Raum-zeitliche Objekt- und Aktionserkennung:

Ein statistischer Ansatz für reale Umgebungen

Dissertation zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der Technischen Fakultät der Universität Bielefeld
vorgelegt von

Markus Hahn

Juli 2010

Abdruck der genehmigten Dissertation zur Erlangung des akademischen Grades *Doktor der Ingenieurwissenschaften (Dr.-Ing.)* an der Technischen Fakultät der Universität Bielefeld.

Dipl.-Inf. Markus Hahn
E-Mail: markus_hahn@gmx.de

Gutachter:
Prof. Dr. Franz Kummert, Universität Bielefeld
Prof. Dr. Christian Wöhler, Universität Dortmund
Prof. Dr. Xiaoyi Jiang, Universität Münster

Prüfungsausschuß:
Prof. Dr. Ralf Möller , Universität Bielefeld
Prof. Dr. Franz Kummert, Universität Bielefeld
Prof. Dr. Christian Wöhler, Universität Dortmund
Prof. Dr. Xiaoyi Jiang, Universität Münster
Dr. Hendrik Koesling, Universität Bielefeld

Gedruckt auf alterungsbeständigem Papier nach DIN-ISO 9706.

Für meine Tochter Sophia Claudia

Danksagung

Ich bedanke mich ganz herzlich bei allen, die mich bei der Durchführung und Erstellung dieser Arbeit unterstützt haben.

Ein besonderer Dank geht an meinen Betreuer Prof. Dr. rer. nat. Christian Wöhler, welcher mich vor etwa dreieinhalb Jahren davon überzeugt hat, die Dissertation zu wagen. Er stand stets für Gespräche zur Verfügung und trug mit seinen Anregungen und Hinweisen wesentlich zur erfolgreichen Umsetzung dieser Arbeit bei.

Meinem Doktorvater Prof. Dr.-Ing. Franz Kummert möchte ich für seine schnellen und wertvollen Rückmeldungen danken. Sein unkompliziertes und freundschaftliches Auftreten sorgte dafür, dass ich mich auf jedes Treffen gefreut habe.

Ein weiterer Dank geht an Prof. Dr.-Ing. Rainer Ott für seine fachlichen Ratschläge und seine Akribie beim Korrekturlesen dieser Arbeit.

Ich danke Dr.-Ing. Ulrich Kreßel für seine fachliche und persönliche Unterstützung. Viele Diskussionen mit ihm gingen über die bloße Betreuung hinaus und haben meinen Horizont erweitert.

Dr.-Ing. Lars Krüger und Frank Lindner danke ich für das stetige Hinterfragen meiner Ideen und den Ansporn, dass es immer eine bessere Lösung gibt. Ihre Diskussionen haben oft meine Kenntnisse erweitert und mich nebenbei erheitert.

Ich danke meinen Doktorandenkollegen Björn Barrois und Christoph Hermes für das angenehme Arbeitsklima, die Zusammenarbeit und ihre Freundschaft.

Meinen Kollegen in der Abteilung Umfelderfassung des Forschungszentrums Ulm der Daimler AG möchte ich danken für ihr Interesse, ihre Ideen und dafür, dass sie stets ein offenes Ohr für meine Fragen hatten.

Ein herzlicher Dank gilt meinen Eltern und meiner ganzen Familie. Die große Hilfsbereitschaft und die moralische sowie finanzielle Unterstützung hat meine Ausbildung erst ermöglicht.

Meiner Frau Stefanie Hahn möchte ich für ihre Hilfe bei der Korrektur, aber ganz besonders für ihre Motivation und Zuversicht sowie ihren liebevollen Rückhalt danken.

Kurzfassung

In der vorliegenden Arbeit wird das Konzept eines intelligenten technischen Überwachungssystems zur Realisierung einer sicheren Interaktion zwischen Mensch und Industrieroboter in einfachen Arbeitsprozessen der Automobilproduktion vorgestellt. Es wird ein ganzheitliches System zur kamerabasierten Überwachung von menschlichen Arbeitern beschrieben. Neben der kamerabasierten Detektion, Segmentierung und Verfolgung der Arbeiter werden auch die ausgeführten Aktionen zur Durchführung des Arbeitsprozesses erkannt. Nur so kann der Industrieroboter geeignet reagieren und seinen Beitrag zum Interaktionsprozess leisten.

Der sich in der Szene befindende Mensch wird mit einem Mehrkamerasystem mit kleiner Basisbreite beobachtet. Dieses Kamerasystem ist dem Überwachungssystem SafetyEYE (www.safetyeye.com) sehr ähnlich. Es werden verschiedene Systeme zur 3D-Pose-Estimation und zum 3D-Tracking menschlicher Körperteile vorgestellt und diese hinsichtlich ihrer Eignung im realen Produktionsszenario untersucht. Zur 3D-Pose-Estimation werden top-down Verfahren, welche nur die synchronen Bilder des Mehrkamerasystems nutzen, und bottom-up Ansätze, welche auf zuvor berechneten 3D-Punktewolken mit Bewegungsinformation basieren, eingesetzt. Die entwickelten Trackingsysteme sind offen für die Erkennung und Verfolgung verschiedenster Körperteile gestaltet, da das Objektmodell einfach austauschbar ist. Am Beispiel der Verfolgung der menschlichen Hand-Unterarm-Extremität und der menschlichen Kopf-Schulter-Partie wird gezeigt, dass eine robuste und zeitlich stabile Verfolgung möglich ist.

Der multiokulare Contracting-Curve-Density (MOCCD) ist ein bekanntes top-down Verfahren zur 3D-Pose-Estimation in mindestens zwei 2D-Bildern. Grundlage der Pose-Estimation ist ein System aus kalibrierten Kameras in einem definierten Weltkoordinatensystem. Das Modell einer parametrischen 3D-Kurve wird dabei an die Bilddaten so angepasst, dass die Pixelstatistiken auf der inneren und äußeren Seite der parametrischen Kurve möglichst unterschiedlich sind. Der MOCCD-Algorithmus wird in einem ersten Trackingsystem verwendet, um die 3D-Pose eines Objekts zu schätzen. Die notwendige 3D-Pose-Prädiktion über die Zeit wird durch Kalman-Filter berechnet.

In einem zweiten Trackingsystem wird der neu entwickelte Shape-Flow-Algorithmus, eine Erweiterung des MOCCD-Algorithmus auf Bildsequenzen, verwendet, um die Kalman-Filter im ersten System zu ersetzen. Wichtig bei diesem System ist, dass das getrackte menschliche Körperteil durch ein raum-zeitliches 3D-Konturmodell beschrieben werden kann. Durch den Shape-Flow-Algorithmus wird eine direkt gemessene Bewegungsschätzung realisiert, dadurch können die zeitlichen Filter ersetzt werden und die Qualität der Bewegungsschätzung wird verbessert. Außerdem werden Latenzzeiten verringert, da es keine Einschwingphasen eines Filters mehr gibt.

In einem weiteren, neu entwickelten Trackingsystem werden zwei unterschiedliche Ansätze

zur 3D-Pose-Estimation miteinander gewichtet kombiniert. Dabei wird ein Verfahren verwendet, welches basierend auf 3D-Punktewolken mit Bewegungsinformation und dem bekannten Iterative-Closest-Point (ICP) das verwendete 3D-Modell anpasst. Zum anderen wird der MOCCD-Algorithmus eingesetzt, um das Objektmodell an die Bilddaten anzupassen. Ein Vorteil der verwendeten Ansätze ist, dass sie auf unterschiedlichen Eingabedaten basieren und somit auch einen Ausfall des jeweils anderen Verfahrens kompensieren können. Die Fusion der zwei Pose-Schätzungen basiert auf drei verschiedenen Ähnlichkeitsmaßen. Durch eine Bewegungsanalyse, basierend auf Szenenflussdaten, werden die Bewegungsparameter des Objekts bestimmt. Diese werden anschließend durch den Shape-Flow-Algorithmus verfeinert und sogar die Tiefengeschwindigkeit des Objekts direkt gemessen. Der Einsatz eines zeitlichen Filters ist nicht notwendig.

Ein viertes, neu entwickeltes Trackingsystem eignet sich zur Verfolgung beliebiger Objekte in Kamerabildern und 3D-Punktewolken. Die in den Kamerabildern getrackten Objekte werden nicht im Detail modelliert, sondern durch Ellipsoide beschrieben. Das System verwendet zur Pose-Estimation Bilddaten und 3D-Punktewolken mit Bewegungsinformation und basiert auf dem Mean-Shift-Algorithmus. Durch das Trackingsystem sollen alle bewegten Objekte in der beobachteten Szene verfolgt werden und auf einer höheren Stufe des Gesamtsystems, der Aktionserkennung, aus den Bewegungsdaten die Zuordnung zum Objekt, Aktionen oder Handlungen abgeleitet werden.

Um die Bewegungen des menschlichen Arbeiters beim Interaktionsprozess mit dem Industrieroboter verstehen zu können, wurde ein neuartiges System zur Aktionserkennung entwickelt. Als aktionsspezifische Merkmale werden Trajektorien von 3D-Punkten ausgewählt. In dieser Arbeit werden daher konsequent 3D-Informationen verwendet. Das Aktionserkennungssystem ist in der Lage, das getrackte Objekt, welches eine bekannte Bewegung ausführt, aus einer Menge getrackter Objekte zu erkennen. Es kann zusätzlich festgestellt auch werden, dass eine unbekannte Bewegung vorliegt, sodass durch die Vorhersage der Bewegung des Arbeiters mögliche Kollisionen mit dem Industrieroboter oder anderen Objekten erkannt werden. Zur Erkennung der bekannten Aktionen wird aus dem vorgegebenen Arbeitsablauf die Struktur eines Hidden-Markov-Modells (HMMs) abgeleitet und durch ein Partikelfilter der aktuellen Hidden-State des HMMs aus den 3D-Trajektorien geschätzt. Neu ist, dass das HMM nichtstationäre Transitionswahrscheinlichkeiten besitzt. Dies wird durch die zusätzliche Schätzung der Phase der zum Hidden-State zugehörigen Referenztrajektorien (3D-Trajektorien als Aktionsprototypen) möglich. Die Phase beschreibt dabei, zu welchem Anteil die zum Hidden-State zugehörigen Referenztrajektorien bereits durchlaufen sind. Zum Vergleich der 3D-Eingabetrajektorien mit den Aktionsprototypen wird die Levenshtein-Distanz von Trajektorien (LDT) verwendet.

Am Beispiel von Montagearbeiten eines Werkers an einem Motorblock wird in der Evaluierung auf realen Testsequenzen gezeigt, dass Erkennungsraten von mehr als 90% möglich sind. Außerdem können Unterbrechungen im Arbeitsablauf erkannt werden. Die Erkennung des bekannten zyklischen Arbeitsablaufs wird fortgesetzt, sobald der Arbeiter wieder bekannte Aktionen ausführt. Die Ergebnisse im betrachteten realitätsnahen Beispielszenario zeigen, dass durch die in dieser Arbeit beschriebenen Algorithmen einfache Arbeitsprozesse, wie z.B. „Handeinlegestationen“, ganzheitlich überwacht werden könnten und eine sichere physische Interaktion des Arbeiters mit dem Industrieroboter möglich ist.

Abstract

Today, industrial production processes in car manufacturing worldwide are characterised by either fully automated production sequences carried out solely by industrial robots or fully manual assembly steps where only humans work together on the same task. Up to now, close collaboration between humans and machines, especially industrial robots, is very limited and usually not possible due to safety concerns. Industrial production processes can increase efficiency by establishing a close collaboration of humans and machines exploiting their unique capabilities.

This thesis describes computer vision and pattern recognition methods to allow a safe interaction between human workers and industrial robots in a production environment. Vision methods are required for marker-less 3D pose estimation and tracking of the motion of human body parts and robot parts. With a motion analysis the vision based safety system is able to slow down or stop the robot early enough to avoid potentially hazardous situations. To have reliable depth information of the investigated scene, we use a small-baseline trinocular camera system similar to the SafetyEYE protection system (www.safetyeye.com).

Based on the example of tracking the human hand-forearm limb and the head-shoulder area it is shown that the developed 3D pose estimation and tracking algorithms allow for a robust, temporally stable, and metric accurate system performance. The developed 3D pose estimation algorithms are either model-based top-down techniques which directly use the three synchronously acquired images or bottom-up approaches which rely on motion-attributed 3D point clouds that are computed from the observed scene.

The Multiocular Contracting Curve Density (MOCCD) algorithm is a known top-down pose estimation technique based on pixel statistics around a contour model projected into the images from several cameras. The MOCCD algorithm is applied to track the 3D pose and some deformation parameters of the hand-forearm limb and the head-shoulder area with a traditional Kalman Filter framework.

In a second system the newly developed Shape Flow algorithm, a temporal extension of the MOCCD approach, is used to replace the Kalman Filter framework. The Shape Flow algorithm uses a spatio-temporal model of the tracked object and is able to obtain an estimate of the instantaneous motion properties, thus no temporal filtering is required. Obtaining instantaneous motion information is of essential importance in the context of safe human-robot interaction, such that the proposed method do not apply temporal filtering – the latency time of a temporal filtering stage would result in delays unacceptable in our application scenario. The developed system allows for a computation of a top-down model-based 3D scene flow using directly the three synchronously acquired images from the small-baseline trinocular camera.

Another tracking system combines the MOCCD technique and a bottom-up approach which uses a motion-attributed 3D point cloud to estimate the object pose with the Iterative Closest

Point (ICP) algorithm. Because of the orthogonal properties of both approaches it is shown that a fusion of the pose estimates is favourable. The tracking is realised by a motion analysis which is based on motion-attributed 3D points belonging to the tracked object using an extended constraint-line approach. A further refinement of the obtained motion properties is done using the Shape Flow algorithm, no temporal filtering is applied.

The idea behind another newly developed tracking approach is to extract the motion of all moving objects in the observed scene with a 3D mean-shift tracking algorithm and a simple ellipsoid model. A graph based clustering stage extracts all moving objects from motion-attributed 3D point cloud of the observed scene. The proposed 3D mean-shift tracking approach in this thesis relies on hue or gray value histograms and motion-attributed 3D point cloud data. At each time step the newly developed recognition stage determines the relevant object (e.g. the hand) which performs the working actions.

To understand the behaviour of the human worker, consistently 3D information from the tracking stage is used. The newly developed recognition system is able to extract an object performing a known action out of a multitude of tracked objects. A sequence of working actions is recognised with a particle filter based non-stationary Hidden Markov Model framework, relying on the spatial context and a classification of the observed 3D trajectories using the Levenshtein Distance on Trajectories as a measure for the similarity between the observed trajectories and a set of reference trajectories. Based on an engine assembly scenario it is shown that the system achieves action-specific recognition rates of more than 90% and that the system is able to detect disturbances, i.e. interruptions of the sequence of working actions, by entering a safety mode, and it returns to the regular mode as soon as the working actions continue.

All experimental investigations in this thesis are performed on real-world image sequences displaying several test persons performing different working actions typically occurring in an industrial production scenario. In all example scenes, the background is cluttered, and the test persons wear various kinds of clothes. For evaluation, independently obtained ground truth data is used.

Inhaltsverzeichnis

Teil I

Einführung

Kapitel 1

Einleitung

1.1 Motivation

Produktionsprozesse im Automobilbau sind stark automatisiert und Industrieroboter aus einer modernen Produktion nicht mehr wegzudenken. Einige Prozesse, wie z.B. Zylinder- und Cockpitmontage, sind noch manuell und werden durch Werker ausgeführt. Ein Werker oder Arbeiter ist dabei eine Person, welche Arbeitsprozesse in der Produktion verrichtet. Diese manuell ausgeführten Prozesse lassen sich noch nicht zuverlässig und kostengünstig automatisiert durchführen. Bei der Cockpitmontage wird das Cockpit über ein Handling-Werkzeug durch den Werker an seine Position navigiert und vollständig manuell integriert. Bei anderen Prozessen, z.B. bei der Anbringung der Dachverkleidung, ist es aufgrund der Komplexität und Sperrigkeit der Aufgabe notwendig, diese vollständig manuell zu gestalten. Folge der manuellen Ausführung ist, dass solche Prozesse oft sehr aufwendig und anfällig für Montagefehler sind. Die Automatisierung vieler manueller Produktionsprozesse ist beim gegenwärtigen Stand der Technik unmöglich, da aktuelle Industrierobotersysteme nicht über die notwendigen hochentwickelten kognitiven sowie motorischen Fähigkeiten verfügen.

Die Interaktion von Mensch und Industrieroboter (Abbildung 1.1) im Produktionsumfeld eröffnet die Möglichkeit, vollständig manuelle Prozesse durch intelligente technische Systeme teilweise zu automatisieren. Weiterhin ist eine Interaktion attraktiv im Hinblick auf Flexibilität der Produktion, erhöhte Produktivität sowie verbesserte Qualität des Produktionsprozesses. Ziel der Zusammenarbeit zwischen Mensch und Roboter ist es, die Stärken beider optimal zu nutzen. Der Mensch kann schnell und intelligent auf unvorhersehbare Ereignisse reagieren sowie sich auf die unterschiedlichsten Aufgabenstellungen einstellen. Ein Industrieroboter hingegen ist in der Lage, kraftaufwendige und sich vielfach wiederholende Tätigkeiten effektiv, präzise und ohne Ermüdungserscheinungen auszuführen. Ein Beispiel für eine solche Zusammenarbeit ist das Halten von schweren Werkstücken, sodass der menschliche Arbeiter in einer ergonomisch günstigen Haltung arbeiten kann. Ein weiteres Einsatzgebiet sind Hol- und Bringdienste, bei denen der Roboter das benötigte Werkstück oder Werkzeug zum menschlichen Arbeiter bringt. Auf diese Weise kann die Arbeitszeit des Arbeiters effektiv genutzt werden und er muss seine Konzentration nicht durch andere Tätigkeiten unterbrechen.

Die direkte physische Interaktion zwischen Mensch und Industrieroboter ist nicht ohne Risiko, denn ein Zusammenstoß könnte verheerende Folgen für den Menschen nach sich ziehen. Aus diesem Grund wird derzeit der Arbeitsraum des Roboters durch Schutzeinrichtungen gegen den Menschen abgeschirmt. Diese Schutzeinrichtungen können Lichtschranken, Laser-

scanner, taktile Sensoren, feste Schutzzäune oder seit 2008 das Kamerasystem SafetyEYE[1] sein. Sobald der Mensch die Schutzeinrichtungen überwindet und dadurch eine Gefahr für ihn besteht, müssen nach *DIN EN ISO 12100* automatische Abschaltsysteme den Roboter zum Stillstand bringen. Sollten sich Roboter und Mensch den Arbeitsraum teilen, herrschen derzeit noch strenge Restriktionen. Der Roboter darf nur mit sehr langsamer Geschwindigkeit arbeiten, und seine Kraft darf 150 Newton nicht überschreiten (*DIN EN ISO 10218*). Durch die entwickelten intelligenten Systeme soll das beschriebene Risiko bei der direkten physischen Interaktion zwischen Mensch und Industrieroboter durch ein intelligentes technisches System beseitigt werden. Das bereits im Automobilbau als Schutzsystem eingesetzte Kamerasystem

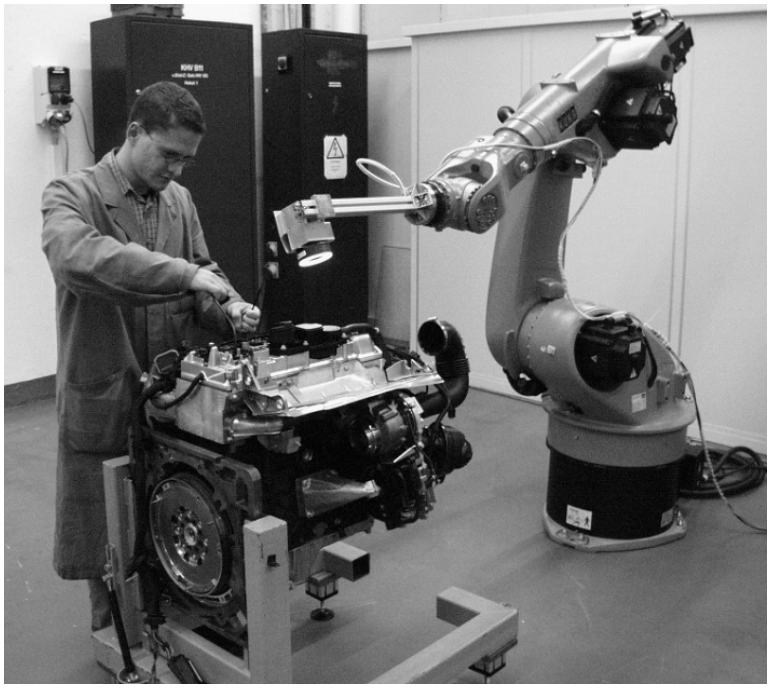

Abbildung 1.1: Vision einer Interaktion zwischen Mensch und Roboter.

SafetyEYE ermöglicht eine Überwachung von Arbeiter und Roboter. Dabei wird erkannt, ob sich irgendeine Form von Materie im zuvor definierten Schutzraum befindet, aber nicht betrachtet um was es sich dabei handelt oder ob dieses Objekt Aktionen ausführt. Franke (2006) stellt in seiner Diplomarbeit ein auf dem SafetyEYE-System basierendes Verfahren vor, mit dem ein Industrieroboter während seiner Arbeit überwacht und beispielsweise der sich zyklisch wiederholende Bewegungsablauf erlernt werden kann. Weiterhin eröffnet sich die Möglichkeit mit dem selben Kamerasystem die am Prozess beteiligten menschlichen Körperteile, z.B. den Unterarm, zu erkennen und deren Bewegungsverhalten zu ermitteln. Durch die Bewegungsvorhersage des Roboters und die Extrapolation des Bewegungsverhaltens der menschlichen Körperteile könnte eine mögliche Kollision vorhergesagt und durch einen Notstopp des Roboters vermieden werden. Außerdem ist es als Grundlage einer Interaktion notwendig, die ausgeführten Handlungen und Aktionen des Menschen zu erkennen, sodass der Roboter als möglicher Interaktionspartner darauf geeignet reagieren kann.

[1] www.safetyeye.com; Winkler (2006)

1.2 Ziele der Dissertation und angestrebtes Szenario

Im Fokus der Arbeit stehen die zwei Schwerpunktthemen zur Realisierung der Vision „Sichere Mensch-Industrieroboter-Interaktion“ mit einem intelligenten technischen Überwachungssystem. Der erste Schwerpunkt eines solchen technischen Systems beinhaltet die Segmentierung, Verfolgung und Bewegungsvorhersage von beobachteten Objekten. Das zweite Schwerpunktthema ist die Erkennung von Aktionen, Handlungen und Handlungsabsichten, welche von den beobachteten Objekten ausgehen. Ziel der Dissertation ist der Entwurf, die Implementierung und Bewertung eines intelligenten technischen Systems, welches eine Lösung für beide Schwerpunktthemen aufzeigt und auf realen Daten aus einem Produktionsszenario evaluiert wird. In realen Umgebungen, wie beispielsweise in der Automobilproduktion, gibt es verschiedene Herausforderungen, welche von einem zukünftigen Überwachungssystem beherrscht werden müssen. Die Herausforderungen sind die sichere Erkennung von Objekten, Aktionen und Handlungsabsichten unter den Szenariobedingungen:

- teilweise oder ganze Verdeckung des Zielobjekts durch andere Objekte in der Szene,
- Belichtungsänderungen,
- unerwartetes Verhalten des Zielobjekts,
- unkontrollierte Bewegungen,
- komplexe Objektformen,
- Veränderung der Objektform und des Aussehens,
- Objekte und/oder Hintergründe, die dem Zielobjekt sehr ähnlich sind,
- Bildrauschen und
- Echtzeitbedingungen.

Das entwickelte System soll in der Lage sein, diese Herausforderungen zu beherrschen. Die ermittelten Bewegungsparameter der Objekte sollen dazu geeignet sein, das Bewegungsverhalten des betrachteten Körperteils des menschlichen Werkers über kurze Zeiträume zu extrapolieren und so beispielsweise die Gefahr einer Kollision mit anderen, in der Szene befindlichen, bewegten Objekten zu erkennen. Am Beispiel von Montagearbeiten eines Werkers an einem Motorblock soll gezeigt werden, dass es möglich ist die am Prozess beteiligten menschlichen Körperteile robust und sicher über die Zeit zu verfolgen. Außerdem könnte eine Langzeitextrapolation der Bewegung mögliche Kollisionen mit anderen bewegten Objekten verhindern. Erst das Erkennen von Handlungen und Handlungsabsichten des menschlichen Arbeiters ermöglicht eine direkte Interaktion mit dem Industrieroboter. Außerdem können von der normalen Tätigkeit abweichende Handlungen erkannt und so mögliche Kollisionen mit dem Industrieroboter verhindert werden. Erste Anwendung könnte das System im Bereich der Vollständigkeit bzw. Absicherung bei der Motorenmontage oder bei „Handeinlegestationen“ finden.

Das dieser Dissertation zugrunde liegende Szenario sieht die Verwendung eines trinokularen Kamerasystems mit kleiner Basisbreite vor, welches ähnlich dem Sensor des bereits zertifizierten und eingesetzten Überwachungssystems SafetyEYE ist. Weiterhin kann in der Arbeit nicht davon ausgegangen werden, dass ein personenspezifisches 3D-Modell des Arbeiters vorliegt. Es kann aber angenommen werden, dass der zyklische vorgegebene Arbeitsablauf des Werkers und die 3D-Position der Interaktionsobjekte (Motor, Roboter, etc.) bekannt sind.

1.3 Gliederung

Diese Arbeit ist in vier Teile und zehn Kapitel untergliedert. Im Anschluss an die Einleitung wird im zweiten Kapitel ein Überblick über den für diese Arbeit relevanten Stand der Technik gegeben.

Im zweiten Teil dieser Dissertation werden in den Kapiteln 3 bis 6 Trackingsysteme zur bildbasierten Verfolgung der Körperteile eines in einem industriellen Produktionsumfeld agierenden Menschen vorgestellt. Das siebte Kapitel schließt den zweiten Teil dieser Dissertation mit den experimentellen Untersuchungen zum 3D-Tracking von Körperteilen ab. Es werden die Ergebnisse der Trackingsysteme aus den Kapiteln 3 bis 6 auf verschiedenen realen Testsequenzen qualitativ und quantitativ evaluiert.

Der dritte Teil dieser Dissertation beschreibt und evaluiert ein neu entwickeltes System zur 3D-Bewegungs- und Aktionserkennung. Notwendige Grundvoraussetzung ist ein Trackingsystem zur kamerabasierten Verfolgung von Objekten. Dabei kann jedes der im zweiten Teil dieser Dissertation beschriebenen Trackingsysteme eingesetzt werden. Im Kapitel 8 wird das Aktionserkennungssystem beschrieben und im neunten Kapitel werden dessen Ergebnisse auf realen Testsequenzen aufgezeigt und ausgewertet.

Abschließend erfolgt im letzten und vierten Teil dieser Dissertation eine Zusammenfassung der Arbeit und es wird ein Ausblick auf weitere Ideen und Möglichkeiten gegeben. Außerdem werden im Anhang die theoretischen Grundlagen zu einigen in dieser Arbeit verwendeten Ansätzen vorgestellt und das Schriftenverzeichnis des Autors dieser Arbeit wiedergegeben.

Kapitel 2

Stand der Technik

Derzeitiger Stand der Technik in der Automobilproduktion ist, dass die Arbeitsbereiche in den Arbeiten von Hand ausgeführt werden, räumlich getrennt sind von Bereichen, in denen Industrieroboter Arbeiten verrichten. Ein technisches System, welches eine enge Kooperation zwischen Mensch und Industrieroboter in der Produktion realisiert, ist also auf dem Markt noch nicht verfügbar. Dies liegt zum einen an den gesetzlichen Vorschriften, denn aus Sicherheitsgründen sind die Arbeitsräume von Menschen und Robotern strikt getrennt. Zum anderen ist ein Gesamtsystem, welches einen engen und intuitiven Interaktionsprozess realisiert, auch in der Forschung noch nicht existent.

Wöhler (2009) gibt eine Übersicht über die Entstehung und Entwicklung von Ansätzen zur Kollisionsvermeidung von Mensch und Industrieroboter in der Produktion. Er geht dabei auf frühe Systeme ein, welche meist auf lokalen Sensoren, z.B. Leitfähigkeits- oder Kapazitätssensoren, am Roboterarm basieren. Neben frühen kamerabasierten Ansätzen, werden auch Systeme vorgestellt, welche auf Laserscannerdaten basieren. Der Entstehungsweg der Sicherheits- und Schutzsysteme endet bei Wöhler (2009) mit dem kamerabasierten Überwachungssystem SafetyEYE [1].

Eine weitere gute Einführung bietet die Dissertation von Thiemermann (2005). Hier werden detailliert verschiedene Ansätze zur Kooperation von Mensch und Industrieroboter in der Montage vorgestellt. Die in der Dissertation entwickelte Pilotanlage zur direkten Mensch-Roboter-Kooperation basiert auf drei über dem Werker angeordneten Farbkameras. Die Hände und der Hals des Arbeiters werden über Farbpixelklassifikation erkannt und die Distanz zum Roboter limitiert dessen Geschwindigkeit.

Eine sehr aktives Forschungsprojekt ist SIMERO (Sichere Mensch/Roboter Koexistenz und Kooperation) (Ebert, 2003; Gecks u. Henrich, 2006, 2007; Henrich u. Gecks, 2008; Gecks u. Henrich, 2009; Fischer u. Henrich, 2009). Wie der Name schon sagt, wird hier ein kamerabasiertes Überwachungssystem zur sicheren Koexistenz von Mensch und Industrieroboter entwickelt. Ebert (2003) sowie Henrich u. Gecks (2008) realisieren mit vier kalibrierten und um die Roboterzelle verteilten Kameras ein Sicherheitssystem. Durch Pixelklassifikation (Gecks u. Henrich, 2006) werden bewegte Vordergrundobjekte vom Schutzraum-Hintergrund getrennt und somit Hindernisse auf der Bahn des Roboters erkannt. Fischer u. Henrich (2009) erweitern das System, indem PMD-Sensoren (engl. *photonic mixer device*) eingesetzt werden. Aus den Tiefenbildern der PMD-Sensoren wird eine 3D-Rekonstruktion der Roboterumgebung berechnet. Da der zukünftige Roboterpfad bekannt ist, können mögliche Kollisionen mit dem

[1] www.safetyeye.com; Winkler (2006)

Roboter erkannt werden. Die Information einer möglichen Kollision wird verwendet, um wie von Gecks u. Henrich (2009) beschrieben, eine neue Bewegungstrajektorie für den Roboter zu berechnen, sodass die Kollision mit dem Objekt im Schutzraum vermieden wird.

Winkler (2007) stellt ein Schutzraumüberwachungssystem basierend auf einem PMD-Sensor vor. Da die Position und die Bewegung des Industrieroboters bekannt ist, kann zu jedem Zeitschritt ein virtuelles Modell des Roboters berechnet und dieses aus dem Tiefenbild entfernt werden. Ein unbekanntes Objekt kann so sehr einfach detektiert und mögliche Kollision vermieden werden.

Alle der bisher erwähnten Arbeiten haben gemein, dass sie irgendeine Form von Materie im zuvor definierten Schutzraum erkennen, aber nicht betrachten, um was es sich dabei handelt oder ob dieses Objekt Aktionen ausführt. In den meisten der bisher erwähnten Arbeiten wird außerdem der Industrieroboter in den Vordergrund gestellt, d.h. es werden vornehmlich Themen wie Robotersteuerung oder Pfadplanung behandelt. In dieser Arbeit werden die Grundlagen eines intuitiven und ergonomischen Interaktionsprozesses aus Sicht des Roboters behandelt, d.h. der Mensch und seine Bewegungen werden in den Vordergrund gestellt. Dies bedeutet, dass die am Prozess beteiligten menschlichen Körperteile, z.B. der Unterarm, ohne zusätzliche Marken oder spezielle Anzüge dreidimensional in den Bildfolgen segmentiert und über die Zeit verfolgt werden müssen. Weiterhin ist es wichtig, die Aktionen des Menschen aus den Bewegungsinformationen, welche aus den Bilddaten extrahiert werden, zu erkennen.

In Abbildung 2.1 ist das Modell eines technischen Systems zur Realisierung des Interaktionsprozesses aus Sicht des Industrieroboters dargestellt. Von einem Eingabesensor (z.B. Kamerasystem, Laserscanner, PMD-Sensor, etc.) werden Daten der überwachten Szene geliefert. In einem zweiten Modul werden die am Prozess beteiligten Körperteile des Menschen in den Bildfolgen des Kamerasystems detektiert, segmentiert und über die Zeit verfolgt. Die berechneten Bewegungsdaten oder daraus extrahierte Merkmale dienen einem Aktionserkennungsmodul als Eingabedaten. Ausgehend vom allgemeinen Modell des technischen Systems zur Realisierung

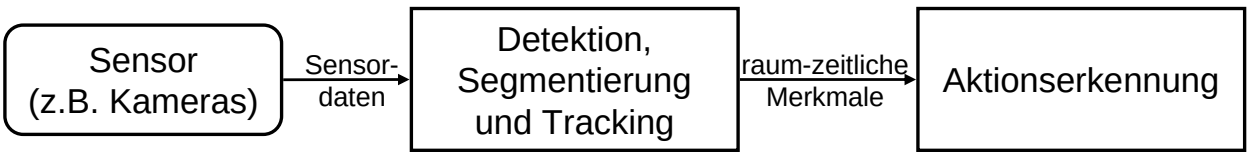

Abbildung 2.1: Das Modell eines technischen Systems zur Verfolgung von Objekten und Erkennung von Aktionen.

des Interaktionsprozesses (Abbildung 2.1) wird nun ein Überblick über vorhandene Ansätze zur Realisierung der beiden zentralen Module gegeben. Die aus den Zielen dieser Arbeit abgeleitete 3D-Verfolgung von Werkern setzt in vielen Fällen die Modellierung einzelner Körperteile oder des gesamten Körpers voraus. Daher wird im nächsten Abschnitt ein Überblick über vorhandene Ansätze zur Modellierung von menschlichen Körperteilen gegeben. Danach werden Verfahren zur bildbasierten Segmentierung und Verfolgung von Menschen vorgestellt. Abschließend werden Ansätze zur Handlungs- und Aktionserkennung aufgezeigt, da dies eine weitere wichtige Grundlage für Interaktionsprozesse zwischen Mensch und Industrieroboter darstellt.

2.1 Modellierung von Körperteilen des Menschen

Dieser Abschnitt beschäftigt sich mit der Modellierung von menschlichen Körperteilen zur modellbasierten Segmentierung und Verfolgung. Als Erstes wird eine Ordnung der Modelle in Anlehnung an Jain u. a. (1998) definiert und bestehende Verfahren mit ihren zugehörigen Modellen eingeordnet. Abbildung 2.2 zeigt einen Überblick über mögliche Ansätze zur Modellierung von Körperteilen.

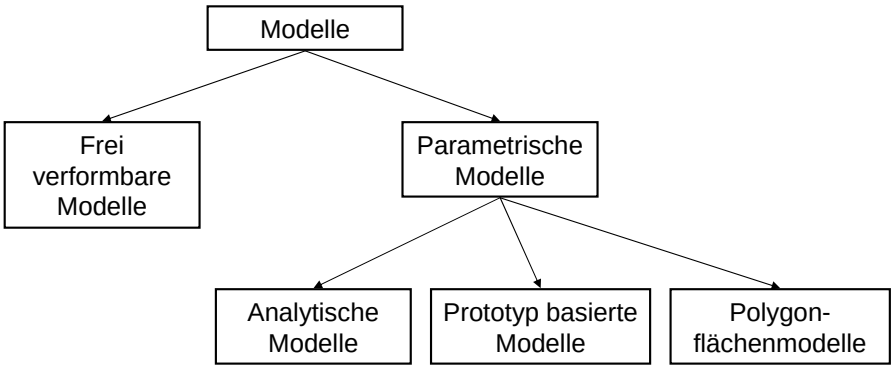

Abbildung 2.2: Überblick über Ansätze zur Modellierung von Körperteilen.

2.1.1 Frei verformbare Modelle

Aktive Konturen (engl. *active contours*) wurden durch Kass u. a. (1988) als sogenannte „Snake-Modelle“ eingeführt. Bei diesem Ansatz wird eine Energiefunktion minimiert und dabei die „Snake“ an die Objektkontur angepasst. Weitere Autoren (Amini u. a., 1988; Cohen, 1991; Williams u. Shah, 1992; Neuenschwander u. a., 1997; Xu u. Prince, 1998) haben diesen Ansatz adaptiert und erweitert. Die dabei genutzten Modelle sind offene und geschlossene Kurven im 2D- oder 3D-Raum, die durch Spline-Interpolation zwischen Kontrollpunkten entstehen. Diese Modelle sind frei verformbar und können sich somit auch an sehr komplizierte Objektkonturen anpassen. Sie besitzen keine globale Struktur, es wird also kein direktes a-priori Wissen über die Objektkontur genutzt. A-priori Wissen wird nur bezüglich der lokalen Randbedingungen verwendet, z.B. dass die Kurve „glatt“ ist, also keine Ecken besitzt.

Anwendung finden frei verformbare Modelle zur Modellierung von Körperteilen bei Kass u. a. (1988) sowie Jain u. a. (1998). Kass u. a. (1988) nutzen ein 2D-Snake-Modell, um den Umriss des menschlichen Mundes zu modellieren. Jain u. a. (1998) verwenden ein 2D-Snake-Modell zur Modellierung der menschlichen Hand.

Wertung: Der Vorteil der frei verformbaren Modelle ist, dass sie sich zur Approximation beliebiger menschlicher Körperteile verwenden lassen. Dies ist zugleich aber auch der größte Nachteil, da es bei der Anpassung von aktiven Konturen häufig vorkommt, dass Teile des Hintergrunds mit in die Segmentierung des Körperteils einbezogen werden. In realen Umgebungen mit strukturierten Hintergründen scheint die Anwendung von aktiven Konturen zur Modellierung daher nicht sinnvoll.

2.1.2 Parametrische Modelle

Diese Modellkategorie wird immer dann genutzt, wenn a-priori Wissen über die geometrische Form der Objektkontur vorhanden ist. Mit parametrischen Modellen ist es möglich, eine kompakte Beschreibung der Form der Objektkontur zu erreichen. Parametrische Modelle untergliedern sich in analytische Modelle, Prototyp-Modelle und Polygonflächenmodelle.

2.1.3 Analytische Modelle

Analytische Modelle werden durch eine Menge analytischer Kurven im 2D oder 3D definiert, z.B. Linien, Ellipsen, Kreise, Rechtecke, Polygone, Kugeln oder Quader. Die geometrische Form des Modells wird verändert, indem die Parameter des Modells geändert werden. Dadurch ist es möglich, ein Objekt sehr kompakt zu beschreiben. Es gibt eine Reihe von Autoren, die analytische Modelle nutzen.

Yuille u. a. (1989) beschreiben mit einem analytischen 2D-Modell ein Auge. Das Modell besteht aus zwei Parabeln, einem Kreis und einem Winkel, welcher die Rotation des Auges beschreibt. Das Modell besteht aus elf Parametern.

Delamarre u. Faugeras (1998) nutzen ein 3D-Modell der menschlichen Hand, welches aus Kegelstümpfen und Kugeln besteht und 27 Parameter besitzt. Jeder Finger besteht aus vier abgeschnittenen Kugeln und zwischen diesen Kugeln aus Kegelstümpfen. Die Handfläche wird im Modell nicht berücksichtigt.

Lu u. a. (2003) setzen ein analytisches 3D-Modell zur Erkennung der menschlichen Hand ein. In diesem Modell werden die Finger mit drei Zylindern und die Handfläche als Quader modelliert.

Eine 3D-Fläche, die durch eine algebraische Gleichung zweiter Ordnung definiert wird, heißt im Englischen *Quadric*. Verschiedene Veröffentlichungen (Gavrila u. Davis, 1996; Stenger, 2004; Sundaresan u. Chellappa, 2006) verwenden Quadrics zur Modellierung von Körperteilen. Gavrila u. Davis (1996) nutzen ein Gesamtkörpermodell bestehend aus Quadrics, um eine 3D-Lagebestimmung der beobachteten Person durchzuführen. Diese 3D-Lagebestimmung wird im englischen als *3D-Pose-Estimation* bezeichnet. Das Modell hat insgesamt 22 Parameter und wird spezifisch auf die zu verfolgende Person angepasst, sodass die Parameter der Quadrics nicht geschätzt werden müssen. Sundaresan u. Chellappa (2006) verwenden ein Ganzkörpermodell bestehend aus sogenannten Super-Quadrics, ähnlich wie Gavrila u. Davis (1996). Auch hier wird das Modell zuvor an die Person angepasst und das angepasste Modell dann zum Tracking verwendet. Der Parametervektor besteht aus der 3D-Pose des Torsos (sechs Freiheitsgrade) und aus jeweils drei Freiheitsgraden für jedes weitere Gelenk.

Stenger (2004) modelliert die menschliche Hand mit Quadrics und erzielt damit eine sehr detailgetreue Modellierung (Abbildung 2.3). Das Modell besitzt 27 Parameter.

Wu u. a. (2005) nutzen ein analytisches 2D-Modell der menschlichen Hand. Handfläche und Daumen werden als starres, planares Objekt und die Finger aus drei planaren Rechtecken modelliert. Die Breite und Länge der Rechtecke wird dabei individuell an die zu erkennende Hand angepasst.

Ein analytisches 2D-Modell wird von Treptow u. a. (2005) genutzt, um die Kopf-Schulter-

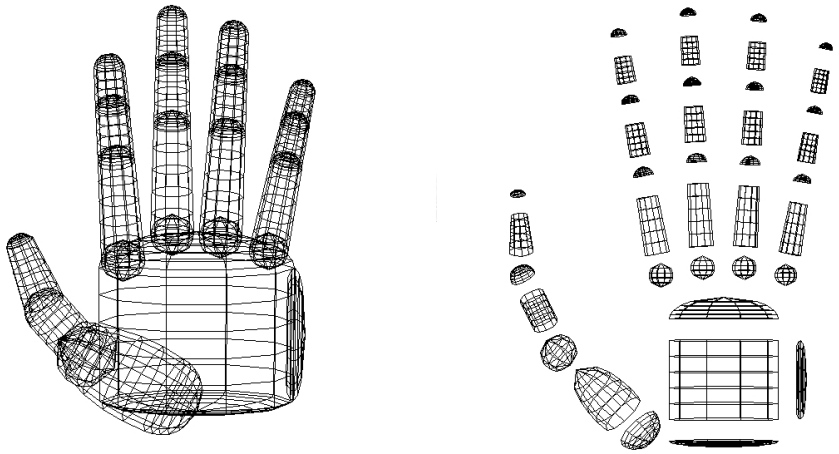

Abbildung 2.3: Links: 3D-Handmodell von Stenger (2004). Rechts: Einzelelemente des 3D-Modells zur besseren Visualisierung. Quelle: Stenger (2004).

Silhouette eines Menschen zu beschreiben. Das Modell besteht aus zwei Ellipsen und besitzt fünf Parameter.

Schmidt u. a. (2006) verwenden ein analytisches 3D-Modell, um den Oberkörper eines Menschen zu modellieren. Dabei werden Hand, Unterarm, Oberarm, Kopf und restlicher Oberkörper mit einem Zylinder mit elliptischer Grundfläche modelliert. Das Modell verfügt über 14 Parameter.

Wertung: Diese Modellkategorie scheint für das angestrebten Szenario sinnvoll, da durch die einfache Beschreibung eine sehr große Modellierungsvielfalt mit wenigen Parametern besteht. Im angestrebten Szenario ist es aber nicht möglich, ein Modell speziell auf eine arbeitende Person anzupassen, daher muss ein hinreichend generelles Modell verwendet werden. Dieses muss sich auch an unterschiedliche Arbeiter anpassen lassen und beispielsweise auch eine Verfolgung von Arbeitern mit Handschuhen gewährleisten.

2.1.4 Prototyp-Modelle

Diese Kategorie von Modellen leitet die geometrische Form von einem Prototypen ab. Objektformen werden durch Ähnlichkeit mit sogenannten Standard- oder Prototyp-Modellen definiert, welche die wahrscheinliche, durchschnittliche oder charakteristische Form einer Klasse von Objekten (z.B. Unterarmen) beschreiben. Die Form des Prototyps kann mit Hilfe von a-priori Wissen manuell festgelegt oder aus einer Menge von Formen gelernt werden.

Erste bedeutende Vertreter dieser Modellklasse sind die *Active-Shape-Models* (ASMs) (Cootes u. a., 1994). Sie sind ein Punktverteilungsmodell (engl. *point distribution model*), bei dem in verschiedenen Trainingsbeispielen semantisch gleichwertige Stellen, an Objektkanten, durch korrespondierende Punkte markiert werden. Die Punkte eines Beispiels bilden dabei einen Formvektor, der die Form des zu segmentierenden Objektes an ausgewählten Stützstellen beschreibt. Durch eine Hauptkomponentenanalyse dieser Formvektoren wird die in den

Beispielen auftretende Formvarianz ermittelt. Der Mittelwertvektor der Formvektoren ergibt das Prototyp-Modell und mit wenigen ausgewählten Eigenvektoren der Kovarianzmatrix wird die Deformierung des Prototypen beschrieben. Ein Punktverteilungsmodell repräsentiert die Formvielfalt einer Objektklasse.

Cootes u. a. (1995) beschreiben mit einem 2D-Punktverteilungsmodell die Form einer künstlichen Hand.

Heap u. Hogg (1996) haben mit einem Magnet-Resonanz-Tomographen (MRT) 3D-Gitternetzmodelle für verschiedene Handposen erstellt. Mit Hilfe dieser 3D-Gitternetzmodelle wurde ein 3D-Punktverteilungsmodell der menschlichen Hand extrahiert, welches fünf Freiheitsgrade bezüglich der Deformierung besitzt. Um das Modell an die Hand anzupassen, sind 12 Parameter notwendig, denn zusätzlich zur Deformierung sind Rotation, Skalierung und Translation des Modells zu berücksichtigen. Mit dem erstellten 3D-Punktverteilungsmodell ist eine sehr genaue Modellierung der menschlichen Hand möglich, solange die zu erkennende Handhaltung nicht zu stark von einer der gelernten Handhaltungen abweicht.

Blake u. Isard (1998) modellieren den Umriss der Kopf- und Schulterpartie des Menschen mit einer 2D-Kurve, die durch die Verbindung von Kontrollpunkten mit kubischen B-Splines entsteht. Zur Anpassung eines B-Splines an eine im Bild enthaltene Personenkontur können die Kontrollpunkte direkt manipuliert werden. Auf diese Weise entstehen jedoch sehr leicht Konturen, die keinerlei Ähnlichkeit mit der Kontur einer Person aufweisen. Daher ist es sinnvoll, die erlaubten geometrischen Transformationen durch Verwendung eines sogenannten Formenraums (engl. *Shape Spaces*) (Blake u. Isard, 1998) einzuschränken. Ein Formenraum besteht aus einer Basiskurve (Prototyp), die durch einen Kontrollpunktvektor gegeben ist, sowie aus einer Transformationsmatrix, welche die an der Basiskurve erlaubten Transformationen (Translation, Skalierung und Rotation) angibt. Ein solches Modell wird als Formenraum-Modell bezeichnet. Blake u. Isard (1998) nutzen 2D-Formenraum-Modelle, um die Kontur einer menschlichen Hand zu modellieren.

Hanek (2004) verwendet ein 2D-Formenraum-Modell, um die Kopf-Schulter-Silhouette eines Menschen zu beschreiben. Das Modell wird in diesem Fall an die Objektkontur angepasst, indem das Prototyp-Modell (Basiskurve) translatiert (horizontal und vertikal), skaliert (horizontal, vertikal und diagonal) und rotiert wird.

Wertung: Ein Vorteil dieser Modellklasse ist die detaillierte Modellierung, auch Deformierungen können sehr kompakt mit wenigen Parametern beschrieben werden. Die angestrebte Applikation dieser Arbeit hat den Nachteil, dass die Erzeugung des 3D-Punktverteilungsmodells oder Formenraums nicht praktikabel ist. Gegenwärtig ist es ungeklärt, ob es aus Datenschutzgründen Probleme gibt, wenn arbeitende Personen in der Produktion dreidimensional vermessen werden.

2.1.5 Polygonflächenmodelle

Diese Modellkategorie basiert auf einem Skelettmodell eines menschlichen Körperteils oder Körpers. Zur Modellierung von Muskeln, Fleisch, Haut und Kleidung wird auf das Skelettmodell ein Polygonflächenmodell gesetzt. Bei diesem werden durch die Verbindung von Polygonen

(Drei- oder Vierecken) Oberflächen geformt. Die Komplexität der Oberfläche ist dabei von der Anzahl der Polygone abhängig. Da moderne 3D-Grafikkarten solche Datenstrukturen direkt verarbeiten können, kann diese Art der Modellierung sehr effizient sein. Ein Körpermodell entsteht durch Kombination mehrerer Flächen. Die Parameter eines Modells sind die Polygoneckpunkte sowie die Parameter des Skelettmodells (3D-Pose und Gelenkwinkel). Mit Hilfe von Polygonflächenmodellen kann der menschliche Körper sehr genau modelliert werden.

Plänkers u. Fua (2003) nutzen ein Polygonflächenmodell zur Modellierung der Haut des menschlichen Oberkörpers (Abbildung 2.4). Das Modell der Haut sitzt dabei auf einem Muskelmodell bestehend aus vielen Ellipsoiden, welche wiederum auf einem menschlichen Skelettmodell angebracht sind. Diese schichtenweise Modellierung führt zu einer sehr detaillierten Nachbildung des menschlichen Oberkörpers.

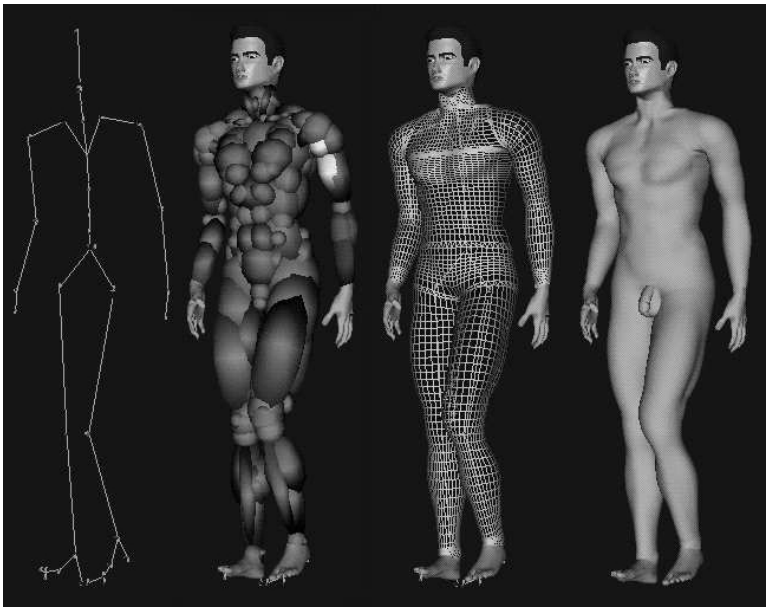

Abbildung 2.4: Das Modell von Plänkers u. Fua (2003). Von links nach rechts sind Skelettmodell, Muskelmodell, Polygonflächenmodell zur Modellierung der Haut und die finale Kombination abgebildet. Quelle: Plänkers u. Fua (2003).

Starck u. Hilton (2003) verwenden ein Polygonflächenmodell bestehend aus 8000 Polygonen und einem Skelett mit 17 bewegten Gelenken. Diese wird zur Rekonstruktion der 3D-Pose des Menschen genutzt. Das Modell besitzt sechs Parameter für die 3D-Pose im Weltkoordinatensystem, neun Parameter für die Knochenlängen und 25 Parameter für die einzelnen Gelenkwinkel.

In den Arbeiten von Rosenhahn u. a. (2005a,b, 2007, 2008) werden sehr detaillierte Polygonflächenmodelle von Körperteilen und des gesamten Körpers verwendet. Die verwendeten 3D-Modelle werden aus den Bildern von vier kalibrierten Kameras berechnet (Rosenhahn u. a., 2005a). Die Kameras müssen dabei so um die zu vermessende Person verteilt sein, dass jede Ansicht der Person aufgenommen wird. Eine weitere Anforderung des Systems ist ein bekannter statischer Hintergrund. Beim Oberkörpermodell von Rosenhahn u. a. (2005a) werden die Gelenkpositionen automatisch berechnet und neben dem Polygonflächenmodell jedes rigiden

Skelettabschnitts abgespeichert. Das somit erzeugte 3D-Modell des Oberkörpers wird von Rosenhahn u. a. (2005a,b) zur kamerabasierten Verfolgung und Pose-Estimation des menschlichen Oberkörpers eingesetzt. Beim Tracking werden neben der 3D-Pose des Torsos auch die Gelenkwinkel geschätzt. Insgesamt hat das Oberkörpermodell 21 Parameter. Die einzelnen Parameter der Polygonflächenmodelle werden nicht geschätzt, was dazu führt, dass für jede zu verfolgende Person ein spezielles Modell erstellt werden muss. Von Rosenhahn u. a. (2007) wird neben dem Menschen sogar die Kleidung des zu verfolgenden Objekts modelliert. Rosenhahn u. a. (2008) tracken einen Radfahrer sowie einen Snowboarder in Bildfolgen über die Zeit. Dabei wird in der Modellierung ein Fuß mit dem Ergometer oder dem Snowboard fest verbunden, der andere Fuß kann frei bewegt werden.

Gall u. a. (2009) verwenden ein Polygonflächenmodell mit integriertem Skelettmodell, dieses besteht aus 28 Parametern. Die Parameter des Flächenmodells werden während der Verfolgung in den Bildern nicht adaptiert, sondern nur die Parameter des Skelettmodells geschätzt. In der Arbeit wurde ein etwas allgemeineres Modell genutzt, dieses wurde an eine Person angepasst, bei der Evaluierung des Systems aber zur Verfolgung mehrerer Personen verwendet.

Wertung: Diese Modellkategorie erlaubt eine sehr detaillierte Nachbildung von Körperteilen, was bei einer modellbasierten Segmentierung zu einer sehr hohen Genauigkeit führt. Daher werden Polygonflächenmodelle mit integriertem Skelettmodell häufig zur Pose-Estimation bei Mehrkamerasystemen eingesetzt, denn diese ermöglichen eine Rundumansicht des zu verfolgenden Objekts. Eine zusätzliche Schätzung der Flächenparameter ist nicht nötig, wenn das Modell an die Person angepasst wurde. Bei einem Kamerasystem mit nur einer Ansicht, ist es bei einer modellbasierten Segmentierung notwendig, das Flächenmodell und nicht nur die Skelettparameter zu schätzen, da das Aussehen des Objekts von der Ansicht abhängig ist. Dies ist ein Nachteil von Polygonflächenmodellen mit integriertem Skelettmodell. Da die Komplexität des 3D-Modells sehr hoch ist, ist die Schätzung der Modellparameter in einer Umgebung mit strukturierten und sich ändernden Hintergründen sehr schwierig. Ein weiterer Nachteil ist, dass es bei einem industriellen Einsatz nicht möglich ist, jeden Arbeiter vor dem eigentlichen Arbeitsprozess zu vermessen.

2.2 Segmentierung und Verfolgung von Körperteilen in Bildern

In vielen Bildverarbeitungssystemen (beispielsweise in Überwachungssystemen oder bei der Mensch-Maschine-Interaktion) werden Personen in Bildern und Bildsequenzen detektiert, segmentiert und verfolgt, um ihre Handlung, z.B. Aktionen und Bewegungsabläufe, zu erkennen und zu überwachen. Die Segmentierung und Verfolgung von beobachteten Objekten stellt somit einen der Grundpfeiler eines Interaktionsprozesses zwischen Mensch und Industrieroboter dar.

Dieser Abschnitt gibt einen Überblick über den aktuellen Stand der Forschung im Bereich Segmentierung, Pose-Estimation und Verfolgung (Tracking) von Menschen oder Körperteilen von Menschen in Bildern und Bildfolgen. Innerhalb der letzten zehn Jahre hat die Zahl der

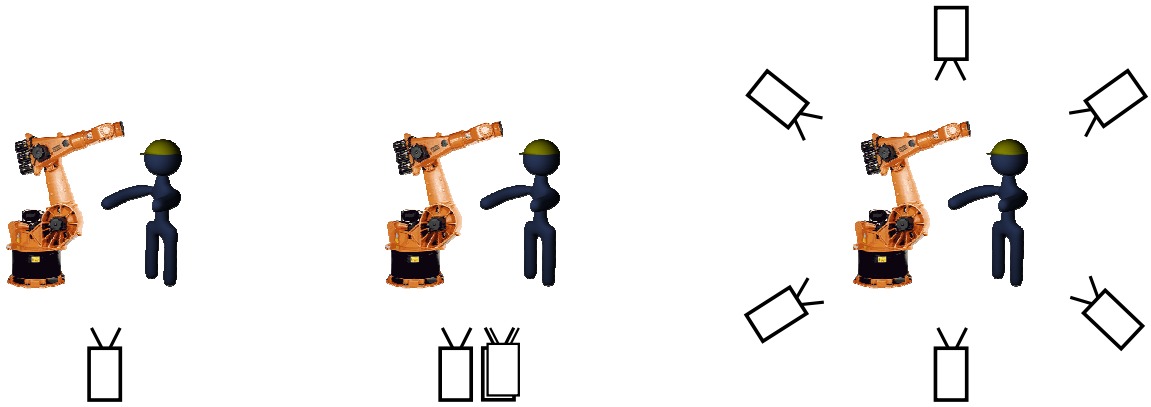

Abbildung 2.5: Schematische Darstellung der drei Kategorien von Kamerasystemen. Links: Monokulare Systeme. Mitte: Mehrkamerasysteme mit kleiner Basisbreite, welche nur eine Blickrichtung haben. Rechts: Mehrkamerasysteme mit großer Basisbreite, die Kameras sind so angeordnet, dass die zu überwachende Szene aus verschiedenen sich überlappenden Blickrichtungen observiert wird.

Veröffentlichungen in diesen Bereichen rapide zugenommen. Der Hauptgrund ist neben dem gestiegenen Bedarf an Sicherheitstechnik vor allem darin zu sehen, dass die Rechenleistung günstiger PCs inzwischen so hoch ist, dass auch rechenintensive Algorithmen in Echtzeit betrieben werden können. Zusammen mit mittlerweile kostengünstigen Aufnahme- und Übertragungsmöglichkeiten von digitalen Videobildern, ist die Grundlage für kommerziell interessante Applikationen und Produkte gegeben. Von einigen Autoren (Gavrila, 1999; Aggarwal u. Cai, 1999; Liang Wang, 2003; Sappa u. a., 2005; Lepetit u. Fua, 2005; Yilmaz u. a., 2006; Moeslund u. a., 2006) gibt es detaillierte Zusammenfassungen über Veröffentlichungen im Bereich der videobasierten Segmentierung, Pose-Estimation, Verfolgung und Bewegungsanalyse von Menschen. Im Folgenden werden einzelne recherchierte Verfahren kurz vorgestellt. Um eine bessere Übersicht zu gewährleisten, werden die Arbeiten nach dem eingesetzten Kamerasystem in: (i) monokulare Systeme, (ii) Systeme mit kleiner Basisbreite und (iii) Mehrkamerasysteme mit großer Basisbreite kategorisiert. Abbildung 2.5 zeigt eine schematische Darstellung der drei Kategorien von Kamerasystemen.

2.2.1 Monokulare Systeme

Die meisten Arbeiten basieren auf monokularen Systemen (Abbildung 2.5 (links)), es ist aber ein Trend erkennbar, denn aufgrund der preiswerten Hardware, der guten Rechenleistung und mittlerweile ausgereiften Methoden zur Stereoauswertung werden immer mehr Stereo- oder sogar Mehrkamerasysteme eingesetzt. In den Zusammenfassungen von Lepetit u. Fua (2005) sowie Yilmaz u. a. (2006) werden explizit nur Ansätze betrachtet, die monokulare Kamerasysteme nutzen. Die Arbeit von Lepetit u. Fua (2005) geht dabei auf modellbasierte Verfahren ein, die ein 3D-Tracking realisieren, während Yilmaz u. a. (2006) nur Ansätze zur 2D-Verfolgung betrachten. Sminchisescu (2006) beschreibt verschiedene Herausforderungen die bei der Segmentierung und dem Tracking von Personen in monokularen Bildsequenzen auftreten.

Heap u. Hogg (1996) nutzen ein 3D-Punktverteilungsmodell (Abschnitt 2.1.4) der menschlichen Hand. Die Segmentierung basiert dabei auf Kantenbildern. Das 3D-Modell wird in der Nähe der Hand initialisiert, d.h. es werden die Parameter für Deformierung, Rotation, Skalierung und Translation vorgegeben. Das 3D-Modell wird danach orthografisch in das Bild projiziert und iterativ angepasst, indem für definierte Punkte entlang des Modells der Abstand zur nächsten Kante ermittelt wird. Das Tracking ist derart realisiert, dass die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t + 1$ genutzt werden.

Blake u. Isard (1998) setzen einen Partikelfilter (CONDENSATION) ein, um die Kontur einer Hand oder den Umriss der Kopf- und Schulterpartie des Menschen zu tracken. In beiden Fällen wird ein 2D-Formenraum-Modell verwendet (Abschnitt 2.1.4). Im Beobachtungsmodell werden entlang von Konturnormalen Grauwertgradienten berechnet und diese über alle betrachteten Pixel entlang der Kontur aufsummiert. Aufgrund des einfachen Beobachtungsmodells ist es möglich, dass die Verfolgung vor strukturierten Hintergründen versagen kann. Ein weiteres Problem können Shading-Kanten sein, die z.B. beim menschlichen Unterarm auftreten.

Ein hierarchisches Template-Matching wird durch Gavrila (1999) eingesetzt, um die Objektkontur einer Person zu erkennen. Zum Einsatz kommen dabei 1000 von Hand gelabelte Templates in je fünf Skalierungsstufen. Mit Hilfe eines Cluster-Algorithmus wird eine Hierarchie erzeugt, welche für eine effiziente Detektion sorgt. Das Matching basiert auf der Korrelation des Templates mit einem distanztransformierten, orientierten Kantenbild.

Beim Mean-Shift-Tracking (Comaniciu u. a., 2000; Comaniciu u. Meer, 2002; Yang u. a., 2003; Russel, 2004) werden Bildmerkmale wie Farben, Helligkeit, Intensitäten oder Kanten benutzt, um Ähnlichkeiten zwischen einem vorher definierten Referenzobjekt und einem Bild zu finden. Die Merkmale werden dabei in Histogrammen repräsentiert, was diese Verfahren sehr effektiv macht. Comaniciu u. a. (2000) charakterisieren das in den Bildern zu verfolgende Objekt durch seine statistische Farbverteilung. Mit Hilfe der Mean-Shift-Optimierung wird die Ähnlichkeit von Zielobjekt und Input-Bild maximiert und somit das Referenzobjekt in den Bildfolgen über die Zeit verfolgt. Der CAMShift-Algorithmus von Bradski (1998) ist dem Mean-Shift-Tracking sehr ähnlich und wurde für die Verfolgung von Gesichtern und Körperteilen konzipiert. Ohne zusätzliche Komponenten, wie zeitliche Filter oder Farb/Schatten-Korrekturen, ist der Algorithmus einfach aufgebaut und implementiert. Porikli u. Tuzel (2003) stellen einen Gesichts-Tracker vor. Im Algorithmus wird der Mean-Shift-Ansatz mit *Gaussian-Mixture-Models* (GMM) kombiniert und es werden u.a. adaptive Hintergrundmodelle und Schattenentfernungstechniken verwendet. Der Algorithmus ist vom Aufbau her komplexer als der Ansatz bei CAMShift.

Stenger u. a. (2001) nutzen ein analytisches 3D-Modell (Abschnitt 2.1.3) der menschlichen Hand zu deren Segmentierung und zum Tracking in den Bildern einer monokularen Kamera. Dieser Ansatz erfordert, wie der von Heap u. Hogg (1996), eine manuelle Initialisierung. Dabei werden die Modellparameter des Handmodells manuell festgelegt und angenommen, dass sich die Form der Hand über die Sequenz nicht ändert. Das Handmodell wird mit dem Bild verglichen, indem entlang der Normalen des projizierten Modells der absolute Abstand zur stärksten Grauwert- oder Hautfarbkante berechnet wird. Je geringer der aufsummierte Abstand aller Normalen ist, desto besser stimmt das Modell mit dem Bild überein. Mit einem „*unscented*“ Kalman-Filter (UKF) und einem konstanten Beschleunigungsmodell wird das 3D-

Handmodell in den Bildern über die Zeit verfolgt, indem Translation (X-, Y- und Z-Richtung) und Rotation (Z-Richtung) ermittelt werden.

Athitsos u. Sclaroff (2003) setzen ein Template-Matching Verfahren zur Erkennung von Handposen ein. Die Templates werden erzeugt, indem synthetische 3D-Modelle der rechten Hand für 26 verschiedene Handposen berechnet werden. Aus 86 Blickrichtungen (auf einer das Modell umschließenden 3D-Kugel) werden die 3D-Modelle in die Bildebene projiziert und 48-mal rotiert. Somit entsteht eine Datenbank, die für jede Handpose 4182 Templates enthält. Das Matching basiert auf einer Korrelation der Templates mit distanztransformierten, orientierten Kantenbildern.

Das System von Agarwal u. Triggs (2004) ist in der Lage sehr schnelle Bewegungen, mit Hilfe trainierter Dynamiken zweiter Ordnung, zu verfolgen. Es werden zeitlich sehr robuste Verfolgungsergebnisse vor strukturierten Hintergründen erzielt. Zum Einsatz kommt ein analytisches 2D-Modell des gesamten Körpers (Abschnitt 2.1.3).

Hanek (2004) beschreibt in seiner Dissertation den Contracting-Curve-Density-Algorithmus (CCD) und den darauf basierenden CCD-Tracker, ein modellbasiertes Verfahren zur Verfolgung von Objekten. Es wird ein 2D-Formenraum-Modell (Abschnitt 2.1.4) verwendet, um die Kopf-Schulter-Silhouette eines Menschen zu tracken. Im Beobachtungsmodell wird der CCD-Algorithmus eingesetzt, welcher das Modell an das Bild anpasst. Zur Prädiktion der Modellparameter (Bewegungsmodell) wird ein zweistufiger autoregressiver Prozess (Blake u. Isard, 1998) eingesetzt, dessen Parameter zuvor auf einem Trainingsset gelernt werden. Kritisch bei dieser Form des Trackings ist, dass die ausgeführte Bewegung des Objekts nicht zu stark von der gelernten Bewegung abweichen darf. Bei einer Abweichung der zu verfolgenden Bewegung kann das Tracking leicht scheitern.

Treptow u. a. (2005) nutzen Wärmebilder und ein einfaches analytisches 2D-Modell (Abschnitt 2.1.3), um die Kopf-Schulter-Kontur eines Menschen mit einem Partikelfilter zu tracken. Im Beobachtungsmodell werden als Merkmale Gradienten entlang von Konturnormalen verwendet.

Schmidt u. a. (2006) verwenden ein analytisches 3D-Modell (Abschnitt 2.1.3), um den menschlichen Oberkörper mit einem Partikelfilter in Kamerabildern zu tracken. Das 3D-Modell muss in der Nähe des zu trackenden Objektes initialisiert werden. Im Beobachtungsmodell kommen als Merkmale Grauwertkanten, Grauwert-Grate[2], Hautfarbe und Farbmittelwerte zum Einsatz. Zum Tracking wird ein Kernel-Partikelfilter eingesetzt, welcher das Modell iterativ in jedem Bild anpasst und somit weniger Partikel benötigt als ein Standard-Partikelfilter. In der Prädiktion wird ein großer Teil der Partikel mit einem linearen Bewegungsmodell prädiziert und der Rest zufällig verstreut. Bei diesem Verfahren ist es besonders wichtig, dass eine genaue Initialisierung der Tiefe des Objekts bzgl. der Kamera sowie die Definition der Halbachsen der im Modell verwendeten Zylinder mit elliptischer Grundfläche vorliegt. Ist diese Initialisierung in der Anwendung möglich, realisiert das Verfahren eine dauerhafte und robuste, bildbasierte Verfolgung des menschlichen Oberkörpers, da im Beobachtungsmodell mehrere Merkmale zum Einsatz kommen. Schmidt u. Castrillon (2008) präsentieren ein Verfahren zur Initialisierung des Trackings.

Ramanan u. a. (2007) verwenden ein analytisches 2D-Modell (Abschnitt 2.1.3) des gesamten

[2]Grauwert-Grate sind längliche Kanten, punktförmige Maxima werden unterdrückt

Körpers und stellen ein selbstadaptierendes System zur Verfolgung von Personen vor. Mit Hilfe von Farb- und Kantenanalyse werden Körperteilhypothesen gespeichert und in neuen Bildern wiedergefunden. Dazu wird eine Mischung aus Template-Matching und Klassifikator benutzt. Das Verfahren ist sehr robust und auch in der Lage mehrere Personen zeitgleich zu verfolgen.

Es gibt eine Reihe von Arbeiten, die Ansätze des überwachten Lernens verwenden, um Menschen in Bildfolgen zu detektieren und zu verfolgen. Enzweiler u. Gavrila (2009) präsentieren eine umfassende experimentelle Studie und Zusammenfassung zur monokularen Fußgängererkennung mit Verfahren des überwachten Lernens. Auf einem Benchmark-Datensatz werden: (i) eine AdaBoost-Kaskade basierend auf Wavelets (Viola u. a., 2003), (ii) eine lineare *Support Vector Machine* (SVM) mit Merkmalen basierend auf *Histograms of Oriented Gradients* (HOG) (Dalal u. Triggs, 2005), (iii) ein Neuronales Netz mit lokal-rezeptiven Feldern (Wöhler u. Anlauf, 1999) und (iv) ein Ansatz zur Form-Textur-Detektion (Gavrila u. Munder, 2007) miteinander verglichen. Als Trackingsystem wird jeweils ein α-β-Tracker verwendet. Die Ergebnisse auf dem veröffentlichten Benchmark-Datensatz zeigen, dass HOG-Merkmale kombiniert mit einer linearen SMV den derzeit besten Ansatz bei den Verfahren des überwachten Lernens darstellen.

Wertung: Monokulare Systeme sind oftmals sehr robuste und zeitlich stabile Ansätze zur Detektion, Segmentierung und Verfolgung von Menschen in Kamerabildern. Auch bei sehr schwierigen Sequenzen, wie z.B. bei weit entfernten Überwachungskameras, Filmen oder Sportübertragungen, gibt es Verfahren, die in der Lage sind, die Menschen oder deren Körperteile in Bildern robust zu verfolgen. Weiterhin gibt es Verfahren, die auch bei extrem schnellen und komplexen Bewegungen, wie beispielsweise bei einem Sprint, das zu verfolgende Objekt nicht verlieren. Die Verfahren arbeiten auf einem sehr großen Tiefenbereich, einige der Ansätze schätzen sehr nahe und komplexe Handstellungen als Eingabesysteme für Computer und andere verfolgen Menschen in großen Entfernungen, z.B. aus dem Auto heraus oder in den Bildern von Überwachungskameras. Ein Großteil der Verfahren hat gemein, dass sie darauf konzipiert wurden, robust und zeitlich stabil zu sein, jedoch metrisch nicht sehr genau. Bei dem in dieser Arbeit betrachteten technischen Überwachungssystem zur Realisierung der Interaktion zwischen Mensch und Industrieroboter kommt es aber vor allem auf metrische Genauigkeit an, denn Kollisionen müssen verhindert werden. Dies führt dazu, dass rein monokulare Ansätze in dieser Arbeit nicht weiter betrachtet werden. Viele Ideen aus monokularen Ansätzen können aber auch in Mehrkamerasystemen verwendet werden.

2.2.2 Mehrkamerasysteme mit kleiner Basisbreite

Mehrkamerasystemen mit kleiner Basisbreite (Abbildung 2.5 (Mitte)) bestehen aus mindestens zwei Kameras und der Abstand des Objekts zum Kamerasystem ist mindestens eine Größenordnung größer als der Abstand der Kameras zueinander. Dies bedeutet, dass das zu verfolgende Objekt nur aus einer Ansicht aufgenommen wird und man daher nicht hinter das Objekt schauen kann. Dies trifft auf die meisten Stereokamerasysteme zu.

Delamarre u. Faugeras (1998) nutzen ein analytisches 3D-Modell (Abschnitt 2.1.3) der menschlichen Hand, um dieses an Tiefenbilder anzupassen. Bei diesem Ansatz wird angenommen, dass die Initialposition der Hand im 3D-Raum bekannt ist. Mit einem Stereokamera-

system wird ein Tiefenbild der Szene berechnet und das 3D-Handmodell mit Hilfe eines *Iterative-Closed-Point* Algorithmus (ICP) an das Tiefenbild angepasst. Zum Tracking wird ein Kalman-Filter mit den 27 Freiheitsgraden der Hand als Zustandsvektor genutzt.

Das von Lin (1999) vorgestellte Verfahren ist in der Lage, die Hand-Arm-Region einer Person in Bildern über die Zeit zu verfolgen. Es wird ein artikuliertes, analytisches 3D-Modell (Abschnitt 2.1.3) verwendet, um Oberarm, Unterarm und Hand zu modellieren. Dabei bestehen die Gelenke aus 3D-Kreisen (kreisrunde Ebenen im 3D-Raum, parallel zum optischen Sensor), welche miteinander verbunden sind. Für jedes Gelenk wird mit Hilfe einer dichten Tiefenkarte die 3D-Position und der Radius geschätzt, indem das Modell an Regionen gleicher Tiefe und ähnlich der Modellgröße angepasst wird. Es wird die Annahme gemacht, dass der Arm immer parallel zum optischen Sensor bewegt wird. Das Tracking ist derart realisiert, dass die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t+1$ genutzt werden. Dieser Ansatz erfordert, wie der von Heap u. Hogg (1996), eine manuelle Initialisierung zum Zeitschritt $t = 0$.

Demirdjian u. Darrell (2002) sowie Demirdjian (2003) verwenden ein analytisches Modell (Abschnitt 2.1.3) des Oberkörpers. Das Modell setzt sich aus jeweils einem Zylinder für Torso, beide Ober- und Unterarme sowie einer Kugel für den Kopf zusammen. Die rigiden Körperteile werden mittels ICP-Algorithmus einzeln an eine Punktwolke angepasst und später zu einer Gesamtpose des Oberköpers fusioniert. Die Initialisierung des Trackings basiert auf dem System von Darrell u. a. (2001). Aus Stereobildern wird ein 3D-Hintergrundmodell berechnet und Abweichungen vom Hintergrundmodell werden als bewegte Objekte in der Szene initialisiert. Das echtzeitfähige Tracking ist derart realisiert, dass die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t+1$ genutzt werden.

Plänkers u. Fua (2003) verwenden ein Stereokamerasystem, um den menschlichen Oberkörper in Bildern über die Zeit zu verfolgen. Es wird ein sehr detailliertes Polygonflächenmodell, wie im Abschnitt 2.1.5 beschrieben, verwendet. Das komplexe Modell wird angepasst, indem es durch Optimierung an eine 3D-Tiefenkarte und die zuvor extrahierte Silhouette des Körpers gefittet wird. Das Tracking wird realisiert, indem die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t+1$ genutzt werden. Eine Besonderheit des Systems ist, dass neben den Pose-Parametern auch das Körpermodell an die Person angepasst wird. Dieser Ansatz erfordert, wie der von Heap u. Hogg (1996), eine manuelle Initialisierung zum Zeitschritt $t = 0$.

Nickel u. a. (2004) verfolgen den menschlichen Kopf und die Hände, um Zeigegesten zu erkennen. Kopf und Hände werden als Ellipsoide modelliert (Abschnitt 2.1.3) und mit Hilfe von Tiefendaten sowie einer Hautfarbsegmentierung angepasst.

Narayanan u. a. (2005) tracken dem menschlichen Kopf, mit einer Ellipse (analytisches Modell, siehe Abschnitt 2.1.3). Aus einer Menge von aufgenommener Hintergrundbilder wird ein Disparitätsbild des Hintergrunds berechnet und damit in der späteren Anwendung die Silhouette der Vordergrundobjekte berechnet. Ausgehend von der Silhouette wird die Position der Kopfellipse und deren Größe geschätzt. Als Bewegungsmodell wird konstante Geschwindigkeit gewählt.

Knoop u. a. (2006) stellen das VooDoo-System vor. Es wird ein analytisches 3D-Modell (Abschnitt 2.1.3) des gesamten Körpers verwendet. Das Modell besteht aus degenerierten

Zylindern, d.h. die Grundflächen des Zylinders sind Ellipsen. Die fusionierten 3D-Eingabedaten des Systems stammen von einer Stereokamera und einem PMD-Sensor. Mit Hilfe eines ICP-Algorithmus wird das Modell an die 3D-Punktwolke angepasst. Das Tracking wird realisiert, indem die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t+1$ dienen. Dieser Ansatz erfordert eine manuelle Initialisierung zum Zeitschritt $t=0$.

Ein mobiles Stereokamera-System wird von Ess u. a. (2008) vorgestellt, die Kameras und der Rechner sind dabei in einen Kinderwagen eingebaut. Mit dem System wird in Fussgängerzonen eine Multi-Personen Verfolgung in Kamerabildern realisiert. Dabei wird ein Ansatz verwendet, welcher auf einem Tracking durch Detektion basiert. Das Detektionsmodul verwendet *Implicit-Shape-Models* (ISM). Durch *Structure-from-Motion* (SfM) wird ständig die Kamerakalibrierung neu berechnet und somit die Bodenebene geschätzt. Es wird angenommen, dass die Personen auf der Bodenebene stehen, dadurch können die verschiedenen 2D-Detektionen in den unterschiedlichen Kamerabildern unter Zuhilfenahme der Stereo-Tiefenkarte zu einer 3D-Detektion fusioniert werden.

Wertung: Die Verwendung eines Kamerasysteme mit kleiner Basisbreite scheint für das in dieser Arbeit angestrebte Applikationsszenario sinnvoll. Hinsichtlich der Messgenauigkeit von 3D-Daten ist ein solches System einem monokularen Kamerasystem überlegen und die Investitionskosten sowie der Hardware- und Kalibrieraufwand sind noch überschaubar. Durch Verwendung eines Kamerasystems mit drei zueinander senkrecht angeordneten Kameras lässt sich außerdem das Aperturproblem für Stereo-Algorithmen lösen. Probleme bei der Korrespondezsuche für Bildbereiche, welche eine ähnliche Orientierung wie eines der Stereo-Kamerapaare besitzen, werden durch das jeweils andere Kamerapaar aufgelöst. Ein weiter Vorteil besteht darin, dass es möglich ist, einen großen Teil der robusten und zeitlich stabilen monokularen Algorithmen auf ein Mehrkamerasystem mit kleiner Basisbreite zu erweitern. Das zertifizierte Überwachungssystem SafetyEYE wird bereits in Teilen der Produktion in der Automobilindustrie zur Schutzraumüberwachung eingesetzt, daher ist es außerdem sehr sinnvoll auf einem Kamerasystem, mit ähnlicher Sensoranordnung aufzubauen.

2.2.3 Mehrkamerasysteme mit großer Basisbreite

Mehrkamerasysteme (engl. *multi-view systems*) nutzen eine Vielzahl von Kameras (4, 8, 16 oder mehr), dabei sind die Kameras um die zu beobachtenden Szene verteilt (Abbildung 2.5 (rechts)). Eine weitere Besonderheit dieser Systeme ist, dass die Basisbreite der Kameras häufig genauso groß ist, wie der Abstand des Objekts zu den einzelnen Kameras. Durch die Anordnung der Kameras sind Mehrkamerasysteme mit großer Basisbreite in der Lage, das zu verfolgende Objekt aus mehreren sich überlappenden Ansichten zu beobachten. Somit kann eine komplette 3D-Objektrekonstruktion in nur einem Zeitschritt durchgeführt werden.

Der Ansatz von Gavrila u. Davis (1996) ist eine der ersten Arbeiten, in denen ein Mehrkamerasystem benutzt wird. Mit vier orthogonal angeordneten Kameras wird ein tanzendes Paar beobachtet und in Bildern verfolgt. Für jede Person werden analytische 3D-Modelle (Abschnitt 2.1.3) des gesamten Körpers verwendet. Die einzelnen Körperteile sind dabei als Quadrics modelliert. Das 3D-Modell ist direkt an die zu verfolgende Person angepasst und wird grob mit einer Hauptkomponentenanalyse (PCA) auf den vier Hintergrund subtrahierten An-

sichten initialisiert. Die bei der Verfolgung genutzte 3D-Pose-Estimation basiert auf einem *Chamfer-Matching* mit zuvor generierten Modell-Templates für alle vier Ansichten. Um den riesigen Suchraum zu verkleinern, wird ausgehend von einer prädizierten Objekt-Pose die neue Objekt-Pose gesucht. Die Arbeit von Hofmann u. Gavrila (2009) erweitert diesen Ansatz um ein Texturmodell. Während der Verfolgung wird das Texturmodell adaptiert und dadurch eine zeitlich robustere Verfolgung erreicht. Ein weitere Neuerung ist die Verwendung eines Multi-Hypothesen Prädiktionssystems, welches auf einem Sliding-Window Ansatz basiert. Dabei werden 200 prädizierte Posen mit Hilfe des Messsystems verifiziert und letztendlich die beste Modelladaption bestimmt.

Deutscher u. Reid (2005) nutzen drei Kameras, um eine Testperson zu verfolgen. Das verwendete analytische 3D-Gesamtkörpermodell (Abschnitt 2.1.3) besteht aus Kegelstümpfen für jedes rigide Körperteil und besitzt 29 Freiheitsgrade. Es wird ein Partikelfilter verwendet, welcher um Ideen aus der simulierten Abkühlung (engl. *simulated annealing*), einem heuristischen Optimierungsverfahren, erweitert wurde. Die Gewichte des Partikelfilters werden durch den Vergleich des projizierten Modells mit Kanten und Silhouetten, welche durch Hintergrundsubtraktion bestimmt werden, berechnet.

Sundaresan u. Chellappa (2006) verfolgen einen Menschen mit einem analytischen Gesamtkörpermodell (Abschnitt 2.1.3) und verwenden eine Systemkonfiguration bestehend aus acht Kameras. Jedes rigide Körperteil wird als Super-Quadric modelliert. Der Ansatz basiert auf der dreidimensionalen Bewegung des Körpermodells, diese wird aus dem optischen Fluss in den Bildern der einzelnen Kameras berechnet. Die Testpersonen tragen hautenge Anzüge mit einem Leopardenfellmuster und der Ansatz erfordert eine manuelle Initialisierung zum Zeitschritt $t = 0$.

Ziegler u. a. (2006) verfolgen den menschlichen Oberkörper mit einem analytischen 3D-Modell (Abschnitt 2.1.3), bestehend aus 14 Parametern. Als Eingabedaten dienen 3D-Punktwolken, welche mit vier Stereokamerasystemen erzeugt werden. Die 3D-Punktwolken werden mit einem Hintergrundmodell ausgedünnt, sodass nur Vordergrundpunkte in die Berechnung eingehen. Ausgehend vom Modell werden 3D-Modellpunkte extrahiert und mit Hilfe des ICP-Algorithmus an die 3D-Punktwolken angepasst. Das Tracking basiert auf einem *Unscented Kalman-Filter* (UKF) (Julier u. Uhlmann, 1997).

In den Arbeiten von Rosenhahn u. a. (2005a,b, 2007, 2008) werden wie im Abschnitt 2.1.5 bereits beschrieben, sehr detaillierte Polygonflächenmodelle zur Verfolgung des menschlichen Körpers in Bildern verwendet. Die Pose-Estimation basiert auf Silhouetten, welche mittels Level-Set-Segmentierung extrahiert werden. Von Rosenhahn u. a. (2005b) wird der menschliche Oberkörper in den Kamerabildern verfolgt und das Tracking realisiert, indem die ermittelten Parameter zum Zeitschritt t als Initialisierung für den Zeitschritt $t + \Delta t$ genutzt werden. Außerdem werden die Ergebnisse mit einem Marken-basierten System verglichen. Rosenhahn u. a. (2007) modellieren neben dem Menschen sogar die Kleidung des zu verfolgenden Objekts und beziehen diese auch in die Pose-Estimation ein. Von Rosenhahn u. a. (2008) wird ein Radfahrer sowie ein Snowboarder in den Bildfolgen verfolgt. Aus der Nutzung eines Sportgerätes ergeben sich Constraints, welche bei der Optimierung mit einbezogen werden, z.B. ergibt sich bei einem Radfahrer eine kreisförmige Bewegung der Füße.

Mündermann u. a. (2007) generieren die visuelle Hülle von Vordergrundobjekten aus den Bildern eines Mehrkamerasystems. Der sogenannte artikulierte ICP-Algorithmus kommt zum

Einsatz, um die Pose-Estimation zwischen Modell und der visuelle Hülle des Objekts durchzuführen. Es wird ein Polygonflächenmodell (Abschnitt 2.1.5) des gesamten menschlichen Körpers eingesetzt. Die Evaluierung des System wurde auf Kamerakonfigurationen mit 4, 8, 16, 32 und 64 Kameras durchgeführt. Es zeigt sich, dass die Genauigkeit des Systems mit der Anzahl der Kameras steigt.

Tyagi u. a. (2007) erweitern das Mean-Shift-Tracking (Comaniciu u. a., 2000; Comaniciu u. Meer, 2002) in den dreidimensionalen Raum. Der Kopf eines Menschen wird mit Hilfe eines Ellipsoiden getrackt. In den Experimenten wird ein System mit vier Kameras verwendet.

Gall u. a. (2009) verwenden wie Rosenhahn u. a. (2005b, 2007, 2008) ein Polygonflächenmodell (Abschnitt 2.1.5) und eine ähnliche lokale Optimierung zur Pose-Estimation. Um eine höhere zeitliche Stabilität der Verfolgung zu erreichen, wird außerdem eine stochastische Optimierung mittels eines Partikelfilters eingesetzt. Die Merkmale zur Berechnung der Partikel-Gewichte sind Silhouetten und Farben.

Wertung: Viele der Ansätze, die Mehrkamerasysteme mit großer Basisbreite verwenden, liefern eine sehr hohe metrische Genauigkeit der Verfolgung. Grund dafür ist die sehr große Basisbreite, durch diese können Mehrdeutigkeiten und Verdeckungen gehandhabt werden. Wenn man mögliche Genauigkeit und Robustheit der Algorithmen betrachtet, ist für das besagte Applikationsszenario dieser Arbeit ein Mehrkamerasysteme mit großer Basisbreite sehr sinnvoll. Ein Nachteil besteht aber in den hohen Investitionskosten sowie im erhöhten Hardware- und Kalibrieraufwand, im Vergleich zu einem System mit kleiner Basisbreite. Viele der Algorithmen benötigen sehr viel Rechenzeit, z.B. geben Rosenhahn u. a. (2007) fünf Minuten pro Zeitschritt an. Eine Echtzeitanwendung ist damit auf heutiger Standardhardware nicht ohne weiteres möglich. Ein großer Teil der Verfahren basiert auf der Annahme, dass ein personenspezifisches Körpermodell bekannt ist, dies ist in der angestrebten Applikation dieser Arbeit außerdem nicht möglich.

2.3 Merkmalsbasierte Handlungs- und Aktionserkennung

Bei einer engen Kooperation zwischen Mensch und Industrieroboter in der Produktion ist es notwendig, die ausgeführten Handlungen und Aktionen des Werkers zu erkennen, sodass der Roboter als Interaktionspartner darauf geeignet reagieren kann. Dieser Abschnitt soll einen Überblick über den aktuellen Stand der Forschung im Bereich der Erkennung und Unterscheidung von Aktionen, Handlungen und Handlungsabsichten geben.

Frühe Arbeiten zur Analyse menschlicher Bewegungen stammen aus der Psychologie. Grundlegend war hier die Arbeit von Johansson (1973). In dieser wurde gezeigt, dass menschliche Beobachter in der Lage sind, Bewegungsmuster anhand weniger sich bewegender Punkte zu erkennen. Dabei werden die Bewegungsmuster von schwarz bekleideten Personen vor einem schwarzen Hintergrund durchgeführt. Die Person ist nur durch wenige auf dem Körper angebrachte Lichtpunkte, den sogenannten *moving light displays* (MLDs), zu erkennen. Eine weitere Erkenntnis war, dass wenn die Versuchsperson keine Bewegungen ausführt, der Körper aus den ruhenden Lichtpunkten nicht zu erkennen ist. Die Arbeit von Johansson (1973) ist bis heute Grundlage vieler weiterer Verfahren und hat den Weg geebnet für eine mathe-

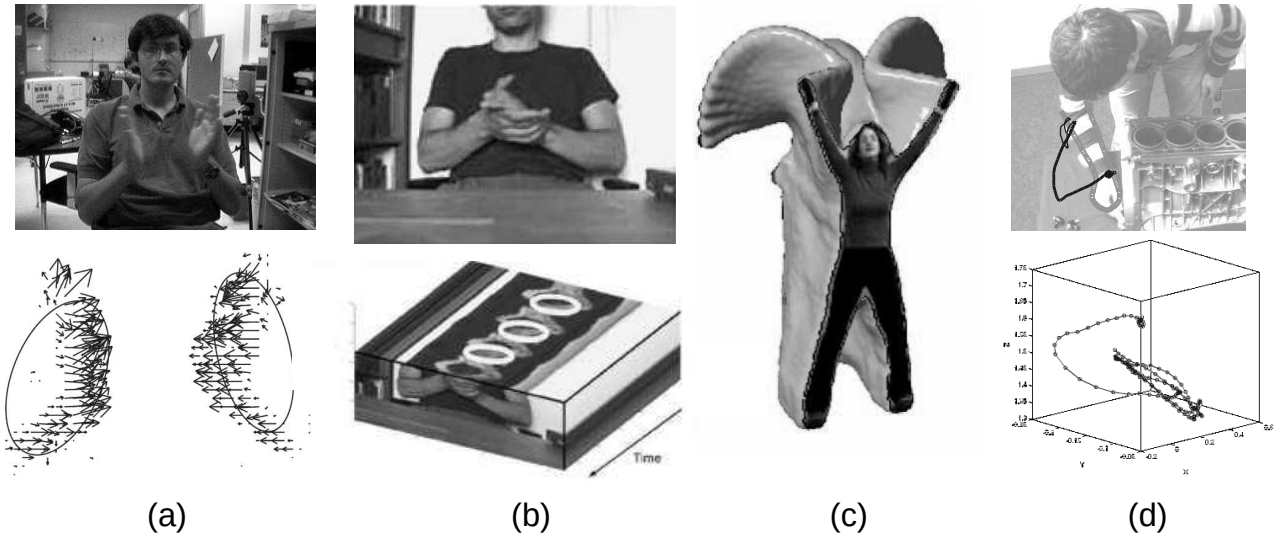

Abbildung 2.6: Raum-zeitliche Merkmale der vier Kategorien (a) optischer Fluss (Quelle: Cutler u. Turk (1998)), (b) raum-zeitliche Filter (Quelle: Laptev (2005)), (c) Vordergrund-segmentierte Bilder oder Bildsequenzen (Quelle: Gorelick u. a. (2007)) und (d) Bewegungstrajektorien.

matische Modellierung von Aktionen und Handlungen im Bereich der Bildverarbeitung und Mustererkennung.

Es gibt eine Vielzahl von Veröffentlichungen zur Handlungs- und Aktionserkennung, denn Anwendungsfelder gibt es z.B. bei Bewegungsanalysen in der Sportwissenschaft oder Rehabilitation, bei Überwachungsaufgaben, bei der Erkennung von Zeichensprache oder bei der Interaktion und Kommunikation von Mensch und Maschine. Detaillierte Zusammenfassungen über Veröffentlichungen, die eine Erkennung und Unterscheidung von Aktionen, Handlungen und Handlungsabsichten behandeln, werden in einigen Arbeiten (Cedras u. Shah, 1995; Pavlovic u. a., 1997; Bobick, 1997; Gavrila, 1999; Moeslund u. a., 2006; Mitra u. Acharya, 2007; Turaga u. a., 2008) gegeben. Im Folgenden werden einige wichtige Verfahren kurz vorgestellt. Zur besseren Übersicht werden die Veröffentlichungen nach den verwendeten raum-zeitlichen Merkmalen für die Aktionserkennung sortiert, siehe Systemmodell in Abbildung 2.1. Es gibt Ansätze die als Merkmale (i) optischen Fluss, (ii) raum-zeitliche Filter (iii) Vordergrund-segmentierte Bilder oder Bildsequenzen und (iv) Bewegungstrajektorien verwenden. Diese Kategorisierung wurde gewählt, da sich die vielen Veröffentlichungen dadurch gut einteilen lassen und eine Schlussfolgerung bezüglich der verwendeten Merkmale als sinnvoll erscheint. Außerdem wird diskutiert, ob die Merkmale den Anforderungen der angestrebten Applikation dieser Arbeit genügen, also trennbare Klassen im Merkmalsraum für verschiedene applikationsspezifische Bewegungen aufweisen.

Zur Erkennung, Unterscheidung und Klassifikation der raum-zeitlichen Merkmale wird für jede Klasse mindestens ein Beispiel oder Trainingsdatensatz benötigt. Das Modell oder auch Template einer Klasse wird üblicherweise aus mehreren Beispielen berechnet und später zur Klassifikation verwendet. Dabei werden sehr häufig Standardverfahren eingesetzt, nach Pavlovic u. a. (1997) und Turaga u. a. (2008) sind dies Hidden-Markov-Modelle (HMM), *Dynamic-Bayes-Nets* (DBN), Dynamische Programmierung (z.B. *Dynamic-Time-Warping* (DTW)),

Neuronale Netze, Partikelfilter und Clusteranalyse. Auf diese Standardverfahren wird nun kurz eingegangen und es werden auch Vor- und Nachteile genannt. Weiterführende Beschreibungen finden sich in den angegebenen Arbeiten.

Hidden-Markov-Modelle: Hidden-Markov-Modelle (HMMs) werden im Bereich der kontinuierlichen Sprach- und Schreibschrifterkennung seit vielen Jahren erfolgreich eingesetzt. HMMs sind stochastische Modelle, welche sich durch zwei Zufallsprozesse beschreiben lassen. Der erste Prozess ist eine Markov-Kette, ein endlicher stochastischer Automat, dessen Zustände von außen nicht direkt beobachtbar sind. Dieser Prozess genügt der Markov-Eigenschaft, die Wahrscheinlichkeit der zukünftigen Zustände hängt also nur vom aktuellen Zustand ab. Der zweite Zufallsprozess erzeugt gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung die für das HMM benötigten beobachtbaren Ausgangssymbole. Ein gute Einführung zur Theorie und Anwendung von HMMs wird von Rabiner (1990) gegeben. Ein weiterer guter Startpunkt ist die Arbeit von Fink (2007), welche sehr umfassend ist und auch auf praktische Systeme eingeht. Die Arbeit von Yamato u. a. (1992) war eine der Ersten, die HMMs erfolgreich zur kamerabasierten Aktionserkennung eingesetzt hat. Bis heute wurden HMMs und ihre unzähligen Erweiterungen in vielen Arbeiten zur Klassifikation von Gesten und Aktionen eingesetzt. Durch HMMs lassen sich aus kontinuierlichen Datenströmen Modelle erkennen und außerdem ist es möglich die zeitlich aufeinander folgenden Daten zu segmentieren. Dazu werden leider riesige Trainingsmengen an Daten zur Bestimmung der internen Wahrscheinlichkeiten des HMMs benötigt. In solch trainierten, stationären HMMs gibt es einige Schwachpunkte, die Modellierung der zeitlichen Dauer von Aktionen sowie die Behandlung von Bewegungspausen und unbekannten Bewegungen. Zur Bewältigung dieser Schwächen wurden in der Literatur, getrieben durch die Spracherkennung, verschiedene Erweiterungen von HMMs vorgestellt. Eine wichtige Erweiterung sind die Hidden-Semi-Markov-Modelle (HSMMs) (Ferguson, 1980), durch diese wird es möglich die zeitliche Dauer von Zuständen zu modellieren. Ein Hidden-State kann also viel länger andauern als ein anderer verborgener Zustand. Hongeng u. Nevatia (2003) verwenden HSMMs erfolgreich zur Event-Erkennung in Video-Streams. Basierend auf der Form und Bewegung von getrackten Objekten werden Events wie z.B, „ ein Auto vermeidet einen Kontrollpunkt“, „ ein Auto fährt in einen Kontrollpunkt“ oder „ ein Auto folgt einem anderen Auto“ erkannt. Duong u. a. (2005) führen Switching-HSMMs ein und verwenden diese zur Erkennung von Aktivitäten und abnormalem Verhalten bei Tätigkeiten des täglichen Lebens.

Dynamic-Bayes-Nets: Dynamic-Bayes-Nets (DBNs) generalisieren HMMs und sind in der Lage auch mehrere stochastische Input-Variablen einzubeziehen. Eine sehr gute und umfangreiche Einführung zur Theorie und Anwendung wird in der Dissertation von Murphy (2002) gegeben. DBNs verbinden die Wahrscheinlichkeits- und Graphentheorie miteinander und können somit auch sehr komplexe Zusammenhänge modellieren. Die Struktur des Netzes wird in vielen Arbeiten manuell vorgegeben, kann aber auch automatisch gelernt werden. Turaga u. a. (2008) machen die Aussage, dass sich DBNs besonders dann eignen, wenn komplexe Interaktionen zwischen mehreren Objekt modelliert werden müssen. Dies zeigt die Arbeit von Park (2004), in der Interaktionen von zwei sich bewegenden Personen erkannt werden. Ein Nachteil ist, dass zum automatischen Lernen der Wahr-

scheinlichkeiten und der Struktur des Netzes sehr viele Trainingsdaten vorhanden sein müssen.

Dynamische Programmierung: Dynamische Programmierung (DP) ist ein in der Informatik weit verbreiteter Ansatz zum Lösen von Optimierungsproblemen. Ein prominenter Vertreter von Verfahren die DP nutzen, ist das *Dynamic-Time-Warping* (DTW), das wie die HMMs in der kontinuierlichen Spracherkennung schon seit vielen Jahren erfolgreich eingesetzt wird. DTW ist in der Lage, Folgen von Werten gleicher oder unterschiedlicher Länge aufeinander abzubilden. Zur Erkennung von Gesten kann beispielsweise ermittelt werden, wie gut die Übereinstimmung zwischen einem Modell und einer Input-Trajektorie ist. Mit DTW können unterschiedlich schnelle Ausführungen einer Geste erkannt werden. Dies ist ein Vorteil, denn dadurch verringert sich die Anzahl der Trainingsbeispiele. Weitere Verfahren sind die Levenshtein-Distanz (Levenshtein, 1966) oder die *Longest-Common-Subsequence* (LCSS) Metrik (Hirschberg, 1977). Vlachos u. a. (2002) erweitern die LCSS-Metrik zur Erkennung von Trajektorien australischer Zeichensprache.

Neuronale Netze: Künstliche Neuronale Netze (engl. *artifical neural networks*) fassen eine sehr große Gruppe von Klassifikationsverfahren zusammen. Die unzähligen Varianten von Neuronalen Netzen werden in den unterschiedlichsten Bereichen zur Klassifikation eingesetzt. Übersichten über die Thematik und Anwendungsgebiete werden von Niemann (1983) und Schürmann (1996) gegeben. Neuronale Netze sind universelle Funktionsapproximatoren. Biologisch inspiriert werden Netzwerke künstlicher Neuronen aufgebaut, welche Eingaben von anderen Neuronen gewichtet aufsummieren. Über nichtlineare Funktionen wird die Ausgabe der Neuronen berechnet. Das Training eines Netzes besteht darin die Gewichtsparameter der Neuronen zu lernen. Häufig werden dazu viele Beispieldaten benötigt. Eine zeitliche Erweiterung der Neuronalen Netze, die *Time-Delay-Neural-Networks* (TDNNs), werden von Waibel u. a. (1990) zur Spracherkennung eingesetzt. Yang u. Ahuja (1999) erkennen mit einem TDNNs amerikanische Zeichensprache und Wöhler u. Anlauf (1999) verwenden TDNNs zur Bildsequenzanalyse. Zu dieser Kategorie können auch statistische Klassifikationsverfahren wie der Polynomklassifikator (Schürmann, 1996) gezählt werden.

Partikelfilter: Einen etwas moderneren Ansatz bilden die Sequenziellen Monte-Carlo-Methoden (Doucet u. a., 2001), ein Verfahren zur probabilistischen Zustandsschätzung. Doucet u. a. (2001) geben einen detaillierten Überblick über die unterschiedlichen Partikelfilter-Varianten und beschreiben die Grundlagen sowie Anwendungsmöglichkeiten. Ein sehr weit verbreiteter Ansatz ist der CONDENSATION-Algorithmus (Isard u. Blake, 1998). Diese Variante wird von Black u. Jepson (1998) zur trajektorienbasierten Gestenerkennung verwendet. Ein Vorteil des Verfahrens liegt in der automatischen Konzentration auf relevante Teile und der geringen Anzahl an Trainingsbeispielen. Li u. a. (2006) schätzen den Zustand von HMMs mit einem Partikelfilter.

Clusteranalyse: Bei einer Clusteranalyse werden Gruppen (Cluster) von Objekten, welche eine irgendwie geartete Ähnlichkeit aufweisen, ermittelt. Ziel ist es, die Cluster so zu wählen, dass sich die Objekte eines Clusters untereinander möglichst ähnlich sind (geringe Distanz), während sich die Objekte aus verschiedenen Clustern möglichst deutlich unterscheiden (große Distanz). Bei der Klassifikation wird ein unbekanntes Objekt, basierend

auf dem Abstand zu den Clustern, zu einer der zuvor definierten Klassen zugeordnet. Die Arbeit von Jain u. a. (1999) gibt einen umfangreichen Überblick über Verfahren und Applikationen. Dollar u. a. (2005) extrahieren Aktionsprototypen mittels eines K-Means-Clustering und verwenden diese zur Klassifikation. Ein Vorteil ist, dass man mit einer geringen Anzahl an Trainingsbeispielen robuste Erkennungsleistungen erzielen kann. Ein weiterer Vorteil besteht in der Möglichkeit des Online-Lernens.

2.3.1 Optischer Fluss als Merkmal

Cutler u. Turk (1998) verwenden als Merkmal zur Handlungserkennung optischen Fluss, ein Beispiel dafür ist in Abbildung 2.6 (a) zu sehen. Durch einen Algorithmus, basierend auf einfachen Regeln werden Handlungen wie „Winken“, „Klatschen“ oder „Marschieren“ erkannt und unterschieden.

Zhu u. a. (2006) erkennen Aktionen von Tennisspielern in Fernsehübertragungen. Das Szenario ist sehr schwierig, da die Bildauflösung gering ist und die Tennisspieler weit entfernt sind. Mit einem Partikelfilter-Ansatz werden die Tennisspieler in den Bildern über die Zeit verfolgt und der optische Fluss in einer Box um den getrackten Spieler berechnet. Zur Aktionserkennung wird der horizontale und vertikale Fluss jeweils in ein Histogramm sortiert und diese Histogramme mit einem SVM-Ansatz klassifiziert.

Wertung: Der optische Fluss ist ein einfaches und effizient zu berechnendes Merkmal zur Aktionserkennung. Eine robuste Unterscheidung einfacher Aktionen wie „Klatschen“, „Winken“, „Springen“ oder „Marschieren“ ist auch bei geringer Bildauflösung möglich. Positiv ist außerdem, dass nur wenige Prototypen zur Aktionserkennung notwendig sind, was eine Voraussetzung der Applikation in dieser Arbeit ist. Eine Limitierung dieses Merkmals stellen komplexe Bewegungen dar, welche bei einer Interaktion von Mensch und Industrieroboter auftreten und mit einem solch einfachen Merkmal nicht zu unterscheiden sind. Die betrachteten Verfahren sind außerdem nicht unabhängig von der Kameraposition, was dazu führt, dass das Erkennungssystem bei einer Änderung in der überwachten Interaktionszelle neu trainiert werden muss.

2.3.2 Raum-zeitliche Filter als Merkmal

In dieser Kategorie werden die zur Erkennung und Unterscheidung von Aktionen verwendeten aktionsspezifischen Merkmale durch Faltung einer Bildsequenz mit einer definierten Filterbank generiert, ein Beispiel findet sich in Abbildung 2.6 (b).

Zhong u. a. (2004) wenden auf eine Sequenz von Bildern räumlich einen Gaußfilter und auf der zeitlichen Achse dessen Ableitung an. Dabei zeigen die Filterantworten hohe Werte bei Bewegungen und werden in ein räumliches 2D-Histogramm für eine Sequenz aggregiert. Mit Hilfe der Histogramme werden Prototypen berechnet, welche später zur Erkennung eingesetzt werden. Die Nächste-Nachbar-Suche basiert dabei auf der Chi-Quadrat-Metrik. Es zeigt sich, dass die verwendeten Merkmale eine effektive und robuste Erkennungsleistung im Fernbereich ermöglichen.

Eine zeitliche Erweiterung des Harris-Ecken-Detektors wird von Laptev (2005) als aktionsspezifisches Merkmal genutzt. Der entwickelte Filter ist ein Interest-Operator für signifikante raum-zeitliche Änderungen in den Bildsequenzen. Zur Erkennung von Aktionen wird eine Bildsequenz unter verschiedenen räumlichen und zeitlichen Skalierungen mit dem Filter gefaltet. Dies ermöglicht eine räumliche und zeitliche Invarianz beim Matching mit zuvor extrahierten Prototypen. Mit dem Ansatz ist Laptev (2005) in der Lage auch bei geringer Auflösung und dynamischen Hintergründen spazierende Personen zu erkennen.

Dollar u. a. (2005) extrahieren markante periodische raum-zeitliche Bereiche, indem auf eine Bildsequenz räumlich ein Gaußfilter und zeitlich ein Gaborfilter angewandt wird. Es wird gezeigt, dass diese schnellen und einfach zu implementierenden Filter sehr nützlich für Szenarios mit geringer Bildauflösung oder schlechter Bildqualität sind, da es hier sehr schwierig ist Objekte genau zu verfolgen, optischen Fluss zu berechnen oder Silhouetten zu extrahieren. Die Erkennung basiert auf Aktionsprototypen, welche mittels eines K-Means-Clustering extrahiert werden. Nur einfache Aktionen wie „Laufen“, „Rennen“, „Boxen“, „Winken“ oder „Klatschen“ sind unterscheidbar.

Wertung: Merkmale, welche durch raum-zeitliche Filter extrahiert werden, sind diskriminativer als optischer Fluss, da die Filterantworten auf einer Bildsequenz berechnet werden und auch bei sehr schlechten Auflösungen verlässliche Merkmale liefern. Besonders für Szenarien mit geringer Bildauflösung oder schlechter Bildqualität wurden raum-zeitlichen Filter entwickelt, da hier ein robustes Tracking von Objekten oder Objektteilen schwer zu realisieren ist. Wie beim optischen Fluss sind nur einfache Aktionsklassen unterscheidbar, daher scheinen raum-zeitliche Filter für die Applikation dieser Arbeit nicht das geeignete Merkmal zu sein.

2.3.3 Vordergrund-segmentierte Bilder oder Bildsequenzen als Merkmal

In vielen Arbeiten werden Vordergrund-segmentierte Bilder oder Bildsequenzen als Merkmale verwendet. Durch Segmentierung in Hinter- und Vordergrund werden binäre Bilder oder Silhouetten von bewegten Objekten extrahiert. Bei der Anwendung auf Bildsequenzen können die Vordergrund-segmentierten Bilder in ein 2D-Bild zusammengefasst werden oder volumetrisch in einer raum-zeitlichen Repräsentation gespeichert werden. Eine Beispiel für eine solche volumetrische Repräsentation zeigt Abbildung 2.6 (c).

Yamato u. a. (1992) erzeugen durch Hintergrundsubtraktion Silhouetten von bewegten Objekten und wenden darauf eine Vektorquantisierung an. Mit einem Framework, bestehend aus HMMs, werden einfache Aktionen von Tennisspielern erkannt und unterschieden.

Bobick u. Davis (2001) verwenden zeitliche 2D-Templates als Repräsentation für Aktionen. Nach einer Segmentierung in Hinter- und Vordergrund werden binäre Bilder von bewegten Objekten in ein Bild aggregiert. Die Aggregation wird dabei auf zwei verschiedene Arten berechnet. Bei den sogenannten *Motion-Energy-Images* (MEI) werden alle binären Bilder einer Sequenz mit dem gleichen Gewicht in das resultierende Bild zusammengefasst. Die zweite Möglichkeit besteht in der Nutzung der *Motion-History-Images* (MHI), bei denen das Gewicht mit der Zeit ansteigt. MEI und MHI bilden zusammen das Template zur Repräsentation der Akti-

on. Zur Erkennung und Unterscheidung der Aktionen werden aus den Templates Hu-Momente extrahiert, welche invariant gegenüber Translation, Rotation und Größenskalierung sind. Es wird gezeigt, dass MEI und MHI geeignet sind, einfache Aktionen wie „Hinsetzen“, „Beugen“ oder „Hocken“ zu unterscheiden. Weitere Vorteile sind der geringe Berechnungsaufwand und die Möglichkeit der Anwendung bei schlechter Bildqualität. Von Bobick (1997) wird angemerkt, dass MEI und MHI ihre Unterscheidungsfähigkeit bei komplexen Aktionen verlieren, da durch die einfache Repräsentation die Bewegungshistorie überschrieben werden kann. Weitere Probleme sind bei Helligkeitsschwankungen, Verdeckungen von Teilen der Person oder durch kombinierte Bewegungen zu erwarten.

Li u. Greenspan (2005) benutzen das DTW-Verfahren auf Sequenzen von Silhouetten, um Armbewegungen, wie Zeigegesten oder Winkbewegungen, zu erkennen. Da die Bewegungen vor statischen, unstrukturierten Hintergründen durchgeführt werden, sind die Bewegungen leicht zu detektieren. DTW ermöglicht, dass auch bei einer unterschiedlich schnellen bzw. weiten Ausführung einer Geste die Ähnlichkeit mit einem zuvor generierten Modell hergestellt werden kann. Für die Bildung der Modelle werden viele Trainingssequenzen mit unterschiedlich schneller Ausführung der Gesten benötigt.

Yilmaz u. Shah (2005) repräsentierten die zu erkennenden Aktionen mit volumetrischen Templates in einem raum-zeitlichen Koordinatensystem. Diese Templates werden berechnet, indem die Silhouetten von bewegten Objekten in Einzelbildern extrahiert und gemäß ihres Zeitstempels hintereinander gespeichert werden. Durch diese Repräsentation ist man in der Lage, neben der Form auch die Bewegung des Objekts zu beschreiben und verhindert ein überschreiben, wie es bei Bobick (1997) möglich ist. Zur Erkennung der Aktionen werden aus der volumetrischen Repräsentation aktionsspezifische geometrische Deskriptoren extrahiert. In der Arbeit wird theoretisch herausgearbeitet, dass eine von der Blickrichtung unabhängige Erkennung möglich ist.

Die von Gorelick u. a. (2007) verwendete volumetrische Repräsentation von Aktionen entsteht durch hintereinander angeordnete binäre Bilder von bewegten Objekten (Abbildung 2.6(c)). Aus der volumetrischen Repräsentation werden für jede Aktion Deskriptoren durch Lösung der Poisson-Gleichung extrahiert und zur Erkennung verwendet. Dieser Ansatz erfordert eine präzise Segmentierung in Vorder- und Hintergrund und eine fixe Position der Kamera. Wiederum werden nur einfache Aktionen unterschieden.

Wertung: Vordergrund-segmentierte Bilder oder Bildsequenzen als Merkmal stellen eine diskriminativere Repräsentation als optischer Fluss oder raum-zeitliche Filter dar. Dies wird durch die aufwendige, häufig volumetrische und nicht mehr so kompakte Repräsentation erreicht. Es wird eine teilweise Invarianz gegenüber der Objekt- oder Kameraposition erzielt, was eine Forderung der Applikation dieser Arbeit ist. Grundlage des Merkmals ist eine stabile und robuste Hintergrundsegmentierung. Die erwähnten Ansätze sind in der Lage, einfache Handlungen zu erkennen. Dies ist auch bei geringer Bildauflösung möglich, bei der ein robustes Tracking von Objekten oder Objektteilen schwer zu realisieren wäre, z.B. bei Fernsehübertragungen von Ballettaufführungen oder Schwimmwettkämpfen. Für die Applikation dieser Arbeit scheint dieses Merkmal nicht geeignet zu sein, da die zu erkennenden Aktionen sehr komplex sind und es schwer ist, Kontext mit einzubeziehen.

2.3.4 Bewegungstrajektorien als Merkmal

Sehr viele Arbeiten verwenden als aktionsspezifisches Merkmal Trajektorien von Punkten oder höherdimensionalen Körperkonfigurationen. Trajektorien sind eine kompakte und sehr diskriminative Repräsentation von Aktionen. Es bieten sich Möglichkeiten, die über einfache Bildsequenzmerkmale hinausgehen, denn zeitliche Aspekte von Bewegungen werden gut repräsentiert und auch Beziehungen zwischen einzelnen Objekten können repräsentiert werden. Voraussetzung für trajektorienbasierte Merkmale ist immer eine robuste und zeitlich stabile Objektverfolgung. Bewegungstrajektorien können grob in drei Arten unterteilt werden, die sich aus dem erzeugenden Trackingsystem ableiten. Es werden (i) 2D-Trajektorien, (ii) 3D-Trajektorien und (iii) Trajektorien hochdimensionaler Körperkonfigurationen unterschieden. 2D-Trajektorien entstehen durch Trackingverfahren, welche zweidimensionale Bildpositionen ausgeben. Bei 3D-Trajektorien gibt das Verfolgungssystem entsprechend 3D-Positionen aus und hochdimensionale Körperkonfigurationen entstehen durch Verfahren, die beispielsweise den gesamten Körper modellbasiert verfolgen.

2D-Trajektorien

Die Arbeit von Black u. Jepson (1998) ist ein sehr bekannter und häufig zitierter Ansatz zur Handlungserkennung mit Trajektorien-Merkmalen. Der CONDENSATION-Algorithmus (Isard u. Blake, 1998) wird zum *Condensation-based Trajectory Recognition* (CTR) Verfahren erweitert. Der Ansatz kann als Generalisierung von HMMs angesehen werden. Durch CONDENSATION wird der Zustand in einem graphischen Modell mit einer diskreten Menge von Zuständen und probabilistischen Zustandsübergängen geschätzt. Für jeden Zustand gibt es eine 2D-Referenztrajektorie, welche manuell aus zuvor aufgezeichneten Beispielen generiert wird. Das Partikelfilter schätzt zu jedem Zeitschritt den Zustand des grafischen Modells, die Phase innerhalb der zugehörigen Referenztrajektorie und zwei Parameter, welche eine Deformierung der Referenztrajektorie modellieren. Die Partikelgewichte ergeben sich, indem in einem Sliding-Window ein Template-Matching der aktuellen Input-Trajektorie mit der durch den Partikelzustand beschriebenen Referenztrajektorie durchgeführt wird. Um eine gewisse Invarianz gegenüber Translation und Rotation zu erreichen, werden nicht die Punkttrajektorien, sondern die daraus abgeleiteten Geschwindigkeitstrajektorien miteinander verglichen. Mit dem System werden Gesten zur Steuerung eines Computers erkannt.

Fritsch u. a. (2004) erweitern das CTR-Verfahren um Kontextinformation. Die menschliche Hand wird mit einem 2D-Tracker in Bildfolgen verfolgt. Das Tracking basiert auf einer Hautfarbsegmentierung. Außerdem werden zusätzliche Informationen bezüglich des Vorhandenseins sowie die räumliche Entfernung zu assoziierten Objekten verwendet. Die Objekte werden dabei einzelnen Aktionen zugeordnet und in den Kamerabildern erkannt. Die Kontextinformation wird in die Gewichtsberechnung im CTR-Verfahren einbezogen. Der entwickelte Ansatz wurde von Haasch u. a. (2005) in ein System zur Interaktion mit einem mobilen Roboter integriert.

Yang u. Ahuja (1999) erkennen mit einem TDNN amerikanische Zeichensprache. Zur Erkennung werden 2D-Trajektorien der Hände als Merkmale verwendet.

Vlachos u. a. (2002) verwenden die *Longest-Common-Subsequence* (LCSS) Metrik, ein Verfahren der Dynamischen Programmierung, zur Nächsten-Nachbar-Suche. An eine Datenbank

aus aufgezeichneten und segmentierten Trajektorien australischer Zeichensprache[3] werden Suchanfragen beantwortet, indem ein Suchindex aufgebaut wird.

Li u. a. (2006) verfolgen zweidimensional die Hände eines Menschen in Bildfolgen mit einem Trackingsystem. Manipulative Gesten wie „Tee zubereiten“ oder „Tasse greifen“ werden mit HMMs erkannt. Die 2D-Input-Trajektorien werden segmentiert, indem nur Trajektorien in der Nähe von erkannten Objekten zur Klassifikation verwendet werden. Jede manipulative Geste wird dabei als HMM mit zugehörigen Referenztrajektorien modelliert. Das Besondere an dem Verfahren ist, dass der Zustand in den HMMs mit einem Partikelfilter, ähnlich dem CTR-Verfahren (Black u. Jepson, 1998), geschätzt wird. Wie Fritsch u. a. (2004) bezieht Li u. a. (2006) den Objektkontext mit ein. Die HMMs werden aus einem kleinen Lernset abgeleitet.

Bashir u. a. (2007) stellen ein Verfahren vor, welches eine Nächste-Nachbar-Suche in einer Datenbank aus aufgezeichneten Trajektorien australischer Zeichensprache realisiert. Die Anfragen werden dabei auf dem selben Datensatz wie in der Arbeit von Vlachos u. a. (2002) gestellt. Die Trajektorien werden an Stellen maximaler Krümmung in Sub-Trajektorien getrennt und diese danach mit einer Hauptkomponentenanalyse normalisiert. Mit den Sub-Trajektorien eines Testsets wird durch Clusteranalyse ein Alphabet erzeugt. Als Ähnlichkeitsmaß werden String-Metriken auf diesem Alphabet verwendet. Eine Anfrage an die Datenbank besteht also aus den Schritten (a) Sub-Trajektorien erzeugen, (b) Sub-Trajektorien normalisieren, (c) Sub-Trajektorien dem Alphabet zuordnen und (d) das erzeugte Wort mit den in der Datenbank vorhandenen Wörtern abgleichen.

Wertung: 2D-Trajektorien stellen ein kompaktes und diskriminativeres Merkmal als die bisher vorgestellten Merkmale dar. Dies ist aus der Tatsache abzuleiten, dass sie z.B. zur Erkennung von komplexer Zeichensprache bereits erfolgreich eingesetzt werden. Eine Grundvoraussetzung ist ein stabiles, genaues und zeitlich robustes Objekttracking. Daher werden 2D-Trajektorien fast ausschließlich zur Erkennung im Nahbereich eingesetzt, weil hier eine robuste Verfolgung gut realisierbar ist. 2D-Trajektorien als Merkmal eignen sich gut, um den Objektkontext einzubeziehen, sind aber sensitiv gegenüber Translation, Rotation und Skalierungsänderungen, was einen Normalisierungsschritt erfordert. Es können Mehrdeutigkeiten auftreten, z.B. können zwei gleiche Aktionen in unterschiedlicher Tiefe nicht unterschieden werden oder zwei verschiedenartige Aktionen in unterschiedlicher Tiefe gleich aussehen. In der Applikation dieser Arbeit wäre dies z.B. ein Schraubvorgang an unterschiedlichen Positionen, auf den der Roboter verschiedenartig reagieren muss.

3D-Trajektorien

Campbell u. a. (1996) verwenden Bewegungstrajektorien von Kopf und Händen, um T´ai-Chi Bewegungen zu erkennen. Der Kopf und die Hände werden dabei mit einem Regionen-Tracker in Bildern verfolgt, welcher auf 3D-Stereodaten basiert. Zur Erkennung der T´ai-Chi Bewegungen werden HMMs verwendet. In der Arbeit werden verschiedene, aus Trajektorien abgeleitete, Merkmale zur Klassifikation miteinander verglichen. Es wird gezeigt, dass aus den 3D-Positionstrajektorien abgeleitete translations- und rotationsinvariante Merkmale die

[3]Der Datensatz ist unter http://kdd.ics.uci.edu/ verfügbar

besten Ergebnisse liefern. Eine weitere Erkenntnis der Arbeit ist, dass die Kopfposition zur Erkennung von T´ai-Chi Bewegungen nicht notwendig ist.

In der Arbeit von Vogler u. Metaxas (1998) wird der menschliche Arm mit von den drei orthogonal angeordneten Kameras gelieferten Bildern verfolgt. Das Tracking basiert dabei auf einem modellbasierten 3D-Ansatz und die geschätzten 3D-Bewegungsdaten werden zur Erkennung von amerikanischer Zeichensprache verwendet. Das Vokabular umfasst 53 Zeichen und das zur Erkennung verwendete HMM-Framework wird mit sehr vielen Trainingsdaten gelernt. Die Trajektorien werden an Stellen starker Geschwindigkeitsänderungen segmentiert. Die so extrahierten Sub-Trajektorien sind Input für die verschiedenen HMMs. In der Arbeit wird gezeigt, dass 3D-Bewegungsdaten als Merkmale bessere Ergebnisse liefern als 2D-Trajektorien.

Keskin u. a. (2003) stellen ein technisches System zur Steuerung eines Zeichenprogramms mit einem farbigen Handschuh vor. Der Handschuh wird mit einem Regionen-Tracker basierend auf 3D-Stereodaten verfolgt. Als aktionsspezifisches Merkmal wird nicht die 3D-Bewegungstrajektorie, sondern die daraus berechnete Geschwindigkeitstrajektorie verwendet. Zur Glättung der Trajektorien wird ein Kalman-Filter eingesetzt. Die Erkennung basiert auf einem HMM-Framework. Wiederum werden sehr viele Trainingssequenzen benötigt.

Nickel u. a. (2004) sind in der Lage, mit einer Stereokamera menschliche Zeigegesten zu erkennen und die Zeigerichtung im Raum zu bestimmen. Ein Neuronales Netz wird verwendet, um die Orientierung des Kopfes zu ermitteln, da der Menschen beim Zeigen sehr häufig in die Zeigerichtung schaut. Die Zeigegesten werden mit einem HMM-Framework erkannt, als Input wird der Abstand der Hand zum Körper, die absolute Richtungsgeschwindigkeit und die Höhenänderung der Hand verwendet. Um translations- und rotationsinvariante Merkmale zu verwenden, wird die Hand in einem zylindrischen Koordinatensystem, dessen Basis in der Kopfposition des Benutzers liegt, repräsentiert.

Croitoru u. a. (2005) stellen ein Verfahren zur Berechnung der Ähnlichkeit von 3D-Trajektorien vor. Wie bei Vlachos u. a. (2002) und Bashir u. a. (2007) stammen die Daten aus einer Datenbank aus aufgezeichneten Trajektorien australischer Zeichensprache. Im Gegensatz zu Vlachos u. a. (2002) und Bashir u. a. (2007) verwenden Croitoru u. a. (2005) die volle 3D-Information. Durch ein Verfahren, welches auf der Gravitationsfeldstärke basiert, werden die Trajektorien normalisiert und die LCSS-Metrik zur Berechnung der Ähnlichkeit verwendet. Das Verfahren wird evaluiert, indem eine Clusteranalyse der kompletten Datenbank gemacht wird und die Zuordnungsfehler gezählt werden. Ein Nachteil ist, dass das Verfahren von bereits segmentierten Trajektorien für jede der Gesten ausgeht.

Wertung: 3D-Trajektorien als aktionsspezifisches Merkmal zur Erkennung sind eine sehr diskriminative Repräsentation der Daten. Es kann sehr leicht Kontexinformation bzgl. des Abstands zu realen Objekten mit einbezogen werden. Grundvoraussetzung ist wiederum ein stabiles, genaues und zeitlich robustes 3D-Objekttracking. Durch die 3D-Repräsentation ist eine Skalierungsinvarianz gegeben. Um translations- und rotationsinvariante Merkmale zu extrahieren muss aber trotzdem eine Normalisierung durchgeführt werden. Ein weiterer Vorteil ist, dass bei einer Änderung der Kameraposition das Erkennungssystem nicht neu belernt werden muss. Mehrdeutigkeiten wie bei 2D-Trajektorien treten nicht auf, d.h. ein Schraubvorgang an unterschiedlichen Raumpositionen kann als solcher erkannt werden.

Trajektorien von Körperkonfigurationen

Grundlage für diese Merkmalskategorie ist ein modellbasiertes Körpertracking. Wie im Abschnitt 2.2 beschrieben, gibt es viele Arbeiten, die modellbasiert den Oberkörper oder den gesamten Körper des Menschen robust und zeitlich stabil verfolgen. Es fällt aber auf, dass diese Systeme eher zur Ersetzung von Marken-basierten Trackingsystemen gedacht sind als zur Erstellung von Eingabedaten für ein Aktionserkennungssystem. Dabei hätte ein System, welches als aktionsspezifische Merkmale Trajektorien von Körperkonfigurationen verwendet, einige Vorteile. Beispielsweise besteht eine Invarianz bezüglich Translation und Rotation, wenn man als Input-Daten die Gelenkwinkel der Arme verwenden würde, da deren Bewegungen immer relativ zum Torso sind. Einige wenige Arbeiten machen sich diese Eigenschaften zu Nutzen.

Park (2004) erkennt Interaktionen von zwei sich bewegenden Personen mit DBNs. Das entwickelte System besteht aus drei Ebenen. In der ersten Ebene werden die Körperteile der Interaktionspartner in den Bildern mit einem Bayes-Netz detektiert. Die zweite Ebene modelliert die Aktionen eines einzelnen Interaktionspartners mit DBNs. In der höchsten Ebene werden die Ergebnisse der unterliegenden Ebenen genutzt, um die Interaktionen zu erkennen.

Hofemann (2007) erweitert in seiner Dissertation die Arbeiten von Black u. Jepson (1998) und Fritsch u. a. (2004) durch die Nutzung von 3D-Information. Das System wurde für die Erkennung von Zeigegesten konzipiert und nutzt das modellbasierte Trackingsystem von Schmidt u. a. (2006) zur Generierung der Bewegungsdaten. Die eingesetzten aktionsspezifischen Merkmale sind der Abstand der Hand zum Körperpunkt in der horizontalen Ebene sowie der Winkel der Hand. Die Merkmale können absolut verwendet werden oder auch ihre erste Ableitung. Dabei werden beide Merkmale in einem schulterzentrierten Zylinder-Koordinatensystem repräsentiert, was den Vorteil hat, dass Invarianz gegenüber Translationen und Rotationen erreicht wird. Wie bei Fritsch u. a. (2004) wird situativer und räumlicher Kontext eingesetzt. Im Gegensatz zu Black u. Jepson (1998) beschreibt Hofemann (2007) eine Methode zum Generieren neuer Modelle (Referenztrajektorien). Ein Vorteil dieses Ansatzes ist, dass die genutzten Referenztrajektorien nicht mehr manuell aus den Beispieldaten erzeugt werden müssen. Schmidt u. a. (2008) integrieren den Ansatz von Hofemann (2007) in ein System zur Interaktion mit einem mobilen Roboter und evaluieren das System auf einem großen Datensatz. Es wird gezeigt, dass durch das monokulare 3D-Tracking (Schmidt u. a., 2006) die Ergebnisse der Aktionserkennung im Vergleich zur vorherigen Version (Haasch u. a., 2005) signifikant verbessert werden. Der Grund dürfte in der Beseitigung von Mehrdeutigkeiten liegen. Dies lässt die Vermutung zu, dass der Einsatz mehrerer Kameras die Erkennungsleistung des Systems weiter steigern kann.

Gu u. a. (2008) setzen als aktionsspezifische Merkmale Trajektorien von Gelenkwinkeln ein. Mit einem modellbasierten Tracking werden die Testpersonen in Bildern verfolgt und mit einem HMM-Framework werden Aktionen wie „Kicken“, „Zeigen“, „Umdrehen“ oder „Auf die Uhr schauen“ erkannt.

Wertung: Trajektorien von Körperkonfigurationen als aktionsspezifisches Merkmal zur Erkennung sind eine sehr diskriminative Repräsentation der Daten. Es kann sehr leicht Kontexinformation bzgl. des Abstands zu realen Objekten mit einbezogen werden. Grundvoraussetzung

ist ein modellbasiertes Objekttracking, welches genau und zeitlich robust ist. Durch die Repräsentation ist eine Invarianz bzgl. Translation und Rotation gegeben, da alle Bewegungen relativ zum Torso ausgeführt werden und es egal ist, wo sich der Mensch im Raum befindet. Bei einem 3D-Ansatz ist auch eine Invarianz bzgl. der Skalierung der Bewegungen gegeben. Ein großer Vorteil dieser Invarianzen ist, dass bei einer Änderung der Kameraposition oder der Objektposition das Erkennungssystem nicht neu trainiert werden muss.

2.4 Zusammenfassung der Wertungen

Abgeleitet vom allgemeinen Modell eines technischen Systems zur Realisierung des Interaktionsprozesses (Abbildung 2.1) soll nun zusammenfassend eine Empfehlung des Autors für einen möglichen Sensor, das Trackingsystem, die aktionsspezifischen Merkmale und das Aktionserkennungssystem gegeben werden. Aufgrund der angestrebten Applikation in dieser Arbeit ist die Nutzung eines Kamerasensors mit drei zueinander senkrecht angeordneten Kameras sinnvoll. Dieser Sensor sollte dem bereits zertifizierten kamerabasierten Überwachungssystem SafetyEYE (Winkler, 2006) ähnlich sein. Hinsichtlich der Genauigkeit ist ein solches System einem monokularen Kamerasystem überlegen und die Investitionskosten sowie der Hardware- und Kalibrieraufwand sind noch überschaubar. Ein weiterer Vorteil besteht in der Möglichkeit, einen großen Teil der robusten und zeitlich stabilen monokularen Trackingverfahren auf ein Kamerasystem mit kleiner Basisbreite zu erweitern. Zur Detektion, Segmentierung und zum Tracking scheinen modellbasierte 3D-Verfahren oder Stereo-Ansätze vernünftig, da die zu erzielende Genauigkeit gut ist und bei einfachen, analytischen Modellen (Abschnitt 2.1.3) die Aussicht auf Realzeitfähigkeit besteht. Aufgrund der kompakten und diskriminativen Repräsentation sowie der möglichen Invarianz bzgl. Skalierung, Translation und Rotation scheinen 3D-Trajektorien oder Trajektorien von Körperkonfigurationen das geeignete aktionsspezifische Merkmal zu sein. Die Nutzung von Kontextwissen wäre für die angestrebte Applikation dieser Arbeit außerdem wichtig. Bei der Aktionserkennung wäre es vorteilhaft, ein System wie das von Li u. a. (2006) einzusetzen, da die Kombination von HMMs mit einem Partikelfilter die Vorteile beider Verfahren vereint. Damit wäre es möglich, Wissen bzgl. der Reihenfolge der ausgeführten Aktionen sowie Kontextwissen aus der Szene in die Erkennung einzubeziehen. Ein weiterer Vorteil ist, dass keine großen Datensätze von Trainingsbeispielen erforderlich sind.

Teil II

3D-Trackingsysteme

In der Einleitung dieser Dissertation (Kapitel 1) werden die beiden Schwerpunktthemen zur Realisierung der Vision „Sichere Mensch-Industrieroboter-Interaktion“ mit einem intelligenten technischen Überwachungssystem beschrieben. Der erste Schwerpunkt eines solchen technischen Systems beinhaltet die Segmentierung, Verfolgung und Bewegungsvorhersage von beobachteten Objekten in den Bildern eines Kamerasystems. Im Folgenden werden vier in dieser Arbeit entwickelte Systeme detailliert vorgestellt, welche eine Realisierung des ersten Schwerpunktthemas darstellen. Anschließend werden die Ergebnisse der Trackingsysteme auf verschiedenen realen Testsequenzen qualitativ und quantitativ evaluiert.

In der Beschreibung aller technischen Trackingsysteme dieser Arbeit wird die Notation von Craig (1989) verwendet. Mit ${}^{C}\mathbf{x}$ wird ein Punkt $\mathbf{x}$ im Koordinatensystem C bezeichnet. Eine ähnliche Notation wird für die Transformations- und Projektionsmatrizen gewählt, die Matrix ${}^{C}_{W}\mathbf{H}$ transformiert einen Punkt im Koordinatensystem W in das Koordinatensystem C: ${}^{C}\mathbf{x} = {}^{C}_{W}\mathbf{H}\ {}^{W}\mathbf{x}$. Außerdem werden Vektoren klein und fett und Matrizen groß und fett geschrieben.

Mit $\mathbf{\Phi}$ wird immer die geschätzte 3D-Pose, also die Lage und Orientierung im Raum, eines der in dieser Arbeit verwendeten Modelle beschrieben. Dabei steht $\mathbf{\Phi}$ für einen Vektor, welcher die 3D-Pose-Parameter eines definierten Modells enthält. Eine 3D-Pose $\mathbf{\Phi}(t)$ zum Zeitschritt t ist in dieser Arbeit immer das Ergebnis der Optimierung einer Zielfunktion. Dabei beschreibt die Zielfunktion wie gut die optimierten Modellparameter $\mathbf{\Phi}(t)$ zu den Daten zum Zeitschritt t passen.

Kapitel 3

Verfolgung von 3D-Konturen

In diesem Kapitel wird ein modellbasiertes Verfahren zur Verfolgung von 3D-Konturen in den Bildern eines Mehrkamerasystems mit geringer Basisbreite vorgestellt. Mit dem entwickelten technischen System lassen sich beliebige menschliche Körperteile über die Zeit kamerabasiert verfolgen (engl. *tracken*). Wichtig ist dabei, dass das getrackte menschliche Körperteil durch ein 3D-Konturmodell beschrieben werden kann. Die 3D-Konturanpassung wird mit Hilfe einer multiokularen Erweiterung des Contracting-Curve-Density (CCD) Algorithmus (Hanek, 2004) durchgeführt. Die Erweiterung zum multiokularen CCD-Algorithmus (MOCCD) ist die wichtigste Systemkomponente und wird in diesem Kapitel ausführlich beschrieben. Das Systemmodell des modellbasierten Ansatzes zur kamerabasierten 3D-Konturverfolgung

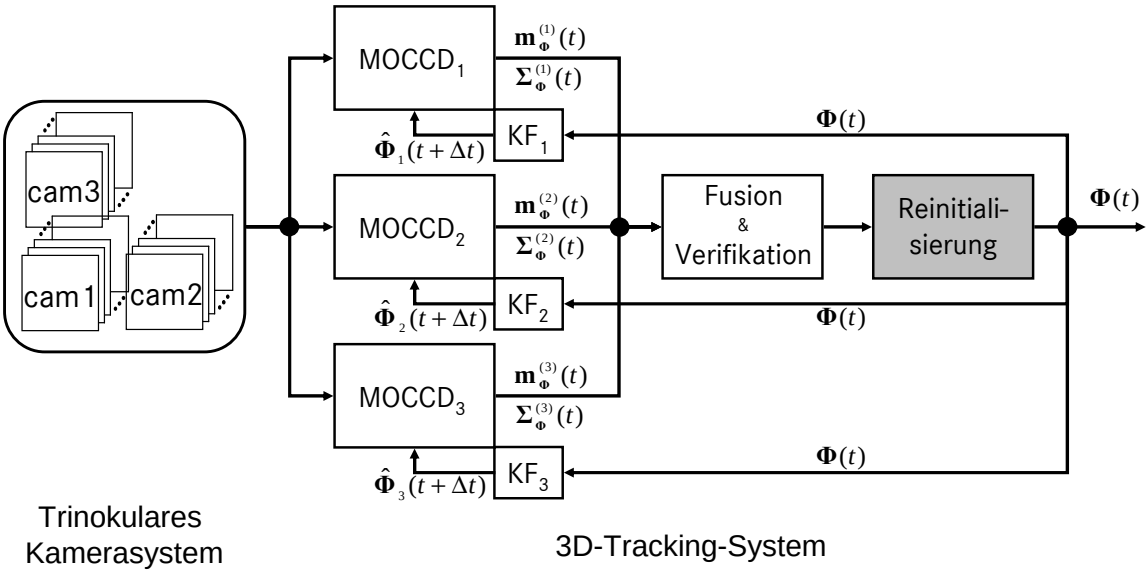

Abbildung 3.1: Struktur des Systemmodells zur 3D-Konturverfolgung.

ist in Abbildung 3.1 dargestellt. Es besteht aus sechs Komponenten: (i) dem Kamerasystem, (ii) dem Körperteil-Modell (nicht dargestellt), (iii) drei MOCCD-Algorithmen, (iv) einem Teilsystem zur Fusion und Verifikation der drei geschätzten 3D-Pose-Vektoren, d.h. $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$, (v) einem Modul zur Reinitialisierung des Systems und (vi) drei zu den MOCCD-Algorithmen zugehörige Kalman-Filter (KF). Das grau unterlegte Rechteck „Reinitialisierung“ steht für das Reinitialisierungsmodul, welches optional verwendet werden kann. Daher ergeben sich zwei Systemvarianten, ein

Trackingsystem mit und ohne Reinitialisierung. Diese werden später in den Experimenten (Abschnitt 7) gegeneinander evaluiert. Im Folgenden wird jede der Komponenten näher erläutert und abschließend die Interaktion der Komponenten im Gesamtsystem dargestellt.

3.1 Kamerasystem

Der sich in der Szene befindende Mensch wird mit einem Mehrkamerasystem mit kleiner Basisbreite beobachtet. Dieses Kamerasystem ist dem Überwachungssystem SafetyEYE[1] sehr ähnlich. Als Kameras kommen drei **Basler Scout** zum Einsatz (Abbildung 3.2 (oben rechts)). Die drei Sensoren des Kamerasystems sind rechtwinklig angeordnet und haben einen horizontalen und vertikalen Abstand von 150 mm. Die Objektive besitzen eine Brennweite von 6 mm, die einzelnen Sensoren liefern bis zu 14 Bilder pro Sekunde in einer Auflösung von 516×389 Pixeln. Das Kamerasystem liefert synchrone Bilder der Szene aus leicht unterschiedlichen Blickrichtungen (Abbildung 3.2), dadurch ist es möglich, 3D-Daten der Szene zu berechnen. Die drei Kameras sind im Weltkoordinatensystem W angeordnet. Dies entspricht dem Kamerakoordinatensystem der Masterkamera. Durch Kamerakalibrierung (Abschnitt A.1.4) werden die intrinsischen und extrinsischen Kameraparameter für jede Kamera $c \in \{1, \ldots, N_c\}$ bestimmt.

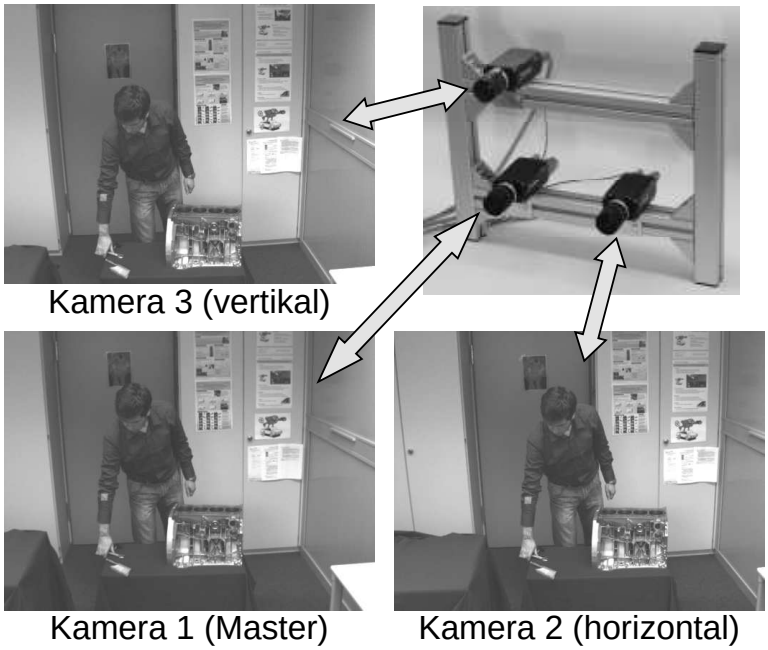

Abbildung 3.2: Trinokulare Kamera (Mehrkamerasystem mit kleiner Basisbreite) und ein Beispieldatensatz. Durch Aufnahme der Szene aus unterschiedlichen Blickwinkeln und -richtungen ist es möglich, die beobachtete Szene dreidimensional zu rekonstruieren.

[1] www.safetyeye.com, (Winkler, 2006)

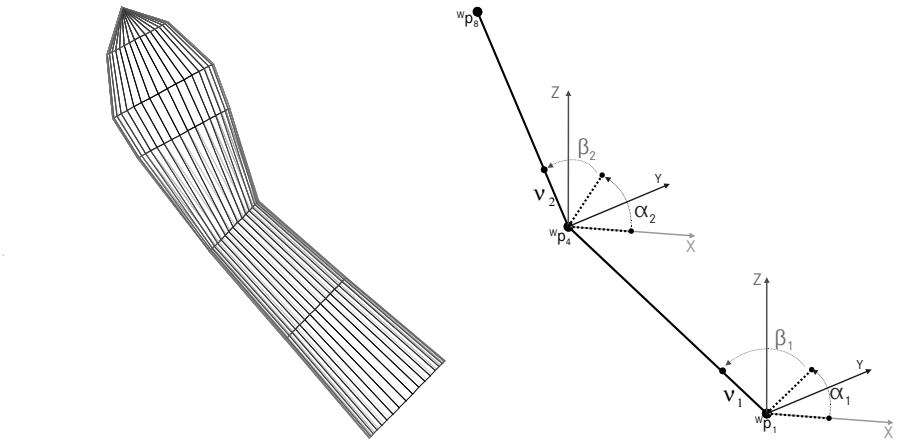

Abbildung 3.3: Links: Die fünf Kegelstümpfe und der Kegel des Hand-Unterarm-Modells. Rechts: Darstellung der Freiheitsgrade des Unterarm- und Handgelenks im 3D-Hand-Unterarm-Modell. Da das Modell rotationssymmetrisch ist, haben Unterarm- und Handgelenk nur jeweils zwei Freiheitsgrade.

3.2 Modellierung von Körperteilen

Ein wesentlicher Bestandteil des entwickelten Verfahrens ist die Modellierung der in den Bildern zu verfolgenden Körperteile. Körperteilmodelle (3D-Konturen) werden mit Hilfe des MOCCD-Algorithmus (Abschnitt 3.3.1) an die Bilddaten des Mehrkamerasystems angepasst. Das entwickelte 3D-Konturverfolgungsystem ist so gestaltet, dass sich beliebige Objekte in den Kamerabildern verfolgen lassen. Die einzige Voraussetzung ist die Verfügbarkeit eines Konturmodells des zu verfolgenden Objekts. In diesem Abschnitt werden Modelle zur kamerabasierten Verfolgung des menschlichen Hand-Unterarms und der Kopf-Schulter-Partie vorgestellt. Beide 3D-Modelle gehören zur Kategorie der analytischen Modelle (Abschnitt 2.1.3). Diese Modellkategorie erlaubt mit relativ wenigen Parametern ein große Modellierungsvielfalt. Außerdem sind alle Parameter gut durch den Menschen zu interpretieren, was z.B. bei Prototyp-Modellen (Abschnitt 2.1.4) nicht so einfach möglich ist.

3.2.1 3D-Hand-Unterarm-Modell

Es wird ein analytisches 3D-Modell eingesetzt. Das Modell zur kamerabasierten Verfolgung der menschlichen Hand-Unterarm-Region besteht aus fünf Kegelstümpfen und einem Kegel (Abbildung 3.3 (links)) und ist somit rotationssymmetrisch. Die einzelnen Finger werden aufgrund der geringen Detaillierungsstufe nicht modelliert. Die geometrischen 3D-Körper des 3D-Modells werden durch acht 3D-Punkte (${}^W\mathbf{p}_1$ bis ${}^W\mathbf{p}_8$) sowie sechs Radien (r_1 bis r_6) im Weltkoordinatensystem W definiert (Abbildung 3.4). Der Parametervektor $\mathbf{\Phi}$ zur Erzeugung des Modells umfasst folgende neun Parameter:

$$\mathbf{\Phi} = [{}^W p_{1X}, {}^W p_{1Y}, {}^W p_{1Z}, \alpha_1, \beta_1, \alpha_2, \beta_2, r_1, r_4]^T. \tag{3.1}$$

Dabei beinhaltet $\mathbf{\Phi}$ 3D-Positionsparameter $({}^W p_{1X}, {}^W p_{1Y}, {}^W p_{1Z})$ und Drehlageparameter $(\alpha_1, \beta_1, \alpha_2, \beta_2)$ des Modells sowie Parameter (r_1, r_4) die die Modellgeometrie beeinflussen, sogenannte Deformationsparameter.

Die 3D-Punkte ${}^W\mathbf{p}_2$ bis ${}^W\mathbf{p}_8$ sowie die vier Radien (r_2, r_3, r_5, r_6), welche nicht explizit Elemente des Parametervektors $\mathbf{\Phi}$ sind, werden mit $\mathbf{\Phi}$ ermittelt. Der 3D-Punkt ${}^W\mathbf{p}_1$ ist der Beginn des Unterarms und Bestandteil des zu optimierenden Parametervektors $\mathbf{\Phi}$. Die Freiheitsgrade des Unterarm- und Handgelenks werden durch die Winkel α_1, β_1, α_2 und β_2 modelliert. Durch diese werden zwei Richtungsvektoren erzeugt, welche die Rotationen beschreiben:

$$\boldsymbol{\nu}_1 = \mathbf{R_Z}(\beta_1) \cdot \mathbf{R_Y}(\alpha_1) \cdot l_{\text{Unterarm}} \cdot [1,0,0]^T \tag{3.2}$$

$$\boldsymbol{\nu}_2 = \mathbf{R_Z}(\beta_2) \cdot \mathbf{R_Y}(\alpha_2) \cdot l_{\text{Hand}} \cdot [1,0,0]^T . \tag{3.3}$$

Die Konstante l_{Unterarm} muss manuell festgelegt werden und beschreibt die Länge des Unterarms. Die Länge der Hand l_{Hand} definiert den euklidischen Abstand von Handgelenk ${}^W\mathbf{p}_4$ und Handspitze ${}^W\mathbf{p}_8$ und muss ebenfalls manuell festgelegt werden. Da das Modell rotationssymmetrisch ist, haben Unterarm- und Handgelenk nur zwei Freiheitsgrade (Abbildung 3.3 (rechts)). Mit den Winkeln α_1 und β_1 werden die Freiheitsgrade des Unterarmgelenks und mit α_2 und β_2 die des Handgelenks beschrieben (Abbildung 3.3 (rechts)). $\mathbf{R_Y}(\alpha_1)$ steht für die orthonormale Rotationsmatrix um die Y-Achse in $\mathbb{R}^3$ mit Winkel α_1 und $\mathbf{R_Z}(\beta_1)$ beschreibt die orthonormale Rotationsmatrix um die Z-Achse in $\mathbb{R}^3$ mit dem Rotationswinkel β_1. Die beiden Rotationsmatrizen sind folgendermaßen definiert:

$$\mathbf{R_Y}(\alpha) = \begin{pmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{pmatrix}, \quad \mathbf{R_Z}(\alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{3.4}$$

Die Punkte ${}^W\mathbf{p}_2$ bis ${}^W\mathbf{p}_8$ des 3D-Modells (Abbildung 3.4) werden wie folgt berechnet:

$$\begin{aligned} {}^W\mathbf{p}_2 &= {}^W\mathbf{p}_1 + 0.5 \cdot \boldsymbol{\nu}_1 \\ {}^W\mathbf{p}_3 &= {}^W\mathbf{p}_1 + 0.95 \cdot \boldsymbol{\nu}_1 \\ {}^W\mathbf{p}_4 &= {}^W\mathbf{p}_1 + \boldsymbol{\nu}_1 \\ {}^W\mathbf{p}_5 &= {}^W\mathbf{p}_4 + 0.4 \cdot \boldsymbol{\nu}_2 \\ {}^W\mathbf{p}_6 &= {}^W\mathbf{p}_4 + 0.6 \cdot \boldsymbol{\nu}_2 \\ {}^W\mathbf{p}_7 &= {}^W\mathbf{p}_4 + 0.85 \cdot \boldsymbol{\nu}_2 \\ {}^W\mathbf{p}_8 &= {}^W\mathbf{p}_4 + \boldsymbol{\nu}_2. \end{aligned} \tag{3.5}$$

Das 3D-Hand-Unterarm-Modell besteht außerdem aus sechs Radien r_1 bis r_6, von denen r_1 sowie r_4 Bestandteil des Parametervektors $\mathbf{\Phi}$ sind. Die restlichen Radien ergeben sich mit:

$$\begin{aligned} r_2 &= 0.8 \cdot r_1 \\ r_3 &= 0.65 \cdot r_1 \\ r_5 &= 1.1 \cdot r_4 \\ r_6 &= 26 \text{ mm} \quad \text{konstant.} \end{aligned} \tag{3.6}$$

Die Skalierungsparameter in den Gleichungen (3.6) werden durch Abbildung 3.4 motiviert und sind aus der menschlichen Anatomie abgeleitet. Durch Akima-Interpolation (Akima, 1970) der Radien entlang der Richtungsvektoren von Unterarm und Hand entsteht ein detailliertes

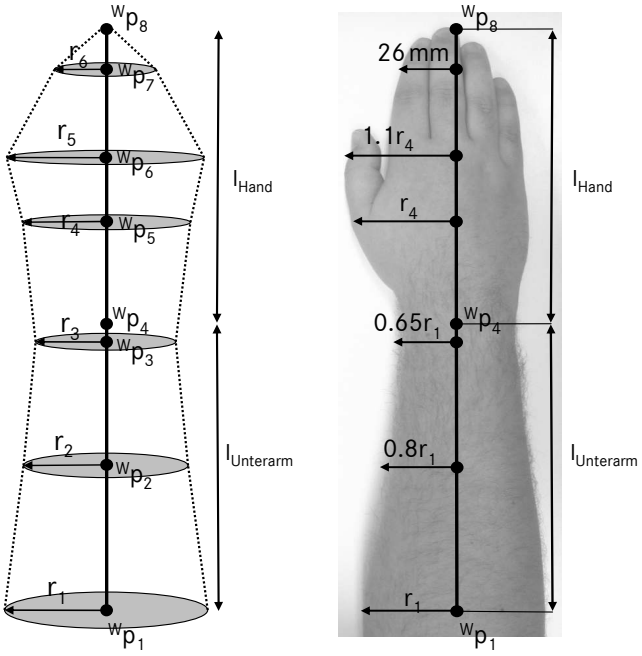

Abbildung 3.4: Links: Definition der fünf Kegelstümpfe und des Kegels. Die 3D-Körper werden durch acht 3D-Punkte (${}^{W}\mathbf{p}_1$ bis ${}^{W}\mathbf{p}_8$) sowie sechs Radien (r_1 bis r_6) definiert. Rechts: Motivation der Parameter-Abhängigkeiten im 3D-Hand-Unterarm-Modell.

3D-Volumenmodell, welches aus vielen Kegelstümpfen besteht. Dieses detaillierte Volumenmodell wird aber nur zur Visualisierung verwendet. Die 3D-Pose-Estimation erfolgt mit dem groben 3D-Volumenmodell, welches aus fünf Kegelstümpfen und einem Kegel besteht. Das entwickelte System basiert auf dem MOCCD-Algorithmus (Abschnitt 3.3.1). Dieser passt ein 3D-Konturmodell an eine Objektkontur in mehreren Kamerabildern durch Verschiebung und Drehung des Modells sowie durch Optimierung der Deformationsparameter an. Daher muss ein 3D-Konturmodell (3D-Kurve) aus dem 3D-Volumenmodell erzeugt werden, welches später in die Kamerabilder projiziert werden kann. Zur Ermittlung des 3D-Konturmodells wird aus dem groben 3D-Volumenmodell (Abbildung 3.3 (links)) ein 3D-Konturmodell (3D-Kurve) erzeugt. Zur Berechnung der 3D-Modellkurve wird ein Sichtvektor ${}^{C_c}\mathbf{p}_4$ aus dem Kameraursprung (Kamerakoordinatensystem C_c der Kamera c) auf das Handgelenk benötigt. Dieser wird mit den durch Kamerakalibrierung (Anhang A.1.4) ermittelten extrinsischen Kameraparametern (Anhang A.1.2) berechnet:

$$ {}^{C_c}\mathbf{p}_{4(h)} = {}^{C_c}_{W}\mathbf{H}\ {}^{W}\mathbf{p}_{4(h)}. \tag{3.7} $$

Mit dem Sichtvektor der Kamera ${}^{C_c}\mathbf{p}_4$ und dem Richtungsvektor des Unterarms $\boldsymbol{\nu}_1$ sowie dem des Handgelenks $\boldsymbol{\nu}_2$ werden Normalenvektoren $\boldsymbol{\eta}_1$ und $\boldsymbol{\eta}_2$ berechnet, mit denen die sichtbaren „Ränder“ des 3D-Volumenmodells, unter Nutzung des Kreuzprodukts, extrahiert werden. Durch den Index (h) wird definiert, dass die Punkte in homogenen Koordinaten vorliegen. Das 3D-Konturmodell – bestehend aus 13 verbundenen 3D-Punkten – wird wie im Anhang A.1.2 beschrieben, in das jeweilige Pixelkoordinatensystem der Kamera projiziert. Eine glatte 2D-Kontur wird durch Akima-Interpolation (Akima, 1970) entlang der 13 projizierten Kurvenpunkte erzeugt. Abbildung 3.5 zeigt schematisch das implementierte Verfahren zur Extraktion

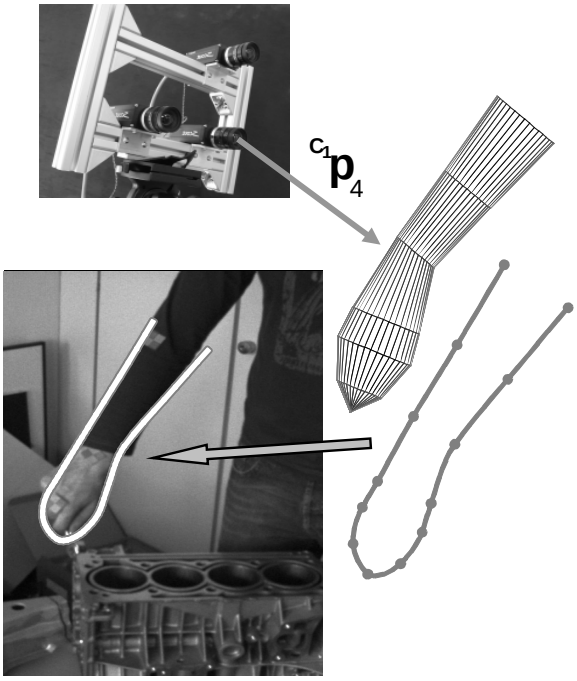

Abbildung 3.5: Extraktion der 3D-Modellkurve aus dem groben 3D-Volumenmodell und Projektion dieser Kurve in das Bild der Masterkamera.

und Projektion des 3D-Volumenmodells für die Masterkamera (Kamera 1).

3.2.2 3D-Kopf-Schulter-Modell

Das verwendete 3D-Modell zur Beschreibung der Kopf-Schulter-Partie ist kein 3D-Volumenmodell, sondern wird direkt als 3D-Modellkurve modelliert. Wiederum wird ein analytisches Modell (Abschnitt 2.1.3) eingesetzt, da diese Modellkategorie eine einfache und gut interpretierbare Modellbeschreibung ermöglicht. Die 3D-Modellkurve besteht aus 13 3D-Kontrollpunkten ${}^W\mathbf{c}_1 - {}^W\mathbf{c}_{13}$ im Weltkoordinatensystems W, diese ergeben sich aus einem Skelett von acht 3D-Punkten (${}^W\mathbf{p}_1$ bis ${}^W\mathbf{p}_8$) sowie sechs Radien (r_1 bis r_6) (Abbildung 3.6 (links)). Außerdem sind die Länge von Hals l_{Hals} und Kopf l_{Kopf} notwendige konstante Parameter. Folgender Parametervektor

$$\mathbf{\Phi} = [{}^W p_{1X}, {}^W p_{1Y}, {}^W p_{1Z}, \alpha, \beta_1, \beta_2, r_4]^T \tag{3.8}$$

erzeugt die 3D-Modellkurve. Alle nicht enthaltenen Modellparameter werden aus $\mathbf{\Phi}$ ermittelt. Die 3D-Kontrollpunkte der 3D-Modellkurve liegen in einer Ebene, welche parallel zur XY-Ebene des Weltkoordinatensystems W ist. Dies bedeutet, dass alle Punkte den gleichen Abstand zur Masterkamera haben und eine Rotation um die Y-Achse, beschrieben durch $\mathbf{R_Y}$ in Gleichung (3.4), nicht modelliert wird. Der 3D-Punkt ${}^W\mathbf{p}_1$ definiert das Zentrum des Halses und ist Bestandteil des Parametervektors $\mathbf{\Phi}$. Die Freiheitsgrade von Hals und Kopf werden durch die Winkel β_1 und β_2 definiert und zwei Richtungsvektoren erzeugt:

$$\boldsymbol{\nu}_1 = \mathbf{R_Z}(\beta_1) \cdot l_{Hals} \cdot [1,0,0]^T \tag{3.9}$$

$$\boldsymbol{\nu}_2 = \mathbf{R_Z}(\beta_2) \cdot l_{Kopf} \cdot [1,0,0]^T . \tag{3.10}$$

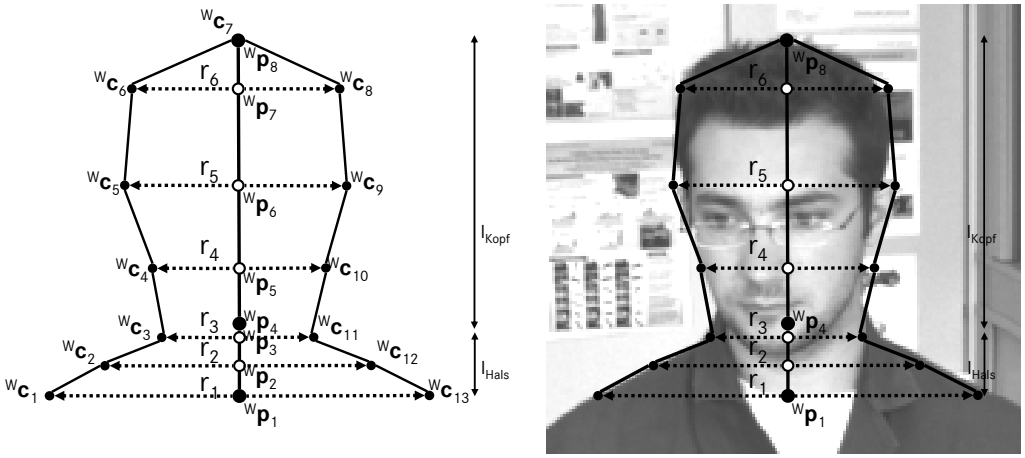

Abbildung 3.6: Extraktion der 3D-Modellkurve und Projektion in eines der Bilder.

Diese beschreiben die Rotation um die Z-Achse (Gleichung (3.4)). Somit können die 3D-Skelettpunkte $^W\mathbf{p}_2$ - $^W\mathbf{p}_8$ folgendermaßen berechnet werden:

$$\begin{aligned}
^W\mathbf{p}_2 &= {}^W\mathbf{p}_1 + 0.4 \cdot \boldsymbol{\nu}_1 \\
^W\mathbf{p}_3 &= {}^W\mathbf{p}_1 + 0.8 \cdot \boldsymbol{\nu}_1 \\
^W\mathbf{p}_4 &= {}^W\mathbf{p}_1 + \boldsymbol{\nu}_1 \\
^W\mathbf{p}_5 &= {}^W\mathbf{p}_4 + 0.2 \cdot \boldsymbol{\nu}_2 \\
^W\mathbf{p}_6 &= {}^W\mathbf{p}_4 + 0.5 \cdot \boldsymbol{\nu}_2 \\
^W\mathbf{p}_7 &= {}^W\mathbf{p}_4 + 0.8 \cdot \boldsymbol{\nu}_2 \\
^W\mathbf{p}_8 &= {}^W\mathbf{p}_4 + \boldsymbol{\nu}_2.
\end{aligned} \tag{3.11}$$

Die Skalierungsparameter in den Gleichungen (3.11) sind aus der menschlichen Anatomie abgeleitet und werden durch Abbildung 3.6 (rechts) motiviert. Außerdem werden sechs Radien r_1 - r_6 zur Erzeugung der Kontrollpunkte benötigt. Von diesen ist nur der Radius r_4 Bestandteil des Parametervektors $\mathbf{\Phi}$. Die restlichen Radien ergeben sich mit:

$$\begin{aligned}
r_1 &= 13 \ mm \quad \text{konstant} \\
r_2 &= 1.5 \cdot r_4 \\
r_3 &= 0.9 \cdot r_4 \\
r_5 &= 1.3 \cdot r_4 \\
r_6 &= 1.15 \cdot r_4.
\end{aligned} \tag{3.12}$$

Die Skalierungsparameter in den Gleichungen (3.12) werden ebenfalls durch Abbildung 3.6 (rechts) motiviert und sind aus der menschlichen Anatomie abgeleitet.

Die Bestandteile des Skelettes des 3D-Modells sind nun definiert und die 13 3D-Kontrollpunkte $^W\mathbf{c}_1 - {}^W\mathbf{c}_{13}$ können berechnet werden. Durch Verbindung der Kontrollpunkte erhält man eine 3D-Modellkurve (Abbildung 3.6 (links)). Diese 3D-Modellkurve liegt in einer Ebene parallel zur XY-Ebene des Weltkoordinatensystems W, somit wird die Rotation um die Y-Achse des Weltkoordinatensystems in diesem Modell nicht beschrieben. Anschaulich betrachtet

bedeutet dies, dass ein Nicken der Kurve in Richtung der Kamera nicht modelliert wird. Solche Bewegungen treten in Realität natürlich auf und werden aber in der Literatur häufig vernachlässigt (siehe Hanek (2004) und Treptow u. a. (2005)), da angenommen wird, dass sich die beobachtete Person parallel zur Kamera bewegt. Um solche Nickbewegungen behandeln zu können, wird Parameter α verwendet. Der Parameter α beschreibt die perspektivische Stauchung und Streckung des Konturmodells (Abbildung 3.7). Alle Kontrollpunkte werden in Richtung der Kopf- und Halsachse, um den Faktor $\cos(\alpha)$ gestaucht. Für die Kurvenpunkte der Schulter (${}^W\mathbf{c}_1 - {}^W\mathbf{c}_4$ und ${}^W\mathbf{c}_{10} - {}^W\mathbf{c}_{13}$) kommt es außerdem zu einer Streckung in der Richtung vom Zentrum des Halses ${}^W\mathbf{p}_1$ zum Punkt hin, um den Faktor $1 + (1 - \cos(\alpha))$ (Abbildung 3.7 (links)).

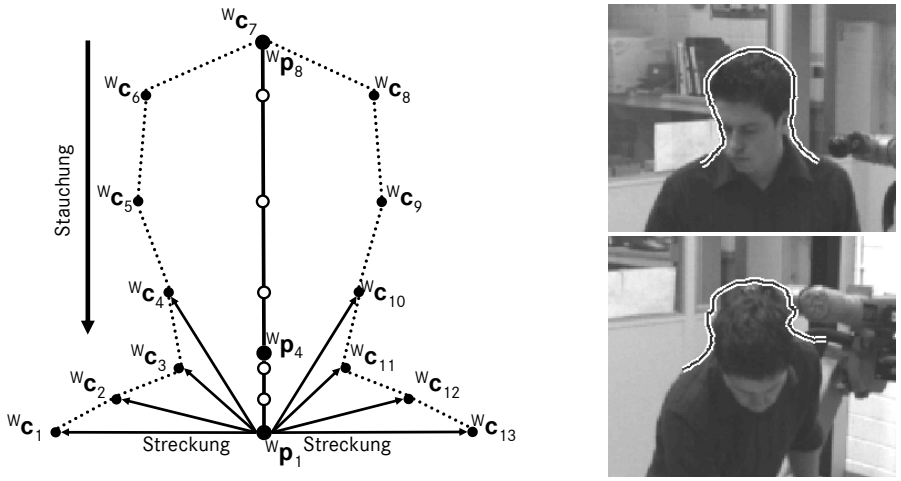

Abbildung 3.7: Links: Perspektivische Stauchung und Streckung der 3D-Modellkurve durch den Parameter α. Rechts: Zwei Beispiele einer Nickbewegung in Richtung der Kamera und projiziertes 3D-Konturmodell.

3.3 3D-Pose-Estimation

In diesem Abschnitt geht es um ein Verfahren zur Bestimmung der dreidimensionalen Lage und Orientierung eines bekannten Objekts im Weltkoordinatensystem W. Diese 3D-Lagebestimmung wird als 3D-Pose-Estimation bezeichnet. Dabei ist es wichtig, dass das zu erkennende Objekt bekannt ist, d.h. ein Modell des Objekts vorhanden ist. In der Bildverarbeitung basiert die 3D-Pose-Estimation auf einem oder mehreren Bildern, aus denen die zur Anpassung notwendigen Merkmale extrahiert werden. Das Ergebnis der 3D-Pose-Estimation ist eine 3D-Pose $\mathbf{\Phi}$. Diese 3D-Pose $\mathbf{\Phi}$ wird berechnet, indem durch Optimierung einer Zielfunktion das bekannte Objektmodell an die aus den Bildern extrahierten Merkmale angepasst (gefittet) wird. Die Arbeit von Wöhler (2009) gibt einen detaillierten Überblick über Verfahren zur 3D-Pose-Estimation in der Bildverarbeitung. Weitere Verfahren werden von Kölzow (2002); Hanek (2004); d'Angelo u. a. (2004); Stößel (2007); Krüger (2007); Barrois u. a. (2009) detailliert beschrieben. Außerdem zeigt die Arbeit von Stößel (2007), dass sich das Pose-Estimation-Problem mit probabilistischen Methoden auch in hochdimensionalen Räumen effektiv lösen

lässt.

In diesem Kapitel wird der multiokulare Contracting-Curve-Density (MOCCD) Algorithmus zur 3D-Pose-Estimation eingesetzt. Der zum Teil neu entwickelte und verbesserte Bayes'sche Ansatz des MOCCD-Algorithmus erlaubt eine stabile Pose-Estimation im angestrebten Szenario (Abschnitt 1.2). Denn in diesem kann es zu erheblichem Bildrauschen sowie stark strukturierten Hintergründen kommen, was bei vielen etablierten 3D-Pose-Estimation-Verfahren zu schlechten Ergebnissen führen kann.

Im Vergleich zu den Arbeiten von Kölzow (2002); d'Angelo u. a. (2004); Stößel (2007); Krüger (2007); Barrois u. a. (2009) wird in dieser Arbeit neben der eigentlichen 3D-Pose-Estimation auch eine Schätzung der Modellgeometrie durchgeführt. Dazu werden Deformationsparameter (Geometrieparameter) in den zu optimierenden Parametervektor aufgenommen (Abschnitt 3.2). Dies ist für das angestrebte Szenario dieser Arbeit besonders wichtig, da nicht davon ausgegangen werden kann, dass ein personenspezifisches 3D-Modell des Arbeiters vorliegt. Die entwickelten Verfahren müssen daher in der Lage sein, sich an unterschiedliche Arbeiter anzupassen und müssen auch dann ein robustes Tracking gewährleisten, wenn der Arbeiter Handschuhe trägt.

3.3.1 Multiokularer Contracting-Curve-Density (MOCCD) Algorithmus

Der multiokulare Contracting-Curve-Density (MOCCD) Algorithmus (Krüger u. Ellenrieder, 2005) ist ein Verfahren zur 3D-Pose-Estimation in mindestens zwei 2D-Bildern, welches den monokularen Contracting-Curve-Density (CCD) Algorithmus (Hanek, 2004) erweitert. Grundlage der Erweiterung ist ein System aus N_c kalibrierten Kameras in einem definierten Weltkoordinatensystem W. Das Modell einer parametrischen Kurve (für den CCD-Algorithmus eine 2D-Kurve und für den MOCCD-Algorithmus eine 3D-Modellkurve) wird dabei an die Bilddaten so angepasst, dass die Pixelstatistiken auf der inneren und äußeren Seite der parametrischen Kurve möglichst unterschiedlich sind. Beide Algorithmen sind Pose-Refinement Verfahren, denn es wird die Gaußverteilung $p(\boldsymbol{\Phi}) \approx p(\boldsymbol{\Phi}|\widehat{\mathbf{m}}_{\boldsymbol{\Phi}}, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}})$ des Modellparametervektors $\boldsymbol{\Phi}$, welche durch den Mittelwertvektor $\widehat{\mathbf{m}}_{\boldsymbol{\Phi}}$ und die Kovarianzmatrix $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{\Phi}}$ beschrieben wird, iterativ angepasst und „verfeinert“ (engl. *refined*). Da der MOCCD-Algorithmus den CCD-Algorithmus auf die Bilder eines kalibrierten Mehrkamerasystems erweitert, werden als Erstes die Grundlagen der verwendeten Implementierung des CCD-Algorithmus (Hanek, 2004) beschrieben.

Contracting-Curve-Density (CCD) Algorithmus

Dieser Abschnitt beschreibt die Grundlagen der in dieser Arbeit verwendeten Implementierung des CCD-Algorithmus, dabei wird auch auf Änderungen und Erweiterungen zur von Hanek (2004) beschriebenen Original-Version eingegangen. Der CCD-Algorithmus ist ein modellbasiertes Verfahren zur Segmentierung eines Objekts vom Hintergrund in Bilddaten. Er optimiert den Parametersatz $\boldsymbol{\Phi}$, der die Pixelstatistik S im Bild $\mathbf{I}^*$ auf der inneren und äußeren Seite der Kurve trennt. In dieser Arbeit werden diese mit Seite 1 und Seite 2 der Kurve bezeichnet. Als Eingabeparameter benötigt der CCD-Algorithmus ein Bild und ein parametrisches

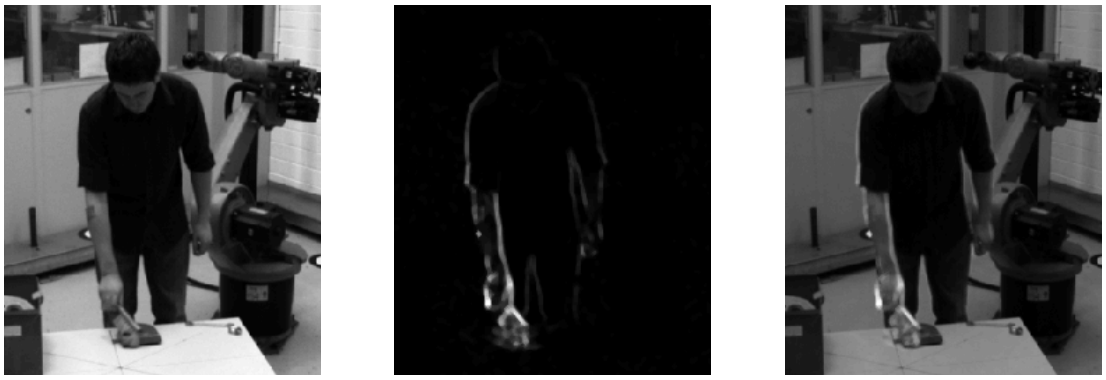

Abbildung 3.8: Links: Original 8-Bit Bild $\mathbf{I}_t$. Mitte: Bild der absoluten Differenzen. Rechts: Erweitertes Eingabebild $\mathbf{I}_t^*$ (skaliert auf 8-bit) des verwendeten CCD-Algorithmus.

Kurvenmodell, dieses wird an die Bilddaten angepasst. Die Bilddaten $\mathbf{I}^*$ sind eine aus Pixelwerten (Grauwerte, Farbe, Bildmerkmale, etc.) bestehende Matrix. Ein erster Unterschied zur Implementierung von Hanek (2004) besteht darin, dass die in dieser Arbeit verwendete Version des CCD-Algorithmus auf einem erweiterten Eingabebild basiert. Eine modellbasierte Segmentierung in der angestrebten Applikation ist sehr anspruchsvoll, da nur eine grobe Beschreibung des zu verfolgenden Objekts möglich ist und es zu erheblichem Bildrauschen sowie stark strukturierten Hintergründen kommen kann. Um eine genaue und robuste Segmentierung zu ermöglichen, wird ausgenutzt, dass in der Applikation die Kamera einen festen Standpunkt besitzt. Das erweiterte Eingabebild $\mathbf{I}_t^*$ zum Zeitschritt t wird berechnet durch:

$$\mathbf{I}_t^* = \mathbf{I}_t + \lambda \left| \mathbf{I}_t - \mathbf{I}_{(t-1)} \right| . \tag{3.13}$$

Dabei beschreibt $\mathbf{I}_t$ das originale Kamerabild zum Zeitschritt t und $\left| \mathbf{I}_t - \mathbf{I}_{(t-1)} \right|$ das Bild der absoluten Differenzen, welches aus dem aktuellen und vorherigen Bild berechnet wird. Der Faktor λ beschreibt dabei den Einfluss des Bildes der absoluten Differenzen. Durch die Kombination zum erweiterten Eingabebild werden Pixelwerte in Bereichen mit Bewegung erhöht, was zu einer robusteren Segmentierung führt. Ein weiterer Vorteil ist, dass wenn keine Bewegung stattfindet, das erweiterte Eingabebild $\mathbf{I}_t^*$ dem von der Kamera gelieferten Bild $\mathbf{I}_t$ entspricht und somit trotzdem eine Segmentierung möglich ist. Abbildung 3.8 zeigt das original 8-Bit Bild, das Bild der absoluten Differenzen und das erweiterte Eingabebild des verwendeten CCD-Algorithmus. Im Folgenden wird angenommen im Zeitschritt t zu sein, daher wird aufgrund der Übersichtlichkeit beim erweiterten Eingabebild $\mathbf{I}_t^*$ der Zeitschritt t weggelassen.

Das an das erweiterte Eingabebild $\mathbf{I}^*$ angepasste Kurvenmodell besteht aus zwei Teilen, einer Kurvenfunktion und einer a-priori Gaußverteilung der Modellparameter $\mathbf{\Phi}$. Die Kurvenfunktion $c(k, \mathbf{\Phi})$ beschreibt das Modell am k-ten Punkt, $k \in [0, K]$, auf der Kurve (Abbildung 3.9 (links)). Die a-priori Gaußverteilung $p(\mathbf{\Phi}) \approx p(\mathbf{\Phi} | \widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}})$ der Modellparameter $\mathbf{\Phi}$ wird durch den Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}$ und die Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}$ definiert. Dabei beschreibt der Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}$ die Kurve und die Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}$ die entsprechende Unsicherheit entlang dieser. Vor der ersten Iteration wird der CCD-Algorithmus initialisiert, indem die optimierten Parameter $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf die gegebenen a-priori Parameter $(\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}})$ gesetzt werden. Der CCD-Algorithmus adaptiert das Modell, beschrieben durch $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$, an das Eingabebild durch folgende zwei Schritte, welche so lange wiederholt werden, bis die Änderungen von $\mathbf{m}_{\mathbf{\Phi}}$

und $\mathbf{\Sigma_\Phi}$ unter einer vorgegebenen Schwelle liegen oder eine bestimmte Anzahl von Iterationen erreicht ist.

1. **Schritt – Berechnung der lokalen Pixelstatistiken:** Zur Berechnung der lokalen Pixelstatistiken $S(\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi})$ auf beiden Seiten der Kurve werden Liniensegmente verwendet (Abbildung 3.9 (rechts)), diese sind senkrecht zur Kurve ausgerichtet. Für Grauwertbilder bedeutet dies, dass für jede Seite und jedes Liniensegment ein Mittelwert und eine Standardabweichung der Grauwerte berechnet wird.
2. **Schritt – Anpassen der Modellparameter:** Durch Verfeinerung der Modellparameter $(\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi})$ wird die Wahrscheinlichkeit $p(\mathbf{\Phi}|\mathbf{I}^*)$ maximiert.

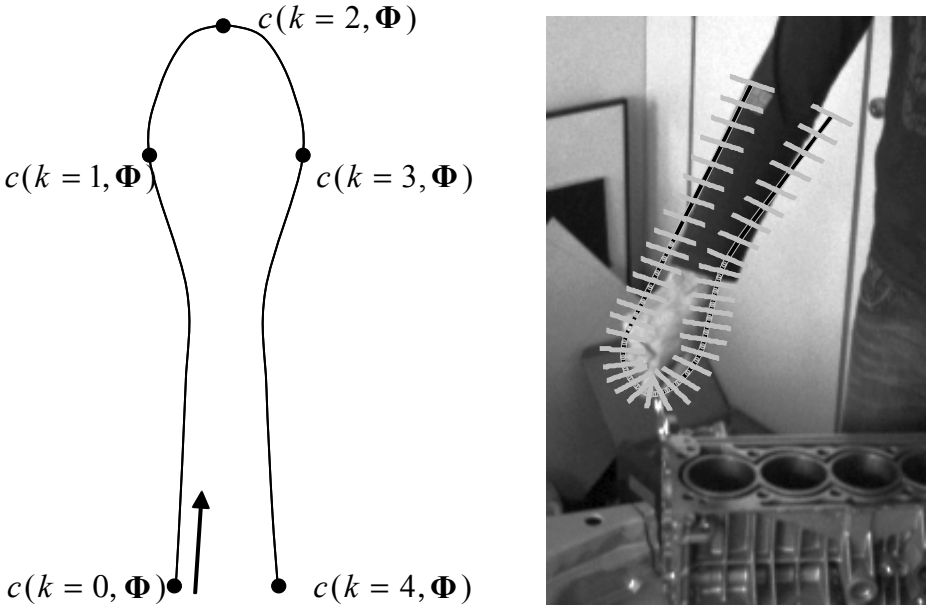

Abbildung 3.9: Links: 2D-Kurvenmodell, welches durch die Kurvenfunktion $c(k, \mathbf{\Phi})$ beschrieben wird. Der Parameter k beschreibt dabei einen Kurvenpunkt und $\mathbf{\Phi}$ enthält die Modellparameter. Rechts: 2D-Kurvenmodell auf dem Eingabebild und die $(K+1)$ Liniensegmente, welche senkrecht zur Kurve ausgerichtet sind.

1. Schritt – Berechnung der lokalen Pixelstatistiken: Zur Berechnung der Pixelstatistiken $S(\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi})$ in der Nachbarschaft der Kurve werden nur Pixel genutzt, die auf $(K+1)$ verschiedenen Liniensegmenten entlang der Modellkurve liegen (Abbildung 3.9 (rechts)). Diese Pixel werden probabilistisch, d.h. mit einer Wahrscheinlichkeit, einer der beiden Seiten zugeordnet und danach erfolgt die Berechnung der lokalen Statistiken für jede Seite.

Ein Liniensegment $k \in \{0, \ldots, K\}$ ist eine zur Modellkurve zentrierte Senkrechte im Modellpunkt

$$\mathbf{c}_k = c(k, \mathbf{\Phi}). \tag{3.14}$$

Im Gegensatz zum originalen CCD-Algorithmus (Hanek, 2004), werden die Liniensegmente in dieser Arbeit nicht gleichmäßig entlang der Kurve verteilt (Abbildung 3.9 (rechts)). Der Grund hierfür liegt in den verwendeten Modellen, diese sind keine geschlossenen Kurven und somit zu einer Seite geöffnet. Es werden mehr Liniensegmente auf der geschlossenen Seite der

Kurve verwendet, um z.B. der Verschiebung entlang des Unterarms oder des Kopfes ein höheres Gewicht zu geben. Die Länge der Liniensegmente ist abhängig von der lokalen Unsicherheit σ_k entlang des Liniensegments k, wird mit

$$\sigma_k^2 = \mathbf{n}_k^T \cdot \mathbf{J}_{\mathbf{c}_k} \cdot \mathbf{\Sigma}_{\mathbf{\Phi}} \cdot \mathbf{J}_{\mathbf{c}_k}^T \cdot \mathbf{n}_k \tag{3.15}$$

berechnet und ändert sich in jedem Iterationsschritt. Dabei beschreibt $\mathbf{n}_k$ die Normale und $\mathbf{J}_{\mathbf{c}_k}$ die erste Ableitung des Modells im Punkt $\mathbf{c}_k$. Beim originalen CCD-Algorithmus (Hanek, 2004) wird für jedes Liniensegment k die lokale Unsicherheit σ_k entlang der Kurve berechnet. In dieser Arbeit wird für alle Liniensegmente eine fixe lokale Unsicherheit σ verwendet. Dieser Schritt vermeidet eine große Anzahl nichtlinearer Funktionsaufrufe und beschleunigt somit den eingesetzten CCD-Algorithmus. Die fixe lokale Unsicherheit σ wird mit

$$\sigma = (\sigma_0 + \sigma_{K/2})/2 \tag{3.16}$$

aus dem Mittel der lokalen Unsicherheiten σ_0 und $\sigma_{K/2}$ der Liniensegmente $k = 0$ und $k = K/2$ unter Nutzung von Gleichung (3.15) berechnet. Die fixe lokale Unsicherheit σ wird nun verwendet, um für jedes Pixel $\mathbf{v}_{k,l}$ entlang des Liniensegments k die probabilistische wahrscheinlichkeitsbasierte Zuordnung zu berechnen. Die Variable l beschreibt dabei den Pixelindex auf dem Liniensegment k. Für alle Pixel $\mathbf{v}_{k,l}$ wird die Zuordnungswahrscheinlichkeit $a_{l,1}(d_l)$, dass ein Punkt zur Seite 1 (innere Seite) der Kurve gehört, berechnet mit

$$a_{l,1}(d_l) = \frac{1}{2}\left[\operatorname{erf}\left(\frac{d_l}{\sqrt{2}\sigma}\right) + 1\right]. \tag{3.17}$$

In Gleichung (3.17) steht erf(x) für die gauß'sche Fehlerfunktion. Die vorzeichenbehaftete Distanz $d_l(\mathbf{v}_{k,l})$ entlang des Liniensegments zwischen einem Punkt $\mathbf{v}_{k,l}$ auf dem Liniensegment und dem Punkt $\mathbf{c}_k$ auf der Kurve wird durch

$$d_l(\mathbf{v}_{k,l}) = \mathbf{n}_k^T \left(\mathbf{v}_{k,l} - \mathbf{c}_k\right) \tag{3.18}$$

berechnet. Die probabilistische Zuordnung $a_{l,2}(d_l)$ für Seite 2 (äußere Seite) ergibt sich mit

$$a_{l,2}(d_l) \;\; = \;\; 1 - a_{l,1}(d_l). \tag{3.19}$$

Der Zweck der lokalen Statistiken auf beiden Seiten der Modellkurve besteht darin, zu beschreiben, ob der Pixelwert $\mathbf{I}^*_{k,l}$ an den Pixelkoordinaten $\mathbf{v}_{k,l}$ besser zur Seite 1 oder Seite 2 der Kurve passt. Die Statistiken werden mit Hilfe von Gewichten berechnet. Dabei beschreiben die Gewichte wie geeignet ein Pixel $\mathbf{v}_{k,l}$ zur Berechnung der Statistik für die jeweilige Seite ist. Bei der Berechnung der Gewichte wird vollständig den Empfehlungen von Hanek (2004) gefolgt. Für das Gewicht $w_{l,s}$ an den Pixelkoordinaten $\mathbf{v}_{k,l}$ der Seite s sollen nach Hanek (2004) folgende Bedingungen gelten:

1. Das Gewicht soll nur positiv sein, wenn es zur Seite s gehört, sonst Null.
2. Es sollen nur Pixel in die Berechnung einbezogen werden, die nicht zu weit von der Kurve entfernt sind.
3. Wenn die fixe lokale Unsicherheit σ gering ist, sollen die Gewichte hoch sein.

Diese drei Bedingungen werden durch folgende Gewichtsfunktion erfüllt:

$$w_{l,s} = w_A(a_{l,s}) \cdot w_B(d_l, \sigma) \cdot w_C(\sigma) \quad \text{mit} \quad s \in \{1,2\}\,, \tag{3.20}$$

Gewicht $w_A(a_{l,s})$:
Das Gewicht $w_A(a_{k,l,s})$ beschreibt die Wahrscheinlichkeit, dass der Pixel $\mathbf{v}_{k,l}$ zur Seite $s \in \{1,2\}$ gehört.

$$w_A(a_{l,s}) \;=\; \max\left(0, \left[\frac{a_{l,s} - \gamma_1}{1-\gamma_1}\right]^{(2\cdot E_A)}\right) \tag{3.21}$$

Der Parameter $\gamma_1 \in [0,1[$ beschreibt wie hoch die probabilistische Zuordnung $a_{l,s}$ mindestens sein muss, dass der Pixel zur Berechnung der Statistiken genutzt wird. In den Experimenten wird $\gamma_1 = 0.5$ und $E_A = 3$ gewählt.
Gewicht $w_B(d_{k,l}, \sigma_k)$:
Das Gewicht $w_B(d_l, \sigma)$ berücksichtigt die Entfernung des Pixels $\mathbf{v}_{k,l}$ zur Kurve.

$$w_B(d_l, \sigma) \;=\; C \cdot \max\left(0, e^{\left(-d_l^2/(2\cdot\widehat{\sigma})\right)} - e^{(-\gamma_4)}\right) \text{ mit} \tag{3.22}$$

$$\widehat{\sigma} \;=\; \gamma_3 \cdot \sigma + \gamma_4. \tag{3.23}$$

Der Parameter C ist eine Normalisierungskonstante, der Autor wählt $C = 5$. Empfohlene Werte für die weiteren Parameter $\gamma_2 \in [3,5]$, $\gamma_3 \in [4,6]$ und $\gamma_4 \in [2,3]$.
Gewicht $w_C(\sigma_k)$:
Die Gewichtsfunktion $w_C(\sigma)$ evaluiert die lokale Unsicherheit σ der Kurve.

$$w_C(\sigma) = (\sigma + 1)^{-E_C}. \tag{3.24}$$

Es wird empfohlen $E_C \in [1,4]$ zu wählen. Da in dieser Arbeit eine fixe lokale Unsicherheit σ betrachtet wird, sind die probabilistische Zuordnung zur jeweiligen Seite (Gleichungen (3.17) und (3.19)) und die Gewichte (Gleichung (3.20)) nur von der vorzeichenbehafteten Distanz d_l abhängig. Dies ermöglicht es, die Berechnungen im Voraus zu machen und diese zu speichern, der spätere Zugriff erfolgt dabei über die vorzeichenbehaftete Distanz d_l. Dadurch wird wiederum eine große Anzahl nichtlinearer Funktionsaufrufe zur Laufzeit vermieden. Für jedes Liniensegment k und jede Seite $s \in \{1,2\}$ der Kurve werden die gewichteten statistischen Momente $(m_{k,s,0}, \mathbf{m}_{k,s,1}, \mathbf{M}_{k,s,2})$ mit Ordnung $o \in \{0,1,2\}$ durch folgende Gleichungen,

$$m_{k,s,0} \;=\; \sum_{l=1}^{L} w_{l,s} \tag{3.25}$$

$$\mathbf{m}_{k,s,1} \;=\; \sum_{l=1}^{L} w_{l,s} \cdot \mathbf{I}^*_{k,l} \tag{3.26}$$

$$\mathbf{M}_{k,s,2} \;=\; \sum_{l=1}^{L} w_{l,s} \cdot \mathbf{I}^*_{k,l} \cdot {\mathbf{I}^*_{k,l}}^T, \tag{3.27}$$

berechnet. Dabei beschreibt $\mathbf{I}^*_{k,l}$ den Pixelwert des Liniensegments k mit Pixelindex l. Beim originalen CCD-Tracker nach (Hanek, 2004) werden die gewichteten statistischen Momente $(m_{k,s,0}, \mathbf{m}_{k,s,1}, \mathbf{M}_{k,s,2})$ räumlich und zeitlich geglättet, um sich die räumliche und zeitliche

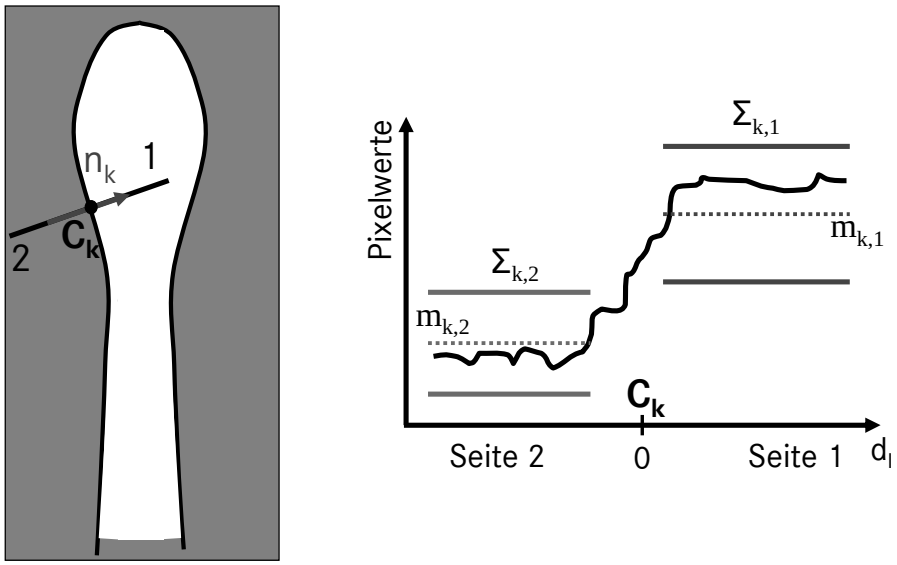

Abbildung 3.10: Darstellung einer Modellkurve mit Punkt $\mathbf{c}_k = c(k, \mathbf{\Phi})$, Kurvennormale $\mathbf{n}_k$ in $\mathbf{c}_k$ und Pixelwerte entlang des Liniensegments. Mit Hilfe der Pixelwerte entlang des Liniensegments werden die lokalen Pixelstatistiken berechnet.

Kohärenz der Pixelwerte zu Nutze zu machen. Dadurch wird ein robusteres Verhalten bei der Anwendung auf Bildfolgen erzielt, da z.B. der Einfluss von strukturierten Hintergründen verringert wird. Im Gegensatz zu Hanek (2004) wird in dieser Arbeit nur eine räumliche Glättung der gewichteten statistischen Momente durchgeführt, da das verwendete Eingabebild $\mathbf{I}^*$ (Gleichung (3.13)) durch Addition des Differenzbildes nur eine geringe zeitliche Kohärenz aufweist. Die räumliche Glättung der statistischen Momente wird mit einem mit Gauß-Filter der Größe 1×5 und einer Standardabweichung von 2 durchgeführt. Damit ergeben sich die geglätteten statistischen Momente $(\widetilde{m}_{k,s,0}, \widetilde{\mathbf{m}}_{k,s,1}, \widetilde{\mathbf{M}}_{k,s,2})$ der Ordnung $o \in \{0, 1, 2\}$ für jedes Liniensegment k und Seite $s \in \{1, 2\}$. Somit können für beide Seiten der Kurve $s \in \{1, 2\}$ der Mittelwert $\mathbf{m}_{k,s}$ und die Kovarianzmatrix $\mathbf{\Sigma}_{k,s}$ der lokalen Statistiken für ein Liniensegment k mit,

$$\mathbf{m}_{k,s}(t) = \frac{\widetilde{\mathbf{m}}_{k,s,1}}{\widetilde{m}_{k,s,0}} \tag{3.28}$$

$$\mathbf{\Sigma}_{k,s}(t) = \frac{\widetilde{\mathbf{M}}_{k,s,2}}{\widetilde{m}_{k,s,0}} - \mathbf{m}_{k,s}(t) \cdot \mathbf{m}_{k,s}^T(t) + \kappa, \tag{3.29}$$

bestimmt werden (Abbildung 3.10). Mit dem Parameter κ werden numerische Instabilitäten vermieden. In den Experimenten wird $\kappa = 2.5$ verwendet. Die lokalen Pixelstatistiken $S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$, welche im 2. Schritt zur Anpassung der Modellparameter verwendet werden, bestehen aus $(K + 1)$ verschiedenen lokalen Mittelwerten $\mathbf{m}_{k,s}(t)$ und Kovarianzmatritzen $\mathbf{\Sigma}_{k,s}(t)$. Im Folgenden wird angenommen im Zeitschritt t zu sein, daher wird der Zeitschritt bei den lokalen Pixelstatistiken $\mathbf{m}_{k,s}(t)$ und $\mathbf{\Sigma}_{k,s}(t)$ weggelassen.

2. Schritt – Anpassen der Modellparameter: Im zweiten Schritt werden die gegebenen a-priori Modellparameter $\mathbf{\Phi}$ durch eine Maximum-A-Posteriori (MAP) Schätzung verfeinert,

indem die Modellkurve an das Eingabebild $\mathbf{I}^*$ angepasst wird. Dies geschieht auf Basis der Maximierung der Wahrscheinlichkeit $p(\mathbf{\Phi}|\mathbf{I}^*)$. Diese Wahrscheinlichkeit ist an der Position maximal, an welcher die Zuordnung der Pixel zur entsprechenden Kurvenseite und die entsprechend berechnete Statistik am besten passen. Dieses Prinzip wird in Abbildung 3.11 gezeigt. Der Output des CCD-Algorithmus beinhaltet den neu geschätzten Mittelwert $\mathbf{m_\Phi}$ und die kor-

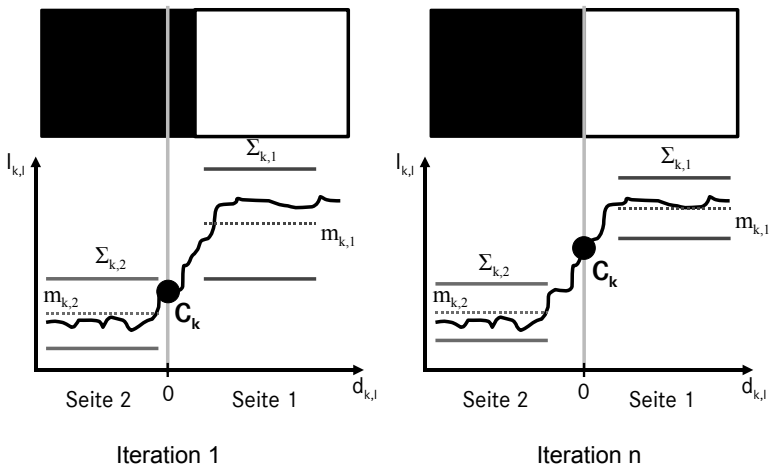

Abbildung 3.11: Das Prinzip des CCD-Algorithmus bei einem Grauwertbild (schematisch dargestellt als schwarz-weiß Übergang). Anpassen der Segmentierungskurve (senkrechte Linie) an die Pixelwerte des Eingabebildes. Abhängig von der Segmentierungskurve wird Mittelwert (gestrichelte Linie) und Standardabweichung (waagerechte Linie) für jede Seite berechnet und die Position der Kurve so angepasst, dass die Pixelstatistiken auf beiden Kurvenseiten möglichst unterschiedlich sind.

respondierende Kovarianzmatrix $\mathbf{\Sigma_\Phi}$. Zur Anpassung des Modellparametersatzes $\mathbf{\Phi}$, welcher durch Mittelwert $\mathbf{m_\Phi}$ und Kovarianzmatrix $\mathbf{\Sigma_\Phi}$ approximiert wird, wird die Wahrscheinlichkeit

$$p(\mathbf{\Phi}|\mathbf{I}^*) \quad = \quad p(\mathbf{I}^*|\mathbf{\Phi}) \cdot p(\mathbf{\Phi}) \tag{3.30}$$

durch eine Maximum-A-Posteriori-Schätzung maximiert. Da die aktuellen Verteilungen der Wahrscheinlichkeiten $p(\mathbf{\Phi})$ und $p(\mathbf{I}^*|\mathbf{\Phi})$ unbekannt sind, werden diese durch folgende Gaußverteilungen approximiert:

$$p(\mathbf{\Phi}) \quad \approx \quad p(\mathbf{\Phi}|\widehat{\mathbf{m}}_\mathbf{\Phi}, \widehat{\mathbf{\Sigma}}_\mathbf{\Phi}) \tag{3.31}$$

$$p(\mathbf{I}^*|\mathbf{\Phi}) \quad \approx \quad p(\mathbf{I}^*|S(\mathbf{m_\Phi}, \mathbf{\Sigma_\Phi})). \tag{3.32}$$

Der Ausdruck $p\left(\mathbf{\Phi}|\widehat{\mathbf{m}}_\mathbf{\Phi}, \widehat{\mathbf{\Sigma}}_\mathbf{\Phi}\right)$, mit

$$p\left(\mathbf{\Phi}|\widehat{\mathbf{m}}_\mathbf{\Phi}, \widehat{\mathbf{\Sigma}}_\mathbf{\Phi}\right) \quad = \quad \frac{1}{(2\pi)^{\frac{N_{\mathbf{m}_\mathbf{\Phi}}}{2}} \cdot \sqrt{\det \widehat{\mathbf{\Sigma}}_\mathbf{\Phi}}} \cdot e^{-\frac{(\mathbf{m}_\mathbf{\Phi} - \widehat{\mathbf{m}}_\mathbf{\Phi})^2}{\widehat{\mathbf{\Sigma}}^2_\mathbf{\Phi}}}, \tag{3.33}$$

beschreibt eine a-priori Gaußverteilung der Modellparameter $\mathbf{\Phi}$, welche durch Mittelwertvektor $\widehat{\mathbf{m}}_\mathbf{\Phi}$ und Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_\mathbf{\Phi}$ definiert ist. Dabei stellt $N_{\mathbf{m}_\mathbf{\Phi}}$ die Anzahl der Dimensionen

von $\widehat{\mathbf{m}}_{\mathbf{\Phi}}$ dar. Der Ausdruck $p\left(\mathbf{I}^*|S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right)$ in Gleichung (3.32) beschreibt die Wahrscheinlichkeit, wie gut die Pixelwerte $\mathbf{I}^*_{k,l}$ entlang aller Liniensegmente zur zuvor berechneten lokalen Pixelstatistik $S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ passen und wird als Produkt von Gaußverteilungen modelliert:

$$\begin{aligned} p\left(\mathbf{I}^*|S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right) &= p(\mathbf{I}^*_{k,l}|\mathbf{m}_{k,l}, \mathbf{\Sigma}_{k,l}) \\ &= \prod_{k,l} \frac{1}{(2\pi)^{\frac{N_{pix}}{2}} \cdot \sqrt{\det \mathbf{\Sigma}_{k,l}}} \cdot e^{-\frac{(\mathbf{I}^*_{k,l} - \mathbf{m}_{k,l})^2}{2\mathbf{\Sigma}^2_{k,l}}}, \end{aligned} \tag{3.34}$$

dabei beschreibt l den Pixelindex auf dem Liniensegment k und N_{pix} die Anzahl der Dimensionen des Pixelwertes. Mittelwert $\mathbf{m}_{k,l}$ und Kovarianzmatrix $\mathbf{\Sigma}_{k,l}$ werden mit

$$\mathbf{m}_{k,l} = a_{l,1} \cdot \mathbf{m}_{k,1} + a_{l,2} \cdot \mathbf{m}_{k,2} \tag{3.35}$$

$$\mathbf{\Sigma}_{k,l} = a_{l,1} \cdot \mathbf{\Sigma}_{k,1} + a_{l,2} \cdot \mathbf{\Sigma}_{k,2}. \tag{3.36}$$

berechnet. Unter Nutzung der Gleichungen (3.33) und (3.34) kann die Anpassung der Modellparameter $\mathbf{\Phi}$ wie folgt beschrieben werden:

$$\hat{\mathbf{\Phi}} = \arg\max_{\mathbf{\Phi}} F(\mathbf{\Phi}) \quad \text{mit} \tag{3.37}$$

$$F(\mathbf{\Phi}) = p\left(\mathbf{I}^*|S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right) \cdot p\left(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}\right). \tag{3.38}$$

Um numerische Probleme zu vermeiden, wird durch Logarithmierung aus dem Maximierungsproblem (Gleichung (3.37)) ein Minimierungsproblem definiert:

$$\hat{\mathbf{\Phi}} = \arg\min_{\mathbf{\Phi}} X(\mathbf{\Phi}) \quad \text{mit} \tag{3.39}$$

$$X = -2\ln\left[p\left(\mathbf{I}^*|S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right) \cdot p\left(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}\right)\right]. \tag{3.40}$$

Die neue Zielfunktion wird in zwei Teile aufgespalten

$$X = X_1 + X_2 \text{ , mit} \tag{3.41}$$

$$X_1 = -2\ln p\left(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}\right) = \ln\left(2\pi \det \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}\right) + \frac{\left(\mathbf{m}_{\mathbf{\Phi}} - \widehat{\mathbf{m}}_{\mathbf{\Phi}}\right)^2}{\widehat{\mathbf{\Sigma}}^2_{\mathbf{\Phi}}} \tag{3.42}$$

$$X_2 = -2\ln p\left(\mathbf{I}^*|S(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})\right) = \sum_{k,l} X_{k,l\ 2} \text{ , mit} \tag{3.43}$$

$$X_{k,l\ 2} = \ln 2\pi + 2\ln \mathbf{\Sigma}_{k,l} + \frac{(\mathbf{I}^*_{k,l} - \mathbf{m}_{k,l})^2}{\mathbf{\Sigma}^2_{k,l}} \tag{3.44}$$

um die Berechnung von Jacobi-Matrix $\mathbf{J}$ und Hesse-Matrix $\mathbf{H}$ der Zielfunktion zu vereinfachen. Da bei der Ableitung der quadratischen Variablen im Nenner der Faktor 2 entstanden ist, wird Gleichung (3.40) mit dem Faktor -2 multipliziert. Mit Hilfe des Logarithmus $\ln(x)$ wird die Multiplikation in Gleichung (3.34) in eine Addition umgewandelt. Auf diese Weise werden numerische Probleme vermieden. Das Anpassen der Modellparameter erfolgt durch ein Newton-Raphson-Optimierungsverfahren (Weisstein, 2008), welches die Jacobi-Matrix $\mathbf{J}$ und die Hesse-Matrix $\mathbf{H}$ der Zielfunktion voraussetzt. Die Anpassung des Mittelwertvektors $\mathbf{m}_{\mathbf{\Phi}}{}^{(iter)}$ und der

Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{\Phi}}^{(iter)}$ im Iterationsschritt $(iter)$ wird folgendermaßen durchgeführt:

$$\mathbf{m}_{\mathbf{\Phi}}{}^{(iter)} = \mathbf{m}_{\mathbf{\Phi}}{}^{(iter-1)} - \left(\mathbf{H}^{(iter)}\right)^{-1} \cdot \mathbf{J}^{(iter)^T} \tag{3.45}$$

$$\mathbf{\Sigma}_{\mathbf{\Phi}}^{(iter)} = c_2 \cdot \mathbf{\Sigma}_{\mathbf{\Phi}}^{(iter-1)} + (1 - c_2) \cdot 2 \cdot \left(\mathbf{H}^{(iter)}\right)^{-1}. \tag{3.46}$$

Dabei steht $\mathbf{m}_{\mathbf{\Phi}}{}^{(iter-1)}$ für den Mittelwertvektor im Iterationsschritt $(iter - 1)$. Weiterhin ist $\mathbf{H}^{(iter)}$ die Hesse-Matrix und $\mathbf{J}^{(iter)}$ der Jacobi-Vektor der Zielfunktion im Iterationsschritt $(iter)$. Der Parameter c_2 bewirkt ein konstantes Verhalten über mehrere Iterationsschritte. Der Autor dieser Arbeit empfiehlt $c_2 = 0.2$. Nachdem der Optimierungsschritt berechnet wurde, beginnt der CCD-Algorithmus wieder mit dem ersten Schritt, der Berechnung der lokalen Pixelstatistiken.

3D-Erweiterung des CCD-Algorithmus

Eine 3D-Erweiterung des CCD-Algorithmus zum multiokularen CCD-Algorithmus (MOCCD) wird von Krüger u. Ellenrieder (2005) beschrieben. Grundlage der Erweiterung ist ein 3D-Kurvenmodell, was an die Bilder eines Mehrkamerasystems angepasst wird. Das Kamerasystem besteht dabei aus N_c kalibrierten Kameras in einem definierten Weltkoordinatensystem W. Von Krüger u. Ellenrieder (2005) werden nach der Projektion des 3D-Kurvenmodells in die Pixelkoordinatensysteme der Kameras die projizierten 2D-Kurven mit Hilfe des CCD-Algorithmus individuell angepasst. Das Update des 3D-Modells wird durch eine 2D-3D-Pose-Estimation der im 2D individuell angepassten Kurven berechnet. In dieser Arbeit wird der Ansatz von Krüger u. Ellenrieder (2005) so erweitert, dass eine direkte 3D-Pose-Estimation durchgeführt wird. Die Erweiterung basiert auf der Integration aller N_c Kamerabilder in die zur Anpassung des Modellparametervektors $\mathbf{\Phi}$ maximierte Wahrscheinlichkeit $p(\mathbf{\Phi}|\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_\mathrm{c}}^*)$.

Die Eingabeparameter des MOCCD-Algorithmus sind N_c Bilder, welche von einem kalibrierten Mehrkamerasystem geliefert werden, und ein parametrisches 3D-Kurvenmodell (Abschnitt 3.2). Wie beim CCD-Algorithmus (Abschnitt 3.3.1) wird die Kurve an erweiterte Eingabebilder angepasst, welche mit der folgenden Formel erzeugt werden.

$$\mathbf{I}_{c,t}^* = \mathbf{I}_{c,t} + \lambda \left|\mathbf{I}_{c,t} - \mathbf{I}_{c,(t-1)}\right| \quad \text{mit } c \in \{1, \ldots, N_\mathrm{c}\} \tag{3.47}$$

Dabei beschreibt $\mathbf{I}_{c,t}$ das von der Kamera c gelieferte Bild zum Zeitschritt t und $\left|\mathbf{I}_{c,t} - \mathbf{I}_{c,(t-1)}\right|$ das Bild der absoluten Differenzen, welches aus dem aktuellen und dem vorherigen Bild der Kamera c berechnet wird. Der Faktor λ beschreibt dabei den Einfluss des Bildes der absoluten Differenzen. Es wurde mit verschiedenen Werten von λ experimentiert, dabei wurde λ im Bereich von $0.5 \leq \lambda \leq 3$ variiert und anschließend die Modellanpassung mit dem MOCCD-Algorithmus durchgeführt. Als Ergebnis der Experimente zeigte sich, dass die besten Ergebnisse der Modellanpassung mit $\lambda = 1$ erreicht werden. Durch die erweiterten Eingabebilder werden Pixelwerte in Bereichen mit Bewegung erhöht, was zu einer robusteren Segmentierung führt, da die Trennung der Statistiken eindeutiger wird. Abbildung 3.8 zeigt die Entstehungsschritte des erweiterten Eingabebildes für eine Kamera. Im Folgenden wird angenommen im Zeitschritt t zu sein, daher wird aufgrund der Übersichtlichkeit beim erweiterten Eingabebild $\mathbf{I}_{c,t}^*$ der Zeitschritt t weggelassen.

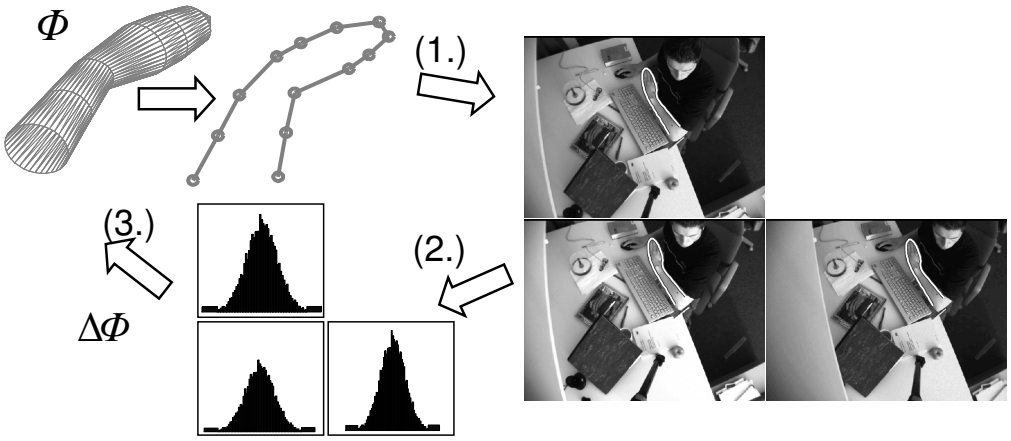

Abbildung 3.12: Ablaufplan des MOCCD-Algorithmus. Im ersten Schritt wird ein 3D-Konturmodell in die Pixelkoordinatensysteme der verwendeten N_c Kameras projiziert. Danach werden im zweiten Schritt entlang der projizierten Modellpunkte die Grauwertstatistiken berechnet. Mit diesen Statistiken und der Zielfunktion des MOCCD-Algorithmus wird im 3. Schritt durch einen Newton-Raphson-Optimierungsschritt die Veränderung $\Delta\mathbf{\Phi}$ des Parametervektors $\mathbf{\Phi}$ berechnet und dieser angepasst.

Wie beim CCD-Algorithmus wird der Modellparametervektor $\mathbf{\Phi}$ durch eine Gaußverteilung mit Mittelwert und Kovarianzmatrix approximiert. Dabei beschreibt der Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}$ die 3D-Modellkurve und die Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}$ die Unsicherheit entlang dieser. Vor der ersten Iteration wird der MOCCD-Algorithmus initialisiert, indem die optimierten Verteilungsparameter $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf die gegebenen a-priori Gaußverteilungsparameter $(\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}})$ gesetzt werden. Der MOCCD-Algorithmus beruht auf folgenden drei Schritten (Abbildung 3.12), welche so lange wiederholt werden, bis die Änderungen an den Verteilungsparametern $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ unter einer Schwelle liegen oder eine bestimmte Anzahl von Iterationen erreicht ist.

1. Schritt – Projektion des 3D-Konturmodells: In diesem Schritt wird das 3D-Konturmodell in die 2D-Pixelkoordinatensysteme der verwendeten Kameras c, $c \in \{1, \ldots, N_c\}$, projiziert. Falls als 3D-Modell ein Volumenmodell eingesetzt wird, muss aus dem 3D-Volumenmodell die für das Kamerasystem sichtbare 3D-Silhouette extrahiert werden. Für das 3D-Hand-Unterarm-Modell ist dies der Fall, daher wird in Abschnitt 3.2.1 ein Verfahren zur Extraktion der sichtbaren 3D-Kontur für ein 3D-Volumenmodell des menschlichen Unterarms und der Hand (Abbildung 3.5) beschrieben. Diese sichtbare 3D-Kontur ist abhängig von der Blickrichtung der Kamera. Bei der Projektion der berechneten 3D-Kontur ist es wichtig, dass die Kameras genau kalibriert sind, sodass die projizierte Kurve auch zur Objektkurve im Bild passt. Die Qualität der Kamerakalibrierung (Anhang A.1.4) wirkt sich dabei direkt auf die Genauigkeit des MOCCD-Algorithmus aus.

2. Schritt – Berechnung der lokalen Pixelstatistiken in allen N_c Kamerabildern: Für alle N_c Eingabebilder $\mathbf{I}_c^*$ werden die lokalen Pixelstatistiken $S_c(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf Liniensegmenten, welche senkrecht zur projizierten 3D-Modellkurve angeordnet sind, berechnet. Durch die Projektion ist eine 2D-Kurve entstanden und die Berechnung der Pixelstatistiken kann iden-

tisch zum ersten Schritt des CCD-Algorithmus (Abschnitt 3.3.1) durchgeführt werden. Neu ist, dass es N_c lokale Pixelstatistiken $S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)$ gibt. Diese werden nun zur Anpassung der 3D-Modellparameter verwendet.

3. Schritt – Anpassen der Modellparameter: Mit den lokalen Pixelstatistiken $S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)$ aus allen verwendeten Eingabebildern wird nun die Anpassung der Modellparameter durchgeführt. Der Output des MOCCD-Algorithmus beinhaltet den neu geschätzten Mittelwert $\mathbf{m}_\Phi$ und die korrespondierende Kovarianzmatrix $\mathbf{\Sigma}_\Phi$. Vor der ersten Iteration wird der MOCCD-Algorithmus initialisiert, indem die optimierten Parameter $(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)$ auf die gegebenen a-priori Gaußverteilungsparameter $(\widehat{\mathbf{m}}_\Phi, \widehat{\mathbf{\Sigma}}_\Phi)$ gesetzt werden.

Zur Anpassung des Modellparametersatzes $\mathbf{\Phi}$, welcher durch Mittelwert $\mathbf{m}_\Phi$ und Kovarianzmatrix $\mathbf{\Sigma}_\Phi$ approximiert wird, wird die Wahrscheinlichkeit

$$p(\mathbf{\Phi}|\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_c}^*) \;=\; p(\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_c}^*|\mathbf{\Phi}) \cdot p(\mathbf{\Phi}) \tag{3.48}$$

durch eine Maximum-A-Posteriori-Schätzung maximiert. Wenn man in Gleichung (3.48) annimmt, dass die verschiedenen Eingabebilder $\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_c}^*$ unabhängige Zufallsvariablen sind, kann man die Gleichung nach MacKay (2003) folgendermaßen umschreiben:

$$p(\mathbf{\Phi}|\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_c}^*) = \left[\prod_{c=1}^{N_c} p(\mathbf{I}_c^*|\mathbf{\Phi})\right] \cdot p(\mathbf{\Phi}). \tag{3.49}$$

Da wie beim CCD-Algorithmus (Hanek, 2004) die aktuellen Verteilungen der Wahrscheinlichkeiten $p(\mathbf{\Phi})$ und $p(\mathbf{I}_c^*|\mathbf{\Phi})$ unbekannt sind, werden diese durch Gaußverteilungen approximiert:

$$p(\mathbf{\Phi}) \;\approx\; p(\mathbf{\Phi}|\widehat{\mathbf{m}}_\Phi, \widehat{\mathbf{\Sigma}}_\Phi) \tag{3.50}$$

$$p(\mathbf{I}_c^*|\mathbf{\Phi}) \;\approx\; p(\mathbf{I}_c^*|S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)), \quad \forall\; c \in \{1, \ldots, N_c\} \tag{3.51}$$

Die finale Maximum-A-Posteriori-Schätzung basiert also auf der Wahrscheinlichkeit:

$$p(\mathbf{\Phi}|\mathbf{I}_1^*, \ldots, \mathbf{I}_{N_c}^*) = \left[\prod_{c=1}^{N_c} p(\mathbf{I}_c^*|S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi))\right] \cdot p(\mathbf{\Phi}|\widehat{\mathbf{m}}_\Phi, \widehat{\mathbf{\Sigma}}_\Phi). \tag{3.52}$$

Dabei repräsentiert $S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)$ die lokalen Pixelstatistiken der Kamera c in der Nähe der projizierten Kurve im Eingabebild $\mathbf{I}_c^*$. Wie im zweiten Schritt des CCD-Algorithmus (Abschnitt 3.3.1) wird durch Logarithmierung aus dem Maximierungsproblem (Gleichung (3.52)) ein Minimierungsproblem definiert, um numerische Probleme zu vermeiden. Dieses Minimierungsproblem wird nun durch einen Schritt einer Newton-Raphson-Optimierung (Weisstein, 2008) minimiert und danach beginnt die nächste Iteration des MOCCD-Algorithmus wieder mit Schritt 1.

Die Ausgabe $(\mathbf{m}_\Phi^{(n)}(t), \mathbf{\Sigma}_\Phi^{(n)}(t))$ eines MOCCD-Algorithmus i zum Zeitschritt t besteht aus den optimierten Parametern Mittelwertvektor $\mathbf{m}_\Phi^{(n)}(t)$ und Kovarianzmatrix $\mathbf{\Sigma}_\Phi^{(n)}(t)$. Der Mittelwertvektor $\mathbf{m}_\Phi^{(n)}(t)$ beschreibt dabei direkt die Parameter des verwendeten Modells (Abschnitt 3.2). Eine anschauliche Erläuterung des MOCCD-Algorithmus wird durch folgende Interpretation gegeben. Die gaußschen Wahrscheinlichkeitsdichten $p\left(\mathbf{I}^*|S_c(\mathbf{m}_\Phi, \mathbf{\Sigma}_\Phi)\right)$, $c \in \{1, \ldots, N_c\}$, sind viele lokale Kantendetektoren, welche in den Eingabebilder $\mathbf{I}_c^*$ senkrecht zur projizierten 3D-Modellkurve die stärkste Kante finden. Durch den Bayes'schen Ansatz können diese Kanten, welche man als Statistikunterschiede der Pixelwerte interpretiert, auch bei starkem Rauschen, geringem Kontrast oder Shading-Kanten gefunden werden. Die Adaption des 3D-Modells an die N_c Bilder erfolgt durch eine implizite Triangulation.

3.4 Fusion und Verifikation

In der Fusions- und Verifikationskomponente werden die drei geschätzten 3D-Pose-Vektoren $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$ und $(\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$ der drei MOCCD-Algorithmen bewertet, siehe Systemmodell in Abbildung 3.1. Dadurch können relativ ungenaue Pose-Schätzungen einzelner MOCCD-Algorithmen erkannt und ausgeschlossen werden. Ungenaue Pose-Schätzungen können durch Fehlprädiktionen des Kalman-Filters oder durch den MOCCD-Algorithmus selbst entstehen, da die Optimierung z.B. in einem unvorteilhaften lokalen Minimum festsitzt. Durch den Ausschluss ungenauer Pose-Schätzungen wird die Qualität des entwickelten Systems deutlich gesteigert.

Die Fusion der drei geschätzten 3D-Pose-Vektoren $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$ und $(\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$ ist so realisiert, dass die Beste der drei Pose-Schätzungen ausgewählt wird. Es wird also ein „*Winner-Takes-All*“ (WTA) Ansatz realisiert. Die Auswahl der besten Pose-Schätzungen basiert auf zwei Ähnlichkeitsmaßen: (i) Ähnlichkeit der Modellorientierung $\mathcal{MO}(\mathbf{\Phi}, t)$ und (ii) Ähnlichkeit der Erscheinung $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-n))$.

Ähnlichkeit der Modellorientierung $\mathcal{MO}(\mathbf{\Phi}(t))$: Zur Berechnung des Ähnlichkeitsmaßes $\mathcal{MO}(\mathbf{\Phi}(t))$ für den Parametervektor $\mathbf{\Phi}(t)$ wird die Ähnlichkeit der Modellorientierung mit der Kantenorientierung in den Kamerabildern zum Zeitschritt t bestimmt. Dazu wird das verwendete 3D-Konturmodell (Abschnitt 3.2) für jede Kamera c in deren zugehöriges Pixelkoordinatensystem P_c projiziert. Dabei wird die selbe Kurvenfunktion wie beim MOCCD-Algorithmus (Abschnitt 3.3.1) genutzt, d.h. die Kurve ist abhängig vom Parametervektor $\mathbf{\Phi}(t)$ zum Zeitschritt t. Für jede Kamera c werden $(K+1)$ Punkte ${}^{P_c}\mathbf{c}_k$ in das Pixelkoordinatensystem P_c der Kamera projiziert. Für jeden projizierten Modellpunkt ${}^{P_c}\mathbf{c}_k$ wird mit Hilfe der Sobel-Operatoren

$$\mathbf{S}_u = \frac{1}{8}\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \quad \text{und} \quad \mathbf{S}_v = \frac{1}{8}\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{3.53}$$

der Gradient in u- und v-Richtung G_u und G_v berechnet. Die Orientierung $O({}^{P_c}\mathbf{c}_k)$ im Punkt ${}^{P_c}\mathbf{c}_k$ berechnet sich mit:

$$O({}^{P_c}\mathbf{c}_k) = \arctan\left(\frac{G_v({}^{P_c}\mathbf{c}_k)}{G_u({}^{P_c}\mathbf{c}_k)}\right) \quad mod \quad 180. \tag{3.54}$$

Dabei steht $G_v({}^{P_c}\mathbf{c}_k)$ für den Gradienten in v-Richtung im Punkt ${}^{P_c}\mathbf{c}_k$ der mit dem Operator $\mathbf{S}_v$ berechnet wurde, analog gilt dies für $G_u({}^{P_c}\mathbf{c}_k)$. Es ist zu beachten, dass zwischen Kantenrichtung (Intervall $[0 \ldots 360]^\circ$) und Kantenorientierung (Intervall $[0 \ldots 180]^\circ$) unterschieden wird.

Für jeden projizierten Punkt ${}^{P_c}\mathbf{c}_k$ wird die Orientierung im Kamerabild $O({}^{P_c}\mathbf{c}_k)$ mit der Orientierung des Modells $M({}^{P_c}\mathbf{c}_k)$ verglichen. Es werden alle Punkte gezählt für die gilt:

$$\left(|O({}^{P_c}\mathbf{c}_k) - M({}^{P_c}\mathbf{c}_k)|\right) \leq \Theta_{WinkelDiff}. \tag{3.55}$$

Durch Division der Summe der gezählten Punkte in allen Kamerabildern, durch alle betrachteten Punkte erhält man das Ähnlichkeitsmaß $\mathcal{MO}(\mathbf{\Phi}(t))$. Da die verwendeten Konturmodelle

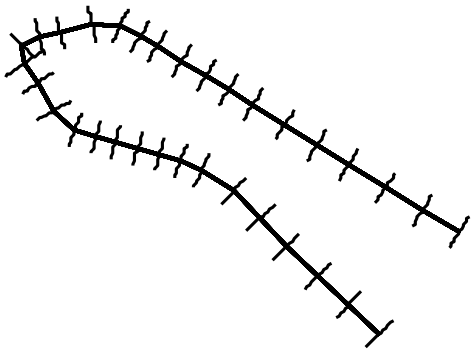

Abbildung 3.13: Projiziertes Kurvenmodell der Hand-Unterarm-Kontur zur Berechnung des Ähnlichkeitsmaßes $\mathcal{MO}(\mathbf{\Phi}(t))$. Die Orientierung im Bild wird nicht nur mit der Orientierung des Modells in den projizierten Modellpunkten verglichen, sondern mit allen Punkten auf einer kurzen Senkrechten.

grob sind, wird als Erweiterung nicht nur für den projizierten Modellpunkt ${}^{P_c}\mathbf{c}_k$ die Orientierung im Bild mit der Orientierung des Modells verglichen, sondern für alle Punkte auf einer kurzen Senkrechten (8 Pixel lang) im Punkt ${}^{P_c}\mathbf{c}_k$ (Abbildung 3.13).

Ähnlichkeit der Erscheinung $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-n))$: Die Ähnlichkeit der Erscheinung oder Ähnlichkeit des Aussehens $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-n))$ wird für eine 3D-Pose $\mathbf{\Phi}(t)$ zum Zeitschritt t durch den Vergleich der aktuellen Erscheinung, also des Bildinhalts zum Zeitschritt t, mit der vorherigen Objekterscheinung zum Zeitschritt $(t-n)$ berechnet. Zu jedem Zeitschritt t ist die 3D-Objektpose $\mathbf{\Phi}(t)$ bekannt und die Kameras sind notwendigerweise kalibriert, daher kann der Bereich der ins Bild projizierten 3D-Modellkurve oder ein Teil dieses Bereichs ausgeschnitten werden. Dieser Ausschnitt wird durch Rotation, Translation und Interpolation in eine definierte Form gebracht (Abbildung 3.14), sodass die Erscheinung zu unterschiedlichen Zeitschritten mit unterschiedlichen Objekttiefen auch miteinander verglichen werden kann. Die ausgeschnittenen und normierten Bereiche werden Referenz-Templates genannt und für die letzten 20 Zeitschritte gespeichert (Abbildung 3.14). Dabei besteht ein Referenz-Template aus N_c Bildbereichen und die Ähnlichkeit zu einem anderen Referenz-Template wird durch normierte Kreuzkorrelation (Jähne, 2002) berechnet. Die miteinander verglichenen Referenz-Templates bestehen aus N_c normalisierten Bildbereichen, dadurch erhält man N_c Ergebnisse der normierten Kreuzkorrelation. Als endgültiges Maß wird immer das schlechteste der N_c Kreuzkorrelationsergebnisse verwendet. Diesem Ähnlichkeitsmaß liegt die Annahme zugrunde, dass die Erscheinung des in den Bildern verfolgten Objekts sich in kleinen Zeiträumen nur geringfügig ändert.

Auswahl der besten 3D-Pose-Schätzung (WTA) Mit den zwei beschriebenen Ähnlichkeitsmaßen werden fünf Kriterien zur Bestimmung der besten 3D-Pose-Schätzung zum Zeitschritt t berechnet:

1. Zeitliche Veränderung der Ähnlichkeit der Modellorientierung:
   $\mathcal{MO}(\mathbf{\Phi}(t)) - \mathcal{MO}(\mathbf{\Phi}(t-1))$
2. Aktuelle Ähnlichkeit der Modellorientierung: $\mathcal{MO}(\mathbf{\Phi}, t)$

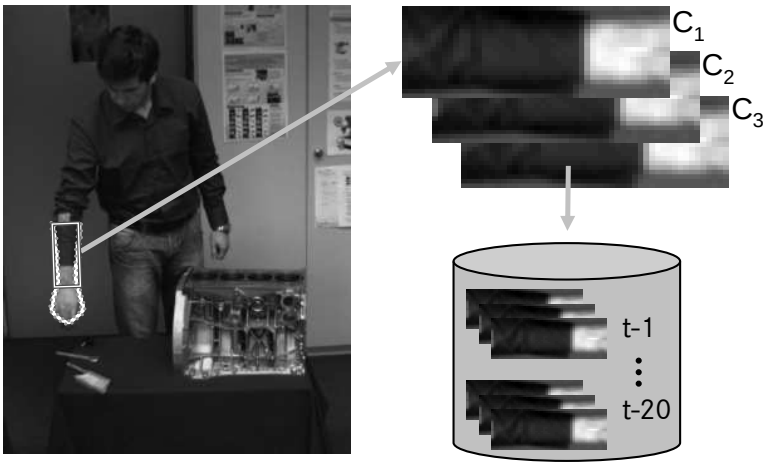

Abbildung 3.14: Ausschneiden der Referenz-Templates für das Beispiel Hand-Unterarm. Mit der in die Kamerabilder projizierten 3D-Modellkurve werden Bildbereiche ausgeschnitten, durch Normierung in eine definierte Form gebracht und in eine Datenbank einsortiert.

3. Zeitliche Veränderung der Ähnlichkeit der Erscheinung:
   $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-1)) - \mathcal{ER}(\mathbf{\Phi}(t-1), \mathbf{\Phi}(t-2))$
4. Ähnlichkeit der aktuellen Erscheinung im Vergleich zum Zeitschritt $(t-1)$:
   $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-1))$
5. Ähnlichkeit der aktuellen Erscheinung im Vergleich zum Zeitschritt $(t-2)$:
   $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-2))$

Als bestes Messergebnis zum Zeitschritt t wird aus allen 3D-Pose-Schätzungen $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$ und $(\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$ diejenige ausgewählt, die in mindestens drei der fünf Kriterien besser ist als alle anderen 3D-Pose-Schätzungen.

Nach der Auswahl der besten 3D-Pose-Schätzung, muss diese noch verifiziert werden. Dies bedeutet, es müssen Bedingungen erfüllt sein, die garantieren, dass das kamerabasierte Trackingsystem das in den Bildern verfolgte Objekt nicht verloren hat. Dazu wird für jedes der fünf Kriterien ein Schwellwert genutzt, welcher erfüllt sein muss. Wenn die beste 3D-Pose-Schätzung die Verifikation besteht, werden die zur 3D-Pose-Schätzung zugehörigen Referenz-Templates in die Datenbank eingeordnet. Falls die Verifikation nicht bestanden wurde, wird angenommen, dass das System das Objekt verloren hat. Nun gibt es zwei Systemvarianten (siehe Abbildung 3.1), die eine hält mit einer Fehlermeldung an und die andere wendet ein Modul zur Reinitialisierung (Abschnitt 3.6) an.

3.5 Kalman-Filter Varianten

Wie im Systemmodell (Abbildung 3.1) zu sehen ist, gehört zu jedem MOCCD-Algorithmus ein Kalman-Filter zur Prädiktion des Parametervektors $\mathbf{\Phi}$ vom Zeitschritt t zum Zeitschritt $t+\Delta t$. Das Kalman-Filter ist ein etabliertes Verfahren zur Zustandsschätzung, die Grundlagen der Zustandsschätzung und weiterführende Literatur werden im Anhang A.2 aufgezeigt.

Die bestbewertete 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ zum Zeitschritt t ist der Input für jeden der

verwendeten N Zustandsschätzer. Mit Hilfe von $\mathbf{\Phi}(t)$ wird in jedem Kalman-Filter für dessen zugehörigen MOCCD-Algorithmus die Prädiktion für den Zeitschritt $t + \Delta t$ berechnet, d.h. drei Prädiktionen $\widehat{\mathbf{\Phi}}_1(t + \Delta t)$, $\widehat{\mathbf{\Phi}}_2(t + \Delta t)$ und $\widehat{\mathbf{\Phi}}_3(t + \Delta t)$. Da unterschiedliche Prädiktionen, also unterschiedliche a-priori Initialisierungen, für die verwendeten MOCCD-Algorithmen erzeugt werden sollen, kommen drei verschiedene Zustandsmodelle in den Zustandsschätzern zum Einsatz: (i) konstante Position, (ii) Bewegung mit konstanter Geschwindigkeit und (iii) konstant beschleunigte Bewegung. Jedes der Zustandsmodelle wird nun näher erläutert.

1. Zustandsmodell – keine Bewegung: In diesem Zustandsmodell wird angenommen, dass die geschätzten Parameter sich nicht verändern. Der Zustandsvektor hat folgende Form:

$$\mathbf{x} = \begin{pmatrix} \mathbf{\Phi} \end{pmatrix} \tag{3.56}$$

mit $\mathbf{\Phi}$ als Parametervektor des verwendeten 3D-Modells (Abschnitt 3.2). Die Systemmatrix zur Prädiktion (Gleichung (A.12)) hat folgende Form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} \end{pmatrix}. \tag{3.57}$$

Die Messmatrix (Gleichung (A.13)) ergibt sich zu:

$$\mathbf{H} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} \end{pmatrix}. \tag{3.58}$$

Mit $\mathbf{E}_{\mathbf{\Phi}}$ einer Einheitsmatrix der Länge des Parametervektors $\mathbf{\Phi}$.

2. Zustandsmodell – Bewegung mit konstanter Geschwindigkeit: In diesem Zustandsmodell wird angenommen, dass die geschätzten Parameter sich mit konstanter Geschwindigkeit bewegen. Der Zustandsvektor hat folgende Form:

$$\mathbf{x} = \begin{pmatrix} \mathbf{\Phi} \\ \dot{\mathbf{\Phi}} \end{pmatrix} \tag{3.59}$$

mit $\mathbf{\Phi}$ als Parametervektor des verwendeten 3D-Modells (Abschnitt 3.2) und $\dot{\mathbf{\Phi}}$ als erste Ableitung von $\mathbf{\Phi}$ nach der Zeit. In den verwendeten Modellen (Abschnitt 3.2) wird angenommen, dass die im Parametervektor $\mathbf{\Phi}$ enthaltenen Radien über kurze Zeiträume nicht veränderlich sind, daher sind die Radien nicht Bestandteil von $\dot{\mathbf{\Phi}}$. Die Systemmatrix zur Prädiktion (Gleichung (A.12)) hat folgende Form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} & \Delta T \\ 0 & \mathbf{E}_{\dot{\mathbf{\Phi}}} \end{pmatrix} \tag{3.60}$$

dabei beschreibt ΔT das Zeitinkrement, dies entspricht dem Abtastintervall, also dem Inversen der Abtastfrequenz des Kalman-Filters. Die Messmatrix (Gleichung A.13) besteht aus:

$$\mathbf{H} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} & 0 \end{pmatrix}. \tag{3.61}$$

3. Zustandsmodell – konstant beschleunigte Bewegung: Bei diesem Zustandsmodell wird angenommen, dass die geschätzten Parameter sich mit konstanter Beschleunigung bewegen. Der Zustandsvektor hat folgende Form:

$$\mathbf{x} = \begin{pmatrix} \mathbf{\Phi} \\ \dot{\mathbf{\Phi}} \\ \ddot{\mathbf{\Phi}} \end{pmatrix} \tag{3.62}$$

mit $\mathbf{\Phi}$ als Parametervektor des verwendeten 3D-Modells (Abschnitt 3.2), $\dot{\mathbf{\Phi}}$ als erste Ableitung von $\mathbf{\Phi}$ nach der Zeit und $\ddot{\mathbf{\Phi}}$ als zweite Ableitung von $\mathbf{\Phi}$ nach der Zeit, wiederum sind die Radien in den verwendeten Modellen nicht Bestandteil der Ableitung, da diese als konstant über kurze Zeiträume angenommen werden. Die Systemmatrix zur Prädiktion (Gleichung (A.12)) hat folgende Form:

$$\mathbf{A} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} & \Delta T & \frac{1}{2}\Delta T^2 \\ 0 & \mathbf{E}_{\dot{\mathbf{\Phi}}} & \Delta T \\ 0 & 0 & \mathbf{E}_{\ddot{\mathbf{\Phi}}} \end{pmatrix} \tag{3.63}$$

und die Messmatrix (Gleichung (A.13)) besteht aus

$$\mathbf{H} = \begin{pmatrix} \mathbf{E}_{\mathbf{\Phi}} & 0 & 0 \end{pmatrix}. \tag{3.64}$$

3.6 Reinitialisierung

Dieses Modul kann je nach Systemvariante ein- und ausgeschaltet werden und führt zu einer robusteren und zeitlich stabileren Verfolgung in den Bilden des Kamerasystems. Das Reinitialisierungsmodul ist in der Lage, ein verlorenes Objekt wiederfinden oder eine relativ ungenaue 3D-Pose-Schätzung verbessern. Die Reinitialisierung verwendet die in der Fusions- und Verifikationskomponente (Abschnitt 3.4) erzeugte Datenbank des vorherigen Aussehens des in den Bildern verfolgten Objekts (Abbildung 3.14). Die Reinitialisierung ist als Pose-Refinement implementiert, d.h. es muss eine initiale Modellpose $\mathbf{\Phi}$ übergeben werden, welche dann verbessert (engl. *refined*) wird. Dabei wird der letzte 3D-Parametervektor $\mathbf{\Phi}$, welcher die Verifikation bestanden hat, verwendet. Zur Reinitialisierung werden zwei Hypothesen angewandt: (i) Pose-Refinement basierend auf einem Wahrscheinlichkeitsbild und (ii) Pose-Refinement durch Template-Matching mit Referenz-Templates.

Hypothese 1 – Pose-Refinement durch Wahrscheinlichkeitsbild: Der hier verwendete Pose-Refinement Algorithmus nutzt die letzten zehn Referenz-Templates der Masterkamera, um ein sogenanntes Wahrscheinlichkeitsbild (Bradski, 1998) zu berechnen. Als Abschätzung für das Aussehen des Objekts wird aus den letzten zehn Referenz-Templates der Masterkamera ein eindimensionales Histogramm berechnet, welches die Verteilung der Grauwerte des Objekts repräsentiert. Aus dem erstellten Histogramm und dem originalen Bild der Masterkamera wird eine Wahrscheinlichkeitsverteilung für die Ähnlichkeit zur vorherigen Objekterscheinung berechnet. Dazu weist man jedem Pixel im Berechnungsbereich b, abhängig vom Grauwert, die

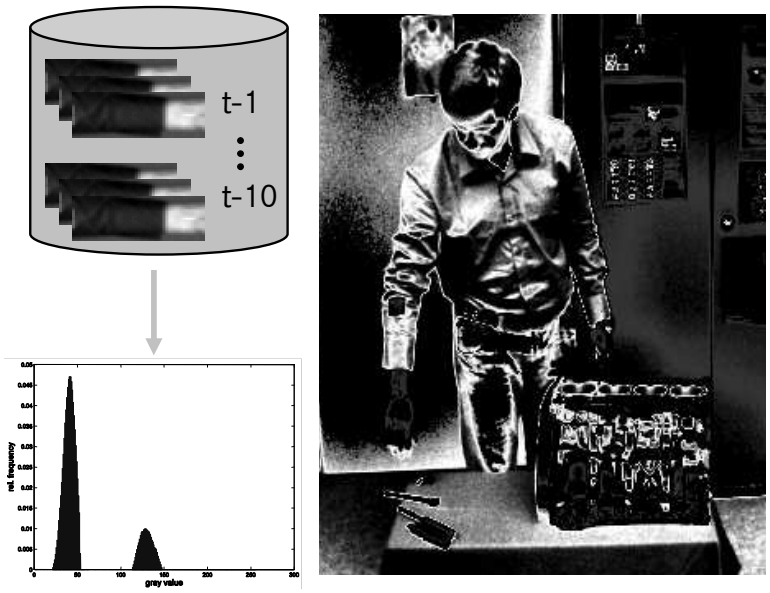

Abbildung 3.15: Aus der Datenbank der letzten zehn Referenz-Templates wird ein Histogramm berechnet, welches die Verteilung der Grauwerte in den Referenz-Templates der Masterkamera beschreibt. Mit den relativen Häufigkeiten im Histogramm wird ein Wahrscheinlichkeitsbild für die Masterkamera bestimmt.

entsprechende relative Häufigkeit aus dem Histogramm zu. Die relative Häufigkeit der Grauwerte wird als Wahrscheinlichkeit interpretiert und das eindimensionale Histogramm wird als *Lookup-Table* verwendet. Bei einem 8-Bit Bild sind 256 unterschiedliche Wahrscheinlichkeitswerte möglich. Das bedeutet, dass in dem entsprechenden Histogramm den Grauwerten von 0 bis 255, relative Häufigkeiten im Intervall $[0, 1]$ zugewiesen werden. Diese relativen Häufigkeiten werden als Wahrscheinlichkeiten interpretiert. Abbildung 3.15 zeigt die Datenbank der letzten zehn Referenz-Templates, das Histogramm der Verteilung der Grauwerte und das Wahrscheinlichkeitsbild für die Masterkamera, welches mit dem eindimensionalen Histogramm berechnet wurde.

Der letzte verifizierte 3D-Parametervektor $\mathbf{\Phi}$ wird dazu verwendet, ein Rechteck in das Wahrscheinlichkeitsbild zu projizieren. Dieses projizierte Rechteck entspricht der Größe des Referenz-Templates für die Masterkamera und ist der Startpunkt für eine lokale Optimierung. Dabei wird ein Suchraum, ein definiertes Gitter im Parameterraum, vollständig durchsucht und die Parameter, welche das lokale Optimum der Zielfunktion beschreiben, als Ergebnis ausgegeben. Durch die lokale Optimierung wird also der beste Mittelpunkt und die beste Orientierung des Rechtecks mit maximaler Summe der Wahrscheinlichkeiten (relative Häufigkeiten) bestimmt. Mit den Parametern des besten Rechtecks wird wieder ein 3D-Parametervektor bestimmt, wobei angenommen wird, dass die Objektentfernung konstant ist. Daher kann die Pixelauflösung an der jeweiligen Tiefe zur Umrechnung der Pixelkoordinaten in Weltkoordinaten verwendet werden. Der berechnete 3D-Parametervektor wird durch Anwendung des MOCCD-Algorithmus nochmals verfeinert und somit die endgültige Ausgabe dieser Hypothese berechnet.

Hypothese 2 – Pose-Refinement durch Template-Matching: Die zweite Hypothese der Reinitialisierung verwendet das Referenz-Template der Masterkamera (Kamera 1) zum letzten verifizierten Zeitschritt, um ein korrelationsbasiertes Template-Matching im aktuellen Bild durchzuführen. Die letzte verifizierte Pose $\mathbf{\Phi}$ wird genutzt, um ein Rechteck in das aktuelle Bild der Masterkamera zu projizieren. Dieses projizierte Rechteck entspricht der Größe des Referenz-Templates für die Masterkamera und ist wiederum der Startpunkt für eine lokale Optimierung. Durch diese wird ein definiertes Gitter im Parameterraum (Mittelpunkt und Orientierung des Rechtecks) abgesucht und der beste Mittelpunkt und die beste Orientierung des Rechtecks mit dem höchster Wert einer normierten Kreuzkorrelation zum Referenz-Template bestimmt. Aus den Gitter-Parametern des besten Rechtecks wird wie bei Hypothese 1 ein 3D-Parametervektor bestimmt und dieser durch Anwendung des MOCCD-Algorithmus nochmals verfeinert und somit die endgültige Ausgabe der zweiten Hypothese berechnet.

Durch die in der Fusions- und Verifikationskomponente (Abschnitt 3.4) beschriebenen Kriterien wird die Beste der beiden Hypothesen bestimmt und falls dieser Parametervektor $\mathbf{\Phi}$ die Verifikation besteht, wird das Tracking mit dieser Hypothese fortgesetzt. Falls die Verifikation nicht bestanden wird, wird das 3D-Trackingsystem angehalten.

3.7 Interaktion im Gesamtsystem

Wie in Abbildung 3.16 zu sehen, ist das Systemmodell als Multi-Hypothesen-Trackingsystem aufgebaut. „Multi“ bedeutet in diesem Fall drei Hypothesen. Die Schätzung der 3D-Pose, also die Anpassung der Modellkontur an die N_c Kamerabilder, wird durch die MOCCD-Algorithmen bestimmt. Die notwendige 3D-Pose-Prädiktion über die Zeit, durch den zum MOCCD-Algorithmus zugehörigen Kalman-Filter (KF) berechnet. Dieser Multi-Hypothesen-Ansatz hat zwei Gründe. Zunächst ist der MOCCD-Algorithmus ein Pose-Refinement Algorithmus, d.h. es werden Initialparameter, welche durch den Kalman-Filter geliefert werden, verfeinert (engl. *refined*). Bei einer ungenauen Prädiktion der Modellparameter durch das Kalman-Filter, d.h. die prädizierte Pose ist zu weit von der tatsächlichen entfernt, kann das Objekt nicht mehr aus den Bilddaten segmentiert werden. Ungenaue Prädiktionen treten häufig bei schnellen Umkehrbewegungen auf, z.B. beim Festziehen einer Schraube, und haben ihre Ursache in einer gewissen Trägheit des Filters. Durch Verwendung mehrerer Kalman-Filter mit unterschiedlichen Bewegungsmodellen (siehe Abschnitt 3.5) können auch solche Bewegungen mit dem System robust verfolgt werden.

Der zweite Grund für die gewählte Designentscheidung ist, dass durch Verwendung mehrerer MOCCD-Algorithmen mit unterschiedlichen initialen a-priori Gaußverteilungen ein robusteres Verhalten erreicht wird. Aufgrund der unterschiedlichen Initialisierung der MOCCD-Algorithmen durch ihre zugehörigen Kalman-Filter, können diese in unterschiedlich gute Minima konvergieren und somit unterschiedlich gute 3D-Pose-Schätzungen liefern. In der Fusions- und Verifikationskomponente (Abschnitt 3.4) werden schlechte 3D-Pose-Schätzungen zur Bestimmung der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ nicht berücksichtigt, was zu einem stabileren Verhalten des Systems führt. Die Verifikation sorgt für einen Selbsttest des Systems, sodass erkannt werden kann, wenn das in den Bildern verfolgte Objekt verloren wurde. Dabei wird durch das Verifikationsmodul eine zeitliche Konsistenz des Objektaussehens überprüft, dies

dient auch einer weiteren Erhöhung der Robustheit, falls dieser Selbsttests nicht bestanden wird, kann ein Modul zur Reinitialisierung das Objekt wiederfinden.

Der folgende Systemablauf gibt einen Überblick über die Interaktion der Teilsysteme, die einzelnen Schritte des Ablaufs sind in Abbildung 3.16 farblich hervorgehoben und wurden durch mögliche Beispielausgaben der Module ergänzt.

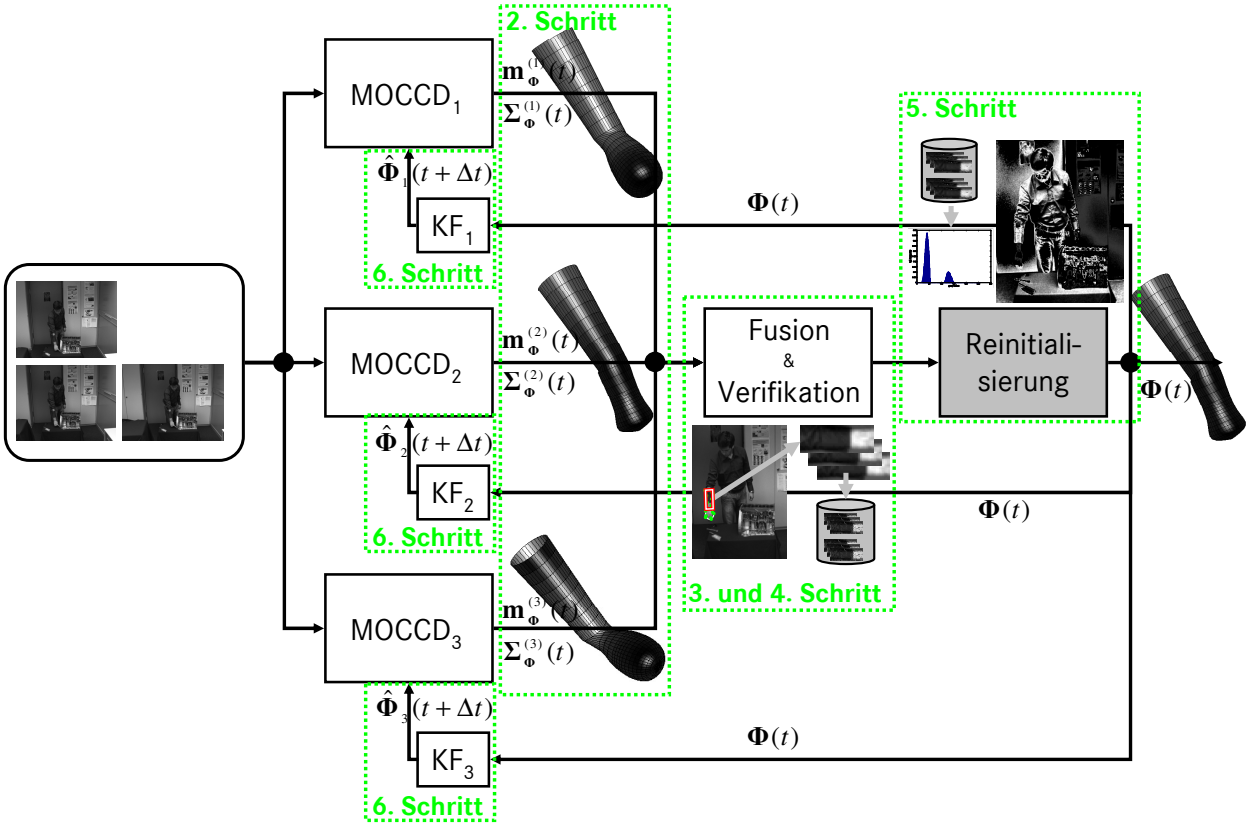

Abbildung 3.16: Struktur des Systemmodells zur 3D-Konturverfolgung. Die einzelnen Schritte des Systemablaufs sind farblich hervorgehoben. Außerdem sind typische Ausgaben einzelner Module gezeigt.

1. Initialisierung
2. Berechne für alle MOCCD-Algorithmen die 3D-Pose-Schätzung zum Zeitschritt t, d.h. $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$, $(\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$
3. Fusion der drei 3D-Pose-Schätzungen aus den drei alle MOCCD-Algorithmen $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t)), (\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t)), (\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$ zur Ergebnismessung des Parametervektors $\mathbf{\Phi}(t)$ zum Zeitschritt t
4. Verifikation der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ (Selbsttest des Systems)
5. Reinitialisierung falls der Selbsttest nicht bestanden wurde, Ausgabe einer neuen Schätzung $\mathbf{\Phi}(t)$ und Verifikation dieser
6. Mit $\mathbf{\Phi}(t)$ wird in jedem Kalman-Filter i, für dessen zugehörigen MOCCD-Algorithmus, die Prädiktion $\widehat{\mathbf{\Phi}}_i(t + \Delta t)$ für den Zeitschritt $t + \Delta t$ berechnet

Bei der Initialisierung wird der Parametervektor $\mathbf{\Phi}(t = 1)$, welcher die 3D-Position und Ori-

entierung des zu verfolgenden Objektes zum Zeitschritt $t = 1$ beschreibt, definiert. Nach der Initialisierung ist das System bereit, das festgelegte Körperteil (Abschnitt 3.2) zu verfolgen. Dies geschieht durch die beim Systemablauf definierten Schritte 2 bis 6.

Im Schritt 2 wird für alle verwendeten MOCCD-Algorithmen die 3D-Pose-Schätzung zum Zeitschritt t durchgeführt (Abschnitt 3.3.1). Die 3D-Pose-Schätzung $(\mathbf{m}_{\mathbf{\Phi}}^{(i)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(i)}(t))$ eines MOCCD-Algorithmus i für den Parametersatz $\mathbf{\Phi}$ zum Zeitschritt t besteht aus Mittelwertvektor $\mathbf{m}_{\mathbf{\Phi}}^{(i)}(t)$ und Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{\Phi}}^{(i)}(t)$. Grundlage dieser 3D-Pose-Schätzung ist eine a-priori Gaußverteilung des Parametervektors $\mathbf{\Phi}$, welche verfeinert wird (Abschnitt 3.3.1). Zum Zeitschritt $t = 1$ erhält man die a-priori Gaußverteilung aus der manuellen Initialisierung. Für alle weiteren Zeitschritte ergibt sich die a-priori Gaußverteilung $p(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}^{(i)}(t), \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}^{(i)}(t))$ aus der Prädiktion $\widehat{\mathbf{\Phi}}_i(t)$ des zum jeweiligen MOCCD-Algorithmus zugehörigen Trackers (Schritt 6). D.h. der initiale Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}^{(i)}(t)$ für den MOCCD-Algorithmus i zum Zeitschritt t wird aus aus dem prädizierten Parametervektor $\widehat{\mathbf{\Phi}}_i(t)$ bestimmt. Die initiale Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}^{(i)}(t)$ ergibt sich immer aus einer konstanten Matrix, welche die Unsicherheit der einzelnen Elemente des Parametervektors $\mathbf{\Phi}$ beschreibt und problemspezifisch gewählt wird.

Nachdem die 3D-Pose-Schätzungen zum Zeitschritt t durchgeführt wurden, muss die beste 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ bestimmt werden. Dies geschieht in der Fusionskomponente durch Auswahl der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ aus den 3D-Pose-Schätzungen der drei MOCCD-Algorithmen $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t)), (\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t)), (\mathbf{m}_{\mathbf{\Phi}}^{(3)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(3)}(t))$. Dabei wird ein „*Winner-Takes-All*“ (WTA) Ansatz mit fünf verschiedene Kriterien verwendet (Abschnitt 3.4).

Mit der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ zum Zeitschritt t wird in der Verifikationskomponente ein Selbsttest des Systems durchgeführt, sodass gewährleistet werden kann, dass das Objekt stabil getrackt wird. Falls die Verifikation scheitert, kann eine Reinitialisierung durchgeführt werden. Dabei wird durch zwei Hypothesen (Abschnitt 3.6) eine neue, verbesserte Ergebnismessung $\mathbf{\Phi}(t)$ berechnet. Diese verbesserte Ergebnismessung muss dann noch verifiziert werden und das Tracking wird fortgesetzt.

Mit der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ zum Zeitschritt t wird in jedem Tracker (Abschnitt 3.5) für dessen zugehörigen MOCCD-Algorithmus die Prädiktion für den Zeitschritt $t + \Delta t$ berechnet, d.h. drei Prädiktionen $\widehat{\mathbf{\Phi}}_1(t), \widehat{\mathbf{\Phi}}_2(t), \widehat{\mathbf{\Phi}}_3(t)$. Mit Hilfe der Prädiktionen werden im Schritt 2 die 3D-Pose-Schätzungen mit den MOCCD-Algorithmen bestimmt und der Ablauf beginnt von vorn.

Kapitel 4

Verfolgung von raum-zeitlichen 3D-Konturen

In diesem Kapitel wird ein modellbasiertes Verfahren zur Verfolgung von raum-zeitlichen 3D-Konturen in den Bildern eines Mehrkamerasystems mit kleiner Basisbreite vorgestellt. Mit dem entwickelten technischen System lässt sich die 3D-Pose von beliebigen menschlichen Körperteilen kamerabasiert schätzten und ihre Geschwindigkeit ermitteln. Wichtig ist dabei, dass das in den Kamerabildern getrackte menschliche Körperteil durch ein raum-zeitliches 3D-Konturmodell beschrieben werden kann. Außerdem darf die Objektbewegung nicht zu stark vom angenommenen Bewegungsmodell abweichen. Die 3D-Konturanpassung wird mit dem MOCCD-Algorithmus (Abschnitt 3.3.1) durchgeführt und die Objektbewegung wird mit Hilfe einer raum-zeitlichen Erweiterung des MOCCD-Algorithmus geschätzt. Diese raum-zeitliche Erweiterung zum Shape-Flow (SF) Algorithmus ist die wichtigste Neuerung in diesem Kapitel und wird ausführlich beschrieben. Mit dem Shape-Flow-Algorithmus kann die Bewegung des in der Bildsequenz verfolgten Objekts direkt gemessen werden. Dadurch können zeitliche Filter zur Bewegungsvorhersage, z.B. Kalman-Filter im System zur 3D-Konturverfolgung (Kapitel 3), ersetzt werden, was außerdem die Latenzzeit verringert. Dies ist insbesondere für ein Sicherheitssystem sehr wichtig.

Das Systemmodell zur kamerabasierten Verfolgung von raum-zeitlichen 3D-Konturen ist in Abbildung 3.1 dargestellt. Es besteht aus fünf Komponenten: (i) dem Kamerasystem , (ii) dem raum-zeitlichen Körperteil-Modell (nicht dargestellt), (iii) einem Zwei-Hypothesen-System zur raum-zeitlichen Pose-Estimation, (iv) einem Teilsystem zur Fusion und Verifikation der 3D-Pose-Schätzungen, (v) einem Modul zur Reinitialisierung des Systems. Das grau unterlegte Rechteck „Reinitialisierung“ steht für das Reinitialisierungsmodul, welches auch in diesem Trackingsystem optional verwendet werden kann. Daher gibt es zwei Systemvarianten, welche später in den Experimenten (Abschnitt 7) auch gegeneinander evaluiert werden.

Das Systemmodell ist eine Erweiterung des Systems zur 3D-Konturverfolgung (Kapitel 3) und teilweise ähnlich zu diesem. Daher wurden Teile des Systems zur kamerabasierten Verfolgung von raum-zeitlichen 3D-Konturen schon im vorherigen Abschnitt detailliert beschrieben. In diesem Kapitel wird bei der Systembeschreibung auf die entsprechenden Abschnitte in Kapitel 3 verwiesen. Neu in diesem Trackingsystem ist der Shape-Flow (SF) Algorithmus und das für den SF-Algorithmus notwendige raum-zeitliche 3D-Konturmodell. Im Folgenden werden alle Komponenten, die bisher nicht bekannt sind, näher erläutert und abschließend die Interaktion der Komponenten im Gesamtsystem dargestellt.

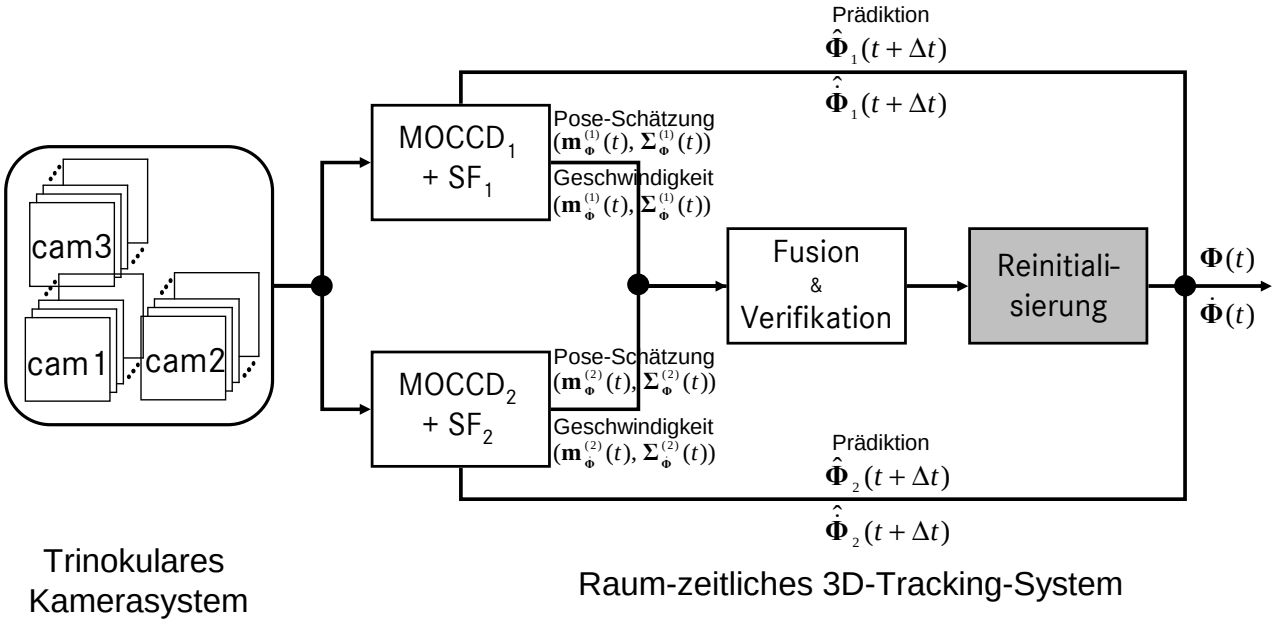

Abbildung 4.1: Struktur des Systemmodells zur raum-zeitlichen 3D-Konturverfolgung.

4.1 Raum-zeitliche Pose-Estimation

In diesem Abschnitt geht es um ein Verfahren zur Bestimmung der 3D-Pose und deren zeitlicher Ableitung in Bildfolgen, was als raum-zeitliche Pose-Estimation bezeichnet wird.

Die raum-zeitliche Pose-Estimation kann als top-down oder bottom-up Ansatz durchgeführt werden. Bei dem top-down Ansatz sind Bildfolgen und ein raum-zeitliches Modell des in den Bildern verfolgten Objekts notwendig. Durch Optimierung einer Zielfunktion, welche beschreibt wie gut die raum-zeitlichen Modellparameter zur Objektbewegung in der Bildfolge passen, wird das raum-zeitliche Modell an die Bildfolge angepasst. Durch die Anpassung des raum-zeitlichen Modells kann anschließend für jeden 3D-Punkt innerhalb des Modells die 3D-Bewegung in der beobachteten Szene abgeleitet werden, ohne dass beispielsweise das 3D-Bewegungsfeld von einzelnen Punkten in der Welt (Huguet u. Devernay, 2007) berechnet wird.

Bei dem bottom-up Ansatz wird die Pose-Estimation auf 3D-Punktewolken mit Bewegungsinformation durchgeführt (Duric u. a., 2002; Barrois u. Wöhler, 2008). Dabei kann die 3D-Punktewolken in einzelne Objekte geclustert werden oder die Pose-Estimation direkt auf der 3D-Punktewolken durchgeführt werden. Die raum-zeitliche Komponente der Pose-Estimation wird hierbei über die bewegten 3D-Punkte eingebracht, da diese aus einer Bildfolge berechnet werden. Eine solche 3D-Punktewolken mit Bewegungsinformation wird auch als Szenenfluss (Huguet u. Devernay, 2007) bezeichnet. Ein weitere Möglichkeit zur Berechnung der 3D-Punktewolken mit Bewegungsinformation stellt das Spacetime-Stereo dar (Schmidt u. a., 2007).

Im Folgenden wird der Shape-Flow (SF) Algorithmus, ein top-down Ansatz zur modellbasierten raum-zeitlichen 3D-Pose-Estimation vorgestellt. Der zum Teil neu entwickelte und verbesserte Bayes'sche Ansatz des Shape-Flow-Algorithmus erlaubt eine stabile raum-zeitliche

3D-Pose-Estimation auf Bildfolgen und damit die Berechnung eines dichten 3D-Szenenfluss. Durch den Shape-Flow-Algorithmus kann neben der raum-zeitlichen 3D-Pose-Estimation auch die zeitliche Änderung der Geometrie des Modells geschätzt werden, dies wird beim Eingangsbeispiel im Abschnitt 4.1.1 des Shape-Flow-Algorithmus sehr gut deutlich.

4.1.1 Shape-Flow (SF) Algorithmus

Der Shape-Flow-Algorithmus ist eine raum-zeitliche Erweiterung des MOCCD-Algorithmus (Abschnitt 3.3.1). Es wird eine raum-zeitliche Kontur an eine Folge von Bildern angepasst. Mit dem Ansatz können die 3D-Pose-Parameter $\mathbf{\Phi}$ und deren zeitliche Ableitung mit einem raum-zeitlichen Konturmodell und den Bildern von N_t Zeitschritten geschätzt werden. Durch die top-down Verfahrensweise muss lediglich ein Konturmodell, welches die Kontur des in den Bildfolgen getracketen Objekts beschreibt, und eine grobe Initialisierung der 3D-Pose-Parameter $\mathbf{\Phi}$ vorhanden sein. Mit dem Shape-Flow-Algorithmus ist man in der Lage einen dichten modellbasierten 3D-Szenenfluss zu berechnen und somit auch die Tiefengeschwindigkeit des Objekts zu schätzen. Anhand eines Eingangsbeispiels wird nun das Prinzip der modellbasierten raum-zeitlichen 3D-Pose-Estimation erläutert und danach werden die algorithmischen Details ausführlich beschrieben.

Eingangsbeispiel

Zum besseren Verständnis der modellbasierten raum-zeitlichen 3D-Pose-Estimation mit dem Shape-Flow-Algorithmus, wird anhand eines einfachen Beispiels die Funktionsweise des Shape-Flow-Algorithmus näher erläutert. Im synthetisch generierten Beispiel bewegt sich ein Kreis nach rechts unten und wird dabei größer (Abbildung 4.2). Mit dem Shape-Flow-Algorithmus soll nun ein raum-zeitliches Modell des sich bewegenden Kreises an die synthetischen Bilddaten angepasst werden. Im Beispiel wird nur eine Kamera verwendet, daher ist das raum-zeitliche Modell, welches durch den Parametervektor $\mathbf{\Phi}$ beschrieben wird, im (u, v, t)-Raum des Pixelkoordinatensystems P_1 der Kamera 1 definiert und hat folgende sechs Parameter:

$$\mathbf{\Phi} = [{}^{P_1}m_u, {}^{P_1}m_v, r, {}^{P_1}\dot{m}_u, {}^{P_1}\dot{m}_v, \dot{r}]^T. \tag{4.1}$$

Dabei beschreibt $({}^{P_1}m_u, {}^{P_1}m_v)$ den Mittelpunkt und r den Radius des Kreises zum Zeitschritt t. Die restlichen Parameter sind zeitliche Ableitungen, $({}^{P_1}\dot{m}_u, {}^{P_1}\dot{m}_u)$ beschreibt die erste zeitliche Ableitung des Mittelpunkts und $\dot{r}$ die erste Ableitung des Radius nach der Zeit. Das verwendete Bewegungsmodell ist konstante Geschwindigkeit, daher ergibt sich Mittelpunkt und Radius des Kreises zum Zeitschritt $(t + \Delta t)$ mit:

$$\begin{pmatrix} {}^{P_1}m_u(t+\Delta t) \\ {}^{P_1}m_v(t+\Delta t) \\ r(t+\Delta t) \end{pmatrix} = \begin{pmatrix} {}^{P_1}m_u(t) \\ {}^{P_1}m_v(t) \\ r(t) \end{pmatrix} + \Delta t \cdot \begin{pmatrix} {}^{P_1}\dot{m}_u(t) \\ {}^{P_1}\dot{m}_v(t) \\ \dot{r}(t) \end{pmatrix}. \tag{4.2}$$

Der Vorteil dieses einfachen Beispiels ist, dass man das verwendete raum-zeitliche Modell geometrisch interpretieren und sich damit vorstellen kann. Im (u, v, t)-Raum kann das raum-zeitliche Kreismodell als schiefer Kegelstumpf mit parallelen Grundflächen interpretiert werden. In Abbildung 4.2 sieht man die synthetischen Bilder des bewegten Kreises zu den drei

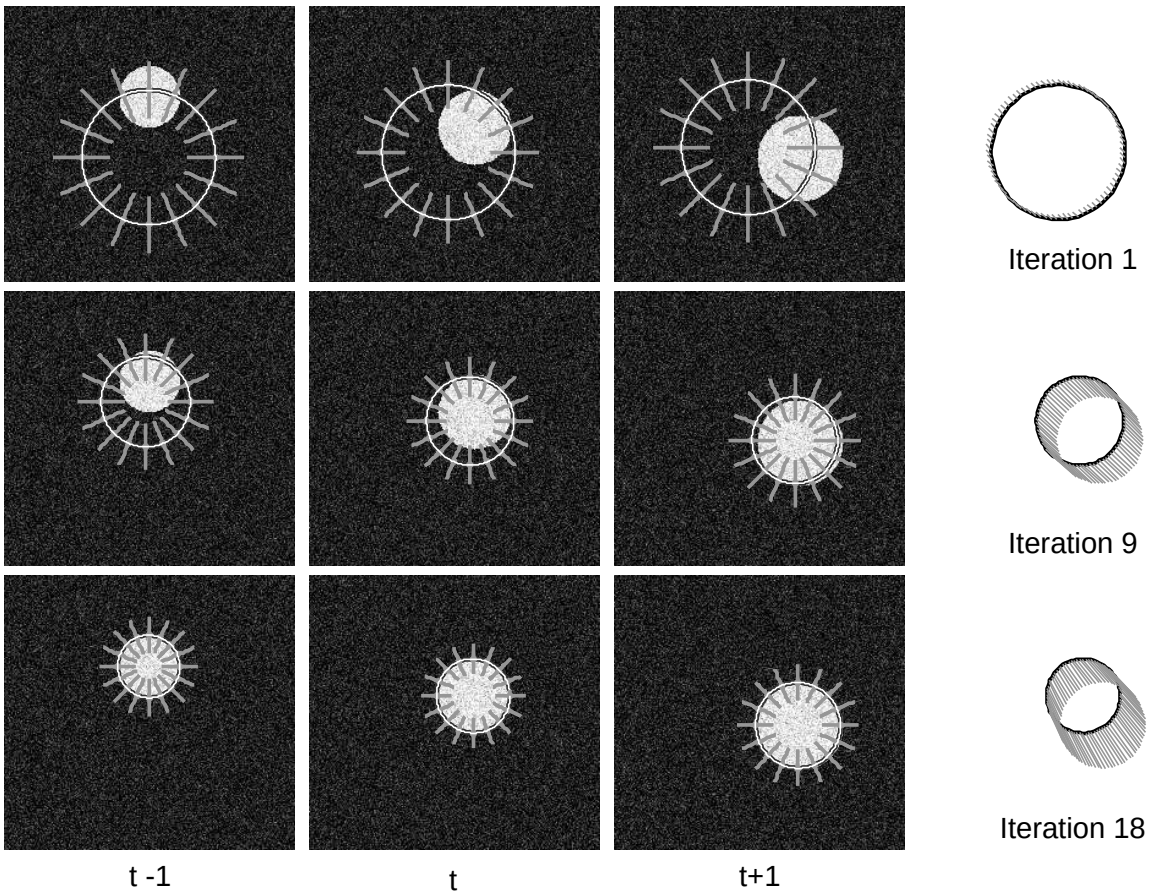

Abbildung 4.2: In den ersten drei Spalten sind die Bilder eines bewegten Kreises zu verschiedenen Zeitschritten dargestellt und in der letzten Spalte sieht man den geschätzten Kreis zum Zeitschritt t sowie die geschätzte zeitliche Ableitung als Flussvektoren. In den Zeilen ist die Bildfolge und das aktuelle adaptierte Modell zu verschiedenen Iterationen dargestellt. Das dargestellte Modell zur ersten Iteration entspricht der Initialisierung des Shape-Flow-Algorithmus.

Zeitschritten $(t - \Delta t), t$ und $(t + \Delta t)$ und das Ergebnis des Shape-Flow-Algorithmus zu drei verschiedenen Iterationen. In der vierten Spalte ist der geschätzte Kreis zum Zeitschritt t und die geschätzte zeitliche Ableitung als Flussvektoren dargestellt. Das dargestellte Modell zur ersten Iteration entspricht der Initialisierung des Shape-Flow-Algorithmus. Wichtig beim Verständnis ist, dass das Modell nicht separat an jeden Zeitschritt angepasst wird, sondern direkt an die Bildfolge. Dadurch werden die Randbedingungen von Bewegungen direkt in der Optimierung berücksichtigt und die Pose-Estimation wird robuster.

Außerdem wird in dem Beispiel auch deutlich, dass durch den Shape-Flow-Algorithmus ein dichter modellbasierter Szenenfluss berechnet wird. Da für jeden 3D-Punkt des Modells die zeitliche Verschiebung über die geschätzten raum-zeitlichen Modellparameter ermittelt werden kann. Ein weiterer Vorteil ist, dass durch die Modellierung die Verschiebung auch subpixelgenau berechnet werden kann.

Die raum-zeitliche Erweiterung zum Shape-Flow-Algorithmus

Der Shape-Flow-Algorithmus ist eine raum-zeitliche Erweiterung des MOCCD-Algorithmus (Abschnitt 3.3.1) und in der Lage, ein raum-zeitliches Konturmodell an die Bilder eines Mehrkamerasystems (N_c Kameras) in N_t Zeitschritten anzupassen. Durch das Anpassen der Modellkontur an die Objektkontur in der Bildfolge wird die 3D-Pose $\mathbf{\Phi}$ und die zeitliche Ableitung der Pose-Parameter geschätzt. Die Erweiterung basiert auf der Integration aller N_c Kamerabilder zu allen Zeitschritten N_t in die zur Anpassung des Modellparametervektors $\mathbf{\Phi}$ maximierte Wahrscheinlichkeit $p(\mathbf{\Phi}|\mathbf{I}_{c,t})$ mit $c \in \{1,\ldots,N_\mathrm{c}\}$ und $t \in \{1,\ldots,N_\mathrm{t}\}$. Die Eingabeparameter des Shape-Flow-Algorithmus sind N_c Eingabebilder zu N_t Zeitschritten, welche von einem kalibrierten Mehrkamerasystem geliefert werden. Außerdem ist ein parametrisches raum-zeitliches 3D-Kurvenmodell notwendig. Wie beim MOCCD-Algorithmus (Abschnitt 3.3.1) wird die raum-zeitliche Kurve an erweiterte Eingabebilder $\mathbf{I}^*_{c,t}$ angepasst, welche mit Gleichung (3.47) berechnet werden. Durch die erweiterten Eingabebilder $\mathbf{I}^*_{c,t}$ wird – wie bereits im Kapitel 3 beschrieben – die Segmentierung robuster, da die Trennung der Statistiken eindeutiger wird.

Wie beim MOCCD-Algorithmus wird der Modellparametervektor $\mathbf{\Phi}$ durch eine Gaußverteilung mit Mittelwert und Kovarianzmatrix approximiert. Dabei beschreibt der Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}$ die raum-zeitliche 3D-Modellkurve und die Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}$ die Unsicherheit entlang dieser. Vor der ersten Iteration wird der Shape-Flow-Algorithmus initialisiert, indem die optimierten Verteilungsparameter $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf die gegebenen a-priori Gaußverteilungsparameter $(\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}})$ gesetzt werden. Der Shape-Flow-Algorithmus beruht auf folgenden drei Schritten, welche so lange wiederholt werden, bis die Änderungen an den Verteilungsparameter $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ unter einer Schwelle liegen oder eine bestimmte Anzahl von Iterationen erreicht ist.

1. Schritt – Projektion des raum-zeitlichen 3D-Konturmodells: In diesem Schritt wird das raum-zeitliche 3D-Konturmodell zu allen Zeitschritten $t,\ t \in \{1,\ldots,N_\mathrm{t}\}$, in die 2D-Pixelkoordinatensysteme der verwendeten Kameras $c,\ c \in \{1,\ldots,N_\mathrm{c}\}$, projiziert. Falls als 3D-Modell ein Volumenmodell eingesetzt wird, muss aus dem 3D-Volumenmodell die für das Kamerasystem sichtbare 3D-Silhouette extrahiert werden. Für das 3D-Hand-Unterarm-Modell ist dies der Fall, daher wird in Abschnitt 3.2.1 ein Verfahren zur Extraktion der sichtbaren 3D-Kontur für ein 3D-Volumenmodell des menschlichen Unterarms und der Hand (Abbildung 3.5) beschrieben. Bei der Projektion ist es wichtig, dass die Kameras hinreichend genau kalibriert sind, denn wie beim MOCCD-Algorithmus wirkt sich die Genauigkeit der Kalibrierung direkt auf die Genauigkeit der Schätzung des Shape-Flow-Algorithmus aus. Abbildung 4.3 zeigt die Projektion des 3D-Konturmodells für die drei Zeitschritte $(t-\Delta t),\ t$ und $(t+\Delta t)$.

2. Schritt – Berechnung der lokalen Pixelstatistiken für alle N_c Kamerabilder zu allen N_t Zeitschritten: Für alle N_c Eingabebilder $\mathbf{I}^*_{c,t}$ werden zu allen N_t Zeitschritten die lokalen Pixelstatistiken $S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf Liniensegmenten entlang der projizierten raum-zeitlichen 3D-Modellkurve berechnet. Diese 2D-Kurven werden zur Berechnung der Pixelstatistiken verwendet, dies ist für jede der 2D-Kurven identisch zum ersten Schritt des CCD-Algorithmus (Abschnitt 3.3.1). Neu ist, dass es für alle N_c Eingabebilder $\mathbf{I}^*_{c,t}$ und alle N_t Zeit-

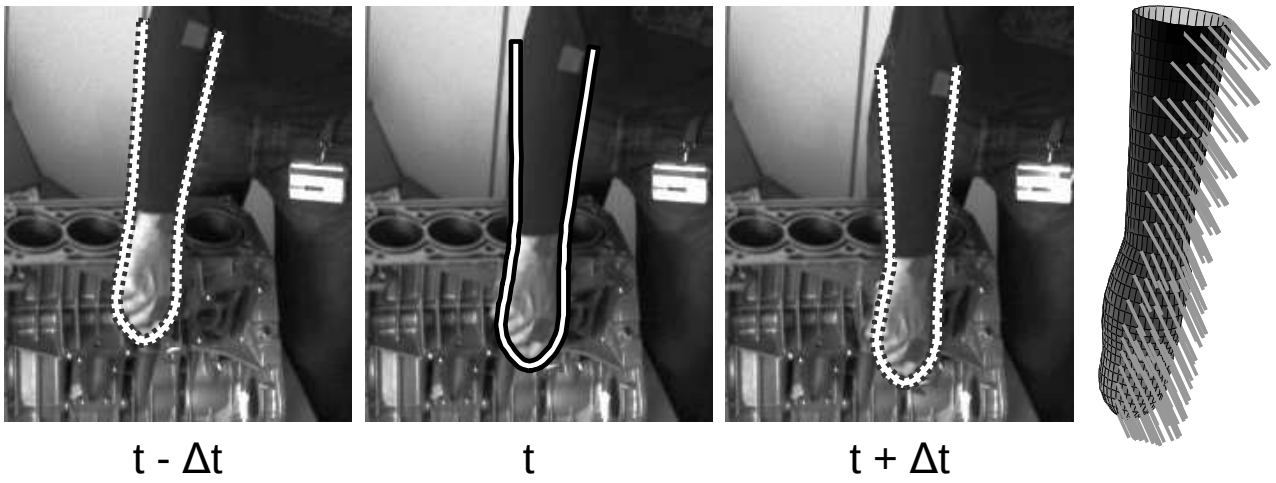

Abbildung 4.3: Darstellung der Projektion des 3D-Konturmodells für die drei Zeitschritte $(t - \Delta t)$, t und $(t + \Delta t)$. Außerdem wird ein 3D-Volumenmodell mit der geschätzten zeitlichen Ableitung gezeigt.

schritte lokale Pixelstatistiken $S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ gibt, diese werden nun zur Anpassung der raum-zeitlichen 3D-Modellparameter verwendet.

3. Schritt – Anpassen der Modellparameter: Mit den lokalen Pixelstatistiken $S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ aus allen verwendeten Eingabebildern und Zeitschritten wird nun die Anpassung der Modellparameter durchgeführt. Der Output des Shape-Flow-Algorithmus beinhaltet den neu geschätzten Mittelwert $\mathbf{m}_{\mathbf{\Phi}}$ und die korrespondierende Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{\Phi}}$. Vor der ersten Iteration wird der Shape-Flow-Algorithmus initialisiert, indem die optimierten Parameter $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ auf die gegebenen a-priori Gaußverteilungsparameter $(\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}})$ gesetzt werden.

Zur Anpassung des Modellparametersatzes $\mathbf{\Phi}$, welcher durch Mittelwert $\mathbf{m}_{\mathbf{\Phi}}$ und Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{\Phi}}$ approximiert wird, wird die Wahrscheinlichkeit

$$p(\mathbf{\Phi}|\{\mathbf{I}^*_{c,t}\}) \;=\; p(\mathbf{I}^*_{c,t}|\mathbf{\Phi}) \cdot p(\mathbf{\Phi}), \quad \forall \;\; c \in \{1, \ldots, N_\mathrm{c}\} \;\wedge\; t \in \{1, \ldots, N_\mathrm{t}\} \tag{4.3}$$

durch eine Maximum-A-Posteriori-Schätzung maximiert. Wenn man in Gleichung (4.3) annimmt, dass die verschiedenen Eingabebilder $\mathbf{I}^*_{c,t}$ unabhängige Zufallsvariablen sind, kann man die Gleichung nach MacKay (2003) folgendermaßen umschreiben:

$$p(\mathbf{\Phi}|\{\mathbf{I}^*_{c,t}\}) = \left[\prod_c \prod_t p(\mathbf{I}^*_{c,t}|\mathbf{\Phi})\right] \cdot p(\mathbf{\Phi}). \tag{4.4}$$

Da wie beim CCD-Algorithmus die aktuellen Verteilungen der Wahrscheinlichkeiten $p(\mathbf{\Phi})$ und $p(\mathbf{I}^*_{c,t}|\mathbf{\Phi})$ unbekannt sind, werden diese durch Gaußverteilungen folgendermaßen approximiert:

$$p(\mathbf{\Phi}) \;\approx\; p(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}) \tag{4.5}$$

$$p(\mathbf{I}^*_{c,t}|\mathbf{\Phi}) \;\approx\; p(\mathbf{I}^*_{c,t}|S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})), \quad \forall \;\; c \in \{1, \ldots, N_\mathrm{c}\} \;\wedge t \in \{1, \ldots, N_\mathrm{t}\} \tag{4.6}$$

Die finale Maximum-A-Posteriori-Schätzung basiert also auf folgender Wahrscheinlichkeit:

$$p(\mathbf{\Phi}|\{\mathbf{I}^*_{c,t}\}) = \left[\prod_c \prod_t p(\mathbf{I}^*_{c,t}|S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}}))\right] \cdot p(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}, \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}). \tag{4.7}$$

Dabei repräsentiert $S_{c,t}(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ die lokalen Pixelstatistiken der Kamera c zum Zeitschritt t in der Nähe der projizierten Kurve im Eingabebild $\mathbf{I}^*_{c,t}$. Wie im zweiten Schritt des CCD-Algorithmus (Abschnitt 3.3.1) und im dritten Schritt des MOCCD-Algorithmus (Abschnitt 3.3.1) wird durch Logarithmierung aus dem Maximierungsproblem (Gleichung (4.7)) ein Minimierungsproblem. Dies ist wichtig, um eine stabile Optimierung zu erhalten und um numerische Probleme zu vermeiden. Dieses Minimierungsproblem wird nun durch einen Schritt einer Newton-Raphson-Optimierung (Weisstein, 2008) minimiert und danach beginnt die nächste Iteration des Shape-Flow-Algorithmus wieder mit Schritt 1.

Der Shape-Flow-Algorithmus ist ein top-down Ansatz zur modellbasierten raum-zeitlichen 3D-Pose-Estimation, denn es werden aus einer Bildfolge eines bewegten Objekts mit einem raum-zeitlichen Objektmodell direkt die 3D-Pose-Parameter und deren zeitliche Ableitungen geschätzt. Ein großer Vorteil ist, dass das raum-zeitliche Konturmodell nicht separat an jeden Zeitschritt angepasst wird, sondern direkt an die Bildfolge. Dadurch werden räumliche und zeitliche Bedingungen der Objektbewegung direkt in der Optimierung berücksichtigt und die Bewegung direkt gemessen. Außerdem ist es möglich, aus den geschätzten raum-zeitlichen Modellparametern einen dichten modellbasierten Szenenfluss zu ermitteln. Dabei kann die Verschiebung subpixelgenau bestimmt werden.

4.2 3D raum-zeitliche Körpermodelle

In diesem Abschnitt wird die Erzeugung eines raum-zeitlichen 3D-Konturmodells der menschlichen Hand-Unterarm-Region und der Kopf-Schulter-Partie beschrieben. Der Shape-Flow-Algorithmus ist in der Lage, raum-zeitliche Konturen an eine Bildfolge anzupassen. Die im Shape-Flow-Algorithmus optimierten Variablen $(\mathbf{m}_{\mathbf{\Phi}}, \mathbf{\Sigma}_{\mathbf{\Phi}})$ enthalten räumlich und zeitlich abhängige Parameter. Das Eingangsbeispiel des Shape-Flow-Algorithmus im Abschnitt 4.1.1 zeigt dies in anschaulicher Weise.

In dieser Arbeit wird der Shape-Flow-Algorithmus dazu verwendet, um die erste zeitliche Ableitung $\dot{\mathbf{\Phi}}$ des Parametervektors $\mathbf{\Phi}$ zu schätzen. Prinzipiell kann der Shape-Flow-Algorithmus, wie das Eingangsbeispiel (Abschnitt 4.1.1) zeigt, die räumlich und zeitlich abhängigen Parameter der Körpermodelle (Abschnitt 3.2) direkt schätzen. Bei den verwendeten Modellen für den Hand-Unterarm-Bereich oder die Kopf-Schulter-Partie hat sich aber bei der Implementierung gezeigt, dass die Dimensionalität des Parametervektors durch Hinzunahme der zeitlich abhängigen Parameter zu groß wird. Außerdem traten dann numerische Instabilitäten bei der Optimierung auf, da die Bilddaten teilweise sehr schlecht sind und die Trennung der Pixelstatistiken somit schwierig wird. Die Optimierung ist dann schlichtweg nicht mehr in der Lage, dass Objekt robust und zuverlässig aus den Bildfolgen zu segmentieren.

In dieser Arbeit wird daher ein hierarchischer Ansatz gewählt, d.h. die Berechnung der raum-zeitlichen Pose-Parameter wird aufgeteilt. Die 3D-Pose-Estimation des Parametervektors $\mathbf{\Phi}(t)$ zum Zeitschritt t wird durch den MOCCD-Algorithmus berechnet und die zeitliche

Ableitung $\dot{\mathbf{\Phi}}(t)$ der 3D-Pose durch den Shape-Flow-Algorithmus geschätzt. Im Folgenden werden die verwendeten zeitlich abhängigen Parameter der Körperteilmodelle aus Abschnitt 3.2 vorgestellt.

4.2.1 Raum-zeitliches 3D-Hand-Unterarm-Modell

Die durch den Shape-Flow-Algorithmus geschätzte zeitliche Ableitung $\dot{\mathbf{\Phi}}$ des Parametervektor $\mathbf{\Phi}$ für das raum-zeitliche 3D-Hand-Unterarm-Modell besitzt sieben Parameter:

$$\dot{\mathbf{\Phi}} = [{}^W\dot{p}_{1X}, {}^W\dot{p}_{1Y}, {}^W\dot{p}_{1Z}, \dot{\alpha}_1, \dot{\beta}_1, \dot{\alpha}_2, \dot{\beta}_2]^T. \tag{4.8}$$

Dabei ergeben sich die Parameter durch zeitliche Ableitung des Parametervektors $\mathbf{\Phi}$ in Gleichung (3.1). Es wird angenommen, dass die im Parametervektor $\mathbf{\Phi}$ enthaltenen Radien r_1 und r_4 über kurze Zeiträume nicht veränderlich sind, daher sind die Radien nicht Bestandteil von $\dot{\mathbf{\Phi}}$. Zur Berechnung des raum-zeitlichen 3D-Konturmodells wird angenommen, dass sich das in den Bildern verfolgte Objekt mit konstanter Geschwindigkeit bewegt. Deshalb wird der Parametervektor $\mathbf{\Phi}(t + \Delta t)$ zur Berechnung des Hand-Unterarm-Modells zum Zeitschritt $(t + \Delta t)$ folgendermaßen berechnet:

$$\mathbf{\Phi}(t + \Delta t) = \mathbf{\Phi}(t) + \dot{\mathbf{\Phi}}(t) \cdot \Delta t. \tag{4.9}$$

Mit dem so berechneten Parametervektor $\mathbf{\Phi}(t + \Delta t)$ kann wie im Abschnitt 3.2.1 beschrieben, das 3D-Hand-Unterarm-Modell zum Zeitschritt $(t+\Delta t)$ berechnet werden. Die Berechnung für den Parametervektor $\mathbf{\Phi}(t-\Delta t)$ zum Zeitschritt $(t-\Delta t)$ erfolgt analog. Abbildung 4.3 zeigt die Projektion in die Bilddaten eines so berechneten 3D-Konturmodells für die drei Zeitschritte $(t - \Delta t),\ t$ und $(t + \Delta t)$.

4.2.2 Raum-zeitliches 3D-Kopf-Schulter-Modell

Die durch den Shape-Flow-Algorithmus geschätzte zeitliche Ableitung $\dot{\mathbf{\Phi}}$ des Parametervektor $\mathbf{\Phi}$ für das raum-zeitliche 3D-Kopf-Schulter-Modell besitzt folgende sechs Parameter:

$$\dot{\mathbf{\Phi}} = [{}^W\dot{p}_{1X}, {}^W\dot{p}_{1X}, {}^W\dot{p}_{1Z}, \dot{\alpha}, \dot{\beta}_1, \dot{\beta}_2]^T. \tag{4.10}$$

Wiederum ergeben sich die Parameter durch zeitliche Ableitung des Parametervektors $\mathbf{\Phi}$ in Gleichung (3.8). Außerdem wird wie beim raum-zeitliches 3D-Hand-Unterarm-Modell angenommen, dass der im Parametervektor $\mathbf{\Phi}$ enthaltene Radius r_4 über einen kurzen Zeitraum nicht veränderlich ist. Daher ist r_4 nicht Bestandteil von $\dot{\mathbf{\Phi}}$. Zur Berechnung des raum-zeitliches 3D-Konturmodells wird wiederum angenommen, dass sich das in den Bildern verfolgte Objekt mit konstanter Geschwindigkeit bewegt. Der Parametervektor $\mathbf{\Phi}(t + \Delta t)$ zur Berechnung des Kopf-Schulter-Modells zum Zeitschritt $(t + \Delta t)$ wird mit Gleichung (4.9) berechnet und damit das Konturmodell, wie im Abschnitt 3.2.2 beschrieben, erzeugt. Die Berechnung des Konturmodells für den Parametervektor $\mathbf{\Phi}(t - \Delta t)$ zum Zeitschritt $(t - \Delta t)$ erfolgt analog.

4.3 Interaktion im Gesamtsystem

Wie in Abbildung 4.4 zu sehen, ist das Systemmodell als Zwei-Hypothesen-Trackingsystem aufgebaut. Neu an diesem Systemmodell ist, dass die 3D-Pose des Objekts und deren zeitliche

Ableitung, also die Geschwindigkeit des Objekts, direkt aus den Bilddaten geschätzt wird und nicht wie beim System zur bildbasierten Verfolgung von 3D-Konturen (Kapitel 3) durch ein Kalman-Filter. Dadurch kann direkt die Geschwindigkeit geschätzt werden, Einschwingphasen des Filters werden vermieden, die Latenzzeit der Reaktion wird geringer und die Prädiktion der Bewegung wird verbessert. In dieser Arbeit wird ein hierarchischer Ansatz zur Schätzung der 3D-Pose des Objekts und deren zeitlicher Ableitung gewählt, d.h. die Berechnung der raum-zeitlichen Pose-Parameter wird aufgeteilt. Die 3D-Pose-Estimation des Parametervektors $\mathbf{\Phi}(t)$ zum Zeitschritt t wird durch den MOCCD-Algorithmus (Abschnitt 3.3.1) berechnet und die zeitliche Ableitung $\dot{\mathbf{\Phi}}(t)$ der 3D-Pose, also die Geschwindigkeit, durch den Shape-Flow-Algorithmus (Abschnitt 4.1.1) geschätzt.

Außerdem ist an diesem Systemmodell neu, dass nur zwei 3D-Pose-Schätzungen durchgeführt werden und nicht wie beim System zur bildbasierten Verfolgung von 3D-Konturen (Kapitel 3) drei 3D-Pose-Schätzungen. Die Begründung dieses Zwei-Hypothesen-Ansatzes liegt in den zur Berechnung der 3D-Pose-Schätzung verwendeten Algorithmen, diese sind Pose-Refinement Algorithmen, welche auf durch ein Bewegungsmodell propagierten Initialparametern basieren. Bei einer schlechten Prädiktion der Modellparameter kann das Objekt nicht mehr aus den Bilddaten segmentiert werden. Schlechte Prädiktionen treten auf, wenn die Bewegung des Objekts nicht zum Bewegungsmodell passt oder die Segmentierung aus den Bilddaten schwierig ist. Durch den Zwei-Hypothesen-Ansatz und die somit unterschiedlichen initialen a-priori Gaußverteilungen von MOCCD-Algorithmus und Shape-Flow-Algorithmus wird ein robusteres Verhalten erreicht. Aufgrund der unterschiedlichen Initialisierung der MOCCD-Algorithmen und Shape-Flow-Algorithmen durch ihr zugehöriges Bewegungsmodell können diese Pose-Refinement Algorithmen in unterschiedlich gute Minima konvergieren und somit unterschiedlich gute 3D-Pose-Schätzungen liefern.

In der Fusions- und Verifikationskomponente (Abschnitt 3.4) werden schlechte 3D-Pose-Schätzungen zur Bestimmung der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ und besten Geschwindigkeitsmessung $\dot{\mathbf{\Phi}}(t)$ nicht berücksichtigt, was zu einem stabileren Verhalten des Systems führt. Die Verifikation sorgt für einen Selbsttest des Systems, sodass erkannt werden kann, wenn das in den Bildern verfolgte Objekt verloren wurde. Dabei wird durch das Verifikationsmodul eine zeitliche Konsistenz des Objektaussehens überprüft. Falls der Selbsttest nicht bestanden wird, kann ein Reinitialisierungsmodul das Objekt wiederfinden. Der folgende Systemablauf gibt einen Überblick über die Interaktion der Teilsysteme, die einzelnen Schritte des Ablaufs sind in Abbildung 4.4 farblich hervorgehoben.

1. Initialisierung
2. Berechne für alle MOCCD-Algorithmen die 3D-Pose-Schätzung zum Zeitschritt t, d.h. $(\mathbf{m}_{\mathbf{\Phi}}^{(1)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(1)}(t))$, und $(\mathbf{m}_{\mathbf{\Phi}}^{(2)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(2)}(t))$
3. Berechne für alle Shape-Flow-Algorithmen die Geschwindigkeitsmessung zum Zeitschritt t, d.h. $(\mathbf{m}_{\dot{\mathbf{\Phi}}}^{(1)}(t), \mathbf{\Sigma}_{\dot{\mathbf{\Phi}}}^{(1)}(t))$, und $(\mathbf{m}_{\dot{\mathbf{\Phi}}}^{(2)}(t), \mathbf{\Sigma}_{\dot{\mathbf{\Phi}}}^{(2)}(t))$
4. Fusion der zur Ergebnisschätzung des Parametervektors $\mathbf{\Phi}(t)$ und der Bewegung $\dot{\mathbf{\Phi}}(t)$ zum Zeitschritt t
5. Verifikation der Ergebnisschätzungen $\mathbf{\Phi}(t)$ und $\dot{\mathbf{\Phi}}(t)$ (Selbsttest des Systems)
6. Reinitialisierung falls der Selbsttest nicht bestanden wurde, Ausgabe einer neuen Ergebnismessung $\mathbf{\Phi}(t)$ mit anschließender Verifikation dieser

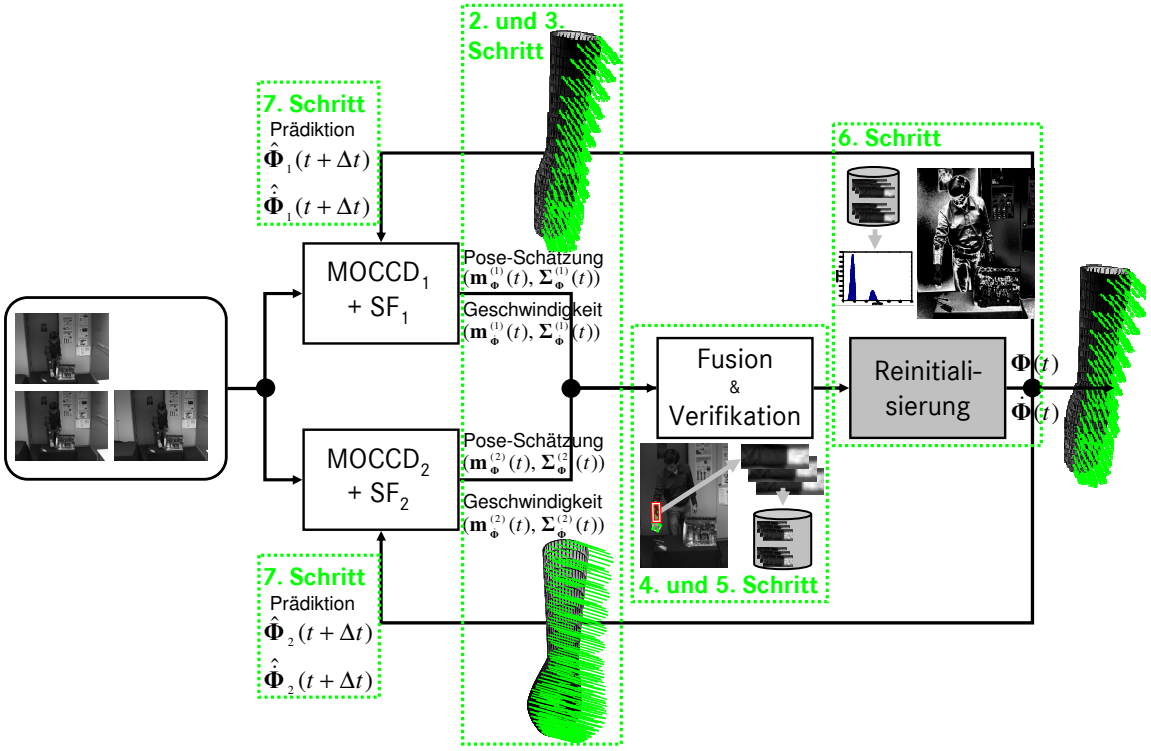

Abbildung 4.4: Struktur des Systemmodells zur raum-zeitlichen 3D-Konturverfolgung. Die einzelnen Schritte des Systemablaufs sind farblich hervorgehoben.

7. Berechnung der Prädiktionen $\widehat{\mathbf{\Phi}}_i(t+\Delta t)$ für den Zeitschritt $t+\Delta t$ basierend auf dem Bewegungsmodell

Bei der Initialisierung wird der Parametervektor $\mathbf{\Phi}(t=1)$, welcher die 3D-Position und Orientierung des zu verfolgenden Objektes zum Zeitschritt $t=1$ beschreibt, definiert. Die Geschwindigkeit $\dot{\mathbf{\Phi}}(t=1)$ wird mit Null initialisiert. Nach der Initialisierung ist das System bereit, das durch das eingesetzte Konturmodell festgelegte Körperteil zu verfolgen. Dies geschieht durch die beim Systemablauf definierten Schritte 2 bis 7.

Im Schritt 2 wird für die zwei verwendeten MOCCD-Algorithmen die Schätzung der 3D-Pose mit dem Bildtripel des Kamerasystems zum Zeitschritt t durchgeführt (Abschnitt 3.3.1). Die 3D-Pose-Schätzung $(\mathbf{m}_{\mathbf{\Phi}}^{(i)}(t), \mathbf{\Sigma}_{\mathbf{\Phi}}^{(i)}(t))$ eines MOCCD-Algorithmus i für den Parametersatz $\mathbf{\Phi}$ zum Zeitschritt t besteht aus Mittelwertvektor $\mathbf{m}_{\mathbf{\Phi}}^{(i)}(t)$ und Kovarianzmatrix $\mathbf{\Sigma}_{\mathbf{\Phi}}^{(i)}(t)$. Grundlage dieser 3D-Pose-Schätzungen ist eine a-priori Gaußverteilung des Parametervektors $\mathbf{\Phi}$, welche verfeinert wird (Abschnitt 3.3.1). Zum Zeitschritt $t=1$ erhält man diese a-priori Gaußverteilung aus der manuellen Initialisierung. Für alle weiteren Zeitschritte ergibt sich die a-priori Gaußverteilung $p(\mathbf{\Phi}|\widehat{\mathbf{m}}_{\mathbf{\Phi}}^{(i)}(t), \widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}^{(i)}(t))$ aus der Prädiktion $\widehat{\mathbf{\Phi}}_i(t)$ des zum jeweiligen MOCCD-Algorithmus zugehörigen Bewegungsmodells (Schritt 7). D.h. der initiale Mittelwertvektor $\widehat{\mathbf{m}}_{\mathbf{\Phi}}^{(i)}(t)$ für den MOCCD-Algorithmus n zum Zeitschritt t ergibt sich aus dem prädizierten Parametervektor $\widehat{\mathbf{\Phi}}_i(t)$. Die initiale Kovarianzmatrix $\widehat{\mathbf{\Sigma}}_{\mathbf{\Phi}}^{(i)}(t)$ ergibt sich immer aus einer konstanten Matrix, welche die Unsicherheit der einzelnen Elemente des Parametervektors $\mathbf{\Phi}$ beschreibt und problemspezifisch gewählt wird.

Im Schritt 3 wird mit den zwei eingesetzten Shape-Flow-Algorithmen die erste zeitliche Ableitung $\dot{\mathbf{\Phi}}(t)$ der 3D-Pose $\mathbf{\Phi}(t)$ zum Zeitschritt t mit Hilfe der Bildtripel der Zeitschritte $(t+\Delta t)$ und $(t-\Delta t)$ geschätzt. Abbildung 4.3 zeigt dies für die Bilder der Masterkamera. Wie beim MOCCD-Algorithmus wird eine a-priori Gaußverteilung der zeitlich abhängigen Parameter verfeinert (Abschnitt 4.1.1). Der initiale Mittelwertvektor für einen Shape-Flow-Algorithmus i zum Zeitschritt t ergibt sich aus dem prädizierten Parametervektor $\widehat{\dot{\mathbf{\Phi}}}_i(t)$. Die initiale Kovarianzmatrix ist wiederum eine konstante Matrix, welche die Unsicherheit der zeitlich abhängigen Parameter beschreibt.

Nachdem die Schätzungen der 3D-Pose und deren Geschwindigkeit zum Zeitschritt t durchgeführt wurden, muss die besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ und deren Geschwindigkeitsschätzung $\dot{\mathbf{\Phi}}(t)$ bestimmt werden. Dies geschieht in der Fusionskomponente durch einen „*Winner-Takes-All*“ (WTA) Ansatz mit fünf verschiedenen Kriterien verwendet (Abschnitt 3.4). Mit der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ zum Zeitschritt t wird in der Verifikationskomponente ein Selbsttest des Systems durchgeführt, sodass gewährleistet werden kann, dass das Objekt stabil getrackt wird. Falls die Verifikation scheitert, kann eine Reinitialisierung durchgeführt werden. Dabei wird durch zwei Hypothesen (Abschnitt 3.6) eine neue, verbesserte Ergebnismessung $\mathbf{\Phi}(t)$ berechnet. Die Schätzung der Geschwindigkeit $\dot{\mathbf{\Phi}}(t)$ wird im Falle einer Reinitialisierung für die zeitlich abhängigen Parameter auf Null gesetzt. Die verbesserte Ergebnismessung $\mathbf{\Phi}(t)$ muss dann noch verifiziert werden (Abschnitt 3.4) und das Tracking wird fortgesetzt.

Im Schritt 7 wird mit der besten 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ und der besten Geschwindigkeits-Schätzung $\dot{\mathbf{\Phi}}(t)$ die Prädiktion für den Zeitschritt $t+\Delta t$ durch das eingesetzte Bewegungsmodell durchgeführt. Die Erste der zwei Hypothesen hat als Bewegungsmodell konstante Geschwindigkeit implementiert, daher ergibt sich die Prädiktion mit $\widehat{\mathbf{\Phi}}_1(t+\Delta t) = \mathbf{\Phi}(t)+\dot{\mathbf{\Phi}}(t)\cdot\Delta t$. Für die zweite Hypothese berechnet sich die Prädiktion mit $\widehat{\mathbf{\Phi}}_2(t+\Delta t) = \mathbf{\Phi}(t)$. Hier wird als Bewegungsmodell konstante Position angenommen. Die Prädiktionen für die Shape-Flow-Hypothesen ergeben sich analog mit $\widehat{\dot{\mathbf{\Phi}}}_1(t+\Delta t) = \dot{\mathbf{\Phi}}(t)$ und $\widehat{\dot{\mathbf{\Phi}}}_2(t+\Delta t) = 0$. Mit Hilfe der Prädiktionen werden in den Schritten 2 und 3 wieder neue raum-zeitliche Pose-Schätzungen mit den MOCCD- und Shape-Flow-Algorithmen bestimmt und der Ablauf beginnt von vorn.

Kapitel 5

Objektverfolgung in Bilddaten und 3D-Punktewolken

In diesem Kapitel wird ein Verfahren zur Verfolgung beliebiger Objekte in Kamerabildern und 3D-Punktewolken vorgestellt. Die 3D-Punktewolken werden aus den Bildern eines Mehrkamerasystems mit kleiner Basisbreite berechnet. Das System verwendet zur Pose-Estimation keine Konturen, wie die Systeme in den beiden vorherigen Kapiteln 3 und 4, sondern Bilddaten und 3D-Punktewolken. Die eingesetzte Pose-Estimation basiert dabei auf dem Mean-Shift-Algorithmus (Anhang A.3).

Mit dem entwickelten technischen System lassen sich beliebige Objekte kamerabasiert verfolgen. Die in den Kamerabildern verfolgten Objekte werden nicht mit hoher Genauigkeit modelliert, sondern durch Ellipsoide beschrieben. Falls ein in den Bildern des Kamerasystems verfolgtes Objekt nicht mit nur einem Ellipsoiden beschrieben werden kann, werden mehrere Ellipsoide zur Beschreibung des Objekts eingesetzt und diese getrennt voneinander in den Bildern getrackt. Die Idee hinter diesem Systemmodell ist in Abbildung 5.1 dargestellt. Es sollen alle bewegten Objekte in der beobachteten Szene mit Ellipsoiden verfolgt werden und auf einer höheren Stufe des Gesamtsystems, z.B. durch eine Aktionserkennung (Abschnitt 8), aus den Bewegungsdaten die Zuordnung zum Objekt, Aktionen oder Handlungen abgeleitet werden. Für die angestrebte Applikation in dieser Arbeit ist es besonders sinnvoll, die Hände

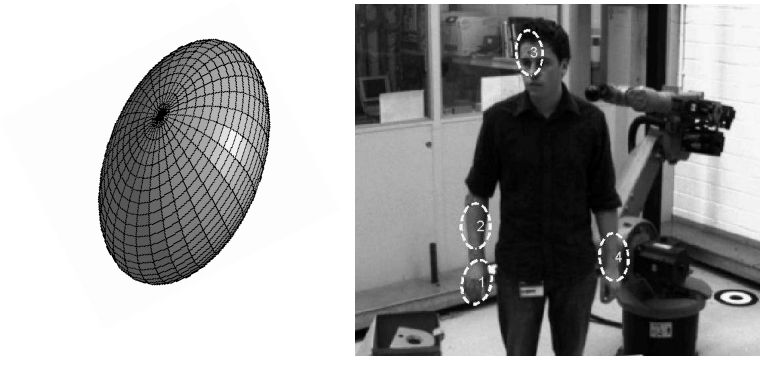

Abbildung 5.1: Links: Das verwendete Modell zur Verfolgung beliebiger Objekte in 3D-Punktewolken. Rechts: Vier verschiedene verfolgte Objekte (Ellipsoide), projiziert in das Bild der Masterkamera.

des Werkers in den Kamerabildern zu verfolgen. Diese führen die Handlungen und Aktionen

aus und sind somit der wichtigste Teil des Menschen bei der Interaktion mit dem Industrieroboter. Deshalb wird beim Tracking eine feste Ellipsoidgröße verwendet, d.h. die Längen der Halbachsen des Ellipsoiden werden an die Größe von menschlichen Händen angepasst.

Wie bereits beschrieben, basiert die Pose-Estimation in diesem System auf dem Mean-Shift-Algorithmus. Dieser Algorithmus wird von vielen Autoren (Comaniciu u. a., 2000; Comaniciu u. Meer, 2002; Yang u. a., 2003; Russel, 2004) zur kamerabasierten 2D-Verfolgung von Objekten verwendet. Es werden Bildmerkmale wie Farben, Helligkeiten, Intensitäten oder Kanten benutzt, um Ähnlichkeiten zwischen einem vorher definierten Ziel- oder Referenzobjekt und einem Bild zu finden. Die Merkmale werden dabei in Histogrammen repräsentiert, was diese Verfahren sehr effektiv macht. In der Literatur zeigt sich, dass der Mean-Shift-Algorithmus sehr robuste und zeitlich stabile Verfolgungsergebnisse liefert und effizient zu implementieren ist. Daher wird der Mean-Shift-Algorithmus zum 3D-Tracking von Objekten in dieser Arbeit verwendet. Im Anhang A.3 werden die Grundlagen zur Mean-Shift-Optimierung vorgestellt, außerdem wird der CAMShift-Algorithmus von Bradski (1998) beschrieben. Dieser stellt die Grundlage der 3D-Erweiterung des Mean-Shift-Trackings in diesem Kapitel dar.

Tyagi u. a. (2007) haben das Mean-Shift-Tracking von Comaniciu u. a. (2000) bereits in den dreidimensionalen Raum erweitert. Der Kopf eines Menschen wird mit Hilfe eines Ellipsoiden getrackt und in den Experimenten wird ein Mehrkamerasystem mit großer Basisbreite und vier Kameras verwendet. Da dieser Ansatz ein Kamerasystem mit großer Basisbreite verwendet und außerdem Farben als Bildmerkmale eingesetzt werden, ist eine 3D-Schätzung aus den Bilddaten sehr gut möglich. Denn bei einem Mehrkamerasystem mit großer Basisbreite wirken sich geringe Fehler (± 2 Pixel) bei der Disparitätsschätzung auf die Tiefenschätzung nur gering aus. Außerdem ist die Unterscheidung des verfolgten Objekts vom Hintergrund sehr gut möglich, da Farbhistogramme das Aussehen des verfolgten Objekts eindeutig beschreiben. In der angestrebten Applikation dieser Arbeit mit einem Mehrkamerasystem mit kleiner Basisbreite und Grauwertkameras kann man sich die beschriebenen Vorteile nicht zu Nutze machen. Geringe Fehler bei der Disparitätsschätzung wirken sich stark auf die Tiefenschätzung aus und in den Grauwertbildern kann das verfolgte Objekt nicht mehr so eindeutig vom Hintergrund unterschieden werden. In dieser Arbeit wird daher durch Stereobildverarbeitung eine 3D-Punktewolke erzeugt und diese zur 3D-Erweiterung des Mean-Shift-Trackings verwendet. Durch das Stereoverfahren sind die 3D-Punkte metrisch genau und durch Projektion dieser in die Kamerabilder, kann die Erscheinung des verfolgten Objekts bei der Pose-Estimation berücksichtigt werden. Die algorithmischen Details finden sich in den nächsten Abschnitten.

Das entwickelte Systemmodell ist in Abbildung 5.2 dargestellt. Es besteht aus sieben Komponenten: dem Kamerasystem (a), dem Szenenflussmodul (b), einem Modul zur Clusteranalyse (c), einer auf der Clusteranalyse basierten Ellipsoid-Detektion (d), dem 3D-Pose-Estimation System (e) und (f) sowie dem Prädiktionsmodul (g).

Das verwendete Kamerasystem wird in Abschnitt 3.1 beschrieben und ist dasselbe wie bei den zuvor beschriebenen Trackingsystemen (Kapitel 3 und 4). Im nächsten Abschnitt werden die Parameter des verwendeten Objektmodells erläutert. Danach werden die Systemkomponenten (b)-(g) in Abbildung 5.2 näher erläutert und abschließend die Interaktion der Systemkomponenten im Gesamtsystem beschrieben.

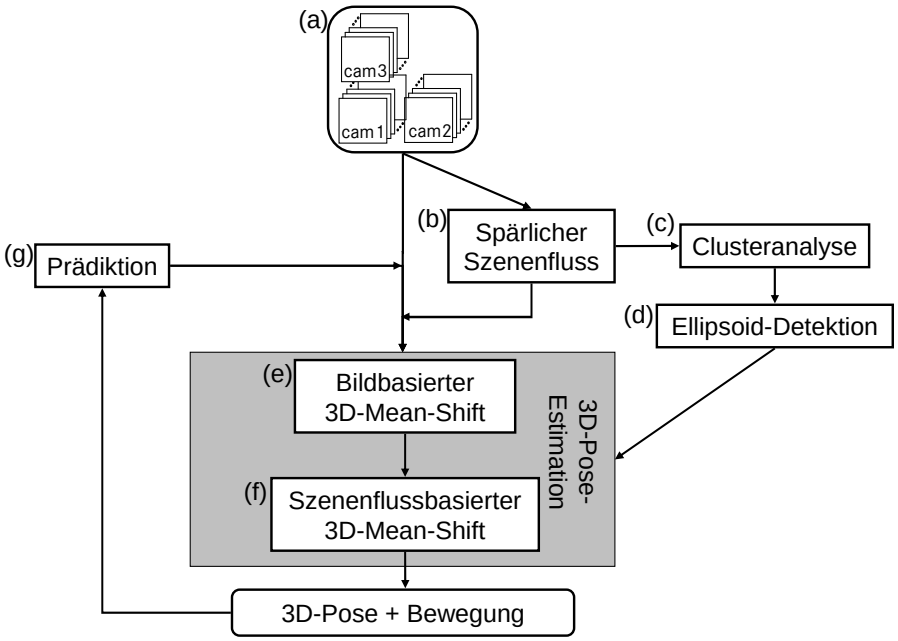

Abbildung 5.2: Struktur des Systemmodells zum 3D-Mean-Shift-Tracking.

5.1 Objektmodell

Es wird ein analytisches 3D-Modell (Abschnitt 2.1.3) zur Verfolgung der Objekte in den Bildern des Mehrkamerasystems eingesetzt. Bei diesem Trackingsystem werden die Objekte nicht detailliert beschrieben, sondern durch Ellipsoide modelliert. Falls ein in den Bildern des Kamerasystems verfolgtes Objekt nicht durch einen Ellipsoiden beschrieben werden kann, werden mehrere Ellipsoide zur Approximation des Objekts verwendet und diese anschließend in den Bildern der Kameras getrennt voneinander verfolgt.

Ein Ellipsoid wird in kartesischen Koordinaten mit folgender Gleichung beschrieben:

$$\frac{X^2}{l_X^2} + \frac{Y^2}{l_Y^2} + \frac{Z^2}{l_Z^2} = 1. \tag{5.1}$$

Dabei definieren die positiven reellen Zahlen l_X, l_Y und l_Z, die Längen der Halbachsen des Ellipsoiden. Die Längen der Halbachsen sind in diesem Trackingsystem fest vorgegeben und an die Größe einer menschlichen Hand angepasst. Diese Restriktion lässt sich aus dem bereits beschriebenen Systemkonzept ableiten. Es sollen vornehmlich menschliche Hände in den Bildern des Mehrkamerasystems verfolgt werden. Durch Rotation und Translation wird das Ellipsoid im Weltkoordinatensystems W positioniert. Folgender Parametervektor beschreibt die 3D-Pose eines Ellipsoiden:

$$\mathbf{\Phi} = [{}^W m_X, {}^W m_Y, {}^W m_Z, \beta]^T. \tag{5.2}$$

Der 3D-Punkt ${}^W\mathbf{m}$ definiert den Mittelpunkt des Ellipsoiden. Der Winkel β beschreibt die Rotation des Ellipsoiden um die um die Z-Achse des Weltkoordinatensystems W, welche durch

$\mathbf{R_Z}$ in Gleichung (3.4) definiert wird. Eine Rotation um die Y-Achse wird nicht modelliert. Dies bedeutet, dass die Ebene – beschrieben durch den Mittelpunkt des Ellipsoiden und die Halbachsen in X- und Y-Richtung – immer parallel zur XY-Ebene des Weltkoordinatensystems W liegt. Der Grund hierfür liegt in der 3D-Erweiterung des bildbasierten Mean-Shift-Algorithmus (Abschnitt 5.5), denn hier wird ein frontoparalleler flacher Ellipsoid an die Bilddaten des Mehrkamerasystems angepasst. Zur Pose-Estimation sind beim szenenflussbasierter 3D-Mean-Shift-Algorithmus (Abschnitt 5.6) außerdem 3D-Punktewolken mit Bewegungsinformation notwendig. Diese werden im Szenenflussmodul berechnet und im Folgenden beschrieben.

5.2 Spärlicher Szenenfluss

Der Szenenfluss ist ein 3D-Bewegungsfeld von Punkten in der Welt (Kamerakoordinatensystem der Masterkamera). Der optische Fluss ist im Vergleich zum Szenenfluss dessen Projektion in das Pixelkoordinatensystem einer der verwendeten Kameras. In der Literatur (Huguet u. Devernay, 2007) werden einige Verfahren zur Berechnung eines dichten Szenenfluss beschrieben, diese sind aber sehr rechenzeitaufwendig. Daher wird in diesem Systemmodul ein anderer Weg gewählt und durch Kombination von optischem Fluss und 3D-Stereo-Punktewolken ein nicht vollständiger sowie spärlicher Szenenfluss berechnet.

Optischer Fluss: In der Bildverarbeitung und in der optischen Messtechnik wird der optische Fluss als Vektorfeld bezeichnet, welches die Bewegungsrichtung und -geschwindigkeit für jeden Bildpunkt einer Bildsequenz angibt. Der optische Fluss kann als die auf die Bildebene projizierten Geschwindigkeitsvektoren von sichtbaren Objekten verstanden werden. In dieser Arbeit wird auf eine vorhandene Implementierung zur Berechnung des optische Flusses zurückgegriffen, welche von Wedel u. a. (2008) näher beschrieben wird. Um das 2D-Bewegungsfeld zu erhalten, wird für die Bilder der Masterkamera ein dichter optischer Fluss (Wedel u. a., 2008) mit Hilfe der Bilder der Zeitschritte t und $t + \Delta t$ berechnet. Der Ansatz basiert auf einer Weiterentwicklung einer Variationsmethode zur Berechnung des optischen Flusses (Horn u. Schunck, 1980).

Stereo: Zur Rekonstruktion der beobachteten 3D-Szene wird ein Ansatz aus der Stereobildverarbeitung eingesetzt. Durch ein sogenanntes Korrelationsstereo (Franke u. Joos, 2000) werden separat für das horizontale und vertikale Bildpaar des trinokularen Kamerasystems 3D-Punktewolken berechnet. Diese werden danach in einem zweiten Schritt im Kamerakoordinatensystem der Masterkamera fusioniert. Der Stereo-Algorithmus basiert auf der Korrelation von Bildausschnitten, was zu einer hohen Genauigkeit bei relativ geringer Rechenzeit führt. Subpixelgenauigkeit wird durch parabolischen Interpolation der Korrelationskoeffizienten erreicht.

Kombination: Der dichte optische Fluss und die 3D-Punktewolke werden kombiniert, um für jeden 3D-Punkt eine Bewegungsinformation zu erhalten, den sogenannten Szenenfluss. Im Gegensatz zum klassischen Szenenfluss (Huguet u. Devernay, 2007) ist in dieser Arbeit aufgrund des zweidimensionalen optischen Flusses die Geschwindigkeitskomponente in Tiefen-

richtung (Z-Richtung) nicht vorhanden. Durch Projektion der 3D-Stereopunkte in das Pixelkoordinatensystem der Masterkamera erhält man für jeden 3D-Punkt den optischen Fluss in Pixeln. Dieser optische Fluss wird unter der Annahme, dass keine Bewegung in Tiefenrichtung (Z-Richtung) stattfindet, mit Hilfe des Strahlensatzes in eine XY-Bewegung in die Einheit Meter pro Zeitschritt umgerechnet. XY-Bewegung bedeutet hierbei, dass die Geschwindigkeit nur in X- und Y-Richtung (horizontale und vertikale Ablage) direkt gemessen werden kann.

Der hier verwendete Szenenfluss ist nicht vollständig, da keine Bewegung in Tiefenrichtung geschätzt wird und er ist spärlich, da nur für die 3D-Punktewolke die Bewegungsinformation berechnet wird. In Abbildung 5.3 sind die projizierten 3D-Punkte sowie die projizierten XY-Geschwindigkeiten für die Masterkamera dargestellt. Diese direkte Kombination von optischem Fluss und 3D-Punktewolken hat den Vorteil, dass sie im Gegensatz zum klassischen Szenenfluss (Huguet u. Devernay, 2007) sehr effizient berechnet werden kann.

Das Ergebnis des Szenenflussmoduls ist eine Menge von Vektoren $\mathbf{s}_i \in S$ mit $i = 1, \ldots, N$. Dabei besteht jeder dieser Vektoren $\mathbf{s}_i = [{}^W s_X, {}^W s_Y, {}^W s_Z, {}^W \dot{s}_X, {}^W \dot{s}_Y]^T$ aus einem 3D-Punkt ${}^W\mathbf{s}$ und den Geschwindigkeiten $({}^W \dot{s}_X, {}^W \dot{s}_Y)$ in X- und Y-Richtung (horizontale und vertikale Ablage).

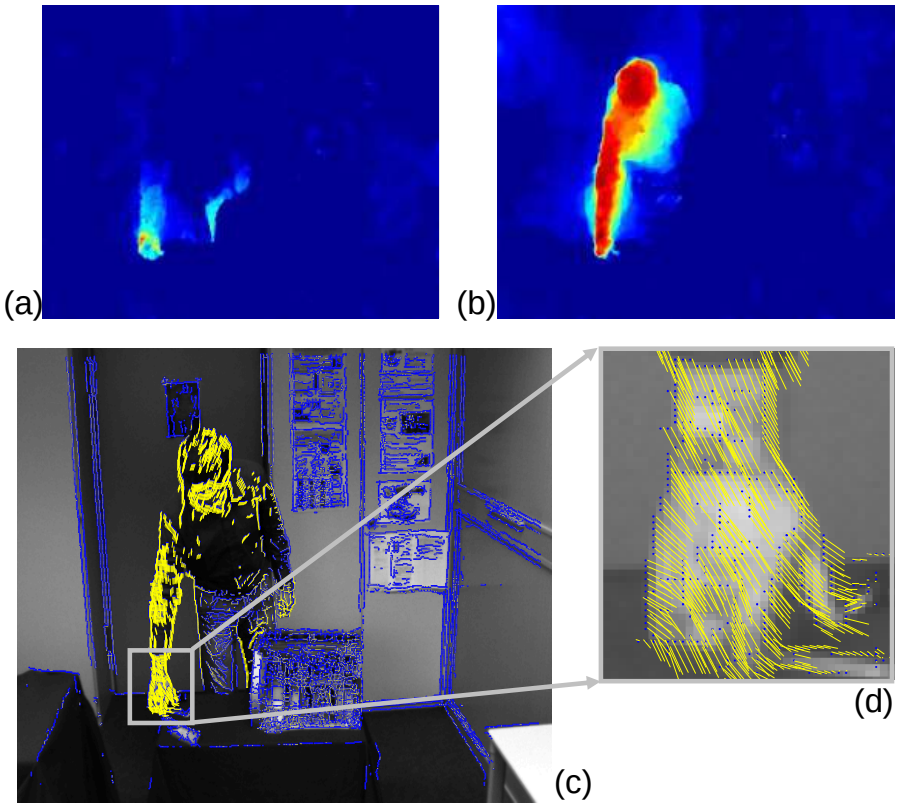

Abbildung 5.3: (a)+(b): Dichter optischer Fluss nach (Wedel u. a., 2008), (a) in horizontaler Richtung und (b) in vertikaler Richtung. Wärmere Farben bedeuten längere Flussvektoren. (c): In das Bild der Masterkamera projizierter spärlicher Szenenfluss. (d): Ein vergrößerter Ausschnitt des projizierten spärlichen Szenenfluss.

5.3 Clusteranalyse

Die Clusteranalyse wird auf den Datenvektoren $\mathbf{s}_i \in S$ mit $i = 1, \ldots, N$ aus dem Szenenflussmodul angewandt. Das Ziel der Auswertung besteht darin, die gegebene Menge $S = \mathbf{s}_1, \ldots, \mathbf{s}_N$ in Cluster (Teilmengen, Gruppen, Klassen) einzuteilen. Dabei sollen sich Objekte, die zum selben Cluster gehören, möglichst ähnlich (homogen) sowie zu verschiedenen Clustern gehörende Objekte möglichst unähnlich (heterogen) sein.

In dieser Arbeit wird ein zweistufiges Clustering angewandt. Die erste Stufe besteht aus einem graphenbasierten Clustering-Verfahren (Bock, 1974) und die zweite Stufe aus dem Mean-Shift-Clustering (Cheng, 1995).

1. Stufe: Graphenbasiertes Clustering Graphenbasierte Clustering-Methoden arbeiten auf der Distanzmatrix Δ für die N Objekte $\mathbf{s}_i \in S$ mit $i = 1, \ldots, N$. Zu jedem Objekt $\mathbf{s}_i \in S$ lässt sich eine so genannte d-Umgebung finden. Zu dieser gehören alle Objekte $\mathbf{s}_j \in S$, deren Abstand $d(\mathbf{s}_i, \mathbf{s}_j)$ kleiner oder gleich einer gewählten Schranke $d > 0$ ist. Daraus kann man den Begriff Gruppen der Stufe d ableiten. $A \subseteq S$ ist solch eine Gruppe, falls die folgenden Eigenschaften erfüllt sind (Bock, 1974):

1. $A \neq 0$
2. Ist $\mathbf{s}_i \in A$, so gehört auch die d-Umgebung von $\mathbf{s}_i$ zu A
3. Keine in A enthaltene Teilmenge $B \neq A$ soll 1 und 2 erfüllen

Ausgehend von einem Objekt $\mathbf{s}_i$ nimmt man alle Objekte in seiner d-Umgebung hinzu sowie die Objekte in deren d-Umgebungen. Ist so keine Erweiterung mehr möglich, ist eine Gruppe der Stufe d gefunden. Analog wird das nächste freie Objekt betrachtet bis alle $\mathbf{s}_i \in S$ zu einer Gruppe gehören.

In dieser Arbeit wird das graphenbasierte Clustering auf vierdimensionale Vektoren angewandt. Ein Input-Vektor $\mathbf{s}_i$ aus der Menge der Input-Daten S hat die Form $\mathbf{s}_i = [{}^W s_X, {}^W s_Y, {}^W s_Z, {}^W \dot{s}_X, {}^W \dot{s}_Y]^T$, wobei ${}^W\mathbf{s}$ die 3D-Koordinaten des Szenenfluss-Punktes beschreiben und $({}^W \dot{s}_X, {}^W \dot{s}_Y)$ für den Betrag aus den XY-Geschwindigkeiten steht. Weiterhin ist zu erwähnen, dass nur Punkte in das Clustering eingehen, die sich bewegen, d.h. deren Geschwindigkeitsbetrag $\sqrt{({}^W \dot{s}_X)^2 + ({}^W \dot{s}_Y)^2}$ größer als eine definierte Schwelle ist.

Abbildung 5.4 zeigt das Ergebnis der ersten Stufe der Clusteranalyse für alle sich bewegenden 3D-Punkte aus dem Szenenflussmodul. Ein Nachteil des graphenbasierten Clustering besteht darin, dass die gefundenen Cluster sehr groß werden können, wenn sich beispielsweise der zu verfolgende Arm eines Menschen homogen bewegt. Da in dieser Arbeit der Mean-Shift-Tracking-Algorithmus mit einem Ellipsoid-Modell angewandt wird, kann die Approximation des kompletten menschlichen Arms durch einen einzigen Ellipsoiden dazu führen, dass sehr viele Hintergrundpixel in das Zielmodell, welches das Aussehen des Objekts beschreibt, einfließen. Um dies zu vermeiden, werden in der zweiten Stufe des Clustering mit Hilfe des Mean-Shift-Clustering Cheng (1995) große Cluster in mehrere Cluster unterteilt, welche gut zur definierten Ellipsoidgröße passen. Dadurch kann außerdem die feste Definition der Halbachsen des Ellipsoiden direkt im Clustering berücksichtigt werden.

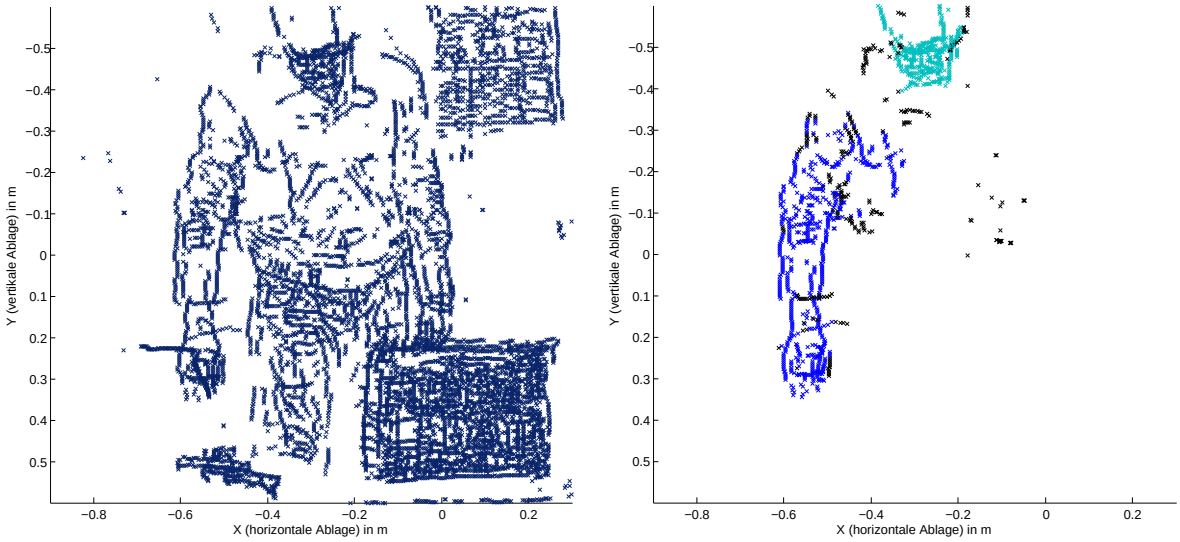

Abbildung 5.4: Links: 3D-Punktewolke der Szene. Rechts: Ergebnis der ersten Stufe der Clusteranalyse. Für alle bewegten 3D-Punkte aus dem Szenenflussmodul wurden zwei Cluster ermittelt (blau (Arm) und cyan (Kopf)). Alle 3D-Punkte, die zu keinem Cluster zugeordnet werden konnten sind schwarz dargestellt. Zur besseren Visualisierung ist der Geschwindigkeitsbetrag der 3D-Punkte nicht dargestellt.

2. Stufe: Mean-Shift-Clustering In der zweiten Stufe der Clusteranalyse wird der Mean-Shift-Clustering Ansatz nach Cheng (1995) verwendet. Dieser Ansatz findet die Clusterzentren durch einen oder mehrere Pfade, welche im selben Clusterzentrum münden. Ein Pfad entsteht durch Anwendung der Mean-Shift-Optimierung (Anhang A.3) von einem zufällig ausgewählten Datenpunkt aus (Abbildung 5.5). Der Fensterradius der Mean-Shift-Optimierung leitet sich aus den Längen der Halbachsen des Ellipsoiden ab. Der Algorithmus wird solange durchgeführt, bis alle 3D-Punkte zu einem Cluster zugeordnet wurden.

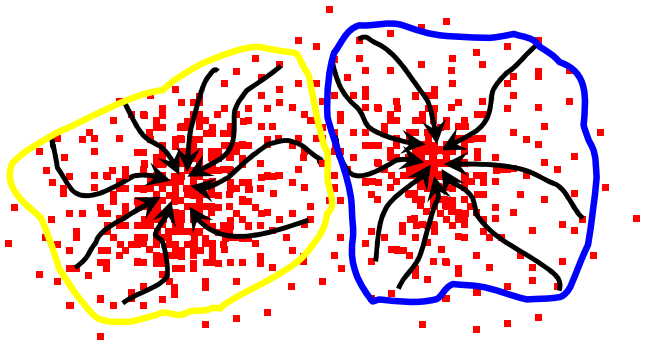

Abbildung 5.5: Mean-Shift-Clustering nach Cheng (1995). Clusterzentren ergeben sich durch einen oder mehrere Mean-Shift-Optimierungs-Pfade, welche im gleichen Clusterzentrum münden.

In dieser Arbeit wird das Mean-Shift-Clustering nur auf 3D-Punkten angewandt. Die Inputdaten stammen aus den vom graphenbasierten Clustering gefundenen Clustern. Es wird also auf jeden in der ersten Stufe gefundenen Cluster ein separates Mean-Shift-Clustering angewandt. Durch diesen Schritt werden zu große Cluster in mehrere kleine Cluster zerteilt, welche

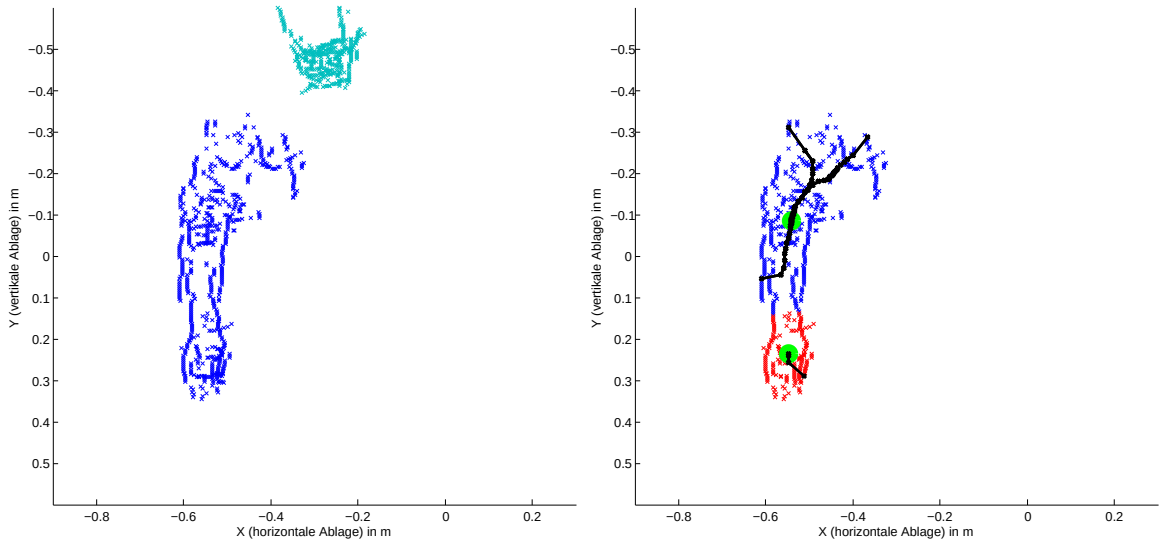

Abbildung 5.6: Links: Ergebnis der ersten Stufe der Clusteranalyse. Rechts: Ergebnis des Mean-Shift-Clustering auf dem Arm-Cluster (blauer Cluster auf linker Seite), es ist zu erkennen, dass dieser in zwei Unter-Cluster zerfällt. Die Pfade der Mean-Shift-Optimierung sind schwarz dargestellt.

besser zu den fest definierten Längen der Halbachsen des Ellipsoiden passen und sich somit gut zur Verfolgung mit dem 3D-Mean-Shift-Tracking eignen. Abbildung 5.6 zeigt das Ergebnis der Clusteranalyse der ersten und zweiten Stufe.

5.4 Ellipsoid-Detektion

Basierend auf den zu Clustern zugeordneten bewegten 3D-Punkten aus dem Szenenflussmodul, wird in diesem Systemmodul an jeden gefundenen Cluster ein Ellipsoid angepasst. Dabei haben alle berechneten Ellipsoide eine fest definierte Größe. Abbildung 5.7 zeigt das Endergebnis der zweistufigen Clusteranalyse und die an die Cluster angepassten Ellipsoide. Die Zentren der Cluster ergeben sich aus den Clusterzentren des Mean-Shift-Clustering. Die Orientierung des Ellipsoiden wird ermittelt, indem die Kovarianzmatrix $\mathbf{\Sigma}_{Clust}$ aus allen zum jeweiligen Cluster zugehörigen 3D-Punkten berechnet wird. Danach erfolgt die Berechnung der Eigenwerte λ_i und Eigenvektoren $\mathbf{v}_i$ von $\mathbf{\Sigma}_{Clust}$. Die Eigenvektoren $\mathbf{v}_i$ werden nach der Größe ihrer Eigenwerte sortiert und nur der Eigenvektor $\mathbf{v}_1$ mit dem größten Eigenwert wird zur Berechnung des Rotationswinkels um die Z-Achse verwendet. Der Winkel β beschreibt die Rotation um die Z-Achse und kann durch die folgende Gleichung berechnet werden:

$$\beta = \begin{cases} \pi - \arctan\left(\frac{v_{1Y}}{v_{1X}}\right) & \text{wenn } v_{1X} \neq 0 \\ \frac{\pi}{2} & \text{wenn } v_{1X} = 0 \end{cases} \tag{5.3}$$

Es wird nur die Rotation β um die Z-Achse geschätzt und die Rotation um die Y-Achse als Null angenommen, da in dieser Arbeit beim bildbasierten 3D-Mean-Shift-Algorithmus ange-

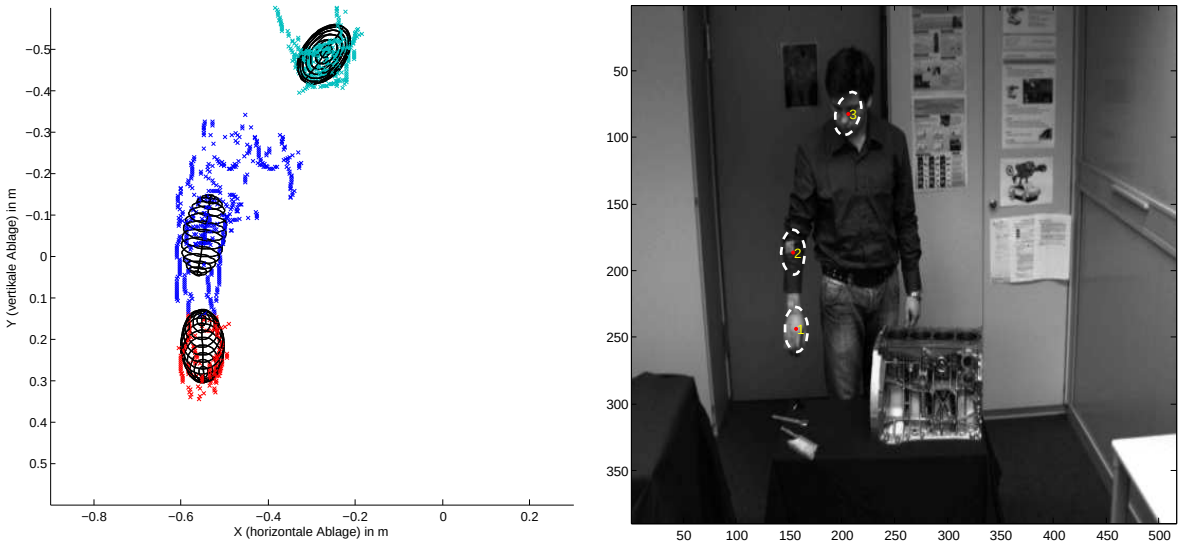

Abbildung 5.7: Links: Endergebnis der Clusteranalyse und die in die Cluster gefitteten Ellipsoide. Rechts: Projektion der gefitteten Ellipsoide in das Bild der Masterkamera.

nommen wird, dass das Ellipsoid flach und frontoparallel zur XY-Ebene des Kamerakoordinatensystems ist. Das Ergebnis der Ellipsoid-Detektion zum Zeitschritt t für einen Ellipsoiden i ist also eine 3D-Pose $\mathbf{\Phi}^{(i)}(t) = [{}^W m_x, {}^W m_y, {}^W m_z, \beta]^T$.

5.5 Bildbasierter 3D-Mean-Shift

In diesem Systemmodul wird basierend auf einer initialen 3D-Pose eines Ellipsoiden i, mit dem bildbasierten 3D-Mean-Shift-Algorithmus eine neue 3D-Pose $\widetilde{\mathbf{\Phi}}^{(i)}(t) = [{}^W \widetilde{m}_X, {}^W \widetilde{m}_Y, {}^W \widetilde{m}_Z, \widetilde{\beta}]^T$ für den aktuellen Zeitschritt t berechnet. Falls das Objekt neu detektiert wurde, stammt die Initialisierung aus dem Detektionsmodul (Abschnitt 5.4). Falls das in den Bildern verfolgte Objekt bereits bekannt ist, ergibt sich die Pose-Initialisierung $\widehat{\mathbf{\Phi}}^{(i)}(t)$ aus dem Prädiktionsmodul (Abschnitt 5.7). Die 3D-Pose $\widetilde{\mathbf{\Phi}}^{(i)}(t)$ eines zu verfolgenden Ellipsoiden i, bestehend aus Mittelpunkt ${}^W\mathbf{m}$ und Rotation um die Z-Achse β, wird mit Hilfe des vorher definierten Zielmodells $\widehat{\mathbf{q}}_i$ an die Bilddaten angepasst. Um ein zeitlich robustes Tracking zu realisieren, wird als Bildvorverarbeitung eine Hintergrundsubtraktion der Kamerabilder durchgeführt.

5.5.1 Bildvorverarbeitung

Die zur Bildvorverarbeitung verwendete Hintergrundsubtraktion (engl. *Background Subtraction*) wird über Differenzbilder zweier aufeinander folgender Zeitschritte realisiert. Für ein von der Kamera c zum Zeitschritt t geliefertes Bild $\mathbf{I}_{c,t}$ werden mit $\left|\mathbf{I}_{c,t} - \mathbf{I}_{c,(t-1)}\right| > \rho$ alle Pixel berechnet, deren zeitliche Änderung des Pixelwertes eine Schwelle ρ überschreiten. Mit Hilfe der Differenzbilder lassen sich Bewegungen sichtbar machen, da das Kamerasystem in unserer Applikation immer einen festen Standpunkt hat. Solange sich am Bildinhalt nichts verän-

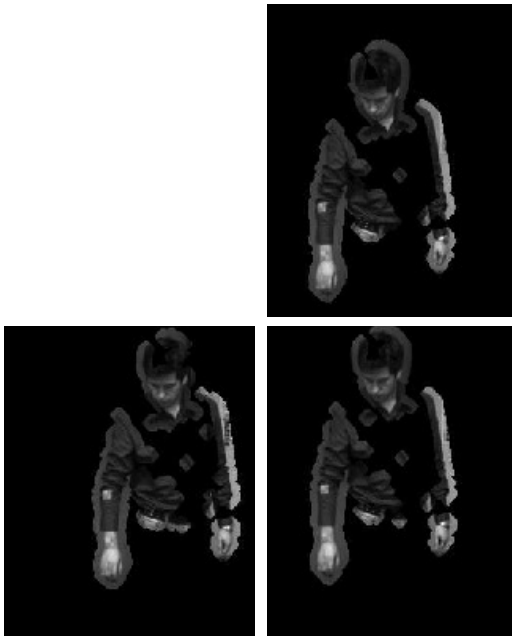

Abbildung 5.8: Hintergrundsubtraktion durch Maskierung, dabei berechnet sich die Maske aus Differenzbildern. Bewegungen werden sichtbar und das Tracking robuster.

dert, bleibt die Differenz Null. Jede Veränderung ergibt jedoch ein Differenzsignal, wodurch sich Veränderungen lokalisieren lassen. Durch die Schwellwertoperationen wird aus dem Differenzbild ein Binärbild, welches durch Anwendung eines Median-Filters sowie einer Dilatation aufbereitet und dann als Maske auf das aktuelle Bild angewandt wird (Abbildung 5.8).

Wie in Anhang A.3.2 beschrieben, basiert das Mean-Shift-Tracking auf einem Zielmodell $\widehat{\mathbf{q}}_i$, welches mit Hilfe eines Merkmalshistogramms das zu verfolgende Objekt beschreibt. Der erste Schritt des bildbasierten 3D-Mean-Shift-Trackings besteht also in der Berechnung des Zielmodells $\widehat{\mathbf{q}}_i$.

5.5.2 Berechnung des Zielmodells

Um das Zielmodell $\widehat{\mathbf{q}}_i$ zu berechnen, ist eine 3D-Pose $\mathbf{\Phi}^{(i)}(t) = [{}^W m_X, {}^W m_Y, {}^W m_Z, \beta]^T$ eines in Bildsequenzen zu verfolgenden Ellipsoiden i für den ersten Zeitschritt t der Verfolgung notwendig. Daher wird das Zielmodell $\widehat{\mathbf{q}}_i$ direkt aus der 3D-Pose $\mathbf{\Phi}^{(i)}(t)$ der Ellipsoid-Detektion (Abschnitt 5.4) berechnet. Es wird dabei die Annahme gemacht, dass das zu verfolgende Objekt für alle 3D-Punkte auf dem Modell die gleiche Tiefe hat, also flach ist. Dies bedeutet, dass jeder sichtbare 3D-Punkt auf der Oberfläche des Ellipsoiden aus dem 3D-Kamerakoordinatensystem in die Pixelkoordinatensysteme der verwendeten Kameras projiziert werden kann. Durch die Projektion können zu jedem 3D-Punkt auf der Modelloberfläche die Pixelwerte in allen drei Kamerabildern ermittelt werden. Abbildung 5.9 zeigt, wie die Pixelwerte für alle 3D-Punkte auf der sichtbaren Oberfläche des Ellipsoiden ermittelt werden und daraus das Merkmalshistogramm des Zielmodells $\widehat{\mathbf{q}}_i$ berechnet wird. Das Merkmalshistogramm repräsentiert relative Häufigkeiten, welche als Wahrschscheinlichkeiten interpretiert werden. Diese beschreiben

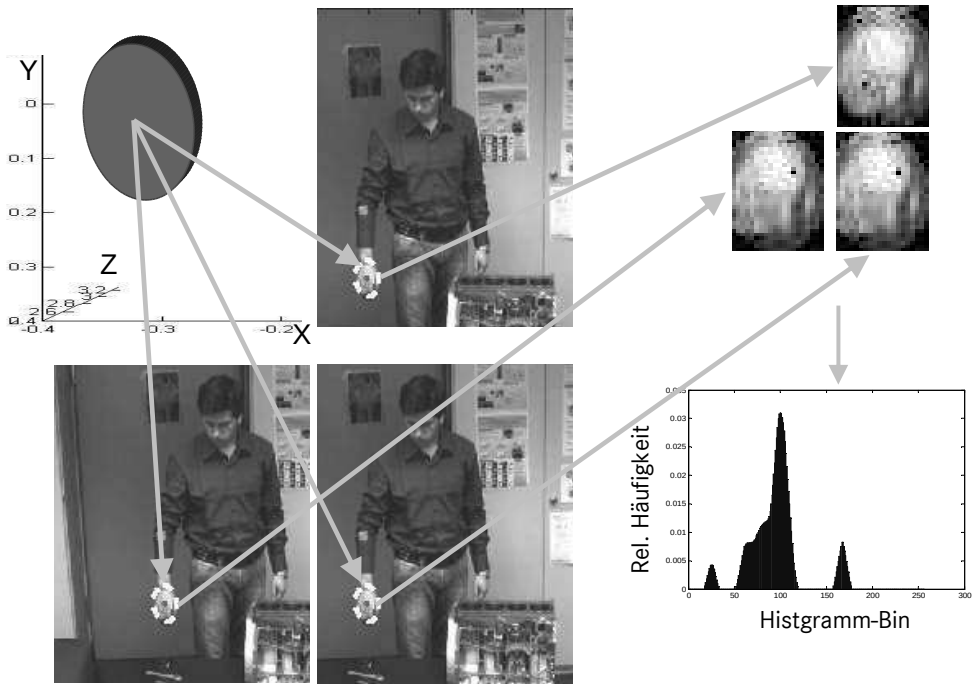

Abbildung 5.9: Berechnung des Zielmodells für den 3D-Mean-Shift-Tracking-Algorithmus. Mit allen 3D-Punkten innerhalb des frontoparallelen Ellipsoiden und ihren zugehörigen Pixelwerten wird ein Histogramm berechnet. Dieses beschreibt die Verteilung der Pixelwerte in den drei Kamerabildern innerhalb des Ellipsoiden.

die Häufigkeit des Pixelwertes im Histogramm und sagen aus, wie wahrscheinlich es ist, dass ein Pixel mit dem jeweiligen Wert zum Zielobjekt gehört. Durch den bildbasierten 3D-Mean-Shift-Algorithmus wird in den Folgebildern mit Hilfe des Merkmalshistogramms die Region gesucht, die ein ähnliches Histogramm aufweist. Da Merkmalshistogramme keine räumliche Information repräsentieren, ist das Mean-Shift-Tracking sehr robust gegenüber Änderungen der Objektgestalt, welche z.B. durch perspektivische Verzerrungen, wechselnde Lichtverhältnisse oder Objektverformungen auftreten können.

Durch Verwendung verschiedener Farbräume und unterschiedlicher Dimensionen des Histogramms werden in dieser Arbeit zwei unterschiedliche Ansätze zur Erzeugung des Merkmalshistogramms untersucht. Eine sogenannte Berechnungsvorschrift beschreibt dabei, wie einem 3D-Punkt ${}^{W}\mathbf{x} = (X, Y, Z)^T$ das jeweilige Histogramm-Bin (Intervall auf der Abszissenachse des Histogramms) mit Hilfe der projizierten 2D-Koordinaten ${}^{P_c}\mathbf{u} = (u, v)^T$ im Pixelkoordinatensystem P_c der Kamera c und der entsprechenden Pixelwerte des verwendeten Farbraums zugeordnet werden. Das Merkmalshistogramm wird mit Hilfe der Berechnungsvorschrift aus allen 3D-Punkten des flachen Ellipsoid-Modells erzeugt. Es werden zwei Systemvarianten unterschieden, bei der Ersten basiert die Berechnung des Zielmodells auf Grauwertbildern und bei der Zweiten auf der Hue-Komponente des HSV-Farbraums (Gonzalez u. Woods, 2002) wie bei Bradski (1998) auch. Im Folgenden werden beide Varianten zur Berechnung des Zielmodells $\widehat{\mathbf{q}}$ vorgestellt.

1. Variante – eindimensionales Grauwert-Histogramm:
- Anzahl Histogramm-Bins: $N_{Bin} = 256$
- Farbraum: Intensitäts-Farbraum
- Berechnungsvorschrift:

$$iBin({}^{W}\mathbf{x}) = \frac{1}{3}\left(\mathbf{I}_{C_1}({}^{P_1}\mathbf{u}) + \mathbf{I}_{C_2}({}^{P_2}\mathbf{u}) + \mathbf{I}_{C_3}({}^{P_3}\mathbf{u})\right) \tag{5.4}$$

Dabei steht $\mathbf{I}_{C_c}({}^{P_c}\mathbf{u})$ für den Intensitätswert (Grauwert) im Bild der Kamera c an der Stelle ${}^{P_c}\mathbf{u}$.

2. Variante – eindimensionales Hue-Histogramm:
- Anzahl Histogramm-Bins: $N_{Bin} = 256$
- Farbraum: HSV-Farbraum (Gonzalez u. Woods, 2002), nur Hue-Komponente
- Berechnungsvorschrift:

$$iBin({}^{W}\mathbf{x}) \;=\; \frac{1}{3}\left(\mathbf{H}_{C_1}({}^{P_1}\mathbf{u}) + \mathbf{H}_{C_2}({}^{P_2}\mathbf{u}) + \mathbf{H}_{C_3}({}^{P_3}\mathbf{u})\right) \tag{5.5}$$

Dabei steht $\mathbf{H}_{C_c}({}^{P_c}\mathbf{u})$ für den Hue-Wert im HSV-Bild der Kamera c an der Stelle ${}^{P_c}\mathbf{u}$.

Mit Hilfe der jeweiligen Eingabedaten (Bilder im definierten Farbraum) und der Berechnungsvorschrift kann nun mit allen 3D-Punkten innerhalb des frontoparallelen Ellipsoiden ein Histogramm über die Verteilung der Pixelwerte berechnet werden. Dieses Histogramm wird anschließend normiert, sodass $\sum_{n=1}^{N_{Bin}} \widehat{\mathbf{q}}(n) = 1$ gilt, dadurch entstehen relative Häufigkeiten für das Vorkommen von Pixelwerten, welche als Wahrscheinlichkeiten interpretiert werden. Mit $\widehat{\mathbf{q}}(iBin({}^{W}\mathbf{x}))$ kann für einen 3D-Punkt ${}^{W}\mathbf{x}$ das zugehörige Histogramm-Bin und somit die assoziierte relative Häufigkeit berechnet werden.

5.5.3 Anpassung des Modells

Das Zielmodell $\widehat{\mathbf{q}}_i$ eines in den Kamerabildern verfolgten Ellipsoiden i wird verwendet, um den Mean-Shift-Algorithmus in einem 3D-Suchbereich, mit der prädizierten 3D-Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$ als Zentrum, effektiv anwenden zu können. Bei der 3D-Erweiterung des Mean-Shift-Trackings ist der 3D-Suchbereich nicht durch Bildkoordinaten, sondern durch 3D-Weltkoordinaten definiert. Der Berechnungsbereich ist ein frontoparalleler 3D-Bereich $\mathbf{R}$ mit definiertem Raster. Dies bedeutet, dass der 3D-Bereich $\mathbf{R}$ parallel zur XY-Ebene des Kamerakoordinatensystems ist, wodurch jeder 3D-Punkt auf dem Raster $\mathbf{R}$ die selbe Tiefe besitzt. Jedem 3D-Rasterpunkt ${}^{W}\mathbf{r}$ wird mit Hilfe des jeweiligen Zielmodells $\widehat{\mathbf{q}}_i$ und der entsprechenden Berechnungsvorschrift, eine relative Häufigkeit im Intervall $[0, 1]$ zugewiesen. Das Merkmalshistogramm des Zielmodells $\widehat{\mathbf{q}}_i$ wird dabei als *Lookup-Table* verwendet. Mit $\widehat{\mathbf{q}}_i(iBin({}^{W}\mathbf{r}))$ kann für einen 3D-Punkt ${}^{W}\mathbf{r}$ das zugehörige Histogramm-Bin und somit die assoziierte relative Häufigkeit berechnet werden. Der Suchbereich ist abhängig von der prädizierten 3D-Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$ und die Rastergröße definiert sich durch die Pixelauflösung, welche abhängig von der Tiefe (Z-Koordinate) des Objekts ist. Abbildung 5.10 (b) zeigt den Suchbereich und das zugehörige 3D-Raster mit berechneten Wahrscheinlichkeiten für jeden 3D-Rasterpunkt.

Das Ellipsoidmodell wird nun mit dem Mean-Shift-Algorithmus (Abschnitt A.3.1) an das 3D-Raster $\mathbf{R}$ angepasst, indem die Region mit der höchsten Wahrscheinlichkeit gesucht wird. Die Startposition der Mean-Shift-Optimierung basiert auf dem Mittelpunkt ${}^{W}\mathbf{m}$ aus der prädizierten Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$, diese ergibt sich aus dem Prädiktionsmodul (Abschnitt 5.7). Die Optimierung wird in diesem Fall nicht mit einem Ellipsoiden, sondern mit einer Ellipse durchgeführt, da das Wahrscheinlichkeitsraster $\mathbf{R}$ flach ist, also keine Ausdehnung in die Tiefe besitzt. Daher wird bei der folgenden Beschreibung die Z-Koordiante vernachlässigt, da diese für alle Rasterpunkte konstant ist.

Die folgende Beschreibung des Algorithmus ist ähnlich zum CAMShift-Tracking nach Bradski (1998) (Anhang A.3.2). Der Unterschied besteht darin, dass hier mit 3D-Koordinaten gerechnet wird und das Zielmodell über die definierte Berechnungsvorschrift direkt aus den drei Kamerabildern berechnet wird. Momente der Ordnung p, q innerhalb eines elliptischen Bereichs auf der diskreten frontoparallelen 2D-Wahrscheinlichkeitsverteilung $\mathbf{R}$ sind definiert als

$$m_{pq}({}^{W}\widetilde{\mathbf{m}}) = \sum_{(X,Y)\in E({}^{W}\widetilde{\mathbf{m}},l_X,l_Y)} \mathbf{R}(X,Y)\cdot X^p \cdot V^q. \tag{5.6}$$

Dieser Ausdruck beschreibt das Moment der Ordnung p, q für den elliptischen Bereich $E({}^{W}\widetilde{\mathbf{m}}, l_X, l_Y)$ in der diskreten 2D-Wahrscheinlichkeitsverteilung $\mathbf{R}$, wobei $\mathbf{R}(X,Y)$ dem Wahrscheinlichkeitswert an der Position (X,Y) entspricht. Der elliptische Bereich $E({}^{W}\mathbf{m}, l_X, l_Y)$ ist dabei abhängig vom Mittelpunkt ${}^{W}\mathbf{m}$ und der Länge der Halbachsen l_X und l_y. Das Moment nullter Ordnung beschreibt die Fläche einer Verteilung und wird folgendermaßen berechnet:

$$m_{00}({}^{W}\widetilde{\mathbf{m}}) = \sum_{(X,Y)\in E({}^{W}\widetilde{\mathbf{m}},l_X,l_Y)} \mathbf{R}(X,Y). \tag{5.7}$$

Um die Position ${}^{W}\widetilde{\mathbf{m}}$ des Maximums innerhalb der elliptischen Region zu ermitteln, benötigt man neben $m_{00}({}^{W}\widetilde{\mathbf{m}})$ noch die Momente erster Ordnung

$$m_{10}({}^{W}\widetilde{\mathbf{m}}) = \sum_{(X,Y)\in E({}^{W}\widetilde{\mathbf{m}},l_X,l_Y)} \mathbf{R}(X,Y)\cdot X \tag{5.8}$$

und

$$m_{01}({}^{W}\widetilde{\mathbf{m}}) = \sum_{(X,Y)\in E({}^{W}\widetilde{\mathbf{m}},l_X,l_Y)} \mathbf{R}(X,Y)\cdot Y. \tag{5.9}$$

Somit kann der Schwerpunkt ${}^{W}\widetilde{\mathbf{m}}_{j+1}$ der elliptischen Region berechnet werden.

$${}^{W}\widetilde{\mathbf{m}}_{j+1} = \begin{pmatrix} \frac{m_{10}({}^{W}\widetilde{\mathbf{m}}_j)}{m_{00}({}^{W}\widetilde{\mathbf{m}}_j)} \\ \frac{m_{01}({}^{W}\widetilde{\mathbf{m}}_j)}{m_{00}({}^{W}\widetilde{\mathbf{m}}_j)} \\ {}^{W}m_{Z,j} \end{pmatrix} \quad j = 2,3,\ldots \tag{5.10}$$

Dies entspricht dem Positionsupdate des Mittelpunkts beim CAMShift-Algorithmus (Anhang A.3.2). Die Größe der elliptischen Region ist abhängig von der Größe der frontoparallelen 3D-Ellipse. Die Initialisierung des Mittelpunkts, also ${}^{W}\widetilde{\mathbf{m}}_{j=1}$ ergibt sich aus dem Mittelpunkt ${}^{W}\widehat{\mathbf{m}}$ der prädizierten Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$ des Ellipsoiden i zum Zeitschritt t. Die geschätzte Tiefe des Objekts bleibt dabei immer gleich.

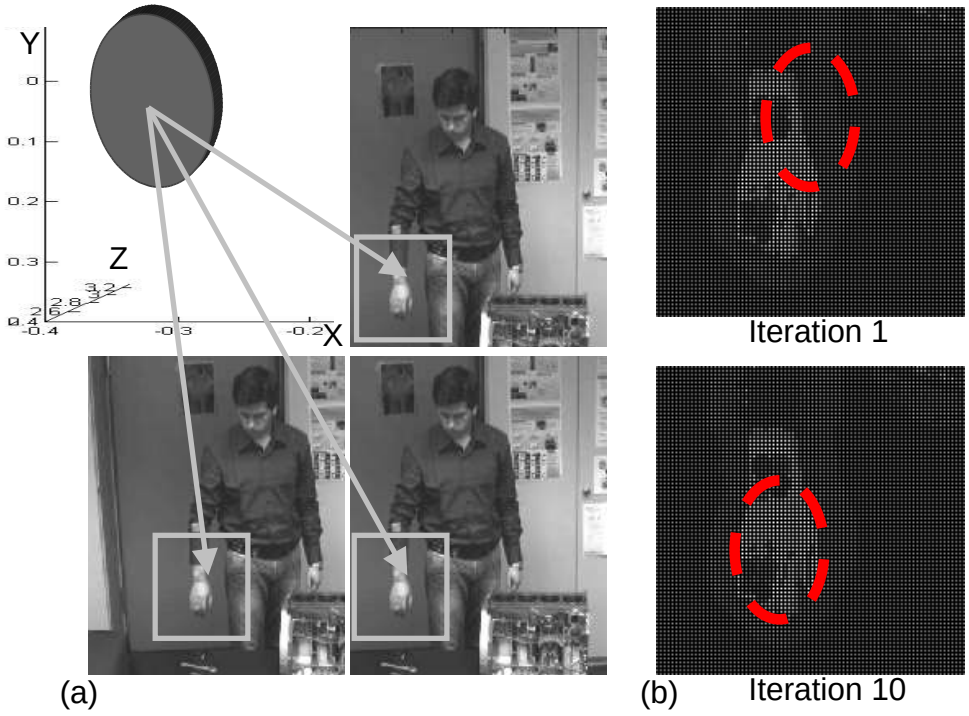

Abbildung 5.10: (a): Suchbereich in den drei Kamerabildern, abhängig von der prädizierten Objektposition $\widehat{\mathbf{\Phi}}^{(i)}(t)$. (b): Anpassung der XY-Position des Ellipsoids mit Hilfe des Wahrscheinlichkeitsrasters $\mathbf{R}$. Je höher die relative Häufigkeit (Wahrscheinlichkeit), desto heller ist der 3D-Rasterpunkt. Außerdem ist die Startposition (Iteration 1) und das Endergebnis der Mean-Shift-Optimierung (Iteration 10) dargestellt.

Bei der Anpassung des Zielmodells wird in jedem Iterationsschritt auch die Orientierung der Ellipse auf dem Wahrscheinlichkeitsraster $\mathbf{R}$ geschätzt. Diese Orientierung entspricht der Rotation um die Z-Achse $\widetilde{\beta}$. Aus dem Wahrscheinlichkeitsraster $\mathbf{R}$ (Abbildung 5.10(b)) wird $\widetilde{\beta}$ berechnet, indem die Kovarianzmatrix $\mathbf{\Sigma}_{E({}^W\mathbf{m},l_X,l_Y)}$ aus allen Raster-Positionen innerhalb des elliptischen Bereichs $E({}^W\mathbf{m},l_X,l_Y)$, welche eine relative Häufigkeit überschreiten, berechnet wird. Aus $\mathbf{\Sigma}_{E({}^W\mathbf{m},l_X,l_Y)}$ lassen sich durch Lösung des Eigenwertproblems die Eigenwerte λ und Eigenvektoren $\mathbf{v}_i$ berechnen. Die Eigenvektoren $\mathbf{v}_i$ werden nach der Größe ihrer Eigenwerte sortiert und nur der Eigenvektor $\mathbf{v}_1$ mit dem größten Eigenwert wird zur Berechnung des Rotationswinkels um die Z-Achse verwendet. Der Winkel β beschreibt die Rotation um die Z-Achse und kann durch folgende Gleichung berechnet werden:

$$\widetilde{\beta} = \begin{cases} \pi - \arctan\left(\frac{v_{1Y}}{v_{1X}}\right) & \text{wenn } v_{1X} \neq 0 \\ \frac{\pi}{2} & \text{wenn } v_{1X} = 0 \end{cases} \tag{5.11}$$

Nach mehreren Iterationen ergibt sich eine Ellipsoid-Position $\widetilde{\mathbf{\Phi}}^{(i)}(t) = [{}^W\widetilde{m}_X, {}^W\widetilde{m}_Y, {}^W\widetilde{m}_Z, \widetilde{\beta}]^T$ des Objekts i zum Zeitpunkt t, welche das Ergebnis des bildbasierten 3D-Mean-Shift-Algorithmus darstellt. Die geschätzte Tiefe unterscheidet sich dabei nicht von der Tiefe der Initialisierung. Um die Objekttiefe zu berechnen und die bildbasierte Schätzung weiter zu verbessern, wird im Anschluss eine szenenflussbasierte 3D-Mean-Shift-Optimierung durchgeführt.

Die bildbasierte Schätzung $\widetilde{\mathbf{\Phi}}^{(i)}(t)$ dient dabei als Initialisierung und ist sehr wichtig für das Tracking, wenn keine oder wenige bewegte 3D-Punkte vorhanden sind.

5.6 Szenenflussbasierter 3D-Mean-Shift

Die Initialisierung des szenenflussbasierten 3D-Mean-Shift-Algorithmus ergibt sich aus dem Ergebnis der bildbasierten 3D-Mean-Shift-Optimierung $\widetilde{\mathbf{\Phi}}^{(i)}(t)$. Beim szenenflussbasierten 3D-Mean-Shift-Algorithmus wird die Mean-Shift-Optimierung (Anhang A.3) mit einem Ellipsoid auf allen sich bewegenden 3D-Punkten aus dem Szenenflussmodul durchgeführt und das Ellipsoid somit in die 3D-Daten gefittet. Dieses Systemmodul ist notwendig, da bei der bildbasierten 3D-Mean-Shift-Optimierung (Abschnitt 5.5) keine Tiefe geschätzt wird und erhöht außerdem die Robustheit des Trackings. Der szenenflussbasierten 3D-Mean-Shift-Algorithmus wird nur ausgeführt, wenn sich bewegte 3D-Punkte innerhalb des Ellipsoiden aus der bildbasierten Schätzung befinden.

Der folgende Ablauf beschreibt die Schritte des szenenflussbasierten 3D-Mean-Shift-Algorithmus:

1. Schritt: Im ersten Schritt des Algorithmus wird die Bandbreite des Kernels gewählt. Diese leitet sich direkt aus der Länge der Halbachsen des getrackten Ellipsoiden ab. Die Initialisierung (Iteration $j = 1$) des Mittelpunkts ${}^{W}\mathbf{m}_{j=1}$ der Mean-Shift-Optimierung ergibt sich aus dem geschätzten Mittelpunkt ${}^{W}\widetilde{\mathbf{m}}$ des bildbasierten 3D-Mean-Shift-Algorithmus.

2. Schritt: Im Schritt 2 wird das Kernelfenster mit dem Mean-Shift-Vektor auf eine neue Position verschoben. Die neue Position des Suchfensters ${}^{W}\mathbf{m}_{j+1}$ wird durch die Gleichung

$$ {}^{W}\mathbf{m}_{j+1} = \frac{\sum_{n=1}^{N} {}^{W}\mathbf{s}_n \cdot g\left({}^{W}\mathbf{s}_n, {}^{W}\mathbf{m}_j, l_X, l_Y, l_Z\right) \cdot \widehat{\mathbf{q}}_i(iBin({}^{W}\mathbf{s}_n))}{\sum_{n=1}^{N} g\left({}^{W}\mathbf{s}_n, {}^{W}\mathbf{m}_j, l_X, l_Y, l_Z\right) \cdot \widehat{\mathbf{q}}_i(iBin({}^{W}\mathbf{s}_n))} \quad j = 2, 3, ... \tag{5.12} $$

berechnet, wobei ${}^{W}\mathbf{m}_j$ die alte Position des Kernelfensters darstellt. Das Gewicht $g\left({}^{W}\mathbf{s}_n, {}^{W}\mathbf{m}_j, l_X, l_Y, l_Z\right)$ des Kernels ist abhängig vom 3D-Szenenfluss-Punkt ${}^{W}\mathbf{s}_n$, dem vorherigen Mittelpunkt ${}^{W}\mathbf{m}_j$ und den Längen der Halbachsen (l_X, l_Y, l_Z) des Ellipsoiden. Der dabei verwendete Kernel ist als abgeschnittener Gauß-Kernel (Cheng, 1995) modelliert. Dieser zweite Schritt wird bis zur Konvergenz wiederholt.

Mit allen 3D-Punkten ${}^{W}\mathbf{s}_n$ innerhalb des 3D-Ellipsoiden wird die Rotation um die Z-Achse berechnet, indem die Kovarianzmatrix $\mathbf{\Sigma}_{3D}$ berechnet wird. Aus $\mathbf{\Sigma}_{3D}$ lassen sich durch Lösung des Eigenwertproblems die Eigenwerte λ und Eigenvektoren $\mathbf{v}_i$ berechnen. Die Eigenvektoren $\mathbf{v}_i$ werden nach der Größe ihrer Eigenwerte sortiert und nur der Eigenvektor $\mathbf{v}_1$ mit dem größten Eigenwert wird zur Berechnung des Rotationswinkels um die Z-Achse verwendet. Der Winkel β beschreibt die Rotation um die Z-Achse und kann durch folgende Gleichung berechnet werden:

$$ \beta = \begin{cases} \pi - \arctan\left(\frac{v_{1Y}}{v_{1X}}\right) & \text{wenn } v_{1X} \neq 0 \\ \frac{\pi}{2} & \text{wenn } v_{1X} = 0 \end{cases} \tag{5.13} $$

Abbildung 5.11 zeigt die Optimierung mit einem Ellipsoiden für drei verschiedene Iterationsschritte. Das Ergebnis dieses Moduls ist für jedes Objekt i eine 3D-Pose $\mathbf{\Phi}^{(i)}(t)$ zum aktuel-

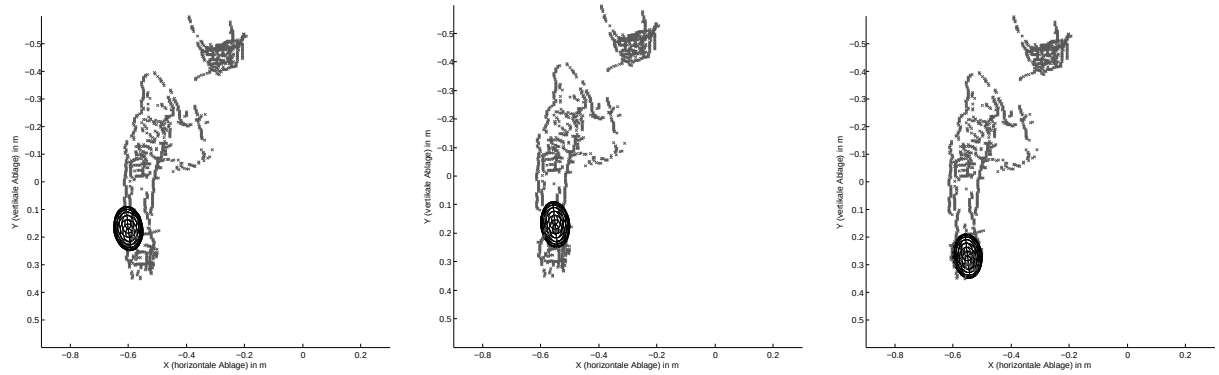

Abbildung 5.11: 3D-Mean-Shift-Optimierung auf einer Menge von 3D-Punkten für die Iterationsschritte 1, 3 und 6 (von links nach rechts). Das Ergebnis der jeweiligen Iteration ist als schwarzer Ellipsoid dargestellt.

len Zeitschritt t. Unter Nutzung dieser Pose und der Szenenfluss-Vektoren $\mathbf{s}_i \in S$ kann die XY-Geschwindigkeit $({}^W\dot{m}_X, {}^W\dot{m}_Y)$ des Objektmittelpunkts ermittelt werden. Diese wird zur Prädiktion der Pose des Ellipsoiden verwendet.

5.7 Prädiktion

Dieses Systemmodul (Abbildung 5.2) dient zur Prädiktion der Objekte über die Zeit. Mit der Annahme einer Bewegung mit konstanter Geschwindigkeit wird die prädizierte 3D-Pose $\widehat{\mathbf{\Phi}}^{(i)}(t + \Delta t)$ eines getrackten Ellipsoids i zum Zeitschritt $(t + \Delta t)$ folgendermaßen berechnet:

$$\widehat{\mathbf{\Phi}}^{(i)}(t + \Delta t) = \mathbf{\Phi}^{(i)}(t) + ({}^W\dot{m}_X, {}^W\dot{m}_Y, 0, 0)^T \cdot \Delta t. \tag{5.14}$$

Dabei steht $\mathbf{\Phi}^{(i)}(t)$ für die aktuell geschätzte 3D-Pose und $({}^W\dot{m}_X, {}^W\dot{m}_Y, 0, 0)^T$ für die zeitliche Ableitung des Mittelpunkts der Pose in X- und Y-Richtung, welche aus den Szenenfluss-Daten ermittelt wurde. Die Änderung von Tiefe und Rotation um die Z-Achse wird als Null angenommen, da diese in den Szenenfluss-Daten nicht vorhanden ist und auch kein zeitliches Filter zur Schätzung eingesetzt wird.

5.8 Interaktion im Gesamtsystem

Das in diesem Kapitel entwickelte Trackingsystem (Abbildung 5.2) realisiert die Verfolgung eines oder mehrerer Objekte über die Zeit in Bilddaten und 3D-Punktewolken. Der Ansatz leistet eine robuste und effiziente Verfolgung von Objekten, indem diese als Ellipsoide modelliert, automatisch detektiert, getrackt und wenn nötig reinitialisiert werden. Im Folgenden beziehen sich alle Modulangaben auf Abbildung 5.12, hier sind auch typische Ausgaben einzelner Module des Systems dargestellt.

Den Input für alle Systemmodule liefert das trinokulare Kamerasystem (Abschnitt 3.1). Die Pose-Estimation besteht aus zwei Modulen (e) und (f). Der bildbasierte 3D-Mean-Shift-

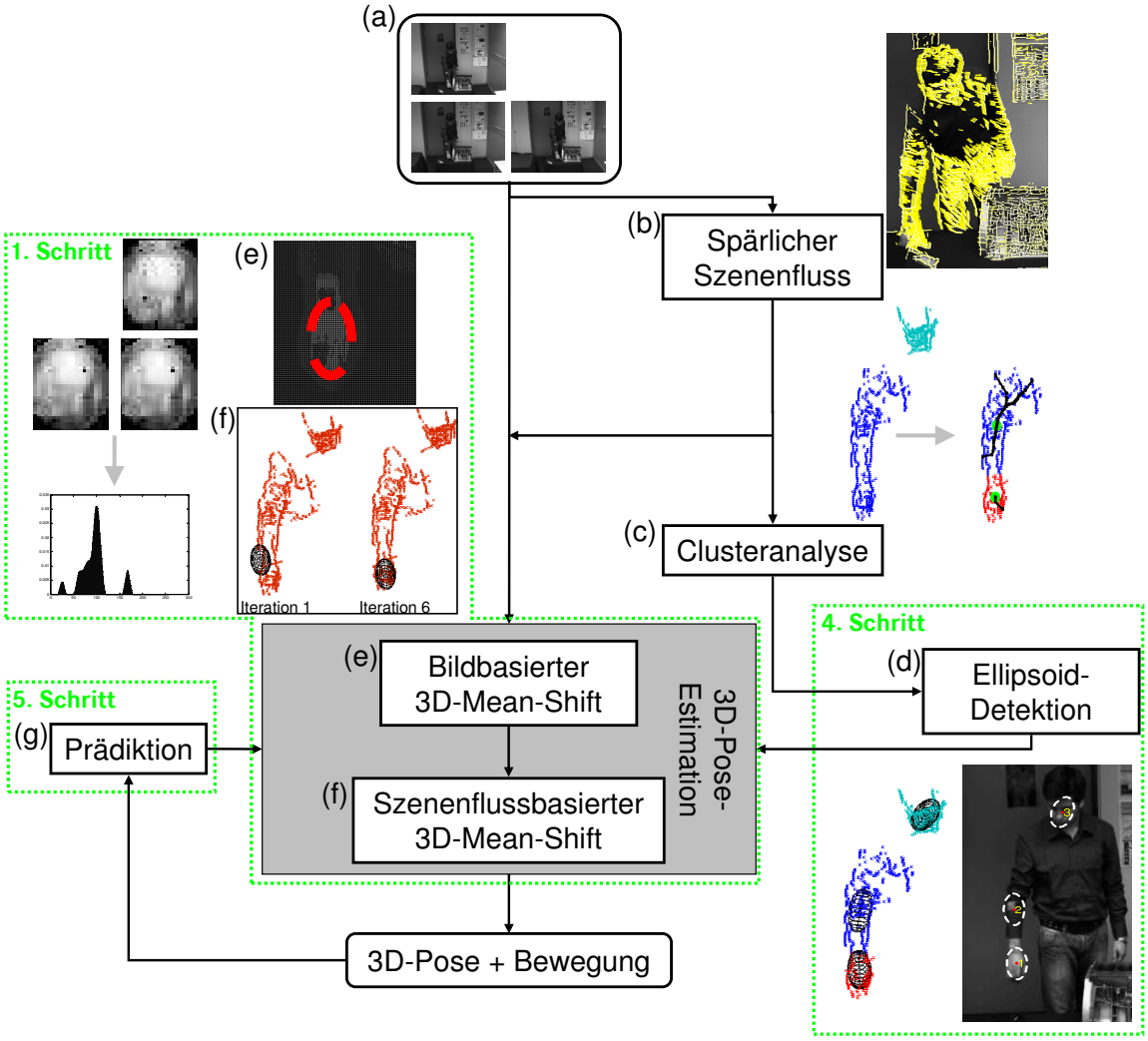

Abbildung 5.12: Struktur des Systemmodells zum 3D-Mean-Shift-Tracking. Es sind zudem typische Ausgaben einzelner Module dargestellt und die Schritte des Systemablauf sind farblich hervorgehoben.

Algorithmus (Modul (e)) arbeitet direkt auf Bilddaten und der szenenflussbasierte 3D-Mean-Shift-Algorithmus (Modul (f)) auf Bilddaten und bewegten 3D-Daten in der beobachteten Szene. Diese bewegten 3D-Daten werden im Szenenflussmodul (b) berechnet und ein spärlicher Szenenfluss erzeugt, welcher aus 3D-Punkten mit zugehörigen Geschwindigkeiten besteht. Die Szenenfluss-Daten sind, wie bereits beschrieben, Input für die Pose-Estimation und außerdem wichtig, um neue Objekte zu detektieren oder verlorene Objekte wiederzufinden. Dies geschieht über ein zweistufiges Clustering mit anschließender Ellipsoid-Detektion.

Der folgende Systemablauf beschreibt den Tracking-Prozess bereits detektierter Objekte:

1. Berechne mit den Modulen (e) und (f) für alle verfolgten Ellipsoide i unter Nutzung der prädizierten Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$ die neue Objektpose $\mathbf{\Phi}^{(i)}(t)$ für den Zeitschritt t.
2. Wenn die Bhattacharyya-Distanz $B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i)$ (Gleichung (5.15)) des Zielmodells $\widehat{\mathbf{q}}_i$ und des

Merkmalshistogramms $\widehat{\mathbf{p}}_i$ an der aktuellen Pose $\mathbf{\Phi}^{(i)}(t)$ eine Unähnlichkeit beschreibt, suche „bessere“ Ellipsoide aus den Ergebnissen der Ellipsoid-Detektion.

3. Wenn die Bhattacharyya-Distanz $B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i)$ über drei Zeitschritte unähnlich ist, lösche das Objekt.
4. Wenn durch die Ellipsoid-Detektion ein unbekanntes Objekt gefunden wird, ordne es in die Liste der zu verfolgenden Objekte ein.
5. Berechne im Prädiktionsmodul für alle Ellipsoide i die prädizierte 3D-Pose $\widehat{\mathbf{\Phi}}^{(i)}(t + \Delta t)$ für den Zeitschritt $t + \Delta t$.

Im ersten Schritt werden im Pose-Estimation-Modul alle Ellipsoide i, welche bereits verfolgt wurden, unter Nutzung der prädizierten Pose $\widehat{\mathbf{\Phi}}^{(i)}(t)$ an die Bild- und 3D-Szenenfluss-Daten angepasst und somit die aktuelle Objektposition $\mathbf{\Phi}^{(i)}(t)$ zum Zeitschritt t bestimmt.

Im zweiten Schritt wird über die Bhattacharyya-Distanz $B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i)$ des aktuellen Zielmodells $\widehat{\mathbf{q}}_i$ und des Merkmalshistogramms $\widehat{\mathbf{p}}_i$ an der aktuellen Pose $\mathbf{\Phi}^{(i)}(t)$ die Ähnlichkeit der beiden Modelle bestimmt. Der Bhattacharyya-Koeffizienten (Comaniciu u. a., 2000) beschreibt die Distanz zwischen zwei Zielmodellen $\widehat{\mathbf{q}}_i$ und $\widehat{\mathbf{p}}_i$ und wird berechnet mit:

$$B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i) = \sum_{n=1}^{N_{Bin}} \sqrt{\widehat{\mathbf{p}}_i(n)\widehat{\mathbf{q}}_i(n)}. \tag{5.15}$$

Geometrisch lässt sich der Bhattacharyya-Koeffizient als der Kosinus des Winkels zwischen den N_{Bin}-dimensionalen Vektoren $(\sqrt{\widehat{\mathbf{q}}_i(1)}, \ldots, \sqrt{\widehat{\mathbf{q}}_i(N_{Bin})})^T$ und $(\sqrt{\widehat{\mathbf{p}}_i(1)}, \ldots, \sqrt{\widehat{\mathbf{p}}_i(N_{Bin})})^T$ beschreiben. $B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i)$ kann maximal den Wert 1 annehmen. In diesem Fall wären $\widehat{\mathbf{q}}_i$ und $\widehat{\mathbf{p}}_i$ vollkommen identisch, und der Winkel zwischen den beiden Vektoren wäre also 0°. Bei $B = 0$ haben die beiden Merkmalshistogramme keine Übereinstimmung und der Winkel hat einen Wert von 90°.

Falls die beiden Merkmalshistogramme unähnlich sind, d.h. $B(\widehat{\mathbf{q}}_i, \widehat{\mathbf{p}}_i)$ ist niedrig, wird angenommen, dass das aktuelle Trackingergebnis schlecht ist und nur ungenügend das Zielobjekt überdeckt. Bei einem schlechten, aktuellen Trackingergebnis wird außerdem in der Menge der detektierten Ellipsoide (Abschnitt 5.4) nach einer besseren Pose gesucht. Dies bedeutet, dass die Merkmalshistogramme aller detektierten Ellipsoide in der Nähe der getrackten Pose $\mathbf{\Phi}^{(i)}(t)$ mit dem Zielmodell $\widehat{\mathbf{q}}_i$ verglichen werden und wenn eine Ähnlichkeit der Histogramme besteht, die getrackte Pose durch die detektierte Pose ersetzt wird.

Der dritte Schritt des Systemablaufs sorgt dafür, dass alle Objekte, die über drei Zeitschritte zum Zielmodell unähnlich sind, gelöscht werden.

Im vierten Schritt werden alle Objekte, die neu detektiert wurden, initialisiert. Dies bedeutet, es wird das Zielmodell $\widehat{\mathbf{q}}_i$ berechnet (Abschnitt 5.5.2) und alle neuen Objekte in die Liste der zu verfolgenden Objekte eingeordnet. Da das Szenenflussmodul bewegungsannotierte 3D-Daten liefert, wird außerdem für die Pose $\mathbf{\Phi}^{(i)}(t)$ aller verfolgten Ellipsoide i die aktuelle Bewegung bestimmt.

Unter Nutzung der aktuellen 3D-Pose $\mathbf{\Phi}^{(i)}(t)$ und der ermittelten Bewegung wird im fünften Schritt die prädizierte Objektposition $\widehat{\mathbf{\Phi}}^{(i)}(t + \Delta t)$ für den nächsten Zeitschritt $(t + \Delta t)$ vorhergesagt und diese dann wieder im ersten Schritt mit dem 3D-Pose-Estimation-Modul angepasst. Damit beginnt der Ablauf von vorn.

Kapitel 6

Objektverfolgung durch 3D-Pose-Estimation basierend auf raum-zeitlichen Konturen und 3D-Punktewolken

Das in diesem Kapitel vorgestellte Trackingsystem entsteht durch Kombination von in dieser Arbeit entwickelten Algorithmen mit den Verfahren aus der Dissertation von Barrois (2010). Die für das Tracking notwendige 3D-Pose-Estimation basiert auf Konturen in den Kamerabildern und 3D-Punktewolken. Die 3D-Punktewolken werden aus den Bildern eines Mehrkamerasystems mit kleiner Basisbreite berechnet. Durch eine Fusion verschiedener Ansätze zur 3D-Pose-Estimation von menschlichen Körperteilen soll die Genauigkeit und Robustheit des Trackings weiter erhöht werden. Die Fusion basiert auf drei verschiedenen Ähnlichkeitsmaßen, wodurch die Pose-Schätzungen der jeweiligen Algorithmen bewertet werden können und somit die Grundlage für die gewichtete Fusion darstellen. Die entwickelte Bewegungsanalyse zur Vorhersage der Bewegung basiert auf einer Bewegungsschätzung in Verschiebungsvektorfeldern und einer Bewegungsschätzung in Bildsequenzen durch den Shape-Flow-Algorithmus (Abschnitt 4.1.1). Durch die unterschiedlichen Algorithmen wird die Bewegungsvorhersage der verfolgten Körperteile verbessert und es kann sogar die Tiefengeschwindigkeit eines verfolgten Objekts instantan, also direkt aus den Bilddaten, geschätzt werden.

Das entwickelte System gewährleistet eine automatische Detektion und Verfolgung der menschlichen Hand-Unterarm-Region im 3D-Raum. Das Systemmodell ist in Abbildung 6.1 dargestellt. Es besteht aus zehn Komponenten: dem Kamerasystem (a), einem Szenenflussmodul (b), einem Modul zur Clusteranalyse (c), einer auf der Clusteranalyse basierten Modellinitialisierung (d), den Modulen (e), (f) und (g) zur 3D-Pose-Estimation, dem Modul zur Bewegungsanalyse basierend auf 3D-Punktewolken, einem optionalen Ansatz zur raum-zeitlichen 3D-Pose-Estimation (h) sowie einem Prädiktionsmodul (j).

Der Shape-Flow-Algorithmus (Modul (i)) kann in diesem Trackingsystem optional zur Bewegungsvorhersage verwendet kann. Daher gibt es für das in diesem Kapitel beschriebene System zwei Systemvarianten, welche später in den Experimenten (Abschnitt 7) auch gegeneinander evaluiert werden.

Das entwickelte System basiert zum Teil auf Algorithmen, welche in den Kapiteln 3 bis 5 bereits beschrieben wurden. Daher wird bei der Systembeschreibung auf die entsprechenden

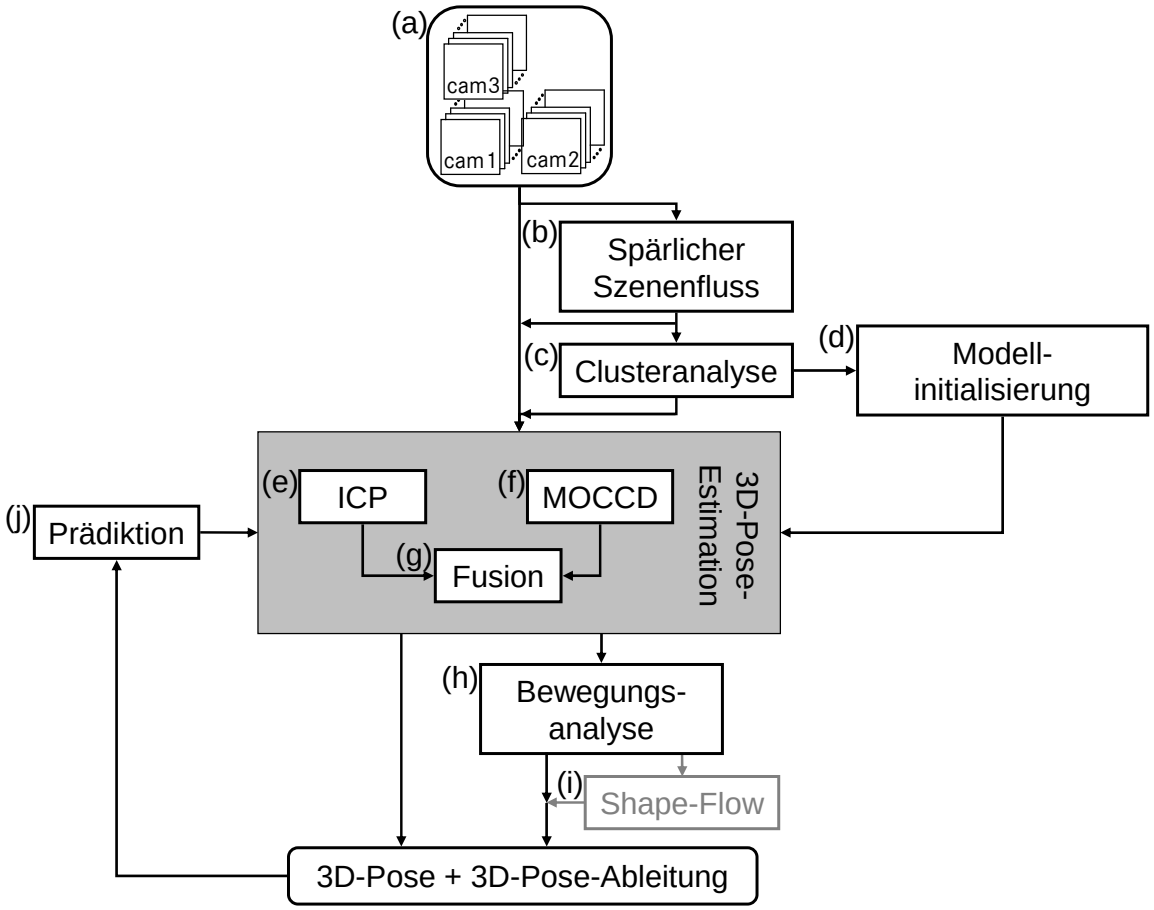

Abbildung 6.1: Objektverfolgung durch 3D-Pose-Estimation basierend auf Konturen und 3D-Punktewolken.

Abschnitte in den Kapiteln 3 bis 5 verwiesen.

Das verwendete Kamerasystem wird in Abschnitt 3.1 beschrieben und ist dasselbe wie bei den zuvor beschriebenen Trackingsystemen (Kapitel 3 bis 5). Das verwendete Objektmodell zur Beschreibung der menschlichen Hand-Unterarm-Region wurde bereits in Abschnitt 3.2.1 erläutert. Die Erzeugung der spärlichen Szenenflussdaten, einem nicht dichten 3D-Bewegungsfeld von Punkten in der Welt, wurde in Abschnitt 5.2 beschrieben. Im Folgenden werden die Systemkomponenten (c)-(j) in Abbildung 6.1 näher beschrieben und abschließend die Interaktion der Systemkomponenten im Gesamtsystem erläutert.

6.1 Clusteranalyse und Modellinitialisierung

Die Module zur Clusteranalyse und Modellinitialisierung basieren auf spärlichen Szenenflussdaten. Deren Berechnung wurde bereits im Abschnitt 5.2 beschrieben. Die Clusteranalyse wird auf den Datenvektoren $\mathbf{s}_i \in S(i = 1, \ldots, N)$ aus dem Szenenflussmodul angewandt. Das Ziel der Auswertung besteht darin, die gegebene Menge $S = \mathbf{s}_1, \ldots, \mathbf{s}_n$ in Cluster ein-

zuteilen. Es wird ein graphenbasiertes Clustering-Verfahren (Bock, 1974) angewandt. Dieses wurde bereits im Abschnitt 5.3 beschrieben. Ein Input-Vektor $\mathbf{s}_i$ aus den Input-Daten S hat die Form $\mathbf{s}_i = [{}^W s_X, {}^W s_Y, {}^W s_Z, {}^W \dot{s}_X, {}^W \dot{s}_Y]^T$, wobei ${}^W\mathbf{s}$ die 3D-Koordinaten des Szenenfluss-Punktes beschreiben und ${}^W\dot{\mathbf{s}} = ({}^W \dot{s}_X, {}^W \dot{s}_Y)$ für den Betrag der XY-Geschwindigkeiten steht. Wie im bereits beschriebenen System zur Objektverfolgung in Bilddaten und 3D-Punktewolken (Kapitel 5) gehen nur Punkte in das Clustering ein, die sich auch bewegen, also wenn die Länge $\sqrt{({}^W \dot{s}_X)^2 + ({}^W \dot{s}_Y)^2}$ des Geschwindigkeitsvektors eine Schwelle Θ_{Bewegung} überschreitet. Die Ausgabe der Clusteranalyse sind zusammengefasste Gruppen (Cluster) aller sich möglichst homogen bewegenden und auch in räumlicher Nähe zueinander befindlichen Vektoren aus den Szenenflussdaten der beobachteten Szene.

Das entwickelte System ist in der Lage, die menschlichen Hand-Unterarm-Region im beobachteten 3D-Raum zu detektieren. Diese Detektion stellt eine Initialisierung des verwendeten 3D-Hand-Unterarm-Modells (Abschnitt 3.2.1) dar. Da häufig nach dem graphenbasierten Clustering mehrere Cluster vorhanden sind, wird eine Heuristik zur Auswahl der zum menschlichen Hand-Unterarm-Bereich gehörigen Cluster verwendet. Dabei wird die Geschwindigkeit des Clusters und die Nähe zum definierten Interaktionsobjekt als Auswahlkriterium herangezogen. Hinter der Heuristik steckt die Annahme, dass zum Beginn des Arbeitsprozesses der menschliche Hand-Unterarm-Bereich sich in der Nähe des Interaktionsobjekts am Schnellsten bewegt.

Nach der Auswahl der Cluster werden die Positions- und Orientierungsparameter des Hand-Unterarm-Modells aus dem Mittelpunkt und den Hauptkomponenten der gewählten Cluster abgeleitet. Die Modellinitialisierung wurde von Barrois (2010) entwickelt und ist detailliert in dessen Dissertation beschrieben. Nach der Initialisierung startet das Tracking und die Modellparameter werden durch die Pose-Estimation-Verfahren angepasst und im Prädiktionsmodul (Abschnitt 6.4) durch die vorherige Pose-Schätzung und die in der Bewegungsanalyse (Abschnitt 6.3) ermittelten Bewegungsparameter prädiziert. Im Folgenden werden die verwendeten Verfahren zur 3D-Pose-Estimation des Hand-Unterarm-Modells vorgestellt.

6.2 3D-Pose-Estimation

Im entwickelten Trackingsystem dieses Kapitels werden zwei unterschiedliche Ansätze zur 3D-Pose-Estimation miteinander gewichtet kombiniert, um die Vorteile beider Verfahren nutzen zu können. Dabei wird zum einen ein Verfahren verwendet, welches basierend auf bewegten und geclusterten Szenenflussdaten des beobachteten Bereichs mit Hilfe des Iterative-Closest-Point (ICP) Algorithmus (Besl u. McKay, 1992) das 3D-Hand-Unterarm-Modell anpasst (Abbildung 6.1 Modul (e)). Zum anderen wird der bereits beschriebene MOCCD-Algorithmus eingesetzt, um das Objektmodell an die Bilddaten anzupassen. Ein Vorteil der verwendeten Ansätze ist, dass sie auf unterschiedlichen Eingabedaten basieren und somit auch einen Ausfall des jeweils anderen Verfahrens kompensieren können. Durch die gewichtete Kombination kann die Genauigkeit und Robustheit des Trackings weiter erhöht werden. Die Fusion der zwei Pose-Schätzungen basiert auf drei verschiedenen Ähnlichkeitsmaßen. Im Folgenden werden die Module (e) und (f) in Abbildung 6.1 näher beschrieben und anschließend die gewichtete Fusion der 3D-Pose-Schätzungen erläutert.

6.2.1 Iterative-Closest-Point (ICP) Algorithmus

In diesem Abschnitt wird das Erste, der in diesem Kapitel zur Pose-Estimation verwendeten Verfahren, vorgestellt. Der implementierte Algorithmus basiert auf dem Iterative-Closest-Point (ICP) Algorithmus (Besl u. McKay, 1992) und wurde zur Pose-Estimation verschiedener Objekte in geclusterten Szenenflussdaten von Barrois (2010) entwickelt und ist dort detailliert beschrieben. In dieser Arbeit wird der von Barrois (2010) entwickelte Ansatz zur Pose-Estimation der menschlichen Hand-Unterarm-Region mit dem in Abschnitt 3.2.1 beschrieben 3D-Hand-Unterarm-Modell verwendet und ist daher nur kurz beschrieben. Für weiterführende Details sei auf die Arbeit von Barrois (2010) verwiesen.

Das verwendete Verfahren ist ein bottom-up Ansatz zur Pose-Estimation, d.h. es müssen zuvor 3D-Punktewolken und Flussvektoren aus kleinen Bildausschnitten berechnet werden. Diese werden basierend auf ihrer Bewegung und der Distanz der 3D-Punkte zueinander durch eine Clusteranalyse gruppiert und erst dann wird das verwendete Hand-Unterarm-Modell an die Cluster mit dem ICP-Algorithmus angepasst.

Der ICP-Algorithmus (Besl u. McKay, 1992) ist ein Standardverfahren, um ein geometrisches Modell an eine 3D-Punktewolke anzupassen. Dabei werden die 3D-Pose-Parameter des Modells iterativ angepasst, indem die summierte quadratische Distanz zwischen den 3D-Punkten und dem 3D-Modell minimiert wird. Das Ergebnis des Verfahrens ist eine 3D-Pose, d.h. die 3D-Position und die Orientierung des Modells. Die von Zhang (1992) vorgestellte Version des ICP-Algorithmus ist in der Lage, Ausreißer in der 3D-Punktewolke auszuschließen. Dadurch wird eine robustere Pose-Schätzung erzielt.

Das von Barrois (2010) entwickelte Verfahren verwendet wie Zhang (1992) einen Ansatz, bei dem Ausreißer ausgeschlossen werden. Neben der Distanz zum Modell werden auch die Bewegungsattribute der 3D-Punkte aus dem Szenenfluss Modul verwendet, um Ausreißer vor Beginn der eigentlichen Optimierung mit dem ICP-Algorithmus auszuschließen. Es werden alle 3D-Punkte zur Menge M_p zugeordnet für die die Bedingung in Gleichung (6.1) erfüllt ist. Dadurch erfolgt eine Segmentierung der Punktewolke aus Szenenflusspunkten.

$$M_p = \left\{ {}^W\mathbf{s} \mid \mathrm{dist}({}^W\mathbf{s}, {}^W\mathbf{p}_m) < \theta_D \quad \cap \quad \| {}^W\dot{\mathbf{s}} - {}^W\dot{\mathbf{s}}_{\mathrm{mean}} \| < \theta_v \right\}. \tag{6.1}$$

In Gleichung (6.1) beschreibt $\mathrm{dist}({}^W\mathbf{s}, {}^W\mathbf{p}_m)$ die euklidische Distanz zwischen einem 3D-Szenenflusspunkt ${}^W\mathbf{s}$ und dem korrespondierenden Punkt ${}^W\mathbf{p}_m$ auf der Oberfläche des Modells. Die Variable ${}^W\dot{\mathbf{s}}_{\mathrm{mean}}$ steht für den mittleren Geschwindigkeitsvektor der 3D-Punkte in der überwachten Szene. Die Distanzschwelle θ_D wird dynamisch während der Optimierung gesetzt und ist von der Verteilung der Distanzen zwischen dem Modell und den Szenenflusspunkten abhängig. Die Geschwindigkeitsschwelle θ_v wird analog festgelegt, für eine detailliertere Beschreibung sei auf Barrois (2010) verwiesen. Die Optimierung der Zielfunktion ist als M-Estimator (Rey, 1983) implementiert. Im Gegensatz zur Arbeit von Zhang (1992) ist das von Barrois (2010) entwickelte Verfahren in der Lage, 3D-Punkte hinzuzunehmen und auszuschließen. Dies ist für die gesamte Pose-Estimation entscheidend, denn durch die Fusion mit den Pose-Schätzungen des MOCCD-Algorithmus gibt es mehrere ICP-Initialisierungen und es ist wichtig, bisher nicht betrachtete 3D-Punkte durch die Neuinitialisierung hinzuzunehmen. Ein Problem des entwickelten ICP-Algorithmus ist, dass die Zielfunktion für die Parameter der Verschiebung des Modells entlang der Unterarm-Achse sehr flach ist, was dazu führt,

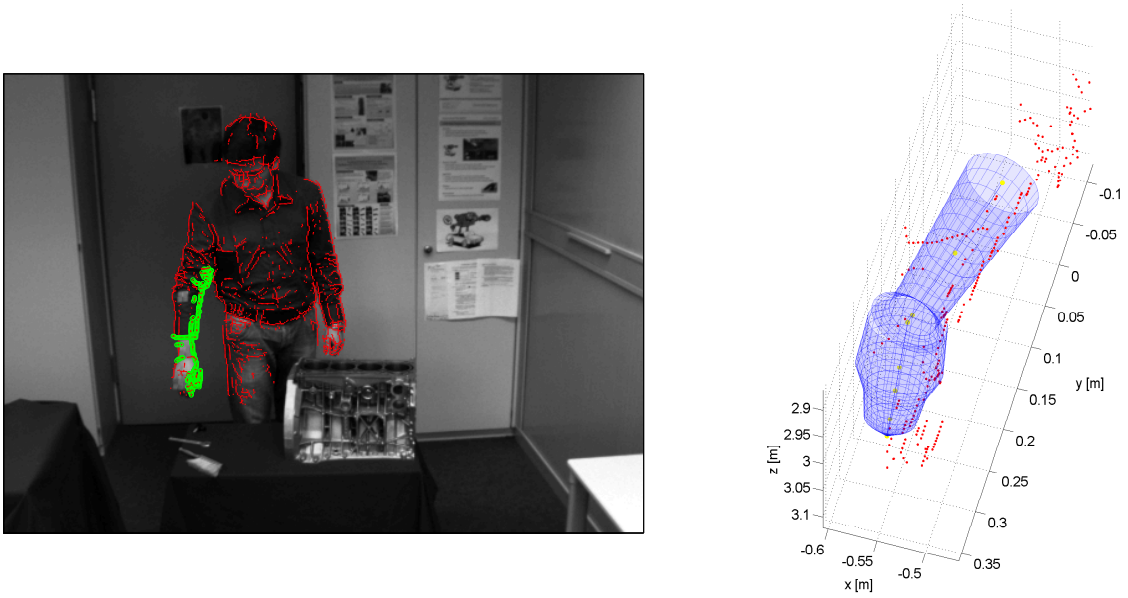

Abbildung 6.2: Links: Alle bewegten Szenenflusspunkte (rot) der beobachteten Szene. Durch Gleichung (6.1) werden alle Punkte der Hand-Unterarm-Region (grün) durch eine Segmentierung der Punktewolke aus Szenenflusspunkten ermittelt. Die segmentierten Szenenflusspunkte werden zur Optimierung mit dem ICP-Algorithmus verwendet. Rechts: Mit dem von Barrois (2010) entwickelten Verfahren angepasstes 3D-Hand-Unterarm-Modell (blau) an die bewegten Szenenflusspunkte (rot).

dass diese Verschiebung schwieriger zu schätzen ist. Die Ursache liegt im länglichen Modell zur Beschreibung des menschlichen Hand-Unterarm-Bereichs. Durch die Verwendung einer erweiterten Zielfunktion im entwickelten ICP-Algorithmus wird die Verschiebung des Modells entlang der Unterarm-Achse behandelt. Die verwendete Zielfunktion ergibt sich mit

$$e_{\text{ICP}} = \|{}^{W}\mathbf{s} - {}^{W}\mathbf{p}_m\| + \lambda_{\text{hs}}\|{}^{W}\mathbf{s} - {}^{W}\mathbf{p}_8\| \tag{6.2}$$

und entsteht durch Kombination der originalen ICP-Zielfunktion und einem zusätzlichen Fehlerterm $\|{}^{W}\mathbf{s} - {}^{W}\mathbf{p}_8\|$, welcher die Verschiebung des Modells entlang der Unterarm-Achse behandelt. Der originale Fehlerterm $\|{}^{W}\mathbf{s} - {}^{W}\mathbf{p}_m\|$ beschreibt die summierte quadratische Distanz zwischen 3D-Szenenflusspunkten ${}^{W}\mathbf{s}$ und korrespondierenden 3D-Punkten ${}^{W}\mathbf{p}_m$ auf der Oberfläche des Modells. Der zusätzliche Fehlerterm $\|{}^{W}\mathbf{s} - {}^{W}\mathbf{p}_8\|$ repräsentiert die summierte quadratische Distanz zwischen den 3D-Szenenflusspunkten ${}^{W}\mathbf{s}$ und der Spitze des Hand-Unterarm-Modells ${}^{W}\mathbf{p}_8$ (Abschnitt 3.2.1). Durch das Gewicht λ_{hs} wird der neue Fehlerterm mit der originalen ICP-Zielfunktion kombiniert.

Die Deformation des Hand-Unterarm-Modells, also die Radien der Kegelstümpfe, wird durch den Ansatz von Barrois (2010) nicht geschätzt, sondern wird als konstant angenommen. Abbildung 6.2 (links) zeigt die zur Pose-Estimation verwendeten Szenenflusspunkte im Bereich der menschlichen Hand-Unterarm-Region. Abbildung 6.2 (rechts) zeigt das Ergebnis der Pose-Estimation mit dem entwickelten Ansatz.

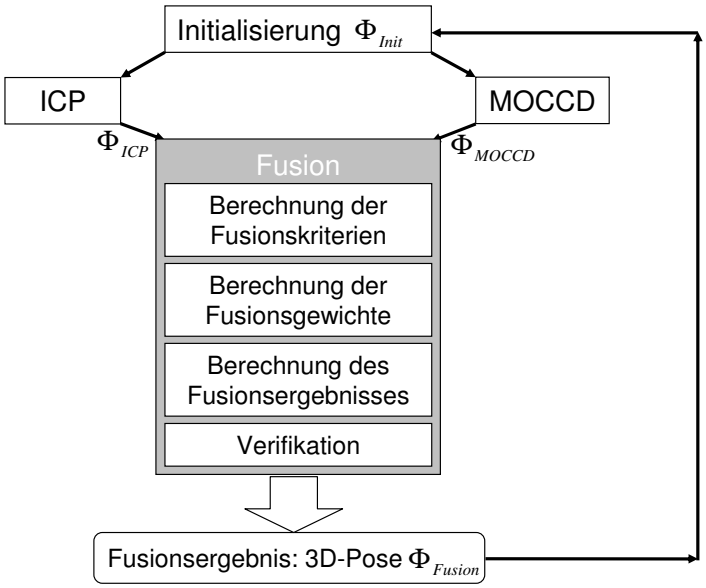

Abbildung 6.3: Ablaufplan des Fusionsmoduls zur gewichteten Fusion der Pose-Schätzungen von ICP-Algorithmus und MOCCD-Algorithmus.

6.2.2 Multiokularer Contracting-Curve-Density (MOCCD) Algorithmus

Der multiokulare Contracting-Curve-Density (MOCCD) Algorithmus ist ein top-down Verfahren zur 3D-Pose-Estimation in mindestens zwei 2D-Bildern. Grundlage der Pose-Estimation ist ein System aus kalibrierten Kameras in einem definierten Weltkoordinatensystem W. Das Modell einer parametrischen Kurve wird dabei an die Bilddaten so angepasst, dass die Pixelstatistiken auf der inneren und äußeren Seite der parametrischen Kurve möglichst unterschiedlich sind. Der MOCCD-Algorithmus wurde bereits in Kapitel 3 zur 3D-Konturanpassung mit einem analytischen Hand-Unterarm-Modell verwendet und ist im Abschnitt 3.3.1 detailliert beschrieben. Zur Pose-Estimation werden ausschließlich Bilddaten verwendet. Im Gegensatz zum ICP-Ansatz von Barrois (2010) ist durch den MOCCD-Algorithmus eine Anpassung des Hand-Unterarm-Modells entlang der Unterarmachse gut möglich. Daher scheint eine Kombination sehr vorteilhaft.

6.2.3 Fusion der Pose-Estimation-Algorithmen

Da im entwickelten Trackingsystem, welches in diesem Kapitel beschrieben wird, zwei unterschiedliche Ansätze zur 3D-Pose-Estimation verwendet werden, müssen die Ergebnisse beider Verfahren kombiniert werden. Eine gewichtete Kombination basierend auf drei unterschiedlichen Kriterien ermöglicht es, die Vorteile beider Verfahren nutzen zu können und ihre Nachteile zu vermeiden. Außerdem kann der Ausfall eines der Verfahrens erkannt und kompensiert werden. Der Programmablauf des Fusionsmoduls ist in Abbildung 6.3 dargestellt. Als Erstes werden die initialen Pose-Parameter $\mathbf{\Phi}_{\text{init}}$ für beide Pose-Estimation-Verfahren gesetzt und parallel die Pose-Estimation des Hand-Unterarm-Modells durchgeführt. Zur Verifikation und

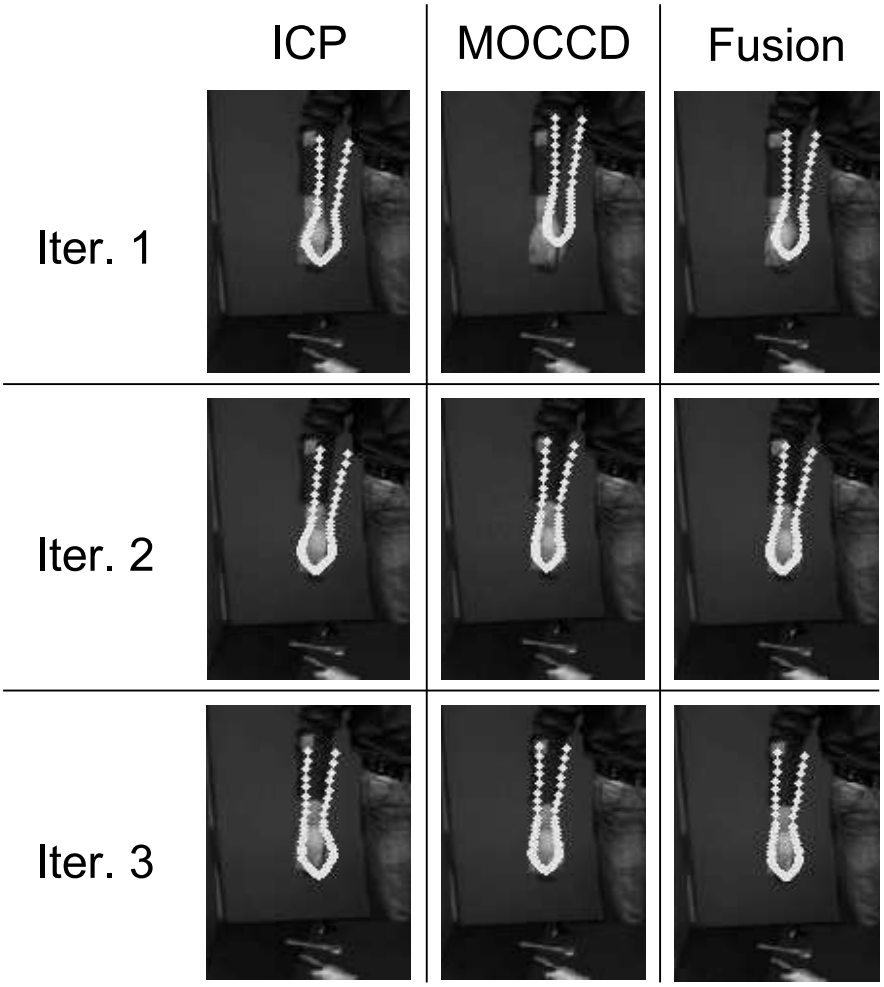

Abbildung 6.4: Konvergenzverhalten des Fusionsmoduls zur gewichteten Fusion der Ergebnisse von ICP- und MOCCD-Algorithmus. Die erste Spalte zeigt die Ergebnisse der Pose-Schätzung des ICP-Algorithmus und die zweite Spalte stellt die Ergebnisse der Pose-Schätzung für den MOCCD-Algorithmus dar. In der letzten Spalte sind die Ergebnisse des Fusionsmoduls zu sehen.

Gewichtung der einzelnen Pose-Schätzungen werden drei Kriterien als Qualitätsmaße berechnet: (i) die Distanz der ausgewählten Szenenflusspunkte zum Modell, (ii) die Ähnlichkeit der Modellorientierung und (iii) die Ähnlichkeit der Erscheinung.

Kriterium 1 – Punktdistanz: Das erste Kriterium zur Bestimmung der Qualität der beiden Pose-Schätzungen berechnet sich aus der Distanz der segmentierten Punktewolke aus Szenenflusspunkten zum Modell, beschrieben durch den geschätzten Parametervektor. Zur Berechnung der Punktdistanz wird die Hülle des Modells verwendet und bestimmt wie viele Punkte innerhalb und außerhalb der Modellhülle liegen. Der Gewichtsparameter $\mathcal{PD}(\mathbf{\Phi}(t))$ des ersten Kriteriums wird für einen Parametervektor $\mathbf{\Phi}(t)$ zum Zeitschritt t berechnet, indem der Quotient aus der Anzahl der Szenenflusspunkte innerhalb des Modells und der Anzahl aller segmentierter Szenenflusspunkte ermittelt wird.

Kriterium 2 – Ähnlichkeit der Modellorientierung: Zur Berechnung des Qualitätsmaßes $\mathcal{MO}(\mathbf{\Phi}(t))$ für eine Pose-Schätzung $\mathbf{\Phi}(t)$ wird die Ähnlichkeit der Modellorientierung mit der Kantenorientierung in den Kamerabildern zum Zeitschritt t bestimmt. Dazu wird das verwendete 3D-Konturmodell (Abschnitt 3.2) für jede Kamera c in deren zugehöriges Pixelkoordinatensystem P_c projiziert. Das Qualitätsmaß des zweiten Kriteriums wurde bereits im Kapitel 3 verwendet und ist dort im Abschnitt 3.4 beschrieben.

Kriterium 3 – Ähnlichkeit der Erscheinung: Das dritte Kriterium ergibt sich aus der Ähnlichkeit der Erscheinung (Aussehen) $\mathcal{ER}(\mathbf{\Phi}(t), \mathbf{\Phi}(t-n))$ für eine 3D-Pose-Schätzung $\mathbf{\Phi}(t)$ zum Zeitschritt t. Durch den Vergleich der aktuellen Erscheinung (Zeitschritt t) mit der vorherigen Objekterscheinung zum Zeitschritt $(t-n)$ wird das Qualitätsmaß berechnet. Auch das dritte Kriterium wurde bereits im Trackingsystem in Kapitel 3 verwendet und ist dort im Abschnitt 3.4 detailliert beschrieben. Die Erzeugung der Datenbank der vorherigen Objekterscheinung ist in Abbildung 3.14 dargestellt.

Fusion: Das Fusionsergebnis wird berechnet, indem die beiden Pose-Schätzungen $\mathbf{\Phi}_{\text{ICP}}$ und $\mathbf{\Phi}_{\text{MOCCD}}$ durch die berechneten Kriterien gewichtet kombiniert werden. Die Kriterien werden für beide Pose-Schätzungen berechnet und anschließend werden die Gewichte w_{ICP} für die Pose-Schätzung des ICP-Algorithmus und w_{MOCCD} für die Schätzung des MOCCD-Algorithmus berechnet

$$w_{\text{ICP}} = \mathcal{PD}(\mathbf{\Phi}_{\text{ICP}}) + \mathcal{MO}(\mathbf{\Phi}_{\text{ICP}}) + \mathcal{ER}(\mathbf{\Phi}_{\text{ICP}}, \mathbf{\Phi}_{\text{Fusion}}(t-1)) \tag{6.3}$$

$$w_{\text{MOCCD}} = \mathcal{PD}(\mathbf{\Phi}_{\text{MOCCD}}) + \mathcal{MO}(\mathbf{\Phi}_{\text{MOCCD}}) + \mathcal{ER}(\mathbf{\Phi}_{\text{MOCCD}}, \mathbf{\Phi}_{\text{Fusion}}(t-1)). \tag{6.4}$$

Die beiden Pose-Updates $\Delta\mathbf{\Phi}_{\text{ICP}}$ und $\Delta\mathbf{\Phi}_{\text{MOCCD}}$ werden nun unter Nutzung der berechneten Werte w_{ICP} und w_{MOCCD} gewichtet zum Parametervektor $\mathbf{\Phi}_{\text{Fusion}}$ fusioniert. Dieser ergibt sich mit

$$\mathbf{\Phi}_{\text{Fusion}} = \mathbf{\Phi}_{\text{init}} + \frac{w_{\text{ICP}}}{w_{\text{ICP}} + w_{\text{MOCCD}}} \cdot \Delta\mathbf{\Phi}_{\text{ICP}} + \frac{w_{\text{MOCCD}}}{w_{\text{ICP}} + w_{\text{MOCCD}}} \cdot \Delta\mathbf{\Phi}_{\text{MOCCD}}. \tag{6.5}$$

Wie im Programmablauf des Fusionsmoduls in Abbildung 6.3 zu sehen ist, wird noch eine Verifikation der fusionierten Pose durch das in Abschnitt 3.4 beschriebene Verifikationsmodul durchgeführt. Die Prozedur der Fusion wird, wie in Abbildung 6.3 zu sehen, mehrfach wiederholt, dabei wird bei jeder weiteren Fusion das Fusionsergebnis als Initialisierung der nächsten Pose-Estimation verwendet. Die Fusion wird durchgeführt bis es kaum noch Änderungen am Fusionsergebnis gibt oder eine spezifizierte Anzahl Iterationsschritte erreicht ist. Wenn die beste ermittelte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}$ verifiziert werden konnte, werden die zur 3D-Pose-Schätzung zugehörigen Referenz-Templates in die Datenbank eingeordnet (Abschnitt 3.4). Abbildung 6.4 zeigt das Konvergenzverhalten des Fusionsmoduls. Es ist gut zu erkennen, wie die initial schlechte Schätzung mit jedem Iterationsschritt der Fusion verbessert wird.

6.3 Bewegungsanalyse

Das entwickelte System zur Bewegungsanalyse wird verwendet, um eine Vorhersage der Bewegung des in den Kamerabildern verfolgten Objekts machen zu können. Die Bewegungs-

analyse basiert zunächst auf einer Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern und einem optionalen Ansatz zur raum-zeitlichen 3D-Pose-Estimation in Bildsequenzen durch den Shape-Flow-Algorithmus (Abschnitt 4.1.1). Die Bewegungsschätzung in Verschiebungsvektorfeldern ist ein bottom-up Verfahren, da es auf zuvor berechneten Bewegungsvektoren basiert und aus diesen eine Bewegungsvorhersage für das Hand-Unterarm-Modell ableitet. Der zweite Ansatz ist ein top-down Verfahren und schätzt die Bewegung der menschlichen Hand-Unterarm-Region in Bildsequenzen durch die Verwendung eines raum-zeitlichen Hand-Unterarm-Modells (Abschnitt 4.2.1). Durch die unterschiedlichen Algorithmen mit unterschiedlichen Vor-und Nachteilen wird die Bewegungsvorhersage der verfolgten Körperteile noch robuster und genauer.

6.3.1 Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern

Der Ansatz zur bottom-up Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern wurde von Barrois (2010) entwickelt und wird in dieser Arbeit verwendet, um eine Bewegungsvorhersage für das Hand-Unterarm-Modell aus spärlichen Szenenflussdaten zu berechnen. Der Ansatz wird nur kurz an einem Beispiel beschrieben. Für detailliertere Informationen sei auf die Arbeit von Barrois (2010) verwiesen. Das entwickelte Verfahren ist unabhängig vom verwendeten optischen Flussverfahren in der Lage, eine robuste und genaue Schätzung für die Objektbewegung zu ermitteln.

Hauptaugenmerk das Ansatzes von Barrois (2010) liegt dabei auf den Mehrdeutigkeiten, die durch das Aperturproblem verursacht werden. Ziel ist es, ein Verfahren zur Bestimmung der Objektbewegung zu entwickeln, dass unabhängig vom optischen Flussverfahren arbeitet und unter Berücksichtigung des Aperturproblems die vollständige Objektbewegung ermittelt.

In der Literatur sind einige Verfahren zur Bestimmung der Objektbewegung bekannt, diese basieren meist auf zweidimensionalen Flussvektoren. Für die Verwendung von eindimensionalen Flussvektoren, wie z.B. vom Spacetime-Stereo (Schmidt u. a., 2007) geliefert, sind diese Ansätze jedoch nicht geeignet. Mithilfe des Constraint-Line-Ansatzes (Schunck, 1986) lässt sich jedoch auch für eindimensionalen Flussvektoren die Objektbewegung ermitteln. Nachteilig an diesem Verfahren ist jedoch, dass mit dem ursprünglichen Verfahren lediglich translatorische Bewegungen geschätzt werden können. Beide Bewegungskomponenten eines 3D-Punkts parallel zur Bildebene können nur ermittelt werden, wenn die Umgebung eine Ecke im Bild darstellt. An Kanten hingegen kann nur eine Geschwindigkeitskomponente sicher ermittelt werden, wie beispielsweise beim Spacetime-Stereo (Schmidt u. a., 2007).

Grundprinzip: Aus der berechneten Geschwindigkeitskomponente (optischer Fluss) und der Kantenorientierung im Bild lässt sich ein sogenannter Normalfluss ϕ_n berechnen. Der Winkel γ ergibt sich aus der Richtung der horizontalen Epipolarlinie und der Richtung des Normalflusses. Die Constraint-Line ergibt sich aus dem Normalfluss ϕ_n, welcher im $\dot{x}\dot{y}$-Raum mit der Einheit Meter pro Zeitschritt definiert ist, und dem aus der Kantenrichtung abgeleiteten Winkel α. Abbildung 6.5 zeigt die Definition einer Constraint-Line im $\dot{x}\dot{y}$-Raum.

Bei gegebenem Normalfluss ϕ_n werden alle möglichen Bewegungen durch die korrespondierende Constraint-Line beschrieben (Abbildung 6.5). Für ein Objekt, welches sich rein transla-

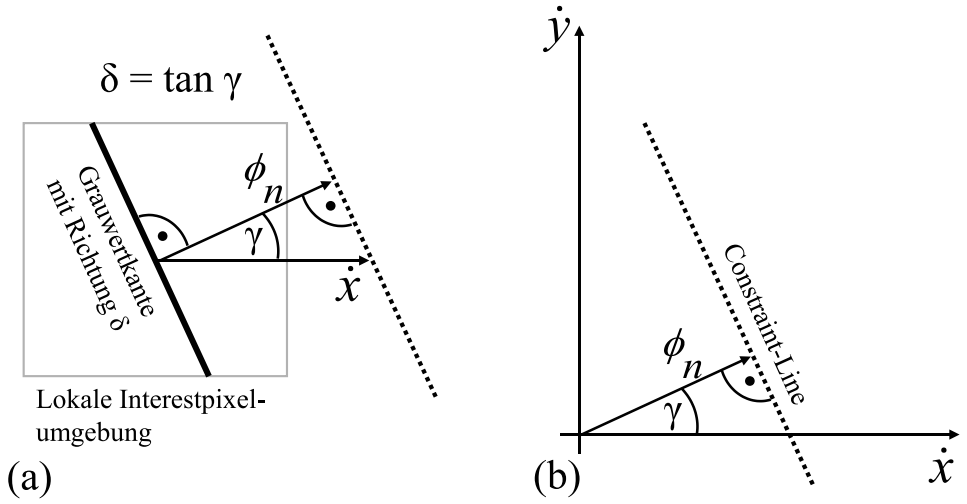

Abbildung 6.5: Definition einer Constrain-Line im $\dot{x}\dot{y}$-Raum. Die Constraint-Line ergibt sich aus dem Normalfluss ϕ_n, einem Geschwindigkeitsvektor mit der Einheit Meter pro Zeitschritt im $\dot{x}\dot{y}$-Raum, und dem aus der Kantenrichtung abgeleiteten Winkel γ zwischen der Richtung der horizontalen Epipolarlinie und der Richtung des Normalflusses. Quelle: Barrois (2010).

torisch bewegt, schneiden sich alle Constraint-Lines von Punkten, die zu dem Objekt gehören in einem einzigen Punkt. Somit ist die Translation eindeutig bestimmt. Für rotatorische Bewegungen bzw. für kombinierte rotatorische und translatorische Bewegungen, entsteht eine ausgedehnte Menge von Schnittpunkten der Constraint-Lines im $\dot{x}\dot{y}$-Raum. Dieser Fall wird jedoch von Schunck (1986) nicht weiter behandelt. Ein Beispiel für eine gegen den Uhrzeigersinn rotierende Ellipse wird in Abbildung. 6.6 gezeigt. Die Koordinaten der Schnittpunkte der Constraint-Lines sind in diesem Beispiel ein Maß für die mittlere horizontale Geschwindigkeit des korrespondierenden Paars von Bildpunkten. Die Schnittpunkteverteilung ist vertikal ausgedehnt, da ein vertikaler Kantendetektor verwendet wurde. Abbildung. 6.6 (c) zeigt die Verteilung der Schnittpunkte aus einer Hand-Unterarm-Testsequenz. Diese Verteilung ist typisch für ein rotationssymmetrisches, längliches Objekt. In dem Beispiel in Abbildung 6.6 (c) bewegen sich Punkte am Handgelenk schneller im Bild als die Punkte am Ellbogen. Die resultierenden Schnittpunkte konzentrieren sich stark an den Punkten $(\dot{x}_1, \dot{y}_1)$ und $(\dot{x}_2, \dot{y}_2)$, welche die Bewegung des Handgelenks und des Ellbogens repräsentieren.

Die Information über den rotatorischen Bewegungsanteil ergibt sich durch die Projektion der Schnittpunkte auf die Hauptkomponente senkrecht der longitudinalen Achse des Objekts (die zweite Hauptkomponente in Abbildung 6.6 (c)). Der rotatorischen Bewegungsanteil korrespondiert dann mit der Geschwindigkeitsdispersion über das Objekt, welche durch die rotatorische Bewegung in der Bildebene verursacht wird. Im vorgestellten System wird dies robust über das 10% und 90% Quantil der Verteilung der projizierten Schnittpunkte ermittelt. Für detailliertere Informationen sei auf die Arbeit von Barrois (2010) verwiesen.

In den Daten aus dem Modul zur Berechnung des spärlichen Szenenflusses sind keine Informationen über die Tiefengeschwindigkeit von 3D-Punkten verfügbar, daher kann die Tiefengeschwindigkeit des getrackten Objekts nicht instantan durch die von Barrois (2010) entwickelte Bewegungsanalyse bestimmt werden, obwohl das Verfahren dazu in der Lage wäre.

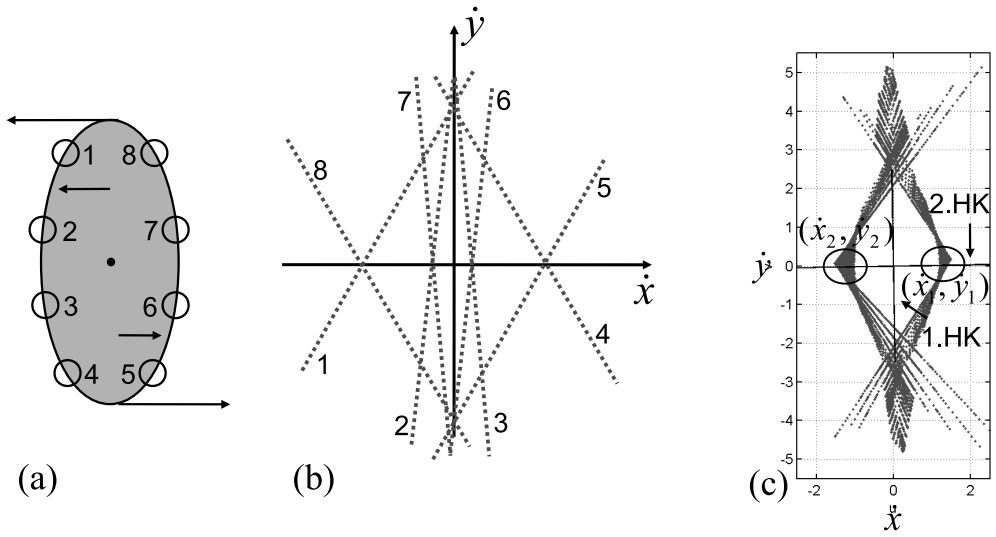

Abbildung 6.6: (a) Senkrecht zur Blickrichtung rotierende Ellipse mit Referenzpunkten auf der Kontur. (b) Resultierende Constraint-Lines der Rotation der Ellipse. Die Berechnung einer Constraint-Line ist in Abbildung 6.5 beschrieben. (c) Typische Verteilung der Schnittpunkte für eine reale Testsequenz. Der Mittelwert der Verteilung wurde bereits von allen Punkten subtrahiert, die erste (1.HK) und zweite (2.HK) Hauptkomponente der Verteilung sind als schwarze Linien gekennzeichnet. Quelle: Barrois (2010).

Die in diesem Abschnitt beschriebene Methode zur Bewegungsschätzung liefert bis auf die Tiefengeschwindigkeit direkt die vollständige zeitliche Ableitung $\dot{\mathbf{\Phi}}$ der Objektpose $\mathbf{\Phi}$, ohne dazu eine zeitliche Filterung benutzen zu müssen. Für ein Sicherheitssystem ist die Vermeidung zeitlicher Verzögerungen, welche z.B. durch Einschwingphasen eines zeitlichen Filters auftreten können, besonders wichtig.

6.3.2 Shape-Flow (SF) Algorithmus

Der Shape-Flow-Algorithmus ist ein top-down Ansatz zur raum-zeitlichen Pose-Estimation und basiert auf einer Erweiterung des MOCCD-Algorithmus (Abschnitt 3.3.1). Eine raum-zeitliche Modellkontur wird an eine Folge von Bildern angepasst. Dies geschieht, indem die Pixelstatistiken entlang von Liniensegmenten in der Bildfolge auf der inneren und äußeren Seite der raum-zeitlichen Kurve getrennt werden. Mit dem Ansatz können die 3D-Pose-Parameter $\mathbf{\Phi}$ und deren zeitliche Ableitung mit einem raum-zeitlichen Konturmodell und den Bildern von N_t Zeitschritten instantan geschätzt werden. Durch die top-down Verfahrensweise muss nur ein Konturmodell, welches die Kontur des in den Bildfolgen getrackten Objekts beschreibt, und eine grobe Initialisierung der 3D-Pose-Parameter $\mathbf{\Phi}$ vorhanden sein. Mit dem Shape-Flow-Algorithmus ist man in der Lage, einen dichten modellbasierten 3D-Szenenfluss zu berechnen und somit auch einen Tiefengeschwindigkeit eines Objekts zu schätzen.

Der Shape-Flow-Algorithmus wurde bereits in Kapitel 4 zur Verfolgung von raum-zeitlichen 3D-Konturen eingesetzt und dort im Abschnitt 4.1.1 detailliert beschrieben. Zur Bewegungsschätzung ist ein raum-zeitliches 3D-Hand-Unterarm-Modell notwendig, auch dieses wurde bereits im Abschnitt 4.2.1 vorgestellt.

Auch in diesem Kapitel wird ein hierarchischer Ansatz gewählt, d.h. die Berechnung der raum-zeitlichen Pose-Parameter wird aufgeteilt. Die 3D-Pose-Estimation erfolgt durch die gewichtete Fusion der Schätzungen des ICP- und MOCCD-Algorithmus (Abschnitt 6.2) und hat als Ergebnis den Parametervektor $\mathbf{\Phi}_{\text{Fusion}}(t)$. Mit dem Shape-Flow-Algorithmus wird die zeitliche Ableitung $\dot{\mathbf{\Phi}}(t)$ der 3D-Pose $\mathbf{\Phi}_{\text{Fusion}(t)}$ mit den Bildtripeln der Zeitschritte $(t - \Delta t)$ und $(t + \Delta t)$ bestimmt (Abschnitt 4.1.1). Die notwendige Initialisierung des Shape-Flow-Algorithmus stammt aus der zuvor berechneten Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern. Durch die Verwendung des Shape-Flow-Algorithmus kann auch eine instantane Tiefengeschwindigkeit des Objekts geschätzt werden, was durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern nicht möglich ist, da die Eingabedaten aus dem spärlichen Szenenflussmodul keine Tiefengeschwindigkeiten enthalten.

Der Shape-Flow-Algorithmus kann in diesem Trackingsystem optional zur Bewegungsvorhersage verwendet werden. Daher ergeben sich für das in diesem Kapitel beschriebene System zwei Systemvarianten, welche später in den Experimenten (Kapitel 7) auch evaluiert und verglichen werden. Falls der Shape-Flow-Algorithmus verwendet wird, ergibt sich die zur Prädiktion notwendige 3D-Pose-Ableitung aus der Ausgabe $\dot{\mathbf{\Phi}}(t)$ des Shape-Flow-Algorithmus, welche auf einer Initialisierung durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern (Abschnitt 6.3.1) basiert.

6.4 Prädiktion

Dieses Systemmodul (Abbildung 6.1) dient zur Prädiktion der Lage und Orientierung der Objekte über die Zeit. Mit der Annahme einer Bewegung mit konstanter Geschwindigkeit wird die prädizierte 3D-Pose $\widehat{\mathbf{\Phi}}(t + \Delta t)$ für den getrackten Hand-Unterarm-Bereich zum Zeitschritt $(t + \Delta t)$ folgendermaßen berechnet:

$$\widehat{\mathbf{\Phi}}(t + \Delta t) = \mathbf{\Phi}_{\text{Fusion}}(t) + \Delta t \cdot \dot{\mathbf{\Phi}}(t). \tag{6.6}$$

Dabei steht $\mathbf{\Phi}_{\text{Fusion}}(t)$ für die aktuell im Fusionsmodul (Abschnitt 6.2.3) geschätzte 3D-Pose und $\dot{\mathbf{\Phi}}(t)$ für die zeitliche Ableitung des Modells, welche durch die Bewegungsanalyse (Abschnitt 6.3) geschätzt wurde.

6.5 Interaktion im Gesamtsystem

Das entwickelte und in diesem Kapitel beschriebene Trackingsystem (Abbildung 6.1) realisiert die Verfolgung der menschlichen Hand-Unterarm-Region aus Basis von Bildfolgen im 3D-Raum. Die für das Tracking notwendige 3D-Pose-Estimation basiert auf Konturen in den Kamerabildern und 3D-Punktewolken. Die 3D-Punktewolken werden aus den Bildern des verwendeten Kamerasystems durch das Szenenflussmodul berechnet. Das entwickelte System gewährleistet eine automatische Detektion der menschlichen Hand-Unterarm-Region in den spärlichen Szenenflussdaten. Nach der Detektion startet das Tracking, indem neben der 3D-Pose der menschlichen Hand-Unterarm-Region auch die Pose-Ableitung zur Bewegungsvorhersage instantan geschätzt wird. Diese instantane Schätzung ist wichtig, da in einem Sicherheitssystem eine zeitliche Verzögerung, wie z.B. durch ein zeitliches Filter verursacht, zu einer

verzögerten Kollisionsdetektion mit einem Industrieroboter führen kann. Zeitliche Verzögerungen sollten in Sicherheitssystemen weitestgehend minimiert werden. Daher wird bei der im Tracking notwendigen Prädiktion (Abschnitt 6.4) direkt die geschätzte Pose-Ableitung aus der Bewegungsanalyse (Abschnitt 6.3) verwendet.

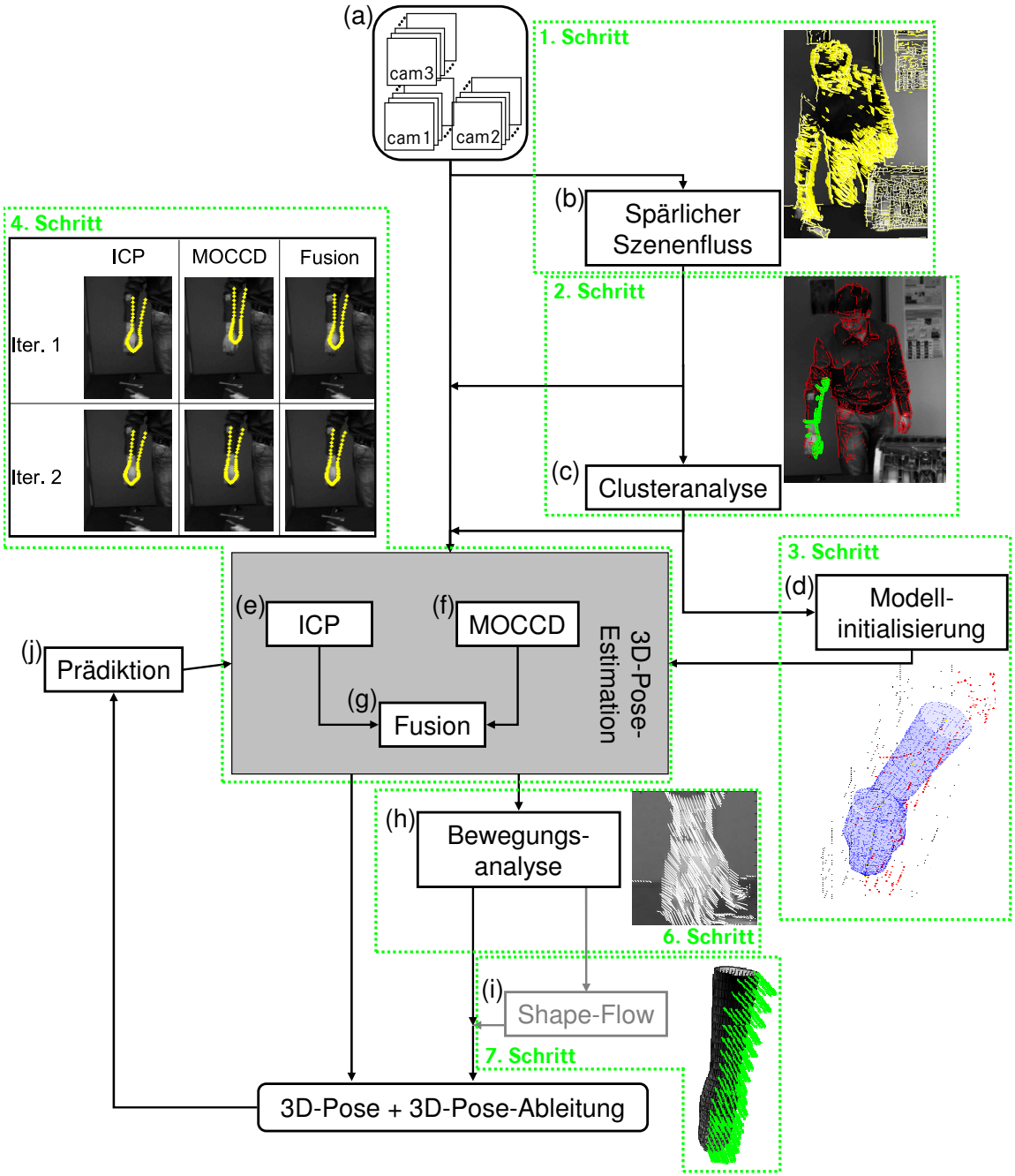

Abbildung 6.7: Objektverfolgung durch 3D-Pose-Estimation basierend auf Konturen und 3D-Punktewolken. Die einzelnen Schritte des Systemablaufs sind farblich hervorgehoben. Der fünfte Schritt des Verfahrens ist nicht dargestellt.

Im Folgenden beziehen sich alle Modulangaben auf Abbildung 6.7. Den Input für alle Systemmodule liefert Modul (a) das trinokulare Kamerasystem (Abschnitt 3.1). Der folgende Systemablauf beschreibt den Tracking-Prozess der menschlichen Hand-Unterarm-Region mit dem entwickelten Verfahren:

1. Berechne mit dem Szenenflussmodul (Modul (b)) aus den Kamerabildern einen spärlichen Szenenfluss des beobachteten Bereichs.
2. Durch eine Clusteranalyse (Modul (c)) werden bewegte und zusammenhängende Objekte in der Punktewolke des spärlichen Szenenfluss ermittelt.
3. Auf Basis der Clusteranalyse wird eine Objektdetektion (Modul (d)) durchgeführt.
4. Berechne für bereits detektierte Objekte, mit den Modulen (e),(f) und (g) die fusionierte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ zum Zeitschritt t unter Nutzung der prädizierten Pose $\widehat{\mathbf{\Phi}}(t)$ als Pose-Initialisierung.
5. Wenn die Verifikation der fusionierten 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ nicht bestanden wird, suche eine „bessere“ Modellinitialisierung in der Nähe durch Modul (d).
6. Berechne für die fusionierte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ die Pose-Ableitung $\dot{\mathbf{\Phi}}(t)$ durch das Modul (h) zur Bewegungsanalyse.
7. Falls Modul (i) verwendet wird, verfeinere die durch Modul (h) ermittelte Pose-Ableitung $\dot{\mathbf{\Phi}}(t)$ mit dem Shape-Flow-Algorithmus, unter Nutzung der fusionierten 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$.
8. Berechnung der Prädiktionen $\widehat{\mathbf{\Phi}}(t + \Delta t)$ für den Zeitschritt $t + \Delta t$.

Im ersten Schritt wird durch das Szenenflussmodul (Modul (b)) ein spärlicher Szenenfluss (Abschnitt 5.2) berechnet, welcher aus 3D-Punkten mit zugehörigen Geschwindigkeiten besteht.

Dieser spärliche Szenenfluss dient im zweiten Schritt als Basis für eine Clusteranalyse (Modul (c), Abschnitt 6.1) zur Ermittlung bewegter und zusammenhängender Objekte in der beobachteten Szene.

Auf den Ergebnissen der Clusteranalyse wird im dritten Schritt eine Objektdetektion (Modul (d), Abschnitt 6.1) durchgeführt, diese entspricht einer Modellinitialisierung, d.h. es wird der Parametervektor $\mathbf{\Phi}(t = 1)$, welcher die 3D-Position und Orientierung des zu verfolgenden Objektes zum ersten Zeitschritt $t = 1$ des Trackings beschreibt, berechnet.

Im vierten Schritt des Systemablaufs wird für bereits detektierte Objekte, mit den Modulen (e),(f) und (g) die fusionierte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ zum Zeitschritt t berechnet (Abschnitt 6.2). Dabei wird die prädizierte Pose $\widehat{\mathbf{\Phi}}(t)$ als Pose-Initialisierung $\mathbf{\Phi}_{\text{init}}(t)$ verwendet. Das Fusionsergebnis $\mathbf{\Phi}_{\text{Fusion}}(t)$ wird iterativ berechnet, indem die Pose-Schätzungen $\mathbf{\Phi}_{\text{ICP}}$ und $\mathbf{\Phi}_{\text{MOCCD}}$ des ICP- und MOCCD-Algorithmus durch drei verschiedene Kriterien gewichtet kombiniert werden.

Im fünften Schritt sorgt eine Verifikation (Abschnitt 3.4) für einen Selbsttest des Systems. Falls die fusionierte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ die Verifikation nicht besteht, wird nach einer „besseren“ Modellinitialisierung in der Nähe von $\mathbf{\Phi}_{\text{Fusion}}(t)$ gesucht. Dabei wird Modul (d) verwendet.

Durch eine Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern (Abschnitt 6.3.1) wird im sechsten Schritt für die fusionierte 3D-Pose-Schätzung $\mathbf{\Phi}_{\text{Fusion}}(t)$ die Pose-

Ableitung $\dot{\mathbf{\Phi}}(t)$ bestimmt. Dabei wird der von Barrois (2010) entwickelte Ansatz auf den spärlichen Szenenflussdaten angewandt, um eine Bewegungsvorhersage für das Hand-Unterarm-Modell zu berechnen.

Der siebte Schritt kann optional verwendet werden. Daher ergeben sich für das in diesem Kapitel beschriebene System zwei Systemvarianten, welche später in den Experimenten untersucht werden. Falls Modul (i) verwendet wird, wird die durch Modul (h) ermittelte Pose-Ableitung $\dot{\mathbf{\Phi}}(t)$ mit dem Shape-Flow-Algorithmus weiter verbessert. Wie im Abschnitt 6.3.2 beschrieben wird mit dem Shape-Flow-Algorithmus die zeitliche Ableitung $\dot{\mathbf{\Phi}}(t)$ der 3D-Pose $\mathbf{\Phi}_{\text{Fusion}(t)}$ aus den Bildern der Zeitschritte $\mathbf{\Phi}(t - \Delta t)$ und $(t - \Delta t)$ berechnet. Durch die Verwendung des Shape-Flow-Algorithmus kann auch die Tiefengeschwindigkeit des Objekts geschätzt werden, was durch die Bewegungsschätzung im Modul (h) nicht möglich ist. Dadurch wird die Bewegungsvorhersage weiter verbessert.

Im achten Schritt des Systemablaufs wird die 3D-Pose $\mathbf{\Phi}_{\text{Fusion}(t)}$ und die geschätzte zeitliche Ableitung des Modells $\dot{\mathbf{\Phi}}(t)$ verwendet, um zum nächsten Zeitschritt die prädizierte 3D-Pose $\widehat{\mathbf{\Phi}}(t + \Delta t)$ zur Pose-Estimation in den Modulen (e),(f) und (g) (vierter Schritt des Systemablaufs) nutzen zu können. Damit beginnt der Ablauf von vorn.

Kapitel 7

Experimentelle Untersuchungen zum Tracking von Körperteilen

In diesem Kapitel werden die Ergebnisse der entwickelten Trackingsysteme auf der Basis von verschiedenen Testsequenzen vorgestellt. Dabei werden die Ergebnisse der Algorithmen mit Ground-Truth-Daten verglichen, um die metrische Genauigkeit zu bestimmen. Es wurden möglichst realistische Testsequenzen gewählt und die Ergebnisse der Systemvarianten quantitativ und qualitativ evaluiert. Als Erstes wird darauf eingegangen wie die notwendige Ground-Truth für die Sequenzen berechnet wird. Danach werden die Systemvariationen der Trackingsysteme in den Kapiteln 3 bis 6, die zur Evaluierung verwendet werden, beschrieben. Es werden dann die Ergebnisse für drei verschiedene Szenarien vorgestellt und am Ende des Kapitels eine Wertung der Ergebnisse gegeben.

7.1 Erzeugung der Ground-Truth

Um 3D-Pose-Estimation Verfahren oder Algorithmen zum 3D-Tracking beurteilen und evaluieren zu können, wird die reale 3D-Position des angemessenen oder getrackten Objekts, beispielsweise des menschlichen Hand-Unterarms, möglichst genau benötigt. Die reale 3D-Position wird als Ground-Truth (GT) bezeichnet. Die Ground-Truth sollte dabei mit einem Referenzsystem berechnet werden, welches eine um den Faktor 10 höhere Genauigkeit als die evaluierten Algorithmen gewährleisten kann. Ground-Truth-Daten werden in dieser Arbeit für die kamerabasierte Verfolgung der menschlichen Hand-Unterarm-Region und der Kopf-Schulter-Partie benötigt. Zur Berechnung der Ground-Truth werden auf dem zu verfolgenden Objekt Marker (Rot-Grün-Ecken) aufgeklebt. Diese Marker werden in den drei Bildern des Kamerasystems automatisch detektiert und verfolgt sowie deren 3D-Position im Weltkoordinatensystem bestimmt.

7.1.1 Ground-Truth für die Hand-Unterarm-Region

Die Ground-Truth besteht aus drei 3D-Punkten im Weltkoordinatensystem W. Diese drei Punkte korrespondieren mit den Punkten ${}^{W}\mathbf{p}_1, {}^{W}\mathbf{p}_4$ und ${}^{W}\mathbf{p}_8$ des Hand-Unterarm-Modells (Abschnitt 3.2.1) und stehen für den Beginn des Unterarms ${}^{W}\mathbf{p}_1$, das Handgelenk ${}^{W}\mathbf{p}_4$ und die Handspitze ${}^{W}\mathbf{p}_8$. Zur Berechnung der Ground-Truth werden auf der zu verfolgenden Hand-Unterarm-Extremität drei Marker (Rot-Grün-Ecken) aufgeklebt und diese in den drei Bildern

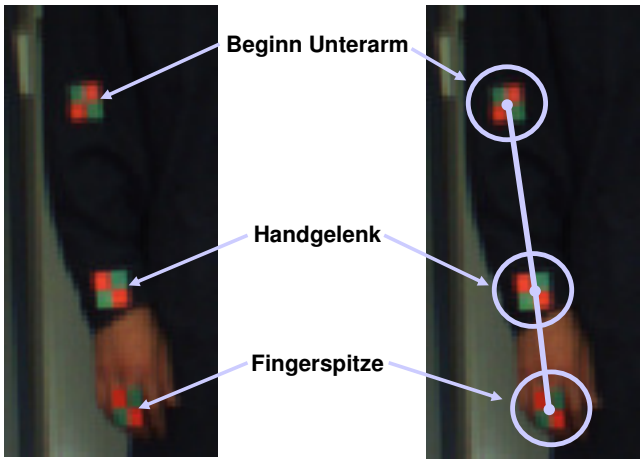

Abbildung 7.1: Als Marker angebrachte Rot-Grün-Ecken zur Berechnung der Ground-Truth für das Hand-Unterarm-Tracking. Zur Evaluierung werden in dieser Arbeit der 3D-Punkt zum Beginn des Unterarms, ein 3D-Punkt im Handgelenk und ein 3D-Punkt in der Nähe der Fingerspitzen verwendet.

des Kamerasystems automatisch detektiert und verfolgt (Abbildung 7.1). Die Grobinitialisierung der Ecke basiert auf einer einfachen Farbklassifikation und hat als Ergebnis eine auf bis zu drei Pixel genaue Position der Rot-Grün-Ecke. Das Ergebnis der Grobinitialisierung wird durch eine Ecken-Feinpositionierung (Krüger u. Wöhler, 2009) verbessert, deren Ausgabe ist eine subpixelgenaue Position einer Ecke. Dies wurde durch Krüger u. Wöhler (2009) auch experimentell nachgewiesen. Da die subpixelgenaue Position einer Ecke in den Pixelkoordinatensystemen der drei Kameras berechnet wurde, kann durch einen Bündelausgleich (Triggs u. a., 2000) die 3D-Position der Ecke im Weltkoordinatensystem berechnet werden.

7.1.2 Ground-Truth für die Kopf-Schulter-Partie

Die Ground-Truth der Kopf-Schulter-Partie berechnet sich ebenfalls aus drei auf dem zu verfolgenden Objekt aufgeklebten Markern (Abbildung 7.2). Die 3D-Position der Marker im Weltkoordinatensystem wird analog zur Berechnung Ground-Truth für die Hand-Unterarm-Region bestimmt. Die Ground-Truth für die Kopf-Schulter-Partie besteht aus dem Zentrum des Halses, welches durch die Mitte der beiden Marker (Rot-Grün-Ecken) auf den Schultern definiert ist (Abbildung 7.2) und den Orientierungen entlang der Blickrichtung von Kopf und Schultern, welche auch aus den Markern berechnet werden können. Die Orientierung der Schulter entlang der Blickrichtung ergibt sich dabei aus den beiden Markern auf den Schultern und die Orientierung des Kopfes durch den Richtungsvektor, welcher durch das Zentrum des Halses und den Marker auf der Stirn definiert ist (Abbildung 7.2).

7.2 Systemvarianten

In den Kapiteln 3 bis 6 wurden mehrere Systeme zum Tracking von Körperteilen vorgestellt. Da es zu jedem System mehrere Ausprägungen durch das Verwenden oder Weglassen einzel-

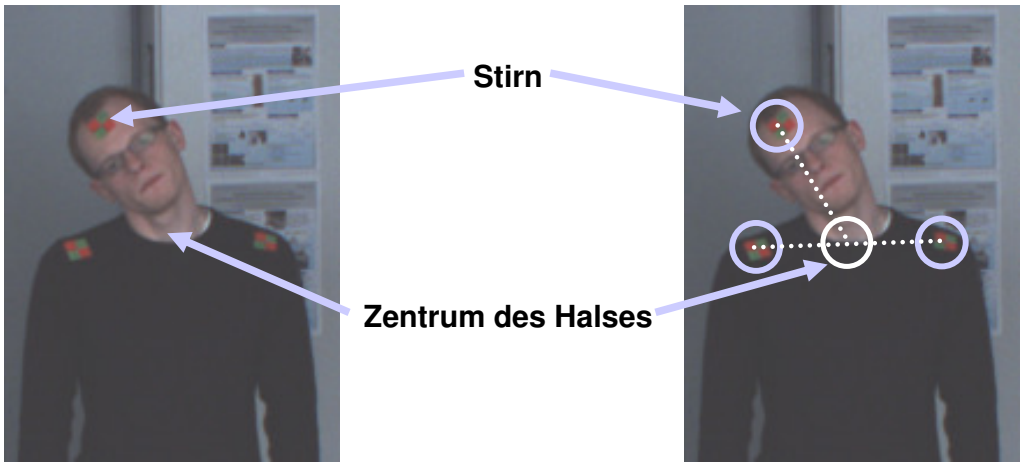

Abbildung 7.2: Angebrachte Marker (Rot-Grün-Ecken) zur Berechnung der Ground-Truth für das Kopf-Schulter-Tracking. Zur Evaluierung werden in dieser Arbeit durch Marker bestimmte 3D-Punkte auf den Schultern und der Stirn verwendet und daraus das Zentrum des Halses und die Orientierung von Schulter und Kopf berechnet.

ner Module gibt, wird ein kurzer Überblick über die verschiedenen Varianten der evaluierten Trackingsysteme gegeben.

System 1 – MOCCD-Tracking: Im Kapitel 3 wurde ein technisches System zur Verfolgung von 3D-Konturen vorgestellt. Das getrackte menschliche Körperteil wird durch ein 3D-Konturmodell beschrieben und die Anpassung des 3D-Konturmodells an die Bilddaten wird mit Hilfe des MOCCD-Algorithmus durchgeführt. Das Systemmodell ist als Multi-Hypothesen-Trackingsystem aufgebaut. Die Schätzung der 3D-Pose, also die Anpassung des 3D-Konturmodells an die Kamerabilder, wird durch drei MOCCD-Algorithmen bestimmt. Die notwendige 3D-Pose-Prädiktion über die Zeit, wird durch drei zu den MOCCD-Algorithmen zugehörige Kalman-Filter berechnet. In jedem der drei Kalman-Filter ist ein unterschiedliches Bewegungsmodell implementiert. Die Eingabebilder für System 1 können RGB- oder Grauwertbilder sein.

System 2 – MOCCD-Tracking mit Reinitialisierung: Die zweite Systemvariante unterscheidet sich von der Ersten nur in der Verwendung eines Reinitialisierungsmoduls. Durch eine Verifikation wird die zeitliche Konsistenz des Objektaussehens überprüft. Falls dieser Selbsttest nicht bestanden wird, kann durch das Reinitialisierungsmodul das getrackte Objekt wiedergefunden werden. Auch dieses Trackingsystem wurde im Kapitel 3 beschrieben.

System 3 – Shape-Flow-Tracking: Das dritte System wurde in Kapitel 4 beschrieben und realisiert eine Verfolgung von raum-zeitlichen 3D-Konturen. Wichtig bei diesem System ist, dass das getrackte menschliche Körperteil durch ein raum-zeitliches 3D-Konturmodell beschrieben werden kann. Die zeitlichen Filter in den Systemen 1 und 2 werden durch eine direkt gemessene Bewegungsschätzung ersetzt, diese wird durch den Shape-Flow-Algorithmus realisiert. Es werden im Vergleich zu den Systemen 1 und 2 nur zwei Hypothesen zum Tracking verwendet. Die 3D-Konturanpassung wird durch den MOCCD-Algorithmus durchgeführt und die Objektbewegungsschätzung mit dem Shape-Flow-

Algorithmus realisiert. Die Eingabebilder für System 3 können RGB- oder Grauwertbilder sein.

System 4 – Shape-Flow-Tracking mit Reinitialisierung: Die vierte Systemvariante entsteht durch Verwendung eines Reinitialisierungsmoduls für das Shape-Flow-Tracking (Kapitel 4). Durch das Reinitialisierungsmodul wird versucht das getrackte Objekt wiederzufinden, wenn der Selbsttest des Systems nicht bestanden wurde.

System 5 – Mean-Shift-Tracking: System 5 ist ein Verfahren zur Verfolgung beliebiger Objekte in Kamerabildern und 3D-Punktewolken (Kapitel 5). Die in den Kamerabildern getrackten Objekte werden nicht detailliert beschrieben, sondern durch Ellipsoide modelliert. Falls ein in den Bildern des Kamerasystems getracktes Objekt nicht mit nur einem Ellipsoiden beschrieben werden kann, werden mehrere Ellipsoide zur Beschreibung des Objekts eingesetzt und diese getrennt voneinander in den Bildern getrackt. Das System verwendet zur Pose-Estimation keine Konturen, wie die Systeme 1 bis 4, sondern Bilddaten und 3D-Punktewolken. Die eingesetzte Pose-Estimation basiert auf dem Mean-Shift-Algorithmus. Als Eingabedaten für die Pose-Estimation können Grauwertbilder oder in den HSV-Farbraum transformierte RGB-Bilder verwendet werden.

System 6 – ICP-MOCCD-Tracking: In Kapitel 6 wird ein sehr komplexes System zur Objektverfolgung durch 3D-Pose-Estimation basierend auf Konturen und 3D-Punktewolken vorgestellt. Im entwickelten Trackingsystem werden zwei unterschiedliche Ansätze zur 3D-Pose-Estimation miteinander gewichtet kombiniert. Dabei wird ein Verfahren verwendet, welches basierend auf bewegten und geclusterten Szenenflussdaten mit Hilfe des ICP-Algorithmus das verwendete 3D-Modell anpasst. Zum anderen wird der MOCCD-Algorithmus eingesetzt, um das Objektmodell an die Bilddaten anzupassen. Ein Vorteil der verwendeten Ansätze ist, dass sie auf unterschiedlichen Eingabedaten basieren und somit auch einen Ausfall des jeweils anderen Verfahrens kompensieren können. Die Fusion der zwei Pose-Schätzungen basiert auf drei verschiedenen Ähnlichkeitsmaßen. Durch eine Bewegungsanalyse basierend auf Szenenflussdaten wird die Position des getrackten Objekts über die Zeit vorhergesagt.

System 7 – ICP-MOCCD-Shape-Flow-Tracking: Der Unterschied zum sechsten System ist, dass die vorhandene Bewegungsanalyse basierend auf Szenenflussdaten, durch den Shape-Flow-Algorithmus erweitert wird. Dadurch ist es möglich auch die Tiefengeschwindigkeit des Objekts vorherzusagen. Durch dieses Trackingsystem kann instantan die 3D-Pose und deren vollständige Ableitung nach der Zeit aus den Bilddaten geschätzt werden.

7.3 Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs

In diesem Abschnitt werden die Ergebnisse verschiedener experimenteller Untersuchungen mit den definierten Systemvarianten (Abschnitt 7.2) auf Basis von neun realen Testsequenzen ausgewertet. Die Sequenzen wurden mit dem in Abschnitt 3.1 beschriebenen Kamerasystem aufgezeichnet. Die drei Sensoren des Kamerasystems sind rechtwinklig angeordnet und haben einen horizontalen und vertikalen Abstand von 150 mm. Die einzelnen Sensoren liefern bis zu 14 Bil-

der pro Sekunde ($\Delta t = 71$ ms) in einer Auflösung von 516×389 Pixeln. Die Sequenzen 1 bis 5 zeigen verschiedene Arbeitsschritte bei der Motorenmontage. Der Prozessablauf wurde in einer Büroumgebung nachgestellt. Die Testsequenzen 6 bis 9 zeigen Montageprozesse in einer Arbeitszelle der Produktions- und Werkstofftechnik (PWT) im Daimler-Werk in Sindelfingen. Die Testsequenzen enthalten Bewegungen verschiedener Hand-Unterarm-Konfigurationen vor stark strukturierten Hintergründen. In allen Testsequenzen ist der Kontrast zwischen den beobachteten Personen und dem Hintergrund oftmals gering, beispielsweise in Abbildung 7.11 zu sehen. Die beobachteten Personen tragen ein T-Shirt, einen Pullover oder einen Arbeitsmantel. Der Unterarm ist dadurch sichtbar, teilweise oder vollständig bedeckt. Außerdem werden in zwei der Sequenzen auch Handschuhe von der beobachteten Person getragen. Die mittlere Entfernung der beobachteten Person zur Kamera variiert zwischen 2.7 m und 3.3 m. Die Sequenzen bestehen im Mittel aus 380 Bildtripeln. Die konstanten Längenparameter des Hand-Unterarm-Modells (Abschnitt 3.2.1) sind dabei für alle Sequenzen gleich und werden zu $l_{\text{Unterarm}} = 210$ mm und $l_{\text{Hand}} = 14$ mm gewählt. D.h. das Modell wird nicht spezifisch auf eine Person angepasst, sondern die Deformationsparameter werden mitgeschätzt, siehe Kapitel 3 und 4. Die neun Testsequenzen gibt es mit zugehörigen Ground-Truth-Daten im Internet zum Download unter `http://aiweb.techfak.uni-bielefeld.de/content/hand-forearm-limb-data-set`.

Die Ergebnisse aller Systemvarianten für das Tracking der menschlichen Hand-Unterarm-Region werden qualitativ und quantitativ auf Basis aller neun Testsequenzen evaluiert. Als Ergebnis eines Testlaufs einer der Systemvarianten (Abschnitt 7.2) auf einer Testsequenz ergibt sich für jeden Zeitschritt der Sequenz die euklidische Distanz zwischen den durch das Trackingsystem geschätzten Punkten ${}^{W}\mathbf{p}_1, {}^{W}\mathbf{p}_4$ und ${}^{W}\mathbf{p}_8$ des Hand-Unterarm-Modells (Abschnitt 3.2.1) und deren Ground-Truth (Abschnitt 7.1.1). Damit kann die mittlere euklidische Distanz der kompletten Sequenz für jeden der drei Punkte berechnet werden. Zur Auswertung einer Systemvariante werden die Ergebnisse aller Sequenzen in einem Diagramm mit Fehlerbalken abgetragen. Die Fehlerbalken für eine Testsequenz werden dabei durch den Mittelwert und die Standardabweichung der euklidischen Distanzen der geschätzten 3D-Punkte zur Ground-Truth definiert. Ein solches Fehlerbalkendiagramm für alle Testsequenzen ist für das System 1 (MOCCD-Tracking) und die Verwendung von Grauwertbildern als Eingabedaten in Abbildung 7.3 (links) dargestellt. Im Folgenden werden die euklidischen Distanzen der geschätzten 3D-Punkte zur Ground-Truth als Positionsfehler bezeichnet.

Weiterhin werden die Fehler der instantanen Bewegungsschätzung für alle Sequenzen und Systemvarianten quantitativ evaluiert. Dabei werden die ermittelten Werte für die Horizontal-, Vertikal- und Tiefengeschwindigkeit mit der Geschwindigkeit der Ground-Truth verglichen. Die Bewegungsschätzung ist zur Vorhersage einer möglichen Kollision mit einem Industrieroboter sehr wichtig. Instantane Bewegungsschätzung bedeutet, dass die Geschwindigkeit direkt aus den Daten geschätzt und nicht durch ein zeitliches Filter ermittelt wird. Bei einem Sicherheitssystem ist es immens wichtig, die Latenzzeit einer möglichen Reaktion auf die Bewegungsvorhersage gering zu halten. Einschwingphasen eines zeitlichen Filters würden die Latenzzeit erhöhen, daher wird bei der quantitativen Evaluierung der instantanen Bewegungsschätzung auf die Verwendung eines Filters verzichtet und direkt die geschätzte Geschwindigkeit oder die zeitliche Differenz der Pose-Schätzungen als Geschwindigkeitsapproximation verwendet (Abbildung 7.3 (rechts)).

7.3.1 Ergebnisse System 1 – MOCCD-Tracking

Die Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth für System 1 (MOCCD-Tracking) sind in Abbildung 7.3 dargestellt. Als Eingabedaten wurden Grauwertbilder verwendet. Nur in den Sequenzen 2, 3 und 4 wird der Unterarm vollständig über die Zeit getrackt (Verfolgungsrate von 100%), dies ist in Abbildung 7.3 (links) an den Labels am oberen Ende der Fehlerbalken für den Positionsfehler zu erkennen. Der mittlere Positionsfehler für die Sequenzen 2, 3 und 4 liegt im Bereich von 50 bis 70 mm, was ein gutes Ergebnis ist. Die mittlere Standardabweichung der Positionsfehler für die Sequenzen 2, 3 und 4 beträgt 32 mm.

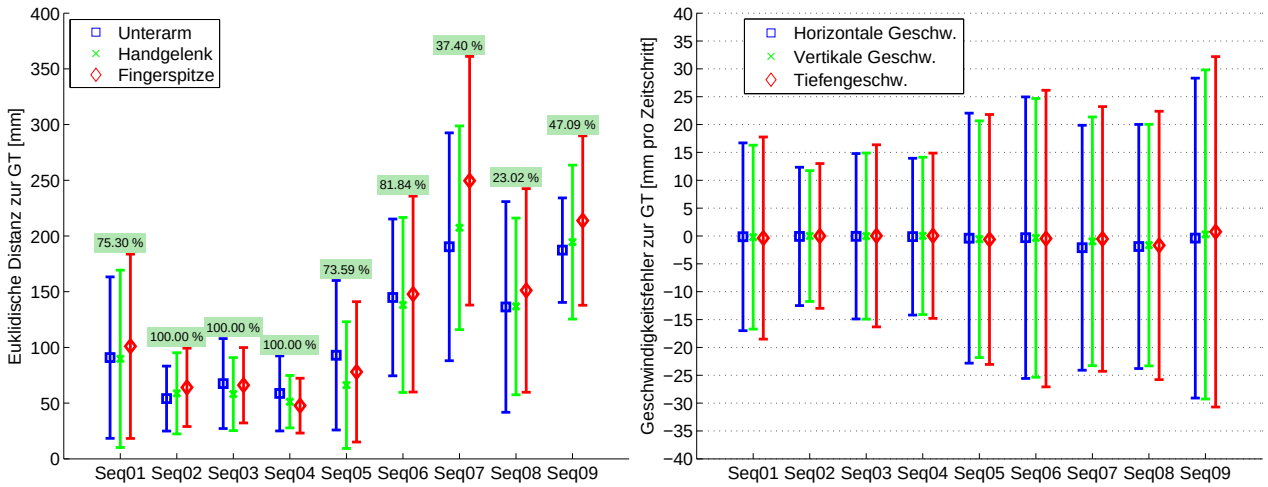

Abbildung 7.3: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 1**. Das Trackingsystem wurde in Kapitel 3 beschrieben und basiert auf drei MOCCD-Algorithmen und drei zugehörigen Kalman-Filtern zur Prädiktion der Bewegung. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde aus den Differenzen unmittelbar aufeinander folgender Pose-Schätzungen ermittelt (ohne Filterung).

In den Sequenzen 1, 5 und 6 wird der Unterarm zu mindestens 70% der Zeit getrackt, die Positionsfehler liegen im Bereich von 80 bis 140 mm. Im Vergleich zu den Sequenzen 2, 3 und 4 sind die Positionsfehler fast doppelt so groß, dies zeigt sich auch für die Standardabweichung des Positionsfehlers. In den Sequenzen 7, 8 und 9 beträgt die Verfolgungsrate des Unterarms weniger als 50% und die Positionsfehler erhöhen sich nochmals auf 150 bis 250 mm.

Abbildung 7.3 (rechts) zeigt die Fehler der instantanen Bewegungsschätzung. Die Mittelwerte der Geschwindigkeitsfehler sind für jede Koordinate nahe Null, was zeigt, dass es keine systematischen Abweichungen gibt. Bei einer systematischen Abweichung wäre der mittlere Fehler in eine Richtung erhöht. Die Standardabweichung der Geschwindigkeitsfehler liegt für System 1 im Bereich von 15 bis 30 mm pro Zeitschritt. Abbildung 7.4 zeigt die ins Bild projizierten Trackingergebnisse des Systems 1 (MOCCD-Tracking) auf einigen Beispielsequenzen.

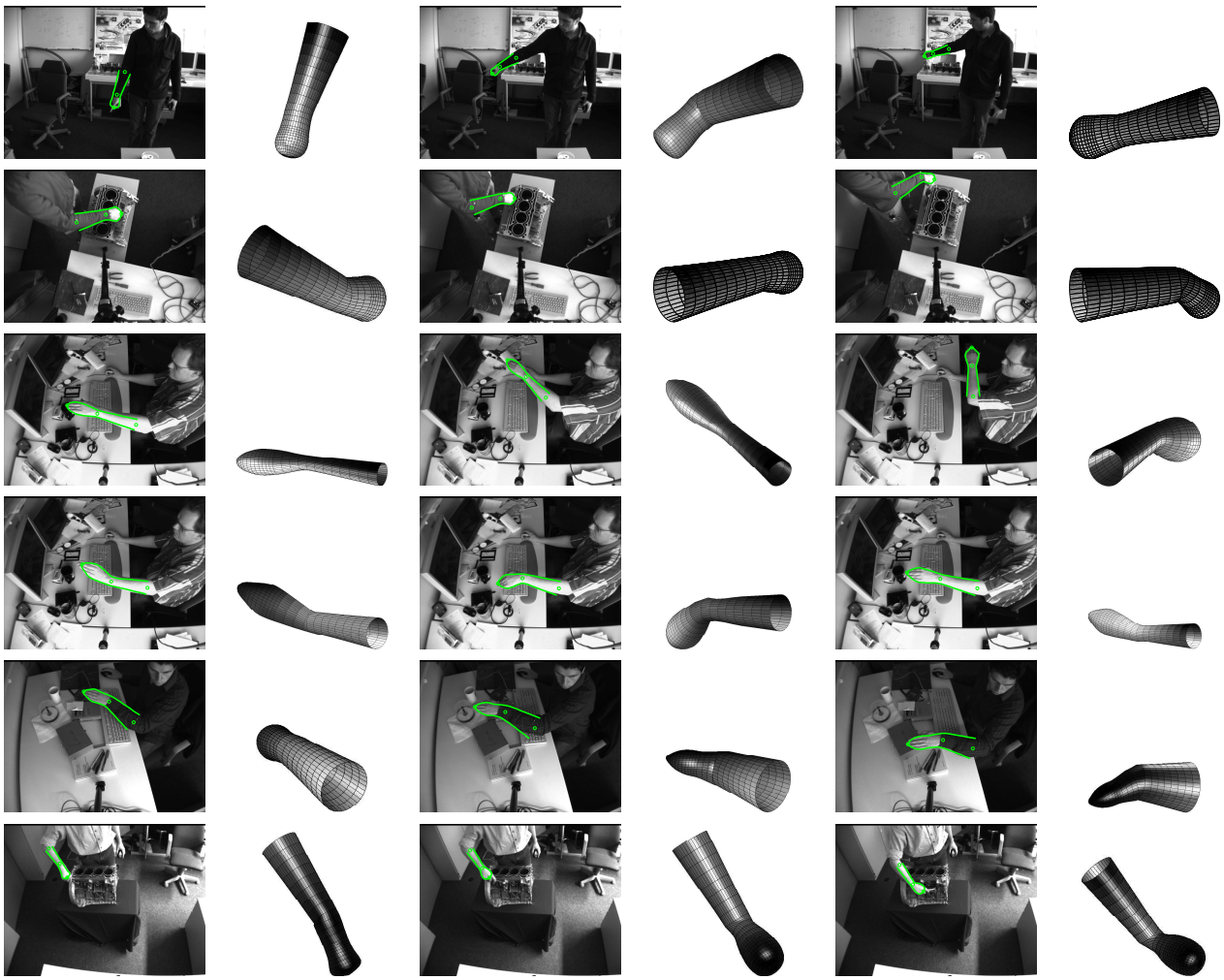

Abbildung 7.4: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs auf einigen Beispielsequenzen. Die Spalten 2, 4 und 6 zeigen 3D-Rekonstruktionen des Modells. Es ist gut zu erkennen, dass die Deformation des Modells durch den MOCCD-Algorithmus adaptiert wird. Dadurch wird nicht nur eine Pose-Estimation des Modells sondern auch eine Deformationsschätzung erreicht.

Die mittlere Laufzeit der Matlab-Implementierung von System 1 (Grauwertbilder) auf Basis aller Testsequenzen beträgt 15 Sekunden pro Bildtripel auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Bei der Implementierung des Algorithmus handelt es sich um unoptimierten Matlab-Code. Ein Großteil der Laufzeit wird durch die drei MOCCD-Algorithmen verbraucht. Hier ist die Berechnung der Statistiken entlang aller Senkrechten des Konturmodell (Schritt 2 im Abschnitt 3.3.1) sowie die Berechnung der ersten und zweiten Ableitung der Zielfunktion entlang aller Senkrechten(Schritt 3 im Abschnitt 3.3.1) sehr aufwendig. Durch eine effektive C-Implementierung sind starke Verbesserungen im Laufzeitverhalten zu erwarten.

Die Ergebnisse für System 1 mit RGB-Bildern als Eingabedaten sind in Abbildung 7.5 dargestellt. In den Sequenzen 1 bis 5 wird der Unterarm vollständig über die Zeit getrackt (Abbildung 7.5 (links)). Der mittlere Positionsfehler für diese Sequenzen beträgt zwischen

40 und 70 mm, was im Mittel etwas besser ist als bei den Ergebnissen auf Basis von Grauwertbildern. Die Standardabweichung der Positionsfehler liegt im Bereich von 20 bis 50 mm. Die Sequenzen 6 bis 9 weisen weitaus schlechtere Ergebnisse auf und der Unterarm wird im Mittel nur zu 40% der Zeit getrackt. Im Vergleich zu den Sequenzen 1 bis 5 ist der Positionsfehler mehr als doppelt so groß, was sich auch für die Standardabweichung des Positionsfehlers zeigt.

Abbildung 7.5 (rechts) zeigt die Fehler der instantanen Bewegungsschätzung. Wiederum sind die Mittelwerte der Geschwindigkeitsfehler für jede Koordinate nahe Null, was zeigt, dass es keine systematischen Abweichungen gibt. Die Standardabweichungen der Geschwindigkeitsfehler unterscheiden sich nicht signifikant zu denen von System 1 unter der Verwendung von Grauwertdaten.

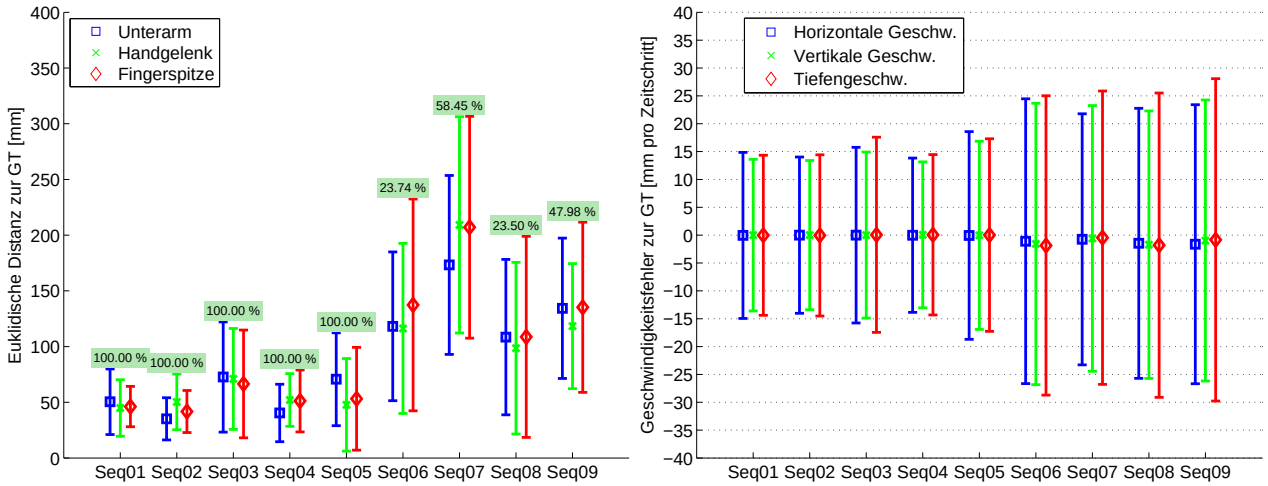

Abbildung 7.5: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 1**. Das Trackingsystem wurde in Kapitel 3 beschrieben und basiert auf drei MOCCD-Algorithmen und drei zugehörigen Kalman-Filtern zur Prädiktion der Bewegung. Die Eingabedaten sind **Farbbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde aus den Differenzen unmittelbar aufeinander folgender Pose-Schätzungen ermittelt (ohne Filterung).

Die mittlere Laufzeit der Matlab-Implementierung von System 1 (RGB-Bildern) auf Basis aller Testsequenzen beträgt 23 Sekunden pro Bildtripel auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Bei der Implementierung handelt es sich wieder um unoptimierten Matlab-Code. Der Unterschied zum System 1 mit Grauwertbildern als Eingabedaten ist durch den erhöhten Aufwand für die Berechnung und Trennung der RGB-Statistiken zu begründen. Bei der Grauwertvariante des MOCCD-Algorithmus werden die Statistiken entlang der Senkrechten durch Mittelwert und Standardabweichung beschrieben (beides skalare Werte) und bei der RBG-Variante durch einen Mittelwertvektor und eine Kovarianzmatrix. Details zur Berechnung der Pixelstatistiken im MOCCD-Algorithmus finden sich in Abschnitt 3.3.1. Ein Großteil der Laufzeit wird wiederum durch die drei MOCCD-Algorithmen verbraucht.

Die Ergebnisse des Systems 1 (MOCCD-Tracking) auf Farb- oder Grauwertbildern sind für ein Sicherheitssystem nicht ausreichend, da bei einem Teil der Testsequenzen der Unterarm

nicht vollständig über die Zeit getrackt werden kann. Die Gründe liegen zum einen in den teilweise vorhandenen schnellen Umkehrbewegungen der Testpersonen, denn durch eine gewisse Trägheit der verwendeten Filter kann es zu Fehlprädiktionen kommen. Zum anderen liegen die Gründe im MOCCD-Algorithmus selbst und in dem verwendeten groben Modell des Hand-Unterarm-Bereichs. Der MOCCD-Algorithmus ist ein Optimierungsalgorithmus zweiter Ordnung, daher kann das Ergebnis zum Zeitschritt t stark vom Ergebnis zum Zeitschritt $t+1$ abweichen, denn der Algorithmus kann in t in ein schlechtes lokales Minimum konvergieren und in $t+1$ ein besseres Minimum finden. Da das Ergebnis des MOCCD-Algorithmus als Messung der einzige Input für Kalman-Filter ist, kann es zu Fehlprädiktionen der Filter kommen, falls es zu den beschriebenen starken Abweichungen kommt. Da der MOCCD-Algorithmus ein Pose-Verfeinerungs-Algorithmus ist und bei einer Fehlprädiktion die prädizierte Modellkurve weit von der realen Objektkontur entfernt ist, ist es nicht möglich, die Modellkurve an die Objektkontur anzupassen und die Objektverfolgung scheitert. Auch die Verwendung von Farbdaten zur 3D-Pose-Estimation mit dem MOCCD-Algorithmus bringt keine signifikante Verbesserung. Der Grund für das teilweise unterschiedliche Verhalten auf den Sequenzen liegt an starken Reflexionen, Schattenbildungen und einem extrem geringen Kontrast zum Hintergrund in den Sequenzen 6 bis 9.

Durch die Verwendung eines Reinitialisierungsmoduls kann die zeitliche Robustheit des Trackingsystems gesteigert werden. Durch ein Verifikationsmodul wird ein Selbsttest durchführt. Wird dieser nicht bestanden, kann durch das Reinitialisierungsmodul das getrackte Objekt wiedergefunden werden.

7.3.2 Ergebnisse System 2 – MOCCD-Tracking mit Reinitialisierung

Die Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth für System 2 (MOCCD-Tracking mit Reinitialisierungsmodul) auf Basis der Testsequenzen sind in Abbildung 7.6 dargestellt. Als Eingabedaten wurden nur Grauwertbilder verwendet, da bereits gezeigt wurde, dass die Verwendung von Farbdaten keine signifikanten Verbesserungen bringt. In allen Testsequenzen wird der Unterarm vollständig über die Zeit getrackt (Abbildung 7.6 (links)). Es ist wiederum ein Unterschied zwischen den Sequenzen erkennbar. In den Sequenzen 1 bis 5 beträgt der mittlere Positionsfehler zwischen 40 und 70 mm, bei einer mittleren Standardabweichung von 30 mm. Die Positionsfehler in den Sequenzen 6 bis 9 sind etwas erhöht und liegen in einem Bereich von 75 bis 105 mm. Die mittlere Standardabweichung der Positionsfehler in den Sequenzen 6 bis 9 beträgt 60 mm und ist damit doppelt so groß wie für die Sequenzen 1 bis 5. Der Grund für das unterschiedliche Verhalten auf den Sequenzen liegt an den unterschiedlichen Aufnahmebedingungen, wodurch die Pose-Estimation mit dem MOCCD-Algorithmus auf den Sequenzen 6 bis 9 erschwert wird. Wichtig ist die Tatsache, dass durch die Verwendung eines Reinitialisierungsmoduls die zeitliche Robustheit des Trackingsystems gesteigert wird, was dazu führt, dass in einem Sicherheitssystem zur Realisierung des in dieser Arbeit angestrebten Szenarios die Verwendung eines Verifikations- und Reinitialisierungsmoduls zwingend notwendig ist.

Abbildung 7.6 (rechts) zeigt die Fehler der instantanen Bewegungsschätzung. Wiederum sind die Mittelwerte der Geschwindigkeitsfehler für jede Koordinate nahe Null, daher gibt es keine erkennbaren systematischen Abweichungen. Die Standardabweichungen der Geschwin-

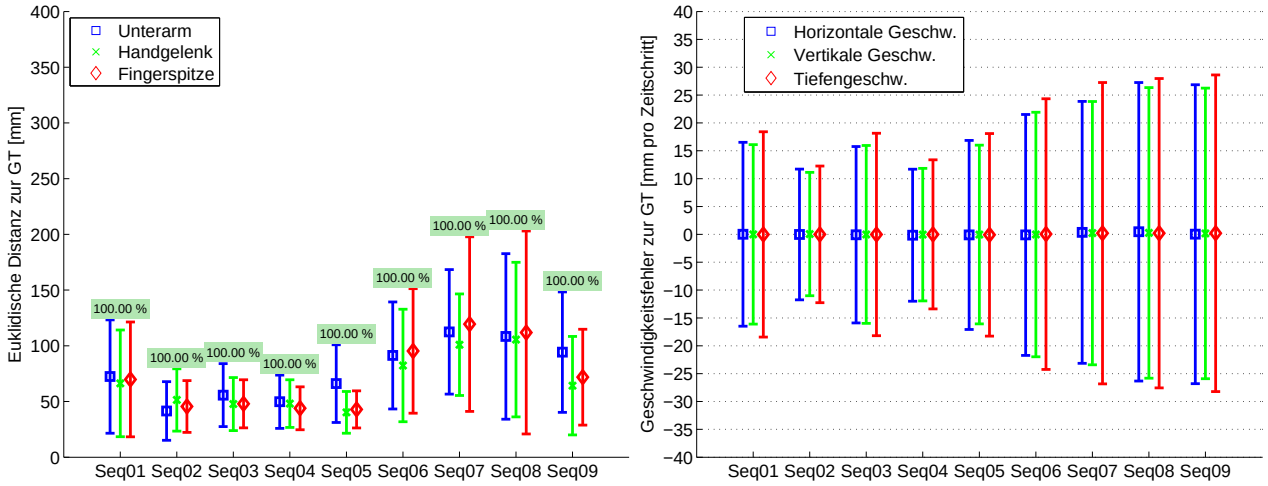

Abbildung 7.6: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 2**. Durch die Verwendung des **Reinitialisierungsmoduls** wird das Tracking zeitlich robuster, alle Sequenzen werden über die komplette Zeit getrackt. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde aus den Differenzen unmittelbar aufeinander folgender Pose-Schätzungen ermittelt (ohne Filterung).

digkeitsfehler unterscheiden sich nicht signifikant zu denen von System 1. Die Geschwindigkeit wurde wie in den Systemen zuvor nicht durch ein zeitliches Filter, sondern durch die Differenz der Pose-Schätzungen bestimmt. Es zeigt sich also, dass das Reinitialisierungsmodul keinen signifikanten Einfluss auf die Geschwindigkeitsschätzung besitzt. Im nächsten Versuch soll durch Verwendung des Shape-Flow-Algorithmus die Genauigkeit der instantanen Bewegungsvorhersage durch ein direktes Messen aus den Bilddaten verbessert werden.

Die mittlere Laufzeit der Matlab-Implementierung von System 2 auf Basis aller Testsequenzen beträgt 18 Sekunden pro Bildtripel auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Bei der Implementierung des Algorithmus handelt es sich um unoptimierten Matlab-Code. Der Laufzeitunterschied zum System 1 wird durch das Reinitialisierungsmodul hervorgerufen. Falls das Reinitialisierungsmodul angewandt wird, erhöht sich die Laufzeit des Algorithmus um das Vierfache. Im Mittel wird das Reinitialisierungsmodul alle 17 Zeitschritte angewandt.

7.3.3 Ergebnisse System 3 – Shape-Flow-Tracking

Die Ergebnisse der kamerabasierten Verfolgung des Unterarms für System 3 (Shape-Flow-Tracking) auf Basis der Testsequenzen sind in Abbildung 7.7 dargestellt. Als Eingabedaten wurden Grauwertbilder verwendet. Nur in den Sequenzen 1, 2, 4 und 5 wird der Unterarm vollständig über die Zeit getrackt (Abbildung 7.7 (links)). Wiederum ist eine klare Trennung zwischen den Sequenzen 1 bis 5 und den Sequenzen 6 bis 9 zu erkennen. Die mittleren Positionsfehler der Sequenzen 1 bis 5 liegen im Bereich von 50 bis 65 mm, was den Ergebnissen des zweiten Systems (MOCCD-Tracking mit Reinitialisierung) entspricht. Die Standardabwei-

chung der Positionsfehler für die Sequenzen 1 bis 5 liegt zwischen 27 und 45 mm.

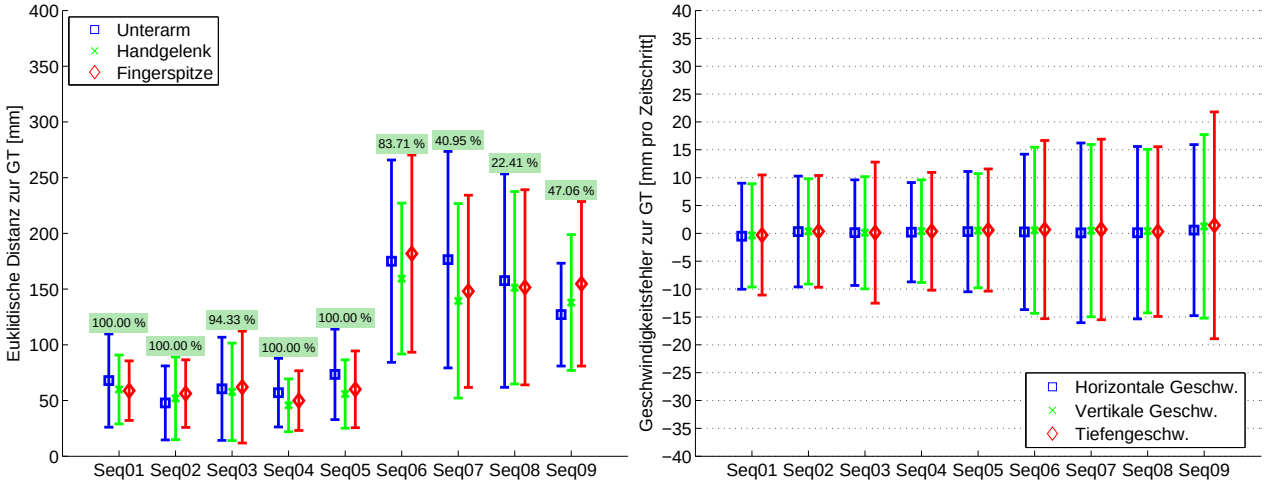

Abbildung 7.7: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 3**. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde durch Shape-Flow-Algorithmus direkt berechnet.

In den Sequenzen 6 bis 9 wird der Unterarm im Mittel nur zu 50% der Zeit getrackt und die mittleren Postionsfehler liegen im Bereich von 140 bis 170 mm. Im Vergleich zu System 2 sind die Positionsfehler der Sequenzen 6 bis 9 um mindestens das Anderthalbfache erhöht. Auch im Vergleich zu den Ergebnissen auf Basis der Sequenzen 1 bis 5 des Systems 3 sind die Standardabweichungen der Postionsfehler erhöht. Dies gilt außerdem im Vergleich zu den Sequenzen 6 bis 9 des Systems 2. Damit zeigt sich, dass auch die direkte Schätzung der Bewegung es nicht ermöglicht, die Positionsgenauigkeit des Trackingsystems grundlegend zu verbessern. Nur die Verwendung eines Reinitialisierungsmoduls kann die zeitliche Robustheit des Trackingsystems steigern.

Abbildung 7.7 (rechts) zeigt die Fehler der instantanen Bewegungsschätzung. Hier zeigt sich im Vergleich zu den vorherigen Systemen eine klare Verbesserung. Dies wurde durch die direkte Messung der Bewegung mit dem Shape-Flow-Algorithmus auch erwartet. Die Mittelwerte der Geschwindigkeitsfehler sind für jede der Koordinaten nahe Null. Die mittlere Standardabweichung der Geschwindigkeitsfehler liegt für System 3 und die Sequenzen 1 bis 5 bei 10 mm pro Zeitschritt, die Fehler in der Bewegungsschätzung konnten also um ca. 30% durch die direkte Messung mit dem Shape-Flow-Algorithmus verringert werden. In Pixeln ausgedrückt, bedeutet eine Standardabweichung des Geschwindigkeitsfehlers von 10 mm pro Zeitschritt und einer ungefähren Entfernung zur Kamera von 3 m einen Fehler von 2 Pixeln pro Zeitschritt, was bei dem verwendeten groben Konturmodell ein guter Wert ist. Für die Sequenzen 6 bis 9 beträgt die mittlere Standardabweichung der Geschwindigkeitsfehler 15 mm pro Zeitschritt, d.h. der Fehler konnte im Vergleich zu System 2 im Mittel um 10 mm pro Zeitschritt gesenkt werden.

Die mittlere Laufzeit der Matlab-Implementierung von System 3 mit Grauwertbildern als Eingabedaten auf Basis aller Testsequenzen beträgt 25 Sekunden pro Zeitschritt auf einem

Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Bei der Implementierung des Algorithmus handelt es sich um unoptimierten Matlab-Code. Der Laufzeitunterschied zum System 1 wird durch die Verwendung des Shape-Flow-Algorithmus hervorgerufen. Zwar kann die Anzahl der zu berechnenden Hypothesen im Vergleich zu System 1 auf zwei verringert werden, es müssen aber insgesamt drei Bildtripel anstatt einem Bildtripel analysiert werden.

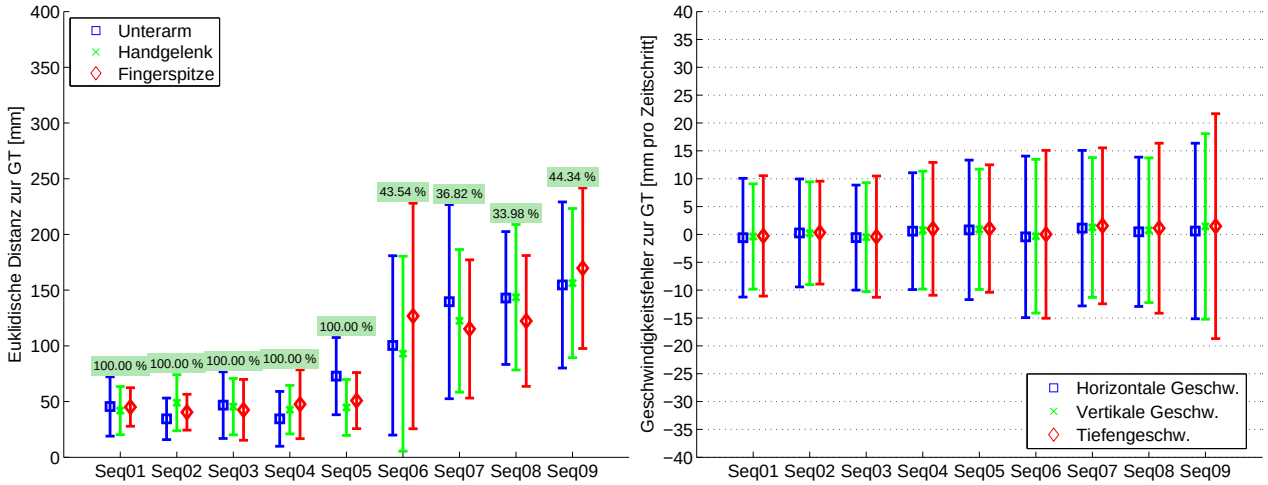

Abbildung 7.8: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 3**. Die Eingabedaten sind **Farbbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde durch den Shape-Flow-Algorithmus direkt berechnet.

Die Ergebnisse für System 3 mit RGB-Bildern als Eingabedaten sind in Abbildung 7.8 dargestellt. Im Vergleich zur Verwendung von Grauwerten (Abbildung 7.7) ergeben sich keine signifikanten Unterschiede. Diese Aussage ist wichtig, da sich durch die Verwendung von Grauwertdaten die Laufzeit der Algorithmen verkürzen lässt. Die Verwendung einer Grauwertkamera sorgt dafür, dass mehr Licht auf dem Sensor auftreffen kann, da keine Farbfilter, wie bei Farbkameras häufig üblich, verwendet werden müssen. Eine höhere Lichtstärke ist für ein industrielles Sicherheitsszenario sehr interessant, da die Leistung und Anzahl der verwendeten Lampen reduziert werden kann oder die Robustheit der Algorithmen durch eine Kontrastvergrößerung erhöht wird.

Die mittlere Laufzeit der Matlab-Implementierung von System 3 mit RGB-Bildern als Eingabedaten auf Basis aller Testsequenzen beträgt 38 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Die Gründe für den Unterschied zu System 3 mit Grauwertbildern als Eingabedaten wurden bereits bei System 1 beschrieben.

7.3.4 Ergebnisse System 4 – Shape-Flow-Tracking mit Reinitialisierung

Die Ergebnisse der kamerabasierten Verfolgung des Unterarms für System 4 (Shape-Flow-Tracking mit Reinitialisierungsmodul) auf Basis der Testsequenzen sind in Abbildung 7.9

dargestellt. Als Eingabedaten wurden Grauwertbilder verwendet. Wie bei den Ergebnissen von System 2 wird der Unterarm in allen Testsequenzen vollständig über die Zeit getrackt (Abbildung 7.9 (links)). Ein Unterschied zwischen den Sequenzen 1 bis 5 und 6 bis 9 ist im Vergleich zu den Ergebnissen von System 2 kaum noch zu erkennen. Die mittleren Positionsfehler aller Sequenzen liegen zwischen 45 und 90 mm, bei Standardabweichungen der Positionsfehler von 20 bis 50 mm. Dies bedeutet eine starke Verbesserung im Vergleich zu System 3 und eine leichte Verbesserung im Vergleich zu System 2. Wiederum führt die Verwendung des Reinitialisierungsmoduls zu einer Erhöhung der zeitlichen Robustheit des Trackings. Die einzelnen Komponenten der Geschwindigkeitsfehler (Abbildung 7.9 (rechts)) zeigen keine

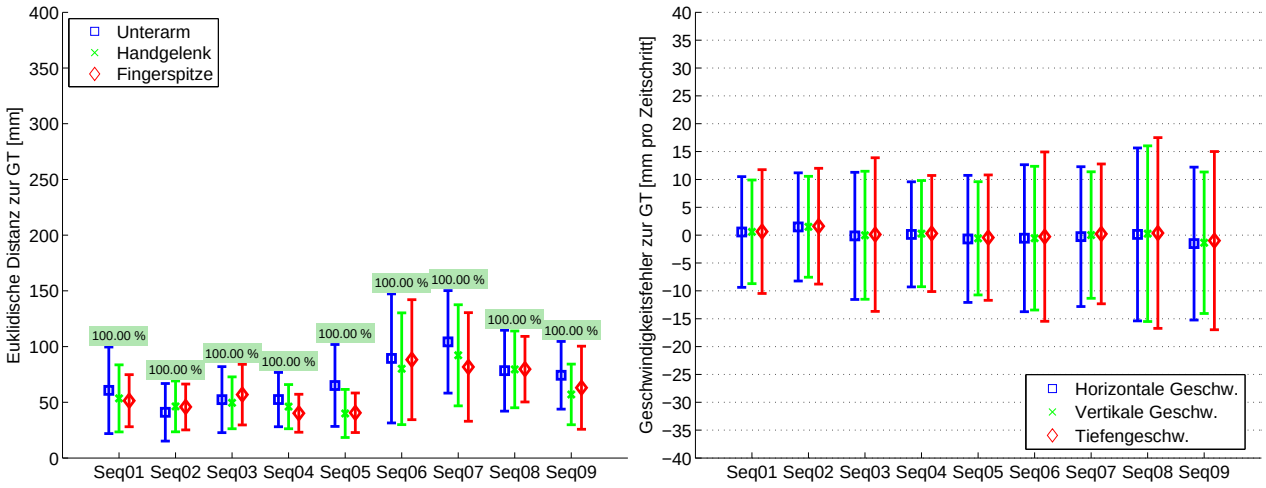

Abbildung 7.9: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 4**. Das Shape-Flow-Tracking mit zusätzlicher Verwendung des **Reinitialisierungsmoduls** ermöglicht es, dass alle Sequenzen über die komplette Zeit getrackt werden. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde durch den Shape-Flow-Algorithmus direkt berechnet.

systematischen Abweichungen. Für alle Sequenzen sind die Standardabweichungen der Geschwindigkeitsfehler ähnlich und liegen in einem Bereich von 10 bis 15 mm pro Zeitschritt. Für ein Kamerasystem mit kleiner Basisbreite ist das ein sehr gutes Ergebnis. Es zeigen sich auch keine signifikanten Unterschiede zwischen den lateralen Geschwindigkeitskomponenten und der Tiefengeschwindigkeit. Durch den Shape-Flow-Algorithmus ist man also in der Lage, die komplette zeitliche Ableitung der 3D-Pose direkt aus den Bildsequenzen in einer guten Genauigkeit zu schätzen. Betrachtet man die effektive Pixelauflösung bei einer Entfernung von 3 m, so schätzt der Shape-Flow-Algorithmus in System 4 die Geschwindigkeit des Unterarms auf 2–3 Pixel pro Zeitschritt genau, was bei dem einfachen 3D-Konturmodell bemerkenswert ist. Abbildung 7.10 zeigt die einzelnen Geschwindigkeitskomponenten für das Ende von Sequenz 5. Es ist zu sehen, dass es keine systematischen Abweichungen bei den Schätzungen gibt, sondern nur zufällige Messfehler. Auch bei einer maximalen Tiefengeschwindigkeit von 40 mm pro Zeitschritt ist der Shape-Flow-Algorithmus in der Lage, die zeitliche 3D-Pose-Ableitung robust zu schätzen.

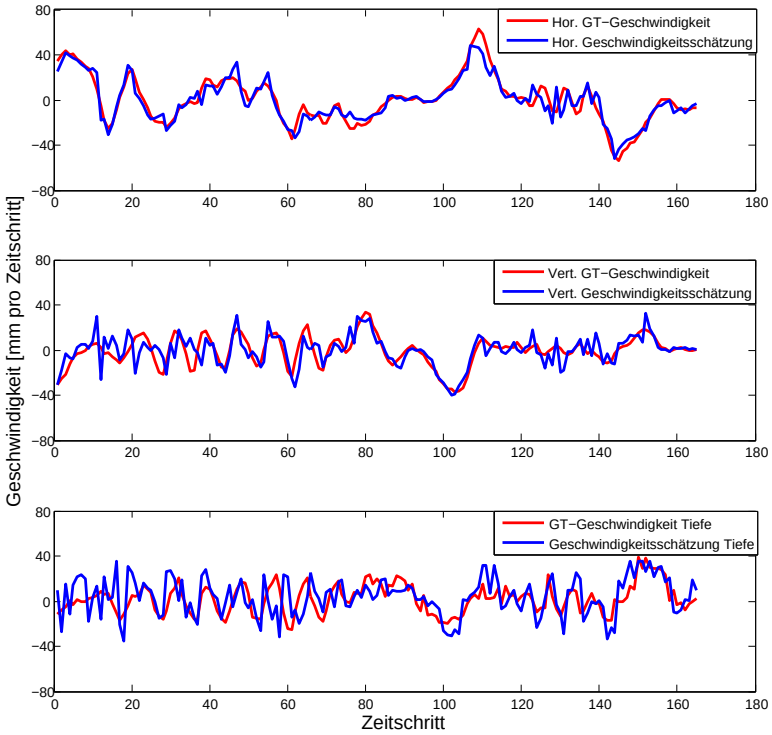

Abbildung 7.10: Die instantan geschätzten Geschwindigkeitskomponenten (blau) des Systems 4 für das Ende von Sequenz 5 im Vergleich zur Ground-Truth (GT) Geschwindigkeit (rot). Die Maximalgeschwindigkeit für die horizontale Komponente beträgt 65 mm pro Zeitschritt, für die vertikale Komponente 35 mm pro Zeitschritt und für die Tiefengeschwindigkeit 40 mm pro Zeitschritt. Dies zeigt, dass die ausgeführte Bewegung nicht langsam ist. Das Trackingsystem 4 ist trotzdem in der Lage, die Geschwindigkeiten robust zu schätzen.

Die mittlere Laufzeit der Matlab-Implementierung von System 4 auf Basis aller Testsequenzen beträgt 27 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Der Laufzeitunterschied zum System 3 wird durch das Reinitialisierungsmodul hervorgerufen. Falls das Reinitialisierungsmodul angewandt wird, erhöht sich die Laufzeit des Algorithmus um das Dreifache. Im Mittel wird das Reinitialisierungsmodul alle 25 Zeitschritte angewandt.

Abbildung 7.11 zeigt die ins Bild der Masterkamera projizierten Trackingergebnisse des Systems 4 (Shape-Flow-Tracking mit Reinitialisierung) auf drei der verwendeten Testsequenzen. Zusätzlich ist eine 3D-Rekonstruktion des Modells mit zugehörigen Geschwindigkeitsvektoren dargestellt. Durch den Shape-Flow-Algorithmus kann ein dichter modellbasierter Szenenfluss, also das 3D-Bewegungsfeld der 3D-Punkte auf der Oberfläche des Modells, aus einer Bildsequenz robust geschätzt werden.

7.3.5 Ergebnisse System 5 – Mean-Shift-Tracking

Das in dieser Arbeit entwickelte Verfahren zum 3D-Mean-Shift-Tracking ist in der Lage, alle sich in der Szene befindlichen bewegten Objekte zu verfolgen. Zur Evaluierung der metrischen Genauigkeit des entwickelten Systems wird aber nur die rechte Hand der Testperson getrackt,

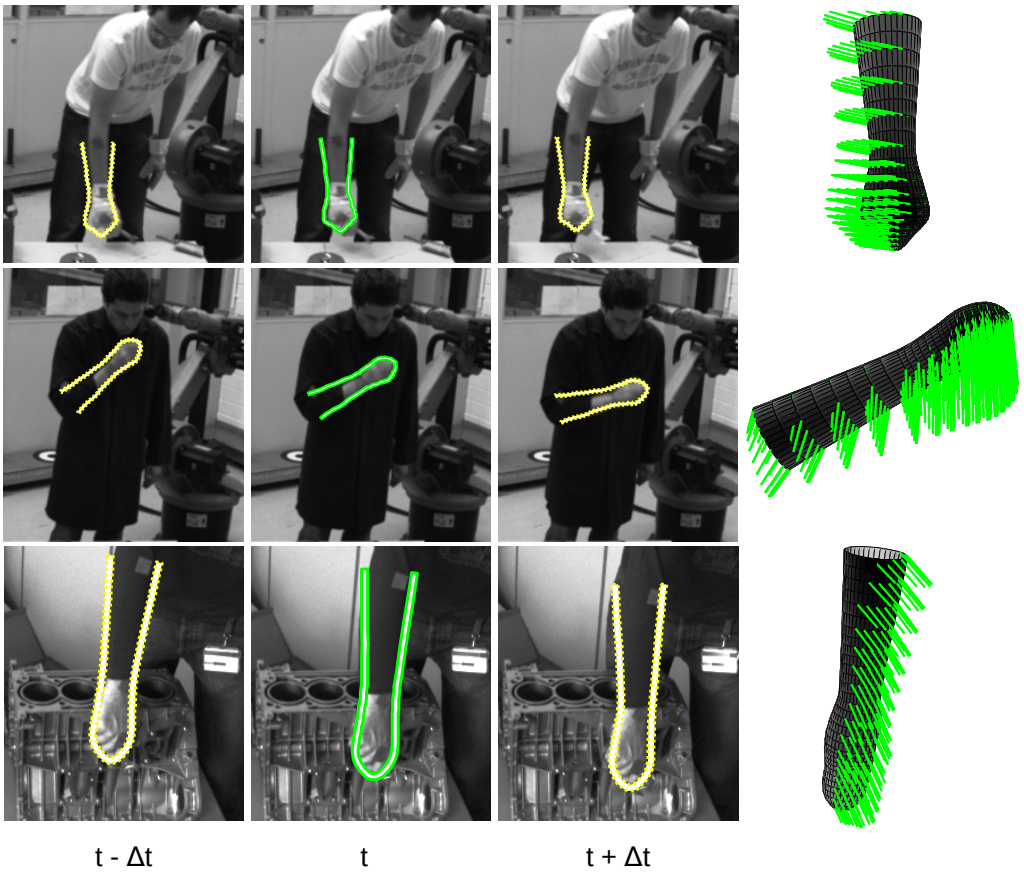

Abbildung 7.11: Ergebnisse der raum-zeitlichen Verfolgung des Hand-Unterarm-Bereichs auf drei der Testsequenzen. Die ersten drei Spalten zeigen das ins Bild projizierte 3D-Konturmodell und die vierte Spalte zeigt eine 3D-Rekonstruktion des Modells mit zugehörigen Geschwindigkeitsvektoren. Durch den Shape-Flow-Algorithmus ist man in der Lage, einen dichten modellbasierten Szenenfluss aus einer Bildsequenz zu berechnen.

da nur für diesen Bereich eine detaillierte Ground-Truth vorhanden ist. System 5 bietet zwei Systemvarianten, bei der Ersten basiert die Berechnung des Zielmodells auf Grauwertbildern und bei der Zweiten auf dem Farbwinkel (Hue-Komponente) des HSV-Farbraums.

Die Ergebnisse der kamerabasierten Verfolgung der menschlichen Hand im Vergleich zur Ground-Truth für System 5 (Mean-Shift-Tracking) sind in Abbildung 7.12 dargestellt. Als Eingabedaten wurden Grauwertbilder verwendet. Bei den Versuchen wurde ein Ellipsoid zum ersten Zeitschritt der Sequenz manuell auf der Hand initialisiert und anschließend mit dem Mean-Shift-Tracking in den Kamerabildern getrackt. Die manuelle Initialisierung gewährleistet den selben Startpunkt für die beiden verschiedenen Varianten des Systems 5. Falls das Tracking abreißt, wird der Ellipsoid automatisch neu initialisiert. Dazu wird der in der beobachteten Szene detektierte Ellipsoid verwendet, welcher sich in nächster Nähe zur aktuellen Objektposition befindet. Wie bei den Ergebnissen der Systeme 2 und 4 wird das getrackte Objekt, in diesem Fall nur die Hand, in allen Testsequenzen vollständig über die Zeit verfolgt (Abbildung 7.12 (links)). Ein signifikanter Unterschied zwischen den Sequenzen 1 bis 5 und 6 bis 9 ist im Vergleich zu den Ergebnissen von System 2 nicht zu erkennen. Die mittleren

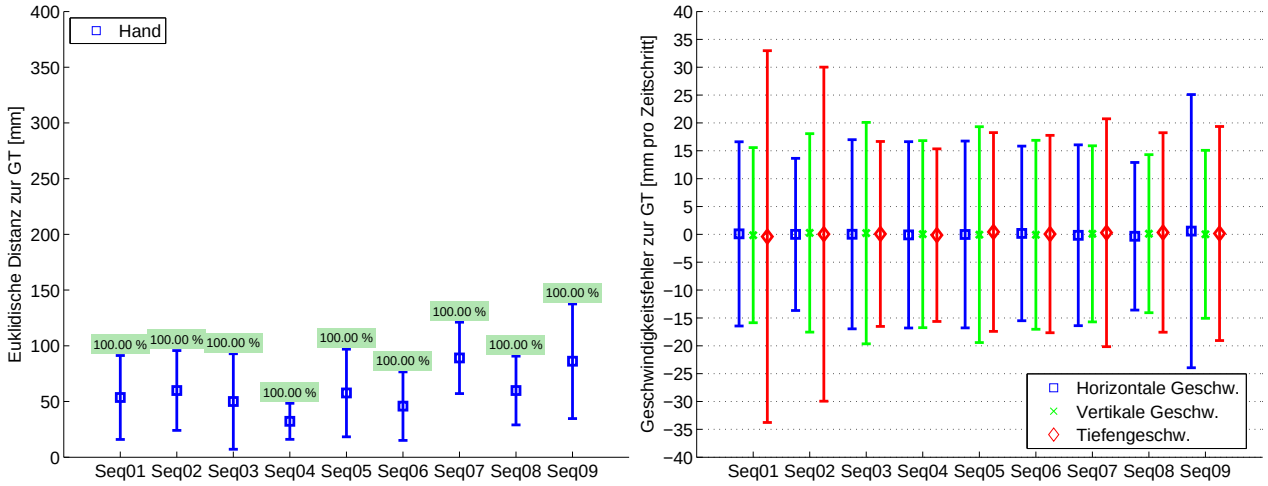

Abbildung 7.12: System 5 unter Nutzung von Grauwertbildern und einem Referenzmodell, welches auf gemittelten Intensitäten über alle drei Bilder basiert. Links: Positionsfehler der Schätzung zur Ground-Truth (GT). Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth (GT). Die Geschwindigkeit wurde aus den Differenzen unmittelbar aufeinander folgender Pose-Schätzungen ermittelt (ohne Filterung).

Positionsfehler aller Sequenzen liegen zwischen 45 und 90 mm, bei Standardabweichungen der Positionsfehler von 16 bis 50 mm. Die erzielten Ergebnisse sind sehr ähnlich zu denen von System 4, mit dem Unterschied, dass das Mean-Shift-Tracking weniger rechenaufwendig ist, dafür wird das getrackte Objekt auch nicht so detailliert modelliert. Außerdem kann es sein, dass beim Mean-Shift-Tracking nicht bemerkt wird, wenn die Hand verloren wurde. Durch das Tracking selbst, kann keine Aussage über die getrackten Objekte gemacht werden, da alle bewegten Objekte in der beobachteten Szene getrackt werden. Beim Mean-Shift-Tracking muss auf einer höheren Stufe des Gesamtsystems, z.B. durch eine Aktionserkennung (Abschnitt 8), aus den Bewegungsdaten die Zuordnung zum Objekt, Aktionen oder Handlungen abgeleitet werden. Die hohe Robustheit des Mean-Shift-Trackings ergibt sich aus der kombinierten Pose-Estimation mit Bilddaten und 3D-Punktewolken, außerdem kann durch das Detektionsmodul eine schlechte Schätzung verbessert werden. Es ist aber möglich, dass es sich um ein anderes Objekt handelt, z.B. ein Wechsel von der Hand auf den Oberarm, denn beide können dasselbe Aussehen, also die selben Farb- oder Grauwerte, haben. In den Experimenten auf Basis der neun Testsequenzen hat sich dies aber nicht gezeigt.

Die einzelnen Komponenten der Geschwindigkeitsfehler (Abbildung 7.12 (rechts)) zeigen keine systematischen Abweichungen. Für die Sequenzen 1 und 2 ist die Standardabweichung des Fehlers der Tiefengeschwindigkeit stark erhöht und liegt über 30 mm pro Zeitschritt. Die Begründung hierfür liegt in den 3D-Punktewolken, da es zu großen Tiefensprüngen kommen kann, wenn 3D-Punkte nur von Zeit zu Zeit auf der Hand zu finden sind. Der verwendete Stereo-Algorithmus hat Probleme, dauerhaft auf der Hand Korrespondenzen zu finden. Die Standardabweichungen der Geschwindigkeitsfehler der restlichen Sequenzen liegen im Bereich von 15 bis 25 mm pro Zeitschritt. Im Vergleich zu den Ergebnissen von System 4 ist das mindestens eine Erhöhung um das Anderthalbfache. Dies zeigt, dass eine direkte Bewegungsschätzung eine Verbesserung der Geschwindigkeitsschätzung für das System 5 bringen könnte.

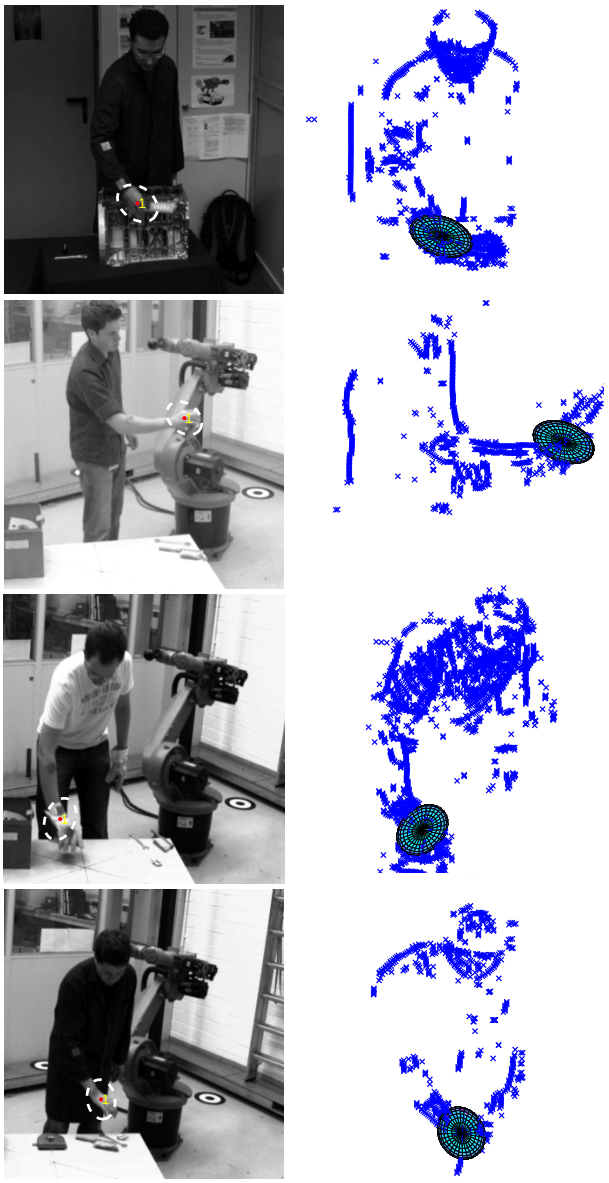

Abbildung 7.13: Ergebnisse des Systems 5 (Mean-Shift-Tracking) auf vier der verwendeten Testsequenzen. Links: Ins Bild der Masterkamera projiziertes Trackingergebnis. Rechts: Alle sich bewegenden 3D-Punkte der Szene und das 3D-Trackingergebnis.

Abbildung 7.13 zeigt die Ergebnisse des Systems 5 (Mean-Shift-Tracking) auf vier der verwendeten Testsequenzen. Es ist der ins Bild der Masterkamera projizierte Ellipsoid dargestellt. Zusätzlich werden alle bewegten 3D-Punkte der Szene gezeigt und der durch den Mean-Shift-Algorithmus in die Punktewolke gefittete 3D-Ellipsoid.

Die mittlere Laufzeit der Matlab-Implementierung von System 5 (Grauwerte im Referenzmodell) auf Basis aller Testsequenzen beträgt für die Verfolgung von einem Ellipsoid 0.26 Sekunden pro Bildtripel auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Das Tracking aller bewegter Objekte in der beobachteten Szene benötigt 1.3 Sekunden pro Bildtripel bei im Mittel 6.3 getrackten Objekten über alle neun Testsequenzen. Bei der Lauf-

zeitabschätzung wurden die Szenenflussdaten nicht Online berechnet, sondern zuvor ermittelt, gespeichert und während des Testlaufs von der Festplatte geladen. Der Laufzeitunterschied zu den vorherigen Systemen ergibt sich durch die Einfachheit des vorgestellten Algorithmus.

Abbildung 7.14 zeigt die Ergebnisse der kamerabasierten Verfolgung der menschlichen Hand im Vergleich zur Ground-Truth für System 5 unter Verwendung der Hue-Komponente des HSV-Farbraums. Wiederum wurde der Ellipsoid zum ersten Zeitschritt der Sequenz manuell auf der Hand initialisiert. Wie bei den Ergebnissen des Mean-Shift-Trackings mit Grauwertbildern als Eingabedaten wird die Hand, in allen neun Testsequenzen vollständig über die Zeit getrackt (Abbildung 7.14 (links)). Ein signifikanter Unterschied zwischen den Sequenzen 1 bis 5 und 6 bis 9 ist nicht zu erkennen. Die mittleren Positionsfehler aller Sequenzen liegen zwischen 25 und 55 mm. Damit erzielt diese Variante die beste mittlere Positionsgenauigkeit aller Sequenzen. Die Standardabweichungen der Positionsfehler liegen im Bereich von 15 bis 40 mm, auch dies ist im Mittel das beste Ergebnis. Der Grund für die hohe Genauigkeit und zeitliche Robustheit liegt in der verwendeten Hue-Komponente (Farbwinkel) des HSV-Farbraums zur Erzeugung des Referenzmodells. Hautfarbe oder die Farbe eines Handschuhs lässt sich mit dem Farbwinkel sehr gut approximieren. Daher ist das eindeutige Wiederfinden eines getrackten Objekts sehr gut möglich und 3D-Punktewolken sorgen für eine hohe Genauigkeit.

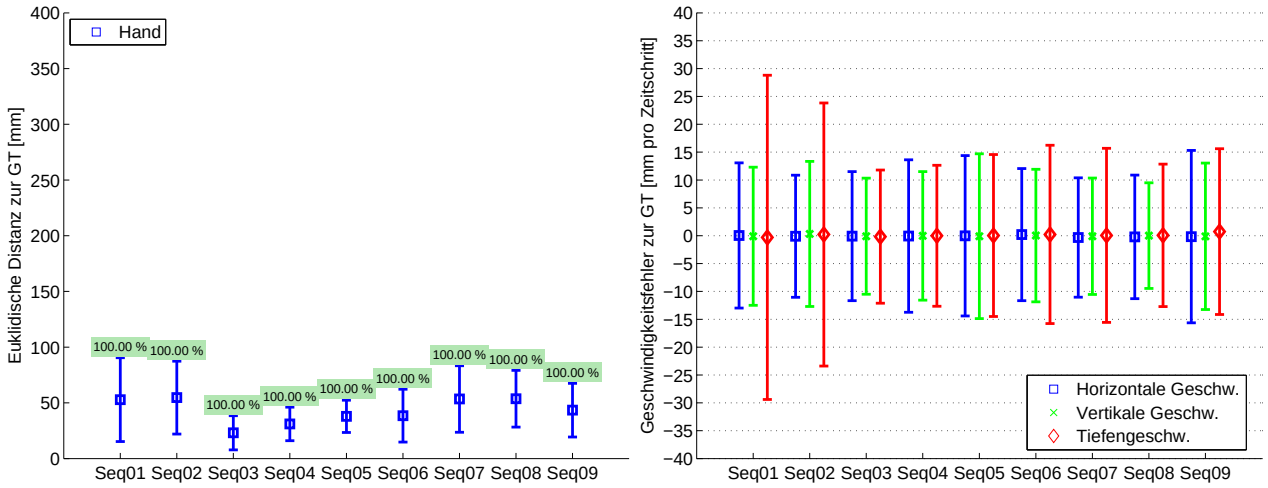

Abbildung 7.14: System 5 unter Nutzung von Farbbildern und einem Referenzmodell beim Tracking, welches auf dem Farbwinkel (Hue-Komponente) des HSV-Farbraums basiert. Links: Positionsfehler der Schätzung zur Ground-Truth (GT). Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth (GT). Die Geschwindigkeit wurde aus den Differenzen unmittelbar aufeinander folgender Pose-Schätzungen ermittelt (ohne Filterung).

Die einzelnen Komponenten der Geschwindigkeitsfehler (Abbildung 7.14 (rechts)) zeigen keine systematischen Abweichungen. Wiederum ist für die Sequenzen 1 und 2 die Standardabweichung des Fehlers der Tiefengeschwindigkeit erhöht und liegt zwischen 25 und 30 mm pro Zeitschritt. Die Begründung hierfür ergibt sich wiederum aus dem verwendeten Stereo-Algorithmus, denn dieser hat Probleme dauerhaft auf der Hand Korrespondenzen zu finden. Dies kann zu großen Tiefensprüngen führen. Die Standardabweichungen der Geschwindigkeitsfehler der restlichen Sequenzen liegen im Bereich von 12 bis 17 mm pro Zeitschritt. Im Vergleich

zu den Ergebnissen von System 5 unter Verwendung von Grauwertdaten ist dies eine leichte Verbesserung.

Die mittlere Laufzeit der Matlab-Implementierung von System 5 (Hue-Daten im Referenzmodell) auf Basis aller Testsequenzen beträgt für das Tracking von einem Ellipsoiden 0.34 Sekunden pro Bildtripel auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Das Tracking aller bewegter Objekte in der beobachteten Szenen benötigt 1.7 Sekunden pro Bildtripel bei im Mittel 5.5 getrackten Objekten über alle neun Testsequenzen. Der Unterschied zu System 5 mit Grauwerten im Referenzmodell ist durch den erhöhten Aufwand bei der Farbraumkonvertierung von RGB zu HSV-Daten zu begründen. Dass im Mittel über alle Sequenzen nur 5.5 Objekte getrackt werden liegt an der stärkeren Trennschärfe der Hue-Komponente im Gegensatz zu Grauwerten. Dies bedeutet, dass bei der Verwendung der Hue-Komponente im Referenzmodell verloren gegangene oder nicht mehr sichtbare Objekte schneller gelöscht werden, da der Vergleich mit dem Referenzmodell (Bhattacharyya-Distanz, Abschnitt 5.8) eindeutiger ist.

7.3.6 Ergebnisse System 6 – ICP-MOCCD-Tracking

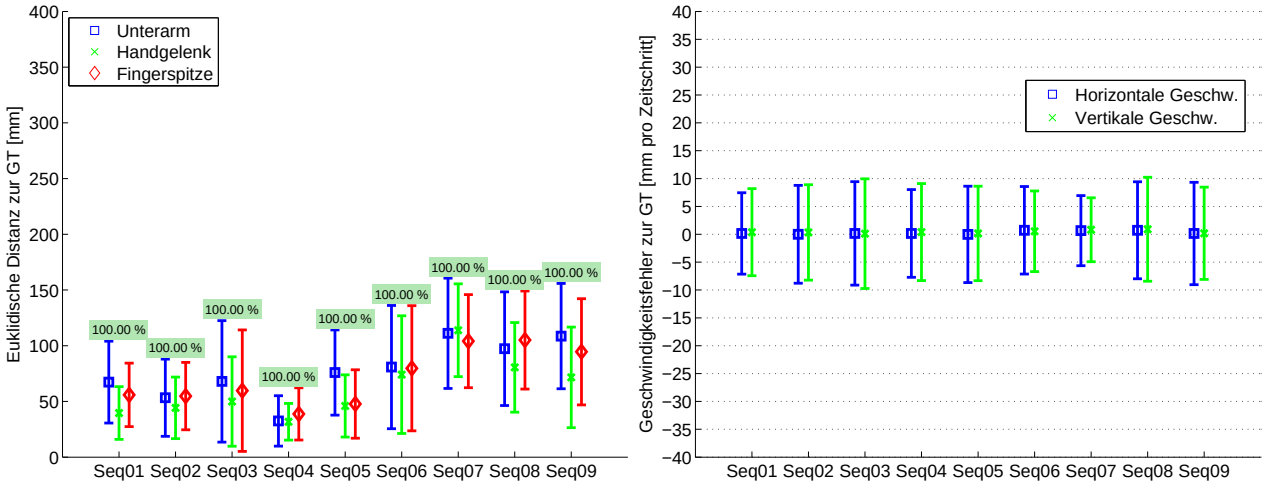

Abbildung 7.15: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 6**. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern direkt berechnet.

Abbildung 7.15 zeigt die Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth für System 6 (ICP-MOCCD-Tracking). Die Evaluation wurde in Zusammenarbeit mit Björn Barrois durchgeführt. Als Eingabedaten wurden Grauwertbilder verwendet. In allen Testsequenzen wird der Unterarm vollständig über die Zeit getrackt (Abbildung 7.15 (links)). Es ist wiederum ein leichter Unterschied zwischen den Sequenzen erkennbar. In den Sequenzen 1 bis 5 beträgt der mittlere Positionsfehler zwischen 35 und 60 mm, bei einer mittleren Standardabweichung von 32 mm. Die Positionsfehler der Sequenzen 6 bis 9 sind etwas erhöht und liegen in einem Bereich von 80 bis 115 mm. Die

mittlere Standardabweichung der Positionsfehler für die Sequenzen 6 bis 9 beträgt 50 mm und ist damit um das Anderthalbfache größer als für die Sequenzen 1 bis 5. Der Grund für das unterschiedliche Verhalten auf den Sequenzen liegt in den unterschiedlichen Aufnahmebedingungen, wodurch die Pose-Estimation mit dem ICP- und MOCCD-Algorithmus auf den Sequenzen 6 bis 9 erschwert wird. Wichtig ist die Tatsache, dass durch Kombination der beiden Pose-Estimation-Algorithmen die zeitliche Robustheit des Trackingsystems gesteigert wird, da die Fehler einzelner Algorithmen durch das Gesamtsystem ausgeglichen werden.

Die einzelnen Komponenten der lateralen Geschwindigkeitsfehler (Abbildung 7.15 (rechts)) zeigen keine systematischen Abweichungen. Durch System 6 können nur die lateralen Geschwindigkeitskomponenten direkt geschätzt werden. Für alle neun Sequenzen sind die Standardabweichungen der lateralen Geschwindigkeitsfehler ähnlich und liegen in einem Bereich von 6 bis 10 mm pro Zeitschritt. Betrachtet man die effektive Pixelauflösung bei einer Entfernung von 3 m so schätzt System 6 die Geschwindigkeit des Unterarms auf im Mittel weniger als 2 Pixel pro Zeitschritt genau. Durch die zusätzliche Verwendung des Shape-Flow-Algorithmus kann auch die Tiefengeschwindigkeit des getrackten Objekts aus einer Bildfolge direkt geschätzt werden.

Die mittlere Laufzeit der Matlab-Implementierung von System 6 auf Basis aller Testsequenzen beträgt 20 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Ein Großteil der Laufzeit wird dabei durch die verschiedenen Iterationsschritte des MOCCD-Algorithmus in der 3D-Pose-Estimation verbraucht. Wiederum wurden die Szenenflussdaten nicht Online berechnet, sondern von der Festplatte geladen.

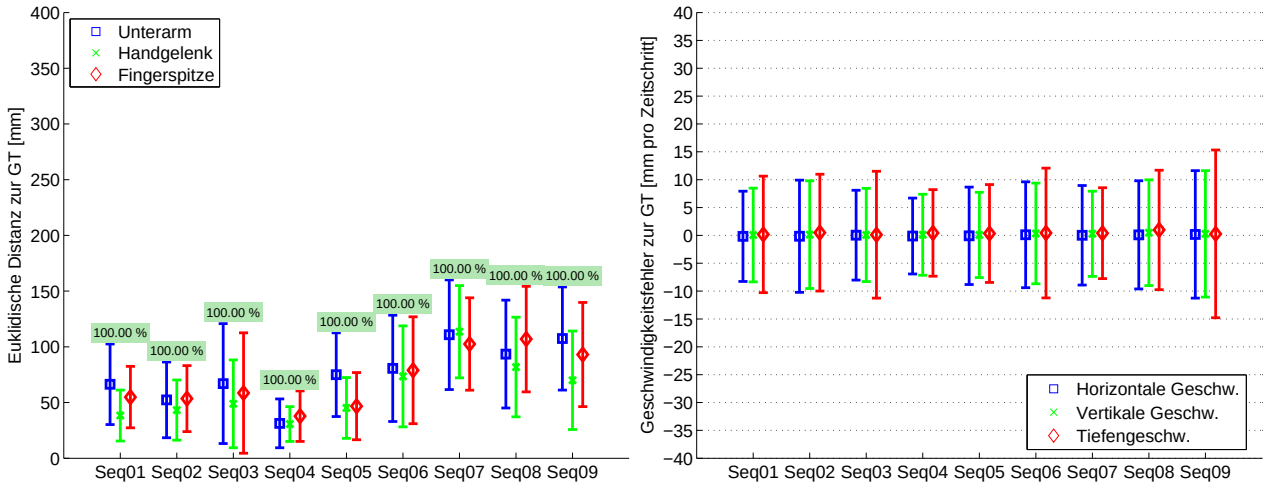

Abbildung 7.16: Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth (GT) für **System 7**. Die Eingabedaten sind **Grauwertbilder**. Links: Positionsfehler der Schätzung zur Ground-Truth. Rechts: Instantaner Geschwindigkeitsfehler für jede geschätzte Koordinate im Vergleich zur Ground-Truth. Die Geschwindigkeit wurde durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern und den Shape-Flow-Algorithmus direkt berechnet.

7.3.7 Ergebnisse System 7 – ICP-MOCCD-Shape-Flow-Tracking

Abbildung 7.16 zeigt die Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs im Vergleich zur Ground-Truth für System 7 (ICP-MOCCD-Shape-Flow-Tracking). Als Eingabedaten wurden Grauwertbilder verwendet. In allen Testsequenzen wird der Unterarm vollständig über die Zeit getrackt (Abbildung 7.15 (links)) und die Positionsgenauigkeit ist sehr ähnlich zu System 6. Der Vorteil von System 7 liegt in der zusätzlichen Verwendung des Shape-Flow-Algorithmus. Dadurch wird es möglich, die Tiefengeschwindigkeit eines getrackten Objekts aus einer Bildfolge direkt zu schätzen.

Die einzelnen Komponenten der Geschwindigkeitsfehler (Abbildung 7.16 (rechts)) zeigen keine systematischen Abweichungen. Für alle neun Sequenzen sind die Standardabweichungen der Geschwindigkeitsfehler ähnlich und liegen im Mittel über alle Sequenzen bei weniger als 10 mm pro Zeitschritt. Der Vorteil gegenüber dem System 6 liegt in der direkten Schätzung der Tiefengeschwindigkeit. Die Standardabweichungen der Tiefengeschwindigkeitsfehler liegen im Bereich von 8 bis 15 mm pro Zeitschritt und sind damit kaum höher als die Standardabweichungen der lateralen Geschwindigkeitsfehler. Für ein Kamerasystem mit kleiner Basisbreite ist das ein auffallend gutes Ergebnis.

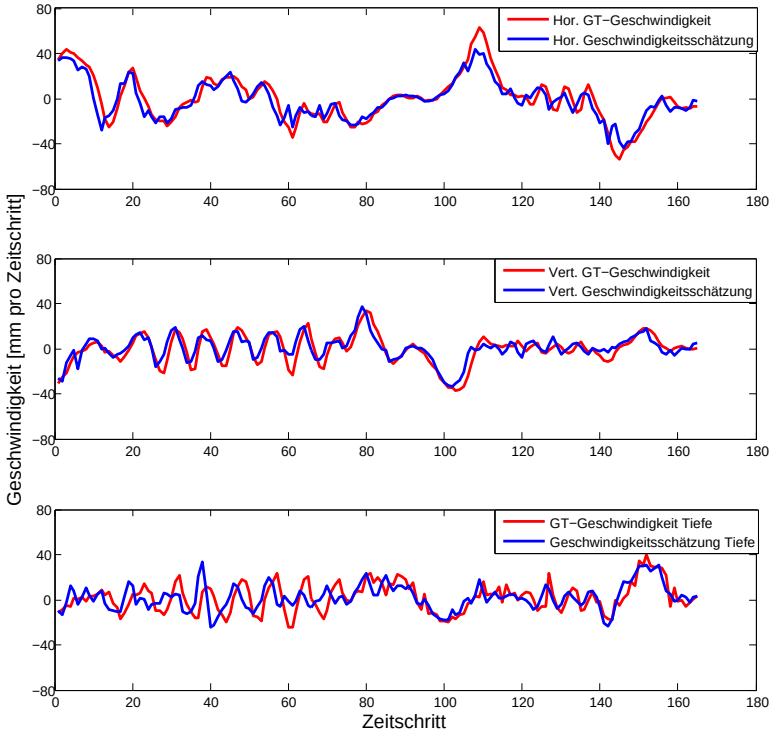

Abbildung 7.17: Die instantan geschätzten Geschwindigkeitskomponenten (blau) des Systems 7 für das Ende von Sequenz 5 im Vergleich zur Ground-Truth (GT) Geschwindigkeit (rot). Die Maximalgeschwindigkeit für die horizontale Komponente beträgt 65 mm pro Zeitschritt, für die vertikale Komponente 35 mm pro Zeitschritt und für die Tiefengeschwindigkeit 40 mm pro Zeitschritt. Dies zeigt, dass die ausgeführte Bewegung nicht langsam ist.

Im Vergleich zu den Ergebnissen von System 4 ist eine leichte Verbesserung der Geschwindigkeitsschätzung zu erkennen, denn in diesem System liegen die Standardabweichungen der

Geschwindigkeitsfehler im Mittel über alle Sequenzen bei 12.5 mm pro Zeitschritt. Die Verbesserung ist damit zu erklären, dass der Shape-Flow-Algorithmus besser initialisiert wird. Die Initialisierung ergibt sich durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern, deren Ergebnisse der Geschwindigkeitsschätzung sehr genau sind (Abschnitt 7.3.6, Abbildung 7.15 (rechts)). Im Vergleich zu allen bisherigen Systemen erzielt System 7 die besten Ergebnisse bei der direkten Schätzung der 3D-Pose-Ableitung.

Abbildung 7.17 zeigt die einzelnen Geschwindigkeitskomponenten für das Ende von Sequenz 5, es ist zu sehen, dass es keine systematischen Abweichungen bei der Schätzung gibt. Auch bei einer maximalen Tiefengeschwindigkeit von 40 mm pro Zeitschritt ist der Shape-Flow-Algorithmus in der Lage, eine direkte zeitliche 3D-Pose-Ableitung robust zu schätzen.

Die mittlere Laufzeit der Matlab-Implementierung von System 7 auf Basis aller Testsequenzen beträgt 25 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Die Szenenflussdaten wurden bei der Laufzeitabschätzung nicht Online berechnet, sondern von der Festplatte geladen.

Abbildung 7.18 zeigt typische Pose-Estimation-Ergebnisse des Systems 7 auf verschiedenen Testsequenzen. Zusätzlich ist eine 3D-Rekonstruktion des Modells mit zugehörigen Geschwindigkeitsvektoren dargestellt. Der Ansatz des Shape-Flow-Algorithmus ermöglicht es, einen dichten modellbasierten Szenenfluss, also das 3D-Bewegungsfeld der 3D-Punkte auf der Oberfläche des Modells, aus einer Bildsequenz robust zu schätzen. Durch die genaue Initialisierung des Shape-Flow-Algorithmus durch die Bewegungsschätzung auf Basis von Verschiebungsvektorfeldern, wird die Geschwindigkeitsschätzung noch besser.

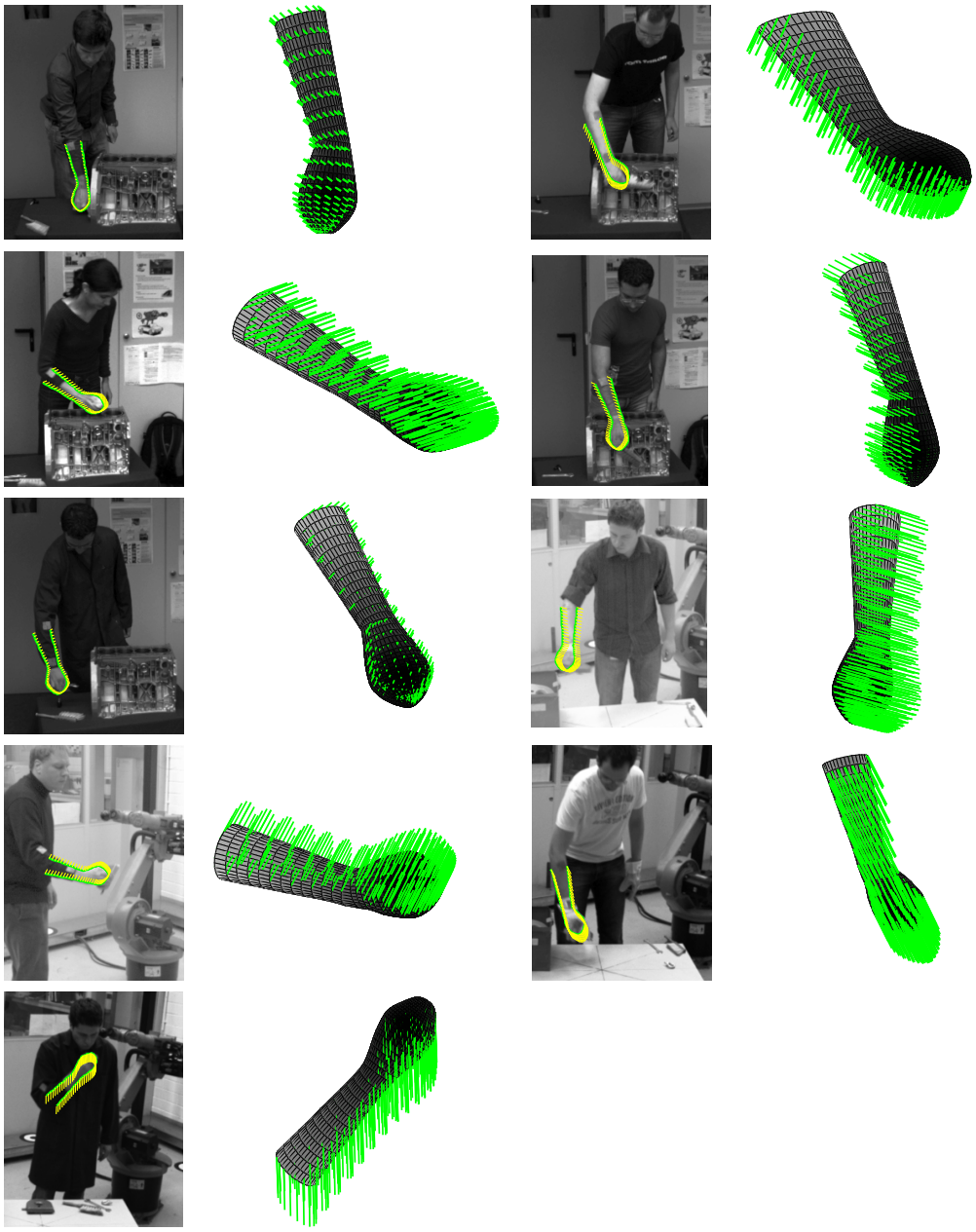

Abbildung 7.18: Typische Ergebnisse der raum-zeitlichen Verfolgung des Hand-Unterarm-Bereichs mit System 7.

7.4 Tracking der Kopf-Schulter-Partie

In diesem Abschnitt werden die Ergebnisse verschiedener Experimente zum Tracking der menschlichen Kopf-Schulter-Partie mit ausgewählten Systemvarianten (Abschnitt 7.2) auf Basis von drei Testsequenzen ausgewertet. Die Sequenzen wurden mit dem in Abschnitt 3.1 beschriebenen Kamerasystem aufgezeichnet. Die Sequenzen zeigen drei verschiedene Testpersonen, welche komplexe Bewegungen vor strukturierten Hintergründen ausführen. Jede der

drei Sequenzen enthält mindestens 200 Bildtripel. Die Distanz der Testpersonen zur Kamera variiert zwischen 1.6 und 4 m.

Da die Ergebnisse der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs sehr ausführlich beschrieben wurden, werden in diesem Abschnitt nur die Ergebnisse ausgewählter Systemvarianten vorgestellt. Wichtig ist es, zu zeigen, dass die in dieser Arbeit entwickelten Trackingsysteme in der Lage sind, beliebige menschliche Körperteile kamerabasiert zu verfolgen. Voraussetzung ist dabei, dass das getrackte menschliche Körperteil durch ein 3D-Konturmodell beschrieben werden kann.

Als Ergebnis eines Testlaufs mit einer der Systemvarianten (Abschnitt 7.2) auf einer Testsequenz ergibt sich für jeden Zeitschritt der Sequenz die euklidische Distanz zwischen dem durch das Trackingsystem geschätzten Zentrum des Halses im 3D-Modell (Abschnitt 3.2.1) und dessen Ground-Truth (Abschnitt 7.1.2). Damit kann die mittlere euklidische Distanz und die Standardabweichung zur Ground-Truth berechnet werden. Weiterhin werden die Fehler der Schätzung der Orientierung von Schultern und Kopf berechnet.

Im Folgenden werden die Ergebnisse der kamerabasierten Verfolgung der menschlichen Kopf-Schulter-Partie vorgestellt und diskutiert. Die notwendigen konstanten Modellparameter, also die Länge von Hals l_{Hals} und Kopf l_{Kopf} werden zu $l_{Hals} = 70$ mm und $l_{Kopf} = 220$ mm gewählt und sind für alle Testpersonen fix. Die Schätzung der Deformation des Modells erfolgt durch den MOCCD-Algorithmus.

7.4.1 Ergebnisse System 1 – MOCCD-Tracking

Die Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth für System 1 (MOCCD-Tracking) auf Basis der drei Testsequenzen sind in Tabelle 7.1 aufgelistet. Als Eingabedaten wurden Grauwertbilder verwendet. In allen drei Sequenzen wird die Kopf-Schulter-Partie vollständig über die Zeit getrackt. Der mittlere Positionsfehler liegt im Bereich von 50 bis 90 mm und die Standardabweichung der Positionsfehler liegt im Bereich von 30 bis 75 mm. Im Vergleich zu den Ergebnissen der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs für System 1 (Abschnitt 7.3.1) sind das verbesserte Ergebnisse. Dies liegt am getrackten Objekt selbst, denn die menschliche Kopf-Schulter-Kontur ist ein stärkeres und stabileres Merkmal als die Kontur des Hand-Unterarm-Bereichs. Die Pose-Estimation mit dem MOCCD-Algorithmus wird also leichter, da sich die Kopf-Schulter-Kontur im Allgemeinen besser vom Hintergrund absetzt.

Um die Fehler der Orientierungsschätzung von Schultern und Kopf entlang der Blickrichtung zu berechnen, wird für jeden der Winkel der RMSE-Fehler bezüglich der Ground-Truth berechnet (Tabelle 7.1). Als RMSE-Fehler (engl. *root mean square error*) wird die Wurzel aus dem mittleren quadratischen Fehler bezeichnet. Der RMSE-Fehler für die Orientierungsschätzung der Schultern entlang der Blickrichtung liegt im Bereich von 3 bis 8 Grad. Die Schätzung der Schulterorientierung kann dabei durch die Kleidung der Testperson beeinflusst werden, denn das Modell kann sich z.B. auch an den Kragen der Person anstatt der eigentliche Schulterkontur anpassen. Für die Orientierung des Kopfes liegt der RMSE-Fehler in einem ähnlichen Bereich von 4 bis 9 Grad. Die Schätzung der Kopforientierung entlang der Blickrichtung wird teilweise durch den unterschiedlichen Haaransatz beeinflusst, denn der MOCCD-Algorithmus kann beispielsweise das Konturmodell an den Haaransatz und nicht an die eigentliche Kopf-

Tabelle 7.1: Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth (GT) für System 1 (MOCCD-Tracking).

	Seq-KS-01	Seq-KS-02	Seq-KS-03
Verfolgungsrate	100%	100%	100%
Distanz zur GT [mm]	74 ± 75	67 ± 73	47 ± 27
RMSE Schulter-Winkel [Grad]	8.13	3.36	3.83
RMSE Kopf-Winkel [Grad]	4.04	5.39	8.94

kontur anpassen.

Abbildung 7.19 stellt Ergebnisdiagramme für System 1 bei der kamerabasierten Verfolgung der Kopf-Schulter-Partie für zwei der Testsequenzen dar. Abbildung 7.19 (links) zeigt die horizontale Position und die Entfernung der Testperson für Sequenz Seq-KS-01. Der große Entfernungsunterschied innerhalb der Sequenz und ein leichtes horizontales Wanken bei der Bewegung auf die Kamera zu, sind gut zu erkennen. Weiterhin sind Ergebnisse im Vergleich zur Ground-Truth für die Schätzung der Kopforientierung entlang der Blickrichtung für Sequenz Seq-KS-02 abgetragen (Abbildung 7.19 (rechts)). Dass die maximalen Orientierungen der Ground-Truth innerhalb der Sequenz nicht erreicht werden können, liegt an dem verwendeten groben Modell. Außerdem kann es sein, dass der MOCCD-Algorithmus das Modell an den Haaransatz der Testperson anpasst (siehe Abbildung 7.20, 3. Zeile, 2. Spalte) und nicht an die eigentliche Kontur des Kopfes.

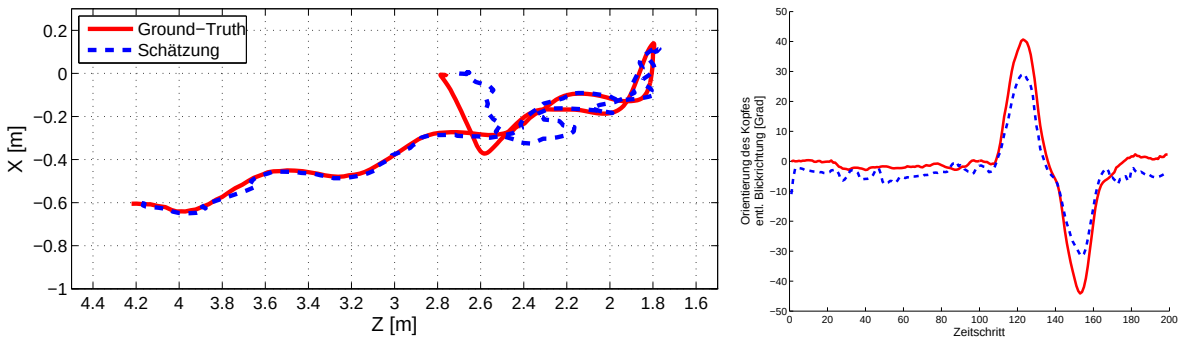

Abbildung 7.19: Links: Ergebnisse der Pose-Estimation im Vergleich zur Ground-Truth (GT) für Sequenz Seq-KS-01, dargestellt sind nur X (horizontale Position) und Z (Entfernung). Rechts: Schätzung der Kopforientierung entlang der Blickrichtung (blau, gestrichelt) im Vergleich zur Ground-Truth (rot, durchgezogen) für Sequenz Seq-KS-02.

Abbildung 7.20 zeigt die Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie auf einigen Beispielsequenzen. Es ist die Projektion der Modellkontur in das Bild der Masterkamera am Anfang, der Mitte und am Ende der jeweiligen Sequenz dargestellt. In der ersten Zeile sind sehr gut die großen Entfernungsunterschiede innerhalb der Sequenz Seq-KS-01 zu erkennen. Die Zeilen zwei und drei zeigen, dass die Orientierungsschätzung entlang der Blickrichtung gut möglich ist. Die vierte Zeile zeigt, dass auch Nickbewegungen in Richtung der Kamera gut behandelt werden können.

Abbildung 7.20: Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie auf einigen Beispielsequenzen. Die Spalten zeigen die Ergebnisse am Anfang, der Mitte und am Ende der jeweiligen Sequenz.

7.4.2 Ergebnisse System 3 – Shape-Flow-Tracking

In Tabelle 7.2 werden die Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth für System 3 (Shape-Flow-Tracking) auf Basis der drei Testsequenzen aufgelistet. Als Eingabedaten wurden wiederum Grauwertbilder verwendet. In allen drei Sequenzen wird die Kopf-Schulter-Partie vollständig über die Zeit getrackt. Der mittlere Positionsfehler liegt im Bereich von 50 bis 60 mm und die Standardabweichung der Positionsfehler liegt im Bereich von 30 bis 50 mm. Im Vergleich zu den Ergebnissen von System 1 sind das etwas verbesserte Ergebnisse. Die Verbesserung lässt sich durch die direkte Bewegungsschätzung begründen. Die Kalman-Filter haben Probleme bei den Umkehrbewegungen, zur Kamera hin und wieder zurück, die die Testpersonen ausführen. Die Fehler der Schätzung der Orientierung von Schultern und Kopf entlang der Blickrichtung sind ähnlich zu denen

Tabelle 7.2: Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth (GT) für System 3 (Shape-Flow-Tracking).

	Seq-KS-01	Seq-KS-02	Seq-KS-03
Verfolgungsrate	100%	100%	100%
Distanz zur GT [mm]	49 ± 27	59 ± 43	60 ± 51
Geschwindigkeitsfehler:			
Horizontal [mm/Zeitschritt]	$+0.09 \pm 4.05$	-0.27 ± 4.84	-1.06 ± 3.78
Vertikal [mm/Zeitschritt]	$+0.51 \pm 3.44$	-0.04 ± 5.18	$+0.87 \pm 4.31$
Tiefe [mm/Zeitschritt]	-0.42 ± 12.47	-2.29 ± 14.97	-1.06 ± 11.84
RMSE Schulter-Winkel [Grad]	6.39	3.57	3.52
RMSE Kopf-Winkel [Grad]	4.87	5.54	8.49

von System 1 und liegen im Bereich von 4 bis 9 Grad. Die Geschwindigkeitsfehler, also die Fehler der instantanen Bewegungsschätzung, sind auch in Tabelle 7.2 aufgezeigt. Die Mittelwerte der Geschwindigkeitsfehler der einzelnen Koordinaten sind nahe Null. Dadurch sind keine systematischen Fehler erkennbar. Es zeigen sich aber deutliche Unterschiede zwischen den lateralen Geschwindigkeitsmessungen und der Messung der Tiefengeschwindigkeit. Die Standardabweichung der lateralen Geschwindigkeitsfehler liegt im Mittel bei ungefähr 4.5 mm pro Zeitschritt, die mittlere Standardabweichung des Fehlers der Tiefengeschwindigkeit liegt bei 13 mm pro Zeitschritt. Dieser Unterschied ist durch das getrackte Objekt selbst und durch die ausgeführten Bewegungen zu erklären. Die menschliche Kopf-Schulter-Kontur ist ein starkes und zeitlich stabiles Merkmal, da sich dieser Bereich im Allgemeinen gut vom Hintergrund absetzt. Die Schätzung der lateralen Pose-Ableitung mit dem Shape-Flow-Algorithmus ist also sehr gut möglich. Die Schätzung der Tiefengeschwindigkeit bei einer Bewegung auf die Kamera zu wird durch das grobe Modell und die geringe Basisbreite begrenzt. Die hohe Genauigkeit der Schätzung der vertikalen Geschwindigkeit ist durch die Bewegungen der Testpersonen zu begründen, denn es gibt kaum vertikale Bewegungen in der Sequenz, die Testpersonen stehen also immer aufrecht. Die Schätzung der vertikalen Geschwindigkeit ist daher immer nahe Null.

Im Vergleich zu den Ergebnissen der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs für System 3 (Abschnitt 7.3.3) zeigen sich beim Tracking der Kopf-Schulter-Partie verbesserte Ergebnisse. Dies liegt wie bereits beschrieben am getrackten Objekt selbst. Die menschliche Kopf-Schulter-Kontur ist ein stärkeres Merkmal als die Hand-Unterarm-Kontur und lässt sich durch ein Konturmodell besser beschreiben. Deformationen des Kopfes sind im allgemeinen nicht möglich. Bei der Hand sind Deformationen viel wahrscheinlicher, z.B. durch ein „Greifen“ oder das „Spreizen der Finger“.

7.4.3 Ergebnisse System 4 – Shape-Flow-Tracking mit Reinitialisierung

Tabelle 7.3 zeigt die Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth für System 4 (Shape-Flow-Tracking mit Reinitialisierung). Als Eingabedaten wurden wiederum Grauwertbilder verwendet. Da die Anwendung der Trackingsysteme 1 und 3 auf Basis der drei Testsequenzen zu einer vollständigen Verfolgung der Kopf-

Tabelle 7.3: Ergebnisse der kamerabasierten Verfolgung der Kopf-Schulter-Partie im Vergleich zur Ground-Truth (GT) für System 4 (Shape-Flow-Tracking mit Reinitialisierung).

	Seq-KS-01	Seq-KS-02	Seq-KS-03
Verfolgungsrate	100%	100%	100%
Distanz zur GT [mm]	46 ± 22	52 ± 38	48 ± 28
Geschwindigkeitsfehler:			
Horizontal [mm/Zeitschritt]	$+0.06 \pm 4.05$	-0.43 ± 4.23	-1.01 ± 3.67
Vertikal [mm/Zeitschritt]	$+0.39 \pm 3.34$	-0.07 ± 4.09	$+0.80 \pm 4.15$
Tiefe [mm/Zeitschritt]	-0.69 ± 11.91	-1.65 ± 13.13	-1.10 ± 11.08
RMSE Schulter-Winkel [Grad]	6.35	3.33	3.30
RMSE Kopf-Winkel [Grad]	4.87	5.54	8.74

Schulter-Partie über die Zeit geführt hat, sind durch die Verwendung des Reinitialisierungsmoduls kaum Verbesserungen zu erwarten. Wie bereits zuvor, wird durch System 4 in allen drei Sequenzen die Kopf-Schulter-Partie vollständig über die Zeit getrackt. Der mittlere Positionsfehler ergibt sich zu 48 mm, was eine leichte Verbesserung zu den Ergebnissen der vorherigen Systeme ist. Deutlicher werden die Unterschiede bei der Standardabweichung der Positionsfehler. Hier liegt der Mittelwert der drei Sequenzen bei ungefähr 30 mm, was eine Verbesserung um das Anderthalbfache darstellt. Die Verbesserung lässt sich durch die direkte Bewegungsschätzung und die Korrektur von schlechten Schätzungen durch das Reinitialisierungsmodul begründen. Die Fehler der Schätzung der Orientierung von Schultern und Kopf entlang der Blickrichtung sind ähnlich zu denen von System 1 und System 3 und liegen im Bereich von 4 bis 9 Grad. Hier zeigt das Reinitialisierungsmodul keine Wirkung, denn die Orientierungsschätzung wird durch das grobe 3D-Konturmodell beeinflusst (Abbildung 7.21 (rechts)). Außerdem kann es immer wieder dazu kommen, dass der MOCCD-Algorithmus oder der Shape-Flow-Algorithmus das jeweilige Modell an den Haaransatz oder den Schatten auf dem Gesicht der Testperson anpasst.

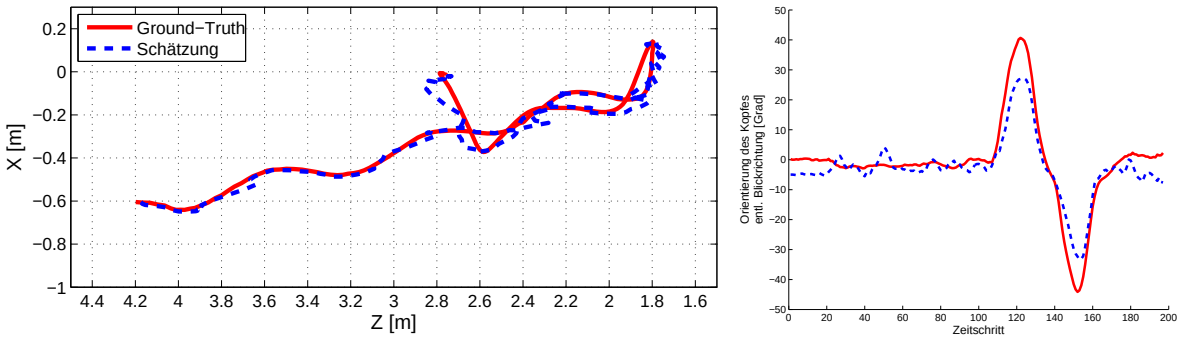

Abbildung 7.21: Links: Ergebnisse der Pose-Estimation im Vergleich zur Ground-Truth für Sequenz Seq-KS-01, dargestellt sind nur X (horizontale Position) und Z (Entfernung). Rechts: Schätzung der Kopforientierung (blau, gestrichelt) entlang der Blickrichtung im Vergleich zur Ground-Truth (rot, durchgezogen) für Sequenz Seq-KS-02.

Die Geschwindigkeitsfehler, also die Fehler der instantanen Bewegungsschätzung, sind auch in Tabelle 7.3 aufgezeigt. Die Mittelwerte der Geschwindigkeitsfehler der einzelnen Koordi-

naten sind nahe Null. Die Standardabweichung der lateralen Geschwindigkeitsfehler liegt im Mittel bei ungefähr 4 mm pro Zeitschritt, die mittlere Standardabweichung des Fehlers der Tiefengeschwindigkeit liegt bei 12 mm pro Zeitschritt. Es zeigen sich also kaum Unterschiede zu den Geschwindigkeitsfehlern von System 3. Dies liegt daran, dass das Reinitialisierungsmodul kaum Einfluss auf die Geschwindigkeitsschätzung nimmt.

Im Vergleich zu den Ergebnissen der kamerabasierten Verfolgung des Hand-Unterarm-Bereichs für System 4 (Abschnitt 7.3.4) zeigen sich wie bei System 3 auch für das Tracking der Kopf-Schulter-Partie verbesserte Ergebnisse. Dies liegt wie bereits beschrieben, an den Vorteilen der menschlichen Kopf-Schulter-Kontur im Bezug auf Modellierung und Pose-Estimation.

Abbildung 7.21 zeigt Ergebnisdiagramme für System 4 bei der kamerabasierten Verfolgung der Kopf-Schulter-Partie für zwei der Beispielsequenzen. Es ist die horizontale Position und die Entfernung der Testperson für Sequenz Seq-KS-01 abgetragen (Abbildung 7.21 (links)). Im Vergleich zum Ergebnisdiagramm für System 1 (Abbildung 7.19 (links)) ist ein verbessertes Tracking zu erkennen. Weiterhin sind Ergebnisse im Vergleich zur Ground-Truth für die Schätzung der Kopforientierung entlang der Blickrichtung für Sequenz Seq-KS-02 abgetragen (Abbildung 7.21 (rechts)). Wiederum werden die maximalen Orientierungen der Ground-Truth innerhalb der Sequenz nicht erreicht. Die Gründe hierfür wurden bereits genannt. Abbildung 7.22

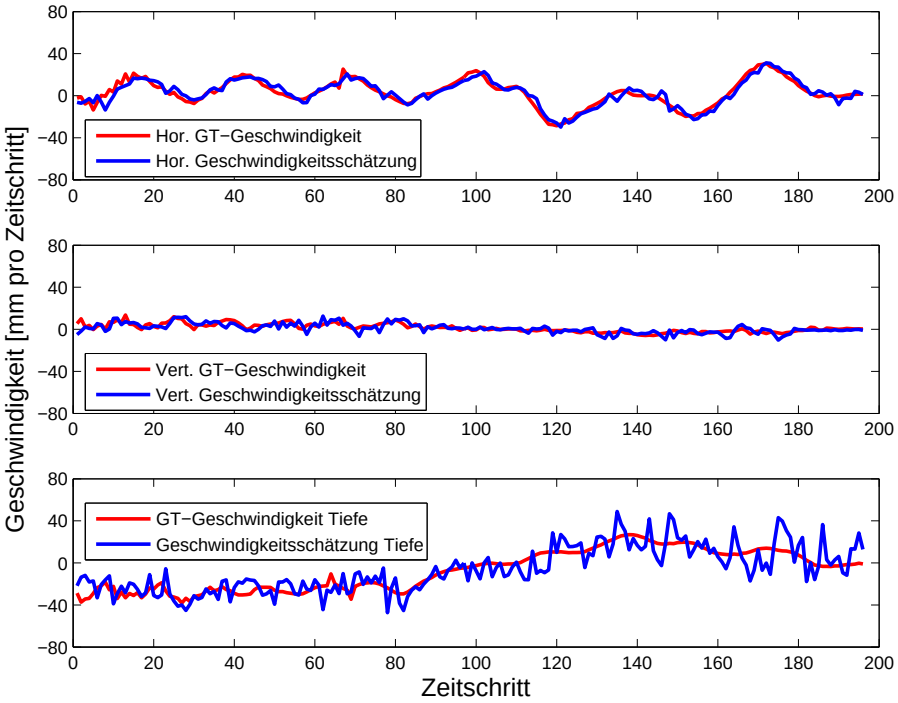

Abbildung 7.22: Die instantan geschätzten Geschwindigkeitskomponenten (blau) des Systems 4 für Sequenz Seq-KS-01 im Vergleich zur Ground-Truth (GT) Geschwindigkeit (rot).

zeigt die einzelnen Geschwindigkeitskomponenten für Sequenz Seq-KS-01. Es ist zu sehen, dass es keine systematischen Abweichungen bei der Schätzung gibt. Das leichte horizontale Wanken bei der Bewegung auf die Kamera zu ist auch in den Schätzungen der horizontalen Geschwindigkeit gut zu erkennen. Hier weicht die Schätzung nur geringfügig von der Ground-Truth ab. Es gibt kaum vertikale Bewegungen in der Sequenz, daher ist die Schätzung der vertikalen Ge-

schwindigkeit immer nahe Null. Auch bei absoluten Tiefengeschwindigkeiten von 30 bis 40 mm pro Zeitschritt ist der Shape-Flow-Algorithmus in der Lage, die zeitliche 3D-Pose-Ableitung des Konturmodells der Kopf-Schulter-Partie robust zu schätzen.

7.5 Weitere Trackingergebnisse

In diesem Abschnitt werden Ergebnisse vorgestellt, die nichts mit dem angestrebten Szenario dieser Arbeit zu tun haben, aber auf in dieser Arbeit entwickelten Ansätzen basieren. Ein aus dem 3D-Mean-Shift-Tracking (Kapitel 5) abgeleitetes Verfahren wurde durch Einhaus (2009) zum Tracking von Verkehrsteilnehmern in einem Kreisverkehrsszenario vorgestellt. Dabei wurde das 3D-Mean-Shift-Tracking mit der kognitiven Trajektorienprädiktion von Hermes u. a. (2009) kombiniert, um auch bei Überdeckungen von Fahrzeugen in den Kamerabildern ein stabiles und zeitlich robustes Trackingergebnis zu erzielen. In diesem Abschnitt wird nur auf die Ergebnisse des aus dem 3D-Mean-Shift-Tracking abgeleiteten Verfahrens eingegangen.

Das entwickelte Systemmodell ist in Abbildung 7.23 dargestellt. Es besteht aus sechs Komponenten: dem Kamerasystem (a), dem Szenenflussmodul (b), einem Modul zur Clusteranalyse (c), dem 3D-Pose-Estimation System (d), dem Modul (e) zur kognitiven Trajektorienprädiktion (Hermes u. a., 2009) und dem Prädiktionsmodul (f).

Die Eingabedaten für die Objektverfolgung sind rektifizierte Bilder eines Stereokamerasystems. Wie beim 3D-Mean-Shift-Tracking (Kapitel 5) wird ein spärlicher Szenenfluss berechnet. Dieser basiert anders als in Kapitel 5 auf einem merkmalsbasierten Verfahren (Stein, 2004) zur Berechnung von Stereodaten und optischem Fluss. Wie in Kapitel 5 wird zur Objektdetektion eine Clusteranalyse verwendet. Im entwickelten System basiert diese aber nur auf dem graphenbasierten Clustering (Abschnitt 5.3).

Ergebnisse der Clusteranalyse sind in Abbildung 7.23 im Modul (c) dargestellt. Die gefundenen Cluster werden als Quader approximiert und dienen als Detektionsergebnis für die Pose-Estimation (Abschnitte 5.5 und 5.6). Diese ist identisch zum Verfahren in Kapitel 5.

Die mit dem 3D-Mean-Shift-Ansatz geschätzten Positionen dienen als Eingabe für die kognitive Trajektorienprädiktion (Hermes u. a., 2009) zur langfristigen Bewegungsvorhersage. Diese ermöglicht es, auch bei Überdeckungen der Fahrzeuge in den Kamerabildern, ein stabiles und zeitlich robustes Trackingergebnis zu erzielen. Außerdem dienen die geschätzten Positionen des 3D-Mean-Shift-Ansatzes als Eingabewerte für die Standardprädiktion zu kurzfristigen Bewegungsvorhersage. Beide Prädiktionsvarianten wurden in das Trackingsystem durch Einhaus (2009) neu integriert. Auf die Ergebnisse der langfristigen Bewegungsvorhersage durch den Ansatz von Hermes u. a. (2009) wird in dieser Arbeit nicht eingegangen. Es werden nur die Trackingergebnisse des Systems ausgewertet. Die notwendige Prädiktion einer 3D-Pose wird im Trackingsystem durch die sogenannte Standardprädiktion (Abbildung 7.23, Modul (f)) übernommen. Dabei wird aus der 3D-Trajektorie des Objekts (die Menge der 3D-Positionen über die Zeit) eine Position für den Zeitschritt $t + \Delta t$ geschätzt. Dies geschieht durch eine Prädiktion mit konstanter Beschleunigung und konstantem Lenkwinkel über den Zeitraum Δt. Die Geschwindigkeit und Fahrtrichtung wird zunächst für jeden Zeitpunkt ermittelt und danach durch eine Regressionsgerade approximiert. Die Lenkwinkeländerung ergibt sich dann aus der Steigung der Geraden.

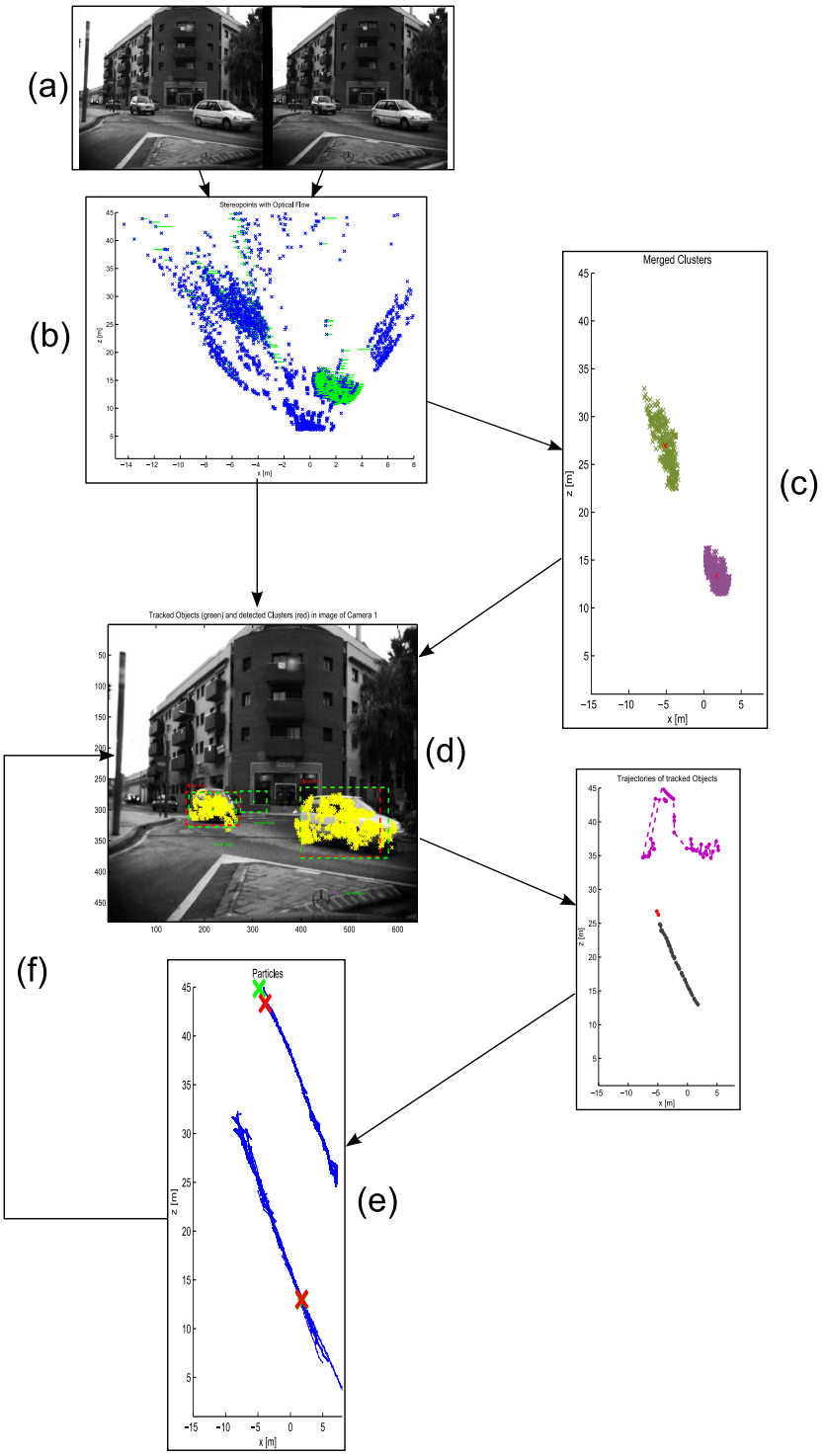

Abbildung 7.23: Systemmodell zum Tracking von Autos. Außerdem werden typische Ausgaben der Systemkomponenten gezeigt.

Nach der Prädiktion beginnt das Tracking mit dem nächsten Bildtupel des Stereokamerasystems. Neu detektierte Objekte werden zur Liste der getrackten Objekte hinzugefügt und

über die Zeit in den Bildern des Kamerasystems verfolgt.

7.5.1 Bewertung des Trackingverfahrens

Zum Zeitpunkt der Arbeit lagen leider keine Ground-Truth-Daten im Sinne von 3D-Objektpositionen vor, daher wurde das Tracking qualitativ anhand der Positionen in der Bildebene bewertet. Hierzu wurden die Testsequenzen manuell gelabelt, d.h. es wurden die Positionen und Ausmaße der zu trackenden Objekte im Bild durch Rechtecke festgelegt.

Die Bewertung der Sequenzen erfolgte durch den Vergleich der manuell gelabelten Objekte mit den Ergebnissen des Systems. Aufgrund der Annahme einer flachen Welt und der fehlenden wahren Tiefeninformation von Bildpunkten wurden Position und Ausmaß der Objekte nur entlang der x-Achse im Bild gemessen. Wie gut ein Objekt vom System erkannt wird, ergibt sich dann mit:

$$E(TrackedObject, LabelBox) = \begin{cases} 1 & \text{falls } \max\left\{\frac{overlap}{width(LabelBox)}, \frac{overlap}{width(TrackedObject)}\right\} > \epsilon \\ 0 & \text{sonst.} \end{cases} \tag{7.1}$$

Wobei *overlap* die Überschneidung des vom System erkannten Objekts mit dem Ground-Truth-Rechteck ist. Außerdem steht $width(LabelBox)$, bzw. $width(TrackedObject)$ für die Breite des Rechtecks der Ground-Truth, bzw. des vom System erkannten Objekts. Abbildung 7.24 verdeutlicht dies. Da alle Fahrzeuge auf dem Boden stehen und somit eine ähnliche vertikale Position haben, wird die Überschneidung der Boxen in vertikaler Richtung nicht betrachtet. Die Objekte in den Sequenzen wurden in drei Kategorien eingeteilt, welche es

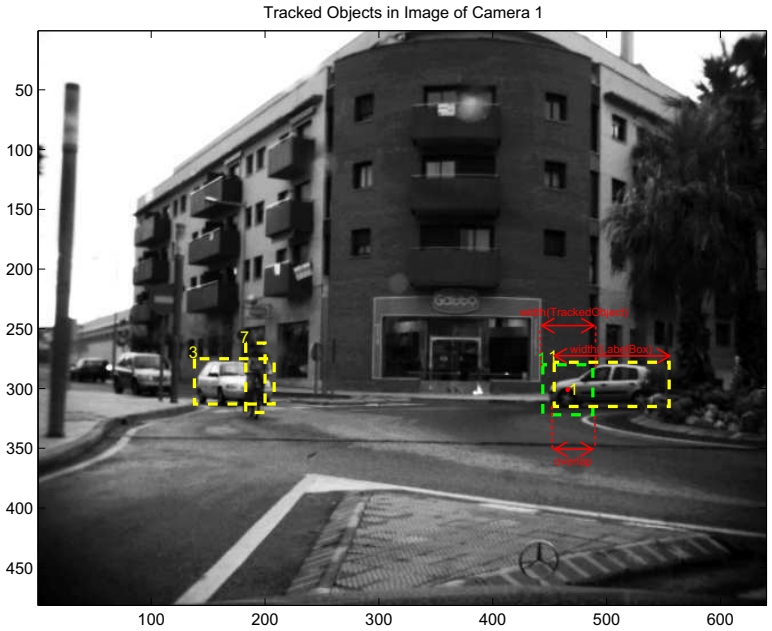

Abbildung 7.24: Bewertung des Trackings anhand von manuell gelabelten Objekten (gelbe Rechtecke). Das grüne Rechteck zeigt das vom System erkannte Objekt an.

ermöglichen, genauer zwischen den Ergebnissen verschiedener Arten von Objekten zu unterscheiden:

- Kategorie I: Objekte, die sich im Bild von links nach rechts bewegen und in ca. 40 m bis 50 m Entfernung von der Kamera in den Kreisverkehr einfahren.
- Kategorie II: Objekte, die sich im Bild von rechts nach links bewegen und in ca. 35 m Entfernung von der Kamera in den Kreisverkehr einfahren.
- Kategorie III: Objekte, die eine Kurvenfahrt im Kreisverkehr durchführen.

7.5.2 Ergebnisse und Auswertung

Zur Bewertung der Trackingergebnisse wurde zwei Testsequenzen verwendet. Die Sequenzen wurden an einem Kreisverkehr in Spanien aufgenommen und beinhalten sehr komplexe Fahrbewegungen von verschiedenen Verkehrsteilnehmern vor strukturierten Hintergründen. In den Tabellen 7.4 bis 7.6 sind die Verfolgungsraten des Trackingsystem aufgezeigt. Es wird der Median der kamerabasierten Verfolgungsraten aller Objekte der Kategorie in der Testsequenz und das 25% sowie 75% Quantil der kamerabasierten Verfolgungsraten aufgezeigt. Für die Kategorie I (Tabelle 7.4) variieren die Ergebnisse stark, in einer Sequenz beträgt der Median der kamerabasierten Verfolgungsraten 94.17% in der anderen nur 52.69%. Diese Unterschiede sind auf fehlerbehaftete Stereodaten und auf das darauf basierende Clustering zurückzuführen.

Tabelle 7.4: Verfolgungsrate für Objekte der Kategorie I mit zugehörigen Quantilen.

Sequenz	Median	25% und 75% Quantile
A	52.69%	-1.54% +23.32%
B	94.17%	-16.18% +5.08%

Für die Kategorie II (Tabelle 7.5) variieren die Ergebnisse extrem. In einer Sequenz beträgt der Median der kamerabasierten Verfolgungsraten 79.71% in der anderen nur 3.33%.

Tabelle 7.5: Verfolgungsrate für Objekte der Kategorie II mit zugehörigen Quantilen.

Sequenz	Median	25% und 75% Quantile
A	79.71%	-9.11% +8.43%
B	03.33%	-2.50% +1.16%

Auffällig sind die Unterschiede von bis zu 90 Prozent zwischen der kamerabasierten Verfolgung der Objektkategorien I und II in den Sequenzen. Diese sind auf fehlerbehaftete Stereodaten zurückzuführen, welche die Detektion und Pose-Estimation beeinflussen. So wurden zum Beispiel in Sequenz B Objekte der Kategorie II nur in drei Prozent der Zeitschritte getrackt. Wie in Abbildung 7.25 (links) zu sehen ist, verdeckt ein Regentropfen an einer Stelle im Bild (rot markiert) die Fahrzeuge. Als Konsequenz sind die daraus resultierenden Punktkorrespondenzen für die Berechnung des optischen Flusses fehlerhaft, wie in Abbildung 7.25

(rechts) gezeigt. Objekte, an deren Position der Median des Flusses nicht signifikant von Null verschieden ist, werden im vorliegenden System gelöscht. Dies hat den Grund, dass ansonsten oft „Pseudoobjekte“ an Stellen erzeugt würden, an denen viele Stereopunkte liegen, obwohl dort kein Objekt vorhanden ist. Da der Fluss an diesen Stellen um Null streut, wird angenommen, dass es sich nicht um ein für das System relevantes Objekt handelt, und es wird gelöscht. Ist nun jedoch das Bild durch einen Tropfen auf der Kamera gestört, so ist der Fehler des optischen Flusses für Stereopunkte im Bereich eines korrekt erkannten Objektes um mindestens eine Größenordnung größer als der mittlere Fehler ohne Verdeckung durch einen Tropfen. Wurden keine Punktkorrespondenzen auf Grund dieser Verdeckung erkannt, ist kein Fluss an dieser Stelle vorhanden. Unter diesen Umständen werden korrekt erkannte Objekte wieder gelöscht.

Ein weiterer Aspekt für die in diesem speziellen Fall vorliegende geringe Verfolgungsrate ist, dass viele der Objekte der Kategorie II nur wenige Zeitschritte (oft nur ca. 0.2–0.4 Sekunden) zu sehen sind und danach von Objekten der Kategorie I im Bild verdeckt werden. Dies führt dazu, dass mit Hilfe der Standardprädiktion die Fahrtrichtung anhand sehr weniger Trajektorienpunkte bestimmt werden muss. Da auch die ermittelten Trajektorienpunkte um die tatsächliche Position schwanken, ist die Schätzung der Fahrtrichtung anhand dieser wenigen Punkte oft mit gößeren Fehlern behaftet. Dies führt dazu, dass die Verfolgung der Objekte während der Verdeckung mit Hilfe der Standardprädiktion fehlschlägt und das Objekt somit nicht mehr getrackt werden kann. Für die Kategorie III (Tabelle 7.6) werden durchweg gute

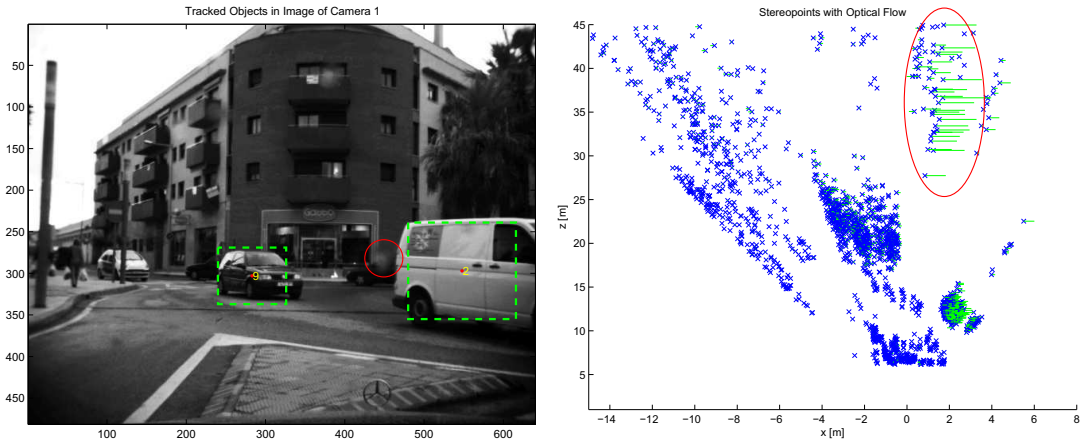

Abbildung 7.25: Links: Ein Tropfen vor der Kamera verdeckt Objekte an dieser Stelle im Bild. Der Tropfen ist mit einem roten Kreis markiert. Rechts: Fehler im optischen Fluss bedingt durch einen Tropfen vor der Kamera. Die falschen Zuordnungen sind mit einem roten Kreis markiert.

Trackingergebnisse erzielt. Der Median der kamerabasierten Verfolgungsraten ist immer größer als 93% und die Quantile sind nah am Median.

Abbildung 7.26: Ergebnisse des ins Bild der Masterkamera projizierten Trackingergebnisses für zwei Zeitschritte. Links: Zeitschritt 97. Rechts: Zeitschritt 113.

Tabelle 7.6: Verfolgungsrate für Objekte der Kategorie III mit zugehörigen Quantilen.

Sequenz	Median	25% und 75% Quantile
A	93.51%	±0%
B	98.65%	-1.35% +1.35%

Abbildung 7.26 zeigt die ins Bild der Masterkamera projizierten Trackingergebnisse für zwei Zeitschritte der Sequenz A.

7.6 Wertung der Trackingergebnisse

In diesem Kapitel wurden qualitative und quantitative Ergebnisse der in dieser Arbeit entwickelten Trackingsysteme vorgestellt. Die Ergebnisse wurden auf Basis verschiedener reeller Testsequenzen für unterschiedliche Szenarien berechnet. In diesem Abschnitt soll eine Wertung vorgenommen werden und die erzielten Ergebnisse mit denen anderer Verfahren aus dem Stand der Technik (Kapitel 2) verglichen werden.

Die in diesem Kapitel ausführlichste Evaluierung wurde im Abschnitt 7.3 vorgestellt. Dabei wurden die Trackingergebnisse von zehn in dieser Arbeit vorgestellten Trackingsystemen für die kamerabasierte Verfolgung des menschlichen Hand-Unterarm-Bereichs quantitativ evaluiert. Weitere Ergebnisse wurden für das Tracking der menschlichen Kopf-Schulter-Partie und die kamerabasierte Verfolgung von Verkehrsteilnehmern aus einem Auto heraus gezeigt.

7.6.1 Modellbasierte 3D-Verfolgung von Körperteilen

Die Auswertung der verschiedenen Versuche mit den Systemvarianten zeigt, dass durch die vorgestellten Algorithmen zur modellbasierten 3D-Objektverfolgung ein genaues und zeitlich stabiles Tracking mit hohen Verfolgungsraten möglich ist. Die Ergebnisse der kamerabasierten

Verfolgung des menschlichen Hand-Unterarm-Bereichs zeigen, dass durch System 1 (MOCCD-Tracking) ein genaues Tracking möglich ist, der Ansatz aber keine einhundertprozentigen Verfolgungsraten ermöglicht. Bei den Ergebnissen von System 1 wurde außerdem gezeigt, dass die Verwendung von RGB-Bilder als Eingabedaten keine signifikanten Verbesserungen gegenüber Grauwertbildern als Eingabedaten bringt. Daraus lässt sich ableiten, dass die Verwendung einer Grauwertkamera ausreichend ist. Dadurch verringert sich die Laufzeit der Algorithmen und es kann mehr Licht auf dem Sensor auftreffen, was für ein industrielles Sicherheitsszenario sehr interessant ist. Die Versuche zum Tracking der menschlichen Kopf-Schulter-Partie zeigen, dass System 1 sich auf andere Körperteile erweitern lässt. Wichtig ist dabei die Modellierung des getrackten Körperteils durch ein Konturmodell.

Eine für ein Sicherheitssystem notwendige zeitliche Robustheit und ein Selbsttest des Trackingsystems muss durch andere Module sichergestellt werden. Die Ergebnisse des MOCCD-Trackings mit der Verwendung des Reinitialisierungsmoduls (System 2) zeigen dies. Ein Verifikationsmodul erkennt ein schlechtes Pose-Estimation-Ergebnis während des Trackings und durch ein Modul zur Reinitialisierung wird dieses verbessert. Durch die Erweiterung um die beiden Module werden einhundertprozentige Verfolgungsraten auf den Testsequenzen erzielt und die Genauigkeit des Systems wird gesteigert. Bei der instantanen Bewegungsschätzung zur Vorhersage der Bewegung des getrackten Objekts gibt es für System 2 noch Verbesserungsmöglichkeiten. Die quantitativen Trackingergebnisse des Systems 4 (Shape-Flow-Tracking mit Reinitialisierung) in den Abschnitten 7.3.4 und 7.4.3 zeigen dies sehr eindeutig. Durch ein Konturtracking, die direkte Bewegungsschätzung mit dem Shape-Flow-Algorithmus und die Verwendung des Reinitialisierungsmoduls wird ein metrisch genaues und zeitlich robustes Tracking mit guten Ergebnissen bei der instantanen Bewegungsvorhersage erzielt. Die mittleren Positionsfehler des Systems 4 für das Tracking des Unterarms und der Kopf-Schulter-Region liegen zwischen 45 und 90 mm, bei Standardabweichungen der Positionsfehler von 20 bis 50 mm. Nimmt man Grauwertbilder als Eingabedaten, so ist die Laufzeit des Systems 4 aber auch die höchste gemessene in dieser Arbeit. In den Verfahren steckt aber das Potential einer realzeitfähigen Implementierung. Dazu sollten die Algorithmen nach C/C++ portiert und dabei die verwendeten Datenstrukturen und Implementierungen optimiert werden.

Ein weiteres Trackingsystem (Systems 7) mit ähnlicher Genauigkeit wie das System 4 verwendet eine Objektverfolgung mit zwei verschiedenen 3D-Pose-Estimation-Verfahren, welche auf Konturen und 3D-Punktewolken basieren. Es werden ähnliche Verfolgungsraten und Positionsgenauigkeiten erzielt. Die Ergebnisse der instantanen Bewegungsschätzung sind im Vergleich zu System 4 etwas besser und somit die besten erzielten Werte zur Bewegungsvorhersage in dieser Arbeit. Die Laufzeit ist ähnlich zu der von System 4.

Im Folgenden werden die Stärken und Grenzen der in dieser Arbeit entwickelten Verfahren zur modellbasierten Objektverfolgung vorgestellt.

Stärken der entwickelten Verfahren

Wie die Versuche auf Basis der verschiedenen realen Testsequenzen mit den teilweise komplexen Hintergrundkonfigurationen gezeigt haben, sind die beiden besten Trackingsysteme (System 4 und System 7) in der Lage, menschliche Körperteile robust und stabil über die Zeit zu verfolgen. Es können viele verschiedene Bewegungen getrackt werden, wie z.B.:

- schnelle Umkehrbewegungen, beim Anziehen von Schrauben
- Handbewegungen entlang der Sichtlinie
- Greif -und Steckbewegungen der Hand
- laterale Bewegungen der gesamten Hand-Unterarm-Extremität
- schnelle Putzbewegungen
- Bewegungen der Testperson auf die Kamera zu
- Nick- und Neigungsbewegungen des Kopfes.

Dabei sind nur grobe Informationen über die Testperson notwendig, denn im Vergleich zu anderen Verfahren (Schmidt u. a., 2006; Stenger u. a., 2001; Rosenhahn u. a., 2005a; Gall u. a., 2009) wird diese nicht individuell ausgemessen, sondern die Werte der Längen von Unterarm, Hand, Hals und Kopf nur grob festgelegt und die Deformationsparameter mitgeschätzt.Mit den entwickelten Verfahren ist es möglich, verschiedene Handkonfigurationen, wie z.B.:

- eine ausgestreckte Hand
- eine Faust
- ein Handfeger in der Hand
- ein Schraubenschlüssel in der Hand
- oder eine Zeigegeste

zu verfolgen. Dazu muss das Modell nicht angepasst werden. In Teilen der Sequenzen ist der Unterarm der Testpersonen mit Kleidung bedeckt oder die Testperson trägt einen Handschuh. Dies beeinflusst die entwickelten Verfahren nicht negativ. Hierzu ist noch zu erwähnen, dass die Kleidung nicht zu stark vom Körper abstehen darf, da das symmetrische Modell eine dadurch entstehende Kontur nicht abbilden kann.

Die besten Trackingsysteme (System 4 und System 7) erlauben eine direkte Schätzung der 3D-Pose-Ableitung aus den Daten einer Bildsequenz. Dies ist für ein Sicherheitssystem entscheidend, denn Latenzzeiten müssen minimal gehalten werden.

Weiterhin muss erwähnt werden, dass sich bei der Verwendung von RBG-Werten im Vergleich zur Verwendung von Grauwerten als Eingabedaten keine signifikanten Unterschiede in den Ergebnissen ergeben. Diese Aussage ist sehr interessant, da sich durch die Verwendung von Grauwertdaten die Laufzeit der Algorithmen verkürzen lässt und mehr Licht auf dem Sensor auftreffen kann, da keine Farbfilter, wie bei Farbkameras häufig üblich, verwendet werden müssen. Eine höhere Lichtstärke ist für ein industrielles Sicherheitsszenario sehr interessant, da die Leistung und Anzahl der verwendeten Lampen reduziert werden kann.

Die Trackingsysteme sind in der Lage, durch ein Verifikationsmodul eine Art Selbsttest des Systems durchzuführen. Dabei kann erkannt werden, dass sich das in den Bildern getrackte Objekt im Aussehen verändert hat und somit möglicherweise verloren wurde. Dazu wird eine zeitliche Konsistenz des Objektaussehens überprüft. Dies dient auch einer weiteren Erhöhung der Robustheit, da falls der Selbsttest nicht bestanden wurde, ein Modul zur Reinitialisierung das Objekt wiederfinden kann.

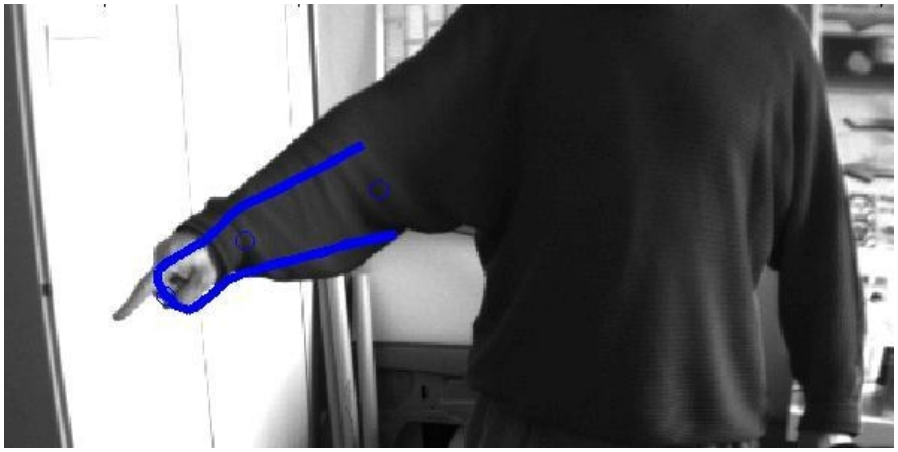

Abbildung 7.27: Ein Proband mit sehr weiter Kleidung ist dargestellt. Wie die beste Messung (blau) zeigt, kann die Kontur der Hand-Unterarm-Extremität nicht besonders gut nachgebildet werden.

Grenzen der entwickelten Verfahren

Bei sehr weiter Kleidung können die Annahmen im Hand-Unterarm-Modell (Abschnitt 3.2.1) zu Problemen führen (Abbildung 7.27). Mit dem Modell kann nämlich eine durch sehr weite Kleidung hervorgerufene Kontur nicht nachgebildet werden. Dies kann zu einem Verlust an Genauigkeit, Robustheit und Stabilität der entwickelten Systeme führen. Ein Vorteil des Produktionsszenarios ist, dass weite Kleidung durch Sicherheitsvorgaben aufgrund möglicher Verletzungsrisiken nicht erlaubt ist. Eine weitere Limitierung der entwickelten Systeme ist notwendige Initialisierung des Modells. Zwar wird im Kapitel 6 eine Lösung für das Initialisierungsproblem gegeben, diese ist aber stark abhängig von einer Heuristik. Hinter der Heuristik steckt die Annahme, dass während des Arbeitsprozesses der menschliche Hand-Unterarm-Bereich sich in der Nähe des Interaktionsobjekts am Schnellsten bewegt. Ist die Annahme verletzt, kann die Objektdetektion, also die Initialisierung des Modells, fehlerbehaftet sein. Außerdem darf die Modellinitialisierung nicht zu weit vom tatsächlichen Objekt entfernt sein, denn die zur 3D-Pose-Estimation verwendeten Verfahren sind Pose-Verfeinerungs-Algorithmen. Diesen Nachteil weisen aber auch andere im Stand der Technik (Kapitel 2) recherchierte Verfahren zur kamerabasierten Verfolgung von Objekten im 3D-Raum auf ((Delamarre u. Faugeras, 1998), (Stenger u. a., 2001), (Schmidt u. a., 2006), (Gall u. a., 2009)), weil der 3D-Suchraum sehr groß ist und eine Detektion schwierig werden kann.

Eine weitere Möglichkeit der Modellinitialisierung ist aber durch das Szenario dieser Arbeit gegeben. Es könnten nämlich Arbeitsprozesse gestaltet werden, in denen die Position des Objekts zum Beginn des Prozesses bekannt ist, beispielsweise muss sich der Unterarm an einer definierten Position im 3D-Raum befinden bevor der Arbeitsprozess beginnen kann.

Nun sollen noch Grenzen des MOCCD-Algorithmus aufgezeigt werden. Es ist möglich, dass die kamerabasierte Verfolgung eines Objekt abbrechen kann, da der MOCCD-Algorithmus das Modell nicht an das tatsächliche Objekt in den Bilddaten anpassen kann. In der Abbildung 7.28 gelingt es dem MOCCD-Algorithmus beispielsweise nicht, die Handspitze (Punkt ${}^{W}\mathbf{p}_8$) über die komplette Zeit zu verfolgen. In Abbildung 7.28 sind die Bilder der Masterkamera zum Be-

ginn, der Mitte und am Ende einer Sequenz, mit zugehöriger Schätzung durch den MOCCD-Algorithmus dargestellt. Man sieht, dass das Modell an der Handspitze nicht mitgeführt werden kann. Der Grund hierfür liegt in den Statistiken der Kamerabilder auf der linken (weißes

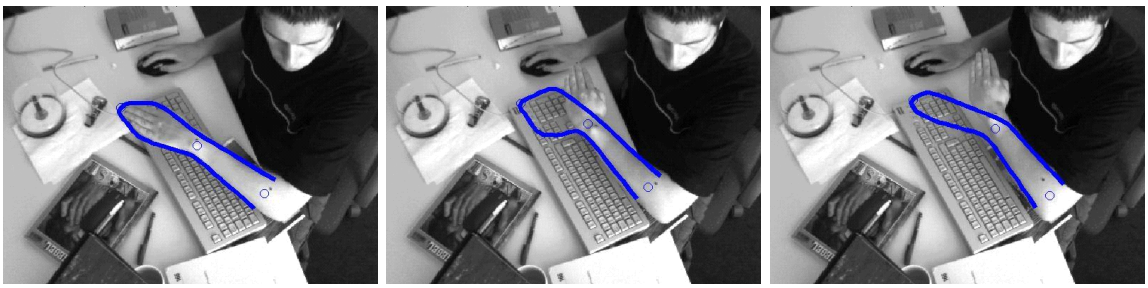

Abbildung 7.28: Die Verfolgung der Handspitze mit System 1 versagt für eine Beispielsequenz. Es sind Bilder einer Kamera zum Beginn, in der Mitte und am Ende, mit zugehöriger bester Schätzung dargestellt.

Blatt) und der rechten Seite der Hand (Tastatur). In Abbildung 7.29 sind diese Statistiken einer Kamera für die Senkrechte 22 (linke Seite) und die Senkrechte 38 (rechte Seite) dargestellt. Es ist zu erkennen, dass die Statistiken der Senkrechten 22 gut zu trennen sind, aber die Statistiken der Senkrechten 38 kaum. Da sich am Anfang der Sequenz 2 die Hand in Richtung der rechten Seite (schlechte Statistiken) bewegt und der Unterarm sich kaum bewegt, bleibt die Handspitze an dem weißen Papier „hängen“. Nachdem die Bewegung des Unterarms einsetzt, wird die „hängengebliebene“ Hand wie in Abbildung 7.28 (Mitte und Ende) zu sehen, mit dem kompletten Unterarm nach rechts gezogen. In dieser Arbeit wurden einige Möglichkeiten zur Erkennung oder zur Vermeidung der besagten Grenzen beim MOCCD-Algorithmus vorgestellt. Dabei sei die Verwendung eines erweiterten Eingabebildes für den MOCCD-Algorithmus (Abschnitt 3.3.1), die Nutzung eines Verifikations- und Reinitialisierungsmoduls (Abschnitte 3.4 und 3.6) oder die gewichtete Kombination der Schätzung des MOCCD-Algorithmus mit dem ICP-Algorithmus (Abschnitt 6.2) zu nennen. All diese Erweiterungen ermöglichen es erst, ein robustes und zeitlich stabiles Tracking zu realisieren.

Ergebnisse im Vergleich zum Stand der Technik

In diesem Abschnitt sollen die erzielten Ergebnisse mit den Resultaten anderer Arbeiten aus dem Stand der Technik (Kapitel 2) quantitativ verglichen werden. Ein solcher Vergleich ermöglicht eine weitere Wertung der in dieser Arbeit erzielten Ergebnisse, ist aber nicht einfach möglich, da das Szenario dieser Arbeit sehr speziell ist und wenige Autoren vergleichbare quantitative Ergebnisse ihrer Verfahren veröffentlichen. Sehr häufig wird evaluiert, wie lange das Objekt über eine Sequenz getrackt werden kann, also die Verfolgungsrate analysiert. Selten geben die Autoren an, wie genau ihre Verfahren wirklich arbeiten, d.h. dass sie ihre Ergebnisse mit einem viel genaueren Referenzsystem vergleichen. Beispielsweise wird in keiner im Stand der Technik (Kapitel 2) erwähnten Veröffentlichung, in der ein Mehrkamerasystem mit kleiner Basisbreite verwendet wird (Abschnitt 2.2.2), ein quantitatives Ergebnis angegeben. Aus diesem Grund werden neun Testsequenzen, aufgezeichnet mit dem in dieser Arbeit verwendeten Kamerasystem (Abschnitt 3.1), mit zugehörigen Ground-Truth-Daten im Internet

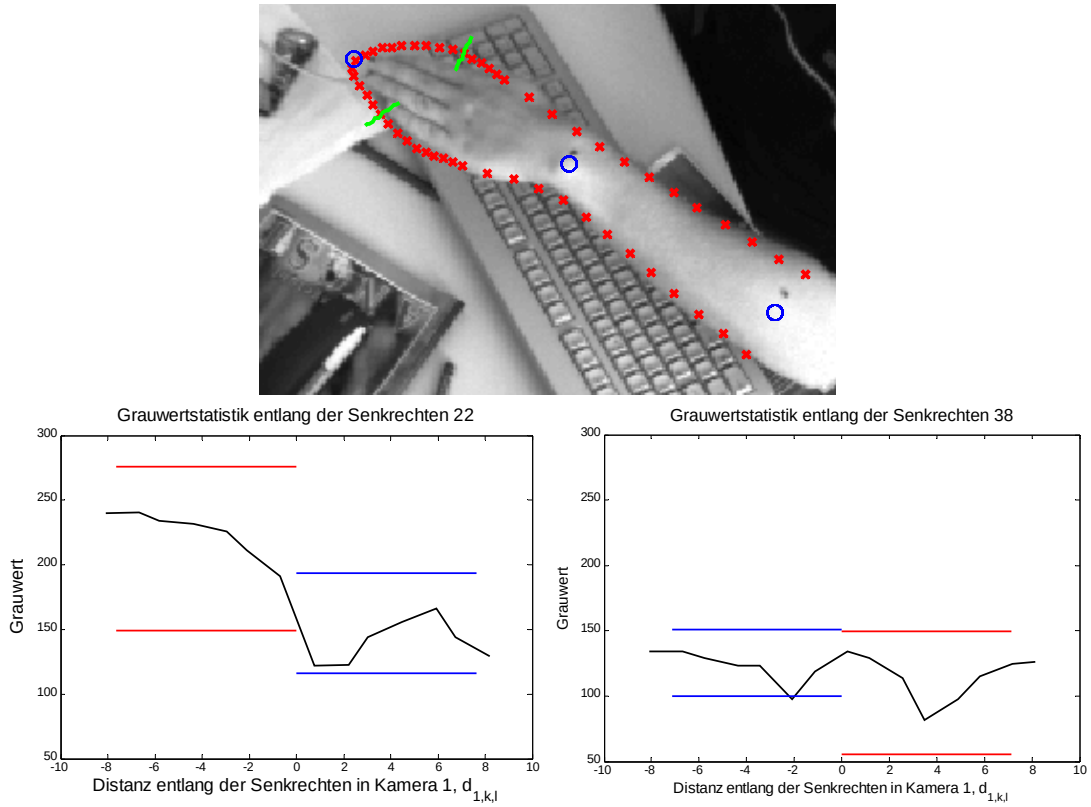

Abbildung 7.29: Grauwertstatistiken für zwei Senkrechte, 22 (links) und 38 (rechts), zum Beginn der Beispielsequenz. Die Statistiken der Senkrechten 22 sind gut zu trennen, aber die der Senkrechten 38 gar nicht. Dadurch bleibt das Verfahren auf der linken Seite „hängen“.

bereitgestellt [1].

Einige andere Verfahren ermöglichen aber einen Vergleich der quantitativen Ergebnisse. Mit dem Verfahren von Schmidt u. a. (2006) ist sogar ein direkter Vergleich möglich, da Schmidt u. a. (2006) die Ergebnisse auf einem Datensatz, welcher mit dem in dieser Arbeit verwendeten Kamerasystem aufgezeichnet wurde, veröffentlicht hat (Schmidt, 2009). Der Datensatz ist sehr ähnlich zu Sequenz 1, er zeigt die selbe Testperson, die selbe Umgebung und eine ähnliche Entfernung zur Kamera. Das Trackingsystem von Schmidt u. a. (2006) wurde manuell initialisiert und erzielt auf der Sequenz mit dem monokularen Farbkameraansatz (nur Masterkamera) einen mittleren Positionsfehler von 210 mm. Das ist ungefähr das Dreifache dessen zu dem die Trackingsysteme 4, 5 und 7 imstande sind. Der Vergleich ist etwas unfair, da Schmidt (2009) nur mit den Bildern der Masterkamera arbeitet. Dies zeigt sich daran, dass der Tiefenfehler den größten Anteil am Gesamtfehler ausmacht. Er liegt im Mittel bei 160 mm und bleibt auch über längere Zeiträume über 200 bis zu 400 mm. Schmidt (2009) berichtet außerdem, dass die Erkennung von kleinen Bewegungen der Hand, z.B. beim Festziehen einer Schraube, problematisch ist. Die in dieser Arbeit entwickelten Ansätze sind in der Lage, diese Umkehrbewegungen zu erkennen.

Hahn u. a. (2007) zeigt, dass die Verwendung des MOCCD-Algorithmus als Pose-Estimation-

[1] http://aiweb.techfak.uni-bielefeld.de/content/hand-forearm-limb-data-set

Verfahren beim Tracking, genauere und zeitlich stabilere Ergebnisse liefert als ein Aktive-Konturen-Ansatz (d'Angelo u. a., 2004) oder eine 3D-Erweiterung des Konturtrackingsystems von Blake u. Isard (1998).

Weitere quantitative Vergleiche sind noch mit Verfahren möglich, die ein Mehrkamerasystem mit großer Basisbreite verwenden (Abschnitt 2.2.3).

Rosenhahn u. a. (2005b) vergleichen die mit vier Kameras erzielten Trackingergebnisse mit einem Marken-basierten System mit acht Kameras. Die Testpersonen tragen hautenge Anzüge und das zum Tracking notwendige 3D-Körpermodell wird speziell an die Testperson angepasst. Die so erzielten Ergebnisse sind hochgenau und nicht von den in dieser Arbeit entwickelten Trackingsystemen zu erreichen. Beispielsweise wird ein Winkelfehler des Ellenbogengelenks von 1.3 Grad bei der Durchführung von Liegestützen erreicht. Die großen Vorteile dieses Systems sind die detaillierte Modellierung der Testperson, die einfachen Hintergründe und die Anordnung der vier verwendeten Kameras. Durch die große Basisbreite gibt es kaum Verdeckungen und die Genauigkeit der Algorithmen steigt.

Ziegler u. a. (2006) verwenden vier Stereo-Kamerasysteme (insgesamt 8 Kameras), welche um die beobachtete Szene angeordnet sind. Auf vier Testsequenzen von vier Testpersonen wird in einer Entfernung von 5 m eine mittlerer Positionsfehler der Torsos von 70 mm erreicht. Die in dieser Arbeit entwickelten Trackingsysteme erreichen ähnliche Genauigkeiten bei einer etwas geringeren mittleren Entfernung der Testpersonen. Wiederum ist die große Basisbreite der Kameras für die Genauigkeit entscheidend.

Eine weitere Arbeitsgruppe, die sich intensiv mit der Pose-Estimation von menschlichen Körpern beschäftigt hat, ist die Gruppe um Lars Mündermann an der Stanford University (Mündermann u. a., 2006, 2007). In den Veröffentlichungen wurden Kamerakonfigurationen mit 4, 8 16, 32 und 64 Kameras untersucht. Die mittlere Genauigkeit der Pose-Estimation des gesamten Körpers liegt bei ca. 10–30 cm, wobei die Genauigkeit mit der Anzahl der Kameras steigt. Wiederum sind die Ergebnisse besser als die Ergebnisse der in dieser Arbeit entwickelten Algorithmen. Da im Szenario dieser Arbeit ein Mehrkamerasystem mit großer Basisbreite nicht vorgesehen ist, können die von Mündermann u. a. (2007) beschriebenen Genauigkeiten kaum erreicht werden.

Hofmann u. Gavrila (2009) verwenden drei Kameras mit großer Basisbreite und erzielen auf Basis von 12 Testsequenzen eine mittlere Positionsgenauigkeit von 100 bis 130 mm. Die Ergebnisse sind vergleichbar mit den in dieser Arbeit erzielten Ergebnissen, mit dem Unterschied, dass Hofmann u. Gavrila (2009) mit einer Basisbreite der Kameras von mehreren Metern arbeitet und in dieser Arbeit die Trackingergebnisse mit einer Basisbreite von 150 mm erreicht wurden. In Szenario von Hofmann u. Gavrila (2009) sind die Testpersonen aber etwas weiter entfernt.

Sigal u. a. (2006) stellen einen Datensatz mit rund 50.000 Bildern und dazugehörigen Ground-Truth Pose-Daten zur Verfügung, um sowohl Pose-Estimation-Algorithmen als auch Handlungserkennungssysteme zu evaluieren. Die Sequenzen dieses sogenannten HumanEva-Datasets zeigen dabei Handlungen wie Gehen, Joggen, Kommunikationsgesten, Werfen, Fangen oder Boxen und wurden mit einem Mehrkamerasystem mit großer Basisbreite aufgenommen. Ziel dieses Datensatzes ist es, eine einheitliche Datenbasis zum Vergleich von Verfahren zur 3D-Pose-Estimation und zum 3D-Tracking zu ermöglichen. Eine Reihe von Autoren eva-

luieren ihre Verfahren auf dem HumanEva-Datensatz, unter anderem auch Gall u. a. (2009). Dabei erzielen Gall u. a. (2009) mittlere Positionsfehler von 30 bis 40 mm bei Standardabweichungen um 5 mm. Besonders die geringe Standardabweichung zeigt, wie genau, robust und zuverlässig der Ansatz arbeitet. Besonders durch die Verwendung von Farbdaten, wenig strukturierten Hintergründen, großen Basisbreiten der Kameras und detaillierten Modellen der Testpersonen können solche genauen Ergebnisse erreicht werden. In dem in dieser Arbeit angestrebten Szenario sind solche Genauigkeiten bei den gegebenen Randbedingungen nur schwer möglich.

7.6.2 Verfolgung beliebiger Objekte

Einen anderen Weg als die modellbasierten Verfahren zur 3D-Verfolgung von Körperteilen in dieser Arbeit geht das Trackingsystem 5. Dieses basiert auf einer 3D-Erweiterung des Mean-Shift-Trackings und ist in der Lage, beliebige Objekte kamerabasiert zu verfolgen. Die in den Kamerabildern getrackten Objekte werden nicht detailliert beschrieben, sondern durch Ellipsoide approximiert. Falls ein in den Bildern des Kamerasystems getracktes Objekt nicht mit nur einem Ellipsoiden beschrieben werden kann, werden mehrere Ellipsoide zur Beschreibung des Objekts eingesetzt und diese getrennt voneinander in den Bildern verfolgt. Dazu werden 3D-Punktewolken mit Bewegungsinformation (Abschnitt 5.2) und Bilddaten verwendet. Die Idee hinter dem System ist, dass alle bewegten Objekte in der beobachteten Szene mit Ellipsoiden getrackt werden und auf einer höheren Stufe des Gesamtsystems, z.B. durch eine Aktionserkennung (Abschnitt 8), aus den Bewegungsdaten die Zuordnung zum Objekt, Aktionen oder Handlungen abgeleitet werden.

Im Abschnitt 7.3.5 werden Ergebnisse der kamerabasierten Verfolgung des menschlichen Hand-Unterarm-Bereichs mit dem 3D-Mean-Shift-Tracking (System 5) vorgestellt. Die Positionsgenauigkeit unterscheidet sich dabei nicht von den besten modellbasierten Verfahren zur 3D-Verfolgung von Körperteilen (Systeme 4 und 7), nur bei der instantanen Bewegungsschätzung ist System 5 schlechter. Wobei die Laufzeit der prototypischen Implementierung von System 5 viel geringer ist als für die Systeme 4 und 7. Ein Nachteil des 3D-Mean-Shift-Trackings ist, dass nicht sichergestellt werden kann, dass immer dasselbe Objekt getrackt wird oder um welches Körperteil es sich dabei handelt. Beispielsweise kann in System 5 bei der kamerabasierten Verfolgung das Objekt von der Hand entlang des Unterarms zum Oberarm wandern, wenn Farbe oder Grauwert des kompletten Arms ähnlich sind.

System 5 ist genereller nutzbar als alle anderen in dieser Arbeit entwickelten Verfahren, dies wird im Abschnitt 7.5 gezeigt. In diesem Abschnitt wurden Ergebnisse vorgestellt, die nichts mit dem angestrebten Szenario dieser Arbeit zu tun haben. Das 3D-Mean-Shift-Tracking wurde durch Einhaus (2009) zum Tracking von Verkehrsteilnehmern in einem Kreisverkehrsszenario angepasst und erweitert. Eine Erweiterung der Verfahren zum Konturtracking auf das Verkehrsszenario wäre auch denkbar. Wichtig ist dabei aber, dass die Verkehrsteilnehmer über 3D-Konturen modelliert werden können.

Teil III

3D-Bewegungs- und Aktionserkennung

In dieser Dissertation geht es um die Realisierung der Vision „Sichere Mensch-Industrieroboter-Interaktion“ mit einem intelligenten technischen Überwachungssystem. Dazu ist es wichtig, die in den Bildern eines Kamerasystems beobachteten Objekten zu segmentierten, verfolgen und deren Bewegungen vorherzusagen. Eine Lösung dafür wurde im vorherigen Teil dieser Dissertation bereits beschrieben. Zur Realisierung der Interaktion zwischen Mensch und Industrieroboter ist es aber noch wichtig, zu „verstehen“ welche Aktionen, Handlungen oder Handlungsabsichten von den mit dem Kamerasystem beobachteten Objekten ausgehen.

In diesem Teil der Dissertation soll es um ein intelligentes technisches System zur Aktionserkennung gehen. Notwendige Grundvoraussetzung ist ein Trackingsystem zur kamerabasierten Verfolgung von Objekten. Dabei kann jedes der im vorherigen Teil dieser Dissertation beschriebenen Trackingsysteme eingesetzt werden.

In der Beschreibung des technischen Systems zur Aktionserkennung wird wiederum die Notation von Craig (1989) verwendet. Mit ${}^{C}\mathbf{x}$ wird ein Punkt $\mathbf{x}$ im Koordinatensystem C bezeichnet. Außerdem werden Vektoren klein und fett und Matrizen groß und fett geschrieben.

Kapitel 8

System zur 3D-Bewegungs- und Aktionserkennung

In diesem Kapitel soll ein System zur Realisierung des zweiten Schwerpunktthemas der Erkennung von Aktionen, Handlungen und Handlungsabsichten, welche von den beobachteten Objekten ausgehen, vorgestellt werden. Die Erkennung basiert hierbei auf einer zuvor durchgeführten 3D-Szenenanalyse. Als aktionsspezifisches Merkmal werden Trajektorien von 3D-Punkten zur Aktionserkennung verwendet.Voraussetzung für trajektorienbasierte Merkmale ist immer eine robuste und zeitlich stabile Objektverfolgung. Diese kann durch die bereits vorgestellten Trackingsysteme erfolgen. Dabei ist das entwickelte System zur 3D-Bewegungs- und Aktionserkennung so gestaltet, dass jedes der in den Kapiteln 3 bis 6 beschriebenen Trackingsysteme die Eingabedaten für die Erkennung liefern kann. Zudem wird der Objektindex des relevanten Objekts aus allen getrackten Objekten geschätzt. Der Objektindex ist dabei eine ganzzahlige Nummer aus der Liste der getrackten Objekte. Das relevante Objekt ist das getrackte Objekt, welches die zu erkennenden Aktionen ausführt, z.B. die Hand des Werkers. Die Schätzung des relevanten Objekts ist notwendig, wenn wie beim 3D-Mean-Shift-Tracking (Kapitel 5) mehrere Objekte verfolgt werden, ohne zu wissen, welches das aktionsausführende Objekt ist.

Erst das Erkennen von Handlungen und Handlungsabsichten eines menschlichen Arbeiters ermöglicht eine direkte Interaktion zwischen Mensch und Industrieroboter. Mit dem gewählten Trackingsystem werden die Bewegungsdaten des Objekts extrahiert. Mit dem Aktionserkennungssystem werden die Wahrscheinlichkeiten der Zugehörigkeiten zu den verschiedenen zuvor definierten Aktionsklassen geschätzt, entschieden wird sich für die Aktionsklasse mit der besten Schätzung. Dies ist als Grundlage einer Interaktion notwendig, sodass der Roboter darauf geeignet reagieren kann. Außerdem müssen von der normalen Tätigkeit abweichende Handlungen erkannt und somit mögliche Kollisionen mit einem Industrieroboter verhindert werden. Im Fall einer unbekannten Bewegung geht das Aktionserkennungssystem in den Sicherheitsmodus. Durch eine Bewegungsvorhersage können mögliche Kollisionen mit anderen bewegten Objekten erkannt und zuverlässig verhindert werden. Nur so kann eine sichere und direkte Interaktion zwischen Mensch und Industrieroboter realisiert werden. Sobald für den Menschen eine Gefahr besteht, müssen nach *DIN EN ISO 12100* automatische Abschaltesysteme den Roboter zum Stillstand bringen.

8.1 Randbedingungen eines Systems zur Aktionserkennung im Produktionsszenario

Ein weiteres Ziel dieser Dissertation ist der Entwurf eines intelligenten technischen Systems zur Aktionserkennung in einem Produktionsszenario. Bei realen Umgebungen, wie beispielsweise in der Automobilproduktion, gibt es verschiedene Herausforderungen, welche von einem zukünftigen Überwachungssystem beherrscht werden müssen. Das getrackte Objekt, z.B. der Werker, kann sich unerwartet verhalten, kann unkontrollierte Bewegungen ausführen, kann den Arbeitsprozess immer wieder unterbrechen oder diesen in einer unbekannten Reihenfolge durchführen. Diese von der normalen Tätigkeit abweichenden Handlungen müssen erkannt werden, um mögliche Kollisionen mit einem Industrieroboter verhindern zu können.

Eine weitere Randbedingung des Szenarios dieser Arbeit gibt vor, dass nur wenige Trainigsdaten zur Aktionserkennung zur Verfügung stehen. Es ist nicht praktikabel und zu kostenintensiv, viele verschiedene Werker bei ihrer Tätigkeit aufzunehmen, die extrahierten Bewegungsdaten mit Aktionsklassen zu labeln und dann das Erkennungssystem zu trainieren. Es ist aber möglich, die Bewegungsdaten weniger Vorarbeiter, welche den Arbeitsprozess detailliert kennen und die Werker angelernt haben, aus aufgenommenen Kamerabildern zu extrahieren und als Grundlage für das Erkennungssystem zu verwenden. Wenige Trainigsdaten bedeutet dabei, dass ungefähr die zehnfache Menge an Testdaten vorhanden ist. Das entwickelte System muss also in der Lage sein, auch bei kleinen Stichproben die Aktionserkennung zu belernen und zu adaptieren. Ein Vorteil eines solchen Systems ist, dass es auf andere Prozesse einfacher übertragbar ist oder sich veränderte Prozessteile einfacher austauschen lassen. Dadurch sinkt der Wartungsaufwand, was in einem realen Produktionsszenario sehr wichtig ist.

In der Anwendung kann nicht davon ausgegangen werden, dass die Interaktionsobjekte, also z.B. Motorenbereiche, immer an der selben definierten Position sich befinden. Es kann sein, dass sich der Kamerastandpunkt verändert oder die Anordnung der Arbeitsstation verändert wird, was die auch die Position der Interaktionsobjekte verschiebt. Es muss also eine Rotations- und Translationsinvarianz der Aktionserkennung realisiert werden. Außerdem ist es nicht praktikabel, das Erkennungssystem nach jeder Änderung der Kameraposition oder der Objektpositionen neu zu trainieren.

Weiterhin kann im angestrebten Szenario dieser Arbeit angenommen werden, dass der zyklische vorgegebene Arbeitsablauf des Werkers und die 3D-Position der Interaktionsobjekte (Motor, Roboter, etc.) bekannt ist. Erste Anwendung könnte das Aktionserkennungssystem im Bereich der Vollständigkeitsprüfung oder Absicherung bei der Aggregatemontage sowie bei der Absicherung von „Handeinlegestationen“ finden.

8.2 Mögliche Ansätze zur Klassifikation von Bewegungen im Produktionsszenario

Das Produktionsszenario stellt besondere Randbedingungen an Ansätze zur Klassifikation von Bewegungen, daher wird in diesem Abschnitt ein Überblick zu möglichen Verfahren gegeben und diese auch bzgl. ihrer Eignung bewertet.

Wie bereits im Stand der Technik (Kapitel 2, Abschnitt 2.3.4) ausführlich beschrieben, scheinen 3D-Trajektorien das geeignete aktionsspezifische Merkmal zu sein. Sie sind eine sehr kompakte und diskriminative Repräsentation der Bewegungsdaten. Zeitliche Aspekte von Bewegungen werden gut abgebildet und auch Beziehungen zwischen einzelnen Objekten können repräsentiert werden. Außerdem ermöglichen 3D-Trajektorien Invarianzen bzgl. Skalierung, Translation und Rotation, was in der Anwendung im Produktionsszenario unbedingt notwendig ist. Ein Vorteil von 3D-Trajektorien gegenüber von Trajektorien von Körperkonfigurationen (Abschnitt 2.3.4) ist, dass es keine gegenseitige Abhängigkeit von Trackingsystem und Aktionserkennungssystem gibt, was beliebige Kombinationen von Verfahren zulässt. Dies ist für Sicherheitssysteme besonders interessant, denn für deren Realisierung müssen häufig verschiedene Algorithmen parallel auf unabhängigen Systemen laufen.

Zur eigentlichen Erkennung, Unterscheidung und Klassifikation von raum-zeitlichen Merkmalen werden in der Literatur sehr häufig Standardverfahren eingesetzt. Im Abschnitt 2.3.4 wurden diese bereits vorgestellt und auch mögliche Vor- und Nachteile genannt. Betrachtet man die Literatur im Bereich der trajektorienbasierten Aktionserkennung, so eignen sich zwei der Standardverfahren besonders gut zur Online-Erkennung und -Klassifikation von Bewegungen im Produktionsszenario. Dies sind Hidden-Markov-Modelle (HMMs) und Partikelfilter zur Handlungserkennung.

8.2.1 Hidden-Markov-Modelle (HMMs) und deren Varianten

HMMs sind endliche stochastische Automaten und eignen sich sehr gut, um den zyklischen Prozess der Arbeitsaktion zu modellieren. Außerdem lassen sich durch HMMs kontinuierliche Datenströme, wie 3D-Trajektorien, segmentieren und Modelle erkennen. Leider werden sehr große Menge an Trainingsdaten benötigt, um die internen Wahrscheinlichkeiten des HMMs zu bestimmen, was die Verwendung eines HMM-Standardansatz im Szenario dieser Arbeit ausschließt. Weitere Schwachpunkte der stationären Standard-HMMs sind die Modellierung der zeitlichen Dauer von Aktionen sowie die Behandlung von Bewegungspausen und unbekannten Bewegungen. Zur Bewältigung der Schwächen von HMMs wurden in der Literatur verschiedene Erweiterungen vorgestellt, dies wurden stark getrieben durch die kontinuierliche Spracherkennung. Eine wichtige Erweiterung sind die Hidden-Semi-Markov-Modelle (HSMMs) (Ferguson, 1980), durch diese wird es möglich die zeitliche Dauer von Zuständen zu modellieren. Ein Hidden-State kann also viel länger andauern als ein anderer verborgener Zustand. Als Beispiel nennen Marhasev u. a. (2006), dass ein Flugzeug-Passagier im Allgemeinen viel mehr Zeit an der Sicherheitskontrolle im Flughafen als auf dem Weg von der Kontrolle zum Gate verbringt.

Ein sehr schöner Überblick über HSMMs und deren Varianten wird von Yu (2010) gegeben, außerdem werden auch nichtstationäre HMM- und HSMM-Varianten vorgestellt. Laut Yu (2010) fassen HSMMs einige HMM Erweiterungen zusammen und werden auch als „*explicit duration HMM*“, „*variable-duration HMM*“, „*HMM with explicit duration*“, „*generalized HMM*“ oder „*segmental HMM*“ bezeichnet.

Duong u. a. (2005) führen „*switching HSMM*“ ein und verwenden diese zur Erkennung von Aktivitäten und abnormalem Verhalten bei Tätigkeiten des täglichen Lebens. Das verwendete Trainingsset ist genauso groß wie das Testset, was im Produktionsszenario leider nicht möglich

ist. Für abnormales Verhalten gibt es auch Testsequenzen.

Nichtstationäre HSMMs (NHSMMs) wurden von Vaseghi (1995) im Bereich der Spracherkennung und von Sin u. Kim (1995) zur Handschrifterkennung eingesetzt. Marhasev u. a. (2006) erweitern die NHSMMs und sind laut eigenen Angaben die Ersten, die NHSMMs im Bereich der Aktivitätserkennung einsetzen. Das Szenario ist die Erkennung verschiedener Aktivitäten von Passagieren auf Flughäfen. Die Trainings- und Testdaten stammen aus einer Simulation. Daher gibt es riesige Datensätze, was ein Lernen der internen Wahrscheinlichkeiten erst ermöglicht. Die nichtstationäre Erweiterung sieht vor, dass die Wahrscheinlichkeit von einem Hidden-State eine Transition in einen anderen verborgenen Zustand durchzuführen, von der Zeitdauer seit dem Eintritt in diesen Zustand abhängt.

Marhasev u. a. (2009) vergleichen HMMs, HSMMs und NSHSMMs zur klinischen Diagnose, Patienten müssen Labyrinthaufgaben lösen und werden dabei untersucht. Es werden die aufgezeichneten Daten von 75 gesunden Testpersonen zum Training verwendet und versucht mit HMMs, HSMMs und NSHSMMs abnormales Verhalten in den Datensätzen zu erkennen. Die trainierten HMMs waren nicht in der Lage das geforderte Problem zu lösen. HSMMs und NSHSMMs konnten durch die Modellierung der Dauer von Zuständen viel bessere Ergebnisse erzielen. In den Ergebnissen lässt sich kein signifikanter Unterschied zwischen HSMMs und den nichtstationären NSHSMMs erkennen. Für das betrachtete Szenario ist es scheinbar wichtiger die Dauer von Zuständen zu modellieren, als die Transitionswahrscheinlichkeit von dieser Dauer abhängig zu machen. Im in dieser Arbeit betrachteten Szenario sind nichtstationäre Transitionswahrscheinlichkeiten aber besonders wichtig, da bestimmte Referenzbewegungen von Aktionen mehrfach durchgeführt werden können oder auch Arbeitsprozesse unterbrochen werden können. Ein Unterbrechen der ausgeführten Handlung würde die von Marhasev u. a. (2009) verwendeten HSMMs und NSHSMMs zum Scheitern verurteilen. Zur Lösung dieses Problems wird in dieser Arbeit eine neuartige Zwei-Schichten-Architektur zur Erkennung verwendet, welche es ermöglicht die zu erkennenden Handlungen auch unterbrechen zu können. Außerdem ist es nicht unbedingt sinnvoll die nichtstationären Transitionswahrscheinlichkeiten nur von der Dauer eines Zustands abhängig zu machen, denn diese berücksichtigt nicht die Geschwindigkeit einer ausgeführten Bewegung. Die Dauer kann stark schwanken und beispielsweise könnte es große Unterschiede in der Ausführungsdauer von Aktionen im Lern- und Testset geben.

8.2.2 Partikelfilter zur Handlungserkennung

Einen modernen Ansatz zur probabilistischen Zustandsschätzung bilden die Sequenziellen Monte-Carlo-Methoden (Doucet u. a., 2001). Stellvertretend sei der CONDENSATION-Algorithmus (Isard u. Blake, 1998) genannt. Partikelfilter erlauben multimodale Schätzungen, sind sehr flexibel in ihrer Handhabung und sind sehr einfach erweiterbar.

Die Arbeit von Black u. Jepson (1998) ist ein sehr bekannter und häufig zitierter Ansatz zur Handlungserkennung mit Trajektorien-Merkmalen. Der CONDENSATION-Algorithmus wird zum *Condensation-based Trajectory Recognition* (CTR) Verfahren erweitert. Ein Vorteil des Verfahrens liegt in der automatischen Konzentration auf relevante Teile und der geringen Anzahl an Trainingsbeispielen. Diese Vorteile machen den Einsatz von Partikelfiltern zur Aktionserkennung im Produktionsszenario sehr interessant.

Fritsch u. a. (2004) erweitern das CTR-Verfahren um Kontextinformation. Es werden zusätzliche Informationen bezüglich des Vorhandenseins sowie die räumliche Entfernung zu assoziierten Objekten verwendet. Die Objekte werden dabei einzelnen Aktionen zugeordnet und in den Kamerabildern erkannt. Die Kontextinformation wird in die Gewichtsberechnung im CTR-Verfahren einbezogen. Es wird sehr schön gezeigt, wie einfach der Ansatz erweiterbar ist, auch neue Aktionsklassen können sehr einfach hinzugefügt werden.

Hofemann (2007) erweitert in seiner Dissertation die Arbeiten von Black u. Jepson (1998) und Fritsch u. a. (2004). Wie bei Fritsch u. a. (2004) wird situativer und räumlicher Kontext eingesetzt. Im Gegensatz zu Black u. Jepson (1998) beschreibt Hofemann (2007) eine Methode zum Generieren neuer Modelle (Referenztrajektorien). Ein Vorteil dieses Ansatzes ist, dass die genutzten Referenztrajektorien nicht mehr manuell aus den Beispieldaten erzeugt werden müssen.

8.2.3 Kombination Partikelfilter und HMMs

Aus der Sicht des Autors dieser Arbeit bietet die Kombination von Partikelfiltern und HMMs sehr großes Potential zur Aktionserkennung. HMMs sind mächtige Werkzeuge zur stochastischen Modellierung von zyklischen Prozessen, was sehr gut zu Arbeitsabläufen im Produktionsszenario passt. Durch die Kombination lassen sich Schwächen von HMMs beseitigen und deren Stärken durch die Multimodalität und Flexibilität von Partikelfiltern noch verbessern.

Li u. a. (2006) beschreiben bereits einen sehr innovativen Ansatz zur Zustandsschätzung in HMMs mit einem Partikelfilter. Wie Fritsch u. a. (2004) beziehen Li u. a. (2006) den Objektkontext mit ein. Die HMMs werden klassisch aus einem Lernset abgeleitet. Dies ist im angestrebten Szenario leider nicht so einfach möglich, da nicht genügend Trainingsbeispiele zur Verfügung stehen. Die Idee der Zustandsschätzung in HMMs mit Partikelfiltern bietet aber noch einiges an Potential.

Zur eigentlichen Erkennung, Unterscheidung und Klassifikation der Arbeitsaktionen im Produktionsszenario wird in dieser Arbeit eine neuartige hierarchische Zwei-Schichten-Architektur realisiert. Diese erlaubt sehr hohe Erkennungsraten bei den besonderen Randbedingungen die das Produktionsszenario an eine Erkennung stellt. In der oberen Schicht werden unbekannte von bekannten Bewegungen getrennt. In der unteren Schicht wird der Zustand im zyklischen Arbeitsablauf des Werkers durch ein neuartiges, nichtstationäres HMM geschätzt.

Neu an dem in dieser Arbeit entwickelten Ansatz ist, dass das System mit extrem wenig Trainigsdaten auskommt, sehr einfach erweiterbar ist, eine multimodale Zustandsschätzung erlaubt und nichtstationäre Transitionswahrscheinlichkeiten in einem HMM realisiert. All dies wird möglich durch die Kombination eines Partikelfilters mit einem nichtstationären HMM. Im Vergleich zum System von Li u. a. (2006) ist besonders hervorzuheben, dass das entwickelte System mit viel weniger Trainigsdaten auskommt und im HMM die Vorteile von nichtstationären Transitionswahrscheinlichkeiten genutzt werden. Auch Marhasev u. a. (2009) verwenden nichtstationäre Transitionswahrscheinlichkeiten in HMMs, machen aber im Gegensatz zum Ansatz in dieser Arbeit die Transitionswahrscheinlichkeiten abhängig von der Dauer im verborgenen Zustand. D.h. es wird nicht berücksichtigt wie schnell eine Bewegung ausgeführt wird, was zu Problemen führen kann, wenn es große Unterschiede in der Ausführungsdauer von Aktionen gibt. Was im Szenario dieser Arbeit und in vielen weiteren Anwendungen

definitiv möglich ist.

Das in dieser Arbeit entwickelte System realisiert die nichtstationären Transitionswahrscheinlichkeiten im HMM durch einen neuartigen und sehr vielversprechenden Ansatz. Im Zustandsvektor des Partikelfilters wird neben dem aktuellen Hidden-State auch die Phase der Bewegung geschätzt und die Transitionswahrscheinlichkeiten im HMM abhängig von dieser Bewegungsphase gemacht. Durch die Phasenschätzung kann sehr elegant der Anteil, zu dem eine Bewegung vollführt wurde, im HMM berücksichtigt werden. Denn erst eine hohe Phase, z.B. größer als 90%, zeigt an, dass die Referenzbewegung weitestgehend durchlaufen ist. Damit steigt natürlich die Transitionswahrscheinlichkeit für den nächsten Zustand in der Markov-Kette. Außerdem wird man durch die Phasenschätzung, im Gegensatz zu Marhasev u. a. (2009), viel robuster gegenüber der Ausführungsgeschwindigkeit von Bewegungen. Denn Marhasev u. a. (2009) machen die nichtstationären Transitionswahrscheinlichkeiten im HMM abhängig von der Dauer im verborgenen Zustand, was bei unterschiedlichen Ausführungsgeschwindigkeiten zu Problemen führen wird. Denn es kann schon zu einem Zustandsübergang kommen, obwohl die Bewegung oder Aktion noch nicht vollständig ausgeführt wurde. Das in dieser Arbeit entwickelte Aktionserkennungssystem ist hier viel robuster, da durch den neuartigen Ansatz der phasenabhängigen Transitionswahrscheinlichkeiten im HMM eine Anpassung an die Bewegungsgeschwindigkeit einer Testperson möglich ist, was das Szenario dieser Arbeit unbedingt notwendig macht. Denn in der Ausführungsgeschwindigkeit von Arbeitsaktionen sind im Feld sehr große Schwankungen zu erwarten, was sich im für die Experimente notwendigen Testset (Abschnitt 9.1) ja schon zeigt.

Die Bewegungsphase bietet sogar noch weitere Möglichkeiten, denn man kann durch deren Schätzung mit Referenztrajektorien auf den Startzeitpunkt und auch das Ende eine Bewegung schließen und dies zur eigentlichen Aktionserkennung verwenden (Abschnitt 8.8.4). Außerdem lässt eine Schätzung der Bewegungsphase sehr gut zu einer genauen Langzeitprädiktion von bekannten Bewegungen verwenden, wie Hahn u. a. (2008a) zeigen.

8.3 Vorstellung des betrachteten Beispielszenarios

Am Beispiel von Montagearbeiten eines Werkers an einem Motorblock soll gezeigt werden, dass eine robuste Erkennungsleistung unter den Randbedingungen des Szenarios dieser Arbeit möglich ist. Das gewählte Arbeitsszenario ist realitätsnah, denn einfache Montageprozesse in der Automobilproduktion, bei denen als Erstes eine Interaktion zwischen Mensch und Industrieroboter realisiert werden könnte, haben keine höhere Komplexität. Das gewählte Beispiel soll also stellvertretend für andere einfache Montageprozesse, wie z.B. „Handeinlegestationen“ stehen.

Im gewählten Beispiel müssen folgende vier Arbeitsaktionen ausgeführt werden:

„Schrauben 1“: In dieser Aktionsklasse muss eine Schraube an einer definierten Position am Motor festgezogen werden.

„Schrauben 2“: Hier wird eine weitere Schraube an einer anderen definierten Position am Motor festgezogen.

„Putzen“: In dieser Aktionsklasse muss ein definierter Bereich am Motor mit einem Handfeger gereinigt werden.

„Stecken“: In dieser Aktionsklasse muss der Werker ein zuvor gegriffenes Objekt an eine definierte Position am Motor stecken.

Diese vier Arbeitsaktionen sind die zu erkennenden Hauptbestandteile des zyklischen Arbeitsprozesses. Jede der Arbeitsaktionen ist einem definierten 3D-Objekt am Motor zugeordnet.

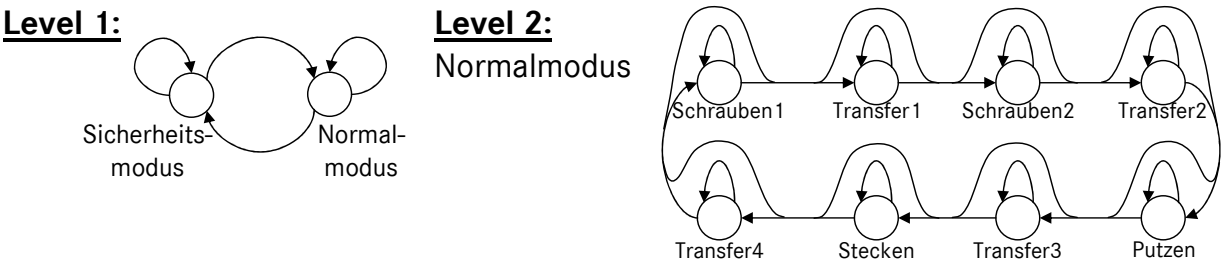

Abbildung 8.1: Zwei-Schichten-Architektur des entwickelten Aktionserkennungssystems. Da es als Sicherheitssystem realisiert ist, besteht es aus einer oberen und unteren Ebene. Die obere Ebene (Level 1) beschreibt den Zustand des Sicherheitssystems und die untere Ebene (Level 2) beschreibt den Zustand im vorgegebenen zyklischen Arbeitszyklus.

8.4 Systemmodell

Das entwickelte System zur Aktionserkennung von Arbeitsschritten bei der Motorenmontage basiert auf Algorithmen zur Trajektorienklassifikation und zum Trajektorienvergleich. Das verwendete Trackingsystem stellt einen kontinuierlichen Datenstrom von Bewegungsdaten eines oder mehrerer getrackter Objekte bereit. Die Bewegungsdaten bestehen dabei aus einer zeitlichen Abfolge von 3D-Positionen des in den Bildern des Kamerasystems verfolgten Objekts. Diese 3D-Punktfolgen werden als 3D-Trajektorien bezeichnet. Um eine mit dem Tracking schritthaltende Aktionserkennung zu gewährleisten, wird eine Sliding-Window-Technik verwendet. Diese macht eine sogenannte Online-Erkennung erst möglich, denn durch den kontinuierlichen Datenstrom ist der Beginn und das Ende einer Handlung nicht exakt vorgegeben. Daten über den zyklische Arbeitsablauf des Werkers sind im Aktionserkennungssystem verfügbar. Dieser kann aber immer wieder durch unbekannte Bewegungsmuster unterbrochen werden, z.B. wenn sich der Arbeiter die Nase putzt oder am Kopf kratzt.

Da das entwickelte Aktionserkennungssystem auch ein Sicherheitssystem ist, wird es als Zwei-Schichten-Architektur realisiert. Abbildung 8.1 zeigt die Struktur des entwickelten Aktionserkennungssystems.

Auf der höheren Ebene (Level 1, Abbildung 8.1 (links)) wird entschieden, ob sich das System im Sicherheits- oder Normalmodus befindet. Falls der Sicherheitsmodus aktiviert ist, wird ständig eine Vorhersage der Bewegung des Werkers durchgeführt, um mögliche Kollisionen mit dem Industrieroboter oder anderen Objekten zu erkennen. Somit kann im Gefahrfall der Roboter gezielt angehalten oder dessen Bewegung verlangsamt werden. Der Zustand im Level 1 der Systemarchitektur wird durch drei Trajektorienklassifikatoren bestimmt, diese werden im Abschnitt 8.7 näher erläutert.

Der Normalmodus (Level 2, Abbildung 8.1 (rechts)) beschreibt den vorgegebenen zyklischen Arbeitsablauf des Werkers, d.h. die Arbeitsschritte die er ausführen muss. Jede der Arbeitsaktionen ist einem definierten und auch bekannten 3D-Objekt am Motor zugeordnet. Wie in der Struktur des zyklischen Arbeitsablaufs (Level 2, Abbildung 8.1 (rechts)) zu erkennen ist, wird jede der Arbeitsaktionen durch Transferbewegungen erreicht. Diese sind wichtig, da man immer nur über Transferbewegungen die Arbeitsaktionen wechselt und auch aus einem unbekannten Bewegungszustand, z.B. „Haare kämmen“ oder „Nase schnäuzen“, über Transferbewegungen in den zyklischen Arbeitsablauf zurückkehrt. Transferbewegungen sind keinem Interaktionsobjekt zugeordnet.

Aus dem zyklischen Arbeitsablauf wird die Struktur eines HMMs abgeleitet. Das HMM wird nicht im klassischen Sinne trainiert und evaluiert, z.B. wie von Fink (2007) beschrieben. Vielmehr wird die Struktur direkt aus dem Arbeitsprozess abgeleitet und die Zustandsschätzung erfolgt durch einen Partikelfilter. Ein System zur partikelfilterbasierten Zustandsschätzung in einem HMM wurde bereits von Li u. a. (2006) vorgestellt. Neu ist, dass das HMM nichtstationäre Transitionswahrscheinlichkeiten besitzt. Dies wird realisiert, indem im Zustandsvektor des Partikelfilters neben dem aktuellen Hidden-State, welcher die Aktion beschreibt, auch die Phase der zum Hidden-State zugehörigen Referenztrajektorien geschätzt wird. Die Phase beschreibt dabei, zu welchem Anteil die zum Hidden-State zugehörigen Referenztrajektorien bereits durchlaufen sind. Eine hohe Phase, größer als 90%, bedeutet also, dass die Referenztrajektorie weitestgehend durchlaufen ist und somit die Transitionswahrscheinlichkeit, in den nächsten Zustand zu wechseln, erhöht ist. Die Zustandsschätzung im Level 2 wird detailliert im Abschnitt 8.8 beschrieben.

8.5 Vorverarbeitung der Trajektorien

In einem Vorverarbeitungsschritt werden Rauschen und Ausreißer in den 3D-Trajektorien, welche vom gewählten Trackingsystem geliefert werden, beseitigt. Zur Glättung der 3D-Trajektorien wird wie in der Arbeit von Croitoru u. a. (2005) ein Kalman-Filter mit konstanter Geschwindigkeit als Bewegungsmodell verwendet. Wie im Abschnitt 8.4 beschrieben, wird eine Sliding-Window-Technik eingesetzt, um eine Online-Erkennung zu ermöglichen. Zum Zeitschritt t ergibt sich die aktuelle Eingabetrajektorie im Sliding-Window $\mathbf{Z}_t$ mit

$$\mathbf{Z}_t = \left[(X_{(t-L_{\mathrm{SW}}+1)}, Y_{(t-L_{\mathrm{SW}}+1)}, Z_{(t-L_{\mathrm{SW}}+1)})^T,\ \ldots, (X_t, Y_t, Z_t)^T \right]. \tag{8.1}$$

Die Länge L_{SW} des Sliding-Windows beträgt dabei im vorgestellten System zur Aktionserkennung acht Zeitschritte. Somit werden immer nur die letzten $L_{\mathrm{SW}} = 8$ Zeitschritte eines mit dem Trackingsystem in den Bildern des Mehrkamerasystems verfolgten Objekts an die Trajektorienklassifikatoren (Level 1) und das Partikelfiltersystem (Level 2) zur Auswertung übergeben.

Da sich die Interaktionsobjekte, z.B. Motorenbereiche, die durch 3D-Positionen definiert werden, an unterschiedlichen Positionen im 3D-Weltkoordinatensystem W befinden können oder sich der Kamerastandpunkt verändern kann, muss eine Rotations- und Translationsinvarianz der Aktionserkennung realisiert werden. Es ist nicht praktikabel, das Erkennungssystem nach jeder Änderung der Kameraposition oder der Objektpositionen neu zu trainieren. Die Rotations- und Translationsinvarianz der Aktionserkennung wird erreicht, indem die

3D-Punkte im Sliding-Window so transformiert werden, dass die aktuelle 3D-Position der Interaktionsobjekte mit der 3D-Position der Interaktionsobjekte in den Trainingsdaten übereinstimmt. Die notwendige Rotationsmatrix und der Translationsvektor werden durch den ICP-Algorithmus (Besl u. McKay, 1992) berechnet. Die Transformation ist möglich, da im Szenario dieser Arbeit die Interaktionsobjekte verschiedenen Motorenbereichen mit definierten 3D-Positionen entsprechen und die relative Anordnung der 3D-Interaktionsobjekte zueinander somit immer fest ist.

8.6 Festlegung und Vorverarbeitung der Referenztrajektorien

Wie bereits beschrieben, gibt eine der Randbedingungen des Szenarios dieser Arbeit vor, dass nur wenige Trainigsdaten zur Aktionserkennung zur Verfügung stehen, d.h. die Größe der Lernstichprobe ist ungefähr einen Faktor zehn kleiner als die Größe des Testsets.

Manuell werden aus den gelabelten Trainingssequenzen Referenztrajektorien ausgeschnitten. Diese Referenztrajektorien beschreiben das Bewegungsverhalten der Hand des Werkers, während die Aktion ausgeführt wird, und dienen später beim Trajektorienvergleich als Aktionsprototypen. Da die Lernstichprobe klein ist und auch Aktionen mit unterschiedlichen Ausführungsgeschwindigkeiten erkannt werden müssen, wird eine zeitliche Skalierung der Referenztrajektorien durchgeführt. Unter Nutzung einer Akima-Interpolation (Akima, 1970) werden die Referenztrajektorien zeitlich von -20% bis $+20\%$ der Trajektoriengesamtlänge skaliert und die Menge der Referenztrajektorien somit erweitert. Dadurch können um bis zu 20% langsamere oder schnellere Bewegungen erkannt werden. Abbildung 8.2 (links) zeigt Referenztrajektorien der Aktionsklasse „Schrauben“. Abbildung 8.2 (rechts) stellt die zeitlich skalierten Trajektorien der blauen Referenztrajektorie (Abbildung 8.2 (links)) dar. Die Referenztrajektorien werden nun basierend auf ihren zugehörigen Aktionsklassen zu den Hidden-States im HMM (Abbildung 8.1 (rechts)) zugeordnet. Es gibt also für jeden Hidden-State des HMMs mehrere unterschiedliche Aktionsprototypen.

8.7 Trajektorienklassifikatoren (Level 1)

Im Level 1 des Aktionserkennungssystems, Abbildung 8.1 (links), wird entschieden, ob sich das System im Sicherheits- oder Normalmodus befindet. Falls der Sicherheitsmodus aktiviert ist, wird ständig eine Vorhersage der Bewegung des Werkers durchgeführt, um mögliche Kollisionen mit dem Industrieroboter oder anderen Objekten zu erkennen. Somit kann im Gefahrfall der Roboter gezielt angehalten oder die Bewegung verlangsamt werden.

Der Zustand im Level 1 der Zwei-Schichten-Architektur (Abbildung 8.1) wird direkt durch drei Trajektorienklassifikatoren bestimmt. Aber auch der Zustand des Partikelfiltersystems im Level 2 ist zum Teil abhängig von der Ausgabe der Trajektorienklassifikatoren. Abbildung 8.3 zeigt die geglättete Eingabetrajektorie eines getrackten Objekts (blau), die 3D-Punkte im aktuellen Sliding-Window (rot) und die Namen der drei Trajektorienklassifikatoren mit den sechs Ausgabeklassen. Die Ausgabe der drei Trajektorienklassifikatoren ist der

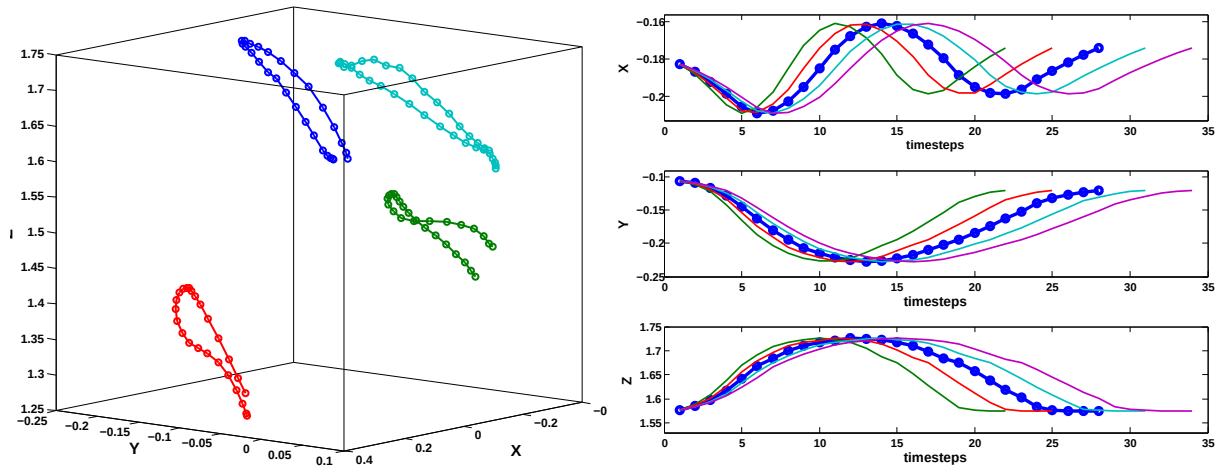

Abbildung 8.2: Links: Referenztrajektorien der Aktionsklasse „Schrauben“. Rechts: Zeitlich skalierte Trajektorien von -20% (Grün), -10%(Rot), ±0% (Blau, Original), +10% (Cyan), +20% (Magenta) der Gesamtlänge der blauen Referenztrajektorie in der linken Grafik. Die verschiedenen Ordinatenwerte sind in Millimeter dargestellt.

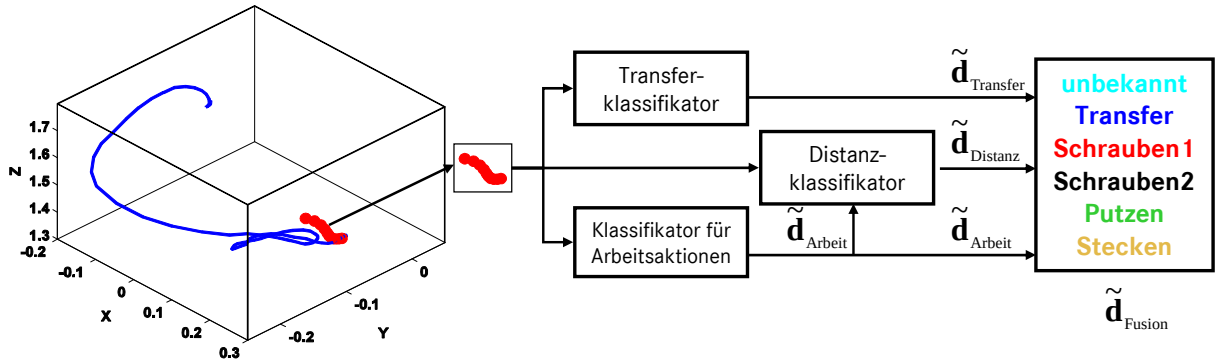

Abbildung 8.3: Geglättete Eingabetrajektorie eines getrackten Objekts (blau, durchgezogen), die 3D-Punkte im aktuellen Sliding-Window (rot, gepunktet) und die Namen der drei verwendeten Trajektorienklassifikatoren mit farblich definiertem Ausgabevektor.

fusionierte Diskriminanzvektor $\mathbf{d}_{\text{Fusion}}$. Dieser beinhaltet die Diskriminanzwerte für die sechs Klassen „unbekannte Bewegung“, „Transferbewegung“, „Schrauben 1“, „Schrauben 2“, „Putzen“ und „Stecken“. Die letzten vier Klassen sind die zu erkennenden Arbeitsaktionen und sind 3D-Interaktionsobjekten zugeordnet. Im angestrebten Szenario in dieser Arbeit sind die Interaktionsobjekte vorgegebene 3D-Punkte an einem Motorblock. An diesen Stellen muss eine zugeordnete Aktion ausgeführt werden. Vor dem Training der drei Trajektorienklassifikatoren werden die Trainingsbeispiele in ein einheitliches Koordinatensystem transformiert. Eine der zum Training der Klassifikatoren aufgezeichnete und gelabelte Sequenz definiert dabei das 3D-Koordinatensystem, indem die Trajektorienklassifikation durchgeführt wird. Alle weiteren Trainingssequenzen und auch später die Testsequenzen werden so transformiert, dass die 3D-Positionen der Interaktionsobjekte übereinstimmen. Damit wird eine Rotations- und Translationsinvarianz der Aktionserkennung erreicht. Die notwendige Rotationsmatrix und der Translationsvektor werden durch den ICP-Algorithmus (Besl u. McKay, 1992) berechnet.

8.7.1 Transferklassifikator

Durch diesen Klassifikator werden Transferbewegungen erkannt. Diese Transferbewegungen sind wichtig, da man immer nur über eine Transferbewegung die Arbeitsaktionen wechselt und auch aus einer unbekannten Bewegung über eine Transferbewegung in den zyklischen Arbeitsablauf zurückkehrt. Transferbewegungen sind keinem Interaktionsobjekt zugeordnet. Sie werden häufig schnell und geradlinig ausgeführt.

Der im Transferklassifikator verwendete Standardklassifikator ist ein quadratischer Polynomklassifikator (Schürmann, 1996). Zur Klassifikation werden zwei Merkmale verwendet: (i) der zurückgelegte Weg im Sliding-Window und (ii) der maximale Winkel zwischen zwei aufeinander folgenden Richtungsvektoren im Sliding-Window. Durch die Merkmale wird ausgenutzt, dass Transferbewegungen häufig geradlinig sind und es keine oder nur wenige Richtungsänderungen gibt, wie sie aber in den anderen Klassen häufig vorhanden sind. Pro Klasse gibt es mindestens 300 Merkmalsvektoren, sodass die Berechnung des quadratischen Polynomklassifikators mit genügend Beispielen unterlegt ist.

Die zum Training verwendeten Merkmale sind in Abbildung 8.4 (links) dargestellt. Es ist gut zu erkennen, dass sich die Transferbewegungen (blau) gut von den anderen Klassen absetzen, d.h. die Trennung der Klassen gut möglich ist. Die Ausgabe des Transferklassifikators ist der Diskriminanzvektor $\mathbf{d}_{\text{Transfer}}$, dieser trennt zwischen den zwei Klassen „Transferbewegung“ und „keine Transferbewegung“. Durch eine Normalisierung, bei der Diskriminanzwerte größer Eins und kleiner Null abgeschnitten werden und anschließend die Diskriminanzwerte so angepasst werden, dass deren Summe Eins ergibt, erhält man den Diskriminanzvektor

$$\tilde{\mathbf{d}}_{\text{Transfer}} = \begin{pmatrix} \tilde{d}_{\text{Transfer}} \\ 1 - \tilde{d}_{\text{Transfer}} \end{pmatrix}, \tag{8.2}$$

dessen Diskriminanzwerte im Intervall $[0, 1]$ liegen.

8.7.2 Klassifikator für die Arbeitsaktionen

Dieser Klassifikator trennt zwischen den vier Arbeitsaktionen „Schrauben 1“, „Schrauben 2“, „Putzen“ und „Stecken“. Diese sind die zu erkennenden Hauptbestandteile des zyklischen Arbeitsprozesses. Jede der Arbeitsaktionen ist einem definierten 3D-Interaktionsobjekt am Motor zugeordnet.

Wie schon beim Transferklassifikator, wird auch hier ein quadratischer Polynomklassifikator (Schürmann, 1996) zur Trennung der vier Arbeitsaktionsklassen eingesetzt. Zur Klassifikation werden vier Merkmale verwendet. Diese sind die minimale euklidische Distanz im Sliding-Window zu den vier aktionsspezifischen 3D-Interaktionsobjekten. Experimentell hat sich gezeigt, dass ein quadratischer Polynomklassifikator bei der kleinen Lernstichprobe im angestrebten Szenario dieser Arbeit Vorteile bei der Klassentrennung gegenüber einem Mahalanobis-Distanzklassifikator (Schürmann, 1996) besitzt. Die Klassengrenzen sind beim Polynomklassifikator nach außen hin geöffnet, daher ist diese Art der Klassifikation genereller und nicht so restriktiv wie ein Mahalanobis-Distanzklassifikator. Durch den quadratischen Polynomklassifikator können auch Klassen erkannt werden, die mit einer unterschiedlichen Handstellung oder in andere Richtungen ausgeführt werden. Die Ausgabe des Klassifikators für Arbeitsaktionen

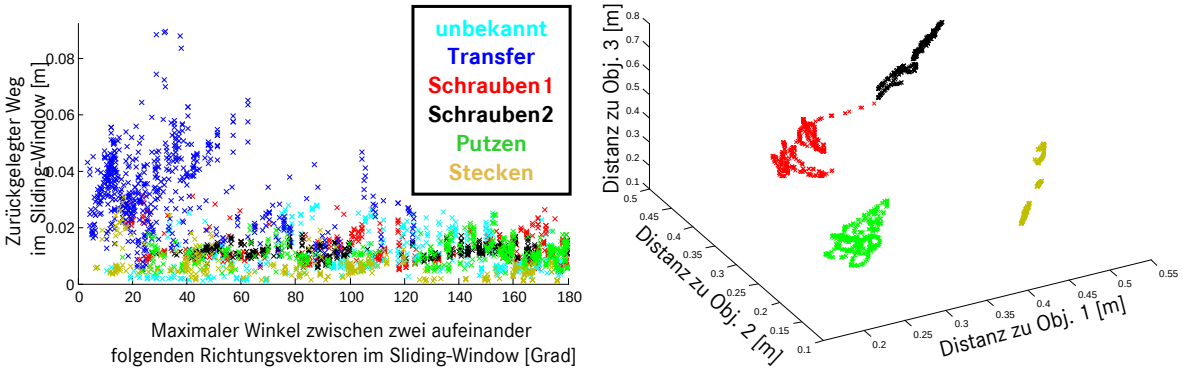

Abbildung 8.4: Darstellung der Trainingsbeispiele des Transferklassifikators (links) und des Klassifikators für Arbeitsaktionen (rechts). Beim Klassifikator für Arbeitsaktionen sind nur die ersten drei Merkmale dargestellt.

ist der Diskriminanzvektor $\mathbf{d}_{\text{Arbeit}}$, dieser trennt zwischen den vier Arbeitsaktionen „Schrauben 1“, „Schrauben 2“, „Putzen“ und „Stecken“. Durch eine Normalisierung erhält man den Diskriminanzvektor

$$\tilde{\mathbf{d}}_{\text{Arbeit}} = \begin{pmatrix} \tilde{d}_{\text{Schrauben1}} \\ \tilde{d}_{\text{Schrauben2}} \\ \tilde{d}_{\text{Putzen}} \\ \tilde{d}_{\text{Stecken}} \end{pmatrix}, \tag{8.3}$$

dessen Diskriminanzwerte liegen im Intervall $[0, 1]$. Wiederum werden Diskriminanzwerte größer Eins und kleiner Null abgeschnitten und anschließend die Diskriminanzwerte so angepasst, dass deren Summe Eins ergibt. Pro Klasse gibt es mindestens 80 Merkmalsvektoren im Trainingsdatensatz, damit stehen ausreichend viele Beispiele zur Berechnung des quadratischen Polynomklassifikators zur Verfügung. Bei vier Merkmalen und einem vollständig quadratischen Polynomklassifikator müssen 15 Koeffizienten pro Klasse optimiert werden. Die Erfahrungen von Schürmann (1996) haben gezeigt, dass mindestens doppelt so viele Trainingsbeispiele wie zu optimierende Koeffizienten zur Verfügung stehen sollten. Abbildung 8.4 (rechts) zeigt die ersten drei Merkmale der Trainingsdaten des Klassifikators für Arbeitsaktionen. Es ist zu erkennen, dass sich die Klassen visuell gut trennen lassen.

8.7.3 Distanzklassifikator

Da die Klassengrenzen im Merkmalsraum des verwendeten Polynomklassifikators im Klassifikator für Arbeitsaktionen nach außen hin geöffnet sind, wird ein einfacher nachgeschalteter Distanzklassifikator benutzt, um zu entscheiden ob die Bewegung noch in der Nähe eines Interaktionsobjekts ausgeführt wird oder nicht. Der Distanzklassifikator ist dabei abhängig von der Ausgabe des Klassifikators für Arbeitsaktionen (Abbildung 8.3). Der maximale Diskriminanzwert der Ausgabe $\tilde{\mathbf{d}}_{\text{Arbeit}}$ des Klassifikators für Arbeitsaktionen bestimmt dabei die getestete Aktionsklasse und somit auch das 3D-Interaktionsobjekt. Zu diesem wird die minimale euklidische Distanz im Sliding-Window berechnet. Der berechnete Distanzwert wird über eine gaußsche Wahrscheinlichkeitsdichtefunktion gewichtet. Die Parameter der gaußschen Wahrscheinlichkeitsdichtefunktion werden dabei aus den Trainingsdaten bestimmt. Der Di-

stanzklassifikator trennt die Klassen „bekannte Bewegung“ und „unbekannte Bewegung“ und hat als Ausgabe den normierten Diskriminanzvektor

$$\tilde{\mathbf{d}}_{\text{Distanz}} = \begin{pmatrix} \tilde{d}_{\text{bekannt}} \\ 1 - \tilde{d}_{\text{bekannt}} \end{pmatrix}, \tag{8.4}$$

wiederum liegen dessen Diskriminanzwerte im Intervall $[0, 1]$ und die Summe aller Diskriminanzwerte ergibt Eins.

8.7.4 Kombination der Klassifikatoren und Zustandsschätzung

Durch Kombination der Diskriminanzvektoren $\tilde{\mathbf{d}}_{\text{Transfer}}, \tilde{\mathbf{d}}_{\text{Arbeit}}, \tilde{\mathbf{d}}_{\text{Distanz}}$ wird die fusionierte Ausgabe $\mathbf{d}_{\text{Fusion}}$ der drei Trajektorienklassifikatoren berechnet. Diese beinhaltet die Diskriminanzwerte für die sechs Klassen „unbekannte Bewegung“, „Transferbewegung“, „Schrauben 1“, „Schrauben 2“, „Putzen“ und „Stecken“. Die Kombination der Klassifikatoren erfolgt durch Gleichung (8.5). Diese wurde durch Expertenwissen modelliert und durch Experimente mit verschiedenen Kombinationsvarianten als sinnvollste Kombination bestätigt. In Gleichung (8.5) wird berücksichtigt, dass nur eine bekannte Bewegung, also in der Nähe der Interaktionsobjekte, eine Transferbewegung oder eine der Aktionsklassen sein kann. Außerdem kann eine der Aktionsklassen nur dann einen hohen Diskriminanzwert erreichen, wenn keine Transferbewegung erkannt wurde.

$$\mathbf{d}_{\text{Fusion}} = \begin{pmatrix} d_{\text{unbekannt}} \\ d_{\text{Transfer}} \\ d_{\text{Schrauben1}} \\ d_{\text{Schrauben2}} \\ d_{\text{Putzen}} \\ d_{\text{Stecken}} \end{pmatrix} = \begin{pmatrix} 1 - \tilde{d}_{\text{bekannt}} \\ \tilde{d}_{\text{bekannt}} \cdot \tilde{d}_{\text{Transfer}} \\ \tilde{d}_{\text{bekannt}} \cdot (1 - \tilde{d}_{\text{Transfer}}) \cdot \tilde{d}_{\text{Schrauben1}} \\ \tilde{d}_{\text{bekannt}} \cdot (1 - \tilde{d}_{\text{Transfer}}) \cdot \tilde{d}_{\text{Schrauben2}} \\ \tilde{d}_{\text{bekannt}} \cdot (1 - \tilde{d}_{\text{Transfer}}) \cdot \tilde{d}_{\text{Putzen}} \\ \tilde{d}_{\text{bekannt}} \cdot (1 - \tilde{d}_{\text{Transfer}}) \cdot \tilde{d}_{\text{Stecken}} \end{pmatrix} \tag{8.5}$$

Durch eine Normalisierung von $\mathbf{d}_{\text{Fusion}}$ erhält man das endgültige Fusionsergebnis $\tilde{\mathbf{d}}_{\text{Fusion}}$ der drei Trajektorienklassifikatoren, die Summe aller Diskriminanzwerte ergibt 1 und die einzelnen Diskriminanzwerte liegen im Intervall $[0, 1]$.

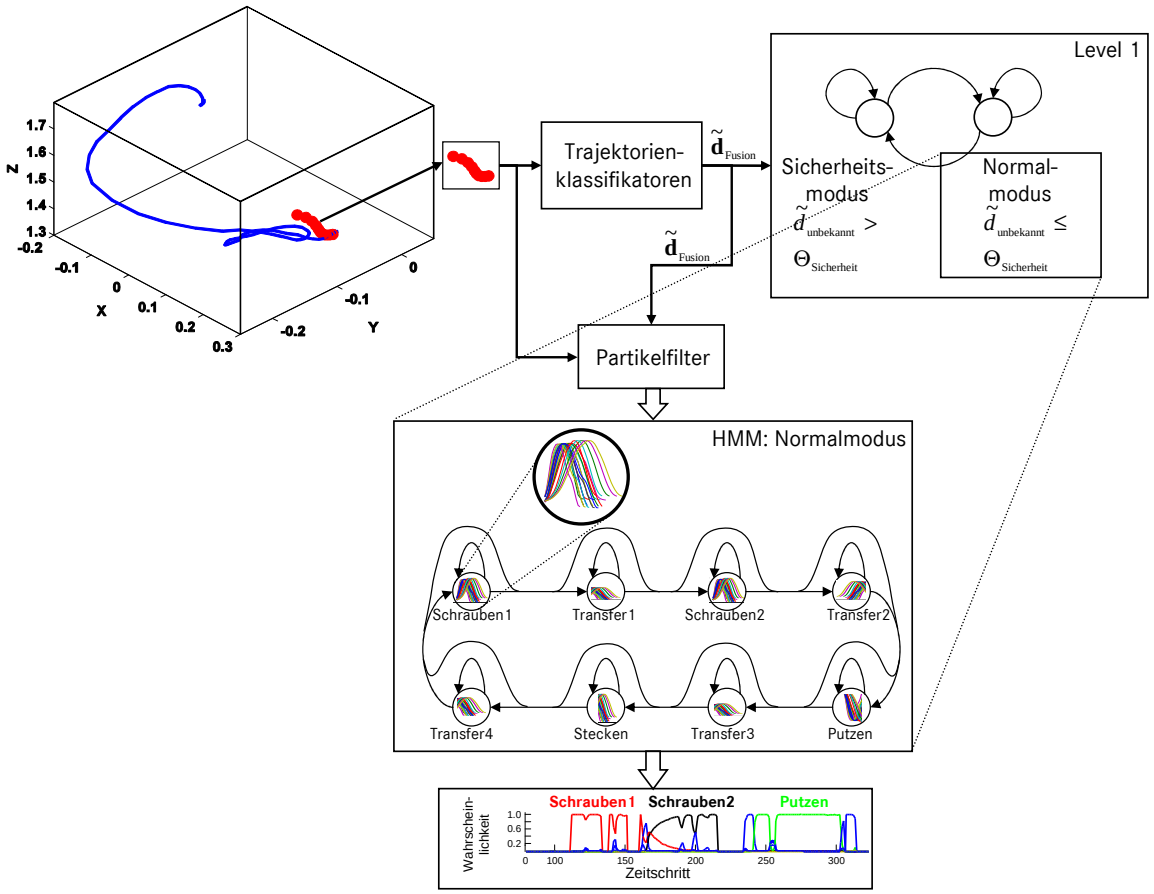

Abbildung 8.5: Ablauf der Zustandsschätzung in der Zwei-Schichten-Architektur des Sicherheitssystems. Der Zustand im Level 1 ergibt sich direkt $\tilde{\mathbf{d}}_{\text{Fusion}}$ dem Ergebnis der Trajektorienklassifikatoren. Die Zustandsschätzung im Normalmodus (Level 2) wird durch ein Partikelfilter durchgeführt, dabei wird neben dem aktuellen Zustand im HMM des Normalmodus, auch die aktuell beste Referenztrajektorie (Aktionsprototyp) geschätzt. Die Ausgabe des Partikelfilters sind Zustandswahrscheinlichkeiten für jeden Zeitschritt, welche zur eigentlichen Aktionserkennung verwendet werden.

Der Zustand im Level 1 der Zwei-Schichten-Architektur wird nun direkt durch das Fusionsergebnis $\tilde{\mathbf{d}}_{\text{Fusion}}$ der Trajektorienklassifikatoren bestimmt. Wenn der Diskriminanzwert für eine unbekannte Bewegung $\tilde{d}_{\text{unbekannt}}$ eine Schwelle $\Theta_{\text{Sicherheit}}$ überschreitet, geht das Aktionserkennungssystem in den Sicherheitsmodus. Falls nicht, befindet sich das System im Normalmodus, dem eigentlichen Arbeitsprozess. Die Zustandsschätzung im Normalmodus wird durch einen Partikelfilter durchgeführt. Dieser schätzt den aktuellen Zustand in einem HMM, welches den zyklischen Arbeitsablauf beschreibt. Wenn sich das Aktionserkennungssystem im Sicherheitsmodus befindet, werden die Iterationsschritte des Partikelfilters nicht durchgeführt. Abbildung 8.5 zeigt den Ablauf der Zustandsschätzung in der Zwei-Schichten-Architektur des Sicherheitssystems.

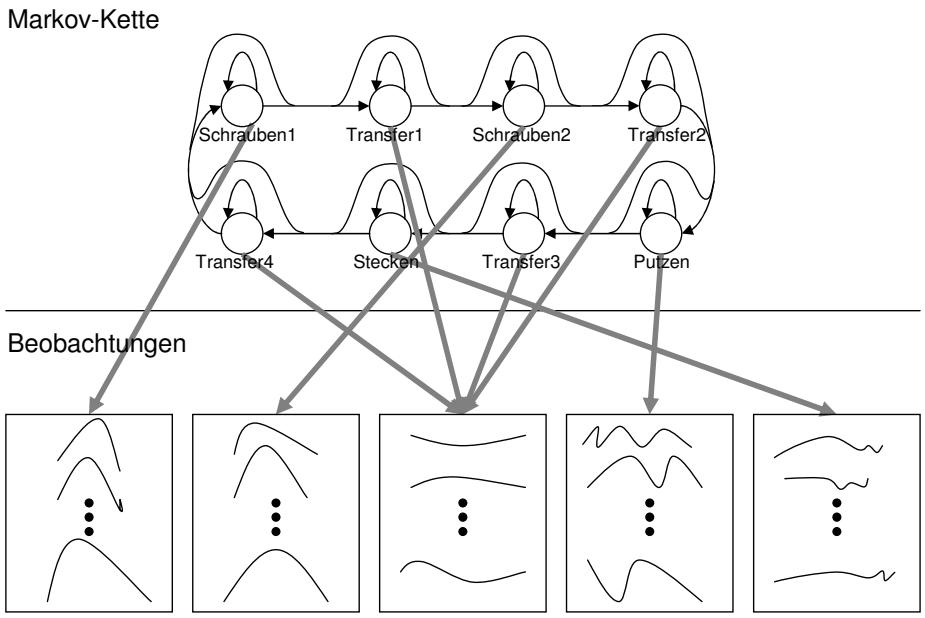

Abbildung 8.6: Markov-Kette des Produktionsprozesses mit zugehörigen Beobachtungen, welche durch den Vergleich und die Klassifikation der aktuellen Input-Trajektorie im Sliding-Window mit Referenzbewegungen gemacht werden können. Die Position innerhalb der Referenztrajektorie ergibt sich aus der geschätzten Bewegungsphase.

8.8 Zustandsschätzung im Normalmodus (Level 2)

Der Normalmodus (Level 2) beschreibt den vorgegebenen zyklischen Arbeitsablauf des Werkers und ist als Markov-Kette modelliert. Abbildung 8.5 zeigt den Ablauf der Zustandsschätzung in der Zwei-Schichten-Architektur des Sicherheitssystems. Falls sich das System im Normalmodus befindet, erfolgt die Zustandsschätzung durch einen neuartigen Ansatz, bei dem ein nichtstationäres HMM mit einem Partikelfilter kombiniert wird.

HMMs sind stochastische Modelle, welche sich durch zwei Zufallsprozesse beschreiben lassen. Der erste Prozess ist eine Markov-Kette, dessen Zustände von außen nicht direkt beobachtbar sind. Der zweite Zufallsprozess erzeugt gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung die für das HMM benötigten beobachtbaren Ausgangssymbole, Abbildung 8.6 verdeutlicht dies. HMMs sind endliche stochastische Automaten und eignen sich sehr gut um den zyklischen Prozess der Arbeitsaktionen zu modellieren. Da nur wenige Trainingsdaten vorhanden sind, wird das HMM nicht im klassischen Sinne trainiert und evaluiert (Fink, 2007). Die Struktur der Markov-Kette wird direkt aus dem Arbeitsprozess abgeleitet und die Zustandsschätzung erfolgt durch einen Partikelfilter. Neu ist, dass es sich dabei um ein nichtstationäres HMM handelt. Dies wird realisiert, indem im Zustandsvektor des Partikelfilters neben dem aktuellen Hidden-State auch die Phase der Bewegung geschätzt wird und die Transitionswahrscheinlichkeiten abhängig von dieser Bewegungsphase sind. Das verwendete nichtstationäre HMM (Abbildung 8.1 (rechts)) zur stochastischen Beschreibung des Normalmodus ist definiert durch $\lambda = (S, A, B, \Pi)$:

– $S = \{q_1, \ldots, q_n\}$, die Menge alle Zustände (Hidden-States);

- $A = \{a_{ij,t} | a_{ij,t} = P(q_t = s_j | q_{t-\Delta t} = s_i)\}$, die nichtstationäre – von der Bewegungsphase ϕ_t abhängige – Zustandsübergangswahrscheinlichkeit von Zustand s_i nach s_j;
- $B = \{b_{i,k} | b_{i,k} = P(o_t = v_k | q_t = s_i)\}$, die Wahrscheinlichkeit im Hidden-State s_i den Zustand v_k zu beobachten;
- $\Pi = \{\pi_i | \pi_i = P(q_1 = s_i)\}$, Anfangswahrscheinlichkeit des Zustands s_i.

Die im Abschnitt 8.6 beschriebenen Referenztrajektorien ermöglichen durch den Vergleich mit der Input-Trajektorie die Beobachtungen im nichtstationären HMM (Abbildung 8.6). Die Referenztrajektorien sind sogenannte Aktionsprototypen und beschreiben das Bewegungsverhalten der Hand des „Lehrers“, Meisters oder Vorarbeiters während dieser die Aktion ausführt.

Die Schätzung des aktuellen Zustands im nichtstationären HMM, bezeichnet als aktueller Hidden-State, wird durch einen Partikelfilter durchgeführt. Ein Partikelfilter ist ein etabliertes Verfahren zur probabilistischen Zustandsschätzung. Es werden statistische oder physikalische Modelle verwendet, um Schätzungen aufzustellen, welche anschließend mit Beobachtungen verglichen werden. Unter Berücksichtigung der Beobachtungen wird die Schätzung verbessert. In einem Partikelfilter ist der Zustand des beobachteten Systems zu einem bestimmten Zeitschritt kein exakter Wert, sondern wird durch die Funktion der Wahrscheinlichkeitsdichte, engl. *probability density function* (pdf), abgebildet. Doucet u. a. (2001) geben einen detaillierten Überblick über die unterschiedlichen Partikelfilter-Varianten. Der CONDENSATION-Algorithmus nach Isard u. Blake (1998) stellt eine weit verbreitete Implementierung eines Partikelfilters dar. Diese Variante wird von Black u. Jepson (1998) zur trajektorienbasierten Gestenerkennung verwendet. Ein Vorteil des Verfahrens liegt in der automatischen Konzentration auf relevante Teile und der geringen Anzahl an Trainingsbeispielen. Li u. a. (2006) schätzen damit den Zustand von HMMs. Auch in dieser Arbeit wird der CONDENSATION-Algorithmus verwendet.

Der geschätzte Zustand zum diskreten Zeitschritt t wird mit $\mathbf{x}_t$ bezeichnet. Die Historie der Zustandsschätzung mit $\mathcal{X}_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_t\}$. In gleicher Weise werden die Beobachtungen $\mathbf{z}_t$ und deren Historie mit $\mathcal{Z}_t = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$ bezeichnet. Durch die Markov-Annahme hängt der aktuelle Zustand nur vom Vorherigen ab und es gilt $p(\mathbf{x}_t | \mathcal{X}_{t-1}) = p(\mathbf{x}_t | \mathbf{x}_{t-1})$. Die Beobachtungen $\mathbf{z}_t$ – also die 3D-Input-Trajektorien im Sliding-Window – werden als unabhängig angenommen. Das Beobachtungsmodell zum Zeitschritt t wird ausgedrückt durch die Wahrscheinlichkeitsdichte $p(\mathbf{z}_t | \mathbf{x}_t)$. Die a-posteriori Wahrscheinlichkeitsdichte $(\mathbf{x}_t | \mathbf{z}_t)$ ergibt sich nach Isard u. Blake (1998) zu:

$$p(\mathbf{x}_t | \mathbf{z}_t) = k p(\mathbf{z}_t | \mathbf{x}_t) p(\mathbf{x}_t). \tag{8.6}$$

Wobei k für eine Normalisierungskonstante steht, welche unabhängig von $\mathbf{x}_t$ ist und $p(\mathbf{x}_t)$ für die a-priori Wahrscheinlichkeit im Zustand $\mathbf{x}_t$ zu sein. Der CONDENSATION-Algorithmus basiert auf der Methode des „*factored sampling*“ und approximiert die a-posteriori Wahrscheinlichkeitsdichte $p(\mathbf{x}_t | \mathbf{z}_t)$ durch eine relativ kleine Anzahl von N Partikeln (Samples) $s_t^{(i)}$:

$$p(\mathbf{x}_t | \mathbf{z}_t) \propto \left\{ s_t^{(i)} = \left\langle \mathbf{x}_t^{(i)}, w_t^{(i)} \right\rangle | i = 1, \ldots, N \right\}. \tag{8.7}$$

Die Approximation basiert auf der a-priori Wahrscheinlichkeitsdichte $p(\mathbf{x}_t | \mathbf{z}_{t-1})$, diese ist im Allgemeinen multimodal und wird abgeleitet aus dem Set der Partikel des vorherigen Zeitschritts. Dabei werden die Partikel durch das Bewegungsmodell prädiziert.

Jedes Partikel $s_t^{(i)}$ wird durch einen Zustand $\mathbf{x}_t^{(i)}$ und ein Gewicht $w_t^{(i)}$ charakterisiert. Der Zustandsvektor $\mathbf{x}_t^{(i)}$ eines Partikels $s_t^{(i)}$ im entwickelten Aktionserkennungssystem besteht aus drei Elementen:

$$\mathbf{x}_t^{(i)} = (q_t^{(i)},\ \phi_t^{(i)},\ id_t^{(i)}). \tag{8.8}$$

Dabei steht $q_t^{(i)}$ für den aktuell geschätzten Hidden-State im HMM des Normalmodus. Mit $\phi_t^{(i)}$ wird die aktuelle Phase innerhalb einer zum Hidden-State zugeordneten Referenztrajektorie, also eines Aktionsprototypen, bezeichnet. Die Phase $\phi_t^{(i)}$ mit $\phi_{min} \leq \phi_t^{(i)} \leq 1$ beschreibt dabei, zu welchem Anteil die zur Aktion, beschrieben durch den Hidden-State, zugehörige Referenztrajektorie durchlaufen ist. Der Wert ϕ_{min} ist abhängig von der zeitlichen Länge des Sliding-Windows und der Referenztrajektorie. Außerdem wird der Index $id_t^{(i)}$ des relevanten Objekts aus allen getrackten Objekten geschätzt. Das relevante Objekt ist das getrackte Objekt, welches die zu erkennenden Aktionen ausführt, z.B. die Hand des Werkers. Die Schätzung des Index des relevanten Objekts ist notwendig, wenn wie beim 3D-Mean-Shift-Tracking (Kapitel 5) mehrere Objekte zeitgleich verfolgt werden, ohne zu wissen, welches der Objekte die zu erkennende Aktion ausführt.

Die zur Gewichtsberechnung des Partikelfilters notwendigen Beobachtungen $\mathbf{z}_t$ sind, wie in Abbildung 8.5 dargestellt, die aktuelle Trajektorie im Sliding-Window und die Ausgabe der $\tilde{\mathbf{d}}_{\text{Fusion}}$ der Trajektorienklassifikatoren. Im Folgenden wird der Ablauf des Partikelfiltersystems beschrieben. Zum ersten Zeitschritt werden die Partikel gemäß der definierten Anfangswahrscheinlichkeit Π in das HMM gestreut. Zu allen weiteren Zeitschritten besteht ein Iterationsschritt aus den drei aufeinander folgenden Schritten Selektion, Prädiktion und Aktualisierung.

Selektion/Resampling: Als Erstes wird eine neue, noch ungewichtete Samplemenge durch Ziehen mit Zurücklegen aus der alten Partikelmenge erzeugt. Es werden $N - M$ Partikel $s_{t-\Delta t}^{(i)}$ aus der Partikelmenge des Zeitschritts $t - \Delta t$ aufgrund ihres entsprechenden Gewichts $w_{t-\Delta t}^{(i)}$ gezogen. Das Resampling ermöglicht, hoch gewichtete Partikel öfter auszuwählen und niedrig gewichtete Partikel zu vernachlässigen. Außerdem werden M Partikel zufällig über alle Hidden-States des HMMs verstreut.

Prädiktion: Im zweiten Schritt werden die gezogenen Partikel der Menge s_t mit dem Transitionsmodell bewegt. Dazu wurde ein Modell der Bewegung eines Partikels durch die Topographie des HMMs entwickelt, bei dem Zustandsübergänge nur vom aktuellen Hidden-State auf den nächsten und übernächsten Hidden-State möglich sind. Die Partikelbewegung ist abhängig von der aktuelle Phase $\phi_t^{(i)}$ und von t_{Aktion}, der Zeit, seit der sich das System im aktuellen Hidden-State befindet. Dies impliziert, dass es nichtstationäre Transitionswahrscheinlichkeiten im HMM gibt.

Aktualisierung: Hier erfolgt die eigentliche Messung, mit dem Beobachtungsmodell wird für jedes Partikel ein normiertes Gewicht $w_t^{(i)}$ bestimmt. Im Beobachtungsmodell des entwickelten Aktionserkennungssystems werden alle durch ein Partikel definierten Referenztrajektorien mit der aktuellen Input-Trajektorie im Sliding-Window des zum Partikel zugeordneten Objekts $id_t^{(i)}$ verglichen, wobei es für jeden Hidden-State mehrere Referenztrajektorien gibt. Dabei wird ein Maß zum Trajektorienvergleich zwischen zwei Trajektorien angewandt, welches auf der Levenshtein-Distanz von Trajektorien (LDT) basiert. Außerdem wird die Ausgabe der Trajektorienklassifikatoren $\tilde{\mathbf{d}}_{\text{Fusion}}$ in die Ge-

wichtsberechnung einbezogen.

Nach den drei Schritten des Partikelfilters wird die Historie der Schätzungen des Partikelfilters analysiert, um kontinuierlich Aktionen erkennen zu können. Im Folgenden werden die drei Schritte des Partikelfilters im Aktionserkennungssystem detailliert beschrieben und anschließend das Modul zur kontinuierlichen Erkennung von Aktionen vorgestellt.

8.8.1 Selektion/Resampling

Als Erstes wird eine neue, noch ungewichtete Partikelmenge durch Ziehen mit Zurücklegen aus der alten Partikelmenge erzeugt. Es werden $N - M$ Partikel $s^{(i)}_{t-\Delta t}$ aus der Partikelmenge des Zeitschritts $t - \Delta t$ aufgrund ihres entsprechenden Gewichts $w^{(i)}_{t-\Delta t}$ gezogen. Für jedes Partikel wurde zuvor zusätzlich der Wert

$$c^{(i)}_t = \sum_{j=1}^{i} w^{(j)}_t, \tag{8.9}$$

gespeichert. Nun wird $N - M$ mal eine auf $[0, 1]$ gleichverteilte Zufallszahl r bestimmt und durch binäre Suche dasjenige Partikel mit dem kleinsten j ausgewählt, für das gilt:

$$c^{(i)}_t \geq r. \tag{8.10}$$

Dieser Schritt sorgt für die Resampling-Prozedur, welche Partikel mit hoher Gewichtung mehrmals auswählt und solche mit niedriger Gewichtung vernachlässigt. Dadurch wird ermöglicht, den Resamplingschritt in einer Laufzeit von $\mathcal{O}(n \log(n))$ statt in $\mathcal{O}(n^2)$ durchzuführen. Das Resampling ermöglicht es, hoch gewichtete Partikel öfter auszuwählen und niedrig gewichtete Partikel zu vernachlässigen.

Außerdem werden M Partikel zufällig über alle Zustände des HMMs verstreut, also neu initialisiert. Bei der Initialisierung eines Partikels wird der Hidden-State $q^{(i)}_t$ zufällig aus der Menge aller Hidden-States S ausgewählt. Die Phase $\phi^{(i)}_t$ wird mit ϕ_{min} initialisiert und der notwendige Objektindex $id^{(i)}_t$ des geschätzten relevanten Objekts wird zufällig aus allen aktuell getrackten Objekten ausgewählt. Dies ist die einzige Möglichkeit, neue Objekte in die Schätzung einfließen zu lassen. Ein zufällig ausgewähltes, aber nicht relevantes Objekt wird sehr schnell wieder verworfen, da das Beobachtungsmodell im Aktualisierungsschritt des Partikelfilters im Vergleich zu relevanten Objekten nur ein sehr geringes Gewicht ausgibt. Das zufällige Streuen der Partikel macht das entwickelte System robuster und ist die Grundlage dafür, dass auch ein Überspringen mehrerer Aktionen möglich ist. Weiterhin kann auch ein Verlassen und Wiedereintreten in verschiedenen Hidden-States des Normalmodus dadurch behandelt werden.

8.8.2 Prädiktion

In diesem Schritt werden die im Selektionsschritt gezogenen Partikel gemäß einem Bewegungsmodell durch das HMM prädiziert. Dazu wurde ein Modell der Bewegung eines Partikels durch die Topographie des HMMs entwickelt. Neuartig an dem entwickelten System zur partikelfilterbasierten Zustandsschätzung ist die Schätzung der Phase ϕ einer Referenztrajektorie eines

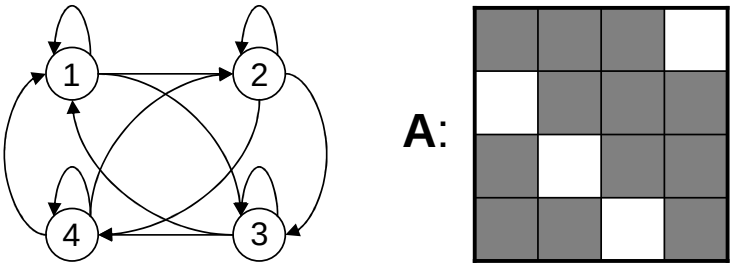

Abbildung 8.7: Links: Beispiel für die Topologie eines zyklischen HMMs, bei dem Zustandsübergänge nur vom aktuellen Hidden-State auf den nächsten und übernächsten Hidden-State möglich sind. Rechts: Matrix der Transitionswahrscheinlichkeiten A des HMMs, ein graues Feld steht für einen vorhandenen Wert, ein weißes Feld entspricht dem Wert Null.

Hidden-States. Dadurch wird die Verwendung von nichtstationären Transitionswahrscheinlichkeiten plausibel, denn bei einer geringen Phase ist ein Zustandsübergang sehr unwahrscheinlich. Eine hohe Phase hingegen bedeutet, dass die zugehörige Referenztrajektorie weitestgehend durchlaufen und die Transitionswahrscheinlichkeit in den nächsten Zustand zu wechseln erhöht ist.

Die Randbedingungen des Szenarios dieser Arbeit (Abschnitt 8.1) sehen vor, dass nur wenige Trainigsdaten zum Belernen der Aktionserkennung zur Verfügung stehen. Nur aufgezeichnete und mit Aktionsklassen gelabelte Bewegungsdaten weniger Vorarbeiter, welche den Arbeitsprozess detailliert kennen, können als Grundlage für das Erkennungssystem verwendet werden. Aufgrund der kleinen Lernstichprobe werden die phasen- und zeitabhängigen Transitionswahrscheinlichkeiten nicht aus den Daten gelernt, sondern durch Expertenwissen modelliert. Ein Training mit einer derart geringen Menge an Trainingsdaten würde ein „Auswendiglernen“ bedeuten.

Die gewählte Modellierung sieht vor, dass Zustandsübergänge nur vom aktuellen Hidden-State auf den nächsten und übernächsten Hidden-State möglich sind. Ein Beispiel für die Topologie eines zyklischen HMMs mit derartigen Zustandsübergängen ist in Abbildung 8.7 dargestellt. Der Ansatz ist ähnlich zu Bakis-Modell (Fink, 2007) für Links-Rechts-HMMs. Diese HMM-Topologie ermöglicht es, Zustände zu überspringen, damit wird die Flexibilität des Erkennungssystems erhöht. Diese spezielle Art der Modellierung begründet sich im Sliding-Window-Ansatz, denn es ist möglich, dass Aktionen nicht erkannt werden. Bei Aktionen, deren Dauer in etwa der zeitlichen Länge des Sliding-Windows entspricht, ist dies durchaus wahrscheinlich. Die nichtstationären Transitionswahrscheinlichkeiten werden zu jedem Zeitschritt über das eingesetzte Bewegungsmodell neu berechnet. Die Transitionswahrscheinlichkeit ist abhängig von der aktuelle Phase ϕ innerhalb der Referenztrajektorie und von t_{Aktion}, der Zeit, die das System im aktuellen Hidden-State bisher verbracht hat. Die Transitionswahrscheinlichkeiten ergeben sich aus den Anteilen der Partikel, die über die Topologie des HMMs prädiziert werden.

Prädiktion der Phase

Als Erstes wird die Phase jedes zuvor selektierten Partikels prädiziert, da die Transitionswahrscheinlichkeit eines Zustandsübergangs im HMM davon abhängig ist. Für jedes selektierte

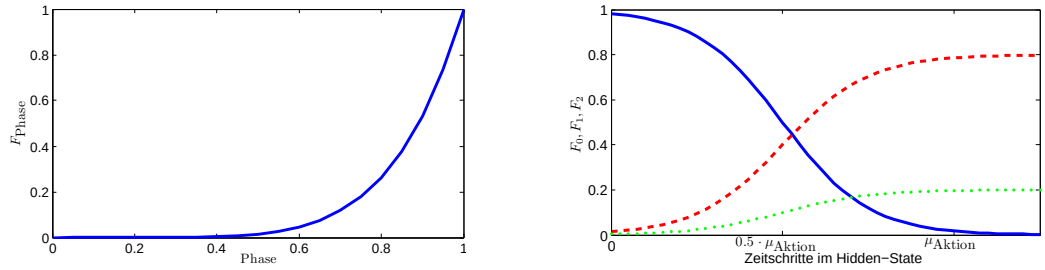

Abbildung 8.8: Links: Funktion $F_{Phase}(\phi)$ für $\gamma_1 = 6$, auch in den Experimenten wird dieser Wert gewählt. Mit zunehmender Phase werden immer mehr Partikel über das eigentliche Zustandsübergangsmodell prädiziert. Rechts: Partikelanteile für den aktuellen Hidden-State $F_0(t_{\text{Aktion}})$ (blau, durchgezogen) sowie die Anteile $F_1(t_{\text{Aktion}})$ (rot, gestrichelt) und $F_2(t_{\text{Aktion}})$ (grün, gepunktet).

Partikel $s_t^{(i)}$ berechnet sich das Update der Phase zu

$$\phi_t^{(i)} = \phi_{t-\Delta t}^{(i)} + \Delta\phi^{(i)}. \tag{8.11}$$

Dabei beschreibt $\phi_{t-\Delta t}^{(i)}$ die Phase zum vorherigen Zeitschritt und $\Delta\phi^{(i)}$ das Phaseninkrement, welches aus der besten Referenztrajektorie bestimmt wird. Das Phaseninkrement $\Delta\phi^{(i)}$ ist abhängig von der Länge L_{ref} der besten Referenztrajektorie und ergibt sich mit $\Delta\phi^{(i)} = \frac{1}{L_{\text{ref}}}$. Zusätzlich wird nach Berechnung der prädizierten Phase $\phi_t^{(i)}$ gaußsches Rauschen addiert.

Prädiktion des Hidden-States

Da eine Referenztrajektorie innerhalb einer Aktion mehrfach ausgeführt werden kann, wird die aktualisierte Phase $\phi_t^{(i)}$ eines Partikels herangezogen, um auszusagen, ob eine Transition in den nächsten oder übernächsten Hidden-State wahrscheinlich ist oder nicht. Die Funktion

$$F_{Phase}(\phi) = \phi^{\gamma_1} \tag{8.12}$$

definiert den Anteil der Partikel, auf den das eigentliche Zustandsübergangsmodell angewandt wird. Alle anderen Partikel bleiben in ihrem aktuellen Hidden-State und behalten ihre prädizierte Phase. Die Funktionswerte von F_{Phase} für einen Phasenbereich $\phi_{min} \leq \phi \leq 1$ liegen im Intervall $[0, 1]$. Dabei ist die Funktion $F_{Phase}(\phi)$ abhängig von der aktuellen Phase ϕ eines Partikels. Dadurch wird berücksichtigt, dass bei geringer Phase ϕ ein Zustandsübergang sehr unwahrscheinlich ist und mit zunehmender Phase immer mehr Partikel über das eigentliche Zustandsübergangsmodell prädiziert werden. Abbildung 8.8 (links) zeigt die Funktion $F_{Phase}(\phi)$ für $\gamma_1 = 6$, auch in den Experimenten wird dieser Wert gewählt. Unter der Annahme, dass genügend Trainingsdaten mit gelabelten Phasen verfügbar wären, könnte eine phasenabhängige Anteilsfunktion auch aus den Daten geschätzt werden.

Für den Anteil aller Partikel F_{Phase}, auf die das eigentliche Zustandsübergangsmodell angewandt wird, wird durch eine Sigmoidfunktion bestimmt, welcher Anteil der Partikel im aktuellen Hidden-State bleibt und welcher Anteil in den nächsten oder übernächsten Hidden-State

wechselt. Durch die Funktionen

$$F_0(t_{\text{Aktion}}) = 1 - \frac{1}{1 + e^{-\gamma_2 \cdot (t_{\text{Aktion}} - 0.5 \cdot \mu_{Aktion})}} \quad , \tag{8.13}$$

$$F_1(t_{\text{Aktion}}) = \gamma_3 * (1 - F_0(t_{\text{Aktion}})) \quad , \tag{8.14}$$

$$F_2(t_{\text{Aktion}}) = (1 - \gamma_3) * (1 - F_0(t_{\text{Aktion}})) \quad , \tag{8.15}$$

werden die Partikelanteile für den aktuellen Hidden-State $F_0(t_{\text{Aktion}})$ sowie die Anteile $F_1(t_{\text{Aktion}})$ und $F_2(t_{\text{Aktion}})$ für den nächsten und übernächsten Hidden-State beschrieben. Die Werte der obigen Funktionen liegen im Intervall $[0, 1]$. Die Variable t_{Aktion} definiert die Anzahl der Zeitschritte, in denen sich das System bereits im Hidden-State befindet. Mit μ_{Aktion} wird eine vorgegebene mittlere Verweildauer im Hidden-State, also eine mittlere Dauer für die Ausführung einer Aktion angegeben. Dieser Wert ist aktionsspezifisch und wird aus den Trainingsdaten abgeleitet. Das Modell wurde so konstruiert, dass zur Hälfte der mittleren Verweildauer einer Aktion genauso viele Partikel in den aktuellen Hidden-State gestreut werden wie in die beiden folgenden Hidden-States zusammen. Durch den Parameter γ_2 wird ein variables Steigungsmaß definiert, welches die Krümmung der Funktionen beeinflusst. In den Experimenten wird $\gamma_2 = 0.2$ und $\gamma_3 = 0.8$ gewählt. Abbildung 8.8 (rechts) zeigt die Funktionen $F_0(t_{\text{Aktion}})$, $F_1(t_{\text{Aktion}})$ und $F_2(t_{\text{Aktion}})$.

Prädiktion des Objektindex

Der Objektindex $id_t^{(i)}$ des geschätzten relevanten Objekts wird nicht prädiziert, sondern als fix angenommen. Der Objektindex ist dabei eine ganzzahlige Nummer aus der Liste der getrackten Objekte. Nur bei der Initialisierung eines Partikels wird $id_t^{(i)}$ gesetzt. Falls das geschätzte relevante Objekt nicht mehr vorhanden ist, wird das zugehörige Partikel neu initialisiert.

8.8.3 Aktualisierung

In diesem Schritt des eingesetzten Partikelfilters erfolgt die eigentliche Messung für alle prädizierten Partikel. Mit dem implementierten Beobachtungsmodell wird für jedes Partikel ein normiertes Gewicht

$$w_t^{(i)} = \frac{p(\mathbf{z}_t|\mathbf{x}_t^{(i)})}{\sum_{j=1}^{N} p(\mathbf{z}_t|\mathbf{x}_t^{(j)})} \tag{8.16}$$

bestimmt, sodass $\sum_{i=1}^{N} w_t^{(i)} = 1$ gilt. Hierbei beschreibt $p(\mathbf{z}_t|\mathbf{x}_t^{(i)})$ die Wahrscheinlichkeit, die Messung $\mathbf{z}_t$, gegeben den Zustand des Partikels $\mathbf{x}_t^{(i)}$, zu beobachten. Der Zustand eines Partikels $\mathbf{x}_t^{(j)}$ besteht aus dem Hidden-State $q_t^{(i)}$, der Phase $\phi_t^{(i)}$ innerhalb der Referenztrajektorie und dem Objektindex $id_t^{(i)}$.

Im Beobachtungsmodell des entwickelten Aktionserkennungssystems werden alle mit dem Partikel beschrieben Referenztrajektorien mit der aktuellen Input-Trajektorie im Sliding-Window verglichen. Es sind mehrere Vergleiche notwendig, da jedem Hidden-State mehrere Referenztrajektorien zugeordnet sind. Das Maß zum Trajektorienvergleich zwischen zwei Trajektorien basiert auf der Levenshtein-Distanz von Trajektorien (LDT). Außerdem wird die Ausgabe $\tilde{\mathbf{d}}_{\text{Fusion}}$ der Trajektorienklassifikatoren in die Gewichtsberechnung einbezogen.

3D-Trajektorienvergleich

In diesem Abschnitt wird ein Ansatz zum Vergleich zweier Trajektorien beschrieben. Der Vergleich einer Input-Trajektorie mit einer Referenztrajektorie wird zur Erkennung von Aktionen und Handlungen verwendet. Die Ähnlichkeit zu einer Referenztrajektorie kann Aufschluss darüber geben, welche Aktion gerade durchgeführt wird, denn bei gleichen Aktionen werden ähnliche Bewegungen durchgeführt. Der verwendete Ansatz zum Trajektorienvergleich besteht aus zwei Stufen. In der ersten Stufe wird die zurückgelegte Strecke entlang der Trajektorie für beide Trajektorien verglichen. In der zweiten Stufe wird die Levenshtein-Distanz von Trajektorien (LDT) berechnet, um die Ähnlichkeit beider Trajektorien zu bestimmen. Eine 3D-Trajektorie $\mathbf{T}$ mit Länge N_T ist definiert als 3D-Punktfolge

$$\mathbf{T} = \left[(X_1, Y_1, Z_1)^T, \ \ldots, (X_{N_T}, Y_{N_T}, Z_{N_T})^T \right] \ . \tag{8.17}$$

Die zurückgelegte Distanz $TD(\mathbf{T})$ entlang der Trajektorie $\mathbf{T}$ wird berechnet durch

$$TD(\mathbf{T}) = \sum_{t=2}^{N_T} \left\| \mathbf{T}_t - \mathbf{T}_{(t-1)} \right\| \quad . \tag{8.18}$$

Dabei beschreibt $\left\| \mathbf{T}_t - \mathbf{T}_{(t-1)} \right\|$ die L^2-Norm zwischen zwei aufeinander folgenden Zeitschritten innerhalb der 3D-Trajektorie $\mathbf{T}$.

Durch die erste Stufe wird angenommen, dass zwei Trajektorien $\mathbf{S}$ und $\mathbf{T}$ nur ähnlich sind, wenn die zurückgelegten Strecken entlang der Trajektorien $TD(\mathbf{S})$ und $TD(\mathbf{T})$ ähnlich sind. Wenn $\mathbf{S}$ und $\mathbf{T}$ eine ähnliche zurückgelegte Strecke besitzen, werden sie in ein einheitliches Koordinatensystem transformiert und durch das LDT-Maß verglichen. Falls die zurückgelegten Strecken $TD(\mathbf{S})$ und $TD(\mathbf{T})$ nicht ähnlich sind, stoppt der Algorithmus und es wird die maximale Länge beider Trajektorien als Vergleichsmaß zurückgegeben. Dadurch wird verhindert, dass unterschiedlich schnell ausgeführte Bewegungen miteinander verglichen werden.

Normalisierung: Da sich die Interaktionsobjekte, z.B. 3D-Positionen an einem Motor, an unterschiedlichen Positionen im 3D-Weltkoordinatensystem W befinden können oder sich der Kamerastandpunkt verändert, muss eine Rotations- und Translationsinvarianz der Aktionserkennung realisiert werden. Beispielsweise könnten zwei Schraubvorgänge an unterschiedlichen Positionen im Raum durchgeführt werden, durch die Rotations- und Translationsinvarianz ist es nicht notwendig die Referenztrajektorien erneut aufzuzeichnen und zu labeln. Die Rotations- und Translationsinvarianz von zwei miteinander verglichenen Trajektorien wird erreicht, indem sie durch das Verfahren von Kearsley (1989) so transformiert werden, dass die Summe der quadratischen Distanzen zwischen korrespondierenden Zeitschritten beider Trajektorien minimiert wird. Die notwendigen Rotations- und Translationsparameter werden durch einen analytischen Quaternionenansatz bestimmt. Quaternionen erlauben eine rechnerisch elegante Beschreibung des dreidimensionalen Raumes, insbesondere bei Rotationen. Der verwendete Ansatz wurde detailliert von Kearsley (1989) beschrieben. Das Ergebnis der Normalisierung sind zwei Trajektorien in einem einheitlich definierten Koordinatensystem. Dieses ist davon abhängig, welche der beiden Trajektorien auf die jeweils andere transformiert wird. Im implementierten System werden die Referenztrajektorien durch den Quaternionenansatz immer

auf die aktuelle Input-Trajektorie transformiert. Dadurch liegen die Referenztrajektorien immer im aktuellen Weltkoordinatensystem. Im nächsten Schritt wird das Ähnlichkeitsmaß der beiden Trajektorien **S** und **T** berechnet.

Levenshtein-Distanz von Trajektorien (LDT): Die Levenshtein-Distanz von Trajektorien (LDT) ist ein Ähnlichkeitsmaß zum Vergleich zweier Trajektorien.Die Levenshtein-Distanz (Levenshtein, 1966) ist eine String-Metrik, welche den Unterschied zwischen zwei Strings bezüglich der minimalen Anzahl der Operationen Einfügen, Löschen und Ersetzen definiert, um die eine Zeichenkette in die andere zu überführen.

Die Erweiterung auf d-dimensionale Trajektorien zum LDT-Ähnlichkeitsmaß wird durch folgenden Algorithmus beschrieben:

```
int LDT(trajectory S[1..d,1..m], trajectory T[1..d,1..n], matchingThresh)
   // D is the dynamic matrix with m+1 rows and n+1 columns
   declare int D[0..m, 0..n]
   for i from 0 to m  D[i, 0] := i
   for j from 1 to n  D[0, j] := j

   for i from 1 to m
       for j from 1 to n
           if(L2Norm(S[1..d,i] - T[1..d,j]) < matchingThresh)
                 then subcost := 0
                 else subcost := 1
           D[i, j] := min(D[i-1, j] + 1,        // deletion
                          D[i, j-1] + 1,        // insertion
                          D[i-1, j-1] + subcost)// substitution
   return D[m, n]
```

$LDT(\mathbf{S}, \mathbf{T})$ berechnet die minimal notwendige Anzahl der Operationen Einfügen, Löschen und Ersetzen, um die Trajektorie **T** in Trajektorie **S** zu transformieren. Ein „*matching threshold*“, eine Art „Schlauch“ um die Trajektorie **S**, bildet die euklidische Distanz zwischen zwei Elementen beider Trajektorien auf 0 oder 1 ab. Dadurch wird der Einfluss von Rauschen und Ausreißern reduziert. Für das LDT-Maß gelten einige obere und untere Schranken: (i) es beträgt mindestens den Unterschied der Längen beider Trajektorien, (ii) es beträgt höchstens die Länge der längeren Trajektorie und (iii) es ist nur dann Null, wenn die eine Trajektorie innerhalb des „Schlauchs“ der anderen liegt. Abbildung 8.9 zeigt anschaulich das Prinzip des LDT-Maßes für zwei eindimensionale Trajektorien 1 und 2. Neben den Punktkorrespondenzen ist auch der „Schlauch“ um die Trajektorie 1 dargestellt. In diesem Beispiel sind zwei Editieroperationen notwendig, um die Trajektorie 2 in Trajektorie 1 zu transformieren.

Gewichtsberechnung

Zur Berechnung des normierten Gewichts $w_t^{(i)}$ eines Partikels $\mathbf{s}_t^{(i)}$ wird die aktuelle Trajektorie im Sliding-Window $\mathbf{Z}_t$ eines getrackten Objekts $id_t^{(i)}$ mit allen durch das Partikel beschriebenen Referenztrajektorien verglichen. Als Ähnlichkeitsmaß wird der zuvor beschriebene Algorithmus zum 3D-Trajektorienvergleich verwendet. Die Referenztrajektorien werden durch den

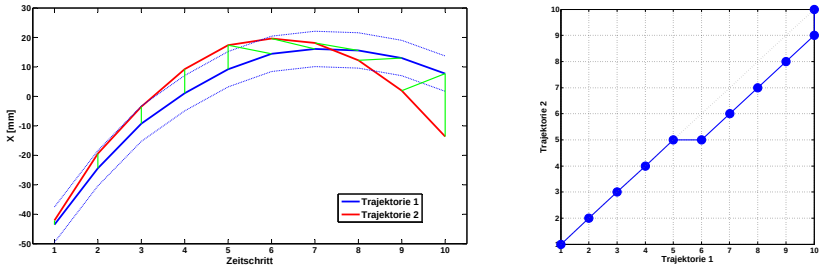

Abbildung 8.9: Links: LDT-Maß für zwei eindimensionale Trajektorien 1 (blau) und 2 (rot). Außerdem ist der „Schlauch", definiert durch den „*matching threshold*" (blau gepunktet) und die Punktkorrespondenzen (grün) abgebildet. Rechts: Optimaler Transformationspfad zwischen beiden Trajektorien. In diesem Beispiel sind zwei Editieroperationen notwendig, um die Trajektorie 2 in Trajektorie 1 zu transformieren.

geschätzten Hidden-State $q_t^{(i)}$ und die im Partikel geschätzte Phase $\phi_t^{(i)}$ festgelegt. Da jedem Hidden-State, also jeder Aktion, mehrere Referenztrajektorien zugeordnet werden, wird der Trajektorienvergleich mehrfach durchgeführt. Mit $LDT_{min}^{(i)}$ wird das minimale LDT-Maß für ein Partikel mit Index i bezeichnet. Da das Trajektorienähnlichkeitsmaß so definiert ist, dass es gering ist, wenn die verglichenen Trajektorien ähnlich sind und groß ist wenn keine Ähnlichkeit besteht, wird das LDT-Maß über folgende gaußsche Wahrscheinlichkeitsdichtefunktion gewichtet:

$$\widetilde{w}_{\mathrm{LDT}}^{(i)} = \frac{1}{\sigma_{\mathrm{LDT}} \cdot \sqrt{2\pi}} \cdot e^{-\frac{(LDT_{min}^{(i)} - \mu_{\mathrm{LDT}})^2}{2\sigma_{\mathrm{LDT}}^2}}. \tag{8.19}$$

Die Parameter $(\mu_{\mathrm{LDT}}, \sigma_{\mathrm{LDT}})$ der gaußschen Wahrscheinlichkeitsdichtefunktion werden dabei als $(\mu_{\mathrm{LDT}} = 0, \sigma_{\mathrm{LDT}} = 1)$ gewählt.

Zur Gewichtsberechnung wird neben dem Trajektorienähnlichkeitsmaß auch die Ausgabe der Trajektorienklassifikatoren (Abschnitt 8.7) in die Gewichtsberechnung einbezogen. Dies sorgt dafür, dass die Ausgabe der Klassifikatoren, welche für die Arbeitsaktionen nur von der Distanz zu den 3D-Interaktionsobjekten abhängig ist, durch den direkten Vergleich der aktuellen Bewegung mit den Aktionsprototypen validiert wird. Das unnormierte Gewicht $\widetilde{w}_t^{(i)}$ eines Partikels $\mathbf{s}_t^{(i)}$wird gemäß

$$\widetilde{w}_t^{(i)} = \widetilde{w}_{\mathrm{LDT}}^{(i)} \cdot \tilde{\mathbf{d}}_{\mathrm{Fusion}}(q_t^{(i)}), \tag{8.20}$$

berechnet. Das unnormierten Partikelgewicht $\widetilde{w}_t^{(i)}$ wird dabei durch Multiplikation des Gewichts $\widetilde{w}_{\mathrm{LDT}}^{(i)}$ aus dem Trajektorienvergleich und der Ausgabe $\tilde{\mathbf{d}}_{\mathrm{Fusion}}$ der Trajektorienklassifikatoren bestimmt. Aus $\tilde{\mathbf{d}}_{\mathrm{Fusion}}$ wird der zum Hidden-State $q_t^{(i)}$ zugehörige Diskriminanzwert verwendet. Ein Vorteil des entwickelten Verfahrens ist, dass neben dem Trajektorienähnlichkeitsmaß auch die Trajektorienklassifikatoren in die Gewichtsberechnung einbezogen werden. Die multiplikative Verknüpfung sorgt dafür, dass dann, wenn eines der Maße gering ist, auch das Partikelgewicht gering ist.

Nachdem alle unnormierten Gewichte $\widetilde{w}_t^{(i)}$ bestimmt wurden, werden die normierten Gewichte $w_t^{(i)}$ berechnet, sodass $\sum_{i=1}^{N} w_t^{(i)} = 1$ gilt.

Wenn sich das Aktionserkennungssystem im Sicherheitsmodus (Abschnitt 8.7) befindet, werden die Iterationsschritte des Partikelfilters nicht durchgeführt, sodass auch keine Gewichte

berechnet werden.

8.8.4 Erkennen von Aktionen

Für das endgültige Erkennen der Aktionen sowie deren Startzeitpunkte werden die normierten und gewichteten Partikel verwendet. Die Wahrscheinlichkeit, dass eine Referenztrajektorie in einem Hidden-State q_t zum Zeitschritt t abgeschlossen ist, wird wie in der Arbeit von Hofemann (2007) mit einer Endwahrscheinlichkeit P_{end} angegeben. Sie berechnet sich aus der Summe der Partikelgewichte $w_t^{(i)}$ mit einem speziellen Hidden-State q_t und einer Phase $\phi_t^{(i)}$ innerhalb einer zugehörigen Referenztrajektorie mit einem Wert von $\phi_t^{(i)} > \Theta_{\text{recog}}$:

$$P_{\text{end}}(q_t) = \sum_{i=1}^{N} \begin{cases} w_t^{(i)} & , \text{ falls } q_t^{(i)} = q_t \ \wedge \ \phi_t^{(i)} > \Theta_{\text{recog}} \\ 0 & , \text{ sonst.} \end{cases} \tag{8.21}$$

In dieser Arbeit wird $\Theta_{\text{recog}} = 0.9$ gewählt, was bedeutet, dass die im Partikel geschätzte beste Referenztrajektorie zu 90% durchlaufen sein muss, bevor eine Aktion erkannt werden kann. Der Hidden-State q_t mit der höchsten Endwahrscheinlichkeit wird zur Erkennung von Aktionen herangezogen. Da die zeitliche Länge der besten Referenztrajektorie bekannt ist, kann auch auf den Startzeitpunkt der Aktion geschlossen werden. Damit kann sehr schön der Zeitbereich einer Aktion ermittelt werden und auch zwischenzeitliche Unsicherheiten in der Erkennung überschieben werden. Denn wenn eine sehr hohe Endwahrscheinlichkeit berechnet wurde, muss auch während der kompletten Dauer der besten Referenztrajektorie die zugehörige Aktion ausgeführt worden sein.

Abbildung 8.10 zeigt das Ergebnis des Aktionserkennungssystems für eine Beispielsequenz.

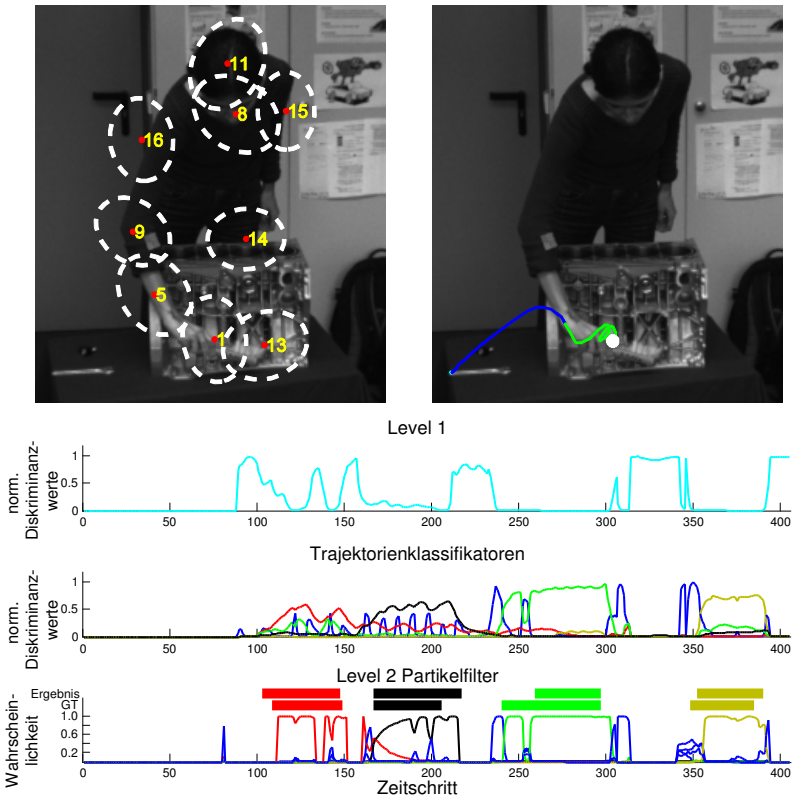

Abbildung 8.10: Das Aktionserkennungssystem für einen Beispieldatensatz. Oben links sind alle getrackten Objekte des 3D-Mean-Shift-Tracking mit ihren zugehörigen Objektindizes dargestellt. Oben rechts wird die ins Bild projizierte Bewegungs- und Erkennungshistorie des als relevant geschätzten Objekts aufgezeigt. Außerdem ist die Ausgabe der Level 1 und 2 dargestellt. Der normalisierte Diskriminanzwert $\tilde{d}_{\text{unknown}}$ (cyan) beschreibt im Level 1 zu welchem Zeitschritt das Aktionserkennungssystem in den Sicherheitsmodus geht. Außerdem sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren für die Aktionen „Transferbewegung“ (blau), „Schrauben 1“ (rot), „Schrauben 2“ (schwarz), „Putzen“ (grün) und „Stecken“ (braun) dargestellt. Die Ausgabe des Levels 2 ergibt sich aus der a posteriori Wahrscheinlichkeit des Partikelfilters und ist ganz unten dargestellt. Außerdem sind die erkannten Aktionen als Balken im Vergleich zur Ground-Truth (GT) aufgezeigt.

Kapitel 9

Experimentelle Untersuchungen zur Aktionserkennung

In diesem Kapitel werden die Ergebnisse der experimentellen Untersuchungen des in dieser Arbeit entwickelten Systems zur Aktionserkennung vorgestellt. Dabei werden die Ergebnisse des Aktionserkennungssystems mit manuell gelabelten Ground-Truth-Daten verglichen. Es wurde ein möglichst realistisches Beispielszenario gewählt. Es geht um Montagearbeiten eines Werkers an einem Motorblock (Abschnitt 8.3). Das gewählte Beispiel ist realitätsnah, denn einfache Montageprozesse in der Automobilproduktion haben keine höhere Komplexität. Das in dieser Arbeit gewählte Beispielszenario soll stellvertretend für andere einfache Montageprozesse, z.B. „Handeinlegestationen“ stehen, bei denen als Erstes eine Interaktion zwischen Mensch und Industrieroboter realisiert werden könnte. Es soll gezeigt werden, dass eine robuste Erkennungsleistung unter den Randbedingungen des Szenarios dieser Arbeit (Abschnitt 8.1) möglich ist.

Im Folgenden wird das Beispielszenario kurz vorgestellt und der gelabelte Datensatz zur Evaluierung des Aktionserkennungssystems beschrieben. Anschließend werden die Ergebnisse der Aktionserkennung aufgezeigt. Diese basieren auf den Bewegungsdaten von zwei unterschiedlichen Trackingsystemen. Damit soll gezeigt werden, dass das entwickelte Aktionserkennungssystem in der Lage ist, mit Bewegungsdaten, welche von einem beliebigen, aber zeitlich robusten 3D-Trackingsystem berechnet werden, gute Erkennungsleistungen zu realisieren. Am Ende des Kapitels wird eine Wertung der Ergebnisse gegeben und diese mit denen anderer Autoren verglichen.

9.1 Beispielszenario und verwendeter Datensatz

Am Beispiel von Montagearbeiten eines Werkers an einem Motorblock soll das in Kapitel 8 beschriebene System zur Aktionserkennung evaluiert werden. Das dabei gewählte Arbeitsszenario ist realitätsnah und entspricht in der Komplexität einfachen Montageprozessen in der Automobilproduktion. Im gewählten Beispiel müssen folgende vier Arbeitsaktionen ausgeführt werden:

„Schrauben 1“: Es muss eine Schraube an einer definierten Position am Motor festgezogen werden.

„Schrauben 2“: Eine weitere Schraube muss an einer anderen definierten Position am Motor

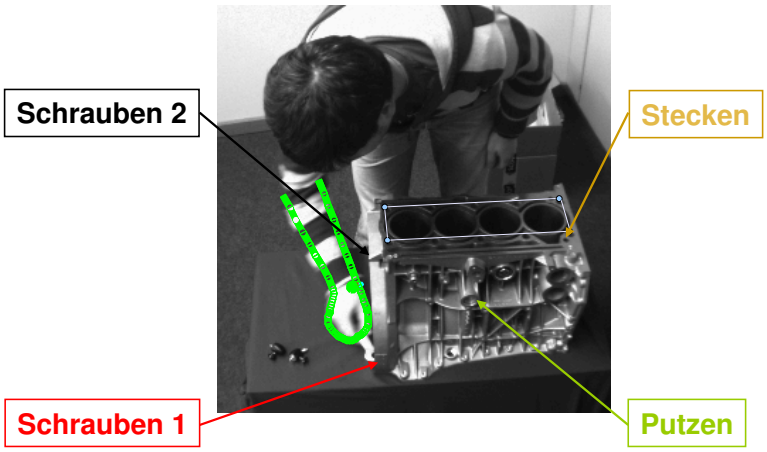

Abbildung 9.1: Ein Beispiel für die Anordnung der 3D-Interaktionsobjekte am Motor. Die tatsächliche zur Aktionsklasse „Schrauben 2“ zugeordnete Position ist im Bild nicht zu sehen.

festgezogen werden.

„Putzen“: In dieser Aktionsklasse muss ein definierter Bereich am Motor mit einem Handfeger gereinigt werden.

„Stecken“: In dieser Aktionsklasse muss der Werker ein zuvor gegriffenes Objekt an eine definierte Position am Motor stecken.

Diese vier Arbeitsaktionen sind die zu erkennenden Hauptbestandteile des zyklischen Arbeitsprozesses. Jede der Arbeitsaktionen ist einem definierten 3D-Objekt am Motor zugeordnet (Abbildung 9.1).

Bei der Aufzeichnung des verwendeten Testdatensatzes zur Evaluierung der Aktionserkennung wurden die vorgegebenen Randbedingungen (Abschnitt 8.1) berücksichtigt. In realen Umgebungen kann sich das getrackte Objekt, also der Werker, unerwartet verhalten, unkontrollierte Bewegungen ausführen, den Arbeitsprozess immer wieder unterbrechen oder diesen in einer unbekannten Reihenfolge durchführen. Der verwendete Datensatz besteht aus 22 realen Testsequenzen, davon werden zwei Sequenzen als Lernstichprobe verwendet und die restlichen 20 Sequenzen als Testdatensatz. In den Sequenzen werden die im Beispielszenario beschriebenen Montagearbeiten von acht verschiedenen Testpersonen durchgeführt. Die Arbeiten werden vor stark strukturierten Hintergründen ausgeführt. Die Position des Kamerasystems variiert dabei, d.h. das Interaktionsobjekt (der Motor) ist immer an einer anderen Position im Weltkoordinatensystem. Die Distanz der Testperson zur Kamera beträgt 2.2 m bis 3.3 m und der Arbeitsprozess wird immer mit der rechten Hand ausgeführt. Der Datensatz beinhaltet die ersten fünf Sequenzen der experimentellen Untersuchungen zum Tracking des Hand-Unterarm-Bereichs (Abschnitt 7.3).

Eine weitere Randbedingung des Szenarios dieser Arbeit gibt vor, dass nur wenige Trainigsdaten zur Aktionserkennung zur Verfügung stehen. Dies wurde auch im Datensatz berücksichtigt, denn es wurden nur zwei Sequenzen als Lernstichprobe verwendet. Für ein Aktionserkennungssystem in der Produktion ist es nicht praktikabel und zu kostenintensiv, viele verschiedene Werker bei ihrer Tätigkeit aufzunehmen, die extrahierten Bewegungsdaten mit Aktionsklassen zu labeln und dann das Erkennungssystem zu trainieren. Es ist aber möglich, die

Tabelle 9.1: Überblick über die für die Aktionserkennung relevanten Arbeitsaktionen und wichtige statistische Daten. Es sind für die jeweiligen Arbeitsaktion des kompletten Datensatzes die durchschnittliche Dauer der Arbeitsaktion, die Standardabweichung der Ausführungsdauer sowie die Häufigkeit der Arbeitsaktion in der Lernstichprobe und im Testset angegeben

Arbeitsaktion	Durchschnittl. Dauer [ms]	Std.-abweichung Dauer [ms]	Häufigkeit in Lernstichprobe	Häufigkeit im Testset
„Schrauben 1“	3111	672	2	26
„Schrauben 2“	3177	376	2	28
„Putzen“	5023	1344	2	31
„Stecken“	2258	689	3	33

Bewegungsdaten weniger Vorarbeiter, welche den Arbeitsprozess detailliert kennen und die Werker angelernt haben, aus aufgenommenen Kamerabildern zu extrahieren und als Grundlage für das Erkennungssystem zu verwenden. Auch dies wurde im Datensatz berücksichtigt, denn die Lernstichprobe stammt nur von zwei der Testpersonen und diese haben alle weiteren Testpersonen mit dem zyklischen Arbeitsablauf vertraut gemacht. Die beiden Testpersonen aus der Lernstichprobe könnten also als Vorarbeiter aufgefasst werden, welche den Arbeitsprozess detailliert kennen. Das Anlernen einer Testperson gestaltet sich folgendermaßen. Der Bewegungsablauf des Arbeitsprozesses wurde durch den Vorarbeiter gezeigt und danach von der Testperson verlangt, den Arbeitsablauf zu wiederholen. Dabei wurde die Durchführung durch die Vorarbeiter kontrolliert. Bei einer falschen Ausführung des Prozesses griffen die Vorarbeiter ein und korrigierten die Testperson. Da der Arbeitsablauf einfach ist, waren im Maximum nur zwei Testläufe des Prozesses notwendig.

In den Datensatz wurden außerdem noch Sequenzen integriert, in denen Fehler im Arbeitsprozess gemacht wurden. Diese Sequenzen stammen aus den Testläufen zum Anlernen der Testpersonen. Dabei wurden Aktionen vergessen, oder die Aktion „Stecken“ mehrfach durchgeführt. Außerdem gibt es Sequenzen, in denen der Arbeitsablauf bis zu vier mal in Folge durchgeführt werden musste. Zusätzlich wurden Sequenzen verwendet, in denen die Testpersonen den Arbeitsablauf bewusst durch ein „Nase fassen“ oder ein „Kratzen am Kopf“ unterbricht. In Tabelle 9.1 findet sich eine Zusammenstellung der Aktionsklassen im Datensatz und einige wichtige statistische Daten. Es ist zu erkennen, dass eine weitere Randbedingung erfüllt ist, nämlich, dass mindestens die zehnfache Menge an Testdaten vorhanden ist.

Jede Sequenz wurde manuell mit Aktionsklassen gelabelt. Dabei wurde für jeden Zeitschritt einer Sequenz manuell das Label einer Aktionsklasse gewählt. Diese Labels sind „unbekannte Bewegung“, „Transferbewegung“, „Schrauben 1“, „Schrauben 2“, „Putzen“ und „Stecken“. Da jeder Zeitschritt gelabelt wurde, kann ein Zeitbereich für eine Aktion extrahiert werden. D.h. es wird der Beginn und das Ende einer Aktion innerhalb einer Sequenz ermittelt. Dadurch kann neben der Erkennungsrate für eine Arbeitsaktion auch der zeitliche Versatz des Erkennungsergebnisses zur Ground-Truth für den Beginn und das Ende einer Arbeitsaktion ermittelt werden. In den Ergebnissen wird daher betrachtet, ob eine Arbeitsaktion richtig erkannt wurde und wann diese richtig erkannt wurde. Tabelle 9.1 zeigt beispielsweise die durchschnittliche Dauer der relevanten vier Arbeitsaktionen.

Da sich die Interaktionsobjekte, welche durch 3D-Positionen am Motor definiert werden, für viele der Sequenzen an unterschiedlichen Positionen im 3D-Weltkoordinatensystem W befinden können, mussten die 3D-Positionen für jede der Testsequenzen durch einen Bündelausgleich (Triggs u. a., 2000) manuell ermittelt werden. Die Interaktionsobjekte, also Motorenbereiche, die durch 3D-Positionen definiert werden, stehen dem Aktionserkennungssystem während der automatischen Erkennung zu Verfügung. Die Position der Motorenbereiche ist wichtig für die Vorverarbeitung der 3D-Trajektorien (Abschnitt 8.5), für die Trajektorienklassifikatoren (Abschnitt 8.7) und für die Zustandsschätzung im HMM mit dem Partikelfilter (Abschnitt 8.8).

9.2 Ergebnisse Aktionserkennung

Sowohl beim Labeln der Testsequenzen, als auch bei der automatischen Erkennung wird eine Aktion als Zeitbereich angegeben. Durch den Vergleich der Zeitbereiche kann ermittelt werden, ob eine Aktion korrekt erkannt wurde (Korrekt K), ob eine Aktion nicht erkannt wurde (Löschung L), ob eine Aktion falsch erkannt wurde (Vertauschung V) oder ob eine Aktion zu einem Zeitbereich mit unbekannter Kennung eingefügt wurde (Einfügung E). Ein Beispiel für Zeitbereiche der Erkennung im Vergleich zur gelabelten Ground-Truth ist in Abbildung 8.10 dargestellt. Die Erkennungsleistung eines Aktionserkennungssystems spiegelt sich in den korrekt erkannten Aktionen (K), den nicht erkannten Aktionen (L), den Vertauschungen (V) und den Einfügungen (E) von Aktionen wieder. Aus diesen Werten lassen sich zwei wichtige Kennziffern berechnen. Zum einen ist das die Erkennungsrate (ER)

$$ER = \frac{K}{N} \tag{9.1}$$

und zum anderen die Wortfehlerrate (engl. *word error rate*) abgekürzt mit WER,

$$WER = \frac{L + V + E}{N}. \tag{9.2}$$

Wobei mit N die Häufigkeit der Aktion im Testdatensatz beschrieben wird. Die Wortfehlerrate (WER) ist eine bekannte Kennziffer aus der Spracherkennung und kann auch für die Aktionserkennung verwendet werden.

Durch den Vergleich der Zeitbereiche kann außerdem der zeitliche Versatz der automatischen Erkennung im Vergleich zu den gelabelten Aktionen ermittelt werden. Dabei wird der zeitliche Versatz des Erkennungsergebnisses zur Ground-Truth für den Beginn und das Ende einer Arbeitsaktion berechnet. Somit kann über alle Testsequenzen der Mittelwert und die Standardabweichung des zeitlichen Versatzes einer Aktion ermittelt werden.

Das in dieser Arbeit beschriebene Aktionserkennungssystem ist bewusst allgemein gestaltet, sodass ein beliebiges 3D-Trackingsystem zur Generierung der aktionsspezifischen Merkmale für die Erkennung verwendet werden kann. Als aktionsspezifisches Merkmal werden Trajektorien von 3D-Punkten verwendet. Wie bereits im Stand der Technik (Kapitel 2, Abschnitt 2.3.4) ausführlich beschrieben, bieten Trajektorien eine kompakte und sehr diskriminative Repräsentation der Bewegungsdaten. Ein weiterer Vorteil ist, dass 3D-Trajektorien sehr allgemein sind

und von jedem beliebigen 3D-Trackingsystem erzeugt werden können. Würden Trajektorien von Körperkonfigurationen (Abschnitt 2.3.4) als aktionsspezifische Merkmale verwendet, so ist ein modellbasiertes Körpertracking unbedingt notwendig.

Um zu zeigen, dass die entwickelte Aktionserkennung generell verwendbar ist, werden die Ergebnisse der Aktionserkennung auf Basis der Testsequenzen für zwei unterschiedliche Trackingsysteme vorgestellt. Die zur Aktionserkennung verwendeten 3D-Trajektorien stammen zum einen von Trackingsystem 4 (Shape-Flow-Tracking mit Reinitialisierungsmodul) aus Kapitel 4. Zum anderen werden die aktionsspezifischen Merkmale mit Trackingsystem 5 (3D-Mean-Shift-Tracking) aus Kapitel 5 erzeugt. Beide Trackingsysteme basieren auf Grauwertbildern als Eingabedaten. Ein Unterschied zwischen den Trackingsystemen besteht darin, dass bei System 4, einem modellbasierten Trackingansatz, direkt der rechte Unterarm in den Kamerabildern verfolgt wird. Bei System 5 hingegen werden alle bewegten Objekte in der beobachteten Szene mit einem einfachen Ellipsoidmodell in den Kamerabildern verfolgt. Dadurch muss zusätzlich geschätzt werden, welches der getrackten Objekte die Aktion ausführt.

Zur Berechnung der Ergebnisse auf Basis der Testsequenzen wurden für die Aktionserkennung immer dieselben Parameter gewählt. Es wurden insgesamt $N = 500$ Partikel verwendet und davon wurden zu jedem Zeitschritt $M = 100$ Partikel zufällig über alle Zustände des HMMs verstreut, also neu initialisiert. Das zufällige Streuen der Partikel macht das entwickelte Aktionserkennungssystem robuster und ist die Grundlage dafür, dass auch ein Überspringen mehrerer Aktionen möglich ist.

9.2.1 Ergebnisse der Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul

Durch das Shape-Flow-Tracking (Kapitel 4) wird der rechte Hand-Unterarm-Bereich der Testperson in den Kamerabildern über die Zeit verfolgt. Die Aktionen gehen dabei von der rechten Hand der Testperson aus. Ein weiteres Körperteil wird nicht getrackt, es ist also immer gewährleistet, dass das aktionsausführende Objekt, also die Hand, getrackt wird. Für alle Testsequenzen wurden dieselben Parameter des Trackingsystems und des verwendeten Hand-Unterarm-Modells (Abschnitt 3.2.1) verwendet. Die qualitativen und quantitativen Ergebnisse des Shape-Flow-Trackings mit Reinitialisierungsmodul wurden im Abschnitt 7.3.4 vorgestellt. Zum ersten Zeitschritt einer Sequenz wird das 3D-Modell manuell auf dem zu verfolgenden Hand-Unterarm-Bereich initialisiert und dieser anschließend durch das Trackingsystem in den Kamerabildern verfolgt. Dabei wurden für alle 20 Testsequenzen Verfolgungsraten von 100% erreicht. An das Aktionserkennungssystem wird durch das Trackingsystem die geschätzte 3D-Trajektorie des Handgelenks (Abschnitt 3.2.1) übergeben. Trajektorien von Körper- oder Modellkonfiguration werden nicht verwendet, d.h. die Freiheitsgrade des Unterarms und der Hand werden nicht geschätzt. Dadurch wird die Aktionserkennung genereller verwendbar für beliebige 3D-Trackingsysteme.

In Tabelle 9.2 sind die Ergebnisse der Aktionserkennung auf Basis der 20 Testsequenzen mit Trackingsystem 4 als Trajektoriengenerator aufgezeigt. Für die vier relevanten Arbeitsaktionen werden Erkennungsraten (ER) von 92% bis 100% erzielt. Dies ist ein gutes Ergebnis, denn im Vergleich zum Testdatensatz wurde eine um mindestens den Faktor 10 kleinere Lernstichprobe verwendet. Besonders bemerkenswert ist die Erkennungsrate von 100% für die Arbeitsaktion

Tabelle 9.2: Erkennungsergebnisse auf Basis der 20 Testsequenzen. Eingesetztes Trackingverfahren: Shape-Flow-Tracking mit Reinitialisierungsmodul (Kapitel 4).

	„Schrauben 1“	„Schrauben 2“	„Putzen“	„Stecken“
Gesamt (N)[#]	26	27	31	33
Korrekt (K) [#]	24	25	29	33
Duplikate (D) [#]	0	1	2	0
Löschung (L) [#]	1	2	2	0
Vertauschung (V) [#]	1	0	0	0
Einfügung (E) [#]	1	0	0	0
Erkennungsrate (ER) [%]	92.3	92.6	93.5	100
Wortfehlerrate (WER) [%]	11.5	7.4	6.5	0.0
Versatz Beginn (mean) [ms]	239	−125	288	268
Versatz Beginn (std) [ms]	345	683	1237	559
Versatz Ende (mean) [ms]	254	−103	−767	305
Versatz Ende, (std) [ms]	324	983	1475	256

„Stecken“. Die Genauigkeit des Trackingsystems wird hier besonders ausgenutzt, denn das Stecken endet immer an derselben bekannten 3D-Position.

Die Wortfehlerrate (WER), welche sich aus der Summe der Löschungen (L), Vertauschungen (V) und Einfügungen (E) zur Gesamtzahl (N) der Aktionen ergibt, liegt im Bereich von 0% bis 11.5% und im Mittel über alle Arbeitsaktionen bei 6%. Die Wortfehlerrate erlaubt aber noch einen tieferen Blick in das Verhalten des Aktionserkennungssystems. Über alle Arbeitsaktionen gesehen, tragen die Löschungen, also die falsch negativen Erkennungen, am stärksten zur Wortfehlerrate bei. Die falsch positiven Erkennungen, die Einfügungen, sind eher selten. Insgesamt gibt es nur eine Einfügung. Ein weiterer Vorteil des Systems ist, dass die Gesamtanzahl der Vertauschungen ($V = 1$) sehr gering ist. Ein solcher Fehler wäre in einem Gesamtsystem, welches aus mehreren Aktionserkennungssystemen besteht, schwerer zu beheben. Bei der Fusion der Erkennungen aus unterschiedlichen Erkennungssystemen kann eine Vertauschung zu widersprüchlichen Aussagen führen. Die Vertauschung stammt aus Testsequenz 19. Hier wurde die Arbeitsaktion „Schrauben 1“ nicht erkannt und der Zeitbereich für „Schrauben 2“ reicht zu einem geringen Teil in den gelabelten Zeitbereich für „Schrauben 1“ hinein. Die Schwierigkeit für die Erkennung liegt darin, dass die beiden Aktionen in dieser Testsequenz extrem dicht aufeinander folgen. Die Transferbewegung, welche beide Aktionen voneinander trennt, dauert nur ca. fünf Zeitschritte an, was mit einem Sliding-Window der Länge $L_{\mathrm{SW}} = 8$ nur schwer zu erkennen ist.

Ein weiterer Vorteil des Systems ist, dass es eine relativ geringe Anzahl an doppelt erkannten Aktionen aufweist. Dies liegt daran, dass über die Endwahrscheinlichkeit einer Aktion (Abschnitt 8.8.4) und der besten Referenztrajektorie der Startzeitpunkt der Handlung gut berechnet werden kann. Bereiche in denen das System unsicher arbeitet, können somit überschrieben werden, da die zeitliche Länge der besten Referenztrajektorie bekannt ist.

In Tabelle 9.2 sind außerdem die Ergebnisse der Aktionserkennung auf Basis der 20 Testsequenzen für die zeitliche Genauigkeit der Erkennung dargestellt. Es wurde also eine Aussage darüber getroffen, wie genau die Erkennung in zeitlicher Hinsicht ist. Es ist zu erkennen, dass

die Arbeitsaktionen mit einem zeitlichen Versatz von wenigen Zehntelsekunden erkannt werden. Nur der zeitliche Beginn der Arbeitsaktion „Schrauben 2“ wird im Mittel mit -125 ms etwas zu früh erkannt. Alle anderen Arbeitsaktionen werden zum Beginn im Mittel mit 265 ms detektiert, also wenige Zehntelsekunden zu spät erkannt. Die Standardabweichung des zeitlichen Versatzes zum Beginn einer Arbeitsaktion zeigt, dass es bei den Aktionen „Schrauben 1“ und „Stecken“ nur mäßige Schwankungen gibt. Bei der Aktion „Putzen“ hingegen ist die Standardabweichung des zeitlichen Versatzes zum Beginn stark erhöht, dies deutet auf starke Schwankungen hin. Der Grund hierfür liegt in den ausgeführten Bewegungen und deren Zeitdauern, denn die Testpersonen putzen häufig unterschiedlich lange und mit unterschiedlichen Bewegungen. Der Unterschied in der Ausführungsdauer ist auch in Tabelle 9.1 zu erkennen. Hier ist die Standardabweichung der Ausführungsdauer für die Aktion „Putzen“ im Vergleich zu allen anderen Aktionen erhöht. Da die ausgeführten Bewegungen für die Aktion „Putzen“ häufig unterschiedlich sind, wird teilweise nur ein Teil des gelabelten Zeitbereichs als „Putzen“ erkannt.

Für den zeitlichen Versatz am Ende einer Aktion zeigen sich ähnliche Ergebnisse, im Mittel wird das Ende einer Aktion auf wenige Zehntelsekunden genau erkannt. Für die Aktionen „Schrauben 2“ und „Putzen“ wird das Ende häufig zu früh erkannt, daher sind die Mittelwerte des zeitlichen Versatzes am Ende negativ. Der Grund für die starken Schwankungen der Arbeitsaktion „Putzen“ ist wurde bereits am Anfang dieser Seite beschrieben.

Im Testdatensatz wurden bewusst Sequenzen verwendet, in denen die Testpersonen den zyklischen Arbeitsablauf durch ein „Nase fassen“ oder ein „kratzen am Kopf“ unterbrochen haben. Da die Erkennungsleistung des Systems trotzdem hoch ist, ist abzuleiten, dass die entwickelte Aktionserkennung auch in der Lage ist, unbekannte Bewegungen auszuschließen und die Erkennung nach Fortsetzung des zyklischen Arbeitsablaufs durch die Testperson fortzuführen.

Die mittlere Laufzeit der Matlab-Implementierung der Aktionserkennung auf Basis aller Testsequenzen beträgt 4.5 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Die Laufzeit des Shape-Flow-Trackings wird nicht berücksichtigt, sondern nur die Laufzeit des Aktionserkennungssystems betrachtet. Der größte Teil der Gesamtlaufzeit des Algorithmus wird durch die vielen notwendigen Trajektorienvergleiche verbraucht. Denn es muss für jedes Partikel der 3D-Trajektorienvergleich (Abschnitt 8.8.3) der Input-Trajektorie mit allen durch das Partikel beschriebenen Referenztrajektorien berechnet werden. Durch eine effektive C-Implementierung sind starke Verbesserungen im Laufzeitverhalten zu erwarten.

Abbildung 9.2 zeigt ein Beispiel für die Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul. Es ist das Trackingergebnis und die Historie der Erkennung im Bild der Masterkamera dargestellt. Weiterhin wird die Partikelverteilung im HMM und die Ausgabe der Level 1 und 2 aufgezeigt. Es wurde als letztes die Arbeitsaktion „Stecken“ erkannt, zuvor wurde eine Transferbewegung detektiert. Ein weiteres Beispiel für die Aktionserkennung ist in Abbildung 9.3 dargestellt. Wiederum ist das Trackingergebnis und die Historie der Erkennung im Bild der Masterkamera dargestellt. Weiterhin wird die Partikelverteilung im HMM und die Ausgabe der Level 1 und 2 aufgezeigt. Es wurde als letztes eine unbekannte Bewegung erkannt, zuvor wurde durch die Testperson die Arbeitsaktion „Schrauben 1“ durchgeführt und diese Aktion auch erkannt. Im Beispiel ist zu erkennen, dass die

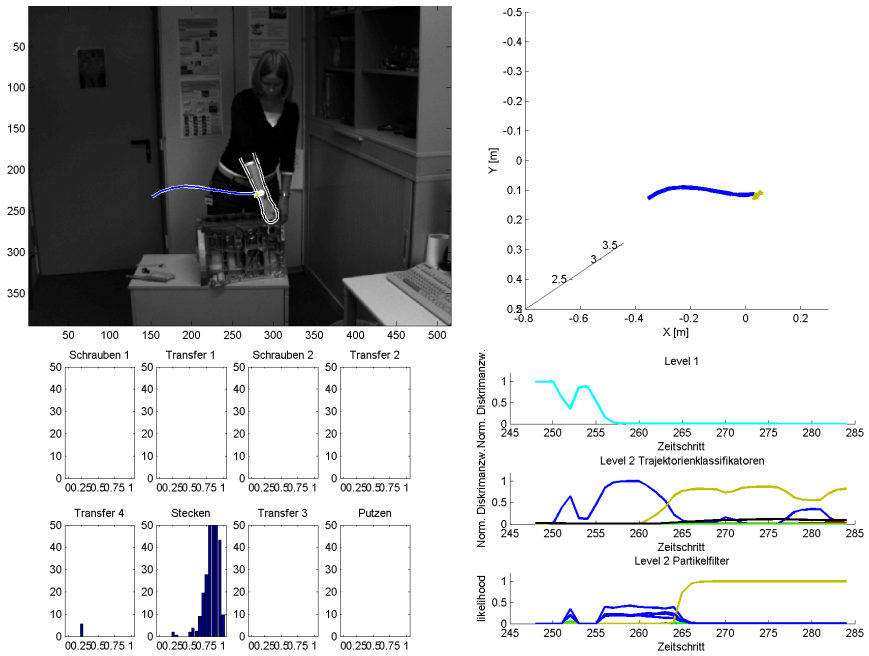

Abbildung 9.2: Beispielergebnis der Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul. Oben links: Trackingergebnis projiziert ins Bild der Masterkamera, außerdem wird die Erkennungshistorie gezeigt. Es wurde ein „Transfer“ (blau) mit anschließendem „Stecken“ (braun) erkannt. Oben rechts: Erkennungshistorie im 3D-Raum. Unten links: Die acht Hidden-States des HMMs sowie die Partikelverteilung für jeden Hidden-State. Es wird ein Histogramm für jeden Hidden-State (Phase auf der Abszissenachse) dargestellt. Ein großer Teil der Partikelmenge befindet sich im Hidden-State der Aktionsklasse „Stecken“, siehe Histogramm. Unten rechts: Ausgabe der Levels 1 und 2. Der normalisierte Diskriminanzwert $\tilde{d}_{\text{unknown}}$ (cyan) beschreibt im Level 1 zu welchem Zeitschritt das Aktionserkennungssystem in den Sicherheitsmodus geht. Weiterhin sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren gezeigt. Außerdem wird die aus dem Partikelfilter abgeleitete Wahrscheinlichkeit für die Hidden-States des HMMs dargestellt.

Iterationsschritte des Partikelfilters nicht durchgeführt werden, da sich das System im Sicherheitsmodus befindet. Ein großer Teil der Partikel ist noch im Hidden-State „Schrauben 1“.

9.2.2 Ergebnisse der Aktionserkennung auf Basis des 3D-Mean-Shift-Trackings

In diesem Abschnitt werden die Ergebnisse der Aktionserkennung auf Basis der Testsequenzen für das 3D-Mean-Shift-Tracking (Kapitel 5) als Generator der notwendigen 3D-Trajektorien vorgestellt. Durch das Trackingsystem werden alle bewegten Objekte in der beobachteten Szene mit einem einfachen Ellipsoidmodell in den Kamerabildern verfolgt. Die Aktionen gehen dabei von der rechten Hand der Testperson aus. Da im Trackingsystem nicht bekannt ist, welches der getrackten Objekte die rechte Hand ist, muss zusätzlich geschätzt werden,

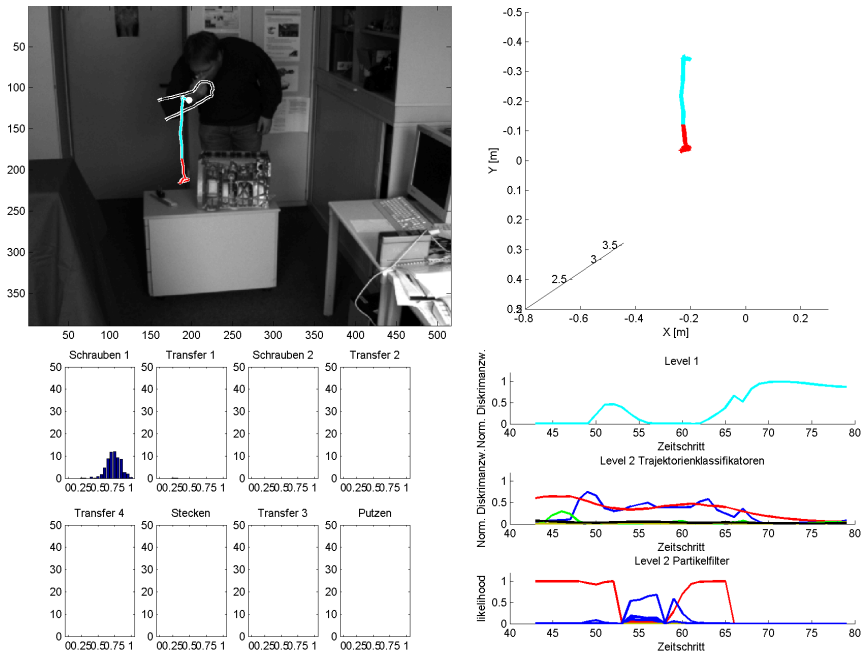

Abbildung 9.3: Beispielergebnis der Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul. Oben links: Trackingergebnis projiziert ins Bild der Masterkamera, außerdem wird die Erkennungshistorie gezeigt. Es wurde „Schrauben 1“ (rot) erkannt, mit einem anschließenden „Nase fassen“ der Testperson, also einer unbekannten Bewegung (cyan). Unten links: Die acht Hidden-States des HMMs sowie die Partikelverteilung in einem Histogramm (Phase auf der Abszissenachse) für jeden Hidden-State. Unten rechts: Ausgabe der Levels 1 und 2. Der normalisierte Diskriminanzwert $\tilde{d}_{\text{unknown}}$ (cyan) zeigt, dass das Aktionserkennungssystem im Level 1 in den Sicherheitsmodus geht. Weiterhin sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren gezeigt. Außerdem wird die aus dem Partikelfilter abgeleitete Wahrscheinlichkeit für die Hidden-States des HMMs dargestellt.

welches der getrackten Objekte die Aktionen ausführt, also das aktionsausführende Objekt ist. Die Schätzung des relevanten Objekts wird durch einen Partikelfilter vorgenommen, dazu wird der Index $id_t^{(i)}$ des relevanten Objekts aus allen getrackten Objekten mit in den Zustandsvektor des Partikelfilters aufgenommen (Abschnitt 8.8). Das relevante Objekt ist das getrackte Objekt, welches die zu erkennenden Aktionen ausführt, also die Hand der Testperson. Für alle Testsequenzen wurden dieselben Parameter des Trackingsystems und des verwendeten Ellipsoidmodells verwendet. Die qualitativen und quantitativen Ergebnisse des 3D-Mean-Shift-Trackings wurden im Abschnitt 7.3.5 vorgestellt. Im Trackingsystem ist eine automatische Objektdetektion implementiert, daher muss das Modell nicht manuell initialisiert werden, wie beispielsweise beim Shape-Flow-Tracking. Beim Tracking wurden für alle 20 Testsequenzen Verfolgungsraten von 100% erreicht, wie die Auswertung der Trackingergebnisse (Abschnitt 7.3) zeigt, besitzen beide zur Aktionserkennung verwendeten Trackingsysteme eine ähnliche Genauigkeit. An das Aktionserkennungssystem werden durch das Trackingsystem die geschätzten 3D-Trajektorien der Mittelpunkte der getrackten Ellipsoide übergeben.

In Tabelle 9.3 sind die Ergebnisse der Aktionserkennung auf Basis der 20 Testsequenzen mit Trackingsystem 5 (3D-Mean-Shift-Tracking) als Trajektoriengenerator aufgezeigt. Für die

Tabelle 9.3: Erkennungsergebnisse auf Basis der 20 Testsequenzen. Eingesetztes Trackingverfahren: 3D-Mean-Shift-Tracking (Kapitel 5).

	„Schrauben 1“	„Schrauben 2“	„Putzen“	„Stecken“
Gesamt (N)[#]	26	27	31	33
Korrekt (K) [#]	24	26	29	31
Duplikate (D) [#]	2	2	3	3
Löschung (L) [#]	2	1	2	1
Vertauschung (V) [#]	0	0	0	1
Einfügung (E) [#]	1	2	0	0
Erkennungsrate (ER) [%]	92.3	96.3	93.5	93.9
Wortfehlerrate (WER) [%]	11.5	11.1	6.5	6.1
Versatz Beginn (mean) [ms]	−324	116	625	−31
Versatz Beginn (std) [ms]	754	826	1192	1262
Versatz Ende (mean) [ms]	−71	78	−461	572
Versatz Ende, (std) [ms]	822	1060	1458	1707

vier relevanten Arbeitsaktionen werden Erkennungsraten (ER) von 92% bis 96% erzielt. Dies ist wiederum ein gutes Ergebnis, denn es wurde natürlich dieselbe Lernstichprobe wie bei der Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul verwendet. Die Wortfehlerrate (WER) liegt im Bereich von 6% bis 11.5%. Im Vergleich zur Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul zeigen sich keine signifikanten Unterschiede in den Erkennungsraten (ER) und Wortfehlerraten (WER). Dies ist bemerkenswert, denn durch das 3D-Mean-Shift-Tracking werden alle bewegten Objekte der beobachteten Szene verfolgt und es muss zusätzlich das aktionsausführende Objekt geschätzt werden. Da es keine signifikanten Unterschiede in den Ergebnissen gibt, gelingt diese Schätzung offensichtlich sehr gut. Dies ist eine wichtige Aussage, auch für zukünftige Sicherheitssysteme.

Die Wortfehlerrate erlaubt aber noch einen tieferen Blick in das Verhalten des Aktionserkennungssystems. Über alle Arbeitsaktionen gesehen, tragen die Löschungen $L = 6$ am stärksten zur Wortfehlerrate bei, gefolgt von den Einfügungen $E = 3$. Wiederum ist die Gesamtanzahl der Vertauschungen ($V = 1$) sehr gering. Im Vergleich zur Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul gibt es aber insgesamt mehr Verdopplungen $D = 10$. Dies liegt an der Schätzung des relevanten Objekts. Teilweise gibt es mehrere getrackte Objekte auf der Hand der Testperson. Die Schätzung des relevanten Objekts kann also zwischen diesen Objekten variieren, was die Aktionserkennung kurzzeitig unterbrechen kann. Die einzelnen Fehler bei der Erkennung lassen sich auf Abweichungen der ausgeführten Bewegungen der Testpersonen von der Lernstichprobe und kleinen Ungenauigkeiten beim Tracking zurückführen.

In Tabelle 9.3 sind außerdem die Ergebnisse der Aktionserkennung auf Basis der 20 Testsequenzen für die zeitliche Genauigkeit der Erkennung dargestellt. Es ist zu erkennen, dass die Arbeitsaktionen mit einem zeitlichen Versatz von wenigen Zehntelsekunden erkannt werden können. Im Vergleich zur Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul zeigt sich bei den Mittelwerten des zeitlichen Versatzes zum Beginn und am Ende des Zeitbereichs kein signifikanter Unterschied. Die Mittelwerte sind teilweise klei-

ner, dafür sind aber die Standardabweichungen des zeitlichen Versatzes erhöht. Dies zeigt sich besonders für die Arbeitsaktion „Putzen“. Hier ist die Standardabweichung des zeitlichen Versatzes um das Doppelte (zum Beginn) und um das Sechsfache (am Ende) erhöht.

Wiederum ist das Aktionserkennungssystem in der Lage, unbekannte Bewegungen, z.B. „Nase fassen“, auszuschließen und die Erkennung nach Fortsetzung des zyklischen Arbeitsablaufs durch die Testperson fortzuführen.

Die mittlere Laufzeit der Matlab-Implementierung der Aktionserkennung auf Basis aller Testsequenzen beträgt 5.8 Sekunden pro Zeitschritt auf einem Notebook mit einem Intel Core 2 Duo Prozessor mit 2.4 GHz. Dabei wird die Laufzeit des Trackingsystems nicht berücksichtigt, sondern nur die Laufzeit des Aktionserkennungssystems betrachtet. Der Unterschied zur Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul ist durch die höhere Anzahl an getrackten Objekten zu begründen. Durch eine effektive C-Implementierung sind starke Verbesserungen im Laufzeitverhalten zu erwarten.

Abbildung 9.4 zeigt ein Beispielergebnis der Aktionserkennung auf Basis des 3D-Mean-Shift-Trackings. Es sind alle getrackten Objekte des 3D-Mean-Shift-Trackings mit ihren zugehörigen Objektindizes dargestellt (oben links). Oben rechts wird die ins Bild projizierte Bewegungs- und Erkennungshistorie des als relevant geschätzten Objekts aufgezeigt. Unten links ist die Partikelverteilung im HMM dargestellt. Unten rechts ist die Ausgabe der Level 1 und 2 aufgezeigt. Der normalisierte Diskriminanzwert $\tilde{d}_{\text{unknown}}$ (cyan) beschreibt Level 1 zu welchem Zeitschritt das Aktionserkennungssystem in den Sicherheitsmodus geht. Außerdem sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren für die Aktionen „Transferbewegung“ (blau), „Schrauben 1“ (rot), „Schrauben 2“ (schwarz), „Putzen“ (grün) und „Stecken“ (braun) dargestellt. Die Ausgabe des Levels 2 ergibt sich aus der a posteriori Wahrscheinlichkeit des Partikelfilters und ist ganz unten dargestellt für die Hidden-States des HMMs. Das Aktionserkennungssystem befindet sich zum letzten gezeigten Zeitschritt im Hidden-State „Schrauben 1“ (rot), zuvor wurde ein „Transfer“ (blau) und eine unbekannte Bewegung (cyan) erkannt.

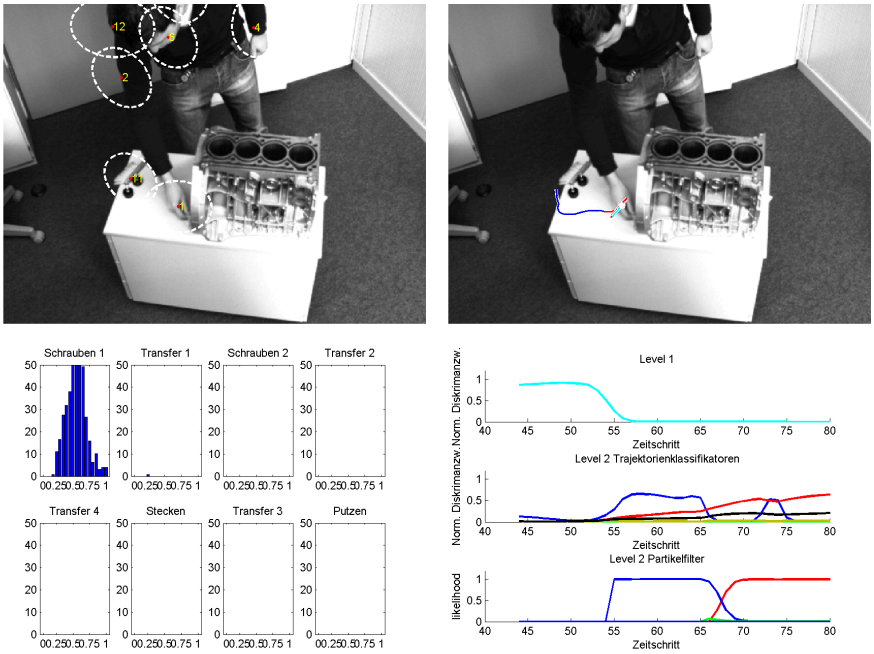

Abbildung 9.4: Beispielergebnis der Aktionserkennung auf Basis des 3D-Mean-Shift-Trackings. Es werden alle getrackten Objekte, das daraus als relevant geschätzte Objekt, die Partikelverteilung im HMM (Histogramme für jede Aktion, Phase auf der Abszissenachse) sowie die Ausgabe der Level 1 und 2 gezeigt. Weiterhin sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren und die aus dem Partikelfilter abgeleitete Wahrscheinlichkeit für die Hidden-States des HMMs dargestellt. Zum letzten gezeigten Zeitschritt wird dir Arbeitsaktion „Schrauben 1“ (rot) erkannt, zuvor wurde ein „Transfer“ (blau) und eine unbekannte Bewegung (cyan) erkannt.

Ein weiteres Beispiel der Erkennung ist in Abbildung 9.5 dargestellt. Das Aktionserkennungssystem befindet sich gerade im Hidden-State „Putzen“ (grün), zuvor wurde ein „Transfer“ (blau) erkannt. Das als relevant geschätzte Objekt ist der Ellipsoid auf der Hand.

9.3 Wertung der Ergebnisse

In diesem Kapitel wurden die Erkennungsergebnisse des in dieser Arbeit beschriebenen Aktionserkennungssystems vorgestellt. In diesem Abschnitt soll eine Wertung vorgenommen und die erzielten Ergebnisse mit denen anderer Verfahren aus dem Stand der Technik (Kapitel 2) verglichen werden.

Der zur Evaluierung der Aktionserkennung verwendete Testdatensatz aus realen Bilder, berücksichtigt die vorgegebenen Randbedingungen (Abschnitt 8.1) einer Aktionserkennung im Produktionsszenario.

Die Ergebnisse wurden auf verschiedenen realen Testsequenzen unter Nutzung zweier unterschiedlicher 3D-Trackingsysteme als Trajektoriengeneratoren berechnet. Zum einen wurden die für die Aktionserkennung notwendigen 3D-Trajektorien vom Trackingsystem 4 (Shape-Flow-Tracking mit Reinitialisierungsmodul) aus Kapitel 4 berechnet. Zum anderen wurden

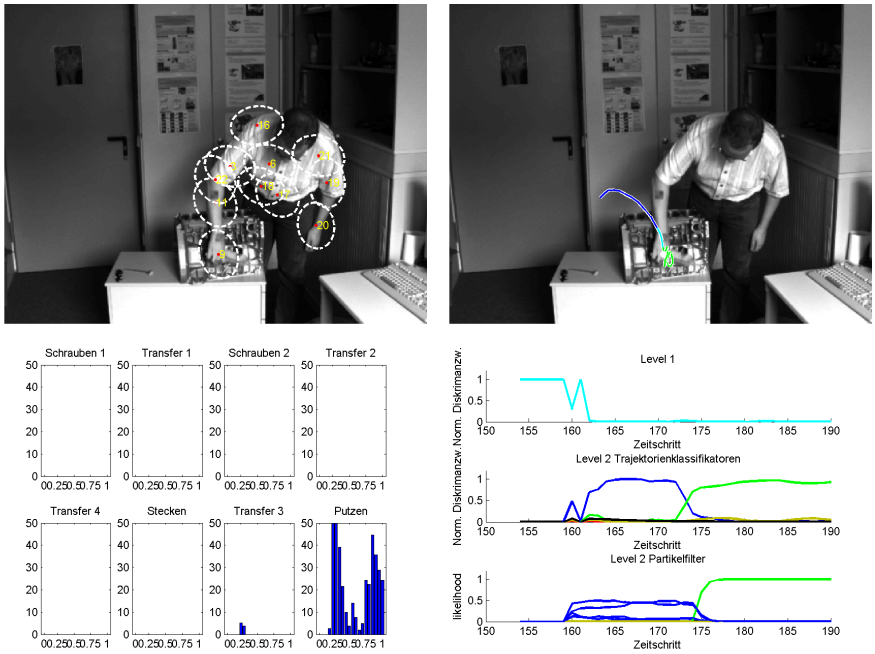

Abbildung 9.5: Beispielergebnis der Aktionserkennung auf Basis des 3D-Mean-Shift-Trackings. Es werden alle getrackten Objekte, das daraus als relevant geschätzte Objekt, die Partikelverteilung im HMM (Histogramme für jede Aktion, Phase auf der Abszissenachse) sowie die Ausgabe der Level 1 und 2 gezeigt. Weiterhin sind die normierten Diskriminanzwerte der Trajektorienklassifikatoren ausgegeben. Außerdem wird die aus dem Partikelfilter abgeleitete Wahrscheinlichkeit für die Hidden-States des HMMs dargestellt. Zum letzten dargestellten Zeitschritt wurde die Arbeitsaktion „Putzen“ (grün) erkannt, zuvor wurde ein „Transfer“ (blau) detektiert.

die aktionsspezifischen Merkmale mit Trackingsystem 5 (3D-Mean-Shift-Tracking) aus Kapitel 5 erzeugt. Beide Trackingsysteme unterscheiden sich stark voneinander. Trotzdem zeigt sich, dass die entwickelte Aktionserkennung generell verwendbar ist, denn die Ergebnisse der Aktionserkennung auf Basis der zwei 3D-Trackingsysteme unterscheiden sich nicht signifikant voneinander. Die qualitativen und quantitativen Ergebnisse der beiden 3D-Trackingsysteme wurden in Kapitel 7 vorgestellt. Beide Verfahren erlauben eine zeitlich robuste und metrisch genaue bildbasierte Verfolgung von menschlichen Körperteilen. Daraus kann abgeleitet werden, dass für ein Aktionserkennungssystem mit einer derart geringen Lernstichprobe ein genaues und zeitlich robustes 3D-Tracking unerlässlich ist. Das Trackingsystem muss in der Lage sein, auch bei kleinen Umkehrbewegungen, z.B. beim Festziehen einer Schraube, oder bei Bewegungen von der Kamera weg eine stabile bildbasierte Verfolgung zu ermöglichen.

Die Ergebnisse des Aktionserkennungssystems zeigen, dass auch bei einer im Vergleich zum Testdatensatz sehr kleinen Lernstichprobe hohe Erkennungsraten erreicht werden können. Dabei werden sogar geringe Wortfehlerraten erzielt. Die zeitliche Genauigkeit der Erkennung ist dabei auf wenige Zehntelsekunden genau.

Das entwickelte Aktionserkennungssystem ist als Sicherheitssystem realisiert, daher ist es in der Lage, auch vom bekannten zyklischen Ablauf abweichende Bewegungsmuster zu erkennen. Da das Aktionserkennungssystem als Zwei-Schichten-Architektur implementiert ist, würde bei

einer unbekannten Bewegung die Zustandsschätzung im HMM mit dem Partikelfilter nicht durchgeführt. Dadurch kann bei einem „Wiedereintreten“ in den zyklischen Arbeitsablauf, dieser fortgesetzt werden. Durch die Zustandsschätzung im HMM mit einem Partikelfilter, können auch falsche Ausführungen der Arbeitsablaufs behandelt und erkannt werden, da immer wieder in alle Hidden-States des HMMs Partikel gestreut werden. Dies wurde auch in den Ergebnissen gezeigt, denn einige der Testsequenzen enthalten absichtlich fehlerhafte Ausführungen des Prozesses.

Ein Vorteil des entwickelten Systems ist die Flexibilität, denn Aktionen im HMM sind einfach austauschbar. Dabei ist kein aufwendiger Trainingsprozess notwendig. Außerdem kann das Aktionserkennungssystem einfach erweitert werden, dazu muss ein neues Interaktionsobjekt definiert, Referenztrajektorien (Aktionsprototypen) manuell extrahiert, die Struktur des HMMs erweitert und die Trajektorienklassifikatoren neu trainiert werden. Dabei ist es ausreichend, wenn wenige Trainingsbeispiele $(2 - 5)$ vorhanden sind.

Ein weiterer Vorteil des entwickelten Aktionserkennungssystems ist, dass das Interaktionsobjekt, in dieser Arbeit ein Motorblock, an einer beliebigen Position im Raum stehen kann oder das Kamerasystem versetzt werden kann. Das Aktionserkennungssystems muss auch dann nicht neu trainiert werden. Dadurch wäre auch eine Aktionserkennung auf einem Fließband möglich. Dazu müsste aber das Interaktionsobjekt zu jedem Zeitschritt neu detektiert werden, beispielsweise durch den multiokularen Template-Matching-Ansatz von Krüger (2007).

Die Grenzen des entwickelten Verfahrens liegen unter anderem in der manuellen Auswahl der Referenztrajektorien, hier könnte man durch ein automatisches Verfahren einigen Arbeitsaufwand sparen. Außerdem würden Probleme auftreten, wenn die 3D-Interaktionsobjekte zweier Aktionsklassen sehr nah, weniger als 50 mm, beieinander sind. Die Klassentrennung durch die Trajektorienklassifikatoren wäre dann erschwert. Falls die zugeordneten Referenztrajektorien außerdem ähnlich sind, ist eine Trennung zwischen diesen Aktionsklassen nur schwer möglich. Dies könnte der Fall sein, wenn beispielsweise zwei unterschiedliche Schraubvorgänge nah beieinander ausgeführt werden. Wenn auf das zufällige Streuen der Partikel (Abschnitt 8.8.2) verzichtet würde, könnten die zwei Aktionsklassen über die Reihenfolge im zyklischen Arbeitsablauf getrennt werden, nur würde dann die Robustheit des Aktionserkennungssystems leiden.

Ergebnisse im Vergleich zum Stand der Technik

In diesem Abschnitt sollen die erzielten Ergebnisse mit den Resultaten anderer Arbeiten aus dem Stand der Technik (Kapitel 2) verglichen werden. Ein solcher Vergleich ermöglicht eine weitere Wertung der erzielten Ergebnisse des Aktionserkennungssystems, ist aber nicht einfach möglich, da das Szenario dieser Arbeit sehr speziell ist und wenige Autoren vergleichbare quantitative Ergebnisse ihrer Verfahren veröffentlichen.

Campbell u. a. (1996) verwenden Bewegungstrajektorien von Kopf und Händen, um T´ai-Chi Bewegungen zu erkennen. Der Ansatz basiert auf einem 3D-Tracking. Zur Erkennung werden 18 HMMs verwendet, da es 18 verschiedene Gesten gibt. In der Arbeit werden verschiedene aus 3D-Trajektorien abgeleitete Merkmale zur Klassifikation miteinander verglichen. Lern- und Testdatensatz sind gleich groß und beinhalten nur die Bewegungsdaten einer Person, die Gestenerkennung wird also sehr speziell auf diese eine Person angepasst. Es werden

Erkennungsraten von 87% bis 94% erreicht. Dies entspricht in etwa den erzielten Ergebnissen dieser Arbeit. Mit dem Unterschied, dass in dieser Arbeit der Lerndatensatz im Vergleich zum Testset um den Faktor 10 geringer ist und im Testset die Bewegungen von acht verschiedenen Testpersonen enthalten sind. Weiterhin wird durch Campbell u. a. (1996) gezeigt, dass aus den 3D-Positionstrajektorien abgeleitete translations- und rotationsinvariante Merkmale die besten Ergebnisse liefern. Eine andere Erkenntnis der Arbeit ist, dass die Kopfposition zur Erkennung von T´ai-Chi Bewegungen nicht notwendig ist.

Wie bereits im Kapitel 8 beschrieben, wurde die entwickelte Aktionserkennung von der Arbeit von Black u. Jepson (1998) beeinflusst und inspiriert. Leider werden in dieser Arbeit keine Erkennungsraten genannt. Ein Vergleich ist aber trotzdem möglich, da Fritsch u. a. (2004) sowie Hofemann (2007) den Ansatz von Black u. Jepson (1998) aufgreifen und erweitern.

Fritsch u. a. (2004) evaluiert das weiterentwickelte *Condensation-based Trajectory Recognition* (CTR) Verfahren auf einem Testdatensatz mit 60 Beispielen pro Aktionsklasse. Der Datensatz wurde durch sechs verschiedene Testpersonen erzeugt. Leider wird nicht angegeben wie groß die Lernstichprobe war. Über alle Aktionen gemittelt ergab sich eine Erkennungsrate von 93.3%, was vergleichbar mit den in dieser Arbeit erzielten Ergebnissen ist. Ein Nachteil des Systems von Fritsch u. a. (2004) ist, dass es auf 2D-Trajektorien basiert. Bei 2D-Trajektorien als aktionsspezifische Merkmale können Mehrdeutigkeiten auftreten, z.B. können zwei gleiche Aktionen in unterschiedlicher Tiefe nicht unterschieden werden oder zwei verschiedenartige Aktionen in unterschiedlicher Tiefe gleich aussehen. Ein weiterer Nachteil des Ansatzes von Fritsch u. a. (2004) ist, dass deren Verfahren nicht unabhängig vom Kamerastandpunkt ist, bei einer Änderung der Kameraposition muss daher neu trainiert werden.

Von Hofemann (2007) wurde das CTR-Verfahren Black u. Jepson (1998) und der Ansatz von Fritsch u. a. (2004) durch die Verwendung von 3D-Information weiterentwickelt. Als aktionsspezifisches Merkmal werden Trajektorien von Körperkonfigurationen (Abschnitt 2.3.4) verwendet. Dadurch wird eine Unabhängigkeit zum Kamerastandpunkt erreicht, denn die Trajektorien sind nur von den durch das Körpertracking geschätzten Parametern abhängig, z.B. Gelenkwinkeln. Das Verfahren kommt mit wenig Trainingsdaten aus. Es wird angegeben, dass für jedes Aktionsmodell zwischen fünf und 20 Trainingsbewegungen vorhanden sein sollten. Im Testdatensatz sind fünf verschiedene Gesten enthalten, insgesamt gibt es 496 Beispiele. Diese wurden von fünf verschiedenen Testpersonen aufgezeichnet. Auf dem Testdatensatz wird eine Fehlerrate von 17.34% erreicht, was etwas schlechter ist als die in dieser Arbeit erreichte Wortfehlerrate. Von Hofemann (2007) wird eine Erkennungsrate von 94.56% angegeben, was in etwa den in dieser Arbeit erzielten Ergebnissen entspricht. Auffällig ist die erhöhte Anzahl doppelt erkannter Gesten in der Arbeit von Hofemann (2007). Hier schneidet das in dieser Arbeit entwickelte Verfahren besser ab. Dies könnte an den verwendeten Trackingsansätzen liegen, denn diese sind metrisch sehr genau und zeitlich robust (Kapitel 7), was eine dauerhafte Erkennung ohne Unterbrechungen ermöglichen kann.

Auch die Arbeit von Li u. a. (2006) hat die entwickelte Aktionserkennung beeinflusst und inspiriert. Denn durch Li u. a. (2006) wurde erstmals ein Partikelfilter zur Zustandsschätzung in einem HMM eingesetzt. Durch Li u. a. (2006) werden drei Aktionen erkannt, diese sind „Blume gießen“, „Tee zubereiten“ und „Kaffee zubereiten“. Der verwendete Datensatz besteht insgesamt aus 108 Beispielen. Für jede Aktion gibt es 36 Beispiele, welche von acht verschiedenen Testpersonen stammen. Zum Training der HMMs werden pro Aktion 20 Beispiele verwendet.

Die übrigen 16 Beispiele pro Aktion werden zum Test verwendet. Eine Erkennungsrate ist in der Arbeit von Li u. a. (2006) nicht angegeben, dafür aber die Rate der falsch positiven Erkennungen und die Vertauschungsrate. Im Mittel über alle drei Aktionen wird eine Rate der falsch positiven Erkennungen (Löschungsrate) von 18.13% erreicht. Die Rate der falsch positiven Erkennungen ist definiert als $L/(L + K)$, wobei die Anzahl der Löschungen (L) durch die Summe der Löschungen (L) und korrekt erkannten Aktionen (K) dividiert wird. Im Mittel über alle Aktionen wird für die Aktionserkennung auf Basis des Shape-Flow-Trackings mit Reinitialisierungsmodul eine Rate der falsch positiven Erkennungen von 4.5% erreicht, was deutlich besser ist als die erzielten Ergebnisse von Li u. a. (2006). Auch beim Vergleich der mittleren Vertauschungsraten schneidet der in dieser Arbeit beschriebene Ansatz etwas besser ab: 1% zu 3.7%. Ein solcher Vergleich ist natürlich immer schwer möglich, da es sich um völlig unterschiedliche Datensätze handelt. Außerdem arbeitet der Ansatz von Li u. a. (2006) auf Basis eines 2D-Trackings und die in dieser Arbeit beschriebene Aktionserkennung auf Basis eines 3D-Trackings. Ein klar ersichtlicher Vorteil des in dieser Arbeit beschriebenen Systems ist aber, dass es mit einem im Vergleich zum Testdatensatz sehr kleinen Lerndatensatz zurecht kommt. Wiederum begründet sich dieser Vorteil durch das metrisch genaue und zeitlich robuste 3D-Tracking.

Teil IV

Zusammenfassung, Ausblick und Anhang

Im vierten Teil dieser Dissertation werden die Ziele, Aussagen, Ansätze, Verfahren und Ergebnisse aus den ersten drei Teilen dieser Arbeit zusammengefasst. Dabei wird außerdem darauf eingegangen wie weit die angestrebten Ziele aus dem Einleitungsteil erreicht wurden und welche offenen Fragen es weiterhin gibt. Zudem wird ein Ausblick gegeben und Ideen vorgestellt, die durch den Autor bisher noch nicht behandelt werden konnten. Abschließend werden im Anhang A die theoretischen Grundlagen zu einigen in dieser Arbeit verwendeten Ansätzen vorgestellt und im Anhang B das Schriftenverzeichnis des Autors dieser Arbeit wiedergegeben.

Kapitel 10

Zusammenfassung und Ausblick

Automobilhersteller sind durch einen stetigen Anstieg der Produktvariationen und einer Verkürzung der Produktlebenszyklen einem immer stärkeren Kosten-, Qualitäts- und Zeitdruck ausgesetzt. Eine Interaktion von Mensch und Industrieroboter im Umfeld von Produktionsprozessen im Automobilbau eröffnet eine Vielzahl von Möglichkeiten, um die Produktivität, Flexibilität und Produktqualität weiter erhöhen zu können. Ziel einer Zusammenarbeit zwischen Mensch und Industrieroboter ist es, die Stärken beider optimal zu nutzen und den Automatisierungsgrad weiter zu erhöhen. Denn einige vollständig manuelle Prozesse könnten durch intelligente technische Systeme teilweise automatisiert werden.

Eine zukünftige sichere und produktive Zusammenarbeit von Mensch und Industrieroboter erfordert intelligente, zuverlässige und flexible Systeme, welche die Interaktion zwischen Mensch und Roboter überwachen, um die Gefahren für den Menschen zu reduzieren. Eine direkte physische Interaktion zwischen Mensch und Industrieroboter ist nicht ohne Risiko, denn ein Zusammenstoß könnte verheerende Folgen für den Menschen nach sich ziehen.

Das seit 2008 im Automobilbau als Schutzsystem eingesetzte Kamerasystem SafetyEYE[1] ermöglicht eine ganzheitliche Überwachung von Arbeiter und Industrieroboter. Bisher erkennt das SafetyEYE-System, ob sich irgendeine Form von Materie im zuvor definierten Schutzraum befindet, berücksichtigt aber nicht die detaillierte Beschreibung eines Objekts bzw. dessen Eigenschaften oder ob dieses Objekt Aktionen ausführt. Im Fokus dieser Arbeit stand daher als erstes Schwerpunktthema die Entwicklung neuartiger Methoden zur Segmentierung, Verfolgung und Bewegungsvorhersage von mit der Kamera beobachteten Objekten. Als zweites Schwerpunktthema wurde die Erkennung von Aktionen, Handlungen und Handlungsabsichten, welche von den beobachteten Objekten ausgehen, betrachtet. Durch die in dieser Arbeit beschriebenen neuartigen Methoden für beide Schwerpunktthemen wurde es möglich gemacht, der Vision „Sichere Mensch-Industrieroboter-Interaktion“ mit einem intelligenten technischen Überwachungssystem einen Schritt näher zu kommen.

Ein allgemeines Modell (Abbildung 10.1 (oben)) eines technischen Systems zur Realisierung des Interaktionsprozesses besteht aus drei Hauptkomponenten. Durch einen Eingabesensor, in dieser Arbeit ein trinokulares Kamerasystem, werden Daten der überwachten Szene geliefert. In einem zweiten Modul werden die am Prozess beteiligten menschlichen Körperteile in den Bildfolgen des Kamerasystems detektiert, segmentiert und über die Zeit verfolgt. Die berechneten Bewegungsdaten oder daraus extrahierte Merkmale dienen einem Aktionserkennungsmodul als Eingabedaten. Ausgehend vom allgemeinen Modell (Abbildung 10.1 (oben))

[1] www.safetyeye.com, (Winkler, 2006)

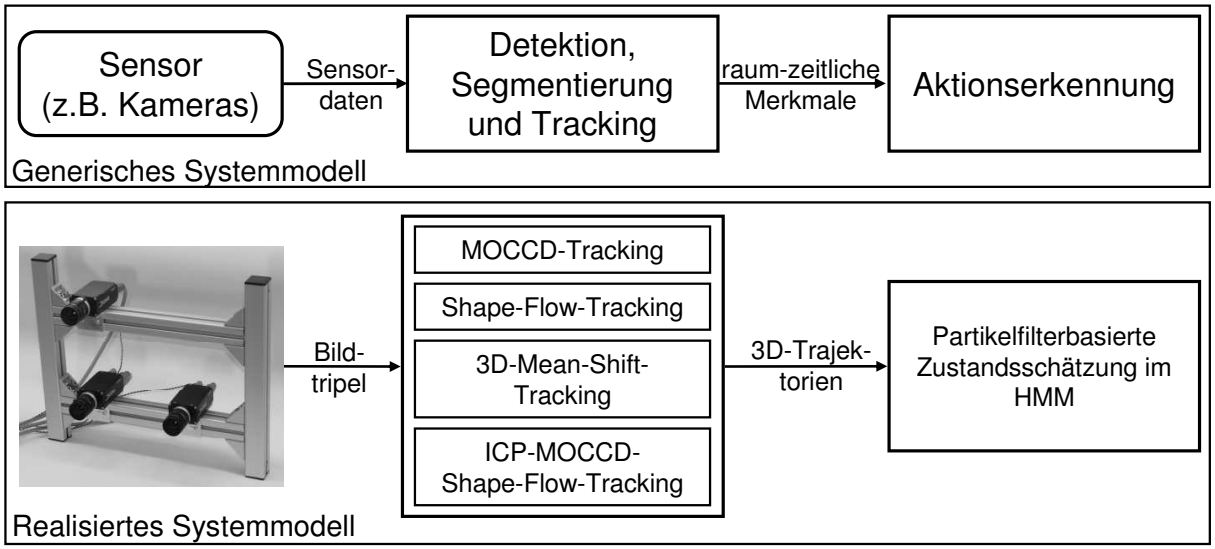

Abbildung 10.1: Oben: Ein allgemeines Modell eines technischen Systems zur Verfolgung von Objekten und Erkennung von Aktionen. Unten: Das realisierte Systemmodell.

wurde in dieser Arbeit im Kapitel 2 ein Überblick über vorhandene Ansätze zur Realisierung der beiden zentralen Module gegeben. Diese entsprechen den Schwerpunktthemen.

Wie das allgemeine Modell gliedert sich auch das realisiere System (Abbildung 10.1 (unten)) in drei Hauptkomponenten. Der Eingabesensor ist ein Mehrkamerasystem mit kleiner Basisbreite. Die kleine Basisbreite und rechtwinklige Anordnung der drei Sensoren des Kamerasystems wurde durch das SafetyEYE-System vorgegeben. Hinsichtlich der Genauigkeit ist ein solches trinokulares Kamerasystem einem monokularen Kamerasystem überlegen und die Investitionskosten sowie der Hardware- und Kalibrieraufwand ist im Gegensatz zu einem Kamerasystem mit großer Basisbreite überschaubar. Außerdem lässt sich mit drei zueinander senkrecht angeordneten Kameras das Aperturproblem für Stereo-Algorithmen lösen. Probleme bei der Korrespondezsuche für Bildbereiche, welche eine ähnliche Orientierung wie eines der Stereo-Kamerapaare besitzen, werden durch das jeweils andere Kamerapaar aufgelöst.

Für das zweite Modul Detektion, Segmentierung und Verfolgung von mit den Kameras beobachteten Objekten, wurden in dieser Arbeit vier verschiedene Realisierungen beschrieben, wie in Abbildung 10.1 (unten) im mittleren Modulkasten zu sehen ist. Alle Trackingsysteme verwenden als Eingabedaten die Bildtripel des Kamerasystems.

Mit MOCCD-Tracking wird ein technisches System zur Verfolgung von 3D-Konturen bezeichnet. Dieses wurde im Kapitel 3 vorgestellt. Beim MOCCD-Tracking wird das in den Kamerabildern verfolgte menschliche Körperteil durch ein 3D-Konturmodell beschrieben und die Anpassung des 3D-Konturmodells an die Bilddaten wird mit Hilfe des multiokularen Contracting-Curve-Density (MOCCD) Algorithmus durchgeführt. Der zum Teil neu entwickelte und verbesserte Bayes'sche Ansatz des MOCCD-Algorithmus erlaubt eine stabile Pose-Estimation auch bei erheblichem Bildrauschen und stark strukturierten Hintergründen. Im Vergleich zu vielen etablierten 3D-Pose-Estimation-Verfahren wurde in dieser Arbeit neben der eigentlichen 3D-Pose-Estimation auch eine Schätzung der Modellgeometrie durchgeführt. Dazu wurden die Deformationsparameter (Geometrieparameter) des 3D-Konturmodells in den

zu optimierenden Parametervektor aufgenommen (Abschnitt 3.2). Dies ist für das angestrebte Szenario dieser Arbeit besonders wichtig, da nicht davon ausgegangen werden kann, dass ein personenspezifisches 3D-Modell des Arbeiters vorliegt. Das entwickelte Verfahren ist daher in der Lage, sich an das Aussehen unterschiedlicher Arbeiter anzupassen und gewährleistet auch dann ein robustes Tracking, wenn der Arbeiter spezielle Arbeitskleidung trägt, z.B. Handschuhe. Das Systemmodell des MOCCD-Trackings ist als Multi-Hypothesen-Trackingsystem aufgebaut. Die Schätzung der 3D-Pose, also die Anpassung des 3D-Konturmodells an die Kamerabilder, wird durch drei MOCCD-Algorithmen bestimmt. Die notwendige 3D-Pose-Prädiktion über die Zeit, wird durch drei zu den MOCCD-Algorithmen zugehörige Kalman-Filter berechnet. In jedem der drei Kalman-Filter ist ein unterschiedliches Bewegungsmodell implementiert. Die für ein Sicherheitssystem notwendige zeitliche Robustheit und ein Selbsttest des Trackingsystems muss durch andere Module sichergestellt werden, was in den experimentellen Untersuchungen (Kapitel 7) gezeigt wurde. Durch ein Modul zur Verifikation und eine Datenbank in der das vorherige Aussehen des verfolgten Objekt gespeichert wurde, wird die zeitliche Konsistenz des Objektaussehens überprüft. Wird dieser Selbsttest nicht bestanden, wird mit einem Modul zur Reinitialisierung, welches durch zwei lokale Suchalgorithmen realisiert wurde, versucht das getrackte Objekt wiederzufinden.

Das zweite System im mittleren Modulkasten in Abbildung 10.1 (unten) ist das sogenannte Shape-Flow-Tracking, welches eine raum-zeitliche Verfolgung von 3D-Konturen realisiert und im Kapitel 4 beschrieben wurde. Dabei ist es wichtig, dass das getrackte menschliche Körperteil durch ein raum-zeitliches 3D-Konturmodell beschrieben werden kann. Die drei zeitlichen Filter aus dem vorherigen MOCCD-Tracking wurden beim Shape-Flow-Tracking durch eine direkt gemessene Bewegungsschätzung ersetzt, welche durch den Shape-Flow-Algorithmus realisiert wird. Dieser neuartige top-down Ansatz zur modellbasierten raum-zeitlichen 3D-Pose-Estimation wurde durch die Erweiterung des MOCCD-Algorithmus auf Bildfolgen möglich. Damit kann nun direkt, also instantan, aus drei aufeinander folgenden Bildern ein dichter 3D-Szenenfluss für das getrackte Objekt berechnet werden. Es werden im Vergleich zum MOCCD-Tracking nur zwei Hypothesen zum Tracking verwendet. Die 3D-Konturanpassung wird mit dem MOCCD-Algorithmus durchgeführt und die Objektbewegungsschätzung mit dem Shape-Flow-Algorithmus realisiert. Wie beim MOCCD-Tracking wird durch ein Modul zur Verifikation die zeitliche Konsistenz des Objektaussehens überprüft. Außerdem kann zusätzlich auch ein Reinitialisierungsmodul verwendet werden. In den Experimenten zeigte sich, dass die instantane, also direkte, Bewegungsschätzung mit dem Shape-Flow-Algorithmus der direkten Bewegungsschätzung beim MOCCD-Tracking überlegen ist. Eine instantane Bewegungsschätzung ist bei einem Sicherheitssystem immens wichtig, denn die Latenzzeit einer möglichen Reaktion auf die Bewegungsvorhersage muss möglichst gering gehalten werden. Der neu entwickelte Shape-Flow-Algorithmus ermöglicht eine sehr genaue und robuste instantane Bewegungsvorhersage aus nur drei aufeinander folgenden Bildern. Der Einsatz eines zeitlichen Filters, z.B. eines Kalman-Filters, würde keine Verbesserungen bewirken, denn die Bewegung der menschlichen Hand ist sehr komplex, schnell, kann abrupt stoppen oder die Richtung umkehren. Ein zeitliches Filter würde die Latenzzeit einer möglichen Reaktion auf die Bewegungsvorhersage nur erhöhen und nicht genauer machen, denn aufgrund der komplexen Bewegung würde es immer wieder Einschwingphasen des zeitlichen Filters geben.

Das dritte Trackingsystem in Abbildung 10.1 (unten) ist das im Kapitel 5 beschriebene

3D-Mean-Shift-Tracking. Dabei werden die in den Kamerabildern getrackten Objekte nicht detailliert beschrieben, sondern durch Ellipsoide modelliert. Falls ein getracktes Objekt nicht mit nur einem Ellipsoiden beschrieben werden kann, werden mehrere Ellipsoide zur Beschreibung des Objekts eingesetzt und diese getrennt voneinander in den Bildern verfolgt. Das System verwendet zur Pose-Estimation keine Konturen sondern Bilddaten und 3D-Punktewolken mit Bewegungsinformation. Diese sind Input für die Pose-Estimation und außerdem wichtig, um neue Objekte zu detektieren oder verlorene Objekte wiederzufinden. Ein großer Vorteil ist die automatische Objektdetektion und dass kein Körperteil modelliert werden muss, sondern alle bewegten Objekte verfolgt werden. Das 3D-Mean-Shift-Tracking ist ein genereller Trackingansatz, denn es wurden auch Ergebnisse des Trackings von Verkehrsteilnehmern in einem Kreisverkehrsszenario vorgestellt. Außerdem ist das entwickelte System zeitlich robust, metrisch genau und sehr rechenzeiteffizient, was die Anwendung in einem Sicherheitssystem interessant macht.

Das vierte entwickelte Trackingsystem in Abbildung 10.1 (unten) ist ein Verfahren zur Objektverfolgung durch 3D-Pose-Estimation basierend auf raum-zeitlichen Konturen und 3D-Punktewolken, welches im Kapitel 6 beschrieben wurde. Die verwendete Bezeichnung ICP-MOCCD-Shape-Flow-Tracking leitet sich aus den Akronymen der eingesetzten Algorithmen ab. Im entwickelten Trackingsystem wurden zwei unterschiedliche Ansätze zur 3D-Pose-Estimation miteinander kombiniert. Zum einen ist dies der Iterative-Closest-Point (ICP) Algorithmus, welcher basierend auf bewegten und geclusterten Szenenflussdaten das verwendete 3D-Modell anpasst. Zum anderen wird der MOCCD-Algorithmus eingesetzt, um das Objektmodell an die Bilddaten anzupassen. Ein Vorteil der verwendeten Ansätze ist, dass sie auf unterschiedlichen Eingabedaten basieren und somit auch einen Ausfall des jeweils anderen Verfahrens kompensieren können. Die Fusion der beiden Pose-Schätzungen basiert auf drei verschiedenen Ähnlichkeitsmaßen. Durch eine Bewegungsanalyse basierend auf Szenenflussdaten und die Anwendung des Shape-Flow-Algorithmus wird die Position des getrackten Objekts über die Zeit vorhergesagt. Durch dieses Trackingsystem kann instantan die 3D-Pose und deren vollständige Ableitung nach der Zeit aus den Bilddaten zu drei aufeinander folgenden Zeitschritten sehr genau geschätzt werden.

Alle Realisierungen haben gemeinsam, dass sie allgemein verwendbar sind, d.h. nicht auf ein spezifisches Objekt angepasst wurden und das Objektmodell einfach austauschbar ist. Daher können sie auf beliebigen Objektformen und in unterschiedlichen Szenarien angewendet werden. Dies wurde im Kapitel 7 in einer sehr ausführlichen und qualitativen sowie quantitativen Evaluierung gezeigt. Es wurden Ergebnisse der Verfolgung des menschlichen Hand-Unterarm-Bereichs und der Kopf-Schulter-Partie vorgestellt und nachgewiesen, dass die entwickelten Realisierungen zur kamerabasierten Detektion, Segmentierung und Verfolgung von Objekten metrisch genau und zeitlich robust sind. Dies gilt auch, wenn das Zielobjekt teilweise oder ganz verdeckt ist, es Belichtungsänderungen gibt, das Zielobjekt sich unerwartet bewegt, das Zielobjekt komplex geformt ist oder sich die Objektform verändert, z.B. durch ein Greifen eines Gegenstands. Weiterhin sind alle Verfahren in der Lage, Objektbewegungen vor strukturierten oder dem Zielobjekt ähnlichen Hintergründen zu tracken. Jedes der entwickelten Systeme ermöglicht in seiner besten Variante das bereits erwähnte, metrisch genaue und zeitlich robuste Tracking sogar auf Grauwertbildern. Daraus wurde abgeleitet, dass die Verwendung einer Grauwertkamera ausreichend ist, womit sich die Laufzeit der Algorithmen verringern lässt und

bei weniger Umgebungslicht gearbeitet werden könnte, was für ein industrielles Sicherheitsszenario sehr interessant ist.

Die entwickelten Trackingsysteme sind so gestaltet worden, dass sie in einem Sicherheitssystem parallel eingesetzt werden könnten, was die zeitliche Robustheit erhöht und Vorteile mit sich bringt, denn die entwickelten Systeme unterscheiden sich teilweise sehr in ihren Eigenschaften sowie Vor- und Nachteilen. Für ein Sicherheitssystem ist dies sehr wichtig, denn auch beim aktuellen SafetyEYE-System laufen zwei unterschiedliche Stereoansätze auf zwei Industrie-PC mit unterschiedlichen Betriebssystemen parallel. Die Unabhängigkeit mehrerer parallel laufender Systeme innerhalb des Sicherheitssystems erhöht die Zuverlässigkeit. Alarm wird ausgelöst, sobald nur einer der Algorithmen eine gefährliche Situation identifiziert hat.

Ziel dieser Dissertation war der Entwurf eines technischen Systems zur ganzheitlichen Interaktion zwischen Mensch und Industrieroboter. D.h. auch das zweite Schwerpunktthema (Abbildung 10.1 (oben)), die Erkennung von Aktionen, Handlungen und Handlungsabsichten, welche von den beobachteten Objekten ausgehen, wurde in dieser Arbeit betrachtet. Damit ist es nun möglich, der Vision „Sichere Mensch-Industrieroboter-Interaktion“ mit einem intelligenten technischen Überwachungssystem näher zu kommen. Wie in Abbildung 10.1 (unten) im rechten Kasten zu sehen ist, gibt es nur eine Realisierung für das Modul zur Aktionserkennung. Dieses Modul wurde aber so gestaltet, dass ein beliebiges 3D-Trackingsystem als Generator der notwendigen aktionsspezifischen Merkmale verwendet werden kann. Als aktionsspezifische Merkmale wurden Trajektorien von 3D-Punkten ausgewählt. Würden 2D-Trajektorien verwendet, könnten manche Aktionen nicht voneinander unterschieden werden, da es Mehrdeutigkeiten gibt. Die Verwendung von 3D-Informationen zieht sich also konsequent durch die vorgestellte Arbeit und die entwickelten intelligenten Überwachungssysteme. 3D-Trajektorien als aktionsspezifische Merkmale bieten eine kompakte und sehr diskriminative Repräsentation der Bewegungsdaten. Außerdem kann die Beziehung der getrackten Bewegung zu bekannten 3D-Interaktionsobjekten hergestellt werden. Dies ist für eine Aktionserkennung im Produktionsszenario besonders wichtig, denn ein großer Anteil möglicher zu erkennender Arbeitsaktionen basiert auf einem 3D-Interaktionsobjekt, in dessen Nähe die Aktion ausgeführt wird oder mit dem die Aktion in Verbindung steht.

Bei einer Aktionserkennung in realen Umgebungen, wie beispielsweise in der Automobilproduktion, gibt es verschiedene Herausforderungen, welche von einem zukünftigen Überwachungssystem beherrscht werden müssen. Das getrackte Objekt, z.B. der Werker, kann sich unerwartet verhalten, kann unkontrollierte Bewegungen ausführen, kann den Arbeitsprozess immer wieder unterbrechen oder diesen in einer unbekannten Reihenfolge durchführen. Diese von der normalen Tätigkeit abweichenden Handlungen müssen erkannt werden, um mögliche Kollisionen mit einem Industrieroboter verhindern zu können. Daher wurde das entwickelte und neuartige Aktionserkennungssystem als Zwei-Schichten-Architektur realisiert. Auf der höheren Ebene, dem Level 1 des Systems, wird durch Trajektorienklassifikatoren entschieden, ob es sich um eine unbekannte oder bekannte Bewegung des getrackten Werkers handelt. Falls eine unbekannte Bewegung registriert wurde, wird der Sicherheitsmodus aktiviert, in diesem können durch die Vorhersage der Bewegung des Werkers mögliche Kollisionen mit dem Industrieroboter oder anderen Objekten erkannt werden.

Im Falle einer bekannten Bewegung befindet sich das System im sogenannten Normalmodus,

d.h. es wird erkannt, dass der Werker den eigentlichen zyklischen Arbeitsprozess ausführt. Nur wenn durch die drei Trajektorienklassifikatoren entschieden wird, dass es sich um ein bekannte Bewegung handelt, wird die Erkennung des aktuellen Zustands im zyklischen Arbeitsprozess durchgeführt. Dazu wurde aus dem vorgegebenen Arbeitsablauf die Struktur eines Hidden-Markov-Modells (HMMs) abgeleitet. Dieses HMM wurde aber nicht im klassischen Sinne trainiert und evaluiert, sondern die Struktur direkt aus dem Arbeitsprozess abgeleitet. Eine weitere Randbedingung einer Aktionserkennung im Produktionsszenario sieht vor, dass nur wenige Trainigsdaten für die Aktionserkennung zur Verfügung stehen. Es ist nicht praktikabel und zu kostenintensiv, viele verschiedene Werker bei ihrer Tätigkeit aufzunehmen, die extrahierten Bewegungsdaten mit Aktionsklassen zu labeln und dann das Erkennungssystem zu trainieren. Es ist aber möglich, die Bewegungsdaten weniger Vorarbeiter, welche den Arbeitsprozess detailliert kennen und die Werker angelernt haben, aus aufgenommenen Kamerabildern zu extrahieren und als Grundlage für das Erkennungssystem zu verwenden. Aufgrund der kleinen Lernstichprobe wurde das HMM nicht im klassischen Sinne aus der Lernstichprobe der Vorarbeiter trainiert, sondern durch Expertenwissen modelliert. Denn ein Training mit einer derart geringen Menge an Trainingsdaten würde ein „Auswendiglernen“ bedeuten. Außerdem wird das HMM während der eigentlichen Aktionserkennung, also der Lösung des Evaluierungsproblems, nicht im klassischen Sinne wie z.B. mit dem Forward-Algorithmus evaluiert, sondern ein Partikelfilter zur Zustandsschätzung im HMM verwendet. Diese bekannte Idee wurde aufgegriffen und verfeinert. Ziel ist es dabei, durch das Partikelfilter den aktuellen Hidden-State des HMMs aus der Beobachtungssequenz zu schätzen, dazu wurden jedem Hidden-State 3D-Trajektorien als Aktionsprototypen zugeordnet und die aktuelle Eingabetrajektorie mit diesen verglichen, um somit die aktuell ausgeführte Aktion zu erkennen. Ein großer Vorteil eines solchen Konzepts ist, dass Aktionen einfach austauschbar sind und die Struktur sowie das Transitionsmodell des HMMs einfach erweiterbar ist.

Neu an der partikelfilterbasierten Zustandsschätzung in einem HMM ist, dass im HMM nichtstationäre Transitionswahrscheinlichkeiten realisiert werden. Im Gegensatz zur Literatur werden die Transitionswahrscheinlichkeiten nicht abhängig von der Dauer im verborgenen Zustand gemacht. Durch die Dauer im Zustand kann nicht berücksichtigt werden wie schnell eine Bewegung ausgeführt wird, was zu Problemen führt, wenn es Unterschiede in der Ausführungsgeschwindigkeit von Aktionen gibt. Das in dieser Arbeit entwickelte System realisiert die nichtstationären Transitionswahrscheinlichkeiten im HMM dadurch, dass im Zustandsvektor des Partikelfilters neben dem aktuellen Hidden-State auch die Bewegungsphase geschätzt wird und die Transitionswahrscheinlichkeiten des HMMs davon abhängig sind. Durch die Phasenschätzung wird elegant der Anteil, zu dem eine Bewegung vollführt wurde, im HMM berücksichtigt und es kommt erst mit einer erhöhten Bewegungsphase zu erhöhten Transitionswahrscheinlichkeiten. Denn erst eine hohe Phase, also beispielsweise größer als 90% , zeigt an, dass die Referenzbewegung weitestgehend durchlaufen ist und der nächste Zustand damit wahrscheinlich wird. Die Bewegungsphase wird in dieser Arbeit auch dazu verwendet, den Startzeitpunkt und das Ende einer Bewegung zu schätzen, denn die beste zugehörige Referenztrajektorie und deren Ausführungsgeschwindigkeit kann bestimmt werden. Außerdem wird mit Hilfe des Partikelfilters der Objektindex des relevanten Objekts aus allen getrackten Objekten geschätzt. Das relevante Objekt ist das getrackte Objekt, welches die zu erkennenden Aktionen ausführt, z.B. die Hand des Werkers. Die Schätzung des relevanten Objekts ist

notwendig, wenn wie beim 3D-Mean-Shift-Tracking mehrere Objekte verfolgt werden, ohne zu wissen, welches das aktionsausführende Objekt ist.

Am Beispiel von Montagearbeiten eines Werkers an einem Motorblock wurde bei den experimentellen Untersuchungen zur Aktionserkennung (Kapitel 9) gezeigt, dass eine robuste Erkennungsleistung unter den Randbedingungen einer Aktionserkennung in der Automobilproduktion möglich ist. Das gewählte Arbeitsszenario ist realitätsnah, denn einfache Montageprozesse in der Automobilproduktion, bei denen als Erstes eine Interaktion zwischen Mensch und Industrieroboter realisiert werden könnte, haben keine höhere Komplexität. Das gewählte Beispiel steht also stellvertretend für andere einfache Montageprozesse, wie z.B. „Handeinlegestationen".

Um zu zeigen, dass die entwickelte Aktionserkennung generell verwendbar ist, wurden die Ergebnisse der Aktionserkennung auf Basis verschiedener Testsequenzen für zwei unterschiedliche Trackingsysteme vorgestellt. Die zur Aktionserkennung verwendeten 3D-Trajektorien stammen zum einen vom Shape-Flow-Tracking aus Kapitel 4. Zum anderen werden die aktionsspezifischen Merkmale mit dem 3D-Mean-Shift-Tracking aus Kapitel 5 erzeugt. Der Unterschied zwischen den Trackingsystemen besteht darin, dass beim Shape-Flow-Tracking, einem modellbasierten Trackingansatz, direkt der rechte Unterarm in den Kamerabildern verfolgt wird. Beim 3D-Mean-Shift-Tracking hingegen, werden alle bewegten Objekte in der beobachteten Szene mit einem einfachen Ellipsoidmodell in den Kamerabildern verfolgt. Dadurch muss zusätzlich geschätzt werden, welches der getrackten Objekte die Aktion ausführt.

Der verwendete Testdatensatz zur Evaluierung der Aktionserkennung berücksichtigt verschiedene Randbedingungen einer Aktionserkennung im Produktionsszenario. Die Lernstichprobe stammt von zwei Testpersonen und diese haben alle weiteren sechs Testpersonen mit dem zyklischen Arbeitsablauf vertraut gemacht. Die beiden Testpersonen aus der Lernstichprobe könnten also eine Art Vorarbeiter, welche den Arbeitsprozess detailliert kennen. Nur auf Basis von gelabelten Bewegungsdaten der beiden Vorarbeiter wurden die Trajektorienklassifikatoren trainiert und die 3D-Trajektorien als Aktionsprototypen für das HMM (Referenztrajektorien) ausgewählt. Durch eine zeitliche Skalierung der Referenztrajektorien wird sichergestellt, dass auch schnellere oder langsamere Bewegungen als die der beiden Vorarbeiter erkannt werden können. Außerdem kann sich in realen Umgebungen das getrackte Objekt, also der Werker, unerwartet Verhalten oder den Arbeitsprozess immer wieder unterbrechen und diesen in einer unbekannten Reihenfolge durchführen. Daher wurden in den Testdatensatz außerdem noch Sequenzen integriert, in denen Fehler im Arbeitsprozess gemacht wurden. Diese Sequenzen stammen aus den Testläufen zum Anlernen der Testpersonen. Auch unter diesen schwierigen Bedingungen lieferte das entwickelte System sehr gute Erkennungsleistungen.

Der entwickelte Ansatz kann zur Identifikation von Gefahren verwendet werden. Denn bei einer Interaktion zwischen Mensch und Industrieroboter ist es besonders wichtig vom Menschen verschuldete Gefahren zu erkennen und zu behandeln, indem die neu entwickelten Technologien in einen definierten sicheren Zustand gehen, z.B. den Industrieroboter stoppen oder dessen Bewegung verlangsamen. Außerdem könnte durch das entwickelte System erkannt werden, dass der Mensch die Arbeitsstation falsch bedient oder Fehler im Arbeitsprozess macht.

Die Ergebnisse im betrachteten realitätsnahen Beispielszenario zeigen, dass durch die in dieser Arbeit beschriebenen Algorithmen einfache Arbeitsprozesse wie z.B. „Handeinlegestationen"ganzheitlich überwacht werden könnten und eine sichere physische Interaktion des

Arbeiters mit dem Industrieroboter, also der „Handeinlegestation“, möglich ist.

Einige Erweiterungen der entwickelten Konzepte sind denkbar hinsichtlich einer Rückkopplung der Ergebnisse der trajektorienbasierten Aktionserkennung, um beispielsweise das Tracking zeitlich stabiler und robuster machen zu können. So könnten beispielsweise Verdeckungen von in den Kamerabildern getrackten Objekten überbrückt werden. Außerdem kann durch Kenntnisse über die aktuell ausgeführten Aktionen eine intelligente Langzeitprädiktion der Bewegungsparameter auf Basis der Referenztrajektorien durchgeführt werden.

Das entwickelte Konzept zur partikelfilterbasierten Zustandsschätzung in einem HMM könnte außerdem im Verkehrsszenario oder für die Beobachtung von Fahrzeuginsassen verwendet werden, denn auch hier ließe sich die Topologie einfacher HMMs aus den möglichen Aktionen ableiten und auch Referenztrajektorien könnten aus zuvor aufgezeichneten Bewegungen extrahiert werden. Auch bei der Extraktion der Referenztrajektorien ergeben sich noch einige Verbesserungsmöglichkeiten. Aktuell werden alle Referenztrajektorien manuell aus der aufgezeichneten Bewegung extrahiert und in die Datenbank eingefügt. Eine automatische Extraktion für bereits gelabelte Zeitbereiche von aufgezeichneten Bewegungen würde den Trainings- und Initialisierungsaufwand der Aktionserkennung um einiges verringern. Weiterhin wäre es sinnvoll eine Online-Erweiterung der Datenbank der Referenztrajektorien zu implementieren, so könnte sich das Aktionserkennungssystem selbstständig adaptieren. An der Absicherung eines solchen Online-Lernen des Systems müsst dann geforscht werden.

In das bestehende System zur Aktionserkennung könnte auch weitere Kontextinformation hinzugefügt werden, d.h. dass durch das intelligente System erkannt wird, dass ein Schraubenschlüssel oder ein Besen gegriffen wurde. Diese Kontextinformation könnte dann die Aktionserkennung weiter verbessern und durch den Partikelfilteransatz auch sehr einfach in das Aktionserkennungssystem integriert werden. Aktuell ist eine solche Erkennung von Gegenständen durch die geringe Auflösung limitiert.

Eine weitere sehr interessante Anwendung des entwickelten Systems wäre die Interaktionen verschiedener Menschen und Roboter. Durch eine prototypische Realisierung eines solches Interaktionsszenarios, wäre es möglich, eine große Studie in der Automobilproduktion durchzuführen, inwieweit ein solches System von den Arbeitern angenommen würde.

Anhang A

Theoretische Grundlagen

Im folgenden Kapitel werden die Grundlagen der Algorithmen und Verfahren erläutert, die für die Realisierung der in dieser Arbeit erarbeiteten Systeme wichtig sind. Die Ausführungen stellen lediglich einen Überblick, jedoch keine vollständige Referenz dar. Für weiterführende Informationen können die angegebenen Quellen verwendet werden.

A.1 Abbildungsvorgang

Die Grundlagen des Abbildungsvorgangs in einem Kamerasystem sind notwendig, da in dieser Arbeit der multiokulare Contracting-Curve-Density-Algorithmus genutzt wird und dieser auf der Projektion eines 3D-Konturmodells im 3D-Weltkoordinatensystem in die 2D-Pixelkoordinatensysteme der verwendeten Kameras basiert.

A.1.1 Lochkameramodell

Eine Kamera bildet eine Szene im dreidimensionalen Raum auf eine zweidimensionale Ebene, die Bildebene der Kamera ab. Um die grundlegende Abbildungsgeometrie zu verstehen, reicht zunächst ein einfaches Lochkameramodell (Abbildung A.1) aus. Die Ebene gegeben durch $Z = -f$ bezeichnet man als Bildebene. Die Kamerakonstante f entspricht dem Abstand des Kamerazentrums zur Bildebene. Die Gerade durch das Kamerazentrum senkrecht zur Bildebene bezeichnet man als optische Achse. Der Schnittpunkt der optischen Achse mit der Bildebene heißt Hauptpunkt. Die Abbildungsgeometrie wird durch Zentralprojektion bestimmt.
In dieser Arbeit wird die Notation von Craig (1989) verwendet. Mit ${}^{C}\mathbf{x}$ wird ein Punkt $\mathbf{x}$ im Koordinatensystem C bezeichnet. Eine ähnliche Notation wird für die Transformations- und Projektionsmatrizen gewählt, die Matrix ${}^{C}_{W}\mathbf{H}$ transformiert einen Punkt im Koordinatensystem W in das Koordinatensystem C: ${}^{C}\mathbf{x} = {}^{C}_{W}\mathbf{H}\,{}^{W}\mathbf{x}$.

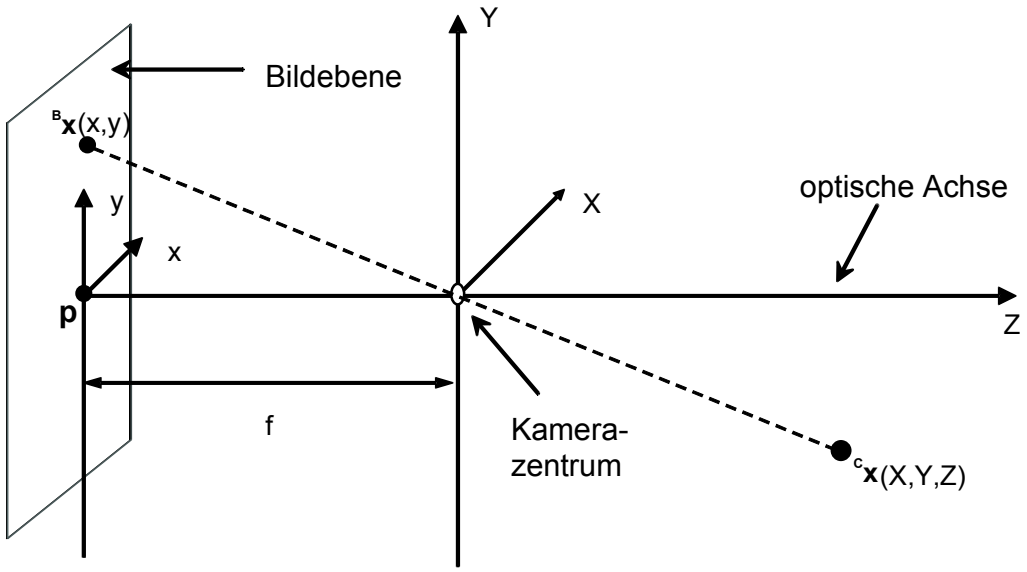

Abbildung A.1: Projektion eines Punktes mit dem Lochkameramodell.

Ein Punkt ${}^{C}\mathbf{x} = (X, Y, Z)$ im Kamerakoordinatensystem C wird über den Schnittpunkt seiner Verbindungsgeraden mit dem Kamerazentrum auf einen Punkt ${}^{B}\mathbf{x} = (x, y)$ in der Bildebene abgebildet. Es ist offensichtlich, dass alle Punkte auf dieser Verbindungsgeraden auf den selben Punkt in der Bildebene abgebildet werden. Die ursprüngliche Tiefeninformation Z geht beim Abbildungsprozess verloren. Für die Bildkoordinaten des Punktes ${}^{B}\mathbf{x} = (x, y)$ in der Bildebene gilt.

$$x = \frac{f}{Z} \cdot X \ , \quad y = \frac{f}{Z} \cdot Y \tag{A.1}$$

Um den (nichtlinearen) Abbildungsprozess in einer linearen Form beschreiben zu können führt man *homogene Koordinaten* ein:

$$\begin{pmatrix} -fX \\ -fY \\ Z \end{pmatrix} = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{A.2}$$

Dividiert man den Lösungsvektor $(-fX, -fY, Z)$ durch seine 3. Komponente Z, so steht in den ersten beiden Komponenten wieder die Bildposition ${}^{B}\mathbf{x} = (x, y)$ entsprechend Gleichung A.1. Gleichung A.2 projiziert den Punkt ${}^{C}\mathbf{x}_{(h)} = (X, Y, Z, 1)$ auf den Punkt ${}^{B}\mathbf{x}_{(h)} = (x_w, y_w, w)$. Durch den Index (h) wird definiert, dass die Punkte in homogenen Koordinaten vorliegen. Die Umrechnung von homogenen Koordinaten in kartesische erfolgt mit ${}^{B}\mathbf{x} = (\frac{x_w}{w}, \frac{y_w}{w})$. Durch die Verlagerung der Bildebene vor das Kamerazentrum ergibt sich folgende Matrixmultiplikation:

$${}^{B}\mathbf{x}_{(h)} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^{C}\mathbf{x}_{(h)} \tag{A.3}$$

und das Bild wird nicht gespiegelt. Die Matrix in Gleichung A.3 wird als *Projektionsmatrix* ${}^{B}_{C}\mathbf{P}$ bezeichnet. Alle weiteren Betrachtungen werden von nun an mit der nach vorne verlagerten Bildebene durchgeführt.

A.1.2 Erweitertes Kameramodell

Zur allgemeinen Beschreibung einer Kamera sind einige Erweiterungen des vereinfachten Lochkameramodells erforderlich. Nach der Abbildung ins Bildkoordinatensystem ${}^{B}\mathbf{x} = (x, y)$, muss in das Pixelkoordinatensystem ${}^{P}\mathbf{u} = (u, v)$ umgerechnet werden:

$$ {}^{P}\mathbf{u}_{(h)} = \begin{pmatrix} s_x & s_\Phi & u_0 \\ 0 & s_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} {}^{B}\mathbf{x}_{(h)}. \tag{A.4} $$

Man bezeichnet die Matrix in A.4 als *Kameramatrix* ${}^{P}_{B}\mathbf{K}$. Die darin auftauchenden fünf Parameter nennt man die *intrinsischen Parameter* der Kamera.
Sie haben im Einzelnen folgende Bedeutung:
Umrechnung auf Pixelkoordinaten: s_x und s_y sind Parameter zur Umrechnung des Kameraparameters f in Pixelkoordinaten.
Verschiebung des Koordinatenursprungs in der Bildebene: (u_0, v_0) gibt die Position des Hauptpunktes bzgl. des Bildkoordinatensystems in Pixeln an.
Nichtrechtwinkliges Koordinatensystem: Der Parameter s_Φ wird als Skew-Parameter (engl. für Schrägheit) bezeichnet und ist ein Maß für die Nichtrechtwinkligkeit der CCD-Elemente zueinander. Bei CCD-Kameras ist dieser Parameter in der Regel gleich Null: $s_\Phi = 0$.
Eine weitere Verallgemeinerung erhält man, wenn man zulässt, dass das Weltkoordinatensystem W und das Kamerakoordinatensystem C gegeneinander verschoben und verdreht sind. Dann lassen sich die Koordinatensysteme durch die Ausführung einer Rotation gegeben durch eine orthonormale Matrix $\mathbf{R}$ gefolgt von einer Translation gegeben durch einen Verschiebungsvektor $\mathbf{t} = [t_1, t_2, t_3]^T$ ineinander überführen. Die Transformation des Punktes ${}^{W}\mathbf{x}$ im Weltkoordinatensystem in den Punkt ${}^{C}\mathbf{x}$ im Kamerakoordinatensystem lässt sich mit:

$$ {}^{C}\mathbf{x}_{(h)} = \left(\begin{array}{ccc|c} & \mathbf{R} & & \mathbf{t} \\ \hline 0 & 0 & 0 & 1 \end{array}\right) {}^{W}\mathbf{x}_{(h)} \tag{A.5} $$

beschreiben. Die Matrix in Gleichung A.5 heißt *Transformationsmatrix* ${}^{C}_{W}\mathbf{H}$, die in ihr enthaltenen Parameter nennt man *extrinsische Parameter* der Kamera.

Man kann den Abbildungsprozess vom Weltkoordinatensystem W in das Pixelkoordinatensystem P_c einer Kamera c als Hintereinanderschaltung der in den Gleichungen A.3, A.4 und A.5 eingeführten Matrizen schreiben:

$$ {}^{P_c}\mathbf{u}_{(h)} = {}^{P_c}_{B_c}\mathbf{K} \quad {}^{B_c}_{C_c}\mathbf{P} \quad {}^{C_c}_{W}\mathbf{H} \quad {}^{W}\mathbf{x}_{(h)}. \tag{A.6} $$

A.1.3 Verzeichnungen in den Bildkoordinaten

Bei einer realen CCD-Kamera erfolgt die Abbildung durch ein System von Linsen. Dies führt zu Verzeichnungen, die man auf unterschiedliche Weise modellieren kann. In dieser Arbeit wird das Modell von Brown (1966, 1971) benutzt, welches von Bouguet (1997) erweitert wurde. Um die verzeichneten Bildkoordinaten ${}^{B}\mathbf{x}_d$ zu berechnen, müssen zuerst die unverzeichneten Bildkoordinaten ${}^{B}\mathbf{x}$, wie in Abschnitt A.1.2 beschrieben, berechnet werden.

$$ {}^{B}\mathbf{x}_{(h)} = {}^{B}_{C}\mathbf{P} \quad {}^{C}_{W}\mathbf{H} \quad {}^{W}\mathbf{x}_{(h)} \tag{A.7} $$

Die verzeichneten Bildkoordinaten ${}^{B}\mathbf{x}_d = (x_d, y_d)$ erhält man aus den unverzeichneten Bildkoordinaten ${}^{B}\mathbf{x}$ gemäß

$$ \mathbf{x}_d = (1 + k_1 r^2 + k_3 r^4 + k_5 r^6)\mathbf{x} + t \ , \tag{A.8} $$

wobei $r^2 = x^2 + y^2$ ist. Der erste Term wird als *radiale Verzeichnung* bezeichnet. Sein Einfluss nimmt radial nach außen hin zu. Der zweite Term t wird als *tangentiale Verzeichnung* bezeichnet:

$$ t \ = \ \begin{pmatrix} 2k_2xy + k_4(r^2 + 2x^2) \\ k_2(r^2 + 2y^2) + 2k_4xy \end{pmatrix}. \tag{A.9} $$

Die tangentiale Verzeichnung lässt sich im Wesentlichen auf eine Dezentrierung des verwendeten Linsensystems zur optischen Achse zurückführen. Für präzise Messungen müssen die Verzeichnungsparameter $(k_1, ..., k_5)$ beim Abbildungsprozess als weitere fünf intrinsische Parameter berücksichtigt werden. Nachdem die verzeichneten Bildkoordinaten ${}^{B}\mathbf{x}_d = (x_d, y_d)$ ermittelt wurden, ergeben sich die Pixelkoordinaten ${}^{P}\mathbf{u} = (u, v)$ durch:

$$ {}^{P}\mathbf{u}_{(h)} = {}^{P}_{B}\mathbf{K} \quad {}^{B}\mathbf{x}_{d(h)}. \tag{A.10} $$

A.1.4 Kamerakalibrierung

Ziel der Kamerakalibrierung ist die Bestimmung der in den vorherigen Abschnitten definierten intrinsischen und extrinsischen Parametern sowie den Verzeichnungsparametern. Dies kann im Prinzip durch Aufnahme von hinreichend vielen Objektpunkten mit bekannten 3D-Weltkoordinaten geschehen. Die Grundlagen und Methoden zur Kamerakalibrierung werden von Slama (1980) ausführlich beschrieben. Im Rahmen dieser Arbeit wurde eine als Open-Source-Code erhältliche Matlab-Bibliothek zur Kamerakalibrierung verwendet (Bouguet, 1997). Krüger u. a. (2004) haben diesen Prozess so erweitert, dass alle drei Kameras gleichzeitig kalibriert werden können.

A.2 Zustandsschätzer

Bei der Betrachtung von Systemen, in denen Größen gemessen werden, ist davon auszugehen, dass dabei Messfehler entstehen. Durch eine geeignete Modellierung des Systems ist es möglich, diese zu verringern. Zur Modellierung eines Systems, welches seinen Zustand mit

fortschreitender Zeit verändert, wird der diskrete Zeitbegriff zugrunde gelegt. Der Zustand eines Systems der Dimension $n \in \mathbb{N}$ zur Zeit t wird modelliert als $x_t \in \mathbb{R}^n$. Ferner wird davon ausgegangen, dass eine Abbildung

$$f : \mathbb{R}^n \to \mathbb{R}^n \tag{A.11}$$

bekannt ist, welche es ermöglicht aus einem Zustand x_t den nächsten Zustand x_{t+1} vorherzusagen. Diese Abbildung ist Teil der Systemmodellierung. Nach der Vorhersage des Modellzustandes für den Zeitpunkt $t+1$ erfolgt die Schätzung des tatsächlichen Systemzustandes basierend auf einer durchgeführten Messung. Für den Übergang von Zeit t zur Zeit $t+1$ sind also folgende Schritte notwendig.

1. Prädiktionsschritt:
Hier erfolgt eine Vorhersage des Systemzustandes zur Zeit $t+1$ allein aus dem Zustand des Modells zur Zeit t, durch Iteration der Abbildung f. Diese Vorhersage wird auch als a-priori Schätzung bezeichnet. Die dadurch bewirkte Veränderung der vermuteten Wahrscheinlichkeit ist deterministisch, weil sie nur vom momentanen Modellzustand und dem Modell abhängt.

2. Innovationsschritt:
Hier werden Messwerte ermittelt die zusammen mit dem vorhergesagten Zustand zur Zeit $t+1$ einem geeigneten Algorithmus übergeben werden. Dieser schätzt daraus den Zustand des tatsächlichen Systems. Der aufgrund der Messwerte korrigierte Modellzustand wird als a-posteriori Schätzung bezeichnet. Im in Kapitel 3 beschriebenen Trackingsystem werden drei Kalman-Filter als Zustandsschätzer eingesetzt, daher wird das Kalman-Filter anschließend beschrieben.

A.2.1 Kalman-Filter

Das Kalman-Filter (Kalman, 1960; Welch u. Bishop, 1995) ist ein 1960 von Rudolf Emil Kalman entwickeltes Verfahren zur statistischen Filterung von Messwerten. Mit einem Kalman-Filter kann der Zustand eines modellierten Systems aufgrund vorliegender Messwerte mit statistischen Methoden abgeschätzt werden. Dazu muss das Modell allerdings diskret und linear sein. Dies bedeutet, dass dem Modell ein diskreter Zeitbegriff zugrunde liegt und der Zustand $\mathbf{x}_k$ zur Zeit k aus dem vorherigen Zustand des Modells zur Zeit $k-1$ durch Anwendung der Iterationsformel

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{w}_{k-1} \tag{A.12}$$

ermittelt werden kann. Die Matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ beschreibt den Übergang des Systemzustandes vom Schritt $k-1$ hin zu Schritt k und wird als Systemmatrix bezeichnet. Die $n \times l$ Matrix $\mathbf{B}$, die so genannte Steuermatrix, modelliert die Wirkung einer zusätzlichen Systemeingabe $\mathbf{u}_k \in \mathbb{R}^l$. Die n-dimensionale Zufallsvariable $\mathbf{w}_{k-1} \in \mathbb{R}^n$ heißt Prozessrauschen und es wird angenommen, dass es sich um ein gaußsches Rauschen $p(\mathbf{w}) \sim N(0, \mathbf{Q})$ mit Mittelwert Null und Kovarianzmatrix $\mathbf{Q}$ handelt. Mit den weiter vorne eingeführten Begriffen bedeutet dies, dass die Abbildung f, die die Zustandsänderung vermittelt, linear ist. Des Weiteren wird davon ausgegangen, dass der Zustand des modellierten Systems nur indirekt als

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \tag{A.13}$$

gemessen werden kann. Die Messmatrix $\mathbf{H} \in \mathbb{R}^{m \times n}$ modelliert den linearen Messvorgang und die Zufallsvariable $\mathbf{v}_k$ (Messrauschen) beschreibt wiederum ein gaußsches Rauschen $p(\mathbf{v}) \sim N(0, \mathbf{R})$ mit Mittelwert Null und Kovarianzmatrix $\mathbf{R}$.
Sei $\hat{\mathbf{x}}_k^-$ eine a-priori Schätzung des Zustands $\mathbf{x}_k$, d.h. eine Schätzung des Systemzustandes $\mathbf{x}_k$ vor der Messung zum Zeitschritt k. Es liegen also nur Messungen $\mathbf{z}_1 \dots \mathbf{z}_{k-1}$ vor. Der mittelwertfreie a-priori Schätzfehler $\mathbf{e}_k^-$ ist dann definiert als

$$\mathbf{e}_k^- = \mathbf{x}_k - \hat{\mathbf{x}}_k^-. \tag{A.14}$$

Ebenso lässt sich die a-posteriori Schätzung $\hat{\mathbf{x}}_k$, d.h. eine Schätzung des Systemzustandes unter Einbeziehung der Messung $\mathbf{y}_k$ definieren. Der mittelwertfreie a-posteriori Schätzfehler $\mathbf{e}_k$ ist daher

$$\mathbf{e}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k. \tag{A.15}$$

Die zugehörigen Schätzfehlerkovarianzen $\mathbf{P}_k^-$ und $\mathbf{P}_k$ sind

$$\mathbf{P}_k^- = E[\mathbf{e}_k^- \mathbf{e}_k^{-T}] \text{ und} \tag{A.16}$$
$$\mathbf{P}_k = E[\mathbf{e}_k \mathbf{e}_k^T]. \tag{A.17}$$

Ziel ist es nun, die a-priori und die a-posteriori Schätzung bzw. deren Kovarianzen in einen mathematischen Zusammenhang zu bringen und damit den Systemzustand $\mathbf{x}_k$ zu jedem Zeitpunkt so genau wie möglich, d.h. optimal, zu schätzen. Dazu wählt man die folgende Vorgehensweise:
Offensichtlich lässt sich der a-priori Schätzwert $\hat{\mathbf{x}}_k^-$ aus dem a-posteriori Schätzwert $\hat{\mathbf{x}}_k$ des vorangegangenen Zeitschrittes nach

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1}^- + \mathbf{B}\mathbf{u}_{k-1} \tag{A.18}$$

berechnen. Die a-posteriori Schätzung wird über das zugrundeliegende Zustandsmodell transformiert und die a-priori Schätzfehlerkovarianz $\mathbf{P}_k^-$ ergibt sich aus

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}^-\mathbf{A}^T + \mathbf{Q}. \tag{A.19}$$

Die a-posteriori Schätzung $\hat{\mathbf{x}}_k$ lässt sich nun als Linearkombination aus der a-priori Schätzung $\hat{\mathbf{x}}_k^-$ und einer über den Kalman-Faktor $\mathbf{K}$ gewichteten Differenz zwischen aktuellem Messwert $\mathbf{z}_k$ und der Prädiktion des Messwertes berechnen

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k^-). \tag{A.20}$$

Die $n \times m$ Matrix $\mathbf{K}$ wird als Kalman-Faktor bezeichnet und ergibt sich folgendermaßen

$$\mathbf{K} = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1}. \tag{A.21}$$

Die a-posteriori Schätzfehlerkovarianz $\mathbf{P}_k$ berechnet sich aus

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}\mathbf{H}\mathbf{P}_k^-. \tag{A.22}$$

Mit den Gleichungen A.18 bis A.22 lässt sich zu jedem Zeitpunkt ein Schätzwert für den wahren Zustand eines linearen, zeitdiskreten, stochastischen Systems inklusive einer Fehlerabschätzung in Form einer Kovarianzmatrix angeben.

A.3 Mean-Shift-Optimierung

Der Mean-Shift-Algorithmus wird in der Literatur in verschiedenen Varianten beschrieben. Als ursprüngliche Veröffentlichung gilt (Fukunaga, 1975). In dieser Arbeit werden Verallgemeinerungen gemäß (Cheng, 1995) und (Comaniciu u. Meer, 2002) beschrieben. Bei der Mean-Shift-Optimierung muss durch den Einsatz von nicht-parametrisierten Methoden keine Verteilungsannahme getroffen werden, sondern es können beliebige Verteilungen betrachtet werden. Anstatt eine Verteilungsdichtefunktion abzuschätzen und den Gradienten berechnen zu müssen, bietet Mean-Shift eine effiziente Methode, um die Maxima eines Merkmalsraums zu lokalisieren. Mean-Shift wird beim Tracking eingesetzt, um den ähnlichsten Kandidaten zu einem definierten Zielobjekt zu finden (Bradski, 1998; Comaniciu u. a., 2003). Außerdem besteht die Möglichkeit Mean-Shift zur effizienten Filterung und Segmentierung von Bildern zu verwenden (Comaniciu u. Meer, 2002). Im Folgenden wird der allgemeine Mean-Shift-Algorithmus beschrieben. Außerdem wird das von Bradski (1998) vorgestellte CAMShift-Tracking beschrieben, da die 3D-Erweiterung im Abschnitt 5 darauf aufbaut.

A.3.1 Der Mean-Shift-Algorithmus

Der Mean-Shift-Algorithmus ist ein statistisches, nicht-parametrisches Verfahren zur Bestimmung der Maxima einer Verteilung. Nicht-parametrische Verfahren sind nicht auf bestimmte Wahrscheinlichkeitsverteilung beschränkt, sondern können mit beliebigen Verteilungen arbeiten. Im Gegensatz zu allgemeinen gradientenbasierten Optimierungsverfahren, wo die Größe des Verschiebungsvektors, die Schrittgröße, direkt gewählt werden muss, ist es beim Mean-Shift-Algorithmus nicht notwendig die Schrittgröße zu wählen. Ein Problem bei Gradientenverfahren ist die Wahl dieser Schrittgröße. Die Laufzeit des Algorithmus hängt direkt von der gewählten Schrittgröße ab. Ist die Schrittgröße zu groß gewählt, divergiert der Algorithmus, bei einer zu klein gewählten Schrittgröße kann die Optimierung sehr lange dauern. Konvergenz ist nur für infinitesimal kleine Schritte garantiert. Beim Mean-Shift-Algorithmus sind keine zusätzlichen Methoden nötig, um die Schrittgröße zu wählen. Die Schrittgröße ergibt sich direkt aus der Größe des Mean-Shift-Vektors und ändert sich somit adaptiv bezüglich der lokalen Steigung der Verteilungsdichte. Durch die Eigenschaft der adaptiven Schrittgrößenänderung konvergiert das Mean-Shift-Verfahren. Ein weiterer Vorteil gegenüber gradientenbasierten Optimierungsverfahren ist, dass der Gradient nicht mehr ausgerechnet werden muss. Es ist ausreichend die effizientere Mean-Shift-Berechnung durchzuführen. Für eine Herleitung des Algorithmus und eine detaillierte Beschreibung sei auf die Arbeiten von Cheng (1995) und Comaniciu u. Meer (2002) verwiesen.

Für N gegebene Merkmalsvektoren $\mathbf{x}_n$ mit $\mathbf{x}_n \in \mathbb{R}^D, n$ und einem Kernel G am Punkt $\mathbf{x} = (x_1, ..., x_d, ..., x_D) \in \mathbb{R}^D$ im Merkmalsraum mit Fensterradius h wird der D-dimensionale Mean-Shift-Vektor $\mathbf{m}_{(h,G)}$ folgendermaßen berechnet:

$$\mathbf{m}_{(h,G)}(\mathbf{x}) = \frac{\sum_{n=1}^{N} \mathbf{x}_n g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_n}{h}\right\|^2\right)}{\sum_{n=1}^{N} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_n}{h}\right\|^2\right)} - \mathbf{x}. \tag{A.23}$$

Dabei beschreibt die Funktion g das Profil des Kernels G. Die N Beobachtungen $\mathbf{x}_n, n \in [1, N]$

werden dabei mittels des Kernels G gewichtet, summiert und mit der Gesamtsumme normalisiert. Die Verschiebung, das *Shift*, entsteht aus der Differenz des Mittelwerts, dem *Mean*, und $\mathbf{x}$, dem Zentrum des Kernels G. In der Arbeit von Cheng (1995) werden wichtige Kernelprofile vorgestellt und detailliert beschrieben.

Der Mean-Shift-Vektor (Gleichung (A.23)) bewegt sich in die Richtung der maximalen Steigung der Dichte und definiert einen Pfad zu einem Maximum der Verteilung. Eine Beispielverteilung und verschiedene Iterationsschritte der Mean-Shift-Optimierung sind in Abbildung A.2 dargestellt. Der folgende Ablauf beschreibt die verschiedenen Schritte der Mean-Shift-Optimierung:

1. Schritt: Im ersten Schritt des Algorithmus wird der Fensterradius gewählt, wobei h_n der Fensterradius oder die Bandbreite des Kernels in der Beobachtung $\mathbf{x}_n$ ist. Wenn die Fensterradien h_n als h gewählt werden, spricht man von *fixierter Bandbreite*. Ein fixiertes h bedeutet, dass in jeder Beobachtung mit identisch skalierten Kernels gearbeitet wird.

2. Schritt: Im zweiten Schritt wird die Startposition $\mathbf{y}_1 \in \mathbb{R}^D$ der Mean-Shift-Optimierung gewählt.

3. Schritt: Im Schritt 3 wird das Kernelfenster mit dem Mean-Shift-Vektor $\mathbf{m}_{(h,G)}(\mathbf{y}_j)$ auf eine neue Position verschoben. Die neue Position des Suchfensters $\mathbf{y}_{j+1} \in \mathbb{R}^D$ im Merkmalsraum wird durch die Gleichung

$$\mathbf{y}_{j+1} = \frac{\sum_{n=1}^{N} \mathbf{x}_n g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_n}{h}\right\|^2\right)}{\sum_{n=1}^{N} g\left(\left\|\frac{\mathbf{y}_j - \mathbf{x}_n}{h}\right\|^2\right)} \quad j = 1, 2, ... \tag{A.24}$$

berechnet, wobei $\mathbf{y}_j \in \mathbb{R}^D$ die alte Position des Kernelfensters ist. Dieser dritte Schritt wird bis zur Konvergenz wiederholt.

Zur Überprüfung, ob das Ergebnis der Mean-Shift-Optimierung auch tatsächlich ein Maxima ist, kann ein gefundener Konvergenzpunkt $\mathbf{y}_{konv}$ durch einen kleinen zufälligen Vektor verschoben werden. So kann man überprüfen, ob das Verfahren erneut zum selben Punkt $\mathbf{y}_{konv}$ konvergiert (Comaniciu u. Meer, 2002).

A.3.2 CAMShift-Tracking

Der folgende Abschnitt beschäftigt sich mit dem CAMShift-Tracking nach Bradski (1998). CAMShift steht für "*Continuously Adaptive Mean Shift*", der Algorithmus basiert auf dem zuvor beschriebenen Mean-Shift-Algorithmus. Da sich der Bildinhalt in einer Bildsequenz ständig ändert, bleiben auch die Größe und die Position der Verteilung nicht gleich, denn das Zielobjekt bewegt sich in der Bildsequenz. An diesem Punkt setzt der CAMShift-Algorithmus an. Nach jedem Bild passt der Algorithmus die Suchfenstergröße s an die veränderten Verhältnisse an. Für das Tracking von Gesichtern wird hier nur die Hue-Komponente des HSV-Farbraums (Gonzalez u. Woods, 2002) verwendet. Als Abschätzung für die Farbverteilung wird ein eindimensionales Histogramm verwendet, das sogenannte Zielmodell $\widehat{\mathbf{q}}$. Aus dem initial erstellten Histogramm (Zielhistogramm) und einem Eingabebild wird anschließend eine Wahrscheinlichkeitsverteilung für die Ähnlichkeit zum Zielobjekt erstellt (Abbildung A.3). Probleme können hier bei sehr geringer Helligkeit (V-Komponente) auftreten, da hier die Hue-Werte sehr stark rauschen. Das

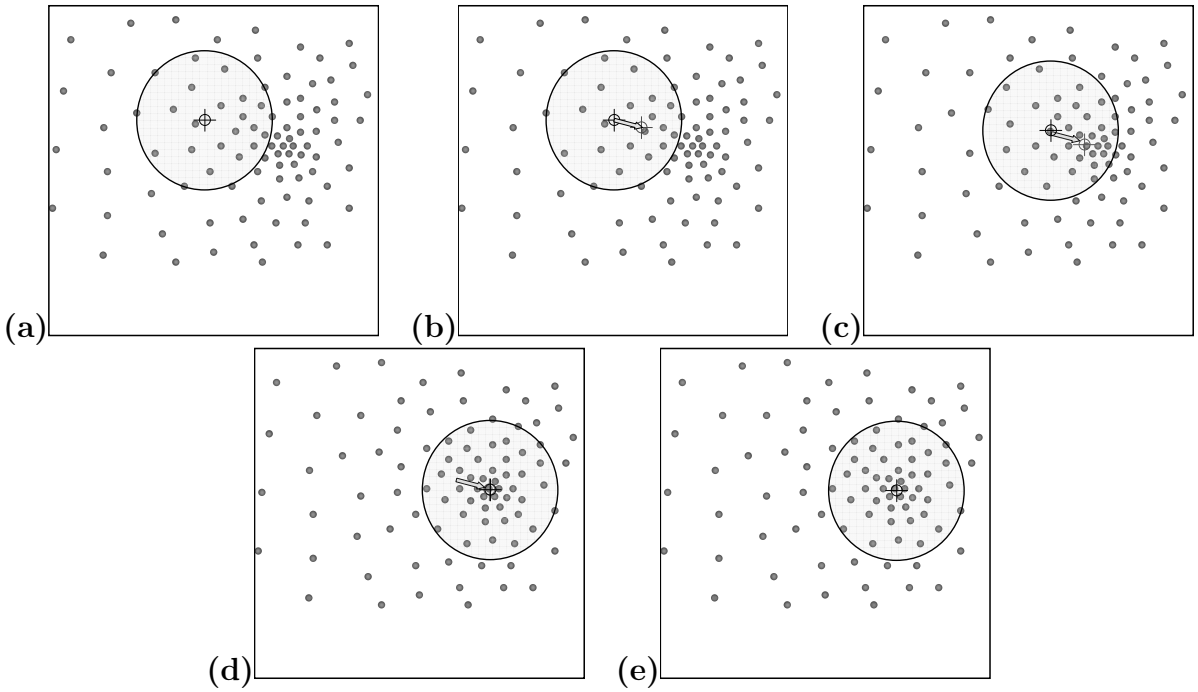

Abbildung A.2: Es ist eine Verteilung von Punkten in einem zweidimensionalen Merkmalsraum dargestellt. Der Mean-Shift-Algorithmus soll dazu verwendet werden, das Maximum der Verteilung zu finden. (a) Der große schwarze Kreis stellt das Suchfenster dar und der kleine schwarze Kreis die aktuelle Position des Suchfensters. (b) Der Pfeil visualisiert den Mean-Shift-Vektor und der rote Kreis die neu berechnete Position des Maximums innerhalb des Suchfensters. (c) Das Suchfenster wurde an die neue Position aus (b) verschoben, und die Berechnung des Maximums wird erneut ausgeführt. (d) Wieder wird das Suchfenster verschoben und das Maximum im Suchfenster ermittelt. (e) Das Suchfenster wird neu positioniert. Bei erneuter Berechnung des Maximums ist die Ergebnisposition gleich der Ausgangsposition. Dies zeigt, die Konvergenz des Mean-Shift-Algorithmus.

Zielmodell $\widehat{\mathbf{q}}$ wird im Bereich des Zielobjekts (Rechteck) berechnet und so normalisiert, dass die Summe der Häufigkeiten in allen Histogramm-Bins Eins ergibt. Das Zielhistogramm enthält somit in jedem Histogramm-Bin eine relative Häufigkeit für das Auftreten des jeweiligen Pixelwerts im Bereich des Zielobjekts. Diese realtive Häufigkeit, wird als Wahrscheinlichkeit interpretiert und dazu verwendet, um die einzelnen Bilder einer Videosequenz in eine Form (Wahrscheinlichkeitsverteilung) zu bringen, sodass man darauf den Mean-Shift-Algorithmus anwenden kann. Dazu weist man jedem Pixel im Berechnungsbereich b, abhängig von seiner H-Komponente, den entsprechenden Wert aus dem Zielhistogramm zu. Das Zielhistogramm wird dabei als *Lookup Table* verwendet. Bei einem 8 Bit H-Kanal sind 256 unterschiedliche Wahrscheinlichkeitswerte möglich. Das bedeutet, dass in dem entsprechenden Histogramm den Farbtönen von 0 bis 255, Wahrscheinlichkeiten im Intervall $[0, 1]$ zugewiesen werden.

Berechnung des Verteilungsschwerpunkts: Durch die Zuweisung der relativen Häufigkeiten, welche als Wahrscheinlichkeiten interpretiert werden, zu allen im Berechnungsbereich befindlichen Pixeln $\mathbf{I}(u, v)$ erhält man eine diskrete 2D-Wahrscheinlichkeitsverteilung $\mathbf{P}$. Mit

Abbildung A.3: Links: Eingabebild und Repräsentation der Form des Zielobjekts als Rechteck. Rechts: Wahrscheinlichkeitsverteilung für die Ähnlichkeit zum Referenzobjekt. Das Bild ist skaliert auf 8 Bit, hellere Pixelwerte bedeuten eine hohe Wahrscheinlichkeit.

dem Mean-Shift-Algorithmus wird nun die Region mit der höchsten Wahrscheinlichkeit gesucht. Innerhalb des Suchfensters kann man das Maximum mit Hilfe statistischer Momente nullter und erster Ordnung berechnen. Die Momente dienen der Beschreibung von Regionen, mit deren Hilfe können beispielsweise Flächen, Schwerpunkte usw. berechnet werden. Zentrale Momente liefern hier größen- und verschiebungsunabhängige Merkmale, die auf den Schwerpunkt $s = (u, v)^T$ einer Region bezogen sind (Bradski, 1998). Momente der Ordnung p, q auf einer diskreten 2D-Wahrscheinlichkeitsverteilung $\mathbf{P}$ sind definiert als

$$m_{pq} = \sum_{(u,v)\in\mathbf{P}} \mathbf{P}(u, v) \cdot u^p \cdot v^q. \tag{A.25}$$

Dieser Ausdruck beschreibt ganz allgemein das Moment der Ordnung p, q für die diskrete 2D-Wahrscheinlichkeitsverteilung $\mathbf{P}$, wobei $\mathbf{P}(u, v)$ dem Wahrscheinlichkeitswert an der Position (u, v) entspricht. Das Moment nullter Ordnung beschreibt die Fläche einer Verteilung wird folgendermaßen berechnet:

$$m_{00} = \sum_{(u,v)\in\mathbf{P}} \mathbf{P}(u, v). \tag{A.26}$$

Um die Position $\mathbf{c}$ des Maximums innerhalb des Suchfensters zu ermitteln, benötigt man neben m_{00} noch die Momente erster Ordnung

$$m_{10} = \sum_{(u,v)\in\mathbf{P}} \mathbf{P}(u, v) \cdot u \tag{A.27}$$

und

$$m_{01} = \sum_{(u,v)\in\mathbf{P}} \mathbf{P}(u, v) \cdot v. \tag{A.28}$$

Somit kann der Schwerpunkt der Region $\mathbf{c} = (c_u, c_v)$ wie folgt berechnet werden:

$$c_u = \frac{m_{10}}{m_{00}}, c_v = \frac{m_{01}}{m_{00}}. \tag{A.29}$$

Der CAMShift-Algorithmus kommt erst zum Einsatz, nachdem die beste Position $\mathbf{c}$ für ein Bild gefunden wurde. Dabei wird die Größe des Suchfensters s für das nächste Bild aus m_{00}

und $\mathbf{P}_{max}$ ermittelt (Bradski, 1998):

$$w_s = 2 \cdot \sqrt{\frac{m_{00}}{\mathbf{P}_{max}}}, h_s = 1.2 \cdot w_s. \tag{A.30}$$

$\mathbf{P}_{max}$ beschreibt den höchsten Wahrscheinlichkeitswert, w_s steht für die Breite und h_s für die Höhe des Suchfensters. Die Multiplikation der Höhe des Suchfensters mit dem Faktor 1.2 wird von Bradski (1998) damit begründet, dass es sich bei den Zielobjekten um Köpfe handelt und diese annähernd die Form einer Ellipse haben. Neben der Fenstergröße (w_s, h_s) berechnet Bradski (1998) auch die Breite w, die Höhe h, und die Rotation ϕ des Zielobjekts. Dafür werden ebenfalls statistische Momente eingesetzt. Die Breite

$$w = 2 \cdot \left(\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2} \right)^{\frac{1}{2}} \tag{A.31}$$

und die Höhe

$$h = 2 \cdot \left(\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2} \right)^{\frac{1}{2}} \tag{A.32}$$

berechnen sich aus den Momenten nullter, erster und zweiter Ordnung:

$$a = \frac{m_{20}}{m_{00}} - c_x^2, b = 2 \cdot \left(\frac{m_{11}}{m_{00}} - c_x \cdot c_y \right), c = \frac{m_{02}}{m_{00}} - c_y^2. \tag{A.33}$$

Die Orientierung ϕ des Zielobjekts wird anhand der 2D-Wahrscheinlichkeitsverteilung bestimmt und ergibt sich mit:

$$\phi = \frac{1}{2} \arctan \left(\frac{2 \cdot \left(\frac{m_{11}}{m_{00}} - c_u c_v \right)}{\left(\frac{m_{20}}{m_{00}} - c_u^2 \right) - \left(\frac{m_{02}}{m_{00}} - c_v^2 \right)} \right). \tag{A.34}$$

Die aktuelle Größe und Orientierung des Zielobjektes werden verwendet, um die Größe und Lage des Suchfensters im nächsten Frame festzulegen. Der Algorithmus wird für kontinuierliches Tracking wiederholt. Der beschriebene Algorithmus ist eine Verallgemeinerung des Mean-Shift-Algorithmus, denn die Berechnung des Schwerpunkts entspricht dem Mean-Shift-Algorithmus. Im folgenden wird kurz auf die Ablaufschritte des CAMShift-Algorithmus eingegangen.

1. Setzen des Rechenbereichs der Wahrscheinlichkeitsverteilung auf den gesamten Bildbereich
2. Wählen der Startposition des 2D-Mean-Shift-Suchfenste
3. Berechnung der 2D-Wahrscheinlichkeitsverteilung $\mathbf{P}$ in einem Bereich um die Position des Suchfensters, welcher etwas größer ist als die Mean-Shift-Fenstergröße.
4. Mit Hilfe des Mean-Shift-Algorithmus wird das Maximum $\mathbf{c}$ der 2D-Wahrscheinlichkeitsverteilung $\mathbf{P}$ berechnet, dazu wird Gleichung (A.29) iterativ angewandt. Außerdem wird die Fenstergröße (Gleichung (A.30)), die Höhe (Gleichung (A.32)), die Breite (Gleichung (A.31)) und Orientierung (Gleichung (A.34)) berechnet.
5. Für das nächste Bild zentriert man das Suchfenster an der Position, die im vierten Schritt berechnet wurde, verwendet die neue Fenstergröße und beginnt den Ablauf mit dem dritten Schritt von neuem.

Anhang B

Schriftenverzeichnis

B.1 Eigene Veröffentlichungen

1. (Hahn u. a., 2007)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Gross, Horst-Michael: Tracking of Human Body Parts using the Multiocular Contracting Curve Density Algorithm. In: *3DIM '07: Proc. of the Sixth Int. Conf. on 3-D Digital Imaging and Modeling*, 2007, S. 257–264
2. (Hahn u. a., 2008a)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: 3D Action Recognition and Long-Term Prediction of Human Motion. In: *Proc. Int. Conf. on Computer Vision Systems, Santorini, Greece.*, 2008, S. 23–32
3. (Hahn u. a., 2008b)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: Spatio-Temporal 3D Pose Estimation and Tracking of Human Body Parts Using the Shape Flow Algorithm. In: *Proc. Int. Conf. on Pattern Recognition, Tampa, USA*, 2008
4. (Hahn u. a., 2009a)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: A Bayesian Approach to 3D Head-Shoulder Pose Tracking. In: *Proc. of Oldenburger 3D-Tage, Oldenburg, Germany.*, 2009, S. 262–269
5. (Hahn u. a., 2009b)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Kummert, Franz: 3D Action Recognition in an Industrial Environment. In: *Proc. of 3rd Int. Workshop on Human-Centered Robotic Systems (HCRS'09), Bielefeld, Germany.*, 2009
6. (Hahn u. a., 2010b)
   Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Sagerer, Gerhard ; Kummert, Franz: Spatio-temporal 3D Pose Estimation and Tracking of Human Body Parts in an Industrial Environment. In: *Proc. of Oldenburger 3D-Tage, Oldenburg, Germany*, 2010
7. (Hahn u. a., 2010d)
   Hahn, Markus ; Wöhler, Christian ; Einhaus, Julian ; Hermes, Christoph ; Kummert, Franz: Tracking and Motion Prediction of Vehicles in Complex Urban Traffic Scenes. In: *Proc. 4. Tagung Sicherheit durch Fahrerassistenz, München, Germany*, 2010

8. (Hermes u. a., 2010)
HERMES, Christoph ;EINHAUS, Julian ; HAHN, Markus ; WÖHLER, Christian ; KUMMERT, Franz: Vehicle Tracking and Motion Prediction in Complex Urban Scenarios. In: *Proc. IEEE Intelligent Vehicles Symposium, San Diego, California*, 2010
9. (Hahn u. a., 2010c)
HAHN, Markus ; QURONFULEH, Fuad ; WÖHLER, Christian ; KUMMERT, Franz: 3D Mean-Shift Tracking and Recognition of Working Actions. In: *Proc. Int. Workshop on Human Behaviour Understanding, held in conjunction with ICPR 2010, Istanbul, Turkey*, 2010
10. (Hahn u. a., 2010a)
HAHN, Markus ; BARROIS, Björn ; KRÜGER, Lars ; WÖHLER, Christian ; SAGERER, Gerhard ; KUMMERT, Franz: Robust and Accurate 3D Pose Estimation and Motion Analysis in an Industrial Environment. In: *Submitted to 3D Research* (2010)

B.2 Patente

1. „Verfahren und Vorrichtung zur dreidimensionalen Konturbestimmung“
Nummer: DE_102007043305
Erfinder: Markus Hahn, Lars Krüger, Christian Wöhler
2. „Verfahren zur Vorhersage einer Aktion eines bewegten Objekts“
Nummer: DE_102007052763
Erfinder: Markus Hahn, Lars Krüger, Christian Wöhler
3. „Situationsabhängige Bestimmung der Verlässlichkeit von Sensordaten bei der Sensorfusion“
Nummer: DE_102009033853
Erfinder: Björn Barrois, Niklas Beuter, Markus Hahn, Christoph Hermes, Ulrich Kreßel, Lars Krüger, Franz Kumert, Rainer Ott, Gehard Sagerer, Christian Wöhler
4. „Modellfreie 3D-Objektdetektion und -verfolgung“
Nummer: DE_102007052763
Erfinder: Markus Hahn, Fuad Quronfuleh, Lars Krüger, Christian Wöhler
5. „Detektion und Verfolgung von Objekten in Bildsequenzen“
Nummer: in Anmeldung
Erfinder: Björn Barrois, Niklas Beuter, Markus Hahn, Christoph Hermes, Ulrich Kreßel, Lars Krüger, Franz Kummert, Rainer Ott, Gehard Sagerer, Christian Wöhler

Literaturverzeichnis

[Agarwal u. Triggs 2004] Agarwal, Ankur ; Triggs, Bill: Tracking Articulated Motion Using a Mixture of Autoregressive Models. In: *Proc. of the 8th European Conf. on Computer Vision*, 2004, S. 54–65

[Aggarwal u. Cai 1999] Aggarwal, J. K. ; Cai, Q.: Human Motion Analysis: A Review. In: *Computer Vision and Image Understanding: CVIU* 73 (1999), Nr. 3, S. 428–440

[Akima 1970] Akima, Hiroshi: A new method of interpolation and smooth curve fitting based on local procedures. In: *Journal of the Association for Computing Machinery* 17 (1970), Nr. 4, S. 589–602

[Amini u. a. 1988] Amini, A. ; Tehrani, S. ; Weymouth, T.: Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In: *Proc., Second Int. Conf. on Computer Vision, Tampa, Fl*, 1988, S. 95–99

[Athitsos u. Sclaroff 2003] Athitsos, Vassilis ; Sclaroff, Stan: Estimating 3D Hand Pose from a Cluttered Image. In: *CVPR '03: Proc. of the 2003 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Volume 2*, 2003, S. 432–442

[Barrois 2010] Barrois, Björn: *Analyse der Position, Orientierung und Bewegung von rigiden und artikulierten Objekten aus Stereobildsequenzen*, Universität Bielefeld, Technische Fakultät, Diss., 2010

[Barrois u. a. 2009] Barrois, Björn ; Hristova, Stela ; Wöhler, Christian ; Kummert, Franz ; Hermes, Christoph: 3D Pose Estimation of Vehicles Using a Stereo Camera. In: *Intelligent Vehicles Symposium (IV'09)*, 2009

[Barrois u. Wöhler 2008] Barrois, Björn ; Wöhler, Christian: Spatio-temporal 3D Pose Estimation of Objects in Stereo Images. In: *6th Int. Conf. on Computer Vision Systems (ICVS)*, 2008

[Bashir u. a. 2007] Bashir, F. I. ; Khokhar, A. A. ; Schonfeld, D.: Real-Time Motion Trajectory-Based Indexing and Retrieval of Video Sequences. In: *IEEE Transactions on Multimedia* 9 (2007), Nr. 1, S. 58–65

[Besl u. McKay 1992] Besl, Paul J. ; McKay, Neil D.: A Method for Registration of 3-D Shapes. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (1992), Nr. 2, S. 239–256

[Black u. Jepson 1998] Black, Michael J. ; Jepson, Allan D.: A Probabilistic Framework for Matching Temporal Trajectories: CONDENSATION-Based Recognition of Gestures and Expressions. In: *ECCV '98: Proc. of the 5th European Conf. on Computer Vision-Volume I*, 1998, S. 909–924

[Blake u. Isard 1998] Blake, Andrew ; Isard, Michael: *Active Contours*. Springer-Verlag, 1998

[Bobick 1997] Bobick, Aaron F.: Movement, Activity, and Action: The Role of Knowledge in the Perception of Motion. In: *Royal Society Workshop on Knowledge-based Vision in Man and Machine* 352 (1997), S. 1257–1265

[Bobick u. Davis 2001] Bobick, Aaron F. ; Davis, James W.: The Recognition of Human Movement Using Temporal Templates. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (2001), Nr. 3, S. 257–267

[Bock 1974] Bock, Hans H.: *Automatische Klassifikation.* Vandenhoeck & Ruprecht, 1974

[Bouguet 1997] Bouguet, Jean-Yves: *Camera Calibration Toolbox for MATLAB.* 1997

[Bradski 1998] Bradski, Gary R.: Real Time Face and Object Tracking as a Component of a Perceptual User Interface. In: *WACV '98: Proc. of the 4th IEEE Workshop on Applications of Computer Vision (WACV'98)*, 1998, S. 214–219

[Brown 1971] Brown, D.C.: Close-Range Camera Calibration. In: *Photometric Engineering* 37 (1971), Nr. 8, S. 855–866

[Brown 1966] Brown, Duane C.: Decentring Distortions of Lenses. In: *Photogrammetric Engineering* 32 (1966), Nr. 4, S. 444–462

[Campbell u. a. 1996] Campbell, L. W. ; Becker, D. A. ; Azarbayejani, A. ; Bobick, A. F. ; Pentland, A.: Invariant features for 3-D gesture recognition. In: *FG '96: Proc. of the 2nd Int. Conf. on Automatic Face and Gesture Recognition (FG '96)*, 1996

[Cedras u. Shah 1995] Cedras, Claudette ; Shah, Mubarak: Motion-Based Recognition: A Survey. In: *Image and Vision Computing* 13 (1995), S. 129–155

[Cheng 1995] Cheng, Yizong: Mean Shift, Mode Seeking, and Clustering. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 17 (1995), Nr. 8, S. 790–799

[Cohen 1991] Cohen, L. D.: On active contour models and balloons. In: *Computer Vision, Graphics and Image Processing. Image Understanding* 53 (1991), Nr. 2, S. 211–218

[Comaniciu u. Meer 2002] Comaniciu, Dorin ; Meer, Peter: Mean Shift: A Robust Approach Toward Feature Space Analysis. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 24 (2002), Nr. 5, S. 603–619

[Comaniciu u. a. 2000] Comaniciu, Dorin ; Ramesh, Visvanathan ; Meer, Peter: Real-time tracking of non-rigid objects using mean shift, 2000, S. 142–149

[Comaniciu u. a. 2003] Comaniciu, Dorin ; Ramesh, Visvanathan ; Meer, Peter: Kernel-Based Object Tracking. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), S. 564–577

[Cootes u. a. 1995] Cootes, T. F. ; Taylor, C. J. ; Cooper, D. H. ; Graham, J.: Active shape models—their training and application. In: *Comput. Vis. Image Underst.* 61 (1995), Nr. 1, S. 38–59

[Cootes u. a. 1994] Cootes, Timothy F. ; Hill, A. ; Taylor, Christopher J. ; Haslam, J.: Use of Active Shape Models for Locating Structures in Medical Images. In: *Image Vision Computing* 12 (1994), Nr. 6, S. 355–365

[Craig 1989] Craig, John J.: *Introduction to Robotics.* 2. Addison-Wesley Publishing Company, 1989

[Croitoru u. a. 2005] Croitoru, Arie ; Agouris, Peggy ; Stefanidis, Anthony: 3D trajectory

matching by pose normalization. In: *GIS '05: Proc. of the 13th annual ACM international workshop on Geographic information systems*, 2005, S. 153–162

[Cutler u. Turk 1998] Cutler, Ross ; Turk, Matthew: View-Based Interpretation of Real-Time Optical Flow for Gesture Recognition. In: *FG '98: Proceedings of the 3rd. International Conference on Face and Gesture Recognition*, 1998, S. 416

[Dalal u. Triggs 2005] Dalal, Navneet ; Triggs, Bill: Histograms of Oriented Gradients for Human Detection. In: *CVPR '05: Proc. of the 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Volume 2*, 2005, S. 886–893

[d'Angelo u. a. 2004] d'Angelo, P. ; Wöhler, C. ; Krüger., L.: Model Based Multi-View Active Contours for Quality Inspection. In: *Proceedings of International Conference on Computer Vision and Graphics*. Warszaw, Poland, 2004

[Darrell u. a. 2001] Darrell, T. ; Demirdjian, D. ; Checka, N. ; Felzenswalb, P.: Plan-view trajectory estimation with dense stereo background models. In: *ICCV '01, Proc. of eighth IEEE Int.l Conf. on Computer Vision*, 2001

[Delamarre u. Faugeras 1998] Delamarre, Q. ; Faugeras, O.: *Finding pose of hand in video images: a stereo-based approach*. 1998

[Demirdjian 2003] Demirdjian, D.: Enforcing Constraints for Human Body Tracking. In: *Computer Vision and Pattern Recognition Workshop* 9 (2003), S. 102

[Demirdjian u. Darrell 2002] Demirdjian, David ; Darrell, Trevor: 3-D Articulated Pose Tracking for Untethered Diectic Reference. In: *Multimodal Interfaces, IEEE Int. Conf. on* 0 (2002), S. 267

[Deutscher u. Reid 2005] Deutscher, Jonathan ; Reid, Ian: Articulated Body Motion Capture by Stochastic Search. In: *Int. Journal on Computer Vision* 61 (2005), Nr. 2, S. 185–205

[Dollar u. a. 2005] Dollar, Piotr ; Rabaud, Vincent ; Cottrell, Garrison ; Belongie, Serge: Behavior recognition via sparse spatio-temporal features. In: *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, 2005, S. 65–72

[Doucet u. a. 2001] Doucet, A. (. ; Freitas, N. de (. ; Gordon, N. (.: *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001

[Duong u. a. 2005] Duong, T. V. ; Bui, H. H. ; Phung, D. Q. ; Venkatesh, S.: Activity recognition and abnormality detection with the switching hidden semi-Markov model. In: *CVPR '05: Proc. of the 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Volume 1*, 2005, S. 838–845

[Duric u. a. 2002] Duric, Zoran ; Li, Fayin ; Sun, Yan ; Wechsler, Harry: Using Normal Flow for Detection and Tracking of Limbs in Color Images. In: *ICPR '02: Proc. of the 16 th Int. Conf. on Pattern Recognition (ICPR'02) Volume 4*, 2002, S. 268–271

[Ebert 2003] Ebert, Dirk: *Bildbasierte Erzeugung kollisionsfreier Transferbewegungen für Industrieroboter*, Schriftenreihe des Fachbereichs Informatik der Technischen Universität Kaiserslautern, ISSN 1610-2673, Band 12, ISBN 3-936890-23-4, Diss., 2003

[Einhaus 2009] Einhaus, Julian: *Tracking- und Prädiktionsverfahren für Fahrzeuge in komplexen Straßenverkehrsszenarien*, Technische Fakultät, Universität Bielefeld, Diplomarbeit, 2009

[Enzweiler u. Gavrila 2009] Enzweiler, Markus ; Gavrila, Dariu M.: Monocular Pedestrian Detection: Survey and Experiments. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009)

[Ess u. a. 2008] Ess, Andreas ; Leibe, Bastian ; Schindler, Konrad ; Gool, Luc V.: A mobile vision system for robust multi-person tracking. In: *CVPR '08: Proc. of the 2008 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2008

[Ferguson 1980] Ferguson, J.D.: Variable duration models for speech. In: *Symp. Application of Hidden Markov Models to Text and Speech*, 1980, S. 143–179

[Fink 2007] Fink, Gernot A.: *Markov Models for Pattern Recognition.* Springer Verlag, 2007

[Fischer u. Henrich 2009] Fischer, Markus ; Henrich, Dominik: 3D Collision Detection for Industrial Robots and Unknown Obstacles using Multiple Depth Images. In: *German Workshop on Robotics - GWR*, 2009

[Franke 2006] Franke, Andre: *Bestimmung und Prädiktion von Objektlagen in multiokularen Videosequenzen zyklischer Abläufe*, Technische Universität Ilmenau, Diplomarbeit, 2006

[Franke u. Joos 2000] Franke, Uwe ; Joos, Armin: Real-time Stereo Vision for Urban Traffic Scene Understanding. In: *Procs. IEEE Intelligent Vehicles Symposium 2000*, 2000, S. 273–278

[Fritsch u. a. 2004] Fritsch, J. ; Hofemann, N. ; Sagerer, G.: Combining Sensory and Symbolic Data for Manipulative Gesture Recognition. In: *Proc. Int. Conf. on Pattern Recognition*, 2004 (3), S. 930–933

[Fukunaga 1975] Fukunaga, K. und L. D. H.: *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. IEEE Transactions on Information Theory.* 1975

[Gall u. a. 2009] Gall, Jürgen ; Rosenhahn, Bodo ; Brox, Thomas ; Seidel, Hans-Peter: Optimization and Filtering for Human Motion Capture - A Multi-Layer Framework. In: *Int. Journal of Computer Vision* (2009), S. 1–18

[Gavrila u. Davis 1996] Gavrila, D. M. ; Davis, L. S.: 3-D model-based tracking of humans in action: a multi-view approach. In: *CVPR '96: Proc. of the 1996 Conf. on Computer Vision and Pattern Recognition*, 1996, S. 73

[Gavrila u. Munder 2007] Gavrila, Dariu M. ; Munder, Stefan: Multi-cue Pedestrian Detection and Tracking from a Moving Vehicle. In: *Int. Journal on Computer Vision* 73 (2007), Nr. 1, S. 41–59

[Gavrila 1999] Gavrila, D.M.: The Visual Analysis of Human Movement: A Survey. In: *CVIU* 73 (1999), Nr. 1, S. 82–98

[Gecks u. Henrich 2006] Gecks, Thorsten ; Henrich, Dominik: Multi-Camera Collision Detection allowing for Object Occlusions. In: *37th Int. Symposium on Robotics (ISR 2006) / 4th German Conf. on Robotics (Robotik 2006)*, 2006

[Gecks u. Henrich 2007] Gecks, Thorsten ; Henrich, Dominik: Path Planning and Execution in Fast-Changing Environments with Known and Unknown Obstacles. In: *Int.l Conf. on Intelligent Robots and Systems*, 2007

[Gecks u. Henrich 2009] Gecks, Thorsten ; Henrich, Dominik: Sensor-based Online Planning of Time-optimized Paths in Dynamic Environments. In: *GWR09, German Workshop on*

Robotics, Braunschweig, Germany, 2009

[Gonzalez u. Woods 2002] Gonzalez, Rafael ; Woods, Richard E.: *Digital Image Processing*. Prentice Hall Press, 2002

[Gorelick u. a. 2007] Gorelick, Lena ; Blank, Moshe ; Shechtman, Eli ; Irani, Michal ; Basri, Ronen: Actions as Space-Time Shapes. In: *Transactions on Pattern Analysis and Machine Intelligence* 29 (2007), Nr. 12, S. 2247–2253

[Gu u. a. 2008] Gu, J.X. ; Ding, X.Q. ; Wang, S.J. ; Wu, Y.S.: Full body tracking-based human action recognition. In: *Proc. Int. Conf. on Pattern Recognition, Tampa, USA*, 2008

[Haasch u. a. 2005] Haasch, A. ; Hofemann, N. ; Fritsch, J. ; Sagerer, G.: A Multi-Modal Object Attention System for a Mobile Robot. In: *Proceedings of the IEEE / RSJ International Conference on Intelligent Robots and Systems*, 2005, S. 1499–1504

[Hahn u. a. 2010a] Hahn, Markus ; Barrois, Björn ; Krüger, Lars ; Wöhler, Christian ; Sagerer, Gerhard ; Kummert, Franz: Robust and Accurate 3D Pose Estimation and Motion Analysis in an Industrial Environment. In: *Submitted to 3D Research* (2010)

[Hahn u. a. 2008a] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: 3D Action Recognition and Long-Term Prediction of Human Motion. In: Gasteratos, A. (Hrsg.) ; Vincze, M. (Hrsg.) ; Tsotsos, J. (Hrsg.): *Proc. Int. Conf. on Computer Vision Systems, Santorini, Greece*, 2008, S. 23–32

[Hahn u. a. 2008b] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: Spatio-Temporal 3D Pose Estimation and Tracking of Human Body Parts Using the Shape Flow Algorithm. In: *Proc. Int. Conf. on Pattern Recognition, Tampa, USA*, 2008

[Hahn u. a. 2009a] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian: A Bayesian Approach to 3D Head-Shoulder Pose Tracking. In: *Proc. of Oldenburger 3D-Tage, Oldenburg, Germany*, 2009, S. 262–269

[Hahn u. a. 2007] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Gross, Horst-Michael: Tracking of Human Body Parts using the Multiocular Contracting Curve Density Algorithm. In: *3DIM '07: Proc. of the Sixth Int. Conf. on 3-D Digital Imaging and Modeling*, 2007, S. 257–264

[Hahn u. a. 2009b] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Kummert, Franz: 3D Action Recognition in an Industrial Environment. In: *Proc. of 3rd Int. Workshop on Human-Centered Robotic Systems (HCRS'09), Bielefeld, Germany.*, 2009

[Hahn u. a. 2010b] Hahn, Markus ; Krüger, Lars ; Wöhler, Christian ; Sagerer, Gerhard ; Kummert, Franz: Spatio-temporal 3D Pose Estimation and Tracking of Human Body Parts in an Industrial Environment. In: *Proc. of Oldenburger 3D-Tage, Oldenburg, Germany*, 2010

[Hahn u. a. 2010c] Hahn, Markus ; Quronfuleh, Fuad ; Wöhler, Christian ; Kummert, Franz: 3D Mean-Shift Tracking and Recognition of Working Actions. In: *Proc. Int. Workshop on Human Behaviour Understanding, held in conjunction with ICPR 2010, Istanbul, Turkey*, 2010

[Hahn u. a. 2010d] Hahn, Markus ; Wöhler, Christian ; Einhaus, Julian ; Hermes, Christoph ; Kummert, Franz: Tracking and Motion Prediction of Vehicles in Complex Urban Traffic Scenes. In: *Proc. 4. Tagung Sicherheit durch Fahrerassistenz, München, Germany*,

2010

[Hanek 2004] Hanek, Robert: *Fitting Parametric Curve Models to Images Using Local Self-adapting Separation Criteria*, Technische Universität München, München, Diss., 2004

[Heap u. Hogg 1996] Heap, T. ; Hogg, D.: Towards 3D hand tracking using a deformable model. In: *Proc. of the 2nd Int. Conf. on Automatic Face and Gesture Recognition (FG '96)*, 1996, S. 140–145

[Henrich u. Gecks 2008] Henrich, Dominik ; Gecks, Thorsten: Multi-camera Collision Detection between Known and Unknown Objects. In: *2nd ACM/IEEE Int.l Conf. on Distributed Smart Cameras*, 2008

[Hermes u. a. 2009] Hermes, C. ; Wöhler, C. ; Schenk, K. ; Kummert, F.: Longterm vehicle motion prediction. In: *Proc. of IEEE Intelligent Vehicles Symposium, Xi'an, China*, 2009, S. 652–657

[Hermes u. a. 2010] Hermes, Christoph ; Einhaus, Julian ; Hahn, Markus ; Wöhler, Christian ; Kummert, Franz: Vehicle Tracking and Motion Prediction in Complex Urban Scenarios. In: *Proc. IEEE Intelligent Vehicles Symposium, San Diego, California*, 2010

[Hirschberg 1977] Hirschberg, Daniel S.: Algorithms for the Longest Common Subsequence Problem. In: *Journal of the Assoclauon for Computing Machinery (ACM)* 24 (1977), Nr. 4, S. 664–675

[Hofemann 2007] Hofemann, Nils: *Videobasierte Handlungserkennung für die natürliche Mensch-Maschine-Interaktion*, Universität Bielefeld, Technische Fakultät, Diss., 2007

[Hofmann u. Gavrila 2009] Hofmann, Michael ; Gavrila, Dariu M.: Multi-view 3D Human Pose Estimation combining Single-frame Recovery, Temporal Integration and Model Adaptation. In: *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2009

[Hongeng u. Nevatia 2003] Hongeng, S. ; Nevatia, R.: Large-scale event detection using semi-HMMs. In: *ICCV '03: Proc. of the Nineth Int. Conf. on Computer Vision*, 2003

[Horn u. Schunck 1980] Horn, Berthold K. ; Schunck, Brian G.: Determining Optical Flow / MIT. Massachusetts Institute of Technology, 1980. – Forschungsbericht

[Huguet u. Devernay 2007] Huguet, Frédéric ; Devernay, Frédéric: A Variational Method for Scene Flow Estimation from Stereo Sequences. In: *IEEE Eleventh Int. Conf. on Computer Vision, ICCV 07, Rio de Janeiro, Brazil*, 2007

[Isard u. Blake 1998] Isard, Michael ; Blake, Andrew: CONDENSATION—Conditional Density Propagation for Visual Tracking. In: *Int. J. of Computer Vision* 29 (1998), August, Nr. 1, S. 5–28

[Jähne 2002] Jähne, Bernd: *Digitale Bildverarbeitung.* 5., überarb. u. erw. Aufl. Springer Verlag, 2002

[Jain u. a. 1999] Jain, A. K. ; Murty, M. N. ; Flynn, P. J.: Data clustering: a review. In: *ACM Comput. Surv* 31 (1999), Nr. 3, S. 264–323

[Jain u. a. 1998] Jain, A. K. ; Zhong, Y. ; Dubuisson-Jolly, M.-P.: Deformable template models: A review. In: *Signal Processing* 71 (1998), S. 109–129

[Johansson 1973] Johansson, Gunnar: Visual perception of biological motion and a model for its analysis. In: *Perception and Psychophysics* 14 (1973), S. 201–211

[Julier u. Uhlmann 1997] Julier, S. ; Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: *International Symposium on Aerospace/Defense Sensing, Simulation and Controls*, 1997

[Kalman 1960] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME - Journal of Basic Engineering* 82 (1960), S. 35–45

[Kass u. a. 1988] Kass, Michael ; Witkin, Andrew ; Terzopoulos, Demetri: Snakes: Active contour models. In: *Int. Journal of Computer Vision* V1 (1988), January, Nr. 4, S. 321–331

[Kearsley 1989] Kearsley, Simon K.: On the orthogonal transformation used for structural comparisons. In: *Acta Cryst* A45 (1989), S. 208–210

[Keskin u. a. 2003] Keskin, C. ; Erkan, A. ; Akarun, L.: Real Time Hand Tracking and 3D Gesture Recognition for Interactive Interfaces using HMM. In: *Proc. of the Joint Conf. ICANN-ICONIP*, 2003

[Kölzow 2002] Kölzow, Thorsten: *System zur Klassifikation und Lokalisation von 3D-Objekten durch Anpassung vereinheitlichter Merkmale in Bildfolgen*, Universität Bielefeld, Technische Fakultät, Diss., 2002

[Knoop u. a. 2006] Knoop, Steffen ; Vacek, Stefan ; Dillmann, Rüdiger: Sensor fusion for 3D human body tracking with an articulated 3D body model. In: *Proc. 2006 IEEE Int. Conf. on Robotics and Automation*, 2006, 1686 - 1691

[Krüger 2007] Krüger, Lars: *Model Based Object Classification and Localisation in Multiocular Images*, University of Bielefeld, Diss., 2007

[Krüger u. Ellenrieder 2005] Krüger, Lars ; Ellenrieder, Marc M.: Pose Estimation Using the Multiocular Contracting Curve Density Algorithm. In: *Proc. of VMV, Erlangen, Germany*, 2005

[Krüger u. Wöhler 2009] Krüger, Lars ; Wöhler, Christian: Accurate chequerboard corner localisation for camera calibration and scene reconstruction. In: *Submitted to Pattern Recognition Letters* (2009)

[Krüger u. a. 2004] Krüger, Lars ; Wöhler, Christian ; Würz-Wessel, Alexander ; Stein, Fridtjof: In-factory calibration of multiocular camera systems. In: *SPIE Photonics Europe*, 2004, S. pp. 126–137

[Laptev 2005] Laptev, Ivan: On Space-Time Interest Points. In: *Int.Journal of Computer Vision* 64 (2005), S. 107 – 123

[Lepetit u. Fua 2005] Lepetit, Vincent ; Fua, Pascal: Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. In: *Found. Trends. Comput. Graph. Vis* 1 (2005), Nr. 1, S. 1–89

[Levenshtein 1966] Levenshtein, Vladimir I.: Binary codes capable of correcting deletions, insertions, and reversals. In: *Soviet Physics Doklady* 10 (1966), Nr. 8, S. 707–710

[Li u. Greenspan 2005] Li, Hong ; Greenspan, Michael: Multi-Scale Gesture Recognition from Time-Varying Contours. In: *Proc. of IEEE Int.l Conf.on Computer Vision (ICCV'05)*, 2005

[Li u. a. 2006] Li, Zhe ; Fritsch, Jannik ; Wachsmuth, Sven ; Sagerer, Gerhard: An Object-oriented Approach Using a Top-down and Bottom-up Process for Manipulative Action Recognition. In: *DAGM06* Bd. 4174, 2006 (Lecture Notes in Computer Science), S. 212–221

[Liang Wang 2003] Liang Wang, Tieniu T. Weiming Hu H. Weiming Hu: Recent Develop-

ments of Human Motion Analysis. In: *Pattern Recognition* 36 (2003), Nr. 3, S. 585–601

[Lin 1999] Lin, Michael H.: Tracking Articulated Objects in Real-Time Range Image Sequences. In: *ICCV*, 1999, S. 648–653

[Lu u. a. 2003] Lu, Shan ; Metaxas, Dimitris ; Samaras, Dimitris ; Oliensis, John: Using Multiple Cues for Hand Tracking and Model Refinement. In: *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, 2003, S. II: 443–450

[MacKay 2003] MacKay, David J. C.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003

[Marhasev u. a. 2006] Marhasev, E. ; Hadad, M. ; Kaminka, G.A.: Non-stationary hidden semi-Markov models in activity recognition. In: *Proc. of the AAAI Workshop on Modeling Others from Observations (MOO-06)*, 2006

[Marhasev u. a. 2009] Marhasev, Einat ; Hadad, Meirav ; Kaminka, Gal A. ; Feintuch, Uri: The use of hidden semi-Markov models in clinical diagnosis maze tasks. In: *Intell. Data Anal.* 13 (2009), December, S. 943–967. – ISSN 1088–467X

[Mitra u. Acharya 2007] Mitra, Sushmita ; Acharya, Tinku: Gesture Recognition: A Survey. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 37 (2007), S. 311–324

[Moeslund u. a. 2006] Moeslund, Thomas B. ; Hilton, Adrian ; Krüger, Volker: A survey of advances in vision-based human motion capture and analysis. In: *Computer Vision and Image Understanding* 104 (2006), Nr. 2, S. 90–126

[Mündermann u. a. 2007] Mündermann, L. ; Corazza, S. ; Andriacchi, T.P: Accurately measuring human movement using articulated ICP with soft-joint constraints and a repository of articulated models. In: *CVPR '07: Proc. of the 2007 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2007, S. 1–6

[Mündermann u. a. 2006] Mündermann, Lars ; Corazza, Stefano ; Andriacchi, Thomas P.: The evolution of methods for the capture of human movement leading to markerless motion capture for biomechanical applications. In: *Journal of NeuroEngineering and Rehabilitation* 3 (2006)

[Murphy 2002] Murphy, Kevin P.: *Dynamic bayesian networks: representation, inference and learning*, Diss., 2002. – Chair-Russell, Stuart

[Narayanan u. a. 2005] Narayanan, Karthik ; Kumaran, Raghunandan ; Gowdy, John: Stereo-based elliptical head tracking. In: *Proc. of European Signal Processing Conference (EUSIPCO 2005)*, 2005, S. 1565–1568

[Neuenschwander u. a. 1997] Neuenschwander, Walter M. ; Fua, Pascal ; Iverson, Lee ; Szekely, Gabor ; Kubler, O.: Ziplock Snakes. In: *Int. Journal of Computer Vision* 25 (1997), Nr. 3, S. 191–201

[Nickel u. a. 2004] Nickel, Kai ; Seemann, Edgar ; Stiefelhagen, Rainer: 3D-Tracking of Head and Hands for Pointing Gesture Recognition in a Human-robot Interaction Scenario. In: *Proc. of Sixth IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2004

[Niemann 1983] Niemann, Heinrich: *Klassifikation von Mustern*. Springer-Verlag, 1983

[Park 2004] Park, Sangho: A hierarchical bayesian network for event recognition of human actions and interactions. In: *Multimedia Systems, Sp.lss. on Video Surveillance* 10 (2004),

Nr. 2, S. 164–179

[Pavlovic u. a. 1997] Pavlovic, Vladimir I. ; Sharma, Rajeev ; Huang, Thomas S.: Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), S. 677–695

[Plänkers u. Fua 2003] Plänkers, Ralf ; Fua, Pascal: Articulated Soft Objects for Multiview Shape and Motion Capture. In: *IEEE Trans. Pattern Anal. Mach. Intell* 25 (2003), Nr. 9, S. 1182–1187

[Porikli u. Tuzel 2003] Porikli, Fatih ; Tuzel, Oncel: Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis. In: *Conf. on Computer Vision Systems, Workshop on PETS, IEEE*, 2003

[Rabiner 1990] Rabiner, Lawrence R.: A tutorial on hidden Markov models and selected applications in speech recognition. In: *Readings in speech recognition* (1990), S. 267–296

[Ramanan u. a. 2007] Ramanan, Member-Deva ; Forsyth, David A. ; Zisserman, Andrew: Tracking People by Learning Their Appearance. In: *IEEE Trans. Pattern Anal. Mach. Intell* 29 (2007), Nr. 1, S. 65–81

[Rey 1983] Rey, W. J. J.: *Introduction to Robust and Quasi-Robust Statistical Methods*. Springer-Verlag, Berlin, 1983

[Rosenhahn u. a. 2005a] Rosenhahn, Bodo ; He, Lei ; Klette, Reinhard: Automatic human model generation. In: *in Computer Analysis of Images and Patterns (CAIP*, 2005, S. 41–48

[Rosenhahn u. a. 2007] Rosenhahn, Bodo ; Kersting, Uwe ; Powell, Katie ; Klette, Reinhard ; Klette, Gisela ; Seidel, Hans-Peter: A system for articulated tracking incorporating a clothing model. In: *Machine Vision and Applications* 18 (2007), Nr. 1, S. 25–40

[Rosenhahn u. a. 2005b] Rosenhahn, Bodo ; Kersting, Uwe ; Smith, Andrew ; Gurney, Jason ; Brox, Thomas ; Klette, Reinhard: A system for marker-less human motion estimation. In: *27th DAGM Symposium*, 2005, 230–237

[Rosenhahn u. a. 2008] Rosenhahn, Bodo ; Schmaltz, Christian ; Brox, Thomas ; Weickert, Joachim ; Cremers, Daniel ; Seidel, Hans-Peter: Markerless motion capture of man-machine interaction. In: *CVPR '08: Proc. of the 2008 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2008

[Russel 2004] Russel, James: *Detecting Humans in Video Footage using Multiple Classifiers*, University of Western Australia, Diss., 2004

[Sappa u. a. 2005] *Kapitel* 1. In: Sappa, A. ; Aifanti, N. ; Grammalidis, N. ; Malassiotis, S.: *Advances in Vision-Based Human Body Modeling*. IRM Press, 2005, S. 1–26

[Schmidt 2009] Schmidt, Joachim: Monokulare Modellbasierte Posturschätzung des Menschlichen Oberkörpers. In: *Proc. of 8. Oldenburger 3D-Tage*. Oldenburg, 28.01.2009 2009

[Schmidt u. Castrillon 2008] Schmidt, Joachim ; Castrillon, Modesto: Automatic Initialization for Body Tracking - Using Appearance to Learn a Model for Tracking Human Upper Body Motions. In: *3rd Int. Conf. on Computer Vision Theory and Applications (VISAPP)*. Funchal, Madeira - Portugal, 2008

[Schmidt u. a. 2006] Schmidt, Joachim ; Fritsch, Jannik ; Kwolek, Bogdan: Kernel Particle Filter for Real-Time 3D Body Tracking in Monocular Color Images. In: *Proc. of the 7th*

Int. Conf. on Automatic Face and Gesture Recognition (FGR06), 2006, S. 567–572

[Schmidt u. a. 2008] Schmidt, Joachim ; Hofemann, Nils ; Haasch, Axel ; Fritsch, Jannik ; Sagerer, Gerhard: Interacting with a Mobile Robot: Evaluating Gestural Object References. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008, S. 3804–3809

[Schmidt u. a. 2007] Schmidt, Joachim ; Wöhler, Christian ; Krüger, Lars ; Gövert, Tobias ; Hermes, Christoph: 3D Scene Segmentation and Object Tracking in Multiocular Image Sequences. In: *The 5th Int. Conf. on Computer Vision Systems Conf. Paper*, 2007

[Schunck 1986] Schunck, Brian G.: The image flow constraint equation. In: *Comput. Vision Graph. Image Process* 35 (1986), Nr. 1, S. 20–46

[Schürmann 1996] Schürmann, Jürgen: *Pattern classification: a unified view of statistical and neural approaches.* John Wiley & Sons, Inc, 1996

[Sigal u. a. 2006] Sigal, Leonid ; Sigal, Leonid ; Black, Michael J. ; Black, Michael J.: HumanEva: Synchronized video and Motion Capture Dataset for Evaluation of Articulated Human Motion / Brown University. 2006. – Forschungsbericht

[Sin u. Kim 1995] Sin, B. ; Kim, J.H.: Nonstationary hidden Markov model. In: *Signal Process.* 46 (1995), January, S. 31–46

[Slama 1980] Slama, C.C: *Manual of Photogrammetry.* Book, 1980

[Sminchisescu 2006] Sminchisescu, Cristian: 3D Human Motion Analysis in Monocular Video Techniques and Challenges. In: *AVSS '06: Proc. of the IEEE Int. Conf. on Video and Signal Based Surveillance*, 2006

[Starck u. Hilton 2003] Starck, Jonathan ; Hilton, Adrian: Model-Based Multiple View Reconstruction of People. In: *ICCV '03: Proc. of the Ninth IEEE Int. Conf. on Computer Vision*, 2003, S. 915

[Stein 2004] Stein, Fridtjof: Efficient Computation of Optical Flow Using the Census Transform. In: *26th DAGM Symposium*, 2004, S. 79–86

[Stößel 2007] Stössel, Dirk: *Automated visual inspection of assemblies from monocular images*, Universität Bielefeld, Technische Fakultät, Diss., 2007

[Stenger 2004] Stenger, B: *Model-Based Hand Tracking Using A Hierarchical Bayesian Filter*, University of Cambridge, Diss., March 2004

[Stenger u. a. 2001] Stenger, B. ; Mendonça, P. R. S. ; Cipolla, R: Model-Based Hand Tracking Using an Unscented Kalman Filter. In: *Proc. of the British Machine Vision Conf* Bd. I, 2001, S. 63–72

[Sundaresan u. Chellappa 2006] Sundaresan, A. ; Chellappa, R: Multi-camera Tracking of Articulated Human Motion Using Motion and Shape Cues. In: *Asian Conf. on Computer Vision*, 2006, S. II:131–140

[Thiemermann 2005] Thiemermann, Stefan: *Direkte Mensch-Roboter-Kooperation in der Kleinteilemontage mit einem SCARA-Roboter*, Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA) Stuttgart, Diss., 2005

[Treptow u. a. 2005] Treptow, A. ; Cielniak, G. ; Duckett, T: Comparing Measurement Models for Tracking People in Thermal Imageson a Mobile Robot. In: *Proc. of the European Conf. on Mobile Robots (ECMR)*, 2005

[Triggs u. a. 2000] Triggs, Bill ; McLauchlan, Philip F. ; Hartley, Richard I. ; Fitzgibbon, Andrew W.: Bundle Adjustment - A Modern Synthesis. In: *ICCV '99: Proc. of the Int. Workshop on Vision Algorithms*, 2000, S. 298–372

[Turaga u. a. 2008] Turaga, Pavan ; Chellappa, Rama ; Subrahmanian, V. S. ; Udrea, Octavian: Machine Recognition of Human Activities: A Survey. In: *IEEE Transactions on Circuits and Systems for Video Technology* 18 (2008), S. 1473–1488

[Tyagi u. a. 2007] Tyagi, Ambrish ; Keck, Mark ; Davis, James W. ; Potamianos, Gerasimos: Kernel-Based 3D Tracking. In: *CVPR '07: Proc. of the 2007 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2007

[Vaseghi 1995] Vaseghi, S. V.: State duration modelling in hidden Markov models. In: *Signal Process.* 41 (1995), January, S. 31–41

[Viola u. a. 2003] Viola, Paul ; Jones, Michael J. ; Snow, Daniel: Detecting Pedestrians Using Patterns of Motion and Appearance. In: *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, 2003, S. 734

[Vlachos u. a. 2002] Vlachos, Michail ; Gunopoulos, Dimitrios ; Kollios, George: Discovering Similar Multidimensional Trajectories. In: *Proc. of Int. Conf. Data Engineering*, 2002

[Vogler u. Metaxas 1998] Vogler, Christian ; Metaxas, Dimitris: ASL Recognition Based on a Coupling Between HMMs and 3D Motion Analysis. In: *ICCV '98: Proc. of the Sixth Int. Conf. on Computer Vision*, 1998, S. 363–369

[Waibel u. a. 1990] Waibel, Alexander ; Hanazawa, Toshiyuki ; Hinton, Geofrey ; Shikano, Kiyohiro ; Lang, Kevin J.: Phoneme recognition using time-delay neural networks. (1990), S. 393–404

[Wedel u. a. 2008] Wedel, Andreas ; Pock, Thomas ; Zach, Christopher ; Bischof, Horst ; Cremers, Daniel: An improved algorithm for TV-L1 optical flow computation. In: *Proc. of the Dagstuhl Visual Motion Analysis Workshop*, 2008

[Weisstein 2008] Weisstein, Eric W.: *From MathWorld–A Wolfram web resource.* WWW: `http://mathworld.wolfram.com`, 2008

[Welch u. Bishop 1995] Welch, Greg ; Bishop, Gary: An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill, 1995. – Forschungsbericht

[Williams u. Shah 1992] Williams, Donna J. ; Shah, Mubarak: A fast algorithm for active contours and curvature estimation. In: *CVGIP: Image Understanding* 55 (1992), Nr. 1, S. 14–26

[Winkler 2007] Winkler, B.: Safe space sharing human-robot cooperation using a 3D time-of-flight camera. In: *Int. Robots & Vision Show*, 2007

[Winkler 2006] Winkler, K.: *Three Eyes Are Better than Two. SafetyEYE uses technical image processing to protect people at their workplaces.* In: DaimlerChrysler Hightech Report 12/2006, DaimlerChrysler AG Communications, Stuttgart, Germany, 2006

[Wöhler 2009] Wöhler, Christian: *3D Computer Vision. Efficient Methods and Applications.* Springer Verlag, 2009

[Wöhler u. Anlauf 1999] Wöhler, Christian ; Anlauf, Joachim K.: An Adaptable Time Delay Neural Network Algorithm for Image Sequence Analysis. In: *IEEE Transactions on*

Neural Networks 10 (1999), Nr. 6, S. 1531–1536

[Wu u. a. 2005] Wu, Ying ; Lin, John ; Huang, Thomas S.: Analyzing and Capturing Articulated Hand Motion in Image Sequences. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (2005), Nr. 12, S. 1910–1922

[Xu u. Prince 1998] Xu, Chenyang ; Prince, Jerry L.: Snakes, shapes, and gradient vector flow. In: *IEEE Transactions on Image Processing* 7 (1998), Nr. 3, S. 359–369

[Yamato u. a. 1992] Yamato, Junji ; Ohya, Jun ; Ishii, Kenichiro: Recognizing Human Action in Time-sequential Images using Hidden Markov Model. In: *CVPR '92: Proc. of the 1992 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 1992, S. 379–385

[Yang u. a. 2003] Yang, C. ; Duraiswami, R. ; Gumerov, N. A. ; Davis, L: Improved fast gauss transform and efficient kernel density estimation. In: *Computer Vision, 2003. Proc.. Ninth IEEE Int. Conf. on*, 2003, S. 664–671 vol.1

[Yang u. Ahuja 1999] Yang, Ming-Hsuan ; Ahuja, Narendra: Recognizing Hand Gesture using Motion Trajectories. In: *CVPR '99: Proc. of the 1999 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Volume 1*, 1999

[Yilmaz u. a. 2006] Yilmaz, Alper ; Javed, Omar ; Shah, Mubarak: Object tracking: A survey. In: *ACM Comput. Surv* 38 (2006), Nr. 4, S. 13

[Yilmaz u. Shah 2005] Yilmaz, Alper ; Shah, Mubarak: Actions Sketch: A Novel Action Representation. In: *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, 2005, S. 984–989

[Yu 2010] Yu, Shun-Zheng: Hidden semi-Markov models. In: *Artificial Intelligence* 174 (2010), Nr. 2, 215 - 243. `http://dx.doi.org/DOI:10.1016/j.artint.2009.11.011`. – DOI DOI: 10.1016/j.artint.2009.11.011. – ISSN 0004–3702. – Special Review Issue

[Yuille u. a. 1989] Yuille, A. ; Cohen, D. ; Halliman, P.: Feature extraction from faces using deformable templates. In: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*, 1989, S. 104– 109

[Zhang 1992] Zhang, Zhengyou: Iterative Point Matching for Registration of Free-Form Curves / INRIA. 1992. – Forschungsbericht

[Zhong u. a. 2004] Zhong, Hua ; Shi, Jianbo ; Visontai, Mirkó: Detecting Unusual Activity in Video. In: *CVPR '04: Proc. of the 2004 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition - Volume 2*, 2004, S. 819 – 826

[Zhu u. a. 2006] Zhu, Guangyu ; Xu, Changsheng ; Gao, Wen ; Huang, Qingming: Action Recognition in Broadcast Tennis Video Using Optical Flow and Support Vector Machine. In: *Proc. of ECCV Workshop on HCI*, 2006

[Ziegler u. a. 2006] Ziegler, Julius ; Nickel, Kai ; Stiefelhagen, Rainer: Tracking of the Articulated Upper Body on Multi-View Stereo Image Sequences. In: *CVPR '06: Proc. of the 2006 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, 2006, 774–781