



**Karlsruher Institut  
für Technologie (KIT)**  
Institut für Prozessrechentechnik,  
Automation und Robotik (IPR)  
der Fakultät für Informatik

# **Markerbasierte Objektverfolgung für die Mensch-Roboter-Kooperation**

**Diplomarbeit  
von  
Beibei Cao**

Stand: 25. September 2012

Referenten: Prof. Dr.-Ing. Heinz Wörn  
Betreuer: Dipl.-Inform. Stephan Puls



# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Aufgabenstellung . . . . .	1
<b>2 Stand der Forschung</b>	<b>3</b>
2.1 Standardmarkenbasierte Verfahren . . . . .	3
2.2 Verfahren basierend auf natürlichen Merkmalen . . . . .	4
2.2.1 Kantenbasiertes Verfahren . . . . .	4
2.2.2 Optischer Fluss basiertes Verfahren . . . . .	5
2.2.3 Templatebasiertes Verfahren . . . . .	5
2.2.4 Punktebasiertes Verfahren . . . . .	5
2.3 Markenbasierte Objekterkennung . . . . .	6
2.3.1 Markenerkennung . . . . .	6
2.3.2 Markenverfolgung . . . . .	7
2.3.3 Schätzung der Transformation . . . . .	7
2.3.3.1 Geschlossene Form . . . . .	7
2.3.3.2 Iterative Closest Point Algorithmus . . . . .	8
2.3.4 Objektkalibrierung . . . . .	9
2.3.4.1 Statistisches Verfahren . . . . .	9
2.3.4.2 Dynamisches Verfahren . . . . .	9
2.3.5 Objekterkennung und Verfolgung . . . . .	10
<b>3 Grundlagen</b>	<b>12</b>
3.1 Generierung der 3D-Daten . . . . .	12
3.1.1 TOF-Sensor . . . . .	12

3.1.1.1	TOF Kamera . . . . .	12
3.1.1.2	PMD Sensor . . . . .	13
3.1.1.3	Unterschied zwischen TOF Kamera und Kinect . . . . .	14
3.1.2	Daten der PMD Kamera . . . . .	15
3.1.2.1	2D Daten . . . . .	15
3.1.2.2	3D Daten . . . . .	16
3.1.2.3	Auflösung und erkennbare Markengröße . . . . .	16
3.2	Vorverarbeitung . . . . .	17
3.2.1	Schwellwert-basierte Segmentierung . . . . .	17
3.2.2	Steuerung der Helligkeit und Kontrast . . . . .	18
3.3	Markenerkennung . . . . .	19
3.3.1	Auswahl des Erkennungsalgorithmus . . . . .	19
3.3.2	CenSurE Algorithmus . . . . .	20
3.3.2.1	Bi-level Filter . . . . .	20
3.3.2.2	Integrale Bilder . . . . .	21
3.3.2.3	Non-maximal Suppression . . . . .	22
3.3.3	Kalman-Filter . . . . .	23
3.3.3.1	Zustandsraummodellierung . . . . .	23
3.3.3.2	Das Kalman-Filter . . . . .	24
3.4	Registrierung . . . . .	25
3.4.1	Singulärwertzerlegung . . . . .	25
3.4.2	Korrespondenzuntersuchung durch Singulärwertzerlegung . . . . .	25
3.5	Bestimmung der Transformation . . . . .	27
3.5.1	Quaternion . . . . .	27
3.5.2	Beschreibung der Drehungen im Dreidimensionalen Raum mit Quaternionen . . . . .	28
3.5.3	Orientierung mit Einheitsquaternion . . . . .	29
3.6	Objekterkennung . . . . .	32
3.6.1	DBSCAN . . . . .	32
3.6.2	Teilgraph Isomorphismus . . . . .	35
<b>4</b>	<b>Implementierung</b> . . . . .	<b>37</b>
4.1	Markenanalyse . . . . .	38

4.1.1	Bildvorverarbeitung . . . . .	38
4.1.1.1	Datenstruktur des Eingabebilds . . . . .	38
4.1.1.2	Abstand Filter . . . . .	38
4.1.2	Markenerkennung . . . . .	40
4.1.2.1	Auswahl der Größe der Marken . . . . .	40
4.1.2.2	STAR Detektor . . . . .	41
4.1.2.3	Markenerkennung . . . . .	42
4.1.2.4	Steuerung der Helligkeit und des Kontrast . . . . .	43
4.1.3	Markenverfolgung . . . . .	45
4.1.3.1	Verbesserung der Singulärwertzerlegungsverfahren . . . . .	45
4.1.4	Segmentierung . . . . .	46
4.2	Objektlernen . . . . .	48
4.2.1	Bestimmung der Orientierung . . . . .	48
4.2.2	Markenanordnung . . . . .	49
4.2.3	Darstellung des Strukturgraphen . . . . .	50
4.2.3.1	Bestimmung der stabilen Knoten . . . . .	50
4.2.3.2	Kanteneinfügung . . . . .	51
4.3	Zugriff des Strukturgraphen von Datei . . . . .	52
4.4	Objekterkennung und Verfolgung . . . . .	53
4.4.1	Kandidaten der Objekterkennung . . . . .	53
4.4.2	Objekterkennung . . . . .	54
4.4.3	Bestimmung der Orientierung . . . . .	56
4.5	Bildersteuerung . . . . .	57
<b>5</b>	<b>Experimentelle Auswertung</b>	<b>59</b>
5.1	Teilweise Evaluation . . . . .	60
5.1.1	Abstand Filter . . . . .	60
5.1.2	Helligkeitssteuerung . . . . .	60
5.1.3	Verbesserung der Singulärwertzerlegungsverfahren . . . . .	60
5.1.4	Aktualisierung des Strukturgraphen . . . . .	60
5.1.5	Teilgraph-Isomorphismus . . . . .	60
5.1.6	Bildersteuerung . . . . .	60
5.2	Globale Evaluation . . . . .	60

<b>6 Schlusswort</b>	<b>61</b>
----------------------	-----------

<b>Literaturverzeichnis</b>	<b>63</b>
-----------------------------	-----------

# Kapitel 1

## Einleitung

### 1.1 Motivation

Roboter haben ihre Stärke in der Wiederholung von einfachen Handhabungstätigkeiten. Menschen dagegen sind mit ihren kognitiven Fähigkeiten einzigartig, etwa mit ihrem Verständnis der Aufgabe. Die Kombination von Mensch und Roboter kann Aufgaben stark rationalisieren, sofern jedem die optimalen Arbeitsteile zugewiesen sind. Die Anwendungsbereiche der Mensch-Roboter-Kooperation vergrößern sich heutig immer schneller auf dem Feld der Medizin sowie der Industrie. Damit Mensch und Roboter in einer geringen Entfernung sicher und effizient zusammenarbeiten können, ist die Erkennung bzw. die Verfolgung von Menschen und Objekten für ein Mensch-Roboter-Kooperation-System notwendig. Die Menschenerkennung garantiert die Sicherheit für den Menschen und liefert gleichzeitig Informationen über die Blickrichtung, um Aussagen über die Aufmerksamkeit des Menschen treffen zu können. Die Objekterkennung vereinfacht die Kommunikation zwischen Mensch und Roboter, dadurch können die Fremdobjekte ohne weitere Programmierung direkt vom Roboter erkannt werden. Außerdem vermeidet die Objektverfolgung auch die Kollision zwischen Roboter und anderen Anlagenteilen.

### 1.2 Aufgabenstellung

Das Rahmenwerk MAROCO wird am IPR entwickelt, damit Menschen und Roboter in einer gemeinsamen Umgebung sicher zusammenarbeiten können. Die Erfassung des Mensch und Handlungsanalyse in der Szene erlaubt es, die Gefahr von der Roboterbewegung für Menschen zu minimieren. Jedoch ist die Erkennung zurzeit auf das Menschmodell beschränkt. Alle anderen Objekte werden von dem System als Zylinder dargestellt. Das Ziel der Arbeit ist, die verschiedenen Objekte zu kalibrieren und die entsprechenden geometrischen Charakteristika in dem System zu speichern, dann die

Objekte mit Hilfe der gespeicherten Informationen zu erkennen und zu verfolgen. Beide Schritte sollen in Echtzeit durchgeführt werden.

Folgende Ziele sollen erreicht werden:

- Objektkalibrierung,
- Darstellung und Speichern des charakteristischen Modells der Objekte,
- Objekterkennung und Verfolgung,
- Einhaltung der Echtzeitbedingungen (Framerate  $\geq 30\text{fps}$ ).

# Kapitel 2

## Stand der Forschung

3D Objekterkennung und Verfolgung kommt in vielen Anwendungsbereichen zum Einsatz. Daher wurden viele Algorithmen bzw. Systeme dafür in den vergangenen Jahrzehnten entwickelt. Ein gutes Beispiel ist das kommerzielle Erkennungssystem von VICON [VICON]. In dem System werden 8 Kameras benutzt, die von verschiedenen Richtungen das Zielobjekt bzw. Person beobachten. Einige weiße Marken werden vorher am Ziel angebracht, damit seine Positionen und Bewegungen von den Kameras gut erkannt werden können. Nach Vergleichen der Bilder von verschiedenen Kameras kann ein 3D Modell des Ziels in Echtzeit erzeugt werden. Das System wird im Bereich von Computerspielen und Filmindustrie sehr oft benutzt. Ein anderes Beispiel ist das neue Gerät Kinect von Microsoft XBox360 [Kinect]. Eine 3D Kamera kann die 3D Daten von Spielern ansammeln, damit die Spieler das Spiel direkt mit ihren Körpern statt des traditionellen Kontroller steuern können. Die Analyseverfahren der Objekterkennung basieren auf unterschiedlichen Charakteristika der Objekte und sind für verschiedene Typen von Objekten geeignet. In der Arbeit von Lepetit und Fua sind die aktuelle Verfolgungsverfahren in zwei große Gruppen unterteilt worden: auf Marken basierte Objektverfolgungen und auf natürlichen Merkmale basierte Objektverfolgungen. Die Verfahren in der zweiten Gruppe können weiter in kantenbasiertes Verfahren, optischer Fluss basiertes Verfahren, Templatebasiertes Verfahren, Punktebasiertes Verfahren und das SLAM-Verfahren unterteilt werden [Lepetit & Fua 2005]. Im folgenden Abschnitt werden kurz die Details jedes Verfahrens erklärt.

### 2.1 Standardmarkenbasierte Verfahren

Die Verfolgungsverfahren können in zwei Schritte unterteilt werden: zuerst der Informationen von Bildsequenzen ansammeln um dann die Position des erkannten Objekts zu bestimmen. Die vordefinierte Marken können in beiden Schritten mehr Information liefern, damit die Objekte schneller und einfacher verfolgt werden können. Deshalb sind in diesem Bereich viele Systeme für die Erweiterte Realität implementiert worden. Ein

Echtzeitsystem für Erweiterte Realität wurde von Zhang und Navab für Objektverfolgung in einer Industrieumgebung realisiert [Zhang & Navab 2000]. Sie haben eine Gruppe von 4 Vierecken als eine Marke benutzt. Die Marke wird durch Farbe und weiße Flecken innerhalb der Vierecke kodiert.

Ein anderes System heißt ARToolKit, was vom HITLab der Universität Washington entwickelt wird. Es ist eine bekannte Software-Bibliothek zur Entwicklung von Anwendungen für die Erweiterte Realität [ARToolKit]. In ARToolKit wird ein Viereck mit schwarzer Umrandung als Marke benutzt. Das Muster in der Mitte kodiert die Marke und kann frei gewählt werden. Das Eingabebild wird zuerst in ein Binärbild umgewandelt und dann alle verbundenen schwarzen Pixel extrahiert. Die Figur innerhalb der schwarzen Umrandung wird segmentiert und mit dem früheren definierten Muster verglichen. Durch den Vergleich kann man die Projektivität zwischen Kamerakoordinatensystem und Musterkoordinatensystem bestimmen.

ARToolKit liefert eine hohe Frame-Rate mit bis zu 30 fps bei niedrigem CPU-Bedarf. Eine dicke schwarze Umrandung garantiert die Stabilität des Systems und die Marke kann in niedriger Auflösung sehr gut erkannt werden. Ein anderer wichtige Vorteil ist, dass die Verfolgung der ARToolKit keine Initialisierung braucht. Dadurch wird nicht nur die Laufzeit am Anfang des Verfahren gespart, kann aber auch Chaos vermeiden, wenn die eingegebene Bildsequenz abgebrochen wird.

## 2.2 Verfahren basierend auf natürlichen Merkmalen

### 2.2.1 Kantenbasiertes Verfahren

Das kantenbasierte Verfahren wurde in früheren Objektverfolgungssystemen häufig benutzt, weil es effizient und einfach zu realisieren ist [Lepetit & Fua 2005]. Die Hauptidee dieses Verfahrens ist entweder die Kanten des Objekts direkt von dem Bild herauszufinden und zu verfolgen, oder den Teil des Bildes mit starkem Gradient zu betrachten, damit man die Konturen des Objekts zum nächsten Zeitpunkt vorhersagen kann. RA-PiD war eines der frühesten 3D Verfolgungsverfahren, das in Echtzeit laufen konnte [Harris 1992]. Vacchetti und Lepetit haben ein neues, effizienteres Verfolgungsverfahren entwickelt, was mehr als eine Voraussagen für die ausgewählten Steuerpunkte darstellen [Vaccetti, Lepetit & Fua 2004]. Diese Erweiterung verstärkt die Stabilität der Verfolgung und erfüllt weiterhin die Echtzeitbedingung.

### 2.2.2 Optischer Fluss basiertes Verfahren

Der Optische Fluss ist ein Vektorfeld, das die Bewegungsrichtung und Bewegungsgeschwindigkeit für jeden Bildpunkt einer Bildsequenz bezeichnet. Die Berechnung des Optischen Fluss kann als eine Differentialgleichung zusammengefasst werden und das Lösungsverfahren wurde von Horn und Schunck entwickelt [Horn & Schunck 1981]. Black und Yacoob benutzten reine Optische Fluss basierte Verfahren für die Verfolgung kleiner Veränderungen auf menschlichem Gesicht, um den Gesichtsausdruck zu bestimmen [Black & Yacoob 1997]. Außerdem wurde ein Verfolgungssystem für den Innerstadt Verkehr von Haag und Nagel durch die Verknüpfung der Information von Optischem Fluss und Kanten des Objekts implementiert [Haag & Nagel 1999].

### 2.2.3 Templatebasiertes Verfahren

Im Templatebasierten Verfahren wird ein Objekt nicht durch lokale Merkmale z.B. Kanten oder Punkte, sondern durch das globale Charakteristikum erkannt und verfolgt. Das Verfahren ist geeignet für komplexe Objekte, die nicht einfach durch lokale Merkmale bezeichnet werden können [Lepetit & Fua 2005]. Der Lucas-Kanade Algorithmus wurde anfänglich entwickelt, um den Optischen Fluss zu berechnen [Lucas & Kanade 1981], ist aber auch für die 2D templatebasierte Verfolgung nutzbar. Jurie und Dhome haben einen Algorithmus für die Verfolgung von ebenen Objekten mithilfe von Hyperebenen entwickelt [Jurie & Dhome 2001]. In ihrer Arbeit wurde die Approximation der Abbildung des Objekts auf Hyperebenen abgeschätzt, dadurch die Translation des Objekts bestimmt werden kann.

### 2.2.4 Punktebasiertes Verfahren

Der Unterschied zwischen dem punktebasierten Verfahren und den oben beschriebenen Verfahren ist, dass nur lokale Merkmale betrachtet werden. Im Vergleich zum Verfahren, das globale Merkmale behandelt, ist das Verfolgungsverfahren mit lokalen Merkmalen viel stabiler, wenn es Kollision für mehr Objekte gibt, oder die Messung der Merkmalen stark stört wird [Lepetit & Fua 2005]. Ein Verfahren wurde von Zhang et al. im Jahre 1995 realisiert, was die nicht kalibrierten Bilder als Eingabe benutzen kann [Zhang et al. 1995]. In dem Verfahren wird kein Modell der Epipolargeometrie verwendet, wodurch viele komplexe Berechnungen vermieden werden. Eine andere Möglichkeit für die Punkteverfolgung ist der Kanade-Lucas-Tomasi(KLT) Tracker, was auf der Arbeit von [Lucas & Kanade 1981] begründet wurde. Sie haben eine Approximation für den Unterschied zwischen zwei Bildern definiert. Mithilfe des Iterationsverfahrens von Newton-Raphson wird die Approximation minimiert. Dadurch kann die Translation des Objekts bestimmt werden. Tomasi erweiterte den Algorithmus von Lucas und Kanade mit einer besseren Strategie zur Auswahl von Merkmalen [Tomasi & Kanade]. Der dritte Schritt wurde von Shi und Tomasi vervollständigt [Shi & Tomasi 1994]. Sie verbesserten

weiter die Auswahl der Punkte mit der Ähnlichkeit zwischen dem Anfangsbild und das aktuelle Bild. Diese Ähnlichkeit wird durch einem Modell von affiner Abbildung bestimmt. Außerdem benutzen sie gleichzeitig zweites Modell von reiner Translation, um das Objekt mit hoher Seriosität und Präzision zu verfolgen.

## 2.3 Markenbasierte Objekterkennung

Rhijn und Mulder haben ein markenbasiertes Verfahren entwickelt, was das Verdeckungsproblem behandelt [Rhijn & Mulder 2005]. Einige runde, hoch reflektierende Marken werden auf dem Objekt angebracht. In der Initialisierungsphase speichert das System die Charakteristika des Objekts als einen 3-dimensionalen vollständigen Graph. Wenn das Objekt wiedererkannt werden soll, führt das System zuerst einen Kalibrationsalgorithmus durch, um die verschiedenen Objekte zu differenzieren. Dann vergleicht das System für jedes Objekt die sichtbaren Marken mit dem in der Initialisierungsphase gespeicherten vollständigen Graphen.

### 2.3.1 Markenerkennung

Es gibt heutzutage viele benutzte Standardalgorithmen der Markenerkennung. Die Harris Matrix wird von der Summe der partiellen Ableitungen des Eingabebildes dargestellt. Eine Ecke wird genau dann als ein Merkmal erkannt, wenn die beiden Eigenwerte der Harris Matrix die positive und genug groß sind [Harris & Stephens 1988]. Ein anderer Algorithmus für diese Eckenerkennung heißt „Features from Accelerated Segment Test“ (FAST) und wurde von Rosten und Drummond im Jahre 2006 veröffentlicht [Rosten & Drummond 2006]. Außer der Eckenerkennung können die Marken sich aber auch als den Klecks annähern. Das Ziel dieser Art der Verfahren ist, die Punkte zu extrahieren, die unterschiedliche Eigenschaften zu der Umgebung haben, z.B Helligkeit und Farbe. Lowe hat seinen Algorithmus „Scale-invariant feature transform“ (SIFT) im Jahre 2004 veröffentlicht, was starke Stabilität gegen Transformation, Beleuchtungsvariation bzw. Bildrauschen hat [Lowe 2004]. Das Detektor arbeitet auf Basis einer Skalenraum-Analyse mit Difference of Gaussians (DoG), was eine effiziente Approximation des skalen-normalisierten LoG-Operators (Laplacian of Gaussian) ist. Das locale Extremum wird in DoG Bildern durch Vergleich der direkten Nachbarn bzw. der Nachbarn aus den benachbarten DoG Bildern gesucht. Der entsprechende Deskriptor wird als 128 dimensionalen Vektor definiert und besteht aus in Teil-Quadranten unterteiltem Gradientenhistogramm. Bay et al. erweiterte die Arbeit von Lowe mit dem „Speeded Up Robust Feature“ (SURF) [Bay et al. 2006]. Der RechtecksfILTER wird in ihrer Arbeit statt DoG verwendet, um den Detektionsablauf zu beschleunigen. Außerdem wird der Deskriptor sich auch um Faktor 2 verkleinert, damit der Vergleich der zwei Merkmalsvektoren weniger Zeit kostet. Der Algorithmus „Center Surround Extremas“ (CenSurE oder STAR) benutzt sogenannten „Center-surround“ Filter, um

den Laplace-Operator zu approximieren, damit die Betrachtung der Merkmale in allen Skalar-Räumen schnell durchgeführt werden kann [Agrawal, Konolige & Blas 2008]. Wegen diesen Veränderungen ist der CenSurE Algorithmus viel effizienter und stabiler. Die vorhandenen Realisierung obiger Algorithmen können in der freien Programm-bibliothek OpenCV gefunden werden. Anhand von OpenCV hat Odessa in seinem Blog die Geschwindigkeit, Stabilität bzw. die Genauigkeit dieser Algorithmen verglichen [Odessa 2011]. Seinem Ergebnis zufolge scheint, dass der STAR Algorithmus die niedrigste durchschnittliche Fehler-Rate hat bzw. den geringsten Berechnungsaufwand benötigt.

### 2.3.2 Markenverfolgung

Nach Bestimmung der Marken, sollen diese Marken in einer Bildsequenz verfolgt werden. Scott und Longuet-Higgins haben einen eleganten und einfachen Algorithmus entwickelt [Scott & Longuet-Higgins 1991]. Sie haben das Problem zu einer Matrix zusammengefasst, deren Elemente als die Distanz zwischen verschiedenen Merkmalen definiert werden. Durch eine Singulärwertzerlegung und Matrixersetzung werden die Abbildungen der Marken zwischen zwei Bildern bestimmt. Rhijn und Mulder verbesserten den Algorithmus von Scott und Longuet-Higgins. Sie haben die Elemente der Matrix neu definiert und fügten die Beschränkung der Epipolargeometrie ein [Rhijn & Mulder 2005].

### 2.3.3 Schätzung der Transformation

Für ein 3D Objekt ist die Markenverfolgung innerhalb von Bildern nicht ausreichend, um die komplette geometrische Information zu rekonstruieren. Deshalb soll die Pose des Objekts gleichzeitig bestimmt werden, damit ein räumlicher charakteristischer Graph für das Objekt erstellt werden kann. Natürlich kann man mit Hilfe der Ergebnisse der Markenverfolgung die relative Rotation und Translation des Objekts zwischen zwei Zeitpunkten berechnen, aber außerdem gibt es Verfahren, die durch Vergleich der Punktwolken von zwei Bildern direkt die Transformation des Objekts bestimmen können. Diese Verfahren wurden von Eggert et al. in ihrer Arbeit durch Merkmale und Lösungsverfahren in verschiedene Typen unterteilt [Eggert, Lorusso & Fisher 1997]. Die Merkmale könnten die Oberfläche, die Kanten bzw. die Punkte sein. Die Verfahren, die auf Punkte basieren, werden in der Praxis häufig benutzt und sind geeignet für die markenbasierte Objekterkennung [Eggert, Lorusso & Fisher 1997]. Die Lösungsverfahren können in iterative Verfahren und geschlossene Form unterschieden werden.

#### 2.3.3.1 Geschlossene Form

Ein effizientes Verfahren mit geschlossener Form für das Berechnen der Transformation wurde von Arun et al. zuerst am Jahr 1987 veröffentlicht [Arun, Huang & Blostein

1987]. Die Hauptidee des Verfahrens ist, eine approximierte Rotations- bzw. Translationsmatrix zwischen zwei aufeinander folgenden Bildern zu bestimmen, damit die Summe des Unterschieds zwischen den durch approximierte Transformation berechnete Positionen der Punkte und die genaue Positionen der Punkte minimiert wird. Dieses Verfahren basiert auf der Singulärwertzerlegung einer Korrelationsmatrix. Die Rotations- und Translationsmatrix werden am Ende ausgegeben. Wenn die zwei eingegebenen Punktwolken auf gleich Oberfläche liegen, liefert das Verfahren leider keine richtige Rotationsmatrix. Deshalb soll eine korrigierte Matrix darauf aufbauen, die von Umeyama [Umeyama 1991] und Kanatani [Kanatani 1994] vorschlagen wurde. Ein anderes Verfahren wurde von Horn entwickelt, was die relative Rotation durch Einheitsquaternionen beschreibt [Horn 1987]. Im Vergleich zu der Standard-Beschreibung der Rotation als Matrix ist die Quaternionendarstellung viel effizienter und stabiler. Die verbesserte Stabilität kann den Fehler vermeiden, wenn der Winkel der Rotation zur Singularität wird, z.B.  $0^\circ$  oder  $180^\circ$ . D.h. dieses Verfahren braucht keine Maßnahme für Behandlung des speziellen Winkels, was aber im Verfahren von Arun nötig ist [Arun, Huang & Blostein 1987].

Eggert et al. haben in ihrer Arbeit die obengenannten zwei Verfahren mit zwei anderen geschlossene Form-Verfahren verglichen [Eggert, Lorusso & Fisher 1997]. Wenn die Anzahl der betrachteten Punkte weniger als 100 ist, braucht das Verfahren mit Einheitsquaternionen weniger Zeit als die anderen Verfahren. Auf der anderen Seite, wenn die Anzahl der betrachteten Punkten weniger als 10 ist, liefert das Verfahren von Arun den kleinsten Fehler.

### 2.3.3.2 Iterative Closest Point Algorithmus

Der Iterative Closest Point Algorithmus ist ein Algorithmus, der es ermöglicht, Punktwolken aneinander anzupassen [ICP Wiki]. In jedem Iterationsschritt wird der korrespondierende Punkt für jeden Punkt einer Punktwolke aus anderer Punktwolke gefunden. Die Transformation zwischen beiden Punktwolken werden so bestimmt, dass die Summe des Abstands der korrespondierenden Punkte minimiert wird. Dieser Vorgang wird wiederholt, bis die Veränderung des mittleren quadratischen Fehlers zwischen zwei folgenden Schritten unter einer Schranke liegt. Der Algorithmus wurde erst von Chen und Medioni [Chen & Medioni 1991] vorgestellt und ICP wurde als Name des Algorithmus von Besl und McKay in ihrer Arbeit erstmals benutzt [Besl & McKay 1992]. Doria et al. erweiterten den Algorithmus mit gewichtetem Kriterium für das Rauschen [Dorai, Weng & Jain 1997]. Ein Vergleich der verschiedene ICP Algorithmen vor dem Jahr 2001 wurde von Rusinkiewicz und Levoy erstellt [Rusinkiewicz & Levoy 2001]. Der grundlegende ICP Algorithmus wurde von ihnen in 6 Schritte unterteilt. In jedem Schritt wurde die Leistung des Algorithmus verglichen und ihr Einfluss auf den ganzen Algorithmus diskutiert. Eine andere Veränderung wurde von Chavarria und Sommer vorgeschlagen, in der die Kontur des Objekts auch in der Schätzung der Pose betrachtet wird [Chavarria & Sommer 2007].

### 2.3.4 Objektkalibrierung

#### 2.3.4.1 Statistisches Verfahren

Im Ablauf der markenbasierten Objekterkennung werden alle Objekte durch einige Marken einzig definiert. Deshalb kann für jedes Bild die Kalibrierung der Objekte als Clusteranalyse der Marken zusammengefasst werden. Josiger und Kirchner haben einen Vergleich über drei moderne Clusteralgorithmen erstellt [Josiger & Kirchner 2003]. Sie sind BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies), DBSCAN (Density Based Spatial Clustering of Applications with Noise) und CHAMELEON-Algorithmus. BIRCH-Algorithmus basiert auf dem Clustering-Feature-Tree, was ein balancierter Baum mit Verzweigungsfaktor und Schwelle als zusätzlichen Parametern ist [Zhang et.al 1996]. Ein vorhandener Clusteralgorithmus wird für die Blätter des Baums durchgeführt, damit die angeforderte Gruppen ausgesucht werden können. In CHAMELEON werden zuerst die Nachbarn des betrachteten Objekts bestimmt. Dann wird die gesamte Menge der Daten durch die Ähnlichkeit in  $M$  Teile unterteilt, wobei  $M$  ein Eingabeparameter ist. Nach der Verschmelzung der  $M$  Teile werden die Daten schlussendlich in  $K$  Gruppen verteilt. Der Parameter  $K$  ist vordefiniert und zeigt die vom Benutzer gewünschte Anzahl der Gruppen [Karypis et.al 1999]. Bei BIRCH oder CHAMELEON muss man die Anzahl der Cluster als Eingabe festlegen. Manchmal ist jedoch diese Zahl nicht bekannt oder schwer vorherzusagen. Der DBSCAN-Algorithmus kann dieses Problem vermeiden, was von Ester et.al in Jahr 1996 veröffentlicht wurde [Ester et.al 1996]. Statt der gewünschten Anzahl der Cluster sollen ein Bereich  $\epsilon$  und die mindeste Anzahl der Nachbarn in diesem Bereich, was die sogenannte Grenzdichte beschreibt, vordefiniert werden. Das Objekt, das nicht genug Nachbarn im anforderten Bereich hat, wird als Rauschen markiert. Die bleibende Objekte werden im gleichen Cluster zusammengefasst, wenn sie nicht weiter als  $\epsilon$  von mindesten  $N$  von ihren Nachbarn liegen, wobei  $N$  ein Eingabeparameter ist. In der Arbeit von Josiger und Kirchner werden die Ergebnisse der drei Algorithmen von verschiedenen Testdaten verglichen. Entweder für die Punktmenge einfacher Gestalt oder die Punktmenge nicht-sphärischer Gestalt können das DBSCAN und CHAMELEON Algorithmen die zufriedene Lösung liefern. In der Situation, wo die Testdaten verrauscht sind, kann der DBSCAN Algorithmus mit geeignetem Parameter auch ein zufriedenes Ergebnis ausgeben. Das Ergebnis bleibt stabil, obwohl der Anteil des Rauschens stark einsteigt.

#### 2.3.4.2 Dynamisches Verfahren

Das Problem der Kalibrierung der Objekte ist ähnlich zum Problem der Segmentierung der Bewegungen in einer langen Bildsequenz. Eine traditionelle Lösungsstrategie ist, die Positionen der Marken im nächsten Zeitpunkt mit Kalman Filter vorherzusagen. Dann werden alle Marken anhand der kinematischen Parameter in verschiedene Gruppen eingeteilt. Mills und Novins haben eine andere Möglichkeit geliefert, damit die Objekte direkt mit 2D Graphen kalibriert werden können [Mills & Novins 2000]. Am Anfang

ihres Algorithmus wird jede Marke mit einander verbunden. Dann werden alle Marken und die Kanten dazwischen zusammen als ein vollständiger Graph dargestellt. Zwischen den Bewegungen der Objekte verändert sich die Länge der Kanten. Die Kanten, die länger als eingesetzte Beschränkung sind, werden aus dem Graphen Schritt um Schritt gelöscht. Eine Marke gehört einem Objekt, genau dann wenn das Dreieck, das von dieser Marke und anderen Marken in diesem Objekt erzeugt wird, mindestens eine gleiche Kante mit den anderen Dreiecken des Objekts hat. Am Ende des Algorithmus wird der ursprüngliche vollständige Graph in viele Teilgraphen zerlegt, die genau den kalibrierten Objekten entsprechen. Es gibt jedoch die Einschränkung in dem Algorithmus von Mills und Novins, dass zwei Objekte nicht auseinanderzuhalten sind, wenn ihre Marken mit einigen besonderen Strukturen angebracht werden [Rhijn & Mulder 2005]. Rhijn und Mulder haben dieses Problem gelöst, indem sie die Voraussetzung des Algorithmus veränderten. In der neuen Voraussetzung darf die Marke in einem Objekt erkannt werden, nur wenn sie mit anderen drei Marken in dem Objekt zusammen eine Pyramide erzeugen kann. Die Pyramide hier bezieht sich auf einen Körper der Geometrie, der vier Knoten hat und jede zwei davon eine Kanten erzeugen. Die neue stärkere Beschränkung erhöht die Erfolgsquote deutlich.

### 2.3.5 Objekterkennung und Verfolgung

Das Ziel dieser Arbeit ist, dass ein Objekt nach Initialisierung von dem System wieder erkannt werden kann. Eine Wiedererkennung des 3D Objekts durch 2D Bildfolgen wird von Lamdan et al. in ihrer Arbeit erfolgreich durchgeführt [Lamdan et al. 1988]. Sie haben einige interessante Punkte ausgewählt, um das totale Objekt zu beschreiben. Alle drei Punkte, die nicht in gleicher Gerade liegen, definieren ein Koordinatensystem, auf dem die entsprechenden Koordinaten von anderen Punkten berechnet und in einem HashMap gespeichert werden. Die richtige Korrespondenz wird so bestimmt, dass das kleinste Quadrate Modell der Transformation zwischen dem neuen Koordinatensystem und 2D Bild die beste Lösung liefert.

Andererseits kann die charakteristische Information jedes Objekts einen einzigen vollständigen Graphen erzeugen. Alle sichtbaren Marken des erkannten Objektes können als ein Teilgraph des vollständigen Graphen definiert werden, wodurch das Problem der Objekterkennung bzw. Objektverfolgung als das Teilgraph Isomorphismus Problem abgeleitet werden kann. Es gibt eine große Menge an Algorithmen, um das Problem zu behandeln, da das Isomorphismus Problem nicht nur im Bereich der Bildanalyse, sondern auch im Vergleich der Struktur von chemischen Verbindungen oder in biometrischer Identifikation betrachtet wird. Conte et al. haben eine Zusammenfassung über diese Algorithmen veröffentlicht [Conte et al. 2004]. Durch ihre Taxonomie können alle diese Verfahren in zwei Gruppen von genauen bzw. ungenauen Graph Matching Algorithmen unterteilt werden.

In einem genauen Graph Matching wird die strenge Korrespondenz zwischen zwei Graphen bestimmt. Die Abbildung von einem Graphen zu einem anderen soll bijektiv sein. Ullmann hat ein rekursives Rücksetzverfahren beschrieben, das sehr bekannt ist und bis heute für genaues Matching häufig benutzt wird [Ullmann 1976]. Was von Rhijn und Mulder in ihrer Arbeit für die Objekterkennung implementiert wurde, ist auch ein genaues Graph Matching Verfahren [Rhijn & Mulder 2005]. Sie folgen der Idee von Lamdan, aber verbessern das Verfahren mit einer Beschränkung für die Größe des Teilgraphs, die ausreichend für die Unterscheidung von zwei Objekten ist. Nach der Verbesserung ist der neue Algorithmus viel effizienter und stabiler. Cordella et al. haben einen Algorithmus mit Namen VF2 für das Graph und Teilgraph Isomorphismus Problem in großen Graphen entwickelt [Cordella et al. 2004]. In dem Vorgang des Matching haben sie einige Regeln definiert, wodurch die Komplexität des Rechnens stark reduziert wird. Eppstein konzentriert auf das Teilgraph Isomorphismus Problem von planaren Graphen [Eppstein 1999]. Der Graph wird in viele kleine Bäume unterteilt. Auf diesen wird mittels dynamischer Programmierung das Matching in linearer Zeit durchgeführt.

Das genaue Graph Matching ist manchmal ungeeignet, beispielsweise für nicht komplett fest definierte Graphen, was z.B. bei Rauschen oder instabilen Komponenten vorkommt. Wegen des Unterschieds zwischen dem beobachteten Modell und idealen Modell, soll das Matchingsverfahren tolerant sein. Dadurch kann eine Korrespondenz zwischen zwei Graphen gefunden werden, obwohl es keine strenge Transformation dazwischen gibt. Außerdem benötigt das genaue Graph Matching Verfahren exponentielle Laufzeit im Worst-Case, die durch eine Approximation in ungenauen Matchingsverfahren stark reduziert werden kann. Messmer und Bunke haben ein Fehler-tolerantes Verfahren für Teilgraph Isomorphismus mit unbekanntem Graph als Eingabe entwickelt [Messmer & Bunke 1998]. Die Modellgraphen werden durch eine Vorverarbeitung in kleine Teilgraphen unterteilt. Alle diese Teilgraphen werden so zusammengefasst, dass die oftmals vorkommenden Teilgraphen nur ein mal repräsentiert werden. Der eingebogene Graph wird mit diesen verdichteten Graphen verglichen. Dadurch hängt die Laufzeit nur von der Anzahl der Modellgraphen ab.

# Kapitel 3

## Grundlagen

### 3.1 Generierung der 3D-Daten

#### 3.1.1 TOF-Sensor

##### 3.1.1.1 TOF Kamera

Die im MAROCO-System verwendete Kamera gehört zur Klasse der TOF-Sensoren, die außer den normalen Graufarbenbildern auch Tiefbilder liefern kann. Die Tiefmessung basiert auf dem sogenannten Laufzeitverfahren. Dazu wird die Szene durch ein Lichtpuls ausgeleuchtet und für jeden Bildpunkt wird die Zeit gemessen, die das Licht bis zum Objekt und wieder zurück braucht. Die Distanz ist direkt proportional zu der benötigten Zeit und kann durch die folgende Formel berechnet werden:

$$d = \frac{t_d}{2c} \quad (3.1)$$

wobei  $t_d$  die gemessene Zeit bezeichnet. Die Konstante  $c$  steht für die Lichtgeschwindigkeit.

Im Vergleich zu anderen 3D Kamerasytemen hat die TOF Kamera viele Vorteilen [[TOF-Kamera Wikipedia](#)]. Zuerst kann die TOF Kamera einfach die interessierenden Bereiche aus einem Bild extrahieren und nur die Pixel nah vor der Kamera betrachten. Zweitens, kann die TOF Kamera eine hohe Bildrate bis zu 80 bps erreichen. Diese Eigenschaft ermöglicht somit Echtzeitanwendungen. Außerdem benötigt die TOF Kamera weniger Platz als z.B. das Triangulationssystem und hat niedrigere Abhängigkeit von der Systemstruktur gegenüber dem Stereosystem.

Parameter	Value *	Notes
Type of Sensor	PhotonICs® PMD 41k-S (204 x 204)	With SBI (Suppression of Background Illumination)
Measurement Range	0.3 to 7 m	
Repeatability ( $1\sigma$ )	< 3 mm	Typical value, central sensor area @2 m distance, 90% reflectivity
Frame Rate (3D)	25 fps	Typical value, depending on camera settings
Field of View	40° x 40°	CS mount lens: f = 12,8 mm; F# = 1,4
Illumination Wavelength	870 nm	Eye safety class 1
Power Supply [V]	12V ± 10%	
Interface	USB2.0	
Operating Temperature	0°C to 50°C	
Storage Temperature	- 20°C to 85°C	

Abbildung 3.1: Die grundlegende Parameter der PMD CamCube2. [[PMD CamCube Einführung](#)]

### 3.1.1.2 PMD Sensor

Der PMD Sensor heißt CamCube ist eine wichtige Komponente der TOF Kamera. Er liefert eine hohe Auflösung bis zum 204x204 Pixel und maximale Bildrate bis zum 40 bps. Durch Formel (3.1) wird der maximale Distanzbereich zum 7 m festgelegt. Die anderen wichtigen Parameter findet man in Abbildung 3.1 [[PMD CamCube Einführung](#)].

Das Hintergrundlicht, z.B. das Sonnenlicht, könnte die Messung der Distanz stark stören. Die PMD Kamera benutzt das aktive Sendersignal und einen Fremdlicht-Filter (SBI), um das Hintergrundsignal zu unterdrücken. Außerdem bietet die PMD Kamera die Möglichkeit, die Integrationszeit der Kamera für jede Messung individuell einzustellen. Die Integrationszeit bezieht sich auf die Zeitspanne, in der die Kamera zur Aufzeichnung eines Bildes dem reflektierten Licht ausgesetzt wird. Für ein schwach reflektierendes Objekt benötigt der Sensor längere Integrationszeit als ein stark reflektierendes Objekt, um genug Information anzusammeln. Andererseits wird aber ausreichendes Licht von hellen Objekten auf den Sensor reflektiert, wenn die Integrationszeit zu lang definiert wird. In Abbildung 3.2 wird ein Beispiel der Tiefbilder mit verschiedenen Integrationszeiten gezeigt [[PMD CamCube Entwicklungstutor](#)].

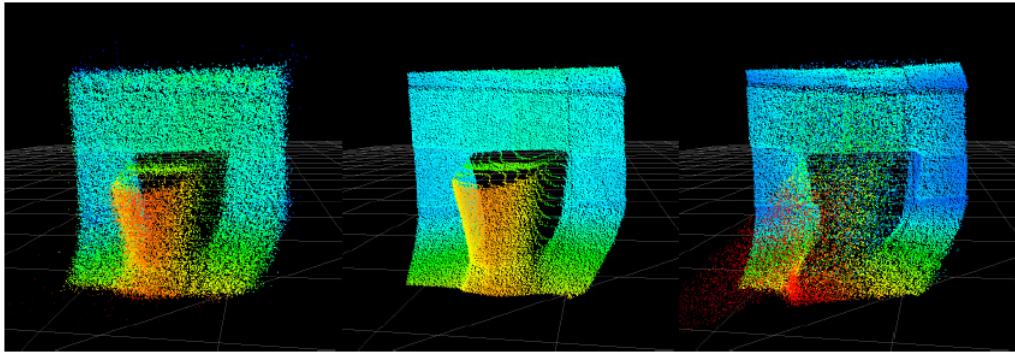


Abbildung 3.2: Integrationszeit von  $140 \mu\text{s}$ ,  $1400 \mu\text{s}$  bzw.  $14000 \mu\text{s}$ . Bitte beachten Sie die niedrige Signalstärke an der linken Seite und die Sättigung an der rechten Seite wegen der unangemessenen Integrationszeit. [PMD CamCube Entwicklungstutor]

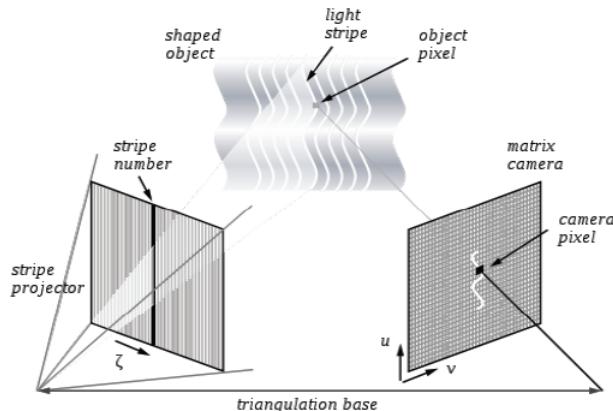


Abbildung 3.3: Das Arbeitsprinzip des Kinects. [Kinect-How it works]

### 3.1.1.3 Unterschied zwischen TOF Kamera und Kinect

Kinect ist eine Hardware zur Steuerung der Videospielkonsole Xbox360, die ein sogenanntes hands-free Kontrollieren liefert, wodurch die Spieler mit einigen bestimmten Gesten oder einer kurzen Bewegung ihres Körpers das Spiel spielen können [Kinect]. Um dieses Ziel zu erreichen, sammelt Kinect außer der normalen Bildeingabe aber auch die Tiefdaten der Szene an. Wegen dieser Eigenschaft wird das Gerät im Bereich von Computer Vision benutzt. Mithilfe des SDK ist die Programmierung der Kinect unter normalen Betriebssystemen z.B. Windows, Linux bzw. MacOS möglich.

Der Unterschied zwischen TOF Kamera und Kinect können auf dem Arbeitsprinzip zurückgeführt werden. In TOF Kameras wird die Tiefdaten durch dem Laufzeitver-

Sensor	PMD CamCube	Kinect
Auflösung	204×204 Pixel	640×480 Pixel
Sichtfeld	40° × 40°	57° × 43°
Max Bildrate	40 fps	30 fps
Messbereich	0.3 → 7.0 m	1.2 → 3.5 m (mit Xbox Software)
Länge der Tiefdaten	8 bit (unsigned char)	11 bit

Tabelle 3.1: Die technische Daten von PMD Kamera und Kinect

fahren berechnet, was im 3.1.1.1 erklärt wird. Das Abtastverfahren der Kinect heißt Light Coding. Eine große Menge von Streifen werden als Mustern auf die Szene bzw. die Objekte durch infrarotes Licht projiziert. Die ganz Szene mit diesen zusätzlichen Mustern wird von einer infraroten Kamera des Kinects aufgenommen. Durch die Verzerrung zwischen dem vordefinierten Muster im infraroten Licht und dem von der infraroten Kamera erkannten Muster kann das Tiefbild der Szene ausgerechnet werden. Die Abbildung 3.3 zeigt dieses Arbeitsprinzip der Kinect. Die weitere Information findet man im technischen Dokument des Firma Cadet [[Kinect-How it works](#)]. Der Vergleich über die genauen technischen Daten von PMD Kamera und Kinect wird in Tabelle 3.1 zusammengefasst. Obwohl Kinect eine bessere Auflösung und größeres Sichtfeld hat, ist die PMD Kamera wegen ihrer hohen Bildrate und großen Messbereich für das MAROCO-System geeignet. Außerdem mithilfe des SBI Systems ist die Arbeit der PMD Kamera unter schwieriger Umgebungsbedingung, z.B. außerhalb des Zimmers mit starker Störung von Sonneneinstrahlung, auch möglich.

### 3.1.2 Daten der PMD Kamera

Die PMD Kamera CamCube kann insgesamt 4 verschiedenen Vermessungsdaten ausgeben. Sie sind Amplitude, Intensität, Distanz und 3D Koordinaten. Die erste Zwei und letzte Zwei Typen der Daten können durch die Dimension in zwei Gruppen unterteilt werden.

#### 3.1.2.1 2D Daten

Die Intensität bezieht sich auf Graustufen. Nach einer Abbildung können diese Graustufen auf das Intervall 0 bis 255 beschränkt werden und das Ergebnis ist das Bild einer normalen Schwarz-Weiß Kamera. Die Amplitude zeigt die Stärke der Beleuchtung, die vom Objekt wegen des aktiven Sendersignal von PMD Kamera selbst reflektiert wird. Dieser Wert kann die Qualität der Distanzinformation schätzen, d.h. das Objekt weit von Kamera liefert niedriger Amplitude als das Objekt in der Nähe von der Kamera,

wenn sie mit identischem Material dargestellt werden. An der Gegenseite sind die Merkmale mit höherem Rückstrahlvermögen durch Amplitudedaten einfacher betrachtet, was für die Erkennung der großen künstlichen Marken in unserer Arbeit sinnvoll ist.

### 3.1.2.2 3D Daten

Die PMD Kamera liefert 3D Daten in zwei Formen: die reine Distanzinformation und die 3D Koordinaten. Die Distanzinformation ist die gemessene Distanz zwischen der Kamera und Objekt, was direkt durch die Formel (3.1) berechnet wird. Bezuglich dieser Distanzinformation und der 2D Daten der normalen Kamera werden die 3D Koordinaten innerhalb der PMD Kamera berechnet und können durch die Schnittstelle abgefragt werden. Die Abbildung 3.5 zeigt die Visualisierung einer Szene mit jeweils der 3D Koordinaten und Distanzinformation von PMD Kamera. Eine Transformation ist notwendig, wenn man direkt die Distanzinformation im kartesischen Koordinatensystem visualisieren möchte.

### 3.1.2.3 Auflösung und erkennbare Markengröße

Die Auflösung der PMD Kamera wurde schon im vorherigen Abschnitt angegeben, wofür man sich noch interessiert ist, wie genau die Objekte von der Kamera in der Praxis beobachtet werden können. D.h. wie groß ein Pixel von Kamera den Bereich in realer Welt beschreibt. Die Vorderansicht des Kamerasytems dieser Arbeit ist in Abbildung 3.4 links gezeigt.

$\theta$  ist der halbe Sichtwinkel und wird hier als  $20^\circ$  angenommen.  $H$  zeigt den Abstand von der Kamera zum Boden, was in diesem System mit 3,2m festgelegt ist. Durch die Trigonometrie kann der Radius des Sichtbereiches auf dem Boden berechnet werden als:

$$R = \tan \theta \cdot H = 0,36397 \cdot 3,2m = 1,16470m.$$

Die halbe Seitenlänge des Sehnenquadrats  $L$  (Sehen Abbildung 3.4 rechts) ist gleich:

$$L = \cos 45^\circ \cdot R = 0,70711 \cdot 1,16470m = 0,82357m.$$

Dann kann man der Flächeninhalt des Sichtbereiches auf dem Boden erhalten:

$$S = 4 \cdot L^2 = 2,7131m^2.$$

Das Ergebnis der Division von dem Flächeninhalt und der Auflösung beschreibt die Größe der Zelle auf dem Boden, die in Kamera als eigenes Pixel dargestellt wird.

$$S_z = S / (204 \times 204) = 2,7131m^2 / 41616 = 6,5194 \times 10^{-5}m^2 = 0,65194cm^2$$

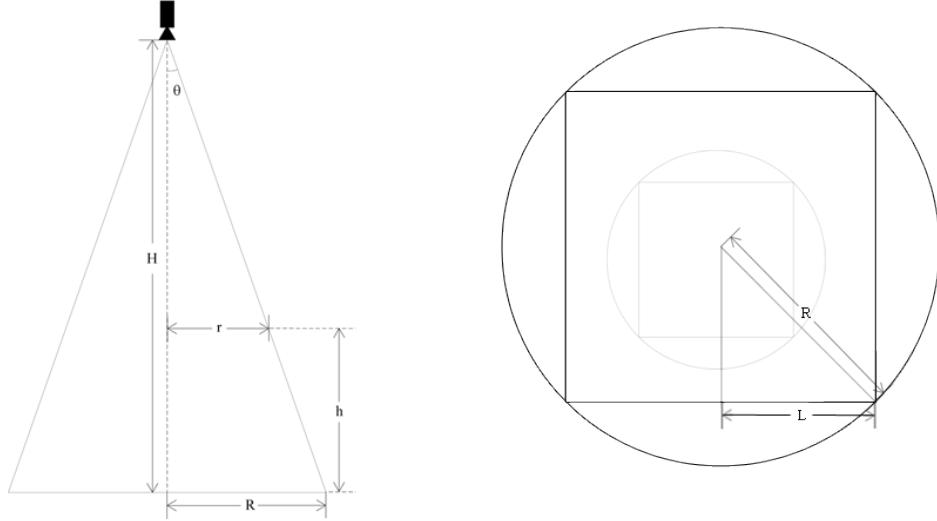


Abbildung 3.4: Die Vorderansicht(links) und die Draufsicht(rechts) des PMD Kamera Systems

Die Marken, die kleiner als  $S_z$  sind, werden in der Kamera kleiner als ein Pixel abgebildet und sind natürlich schwer zu erkennen. D.h.  $S_z$  definiert die minimale Größe der erkennbaren Marken. Die minimale Seitenlänge kann nun berechnet werden durch:

$$L_z = \sqrt{S_z} = 0,80743\text{cm}$$

Die minimalen Größen der Marken für anderen zum Boden parallelen Ebenen können analog berechnet. Z.B. in der normalen Arbeitsebene dieser Arbeit, die zum Boden 1, 1m entfernt, können die mindesten Quadrate mit Seitenlänge von 0, 52987cm erkannt werden.

## 3.2 Vorverarbeitung

### 3.2.1 Schwellwert-basierte Segmentierung

Um bessere Erkennungsergebnisse zu erhalten, sollen die wesentlichen Bereiche von der Umgebung getrennt werden. In dieser Arbeit wird eine Schwellwert-basierte Segmentierung bezüglich der 3D Daten verwendet. Eine Schwellwert-basierte Segmentierung kann als eine Abbildung  $f$  vom originalen Bild  $I$  zum Ergebnisbild  $H$  definiert werden:

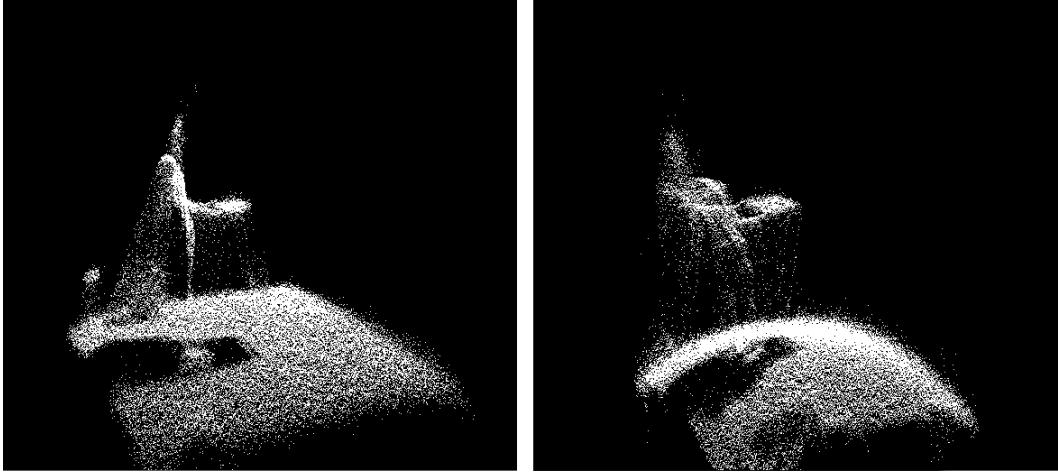


Abbildung 3.5: Die Visualisierung von 3D Koordinaten(link) und Distanzinformation(recht).

$$f : I \mapsto H$$

mit

$$H_{ij} = \begin{cases} 1 & \text{für } I_{ij} > \Theta \\ 0 & \text{sonst} \end{cases} \quad (3.2)$$

wobei  $\Theta$  der eingegebene Schwellwert ist.

### 3.2.2 Steuerung der Helligkeit und Kontrast

Außer der Umgebung beeinflussen die Helligkeit und der Kontrast die Qualität bzw. Stabilität der Markenerkennung. Deshalb ist die Optimierung dieser zwei Parameter in der Vorverarbeitungsphase notwendig. In dieser Arbeit wird ein affiner Operator auf jedem Punkt durchgeführt, um die geeignete Helligkeit bzw. den Kontrast zu bestimmen. Der affine Operator ist eine Abbildung von Originalbild  $I$  zum Ergebnisbild  $H$  mit:

$$H_{ij} = aI_{ij} + b, \quad (3.3)$$

wobei die Parameter  $a \in R^+$  und  $b \in R$  die Helligkeit und den Kontrast kontrollieren. Es gibt vier Möglichkeiten für die verschiedenen Zuordnungen dieser Parameter:

- $a > 1, b = 0$  Kontrasterhöhung



Abbildung 3.6: Die vier Beispielbilder. Von link nach recht sind Barbara, Lena, Peppers und Mandril. [Odessa 2011]

- $0 < a < 1, b = 0$  Kontrastminderung
- $a = 1, b > 0$  Helligkeitserhöhung
- $a = 1, b < 0$  Helligkeitsminderung

## 3.3 Markenerkennung

Wie im Abschnitt 2.3.1 beschrieben, sind viele Erkennungsalgorithmen in der Open Source Library OpenCV realisiert. Wegen verbreiteter Benutzung von OpenCV, werden häufig Vergleiche dieser Algorithmen gemacht. Im folgenden Abschnitt wird auf einen Vergleich eingegangen, um den geeigneten Algorithmus für diese Arbeit auszuwählen.

### 3.3.1 Auswahl des Erkennungsalgorithmus

Odessa hat in seinem Blog einen sehr guten Vergleich für alle Erkennungsalgorithmen von OpenCV durchgeführt [Odessa 2011]. Vier häufig benutzte Beispielbilder wurden betrachtet (s. Abb. 3.6).

Was besonders interessiert in seiner Arbeit, ist der Vergleich über die Anzahl der betrachteten Punkte bzw. der durchschnittlichen Fehler. Wegen der verschiedenen Prinzipien erkennt der Algorithmus FAST viel mehr Merkmale als SURF und STAR (Name von CenSurE Algorithmus in OpenCV). Den Unterschied sieht man deutlich in der Abbildung 3.7. Je mehr Punkte betrachtet werden, desto mehr Rauschen wird in das System gebracht, weil viele normale Pixel auch als Merkmale erkannt werden. Das ist offensichtlich ein negativer Einfluss für die weitere Analyse der Daten. Abbildung 3.8 zeigt den durchschnittlichen Fehler in Pixeln von den Punktpaaren, was durch gleichen Erkennungsalgorithmus von Bezugsbildern erkannt wird. Der Algorithmus STAR erzeugt deutlich weniger Fehler in der Erkennung, und das Ergebnis hängt auch leicht von der Eingabe ab. Wegen der niedrigeren Fehlerquote und besserer Konzentration

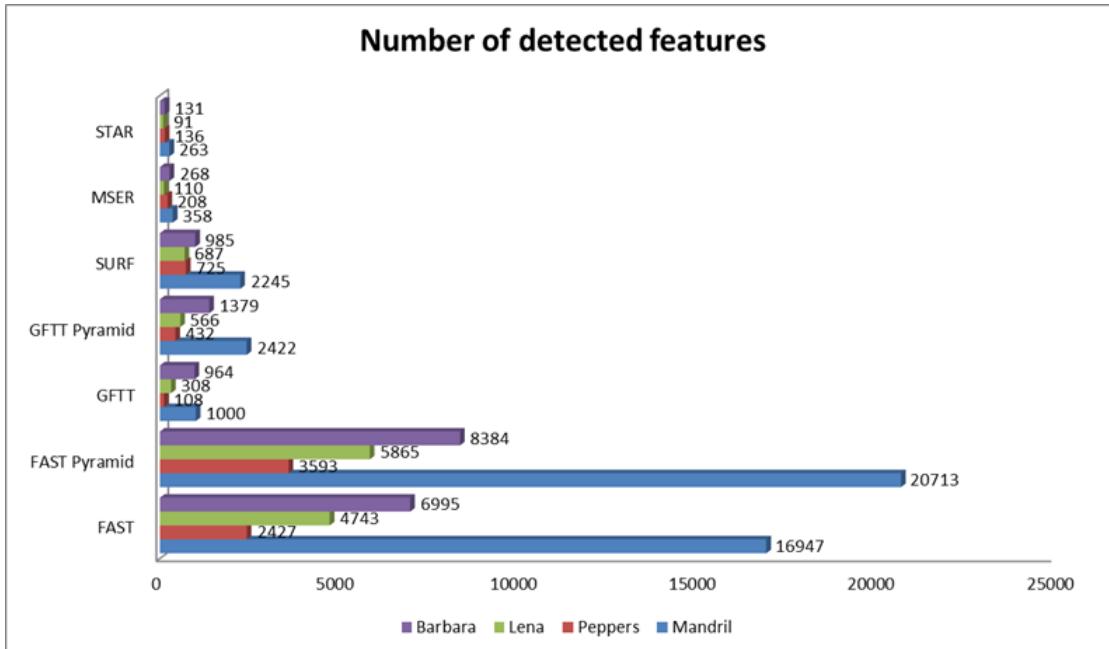


Abbildung 3.7: Anzahl der erkannten Merkmale von alle vier Beispielbildern durch verschiedene Erkennungsalgorithmen. [Odessa 2011]

an großen kreisförmigen Merkmalen ist der STAR Algorithmus für die Erkennung der Objekte mit künstlichen Marken sehr geeignet.

### 3.3.2 CenSurE Algorithmus

Der Algorithmus CenSurE (Center Surround Extrema) wird von Agrawal et al. 2008 entwickelt [Agrawal, Konolige & Blas 2008]. Sie verbessern die SIFT bzw. SURF Verfahren durch Berücksichtigung aller Merkmale in allen Skalenräumen. Das Extremum durch die Skalen und Lagen werden ausgewählt, um die Merkmale zu bestimmen. Der Bi-level Filter wird hier statt Gaussian Filter verwendet, damit der Algorithmus in Echtzeit laufen kann, obwohl große Menge der Berechnung für alle Merkmale aller Skalenräume nötig sind.

#### 3.3.2.1 Bi-level Filter

Der Bi-level Filter ist eine einfache Approximation des Laplacian-Operators durch die Multiplikation der Bilder mit 1 und -1. Die Abbildung 3.9 zeigt die Progression des Bi-level Filters mit verschiedenen Symmetrischen Stufen. Der kreisförmige Filter an linker Seite der Abbildung 3.9 kann den Laplacian-Operator am Beste approximieren, ist aber leider schwierig zu implementieren. Deshalb werden die Progressionen der Filter durch

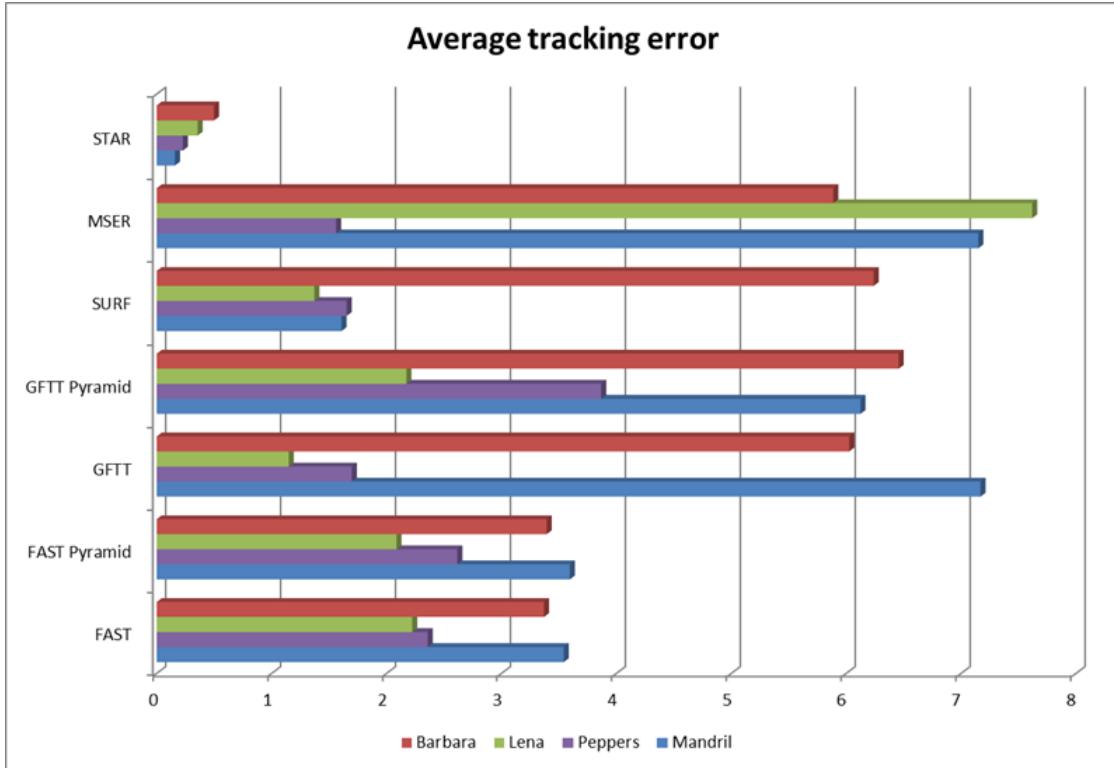


Abbildung 3.8: Der durchschnittliche Fehler (in Pixeln) zwischen den assoziierten Punkten von zweier folgendes Bilder. [Odessa 2011]

Polygone angenähert, damit die Berechnung vereinfacht werden kann. Zum Vergleich der übrigen Formen der Abbildung 3.9 liefert der Filter mit Achteck die beste Leistung und der Filter mit Rechtecke die kürzeste Laufzeit. Diese Polygon-Filter können durch die integralen Bilder einfach dargestellt werden.

### 3.3.2.2 Integrale Bilder

Ein integrales Bild  $I$  ist eine mittlere Repräsentation eines Bildes, was die Summe der Grauwerte von Bild  $N$  mit Breite  $x$  und Höhe  $y$  enthält.

$$I(x, y) = \sum_{x'=0}^x \sum_{y'=0}^y N(x', y') \quad (3.4)$$

Das integrale Bild ist rekursiv berechenbar und benötigt nur einmal Durchführung aller Pixeln des Bildes. Mithilfe des integralen Bildes kann die Intensität des beliebigen rechteckigen Bereiches einfach durch vier Additionen berechnet werden. Die Erweiterung des integralen Bildes wird für die Berechnung der Polygone Filter benutzt. Die

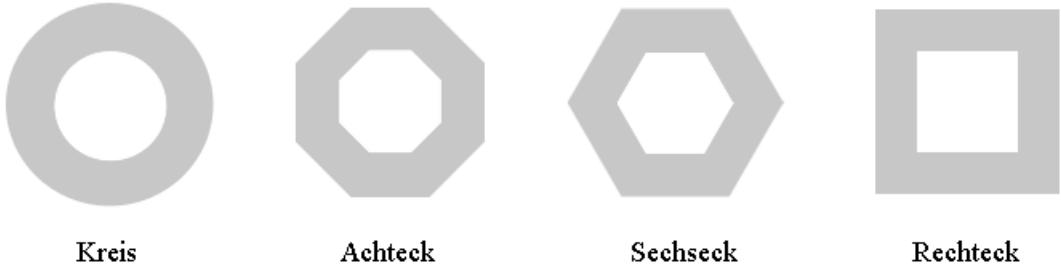


Abbildung 3.9: Progression der Center-Surround Bi-level Filter. Der Kreis ist die ideale voll symmetrische Approximation der Laplacian. Die Filter daneben von links nach rechts haben niedriger Symmetrie, brauchen aber weniger Zeit zur Berechnung. [Agrawal, Konolige & Blas 2008]

Kombination zweier verschiedenen schrägen integralen Bilder kann den für Polygone benötigten trapezförmigen Bereich einfach darstellen. Die mathematische Beschreibung des schrägen integralen Bildes ist:

$$I_\alpha(x, y) = \sum_{y'=0}^y \sum_{x'=0}^{x+\alpha(y-y')} N(x', y'), \quad (3.5)$$

wobei  $\alpha$  den schrägen Winkel erklärt und wenn es 0 gleich, ist die Formel (3.4) genau so wie die Formel (3.5), und beschreibt ein rechteckiges integrales Bild. Die in der Abbildung 3.9 gezeigte Achtecke-Filter und Sechsecke-Filter können mit jeweils 3 bzw. 2 Trapezen schnell aufgebaut werden.

### 3.3.2.3 Non-maximal Suppression

Non-maximal Suppression ist eine Strategie, die das lokale Extremum finden kann. Die Response des Pixels wird unterdrückt, wenn es ein Pixel in seiner Nachbarschaft für Lage bzw. Skala gibt, dessen Response größer oder kleiner als das betrachtende Pixel ist. Die Pixel mit entweder Maximum oder Minimum Response werden als Merkmale bekannt. Der Suchumfang wird in der Arbeit von Agrawal als 3x3x3 eingestellt, d.h. 8 Pixeln um dem betrachteten Pixel und jeweils 9 Pixeln im zwei benachbarten Skalenräumen, werden zusammen berücksichtigt. Die Pixel mit höherer Response können zwischen der Transformation des Bildes stabiler wieder erkannt werden, deshalb nach der Non-maximal Suppression werden die ausgewählten Extremums noch mal durch einem Schwellwerte-Filter gefiltert, um die besten Merkmale zu bestimmen.

### 3.3.3 Kalman-Filter

Das Kalman-Filter basiert auf einem linearen, dynamischen System in einem diskretisierten Zeitraum. Die Zustandsgleichung des Systems wird häufig durch eine Differenzengleichung beschrieben. In vielen Fällen werden die Zustände nur durch einen voneinander getrennten Zeitpunkt bestimmt. Kalman hat den Sonderfall der linearen Abhängigkeit der Zustände untereinander betrachtet, und vereinfachte die Zustandsgleichung zur linearen Differenzengleichung.

#### 3.3.3.1 Zustandsraummodellierung

Der nächste Systemzustand kann basierend auf dem aktuellen Systemzustand durch:

$$X_k = F_{k-1}X_{k-1} + B_{k-1}u_{k-1} + w_{k-1} \quad (3.6)$$

schätzen. Der Index  $k$  bzw.  $k - 1$  beziehen sich auf den Zeitpunkt  $t_k$  und  $t_{k-1}$ , wobei  $t_k = t_0 + k\Delta t$  und  $t_0$  der Anfangszeitpunkt und  $k$  eine natürliche Zahl von Interesse ist. Deshalb beschreibt der mehrdimensionale Vektor  $X_k$  den Zustand des Systems zur Zeitpunkt  $t_k$ . Die Matrix  $F_{k-1}$  ist die Übergangsmatrix für die zeitlich aufeinanderfolgenden Zustände  $X_k$  und  $X_{k-1}$ . Die Matrix  $B_{k-1}$  und Kontrollvektor  $u_{k-1}$  stellen den deterministischen Anteil der weiteren äußeren Einflüsse auf das System dar. Die zufälligen, nicht erfassbaren Komponenten der äußeren Einflüssen werden durch die stochastische Größe  $w_{k-1}$  geschätzt, die einer Normalverteilung mit Mittelwert 0 und Kovarianz  $Q_{k-1}$  folgt.

$$w_{k-1} \sim N(0, Q_{k-1})$$

Wegen dieser Zufallsvariable bilden die Menge aller Zustandsvektoren eine Markov-Kette, d.h. der Zustand zu einem Zeitpunkt  $k$  hängt lediglich vom unmittelbaren zeitlichen Vorgänger an  $k - 1$  ab.

Die Beobachtungen des Systems werden aus modellierbarer Verzerrung und unvorhersagbarem Messrauschen bestanden:

$$Z_k = H_k X_k + v_k. \quad (3.7)$$

$Z_k$  bezieht sich auf die Messung zum Zeitpunkt  $k$ . Die Multiplikation von der Beobachtungsmatrix  $H_k$  und Zustandsvektor  $X_k$  beschreibt die lineare Approximation der Verzerrung des Systems. Das Rauschen  $v_k$  wird im Kalman-Filter als zeitlich unkorreliert und normalverteilt angenommen:

$$v_k \sim N(0, R_k).$$

### 3.3.3.2 Das Kalman-Filter

Das Ziel eines Filters ist, durch die Informationen einer Messreihe die Zustände besser schätzen zu können. Da die Rauschterme  $w$  und  $v$  für alle Zeit die Normalverteilung erfüllen, können die zeitdiskreten Zustände  $X_k$  auch durch eine Normalverteilung mit dem Mittelwert  $\hat{x}_k$  und der Kovarianz  $\hat{P}_k$  ermessen werden.

$$\hat{X}_k \sim N(\hat{x}_k, \hat{P}_k)$$

Die Idee des Kalman-Filter ist, eine rekursive Formulierung aufzubauen, die aber nur die Schätzung eines vorherigen Zeitpunkts und die aktuelle Messung benötigt, um die Schätzung des aktuellen Zeitpunkt zu bestimmen. Es gibt hauptsächlich zwei Phasen im Kalman-Filter.

#### Prädiktion

In dem ersten Schritt dieser Phase wird eine vorangegangene Schätzung  $X_{k|k-1}$  für den aktuellen Zeitpunkt vorausgesagt.

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1} + B_{k-1}u_{k-1} \quad (3.8)$$

Die Indizierungsschreibweise  $k|k-1$  bezieht sich auf die Bedingtheit zu den Zeitpunkten  $k$  und  $k-1$ . Für die Kovarianz gilt:

$$\hat{P}_{k|k-1} = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + Q_{k-1}. \quad (3.9)$$

#### Korrektur

Die Vorhersagen von letztem Schritt werden hier durch die neue Messung korrigiert:

$$\hat{x}_k = \hat{x}_{k|k-1} + \hat{K}_k \tilde{y}_k, \quad (3.10)$$

$$\hat{P}_k = \hat{P}_{k|k-1} - \hat{K}_k S_k \hat{K}_k^T. \quad (3.11)$$

Die Hilfsgröße der Innovation  $\tilde{y}_k$  beschreibt, wie genau die aktuellen Messungen von der vorhergesagten Schätzungen mithilfe der Beobachtungsgleichung approximiert werden:

$$\tilde{y}_k = Z_k - H_k \hat{x}_{k|k-1}.$$

$S_k$  bezieht sich auf die Residualkovarianz, wobei gilt:

$$S_k = H_k \hat{P}_{k|k-1} H_k^T + R_k$$

und  $\hat{K}_k$  ist die zugehörige Kalman-Matrix:

$$\hat{K}_k = \hat{P}_{k|k-1} H_k^T S_k^{-1}.$$

## 3.4 Registrierung

### 3.4.1 Singulärwertzerlegung

Sei  $M$  eine komplexe  $m \times n$  Matrix mit Rang  $r$ . Dann bezeichnet die Singulärwertzerlegung das Produkt:

$$M = U\Sigma V^* \quad (3.12)$$

wobei  $U$  eine Unitäre Matrix mit Größe  $m \times m$  und  $V^*$  die Adjungierte einer Unitären Matrix mit Größe  $n \times n$  ist.  $\Sigma$  bezieht sich auf eine  $m \times n$  Diagonalmatrix:

$$\Sigma = \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_r & & & \\ \hline & & & \vdots & & \\ & & & & \ddots & \\ & & & & & \vdots \end{pmatrix}$$

mit  $\sigma_1 \geq \dots \geq \sigma_r > 0$ , wobei  $\sigma_i, i = 1, \dots, r$  als die Singulärwerte von  $\Sigma$  genannt werden.

### 3.4.2 Korrespondenzuntersuchung durch Singulärwertzerlegung

Mithilfe der Singulärwertzerlegung haben Scott und Longuet-Higgins einen Algorithmus zur Bestimmung der assoziierenden Merkmale entwickelt [Scott & Longuet-Higgins 1991]. Seien  $I$  und  $J$  zwei nachfolgende Bilder und haben jeweils  $m$  und  $n$  Merkmale, die als  $I_i (i = 1, \dots, m)$  und  $J_j (j = 1, \dots, n)$  bezeichnet werden. Dann wird eine  $m \times n$  Matrix  $G$  mit den Elementen

$$G_{ij} = \exp\left(-\frac{r_{ij}^2}{2\sigma^2}\right)$$

definiert, wobei  $r_{ij}$  den Abstand zwischen Merkmale  $I_i$  und  $J_j$  beschreibt.  $\sigma$  wird als einen Standard für Abstand definiert, wodurch das Vergrößern oder Verkleinern der Verschiebung des Objekts geschätzt werden kann. Der Wert von  $G_{ij}$  nimmt durch die

Erhöhung der Distanz von 1 bis 0 monoton ab. Der zweite Schritt des Algorithmus von Scott und Longuet-Higgins ist die Singulärwertzerlegung der Matrix  $G$ :

$$G = TDU.$$

wobei  $T$  und  $U$  die unitäre Matrix mit jeweils Größe  $m \times m$  und  $n \times n$  sind, und  $D$  eine Diagonalmatrix ist. Sei  $E$  eine neue Matrix mit gleicher Größe von  $D$ , in der aber jedes diagonale Element als 1 ersetzt wird. Nach Austausch der Matrix  $D$  durch Matrix  $E$  erhält man eine neue orthogonale Matrix:

$$P = TEU$$

Die Aufgabe des dritten Schritts ist das Element  $P_{ij}$  zu finden, was gleichzeitig das Maximum der Reihe und Spalte ist. Wenn  $P_{ij}$  diese Bedingung erfüllt, sagt man, dass es eine Eins zu Eins Korrespondenz zwischen den Merkmalen  $I_i$  und  $J_j$  gibt. Der ganze Algorithmus ist in Algorithmus 1 aufgeführt, der durch die Eingabe von [Scott & Longuet-Higgins 1991] erzeugt wird.

---

**Algorithm 1** Bestimmung der Korrespondenz der Merkmalen von zwei Bildern

---

```

 $I, J, \sigma, Result$ 
for  $i = 1 \rightarrow m, j = 1 \rightarrow n$  do
     $r_{ij} \leftarrow Dis(I_i, J_j)$ 
     $G_{ij} \leftarrow exp(-\frac{r_{ij}^2}{\sigma^2})$ 
end for
 $T, U \leftarrow$  Singulärwertzerlegung von G
 $E \leftarrow m \times n$  Diagonalmatrix mit  $E_{ii} = 1$ 
 $P \leftarrow TEU$ 
for  $i = 1 \rightarrow m$  do
     $MaxSpalteIndex[i] \leftarrow$  Index der Spalte des maximalen Elements an Reihe  $i$ .
end for
for  $i = 1 \rightarrow m$  do
    if  $P_{iMaxSpalteIndex[i]}$  ist Maximum der Spalte  $MaxSpalteIndex[i]$  then
         $Result \leftarrow$  Punktpaar( $I_i, J_{MaxSpalteIndex[i]}$ )
    end if
end for

```

---

## 3.5 Bestimmung der Transformation

### 3.5.1 Quaternion

Ein Quaternion besteht aus einem Vektor mit 4 Elementen, wobei ein Element ein Skalar bezeichnet und die anderen drei eine Richtung im 3D Raum beschreiben. Quaternionen können aber auch als eine Erweiterung der komplexen Zahlen betrachtet werden, deren Imaginärteil nach drei neuen Zahlen  $i$ ,  $j$  und  $k$  entwickelt wird. Eine Normalform der Quaternion ist gegeben durch:

$$q = q_0 + iq_x + jq_y + kq_z,$$

wobei  $i$ ,  $j$  und  $k$  die sogenannte Hamilton-Regeln erfüllen:

$$\begin{aligned} i^2 &= j^2 = k^2 = ijk = -1, \\ ij &= k, \quad jk = i, \quad ki = j, \\ ji &= -k, \quad kj = -i, \quad ik = -j, \end{aligned}$$

Eine andere Form mit getrennten Realteil und Imaginärteil wird definiert durch:

$$q = (q_0, \vec{q}), \quad (3.13)$$

wobei  $q_0 \in \mathbb{R}$  ein Skalar und  $\vec{q} \in \mathbb{R}^3$  ein Vektor ist.

Sei  $r$  ein anderes Quaternion mit:

$$r = r_0 + ir_x + jr_y + kr_z.$$

Analog zu Vektoren im  $\mathbb{R}^4$  wird das Skalarprodukt zwischen zwei Quaternion definiert als:

$$\langle q, r \rangle := q \cdot r := q_0 r_0 + q_x r_x + q_y r_y + q_z r_z.$$

Weiterhin kann die Quaternionmultiplikation mithilfe (3.13) berechnet werden als:

$$qr = (q_0 r_0 - \vec{q} \cdot \vec{r}, \quad q_0 \vec{r} + \vec{q} r_0 + \vec{q} \times \vec{r}) \quad (3.14)$$

$$\begin{aligned} &= (q_0 r_0 - q_x r_x - q_y r_y - q_z r_z) \\ &\quad + i(q_0 r_x + q_x r_0 + q_y r_z - q_z r_y) \\ &\quad + j(q_0 r_y - q_x r_x + q_y r_0 + q_z r_z) \\ &\quad + k(q_0 r_z + q_x r_y - q_y r_x + q_z r_0). \end{aligned} \quad (3.15)$$

Die rechte Multiplikation von  $r$  in Formel (3.15) kann aber auch zu einer links multiplizierten Matrix umgeschrieben werden:

$$qr = \begin{pmatrix} r_0 & -r_x & -r_y & -r_z \\ r_x & r_0 & r_z & -r_y \\ r_y & -r_z & r_0 & r_x \\ r_z & r_y & -r_x & r_0 \end{pmatrix} = \mathbf{R}q. \quad (3.16)$$

Das konjugierte Quaternion von  $q$  ist definiert als:

$$\bar{q} = q_0 - iq_x - jq_y - kq_z.$$

Das Produkt eines Quaternion und dessen Konjugierte ist eine nicht negative reelle Zahl:

$$q \cdot \bar{q} = q_0^2 + q_x^2 + q_y^2 + q_z^2.$$

Mithilfe des konjugierten Quaternion kann man die Länge des Quaternion  $|q|$  definieren:

$$|q| = \sqrt{q \cdot \bar{q}}.$$

Ist die Länge eines Quaternion gleich 1, nennt man das Quaternion ein Einheitsquaternion. Für ein Einheitsquaternion gilt:

$$q \cdot \bar{q} = 1 \iff \bar{q} = q^{-1}.$$

D.h. die Inverse und Konjugierte sind identisch. Für jedes Einheitsquaternion  $q \neq \pm 1$  gibt es eine entsprechende Polardarstellung:

$$q = \cos \alpha + v \cdot \sin \alpha \quad (3.17)$$

mit  $\alpha = \arccos(q_0) \in (0, \pi)$  und  $v = \frac{1}{\sin \alpha}(iq_x + jq_y + kq_z)$ .

### 3.5.2 Beschreibung der Drehungen im Dreidimensionalen Raum mit Quaternionen

Die Drehungen im dreidimensionalen Raum können durch die Einheitsquaternionen sehr gut beschrieben werden. Eine Abbildung der Rotation  $\rho_q$  kann in folgender Form definiert werden:

$$\rho_q : x \rightarrow qx\bar{q},$$

wobei  $q$  ein Einheitsquaternion und  $\bar{q}$  dessen Konjugierte ist. Mithilfe der Polardarstellung (3.17) kann die Abbildung  $\rho_q$  sich auf eine Drehung im  $\mathbb{R}^3$  um die Achse  $v$  mit Winkel  $2\alpha \in (0, 2\pi)$  beziehen. Die entsprechende orthogonale Matrix von  $q$  ist

$$R = \begin{pmatrix} q_0^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y - 2q_0q_z & 2q_xq_z + 2q_0q_y \\ 2q_xq_y + 2q_0q_z & q_0^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z - 2q_0q_x \\ 2q_xq_z - 2q_0q_y & 2q_yq_z + 2q_0q_x & q_0^2 - q_x^2 - q_y^2 + q_z^2 \end{pmatrix} \quad (3.18)$$

was zur Drehgruppe  $\text{SO}(3)$  gehört und eine Drehung in der Matrixform repräsentiert.

### 3.5.3 Orientierung mit Einheitsquaternion

Das Verfahren für die Orientierung der Objekte mithilfe der Quaternionen wurde von Horn im 1987 veröffentlicht [Horn 1987]. Seien  $D$  und  $M$  zwei Punktmengen mit gleicher Größe  $n$ . Dann kann die Transformation zwischen den Punkten von zwei Mengen formuliert werden als:

$$d_i = \mathbf{R}m_i + \mathbf{T} + e_i, \quad (3.19)$$

wobei  $d_i$  und  $m_i$  die i-ten Punkte der Punktmengen  $D$  bzw.  $M$  bezeichnen.  $\mathbf{R}$  ist die Rotationsmatrix und  $\mathbf{T}$  ist die Translationsmatrix.  $e_i$  beschreibt den Fehler für die Transformation, und kann umformuliert werden als:

$$e_i = d_i - \mathbf{R}m_i + \mathbf{T}. \quad (3.20)$$

Das Ziel des Verfahrens ist, eine Rotations- bzw. Transformationsmatrix mit minimalem Fehler zu finden, dadurch wird die Summe des Quadrats von  $e_i$  betrachtet:

$$\sum_{i=1}^n \|e_i\|^2 = \sum_{i=1}^n \|d_i - \mathbf{R}m_i + \mathbf{T}\|^2. \quad (3.21)$$

Seien  $\bar{d}$  und  $\bar{m}$  jeweils die Schwerpunkte der Punktmengen  $D$  und  $M$ . Dann gilt

$$\bar{d} = \frac{1}{n} \sum_{i=1}^n d_i \quad , \quad \bar{m} = \frac{1}{n} \sum_{i=1}^n m_i. \quad (3.22)$$

Der Abstand von jedem Punkt zum Schwerpunkt wird berechnet als:

$$d'_i = d_i - \bar{d} \quad , \quad m'_i = m_i - \bar{m}. \quad (3.23)$$

und die Summe der Abstände erfüllt natürlich

$$\sum_{i=1}^n d'_i = 0 \quad \text{und} \quad \sum_{i=1}^n m'_i = 0. \quad (3.24)$$

Dann kann der Fehler in Formel (3.20) mit den Abständen zum Schwerpunkt  $\bar{d}$  und  $\bar{m}$  umgeschrieben werden:

$$e_i = d'_i - \mathbf{R}m'_i + \mathbf{T}', \quad (3.25)$$

wobei  $\mathbf{T}'$  als

$$\mathbf{T}' = \mathbf{T} - \bar{d} + \mathbf{R}\bar{m}$$

definiert wird. Analog kann die Summe des Quadrats des Fehlers neu formuliert werden.

$$\begin{aligned} \sum_{i=1}^n \|e_i\|^2 &= \sum_{i=1}^n \|d'_i - \mathbf{R}m'_i + \mathbf{T}'\|^2 \\ &= \sum_{i=1}^n \|d'_i - \mathbf{R}m'_i\|^2 - 2\mathbf{T}' \cdot \sum_{i=1}^n (d'_i - \mathbf{R}m'_i) + n\|\mathbf{T}'\|^2 \end{aligned} \quad (3.26)$$

Wegen (3.24) ist der zweite Term gleich 0. Der dritte Term kann nicht negativ und wird auch 0 sein, wenn der gesamte Fehler minimiert wird. D.h.:

$$\begin{aligned} \mathbf{T}' &= \mathbf{T} - \bar{d} + \mathbf{R}\bar{m} = 0 \\ \Rightarrow \mathbf{T} &= \bar{d} + \mathbf{R}\bar{m}. \end{aligned} \quad (3.27)$$

Die Formel (3.27) berechnet direkt die Translationsmatrix durch die Rotationsmatrix und die Schwerpunkte der beiden Punktmengen. Der erste Term von (3.26) kann zu

$$\sum_{i=1}^n \|d'_i - \mathbf{R}m'_i\|^2 = \sum_{i=1}^n (d'^t_i d'_i + m'^t_i m'_i - 2d'^t_i \mathbf{R}m'_i) \quad (3.28)$$

weiter formuliert werden. Dann wird die Minimierung des Fehlers durch die Bestimmung des Maximums der Summe

$$\sum_{i=1}^n d'^t_i \mathbf{R}m'_i \quad (3.29)$$

erreicht. Durch Ersetzen der Rotationsmatrix mit Quaternion  $q$  wird das maximierte Problem umformuliert als:

$$\sum_{i=1}^n (qm''_i \bar{q}) \cdot d''_i, \quad (3.30)$$

wobei  $\bar{q}$  das konjugierte Quaternion von  $q$  ist,  $m''_i = (0, m'_{i,x}, m'_{i,y}, m'_{i,z})$  und  $d''_i = (0, d'_{i,x}, d'_{i,y}, d'_{i,z})$  die erweiterte Quaternion für Punkte  $m'_i$  bzw.  $d'_i$  sind. Dann gilt:

$$\begin{aligned} \sum_{i=1}^n (qm''_i \bar{q}) \cdot d''_i &= \sum_{i=1}^n (qm''_i \bar{q}) \cdot (d''_i q \bar{q}) \\ &= \sum_{i=1}^n (qm''_i) \cdot (d''_i q). \end{aligned} \quad (3.31)$$

Die beiden Multiplikationen in Klammern können als Formel (3.15) zum Produkt von einem Quaternion und einer Matrix umschrieben werden:

$$qm''_i = \begin{pmatrix} 0 & -m'_{i,x} & -m'_{i,y} & -m'_{i,z} \\ -m'_{i,x} & 0 & -m'_{i,z} & -m'_{i,y} \\ -m'_{i,y} & -m'_{i,z} & 0 & -m'_{i,x} \\ -m'_{i,z} & -m'_{i,y} & -m'_{i,x} & 0 \end{pmatrix} = \mathbf{M}_i q$$

und

$$d''_i q = \begin{pmatrix} 0 & -d'_{i,x} & -d'_{i,y} & -d'_{i,z} \\ d'_{i,x} & 0 & -d'_{i,z} & d'_{i,y} \\ d'_{i,y} & d'_{i,z} & 0 & -d'_{i,x} \\ d'_{i,z} & -d'_{i,y} & d'_{i,x} & 0 \end{pmatrix} = \mathbf{D}_i q.$$

Dann kann (3.31) weiter abgeleitet werden:

$$\begin{aligned} \sum_{i=1}^n (\mathbf{M}_i q) \cdot (\mathbf{D}_i q) &= \sum_{i=1}^n q^t \mathbf{M}_i^t \mathbf{D}_i q \\ &= q^t \left( \sum_{i=1}^n \mathbf{M}_i^t \mathbf{D}_i \right) q \\ &= q^t \left( \sum_{i=1}^n \mathbf{N}_i \right) q \\ &= q^t \mathbf{N} q, \end{aligned} \quad (3.32)$$

wobei  $\mathbf{N}_i = \mathbf{M}_i^t \mathbf{D}_i$  ist, und  $\mathbf{N}$  die Summe von  $\mathbf{N}_i$  beschreibt. Sei  $\mathbf{H}$  die Summe der Kreuzprodukte des Punktpaars von Punktmengen  $D$  und  $M$ :

$$\mathbf{H} = \sum_{i=1}^n m_i d_i^t. \quad (3.33)$$

Es ist deutlich, dass die Größe der Matrix  $\mathbf{H}$   $3 \times 3$  ist, deshalb kann  $\mathbf{H}$  auch als

$$\mathbf{H} = \begin{pmatrix} S_{xx} & S_{xy} & S_{xz} \\ S_{yx} & S_{yy} & S_{yz} \\ S_{zx} & S_{zy} & S_{zz} \end{pmatrix} \quad (3.34)$$

geschrieben werden, wobei

$$S_{xx} = \sum_{i=1}^n m'_{i,x} d'_{i,x}, \quad S_{xy} = \sum_{i=1}^n m'_{i,x} d'_{i,y}, \quad \dots$$

Dann kann die Matrix  $\mathbf{N}$  im (3.32) durch die Elemente von  $\mathbf{H}$  dargestellt werden als:

$$\mathbf{N} = \begin{pmatrix} S_{xx} + S_{yy} + S_{zz} & S_{yz} - S_{zy} & S_{zx} - S_{xz} & S_{xy} - S_{yx} \\ S_{yz} - S_{zy} & S_{xx} - S_{yy} - S_{zz} & S_{xy} + S_{yx} & S_{zx} + S_{xz} \\ S_{zx} - S_{xz} & S_{xy} + S_{yx} & -S_{xx} + S_{yy} - S_{zz} & S_{yz} + S_{zy} \\ S_{xy} - S_{yx} & S_{zx} + S_{xz} & S_{yz} + S_{zy} & -S_{xx} - S_{yy} + S_{zz} \end{pmatrix} \quad (3.35)$$

Nach dem Beweis von Horn [Horn 1987] wird die Formel (3.32) maximal, genau dann, wenn  $q$  der zu dem maximalen positiven Eigenwert der Matrix  $\mathbf{N}$  entsprechende Eigenvektor ist.

## 3.6 Objekterkennung

### 3.6.1 DBSCAN

DBSCAN, kurz von dem englischen Namen „Density-Based Spatial Clustering of Applications with Noise, ist ein auf Dichte basierter Data-Mining-Algorithmus“ [Ester et.al 1996]. Die Hauptidee des Algorithmus hängt stark von dem Begriff „Dichteverbundenheit“ ab, was durch folgende Definitionen erklärt wird.

Seien  $D$  eine Punktmenge im Raum  $\mathbb{R}^n$  und  $Dist(p, q)$  eine darauf definierte Distanzfunktion.  $\epsilon$  und MinPts sind zwei Eingaben des Algorithmus.

**Definition 3.1**  $\epsilon$ -Umgebung  $N_\epsilon(p)$  ist eine Menge der Punkte um  $p$ , die

$$N_\epsilon(p) = \{q \in D \mid \text{Dist}(p, q) \geq \epsilon\}$$

erfüllt.

**Definition 3.2** Ein Punkt  $p$  ist direkt Dichte-erreichbar zum Punkt  $q$ , g.d.w:

- 1)  $p \in N_\epsilon(q)$
- 2)  $|N_\epsilon(q)| \geq \text{MinPts}$

**Definition 3.3** Ein Punkt  $p$  ist Dichte-erreichbar zum Punkt  $q$ , g.d.w es eine Kette von Punkten  $p_1, \dots, p_n$  mit  $p_1 = p$  und  $p_n = q$  gibt, wobei  $p_{i+1}$  direkt Dichte-erreichbar zum  $p_i$  für alle  $i \in [1, n]$  ist.

**Definition 3.4** Zwei Punkte  $p$  und  $q$  heißt Dichte-verbunden, g.d.w es einen Punkt  $o$  gibt, wobei  $p$  und  $q$  jeweils zu  $o$  Dichte-erreichbar sind.

Mithilfe dieser Definitionen der Beziehung zwischen den Punkten kann man eine einzige Definition des Clusters ausgeben.

**Definition 3.5** Sei  $C$  eine nicht leere Teilmenge von  $D$ . Dann heißt  $C$  ein Cluster, wenn die folgenden zwei Bedingungen erfüllt werden:

- 1) für  $\forall p, q \in D$ , wenn  $p \in C$  und  $q$  ist Dichte-erreichbar zum  $p$ , dann gilt  $q \in C$ ,
- 2) für  $\forall p, q \in C$ ,  $p$  ist Dichte-verbunden zu  $q$ .

Dann können alle Punkte von  $D$  in drei verschiedene Typen zusammengefasst werden.

- Kernpunkt: Der Punkt, dessen  $\epsilon$ -Umgebung größer als MinPts ist.
- Grenzpunkt: Der Punkt, dessen  $\epsilon$ -Umgebung nicht groß genug ( $< \text{MinPts}$ ), aber zu anderen Punkten Dichte-erreichbar ist.
- Geräusch: Der Punkt, der zu keinem Cluster gehört.

Abbildung 3.10 zeigt ein Beispiel für alle diesen drei Typen eines Punkts, wobei die Kernpunkte mit Rot, Grenzpunkte mit Gelb und Rauschen mit Blau dargestellt werden. Die Kreise zeigen die  $\epsilon$ -Umgebungen für die Punkte an ihren Ursprüngen.

Der Algorithmus durchläuft für jeden Punkt von Punktmenge  $D$  und die Lösung hängt von der Reihenfolge der Punkte nicht ab. Ein Kernpunkt soll zuerst gefunden werden, und dann die andere Punkte in ihrer  $\epsilon$ -Umgebung betrachtet werden. Zwei  $\epsilon$ -Umgebung werden verknüpft, wenn sie mindesten einen identische Punkt haben. Der genaue Ablauf des DBSCAN-Algorithmus ist in Algorithmus 2 ausgegeben.

---

**Algorithm 2** DBSCAN

---

$D, \epsilon, \text{MinPts}$

Setzen  $C$  zum ersten Cluster

**for** jede  $P_i \in D$  **do**

**if**  $P_i$  ist nicht besucht **then**

        Markieren  $P_i$  als besucht

$N = \{P_j \in D | \text{Dist}(P_i, P_j) < \epsilon\}$

**if** Anzahl der Elemente von  $N < \text{MinPts}$  **then**

            Markieren  $P_i$  als Geräusch

**else**

            Fügen  $P_i$  in aktuellem Cluster  $C$  ein

**for** jede  $P_j \in N$  **do**

**if**  $P_j$  ist noch nicht besucht **then**

                    Markieren  $P_j$  als besucht

$N' = \{P_k \in D | \text{Dist}(P_j, P_k) < \epsilon\}$

**if** Anzahl der Elemente von  $N' \geq \text{MinPts}$  **then**

$N = N \cup N'$

**end if**

**end if**

**if**  $P_j$  gehört zu keinem Cluster **then**

                    Fügen  $P_j$  in aktuellem Cluster  $C$  ein

**end if**

**end for**

        Setzen  $C$  zum nächsten Cluster

**end if**

**end if**

**end for**

---

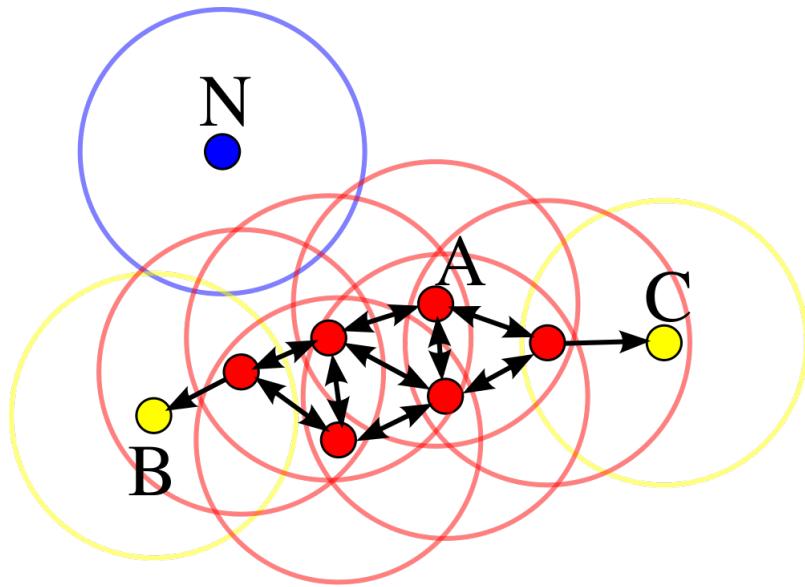


Abbildung 3.10: Ein Beispiel für die drei Typen der Punkte in DBSCAN-Algorithmus. Der rote Punkt A ist ein Kernpunkt. Die gelbe Punkte B und C sind die Grenzpunkte, die von roten Punkten Dichte-erreichbar sind. Wegen keine Dichte-Verbindung zwischen blau Punkt N und andere Punkte, wird N als Geräusch erkannt. [DBSCAN Wiki]

### 3.6.2 Teilgraph Isomorphismus

Die Objekterkennung bzw. Objektverfolgung wird durch den Vergleich der Modellgraphen und Eingabegraphen realisiert. Ein für diese Arbeit hilfreicher Algorithmus wurde von Rhijn und Mulder entwickelt [Rhijn & Mulder 2005]. Um das Problem des Graphisomorphismus zu vereinfachen, werden nur ein Teilgraph  $S_{min}$  von Modellgraphen betrachtet.  $S_{min}$  soll ein vollständiger Graph sein und einen Modellgraphen eindeutig bestimmen können. Zuerst wird ein Punkt  $p_i$  vom Eingabograph ausgewählt und die Distanzen von diesem zu allen seinen Nachbarn berechnet. Alle diese Kantenlängen werden mit den Kanten der Modellgraphen verglichen, damit ein Kandidatenpunkt  $v_k$  im Modellgraph gefunden werden kann, der genug Kanten mit dem ausgewählten Punkt  $p_i$  identisch hat. Zweitens sollen drei Nachbarn von  $p_i$  gefunden werden, die mit  $p_i$  zusammen einen vollständigen Graphen (Pyramide) darstellen können. Wenn ein Teilgraph von Modellgraphen zum obigen vollständigen Graphen assoziiert wird, ist ein Teilgraph Isomorphismus zwischen den Eingabegraphen und Modellgraphen gefunden. Der Algorithmus 3 gibt eine genauere Beschreibung des Algorithmus von Rhijn und Mulder an.

---

**Algorithm 3** Teilgraph Isomorphismus

---

Modellgraph: $G_m$ , Eingabegraph: $G_d$

```

for jede  $p_i \in G_d$  do
    for jeder Nachbar  $p_j$  von  $p_i$  do
        for alle Punktpaare  $(v_k, v_l) \in G_m$  do
            if  $dist(p_i, p_j) = dist(v_k, v_l)$  then
                Fügen assoziierte Punktpaar  $< p_j, v_l >$  zum  $S_i$  ein
            end if
        end for
        if  $\|S_i\| \geq \|S_{min}\|$  then
            if Drei Punktpaare in  $S_i$  gefunden können, damit deren ersten Punkte mit
             $p_i$  ein Pyramid aufbauen können then
                Teilgraph Isomorphismus ist gefunden
                Break
            end if
        end if
    end for
end for

```

---

# Kapitel 4

## Implementierung

Die Implementierung kann in vier Hauptmodule unterteilt werden:

- Markenanalyse,
- Objekteinlernen,
- Objekterkennung und Verfolgung,
- Bilder Steuerung.

Der allgemeine Ablaufprozess des Programms ist in Abbildung 4.1. Das ganze Programm ist eine Schleife, in jedem deren Schritt die Information der PMD Kamera angesammelt und als Eingabe genutzt wird. Nach Bewertung der neuen Eingabe bzw. der Rückkopplung vom letzten Ablauf werden die entsprechenden Bilddaten vom Modul *Bilder Steuerung* für die weiteren Schritte ausgewählt. Das Modul *Markenanalyse*

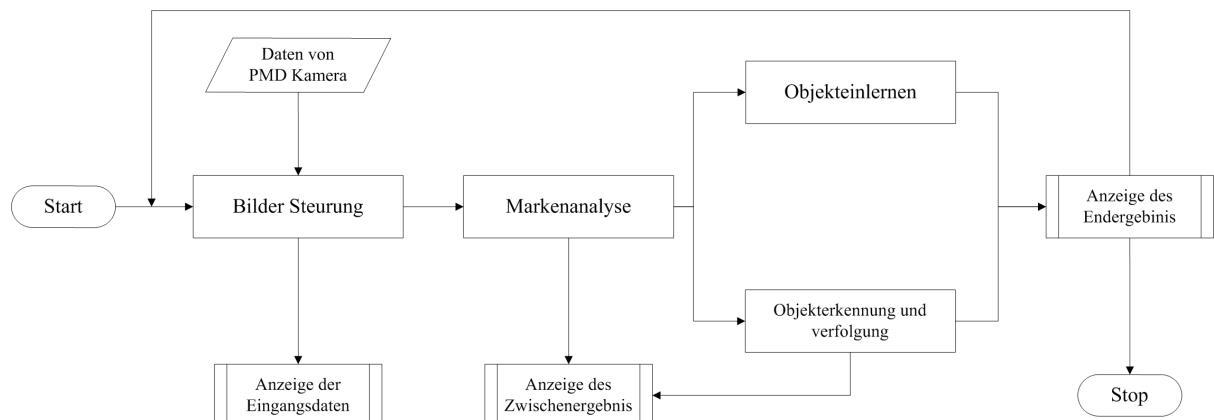


Abbildung 4.1: Ablaufdiagramm

analysiert die eingegebene Bilddaten und versucht die Marken zu finden und zu verfolgen. Die ausgefundenen Marken werden entweder von dem *Objekteinlernen* oder durch die *Objekterkennung und Verfolgung* benutzt, was am Anfang des Programms durch den Benutzer entschieden wird. In der Einlernensphase wird zuerst die Transformation des Objekts bestimmt und dann der charakteristische Graph des Objekts durch die Marken dargestellt. Wenn man ein Objekt wieder erkennen möchte, werden die Marken von neuem Objekt zu den vorhandenen charakteristischen Graphen verglichen, die nach dem Objekteinlernen im Speicher gespeichert werden. Das Ergebnis wird in einem OpenGL Fenster ausgezeigt und als Rückkopplung für den nächsten Schritt genutzt. Alle diese Module werden in den folgenden Unterabschnitten genau erklärt. Ein ausführliches Ablaufdiagramm wird in Abbildung 4.2 gezeigt.

## 4.1 Markenanalyse

### 4.1.1 Bildvorverarbeitung

#### 4.1.1.1 Datenstruktur des Eingabebilds

Wie im Abschnitt 3.1.2 erklärt, liefert die PMD Kamera insgesamt vier verschiedenen Arten der Vermessungsdaten. Alle diese Daten eines Bildes werden in einer Instanz von Klasse **BildData** gespeichert. Dazwischen sind die Amplituden und 3D Koordinaten ganz wichtig und werden in verschiedenen Teilen des Programms verwendet: die Amplituden werden für die Markenerkennung und die 3D Koordinaten mit räumlicher Information werden später für die Korrespondenzuntersuchung und Schätzung der Lage des Objekts benutzt. Da die definierte Dimension und das definierte Intervall dieser zwei Arten der Daten sich von anderem unterscheidet, ist vor der weiteren Bildverarbeitung dieser Arbeit die Übereinstimmung beider Daten für jeden Pixel notwendig. Die Klasse **PMDPoint** wird nun definiert, um den Zweck zu erfüllen. Die Klassendiagramme beider Klassen werden in der Abbildung 4.3 gezeigt.

#### 4.1.1.2 Abstand Filter

Die Verbesserung der Ergebnisse durch die Segmentierung des fokussierten Objekts aus der Umgebung wurde im Abschnitt 3.2.1 bemerkt. Deshalb soll am Anfang der Markenanalyse einen Abstand-Filter definiert werden. Der Abstand-Filter dieser Arbeit besteht aus zwei Teilen: der Teil der Initialisierung bzw. der Teil der Filterung. Zwischen der Initialisierung des Filters werden eine Menge der Tiefdaten der Eingabebilder mit vordefinierter Größe angesammelt, damit das durchschnittliche Tiefbild der Szene erzeugt werden kann. Der Pseudocode wird in Algorithmus 4 gezeigt.

Durch Vergleich der in Initialisierungsphase erzeugten Szene und die aktuelle Tiefdaten der 3D Eingabe, können die neu in der Szene eingefügten Objekte aus den Hin-

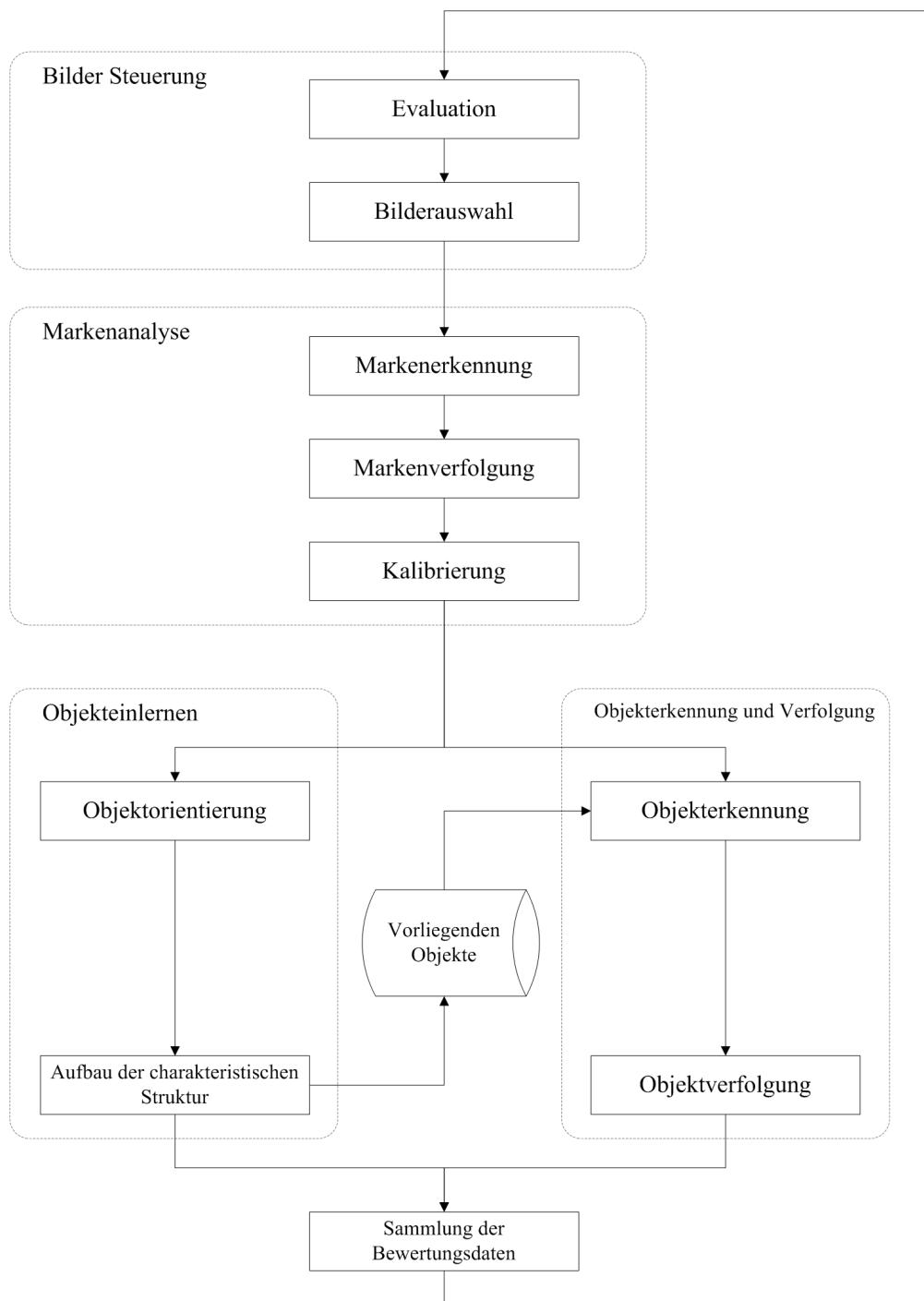


Abbildung 4.2: Ausführliches Ablaufdiagramm

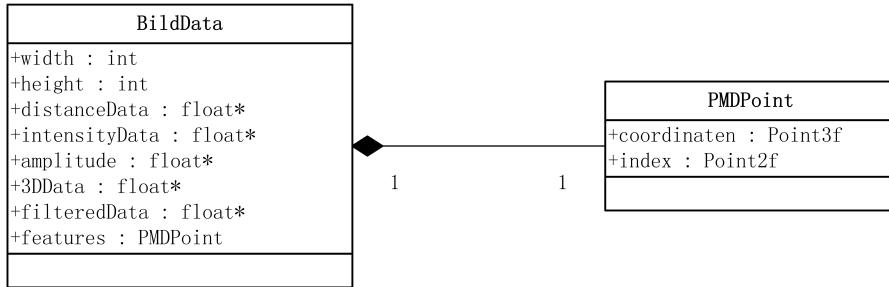


Abbildung 4.3: Die Klassendiagramme für **BildData** und **PMDPoint**.

---

#### Algorithm 4 Initialisierung des Abstand-Filters

---

Anzahl der notwendigen Eingabebilder:  $N$   
 Tiefbild für die durchschnittliche Szene:  $D$

```

for  $i$  von 1 bis  $N$  do
   $E \leftarrow$  aktuelle BildData
  if  $D == \text{NULL}$  then
     $D \leftarrow E.3DKoordinaten.Z$ 
  else
    for jedes Pixel  $d_j$  von  $D$  do
       $d_j \leftarrow \frac{1}{2}(d_j + e.3DKoordinaten_{j_z})$ 
    end for
  end if
end for
  
```

---

tergrund bestimmt werden. Wenn der Abstand zwischen einem Pixel und dem Hintergrund größer als der vordefinierte Schwellenwert ist, wird die Amplitude des gefilterten Bildes in diesem Pixel von originaler Amplitude übertragen. Sonst wird die Amplitude als null ersetzt. Die Anzahl der unterschiedlichen Bildpunkten wird gleichzeitig abgezählt, damit eine boolesche Ausgabe über die Verschiedenheit mit dem gefilterten Bild zusammen zurück gegeben werden kann. Der genaue Ablauf wird in Algorithmus 5 beschrieben.

## 4.1.2 Markenerkennung

### 4.1.2.1 Auswahl der Größe der Marken

Die Erkennungsergebnisse der Objekte werden von der Größe der Marken stark beeinflusst, weshalb ein Test über die Markengröße vor der Implementierung notwendig

---

**Algorithm 5** Filterungsphase des Abstand-Filters

---

```

Schwellenwert für Abstandsvergleich:  $\epsilon$ 
Schwellenwert für Verschiedenheit:  $\alpha$ 
 $E \leftarrow$  aktuelle BildData
for jedes Pixel  $d_i$  von  $D$  do
    if  $\|d_i - e.3DKoordinaten_{i_z}\| < \epsilon$  then
         $e.filteredAmplitude_i \leftarrow 0$ 
    else
         $e.filteredAmplitude_i \leftarrow e.Amplitude_i$ 
    end if
end for
 $q \leftarrow \|veränderten\ Pixeln\| / \|E\|$ 
if  $q < \alpha$  then
    return Falsch
else
    return True,  $E$ 
end if

```

---

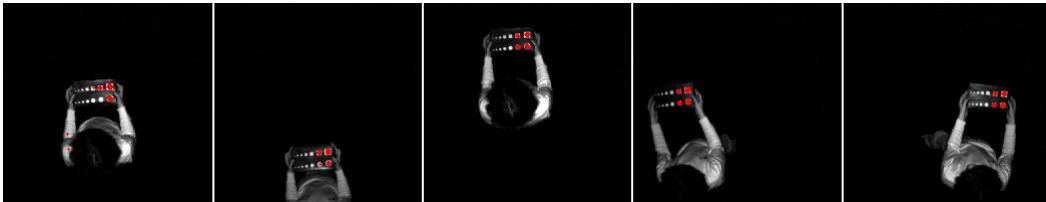


Abbildung 4.4: Die Erkennungsergebnisse der Marken für unterschiedliche Positionen.

ist. Die Testmarken werden als weiße Punkte auf einem schwarzen Papier gedruckt. Es gibt insgesamt 7 verschiedenen Größenstufen. Die Marken von jeder Stufe werden jeweils durch Quadrat und Kreis dargestellt. Abbildung 4.4 zeigt die unterschiedliche Erkennungsergebnisse der Marken, wenn das Objekt an verschiedenen Positionen aber auf der gleichen Höhenebene liegt. Abbildung 4.5 zeigt den Einfluss auf die Ergebnisse von der Distanz zu der Kamera. Was deutlich ist, dass je näher das Objekt zu der Kamera angebracht wird, desto kleinere Markengrößen werden benötigt.

#### 4.1.2.2 STAR Detektor

Wegen der in Abschnitt 3.3.1 festgestellten Gründe wird der STAR Detektor (CenSurE Algorithmus) in dieser Arbeit als Erkennungsalgorithmus ausgewählt. Die Parameter des Detektors werden im Dokument von OpenCV aufgelistet [[StarDetector OpenCV](#)] und von „ButterCookies“ in OpenCV Adventure [[StarDetector OpenCV Adaventure](#)]

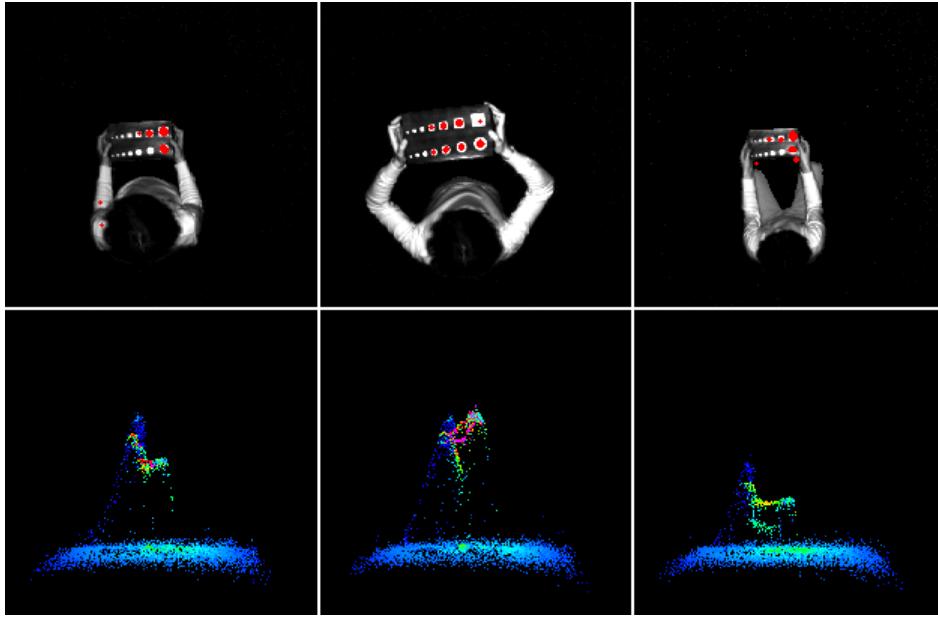


Abbildung 4.5: Die Erkennungsergebnisse der Marken für unterschiedliche Distanz zu der Kamera. Auf der ersten Reihe liegen die Graustufenbilder mit roten erkannten Marken; Die entsprechende 3D Bilder auf der zweiten Reihe zeigen die vertikalen Positionen des Objekts.

weiter deutlicher erklärt. Der Parameter *maxSize* definiert die maximale Größe der Marken. *responseThreshold* ist ein Schwellenwert über die Antwort von approximierten Laplacian. Die erkannten Marken mit niedrigerer Antwort als diesen Schwellenwert werden dann eliminiert. Die Parameter *lineThresholdProjected* und *lineThresholdBinarized* stellen die Stärke der Linie-Suppression ein. Der letzte Parameter *suppressNonmaxSize* beschreibt die Größe des betrachteten Raums der Non-Maximal Suppression. In dieser Arbeit benutzt *responseThreshold* als den variablen Parameter, d.H. dessen Wert in jedem Schritt verändert wird.

#### 4.1.2.3 Markenerkennung

Algorithmus 6 zeigt den Verlauf der Markenerkennung mit STAR Detektor. Der ganze Erkennungsprozess wird als eine Schleife mit vordefinierter maximaler Anzahl der Durchläufe dargestellt. In jedem Schleifenrumpf wird das Graustufenbild als Eingabe des STAR Detektors mit dem aktuellen Kontrast vor der Erkennung erzeugt. Nach der Erkennung schickt das Programm dann das aktuelle Graustufenbild und die Anzahl der erkannten Marken zu der Funktion **Kontraststeuerung** (sehen 4.1.2.4 und Algorithmus 7), und wird von ihr den neuen Kontrast bzw. ResponseThreshold für nächsten Durchlauf erhalten.

---

**Algorithm 6** Markenerkennung

---

```

Eingabebild:  $I$ , maximale Schleife:  $N$ 
 $a \leftarrow$  Anfangswert für Kontrast
 $r \leftarrow$  Anfangswert für ResponseThreshold
 $b \leftarrow$  feste Helligkeit
for  $i \text{ von } 1 \text{ bis } N$  do
    Eingabe für STAR Detektor:  $H_i \leftarrow aI + b$ 
    Erkannten Marken:  $P \leftarrow CenSurE(H_i, r)$ 
    Result  $\leftarrow$  Kontraststeuerung( $H_i, a, r, \|P\|$ )
    if Result = False then
        continue mit aktualisiertem Kontrast und ResponseThreshold
    else
        break
    end if
end for

```

---

#### 4.1.2.4 Steuerung der Helligkeit und des Kontrast

Die Helligkeit bzw. der Kontrast können die Ergebnisse der Markenerkennung stark beeinflussen. Eine viele bekannte Arbeit über radiometrische Kalibrierung wurde von Debevec und Malik in [Debevec & Malik 2008] gemacht. Ihres Verfahren ist robust und präzise, aber benötigt viele Bilder von gleicher Szene mit verschiedenen Belichtungszeiten, damit alle Information der Szene für jeweils helleren Bereich bzw. dunkleren Bereich versammelt wird. Wegen der Beschränkung der Kamera kann diese Bedingung in unserer Arbeit leider nicht erfüllt werden. Außerdem ist das Verfahren von Debevec sehr rechenintensiv. Glücklicherweise sollen in der Markenerkennung unserer Arbeit aber nur die Bilddaten um Marken betrachtet werden. D.h. die sonstige Informationen des Bildes können einfach ignoriert werden. Durch diese Grundidee kann ein Algorithmus erzeugt werden, dessen Endbedingung durch Untersuchung der Anzahl der erkannten Marken eingestellt wird.

Der formulierte Durchlauf wird im Algorithmus 7 gefunden. Die Anzahl der erkannten Marken ist die erste Stufe der Prüf norm von Kontraststeuerung. Wenn nicht genug Marken erkannt werden, wird die Strahlungsenergie des Graustufenbildes als die zweite Stufe der Prüf norm überprüft. Für die zu große bzw. zu kleine Strahlungsenergie, verringert bzw. erhöht der Kontrast sich im erlaubten Intervall, was als Eingangsparameter vorher definiert wird. Wenn die Strahlungsenergie zufrieden ist aber noch nicht genug Marken gefunden sind, nimmt das ResponseThreshold ab, damit mehr Marken mit schwächeren Antworten erkannt werden können. Für die Gegenseite für die zu viele erkannten Marken, wird der Wert von ResponseThreshold vergrößert. Bei dieser Situation wird die Strahlungsenergie nicht mehr betrachtet, damit der Algorithmus zur der Konvergenz halten kann. Ein boolescher Wert wird von dem Algorithmus zurückgegeben.

ben, was zeigt, ob befriedigend viele Marken gefunden werden.

---

**Algorithm 7** Kontraststeuerung( $H, \&a, \&r, \|P\|$ )

---

Intervall der erlaubten Strahlungsenergie:  $(E_{min}, E_{max})$   
 Intervall des erlaubten Kontrastes:  $(A_{min}, A_{max})$   
 Intervall des erlaubten ResponseThreshold von STAR Detektor:  $(R_{min}, R_{max})$   
 Intervall des Erwartens der Anzahl der bekannten Marken:  $(P_{min}, P_{max})$   
 $e \leftarrow$  aktuelle Strahlungsenergie von  $H$

```

if  $\|P\| < P_{min}$  then
  if  $e < E_{min}$  then
     $a \leftarrow a - 0.5$ 
    if  $a < A_{min}$  then
       $a \leftarrow A_{min}$ 
      return False
    end if
  else if  $e > E_{max}$  then
     $a \leftarrow a + 0.5$ 
    if  $a > A_{max}$  then
       $a \leftarrow A_{max}$ 
      return False
    end if
  else
     $r \leftarrow r - 5$ 
    if  $r < R_{min}$  then
      return False
    end if
  end if
else if  $\|P\| > P_{max}$  then
  if  $r > R_{max}$  then
    return False
  else
     $r \leftarrow r + 4$ 
  end if
else
  return True
end if
```

---

### 4.1.3 Markenverfolgung

#### 4.1.3.1 Verbesserung der Singulärwertzerlegungsverfahren

Nach erfolgreicher Festlegung der Marken jedes Bildes sollen dann die Korrespondenzen der Marken von zwei nachfolgenden Bildern untersucht werden. Die Korrespondenzpunkte können durch das Verfahren der Singulärwertzerlegung von Scott und Hignnis [Scott & Longuet-Higgins 1991] bestimmt werden. Die genaue Beschreibung ihres Verfahrens wird im Schnitt 3.4.2 aufgeführt, und Algorithmus 1 gibt den Pseudocode des Verfahrens an. In dieser Arbeit werden vier Verbesserungen für das Verfahren von Scott und Hignnis gemacht, damit der die stabileren Ergebnisse erzeugt und die Qualität der Korrespondenzuntersuchung bewertet werden kann.

##### Nebenbedingung für die Bestimmung der größten Elemente

In originalem Algorithmus werden die Punkte  $I_i$  und  $J_j$  als Korrespondenzpunkte erkannt, genau dann, wenn das Element  $P_{ij}$  das größte Element von beide Zeile  $i$  und Spalte  $j$  ist. Aber manchmal liefert das größte Element nicht die beste Korrespondenz, insbesondere in den Fall, dass es große Transformation zwischen zwei betrachteten Bildern gibt. Die Lösung ist, eine weitere Beschränkung für die Untersuchung der größten Elemente einzusetzen. D.h. die größten Elemente werden nur akzeptiert, wenn sie gleichzeitig größer als ein eingegebener Schwellenwert  $\epsilon$  sind. Der Schwellenwert  $\epsilon$  wird in  $[0, 1]$  definiert. Je höher  $\epsilon$  ist, desto ordentlicher sind die gefundenen Korrespondenzpunkte. In dem idealen Fall sind alle größten Elemente  $P_{ij}$  gleich 1, z.B. wenn die beide betrachteten Bilder identisch sind.

##### Erfolgreiche Behauptung

Wenigere Korrespondenzpunkte werden mit obiger stärkeren Nebenbedingung herausgefunden, was aber die weiteren Arbeitsschritte wenig beeinflusst, weil es zumindest nur 3 korrespondierenden Punktpaare benötigt, um die Orientierung zwischen zwei Bildern zu bestimmen. Trotzdem ist die Überprüfung der Anzahl der gefundenen Punktpaare notwendig, und deren Ergebnis wird als eine boolesche Ausgabe des Algorithmus zurückgegeben.

##### Qualitätsmanagement der Korrespondenz

Außer der binären Behauptung des Algorithmus ist die Bewertung der Korrespondenzqualität auch wichtig, damit die gute und stabile Bilderkette für Markenverfolgung ausgewählt werden kann. Das wurde aber leider von Scott und Hignnis in ihrer Arbeit nicht genannt. In der ersten Verbesserung wird die Größe der größten Elemente von Matrix  $P$  mit einem Schwellenwert weiter beschränkt, um das bessere Korrespondenzergebnis zu erhalten. Deshalb zur Umkehr kann die Größe der größten Elemente von  $P$  als die Messung der Korrespondenzqualität definiert werden. Dann wird die Summe aller größten Elemente  $\sum P_{ij}$  in dieser Arbeit zur Bewertung der Korrespondenzuntersuchung verwendet. Hier wird kein arithmetische Mittel benutzt, weil die Anzahl der

betrachteten Punkte, die als Eingaben in den Algorithmus eingegebene werden, auch ein wichtiger Faktor der Bewertung der Korrespondenz ist.

### Die Auswahl von $\sigma$ mit Rückkopplung

Die Einheit des Abstands  $\sigma$  ist der wesentliche Parameter des Singulärwertzerlegungsverfahrens. Das ungeeignete  $\sigma$  erzeugt dann großes Chaos in Korrespondenzlösungen, was schon in [Scott & Longuet-Higgins 1991] mit Schaubildern vielmals gezeigt wird. Um die besten Lösungen zu finden, soll die Abstandseinheit nicht kleiner als die durchschnittlichen Distanz zwischen den Korrespondenzpunkten definiert werden. Es gibt zwei Schwierigkeiten für die Bestimmung des  $\sigma$ . Erstens sind die korrespondierenden Punktpaare vor dem Durchlauf des Algorithmus noch nicht bekannt, weshalb kann die Abstandseinheit nicht direkt berechnet sondern nur geschätzt werden. Zweitens ist die Auswahl einer festen Abstandseinheit schwierig und ineffizient, wenn sich die betrachteten Punkte oder Merkmale, z.B. in dieser Arbeit, uneingeschränkt bewegen können. Deshalb wird hier der Parameter  $\sigma$  für jedes Bild mit der durchschnittlichen Distanz zwischen allen korrespondierenden Punktpaaren vorheriges Bildes festgelegt. Wegen der festen Bildwiederholfrequenz der Eingabebilder sind in theoretischen Fall die durchschnittlichen Abstand der Korrespondenzpunkte jedes Bildes gleich, wenn sich alle betrachteten Punkte gleichförmig bewegen. Aber für die schwach beschleunigenden Bewegungen der Merkmale kann das anpassende  $\sigma$  trotzdem herausgefunden werden und die Korrespondenzlösungen sich stabil erzeugen lassen.

Das verbesserte Algorithmus wird im Algorithmus 8 gezeigt.

#### 4.1.4 Segmentierung

Bis jetzt sind alle erkannten Marken eines Bildes als eine Menge der Punkte zusammen betrachtet, die dann segmentiert werden soll, um die Marken für verschiedene Objekte zu verteilen. Die Hauptidee ist, dass die Marken als die Merkmale von gleichem Objekt erkannt werden, genau dann, wenn die Abstände zwischen diesen Marken viel kleiner als die Abstände von ihr zu den anderen erkannten Marken sind. Die Problemstellung der Segmentierung ist gleich wie ein Clustering-Problem, deshalb wird in dieser Arbeit der Clustering-Algorithmus DBSCAN für die Segmentierung benutzt. Die Beschreibung des Verfahrens wird im Abschnitt 3.6.1 gezeigt und der genaue Durchlauf wird von Algorithmus 2 erklärt. Eine Liste der Punktmengen werden von DBSCAN ausgegeben, und jedes deren Element stimmt mit der Merkmalen eines Objekts überein. Da es zumindest 3 Punkte benötigt, um die Lage des räumlichen Körpers im 3D Raum zu bestimmen, werden die Punktmengen von Ausgabe der DBSCAN als Rauschen erkannt, die weniger als 3 Punkte enthalten.

---

**Algorithm 8** Korrespondenzuntersuchung

---

Korrespondierende Punktpaare:  $Result$

Eingabebilder von jeweils Zeitpunkt  $t_{i-1}$  und  $t_i$ :  $I, J$

Aktuelle approximierte Abstandseinheit:  $\sigma_i$

Schwellenwert für die Beschränkung der größten Elemente:  $\epsilon$

Messung der Korrespondenzqualität:  $mess$

Summe der Abstände der Korrespondenzpunkte:  $sumDistance$

**for**  $i = 1 \rightarrow m, j = 1 \rightarrow n$  **do**

$r_{ij} \leftarrow Dis(I_i, J_j)$

$G_{ij} \leftarrow exp(-\frac{r_{ij}^2}{\sigma_i^2})$

**end for**

$T, U \leftarrow$  Singulärwertzerlegung von  $G$

$E \leftarrow m \times n$  Diagonalmatrix mit  $E_{ii} = 1$

$P \leftarrow TEU$

$minMN \leftarrow Min(m, n)$

**for**  $i = 1 \rightarrow minMN$  **do**

$MaxSpalteIndex[i] \leftarrow$  Index der Spalte des maximalen Elements an Reihe  $i$ .

**end for**

**for**  $i = 1 \rightarrow minMN$  **do**

**if**  $P_{iMaxSpalteIndex[i]}$  ist Maximum der Spalte  $MaxSpalteIndex[i]$  **then**

**if**  $P_{iMaxSpalteIndex[i]} > \epsilon$  **then**

$Result \leftarrow$  Punktpaar( $I_i, J_{MaxSpalteIndex[i]}$ )

$mess \leftarrow mess + P_{iMaxSpalteIndex[i]}$

**end if**

**end if**

**end for**

**for** Alle Punktpaare  $(I_i, J_j) \in Result$  **do**

$sumDistance \leftarrow sumDistance + DistanceOf(I_i, J_j)$

**end for**

$\sigma_{i+1} \leftarrow sumDistance / \|Result\|$

**if**  $\|Result\| < 3$  **then**

**return** False

**else**

**return** True,  $mess$

**end if**

---

## 4.2 Objektlernen

Nach der erfolgreichen Markenanalyse werden eine Menge der Marken aus den Eingabebildern herausgefunden. Weiterhin sind die Abhängigkeiten dieser Marken zwischen benachbarten Bildern auch bekannt. Durch die Segmentierung werden diese Marken dann für unterschiedlichen Objekte weiter verteilt. Im diesen Abschnitt wird es erklärt, wie ein charakteristischer Graph des Objekts mithilfe der erkannten Marken dargestellt werden kann. In dieser Arbeit wird im Objektlernen die vereinfachte Situation berücksichtigt, dass einmal nur ein Objekt in den Eingabebildern vorkommt. Deshalb wird nur die größten Punktmenge von Segmentierungsergebnis für Objektlernen ausgewählt, die als die Eingabe der folgenden Arbeitsschritte eingestellt wird.

### 4.2.1 Bestimmung der Orientierung

Um ein Objekt zu lernen, muss das Objekt mit Marken zuerst zu der Kamera gezeigt und sich langsame umdreht werden. Zwischen der Umdrehung werden dann die relativen Positionen der Marken an jeder Ebene bestimmt. Deshalb sollen die Lagen der bekannten Marken in diesem Ablauf rechtzeitig aktualisiert werden, was als ein Orientierungsproblem formuliert werden kann. In dieser Arbeit wird der Verfahren von [Horn 1987] verwendet, um die Transformation der bekannten Marken zwischen nachfolgenden Bildern zu berechnen. Der genaue Durchlauf wird im Abschnitt 3.5.3 aufgeführt. Zuerst sollen die Schwerpunkte der erkannten Marken herausgefunden werden (sehen Formel (3.22)). Dann berechnet man die Vektoren, die von alle Marken zu ihren entsprechenden Schwerpunkten richten. Drittens wird die Korrelationsmatrix  $H$  durch Formel (3.33) mit diesen Vektoren bestimmt werden. Die Hilfematrix  $N$  in Formel (3.35) besteht aus den Elementen der Matrix  $H$ , und Ein von ihnen Eigenvektoren ist die Einheitsquaternion für die Rotation, genau dann, wenn der den größten positiven Eigenwert entspricht. Die Rotation- bzw. die Translationsmatrix unter kartesischem Koordinatensystem werden durch jeweils die Formel (3.18) und (3.27) berechnet.

Die Lage des Objekts von jedem Bild hängt nur von der Lage des Objekts von vorherigem Bild ab. Wegen der kontinuierlichen Bestimmung der Orientierung des Objekts während des Objektlernens, kommt häufig großes Verschieben zwischen der realen und berechneten Lage des Objekt in einem langen Bilderstrom vor, obwohl es nur winziger Fehler zwischen jedem zwei Bildern gibt. In dieser Arbeit wird das Kalman-Filter über die Translation des Schwerpunkts von Objekt benutzt, um die kumulative negative Wirkung jedes Schritts zu verdecken. Die Grundlage des Kalma-Filters wird im Abschnitt 3.3.3 aufgeführt. Die Übergangsmatrix des Schwerpunkts des Objekts im 3D Raum kann definiert als:

$$F = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

was die Lage und Geschwindigkeit der Bewegung gleichzeitig betrachtet. Die Beobachtungsmatrix ist die Einheitsmatrix mit Größe  $3 \times 3$ , weil der Schwerpunkt des Objekts  $\in R^{3 \times 1}$  als Eingabe zur Korrektorschleife eingegeben wird. OpenCV liefert eine komplette Implementierung über Kalman-Filter, was hier direkt verwendet wird.

### 4.2.2 Markenanordnung

Wie im Anfang des Abschnittes 4.2.1 geschrieben hat, soll während der Lernensphase die Lage betrachtetes Objekts in jedem Bild bestimmt werden. Es benötigt zumindest in zwei Bildern jeweils 3 nicht kollineare Punkten des Objekts, damit die Transformation zwischen der Lage dieses Objekts in diesen zwei Bildern bestimmt werden kann. Je mehr die Punkte betrachtet werden, desto exaktere Transformation in der Theorie berechnet werden kann. Aber wegen der Auflösung der Kamera in dieser Arbeit wird die Größe der Marken bzw. die Abstände der Marken stark beschränkt. D.h. es ist unmöglich, beliebig viele Marken an dem Objekt anzubringen. Deshalb soll eine Strategie für die Anordnung der Marken entworfen werden, damit die beste Transformation mit möglich wenigen Marken bestimmt werden kann.

Die Anordnungsstrategie wird in 5 Regeln zusammengefasst.

1. Die Größen der Marken dürfen nicht zu klein sein.
2. Die Abstände zweier Marken sollen nicht zu eng.
3. Jedes Tripel der Marken soll nicht kollinear sein.
4. Um Grenze zweier Ebenen des Objekts sollen mehr Marken als in Mitte angebracht werden.
5. Der Strukturgraph jeder Ebene soll möglich unsymmetrisch sein.

Die erste und zweite Regel hängen direkt von der Auflösung der Kamera ab, was schon im Abschnitt 3.1.2.3 diskutiert wurde. Die dritte Regel garantiert, dass die Orientierung des Objekts durch beliebigen drei Marken immer berechnet werden kann. Was in Regel 1 definierte minimale erkennbare Markengröße entspricht aber nur die Situation,

dass die Marken mit der Bildebene der Kamera parallel sind. Wenn ein Objekt sich umdreht, verändert der Winkel zwischen der aktuellen beobachteten Ebene und der Kamera, wodurch werden die Marken auf dem Ebene zum kleineren Bereich auf dem Bild abgebildet. Die Größen der Marken auf dieser Ebene scheinen immer kleiner in dem Bild mit der Vergrößerung des Drehwinkels, bis sie komplett senkrecht zu der Bildebene liegen und die nächste Ebene vollständig unter Kamera vorkommt. Je weniger die Marken sind, desto schwieriger erkannt werden. Deshalb sollen mehr Marken im Bereich zwischen zwei benachbarten Ebenen angebracht werden, damit das Programm genug Marken für die Berechnung der Orientierung erhalten kann. Das ist genau was in Regel 4 erklärt wird. Die Asymmetrie von Regel 5 hält die Einzigartigkeit der Transformation zwischen zwei unterschieden Lagen des Objekts, d.h. nur eine Lösung wird von dem Orientierungsverfahren bestimmt.

### 4.2.3 Darstellung des Strukturgraphen

Das Ziel des Lernens ist, dass ein Strukturgraph für jedes eingegebenen Objekt erzeugt wird, welche als die Charakteristiken für Wiedererkennung verwendet werden können. Um das Ziel zu erreichen, sollen die Strukturgraphen stabil sein und genug charakteristische Information des Objekts enthalten. Die Stabilität bedeutet, dass der Strukturgraph für ein Objekt während viel mal Lernen gleich dargestellt werden sollen. Die Information, die in Wiedererkennung benötigt ist, wird durch die Positionen der Marken bzw. die Abstände dazwischen erzeugt. Die Verfahren der Bestimmungen der charakteristische Marken und Kanten werden in folgenden Abschnitten diskutiert.

#### 4.2.3.1 Bestimmung der stabilen Knoten

Nach der Markenerkennung werden viele Marken aus dem Bild erkannt, die aber viel Rauschen enthalten. Das Rauschen wird von den falschen erkannten Merkmalen des Objekt bzw. der Umgebung oder den schlechten Korrespondenzpunkten verursacht. Deshalb benötigt das Programm eine Strategie für die Auswahl der gültigen Marken. In dieser Arbeit wird die Auswahl auf den Erscheinungshäufigkeiten basiert, d.h. nur die Marken, die vielmals kontinuierlich vorgekommen sind, werden zu den gültigen Marken erkannt und in den Strukturgraphen eingefügt. Diese Marken werden als stabilen Knoten in dem Strukturgraphen markiert, und dürfen nicht verändert werden. Außer der Beschränkung der Anzahl der Erscheinungen soll aber auch der Begriff von „Identität“ zweier Marken in unterschiedlichen Bildern definiert werden. Zwei Marken von verschiedenen Bildern sind identisch, genau dann, wenn der Abstand von einer Marke zu dem Punkt, der von anderer Marke nach Transformation erzeugt wird, kleiner als einen vorgegebenen Schwellenwert ist. Die Transformation umfasst die Rotation- bzw. Translationsmatrix, die mit dem Verfahren vom Abschnitt 4.2.1 bestimmt werden.

### 4.2.3.2 Kanteneinfügung

Die ungerichteten Kanten von Strukturgraphen speichern die Abstände zwischen den Knoten des Graphen, was die wichtigste Eigenschaft für die Wiedererkennung ist. Die Knoten, die gleichzeitig beobachtet werden können, werden in einem vollständigen Graph mit den anderen verbunden. Wenn irgendwann ein neuer Knoten in den Graphen eingefügt wird, verbindet der mit allen vorhandenen Knoten des Graphen. Der analoge Durchlauf wird während der Entfernung eines ungültigen Knotens aus dem Graphen durchgeführt. Der Algorithmus 9 zeigt, wie der aktuelle Strukturgraph mit neu eingegebenen Punkten aktualisiert.

---

**Algorithm 9** GraphUpdate(*Punkte P, Mat R, Mat T*)

---

Schwellenwert des Abstand:  $\epsilon$

Minimale Lebenszeit des stabilen Knoten:  $minT$

Aktueller Graph:  $G$

$G_{temp} \leftarrow createCompleteGraph(P)$

**for** jeder Knoten  $v_i \in G$  **do**

$v_i \leftarrow Rv_i + T$

**end for**

**for** jeder Knoten  $v_i \in G$  **do**

**for** jeder Knoten  $v_{temp_j} \in G_{temp}$  **do**

**if**  $DistanceOf(v_i, v_{temp_j}) < \epsilon$  **then**

$v_i.Lifetime \leftarrow v_i.Lifetime + 2$

**if**  $v_i.Lifetime > minT$  **then**

$v_i.isFixed \leftarrow True$

**end if**

Verbinden allen mit  $v_{temp_j}$  verbundenen Knoten in  $G_{temp}$  mit  $v_i$

Entfernen  $v_{temp_j}$  aus  $G_{temp}$

**end if**

**end for**

**if** Keiner entsprechende Knoten für  $v_i$  aus  $G_{temp}$  gefunden wird **then**

$v_i.Lifetime \leftarrow v_i.Lifetime - 1$

**if**  $v_i.Lifetime < 0$  und  $!v_i.isFixed$  **then**

Entfernen  $v_i$  aus  $G$

**end if**

**end if**

**end for**

Fügen allen übrigen Knoten von  $G_{temp}$  in  $G$

---

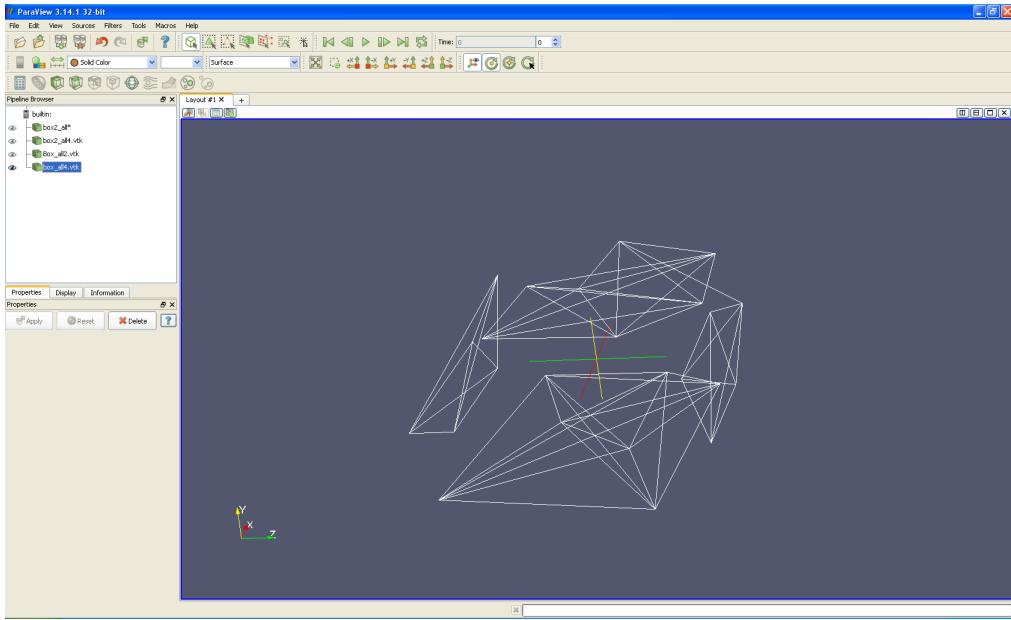


Abbildung 4.6: Visualisierung eines Strukturgraphen des Kästchens mit ParaView.

### 4.3 Zugriff des Strukturgraphen von Datei

Das Ergebnis des Objektlernens ist ein Strukturgraph, der die Charakteristiken des betrachteten Objekts beschreibt. Die Knoten des Graphen werden als die 3D Punkten mit Gleitkommazahl definiert. Jeder Knoten enthält ein Map, wo seine Nachbarn und die Kantenlänge dazwischen paarweise gespeichert werden. Der Zugriff des Graphen von Datei soll dann implementiert werden, damit die Lernensergebnisse in der Festplatte gespeichert werden können. In dieser Arbeit wird die Dateiform von VTK ([[Visualization Toolkit](#)]) benutzt, was als eine Open-Source-C++-Klassenbibliothek für die 3D Computergraphik und wissenschaftliche Visualisierung häufig verwendet wird. Der Vorteil der Verwendung der VTK Datei ist, dass die Ergebnisgraphen einfach weiter von den anderen Framework bzw. Software verwendbar sind. Das Abbildung 4.6 ist das Screenshot von der Visualisierung eines Strukturgraphen mit Software ParaView [[ParaView](#)]. Vor dem Speichern des Graphen, werden alle Knoten noch mal überprüft. Die ähnlichen Knoten sollen als einen einzigen Knoten kombiniert werden, damit nur ein vereinfachter Graph ohne verdoppelten Knoten gespeichert wird. VTK-Dateiform liefert vielen Stichwörter für Beschreibung der unterschiedlichen grundsätzlichen geometrischen Elemente, z.B. die Punkte, die Gerade und die Ebene usw.. Deshalb werden die Knoten und die Kanten des Strukturgraphen mit jeweils dem Stichwort „POINTS“ und „LINES“ in der VTK-Datei geschrieben. Diese Stichwörter sind auch die Kennzeichen bei dem Lesen der VTK-Datei. Die Punkte werden zuerst in einen neuen Graphen eingefügt, und verbindet dann mit anderen Knoten durch die gespeicherten Gerade.

## 4.4 Objekterkennung und Verfolgung

Die Strukturgraphen aller von Programm erlernten Objekte sollen am Anfang der Wiedererkennungsphase von den in der Festplatte gespeicherten VTK-Dateien wieder zum Programm eingelesen werden. Danach wird die Markenanalyse genau wie in der Lernensphase durchgeführt und wird am Ende eine Liste von Punktmenge erzeugt. Das Programm versucht dann, die Teilgraphen dieser Strukturgraphen aus die von der Punktmengen neu darstellenden Graphen zu finden, was zum sogenannten Teilgraph-Isomorphismus-Problem zusammengefasst wird. Für einziges Bild werden alle mögliche Kombination der Strukturgraphen und Punktmengen dieses Bildes durchgesucht. Die Objekte, deren Teilgraphen aus dem Eingabebild gefunden wurden, werden als erkannt genannt. Die entsprechenden Punktmengen im Eingabebild beschreiben nun die aktuellen Lagen der Objekte. Mit dem gleichen Ablauf können die Objekte in einem Bilderstrom kontinuierlich erkannt werden, aber was nicht möglich ist, die entsprechenden Punktmengen kontinuierlich festzulegen, weil die Marken von jedem Bild neu segmentiert werden und keine Verbindungen zwischen den „gleichen“ Punktmengen von unterschiedlichen Bildern existieren. Ein direkter Lösungsverfahren ist, dass jede Punktmenge aus Segmentierung sofort als ein neues Objekt erkannt wird. Der Vergleich von Teilgraph-Isomorphismus-Problem wird dann zwischen die Strukturgraphen zweier Objekte stattgefunden. Dadurch sollen viele Lernensprozesse während der Wiedererkennungsphase gleichzeitig durchgeführt werden, was kostet aber zu viele Zeit und ist schwierig zu implementieren. Die alternative Lösung ist, die sogenannte Kandidaten zu verwenden, in der nur die Abhängigkeiten der gleichen Punktmengen von unterschiedlichen Bildern gespeichert aber kein einziger Strukturgraph dargestellt werden.

### 4.4.1 Kandidaten der Objekterkennung

Wie im Abschnitt 4.1.4 geschrieben werden viele Punktmengen nach der Segmentierung erzeugt, welche mit unterschiedlichen Objekten übereinstimmen können. Die Kandidaten werden aus diesen Punktmengen bestehen und in einer Liste im Speicher gespeichert. Wenn das Programm ein neuen Eingabebild einliest, werden alle neue von Segmentierung erhaltenen Punktmengen mit vorhandenen Kandidaten verglichen. Der Kandidat, der mit Einer der neuen Punktmengen assoziiert, wird dann aktualisiert. Die übrige Punktmengen, die keine entsprechenden Kandidaten finden können, werden als neuen Kandidaten in der Liste eingefügt. Um die Leistungsfähigkeit zu halten, sollen die Kandidaten, die in langer Zeit nicht aktualisiert wurden, aus der Liste entfernt werden. Die Erkennung werden dann für jeden neu aktualisierten Kandidat durchgeführt. Der Index des am besten entsprechenden Objekts wird im Kandidat gespeichert.

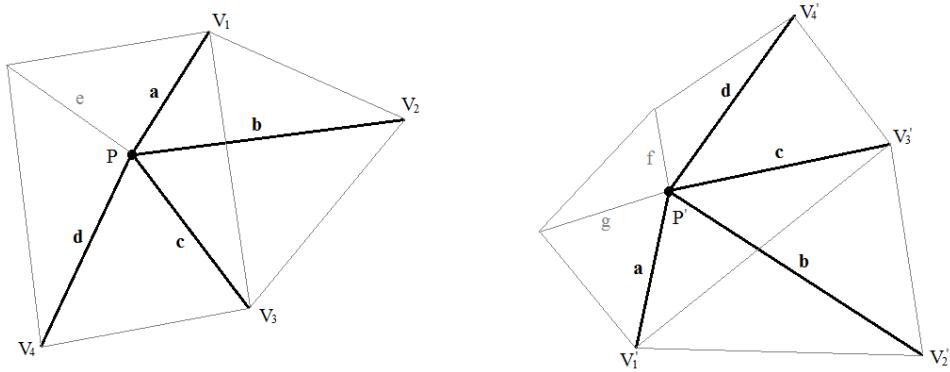


Abbildung 4.7: Ein Beispiel der isomorphen Knoten.  $P$  und  $P'$  sind zwei isomorphe Knoten, die jeweils vier Nachbarn mit gleicher Abstände haben.  $((P, V_1) = (P', V'_1), (P, V_2) = (P', V'_2), (P, V_3) = (P', V'_3), (P, V_4) = (P', V'_4))$

#### 4.4.2 Objekterkennung

Die Objekterkennung basiert auf dem Vergleich zwischen den Strukturgraphen der erlernten Objekte und den Kandidaten, die von den neu segmentierten Punktmenigen erzeugt werden. Ein neuer Graph wird zuerst von den Punkten eines Kandidaten dargestellt. Der Vergleich zwischen diesem Kandidat und den vorhandenen Strukturgraphen können dann zum Teilgraph-Isomorphismus Problem abgleitet werden. Der Lösungsverfahren übernimmt die Idee von Rhijn und Mulder [Rhijn & Mulder 2005], was im Abschnitt 3.6.2 erklärt wurde. Die isomorphen Knoten zwischen zwei Graphen werden zuerst herausgefunden, welche jeweils genug isomorphen Nachbarn haben, die mit gleichen Abständen zu den isomorphen Knoten liegen. Ein Beispiel der isomorphen Knoten wird in Abbildung 4.7 gezeigt. Der Teilgraph-Isomorphismus zwischen diesen Graphen existiert, genau dann, wenn mindesten ein gefundener isomorpher Knoten und seine Nachbarn zusammen die gleichen Pyramiden darstellen können. In dieser Arbeit wird der Verfahren von Rhijn und Mulder in zwei Bereichen verbessert: die effizientere Nebenbedingung für die Suchung der isomorphen Knoten und die vereinfachte Endbedingung statt Feststellung der Pyramide.

##### Nebenbedingung für die Suchung der isomorphen Knoten

Um die isomorphen Knoten schneller zu finden, werden zwei Quoten für die Suchung definiert. Die Abstandquote wird statt dem absoluten Schwellenwert benutzt, damit der Vergleich der Distanzen zwischen dem betrachteten Knoten und seiner Nachbarn bewertet werden kann. Offensichtlich wird die Kante zwischen zwei Marken auf dem Objekt zu mehr Bildpunkten abgebildet, wenn das Objekt zu der Kamera bewegt. Je größer die Auflösung des Objekt ist, desto das bessere Erkennungsergebnis von dem

Programm erzeugt werden kann. Deshalb lässt die Verwendung der Abstandquote die Genauigkeit der Erkennung nicht von der Distanz zwischen dem Objekt und der Kamera beeinflussen.

Die zweite Nachbarquote beschränkt die minimale Anzahl der nötigen assoziierten Nachbarn für Bestimmung eines isomorphen Knotens, was im Verfahren von Rhijn und Mulder aber als Invariante definiert wird. Die Beziehung zwischen der Nachbarquote und der minimalen Anzahl der Nachbarn kann in

$$N_{min} = \begin{cases} 3 & \text{if } \|V\| * \text{Nachbarquote} < 3 \\ \|V\| * \text{Nachbarquote} & \text{sonst} \end{cases} \quad (4.1)$$

formuliert werden, wobei  $V$  die Knotenmenge des Graphen beschreibt. Die minimale Anzahl der Nachbarn darf nicht kleiner als 3 sein, weil es mindesten 3 Nachbarn benötigt, um einziger Knoten eines Graphen durch Vergleich der Kantenlänge seiner Nachbarn im 3D Raum zu bestimmen. Diese Veränderung liefert eine bessere Toleranz für die Erkennung: wenn die Graphen mit nur wenigen Knoten als die Eingaben dem Verfahren eingegeben werden, garantiert der Verfahren aber auch genug Nachbarn, um die isomorphen Knoten auszuwählen; außerdem können die Genauigkeit und Stabilität der Teilgraph-Isomorphismus mit dem Vergrößern der Anzahl der Knoten auch ansteigen.

### Vereinfachung der Endbedingung

Im Verfahren von Rhijn und Mulder sollen schließlich die Pyramiden für den isomorphen Knoten und seine Nachbarn gefunden werden, damit der Isomorphismus bestimmt werden kann. Die Pyramide ist ein vollständiger Graph mit 4 Knoten, deshalb für einen Knoten, der  $n$  Nachbarn hat, sollen höchstensfalls

$$C_n^3 = \binom{n}{3} = \frac{n!}{k!(n-k)!} = \frac{1}{6}n(n-1)(n-2) \approx \mathcal{O}\left(\frac{1}{6}(n^3 - 3n^2)\right)$$

verschiedenen Kombinationen seiner Nachbarn betrachtet werden, um die Pyramide zu finden. Seien  $m$  die Anzahl der gefundenen isomorphen Knoten und der Zeitaufwand von Überprüfung der Nachbarschaft der Einheitszeitwand  $\mathcal{O}(1)$ , dann ist der Zeitaufwand der Suchung der Pyramide in den schlechtesten Fall  $\mathcal{O}(\frac{1}{6}m(n^3 - n^2))$ . Offensichtlich gilt es  $m < n$  wegen der Definition des isomorphen Knoten. Die Idee der Vereinfachung ist, statt der Nachbarn eines isomorphen Knoten aber die isomorphen Knoten selbst zu betrachten, um den obigen schlechtesten Fall zu vermeiden. Die Bedingung für die Pyramide kann auch geschwächt werden, d.h. man kann die einfacheren geometrischen Elemente benutzen, z.B. das Dreieck sogar eine Ecke mit drei Knoten aber nur zwei Kanten.  $m$  isomorphe Knoten werden durchgesucht, um zu bestimmen, ob sie zumindest mit zwei anderen isomorphen Knoten verbinden. Der Zeitaufwand dieses Ablaufs ist nur  $\mathcal{O}(m^2)$ , was deutlich kleiner als den Verfahren mit Suchung der Pyra-

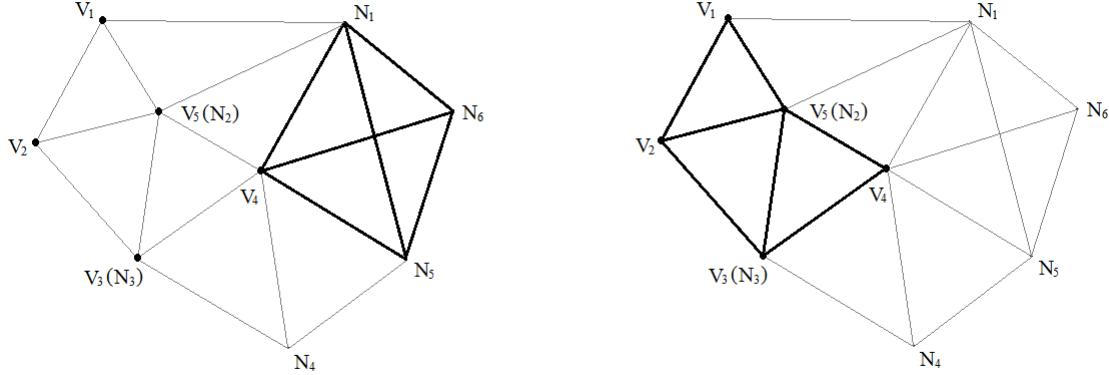


Abbildung 4.8: Ein Beispiel für den Vergleich der zwei Endbedingungen.  $V_1 \sim V_4$  sind die isomorphen Knoten und  $N_1 \sim N_6$  sind die Nachbarn von Knoten  $V_4$ . An der linken Seite wird der Verfahren mit Pyramide gezeigt, wobei alle isomorphen Knoten durchgesucht werden, bis eine Pyramide  $V_4N_5N_6N_1$  gefunden ist. An der rechten Seite werden aber nur die Kanten zwischen den isomorphen Knoten betrachtet.

mide. Abbildung 4.8 zeigt ein Beispiel für den Vergleich der beiden Nebenbedingung des Isomorphen-Problems. An der linken Seite versucht das Programm, eine Pyramide an den isomorphen Knoten zu finden. Alle Kombinationen der drei Nachbarn werden für jeden isomorphen Knoten betrachtet, dadurch ist der Zeitaufwand der linken Seite:

$$\binom{3}{3}(V_1) + \binom{3}{3}(V_2) + \binom{4}{3}(V_3) + \binom{6}{3}(V_4) + \binom{5}{3}(V_5) = 1 + 1 + 4 + 20 + 10 = 36.$$

An der rechten Seite werden aber nur die isomorphe Knoten selbst betrachtet, wobei  $m^2 = 5^2$  Einheitszeit kostet. Alle isomorphen Knoten, die diese Bedingung erfüllen, werden dann gespeichert, um die Orientierung des erkannten Objekts zu berechnen.

#### 4.4.3 Bestimmung der Orientierung

Verfolgung des Objekts ist die andere wichtige Aufgabe dieser Arbeit, was durch die Berechnung der Orientierung realisiert wird. Die Anfangslage eines Objekts wird von dem Strukturgraphen definiert. In jedem Bild werden die Rotation- bzw. Translationsmatrix zwischen dem Anfangs- und aktuellen Zustand mithilfe der ausgewählten isomorphen Knoten bestimmt. Der Orientierungsverfahren läuft gleich wie den Verfahren im Lernensprozess, der im Abschnitt 4.2.1 erklärt wurde.

## 4.5 Bildersteuerung

Speichern und Auswahl der Bildern sind die Hauptaufgaben des Teilprogramm der Bildersteuerung. Die Verfolgung der Marken wird durch den Vergleich der Marken von zwei nachfolgenden Bildern realisiert, d.h. soll das Programm immer zwei Bilder gleichzeitig betrachten: das aktuelle- und das historische Bild. Außerdem wegen Rauschen und andere Störung aus den Eingabebildern, werden manchmal die Marken total schlecht erkannt. Die falschen Marken verstören die Korrespondenzuntersuchung und verschlechtern weiterhin die Orientierung der Objektslagen zwischen zwei Bildern, was aber ganz wichtig für das Lernen des Objekts. Wenn irgendwelche kleinen Fehler in der Transformation vorkommen, wird der Strukturgraph des lernenden Objekts komplett andere dargestellt. Deshalb sollen die schlechten Eingabebilder vor dem Darstellung des Strukturgraphen ausgewählt und von dem Bilderstrom entfernt werden.

In dieser Arbeit werden allen Eingabebilder zuerst in einer Warteschlange gespeichert. Die Warteschlange wird mit fester Größe definiert, und wenn neues Bild eingegeben wird, wird es im Ende der Schlange eingefügt und das älteste Bild aus die erste Stelle der Schlange gestrichen. In jedem Zyklus des Programms werden das erste und letzte Bild verglichen und die korrespondierenden Punktpaare daraus gefunden. Mit anderen Worten definiert die Größe der Warteschlange aber auch das Intervall der betrachteten Bilder. Die Markenanalyse und die Orientierung von Objektlernen werden regelmäßig für jedes Eingabebild durchgeführt, aber die davon erhaltenen Rotation- bzw. Translationsmatrix werden nicht direkt für die Aktualisierung des Strukturgraphen benutzt, sondern zuerst von anderem Teilprogramm geprüft werden, damit die Qualität der Transformation bewerten werden kann.

Das Prüfungsprogramm besteht aus zwei Teilen. Der erste Teil ist ein Prüfer, der eine boolesche Aussage liefert, ob aus dem Bild genug hochqualitativen Marken gefunden werden können. Der Prüfer läuft analog wie die Aktualisierung der Knoten von dem Teilprogramm des Strukturgraphen, was im Abschnitt 4.2.3.1 beschrieben wurde. Der Unterschied liegt daran, dass der Prüfer die Lebenszeit der Marken nicht betrachtet, aber nur die neu gefundenen Marken zählen, die mit den vorhandenen Knoten des Strukturgraphen nach gerade berechneter Transformation identisch sind. Nur das Bild mit genug Marken, die obige Bedingung erfüllen, wird von dem Prüfer akzeptiert. Die Beurteilung über die Anzahl der Marken wird durch den Vergleich der Quote, die der Anteil der mit vorhandenen Knoten übereinstimmten Marken an allem neuen Marken beschreibt, mit einem vordefinierten Schwellenwert realisiert. Die von Prüfer akzeptierten Bilder können direkt für die Aktualisierung der Strukturgraphen benutzt. Die anderen Bilder sind überflüssig, können aber nicht direkt alles geschmissen werden. Unter einer total schlechten Beobachtungssituation liefert die Kamera möglicherweise in langer Zeit gar keine hochqualitativen Bilder. Wenn das Programm alle diesen Bilder überspringt, wird die Verfolgung des Objekts abgebrochen und falschen Erkennungsergebnisse erzeugt. Ein extremes Beispiel wird in der Umdrehung des Kästchens

gefunden. Wenn nur die Bilder von zwei benachbarten Ebenen des Objekts von dem Programm akzeptiert aber die Bilder des Rotationsablaufs dazwischen nicht berücksichtigt werden, hat das Programm aber gar keine Möglichkeit, diese zwei Ebenen zu unterscheiden. Sie werden als eine große Ebene erkannt und gespeichert. Der zweite Teil des Prüfungsprogramms vermeidet diese Situation. Die maximale Anzahl der übersprungenen Bilder wird von dem beschränkt. Wenn kein hochqualitatives Bild gefunden werden kann, wird ein verhältnismäßig besseres Bild aus die schlechten Bilder ausgewählt. Die Summe aller größten Elemente von  $P$ , die im Abschnitt **Qualitätsmanagement der Markenverfolgung** von 4.1.3 erklärt wurde, wird hier als die Bewertungsvariable verwendet. Der Durchlauf des Prüfungsprogramms wird im Algorithmus 10 aufgeführt.

---

**Algorithm 10** *Bilderprüfer(*Strukturgraph*, *Eingabebild*, *UebersprungeneAnzahl*)*

---

die minimale Quote der mit vorhandenen Knoten übereinstimmten Marken:  $q$   
 die maximale Anzahl der Bilder, die übersprungen werden dürfen:  $N_{jump}$   
 Iterator des Besten Bild in *Bildschlange*:  $i_b$

```

if Korrespondenzuntersuchung() == True und IdentischeQuote >  $q$  then
  Aktualisieren Strukturgraph mit Eingabebild
else
  if Korrespondenzuntersuchung() == False then
    Überspringen
  end if
  if Eingabebild.SumP > Bildschlange[ $i_b$ ].SumP then
    Entfernen Bildschlange[ $i_b$ ]
     $i_b \leftarrow Eingabebild.iterator$ 
  else
    Überspringen
  end if
  UebersprungeneAnzahl ++
  if UebersprungeneAnzahl >  $N_{jump}$  then
    Aktualisieren Strukturgraph mit Bildschlange[ $i_b$ ]
    UebersprungeneAnzahl  $\leftarrow 0$ 
  end if
end if

```

---

# **Kapitel 5**

## **Experimentelle Auswertung**

### **5.1 Teilweise Evaluation**

asdadasdasdad

#### **5.1.1 Abstand Filter**

skdakljdkasdjklasdjaskljlaksdjladjladjakljdlajdakl

#### **5.1.2 Helligkeitssteuerung**

#### **5.1.3 Verbesserung der Singulärwertzerlegungsverfahren**

#### **5.1.4 Aktualisierung des Strukturgraphen**

#### **5.1.5 Teilgraph-Isomorphismus**

#### **5.1.6 Bildersteuerung**

### **5.2 Globale Evaluation**

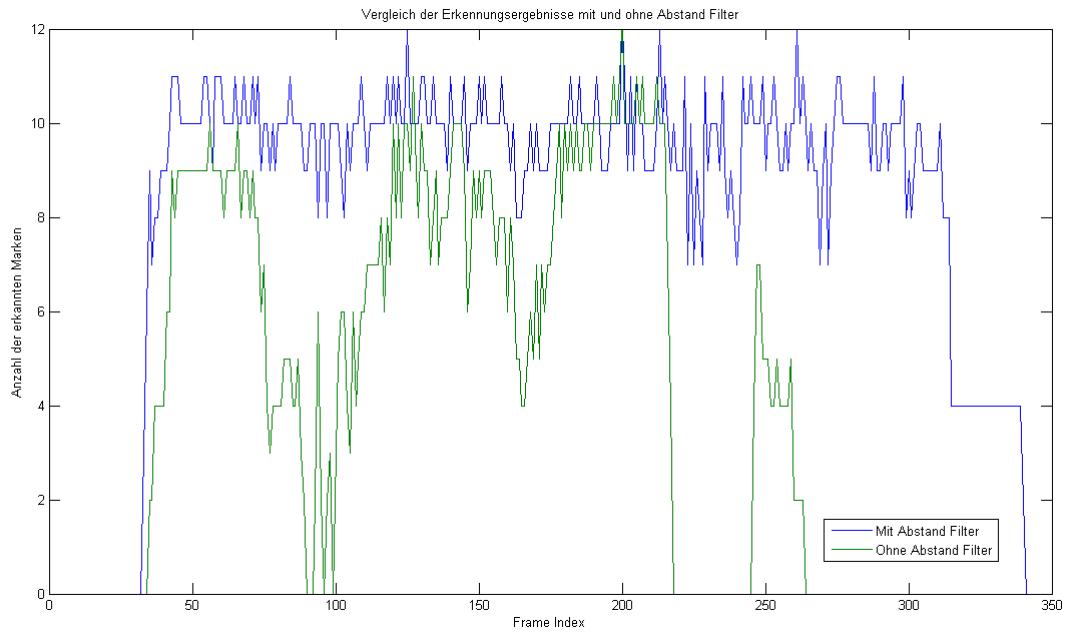


Abbildung 5.1: Vergleich der Erkennungsergebnisse mit und ohne Abstand Filter. Die Eingabebilder enthalten nur ein Objekt.

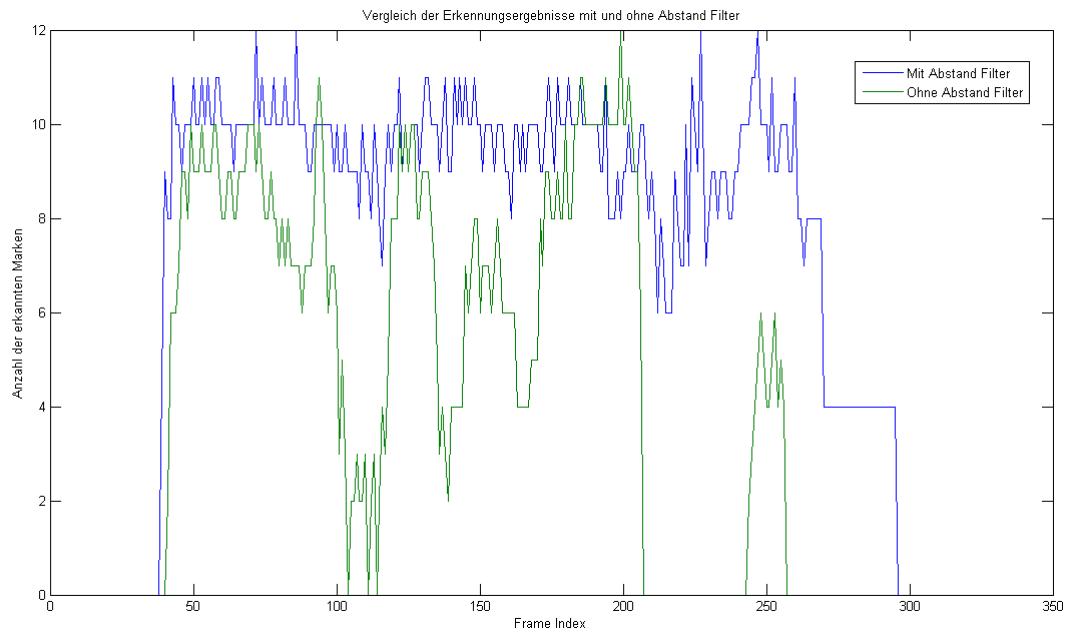


Abbildung 5.2: Vergleich der Erkennungsergebnisse mit und ohne Abstand Filter. Die Eingabebilder enthalten mehr Objekte.

# Kapitel 6

## Schlusswort

Diese Arbeit implementiert die auf Marker basierte Objekterkennung bzw. Verfolgung im Grund auf dem Rahmenwerk MAROCO. Die Oberflächen der Zielobjekt werden mit retroreflektierenden Marker markiert. Eine PMD-Kamera beobachtet die ganze Szene von oben und liefert direkt die 3D-Daten. Die totale Laufzeit kann in zwei Phasen zusammengefasst werden. In der Initialisierungsphase wird die Fremdobjekt unter der Kamera gezeigt und die Marker darauf sollen herausgekannt und im System gespeichert werden. Wenn es mehr Objekte gibt, wird eine Kalibrierung am Anfang durchgeführt. Mills in [Mills & Novins 2000] hat eine kompakte Segmentierung der Bewegung mithilfe des sogenannten „Feature Interval Graph“ dargestellt. [Rhijn & Mulder 2005] erweitert die Arbeit von Mills. Ein auf Pyramide basiertes Clustering-Verfahren wird statt des alten auf Dreiecke basierten Clustering-Verfahren vorgeschlagen. Nach der Initialisierungsphase wird das Objekt aus dem Gesichtsfeld der Kamera verschoben. Die Erkennungsphase fängt genau an, wenn das gleiche Objekt wieder unter der Kamera eingebracht wird. Die reflektierende Marker sollen nochmal gesammelt werden und ein von [Rhijn & Mulder 2005] repräsentierter Teilgraph-Tracker wird dann implementiert, um die Objekt zu kennen und die Position zu bestimmen.



# Literaturverzeichnis

- [Agrawal, Konolige & Blas 2008] M. Agrawl, K. Konolige and M. Blas: *CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching*. in Computer Vision-ECCV 2008, pp.102-115, Springer Berlin/Heidelberg (Zitiert auf Seiten 7, 20 und 22)
- [ARToolKit] ARToolKit <http://www.hitl.washington.edu/artoolkit/> Zuletzt besucht: 02.12.2011 (Zitiert auf Seite 4)
- [Arun, Huang & Blostein 1987] K.S. Arun, T.S. Huang and S.D. Blostein: *Least-squares fitting of two 3-D point sets*. IEEE Trans Pattern Anal Machine Intell (1987) 9:698?700. (Zitiert auf Seite 8)
- [Bay et al. 2006] H. Bay, A. Ess, T. Tuytelaars and L.V. Gool: SURF:Speeded Up Robust Features. European Conference on Computer Vision, 2006 (Zitiert auf Seite 6)
- [Besl & McKay 1992] P. Besl and N. McKay: *A Method for Registration of 3-D Shapes*. Trans. PAMI, Vol. 14, No. 2, 1992. (Zitiert auf Seite 8)
- [Black & Yacoob 1997] M.J. Black and Y. Yacoob: *Recognizing Facial Expressions in Image Sequences Using Local Parameterized Models of Image Motion* International Journal of Computer Vision 25(1), 23-48 (1997) (Zitiert auf Seite 5)
- [Chavarria & Sommer 2007] M.A. Chavarria and G. Sommer: *Structural ICP algorithm for pose estimation based on local features*. Computer Vision Theory and Applications - VISAPP , pp. 341-346, 2007 (Zitiert auf Seite 8)
- [Chen & Medioni 1991] Y. Chen, G. Medioni: *Object Modeling by Registration of Multiple Range Images*. Proc. IEEE Conf. on Robotics and Automation, 1991. (Zitiert auf Seite 8)
- [Conte et al. 2004] D. Conte, P. Foggia, C. Sansone and M. Vento: *Thirty years of graph matching in pattern recognition*. International Journal of Pattern Recognition and Artificial Intelligence Vol. 18, No. 3 (2004) 265-298 (Zitiert auf Seite 10)
- [Cordella et al. 2004] L.P. Cordella, P. Foggia, C. Sansone and M. Vento: *A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs*. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 26, NO. 10, OCTOBER 2004 (Zitiert auf Seite 11)
- [DBSCAN Wiki] DBSCAN Wikipedia: <http://de.wikipedia.org/wiki/DBSCAN> Zuletzt besucht: 11.05.2012 (Zitiert auf Seite 35)

- [Debevec & Malik 2008] P.E. Debevec and J. Malik: *Recovering High Dynamic Range Radiance Maps from Photographs* ACM SIGGRAPH 2008 (Zitiert auf Seite 43)
- [Dorai, Weng & Jain 1997] C. Dorai, J. Weng and A.K. Jain: *Optimal registration of object views using range data.* IEEE Transactions on Pattern Analysis and Machine Intelligence. 19(10):1131-1137, 1997 (Zitiert auf Seite 8)
- [Eggert, Lorusso & Fisher 1997] D.W. Eggert, A. Lorusso, R.B. Fisher: *Estimating 3-D rigid body transformations: a comparison of four major algorithms.* Machine Vision and Applications (1997) 9: 272-290. (Zitiert auf Seiten 7 und 8)
- [Eppstein 1999] D. Eppstein: *Subgraph Isomorphism in Planar Graphs and Related Problems.* Journal of Graph Algorithms and Applications, vol. 3, no. 3, pp. 1?27 (1999) (Zitiert auf Seite 11)
- [Ester et.al 1996] M. Ester, H.P. Kriegel, J. Sander and X. Xu: *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Knowledge Discovery and Data Mining. Seite 226-231, 1996 (Zitiert auf Seiten 9 und 32)
- [Haag & Nagel 1999] M. Haag and H. Nagel: *Combination of Edge Element and Optical Flow Estimates for 3D-Model-Based Vehicle Tracking in Traffic Image Sequences.* International Journal of Computer Vision, Volume 35, Number 3, 295-319, 1999 (Zitiert auf Seite 5)
- [Harris 1992] C. Harris: *Tracking with Rigid Objects.* MIT Press. 1992 (Zitiert auf Seite 4)
- [Harris & Stephens 1988] C. Harris and M. Stephens: *A combined corner and edge detector.* Alvey Vision Conference, pp.147-151, 1988 (Zitiert auf Seite 6)
- [Horn & Schunck 1981] B.K.P. Horn and B.G. Schunck: *Determining optical flow.* Artificial Intelligence Volume 17, Issues 1?3, August 1981, Pages 185-203 (Zitiert auf Seite 5)
- [Horn 1987] B.K.P. Horn: *Closed-form solution of absolute orientation using unitquaternions.* J. Opt. Soc. Am. A, vol. 4, 629-642, 1987. (Zitiert auf Seiten 8, 29, 32 und 48)
- [ICP Wiki] ICP Wikipedia [http://de.wikipedia.org/wiki/Iterative\\_Closest\\_Point\\_Algorithm](http://de.wikipedia.org/wiki/Iterative_Closest_Point_Algorithm) Zuletzt besucht: 02.02.2012 (Zitiert auf Seite 8)
- [Josiger & Kirchner 2003] M. Josiger and K. Kirchner: *Moderne Clusteralgorithmen - eine vergleichende Analyse auf zweidimensionalen Daten.* Proc. FGML Workshop (FGML 2003), S.80-84, Karlsruhe 2003 (Zitiert auf Seite 9)
- [Jurie & Dhome 2001] F. Jurie and M. Dhome: *A simple and efficient template matching algorithm.* International Conference on Computer Vision (ICCV 01) 2 (2001) 544-549. (Zitiert auf Seite 5)
- [Kinect] Kinect <http://www.xbox.com/en-US/kinect> Zuletzt besucht: 02.12.2011 (Zitiert auf Seiten 3 und 14)
- [Kinect-How it works] Kinect-How it works [http://www.caedt.at/wp-content/uploads/2011/02/kinect\\_tech.pdf](http://www.caedt.at/wp-content/uploads/2011/02/kinect_tech.pdf) Zuletzt besucht: 15.03.2012 (Zitiert auf Seiten 14 und 15)

- [Kanatani 1994] K. Kanatani: *Analysis of 3-D rotation fitting*. IEEE Trans Pattern Anal Machine Intell (1994) 16:543-549 (Zitiert auf Seite 8)
- [Karypis et.al 1999] G. Karypis, E.U. Han and V. Kumar: *CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling*. IEEE Computer, 32:8, S.68-75, 1999 (Zitiert auf Seite 9)
- [Lamdan et al. 1988] Y. Lamdan, J.T. Schwatrtz and H.J. Wolfson: *On recognition of 3-D objects from 2-D images*. In Proceedings of IEEE International Conference on Robotics and Automation, 1988 (Zitiert auf Seite 10)
- [Lepetit & Fua 2005] V. Lepetit and P. Fua: *Monocular Model-Based 3D Tracking of Rigid Objects: A Survey*. Foundations and Trends in Computer Graphics and Vision Vol.1, No 1(2005) 1-89. (Zitiert auf Seiten 3, 4 und 5)
- [Lowe 2004] D.G. Lowe: *Distinctive Image Features from Scale-Invariant Keypoints*. Proceedings of the International Conference on Computer Vision. 2. pp. 1150?1157, 2004 (Zitiert auf Seite 6)
- [Lucas & Kanade 1981] B.D. Lucas and T. Kanade: *An Iterative Image Registration Technique with an Application to Stereo Vision*. Proceedings of Imaging Understanding Workshop, pp. 121-130 (1981). (Zitiert auf Seite 5)
- [Messmer & Bunke 1998] B.T. Messmer and H. Bunke: *A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection*. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 20, NO. 5, MAY 1998 (Zitiert auf Seite 11)
- [Mills & Novins 2000] S. Mills and K. Novins: *Motion Segmentation in Long Image Sequences*. In Proceedings of the British Machine Vision Conference 2000, pp.162-171. (Zitiert auf Seiten 9 und 61)
- [Odessa 2011] Odessa <http://computer-vision-talks.com/2011/01/comparison-of-the-opencv-feature-detection-algorithms-2/> Zuletzt besucht: 02.12.2011 (Zitiert auf Seiten 7, 19, 20 und 21)
- [ParaView] <http://www.kitware.com/> Zuletzt besucht: 16.09.2012 (Zitiert auf Seite 52)
- [PMD CamCube Einführung] PMDs CamCube Einführung: <http://www.pmdtec.com/fileadmin/pmdtec/downloads/documentation/datenblatt-camcube3.pdf> Zuletzt besucht: 13.03.2012 (Zitiert auf Seite 13)
- [PMD CamCube Entwicklungstutor] PMDs CamCube Development Tutorial: <http://www.pmdtec.com/fileadmin/pmdtec/downloads/documentation/camcube-softwaredevelopmenttutorial.pdf> Zuletzt besucht: 14.03.2012 (Zitiert auf Seiten 13 und 14)
- [Rhijn & Mulder 2005] A. van Rhijn and J. D. Mulder: *Optical Tracking and Calibration of Tangible Interaction Devices*. IPT & EGVE Workshop, 2005. (Zitiert auf Seiten 6, 7, 10, 11, 35, 54 und 61)
- [Rosten & Drummond 2006] E. Rosten and T. Drummond: *Machine learning for high-speed corner detection*. European Conference on Computer Vision, vol.1, 2006 (Zitiert auf Seite 6)

- [Rusinkiewicz & Levoy 2001] S. Rusinkiewicz and M. Levoy: *Efficient Variants of the ICP Algorithm*. In Proceeding of the Third Intl. Conf. on 3D Digital Imaging and Modeling, pages 145-152, Quebec City, Canada (Zitiert auf Seite 8)
- [Scott & Longuet-Higgins 1991] G.L. Scott and H.C. Longuet-Higgins: *An algorithm for associating the features of two images*. In Proc. Royal Society London, 1991, vol.B244, pp.21-26. (Zitiert auf Seiten 7, 25, 26, 45 und 46)
- [Shi & Tomasi 1994] J. Shi and C. Tomasi: *Good features to track*. in IEEE Computer Society Conference: Computer Vision and Pattern Recognition, 1994. (Zitiert auf Seite 5)
- [StarDetector OpenCV] [http://opencv.willowgarage.com/documentation/cpp/feature\\_detection.html](http://opencv.willowgarage.com/documentation/cpp/feature_detection.html) Zuletzt besucht: 24.08.2012 (Zitiert auf Seite 41)
- [StarDetector OpenCV Adaventure] <http://experienceopencv.blogspot.de/2011/01/star-feature-detector.html> Zuletzt besucht: 24.08.2012 (Zitiert auf Seite 41)
- [TOF-Kamera Wikipedia] TOF-Kamera Wikipedia: <http://de.wikipedia.org/wiki/Time-of-flight-Sensor> Zuletzt besucht: 02.04.2012 (Zitiert auf Seite 12)
- [Tomasi & Kanade] C. Tomasi and T. Kanade: *Detection and Tracking of Point Features*. CiteSeerX - Scientific Literature Digital Library and Search Engine (United States) (Zitiert auf Seite 5)
- [Ullmann 1976] J.R. Ullmann: *An Algorithm for Subgraph Isomorphism*. Journal of the ACM (JACM), Volume 23 Issue 1, Jan. 1976 (Zitiert auf Seite 11)
- [Umeyama 1991] S. Umeyama: *Least-squares estimation of transformation parameters between two point patterns*. IEEE Trans Pattern Anal Machine Intell (1991) 13:376?380 (Zitiert auf Seite 8)
- [Vaccetti, Lepetit & Fua 2004] L. Vaccetti, V. Lepetit and P. Fua: *Combining edge and texture information for real-time accurate 3D camera tracking*. Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality, 2004 (Zitiert auf Seite 4)
- [VICON] <http://www.vicon.com> Zuletzt besucht: 01.12.2011 (Zitiert auf Seite 3)
- [Visualization Toolkit] [http://vtk.org/Wiki/VTK\\_ Tools](http://vtk.org/Wiki/VTK_ Tools) Zuletzt besucht: 16.09.2012 (Zitiert auf Seite 52)
- [Zhang et.al 1996] T. Zhang, R. Ramakrishnan and M. Livny: *BIRCH: An efficient data clustering method for very large databases*. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, 1996, S.103-144 (Zitiert auf Seite 9)
- [Zhang & Navab 2000] X. Zhang and N. Navab: *Tracking and pose estimation for computer assisted localization in industrial environments*. Applications of Computer Vision, 2000, Fifth IEEE Workshop on. (Zitiert auf Seite 4)
- [Zhang et al. 1995] Z. Zhang, R. Deriche, O. Faugeras and Q. Luong: *A robust technique for matching two uncalibrated images through the recovery of the unknown*

*epipolar geometry.* Artificial Intelligence Volume 78, Issues 1?2, October 1995,  
Pages 87-119 (Zitiert auf Seite 5)