



**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: 2. Dezember 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 Template basiertes 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 . . . . .	8
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 . . . . .	10
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 . . . . .	15
3.1.2.3	Auflösung und erkennbare Markengröße . . . . .	16
3.2	Vorverarbeitung . . . . .	18
3.2.1	Schwellwert-basierte Segmentierung . . . . .	18
3.2.2	Steuerung der Helligkeit und Kontrast . . . . .	18
3.3	Markenerkennung . . . . .	19
3.3.1	Auswahl des Erkennungsalgorithmus . . . . .	20
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 . . . . .	22
3.3.3.1	Zustandsraummodellierung . . . . .	23
3.3.3.2	Das Kalman-Filter . . . . .	24
3.4	Markenverfolgung . . . . .	25
3.4.1	Singulärwertzerlegung . . . . .	25
3.4.2	Korrespondenzuntersuchung durch Singulärwertzerlegung . . . . .	25
3.5	Bestimmung der Transformation . . . . .	27
3.5.1	Quaternionen . . . . .	27
3.5.2	Beschreibung der Drehungen im Dreidimensionalen Raum mit Quaternionen . . . . .	29
3.5.3	Orientierung mit Einheitsquaternion . . . . .	29
3.6	Objekterkennung . . . . .	33
3.6.1	DBSCAN . . . . .	33
3.6.2	Teilgraph Isomorphismus . . . . .	36
<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 . . . . .	40
4.1.2	Markenerkennung . . . . .	41
4.1.2.1	Auswahl der Größe der Marken . . . . .	41
4.1.2.2	STAR Detektor . . . . .	43
4.1.2.3	Markenerkennung . . . . .	43
4.1.2.4	Steuerung der Helligkeit und des Kontrast . . . . .	43
4.1.3	Verbesserung der Singulärwertzerlegungsverfahren . . . . .	46
4.1.4	Segmentierung . . . . .	47
4.2	Objektlernen . . . . .	47
4.2.1	Bestimmung der Orientierung . . . . .	49
4.2.2	Markenanordnung . . . . .	50
4.2.3	Darstellung des Strukturgraphen . . . . .	51
4.2.3.1	Bestimmung der stabilen Knoten . . . . .	51
4.2.3.2	Kanteneinfügung . . . . .	51
4.3	Zugriff des Strukturgraphen von Datei . . . . .	53
4.4	Objekterkennung und Verfolgung . . . . .	53
4.4.1	Kandidaten der Objekterkennung . . . . .	54
4.4.2	Objekterkennung . . . . .	55
4.4.3	Bestimmung der Orientierung . . . . .	59
4.5	Bildersteuerung . . . . .	59
<b>5</b>	<b>Experimentelle Auswertung</b>	<b>61</b>
5.1	Teilweise Evaluation . . . . .	61
5.1.1	Abstands-Filter . . . . .	62
5.1.2	Helligkeitssteuerung . . . . .	64
5.1.3	Verbesserung des Singulärwertzerlegungsverfahrens . . . . .	66
5.1.4	Aktualisierung des Strukturgraphen . . . . .	70
5.1.4.1	Lebenszeit . . . . .	70
5.1.4.2	Abstandschwellenwert für die identischen Knoten . . . . .	71
5.1.4.3	Kombination der mehrfach erkannten Knoten . . . . .	73
5.1.5	Bildersteuerung . . . . .	73

5.1.6	Teilgraph-Isomorphismus . . . . .	73
5.2	Globale Evaluation . . . . .	75
5.2.1	Objektlernen . . . . .	76
5.2.1.1	Zeitaufwand . . . . .	79
5.2.2	Objektwiedererkennung . . . . .	79
5.2.2.1	1 Eingabeobjekt mit 1 Eingabemodell . . . . .	81
5.2.2.2	1 Eingabeobjekt mit 2 Eingabemodellen . . . . .	82
5.2.2.3	Verbesserung mit Kandidaten . . . . .	82
5.2.2.4	2 Eingabeobjekte mit 2 Eingabemodellen . . . . .	84
<b>6</b>	<b>Zusammenfassung und Ausblick</b>	<b>87</b>
	<b>Literaturverzeichnis</b>	<b>89</b>

# Kapitel 1

## Einleitung

### 1.1 Motivation

Die Stärken von Robotern liegen in der Wiederholung von einfachen Handhabungstätigkeiten. Dagegen sind Menschen mit ihren kognitiven Fähigkeiten einzigartig, etwa in Bezug auf ihren Verständnis der Aufgabe. Die Kombination von Mensch und Roboter kann Aufgaben stark rationalisieren, sofern jedem die optimalen Arbeitsteile zugewiesen wird. Die Anwendungsbereiche der Mensch-Roboter-Kooperation vergrößern sich derzeit auf dem Feld der Medizin sowie der Industrie immer schneller. Damit Menschen 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 dessen Blickrichtung, um Aussagen über die Aufmerksamkeit des Menschen treffen zu können. Die Objekterkennung vereinfacht die Kommunikation zwischen Mensch und Roboter, wodurch die Fremdobjekte ohne weitere Programmierung direkt vom Roboter erkannt werden können. Außerdem vermeidet die Objektverfolgung auch die Kollision zwischen Roboter und anderen Anlagenteilen.

### 1.2 Aufgabenstellung

Das Rahmenwerk MAROCO wird am Institut für Prozessrechentechnik, Automation und Robotik (IPR) entwickelt, damit Menschen und Roboter in einer gemeinsamen Umgebung sicher zusammenarbeiten können. Die Erfassung des Menschen und die Handlungsanalyse in der Szene erlauben es, die Gefahren von Roboterbewegungen 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 dieser Arbeit besteht darin, die verschiedenen Objekte zu kalibrieren und die ent-

sprechenden geometrischen Charakteristika in dem System zu speichern, um dann die Objekte mit Hilfe der gespeicherten Informationen wiederzuerkennen und zu verfolgen. Beide Schritte sollen in Echtzeit durchgeführt werden.

Folgende Ziele sollen erreicht werden:

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

# Kapitel 2

## Stand der Forschung

Die 3D Objekterkennung und Verfolgung kommt in vielen Anwendungsbereichen zum Einsatz. Daher wurden in den vergangenen Jahrzehnten viele Algorithmen bzw. Systeme dafür entwickelt, wie beispielsweise das kommerzielle Erkennungssystem von VICON [VICON]. In diesem System werden acht Kameras benutzt, die das Zielobjekt bzw. die Person von verschiedenen Richtungen beobachten. Hierzu werden einige weiße Marken vorher am Ziel angebracht, damit dessen 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 findet im Bereich von Computerspielen und Filmindustrie sehr häufig Verwendung. Ein anderes Beispiel ist das neue Gerät „Kinect“ von Microsoft XBox360 [Kinect]. Eine 3D Kamera kann die 3D Bewegungsdaten von Spielern ansammeln, wodurch die Spieler das Spiel direkt mit ihren Körpern anstelle des traditionellen Kontrollers steuern können. Die Analyseverfahren der Objekterkennung basieren auf unterschiedlichen Charakteristika der Objekte und sind für verschiedenen Typen von Objekten geeignet. In der Arbeit von Lepetit und Fua sind die aktuellen Verfolgungsverfahren in zwei große Gruppen untergliedert worden: auf Marken basierte Objektverfolgungen und auf natürlichen Merkmale basierte Objektverfolgungen [Lepetit & Fua 2005]. Die Verfahren in der zweiten Gruppe können weiter in kantenbasiertes Verfahren, auf optischen Fluss basiertes Verfahren, Template basiertes Verfahren, Punktebasiertes Verfahren und das SLAM-Verfahren unterteilt werden. Im folgenden Abschnitt wird kurz auf die einzelnen Verfahren eingegangen.

### 2.1 Standardmarkenbasierte Verfahren

Die Verfolgungsverfahren können in zwei Schritte unterteilt werden: Zuerst das Sammeln von Informationen über Bildsequenzen, um dann die Position des erkannten Objekts zu bestimmen. Die vordefinierten Marken können in beiden Schritten mehr Information liefern, wodurch 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 vier Vierecken als eine Marke benutzt. Die Marke wird durch Farbe und weiße Flecken innerhalb der Vierecke kodiert.

Ein anderes System heißt ARToolKit, und ist vom HITLab der Universität Washington entwickelt worden. 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 danach 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 das Projizieren zwischen dem Kamerakoordinatensystem und dem 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, sodass die Marke in niedriger Auflösung sehr gut erkannt werden kann. Ein anderer wichtige Vorteil ist, dass die Verfolgung der ARToolKit keine Initialisierung benötigt. Dadurch wird nicht nur die Laufzeit am Anfang des Verfahrens gespart, sondern auch ein 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 liegt entweder darin, 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. RAPiD war eines der frühesten 3D Verfolgungsverfahren, das in Echtzeit laufen konnte [Harris 1992]. Vaccetti und Lepetit haben ein neues, effizienteres Verfolgungsverfahren entwickelt, welches mehr als eine Voraussage für die ausgewählten Steuerpunkte darstellt [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 die Bewegungsgeschwindigkeit für jeden Bildpunkt einer Bildsequenz bezeichnet. Die Berechnung des optischen Flusses kann als eine Differentialgleichung zusammengefasst werden, dessen Lösungsverfahren von Horn und Schunck entwickelt [Horn & Schunck 1981] wurde. Black und Yacoob benutzten ein auf reinen optischen Fluss basiertes Verfahren für die Verfolgung kleiner Veränderungen auf dem menschlichen Gesicht, um den Gesichtsausdruck zu bestimmen [Black & Yacoob 1997]. Außerdem wurde ein Verfolgungssystem für den Innerstadtverkehr von Haag und Nagel durch die Verknüpfung der Information von optischem Fluss und den Kanten des Objekts implementiert [Haag & Nagel 1999].

### 2.2.3 Template basiertes Verfahren

Im Template basierten Verfahren wird ein Objekt nicht durch lokale Merkmale, wie 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 Template basierte 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, wodurch 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 eine Kollision für mehrere Objekte gibt, oder die Messung der Merkmale stark gestört wird [Lepetit & Fua 2005]. Ein Verfahren wurde von Zhang et al. im Jahre 1995 realisiert, welches die nicht kalibrierten Bilder als Eingabe benutzen kann [Zhang et al. 1995]. In diesem Verfahren wird kein Modell der Epipolargeometrie verwendet, wodurch viele komplexe Berechnungen vermieden werden können. Eine andere Möglichkeit für die Punkteverfolgung ist der Kanade-Lucas-Tomasi (KLT) Tracker, welcher 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 entwickelten die Auswahl der Punkte mit der Ähnlichkeit zwischen dem Anfangsbild und dem aktuelle Bild weiter. Diese Ähnlichkeit wird durch ein Modell von affinen Abbildungen bestimmt. Außerdem benutzen sie gleichzeitig ein zweites Modell mit einer reinen Translation, um das Objekt mit hoher Stabilität und Präzision zu verfolgen.

## 2.3 Markenbasierte Objekterkennung

Rhijn und Mulder haben ein markenbasiertes Verfahren entwickelt, welches 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 dreidimensionalen, vollständigen Graph. Wenn das Objekt wiedererkannt werden soll, führt das System zuerst einen Kalibrationsalgorithmus durch, um die verschiedenen Objekte zu differenzieren. Daraufhin vergleicht das System für jedes Objekt die sichtbaren Marken mit dem in der Initialisierungsphase gespeicherten vollständigen Graphen.

### 2.3.1 Markenerkennung

Heutzutage gibt es viele benutzte Standardalgorithmen der Markenerkennung. Die Harris Matrix wird von der Summe der partiellen Ableitungen des Eingabebildes dargestellt. Dabei wird eine Ecke genau dann als ein Merkmal erkannt, wenn die beiden Eigenwerte der Harris-Matrix positiv und groß genug sind [Harris & Stephens 1988]. Ein anderer Algorithmus für diese Eckenerkennung laute „Features from Accelerated Segment Test“(FAST) und wurde von Rosten und Drummond im Jahre 2006 veröffentlicht [Rosten & Drummond 2006]. Außer vorherigen Eckenerkennungsverfahren können sogar die Marken durch ähnliche Ideen erkannt werden. Das Ziel dieser Verfahrensart ist, die Punkte zu extrahieren, die unterschiedliche Eigenschaften zu ihrer Umgebung aufweisen, wie z.B Helligkeit und Farbe. Lowe hat seinen Algorithmus „Scale-invariant feature transform“ (SIFT) im Jahre 2004 veröffentlicht, welcher eine starke Stabilität gegen Transformation, Beleuchtungsvariation bzw. Bildrauschen aufbringt [Lowe 2004]. Der Detektor arbeitet auf Basis einer Skalenraum-Analyse mit Difference of Gaussians (DoG), welches einer effiziente Approximation des skalen-normalisierten LoG-Operators (Laplacian of Gaussian) entspricht. Das lokale Extremum wird in DoG Bildern durch den Vergleich der direkten Nachbarn bzw. der Nachbarn aus neben-einanderliegenden DoG Bildern gesucht. Der entsprechende Deskriptor wird als 128 dimensionaler Vektor definiert und besteht aus einem in Teil-Quadranten untergliederte Gradienten Histogramm. Bay et al. erweiterte die Arbeit von Lowe mit dem „Speeded Up Robust Feature“ (SURF) [Bay et al. 2006]. Der Rechtecks-Filter wird in ihrer Arbeit anstelle eines DoGs verwendet, um den Detektionsablauf zu beschleunigen. Außerdem wird der Deskriptor auch um den Faktor 2 verkleinert, damit der Vergleich der

zwei Merkmalsvektoren weniger Zeitintensiv ist. Der Algorithmus „Center Surround Extremas“ (CenSurE oder STAR) benutzt einen 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 im Vergleich viel effizienter und stabiler. Die vorhandenen Realisierungen obiger Algorithmen können in der freien Programmabibliothek OpenCV gefunden werden. Anhand von OpenCV hat Odessa in seinem Blog die Geschwindigkeit, die Stabilität bzw. die Genauigkeit dieser Algorithmen verglichen [Odessa 2011]. Seinem Ergebnis zufolge scheint, der STAR Algorithmus die niedrigste durchschnittliche Fehler-Rate zu haben bzw. den geringsten Berechnungsaufwand zu benötigen.

### 2.3.2 Markenverfolgung

Nach Bestimmung der Marken, sollen diese in einer Bildsequenz verfolgt werden. Scott und Longuet-Higgins haben einen eleganten und einfachen Algorithmus entwickelt und das Problem zu einer Matrix zusammengefasst, deren Elemente als die Distanz zwischen verschiedenen Merkmalen definiert werden [Scott & Longuet-Higgins 1991]. 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, indem sie die Elemente der Matrix neu definiert und die Beschränkung der Epipolargeometrie eingefügt haben [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 die Transformation des Objekts direkt bestimmen können. Diese Verfahren wurden von Eggert et al. in ihrer Arbeit durch Merkmale und Lösungsverfahren in verschiedene Typen unterteilt [Eggert, Lorusso & Fisher 1997]. Hierbei können die Merkmale Oberflächen, Kanten oder Punkte sein. Die Verfahren, die auf Punkten basieren, werden in der Praxis häufig benutzt und sind für die markenbasierte Objekterkennung geeignet [Eggert, Lorusso & Fisher 1997]. Bei den Lösungsverfahren kann zwischen iterativen Verfahren und geschlossene Formen 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, um die Summe des Unterschieds zwischen den durch approximierte Transformation berechnete Positionen der Punkte und den genauen Positionen der Punkte zu minimieren. Dieses Verfahren basiert auf der Singulärwertzerlegung einer Korrelationsmatrix. Die Rotations- und Translationsmatrix werden am Ende ausgegeben. Wenn die zwei eingegebenen Punktwolken auf gleicher Oberfläche liegen, liefert das Verfahren leider keine richtige Rotationsmatrix. Deshalb soll eine korrigierte Matrix darauf aufbauen, welche von Umeyama [Umeyama 1991] und Kanatani [Kanatani 1994] vorgeschlagen wurde. Ein anderes Verfahren wurde von Horn entwickelt, welches die relative Rotation durch Einheitsquaternionen beschreibt [Horn 1987]. Im Vergleich zu der Standard-Beschreibung der Rotation als Matrix ist die Quaternion-Darstellung 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, das 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 beträgt, braucht das Verfahren mit Einheitsquaternionen weniger Zeit als die anderen Verfahren. Auf der anderen Seite liefert das Verfahren von Arun den kleinsten Fehler, wenn die Anzahl der betrachteten Punkte weniger als 10 ist.

### 2.3.3.2 Iterative Closest Point Algorithmus

Der Iterative Closest Point Algorithmus (ICP) ist ein Algorithmus, der es ermöglicht, die Punktwolken aneinander anzupassen [ICP Wiki]. In jedem Iterationsschritt wird der korrespondierende Punkt für jeden Punkt einer Punktwolke aus einer anderen Punktwolke gefunden. Die Transformation zwischen beiden Punktwolken wird so bestimmt, dass die Summe des Abstands der korrespondierenden Punkte minimiert wird. Dieser Vorgang wird solang wiederholt, bis die Veränderung des mittleren quadratischen Fehlers zwischen zwei folgenden Schritten unter einem bestimmten Schwellenwert liegt. Der Algorithmus wurde zuerst von Chen und Medioni [Chen & Medioni 1991] vorgestellt und unter dem Namen ICP erstmals in der Arbeit von Besl und McKay benutzt [Besl & McKay 1992]. Doria et al. erweiterten den Algorithmus mit einem gewichteten Kriterium für das Rauschen [Dorai, Weng & Jain 1997]. Ein Vergleich der verschiedenen ICP Algorithmen vor dem Jahr 2001 wurde von Rusinkiewicz und Levoy erstellt [Rusinkiewicz & Levoy 2001]. Der grundlegende ICP Algorithmus wurde von ihnen in

sechs 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 alleine 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]. Diese laufen „Balanced Iterative Reducing and Clustering using Hierarchies“ (BIRCH), „Density Based Spatial Clustering of Applications with Noise“ (DBSCAN) und CHAMELEON-Algorithmus. BIRCH-Algorithmus basiert auf dem Clustering-Feature-Tree, welcher einen balancierten Baum mit Verzweigungsfaktor und Schwelle als zusätzlichen Parameter darstellt [Zhang et.al 1996]. Ein vorhandener Clusteralgorithmus wird für die Blätter des Baums durchgeführt, damit die angeforderten Gruppen ausgesucht werden können. In CHAMELEON werden zuerst die Nachbarn des betrachteten Objekts bestimmt. Daraufhin 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, der von Ester et.al in Jahr 1996 veröffentlicht wurde, kann dieses Problem vermeiden [Ester et.al 1996]. Statt der gewünschten Anzahl der Cluster sollen ein Bereich  $\epsilon$  und die Mindestanzahl der Nachbarn in diesem Bereich, die sogenannte Grenzdichte, vordefiniert werden. Das Objekt, das nicht genug Nachbarn im angeforderten Bereich aufweist, wird als Rauschen markiert. Die bleibenden Objekte werden im gleichen Cluster zusammengefasst, wenn sie nicht weiter als  $\epsilon$  von mindesten  $N$  ihrer 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 in einfacher Gestalt oder für die Punktmenge in nicht-sphärischer Gestalt können die DBSCAN und CHAMELEON Algorithmen die zufriedenstellende Lösung liefern. In der Situation, in der die Testdaten verrauscht sind, kann der DBSCAN Algorithmus mit geeignetem Parameter auch ein zufriedenstellendes Ergebnis ausgeben. Das Ergebnis bleibt stabil, obwohl der Anteil des Rauschens stark ansteigt.

### 2.3.4.2 Dynamisches Verfahren

Das Problem der Kalibrierung der Objekte ist ähnlich im Vergleich zum Problem der Segmentierung der Bewegungen in einer langen Bildsequenz. Eine traditionelle Lösungsstrategie ist, die Positionen der Marken im nächsten Zeitpunkt mit einem Kalman Filter vorherzusagen. Dann werden alle Marken anhand der kinematischen Parameter in verschiedene Gruppen eingeteilt. Mills und Novins haben eine andere Möglichkeit geliefert, um die Objekte direkt mit zweidimensionalen Graphen kalibrieren zu können [Mills & Novins 2000]. Am Anfang ihres Algorithmus wird jede Marke mit einander verbunden. Danach werden alle Marken und die dazwischenliegenden Kanten 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 schrittweise gelöscht. Eine Marke gehört genau dann zu einem Objekt, 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 nur dann in einem Objekt erkannt werden, wenn sie mit anderen drei Marken in dem Objekt zusammen eine Pyramide erzeugen kann. Hierbei bezieht sich die Pyramide auf einen Körper der Geometrie, der vier Knoten hat, von denen je zwei eine Kante erzeugen. Die neue stärkere Beschränkung erhöht die Erfolgsquote deutlich.

### 2.3.5 Objekterkennung und Verfolgung

Das Ziel dieser Arbeit liegt darin, 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 auf einer gleichen 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 sogenannte Teilgraph Isomorphismus

Problem abgeleitet werden kann. Es gibt eine große Menge der Algorithmen, die das Problem behandeln, da das Isomorphismus Problem nicht nur im Bereich der Bildanalyse, sondern auch im Vergleich der Struktur von chemischen Verbindungen oder in biometrischer Identifikation häufig vorkommt. Conte et al. haben eine Zusammenfassung über diese Algorithmen veröffentlicht [Conte et al. 2004]. Durch ihre Taxonomie können sämtliche 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 ein genaues Matching häufig benutzt wird [Ullmann 1976]. Was von Rhijn und Mulder in ihrer Arbeit für die Objekterkennung implementiert wurde, bezieht sich auch auf 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 diesem 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], wobei der Graph in viele kleine Bäume unterteilt wird. Auf diesen wird dann mittels dynamischer Programmierung das Matching in linearer Zeit durchgeführt.

Das genaue Graph Matching ist manchen Fällen ungeeignet, beispielsweise für nicht komplett fest definierte Graphen, das z.B. bei Rauschen oder instabilen Komponenten vorkommt. Wegen des Unterschieds zwischen dem beobachteten Modell und dem idealen Modell, soll das Matchingverfahren 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 eine exponentielle Laufzeit im Worst-Case, die durch eine Approximation in ungenauen Matchingverfahren stark reduziert werden kann. Messmer und Bunke haben ein Fehler-tolerantes Verfahren für einen 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 öfter vorkommenden Teilgraphen nur einmal repräsentiert werden. Der eingegebene Graph wird mit diesen verdichteten Graphen verglichen, wodurch die Laufzeit nur von der Anzahl der Modellgraphen abhängt.

# 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 benötigt. Die Distanz ist direkt proportional zu dieser 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-Kamerasystemen hat die TOF (englisch: time of flight) Kamera viele Vorteile [[TOF-Kamera Wikipedia](#)]. Zuerst kann die TOF Kamera die interessierenden Bereiche einfach aus einem Bild extrahieren und nur die Pixel nah vor der Kamera betrachten. Zweitens kann die TOF Kamera eine hohe Bildrate bis zu 100 fps erreichen. Diese Eigenschaft ermöglicht somit eine Echtzeitanwendungen. Außerdem benötigt die TOF Kamera weniger Platz im Vergleich zum Triangulationsystem, und hat eine niedrigere Abhängigkeit von der Systemstruktur gegenüber dem Stereosystem.

Parameter	Value *	Notes
Type of Sensor	Photonic® 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 (englisch: Photonic Mixing Device) Sensor ist eine wichtige Art der TOF Kamera. Er liefert eine hohe Auflösung bis zu 204x204 Pixel und einer maximalen Bildrate bis zu 25 fps. Durch die Formel (3.1) wird der maximale Distanzbereich auf 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 eine 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](#)].

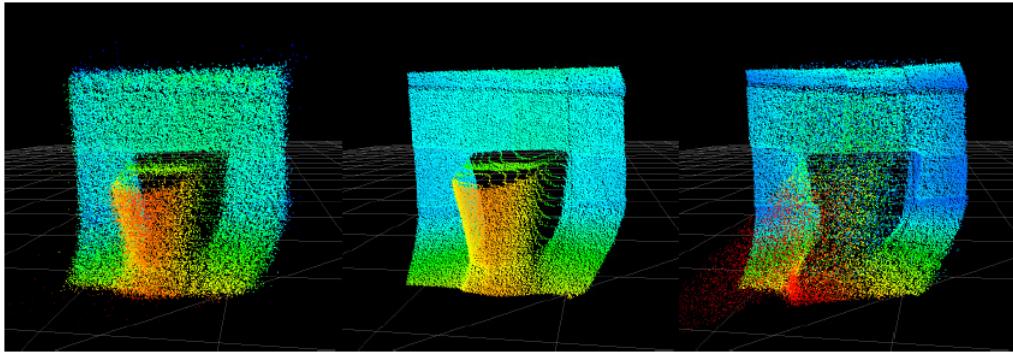


Abbildung 3.2: Integrationszeit von  $140 \mu\text{s}$ ,  $1400 \mu\text{s}$  bzw.  $14000 \mu\text{s}$ . Bitte beachten Sie die niedrige Signalstärke an der linken Seite und die Sättigung an der rechten Seite wegen der unangemessenen Integrationszeit. [PMD CamCube Entwicklungstutor]

### 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 steuern können [Kinect]. Um dieses Ziel zu erreichen, sammelt Kinect abgesehen von 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, wie z.B. Windows, Linux bzw. MacOS möglich.

Der Unterschied zwischen TOF Kamera und Kinect kann auf dem Arbeitsprinzip zurückgeführt werden. In TOF Kameras wird die Tiefdaten durch das Laufzeitverfahren berechnet, was im 3.1.1.1 erklärt wird. Das Abtastverfahren der Kinect heißt „Light Coding“. Eine große Menge von Punkten wird als Mustern auf die Szene bzw. die Objekte durch infrarotes Licht projiziert. Die ganze 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. Weitere Informationen 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 ihres großen Messbereichs für das MAROCO-System geeignet. Außerdem ist mithilfe des SBI Systems die Arbeit der PMD Kamera unter schwieriger Umgebungsbedingung, wie z.B. außerhalb des Zimmers mit starker Störung von Sonneneinstrahlung, auch möglich.

Sensor	PMD CamCube	Kinect
Auflösung	$204 \times 204$ Pixel	$640 \times 480$ Pixel
Sichtfeld	$40^\circ \times 40^\circ$	$57^\circ \times 43^\circ$
Max Bildrate	25 fps	30 fps
Messbereich	$0.3 \rightarrow 7.0$ m	$1.2 \rightarrow 3.5$ m (mit Xbox Software)
Länge der Tiefdaten	8 bit (unsigned char)	11 bit

Tabelle 3.1: Die technische Daten von PMD Kamera und Kinect. [[PMD CamCube Einführung](#)]

### 3.1.2 Daten der PMD Kamera

Die PMD Kamera CamCube kann insgesamt vier verschiedene Vermessungsdaten ausgeben. Hierzu zählen Amplitude, Intensität, Distanz und 3D-Koordinaten. Die ersten und letzten beiden Datentypen 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 zwischen 0 und 255 beschränkt werden. Das Ergebnis ist das Bild einer normalen Schwarz-Weiß Kamera. Die Amplitude zeigt die Stärke der Beleuchtung an, die vom Objekt wegen des aktiven Sendersignals von der PMD Kamera selbst reflektiert wird. Dieser Wert kann die Qualität der Distanzinformation abschätzen, 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 die Daten der Amplitude einfacher betrachtet, was für die Erkennung der großen künstlichen Marken in dieser 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 dem Objekt, welches 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 3.4 zeigt jeweils die Visualisierung einer Szene mit der 3D-Koordinaten und Distanzinformation von PMD Kamera. Eine

Transformation ist notwendig, wenn man die Distanzinformation direkt im kartesischen Koordinatensystem visualisieren möchte.

### 3.1.2.3 Auflösung und erkennbare Markengröße

Von weiterem Interesse ist, wie genau die Objekte von der Kamera in der Praxis beobachtet werden können, d.h. wie groß ein Pixel der Kamera ist, der den Bereich in realer Welt beschreibt. Die Vorderansicht des Kamerasytems dieser Arbeit ist in Abbildung 3.3 links gezeigt.

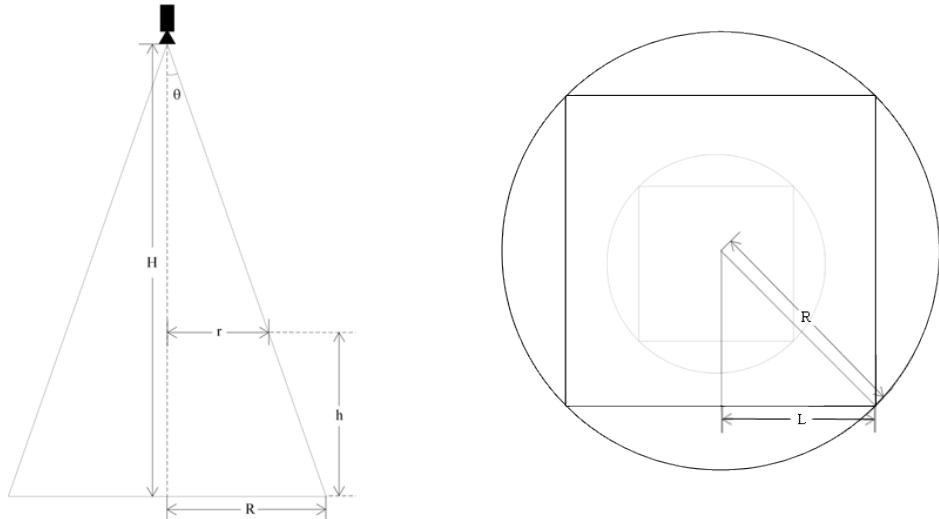


Abbildung 3.3: Die Vorderansicht (links) und die Draufsicht (rechts) des PMD Kamera Systems

$\theta$  ist der halbe Sichtwinkel und wird hier als  $20^\circ$  angenommen.  $H$  zeigt den Abstand von der Kamera zum Boden, der in diesem System mit  $3,2m$  festgelegt ist. Durch 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  $L$  (Siehe Abb 3.3 rechts) ist gleich:

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

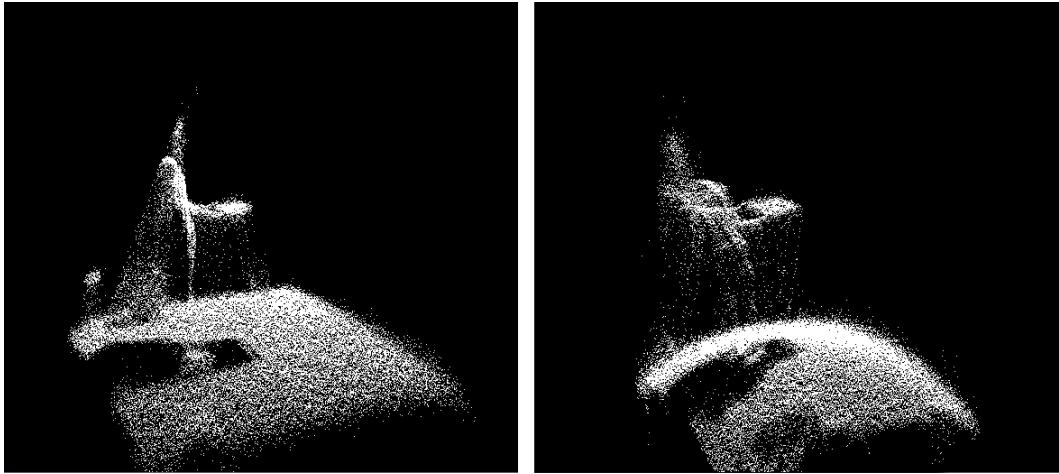


Abbildung 3.4: Die Visualisierung von 3D Koordinaten (links) und Distanzinformation (rechts).

Dadurch erhält man den Flächeninhalt des Sichtbereiches auf dem Boden:

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

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

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

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

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

Die minimalen Größen der Marken für andere, zum Boden parallelen Ebenen können analog berechnet werden. So kann z.B. in der normalen Arbeitsebene dieser Arbeit, die zum Boden 1,1m entfernt ist, die Mindestanzahl der 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:

$$f : I \mapsto H$$

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 jeden 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  $b \in R$  jeweils den Kontrast und die Helligkeit kontrollieren. Es gibt vier Möglichkeiten für die verschiedenen Zuordnungen dieser Parameter:

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



Abbildung 3.5: Die vier Beispielbilder. Von links nach rechts sind Barbara, Lena, Peppers und Mandril. [Odessa 2011]

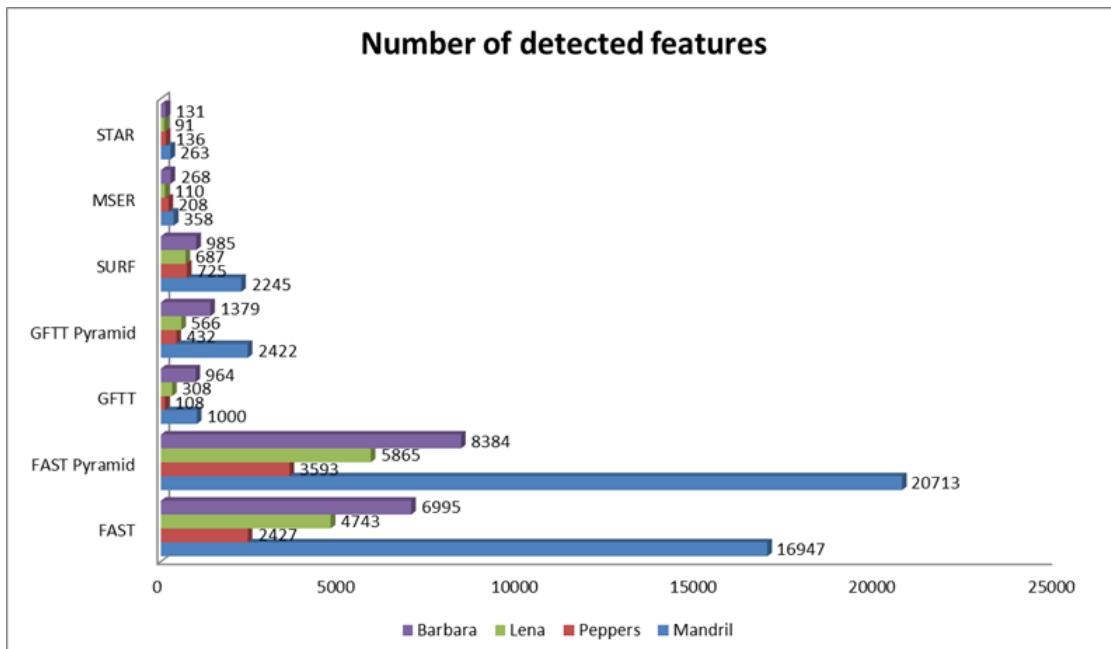


Abbildung 3.6: Die Anzahl der erkannten Merkmale von allen vier Beispielbildern durch verschiedene Erkennungsalgorithmen. [Odessa 2011]

### 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 der Vergleich dieser Algorithmen häufig 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 (Siehe Abb. 3.5).

Von besonderem Interesse 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 des CenSurE Algorithmus in OpenCV). Den Unterschied sieht man deutlich in der Abbildung 3.6. 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 3.7 zeigt den durchschnittlichen Fehler in Pixeln von den Punktpaaren, der durch den 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 der besseren Konzentration 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]. Analog zum SIFT Algorithmus werden die Extreme zwischen den Skalenräumen herausgefunden, welche als die Merkmale des betrachteten Bilds erkannt werden. Der Bi-Level Filter wird in dieser Arbeit anstelle der Differenz der Gaussian verwendet, um die Laufzeit zu reduzieren. Weiterhin wird die Approximation des gefilterten Bereichs von Bi-Level Filter durchgeführt. Dadurch kann das Algorithmus in Echtzeit laufen lassen, obwohl alle Merkmale in allen Skalenräumen betrachtet werden sollen.

#### 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 3.8 zeigt die Progression des Bi-level Filters mit verschiedenen Symmetrischen Stufen. Der kreisförmige Filter an linker Seite der Abbildung 3.8 kann den Laplacian-Operator zwar am besten approximieren, ist aber leider schwierig zu implementieren. Deshalb werden die Progressionen der Filter durch Polygone angenähert, wodurch die Berechnung vereinfacht werden kann. Zum Vergleich der übrigen Formen der Abbildung 3.8 liefert der Filter mit Achtecken die beste Leistung und der Filter mit Rechtecken die kürzeste Laufzeit. Diese Polygon-Filter können durch die integralen Bilder einfach dargestellt werden.

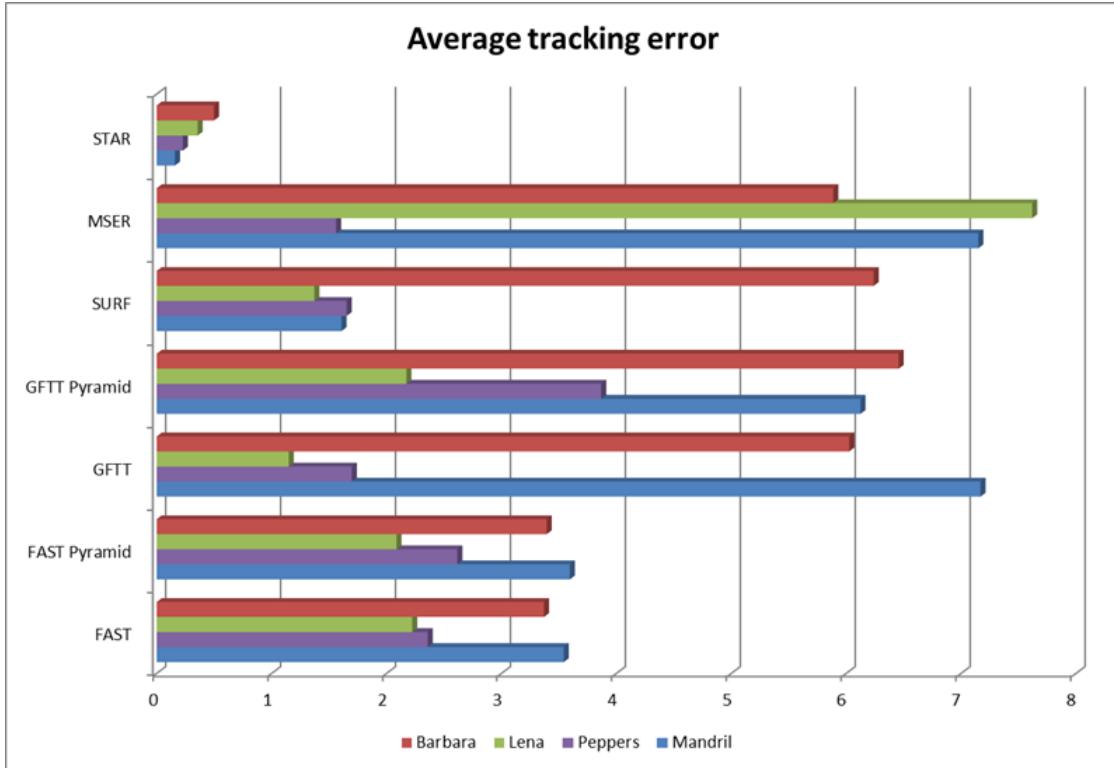


Abbildung 3.7: Der durchschnittliche Fehler (in Pixeln) zwischen den assoziierten Punkten zweier folgender Bilder. [Odessa 2011]

### 3.3.2.2 Integrale Bilder

Ein integrales Bild  $I$  ist eine mittlere Repräsentation eines Bildes, welches 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 eine einmalige Durchführung aller Pixel 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 Polygonen-Filter benutzt. Die Kombination von zwei verschiedenen Bildern, die schräg integriert werden, kann den für Polygone benötigten, trapezförmigen Bereich einfach darstellen. Die mathematische Beschreibung des schrägen Integrals des Bildes ist:

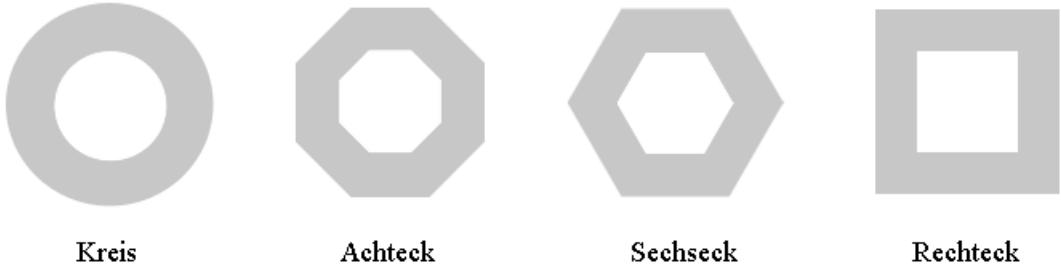


Abbildung 3.8: Progression der Center-Surround Bi-level Filter. Der Kreis ist die ideale voll-symmetrische Approximation der Laplacian. Die Filter daneben, von links nach rechts, haben eine niedrigere Symmetrie, brauchen aber weniger Zeit zur Berechnung. [Agrawal, Konolige & Blas 2008]

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

wobei  $\alpha$  den schrägen Winkel erklärt und wenn es 0 gleicht, ist die Formel (3.4) genauso wie die Formel (3.5), und beschreibt ein rechteckiges integrales Bild. Die in der Abbildung 3.8 gezeigte Achteck-Filter und Sechseck-Filter können mit jeweils drei bzw. zwei 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 die Lage bzw. die Skala gibt, dessen Response größer oder kleiner ist, als das zu betrachtende Pixel. Die Pixel mit entweder Maximum oder Minimum Response werden als Merkmale erkannt. Der Suchumfang wird in der Arbeit von Agrawal als 3x3x3 eingestellt, d.h. acht Pixel um den betrachteten Pixel und jeweils neun Pixel in zwei benachbarten Skalenräumen werden zusammen berücksichtigt. Die Pixel mit höherer Response können zwischen der Transformation des Bildes stabiler wiedererkannt werden, weshalb die ausgewählten Extrema nach der Non-maximal Suppression noch einmal durch einen Schwellwerte-Filter gefiltert werden sollen, um die besten Merkmale zu bestimmen.

### 3.3.3 Kalman-Filter

Der Kalman-Filter basiert auf einem linearen, dynamischen System in einem diskreten Zeitraum. Die Zustandsgleichung des Systems wird häufig durch eine Differenzengleichung

chung 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 die Zustandsgleichung zur linearen Differenzengleichung vereinfacht.

### 3.3.3.1 Zustandsraummodellierung

Der nächste Systemzustand kann basierend auf dem aktuellen Systemzustand durch:

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

geschätzt werden. Der Index  $k$  bzw.  $k - 1$  bezieht 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 zum 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 der Kontrollvektor  $u_{k-1}$  stellen den deterministischen Anteil der weiteren, äußeren Einflüsse auf das System dar. Die zufälligen, nicht erfassbaren Komponenten der äußeren Einflüsse 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 bestehen aus einer modellierbaren Verzerrung und einem unvorhersehbaren Messrauschen:

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

$Z_k$  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 unabhängig und normalverteilt angenommen:

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

### 3.3.3.2 Das Kalman-Filter

Das Ziel eines Filters besteht darin, die Zustände durch die Informationen einer Messreihe 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-Filters ist es, eine rekursive Formulierung aufzubauen, die aber nur die Schätzung eines vorherigen Zeitpunktes und die aktuelle Messung benötigt, um die Schätzung des aktuellen Zeitpunktes zu bestimmen. Es gibt hauptsächlich zwei Phasen im Kalman-Filter.

#### Prädiktion

In 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 Markenverfolgung

Die Marken, die vom Markendetektor erkannt werden, sollen während der Bildsequenz verfolgt werden. Damit kann die Transformation des betrachteten Objekts zwischen je zwei Bildern bestimmt werden, d.h. die gleichen Marken von verschiedenen Bildern sollen zuerst erkannt werden. In dieser Arbeit wird das Singulärwertzerlegungsverfahren verwendet, um diese Korrespondenzpunktpaare herauszufinden.

### 3.4.1 Singulärwertzerlegung

Sei  $M$  eine komplexe  $m \times n$  Matrix mit Rang  $r$ . Die Singulärwertzerlegung bezeichnet dann 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:

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

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 den Merkmalen  $I_i$  und  $J_j$  beschreibt.  $\sigma$  wird als ein Standard für den 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  $U$  unitäre Matrizen mit den jeweiligen Größen  $m \times m$  und  $n \times n$  sind, und  $D$  eine Diagonalmatrix bildet. 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, welches 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 Arbeit von Scott und Longuet-Higgins [Scott & Longuet-Higgins 1991] erzeugt wurde.

---

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

---

```

 $I, J, \sigma, Result$ 
for  $i = 1 \rightarrow m, j = 1 \rightarrow n$  do
     $r_{ij} \leftarrow Dis(I_i, J_j)$ 
     $G_{ij} \leftarrow \exp\left(-\frac{r_{ij}^2}{\sigma^2}\right)$ 
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

Es gibt viele Möglichkeiten, die Lage eines Objekts im dreidimensionalen Raum zu schätzen. In dieser Arbeit wird das Verfahren von Horn [Horn 1987] verwendet, das durch Einheitsquaternionen die Transformation eines Objekts zwischen zwei Zeitpunkten bestimmen kann.

### 3.5.1 Quaternionen

Ein Quaternion besteht aus einem Vektor mit vier Elementen, wobei ein Element als ein Skalar bezeichnet und die anderen drei eine Richtung im dreidimensionalen 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 einer 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 getrenntem Realteil und Imaginärteil wird definiert durch:

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

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

Sei  $r$  eine andere Quaternion mit:

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

Analog zu Vektoren im  $\mathbb{R}^4$  wird das Skalar Produkt zwischen zwei Quaternionen 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 Quaternion Multiplikation mithilfe von (3.13) berechnet werden als:

$$qr = (q_0 r_0 - \vec{q} \cdot \vec{r}, \quad q_0 \vec{r} + \vec{q} \vec{r}_0 + \vec{q} \times \vec{r}) \quad (3.14)$$

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

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

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

Die konjugierte Quaternion von  $q$  ist definiert als:

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

Das Produkt einer 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 der konjugierten Quaternion kann man die Länge der Quaternion  $|q|$  definieren:

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

Ist die Länge einer Quaternion gleich 1, nennt man die Quaternion eine Einheitsquaternion. Für eine Einheitsquaternion gilt:

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

D.h. die Inverse und Konjugierte sind identisch. Für jede 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  $v = \frac{1}{\sin \alpha}(iq_x + jq_y + kz)$ .

### 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  $q$  eine 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

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

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 Jahre 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

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

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 werden 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

$$\sum_{i=1}^n d_i'^t \mathbf{R} m_i' \quad (3.29)$$

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:

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

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 Punktpaares von Punktmengen  $D$  und  $M$ :

$$\mathbf{H} = \sum_{i=1}^n m'_i d_i'^t. \tag{3.33}$$

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

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

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:

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

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

## 3.6 Objekterkennung

Die gesamte Objekterkennung kann in zwei Phasen aufgeteilt werden: die Segmentierung und der Vergleich des Strukturgraphen. In diesem Abschnitt werden die wichtigsten Algorithmen für beide Teile erklärt.

### 3.6.1 DBSCAN

DBSCAN, kurz für den 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, der 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

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

erfüllt.

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

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

**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  $q$  heißen 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.

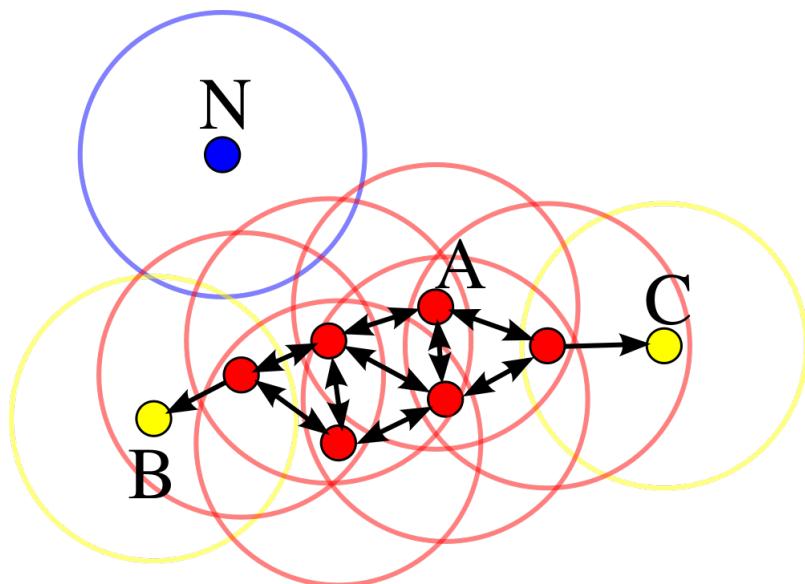


Abbildung 3.9: Ein Beispiel für die drei Typen der Punkte in DBSCAN-Algorithmus. Der rote Punkt A ist ein Kernpunkt. Die gelbe Punkte B und C sind die Grenzpunkte, die von roten Punkten Dichte-erreichbar sind. Wegen fehlender Dichte-Verbindung zwischen Punkt N und anderen Punkten, wird N als Rauschen erkannt. [DBSCAN Wiki]

Abbildung 3.9 zeigt ein Beispiel für alle diese 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  $\epsilon$ -Umgebungen werden verknüpft, wenn sie mindestens einen identischen 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**

---

### 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 Graph Isomorphismus zu vereinfachen, wird nur ein Teilgraph  $S_{min}$  von Modellgraphen betrachtet.  $S_{min}$  soll ein vollständiger Graph sein, um einen Modellgraphen eindeutig bestimmen zu können. Zuerst wird ein Punkt  $p_i$  vom Eingabographen ausgewählt und die Distanzen von diesem zu allen seinen Nachbarn berechnet. Sämtliche Kantenlängen werden mit den Kanten der Modellgraphen verglichen, damit ein Kandidatenpunkt  $v_k$  im Modellgraphen 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: $G_m$ , Eingabograph: $G_d$

```

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 Pyramide 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 gegeben. Das ganze Programm ist eine Schleife, in jeden deren Schritte 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*

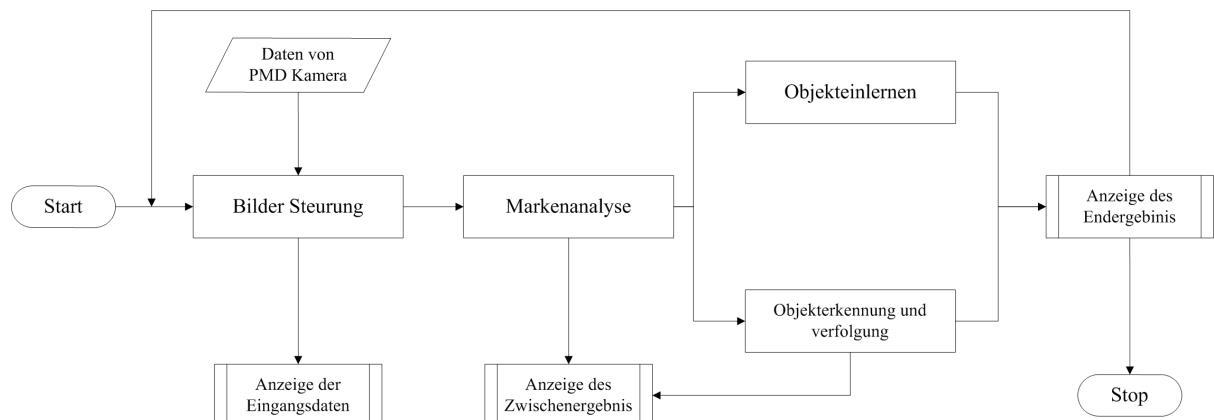


Abbildung 4.1: Ablaufdiagramm

analysiert die eingegebenen Bilddaten und versucht die Marken zu finden und zu verfolgen. Die gefundenen Marken werden entweder von dem *Objekteinlernen* oder durch die *Objekterkennung und Verfolgung* benutzt, das am Anfang des Programms durch den Benutzer entschieden wird. In der Einlernphase 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 vom neuen Objekt zu den vorhandenen charakteristischen Graphen verglichen, die nach dem Objekteinlernen im Speicher hinterlegt werden. Das Ergebnis wird in einem OpenGL Fenster angezeigt 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

Zuerst müssen alle Marken aus den Eingabebildern entweder für das Objekteinlernen oder die Objekterkennung und Verfolgung erkannt werden. In diesem Abschnitt werden alle dafür benötigten Verfahren und Algorithmen vorgestellt. Die Objekte sollen aus den Eingabebildern vor dem Lernen segmentiert werden, damit die Störung von der Umgebung vermieden werden kann. Danach vergrößert die Helligkeitssteuerung den Unterschied zwischen den Marken und anderen Bildpunkten. Daraufhin wird der STAR Detektor durchgeführt. Dadurch können viele Merkmale über die künstlichen Marken erkannt werden. Da der Detektor keine 1 zu 1 Korrespondenz zwischen den realen Marken und den erkannten Merkmalen garantiert, ist die zusätzliche Kombination der Merkmale für gleiche Marke notwendig. Die Segmentierung verteilt die Marken auf die verschiedenen Objekte und die Verfolgung liefert die Abhängigkeit der Marken zwischen den unterschiedlichen Bildern.

### 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 Klassen **BildData** gespeichert. Dazwischen sind die Amplituden und 3D-Koordinaten von hoher Relevanz 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 von Daten sich von anderen unterscheiden, ist vor der weiteren Bildverarbeitung dieser Arbeit die Übereinstimmung beider Daten für jeden Pixel notwendig. Die Klasse **PMDPoint** wird nun definiert, um diesen Zweck zu erfüllen. Die Klassendiagramme

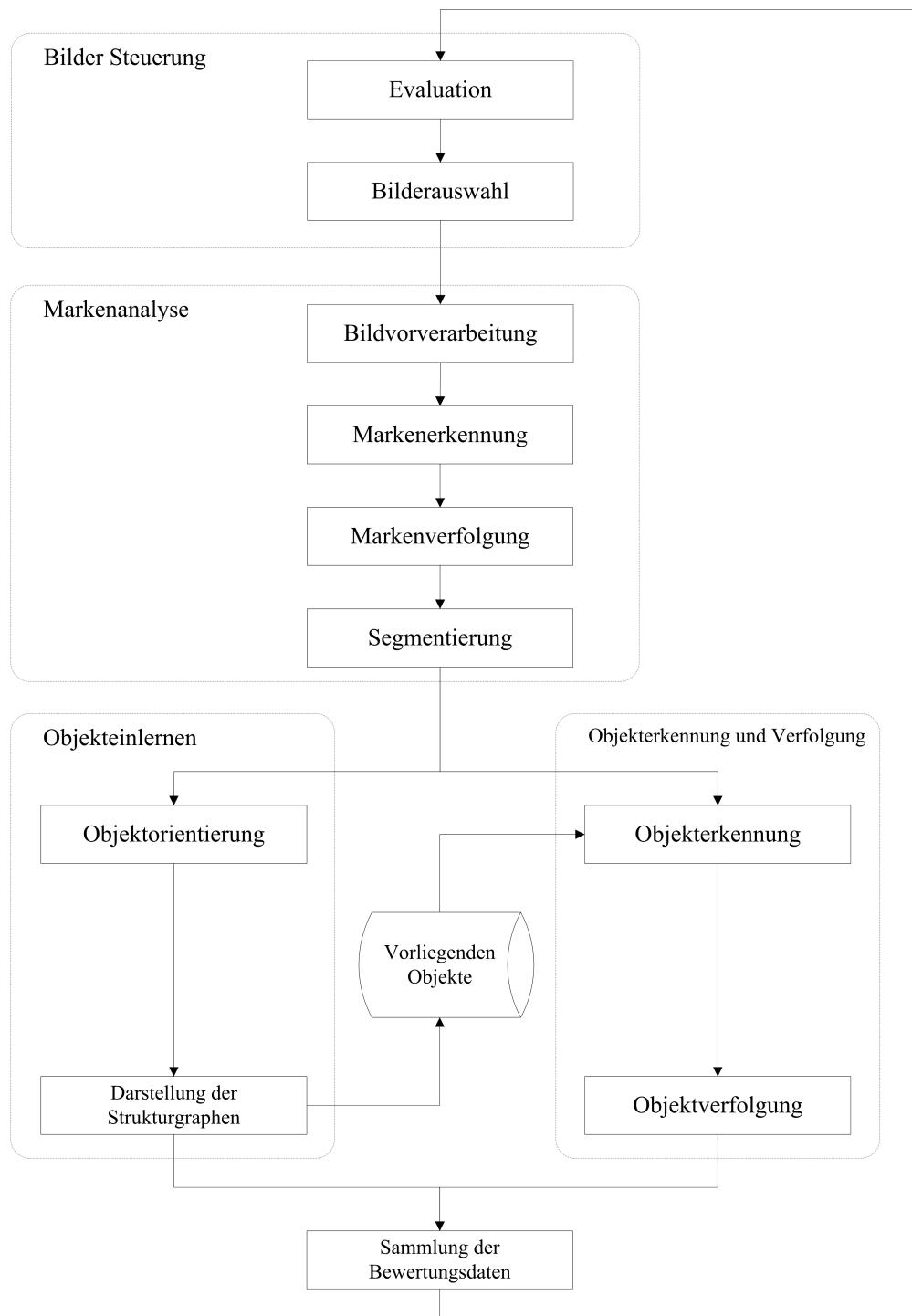


Abbildung 4.2: Ausführliches Ablaufdiagramm

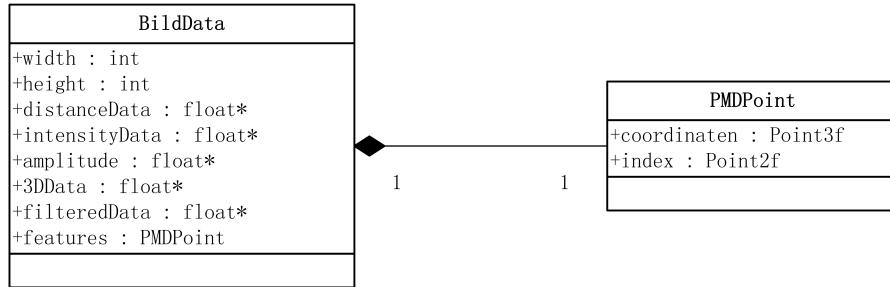


Abbildung 4.3: Die Klassendiagramme für **BildData** und **PMDPoint**.

beider Klassen werden in der Abbildung 4.3 gezeigt.

#### 4.1.1.2 Abstand Filter

Die Verbesserung der Ergebnisse der Segmentierung des fokussierten Objekts aus der Umgebung wurde im Abschnitt 3.2.1 bemerkt. Deshalb soll am Anfang der Markenanalyse ein 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 Tiefenbild der Szene erzeugt werden kann. Der Pseudocode wird in Algorithmus 4 gezeigt.

---

#### 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
       $d_j \leftarrow \frac{1}{2}(d_j + e.3DKoordinaten_{jz})$ 
    end for
  end if
end for
  
```

---

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

grund bestimmt werden. Wenn der Abstand zwischen einem Pixel und dem Hintergrund größer als der vordefinierte Schwellwert 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 Bildpunkte wird gleichzeitig abgezählt, damit eine boolesche Ausgabe über die Verschiedenheit mit dem gefilterten Bild zusammen zurückgegeben werden kann. Der genaue Ablauf wird in Algorithmus 5 beschrieben.

---

**Algorithm 5** Filterungsphase des Abstand-Filters
 

---

```

Schwellwert für Abstandsvergleich:  $\epsilon$ 
Schwellwert 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
```

---

## 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 ist. Die Testmarken werden als weiße Punkte auf einem schwarzen Papier gedruckt. Es gibt insgesamt sieben verschiedene Größenstufen. Die Marken von jeder Stufe werden jeweils durch Quadrate und Kreise dargestellt. Abbildung 4.4 zeigt die unterschiedliche Erkennungsergebnisse der Marken, wenn das Objekt zwar an verschiedenen Positionen aber auf der gleichen Höhenebene liegt. Abbildung 4.5 zeigt den Einfluss auf die Ergebnisse von der Distanz zu der Kamera. Hierbei wird deutlich, dass, je näher das Objekt zu der Kamera angebracht wird, desto kleinere Markengrößen benötigt werden. (folgt aus dem Kapitel 3.1.2.3)

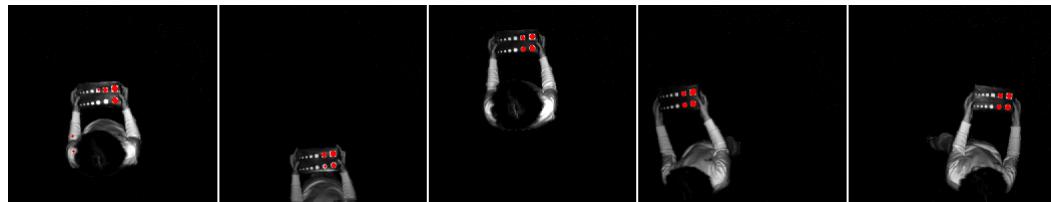


Abbildung 4.4: Die Erkennungsergebnisse der Marken für unterschiedliche Positionen.

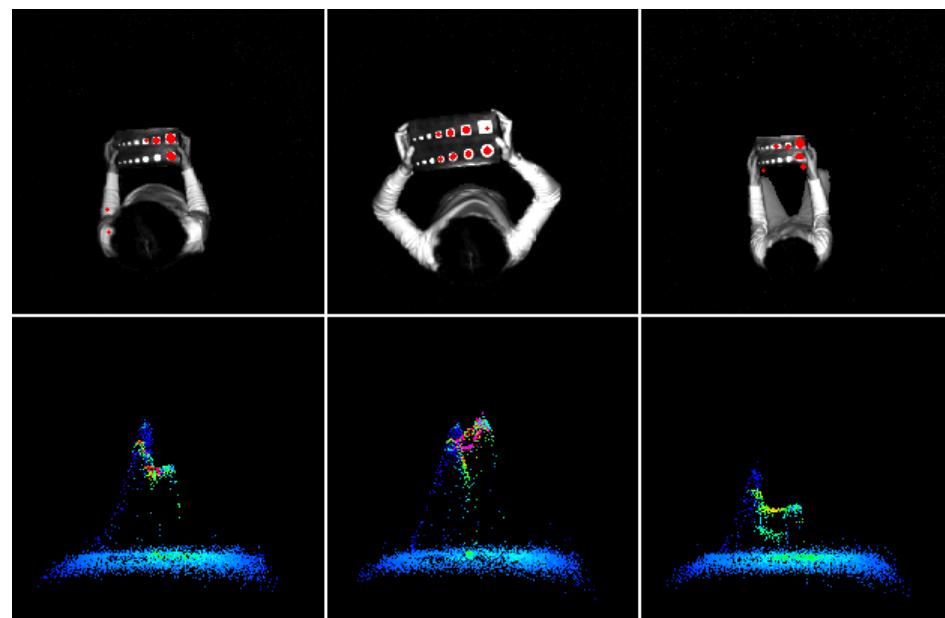


Abbildung 4.5: Die Erkennungsergebnisse der Marken für unterschiedliche Distanzen zur Kamera. Zu der ersten Reihe liegen die Graustufenbilder mit roten erkannten Marken; die entsprechenden 3D-Bilder in der zweiten Reihe zeigen die vertikalen Positionen des Objekts.

#### 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](#)] weiter deutlich erklärt. Der Parameter *maxSize* definiert die maximale Größe der Marken. *responseThreshold* ist ein Schwellwert über die Antwort von approximierten Laplacian. Die erkannten Marken mit niedrigerer Antwort als diesen Schwellwert 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 wird ResponseThreshold als der variable Parameter benutzt, d.h., dass 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** (siehe Kap.4.1.2.4 und Algorithmus 7). Die Kontraststeuerung überprüft, ob zufriedenstellend so viel Marken erkannt werden. Eine Vorhersage über Kontrast bzw. ResponseThreshold für die nächste Schleife wird gleichzeitig erzeugt, wenn es zu wenig oder zu viel Marken gefunden werden.

Die Punktmenge, die durch STAR-Detektor erkannt wird, hat leider keine 1 zu 1 Korrespondenz zu den realen Marken, die auf der Oberfläche des Objektes angebracht werden. Daraus bedeutet, dass eine Marke des Objektes von dem Detektor häufig als viele verschiedene Merkmale erkannt wird. Deshalb ist eine Kombination der Merkmale nötig, die zur gleichen Marke gehören. Dieser Prozess kann in zwei Schritten zusammengefasst werden. Zuerst werden alle Merkmale durch den Algorithmus DBSCAN (siehe Kapitel 3.6.1) in viele Gruppen verteilt. Daraufhin wird für jede Gruppe der zentrale Punkt berechnet, der als einziges Merkmal eine reale Marke beschreibt.

#### 4.1.2.4 Steuerung der Helligkeit und des Kontrast

Die Helligkeit bzw. der Kontrast kann die Ergebnisse der Markenerkennung stark beeinflussen. Eine bekannte Arbeit über die radiometrische Kalibrierung wurde von Debevec und Malik in [[Debevec & Malik 2008](#)] veröffentlicht. Ihr Verfahren ist robust und präzise, benötigt aber viele Bilder der gleichen Szene mit verschiedenen Belichtungszeiten,

---

**Algorithm 6** Markenerkennung

---

```

Eingabebild:  $I$ , maximale Schleife:  $N$ 
 $a \leftarrow$  Anfangswert für Kontrast
 $r \leftarrow$  Anfangswert für ResponseThreshold
 $b \leftarrow$  feste Helligkeit
for  $i = 1 \rightarrow N$  do
    Eingabe für STAR Detektor:  $H_i \leftarrow aI + b$ 
    Erkannte 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 return  $P$ 

```

---

um alle Informationen der Szene für einen jeweils helleren Bereich bzw. dunkleren Bereich zu sammeln. Wegen der Einschränkung der Kamera kann diese Bedingung in unserer Arbeit leider nicht erfüllt werden. Außerdem ist das Verfahren von Debevec sehr rechenintensiv. nur die Bilddaten um Marken sollen aber in der Markenerkennung dieser Arbeit betrachtet werden, wodurch die sonstigen Informationen des Bildes einfach ignoriert werden können. Durch diese Grundidee kann ein Algorithmus erzeugt werden, dessen Endbedingung durch die Untersuchung der Anzahl der erkannten Marken eingestellt wird.

Der formulierte Durchlauf wird im Algorithmus 7 dargestellt. Die Anzahl der erkannten Marken ist die erste Stufe der Prüfnorm der Kontraststeuerung. Wenn nicht genug Marken erkannt werden, dann wird die Strahlungsenergie des Graustufenbildes als die zweite Stufe der Prüfnorm überprüft. Bei einer zu großen bzw. zu kleinen Strahlungsenergie verringert bzw. erhöht sich der Kontrast im erlaubten Intervall, welches als Eingangsparameter vorher definiert wird. Wenn die Strahlungsenergie zufriedenstellend ist, aber noch nicht genug Marken gefunden sind, nimmt das ResponseThreshold ab, damit mehr Marken mit schwächeren Antworten erkannt werden können. Falls zu viele Marken erkannt werden, wird der Wert von ResponseThreshold vergrößert. Bei dieser Situation wird die Strahlungsenergie nicht mehr betrachtet, damit der Algorithmus zur Konvergenz halten kann. Ein boolescher Wert wird von dem Algorithmus zurückgegeben, der 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 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 drei Verbesserungen für das Verfahren von Scott und Hignnis durchgeführt, damit das stabilere Ergebnis erzeugt werden kann. Die Qualität der Korrespondenzuntersuchung kann auch nach der Verbesserung bewertet werden.

#### Nebenbedingung für die Bestimmung der größten Elemente

In dem originalen Algorithmus werden die Punkte  $I_i$  und  $J_j$  genau dann als Korrespondenzpunkte erkannt, wenn das Element  $P_{ij}$  das größte Element von beiden Zeile  $i$  und Spalte  $j$  ist. Aber manchmal liefert das größte Element nicht die beste Korrespondenz, insbesondere in dem Fall einer großen Transformation zwischen zwei betrachteten Bildern. Die Lösung besteht darin, eine weitere Beschränkung für die Untersuchung der größten Elemente einzusetzen. D.h., dass die größten Elemente nur dann akzeptiert werden, wenn sie gleichzeitig größer als ein eingegebener Schwellwert  $\epsilon$  sind. Der Schwellwert  $\epsilon$  wird in  $[0, 1]$  definiert. Je höher  $\epsilon$  ist, desto ordentlicher sind die gefundenen Korrespondenzpunkte. In Idealfall sind alle größten Elemente  $P_{ij}$  gleich 1, z.B. wenn die beiden betrachteten Bilder identisch sind. Der Nachteil der Verwendung des Schwellwerts liegt darin, dass manchmal keine Korrespondenz zwischen zwei Bildern gefunden kann. Deshalb ist eine boolesche Ausgabe des Algorithmus notwendig, welches sich als eine wichtige Variable der Bildsteuerung darstellt (siehe Kapitel 4.5). Das Programm kann durch diese Variable erkennen, ob die Korrespondenzuntersuchung erfolgreich durchgeführt wird.

#### Qualitätsüberprüfung der Korrespondenz

Neben der binären Behauptung des Algorithmus ist auch die Bewertung der Korrespondenzqualität wichtig, damit die gute und stabile Bilderkette für Markenverfolgung ausgewählt werden kann. Das wurde aber leider in der Arbeit von Scott und Hignnis nicht genannt. In der ersten Verbesserung wird die Größe der größten Elemente von Matrix  $P$  mit einem Schwellwert weiter beschränkt, um ein besseres Korrespondenzergebnis zu erhalten. Deshalb kann zur Umkehr 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 arithmetisches Mittel benutzt, weil die Anzahl der betrachteten Punkte, die als Eingaben in den Algorithmus eingegeben werden, auch ein wichtiger Faktor der Bewertung der Korrespondenz sind.

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

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

Der verbesserte Algorithmus wird im Algorithmus 8 gezeigt.

#### 4.1.4 Segmentierung

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

## 4.2 Objektlernen

Nach der erfolgreichen Markenanalyse wird 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

---

**Algorithm 8** Korrespondenzuntersuchung(*Punktpaare Result*)

---

Eingabebilder von jeweils Zeitpunkt  $t_{i-1}$  und  $t_i$ :  $I, J$   
Aktuelle approximierte Abstandseinheit:  $\sigma_i$   
Schwellwert 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

```

---

danach für unterschiedliche Objekte weiter verteilt. Im diesen Abschnitt wird erklärt, wie ein charakteristischer Graph des Objektes mithilfe der erkannten Marken dargestellt werden kann. In dieser Arbeit wird im Objektlernen die vereinfachte Situation berücksichtigt, dass ein Objekt nur einmal in den Eingabebildern vorkommt. Deshalb wird nur die größte Punktmenge des Segmentierungsergebnisses für das Objektlernen ausgewählt, die als die Eingabe für die folgenden Arbeitsschritte genutzt wird.

### 4.2.1 Bestimmung der Orientierung

Um ein Objekt zu lernen, muss dieses zuerst mit Marken zu der Kamera gezeigt und langsam umgedreht 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 das 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 (siehe Formel (3.22)). Dann berechnet man die Vektoren, die von allen Marken zu ihren entsprechenden Schwerpunkten reichen. Als Drittes wird die Korrelationsmatrix  $H$  durch Formel (3.33) mit diesen Vektoren bestimmt. Die Hilfsmatrix  $N$  in der Formel (3.35) besteht aus den Elementen der Matrix  $H$ . Einer ihrer Eigenvektoren ist genau dann die Einheitsquaternion für die Rotation, wenn dieser dem größten positiven Eigenwert entspricht. Die Rotations- bzw. Translationsmatrix unter kartesischem Koordinatensystem wird jeweils durch die Formel (3.18) und (3.27) berechnet.

Die Lage des Objekts in jedem Bild hängt nur von der Lage des Objekts in dem vorherigen Bild ab. Die kleinen Fehler der Orientierung zwischen den nachfolgenden Bildern werden in einer langen Bildsequenz akkumuliert, das häufig einen großen Unterschied zwischen der realen Lage und berechneten Lage des Objekts erzeugt. In dieser Arbeit wird der Kalman-Filter über die Translation des Schwerpunktes des Objekts benutzt, um die kumulative negative Wirkung jedes Schrittes zu vermindern. Die Grundlage des Kalman-Filters wird im Abschnitt 3.3.3 aufgeführt. Die Übergangsmatrix des Schwerpunkts des Objekts im dreidimensionalen 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 Beob-

achtungsmatrix ist die Einheitsmatrix mit der Größe  $3 \times 3$ , weil der Schwerpunkt des Objekts  $\in R^{3 \times 1}$  als Eingabe zur Korrekturphase eingegeben wird. OpenCV liefert eine komplette Implementierung des Kalman-Filters, was hier direkt verwendet wird.

### 4.2.2 Markenanordnung

Wie im Anfang des Abschnittes 4.2.1 beschrieben, soll während der Lernphase die Lage des betrachteten Objektes in jedem Bild bestimmt werden. Hierzu benötigt man mindestens in zwei Bildern jeweils drei nicht kollineare Punkte des Objektes, damit die Transformation zwischen der Lage dieses Objektes in diesen zwei Bildern bestimmt werden kann. Je mehr die Punkte betrachtet werden, desto exaktere Transformationen können in der Theorie berechnet werden. Wegen der Auflösung der Kamera in dieser Arbeit werden aber 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 möglichst 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 sein.
3. Jedes Tripel der Marken soll nicht kollinear sein.
4. Um die Grenzen zweier Ebenen des Objekts sollen mehr Marken als in Mitte angebracht werden.
5. Der Strukturgraph jeder Ebene soll möglichst 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 immer durch drei beliebige Marken berechnet werden kann. Die in der ersten Regel definierte minimale erkennbare Markengröße entspricht aber nur der Situation, in der die Marken mit der Bildebene der Kamera parallel sind. Wenn sich ein Objekt umdreht, verändert sich der Winkel zwischen der aktuell beobachteten Ebene und der Bildebene. Deswegen werden die gleichen Marken aber mit weniger Pixel in der Kamera abgebildet. Die Größen der Marken auf dieser beobachteten Ebene erscheinen immer kleiner in dem Bild mit der Vergrößerung des Drehwinkels, bis sie komplett senkrecht zu der Bildebene liegen und die nächste Ebene vollständig unter der Kamera vorkommt. Je kleiner die Marken sind, desto schwieriger werden sie erkannt. 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 der vierten Regel erklärt wird. Die Asymmetrie von der

fünften Regel hält die Einzigartigkeit der Transformation zwischen zwei unterschiedlichen 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 eingegebenes Objekt erzeugt wird, welcher als die Charakteristiken für die Wiedererkennung verwendet werden kann. Um das Ziel zu erreichen, sollen die Strukturgraphen stabil sein und genug charakteristische Information des Objektes enthalten. Die Stabilität bedeutet, dass die Strukturgraphen für ein Objekt bei öfterem Lernen gleich dargestellt werden sollen. Die Information, die zur Wiedererkennung notwendig ist, wird durch die Positionen der Marken bzw. die Abstände dazwischen erzeugt. Die Verfahren der Bestimmungen der charakteristischen 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 falsch erkannten Merkmalen des Objektes bzw. der Umgebung oder den schlechten Korrespondenzpunkten verursacht. Deshalb benötigt das Programm eine Strategie für die Auswahl der gültigen Marken. In dieser Arbeit basiert die Auswahl auf Erscheinungshäufigkeiten, d.h. nur die Marken, die mehrmals kontinuierlich vorgekommen sind, werden zu den gültigen Marken gerechnet und in den Strukturgraphen eingefügt. Diese Marken werden als stabile Knoten in dem Strukturgraphen markiert, und dürfen nicht verändert werden. Abgesehen von 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 genau dann identisch, wenn der Abstand von einer Marke zu dem Punkt, der von anderer Marke nach Transformation erzeugt wird, kleiner als ein vorgegebener Schwellwert ist. Die Transformation umfasst die Rotation- bzw. Translationsmatrix, die mit dem Verfahren von Abschnitt 4.2.1 bestimmt werden.

#### 4.2.3.2 Kanteneinfügung

Die ungerichteten Kanten des Strukturgraphen speichern die Abstände zwischen den Knoten des Graphen, welche die wichtigste Eigenschaft für die Wiedererkennung ist. Die Knoten, die gleichzeitig beobachtet werden können, werden in einem vollständigen Graphen mit den anderen verbunden. Wenn irgendwann ein neuer Knoten in den Graphen eingefügt wird, verbindet er 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 wird.

---

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

---

Schwellwert 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 aller mit  $v_{temp_j}$  verbundenen Knoten in  $G_{temp}$  mit  $v_i$

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

**end if**

**end for**

**if** Kein entsprechender 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**

Einfügen aller übrigen Knoten von  $G_{temp}$  in  $G$

---

## 4.3 Zugriff des Strukturgraphen von Datei

Das Ergebnis des Objektlernens ist ein Strukturgraph, der die Charakteristiken des betrachteten Objektes beschreibt. Die Knoten des Graphen werden als dreidimensionale Punkte mit Gleitkommazahl definiert. Jeder Knoten enthält eine Map, in der seine Nachbarn und die Kantenlänge dazwischen paarweise gespeichert werden. Die Strukturgraphen sollen auf der Festplatte gespeichert und eingelesen werden können. In dieser Arbeit wird die Dateiform von VTK ([[Visualization Toolkit](#)]) benutzt, welche 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 dem anderen Framework bzw. der Software verwendbar sind. Die Abbildung 4.6 ist der Screenshot von der Visualisierung eines Strukturgraphen mit der Software ParaView [[ParaView](#)]. Vor dem Speichern des Graphen werden alle Knoten noch einmal überprüft. Die ähnlichen Knoten sollen in einem einzigen Knoten kombiniert werden, damit nur ein vereinfachter Graph ohne verdoppelten Knoten gespeichert wird. Die VTK-Dateiform liefert viele Stichwörter für Beschreibung der unterschiedlichen grundsätzlichen, geometrischen Elemente, wie z.B. die Punkte, die Gerade und die Ebene, usw. Deshalb werden die Knoten und die Kanten des Strukturgraphen jeweils mit dem Stichwort „POINTS“ und „LINES“ in der VTK-Datei geschrieben. Diese Stichwörter sind auch die Kennzeichen beim Lesen der VTK-Datei. Die Punkte werden zuerst in einen neuen Graphen eingefügt, und dann mit anderen Knoten durch die gespeicherten Gerade verbunden.

## 4.4 Objekterkennung und Verfolgung

Die Strukturgraphen aller vom Programm erlernten Objekte sollen am Anfang der Wiedererkennungsphase von den auf der Festplatte gespeicherten VTK-Dateien wieder in das Programm eingelesen werden. Danach wird die Markenanalyse über die Eingabebilder genau wie in der Lernphase durchgeführt, und am Ende wird eine Liste von der Punktmenge erzeugt. Das Programm versucht dann, die Teilgraphen dieser Strukturgraphen aus den von den Punktmengen neu dargestellten Graphen zu finden, was im sogenannten Teilgraph-Isomorphismus-Problem zusammengefasst wird. Für ein Bild werden alle möglichen Kombinationen der Strukturgraphen und Punktmengen dieses Bildes durchgesucht. Die Objekte, deren Teilgraphen aus dem Eingabebild gefunden wurden, werden als erkannt angenommen. 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, was aber nicht möglich ist, ist die kontinuierliche Festlegung der entsprechenden Punktmengen, weil die Marken von jedem Bild neu segmentiert werden und keine Verbindungen zwischen den „gleichen“ Punktmengen von unterschiedlichen Bildern existieren. Ein direktes Lösungsverfahren besteht darin, dass jede Punktmenge aus der Segmentierung sofort als ein neues Objekt erkannt wird. Der Vergleich des Teilgraph-Isomorphismus-

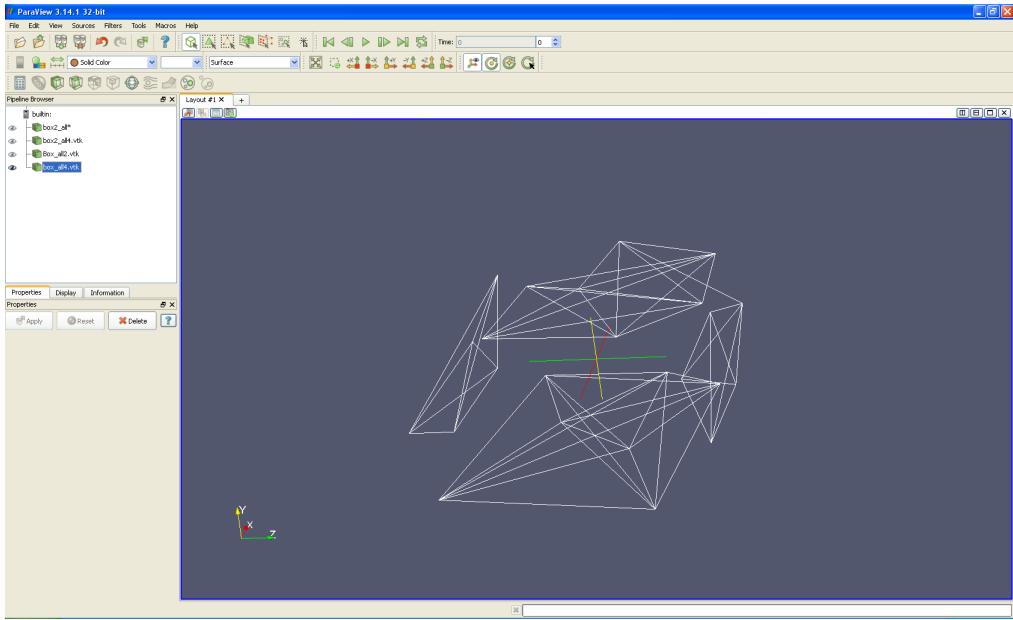


Abbildung 4.6: Visualisierung eines Strukturgraphen eines Kästchens mit ParaView. Das Foto des Kästchens ist in Abbildung 5.1 dargestellt.

Problems findet dann zwischen den Strukturgraphen zweier Objekte statt. Dadurch sollen viele Lernprozesse während der Wiedererkennungsphase gleichzeitig durchgeführt werden. Das kostet aber zu viel Zeit, und ist schwierig zu implementieren. Die alternative Lösung liegt in der Darstellung der sogenannten Kandidaten. Diese Kandidaten interessieren nur die Abhängigkeiten der gleichen Punktmengen von unterschiedlichen Bildern, enthalten aber keine stabilen Knoten wie den Strukturgraphen. Die Abbildung 4.7 zeigt ein Beispiel über die gesamte Erkennungsphase.

#### 4.4.1 Kandidaten der Objekterkennung

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

gespeichert. In dem Block „Behandlung der Kandidaten“ von Abbildung 4.7 wird ein Beispielablauf über die Aktualisierung bzw. Darstellung der Kandidaten aufgeführt. Die neu erzeugten Kandidaten werden in dem Ende der Kandidatenliste eingefügt. Der zweite und dritte Kandidat hat in dem Beispiel keine entsprechenden Punktmenge und wird nach der Aktualisierung aus der Liste entfernt. Nach der Behandlung der Kandidaten sollen nur drei Kandidaten im Teilgraph-Isomorphismus mit dem Eingabemodell verglichen werden.

#### 4.4.2 Objekterkennung

Die Objekterkennung basiert auf dem Vergleich zwischen den Strukturgraphen der erlernten Objekte und den Kandidaten, die von den neu segmentierten Punktmengen erzeugt werden. Ein neuer Graph wird zuerst von den Punkten eines Kandidaten dargestellt. Der Vergleich zwischen diesen Kandidaten und den vorhandenen Strukturgraphen kann dann zum Teilgraph-Isomorphismus Problem abgeleitet werden. Das Lösungsverfahren übernimmt die Idee von Rhijn und Mulder [Rhijn & Mulder 2005], die bereits im Abschnitt 3.6.2 erklärt wurde. Es werden zuerst die isomorphen Knoten zwischen zwei Graphen herausgefunden, welche jeweils genug isomorphe Nachbarn haben, die in gleichen Abständen zu den isomorphen Knoten liegen. Ein Beispiel der isomorphen Knoten wird in Abbildung 4.8 gezeigt. Der Teilgraph-Isomorphismus zwischen diesen Graphen existiert genau dann, wenn mindestens ein gefundener isomorpher Knoten und seine Nachbarn zusammen die gleichen Pyramiden darstellen können. In dieser Arbeit wird das Verfahren von Rhijn und Mulder in zwei Bereichen verbessert: Die effizientere Nebenbedingung für das Suchen der isomorphen Knoten und die vereinfachte Endbedingung anstatt der Feststellung der Pyramide.

##### Nebenbedingung für das Suchen der isomorphen Knoten

Um die isomorphen Knoten schneller zu finden, werden zwei Quoten für das Suchen definiert. Die Abstandsquote wird anstelle des absoluten Schwellwertes 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 sich das Objekt zur Kamera bewegt. Je größer die Auflösung des Objektes ist, desto besser ist das Erkennungsergebnis. Hieraus folgt, dass die absolute Kantenlänge von der Distanz zwischen dem Objekt und der Kamera abhängt. Analog dazu liefert der gleiche absolute Schwellwert für verschiedene Abstände zur Kamera aber unterschiedliche Vergleichsgenauigkeiten. Die Verwendung der Abstandsquote kann die Wirkung der Entfernung der Kamera gut vermeiden.

Die zweite Nachbarquote beschränkt die minimale Anzahl der benötigen, assoziierten Nachbarn zur 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

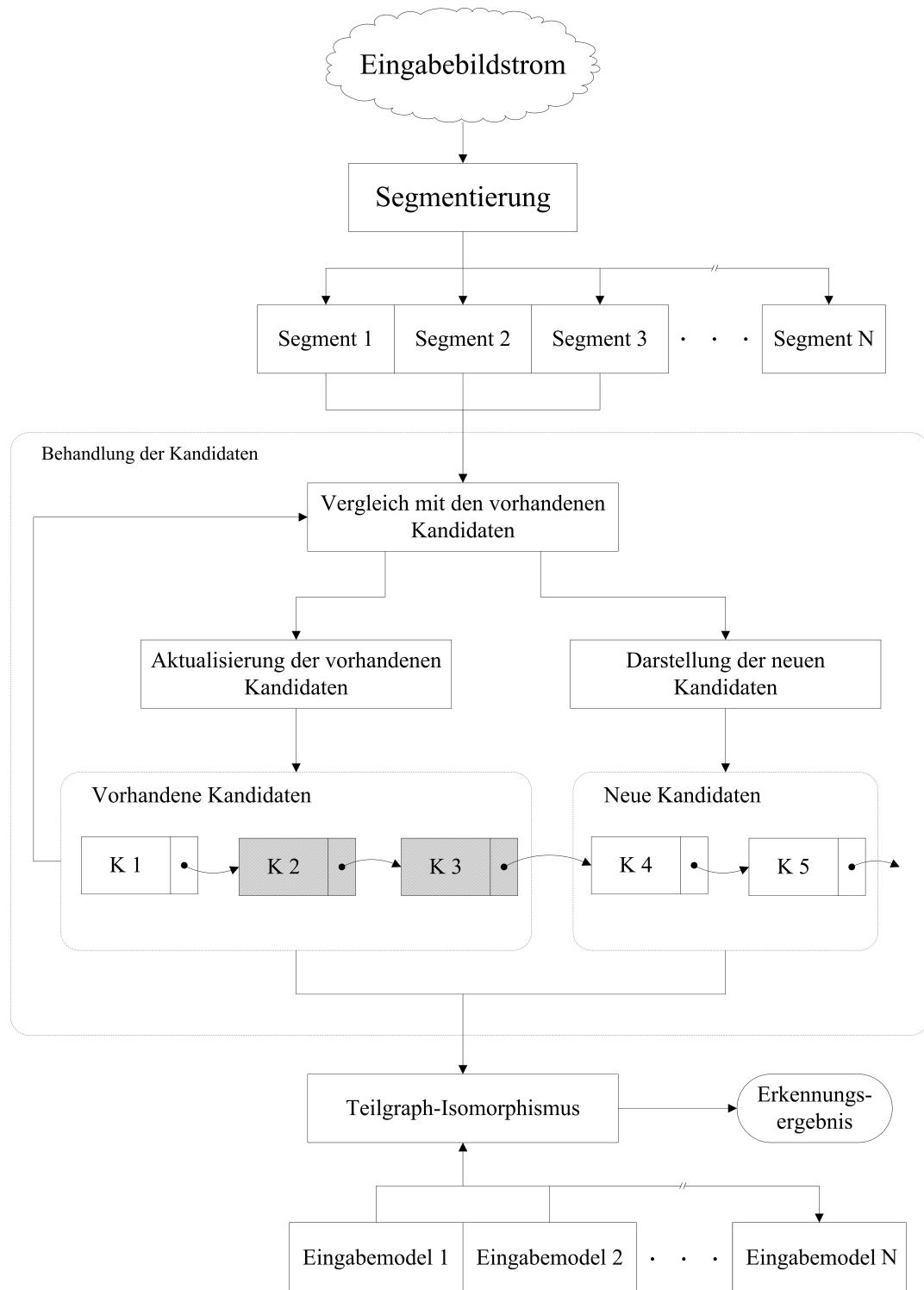


Abbildung 4.7: Ausführliches Ablaufdiagramm der Objekterkennung. Die Kandidaten, die schraffiert markiert sind (K2, K3), werden nach der Aktualisierung aus der Liste entfernt.

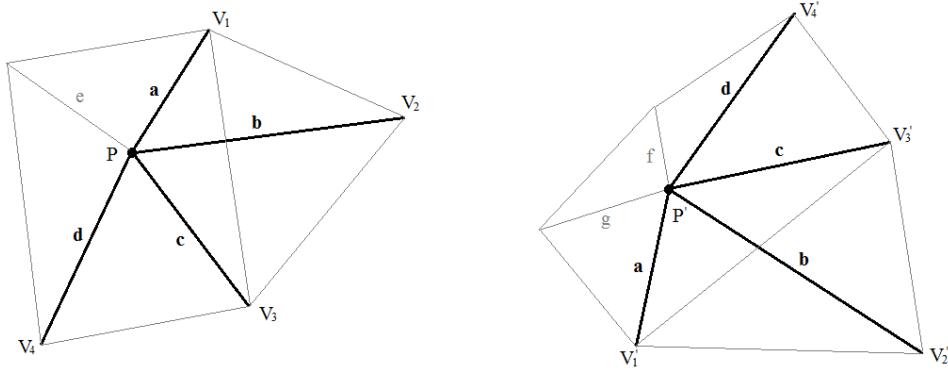


Abbildung 4.8: Ein Beispiel der isomorphen Knoten.  $P$  und  $P'$  sind zwei isomorphe Knoten, die jeweils vier Nachbarn mit gleicher Abstände haben.  $((P, V_1) = (P', V'_1), (P, V_2) = (P', V'_2), (P, V_3) = (P', V'_3), (P, V_4) = (P', V'_4))$

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

formuliert werden, wobei  $V$  die Knotenmenge des Graphen beschreibt. Die minimale Anzahl der Nachbarn darf nicht kleiner als drei sein, weil es mindesten drei Nachbarn benötigt, um einen einzigen Knoten eines Graphen durch den Vergleich der Kantenlänge seiner Nachbarn im dreidimensionalen Raum zu bestimmen. Diese Veränderung liefert eine bessere Toleranz für die Erkennung. Wenn die Graphen nur mit weniger Knoten als die Eingaben des Verfahrens angegeben werden, garantiert das Verfahren aber auch genug Nachbarn für die Auswahl der isomorphen Knoten. Außerdem können die Genauigkeit und die Stabilität des Teilgraph-Isomorphismus mit dem Vergrößern der Anzahl der Knoten gleichzeitig 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 vier Knoten. Deshalb sollen für einen Knoten, der  $n$  Nachbarn hat, höchstens

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

verschiedenen Kombinationen seiner Nachbarn betrachtet werden, um die Pyramide zu finden. Seien  $m$  die Anzahl der gefundenen isomorphen Knoten und der Zeitaufwand

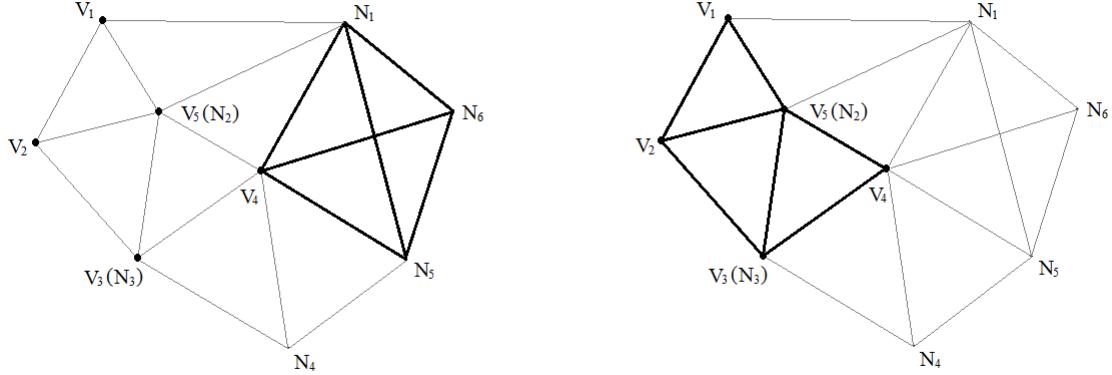


Abbildung 4.9: Ein Beispiel für den Vergleich der zwei Endbedingungen.  $V_1$  bis  $V_4$  sind die isomorphen Knoten und  $N_1$  bis  $N_6$  sind die Nachbarn von Knoten  $V_4$ . Auf der linken Seite wird das Verfahren mit der Pyramide gezeigt, wobei alle isomorphen Knoten durchgesucht werden, bis eine Pyramide  $V_4N_5N_6N_1$  gefunden ist. Auf der rechten Seite werden aber nur die Kanten zwischen den isomorphen Knoten betrachtet.

von der Überprüfung der Nachbarschaft der Zeitkomplexität  $\mathcal{O}(1)$ , dann ist der Zeitaufwand der Suche der Pyramide im 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, anstatt der Nachbarn eines isomorphen Knoten 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, wie z.B. das Dreieck sogar nur zwei verbundenen Kanten.  $m$  isomorphe Knoten werden durchgesucht, um zu bestimmen, ob sie sich zumindest mit zwei anderen isomorphen Knoten verbinden. Der Zeitaufwand dieses Ablaufes ist nur  $\mathcal{O}(m^2)$ , was deutlich kleiner als das Verfahren mit Suche der Pyramide ist. Abbildung 4.9 zeigt ein Beispiel für den Vergleich der beiden Nebenbedingung des Isomorphen-Problems. Auf 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 isomorphen Knoten selbst betrachtet, wobei  $m^2 = 5^2$  Zeitaufwand kostet. Alle isomorphen Knoten, die diese Bedingung erfüllen, werden dann gespeichert, um die Orientierung des erkannten Objektes zu berechnen.

### 4.4.3 Bestimmung der Orientierung

Die Verfolgung des Objekts ist die andere wichtige Aufgabe dieser Arbeit, was durch die Berechnung der Transformation realisiert wird. Die Anfangslage eines Objekts wird von dem Strukturgraphen definiert. In jedem Bild wird die Rotations- bzw. die Translationsmatrix zwischen dem Anfangs- und aktuellen Zustand mithilfe der ausgewählten isomorphen Knoten bestimmt. Das Orientierungsverfahren verläuft in gleicher Weise wie das Verfahren im Lernprozess, der im Abschnitt 4.2.1 erklärt wurde.

## 4.5 Bildersteuerung

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

In dieser Arbeit werden alle Eingabebilder zuerst in einer Warteschlange gespeichert. Die Warteschlange wird mit fester Größe definiert. Wenn ein neues Bild eingegeben wird, wird es am Ende der Schlange eingefügt und das älteste Bild aus der ersten 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 Rotations- bzw. Translationsmatrix werden nicht direkt für die Aktualisierung des Strukturgraphen benutzt, sondern zuerst geprüft, 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 hochqualitative 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 darin, dass der Prüfer die Lebenszeit der Marken nicht betrachtet, sondern nur die neu gefundenen Marken zählt, die mit den vorhandenen Knoten des Strukturgraphen nach gerade berechneter Transformation identisch sind. Nur das Bild mit genug Marken, die die obige Bedingung erfüllen, wird von dem Prüfer akzeptiert.

Die Beurteilung über die Anzahl der Marken wird durch den Vergleich der Quote, die den Anteil der mit vorhandenen Knoten übereinstimmenden Marken an allen neuen Marken beschreibt, mit einem vordefinierten Schwellwert realisiert. Die vom Prüfer akzeptierten Bilder können direkt für die Aktualisierung der Strukturgraphen genutzt werden. Die anderen Bilder sind zwar überflüssig, können aber nicht direkt verworfen werden. Zu einer schlechten Beobachtungssituation liefert die Kamera möglicherweise in langer Zeit gar keine hochqualitativen Bilder. Wenn das Programm all diese Bilder überspringt, wird die Verfolgung des Objekts abgebrochen und falsche Erkennungsergebnisse erzeugt. Ein extremes Beispiel ist bei der Drehung des Kästchens gegeben. 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 beschränkt. Wenn kein hochqualitatives Bild gefunden werden kann, wird ein verhältnismäßig besseres Bild aus den schlechten Bildern ausgewählt. Die Summe aller größten Elemente von  $P$ , die im Abschnitt **Qualitätsüberprüfung 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

In diesem Abschnitt wird das Programm von verschiedenen Richtungen ausgewertet. Zwei schwarze Kästchen mit weißen Marken werden als Testobjekte in der Evaluation verwendet (siehe Abb. 5.1). Jede Translation bzw. Rotation des Objektes wird manuell durchgeführt, d.h. die kleine Schwingung von menschlicher Bewegung ist in unserem Test auch betrachtet.



Abbildung 5.1: Die Testobjekte. Das größere Kästchen auf der linken Seite ist das Haupttestobjekt, was in den meisten Evaluationen über ein Objekt verwendet wurde.

### 5.1 Teilweise Evaluation

Um die bestmöglichen Ergebnisse zu erhalten, werden viele Hilfsteilprogramme neben dem Lernen- bzw. Wiedererkennungsprozess gleichzeitig durchgeführt. Die Wirkungen und der zusätzlicher Zeitaufwand dieser Teilprogramme werden in diesem Abschnitt separat diskutiert.

### 5.1.1 Abstands-Filter

Das Ziel des Abstands-Filters ist, die interessanten Objekte aus der Umgebung zu extrahieren. Dadurch kann der Detektor nur den kleinen Bereich um die Objekte fokussieren, wodurch Störungen der Umgebung vermieden werden können. Wegen der starken Abhängigkeit zwischen dem Detektor und dem Abstands-Filter, kann der Einfluss des Filters durch die Anzahl der erkannten Marken bewertet werden. Hier werden aber zwei Testbildströme für ein bewegendes Objekt mit unterschiedlichen Hintergründen verwendet, um die Wirkung des Abstands-Filters zu erklären. Die Screenshots für je 30 Bilder wird in Abbildung 5.2 bzw. 5.4 gezeigt. Insgesamt acht Marken werden auf der Ebene aufgebracht, die von der Kamera als weiße Kreise aufgenommen werden sollen. Die Marken, die von dem Programm erkannt werden können, werden dann wieder in rot gefärbt.

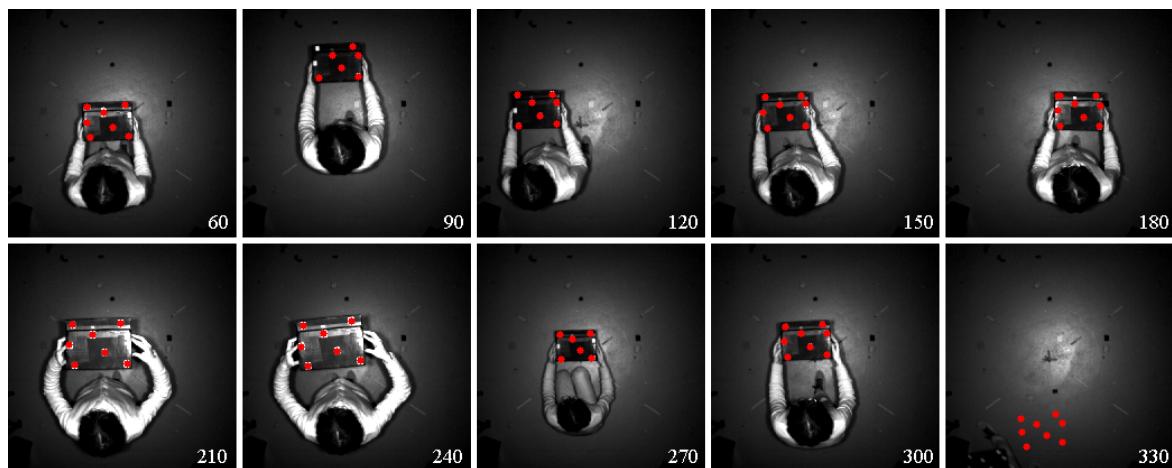


Abbildung 5.2: Der Bildstrom mit homogenem Hintergrund. Von links nach rechts und oben nach unten sind die Screenshots der Bilder, deren Index ab 60 beginnt und deren Schrittweite 30 beträgt.

Abbildung 5.3 zeigt die Anzahl der erkannten Marken aus dem Bildstrom mit homogenem Hintergrund. Die blaue und die grüne Kurve zeigen jeweils das Erkennungsergebnis mit und ohne Abstands-Filter. Die Kurve der Erkennungsergebnisse mit Abstands-Filter schwingt zwischen dem Intervall von sechs bis neun, was aber deutlicher schwächer als die Grüne ist. D.h, dass die Verwendung des Abstands-Filters die Erkennungsergebnisse verbessert und die Ausgaben viel stabiler sein lässt. Die Vergleichsergebnisse für den Bildstrom mit inhomogener Umgebung wird in der Abbildung 5.5 gezeigt. Die Erkennungsergebnisse mit und ohne Abstands-Filter dieses Tests scheinen ein bisschen schlechter als die Ergebnisse in der Abbildung 5.3 zu sein. Die Verbesserung durch den Abstands-Filter ist dennoch deutlich.

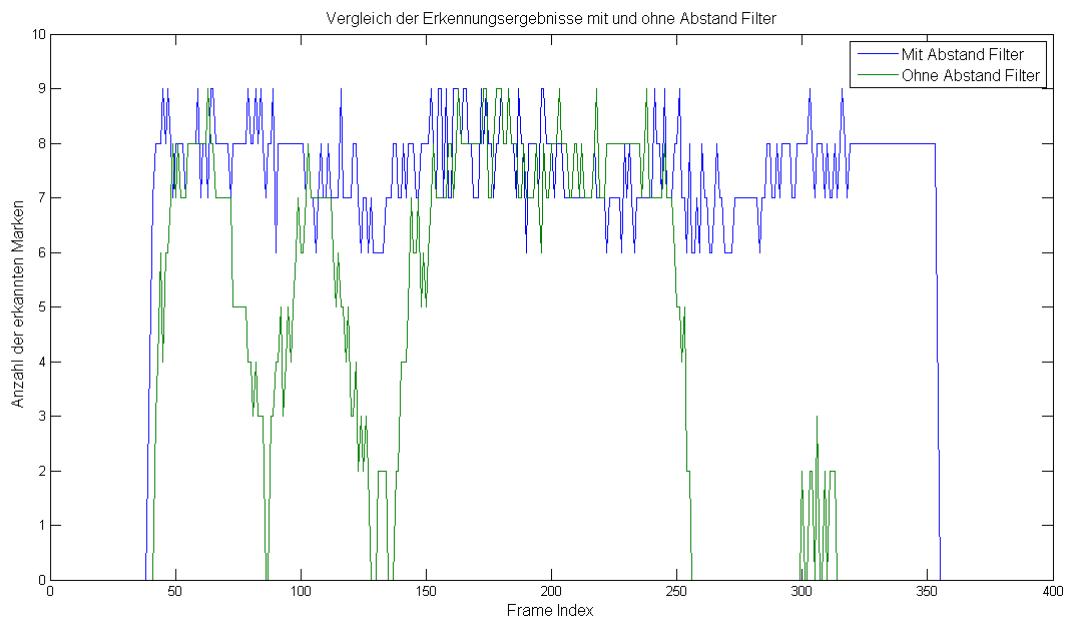


Abbildung 5.3: Der Vergleich der Erkennungsergebnisse mit und ohne Abstands-Filter. Die Eingabebilder enthalten nur ein Objekt und die homogene Umgebung, was in Abbildung 5.2 teilweise gezeigt wird.

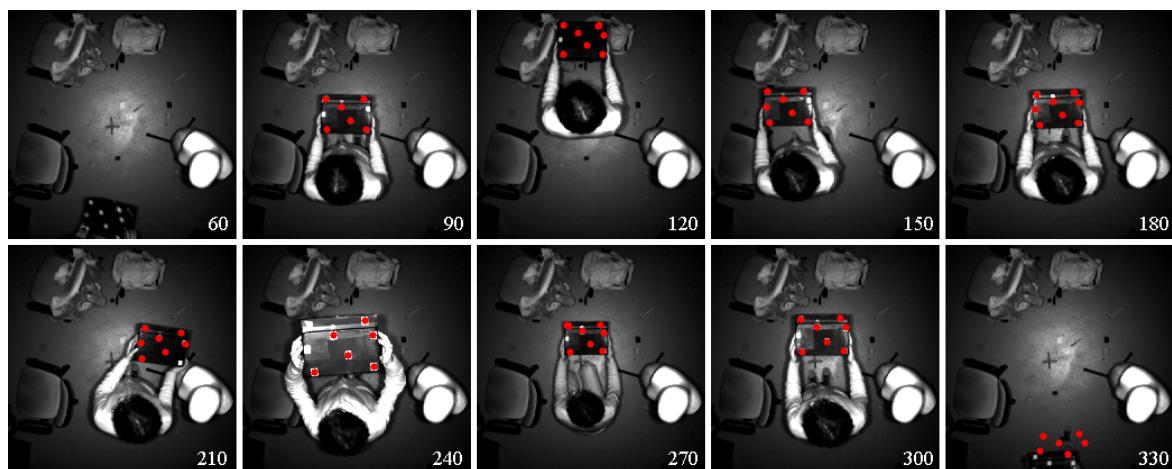


Abbildung 5.4: Der Bildstrom mit inhomogener Umgebung. Von links nach rechts und oben nach unten sind die Screenshots der Bilder, deren Index ab 60 beginnt und deren Schrittweite 30 beträgt.

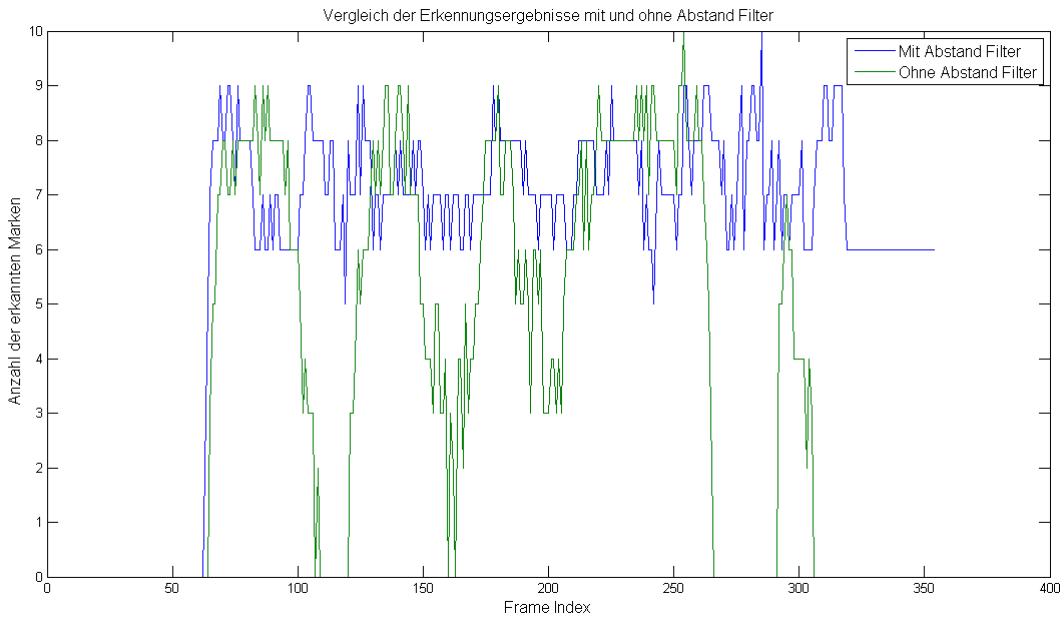


Abbildung 5.5: Der Vergleich der Erkennungsergebnisse mit und ohne Abstand-Filter. Die Eingabebilder enthalten mehr Objekte, was in Abbildung 5.4 teilweise gezeigt wird.

### 5.1.2 Helligkeitssteuerung

In der Bewertung der Helligkeitssteuerung wird der gleiche Testbildstrom verwendet, der in der Abbildung 5.2 aufgeführt wird. Die Abbildung 5.6 zeigt die Vergleichsergebnisse der Anzahl der erkannten Marken von dem Programm, das jeweils mit und ohne Helligkeitssteuerung durchgeführt wird. Die blaue Kurve ist die normale statistische Kurve für die erkannten Marken, welche identisch zu der blauen Kurve in Abbildung 5.3 ist. Die grüne Kurve, die die Erkennungsergebnisse ohne Helligkeitssteuerung beschreibt, schwingt zwischen dem Intervall des Bilderindexes von 200 bis 300 ziemlich stark. Diese unregelmäßige Schwingung liegt darin, dass das Objekt während diesen Bildern entlang der Blickrichtung der Kamera bewegt wird. Wegen dem Arbeitsprinzip der PMD Kamera steigt die Amplitude über das Objekt in dem Bild an, wenn das Objekt zu der Kamera bewegt wird. Das verringert aber den Unterschied zwischen den Marken und deren Umgebung. Dadurch werden viel mehr Marken erkannt als tatsächlich vorhanden sind, weil einige Bereiche auf der Ebene des Objektes eine ähnliche Helligkeit wie die Marken haben. Im umgekehrten Falle sinkt die Amplitude des Objektes ab, wenn das Objekt weiter von der Kamera entfernt wird. Dann werden aber nur weniger Marken als die gewünschte Anzahl aus dem Objekt erkannt.

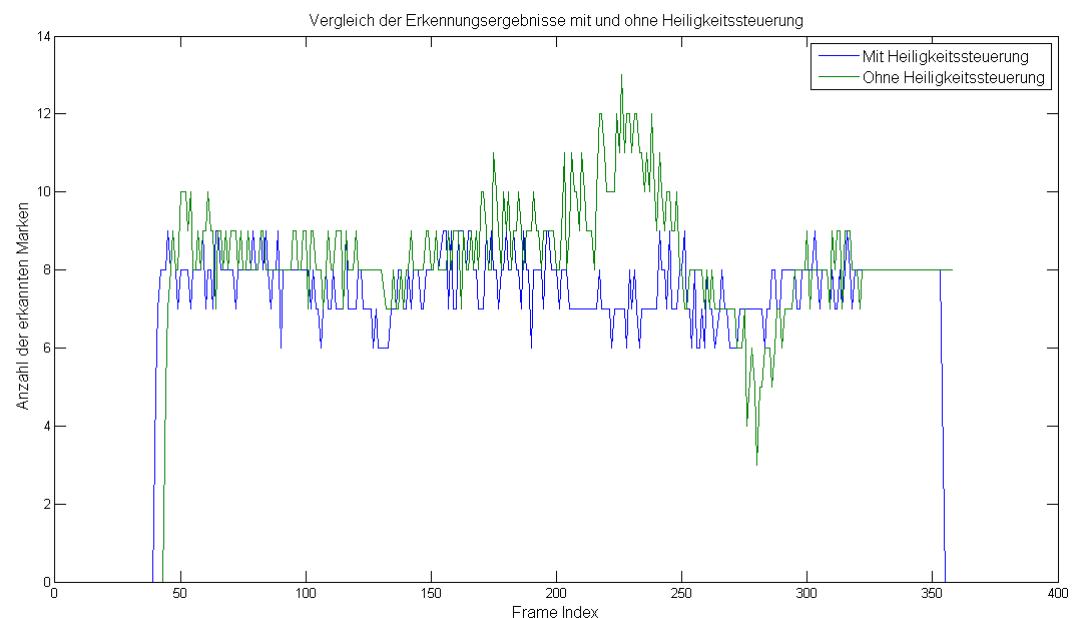


Abbildung 5.6: Der Vergleich der Erkennungsergebnisse mit und ohne Helligkeitssteuerung. Die Eingabebilder enthalten nur ein Objekt und die homogene Umgebung, was in Abbildung 5.2 teilweise gezeigt wird.

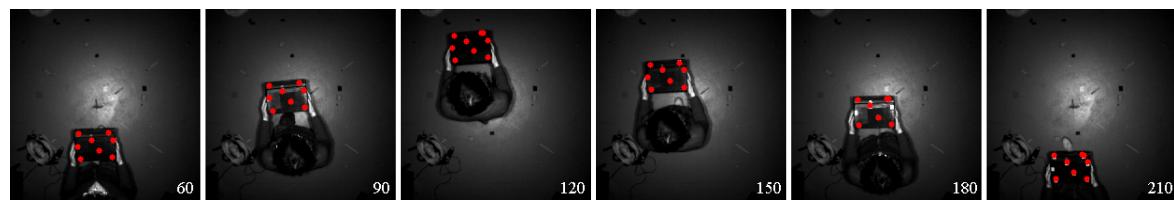


Abbildung 5.7: Der Testbildstrom mit dem Objekt, das nur in der Ebene bewegt wird.

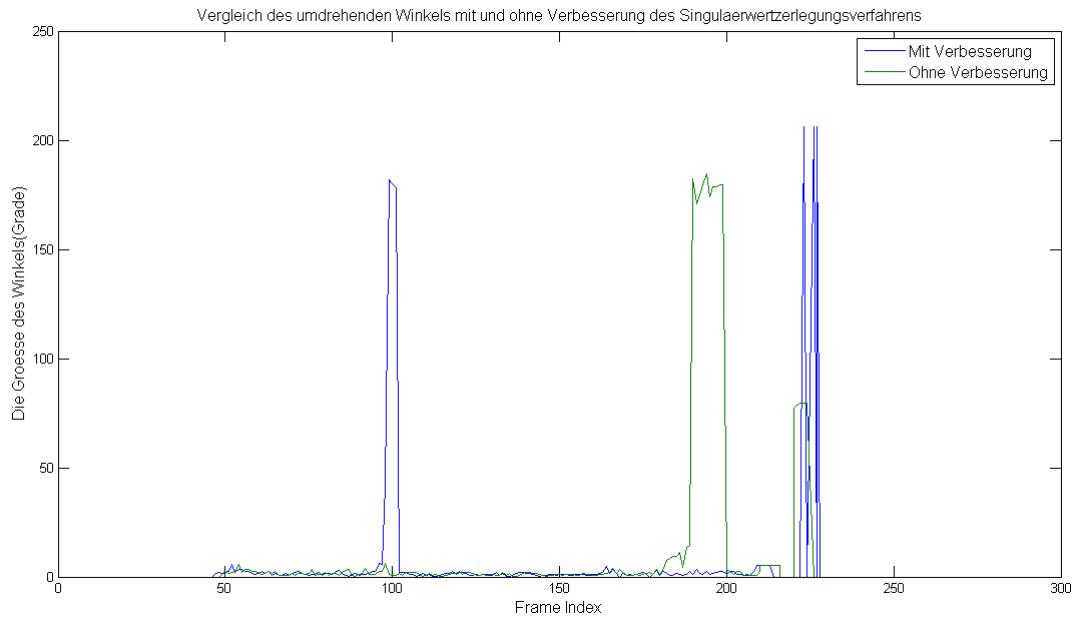


Abbildung 5.8: Der Vergleich der umdrehenden Winkel mit und ohne Verbesserung für ein im zweidimensionalen Raum bewegendes Objekt.

### 5.1.3 Verbesserung des Singulärwertzerlegungsverfahrens

Im Abschnitt 5.1.3 wird die Verbesserung des Singulärwertzerlegungsverfahrens erklärt. In diesem Teil der Evaluation werden die Wirkungen dieser Verbesserung bewertet. Die verschiedenen bewegenden bzw. umdrehenden Bewegungen des Objekts im zweidimensionalen bzw. dreidimensionalen Raum werden in drei Testbildströmen zusammengefasst, welche in Abbildungen 5.7, 5.9 und 5.11 teilweise gezeigt werden. Das Ziel des Singulärwertzerlegungsverfahrens ist, die Korrespondenzpunkte zu finden, damit die Orientierung des Objekts zwischen zwei Zeitpunkten berechnet werden kann. Deshalb soll die Transformation des Objekts möglich genau bestimmt werden, um das bessere Ergebnis zu erhalten. Aus diesem Grund wird der umdrehende Winkel des Objekts als Bewertungsparameter in diesem Teil der Evaluation verwendet. Die Diagramme über den Vergleich des umdrehenden Winkels mit und ohne Verbesserung des Singulärwertzerlegungsverfahrens werden in Abbildungen 5.8, 5.10 und 5.12 gezeigt.

In den Testbeispielen von zweidimensionaler Bewegung und Rotation wird der Vorteil der Verbesserung nicht deutlich gezeigt: In der Abbildung 5.8 gibt es entweder in der blauen Kurve oder in der grünen Kurve einige hohen Spitzen; in der Abbildung 5.10 laufen beide Kurven aber ruhig durch. Die Wirkung der Verbesserung scheint in Abbildung 5.12 deutlicher, welche eine Statistik über den umdrehenden Winkel des Objekts im dreidimensionalen Raum erstellt. Mithilfe der Screenshots von Abbildung 5.11 findet man, dass die umdrehenden Winkel zu groß berechnet werden, wenn eine neue

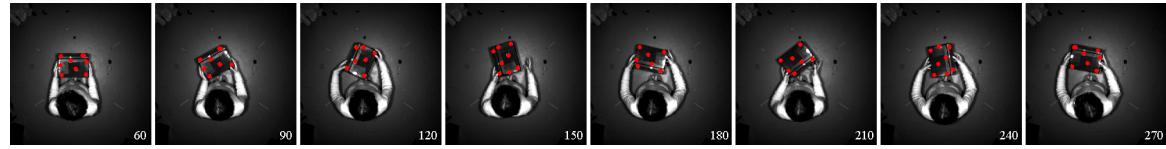


Abbildung 5.9: Der Testbildstrom mit dem Objekt, das nur in der Ebene umgedreht wird.

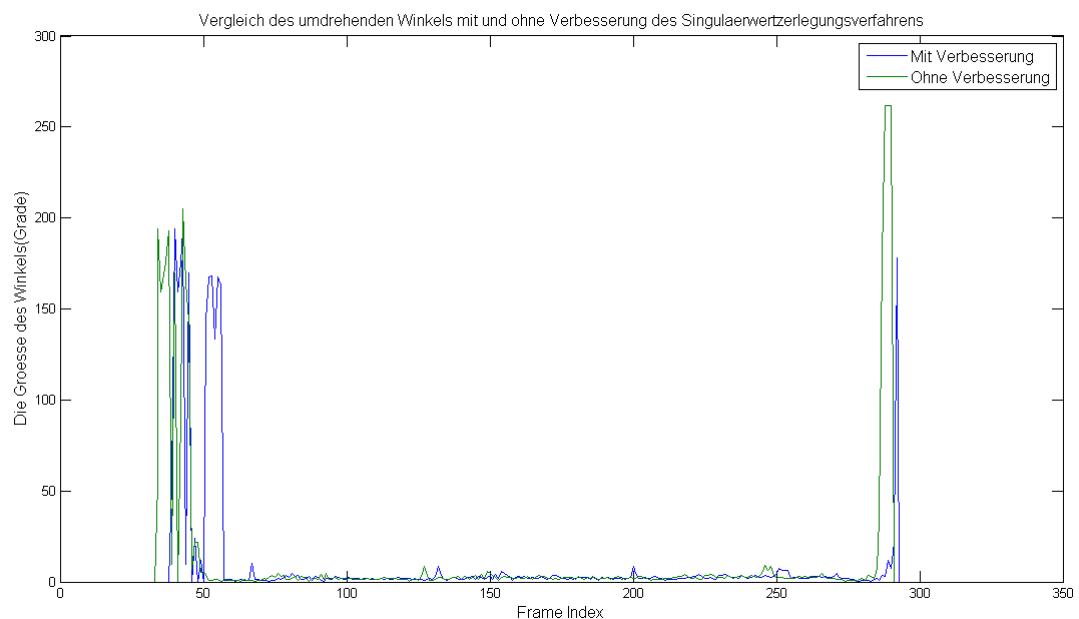


Abbildung 5.10: Der Vergleich der umdrehenden Winkel mit und ohne Verbesserung für ein im zweidimensionalen Raum umdrehendes Objekt.

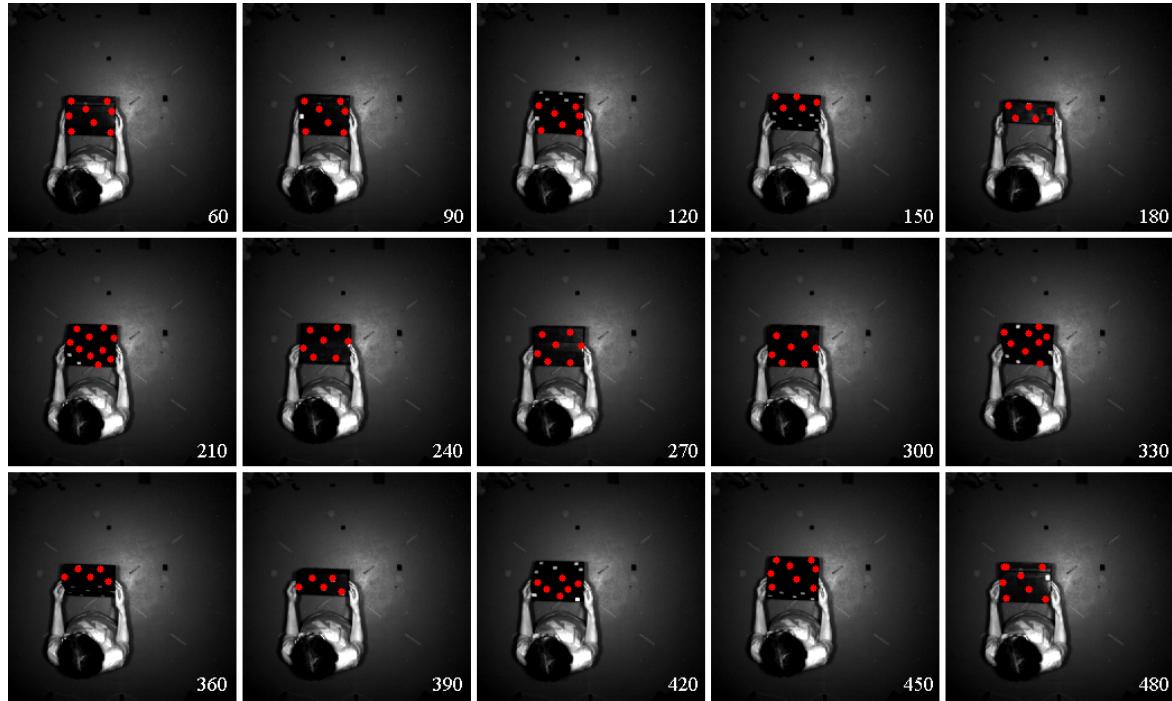


Abbildung 5.11: Der Testbildstrom mit dem Objekt, das im dreidimensionalen Raum umgedreht wird.

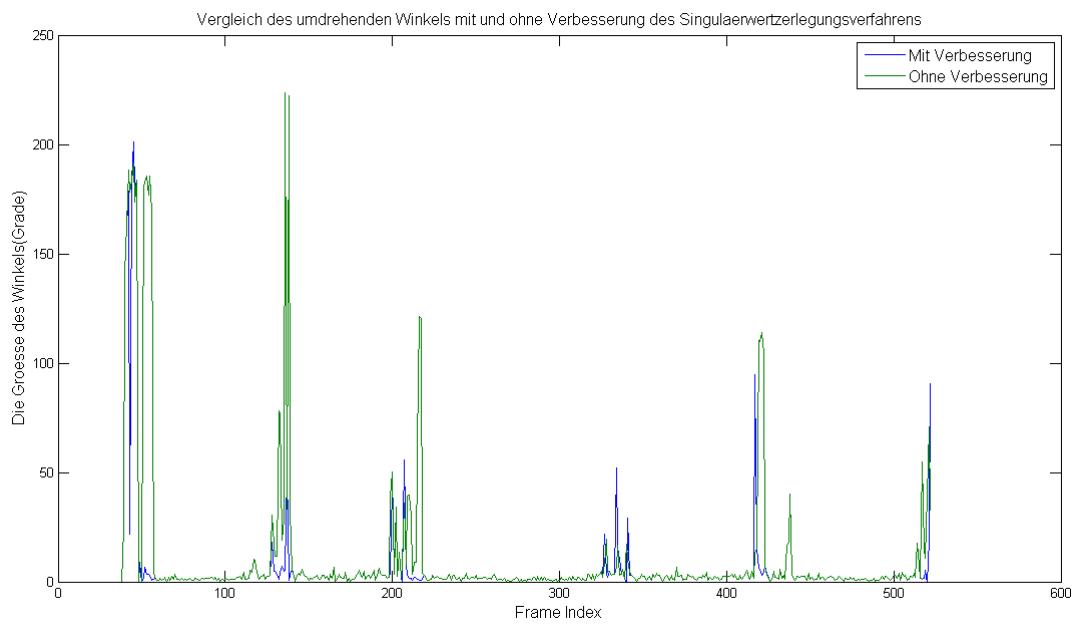


Abbildung 5.12: Der Vergleich der umdrehenden Winkel mit und ohne Verbesserung für ein im dreidimensionalen Raum umdrehendes Objekt.

Ebene des Objektes unter der Kamera vorkommt. Die Verbesserung des Singulärwertzerlegungsverfahrens lässt den Fehler des berechneten Winkels in den Bereichen von Größe und Zeitraum (in wie vielen Bildern der Fehler vorkommt) unterdrücken. Abbildung 5.13 zeigt die Endergebnisse mit und ohne Verbesserung über den Testbildstrom mit dreidimensionaler Umdrehung. Die grünen, kleinen Kugeln sind die neu erkannten Marken und die großen Kugeln beschreiben die stabilen Knoten. Um die Struktur des Objekts deutlicher zu zeigen, werden die stabilen Knoten in gleicher Farbe gesetzt, wenn sie in der gleichen Ebene erkannt worden sind. Die roten Knoten beschreiben die erste Ebene des Kästchens. Danach befindet sich die zweite, dritte bzw. vierte Ebene, die jeweils in Gelb, Cyan und Magenta gezeichnet werden. Die fünfte Farbe, Orange, beschreibt die gleiche Ebene wie Rot, die nach einer kompletten Rotation wieder von dem Programm erkannt wird. Die Verbesserung durch das verbesserte Singulärwertzerlegungsverfahren erkennt man durch die zwei Strukturgraphen aus Abbildung 5.13 deutlich. Die Knoten aus dem Schaubild der ersten Zeile können die vier Ebene eines Hexaeders sehr gut darstellen. Aber die Situationen in zweiter Zeile sind ungeordnet. Es gibt keine Möglichkeit, eine Gestalt aus den stabilen Knoten herauszufinden.

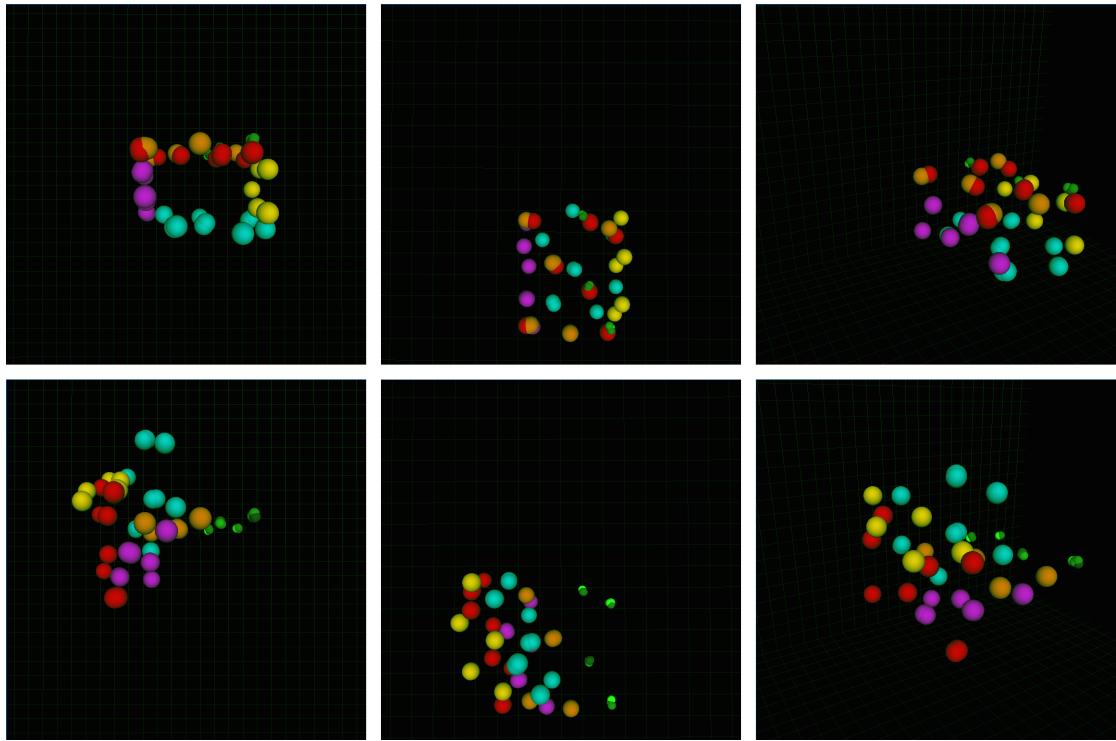


Abbildung 5.13: Die Strukturgraphen eines Kästchens nach Lernen mit (erste Zeile) und ohne (zweite Zeile) Verbesserung des Singulärwertzerlegungsverfahrens, die jeweils von vorne, oben, und einem Punkt an der Verlängerung der Diagonale des Objektes beobachtet werden. Die Knoten mit gleicher Farbe werden in gleicher Ebene erkannt.

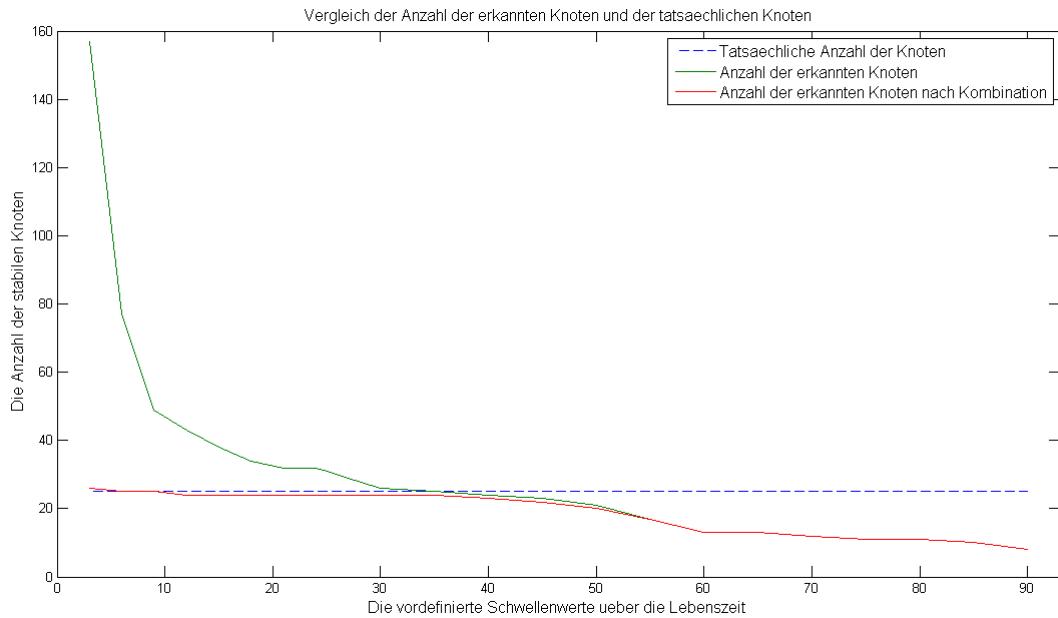


Abbildung 5.14: Der Vergleich der Anzahl der erkannten Knoten und der tatsächlichen Knoten, die am Testobjekt angebracht werden. Die verschiedenen Lebenszeiten werden als die vordefinierten Parameter des Programms getestet.

### 5.1.4 Aktualisierung des Strukturgraphen

Der Strukturgraph des Objektes wird durch den Algorithmus 9 in Kapitel 4 dargestellt. Die „Lebenszeit“ und der „maximale Abstand der identischen Knoten“ sind die zwei wichtigsten Parameter des Algorithmus, wodurch die stabilen Knoten aus dem Rauschen erkannt werden können. In diesem Abschnitt werden die Wirkungen mit verschiedenen Zuordnungen dieser zwei Parameter bzw. die Verbesserung der zusätzlichen Kombination der mehrfach erkannten Knoten diskutiert. Das erste Objekt mit 25 Marken auf vier Ebenen wird hier als das Testobjekt benutzt. Eine Statistik über die Anzahl der stabilen Knoten, die nach dem Ablauf des Programms in VTK Daten gespeichert werden sollen, wird zuerst erstellt und dann mit der tatsächlichen Anzahl der Knoten des Objekts verglichen, damit die unterschiedliche Auswahl der Parameter bewertet werden kann.

#### 5.1.4.1 Lebenszeit

Die blau gepunktete Linie in Abbildung 5.14 bleibt bei 25, was der tatsächlichen Anzahl der Marken auf dem Testobjekt entspricht. Die grüne Kurve zeigt die Anzahl der erkannten Knoten im Strukturgraphen nach dem Lernen mit verschiedenen Eingaben der Lebenszeit. Wenn die Lebenszeit zu klein definiert ist, werden die neu erkann-

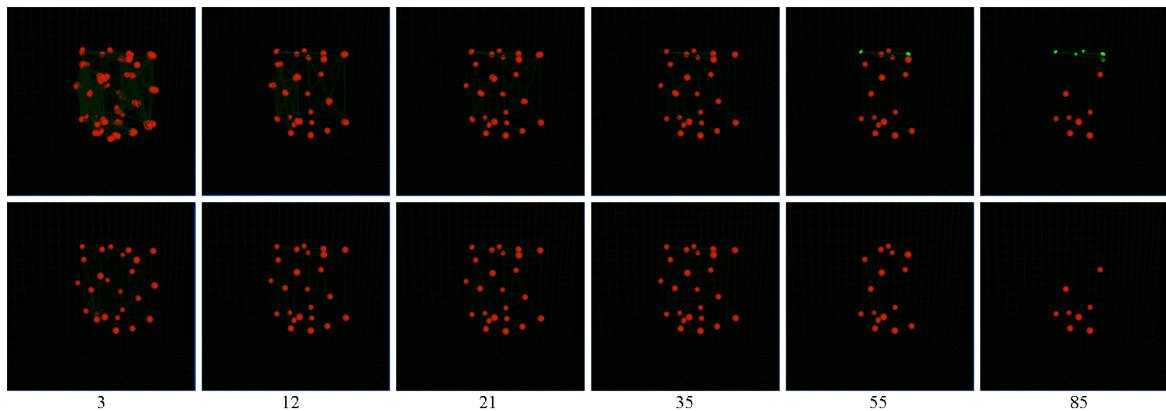


Abbildung 5.15: Die Screenshots für die Endergebnisse des Lernens mit verschiedenen Lebenszeit. Die Zahlen am Boden der Abbildung zeigen die Testwerte der Lebenszeit. Die erste Zeile zeigt die direkten Ergebnisse von dem Programm. Die Lösungen mit der zusätzlichen Kombination der mehrfach erkannten Knoten werden in der zweiten Zeile dargestellt.

ten Marken ziemlich schnell als die stabilen Knoten markiert und im Strukturgraphen eingefügt. Es folgen zwei negative Wirkungen. Zuerst werden gleiche Marken mehrmals als unterschiedliche Knoten erkannt. Zweitens werden die zufällig vorkommenden Rauschpunkte auch als stabile Knoten erkannt und im Endergebnis gespeichert. Die beste Anpassung kommt zwischen 30 bis 40 vor, wo fast gleich so viele Marken wie am realen Objekt erkannt werden. Wegen der Verstärkung der Erkennungsbedingung über die stabilen Knoten sinkt danach die Anzahl der stabilen Knoten langsam mit der Vergrößerung der Lebenszeit ab. Die erste Zeile des Schaubildes 5.15 zeigt die Strukturgraphen nach dem Lernen mit verschiedener Lebenszeit.

#### 5.1.4.2 Abstandsschwellenwert für die identischen Knoten

Der Abstandsschwellenwert für die identischen Knoten erklärt, wie nahe zwei Knoten beieinander liegen sollen, die als identische Knoten des Strukturgraphen erkannt werden. Die Veränderung für die Anzahl der stabilen Knoten mit Anstieg des Abstandsschwellwerts wird durch die grüne Kurve in Abbildung 5.16 gezeigt. Je kleiner der Abstandsschwellwert ist, desto schwieriger werden die neu gefundenen Knoten erkannt, die mit den vorhandenen Knoten identisch sind. Aber auf der anderen Seite werden die Strukturgraphen falsch dargestellt, wenn der Abstandsschwellwert zu groß definiert wird. In diesem Fall könnten zwei benachbarte Marken als ein Knoten erkannt werden, was sogar die Orientierung stören kann. Ein Beispiel dafür findet man im letzten Schaubild der Abbildung 5.17 mit dem Abstandsschwellenwert gleich 0.0045.

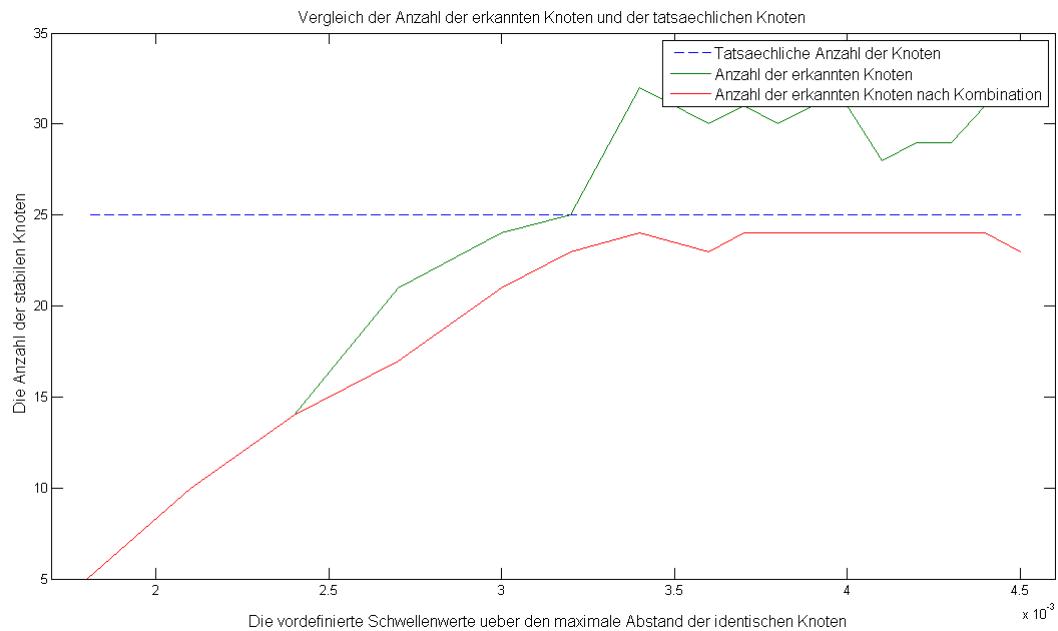


Abbildung 5.16: Der Vergleich der Anzahl der erkannten Knoten und der tatsächlichen Knoten, die am Testobjekt angebracht werden. Die verschiedenen, maximalen Abstände der identischen Knoten werden als die vordefinierten Parameter des Programms getestet.

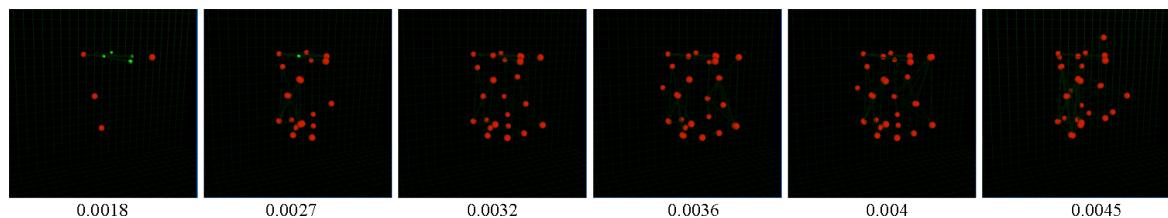


Abbildung 5.17: Die Screenshots für die Endergebnisse des Lernens mit verschiedenen Abstandsenschwellewerten für die identischen Knoten. Die Zahlen an dem Boden der Abbildung listen die getesteten Werte auf.

### 5.1.4.3 Kombination der mehrfach erkannten Knoten

Was deutlich in beiden Abbildungen 5.14 und 5.16 erscheint, ist, dass es nur ein Schnittpunkt zwischen der blauen und der grünen Kurven gibt. D.h., dass entweder die Lebenszeit oder der Abstandschwellwert schwierig zu bestimmen ist, damit genau so viele stabile Knoten wie die tatsächliche Anzahl der Marken am Objekt erkannt werden. Deshalb soll die Kombination der mehrfach erkannten Knoten nach dem Lernen durchgeführt werden. Dadurch kann die gültige Definitionsmenge der zwei Parameter vergrößert werden. Die roten Kurven in den Abbildungen 5.14 und 5.16 zeigen die Anzahl der stabilen Knoten nach der Kombination. In beiden Kurven gibt es ein relativ großes Intervall, in dem sich die Anzahl der erkannten stabilen Knoten der tatsächlichen Anzahl der Marken des Objekts sehr gut anpasst (24 gegen 25). Der Vergleich der Lernergebnisse mit und ohne die Kombination wird durch das Schaubild 5.15 erklärt. Die Bilder in zweiter Zeile zeigen die Strukturgraphen nach dem Lernen mit der Kombination über verschiedenen Eingaben der Lebenszeit. Wegen dieser Veränderung gibt es kaum einen Unterschied zwischen den Lösungen mit Lebenszeit von 12 bis 35.

### 5.1.5 Bildersteuerung

Wie in der Abbildung 5.12 dargestellt, werden die umdrehenden Winkel zu groß berechnet, wenn eine Ebene des Objekts allmählich verschwindet und die nachfolgende Ebene langsam vorkommt. Diese falschen Winkel können einen großen Fehler für die Darstellung des Strukturgraphen erzeugen, was in Abbildung 5.19 klar gezeigt wird. Alle Ebenen sind richtig erkannt, können aber leider keine sinnvolle Gestalt aufbauen. Deshalb ist die Bildersteuerung notwendig, damit die schlechten Bilder aus dem Bildstrom entfernt werden können. Abbildung 5.18 zeigt das statische Diagramm über den Vergleich der umdrehenden Winkel mit und ohne Bildersteuerung während des Testbildstroms aus Abbildung 5.11. Abgesehen von der großen Spitze der blauen Kurve am Anfang, die wegen der schnellen Bewegung zu dem Bildbereich der Kamera als Rauschen erzeugt wird, läuft die blaue Kurve ruhig in einem kleinen Intervall ohne große Spitze, die aber bei der grünen Kurve häufig beobachtet werden. D.h. mithilfe der Bildersteuerung kann das Programm die negative Wirkung der falsch berechneten Winkel vollständig vermeiden.

### 5.1.6 Teilgraph-Isomorphismus

Um die Suche der sogenannten isomorphen Knoten zu vereinfachen, wurden zwei Quoten im Abschnitt 4.4.2 definiert. In diesem Teil der Evaluation über Teilgraph-Isomorphismus wird das Wiedererkennungsteilprogramm mehrmals mit unterschiedlichen Abstandsquoten und Nachbarquoten durchgeführt, und die Anzahl der Bilder, in denen das Objekt erfolgreich erkannt werden kann, gespeichert. Die Quote dieser Anzahl und der Anzahl gesamter Eingabebilder, welche Erkennungsquote genannt wird,

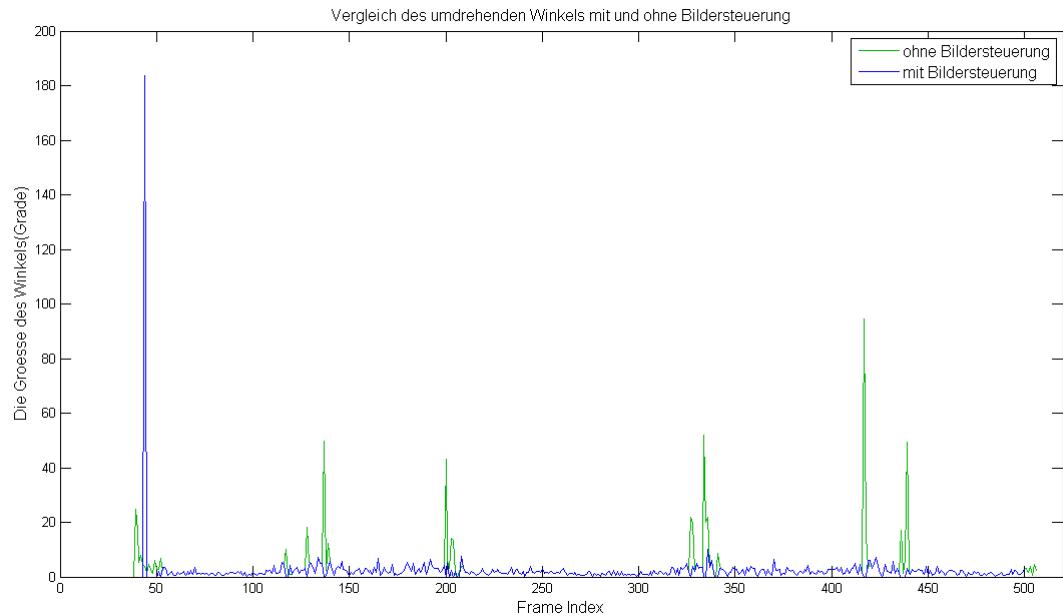


Abbildung 5.18: Der Vergleich der umdrehenden Winkel mit und ohne Bildersteuerung für ein im dreidimensionalen Raum umdrehendes Objekt. Der Bildstrom wird im Abbildung 5.11 teilweise gezeigt.

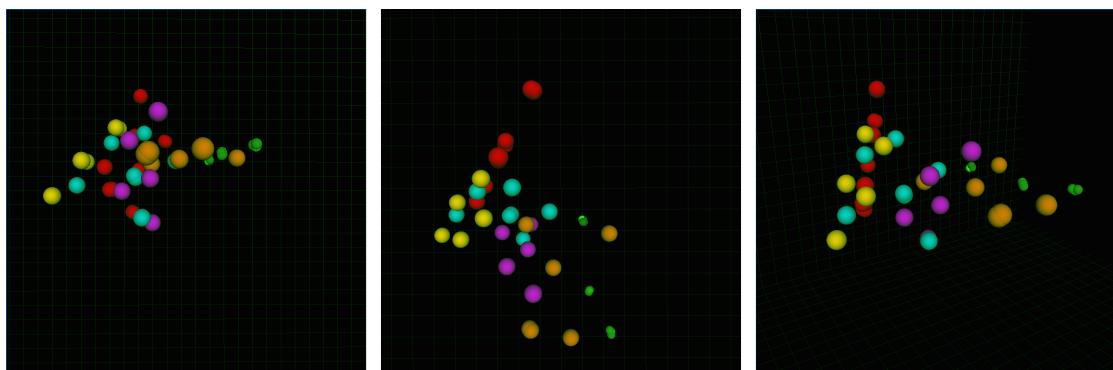


Abbildung 5.19: Der Strukturgraph eines Kästchen nach dem Lernen ohne Bildersteuerung, der jeweils von vorne, oben und einem Punkt an der Verlängerung der Diagonale des Objekts beobachtet wird. Die Knoten mit gleicher Farbe werden in gleicher Ebene erkannt.

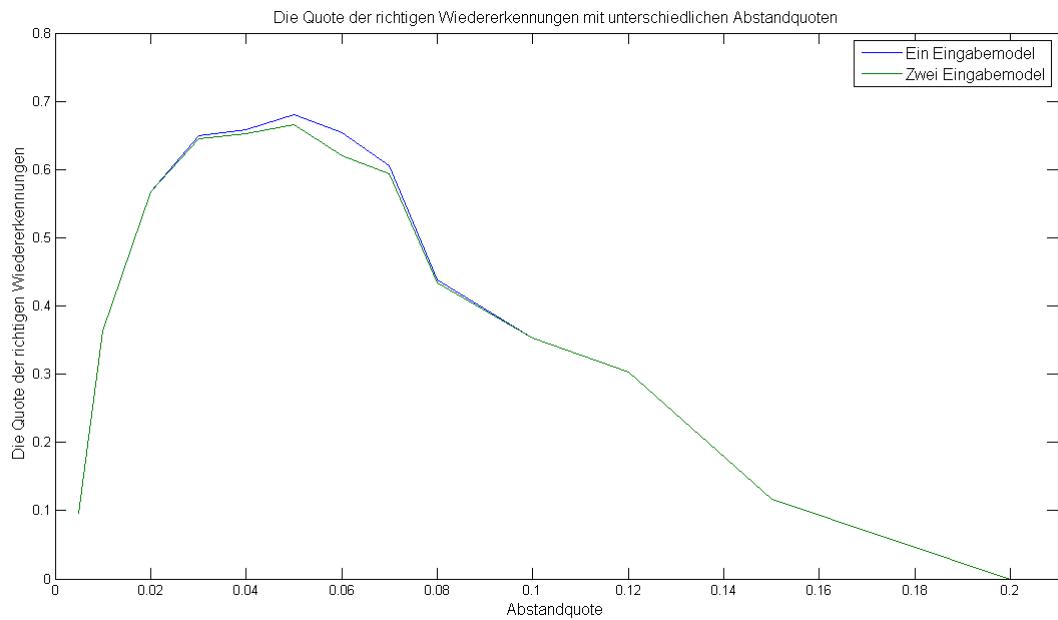


Abbildung 5.20: Die Quote der richtigen Wiedererkennungen mit unterschiedlichen Abstandschwellenwerten für ein im dreidimensionalen Raum umdrehendes Objekt. Der Bildstrom wird im Abbildung 5.11 teilweise gezeigt.

kann die Qualität der Wiedererkennung bewerten. Die blaue Kurven in den Abbildungen 5.20 und 5.21 zeigen genau diese Erkennungsquote mit verschiedenen Vorgaben der Abstandsquote bzw. der Nachbarquote. Der Testbildstrom ist den Testdaten identisch, was in den Abschnitten 5.1.3 und 5.1.5 verwendet, und in Abbildung 5.11 teilweise aufgeführt wird. Aus den statistischen Diagrammen ist zu entnehmen, dass die beste Wiedererkennungsquote dann vorkommt, wenn die Abstandsquote gleich 5% und die Nachbarquote gleich 80% ist.

Die grünen Kurven beschreiben die richtigen Wiedererkennungsquoten, wenn ein zusätzlicher Strukturgraph der erlernten Objekte von dem Programm als zweites Eingabemodell eingelesen wird. Der Unterschied zwischen den grünen und den blauen Kurven zeigt die Störung von dem zweiten Eingabemodell, welches in einem akzeptierten Intervall (weniger als 2%) liegt. Das erbringt auch einen überzeugenden Nachweis für die hohe Stabilität unseres Teilgraph-Isomorphismus-Algorithmus.

## 5.2 Globale Evaluation

In diesem Abschnitt werden die kompletten Abläufe von Lernen und Wiedererkennung bewertet. Für das Lernen werden die Strukturgraphen, die während des Lernprozesses

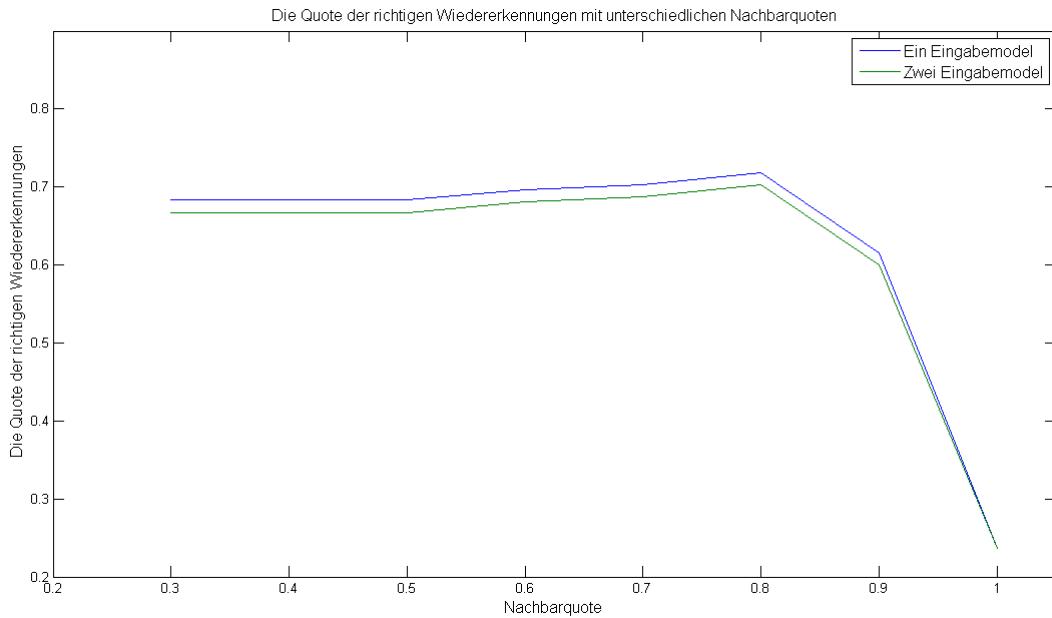


Abbildung 5.21: Die Quote der richtigen Wiedererkennungen mit unterschiedlichen Nachbarquoten für ein im dreidimensionalen Raum umdrehendes Objekt. Der Bildstrom wird im Abbildung 5.11 teilweise gezeigt.

durch die erkannten Marken erzeugt werden, mit den originalen Objekten verglichen. Für die Wiedererkennung wird die sogenannte richtige Erkennungsquote betrachtet, die die Qualität der Wiedererkennung sehr gut beschreiben kann. Natürlich ist der Zeitaufwand beider Teile von hoher Bedeutung, was jeweils in einigen Tabellen aufgelistet wird.

### 5.2.1 Objektlernen

Wegen der fehlenden Fähigkeit der Messungen ist die quantitative Evaluation des Lernens nicht möglich. Die Lernergebnisse können aber mithilfe der graphischen Visualisierung auch sehr gut beobachtet werden. Deshalb werden im folgenden Teil dieses Abschnitts einige Screenshots des Programms gezeigt, wodurch man die komplette Lernphase für verschiedene Eingabeobjekte bewerten kann.

Die Abbildung 5.22 zeigt einen normalen Lernprozess eines Objekts basierend auf dem Eingabebildstrom von 5.11. Die Strukturgraphen, die in der dritten Zeile gezeichnet werden, werden schrittweise durch die Eingabebilder dargestellt und orientieren sich an der aktuellen Position des Objektes. Um die verschiedenen Ebenen des Objektes deutlich zu unterscheiden, werden die Farben jeder Ebene manuell verändert. Das Pro-

gramm stoppt, wenn die erste Ebene wieder vorkommt, welche in der Abbildung 5.22 in Rot und Organe für die jeweils erste und zweite Erscheinung gefärbt werden. Was deutlich beobachtet werden kann, ist, dass alle Knoten, die zu einer gleichen Ebene gehören, in einer Fläche sehr gut erkannt werden. Weiterhin passen sich die relativen Ausrichtungen zwischen je zwei Ebenen der Wirklichkeit gut an. Der Nachweis dafür ist die teilweise Übereinstimmung von den roten Knoten und den orangenen Knoten, welche eigentlich von den gleichen Marken des Objekts erkannt werden. Die Abbildung 5.23 zeigt den Vergleich zwischen den Strukturgraphen und der realen Anordnung der Marken am Objekt.

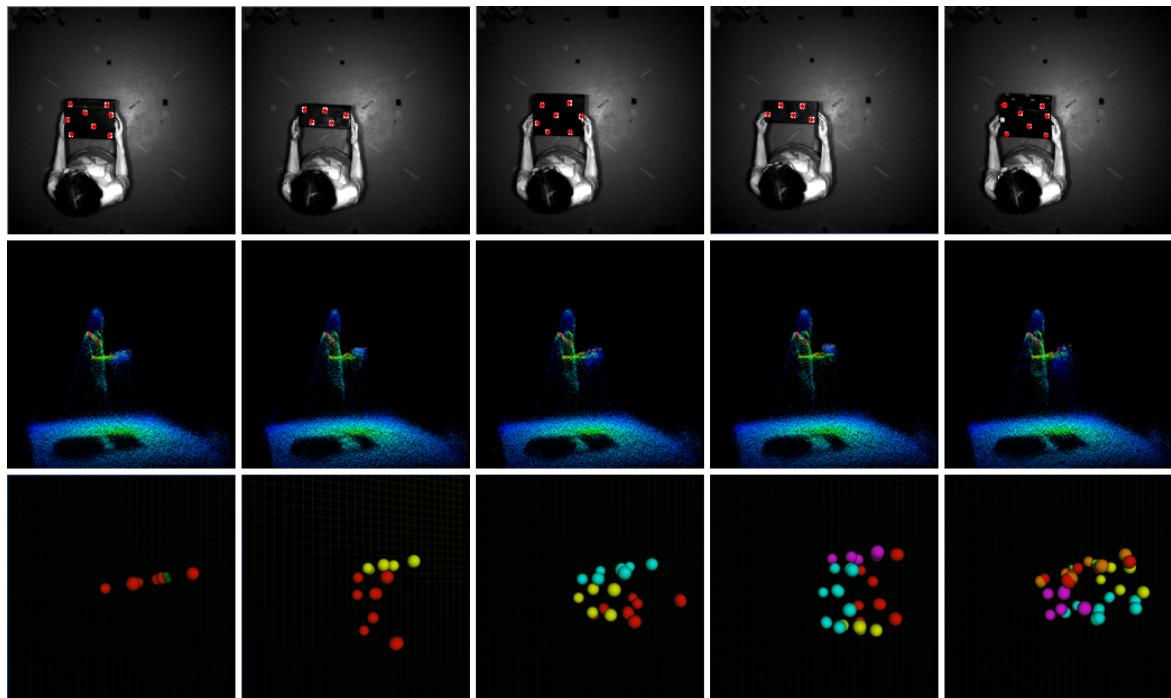


Abbildung 5.22: Die Screenshots des Programms. Die Schaubilder der ersten Zeile zeigen die Grauwertbilder der Amplitude mit roten Marken. Die entsprechenden 3D-Daten werden in der zweiten Zeile aufgeführt. In der dritten Zeile werden die mit bis aktuellen Bildern erzeugenden Strukturgraphen (ohne Kanten, ohne Kombination des mehrfach erkannten Knoten) gezeichnet.

In Abbildung 5.24 werden die Screenshots des Eingabebilderstroms für das zweite Objekt teilweise dargestellt. Da dieses Objekt schlanker ist als das erste Objekt, werden die Abstände zwischen den Marken verkleinert, was die Erkennung bzw. die Verfolgung der Marken schwieriger werden lässt. Aber mithilfe der Veränderung der Parameter der Algorithmen in Lernprozess kann trotzdem das erwünschte Lernergebnis erhalten werden. Die erste Zeile der Abbildung 5.25 zeigt den Strukturgraph des zweiten Objektes

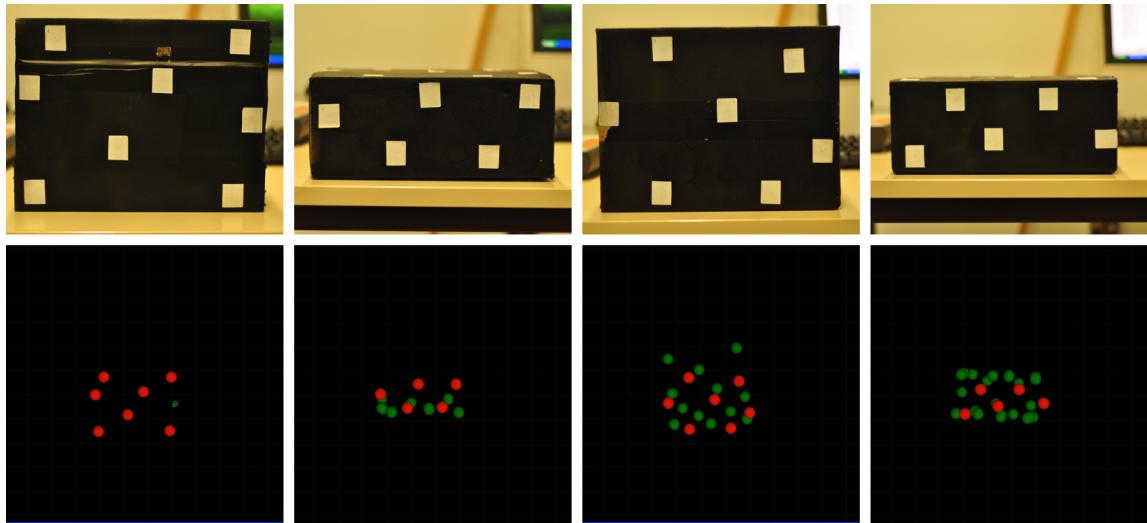


Abbildung 5.23: Der Vergleich zwischen den Strukturgraphen und der realen Anordnung der Marken des Objekts. Die roten Kugeln in den Schaubildern in der zweiten Zeile zeigen die stabilen Knoten des Strukturgraphen, welche den weißen Marken der Fotos in der ersten Zeile entsprechen.

mit gleichen Parametern wie beim Lernen des ersten Objektes. Die falsche Orientierung zwischen der gelben und blauen Ebene sieht man deutlich im ersten Bild. Nach Abstieg des Erwartungsintervalls der Anzahl der bekannten Marken in der Helligkeitssteuerung (siehe Algorithmus 7), das Verstärken der Beschränkung der größten Element in Korrespondenzuntersuchung (siehe Algorithmus 8) und das Verkleinern der minimalen Lebenszeit des stabilen Knoten in der Darstellung der Strukturgraphen (siehe Algorithmus 9) kann das Lernergebnis wie die zweite Zeile in der Abbildung 5.25 gefunden werden.

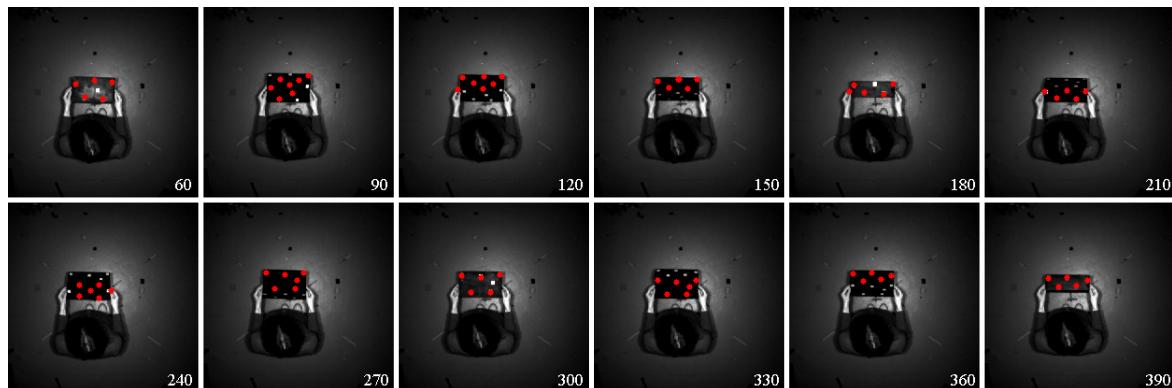


Abbildung 5.24: Die Screenshots des Testbildstroms für das zweite Objekt.

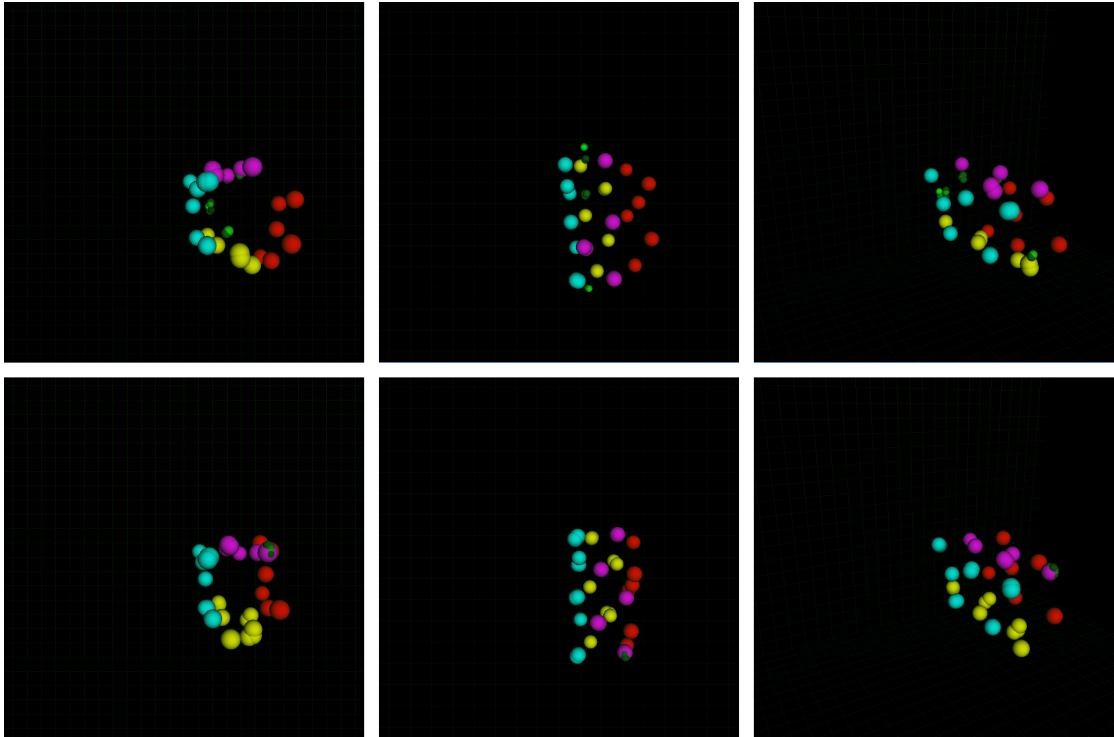


Abbildung 5.25: Die Strukturgraphen des zweiten Objekts, wobei die erste Zeile das Lernergebnis mit gleichen Parametern wie beim Lernen des ersten Objektes zeigt, und die zweite Zeile das Lernergebnis mit unterschiedlichen Parametern aufzeichnet.

### 5.2.1.1 Zeitaufwand

Die Tabelle 5.1 führt die Laufzeit aller Teilprogramme in der Lernphase auf. Die Teilprogramme von CenSurE Detektor und Helligkeitssteuerung, Markenerkennung und Visualisierung kosten viel mehr Zeit als die anderen Teilprogramme, was 77.55% des gesamten Zeitaufwands beträgt (siehe Abb. 5.26). Die größte Anforderung der Markenerkennung ist ungeplant. Der Grund liegt darin, dass die Marken des Objektes nicht direkt durch den CenSurE Detektor erkannt werden können, sondern eine zusätzliche Kombination der Merkmale benötigt. Diese Merkmale sind direkt von dem Detektor erkennbar und liegen in der Nähe von einem anderen. Der reine durchschnittliche Berechnungszeitaufwand ohne Visualisierung beträgt 53.7065  $ms$  und die entsprechende Framerate liegt bei 18.6 fps, was die Echtzeitbedingung leider nicht sehr gut erfüllt.

### 5.2.2 Objektwiedererkennung

In diesem Abschnitt wird die Qualität der Wiedererkennungen mit verschiedenen initialisierten Bedingungen betrachtet. Wie im 4.4.2 erklärt, werden die Strukturgraphen

Abstand-Filter	CenSurE Detektor und Helligkeitssteuerung	Markenerkennung		gesamter Zeitaufwand
3.5325	13.2369		24.2683	
Segmentierung	Korrespondenzuntersuchung	Orientierung	Bildersteuerung	Visualisierung
0.7505	0.1258	1.0755	2.3291	18.4927

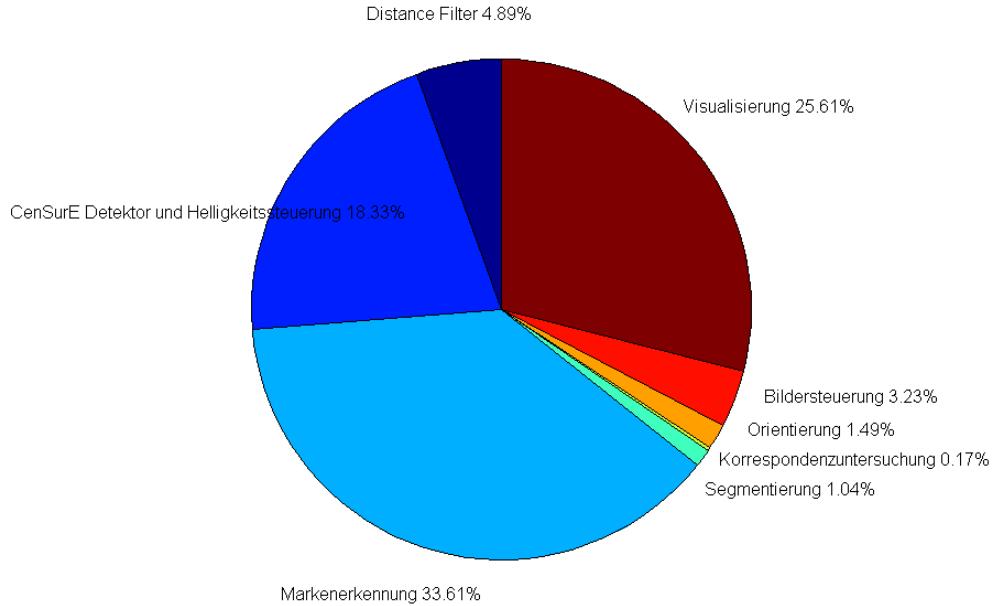
Tabelle 5.1: Die Zeitaufwände (in *ms*) aller Teilprogramme der Lernensphase.

Abbildung 5.26: Die Anteile der Laufzeit aller Teilprogramme der Lernphase.

der erlernten Objekte, die auch als Eingabemodelle bezeichnet werden, am Anfang der Wiedererkennungsphase in dem Programm eingegeben. Die Anzahl der Eingabemodelle liefert einen starken Einfluss auf die Erkennungsergebnisse bzw. den Zeitaufwand. Je mehr die Eingabemodelle berücksichtigt werden, desto schlechter die Erkennungsergebnisse sind und mehr Zeit dafür aufgewendet werden muss. Außerdem ist die Anzahl der Objekte im aktuellen Eingabebildstrom auch eine wichtige Variable für Bewertung unseres Programmes. Deshalb werden drei verschiedene Kombinationen von der Anzahl der Eingabemodelle und der Anzahl der Eingabeobjekte in folgenden Teilabschnitten diskutiert.

Die Testbildströme, die im Abschnitt 5.1.3 verwendet wurden, werden hier für den Test mit nur einem Eingabeobjekt wieder benutzt. Diese drei Testbildströme beschreiben jeweils ein Objekt, das sich in der Ebene bewegt (siehe Abb. 5.7) und im zweidimensionalen bzw. dreidimensionalen Raum umdreht (siehe Abb. 5.9 und Abb. 5.11). Analog zu der Teilevaluation über den Teilgraph-Isomorphismus wird die Quote der richtigen Wiedererkennungen nach dem kompletten Durchlauf des Testbildstroms berechnet, was als die wichtigste Variable der Bewertung betrachtet wird. Außerdem wird der durchschnittliche Zeitaufwand bzw. die Anzahl der verglichenen Knoten im Teilgraph-Isomorphismus aufgezeichnet.

### 5.2.2.1 1 Eingabeobjekt mit 1 Eingabemodell

Die Testergebnisse findet man in der Tabelle 5.2. Da nur ein Eingabemodell mit den aktuellen Eingabebildern verglichen werden soll, sind die gesamte und die richtige Erkennungsquote identisch. Der durchschnittliche Zeitaufwand liegt unter 26 ms, was die Echtzeitbedingung sehr gut erfüllt.

Testbildströme	gesamte Erkennungsquote	richtige Erkennungsquote	Anzahl der isomorphen Knoten	Zeitaufwand jedes Bilds
Statisch	91.46%	91.46%	4.1068	17.0391 ms
2D Translation	55.81%	55.81%	5.7813	25.7035 ms
2D Rotation	66.67%	66.67%	6.6237	22.9068 ms
3D Rotation	59.94%	59.94%	2.6265	18.8584 ms

Tabelle 5.2: Die durchschnittlichen statistischen Daten der Wiedererkennung für unterschiedliche Testsamples, wobei nur ein Eingabeobjekt und ein Eingabemodell berücksichtigt werden.

Die andere Situation für ein Eingabeobjekt und ein Eingabemodell ist, dass das Eingabeobjekt und Eingabemodell sich auf verschiedenen Objekten beziehen. In Idealfall soll das Eingabeobjekt während der Bildsequenz gar nicht erkannt werden. Die Tabelle 5.3 listet die falschen Erkennungsquoten für verschiedenen Testbildströme auf. Im schlechtesten Fall gibt es aber nur weniger als 1% Bilder in einer Bildsequenz, in den das Eingabeobjekt als das unabhängige Eingabemodell falsch erkannt wird.

Testbildströme	Statisch	2D Translation	2D Rotation	3D Rotation
Falsche Erkennungsquote	0.38%	0%	0%	0.86%

Tabelle 5.3: Die durchschnittlichen falschen Erkennungsquoten für unterschiedliche Testsamples, wobei nur ein Eingabeobjekt und ein Eingabemodell berücksichtigt werden, die sich aber auf verschiedenen Objekten beziehen.

### 5.2.2.2 1 Eingabeobjekt mit 2 Eingabemodellen

Der deutliche Unterschied zu den Testergebnissen im letzten Abschnitt besteht darin, dass die richtige Erkennungsquote nicht immer gleich wie die gesamte Erkennungsquote ist. Der Grund liegt darin, dass das Eingabeobjekt in einigen Bildern als falsches Modell erkannt wird. Aber wegen der zufriedenstellenden Stabilität des Teilgraph-Isomorphismus-Algorithmus ist die Quote der falschen Wiedererkennungen klein (weniger als 2%). Außerdem steigt die durchschnittliche Laufzeit jedes Bildes mit der Zunahme der Eingabemodelle deutlich an, weil jetzt für jedes Eingabeobjekt der Teilgraph-Isomorphismus zweimal durchgeführt werden muss. Alle Testdaten werden in der Tabelle 5.4 aufgeführt.

Testbildströme	gesamte Erkennungsquote	richtige Erkennungsquote	Anzahl der isomorphen Knoten	Zeitaufwand jedes Bilds
Statisch	91.46%	91.46%	4.1174	29.4591 ms
2D Translation	56.98%	55.81%	5.7245	43.3256 ms
2D Rotation	66.67%	66.67%	6.6237	38.6487 ms
3D Rotation	60.24%	59.94%	2.7078	33.0331 ms

Tabelle 5.4: Die durchschnittlichen, statistischen Daten der Wiedererkennung für unterschiedliche Testsamples, wobei nur ein Eingabeobjekt aber zwei Eingabemodelle berücksichtigt werden.

### 5.2.2.3 Verbesserung mit Kandidaten

Mithilfe der Kandidaten kann ein Eingabeobjekt in dem Bildstrom verfolgt werden. Die Erkennungsergebnisse werden in den Kandidaten gespeichert. Dadurch liefert das Programm ständig die Ausgaben, obwohl von dem aktuellen Bild kein entsprechendes Objekt gefunden werden kann. Tabelle 5.5 listet die Erkennungsquoten für unterschiedlichen Testbildströme mit verschiedener Anzahl der Eingabemodelle auf, in der die Verbesserung der Nutzung der Kandidaten deutlich gezeigt wird. Diese Veränderungen können auch in dem Balkendiagramm in Abbildung 5.27 beobachtet werden.

Neben der Vergrößerung der Erkennungsquote steigt die durchschnittliche Laufzeit jedes Bildes deutlich an. Die Zunahme des Zeitaufwandes, die in der rechten Spalte der Tabelle 5.5 aufgelistet wird, schwankt um 10 ms. D.h., dass diese Zunahme nicht von

Testbildströme	Anzahl der Eingabemodelle	ohne die Verbesserung der Kandidaten		mit der Verbesserung der Kandidaten		Differenz des Zeitaufwands (ms)
		Erkennungsquote	Zeitaufwand (ms)	Erkennungsquote	Zeitaufwand (ms)	
Statisch	1	91.46%	17.0391	97.86%	25.8078	8.7687
	2	91.46%	29.4591	97.86%	40.5302	11.0712
2D Translation	1	55.81%	25.7035	100%	33.5872	7.8837
	2	55.81%	43.3256	97.67%	53.7907	10.4651
2D Rotation	1	66.67%	22.9068	89.96%	31.0645	8.1577
	2	66.67%	38.6487	89.96%	48.2867	9.6380
3D Rotation	1	59.94%	18.8584	99.40%	27.8223	8.9639
	2	59.94%	33.0331	99.10%	45.4458	12.4127

Tabelle 5.5: Die Erkennungsquote und der Zeitwand mit und ohne die Verbesserung der Kandidaten für unterschiedliche Testbildströme mit verschiedener Anzahl der Eingabemodelle.

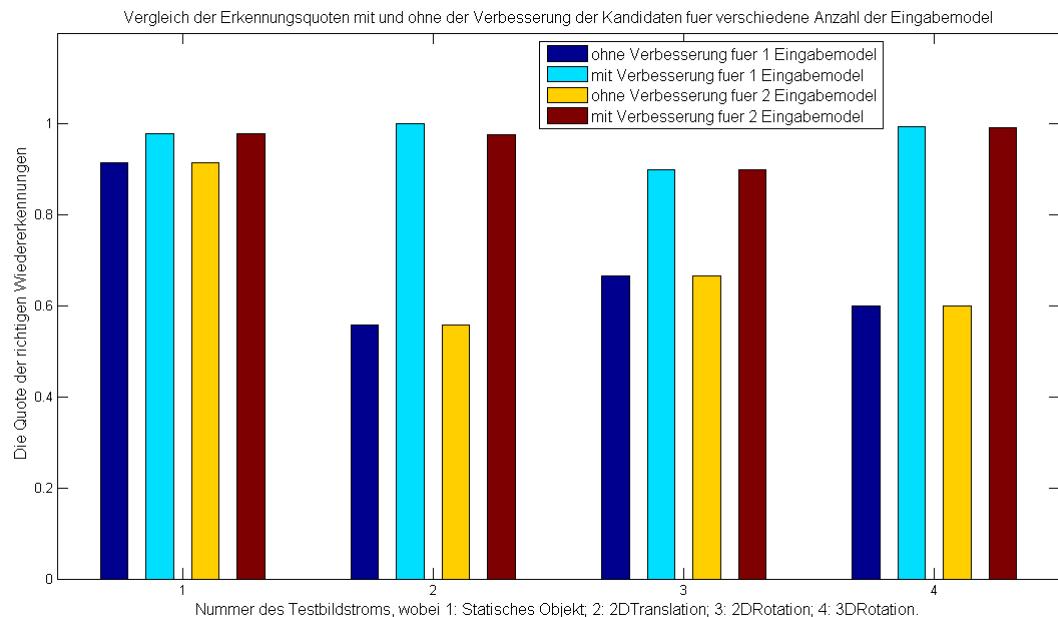


Abbildung 5.27: Der Vergleich der Erkennungsquoten mit und ohne Verbesserung der Verwendung der Kandidaten. Alle drei Testbildströme werden mit jeweils 1 bzw. 2 Eingabemodellen betrachtet.

der durchschnittlichen Laufzeit der Erkennung stark beeinflusst wird, was die Konvergenz des Zeitaufwands unserer Verbesserung mit Kandidaten experimentell nachweist. Das entsprechende Balkendiagramm findet man in Abbildung 5.28.

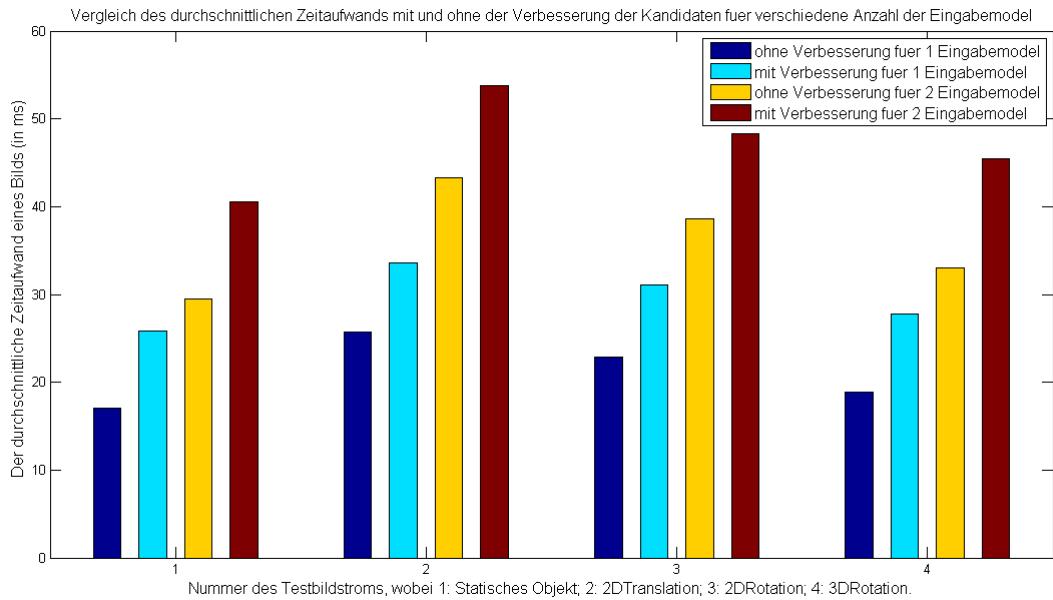


Abbildung 5.28: Der Vergleich des Zeitaufwandes mit und ohne Verbesserung der Verwendung der Kandidaten. Alle drei Testbildströme werden mit jeweils 1 bzw. 2 Eingabemodellen betrachtet.

#### 5.2.2.4 2 Eingabeobjekte mit 2 Eingabemodellen

Die Wirkung der Erkennungskandidaten vergrößert sich, wenn zwei Eingabemodelle von dem Programm eingelesen werden und gleichzeitig zwei Objekte im Testbildstrom vorkommen. Wegen dem überlappenden Vergleich zwischen den Eingabemodellen und Eingabeobjekten wird die falsche Erkennungsquote deutlich erhöht. Außerdem stört die unterschiedliche Qualität der Markenerkennung verschiedener Objekte auch die Wiedererkennung. In unserem Test ist es beispielsweise häufig schwierig, das zweite Objekt ständig zu verfolgen, weil nicht genug Marken für das Objekt aus dem aktuellen Bild erkannt werden können. Dadurch nimmt die Genauigkeit der Korrespondenzuntersuchung und Orientierung weiterhin ab. Deshalb sind die historischen Erkennungsergebnisse für die Vorhersage bzw. Korrektur des aktuellen Erkennungsergebnisses besonders wichtig, welches in den Kandidaten regelmäßig gespeichert wird. Die Tabelle 5.6 gibt den Vergleich der Erkennungsergebnisse mit und ohne die Verbesserung der Kandidaten

wieder. Was unbedingt beobachtet werden soll, ist der 50% Anstieg der Erkennungsquote für das zweite Objekt. Einige Screenshots werden in Abbildung 5.29 gezeigt.

	Erkennungsquote erstes Objekts	Erkennungsquote zweites Objekts	Zeitaufwand (ms)
Ohne die Verbesserung der Kandidaten	53.09%	29.06%	49.1496
Mit der Verbesserung der Kandidaten	94.53%	79.14%	59.9338

Tabelle 5.6: Die statistischen Daten der Wiedererkennung, welche zwei Eingabeobjekte und zwei Eingabemodelle hat.

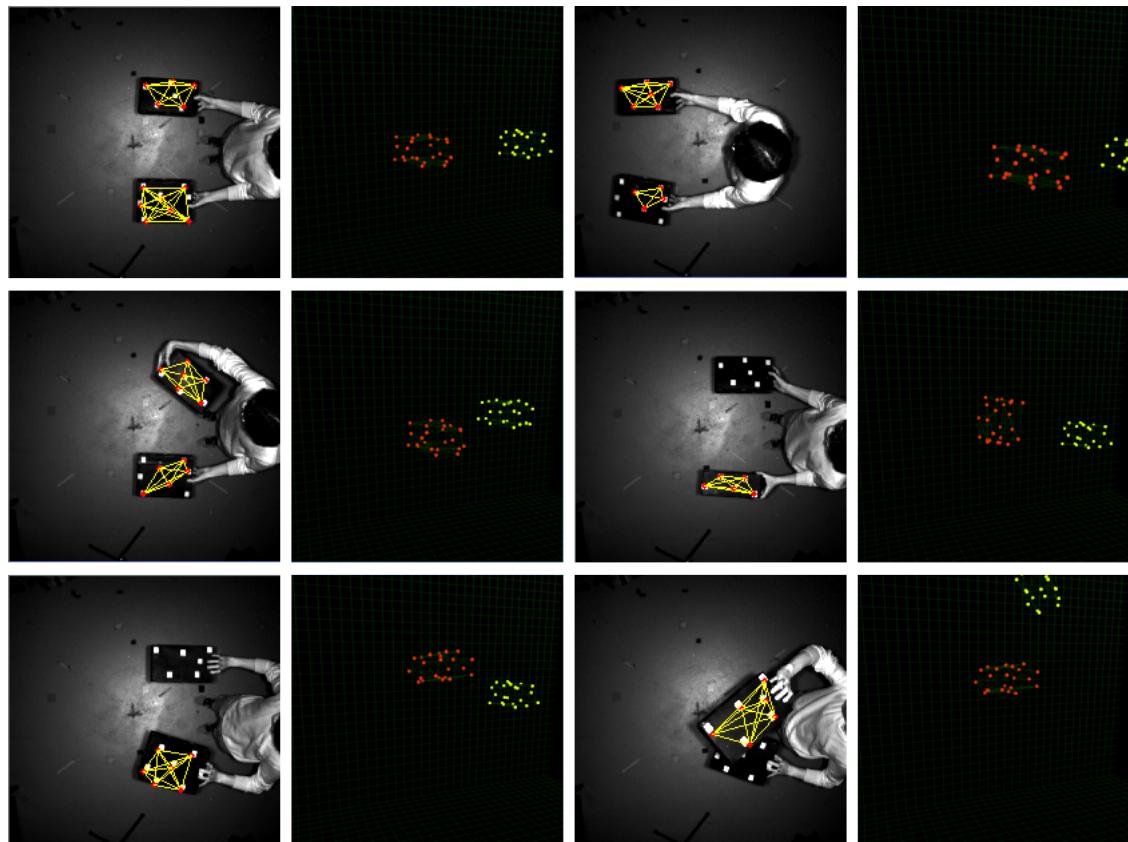


Abbildung 5.29: Die Visualisierung der Erkennungs- und Verfolgungsergebnisse.



# Kapitel 6

## Zusammenfassung und Ausblick

Diese Arbeit implementiert die auf Marken basierte Objekterkennung bzw. Verfolgung. Die Testobjekte werden in schwarz gefärbt und mit weißen Marken auf der Oberfläche markiert. Eine PMD-Kamera beobachtet die ganze Szene von oben und liefert direkt die 3D-Daten. Das gesamte Programm kann in zwei Teilen, Lernen und Wiedererkennung, zusammengefasst werden. Im Lernprozess wird ein Fremdobjekt unter der Kamera gezeigt. Die Marken des Objekts werden von dem Programm erkannt und in einem Strukturgraph eingefügt, was die räumliche Struktur des Objekts beschreiben kann. Der Strukturgraph wird nach dem Lernen als eine VTK-Datei gespeichert. Wenn es mehr Objekte im Eingabebildstrom gibt, wird die Markenmenge für jedes Objekt zuerst segmentiert. Am Anfang der Wiedererkennungsphase werden die vorhandenen Strukturgraphen eingelesen, was als die Eingabemodelle der Wiedererkennung definiert wird. Für jedes Eingabebild werden alle Marken zuerst erkannt, genau wie es beim Lernen durchgeführt wird. Die erkannten Marken werden dann in unterschiedlichen Kandidaten des Objekts aufgeteilt. Ein Eingabemodell ist genau dann wiedererkannt, wenn mindestens ein Kandidat existiert, der den gleichen Teilgraph zu diesem Eingabemodell erhält. Die aktuelle Orientierung und Lage des Objektes kann danach durch die Korrespondenzpunktpaare bestimmt werden.

Diese Arbeit verfolgt die Idee und einige Algorithmen von [Rhijn & Mulder 2005]. Aber wegen der verschiedenen Arbeitsumgebung ist die Implementierung in dieser Arbeit im viele Bereichen unterschiedlich: Nur eine 3D-Kamera beobachtet die ganz Szene und die Objekte; Die Auflösung der PMD Kamera ist nur  $204 \times 204$ , welche viel kleiner als das Stereokamerasytem in [Rhijn & Mulder 2005] ist; Der Abstand zwischen der Kamera und den Objekt ist viel größer. Dazu sollen die Markenerkennung bzw. die Korrespondenzuntersuchung verbessert werden, und die verrauschte Bilder aus der Bildsequenz löschen lassen. Alle diese Verbesserungen wurden in dieser Arbeit sehr gut implementiert. Außerdem werden den 3D Strukturgraphen nach dem Lernen erzeugt, welche nicht nur die Charakteristik sondern auch die geometrischen Informationen der Objekte enthalten. Mithilfe dieser Strukturgraphen können die Modelle für die Objekte

in einem Mensch-Roboter-Kooperation System, wie z.B. MAROCO, einfacher dargestellt werden. Die dazu entsprechende VTK-Datei können auch durch andere Software weiter verarbeitet werden.

Natürlich könnte das Programm weiter entwickelt werden. Die weitere Verbesserungen können in zwei Bereichen, Stabilität und Zeitaufwand, zusammengefasst werden.

### Stabilität

Die Größe und Position der Marken muss für aktuelles Programm exakt entworfen werden, damit die richtigen Erkennungsergebnisse erhalten werden können. Sonst ist das Objekt schwierig zu erkennen. Aus diesem Grund sind die Länge bzw. die Breite des Objektes stark beschränkt. Die Anordnung der Marken beeinflusst auch die Quote der erfolgreichen Erkennungen in der Wiedererkennungsphase.

### Zeitaufwand

Wie in den Tabellen 5.1 bis 5.6 gezeigt, erfüllt der Zeitaufwand aber nur teilweise die Echtzeitbedingung. In der Lernphase kann das Programm aber nur die Framerate bis zu 18.6 fps erreichen. In der Wiedererkennungsphase hängt die Framerate von der Anzahl der betrachteten Eingabeobjekte und Eingabemodelle ab. Wenn nur einen Eingabemodell betrachtet wird, kann das Programm im Durchschnitt 30 Bilder pro Sekunde bearbeiten. Für zwei Eingabemodelle wird diese Zahl aber leider sofort auf 20 sinken.

### Mögliche Verbesserungsverfahren

Entweder die Stabilität oder der Zeitaufwand hängt stark von dem Detektor ab. Deshalb ist Verwendung eines neuen, besseren Detektors eine Möglichkeit, um das Programm im vorherigen Bereichen zu verbessern. Dieser Detektor soll bessere Genauigkeit und Stabilität haben. Im Idealfall sollen alle Marken eindeutig erkannt werden können, und die Ergebnisse unabhängig von der Helligkeit der Eingabebilder sein. Dadurch kann die Anzahl der Schleifen in der Helligkeitssteuerung stark reduziert und die Kombination der Merkmale in der Markenerkennung sogar komplett entfernt werden. Diese Vereinfachungen des Programmes können mehr als die Hälfte der gesamten Laufzeit einsparen, was in dem Kreisdiagramm in Abbildung 5.26 deutlich gezeigt wird. Die bessere Stabilität des Detektors fordert an, dass die Erkennungsergebnisse nicht empfindlich für die Abstände zwischen den Marken sein sollen. Damit können die Marken nicht genau mit den Regeln vom Abschnitt 4.2.2 angebracht werden.

Außerdem kann der Berechnungsteil des Programmes mit CUDA neu formuliert werden, um damit die Laufzeit zu reduzieren. Der GPU hat eine stärkere Berechnungsfähigkeit über die Gleitkommazahl im Vergleich zur CPU, und liefert gleichzeitig eine bessere Möglichkeit für die parallelen Arbeiten.

# Literaturverzeichnis

- [Agrawal, Konolige & Blas 2008] M. Agrawal, 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 9)
- [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 11)
- [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 34)

- [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 33)
- [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, 27, 29, 32 und 49)
- [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 Seite 14)

- [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 Seite 10)
- [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 53)
- [PMD CamCube Einführung] PMDs CamCube Einführung: <http://www.pmdtec.com/fileadmin/pmdtec/downloads/documentation/datenblatt-camcube3.pdf> Zuletzt besucht: 13.03.2012 (Zitiert auf Seiten 13 und 15)
- [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, 36, 55 und 87)
- [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, 46 und 47)
- [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 6)
- [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 43)
- [StarDetector OpenCV Adaventure] <http://experienceopencv.blogspot.de/2011/01/star-feature-detector.html> Zuletzt besucht: 24.08.2012 (Zitiert auf Seite 43)
- [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 53)
- [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)