aus [ZX07])

900 MHz. Auch [ZX07] schlägt eine Verbesserung vor, indem als Merkmale kleine Fenster mittels „Fast Normalized Cross-Correlation“, statt Einzelpixel, verwendet werden. Dies steigere die Robustheit und wäre in 3 bis 4 Sekunden pro Bild berechenbar (640x480 Pixel, 2,6 GHz CPU mit 1GB Speicher).

Der Hauptnachteil liegt jedoch in der Bestimmung der Strafkosten für horizontale bzw. vertikale Bewegung im Bild. Wählt man sie zu gering, springen die Korrespondenzen zu stark und es entstehen große Lücken im Bild. Wählt man sie zu hoch geht Objektoberflächenstrkutur verloren. Der richtige Wert variiert jedoch mit unterschiedlichen Objekten und Szenen.

Insgesamt bietet die Musterdekodierung, insbesondere mit zwei Kameras einen besseren Kompromiss für unsere Anwendung und bildet daher wie im nächsten Kapitel 3.3 beschrieben die Grundlage unserer Umsetzung.

2.2.3.2 Musterextraktion im Bild

Die im letzten Abschnitt beschriebenen Musterzuordnungsverfahren basieren auf im Bild gefundenen Merkmalen, also Musterstücken. Im Folgenden wird ein eindeutiges Musterstück ***Label*** genannt. In diesem Abschnitt wird Literatur vorgestellt, die sich mit der Segmentierung solcher Muster beschäftigt. Die vorliegende Arbeit konzentriert sich auf One-Shot Streifenmuster, da nur sie in unserer Sensorik verwendet werden.

Die Information, aus der die Tiefe gewonnen werden kann, ist die Position der Streifenmittelpunkte oder -übergänge, sowie die Farbe des jeweiligen Streifens. Die Verfahren unterscheiden sich in der Art und Weise, wie diese Positionen ermittelt werden. In [ZL02] wird ein Verfahren vorgestellt, bei dem in den Farbkanälen eines RGB-Bildes jeweils orthogonal zur Streifenrichtung Gradienten der Farbwerte berechnet werden. Aus den drei Werten wird jeweils die euklidsche Norm gebildet (Summe der Quadrate), deren Maxima die Kanten liefern.

Bei Zhang [ZL02] muß sich ein Streifen zum Nachbar in mindestens einem Kanal unterscheiden. Forster [For06] verwendet für die Segmentierung seiner Streifen bestehend aus Farben der acht Ecken des RGB-Farbraumes die Besonderheit, dass sich Streifen immer in mindestens zwei Kanälen binär (durch einen Intensitätssprung von 0 auf maximal oder umgekehrt) unterscheiden. Liegen in den berechneten Farbkanalgradientenbildern zwei Nulldurchgänge deckungsgleich, wird dies als Streifenübergang erkannt (andere Kanten im Bild haben meist nur *einen* Nulldurchgang). Zur subpixelgenauen Berechnung der Position wird eine Parabel an den Pixel des Maximums und seine beiden Nachbarpixel im Gradientenbild der einzelnen Farbkanäle gefittet. Der Scheitel der Parabel ist die präzisierte Position. Die Farben werden über die Übergänge von Streifen zu Streifen im Hue-Raum ermittelt.

2.2.3.3 Rekonstruktion von Raumpunkten - Funktionsprinzip

Nachdem die Label im Bild gefunden wurden, muss daraus die Tiefe berechnet werden. In diesem Abschnitt wird das zu Grunde liegende Triangulationsprinzip beschrieben.

Seien Kamera und Projektor im Stereonormalfall. Dann genügt es, eine Zeile im Bild zu betrachten. Bezeichne u die Position im Bild in dieser Zeile. Betrachte nun das n-te Label des Musters. Man kann sich die Projektion dieses Labels als Strahl vom Projektor in die Szene vorstellen, wie in Abbildung 2.4 illustriert. Da Kamera und Projektor in einem Abstand $b > 0$ zueinander stehen, verändert sich die Position u, an der das Label zu sehen ist, mit dem Abstand h vom Sensorsystem zum Objekt. Damit gibt es einen Zusammenhang zwischen Position im Bild und Abstand zum Sensor.

$$u = f(h)$$

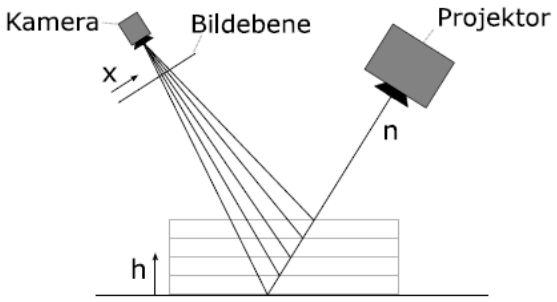


Abbildung 2.4: Schema der Triangualation des Abstandes eines Bildpunktes mit Hilfe einer Projektorkalibrierung (aus [Dos09])

Eine genaue Herleitung der Transformationsfunktion f findet sich im Konzeptkapitel 3.6.2 zusammen mit deren konkreter Berechnung während der Projektorkalibrierung.

2.3 Segmentierung

Nach der Aufnahme eines Tiefenbildes können Bildvorverarbeitungsschritte folgen, auf die an dieser Stelle nicht eingegangen wird. Sie dienen zur Verbesserung der Bildqualität (Glättung, Rauschkontrolle,...). Danach muss als erster Schritt der Bildverarbeitung erkannt werden, welche Bildpunkte Teil eines zu suchenden Objektes sind und welche zum Hintergrund gehören. Dabei entsteht ein Binärbild, in dem Objektpunkte mit 1 und Hintergrundpunkte mit 0 bezeichnet werden. Dieser Schritt heißt *Segmentierung*, genauer gesagt *vollständige, überdeckungsfreie* Segmentierung. Das heißt, jeder Punkt (vollständig) wird genau einer (überdeckungsfrei) Kategorie zugeordnet. Als Hauptquelle diente mir hier das Buch von B. Jähne [Jäh05].

In unseren Bildern entspricht der Grauwert eines Pixels dem Abstand des entsprechnenden Oberflächenpunktes zur Kameraebene. Darauf können elementare Bildverarbeitungsmethoden zur Segmentierung angewandt werden. Diese kann man in pixelbasierte, kantenbasierte und regionenorientierte Verfahren einteilen. Alle drei beruhen ausschließlich auf lokalen Informationen, wobei pixelbasierte Verfahren sogar nur den Grauwert des einzelnen Pixels verwenden. Kantenbasierte Verfahren sind sensitiv auf Diskontinuitäten und regionenorientierte Verfahren auf homogene Bereiche. Wenn die Form des gesuchten Objekts bekannt ist, können auch *modellbasierte Segmentierungsverfahren* verwendet werden. In den folgenden Abschnitten werden diese vier Methodentypen beschrieben.

2.3.1 Pixelorientierte Segmentierung

Dieses Verfahren bietet sich an, wenn das Grauwerthistogramm des Bildes klare Maxima aufweist. Dann wird, in einem möglicherweise vorverarbeiteten Bild, ein *globaler Schwellwert* gesetzt, wann ein Punkt den typischen Grauwert des Hintergrundes und wann den

des Objektes hat. Die Schwierigkeit liegt darin, den Schwellwert richtig zu wählen, da an Kanten immer Zwischenwerte auftreten.

Besser als ein Schwellwert für die Entfernung zur Kamera ist der Abstand zu einer Trennebene im Raum. Hierbei wird das Tiefenbild als Punktewolke im Raum betrachtet und alle Punkte, die von einer in der Trainingsphase eingelernten Tischebene nur wenig erhaben sind, werden als Hintergrund markiert. Hierfür muss natürlich die genaue Lage der Tischebene im Raum zum Objekterkennungszeitpunkt bekannt sein. Dies kann z.B. durch die Erkennung dreier vorher aufgebrachter Marker oder anderer Referenzierungsverfahren erreicht werden. In manchen Anwendungen ist der Sensorkopf relativ zum Tisch fest montiert, so dass keine Referenzierung in der Detektionsphase nötig ist. Die Segmentierung der Objekte wird also auf eine deutlich einfachere Erkennung der Umgebung zurückgeführt.

Für die Anwendung, dass einzelne Objekte auf ebenem Hintergrund erkannt werden sollen und sich deutlich von diesem abheben, stellt dies eine robuste und rechenzeiteffiziente Möglichkeit dar. Es ist auch denkbar, das Verfahren auf komplexere Hintergründe (z.B. eine Palette oder ein Prozessnest) zu erweitern. Hierbei muss im Training der Hintergrund durch dichte Rekonstruktion vollständig erfasst werden, oder das Teil über eine lokale Referenzebene erhaben sein, welche ebenfalls im Training ermittelt werden muss. In der Erkennungsphase muss von jedem Bildpunkt der senkrechte Abstand zum Referenzhintergrund berechnet und das Pixel bei zu geringem Abstand als Hintergrund markiert werden.

2.3.2 Kantenorientierte Segmentierung

Bei kantenorientierter Segmentierung wird nicht der Absolutwert der Helligkeit verwendet, sondern das Maximum der Ableitung. Sind diese Maxima (etwa mittels eines Sobel- oder Laplace-Operators oder einer Gradientensuche) gefunden, werden *Kantenverfolgungsalgorithmen* angewandt, um sie zu komplettieren, wofür es wiederum eine ganze Reihe von Algorithmen gibt.

In den Daten des im Rahmen dieser Arbeit verwendeten Sensorsystems hat man typischerweise keine geschlossenen Kanten, so dass Kantenverfolgungsalgorithmen ungeeignet sind.

2.3.3 Regionenorientierte Segmentierung

Bei *regionenorientierten Verfahren* sucht man nach homogenen Regionen im Bild. Hierzu gehören einerseits die pixelorientierten Verfahren wie Pyramid Linking [Jäh05] sowie die Standardverfahren Split and Merge, Region Growing, Region Splitting und Watersheds, bei denen auf die entsprechende Literatur verwiesen sei, etwa [AR82].

Eine mögliche Übertragung einer Region Growing Methode wurde vom Fraunhofer IPA in Stuttgart umgesetzt. Hierbei wird der oberste Wert der Tiefenkarte als Ausgangspunkt verwendet. Handelt es sich hierbei nicht um einen Ausreißer (gibt es also ausreichend viele plausible Punkte in der Umgebung), werden alle Punkte in einer Region um diesen Startpunkt als Objektpunkte definiert.

In anderen Fällen finden komplexere *Energiemethoden* Anwendung. Hierbei wird jeder möglichen Segmentierung ein Energiewert zugeordnet und das Funktional der Energiewerte minimiert. Mögliche Energien sind hierbei:

- Der Unterschied zwischen Segmentierung und Originalbild:

$$\int (u - u_0)^2$$

- Ein Maß der Länge der Kanten C zwischen segmentierten Bereichen, z.B. durch das zweidimensionale Haußdorff-Maß $H^2(C)$
- Ein Maß für die Inhomogenität in den einzelnen Bereichen R:

$$\int_{R \setminus C} |\nabla u|$$

- Ein Term, der die Abweichung von der erwarteten Objektform angibt, falls diese a priori bekannt ist.

Als mögliche Lösungsverfahren werden klassische *Variationsmethoden* aus der Physik oder *Graph Cut Verfahren* aus der Informatik eingesetzt.

2.3.4 Modellbasierte Segmentierung

Bei den bisher vorgestellten Methoden wurde, außer als optionaler Term im Energiefunktional, kein Vorwissen über die zu suchenden Objekte verwendet. Ist die Form jedoch bekannt, kann man diese in die Segmentierung einfließen lassen. Eine Möglichkeit hierfür stellen die sogenannten *Merkmalbasierenden Verfahren* dar. Diese werden bei den Grobregistrierungsmethoden 2.4.2 erläutert, da sie neben der Segmentierung direkt eine Objektlageschätzung ergeben. Ergänzend bleibt noch das *Template Matching* zu erwähnen, bei dem das gesamte Bild nach gegebenen Vorlagen durchsucht wird.

2.3.5 Clustering

Ist das Binärbild erstellt, durch das Objektpunkte markiert sind, müssen Regionen in diesem Bild zu einzelnen Objekten geclustert werden. Die Herausforderung hierbei ist, einen Clusteralgorithmus verwenden zu müssen, bei dem die Anzahl der Cluster zu Beginn noch nicht bekannt ist. Erschwerend kommt hinzu, dass die Objektgröße im Bild stark mit dem Blickwinkel variiert. Vereinzelte, deutlich zu kleine Cluster können aussortiert werden. Ist ein zusammenhängender Cluster deutlich zu groß für ein einzelnes Objekt, handelt es sich um zwei aneinander angrenzende Objekte. Die Schwierigkeit besteht dann darin, diese zu trennen. Das kann nur mittels Modellwissen erreicht werden, d.h. es müssen Segmentierungsverfahren verwendet werden, die *Formen* erkennen, wie das Template Matching.

2.4 Groblageerkennung

In der Literatur werden vielerlei Methoden zur automatischen Groblageerkennung (Grobregistrierung) vorgeschlagen, deren Effektivität jedoch stark von der Vollständigkeit, Genauigkeit, Dichte und Verteilung der ermittelten Punktewolke, ihrer Deckungsgleichheit zum Modell, sowie der Güte der Segmentierung abhängen.

Deckungsgleichheit von Modell- und Objektpunktewolke bedeutet in diesem Zusammenhang, dass das Funktional f aus Kapitel 2.3 im globalen Minimum null ist. D.h. insbesondere, dass zu jedem Punkt der Punktewolke der entsprechende Punkt des Modells in den Dreiecksflächen der Triangulierung enthalten und nicht nur angenähert ist. In realen

Modellen wird das Funktional also nie *gleich* null sein, so dass eine Schwelle gesetzt werden muss.

Es kristallisieren sich zwei Klassen von Grobregistrierungsverfahren heraus. *Globale Verfahren* verwenden Invarianten, die aus der Ganzheit des Objekts berechnet werden. Diese Verfahren sind typischerweise schnell, aber wenig robust gegen Verdeckungen und Glanzpunkte. *Lokale Verfahren* verwenden dagegen nur lokale Objektinformation und können deshalb prinzipiell auch teilweise verdeckte Objekte erkennen.

2.4.1 Globale Verfahren

In diesem Abschnitt soll ein kurzer Überblick über mögliche globale Grobregistrierungsverfahren geben werden (vgl. hierzu auch [TM06]).

Korrelation

Theoretisch kann man das Problem durch vollständige Suche im sechsdimensionalen Suchraum $(\alpha, \beta, \gamma, x, y, z)$ mit einem Aufwand von $O(n^6)$ lösen, d.h. das Funktional f durch Ausprobieren aller möglichen Korrelationen und Merken des besten Ergebnisses minimieren („Brute Force“). In der Praxis hat dieser Ansatz durch die große Datenmenge (und damit riesige Rechenzeit) keine Relevanz. Die natürliche Beschleunigung dieses Ansatzes ist der Iterative Closest Points - Algorithmus (siehe Abschnitt 2.5.2), der allerdings den Suchraum stark einschränkt und damit für die Grobregistrierung ungeeignet ist. Er wird in Kapitel 2.5 behandelt.

PCA

Das andere Extrem der Verfahren ist die Hauptkomponentenanalyse (**P**rincipal **C**omponent **A**nalysis). Zunächst werden hierbei die Schwerpunkte des Objekts und des Modells übereinander gelegt, um die drei translatorischen Freiheitsgrade zu bestimmen. Die rotatorische Freiheitsgrade ergeben sich aus den Trägheitsachsen, die als Eigenvektoren der Kovarianzmatrix C aller Punkte der Punktewolke schnell berechnet werden können (siehe hierfür Kapitel 3.4.2). Die Eigenvektoren, welche den Hauptachsen entsprechen, werden mit der numerisch stabilen und schnellen Singulärwertzerlegung *SVD* (**S**ingular **V**alue **D**ecomposition) berechnet. Das Objekt wird so gedreht, dass jede Hauptachse mit der jeweils korrespondierenden Hauptachse des Modells übereinstimmt. Diese Methode ist von allen vorgestellten Ansätzen am schnellsten zu berechnen und wurde als erster Ansatz implementiert (siehe Kapitel 3.4).

Hauptkomponenten Hough-Transformation

Basierend auf einer von Paul Hough in den 60er Jahren veröffentlichten Methode [Hou62] kann ein sechsdimensionaler „Hough-table“ aufgebaut werden (Für eine genaue Beschreibung einer Hough-Transformation siehe 2.4.2.1). In dem in [SS99] veröffentlichten Ansatz werden hierbei an jedem Punkt die Hauptkomponenten einer lokalen Umgebung gebildet. Jedes dieser Tupel wird mit jedem aus dem Modell verglichen und jeweils die sechs Transformationsparameter in ein globales Histogramm eingetragen. Maxima in jedem der Hough-Akkumulatorzellen deuten auf die global richtige Transformation hin (vgl. 2.4.2.1). Dieses Verfahren ist für unsere Anwendung zu rechenzeitintensiv.

Evolutionäre Algorithmen

Bei evolutionären Algorithmen wird versucht eine Suche so aufzubauen, dass sie die Natur in ihrer Methode der Mutation und Selektion nachahmt. Transformationen entsprechen in diesem Bild den Chromosomen. Diese werden zuerst modifiziert und rekombiniert, dann mittels einer Fitnessfunktion (eine Art Kostenfunktion im Datensatz) bewertet und schließlich selektiert [KB96]. Maier und Häussler [TM06] zu Folge werden diese Algorithmen zur Zeit nur als Feinregistrierungen eingesetzt, da sie sehr hohe Rechenzeiten benötigen.

Extended Gaussian Images

Diese von [Hor84] beschriebene Darstellung wurde für Bildgebungsverfahren entwickelt, die in jedem Punkt die Oberflächennormale liefern (siehe auch die Erläuterungen in seinem Buch [Hor86]). Ausgangspunkt ist eine Punktewolke, bei der zusätzlich an jedem Raumpunkt die Oberflächennormale n bekannt ist (sog. „Needle Diagram“). Das *Gaußbild* („Gaussian Image“) eines Objekts wird erzeugt, indem jedem Oberflächenpunkt p der Punkt der Einheitskugel (auch *Gaußsphäre*) q zugeordnet wird, der die selbe Orientierung hat. Man erhält also die Verteilung, welche Richtungen wie oft vorkommen (eine Art Richtungshistogramm). Diese Zuordnung ist eindeutig und für konvexe Objekte invertierbar, da jede Richtung bei konvexen Körpern nur einmal vorkommt. Speichert man zusätzlich an jedem Punkt der Gaußsphäre q das Inverse der *Gaußkrümmung* K des Objektes an p, so erhält man das *Extended Gaussian Image*

$$G(\chi, \eta) = \frac{1}{K(u, v)}$$

des Objekts, wobei χ und η den Punkt auf der Gaußsphäre beschreiben (z.B. Raumwinkel) und u, v den Punkt auf der Objektoberfläche.

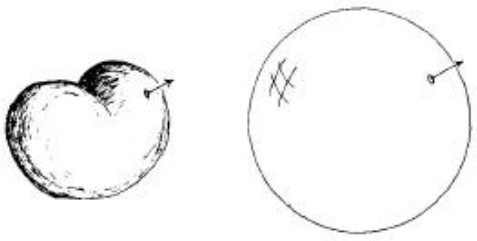


Abbildung 2.5: Das Gaußbild eines Objekts wird erzeugt, indem jedem Oberflächenpunkt der Punkt der Gaußsphäre zugeordnet wird, der die selbe Orientierung hat. (Bild aus [Hor84] entnommen)

Um ein Objekt zu erkennen, wird das erweiterte Gaußbild G des Objektes mit dem des Modells verglichen und die notwendige Transformation ermittelt, sie in einander zu überführen. Die Berechnung der Oberflächennormalen ist, wenn diese nicht direkt vom Bildgebungsverfahren geliefert werden, sehr rechenintensiv, ebenso wie die Berechnung der Gaußkrümmungen (2. Ableitungen). Ferner ist das Verfahren nur bei dichten Tiefenbildern anwendbar. Dennoch wird es in der Literatur weiterentwickelt [Dol05].

Exkurs: Gaußkrümmung

Sei N die Oberflächennormale eines Punktes p einer regulären, zweidimensionalen Fläche M im dreidimensionalen Raum. Der Schnitt von M entlang einer Ebene, die N enthält ergibt eine Kurve. Die Krümmung (2. Ableitung) einer solchen Kurve am Punkt p wird als eine *Schnittkrümmung* von M an p bezeichnet (siehe Abb. 2.6). Die maximale bzw. minimale Schnittkrümmung von M an p werden als erste bzw. zweite *Hauptkrümmung* κ_1, κ_2 von M an p bezeichnet. Das Produkt dieser beiden Hauptkrümmung heißt *Gaußkrümmung* $K = K(M, p)$. (vgl. z.B. [Bär01]).

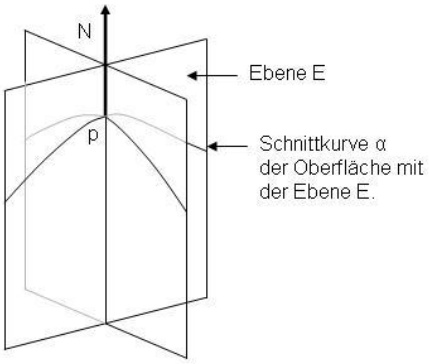


Abbildung 2.6: Schnittkrümmungen erhält man durch Schnitte der Oberfläche mit Ebenen E, die den Normalenvektor N in Punkt p enthalten. Sei α eine dadurch entstandene Kurve. Dann ist die zweite Ableitung von α eine Schnittkrümmung S

2.4.2 Lokale Verfahren

Globale Verfahren benötigen stets eine Segmentierung, d. h. das Objekt muss schon vorher aus dem Hintergrund herausgelöst werden. Die in diesem Abschnitt beschriebenen Verfahren, die *Merkmalregistrierungsverfahren*, können dagegen direkt auf das Bild angewandt werden. Eine eventuell vorangestellte Segmentierung dient dabei nur zur Beschleunigung. Die Verfahren beruhen auf einer Hough-Transformation, welche am Beispiel gerader Kanten im nächsten Abschnitt erläutert wird.

2.4.2.1 Hough-Transformation: Gerade

Betrachte ein Bild mit n Zeilen und n Spalten. Angenommen wir wollen ein durch gerade Kanten begrenztes Objekt darauf erkennen, z. B. ein Rechteck. Eine solche Kante kann durch die Geradengleichung

$$g(x) = sx + b$$

beschrieben werden, wobei $s \in \mathbb{R}$ die Steigung und $b \in \mathbb{R}$ der y-Achsenabschnitt ist. Trägt man nun für jeden Bildpunkt die Parameter aller Geraden, die durch ihn verlaufen, gewichtet mit seiner Helligkeit, in ein Parameterhistogramm (x-Achse entspricht beispielsweise den Steigungen und die y-Achse den y-Achsenabschnitten der Geraden) so erhält man eine Hough-Transformation des Bildes. Jeder Punkt im Originalbild ergibt hierin eine Gerade. Umgekehrt ergibt jede Gerade im Originalbild einen Punkt im Parameterhistogramm! Gibt es also im Originalbild deutliche Kanten, kann man diese als Cluster aus dem Hough-Diagramm herauslesen.

Machen wir eine kurze Aufwandsbetrachtung. Für jeden Punkt im Bild müssen alle Geraden berechnet werden, die durch diesen verlaufen. Sei k die Anzahl der Parameter, die berechnet werden müssen (bei einer Gerade gilt $k = 2$) und m die Diskretisierungsstufen im Hough-Diagramm. Dann muss für jeden Punkt im Bild, also n^2-mal, m^{k-1} Gleichungen berechnet werden ($k-1$, da der letzte Parameter dann festliegt). Das ergibt einen Aufwand von $\mathrm{O}(n^2 \cdot m^{k-1})$. Das heißt der Aufwand ist exponentiell mit der Anzahl der Parameter, die berechnet werden müssen.

Übertragen wir das Problem in unseren dreidimensionalen Raum. Angenommen wir wollen ein quaderförmiges Objekt finden. Wir suchen hierfür mit einer Hough-Transformation nach begrenzenden Ebenen. Eine Ebene im Raum wird durch drei Parameter beschrieben, zwei Winkel und den Abstand zum Ursprung. Nehmen wir weiter an, dass wir ein Megapixelbild ($n = 1000$) und eine Diskretisierung im Hough-Raum von $m = 100$ Schritten für jeden Parameter (z.B. Winkel und Abstand vom Ursprung bei Geraden-Hough-Transformation) haben. Damit hat die zugehörige Hough-Transformation den Aufwand

$$\mathrm{O}(n^2 \cdot m^{k-1}) = \mathrm{O}(n^2 \cdot m^2)$$

Um zu veranschaulichen, was es bedeutet, dass die Anzahl der Parameter k im Exponenten in den Aufwand eingeht, betrachten wir das Beispiel des entsprechende 2-Parameter-Problem (Geraden suchen) und nehmen an, es könne in 10ms berechnet werden. (In einer Implementierung der Bildverarbeitungsbibliothek „Halcon“ hat eine Hough-Transformation im zweidimensionalen Raum, die Geraden detektiert ($k = 2$) in einer Region mit ca. 60000 Messwerten 700ms benötigt. Typische Bauteile ergeben mit der verwendeten Sensorik oft nur weniger Messpunkte (minimal ca. 2000) und die Regionenauswahl könnte noch weiter optimiert werden. Trotzdem sind 10ms eine knapp bemessene Annahme). Dann würde man für das Suchen von Ebenen schon

$$100 \cdot 10\mathrm{ms} = 1 \text{ Sekunde}$$

benötigen. Die in Abschnitt 2.4.1 beschriebene Hough-Transformation hat sechs Parameter, wäre also um einen weiteren Faktor $m^3 = 1000000$ langsamer, was etwas mehr als 13 Tagen entspricht! Man benötigt also eine schnellere Methode.

2.4.2.2 Merkmalregistrierungsverfahren

Da Hough-Transformationen auf jedem Pixel gerechnet werden, sind sie gut parallelisierbar. Dies genügt jedoch noch nicht, um realistische Rechenzeitanforderungen zu erfüllen. Dafür müssen die Daten reduziert werden. Die Idee ist also, eine Transformation nicht mehr an jedem Punkt zu berechnen, sondern nur an ausgewählten Stellen. Dies führt zu den *Merkmalregistrierungsverfahren.*

Hierbei werden Merkmale (z.B. Spin Images, Splash Images, ...) des Objektes im gesamten Bildausschnitt (Hintergrund und Objekt) gesucht. Dabei werden die Segmentierung und die Grobregistrierung in einem Schritt erledigt. Die Hauptschwierigkeit besteht in der Korrespondenzsuche, also zu jedem gefundenen Merkmal im Bild das zugehörige Merkmal im Modell zu finden. Sei n die Anzahl der Merkmale, die in einem Erkennungsschritt gefunden wurde und m die Anzahl der im Modell gefundenen Merkmale. Schon in einfachen Modellen und Szenen sind m und n größer 500. Durch die Hochdimensionalität ist es nur in Spezialfällen möglich, schnellere Algorithmen als den „brute force“ Ansatz mit Aufwand $\mathbf{O}(n \cdot m)$ zu finden. Grundsätzlich hat das Problem die Dimension m und kann nur manchmal wesentlich reduziert werden, z.B. durch eine Hauptkomponentenanalyse, die nur noch

stark diskriminative Dimensionen zur Betrachtung übrig lässt. Eine übliche Rechenzeit für obiges Beispiel auf einem Pentium DualCore beträgt 30 Sekunden. Dies kann durch Grafikkartenprogrammierung oder geschickte Datenverwaltung (Bäume) verbessert werden. Durch die geringere Datenmenge sind diese Verfahren dennoch viel schneller als reine Hough-Transformation und kommen deshalb vielfach zur Anwendung.

In den folgenden Abschnitten sollen mögliche 3D-Merkmale beleuchtet werden. Einen guten Überblick über das Thema gibt der Artikel von Campbell und Flynn [RJC01] im Abschnitt „Free-Form Object Recognition Systems“. Eine Bewertung der Merkmale für unsere Daten wird in Abschnitt 2.4.3 durchgeführt.

Spin Images

Zunächst wird die Oberfläche rekonstruiert und Normalenvektoren auf ihr berechnet. Wie berechnet man das Spin Image in Punkt p mit Oberflächennormalenvektor N? Betrachte hierfür eine Ebene, die N enthält. Die Ebene wird in Boxen (sog. ”bins”) unterteilt. Rotiert man sie nun um den Normalenvektor und summiert über Oberflächenpunkte, die bei der Rotation die Ebene in der jeweiligen Box schneiden, erhält man das Spinimage [AEJ97], wie in Abbildung 2.7 dargestellt. Mit rot sind dabei die Bins markiert, deren Zähler um eins erhöht würde. Das SpinImage ist eine Matrix mit ganzen Zahlen, deren Spaltensumme k in jeder Spalte gerade den Diskretisierungsschritten der Rotationsausrichtungen entspricht. Das Spin Image einer Ebene hätte beispielsweise nur in der mittleren Zeile Einträge, die alle gleich k wären. Es wird also ein Histogramm über die lokale Krümmung an einem Punkt gebildet. Die Richtungsauflösung der Krümmung geht dabei verloren. (Es gibt jedoch Ansätze diese zu erhalten, indem man sich Histogramme immer über Teilrotationen merkt.)

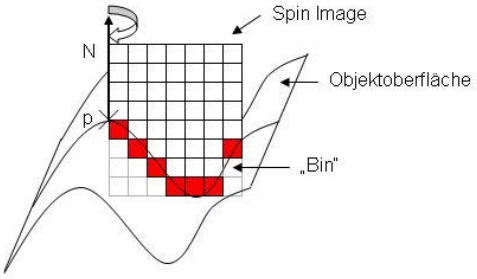


Abbildung 2.7: Spin Image: Eine Ebene mit „Bins“ wird um den Normalenvektor N an Punkt p rotiert. Mit rot sind die Bins markiert, die für diesen Winkelbereich gespeichert werden.

Splash Images

Um ein Splash Image an einem Punkt p mit Oberflächennormale N zu berechnen (vgl. Abbildung 2.8), werden die Normalen in einem bestimmten Abstand r im Raum von N betrachtet. Seien $N_1 \dots N_k$ die Normalen im Abstand r und $\alpha(\Theta)$ die Kurve, die im Abstand r von p auf der Oberfläche verläuft. Trägt man den Winkel Φ zwischen N und N_i für $i = 1 \dots k$ über i auf, so erhält man eine periodische Kurve (diskretisiert einen Vektor). Das Splash Image S_{Splash} ist also ein Vektor mit Winkeln:

$$S_{\text{Splash}} = (\Phi_1, \Phi_2, \dots, \Phi_k)^T \qquad \text{mit } \Phi_i = \angle(N, N_i) \text{ und } N_i = N_{\alpha(\Theta)} \forall i$$

Zusätzlich kann die Richtung der Verkippung betrachtet werden. Wählt man als Startpunkt der Kurve α einen ausgezeichneten Punkt, z.B. den Punkt mit maximalem Φ, erreicht man eine Rotationsinvarianz des Splash Images. [FS92]

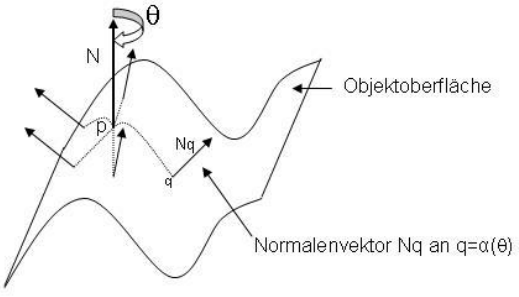


Abbildung 2.8: Splash Image: Die Winkel zwischen den Normalen $N_{\alpha(\Theta)} = N_i$ an $q = \alpha(\Theta)$ nahe p und der Normale N an p werden gespeichert.

Point Signature

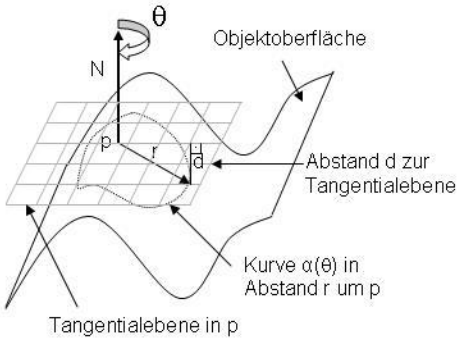
Sei $\alpha(\Theta)$ wiederum die Kurve auf der Oberfläche im Abstand r von p und E die Tangentialebene an die Oberfläche in p (das ist die Ebene, die senkrecht auf dem Normalenvektor N in p steht). Im Gegensatz zu Splash Images wird für die Point Signature eines Punktes p nicht die Verkippung der Normalen berechnet, sondern der Abstand umgebender Punkte zur Tangentialebene in p (vgl. Abbildung 2.9(a)). Das Merkmal selbst ist dann eine Kurve, die den Abstand d in Abhängigkeit vom Winkel Θ enthält. Der Startpunkt $\Theta = 0$ ist dabei durch beispielsweise den höchsten Wert für d festgelegt.

2.4.3 Bewertung der Grobregistrierungsverfahren

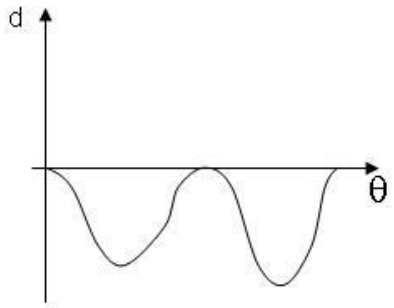
Die Grobregistrierungsverfahren sind in Tabelle 2.1 zusammengefasst und für unsere Anwendung bewertet:

- Geschwindigkeit: In dieser Kategorie wird die benötigte Rechenzeit bei der Objektlageerkennung bewertet.
- Genauigkeit: Damit soll die erwartete Fitgenauigkeit bewertet werden.
- Robustheit gegen globale Rekonstruktionslücken: Ein Beispiel hierfür sind Glanzpunkte. Das ist ein Maß dafür, welchen Einfluss es hat, wenn ein Objekt nicht vollständig rekonstruiert wird.
- Robustheit gegen lokale Rekonstruktionslücken: Ein Beispiel hierfür sind die Lücken, die dadurch entstehen, dass Tiefenwerte nur auf den Streifen zur Verfügung stehen. Es wird bewertet, ob der Algorithmus in kleinen Bereichen eine dichte Rekonstruktion benötigt.
- Robustheit gegen Rauschen: Hier wird bewertet, wie empfindlich der Algorithmus auf verrauschte Messwerte reagiert.

Als Bewertung wurden Punkte von 0 (schlechteste) bis 5 (beste) in jeder Kategorie vergeben. Eine Null bedeutet hierbei, dass das Verfahren für die von uns verwendete Sensorik



(a) Abbildung der Berechnung des Merkmals



(b) die Kurve, die das Merkmal darstellt

Abbildung 2.9: Point Signature: Die Kurve $\alpha(\Theta)$ verläuft auf der Oberfläche im Abstand r von p. Das Merkmal besteht aus dem Abstand d des Punktes $\alpha(\Theta)$ zur Tangentialebene an die Oberfläche in p (Ebene senkrecht zu dem Normalenvektor N in p)

Verfahren	**lokal**	**global**	**Geschwindigkeit**	**Genauigkeit**	**Robustheit (globale Rekonstr.-Lücken)**	**Robustheit (lokale Rekonstr.-Lücken)**	**Robustheit (Rauschen)**	**Summe**
vollständige Korrelation		x	0	5	5	5	4	-
PCA der Punktewolke		x	5	3	1	5	4	18
PCA Hough-Transformation		x	0	4	4	2	4	-
evolutionäre Algorithmen		x	0					-
extended Gaussian Images		x	2	4	3	2	2	13
vollst. 6D-Hough-Transformation	x		0	5	4	4	4	-
Spin Images	x		1	4	5	1	2	13
Splash Images	x		1	4	5	1	1	12
Point Signature	x		2	4	5	1	1	13

Tabelle 2.1: Bewertungsmatrix der vorgestellten Grobregistrierungsverfahren. Es werden Punkte von 0 (schlechteste) bis 5 (beste) vergeben

ausgeschlossen werden muss (bzw. wesentlich modifiziert werden müsste, um anwendbar zu sein). Die Werte beruhen dabei auf Schätzungen, da der Implementierungsaufwand der Verfahren zu hoch ist und in der Literatur keine aussagekräftigen Vergleiche zu finden waren (für die Einschätzung der Einzelverfahren siehe Quellen bei den jeweiligen Beschreibungen).

In der Kategorie Genauigkeit wurden 5 Punkte für die Verfahren gegeben, die das globale Minimum aller möglichen Zuordnungen der Messdaten zum Modell, also das globale Minimum des Funktionals f aus Abschnitt 2.3, ermitteln.

Die lokalen Verfahren (Spin Images, Splash Images und Point Signature) sind anfällig für lokale Änderungen der Daten und deshalb in den Kategorien „Robustheit gegen lokale Rekonstruktionslücken“ und „Robustheit gegen Rauschen“ sehr niedrig bewertet. Will man beispielsweise die Point Signature auf Daten mit Rekonstruktionslücken, wie unseren Streifenprojektionsdaten, an einem Punkt ermitteln, so hängt das Ergebnis von der genauen Lage der Streifen in der Aufnahme ab (Abtastproblem, vgl. Abbildung 2.10). Ähnliches gilt für andere lokale Merkmale. Dafür sind die lokalen Verfahren robust gegen globale Rekonstruktionslücken, da nicht über die gesamte Punktewolke gemittelt wird.

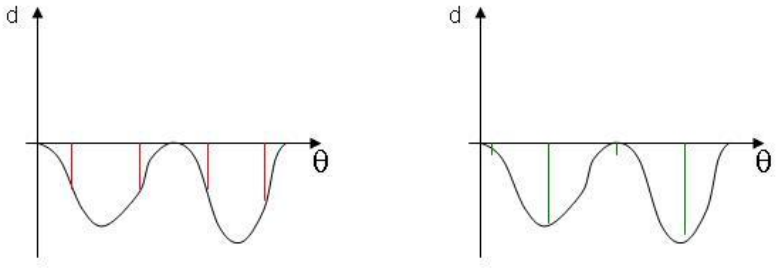


Abbildung 2.10: Die richtige Point Signature eines Punktes wird unterschiedlich abgetastet (links rot, rechts grün). Die linke Abtastung würde ein falsches Ergebnis liefern, da das *Nyquist Abtasttheorem* verletzt ist.

In Kapitel 3.4 ist die Umsetzung der Hauptachsentransformationslageschätzung (PCA) beschrieben, welche für unseren Algorithmus als Grobschätzung ausreichte.

2.5 Feinlageerkennung

Ausgehend von einem guten Startwert, der in einer Groblageerkennung ermittelt wurde, und einer segmentierten Punktewolke, sollen in diesem Abschnitt Verfahren zur Lösung des Funktionals f, das im ersten Abschnitt 2.5.1 entwickelt wird, vorgestellt werden.

Der erste Ansatz für diese *Feinlageerkennung* (Feinregistrierung) ist die direkte mathematische Minimierung. Rabbani und van der Heuval [TR04] schlagen hierfür das Levenberg-Marquardt-Verfahren vor, ein Standardverfahren für nichtlineare, quadratische Optimierung. Die dabei benötigten partiellen Ableitungen werden durch finite Differenzen approximiert und die benötigten Rotationen mittels Quaternionen berechnet. Dies ergibt ein sogar global konvergentes Verfahren, das in jedem Schritt einen Abstieg garantiert. Der

große Nachteil ist die sehr zeitintensive Berechnung der benötigten Jacobi-Matrix in jedem Schritt.

Hier setzt der in der Literatur sehr häufig genannte **I**terative **C**losest **P**oints Algorithmus (kurz *ICP*) an. Dieser geht auf Besl und McKay [PJB92] zurück und wurde für das Einpassen einer Punktewolke in eine Referenzpunktewolke entwickelt. Dabei wird für jeden Punkt derjenige Punkt der Referenzpunktewolke berechnet, der den geringsten Abstand hat. Dann wird für diese Punktepaare diejenige Rotation R und derjenige Translationsvektor t berechnet, für die das Funktional f minimal wird. Die exakte Formulierung wird im Abschnitt 2.5.2 vorgestellt.

2.5.1 Formulierung des Optimierungsproblems

Betrachten wir den Modellbegriff, wie er in der differentialgeometrischen Formulierung einer Triangulierung in Abschnitt 2.7.2.1 entwickelt wird. Dann wird eine Projektion wie folgt definiert:

Definition 2.1 *Sei M ein Modell und $p \in \mathbb{R}^3$ ein Punkt. Seien $\Delta_1, \ldots, \Delta_m$ die Dreiecksflächen der zugehörigen Dreiecke von M. Dann ist die **Projektion** Π von p auf M gegeben durch:*

$$\Pi_M(p) = q \in \Delta_k \qquad k \in \{1, \ldots, m\}$$

so dass

$$d = \|p - q\|_2^2 = (p_1 - q_1)^2 + (p_2 - q_2)^2 + (p_3 - q_3)^2$$

minimal ist.

Die Projektion ist so nicht wohldefiniert. Es könnte mehrere Punkte q geben, die zu p minimalen Abstand haben. Wähle in diesem Fall den Punkt q mit dem geringsten Index.

Definition 2.2 *Sei $M = (T, S)$ ein Modell, $R \in SO(3)$ eine orthogonale Matrix mit $\det R = 1$ und t ein Translationsvektor. Dann bezeichne die Funktion*

$$T_{R,t}(M) = (Rp + t, Rs + t) \quad \forall p \in T,\ \forall s \in S$$

*als **Transformation des Modells**.*

Die Einschränkung $\det R = 1$, so dass $R \in \mathrm{SO}(3)$ (also eine *Rotationsmatrix*) und nicht aus $\mathrm{O}(3)$ (*Drehspiegelungen*) sein kann, wird gewählt um nur Drehungen und keine Spiegelungen zuzulassen. Anderenfalls könnte eine Spiegelung des Modells matchen, die jedoch „unphysikalisch“ wäre.
Mit den obigen Begriffen kann nun unser Problem formuliert werden.

Satz 2.3 *Sei M ein Modell und $\Delta_1, \ldots, \Delta_m$ die Dreiecksflächen der zugehörigen Dreiecke $D_1, \ldots, D_m$, sowie $P = \{p_1, p_2, \ldots, p_n\}$ die Punktewolke eines Objekts. Dann muss für einen **Match**, also für die gesuchte optimale Zuordnung $f_{opt}(R, t)$ folgendes Optimierungsproblem gelöst werden:*

$$f_{opt}(R,t) := \min_{R \in SO(3), t \in \mathbb{R}^3} f(R,t) = \min_{R \in SO(3), t \in \mathbb{R}^3} \sum_{p \in P} \|\Pi_M(T_{R,t}(p)) - p\|_2^2$$

wobei $R \in SO(3)$ *eine orthogonale Matrix mit* $\det R = 1$ *und* $t \in \mathbb{R}^3$ *ein Translationsvektor ist.*

Man beachte, dass diese Formulierung auch für Modelle, die Merkmale enthalten, anwendbar ist. Wähle hierfür $p \in S$ und als Projektion $\Pi(p)$ die Zuordnung, die jedem Merkmalbasispunkt aus dem Bild den entsprechenden Merkmalbasispunkt des Modells zuordnet.

2.5.2 Iterative Closest Points Algorithmus

Betrachte zwei Punktewolken $A, B \subseteq R^d$ der Größe $|A| = n$ und $|B| = m$. Gesucht ist eine Zuordnungsfunktion $\mu : A \to B$, die die Abstandsquadrate

$$\tilde{L}(A, B, \mu) = \sqrt{\frac{1}{n} \sum_{a \in A} \|a - \mu(a)\|^2}$$

minimiert. Mit $R \in SO(d)$ der Rotationsmatrix in $d = 3$ Dimensionen und dem Translationsvektor t ist dies für die Minimierung gleichbedeutend mit

$$L(A, B, \mu) = \sum_{a \in A} \|Ra - t - \mu(a)\|^2$$

und es ist

$$\min_{\mu: A \to B, t \in R^d, R \in SO(d)} \mu = L(A, B, \mu)$$

gesucht.

Die Idee des Ansatzes ist, dass man nicht gleichzeitig die Zuordnungsfunktion μ und die Transformation definiert durch R und t minimiert, sondern nacheinander!

1. Initialisierung: R Rotation der Groblageschätzung, t Translation der Groblageschätzung
2. Zuordnungsschritt: Bei festem R und t wird die optimale Zuordnung μ mittels Minimierung $\min_\mu L(A, B, \mu)$ gesucht.
3. Transformationsschritt: Bei festem μ werden die Rotation R und die Translation t optimiert um $\min_{R,t} L(A, B, \mu)$ zu finden.
4. wiederhole ab Schritt 2 bis sich μ nicht mehr ändert.

Es gibt unterschiedliche Varianten des ICP, die sich in der genauen Ausführung des Zuordnungs- und Transformationsschrittes unterscheiden. Der Artikel von Rusinkiewicz und Levoy [SR01] gibt einen guten Überblick über die entsprechende Literatur.

Im Zuordnungsschritt wird zu jedem $a \in A$ das nächste $b \in B$ gesucht. Die brute-force-Methode zur Berechnung hat einen Aufwand $O(n \cdot m)$ und ist damit bei typischen Punktwolkengrößen von mehreren hunderttausend Punkten zu langsam. In der Literatur werden viele Methoden beschrieben, um dieses sogenannte *nächste Nachbarn* Problem zu lösen, da es der bei weitem zeitaufwändigste Teil des Algorithmus ist. Meist wird ein Baum aufgebaut, was zwar Rechenaufwand bedeutet ($O(m \log m)$), die spätere Suche pro Punkt aber

wesentlich beschleunigt ($O(\log m)$). Übliche Varianten hierfür sind der k-d tree und der sehr einfache Octree (vgl. Diplomarbeit [Tra09]). Zusätzlich gibt es mehrere Varianten die Daten zu reduzieren und z.B. nicht Punktkorrespondenzen sondern Korrespondenzen, von wenigen aus ihnen berechneten Merkmalen, etwa Normalenvektoren zu suchen. Bei ungeordneten Punktewolken ergibt sich noch eine weitere Schwierigkeit. Man will eigentlich nicht Punkte $a \in A$ anderen *Punkten* $b \in B$ zuordnen, sondern viel mehr der *Oberfläche*, auf der die Punkte aus B liegen. Das führt zum Problem der Oberflächenrekonstruktion und Triangulierung unstrukturierter Punktewolken, welches gut in der Doktorarbeit [Wil00] und darauf aufbauend in der Diplomarbeit [Kie09] behandelt wird.

Im Transformationsschritt werden die Rotationsmatrix R und der Transformationsvektor t gesucht, die die Summe

$$L(R,t) = \sum_{i=1}^{n} \|Ra_i - t - b_i\|^2$$

mit $\mu(a_i) = b_i$, $a_i \in A$ und $b_i \in B$ minimieren. Seien $\bar{a}$ und $\bar{b}$ die Mittelwerte von A bzw. B und $a'_i = a_i - \bar{a}$ sowie $b'_i = b_i - \bar{b}$, dann ist die Summe äquivalent zu

$$L(R,t) = \sum_{i=1}^{n} \left\|R(a'_i + \bar{a}) - t - (b'_i + \bar{b})\right\|^2 = \sum_{i=1}^{n} \left\|Ra'_i - b'_i + (R\bar{a} - \bar{b} - t)\right\|^2$$

und man kann die Berechnung der beiden Variablen mit $t = R\bar{a} - \bar{b}$ und

$$L(R) = \sum_{i=1}^{n} \left\|Ra'_i - b'_i\right\|^2$$

aufspalten. Weiter lässt sich $L(R)$ mit der Spur tr umformen

$$L(R) = \underbrace{RR^T}_{=I} \sum_{i=1}^{n} \left\|a'_i\right\|^2 - 2\mathrm{tr}(\sum_{i=1}^{n} a'_i b'^T_i) + \sum_{i=1}^{n} \left\|b'_i\right\|^2$$

Damit wurde das Problem auf die Maximierung der Spur

$$\mathrm{tr}(RN)$$

mit $N = \sum_{i=1}^{n} a'_i b'^T_i$, reduziert. Betrachte die Singulärwertzerlegung

$$N = U\Sigma V^T$$

wobei U und V orthogonale Matrizen und $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d)$, so dass $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d \geq 0$ sind (Eigenwerte). Nach [KSA87] nimmt $L(R,t)$ ihr gesuchtes Maximum für $R = VU^T$ an.
Der Rechenaufwand für diesen Schritt ist mit $O(n)$ sehr niedrig.

Ein Problem des ICP sind nahezu symmetrische Objekte. Wie man sich leicht klar machen kann, konvergiert der ICP-Algorithmus stets gegen das zur Ausgangslage nächste lokale Minimum. Vor allem bei einem nahezu symmetrischen Objekt muss dieses aber nicht das globale Minimum sein. Um also ein falsches Ergebnis zu vermeiden, muss entweder vor Start des Algorithmus garantiert werden, dass das nächste lokale Minimum auch das globale Minimum ist, oder der ICP muss einen globalen Korrekturterm haben, der aber auf Kosten der Konvergenzgeschwindigkeit geht. Um erste Bedingung zu erfüllen werden Grobregistrierungsverfahren eingesetzt.

Eine Erweiterung des ICP, um seine Robustheit gegenüber einer Variation des Startwerts zu verbessern, stellt der sogenannte *Expectation Maximization ICP* kurz *EM-ICP* dar. Granger und Pennec [SG02] gehen in diesem Ansatz davon aus, dass die Punkte aus A gegenüber den Korrespondenzpunkten in B gemäß einer Normalverteilung $N(\mu, \Sigma)$ verrauscht sind. Die Registrierung der beiden Punktewolken wird nun als Likelihood-Schätzung bezüglich der Transformation mit zusätzlichen Matching-Gewichten formuliert, die der Wahrscheinlichkeit der Korrespondenz zweier Punkte entsprechen. Im Grenzfall ergibt dies den oben beschriebenen ICP.

2.6 Kalibrierung

In diesem Abschnitt soll der Stand der Technik zur Kalibrierung von Kameras und Projektoren angerissen werden. Das Gebiet ist sehr groß und Gegenstand aktueller Forschung, daher hat dieser Abschnitt nur den Anspruch, später verwendete Basiskonzepte zu erläutern.

2.6.1 Kalibrierung der intrinsischen Kameraparameter

Um mit einer Kamera Objektlagen in einer Szene quantitativ auswerten zu können, wird ein *geometrisches Kameramodell* zu Grunde gelegt. Im Wesentlichen soll dieses aus dem Ort im Bild auf einen Ort in der Szene rückschließen können (*Rückwärtsmodell*). Ein System aus nur einer Kamera ist hierbei unterbestimmt (in der Regel fehlt der Abstand eines Objektpunktes von der Kamera). Das inverse Problem, nämlich aus einem Szenepunkt den Ort im Kamerabild zu berechnen (*Vorwärtsmodell*) ist dagegen eindeutig lösbar.

Ein idealisiertes geometrisches Kameramodell ist das *Lochkameramodell*. Hierbei wird die Szene durch ein Loch betrachtet, hinter dem ein auf dem Kopf stehendes Bild entsteht. Je kleiner hierbei das Loch ist, desto schärfer wird das Bild. In realen Kameras wird dieses Loch durch ein Linsensystem ersetzt, dessen Brennweite f im Wesentlichen die gesuchte Transformation von Szenepunkt (x, y, z) zu Bildpunkt (u, v) bestimmt. Es gilt:

$$u = -f\frac{x}{z}$$

$$v = -f\frac{y}{z}$$

wobei z parallel zur optischen Achse steht.

Reale Linsensysteme können eine *Scherung* zwischen der Brennweite in x-Richtung und der Brennweite in y-Richtung aufweisen. Ferner haben sie eine *Verzeichnung*, d.h. die Vergrößerung des Bildes des Linsensystems ändert sich lokal. Um die Szene vermessen zu können, muss diese Verzeichnung bekannt sein und herausgerechnet werden. Man nennt die Gesamtheit aller dieser Parameter, die zwar von der Kamera, nicht jedoch von der betrachteten Szene abhängen *intrinsische Kameraparameter*.

Zur Bestimmung dieser wird üblicherweise ein genau vermessenes Kalibrierobjekt in unterschiedlichen Posen von der Kamera aufgenommen. Das Objekt wird im Bild erkannt und somit die gesuchte Verzerrungsfunktion, sowie Brennweite des Linsensystems durch ein Bündelausgleich bestimmt.

2.6.2 Kalibrierung der extrinsischen Kameraparameter

Hierbei muss ein Rückwärtsmodell gelöst werden, d.h. aus den Kamerabildern auf die Lage eines Objekts in der Szene schließen. Um mit dem im Rahmen dieser Arbeit verwendeten Stereokamerasystem auf den fehlenden Parameter - den Abstand eines Objektpunktes von

der Kamera - schließen zu können, müssen weitere Paramter bekannt sein - die sogenannten *extrinsischen Kameraparameter*. Hierbei handelt es sich um die genaue Position der Kameras zueinander. Das von den Kameras aufgenommene Bild wird dabei so verzerrt, dass es wie ein Bild im Stereonormalfall erscheint. Diesen Vorgang nennt man *Rektifizierung*. Hierbei muss nicht nur der Abstand der Kameras zueinander, sondern auch eine mögliche Verschiebung der Bildzeilen von linker zu rechter Kamera und eine mögliche Verdrehung oder Verkippung der jeweiligen Bildebenen ausgeglichen werden.

Die übliche Methode ist, wie bei der Bestimmung der intrinsischen Kameraparameter, das Vermessen eines bekannten Objektes in verschiedenen Posen mit anschließendem Bündelausgleich zur Schätzung der Parameter.

2.6.3 Projektorkalibrierung

Die Kalibrierung von Projektor zu Kamera kann ähnlich durchgeführt werden wie Stereokalibrierungen. Zunächst werden die intrinsischen Kameraparameter gemäß dem in Abschnitt 2.6.1 beschriebenen Verfahren geschätzt. Das nächste Ziel ist eine Zuordnung, die aus der Position des Musterstücks (Labels) im Kamerabild auf den Abstand von der Kamera zum Objekt schließt. Ähnlich wie bei der Stereokalibrierung auch, wird hierfür ein (Projektor-)Modell hinterlegt, dessen Parameter aus zugeordneten, markanten Punkten berechnet werden können. Die verwendeten markanten Punkte variieren mit den beschriebenen Verfahren.

Forster [For05] verwendet eine Kalibrierplatte mit exakt vermessenen Quadraten. Mit dem Projektor wird ein ähnliches Muster in einer anderen Farbe projiziert. Durch Lokation auf der Kalibrierplatte und durch die zuvor durchgeführte Kamerakalibrierung, können die Positionen der Ecken des projizierten Musters im Raum bestimmt werden. Mehrere solche markanten Punkte werden in mehrern Bildern bestimmt und daraus die Transformationsmatrizen $T_i : B \subset \mathbb{R}^2 \rightarrow \mathbb{R}^3$, die ein an einer Stelle im Bild B gefundenes Label i auf einen Raumpunkt abbilden, berechnet.

Bei dem Verfahren von Xu et al [ZX07] wird ein Schachbrettmuster genau in die x-y-Ebene der zuvor kalibrierten Kamera projiziert.

Bei beiden Verfahren ist die Projektion eines besonderen Kalibrierungsmusters notwendig, allerdings benötigen sie außer der Kamera und dem Projektor kein weiteres Sensorsystem.

Ein Verfahren für ein Streifenmuster, ohne das Muster wechseln zu müssen, wird von Matabosch [JS06] beschrieben. Hierbei werden als markante Punkte die Enden der Streifen benutzt. Dies ist möglich, da nicht das ganze Bildfeld mit dem Streifenmuster überdeckt wird. Auf Grundlage dieser Streifenenden werden zur Verbesserung der Genauigkeit noch weitere markante Punkte gewählt. Die restlichen Schritte werden analog den zuvor beschriebenen Verfahren durchgeführt.

2.7 Strukturierung ungeordneter Punktewolken

Ziel der Arbeit ist die möglichst exakte Objektlageerkennung aus Tiefenbildern. Wie oben beschrieben, wird hierfür in die aktuelle Aufnahme ein Modell eingepasst, d.h. die gesuchte Objektlage ist diejenige Starrkörpertransformation $T : M \rightarrow P$ mit Modell M und Objekt P aus der zu vermessenden Ansicht, die den Abstand zwischen Modell und Objekt(-Punktewolke) minimiert. Das benötigte Modell soll mit der gleichen Sensorik generiert werden. In diesem Kapitel geht es um die Erzeugung und Handhabung von Punktewolken. Die Umsetzung in dieser Arbeit ist in Kapitel 3.7 beschrieben.

Wird das Modell eines Objekts aus Aufnahmen des Objekts aus unterschiedlichen Ansichten erzeugt, so kann die Genauigkeit im Vergleich zu einer einzelnen Objektansicht

erhöht werden. Die entstandenen Tiefenbilder werden anschließend in 3D-Punktewolken transformiert und zu einem Oberflächenmodell zusammengefügt. Um diese Schritte effizient durchführen zu können, benötigt man eine geeignete Datenstruktur für ungeordnete Punktewolken (2.7.1), sowie deren Verarbeitung zu einem Oberflächenmodell (2.7.2).

2.7.1 Effiziente Datenspeicherung - Bäume

Eine Datenstruktur für Punktewolken muss so angelegt sein, dass man schnell auf Punkte zugreifen und deren räumliche Nachbarn ermitteln kann. Hierfür eignen sich hierarchische Strukturen, insbesondere Baumstrukturen. [Roh11], [Tra09]

Grundbegriffe:

Ein ***Graph*** G ist eine Menge von Punkten (*Ecken* oder *Knoten*), die mit Linien (sog. *Kanten* oder *Bögen*) verbunden sind. Er heißt ***zusammenhängend***, wenn jeder Knoten von jedem anderen Knoten über einen Weg erreichbar ist. Ein Graph heißt ***gerichtet***, wenn die Kanten eine Richtung haben. Ein gerichteter Graph heißt zusammenhängend, wenn der zugehörige ungerichtete Graph zusammenhängend ist. Falls G nicht zusammenhängend ist heißt er ***unzusammenhängend***. Ferner heißt G ***zyklenfrei***, wenn zu je zwei Punkten genau ein solcher Weg existiert. Ein gerichteter Graph heißt zyklenfrei, wenn der zugehörige ungerichtete Graph zyklenfrei ist.

Ein ***Baum*** ist ein zusammenhängender, zyklenfreier Graph.

Ein ***Wurzelbaum*** ist ein *gerichteter* Baum. D.h. die Pfade der Zuordnung haben eine Richtung. Dadurch ist einer der Knoten als *Wurzel* ausgezeichnet, nämlich derjenige Knoten, in den entweder alle Pfade münden (*In-Tree*), oder aus dem alle Pfade entspringen (*Out-Tree*).

Balanciert heißt ein Baum, wenn er auf minimale Tiefe optimiert ist und

binär, wenn jeder Knoten höchstens zwei Kindknoten hat, d.h. die Vergleichsoperation, die entscheidet in welchem der Äste der gesuchte Punkt liegt, muss eine *totale Quasiordnung* bilden (d.h. die zugehörige zweistellige Relation auf der Punktmenge muss *reflexiv* und *transitiv* sein, sowie je zwei Elemente der Punktmenge müssen vergleichbar sein (*total*)).

Eine ***Bereichsanfrage*** ist die Antwort auf die Frage, welche Punkte in der Kugel $B_r(p)$ mit Radius r und Mittelpunkt p liegen. Diese kann die Leere Menge sein.

***kNN-Anfragen*:** Gesucht sind die k nächsten Nachbarn („**k N**earest **N**eighbours“) eines Punktes p (des Baumes). Ist $k = 1$ spricht man auch einfach von einer *NN-Anfrage*.

Bei Bäumen sind die Laufzeiten für solcke kNN-Anfragen deutlich kürzer, als in ungeordneten Punktewolken, nämlich $O(n) = k \cdot \log_2(n)$ bei einem balancierten Binärbaum (der dann eine Tiefe von $\log_2(n)$ hat) statt $O(n) = k \cdot n$ für den ausschöpfenden Vergleich.

Octree

Die Idee bei allen Bäumen ist, die Daten in möglichst gleiche Portionen zu teilen, um bei einer Suche jeweils nur noch eine dieser Portionen durchsuchen zu müssen. Man unterscheidet zwischen *datenbasierten* und *raumbasierten* Strukturen. Erstere trennen die Datenmenge möglichst gleichmäßig, zweitere den Raum, in dem die Daten liegen. Nur datenbasierte Bäume können i.A. balanciert werden, was bei ungleichmäßig verteilten Daten zu schnelleren Zugriffszeiten führt. Bei raumbasierten Bäumen ist dagegen die Entscheidungslogik, in welchem der Teilbäume weitergesucht werden muss, oft einfacher.

Teilt man einen dreidimensionalen Datenraum Quaderförmig in acht gleiche Teile, wie in Abbildung 2.11 zweidimensional aufgezeigt, so erhält man den sogenannten *Octree*. Als

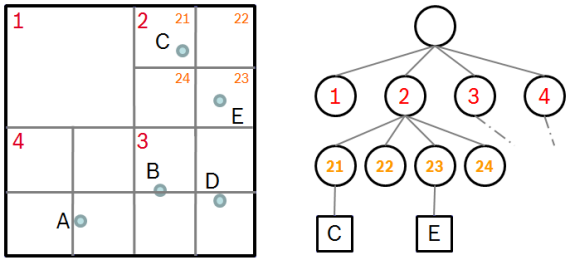


Abbildung 2.11: Aufbau eines Quadtrees als beispiel einer Raumteilenden Datenstruktur. Abb. aus [Roh11]

Ordnungsrelation des so entstandenen raumbasierten Baumes wird das „$\leq$“ der reellen Zahlen in den drei Koordinatenachsen gewählt (ergibt $2^3 = 8$ Kindknoten pro Knoten). Die Punkte werden in den Blättern gespeichert. Als Datenraum wird üblicherweise die Boundingbox der Daten gewählt und Hierarchieebenen, bei denen alle Punkte in einer Box sind, werden übersprungen. Man kann die Tiefe des Baumes beschränken, was jedoch dazu führen kann, dass viele Punkte in einem Blatt sind (schlechtestenfalls alle). Wenn man die Tiefe unbeschränkt lässt, kann sie der Anzahl der Punkte entsprechen (Beweis siehe A.1). Damit hat der Baum bei ideal verteilten Daten eine logarithmische Zugriffszeit ($O(n) = \log_2(n)$ bei einer NN-Anfrage), schlechtestenfalls jedoch eine lineare ($O(n) = n$ bei einer NN-Anfrage), nämlich wenn alle Punkte in einem Blatt sind und somit alle einzeln geprüft werden müssen, oder die Tiefe n beträgt.

kd-Tree

Teilt man einen k-dimensionalen Suchraum nicht in räumlich gleich große Unterräume, sondern wählt Split(-hyper-)ebenen jeweils abwechselnd in jeder Koordinate und im *Median* der enthaltenen Punktemenge bezüglich der jeweiligen Koordinate, so erhält man einen datenbasierten, balancierten Binärbaum, den sogenannten kd-Tree. Für ein Beispiel eines 2d-Trees siehe Abbildung 2.12 aus [Roh11].

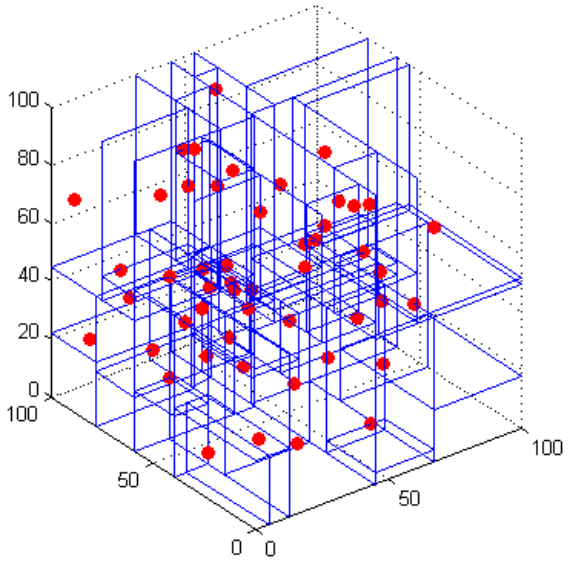


Abbildung 2.12: Beispiel eines kd-Trees mit $k = 3$, aus [Roh11]

Als Variation kann der Aufbau des Baumes schon abgebrochen werden, wenn nur noch eine feste Anzahl Punkte $m > 1$ im jeweiligen Blatt enthalten ist. Außerdem kann nicht abwechselnd entlang der Koordinaten geteilt, sondern es kann jeweils die Koordinate mit

der größten Streuung gewählt werden. Wählt man ferner statt dem Median den räumlichen Mittelpunkt der jeweiligen Zelle, erhält man eine ähnliche Unterteilung wie beim Octree (also einen „binären Octree“), verliert jedoch die Balancierung. Ein Mittelweg ist der sogenannte „Sliding Midpoint Split“. Hierbei wird der räumliche Mittelpunkt der Zelle gewählt, außer eine der beiden Zellen wäre leer. Dann wird die Trennebene in den nächsten Punkt der Datenmenge in der Zelle gelegt.

2.7.2 Oberflächenrekonstruktion durch Triangulierung

Die in dieser Arbeit betrachtete Objektlageerkennung basiert auf keinen echten dreidimensionalen Daten, sondern es können nur Punkte auf der *Oberfläche* dreidimensionaler Objekte gemessen werden. Damit ist die natürliche Struktur unserer Punktewolken flächenhaft. Die im letzten Abschnitt betrachteten Datenstrukturen ermöglichen zwar schnell nahegelegene Punkte zu finden, beinhalten diesen Flächencharakter jedoch nicht. In diesem Kapitel sollen nun Grundlagen für die Triangulierung von Oberflächen erklärt werden.

2.7.2.1 Differentialgeometrische Formulierung

In diesem Abschnitt wird die Feinregistrierung mathematisch exakt gefasst. Viele der Definitionen sind der Fassung der Aufgabe von [Bor] entnommen, beginnend mit der grundlegenden Definition einer Punktewolke.

Definition 2.4 *Sei $p \in \mathbb{R}^3$, dann nennt man p **Punkt** und eine Menge von n Punkten $P = \{p_1, p_2, \ldots, p_n\}$ **Punktewolke** mit $n \in N$.*

Um ein Oberflächenmodell beschreiben zu können, werden Dreiecke als Grundelement einer Triangulierung benötigt.

Definition 2.5 *Seien q_0, q_1, q_2 drei Punkte. Dann wird das geordnete Tripel $D = (q_0, q_1, q_2)$ als **Dreieck** bezeichnet, falls die Vektoren $q_1 - q_0$ und $q_2 - q_0$ linear unabhängig sind. Die Menge*

$$\Delta := \{r \in \mathbb{R}^3 : r = q_0 + \lambda_1(q_1 - q_0) + \lambda_2(q_2 - q_0) \text{ mit } \lambda_1, \lambda_2 \in \mathbb{R}_+, \lambda_1 + \lambda_2 \leq 1\}$$

*nennt man die dem Dreick D zugehörige **Dreiecksfläche**, den Vektor $N \in \mathbb{R}^3$ mit $N = (q_0 - q_1) \times (q_0 - q_2)$ den **Normalenvektor** des Dreiecks D, wobei $\times$ das Kreuzprodukt ist. Die Punkte q_0, q_1, q_2 heißen **Eckpunkte** des Dreiecks.*

Die einfachste Form eines Modells ist das triangulierte Oberflächenmodell einer Punktewolke selbst. Für eine mathematische Beschreibung einer Triangulierung gibt es in der Literatur mehrere Ansätze. [Bor] wählt die Beschreibung als ebenen Graphen. Im Folgenden wird jedoch die differentialgeometrische Definition von C. Bär [Bär01] verwendet:

Definition 2.6 *Ein **Polyeder** X ist eine endliche Menge von Dreiecken $D_j \subset \mathbb{R}^3$,*

$$X = \{D_1, \ldots, D_k\}$$

mit folgender Eigenschaft: Je zwei dieser Dreiecke schneiden sich entweder

- *gar nicht oder*
- *in genau einer Ecke oder*
- *in genau einer Kante*

Die Vereinigung der zugehörigen Dreiecksflächen

$$|X| := \bigcup_{j=1}^{k} \Delta_j$$

heißt ***geometrische Realisierung*** *von* X*.*

Anders als C. Bär wird hier eine Triangulierung mit ebenen Dreiecken und nicht als eine mit Kurven zerteilte Oberfläche definiert (also nicht als Teilmenge der Mannigfaltigkeit).

Definition 2.7 *Sei X ein Polyeder, wobei die Eckpunkte seiner Dreiecke auf einer zweidimensionalen Untermannigfaltigkeit (der Oberfläche unseres Objekts) liegen. Dann heißt seine geometrische Realisierung*

$$T = |X|$$

auch ***differentialgeometrische Triangulierung*** *der Untermannigfaltigkeit.*

Hieraus lässt sich direkt unser Modellbegriff ableiten. Er besteht aus einer differentialgeometrischen Triangulierung und eventuell zusätzlich ermittelten Oberflächenmerkmalen. Auf unterschiedliche Merkmale wurde in Abschnitt 2.4.2.2 eingegangen. Hier sei nur erwähnt, dass jedes Merkmal einen Basispunkt besitzt. Aus der Krümmung der Umgebung dieses Basispunktes, wird das Merkmal dann berechnet. Wählt man als mögliche Basispunkte ausschließlich Eckpunkte der Triangulierung, so erhält man einen natürlichen Modellbegriff.

Definition 2.8 *Sei T eine differentialgeometrische Triangulierung mit Eckpunktemenge Q und $S := s_1, ..., s_n \subseteq Q$ eine Menge von Basispunkten für Merkmale. Dann heißt das Tupel $M = (T, S)$* ***Modell****.*

Man beachte, dass $S = \emptyset$ sein kann. Zur Abkürzung spricht man auch von „Dreiecken des Modells“, wenn „Dreiecke der differentialgeometrischen Triangulierung T des Modells“ gemeint sind. Um formulieren zu können, wie gut eine Zuordnung von Modell zur Punktewolke ist, muss die Projektion eines Punktes auf das Modell berechnet werden können. Gesucht ist für einen Punkt p also der nächstgelegenen Punkt auf dem Modell.

2.7.2.2 Formulierung als Graph

Definition 2.9 *Ein* **Triangulationsgraph** *$T(P)$ einer Menge P von Punkten, die in einer Ebene liegen, ist eine Zerlegung der konvexen Hülle von P in Dreiecke, bei der die Eckpunkte genau die Punkte aus P sind. Damit ist ein Triangulationsgraph ein* ebener Graph*.*

D.h. einander angrenzende Dreiecke haben genau eine ganze Seite oder genau einen Eckpunkt gemeinsam. Um aus numerischer Sicht gut proportionierte Dreiecke zu erhalten, ist darauf zu achten, dass die Dreiecke möglichst keine allzu stumpfen Winkel enthalten. Dies erreicht man durch Minimierung des maximalen Winkels (oder gleichbedeutend durch Maximierung des minimalen Winkels).

Definition 2.10 *Ein Triangulationsgraph T heißt* **winkel-optimal**, *wenn er von allen möglichen Triangulationsgraphen den größten minimalen Winkel enthält.*

Dies wird durch Eliminierung aller illegaler Kanten erreicht. Eine Kante e heißt *illegal*, wenn $\min_{1\leq i\leq 6} \alpha_i \leq \min_{1\leq i\leq 6} \alpha'_i$ gilt und wird durch e' ersetzt (vgl. Abb. 2.13).

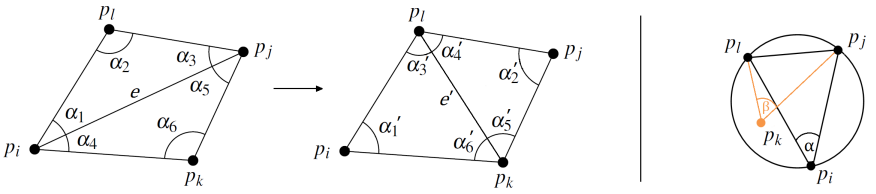


Abbildung 2.13: links: „Edge-Flip“, rechts: p_k verletzt das Kreiskriterium (Abb. aus [Roh11])

Ein Triangulationsgraph enthält besonders gleichmäßige Dreiecke, wenn er keine illegalen Kanten enthält. Dies führt auf den Begriff des Kreiskriteriums.

Definition 2.11 *Ein Triangulierungsgraph T erfüllt das* **Kreiskriterium**, *wenn in keinem Umkreis eines Dreiecks aus T ein anderer Punkt von T enthalten ist.*

Nach dem Peripheriewinkelsatz (siehe Abbildung 2.13 - rechts) bilden alle Dreiecke mit dem selben Umkreis und der selben Sekante $p_l p_j$ an p_i den gleichen Winkel $\alpha = \alpha_{p_i}$, solange alle p_i auf dem Kreis auf der gleichen Seite von $p_l p_j$ liegen (siehe z.B. [RB76] Es ist eine Verallgemeinerung des Satzes von Thales und eine Spezialisierung des Kreiswinkelsatzes). Dreiecke mit einem Punkt p_k innerhalb des Umkreises hätten einen größeren Winkel β an p_k. Eine Kante $p_l p_j$ ist genau dann illegal, wenn ein solcher Punkt $p_k \in P$ existiert.

Eine besonders häufig verwendete Form der Triangulierung ist daher die Delauney-Triangulierung, die wie folgt definiert wird.

Definition 2.12 *Sei $P \subset \mathbb{R}^2$ eine Menge von Punkten in* allgemeiner Lage, *d.h. auf einem beliebigen Kreis befinden sich stets maximal 3 Punkte. Sei T ein Triangulierungsgraph von P. Dann heißt T eine* **Delauney-Triangulierung** *von P, wenn das Kreiskriterium für den Umkreis jedes Dreiecks von T erfüllt ist.*

Zur Berechnung einer Delauney-Triangulierung kann dabei ihr Dual verwendet werden, das Voronoi-Diagramm.

Definition 2.13 *Zwei Graphen G und H heißen* **dual**, *wenn für jede Fläche F auf G der Graph H einen Knoten enthält und für je zwei aneinander grenzende Flächen $F', F'' \in G$ der Graph H eine Kante enthält, die die Knoten verbindet.*

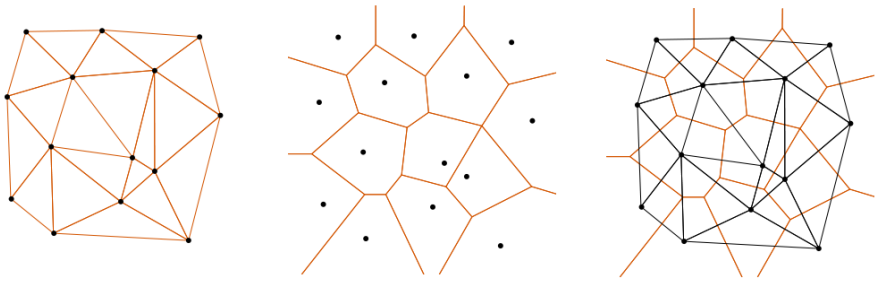


Abbildung 2.14: Eine Dalauney-Triangulierung (links) und das zugehörige Voronoi-Diagramm (mitte) (Abb. aus [Roh11])

Das Voronoi-Diagramm einer Punktemenge P wird erzeugt, indem jedem Punkt p_i die Region aller Punkte $R(p_i) \subset \mathbb{R}^2$ zugeordnet wird, die näher an p_i als an einem anderen Punkt aus P liegen. R heißt dann *Voronoi-Region*. Zur Verdeutlichung ist eine Delauney-Triangulierung und das zugehörige Voronoi-Diagramm sind in Abbildung 2.14 dargestellt. Steht also ein Voronoi-Diagramm zur Verfügung, kann eine Delauney-Triangulierung einfach berechnet werden.

Für Oberflächen im dreidimensionalen Raum kann ein Trick verwendet werden. Die Oberfläche wird lokal auf eine Ausgleichsebene der Punkte projiziert und die Triangulation dort berechnet. Danach projiziert man die Punkte einschließlich des Graphen zurück.

In der Literatur sind unterschiedliche Delauney-Triangulierungsalgorithmen beschrieben. Sie lassen sich grob in drei Kategorien einteilen. Die erste Kategorie sind die *abstandsbasierenden Methoden*. Alle Algorithmen dieser Kategorie haben gemeinsam, dass ein Distanzmaß angewandt wird, um benachbarte Oberflächenpunkte zu finden [HH92], [NA01].

Die zweite Kategorie von Verfahren sind die *Netzreduktionsalgorithmen*. Hierbei wird zunächst dreidimensional vernetzt, die Punkte also zu einem Tetraedernetz zusammengefasst. Dies geschieht beispielsweise durch eine 3D-Delauney-Triangulierung. Anschließend wird ermittelt, welche Tetraderoberflächen zur Oberfläche des Gesamtkörpers gehören und alle weiteren werden gelöscht [HE94]. Die Methode ist durch die benötigte 3D-Triangulation sehr aufwändig.

Die dritte Kategorie der Triangulationsalgorithmen, sind die *incrementellen Methoden*. Hierbei wird ein *„Region Growing“* angewandt und die Triangulation wird von Punkt zu Punkt fortgesetzt. Die Auswahl des nächsten Punktes basiert hierbei auf lokalen Kriterien, wie z.B. dem Normalenvektor [FB99].

3. Konzeption und Umsetzung eines neuen selbstlernenden Algorithmus zur Objektlageerkennung

In diesem Kapitel wird das im Rahmen dieser Arbeit realisierte Komplettsystem zur Objektlageerkennung vorgestellt. In der industriellen 2,5D-Bildverarbeitung sind bisher viele spezielle Probleme gelöst. Um ein paar Beispiele zu nennen: das Abgreifen zylindrischer Bauteile aus einer Kiste (vgl. [Sto11]), 2,5D Lageerkennung von Freiformkörpern aus einem speziellen CAD-Modell oder einer Punktewolke (vgl. [Gmb80]) uVm. Üblich ist hierbei, dass das komplette System, also Sensor, Verarbeitungsalgorithmus und Modelltraining, auf das spezielle Problem angepasst werden. Die Herausforderung dieser Arbeit liegt in der Umsetzung der Objektlageerkennung unter folgenden besonderen Randbedingungen:

Mit Blick auf eine hohe Flexibilität in der Fertigung und niedrige Personalkosten für die Umrüstung bei Wechsel des zu fertigenden Erzeugnisses einer Montagelinie ist es ein immer stärker gefordertes Ziel, ein selbstadaptierendes System zu entwickeln, das mit *einem* Sensorsystem und *einem* Algorithmus *ohne Anpassung durch einen Bildverarbeiter* Modelle trainieren kann, um später ihre Lage zu erkennen. Das Training soll mit der selben Sensorik wie die spätere Lageerkennung durchführbar sein. Die Herausforderung liegt also als Erstes in der geforderten Universalität des Komplettsystems.

Zweitens ist das Teilespektrum eine besondere Herausforderung. Das System soll die Objektlage von einem möglichst großen Prozentsatz typischer Bauteile des Bosch-Erzeugnisspektrums ermitteln können. Hierfür wurde ein Teilekoffer erstellt, der das Spektrum mit seinen vielen Facetten möglichst gut abbildet (siehe [Say11]).

Wie man in Abbildung 3.1 sieht, reicht das Spektrum der Bauteile von schwarzen, diffus streuenden Plastikbauteilen (Steckergehäuse) über matte Keramikbauteile (Zündkerze) bis hin zu stark glänzenden Metallteilen (Kupferspule). Es gibt farbige, stark texturierte Objekte (Leiterplatte), aber auch einfarbige, untexturierte (Magnetkern).

Die wenigen stark texturierten Bauteile können mit texturbasierten Objektlageerkennungsverfahren sehr robust verarbeitet werden (vgl. [Hau11] für eine ausführliche Beschreibung). Alle anderen müssen über ihre Geometrie lokalisiert werden und sind Zielobjektklasse der vorliegenden Arbeit.

Schließlich sind drittens Anforderungen an Kosten- und Zeiteffizienz gestellt. Eine Objektlageerkennung soll mit möglichst geringen Sensorkosten weniger als 500 ms pro Objektla-

Abbildung 3.1: Bauteilekoffer: 34 typische Bauteile, die das Anforderungsspektrum für die Objektlageerkennung bei Bosch möglichst gut wiedergeben (siehe [Say11])

geerkennungszyklus dauern (Bildaufnahme und Berechnung der Objektlage).

Im folgenden Abschnitt 3.1 wird das verwendete Sensorsystem beschrieben und begründet, warum es für die Aufgabenstellung besonders geeignet ist. Abschnitt 3.2 geht auf den Algorithmus ein und in den folgenden Abschnitten 3.3, 3.4 und 3.5 wird die Umsetzung von dessen Einzelschritten genauer beleuchtet. Abschnitt 3.6 und 3.7 stellen schließlich die Kalibrierung des Sensorsystems und die Umsetzung der automatischen Modellbildung vor.

3.1 Sensorsystem

In Abschnitt 2.2 wurden mögliche Sensorsysteme zur Ermittlung von 2,5D-Bildern der Szene beschrieben.

Sehr breite Anwendung im industriellen Umfeld findet hierbei der Laserscanner - oft in der Ausführung eines Linienscanners. Das Laserlicht kann hierbei so intensiv gewählt werden, dass es gut auf allen möglichen Oberflächen selbst unter Fremdlichteinfluss zu sehen ist. Ein großer Nachteil ist jedoch die Aufnahmezeit. Da der Abstand zu jedem Zeitpunkt immer nur auf einem schmalen Objektstreifen trianguliert werden kann, muss das Objekt „gescannt“ werden. Hierbei können nicht beliebig viele Scanzeilen pro Zeit eingelesen werden, schon allein weil die Belichtungszeit der einzelnen Objektzeile mit zunehmender Scangeschwindigkeit abnimmt. Dies kann teilweise durch stärkere Laser ausgeglichen werden, was wiederum durch Sicherheitsaspekte begrenzt ist (etwa Augenschutz des Personals in der Produktion). Die typische Geschwindigkeit aktueller Systeme beträgt bis zu 70 kHz Zeilenfrequenz bei gutmütigen Objektoberflächen. Bei einer Auflösung von 1000 Zeilen pro Bild benötigt der Sensor also ca. 14 ms pro Aufnahme. Die Belichtungszeit würde dann aber nur noch 0,014 ms in jeder Zeile betragen. Für unsere Randbedingungen, also unkooperative Oberflächen unter Fremdlichteinfluss und bei einer Laserschutzklasse von „unbedenklich“ (Klasse 2a), ist jedoch eher 1 ms Belichtungszeit je Zeile erforderlich, d.h. 1 Sekunde Scanzeit je Bild, was zu langsam ist. Ferner sind Laserscanner mit typischen Preisen von mehreren Tausend Euro recht teuer und je nach Ausführung recht schwer und groß, womit sie schon deshalb nicht für die Installation auf einem Roboterarm geeignet wären.

Viel kompakter und schneller sind dagegen Laufzeitkameras. Ihre flächige Tiefenbildaufnahme macht sie One-Shot-fähig, d.h. ein Tiefenbild kann ohne Scannen erzeugt werden, womit die Belichtungszeiten sehr viel länger gewählt werden können. Außerdem ist die

Aufnahme bei nicht statischen Szenen möglich, es bleibt jedoch eine Bewegungsunschärfe, da mindestens drei Aufnahmen nötig sind (sehr kurz hintereinander). Ein systembedingtes Problem ergibt sich jedoch aus der Höhe der Lichtgeschwindigkeit. Es müssen sehr kurze Zeitdifferenzen gemessen werden, so dass keine beliebige Tiefengenauigkeit möglich ist. Zum Zeitpunkt dieser Arbeit haben State-of-the-art-Laufzeitkameras eine typische z-Auflösung im Zentimeterbereich und sind damit für Objekte mit Strukturen im Milimeterbereich ungeeignet. Ferner sind auch diese Systeme noch recht teuer (1000 Euro aufwärts). Dennoch könnte eine Weiterentwicklung auf diesem Gebiet die Kameras für zukünftige Anwendungen im industriellen Umfeld interessant machen.

Es bleibt also ein Stereokamerasystem. Es kann eine bessere z-Auflösung als Laufzeitkameras erreichen (nur von der Basisbreite und der Pixelauflösung der Kamera, nicht jedoch von der Laufzeit abhängig), es ist aber trotzdem im Gegensatz zu Laserscannern One-Shotfähig. Außerdem ist eine Stereokamera die kostengünstigste Alternative. Wie in Kapitel 2.2 ausgeführt ist die Tiefenbildrekonstruktion jedoch auf Textur am Objekt angewiesen, welche bei unserer Anwendung nicht vorhanden ist. Einen Ausweg bietet das Aufbringen der Textur durch einen zusätzlichen Projektor.

Aus diesen Gründen wurden als Sensorsystem zwei Kameras (Stereokamera) mit einem in der Mitte montiertem Projektor gewählt. Es bleibt die Frage, welches *Muster* gewählt werden sollte? Eine ausführliche Untersuchung ist in [JB97] zu finden, wobei eine Sequenz von Mustern als sehr genaue Möglichkeit identifiziert wird. Nachteilig an dieser Lösung ist jedoch, dass sie etwas teuerer, langsamer und lichtschwächer als ein festes Muster ist. Bei festen Mustern sind Streifen, Regenbogen oder Rauschmuster üblich. Streifen haben den Vorteil, dass der Tiefenrekonstruktionsalgorithmus zeilenweise sehr gut parallelisiert werden kann, sowie Streifen senkrecht zur Stereobasis sehr guten Kontrast entlang der Epipolarlinien des Stereosystems bieten. Bei Rauschmustern dagegen ist die Eindeutigkeit im Bild leichter herzustellen.

Nach Untersuchungen überwogen die Vorteile eines Streifenmusters (vgl. [Kno08]).

Ein weiterer Vorteil eines zusätzlichen Musterprojektors ist die Möglichkeit, den Projektor gegen jede der Einzelkameras zu kalibrieren. Diese Option wurde ausführlich im Rahmen der Diplomarbeit von Dose [Dos09] untersucht. Die Umsetzung wird im Abschnitt 3.3 vertieft und bietet verglichen mit nur einer Kamera und einem Projektor den großen Vorteil der Redundanz.

In dieser Arbeit wurde letztendlich ein Sensorkopf verwendet (vgl. [Kno09] und Abbildung 3.2), der aus einem Stereokamerapaar mit einer Basisbreite von 12 cm, sowie einem dazwischen positionierten Musterprojektor mit festem Streifenmuster besteht. Wegen des gewünschten kurzen Arbeitsabstands (10 bis 20 cm), sind die Kameras nicht parallel, sondern „schielend“ mit einem Winkel von ca. 22 Grad zueinander ausgerichtet.

Die beiden CCD-Kameras haben ein Auflösung von 768x576 Pixel mit jeweils 8Bit auf den drei Kanälen des RGB-Farbraumes.

Der Sensorkopf ist auf einem Roboterarm montiert und kann daher sehr exakt positioniert werden. Die Auswertung findet auf einem handelsüblichen PC statt, der mit einem 3 GHz Quadcore Prozessor und 4 GB Arbeitsspeicher ausgestattet ist.

3.2 Gesamtkonzept des Algorithmus

Um der Forderung nach Universalität bezüglich des Teilespektrums gerecht zu werden, ist der Ablauf des Algorithmus zur Objektlageerkennung in zwei Phasen gegliedert - Trainings- und Detektionsphase. In der Trainingsphase wird das zu erkennende Objekt vor bekanntem

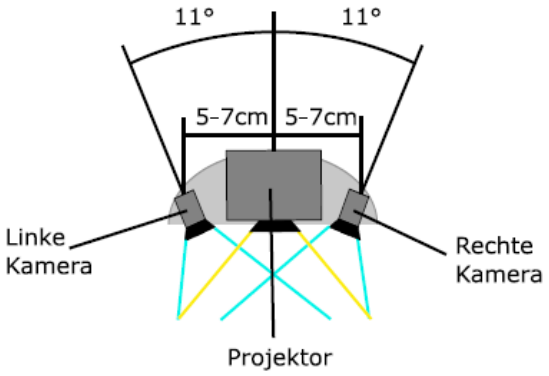


Abbildung 3.2: Schema des verwendeten Sensorkopfes (aus [Dos09])

Hintergrund präsentiert. Der Algorithmus ermittelt semiautonom alle zur Erkennung benötigten Daten. Dieser Vorgang ist im Abschnitt 3.7 genauer beschrieben. Der Algorithmus zur Objektlageerkennung ist Gegenstand diesen Abschnitts.

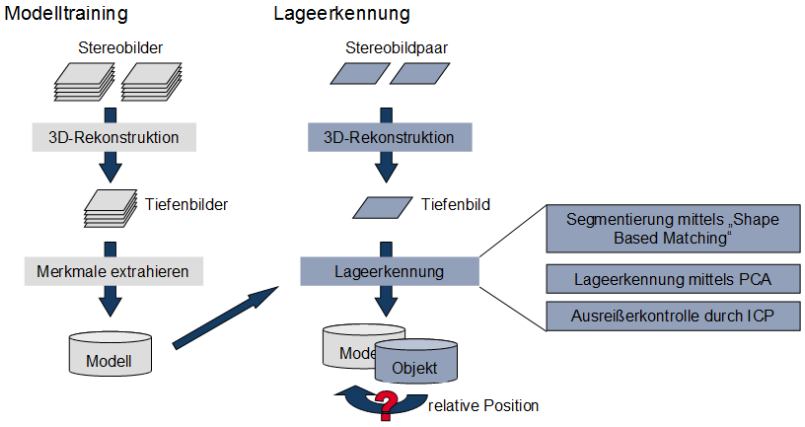


Abbildung 3.3: Schema: Ablauf des Modelltrainings und der Objektlageerkennung

Welche Algorithmen zur Objektlageerkennung aus Tiefenbildern sind für unsere Problemstellung geeignet? Der Hauptvorteil von Verfahren, die auf lokalen 3D-Merkmalen beruhen (z.B. „Spin Images“, vgl. Abschnitt 2.4.3) liegt, wie bereits im Grundlagenkapitel erörtert, auf der Hand. Die Lage kann bereits aus einer kleinen, korrekt rekonstruierten Objektregion bestimmt werden. Sie sind damit robust gegen Fehlrekonstruktionen und Überdeckungen, solange diese nicht einen Großteil des Objekts betreffen. Dies würde sie für unsere Aufgabenstellung prädestinieren, sofern die Auflösung unseres Sensorsystems ausreicht. Wie in Abschnitt 3.1 ausgeführt, hat die Sensorik zwar entlang der Bildvertikalen v pixelgenaue Auflösung (entspricht bei üblichem Arbeitsabstand von 10 cm etwa 0,2 mm/Pixel auf dem Objekt), in der Bildhorizontalen u durch die Abstände zwischen Streifen des Musterprojektors jedoch nur einen Abstandswert auf etwa 6 Pixel (je nach Arbeitsabstand etwa 1,2 mm auf dem Objekt). Diese Anisotropie würde sich negativ auf die Merkmalsberechnung auswirken, so dass die Auflösung entlang der Bildvertikalen entsprechend abgesenkt werden müsste. Die Größe der Region, aus der ein einzelnes Merkmal berechnet wird, müsste mindestens drei rekonstruierte Punkte enthalten, was zu einer Größe auf dem Objekt von etwa 3,6 mm führte. Dies ist für die Strukturen der zu erkennenden Bauteile des Referenzbauteilesatzes zu groß.

Aus diesem Grund wird in dieser Arbeit im Gegensatz zu einem lokalen, nur Bildregionen verwendenden Algorithmus ein globaler Ansatz verfolgt. Die Registrierung zweier Punktewolken aneinander (Modell und Objekt) ist ein Optimierungsproblem (siehe Abschnitt

2.5.2), dessen literaturübliche Lösung der ICP-Algorithmus ist. In Abschnitt 3.5 wird die verwendete Variante dieses Algorithmus näher beschrieben. Die Hauptschwäche dieses Algorithmus ist zunächst, dass er gegen das nächste *lokale* Minimum des mittleren quadratischen Abstands der Punktewolken konvergiert, welches nicht die korrekte Lage sein muss.

Damit das nächste lokale Minimum unsere gesuchte Lageschätzung ist, benötigen wir also eine ausreichend genaue initiale Lageschätzung - einen guten Startwert. Nach Betrachtung und Bewertung von Groblageerkennungsalgorithmen liegt der Schluss nahe, dass der am besten hierfür geeignete Algorithmus die Hauptachsentransformation ist (vgl. 2.4.3 für die ausführliche Bewertung), dessen Umsetzung in Abschnitt 3.4 erläutert wird.

Die Hauptachsentransformation ist bezüglich Laufzeiteffizienz und Genauigkeit das Mittel der Wahl. Die Schattenseite dieser Art von Algorithmus ist jedoch, dass *alle* Punkte *gleichgewichtet* in die Kalkulation eingehen, also auch Fehlrekonstruktionen oder Punkte, die zum Hintergrund und nicht zum Objekt gehören. Um dennoch eine gewisse Robustheit zu gewährleisten wird eine Segmentierung benötigt, die Objektpunkte aus dem Hintergrund löst. Diese wird ebenfalls in Abschnitt 3.4 erklärt.

Die Detektionsphase umfasst also nach der sensorischen Erfassung (Abschnitt 3.1) die Verarbeitung der Sensordaten zu einem Tiefenbild (beschrieben in Abschnitt 3.3), die Segmentierung des Objekts vom Hintergrund (Abschnitt 3.4) sowie die grobe und feine Objektlagebestimmung aus der segmentierten Punktewolke (Abschnitt 3.5). Der schematische Ablauf ist in Abbildung 3.3 dargestellt.

3.3 Umsetzung der Tiefenbildaufnahme

Dieser Abschnitt ist der Rekonstruktion von Raumpunkten aus der Sensorinformation, also einem Stereobildpaar, bei dem die Szene mit einem Streifenmuster beleuchtet wurde, gewidmet.

Die Auswahl des verwendeten Musters wird in 3.3.1 begründet. Der erste Schritt bei der Rekonstruktion ist die Segmentierung der Streifen in den Einzelbildern (Abschnitt 3.3.2), gefolgt von der Berechnung der Raumpunkte aus den Streifenpositionen - zunächst mittels der kalibrierten Stereokamera (Abschnitt 3.3.3), dann mittels der Erweiterung durch einen kalibrierten Projektor (Abschnitt 3.3.4). Zuletzt werden alle Daten zu einem Tiefenbild bzw. einer Punktewolke im Raum fusioniert (Abschnitt 3.3.5).

Das hier beschriebene Verfahren zur Tiefenbildrekonstuktion ist im Rahmen einer von S. Dyblenko, C. Knoll, K. Janschek und mir betreuten und bei der Robert Bosch GmbH CR/APA2 durchgeführten Diplomarbeit [Dos09] entstanden und umgesetzt worden. Teile der beschriebenen Algorithmen sind dort noch detaillierter ausgeführt.

Eines unserer Ziele ist die zeiteffiziente Umsetzung des Gesamtalgorithmus. Die Tiefenbildberechnung nimmt einen großen Teil dieser Zeit in Anspruch. Deshalb wurde sie teilweise in C++ parallelisiert und beschleunigt (andere Teile wurden mittels der Bildverarbeitungsbibliothek Halcon der Firma MVTec programmiert [Gmb80]). In Abschnitt 3.3.6 wird auf Details diesbezüglich eingegangen.

3.3.1 Musterauswahl

Die Hauptherausforderung bei der Tiefenbildrekonstruktion mittels Stereokamerasystemen ist die Korrespondenzpunktsuche. Der Abstand zur Kamera kann nämlich nur für Punkte im Raum trianguliert werden, deren Position in linker und rechter Kamera bekannt ist. Diese sind beispielsweise bei homogen texturierten Flächen schon prinzipiell nicht erkennbar.

Um diesem *Problem homogen texturierter Flächen* zu entgehen, findet ein Musterprojektor Verwendung. Es werden also nicht mehr Korrespondenzen von Texturmerkmalen des Objekts gesucht, sondern nur noch Korrespondenzen von projizierten Musterstücken. Also findet sich Label A (eindeutiges Musterstück) im linken und rechten Kamerabild. Damit diese Zuordnung eindeutig ist, darf ein Label im Bild auch nur einmal vorkommen.

3.3.2 Musterextraktion

Als erster Schritt zur Tiefenbildrekonstruktion müssen die genaue Position und Farbe jedes Streifens in jedem der beiden Kamerabilder ermittelt werden. Aus jedem Einzelbild werden hierfür zwei Matrizen erstellt. In der ersten steht für jede Zeile die Sequenz der Streifen (also z.B. gelb, gelb, rot, cyan, rot,...), in der zweiten die subpixelgenauen x-Koordinaten des jeweiligen Streifenmittelpunkts im Bild. Man wird jeweils nur einen Tiefenwert pro Streifen erhalten und im Rahmen der vorliegenden Arbeit wird der Mittelpunkt verwendet (vgl. [Dos09]). Die Matrizen sind also maximal (Bildhöhe in Pixeln = 576) mal (Anzahl der Label = 162) groß.

Wie findet man nun am besten die Streifen im Bild? Die Intensität des Projektors wurde maximiert, so dass auch auf schwarzem Plastik noch Streifen zu sehen sind. Damit liegt die Vermutung nahe, dass auch in den Bildern primär die Streifen zu sehen sind.

Der erste Ansatz, die Streifen durch Gradientenbildung in den Farbkanalbildern zu detektieren (also einen Streifen genau dann als z.B. cyan zu erkennen, wenn er im blauen und grünen Kanal erkannt wurde) schlug fehl, da der Blaukanal zwischen zwei gelben Streifen ein Minimum aufweist (siehe Abbildung). Aus Applikationsgründen werden an den Kameras keine Infrarotschutzfolien verwendet. Der Blaukanal typischer Farbkameras ist jedoch für Infrarotlicht empfindlich. Eine mögliche Erklärung ist daher, dass der Rotanteil in den gelben Streifen auch Infrarotanteile besitzt, die im Blaukanal detektiert werden.

Einen Ausweg bietet die Transformation des RGB-Bildes in die HSV Darstellung (englisch: Hue = Farbton, Saturation = Sättigung, Value = Hellwert). Die Streifen können dabei direkt im Hellwertbild robust erkannt werden, da - egal welche *Farbe* der Streifen hat - die *Helligkeit* im Bild an einem Streifenmittelpunkt jeweils lokal am höchsten ist.

Da die Helligkeit eines Streifens im Bild aber je nach Reflektionseigenschaften des beleuteten Objekts und je nach Winkel der Objektoberläche zu Projektor und Kamera (Glanzflecken) variiert, muss der Schwellenwert für die Streifenerkennung dynamisch angepasst werden. Da sich die Umgebungsbedingungen von Detektion zu Detektion ändern können sollen und um eine Parametrisierung durch den Benutzer zu vermeiden, wird die Helligkeit nur relativ zur Umgebung im Bild gemessen. Kritisch ist hierbei die Wahl der Größe der Umgebung. Diese soll ein konstantes Verhältnis von Streifen zu Hintergrund enthalten, muss also die Breite eines Vielfachen des Streifenabstands haben. Da der Streifenabstand im Bild nahezu invariant gegenüber Arbeitsabstandsänderung und Position im Bild ist [Dos09], kann die Umgebungsbreite fest gewählt werden - bei uns als einfacher Streifenabstand.

Wie in Abbildung 3.5 zu sehen, ist jeder Einzelstreifen nicht überall gleich hell, vielmehr ist der Helligkeitsverlauf parabelförmig (Beugung am Musterdia). Um die Position der Streifenmittelpunkte zu präzisieren, wird deshalb in jeden Streifen eine Parabel eingepasst, deren x-Wert des Scheitelpunkts im weiteren Verlauf als Position des Streifenmittelpunkts verwendet werden soll.

Eine Parabel ist eine Funktion zweiter Ordunung $y = a \cdot x^2 + b \cdot x + c$ und kann damit aus drei Stützpunkten berechnet werden. Hierfür verwendet man drei aufeinanderfolgende Pixel x_1, x_2, x_3 und deren Werte im Helligkeitsbild y_1, y_2, y_3. Der gesuchte Scheitelpunkt ist die Nullstelle der ersten Ableitung $x_0 = -\frac{b}{2 \cdot a}$, welche durch die Parameter a und b bestimmt

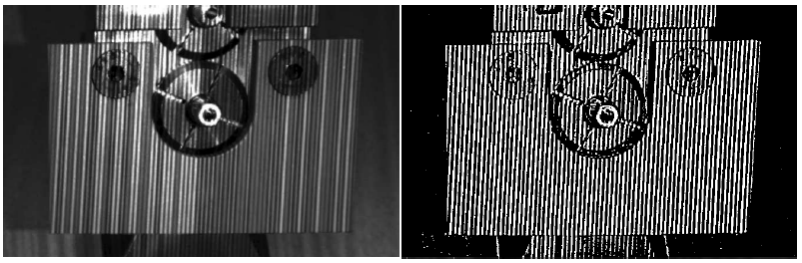


Abbildung 3.4: links: Hellwertbild der HSV-Transformation des Orginalbildes, rechts: Mittels adaptivem Schwellwert berechnete Streifensegmentierung (aus [Dos09])

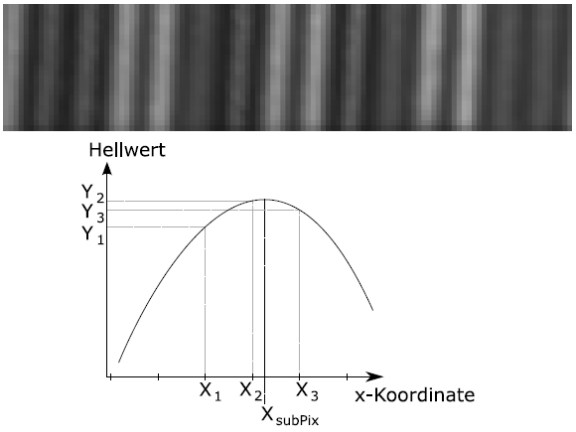


Abbildung 3.5: oben: Vergrößerung des Hellwertverlaufes der Streifen, unten: Parabelfit zur Subpixelgenauen Bestimmung der Streifen-x-Position (aus [Dos09])

ist. Damit ergibt sich durch Auflösen des Gleichungssystems und mit der Vereinfachung für direkt aufeinanderfolgende Stützstellen

$$(x_1 - x_2) = (x_2 - x_3) = -1 \text{ und } (x_3 - x_1) = 2$$

die Formel (vgl. [Dos09]):

$$x_{\text{subPix}} = -\frac{y_1 \cdot (x_3^2 - x_2^2) + y_2 \cdot (x_1^2 - x_3^2) + y_3 \cdot (x_2^2 - x_1^2)}{2 \cdot (2 \cdot y_2 - y_1 - y_3)}$$

Ist die subpixelgenaue Position der Farbstreifen im Bild bestimmt, fehlt nur noch deren Farbzuordnung. Hierfür wird der entsprechende Wert an der Stelle des Farbtonbildes (Hue) der HSV-Zerlegung verwendet. Das ist ein Wert zwischen 0 und 360 auf dem Farbkreis (siehe Abbildung 3.6).

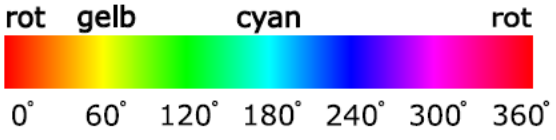


Abbildung 3.6: Farbzuordnung des Wertes im Farbtonbild (Hue) einer HSV-Zerlegung

Da unsere zu vermessenden Objekte jedoch farbig sein können, müssen die Schwellwerte der Zuordnung: Hue-Wert -> Streifenfarbe, beim Modelltraining gelernt werden. Der hierfür entwickelte Algorithmus bekommt als Eingang zwei Bilder. Eines mit dem Objekt vor einem möglichst homogenen Hintergrund, beleuchtet mit dem Streifenmuser, und ein

zweites aus der selben Kameraposition ohne das Objekt. Durch Differenzwertbildung wird das Objekt segmentiert, um sicherzustellen, dass nur mit Objektpunkten trainiert wird.

Auf dem so segmentierten Objekt, wird ein Histogramm über das Farbtonbild der HSV-Zerlegung gebildet. Dieses Histogramm wird geglättet, bis nur noch drei Maxima übrig sind (nämlich die drei Farbtöne der drei Streifen des Alphabets des Musters). Als Schwellwert wird der Mittelpunkt von je zwei Maxima verwendet (vgl. Abbildung 3.7). Der hier beschriebene Abschnitt wurde in der Bildverarbeitungsbibliothek Halcon der Firma MVTec implementiert.

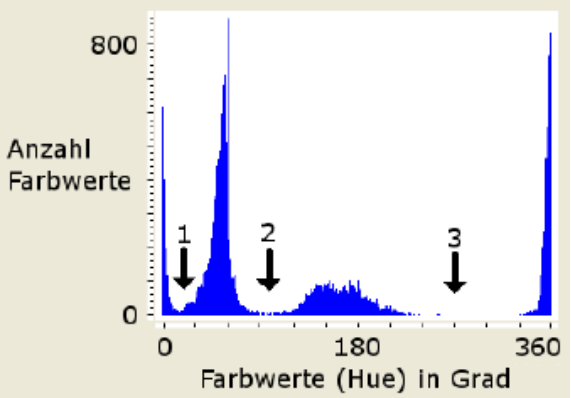


Abbildung 3.7: Histogramm der Farbwerte auf einem Beispielobjekt

3.3.3 Tiefenbild mittels Stereokamera

Das verwendete, kalibrierte Kamerasystem befindet sich durch die Rektifizierung in Stereonormalanordnung (zu Kalibrierung siehe Abschnitt 3.6, für den Begriff des Stereonormalfalls siehe Grundlagenkapitel 2.2.1). Der Abstand des betrachteten Objektpunktes zur Kamera wird aus der Disparität des Punktes von linker zu rechter Kamera berechnet, also aus dem Versatz der x-Position im Bild. Die Herausforderung ist dabei, zu jedem Bildpunkt den korrespondierenden Punkt im anderen Bild zu finden. Mit dem bei uns verwendeten Streifenmuster entspricht dies einer *Erkennung des Labels* in beiden Einzelbildern. Die Streifensequenz muss also der Position im Muster zugeordnet werden. Das Ergebnis ist die Zuordnung f aus der Menge der Label L (also Streifennummern im Muster) zu ihrer x-Position im Bild u in jeder Zeile jeweils für die linke und rechte Kamera.

$$f_n : L \to U, \quad f_n(l) = u$$

mit der Bildzeilennummer n, der Menge der gefundenen Label $\widetilde{L} \subset L = \{1 \dots 162\}$, $l \in L$ einem Label, $u \in U$ der x-Position im Bild mit $U = [0, 768] \subset \mathbb{R}$.

Korrespondierende Punktpaare liegen durch die Rektifizierung der Kamerabilder immer in der gleichen Zeile der Einzelbilder. Aus diesem Grund wurde auch das Streifenmuster gewählt, da es ja nur auf die Reihenfolge der Farbstreifen in jeder Zeile ankommt. Weitere Details sind in [Dos09] zu finden. Hier wird nur das für diese ARbeit wichtige Problem möglicher Fehlkorrespondenzen diskutiert.

Diese können zwei Ursachen haben. Erstens ein Teil der Streifen ist verdeckt oder eine Farbe wurde falsch segmentiert. Da das Muster aber alle Kombinationen enthält, wird es einem falschen Label zugeordnet.

Die Fehlzuordnungen können nur mittels Redundanz erkannt werden. Das verwendete Verfahren wird im folgenden Abschnitt 3.3.4 vorgestellt. Allerdings können schon direkt nach der Segmentierung einige Plausibilitätsüberprüfungen durchgeführt werden, um die Fehlerquote zu verringern.

Zunächst kann die überprüfte Sequenz verlängert werden, indem fünf statt nur vier Streifen um ein Label zuzuordnen verwendet werden. Das Muster ist mit vier Streifen eindeutig, stimmt der fünfte nicht überein, so muss eine Fehlsegmentierung vorliegen. Je mehr Streifen in Folge richtig sind, bzw. für die Zuordnung verwendet werden, desto robuster ist das Ergebnis gegen Fehlzuordnungen. Die Kehrseite der Medaille ist jedoch, dass die Bauteilstrukturen nicht zu dünn sein dürfen. Eine Struktur, auf der nur vier oder weniger Streifen zu sehen ist, kann dann nicht mehr tiefenrekonstruiert werden. Fünf Streifen haben sich empirisch bei unserem Bauteilsatz und unserem Sensoraufbau als bester Kompromiss zwischen Dichte und Fehlerquote herausgestellt. Verwendet man zusätzlich die Projektorkalibrierung nach Abschnitt 3.3.4, kann die Streifenanzahl für eine korrekte Zuordnung sogar auf drei reduziert werden (bei einer Mustereindeutigkeit von vier Streifen!). Wie das geht, wird in Abschnitt 3.3.5 vorgestellt.

Ein zweiter häufiger Fehler ist, dass Streifen gar nicht erkannt werden oder zusätzliche Streifen durch Reflexionen entstehen. Dies kann durch die Überprüfung des Streifenabstands in der Region erkannt werden. Sind zwei Streifen beispielsweise doppelt so weit auseinander, als die übrigen Streifen der Region, wurde ein Streifen nicht erkannt. Dies wird bei der Labelzuordnung entsprechend berücksichtigt. Am einfachsten dadurch, dass eine neue Sequenz begonnen wird.

3.3.4 Tiefenbild mittels Projektor

Im letzten Kapitel wurde erwähnt, dass man bereits mit *einer* Kamera und einem Projektor 3D-Punkte rekonstruieren kann. Dieser Abschnitt ist der Umsetzung davon mit einem Sensorkopf gewidmet, der neben der Stereokamera auch einen Projektor mit feststehendem Muster aufweist (siehe Abschnitt 3.1).

Zunächst wird der Projektor zu einer Kamera wie in Abschnitt 3.6.2 beschrieben, kalibriert. Hieraus erhält man eine Funktion h_n für jede Bildzeile $h_n : (U, L) \rightarrow [0, \inf] \subset R$ mit der Bildzeilennummer n, $U = [0, 768] \subset R$ der Menge der x-Positionen im Bild und der Menge der Label $L = \{1 \dots 162\}$, die einem Label an einer bestimmten Stelle im Bild den Abstand zur Kamera zuordnet. Da die projizierten Streifen nicht exakt gerade sind und durch Ungenauigkeiten in der Kamerarektifizierung ist h von der Bildzeile abhängig.

Zur Rekonstruktion eines Tiefenbildes wird die Musterextraktion und Streifensegmentierung wie bei der Stereotriangulation durchgeführt. Das Ergebnis ist die Zuordnung f aus der Menge der Label L (also Streifennummern im Muster) zu ihrer x-Position im Bild u in jeder Zeile der Kamera.

$$f_n : L \rightarrow U, \quad f_n(l) = u$$

mit der Bildzeilennummer n, der Menge der gefundenen Label $\widetilde{L} \subset L = \{1 \dots 162\}$, $l \in L$ einem Label, $u \in U$ der x-Position im Bild mit $U = [0, 768] \subset R$.

Kombiniert man f_n und h_n zeilenweise erhält man den Abstand jedes gefundenen Labels im Bild von der Kamera:

$$g_n : (f_n(L), L) \rightarrow [0, \inf] \subset R, \quad g_n(l) = x$$

mit gefundenem Label $l \in \widetilde{L}$ und Tiefenwert $x \in [0, \inf]$.

Hierdurch ist die gewünschte Redundanz erzeugt, da man sowohl die linke, als auch die rechte Kamera gegen den Projektor kalibrieren kann. Man erhält also eigentlich drei Tiefenbilder, eines aus der Kalibrierung der linken Kamera zum Projektor, eines aus der Kalibirierung der rechten Kamera zum Projektor und eines aus der Stereotriangulation aus linker und rechter Kamera. Der Berechnungsaufwand ist hierbei minimal, da die Tiefenwerte einfach aus einer Tabelle abgelesen werden können (bzw. linear interpoliert s.u.).

Durch die Projektorkalibrierung kann die Redundanz, dass sowohl im linken als auch im rechten Kamerabild segmentiert wird, genutzt werden, was mit der Stereotriangulation alleine nicht möglich wäre. Zu beachten ist hierbei jedoch, dass auf zwei Kamerabilder (linkes und rechtes) der gleichen Szene *das gleiche Verfahren* angewandt wird. Damit ist die Rekonstruktion anfällig gegen Fehler, die verfahrensbedingt sind, wie beispielsweise farbige Oberflächen, die zu einer falschen Streifenfarbsegmentierung führen. Die Hauptfehlerursachen jedoch, dass Streifen verdeckt werden oder dass Sequenzen durch Tiefensprünge im Objekt falsch erkannt werden, können wesentlich verringert werden, da sie ebenso wie Glanzflecke meist an einer Objektstelle nur in *einem* Bild auftreten. Das andere Bild kann dann zur richtigen Rekonstruktion verwendet werden.

3.3.5 Optimierter Gesamtalgorithmus zur Tiefenbildrekonstruktion

Mit der Projektorkalibrierung können also redundante Daten bereit gestellt werden. Wenn die konkurrierenden Rekonstruktionen jedoch unterschiedliche Tiefen ergeben, muss entschieden werden, welche für das Endergebnis verwendet werden soll. Ferner bietet diese weitere Möglichkeiten zur Verbesserung der Robustheit und Dichte der Rekonstruktion (siehe auch [Gro11]).

Der optimierte Algorithmus setzt sich aus folgender Sequenz zusammen:

1. Bildaufnahme linke und rechte Kamera
2. Rektifizierung der Einzelbilder
3. Musterextraktion in den Einzelbildern
4. Zuordnung über 5er Sequenzen (für Eindeutigkeit genügen schon 4-er Sequenzen)
5. Zuordnung über kürzere Sequenzen
6. Verdichtung der Zuordnung durch Vergleich verschiedener Zeilen
7. letzte Verdichtung durch Projektorkalibrierung
8. Disparitätsberechnung der Musterpunkte
9. Triangulation der Raumpunkte

Die Punkte 1 bis 3 und teilweise 4 wurden in den vorstehenden Abschnitten des Kapitels beschrieben. Der Rest ist Gegenstand dieses Abschnitts.

Nach Durchführung der Bildaufnahme, Rektifizierung und Musterextraktion können die Farbstreifen gelabelt werden. Wie oben erwähnt, werden dafür zunächst 5er-Sequenzen verwendet, um Segmentierungsfehler zu minimieren. Man erhält eine lückenhafte Zuordnungsmatrix, in der in jeder Zeile die aufeinanderfolgend gefundenen Label und deren x-Position im Bild stehen, z.B. könnte die Sequenz

$$S = (12, 13, 14, 15, 16, 22, 23, 24, \ldots)$$

enthalten sein. Es gilt nun diese zu vervollständigen.

In den Lücken wurde keine zusammenhängende Sequenz von 5 oder mehr Streifen mehr gefunden. Es könnten aber kürzere Sequenzen gefunden worden sein. Grund kann entweder sein, dass ein Streifen nicht richtig segmentiert wurde, oder aber dass er tatsächlich durch einen Sprung in der Objektoberfläche vom Objekt selbst verdeckt ist. Erst Sequenzen ab vier Streifen sind global eindeutig, *lokal* können jedoch auch kürzere Sequenzen eindeutig

sein. Wurde in obigem Beispiel etwa die Sequenz $\widetilde{S} = (\text{cyan}, \text{rot}, \text{rot})$ zwischen Label 16 und Label 22 gefunden, könnte sie an dieser Stelle im Muster eindeutig sein. In diesem Schritt wird also Lücke für Lücke durchgegangen und die Stücke auf lokale Eindeutigkeit geprüft. Falls genau eine Übereinstimmung für die Sequenz gefunden wird, wird sie gelabelt und der Matrix hinzugefügt.

Dies setzt jedoch die Erfüllung der Monotoniebedingung durch die Oberfläche voraus. Im Prinzip ist es möglich, dass beispielsweise zwischen Label 16 und 22 tatsächlich Label 25 zu sehen ist. Betrachten wir als Beispiel hierfür den Henkel einer Kaffeetasse. Auf der Tasse neben dem Henkel sind die Label 16 bzw. 22 zu sehen, auf dem Henkel dazwischen jedoch das Label 25. In der Praxis kam dieser Fall nicht vor. Formal wird die zusätzliche Randbedingung der ausreichenden Monotonie der Objektoberfläche gesetzt, d.h. dass Sprünge an Stellen, *hinter* die man sehen kann, nicht zu groß sein dürfen. Sprungstellen können mittels der Projektorkalibrierung auch während der Laufzeit entdeckt werden und somit auf mögliche Rekonstruktionsfehler hingewiesen werden.

Wegen der schnelleren Zuordnung wird im Rahmen dieser Arbeit eine Baumstruktur verwendet. Der Baum ist nach Farbstreifen verzweigt und besitzt die Tiefe vier, wobei an den Verzweigungen der Tiefe zwei und drei zusätzliche Blätter hängen. In den Blättern stehen alle möglichen Label für die gesuchte Sequenz. Würde man also beispielsweise nach der Sequenz $\widetilde{S}$ suchen, würde in der ersten Ebene der Ast „cyan“, in der zweiten der Ast „rot“ und in der dritten wieder der Ast „rot“ genommen werden. Ein vierter Streifen wurde in dieser Beispielsequenz nicht gefunden, also wird direkt das Blatt auf der dritten Hierarchieebene genommen. Dieses Blatt enthält mehrere mögliche Label, jedoch nur eines zwischen 17 und 21, so dass eindeutig zugeordnet werden kann. Die Sequenz S kann also durch die drei Streifen ergänzt werden und wir erhalten

$$S' = (12, 13, 14, 15, 16, \textit{18,19,20}, 22, 23, 24, \ldots)$$

.

Die bisherigen Schritte dieses Abschnitts 3.3.5 beruhen auf folgendem Grundgedanken: Man stelle sich das Muster wie ein Netz vor, das über die Oberfläche gespannt ist. Man weiß a priori, wie das Netz aussieht, man weiß aber nicht, wie es im Bild auftaucht. Welche Stelle des Netzes wo ist, ergibt gerade die Oberflächenstruktur. In horizontaler Richtung im Bild wurde dieser Gedanke ausgereizt. In diesem Abschnitt sollen nun verschiedene Zeilen in vertikaler Richtung verglichen werden, um die Rekonstruktion des Tiefenbildes weiter zu verdichten.

Alle bisher gefunden sicheren Korrespondenzen werden verwendet, es wird lediglich versucht verbleibende Lücken aufzufüllen. Ist ein Label gefunden, muss es sich in der nächsten Zeile, sollte kein Höhensprung in der Objektoberfläche vorliegen, an der selben x-Position befinden. Die Verschiebung kann maximal ein Pixel weit sein. Kommt man von einem Label in der nächsten Zeile innerhalb dieser 3-Pixelmaske (dem Pixel direkt darunter und den Pixeln links und rechts daneben) zu einem gefunden Streifen gleicher Farbe, so wird versucht den Streifen vertikal fortzusetzen. Als korrekte Zuordnung wird er jedoch erst dann bewertet, wenn man mit dieser Methode zu irgendeinem Zeitpunkt wieder auf das gleiche bereits zugeordnete Label in einer anderen Zeile stößt. Es werden somit nur Lücken aufgefüllt, die *oben und unten* mit gefunden Korrespondenzen begrenzt sind.

Dies soll verhindern, dass ein fortgesetzter Streifen zwar die gleiche Farbe, aber korrekt zugeordnet ein anderes Label hätte, was an Sprungkanten in vertikaler Richtung der Fall ist. Außerdem ist die Farbe an sehr schwach oder sehr stark beleuchteten Stellen nur schlecht bestimmbar. Welche Farbe hat ein schwarzer bzw. weißer Streifen? Die korrekte Antwort auf die Frage ist „Keine“. Der hier verwendete Halcon-Algorithmus liefert bei der

entsprechenen HSV-Raum Zerlegung in Ermangelung eines „nicht anwendbar“-Ergebnisses ersatzweise jedoch die Farbe „Rot“. Deshalb kam es im ersten Ansatz mit dieser Zuordnungsmethode zu vielen Fehlfortsetzungen in dunkle Randbereiche hinein, wenn der untere bzw. obere Schwellwert für die Farbsegmentierung zu wenig restriktiv gesetzt worden war. Mit der Bedingung, dass das Label sowohl darüber als auch darunter schon a priori zugeordnet worden sein musste, wurden diese Fehler nahezu vollständig vermieden. Dennoch bleibt ein Fehlerrisiko, beispielsweise können „Löcher“ im Objekt verschlossen werden. Deshalb wird dieser Schritt erst angewandt, wenn bereits eine möglichst dichte Karte aus sicheren Korrespondenzen erstellt wurde. In der Praxis hat sich gezeigt, dass Fehlfortsetzungen an vertikalen Sprungkanten jedoch sehr selten sind, da selbst eine zufällige Fortsetzung meist eine Lücke von mindestens einem Pixel aufweist. Außerdem würden auf diese Weise „zugeschmierte“ Löcher in der Rekonstruktion nicht wesentlich zu Fehlerkennungen beitragen, wie sich in Abschnitt 3.4 zeigt.

Als letzter Schritt der Labelzuordnung wird die Projektorkalibrierung explizit ausgenutzt. Wie bereits erklärt, kann allein durch Projektor und eine Kamera bereits ein Tiefenwert berechnet werden. Zuvor diente die Projektorkalibrierung auf diese Weise als Redundanz, um korrekte Zuordnungen zu überprüfen. Insbesondere wurde die Uneindeutigkeit der Korrespondenzen durch das doppelt verwendete Muster aufgelöst. Zur Erinnerung: Eine Sequenz aus vier Streifen kommt im Muster genau *zwei* mal vor. Durch die beschränkte Schärfentiefe können Objektpunkte aber nur in einem bestimmten Bereich vor der Kamera gefunden werden. In unserer Konfiguration hat sich herausgestellt, dass an jedem Ort im Bild mindestens *eine* der beiden möglichen Zuordnungen einer Sequenz ins Muster *außerhalb* dieses Bildbereichs liegt (siehe hierzu Abschnitt 3.6.2), d.h. die möglichen x-Positionen im Kamerabild für das linke bzw. rechte Label der gleichen Sequenz überlappen sich nicht. Die Uneindeutigkeit ist also effektiv aufgelöst. Für eine Verdreifachung oder Vervierfachung der Sequenz würde dies mit unserer Kamerakonfiguration nicht mehr gelten. Die doppelte Sequenz stellt deshalb ein Optimum dar.

Die Projektorkalibrierung kann nicht nur als Redundanz dienen, sondern auch zur Verdichtung der Rekonstruktion. Ist etwa ein Bildbereich in einer der beiden Kamerabilder durch einen Glanzfleck überblendet oder durch Selbstüberdeckung der Szene nicht sichtbar, gilt dies für das andere Kamerabild meist nicht. Unter Aufgabe der Redundanz kann die Projektorkalibrierung in diesen Bereichen als letzter Schritt der Rekonstruktion zur Verdichtung der Tiefenkarte verwendet werden. Abbildung 3.8 zeigt das Verbesserungspotential.

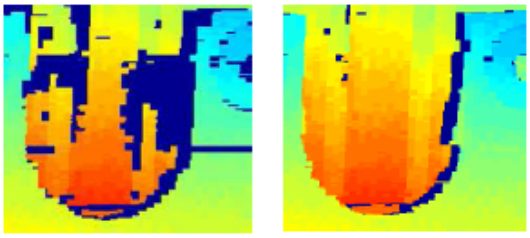


Abbildung 3.8: Verdichtung des Tiefenbildes durch Verwendung der Projektorkalibrierung. Links: ohne Projektorkalibrierung, rechts: mit Projektorkalibrierung

Allerdings ist diese Rekonstruktion mit Vorsicht zu betrachten, da sie nicht mehr durch unterschiedliche Verfahren überprüft werden kann. In Abbildung 3.8 rechtes Bild, sieht man beispielsweise am rechten Rand der Mulde einen kleinen Zacken, der im wirklichen Objekt nicht vorhanden ist. Sehr grobe Fehler können jedoch mittels der Einschränkung des möglichen Rekonstruktionsbereiches ausgeschlossen werden. Ferner kann man den Re-

konstruktionsbereich für eine zu vermessende Szene weiter einschränken (siehe Abschnitt 3.4), um die Güte zu erhöhen.

Auf diesen letzten Schritt kann verzichtet werden, wenn es stärker auf die Korrektheit, als auf die Dichte der Rekonstruktion ankommt. In unserer Anwendung wird der Schritt während des Modelltrainings zur Erstellung einer Triangulierung ausgelassen, während der Objektlageerkennung jedoch wieder verwendet.

Nun sind alle Label so weit wie möglich zugeordnet und das Tiefenbild wird erzeugt. Als Referenzkoordinatensystem dient das linke rektifizierte Kamerakoordinatensystem. Dies ermöglicht eine gute Überprüfung der Ergebnisse am Originalbild. Es hätte auch ein beliebiges anderes Koordinatensystem für die Tiefenwerte gewählt werden können. Die Rekonstuktion liefert maximal an jedem Streifenmittelpunkt einen Abstandswert, also kein dichtes Bild. Um die Tiefenbilder nicht zu verzerren, werden diese Lücken bei der Visualisierung aufgefüllt, so dass das Tiefenbild die gleiche Breite wie das rektifizierte Originalbild hat (in dieser Weise wurden die oben gezeigten Abbildungen erzeugt). Die Lücken werden mit konstanten Werten aufgefüllt, d.h. für jeden rekonstruierten Wert werden die drei davor und die drei danach liegenden Pixel mit dem gleichen Tiefenwert befüllt (da der Streifenabstand im Bild gerade sechs Pixel beträgt und der rekonstruierte Wert subpixelgenau ist). Man kann Zwischenräume auch linear interpolieren, was aber bei eckigen Bauteilen zu Verfremdung führt und Rechenzeit kostet und worauf deshalb in den hier gezeigten Bildern verzichtet wurde.

Die Darstellung der Tiefenbilder erfolgt dann über eine Falschfarbenkarte. Punkte, die nur von der rechten Projektorkalibrierung erkannt wurden, können im linken Tiefenbild hinter Kanten liegen. In diesem Fall werden sie nicht angezeigt (wird bei der Projektion überprüft), zur Objektlokalisation aber normal verwendet. Die meisten Abbildungen von Tiefenbildern in der vorliegenden Arbeit wurden mittels der kommerziellen Software Matlab der Firma MathWorks [Mat07] oder einem Halcon-Fenster (MVTec [Gmb80]) erzeugt, alle anderen verwenden das Visualisierungstool der Open Source-Bildverarbeitungsbibliothek OpenCV [ver09].

3.3.6 Speedup

In der industriellen Anwendung ist die Taktzeit der Objektlageerkennung von besonderer Bedeutung. Natürlich kann Geschwindigkeit durch Implementierung in Hardware erreicht werden, was allerdings aufwändig und unflexibel ist. Um trotzdem eine annehmbare Taktzeit zu erreichen, wurden einige softwaretechnische Verbesserungen vorgenommen.

Da einzelne Prozessoren nicht mehr ohne weiteres kleiner ausgeführt (Belichtung der Masken bei der IC-Herstellung bereits mit UV-Licht) und auch nicht sehr viel schneller getaktet werden können (Luftkühlungslimit), geht der Trend zur stärkeren Parallelisierung. In diesem Bereich bekommen die klassischen Prozessor-Konfigurationen (PC) hinsichtlich Rechenleistung Konkurrenz von Grafikkarten. Zum Zeitpunkt der Entstehung dieser Arbeit sind letztere jedoch noch aufwändig zu programmieren und durch die langsame Konvertierung der Bilder in grafikkartenfähige Datenpakete in vielen Anwendungsfällen ineffizient [Tra09].

Effizienter dagegen sind Vorverarbeitungen in Hardware. Einer der Flaschenhälse bei der Bildverarbeitung ist die riesige Datenmenge. Schon der Transfer der Bilder von der Kamera in den PC stellt bei schnellen Bildfolgen eine Herausforderung dar. Damit sind auch einfache Bearbeitungsalgorithmen äußerst rechenzeitintensiv. Deshalb muss das Ziel sein, die Datenmenge so schnell wie möglich auf die relevante Kerninformation zu reduzieren. Bei Kameraherstellern geht der Trend daher zu sogenannten „Smart Cameras“, bei denen eine Vorverarbeitung der Bilder direkt in der Kamera möglich ist und keine vollständigen

Bilder mehr an den PC übermittelt werden müssen. Nachteil dieser Entwicklung ist zum Zeitpunkt der Entstehung dieser Arbeit, ähnlich wie bei Grafikkarten, die aufwändige und vor allem unflexible Programmierung, welche aber voraussichtlich in den nächsten Jahren vereinfacht werden wird.

Für das hier geforderte Low-Cost-System bleibt allein aus Kamerakostengründen nur die klassische Variante ohne Hardwarebeschleunigung. Der Großteil der Tiefenbildberechnung wurde ebenso wie die Objektsegmentierung (siehe 3.4) zunächst in der Bildverarbeitungsskriptsprache Halcon geschrieben. Die Bibliothek ist sehr schnell in regionenbasierter Bildverarbeitung und die dort verwendeten Operatoren sind hoch optimiert. Die Kehrseite ist jedoch die langsame Ausführung selbstentwickelter Operationen. Beispielsweise sind einfache Schleifen dadurch sehr langsam, dass in der Skriptsprache nur komplexe Datentypen (sog. Tupel) verwendet werden können. Um trotzdem die notwendige Taktzeit zu erreichen, wurde schließlich ein möglichst großer Teil dieser eigenen Operationen in C++ programmiert.

Der zweite große Baustein zur Beschleunigung ist die Parallelisierung. Der naheliegende Ansatz dafür ist die parallele Verarbeitung des linken und rechten Kamerabildes. Es war also ein erklärtes Ziel, die beiden Bilder möglichst spät zusammenzuführen, um einen möglichst großen Teil parallel verarbeiten zu könnnen. Dafür muss die Software so geschrieben werden, dass zwei Instanzen gleichzeitig verarbeitet werden können (bspw Schutz von gemeinsam genutzten Ressourcen durch Semaphoren, sowie Vermeidung globaler Variablen).

Ferner ist die feinkörnige Parallelisierung einzelner Verarbeitungsschritte möglich und wurde in C++ mit Hilfe der Open Source Software OpenMP realisiert. Ein Beispiel hierfür ist die parallele Verarbeitung mehrerer Bildzeilen (großer Vorteil des gewählten Musters!).

Die dritte Maßnahme ist die möglichst frühe Reduktion der Datenmenge. So wird das Bild nach der Rektifizierung direkt nach Farbstreifen segmentiert. Aus einem 768 x 576 großen Bild wird hierbei eine 162 (Streifenanzahl im Muster) x 576 (Bildzeilen) große Matrix, auf der alle weiteren Verarbeitungschritte ausgeführt werden (genau genommen sind es zwei solche Matrizen, eine mit dem Farbcode (1,2 oder 3 für die jeweilige Farbe des Streifens an der Stelle) und eine mit der zugehörigen x-Position im Bild).

3.4 Umsetzung der Segmentierung und Groblageerkennung

Der in dieser Arbeit verwendete Feinlageerkennungalgorithmus gehört zur Klasse der ICP-Algorithmen (**I**terative **C**losest **P**oint - siehe Kapitel 3.5). Deren Hauptnachteil ist, dass sie typischerweise gegen ein *lokales* Minimum des Optimierungsproblems „Einpassen“ konvergieren. Damit diese Konvergenz trotzdem die gesuchte Objektlage ergibt, muss ein geeigneter Startwert vorgegeben werden, um sicherzustellen, dass es sich dabei um das *globale* Minimum handelt.

Dieser Abschnitt ist deshalb der Berechnung eines geeigneten Startwertes für die Feinlageerkennung gewidmet. Das Problem ist unterteilt in zwei Teilaufgaben, nämlich die Objektsegmentierung 3.4.1 und die Groblageerkennung. 3.4.2.

3.4.1 Umsetzung der Objektsegmentierung

Die Aufgabenstellung bei der Segmentierung ist die Trennung von Punkten auf dem gesuchten Objekt von Hintergrundpunkten. Dies ist im Allgemeinen nur durch Objekterkennung möglich, was zu einem Henne-Ei-Problem führt. Denn für eine sichere Objektlageerkennung mit unserem Algorithmus benötigt man eine gute Segmentierung.

Unsere Anwendung ist jedoch nicht ganz *allgemein*. Das übliche Szenario im industriellen Umfeld ist, dass ein Objekt in beliebiger oder auch eingeschränkter nicht genau bekannter

Lage von einem Tisch oder aus einem Nest abgegriffen werden soll. Da der Roboter genau über dem Nest positioniert werden kann, ist der *Hintergrund* bei jedem Abgriff nahezu gleich, lediglich die *Objektlage* ist unterschiedlich. Damit ist es vielleicht nicht möglich *Objektpunkte* zu erkennen, *Hintergrundpunkte* jedoch schon.

Hintergrundsegmentierung

Der erste hier vorgestellte Algorithmus macht sich dies zu Nutze. Außerdem verwendet er, dass der Hintergrund lokal meist gut durch eine Boundingbox (es wird nur ein quaderförmiger Raumteil betrachtet) ausgeschlossen werden kann und lokal eben ist.

Das Prozessnest (im einfachsten Fall ein ebener Tisch) wird während des Trainings einmal mit Objekt und einmal, aus gleicher Position P_A (A für Ausgangslage), ohne Objekt aufgenommen. Aus beiden Aufnahmen wird das Tiefenbild rekonstruiert. Im Tiefenbild *ohne Objekt* wird eine Ausgleichsebene aller Punkte ermittelt. Bei der Berechnung einer Regressionsebene minimaler Fehlerquadrate, schlagen wenige starke Ausreißer stark zu Buche. Dies kann bei Daten mit solchen Ausreißern zu mangelhaften Schätzungen und damit zu Fehlrekonstruktionen bei der Tiefenschätzung führen. Deshalb wird ein Verfahren benötigt, das robust gegen diese Art von Ausreißern ist, z.B. der RANSAC-Algorithmus [MAF81] der deshalb hier auch verwendet wird.

Anwendung findet dieser Algorithmus primär in der Bildverarbeitung, wenn ein Modell in eine verrauschte Datenmenge eingepasst werden soll. Der Algorithmus setzt sich aus folgenden Schritten zusammen:

1. Wähle aus der Datenmenge zufällig so viele Punkte aus, wie für die Schätzung des Modells unter der Annahme ausreißerfreier Daten notwendig sind.
2. Berechne die Modellparameter mit den gewählten Datenpunkten
3. Bestimme den Anteil der Messwerte C (*Consensus set*), deren Abstand zur Modellkurve kleiner als ein vorher festgelegter Grenzwert G ist. Speichere die Modellparameter, falls dieser Anteil *genügend* groß ist.
4. Wiederhole Schritte 1-3 mehrmals

Für die konkrete Anwendung sind hierbei der Grenzwert G für den Abstand der Datenpunkte von der Modellkurve, der nötige Schwellwert für die Speicherung eines Modellparametersatzes und eine maximale Anzahl von Iterationen festzulegen.

In unserer Anwendung, ist das Modell eine Ebene. Zu ihrer Schätzung werden drei Stützpunkte im Raum benötigt (Schritt 1 des Algorithmus). Kritisch ist die Wahl des Grenzwertes G. Wird er zu groß gewählt, gilt praktisch jede Ebene als richtig und das Ergebnis wird zufällig. Bei zu kleiner Wahl gilt entsprechend praktisch keine Ebene als richtig und das Ergebnis wird wieder zufällig. Empirisch hat sich ein Wert in der Größenordnung der Auflösung der Tiefenbildrekonstruktion als optimal herausgestellt. Hier wird $G = 1$ mm verwendet.

Damit der Algorithmus ein zufriedenstellendes Ergebnis liefert, muss die Ebene mindestens einmal ohne Ausreißer ermittelt werden. Dass mit s Datenpunkten mit relativem Anteil an Außreißern α bei n Wiederholungen in jeder Messung mindestens ein Ausreißer ist, liegt die Wahrscheinlichkeit bei

$$(1 - (1 - \alpha)^s)^n$$

Die Anzahl der Wiederholungen n muss mindestens so groß gewählt werden, dass diese Wahrscheinlichkeit höchstens $1 - p$ wird, wobei p die vorher festgelegte Wahrscheinlichkeit

einer Fehlrekonstruktion der Ausgleichsebene ist. Damit ergibt sich eine Mindestanzahl von

$$n_{min} = \frac{\log(1-p)}{\log(1-(1-\alpha)^s)}$$

Wiederholungen. Für $p = 99\%$ ist in Tabelle 3.1 die notwendige Anzahl der Iterationen in Abhängigkeit vom Ausreißeranteil α der Datenmenge aufgetragen.

Ausreißeranteil	10%	20%	30%	40%	50%	60%	70%
Anzahl erforderlicher Iterationen	4	7	11	19	35	70	169

Tabelle 3.1: Anzahl der notwendigen Iterationen bei $p = 99\%$ zur Ermittlung einer Ebene (3 erforderliche Punkte)

Im Allgemeinen kann man die hinreichende Anzahl der Datenpunkte mit kleinerem Abstand zur ermittelten Modellkurve (also das hinreichende *Consensus Set*) festlegen, um die Berechnung vorzeitig abzubrechen, wenn eine gute Ausgleichsebene gefunden wurde. Als Daumenwert kann laut Literatur [MAF81] bei k Datenpunkten $(1-\alpha)\cdot k$ gelten, also gerade so viele Punkte, wie es Nichtausreißer in der Datenmenge gibt. Da die Berechnung der Ausgleichsebene bei uns aber nicht zeitkritisch (während Training), mit geringer Laufzeit und zusätzlich gut parallelisierbar (in jedem Thread eine eigene Ausgleichsebene) ist, wurde eine feste Anzahl von Iterationen zwischen 100 und 1000 gewählt (je nach Messung). Dies führt bei einem geschätzten Ausreißeranteil von $< 10\%$ zu einer sehr geringen Fehlschätzungswahrscheinlichkeit. Selbst bei unkooperativen Oberflächen, bei denen die Ausreißerwahrscheinlichkeit deutlich erhöht ist, konnte während der Experimente (bei 100 Iterationen) keine Fehlschätzung festgestellt werden.

Wir haben gesehen, dass sich das Verfahren gut zum Einpassen einer Ebene in unseren Hintergrund eignet. Aber wie gut eignet sich eine Ebene prizipiell als Approximation für den Bauteilhintergrund? Zunächst muss dazu gesagt werden, dass mit unserem System nicht alle Punkte, die in beiden Kameras sichtbar sind, auch rekonstruiert werden. Der Bereich, in dem beide Kameras und auch der Projektor scharf sind, ist sehr eingeschränkt. Nur dort ist jedoch eine Tiefenrekonstruktion möglich. Je nach Sensorkopf sind dies ca. 5 cm (starke Vergrößerung) oder ca. 12 cm (schwache Vergrößerung) maximaler Schärfentiefebereich. Dieser Bereich wird auf das zu erkennende Objekt zentriert. Hintergrundpunkte, die weit davon abweichen, werden also gar nicht erst rekonstruiert. Des Weiteren müssen Strukturen mindestens 4 aufeinanderfolgende Streifen tragen, damit deren Tiefe rekonstruiert werden kann. Dies entspricht auf dem Objekt ca. $3 \cdot 1,3 \text{ mm} = 3,9 \text{ mm}$. Feinere Details sind unter Umständen ebenfalls nicht sichtbar (Sie sind nur sichtbar, wenn zusätzlich Streifen neben dem Objekt gefunden wurden). In dieser Art der Rekonstruktion hat sich die Ebene als in den allermeisten Fällen anwendbare Approximation des Hintergrundes herausgestellt.

Alle Punkte, die von der Kamera aus gesehen *hinter* der Ebene liegen, werden als Hintergrund markiert und nicht für die weitere Segmentierung und Objektlageerkennung verwendet. Zusätzlich können Punkte, die zu weit *vor* der Ebene liegen, ausgeblendet werden. Das eliminiert manche starke Ausreißer von der Tiefenrekonstruktion, ist in den meisten Fällen aber nicht nötig (siehe nächstes Kapitel).

In der Erkennungsphase wird der Roboter relativ zum Hintergrund wieder auf die gleiche Position gefahren. Der Ort der Ebene wurde gespeichert, so dass ohne weiteren Rechenaufwand segmentiert werden kann. Bei Hintergründen, die einer Ebene sehr nahe kommen, genügt diese Segmentierung und es kann direkt die Groblageerkennung durchgeführt werden.

Bei komplexeren Ablagenestern kann anstatt einer Ausgleichebene auch das rekonstruierte Tiefenprofil direkt genommen und alle davon erhabenen Punkte als Objektpunkte definiert

werden. In diesem Fall spielt aber die Feinpositionierung des Roboters und die dichte Rekonstruktion des Hintergrundes eine maßgebliche Rolle. Während erstere im Rahmen liegt (Wiederholgenauigkeit für Stäubli-Roboter ca. 0,3mm), ist zweiteres bei unserem Sensorsystem kritisch. Die gegebenen Einsatzfälle machten diese Art der Anwendung aber bisher noch nicht nötig.

Shape Based Matching

Die Erkennung des Hintergrundes anstatt des Objektes zur Segmentierung, wie sie im letzten Abschnitt beschrieben wurde, ist zwar auf Grund der immer gleichen Positionierung des Sensors gegenüber dem Hintergrund sinnvoll, birgt aber Probleme. Neben schleichenden (langsame Dekalibrierung) und auch schnellen (Veränderung der Position von Roboter oder Hintergrundobjekten) Veränderungen des Hintergrundes, ist das Hauptproblem, dass mittels der Hintergrunderkennungsmethode vom Modell abweichende Objekte nicht mehr erkannt werden können. Die Aufgabenstellung schließt zwar nur die Objektlageerkennung jeweils eines Bauteiltypes ein, eine gute Erweiterung und gleichzeitig Verbesserung der Robustheit gegenüber Fehlern wäre jedoch, zumindest grob zu überprüfen, ob das segmentierte Objekt vom gesuchten Typ ist.

Ein Standardverfahren für eine solche prüfende Objekterkennung ist die Umrißerkennung, das sogenannte *Shape Based Matching*. Im Rahmen dieser Arbeit wird der gleichnamige Operator der Halcon Bibliothek [Gmb80] verwendet. Im Training wird der Umriss des Objektes aus der gewählten Ansicht gespeichert. Hierfür verwendet man das oben erläuterte Verfahren, um das Objekt von dem sehr kontrollierten Hintergrund einer Einlernplatte (perfekte Ebene) zu trennen. Der Operator muss im Training dann lediglich die Kante der verbleibenden Punkte im Bild zusammensetzen. Dabei wird geglättet, um Rekonstruktionslücken an der Kante auszugleichen. Die so gewonnene Schablone wird im Objekterkennungszyklus dann im Bild gesucht. Alle Punkte, die sich innerhalb des besten Matches befinden, werden dann als Objektpunkte segmentiert. Hierbei werden leichte Veränderungen der Form und Skalierung zugelassen. Unterschreitet der beste gefundene Match eine Güteschwelle (siehe Messungen in Kapitel 4.1.2), so wird der Fehler „Objekt nicht gefunden“ ausgegeben.

Empirisch konnte eine gute Erkennungsquote mit einer Verkippung um 10-30 Grad (je nach Objekt) der Objektlage zur eingelernten Objektlage erreicht werden. Dies genügt für unsere Anwendung. Um den möglichen Verkippungsbereich zu vergrößern, können auch mehrere Objektansichten eingelernt und simultan überprüft werden. Es wird dann nur das beste Ergebnis verwendet. Dies führt allerdings mit n der Anzahl der eingelernten Ansichten zu einer Laufzeitverlängerung um nahezu den Faktor n, da der Operator an sich schon im Falle nur einer Objektansicht mehrere Prozessorkerne ansteuern kann und die Parallelausführung damit zu keiner starken Beschleunigung führen würde.

3.4.2 Umsetzung der Groblageerkennung

Wie in Abschnitt 3.4 gesagt, wird zur Erzielung der Konvergenz des ICP-Verfahrens auf das „richtige“ Minimum eine Groblageerkennung benötigt.

Bei der Studie möglicher Algorithmen zur Objektlageerkennung findet man eine ganz einfache Möglichkeit. Ziel ist es, die große Datenmenge einer Punktewolke möglichst schnell auf einfache Merkmale zu reduzieren, um die Rechenzeit annehmbar zu halten. Diese Merkmale sollten möglichst wenig unwichtige Information enthalten. Wir suchen also gerade den Teil der Information in der Punktewolke, der die Lage enthält. Die Lage im Raum wird durch Bestimmung von sechs Freiheitgraden festgelegt. Den Schwerpunkt (drei translatorische Freiheitsgrade) und die Ausrichtung der Hauptachsen (drei rotatorische Freiheitsgrade). Warum bestimmen wir diese nicht direkt?

Dies ist möglich. Der Schwerpunkt $\vec{x_s}$ der diskreten Punktewolke $\left\{\vec{x_i} \mid i = 1 \dots n\right\}$ (einfach jeder Messpunkt des Tiefenbildes in den 3D-Raum projiziert) lässt sich über die Formel

$$\vec{x_s} = \frac{1}{M} \sum_{i=1}^{n} \vec{x_i} \cdot m_i$$

berechnen, wobei M die Gesamtmasse und m_i die Einzelmassen sind. Gewichtet man alle Punkte gleich stark $m_i = 1 \forall i$, so erhält man $M =$ Anzahl der Punkte. Die Hauptachsen können, wie in Kapitel 2.4.1 erwähnt, mittels einer Hauptachsenzerlegung der Kovarianzmatrix C der Punktewolke bestimmt werden.

Sei $\vec{x_i} = (x_1^i, x_2^i, x_3^i)^T \in \mathbb{R}^3$ ein Punkt der Objektpunktewolke und $\vec{x_s} = (x_1^s, x_2^s, x_3^s)^T \in \mathbb{R}^3$ deren Schwerpunkt, wie oben ermittelt, so gilt für die schwerpunktbereinigte Kovarianzmatrix

$$C := \frac{1}{M} \sum_{i=1}^{n} \begin{pmatrix} (x_1^i - x_1^s)^2 & (x_1^i - x_1^s) \cdot (x_2^i - x_2^s) & (x_1^i - x_1^s) \cdot (x_3^i - x_3^s) \\ (x_2^i - x_2^s) \cdot (x_1^i - x_1^s) & (x_2^i - x_2^s)^2 & (x_2^i - x_2^s) \cdot (x_3^i - x_3^s) \\ (x_3^i - x_3^s) \cdot (x_1^i - x_1^s) & (x_3^i - x_3^s) \cdot (x_2^i - x_2^s) & (x_3^i - x_3^s)^2 \end{pmatrix}$$

Die Eigenvektoren von C sind gerade die Hauptachsen der Punktewolke, daher auch der Name des Verfahrens (**P**rincipal **C**omponent **A**nalysis). Bei der Berechnung der Eigenvektoren muss beachtet werden, dass das Standard-Gauss-Verfahren nicht numerisch stabil ist. Es ist zwar davon auszugehen, dass die Matrix nicht entartet ist (das hieße gerade, dass die Punkte der Punktewolke exakt auf einer Ebene (Rang 2) bzw. einer Geraden (Rang 1) liegen müssten), aber durch den Einfluss der Maschinengenauigkeit bei der Berechnung können nahezu singuläre Matrizen zu Problemen führen. Deshalb eignet sich nur ein Verfahren, das auch für Matrizen mit nicht vollem Rang anwendbar ist. Das übliche Verfahren hierfür ist die Singulärwertzerlegung (**S**ingular **V**alue **D**ecomposition), in diesem Anwendungsfall auch *Karhunen-Loève-Transformation* genannt. Hierbei wird die bei uns reellwertige, quadratische Matrix in drei Matrizen zerlegt

$$C = UDV^T$$

wobei $U, V \in O(3 \times 3)$ orthonormale Matrizen sind und D eine Diagonalmatrix mit den nach Größe sortierten Singulärwerten $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ von C auf der Hauptdiagonalen beginnend mit dem größten, wobei ein Wert von 0 nicht als Singulärwert bezeichnet wird. Ist C diagonalisierbar, so entsprechen die Singulärwerte gerade den Eigenwerten von C. Ferner enthält V die Eigenvektoren von C als Spaltenvektoren.

In den 60er Jahren entwickelten Gene Golub, William Kahan und Christian Reinsch direkte, iterative und damit numerisch stabile Algorithmen zur Berechnung einer Singulärwertzerlegung, welche das Problem mittels orthogonaler Transformationen auf eine QR-Zerlegung zurückführen. In dieser Arbeit wird die Implementierung der offenen Bildverarbeitungsbibliothek OpenCV verwendet [ver09].

Wir sind in erster Linie an den Eigenvektoren interessiert. Ist die Singulärwertzerlegung von der Modellpunktewolke und der Punktewolke zum Zeitpunkt der Objektlageschätzung bekannt, so ist die gesuchte Objektlage gerade die Transformation, die den Schwerpunkt der beiden Punktewolken und deren Hauptachsen in Reihenfolge der Größe der Eigenwerte ineinander überführt. Der Translationsvektor $\vec{t}$ für die Schwerpunktverschiebung lässt sich elementar berechnen. Mit den Schwerpunkten $\vec{t_m}$ und $\vec{t_o}$ von Modell respektive Objekt gilt:

$$\vec{t} = \vec{t_m} - \vec{t_o}$$

Die einfachste Repräsentation der Rotationslage ist ein Koordinatensystem, dessen Achsen gerade den Hauptachsen des Objekts entsprechen und dessen Ursprung gerade im Schwerpunkt des Objekts liegt. Die Rotationsmatrix der Transformation entspricht gerade dem Basiswechsel vom einen zum anderen solchen Schwerpunktsystem. Seien $M = \left\{\vec{m_1}, \vec{m_1}, \vec{m_1}\right\}$ die Eigenvektoren der Modellpunktewolke und $O = \left\{\vec{o_1}, \vec{o_1}, \vec{o_1}\right\}$ die Basisvektoren der Objektpunktewolke, transformiert in die selbe Basis, so erhält man die gesuchte Transformation durch Darstellung der Vektoren $\vec{o_j}$ als Linearkombination der Vektoren $\vec{m_i}$.

$$\vec{o_j} = a_{1j}\,\vec{m_1} + a_{2j}\,\vec{m_2} + a_{3j}\,\vec{m_3} \qquad j \in \{1,2,3\}$$

Die Koeffizienten $a_{ij} \in \mathbb{R}$ bilden die j-te Spalte der gesuchten Transformationsmatrix

$$T_M^O = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Prinzipiell können mit dieser Methode nur die Hauptachsen des Modells auf die entsprechenden Hauptachsen der Objekts gedreht werden. Dafür gibt es jedoch je Achse *zwei* Möglichkeiten! Nehmen wir als Beispiel die erste Hauptachse. Das ist die Achse zum größten Eigenwert der Kovarianzmatrix, also die Hauptachse entlang derer die Punktewolke am stärksten ausgebreitet ist. Diese ist, bei einem Blickwinkel lotrecht auf die Hintergrundebene, bei den meisten Objekten in etwa in der Bildebene. Um nun Modell- und Objektachse ineinander zu überführen, muss um eine Achse senkrecht dazu (z.B. die 3. Hauptachse, die meist in etwa lotrecht zur Hintergrundebene liegt) um maximal 180 Grad gedreht werden. Die Frage ist aber: In welche Richtung?

Von den drei Freiheitsgraden können wir den ersten bestimmen, indem wir festlegen, dass das Objekt nicht verkehrt herum auf dem Tisch liegen darf (wir also nicht nur die Rückseite sehen). Damit wäre in obigem Beispiel die 3. Hauptachse definiert. Es wird festgelegt, dass die Achse in Richtung Kamera zeigt (und nicht von der Kamera weg). Die zweite Hauptachse ist durch die Festlegung bestimmt, dass das Koordinatensystem des Objekts stets rechtshändig sein soll (dies wird nach der Singulärwertzerlegung überprüft und ein Achsenvektor durch seinen negativen Vektor ersetzt falls notwendig). Bleibt also die 1. Hauptachse, die in unserem Beispiel die Rotation des Objekts in etwa in der Kameraebene darstellt. Hierfür wirde die Segmentierung mittels Shape Based Matching verwendet. Der Umriss des Modells ist im Allgemeinen nämlich nicht rotationssymmetrisch. Der Rotationswinkel kann also bestimmt werden und die Zweideutigkeit der Rotationsrichtung ist damit aufgelöst. Der Grobfit ist komplett.

Betrachten wir Objekte mit Symmetrien. Häufig kommen n-zählige Symmetrien (meist $n = 2$, d.h. Drehung um 180 Grad, aber auch $n = 4$, also ein Quadrat, oder $n = 6$, z.B. eine Schraubenmutter, sind denkbar) oder Rotationssymmetrien (könnte man als n-Symmetrie mit $n = \infty$ bezeichnen) um eine der Hauptachsen vor. Diese können einfach durch Einschränkung des Rotationswinkels bei der Auflösung der Zweideutigkeit wie im letzten Absatz beschrieben gelöst werden. Bei einer 2-zähligen Symmetrie wird beispielsweise dann niemals mehr als um den Winkel 90 Grad gedreht. Es wird also die am nächsten liegende, passende Rotationslage gewählt.

Bei einer $n = \infty$-Symmetrie wird ein schnelleres Ausgleichsverfahren verwendet, als die oben beschriebene Basistransformation mit anschließenden Achsenflips. Stattdessen berechnet man die Rotation, die die beiden Hauptachsen, um die die Symmetrie vorliegt, auf kürzestem Weg übereinanderlegt. Seien $\vec{x_m}, \vec{x_o} \in \mathbb{R}^3$ die Hauptachsen des Modells bzw. des Objektes, um die das Objekt rotationssymmetrisch ist. Dann wird zunächst die Achse $\vec{r}$ berechnet, um die gedreht werden soll, mit dem Kreuzprodukt:

$$\vec{r} = \vec{x_m} \times \vec{x_o}$$

Der Rotationswinkel α ist gerade gleich dem kleineren der beiden Winkel zwischen $\vec{x_m}$ und $\vec{x_o}$ in der von ihnen aufgespannten Ebene (mit Normalenvektor $\vec{r}$):

$$\widetilde{\alpha} = \angle\left(\vec{x_m}, \vec{x_o}\right) = \arccos\left(\frac{\vec{x_m} \cdot \vec{x_o}}{\left|\vec{x_m}\right| \left|\vec{x_o}\right|}\right)$$

mit $\widetilde{\alpha} \in [0, \pi]$ und damit gilt für den gesuchten Winkel

$$\alpha = \min\left\{\widetilde{\alpha}, \pi - \widetilde{\alpha}\right\}$$

Die Symmetrie kann vom Benutzer im Vorfeld angegeben werden, ist aber auch teilweise automatisch bestimmbar. Hierfür werden die Eigenwerte zu den Eigenvektoren von C verwendet. Sind zwei Eigenwerte praktisch gleich, liegt wahrscheinlich eine Rotationssymmetrie um die Achse vor, die zu dem unterschiedlichen Eigenwert gehört. Liegt eine n-zählige Symmetrie vor, die nicht erkannt wurde, führt die nicht zu Fehlschätzungen bei der Objektlageerkennung, sondern lediglich zu längeren Fahrwegen des Roboters (er gleicht eine Rotationslage aus, die er eigentlich gar nicht ausgleichen müsste).

Verfahrensinhärente Fehler können auftreten und werden im nächsten Kapitel 4.3 genau analysiert. Trotz aller Schwächen wird sich im nächsten Kapitel jedoch herausstellen, dass dieses Verfahren äußerst rechenzeiteffizient schon sehr gute Objektlageschätzungen ergibt.

Exkurs: Quaternionen

Für Roboterpositionen und Ausrichtungen und für die gesuchten Transformationsmatrizen zur Objektlagebestimmung müssen Matrizenrechnungen effizient ausgeführt werden. Numerisch ist eine elementar ausgeführte Matrixmultiplikation (also tatsächliche Multiplikation der Komponenten) aber fehleranfällig. Die Maschinengenauigkeit schlägt sich bei jeder Multiplikation in jedem Koeffizienten nieder. Die meisten hier verwendeten Matrizen sind 3×3-Rotationsmatrizen, also in der *speziellen orthogonalen Gruppe*

$$\mathrm{SO}(3) = \{M \in \mathrm{O}(3) \subset \mathrm{Mat}(3 \times 3) | \det M = 1\}$$

mit O(3) der orthogonalen Gruppe der 3×3-Matrizen, deren Spaltenvektoren eine Orthonormalbasis bilden (also orthogonal stehen und Länge 1 haben). Die Verknüpfung auf der Gruppe ist gerade die Matrixmultiplikation. Da es sich um eine Gruppe handelt, erhalten wir bei einer Multiplikation zweier Matrizen aus SO(3) wieder eine Matrix aus SO(3). Durch die beschränkte Maschinengenauigkeit ist dies jedoch nicht mehr der Fall, d.h. die Ergebnismatrix M hat nicht mehr $\det(M) = 1$. Sie muss im Nachhinein wieder normalisiert werden, aber wie?

Wir suchen also nach einer numerisch geschickteren Repräsentation von Drehungen im Raum. In der Bildverarbeitung sind hierfür *Quaternionen* üblich. [Kui02] Dabei handelt es sich um ein von W.R. Hamilton im Jahre 1843 erdachtes Zahlensystem $\mathbb{H}$, das ähnlich wie die Komplexen Zahlen, eine Erweiterung von $\mathbb{R}$ ist, wobei es im Gegensatz zu ersterem keinen Körper mehr bildet. Ein Quaternion $q \in \mathbb{H}$ besteht auch nicht aus einem Zweier-, sondern aus einem Vierertupel reeller Zahlen. Es lässt sich eindeutig in der Form

$$q = w + x \cdot i + y \cdot j + z \cdot k$$

schreiben, wobei $w, x, y, z \in \mathbb{R}$ reelle Zahlen sind und für i, j, k gilt:

$$i^2 = j^2 = k^2 = i \cdot j \cdot k = -1$$

Quaternionen bilden keinen Körper, da die Kommutativität verletzt ist:

$$q_1 \cdot q_2 \neq q_2 \cdot q_1$$

für $q_1, q_2 \in \mathbb{H}$. Allerdings ist $\mathbb{H}$ eine Gruppe bezüglich der Addition und es gilt das Assoziativ- und das Distributivgesetz. Ferner gibt es ein neutrales Element der Multiplikation (nämlich $q = 1$) und es gibt für jedes $q \in \mathbb{H}$ ein Inverses Element der Multiplikation $q^{-1} \in \mathbb{H}$ mit

$$q \cdot q^{-1} = q^{-1} \cdot q = 1$$

(linksseitiges Inverses gleich rechtsseitiges Inverses). Das macht die Quaternionen zu einem *Schiefkörper*.

Betrachten wir die Abbildung durch Konjugation in $\mathbb{H}$. Mit Einheitsquaternionen $q, x \in \widehat{\mathbb{H}} = \{q \in \mathbb{H} : \|q\| = 1\}$ ist die Abbildung

$$\rho_q : x \mapsto qx\overline{q}$$

gerade eine Drehung mit Drehachse im Ursprung um den Winkel $0 < \alpha < 2\pi$, wenn man q wie folgt definiert

$$q = \cos\frac{\alpha}{2} + v \cdot \sin\frac{\alpha}{2}$$

wobei $v \in \{i, j, k\}$ ist. q ist damit ein Einheitsquaternion und jedes Einheitsquaternion kann so dargestellt werden. Bei ρ_q und ρ_{-q} handelt es sich um dieselbe Drehung, so dass es für jede Drehung genau *zwei* entsprechende Einheitsquaternionen gibt.

Die Multiplikation entspricht gerade der Hintereinanderausführung von Drehungen:

$$\rho_{q_1} \cdot \rho_{q_2} = \rho_{q_1 \cdot q_2}$$

und die komplexe Konjugation entspricht gerade der Umkehrung der Drehrichtung:

$$\rho_{\overline{q}} = \rho_q^{-1}$$

Die umgekehrte Transformation eines Einheitsquaternions

$$q = w + x \cdot i + y \cdot j + z \cdot k, \qquad w^2 + x^2 + y^2 + z^2 = 1$$

in eine orthogonale Matrix ist gegeben durch:

$$\begin{pmatrix} w^2 + x^2 - y^2 - z^2 & -2wz + 2xy & 2wy + 2xz \\ 2wz + 2xy & w^2 - x^2 + y^2 - z^2 & -2wx + 2yz \\ -2wy + 2xz & 2wx + 2yz & w^2 - x^2 - y^2 + z^2 \end{pmatrix}$$

Damit kann man, statt Drehmatrizen miteinander zu multiplizieren, beide in die entsprechenden Einheitsquaternionen umrechnen, auf dem Raum der Quaternionen die Multiplikation ausführen und das Ergebnis zurücktransformieren. Das ist bei langen Rechnungen laufzeiteffizienter, aber noch wichtiger ist, dass sich numerische Rechenfehler nur noch auf vier Komponenten und nicht auf 9 Komponenten addieren. Zusätzlich lässt sich ein Ergebnisquaternion, das nicht mehr exakt Länge 1 hat, einfach normieren. Die Antwort also, welche Drehung nahe des fehlerbehafteten Ergebnisses verwendet werden soll, ist trivial.

In der vorliegenden Arbeit wird die Implementierung der Quaternionenrechnungen aus der Dissertation von F. Haug [Hau11] verwendet. Praktisch sämtliche Drehmatrizenrechnungen, werden mittels Quaternionen ausgeführt und die Ergebnisse am Schluss normiert.

3.5 Umsetzung der Feinlageerkennung

„Der ICP-Algorithmus (...) ist die dominante Methode zur Registrierung von dreidimensionalen Modellen, die nur auf Geometrie und manchmal Farbe und Gitternetzen, beruhen“ (Rusinkiewicz [SR01]). Auch in dieser Arbeit wird zur Feinlageerkennung eine Variante des ICP-Algorithmus (Iterative Closest Points, oder später auch Iterative Corresponding Points) verwendet. Die genannte Quelle gibt einen guten Überblick über übliche Varianten. Das zu Grunde liegende Optimierungsproblem (Kapitel 2.5.1) und die allgemeine Form der Lösung mittels ICP (Kapitel 2.5.2) wurden im Grundlagenkapitel ausführlich besprochen.

In dieser Arbeit wurde eine Abwandlung der ICP-Variante von Chen [YC91], auch *Chens-Algorithmus* genannt, verwendet. Hierbei wird im Gegensatz zu anderen ICP-Algorithmen nicht eine Punktewolke (Modell) gegen eine Punktewolke (Objekt) registriert, sondern es wird eine Punkt-zu-Tangentialebene Registrierung verwendet. Das Modell wird also aus Flächen zusammengesetzt. Mit diesen Korrespondenzen wird die Berechnung der Transformation von Modell zu Objekt analog zum ICP mittels Minimierung der Fehlerquadrate berechnet.

Es ist strittig, ob es sich bei Chens-Algortihmus um einen eigenständigen Ansatz oder lediglich um eine ICP-Variante handelt. [JS06] weisen darauf hin, dass die Methode bereits zeitgleich mit dem ICP entwickelt wurde und die Korrespondenzbildung zwischen Modell und Objekt einer der Kernbestandteile des Algorithmus seien, Chens Algorithmus damit also als eigenständig betrachtet werden müsse.

Grundsätzlich ist Chens Algorithmus im einzelnen Iterationsschritt genauer, da der nächste Punkt auf der ganzen Oberfläche des durch Flächen approximierten Modells gesucht wird und nicht nur einzelne Oberflächenpunkte (Punktewolke) verwendet werden. Der Rechenaufwand ist dadurch erhöht, aber die Konvergenz ist schneller.

Um aus einer Punktewolke ein flächenhaftes Modell zu berechnen, gibt es mehrere Möglichkeiten. Eine davon, die Triangulierung, wird in Abschnitt 2.7.2 vorgestellt. Am meisten Rechenzeit wird hierbei mit der Nachbarschaftssuche verbraucht. Um eine lokale Tangentialebene (bzw. deren Normalenvektor n_p) zu berechnen, benötigt man nämlich neben dem Fußpunkt p alle Modellpunkte *nahe* diesem Punkt. Hat man die Punktewolke jedoch als ein einziges Tiefenbild gegeben, so kann die Nachbarschaft aus diesem abgeleitet werden. Zur Normalenberechnung werden schlicht die *im Bild* benachbarten Punkte verwendet. Diese Methode ist so schnell, dass Normalenvektoren sogar in der Objektpunktewolke berechnet werden können, um die Registrierung robuster zu machen.

In dieser Arbeit wurde der Chen-Algorithmus mit einer robusten Korrespondenzpunktsuche in der Umsetzung der Firma Aqsense [Aqs] verwendet (siehe Abbildung 3.9). Benötigt werden hierfür jeweils ein Modell- und ein Objekttiefenbild sowie der Startpunkt. Es kann entweder die gewünschte Genauigkeit oder die Anzahl der Iterationen festgelegt werden. In unserem Algorithmus wurde zweitere Option gewählt. Die hierfür durchgeführten Messungen finden sich in Kapitel 4.2.2.

Unsere Tiefenbilder haben die Besonderheit, dass selbst bei einer perfekten Rekonstuktion nicht an jedem Bildpunkt ein Messpunkt vorhanden ist. Vielmehr sind die Punkte entlang der Musterstreifen (vertikal im Bild) dicht, quer dazu (horizontal im Bild) wird aber nur ein Punkt pro Musterstreifen, also nur etwa jeder sechste Bildpunkt rekonstruiert. Als Nachbarn für die Normalenvektorberechnung werden daher in horizontaler Richtung nicht die nächsten Bildpunkte, sondern die Punkte des nächsten *Musterstreifens* verwendet.

Neben der Nachbarschaftsbeziehung der Punkte untereinander und ihrem Abstand von der Kamera, wird auch deren tatsächliche Lage im Raum benötigt. Da die Nachbarschaft aber aus dem Tiefenbild entnommen werden soll, ist die Datenspeicherung als Punktewolke (als aufeinanderfolgende (x, y, z)-Koordinaten der Einzelpunkte) ineffizient. Besser ist

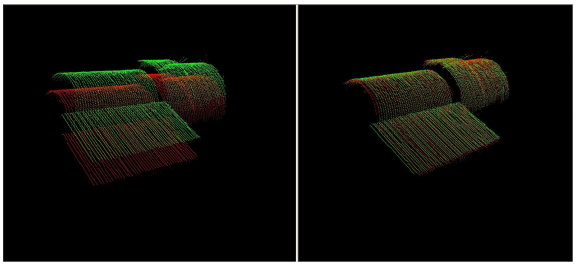


Abbildung 3.9: Modell- (rot) und Objektpunktewolke (grün) eines Messkörpers vor (l.) und nach (r.) der Registrierung durch Feinfit mittels Chens-Algorithmus. (Grafik und Registrierung erstellt mit Hilfe der Software der Firma Aqsense)

die Speicherung in Form von *drei „Bildern“*, für jede Koordinate eines. Die drei Koordinaten eines Punktes sind dann jeweils an der gleichen Stelle der drei Bilder gespeichert. Dadurch bleibt die Nachbarschaftsinformation durch den Speicherort im Bild erhalten und die Weiterverarbeitung ist effizient möglich. Als Höhe der Bilder wird die Kamerabildhöhe verwendet, als Breite jedoch die Anzahl der Musterstreifen (162), da dies die maximale Anzahl rekonstruierter Punkte ist.

3.6 Umsetzung der Kalibrierung

In diesem Abschnitt soll kurz erklärt werden, wie die Kalibrierungen bei uns umgesetzt wurden.

3.6.1 Kamerakalibrierung

Zur Ansteuerung der Kameras wird die Bildverarbeitungssoftware Halcon (Version 9.0) verwendet. Die hierin enthaltenen Operatoren zur Kamerakalibrierung sind in zwei Schritte geteilt. Zunächst werden die *intrinsischen Kameraparameter* geschätzt. Dabei wird eine *Kalibrierplatte* verwendet, die als „markante Punkte“ ein quadratisches Muster aus x mal x schwarz ausgefüllten Kreisen enthält, das von einem schwarzen Quadrat mit einer Markierung an einer Ecke umgeben ist. Das Quadrat dient dem Algorithmus zur Erkennung, in welchem Bereich die markanten Punkte gesucht werden sollen. Die Markierung dient hierbei der Rotationsausrichtung. Die Platte wurde exakt vermessen, so dass die Mittelpunkte der Kreise auf einer Mikrometerskala bekannt sind. Die Kalibrierplatte wird nun in unterschiedlichem Abstand und in unterschiedlicher Verkippung und Position vor die Kamera gehalten und je ein Bild aufgenommen. In jedem dieser Bilder wird die Position aller Mittelpunkte der schwarzen Kreise bestimmt. Aus diesen Punkten und der bekannten Geometrie der Platte werden mittels eines *Bündelausgleichverfahrens* die internen Kameraparameter ermittelt (also auf welche Weise die Optik und die Kamera das real aufgenommene Bild verzerren) [Gmb80]. Der wichtigste Parameter ist dabei die Brennweite f der Gesamtoptik. Aus diesem Grund ist es nicht möglich, die Kamera nach der Kalibrierung auf einen anderen Punkt scharf zu stellen, ohne das Ergebins wesentlich zu verfälschen. Dies ist der Grund weswegen Fixfokuskameras verwendet werden.

Nach der Kalibrierung der Einzelkameras werden die Kameras zueinander und zum Roboter kalibriert (*extrinsische Kamerakalibrierung*). Hierfür werden verschiedene Posen über einer Kalibrierplatte angefahren und wieder jeweils die Position der „Markanten Punkte“ (Kreismittelpunkte) ermittelt. Da zusätzlich die Roboterposition bekannt ist, kann nach einigen Iterationen die Position der Kamera relativ zur Basispose der Roboters bestimmt werden [Gmb80] (hand_eye_calibration)

3.6.2 Projektorkalibrierung

Der verwendete Projektor kann nur durch Umbau unterschiedliche Muster projizieren. Ein solcher Umbau verändert Projektorparameter leicht, so dass hinterher neu kalibriert werden müsste. Daher eignen sich Kalibrierungsverfahren, die auf der Projektion eines speziellen Kalibriermusters beruhen, nicht für unseren Aufbau.

Auch das von Matabosch beschriebene Verfahren für Streifenmuster (siehe Abschnitt 2.6.3) ist so nicht anwendbar, da unser projiziertes Muster das ganze Bildfeld abdeckt.

Die Halcon Bibliothek enthält keinen Algorithmus zur Projektorkalibrierung. Deshalb wurde das Projektorkalibrierungsverfahren in einer von mir mitbetreuten Diplomarbeit entwickelt [Dos09] und später verfeinert.

3.6.2.1 Herleitung der Transformationsfunktion

Abbildung 3.10 zeigt eine schematische Darstellung des Stereokamerasystems. Wir möchten einen Zusammenhang zwischen der Disparität (Verschiebung von linkem zu rechtem Kamerabild) eines Objektpunktes und dessen Abstand von dem Kamerasystem herleiten.

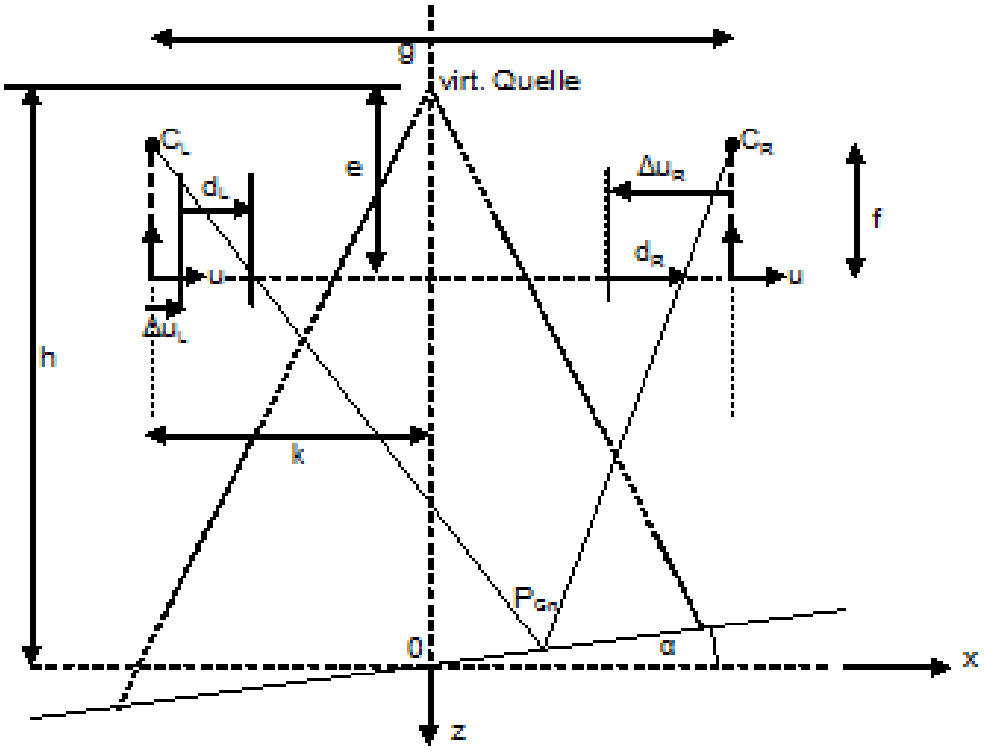


Abbildung 3.10: Schematische Darstellung eines Stereokamerasystems mit linker Kamera C_L, rechter Kamera C_R und Projektor „*virtuelle Quelle*“

Gehen wir von einem Stereokamerasystem (linke Kamera C_L und rechte Kamera C_R mit Basisweite g) im Stereonormalfall aus. Dies wird durch die Kalibrierung und Rektifizierung erreicht. In der Mitte der beiden Kameras befinde sich der Projektor mit der *virtuellen Projektionsquelle*. Betrachte ferner den Fall, in dem die rektifizierte Bildebene der Kameras parallel zur Grundebene liegt, auf die das Muster projiziert werden soll (Winkel $\alpha = 0$). Die Ursprünge der Kamerakoordinatensysteme (u,v) seien in der Bildebene, die sich im Abstand der Brennweite f vor den Zentralpunkten C_L und C_R der Kameras befindet. Ihre Ursprünge sind gegenüber dem Lot der Kamerazentralpunkte auf die Bildeben um Δu_L bzw. Δu_R verschoben. Des Weiteren sei der Ursprung unseres Koordinatensystems

(x,y,z) auf der optischen Achse der virtuellen Quelle des Projektors auf der beleuchteten Ebene und die x-Achse parallel zur u Achse der Kamerakoordinatensysteme, sowie die y-Achse parallel zu deren v-Achsen. Der Abstand vom Projektor zur Grundebene sei h. Der Abstand vom Projektor zu den Kamerazentralpunkten sei e.

Betrachten wir einen Objektpunkt P_{Gn}, der sich auf der beleuchteten Grundebene im Bildfeld beider Kameras befindet. Das Bild erscheint in der linken Kamera auf der u-Achse um d_L gegenüber dem Ursprung des linken Kamerakorrdinatensystems verschoben (entsprechend um d_R verschoben gegenüber dem rechten Kamerakoordinatensystem). Es gilt:

$$P_{Gn} = \begin{pmatrix} x \\ y \end{pmatrix}, C_L = \begin{pmatrix} -\frac{g}{2} \\ -(h-e)-f \end{pmatrix}, C_R = \begin{pmatrix} \frac{g}{2} \\ -(h-e)-f \end{pmatrix}$$

Weiter gilt für die linke Kamera C_L nach dem Strahlensatz:

$$\frac{x+k}{\Delta u_L + d_L} = \frac{h-e+f-y}{f}$$

aufgelöst nach d_L:

$$d_L = f\frac{x+k}{h-e+f-y} - \Delta u_L = f\frac{\frac{hx_n}{f_2 - x_n \tan\alpha} + k}{h-e+f-\frac{hx_n}{\frac{f_2}{\tan\alpha}-x_n}}$$

Im Falle $\alpha = 0$ gilt dann für d_L und analog für d_R:

$$d_L = f\frac{k+\frac{hx_n}{f_2}}{h-e+f} - \Delta u_L \text{ und } d_R = -f\frac{g-k-\frac{hx_n}{f_2}}{h-e+f} + \Delta u_R$$

womit für die Disparität gilt:

$$\text{disp} = d_R - d_L = \Delta u_R + \Delta u_L - f\left(\frac{g}{h-e+f}\right)$$

Damit gelten folgende Zusammenhänge:

- Die Pixelabstände d_L und d_R sind bei konstantem Abstand h proportional zu x_n
- Die Disparität ist nur von dem variablen Abstand h abhängig (zur Seite und nach oben verschobene Hyperbel)

3.6.2.2 Umsetzung der Projektorkalibrierung

Unser Ziel ist eine Funktion g, die für ein bekanntes Musterstück (Label) aus der Position im Bild den Abstand von der Kamera herleitet, wie es in Abbildung 3.11 schematisch dargestellt ist.

Genau gesagt benötigen wir die Abbildungsvorschrift $g : U \times L \to [A, B]$, die einer horizontalen Position im Bild $u \in U$ eines Labels $l \in L$ einen Abstandswert $h \in [A, B]$ zuordnet, wobei $[A, B]$ das Intervell der möglichen Tiefenwerte unseres Sensors ist.

Um diese Funktion g zu erstellen, kann das schon kalibrierte Stereokamerasystem verwendet werden. Dabei wird das Sensorsystem dicht und planparallel ($\alpha = 0$) über eine Ebene mit einer matten grauen Oberfläche (damit die Streifen gut abbildbar sind und die Farben nicht verfälscht werden) gefahren und ein Tiefenbild berechnet. Dann fährt das

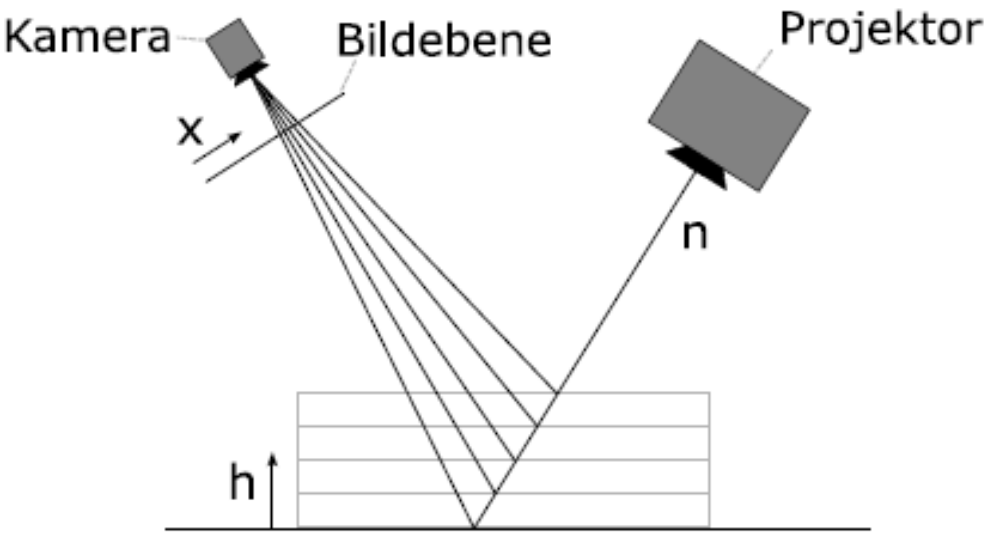


Abbildung 3.11: Funktionsprinzip der Tiefenberechnung aus der Position eines Labels im Bild (Projektorkalibrierung)

Sensorsystem Schrittweise immer weiter weg, bis durch die Unschärfe der Kameras keine Tiefenbilder mehr berechnet werden können. Das gesamte Messvolumen wurde somit abgefahren.

Sei $l_1 \in L$ ein Label. Durch unsere Kalibrierung wurden mehrere Messpunkte auf der Kurve $g(u, l_1)$ erzeugt. Um sie zu vervollständigen, kann entweder *linear interpoliert* (zwischen den Messwerten) und *linear extrapoliert* (jenseits des Randes der Messwerte) werden oder es kann gemäß der Herleitung aus Abschnitt 3.6.2.1 eine Kurve eingefittet werden.

Da der hieraus errechnete Abstandswert später mit den Abstandswerten aus der Stereoabstandsmessung verglichen werden soll, hat es sich als praktisch einfacher erwiesen, nicht die Funktion $g_l : U \to [A, B]$, die jeder Bildposition eines Labels direkt die Tiefe zuordnet, sondern die Funktion $\widetilde{g}_l : U \to D$, mit $D =$ (Menge der Disparitäten), die einer Bildposition die zugehörige Disparität zuordnet. $\widetilde{g}_l$ ist eine Gerade. Das Ergebnis unserer Projektorkalibrierung ist also eine Geradenschar.

Wie jede Messung ist auch die Projektorkalibrierung fehlerbehaftet. Da diese Fehler jedoch verheerend für die Tiefenschätzung wären, wird besonderes Augenmerk auf die Fehlerminimierung gelegt. Zunächst werden nicht nur die 4 nötigen Streifen zur Erkennung eines Labels benutzt, sondern 5. Damit wird eine Farbfehlsegmentierung in den meisten Fällen erkannt. Des Weiteren wurde der Lack der Basisebene optimiert, um farbtreu zu sein und gleichzeitig gute Streuungseigenschaften zu haben.

Dadurch, dass die Projektionsfläche eben ist, können ferner falsche Disparitätswerte erkannt werden. Hierfür wird zunächst eine Regressionsebene analog zur Berechnung der Hauptachsen einer Punktewolke eingepasst. D.h. der Schwerpunkt $m \in \mathbb{R}^3$ aller Punkte $p_k \in \mathbb{R}^3$, $k = 1,\ldots,n$ wird berechnet:

$$M = \frac{1}{n} \sum_{k=0}^{n-1} p_k$$

und daraus die schwerpunktbereinigte Kovarianzmatrix C erstellt

$$C = \sum_{k=0}^{n-1} (p_k - m) \cdot (p_k - m)^T$$

deren Eigenvektor zum kleinsten Eigenwert die Normale der Regressionsebene bildet. Die Eigenwerte werden wieder durch das numerisch stabile Verfahren der Singulärwertzerlegung bestimmt.

Die Regressionsebene soll zur Ausreißererkennung verwendet werden, die Ausreißer werdem aber mit bei ihrer Berechnung benutzt, was das Ergebnis bei starken Ausreißern stark verfälschen könnte. Um dieses Dilemma zu lösen, wird ein RANSAC-Algorithmus [MAF81] verwendet. D.h. man verwendet nicht alle Punkte p einer Tiefenschätzung um die Rekursionsebene zu bestimmen, sondern nur einen Teil. Danach wird geprüft, wie viele aller Punkte auf der Ebene liegen (also in einem kleinen Bereich um die Ebene herum). Dies ist ein Gütemaß für die Ebene. Der Vorgang wird eine feste Anzahl oft (in unserem Fall 100 Iterationen) wiederholt und die Regressionsebene mit der besten Güte gewählt. Alle Ausreißer bezüglich dieser besten Regressionsebene werden gelöscht (alle Punkte die mehr als 0,5 mm Abstand haben), bevor g berechnet wird.

Auch für das Einpassen der Regressionsgerade g in die ausreißerbereinigten Messwerte wird ein RANSAC-Algorithmus verwendet, so dass auch Fehler entdeckt werden, die sich auf eine gesamte Tiefenbildaufnahme aus der Messreihe beziehen.

Die gesamte Kalibrierung kann vollautomatisch durchgeführt werden. Der Sensorkopf muss lediglich planparallel zur Messoberfläche ausgerichtet und dicht über diese gefahren werden.

Zusätzlich zur eigentlichen Projektorkalibrierung können weitere Optimierungsparameter während des Kalibriervorgangs bestimmt werden. Zunächst verwendet der Algorithmus, wie bereits im Kapitel 3.3.1 über das Muster erwähnt, die Streifensequenz doppelt, um also 162 Streifen aus 81 zu machen. Beispielsweise ist Label 20 der gleichen Streifensequenz zugeordnet wie Label 101. Mit der Projektorkalibrierung kann die Uneindeutigkeit der Zuordnung einer Sequenz auf das richtige Label aufgelöst werden. Wie wir hier bestimmt haben, ist jede Bildposition eines Labels einem Abstand des Objektpunktes von der Kamera zugeordnet. Schränkt man den möglichen Tiefenbereich des Sensorsystems auf den Schärfentiefebereich ein, so kann ein bestimmtes Label nicht mehr überall im Bild vorkommen. Es zeigt sich, dass sich die Bereiche je zweier Label, die der gleichen Streifensequenz zugeordnet sind, bei unserem Sensorsystem nicht überlappen. Die Uneindeutigkeit ist aufgelöst und mehr noch - viele Fehlrekonstruktionen bei der Tiefenbildrekonstruktion können so erkannt werden, weil ein vermeintlich erkanntes Label an der entsprechenden Position überhaupt nicht vorkommen kann.

Ferner ist nicht das gesamte Muster im Bildfeld der Kamera. Um später Rechenzeit zu sparen, können Label, die während der Projektorkalibrierung nicht gefunden wurden, bei der späteren Rekonstruktion weggelassen werden.

3.7 Umsetzung des Modelltrainings

Das Modelltraining wird anhand hierfür programmierter Einlernmasken durchgeführt. Als Hintergrund dient eine ebene Fläche mit gutmütiger Oberfläche für die Tiefenbildrekonstruktion (matt grau). Zunächst wird die Kamera senkrecht über dem Objekt nach Augenmaß so positioniert, dass es scharf in der Mitte des Bildfeldes zu sehen ist. Daraufhin wird die Hintergrundebene eintrainiert, um eine grobe Segmentierung zu ermöglichen. Die Toleranz, Punkte welchen Abstands zur Hintergrundebene noch als Hintergrund gewertet werden, wurde hierbei je nach Objektgröße zwischen 1 und 5 mm eingestellt.

Im zweiten Schritt werden automatisiert die Beleuchtungsparameter (Belichtungszeit und Verstärkungsfaktor der Kamera) trainiert. Hierfür wird die Belichtungszeit zwischen 5 und 30 ms in 5-er Schritten erhöht. Danach wird der Verstärkungsfaktor von 0 bis 100 Prozent der Maximalverstärkung (Skala siehe [Gmb80] unter „gain“, es werden zunehmend

Bits der 256 Bit tiefen Helligkeit jedes Pixels ignoriert) erhöht. Das Bewertungskriterium für die Auswahl des richtigen Parametersatzes ist hierbei die Anzahl der rekonstruierten Punkte. Bei zu dunkler Beleuchtung werden nur sehr wenige Punkte rekonstruiert, so dass die Anzahl im Verlauf der automatischen Ermittlung zunächst schnell zunimmt. Wird die Aufnahme zu hell, so sind zunehmend Pixel überblendet, deren Farbwert dann nicht mehr ermittelt werden kann. Die Anzahl der auf dem Objekt rekonstruierten Punkte nimmt wieder ab. Gleichzeitig führt diese Überblendung jedoch zu Fehlrekonstruktionen auf Objekt und Hintergrund. Zweitere können vor der abgeschnittenen Ebene liegen und damit als zum Objekt gehörig gewertet werden, womit die Gesamtzahl der rekonstruierten Punkte u.U. weiter leicht zunehmen kann. Da es sich aber in zunehmendem Maße um *fehlrekonstruierte Punkte* handelt, ist eine solche Einstellung nicht erwünscht. Diesem Verhalten konnte durch die Einführung einer Schwelle entgegengewirkt werden. Die Anzahl der rekonstruierten Punkte muss demnach für einen zielführenden Optimierungsschritt zumindest um den Wert dieser Schwelle größer sein, als beim bisherigen Optimum (erfahrungsgemäß etwa 5%). Der Benutzer kann die Beleuchtung nach Augenmaß nachstellen. Empirisch hat es sich als günstiger erwiesen, die Kamera um eine Stufe dunkler einzustellen, als von diesem Algorithmus vorgeschlagen (weniger Punkte insgesamt zugunsten von weniger Fehlrekonstruktionen durch Überblendung).

Der nächste Schritt ist das Einlernen der Formbasierten Erkennung (SBM - Shape Based Matching). Ab diesem Zeitpunkt darf das Objekt nicht mehr bewegt werden, um die Ausgangslage nicht zu verfälschen. Hierfür werden die Umrisse des Objekts im Tiefenbild erkannt und geglättet (siehe Abschnitt 3.4). Alle Punkte innerhalb der so entstandenen Maske werden später als Objektpunkte segmentiert. Dabei dürfen keine Hintergrundpunkte im Tiefenbild enthalten sein, da diese die Segmentierungsmaske verfälschen würden. Sehr kleine Regionen, wie sie oft durch Fehlrekonstruktionen (Überblendung) im Hintergrund entstehen, können bei der Segmentierung ausgeschlossen werden. Schließt man jedoch zu große Regionen aus, werden feine Strukturen des Objekts nicht mehr erkannt. Mittels Einstellung des Parameters für die Stärke der Glättung, sowie für die maximale Größe der zu ignorierenden Regionen wird die Segmentierung so lange trainiert, bis die Maske nach Augenmaß gut die Objektregion umfasst. Bei den meisten Objekten handelt es sich dabei um ein bis zwei Iterationen.

Nachdem die Segmentierung fertig trainiert ist, wird die Objektlage im Bild in zwei Schritten bestimmt. Der erste, die Groblageerkennung, erstellt den Schwerpunkt der verbliebenen Punkte sowie deren Hauptachsen. Hierfür müssen Symmetrien (Rotationssymmetrie und n-fache Rotationssymmetrie) um die Hauptachsen angegeben werden.

Der letzte Schritt ist die Feinlagebestimmung. Hierbei muss im Wesentlichen bestimmt werden, wann die Iteration abgebrochen werden soll. Das ist eine Abwägung zwischen Genauigkeit und Geschwindigkeit. Da dieser Programmteil, wie später gezeigt wird, in unserer Implementierung nur einen sehr kleinen Teil der Laufzeit ausmacht, kann der Fokus auf ein optimales Ergebnis gelegt werden. Wie Messungen mit verschiedenen Bauteilen (vgl. 4.2.2) ergeben haben, nimmt die Güte bis 10 Iterationen stark und bis 20 Iterationen noch leicht zu. Danach fällt sie unter Umständen sogar leicht ab. Für unsere Experimente wird daher die fixe Anzahl von 20 Iterationen verwendet.

4. Experimente

„Keine noch so große Zahl von Experimenten kann beweisen, daß ich recht habe; ein einziges Experiment kann beweisen, daß ich unrecht habe.“
(Albert Einstein, 1879-1955)

In diesem Kapitel werden die Experimente beschrieben, die neben dem Nachweis der Funktionalität der im letzten Abschnitt vorgestellten Algorithmen deren Parametrisierung, Genauigkeit, Geschwindigkeit und Abdeckung des Teilespektrums untersuchen. In Abschnitt 4.1 werden die Genauigkeitsmessungen und die Abdeckung des Teilespektrums untersucht. Abschnitt 4.2 beinhaltet die Messungen zur Geschwindigkeit und deren Auswertung, sowie weitere Messungen zur Parametrierung der Algorithmen. Der letzte Abschnitt 4.3 dient der Fehleranalyse und den daraus zu ziehenden Folgerungen.

4.1 Genauigkeitsmessungen

Hauptaspekt eines jeden weiterentwickelten oder neuen Algorithmus ist dessen grundsätzliche Funktionalität. Hier wird diese auf dem vordefinierten Musterteilesatz überprüft, was einen Kernaspekt der vorliegenden Arbeit darstellt. Es entsteht eine Abschätzung für die Leistungsfähigkeit des Gesamtsystems für die Lageerkennung typischer industrieller Bauteile, sowie dessen Genauigkeit mit gegebener Hardware und der im Rahmen dieser Arbeit realisierten Software-Implementierung.

In Abschnitt 4.1.1 werden zunächst die Randbedingungen definiert und beschrieben, unter denen die Messungen durchgeführt wurden. Der nächste Abschnitt 4.1.2 enthält die Ergebnisse dieser Messungen bezüglich ihrer Genauigkeit auf allen Teilen des Bauteilekoffers.

4.1.1 Durchführung und Randbedingungen der Messreihen

Der Algorithmus, wie er in Kapitel 3 beschrieben wurde, besteht aus zwei grundlegenden Teilen: dem Modelltraining und der eigentlichen Objekterkennung. Beide Teile wurden nacheinander für jedes der Objekte aus dem Teilekoffer (und einzelnen zusätzlichen Objekten) durchgeführt. Wie kann man aber die Genauigkeit bewerten?

Um zu ermitteln, wie weit die Objektlageschätzung von der wahren Lage abweicht, muss diese wahre Lage (sog. *Ground Truth*) bekannt sein.

Zunächst wird für jedes Bauteil das Modelltraining wie in Abschnitt 3.7 beschrieben durchgeführt. Das Objekt liegt hierbei in Ausgangslage, so dass damit eine „wahre“ Objektlage ermittelt wird. Dannach folgt die Objektlageerkennung.

Durchführung der Objektlageerkennung

Die Objektlageschätzung liefert eine Transformationsmatrix (Basiswechsel) vom Kamerasystem in das Objektsystem, welche aus Translationsvektor (drei Freiheitsgrade) und Rotationsmatrix (genauer: Einheitsquaternion, 3 Freiheitsgrade) besteht, also die Lage des Objekts relativ zur Kameraposition beschreibt. Als *Ausgangslage* dient die Position der Kamera über dem Objekt beim Training. Nach dem Modelltraining wird das Objekt auf dem Tisch nicht mehr bewegt. Relativ zu dieser Ausgangslage werden mit dem Roboter mit aufmontiertem Kamerasystem dann festgelegte Positionen angefahren und jeweils eine Objektlageschätzung durchgeführt. Der Algorithmus liefert hierbei eine Transformation, wie sich der Roboter bewegen müsste, um wieder in der gleichen Position über dem Bauteil zu sein wie beim Einlernen. Unter der Annahme, dass die Genauigkeit des Roboters wesentlich höher ist als die Genauigkeit des zu bewertenden Algorithmus, kann davon ausgegangen werden, dass die wahre Lage des Objektes damit bekannt ist. Die geschätzte Position und die wahre Position können also miteinander verglichen werden. Die Abweichung beider wird in allen sechs Freiheitsgraden (3 translatorische, 3 rotatorische) in einer Tabelle aufgetragen (siehe Abschnitt 4.1.2).

Auf diese Weise werden drei Messungstypen für jedes Objekt durchgeführt:

A Wiederholgenauigkeit: Es werden 25 unabhängige Objektlageschätzungen ohne Veränderung der Position durchgeführt.

B Objektlageschätzung: Zunächst wird die relative Translationslage von Kamera zu Objekt variiert. Hierfür wird ein würfelförmiges Raster mit Kantenlänge 5 mm abgefahren, wobei jede der Achsen (x,y,z) drei gleichverteilte Positionen hat (je die Ecken und die Mitte zwischen zwei Ecken), was $3^3 = 27$ Positionen ergibt. An jeder dieser Positionen wird eine Objektlageschätzung senkrecht durchgeführt und die Kamera für jeweils 5 weitere Schätzungen verkippt, so dass die Kamerapositionen möglichst gleichverteilt auf einem Sphärenausschnitt einer Kugel mit Radius gleich dem Arbeitsabstand (10 cm) und Öffnungswinkel gleich 5° befinden. Dies ergibt insgesammt $3^3 \cdot (5 + 1) = 162$ Objektlageschätzungen.

C Rotationslageschätzung: Die Kamera wird in 5 Grad-Schritten um die senkrechte Achse (z-Achse = Achse durch Mittelpunkt der beiden Kameras und dem Schnittpunkt der „schielenden“ optischen Achsen der Kameras) gedreht, d.h. das Objekt dreht sich um 360 Grad im Bild. Dies ergibt $(360 : 5) - 1 = 71$ Messpunkte.

Die Messung vom Typ A dient dabei der Ermittlung der Streuung der Tiefenbildrekonstruktion und der grundlegenden Genauigkeit des Messverfahrens (Ruhestreuung). Die Messungen vom Typ B sind die eigentliche Hauptmessung der Genauigkeit. Um die Objektlage für einen Griff von einer ebenen Fläche oder aus einem Prozessnest zu ermitteln, muss ein Algorithmus die hiermit abgedeckte Lagevariation erkennen können (vgl. [Say11]). Die Größe der zu messenden Variation wurde also anhand der typischen Anwendungsfälle definiert. Die Translationsvariation in x und y ist gerade so gewählt, dass selbst große Objekte stets vollständig im Bildfeld zu sehen sind. Z wird grob innerhalb der Schärfentiefe variiert. Messungen vom Typ C decken den letzten bisher nicht variierten Freiheitsgrad ab, nämlich die Rotation im Bild (um die z-Achse = optische Achse des rektifizierten Stereokamerasystems). Die Analyse der sich hieraus ergebenden systematischen Fehler befindet sich in Abschnitt 4.3.

4.1.2 Ergebnisse der Messreihen

Ziel unserer gesamten Bemühungen ist, wie es bereits in der Einleitung dieser Arbeit formuliert wurde, ein Algorithmus zur Objektlageerkennung, der

1. genau genug ist, um erkannte Objekte greifen zu können.
2. eine möglichst gute Abdeckung des Bosch Teilespektrums bietet, wie es im Bauteilekoffer zusammengefasst ist.
3. möglichst laufzeiteffizient ist. Als Taktzeit für die Objektlageerkennung wurde zunächst 1 Sekunde angegeben, was im Verlauf dieser Untersuchung auf 0,5 Sekunden verschärft wurde.

Dieser Abschnitt beschäftigt sich mit den Ergebnissen zu den Zielen eins und zwei. Ziel drei wird in Abschnitt 4.2 untersucht.

Ziel 1: Genauigkeit

Betrachten wir zunächst Ziel eins, die Genauigkeit. Wie genau muss ich ein Objekt erkennen, damit es später abgegriffen werden kann? Zur Festlegung von Grenzen dienen die Untersuchungen der Arbeit von Sayler [Say11]. Dort wurde festgelegt, dass mindestens eine Genauigkeit von $\pm 2,5$mm in jedem der drei Translationsfreiheitsgraden und von $\pm 2,5$ Grad in jedem der drei Rotationsfreiheitsgraden erreicht werden muss. Man möchte diese Vorgabe aber trotz statistischer Streuung mit einer sehr hohen Wahrscheinlichkeit einhalten. Dazu nimmt man eine Normalverteilung der Messergebnisse an und fordert eine Einhaltung in über 99,8 Prozent der Fälle, was gerade die Wahrscheinlichkeit ist, mit der ein Messergebnis innerhalb eines Intervalls von 3σ liegt, wobei σ die Standardabweichung angibt. Unser Algorithmus erfüllt diese Voraussetzung also, wenn er bei einem Erwartungswert von $\mu = 0$ keine größere Standardabweichung, als $\sigma_{Trans} = 0,8$ mm bzw. $\sigma_{Rot} = 0,8$ Grad aufweist. (Dies entspricht einer Toleranz von ± 5 mm bzw. ± 5 Grad bei einem 6σ-Intervall)

Betrachten wir als Beispiel einer Genauigkeitsmessung das Bauteil Nummer 17 (Sensorelement), welches in Abbildung A.14 dargestellt ist. Zunächst wurde das Modell des Bauteils, wie in Abschnitt 4.1.1 beschrieben trainiert. Danach wird die Lage des Objekts auf dem Einlerntisch nicht mehr verändert und die Messungen vom Typ A, B und C (siehe 4.1.1) durchgeführt. Die Ergebnisse sind in Tabelle A.13 aufgetragen.

Tabelle 4.1: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 17 (Sensorelement) in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,083	0,054	0,002	0,125	0,086	0,034
Versuch A: Rng	0,329	0,197	0,009	0,467	0,323	0,120
Versuch B: σ	0,572	0,288	0,278	0,491	0,338	0,100
Versuch B: Rng	2,510	1,384	0,995	3,049	2,020	0,611
Versuch C: σ	0,769	0,339	0,325	1,421	0,804	0,252
Versuch C: Rng	2,579	0,962	0,954	5,092	2,997	0,999

Betrachten wir zunächst die Messung vom Typ A (erste und zweite Zeile der Tabelle). Dabei misst man 25 mal die Objektlage mittels unseres Algorithmus ohne die Kameraposition zu verändern und betrachtet jeweils den Abstand der geschätzten von der wahren

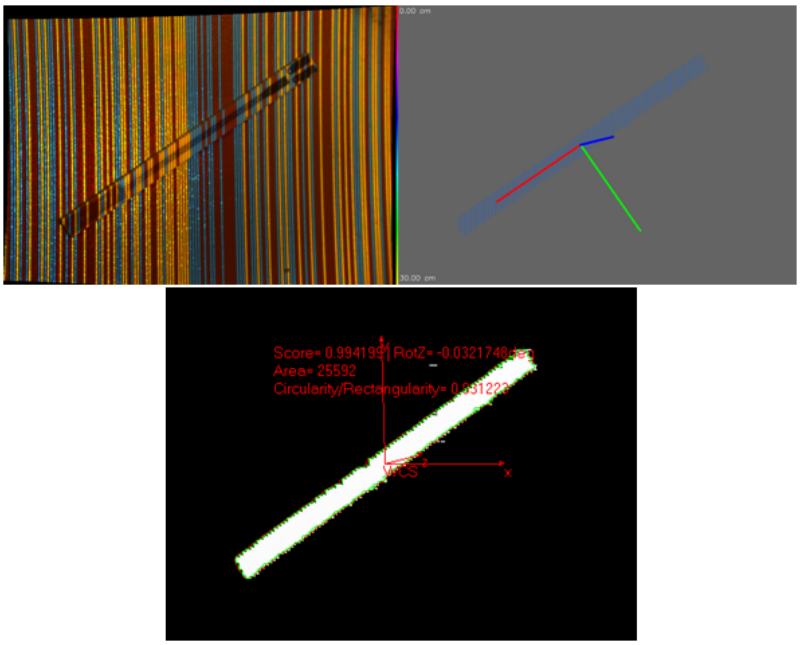


Abbildung 4.1: oben links: linkes rektifiziertes Kamerabild des Bauteils Nummer 17 (Sensorelement) mit Streifenmusterbeleuchtung; oben rechts: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); unten: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert)

Position (Einlernposition). Die Tabelle enthält die Standardabweichung (Wurzel der Abstandsquadrate von jeder geschätzten zur wahren Lage) in jedem Parameter (Translation: x,y,z; Rotation um x, Rotation um y, Rotation um z). Sei also $\overline{x}$ die x-Koordinate der wahren Position, wie sie oben definiert ist, und x_i^A die x-Koordinate der Objektlage, wie sie in der i-ten Messung der Messreihe vom Typ A ermittelt wurde, dann enthält das erste Feld gerade

$$\sigma_x^A = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i^A - \overline{x})^2}$$

wobei $n = 25$ die Anzahl der Messungen ist.

Die anderen Felder der Zeile enthalten die entsprechenden Werte für die weiteren Translationsgrößen y und z (jeweils in mm), sowie der Rotationsabweichungen um die x-, y- und z-Achse des Objekts, wie sie im Modelltraining geschätzt wurden. Man könnte auch ein beliebiges anderes Koordinatensystem verwenden. Die hier gewählte Darstellung hat aber den Vorteil, dass eine erhöhte Streuung, die mit der Ausdehnung des Objekts in einer bestimmten Richtung zu tun hat, entkoppelt dargestellt wird.

Die Zeile darunter enthält die maximale Abweichung der Messung in der jeweiligen Koordinate, also z.B in der x-Koorinate:

$$x_{max}^A = \max_{i=1}^{n}(x_i^A - \overline{x})$$

Da der Roboter nicht bewegt wird und die „wahre Position“ der gleichen Schwankung unterliegt wie jede andere dieser Objektlagebestimmungen, kann mit dieser Messung eine Aussage über die Grundgenauigkeit des Modells getroffen werden. Sie liegt bei diesem Bauteil selbst im schlechtesten Wert (Rotation um die x-Achse) mit $Rot_x = 0,125$ Grad um mehr als einen Faktor 6 unter der gewünschten Genauigkeit. Da es sich hier nur um die Grundgenauigkeit handelt, wäre eine Schwankung, die eine Größenordnung (also Faktor

10) besser als die gesuchte Genauigkeit ist, erstrebenswert. Die Messung zeigt aber, dass das Ergebnis nahe am Zielbereich ist. Eine Fehleranalyse und Verbesserungsvorschläge folgen in Abschnitt 4.3. Dass gerade dieser Wert ungenau ermittelt wird, ist objektspezifisch. Wie man in Abbildung A.14 sieht, handelt es sich um ein sehr langes, dünnes Bauteil, so dass die Verkippung um die lange Achse (x-Achse) nur „ungenau“ (man beachte, dass es sich hierbei nur um $\frac{1}{8}$ Grad handelt) geschätzt wird. Die Genauigkeit der Translationsschätzung ist typisch für die meisten Bauteile. Die Genauigkeit in z ist sehr gut, da das Bauteil in diese Richtung die kleinste Ausdehnung hat und recht flach ist. Dies ergibt viele Messwerte, die kaum variieren.

Die Schätzung in x ist etwas schlechter als in y. Das liegt mehr an der Auflösung des Messsystems, als am Bauteil selbst. In der Achse, die im Bild vertikal ist (entlang der Farbmusterstreifen, oft die y-Achse des Objekts) ist die Auflösung pixelgenau. Das entspricht bei unserem Arbeitsabstand (16 cm), unserer Kameraauflösung (768x576 Pixel) und unserem Bildfeld einer Auflösung von ca. 0,24 mm/Pixel in u (horizontale Koordinate im Bild) und v (vertikale Koordinate im Bild), sowie einer Tiefenwert-Auflösung von ca. 0,43 mm/Pixel (der Tiefenabstand auf dem Objekt, der der Disparität von einem Pixel entspricht). Die Messwertdichte in v (viele Objekte sind so gelegt, dass dies der x-Achse entspricht), ist jedoch nicht gleich der Pixelauflösung, sondern nur gleich dem Streifenabstand auf dem Objekt. Der Streifenabstand ist etwa 6 Pixel, so dass die Auflösung etwa 1,44 mm beträgt. Daher ist die Messgenauigkeit in dieser Richtung oft etwas schlechter und mit 0,083 mehr als eine Größenordnung unter der Auflösung (man kann nach einem üblichen Daumenwert maximal etwa eine Größenordnung subpixelgenaue Auflösung erreichen). Dies ist auch an anderen Objekten wie z.B. in Abbildung A.15 zu sehen.

Die Messungen vom Typ B sind die eigentlichen Hauptmessungen der Objektlagebestimmung an dem jeweiligen Bauteil. Hierbei werden analog zu Messungen vom Typ A in der dritten Zeile der Tabelle die Standardabweichung der Abweichungen von der geschätzten Lage, also z.B. in der x-Koordinate

$$\sigma_x^B = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i^B - \overline{x})^2}$$

sowie in der vierten Zeile der entsprechende Maximalwert innerhalb der Messreihe

$$x_{max}^B = \max_{i=1}^{n}(x_i^B - \overline{x})$$

eingetragen, wobei x_i^B der i-te Messwert dieser Messreihe und $n = 162$ die Anzahl der Messungen sind.

Das Modelltraining wird nicht erneut ausgeführt (das Modell ist also dasselbe, wie in den Messungen vom Typ A und C), der Roboter wird im Gegensatz zu Typ A jedoch nach dem oben angegebenen Muster bewegt (vgl. 4.3 für die daraus folgende Fehlerbetrachtung). Damit soll die Güte der Objektlageerkennung unter Variation des Blickwinkels innerhalb der vom Anwendungsfall gegebenen Größenordnung untersucht werden. Unser Beispielbauteil erfüllt die Genauigkeitsanforderung und gehört daher zu

$$\text{Kategorie } 1 := \{\text{Bauteile } B \| B \text{ erfüllt die Genauigkeitsanforderung in jedem Parameter}\}$$

Die Messwerte entsprechen den aus der Messreihe vom Typ A geweckten Erwartungen mit etwas erhöhten Werten in der Abweichung der geschätzten x-Position bzw. der geschätzten Rotation um die x-Achse. Insgesamt sind alle Abweichungen natürlich höher als die Grundgenauigkeit (Faktor 4 bis Faktor 20), was der Erwartung entspricht, da der Blickwinkel im Vergleich zur Modellansicht variiert wurde. Die Genauigkeit in der Translation

entspricht etwa dem 1/3 - (in der x-Koordinate) bis 1- fachen (in der y-Koordinate) der jeweiligen Auflösung des Messsystems. Das sind typische Werte für Bauteile der Kategorie 1.

Der einzige in Typ B nicht variierte Freiheitsgrad ist die Rotation in der Bildebene. Hierfür wird eine Sondermessung (Typ C) wiederum mit dem selben Modell durchgeführt. Besonderes Augenmerk liegt hierbei natürlich auf dem variierten Parameter, welcher bei fast allen Bauteilen der Rotation um die z-Achse entspricht mit einer leichten Translationsverschiebung, da die z-Achse des Bauteils üblicherweise nicht *gleich* der Rotationsachse (optischen Achse des Stereokamerasystems) sondern nur *parallel* zu dieser ist.

Bei diesem Bauteil fällt auf, dass der Streuungswert für die z-Rotation *nicht* innerhalb der gewünschten Grenzen liegt. Nach Betrachtung der Einzelbilder, bei denen der Messwert sehr hoch war, fällt auf, dass das Objekt in diesen nicht mehr vollständig im Bild war. Das Bauteil hat also für die Auflösung eine zu große Ausdehnung. Wenn diese Rotationsausrichtung im gegebenen Anwendungsfall ausgeschlossen werden kann, gehört das Bauteil in Kategorie 1 (Algorithmus anwendbar), wenn nicht, in

$$\text{Kategorie 3} := \{\text{Bauteile } B \| B \text{ erfüllt die Genauigkeitsanf. nicht in allen Parametern}\}$$

Die noch verbleibende

$$\text{Kategorie 2} := \{\text{Bauteile } B \| B \text{ Alle Ausreißer der Messungen sind hebbar}\}$$

soll anhand des Bauteils Steckergehäuse (siehe Abbildung 4.2) erläutert werden.

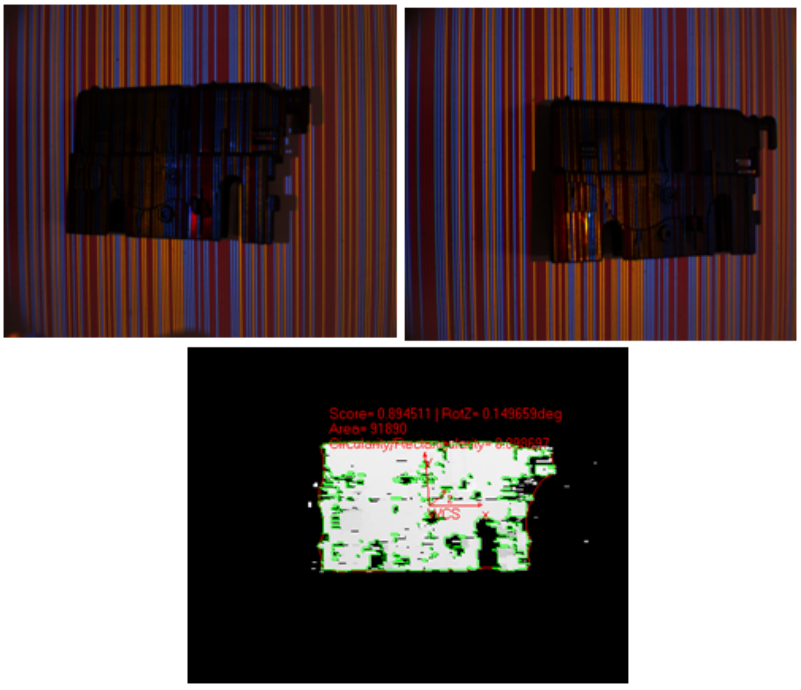


Abbildung 4.2: oben: Beispielbildpaar des Bauteils Nummer 32 (Steckergehäuse) mit Streifenmusterbeleuchtung; unten: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert

Wie aus Tabelle 4.2 ersichtlich ist, erfüllt das Bauteil nicht in allen Parametern die geforderte Genauigkeit. Die Tabelle ist hierbei analog zu Tabelle A.13 zu lesen.

Es könnte beispielsweise die Kamera defokusiert, bzw. das Objekt außerhalb des Tiefenschärfebereichs platziert sein. Dies würde unter Umständen zu einem so verschwommenen

Tabelle 4.2: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 32 (Steckergehäuse) in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,056	0,043	0,019	0,042	0,074	0,125
Versuch A: Rng	0,199	0,204	0,075	0,169	0,252	0,525
Versuch B: σ	0,402	0,636	0,121	0,246	0,289	1,118
Versuch B: Rng	2,009	2,759	0,719	1,972	1,412	8,162
Versuch C: σ	0,621	0,715	0,181	0,298	0,451	1,439
Versuch C: Rng	2,638	3,175	0,770	1,592	2,322	7,605

Bild führen, dass keine Segmentierung und damit auch keine 3D-Rekonstruktion mehr möglich ist. Der Rekonstruktionsalgorithmus würde dies jedoch bemerken und eine entsprechende Warnung ausgeben. Auf eine solche Warnung kann einfach reagiert werden, indem der Roboter ein kleines Stück verfährt und die Objektlageerkennung erneut durchgeführt wird. Kann mit einer solchen Algorithmuserweiterung erreicht werden, dass die Standardabweichung um Null der Lageschätzung die geforderte Genauigkeit erfüllt, so werden diese Ausreißer als *„hebbar“* bezeichnet. Die Bauteile mit ausschließlich hebbaren Ausreißern bilden die Kategorie 2.

Neben der Anzahl der rekonstruierten Punkte als Parameter für die Güte der Messung, deren Unterschreitung zu einer Warunung und gegebenenfalls zu einer erneuten Messung führen kann, haben auch andere Teile des Algorithmus solche Gütemaße. Der zweite Teil hierfür ist die formbasierte Erkennung (Shape Based Matching). Der Algorithmus gibt in einem Parameter an, wie genau der gefundene Umriss der trainierten Schablone entspricht. Schon ohne Rekonstruktionsfehler passt die Schablone durch die leicht veränderte Ansicht nicht exakt, aber eine wesentlich verschlechterter Wert deutet auf eine Fehlerkennung bzw. Fehlrekonstruktion hin. Während des Modelltrainings kann hier, wie für alle anderen Gütemaße, eine Schwelle gesetzt werden, ab wann die Messung als missglückt betrachtet werden soll. Die typischen Ausreißer von Bauteilen der Kategorie 2, die am SBM-Gütemaß liegen, haben *deutlich* schlechtere Werte, als eine gute Erkennung. Ein üblicher Wert für das Gütemaß liegt bei 0,9 auf einer Skala von 0,0 (keine Übereinstimmung) bis 1,0 (hundertprozentige Übereinstimmung). Typische Ausreißerwerte liegen bei einem Wert von kleiner als 0,6. Die Schwelle ist deshalb meist auf 0,7 eingestellt (je höher, desto langsamer der Gesamtalgorithmus, da Messungen häufig wiederholt werden müssen, je niedriger, desto wahrscheinlicher wird eine falsche Erkennung nicht bemerkt).

Der dritte Teil des Algorithmus, der über ein Gütemaß verfügt, ist die Hauptachsentransformation. Zur Bestimmung der Objektlage werden von ihr nur die Eigenvektoren benötigt, die allerdings nach Größe des zugehörigen Eigenwertes sortiert sein müssen, weil sonst z.B. die x-Achse des Modells auf die y-Achse des Objekts transformiert werden würde. Der Eigenwert selbst enthält aber auch eine Information. Weicht der gemessene Eigenwert während der Objektlageerkennung stark vom Wert während des Modelltrainings ab, so kann davon ausgegangen werden, dass ein Teil des Objektes nicht rekonstruiert wurde. Auch hierfür kann eine Schwelle gesetzt werden. In der Praxis trat dieser Fall jedoch nur selten auf, da meist schon eines der vorherigen Gütemaße die Lageerkennung als Fehlerkennung eingestuft hatte, wenn die Eigenwertdifferenzschwelle überschritten wurde.

Treten Fehlerkennungen auf, die nicht durch den Algorithmus erkannt werden können, oder handelt es sich bei den Fehlerkennungen nicht um wenige Ausreißer, sondern viel mehr um den Normalfall, so dass eine erneute Messung mit hoher Wahrscheinlichkeit zu keinem besseren Ergebnis führt als die erste, so gehört das Bauteil in *Kategorie 3* der nicht erkennbaren Bauteile. Bauteil Nummer 6 (Zündkerze, siehe Abbildung A.6 und die zugehörigen Messwerte A.6) ist ein Beispiel eines Bauteils dieser Kategorie.

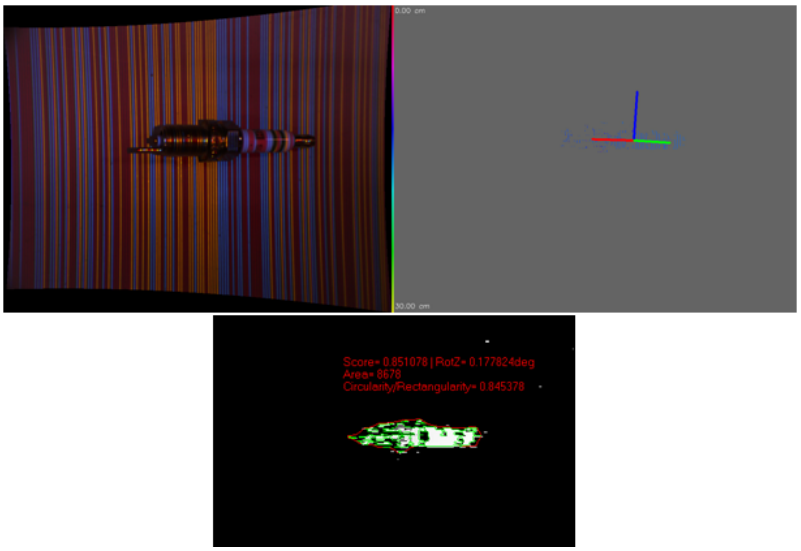


Abbildung 4.3: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 6 (Zündkerze) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert

Tabelle 4.3: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 6 in 6 Freiheitsgraden (drei Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,253	0,658	0,136	0,398	2,345	1,727
Versuch A: Rng	0,992	2,788	0,611	1,616	9,313	7,427
Versuch B: σ	0,641	1,979	0,445	1,888	5,740	2,738
Versuch B: Rng	3,694	10,020	2,882	17,106	33,504	14,218
Versuch C: σ	0,662	1,612	0,417	11,745	8,458	45,772
Versuch C: Rng	2,307	4,701	1,637	45,376	27,700	176,608

Zeigte(n) bei einem solchen Bauteil schon die Messreihe(n) vom Typ A oder B eine zu hohe Streuung, wurde(n) die folgende(n) Messreihe(n) nicht mehr durchgeführt. Die Zündkerze ist ein sehr typischer Vertreter eines Bauteils dieser Kategorie. Die polierte Keramikoberfläche, sowie das glatte Metall reflektieren das Licht stark. Deshalb ist der Anteil des Lichts, das lambertsch (matt) gestreut wird und somit für die Tiefenbildrekonstruktion mittels Musterprojektion als einziges verwendet werden kann, sehr klein. Dadurch schlägt die Rekonstruktion auf dem Großteil des Objektes fehl. Ferner führt der große spekular (glänzend) gestreute Anteil des Projektionsmusters zu Überblendung (Glanzflecke) und

Fehlrekonstruktionen (ein Scheinobjekt hinter der eigentlichen Oberfläche wird erkannt). Für diese Bauteile ist der hier vorgestellte Algorithmus - oder genauer gesagt die hier verwendete Tiefenbildrekonstruktion - unbrauchbar.

Weitere Fehlerursachen und Schlussfolgerungen daraus werden in den Abschnitten 4.3 bzw. 4.4 behandelt.

Ziel 2: Bauteileabdeckung

Um die Abdeckung des Algorithmus auf dem Bauteilekoffer zu prüfen (Ziel zwei aus der Aufzählung oben), wurde die im letzten Abschnitt beschriebene Prozedur auf alle Bauteile angewandt. Die Ergebnisse der Messungen sind in Tabelle 4.4 zusammengefasst. Die Darstellung der Messergebnisse für alle Bauteile würde dieses Kapitel überfrachten. Der interessierte Leser sei hierfür auf den Anhang verwiesen, in dem Beispielbilder und die Messergebnisse für alle hier nicht besprochenen Bauteile angegeben sind.

Die erste Spalte nach dem Bauteilnamen enthält das Ergebnis der Genauigkeitsuntersuchung, also die Kategorie 1,2 oder 3, ob das Bauteil nach Anforderung genau genug erkannt werden kann (Kategorie 1), einige (<10%) hebbare (s.o.) Ausreißer enthält (Kategorie 2), oder ob der Algorithmus für das Bauteil unbrauchbar ist (Kategorie 3).

Ausreißer innerhalb der Messreihen eines Bauteils wurden auf Fehlerursachen untersucht. Die nächsten fünf Spalten der Tabelle enthalten die häufigsten Fehlerursachen und jedes Bauteil, für das diese Ursache maßgeblich zutrifft, wurde mit einem Kreuz in der entsprechenden Spalte versehen. Im einzelnen sind diese:

- **Glanzpunkte**: Stark spekular reflektierende Objektoberflächen führen zu Glanzpunkten, die je nach Blickwinkel an einer anderen Stelle des Objekts liegen. Da die Kamera an dieser Stelle überblendet ist, kann lokal keine Rekonstruktion durchgeführt werden. Es fehlen je nach Ansicht unterschiedliche Stücke der Objektoberfläche. Führt dies zu Fehlerkennungen, wurde diese Kategorie mit einem Kreuz versehen.

- **kein Muster abbildbar**: Manche Bauteile sind (teilweise) aus Materialien, auf denen kein Muster abbildbar ist. Ein Beispiel hierfür ist Schaumstoff (Bauteil Dämpfer EPW2), auf dem die Streifen nur als verschwommene Farbgebung oder überhaupt nicht zu erkennen sind. Wenn der Rest des Objektes nicht für eine Objektlageschätzung ausreicht, wurde diese Kategorie angekreuzt.

- **Bauteilstrukturen zu fein**: Manche Bauteile besitzen Strukturen, die so klein sind, dass keine 3 aufeinanderfolgenden Streifen darauf passen. Das Muster ist erst ab vier aufeinanderfolgenden Streifen eindeutig, wenn links und rechts aber genügend große Musterstücke rekonstruiert wurden, können 3 Streifen ausreichen. Noch kleinere Strukturen sind jedoch unterhalb der Auflösungsgrenze und können nicht erkannt werden. Enthält das Bauteil Strukturen unterhalb dieser Grenze, die für die Erkennung maßgeblich notwendig sind, wurde diese Kategorie angekreuzt.

- **Hauptachse nicht eindeutig (Rechteck)**: Die erste Hauptachse eines im Bild rechteckigen Objektes liegt parallel zu den beiden langen Seiten, in deren Mitte. Wird jedoch beispielsweise die rechte obere Ecke in einer Ansicht nicht rekonstruiert (z.B. durch einen Glanzfleck dort), springt die erste Hauptachse auf die Diagonale zwischen linker oberer und rechter unterer Ecke. Dies entspricht einer Fehlerkennung in der Rotationsrichtung um die z-Achse (senkrecht zur Bildebene) um z.B. 30 Grad. Je nach aufgelöster Objektoberflächenstruktur kann der folgende ICP-Algorithmus diesen schlechten Startwert nicht mehr ausgleichen und es kommt zu einer Fehlerkennung. Kommt dies bei einem Bauteil vor, so wurde diese Kategorie angekreuzt.

Bauteilnummer	Bauteilname	Bewertung	Glanzpunkte	Bauteilstrukturen zu fein	kein Muster abbildbar	Rechteck	Sonstige	Symmetrie ignoriert	Optik
1	Stecker	3			X				B
2	Injektor	2					X		A
3	Anker	3	X	X			X		A
4	Steuergerät	2	X				X		A
5	Ankerrohling	3	X				X	X	A
6	Zündkerze	3	X					X	A
7	Leiterplatte	2	X				X		B
8	Aggregat	3	X			X			B
9	Keramikscheibe	3			X			X	A
10	Koppeleinheit	2			X			X	A
11	Widerstand	3		X					A
12	Magnetventil	1						X	A
13	Ventilplatte	1		X				X	A
14	Rohrwinkel	3	X						A
15	Bohrer	3	X	X				X	A
16	Magnetkern	1						X	B
17	Sensorelement	1		X				X	A
18	Wandler	1						X	A
19	Feder	3		X				X	B
20	Gehäuse	1						X	B

Tabelle 4.4: Bewertungsmatrix der Genauigkeit des Algorithmus auf allen Bauteilen des Teilekoffers

Bauteilnummer	Bauteilname	Bewertung	Glanzpunkte	Bauteilstrukturen zu fein	kein Muster abbildbar	Rechteck	Sonstige	Symmetrie ignoriert	Optik
21	Federteller	2		X				X	A
22	Gehäusedeckel	3	X				X	X	A
23	Spulenkörper	3	X						A
24	Spule	3	X					X	B
25	Keramikhülse	3		X	X			X	A
26	Dämpfer	3			X	X		X	B
27	Deckel	1	X				X	X	B
28	Nadel	3	X	X				X	A
29	Ventil	3							A
30	Drehfeder	3		X					B
31	O-Ring	2		X				X	A
32	Steckergehäuse	2				X			B
33	Wischer	3					X		B
34	Zündspule	3	X						B
101	Zylinder	2	X					X	A
102	Rohling	1	X						B
103	Rohr T-Stück	1							B
106	Gussteil	1						X	B
108	Düsennadel	1						X	B
	Summe		16	11	5	4	7	23	
	Anteil in %		42	29	13	11	18		

Tabelle 4.5: Bewertungsmatrix der Genauigkeit des Algorithmus auf allen Bauteilen des Teilekoffers (Fortsetzung)

- **Sonstige** Hierunter fallen alle sonstigen Gründe, die zu selten für eine eigene Kategorie waren. U.A. Selbstüberdeckungen des Bauteils, absolut flache Struktur (ICP-Probleme), starke Eigentextur.

Wie man sieht, sind Glanzpunkte auf der Objektoberfläche die Hauptursache für Ausreißer (42 % aller Bauteile), gefolgt von Bauteilstrukturen unter der Auflösungsgrenze (29 % aller Bauteile).

Die 3D-Rekonstuktionen mancher Bauteile sind nahezu symmetrisch bezüglich einer Achse. Beispiele hierfür sind rotationssymmetrische Bauteile (Schrauben, Zündkerze, Mittenspannteil) oder Bauteile die bei einer endlich-zähligen Rotation selbstähnlich sind, also z.B. ein Rechteck, das bei einer Rotation um 180 Grad wieder nahezu gleich aussieht (2-zählig). In diesem Fall kann die Objektlageschätzung angewiesen werden, den entsprechenden Freiheitsgrad zu ignorieren, und die räumlich *nächste* der möglichen Lagen anzufahren. War dies bei einem Objekt der Fall, so wurde es in der Spalte **Symmetrie ignoriert** vermerkt.

Es werden je nach Größe des Objektes zwei unterschiedliche Objektivsätze verwendet und den Arbeitsabstand entsprechend angepasst. In der letzten Spalte ist vermerkt, welcher Satz für die Messung verwendet wurde (Objektivsatz A für kleinen Arbeitsabstand oder Objektivsatz B für großen Arbeitsabstand).

4.2 Geschwindigkeitsmessung und Parametrierung

Neben der Hauptmessung auf dem gesamten Bauteilesatz müssen Messungen durchgeführt werden, welche die Geschwindigkeit und die richtige Parametrisierung der Algorithmen ermitteln.

4.2.1 Geschwindigkeitsmessung und -optimierung

Bei der Bewertung von Algorithmen für den industriellen Nutzen ist neben der Genauigkeit die Geschwindigkeit ein wichtiges Bewertungskriterium. Wie eingangs beschrieben, ist eine Gesamtlaufzeit für die Bildverarbeitung von unter 1 Sekunde, besser nur 500 Millisekunden gefordert, um Taktzeiten bei üblichen Anwendungen einzuhalten. Dies wurde sowohl in der Konzeption des Algorithmus, als auch durch spätere Beschleunigungsmaßnahmen berücksichtigt.

Ein wichtiger Bestandteil für die möglichst schnelle spätere Ausführung des Algorithmus ist die Wahl von Hardware und Programmiersprache. Hierbei muss ein Kompromiss zwischen einfacher und damit schneller Entwicklung der Algorithmen und tatsächlicher Laufzeit gefunden werden. Grundsätzlich kann man sagen, dass je hardwarenäher entwickelt wird desto aufwändiger die Entwicklung, aber auch kürzer die Laufzeit ist. (hardwarenah: Smartkamera, FPGA oder DSP; mittel: PC mit hoher Programmiersprache; hardwarefern: Bildverarbeitungsbibliothek wie etwa Matlab oder Halcon)

Die Hardwarevorgabe ist aus Kostengründen ein handelsüblicher PC. Aus Implementierungszeitgründen wurden Teile des Algorithmus in Halcon programmiert (siehe unten), aus Laufzeitgründen das Framework und große Teile der Algorithmen jedoch in C++ geschrieben. Da es sich bei Bildern um sehr große Datenmengen handelt, spart es bei derzeitigen Compilern Laufzeit, eine Programmiersprache zu verwenden, die eine manuelle Speicherkontrolle zulässt. Damit scheiden die sonst üblichen Sprachen C#, Java oder Matlab aus. C++ stellt einen guten Kompromiss zwischen Komfort und Maschinennähe (und damit Laufzeiteffizienz) dar. Ferner wurde bei der Programmierung auf die Objektorientierung und damit Modularisierung des Codes geachtet, nicht zuletzt um eine spätere makroskopische Parallelisierung zu ermöglichen.

Die in den Messungen gezeigte Laufzeiteffizienz wurde neben der Konzeption durch drei Beschleunigungsmaßnahmen erreicht:

1. makroskopische Parallelisierung: Ein möglichst großer Anteil des Algorithmus wird auf den Einzelbildern der linken und rechten Kamera berechnet. Diese Berechnungen werden parallel ausgeführt. (Zeitersparnis ca. 150 Millisekunden)

2. mikroskopische Parallelisierung: Programmstücke, die zeilenweise im Bild (Streifensegmentierung) oder für viele Punkte der Punktewolke (Punktewolke erstellen; PCA) durchgeführt werden müssen, können parallel berechnet werden. Als Anzahl der Threads wurde hierbei die Anzahl der Prozessoren gewählt (4).

3. Möglichst große Programmanteile werden in C++, statt in der langsameren Interpretersprache HDevelop (Halcon) ausgeführt (siehe Tabelle 4.6).

Nach allen Verbesserungen zur Beschleunigung des Algorithmus wurde die Laufzeit für verschiedene Bauteile gemessen. In Tabelle 4.6 ist exemplarisch eine solche Messung nach Programmabschnitten aufgeschlüsselt abgetragen. Es handelt sich hierbei um den Mittelwert aus 25 Wiederholungen einer Objekterkennung des Bauteils Bremskolben. Dieser hat ca. 9400 ($\pm$100) gemessene Objektpunkte, was bezüglich des Bauteilekoffers eine mittlere Objektgröße ist. Gemessen wurde auf einem Quadcore 3GHz Prozessor bei einer Kameraauflösung von 768x576 Pixeln. Hierbei wurden Programmteile auf maximal 4 Threads parallel ausgeführt.

Der Gesamtlaufzeit wurde hierbei mittels Systemuhr ab dem Zeitpunkt, zu dem der Roboter über dem Objekt ausgerichtet ist, bis zur fertig berechneten Zielposition des Roboters aufgenommen, also ab und einschließlich der Bildaufnahme bis zum Start der Roboterbewegung. Der Gesamtalgorithmus kann dabei in drei Teile gegliedert werden.

1. Bildaufnahme und Einzelbildverarbeitung (Feature Extractor). Dieser Teil wird parallel für das linke und rechte Bild durchgeführt. Hier ist die längere der beiden Zeiten aufgeführt.

2. Tiefenbildberechnung und Segmentierung: Aus den vorverarbeiteten Einzelbildern wird ein Tiefenbild berechnet und auf diesem die Objektpunkte vom Hintergrund getrennt. Dafür wird grob segmentiert, indem alle Punkte, die hinter einer Hintergrundebene liegen, vernachlässigt werden, als auch eine formbasierte Erkennung (Shape Based Matching) durchgeführt.

3. Der letzte Teil ist die eigentliche Objektlageerkennung, die aus Groblageerkennung (PCA) und Feinlageerkennung (ICP - mit 20 Iterationen) besteht. Die 20 Iterationen wurden empirisch als Optimum ermittelt, da sich bei mehr Iterationen keine Verbesserung der Schätzung mehr ergab (siehe Abschnitt 4.2.2)

Teile des Programms wurden wegen der schnelleren Laufzeit und besseren Handhabbarkeit in C++ implementiert. Für andere Teile wurden Algorithmen aus der Bildverarbeitungsbibliothek Halcon der Firma MVTec [Gmb80] verwendet. Diese werden über eine Engine (HDevelop) angesteuert. Beide Programmteile sind in der Tabelle 4.6 entsprechend bezeichnet.

Die Standardabweichung der Gesamtlaufzeit lag bei 20ms also ca. 5% der Laufzeit. Messungen der gleichen Art wurden auch mit anderen Bauteilen durchgeführt. Hierbei war festzustellen, dass fast ausschließlich die Anzahl der rekonstruierten Punkte, also die Objektgröße, Einfluss auf die Laufzeit hatte. Bauteile aus dem Teilesatz haben zwischen 2000 (BTN 3) und 20 000 (BTN 32, sehr groß im Bild aufgenommen) Objektpunkte und die mittlere Laufzeiten waren deshalb entsprechend zwischen ca. 320 ms und 500 ms.

Programmabschnitt	**Laufzeit in Millisekunden**	(Zwischen-)Summe der Laufzeiten
1.1 Hdev: Bild aus Kamera auslesen und rektifizieren	21	
1.2 Hdev: Streifensegmentierung	110	
1.3 Übergabezeit (Hdev - C++)	74	
1.4 C++: Lücken füllen	8	
Zwischensumme 1: Bildaufnahme und Einzelbildverarbeitung		226
2.1 C++: Disparitätsberechnung	10	
2.2 C++: Hintergrundsegmentierung	11	
2.3 Hdev: Shape Based Matching	82	
2.4 Hdev: Region Growing	9	
2.5 Übergabezeit (Hdev - C++)	26	
2.6 C++: Punktewolke erstellen	6	
Zwischensumme 2: Tiefenbildberechnung und Segmentierung		186
3.1 C++: Hauptachsenfit (PCA)	13	
3.2 C++: 3D-Punktewolkenfit (ICP)	24	
Zwischensumme 3: Objektlageerkennung		40
Summe der Einzelzeiten		424
gesamte Zeit (gemessen)		**462**

Tabelle 4.6: Geschwindigkeitsmessung nach allen Verbesserungen. Beispiel: Zylinder (BTN 101) auf 3GHz Quadcore einschließlich Bildaufnahme, Segmentierung, Objektlageerkennung und Roboterpositionsermittlung.

Die Forderung einer Laufzeit von 500 Millisekunden wurde hiermit erreicht. Die Diskrepanz zwischen der Summe der Einzelzeiten (vorletzter Eintrag der Tabelle) und der gemessenen Gesamtzeit (letzter Eintrag der Tabelle) ist einerseits durch Laufzeitverluste wegen Datentransfer zwischen Modulen (erstere werden *innerhalb* der Module gemessen, zweitere ist die Systemzeit vor und nach dem Gesamtprogramm), sowie andererseits durch Nebentasks, die zwar nicht zum Bildverarbeitungsalgorithmus gehören, für die Messung aber notwendig sind (z.B. die Dateizugriffe zum Speichern der gemessenen Zeiten) zu erklären.

Die Hauptzeit innerhalb des Algorithmus wird für die Bildaufnahme, Rektifizierung und Segmentierung verwendet. Dies liegt daran, dass eine große Datenmenge (Bilder) zunächst von der Kamera in den PC transferiert und bearbeitet werden muss. Des Weiteren ist der Transfer von Daten zwischen C++ und der Entwicklungsumgebung Halcon zeitintensiv. Ferner ist HDev eine Interpretersprache, was Laufzeiteffizienz für alle dort ausgeführten Programmteile kostet.

Weitere Verbesserungsmaßnahmen wären daher der Verzicht auf die Entwicklungsumgebung HDevelop zu Gunsten von C++-Implementierungen. Um noch größere Beschleunigungsfaktoren zu erreichen, könnte ein Teil oder der gesamte Algorithmus auf die Kamera verlegt werden (SmartKamera). Hierdurch sind Tiefenbildrekonstruktionen in Echtzeit (20 Hertz) schon heute realisierbar. Die Messung hat gezeigt, dass die eigentliche Objektlageerkennung nicht sehr rechenintensiv ist und maschinennah implementiert wohl auch damit Echtzeit erreicht werden kann.

4.2.2 Parametrisierungsmessungen

Als ersten Parameter soll die geeignete Anzahl von Iterationen des ICP-Algorithmus bestimmt werden. Das Diagramm 4.4 zeigt die Abweichung gegenüber dem Groundtruth der Objektlageerkennung am Beispiel des Bauteils 32 (Steckergehäuse) in Abhängigkeit von der Anzahl der Iterationen des verwendeten ICP-Algorithmus. Zu Grunde liegt hierbei eine Messung vom Typ A (20 Messungen aus der selben Ausgangslage). Null Iterationen des ICP bedeutet, dass die Güte der Lageschätzung allein mit der Groblagebestimmung (PCA) aufgetragen ist. Es ist klar ersichtlich, dass der Schätzungsfehler mit der Anzahl der Iterationen bis 20 Iterationen abnimmt. Dannach nimmt er weiter leicht ab (Translationsabweichung - blau) bzw. wieder leicht zu (Rotationsabweichung - rot).

Dieses Verhalten ist typisch für praktisch alle Messungen in sofern, dass sich die Güte bis 10 oder 20 Iterationen verbessert und dannach entweder etwas besser wird oder sogar etwas schlechter (manchmal in der Translation, manchmal in der Rotation). Die Verbesserung ist klar - der iterative Algorithmus konvergiert. Die Verschlechterung ist vielleicht dadurch zu erklären, dass die Güte an der Auflösungsgrenze angelangt ist. Beachte: der Abstand zweier Streifen auf dem Objekt und damit der Abstand zweier Messpunkte in y-Richtung auf dem Objekt beträgt je nach Arbeitsabstand etwa 1,3 mm, die hier gemessene Güte nach 10 Iterationen ist aber schon bei etwa 0,13 mm. Das sind 10% der Auflösung, was überlicherweise als Daumenwert für die untere Schranke eines solchen Verfahrens angegeben wird.

Bei dieser Genauigkeit ist der Fehler dadurch zu erklären, dass die Messpunkte bei jeder Messung an einer leicht unterschiedlichen Stelle auf dem Objekt liegen. Waren die Streifen etwa beim Modelltraining zufällig ein Gradbruchteil verdreht gegenüber dem Mittel, führt dies bei der Objektlageerkennung zu einem Fehler in dieser Größenordnung. Ferner konkurriert dann bei vielen Iterationen die tatsächliche Ausrichtung der Objektes mit dem genauen Passen der Messpunkte auf der Objektoberfläche. Dies kann zu einer leicht zunehmenden Abweichung bei mehr als 20 Iterationen führen. Die Versuche wurden mit 20 Iterationen als Einstellung des Parameters durchgeführt, da diese manchmal noch etwas

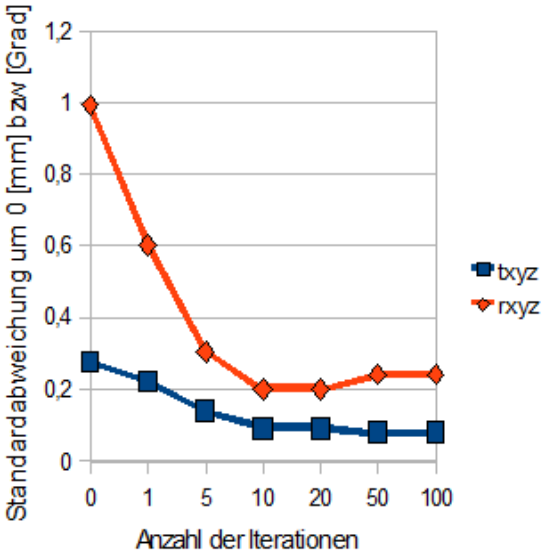


Abbildung 4.4: Diagramm zur Ermittlung der besten Anzahl der Iterationen des ICP-Algorithmus (Wurzel aus dem Fehlerquadrat der Abweichungen gegenüber Groundtrouth in Abhängigkeit von der Anzahl der Iterationen) am Beispeil des Bauteils 32 (Steckergehäuse). In blau (txyz) Translationsabweichung in mm; In rot (rxyz) Rotationsabweichung in Grad

genauer waren, als nur 10 Iterationen und dieser Teil des Algorithmus mit 24 ms (20 Iterationen) nur einen sehr kleinen Teil der Gesamtlaufzeit ausmacht. Weniger Iterationen führen auch nur zu leicht verbesserter Laufzeit (ca. 18 ms bei 10 Iterationen). Mehr Iterationen verschieben die Güte manachmal etwas zum Besseren aber manchmal auch etwas zum Schlechteren und führen zu einer etwas verlängerten Laufzeit.

4.3 Fehleranalyse

In diesem Abschnitt werden sowohl systematische als auch statistische Fehler besprochen, die bisher noch nicht erläutert wurden, bei den Messreihen zur Genauigkeitsuntersuchung aber auftraten.

Betrachten wir zunächst die Randbedingungen der Messungen, insbesondere den Ground Truth. Er ist Voraussetzung, um abschätzen zu können, wie genau eine Messung war. Zunächst sei hier die Modellgenauigkeit erwähnt. Wie oben beschrieben, wird die Ausgangslage des Roboters über dem Objekt aus *einer einzelnen* Objektlagebestimmung während des Modelltrainings bestimmt. Diese Objektlage wird nach Augenmaß optimiert (mehrere Aufnahmen und nur, wenn die Rekonstuktion „gut“ aussah, wurde die Messung als Modell genommen), unterliegt aber prinzipiell einer statistischen Streuung, wie sie in der Messung vom Typ A gemessen wurde. Liegt die Objektlageschätzung für die Modellierung zufällig am Rande der Statistik, ergibt dies einen Fehler für alle folgenden Messungen mit diesem Modell.

Sobald die Ansicht auf das Objekt variiert wird (Messreihen von Typ B und C), ergeben sich weitere Fehlerquellen. Die erste ist die Annahme, dass uns der Roboter eine exakte Position über dem Objekt liefert. Dies stimmt aber natürlich nicht genau. Üblicherweise wird von Roboterherstellern keine absolute Genauigkeit angegeben, sondern nur eine (sehr viel bessere) Wiederholgenauigkeit. Für eine exakte Fehlerabschätzung wird jedoch erstere benötigt. Es gibt zwei Gründe weshalb die Roboterposition nicht exakt ist.

Erstens kann der Roboter nicht jede Position stufenlos anfahren. Er hat so etwas wie eine kleinste Bewegungseinheit. Das Roboterinterface bietet aber die Möglichkeit, nach Anfahren der Position, die der gewünschten Position am nächsten liegt, diese auszulesen. Da die Abweichung des Modells vom Ground Truth mit der ausgelesenen Position durchgeführt worden ist, fällt diese Ungenauigkeit nicht ins Gewicht.

Zweitens biegt sich der Roboterarm je nach Position im Raum unterschiedlich stark durch. Die Arme sind normalerweise so steif wie möglich gebaut, aber eine minimale Abweichung ist unvermeidlich. Da aber nur lokal verfahren wird, ist diese Abweichung von der Wiederholgenauigkeit ebenfalls sehr gering.

Als Wiederholgenauigkeit wird bei dem bei uns verwendeten Robotermodell 0,03 mm (Translation alle 6 Achsen aufaddiert), sowie $6 \cdot 0,1 \cdot 10^{-3}\text{Grad} = 0,0006\text{Grad}$ angegeben, was selbst in der sehr viel schlechteren Translation eine Größenordnung besser als die erwartete Güte unseres Algorithmus ist.

Bei Messungen der Typen B und C ändert sich die Ansicht des Objekts. Dies hat Einfluss auf mehrere Teile des Algorithmus.

- Die formbasierte Erkennung zur Segmentierung des Bauteils vom Hintergrund basiert auf einer Schablone, die aus der Modellansicht aufgenommen wurde. Je stärker die Ansicht verkippt ist (Translation ist bis auf die Skalierung egal), desto schlechter passt die Schablone. Das führt zu einigen Objektpunkten, die als Hintergrund markiert werden und umgekehrt.
- Die Hauptachsenanalyse ist ansichtsabhängig (siehe nächster Abschnitt)
- Durch die Variation der Ansicht verändern sich die Lichtverhältnisse bezüglich Fremdlicht. Dies ist einer der Einflussfaktoren für Fehlrekonstruktionen. Fehlrekonstruktionen wiederum führen zu Ungenauigkeiten in jedem Schritt des Algorithmus. Besonders stark schlägt dies bei Oberflächen zu Buche, die ohnehin schlechte Reflektionseigenschaften haben (d.h. die wenig lambertsch streuen).

Betrachten wir nun die unterschiedlichen Teile des Algorithmus der Reihe nach. Diese können sowohl beim Modelltraining, als auch bei der späteren Objektlageerkennung Anwendung finden.

Beim Modelltraining werden auch die Belichtungsparameter der Kamera eingestellt. Hierbei muss ein Kompromiss zwischen hellen und dunklen Oberflächen des Bauteils gefunden werden. Eine besondere Herausforderung stellen deshalb Bauteile dar, die beides enthalten. Ein Bespiel hierfür ist Bauteil 29 (Ventil), das sowohl Metall als auch schwarze Plastikoberflächen enthält. Es kann stellenweise zu Überblendung (Kamerapixel voll ausgesteuert und möglicherweise Überlaufen zu Nachbarpixeln) oder zu sehr dunklen Pixeln führen. Beides führt zu Farbfehlsegmentierung (siehe unten) und damit Fehlern im Tiefenbild. Da die Kameras auf einem beweglichen Arm neben dem Projektor montiert sind, können Hitzeentwicklung und Bewegung zu Aufnahmefehlern führen. Üblicherweise fällt dabei ein ganzes Bild aus. Dies wird erkannt und die Aufnahme wiederholt.

Der selben Hitzeentwicklung und Erschütterung ist auch der Projektor ausgesetzt. Durch seine Konstruktion kann sich dabei die Position des durchleuchteten Musterbildes leicht verändern, was zu einer schleichenden Dekalibrierung des Projektors zur Kamera führen kann. Die Messungen wurden aber an einer nicht im Dauerbetrieb verwendeten Apparatur durchgeführt, um diese Fehlerquelle zu minimieren.

Um die Hitzeentwicklung zu minimieren, wird der Projektor nicht dauerhaft, sondern nur geblitzt eingeschaltet. Dies führt zwar zu einer eingeschränkten möglichen Belichtungszeit für die Kamera (30ms), ist aber unumgänglich. Um die Lichtausbeute zu maximieren

und den Fremdlichteinfluss zu minimieren, muss die Belichtungszeit genau parallel zur Beleuchtung durch den Projektor erfolgen. Der Projektor wird daher von der Kamera ausgelöst, und gleichzeitig die Bildaufnahme verzögert, um die Verzögerung der Elektronik vom Trigger bis zur tatsächlichen Beleuchtung auszugleichen. Bei leichter Fehleinstellung dieses „Delays“ kann es zu erhöhtem Fremdlichtanteil im Bild kommen, was die Qualität des Musters im Bild schmälert. Ferner muss in der Anwendung teilweise Fremdlicht (Deckenbeleuchtung) zugelassen werden, was zu zusätzlicher Verschlechterung der Bildqualität führt und vor allem Glanzflecke durch Veränderung der Kameraposition auf dem Objekt an variierender Position erzeugt.

Um die Tiefenbildrekonstruktion zeiteffizient durchführen zu können, werden die Kamerabilder rektifiziert und in den Stereonormalfall gebracht. Hierbei wird das Kamerabild mittels einer während der Kalibrierung erstellten Maske verzerrt (siehe Abschnitt 3.6). Dabei werden die Bildpixel jeweils aus anderen Pixeln berechnet, was an sich schon zu Fehlern führen kann. Auch die Kalibrierung selbst ist fehlerbehaftet, was die Bildqualität mindert. Des Weiteren muss mit Fixfokus gearbeitet werden (bisher ist kein Kamerasystem auf dem Markt, mit dem mehrere Fokuseinstellungen kalibriert werden können), wodurch die Variation von z bei endlicher Blendengröße zu einer leichten Verschlechterung der Bildschärfe führt. Ferner können die Schärfentiefe von den beiden Kameras und dem Projektor leicht unterschiedlich sein, was zu Fehlern vor allem im Randbereich führt.

Der nächste Schritt ist die Segmentierung der Farbmusterstreifen im Bild. Probleme treten hierbei vor allem bei sehr hellen und sehr dunklen Stellen auf. Ein schwarzes (Intensität auf allen Farbkanälen gleich Null) oder weißes (Intensität auf allen Farbkanälen voll) Pixel ist per Definition farblos. Durch den bei der Kalibrierung durchgeführten Weißabgleich werden die drei Farbkanäle der Kamera unterschiedlich gewichtet. Dies führt dazu, dass die Farbe sehr dunkler und sehr heller Pixel, besonders bei seit dem Weißabgleich veränderten Beleuchtungsbedingungen /-positionen, sowie bei stark farbigen Objekten falsch erkannt werden kann. Bei unserem Weißabgleich führte dies für gewöhnlich zu roten Streifen an sehr dunklen Stellen, die eigentlich eine andere Farbe haben.

Farbfehlsegmentierungen können während der Tiefenbildrekonstruktion teilweise ausgebessert werden. Da durch die Kalibrierung von linker zu rechter Kamera, sowie durch die Kalibrierung jeder der beiden Kameras zum Projektor eine Redundanz erreicht wurde, können Schätzungen auf Plausibilität überprüft werden. Beispielsweise ist eine Musterfolge von vier roten Streifen (könnte in einem sehr dunklen Bereich fälschlich geschätzt werden) nach Projektorkalibrierung nur in einem sehr eingeschränkten Bereich des Bildes möglich wenn man die Schärfentiefe mit einbezieht. Diese Plausibilierung erkennt jedoch nicht jede Fehlzuordnung so dass mit einer gewissen Anzahl an Fehlrekonstruktionen im Tiefenbild zu rechnen ist. Ferner können Sprünge in der Objektoberfläche zu fehlerhaften Farbstreifenfolgen führen, wie [Dos09] in seiner Arbeit ausführt.

Des Weiteren wurde, um die Dichte der Tiefenkarte auch bei schmalen Bauteilstrukturen zu erhöhen, eine Rekonstruktion bei Musterlängen von weniger als 4 Streifen eingeführt. Diese sind jedoch nur lokal eindeutig, d.h. nur unter der Bedingung, dass sie sich in einem bestimmten Musterabschnitt befinden. Deshalb funktioniert dies nur, wenn Musterstücke rechts und links davon bereits erkannt wurden. Liegt hierbei jedoch eine Fehlerkennung vor, wird dies auch zu weiteren Fehlrekonstruktionen führen.

Ähnliche Fehler können auftreten, wenn Streifen von Bildzeile zu Bildzeile aufgefüllt werden sollen. Sind Musterabschnitte oben und unten falsch oder liegt ein Sprung in der Bauteiloberfläche vor, so kann dies zu weiteren Fehlern führen.

Der Grobfit per Hauptachsentransformation hat vier Schwachstellen:

1. Es kann eine Rotationssymmetrie um eine Nicht-Hauptachse der Punktewolke vorliegen
2. Das Objekt kann nichtsymmetrisch sein, die gemessenen Punktewolke ist jedoch (nahezu) symmetrisch.
3. Keine der Achsen könnte nahezu parallel zur Bildebene stehen.
4. Die Bestimmung der Hauptachse kann chaotisch sein oder es können Achsen vertauscht werden.

Mit dem hier vorgesellten Verfahren werden nicht die tatsächlichen Trägheitsachsen bzw. Hauptachsen des Objekts berechnet, sondern nur die Hauptachsen der *gemessenen Punktewolke*. Dabei ist erstens nur die Oberseite des Objektes zu sehen, womit der Schwerpunkt meist etwas zu hoch (nah an der Kamera) geschätzt wird, und zweitens werden alle gemessenen Punkte gleich stark gewichtet. Die Dichte der Messpunkte hängt aber von der Kooperativität der Oberfläche an der jeweiligen Stelle, sowie von möglichen Glanzflecken ab. Dadurch kann es zu einer Verschiebung des Schwerpunktes und im schlechtesten Fall zu einer Veränderung der Hauptachse kommen. Dass man die Unterseite nicht sieht, führte in der Praxis nur wenig zu Fehlern, da man sowohl bei der Modellgenerierung, als auch bei der späteren Messung einen ähnlichen Ausschnitt betrachtet. Die gleiche Gewichtung aller gemessenen Punkte führte jedoch durchaus zu Fehlerkennungen, wie im nächten Kapitel ausgeführt ist.

Der Bauteilesatz enthält Objekte, die zwar eigentlich nicht symmetrisch sind, deren Punktewolke aber nahezu symmetrisch ist. Bei diesen Bauteilen wurde in der Tabelle ein Kreuz in „Symmetrie ignoriert“ gesetzt. Ein typisches Bespiele wäre Bauteil Nummer 16 (Magnetkern, Abbildung A.13). Siehe auch Abschnitt 4.1.2.

Liegt keine Objektsymmetrie vor und liegt auch keine der Achsen nahezu senkrecht zur Bildebene, kann ein Freiheitsgrad der Objektlage nicht aufgelöst werden, da die aus dem Shape Based Matching erhaltene Rotationslage nicht verwendet werden kann. Es kann zu 180 Grad-Fehlschätzungen kommen.

Das Verfahren hat als zwingende Voraussetzung, dass die Hauptachsen im Objekt immer gleich geschätzt werden. Durch Messfehler kann es bei ungünstigen Objekten jedoch vorkommen, dass die Hauptachse an einer ganz anderen Stelle in der gemessenen Punktewolke liegt, als dies für das Objekt richtig wäre. Eine Fehlschätzung ist in diesem Fall unvermeidlich. Ferner kann passieren, dass zwei Hauptachsen vertauscht werden, da in der Messung zufällig durch fehlende Objektpunkte die Größe der Ausdehung der Punktewolke (Eigenwerte) vertauscht ist. Ein dritter Fehler kann auftreten: Es gibt Objekte, deren Hauptachse verdreht wird, sobald ein bestimmter Bereich nicht rekonstruiert wurde (siehe Tabelle 4.4 unter „Hauptachse nicht eindeutig“). Insbesondere tritt dies bei rechteckigen Objekten auf, wie oben beschrieben. Beispiele für solche Objekte sind Bauteil 7 (Leiterplatte, Abbildung A.7) und Bauteil 32 (Steckergehäuse, Abbildung 4.2)

Kommen wir zum letzten Teil des Algorithmus, dem ICP. Um eine möglichst kurze Laufzeit zu erreichen, wird in dem von uns verwendeten Algorithmus die Datenmenge möglichst schnell reduziert, in diesem Falle auf Normalenvektoren. Dies führt jedoch zu Fehlern bei bestimmten Bauteilen. Handelt es sich z.B. um ein Objekt ohne charakteristische Erhebung wie etwa, unter vielen anderen, die Leiterplatte (Bauteil 7: Abbildung A.7), so ist die *Kante* des Objekts die einzige für den Algorithmus erkennbare Stelle für den x- und y-Translationsfreiheitsgrad, sowie die Rotationsausrichung um die z-Achse. Die Kante ist jedoch nicht nur durch den Algorithmus schlecht rekonstruiert, sondern besonders bei flachen Objekten kann es passieren, dass von der Seitenwand überhaupt keine Punkte rekonstruiert werden können. Das führt zu Oberflächennormalen, die alle in dieselbe Richtung

zeigen. Die erwähnten Freiheitsgrade können durch den Algorithmus in einem solchen Fall nicht bestimmt werden.

Dies führt zu einem oder mehreren Freiheitsgraden in der Einpassung, die praktisch nicht oder nur sehr schlecht bestimmt werden können, oder deren Genauigkeit sehr sensibel auf Fehlrekonstruktionen reagiert (z.B. wenn eine auszeichnende Struktur sehr fein ist und deshalb nicht aus jedem Blickwinkel erkannt werden kann). Da viele Bauteile bei ihrer Größe im Verhältnis zu der Auflösung unserer Tiefenbildrekonstruktion sehr grobe Formen haben, kommt diese Schwäche nicht selten vor. Wenn der ICP an dem gesuchten Freiheitsgrad nichts verbessern kann, wird nur die Schätzung des vorangegangenen Schrittes - der Hauptachsentransformation - verwendet. Trotz der Schwäche oft lokal nur wenig gekrümmter Oberflächen verbessert die Anwendung des ICP die Güte der Objektlageschätzung um einen Faktor 2 bis 10 (siehe Abschnitt 4.2.2).

4.4 Zusammenfassung und Ausblick

Wir haben gesehen, dass der vorgestellte Algorithmus die geforderten Anforderungen an Geschwindigkeit und Genauigkeit bei einer Bauteileabdeckung von 42 % erfüllt.

Die meisten Ausreißer bei der Genauigkeit traten bei starker Verkippung auf. Dies liegt daran, dass sich sowohl die rekonstuierten Oberflächenpunkte und damit die Hauptachsenschätzung, als auch der Umriss und damit die SBM-Schätzung mit der Ansicht ändern. Ein Ausweg für beide Algorithmenteile würde das mehrfache Training des Objekts aus unterschiedlichen Ansichten bieten. Damit könnte im selben Zuge bei gutmütigen Objekten eine größere Ansichtsvariation toleriert werden. In der Shape-Based-Matching Phase würde dann die Ansicht ausgewählt werden, deren Umrissschablone am besten auf das Objekt passt. Neben der Gefahr einer Fehlzuordnung ginge dieser Schritt in jedem Falle massiv auf Kosten der Laufzeit. Da dieser Algorithmenteil schon mit parallelisierten Operatoren arbeitet, würde sich die Zuordnungszeit bei n Ansichten in etwa ver-n-fachen. Tabelle 4.6 zeigt, dass das SBM mit 82 Millisekunden etwa 20% der gesamten Laufzeit ausmacht. Bei nur fünf weiteren Ansichten (etwa die Menge, die für eine Erweiterung des Winkelbereichs in jede Richtung um weitere 10 Grad benötigt würde) ist das eine Verdoppelung der Gesamtlaufzeit.

Die Haupteinschränkung im darauf folgenden Schritt, der Groblageerkennung mittels Hauptachsentransformation, ist die Einschränkung bei Objekten mit Symmetrien. Die Punktewolke war bei unserem Messsystem und den gegebenen Bauteilen oft nicht gut genug aufgelöst, um kleine Details zu erkennen, die maßgeblich für die Asymmetrie eines Bauteils waren. Ferner mussten Symmetrien per Hand angegeben werden.

Zur Vereinfachung könnten die Symmetrien auch vom Algorithmus geschätzt werden. Die Hauptachsenzerlegung liefert neben der gesuchten Orthonormalbasis des Objekts, die durch die Eigenvektoren aufgespannt wird, auch die Eigenwerte. Die Größe der Eigenwerte gibt an, wie stark die Punktewolke entlang der entsprechenden Achse ausgedehnt ist. Sind mindestens zwei Eigenwerte nahezu gleich, hat das Objekt für den Algorithmus eine Symmetrie, die entweder einer tatsächlichen Symmetrie entspricht und ignoriert werden kann oder die Fehlerwahrscheinlichkeit drastisch erhöht.

Maßgeblich für die Erkennungsgüte war durch ihre Hebbarkeit aber keiner der genannten Faktoren. Die Analyse hat ergeben, dass:

1. Die Hauptursache für Nichtanwendbarkeit des Algorithmus auf ein Bauteil, die bruchstückhafte oder statistisch stark schwankende Rekonstruktion des Tiefenbildes ist.

2. Die Hauptursache für schlechte Rekonstruierbarkeit Glanzflecken sind.

Das Hauptaugenmerk sollte also auf der Lösung dieses Problems liegen.

Tiefenbildrekonstuktionsverfahren durch Triangulation, wie das hier vorgestellte Verfahren, beruhen darauf, dass ein Objektpunkt in der linken Kamera, dem entsprechenden Punkt in der rechten Kamera zugeordnet werden kann (Korrespondenzpunktsuche). Ist hierfür nicht genug Textur auf den Objekten vorhanden, wie bei 90 % unserer Bauteile der Fall, so kann diese durch einen Projektor von außen aufgebracht werden. Voraussetzung hierfür ist jedoch, dass das projizierte Bild von beiden Kameras am selben Objektpunkt gesehen wird. Dies ist genau dann der Fall, wenn das Objekt das Licht vorwiegend lambertsch reflektiert, also matt und nicht lichtdurchlässig oder stark strukturiert ist. Bei polierten Metalloberflächen (viele der Bauteile enthalten solche), wird der lambertsch gestreute Lichtanteil oft stellenweise vom reflektierten Anteil überblendet, was bei uns unter „Glanzfleck“ zusammengefasst wurde. Das Problem ist damit also nicht ein algorithmeninhärentes Problem, sondern ein *prinzipielles* Problem von Triangulationsverfahren!

Nebenbei bemerkt führen stark reflektierende Oberflächen durch Doppelreflektionen auch bei der anderen großen Klasse von Tiefenbildrekonstruktionsalgorithmen, den Laufzeitverfahren, zu ähnlichen Schwierigkeiten. Einzig deflektometrische Verfahren machen sich diesen Umstand zu Nutze. Diese beruhen jedoch auf einer genauen Kontrolle der Beleuchtungsumgebung, welche bei unseren Anwendungen explizit nicht gegeben ist.

Ein mögliche Lösung dieses Dilemmas bietet ein in der Tiefenbildrekonstruktion der Bildverarbeitung bisher nicht sehr breit verwendetes Verfahren, nämlich die Rekonstruktion des gesammten Lichtraums, des Lichtfeldes. Mit den Grundlagen dafür, sowie der Untersuchung einer Anwendbarkeit auf unsere Aufgabenstellung beschäftigt sich Kapitel 5.

5. Lichtfeldtheorie

Aus unseren Messungen in Kapitel 4 konnte geschlussfolgert werden, dass das größte Problem der Objektlageerkennung durch eine Tiefenbildrekonstruktion gelöst werden könnte, die Abstandswerte an Glanzflecken ermitteln kann. Ein solches Verfahren bieten Lichtfelder, welche Gegenstand dieses Kapitels sind.

Hier soll erklärt werden, was ein Lichtfeld ist (Abschnitt 5.1), wie man es messen kann (Abschnitt 5.2) und wie es uns bei Glanzflecken helfen kann (Abschnitt 5.3). Danach folgen Experimente zur Verifikation (Abschnitt 5.4) und deren Bewertung (Abschnitt 5.5).

5.1 Lichtfeld - Was ist das?

Wir betrachten in der vorliegenden Arbeit Objekte, die wesentlich größer als die Lichtwellenlänge sind und inkohärent beleuchtet werden. Dies ist Gegenstand der *geometrischen Optik*, in der Licht als *Strahl* betrachtet werden kann. Betrachtet man die Gesamtheit aller Strahlen, die durch jeden Punkt im Raum gehen, erhält man eine fünfdimensionale Funktion, die jedem Raumpunkt und jeder Richtung eine Helligkeit zuordnet. Diese Funktion heißt *Plenoptische Funktion* $L(x, y, z, \Phi_x, \Phi_y)$. Die beiden Parameter für die zeitliche Entwicklung und die Wellenlänge des Lichts wurden hierbei vernachlässigt.

Heute findet sie als Verständnisgröße Anwendung vor allem in der Computergrafik (Rendering). Ist L einer Szene bekannt, so kann jede Kameraansicht aus ihr generiert werden. Erst neueste Entwicklungen lassen die Betrachtung von Lichtfeldern in der Bildverarbeitung wieder aufleben.

Beschränkt man sich in einer Szene auf ein einzelnes *konvexes* Objekt, so enthält bereits die Plenoptische Funktion auf einer Kugeloberfläche um das Objekt die gesamte Information. Diese Einschränkung der Plenoptischen Funktion heißt *4D-Lichtfeld* $L(x, y, \Phi_x, \Phi_y)$ (der Begriff stammt aus der Computergrafik [ML96]). Eine alternative Repräsentation dieses Lichtfeldes ist die *Zwei-Ebenen-Darstellung* $L(u, v, s, t)$. Hierbei wird die Richtung eines Lichtstrahls durch seine zwei Durchstoßpunkte auf zwei planparallelen Ebenen parametrisiert.

In der Bildverarbeitung spricht man bei einer Repräsentation des 4D-Lichtfeldes auf einer begrenzten Ebene (ähnlich einer Kamera, nur dass zusätzlich die beiden Einfallsrichtungsparameter aufgenommen werden) von einem *Epipolar-Bild (EPI - Epipolar-Image)* [AL08a].

Betrachtet man einen Punkt im Fokus einer Kamera (siehe Abbildung 5.1), so werden alle von ihm ausgehenden Lichtstrahlen von der Kameraoptik auf einen Punkt im Brennpunkt der Linse abgebildet.

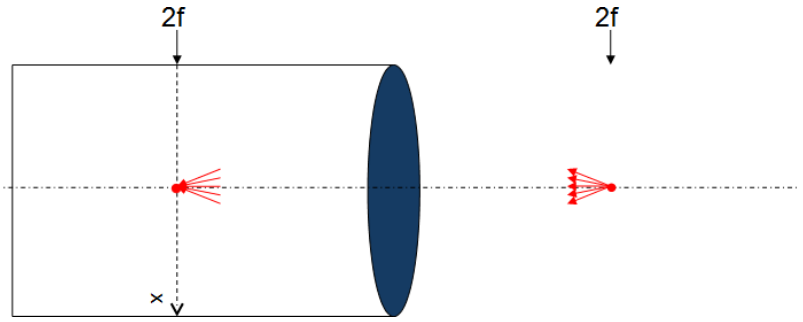


Abbildung 5.1: Schema der Abbildung eines Objektpunktes (rechts) in eine Kamera (links im Bild) - Abbildung aus [Zin]

Abbildung 5.2 zeigt einen 2D-Schnitt durch das Lichtfeld, bei dem in x-Richtung die Pixel einer Zeile (Parameter x des Lichtfeldes) der Aufnahme und in y-Richtung der Blickwinkel entlang dieser Achse (Parameter Φ_x) aufgetragen sind. D.h. in einer anderen Zeile des Schaubildes ist die gleiche Bildzeile einer weiter links oder rechts liegenden *Kamera* gezeigt. Das Diagramm ist hierbei so normalisiert, dass ein Objekt, das sich in der Fokusebene der Kameras befindet, gerade als senkrechte Linie dargestellt wird.

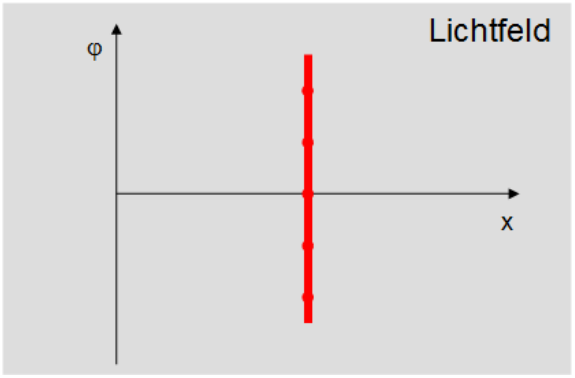


Abbildung 5.2: Trajektorie eines Objektpunktes im x-Φ_X-Lichtfeldschnitt (Abbildung aus [Zin])

Betrachtet man nun einen zweiten Objektpunkt, der näher an unserer Lichtfeldkamera liegt, wie in Abbildung 5.3 dargestellt, und erstellt die Trajektorie dieses zweiten Objektpunktes (blau) im selben 2D-Lichtschnitt, so ist der Punkt in weiter links liegenden Kameras noch weiter links in der Zeile zu finden und in weiter rechts liegenden Kameras weiter rechts, da die Disparität des Objektpunktes größer geworden ist (Abbildung 5.4). Die Trajektorie des Punktes im Lichtschnittbild ist verglichen mit dem weiter weg liegenden Punkt nach links geneigt.

Betrachtet man eine Beispielszene mit drei Objekten mit unterschiedlichem Abstand, so ergibt sich im 2D-Lichtfeldschnitt eine Überlagerung von Trajektorien, wobei die am weitsten nach links geneigten Linien im Vordergrund liegen und daher die anderen Linien verdecken (siehe Abbildung 5.5).

In dem Paper [RZ07] ist eine Methode für diese Tiefenbildberechnung beschrieben. Hierbei wird der 2D-x-Φ_x-Schnitt durch das 4D-Lichtfeld nach und nach immer weiter nach rechts geschert. Man beginnt mit der am stärksten nach links geneigten Linie (Vordergrund)

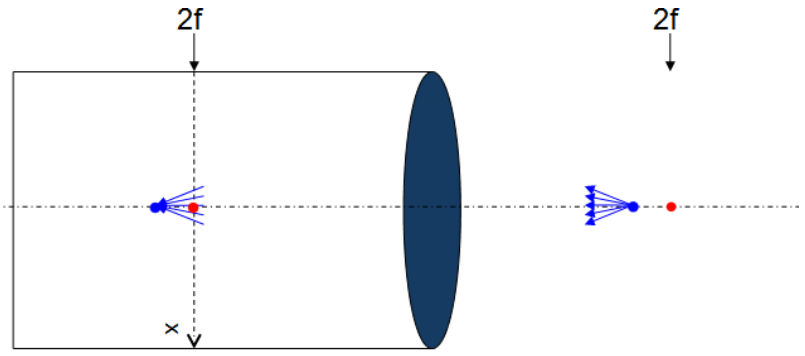


Abbildung 5.3: Schema der Abbildung eines Objektpunktes weiter im Vordergrund (rechts) in eine Kamera (links im Bild) - Abbildung aus [Zin]

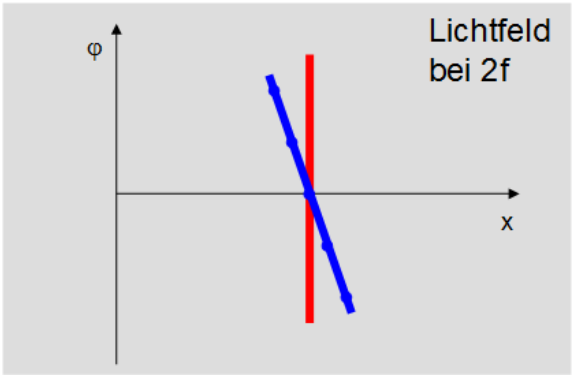


Abbildung 5.4: Trajektorie zweier Objektpunkte unterschiedlichen Abstandes von der Kamera im x-Φ_X-Lichtfeldschnitt (Abbildung aus [Zin])

Abbildung 5.5: x-Φ_X-Lichtfeldschnitt (links) einer Szene mit drei Objekten (rechts) (Abbildung aus [AL08b])

und arbeitet sich bis zum Hintergrund durch. Nach jeder Scherung wird mittels einer Fouriertransformation nach senkrechten Linien gesucht. Ist eine solche gefunden, so merkt man sich die Tiefe des Objektpunktes und löscht die Struktur aus dem Lichtfeld. Damit kann das Tiefenbild erschlossen werden. Abbildung 5.6 zeigt dies in unserer Beispielszene.

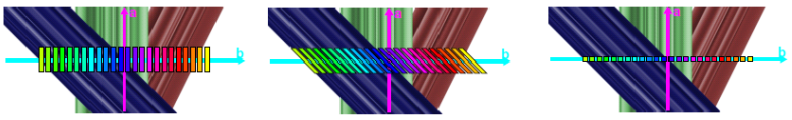


Abbildung 5.6: Drei Scherungen des x-Φ_X-Lichtfeldschnitts unserer Beispielsszene. Die regenbogenfarbenen Streifen zeigen, welcher Teil des Lichtfeldes bei einer normalen Kamera jeweils in ein Pixel fallen würde. Das Pixel enthält dann das Integral über die Intensitäten. Wenn diese Linien parallel zu den Lichtfeldtrajektorien sind, fällt also gerade alles Licht, dass von einem einzelnen Objektpunkt ausgeht in ein Pixel der Kamera - der Objektpunkt wird scharf abgebildet. In jedem anderen Fall ist er verschwommen. Links: mit Fokus auf des grüne Objekt, Mitte: mit Fokus auf das vordergründige blaue Objekt, Rechts: wie mit einer Lochkamera aufgenommen (alle Punkte scharf) (Abbildung aus [AL08b])

Dies hat als Hauptproblem, dass Strukturen erkennbar sein müssen. Ist die Szene untexturiert, so sind auch im 2D-Lichtfeldschnitt keine Linien erkennbar und die Tiefe kann nicht berechnet werden. Der Trick in der praktischen Umsetzung ist also, wie man mit den weitgehend homogenen Bereichen des 2D-Schnittes umgeht. Siehe hierfür auch Abschnitt 5.3.

5.2 Messen von Lichtfeldern

Nimmt man eine Szene mit einer gewöhnlichen CCD-Kamera auf, so werden in jedem Pixel des zweidimensionalen Sensorchips für eine kurze Zeit (Belichtungszeit) alle Photonen gesammelt. Die Anzahl der eingetroffenen Photonen ergeben die Helligkeit des Bildpunktes. Dies ist aber nur ein kleiner Teil der durch die Kameralinse fallenden Bildinformation. In jedem Pixel geht nämlich verloren *aus welcher Richtung* das einfallende Licht kommt. Möchte man aber das Lichtfeld aufnehmen, muss diese Information konserviert werden. Dafür gibt es einfache, praktische Methoden.

Die einfachste Möglichkeit neben dem Ort auf dem Chip die Richtung des einfallenden Lichtes zu erhalten, ist die Aufnahme mit nicht nur einer, sondern einem ganzen Array von Kameras [BW05]. Danach muss ein Korrespondenzproblem gelöst werden, ähnlich wie bei der Tiefenbildberechnung bei Stereokamerasystemen, nur eben mit vielen Kameras. Ist ein Objektpunkt in mehreren Kamerabildern erkannt, so hat man mehrere Blickwinkel auf den Objektpunkt, kann also in den jeweiligen Kameras die Richtung des Lichtstrahls angeben. Die Winkelauflösung entspricht hierbei der Anzahl der Kameras in der entsprechenden Richtung (vorausgesetzt, der Objektpunkt ist in allen Kameras sichtbar).

Die Aufnahme eines Kameraarrays kann in die vorteilhafte Epipolarbilddarstellung umgerechnet werden. Hieraus können einfach Kameraansichten oder ein überall scharfes Bild berechnet werden [SW11].

Alternativ zu einem Kameraarray kann bei statischen Szenen auch eine einzelne Kamera verwendet werden, die exakt verfahren wird. Diese fährt Positionen eines Rasters vor der Szene ab und nimmt jeweils ein Bild auf. Man nennt diese Vorrichtung *Gantry*. Vorteile hat dies durch den geringeren Hardwareaufwand, die große Flexibilität in der Winkelauflösung des Lichtfeldes, sowie die Möglichkeit sehr dichte Bilder aufzunehmen. Der große Nachteil ist jedoch die lange Aufnahmezeit. Trotzdem wird im Rahmen der vorliegenden Arbeit ein solcher Aufbau verwendet (siehe Abschnitt 5.3.2).

Die moderne Kamerachipherstellung ermöglicht sehr große Auflösungen, so groß, dass die Bildgüte moderner Consumer-Kameras nicht durch die Chip-Auslösung, sondern hauptsächlich durch die Optik und andere Parameter (Dunkelrauschen usw.) bestimmt wird. Große Kamerachips ermöglichen aber eine andere Art der Aufnahme eines Lichtfeldes - die *Plenoptische Kamera*.

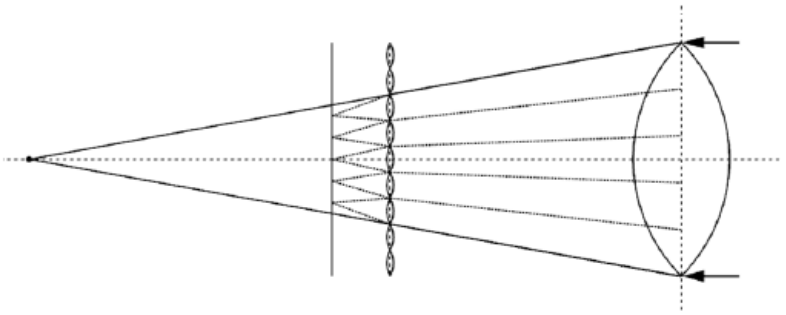


Abbildung 5.7: Die Abbildung zeigt den schematischen Aufbau einer Plenoptischen Kamera mit einem Mikrolinsenarray zwischen Bildebene und Kamerachip. (Abbildung aus [CP11])

Hierbei wird der Sensorchip nicht mehr an die Stelle der Optik des virtuellen Bildes gestellt, sondern etwas nach hinten versetzt. Zwischen virtuellem Bild und dem Sensorchip platziert man ein Mikrolinsenarray (siehe Abbildung 5.7). Dieses Mikrolinsenarray bildet zusammen mit dem dahinterliegenen Kamerachip so etwas wie ein Kameraarray, nur dass es nicht die Szene, sondern das Bild der Szene betrachtet. Das Ergebnis ist ähnlich wie bei einem Kameraarray - man erhält neben der Helligkeit in jedem Bildpunkt zusätzlich die Richtung aus der der Lichtstrahl kam. Dies ist eine Möglichkeit, ein Lichtfeld mit einer kompakten Kamera aufzunehmen. Eine solche Kamera heißt *plenoptische Kamera 1.0*.

Es gibt eine kommerzielle Umsetzung einer plenoptischen Kamera auf dem Markt. Sie stammt von der Firma Raytrix [CP11], allerdings wird hierbei nicht ein Mikrolinsenarray mit lauter gleichen Linsen verwendet, sondern mit drei Linsensorten unterschiedlicher Brennweite. Man nennt diese Art *plenoptische Kamera 2.0*. Dies reduziert zwar die Auflösung, erhöht die Tiefenschärfe aber wesentlich. Wanner et al. [SW11] haben hierfür einen Algorithmus entwickelt, mit dem die erhaltenen Daten in ein 4D-Lichtfeld umgerechnet werden können.

5.3 Zielsetzung und Konzept

Wie in der Einleitung dieses Kapitels formuliert, ist unser Hauptziel die Evaluation einer Tiefenbildrekonstruktion in Hinblick auf Glanzflecke. Als mögliches Verfahren wurden Lichtfelder identifiziert. Warum können diese helfen?

Das Problem bei Stereorekonstruktionen ist, dass Glanzflecke genau wie Texturen auf dem Objekt aussehen und zur Rekonstruktion verwendet werden. Sie sind aber nicht fest auf dem Objekt, sondern ändern ihre Lage mit dem Blickwinkel. Dadurch kommt es zu einer Fehlrekonstruktion.

5.3.1 Zielsetzung der Lichtfeldmessungen

Wir haben in Abschnitt 5.1 einen 2D-Schnitt aus einem Lichtfeld betrachtet. Die Tiefe eines Objektpunktes kann hierbei aus der Neigung seiner Trajektorie im Schnitt ermittelt werden. Ein Glanzfleck bildet eine über verschieden solche Trajektorien hinweg verlaufende Struktur. Das erste Ziel dieser Arbeit soll also sein herauszufinden, *wie Glanzflecke in Lichtfeldern aussehen* und natürlich, ob sie von Objekttexturen unterschieden werden können.

Um aus den Messdaten eines Kamerarrays ein Tiefenbild berechnen zu können, müssen die Objekte der Szene ausreichend Textur besitzen. Letztendlich wollen wir aber untersuchen, wie nützlich Lichtfeldaufnahmen auf dem von uns untersuchten Teilespektrum sind. Da diese Bauteile aber gerade wenig texturiert sind, müssen wir ähnlich wie bei den konventionellen Stereoaufnahmen einen Ausweg finden. Im Rahmen dieser Arbeit soll untersucht werden, ob analog zur Tiefenbildrekonstruktion mit Stereokameras die Beleuchtung der Szene mit einem Muster dieses Problem löst. Das zweite Ziel ist also die Frage: „Kann die Tiefenrekonstruktion aus einem Lichtfeld durch die Verwendung eines Musters verbessert werden?“

Das dritte Ziel ist das Resumee: Ist zu erwarten, dass die Objektlageerkennung durch den Einsatz von Lichtfeldern verbessert werden kann?

Die Ziele sind die Antworten auf die drei Fragen:

1. Kann die Tiefenbildrekonstruktion an Glanzpunkten mit Hilfe von Lichtfeldaufnahmen im Vergleich zu Stereoaufnahmen verbessert werden?
2. Kann die Qualität der Lichtfeldaufnahmen mit einem Musterprojektor auf untexturierten Objekten in gleichem Maße verbessert werden wie bei Stereoaufnahmen?
3. Kann der Einsatz von Lichtfeldern voraussichtlich die Objektlageerkennung verbessern?

Hierfür wurden Objekte aus dem Teilespektrum gewählt, die untexturiert (Magnetkern, Messkörper), teilweise glänzend (Zündkerze) oder stark glänzend (Spule, Bohrer) sind.

5.3.2 Versuchsaufbau der Lichtfeldmessungen

Anstatt ein ganzes Kameraarray aufzubauen, kann das Lichtfeld einer statischen Szene auch gemessen werden, indem man mit einer Kamera viele Positionen anfährt. In unserem Messaufbau ist dies realisiert, indem eine Kamera auf zwei Verschiebetischen montiert wurde (siehe Abb. 5.8). Diese können jeweils in einem Bereich von $\pm 5\,cm$ mit einer Genauigkeit unter $1\mu m$ (vgl. Homepage der Herstellerfirma PI [Phy]) positioniert werden. Die verwendete CCD Kamera hat eine Auflösung von 1280 x 960. Es wurde ein Sampling von ca. 20x40 bis 50x50 (je nach Objekt) Kamerapositionen in einem Abstand von je 1 mm gewählt, jeweils so, dass das Objekt in allen Bildern vollständig zu sehen ist. Damit ergibt sich ein planares Lichtfeld mit einer Ortsauflösung in Größe der Kamerapixelauflösung und einer Winkelauflösung in Höhe der Anzahl der Kamerapositionen in der jeweiligen Richtung. Abbildung 5.8 zeigt eine Fotografie des von uns verwendeten Messaufbaus.

Um mit der hohen Helligkeitsdynamik an glänzenden bzw. matten Oberflächen umgehen zu können, wurde an jeder Position eine Bildserie von vier Bildern mit unterschiedlicher Belichtungszeit aufgenommen. Aus der Bildserie wird dann mittels der sogenannten *HDR-Methode* (engl. **H**igh **D**efinition **R**ange) in der Umsetzung von Markus Jehle (HCI) ein Bild errechnet, als wäre die Aufnahme mit einer Kamera mit logarithmischer Kennlinie aufgenommen worden. Das erhaltene gleitkommazahlwertige Bild wird dann, um eine Datenreduktion zu erreichen, auf 8 Bit pro Pixel abgebildet.

Auf untexturierten Oberflächen ist keine Abstandsberechnung möglich. Deshalb wurde ein schwarz-weiß Binär-Rauschmuster aufprojiziert. Der Projektor entspricht dem bei den Stereobildmessreihen (4) verwendeten, der in Abschnitt 3.1 beschrieben wurde.

Wie soll nun die Güte der aus dem Lichtfeld berechneten Tiefenbilder ermittelt werden? Ein objektives Maß hierfür zu finden gestaltet sich schwer. Die meisten Verfahren benötigen als grundlegende Voraussetzung einen „Ground Truth“, der mit einem anderen Verfahren

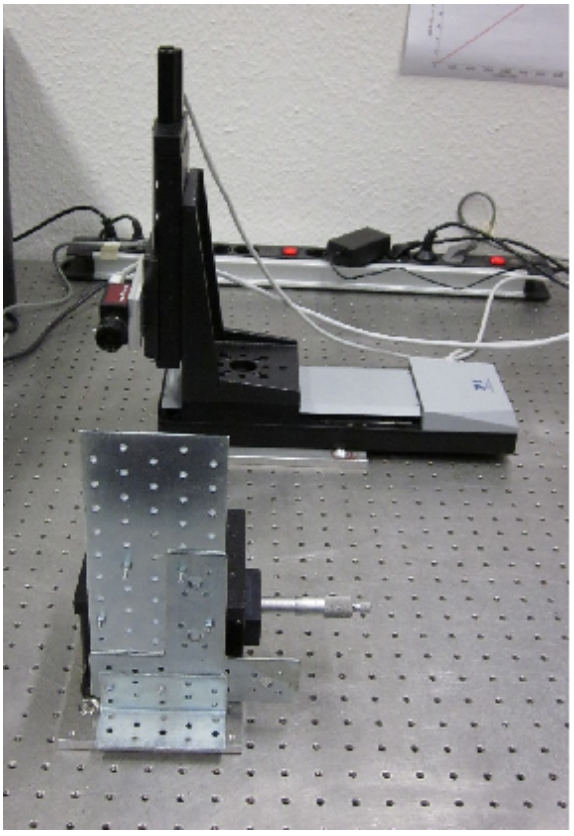


Abbildung 5.8: Fotografie des Gantry-Messaufbaus mit zwei Linearachsen zur genauen Ansteuerung der Kamerapositionen. Die Rückseite des Kalibriertargets ist im Vordergrund zu sehen.

aufgenommen wurde, das mindestens eine Größenordnung genauer ist, als das zu bewertende Verfahren. Dann wird üblicherweise die Abweichung des Abstandswerts in jedem Pixel errechnet. Das mittels einer Metrik berechnete Mittel dieser Abweichung (z.B. mittlere Abstandsquadrat mit der euklidschen Metrik) ergibt die Güte der Tiefenschätzung.

In dieser Arbeit wird dagegen der pragmatische Ansatz verfolgt, die Güte der Tiefenschätzung zu bewerten, indem die eigentliche Aufgabenstellung damit durchgeführt wird. D.h. mit dem Tiefenbild, das aus dem Lichtfeld berechnet wurde, wird eine Objektlageerkennung durchgeführt. Um ein Modell erstellen und ein anderes Bild dagegen registrieren zu können, muss das Objekt also aus mindestens zwei Ansichten aufgenommen werden (jeweils ein ganzes Lichtfeld). Die wahre Objektbewegung muss hierbei bekannt sein (Ground Truth). Daher wurde das Objekt auf einem Kipptisch angebracht, mit dem die Position in zwei Achsen variiert werden kann.

Der Messablauf folgt damit folgendem Schema unter Beibehaltung der internen Kameraparameter:

1. Je eine Lichtfeldaufnahme mit Projektor unter Variation der Objektlage mittels Kipptisch
2. Die gleiche Aufnahmeserie ohne Projektor, dafür mit einer homogenen Beleuchtung
3. Die gleiche Aufnahmeserie ohne Objekt, dafür mit einem Kalibriertarget, um eine externe Kamerakalibrierung und den „Ground Truth“ für die Objektlageerkennung zu ermitteln
4. Eine Bildserie mit einem großen Kalibriertarget unter Positionsvariation zur Ermittlung der exakten internen Kameraparameter, deren Genauigkeit sich als maßgeblich für die Güte des Lichtfeldes herausstellt hat.

5.3.3 Rekonstruktion des Lichtfeldes und der Tiefenbilder aus den Aufnahmen

In diesem Abschnitt wird die Auswertungssoftware beschrieben. Sie wurde am Heidelberg Collaboratory for Image Processing (HCI) erstellt. Die Kamerakalibrieralgorithmen, sowie die Erstellung des Lichtfeldes stammt von Janis Fehr, die Tiefenbildberechnung von Sven Wanner. Teile der Software sind im Internet verfügbar gemacht (für eine genauere Beschreibung und die Internetquelle siehe [SW11]). Die Messsoftware wurde von Markus Jehle in Heurisko geschrieben und von mir angepasst. Alle hier verwendeten Messungen habe ich selbst durchgeführt.

Als erster Schritt müssen nach der Einstellung des Fokus auf das betrachtete Bauteil die *intrinsischen Kameraparameter* (Fokus, Verzeichnung, ...) kalibriert werden. Hierfür wurde eine Kalibrierplatte mit großem Schachbrettmuster erstellt. Von dieser werden in unterschiedlichen Posen Bilder aufgenommen, so dass möglichst das gesamte Messvolumen und jede mögliche Verkippung abgedeckt wird. In den Bildern werden mittels der in OpenCV [ver09] enthaltenen Routine die Lage der Eckpunkte der Quadrate im Bild ermittelt und einem Bündelausgleichsverfahren zur Kamerakalibrierung zugeführt. Die damit bestimmten intrinsischen Kameraparameter werden in den folgenden Schritten verwendet.

Für die Kalibrierung der *extrinsischen Kameraparameter* wird ein kleineres Kalibriertarget (wieder ein Schachbrettmuster auf einer Ebene) an die spätere Position des zu vermessenden Objektes im Bild platziert. Mit dem Gantry werden alle Positionen angefahren, aus denen später das Lichtfeld des Objektes erstellt werden soll (siehe Abschnitt 5.3.2). Mit den gewonnenen Bildern wird wieder eine Kalibrierung ähnlich der Kalibrierung der intrinsischen Kameraparameter durchgeführt (Ermittlung der Eckpunkte des Schachbrettmusters

im Bild), allerdings nur um die Kameraposition gegenüber dem Target zu ermitteln. Jede Kameraposition wird gespeichert und später für die Erstellung des Lichtfeldes aus den Einzelbildern verwendet.

Mit der vollständigen Kamerakalibrierung können die Bauteilbilder zu einem 4D-Lichtfeld zusammengefasst werden. Mit der intrinsischen Kalibrierung wird die Kamera rektifiziert und mit der extrinsischen können die Einzelbilder einem Betrachtungswinkel zugeordnet werden. Das Lichtfeld wird dann mit der in [SW11] beschriebenen Software zu einem 4D-Lichtfeld zusammengefasst, in dem sehr einfach 2D-Schnitte, wie in Abschnitt 5.1 beschrieben, vorgenommen und betrachtet werden können.

Der letzte Schritt ist die Erstellung eines Tiefenbildes aus dem Lichtfeld. Es ist möglich ein Tiefenbild einfach dadurch zu bestimmen, dass zwei Ansichten gewählt werden und ein Stereorekonstruktionsalgorithmus verwendet wird. Wir wollen aber untersuchen, ob es für Lichtfelder eine bei Glanzflecken vorteilhaftere Methode gibt. Ein Ansatz hierfür wurde bereits in Abschnitt 5.1 erklärt. Man schert den 2D-Lichtfeldschnitt von Vordergrund zu Hintergrund und identifiziert die senkrechten Strukturen. Dadurch erhält man den Abstand des zugehörigen Objektpunktes und entnimmt diese sukzessive.

[RZ07] schlägt diesen Ansatz mit einer Fouriertransformation zur Ermittlung der senkrechten Strukturen vor. Allerdings muss die Szene hochfrequente Texturen enthalten, damit die Strukturen in der 2D-Darstellung zu erkennen sind. Die geeignete Textur soll bei unserer Messreihe durch die Beleuchtung mit einem Rauschmuster aufgebracht werden.

Als Tiefenrekonstruktionsverfahren wurde das Verfahren von Wanner in seiner eigenen Implementierung verwendet, das erst in Kürze veröffentlicht werden wird [SW12]. Wie bereits festgestellt, erzeugt ein Objektpunkt eines lambertsch streuenden Objekts im 2D-Schnitt des Lichtfeldes eine Linie konstanter Helligkeit (spekular reflektierte Lichtanteile modulieren eine niederfrequente Helligkeitsänderung auf). Die Steigung dieser Linien lassen sich 1 zu 1 in Abstandswerte des Objetks zur Kamera umrechnen. Zielsetzung ist also, die Steigung solcher Linien zu ermitteln.

Die lokale Steigung wird dabei mittels des Strukturtensors ermittelt. Im Wesentlichen wird also der Gradient an jedem Bildpunkt des 2D-Lichtfeldschnittes gesucht. Um das Ergebnis an Objektkanten zu verbessern, wird die Randbedingung verwendet, dass Objekte im Vordergrund (also Objektpunkt deren Steigung weiter nach links geneigt sind) nicht von Objekten im Hintergrund überdeckt werden können. Diese Bedingung wird als „soft constrained“ in ein Energiefunktional eingeführt. D.h. wenn eine errechnete Steigung an einem Punkt diese Bedingung verletzt, unterliegt sie einem Bestrafungsterm.

Die auf diese Weise verbesserte Richtungskarte dient als Ausgangspunkt einer globalen Optimierung. Hierbei wird das Funktional

$$E(u) = \int_{\Omega} g\,|Du| + \rho(u, x, y)d(x, y)$$

minimiert, wobei $u : \Omega \to \mathbb{R}$ die Funktion ist, die einer Lichtfeldschnittrichtung den Tiefenwert zuordnet, Du entsprechend das Ergebnis der lokalen Gradientenbildung enthält und der zweite Summand einen Bestrafungsterm darstellt, der Entrauschen und Kantentreue herstellen soll. Das Funktional bildet ein konvexes Optimierungsproblem und ist global lösbar. Die genauen Terme sind in [SW12] enthalten.

Als Muster dient ein Zufallsbinärmuster (schwarz - weiß) mit einer Granularität von 720 x 480. Ein Binärmuster, da es auf die Maximierung des Kontrastes auch leicht außerhalb des Kamerafokuses ankommt. Ein Zufallsmuster, da zum Vergleich Stereorekonstruktionsalgorithmen auf die gleichen Bilder angewendet werden sollen, die bei wiederkehrenden

Strukturen Artefakte bilden würden. Die geeignete Granularität wurde empirisch ermittelt. Eine zu feine Granularität hat bei zunehmender Unschärfe zu wenig Kontrast und eine zu grobe Granularität führt zu großen homogenen Flächen im 2D-Schnitt, bei denen die Tiefe nicht mehr genau bestimmt werden kann. Sie geht also zu Lasten der Tiefenauflösung.

5.4 Ergebnisse der Messreihen

Als erstes Objekt für unsere Untersuchung wurde ein Messkörper gewählt. Das ist ein Metallzylinder, der mit einem matten grauen Lack beschichtet wurde und somit sehr kooperativ bei der Abbildung des Rauschmusters ist, an sich aber kaum Textur hat. Abbildung 5.9 zeigt das zentrale Bild des Lichtfeldes des Messkörpers mit homogener Beleuchtung, sowie das entsprechende Bild mit aufprojiziertem Binärmuster.

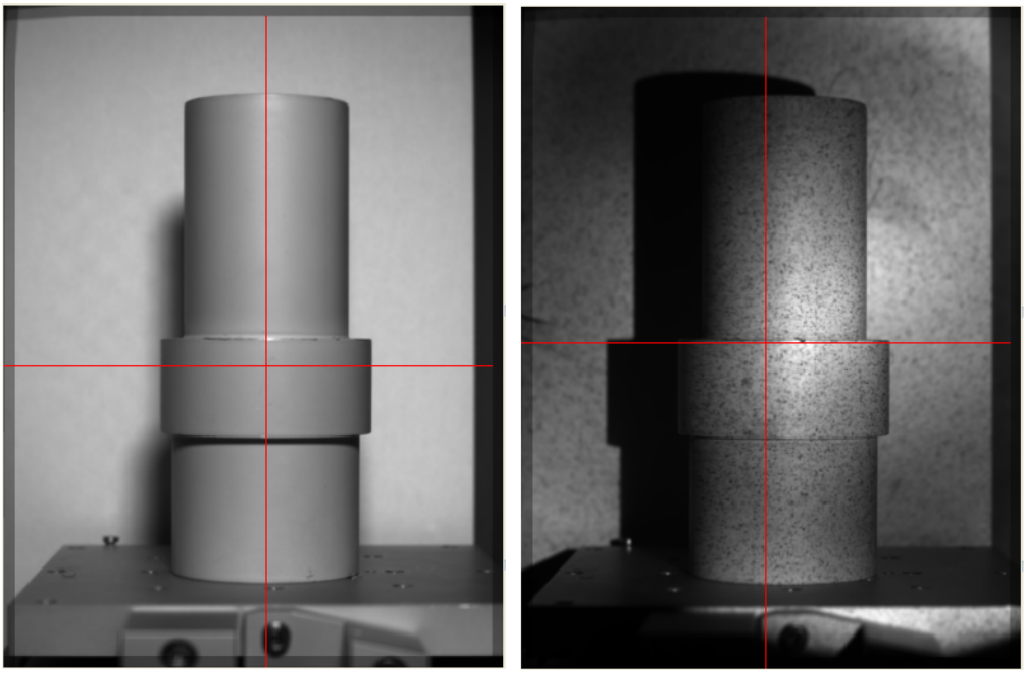


Abbildung 5.9: Zentralansicht aus dem Lichtfeld, das mit dem Gantry von einem Messkörper aufgenommen wurde. Links: mit homogener Beleuchtung, rechts: mit projiziertem Binärmuster

Betrachten wir nun die 2D-Lichtfeldschnitte entlang der rot eingezeichneten Linien, wie sie in Abbildung 5.10 dargestellt sind. Die Struktur in der Mitte ist eine Zeile des Messkörpers. In y-Richtung ist Φ_x, also die gleiche Zeile aus einer anderen Kameraansicht aufgetragen. Die leichte Neigung der rechten Kante des Zylinders nach rechts deutet darauf hin, dass die Kamera leicht hinter dieser Kante fokusiert war. Die schwarzen Keile an den Rändern der 2D-Schnitte sind Artefakte. Sie bedeuten nur, dass die entsprechenden Punkte aus dieser Kameraposition nicht mehr im Bild waren.

Es lässt sich erkennen, dass die Musterprojektion die hochfrequente Struktur wesentlich verbessert. Dies ist die Vorraussetzung für eine gute Tiefenbildrekonstruktion.

Abbildung 5.11 zeigt die aus den Lichtfeldern rekonstruierten Tiefenbilder. Wichtig ist hierbei die Rekonstruktion auf dem Objekt, also innerhalb des rot umrandeten Bereichs. Auf dem rechten Tiefenbild, das aus dem Lichtfeld mit Musterprojektion berechnet wurde, ist die Rekonstruktion vollständig und gut. Nur an der Stelle des Schattens im Originalbild ist nicht genug Textur für eine Tiefenbildrekonstruktion vorhanden.

Auf dem linken Tiefenbild dagegen, das aus dem lediglich homogen beleuchteten Messkörperlichtfeld berechnet wurde, ist das Objekt so schlecht rekonstruiert, dass es kaum zu

Abbildung 5.10: 2D-Lichtfeldschnitte der in Abbildung 5.9 in rot eingezeichneten Zeilen mit homogener Beleuchtung (oben) bzw. aufprojiziertem Binärmuster (unten)

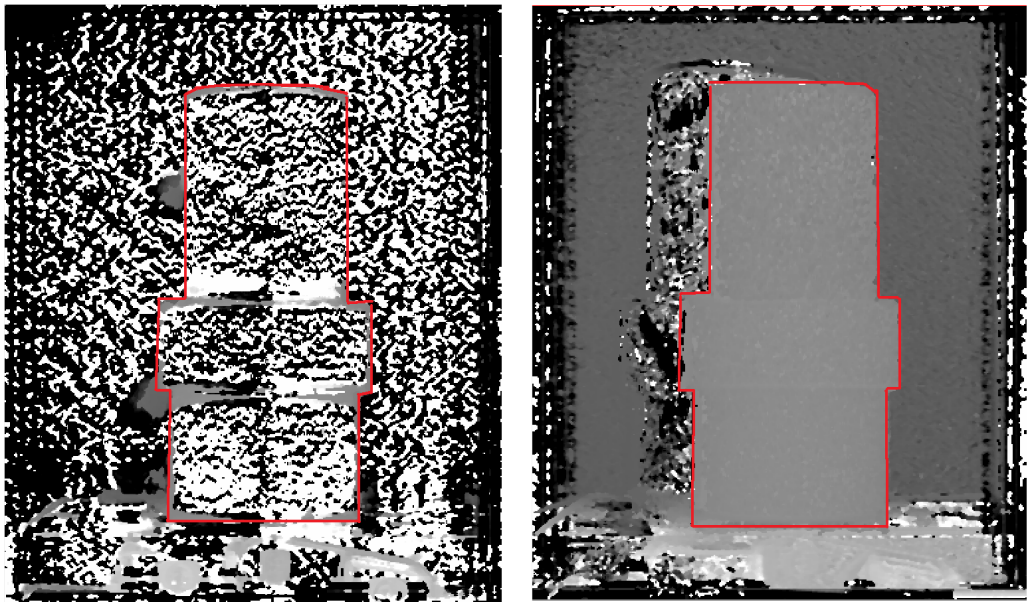


Abbildung 5.11: Tiefenbilder berechnet aus den Lichtfeldern aus Abbildung 5.9 des Messkörpers nach der Methode und mit der Implementierung von Wanner [SW12]. Links: aus dem homogen beleuchteten Lichtfeld, rechts: aus dem mit Projektor beleuchteten Lichtfeld. Zur Verdeutlichung wurde die Objektkontur per Hand im Bild eingezeichnet.

erkennen ist. Dies war bei einem nahezu texturlosen Objekt zu erwarten. Der Algorithmus findet keine hochfrequenze Struktur und kann deshalb auch aus dem Lichtfeld kein Tiefenbild berechnen, so wie es im analogen Fall einer Stereotriangulation der Fall wäre. Die Musterprojektion verbessert also tatsächlich wesentlich die Tiefenbildrekonstruktion mit dem Algorithmus von Wanner [SW12] aus Lichtfeldern.

Wenn man genau hinsieht, lässt sich in Abbildung 5.10 auch eine leichte Variation *entlang* der Helligkeit der schrägen, von unten nach oben verlaufenden Streifen entdecken. D.h. die Helligkeit desselben Objektpunktes variiert leicht von Ansicht zu Ansicht. Hierbei handelt es sich um die gesuchten *Glanzflecke*. Bei diesem kooperativen Objekt stören sie offensichtlich nicht bei der Abstandsermittlung, da die von ihnen aufgebrachte Struktur, bei dieser Objektoberflächentopografie verglichen mit der Textur sehr niederfrequent ist. Es würde nur zu Problemen führen, wenn sie eine ähnliche Frequenz wie die aufgebrachte Textur hätte. In diesem Falle könnte der Algorithmus die Glanzflecken für die eigentliche Struktur halten und den Abstand entsprechend falsch schätzen. Die einzige Möglichkeit eine solche Fehlschätzung zu erkennen wäre, dass die Tiefe schlicht nicht im Fokalbereich der Kamera wäre und damit kein richtiges Ergebnis sein könnte.

Als nächstes wollen wir das Lichtfeld eines Bauteils und als Beispiel hierfür Bauteil Nummer 6 (Zündkerze) betrachten. Die Abbildung 5.12 zeigt das Lichtfeld, wie man es im Orthoviewer (Software auf HCI-Homepage [SW11] erhältlich) anzeigen kann. Um die Glanzflecken deutlicher zu sehen, wurde eine Aufnahme mit homogener Beleuchtung gewählt.

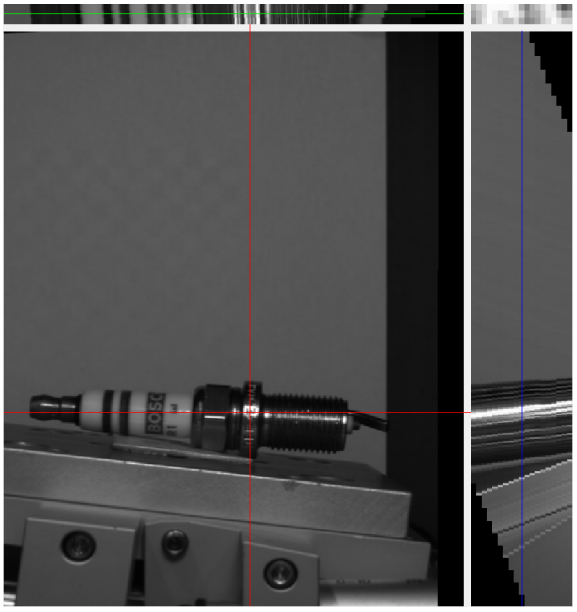


Abbildung 5.12: Ansicht aus dem Orthoviewer (Software auf der HCI-Hompage [SW11] erhältlich) des Lichtfeldes von Bauteil 6 (Zündkerze). Über bzw. rechts vom Bild sind die 2D-Lichtfeldschnitte entlang der im Bild eingezeichneten roten Linien dargestellt (oben: (x,Φ_x), rechts: $((y,\Phi_y))$.

Richten wir die Aufmerksamkeit auf den 2D-Lichtfeldschnitt am rechten Rand der Abbildung. Hier wird vertikal die im Bild rot markierte Bildspalte dargestellt. Horizontal sieht man die gleiche Bildzeile aus einem jeweils anderen Blickwinkel. Die schwarzen treppenförmigen Artefakte oben rechts und unten links bedeuten, dass der jeweilige Punkte in dieser Ansicht nicht mehr im Bildfeld war. Die horizontalen Linien sind die Trajektorien jeweils eines Punktes durch die verschiedenen Ansichten. Wenn man jetzt die untere der beiden sehr hellen Linien im Bild betrachtet, so sieht man, dass sie von links nach rechts immer heller wird. Genau das ist ein Glanzfleck. Es ist trotzdem deutlich zu erkennen, dass die

horizontale Struktur dominiert, d.h. es sind trotz Glanzfleck die Trajektorien der Punkte gut zu erkennen. Dies zeigt sich auch in der Tiefenbildrekonstruktion dieses Lichtfeldes (Abbildung 5.13).

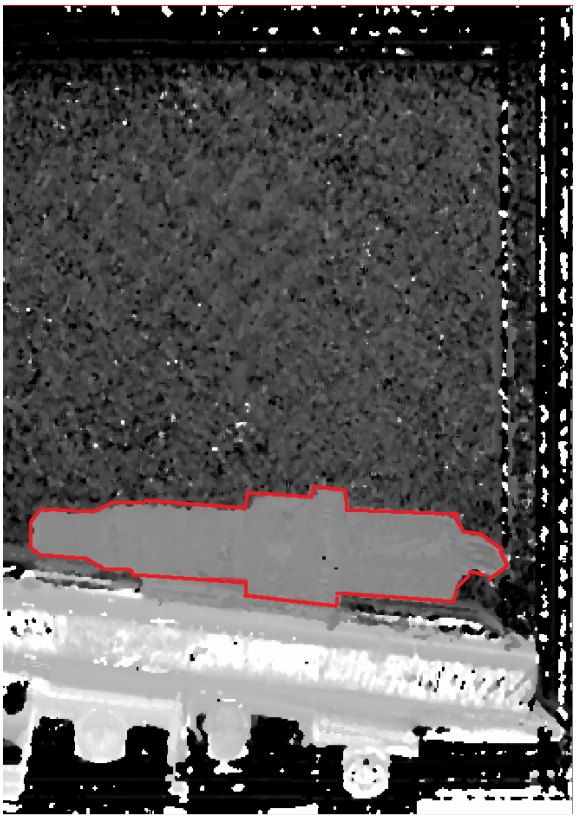


Abbildung 5.13: Tiefenbildrekonstruktion mittels des Algorithmus von Wanner [SW12] aus dem Lichtfeld des Bauteils Nummer 6 (Zündkerze)

Bei diesem Bauteil war genug Eigentextur vorhanden, so dass die Tiefe auch mit homogener Beleuchtung gut geschätzt werden konnte. Die Stereorekonstruktion und darauf folgende Objektlageerkennung in den in Kapitel 4 beschriebenen Messungen schlug wegen der stark glänzenden Oberfläche fehl (siehe auch Abbildung A.6 und zugehörige Messtabelle A.6 im Anhang). Damit ist auch das zweite Ziel nachgewiesen. Man kann die Tiefenbildrekonstruktion an Glanzflecken mit Hilfe von Lichtfeldern gegenüber der Stereorekonstruktion verbessern.

5.5 Bewertung und Ausblick

Als Erstes wollten wir untersuchen, ob die Tiefenbildrekonstruktion aus einem Lichtfeld durch eine Musterprojektion verbessert werden kann. Wir haben dies am Beispiel eines untexturierten, matten Objekts gesehen. Das Tiefenbild konnte ohne die Musterprojektion überhaupt nicht rekonstruiert werden, mit Musterprojektion war sie auf dem Objekt praktisch lückenlos.

Die zweite These war, dass es mittels Lichtfeldern möglich ist, Tiefenbilder an Glanzflecken zu rekonstruieren. Dies wurde an Hand eines Bauteils aus dem Bosch Bauteilespektrum (Zündkerze, BTN6) nachgewiesen, an dem unser Stereorekonstruktionsalgorithmus wegen Glanzflecken gescheitert ist. Wir haben gesehen, dass die Tiefenbildrekonstruktion auf dem Bauteil trotz Glanzflecken sehr gut war.

Was gibt es noch zu tun? Nachdem die Tiefenbildrekonstruktion an Glanzflecken mittels Lichtfeldern vielversprechend gut funktioniert, sollte diese Methode auf einem breiten

Spektrum evaluiert und eine nachfolgende Objektlageerkennung durchgeführt werden. Um Lichtfelder im industriellen Umfeld einsetzen zu können, müssen allerdings noch weitere praktische Herausforderungen gemeistert werden. Zunächst muss die Aufnahme beschleunigt werden. Eine Möglichkeit hierfür wäre die plenoptische Kamera. Mit ihr sind One-Shot-Aufnahmen möglich. Dabei müsste jedoch untersucht werden, ob die Lichtfeldqualität (insbesondere die sehr viel geringere Winkelauflösung) für die Tiefenbildrekonstruktion ausreicht. Des weiteren müssten alle Algorithmenteile beschleunigt werden, um den in der Industrie üblichen Taktzeitanforderungen gerecht zu werden. Ein so aufgebautes Gesamtsystem müsste dann wieder an den Genauigkeitsanforderungen der Industrie gemessen und evaluiert werden.

6. Schluss

6.1 Zusammenfassung

Im Rahmen der vorliegenden Arbeit wurde ein neues System zur Objektlageerkennung aus Tiefenbildern aus teilweise bekannten Einzelalgorithmen zusammengesetzt und komplett implementiert. Dieses umfasst die Bildaufnahme, Tiefenbildrekonstruktion aus Stereokameras mit Musterbeleuchtung, Segmentierung der Objektpunkte vom Hintergrund, Groblageerkennung mittels Hauptkomponentenanalyse (PCA) und Feinlageerkennung mittels Iterative Closest Point - Algorithmus (ICP).

Ein Ziel der Arbeit war, auch ein Modelltraining zu ermöglichen, mit dem alle für die Erkennung eines Objektes notwendigen Daten *automatisch* und innerhalb *kurzer Zeit* (weniger als 15 Minuten) trainiert werden können. Dies wurde realisiert und beinhaltet die Projektorkalibrierung, das Training von Beleuchtungsparametern und der Modelle für die Segmentierung, sowie Grob- und Feinlageerkennung.

Das implementierte Gesamtsystem wurde auf einem für die industrielle Produktion (am Beispiel der Robert Bosch GmbH) *repräsentativen Bauteilespektrum* evaluiert und es wurde eine Abdeckung von 42,5% der Bauteile erreicht, so dass diese im üblichen Anwendungsfall mit ausreichender Genauigkeit von einem Roboter gegriffen werden könnten. Die Taktzeitvorgabe von 0,5 Sekunden pro durchschnittlichem Objektlageerkennungsvorgang wurde hierbei ohne Hardwarebeschleunigung eingehalten.

Es wurde eine umfangreiche Fehlerursachenanalyse durchgeführt und gezeigt, dass

1. der Hauptgrund für eine Fehlerkennung zu viele Fehler in der Tiefenbildrekonstruktion sind
2. Glanzpunkte der Hauptgrund für eine fehlerhafte Tiefenbildrekonstruktion sind (42% der Fälle), gefolgt von zu feinen Bauteilstrukturen (29%) und für eine Musterprojektion ungeeigneten Objektoberflächen (13%).

In einem letzten Kapitel wurde gezeigt, dass *Lichtfelder* eine Lösung für die Hauptfehlerursache bieten können, da es auf ihnen basierende Methoden zur Tiefenbildrekonstruktion gibt, die robuster gegenüber Glanzflecken sind. Die Tiefenbildrekonstruktion konnte durch Übertragung der Methode der Musterprojektion aus der Stereotiefenbildrekonstruktion wesentlich verbessert werden.

6.2 Ausblick

Das im Rahmen der vorliegenden Arbeit vorgestellte System zur Objektlageerkennung aus Tiefenbildern kann an einigen Stellen verbessert werden.

Es hat sich in der Evaluation herausgestellt, dass die Tiefenbildrekonstruktion maßgeblich für die Güte der Lageschätzung ist. Nach Glanzpunkten war die *zweithäufigste* Fehlerursache, dass Bauteilstrukturen zu fein waren, um eine Tiefenbildrekonstruktion aus jeder Lage gewährleisten zu können. Ein vielversprechender Ansatz wäre also die Auflösung des Sensorsystems zu verbessern. Hierfür könnte z.B. eine Kamera mit höherer Auflösung, sowie eine Verbesserung (Verfeinerung) des Musters verwendet werden.

Im vorgestellten Algorithmus wurde nur *eine* Ansicht für das Training verwendet. Um die tolerierbare Verkippung der zu erkennenden Objekte zu vergrößern, könnten mehrere Ansichten trainiert werden und das Matching jeweils mit der am besten passenden Ansicht durchgeführt werden. Dies würde die Laufzeit für das Shape Based Matching zwar wesentlich erhöhen (damit könnte voraussichtlich die passende Ansicht ermittelt werden), den Einzugsbereich aber auf beliebige Objektlagen erweitern. Damit wäre dann auch ein „Griff in die Kiste“ denkbar.

In der jetztigen Implementierung müssen Objektsymmetrien noch per Hand angegeben werden. Durch eine geeignete Auswertung der Eigenwerte der Hauptachsenanalyse (Groblageerkennung) könnten diese Symmetrien automatisch bestimmt werden, was zu einer weiteren Automatisierung des Modelltrainings führen würde.

Einige Komponenten sind noch in der Halcon-Skriptsprache „HDevelop“ geschrieben und werden mit komplizierten Datenübergaben angesteuert (vor allem die Segmentierung mittels Shape Based Matching). Eine Implementierung dieser Programmteile in C++ könnte den Gesamtalgorithmus beschleunigen, da die Datentransformationen wegfallen würden. Ferner könnten Teile oder der gesamte Algorithmus in Hardware auf einer Smartkamera implementiert werden, womit die Objektlageerkennung echtzeitfähig werden könnte.

Ferner haben wir gesehen, dass Lichtfelder einen Ausweg für die Glanzfleckproblematik bieten können. Zunächst sollte dies auf breiterer Basis (ähnlich wie hier mit der Stereobilderkennung gezeigt) evaluiert werden. Zu erwarten wäre, dass Lichtfelder die *Bauteilabdeckung* einer Objektlageerkennung wesentlich verbessern.

Die Tiefenbildberechnung aus Lichtfeldern ist Gegenstand aktueller Forschung (u. A. am HCI Heidelberg). Die Entwicklung einer schnellen und robusten Tiefenbildberechnung ist Grundlage für den Einsatz von Lichtfeldern für die Objektlageerkennung aus Tiefenbildern. Die Tiefenbildberechnung aus Lichtfeldern, die von einer plenoptischen Kamera aufgenommen wurden, sollten evaluiert und verbessert werden. Dies würde eine kostengünstige Alternative für Stereosysteme darstellen, die ebenso echtzeitfähig wäre und die Güte der Objektlageerkennung wesentlich verbessern könnte.

Die Nutzung von Lichtfeldern würde nicht nur das Bauteilespektrum erweitern, sondern auch *weitere Optionen* der industriellen Nutzung eröffnen. Beispielsweise kann aus einem Lichtfeld ein überall scharfes Bild berechnet werden, oder *Bauteildefekte*, die bisher noch mit komplizierter Beleuchtung sichtbar gemacht werden müssen, direkt im Lichtfeld gefunden werden. Dies könnte als Zusatzoption bei der Objektlageerkennung mitimplementiert werden.

Die Objektlageerkennung wird immer intelligenter und es können heute schon viele einfache Aufgaben mit ihrer Hilfe automatisiert werden. Dieser Trend wird sich fortsetzen. Lichtfelder bieten viele neue spannende Möglichkeiten und werden in Zukunft im industriellen Umfeld zunehmende Anwendung finden.

A. Anhang A

A.1 Beweis: Octree kann beliebig tief werden

Zu beweisen ist, dass ein Octree bei „schlechten“ Daten beliebig tief werden kann. Sei also $n \in \mathbb{N}$ eine natürliche Zahl. Zu zeigen ist, dass es eine Datenmenge gibt, so dass deren Octree die Tiefe n hat. Sei der Einheitswürfel $W \subset \mathbb{R}^3$ unser Suchraum, dann ist $M = \left\{p \in \mathbb{R}^3 | p_1 = 1/k, p_2 = p_3 = 0\right\} \subset M$ mit $k \in 1, \ldots, n$ eine solche Datenmenge. *q.e.d.*

A.2 Bilder und Messergebnisse der Bauteile

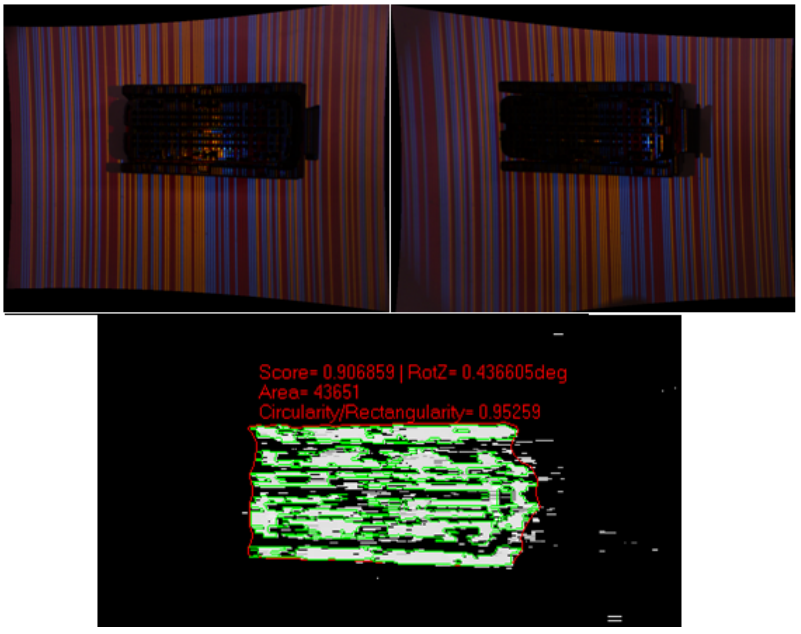


Abbildung A.1: o.: Beispielbildpaar des Bauteils Nummer 1 (Stecker) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert, Abb. aus Halcon

Tabelle A.1: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 1 (Stecker) in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,865	0,987	0,060	0,218	0,593	2,779
Versuch A: Rng	4,727	5,581	0,289	1,074	2,902	13,722
Versuch B: σ	2,441	2,518	1,039	14,306	19,025	20,381
Versuch B: Rng	17,040	14,343	11,307	119,723	160,389	272,228
Versuch C: σ	1,450	4,823	0,673	41,777	12,842	68,123
Versuch C: Rng	8,979	19,233	5,860	246,300	102,041	355,465

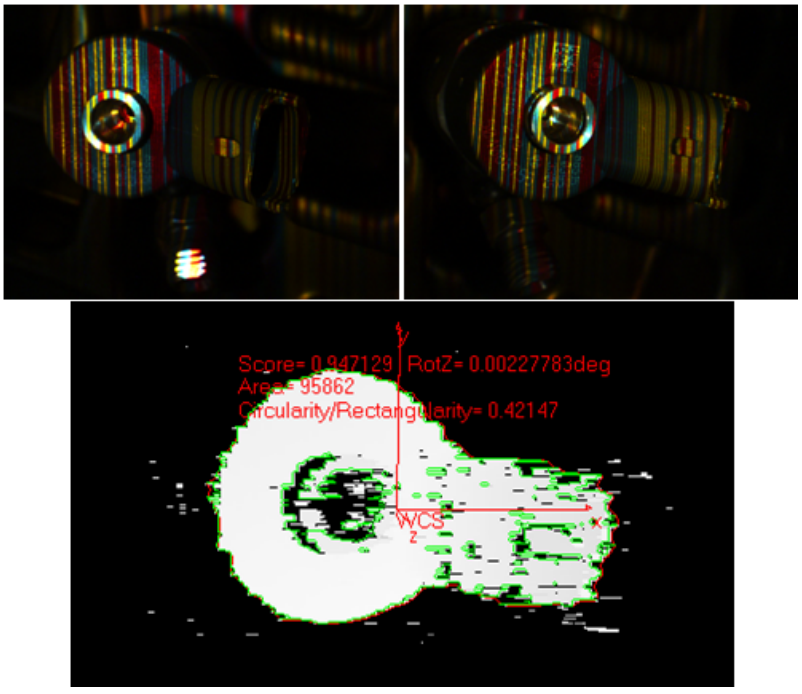


Abbildung A.2: o.: Beispielbildpaar des Bauteils Nummer 2 (Injektor) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.2: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 2 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,030	0,040	0,013	0,120	0,063	0,114
Versuch A: Rng	0,125	0,179	0,049	0,441	0,226	0,423
Versuch B: σ	0,250	0,225	0,401	0,261	0,282	0,392
Versuch B: Rng	1,448	1,005	1,599	1,289	1,450	2,615
Versuch C: σ	0,079	0,111	0,348	0,244	0,117	0,380
Versuch C: Rng	0,219	0,324	1,091	0,828	0,369	1,388

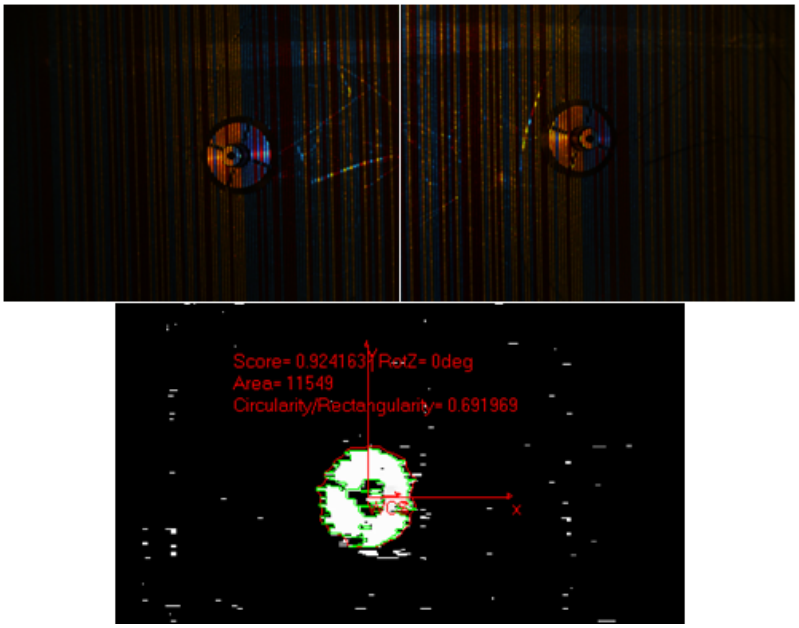


Abbildung A.3: o.: Beispielbildpaar des Bauteils Nummer 3 (Anker) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.3: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 3 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,268	0,330	0,025	0,117	0,131	0,003
Versuch A: Rng	1,461	1,356	0,095	0,524	0,651	0,014
Versuch B: σ	0,640	0,619	0,294	0,376	0,253	0,133
Versuch B: Rng	4,547	4,210	1,077	2,256	2,301	0,532

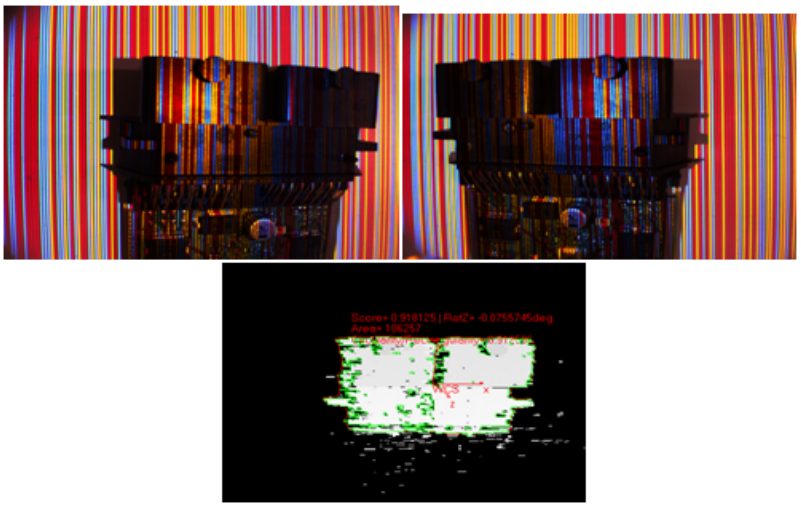


Abbildung A.4: o.: Beispielbildpaar des Bauteils Nummer 4 (Steuergerät) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.4: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 4 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,108	0,108	0,043	0,079	0,196	0,262
Versuch A: Rng	0,390	0,431	0,210	0,275	0,900	1,007
Versuch B: σ	0,515	0,826	0,082	0,169	0,295	1,796
Versuch B: Rng	3,044	4,899	0,519	0,869	1,826	10,443
Versuch C: σ	0,497	1,235	0,154	0,130	0,254	1,314
Versuch C: Rng	2,507	5,525	0,511	0,541	1,261	5,652

Tabelle A.5: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 5 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,383	1,413	0,104	0,557	0,970	1,024
Versuch A: Rng	1,450	6,275	0,431	2,155	3,928	4,364
Versuch B: σ	1,341	3,154	0,137	0,795	3,611	1,974
Versuch B: Rng	6,297	13,874	0,755	4,207	16,766	12,477
Versuch C: σ	0,618	2,281	0,096	0,503	2,121	1,819
Versuch C: Rng	2,816	8,857	0,370	2,202	8,599	6,724

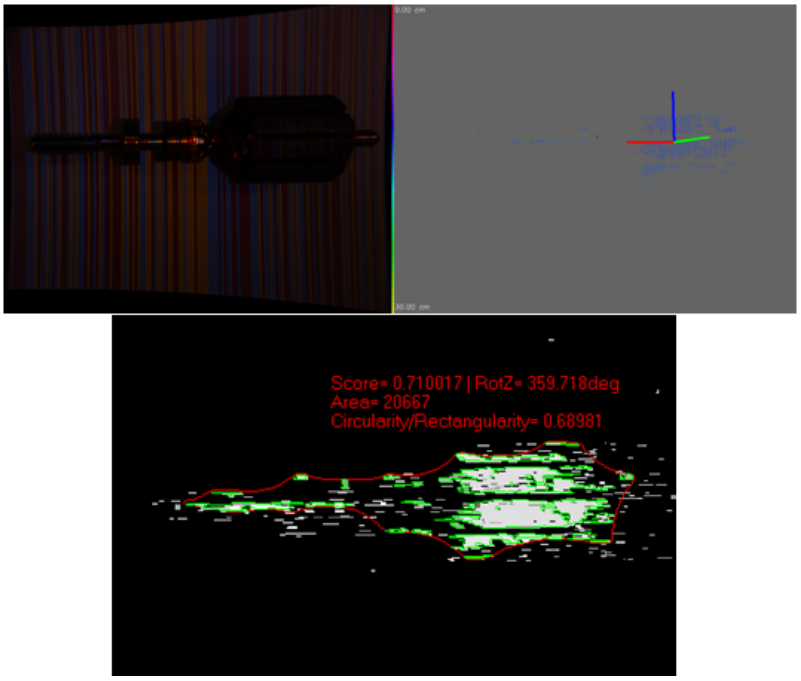


Abbildung A.5: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 5 (Ankerrohling) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

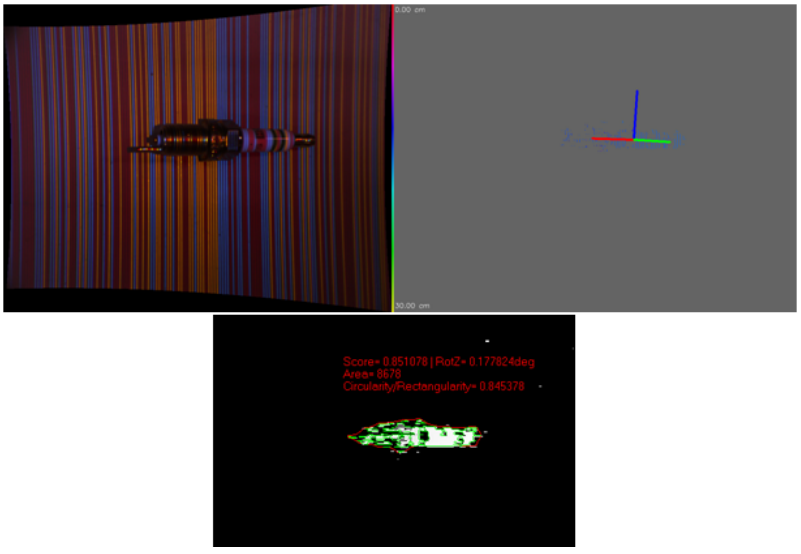


Abbildung A.6: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 6 (Zündkerze) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.6: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 6 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,253	0,658	0,136	0,398	2,345	1,727
Versuch A: Rng	0,992	2,788	0,611	1,616	9,313	7,427
Versuch B: σ	0,641	1,979	0,445	1,888	5,740	2,738
Versuch B: Rng	3,694	10,020	2,882	17,106	33,504	14,218
Versuch C: σ	0,662	1,612	0,417	11,745	8,458	45,772
Versuch C: Rng	2,307	4,701	1,637	45,376	27,700	176,608

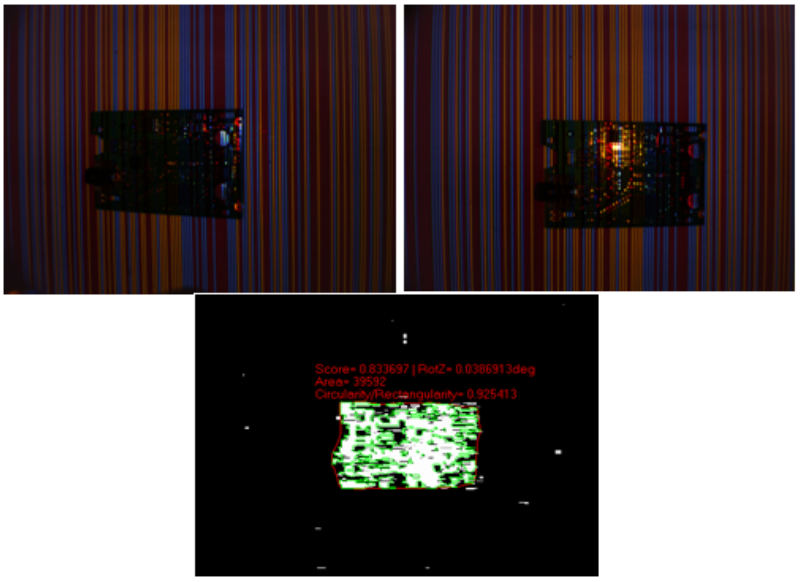


Abbildung A.7: o.: Beispielbildpaar des Bauteils Nummer 7 (Leiterplatte) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.7: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 7 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,297	0,764	0,058	0,232	0,425	71,907
Versuch A: Rng	1,028	2,449	0,210	0,842	2,087	183,158
Versuch B: σ	2,256	1,354	0,162	0,599	0,774	72,681
Versuch B: Rng	9,436	6,231	0,755	3,958	5,174	358,171
Versuch C: σ	1,551	1,672	0,186	1,005	1,102	56,925
Versuch C: Rng	6,164	7,538	0,870	4,776	4,643	354,791

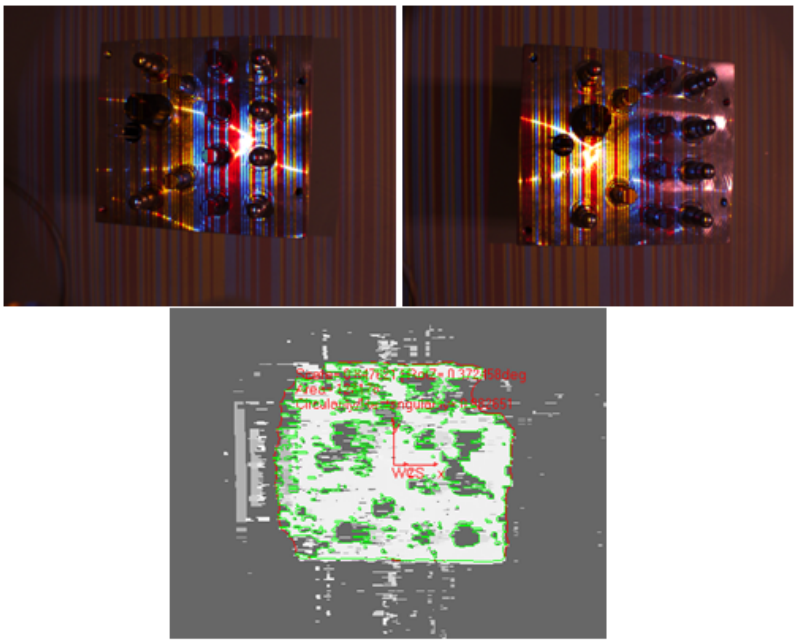


Abbildung A.8: o.: Beispielbildpaar des Bauteils Nummer 8 (Aggregat) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.8: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 8 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,184	0,088	0,028	0,080	0,061	0,779
Versuch A: Rng	1,003	0,317	0,114	0,313	0,231	4,397
Versuch B: σ	6,242	2,973	1,674	1,156	1,285	77,503
Versuch B: Rng	29,691	14,528	10,160	6,642	7,711	359,243
Versuch C: σ	2,110	1,496	0,134	0,348	0,428	89,975
Versuch C: Rng	10,144	8,912	0,529	1,829	1,881	353,677

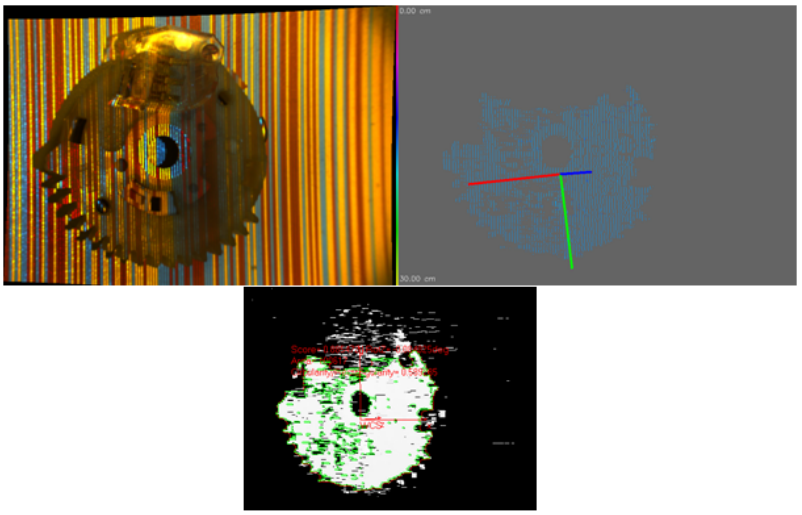


Abbildung A.9: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 10 (Koppeleinheit) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.9: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 10 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,695	0,417	0,036	0,236	0,216	4,614
Versuch A: Rng	2,046	1,327	0,147	0,946	1,046	20,052
Versuch B: σ	0,944	0,995	0,099	0,532	0,584	58,090
Versuch B: Rng	6,369	5,088	0,565	3,105	3,590	356,925

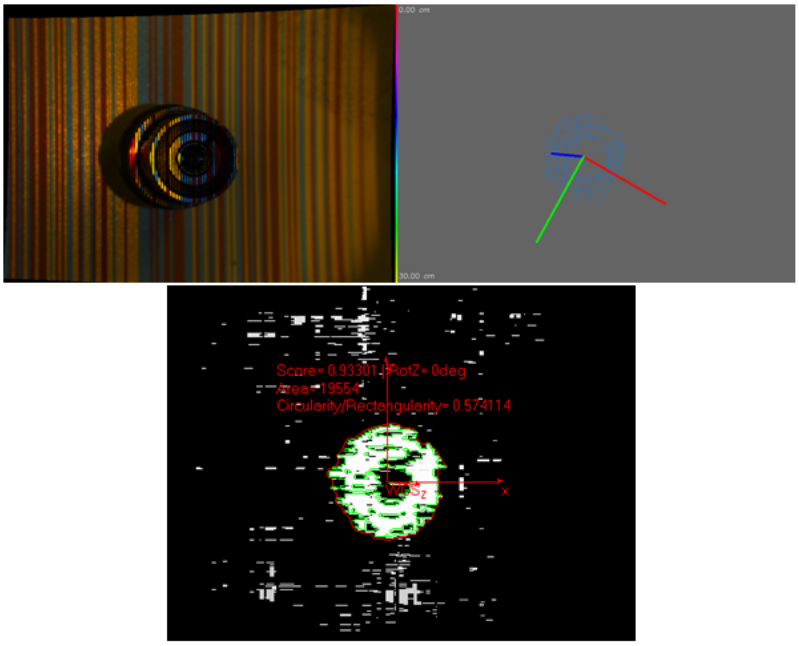


Abbildung A.10: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 12 (Magnetventil) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.10: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 12 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,125	0,657	0,012	0,175	0,126	
Versuch A: Rng	0,425	1,805	0,057	0,681	0,551	
Versuch B: σ	0,538	0,623	0,368	0,259	0,393	
Versuch B: Rng	2,859	2,778	1,329	1,281	1,981	

Tabelle A.11: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 13 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,170	0,219	0,039	0,625	0,871	
Versuch A: Rng	0,675	0,878	0,165	2,392	3,590	
Versuch B: σ	0,452	0,323	0,279	0,614	0,768	
Versuch B: Rng	2,105	1,525	1,087	3,050	4,515	

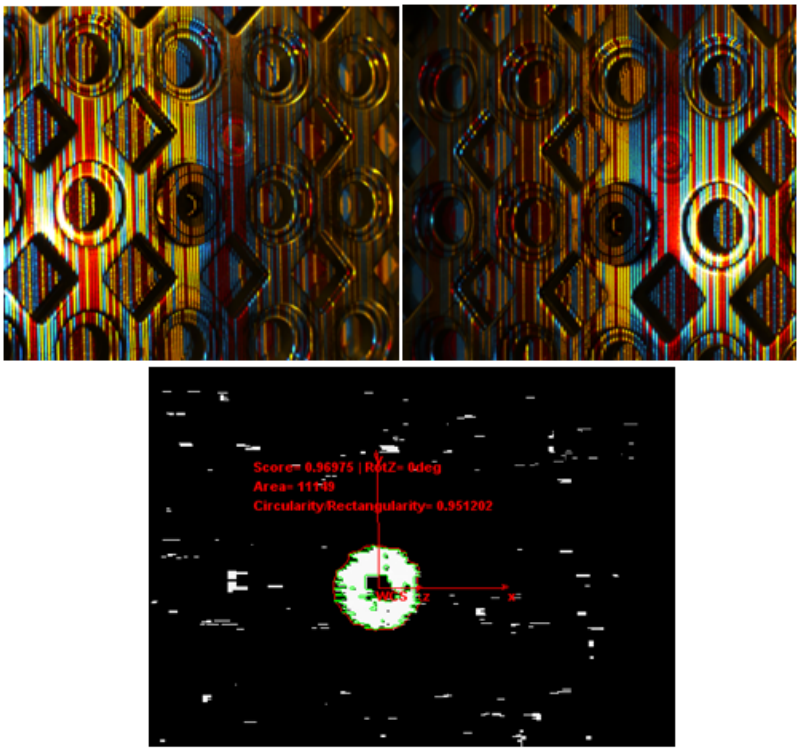


Abbildung A.11: o.: Beispielbildpaar des Bauteils Nummer 13 (Ventilplatte) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

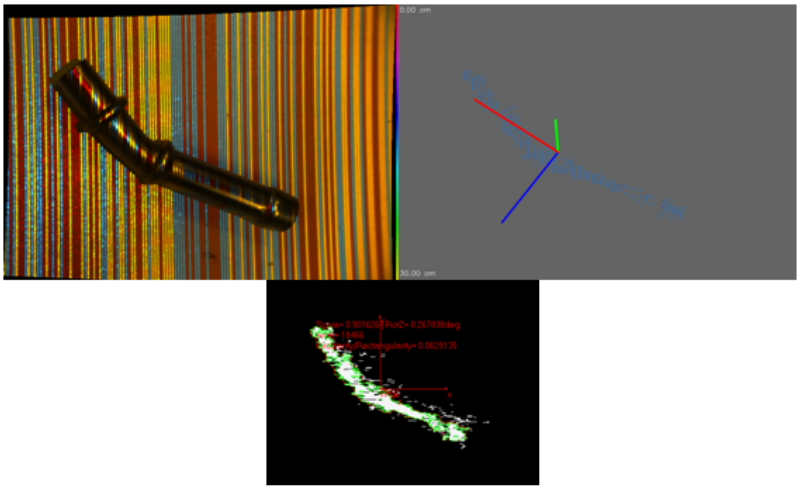


Abbildung A.12: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 14 (Rohrwinkel) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

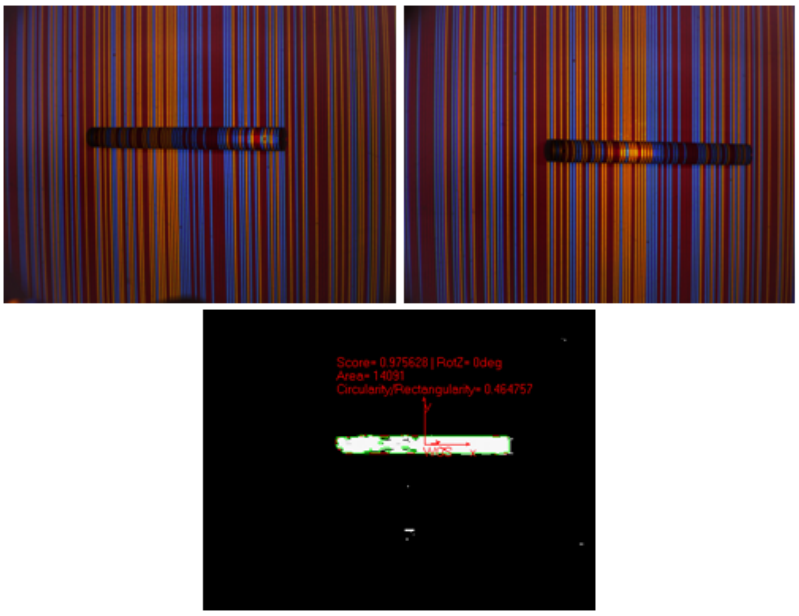


Abbildung A.13: o.: Beispielbildpaar des Bauteils Nummer 16 (Magnetkern Zündspule) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.12: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 16 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,029	0,205	0,014	0,013	0,636	0,027
Versuch A: Rng	0,126	0,949	0,057	0,042	2,316	0,069
Versuch B: σ	0,199	0,953	0,041	0,054	2,510	0,095
Versuch B: Rng	0,895	5,879	0,242	0,410	11,865	0,677

Tabelle A.13: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 17 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,083	0,054	0,002	0,125	0,086	0,034
Versuch A: Rng	0,329	0,197	0,009	0,467	0,323	0,120
Versuch B: σ	0,572	0,288	0,278	0,491	0,338	0,100
Versuch B: Rng	2,510	1,384	0,995	3,049	2,020	0,611
Versuch C: σ	0,769	0,339	0,325	1,421	0,804	0,252
Versuch C: Rng	2,579	0,962	0,954	5,092	2,997	0,999

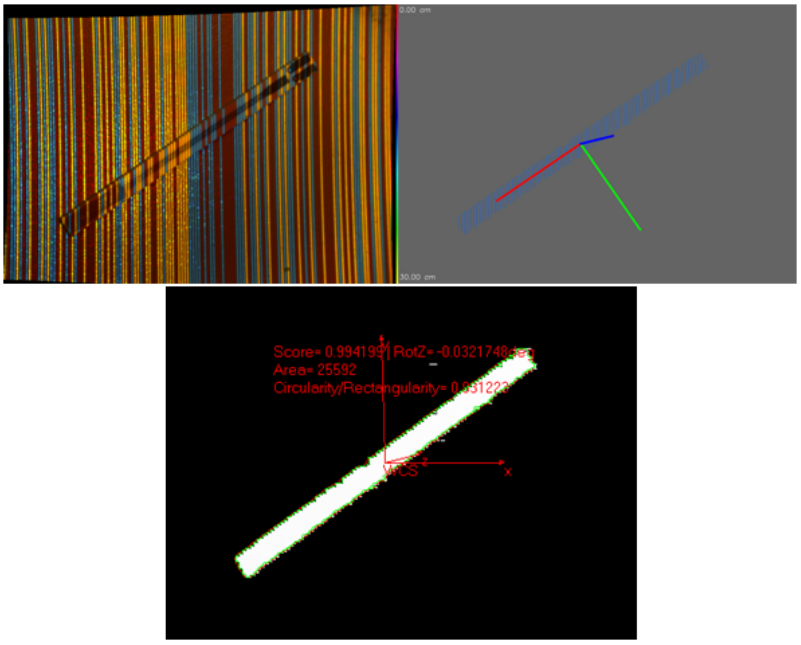


Abbildung A.14: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 17 (Sensorelement) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

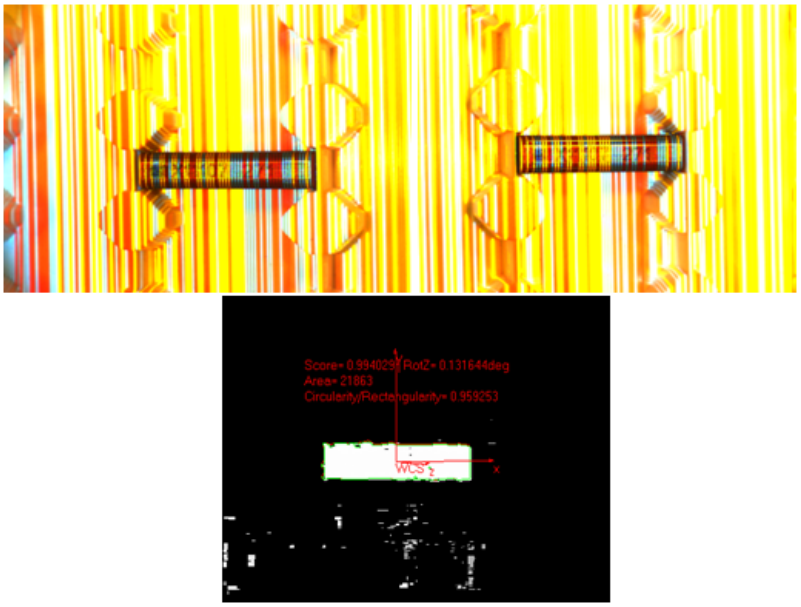


Abbildung A.15: o.: Beispielbildpaar des Bauteils Nummer 18 (Wandler) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.14: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 18 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,074	0,017	0,011	0,269	0,049	0,080
Versuch A: Rng	0,298	0,071	0,042	1,316	0,185	0,307
Versuch B: σ	0,826	0,381	0,274	1,167	0,981	0,695
Versuch B: Rng	5,121	1,658	1,126	7,506	3,726	3,827
Versuch C: σ	1,842	0,416	0,694	1,863	0,858	1,112
Versuch C: Rng	7,396	1,389	2,112	8,403	3,096	5,788

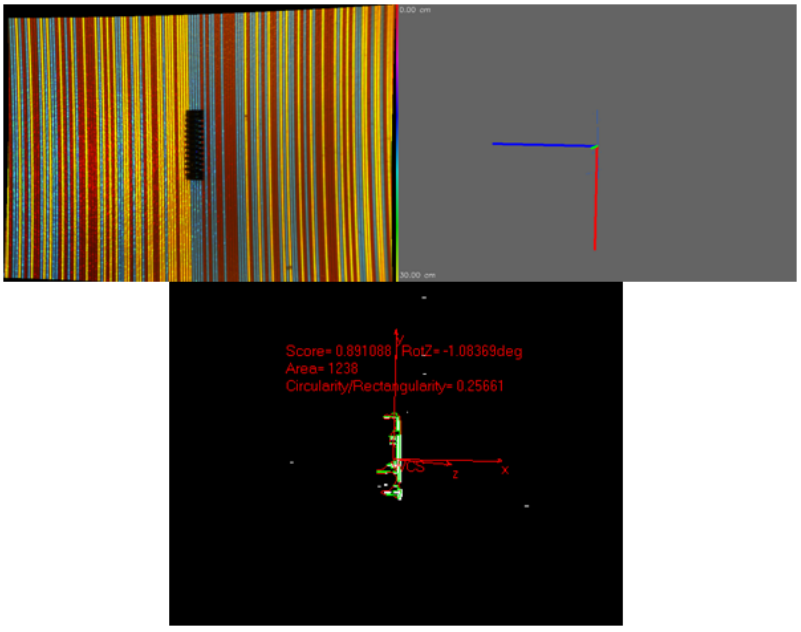


Abbildung A.16: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 19 (Feder) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

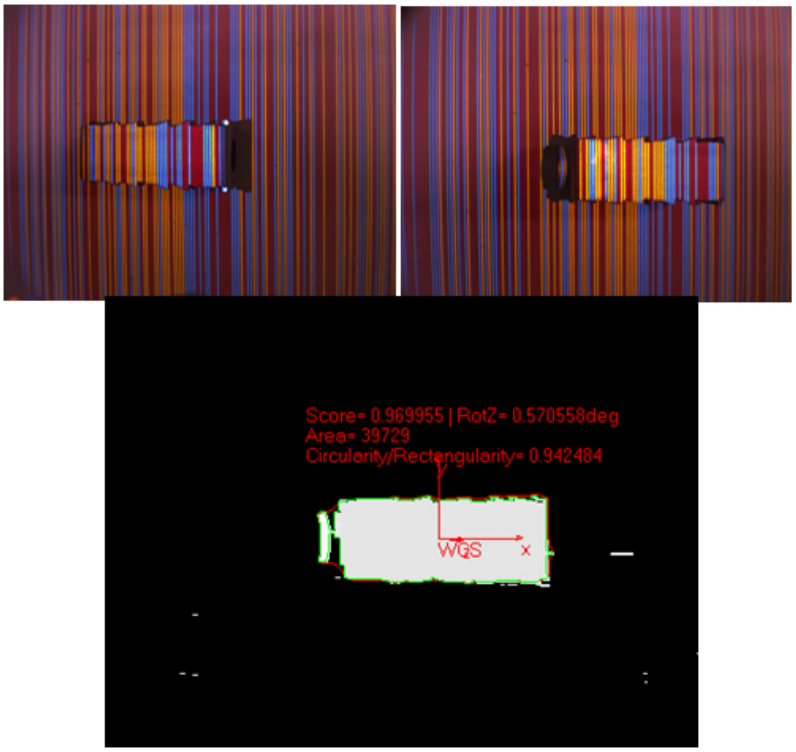


Abbildung A.17: o.: Beispielbildpaar des Bauteils Nummer 20 (Gehäuse) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.15: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 20 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,034	0,039	0,009	0,027	0,060	0,198
Versuch A: Rng	0,129	0,210	0,038	0,112	0,236	0,953
Versuch B: σ	0,194	0,566	0,062	0,216	0,264	0,640
Versuch B: Rng	1,288	3,117	0,337	1,127	1,307	3,949
Versuch C: σ	0,346	0,513	0,117	0,265	0,419	0,939
Versuch C: Rng	1,677	2,876	0,492	1,135	2,079	4,163

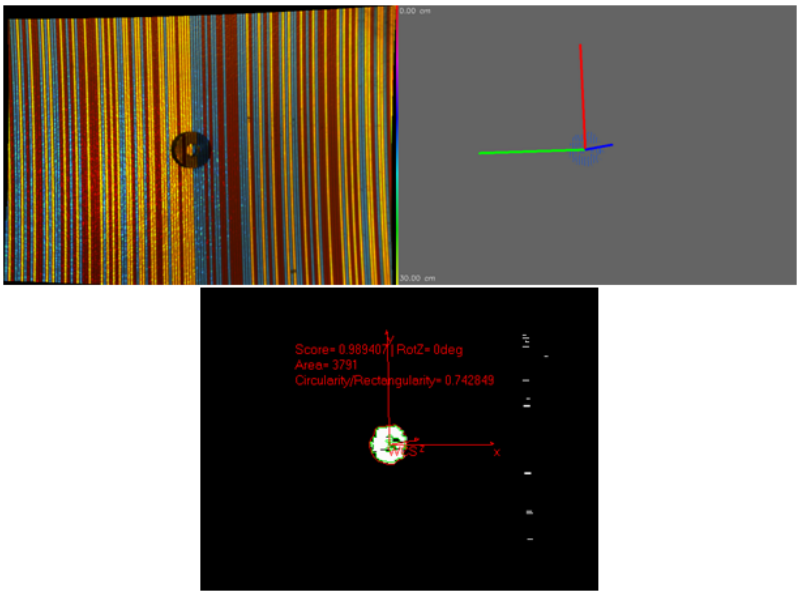


Abbildung A.18: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 21 (Federteller) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.16: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 21 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,079	0,064	0,019	0,616	1,063	0,058
Versuch A: Rng	0,385	0,255	0,081	2,110	5,114	0,278
Versuch B: σ	0,454	0,265	0,186	1,024	3,244	0,396
Versuch B: Rng	2,000	2,028	1,055	6,414	18,128	2,268

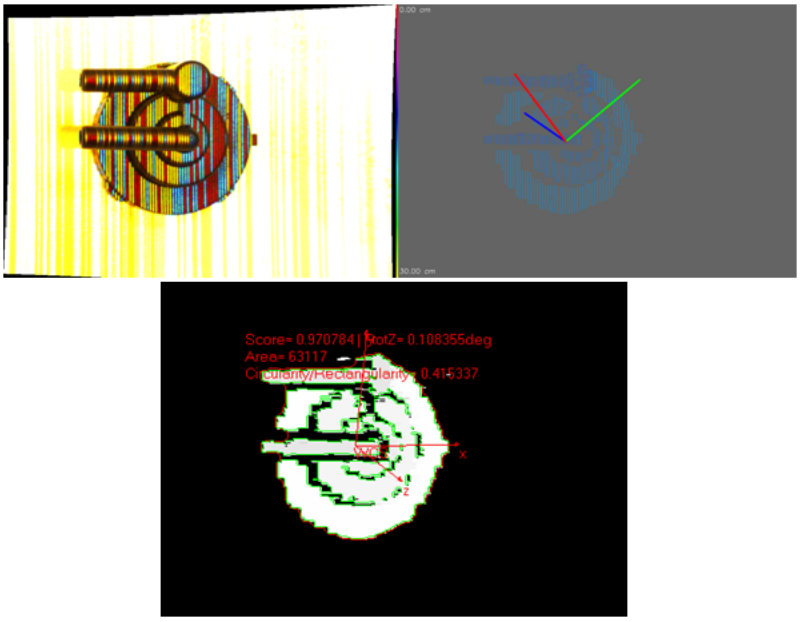


Abbildung A.19: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 22 (Gehäusedeckel) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.17: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 22 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,063	0,061	0,013	0,083	0,070	0,391
Versuch A: Rng	0,260	0,230	0,044	0,366	0,294	1,695
Versuch B: σ	0,727	0,481	0,403	1,607	1,063	14,692
Versuch B: Rng	8,188	3,617	1,418	21,074	13,697	145,504
Versuch C: σ	1,475	1,358	1,130	14,715	14,196	81,885
Versuch C: Rng	5,487	5,290	5,507	45,563	81,168	352,060

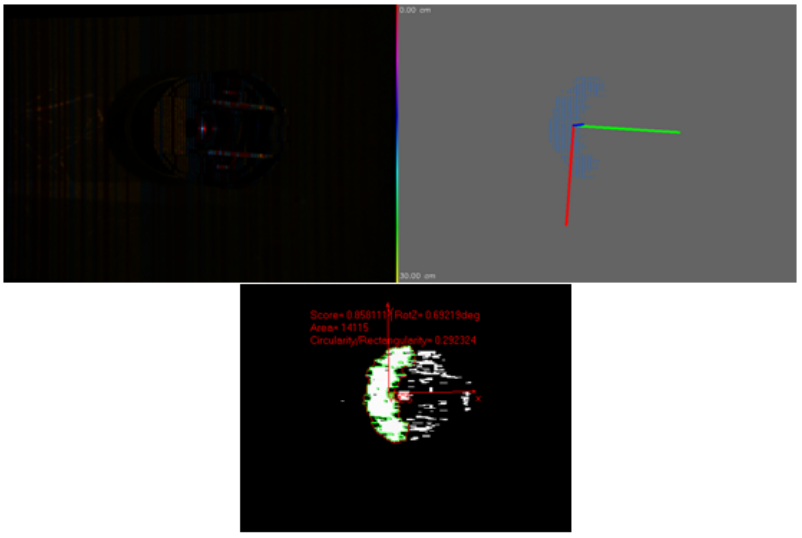


Abbildung A.20: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 23 (Spulenkörper) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

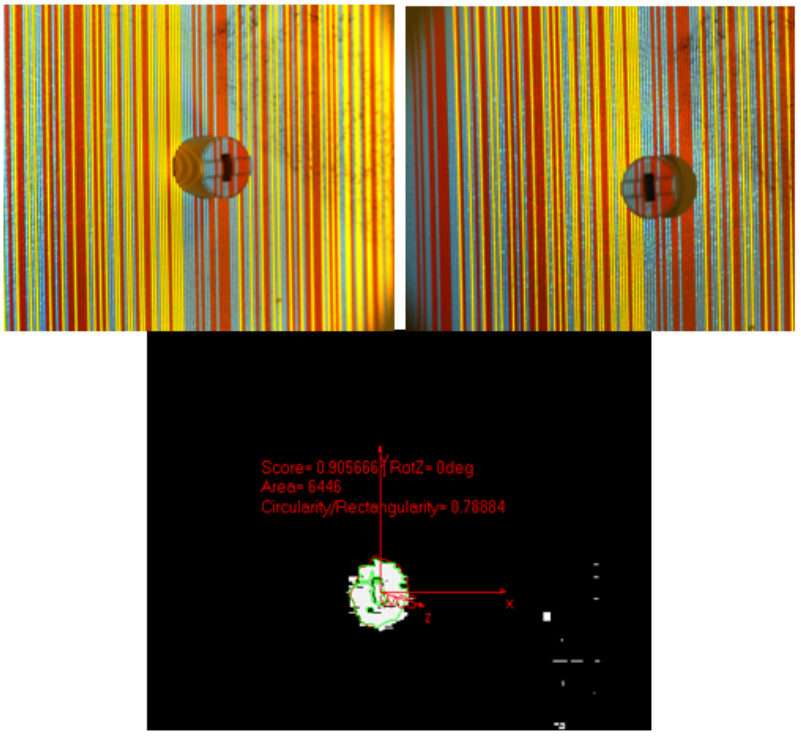


Abbildung A.21: o.: Beispielbildpaar des Bauteils Nummer 25 (Keramikbauteil) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

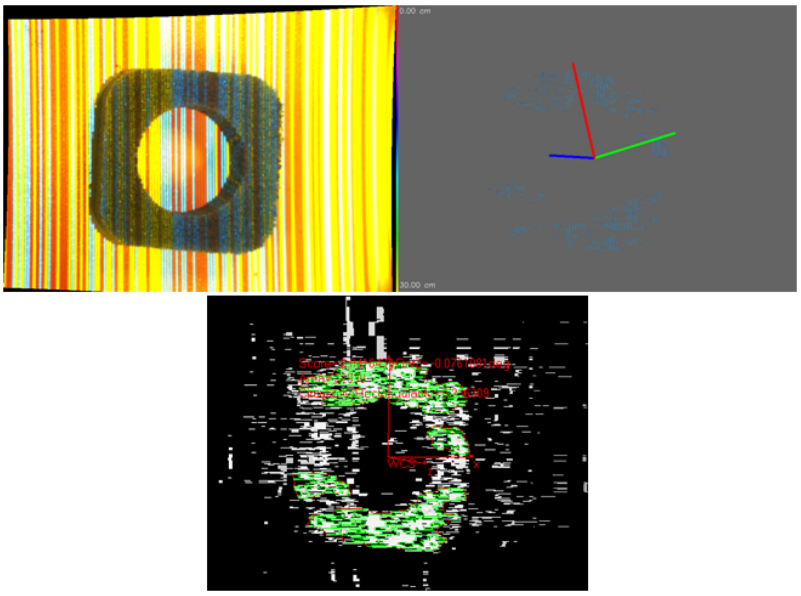


Abbildung A.22: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 26 (Dämpfer) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

Tabelle A.18: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 27 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,076	0,108	0,031	0,107	0,087	0,163
Versuch A: Rng	0,336	0,510	0,113	0,356	0,342	0,534
Versuch B: σ	0,305	0,423	0,092	0,273	0,306	1,721
Versuch B: Rng	1,824	2,965	0,408	1,331	1,649	14,729
Versuch C: σ	0,805	0,948	0,136	0,308	0,426	1,996
Versuch C: Rng	3,471	4,685	0,614	1,439	2,045	10,922

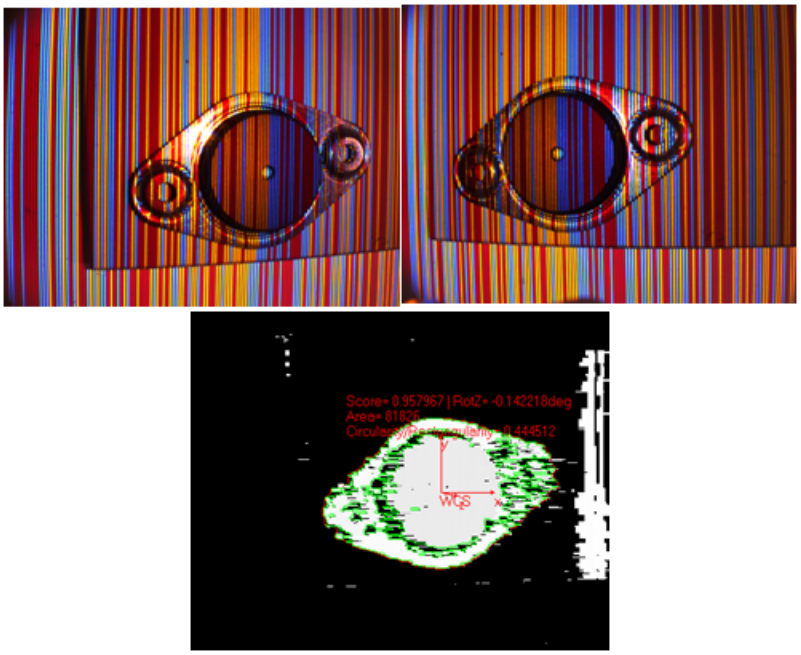


Abbildung A.23: o.: Beispielbildpaar des Bauteils Nummer 27 (Deckel) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

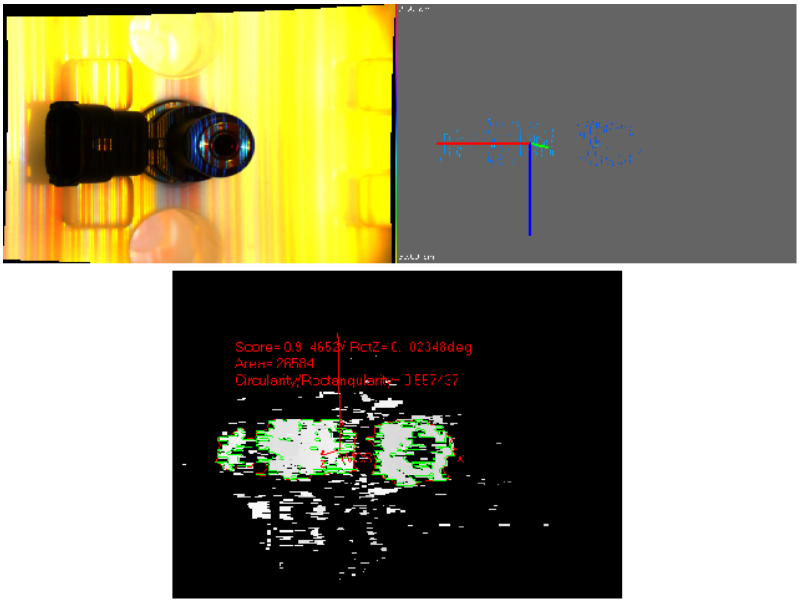


Abbildung A.24: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 29 (Ventil) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon - Das Modelltraining war so schlecht, dass keine Statistik erstellt wurde

Tabelle A.19: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 29 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,128	0,156	0,061	1,068	0,281	1,014
Versuch A: Rng	0,714	0,552	0,347	4,043	1,680	4,262
Versuch B: σ	2,270	1,275	2,051	26,340	21,351	76,462
Versuch B: Rng	20,372	9,916	16,469	349,702	142,882	358,318

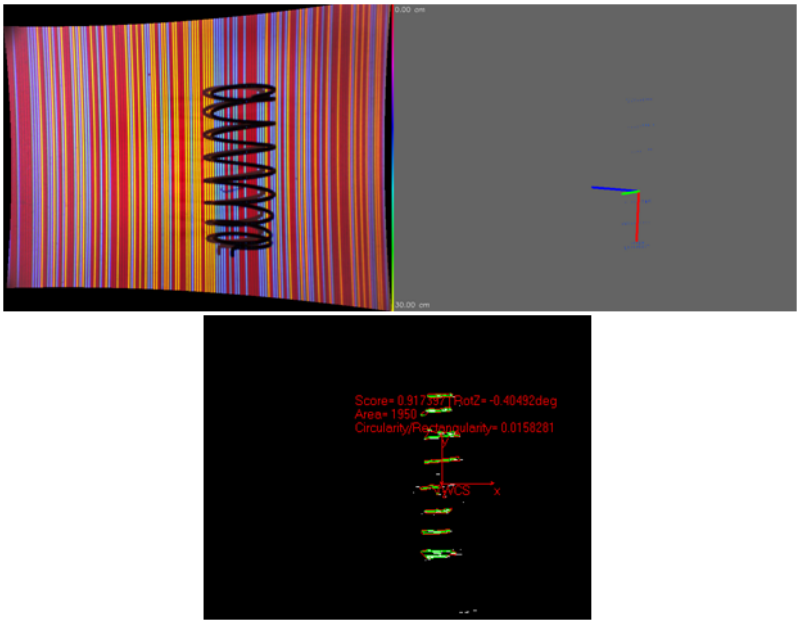


Abbildung A.25: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 30 (Drehfeder) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.20: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 30 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	3,118	0,351	0,409	43,826	2,074	89,303
Versuch A: Rng	13,316	1,294	1,560	158,738	6,711	180,520
Versuch B: σ	4,598	0,954	0,652	48,854	2,402	91,294
Versuch B: Rng	26,752	4,199	3,900	177,567	10,955	358,733

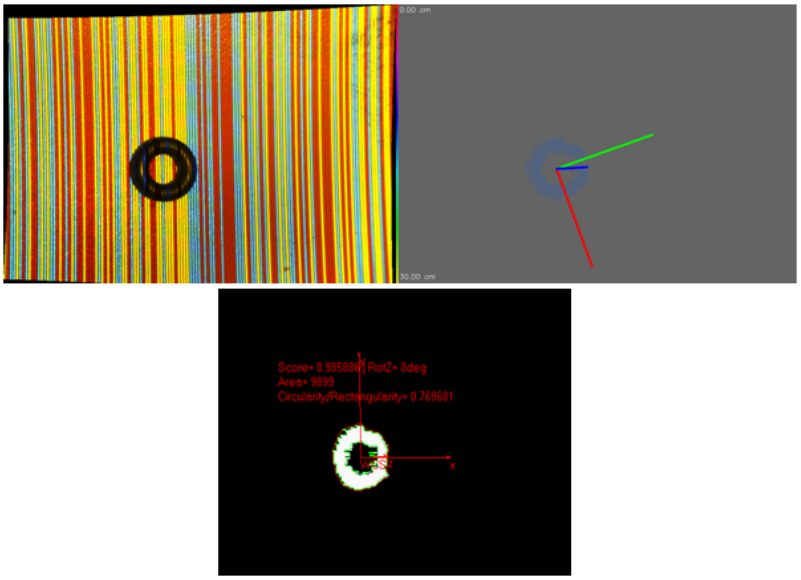


Abbildung A.26: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 31 (O-Ring) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.21: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 31 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,177	0,122	0,019	0,434	0,460	
Versuch A: Rng	0,628	0,471	0,068	1,915	2,401	
Versuch B: σ	0,572	0,355	0,354	0,722	1,242	
Versuch B: Rng	2,506	1,611	1,322	5,804	8,809	

Tabelle A.22: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 33 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,169	0,273	0,044	0,090	0,361	0,252
Versuch A: Rng	0,731	1,323	0,192	0,375	1,470	1,118
Versuch B: σ	0,417	3,626	0,146	0,171	0,829	1,200
Versuch B: Rng	2,715	14,763	0,857	0,949	5,528	5,044
Versuch C: σ	0,469	5,008	0,172	0,266	1,304	1,990
Versuch C: Rng	2,706	24,991	0,757	1,465	6,843	10,976

Abbildung A.27: o.: Beispielbildpaar des Bauteils Nummer 33 (Wischer) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

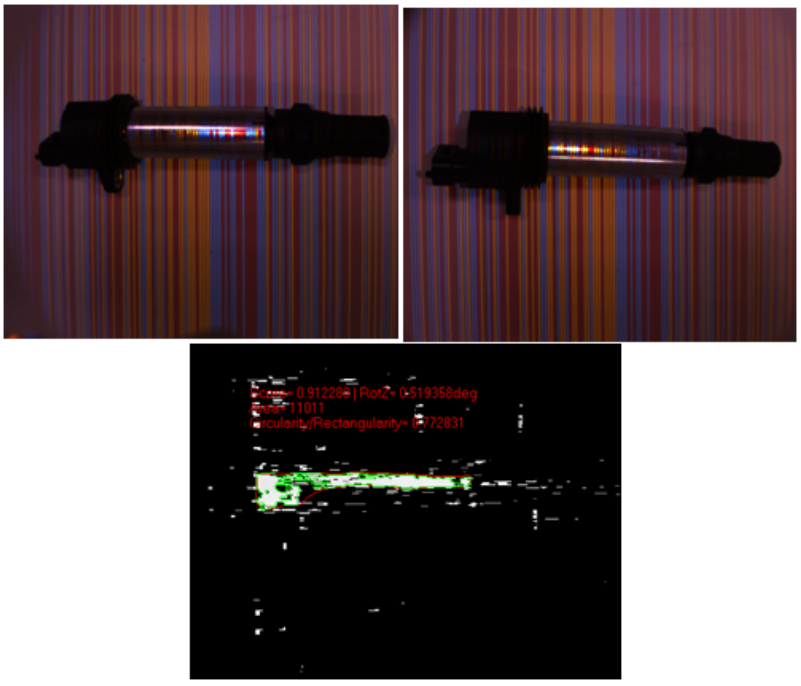


Abbildung A.28: o.: Beispielbildpaar des Bauteils Nummer 34 (Zündspule) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.23: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 34 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	7,392	9,864	11,863	39,979	9,456	35,925
Versuch A: Rng	27,862	58,062	50,055	154,712	52,392	233,897
Versuch B: σ	5,573	13,846	6,927	37,430	15,856	46,517
Versuch B: Rng	54,714	88,651	62,429	348,510	140,193	356,854
Versuch C: σ	8,364	14,552	8,484	54,568	20,690	84,282
Versuch C: Rng	77,940	91,445	71,013	335,849	105,007	355,405

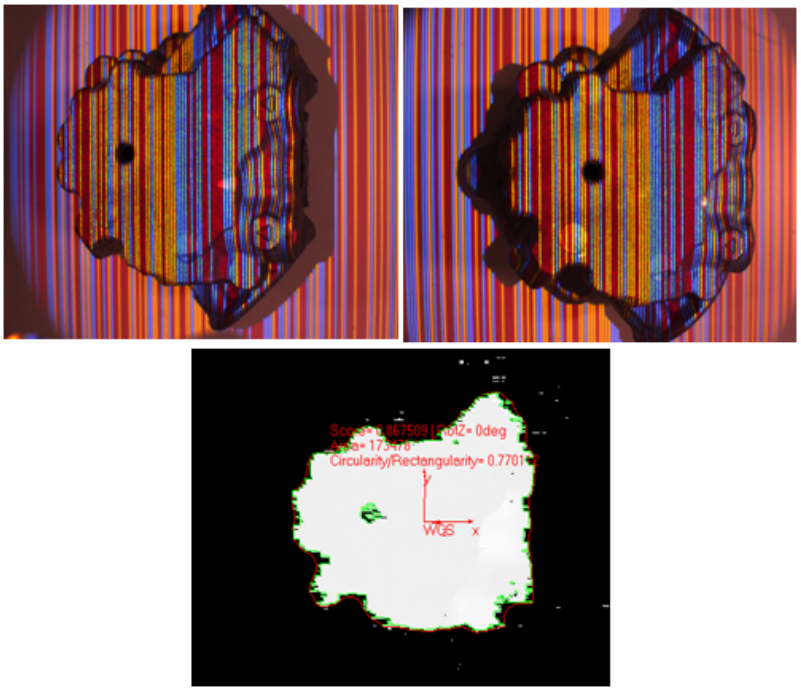


Abbildung A.29: o.: Beispielbildpaar des Bauteils Nummer 37 (Gehäuse) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.24: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 37 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,018	0,014	0,012	0,019	0,019	0,040
Versuch A: Rng	0,088	0,059	0,042	0,067	0,063	0,197
Versuch B: σ	0,220	0,164	0,067	0,078	0,072	0,339
Versuch B: Rng	1,243	0,891	0,330	0,544	0,382	1,856
Versuch C: σ	0,296	0,417	0,111	0,146	0,136	0,422
Versuch C: Rng	1,383	1,485	0,484	0,685	0,632	1,830

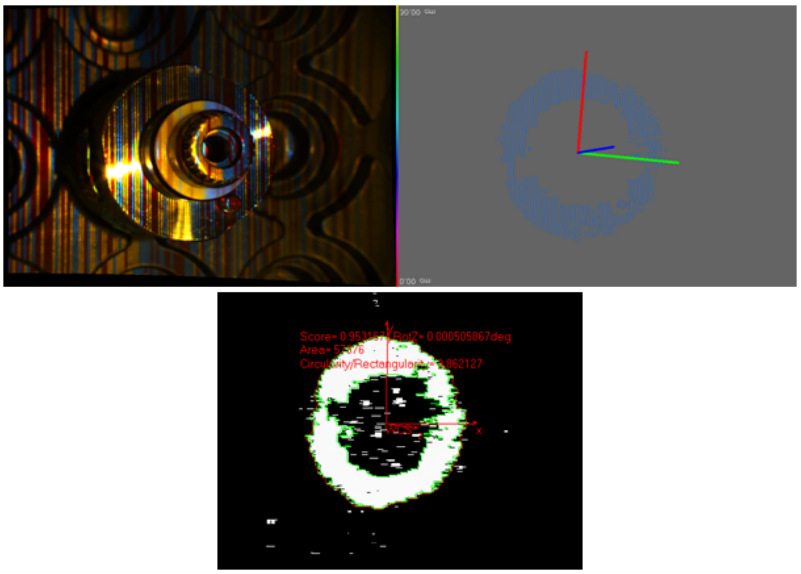


Abbildung A.30: ol: linkes rektifiziertes Kamerabild des Bauteils Nummer 101 (Zylinder) mit Streifenmusterbeleuchtung; or: segmentiertes Tiefenbild mit berechneten Hauptachsenrichtugen (1.HA rot, 2.HA grün, 3.HA blau); u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.25: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 101 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,176	0,188	0,031	0,125	0,144	
Versuch A: Rng	0,746	0,759	0,133	0,430	0,601	
Versuch B: σ	0,960	0,748	0,470	0,200	0,261	
Versuch B: Rng	4,156	4,422	1,623	1,188	1,251	

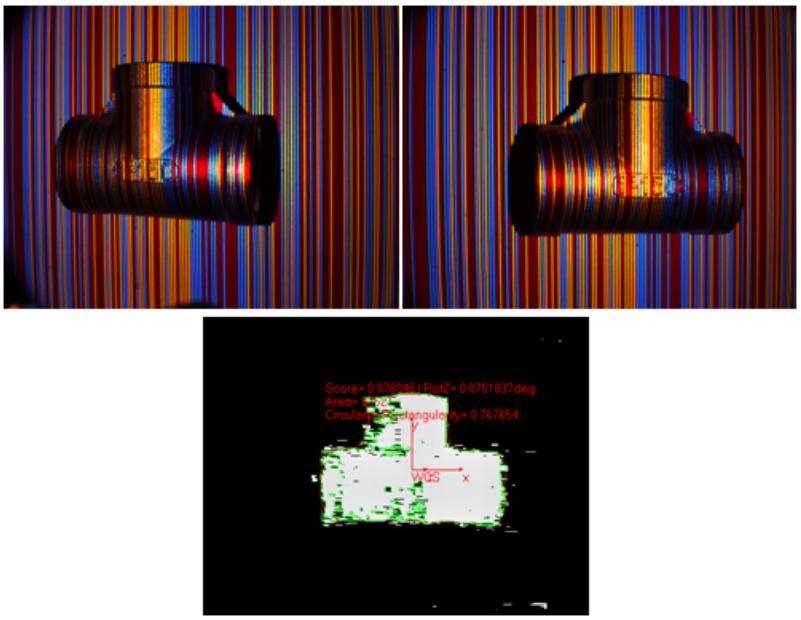


Abbildung A.31: o.: Beispielbildpaar des Bauteils Nummer 103 (Rohr-T-Stück) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.26: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 103 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,050	0,052	0,011	0,043	0,053	0,093
Versuch A: Rng	0,232	0,201	0,047	0,178	0,222	0,325
Versuch B: σ	0,186	0,313	0,071	0,120	0,231	0,329
Versuch B: Rng	0,899	1,690	0,367	0,619	1,254	1,649
Versuch C: σ	0,595	0,831	0,122	0,141	0,303	0,558
Versuch C: Rng	2,132	2,828	0,491	0,615	1,478	2,620

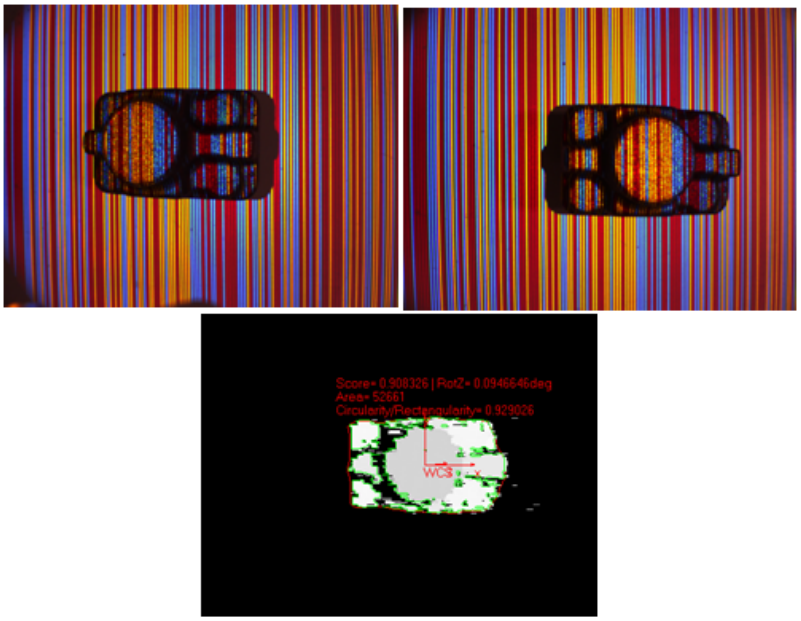


Abbildung A.32: o.: Beispielbildpaar des Bauteils Nummer 104 (Gußteil) mit Streifenmusterbeleuchtung; u.: rekonstruiertes Tiefenbild (Grauwert entspricht dem Tiefenwert), Abb. aus Halcon

Tabelle A.27: Ergebnisse der Genauigkeitsmessung des Bauteils Nummer 104 in 6 Freiheitsgraden (3 Translation, 3 Rotation) jeweils Streuung um Null und Range (Maximalwert - Minimalwert)

	$\sigma(T_x)$ /Rng(T_x) in mm	$\sigma(T_y)$ /Rng(T_y) in mm	$\sigma(T_z)$ /Rng(T_z) in mm	$\sigma(R_x)$ /Rng(R_x) in Grad	$\sigma(R_y)$ /Rng(R_y) in Grad	$\sigma(R_z)$ /Rng(R_z) in Grad
Versuch A: σ	0,043	0,031	0,010	0,045	0,053	0,107
Versuch A: Rng	0,220	0,126	0,044	0,201	0,181	0,371
Versuch B: σ	0,354	0,772	0,089	0,201	0,363	0,922
Versuch B: Rng	2,300	5,788	0,421	1,455	2,031	7,222
Versuch C: σ	1,117	1,225	0,200	0,515	0,719	2,447
Versuch C: Rng	4,241	4,915	0,821	2,620	3,771	10,004

Danksagungen

Sehr herzlich danken möchte ich allen, die das Entstehen dieser Arbeit ermöglicht und gefördert haben, ganz besonders meinem Doktorvater Prof. Dr. Bernd Jähne vom Heidelberg Collaboratory for Image Processing (HCI) an der Universität Heidelberg für sein stets offenes Ohr und viele hilfreiche Ratschläge zur rechten Zeit, sowie Dr. Christian Knoll, meinem Betreuer bei der Robert Bosch GmbH, für die Hilfe bei so manchem Algorithmus.

Mein besonderer Dank gilt auch meinen weiteren Kollegen bei Bosch, insbesondere Florian Haug, für die gute Zusammenarbeit und ihre Unterstützung in Rat und Tat während unserer gemeinsamen Jahre 2008-2011 in Labor und Büro in Schwieberdingen bei Stuttgart, sowie Carles Matabosch von der Firma Aqsense für die fruchtbare Zusammenarbeit.

Von ganzem Herzen danke ich meiner Familie, insbesondere meinem Bruder Benjamin Glas und meinem Vater Hans Joachim Glas, für viele wertvolle Hinweise und das gewissenhafte Korrekturlesen und ganz besonders meiner Partnerin Sabine Haug und allen meinen Freunden für ihren Ansporn, ihre Unterstützung und ihre Geduld während aller Höhen und Tiefen auf dem Weg zum Abschluss dieser Arbeit.

Erklärung gemäß § 8(3)b) und c) der Promotionsordnung

Hiermit erkläre ich, dass ich die vorliegende Dissertation selbstständig verfasst und mich dabei keiner anderen als der von mir ausdrücklich bezeichneten Quellen und/oder Hilfen bedient habe. Des Weiteren bestätige ich hiermit, dass ich an keiner anderen Stelle ein Prüfungsverfahren beantragt, bzw. diese Dissertation in der vorliegenden oder anderer Form bereits anderweitig verwendet oder einer anderen Fakultät als Dissertation vorgelegt habe.

Manuel Glas

Literaturverzeichnis

[AEJ97] M. Hebert A. E. Johnson. Surface Registration by Matching Oriented Points. *Proc. IEEE int. Conf. On Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, Canada*, pages 121–128, 1997.

[AL08a] F. Durand A. Levin, W.T. Freeman. Understanding Camera Trade-offs Through a Bayesian Analysis of Light Field Projections. *ECCV*, pages 88–101, 2008.

[AL08b] F. Furand A. Levin, W. T. Freeman. Understanding Camera Trade-Offs through a Bayesian Analysis of Light Field Projections. *ECCV*, pages 88–101, 2008.

[AMAD11] P. Azad, D. Münch, T. Asfour, and R. Dillmann. 6-DoF Model-based Tracking of Arbitrarily Shaped 3D Objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[Aqs] Firma Aqsense. SAL3D Machine Vision Software. Internet: www.aqsense.com.

[AR82] A. C. Kak A. Rosenfeld. *Digital Picture Processing, Band I und II.* Academic Press San Diego, 2 edition, 1982.

[Bel57] Richard Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[Bor] Rüdiger Borsdorf. Matching hochaufgelöster 3D-Scandaten von teildeformierten Objekten an CAD-Modelle: Verfahrens-, Laufzeit- und Präzisionsoptimierung. Mathematische Fakultät, TU Chemnitz.

[Bär01] Christian Bär. *Elementare Differentialgeometrie.* Walter de Gruyter, Berlin New York, 2001.

[BW05] V. Vaish E.V. Talvala E. Antunez A. Barth A. Adams M. Horowitz M. Levoy B. Wilburn, N. Joshi. High Performance Imaging using Large Camera Arrays. *ACM Trans. Graph*, (24):765–776, 2005.

[CP11] Firma Raytrix Christian Perwass. Plenoptische Kamera. Internet: www.raytrix.de, 2011.

[dB46] N. G. de Bruijn. A Combinatorial Problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, (49):758–764, 1946.

[Dol05] C. Dold. Extended Gaussian Images for the Registration of Terrestrial Scan Data. *Proceedings of the ISPRS Workshop Laser scanning*, 2005.

[Dos09] Sven Dose. Schnelle 3D-Oberflächenrekonstuktion durch Stereokorrespondenzzuordnung von Streifenmustern. Fakultät für Elektro und Informationstechnik, TU Dresden, 2009.

[DS02] Richard Szeliski Daniel Scharstein. A Taconomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 24:7–42, 2002.

[FB99] H. Rushmeier C. Silva G. Taubin F. Bernardini, J. Mittleman. The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 1999.

[For05] Frank Forster. Real-Time Range Imaging for Human-Machine Interfaces. Dissertation TU München, 2005.

[For06] Frank Forster. *A High-Resolution and High Accuracy Real-Time 3D Sensor Based on Structured Light.* 2006.

[For09] Forschergruppe. Willow Garage. http://www.willowgarage.com/, 2009.

[FS92] G. Medoni F. Stein. Structural Indexing: Efficient 3-D Object Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:125–145, 1992.

[Gmb80] MVTec Software GmbH. Halcon - Solution Guide II-F: Machine Vision. Softwaredokumentation, Version 8.0.

[Gro11] Marc Gronle. Praktikumsbericht. Universität Stuttgart - Praktikumsbericht, 2011.

[Hau09] Florian Haug. APA2-09-08 Ansichtsbasierte Objekterkennung. Interner Entwicklungsbericht Robert Bosch GmbH CR/APA2, 2009.

[Hau11] Florian Haug. Ansichtsbasierte 6 DoF Objekterkennung mit lokalen kovarianten Regionen. Dissertation Universität Heidelberg, 2011.

[HE94] E. Mücke H. Edelsbrunner. Three Dimensional Alpha Shapes. *ACM Transactions on Graphics*, 1994.

[HH92] T. Duchamp J. McDonald W. Stützle H. Hoppe, T. Derose. Surface Reconstuction from Unorganized Point Clouds. *ACM Siggraph*, 1992.

[HM95] S. K. Nayar H. Murase. Visual Learning and Recognition of 3-D Objects from Appearance. *Int. J. Comput. Vision*, 14:5–24, 1995.

[Hor84] B. Horn. Extended Gaussian Images. *Proceedings of the IEEE*, 72(12):1671–1686, 1984.

[Hor86] B. Horn. *Robot Vision.* MIT Electrical Engineering and Computer Science Series, 1986.

[Hor87] B. Horn. Closed-Form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America*, (4):629–642, 1987.

[Hou62] P. Hough. Methods and Means for Recognising Complex Patterns. United States Patent 3 069 654, 1962.

[JB97] J. Salvi J. Batlle. Recent Progress in Structured Light in order to Solve the Correspondence Problem in Stereo Vision. *Proceedings of the 1997 IEEE Internationl Conference on Robotics and Automation*, pages 130–136, 1997.

[Jäh05] Bernd Jähne. *Digitale Bildverarbeitung*, volume 6. Springer Berlin Heidelberg New York, 2005.

[JLC98] G. C. Stockman J. L. Chen. 3D Free-Form Object Recognition using Indexing by Contour Features. *Comput. Vision Image Understand.*, 71:334–335, 1998.

[Jos94] T. Joshi. HOT Curves for Modeling and Recognition of Smooth Curved 3D Objects. *IEEE Conf. Computer Vision and Pattern Recognition, Seattle, Washington*, pages 876–880, 1994.

[JS06] D. Fofi J. Forest J. Salvi, C. Matabosch. A Review of Recent Image Registration Methods with Accuracy Evaluation. *Image and Vision Computing*, 2006.

[KB96] A. J. Stoddart K. Brunnström. Genetic Algorithms for Free-Form Surface Matching. *Int. Conf. on Pattern Recognition*, pages 689–693, 1996.

[Kie09] Andreas Kießelbach. 3D Modellerstellung aus Disparitätsbildern. Diplomarbeit FH Gießen-Friedberg, 2009.

[Kno08] Christian Knoll. APA2-08-08 RobotVision-3D-RangeImage-Entwicklungsstand 2008. Interner Entwicklungsbericht Robert Bosch GmbH CR/APA2, 2008.

[Kno09] Christian Knoll. APA2-09-11 Aktueller Entwicklungsstand der Robot-Vision. Interner Entwicklungsbericht Robert Bosch GmbH CR/APA2, 2009.

[KSA87] S. D. Blostein K. S. Arun, T. S. Huang. Least Square Fitting of Two 3-D Point Sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987.

[Kui02] J. B. Kuipers. *Quaternions and Rotation Sequences.* Princeton University Press, 2002.

[MAF81] Robert C. Bolles Martin A. Fischler. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[Mat07] Firma Mathworks. Matlab Software. Internet: www.mathworks.com, 2007.

[MG00] C. T. Silva M. Gopi, S. Krishnan. Surface Reconstruction Based on Lower Dimensional Delaunay Triangulation. *EUROGRAPHICS 2000*, 2000.

[MK90] L. Sirovich M. Kirby. Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12:103–108, 1990.

[ML96] P. Hanrahan M. Levoy. Light Field Rendering. *Proc. ACM Siggraph*, pages 31–42, 1996.

[Mok95] F. Mokhtarian. Silhouette-Based Isolated Object Recognition through Curvature Scale Space. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:539–544, 1995.

[NA01] R. K. Kolluri N. Amenta, S. Choi. The Power Crust. *SMA 2001*, 2001.

[PA06] Ruediger Dillmann Pedram Azad, Tamim Asfour. Combining Appearance-Based and Model-Based Methods for Real-Time Object Recognition and 6D Localization. *Proc IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 9–15, 2006.

[Phy] Firma Physikinstrumente. Datenblatt Linearaktor. Internet: www.physikinstrumente.de.

[PJB92] Neil D. McKay Paul J. Besl. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–255, 1992.

[RB76] W. Tietz S. Kubicek R. Bittner, D. Ilse. *Mathematik in Übersichten.* Aulis Verlag Deubner und Co KG Köln, 1 edition, 1976.

[RJC99] Patrick J. Flynn Richard J. Campbell. Eigenshapes for 3D Object Recognition in Range Data. *Proc. IEEE Conf. Computer Vision and Pattern Recognition, Fort Collins, Colorado*, pages 173–180, 1999.

[RJC01] Patrick J. Flynn Richard J. Campbell. A Survey of Free-Form Object Representation and Recognition Techniques. *Computer Cision and Image Understanding*, 81:166–210, 2001.

[Roh11] Simone Rohleder. Rekonstruktion der Oberfläche einer ungeordneten 3D-Punktewolke zur Anwendung in der Objektlageerkennung. Bachelorarbeit Hochschule für Technik Stuttgart, 2011.

[RZ07] L. Ahrenberg M. Magnor M. Gross R. Ziegler, S. Bucheli. A Bidirectional Light Field-Hologram Transform. *Computer Graphics Forum*, (26):435–446, 2007.

[SAN97] S. K. Nayar S. A. Nene. A Simple Algorithm for Nearest-Neighbor Search in High Dimensions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19:989–1003, 1997.

[Say09] Sabine Sayler. APA1-09-04 Stand der Technik Manipulation. Interner Entwicklungsbericht Robert Bosch GmbH CR/APA1, 2009.

[Say11] Sabine Sayler. Universelle Manipulationsstrategien für die industrielle Montage. Dissertation Karlsruher Institut für Technologie (KIT), 2011.

[SG02] X. Pennec S. Granger. Multi-scale EM-ICP: A Fast and Robust Approach for Surface Registration. *ECCV*, (IV):418–432, 2002.

[SR01] Marc Levoy Szymon Rusinkiewicz. Efficient Variants of the ICP Algorithm. *Proceedings of the Third Intl. Conf. On 3D Digital Imaging and Modeling*, pages 145–152, 2001.

[SS99] G. Häusler S. Seeger. A Robust Multiresolution Registration Approach. *Proceedings VMV*, pages 75–82, 1999.

[Sto11] M. Stotz. Adaptive Segmentierung von Tiefenbildern für die 3-D-Objektlageerkennung auf Basis von kombinierten regelgeometrischen Elementen. Dissertation Universität Stuttgart (Fraunhofer IPA), 2011.

[SW11] B. Jähne S. Wanner, J. Fehr. Generating EPI Representations of 4D Light Fields with a Single Lens Focused Plenoptic Camera. *7th International Symposium on Visual Computing, Las Vegas*, (7):24–26, 2011.

[SW12] B. Goldluecke S. Wanner. Globally Consistent Depth Labeling of 4D Light Fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. accepted.

[TM06] G. Häusler T. Maier. Segmentation Based Fast Registration of Free Form Surfaces in the Euclidian Space. Technical Report, Institute of Optics, Information and Photonics, University of Erlangen-Nürnberg, 2006.

[TR04] F.A. van den Heuvel T. Rabbani. Methods for Fitting CSG Models to Point Clouds and their Comparison. *The 7th IASTED International Conference on Computer Graphics and Imaging*, pages 279–284, 2004.

[Tra09] Khanh Quan Tran. Effiziente Ähnlichkeitssuche in hochdimensionalen Bilddatenbanken. Masterarbeit Hochschule Darmstadt, 2009.

[ver09] verschiedene. Open Source Softwarepaket OpenCV mit einer Zusammenfassung üblicher Bildverarbeitungsalgorithmen. Internet, 2009.

[Wil00] Wilhelm Wilke. Segmentierung und Approximation großer Punktewolken. Dissertation TU Darmstadt, 2000.

[YC91] G. Medioni Y. Chen. Object Modeling by Registration of Multiple Range Images. *IEEE International Conference on Robotics and Automation*, pages 2724–2729, 1991.

[YW08] Xianyu Su Yongchao Wei. Novel and Fast Mapping Triangulation Algorithm for Uniorganized Point Clouds. *Optical Engineering*, 47(11)(117205), 2008.

[Zin] Ralf Zink. Folienvortrag zu Lichtfeldern bei der Robert Bosch GmbH - Abteilung CR/APA2.

[ZL02] Steven M. Seitz Zhang Li, Brian Curless. Rapid Shape Acquisition Using Color Structured Light and Multi-Pass Dynamic Programming. *Proceedings First International Symposium on 3D Data Processing Visualization and Transmission*, pages 24–36, 2002.

[ZX07] Wuzheng Tan Zhiliang Xu, Lizhuang Ma. Robust Dense Depth Acquisition Using 2-D De Bruijn Structured Light. *Entertainment Computing ICEC 2007 - Lecture Notes in Computer Science*, 4740/2007:304–314, 2007.