Natürlich-räumliche Industrieroboterprogrammierung auf Basis markerloser Gestenerkennung und mobiler Augmented Reality

vorgelegt von **Dipl.-Ing. Jens Lambrecht**
geb. in Hamm (Westf.)

von der Fakultät V - Verkehrs- und Maschinensysteme der Technischen Universität Berlin
zur Erlangung des akademischen Grades Doktor der Ingenieurwissenschaften
- Dr.-Ing. -
genehmigte Dissertation

Promotionsausschuss:
Vorsitzender: Prof. Dr.-Ing. Rainer Stark
Berichter: Prof. Dr.-Ing. Jörg Krüger
Berichter: Prof. Dr.-Ing. Gunther Reinhart

Tag der wissenschaftlichen Aussprache: 15.10.2014

Berlin 2014

Vorwort und Danksagung des Autors

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter des Fachgebiets Industrielle Automatisierungstechnik am Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der TU Berlin.

Prof. Dr.-Ing. Jörg Krüger gilt mein großer Dank für die umfangreiche Unterstützung, die wohlwollende Förderung sowie das entgegengebrachte Vertrauen.

Ich danke außerdem *Prof. Dr.-Ing. Gunther Reinhart* vom Institut für Werkzeugmaschinen und Betriebswissenschaften (IWB) der TU München für die Übernahme der Gutachtertätigkeit sowie dem Vorsitzenden des Promotionsausschusses *Prof. Dr.-Ing. Rainer Stark*, Leiter des Fachgebiets Industrielle Informationstechnik am IWF der TU Berlin.

Bei den zahlreichen Kollegen des Fachgebiets und des Fraunhofer IPK möchte ich mich für die Zusammenarbeit, für die angenehme Arbeitsatmosphäre sowie für die vielen konstruktiven Beiträge zu dieser Arbeit bedanken. Namentlich zu erwähnen sind mein Bürokollege *Matthias Blankenburg* sowie *Axel Vick*, *Moritz Chemnitz*, *Christian Horn* und *Martin Kleinsorge*. An dieser Stelle geht spezieller Dank an *Udo Templiner* und *Martin Rosenstrauch* für die umfangreiche Unterstützung bei der Betreuung von Lehrveranstaltungen. Für die kritische Begutachtung und vielen wertvollen Anregungen bezüglich des Manuskripts möchte ich mich bei *The Duy Nguyen*, *Matthias Brüning* sowie besonders bei *Franziska* und *Eugen Funk* bedanken. Weiter danke ich *Dr.-Ing. Marc Osthues* für anregende Diskussionen und den wertvollen Erfahrungsaustausch zur Anfertigung der Dissertation.

Ein ganz besonderer Dank geht an *Rosa Maria Rühling* für viel Geduld, Unterstützung und Verständnis. Ebenfalls bedanke ich mich bei ihren Eltern *Maria Eugenia* und *Andreas*.

Meinen Eltern *Ursula* und *Gerhard*, die mir die akademische Ausbildung ermöglicht und mich immer motiviert haben, möchte ich in besonderer Weise danken. Abschließender Dank geht an meine Brüder *Mark* und *Kai* für den persönlichen Austausch und die moralische Unterstützung.

Berlin, Oktober 2014 Jens Lambrecht

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen und Formelzeichen

Abkürzungen

An dieser Stelle werden fachspezifische Abkürzungen definiert, die kapitelübergreifend genutzt werden. Abkürzungen, welche lediglich in einzelnen Kapiteln genutzt werden, werden in den jeweiligen Kapiteln gesondert definiert.

App	Applikation
AR	Augmented Reality
CAD	Computer-aided Design
$CAVE$	Cave Automatic Virtual Environment
DH	Denavit-Hartenberg
DoF	Degrees of Freedom
fps	Frames per Second
GPS	Global Positioning System
GUI	Graphical User Interface
HMD	Head-Mounted Display
KMU	Kleine und mittlere Unternehmen
KS	Koordinatensystem
MES	Manufacturing Execution System
MMI	Mensch-Maschine-Interaktion
MMS	Mensch-Maschine-Schnittstelle
OLP	Offline-Programmiersystem
OST	Optical See Through
PbD	Programming by Demonstration
PTP	Point to Point
SPS	Speicherprogrammierbare Steuerung
TCP	Tool Center Point
TCP/IP	Transmission Control Protocol/Internet Protocol
VR	Virtual Reality
VST	Video See Through
$WLAN$	Wireless Local Area Network

Formelzeichen

Vektoren werden durch fettgedruckte Kleinbuchstaben beschrieben, bspw. $\boldsymbol{a}$*. Matrizen werden als fettgedruckte Großbuchstaben bezeichnet, bspw.* $\boldsymbol{A}$*.*

Bereich	Zeichen	Bedeutung
Allgemein	$\boldsymbol{p}$	Punkt/Punktwolke
	$\boldsymbol{R}$	Rotationsmatrix
	$\boldsymbol{S}$	Skalierungsmatrix
	$\boldsymbol{T}_A^B$	Homogene Transformationsmatrix von Koordinatensystem A nach Koordinatensystem B
	X	Raumkoordinate in X-Richtung
	Y	Raumkoordinate in Y-Richtung
	Z	Raumkoordinate in Z-Richtung
Roboterkinematik	$\boldsymbol{\theta}$	Vektor der Achsvariablen
	$\dot{\theta}$	Achsgeschwindigkeit
	Q	Einheitsquaternion
	r	Radius
	t	Zeit
	$\boldsymbol{t}$	Translationsvektor
	$\boldsymbol{x}$	Kartesische Endeffektorkoordinaten
Finger-/Hand-Tracking	$p(a\|b)$	Wahrscheinlichkeit von a unter der Bedingung b
	$\boldsymbol{P}$	Kameramatrix
	R, r	Radien
	UV	Chrominanz, bestehend aus den Unterkomponenten U und V
	π	Gewichtungsfaktor der Partikel
	$\boldsymbol{W}$	Shape Matrix
	$\boldsymbol{x}_t$	Zustandsvektor im Zeitschritt t
	Y	Luminanz
	z	Messung
Splines	$\boldsymbol{B}$	Matrix der Basisfunktionen der B-Splines
	C	Beschreibung der Spline-Kurve
	$\boldsymbol{N}$	Matrix segmentweiser Basisfunktionen der B-Splines und NURBS
	$\boldsymbol{p}$	Kontrollpunkte der Spline-Kurve
	$\boldsymbol{t}$	Trägervektor
	w	Gewichtungsfaktor der Kontrollpunkte

Kurzfassung

Vor dem Hintergrund steigender Anforderungen an produzierende Unternehmen im globalen Wettbewerb übernimmt die Industrierobotik in den westlichen Industrienationen eine Schlüsselrolle zur Flexibilisierung und kosteneffizienten Gestaltung von Produktionsprozessen. Als Flaschenhals erweist sich für kleine und mittlere Unternehmen oftmals der Prozess der manuellen Programmierung von Industrierobotern. Dementsprechend besteht die Motivation, den Programmierprozess effizienter und einfacher zu gestalten. Zu diesem Zweck soll die Mensch-Roboter-Kommunikation der natürlichen Mensch-Mensch-Kommunikation angenähert werden.
In dieser Arbeit wird ein Interaktionskonzept unter der Verwendung natürlicher Kommunikation in Form von markerloser Gestenerkennung vorgestellt. Der Anwender wird in die Lage versetzt, das Programm im Sinne des Paradigmas „Programming by Demonstration“ durch gestenbasiertes Vormachen zu definieren. Dabei zeichnet der Anwender die Posen oder Bahnen aus freier Hand in den Raum. Zur Adressierung von Nutzergruppen mit unterschiedlichem Qualifikationsgrad wird neben der bewegungsorientierten Programmierebene eine aufgabenorientierte Programmierung betrachtet. Kombiniert wird die prozessnahe Definition des Roboterprogramms mit Simulationstechniken der Augmented Reality. Durch den Einsatz von Handheld-Geräten wird eine mobile Visualisierung und Simulation des Roboterprogramms auf beliebigen Smartphones und Tablet-PCs ermöglicht. Zudem entsteht durch synergetische Kombination von Gesten und Augmented Reality eine intuitive Art der Interkation, die es dem Anwender ermöglicht, Roboterprogramme über natürliche Bewegungen interaktiv im Raum zu manipulieren.
Die Umsetzung des Interaktionskonzepts in Form eines Programmiersystems erfolgt kosteneffizient durch die Verwendung von Motion-Tracking-Sensorik aus dem Consumer-Bereich. Die zentrale Programmierumgebung stellt eine Android-App, welche mit einer vereinheitlichten Schnittstelle zur Übertragung und Ausführung der Programme auf beliebigen Robotersteuerungen ausgestattet wird. Darüber hinaus erfolgt eine drahtlose Anbindung der App an betriebliche Informations- und Kommunikationssysteme.
Zur Erprobung der aufgabenorientierten Programmierebene wurde ein Pick-and-Place-Prozess umgesetzt, bei dem der Anwender virtuelle Objekte im Raum manipuliert. Aus der Wahrnehmung der Manipulation lässt sich automatisiert das entsprechende Roboterprogramm generieren. Eine umfangreiche Nutzerstudie vergleicht das räumliche Programmiersystem mit der klassischen Teach-In-Programmierung sowie der Offline-Programmierung. Verglichen werden die Kriterien der Programmierdauer, der Programmierfehler und eine subjektive Bewertung der Programmierverfahren, um Aussagen über die allgemeinen Ziele der Effizienz, Effektivität und Nutzerfreundlichkeit zu treffen. Die Auswertung der Studie belegt eine signifikante Reduzierung der Programmierdauer sowie eine verbesserte Wahrnehmung der Nutzerzufriedenheit für die räumliche Programmierung. Zudem wird im Vergleich zum Teach-In eine Verringerung der Programmierfehler erreicht.

Abstract

Against the background of increasing demands on manufacturing companies in global competition, industrial robotics plays a key role shaping flexibility and cost-efficiency of production processes in western industrialized countries. In regards to small and medium-sized enterprises, manual robot programming often turns into a bottleneck. Accordingly, there is the motivation to ease the programming process and make it more time-efficient. To this end, scientific approaches adapt human-robot-communication inspired by human-human-communication.

Within this thesis, an interaction concept for industrial robot programming is presented taking into account natural communication through markerless gesture recognition. The programmer is enabled to define the robot program by gestures following the principle "programming by demonstration". Therefore, the programmer is capable of drawing poses and trajectories into space through natural 3D-bare-hand interaction. In terms of the individual support of user groups with varying levels of qualification, a task-oriented as well as a motion-oriented programming level is considered. The gestural definition of the robot program is combined with an Augmented Reality simulation. Due to the use of handheld devices, mobile visualisation and simulation are carried out on arbitrary smartphones and tablet PCs. Based on the synergetic combination of gestures and Augmented Reality, a novel kind of intuitive interaction arises providing interactive manipulation of the robot program in space.

The implementation of the interaction concept within a programming system is conducted cost-efficiently through the use of motion tracking sensors from the consumer domain. The central programming environment is an Android app, which is equipped with a unified interface for the transmission and execution of programs on arbitrary robot controllers. In addition, a wireless connection between app and enterprise information systems is established.

The task-oriented programming level is tested within a pick-and-place scenario enabling the programmer to manipulate virtual objects. Based on the perception of the gestural interaction, a robot program is derived automatically from task demonstration. With the help of a user study, the programming duration, programming errors and subjective assessment compared with Teach-In and Offline Programming are evaluated. The analysis of the results shows a significant reduction of programming duration as well as a reduction of programming errors compared with Teach-In. Furthermore, most participants favor the spatial programming system.

1 Einleitung

1.1 Motivation und Problemstellung

Neben Kosten- und Zeitdruck stellen auch der demografische Wandel und der Fachkräftemangel zunehmende Herausforderungen an die Produktionstechnik in vielen westlichen Industrienationen. Aufgrund der im internationalen Vergleich hohen Lohnkosten sind Adaptionsvermögen und Automatisierung von Produktionssystemen unerlässlich. Die Automatisierungstechnik und die industrielle Informationstechnik übernehmen demgemäß eine hohe gesellschaftliche Verantwortung zur Sicherung der Wettbewerbsfähigkeit der produzierenden Industrie [1]. Der Paradigmenwechsel zu einer humanzentrierten Automatisierungstechnik beschreibt eine Abkehr von autarken technischen Lösungen hin zu einer kooperativen Einbindung des Menschen mit dem übergeordneten Ziel, dessen Lebensqualität zu erhöhen. Um dieses Ziel zu erreichen, müssen die technischen Systeme mit Mensch-Maschine-Funktionen ausgestattet werden, welche die Interaktion trotz Komplexität einfach, effizient und möglichst fehlerfrei gestalten. Bezüglich der demografischen Herausforderungen umfasst diese Zielstellung auch eine zunehmende Adressierung von fachfremdem und weniger bzw. geringer qualifiziertem Personal. Sogenannte „Assistenzsysteme“ versprechen als Antwort eine nachhaltige und effiziente Gestaltung von manuellen Arbeitsabläufen u. a. in der Produktion. Assistenzsysteme sollen bei Entscheidungs-, Engineering- und Dienstleistungsprozessen unterstützen und durch ihre Nutzung die Kompetenz des handelnden Menschen erhöhen [2].

1.2 Industrieroboterprogrammierung als Schlüsseltechnologie

Der Industrieroboter ist die Kernkomponente von intelligenten Automatisierungssystemen. Als flexibel einsetzbares Produktionsmittel ist er in der Lage, sowohl Handhabungs- als auch Fertigungsaufgaben automatisiert auszuführen. Im Jahr 2010 waren weltweit 1.035.000 Industrieroboter im Einsatz. 14.000 neue Industrieroboter wurden in diesem Jahr allein in Deutschland neu ausgeliefert. Die Haupteinsatzgebiete von Industrierobotern sind die Handhabung (39 %), das Schweißen (30 %) und die Montage (9 %) [3]. Zur Inbetriebnahme und Anpassung der robotergestützten Produktion muss ein Programmierprozess durchlaufen werden. Dieser wird aufgrund hoher Anforderungen an Kognition und Abstraktionsfähigkeit zumeist manuell durchgeführt. Die manuelle Online-Programmierung als vorrangig eingesetztes Verfahren stellt einen kritischen Prozess dar, der sich nicht selten als Flaschenhals erweist. Zum einen erfordert die Programmierung einen hohen Grad an Expertise. Heutzutage sind mehrtägige bis zu mehrwöchige Schulungen notwendig,

	WAHRSCHEINLICHKEIT DER ANWENDUNG	WERTZUWACHS	KOMPLEXITÄT (ALS HÜRDE FÜR DEN MARKTEINSATZ)
Sensorfusion	hoch	hoch	hoch
Mensch-Interaktion	hoch	hoch	hoch
Systemintegration	hoch/zwingend	mittel/hoch	mittel
Kognitive und lernende Systeme	niedrig	mittel/hoch	hoch
Bildverarbeitungssysteme	hoch	mittel	mittel/hoch
Positioniersysteme	mittel/hoch	mittel	mittel
Mobilität & Bewegung	mittel	mittel	mittel
Greifen/Ablegen	mittel	mittel	mittel

Tabelle 1.1: Schlüsseltechnologien der Robotik (Hardware und Software), Auszug nach [6].

um die Programmierung eines Industrieroboters zu erlernen [4]. Zum anderen gestaltet sich der Programmierprozess oftmals sehr komplex und zeitintensiv. Aus diesem Grund haben vor allem kleine und mittlere Unternehmen (KMU) immer noch Vorbehalte gegenüber Investitionen in Industrieroboter [5]. Dies führt dazu, dass Potential zur optimierten Gestaltung von Produktionssystemen oftmals aufgrund fehlender technischer bzw. wirtschaftlicher Lösungen ungenutzt bleibt. Bestätigt wird diese Aussage durch eine kürzlich veröffentlichte Studie im Auftrag der Europäischen Kommission [6]. Die Studie identifiziert gleichzeitig die Mensch-Roboter-Interaktion als Schlüsseltechnologie mit hohem Potential zur Anwendung (s. Tab. 1.1) und fordert daraus folgend die „Roboterprogrammierung für jedermann“ [6]. Für die Forschung und Entwicklung im Bereich der industriellen Robotik gilt demnach die Maxime, neue Programmiertechniken zu entwickeln, die im Vergleich zu den bekannten Methoden eine verbesserte Interaktion mit dem technischen System bereitstellen. Eine weitere Studie [7], basierend auf einer Befragung von Unternehmen im Logistikbereich, stellt fest, dass sich 71 % der Unternehmen vorstellen können, bestehende Roboterlösungen mit neuen intuitiven Programmierverfahren nachzurüsten. Außerdem bewerten 81 % der befragten Unternehmen intuitive Programmierlösungen als relevant und attestieren entsprechenden Verfahren einen starken Einfluss für die Investition in Industrierobotersysteme.
Neben der allgemeinen Motivation der Anwender zur Nutzung neuer Programmiertechniken sehen sich Hersteller von Industrierobotern mit einer annähernden Sättigung im traditionellen Markt konfrontiert [6]. Innovative Technologien bieten den Herstellern die Möglichkeit der Kundenbindung sowie der strategischen Erschließung neuer Anwendergruppen und Anwendungsfelder durch herausragende Leistungsmerkmale. Ergänzend zu allgemeinen Trends in der industriellen Robotik hin zur Bereitstellung von Systemlösungen und technischen Dienstleistungen haben im Bereich der Roboterprogrammierung in den letzten Jahren verschiedene Produktinnovationen ihren Weg in Standardsysteme gefunden. Als Beispiele können kabellose Handbediengeräte und die Verwendung von Touch-Gesten, bspw. reisPad und Comau-Programmiergerät, für die manuelle Teach-In-Programmierung aufgeführt werden. In diesem Zusammenhang bleibt festzustellen, dass sich die äußerliche Gestaltung und die Eigenschaften der Programmiergeräte zunehmend an modernen Mobilgeräten aus dem Consumer-Bereich, im Speziellen Tablet-PCs, orientieren.

1.3 Neue Technologien für die Mensch-Technik-Interaktion

Im scheinbaren Gegensatz zu einer alternden Gesellschaft wächst eine neue Generation der „Digital Natives“[1] heran. Dabei handelt es sich um jene technikaffine Bevölkerungsgruppe, die mit Computern, Smartphones, Internet etc. aufgewachsen ist und die Technologien im Alltagsleben aus einer Selbstverständlichkeit heraus nutzt. Im Jahr 2012 nutzten laut Branchenverband BITKOM bereits 21 Mio. Deutsche Applikationen (Apps) auf Mobilgeräten [9]. Insbesondere durch die mobilen Technologien stehen scheinbar beliebige Informationen unabhängig vom Standort in Sekundenschnelle zur Verfügung. Gestenbasierte Techniken der Mensch-Maschine-Kommunikation, bspw. Touch-Gesten, ermöglichen dank der Nutzung natürlicher Kommunikationskanäle in Anlehnung an die Mensch-Mensch-Kommunikation und anwendungsgerechter Gestaltung eine intuitive, d. h. eine einfache und effiziente Bedienung. Im Bereich der Videospiele werden jüngst alternativ zu klassischen haptischen Eingabegeräten Kameras zur gestenbasierten 3D-Eingabe eingesetzt. Diese Sensoren haben aufgrund der Leistungsfähigkeit und Robustheit längst neue Anwendungsgebiete in der Forschung und Entwicklung erschlossen. Speziell der Kinect-Sensor verfügt über einen vergleichbaren technischen Leistungsumfang wie industriell eingesetzte PMD-Kameras [10]. Jedoch liegt die Kinect mit einem Anschaffungspreis von ca. 100 € bei nur ca. 5–20 % des Anschaffungspreises von gängiger industrieller 2,5D-Sensorik.
In der Produktion ist eine stetig zunehmende Nutzung mobiler, vernetzter Technologien zu beobachten. Ubiquität, multifunktionale Einsetzbarkeit, einfache Bedienung sowie ein kontinuierlich erhöhter Funktions- und Leistungsumfang von Smartphones und Tablet-PCs erhöhen ständig die Anzahl der Anwendungsfelder. Aktuell setzen 34 % der Unternehmen in Deutschland Tablet-PCs ein [11]. Werden diese momentan noch zumeist zu Vertriebs- und Marketingzwecken gebraucht, besteht durch dezentrale Informations- und Kommunikationsstrukturen großes Potential im Rahmen der Produktionsplanung, -überwachung und -steuerung, bspw. zum mobilen Einsatz von Manufacturing-Execution-Systemen (MES). Das Potential der mobilen Eingabegeräte in der Produktion liegt neben der ubiquitären Einsetzbarkeit vor allem in der Möglichkeit der effizienten Modifizierung von Steuerungsaufgaben durch die gestenbasierte Interaktion [12].

1.4 Spezifizierung des Anwenderbereichs und eigener Beitrag

In den vorangegangenen Absätzen wurde dargelegt, dass die Industrieroboterprogrammierung aufgrund der allgemeinen Qualifikations- und Kostenproblematik der KMU an neue Bedürfnisse und Anforderungen angepasst werden muss. Die Nutzung neuer Formen der Mensch-Maschine-Interaktion (MMI) sowie mobiler Technologien bietet diese Möglichkeit und verspricht einen weiterführenden Mehrwert durch enge Integration der Roboterprogrammierung in gegenwärtige und zukünftige Informations- und Kommunikationsstrukturen der Produktion. Infolgedessen ist es Zielsetzung dieser Arbeit, ein neuartiges Programmiersystem für Industrieroboter auf Basis einer intuitiven, mobil einsetzbaren Form der Mensch-Maschine-Interaktion zu entwickeln. Die Konzeption des Programmiersystems soll sich speziell auf KMU und deren Forderungen nach

[1] Vgl. Begriffsdefinition in [8].

geringen Kosten und einfacher Bedienbarkeit beziehen. Der wissenschaftliche Beitrag der vorliegenden Arbeit besteht in der methodischen Erarbeitung eines Interaktionskonzepts unter Betrachtung natürlicher Kommunikation für die prozessnahe Industrieroboterprogrammierung sowie dessen prototypischer Umsetzung, Erprobung und Evaluation. Im Vergleich zu bestehenden Ansätzen in Forschung und Entwicklung werden zu diesem Zweck die Modalitäten Gesten und Augmented Reality (AR) kombinatorisch betrachtet und bezüglich synergetischer Gebrauchstauglichkeit für die Industrieroboterprogrammierung ausgelegt.
Kapitel 2 dieser Arbeit legt zunächst Grundsätze und aktuelle Entwicklungen von Mensch-Maschine-Schnittstellen (MMS) dar. Parallel wird der Stand der Industrieroboterprogrammierung dargestellt. Im Weiteren werden aktuelle Forschungsansätze zu intuitiven Benutzerschnittstellen im Bereich der industriellen Robotik vorgestellt. Darauf aufbauend erfolgt in Kapitel 3 eine Analyse der Defizite dieser Ansätze, resultierend in einer Präzisierung der Zielstellung der vorliegenden Arbeit. In Kapitel 4 werden anhand einer Anforderungsanalyse Teilaufgaben für das zu erstellende Programmiersystem abgeleitet. Die Definition der Teilaufgaben dient im Folgenden zur Erstellung eines Interaktionskonzepts und des Entwurfs des Programmiersystems. Kapitel 5 beschreibt die prototypische Umsetzung und in Kapitel 6 erfolgt die Erprobung und Evaluation durch die Darlegung eines speziellen Anwendungsszenarios sowie die Durchführung und Auswertung einer Nutzerstudie. Abschließend gibt Kapitel 7 eine Zusammenfassung sowie einen Ausblick.

2 Stand der Technik zur Mensch-Maschine-Interaktion und Industrieroboterprogrammierung

In diesem Kapitel soll ein integrativer Überblick über Grundlagen sowie jüngste Forschungs- und Entwicklungsansätze in der Mensch-Maschine-Interaktion und Industrieroboterprogrammierung gegeben werden. In Abschnitt 2.1 werden zunächst Grundlagen der Mensch-Maschine-Interaktion aufgeführt. Weiterführend werden Trends zu natürlichen Benutzerschnittstellen mit Fokus auf die Modalitäten Gesten und Augmented Reality beschrieben. Anschließend werden innovative Benutzerschnittstellen unter Verwendung dieser Modalitäten vorgestellt. Abschnitt 2.2 befasst sich mit den Grundlagen der Industrieroboterprogrammierung, während Abschnitt 2.3 intuitive Forschungsansätze zur Industrieroboterprogrammierung durch intuitive Gestaltung der Mensch-Maschine-Interaktion beschreibt.

2.1 Mensch-Maschine-Interaktion

2.1.1 Mensch-Maschine-Interaktion in industriellen Steuerungssystemen

Optimierte Mensch-Maschine-Schnittstellen unterstützen in (teil-)automatisierten technischen Systemen geistige Tätigkeiten wie das Planen, Programmieren, Diagnostizieren, Überwachen, Instandhalten und Warten. Der Mensch wird in derartigen Systemen aufgrund seiner Fähigkeiten zur Abstraktion, Interpolation und Anpassung benötigt und bspw. durch eine Software, welche in technischen Systemen genau vorgegebene, programmierte Abläufe verfolgt, unterstützt. Eine intelligente Abweichung von der vorprogrammierten Verhaltensweise der Software unter Beachtung der Umwelt ist im Allgemeinen nur beschränkt durch Nutzung von Methoden der künstlichen Intelligenz möglich [13].
Moderne industrielle Steuerungssysteme zeichnen sich in der Regel durch einen hohen Funktionsumfang und komplexe Bedien-, Überwachungs- und Koordinationsaufgaben aus. Im Kontrast orientiert sich die Gestaltung von Consumer-Produkten primär an den Bedürfnissen und Fähigkeiten der Nutzer und stellt demgemäß nur einen beschränkten, leicht bedienbaren Umfang an Steuerungsfunktionen zur Verfügung. Die anwendungsgerechte Anpassung der Bedienung an den Nutzer ist in industriellen Systemen jedoch aufgrund der komplexen Eigenschaften oft nur mit Einschränkungen möglich bzw. mit einem hohen Aufwand verbunden [14].

Ziel der Gestaltung der MMS in der industriellen Steuerungstechnik ist es, eine effektive und wirtschaftliche Prozessführung des Informationsaustausches zwischen Mensch und technischem System zu realisieren. Durch angepasste Interaktionskonzepte kann neben einer Effizienzsteigerung die Akzeptanz des technischen Systems gesteigert sowie eine motivierende und ansprechende Gestaltung der Arbeitstätigkeit ermöglicht werden. T. Stiedl [15] subsumiert den Aufbau der MMS industrieller Steuerungssysteme in Bediensoftware, Bedienhardware sowie Interaktionskanäle zur Benutzerinteraktion und Anbindung an das computerbasierte Steuerungssystem (s. Abb. 2.1). Benutzer und System werden demnach in die Lage versetzt, über bereitgestellte Interaktionskanäle miteinander zu kommunizieren. Dabei wird je nach Richtung des Informationsflusses unterschieden in Eingabe und Rückgabe bzw. Ausgabe. Zur Eingabe werden im industriellen Umfeld als Bedienhardware typischerweise Taster, Schalter oder andere Sensorik eingesetzt. Die Ausgabe erfolgt primär visuell in Form von Displays oder akustisch, bspw. über Lautsprecher und Sprachausgabe. Die zugrunde liegende Bediensoftware liefert die entsprechenden Funktionen zur Umsetzung der Eingaben bzw. zur Bereitstellung der Ausgabe.

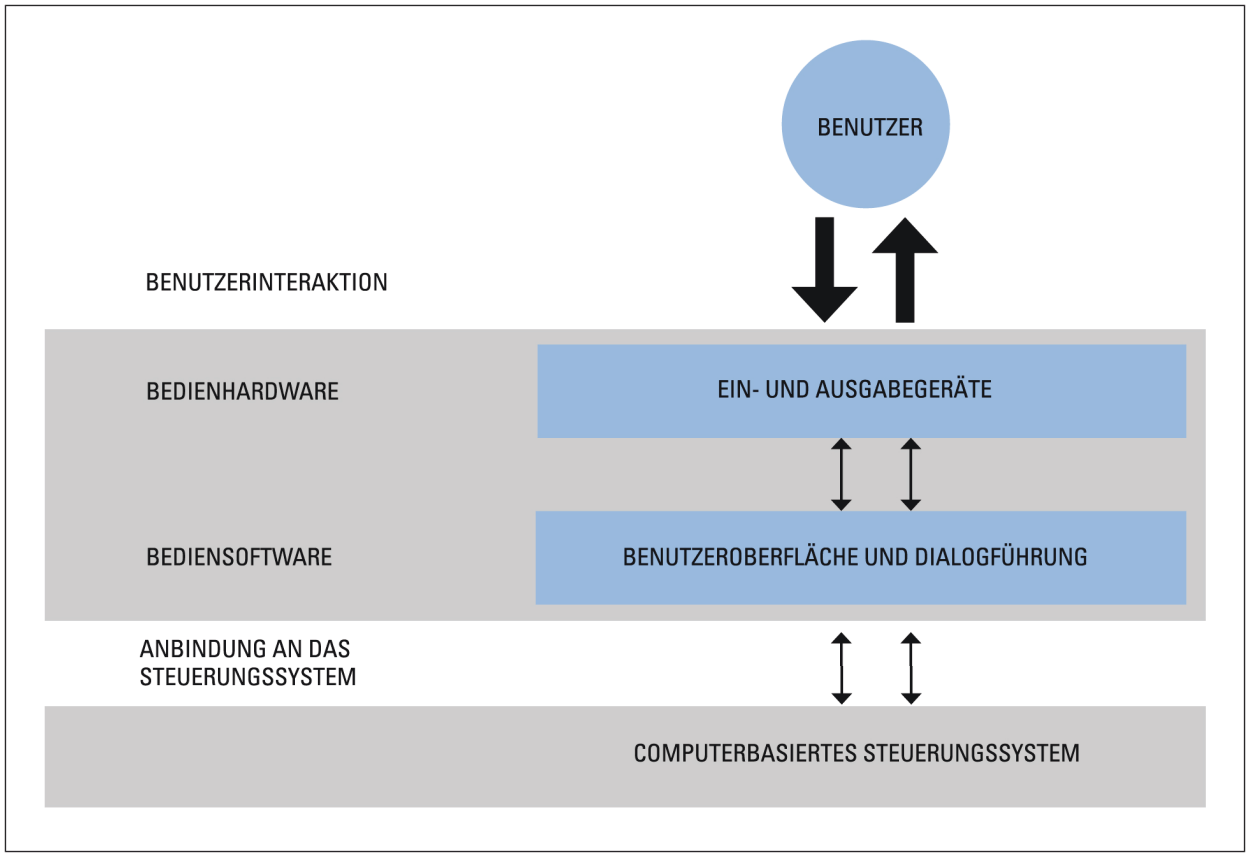

Abbildung 2.1: Aufbau einer Schnittstelle zur Mensch-Maschine-Interaktion in der Steuerung automatisierter Produktionssysteme nach [15].

2.1.2 Grundlagen zur Gestaltung von Benutzerschnittstellen

Die grundlegende Norm zur Gestaltung von Mensch-Computer-Interaktion DIN EN ISO 9241 beschäftigt sich vorrangig mit der ergonomischen Gestaltung für Bürotätigkeiten an Bildschirmgeräten und definiert in Teil 11[1] die Leitkriterien, die jedes interaktive System erfüllen sollte:

Effektivität: Genauigkeit und Vollständigkeit, mit der der Benutzer ein bestimmtes Ziel erreicht,

Effizienz: Aufwand des Benutzers im Verhältnis zur Genauigkeit und Vollständigkeit des zu erzielenden Effekts sowie

Zufriedenheit: positive Einstellung des Benutzers gegenüber der Nutzung des Systems.

Als Maß dieser Kriterien wird die Gebrauchstauglichkeit (engl. *usability*) abgeleitet. Teil 210 der Norm[2] definiert den Prozess der nutzergerechten Gestaltung von interaktiven Systemen als iteratives Vorgehen, bestehend aus den allgemeinen Teilschritten der Anforderungsanalyse, Lösungserstellung und Evaluation. Eine Ergänzung der Norm für die Anforderungen von multimedialer Gestaltung nimmt DIN EN ISO 14915-2[3] vor. In Abgrenzung zu den vorrangig adressierten Büroarbeitsplätzen in DIN 9241 werden weiterführende Kontexte zum allgemeinen Einsatz multimedialer Systeme betrachtet.
DIN EN ISO 6385[4] beschreibt ergonomische Grundsätze zur Gestaltung von Arbeitssystemen. In Arbeitssystemen wirken menschliche Arbeiter zusammen mit Maschinen oder Werkzeugen. Die Norm umfasst allgemeine Aspekte der Gestaltung von Arbeitsplätzen und der Organisation der Arbeit. Die VDI/VDE Richtlinie 3850 konkretisiert die Gestaltung von Maschinen-Bediensystemen. Dies ist notwendig, da die Steuerung von Maschinen, speziell in der Produktionstechnik, in der Regel höheren Komplexitätsanforderungen als Bürosoftware unterliegt. Zudem sind Ausgabe- und Eingabegeräte hier vielfältiger und weniger standardisiert. Die Grundlagen zur Gestaltung in Blatt 1[5] ergänzen die DIN 9241. Weiter werden in Blatt 2[6] Regeln und Empfehlungen zur Auswahl koordinatengebender und nicht-koordinatengebender Interaktionsgeräte bei grafischen Bediensystemen im stationären Einsatz gegeben. Blatt 3[7] beschreibt Regeln und Empfehlungen zum Einsatz von Touchscreens an Maschinen. Eine gesonderte Betrachtung von mobilen Geräten wie Handheld-Geräten im Hinblick auf die Maschinenbedienung ist aktuell noch nicht in der Richtlinie enthalten. Diesbezüglich besteht zurzeit die Motivation zu einer Überarbeitung und Ergänzung der Richtlinie. Im Folgenden werden relevante Grundsätze der ergonomischen Gestaltung von Dialogführung mittels direkter Manipulation auf Basis von DIN EN ISO 9241-16[8] näher erläutert.

[1] Vgl. DIN EN ISO 9241-11:1999-01 [16].
[2] Vgl. DIN EN ISO 9241-210:2011-08 [17].
[3] Vgl. DIN EN ISO 14915-2:2003-11 [18].
[4] Vgl. DIN EN ISO 6385:2004-05 [19].
[5] Vgl. VDI 3850-1:2012-08 [20].
[6] Vgl. VDI 3850-2:2012-11 [21].
[7] Vgl. VDI 3850-3:2004-03 [22].
[8] Vgl. DIN EN ISO 9241-16:2000-03 [23].

A) Metaphern

Beim Erlernen von Funktionen sollen nach Möglichkeit kognitive Prozesse angesprochen werden, die neue Informationen mit bereits bekanntem Wissen verknüpfen. Je besser dies gelingt, desto mehr festigt sich das neue Wissen und desto schneller werden diese Informationen dem Langzeitgedächtnis dauerhaft hinzugefügt. Bei der Gestaltung von Nutzerschnittstellen dienen Metaphern dazu, durch entsprechende bildliche Darstellung Assoziationen über mögliche Funktionen eines Arbeitsobjekts zu erzeugen.

B) Direktheit der Interaktion

Aufbauend auf dem Visualisierungsprinzip der Metaphern bezeichnet die direkte Manipulation eine Interaktionstechnik, bei der der Benutzer den Eindruck erhält, die Objekte auf dem Anzeigegerät direkt zu bearbeiten, bspw. indem er mit Hilfe eines Zeigeinstruments auf sie zeigt, sie verschiebt und/oder ihre physikalischen Eigenschaften verändert [13]. Direkte Manipulation unterscheidet sich von indirekter Manipulation dadurch, dass eine physische Interaktion mit einer visuellen Objektrepräsentation der Informationen stattfindet. Dazu müssen keine speziellen Befehle erlernt werden. Die Manipulation der Objekte ist für den Anwender prinzipiell logisch, schnell zu erlernen und intuitiv durchführbar, da sie sich in Anlehnung an reale physische Interaktion gestalten lässt. Je mehr die durchgeführte physische Interaktion der logischen Funktion entspricht, desto selbstverständlicher erscheint die Interaktion für den Nutzer. Dieser Grad der Direktheit der Manipulation kann durch entsprechende Gestaltung der Ein- und Ausgabe unterstützt werden. Durch direkte Manipulation lassen sich zudem auch Fehlbedienungen durch die Minimierung von Verwechslungen vermeiden. Das Prinzip der direkten Manipulation gilt bei der Gestaltung grafischer Benutzerschnittstellen als Grundsatz. Die gängige Umsetzung erfolgt in Form von Fenstern, Symbolen, Menüs und Zeigern (engl. Akronym: WIMP). Das What-You-See-Is-What-You-Get-Prinzip (WYSIWYG-Prinzip) beschreibt weiterführend eine menschengerechte Gestaltung der Darstellung von Anzeige- und Manipulationsprozessen von Objekten. Diese soll sich an der Ausgabe bzw. der realen Verkörperung der Informationen, bspw. der Anzeige eine Blattes mit Text in Textverarbeitungsprogrammen, orientieren. Dies umfasst auch die Darstellung, die Manipulation und das Testen von 3D-Objekten in virtuellen Umgebungen unter Einsatz moderner Computer-aided-Design-Anwendungen (CAD-Anwendungen).

C) Feedback

Rückmeldungen (engl. *feedback*) bezeichnen Ausgaben des maschinellen Systems, welche direkt aus Nutzereingaben resultieren oder Ausgaben zu Statusinformationen des Systems darstellen [24]. Die allgemeine Motivation zur Nutzung von Rückmeldungen ist eine Vereinfachung der Prozesse durch adäquate Informationsbereitstellung. Im Speziellen werden eine Bestätigung von durchgeführten Aktionen, eine beabsichtigte Änderung des Nutzerverhaltens sowie eine allgemeine Förderung des Verständnisses des Systems durch die Bereitstellung von Feedbackinformationen bezweckt.
Bei der Ausgestaltung von Feedbackfunktionen sind verschiedene Ebenen der Rückgabe bezogen auf Bedeutung, Darstellung und Ereignis zu betrachten. Der Bedeutungsraum beschreibt die

Intention hinter dem bereitgestellten Feedback. Der Darstellungsraum erläutert die Repräsentation, bspw. auf dem Bildschirm, und der Ereignisraum beschreibt Programmabläufe und Benutzerinteraktionen. Neben taktilen und akustischen Rückmeldungen werden in modernen Mensch-Maschine-Systemen primär visuelle Rückmeldungen verwendet [24, 25]. Bezogen auf die direkte Manipulation von Informationen ist das Resultat der Manipulation sofort zu sehen. Neben der Selektion von Objekten wird in der Regel schon während der Interaktion eine visuelle Rückmeldung zu deren Status gegeben. Das Feedback lässt sich somit zum Bestandteil des aktiven Interaktionsprozesses erweitern [13].

2.1.3 Natürliche Gestaltung von Benutzerschnittstellen

Vorbild für die Gestaltung moderner Mensch-Maschine-Kommunikation ist die natürliche Mensch-Mensch-Kommunikation, welche zum größten Teil multimodal über Sprache (auditiv) und über Gesten (visuell oder haptisch) stattfindet. Ziel der Nutzung von natürlichen Kommunikationskanälen ist neben der Erhöhung der Akzeptanz des technischen Systems auch eine Verkürzung von Anlernzeiten und Bedienzyklen. Dabei werden aufbauend auf dem Paradigma der direkten Interaktion Handlungsprinzipien und kognitive Prozesse angesprochen, die die Interaktion mit realen Objekten möglichst nah auf die Interaktion mit virtuellen Objekten übertragen. Dabei ist die Verwendung von substituierbaren technischen Hilfsmitteln, bspw. dem Anlegen eines speziellen Handschuhs oder die Befestigung von farblichen Markern am Körper, zu vermeiden, da sich die künstlichen Hilfsmittel negativ auf die Leitkriterien der Effektivität, Effizienz und Zufriedenheit auswirken [26, 27].
In diesem Zusammenhang hat sich in den letzten Jahren der Begriff des Natural User Interface (NUI) in Abgrenzung zum klassischen Graphical User Interface (GUI) herausgebildet [28, 29]. Werden bei GUIs für die Eingabe technische Geräte wie Tastatur und Maus benötigt, halten bei NUIs neue Interaktionsformen wie Spracheingaben, Touch- und Multitouch-Bedienung sowie räumliche Interaktion in Virtual- und Augmented-Reality-Anwendungen Einzug. Im Folgenden werden das Prinzip der Multimodalität sowie die beiden angesprochenen Modalitäten Gesten und Augmented Reality näher beschrieben. Auf eine nähere Betrachtung von Sprachsteuerung soll im Rahmen dieser Arbeit aufgrund der begrenzten Nutzbarkeit im industriellen Umfeld hinsichtlich mangelnder Robustheit und individueller Übertragbarkeit aktueller Ansätze verzichtet werden. Sprachsteuerung bietet sich im Bereich der Robotik bisher lediglich für die teleoperative Robotersteuerung [30] und für die enge Zusammenarbeit von Mensch und Roboter in Laborumgebungen an [31].

A) Multimodalität

Multimodalität, d. h. die Nutzung mehrerer Kommunikationskanäle gleichzeitig oder hintereinander, spielt eine wichtige Rolle bei der Gestaltung natürlicher Steuerungssysteme. In zahlreichen wissenschaftlichen Veröffentlichungen, u. a. [15, 32, 33, 34, 35], wurde gezeigt, dass die multimodale Gestaltung industrieller Steuerungs- und Programmiersysteme die Produktivität und Akzeptanz der Nutzer steigert. Voraussetzung ist ein anwendungs- und anwenderspezifischer Entwurf

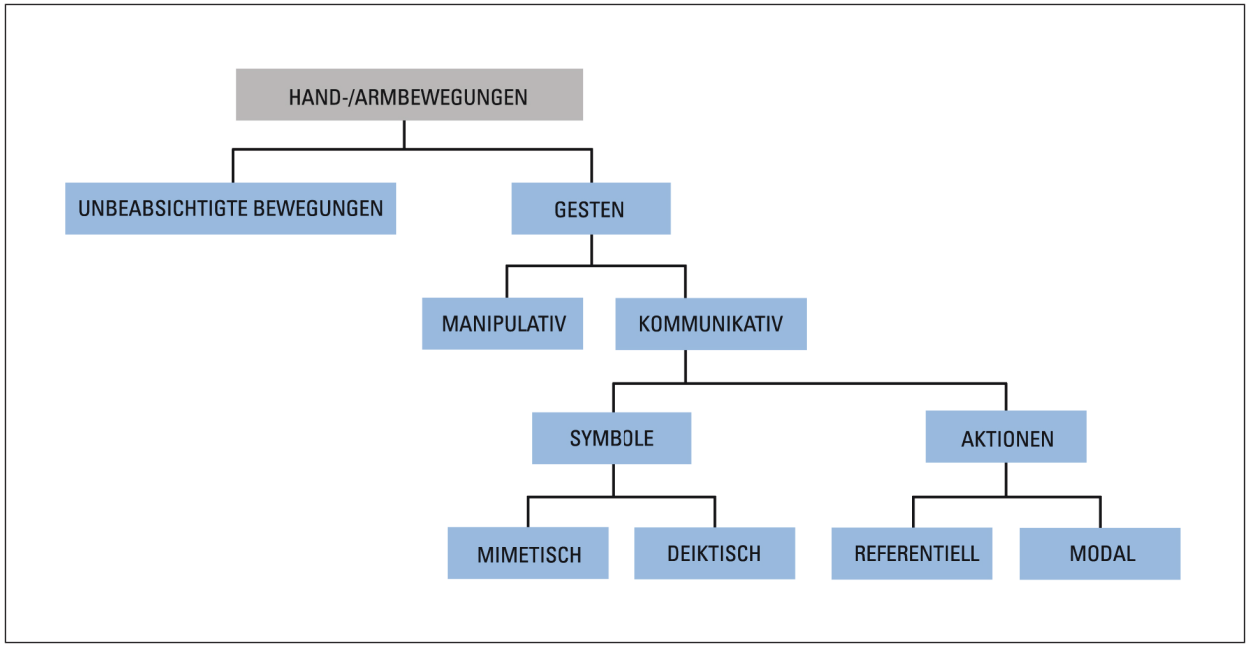

Abbildung 2.2: Taxonomie der Gesten nach [36].

der Systeme. Multimodale Steuerungssysteme kombinieren dabei bspw. Gesten (Finger-, Hand- und Touch-Gesten), Sprache sowie verschiedene Formen der Ausgabe, wie bspw. auch Augmented Reality.

B) Gestenbasierte Eingabe

Gestik ist neben Mimik und Haptik eine Form der nonverbalen menschlichen Kommunikation und kann in Kombination mit oder unabhängig von anderen Kommunikationsformen eingesetzt werden. In natürlicher Umgebung werden die kommunikativen Gesten typischerweise durch Sprache begleitet. Nach Pavlovic [36] beschreiben Gesten beabsichtigte Bewegungen, meist der Arme und Hände, mit manipulativem oder kommunikativem Charakter. Manipulative Gesten werden genutzt, um Objekte bzw. deren Eigenschaften zu beeinflussen, bspw. Translation, Rotation, Deformation. Kommunikative Gesten beinhalten einen übertragenden Zweck bzw. eine übertragende Absicht, d. h. Informationen, die auf verschiedenen Ebenen an einen Empfänger weitergeleitet werden. Der gestenbasierte Informationsaustausch kann auf pragmatischer Ebene (sprachliches Handeln), semantischer Ebene (Bedeutung der Zeichen) sowie syntaktischer Ebene (Zeichen und Regeln) betrachtet werden [13]. Ferner können kommunikative Gesten in Symbole oder Aktionen untergliedert werden. Die Symbole haben eine linguistische Funktion, die entweder referenziellen Charakter, bspw. durch das Andeuten eines Gegenstands, oder eine modale Funktion, bspw. zur symbolischen Erweiterung einer sprachlichen Aussage, besitzen. Als Handlungsgesten (Aktionen) können Gesten betrachtet werden, deren Intention eine direkte Verbindung zur Bewegung selbst hat. Mimetische Gesten fordern zur Imitation der angedeuteten Handlung auf, wobei deiktische Gesten (Zeigegesten) einen örtlichen Bezug zum Inhalt der Aktion darstellen. Abbildung 2.2 veranschaulicht die beschriebene Taxonomie der Gesten. Neben der hier gewählten Klassifizierung der Gesten nach Pavlovic, welche den MMI-Bereich fokussiert, bestehen weitere Ansätze der Glie-

derung und andere Bezeichnungen, u. a. nach [37]. Die Ansätze sind jeweils von der disziplinären Blickrichtung geprägt, betrachten jedoch übereinstimmend statische und dynamische Bewegungen, mit Schwerpunkt auf Bewegungen der Arme, Hände und einzelner Finger. Während statische Gesten auf die Betrachtung einer einzelnen Pose reduziert werden können, lassen sich dynamische Gesten über Trajektorien einzelner oder verschiedener Körperteile beschreiben. In der Praxis gibt es zudem hybride Formen.

Gestenbasierte Steuerung in Mensch-Maschine-Systemen
Im einfachsten Fall besteht eine Steuerungsgeste aus einer einzelnen Pose, bspw. dem Zeigen in eine bestimmte Richtung. Bei manipulativen Gesten kommt es hingegen zur direkten Interaktion mit realen oder virtuellen Objekten durch Bewegungen des Anwenders. Kommunikative Gesten sind demgegenüber abstrakt, können aber als Zeigegesten auch einen Zusammenhang mit der Umgebung besitzen. Zu den kommunikativen Gesten gehören jedoch auch die Kommandogesten, welche in Steuerungssystemen für das Auslösen einzelner Funktionen verantwortlich sind und oft Symbole darstellen. Generell gilt es beim Entwurf von Mensch-Maschine-Systemen, selbige gegenüber dem unbeabsichtigten Auslösen von Steuerungsbefehlen durch fälschlicherweise erkannte Kommandogesten robust zu gestalten. Komplexe Kommandogesten können einen inhaltlichen Zusammenhang zwischen Trajektorie (Symbol) und Steuerungsaufgabe haben; dies ist jedoch nicht zwangsläufig der Fall. Komplexe Gesten mit Zusammenhang zur Funktion sind intuitiver, einfacher zu erlernen und beim Entwurf eines gestenbasierten Steuerungssystems zu bevorzugen.

Gestenerkennung
Ein System zur Gestenerkennung umfasst zunächst einen Sensor zur Erfassung der Gesten. Die Erfassung kann über das Messen von elektrischen, optischen, akustischen, magnetischen oder mechanischen Größen erfolgen [38]. Je nach Messprinzip und Methode müssen eventuell Sensoren oder zusätzliche Hilfsmittel direkt am Körper des Nutzers angebracht oder getragen werden. Visuelle Messprinzipien setzen zur Unterstützung der Robustheit oftmals auf den Einsatz künstlicher Merkmale, bspw. farbigen Markierungen, welche am Körper befestigt werden. Die Ausprägung von unterschiedlichen kamerabasierten Techniken für die Erkennung von Handgesten wird umfassend in [39] vorgestellt. Die Weiterverarbeitung der sensoriell erfassten Daten beinhaltet Algorithmen zur Segmentierung und zum Tracking sowie zur Merkmalsextraktion. Durch Methoden der Mustererkennung erfolgt schließlich die Detektion einer bekannten Geste. Zur Mustererkennung bei unsicheren Informationen, bspw. durch Sensorrauschen, werden hauptsächlich stochastische Verfahren wie das Dynamic-Time-Warping oder Hidden-Markov-Modelle eingesetzt. Abschließend können die erkannten Gesten in Steuerungskommandos übersetzt werden. Abbildung 2.3 verdeutlicht den beschriebenen Ablauf der kamerabasierten Gestenerkennung. Es empfiehlt sich in der Praxis, eine Feedbackfunktion zur Information des Anwenders über das erfolgreiche Erkennen einer Geste bzw. das Auslösen einer Steuerungsfunktion vorzusehen.

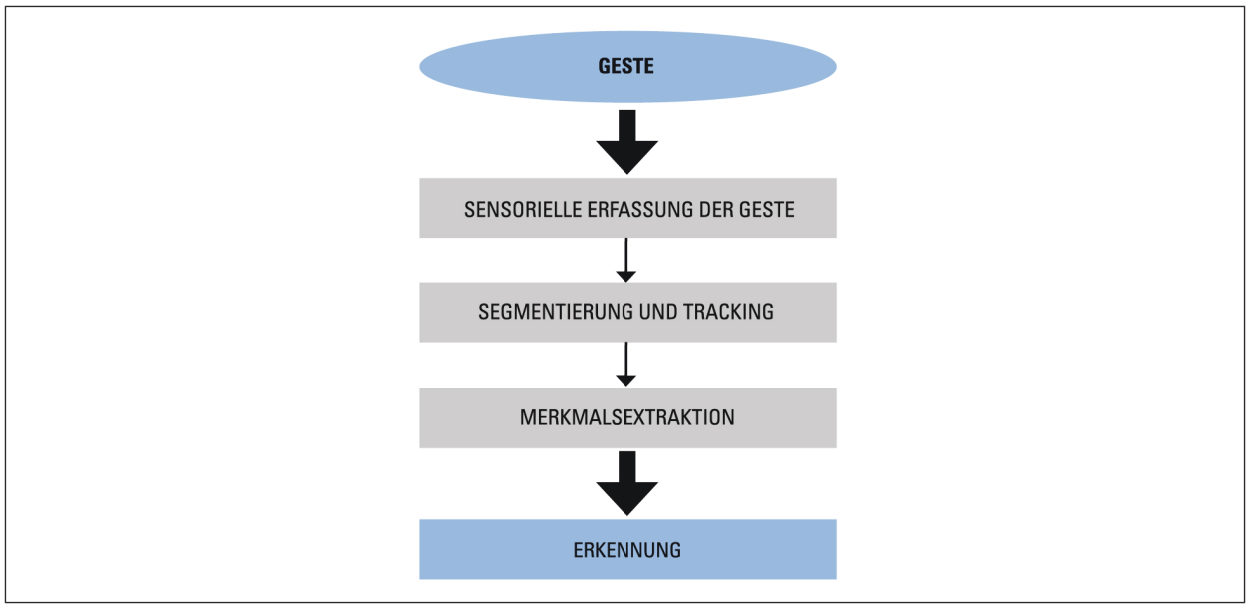

Abbildung 2.3: Grundlegender Aufbau eines kamerabasierten Systems zur Gestenerkennung in Anlehnung an [38].

C) Visualisierung in Augmented Reality

Definition und Abgrenzung

Augmented Reality oder dt. Erweiterte Realität bezeichnet ein Interaktionsparadigma, welches vorrangig zur Visualisierung von räumlichen Daten genutzt wird. Die gebräuchliche Definition eines AR-Systems geht zurück auf R. Azuma [40, 41] und beinhaltet folgende zwingende Merkmale:

- die Kombination von realen und virtuellen Objekten in realer Umgebung,
- die Anpassung virtueller Objekte an die reale Umgebung sowie
- die Interaktion in 3D und Echtzeit.

Nach dieser Definition besteht weder eine Restriktion auf eine bestimmte Technologie, bspw. Displaytechnologie, noch auf die genutzte Interaktionsmodalität, bspw. Visualisierung. Theoretisch lassen sich mit der AR neben der visuellen Wahrnehmung weitere Sinne wie das Hören, Fühlen und Schmecken adressieren [42]. Anders als bei der Virtual Reality (VR) geht es bei der AR nicht um eine möglichst perfekte und aufwendige Nachbildung realer Umgebungen, sondern um eine sinnvolle Ergänzung der Realität zur Unterstützung des Nutzers durch lokal platzierte virtuelle Objekte. Reale Objekte können demgemäß auch versteckt werden, indem sie durch virtuelle Objekte überdeckt werden.

Die AR ist zusammen mit der Augmented Virtuality den Verfahren der Mixed Reality zuzuordnen. Abbildung 2.4 verdeutlicht die Einordnung der AR im sogenannten „Realitäts/Virtualitäts-Kontinuum“, welches zur Beschreibung des Bereichs zwischen 100 % Realität und 100 % Virtualität herangezogen wird [43].

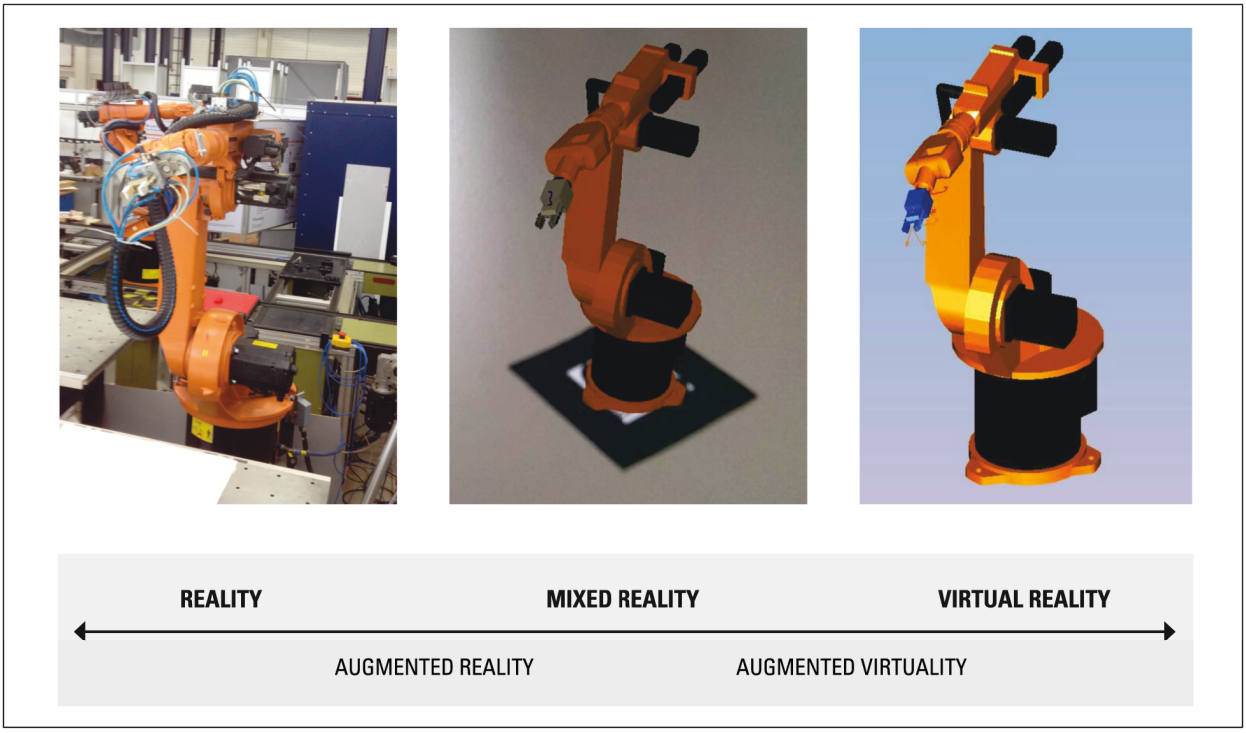

Abbildung 2.4: Realitäts/Virtualitäts-Kontinuum in Anlehnung an [43, 44] am Beispiel einer Industrieroboterdarstellung.

Displaytechnologie

Wie aus der Definition der AR hervorgeht, erweitern AR-Systeme die reale Sinneswahrnehmung durch das künstliche Einbringen virtueller Objekte. Es bestehen neben visuellen Displaysystemen auch Ansätze der AR für olfaktorische, haptische, gustatorische und auditive Wahrnehmung. Die folgenden Ausführungen beschränken sich jedoch auf die visuellen Systeme. Es existieren drei Grundprinzipien zur visuellen Erzeugung von AR.

Bei *Video-See-Through-Systemen* (VST-Systemen) wird ein optisch undurchlässiges Display verwendet. Auf der Oberfläche des Displays werden sowohl virtuelle Objekte als auch ein Kamerabild dargestellt. Diese Methodik kann in der Regel günstig und einfach umgesetzt werden. Zudem kann das Kamerabild weiterführend verarbeitet werden, bspw. zur Realisierung von Tracking-Anwendungen zur Lokalisation oder Interaktion. Der vorrangige Nachteil dieser Methodik liegt in der Limitierung des Darstellungsbereichs durch die Auflösung der Kamera bzw. des Displays sowie durch den Sichtbereich der Kamera.

Optical-See-Through-Systeme (OST-Systeme) nutzen transparente oder halbtransparente Displays. Lediglich die virtuellen Informationen werden auf der Oberfläche dargestellt. Zur Lokalisierung und Interaktionserkennung ist bei diesem Prinzip aufgrund der fehlenden Kamera zusätzliche Sensorik einzubinden.

Des Weiteren existieren *Projektionssysteme*, bei denen Projektoren die virtuellen Einblendungen direkt auf reale Objekte im Raum, meist ebene Oberflächen, projizieren. Neben einer notwendigen Kalibrierung des Projektors müssen auch hier zur dynamischen Lokalisierung und zur Interaktionserkennung zusätzliche Sensoren eingebunden werden. Die Darstellungsfläche ist zudem räumlich auf die Oberflächen der Objekte begrenzt.

Gerätetechnische Umsetzung der Displaytechnologie
Eine weitere Unterscheidung der AR-Systeme kann auf Basis der Positionierung der Anzeige zwischen Benutzer und realer Umgebung vorgenommen werden. Am Kopf getragene (engl. *head-worn*) Displays umfassen Head-Mounted Displays (HMD), Virtual Retinal Displays (VRD) sowie Head-Mounted Projective Displays (HMPD). HMDs existieren sowohl in OST- als auch in VST-Ausführung. Zusammen mit den Handheld-Geräten bilden die am Kopf getragenen Displays die mobile Displaytechnologie zur Realisierung der AR. Zu den Handheld-Geräten zählen mobile See-Through sowie projektive Systeme, welche mit der Hand positioniert werden. Dazu gehören bspw. Smartphones oder mobile Projektoren. Jüngste Untersuchungen zeigen, dass das dauerhafte Tragen moderner Handheld-Geräte im Gegensatz zu HMDs keine körperliche Belastung mehr darstellt [42]. Durch das Halten des Geräts besteht jedoch eine Einschränkung der händischen Interaktion. Räumlich fest platzierte Displays können in der Regel nicht an den individuellen Sichtbereich des Nutzers angepasst werden und werden aus diesem Grund zumeist in statischen Anwendungen eingesetzt. Diese umfassen monitorbasierte VST-Systeme, OST-Systeme, wie transparente räumliche Displays, sowie stationäre projektive Systeme.

Registrierung
Zur Registrierung von virtuellen Objekten in der realen Welt, d. h. zur perspektivegerechten Darstellung, wird eine Lokalisierung des Displays in einem Referenzkoordinatensystem (Referenz-KS) benötigt. Die Lokalisierung umfasst die Pose: Position und Orientierung in insgesamt sechs Bewegungsfreiheitsgraden (engl. *degrees of freedom* (DoF)). Bei stationären AR-Anwendungen lässt sich diese Transformation anhand von einmalig anzuwendenden Mess- bzw. Kalibriermethoden bestimmen. Mobile AR-Anwendungen sind auf eine kontinuierliche Lokalisierung in Form einer Verfolgung (engl. *tracking*) angewiesen. In Außenumgebungen wird zur Bestimmung der Position oftmals das Global Positioning System (GPS) eingesetzt. Zur Bestimmung der Orientierung werden teilweise hybride Ansätze durch Sensorfusion von visuellem Tracking und Inertialsensorik, also Beschleunigungssensoren und Gyroskop, gewählt. Beschleunigungssensoren sind zur alleinigen Positionsbestimmung in der Regel ungeeignet, da sich durch Integration des Tracking-Fehlers eine Positionsdrift einstellt [45].
Die meisten mobilen Innenraum-AR-Anwendungen nutzen optische Verfahren zur Posenbestimmung. Eine umfangreiche Darstellung verschiedener visueller Tracking-Verfahren für AR ist [46] zu entnehmen. Zur dynamischen Ermittlung der Pose kann grob in das Tracking von aktiven oder passiven Markern unterschieden werden. Eines der verbreitetsten optischen Verfahren ist das Tracking eines raumfesten, künstlichen Markers auf Basis des 2D-Bilds einer Kamera. Bei dem Marker handelt es sich in der Regel um ein passives ebenes Objekt mit bekannter Form und bekannten Abmaßen. Über die 2D/3D-Korrespondenz der Punkte wird auf Basis der Kameraparameter der Abstand vom Marker zur Kamera geschätzt [47]. Die Robustheit des optischen Marker-Tracking ist prinzipiell abhängig von der Beleuchtungssituation. Die Genauigkeit der Darstellung ist weiterführend abhängig vom eingesetzten Tracking-Algorithmus, von der Auflösung der Kamera sowie von der Entfernung und dem Winkel zwischen Kamera und Marker. Durch ein gleichzeitiges Tracking mehrerer Marker lässt sich eine Genauigkeitssteigerung sowie eine Erhöhung der Robustheit erreichen [48]. Weitere optische Tracking-Verfahren adressieren statt der passiven raumfesten Marker aktive Marker, bspw. in Form von ligh emitting diodes (LEDs) [46]. In jüngs-

ter Zeit bestehen zunehmend Ansätze zum markerlosen optischen Tracking anhand von inhärenten Merkmalen der Umgebung oder des Anzeigegeräts. Pilet et al. [49] zeigen bspw. einen Ansatz, heterogene, biegsame Oberflächen zur Posenbestimmung mit 2D-Kameras zu nutzen. Nachteil der markerlosen Ansätze ist, dass eine Übertragung auf beliebige Umgebungen oder Anzeigegeräte meist nur bedingt gegeben ist.
Anstatt das Tracking durch das Mobilgerät vorzunehmen, kann externe raumfeste Sensorik zur Verfolgung des Mobilgeräts eingesetzt werden. Anhand von 2,5D-Sensorik können zusätzlich Objekte erkannt und in die AR-Anwendung übertragen werden [50]. Eine punktwolkenbasierte Rekonstruktion von Oberflächen ermöglicht weiterführend die zunehmende Verschmelzung von AR und Realität. Komplexe Interaktionen von virtuellen und realen Objekten lassen sich derart darstellen [51].

Software-Frameworks
Mittlerweile existieren zahlreiche kommerzielle und nicht-kommerzielle Software-Frameworks für VST-basierte AR-Anwendungen, die in der Regel für statische und mobile Systeme, bspw. Android und iOS, verfügbar sind. Neben der Aufnahme eines Kamerabilds und des Tracking gehören das Rendern des überlagerten Kamerabilds, die Kamerakalibrierung sowie der Import von gängigen CAD-Formaten zum allgeminen Funktionsumfang der Frameworks (s. Abb. 2.5). Eines der ersten frei verfügbaren AR-Frameworks war im Jahr 2004 das ARToolkit[9] der Universität Washington. Trotz mittlerweile veralteten Algorithmen zum Marker-Tracking ist dieses Framework zur Erstellung von Prototypen im Bereich der Forschung und Entwicklung noch immer sehr beliebt. Weiterentwicklungen des Framework (ARToolKitPlus[10]) umfassen schnellere und robustere Algorithmen zum Marker-Tracking sowie einen erweiterten Funktionsumfang.

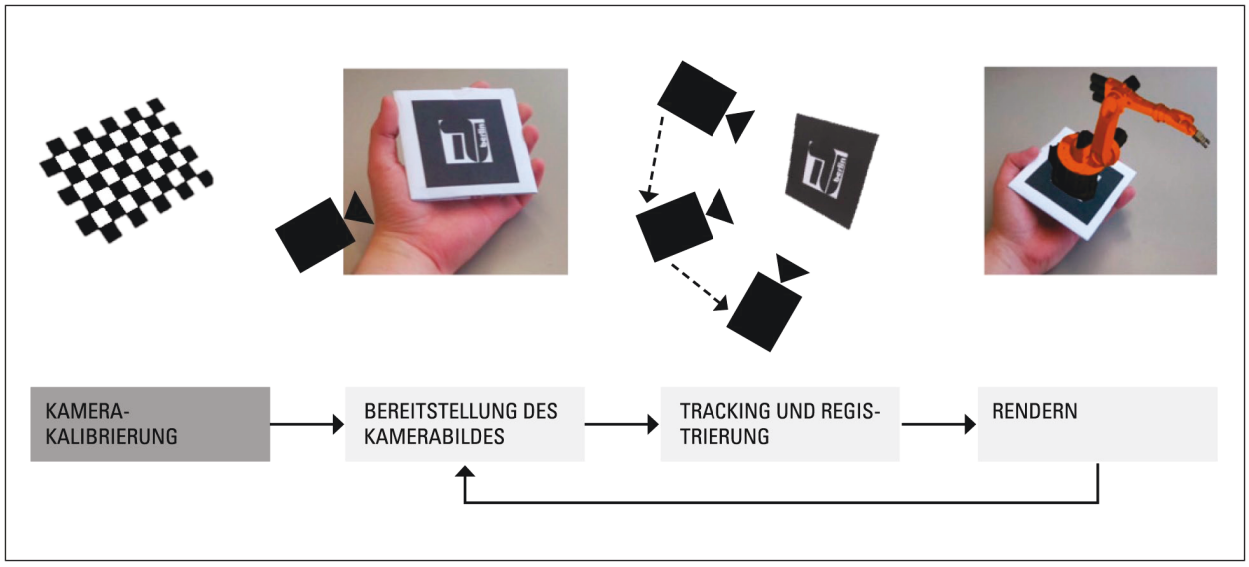

Abbildung 2.5: Hauptfunktionen von AR-Software-Frameworks basierend auf der VST-Displaytechnologie.

[9] Vgl. http://www.hitl.washington.edu/artoolkit/ - 20.01.2014.
[10] Vgl. http://handheldar.icg.tugraz.at/artoolkitplus.php - 20.01.2014.

Koordinatengebende Eingabe in der Augmented Reality
AR verbindet die reale Umgebung und virtuelle Objekte. Erweiterte Benutzerschnittstellen auf Basis dieser Technologie unter Verwendung von räumlicher Eingabe müssen prinzipiell Interaktionsmethoden für reale und virtuelle Objekte bereitstellen. Für die räumliche Interaktion greifen aber weder herkömmliche Interaktionsparadigmen, noch sind klassische Eingabegeräte wie Maus oder Tastatur für die Interaktion in sechs Freiheitsgraden geeignet [13]. Trotz der unzureichenden Übertragbarkeit des WIMP-Paradigmas auf die Interaktion werden Methoden zur räumlichen Anwahl, Positionierung und Rotation in Form von direkter Manipulation von Objekten benötigt. Neben der Interaktion mit Objekten eröffnet eine koordinatengebende Eingabe die Möglichkeit, Pfade und Trajektorien durch natürliche Bewegungen zu nutzen. Erfolgt die Manipulation teleoperativ an einem Computersystem, werden alternativ zur räumlich-gestenbasierten Interaktion zumeist 3D-Mäuse oder 6DoF-Phantomgeräte in Kombination mit den klassischen Eingabegeräten Maus und Tastatur verwendet. Der natürlichste Weg zur räumlichen Interaktion mit realen oder virtuellen Objekten führt dabei über Gesten, welche frei im Raum durchgeführt werden. Zur Detektion der Gesten können grundlegend Inertialsensorik, Datenhandschuhe oder optische Geräte, wie Kameras, verwendet werden [52].

2.1.4 Innovative Benutzerschnittstellen

A) Touch-Gesten

Touchscreens sind zugleich Anzeige- sowie Eingabegeräte. Eine direkte Interaktion mit Objekten auf dem Anzeigeelement wird zumeist über Fingergesten ohne Zuhilfenahme eines künstlichen Zeigeinstruments ermöglicht. Liegt bei der indirekten Steuerung des Cursors über die Maus noch ein erhöhter Aufwand für die Hand-Auge-Koordination vor, kann durch direktes Anwählen der Objekte eine intuitive Interaktion realisiert werden. Durch die wahlweise Nutzung von Eingabestiften bei passender Displaytechnologie kann eine präzisierte Anwahl von Objekten erfolgen. Touchscreens werden als Standard in modernen Handheld-Geräten eingesetzt. Als Multitouch werden Eingaben bezeichnet, welche mit zwei oder mehr Berührungspunkten zur gleichen Zeit durchgeführt werden. Klassischerweise werden durch Multitouch räumliche Funktionen wie Rotieren, Zoomen oder kooperative Eingaben mehrerer Nutzer realisiert. Die Berührungsempfindlichkeit der Displays lässt sich durch resistive, kapazitive und induktive Funktionsprinzipien erreichen. Vereinzelt werden für Touchscreens auch Ultraschall oder kamerabasierte Technologien eingesetzt. Für die Bedienung von Maschinen existieren zumeist spezielle stationäre Ausführungen von Touchscreens mit den entsprechenden industriellen Schutzklassen nach DIN EN 60529[11].

B) Taktile Benutzerschnittstellen

Taktile bzw. anfassbare (engl. *tangible*) Benutzerschnittstellen erlauben die haptische Interaktion mit physischen Objekten. Die Translation von physischen Objekten durch den Anwender dient der koordinatengebenden Eingabe. Um die praktische Umsetzung zu erleichtern, handelt es sich

[11] Vgl. DIN EN 60529:2000-09 [53].

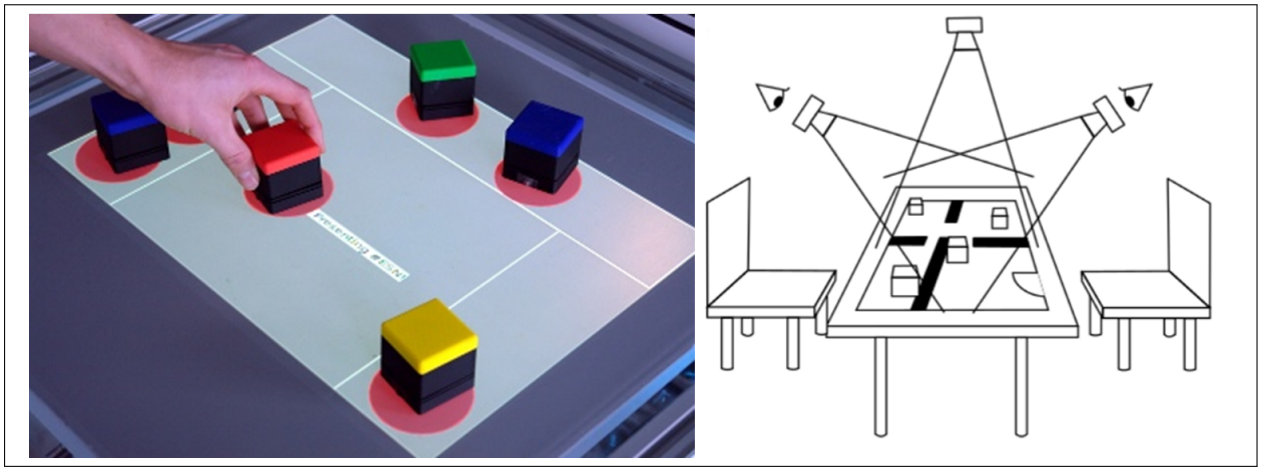

Abbildung 2.6: Tangible User Interface unter Verwendung einer Displayebene und Interaktionsobjekten in Form von mobilen Plattformen (links), Sichtbereiche der Nutzer und Sensorik (rechts), Quelle: [54].

bei den Objekten in den meisten Fällen um geometrische Primitive, welche in einer Ebene bewegt werden. Die Ebene kann in Ausführung eines Displays eine einfache visuelle Anzeigefläche (s. Abb. 2.6) oder als Projektionsfläche für AR-Anwendungen erweiternde Ausgabe- und Feedbackfunktionen bereitstellen. Im Rahmen einer AR-basierten Rückgabe können die physischen Objekte weiterführend mit Markern zur Lokalisation ausgestattet werden, um ihr Aussehen und ihre Maße virtuell zu verändern, während die Möglichkeit der physischen Interaktion zur Eingabe erhalten bleibt [54]. Somit wird ein Zusammenhang der Interaktion von physischem Objekt und seiner virtuellen Darstellung hergestellt. Diese Art der AR-Anwendungen lässt sich anhand von Projektoren, HMDs oder Handheld-Geräten realisieren. Fischer et al. [55] präsentieren eine anfassbare Benutzerschnittstelle zur interaktiven Stadtplanung anhand realer und virtueller Modelle von Gebäuden (s. Abb. 2.7). Die Objekte lassen sich per Gesten frei in einer virtuellen Umgebung positionieren. Die Darstellung sowie Lokalisation der virtuellen Objekte erfolgt anhand einer markerbasierten AR-Anwendung. Schmalstieg et al. [56] entwickeln eine AR-Anwendung, welche sowohl mehrere Nutzer zur gleichen Zeit als auch verschiedene Anzeigegeräte unterstützt. Der Anwender wird in die Lage versetzt, mit einem virtuellen Anzeigegerät zu interagieren, welches 2D- und 3D-Eingaben mit einem Eingabestift zulässt. Das Anzeigegerät verfügt über eine vereinfachte physische Repräsentation und ermöglicht somit haptisches Feedback beim Halten und bei der 2D-Eingabe. Anfassbare Benutzerschnittstellen, welche zusätzlich eine haptische Ausgabe bzw. Rückgabe in Form bidirektionaler Kommunikation unterstützen, werden auch als haptische Benutzerschnittstellen (engl. *haptic user interfaces*) bezeichnet. Die räumliche Interaktion mittels haptischer Interaktion wird oftmals durch Teleoperation und Führung eines 6DoF-Phantomgeräts ausgeführt.

C) Räumliche Benutzerschnittstellen

Virtual Reality

Die Cave Automatic Virtual Environment (CAVE) bezeichnet ein räumliches System der virtuellen Realität. Der Nutzer begibt sich in einen Raum, dessen Wände als Projektionsflächen für eine

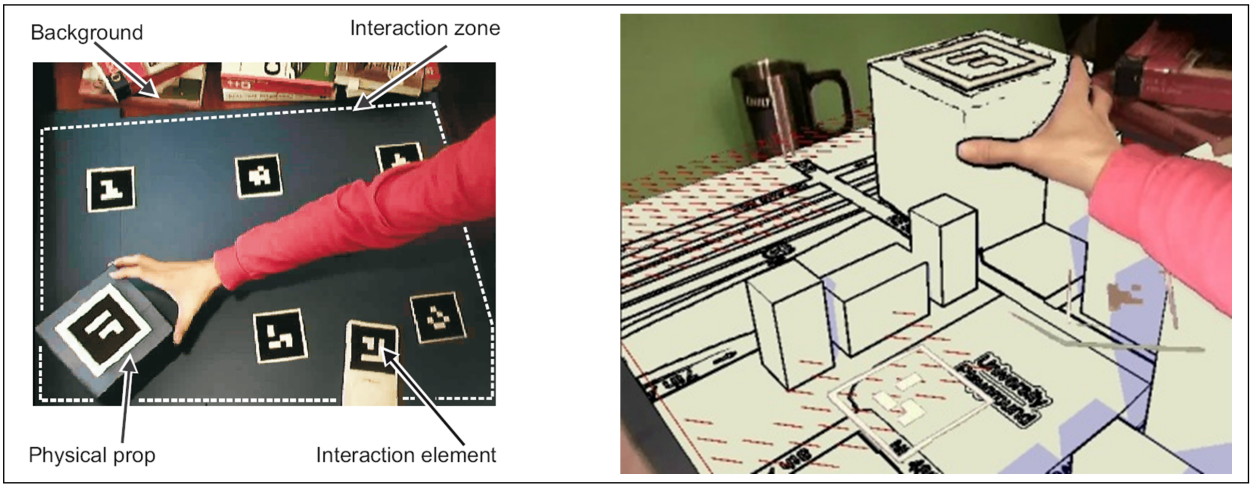

Abbildung 2.7: AR-basierte Interaktion in anfassbaren Benutzerschnittstellen: Marker und Aufbau (links) und gerenderte Szene zur interaktiven Stadtplanung (rechts), Quelle: [55].

virtuelle Umgebung dienen. Die Projektion lässt sich zu einer stereoskopischen 3D-Darstellung erweitern. Ermöglicht werden somit eine direkte räumliche Interaktion mit virtuellen Objekten und eine physische Bewegung in der Umgebung. Unterstützt wird die Wahrnehmung und Interaktion je nach Ausführung durch zusätzliche Hardware, wie bspw. Zeigegeräte oder Datenhandschuhe, sowie zusätzliche Feedbackfunktionen. Im Vergleich zu Desktopanwendungen der virtuellen Realität wird ein höherer Grad der Immersion erreicht [57]. Das bedeutet, der Nutzer erreicht eine realitätsnähere Wahrnehmung der eigenen Person in der virtuellen Umgebung. Dies führt zu einer besseren räumlichen Vorstellungsfähigkeit und erhöhter Akzeptanz des technischen Systems. Industrielle Anwendungsbereiche der CAVE sind neben Trainingsszenarien vor allem die Anlagen- und Fabrikplanung sowie die Konstruktion und Telepräsenz. Abbildung 2.8 zeigt eine CAVE-Anwendung zur gestenbasierten Programmierung eines Industrieroboters. Nachteile der

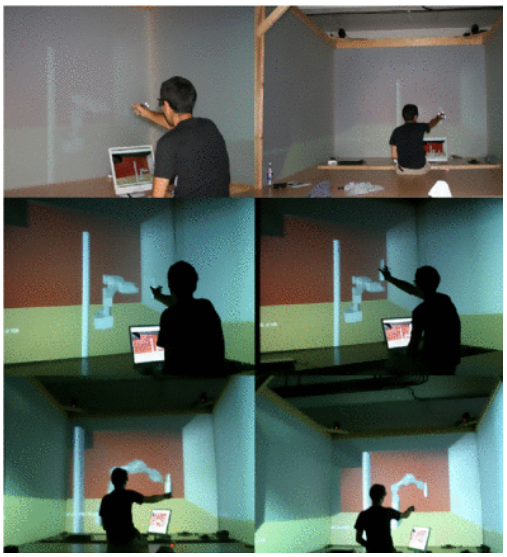

Abbildung 2.8: Räumliche Interaktion zur Industrieroboterprogrammierung in der CAVE, Quelle: [57].

CAVE liegen in der mangelnden mobilen Einsetzbarkeit der räumlichen Interaktionsverfahren, in den hohen Kosten und in der Komplexität des Systems. Ein ähnliches System ist die sogenannte „Powerwall“. Dabei handelt es sich lediglich um eine einzelne Projektionswand, welche sich auch als stereoskopische 3D-Darstellung realisieren lässt. Diese Ausführung ist weniger kostenintensiv als die CAVE, bietet allerdings einen geringeren Immersionsgrad. Eine weitere Möglichkeit zur Realisierung von immersiver virtueller Realität bieten HMDs.

Augmented Reality

Die räumliche Bereitstellung von Informationen zur Unterstützung des Nutzers findet sich bereits in zahlreichen Umsetzungen von AR-Systemen mit Anwendungsbereichen in der Überwachung und Instandhaltung, Montage, Kommissionierung sowie in der computergestützten Chirurgie [40, 58, 59]. AR-Anwendungen ermöglichen jedoch nicht nur die Anzeige von virtuellen Objekten in der Realität, sondern bilden auch die Grundlage für eine freie räumliche Interaktion mit realen und virtuellen Objekten. Für diese räumliche Interaktion lassen sich verschiedene Eingabegeräte nutzen. Shen et al. [60] präsentieren verschiedene markerlose, gestenbasierte Interaktionstechniken zur Manipulation von virtuellen Objekten (s. Abb. 2.9). Weiterführend wird diese Art der natürlichen Interaktion in einer statischen AR-Umgebung zur Definition einer Montageaufgabe durch räumliche Manipulation von virtuellen Objekten genutzt [27]. Neben der komplexen 3D-Interaktion lassen sich virtuelle Eingabegeräte in Form von 2D-Projektionen, bspw. virtuellen Tastaturen, realisieren.

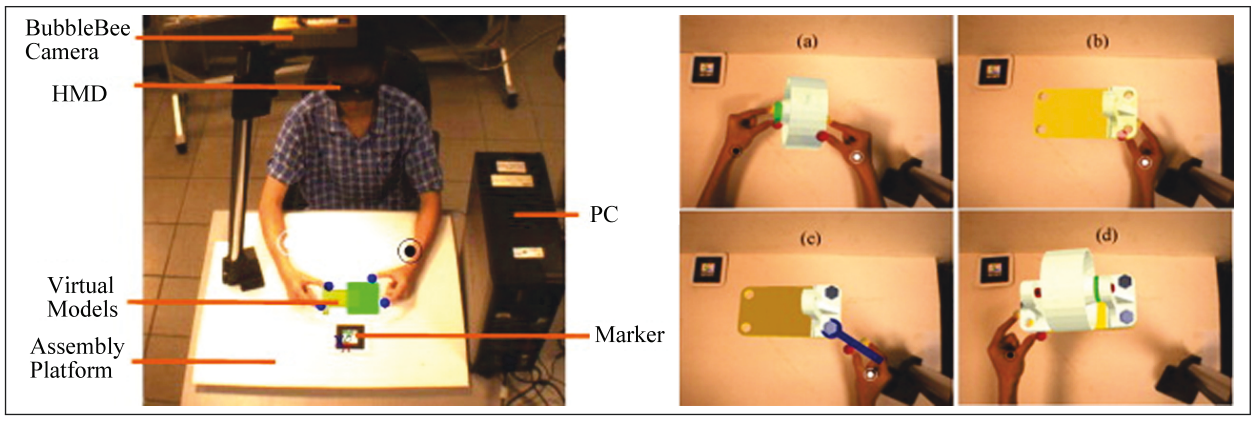

Abbildung 2.9: Automatisierte Generierung von Montageplänen durch gestenbasierte Interaktion mit virtuellen Objekten in einer AR-Umgebung (Schritte a-d), Quelle: [27].

2.2 Industrieroboterprogrammierung

2.2.1 Das Programmiersystem

A) Das Steuerungssystem als Grundlage der Programmierung

Ein Robotersystem besteht grundlegend aus Kinematik (Roboterarm), Steuerungssystem, Effektor sowie Peripherieeinheiten wie Sensoren, Werkzeugwechselsystemen und Schutzeinrichtungen. Ein wichtiges Abgrenzungsmerkmal des Industrieroboters ist nach VDI-Richtlinie 2860[12] die freie Programmierbarkeit der Achsen hinsichtlich Bewegungsfolge und Wegen bzw. Winkeln.
Beim Steuerungssystem handelt es sich um einen Verbund von Hard- und Softwarekomponenten. Industrierobotersteuerungen werden weitgehend auf Basis von Industrie-PCs, zum Teil in Mehrkernsystemen, unter Einsatz von Echtzeitbetriebssystemen realisiert. Vereinfacht betrachtet gibt die Steuerung die Bewegungen und Aktionen an den Roboter vor, die über die Programmierung festgelegt wurden, und kontrolliert deren Ausführung. Dazu verfügt sie über entsprechende Schnittstellen zum Bediener (Benutzerschnittstelle), zum Roboter (Servoregelung) sowie zum Prozess (externe Kommunikationsschnittstellen). Die Hauptsoftwarekomponenten der Steuerung sind die Bewegungssteuerung, die Ablauflaufsteuerung und die Aktionssteuerung. Weitere Komponenten und deren Abhängigkeiten werden in Abbildung 2.10 veranschaulicht. Das Programmiersystem stellt eine weitere Teilfunktion der Steuerung dar [62].
Die Ablaufsteuerung realisiert die sequentielle Abarbeitung des Programms durch Interpretation der textuellen Befehlssätze. Im Rahmen der Ablaufsteuerung werden die interpretierten Befehle an Funktionen der Bewegungs- oder Aktionssteuerung weitergegeben. Die Bewegungssteuerung erzeugt auf Basis der Bewegungsanweisungen im Programm Führungsgrößen für die Servoregler der Roboterachsen. Dazu werden zunächst in Abhängigkeit der vorgegebenen Interpolationsart Bahnstützpunkte durch den Interpolator erzeugt. Durch die inverse Kinematik werden für die einzelnen Bahnstützpunkte Sollwerte für die Einzelachsen berechnet. Die Feininterpolation erzeugt wiederum Stützpunkte auf Ebene der Einzelachsen als Sollwertvorgabe für die kaskadierte dezentrale Regelungsstruktur, bestehend aus Lage-, Drehzahl- und Stromregler.
Die Aktionssteuerung realisiert die Interaktion des Roboters mit dem Prozess [63]. Für die Anbindung externer Aktuatoren und Sensoren wie auch übergeordneter Steuerungssysteme werden Schnittstellen zu standardisierten Feldbussystemen wie Profibus, CAN-Bus sowie weiteren Systemen aus dem Bereich des Industrial-Ethernet bereitgestellt. Zum vereinheitlichten Transfer von Informationen über die Bussysteme für nicht-Echtzeit-gebundene Informationen, bspw. Visualisierungen und MES, wird in der Regel der OPC-Standard zum Austausch von Daten in der Automatisierungstechnik unterstützt [64]. Zusätzlich wird oft die direkte Anbindung peripherer Prozesse, bspw. Schweißsteuerung, Fördersysteme, und externer Sensorik über analoge und digitale Ein- und Ausgänge ermöglicht.

[12] Vgl. VDI 2860:1990-05 [61].

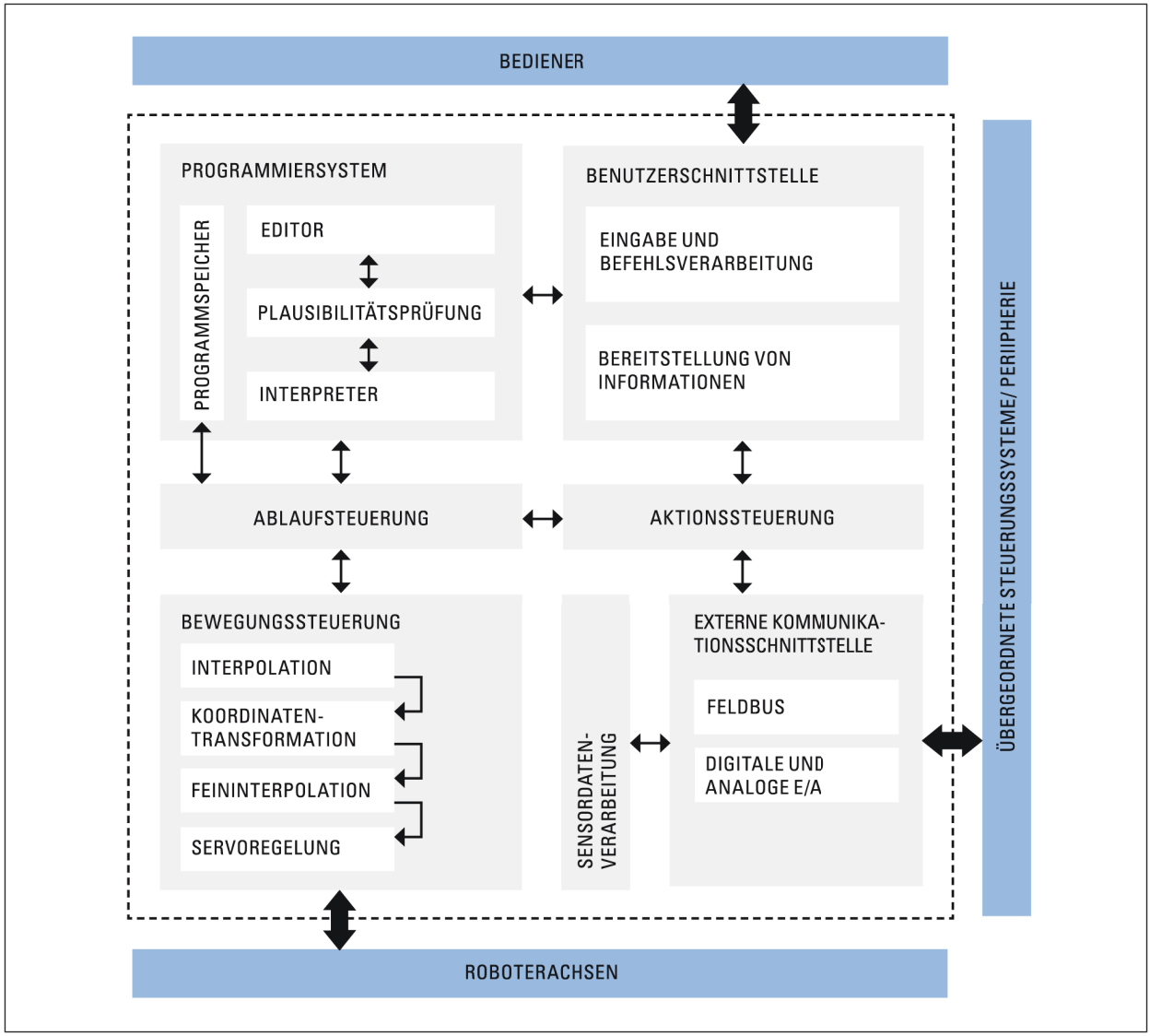

Abbildung 2.10: Komponenten und Schnittstellen einer Industrierobotersteuerung in Anlehnung an [62, 63, 65].

B) Das Programmiersystem als Steuerungskomponente

Das Programmiersystem als Komponente der Robotersteuerung versetzt den Anwender in die Lage, Befehle und Bewegungsprogramme zu definieren, zu adaptieren und zu testen. Unter der Tätigkeit der Programmierung von Industrierobotern wird die Definition von Bewegungsfolgen und Aktionen des Roboters in Form eines Anwenderprogramms verstanden [62]. Ein Anwenderprogramm ist eine Sequenz von Anweisungen mit dem Zweck, eine vorgegebene Handhabungs- oder Fertigungsaufgabe zu erfüllen. Das Anwenderprogramm liegt dabei bei gängigen Industrieroboter in textueller Form und in einer spezifischen Programmiersprache vor. Es beinhaltet neben Bewegungsanweisungen, Effektoranweisungen, Sensorabfragen, Programmablaufkontrollanweisungen, arithmetische Ausdrücke und technologische Anweisungen. Diese Anweisungen werden beim Ausführen eines Roboterprogramms mittels Ablauf- und Aktionssteuerung durch den Roboter bzw. die beteiligte Hardware realisiert. Das Programmiersystem umfasst weiterführend Funk-

tionen zur Erstellung, Wartung und Verwaltung von Anwenderprogrammen. Zur Definition von Bewegungen stehen Befehle der Punkt-zu-Punkt-Steuerung und der Bahnsteuerung zur Verfügung. In der Regel verfügen moderne Industrierobotersteuerungen neben der textuellen Eingabemöglichkeit zusätzlich über Eingabemasken oder grafische Benutzeroberflächen zur Definition der Roboterprogramme.

C) Programmiersprachen und -schnittstellen

Versuche zur Definition standardisierter Schnittstellen und Programmiersprachen, ähnlich ISO 6983[13] für Werkzeugmaschinen, erfolgten in der Vergangenheit in Form der generischen Programmiersprache Industrial Robot Language (IRL) und der Schnittstellendefinition Industrial Robot DATA (IRDATA). Obwohl die Erarbeitung dieser Standards durch ein breites Konsortium aus Forschungseinrichtungen und Industrievertretern vorgenommen wurde, konnten sich die Normen nie zu einem Industriestandard entwickeln. Die ehemaligen Normen und eine zugehörige VDI-Richtlinie sind mittlerweile technisch überholt und wurden ersatzlos zurückgezogen [67]. Der Grund für das Scheitern der Standardisierung ist weniger in der technischen Realisierbarkeit als in der Produktpolitik (Einsatz als Alleinstellungsmerkmal und Mittel der Kundenbindung) der führenden Industrieroboterhersteller zu suchen.
Vereinzelte Industrierobotersteuerungen gängiger Hersteller unterstützen die Ausführung von CNC-Programmen. Die Bewegungsanweisungen im G-Code werden über einen entsprechenden Interpreter in steuerungsinterne Befehle umgewandelt, sodass eine Übertragung der Bewegungen auf die Kinematik des Industrieroboters erfolgt. Auch die Unterstützung der Programmiersprachen von Speicherprogrammierbaren Steuerungen (SPS) sowie die direkte Integration der SPS in die Robotersteuerung werden von einigen Herstellern in jüngster Zeit angeboten [68]. Zum Übertragen, Ausführen und Manipulieren von Anwenderprogrammen durch externe Steuerungssysteme existieren verschiedenste herstellerspezifische Schnittstellen, welche sich stark in ihrem Funktionsumfang unterscheiden. Zum Standardumfang gehört in der Regel das datenträgerlose Übertragen, Starten und Stoppen von Programmen. Bei offenen Steuerungssystemen, welche speziell in der Forschung und Entwicklung eingesetzt werden, ist prinzipiell eine flexiblere Beeinflussung des Roboterverhaltens bis hin zur Manipulation von Regelparametern der Einzelachsen möglich. Entsprechend dem oft eingeschränkten Umfang der Schnittstellenfunktionen hängt die Einsetzbarkeit individueller Programmierverfahren und -systeme prinzipiell von den bereitgestellten Schnittstellen des jeweiligen Roboterherstellers ab.

2.2.2 Programmierverfahren

Programmierverfahren beschreiben das planmäßige Vorgehen zur Erzeugung von Anwenderprogrammen [62]. Die gängigen Programmierverfahren lassen sich in prozessnahe Verfahren unter Verwendung des Industrierobotersystems (online) und prozessferne Verfahren unter Verwendung eines externen PCs (offline) unterscheiden. Daneben existieren hybride Programmierverfahren, welche beide Ansätze kombinieren. Bei den hybriden Ausführungen wird der Programmablauf

[13] Vgl. ISO 6983-1:2009-12 [66].

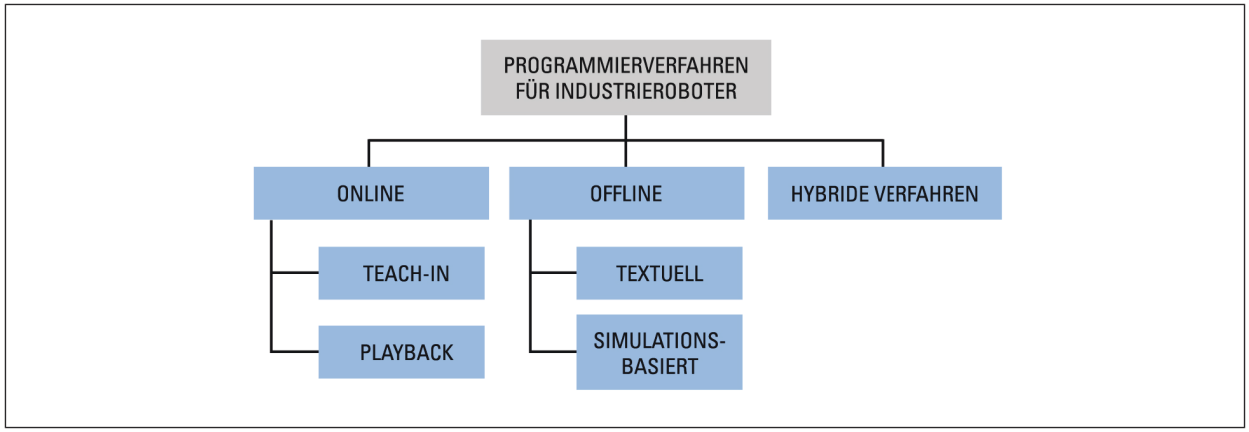

Abbildung 2.11: Taxonomie der Programmierverfahren für Industrieroboter in Anlehnung an [62, 63].

meist durch Offline-Verfahren festgelegt und der Bewegungsteil in Form genauer Posen prozessnah am Roboter definiert. Eine weiterführende Einteilung der gängigen Programmierverfahren ist Abbildung 2.11 zu entnehmen.

A) Online-Verfahren

Teach-In
Bei der Teach-In-Programmierung verfährt der Programmierer den Roboter über ein Handbediengerät an gewünschte Posen und speichert diese ab. Anschließend werden die Posen mit Bewegungsanweisungen verknüpft und in einen sequentiellen Ablauf gebracht. Zusätzlich werden Programmverzweigungen, Aktionen und weitere Anweisungen definiert. Für die Erstellung des textuellen Programms werden meist eine textuelle Eingabe, Eingabemasken und weitere Elemente von grafischen Benutzerschnittstellen genutzt.

Playback
Bei den Playback-Verfahren wird der Roboter durch den Bediener haptisch geführt. Durch die Steuerung werden einzelne Posen oder ganze Trajektorien aufgezeichnet. Zur flexiblen Führung des Roboters werden eine Bewegungssteuerung auf Basis der gemessenen Motorströme oder auf Basis von Kraft-Momenten-Sensoren und eine entsprechende Nachgiebigkeitsregelung benötigt. Sind diese am adressierten Robotersystem nicht vorhanden, lassen sich Bewegungen auch über ein zusätzliches Programmier- oder Phantomgerät aufzeichnen und auf den Zielroboter übertragen. Eine typische Anwendung ist die Programmierung von Lackierrobotern. Bei der Master-Slave-Programmierung werden Bewegungen durch das Führen eines alternativen Robotersystems aufgezeichnet und auf das Zielsystem übertragen. Aufgrund der Ähnlichkeit des Verfahrens kann auch auch diese Programmiermethode den Playback-Verfahren zugeordnet werden.

B) Offline-Verfahren

Offline-Programmierverfahren dienen prozessfern der Erstellung von Industrieroboterprogrammen. Daraus resultiert ein entscheidender Vorteil: die potentielle Minderung von Produktionsstillstandszeiten. Der Prozess der Roboterprogrammierung lässt sich somit bspw. in die Arbeitsvorbereitung auslagern. Als Bestandteil der Fertigungsplanung kann der Programmierprozess durch

die Integration betrieblicher Informationssysteme, bspw. der Digitalen Fabrik, unterstützt werden. Nach Erstellung und einem erfolgreichen Test des Programms wird dieses über Datenträger oder Bussysteme auf die Robotersteuerung übertragen.

Textuelle Offline-Programmiersysteme
Bei textuellen Offline-Programmiersystemen (OLPs) erfolgt die Eingabe von einzelnen Roboteranweisungen über die Tastatur eines externen PCs. Benötigte Geometrieeingaben können über textuelle Eingabe mit grafischer Unterstützung übernommen werden. OLPs mit ausschließlich textueller Eingabe sind in der Praxis allerdings nur noch selten anzutreffen. Die Funktionalität der textuellen Eingabe ist in die meisten kommerziellen simulationsgestützten Programmiersysteme integriert.

Simulationsgestützte Offline-Programmiersysteme
Simulationsgestützte OLPs nutzen entweder bestehende CAD-Systeme oder bringen eigene 3D-Funktionalitäten und Schnittstellen mit. Das Robotersystem liegt als virtuelles Modell vor. Die Simulation der Roboterzelle und Roboterkinematik ermöglicht ein virtuelles Testen von Roboterprogrammen mit der Ermittlung von Taktzeiten, Erreichbarkeits- und Kollisionskontrollen. Der Standardumfang der OLPs umfasst das Bewegen des Robotermodells über ein virtuelles Handbediengerät. Analog zur Teach-In Programmierung können Posen angefahren, abgespeichert und mit Bewegungsparametern wie Interpolationsart, Geschwindigkeit und Beschleunigungen versehen werden. Unterstützt werden diese Funktionen in der Regel durch Methoden der textuellen und grafischen Programmierung. Da die Angabe der Bewegungsparameter durch den Benutzer erfolgt, wird diese Art der Programmierung auch als explizit bzw. bewegungsorientiert bezeichnet. Erweiterte Funktionen der automatisierten Programmierung für spezifische Aufgaben stellen sogenannte „Technologiemodule“ dar [69]. Diese beinhalten Expertenwissen, welches auf Basis vorhandener Geometrieinformationen des Werkstücks zur teil- oder vollautomatisierten Programmerstellung genutzt werden kann. Da die Programmierung der Fertigungsaufgabe derart ohne breites Wissen über den Prozess und die Industrieroboterprogrammierung durchgeführt werden kann, wird diese Art der Programmierung auch als implizit bezeichnet.
Führende Industrieroboterhersteller bieten eigene Simulationssysteme an, die Roboterprogramme in der herstellerspezifischen Programmiersprache importieren und exportieren. Herstellerunabhängige Simulationssysteme unterstützen in der Regel mehrere Programmiersprachen. Intern liegt oft eine herstellerneutrale Repräsentation des Roboterprogramms vor. Über entsprechende Schnittstellen, sogenannte „Prä- und Postprozessoren“, werden die Roboterprogramme übersetzt. Sollen verschiedene Programmiersprachen adressiert werden, steigert dies entsprechend den Aufwand zur Erstellung und Wartung der Schnittstellen. Diesbezüglich stellen Freund et al. [70] ein Framework zur vereinfachten Übersetzung des Robotercodes vor.

2.2.3 Weiterführende Betrachtung der Programmierverfahren

A) Abstraktionsgrad zur Auslegung von Programmierverfahren

Wird der Begriff der Programmierung steuerungstechnisch näher in Richtung Roboterhardware bzw. Benutzeranwendung abstrahiert, handelt es sich um Low-Level- bzw. High-

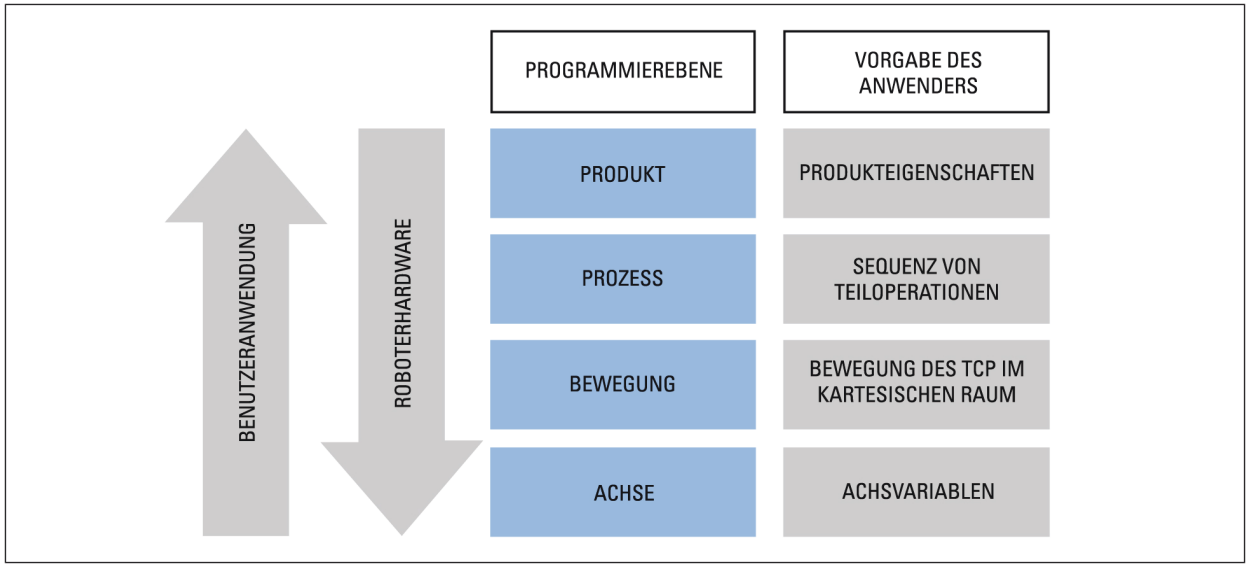

Abbildung 2.12: Abstraktionsebenen der Roboterprogrammierung in Anlehnung an [72].

Level-Programmierung [71]. Die unterschiedlichen Abstraktionsgrade werden auch als Programmierebenen bezeichnet, welche sich grundlegend in Produkt, Prozess bzw. Aufgabe, (Endeffektor-)Bewegung und Achse unterscheiden lassen. Abbildung 2.12 verdeutlicht diese Einteilung. Für den Programmierer unterscheiden sich abhängig vom Abstraktionsgrad die Art und die Anzahl der Vorgaben zur Definition eines Anwenderprogramms. Auf der untersten Ebene erfolgt die Definition des Verhaltens des Roboters über die Vorgabe von Achsvariablen für die Einzelachsen. Auf der Bewegungsebene werden Bewegungen des Roboterflanschs oder Endeffektors durch kartesische Koordinatenangaben definiert. Auf Prozessebene erfolgt die Vorgabe einer Sequenz von Teiloperationen. Auf Produktebene lassen sich nur noch die gewünschten Eigenschaften des Werkstücks übergeben. Der Roboter kann diese eigenständig abstrahieren und in Aktionen übertragen und ausführen. Die Durchführung einer Programmierung auf Prozessebene wird auch als aufgabenorientierte Programmierung bezeichnet und ist in ihrer Ausführung mit der impliziten Offline-Programmierung vergleichbar [72].

Je geringer der gewählte Abstraktionsgrad der Steuerungsfunktionen desto genauer und schneller kann der Anwender das Verhalten des Roboters beeinflussen. High-Level-Programmierung ist prinzipiell effizienter und erfordert kein spezifisches Fachwissen. Zudem lässt sich ihre Ausführung näher an der natürlichen Kommunikation des Menschen anlehnen, stellt jedoch erhöhte Anforderungen an die sensorischen und kognitiven Fähigkeiten des Robotersystems, da umfassendes Wissen über die Umgebung und die automatisierte Ableitung von Low-Level-Anweisungen erforderlich ist. Der optimale Abstraktionsgrad der Roboterprogrammierung ist anwendungsspezifisch und richtet sich nach Prozessanforderungen, Standardisierung des Prozesses und Expertise des Programmierers.

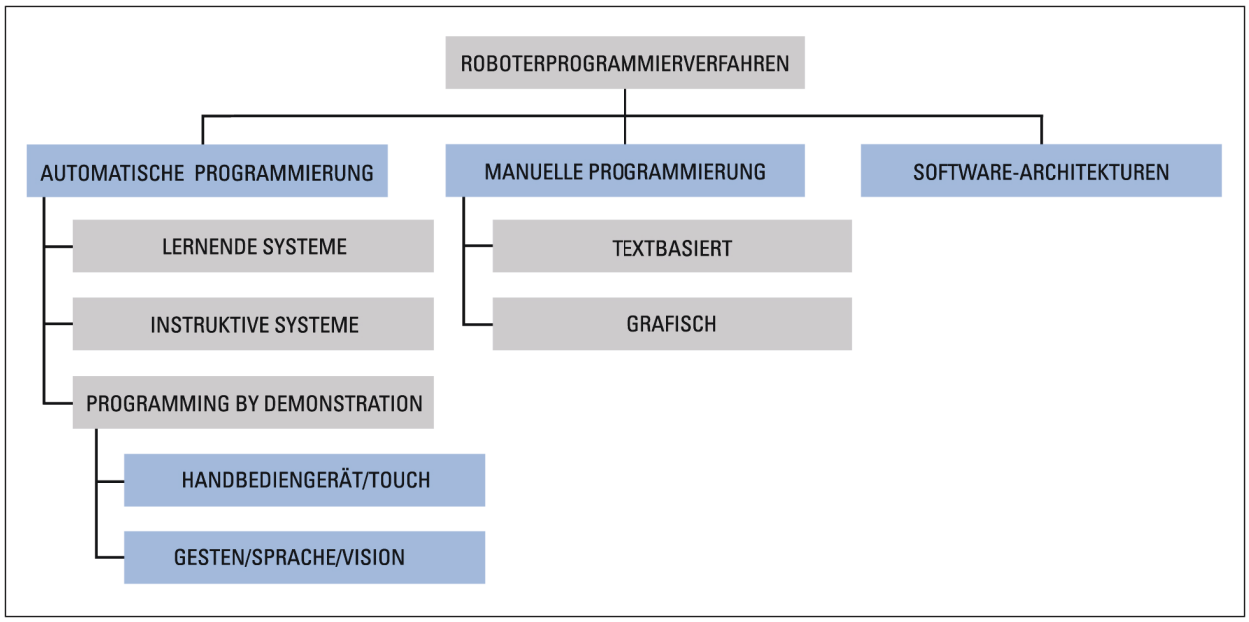

Abbildung 2.13: Alternative Einteilung der Programmierverfahren für Roboter nach [73].

B) Alternative Gliederungen der Programmierverfahren

Die klassische Einteilung der Programmierverfahren für industrielle Roboter[14] gibt eine grobe Taxonomie vor. Diese unterliegt jedoch keinerlei Normung. In der Praxis ist eine eindeutige Zuordnung einzelner Programmierverfahren meist nur schwer möglich bzw. sind die Kriterien online oder offline unpassend zur exakten Beschreibung der Verfahren. Englischsprachige Veröffentlichungen der Fachliteratur wählen zumeist eine alternative Gliederung, welche weniger das Kriterium der Prozessnähe, als mehr das zugrunde liegende Interaktionsprinzip der Programmierung betrachtet. Hier wird zunächst unterschieden zwischen manueller oder automatisierter Programmerstellung [73]. Diese Form der Gliederung (s. Abb. 2.13) ist weiter gefasst und adressiert prinzipiell nicht ausschließlich die Programmierung von Industrierobotern, sondern ist ebenso auf mobile und autonome Robotersysteme übertragbar. Es werden bspw. auch Programmierverfahren wie lernende Systeme und komplexe Softwarearchitekturen erfasst, die im industriellen Umfeld noch keine bzw. wenig Verbreitung gefunden haben.
Weiterführend bietet diese Taxonomie im Vergleich zur klassischen Betrachtung eine differenzierte Betrachtung neuartiger Industrieroboterprogrammierverfahren aus der Forschung und Entwicklung. Neben der manuellen grafischen Programmierung rücken das Programmierparadigma „Programming by Demonstration“ (PbD) sowie die instruktive Programmierung im Rahmen aktueller Forschungsprojekte zur intuitiven Industrieroboterprogrammierung immer mehr in den Vordergrund [74].

Programming by Demonstration
Ein umfassender Ansatz, die Programmierung von Industrierobotern intuitiver zu gestalten, lässt sich aus dem PbD-Prinzip ableiten. Dieses Paradigma beschreibt das Lernen durch Imitation mit dem Ziel, Fähigkeiten des Menschen auf den Roboter zu übertragen. Dabei handelt es sich mitt-

[14] Vgl. Kapitel 2.2.2.

lerweile um ein breites Forschungsfeld, welches neben Mensch-Roboter-Interaktion oftmals die Disziplinen künstlicher Intelligenz, Bildverarbeitung, Bahnplanung und Motorsteuerung mit einbezieht. Kernpunkt des PbD ist ein sensorielles Wahrnehmen (engl. *perception*) von Aktionen, die meist durch einen menschlichen Anwender durchgeführt werden. Auf Grundlage dieser Informationen wird versucht, ein Roboterprogramm automatisiert oder anhand einer vorgegebenen Vorschrift abzuleiten, welches bestimmte Verhaltensweisen der vorgemachten Aktion (engl. *action*) imitiert [75].
Das Vormachen und Imitieren kann auf unterschiedlichen Abstraktionsebenen und unter Verwendung verschiedener Hilfsmittel erfolgen. Gegenstand der Imitation können demnach sowohl einzelne Posen als auch Bewegungen, Bewegungssequenzen (engl. *motion learning from demonstration*) sowie Prozesse oder Aufgaben (engl. *task learning from demonstration*) sein [76]. Aufgrund der Spannweite in der Auslegung umfasst PbD sowohl die klassischen Verfahren der Industrierobotik, bspw. Teach-In und Playback, als auch komplexe Verfahren in der autonomen Robotik unter Verwendung von Methoden der künstlichen Intelligenz und Kognition [77].
Die Umsetzung des Paradigmas in eine Programmiermethode gestaltet sich verhältnismäßig einfach, wenn sich die Vorschrift zur Erzeugung des Programms analytisch herleiten und über feste Bedingungen abbilden lässt. Lediglich die Ausführung der Demonstration obliegt hier dem Endanwender, wobei auf aufwendige Anlernvorgänge zum maschinellen Erlernen der Vorschrift verzichtet werden kann. Derartige Programmiersysteme lassen sich durch die Durchführung der Imitation auf der Ebene der Aufgabe und der Nutzung von Mensch-zu-Mensch-Kommunikation intuitiv und effizient gestalten.

Instruktive Programmierung
Instruktive Systeme stellen im Vergleich zum PbD eine höhere Ebene der Abstraktion in der Interaktion zwischen Mensch und Roboter dar. Im Gegensatz zum PbD-Ansatz wird die auszuführende Aufgabe nicht mehr durch Vormachen angelernt, sondern lediglich angedeutet. Somit wird der Aufwand zum Auslösen einer Aktion weiter minimiert und automatisiert. Als Modalitäten bieten sich analog zur menschlichen Kommunikation Sprache und Gesten, ggf. in multimodaler Ausführung, an. Zugrunde liegt bei diesen Systemen immer eine vorgegebene Aktion bzw. Bewegungsfolge, die sich nach festen Vorschriften und Beobachtung der Instruktion parametrieren und ausführen lässt. Die Aktionen und Bewegungsfolgen lassen sich bei Bedarf vor Einsatz der instruktiven Programmierung durch ein PbD-System erstellen.

2.3 Forschungsaktivitäten zur intuitiven Industrieroboterprogrammierung

In diesem Abschnitt wird der Stand der Forschung bezüglich intuitiver Industrieroboterprogrammierung dargelegt. Die allgemeine Motivation zur Entwicklung neuer Programmiermethoden und -geräte wurde mit Schwerpunkt auf KMU bereits umfangreich in wissenschaftlichen Publikationen, unter anderem in [4, 78, 79, 80, 81, 82], diskutiert. Im Weiteren erfolgt zunächst eine chronologische, projektspezifische Betrachtung der Forschungsaktivitäten der letzten Jahre. Anschließend

werden ergänzend Forschungsergebnisse zur räumlichen Programmierung auf Basis von Gesten und AR näher betrachtet.

2.3.1 Projektspezifische Betrachtung

A) Morpha (2000-2003)

Im Jahr 2003 endet das Verbundprojekt Morpha. Das Ziel des Projekts besteht darin, intelligente mechatronische Systeme, insbesondere Robotersysteme, mit leistungsstarken Kommunikations-, Interaktions- und Verhaltensmechanismen auszustatten. Der Fokus des Forschungsprojekts liegt auf der Entwicklung interaktiver und innovativer Technologien für die Mensch-Roboter-Interaktion durch natürliche Kommunikation. Im Nachhinein erweist sich dieses Projekt als wegweisende Grundlage für zahlreiche Folgeprojekte zur intuitiven Steuerung und Programmierung im Bereich der Service- und Industrierobotik. Das Konsortium stellt bereits rudimentäre sprach- [83] und berührungsbasierte [84, 85] Programmiersysteme vor. Gesten werden als instruktive Form sowie zur Ableitung von Aufgaben nach dem Prinzip des PbD eingesetzt. In diesem Zusammenhang evaluieren Dillmann et al. [86] verschiedene Methoden der interaktiven natürlichen Programmierung und implementieren eine Anwendung nach dem PbD-Prinzip für Dual-Arm-Aufgaben unter der Verwendung von Petri-Netzen zur Aufgabenkoordination [87]. Die Forschungsarbeiten zum automatisierten Erlernen von Dual-Arm-Aufgaben durch Imitation werden im Sonderforschungsbereich Humanoid Robots (SFB-588) fortgesetzt [88]. Der Roboterhersteller KUKA verfolgt im Projekt Morpha das Ziel, die Programmierung über einen Ansatz der grafischen Programmierung zu vereinfachen [89]. Im Anschluss an das Morpha-Projekt stellt KUKA den Prototypen eines neuen kabellosen Handbediengeräts mit Touchscreen vor [90]. In diesem Zusammenhang wird bereits die Verwendung von AR zur Unterstützung des Anwenders bei der Bedienung der 3D-Maus am Handbediengerät und zur Visualisierung von Koordinatensystemen, Bewegungsrichtung sowie der Geschwindigkeit des Endeffektors diskutiert.

B) IRoProg (2002-2005)

Das Forschungsprojekt „Innovative Roboter-Programmiermethoden“ (IRoProg) [4] widmet sich der Entwicklung neuer Methoden, um die Dauer und die Kosten der Roboterprogrammierung zu senken. Es besteht die Motivation, vor allem KMU eine erweiterte Möglichkeit zur wirtschaftlichen Nutzung von Industrierobotern zu geben. Die Projektergebnisse umfassen eine 3D-Offline-Programmierumgebung mit haptischer Eingabe von Punkten, Kollisionskontrolle durch Kraftrückkopplung, automatische Bahnplanung und verbesserte Kalibriermethoden, die eine schnelle Inbetriebnahme von Roboteranlagen ermöglichen. Zusätzlich werden neue Geräte und Methoden zum Handverfahren des Roboters beim Teach-In betrachtet. Über einen Zeigestift werden Posen für den Roboter im Arbeitsraum vorgegeben, die dieser anschließend anfährt. Neben optischem Tracking des Zeigestifts wird Inertialsensorik zur Bestimmung der Orientierung genutzt. Eine weitere betrachtete Interaktion ist das Führen des Roboters über einen am Flansch montierten Joystick.

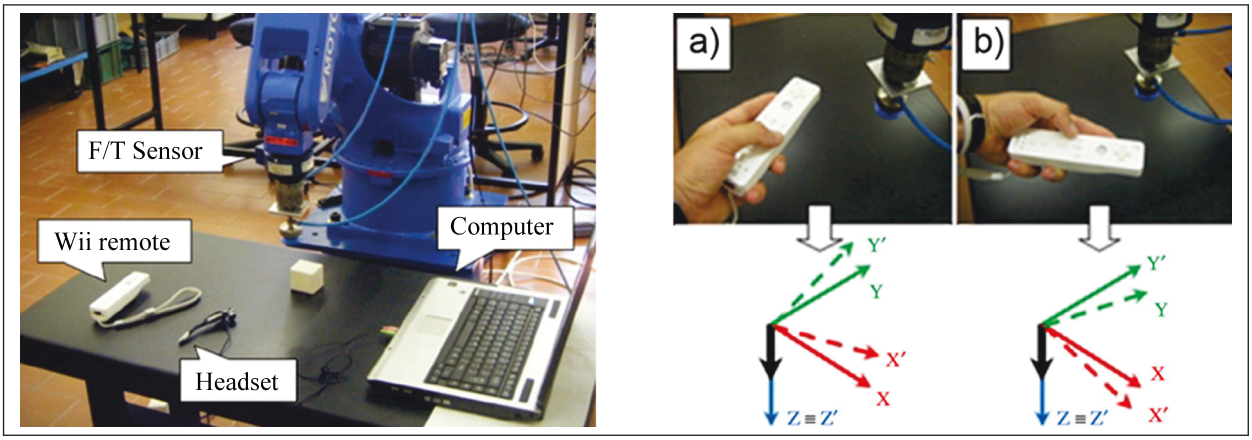

Abbildung 2.14: Interaktives Programmieren über Gesten und Sprachkommandos, Aufbau (links) und gestenbasierte Steuerung des Roboters über Gestenerkennung (Gesten a und b) durch die Beschleunigungssensoren des Wii-Controllers (rechts), Quelle: [71].

C) SMErobot (2006-2009)

Das europäische Verbundprojekt SMErobot versteht sich als Robotikinitiative zur Stärkung der Wettbewerbsfähigkeit von KMU. Angestrebt werden innovative Robotiklösungen zur sicheren Mensch-Roboter-Kooperation, zum Verständnis von menschlichen Anweisungen (Sprache, Gesten, Zeichnungen) und zur schnellen Einsatzbereitschaft von Robotersystemen [91]. Neto und Pires [71] beschäftigen sich mit interaktiven Programmiermethoden durch menschliche Instruktionen in Form von Sprache und Gesten (s. Abb. 2.14). Als Eingabegerät wird unter anderem der mit Beschleunigungssensoren ausgestattete Controller der Spielekonsole Wii von Nintendo genutzt. Zum Anlernen und Erkennen von dynamischen und statischen Instruktionsgesten dienen künstliche neuronale Netze. In einem weiteren Teilprojekt zur multimodalen Roboterprogrammierung [92] wird neben Sprachkommandos auch das haptische Führen des Roboters betrachtet. Die Trajektorien werden aufgenommen und anwendungsspezifisch nachbearbeitet. Automatisiert lässt sich daraus im Anschluss ein Roboterprogramm erzeugen [93]. Zusätzlich fungiert ein Industrie-Personal-Digital-Assistant (Industrie-PDA) der Schutzklasse IP65 als mobiles, drahtloses Teach-In-Gerät mit unterstützender 3D-Simulation des Roboterprogramms in rudimentärer Ausführung. Als potentielle Anwendungen werden das Kleben und Bahnschweißen identifiziert. Pires et al. [94] evaluieren weiterführend die Nutzung eines digitalen Stifts zum virtuellen Zeichnen von Trajektorien. Der Stift soll weiterführend zur Parametrierung und zur Instruktion des Roboters genutzt werden. Unterstützt wird das angestrebte interaktive Programmiersystem durch eine integrierte CAD-Anwendung und automatische Programmgenerierung.

D) ProDemo (2007-2009)

Das Projekt „Modellbasierte Roboterprogrammierung by Demonstration“ (ProDemo) verfolgt das Ziel der Aufwandsreduzierung für die werkstattorientierte Programmierung von Industrierobotern durch einen hybriden Ansatz aus Online- und Offline-Programmiermethoden [95]. Ein markerbehaftetes räumliches Zeigegerät mit verschiedenen Tastern dient zur Definition von Positionen und Orientierungen im Raum. Das Tracking erfolgt über ein Kamerasystem, befestigt am Flansch

des Industrieroboters. Somit kann der Roboter den Markern durch einen Visual-Servoing-Ansatz folgen. Die Befestigung des Kamerasystems am Roboterflansch resultiert zudem in einer Vergrößerung des Sichtbereichs des Kamerasystems im Vergleich zu einer stationären Positionierung. Die aufgenommenen 3D-Bahndaten werden über Schnittstellen in eine Simulationsumgebung auf einem fest stehenden PC in der Werkstattumgebung übertragen und visualisiert. Das Roboterprogramm wird innerhalb der Simulationsumgebung automatisch generiert, kann in herstellerspezifische Programmiersprachen umgewandelt und durch die Nutzung einer grafischen Programmiersprache angepasst werden. Neben dem Zeigegerät werden auch Werkstücke mit Markern ausgestattet. Diese können vom Anwender durch Führung mit der Hand auf einer gewünschten Bahn durch den Arbeitsraum des Roboters bewegt werden. Anhand der aufgenommenen Werkstückposen lässt sich wiederum automatisiert ein Roboterprogramm ableiten, welches das Werkstück zunächst an einer Ablageposition greifen lässt und anschließend auf der imitierten Bahn zu einer Ablageposition bewegt. Zusätzlich wird durch das Zeigegerät die Geometrie der Arbeitszelle in die Simulation zurückgeführt. Dies erfolgt lediglich rudimentär, um einer virtuellen Kollisionserkennung gerecht zu werden. Eine komplette Erfassung von unbekannten Umgebungen wird nicht adressiert. Stattdessen können bekannte Objekte, deren 3D-CAD-Daten vorliegen, durch die Definition von drei Raumpunkten in der Simulation platziert werden.

E) Räumlich-interaktive Roboterprogrammierung (2005-2010)

W. Vogl [96] befasst sich im Rahmen eines Forschungsprojekts zur Entwicklung von AR-Methoden in der Montageplanung sowohl mit der räumlichen Definition von Trajektorien für Industrieroboter als auch mit der AR-basierten Evaluation von Industrieroboterprogrammen. Bahnpunkte und Trajektorien werden im Raum über ein Eingabegerät definiert (s. Abb. 2.15). Ein 6DoF-

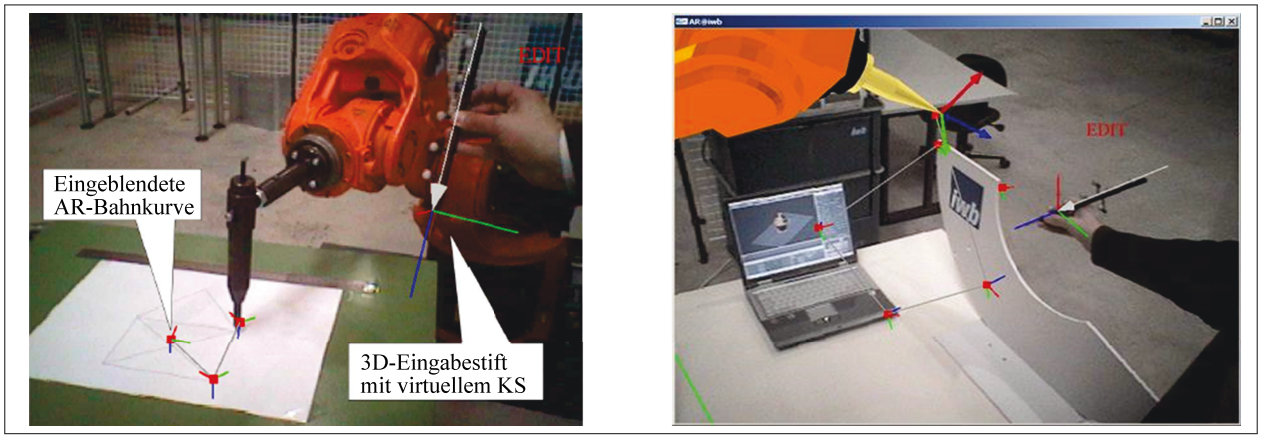

Abbildung 2.15: Räumlich-interaktive Industrieroboterprogrammierung: räumliche Definition von Posen über Zeigestift (links) und Simulation über virtuellen Roboter in der AR (rechts), Quelle: [96].

Tracking-System ermittelt Position und Orientierung des Eingabegeräts. Unterstützt wird der Anwender durch eine AR-Visualisierung. Diese umfasst eine kinematische Simulation des Roboterprogramms und wird auf einem Tablet-PC der Schutzklasse IP54 umgesetzt. Allerdings erfolgt lediglich eine Anzeige des erweiterten Bilds einer stationären Kamera auf dem Mobilgerät. Auch

die Berechnungen der Simulation werden auf einem Basisrechner durchgeführt. In Ergänzung zur 3D-Darstellung der Roboterprogramme erfolgt eine projektionsbasierte Visualisierung mittels Laserprojektoren. Die Visualisierungen auf der Werkstückoberfläche werden zur interaktiven Definition von Trajektorien und Aufgaben am Werkstück genutzt. Darüber hinaus werden Methoden zur Interaktion mit virtuellen Displays umgesetzt. Diese lassen sich unter anderem zur Parameteranpassung des Roboterprogramms nutzen. Evaluiert wird das System an den Anwendungsszenarien Laserhärten und Remote-Laserstrahlschweißen. In Experimenten kann, verglichen mit dem Teach-In, eine Verringerung der Programmierzeit von über 30 % nachgewiesen werden [97].

F) ImRoNet (2007-2010)

Die Entwicklung multimedialer und multimodaler Nutzerschnittstellen für die internetbasierte Telerobotik ist das Ziel des Verbundprojekts ImRoNet. Die teleoperierte Steuerung mobiler Roboterplattformen adressiert die Anwendungsgebiete der Wartung und Inspektion. Zur Vorgabe von Bewegungen und Aufgaben wird ein haptisches Gerät mit Vibrationsfeedback eingesetzt. Dieses dient zur Steuerung einer mobilen Roboterplattform, ausgerüstet mit einem Roboterarm. Zur Unterstützung der Anwender wird die Roboterplattform um Display und Kamera sowie eine mobile AR-Umgebung ergänzt. Das AR-System, welches am Endeffektor des Roboterarms befestigt ist, lässt sich durch manuelles Führen des Arms ausrichten. Die Darstellung der AR kalibriert sich markerlos anhand von bekannten, inhärenten Objektmerkmalen [34, 98]. Ergänzend zur Display-Technologie wird eine weitere mobile Roboterplattform mit einem Laserprojektor ausgestattet. Dieser ist auf verschiedenen Servoachsen positioniert und lässt sich somit flexibel orientieren. Die virtuelle Projektion auf Oberflächen unterstützt den Anwender weiterführend durch die flexible räumliche Darstellung von Informationen [99].

G) AVILUS (2008-2011)

Der Roboterhersteller KUKA stellt 2006 erstmalig einen erweiterten Prototyp einer AR-Visualisierung, den KUKA Augmented Reality Viewer (KARV), vor (s. Abb. 2.16). Dieser basiert auf einem kommerziellen AR-Framework der Firma metaio GmbH und soll den Programmierer durch Visualisierung der Bewegungsbahnen und Simulation des Roboterprogramms unterstützen [100, 101]. Genutzt wird eine statische monitorbasierte Visualisierung. Das Verbundprojekt AVILUS beschäftigt sich mit der Forschung, Entwicklung und Evaluierung von Technologien der Virtual und Augmented Reality im Produkt- und Produktionsmittellebenszyklus [102]. Innerhalb des Projekts soll der KARV zu einer interaktiven Trainingsplattform erweitert werden. Weiterführend werden neue Funktionen zur Visualisierung virtueller Werkstücke und Kräfte am Endeffektor hinzugefügt. Ergebnis der Evaluation des bestehenden Systems ist, dass die Nutzung eines statischen 2D-Bildschirms für 3D-Interaktionen in der AR unzureichend ist. Es wird die Nutzung eines Stereo-HMD empfohlen, welches im Projekt nicht weiter evaluiert werden kann [103].

[15] Vgl. http://www.kuka-robotics.com - 20.01.2014.

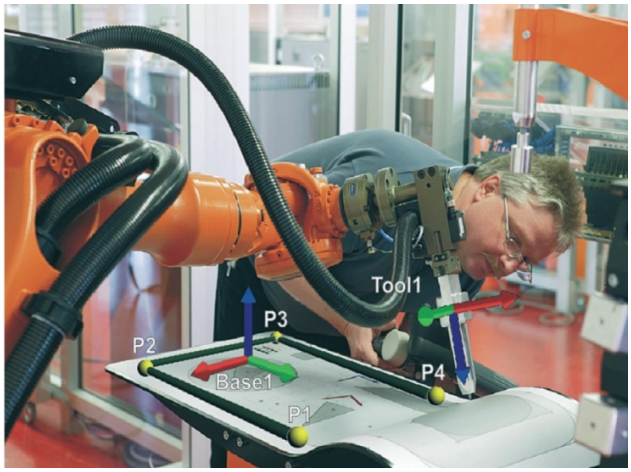

Abbildung 2.16: KUKA Augmented Reality Viewer, Anzeige von Koordinatensystem, Positionen und Bewegungsbahnen, Quelle: Homepage der Fa. KUKA[15].

H) SMErobotics (2010-2013)

Das Nachfolgeprojekt zu SMErobot fokussiert auf technischer Ebene die kognitiven Fähigkeiten des Robotersystems zum Erkennen und Erlernen von Situationen. Zudem werden die im Vorgängerprojekt entwickelten Verfahren zur haptischen Führung des Roboters weiterentwickelt. In diesem Zusammenhang stellen Thomas et. al. [104] einen Ansatz für eine neue Programmiersprache auf verschiedenen Abstraktionsebenen der Roboterprogrammierung vor. Dietz et al. [105] identifizieren Anforderungen für eine effiziente Gestaltung der Industrieroboterprogrammierung für KMU. Danach enthält das erstellte Konzept eines Programmiersystems das Programmieren durch haptisches Führen des Roboters, die Integration von CAD-Modellen in den Programmierprozess sowie die Ausstattung des Roboters mit Sensorik zur Identifikation von Werkstücken. Gaschler et al. [106] entwickeln basierend auf den Arbeiten von W. Vogl [96] ein Programmiersystem, welches die räumliche Definition von Hindernissen, Posen und Aufgaben durch ein markerbehaftetes Hilfsgerät ermöglicht. In Kombination mit einer virtuellen Simulation durch einen bereitgestellten Monitor wird eine projektionsbasierte AR-Visualisierung vorgenommen. Ergänzend zu den Arbeiten von W. Vogl wird eine Diskussion zur Definition der Orientierung von Posen sowie eine Evaluation verschiedener Ansätze durch Erfassung der Programmierdauer im Rahmen einer Nutzerstudie durchgeführt.

I) TAPAS (2010-2013)

Das Ziel des europäischen Verbundprojekts TAPAS [107, 108] ist die Entwicklung roboterbasierter Automatisierungs- und Logistiktechnologien, welche zukünftigen industriellen Anforderungen genügen. Eine flexible Anpassungsfähigkeit roboterbasierter Systeme an Stückzahlen und Produktionsart wird angestrebt. Neben dem Transport werden auch Maschinenbestückung, Vormontage und Qualitätskontrolle als potentielle Anwendungen für eine mobile Plattform mit einem Roboterarm adressiert. Im Sinne der flexiblen Adaption der Aufgabe wird neben der Navigation der Plattform auch die Programmierung des Roboters betrachtet. Tablet-PCs werden zur Definition von Roboteraufgaben genutzt. Eine angepasste GUI ermöglicht es, aus verschiedenen Unteraufgaben per

Abbildung 2.17: Bewegungssequenz zur gestenbasierten Steuerung eines Roboterarms, Quelle: [109].

Touch-Gesten Aufgabendefinitionen zu erstellen. Die Unteraufgaben lassen sich auch durch fachfremde Anwender in Form des haptischen Führens des Roboterarms definieren. Des Weiteren werden markerlose Gesten und deren kamerabasierte Erkennung zum instruktiven Programmaufruf genutzt. In der Ausführung dieser Methode werden vorprogrammierte Pick-and-Place-Aufgaben parametriert, indem der Nutzer auf das Zielobjekt zeigt und anschließend per Geste auf die gewünschte Ablageposition verweist.

2.3.2 Modalitätsspezifische Betrachtung

A) Gestenbasierte Programmieransätze

Räumliche Gesten haben vielfältige Einsatzmöglichkeiten in der Steuerung und Programmierung von Industrierobotern. Grundsätzlich lassen sich in aktuellen Publikationen drei verschiedene Anwendungsgebiete unterscheiden:

1. Kommandogesten,
2. Bewegungsdefinition sowie
3. Aufgabendefinition durch Demonstration.

Je nach Sensorik und Ausführung der Tracking-Anwendungen werden ggf. künstliche Hilfsmittel wie Datenhandschuhe oder Marker verwendet. Die erreichbaren Genauigkeiten hängen ebenfalls von der Sensorik ab. In jüngsten Forschungsprojekten zur Industrieroboterprogrammierung werden Gesten fast ausschließlich in multimodalen Steuerungssystemen verwendet, oft in Verbindung mit einer Sprachsteuerung. Im Folgenden wird der Stand der Forschung zur gestengestützten Online-Programmierung näher vorgestellt. Dabei erfolgt eine Beschränkung auf gestenbasierte Systeme, bei denen kein haptischer Kontakt zwischen Mensch und Industrieroboter erfolgt.
Aleotti et al. [109] definieren Bewegungsbahnen für einen Roboterarm über einen biege- und verdrehungssensitiven Sensor, der am Arm des Nutzers befestigt wird und die gesamte Länge des Arms abdeckt (s. Abb. 2.17). Das Ziel ist eine möglichst exakte Imitation der Armbewegungen durch den Roboter nach menschlicher Vorgabe. Durch das Training von künstlichen neuronalen Netzen wird ein steuerungstechnischer Zusammenhang zwischen Sensordaten und Bewegung des realen Roboterarms angelernt. Die erzielten Genauigkeiten der Bewegungsvorgabe genügen lediglich für eine grobe Positionierung des Roboterarms.
Dixon et al. [110] realisieren anhand eines Datenhandschuhs die gestenbasierte Vorgabe von Absolutpositionen im Arbeitsraum des Roboters sowie eine Bewegungssteuerung per Zeigegesten. Es

Abbildung 2.18: Relative Bewegungsvorgabe durch Führung eines Programmiergeräts unter Verwendung von visuellem Tracking, Quelle: [111].

werden insgesamt drei verschiedene Zeigegesten anhand der Stellungen einzelner Finger erkannt. Die Pose des Datenhandschuhs wird in 6Dof verfolgt. Mit den erreichten Genauigkeiten lässt sich die Programmierung von Linearbahnen für das robotergestützte Schweißen realisieren. Zur weiterführenden multimodalen Steuerung des Roboters, bspw. Starten oder Stoppen von Programmen, werden zusätzlich Sprachkommandos genutzt.

Hein et al. [111] stellen eine 6Dof-Bewegungsführung des Industrieroboters über ein handgeführtes Eingabegerät vor, welches mit kugelförmigen passiven Markern ausgestattet ist und von einem visuellen Tracking-System lokalisiert wird. Neben der Übertragung der Relativbewegungen der menschlichen Hand auf den Roboter im freien Raum wird eine Anwendung zur Imitation der menschlichen Schrift präsentiert (s. Abb. 2.18). Die Bewegungssteuerung stellt eine Teilfunktion eines komplexen Programmiersystems dar, welches weiterführende Methoden der Telerobotik und Offline-Programmierung, wie Kollisionsvermeidung und automatische Pfadplanung, unterstützt [112].

Stipancic et al. [113] verwenden den Kinect-Sensor zur Realisierung einer markerlosen Bewegungssteuerung eines Industrieroboters über Hand- und Körpergesten. Das gezielte Anfahren von Posen wird zur Erstellung von Roboterprogrammen genutzt. Die Autoren diskutieren zunächst die Vorteile der kostengünstigen Sensorik aus dem Consumer-Bereich. Weiterführend werden zwei verschiedene Steuerungsmodi unterschieden. Zum einen lässt sich der Endeffektor über relative Bewegungen der Hand verfahren. Um trotz mangelnder Genauigkeiten des Sensors auch Feinpositionierungen vornehmen zu können, wird ein zusätzlicher Modus zur skalierten Übertragung der Relativbewegungen implementiert. Zum anderen werden statische Kommandogesten zum Auslösen von Aktionen, bspw. Abspeichern der aktuellen Pose des Endeffektors und Betätigen des Greifers, vorgesehen. Ein Feedback über den aktuellen Zustand des Tracking und die Erkennung der Gesten liefert ein Monitor, welcher im Arbeitsraum in der Roboterzelle aufgestellt wird. Diese Form des statischen monitorbasierten Feedbacks wird als unzureichend identifiziert. Ein weiterer Nachteil liegt im komplexen Umfang der genutzten Kommandogesten. Diese müssen durch den Nutzer wie eine Sprache erlernt werden. Für weitere Entwicklungsarbeiten ist ein verstärkter Kontextbezug zwischen den Kommandogesten und den Steuerungsfunktionen geplant.

León et al. [114] stellen einen PbD-Ansatz zur Imitation von Pick-and-Place-Aufgaben durch einen Roboterarm vor. Zum Tracking von Hand und Greifobjekt wird ebenfalls der Kinect-Sensor verwendet. Die Bewegungsbahnen und Aktionen werden jedoch nicht fest vorgegeben, sondern werden aus einer bestimmten Anzahl von Demonstrationen erlernt. Hierzu dient eine Methode des explorativen Lernens, die vom Nutzer zusätzlich überwacht und bei Bedarf angepasst wird. Dieser gibt per Sprachsteuerung ein Feedback zum aktuellen Imitationsversuch des Roboters und hilft somit zusammen mit einem Gütekriterium, welches unter anderem die Dauer der Ausführung berücksichtigt, die Trajektorien und die Aktionssteuerung zu optimieren. Es kann gezeigt werden, dass sich durch das Feedback des Nutzers eine signifikante Reduzierung der Anlerndauer und der Lernzyklen erreichen lässt.

B) Programmieransätze anhand von Augmented Reality

Bischoff et al. [101] untersuchen für den Roboterhersteller KUKA die generellen Anwendungsmöglichkeiten von AR über den gesamten Lebenszyklus eines Industrieroboters. Das potentielle Anwendungsfeld ist weitreichend und erstreckt sich von der Verkaufspräsentation über die Planung, Installation und Inbetriebnahme von Roboterzellen bis hin zur Programmierung und Überwachung robotergestützter Prozesse. Im Bereich des Training und Service liegt potentiell zusätzlicher Nutzen durch eine AR-Unterstützung. Die allgemeine Motivation zum Einsatz von AR in der Roboterprogrammierung liegt darin, durch eine Kombination von realer Roboterzelle und virtuellen CAD-Modellen die Vorteile der Online- und Offline-Programmierung synergetisch miteinander zu verbinden. So kann die reale Roboterzelle in die virtuelle Planung der robotergestützten Aufgabe in Form von klassischen Offline-Programmiersystemen oder Telerobotikanwendungen integriert werden. Weiterführend lässt sich die Online-Programmierung durch virtuelle Darstellung und Simulation unterstützen [115, 116].
Der Forschungsschwerpunkt der AR im Bereich der Robotik lag in der Vergangenheit in der Unterstützung des Nutzers im Bereich der Telerobotik [117, 118, 119]. Der Nutzer interagiert nicht direkt mit dem Robotersystem, sondern steuert den Roboter über einen entfernten Desktoparbeitsplatz. Der Fokus dieser AR-Systeme liegt in der Visualisierung der definierten Posen, Trajektorien und Aufgaben zur räumlichen Evaluation durch den Nutzer. Dabei wurden stets fest stehende Kameras genutzt, welche im Arbeitsraum des Roboters positioniert sind und das Kamerabild über Netzwerk- oder Internetverbindungen übertragen. Boudoin et al. [33] präsentieren ein umfangreiches System der Telerobotik. Dieses umfasst verschiedene räumliche Eingabemethoden unter Verwendung von Datenhandschuhen und 6Dof-Zeigegeräten sowie eine stereoskopische AR-Visualisierung.
Pettersen et al. [120] entwickeln einen ersten Prototypen zur AR-unterstützten Online-Programmierung von Industrierobotern. Die videobasierte Anzeige erfolgt anhand eines HMD und eines tragbaren Computers. Über eine am Kopf des Nutzers befestigte Kamera wird ein markerbehaftetes Zeigegerät zur Definition von Positionen verfolgt. Ein reales Robotersystem wird noch nicht eingebunden. Jedoch wird gezeigt, dass die AR-unterstützte Programmierung einer Beispielaufgabe gegenüber Standardverfahren der Roboterprogrammierung um den Faktor 5 schneller ausgeführt werden kann.
AR ermöglicht jedoch mehr als die räumliche Darstellung des Roboterprogramms in realer Umgebung. Chong et al. [57] übertragen die kinematische Robotersimulation eines Roboterarms in eine

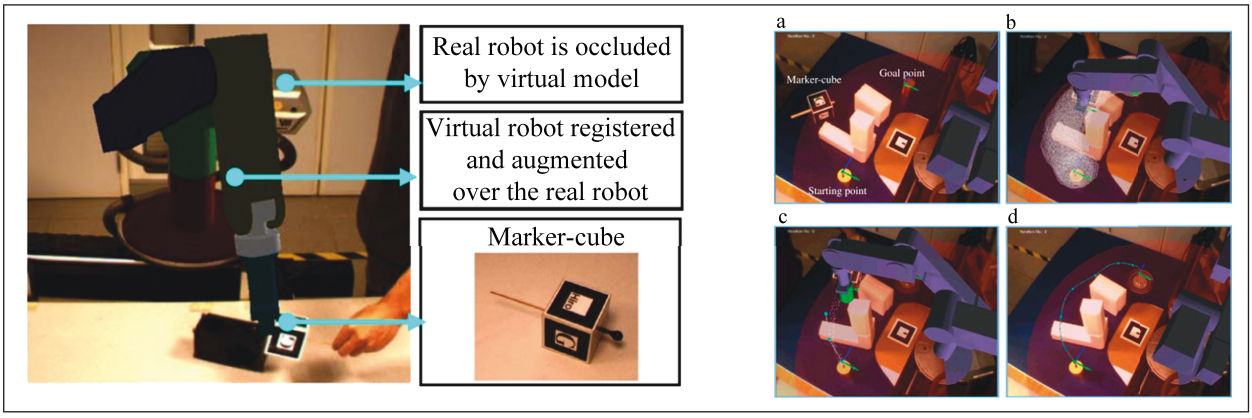

Abbildung 2.19: AR-Simulation eines virtuellen Robotermodells in realer Umgebung (links) und gestenbasierte Definition von kollisionsfreien Bewegungsbahnen (rechts, Schritte a-d) durch Markerwürfel, Quelle: [121, 123].

skalierte und grafisch vereinfachte, markerbasierte AR-Anwendung. Die Umsetzung der AR erfolgt anhand eines videobasierten HMD. Weiterführend erfolgt eine Interaktion des Nutzers durch räumliche Führung eines Markers. Dieser wird zur Definition von kollisionsfreien Trajektorien, bspw. zum Umfahren von realen Hindernissen (s. Abb. 2.19), genutzt [121]. Gleichzeitig wird ein Konzept zur Definition von Schweißbahnen durch die markerbasierte Vorgabe von Start- und Endpunkten entwickelt und evaluiert [122]. Im Vergleich zu klassischen Offline-Programmiersystemen kann hier eine Simulation ohne zwingende Modellierung von Werkstücken oder Kollisionspartnern erfolgen. Zudem wird angedeutet, dass sich durch die Vermischung von Simulation und realer Roboterzelle die Zellkalibrierung, d. h. ein geometrischer Abgleich zwischen Simulation und Realität, vereinfachen lässt. Diese Betrachtungen werden ohne die praktische Einbindung eines realen Robotersystems vollzogen. In einer Weiterentwicklung des beschriebenen Systems wird zunächst die virtuelle Darstellung des Roboters einem realen Robotermodell angepasst. Diese Anpassungen betreffen auch die Skalierung, sodass sich der reale Roboter räumlich mit dem virtuellen Modell im Kamerabild überlagert. Zum Tracking des eingesetzten Markers wird ein Stereo-Kamera-System genutzt, welches gleichzeitig anhand der gelieferten Tiefeninformation zur Objekterkennung dient. Die erkannten Objekte werden zur Kollisionsvermeidung und zur Vermeidung von perspektivischen Verdeckungen herangezogen [121]. Weitere Arbeiten umfassen eine Optimierung des dynamischen Verhaltens des simulierten Roboters. Hierzu werden dynamische Bewegungsparameter des realen Roboters ermittelt und durch mathematische Modellierung des dynamischen Modells in die Simulation integriert [123].

Akan et al. [124] diskutieren den Einsatz multimodaler Kommunikation zur Programmierung von Industrierobotern und identifizieren Gesten und Sprache in Form einer aufgabenorientierten Programmierung als vielversprechende Anwendertechnologie für KMU. Darauf aufbauend wird ergänzend zu einer Offline-Simulation ein Ansatz zur AR-basierten Visualisierung der Robotertrajektorien vorgestellt [125]. Die Kamera der AR-Anwendung ist fest im Arbeitsraum des Roboters positioniert bzw. am Roboterflansch befestigt. An einem Computerarbeitsplatz wird das Kamerabild des Arbeitsraums dargestellt. Um eine aufgabenorientierte Manipulation für Pick-and-Place-Aufgaben vornehmen zu können, wird die räumliche Manipulation von virtuellen Objekten in der AR-Anwendung durch Sprache angestrebt und durch eine räumliche Beschreibungssprache er-

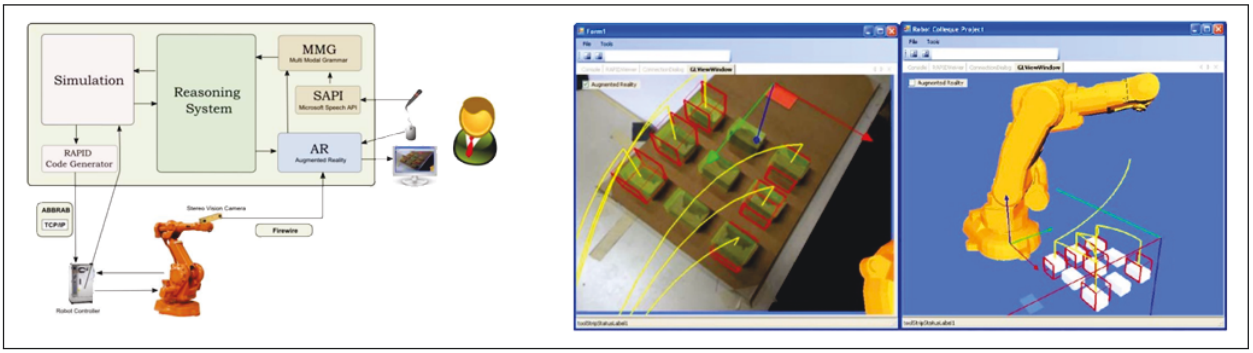

Abbildung 2.20: Multimodales Programmiersystem, Aufbau (links), unter Verwendung von Sprache und einer AR-Visualisierung (rechts), Quelle: [35].

möglicht [126]. Im vorgestellten Gesamtsystem (s. Abb. 2.20) wird der Anwender in die Lage versetzt, durch das schrittweise Verschieben von virtuellen Werkstücken in der desktopbasierten AR-Anwendung Pick-and-Place-Aufgaben zu definieren. Die Interaktion erfolgt anhand von Cursorbewegungen und Sprachbefehlen.

Abbas et al. [127] diskutieren in einem Konzept-Paper zunächst die Nachteile der Teach-In-Programmierung mittels gängiger Handbediengeräte. Es wird die Nutzung von Smartphones mit integrierten AR-Anwendungen zur Steuerung und Programmierung vorgeschlagen. Ferner wird angestrebt, die Programmierung durch Teach-In an einem virtuellen Robotermodell in realer Umgebung auszuführen. Hierzu werden verschiedene dialogbasierte Interaktionstechniken auf dem Smartphone vorgeschlagen. Die Veröffentlichung geht jedoch über eine Konzeptvorstellung nicht hinaus.

J. A. Neuhöfer [128] untersucht die Eignung der AR-basierten Simulation als Unterstützung des menschlichen Anwenders im Bereich der Mensch-Roboter-Kooperation mit geteilten Arbeitsbereichen. Eine umfangreiche Studie stellt fest, dass die Sicherheit und Effektivität der Kooperation durch den Einsatz von AR- und VR-Simulationstechniken gesteigert werden können. Die quantitativen Ergebnisse der Versuchsdurchführungen liefern keine Unterscheidung der Effektivität zwischen AR und VR. Von den Probanden wird jedoch AR bevorzugt.

3 Zielstellung

3.1 Analyse der Ausgangssituation

Aufbauend auf dem Stand der Technik und allgemeinen Tendenzen bei KMU zu kleineren Losgrößen und steigender Variantenvielfalt bleibt zunächst festzustellen, dass die zurzeit in der Industrie eingesetzten Roboterprogrammierverfahren den allgemeinen Anforderungen hinsichtlich einfacher Anwendbarkeit, Effizienz und Effektivität nur unzureichend genügen. Diese Defizite betreffen vorrangig die prozessnahe Programmierung. Im Vergleich zu innovativen räumlichen Benutzerschnittstellen können ein mangelnder Grad an Direktheit der Interaktion sowie eine mangelnde Nutzung natürlicher Interaktionsmodalitäten identifiziert werden.
Die Definition und Manipulation von räumlichen Objekten wie Posen, Bewegungsbahnen oder Aufgaben stellt bei sämtlichen Programmierverfahren einen Hauptanteil des Programmierprozesses dar. Bei den Standardverfahren werden diese räumlichen Objekte aus Sicht des Anwenders unnötig abstrahiert, indem sie bspw. in eine textuelle oder grafisch unzusammenhängende Form übertragen werden. Die eingesetzten Hilfsgeräte und Funktionen zur Ein- und Ausgabe sind auf diese Art der Repräsentation des Roboterprogramms zugeschnitten und vermögen in der Regel nur einen begrenzten Zusammenhang mit den tatsächlichen Bewegungen und Aktionen des Roboters herzustellen.
Dementgegen bietet sich die direkte räumliche Interaktion des Nutzers in der Roboterprogrammierung an, da es sich bei dem Inhalt des Roboterprogramms größtenteils um Informationen mit räumlichem Kontext handelt. Spätestens beim Abfahren des Programms erfahren die Informationen eine reale räumliche Abbildung. Für Ansätze der räumlichen Interaktion ist der räumliche Kontext somit inhärent gegeben und muss nicht künstlich oder metaphorisch hergestellt werden.

3.1.1 Kritische Betrachtung prozessnaher Standardverfahren der Industrieroboterprogrammierung

Beim Teach-In stellt das Anfahren der angestrebten Posen einen anschaulichen Demonstrationsvorgang dar. Ausgehend vom Paradigma der natürlichen Interaktion ist es hinderlich, dass zur Durchführung Knöpfe auf einem Bediengerät gedrückt werden müssen. Ferner müssen die angefahrenen Posen nachträglich in einen textuellen Programmablauf in herstellerspezifischer Programmiersprache integriert werden. Die Playback-Programmierung bietet einen direkteren Ansatz der Interaktion durch das haptische Führen des Roboters. Die automatische Programmgenerierung auf Basis des haptischen Demonstrationsprozesses bewirkt eine weiterführende entscheidende Ver-

einfachung der Programmierung. Ein praktischer Nachteil ist die unzureichende Übertragbarkeit dieser Programmiermethode auf beliebige Robotersysteme sowie die eingeschränkte Anwendbarkeit auf Robotersysteme mit großen Arbeitsräumen. Da die Geschwindigkeiten beim Führen des Roboters aus Sicherheitsgründen beschränkt werden müssen[1], lassen sich weit entfernte Posen oder lange Bewegungsabschnitte nur unter relativ hohem Zeitaufwand und schlechten ergonomischen Randbedingungen definieren. Ein weiterer Nachteil dieser Methode liegt in der mangelnden räumlichen Manipulierbarkeit des Programms. Einzelne Posen und Bewegungsabschnitte lassen sich zwar grundsätzlich entfernen und durch neue haptische Interaktion ersetzen, dies entspricht jedoch einer Neudefinition anstatt einer direkten Manipulation.
Ein grundsätzlicher Nachteil aller gängigen Online-Programmierverfahren liegt in der fehlenden Simulationsunterstützung. Programmierfehler und die mangelnde Erreichbarkeit angestrebter Posen werden meist erst beim realen Abfahren des Programms erkannt. Die Integration von unterstützenden Simulationslösungen bildet hier einen vielversprechenden Ansatz. In [74] wird eine Integration von Offline-Programmiersystemen in die Zellumgebung vorgeschlagen. Dieses Vorgehen kann noch als unzureichend angesehen werden, da als Voraussetzung ein statischer Computerarbeitsplatz genutzt werden muss. Erstrebenswert ist eine mobile, ubiquitär anwendbare Simulationslösung, welche den Nutzer nicht in seiner freien Bewegungs- und Interaktionsfähigkeit einschränkt. Zusammenfassend ergeben sich folgende allgemeine Defizite der gängigen Online-Programmierverfahren:

- Die Verfahren verfügen über eine mangelnde Direktheit der Interaktion für die Definition und Manipulation von Posen, Trajektorien und Aufgaben.
- Natürliche Kommunikation wird nicht oder unzureichend genutzt.
- Die Unterstützung des Anwenders durch räumliches Feedback und eine prozessnahe Simulation ist nicht gegeben.

3.1.2 Kritische Betrachtung ausgewählter Forschungsansätze

Aufseiten der Forschung und Entwicklung zur intuitiven Programmierung wurden in den letzten Jahren neuartige Programmiermethoden und -geräte entwickelt, welche zumeist hinter dem technischen Stand moderner räumlicher Benutzerschnittstellen zurückbleiben oder deren Potentiale nur bedingt ausnutzen. Dies betrifft vorrangig die Kriterien der räumlichen sowie natürlichen Kommunikation, der Mobilität und Ubiquität der Programmiermethoden sowie deren allgemeine Übertragbarkeit auf beliebige Robotersysteme oder Produktionsaufgaben. Zur Ableitung einer detaillierten Zielsetzung für die vorliegende Arbeit sollen hinsichtlich dieser und erweiterter Kriterien Defizite an fortgeschrittenen Forschungsansätzen zur Mensch-Maschine-Interaktion in der Roboterprogrammierung identifiziert werden. Betrachtet werden dementsprechend die räumliche Interaktion in der CAVE sowie umfangreiche Forschungsansätze zur räumlichen Interaktion basierend auf Gesten und AR.

[1] Vgl. DIN EN ISO 10218-1:2012-01 [129].

Räumliche Programmieransätze der VR in der CAVE ermöglichen je nach Ausführung einen hohen Grad an Immersion und natürlicher Interaktion. Die Mobilität dieser Verfahren und ihre Anwendung am realen Robotersystem sind jedoch nicht gegeben. Zudem übersteigen die Kosten derartiger Systeme ein Vielfaches des Preises für ein Standardindustrierobotersystem, sodass sich für KMU in der Regel eine mangelnde Wirtschaftlichkeit ergibt.
Akan et al. [35] nutzen eine räumliche Visualisierung der Trajektorien und eine Simulation in der realen Umgebung des Roboters mittels einer AR-Anwendung. Diese Art der Simulation bietet durch die Verschmelzung von Virtualität und Realität ebenfalls einen hohen Grad an Immersion, lässt sich jedoch im Vergleich zur CAVE kostengünstig umsetzen. Die Visualisierung und Interaktion sind im aufgezeigten Forschungsansatz allerdings an einen festen Computerarbeitsplatz gebunden. Zudem ist die Kamera zur Realisierung der AR nicht mobil, sondern fest im Arbeitsraum des Roboters angebracht. Demnach kann dieser Ansatz der Telerobotik zugeschrieben werden. Des Weiteren wird zwar eine aufgabenorientierte Definition von Pick-and-Place-Aufgaben ermöglicht, Defizite liegen indes in der Eingabe über Cursor- und Mausbewegungen. Eine koordinatengebende Eingabemethode der räumlichen Programmierung sowie natürliche Interaktionsmethoden werden nicht betrachtet.
Chong et al. [57, 121] nutzen eine mobile AR-Anwendung anhand eines HMD zur räumlichen Definition von Bewegungsbahnen über einen handgeführten Marker. Dieser Marker wird mithilfe der Kamera des HMD bzw. eines externen Stereo-Kamerasystems verfolgt. Neben der Visualisierung der Bewegungsbahnen umfasst der Ansatz eine kinematische Simulation der Bewegungsbahnen durch einen virtuellen Roboter, welcher über einen realen Roboter gelegt wird. Das vorgestellte Programmiersystem ist prinzipiell mobil und somit als prozessnahe Programmiermethode einsetzbar. Ermöglicht wird die Definition von Bewegungsbahnen über gestenbasiertes Vormachen. Dieser Ansatz beschreibt eine intuitive, natürliche Methode der Definition von Bewegungsbahnen, wenn auch unter Verwendung des Markers als künstliches Hilfsmittel. Das vorgestellte System ist allerdings auf die Definition von Bewegungsbahnen beschränkt. Räumliche Manipulationen der Bahnen sowie eine komplexere Programmverwaltung und Schnittstellen zu gängigen Industrieroboterherstellern werden nicht betrachtet. In [122] wird zwar eine Übertragung der gestenbasierten Definition von Bewegungsbahnen auf das Anwendungsfeld des Bahnschweißens diskutiert, praktisch allerdings nicht verfolgt.
W. Vogl [96] stellt ein umfangreiches räumlich-interaktives Programmiersystem vor. Neben der gestenbasierten Definition von Trajektorien im Arbeitsraum des Roboters über einen Eingabestift und der Visualisierung der Bahnen in der AR werden projektionsbasierte 2D-Displays zur Manipulation des Roboterprogramms verwendet. Darüber hinaus wird eine kinematische Simulation des Programms durch einen virtuellen Roboter durchgeführt. Ein Kritikpunkt an diesem Ansatz besteht in der beschränkten Mobilität. Das Kamerasystem ist stationär angebracht, das Bild wird über das Remote-Desktop-Verfahren per Wireless Local Area Network (WLAN) auf ein Handheld-Gerät übertragen. Derart lässt sich keine flexible Darstellung des Programms aus Sicht des Nutzers realisieren. Die Manipulation der Trajektorien erfolgt durch eine räumliche Anwahl über den Eingabestift und das Betätigen von unterschiedlichen Tasten am Stift. Trajektorien lassen sich derart über eine Drag-and-Drop-Metapher räumlich verschieben. Diese intuitive Form der Manipulation wird lediglich für projizierte Trajektorien auf Oberflächen realisiert. Trajektorien im freien Raum

werden nicht näher betrachtet. Die Interaktionsmethodik über den Stift und die Betätigung von Tasten offenbart somit weitere Verbesserungspotentiale in der Direktheit und Natürlichkeit der Interaktion. Zur Realisierung des Stift-Tracking wird dieser mit aktiven LEDs ausgestattet, die über Zeilenkameras detektiert und verfolgt werden. Dieses Vorgehen stellt zum einen hohe Anforderungen an die Kalibrierung des Systems. Zum anderen stellt das eingesetzte Sensorsystem einen nicht unerheblichen monetären Aufwand dar, der ggf. von KMU nicht erbracht werden kann.
Es können zusammenfassend folgende spezielle Probleme der aktuellen Forschungsansätze festgestellt werden:

- Die Programmiermethoden verfügen über mangelnde Mobilität und Ubiquität,
- über mangelnden Umfang der Programmiersysteme hinsichtlich Funktionen (Programmverwaltung, implizite Programmierung), Konfiguration und Übertragbarkeit sowie
- unzureichende Wirtschaftlichkeit.

3.1.3 Vergleichende Auswertung

In Tabelle 3.1 erfolgt aufbauend auf der vorangegangenen problemspezifischen Analyse eine zusammenfassende Darstellung der Eigenschaften der Standardprogrammierverfahren sowie der beschriebenen Forschungsansätze. Zur Gegenüberstellung wird näherungsweise eine Einordnung der aufgezeigten Kriterien in verschiedene Kategorien von „nicht erfüllt“ bis „voll erfüllt“ vorgenommen. Es lässt sich feststellen, dass räumliche Benutzerschnittstellen unter Verwendung von Gesten und Augmented Reality das grundlegende Potential besitzen, diese Defizite zu kompensieren, bestehende Forschungsansätze dieses Potential jedoch nicht voll ausnutzen. Im Folgenden dienen die aufgezeigten Defizite zur Definition einer Zielstellung für die vorliegende Arbeit.

	TEACH-IN	PLAYBACK	AKAN ET AL.	CHONG ET AL.	VOGL
Direktheit der räumlichen Interaktion (Definition und Manipulation)	◐	●	◐	◐	◔
Natürliche Kommunikation	○	●	○	◔	◔
Feedback und Simulation	○	○	●	●	●
Mobilität und Ubiquität	●	○	○	●	◐
Vollständigkeit des Programmiersystems	●	◐	●	○	●
Wirtschaftlichkeit	◐	◐	●	●	○

● voll erfüllt ◔ teilweise/voll erfüllt ◐ teilweise erfüllt ○ nicht erfüllt

Tabelle 3.1: Gegenüberstellung verschiedener Programmierverfahren nach defizitären Kriterien: Standardverfahren und aktuelle Forschungsansätze.

3.2 Zielsetzung

Ziel dieser Arbeit ist die Entwicklung eines prozessnahen natürlich-räumlichen Programmiersystems für Industrieroboter. In Abgrenzung zum derzeitigen Stand der Technik soll der Schwerpunkt der Arbeit auf der Entwicklung eines adäquaten Interaktionskonzepts zur räumlichen Programmierung über markerlose Gesten sowie einer prozessnahen Simulationsunterstützung mittels Augmented Reality liegen. Beide Modalitäten sollen in diesem Zusammenhang synergetisch miteinander kombiniert werden. Ferner wird ein mobiler, ubiquitärer Einsatz des Programmiersystems mit der Möglichkeit der aufwandsminimierten Übertragbarkeit auf beliebige Industrierobotersysteme angestrebt. In weiterer Abgrenzung soll im Interesse von KMU auf kostenintensive Systemkomponenten nach Möglichkeit verzichtet werden.
Zur Ausgestaltung der gestenbasierten Interaktion soll das PbD-Paradigma dienen, sodass einzelne Posen, komplexe Trajektorien und Aufgaben durch den Nutzer über natürliche Bewegungen, bspw. der Hand, in Form eines Demonstrationsprozesses definiert werden. Neben der Bereitstellung von mobilem Feedback und einer Simulationsunterstützung soll eine räumliche Manipulation der Programme ermöglicht werden. Des Weiteren sollen im Hinblick auf den Fachkräftemangel Anwender mit unterschiedlicher Qualifikation individuell unterstützt werden. Zusammenfassend bedeutet dies, dass ein natürlich-räumliches Programmiersystem zur Definition, Evaluierung und Anpassung von Roboterprogrammen entwickelt werden soll. Zu diesem Zweck wird eine zentrale Forschungshypothese formuliert:

> *Es ist möglich, ein räumliches System zur prozessnahen Programmierung von Industrieroboter unter Nutzung von natürlicher Kommunikation auf Basis von Gesten und Augmented Reality zu gestalten und als kostengünstigen Systementwurf prototypisch umzusetzen.*

Bezüglich der Konkretisierung der zentralen Forschungshypothese lässt sich eine Aufschlüsselung in detaillierte Teilfragen anhand des Nutzungskontexts[2] vornehmen. Zu diesem Zweck werden die Funktionsbereiche der Mensch-Maschine-Interaktion, der Programmierumgebung sowie der Schnittstellen und Inbetriebnahme unterschieden. Weitere Betrachtungen gelten der benutzergerechten Auslegung einzelner Funktionen.

Mensch-Maschine-Interaktion:

- Wie kann ein natürlich-räumliches Interaktionskonzept gestaltet werden und welche Modalitäten und Funktionen sind bereitzustellen?
- Kann eine wirtschaftliche Umsetzung des Interaktionskonzepts hinsichtlich des Einsatzes speziell für KMU erfolgen?

Programmierumgebung:

- Welche grundlegenden Funktionen muss das Programmiersystem bereitstellen?
- Inwiefern kann das Programmiersystem bedarfsgerecht modularisiert werden?

[2] Vgl. DIN EN ISO 9241-11:1999-01 [16].

Schnittstellen und Inbetriebnahme:

- Wie lässt sich eine breite Anwendbarkeit des Programmiersystems auf Industrierobotersysteme verschiedener Hersteller realisieren?
- Wie kann das Programmiersystem in die bestehende informationstechnische Infrastruktur im Unternehmen eingebunden werden?
- Wie lässt sich die Inbetriebnahme des Programmiersystems realisieren?

Benutzergerechte Auslegung:

- Wie können Anwender mit verschiedener Qualifikation in der Industrieroboterprogrammierung durch adäquate Funktionsbereitstellung individuell unterstützt werden?

3.3 Vorgehen

Im weiteren Verlauf dieser Arbeit soll zunächst eine Konzeption des Programmiersystems zur grundlegenden Beantwortung der aufgezeigten Forschungsfragen erstellt werden. Dazu werden im Rahmen einer Anforderungsanalyse Teilziele aus der aufgeschlüsselten Forschungsfragestellung abgeleitet. Die Teilziele dienen im Folgenden zur Erstellung eines Interaktionskonzepts, welches anschließend durch einen Entwurf des Programmiersystems konkretisiert wird. Auf Basis des Entwurfs erfolgt die Beschreibung der prototypischen Umsetzung. Anschließend erfolgt die Erprobung anhand einer speziellen aufgabenorientierten Umsetzung für Pick-and-Place-Aufgaben. Eine Evaluation wird mithilfe einer Nutzerstudie und einer technologischen Bewertung der aktuellen Umsetzung vorgenommen. Abschließend wird im Rahmen der Zusammenfassung eine Überprüfung der zentralen Forschungshypothese vorgenommen. Zudem wird ein Ausblick auf die industrielle Anwendbarkeit des Programmiersystems gegeben.

4 Konzeption des natürlich-räumlichen Programmiersystems für Industrieroboter

In diesem Kapitel wird gemäß der Zielstellung dieser Arbeit eine Konzeption für das angestrebte natürlich-räumliche Programmiersystem erstellt. Das Vorgehen orientiert sich am Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme[1]. In Abschnitt 4.1 werden zunächst Teilziele abgeleitet. Um diese zu erreichen, wird in Abschnitt 4.2 ein allgemeines Interaktionskonzept erstellt und dessen methodische Ausgestaltung für die prozessnahe Industrieroboterprogrammierung konkretisiert. In Abschnitt 4.3 erfolgt darauf aufbauend der gestalterische Entwurf des Programmiersystems unter Berücksichtigung von erweiterten Randbedingungen der Industrieroboterprogrammierung. Abschnitt 4.4 fasst abschließend das Interaktionskonzept und den Systementwurf des Programmiersystems kompakt zusammen.

4.1 Anforderungen an das natürlich-räumliche Programmiersystem

Aus der Zielsetzung der Arbeit, genauer aus der aufgeschlüsselten Forschungsfrage, lassen sich umfangreiche Anforderungen ableiten. Hierzu werden zunächst anhand der Anforderungen des Nutzungskontexts detailliertere Teilaufgaben identifiziert und für diese Zielstellungen abgeleitet. Anschließend erfolgt eine nutzerspezifische Anforderungsanalyse in deren Rahmen spezielle Teilziele für die Adressierung einzelner Nutzergruppen aufgestellt werden.

4.1.1 Identifikation von Teilaufgaben

A) Teilaufgaben der Mensch-Maschine-Interaktion

Die Gestaltung der Interaktion umfasst die Bereitstellung adäquater Funktionen, welche eine intuitive und effektive Kommunikation des Nutzers mit dem Programmiersystem hinsichtlich der Durchführung der Programmieraufgabe ermöglichen. Die Interaktion soll demgemäß möglichst natürlich-räumlich und direkt erfolgen. Für die verschiedenen programmiertechnischen Ein- und Ausgaben sind zunächst Modalitäten zu wählen. Da durch die Zielstellung bereits die Nutzung von Gesten und AR vorgesehen ist, ist im Rahmen des Interaktionskonzepts deren Anwendung

[1] Vgl. DIN EN ISO 9241-210:2011-08 [17].

und Auslegung zu konkretisieren. Zudem gilt es zu untersuchen, ob zusätzliche Modalitäten unterstützend eingesetzt werden können. Auf dieser Basis sollen speziell Methoden zur räumlichen Definition und Manipulation von Objekten, welche das Roboterprogramm repräsentieren, entwickelt werden. Diese Methoden sollen durch den Anwender mobil durchgeführt werden können, sodass sich dieser frei in der Roboterzelle bewegen kann.
Aus wirtschaftlicher Sicht stehen der Rentabilität von Industrierobotern Anschaffungs- und Betriebskosten entgegen. Ein Großteil der Betriebskosten eines Roboters wird bei Rekonfiguration der Anwenderaufgabe durch die manuell durchgeführte Programmierung verursacht; diese führt zudem bei prozessnaher Durchführung zu Stillstandszeiten [130]. Mit der Anwendung intuitiver Programmiersysteme wird die Senkung der Betriebskosten angestrebt. Je eher die Amortisation erfolgt, desto höher ist prinzipiell die Rentabilität der Programmiermethode. In diesem Zusammenhang wird die Hypothese aufgestellt, dass hohe Systemkosten vorangegangener Ansätze der intuitiven Industrieroboterprogrammierung einen Grund für deren mangelnde industrielle Verbreitung darstellen. In der Vergangenheit wurde beim Systementwurf vorrangig das Kriterium der Genauigkeiten gesteuert [96]. Die anfallenden Kosten für die Hardware wurden als notwendiges Resultat hingenommen. In Abgrenzung dazu soll im vorliegenden Systementwurf zunächst der Fokus auf die wirtschaftliche Umsetzung des Programmiersystems gelegt werden.
Um die breite Einsetzbarkeit des Programmiersystems bei KMU zu gewährleisten, wird angestrebt, die Interaktionen mit Standardkomponenten zu realisieren. Dies umfasst einerseits den Verzicht auf kostenintensive Hardware und motiviert andererseits zur Verwendung von kostengünstigen Alternativen aus dem Consumer-Bereich. Ein entsprechendes Szenario soll mittels einer prototypischen Umsetzung erprobt und hinsichtlich eventueller Defizite evaluiert werden. Abbildung 4.1 fasst die aufgeführten Teilziele für die Mensch-Maschine-Interaktion kompakt zusammen.

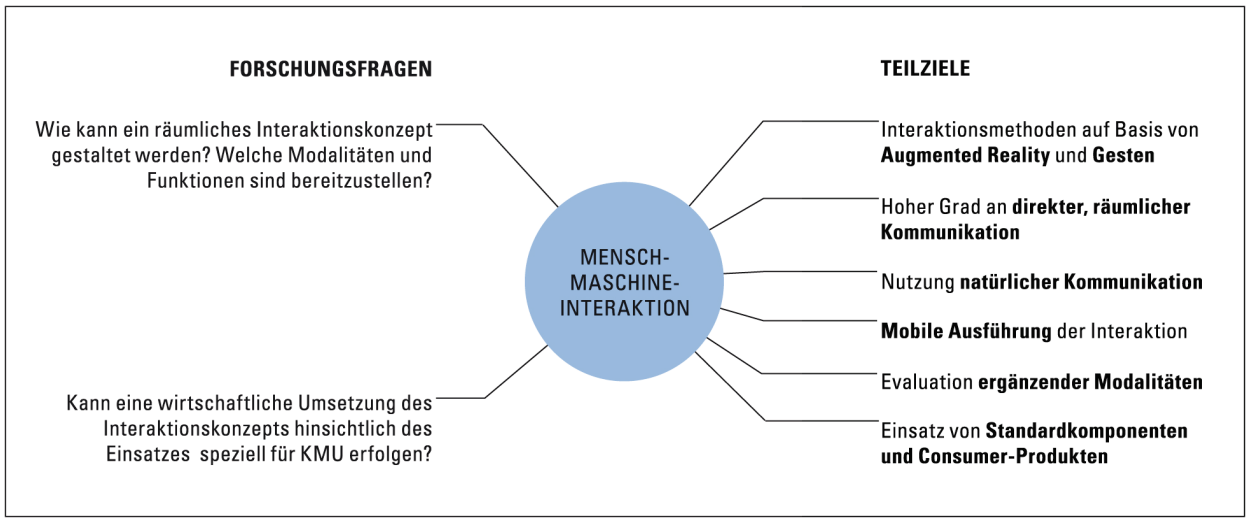

Abbildung 4.1: Definition von Teilzielen für die Gestaltung der Mensch-Maschine-Interaktion.

B) Teilaufgaben der Programmierumgebung

Das angestrebte Programmiersystem setzt die Existenz einer Programmierumgebung mit entsprechenden Funktionen zur Programmverwaltung voraus. Diese Funktionen sollen das Anlegen neuer Programme sowie das Speichern, Laden und Entfernen von Roboterprogrammen ermöglichen. Zudem muss ein Satz gängiger Programmierbefehlen zur Bewegungs- und Aktionssteuerung abgebildet werden. Um eine generelle Übertragbarkeit der Programme zu gewährleisten, wird intern die Wahl einer neutralen, herstellerunabhängigen Spezifikation von Roboterprogrammen angestrebt.
Teilaufgaben der Programmierung umfassen die Definition, Evaluation und Manipulation. Als Grundlage für die Interaktion sind für diese Teilschritte geeignete Programmierfunktionen bereitzustellen. Aus der zentralen Zielstellung geht hervor, dass zur Evaluation eine Simulation der Programme in AR angestrebt wird. Hierzu sind entsprechende Funktionen zur Visualisierung und zur Berechnung der Kinematik bereitzustellen.
Im Rahmen der Modularisierung des Systems soll eine entsprechende Kombinationsfähigkeit einzelner Teilfunktionen des Programmiersystems mit anderen Programmierverfahren vorgesehen werden. Dies erhöht einerseits die Flexibilität des Programmierers und trägt andererseits zur bedarfsgerechten Integration anwendungsspezifischer Programmierfunktionen bei. Ferner wird derart die schrittweise Integration des Programmiersystems in bestehende betriebliche Abläufe zur Erstellung von Industrieroboterprogrammen ermöglicht. Des Weiteren wird die Austauschbarkeit der eingesetzten Hardware, d. h. Anzeigegeräte und Sensorik, hinsichtlich verschiedener Anwendungsszenarien und deren spezifischen Anforderungen, bspw. Genauigkeiten und Prozessbedingungen, gefordert. Abbildung 4.2 visualisiert die Gegenüberstellung von Forschungsfragen und abgeleiteten Teilzielen für den Bereich der Programmierumgebung.

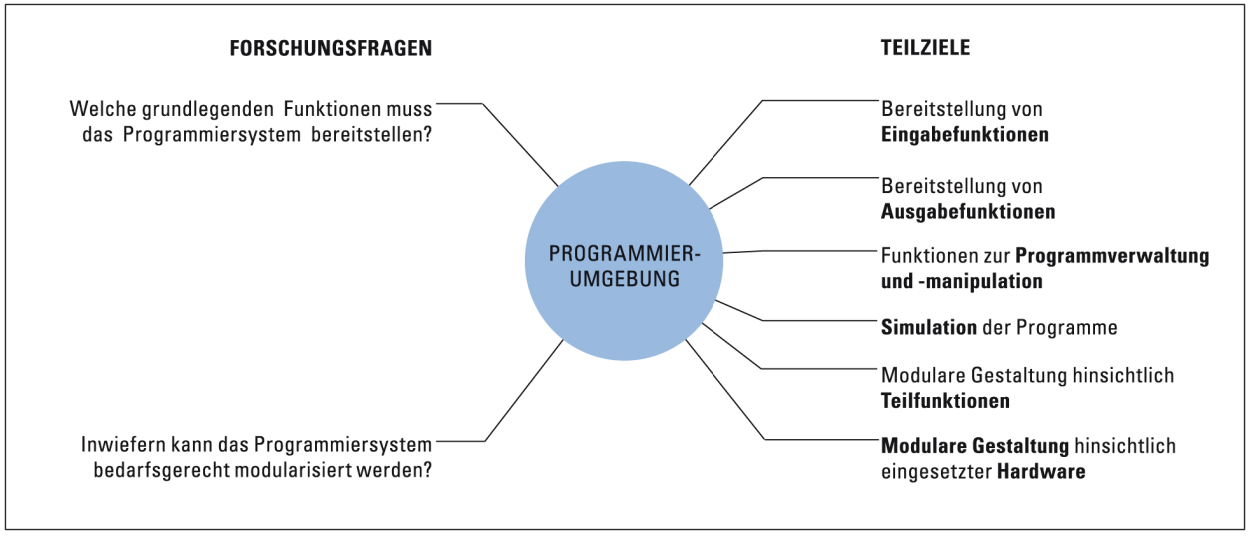

Abbildung 4.2: Definition von Teilzielen für die Programmierumgebung.

C) Teilaufgaben für Schnittstellen und Inbetriebnahme

Die Gestaltung der externen Schnittstelle des Programmiersystems zur realen Robotersteuerung entscheidet über die Übertragbarkeit des Programmiersystems auf Robotersysteme verschiedener Hersteller. Da keine einheitlichen Standards zu Programmiersprachen existieren und die verschiedenen Hersteller über spezifische Schnittstellen mit differierendem Funktionsumfang verfügen[2], stellt die allgemeine Übertragbarkeit des Programmiersystems eine hohe Herausforderung dar. Es ist zu vermuten, dass bestehende Ansätze der intuitiven Programmierung aufgrund mangelnder Übertragbarkeit wenig industrielle Verbreitung finden bzw. dass die fehlende Übertragbarkeit eine Hürde für den Transfer zwischen Forschungsobjekt und Produkt darstellt. Aus diesem Grund wird eine vereinheitlichte Schnittstelle zur Robotersteuerung angestrebt, welche sich mit geringem Aufwand auf Industrierobotersysteme beliebiger Hersteller übertragen lässt. Prinzipiell steigt mit geforderter Flexibilität und Übertragbarkeit der Schnittstelle der Aufwand zu deren Erstellung. Wird eine breite Einsetzbarkeit des Programmiersystems beabsichtigt, müssen Schnittstellenfunktionen ggf. redundant betreffend herstellerspezifischer Varianten ausgelegt werden.
Die Erstellung von Industrieroboterprogrammen ist in der Regel in den allgemeinen betrieblichen Prozess der Produktionsplanung und -steuerung integriert. Hierzu erfolgt eine informationstechnische Anbindung des Programmiersystems an übergeordnete Systeme der Fertigungssteuerung sowie der Ressourcenplanung. Die eingesetzten Systeme besitzen häufig eine durchgängige Datenbasis. Als Oberbegriff dieser Systeme steht die Digitale Fabrik als Netzwerk von digitalen Modellen und Methoden, u. a. der 3D-Visualisierung und Simulation, die durch ein durchgängiges Datenmanagement integriert werden[3]. Um eine flexible Integration des mobilen Programmiersystems in bestehende Infrastrukturen der Digitalen Fabrik bereitzustellen, sind entsprechende bidirektionale Schnittstellen zum Informationsaustausch vorzusehen.
Zum erstmaligen Einsatz des Programmiersystems in neuen Produktionsumgebungen ist eine Inbetriebnahme vorzunehmen. Im Sinne einer wirtschaftlichen Gestaltung der Inbetriebnahme soll der Aufwand hinsichtlich Zeit und Kosten minimal ausfallen. Bezüglich KMU bestehen aufgrund mangelnder Auslastung von Industrierobotern im stationären Betrieb teilweise spezifische Motivationen zum mobilen Einsatz von Industrierobotern [132]. Diese Motivation verstärkt die prinzipielle Relevanz einer effizienten Übertragung bzw. Wiederverwendung des Programmiersystems auf verschiedene Robotersysteme bzw. Produktionsumgebungen. Die Inbetriebnahme des Systems umfasst neben der Installation von Hardwarekomponenten hauptsächlich die Identifikation der räumlich-geometrischen Beziehungen zwischen verschiedenen Komponenten, bspw. Sensorik und Roboter [133]. Die Ermittlung der genauen räumlichen Abhängigkeiten bedarf einer Messung bzw. Kalibrierung der Abstände der verschiedenen Koordinatensysteme der Komponenten. Anschließend müssen die ermittelten Informationen in das Programmiersystem übertragen werden. Um eine effiziente Anpassungsfähigkeit und Übertragbarkeit des Programmiersystems zu erreichen, ist ggf. zu prüfen, ob eine Automatisierung der Kalibrierung möglich ist. In Abbildung 4.3 werden die Teilziele für die Schnittstellen des Programmiersystems und die Inbetriebnahme auf Basis der entsprechenden Forschungsfragen zusammengefasst.

[2] Vgl. Kapitel 2.2.1.
[3] Vgl. VDI/VDE 4499-1:08-02 [131].

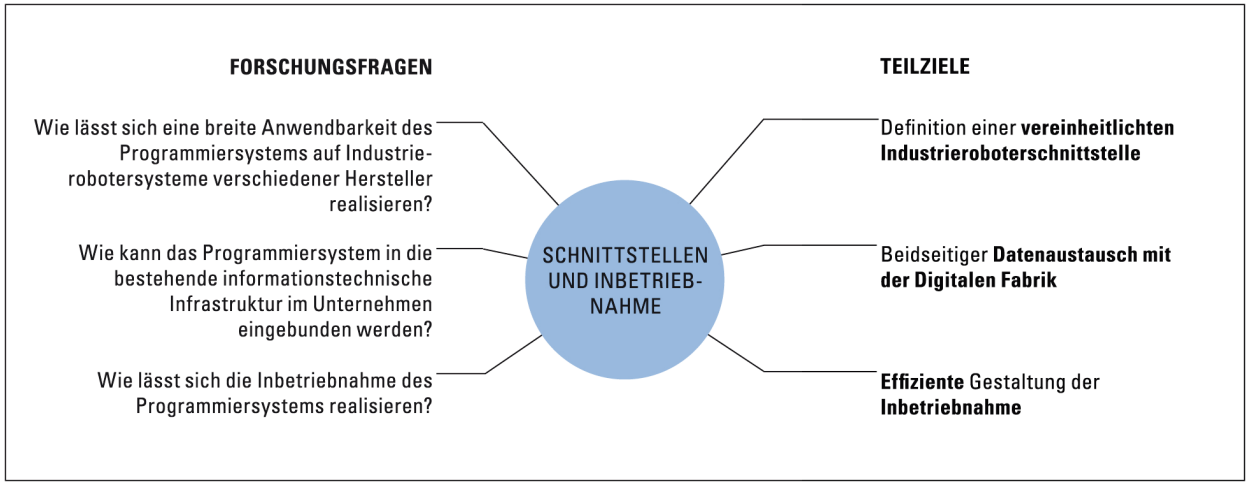

Abbildung 4.3: Definition von Teilzielen für die Schnittstellen und die Inbetriebnahme.

4.1.2 Ableitung benutzergerechter Anforderungen

Die Bestimmung nutzerspezifischer Anforderungen erfolgt in Übereinstimmung mit der Richtlinie VDI 3850-1, welche das Vorgehen zur benutzungsgerechten Gestaltung von Bediensystemen für Maschinen in Anlehnung an DIN EN ISO 9241-210 vorgibt. Nachdem der allgemeine Nutzungskontext durch die vorangegangenen Anforderungen und Teilziele näher konkretisiert wurde, werden Nutzergruppen identifiziert und spezifische Anforderungen zur Gestaltung des Gesamtkonzepts der Interaktion festgelegt. Abbildung 4.4 visualisiert die Ableitung von nutzerspezifischen Teilzielen auf Basis der entsprechenden Forschungsfrage.

A) Benutzergruppen

Als Benutzergruppe sollen sowohl ausgebildete Industrieroboterprogrammierer als auch Nutzer ohne extensives Vorwissen im Bereich der industriellen Robotik adressiert werden. Diese Motivation basiert auf dem allgemeinen Hintergrund des Fachkräftemangels und des demografischen Wandels[4]. Für diese beiden Nutzergruppen lässt sich ein unterschiedlicher Nutzungskontext ableiten.

Nutzungskontext des Roboterprogrammierers
Der „Industrieroboterprogrammierer“ ist vertraut mit dem kinematischen Aufbau und den grundlegenden Steuerungsfunktionen von verschiedenen Industrierobotertypen. Er verfügt über Kenntnisse herstellerspezifischer Bewegungs- und anderer Programmierbefehle und ist prinzipiell in der Lage, diese auf die Syntax einer anderen Roboterprogrammiersprache zu übertragen. Er ist fähig, aus einer produktionstechnischen Aufgabendefinition die Bewegungen und Aktionen des Roboters zur Realisierung der Aufgabe abzuleiten. In diesem Vorgehen ist er geübt und kann diese Aktionen sicher und schnell durchführen.

[4] Vgl. Kapitel 1.

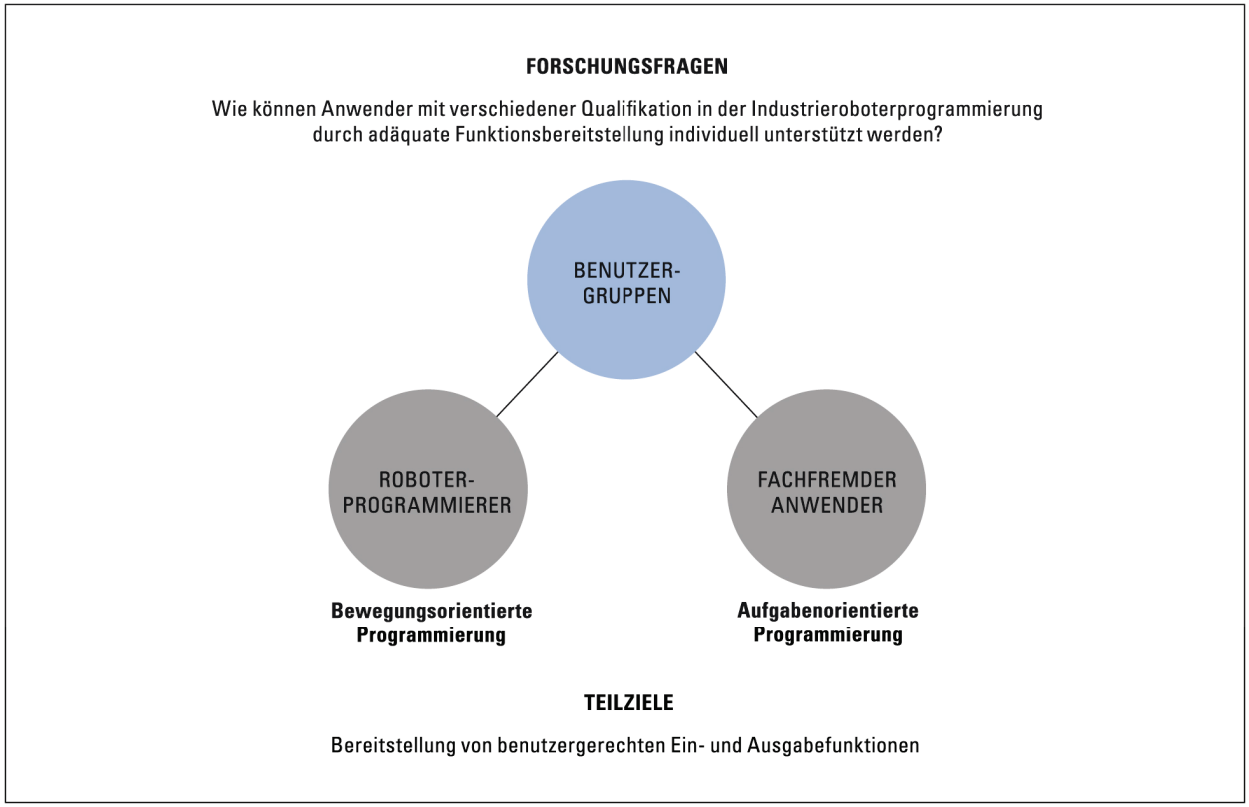

Abbildung 4.4: Definition von nutzerspezifischen Teilzielen für die Gestaltung der Programmierung.

Nutzungskontext des fachfremden Nutzers

Der „fachfremde Nutzer“ verfügt über keinerlei Kenntnisse zur Programmierung von Industrierobotern. Neben Programmiersprachen sind ihm auch die Unterscheidung verschiedener Koordinatensysteme und Interpolationsbefehle fremd. Er ist jedoch in der Lage, die Produktionsaufgabe und deren Anforderungen zu verstehen und diese zu kommunizieren sowie zu imitieren bzw. diese instruktiv anzudeuten.

B) Definition der Nutzungsanforderungen

Zur Realisierung der Programmierung benötigt der Nutzer vom technischen System sowohl Eingabe- als auch Ausgabefunktionen. Nutzerspezifische Anforderungen an diese Funktionen lassen sich aus dem Nutzerkontext herleiten. Daraus ergibt sich, dass zwischen den Nutzergruppen verschiedene Fähigkeiten zur Abstraktion der Aufgabe vorliegen. Der fachfremde Nutzer ist nicht in der Lage, die Aufgabe auf eine roboterspezifische Ebene zu übertragen. Demgemäß müssen ihm aufgabenorientierte Eingabe- und Ausgabefunktionen bereitgestellt werden. Diese Funktionen sollen es ihm ermöglichen, die Aufgabe durch Vormachen oder Andeuten zu definieren.
Der Roboterprogrammierer arbeitet effizient auf der bewegungsorientierten Ebene. Er ist auf dieser Programmierebene in der Lage, komplexe Programme zu erstellen und diese flexibel an spezifische Anforderungen anzupassen. Entsprechend müssen Eingabe- und Ausgabefunktionen für diese Nutzergruppe bewegungsorientiert gestaltet werden.

4.2 Interaktionskonzept zur natürlich-räumlichen Programmierung

Das Interaktionskonzept zielt darauf ab, gemäß den aufgestellten Teilzielen für die Mensch-Maschine-Interaktion, die Nutzung der angestrebten Modalitäten zu präzisieren und einen konzeptionellen Ansatz für deren Umsetzung auf die Roboterprogrammierung bereitzustellen. Laut Zielsetzung sollen für das räumliche Programmiersystem Gesten als Eingabemodalität und Augmented Reality als Ausgabemodalität genutzt werden. Im Folgenden wird die Anwendbarkeit der Modalitäten diskutiert sowie eine Eingrenzung des Anwendungsfokus vorgenommen.

4.2.1 Interaktionsgestaltung und Auslegung der Modalitäten

A) Gestenbasierte Interaktion

Gesten sollen im Programmiersystem zur räumlichen Eingabe und somit zur Definition und Manipulation von Roboterprogrammen eingesetzt werden. Um einen hohen Grad an natürlicher Interaktion herzustellen, soll die Registrierung der Gesten nach Möglichkeit markerlos erfolgen. Irritationen des Nutzers durch zusätzliche tragbare Hilfsmittel werden somit vermieden. Zudem erfolgt derart, aus Sicht des Nutzers, eine Vereinfachung des Verfahrens. Zusätzliche Kosten für etwaige Hilfsmittel wie Marker, Handschuhe etc. werden darüber hinaus vermieden.

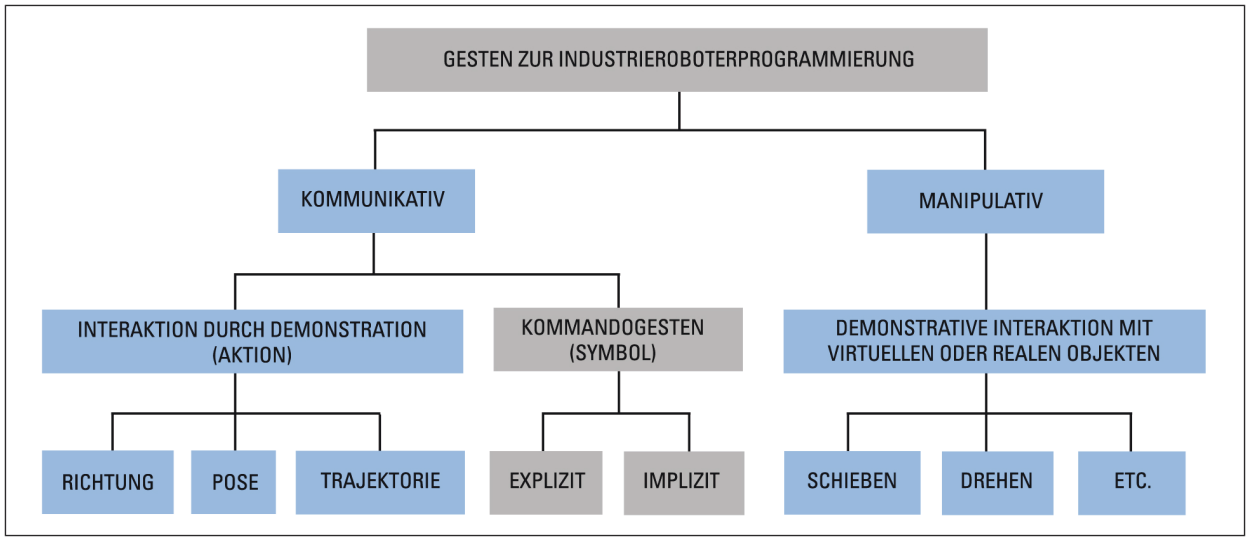

Abbildung 4.5: Taxonomie von 3D-Gesten für den Einsatz in der Steuerung und Programmierung von Industrierobotern.

Basierend auf der Einteilung der Handgesten zur Mensch-Computer-Interaktion nach Pavlovic[5] sind in Abbildung 4.5 generelle Anwendungsmöglichkeiten der gestenbasierten Eingabe zur Steuerung und Programmierung von Industrierobotern aufgeführt. Für kommunikative Gesten bestehen

[5] Vgl. Kapitel 2.1.3.

Hauptanwendungsgebiete als Kommando- sowie als Handlungsgesten zur Bewegungsführung des Roboters. Bei der Bewegungsführung soll begrifflich und inhaltlich zwischen manueller Bewegungssteuerung und manueller Bewegungsvorgabe unterschieden werden.

- **Bewegungssteuerung:** Die Roboterbewegungen erfolgen simultan zur Bewegung des Anwenders. Die Definition von Bewegungsbahnen oder Posen erfolgt relativ, ausgehend von einer Startposition. Der Roboter folgt der Bewegung des Anwenders.
- **Bewegungsvorgabe:** Der Nutzer definiert die räumlichen Parameter des Roboterprogramms absolut. Dazu befindet er sich im Arbeitsraum des Roboters. Die Übertragung der Bewegungen auf den Roboter findet nach Beendigung der Interaktion statt.

Als weiteres Anwendungsszenario lassen sich Steuerungsbefehle aus der gestenbasierten Manipulation von Objekten ableiten. Im Folgenden wird die Eignung der verschiedenen Anwendungsmöglichkeiten von Handgesten im räumlichen Programmiersystem näher untersucht. Eine ausführliche Diskussion verschiedener markerloser gestenbasierter Interaktionsprinzipien in der Roboterprogrammierung erfolgte durch den Autor der vorliegenden Arbeit in [134].

Kommandogesten
Symbolische Kommandogesten lassen sich zum Auslösen spezifischer Steuerungsfunktionen einsetzen. In Abhängigkeit von einem offensichtlichen inhaltlichen Kontext zwischen räumlichem Symbol der Geste und Steuerungsfunktion soll zwischen impliziten und expliziten Kommandogesten unterschieden werden. In der Industrieroboterprogrammierung lassen sich derart bspw. einzelne Programmkommandos des Roboters durch Kommandogesten erstellen, manipulieren oder löschen. Weiterführend könnten Parameter wie Geschwindigkeiten oder Beschleunigungen durch Kommandogesten vorgegeben oder manipuliert werden. Eine weitere mögliche Anwendung liegt im gestenbasierten Programmaufruf zum Starten und Stoppen der automatischen Ausführung bestimmter Aufgaben (s. Abb. 4.6).

Bewegungssteuerung
Grundlegende Standardfunktionen der Bewegungssteuerung von Industrierobotern beinhalten die Bewegung einzelner Achsen und des Endeffektors in verschiedenen kartesischen Koordinatensystemen. Diesbezüglich kann eine Zeigegeste der Hand in eine bestimmte Richtung zur Steuerung der Bewegung einer Einzelachse oder des Endeffektors in die entsprechende Richtung genutzt werden. Da sich der Roboter simultan zur Interaktion des Menschen bewegt, erfolgt ein simultanes Feedback, welches anhand von Hand-Auge-Koordination eine schnelle Anpassung der Pose des Roboters durch den Anwender ermöglicht (s. Abb. 4.7). Wird dieses Verfahren um eine zusätzliche Funktion zum Speichern der angefahrenen Posen erweitert, entspricht diese Funktionalität dem Handverfahren des Industrieroboters über das Handbediengerät beim Teach-In.
Eine komplexere Bewegungssteuerung lässt sich durch simultane Imitation der Bewegungsbahnen der Hand realisieren. Derart kann der Endeffektor ausgehend von einer Startposition auf flexiblen Bahnen frei gesteuert werden. Zum Zweck der Programmierung lassen sich die Bahnen des Roboters aufzeichnen und in Form eines Roboterprogramms reproduzieren. Wahlweise kann auch das Geschwindigkeitsprofil der menschlichen Bewegung direkt auf den Roboter übertragen werden.

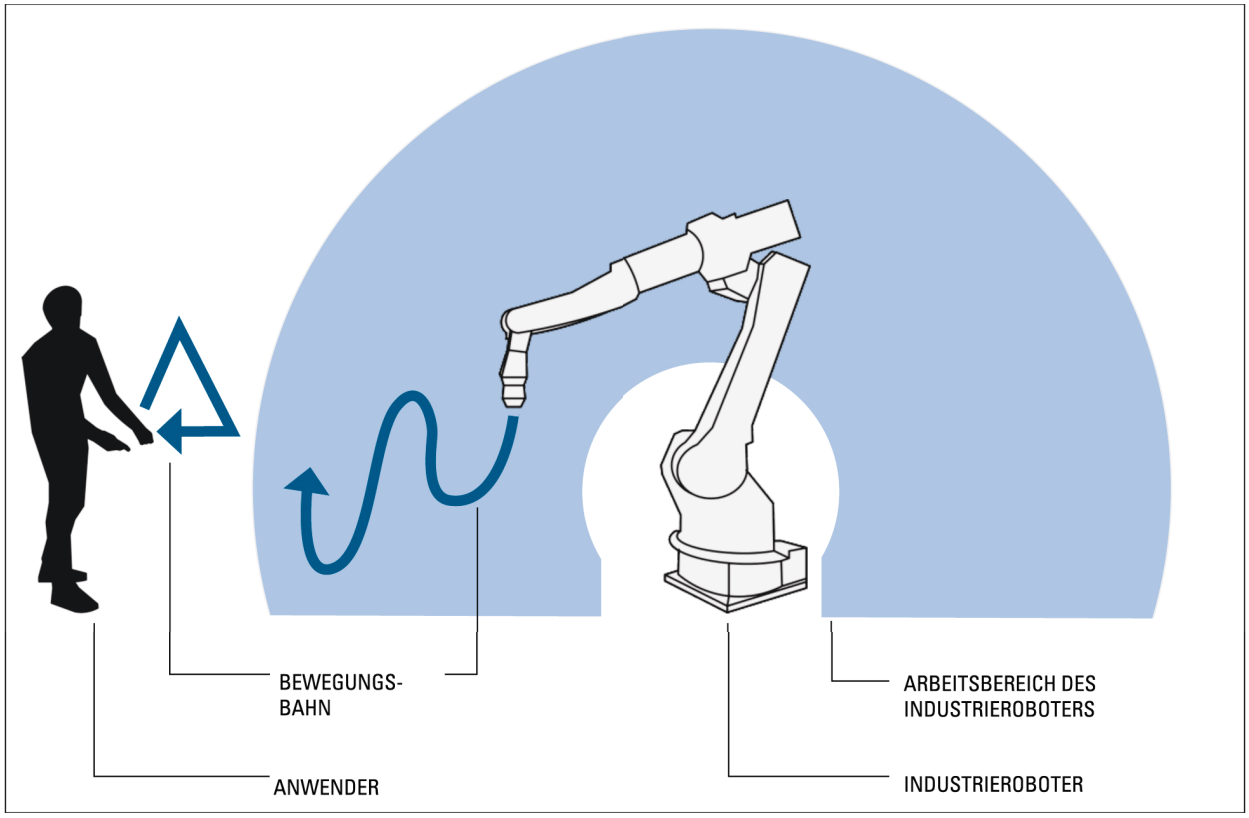

Abbildung 4.6: Mögliche Ausführung von Kommandogesten (hier: Dreieck) zum Start einer programmierten Roboterbewegung.

Bewegungsvorgabe

Die Bewegungsvorgabe bezeichnet in Abgrenzung zur Bewegungssteuerung eine absolute Vorgabe räumlicher Programmparameter. Der Industrieroboter ist während der räumlichen Definition nicht aktiv, sodass sich der Anwender im Arbeitsbereich des Roboters frei bewegen kann. Durch Gesten werden Posen, Trajektorien und Aufgaben als räumliche Objekte an Stelle ihrer angestrebten Ausführung definiert. Die Posen und Trajektorien des Anwenders lassen sich aufzeichnen und anschließend in ein Roboterprogramm übertragen. Nach abgeschlossener Interaktion entfernt sich der Nutzer aus dem Arbeitsbereich des Roboters und das Roboterprogramm zur Reproduktion der Demonstration wird gestartet. Das Prinzip wird in Abbildung 4.8 am Beispiel der Definition einer komplexen Trajektorie verdeutlicht. Bei fehlender Evaluationsmöglichkeit des Programms, bspw. textuell oder durch Simulation, werden Fehler erst bei der Ausführung des Programms erkannt. Das Vorgehen der Bewegungsvorgabe entspricht einem PbD-Ansatz mit einem hohen Grad an Direktheit und Anschaulichkeit, da Bewegungen an beabsichtigter Stelle im Raum über natürliche Bewegungen vorgegeben werden. Voraussetzung für die absolute Definition der Bewegungsparameter ist jedoch die Zugänglichkeit des angestrebten Interaktionsbereichs für den Anwender.

Manipulative Interaktion mit räumlichen Objekten

Die manipulative Interaktion mit physischen oder virtuellen Objekten bietet die Möglichkeit der räumlichen Anpassung des Roboterprogramms. Diesbezüglich besteht die Voraussetzung, dass die Objekte das Roboterprogramm auf einer mehr oder weniger abstrahierten Ebene repräsentieren. Das räumliche Verschieben und Verdrehen von Objekten stellt auf der bewegungsorientierten Ebe-

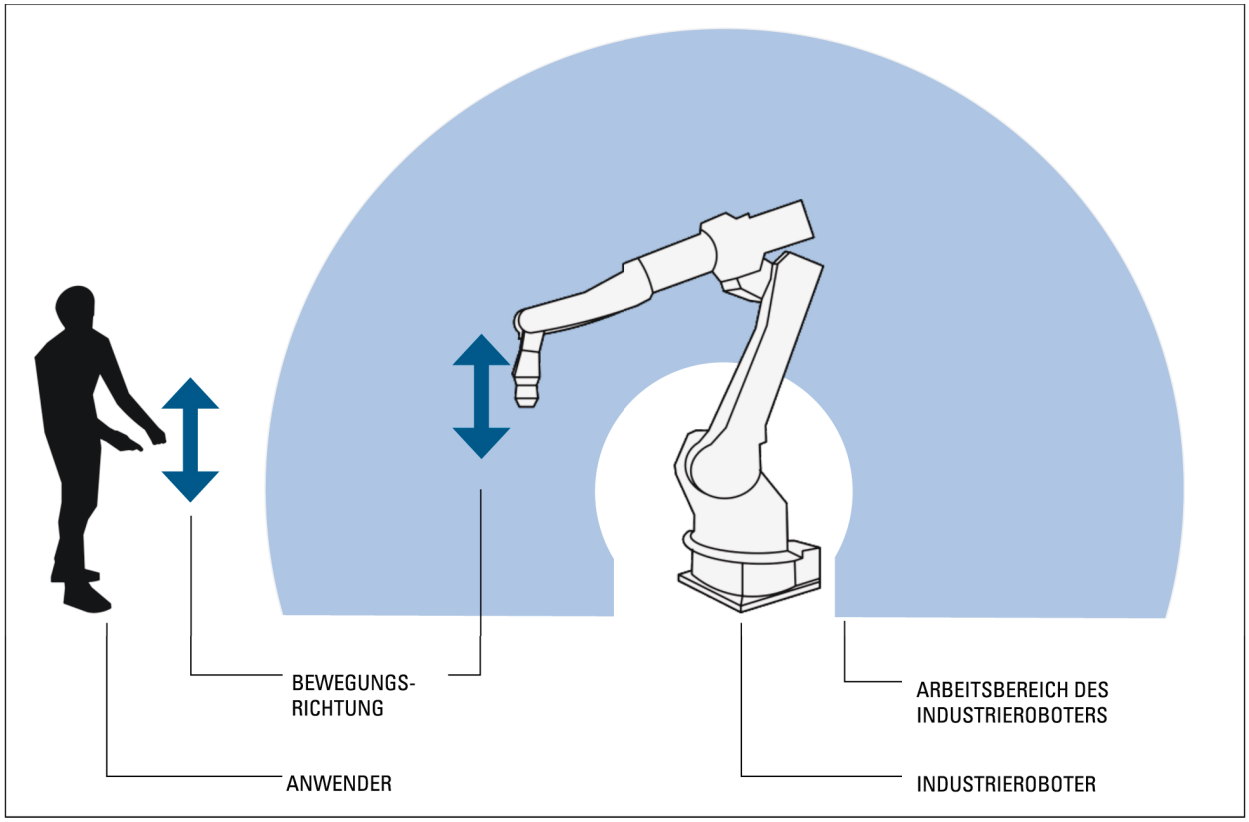

Abbildung 4.7: Bewegungssteuerung durch Vorgabe der Bewegungsrichtung für den Endeffektor.

ne eine direkte Interaktionsform zur Manipulation von Posen und Trajektorien dar. Weiterführend können aus einer räumlichen Manipulation Aufgaben, die vom Roboter zu imitieren sind, abgeleitet werden.

Anwendungsrestriktionen gestenbasierter Bewegungssteuerung und Kommandogesten
Auf die Verwendung von symbolischen Kommandogesten sowie auf eine gestenbasierte Bewegungssteuerung des Industrieroboters soll im vorliegenden Programmiersystem verzichtet werden. Praktische Untersuchungen des Autors der vorliegenden Arbeit zur Erkennung von symbolischen Kommandogesten haben gezeigt, dass sich räumliche Kommandogesten mit kostengünstiger Sensorik robust erkennen und individualisieren lassen [134]. Der Nachteil dieser Eingabemethode liegt jedoch darin, dass einzelne Gesten wie Vokabeln durch den Anwender erlernt werden müssen. Durch einen symbolischen Kontextbezug von Trajektorie und verbundener Steuerungsfunktion lässt sich dieser Lernprozess erleichtern. Die notwendige Assoziation stellt jedoch bereits bei einer übersichtlichen Anzahl möglicher Kommandos eine hohe Herausforderung dar, die die Vorteile der natürlichen Ausführung dieser Eingabemethode weitestgehend kompensiert. Potentielle Anwendungsgebiete symbolischer Gesten liegen in komplexen Kooperationsszenarien mit hohem Abstimmungsbedarf zwischen Mensch und Roboter. Krüger et al. [135] präsentieren entsprechende Szenarien der Mensch-Roboter-Kooperation unter gemeinsamer Nutzung von Arbeitsplätzen.
Zur kritischen Bewertung der industriellen Einsetzbarkeit der Bewegungssteuerung werden die Normen zu Sicherheitsanforderungen an Industrieroboter DIN EN ISO 10218-1[6] und

[6]Vgl. DIN EN ISO 10218-1:2012-01 [129].

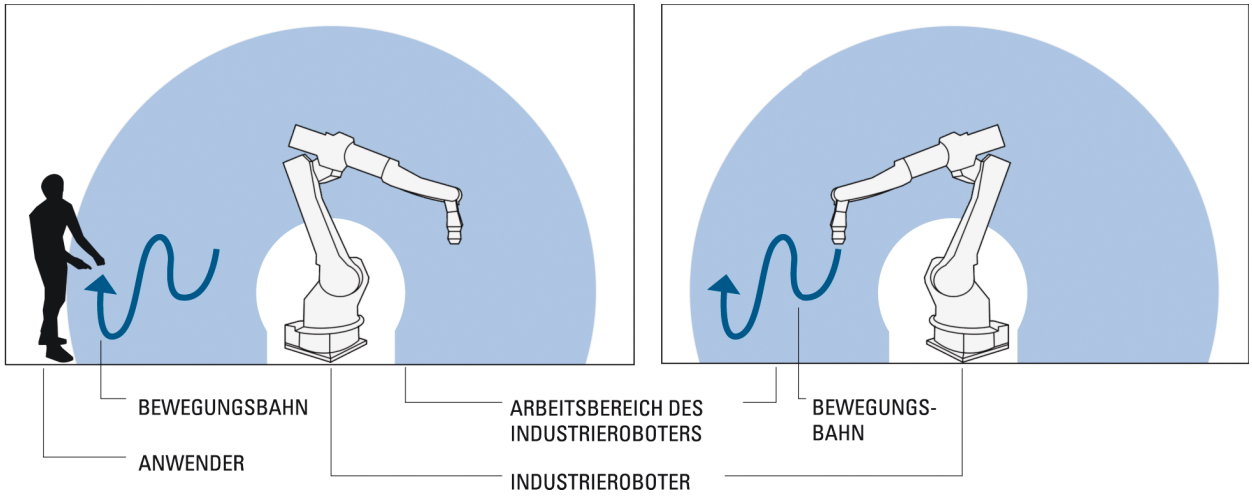

Abbildung 4.8: Bewegungsvorgabe durch Demonstration (links) und Imitieren der Bewegung durch den Industrieroboter (rechts).

DIN EN ISO 10218-2[7] herangezogen. Insbesondere DIN EN ISO 10218-1, 5.8.6 regelt Anforderungen an kabellose Programmierhandgeräte und DIN EN ISO 10218-2, 5.7.2 Anforderungen an kabellose Kommunikation. Danach muss für eine gestenbasierte Bewegungsführung in simultaner Ausführung ein zusätzlicher dreistufiger Zustimmtaster zum Starten der Bewegung des Industrieroboters eingesetzt werden. Zudem hat der Programmierer bzw. das Programmiergerät über einen Not-Halt-Taster zu verfügen. Im Fachhandel sind externe Zustimmtaster mit Not-Aus-Funktion für den einhändigen Betrieb durchaus erhältlich. Eine derartige Ausführung der Interaktion würde jedoch einerseits die Bewegungsfreiheit des Programmierers einschränken; andererseits besteht ein nicht zu unterschätzendes Sicherheitsrisiko durch unbeabsichtigte Bewegungen oder fehlerhafte Sensorwerte der Gestenerkennung. Die Einschränkung der Mobilität sowie die unklare Sicherheitslage bilden somit ein Ausschlusskriterium für die momentane industrielle Anwendbarkeit der Bewegungssteuerung.

Gestaltung der räumlichen Eingabe des Roboterprogramms

Auf Basis der vorangegangenen Diskussion wird festgestellt, dass die gestenbasierte Bewegungsvorgabe sowie die manipulative Interaktion mit räumlichen Objekten vielversprechende räumliche Eingabemethoden darstellen. Das Programm besitzt bei Ausführung durch den Roboter eine räumliche Abbildung, welche sich intuitiv an der beabsichtigten Position im Raum durch Bewegungsvorgabe definieren und durch Objektmanipulation anpassen lässt. Somit wird ein hoher Grad an Direktheit der Interaktion erreicht. Auf Basis der sequentiellen Durchführung gestenbasierter Interaktion und Ausführung des abgeleiteten Programms durch den Roboter lässt sich zwischen diesen beiden Schritten eine räumliche Evaluation des Roboterprogramms vornehmen. Die Bewegungssteuerung über Handgesten wird aufgrund der aufgeführten sicherheitstechnischen Beschränkungen und mangelnden Verbesserungsmöglichkeiten der Nutzerfreundlichkeit im Vergleich zu bestehenden Lösungen vernachlässigt. Zudem soll der Einsatz von Kommandogesten minimiert bzw. vermieden werden, um dem Anwender das Erlernen eines komplexen Vokabulars von 3D-Gesten zu ersparen.

[7] Vgl. DIN EN ISO 10218-2:2012-06 [136].

B) Ausgabe in Augmented Reality

Für die Ausgabe räumlicher Informationen wird eine mobile AR-Anwendung gewählt. Diese hat im Vergleich zur ausschließlich virtuellen Darstellung den Vorteil, dass durch die Erweiterung der realen Umwelt um virtuelle Informationen ein höherer Grad an Immersion erreicht werden kann. Zudem können bei mobiler Realisierung von AR-Anwendungen die bereitgestellten Informationen standortabhängig angepasst werden. Somit lässt sich eine direkte Erweiterung des natürlichen Sichtfelds des Anwenders erreichen, während dieser seine Position und Blickrichtung frei bestimmen kann. Zur Gestaltung der praktischen Ausführung der AR-Anwendung im räumlichen Programmiersystem sind des Weiteren die Art der Displaytechnologie sowie die gerätetechnische Ausführung zu wählen. Dabei gilt es, eine Entscheidung zwischen Video-See-Through (VST) und Optical-See-Through (OST) sowie zwischen Handheld-Geräten sowie HMDs zu treffen[8]. Als Kriterien werden sowohl technische und ergonomische als auch wirtschaftliche Aspekte betrachtet.

Auswahl der Displaytechnologie

Eine Voraussetzung für die Anwendung der AR auf Basis von VST ist, dass ein Kamerabild der Umgebung aufgenommen wird. Dieses wird dem Nutzer auf einem Anzeigegerät zur Verfügung gestellt. Durch die Darstellung des Kamerabilds ergibt sich eine Verzögerung der Anzeige. Im Vergleich zum menschlichen Auge decken Kameras für VST-Ausführungen ein geringeres Sichtfeld ab. Somit ist bei VST lediglich die Möglichkeit zu einer begrenzten Augmentierung des Sichtfelds gegeben [137]. Die Ausführung von VST, bspw. auf modernen Handheld-Geräten mit Display und

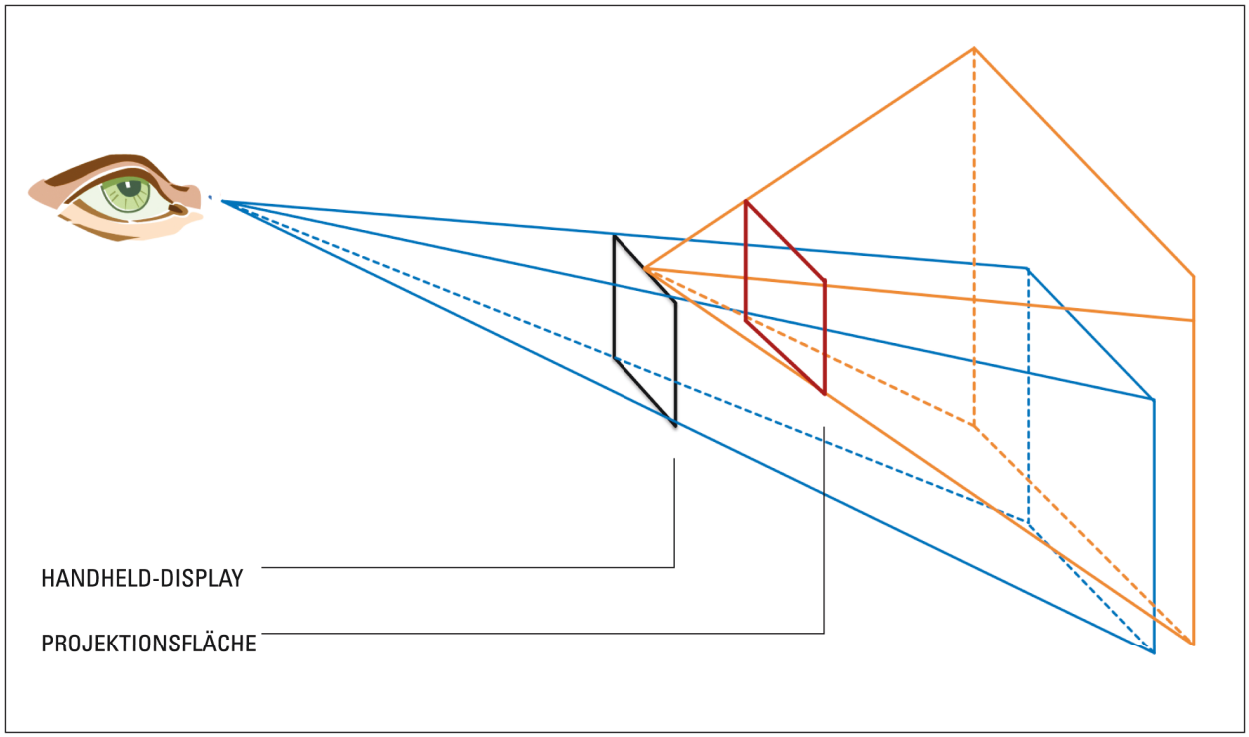

Abbildung 4.9: Divergenz des Sichtbereichs des Nutzers auf das Handheld-Display (blau) und des Sichtbereichs der Kamera des Handhelds (orange).

[8] Vgl. Augmented Reality in Kapitel 2.1.3.

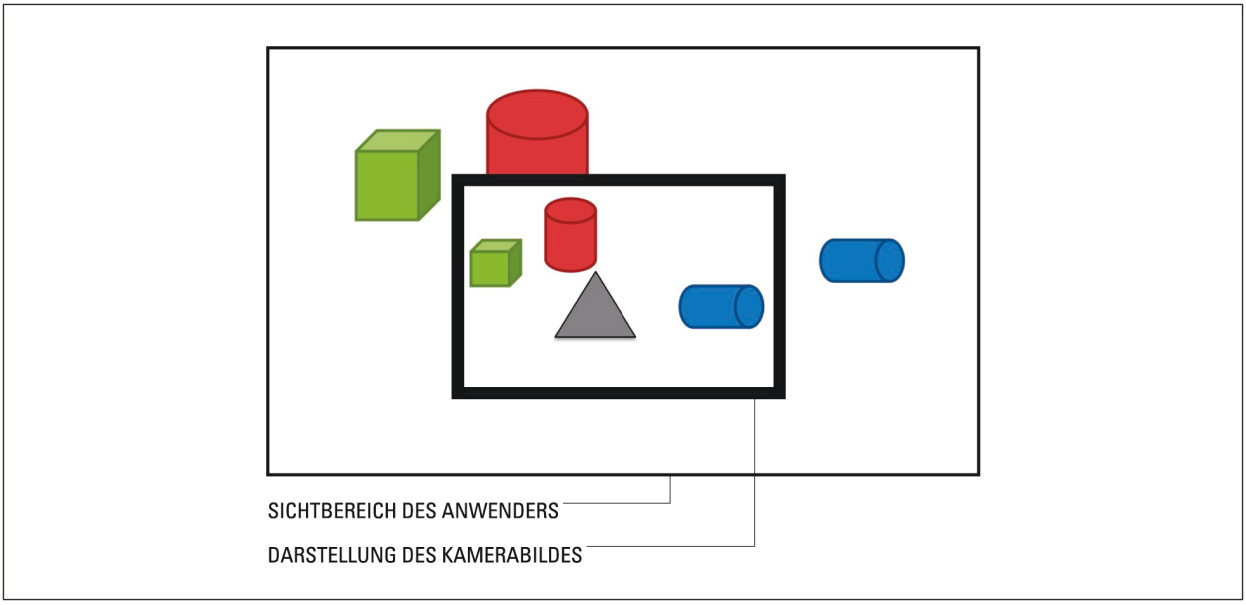

Abbildung 4.10: Resultierendes Sichtfeld bei der Betrachtung von Objekten.

Kamera auf der Gegenseite, ermöglicht die Realisierung von AR-Anwendungen als scheinbare See-Through-Ausführung. Dieser Effekt, der auch mit „Magic Lense“ bezeichnet wird, modifiziert den Sichtbereichs des Nutzers [138]. Die Darstellung des Kamerabilds bildet jedoch bei größerem Versatz zwischen Auge und Kamera auf dem Display typischerweise einen größeren Sichtbereich ab, als der Mensch beim Schauen durch einen äquivalenten quadratischen Rahmen wahrnehmen würde. Dies führt im Vergleich zu echten See-Through-Displays zu einer weniger intuitiven Wahrnehmung des erweiterten Sichtfelds [139]. Die Abbildungen 4.9 und 4.10 verdeutlichen die Divergenz der Sichtbereiche von Kamera und Anwender am Beispiel einer Handheld-Ausführung von VST sowie die beispielhafte Wahrnehmung von Objekten durch ein VST-Display.
OST-Displays verzichten auf Kameras. Die virtuellen Objekte werden in oder auf eine transparente Oberfläche, die gleichzeitig als Sichtfenster dient, eingeblendet. Im Vergleich entsteht derart eine realistischere Umgebungswahrnehmung als bei VST. Es treten jedoch geringe Verzögerungen bei der Darstellung der virtuellen Objekte auf, was zur Irritation des Nutzers führen kann. Ferner existiert ein Parallaxenfehler, welcher bei einem winkligen Versatz der Augen zum Display in einer fehlerhaften räumlichen Wahrnehmung der Position und Orientierung virtueller Objekte im Raum resultiert. Die wahrgenommene Pose der virtuellen Objekte kann je nach Ausführung sehr stark von der beabsichtigten Position der Anzeige abweichen. Abbildung 4.11 verdeutlicht diesen Effekt im Vergleich der beiden Display-Technologien. Zwar existieren in der Fachliteratur Ansätze für parallaxenfreie OST-Displays [140, 141], diese benötigen jedoch einen hohen zusätzlichen technischen Aufwand zur Anpassung des Sichtbereichs, bspw. durch Iris-Tracking und entsprechender Anpassung der Darstellung.
Bei VST tritt dieser Effekt nicht auf. Zwar passen bei winkligem Versatz vom Anwender zum Display Blickrichtung und angezeigtes Bild nicht überein. Die Darstellung der Position der virtuellen Objekte bleibt jedoch bezogen auf das Kamerabild korrekt. Aus diesem Grund wird VST als Displaytechnik gewählt.

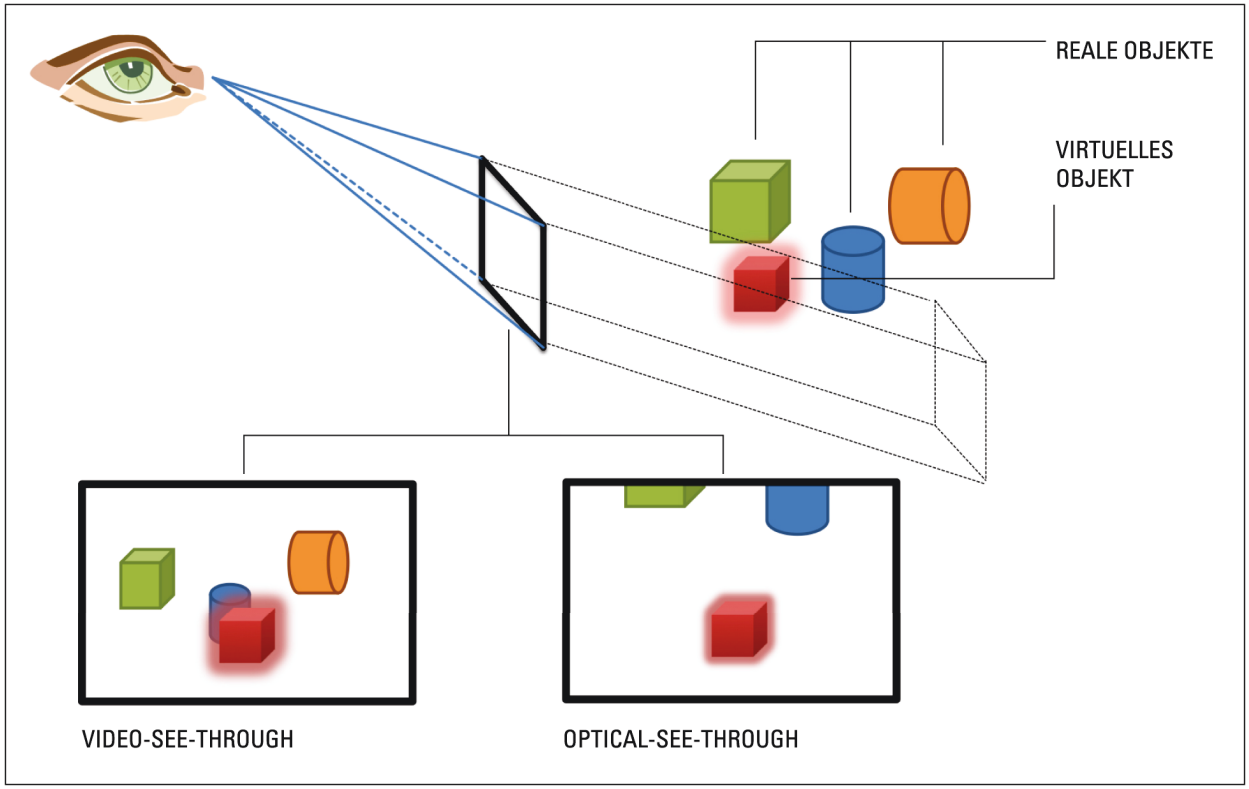

Abbildung 4.11: Auswirkung des Parallaxenfehlers bei der Einblendung virtueller 3D-Objekte in der AR.

Gerätetechnische Umsetzung der Augmented Reality

Eine Voraussetzung für die mobile Anwendung der AR auf Basis von VST ist, dass das Kamerabild nicht stationär durch eine festmontierte Kamera aufgenommen wird, sondern der Bewegung und der Sicht des Anwenders folgt bzw. durch diesen flexibel bestimmt werden kann. Demgemäß erfolgt durch die AR eine Erweiterung des natürlichen Sichtfelds des Anwenders durch virtuelle Informationen, während dieser seine Position und Blickrichtung frei bestimmt. Zur Ausführung einer derartigen Anzeige in Verbund von Kamera und Display lassen sich Handheld-Geräte oder HMDs einsetzen.

Haltung und Interaktionsfreiheit: Bei Handheld-Geräten bleibt ein Großteil des natürlichen Sichtfelds erhalten. Um eine Anpassung des Kamerabilds an die aktuelle Position des Anwenders zu ermöglichen, müssen die Position und Orientierung des Displays über eine Veränderung der Handpose vorgegeben werden. HMDs bieten den Vorteil, dass sie direkt das Sichtfeld in Blickrichtung erweitern. Dabei wird beabsichtigt, den kompletten Sichtbereich des Anwenders durch ein oder mehrere Displays abzudecken. Bei der Verwendung von Handheld-Geräten erfolgt die Erweiterung des Blickfelds über einen Umweg, da das Gerät in der Regel vor dem Körper gehalten wird. Die Anzeige des Kamerabilds entspricht demgemäß nicht der direkten Blickrichtung des Nutzers. Somit ist der Grad der Immersion bei Handheld-Geräten prinzipiell geringer. Bei einer potentiellen gestenbasierten Interaktion vor dem Gerät und gleichzeitiger Betrachtung der Anzeige steigt hier weiterführend der kognitive Aufwand zur räumlichen Koordination der Eingabe [142]. Ein weiterer Nachteil der Handheld-Geräte ist, dass diese in der Hand gehalten werden müssen. Das hat zur Konsequenz, dass während der Betrachtung der Anzeige durch den Nutzer, gestenba-

sierte Eingaben im Raum nur mit der freien Hand erfolgen können. Beidarmige Interaktionen im Raum sind demgemäß ausgeschlossen.

Gewicht und Tragekomfort: In der Vergangenheit zeichneten sich HMDs durch ein hohes Eigengewicht und Zusatzgewichte in Form von Recheneinheiten und Energieversorgung aus. Die Zusatzgewichte wurden zumeist am Körper als sogenanntes „Backpacksystem“ getragen. Somit lag eine eingeschränkte Mobilität und schlechte Ergonomie vor. Der stark begrenzte Anwendungscharakter beschränkte die Nutzung meist auf Forschungsprojekte zum Zweck eines Machbarkeitsnachweises (engl. *proof of concept*). Aktuell verfügbare HMDs sind kleiner, leichter, besitzen leistungsstarke Energiespeicher und verzichten durch Integration der Recheneinheit in das Kopfteil auf das Backbacksystem. Für den mobilen Einsatz sind moderne HMDs aufgrund des relativ schweren Kopfteils aus ergonomischer Sicht jedoch immer noch nur beschränkt einsetzbar. Weiterführend erzeugt das Kamerabild in VST-Ausführung einen unnatürlichen Sichtbereich, der für den Anwender gewöhnungsbedürftig und auf Dauer anstrengend ist. Das Handheld-Gerät kann im Gegensatz flexibler, je nach Bedarf eingesetzt werden. Smartphones können bspw. aufgrund ihrer geringen Größe bei Nichtverwendung in den Taschen der Kleidung des Anwenders untergebracht werden [142].

Zusatzfunktionen und Kosten: Die Produktentwicklungen im Bereich der Handheld-Geräte sind getrieben durch das breite Interesse der Konsumenten. Dies beschleunigt die Entwicklungsprozesse in diesem Bereich und resultiert in besseren Leistungskenngrößen, sensorieller Ausstattung sowie Zusatzfunktionen. Auch der durchschnittliche Preis von Handheld-Geräten liegt im Vergleich zu HMDs deutlich niedriger. Je nach Ausführung bieten Handheld-Geräte verschiedene Zusatzfunktionen, die bspw. ergänzende Eingabefunktionen oder Kommunikationsschnittstellen bereitstellen. Zudem liegt bereits eine hohe Verbreitung dieser Geräte im privaten und industriellen Umfeld vor.

Entscheidung: Tabelle 4.1 fasst die Vor- und Nachteile beider Technologien vergleichend zusammen. Aufgrund der ergonomischen Einschränkungen von HMDs, dem beschränkten Funktionsumfang und dem höheren Preis werden für das räumliche Programmiersystem Handheld-Geräte als Anzeigemedium der AR gewählt.

	HEAD MOUNTED DISPLAYS	HANDHELD-GERÄTE
Interaktionsfreiheit	●	○
Tragekomfort	○	●
Zusatzfunktionen	○	●
Wirtschaftlichkeit	○	●

● erfüllt ○ nicht erfüllt

Tabelle 4.1: Vergleich der Eigenschaften von HMDs und Handheld-Geräten.

Darstellung der AR

Für eine perspektivisch korrekte Ausrichtung der virtuellen Objekte im Kamerabild erfordern mobile AR-Anwendungen eine dynamische Posenbestimmung des Mobilgeräts[9]. Zu diesem Zweck dient ein 6DoF-Tracking, welches für jeden angezeigten Frame die räumliche Verschiebung zwischen dem Kamerakoordinatensystem und einem Referenzkoordinatensystem der AR ermittelt und die Visualisierung dementsprechend anpasst. Abbildung 4.12 veranschaulicht die Funktionsstruktur von Tracking-basierten AR-Anwendungen auf Handheld-Geräten. Bei statischen Anwendungen genügt die einmalige Bestimmung der Translation und Rotation zwischen dem Kamera-KS und einem Referenz-KS durch eine Mess- oder Kalibriermethode. Durch die kontinuierliche Ermittlung steigt die benötigte Rechenleistung zur Ermittlung und Verarbeitung der aktuellen Transformation [143].

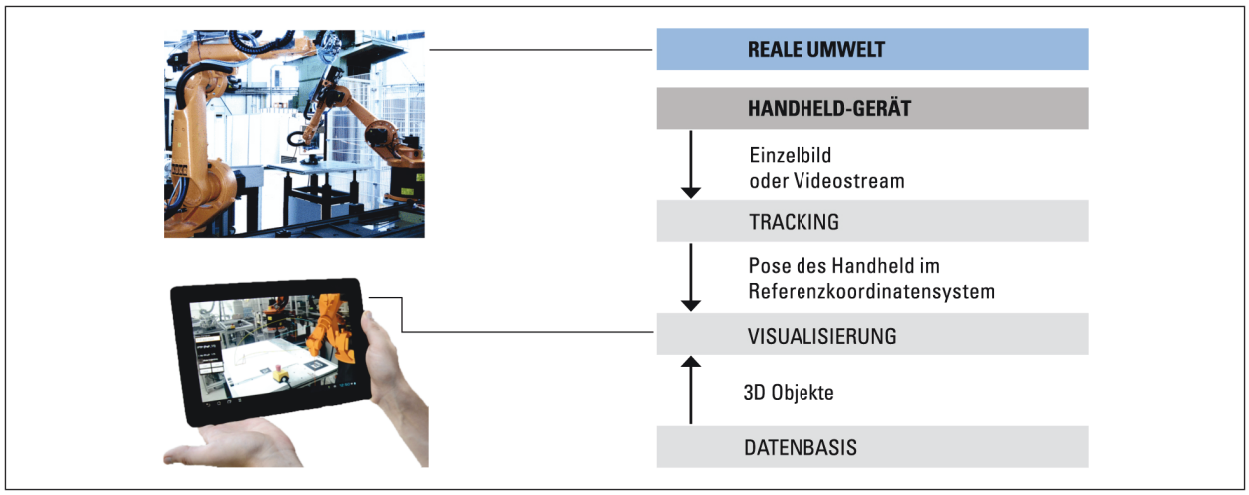

Abbildung 4.12: Vereinfachte Funktionsstruktur der AR-Darstellung auf Handheld-Geräten.

In der vorliegenden Arbeit wird ein kamerabasiertes, optisches Tracking-Verfahren von ebenen Markern (engl. *fiducial marker*), sogenannten „Flachmarkern“, gewählt. Diese sind in den meisten Fällen quadratisch und enthalten im Inneren ein asymmetrisches Muster. Abbildung 4.13 zeigt verschiedene Flachmarker mit unterschiedlichen Ausprägungen des inneren Musters. Optische Tracking-Verfahren erlauben eine flexible Positionierung der Kamera. Der Vorteil der gewählten Flachmarker im Vergleich zu anderen optischen Trackingverfahren liegt in der kostengünstigen Durchführung des Tracking. Die Marker lassen sich in wenigen Minuten an beliebigen PCs generieren, ausdrucken und in der Umwelt frei platzieren. Die Erfassung der Flachmarker lässt sich über die 2D-Kamera realisieren, welche auch die Grundlage für die Darstellung der realen Szene in der AR bildet. In Anhang A wird der allgemeine Ansatz zum Marker-Tracking auf Basis eines 2D-Kamerabilds sowie die perspektivische Darstellung virtueller Objekte im Kamerabild vorgestellt. Ein Nachteil dieses Verfahrens ist, dass die Erkennung der Marker von der Beleuchtung abhängig ist [144]. Ferner sind die Genauigkeiten des eingesetzten Marker-Tracking abhängig von der Auflösung der Kamera sowie von der Distanz und dem Winkel zwischen Kamera und Marker.

[9] Vgl. Registrierung in Kapitel 2.1.3.

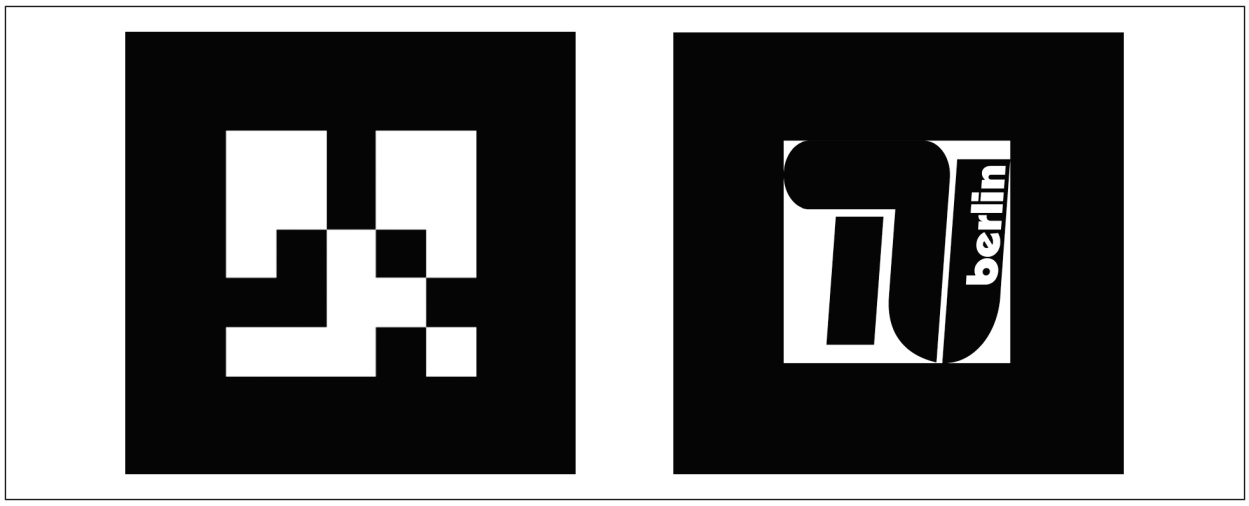

Abbildung 4.13: Flachmarker: binäre Kodierung (links) und asymmetrisches Bildmuster (rechts).

In Anlehnung an [145] werden Anforderungen für ein anwendungsgerechtes Marker-Tracking für das räumliche Programmiersystem aufgestellt:

- Die Identifizierung und das Tracking der Marker erfolgt effizient hinsichtlich begrenzter Ressourcen auf Handheld-Geräten.
- Das Marker-Tracking verfügt über eine hohe Identifikationsrate.
- Das Marker-Tracking bzw. der Algorithmus ist resistent gegen Bildrauschen und Veränderung der Beleuchtungssituation und somit im industriellen Umfeld robust einsetzbar.
- Über das Marker-Tracking bzw. die Markerpositionierung wird ein hinreichend großer Interaktionsbereich (Roboterzelle) abgedeckt.

Zur Abdeckung größerer Interaktionsbereiche und zur Steigerung der Robustheit lässt sich der Ansatz des Marker-Tracking auf mehrere Marker, ein sogenanntes „Multi-Marker-Tracking“, erweitern. Bei der Verwendung verschiedener Marker können diese flexibel im Raum verteilt werden. Durch eine Variation der Maße der eingesetzten Marker können größere Entfernungen von der Kamera zum Marker und die damit verbundenen Einbußen bei den Genauigkeiten des Tracking teilweise kompensiert werden. Eine umfangreiche Untersuchung der Genauigkeiten des Flachmarker-Tracking kann [48] entnommen werden. Ansätze des Tracking von inhärenten Eigenschaften der Umgebung (engl. *natural feature tracking*), bspw. [146], werden vernachlässigt, da durch ein fehlendes Tracking von 3D-Merkmalen mit bekannten Abmaßen kein akkurates Tracking für flexible Anwendungsszenarien gewährleistet werden kann.

4.2.2 Methodischer Ansatz der interaktiven Programmierung

In diesem Abschnitt wird ein methodischer Ansatz zur Anwendung der Interaktionsgestaltung im natürlich-räumlichen Programmiersystem dargestellt [147]. Das Ergebnis bildet das angestrebte Interaktionskonzept ab, welches die gestenbasierte Programmdefinition, die räumliche Evaluation

des Programms in AR und eine Möglichkeit zur natürlich-räumlichen Programmmanipulation unter synergetischer Kombination beider Modalitäten umfasst (s. Abb. 4.14).

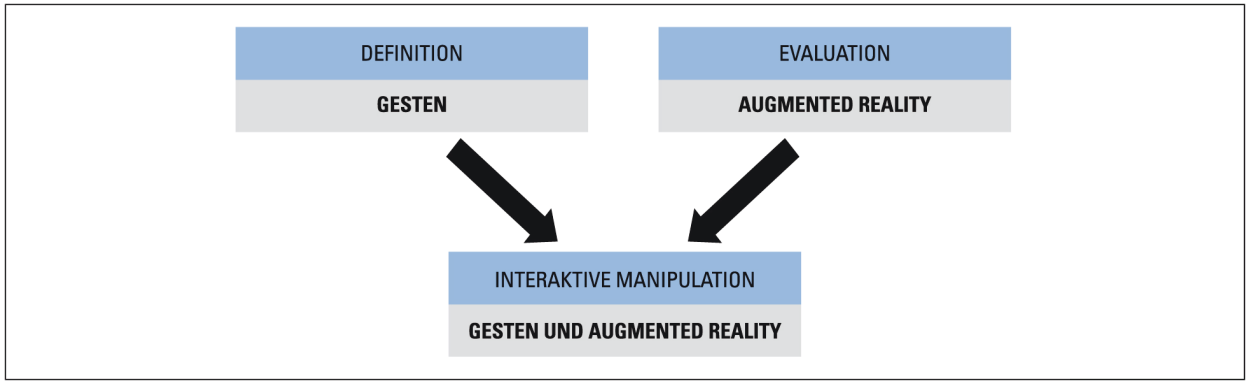

Abbildung 4.14: Darstellung des Interaktionskonzepts zur natürlich-räumlichen Programmierung von Industrierobotern.

Im Rahmen des Interaktionskonzepts wird gezeigt, dass sich mit den gewählten Modalitäten in diesen drei Programmierschritten bezüglich der Adressierung von Nutzern mit unterschiedlicher Expertise verschiedene Abstraktionsebenen der Roboterprogrammierung abbilden lassen. Zu diesem Zweck werden folgende Programmierebenen unterschieden:

1. Bewegungsorientierte Programmierung
   (a) Posen
   (b) Trajektorien
2. Aufgabenorientierte Programmierung

Die Unterteilung von Posen und Trajektorien wird vorgenommen, um eine zusätzliche Flexibilisierung in der bewegungsorientierten Programmierung zu realisieren. Somit lassen sich Posen unabhängig von der Interpolationsart erzeugen und gegebenenfalls individuell anpassen sowie in verschiedenen Bewegungsabschnitten und Programmen mehrfach verwenden.

A) Gestenbasierte Eingabe

Die gestenbasierte Definition des Roboterprogramms erfolgt nach dem Prinzip der Bewegungsvorgabe. Das bedeutet, dass das Programm durch Demonstration im 3D-Raum erzeugt wird. Es werden verschiedene Abstraktionsebenen des Roboterprogramms betrachtet. Abbildung 4.15 verdeutlicht die unterschiedlichen Programmierebenen anhand einer Prinzipdarstellung zur Definition. Neben der Definition einzelner Posen und Trajektorien sollen Aufgaben zur Adressierung des fachfremden Anwenders durch Demonstration definiert werden.

Bewegungsorientierte Programmdefinition

Posen - Position: Der Anwender befindet sich im Arbeitsraum des Industrieroboters, während dieser stillsteht. Durch Zeigegesten des Arms und der Finger an bestimmte Positionen im Arbeits-

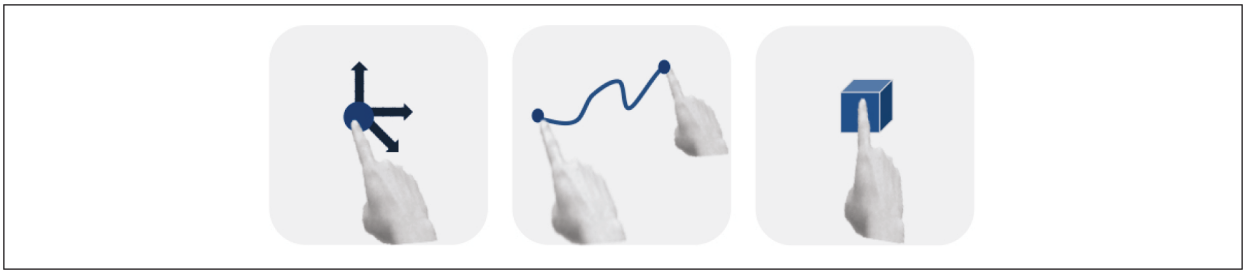

Abbildung 4.15: Gestenbasierte Definition von Posen (links), Trajektorien (Mitte) und Aufgaben (rechts).

raum werden dort Posen mit den absoluten Koordinaten der Zeigepunkte im Roboterkoordinatensystem angelegt. Diese Posen stehen im weiteren Prozess der Programmierung zur Verknüpfung mit Interpolationsbefehlen oder zur Parametrierung von Aufgaben zur Verfügung.

Posen - Rotation: Die Definition der Rotation der Posen lässt sich nicht über eine einzelne Positionsdefinition vornehmen. Auf Hilfsmittel, bspw. einen Zeigestift, zur Demonstration der Orientierung soll zugunsten einer natürlichen Umsetzung der gestenbasierten Interaktion verzichtet werden. Entsprechend muss über positionsbasierte Interaktion eine anschauliche Art der Definition der Rotation vorgenommen werden. Da das Bezugs-KS der Posendefinition dem Anwender möglicherweise nicht bekannt ist, soll eine relative Darstellung der Rotation für den Definitionsprozess, bspw. in Form von Euler-Winkeln, vermieden werden. Vielmehr soll die Orientierung absolut mit einem räumlichen Bezug zur resultierenden Orientierung des Endeffektors festgelegt werden. Die Darstellung der Rotation in Form eines Koordinatensystems bietet einen anschaulichen Ansatzpunkt zur Definition der Rotation, der sich simultan mit adäquatem visuellen Feedback kombinieren lässt. Um die Darstellung des resultierenden Effektor-KS weiterführend zu vereinfachen, soll die Definition des Koordinatensystems auf die Arbeitsrichtung des Tool Center Point (TCP) und einer einzelnen Drehung zur Definition der zweiten Koordinatenachse reduziert werden. Beide Vektoren bilden Basisvektoren für das resultierende Effektor-KS. Der dritte Basisvektor ergibt sich aus dem Kreuzprodukt der beiden Vektoren. Abbildung 4.16 veranschaulicht die Arbeitsrichtungen verschiedener Endeffektoren am Beispiel eines Greifers und eines Schweißbrenners.
Über eine zweite Zeigeposition im Anschluss an die Definition der Position der Pose soll zunächst der Vektor der Arbeitsrichtung des TCP definiert werden. Anschließend kann über eine dritte Position der Winkel bzw. die Richtung der zweiten Koordinatenachse festgelegt werden. In Anhang B ist eine analytische Lösung zur Ermittlung der Rotation in Form der Quaterniondarstellung anhand der drei Positionen zu finden.

Trajektorien: Zur Definition der Trajektorien werden per Zeigegeste definierte Posen automatisch mit einem festgelegten Interpolationsbefehl verbunden. Der resultierende Bewegungsbefehl wird an einer ausgewählten Stelle im Roboterprogramm eingefügt. In Verbindung mit der Ausgangsposition des vorherigen Programmschritts ergibt sich die Bewegungsbahn für einen einzelnen Interpolationsbefehl. Darüber hinaus sollen komplexe Bahnbewegungen durch Vormachen der Bewegung mit der Hand festgelegt werden können. Zur Definition dieser Freiformkurven muss eine hohe Anzahl von Positionen aufgezeichnet werden. Das Starten und Beenden der Bahnaufzeichnung soll durch den Anwender per Fingergeste gesteuert werden.

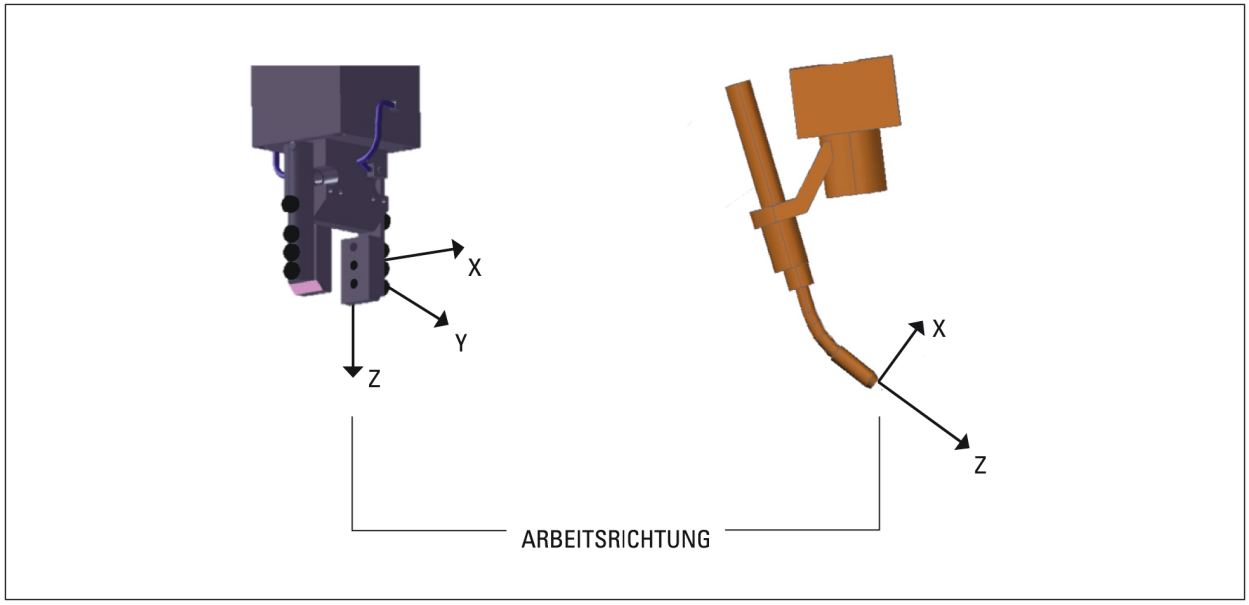

Abbildung 4.16: Verschiedene Endeffektoren und Ausrichtung der Endeffektor-KS, Greifer (links) und Schweißbrenner (rechts).

Aufgabenorientierte Programmdefinition

Die aufgabenorientierte Programmdefinition soll mittels Vormachen der Aufgabe durch den Anwender im Arbeitsbereich des Roboters erfolgen. Das Roboterprogramm wird auf Basis der beobachteten gestenbasierten Interaktion nach festen Regeln erstellt, sodass der Roboter die Aufgabe imitiert. Zu diesem Zweck müssen aus den beobachteten Bewegungen automatisiert aufgabenspezifische Parameter extrahiert werden. Basierend auf Bewegungsdaten lassen sich bspw. Arbeitsschritte erkennen und aus diesen anwendungsspezifische Bewegungsparameter ableiten. Diesbezüglich kann [148] ein Ansatz, basierend auf Bewegungsdaten eines Motion-Tracking-Systems, entnommen werden. Die ermittelten Bewegungsparameter dienen im nächsten Schritt zur regelbasierten Erstellung eines Roboterprogramms anhand von Programmmakros. Diese bildet die grundlegende Durchführung von Teilaufgaben für den Roboter ab, verfügen jedoch noch nicht über die anwendungsspezifischen Bewegungsparameter. Abbildung 4.17 verdeutlicht den prozessualen Ablauf zur automatischen Erstellung des Roboterprogramms aus der beobachteten Demonstration. Sowohl die Regeln zur Parametrierung als auch eine neutrale Repräsentation des Vorgehens in Form von Programmmakros sollen im Vorfeld durch einen Experten aufgabenspezifisch erstellt werden.

Je nach Ausführung kann eine intuitive Demonstration die beabsichtigte Produktionsaufgabe exakt abbilden oder lediglich instruktiv andeuten. Eine räumliche Andeutung der Aufgabe empfiehlt

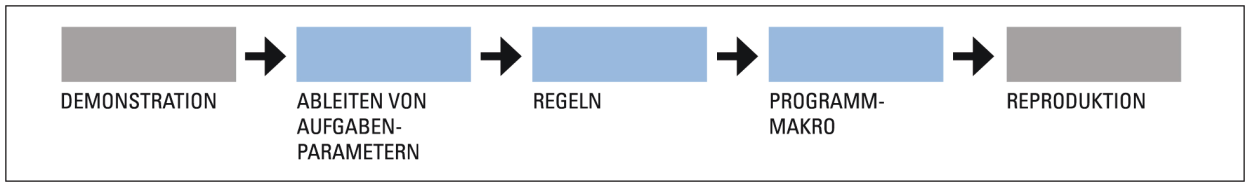

Abbildung 4.17: Prozess zur Imitation von aufgabenorientierten Demonstrationen.

sich bspw. für die Definition einer Schweißbahn. Hier ist es nicht zielführend, den Anwender den Schweißprozess im Rahmen der Programmierung real durchlaufen zu lassen. Alternativ bietet es sich an, die Schweißbahnen über Zeigegesten lediglich räumlich anzudeuten. Derart kann auf eine natürliche Geste zurückgegriffen werden, die es ermöglicht, die Schweißbahnen auf anschauliche Weise zu definieren, ohne dass Schweißbrenner oder andere Hilfsmittel eingesetzt werden müssen. Dass eine genaue Übereinstimmung der Interaktion zu einer intuitiven Art der räumlichen Interaktion führen kann, lässt sich am Beispiel von Pick-and-Place- und Montageaufgaben verdeutlichen. Hier bietet sich bspw. eine reale Bewegung der Objekte durch den Anwender in exakter Ausführung der Aufgabe an. Stehen die realen Objekte nicht zur Verfügung, kann stattdessen eine räumliche Interaktion mit virtuellen Platzhaltern erfolgen. Dies setzt jedoch eine Visualisierung, bspw. in der AR, voraus und reduziert prinzipiell die Simplizität des Verfahrens für den Anwender. Da bei der Interaktion mit virtuellen Objekten das reale Bezugsobjekt fehlt, müssen für die intuitive Gestaltung der Interaktion gegebenenfalls weitere Feedbackfunktionen bereitgestellt werden. Da eine passende Gestaltung der aufgabenorientierten Definition sowohl von der Produktionsaufgabe als auch von weiteren Randbedingungen abhängt, ist die Ausführung gemäß den aufgezeigten Interaktionsformen individuell zu treffen.

B) Feedback und Evaluation in einer mobilen AR-Anwendung

Mobile Programmevaluation in Augmented Reality
Die Darstellung der AR auf dem Handheld-Gerät dient der mobilen Visualisierung und Simulation von Roboterprogrammen und ermöglicht zusätzlich visuelles räumliches Feedback zur gestenbasierten Interaktion. Im Vergleich zur klassischen Simulation in CAD-gestützten OLPs erfolgt die Verschmelzung von VR und realer Umwelt. Der Anwender ist bei der Evaluation des Roboterprogramms nicht an einen Computerarbeitsplatz gebunden. Die Darstellung der virtuellen Objekte ist abhängig von der Platzierung der AR-Marker. Diese sollen für die Realisierung der prozessnahen räumlichen Programmierungen in der Roboterzelle angebracht werden, sodass eine Erweiterung des Raumes um virtuelle Informationen ermöglicht wird.
Prinzipiell wird durch dieses Vorgehen auch eine prozessferne Simulation von Roboterprogrammen unter ortsflexibler Verwendung der entsprechenden Marker realisiert. Somit lässt sich eine prozessnahe aber auch mobile räumliche Simulation umsetzen, welche die klassischen Funktionen von OLPs abbildet und diese mit Zusatzfunktionen der gestenbasierten Eingabe ergänzt.

Visualisierung: Zur Programmevaluation in realer Umgebung sollen virtuelle Informationen des Roboterprogramms in das Kamerabild eingeblendet werden. Die Visualisierung des Roboterprogramms soll gemäß Abstraktionsgrad der Programmierung in unterschiedlichen Ausführungen erfolgen. Abbildung 4.18 veranschaulicht in einer Prinzipdarstellung die Visualisierung von Roboterprogrammen für verschiedene Programmierebenen. Während die bewegungsorientierten Darstellungen allgemeine Gültigkeit besitzen, ist eine anschauliche, visuelle Darstellung auf Aufgabenebene analog zur Gestaltung der gestenbasierten Definition domänenspezifisch vorzunehmen.

Simulation: Mit der Modellierung virtueller Roboter in der AR-Anwendung lassen sich Roboterprogramme im realen Umfeld kinematisch simulieren. Der virtuelle Roboter lässt sich demnach als virtuelle Repräsentanz räumlich an der Position des realen Roboters darstellen. Durch die kinema-

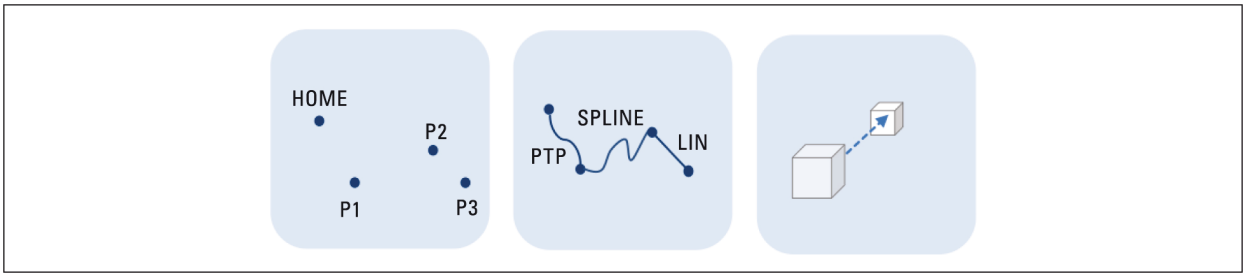

Abbildung 4.18: Visualisierung des Roboterprogramms auf verschiedenen Programmierebenen: Posen (links), Trajektorien (Mitte) und Aufgaben, hier: Pick and Place (rechts).

tische Modellierung des Industrieroboters können einzelne Posen oder vorgegebene Trajektorien abgefahren werden. Abbildung 4.19 veranschaulicht die Visualisierung und Simulation des Roboterprogramms auf mobilen Handheld-Geräten in realer Produktionsumgebung. Neben der groben Ermittlung von Taktzeiten lässt sich das Roboterprogramm hinsichtlich Erreichbarkeit und Kollisionsvermeidung verifizieren. Eine aufwendige Modellierung der Peripherie wie in CAD-gestützten OLPs entfällt. Aus diesem Grund kann die Kollisionskontrolle durch den Anwender zunächst lediglich qualitativ in Form eines visuellen Abgleichs von realen Objekten und Bewegungen des virtuellen Roboters erfolgen. Eine weiterführende automatische Kollisionskontrolle erfordert prinzipiell die virtuelle Modellierung potentieller Kollisionsobjekte in der AR.

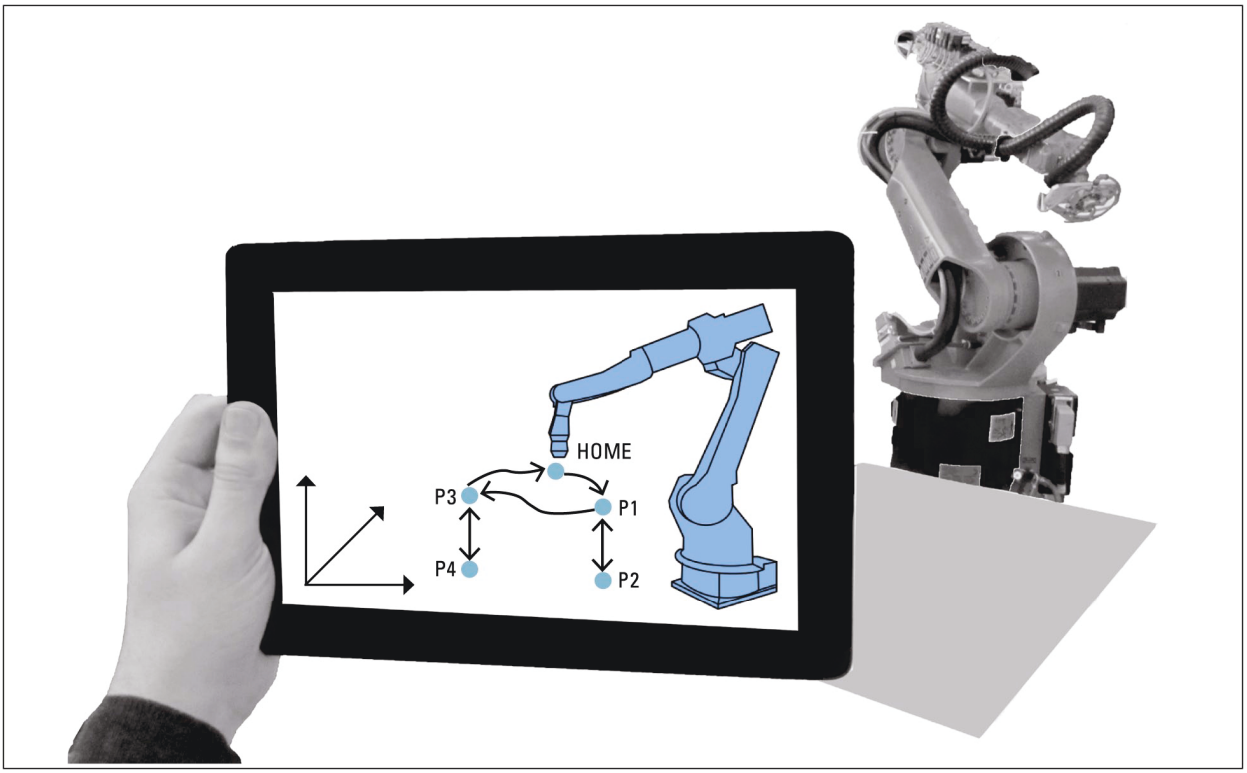

Abbildung 4.19: Prinzipdarstellung der Visualisierung und Simulation des Roboterprogramms in der mobilen AR-Anwendung auf dem Handheld-Gerät.

Feedback zur räumlichen Eingabe
Mithilfe der AR lässt sich in integrativer Nutzung mit der gestenbasierten Programmdefinition und Manipulation bereits während der Interaktion ein räumliches visuelles Feedback bereitstellen. Abhängig von der dargestellten Rückgabe ermöglicht dies, etwaige Fehler frühzeitig zu erkennen und zur Kompensation über entsprechende Mechanismen der Hand-Auge-Koordination eine Anpassung der gestenbasierten Interaktion vorzunehmen.

C) Räumlich-interaktive Programmmanipulation

Durch die Kombination der AR-Anwendung mit markerlosen 3D-Gesten entsteht im Bereich der Industrieroboterprogrammierung eine neuartige Form der Interaktion: Der Anwender wird in die Lage versetzt, mit den virtuell im Kamerabild dargestellten Objekten räumlich zu interagieren. Das bedeutet, dass sich die virtuellen Objekte verschieben oder verdrehen lassen. Der Anwender hält in der einen Hand das Handheld-Gerät, während er mit der anderen Hand die räumlichen Manipulationen durchführt. Der Nutzer kann sich somit in der Roboterzelle frei bewegen, sich Objekten nähern und diese an ihrer Position räumlich manipulieren, indem er sie bspw. greift, bewegt und an einer neuen Position wieder ablegt. Über die AR lässt sich simultan zur Interaktion ein Feedback zur Manipulation geben, welches die temporäre Pose der Objekte räumlich andeutet. Zusätzlich lässt sich ein numerisches Feedback zur Manipulation der Koordinaten bereitstellen.
Bei der Manipulation der Objekte können analog zur Definition und Evaluation wiederum verschiedene Abstraktionsebenen der Roboterprogrammierung adressiert werden. Neben einzelnen Posen lassen sich Bahnabschnitte oder aufgabenspezifische Objekte manipulieren (s. Abb. 4.20). Das Roboterprogramm wird jeweils entsprechend der Interaktion angepasst. Diese Form der Interaktion wird mithilfe einer Prinzipdarstellung in Abbildung 4.20 veranschaulicht.

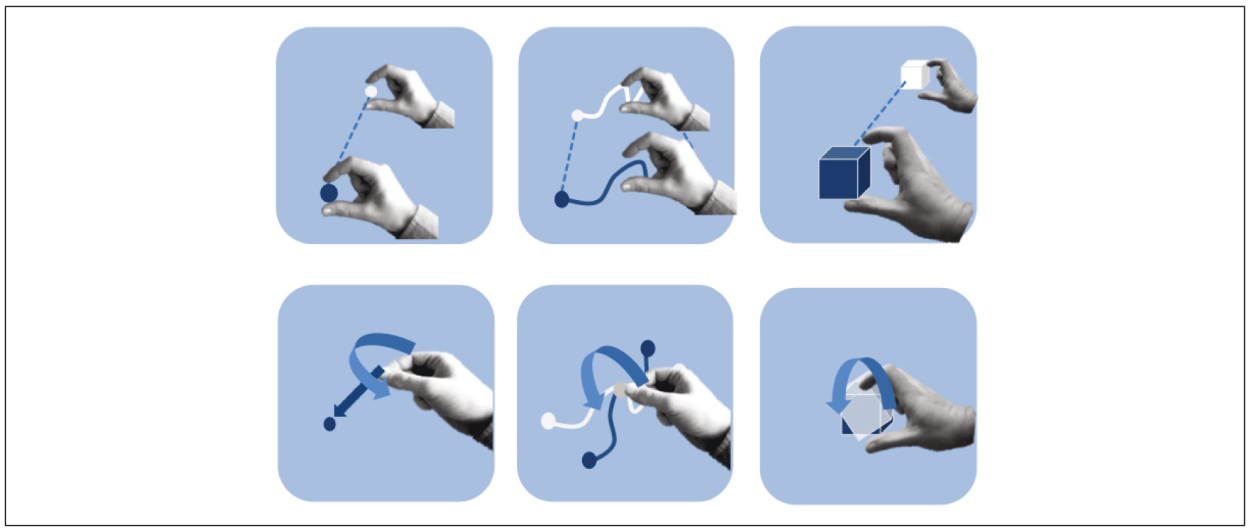

Abbildung 4.20: Darstellung der räumlichen Manipulation von Posen (links), Trajektorien (Mitte) und Aufgaben (rechts).

4.3 Entwurf des natürlich-räumlichen Programmiersystems

Der Entwurf des Programmiersystems konkretisiert die gestalterische Realisierung des Interaktionskonzepts. Zu diesem Zweck werden neben einer Determination des Interaktionskonzepts die Randbedingungen aus dem Bereich der Programmierung sowie aus den Bereichen der Schnittstellen und Inbetriebnahme einbezogen.
Die Hauptkomponenten (s. Abb. 4.21) des Programmiersystems stellen neben dem Industrieroboter das Handheld-Gerät sowie ein Motion-Tracking-System zur Wahrnehmung der Interaktion des Anwenders dar.

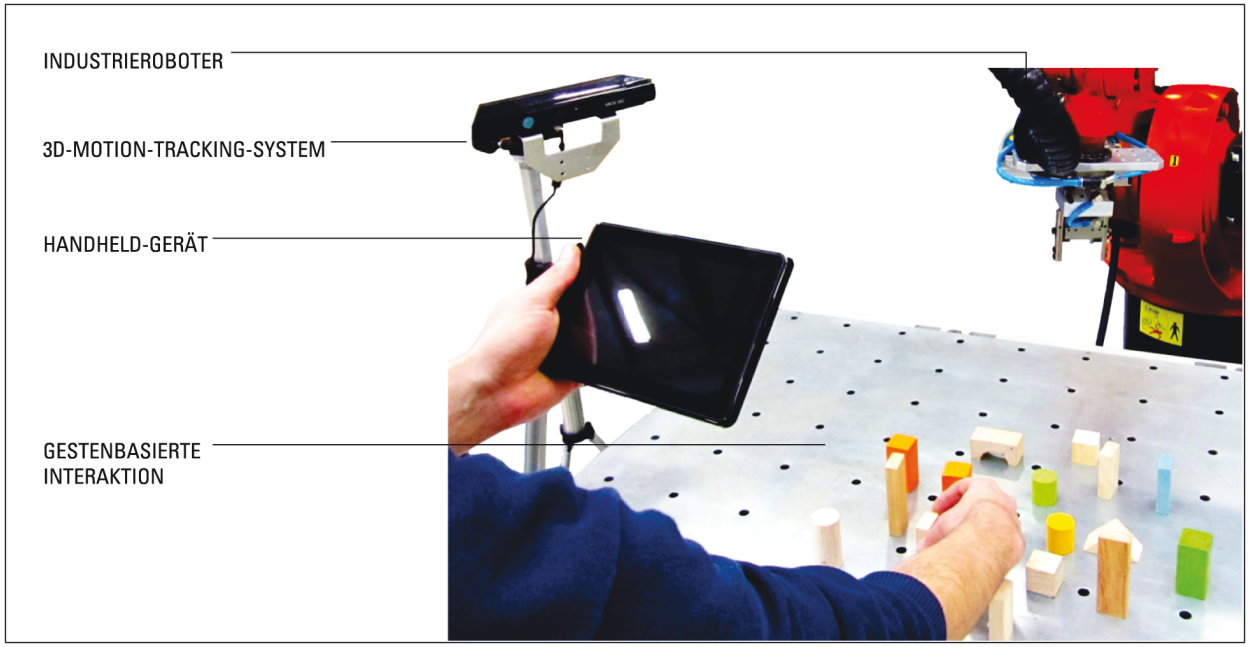

Abbildung 4.21: Grundkomponenten des Programmiersystems: Motion-Tracking-System, Handheld-Gerät und Industrieroboter.

4.3.1 Programmiergeräte: Smartphones und Tablet-PCs

Als zentrale Programmiergeräte des räumlichen Programmiersystems werden herkömmliche Smartphones und Tablet-PCs eingesetzt. Die Programmierumgebung liegt in Form einer App für ein bestimmtes Betriebssystem vor und ist somit prinzipiell übertragbar auf beliebige Geräte, die das gleiche Betriebssystem, bspw. Android oder iOS, nutzen. Im Rahmen der vorliegenden Arbeit entsteht eine spezialisierte App für die Industrieroboterprogrammierung, anwendbar für eine Klasse multifunktional einsetzbarer Handheld-Geräte. Somit wird eine hohe Einsetzbarkeit und Übertragbarkeit hergestellt, da eine Vielzahl von Unternehmen bereits über Tablet-PCs oder Mitarbeiter-Smartphones verfügt. Aktuelle Handheld-Geräte enthalten in der Regel über mehrere Prozessorkerne auf Basis der ARM-Architektur. Die wachsende Leistungsfähigkeit ist jedoch im Vergleich zu modernen PCs noch relativ gering. Die Rechenleistung des Zweikernprozessors

Apple A5 mit einem Standardtakt von jeweils 1 GHz entspricht bspw. nach Benchmarking[10] ungefähr der Rechenleistung des ersten Single-Core-Atoms aus dem Jahr 2008. Auch die Speicherausstattung ist noch vergleichsweise gering und nicht über virtuellen Speicher erweiterbar. Bei den Prozessoren handelt es sich um ein auf einen beschränkten Funktionsbereich und Energieeffizienz abgestimmtes System nach dem Prinzip Systems-on-Chip (SoC). Dabei werden CPU und Grafikkerne (GPU) mit einem Speicher-Controller, Mobilfunk-, GPS- und anderer Sensor- und Kommunikationseinheiten auf einem Chip integriert.
Neben der flexiblen Einsetzbarkeit von Smartphones und Tablet-PCs besteht ein weiterer Vorteil darin, dass diese bereits über Ein- und Ausgabeschnittstellen in Form des Touchscreens, über zusätzliche Sensorik sowie über Kommunikationseinheiten verfügen. Eine Eingabe über Touch-Gesten mit entsprechender Menüführung soll im Programmiersystem zur Programmverwaltung dienen. Das Display wird zur Darstellung der AR-Anwendung genutzt. Das für die Realisierung einer VST-Anwendung benötigte Kamerabild wird von der integrierten Kamera, welche zum Standardumfang der Geräte gehört, akquiriert. Über das Bild der Kamera lässt sich eine ergänzende Erkennung der Umwelt und Interaktion realisieren. Weiterführend können die vorhandenen Kommunikationsschnittstellen über Internet, Bluetooth oder WLAN zur externen Kommunikation genutzt werden. Die einzelnen Module moderner Handheld-Geräte und deren potentielle Anwendungen im Programmiersystem sind in Tabelle 4.2 zusammengefasst. Die Intertialsensorik kann zusätzlich zur Genauigkeitssteigerung der Posenbestimmung des Handheld-Geräts durch das visuelle Marker-Tracking dienen. Der Autor der vorliegenden Arbeit hat in [147] die erfolgreiche Implementierung eines entsprechenden Ansatzes zur Datenfusion anhand eines Kalman-Filters vorgestellt und eine Genauigkeitssteigerung nachgewiesen.
Ein Nachteil herkömmlicher Mobilgeräte im Vergleich zu spezialisierten Programmiergeräten ist deren mangelnde Eignung für den industriellen Einsatz in potentiell verschmutzten Umgebungen. Dem kann gegenübergestellt werden, dass für die meisten Handheld-Geräte sogenannte „Ruggedized Cases“, also robuste Gehäuse, angeboten werden, die den Nutzer nicht oder nur kaum in seiner Interaktionsfähigkeit einschränken und über die entsprechenden industriellen Schutzklassen verfügen. Ferner existieren die ersten Ausführungen von robusten Smartphones und Tablet-PCs speziell für den industriellen Einsatz.
Aufgrund der beschränkten Leistungs- und Speicherfähigkeit dieser Geräte soll in der prototypischen Umsetzung die allgemeine Eignung dieser Geräte als Programmiergerät untersucht werden. Dabei sind die Kinematiksimulation sowie Bildverarbeitungsalgorithmen als kritische, d. h. rechenintensive Prozesse anzusehen. In diesem Rahmen ist zu untersuchen, ob und inwiefern diese Funktionen umgesetzt werden können. Da in der App ein kontinuierlicher Videostream verwendet wird, verursachen erhöhte Rechenzeiten Verzögerungen in der Anzeige von Frames bzw. eine Reduzierung der Bildfrequenz. Die Geschwindigkeit der Rückmeldung hat einen erheblichen Einfluss auf das Benutzerverhalten und die Leistungsfähigkeit der Anwendung [21]. Betroffen von den Auswirkungen ist vorrangig die Usability der Gestenerkennung und der Simulation, da Ausgaben erst verzögert dargestellt werden können und somit nicht mehr mit der aktuellen Situation des Anwenders übereinstimmen. Im Extremfall führt dies zu einer generellen Gebrauchsuntauglichkeit der Anwendung. Die Fachliteratur betrachtet für Eingaben in VR-Umgebungen allge-

[10] Vgl. http://www.roylongbottom.org.uk/linpack%20results.htm - 20.01.2014.

KOMPONENTE DES HANDHELDS	EINSATZ IM PROGRAMMIERSYSTEM	
Touch-Screen	**Nutzerinteraktion**	
	Eingabe Allgemeine Nutzerinteraktion, Programmverwaltung	**Ausgabe** Ausgabe von visuellem Feedback, Darstellung der Simulation in Augmented Reality
Kommunikation **(Bluetooth, WIFI, Internet)**	**Externe Kommunikationsschnittstellen** Industrieroboter, Motion-Tracking-System, Digitale Fabrik	
Kamera	**2D-Bildverabeitung** Bildakquisition, 2D-Handgestenerkennung, Marker-Erkennung Augmented Reality	

Tabelle 4.2: Standardkomponenten der Handheld-Geräte und deren potentielle Anwendung im Programmiersystem.

mein eine Reaktionszeit des Feedbacks von weniger als 75–120 *ms* als ausreichend [149, 150]. Die angeführten Studien belegen, dass unterhalb dieser Reaktionszeiten keine signifikanten Einschränkungen der Nutzbarkeit des Systems vorliegen und die Reaktion des Systems nicht als störend wahrgenommen wird. Vergleichbare Anwendungen in der CAVE unter Verwendung von Tracking benötigen nach DIN EN ISO 9241-171[11] ohne Beeinträchtigung der Usabilty mehr als 30 *Frames pro Sekunde* (fps). Für das vorliegende System soll für die prototypische Umsetzung zunächst eine Bildfrequenz von 15–30 fps angestrebt werden.

4.3.2 Wahrnehmung der Gesten

A) Sensorkonzept zur gestenbasierten Interaktion

Die Realisierung der markerlosen gestenbasierten Interaktion erfordert die Erkennung und kontinuierliche Verfolgung der Handbewegungen des Anwenders im Arbeitsbereich des Roboters. Darüber hinaus müssen Fingergesten erkannt werden, die für die Interaktion im Raum als Auslöser dienen. Aufgrund der angestrebten natürlichen Interaktion soll auf Hilfsmittel wie Datenhandschuhe oder Marker verzichtet werden.
Zur Aufnahme von Handposen und Handtrajektorien dient ein kamerabasiertes 3D-Motion-Tracking-System. Die berührungslose optische Messung ermöglicht die Bewegungs- und Interaktionsfreiheit des Anwenders in der Zellumgebung des Roboters. Neben der Kamera besteht das System aus einem Industrie-PC, auf dem eine Bildverarbeitungssoftware zur Erkennung und zum Tracking menschlicher Extremitäten appliziert ist. Bei der Wahl eines geeigneten Kamerasystems zum 3D-Tracking kann neben Stereo- und Multi-Vision-Ansätzen auch auf Ansätze des Time of Flight (ToF) und Structured Light (SL) zurückgegriffen werden. Der Einsatz eines Multi-Kamera-Systems bedeutet höhere Systemkosten und erfordert zumeist hohen Rechenaufwand zur Rekonstruktion der 3D-Szene. TOF-Kameras und SL-Kameras verfügen in der Regel über eine geringe

[11] Vgl. DIN EN ISO 9241-171:2008-10 [151].

laterale Auflösung, die eine Erkennung einzelner Finger nur bei geringer Entfernung zur Kamera ermöglicht [152]. Während eine Handerkennung und -verfolgung auf Basis dieser Sensorik weiterhin gegeben ist [153], ist diese für die Erkennung von Fingergesten ungeeignet. Ferner ist zu beachten, dass sich der Anwender im Arbeitsraum des Industrieroboters frei bewegen soll und dass beim Einsatz einer einzelnen Kamera etwaige Verdeckungen einzelner Finger durch Partien der Hand auftreten können.

Der Kinect-Sensor

Alternativ zu den industriellen Motion-Tracking-Systemen stellt der Kinect-Sensor der Firma Microsoft eine Innovation aus dem Consumer-Bereich dar. Der Preis des Sensors ist mit ca. 100 € im Vergleich zu industriellen 2,5D-Kameras sehr gering. Die Kamera besteht aus einer 2D-Farbkamera und einem Tiefensensor nach dem SL-Imaging-Prinzip. Dabei wird ein pseudozufälliges, intern bekanntes Punktmuster über einen Infrarotprojektor in die Umgebung projiziert. Anhand einer zusätzlichen Infrarotkamera kann mithilfe einer wahrgenommenen Veränderung des Musters und einer Ebenenprojektion auf die Tiefeninformationen zurückgeschlossen werden [154]. Das offizielle Programmier-Framework der Firma Microsoft sowie das Open-Source-Framework OpenNI ermöglichen ein robustes Tracking von Personen mit bis zu 30 Hz. Ergebnis des Personen-Tracking ist ein sogenanntes „Skeleton-Modell“ der Personen, welches aus charakteristischen Körperpunkten und deren räumlichen Verbindungen besteht. Die Absolutgenauigkeit liegt bei 1 m Abstand bei ca. ± 1 cm und steigt bis zum maximalen Abstand von 5 m auf ± 4 cm für die einzelnen Körperpunkte [10, 154, 155]. Zusätzlich lassen sich die Körperpunkte des Skeleton-Modells realen Körperpunkten des Anwenders räumlich nicht exakt zuordnen. Über das Skeleton-Modell hinaus existieren Anwendungen zum Tracking einzelner Finger, bis hin zur Fingerspitze. Diese funktionieren jedoch derzeit nicht robust, bzw. nur innerhalb eines geringen Abstandsbereichs zum Sensor [156].

Kombinatorischer Ansatz unter Verwendung von 2D-Fingergestenerkennung und 3D-Motion-Tracking

Im Rahmen einer prototypischen Implementierung soll ein kombinatorischer Ansatz zur Wahrnehmung der gestenbasierten Interaktion gewählt werden. Der Ansatz soll aktuelle Low-Cost-Systeme aus dem Consumer-Bereich, wie den Kinect-Sensor, einbeziehen und deren Eignung für den industriellen Einsatz im räumlichen Programmiersystem evaluieren.

Im Rahmen des Ansatzes sollen das externe Motion-Tracking-System zur Ermittlung der 3D-Position der Hand und das 2D-Bild der Handheld-Kamera zur Erkennung von Fingergesten eingesetzt werden. Beispielhafte Fingergesten zur räumlichen Definition und Manipulation des Roboterprogramms werden in Form einer Prinzipdarstellung gemäß Abbildung 4.22 veranschaulicht.

Die Motivation zum gewählten Ansatz entspringt vorrangig der Motivation einer wirtschaftlichen Umsetzung des Interaktionskonzepts. Bei einer natürlichen Ausübung der Interaktionsgesten über die Hand und gleichzeitiger Verifizierung der Gesten über die AR-Anwendung befinden sich die Finger in der Regel in einer definierten Pose innerhalb des Kamerabilds. Somit kann die Erkennung der Finger über das 2D-Bild ohne ein hochgenaues externes 3D-Messsystem erfolgen. Ein weiterer Vorteil dieser Methodik liegt darin, dass bei Single-Kamera-Ansätzen für das 3D-Motion-Tracking eventuelle Verdeckungen einzelner Finger vermieden werden können, da der Nutzer die

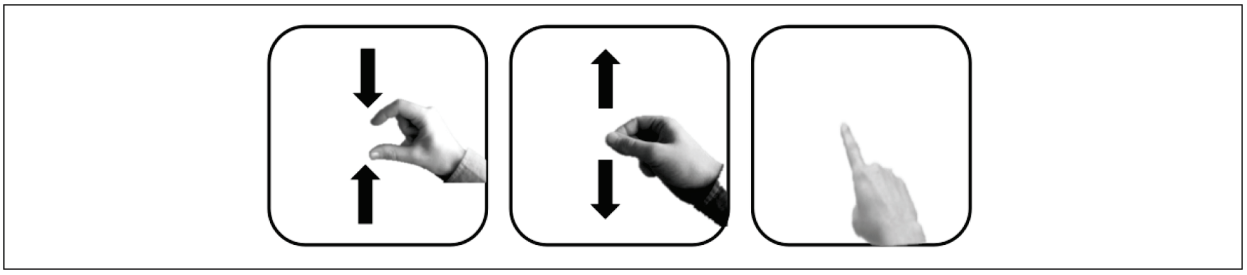

Abbildung 4.22: Fingergesten zur räumlichen Definition und Manipulation: Greifen (links), Loslassen (Mitte) und Zeigegeste (rechts).

Kamera über die Haltung des Handheld-Geräts ggf. nachpositionieren kann. Zudem verringern sich mit einer Reduzierung der Dimension der Rechenaufwand des Finger-Tracking sowie die benötigte Komplexität des Algorithmus zur Fingergestenerkennung. Abbildung 4.23 veranschaulicht die kombinatorische Erfassung von 3D-Koordinaten der Hand über das externe Motion-Tracking-System sowie die 2D-Fingergestenerkennung über das Handheld-Gerät zur Realisierung der räumlichen Interaktion.

Randbedingungen des 3D-Motion-Tracking

Ziel des 3D-Motion-Tracking ist die Ermittlung der 3D-Koordinaten der Hände. Diese Koordinaten werden über den zugehörigen Industrie-PC kontinuierlich an die mobile Programmierumgebung übersendet und dienen dort der interaktiven Definition und Manipulation von räumlichen Objekten. Zur Abdeckung des angestrebten Interaktionsbereiches ist das Kamerasystem an geeigneter Stelle zu positionieren. Die Wahl einer geeigneten Position und Ausrichtung ist individuell zu lösen, da es aufgrund variabler Umgebungsparameter einer Abstimmung mit dem spezifischen Zellaufbau bedarf. Zu betrachten sind neben geometrischen Gegebenheiten Störeinflüsse aus Umgebung und

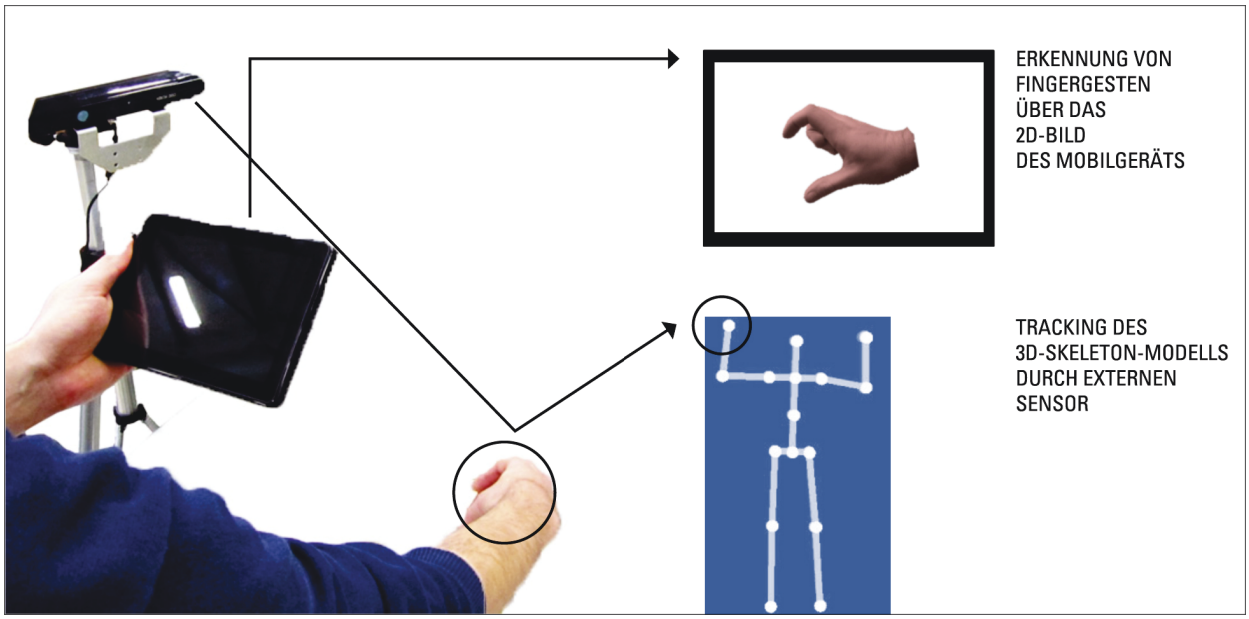

Abbildung 4.23: Kombinatorische Verbindung von 2D-Fingergestenerkennung und 3D-Motion-Tracking.

Prozess, die sich negativ auf die Funktionalität und Qualität der kamerabasierten Bildakquise und Tiefenwahrnehmung auswirken. Die wichtigsten Einflüsse bilden:

1. Maße des Arbeitsbereichs und des benötigten Interaktionsbereichs des Roboters zur Erfüllung der betrachteten Produktionsaufgaben,
2. Zugänglichkeit des Arbeitsbereichs, möglicherweise wenig Platz zum Aufstellen eines Kamerasystems,
3. partielle Verdeckungen, bspw. durch Peripherieeinrichtungen oder den menschlichen Körper sowie
4. Abschattungen und prozessbedingte optische Einflüsse, bspw. beim Schweißen.

Um den Interaktionsbereich bei großen Robotersystemen zu erhöhen, muss das Sensorkonzept gegebenenfalls auf einen Multi-Kamera-Ansatz mit entsprechender Datenfusion erweitert werden.

B) Realisierung der Fingergestenerkennung

In [147] und [157] wurde durch den Autor der vorliegenden Arbeit zunächst ein 2D-Tracking-Algorithmus zur Erkennung von Fingergesten auf Basis eines nicht-parametrierten Hautverteilungsmodells und der Rückprojektion jedes einzelnen Pixels im Bild vorgestellt. Dieser Ansatz erwies sich in der Praxis als wenig robust und bei den beschränkten Rechenkapazitäten moderner Handheld-Geräte als zu langsam. Die angestrebte Bildfrequenz von 15–30 fps[12] konnte nicht erreicht werden.
Demnach besteht die Motivation, einen gegen Umgebungseinflüsse, bspw. wechselnde Beleuchtungssituationen, robusten und gleichzeitig schnellen Algorithmus zur Fingergestenerkennung zu entwickeln. Im Rahmen dieser Arbeit soll ein Ansatz von M. Tosas [158] für das robuste Hand-Tracking auf Basis des Conditional Density Propagation (Condensation-)Algorithmus für die effiziente Nutzung auf Handheld-Geräten angepasst werden. In [159] wird dieser Ansatz durch den Autor der vorliegenden Arbeit tiefergehend vorgestellt sowie für die Anwendung bei begrenzten Ressourcen optimiert. Die Realisierung der Gestenerkennung im 2D-Bild lässt sich in die Schritte der Hauterkennung, des Tracking durch den Condensation-Algorithmus und der Interaktionserkennung gliedern. Neben der Entwicklung eines effizienten Klassifikators zur Hautsegmentierung sollen die Arbeiten von Tosas gezielt um weitere Verbesserungen hinsichtlich des Einsatzes im Rahmen des räumlichen Programmiersystems erweitert werden.

Hauterkennung
Die Hauterkennung ist eine fundamentale Anforderung für das markerlose Tracking von menschlichen Händen. Neben dem Hand-Tracking gibt es eine weite Reihe von Anwendungsgebieten im Bereich der Überwachung und Mensch-Maschine-Interaktion, bspw. speziell in der Mensch-Roboter-Kooperation [160]. Entsprechend der Menge an Anwendungen existiert keine allgemeine Lösung für diese spezielle Problemstellung der Bildverarbeitung. Der Fachliteratur sind lediglich zugeschnittene Lösungen zu entnehmen. Die Hauptcharakteristik, welche zur Unterscheidung von Hautregionen und Nicht-Hautregionen in einem 2D-Bild genutzt wird, ist die Hautfarbe. Die

[12] Vgl. Kapitel 4.3.1.

Hauterkennung verfolgt demgemäß das Ziel, adäquate Regeln zur pixelbasierten Unterscheidung bereitzustellen. In der Praxis kann die Entwicklung eines entsprechenden Klassifikators in zwei Schritte unterteilt werden:

1. Wahl eines Farbraums und
2. Definition/Wahl eines Hautmodells.

Bezüglich des Hautmodells existieren nach [161] verschiedene Ansätze:

1. nicht-parametrierte Hautverteilungsmodelle,
2. parametrierte Hautverteilungsmodelle,
3. explizit definierte Hautregionen sowie
4. dynamische Hautverteilungsmodelle.

Nicht-parametrierte Hautverteilungsmodelle nutzen Trainingsdaten für die Erstellung einer sogenannten „Skin Probability Map“ (SPM) [162, 163]. Die SPM beschreibt die Wahrscheinlichkeiten, mit denen spezifische Farbwerte der Hautfarbe zuzuordnen sind. Typische Umsetzungen dieser Modelle verwenden Farbhistogramme zur Beschreibung der Häufigkeiten des Auftretens von Farbwerten in Trainingsdaten. Werden die Werte des Histogramms normalisiert, erhält man die Wahrscheinlichkeitsverteilung. Eine weitere Wahrscheinlichkeitsverteilung kann auf Histogramm-Basis für die Nicht-Hautfarbepixel erzeugt werden. Mit diesen beiden Wahrscheinlichkeitsverteilungen lässt sich eine Bayes-Klassifikation zur Hautfarbenerkennung vornehmen [161]. Der größte Vorteil dieser Methode besteht in ihrer hohen Leistungsfähigkeit und ihrem schnellen Durchlauf. Allerdings ist im Gegensatz eine hohe Speicherbelegung zur Definition der Histogramme notwendig. Der Erfolg dieser Methode hängt zudem grundsätzlich von der Qualität der Trainingsdaten und deren Übereinstimmung mit der Hautfarbe des aktuellen Nutzers ab.
Parametrierte Hautverteilungsmodelle nutzen ebenfalls Trainingsdaten. Sie beschreiben die Farbverteilung der Haut allerdings anhand von Gauß-Kurven. Die Funktionsparameter des Hautmodells können anhand der Trainingsdaten gebildet oder angepasst werden. Somit können auch unvollständige Trainingsdaten zur Definition eines Klassifikators mit hinreichender Leistungsfähigkeit dienen [161].
Explizit definierte Hautregionen sind eine einfache und intuitive Art der Beschreibung für die Verteilung der Hautfarbe im Farbraum. Nach diesem Ansatz werden hautfarbene Regionen in Trainingsdaten anhand von adäquaten Regeln lokalisiert. Ein derartiger Klassifikator zeichnet sich vor allem durch seine Schnelligkeit und durch einen sehr geringen Speicherplatzbedarf aus. Eine Hauptherausforderung bei der Entwicklung eines entsprechenden Klassifikators liegt in der empirischen Ermittlung eines adäquaten Farbraums und in der Definition von Grenzwerten für die Bereiche der Hautfarbe im Farbraum. In [164] wird diesbezüglich ein adäquates Verfahren vorgestellt.
Dynamische Hautverteilungsmodelle wurden speziell für die Hautdetektion innerhalb von Tracking-Anwendungen entwickelt. Im Vergleich zu den statischen Methoden der Bildauswertung wird von höheren Anforderungen an die Hauterkennung ausgegangen. Dies betrifft primär die Möglichkeit der dynamischen Anpassung des Modells an veränderte Lichtverhältnisse. Ein zu-

sätzlicher Initialisierungsschritt kann das grundlegende Hautverteilungsmodell entsprechend der Umgebungsumstände und der individuellen Haut des Nutzers anpassen. Dementsprechend kann auch der automatische Weißabgleich kompensiert werden, der auf vielen Mobilgeräten durchgeführt wird und in der Regel nicht zu deaktivieren ist.
Im Rahmen dieser Arbeit soll ein Klassifikator betrachtet werden, welcher für die Echtzeitanwendungen auf modernen Smartphones und Tablet-PCs geeignet ist. Der Klassifikator soll ein dynamisches Hautverteilungsmodell beschreiben und sich durch geringen Speicherbedarf und Rechenaufwand auszeichnen.

Condensation-Algorithmus
Blake und Isard [165] entwickelten in Form des Condensation-Algorithmus ein Partikelfilter, welches dafür zugeschnitten ist, Konturen in Videosequenzen zu verfolgen. Das Partikelfilter ist ein stochastisches Verfahren zur Zustandsschätzung in dynamischen Prozessen. Der Condensation-Algorithmus wurde bereits in verschiedenen Bereichen der Bildverarbeitung erfolgreich angewandt, bspw. [158, 166, 167]. Der entscheidende Vorteil gegenüber dem weit verbreiteten Kalman-Filter besteht im Bereich des Tracking in der Möglichkeit, verschiedene Hypothesen über mehrere Zeitschritte zu betrachten. Demzufolge wird während des Tracking das Risiko verringert, durch Störungen das Zielobjekt zu verlieren. Um des Weiteren eine Fingergestenerkennung zu realisieren, ist es das Ziel des Condensation-Algorithmus, zunächst die Handkontur zu verfolgen. Genauer bedeutet dies, dass in jedem Bild ein sogenannter „State Vector“ $\boldsymbol{x}_t$ mit optimaler Übereinstimmung von hypothetischer zu realer Handkontur gefunden werden muss. Zu diesem Zweck müssen in jedem Schritt Messungen anhand der Hauterkennung für verschiedene hypothetische Konturen durchgeführt werden. Die Bewertung eines möglichen Zustands in einem Zeitschritt t bezüglich aller bisherigen Messungen z wird durch die bedingte Wahrscheinlichkeit $p_t(\boldsymbol{x}_t|z_t)$ ausgedrückt. Diese gibt die Wahrscheinlichkeitsverteilung an, dass $\boldsymbol{x}_t$ der Handkontur unter Berücksichtigung der Beobachtungen z_t entspricht. Wird das Bayes-Theorem zur Beschreibung dieser Problemstellung angewandt, resultiert dies in:

$$p_t(\boldsymbol{x}_t|z_t) = \frac{p_t(z_t|\boldsymbol{x}_t) \cdot p_{t-1}(\boldsymbol{x}_t|z_{t-1})}{p_t(z_t)} \tag{4.1}$$

$p_{t-1}(\boldsymbol{x}_t|z_{t-1})$ wird als „Probability Distribution“ bezeichnet und gibt die Übereinstimmung des aktuellen Zustands mit den Bildmerkmalen des vorherigen Zeitschritts an. $p_t(\boldsymbol{x}_t|z_t)$ ist die „Observation Density“, welche die hypothetische Kontur anhand der aktuellen Bildmerkmale bewertet. Die Berechnung dieses Werts wird durch ein Messmodell realisiert. Das Hinzufügen eines weiteren Terms $p_t(\boldsymbol{x}_t|x_{t-1})$ ergibt:

$$p_t(\boldsymbol{x}_t|z_t) = \frac{p_t(z_t|\boldsymbol{x}_t) \cdot \int_{\boldsymbol{x}_{t-1}} p_t(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \cdot p_{t-1}(\boldsymbol{x}_{t-1}|z_{t-1})}{p_t(z_t)} \tag{4.2}$$

Das eingesetzte Partikelfilter hat nun die Aufgabe, $p_t(\boldsymbol{x}_t|z_t)$ (s. Formel 4.2) möglichst exakt zu berechnen bzw. abzuschätzen. Dazu nutzt der Condensation-Algorithmus gewichtete Partikelsätze. Die Partikelsätze sind Listen von $n \in \mathbb{N}$ Paaren $(\boldsymbol{x}_i, \pi_i)$, $i = 1 \ldots n$, bestehend aus dem State Vector $\boldsymbol{x}_t$ und seinem Gewichtungsfaktor $\pi_i \in [0,1]$, und $\sum_{i=1}^{n} \pi_i = 1$. Diese Paare werden als

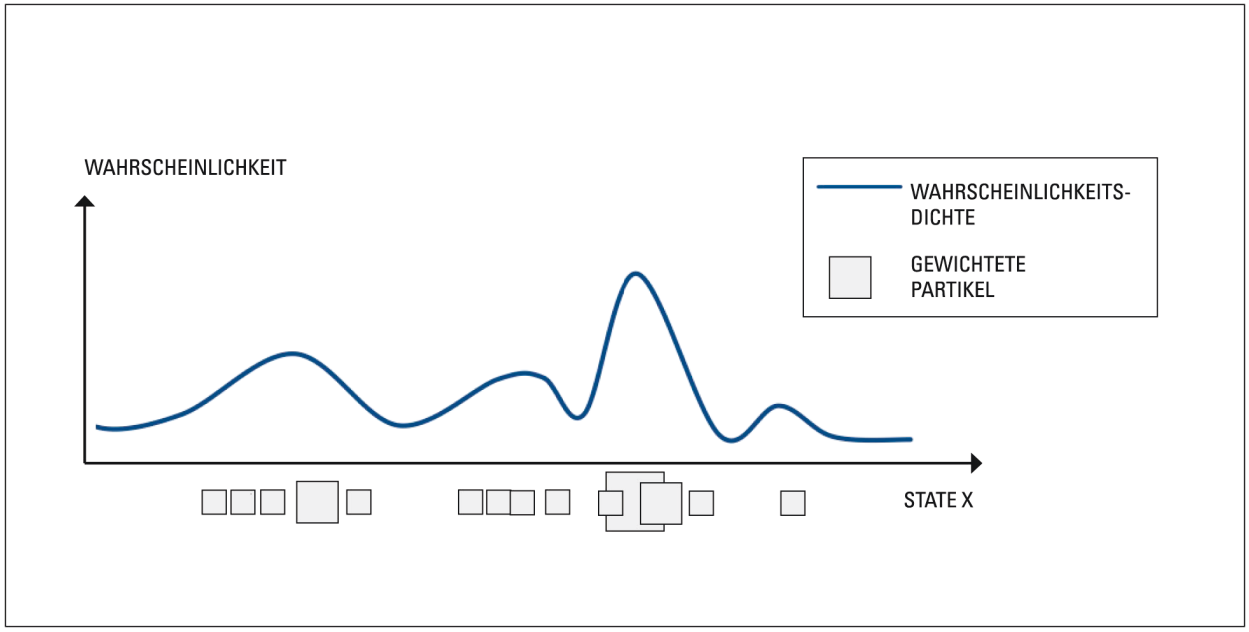

Abbildung 4.24: Gewichteter Partikelsatz zur Abschätzung der Wahrscheinlichkeitsverteilung, Quelle: [168].

Partikel bezeichnet und repräsentieren jeweils einen hypothetischen Konturzustand. Anhand der Partikel ist der Condensation-Algorithmus in der Lage, multimodale Verteilungsfunktionen, die mehrere lokale Maxima besitzen, angemessen abzuschätzen (s. Abb. 4.24).

Tracking zusammengesetzter Körper

In den vorherigen Schritten des Tracking-Algorithmus wurde noch von der Annahme ausgegangen, dass es sich bei der Hand um einen einzelnen, starren Körper handelt. In der Realität besteht die menschliche Hand aus verschiedenen Gelenken mit mehreren Bewegungsfreiheitsgraden. Vor allem für die Erkennung von Fingergesten ist es demnach unerlässlich einzelne Finger zu unterscheiden und diese separat zu verfolgen. Das Tracking-Problem muss demnach auf bewegliche bzw. gelenkige Objekte erweitert werden. Zu diesem Zweck wird die Hand als Baumstruktur betrachtet. Aus Gründen der Vereinfachung wird im Rahmen dieser Arbeit die Betrachtung auf einen Finger beschränkt. Somit ergibt sich anstatt der Baumstruktur die vereinfachte Struktur einer mehrgliedrigen Kette.

Partition Sampling: MacCormick und Isard stellen in [169] das Partition Sampling als effiziente Methode für das Tracking von gelenkigen Objekten vor. Im Allgemeinen verfügen die menschlichen Hände über eine hohe Anzahl an Bewegungsfreiheitsgraden. Dies würde nach dem bisher vorgestellten Tracking-Ansatz in einem komplexen State Vector resultieren. Obwohl der Condensation-Algorithmus prinzipiell in der Lage ist, komplexe Zustandsvektoren zu verarbeiten, müsste die Anzahl der benötigten Partikel für das Tracking komplexer Gelenkstrukturen drastisch erhöht werden. Dies hätte einen nicht hinnehmbaren Effekt auf die benötigte Rechenleistung. Der Ansatz des Partition Sampling verfolgt hingegen die Annahme, dass die einzelnen Gelenke bzw. Partitionen eines beweglichen Objekts separat als einzelne Konturen mit getrennten Partikelsät-

zen verfolgt werden. Der Condensation-Algorithmus wird auf diese Partitionen in hierarchischer Abfolge der Struktur angewendet.

Sweep Implementation: Aufbauend auf dem Ansatz des Partition Sampling entwickelte Tosas [158] die Methode Sweep Implementation. Diese ist speziell zugeschnitten auf das Tracking von menschlichen Fingern. Zunächst wird die Hand in verschiedene Teile aufgeteilt: Handballen und Finger. Anschließend wird das Tracking in hierarchischer Abfolge für die einzelnen Elemente unter speziellen Voraussetzungen durchgeführt. Bei der Verfolgung der Finger wird weiterführend davon ausgegangen, dass die Suche nach der Fingerkontur auf einen begrenzten Bereich um die vorherige Fingerposition beschränkt werden kann. Bei der Bewegung wird zunächst nur die Rotation der Fingerkontur um einen festen Basispunkt im Handballen betrachtet. Anschließend wird die Länge des Fingers durch parallel zur Fingerrichtung angeordnete Messlinien bestimmt. Abbildung 4.25 veranschaulicht das Vorgehen anhand einer Visualisierung der Rotation eines Fingers sowie der Messlinien im Kamerabild.

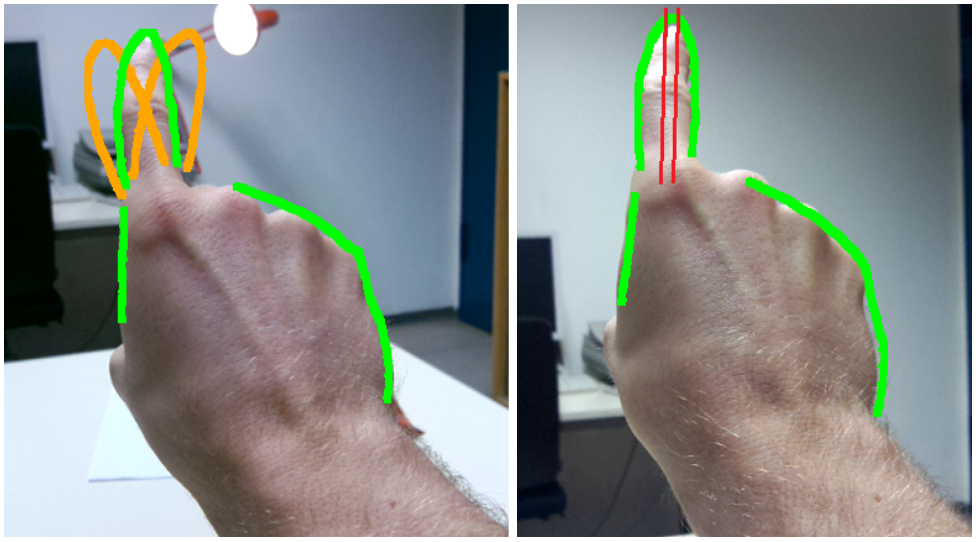

Abbildung 4.25: Kontur des Handballens mit hypothetischen Fingerkonturen (links) und Handkontur mit Messlinien zur Ermittlung der Fingerlänge (rechts).

Interaktionserkennung

Für die Umsetzung der Fingergestenerkennung werden im Rahmen dieser Arbeit zwei Arten von Gesten adressiert: Klicken und Greifen. Diese sollen das räumliche Erstellen und Manipulieren von virtuellen Objekten auf unterschiedliche Art ermöglichen. Auf der einen Seite können über das räumliche Klicken schnell neue Objekte, bspw. einzelne Posen, im Raum angelegt werden, zudem können bestehende Objekte ausgewählt werden. Auf der anderen Seite lässt sich über die Greifgeste eine natürlich-räumliche Manipulation von virtuellen Objekten realisieren, bspw. durch Verschieben. Die Gestenerkennung soll sich aus der geometrischen Anordnung verschiedener Handpartitionen im Bild ohne Implementierung von komplexen Algorithmen der Mustererkennung ableiten lassen.

4.3.3 Programmierumgebung

Die Programmierumgebung bündelt zentrale Funktionen des Programmiersystems auf dem Mobilgerät und stellt Schnittstellen für die externe Kommunikation bereit. Die entsprechenden Softwaremodule bilden die Grundlage für die funktionale Umsetzung des Interaktionskonzepts. Neben der Datenhaltung und Programmverwaltung umfasst die Programmierumgebung Basisfunktionen zur Realisierung der Visualisierung und Simulation sowie Schnittstellenfunktionen zur Ein- und Ausgabe von Roboterprogrammen. Abbildung 4.26 skizziert die wichtigsten Module der Programmierumgebung und deren Schnittstellen.

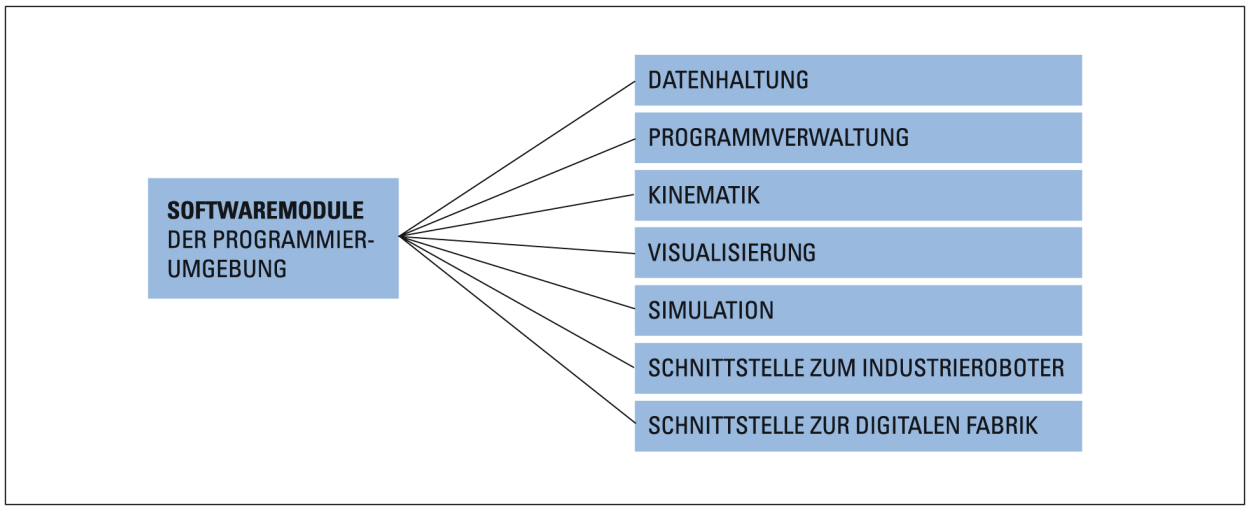

Abbildung 4.26: Module der Programmierumgebung auf dem Programmiergerät.

A) Datenhaltung und Programmverwaltung

Datenhaltung
Die Programmierumgebung dient der Verwaltung von Aufgabendefinitionen, bewegungsorientierten Programmen und Makros sowie Posen. Zudem ist die Programmrepräsentation die Grundlage für die räumliche Visualisierung und Simulation der Programme. Da eine Neueingabe von Posen und Programmen unerwünscht ist, sollen plattformunabhängige, relationale Datenbanken genutzt werden, in denen Programmparameter hinterlegt werden.

Programmverwaltung
Auf Basis einer neutralen Programmrepräsentation wird dem Nutzer eine Entwicklungsumgebung zur Definition, Evaluation und Manipulation von Roboterprogrammen bereitgestellt. Auf diesen Grundfunktionen sollen die Methoden der räumlichen Interaktion aufsetzen. Als Basis der Programmdarstellung für den Nutzer soll eine Benutzeroberfläche dienen, welche den sequentiellen Programmablauf textuell oder grafisch darstellt.

Programmdarstellung
Zur bewegungsorientierten Modellierung wird das Programm aus verschiedenen sequentiellen Einzelbewegungen zusammengesetzt. Einer Bewegung müssen Interpolation, Geschwindigkeit und

Beschleunigung sowie Posen zugewiesen werden. Auf die Darstellung von Bedingungen, Operatoren, Schleifen etc. soll im Rahmen dieser Arbeit aus Gründen der Vereinfachung zunächst verzichtet werden.
Zur Definition von abstrakten Aufgaben wird die Darstellung der Programme um eine übergeordnete, parametrierbare Ebene mit der Bezeichnung „Task“ erweitert. Ein Task kann manuell durch verschiedene Unterprogramme, sogenannte „Skills“ gebildet werden. Diese Art der Darstellung erfolgt in Anlehnung an [170]. Die Skills stellen Teilaufgaben zum Erreichen der Gesamtaufgabe dar. Die Gestaltung der Datenbanken ist um die entsprechende Ebene zur Abbildung der aufgabenorientierten Programmabstraktion zu erweitern. Diese Art der Darstellung ermöglicht die Definition von festen Programmmakros für sich wiederholende Programmabläufe in unterschiedlichem Kontext. Die Tasks können von einem Experten aufgabenspezifisch angelegt und im Rahmen eines PbD-Ansatzes zur aufgabenorientierten Programmierung automatisiert parametriert werden.

B) Kinematik, Visualisierung und Simulation der Roboterprogramme

Kinematik
Zur realistischen Darstellung der Kinematik des virtuellen Robotermodells in der Simulation dient die Abbildung steuerungstechnischer Grundfunktionen zur Ansteuerung des Modells. Auf eine weiterführende realistische Abbildung der Dynamik mit Berücksichtigung von Massenträgheit sowie Zentripetal-, Coriolis- und Reibungskräft soll zunächst aus Gründen der Vereinfachung verzichtet werden. Eine zukünftige Erweiterung der Simulation kann ggf. die Schnittstellendefinition zur Realistischen Robotersimulation [171] mit einbeziehen. Der Aufbau des Programmier- und Simulationssystems erfolgt analog zur Struktur der Ablaufsteuerung von Industrierobotern[13]. Die Basis der Bewegungssimulation bildet das Roboterprogramm, welches zunächst durch einen Interpreter ausgelesen wird. Weiterführend sind Funktionen für die Durchführung gängiger Interpolationsarten sowie zur Vorwärts- und Rückwärtskinematik vorzusehen. Insbesondere soll eine Spline-Interpolation zur Darstellung komplexer Freiformkurven genutzt werden. Zur Modellierung der geometrischen Abhängigkeiten der Roboterkinematik soll auf die Denavit-Hartenberg-Konvention (DH-Konvention) zurückgegriffen werden.

Programmvisualisierung und Bewegungssimulation
Der virtuelle Industrieroboter ist anhand von CAD-Daten der einzelnen Gelenke und des Endeffektors zu modellieren. Die Modellierung als Einzelobjekte ermöglicht eine relative Bewegung der Einzelkomponenten entsprechend dem Verhalten des realen Roboters. Zur Zusammensetzung des Gesamtmodells des Roboters dienen definierte Koordinatensysteme der CAD-Daten und die geometrischen Parameter der DH-Darstellung. Die Bewegungssimulation des virtuellen Roboters erfolgt anhand der Werte der Feininterpolation, welche als Sollvorgaben für die Rotation der einzelnen Achsen dienen.
Die bewegungsorientierte Programmvisualisierung dient der Darstellung von Posen und Trajektorien. Während Posen sich aus dem sequentiellen Programmablauf mithilfe des Interpreters extrahieren lassen, werden die Trajektorien anhand der Ausgangswerte des Interpolators erzeugt. In Abgrenzung zur Bewegungssimulation kann für die visuelle Darstellung ein differierender Interpo-

[13] Vgl. Kapitel 2.2.1.

lationstakt gewählt werden, bspw. um den Rechenaufwand für die Visualisierung zu beschränken. Für die Visualisierung der Orientierung der Posen soll auf eine anschauliche Darstellung[14] zurückgegriffen werden. Die aufgabenorientierte Programmvisualisierung ist anwendungsspezifisch durchzuführen.

C) Schnittstellen zum Programmaustausch

Das angestrebte Programmiersystem soll prozessnah und simulationsgestützt am Robotersystem angewendet werden, ohne während des Programmierprozesses auf die Robotersteuerung oder den Roboter zurückzugreifen. Eine eindeutige Zuordnung zu den Online- oder Offline-Programmiersystemen lässt sich demnach nicht treffen. Um eine effiziente Anwendbarkeit herzustellen, ist eine drahtlose Anbindung an die Robotersteuerung und die betrieblichen Informations- und Kommunikationsstrukturen unerlässlich (s. Abb. 4.27).

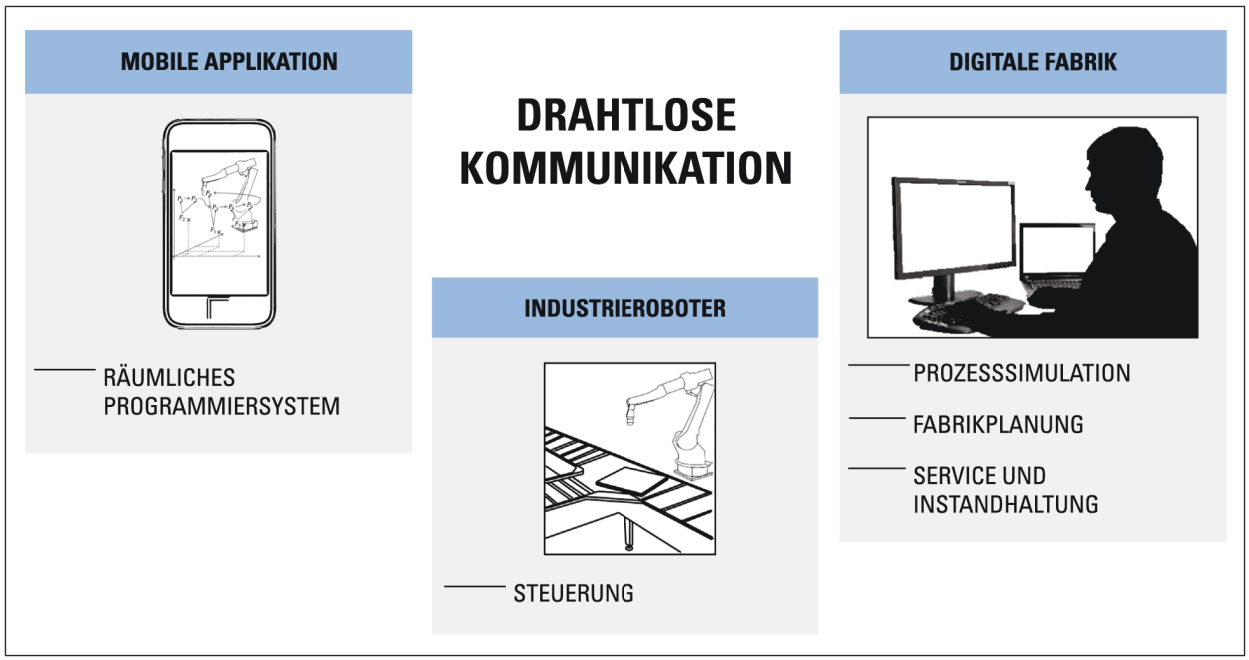

Abbildung 4.27: Drahtlose Anbindung des Programmiersystems an die reale Robotersteuerung und an die Digitale Fabrik.

Schnittstelle zum Industrieroboter

Zur Übertragung und Ausführung der Roboterprogramme vom Mobilgerät auf die Industrierobotersteuerung wird eine vereinheitlichte Schnittstelle angestrebt. Der Grund dafür liegt darin, dass nahezu jeder Industrieroboterhersteller über eine eigene Programmiersprache und über eigene Schnittstellen zur Ansteuerung des Roboters verfügt. Dabei unterscheiden sich die bereitgestellten externen Schnittstellen in ihrem Funktionsumfang teilweise stark voneinander[15]. Um eine Lösung mit breiter Anwendbarkeit und einfacher Implementierung bereitzustellen, gilt es, eine Schnittstelle zu definieren, welche unabhängig vom Kommunikationsmedium sowie von der

[14] Vgl. Definition der Endeffektororientierung in Kapitel 4.2.2.
[15] Vgl. Kapitel 2.2.1.

STOP			
TYP	NAME	WERT	BESCHREIBUNG
uint16	id	0	commandID
int8[max_axes]	axes	[0,1]	0: Zustand unverändert 1: Stoppe einzelne Achse
GRIP			
TYP	NAME	WERT	BESCHREIBUNG
uint16	id	1	commandID
uint8	grip	[0,1,n,255]	0: Zustand unverändert 1: Greifer geschlossen n: diskrete Greiferstellung oder Kraft 255: Greifer geöffnet
GETCURRENTPOSITION			
TYP	NAME	WERT	BESCHREIBUNG
uint16	id	6	commandID
int16	frameid		Referenz-Frame
uint8	type	[0,1]	Bit 0: 0 = Antwort 1 = Anforderung Bit 1: 0 = Kartesisch 1 = Achswerte
uint8	length		Anzahl der Rückgabewerte
int32[length]	pos		Position

Tabelle 4.3: Objektorientierte Struktur der einheitlichen Roboterschnittstelle: exemplarische Schnittstellenbefehle.

Roboterprogrammiersprache genutzt werden kann. Dies meint, dass die Programmrepräsentation auf dem Mobilgerät auf beliebige Industrierobotersteuerungen mit möglichst geringem Anpassungsaufwand übertragen werden kann. Auf eine Anpassung der Programmierumgebung auf dem Mobilgerät soll dabei gänzlich verzichtet werden.
Im Folgenden wird die angestrebte Funktionalität der Schnittstelle kompakt dargelegt. Detaillierte Ausführungen zur Schnittstellendefinition, zur Umsetzung sowie weitere Beispielanwendungen können der diesbezüglichen Veröffentlichung des Autors der vorliegenden Arbeit [172] entnommen werden.
Hauptbestandteil der Schnittstelle ist die Definition generischer Steuerungsbefehle. Diese bilden eine objektorientierte Repräsentation gängiger Industrieroboterkommandos und können mit nahezu beliebigen Programmiersprachen objektorientiert abgebildet werden. Zum Umfang der Befehle gehören Bewegungsanweisungen, Anweisungen für Werkzeuge und Greifer sowie das Abfragen bzw. Setzen von digitalen sowie analogen Eingängen und Ausgängen. Ergänzend wird eine Abfrage der aktuellen TCP-Pose des Roboters vorgesehen. In Tabelle 4.3 ist die Struktur exemplarischer Schnittstellenbefehle dargestellt. Eine umfangreiche Darstellung der Schnittstellendefinition ist in Anhang C zu finden.
Aufseiten der Robotersteuerung besteht die Schnittstelle aus einem Interpreter-Programm. Dieses läuft in Form von nativem Code zyklisch auf der jeweiligen Robotersteuerung und hat eine defi-

nierte, herstellerunabhängige Struktur. Die Aufgabe des Programms besteht darin, die generischen Steuerungsbefehle zu empfangen, zu interpretieren und in die herstellerspezifischen Entsprechungen zu übersetzen. Zur Kommunikation können die physischen Kommunikationsschnittstellen der jeweiligen Robotersteuerungen, wie Ethernet, RS-232 sowie beliebige Feldbussysteme, genutzt werden. Die gesendeten und empfangenen Datenpakete sind demnach sowohl unabhängig vom Roboterhersteller und von der Programmiersprache als auch von den genutzten Kommunikationsmedien. Folglich ist zur Übertragung der externen Roboterprogrammierung auf ein neues Robotersystem lediglich eine Anpassung des Interpreter-Programms an die aktuelle Programmiersprache sowie die aktuelle Kommunikationsschnittstelle erforderlich. Als eigentliche Steuerungs- und Programmiergeräte des Roboters können demnach neben Handheld-Geräten auch beliebige Steuerungssysteme, bspw. SPS oder Prozessleitsysteme, eingesetzt werden. Abbildung 4.28 verdeutlicht die Struktur der Kommunikation der Schnittstelle.

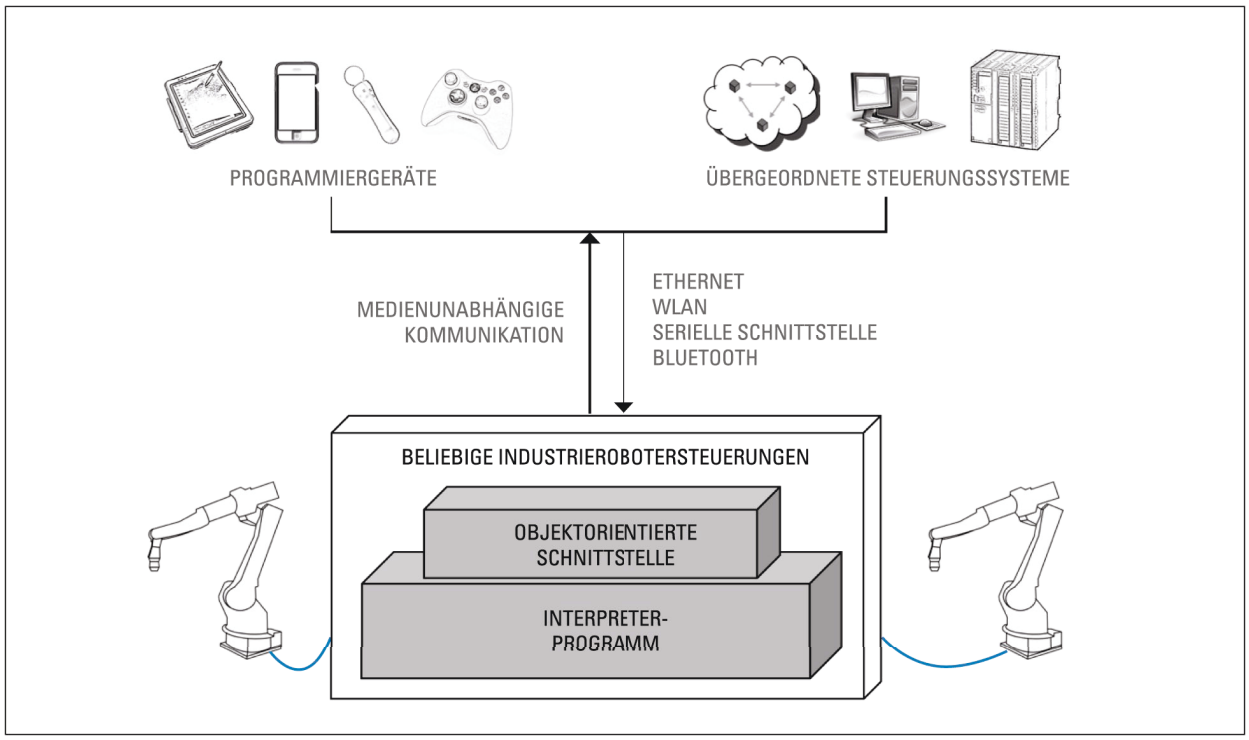

Abbildung 4.28: Topologie der vereinheitlichten Kommunikationsschnittstelle zur Programmübertragung auf die Robotersteuerung.

Schnittstelle zur Digitalen Fabrik

Die Digitale Fabrik als Zusammenschluss verschiedener Simulationstechnologien für die Produktionsplanung wird stellvertretend für die betrieblichen Informations- und Kommunikationsstrukturen betrachtet. Das angestrebte Programmiersystem soll Schnittstellen bereitstellen, welche den beidseitigen Austausch von Daten zwischen mobilem Programmier- und Simulationssystem und der Digitalen Fabrik ermöglichen. In der Regel verfügen die Werkzeuge der Digitalen Fabrik über eine einheitliche Datenbasis. Auf dieser Basis soll zunächst ein flexibler Austausch von Roboterprogrammen ermöglicht werden. Neben zentraler Archivierung und dezentraler Programmanpassung ermöglicht die angestrebte Schnittstelle prinzipiell auch die weiterführende Nutzung von

hybriden bzw. kooperativen Programmieransätzen. Eine Datenrückführung aus Simulation und Programmierung bildet zudem die Basis für eine hohe Transparenz und eine weiterführende Optimierung der beteiligten produktionstechnischen Prozesse.
Als Kommunikationsschnittstelle sollen drahtlose Verbindungen auf Basis des Transmission Control Protocol/Internet Protocol (TCP/IP) genutzt werden. Somit kann der Austausch bequem über firmeninterne Lokalnetzwerke erfolgen. Sind entsprechende Programmierschnittstellen für die Simulationstools vorhanden, soll die Kommunikation direkt in die Anwendung integriert werden. Beide Seiten sind mit Push/Pull-Funktionen auszustatten, die es erlauben, Daten zu senden oder abzufragen. Des Weiteren muss zum Übersetzen des Programms ein Post- bzw. Präprozessor erstellt und auf beiden Seiten der Schnittstelle implementiert werden. Dies ermöglicht die Integration und die Austauschbarkeit des Roboterprogramms. Zur prototypischen Umsetzung sollen zunächst nur die in der App bereits vorhandenen Befehle der Schnittstelle abgebildet werden. Über Roboterprogramme hinaus ist eine Ausweitung der Schnittstelle zum Austausch weiterer Daten zu prüfen.

4.3.4 Koordinatentransformationen und Inbetriebnahme

Zur Übertragung von Koordinaten zwischen den komponentenspezifischen KS des Programmiersystems müssen die geometrisch-räumlichen Beziehungen zwischen diesen bekannt sein. Zur Darstellung der räumlichen Abhängigkeiten der KS dienen homogene Koordinatentransformationen. Während die Positionen der Hand im KS des Motion-Tracking-Systems und die Position des Handheld-Geräts, genauer der Kamera, im Marker-KS durch Bildverarbeitungsalgorithmen dynamisch bestimmt werden, sind die Transformationen zwischen Industrieroboter und Motion-Tracking-System sowie die Transformation zwischen Industrieroboter und Marker-KS statisch zu ermitteln. Abbildung 4.29 zeigt den Aufbau der verschiedenen Koordinatensysteme des Systems.

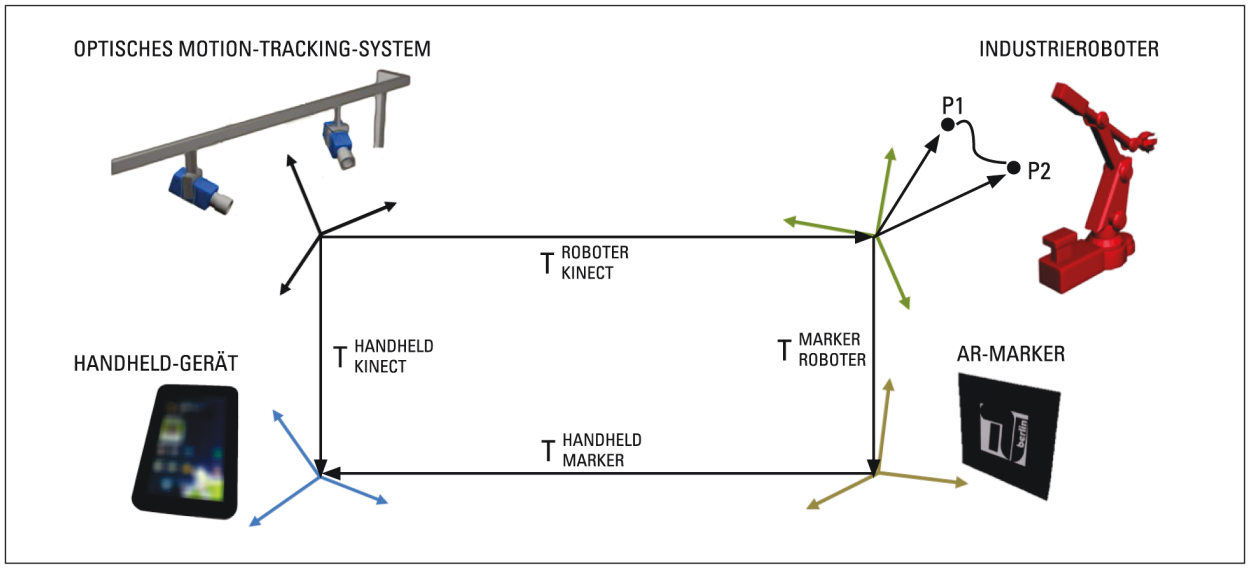

Abbildung 4.29: Räumliche Beziehung der Koordinatensysteme des Programmiersystems.

Eine entsprechende Mess- bzw. Kalibriermethode dient im Rahmen der Inbetriebnahme des Systems zur Ermittlung der statischen Koordinatentransformationen. Die allgemeine mathematische Abhängigkeit zwischen den Koordinatensystemen von Roboter, AR-Marker und Motion-Tracking-System (Kinect) im 3D-Raum kann dabei durch folgende Verkettung der homogenen Transformationsmatrizen beschrieben werden:

$$\boldsymbol{T}_{Roboter}^{Kinect} = \boldsymbol{T}_{Roboter}^{Marker} \cdot \boldsymbol{T}_{Marker}^{Kinect} \tag{4.3}$$

Im Programmiersystem dient die Transformationsmatrix zwischen Roboter und Motion-Tracking-KS $\boldsymbol{T}_{Roboter}^{Kinect}$ zur Umrechnung der Handkoordinaten in Koordinaten des Roboter-KS. Für die Ermittlung der Handposition im Roboter-KS $\boldsymbol{p}_{Roboter}$ gilt in Abhängigkeit von der Position des Motion-Tracking-Systems $\boldsymbol{p}_{Kinect}$ folgende Berechnungsvorschrift:

$$\boldsymbol{p}_{Roboter} = \boldsymbol{T}_{Roboter}^{Kinect} \cdot \boldsymbol{p}_{Kinect} \tag{4.4}$$

Für die Visualisierung der AR müssen die 3D-Daten sämtlicher virtueller Objekte im Marker-KS vorliegen. Da es sich bei den virtuellen Objekten hauptsächlich um Programmparameter des Roboter-KS handelt, dient hierfür die Transformation zwischen Roboter-KS und Marker-KS $\boldsymbol{T}_{Marker}^{Roboter}$. Eine Verrechnung der einzelnen Koordinaten zur Visualisierung wird wie folgt vorgenommen:

$$\boldsymbol{p}_{Marker} = \boldsymbol{T}_{Marker}^{Roboter} \cdot \boldsymbol{p}_{Roboter} \tag{4.5}$$

Zur direkten Visualisierung der aktuellen Handposition wird die Transformation zwischen Marker und Motion-Tracking-KS $\boldsymbol{T}_{Marker}^{Kinect}$ benötigt, welche sich aus den zwei vorherigen Transformationen zusammensetzen lässt:

$$\boldsymbol{T}_{Marker}^{Kinect} = \boldsymbol{T}_{Marker}^{Roboter} \cdot \boldsymbol{T}_{Roboter}^{Kinect} \tag{4.6}$$

Zur Ermittlung der statischen Transformationsmatrizen können verschiedene Messverfahren und Algorithmen genutzt werden [173]. Aus Gründen der Wirtschaftlichkeit wird die Nutzung eines zusätzlichen, externen Messsystems ausgeschlossen. Für den allgemeinen Fall werden die Koordinaten korrespondierender Punkte in zwei verschiedenen Koordinatensystemen ermittelt. Gesucht wird die Transformationsmatrix $\boldsymbol{T}_1^2$, welche die Abhängigkeit der beiden Punktwolken $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ angibt:

$$\boldsymbol{p}_1 = \boldsymbol{T}_1^2 \cdot \boldsymbol{p}_2 \tag{4.7}$$

Im Folgenden wird ein beispielhaftes Vorgehen zur Ermittlung der Transformationsmatrix beschrieben. Zur Veranschaulichung dient Abbildung 4.30. Die Rotation zwischen beiden Koordinatensystemen lässt sich bestimmen, indem zunächst der arithmetische Mittelwert c_1 und c_2 der $N \in \mathbb{N}$ korrespondierenden Positionen in beiden Koordinatensystemen bestimmt wird:

$$\boldsymbol{c}_1 = \frac{1}{N} \sum_{i=1}^{N} p_{1,i} \qquad \text{und} \qquad \boldsymbol{c}_2 = \frac{1}{N} \sum_{i=2}^{N} p_{2,i} \tag{4.8}$$

Auf diese Weise lassen sich die Punktkoordinaten durch $\boldsymbol{p}_1\prime$ und $\boldsymbol{p}_2\prime$ in zwei Koordinatensystemen ausdrücken, die ihren Ursprung an gleicher Stelle besitzen. Somit ergibt sich zunächst eine

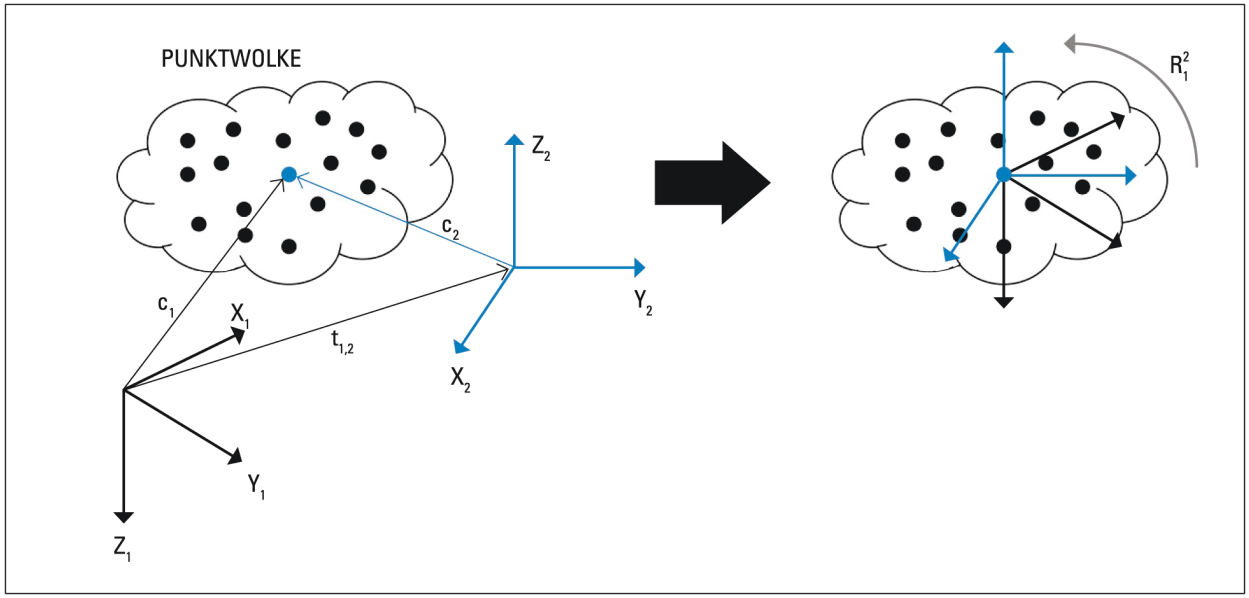

Abbildung 4.30: Ermittlung der Transformationsmatrix zwischen zwei Koordinatensystemen mithilfe von 3D-Punktwolken korrespondierender Positionen

Reduzierung der Transformation T_1^2 auf die Rotation R_1^2:

$$p_1' = R_1^2 \cdot p_2' \tag{4.9}$$

Zur Ermittlung ist für eine einzelne Positionskorrespondenz ein Gleichungssystem, bestehend aus drei Gleichungen und neun Unbekannten, zu lösen. Da die Rotationsmatrix orthogonal ist und ihre Spalten eine Orthonormalbasis bilden, lässt sich die Problemstellung auf drei Unbekannte, bspw. Eulerwinkel, reduzieren. Durch die Vielzahl der aufgenommenen Positionen lässt sich das Ergebnis der Rotationsmatrix durch Regression, bspw. durch die Methode der kleinsten Quadrate, weiterführend präzisieren. Der Translationsvektor $t_{1,2}$ lässt sich darauf anhand der Differenz der beiden Mittelwerte und einer Verrechnung der ermittelten Rotation bestimmen:

$$t_{1,2} = c_1 - R_1^2 \cdot c_2 \tag{4.10}$$

4.4 Zusammenfassung von Interaktionskonzept und Systementwurf

In den vorangegangenen Abschnitten wurde auf Basis der Anforderungsanalyse ein Interaktionskonzept[16] sowie ein Entwurf[17] des Programmiersystems erstellt. An dieser Stelle werden die wichtigsten Merkmale des Zielsystems kompakt zusammengefasst.
Im Rahmen der Interaktionsgestaltung[18] wurden zunächst die Verwendung der gestenbasierten Interaktion sowie die Ausgabe in der AR konkretisiert.

[16] Vgl. Kapitel 4.2.
[17] Vgl. Kapitel 4.3.
[18] Vgl. Kapitel 4.2.1.

Interaktionsgestaltung und Auslegung der Modalitäten

- Gestenbasierte Interaktion:
  - Bewegungsvorgabe nach dem Prinzip des PbD
  - markerlose Gestenerkennung
- Ausgabe in der AR:
  - Anzeigetechnologie: VST
  - gerätetechnische Umsetzung: Handheld-Geräte
  - Registrierung der mobilen Anwendung: Marker-Tracking

Weiterführend wurde ein methodischer Ansatz zur interaktiven Programmierung erstellt[19]. Dieser Ansatz dient der Übertragung der Interaktionsgestaltung auf den Prozess der Industrieroboterprogrammierung. Dazu wurden zunächst die Programmierschritte der Definition, Evaluation und Manipulation unterschieden und anschließend eine methodische Ausgestaltung durch die Modalitäten der Gesten und der AR vorgenommen.

Interaktive Programmierung

- Programmdefinition: gestenbasierte Definition von Posen, Trajektorien und Aufgaben
- Programmevaluation: mobile Visualisierung und Simulation von Programmen in der AR
- Programmmanipulation: interaktive Adaption von virtuellen Objekten, welche das Roboterprogramm repräsentieren, durch räumliche Gesten

Der Entwurf des Programmiersystems sieht eine Ausgestaltung des Interaktionskonzepts in Bezug auf Randbedingungen sowie auf die geplante prototypische Umsetzung des Programmiersystems vor[20]. Neben der Bestimmung von Programmiergerät und Wahrnehmungskonzept für die Gestenerkennung werden die Programmierumgebung sowie deren Schnittstellen und die Inbetriebnahme näher bestimmt.

Entwurf des Programmiersystems

- Programmiergeräte: Smartphones und Tablet-PCs
- Wahrnehmung der Gesten: integrativer Ansatz aus 3D-Motion-Tracking und 2D-Fingergestenerkennung, prototypische Umsetzung durch Kinect-Sensor und Kamera des Handheld-Geräts
- Hierarchische Gestaltung der Programmierebenen: bewegungsorientiert und aufgabenorientiert
- Drahtlose Schnittstellen zum Programmaustausch:
  - Robotersteuerung - Definition einer vereinheitlichten Industrieroboterschnittstelle
  - Digitale Fabrik - TCP/IP-Schnittstelle mit Post-/Präprozessor
- Inbetriebnahme: Ermittlung der homogenen Transformationsmatrizen zwischen den statischen Komponenten anhand von Positionskorrespondenzen

[19] Vgl. Kapitel 4.2.2.
[20] Vgl. Kapitel 4.3.

5 Umsetzung des natürlich-räumlichen Programmiersystems für Industrieroboter

In diesem Kapitel werden umsetzungsspezifische Aspekte des natürlich-räumlichen Programmiersystems dargelegt. In Abschnitt 5.1 wird zunächst die Umsetzung der Programmierumgebung und AR-Simulation in einer App für Mobilgeräte vorgestellt. In Abschnitt 5.2 wird die Umsetzung der gestenbasierten Eingabe durch den kombinatorischen Ansatz aus 3D-Motion-Tracking und 2D-Fingergestenerkennung beschrieben. Darauf aufbauend wird in Abschnitt 5.3 die Umsetzung der natürlich-räumlichen Interaktion erläutert. Abschließend werden in Abschnitt 5.4 die Realisierung der Schnittstellen zur Robotersteuerung und zur Digitalen Fabrik sowie die Inbetriebnahme des Programmiersystems dargestellt.

5.1 Umsetzung der mobilen Applikation

Die Umsetzung der räumlichen Programmierumgebung erfolgt in Form einer App für das Betriebssystem Android OS, welches seit 2011 über den größten Marktanteil bei Smartphones und Tablet-PCs verfügt [174]. Eine Übertragung des Programmiersystems auf andere Betriebssysteme von Mobilgeräten, wie bspw. Apple iOS, ist möglich, jedoch mit entsprechendem Aufwand für die Portierung des Quellcodes verbunden. Android basiert auf einem Linux-Kernel und dient als Softwareplattform für die Apps, welche sich in Java programmieren lassen und nach dem Kompilieren als Programm auf der virtuellen Maschine Dalvik ausgeführt werden. Als Programmierumgebung dient Eclipse IDE[1] in Kombination mit dem AndroidSDK[2]. Um eine effiziente Realisierung der Simulation und der Algorithmen zur Bildverarbeitung zu gewährleisten, werden CPU- und speicherintensive Programmierschritte in nativem C-Code implementiert. Dieser kann über das Android-NDK[3] in die Dalvik-App integriert werden. Untersuchungen des Autors der vorliegenden Arbeit haben ergeben, dass dieses Vorgehen bei der Visualisierung komplexer 3D-Modelle sowie bei Operationen mit großen Matrizen in der Bildverarbeitung erhebliche Performancevorteile bietet.

[1] Vgl. http://www.eclipse.org - 20.01.2014.
[2] Vgl. http://developer.android.com - 20.01.2014.
[3] Vgl. http://developer.android.com/tools/sdk/ndk - 20.01.2014.

5.1.1 Programmierung und Programmverwaltung

A) Bewegungsorientierte Programmierung

Die Grundlage für die Programmierung bildet eine bewegungsorientierte Programmierebene, welche über einen Satz von Interpolationsbefehlen und Anweisungen zur Ansteuerung des Endeffektors verfügt. Die bewegungsorientierte Programmierebene dient im Folgenden auch als Ausgangspunkt zur Übertragung der Roboterprogramme auf die reale Robotersteuerung und zur Digitalen Fabrik, da hier entsprechend abstrahierte Steuerungsbefehle und Schnittstellen vorliegen. Die Beschreibung der Programmierbefehle in der App erfolgt neutral und herstellerunabhängig. Aufgrund der Fehleranfälligkeit und mangelnder Effizienz wird die textuelle Eingabe bei der Programmierung auf ein Minimum reduziert. Anstatt textueller Programmierung lassen sich die einzelnen Programmierkommandos aus einer Dropdown-Liste über Touch-Gesten wählen. Posen können aus dem globalen Speicher als Parameter für Formeln oder Bewegungsanweisungen gewählt oder numerisch eingegeben werden. Die gesamte Bedienung der Dialoge der App erfolgt über Touch-Gesten, mit denen auch das intuitive Verschieben einzelner Programmzeilen per Drag-and-Drop-Metapher ermöglicht wird.

B) Aufgabenorientierte Programmierung

Die interne Darstellung der Roboterprogramme erfolgt über eine hierarchische Struktur verschiedener Objektinstanzen. Um gleichzeitig einer bewegungs- sowie einer aufgabenorientierten Verwendung gerecht zu werden, werden zunächst die abstrakten Instanzen Skill und Task eingeführt[4].

Skill- und Task-Instanz
Der Task bildet eine gesamte Aufgabe ab, welche aus verschiedenen Teilaufgaben, den Skills, zusammengesetzt werden kann. Skills können wiederum gesamte Aufgaben, Bewegungen oder andere Anweisungen, bspw. zur Ansteuerung des Endeffektors, beinhalten. Der Aufruf eines Tasks durch sich selbst wird bei der Ausführung verhindert. Um eine allgemeine Definition von Skills und Tasks ohne die konkrete Zuweisung von spezifischen Posen durchführen zu können, werden diese Instanzen parametrierbar ausgeführt. Derart lässt sich durch den Expertennutzer eine allgemeine Aufgabe in Form eines Programmmakros erstellen. Den Parametern lassen sich darauf bspw. Handposen in Verbindung mit bestimmten Fingergesten zuordnen, sodass bei gestenbasierter Interaktion eine automatische Parametrierung der Ausgaben erfolgt. Abbildung 5.1 verdeutlicht den strukturellen Aufbau der Objektinstanz „Task“.

Parameterinstanz
Die Parameter bilden das Argument, welches den vordefinierten Aufgabenstrukturen übergeben wird und deren Ausführung individuell beeinflusst. Als Parameter werden zwei verschiedene Objekttypen betrachtet: Konstanten und Posen (s. Abb. 5.2). Mithilfe von Konstanten kann bspw. ein vorgegebener Abstand von Zustellbewegung zum Werkstück zugewiesen werden. Posenparameter übergeben die absolute Position und die Orientierung des Endeffektors in Form kartesischer Koordinaten oder Achswinkeln. Posen lassen sich aus der Datenbank zuweisen, von Interaktions-

[4] Vgl. Kapitel 4.3.3.

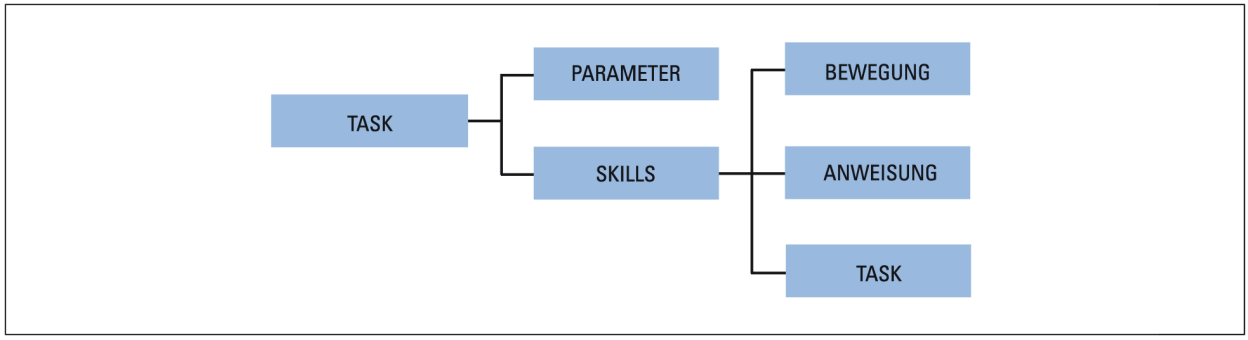

Abbildung 5.1: Aufbau der Objektabhängigkeiten bezüglich der Task-Instanz.

objekten ableiten oder direkt mit der Handpose des Anwenders in Verbindung mit bestimmten Fingergesten verknüpfen.

Bewegungsinstanz

Die Bewegungsinstanz beschreibt Bewegungsanweisungen, welche die Interpolationsart, Geschwindigkeits- und Beschleunigungsangaben sowie die Anzahl der zugewiesenen Posen beinhalten (s. Abb. 5.3). Die Anzahl der Posen ist abhängig von der Interpolationsart. Während bei Point-to-Point- (PTP) und Linearbewegungen jeweils nur eine Pose zugewiesen wird, werden bei Zirkularbewegungen zwei Posen verlangt, und bei der Spline-Interpolation sind neben Start- und Endpose $n \in \mathbb{N}$ Stützstellen der Kurve anzugeben.

Instanz der parametrierbaren Posen

Die Instanz der parametrierbaren Posen (s. Abb. 5.4) repräsentiert Posen des Roboters, bestehend aus mindestens sechs Einträgen, welche Position und Orientierung des Endeffektors im kartesischen Raum oder die Achswinkel des Roboters beinhalten. Bezüglich der Orientierung können verschiedene Darstellungen wie Quaternionen oder die Winkelkonventionen ZY'Z"-Euler und Roll-Pitch-Yaw (RPY) gewählt werden. Die einzelnen Einträge der Posen lassen sich über Konstanten, variable Parameter oder als Ergebnis einer mathematischen Operation in Form einer Formel zuweisen.

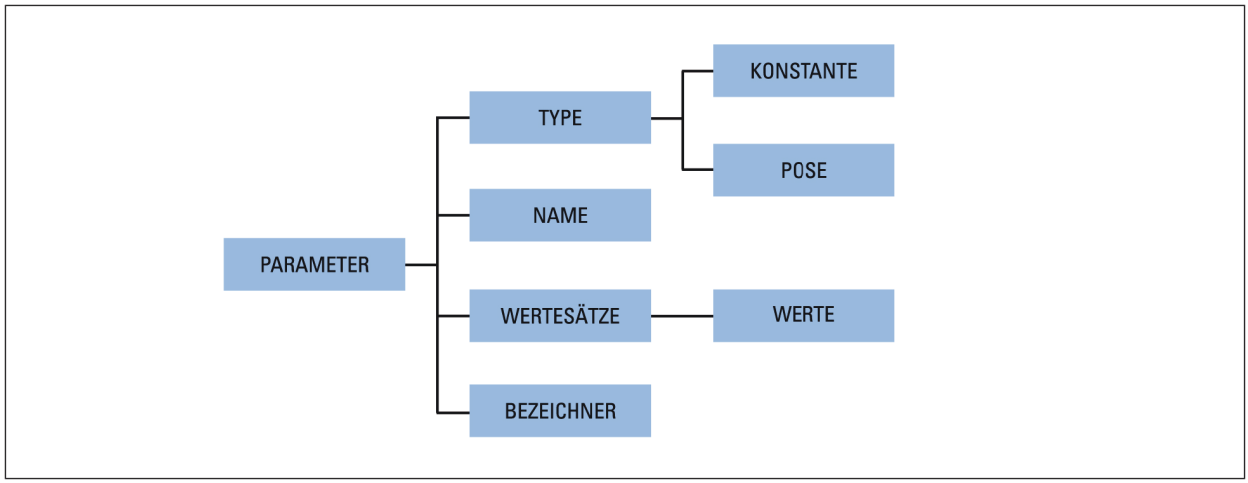

Abbildung 5.2: Struktur der Parameterinstanz

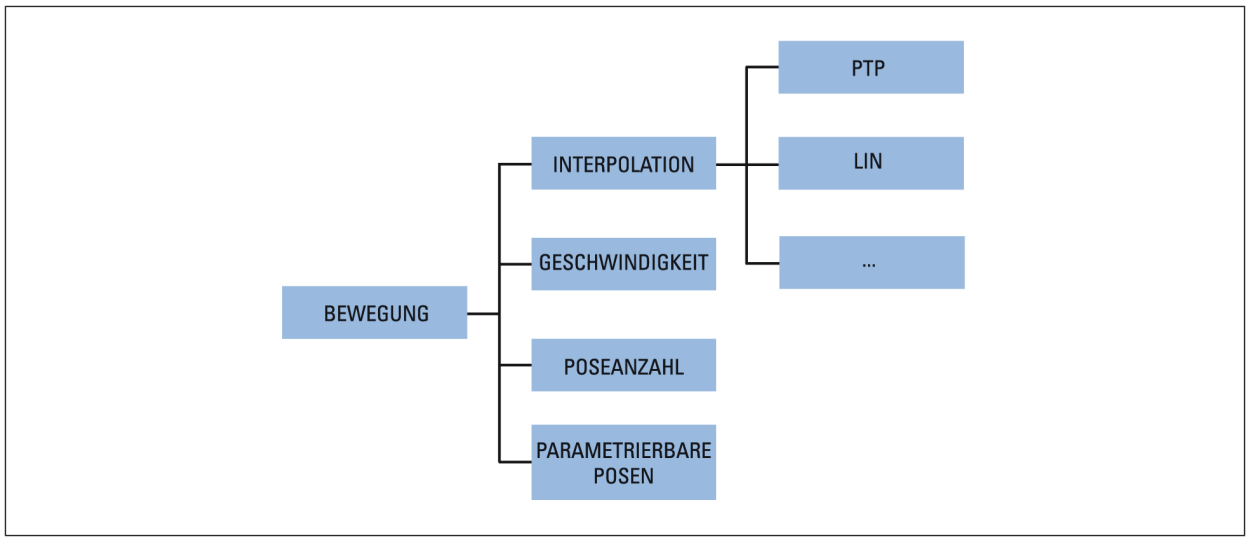

Abbildung 5.3: Struktur der Bewegungsinstanz.

Formelinstanz

Eine Formelinstanz entspricht einer mathematischen Gleichung, welche Rechenzeichen, Posenvariablen, Konstanten und Referenzen auf einzelne Parameter enthalten kann. Somit lässt sich ein funktionaler Zusammenhang zwischen einem Ergebnis und den Einträgen einzelner Parameter herstellen. Rechenzeichen und Zahlenwerte können über eine entsprechende Dialogführung mit textueller Eingabefunktion zugewiesen werden.

Umsetzung der Objektinstanzen in einem Datenbankmodell

Zur Realisierung und Verwaltung der Datenbanken, welche die einzelnen Instanzen und Verbindungen der Programmierobjekte enthalten, dient die Programmbibliothek SQLite[5]. Diese wird von Android ohne das Einbinden zusätzlicher Bibliotheken unterstützt. Die Datenbanken werden nicht zentral auf einem Server gespeichert, sondern liegen lokal als *.db*-Datei vor. Somit wird ein Mehr-

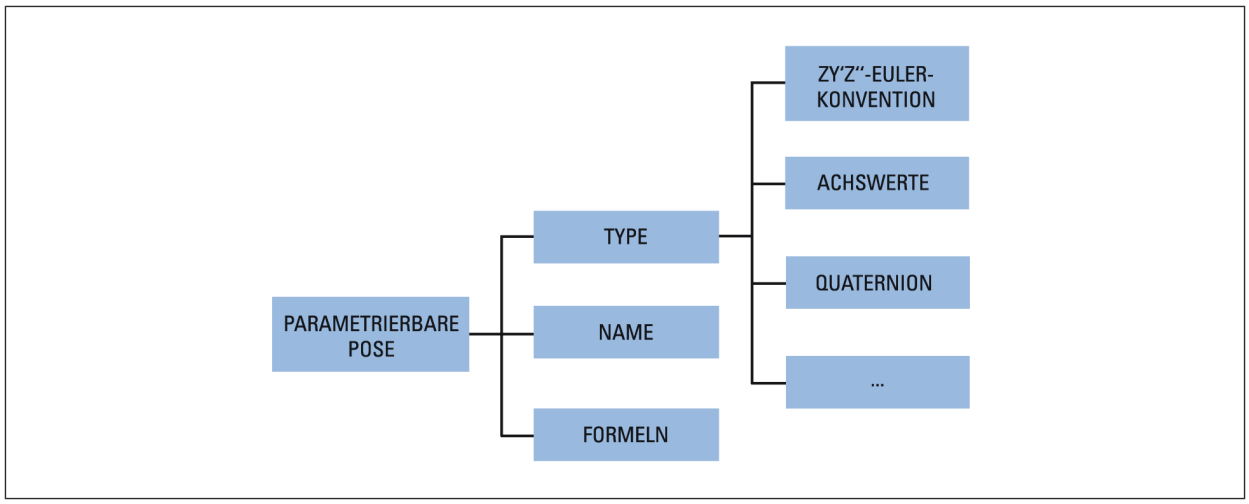

Abbildung 5.4: Struktur der Instanz der parametrierbaren Posen.

[5] Vgl. http://www.sqlite.org - 20.01.2014.

fachzugriff der Datenbanken durch verschiedene Teilnehmer unterbunden. Stattdessen bleibt die Übertragbarkeit der Datenbanken erhalten.
Zum Speichern und Abrufen der Programminstanzen dient in der App ein Datenbankmodul, welches Lese- und Schreibzugriff auf die Tabellen und Datensätze der Datenbanken bereitstellt. Für jede der aufgeführten Objektinstanzen wird eine Tabelle vorgesehen. Entsprechend der Abhängigkeit der Instanzen untereinander werden auch diese durch das Datenbankmodell abgebildet. In Anhang D wird die Struktur des gesamten Datenbankmodells, speziell die Realisierung der Abhängigkeiten der einzelnen Instanzen, in der IDEF1X-Notation veranschaulicht.
Den Parametern der einzelnen Instanzen werden in den jeweiligen Tabellen Spalten und Datentypen der abhängigen Objektinstanz zugeordnet. Für die eindeutige Identifikation einzelner Einträge werden Primär- und Sekundärschlüssel deklariert. Ist die Reihenfolge von Einträgen einzelner Instanzen von Bedeutung, bspw. bei Tasks und Skills, wird diese über eine zusätzliche Spalte mit numerischem Eintrag definiert. Diese Reihenfolge dient schließlich der Ausführung und Übertragung der Programme. Derart bildet bspw. eine Skill-Tabelle den Ablaufplan einer Teilaufgabe ab. Referenzierte Fremdschlüssel verweisen auf den Primärschlüssel definierter Aufgaben, Bewegungen oder Anweisungen.

Generieren von aufgabenorientierten Roboterprogrammen
Soll auf Basis einer Aufgabendefinition ein ausführbares Programm für die AR-Simulation oder den realen Roboter erzeugt werden, ist diese Definition auf eine bewegungsorientierte Programmierebene zurückzuführen. Zur Generierung des Programms werden die Einträge der Aufgabe sequentiell durchlaufen, Referenzen aufgelöst und das Ergebnis von Formeln berechnet. Dieses Vorgehen lässt sich mithilfe eines Ablaufdiagramms (s. Abb. 5.5) veranschaulichen.
Eine Herausforderung stellt die Berechnung der Formeln dar. Durch das Auflösen der Referenzen in den Formelvariablen liegen diese in Form von Zahlen und Formelzeichen textuell vor. Zur Berechnung der Formel wird die quelloffene Java-Bibliothek Beanshell[6] eingebunden. Diese stellt eine passende Skriptsprache mit Interpreter bereit.

C) Benutzeroberflächen

Benutzeroberfläche für die bewegungsorientierte Programmierung
Die Programmverwaltung erfolgt über einen Dialog zur Programmdarstellung, der aus dem Hauptmenü der App aktiviert werden kann und sich über Touch-Gesten steuern lässt (s. Abb. 5.6). Die Dialoge der App werden über das Kamerabild der AR-Anwendung gelegt, sodass das Bild nur teilweise überdeckt wird. Sie lassen sich entsprechend unterschiedlichen Nutzeranforderungen flexibel skalieren und auf dem Display frei verschieben. Auf Basis der eingebundenen Datenbanken können Programme neu angelegt, geladen und gespeichert werden. Zur textuellen Eingabe von Bezeichnungen dient die Touch-Tastatur. Beim Anlegen von Bewegungsbefehlen können zugehörige Posen aus der zentralen Datenbank gewählt oder numerisch angelegt werden.

[6] Vgl. http://www.beanshell.org - 20.01.2014.

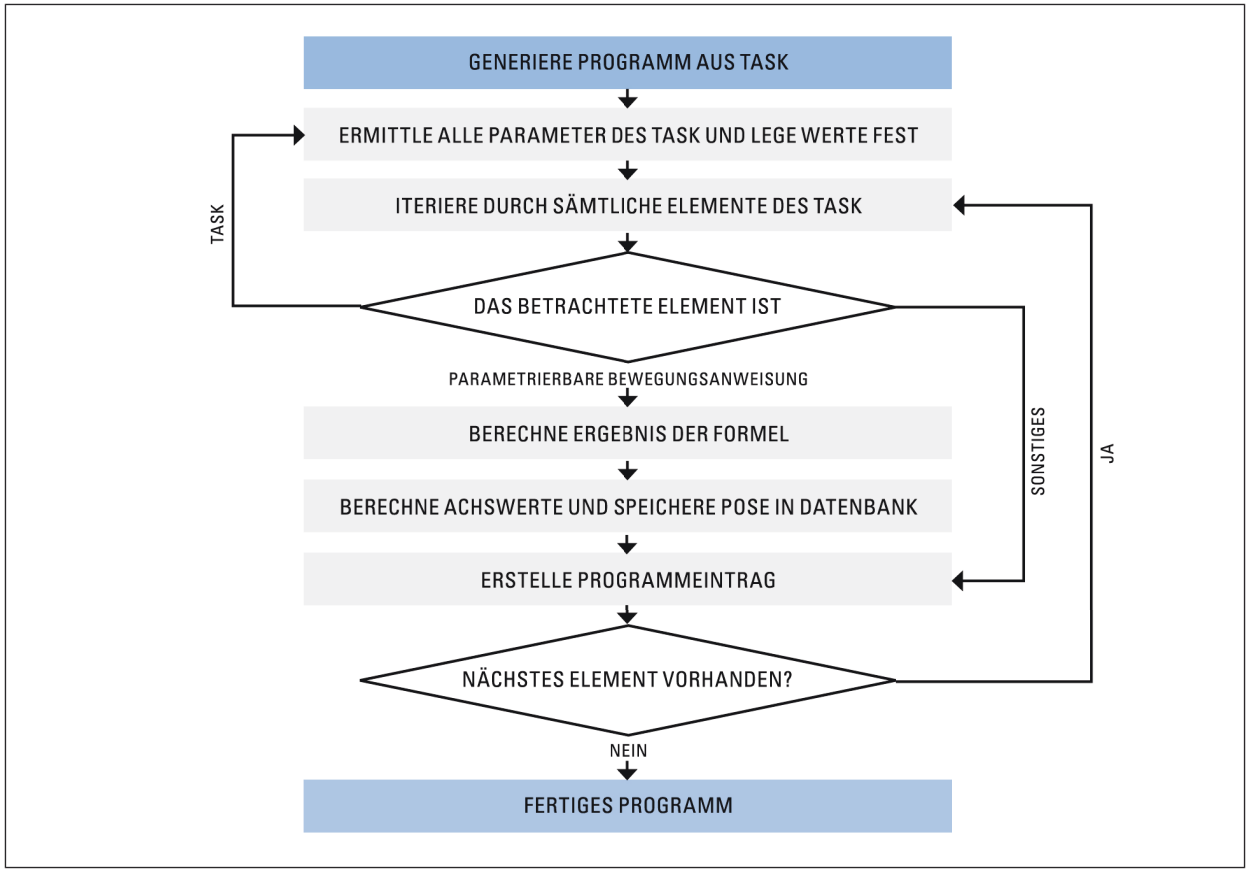

Abbildung 5.5: Ablaufdiagramm zum Generieren eines bewegungsorientierten Roboterprogramms aus einer Aufgabenbeschreibung.

Benutzeroberfläche für die aufgabenorientierte Programmierung
Zur Definition von Aufgaben durch den Expertennutzer werden verschiedene Dialoge zur Verwaltung und Verknüpfung der Datenstrukturen zur Verfügung gestellt (s. Abb. 5.7). Diese ergänzen den klassischen Dialog zur bewegungsorientierten Programmierung.

5.1.2 Visualisierung und Simulation in der Augmented Reality

A) Mobile Augmented-Reality-Anwendung

Zur Umsetzung der AR-Umgebung auf dem Mobilgerät wurde zunächst eine Android-Portierung des Open-Source-Projekts ARToolkit[7] gewählt. Der Funktionsumfang dieses Software-Framework beinhaltet eine Kameraanwendung, ein Marker-Tracking für quadratische Flachmarker mit Bildmuster, einen Parser für CAD-Dateien im Wavefront-Format (*.obj*) und einen Renderer auf Basis von OpenGL ES1 zur Darstellung virtueller Objekte in einer Bildebene (GL Surface) über dem Kamerabild (s. Abb. 5.8). Individuelle Markermuster lassen sich erstellen und in die Anwendung implementieren. Prinzipiell ist durch das ARToolkit auch die Möglichkeit eines Multi-Marker-Tracking gegeben. Praktische Untersuchungen zeigen jedoch, dass der Algorithmus zur

[7] Vgl. http://www.hitl.washington.edu/artoolkit/ - 20.01.2014.

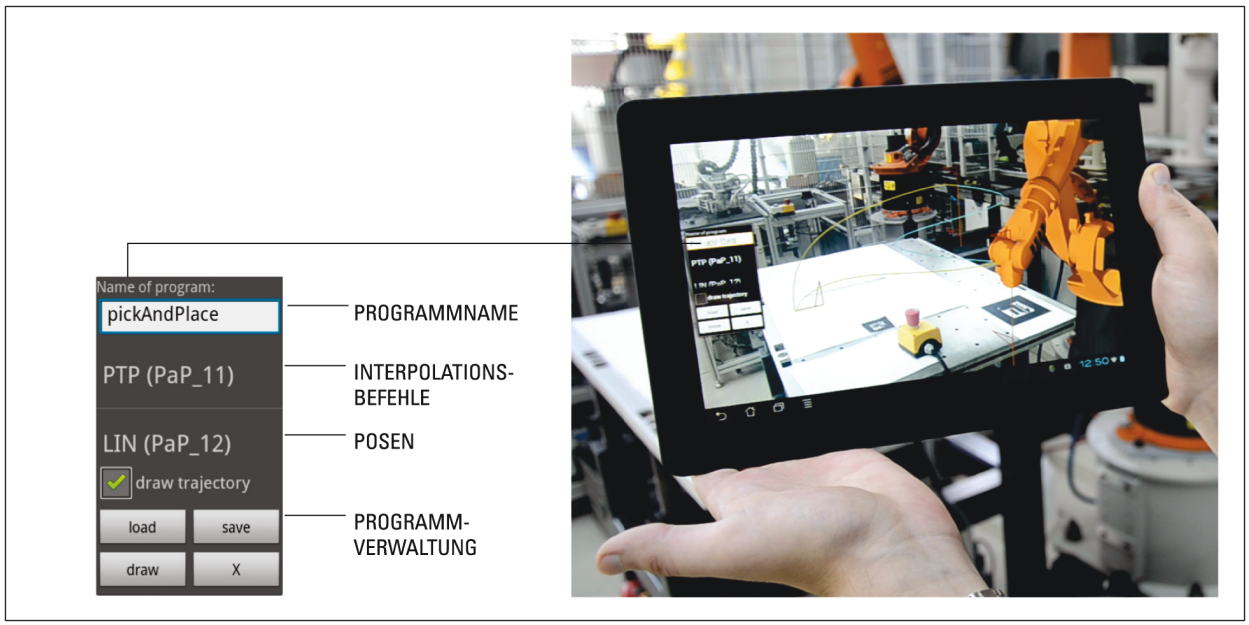

Abbildung 5.6: Programmdialog: frei positionierbar und skalierbar.

Markererkennung bei mehreren Markern zu einer signifikanten Erhöhung der Rechenzeit führt. Aus diesem Grund wurde der vorhandene Algorithmus durch eine Android-Portierung und Anpassung des Marker-Tracking des ALVAR-Framework[8] ersetzt. Der gewählte Algorithmus [175] betrachtet Binärmarker[9], welche zusätzlich die Vorteile einer erhöhten Robustheit und geringerer Erkennungsfehler aufweisen. Auch der vorhandene Renderer wurde durch eine aktuelle Version von OpenGL ES ersetzt und individuell angepasst.

Realisierung des Multi-Marker-Tracking

Auf Basis des Marker-Tracking wird die App um einen Algorithmus erweitert, welcher die Pose des Handheld-Geräts in Abhängigkeit zur Pose eines Referenzmarkers ermittelt. Das KS des

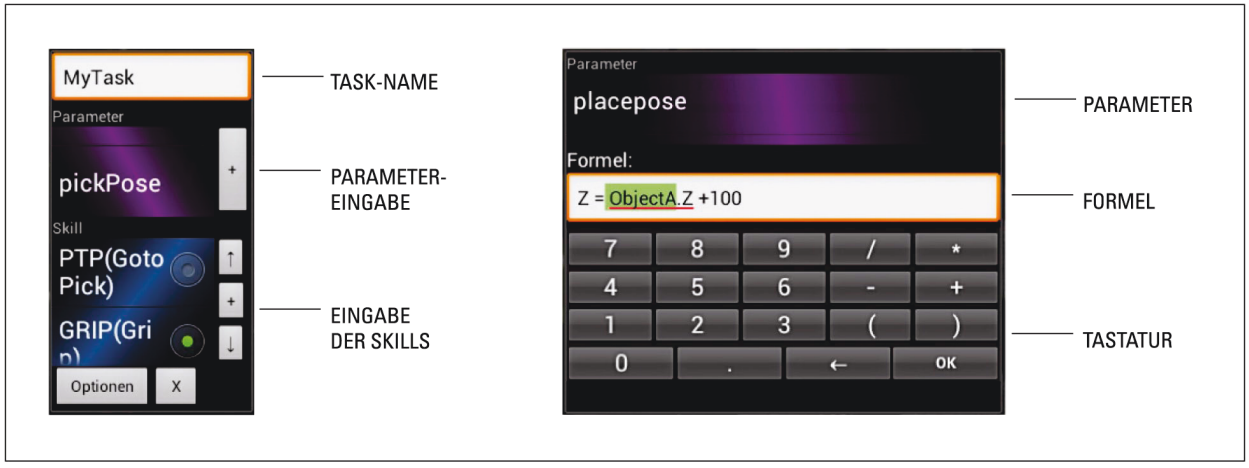

Abbildung 5.7: Benutzeroberflächen zur Definition von Tasks (links) und Formeln (rechts).

[8] Vgl. http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/ - 20.01.2014.
[9] Vgl. Abbildung 4.13 in Kapitel 4.2.1.

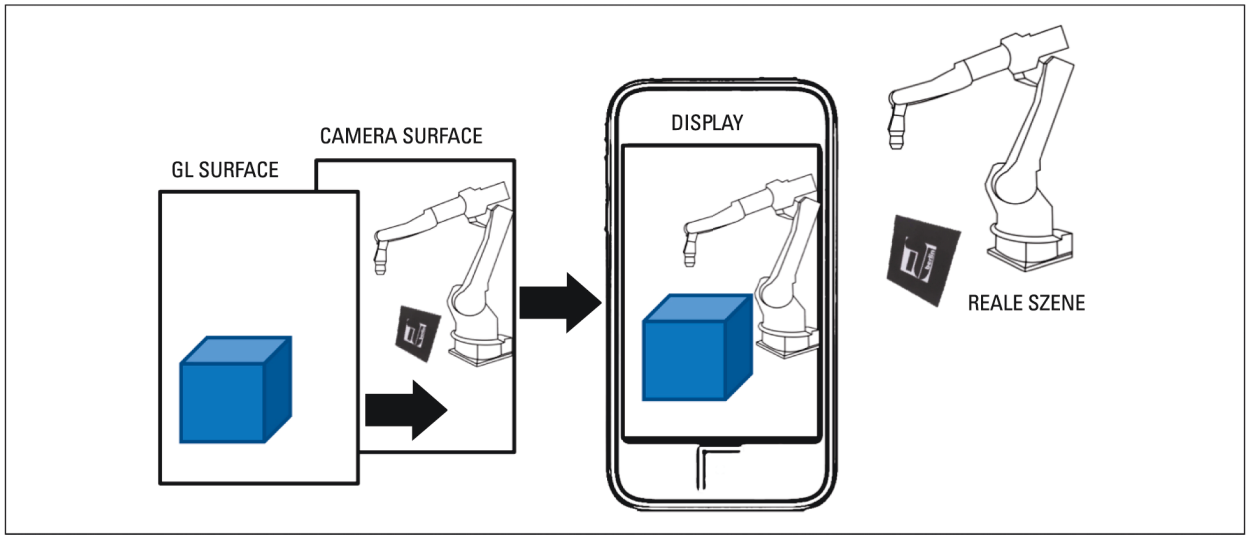

Abbildung 5.8: Rendern der virtuellen Objekte (hier: blauer Kubus) auf eine transparente Ebene (GL Surface) über dem Kamerabild (Camera Surface).

Referenzmarkers dient als Basis der Darstellung der virtuellen Objekte in der AR-Umgebung. Die aktuelle Pose des Mobilgeräts, bezogen auf den Referenzmarker, wird als Pose einer virtuellen dynamischen Kamera im Renderer der AR-Umgebung genutzt. Sind mehrere Marker sichtbar (s. Abb. 5.9), gibt der Algorithmus die Transformationen zwischen den einzelnen Markern, welche über eine definierte ID verfügen, und der Kamera des Mobilgeräts zurück. Ist die reale Transformation zwischen Referenzmarker und einem weiteren Marker $\boldsymbol{T}_{Referenz}^{Marker_X}$ bekannt, kann aus den einzelnen Transformationen des erkannten Markers $\boldsymbol{T}_{Marker_X}^{Kamera}$ auf die Transformation der Kamera zum Referenzmarker $\boldsymbol{T}_{Referenz}^{Kamera}$ rückgeschlossen werden (s. Formel 5.1).

$$\boldsymbol{T}_{Referenz}^{Kamera} = \boldsymbol{T}_{Referenz}^{Marker_X} \cdot \boldsymbol{T}_{Marker_X}^{Kamera} \tag{5.1}$$

Prinzipiell genügt das Tracking eines einzelnen Markers für die perspektivische Darstellung virtueller Objekte im Kamerabild. Bei mehreren sichtbaren Markern liegen redundante Informationen vor, die durch Bildung eines arithmetischen Mittelwerts zur Unterdrückung von Störeinflüssen wie Bildrauschen eingesetzt werden. Da die Genauigkeiten des Algorithmus von der Entfernung des Markers zur Kamera abhängen[10], wird bei mehreren sichtbaren Markern zusätzlich ein Gewichtungsfaktor in Abhängigkeit von der Entfernung eingeführt.
Zur Umsetzung des Marker-Tracking können in der App IDs für den Referenzmarker und für Zusatzmarker registriert werden. Für die Zusatzmarker ist die Translation und Rotation in der Ebene in Bezug auf den Referenzmarker anzugeben. Die Eingabe kann über einen Programmdialog oder über das Laden eines Tabellendokuments erfolgen. Letzteres bietet sich bei einer größeren Anzahl von Markern an.

[10] Vgl. Kapitel 4.2.1.

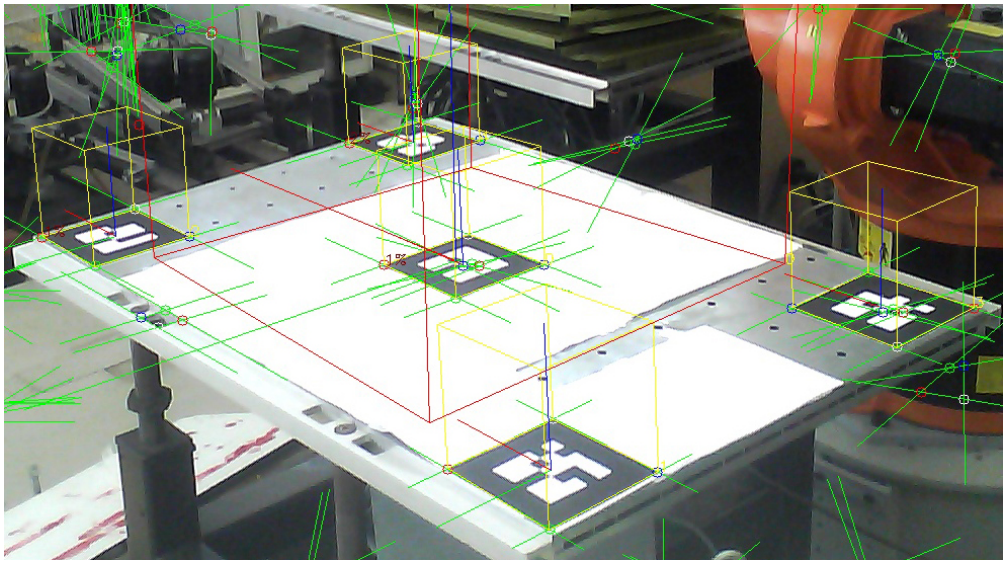

Abbildung 5.9: Visualisierung des Multi-Marker-Tracking und der Marker-KS: erkannte Marker (gelber Kasten) und Referenzmarker in der Mitte (gelber und roter Kasten) sowie erkannte Linien (grün) des Algorithmus.

Behandlung von Verdeckungen realer Objekte

Ein bekanntes Problem in AR-Anwendungen ist die fehlerhafte Darstellung von potentiellen Verdeckungen virtueller Objekte durch reale Objekte des Kamerabilds. Dies führt zu einer unrealistischen Darstellung, welche prinzipiell den Grad der Immersion senkt und zur Irritation des Anwenders führt. Der Grund für diesen Effekt liegt darin, dass die virtuellen Objekte stets in einer Ebene über dem Kamerabild dargestellt werden (s. Abb. 5.10, links) und in der Regel keine Informationen über reale Objekte und deren Position im Raum vorliegen. Eine Lösungsmöglichkeit für dieses Problems besteht bei bekannter Pose und Geometrie der realen Objekte darin, diese als virtuelle Objekte ohne Farb-, jedoch mit Tiefeninformationen zu rendern [176].
In der App wird dieser Ansatz grundlegend umgesetzt, indem eine Definition von Verdeckungsobjekten anhand von geometrischen Primitiven, bspw. Würfeln und Kugeln, bereitgestellt wird. Diese können wiederum über Dialogeingabe oder durch die Definition und das Laden eines Tabellendokuments angelegt werden. In Abbildung 5.10 (rechts) ist die entsprechende Modellierung eines Tisches als Würfelgeometrie zur partiellen Verdeckung des virtuellen Roboters durch einen realen Tisch in der AR dargestellt.

B) Effiziente Realisierung der 2D-Bildverarbeitung auf dem Handheld-Gerät

Die effiziente Umsetzung des räumlichen Programmiersystems auf dem Mobilgerät stellt nicht nur hohe Anforderungen an die Effizienz einzelner Algorithmen, sondern auch an die generelle Struktur der Informationsverarbeitung. Kritisch aus Sicht der Usability ist die videobasierte Ausgabe, bei der eine möglichst hohe Bildfrequenz von 15–30 fps angestrebt wird[11]. Als rechenintensive Prozesse wurden die Bildverarbeitungsalgorithmen zur Fingergestenerkennung und zum Marker-Tracking identifiziert. Beide Algorithmen nutzen den aktuellen Frame des Videostreams der Ka-

[11] Vgl. Kapitel 4.3.1.

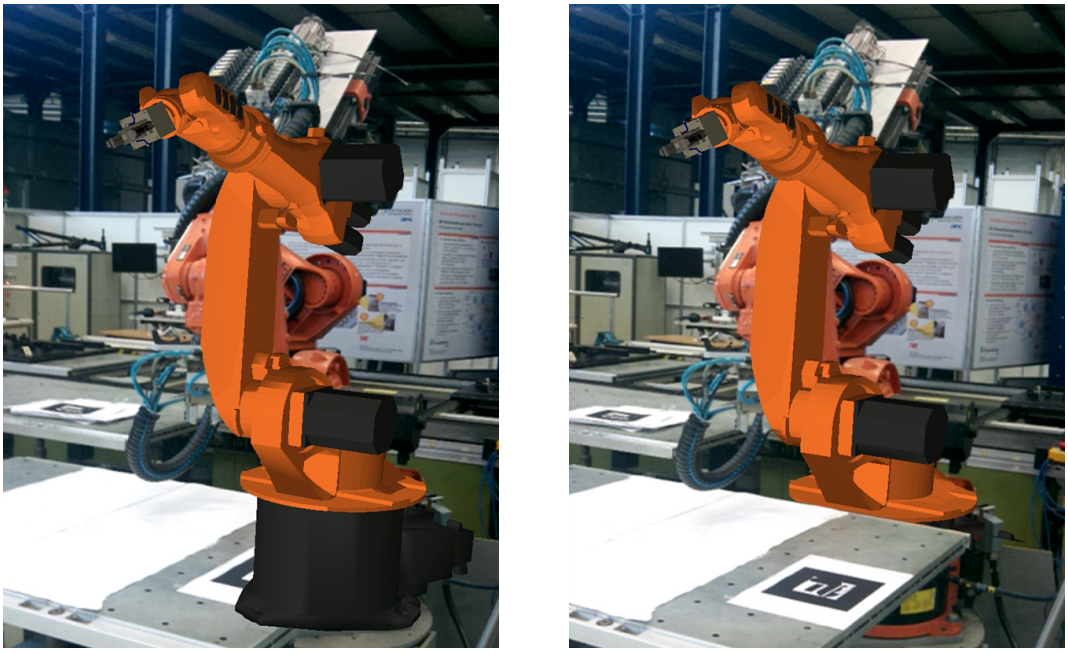

Abbildung 5.10: AR-Darstellung des Roboters ohne (links) und mit (rechts) Modellierung eines Verdeckungsobjekts.

mera als Eingangsgröße. Die in jedem Frame durchlaufene Prozesskette besteht aus Aufnahme des Frames, Verarbeitung und Ausgabe. Diese grundlegenden Schritte müssen gemäß der Anforderung echtzeitfähig ablaufen. Praktische Untersuchungen haben gezeigt, dass mit einer sequentiellen Abarbeitung von Marker-Tracking und Fingergestenerkennung die angestrebte Bildfrequenz auf modernen Handheld-Geräten nicht erfüllt werden konnte.
Durch gezieltes Nutzen der Mehrkernarchitektur aktueller Mobilgeräte in Form einer Parallelisierung von Gestenerkennung und Marker-Tracking werden auf den betrachteten Testgeräten 20–30 fps erreicht. Zur Umsetzung der Parallelisierung wird nach Bereitstellung eines neuen Frames zunächst eine Kopie erstellt. Originalframe und Kopie werden auf zwei verschiedene Threads für die Gestenerkennung und die AR verteilt. Die beiden kritischen Prozesse werden somit parallel ausgeführt, während nach erkannter Geste über den sogenannten „Interaktionsmanager“ Einfluss auf die Darstellung in der AR genommen wird. Erst wenn beide Threads durchlaufen sind, wird die Ausgabe durch Aufbau des Framelayouts aus Kamerabild (Camera Surface) und virtueller Ebene (GL Surface) dargestellt. Nach erfolgreicher Darstellung wird der nächste Frame von der Kamera angefordert. Abbildung 5.11 stellt diesen Programmablauf grafisch dar.

C) Umsetzung der Simulationsumgebung

Die Umsetzung der Simulation erfolgt für klassische 6-Achs-Knickarmroboter. Eine Erweiterung auf andere Kinematiken ist gegeben. An der Darstellung des Robotermodells und der Durchführung der Simulation sind verschiedene Programmmodule beteiligt. Abbildung 5.12 veranschaulicht die Hauptkomponenten und deren Funktionsmodule sowie die Schnittstellen der Programmierumgebung.

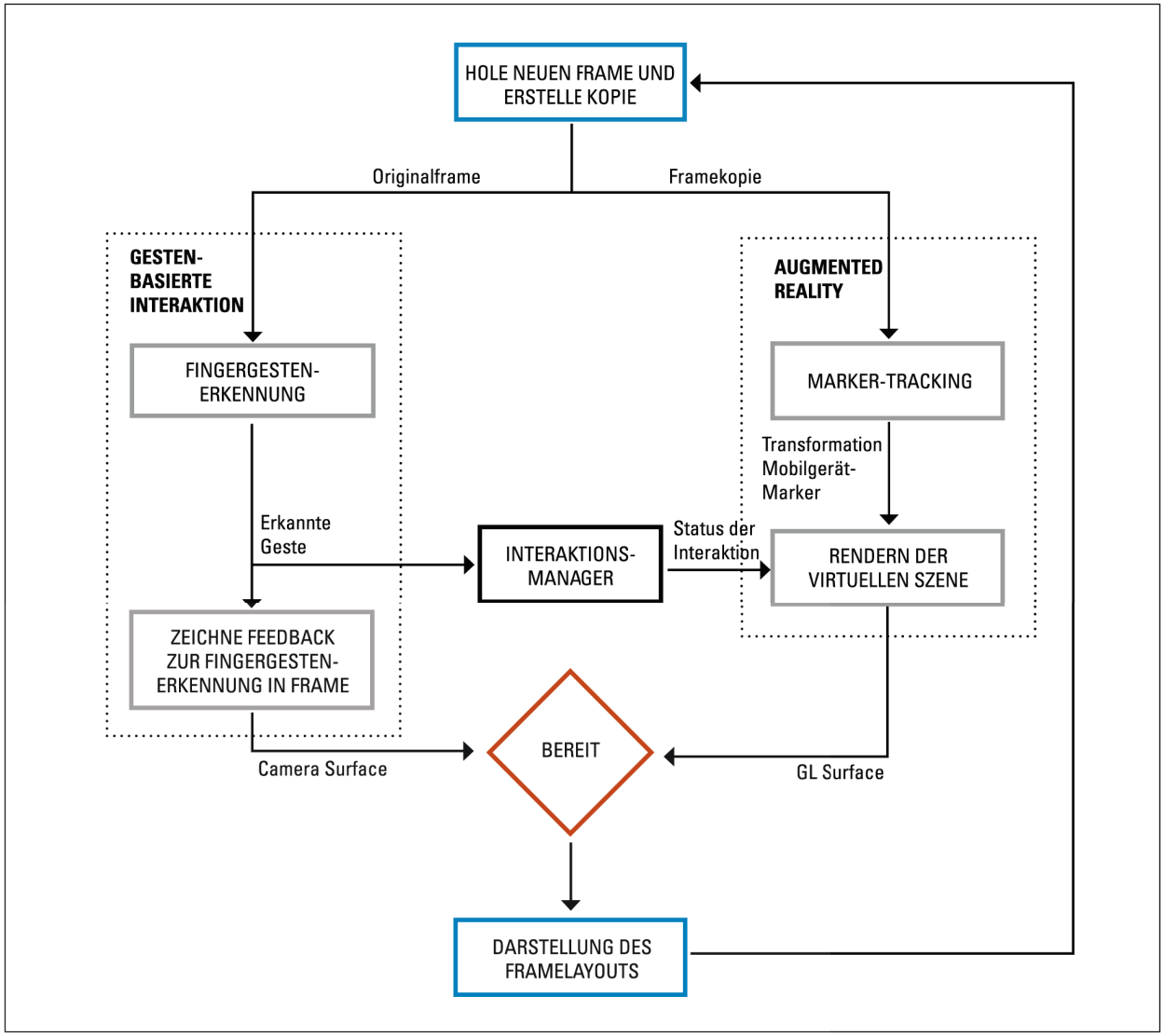

Abbildung 5.11: Parallelisierung des Programmablaufs für die Verarbeitung der Frames: Verteilung des Kamerabilds auf die Algorithmen der Fingergestenerkennung und des Marker-Tracking.

Modellierung der Industrieroboter

Zur Darstellung der seriellen Roboterkinematik wird ein hierarchischer Ansatz der 3D-Modellierung anhand eines Szenengraphs gewählt. Die einzelnen Achsen des Roboters werden als Knotenobjekte definiert. Die Posen der Einzelkomponenten des Roboters lassen sich demnach bei der Drehung von Einzelachsen automatisch anpassen. Zur Umsetzung des Szenengraphs wird das frei verfügbare Toolkit OpenSceneGraph[12] genutzt.

Den einzelnen Knotenobjekten werden 3D-Objekte, welche aus CAD-Dateien importiert werden, zugewiesen. Zum Import der CAD-Dateien wird die Klasse OBJ-Reader des ARToolkit genutzt. Die CAD-Daten bestehen aus einer *.obj*-Datei, welche die geometrischen Daten des Objekts enthält, und einer *.mtl*-Datei, welche optische Daten wie Farbe, Material und Transparenz beinhaltet. Das Format wird von gängigen CAD-Anwendungen unterstützt. Eine Erweiterung der App für Parser anderer CAD-Datenformate ist gegeben.

[12] Vgl. http://www.openscenegraph.org/ - 20.01.2014.

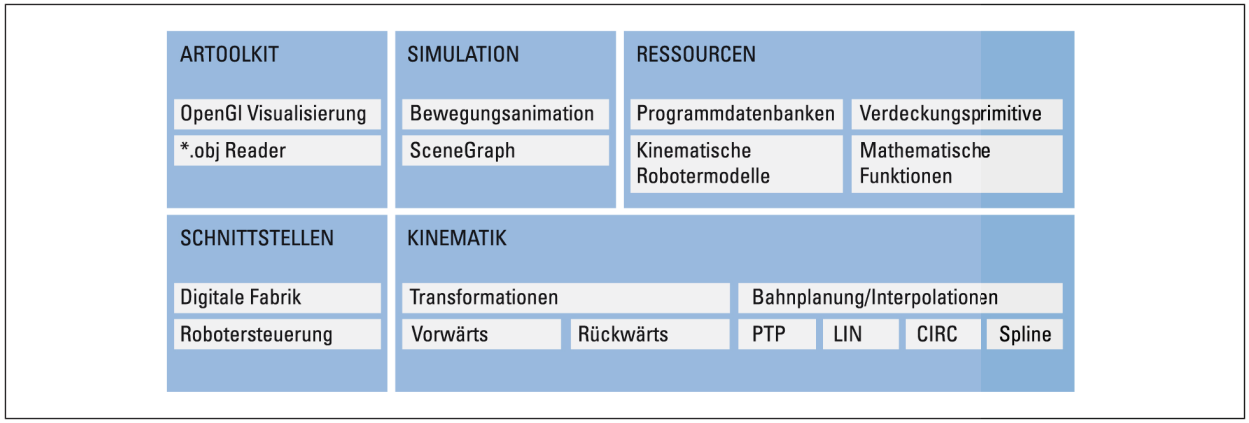

Abbildung 5.12: Aufbau des Simulationssystems: Hauptkomponenten (blau) und Funktionsmodule (grau)

Die Definition der kinematischen Kette erfolgt über eine Beschreibung der Roboterkinematik durch die DH-Konvention. Weitere Angaben umfassen Achsbeschränkungen sowie Maximalwerte für Geschwindigkeit und Beschleunigung der einzelnen Achsen. Eine homogene Transformationsmatrix beschreibt die Verbindung zum nächsten Knoten. Die jeweilige Transformation kann aus den DH-Parametern des Knotens und einer vorgegebenen Winkelstellung der Achse abgeleitet werden. Abbildung 5.13 veranschaulicht das Vorgehen zur Modellierung des virtuellen Roboters aus verschiedenen Knotenobjekten. Durch den modularen Aufbau des Szenengraphs lassen sich Änderungen der Kinematik bzw. ein Wechsel des Robotermodells einfach vollziehen. Hierzu sind lediglich DH-Parameter, Achsbeschränkungen sowie Verweise auf CAD-Daten zu ändern. Zusätzliche Objekte wie Endeffektoren lassen sich ohne großen Aufwand in die vorhandene Hierarchie integrieren und aus dieser entfernen. Weiterführend lassen sich über den Szenengraph weitere Objektabhängigkeiten für Koordinatensysteme, bspw. die Verschiebung des Roboters im Welt-KS, sowie für die aufgabenorientierte Programmierung definieren. Zudem können die Funktionsmodule zur Vorwärts- und Rückwärtskinematik auf der Struktur des Szenengraphs aufsetzen und durch diese iterieren.

D) Manipulation der Roboterbasis

Grundlegende Funktionen der Visualisierung des Robotermodells umfassen das Verschieben, Verdrehen und Skalieren der Roboterbasis in der dargestellten Szene. Die freie Positionierung und Orientierung der virtuellen Szene im Raum ist die Grundlage für die Überlagerung des realen Roboters durch ein virtuelles Abbild. Die Skalierung ermöglicht eine realitätsgetreue Darstellung des Roboters für CAD-Daten, deren Maße nicht dem realen Roboter entsprechen. Zusätzlich wird durch die Skalierung eine Übertragung der AR-Simulation auf Büroanwendungen ermöglicht. Hier kann aus Gründen der Übersichtlichkeit eine geringe Skalierung gewählt werden. Abbildung 5.14 veranschaulicht die Darstellung eines Industrieroboters mit variierter Positionierung, Orientierung und Skalierung. Realisiert wird die Manipulation des Robotermodells durch eine Anpassung der Transformation des Basisknotens im Szenengraph. Dieser beschreibt die Verschiebung der Roboterbasis zum KS der Darstellung, welches sich auf dem AR-Marker befindet. Durch den hierarchischen Aufbau werden sämtliche Knoten durch eine Verkettung der Transformationsmatrizen angepasst. Formel 5.2 beschreibt die Bildung der Transformationsmatrix $\boldsymbol{T}_{Marker}^{Roboter}$ bestehend aus der Transla-

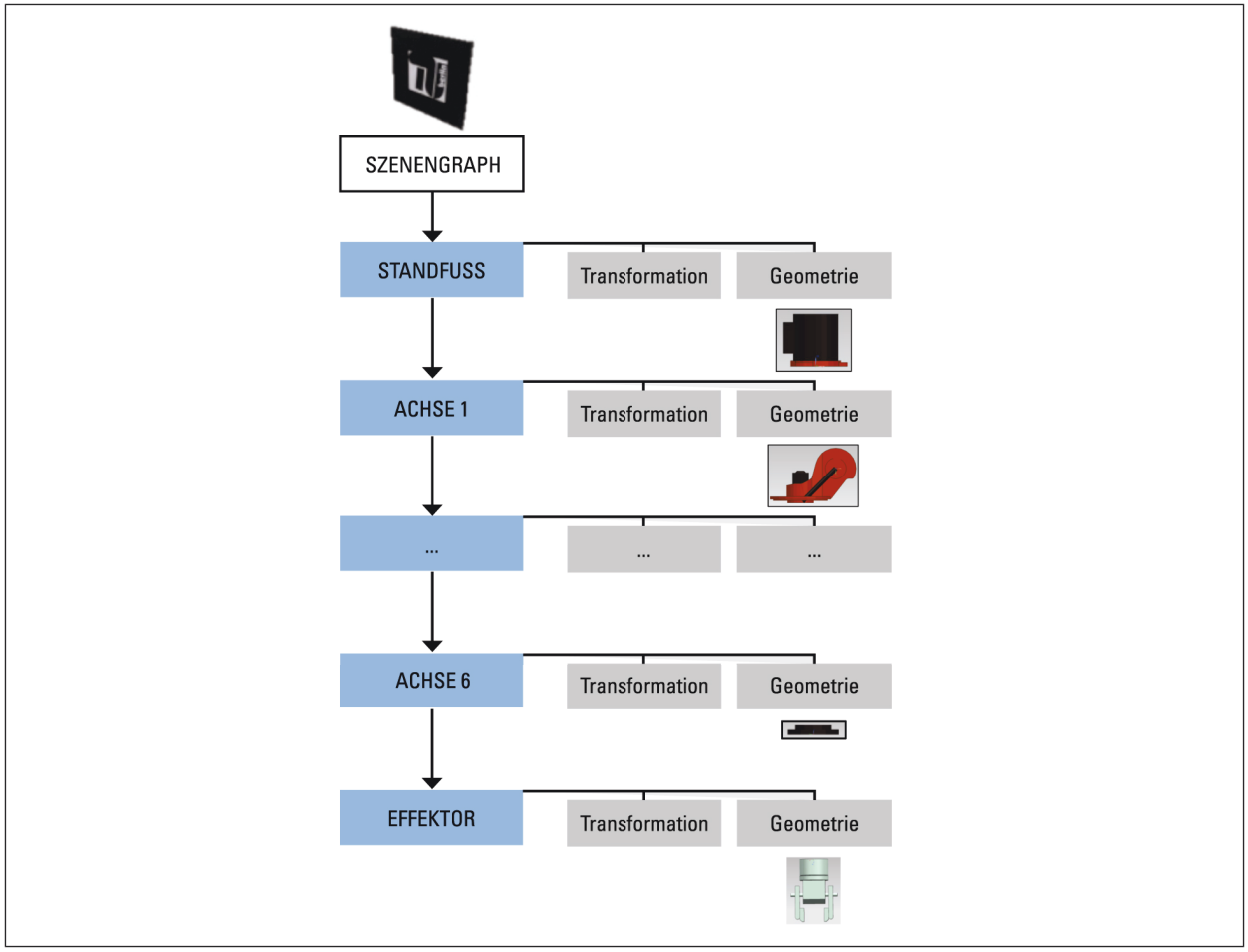

Abbildung 5.13: Aufbau eines Robotermodells über den Szenengraph, Knoten (blau) und deren Inhalt: Transformationen sowie 3D-Geometrie der virtuellen Objekte (grau).

tionsmatrix $\boldsymbol{T}(t_X, t_Y, t_Z)$, der Rotation $\boldsymbol{R}(Q)$ anhand einer Einheitsquaternion $Q(q_X, q_Y, q_Z, q_W)$ und einer Skalierung anhand der Matrix $\boldsymbol{S}(s_X, s_Y, s_Z)$.

$$
\begin{aligned}
\boldsymbol{T}_{Marker}^{Roboter} =& \boldsymbol{T}(t_X, t_Y, t_Z) \cdot \boldsymbol{R}(Q) \cdot \boldsymbol{S}(s_X, s_Y, s_Z) \\
=& \begin{pmatrix} 1 & 0 & 0 & t_X \\ 0 & 1 & 0 & t_Y \\ 0 & 0 & 1 & t_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \cdot \begin{pmatrix} 1-2\cdot(q_Y^2+q_Z^2) & 2\cdot(q_X\cdot q_Y - q_W\cdot q_Z) & 2\cdot(q_X\cdot q_Z - q_Y\cdot q_W) & 0 \\ 2\cdot(q_X\cdot q_Y + q_Z\cdot q_W) & 1-2\cdot(q_X^2+q_Z^2) & 2\cdot(q_Y\cdot q_Z - q_X\cdot q_W) & 0 \\ 2\cdot(q_X\cdot q_Z + q_Y\cdot q_W) & 2\cdot(q_Y\cdot q_Z - q_X\cdot q_W) & 1-2\cdot(q_X^2+q_Y^2) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\
& \cdot \begin{pmatrix} s_X & 0 & 0 & 0 \\ 0 & s_Y & 0 & 0 \\ 0 & 0 & s_Z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{5.2}
$$

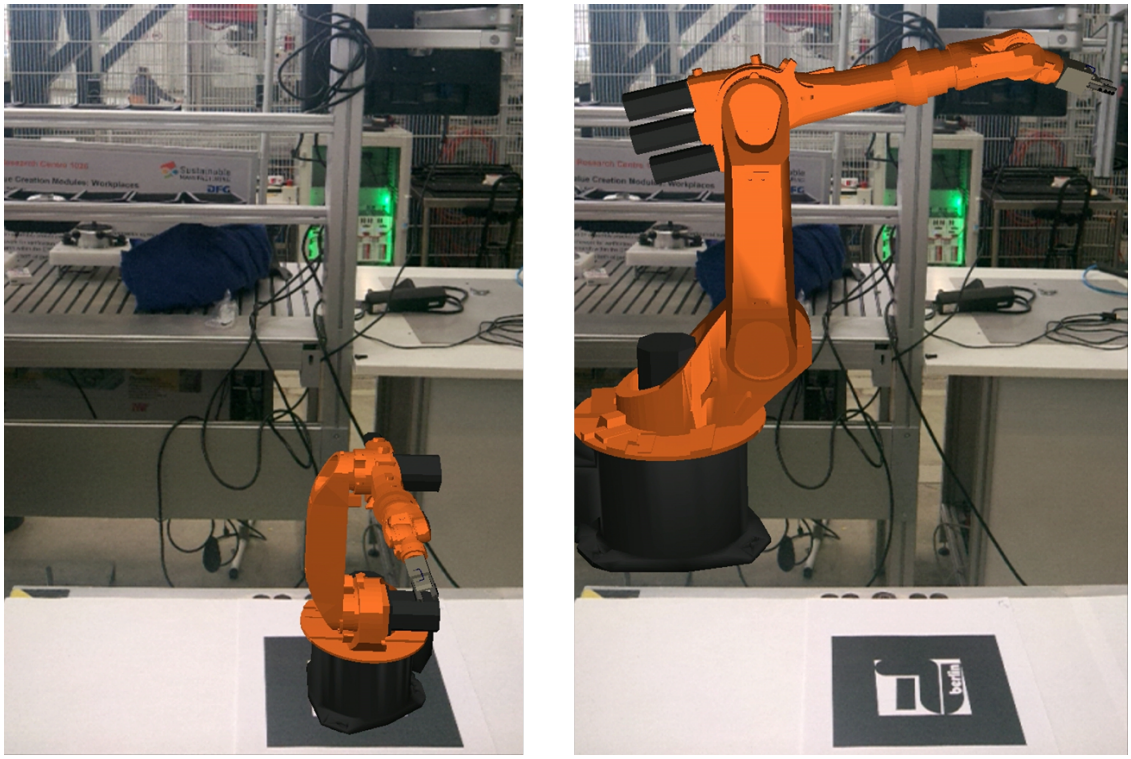

Abbildung 5.14: Darstellung des Robotermodells mit unterschiedlicher Positionierung, Orientierung und Skalierung.

E) Kinematik

Vorwärtskinematik

Zur Ermittlung der Flansch- bzw. Endeffektorpose $\boldsymbol{x}$ im kartesischen Basiskoordinatensystem des Roboters mithilfe der Gelenkwinkel $\boldsymbol{q}$ dient die Vorwärtstransformation (s. Formel 5.3). Die Berechnung der externen Koordinaten wird anhand von Formel 5.4 vorgenommen.

$$\boldsymbol{x} = f(\boldsymbol{q}) \tag{5.3}$$

$$\boldsymbol{T}_0^6 = \boldsymbol{T}_0^1(q_1) \cdot \boldsymbol{T}_1^2(q_2) \cdot \ldots \cdot \boldsymbol{T}_5^6(q_6) \tag{5.4}$$

Es erfolgt eine Verkettung der gelenkspezifischen DH-Transformationsmatrizen unter Berücksichtigung der aktuellen Gelenkvariablen. Die daraus resultierende Transformationsmatrix besteht aus der Rotationsmatrix $\boldsymbol{R}$ und einem Translationsvektor $\boldsymbol{t}$ der resultierenden Pose (s. Formel 5.5). Aus der Rotationsmatrix lässt sich gemäß verschiedenen Winkelkonventionen eine vereinfachte Darstellung der Orientierung extrahieren (s. Formel 5.6). Implementiert werden eine Darstellung der Orientierung als Quaternion-, ZY'Z"-Euler- sowie RPY-Notation (Berechnung, s. Anhang E).

$$\text{Translation: } \boldsymbol{T}_0^6(q_1, \ldots, q_6) \rightarrow \boldsymbol{t} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{5.5}$$

$$\text{Rotation (ZY'Z''-Euler): } \boldsymbol{T}_0^6(q_1,\ldots,q_6) \rightarrow \begin{pmatrix} \alpha_Z \\ \beta_{Y'} \\ \gamma_{Z''} \end{pmatrix} \tag{5.6}$$

Rückwärtstransformation
Mithilfe der inversen Kinematik werden bei gegebener externer Pose $\boldsymbol{x}$ die entsprechenden internen Gelenkwinkel $\boldsymbol{q}$ ermittelt (s. Formel 5.7). Da eine eindeutige Lösung des inversen Problems für klassische 6-Achs-Knickarm-Strukturen nicht gegeben ist, wird ein analytischer Algorithmus unter Vorgabe der DH-Parameter sowie einer spezifischen Achskonfiguration nach [177] implementiert. Als Eingabe des Algorithmus dient die homogene Transformationsmatrix der Endeffektorpose. Eine Beschreibung der Endeffektorpose durch die betrachteten Winkelkonventionen wird wiederum berücksichtigt, indem zunächst eine neutrale Darstellung in Form der Rotationsmatrix ermittelt wird.

$$\boldsymbol{q} = f^{-1}(\boldsymbol{x}) \tag{5.7}$$

F) Interpolation

Zum Verfahren des virtuellen Roboters in der Simulation werden verschiedene Interpolationsprinzipien genutzt. Analog zur realen Robotersteuerung dient die Interpolation zur Erzeugung von Bahnposen zwischen zwei vorgegebenen Posen im kartesischen Raum. Anschließend werden für die Bahnpunkte durch die inverse Kinematik Positionsvorgaben für die Einzelachsen berechnet. Die Achswerte werden erneut linear interpoliert und dienen zu diskreten Zeitpunkten als Positionsvorgabe für die virtuellen Achsen des Robotermodells.

Point to Point
Bei der PTP-Interpolation sind lediglich Start- und Endpose der Bewegung vorgegeben. Die einzelnen Achsen werden auf direktem Weg zu den jeweiligen Endpositionen bewegt. Die Bahn, welche der Endeffektor zurücklegt, ist nicht definiert, sondern resultiert aus den Bewegungen der Einzelachsen. Es werden drei Kategorien der PTP-Bewegung unterschieden: asynchrone, synchrone und vollsynchrone PTP-Steuerung. Bei den verschiedenen Ausführungen unterscheiden sich die Geschwindigkeitsprofile der Einzelachsen. So werden bei der synchronen PTP-Bewegung die Geschwindigkeiten der Einzelachsen derart angepasst, dass sämtliche Achsen für die Bewegung die gleiche Anfangs- und Endzeit besitzen. Bei der vollsynchronen PTP-Bewegung wird zusätzlich die Dauer der Beschleunigungs- und Bremsphasen der Einzelachsen angeglichen. In der App wird eine synchrone PTP-Bewegung realisiert. Das qualitative Vorgehen zur Erzeugung von synchronen PTP-Bewegungen durch Anpassung der Achsgeschwindigkeiten veranschaulicht Abbildung 5.15.

Linearinterpolation
Die lineare Bahnsteuerung beschreibt die Bewegung des TCP auf einer definierten Trajektorie im kartesischen Raum zwischen einer Start- und einer Endpose. Neben der Position wird auch die Orientierung des TCP zwischen Start- und Zielwert linear interpoliert. Die daraus resultierenden Winkeländerungen der Einzelachsen lassen sich demgemäß nicht mithilfe eines einzelnen trapezförmigen Geschwindigkeitsprofils beschreiben. Vielmehr ist das Ziel des Algorithmus die

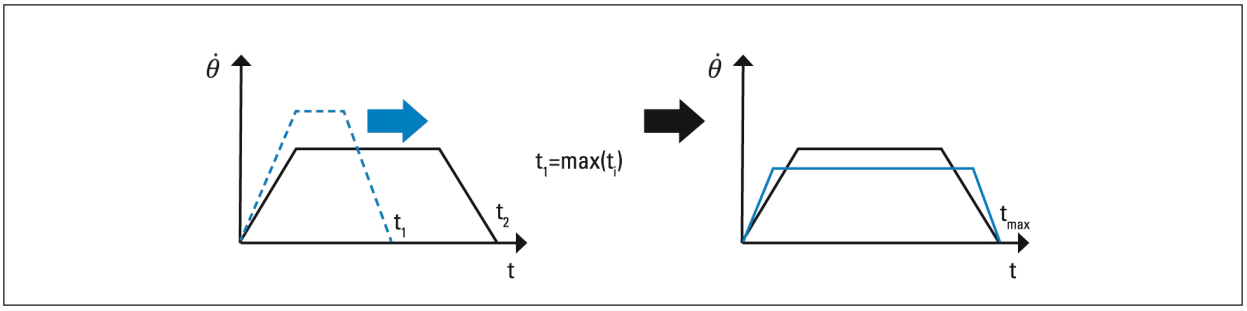

Abbildung 5.15: Qualitative Darstellung des Vorgehens zur Bestimmung der Geschwindigkeitsprofile bei synchroner PTP-Bewegung.

Beschreibung der Bahn anhand diskreter Bahnpunkte mithilfe eines Geschwindigkeitsprofils im kartesischen Raum. Zur Durchführung der Interpolation werden zunächst die translatorische und die rotatorische Verschiebung zwischen Start- und Zielpose berechnet (s. Abb. 5.16). Zu diesem Zweck wird die homogene Transformationsmatrix zwischen den beiden Posen ermittelt. Aus dieser Matrix werden Translation und Rotation in Quaterniondarstellung extrahiert. Diese Darstellung der Orientierung wird gewählt, da sie im Vergleich zur Darstellung über Euler- oder RPY-Winkel eine eindeutigere Beschreibung und im Vergleich zur Rotationsmatrix eine kompaktere Form aufweisen. Für die Interpolation ergeben sich zudem spezifische Vorteile der Berechnung durch geringere Rundungsfehler und Orthonormalisierung bei Verkettung von Rotationen. Für die Translation und Rotation wird jeweils ein Geschwindigkeitsprofil gemäß den Vorgaben für die Bahngeschwindigkeit und Beschleunigung ermittelt. Anschließend erfolgt auf Basis des Geschwindigkeitsprofils die Erzeugung diskreter Bahnposen.

Zirkularinterpolation

Das Bewegen des TCP auf einem Kreisabschnitt erfolgt durch die Vorgabe von drei Posenwerten, da sich anhand von drei Positionen eine Kreisbahn im 3D-Raum eindeutig beschreiben lässt. Der TCP ist lediglich auf dem Abschnitt zwischen Startposition $P1$ und Endposition $P3$ über den Zwischenpunkt $P2$ zu bewegen. Abbildung 5.17 beschreibt den geometrischen Zusammenhang der Kreisdefinition mit der Einführung verschiedener Hilfsgrößen. Das Vorgehen zur Erzeugung der

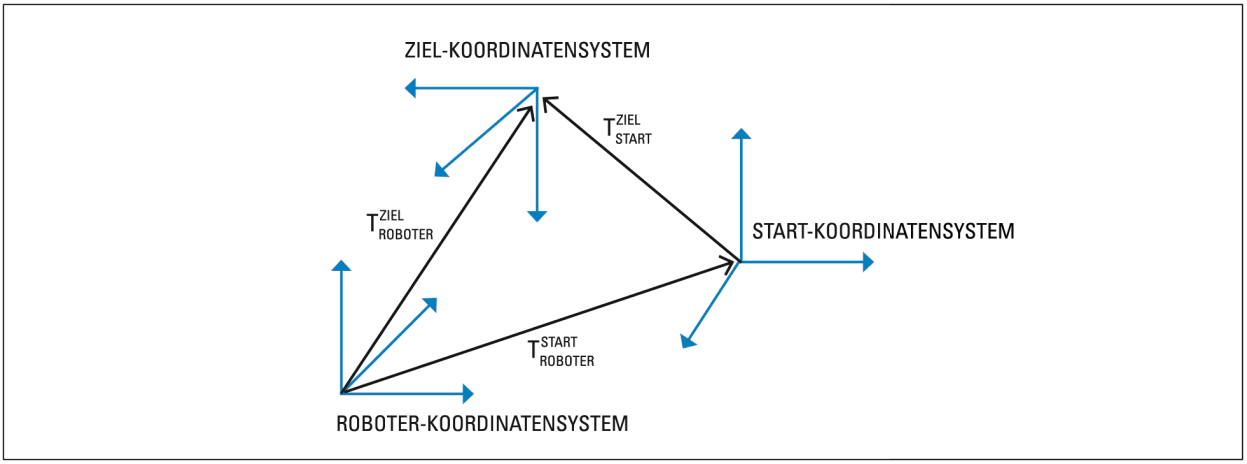

Abbildung 5.16: Geometrische Zusammenhänge von Linearbewegungen.

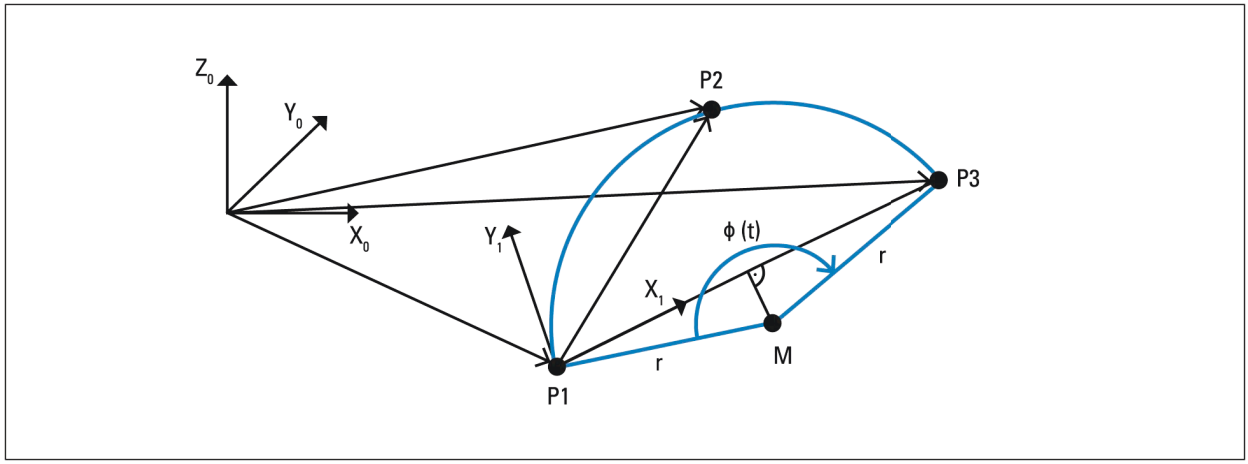

Abbildung 5.17: Darstellung geometrischer Zusammenhänge der Kreisinterpolation in Anlehnung an [63].

interpolierten Kreispunkte orientiert sich an [63, 177]. Dazu werden die Poseninformationen der vorgegebenen Bahnpunkte zunächst vom Welt-KS K_0 in das Hilfs-KS K_1 transformiert, um eine vereinfachte Ermittlung des Kreismittelpunkts durchführen zu können. Das Hilfs-KS wird derart gewählt, dass alle Kreispunkte in der XY-Ebene liegen. Der X-Wert des Mittelpunkts ergibt sich aus Formel 5.8. Der Y-Wert des Mittelpunkts lässt sich, unter Annahme, dass sämtliche Punkte der Kreisbahn denselben Abstand zum Mittelpunkt aufweisen, anhand von Formel 5.9 bestimmen.

$$m_X = \frac{P3_X - P1_X}{2} \tag{5.8}$$

$$m_Y = \frac{(P2_X - P1_x)^2 + (P2_Y - P1_Y)^2 - (P2_X - P1_X) \cdot (P3_x - P1_X)}{2 \cdot (P2_Y - P1_Y)} \tag{5.9}$$

Anhand des Mittelpunkts M und des Radius r lässt sich in Abhängigkeit vom Winkel ϕ anschließend die resultierende Kreisbahn beschreiben (s. Formeln 5.10 und 5.11). Diese dient erneut zur Definition eines Geschwindigkeitsprofils im kartesischen Raum. Aus diesem lassen sich einzelne Bahnpunkte ableiten. Bevor die Koordinaten an die inverse Kinematik übergeben werden, werden sie noch zurück in das ursprüngliche KS K_0 transformiert. Die Orientierung des TCP wird zwischen Startpose und Hilfspose sowie zwischen Hilfspose und Zielpose in Abhängigkeit von ϕ linear interpoliert.

$$P_{CircX} = 2r \cdot \sin(\frac{\phi}{2}) \cdot \sin(\frac{\phi}{2}) - 2(M_Y, M_X) \tag{5.10}$$

$$P_{CircY} = 2r \cdot \sin(\frac{\phi}{2}) \cdot \cos(\frac{\phi}{2}) - 2(M_Y, M_X) \tag{5.11}$$

Spline-Interpolation

Zur Darstellung von Freiformkurven soll eine Spline-Interpolation verwendet werden. Diese dient der Rekonstruktion der räumlichen Handbewegungen für komplexe Bewegungsbahnen. Auf eine Rückführung von Freiformkurven auf Linear- und Zirkularbewegungen wie bspw. in [93] soll aufgrund eines erhöhten Aufwands und unstetigem Übergang zwischen den Teilsegmenten verzich-

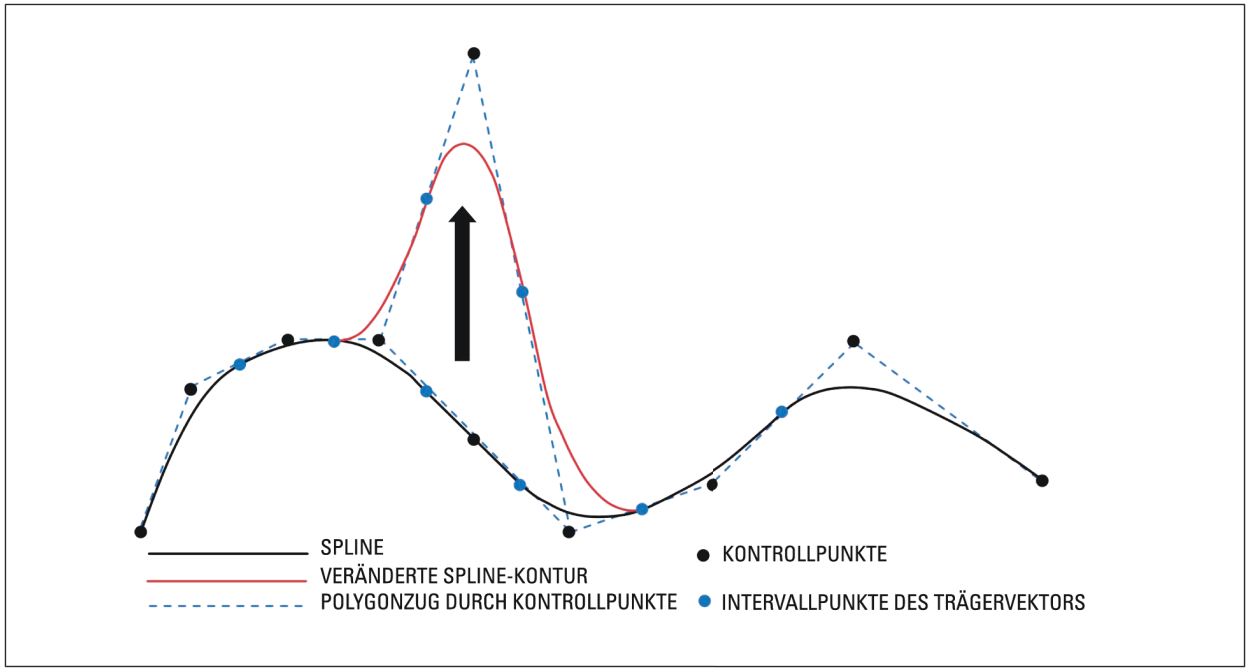

Abbildung 5.18: Darstellung eines B-Splines mit modifiziertem Kontrollpunkt in Anlehnung an [178].

tet werden. Es existieren verschiedene Verfahren der Spline-Interpolation mit spezifischen Eigenschaften, wobei die Wahl des Verfahrens vom jeweiligen Anwendungsfall abhängt. Eine gängige Spline-Art zur Darstellung von 3D-Freimformkurven in der Robotik sind kubische Splines [67]. Diese sind stückweise Polynome zwischen den Stützstellen vom Grad drei mit Stetigkeit in den Übergängen bis zu Grad zwei. Kubische Splines zeichnen sich durch geringen Berechnungsaufwand aus. Jeder Stützpunkt hat jedoch Einfluss auf den gesamten Verlauf der Kurve, sodass lokale Änderungen in einer kompletten Neuberechnung resultieren. Zudem sind kubische Splines in der Darstellung von beliebigen Kurvengeometrien lediglich beschränkt einsetzbar.
B(asis)-Splines verwenden im Unterschied zu kubischen Splines Basisfunktionen zur Bildung der stückweise polynominalen Funktionen. Somit ist zur Beschreibung der Kurve neben Raumpunkten auch ein Parametervektor, der sogenannte „Knotenvektor“, erforderlich. Durch dieses Vorgehen unterscheidet sich der resultierende Kurvenverlauf im Vergleich zu den kubischen Splines. Die verwendeten Stützpunkte stellen keine Kurvenpunkte dar, sondern approximieren diese lediglich. Der erhebliche Vorteil der B-Splines liegt darin, dass sich Änderungen an den Kontrollpunkten nur lokal auswirken (s. Abb. 5.18), weshalb sie ein beliebtes Tool zur Anpassung von Oberflächen in CAD-Anwendungen sind [178].
Anders als bei den B-Splines lassen sich bei Non-Uniform Rational Basis-Splines (NURBS) Knotenpunkte in ungleichmäßigen Intervallen anlegen, sodass durch den Intervallabstand ein weiterer Verstellparameter zur Anpassung des Kurvenverlaufs bereitgestellt wird. Durch das ungleichmäßige Anpassen der Knotenvektoren sind NURBS im Gegensatz zu allgemeinen B-Splines in der Lage, mathematisch exakt auch Kreise und Kegelschnitte zu beschreiben und somit beliebige vorgegebene Bahnpunkte exakt anzufahren [179]. Aufgrund der Darstellung beliebiger Kurvengeometrien, der lokalen Adaptierbarkeit der Kurven ohne Neuberechnung der gesamten Kurve und der Möglichkeit zum direkten Anfahren der Stützpunkte wird in der Simulation eine Interpolation

auf Basis von NURBS umgesetzt. Das mathematische Vorgehen zur Ermittlung der interpolierten Bahnpunkte kann Anhang F entnommen werden.

G) Umsetzung der Visualisierung und Bewegungsanimation

Die Umsetzung der Bewegungsanimation des Roboters erfolgt gemäß den kinematischen Vorgaben der Feininterpolation. Die Datenstruktur des Szenengraphs beinhaltet Informationen zum Aufbau der virtuellen Szene. Diese umfasst sowohl die Struktur des Roboters als auch das Programm in Form von einzelnen Posen, Trajektorien oder aufgabenorientierten Objektdarstellungen. Der Szenengraph selbst verfügt nicht über die entsprechende Funktion, um die virtuelle Szene visuell darzustellen. Diese Funktion wird durch eine gekapselte Datenstruktur bereitgestellt. Die Trennung von Datenhaltung und Darstellung besitzt den Vorteil, dass die Modellierung der virtuellen Szene unabhängig von der Form der grafischen Darstellung vorgenommen werden kann. Das Zeichnen des Roboters und des Programms im einzelnen Frame erfolgt über einen speziellen Algorithmus, dessen Ablauf Abbildung 5.19 veranschaulicht. Erfolgt der Befehl zur Darstellung der Szene im neuen Frame, werden zunächst mittels der Feininterpolation Achswerte erzeugt. Anhand dieser Achswertvorgaben wird der Szenengraph aktualisiert. Anschließend erfolgt der Aufruf des sogenannten „PrintVisitor“, welcher die Aufgabe eines Visitor Pattern [180] übernimmt. Der PrintVisitor durchläuft die Datenstruktur des Szenengraphs sequentiell und zeichnet durch den Renderer auf Basis der aktuellen Parameter die einzelnen Komponenten auf die Zeichenoberfläche GL Surface.

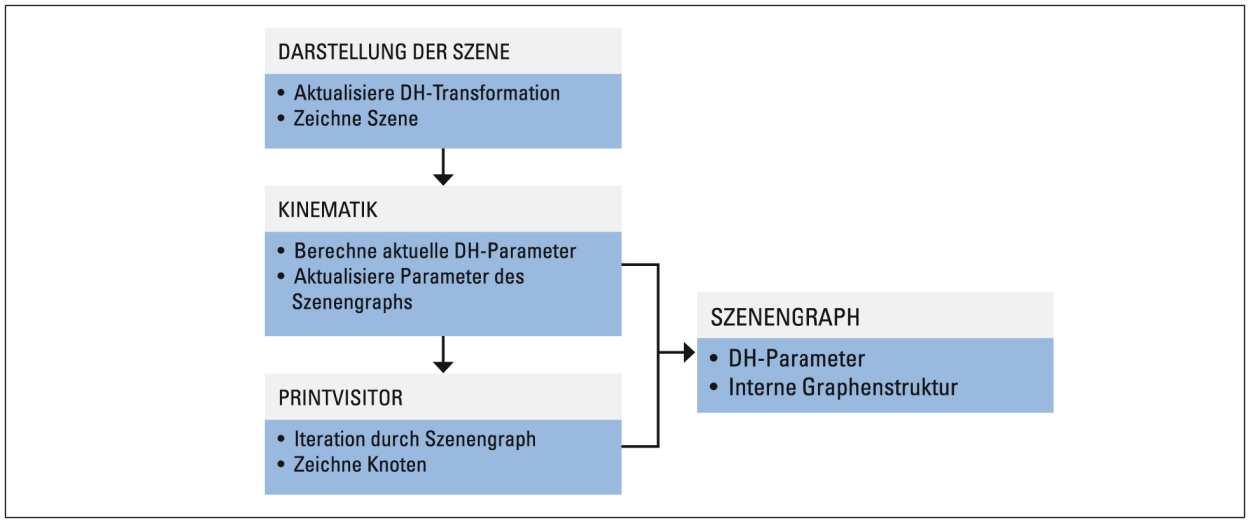

Abbildung 5.19: Ablauf der Roboterdarstellung durch den PrintVisitor anhand des Szenengraphs und der Kinematikberechnung.

H) Arbeitsraumüberwachung und Konfigurationstreue

Für die Bahnplanung werden zwei zusätzliche Methoden zur Unterstützung der Simulation implementiert. Zunächst erfolgt eine Arbeitsraumüberwachung auf Basis einer Überprüfung der definierten Minimal- und Maximalwerte der Einzelachsen. Somit können Achsvorgaben, welche sich außerhalb des Arbeitsbereichs der Einzelachsen befinden, schon bei der Eingabe detektiert werden.

Zusätzlich wird eine Überprüfung auf Arbeitsraumverletzung für kartesische Positionsvorgaben vorgenommen.
Eine weiterführende Kontrolle gilt der Realisierung der Konfigurationstreue der Achsen für einzelne Bahnbewegungen. Zu diesem Zweck erfolgt eine erweiterte Betrachtung der inversen Kinematik. Ziel ist es, bei der Bahnplanung eine Achskonfiguration zu identifizieren, welche für die Gesamtheit der interpolierten Posen ohne Überschreitung des Arbeitsbereichs von Einzelachsen realisierbar ist. Bei der Wahl der Konfiguration wird ferner ein möglichst geringer Verfahrweg der Einzelachsen als Kriterium verwendet. Dieses Vorgehen orientiert sich am Ablauf der Bahnplanung moderner Industrierobotersteuerungen. Abbildung 5.20 veranschaulicht die Wahl verschiedener Achskonfigurationen für eine Linearbahn. Eine Überprüfung auf einen potentiellen Konfigurationswechsel wird vor der Ausführung der Simulation durchgeführt. Wird für einzelne Posen eine Arbeitsraumüberschreitung einzelner Achsen festgestellt, versucht der Algorithmus die vorgegebene Achskonfiguration zu variieren, bis sämtliche Posen der Trajektorie erreichbar sind. Kann keine passende Achskonfiguration gefunden werden, wird ein Fehler zurückgegeben.

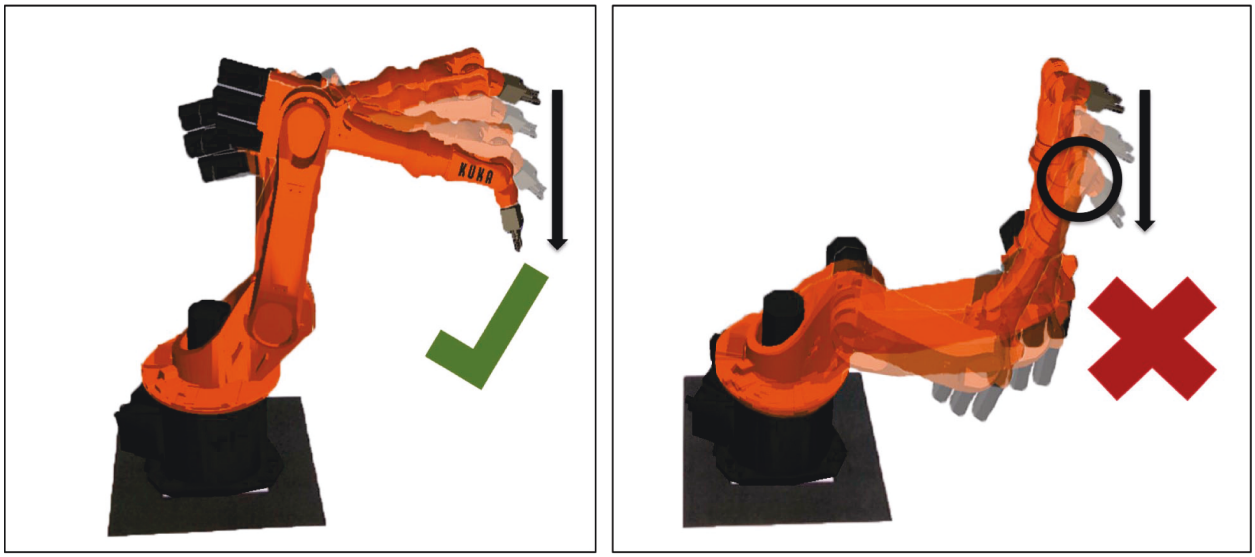

Abbildung 5.20: Linearbewegung ohne Arbeitsraumverletzung (links) und mit Arbeitsraumverletzung bei Wahl einer alternativen Achskonfiguration (rechts).

5.2 Umsetzung der gestenbasierten Eingabe

5.2.1 3D-Motion-Tracking

Die Umsetzung des 3D-Motion-Tracking wird durch den Kinect-Sensor[13] und das Skeleton-Tracking auf Basis des OpenNi-Framework[14] realisiert. Die Kinect wird an einen externen PC angeschlossen, während die Position der Hände kontinuierlich über drahtlose TCP/IP Kommunikation an die App versendet wird und dort zur weiteren Verarbeitung bereitsteht.

[13] Vgl. Kapitel 4.3.2.
[14] Vgl. http://www.openni.org/ - 20.01.2014.

Vor dem Versenden erfolgt die Vorverarbeitung der Sensordaten in Form einer Plausibilitätsprüfung zur Identifikation und Vernachlässigung von Ausreißern sowie einer gleitenden Mittelwertfilterung mit einer Fensterbreite $n \in \mathbb{N}$ (s. Formel 5.12) zur Steigerung der Robustheit der 3D-Daten. $m(t)$ bezeichnet den gemittelten Koordinatenwert, welcher anhand des aktuellen Messwerts $x(t)$ und n vorherigen Messwerten ermittelt wird. Abbildung 5.21 veranschaulicht das Vorgehen zur Ermittlung der 3D-Information der Hand von der Aufnahme der Sensordaten bis zum Empfang der Handkoordinaten auf dem Mobilgerät.

$$m(t) = \frac{1}{n} \sum_{i=0}^{n-1} x(t-i) \tag{5.12}$$

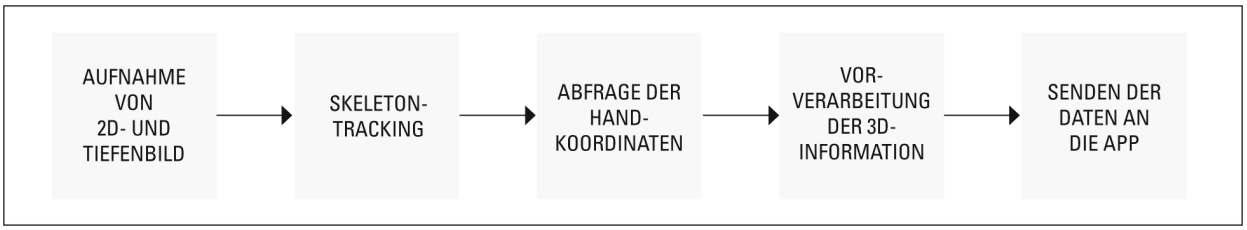

Abbildung 5.21: Vorgehen zur Ermittlung und Übersendung der Handkoordinaten.

5.2.2 Fingergestenerkennung

A) Umsetzung der Hauterkennung

Circular Ring Klassifikator

Aufbauend auf dem in [158] vorgestellten regelbasierten LC-Klassifikator wird im Rahmen dieser Arbeit ein weiterer Klassifikator implementiert. Die Motivation hierzu liegt in der Reduzierung der Anzahl der betrachteten Dimensionen zur Steigerung der Performance auf dem Mobilgerät. Nach [158] ist dies im RGB-Raum lediglich durch die Normalisierung der RGB-Werte möglich, was wiederum in einem erhöhtem Rechenaufwand resultiert. Durch die Trennung der Luminanz (Y) und der Farbwerte (UV) im YUV-Raum ist für den vorliegenden Anwendungsfall eine Reduktion der Dimension auf Zwei möglich. Des Weiteren stellt der YUV-Farbraum auf Mobilgeräten in der Regel einen nativen Farbraum der Kameraanwendungen dar. Somit kann auf eine zusätzliche Farbtransformation verzichtet werden.

Für die Analyse der Farbverteilung der Haut wird eine große Menge von Trainingsdaten verschiedener hellhäutiger Nutzer unter verschiedenen Umgebungsbedingungen ausgewertet. Trainingsdaten dunkelhäutiger Nutzer konnten aufgrund fehlender Verfügbarkeit nicht aufgenommen werden. Die Ergebnisse der Versuchsdurchführungen (s. Abb. 5.22) identifizieren eine charakteristische Region im YV-Raum. Diese Region lässt sich annähernd über ein Kreisringsegment (engl. *circular ring* (CR)) beschreiben. Mithilfe folgender Formeln lässt sich demnach erkennen, ob ein Pixel der Haut zuzuordnen ist:

$$\sqrt{(Y - Y_0)^2 + (V - V_0)^2} < r \tag{5.13}$$

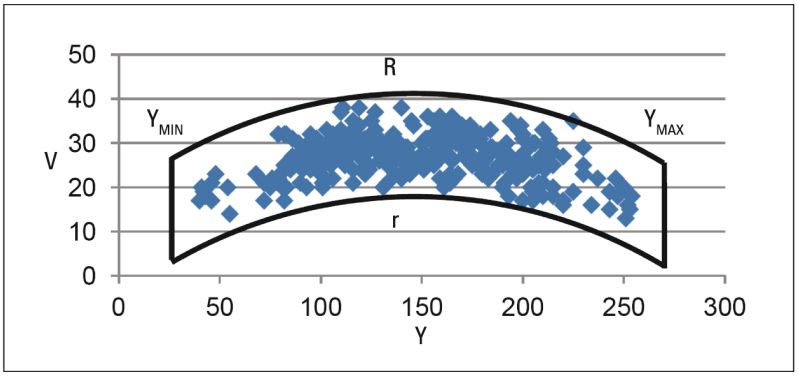

Abbildung 5.22: Erhobene Trainingsdaten zur Hautfarbe in der YV-Ebene mit dem regelbasierten CR-Klassifikator.

$$\sqrt{(Y - Y_0)^2 + (V - V_0)^2} > R \tag{5.14}$$

$$Y > Y_{min} \tag{5.15}$$

$$Y < Y_{max} \tag{5.16}$$

Die Anwendung dieser vier einfachen Regeln mit den Parametern Y_0, V_0, r, R, Y_{min} und Y_{max} ermöglicht die schnelle regelbasierte Klassifikation. Des Weiteren kann der Klassifikator anhand dieser Parameter gemäß veränderten Umgebungseinflüssen angepasst werden. Abbildung 5.23 illustriert das Ergebnis der Hautfarbenerkennung mithilfe des vorgestellten CR-Klassifikators.

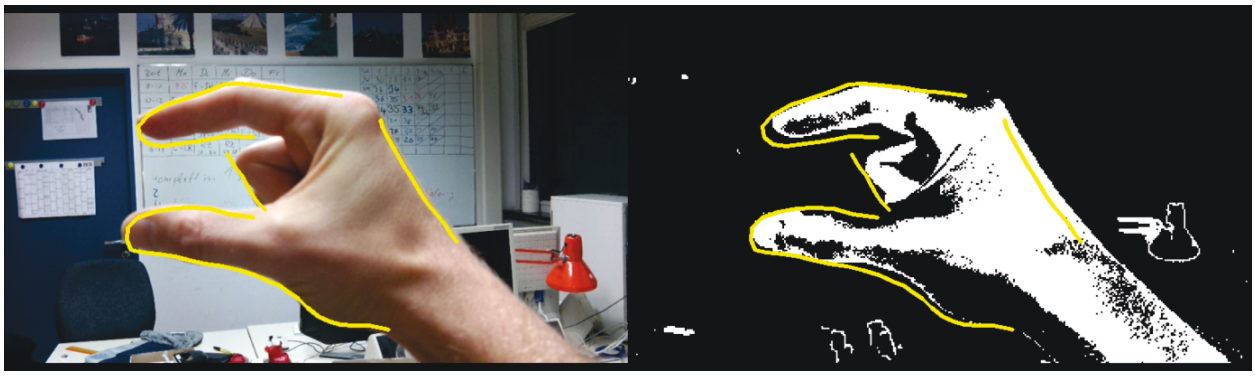

Abbildung 5.23: Ergebnis der pixelbasierten Hautfarbenerkennung mit dem CR-Klassifikator.

Initialisierung des Klassifikators

Die Parametrierung des CR-Klassifikators ermöglicht eine Adaption des Hautmodells an die spezielle Hautfarbe des Nutzers und an die aktuelle Beleuchtungssituation. Zu diesem Zweck wird der in [158] aufgestellte Ansatz auf den CR-Klassifikator übertragen. Demnach wird zunächst eine Initialisierung vorgenommen, welche die Hand des Nutzers an einer vorgegebenen Stelle im Bild detektiert. Durch die Multiplikation des Bilds mit zwei Masken ergeben sich eine Maske mit

hautfarbenen Pixeln und eine für den Hintergrund. Um die Parameter des Hautmodells mithilfe der hautfarbenen Maske zu optimieren, wird ein sogenannter „Fitness“-Wert definiert, den es zu maximieren gilt:

$$fitness = \frac{n_{s,SM}}{N_{SM}} \cdot TI - \frac{n_{s,BM}}{N_{BM}} \tag{5.17}$$

$n_{s,SM}$ und $n_{s,BM}$ sind die Anzahl der hautfarbenen Pixel innerhalb der Haut- und Hintergrundmaske. N_{SM} und N_{BM} sind die Größen beider Masken. TI ist der sogenannte „Target Importance Value“. Dieser beschreibt den Fokus der Optimierung. Hohe Werte für TI bewirken eine hohe Erkennungsrate von hautfarbenen Pixeln innerhalb der Hautmaske. Gleichzeitig werden jedoch auch verstärkt Pixel im Hintergrund als Hautfarbe eingestuft. Zur Optimierung des Fitness-Werts wird ein Algorithmus mit adäquatem Abbruchkriterium erstellt. Abbildung 5.24 veranschaulicht die Anpassung der Parameter des CR-Klassifikators durch den Algorithmus.

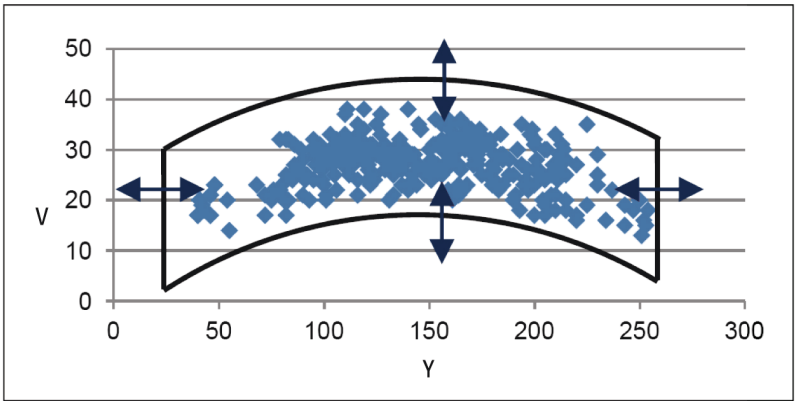

Abbildung 5.24: Initialisierung des CR-Klassifikators durch Parameteranpassung.

B) Beschreibung der Handkontur

Die Erkennung der Hand im Bild basiert auf einer Detektion und Verfolgung der Handkontur. Für den vorliegenden Ansatz beschreibt die Kontur die Umrisslinie der Hand im Bild. Diese dient somit als Trennung zwischen Handregion und Hintergrund. Bei hohen Bildauflösungen führt eine pixelgenaue Beschreibung der Handkontur schnell zu einer hohen Komplexität der weiteren Tracking-Schritte. Aus diesem Grund wird in dieser Arbeit eine kompakte Beschreibung der Handkontur durch eine begrenzte Anzahl von charakteristischen Stützpunkten gewählt. Eine genauere Beschreibung der Handkontur wird durch Interpolation zwischen diesen Kontroll- bzw. Stützstellen erreicht. Blake und Isard [165] stellen zu diesem Zweck die Verwendung von B-Splines vor. Diese bieten eine einfache und effiziente Möglichkeit, natürlich wirkende Bahnkurven durch eine geringe Anzahl an Kontrollpunkten zu beschreiben. Demgemäß wird die Handkontur als parametrierte B-Spline folgendermaßen definiert:

$$\begin{pmatrix} x(s) \\ y(s) \end{pmatrix} = \boldsymbol{B}(s) \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \tag{5.18}$$

$\boldsymbol{x}$ und $\boldsymbol{y}$ sind Spaltenvektoren, welche die X und Y-Koordinaten der einzelnen Kontrollpunkte beinhalten. $B(s)$ ist eine metrische Matrix, deren Einträge die Basisfunktionen an der Position s repräsentieren. Für weitere Details des genutzten B-Spline-Algorithmus sei auf Anhang F verwiesen.

Deformable Templates

Für die robuste, markerlose Gestenerkennung ist das Tracking der Hand in der Bild- bzw. Videosequenz ein notwendiger Schritt. Für die vorliegende Beschreibung der Handkontur bedeutet dies, dass eine Tracking-Lösung für die einzelnen Kontrollpunkte gefunden werden muss. Hierzu kann die Verschiebung eines Kontrollpunkts zwischen zwei Bildern bzw. Zuständen allgemein durch eine Transformationsmatrix beschrieben werden:

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \boldsymbol{T}_m^0 \cdot \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \tag{5.19}$$

x_0 und y_0 beschreiben die Koordinaten der Kontrollpunkte im ursprünglichen Koordinatensystem. Mithilfe der Transformationsmatrix T_m^0 kann die Kontur in einen Zustand m durch Translation, Rotation und Skalierung überführt werden. Bezüglich der Translation wird eine Verschiebung in X- und Y-Richtung der Bildebene sowie eine Verschiebung in Z-Richtung, also in die Bildebene hinein oder aus ihr hinaus, vorgesehen. Die Betrachtung der Rotation wird aus Komplexitätsgründen auf eine Rotation um die Z-Achse beschränkt. Im komplexeren Fall würde ein 3D-Modell der Hand benötigt. Der damit verbundene erhöhte Rechenaufwand stände in einem unverhältnismäßigen Bezug zum erreichten Nutzen. Des Weiteren wäre die echtzeitfähige Bedienbarkeit der Anwendung auf aktuellen Handheld-Geräten nicht gegeben. Abbildung 5.25 veranschaulicht die betrachteten Freiheitsgrade der Handkontur.

Die Translation der Handkontur (s. Abb. 5.25, oben) wird durch den Vektor $\begin{pmatrix} t_x & t_y \end{pmatrix}^T$ und die Rotation um die Z-Achse durch den Winkel ϕ bzw. die zugehörige Rotationsmatrix beschrieben. Die Translation um die Z-Achse beschreibt ein Skalierungsvektor $\boldsymbol{s}$ mit uniformer Skalierung $s = s_x = s_y$. Somit ergibt sich folgende Gleichung:

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{5.20}$$

Werden die Parameter t_x, t_y, ϕ, s_x und s_y in einem definierten State Vector $\boldsymbol{x}$ separiert, ergibt sich:

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_0 & 0 & 0 & -y_0 \\ 0 & 1 & 0 & y_0 & x_0 & 0 \end{pmatrix} \cdot \begin{pmatrix} t_x \\ t_y \\ s_x\cos\phi \\ s_y\cos\phi \\ s_x\sin\phi \\ s_y\sin\phi \end{pmatrix} \tag{5.21}$$

Gleichung 5.21 lässt sich vereinfacht ausdrücken als:

$$\begin{pmatrix} x_m \\ y_m \end{pmatrix} = \boldsymbol{W} \cdot \boldsymbol{x} \tag{5.22}$$

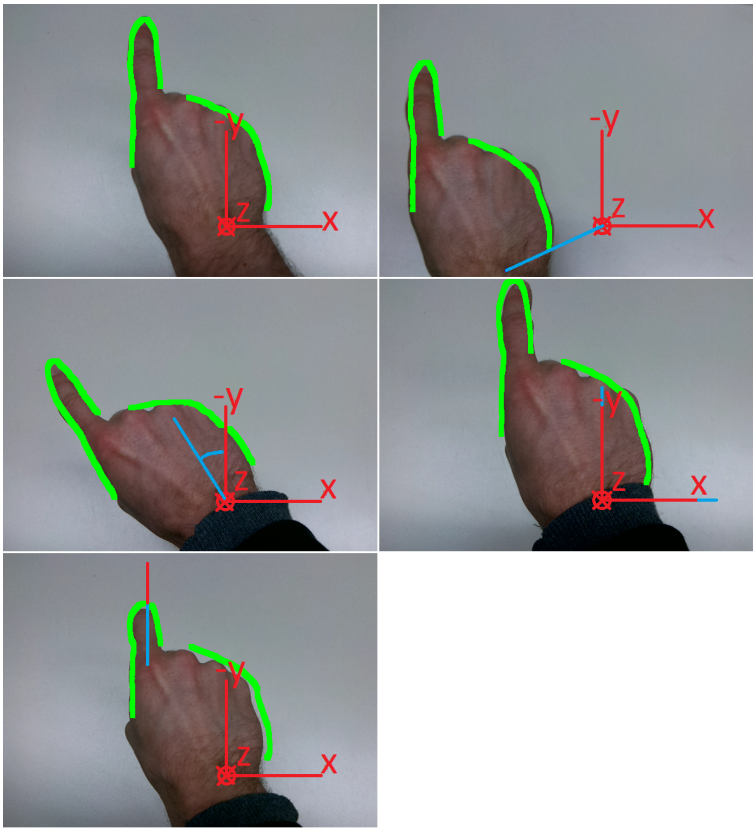

Abbildung 5.25: Betrachtete Handbewegungen: Translation (oben), Rotation (Mitte) und Skalierung, hier: einzelner Finger (unten).

W ist die sogenannte „Shape Matrix“. Des Weiteren ist es die entscheidende Aufgabe des Tracking, den State Vector für jedes neue Frame der Videosequenz zu ermitteln.

C) Messmodell

Der vorgestellte Erkennungsalgorithmus benötigt einen Messwert, um die Übereinstimmung zwischen Bildmerkmalen der Hand des Nutzers und der hypothetischen Kontur des Algorithmus festzustellen. Die hypothetische Kontur wird dargestellt durch den State Vector. Die Übereinstimmung lässt sich als bedingte Wahrscheinlichkeit beschreiben, welche im Folgenden als „Contour Likelihood“ bezeichnet wird. In der Fachliteratur bestehen verschiedene Ansätze zur Berechnung der Contour Likelihood. MacCormick [181] präsentiert eine breite Diskussion der Vor- und Nachteile der einzelnen Berechnungsmöglichkeiten. Der vorliegende Ansatz erfolgt in Übereinstimmung mit den Anwendungen in [158] und [168], wonach Messlinien an spezifischen Punkten senkrecht zur Handkontur erstellt werden (s. Abb. 5.26). Die Messlinien dienen der gezielten Suche nach Bildmerkmalen. Entlang der Messlinie wird nach einem Übergang von nicht-hautfarbenen zu hautfarbenen Pixeln gesucht und der Abstand σ zum Linienmittelpunkt ermittelt. In Anlehnung an [168] wird eine Gaußfunktion zur weiteren Verarbeitung der Messlinie genutzt.

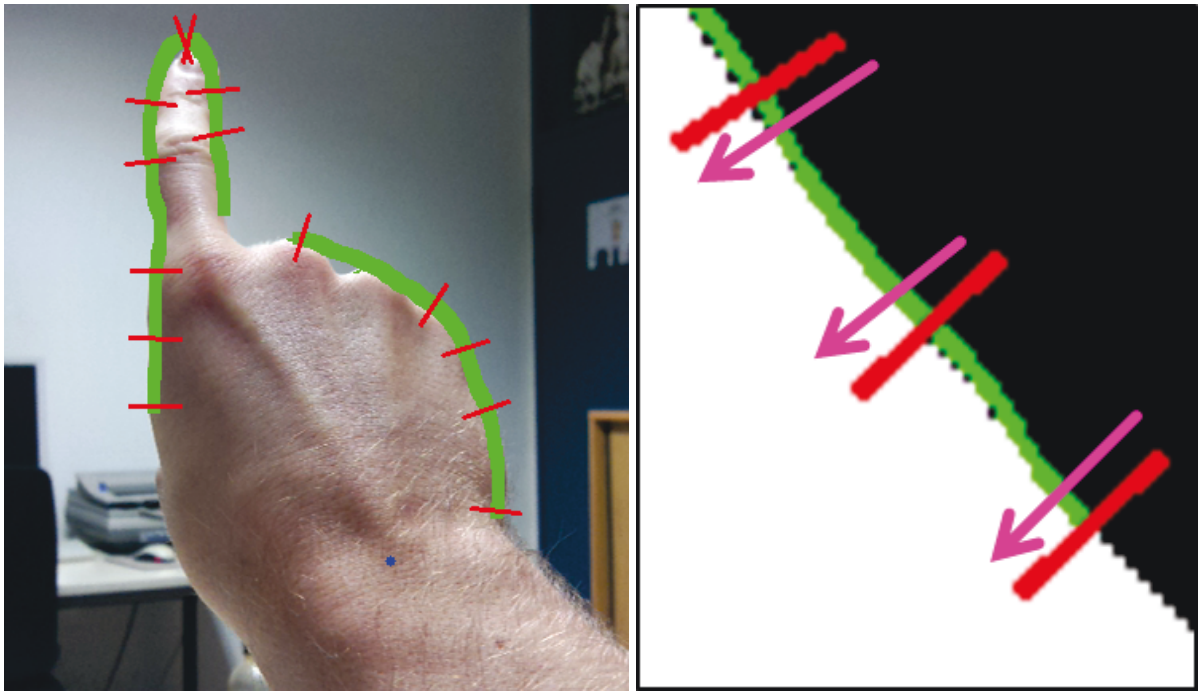

Abbildung 5.26: Ausrichtung (links) der Messlinien (rot) an Handkontur (grün) und Suchbewegung (rechts) nach Übergang: Hintergrund zu Haut.

Die Übereinstimmung der gesamten Kontur, bestehend aus $M \in \mathbb{N}$ Messlinien, ergibt sich aus dem Produkt der Einzelwahrscheinlichkeiten der Messlinien:

$$p(z_t|\boldsymbol{x}_t) = \prod_{m=1}^{M} p(z_m|\boldsymbol{x}_m) \tag{5.23}$$

Die benötigte Rechenzeit zur Ermittlung der Übereinstimmung $p(z_t|\boldsymbol{x}_t)$ hängt direkt von der Anzahl der Messlinien ab. Dementsprechend wird für den konkreten Anwendungsfall eine für das Tracking hinreichende, aber möglichst geringe Anzahl benötigter Messlinien entlang der Kontur empirisch ermittelt.

D) Umsetzung des Tracking durch den Condensation-Algorithmus

Die Umsetzung des Condensation-Algorithmus besteht aus drei Schritten, die für jeden Partikelsatz nacheinander durchgeführt werden:

1. Resampling,
2. Vorhersage und
3. Messung.

Resampling

Im Anschluss an den einzelnen Durchlauf des Condensation-Algorithmus besitzen die Partikel unterschiedliche Gewichtungsfaktoren. Geringe Werte stehen für eine geringe Übereinstimmung der hypothetischen Kontur mit der aktuellen Handkontur des Nutzers. Hohe Werte unterstellen eine hohe Übereinstimmung. Mit dem Ziel, die Weiterverarbeitung der Partikel effizient zu halten, werden

Partikel mit geringem Gewichtungsfaktor verworfen. In der Fachliteratur lassen sich verschiedene Ansätze zur Realisierung des Resamplings finden, u. a. [166] und [182]. Tosas [158] schlägt einen heuristischen Ansatz vor, welcher eine konstante und begrenzte Rechenzeit für das Resampling in jedem Schritt sicherstellt. Abhängig vom Wert der Gewichtungsfaktoren werden die besten 10 % der Partikel aus dem vorherigen Durchlauf ausgewählt. Diese werden als Kopien in den aktuellen Partikelsatz übernommen. Anschließend werden die Gewichtungsfaktoren $\pi_{s,i}$ normalisiert und mit der Anzahl n der Partikel multipliziert. Abhängig vom Wert der Gewichtungsfaktoren wird eine Anzahl r_i an Kopien der Partikel erstellt (s. Formel 5.24). Abbildung 5.27 verdeutlicht diesen Ansatz beispielhaft für eine Anzahl von $n = 30$ Partikeln.

$$r_i = \frac{\pi_{s,i}}{\sum_{i=1}^{n} \pi_{s,i}} \cdot n \tag{5.24}$$

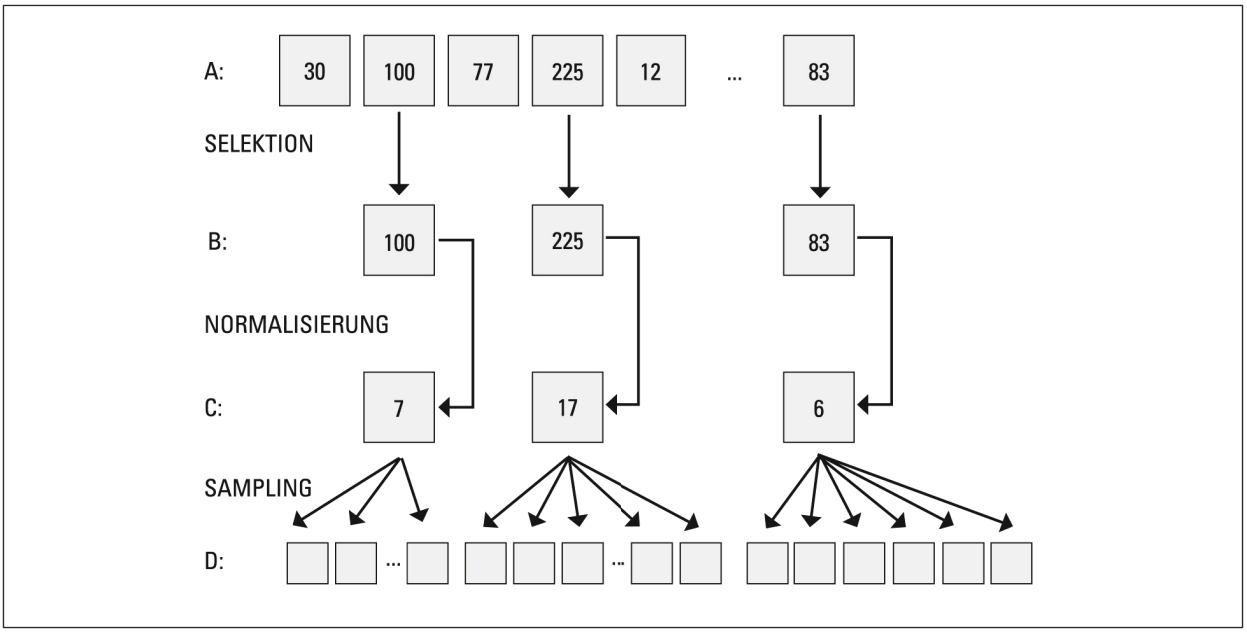

Abbildung 5.27: Ablauf des Resamplings: Ausgangszustand (A), Selektion der besten Partikel (B), Normalisierung der Gewichtungsfaktoren (C) und Erstellung neuer Partikel (D).

Vorhersage

Im Schritt der Vorhersage (engl. *prediction*) werden hypothetische Bewegungen der ausgewählten Partikel simuliert. Sowohl Isard und Blake [168] als auch Tosas [158] nutzen dazu einen autoregressiven Prozess zweiter Ordnung. Demgemäß lässt sich der aktuelle State Vector $\boldsymbol{x}_t$ als lineare Kombination aus einer deterministischen und einer stochastischen Verschiebung ausdrücken:

$$\boldsymbol{x}_t = \boldsymbol{A}_2\boldsymbol{x}_{t-2} + \boldsymbol{A}_1\boldsymbol{x}_{t-1} + \boldsymbol{B}\boldsymbol{w} \tag{5.25}$$

$\boldsymbol{A}_1$, $\boldsymbol{A}_2$ und $\boldsymbol{B}$ sind $d \times d$ Matrizen, welche konstante, empirisch ermittelte Einträge enthalten. $\boldsymbol{w}$ ist ein $d \times 1$ Vektor, der unabhängige, normalverteilte Zufallszahlen $N(0,1)$ enthält. $\boldsymbol{A}_2\boldsymbol{x}_{t-2} + \boldsymbol{A}_1\boldsymbol{x}_{t-1}$ bezeichnet den deterministischen Drift der Kontur, welcher die Fortführung der aktuellen Bewegung repräsentiert. $\boldsymbol{B}\boldsymbol{w}$ ist die stochastische Komponente, welche den State Vector streut, um

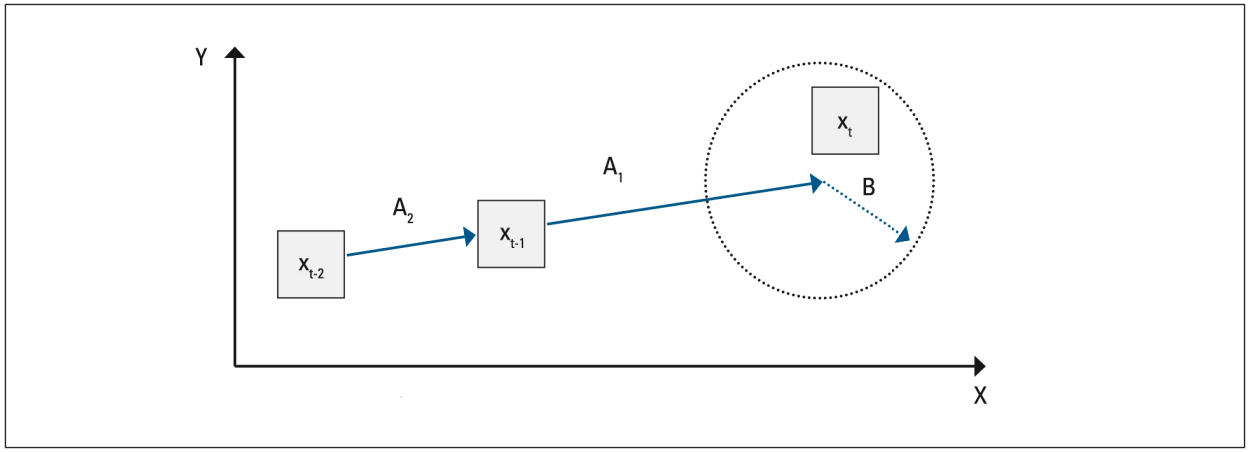

Abbildung 5.28: Vorhersage, bestehend aus deterministischem (A_1 und A_2) und stochastischem (B) Anteil.

Ungenauigkeiten der deterministischen Simulation auszugleichen. Abbildung 5.28 veranschaulicht den Vorhersageschritt für einen zweidimensionalen State Vector.

Messung
Im Messschritt wird die Übereinstimmung der in den vorherigen Schritten erstellten und verteilten Partikel mit der realen Hand im Bild bestimmt. Die Übereinstimmung der Partikel wird durch den Gewichtungsfaktor ausgedrückt, welcher durch die Berechnung der Contour Likelihood bestimmt wird[15]. Im Anschluss an die Messung ergeben sich Partikelsätze, welche mehr oder weniger die Kontur der realen Hand im Bild abbilden. Im Rahmen des Tracking wird der State Vector mit maximaler Übereinstimmung für die weitere Verarbeitung – das betrifft Gestenerkennung und visuelles Feedback – verwendet.

E) Dynamische Adaption des Hautmodells

Neben der Initialisierung des Hautmodells[16] ergibt die Parametrierung die Möglichkeit der Anpassung des Hautmodells während der Laufzeit des Tracking. Entscheidend ist dies vor allem für die Robustheit in mobilen Anwendungen, da durch Bewegungen des Nutzers ein ständiger Wechsel der Beleuchtung stattfindet. Dies resultiert ohne Anpassung in einer Verschlechterung der Hautsegmentierung, welche zum Verlust des Tracking führen kann. Somit soll durch eine dynamische Anpassung der Hautparameter an die geänderten Umgebungsbedingungen eine Fehlfunktion bis hin zum Ausfall des Tracking verhindert werden.
Zwar ist der vorgestellte Initialisierungsalgortithmus prinzipiell in der Lage, eine dynamische Anpassung der Parameter vorzunehmen, doch wird jeder Pixel des Bilds betrachtet. Der erhöhte Rechenaufwand würde die Nutzbarkeit der Anwendung auf aktuellen Handheld-Geräten unterbinden. Um dennoch eine Anpassung der Parameter zur Laufzeit durchführen zu können, werden die hypothetischen Handpixel als hautfarben betrachtet.

[15] Vgl. C) Messmodell.
[16] Vgl. A) Initialisierung des Klassifikators.

Dies ergibt jedoch zwei Hauptherausforderungen:

1. Die Lage der Hand ist zu Beginn eines Durchlaufs des Algorithmus unbekannt.
2. Für die Einbettung in eine Echtzeitanwendung muss eine feste Maximallaufzeit gewährleistet sein.

Zur Anpassung des Hautmodells wird angenommen, dass sich der Startpunkt der Messlinien im Bereich der Haut und der Endpunkt im Bereich des Hintergrunds befinden. Um sicher zu gehen, dass sich Start- und Endpunkt jeweils innerhalb bzw. außerhalb der realen Kontur befinden, wird ein Schwellwert für die gemittelten Messwerte verwendet. Dies ermöglicht wiederum die Anwendung des Fitness-Werts. Die Gleichung zur Berechnung dieser Werte wird folgendermaßen angepasst:

$$fitness = \frac{n_{s,SP}}{N_M} \cdot TI - \frac{n_{s,EP}}{N_M} \tag{5.26}$$

$n_{s,SP}$ und $n_{s;EP}$ sind die Anzahl der Start- und Endpunkte der Messlinien, welche als Hautfarbe klassifiziert werden. N_M beschreibt die Anzahl der Messlinien. Die Bedeutung von TI als Target Importance bleibt unverändert. Durch die begrenzte Betrachtung von N_m Messlinien anstatt sämtlicher Pixel im Bild wird im Vergleich zur Initialisierung eine effiziente Anpassung der Parameter während der Laufzeit des Tracking ermöglicht.

F) Handmodelle für die gestenbasierte Interaktion

Entsprechend den angestrebten Formen der Interaktion im Programmiersystem werden zwei Handmodelle implementiert (s. Abb. 5.29). Kann beim „Greifmodell“ die Orientierung des Zeigefingers zur Erkennung der Geste direkt aus dem Tracking entnommen werden, wird auf die Rotation des Zeigefingers beim „Klickmodell“ über die Länge geschlossen. Die Konturen beider Modelle bestehen aus Segmenten, welche wie beschrieben als B-Spline abgebildet werden. Die Winkel zur Erkennung der Greifgeste lassen sich direkt von den jeweiligen Partitionen und deren Zustandsvektoren ableiten. Für die Klickgeste werden die Längen der Messlinien des Zeigefingers betrachtet[17]. Auf dieser Basis werden zur Gestenerkennung einfache Schwellwerte für die Winkel bzw. die Länge der Messlinien genutzt, sodass auf komplexe Algorithmen der Mustererkennung verzichtet wird. Gleichzeitig werden Grenzwerte für die Fingerstellungen betrachtet, die die Robustheit der Gestenerkennung erhöhen.
Anhand der vorgestellten Handmodelle lassen sich verschiedene Formen gestenbasierter Interaktion realisieren. Zudem wird Feedback zum aktuellen Zustand des Tracking und der Gestenerkennung gegeben. Abbildung 5.30 zeigt beispielhaft das visuelle Feedback zur Initialisierung des Tracking.

[17] Vgl. Kapitel 4.3.2.

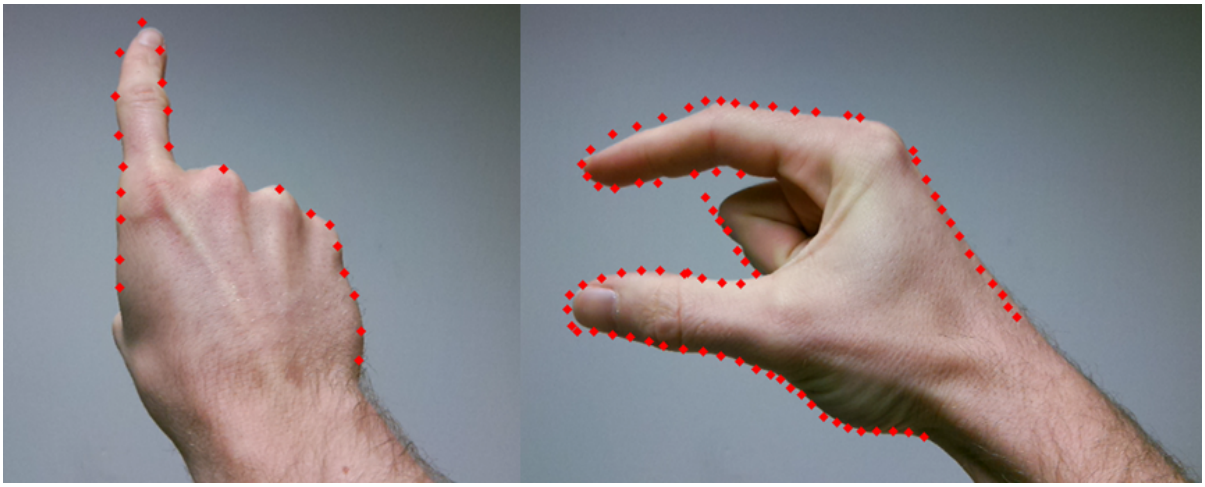

Abbildung 5.29: Handmodelle „Klicken“ (links) und „Greifen“ (rechts) mit Kontrollpunkten der Konturbeschreibung.

5.3 Umsetzung der räumlichen Interaktion

Für die Eingabe werden die erkannten Fingergesten als Auslöser (engl. *trigger*) der Interaktion verwendet, während für die eigentliche räumliche Interaktion die 3D-Koordinaten der Hand herangezogen werden. Zur Verarbeitung der Handposition muss diese vom KS des Motion-Tracking-Systems in das KS des Roboters übertragen werden. Hierzu dient die durch die Inbetriebnahme ermittelte Transformationsmatrix[18].
Zur klaren funktionalen Trennung wird das Handmodell der Klickgesten für die Definition und das Modell der Greifgesten für die Manipulation verwendet. Die Greifgeste deutet somit die Interaktion eines schon vorhandenen Objekts an, während durch die Klickgeste neue Objekte erstellt werden. Zum Starten der räumlichen Interaktion muss der Anwender per Touch-Geste im Menü der App zunächst die Art der Interaktion wählen: Definition oder Manipulation. Darauf wird das jeweilige Handmodell geladen.
Die Auswahl der zu betrachtenden Programmierebene lässt sich per Menüanwahl zu Beginn der Interaktion festlegen. Die Ebene der Programmierung kann demgemäß geändert werden, sodass zwischen einer Darstellung des Programms in Form von Posen und Trajektorien oder als Aufgabe gewählt und gewechselt werden kann. Der Interaktionsmanager (s. Abb. 5.31) beschreibt eine

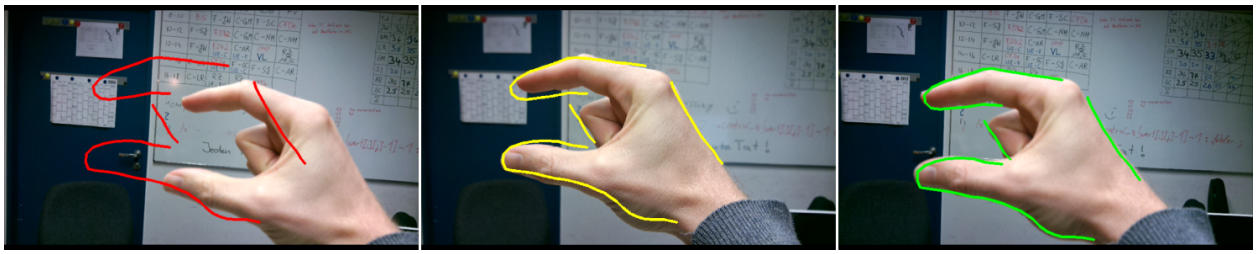

Abbildung 5.30: Initialisierungsprozess mit virtuellem Feedback: Konturvorgabe (rot), aktive Initialisierung (gelb) und erfolgreiches Tracking (grün).

[18] Vgl. Kapitel 4.3.4.

komplexe Datenklasse, welche die Verarbeitung der gestenbasierten Interaktion auf die jeweilige Programmierebene und deren virtuelle Objekte in der AR sowie deren Repräsentation im Roboterprogramm und in der Datenbank umsetzt.

5.3.1 Feedback

Die Bereitstellung des räumlichen Feedbacks erfolgt durch virtuelle 3D-Objekte auf den jeweiligen Programmierebenen. Zur Unterstützung der gestenbasierten Interaktion erfolgt eine Rückgabe über erkannte Gesten durch eine veränderte Farbdarstellung der Handkontur sowie über eine textuelle Ausgabe der erkannten Gesten (s. Abb. 5.32). Ergänzend wird eine numerische Ausgabe von Positionsinformationen bereitgestellt.

A) Erweiterte Feedbackfunktion durch haptische Rückmeldung

Eine Erweiterung der visuellen Feedbackfunktion erfolgt in der praktischen Umsetzung durch die Vibrationsfunktion der Handheld-Geräte. Diese unterstützt einerseits das visuelle Feedback zur erfolgreichen Erkennung der Fingergesten (*true-positiv*). Andererseits wird der Anwender auf nicht beabsichtigte und fälschlich ausgelöste Gesten (*false-positiv*) aufmerksam gemacht.
Weiterführend werden individuelle Vibrationsmuster für verschiedene Gesten bzw. Funktionen vorgesehen. Während für die Definition eines Objekts bspw. eine einfache Vibration von 0,4 *s* genutzt wird, wird beim Löschen von räumlichen Objekten eine dreifache Vibration mit einem kurzen Intervall von 0,2 *s* verwendet. Diese spezifische Zuweisung verschiedener Vibrationsmuster soll den Anwender dabei unterstützen, verschiedene Gesten und deren Funktionen zu unterscheiden.

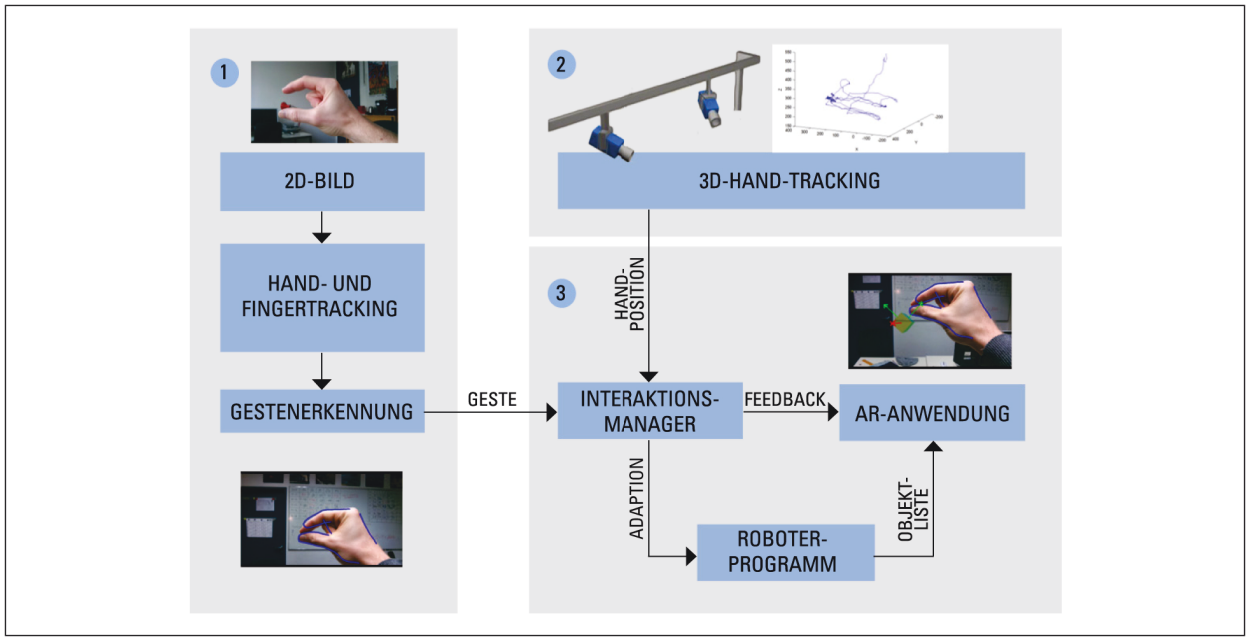

Abbildung 5.31: Informationsfluss zur gestenbasierten Interaktion: (1) Fingergestenerkennung, (2) Motion-Tracking und (3) Simulationsumgebung.

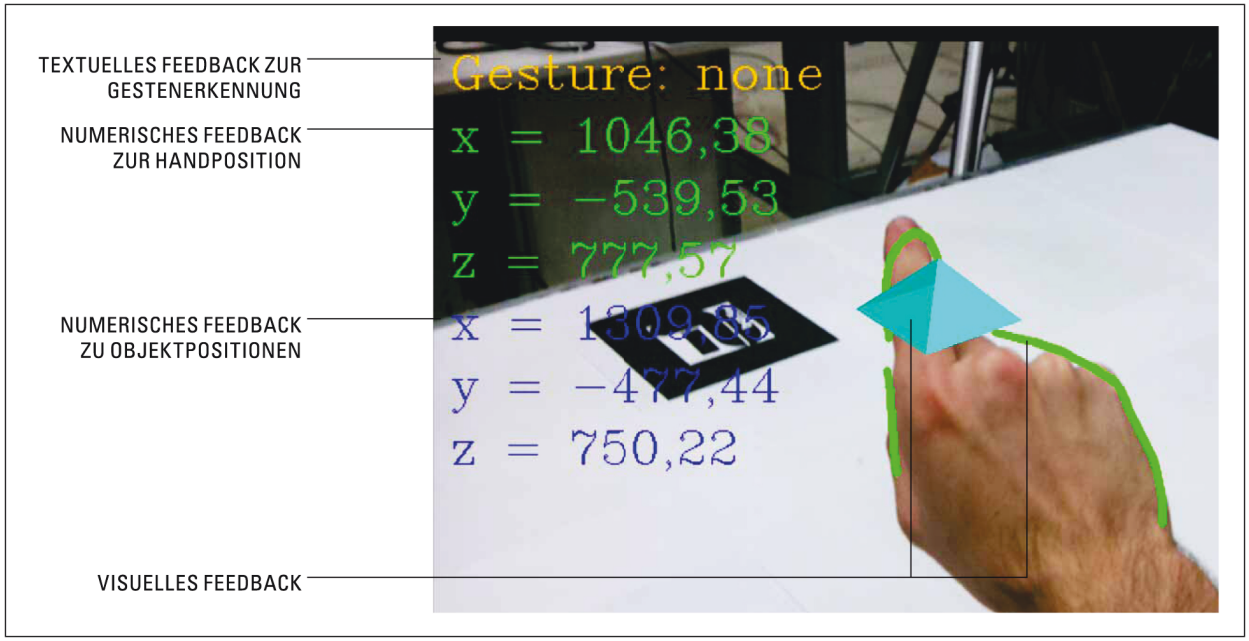

Abbildung 5.32: Feedback zur aktuellen Handposition (numerisch und visuell), zur Gestenerkennung (textuell und visuell) sowie zu potentiellen Interaktionsobjekten (hier: numerisch).

B) Feedback zur aktuellen Handposition

Eine ergänzende Funktion, welche durch den Anwender wahlweise aktiviert werden kann, bildet die Visualisierung der aktuellen 3D-Positon der Hand. Zu diesem Zweck wird in der AR-Umgebung ein virtuelles Objekt an die Handposition gezeichnet (s. Abb. 5.32). Dies ermöglicht dem Anwender einerseits eine Überprüfung des Status des Hand-Tracking sowie eine Überprüfung der Kalibrierung der KS. Andererseits wird eine Rückmeldung über die tatsächliche Lage der Handposition des Motion-Tracking gegeben, welche durch das Skeleton-Modell der Kinect nur ungenau definiert wird. Im Idealfall liegt die Position im Zentrum des Handballens. Je nach Ausrichtung des Anwenders zum Sensor kann die Handposition sich jedoch leicht verschieben. Somit kann bspw. vor der Definition einer Pose die aktuelle Position des Hand-Tracking abgeschätzt und ggf. korrigiert werden.

5.3.2 Räumliche Definition und Manipulation

A) Definition von Posen

Die Umsetzung der Definition einer neuen Pose gliedert sich prinzipiell in die beiden Teilschritte zur Definition von Position und Orientierung. Die gestenbasierte Definition der Orientierung lässt sich per Menüfunktion durch den Anwender aktivieren und deaktivieren. Bei Deaktivierung erfolgt ein beschleunigtes Anlegen von Posen mit einer einheitlich vorgegebenen Orientierung. Der Standardwert für die Orientierung muss im Vorfeld durch den Anwender per Menüeingabe festgelegt werden.

Position

Zunächst wird durch eine kurze Klickgeste ($< 1\ s$) die Position im Raum festgelegt. Dabei wird wie folgt vorgegangen:

1. Bereitstellung von visuellem und vibrotaktilem Feedback zur erfolgreichen Gestenerkennung,
2. Anlegen einer neuen Pose in der Datenbank, Festlegen der Positionsinformationen und
3. Zeichnen des virtuellen Objekts zur Repräsentation der Position in der AR.

Für das Anlegen der Position werden die aktuellen 3D-Koordinaten der Hand verwendet. An den Koordinaten wird zur Andeutung der Positionen in der AR ein virtueller Würfel (s. Abb. 5.33) gezeichnet.

Orientierung

Wird kein Standardwert für die Orientierung gewählt, schließt sich deren räumliche Definition an die Definition der Position an. Die Visualisierung der Orientierung wird über eine „Fahnendarstellung“ vorgenommen (s. Abb. 5.33). Diese Form der Visualisierung entspricht einer vereinfachten, eindeutigen Darstellung der Endeffektorrotation. Die Fahnenstange deutet die Arbeitsrichtung des Endeffektors an. Die Drehung des Endeffektors um den Vektor der Arbeitsrichtung wird durch die Ausrichtung der eigentlichen Fahne beschrieben. Zur Definition der Fahne werden zwei weitere

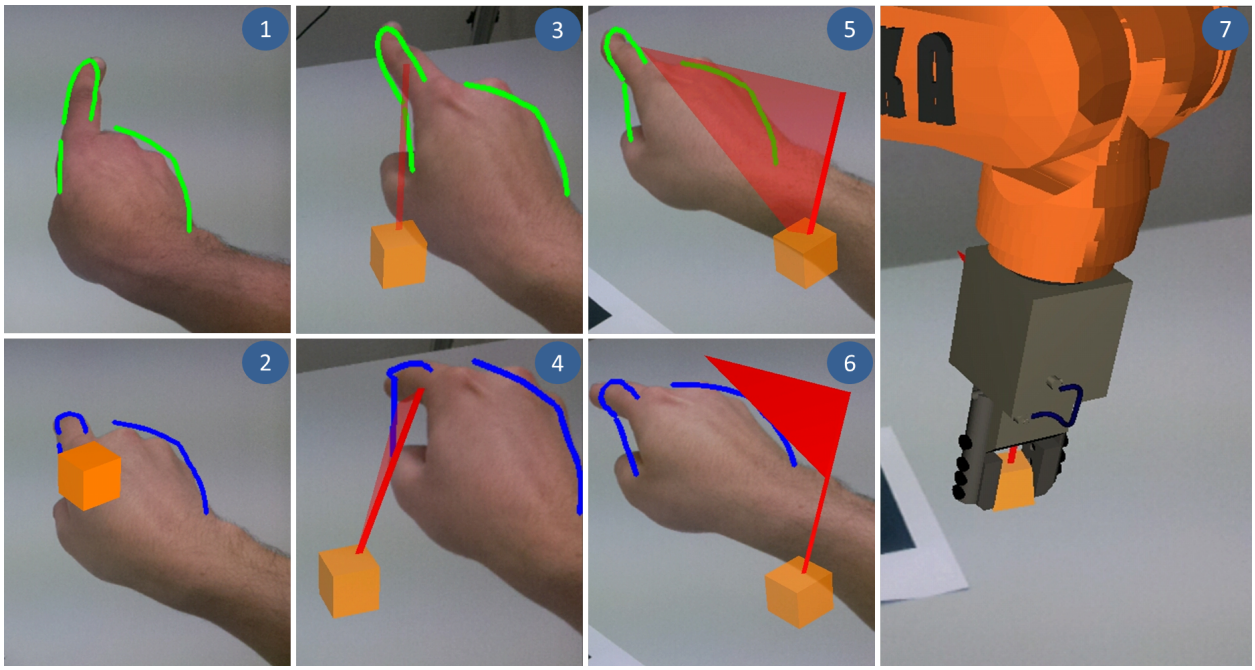

Abbildung 5.33: Sequenz zur Definition von Posen: (1) Positionierung der Hand im Arbeitsraum des Roboters, (2) Definition der Position, (3) Feedback zur potentiellen Arbeitsrichtung des TCP, (4) Definition der Arbeitsrichtung des TCP, (5) Feedback zur potentiellen Ausrichtung der zweiten Koordinatenachse des TCP, (6) Definition der zweiten Koordinatenachse und (7) Anfahren der Pose durch den virtuellen Roboter.

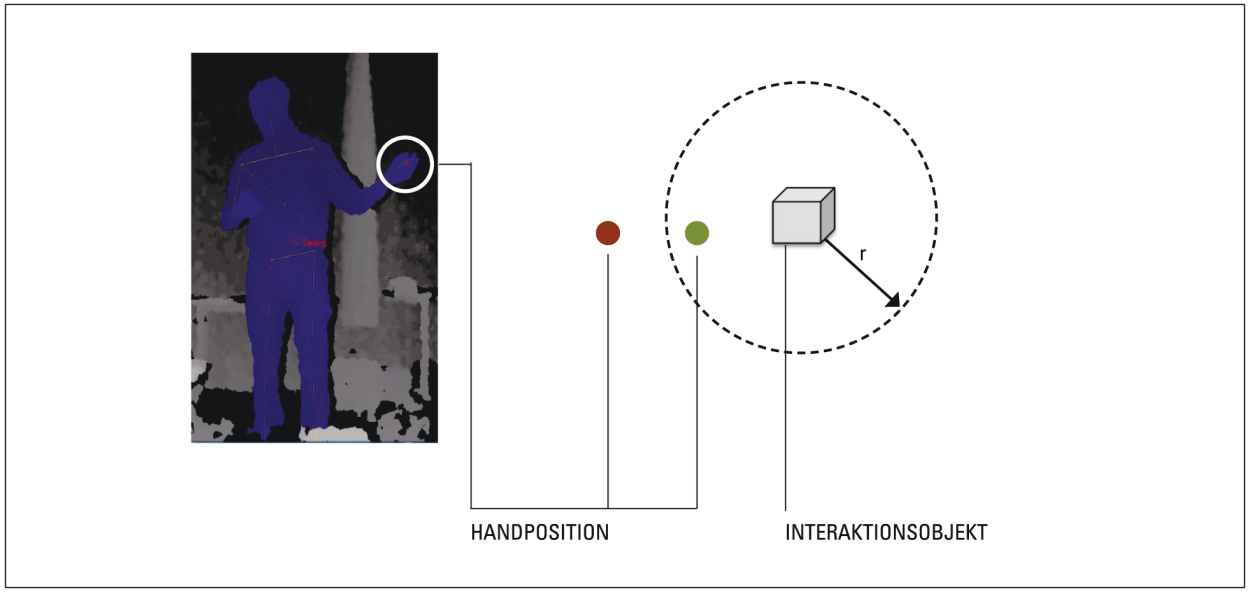

Abbildung 5.34: Berücksichtigung eines Interaktionsradius für die räumliche Manipulation von Objekten: Handposition liegt innerhalb des Radius (grün), Handposition liegt außerhalb des Radius (rot).

Positionen aufgenommen[19].
Bewegt der Anwender seine Hand im Anschluss an die Definition der Position, wird ausgehend von der angelegten Position eine Fahnenstange hin zur aktuellen Handposition angedeutet. Der Anwender wird somit in die Lage versetzt, unterstützt von simultanem Feedback die Definition der Orientierung durch Variation der Handposition intuitiv vorzunehmen. Die Arbeitsrichtung lässt sich erneut durch eine lange Klickgeste bestätigen. Die Fahnenstange wird gezeichnet. Es schließt sich die Definition des Verdrehwinkels um die Fahnenstange an. Dieser wird durch die Ausrichtung der Fahne angedeutet, welche der aktuellen Handposition in der Visualisierung räumlich folgt. Auch der Verdrehwinkel wird durch eine Klickgeste bestätigt. Nach einer erfolgreichen Erreichbarkeitskontrolle der Pose[20] wird die Rotation in der Datenbank hinzugefügt.

B) Manipulation von Posen

Translation von Posen

Die Translation der Posen gestaltet sich in direkter Anlehnung an die natürliche Translation von realen Objekten. Durch die Greifgeste wird der Verschiebungsprozess der virtuellen Würfel, welche die Position darstellen, gestartet. Mit Lösen der Greifbewegung wird die Verschiebung beendet. Als Voraussetzung zum Auslösen des Verschiebungsprozesses wird überprüft, ob sich eine Pose in unmittelbarer Nähe der Handposition befindet. Hierzu wurde ein Mindestabstand der Hand zum Objekt in Form eines Interaktionsradius definiert (s. Abb. 5.34). Befindet sich die Handposition außerhalb dieses Radius, wird bei Auslösen der Greifgeste keine Manipulation gestartet. Stattdessen wird eine entsprechende Fehlermeldung zurückgegeben. Während des Verschiebungsvorgangs erfolgt die Visualisierung des Prozesses dadurch, dass der Würfel in Andeutung der po-

[19] Vgl. Kapitel 4.2.2.
[20] Vgl. Kapitel 5.1.2.

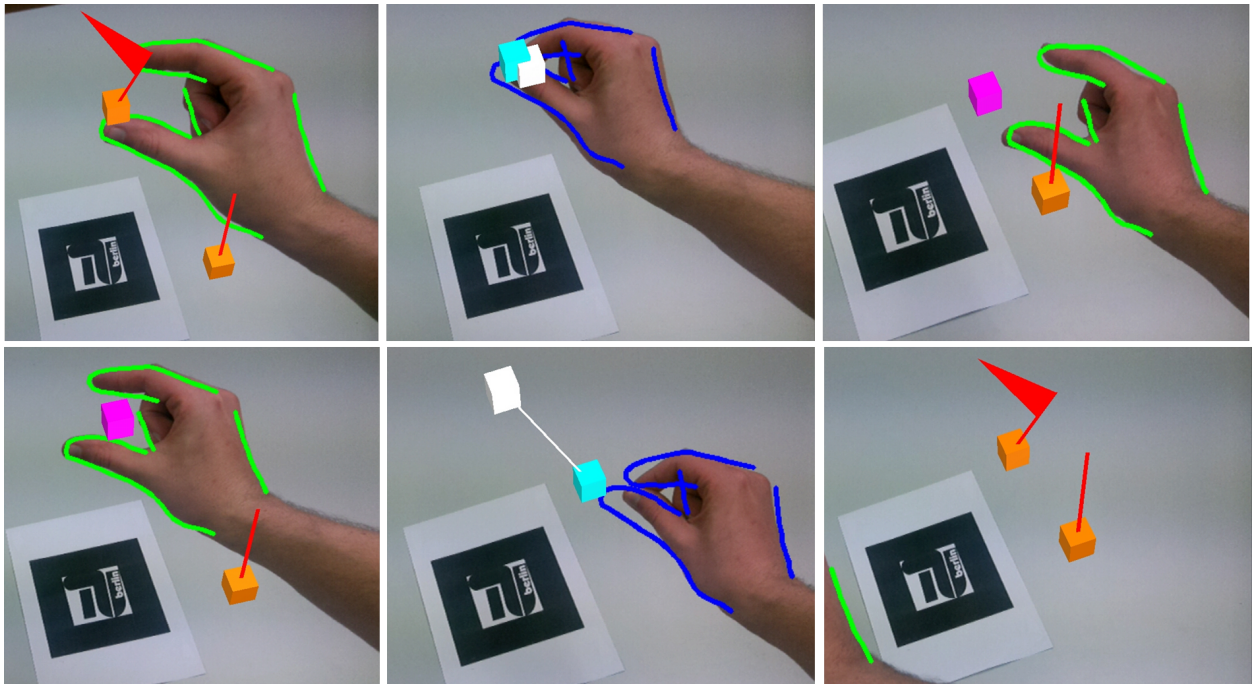

Abbildung 5.35: Räumliche Translation einer Pose über eine Greifgeste und visuelles Feedback: Näherung an Zielpose (links oben), Hand innerhalb des Interaktionsradius (links unten), Start der Translationsbewegung (Mitte oben), Translation der Pose (Mitte unten) sowie Lösen der Pose (rechts oben) und Entfernung der Hand (rechts unten).

tentiellen neuen Position der aktuellen Handposition folgt (s. Abb. 5.35). Hierzu wird zunächst ein temporäres Objekt erzeugt, während die ursprüngliche Position durch einen halbtransparenten Würfel angedeutet wird. Gleichzeitig erfolgt die Ausgabe eines numerischen Feedbacks zur translatorischen Verschiebung der Position, welche über die Differenz von Start- und Istposition der Hand ermittelt wird. Durch diese relative Verrechnung der Positionswerte wird ein anfänglicher Positionssprung bei ungenauer Greifposition verhindert. Im Anschluss an die Verschiebung wird die Position der Pose in der Datenbank aktualisiert.

Rotation von Posen

Die Manipulation der Rotation ist funktional getrennt von der Manipulation der Translation. Die Umsetzung erfolgt analog zur Definition durch eine Neudefinition über die räumliche Eingabe von zwei Positionswerten mithilfe der Fahnendarstellung. Anhand der Durchführung einer Greifgeste in der Nähe einer vorhandenen Pose lässt sich die Manipulation der Orientierung starten. Durch erneutes Durchführen der Geste werden die beiden zusätzlichen Positionen zur Definition der Orientierung aufgenommen. Die ursprüngliche Orientierung wird vor Abschluss der Manipulation als halbtransparentes Objekt visualisiert. Nach abgeschlossener Anpassung der Orientierung wird wiederum die Poseninformation in der Datenbank aktualisiert.

Entfernen von Posen

Wurde eine Pose falsch oder unbeabsichtigt im Raum angelegt, lässt sich diese intuitiv über eine räumliche Geste entfernen. Zu diesem Zweck muss sich die Hand des Anwenders in unmittelbarer Nähe der zu entfernenden Pose befinden. Im Anschluss an die Ausführung der Geste erfolgt visu-

elles und vibrotaktiles Feedback. Die Darstellung der Pose wird aus der AR-Darstellung und der Datenbank entfernt.

C) Definition von Trajektorien

Die räumliche Erzeugung von Bewegungsbahnen verknüpft die Definition von Posen mit Interpolationsbefehlen und bildet auf dieser Basis einen Bewegungsbefehl für das Roboterprogramm. Im Dialog zur bewegungsorientierten Programmierumgebung wählt der Anwender zunächst die gewünschte Interpolationsart. Durch die Klickgeste werden Posen angelegt, welche automatisch mit der gewählten Interpolation verknüpft werden. Alternativ können bereits vorhandene Posen über die Menüführung mit Interpolationsbefehlen verknüpft werden. Je nach Interpolationsart sind für einen Bewegungsbefehl eine oder mehrere Posen zu definieren. Die Definition von PTP- und Linearbewegungen erfordert lediglich die Parametrierung über eine einzelne Pose. Eine Definition der Zirkularbewegung erfolgt anhand der aufeinanderfolgenden Posendefinitionen von Hilfs- und Zielpose. Die räumliche Definition komplexer Freiformkurven erfolgt in einer gesonderten Form über eine Aufnahme von Handpositionen in diskreten räumlichen Abständen. Start und Ende der Freiformkurve werden durch eine Klickgeste gesteuert.

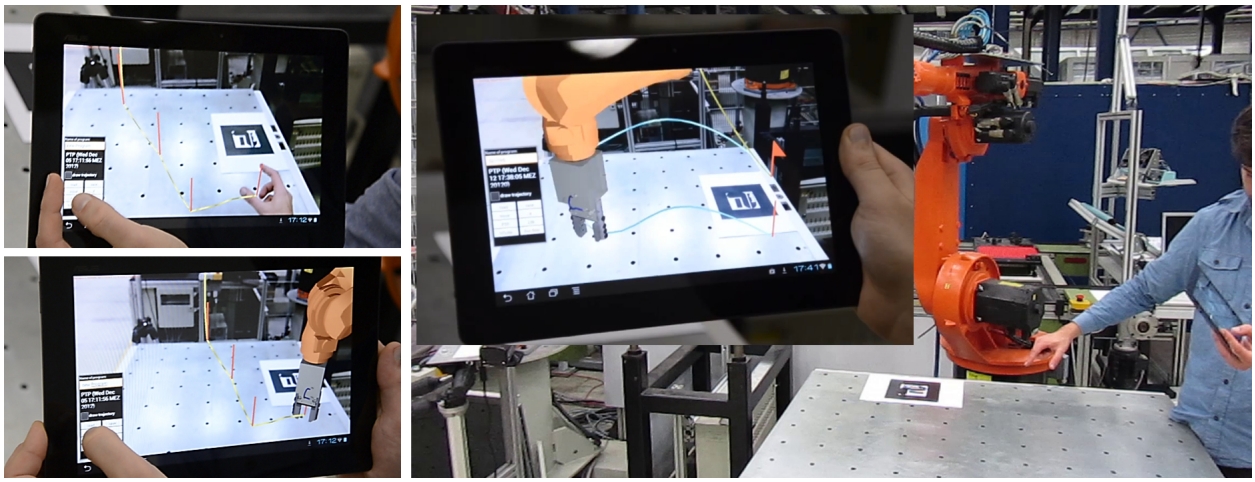

Abbildung 5.36: Gestenbasierte Definition einzelner PTP-Bewegungen (links oben), Simulation des Roboterprogramms (links unten) sowie Definition und Simulation einer Spline-Bewegung (rechts).

Im Anschluss an die Definition eines Bewegungsbefehls erfolgt die Visualisierung der resultierenden Bewegungsbahn. Dies wird auf Basis der implementierten Algorithmen zur Interpolation vorgenommen[21]. Zum Zeichnen werden die Bewegungsbefehle des Programms sequenziell durchlaufen. Ein Bewegungsabschnitt lässt sich jeweils aus der letzten Pose des vorangegangenen Bewegungsbefehls als Startpose und den Bahnposen des aktuellen Bewegungsbefehls definieren. Eine bessere Unterscheidbarkeit erhalten die verschiedenen Interpolationsarten durch eine Visualisierung mittels individueller Farben. Abbildung 5.36 veranschaulicht das Anlegen von Trajektorien und die Visualisierung und Simulation des Roboterprogramms.

[21] Vgl. Kapitel 5.1.2.

Die Parametrierung der Bewegungsbefehle hinsichtlich Geschwindigkeiten, Beschleunigungen und Achskonfigurationen wird zunächst gemäß Standardvorgaben gesetzt und kann bei Bedarf durch den Anwender über das Touch-Display angepasst werden.

D) Manipulation von Trajektorien

Die räumliche Manipulation der Bewegungsbahnen bezieht sich auf die Posen einzelner Bewegungsbefehle. Die Betrachtung soll dabei im Rahmen dieser Arbeit auf die Translation beschränkt werden. Bezüglich der Translation kann zwischen dem Verschieben eines gesamten Bahnabschnittes, also dem gleichzeitigen Verschieben mehrerer Posen, oder dem Verschieben einzelner Posen unterschieden werden. Beide Funktionen wurden umgesetzt und können durch den Anwender jeweils durch das Ausführen einer langen bzw. einer kurzen Greifgeste gesteuert werden. Bei der Manipulation einzelner Bahnsegmente über eine lange Greifgeste muss sich die Hand analog zur Manipulation von Posen in der Nähe eines Bahnabschnitts befinden. Der betrachtete Interaktionsradius bezieht sich auf das geometrische Mittel der Bahnpositionen des jeweiligen Bahnbefehls. Verschoben werden darauf sämtliche Bahnposen des Befehls. Zusätzlich wird die Startposition in Form der Zielposition des vorangegangenen Bewegungsbefehls mit einbezogen. Während der Interaktion werden simultan ein Feedback zum potentiellen und eine halbtransparente Darstellung des ursprünglichen Bahnverlaufs bereitgestellt.

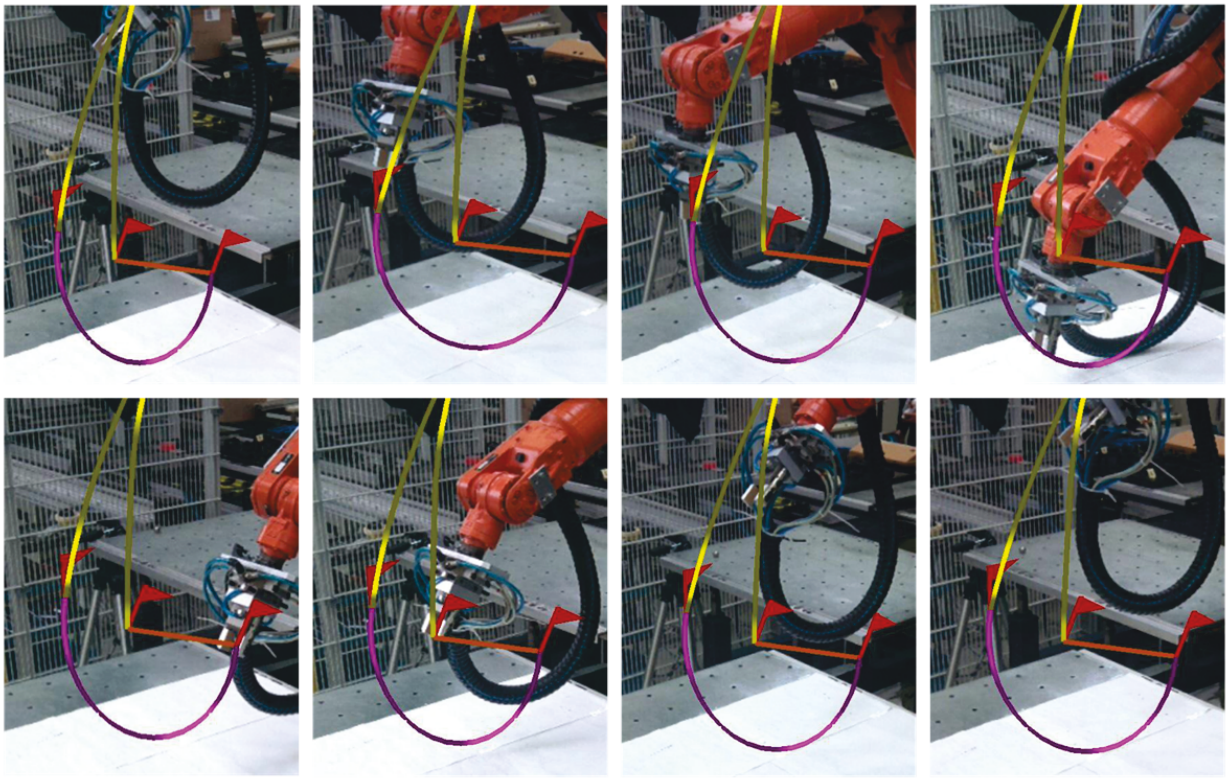

Abbildung 5.37: Bewegungssequenz zur Übertragung und Ausführung eines in der AR visualisierten Roboterprogramms auf die reale Robotersteuerung: PTP-Bewegung (gelb), Linearbewegung (orange) und Zirkularbewegung (lila).

5.4 Umsetzung der Schnittstellen und der Inbetriebnahme

5.4.1 Vereinheitlichte Roboterschnittstelle

Die Roboterschnittstelle wurde gemäß der Definition[22] in der App implementiert. Die Schnittstelle unterstützt bilaterale Kommunikation. Abbildung 5.37 veranschaulicht in einer Bewegungssequenz des realen Roboters die Ausführung eines in der AR-visualisierten Roboterprogramms. Aufseiten der Robotersteuerung wurde ein entsprechendes Interpreterprogramm für die Programmiersprache KUKA Robot Language (KRL) umgesetzt. Zusätzlich wurde die Schnittstelle in eine eigenentwickelte Steuerung für einen modularen Roboter der Fa. Amtec Robotics integriert. Da die für die Umsetzung verwendete KUKA-Steuerung KRC1 weder über eine drahtlose noch über eine Ethernet-Schnittstelle verfügt, wurde über einen zusätzlichen Industrie-PC die Weiterleitung der Kommunikation über die RS-232-Schnittstelle vorgenommen. Abbildung 5.38 beschreibt den Ablauf der Kommunikation am vorliegenden Beispiel.

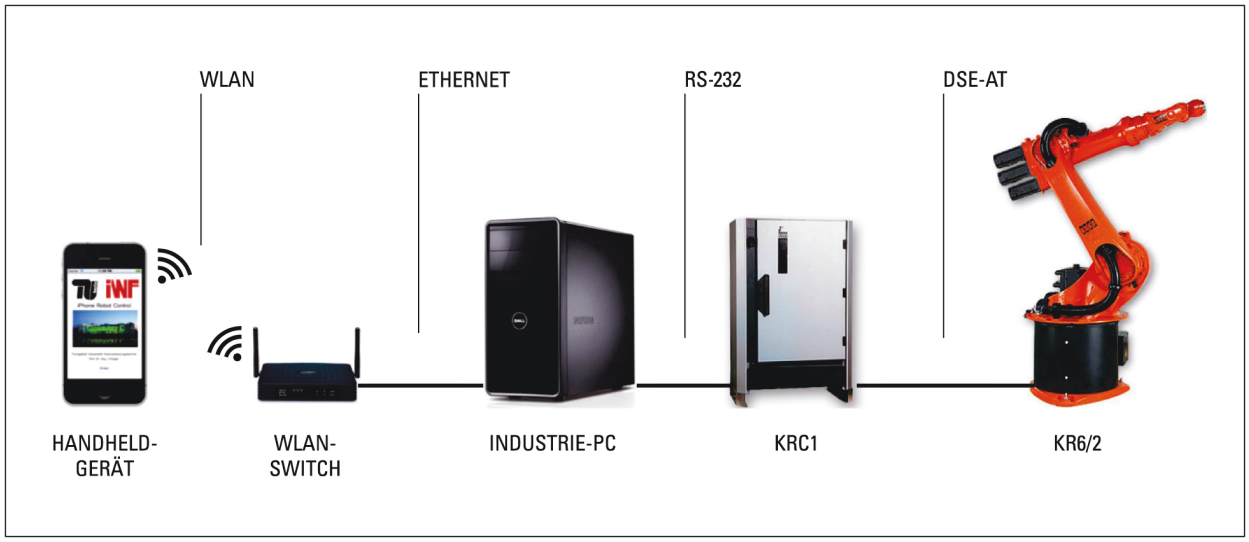

Abbildung 5.38: Veranschaulichung der Umsetzung der Kommunikationsstruktur der Roboterschnittstelle für die Steuerung KUKA KRC1.

5.4.2 Schnittstelle zur Digitalen Fabrik

Zur Anbindung der App an die Digitale Fabrik wird eine Schnittstelle zum Austausch von Roboterprogrammen zwischen der App und Siemens Tecnomatix erstellt. Genutzt wird das Erweiterungsmodul Robot-Controller-Simulation (RCS), welches eine Ein-und Ausgabe von herstellerspezifischen Roboterprogrammen bereitstellt. Bei der Umsetzung wird wiederum KUKA als Hersteller mit der Programmiersprache KRL betrachtet. Da das RCS-Modul über keine geeignete Programmierschnittstelle (API) zur Integration der Kommunikation mit der App verfügt, wird diese aufseiten von Tecnomatix als Zusatzanwendung umgesetzt, welche dementsprechend Zugriff

[22] Vgl. Kapitel 4.3.3.

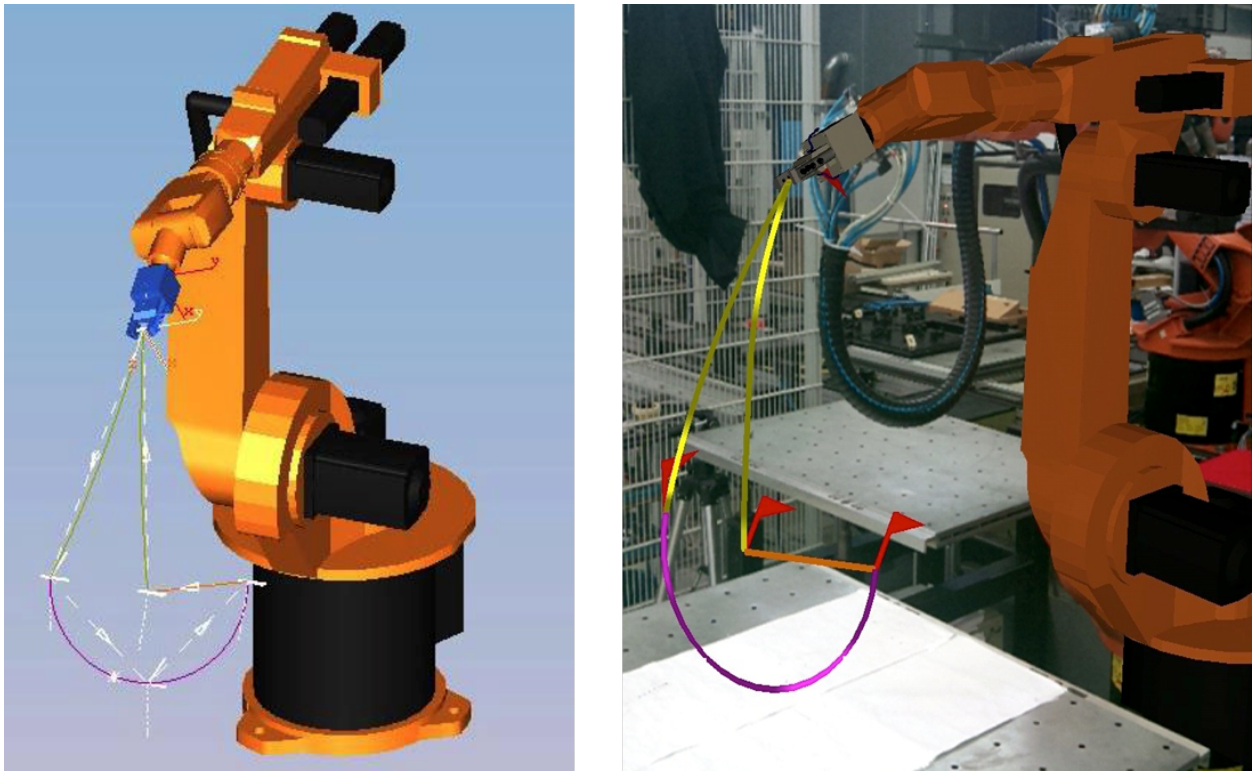

Abbildung 5.39: Darstellung eines übertragenen Roboterprogramms zwischen Tecnomatix (links) und AR-Simulation (rechts): PTP-Bewegung (gelb), Linearbewegung (orange) und Zirkularbewegung (lila).

auf die Export- und Importordner des RCS-Moduls erhält. Eine Übertragbarkeit des Schnittstellenprogramms auf andere Offline-Programmiersysteme ist somit gegeben. Abbildung 5.39 visualisiert ein übertragenes Bewegungsprogramm in Tecnomatix und in der AR-Darstellung der App.

A) Kommunikation

Die Kommunikation erfolgt ebenfalls bidirektional über das Übertragungsprotokoll TCP/IP. Dieses kann plattformunabhängig verwendet werden und ermöglicht eine Kommunikation über lokale Netzwerke sowie über das Internet. Die TCP/IP Verbindung wird als Server-Client-Konzept umgesetzt, wobei die App die Rolle des Clients erhält. Die zentrale Anwendung der Digitalen Fabrik fungiert als Server und ist somit prinzipiell befähigt, Daten von mehreren und für mehrere Clients zu verarbeiten. Abbildung 5.40 veranschaulicht den Datenfluss der Kommunikation zwischen Tecnomatix sowie mehreren App-Clients und realen Robotersteuerungen.

B) Sende- und Empfangssteuerung

Zum Senden und Empfangen von Roboterprogrammen werden sowohl die App als auch die Zusatzanwendung zum RCS-Modul mit Push/Pull-Funktionen ausgestattet, die für einzelne Programme sowie für sämtliche Programmdaten durchgeführt werden können. In der App werden die entsprechenden Funktionen in den Dialog zur Programmverwaltung integriert. Derart kann bspw. auf Anforderung eine Übertragung sämtlicher offline generierter Programme an die App zur realitäts-

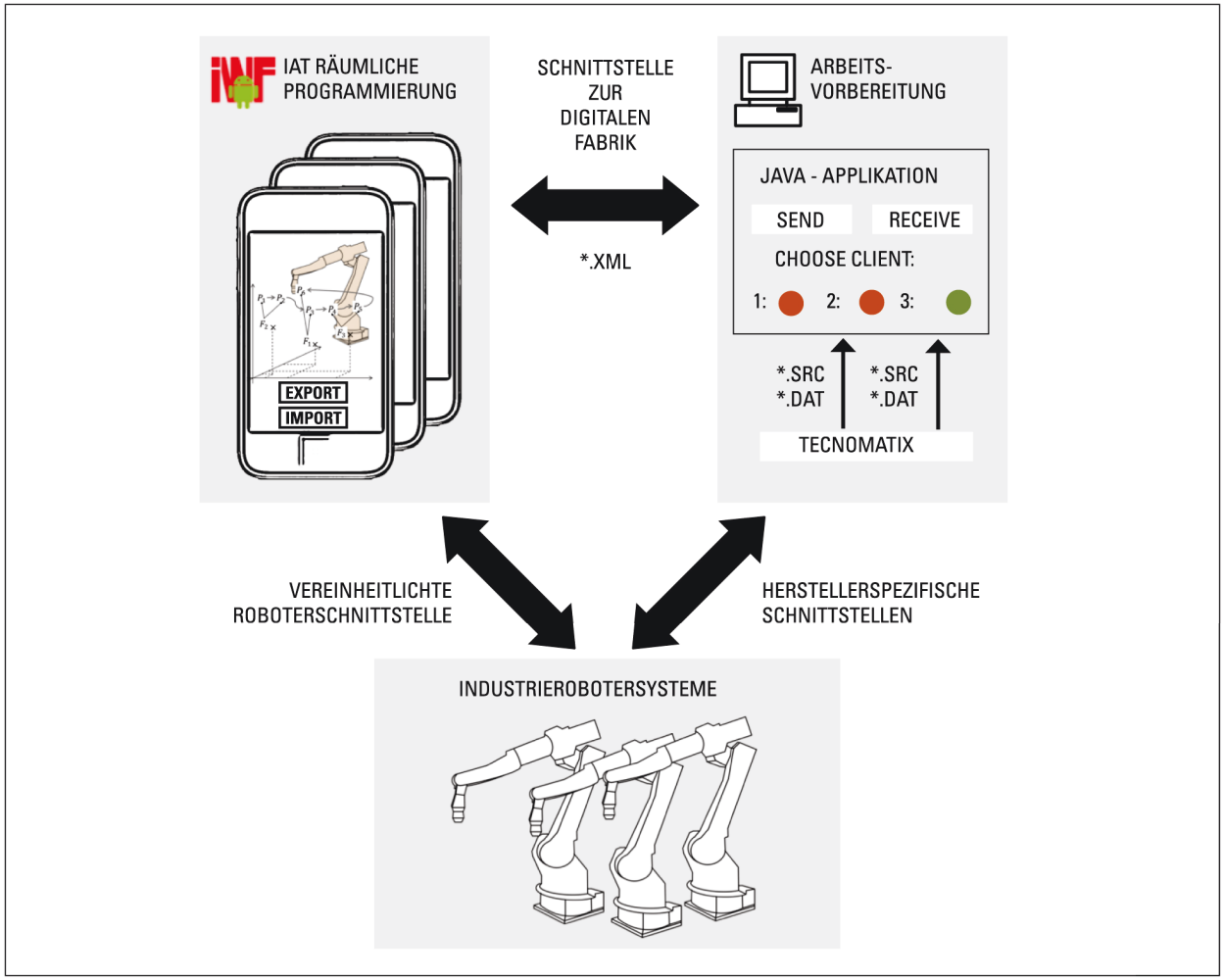

Abbildung 5.40: Datenfluss der Kommunikation zwischen Robotersteuerungen, Digitaler Fabrik (Tecnomatix) und mehreren mobilen Apps.

nahen Evaluation erfolgen. Weiterführend können auch gestenbasiert erstellte Programme aus der App an den Server zur Archivierung oder weiterführenden Bearbeitung im Sinne eines hybriden oder kooperativen Programmieransatzes übertragen werden.

C) Verarbeitung der Programmdaten

Die Übersendung der Programme erfolgt in einer neutralen Darstellung, gemäß der Repräsentation der Programme in der App. Da jedoch eine Übersendung von Datenbankobjekten wenig praktikabel ist, erfolgt eine Übertragung der Roboterprogramme in das Datenformat der Extensible Markup Language (XML). Der XML-Standard[23] ermöglicht ein schnelles Versenden, Auslesen und Weiterverarbeiten der Programminhalte. Umgesetzt wird dies unter anderem durch Serialisierung (engl. *serialisation*), welche das einfache Hinzufügen von Daten zum Versand ermöglicht. Zudem ist das XML-Format für den Anwender leicht zu lesen und zu interpretieren.

[23] Vgl. http://www.w3.org/XML/ - 20.01.2014.

Postprozessor/Präprozessor
Da die Roboterprogramme aufseiten des RCS-Moduls in herstellerspezifischem Code vorliegen, wird zum Versenden bzw. Empfangen ein Prä- bzw. Postprozessor genutzt. Dieser gestaltet das Programm für den jeweiligen Empfänger lesbar (s. Abb. 5.41). Neben der Übersetzung der einzelnen Programmkommandos werden Winkelkonventionen und Darstellungen der Achskonfigurationen angepasst. Im Fall von KRL werden die ABC-Winkelkonvention und eine Definition der Konfiguration anhand der Parameter *s* und *t* berücksichtigt. Aufseiten der App beschränkt sich die Verarbeitung auf eine Serialisation bzw. Deserialisation der XML-Dokumente und einen Abgleich mit den Datenbanken. Die Betrachtung möglicher Programmierkommandos beschränkt sich für die prototypische Umsetzung auf die Bewegungsbefehle und Anweisungen, welche aktuell durch die App abgebildet werden. Eine Erweiterung der Schnittstelle lässt sich mit geringem Aufwand vornehmen.

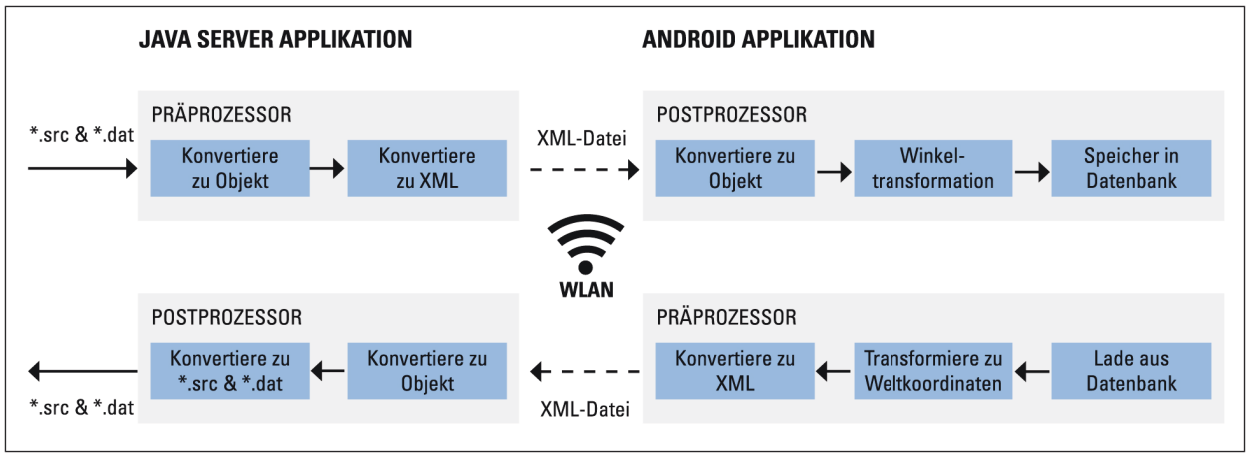

Abbildung 5.41: Post- und Präprozessoren zum Austausch von Roboterprogrammen zwischen App und Tecnomatix.

5.4.3 Inbetriebnahme

Zur Ermittlung der Transformationsmatrizen zwischen den Koordinatensystemen des Roboters, des Motion-Tracking-Systems sowie des AR-Markers werden korrespondierende Positionen aufgenommen. Diese dienen zur Berechnung der Transformationsmatrizen[24].
Zur Aufnahme der Referenzpunkte im Roboterkoordinatensystem werden definierte Positionen an einem Muster (s. Abb. 5.42) bzw. am AR-Marker mithilfe einer Messspitze manuell durch das Handbediengerät des Roboters angefahren und aufgenommen. Als Referenzpunkte des Markers werden die charakteristischen Eckpunkte gewählt. Deren Koordinaten sind im Marker-KS bereits bekannt. Die Positionskorrespondenzen im KS des Motion-Tracking-Systems werden über eine spezielle Anwendung durch manuelles Anwählen der Raumpunkte im Bildausschnitt des Kamerasystems ermittelt. Die aufgenommenen Positionen des Motion-Tracking-KS unterliegen demnach in der prototypischen Umsetzung der Absolutgenauigkeit des Kinect-Sensors. Zur Ermittlung der

[24] Vgl. Kapitel 4.3.4.

optimalen Transformationsmatrix zwischen den Positionskorrespondenzen dient das aufgezeigte Vorgehen[25] in Kombination mit einer Singulärwertzerlegung.

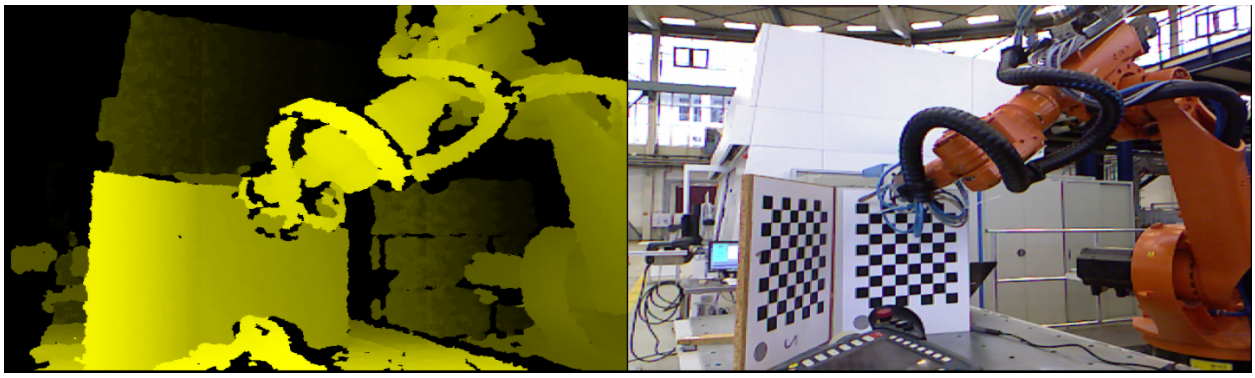

Abbildung 5.42: Tiefenbild des Kinect-Sensors zur Aufnahme von Referenzpositionen an einem Schachbrettmuster (links) und manuelles Anfahren der Positionskorrespondenzen über das Handbediengerät des Roboters (rechts).

[25] Vgl. Kapitel 4.3.4.

6 Erprobung und Evaluation des natürlich-räumlichen Programmiersystems

In diesem Kapitel wird in Abschnitt 6.1 zunächst der allgemeine Aufbau zur Durchführung der Erprobungen und Evaluation thematisiert. Des Weiteren wird in Abschnitt 6.2 die Erprobung der aufgabenorientierten Programmierung am Beispiel einer Pick-and-Place-Aufgabe beschrieben. Anschließend werden in Abschnitt 6.3 die Durchführung und die Evaluation einer Nutzerstudie zur vergleichenden Evaluation der natürlich-räumlichen Programmierung aufgeführt. In Abschnitt 6.4 erfolgt eine Technologiebewertung der prototypischen Umsetzung. Abschnitt 6.5 nimmt abschließend eine zusammenfassende Bewertung des Programmiersystems anhand einer Beantwortung der aufgeworfenen Forschungsfragen vor.

6.1 Aufbau des Programmiersystems

Zur Erprobung und Evaluation des Programmiersystems werden zwei Android-Geräte eingesetzt: das Smartphone Samsung Galaxy S II und der Tablet-PC Asus Transformer Prime TF 201. Simulation und Gestenerkennung basieren jeweils auf den nativen Displayauflösungen von 480×800 bzw. 800×1280 Pixeln. Zusammen mit dem Kinect-Sensor und einem WLAN-Switch liegen die kompletten Hardware-Kosten des Systems bei ca. 600 €. Für die Erprobung des Programmiersystems wird ein KUKA KR6 mit KRC1-Steuerung verwendet. Der Aufbau zur Erprobung wird in Abbildung 6.1 veranschaulicht. Die Nutzung der Roboterschnittstelle erfolgt mittels der beschriebenen Umsetzung[1].

6.2 Erprobung der aufgabenorientierten Programmierung an einem Pick-and-Place-Szenario

Zur Erprobung der aufgabenorientierten Programmierung wird die Applikation der natürlich-räumlichen Programmierung auf eine Pick-and-Place-Aufgabe betrachtet. Deren Wahl erfolgt aus Gründen der Simplizität und Anschaulichkeit im Hinblick auf eine Übertragung auf weitere Anwendungsbereiche. Das methodische Vorgehen zur Definition der Umsetzbewegung durch Vormachen orientiert sich an [183]. In dieser Veröffentlichung wurden Pick-and-Place-Aufgaben durch

[1] Vgl. Kapitel 5.4.1.

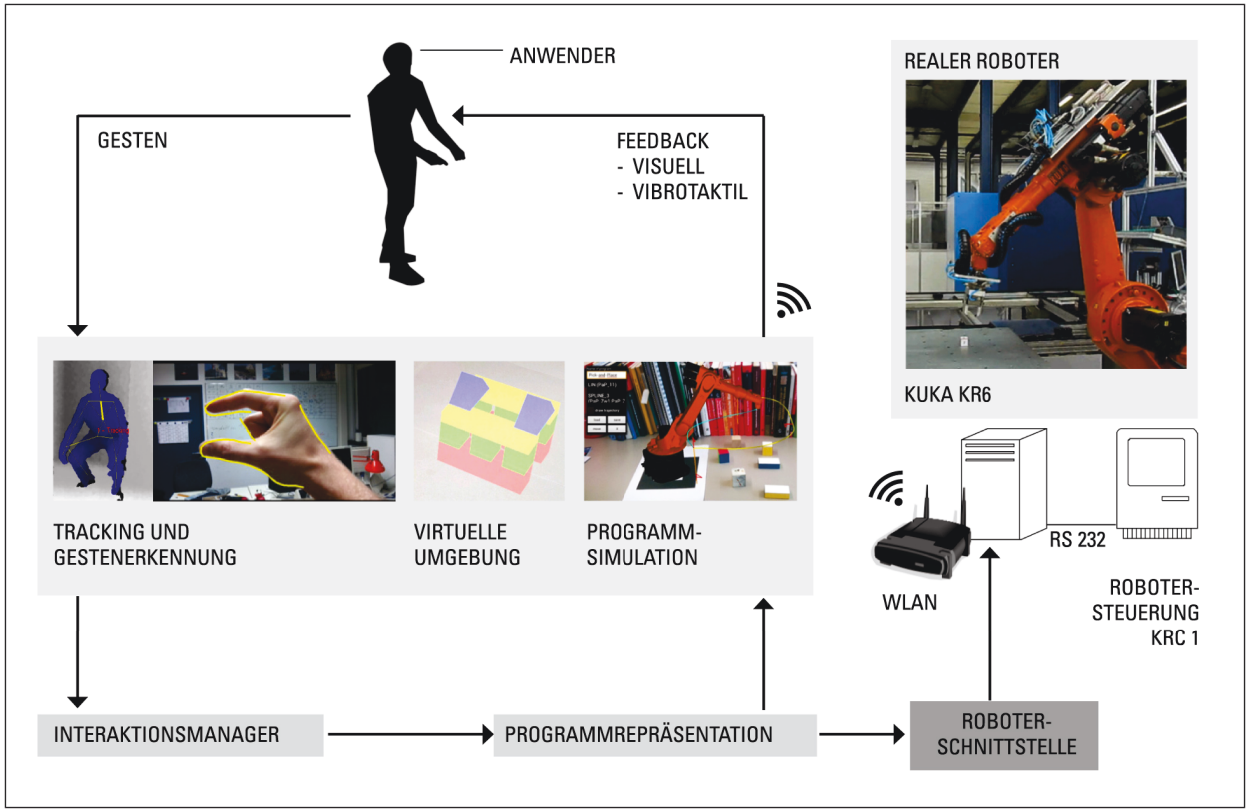

Abbildung 6.1: Prototypischer Aufbau des räumlichen Programmiersystems zur Erprobung und Evaluation.

das gestenbasierte Verschieben von virtuellen Objekten in einer Simulationsumgebung definiert. Zum Verschieben interagierte der Nutzer über einen Datenhandschuh mit virtuellen Objektrepräsentationen in der VR-Umgebung. Aus der real ausgeführten Bewegung erfolgte eine Ableitung von Programmen für einen simulierten Roboter.
Im Kontext der räumlichen Programmierung soll dieses Interaktionsprinzip auf eine markerlose AR-basierte Interaktion im Arbeitsbereich des Industrieroboters übertragen werden. Dieser Ansatz wurde vom Autor der vorliegenden Arbeit in [157] vorgestellt. In der folgenden Gestaltung der Pick-and-Place-Aufgabe wird davon ausgegangen, dass die Verteilung der Interaktionsobjekte bezüglich Typ und Pose in der Initialszene als bekannt vorausgesetzt werden kann. Ein weiterführender Ansatz zur automatischen Detektion und Posenschätzung geometrischer Primitive über die Handheld-Kamera wurde in [157] vorgestellt, soll an dieser Stelle jedoch nicht näher vertieft werden. Abbildung 6.2 zeigt die allgemeine Prozesskette der aufgabenorientierten räumlichen Programmierung.

6.2.1 Gestaltung des Demonstrationsprozesses

Die Gestaltung der Manipulation der virtuellen Objekte orientiert sich an der räumlichen Manipulation der Posen in der bewegungsorientierten Programmierebene[2]. Die räumliche Anordnung eines Werkstücks lässt sich ebenfalls über Position und Orientierung beschreiben. Demgemäß findet

[2]Vgl. Kapitel 5.1.1.

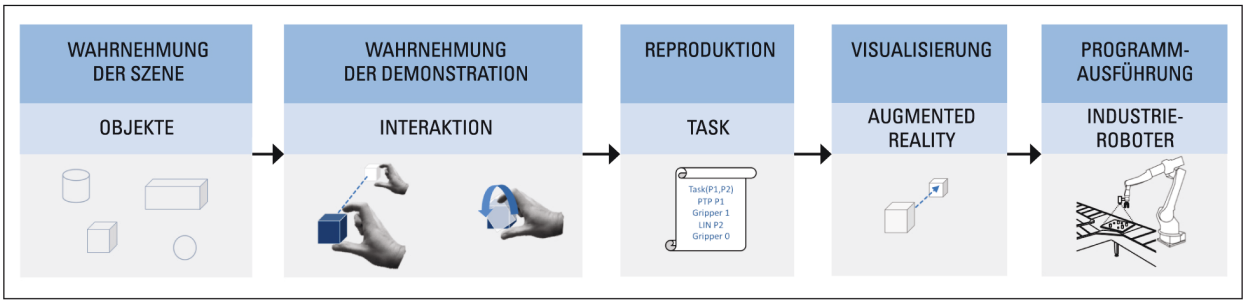

Abbildung 6.2: Prozesskette der aufgabenorientierten Programmierung durch Vormachen.

zur Definition der Umsetzbewegung eine virtuelle Verschiebung der einzelnen virtuellen Werkstücke über die Hand an eine Zielpose statt. Während der translatorischen Umsetzbewegung folgt das virtuelle Werkstück der Handbewegung, sodass der Anwender simultan ein Feedback zur erzielten Objektposition erhält. Die Veränderung der Rotation erfolgt analog zur bewegungsorientierten Ebene über die Fahnendarstellung. Ergänzend wird eine simultane Darstellung der potentiellen Objektpose für die aktuelle Rotation bereitgestellt. Zur Unterstützung der räumlichen Manipulation und zum Ausgleich von Ungenauigkeiten des Sensorsystems wurde eine sogenannte „Fangfunktion“ implementiert. Diese ermöglicht die automatische Ausrichtung des aktuellen Interaktionsobjekts an einem Referenzobjekt, sobald sich diese innerhalb eines definierten Abstands zueinander befinden.

6.2.2 Reproduktion der Aufgabe

Zur Reproduktion eines Roboterprogramms aus der beobachteten Demonstration wird zunächst von einem Experten eine bewegungsorientierte Definition des Programmmakros vorgenommen. Dies erfolgt anhand der aufgabenorientierten Programmierebene[3]. Abbildung 6.3 beschreibt beispielhaft das erstellte Programmmakro für Pick-and-Place-Aufgaben, welches über die zwei Parameter der Greif- und Ablagepose zu parametrieren ist. Diese Posenparameter lassen sich bei bekannten Werkstücken aus der Objektpose ableiten. Zur Durchführung der Programmierung werden beide Posenparameter automatisch aus dem Demonstrationsvorgang extrahiert und in das aufgabenorientierte Programmmakro übertragen. Dies erfolgt in Vorbereitung der praktischen Umsetzung über eine Verknüpfung der Programmparameter mit einzelnen Fingergesten.

6.2.3 Visualisierung der Aufgabe

Die Visualisierung der Pick-and-Place-Aufgabe erfolgt analog zur Definition des Vorgehens zur Reproduktion anwendungsspezifisch. Das bedeutet, dass das visuelle Feedback auf Aufgabenebene in der Regel nicht als übertragbar gelten kann. Dies resultiert aus der Motivation, auch dem fachfremden Nutzer ein adäquates Feedback der Aufgabe zu geben, welches anschaulich ist und somit ermöglicht, eventuelle Fehler im Roboterprogramm zu erkennen und gegebenenfalls zu kor-

[3] Vgl. Kapitel 5.1.1.

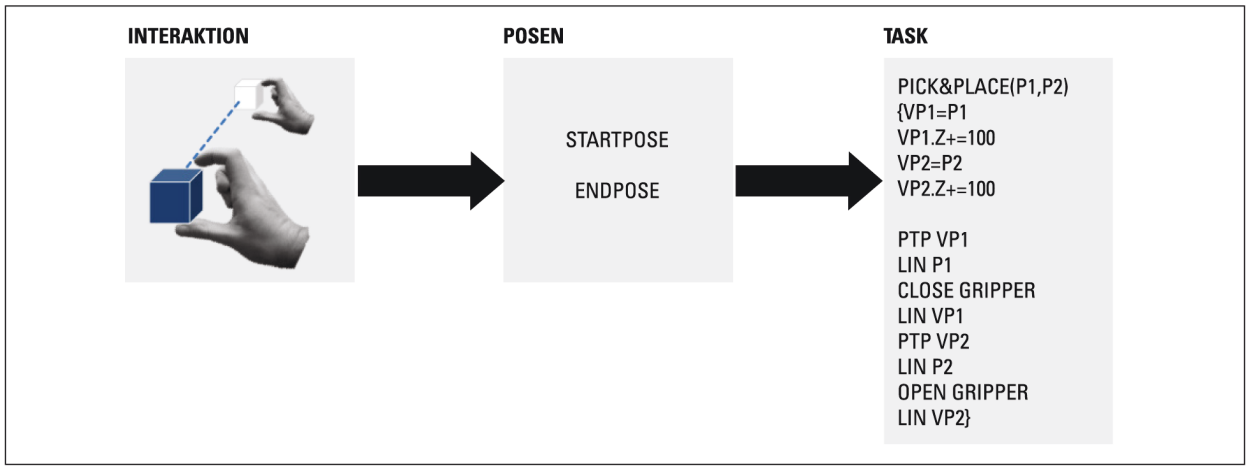

Abbildung 6.3: Verknüpfung von Fingergesten und Posenparametern zur automatischen Parametrierung von Pick-and-Place-Aufgaben (bewegungsorientierte Darstellung).

rigieren. Ergänzend zur Darstellung der virtuellen Objekte in ihrer beabsichtigten Ablagepose erfolgt nach Beendigung der Demonstrationsbewegung wahlweise eine zusätzliche Pfeildarstellung von ursprünglicher Pose zur Zielpose der Objekte. Derart lassen sich nach der Aufgabendefinition reales Objekt und angestrebte Ablagepose einander zuordnen. Gegebenenfalls kann eine erneute Manipulation der Zielpose vorgenommen werden. Bei Bedarf kann ein Wechsel zur bewegungsorientierten Darstellung der Aufgabe erfolgen.

6.2.4 Interaktionsszenario

Zur Erprobung der natürlich-räumlichen Programmierung der Pick-and-Place-Aufgabe wird eine initiale Tischbelegung mit realen Werkstücken betrachtet. Die Objektposen der Werkstücke sowie deren CAD-Daten sind bekannt, sodass in der AR zunächst die virtuellen Objektrepräsentationen die realen Objekte überlagern (s. Abb. 6.4, links oben). Der Anwender baut sich nun durch Translation und Rotation der virtuellen Objekte eine virtuelle Zielszene auf (s. Abb. 6.4, Mitte oben). Bei Übertragung und Ausführung des generierten Roboterprogramms auf die reale Steuerung erfolgt eine Umsetzung der einzelnen Objekte sequentiell gemäß dem angelegten Programmmakro (s. Abb. 6.4, links unten und Mitte unten), sodass nach Beendigung der Ausführung virtuelle Zielszene und reale Szene übereinstimmen (s. Abb. 6.4, rechts).

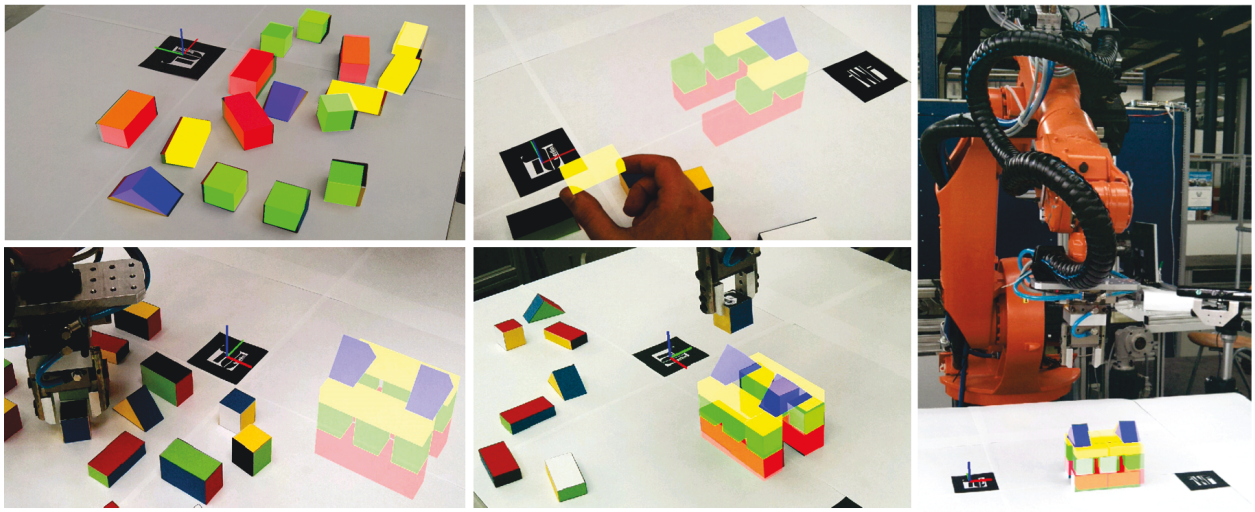

Abbildung 6.4: Visualisierung der Initialszene (links oben), des Translationsprozesses einzelner Werkstücke (Mitte oben), der Ausführung der Aufgabe durch den realen Roboter (links unten und Mitte unten) sowie abgeschlossene Durchführung der Aufgabe (rechts).

6.3 Evaluation des Programmiersystems anhand einer Nutzerstudie

6.3.1 Beschreibung der Versuchsdurchführungen

Die Versuchshypothese zur Evaluierung der räumlichen Programmierung orientiert sich an den Leitkriterien zur Nutzerfreundlichkeit aus DIN EN ISO 9241, Teil 11[4]: Die natürlich-räumliche Programmierung ist den Standardverfahren der Roboterprogrammierung in Effizienz, Effektivität und Nutzerzufriedenheit überlegen.
Als Ergebnis der Auswertung der Nutzerstudie lassen sich demgemäß Aussagen zur Überprüfung der zentralen Forschungshypothese der vorliegenden Arbeit ableiten. Zu diesem Zweck sollen im Rahmen der Versuchsdurchführungen die Messgrößen der Programmierdauer, der Programmierfehler sowie der subjektiven Zufriedenheit der Nutzer anhand von Beispielszenarien der Programmierung ermittelt werden. Neben der räumlichen Programmierung werden vergleichend das Teach-In und die simulationsgestützte Offline-Programmierung betrachtet.

A) Programmierszenarien

Die Studie umfasst die Durchführung von zwei verschiedenen Programmieraufgaben. Das erste Programmierszenario beinhaltet eine bewegungsorientierte und das zweite Szenario eine aufgabenorientierte Umsetzung der räumlichen Programmierung. Bei der Programmieraufgabe „Labyrinth“ ist der Endeffektor des Roboters durch eine Anordnung von Hindernissen (s. Abb. 6.5, links) zu bewegen. Neben Start- und Endpose sind mindestens zwei weitere Posen zur kollisionsfreien Durchfahrt anzulegen und mit Interpolationsbefehlen im sequentiellen Ablauf des textuellen Programms zu verbinden. Das Szenario „Pick and Place“ beinhaltet ein simples Handhabungsszena-

[4] Vgl. Kapitel 2.1.2.

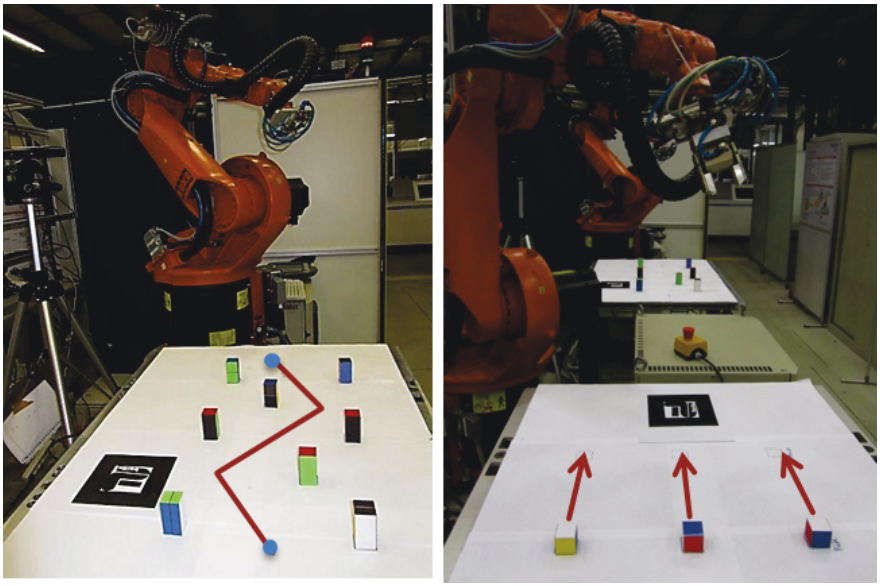

Abbildung 6.5: Programmierszenarien: „Labyrinth“ (links) mit angedeuteter Bewegungsbahn und „Pick and Place“ (rechts) mit Andeutung der Umsetzbewegung.

rio, bei dem drei Werkstücke gegriffen und an einer jeweiligen Zielpose geordnet abgelegt werden sollen (s. Abb. 6.5, rechts).

B) Umsetzung der Versuchsdurchführungen

Als Industrieroboter wird für die Teach-In-Programmierung ein KUKA KR6 eingesetzt. Als OLP dient Siemens Tecnomatix. Die reale Anordnung der Roboterzelle wird vorbereitend in Tecnomatix modelliert. Für die räumliche Programmierung der Labyrinthaufgabe dienen reale Hindernisse. Statt der Definition mehrerer Bewegungsbefehle und Posen kann bei der räumlichen Programmierung alternativ die gesamte Trajektorie durch das Labyrinth mithilfe einer einzelnen Bewegung der Hand, welche als Spline umgesetzt wird, definiert werden. Die Pick-and-Place-Aufgabe wird für die räumliche Programmierung anhand der Interaktion mit virtuellen Werkstücken umgesetzt[5]. Die Startpose der Objekte ist zu Beginn eines Versuchsdurchlaufs gegeben. Für die Ermittlung der Programmierdauer werden beim Teach-In die Dauer zum Verfahren des Roboters, das Anlegen der Posen sowie die textuelle Programmierung miteinbezogen. Sowohl bei der Offline-Programmierung als auch bei der räumlichen Programmierung werden ergänzend die Dauer der Simulation und die Dauer etwaiger Manipulationen des Roboterprogramms erfasst. Der Zeitaufwand zum Testen des Programms durch reales Abfahren wird somit in den Versuchsdurchführungen aus Gründen der Vereinfachung für alle Programmiermethoden vernachlässigt. Als Programmierfehler werden Fehler in der Umsetzung der Programmieraufgabe angesehen. Dazu gehören potentielle Kollisionen, das Vertauschen von Posen oder Befehlen im sequentiellen Ablauf sowie eine allgemeine Untauglichkeit des erstellten Programms. Die subjektive Zufriedenheit der Nutzer wird anhand einer Befragung mit numerischer Bewertung der Programmiermethoden zur Einfachheit und Verständlichkeit erhoben.

[5] Vgl. Kapitel 6.2.

C) Nutzergruppe und Versuchsplanung

An der Nutzerstudie nahmen insgesamt 32 Personen teil (31 Männer, 1 Frau). Die Gruppe der Studienteilnehmer setzte sich aus 30 Studierenden und 2 Mitarbeitern des Fachgebiets Industrielle Automatisierungstechnik im Alter zwischen 23 und 48 Jahren zusammen. Das Durchschnittsalter der Teilnehmer betrug 27,5 Jahre. 8 Teilnehmer verfügten vor der Teilnahme an der Studie über umfangreiche Erfahrungen in der Industrieroboterprogrammierung und werden im Folgenden als „Roboterprogrammierer“ bezeichnet. Sämtliche Teilnehmer erhielten vor den Versuchsdurchführungen eine umfangreiche Einführung in die betrachteten Programmierverfahren. Des Weiteren wurden Testläufe unter Anleitung der Versuchsleiter durchgeführt. Die Reihenfolge, in der die Teilnehmer die drei Programmiermethoden durchführten, wurde randomisiert, um mögliche Effekte auf die Versuchsergebnisse zu nivellieren, welche durch eine korrelierte verborgene Störvariable hervorgerufen werden, bspw. durch Lerneffekte fachfremder Teilnehmer.

D) Umgang mit Störquellen

Im Rahmen der Analyse der Messergebnisse wurden insgesamt sieben einzelne Versuchsdurchführungen als starke Ausreißer identifiziert und von der Auswertung ausgeschlossen. Die Messwerte hätten einen verzerrenden Effekt auf die Ermittlung des Mittelwerts sowie eine Aufblähung der Varianz bzw. der Standardabweichung zur Folge gehabt. Für dieses Vorgehen bestanden sowohl statistische als auch praktische Gründe aus der Versuchsdurchführung. Die Ausreißer betreffen aufgenommene Zeitwerte, welche jeweils über das dreifache der Standardabweichung vom Mittelwert abweichen. Die vier betroffenen Versuchsdurchführungen der räumlichen Programmierung unterlagen der unzureichenden Robustheit des 3D-Motion-Tracking aufgrund des individuellen Körperbaus der Teilnehmer. Die Versuchsdurchführungen wurden dementsprechend abgebrochen oder dauerten auffallend lange. Insgesamt waren somit 93,75 % der Versuchsdurchführungen der räumlichen Programmierung erfolgreich und werden im Rahmen der Auswertung betrachtet. Im Falle der drei betroffenen Durchführungen der Offline-Programmierung wurde die Programmierung des Pick-and-Place-Szenarios nach über einer Stunde abgebrochen, da die jeweiligen Versuchsteilnehmer sichtlich mit der Bedienung der Funktionen zur koordinatengebenden Eingabe der Posen überfordert waren. Die negativen Abweichungen der Programmierdauer riefen bei den betroffenen Teilnehmern zudem leichte Missstimmung bzw. Konfusion hervor, sodass aus Gründen der Vergleichbarkeit auch die subjektive Bewertung der betroffenen Versuchsdurchführungen vernachlässigt wurde. Ein weiterer Ausreißer, welcher bei der Offline-Programmierung vernachlässigt wurde, betraf den Messwert des zugehörigen Versuchsleiters. Dieser gestaltete im Vorfeld seiner Versuchsdurchführung die Aufgabe und wirkte betreuend an den Durchführungen der übrigen Teilnehmer mit.

LABYRINTH	MIN.	1. QUARTIL	MEDIAN	MITTELWERT	3. QUARTIL	MAX.	STD.-ABW.
Offline-Pgm.	04:09	06:54	10:47	10:28	13:58	16:34	04:05
Teach-In	03:55	05:04	06:04	06:13	06:35	09:48	01:35
Räumliche Pgm.	00:17	00:31	00:42	00:48	00:53	01:56	00:24
PICK&PLACE	MIN.	1. QUARTIL	MEDIAN	MITTELWERT	3. QUARTIL	MAX.	STD.-ABW.
Offline-Pgm.	10:15	14:53	18:42	20:39	25:44	41:44	07:54
Teach-In	06:38	10:05	12:51	12:24	13:50	20:00	03:10
Räumliche Pgm.	00:26	00:48	01:06	01:14	01:42	02:27	00:35

Tabelle 6.1: Statistische Kennwerte der Programmierdauer (*min*:*s*) in den Szenarien „Labyrinth“ (oben) und „Pick and Place“ (unten).

6.3.2 Auswertung der Versuchsergebnisse

A) Programmierdauer

Tabelle 6.1 fasst die statistischen Kennwerte der ermittelten Programmierzeiten für beide Programmierszenarien zusammen. In Abbildung 6.6 wird die Verteilung der ermittelten Programmierzeiten anhand eines Boxplots dargestellt. Die Darstellung beinhaltet das untere Quartil (unterer Rand der Box), den Median (Strich innerhalb der Box), das obere Quartil (oberes Ende der Box) sowie Maximal- und Minimalwerte („Whisker“), sofern diese nicht um mehr als das 1,5-fache des Interquartilabstands vom Median abweichen. Liegen höhere Abweichungen einzelner Messwerte vor, sind diese als Ausreißer (Kreise) dargestellt.
Die Auswertungen belegen, dass durch Anwendung der räumlichen Programmierung eine signifikante Reduzierung der Programmierdauer und somit eine Erhöhung der Effizienz erreicht werden. Bezüglich der Mittelwerte konnte insgesamt aus Sicht der Offline-Programmierung beim Labyrinth eine Senkung auf 7,6 % (Roboterprogrammierer: 8,54 %) der ursprünglichen Programmierdauer erfolgen und aus Sicht des Teach-In eine Senkung auf 12,9 % (Roboterprogrammierer: 15,36 %). Bei der aufgabenorientierten Umsetzung der Pick-and-Place-Aufgabe fiel dieser Effekt erwartungsgemäß etwas stärker aus. Hier konnten insgesamt Reduzierungen der Programmierdauer auf 6,0 % (Roboterprogrammierer: 4,82 %) für die Offline-Programmierung sowie auf 10,3 % (Roboterprogrammierer: 7,5 %) für das Teach-In erreicht werden. Somit kann auch für die Gruppe der Roboterprogrammierer eine Reduzierung der Programmierdauer nachgewiesen werden. Weiterhin kann unter Betrachtung der Standardabweichung festgestellt werden, dass die Streuung der Programmierdauer der räumlichen Programmierung im Vergleich wesentlich geringer ausfällt. Auch Minimal- und Maximalwerte liegen näher beieinander; der Maximalwert der räumlichen Programmierung liegt für beide Szenarien deutlich unter dem Minimalwert der übrigen Programmiermethoden. Dies deutet darauf hin, dass auch bei fehlender Vorkenntnis im Bereich der Robotik sowie in der Bedienung von modernen Handheld-Geräten eine deutliche Reduzierung der Programmierzeit im Vergleich zu geübten Programmierern der Standardverfahren erreicht werden kann.

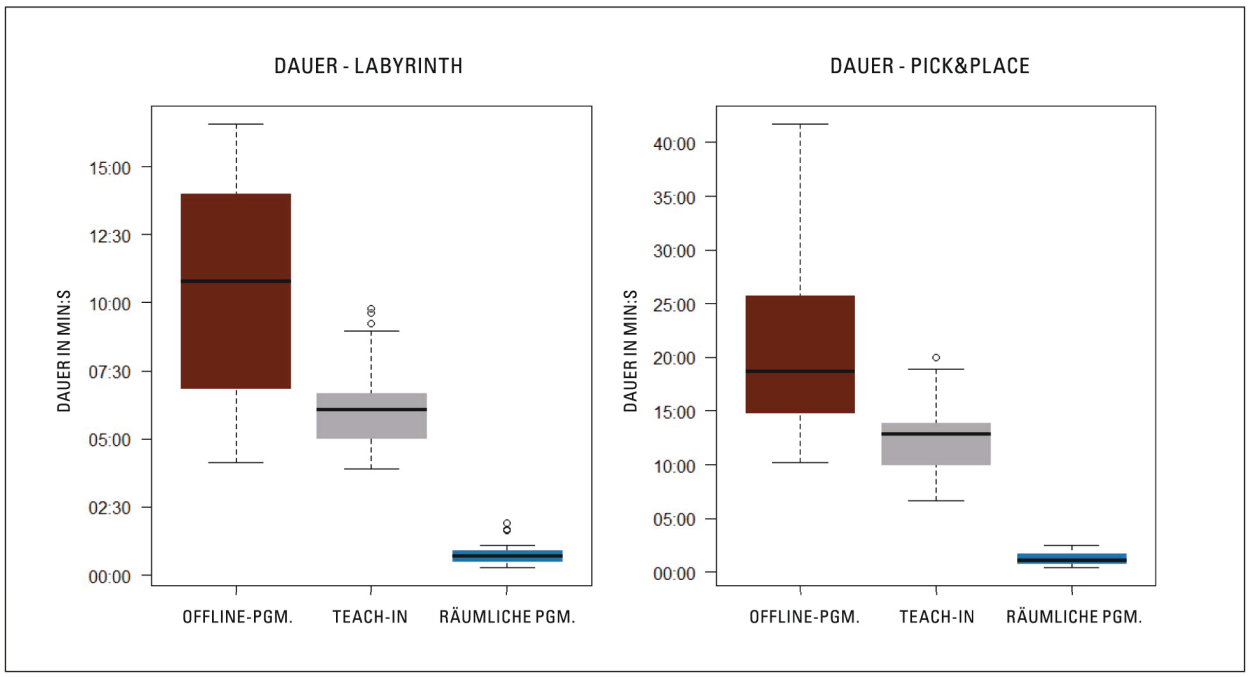

Abbildung 6.6: Boxplot der Ergebnisse für die Programmierdauer in den beiden Programmierszenarien.

Die Offline-Programmierung schneidet für die Programmierdauer am schlechtesten ab. Zudem streuen die Messwerte stärker als bei den übrigen Methoden. Eine erhöhte Programmierdauer kann im Vergleich zum Teach-In zunächst mit der Anwendung der Simulation und einer etwaigen Manipulation des erstellten Programms begründet werden. Darüber hinaus ist anzunehmen, dass die teilweise fehlende Erfahrung der Versuchsteilnehmer im allgemeinen Umgang mit CAD-gestützter Software sowie die Komplexität des eingesetzten OLPs zu einer weiterführenden Erhöhung der Programmierdauer beitragen. Das Teach-In bietet dem fachfremden Anwender im Vergleich durch das reale Anfahren der Posen einen anschaulicheren Einstieg in die Programmierung. Einen Beleg für diese Hypothese bietet die Betrachtung der Minimalwerte für Offline-Programmierung und für das Teach-In. Diese liegen beim Labyrinth relativ nah beieinander, sodass angenommen werden kann, dass bei intensiver Erfahrung im Umgang mit dem OLP ähnliche Programmierzeiten wie beim Teach-In erreicht werden können.

B) Programmierfehler

Die Auswertung der Programmierfehler ergibt für das Labyrinth keine Fehler für die Offline-Programmierung, neun Fehler für das Teach-In und fünf Fehler bei der räumlichen Programmierung. Das entspricht einer relativen Fehlerquote von 28 % für das Teach-In und 16 % für die räumliche Programmierung. Bei der Pick-and-Place-Aufgabe wurden insgesamt ein Fehler für die Offline-Programmierung, neun Fehler für das Teach-In und drei Fehler für die räumliche Programmierung festgestellt. Die daraus resultierende Fehlerquote liegt für die Offline-Programmierung bei 3,7 %, für das Teach-In bei 31 % und für die räumliche Programmierung bei 19 %. Abbildung 6.7 fasst die ermittelten Fehlerquoten für beide Programmierszenarien zusammen.
Erwartungsgemäß liegt die Fehlerquote der Offline-Programmierung durch die Anwendung der Simulation und anschließender Möglichkeit zur Manipulation der Programmparameter bei genau

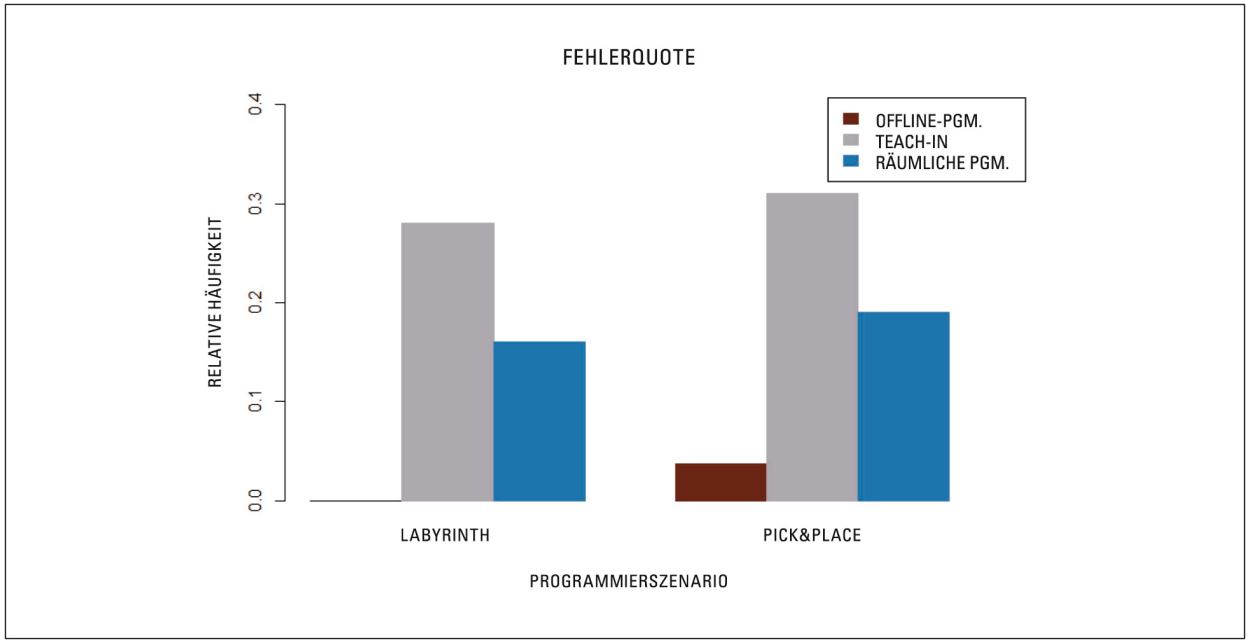

Abbildung 6.7: Fehlerquote der Programmiermethoden in den beiden Programmierszenarien.

bzw. nahezu 0 %. Dagegen liegt die Fehlerquote der Teach-In-Programmierung durch die fehlende räumliche Evaluationsmöglichkeit des Programms in beiden Szenarien bei knapp einem Drittel. Das Auftreten der Fehler hätte in der Praxis zusätzlichen Aufwand zur Korrektur des Programms zur Folge. Die Fehler der räumlichen Programmierung sind ausschließlich auf die begrenzte Robustheit des Tracking zurückzuführen. Trotz Simulation und dem Versuch der Manipulation war es bei den betrachteten Versuchsdurchführungen nicht möglich, einzelne Posen korrekt anzulegen. Der Grund dafür liegt wahrscheinlich in einer ungünstigen Positionierung des Motion-Tracking-Systems im Zusammenspiel mit der Körperstatur einzelner Versuchsteilnehmer. Festzuhalten bleibt, dass durch Anwendung der räumlichen Programmierung die Häufigkeit von Programmierfehlern im Vergleich zum Teach-In gesenkt werden konnte.

C) Subjektive Bewertung der Nutzerzufriedenheit

Tabelle 6.2 fasst die statistischen Kennwerte der ermittelten Nutzerzufriedenheit (1 = „sehr zufrieden“ bis 6 = „gar nicht zufrieden“). zusammen. Für beide Programmierszenarien favorisieren die Versuchsteilnehmer unter Betrachtung der Mittelwerte eindeutig die räumliche Programmierung (Mittelwerte 1,64 und 1,71), gefolgt vom Teach-In (Mittelwerte 2,16 und 2,14) und der Offline-Programmierung (Mittelwerte 2,5 und 2,74). In beiden Szenarien wird zudem die Zufriedenheit mit der räumlichen Programmierung von mehr als der Hälfte der Teilnehmer (s. Abb. 6.8) mit der besten Bewertung („sehr zufrieden“) eingestuft (Median jeweils 1). Somit kann festgestellt werden, dass durch die Interaktionsmethoden der räumlichen Programmierung eine Verbesserung der Nutzerzufriedenheit im Vergleich zu den Standardverfahren der Industrieroboterprogrammierung erreicht wird. Es wird vermutet, dass bei erhöhter Robustheit des Motion-Tracking die Nutzerzufriedenheit der räumlichen Programmierung weiterführend gesteigert werden kann. Die verhältnismäßig negative Bewertung der Offline-Programmierung ist analog zur Programmierdauer mit der

LABYRINTH	MIN.	1. QUARTIL	MEDIAN	MITTELWERT	3. QUARTIL	MAX.	STD.-ABW.
Offline-Pgm.	1	2	2	2,50	3	5	1,07
Teach-In	1	1,75	2	2,16	3	5	0,46
Räumliche Pgm.	1	1	1	1,64	2	4	0,94
PICK&PLACE	MIN.	1. QUARTIL	MEDIAN	MITTELWERT	3. QUARTIL	MAX.	STD.-ABW.
Offline-Pgm.	1	2	3	2,74	3	5	1,02
Teach-In	1	2	2	2,34	3	5	0,97
Räumliche Pgm.	1	1	1	1,71	2	4	0,90

Tabelle 6.2: Statistische Kennwerte der Nutzerzufriedenheit in den Szenarien „Labyrinth“ (oben) und „Pick and Place“; 1 = „sehr zufrieden“ bis 6 = „gar nicht zufrieden“.

Komplexität der eingesetzten Software und mit der mangelnden Erfahrung der Teilnehmer zu begründen. Die Bewertungen für das Pick-and-Place-Szenario sind im Vergleich zum Labyrinth für alle Programmiermethoden unwesentlich schlechter. Die Differenz der Mittelwerte der Bewertungen von räumlicher Programmierung zum Teach-In steigt zudem leicht von 0,52 auf 0,63. Die entsprechende Differenz der Bewertungen zur Offline-Programmierung steigt ebenfalls von 0,86 auf 1,03. Dieser Umstand weist darauf hin, dass die relative Nutzerzufriedenheit durch eine aufgabenorientierte Umsetzung der räumlichen Programmierung leicht gesteigert werden kann. Zunächst unerwartet ist, dass die Zufriedenheit der räumlichen Programmierung bei der aufgabenorientierten Umsetzung schlechter eingestuft wir als beim bewegungsorientierten Programmierszenario. Als Grund kann die mangelnde Vergleichbarkeit der Programmieraufgaben, im Speziellen die höhere Komplexität der Pick-and-Place-Aufgabe, angesehen werden.

6.3.3 Eingrenzung der Aussagekraft der Studienergebnisse

Die Aussagen der Auswertung haben nur begrenzte Allgemeingültigkeit. Kritik an der Repräsentativität lässt sich aufgrund der homogenen Zusammensetzung der Nutzergruppe üben. Diese bestand überwiegend aus männlichen Studierenden ingenieurwissenschaftlicher Studiengänge in einem Altersbereich zwischen 23 und 30 Jahren. Bei dieser Zusammensetzung kann davon ausgegangen werden, dass eine verstärkte Technikaffinität, eine gute Auffassungsgabe sowie eine allgemeine Vertrautheit im Umgang mit Smartphones und Tablet-PCs vorliegt. Eine Beschränkung der Repräsentativität besteht zudem aufgrund der begrenzten Teilnehmerzahl. Eine differenzierte Auswertung bezüglich fachfremder Nutzer und Roboterprogrammierer wurde aufgrund der beschränkten statistischen Aussagekraft lediglich partiell vorgenommen.
Weitere Einschränkungen liegen in der Übertragbarkeit der Ergebnisse auf industrielle Anwendungsszenarien. Die Gestaltung der Szenarien wurde vorrangig auf die Definition von Bewegungen ohne hohe Genauigkeitsanforderungen ausgerichtet. Gründe hierfür liegen zum einen in einer Aufwandsminimierung zur erstmaligen Evaluation des räumlichen Programmiersystems, zum anderen in den beschränkten Genauigkeiten des eingesetzten Motion-Tracking-Systems. Die Ergeb-

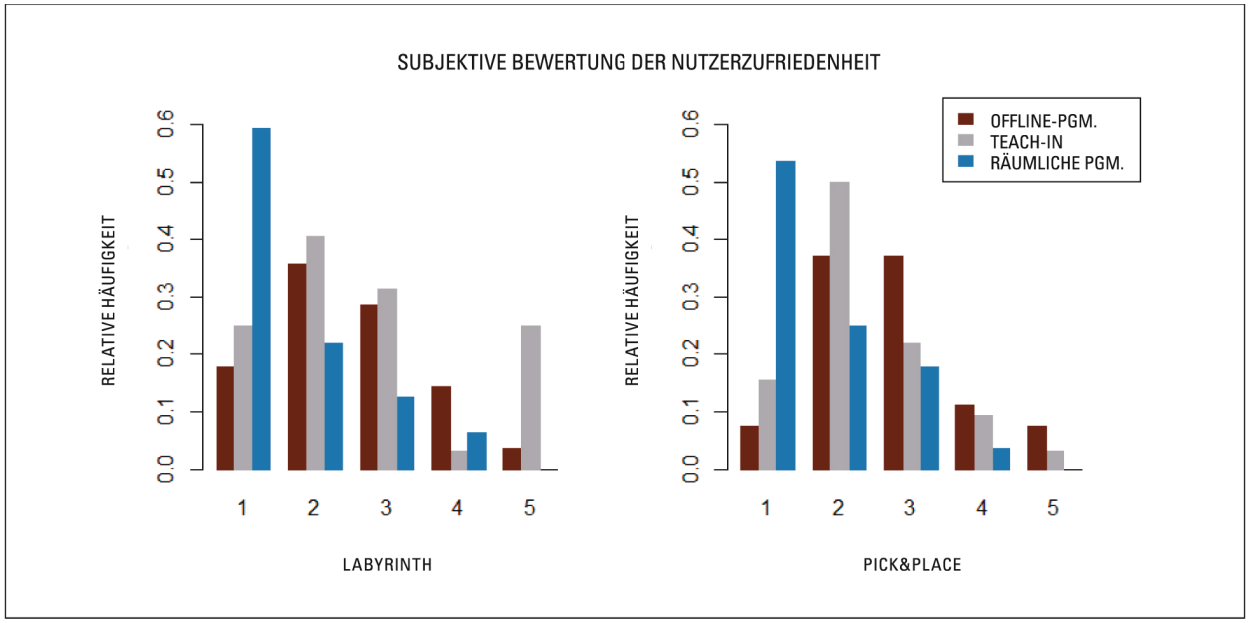

Abbildung 6.8: Relative Häufigkeit der Kennwerte der Nutzerzufriedenheit bei beiden Programmierszenarien; 1 = „sehr zufrieden“ bis 6 = „gar nicht zufrieden“.

nisse des OLPs beziehen sich zunächst auf Siemens-Tecnomatix. Eine Übertragung der Ergebnisse auf andere OLPs ist aufgrund der Unterschiede in Softwareergonomie, Softwarekomplexität und ggf. der Nutzung von Technologiemodulen zur aufgabenorientierten Programmierung nur bedingt möglich. Allgemein besteht die Motivation, die Versuchsdurchführungen an einem Industrieszenario unter Verwendung eines genaueren Motion-Tracking-Systems und mit einer inhomogenen Nutzergruppe, einschließlich eines hohen Anteils an Industrieroboterprogrammierern, zu wiederholen.

6.4 Technologiebewertung

Die aufgabenorientierte Erprobung und die Nutzerstudie belegen die allgemeine Funktionsfähigkeit des natürlich-räumlichen Programmiersystems sowie dessen Vorteile gegenüber den klassischen Programmierverfahren. Anhand allgemeiner industrieller Anforderungen bezüglich Genauigkeiten, Zuverlässigkeit sowie Integrationsfähigkeit wird im Rahmen einer vertiefenden Technologiebewertung die prototypische Umsetzung des Programmiersystems näher betrachtet. Zudem werden Lösungsmöglichkeiten für einzelne Problemstellungen aufgezeigt.

6.4.1 Gestenbasierte Interaktion

A) Fingergestenerkennung

Das 2D-Tracking der Hand und der Finger funktioniert auch bei Bewegungen vor heterogenem Hintergrund robust und in Echtzeit. Die Erkennungsraten der Gesten liegen bei anwendungsge-

rechter Nutzung bei nahezu 100 %. Die Grenzen des Algorithmus werden lediglich bei schnellen Bewegungen der Hand und bei Hintergründen, die zu hohem Anteil aus hautfarbenen Objekten bestehen, erreicht. Beide Einflüsse können zu einem Verlust der Handverfolgung führen. Die mangelnde Robustheit unter diesen Voraussetzung resultiert primär aus der limitierten Anzahl der verwendeten Partikel für den Condensation-Algorithmus. Die Wahl einer höheren Partikelanzahl beeinflusst die Echtzeitfähigkeit negativ. Bei der Erprobung wurde eine adäquate Partikelanzahl von 100 identifiziert, die aus einem Trade-off zwischen Performance (Kriterium: fps) des Tracking sowie der Robustheit von Simulation und Tracking resultiert. Durch den Einsatz von leistungsfähigeren Mobilgeräten kann eine erhöhte Anzahl an Partikeln gewählt und somit die Robustheit in Zukunft weiterführend gesteigert werden. Eine begrenzte Robustheit des Tracking konnte zudem bei Nutzern identifiziert werden, deren individuelle Handkontur von den betrachteten Vorlagen abweicht. Neben der vorhandenen Anpassung des Farbmodells an die individuelle Hautfarbe des Nutzers ist an dieser Stelle eine Anpassung der Kontur an die individuelle Form der Handkontur des Nutzers erstrebenswert. Die Umsetzung des effizienten CR-Klassifikators zur Messung der Hautfarbe orientiert sich an Trainingsdaten von hellhäutigen Anwendern. Ob eine Funktionalität des Klassifikators auch bei dunkelhäutigen Personen gegeben ist, konnte im Rahmen der vorliegenden Arbeit nicht untersucht werden.
Durch die Betrachtung festgelegter 2D-Konturen für die Hand ist die natürliche 3D-Interaktion lediglich beschränkt möglich. Im Sinne einer Verbesserung der Gebrauchstauglichkeit des Entwurfs zur gestenbasierten Interaktion wird die Betrachtung weiterer Handmodelle bis hin zu einem Tracking basierend auf einem 3D-Modell angestrebt. Letztere Technologie ermöglicht eine Interaktionsfreiheit der Hände in sechs Freiheitsgraden mit einem maximalen Grad an Natürlichkeit. Aufgrund der umfangreichen Anforderungen an die Rechenleistung wird ein 3D-Modell-basiertes Tracking jedoch in naher Zukunft auf Mobilgeräten nicht in Echtzeit umgesetzt werden können. Ein alternativer Ansatz besteht darin, die Bildverarbeitung auf externe, leistungsstarke Recheneinheiten auszulagern. Für die Sicherstellung der Usability müssten allerdings die Verzögerungen für das Versenden der Bilddaten vom Mobilgerät minimal gehalten werden. Als weitere, jedoch kostenintensive Möglichkeit zur Realisierung der Fingergestenerkennung ließe sich ein externer Multi-Kamera-Ansatz wählen.

B) 3D-Motion-Tracking

Das 3D-Motion-Tracking der Hand über den Kinect-Sensor funktioniert in der Regel robust in Innenräumen mit knapp 30 Hz. Die Robustheit sinkt aufgrund des Sensorprinzips mit dem Einfall von natürlichem Licht. Bei der Erprobung des Programmiersystems wurde durch die Anwender eine leicht verzögerte Reaktionszeit durch Berechnung des Skeleton-Modells und durch das Versenden der Handkoordinaten festgestellt. Dieser Umstand beeinflusst die Usability der räumlichen Interaktion jedoch nicht signifikant. Das Skeleton-Tracking unterliegt der aufgeführten Absolutgenauigkeit von $\pm$1–4 cm[6]. Diese Genauigkeit wurde im Rahmen der vorliegenden Arbeit durch stichprobenartige Untersuchungen verifiziert. Ergänzend zur allgemeinen Ungenauigkeit des Tracking bezieht sich die Position aus dem Skeleton-Modell auf einen willkürlichen Punkt der Hand und nicht auf einen definierten Punkt der Fingerspitze. Diesbezüglich wurden durch Stich-

[6] Vgl. Kapitel 4.3.2.

proben im Rahmen dieser Arbeit eine Wiederholgenauigkeit des Skeleton-Tracking von $\pm$0,4–2 *cm* ermittelt. Der Anwender wird durch das visuelle Feedback der aktuellen Handposition in der AR jedoch in die Lage versetzt, mittels Hand-Auge-Koordination die aktuelle Position durch Nachführen der Hand anzupassen. Die durch dieses Vorgehen zu erreichende Genauigkeit unterliegt prinzipiell der Absolutgenauigkeit des Marker-Tracking bzw. der AR-Visualisierung.
Da die erreichte Abstandsgenauigkeit des Tracking wesentlich genauer als die Absolutgenauigkeit ist, kann die gestenbasierte Manipulation von Objekten präziser erfolgen. Trotz der Adaptierbarkeit der angelegten Position durch die AR-Darstellung und die räumliche Manipulation können die Genauigkeitsanforderungen der meisten industriellen Robotikanwendungen durch den eingesetzten Kinect-Sensor nicht erreicht werden. Um dennoch eine Anwendbarkeit zu ermöglichen, lässt sich anwendungsspezifisch auf zusätzliche Sensorik zur Steigerung der Genauigkeiten, bspw. Sensoren zur Nahtfindung und -verfolgung beim robotergestützten Schweißen, zurückgreifen. Alternativ lässt sich durch den modularen Aufbau des Gesamtsystems der Kinect-Sensor gegen ein genaueres Motion-Tracking-System tauschen. Neben dem Einsatz von kostenintensiveren Tracking-Systemen kann der bestehende Ansatz unter Verringerung der Natürlichkeit der Interaktion auf zusätzliche Geräte zur präzisen Eingabe, wie bspw. Zeigestifte, erweitert werden.

6.4.2 Visualisierung und Simulation

Die Umsetzung der AR bietet eine anschauliche Visualisierung des Roboterprogramms auf beliebigen Mobilgeräten in realer Umgebung. Der visuelle Abgleich zwischen virtuellen und realen Bahnen des Roboters über die App erscheint präzise. Die Darstellung ist in der Regel auch bei Nutzung eines einzelnen Markers für eine qualitative Evaluation des Programms ausreichend. Für höhere Genauigkeiten bzw. die Abdeckung eines höheren Interaktionsbereichs ist zunächst die Positionierung und Größe der Marker zu optimieren, da die Genauigkeit der Darstellung vom Abstand und vom Winkel zwischen Marker und Mobilgerät abhängt[7]. Die in [48] angegebenen Genauigkeiten für das Marker-Tracking konnten im Rahmen dieser Arbeit durch Stichproben bestätigt werden. Bei größeren Arbeitsräumen sowie einer hohen Flexibilität des Nutzers zur freien Bewegung in der Roboterzelle empfiehlt sich ein entsprechendes Multi-Marker-Tracking.
Einzelne Probleme bei der Wahrnehmung bestehen durch die allgemeinen Grenzen der AR-Technologie. Diese liegen unter anderem in der begrenzten Tiefenwahrnehmung bei statischer Betrachtung virtueller Objekte. Etwaige Konfusionen des Nutzers über die Tiefeninformationen von virtuellen Objekten lassen sich in mobilen Anwendungen wie dem vorliegenden System durch Bewegungen des Anzeigegeräts kompensieren. Zusätzlich bestehen Ansätze, die Tiefenwahrnehmung über Variation der Farben der virtuellen Objekte bzw. Variation der virtuellen Beleuchtung zu verbessern. Weiterhin besteht die Problematik der Verdeckungen, der im Rahmen dieser Arbeit rudimentär durch die Darstellung von transparenten geometrischen Primitiven im Kamerabild begegnet wurde[8].
Auf die Verwendung des Marker-Tracking zur Darstellung der AR kann durch ein hinreichend genaues externes Tracking des Mobilgeräts verzichtet werden. Als Resultat müssten keine Marker

[7] Vgl. Kapitel 4.2.1.
[8] Vgl. Kapitel 5.1.2.

als künstliche Merkmale im Arbeitsraum des Roboters angebracht und kalibriert werden. Um eine entsprechende Interaktionsfreiheit zu gewährleisten, empfiehlt sich zu diesem Zweck die Verwendung eines multisensoriellen Ansatzes, bspw. eines Tracking des Mobilgeräts durch ein externes Multi-Kamera-System.
Bezüglich der Visualisierung der bewegungsorientierten Programmierebene werden im Moment lediglich Posen und Trajektorien dargestellt. Eine Weiterentwicklung des Programmiersystems hin zur industriellen Einsetzbarkeit sollte darauf abzielen, die Anzahl der betrachteten Programmierbefehle auf den gebrauchsüblichen Umfang der Roboterprogrammierung zu erweitern und ggf. eine adäquate räumliche Visualisierung der Programmbefehle bereitzustellen. Aktionen lassen sich bspw. räumlich mit Posen, an denen sie ausgeführt werden, symbolisch verknüpfen. Hierzu kann eine entsprechende anschauliche Visualisierung, bspw. als Icon, an der räumlichen Stelle der entsprechenden Pose vorgenommen werden. Des Weiteren könnten Schleifen und Bedingungen mit in die Simulation integriert werden. Die Visualisierung im räumlichen Programmiersystem stößt analog zur Offline-Programmierung an ihre Grenzen, wenn Bedingungen oder Bewegungen abgebildet werden sollen, welche auf externen Signalen basieren.

6.4.3 Integrationsfähigkeit des Programmiersystems

Die Inbetriebnahme des Programmiersystems erfolgt über die manuelle Aufnahme von Positionskorrespondenzen in den verschiedenen KS der beteiligten Komponenten[9]. Der Vorteil des manuellen Anfahrens der Referenzpunkte durch den Roboter liegt in der Genauigkeit, die lediglich von der Absolutgenauigkeit des Roboters und der Kalibrierung des Messwerkzeugs abhängt. Die Koordinaten der Eckpunkte des Markers lassen sich bezogen auf den Marker-Mittelpunkt als Ursprung des KS exakt aus den bekannten Abmaßen ableiten. Die aufgenommenen Positionen des Motion-Tracking-KS unterliegen der Absolutgenauigkeit des Kinect-Sensors[10]. Der Nachteil dieses Verfahrens besteht darin, dass der Roboter manuell verfahren werden muss. Je nach Anzahl der aufgenommenen Positionen und der Qualifikation des Anwenders kann ein derartiges Vorgehen ca. 30–60 *min* dauern. Zudem muss der Nutzer über Kenntnisse zum manuellen Verfahren des Roboters verfügen. Um den Aufwand der manuellen Kalibrierung zu reduzieren, ist es aus Kostengründen und zur Sicherstellung einer konstanten Qualität erstrebenswert, die Kalibrierung der KS voll- oder teilautomatisiert durchzuführen. Eine Möglichkeit für die Realisierung eines automatisierten Kalibrierverfahrens besteht im Abgleich von Referenzmerkmalen an Marker und Roboter durch das eingesetzte kamerabasierte Motion-Tracking-System. Zu diesem Zweck lässt sich auf AR-Marker mit dem in Anhang A beschriebenen Algorithmus zur Ermittlung der Transformationsmatrix zurückgreifen.

[9] Vgl. Kapitel 5.4.3.
[10] Vgl. Kapitel 6.4.1 B).

6.5 Zusammenfassende Bewertung

Abschließend soll auf Basis der prototypischen Implementierung sowie der Erprobung eine explizite Beantwortung der aufgestellten Forschungsfragen zum Zweck der Verifizierung der zentralen Forschungshypothese vorgenommen werden. Die Antworten dienen des Weiteren der zusammenfassenden Darstellung der Evaluation.

6.5.1 Mensch-Maschine-Interaktion

Wie kann ein natürlich-räumliches Interaktionskonzept gestaltet werden und welche Modalitäten und Funktionen sind bereitzustellen?

Grundlegend sieht das Interaktionskonzept die räumliche Eingabe über Gesten und eine räumliche Ausgabe über eine AR-Anwendung vor. Die Gestenerkennung lässt sich markerlos und ohne Hilfsmittel ausführen, um einen hohen Grad an Natürlichkeit in der Interaktion zu erreichen. Posen, Bewegungsbahnen und Aufgaben lassen sich demnach über natürliche Bewegungen an räumlicher Stelle ihrer geplanten Ausführung definieren, sodass ein ein hoher Grad an Direktheit der Interaktion erreicht wird. Anhand von Marker-Tracking lässt sich die Augmented Reality auf Handheld-Geräten mobil ausführen, sodass sich die Anzeige auf VST-Geräten an die Pose des Handheld-Geräts anpasst. Die synergetische Nutzung von Gesten und AR ermöglicht zudem eine neuartige Form der interaktiven Programmmanipulation durch räumliche Interaktion mit der virtuellen Darstellung des Roboterprogramms im Raum. Ergänzend werden Touch-Gesten zur Programmverwaltung sowie ein unterstützendes vibrotaktiles Feedback zur räumlichen Gestenerkennung eingesetzt.

Kann eine wirtschaftliche Umsetzung des Interaktionskonzepts hinsichtlich des Einsatzes speziell für KMU erfolgen?

Die Erprobung der prototypischen Umsetzung zeigt, dass sich eine kosteneffiziente Umsetzung des Programmiersystems durch die Nutzung von Standardkomponenten und Sensorik aus dem Consumer-Bereich erfolgreich realisieren lässt. Der aufgezeigte Ansatz zur Gestenerkennung umfasst die integrative Betrachtung von 2D-Fingergesten und 3D-Motion-Tracking. Die Genauigkeiten des eingesetzten Kinect-Sensors genügt jedoch nicht den Anforderungen gängiger Industrieroboterapplikationen. Eine Kompensation der Ungenauigkeit lässt sich einhergehend mit einer Kostensteigerung durch die Einbindung zusätzlicher Sensorik oder die Wahl präziserer Sensorik für das 3D-Motion-Tracking realisieren. Als zentrale Ausgabe- und Programmiergeräte fungieren Smartphones und Tablet-PCs. Diese überzeugen durch vergleichsweise geringe Kosten und Zusatzfunktionen, welche eine vielfältige Einsetzbarkeit im Unternehmen eröffnen.

6.5.2 Programmierung

Welche grundlegenden Funktionen muss das Programmiersystem bereitstellen?

Die Programmierumgebung stellt grundlegende Funktionen zur Programmrepräsentation, Programmverwaltung sowie zur Ein- und Ausgabe bereit. Die bewegungsorientierte Programmrepräsentation umfasst einen neutralen Satz an Roboterbefehlen. Aufbauend wurde eine aufgabenorientierte Programmierebene in Form eines hierarchischen Ansatzes erstellt, auf deren Funktionen die Programmdialoge der App, die Simulation sowie die räumliche Interaktion aufsetzen. Ferner dient die neutrale Programmrepräsentation als Grundlage des Austauschs der Roboterprogramme über die Kommunikationsschnittstellen der App.

Inwiefern kann das Programmiersystem bedarfsgerecht modularisiert werden?

Eine bedarfsgerechte Ausführung des Programmiersystems hinsichtlich geforderter Prozessgenauigkeiten sieht die Anpassung des Wahrnehmungskonzepts der Gestenerkennung vor. Diese Anpassungen betreffen vorrangig die Wahl alternativer Sensorsysteme sowie eine Anpassung der Signalverarbeitung und Signalaufbereitung. In der vorliegenden Umsetzung wird das 3D-Motion Tracking über eine externe Zusatzanwendung realisiert, welche die Positionswerte der Hand an die App über drahtlose Kommunikation übermittelt. Die externe Anwendung lässt sich bezogen auf das Programmiersystem ohne Aufwand austauschen, sodass eine effiziente Anbindung verschiedener Sensorsysteme gewährleistet ist. Einzelne Softwarefunktionen wurden in der Umsetzung anhand von Hauptkomponenten, Funktionsmodulen sowie der Definition interner Schnittstellen gekapselt, sodass sich Anpassungen bspw. ein Austausch des Robotermodells oder der Interpolationsalgorithmen unkompliziert umsetzen lassen.

6.5.3 Schnittstellen und Inbetriebnahme

Wie lässt sich eine breite Anwendbarkeit des Programmiersystems auf Industrierobotersysteme verschiedener Hersteller realisieren?

Eine breite Anwendbarkeit des Programmiersystems wird durch die vereinheitlichte Roboterschnittstelle gewährleistet. Diese ermöglicht unabhängig von herstellerspezifischen externen Schnittstellen die Programmübertragung zwischen mobilem Programmiergerät und Robotersteuerung. Die Schnittstelle wurde an Robotersystemen mehrerer Hersteller getestet. Eine Übertragung des Programmiersystems auf weitere Roboterhersteller erfordert lediglich die einmalige Anpassung des Interpreterprogramms an die jeweilige Programmiersprache.

Wie kann das Programmiersystem in die bestehende informationstechnische Infrastruktur im Unternehmen eingebunden werden?

Eine Einbindung des Programmiersystems in die informationstechnische Infrastruktur von Unternehmen lässt sich durch Integration in bestehende drahtlose Netzwerke vornehmen. In der vorliegenden Arbeit erfolgt eine Anbindung des mobilen Programmiersystems an die Robotersimulation der Digitalen Fabrik. Ein beidseitiger Austausch von Roboterprogrammen zwischen prozessnaher Programmierung in der App und prozessferner Robotersimulation wird in der prototypischen

Umsetzung des Programmiersystems realisiert und erprobt. Eine Erweiterung der ausgetauschten Daten, bspw. auf CAD-Daten oder Prozessparameter, ist gegeben.

Wie lässt sich die Inbetriebnahme des Programmiersystems realisieren?

Die Inbetriebnahme des Programmiersystems erfordert die Ermittlung der statischen Verschiebungen zwischen den KS des Motion-Tracking-Systems, der AR-Marker sowie des Roboters. Diesbezüglich wurde in der vorliegenden Arbeit ein Ansatz zur Berechnung der homogenen Transformationsmatrizen anhand von Positionskorrespondenzen vorgestellt. Die Durchführung der Inbetriebnahme erfordert zurzeit unter anderem das manuelle Verfahren des Roboters. Das aufgezeigte Vorgehen bildet die Basis für weiterführende Arbeiten zur effizienten Ermittlung der Transformationen über Ansätze der Bildverarbeitung zur automatisierten Kalibrierung der KS.

6.5.4 Benutzergerechte Auslegung

Wie können Anwender mit verschiedener Qualifikation in der Industrieroboterprogrammierung durch adäquate Funktionsbereitstellung individuell unterstützt werden?

Als potentielle Nutzergruppen des Programmiersystems werden erfahrene Roboterprogrammierer und fachfremde Anwender identifiziert. Beide Gruppen lassen sich hinsichtlich ihrer individuellen Voraussetzungen gezielt durch einen unterschiedlichen Abstraktionsgrad der Programmierung unterstützen. Während der Roboterprogrammierer auf der bewegungsorientierten Programmierebene hohe Flexibilität in der Programmgestaltung erfährt, wird dem fachfremden Anwender eine aufgabenorientierte Definition von Standardaufgaben ermöglicht. Somit ist der fachfremde Anwender in der Lage, ohne breites Wissen im Bereich der Robotik Programme durch Demonstration oder Instruktion zu definieren. Entsprechende Auslegungen der Ein- und Ausgabefunktionen wurden im Programmiersystem vorgesehen. Die Erprobung zeigt eine erfolgreiche Umsetzung der aufgabenorientierten Programmierung am Beispiel der Definition einer Pick-and-Place-Aufgabe durch Demonstration. Zur Erweiterung des Anwendungsspektrums der aufgabenorientierten Programmierung sind jeweils spezifische Anpassungen an der methodischen Umsetzung des Interaktionskontexts vorzunehmen.

7 Zusammenfassung und Ausblick

7.1 Zusammenfassung

Die steigenden Anforderungen an die produzierenden Unternehmen in den westlichen Industrienationen erfordern neue Forschungsansätze zur wirtschaftlichen und flexiblen Durchführung der Industrieroboterprogrammierung. Die bisher in der Industrie verbreiteten prozessnahen Verfahren weisen vor allem Defizite bei der Gestaltung der MMI zur Eingabe und Abstraktion des Programms sowie bei der Evaluation der Programme auf. Im Bereich der MMI bestehen Forschungs- und Entwicklungsansätze zu räumlichen Nutzerschnittstellen, welche unter anderem natürliche Kommunikation in Form von 3D-Gesten sowie eine Simulation virtueller Objekte in realer Umgebung durch AR verwenden. Zielsetzung der vorliegenden Arbeit war in Abgrenzung zum Stand der Technik die Entwicklung eines prozessnahen, räumlichen Programmiersystems unter Verwendung natürlicher MMK. Im Rahmen einer Anforderungsanalyse wurden zunächst Teilziele zur Konzeption des Programmiersystems in den Bereichen der Interaktion, Programmierumgebung, Schnittstellen und Inbetriebnahme sowie der anwenderspezifischen Auslegung abgeleitet.
Die Gestaltung der Interaktion sieht die Definition von Posen, Bewegungsbahnen sowie Aufgaben des Roboters über markerlose Gesten durch Vormachen nach dem PbD-Paradigma vor. Der Anwender wird dabei in die Lage versetzt, auch komplexe Trajektorien ohne Hilfsmittel, bildlich gesprochen, in die Luft zu zeichnen. Ergänzend zur klassischen bewegungsorientierten Erstellung und Darstellung der Roboterprogramme bietet eine aufgabenorientierte Programmierebene die Möglichkeit, Roboterprogramme auf einem höheren Abstraktionsgrad, bspw. für den fachfremden Anwender, durch Vormachen oder Instruktion zu erstellen. Neben der gestenbasierten Eingabe von Bewegungsparametern wird anhand einer mobilen AR-Anwendung auf herkömmlichen Smartphones und Tablet-PCs die Visualisierung des Roboterprogramms in realer Umgebung ermöglicht. Somit lassen sich angelegte Posen, Trajektorien und Aufgaben im Anschluss an die Definition an Ort und Stelle verifizieren und evaluieren. Hierzu dient zusätzlich eine Simulation des Roboterprogramms in der AR durch einen virtuellen Roboter, welcher im Bild des Mobilgeräts über den realen Roboter gelegt wird. Zur räumlichen Manipulation des Roboterprogramms wird ein kombinatorischer Ansatz unter synergetischer Verwendung von Gesten und AR verfolgt. Dieser umfasst die gestenbasierte räumliche Manipulation der virtuell im Kamerabild dargestellten Objekte, welche Bewegungsparameter des Roboterprogramms repräsentieren. Demnach lassen sich Posen und Trajektorien frei im Raum verschieben und verdrehen. Das Roboterprogramm wird entsprechend der Interaktion angepasst. Eine derartige Programmierung ermöglicht eine anschauliche und effiziente Erstellung von Roboterprogrammen sowohl für fachfremde Anwender

als auch für erfahrene Roboterprogrammierer.
Als zentrale Programmiergeräte dienen Smartphones und Tablet-PCs. Die zugehörige App wurde für das Android-Betriebssystem umgesetzt und umfasst sowohl eine Programmierumgebung zur Programmverwaltung als auch die AR-Anwendung. Die Umsetzung der Gestenerkennung erfolgt durch einen hybriden Ansatz in Form eines externen Sensors zum 3D-Motion-Tracking der Hand des Anwenders sowie der Erkennung von Fingergesten auf Basis des 2D-Kamerabilds des Handheld-Geräts. Bei der Umsetzung der Bildverarbeitung auf dem Mobilgerät wird ein Kompromiss zwischen Robustheit und Performance erzielt, sodass eine Echtzeitfähigkeit der Simulation und der Gestenerkennung gewährleistet wird. Die drahtlose Übertragung und Ausführung der Roboterprogramme auf der realen Steuerung wird über eine vereinheitlichte Schnittstellendefinition ermöglicht. Die entwickelte vereinheitlichte Roboterschnittstelle stellt die Übertragbarkeit des Programmiersystems auf Robotersysteme beliebiger Hersteller her. Somit ist eine breite Anwendbarkeit des Programmiersystems gegeben. Des Weiteren wurde eine drahtlose Schnittstelle zum Austausch von Informationen zur Digitalen Fabrik entwickelt und in die App implementiert.
Erprobt wurde das Programmiersystem an einer aufgabenorientierten Umsetzung einer Pick-and-Place-Aufgabe, bei der der Nutzer den Handhabungsprozess durch die Interaktion mit virtuellen Objekten vormacht. Das entsprechende Roboterprogramm wird automatisch aus der beobachteten Interaktion abgeleitet. Die Evaluation des prototypischen Programmiersystems erfolgt anhand einer umfangreichen Nutzerstudie sowie mithilfe einer Technologiebewertung der prototypischen Umsetzung. Die Studie vergleicht die räumliche Programmierung in zwei Anwendungsszenarien mit der klassischen Teach-In-Programmierung sowie mit der simulationsgestützten Offline-Programmierung. Die Ergebnisse belegen die Vorteile der räumlichen Programmierung in Effizienz, Effektivität sowie Nutzerzufriedenheit, darunter eine signifikante Reduzierung der Programmierdauer sowie eine Verbesserung der subjektiven Nutzerzufriedenheit. Zusätzlich lässt sich die Häufigkeit von Programmierfehlern im Vergleich zum Teach-In senken. Die technologische Bewertung stellt fest, dass die industrielle Anwendbarkeit des Programmiersystems prinzipiell gegeben ist, identifiziert jedoch Verbesserungspotentiale in Bezug auf die aufgezeigte Umsetzung.

Durch die Erprobung und Evaluation wurde die aufgestellte Forschungshypothese erfolgreich belegt. Demnach ist es möglich, ein räumliches System zur prozessnahen Programmierung von Industrierobotern unter Nutzung von Gesten und Augmented Reality natürlich zu gestalten und als kostengünstigen Systementwurf prototypisch umzusetzen.

7.2 Ausblick

Ergänzend zur Darlegung von Verbesserungspotentialen für einzelne Problemstellungen der vorgestellten prototypischen Umsetzung im Rahmen der Technologiebewertung[1] soll an dieser Stelle ein allgemeiner Ausblick zu verschiedenen Aspekten der Anwendung und Interaktion sowie zur

[1] Vgl. Kapitel 6.4.

weiterführenden Integration des Programmiersystems in moderne und zukünftige Strukturen der Produktionsorganisation gegeben werden.

7.2.1 Anpassung des Anwendungs- und Interaktionsszenarios

A) Integration und Ausführung alternativer Sensorkonzepte

Die Genauigkeiten der gestenbasierten Programmdefinition und -manipulation werden durch die eingesetzte Sensorik bestimmt und lassen sich mit dieser im Vergleich zur prototypischen Umsetzung erhöhen. Durch den modularen Aufbau des Programmiersystems wird der Aufwand zur Integration alternativer Sensorik für das Motion-Tracking auf ein Minimum reduziert. Die zweite Generation des Kinect-Sensors verspricht höhere Genauigkeiten bei einem Preis zwischen 100 und 200 €, sodass eine Applikation im räumlichen Programmiersystem vielversprechend erscheint. Alternativ können industrielle Bildverarbeitungssysteme mit höheren Genauigkeiten eingesetzt werden, die ggf. in Multi-Kamera-Ausführung sowohl die Fingergestenerkennung als auch das Tracking des Mobilgeräts abdecken. Auf Basis der kameragestützten Sensorik des Motion-Tracking lassen sich zudem automatische Kalibriermethoden implementieren. Diese ermöglichen eine breite Anwendbarkeit und rasche Inbetriebnahme des Programmiersystems schon innerhalb weniger Minuten, sind jedoch bezüglich realisierbarer Genauigkeiten zu erproben und evaluieren.

B) Erschließung industrieller Anwendungen

Zum allgemeinen industriellen Einsatz der natürlich-räumlichen Programmierung ist eine KMU-taugliche Umsetzung des Programmiersystems vorzunehmen. Die vorliegende Arbeit zeigt einen entsprechenden Ansatz auf, der jedoch mit dem Kinect-Sensor eine allgemeine Anwendbarkeit der gestenbasierten Eingabe nicht erreicht. Die aufgezeigte Integration alternativer Sensorik zum Motion-Tracking bzw. die Einbindung unterstützender Sensorik stellen vielversprechende Ansätze für die konkrete industrielle Umsetzung dar. Eine allgemeine Anwendbarkeit der AR-Simulation zur räumlichen Evaluation von Roboterprogrammen ist mithilfe der prototypischen Umsetzung gegeben. Je nach Bedarf lassen sich die gestenbasierte Programmdefinition und die Programmevaluation in der AR unabhängig voneinander einsetzen. Zur Erschließung industrieller Anwendungen und breiter Anwendbarkeit bei verschiedenen Nutzergruppen ist zudem die aufgabenorientierte Programmierung auf weitere Anwendungsszenarien zu erweitern.
Aktuell ist eine Umsetzung der aufgabenorientierten Programmierung für das robotergestützte MSG-Lichtbogenschweißen geplant. Durch räumliche Interaktion lässt sich eine einfache Definition von Zustellbewegungen und von komplexen Schweißtrajektorien über natürliche Bewegungen vornehmen. In diesem Zusammenhang wird aktuell die ergänzende Nutzung prozessintegrierter Sensorik zur Nahtfindung und -verfolgung evaluiert.

C) Zweiarmige Auslegung der Interaktion für Dual-Arm-Roboter

In den nächsten Jahren sind Produktinnovationen bezüglich der Anzeigetechnologie der AR zu erwarten. Neben dem Projekt „Google Glass“ unter Nutzung des OST-Prizips sind weitere Bril-

len zur VST-Darstellung von virtuellen Objekten im Kamerabild angekündigt. Es ist anzunehmen, dass die Brillen im Vergleich zu HMDs zu einer signifikanten Verbesserung der Ergonomie und des Tragekomforts beitragen. Da ein breiter Markt fokussiert wird, werden diese Anzeigeelemente voraussichtlich auch den Preis von vergleichbaren HMDs deutlich unterschreiten. Die Brille ermöglicht im Vergleich zum Handheld-Gerät eine größere Interaktionsfreiheit. In Bezug auf die räumliche Programmierung lässt sich eine Erweiterung auf Dual-Arm-Roboter vollziehen. Beidhändige Demonstrationen von Aufgaben oder Bewegungen ließen sich zur Programmierung einsetzen. Als Basis lässt sich die Programmiersprache auf zweiarmige Operationen erweitern. Krüger et al. zeigen einen diesbezüglichen Ansatz auf [184]. Bei der Integration der Brillen in das räumliche Programmiersystem muss jedoch beachtet werden, dass der potentielle Parallaxenfehler der OST-Technologie die Nutzbarkeit extrem einschränkt. Teilweise wurde von einzelnen Herstellen bereits angekündigt, dass zur Kompensation der Anzeigefehler in der räumlichen Darstellung eine Integration des Iris-Tracking in die Brillen angestrebt wird.

7.2.2 Das räumliche Programmiersystem als vernetztes Cyber-physisches System

Mit den umgesetzten Schnittstellen zur Robotersteuerung und zur Digitalen Fabrik lässt sich das Programmiersystem in seiner Gesamtheit als vernetztes Cyber-physisches System entsprechend der aktuellen Hightech-Strategie der Bundesregierung betrachten [1]. Das natürlich-räumliche Programmiersystem entspricht demgemäß dem geforderten Paradigmenwechsel in der Mensch-Technik-Interaktion [185]. Das Programmiersystem zeigt nicht nur neuartige Wege der Interaktion im industriellen Kontext auf, sondern trägt auch durch die Integration in die betrieblichen Informations- und Kommunikationssysteme zur hochflexiblen Bereitstellung und zum Austausch von Informationen bei. Durch den drahtlosen Austausch von Roboterprogrammen ist die Grundlage für eine horizontale Integration des Programmiersystems bereits gegeben. Weiterführende Potentiale liegen hier bspw. in der Entwicklung dezentraler Programmiermethoden. Vorstellbar ist bspw. eine anforderungsspezifische Kombination von prozessferner und prozessnaher Programmierung. Durch eine durchgängige, gemeinsame Datenbasis lässt sich der Umfang der ausgetauschten Informationen zwischen App und Robotersteuerung sowie zwischen App und Digitaler Fabrik steigern. Zu diesem Zweck lässt sich bspw. die Schnittstelle zur Realistischen Robotersimulation [171] integrieren. Darauf aufbauend ist im aktuellen Projekt pICASSO[2] eine Erweiterung des Datenaustauschs durch cloudbasierte Strukturen sowohl auf Algorithmen und Kinematikparameter als auch auf 3D-Modelle der Simulation vorgesehen. Weiterführend lässt sich durch die App die betriebliche und überbetriebliche Aufnahme, die Verarbeitung und der Austausch von Prozessinformationen im Sinne einer vertikalen Integration des Programmiersystems realisieren. Die erhobenen Daten stellen demgemäß Ansätze zur Optimierung der Organisation sowie zur Gestaltung von Aktivitäten entlang der gesamten Wertschöpfungskette dar.

[2] Vgl. http://www.projekt-picasso.de/ - 20.01.2014.

A Wahrnehmung der Augmented-Reality-Marker

Zur perspektivisch korrekten Darstellung der virtuellen Objekte im Kamerabild anhand der angestrebten Verfolgung von Flachmarkern sind verschiedene Schritte zu durchlaufen. Dazu gehören eine einmalige Kamerakalibrierung sowie folgende Schritte, die für jeden Frame durchgeführt werden:

1. Aufnahme des Kamerabilds,
2. Durchsuchen des Kamerabilds nach Markern,
3. Ermittlung der translatorischen und rotatorischen Verschiebung zwischen Marker und Kamerakoordinatensystem und
4. Zeichnen der virtuellen Objekte in Abhängigkeit von der Markerposition.

Die detaillierte Darstellung der Abfolge der einzelnen Schritte ist in Abbildung A.1 dargestellt und wird im Folgenden näher erläutert.

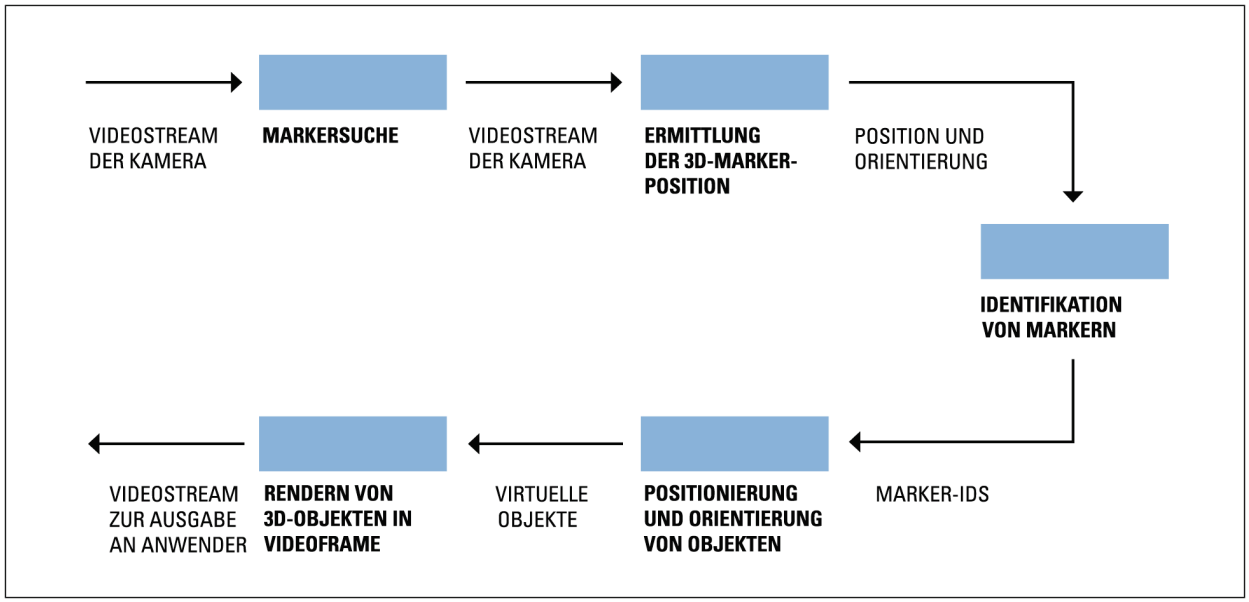

Abbildung A.1: Schritte des Marker-Tracking und der Anzeige in Anlehnung an ARToolkit[1].

[1] Vgl. http://www.hitl.washington.edu/artoolkit/ - 20.01.2014.

A.1 Markererfassung

Nach Aufnahme eines neuen Frames müssen potentielle Marker detektiert und identifiziert werden. Zu diesem Zweck werden Bildverarbeitungsalgorithmen verwendet, die anhand der Charakteristik der quadratischen Kontur die Bildkoordinaten der Markereckpunkte zurückgeben. Die Identifizierung der Marker erfolgt anhand des inneren Musters und bedient sich bei Binärmustern der Dekodierung bzw. bei Bildmustern des Template-Matching. In der praktischen Ausführung des Marker-Tracking bestehen verschiedene Ansätze, welche sich durch die benötigte Rechenzeit sowie Robustheit gegenüber verschiedenen Lichtverhältnissen und partieller Verdeckung unterscheiden. Alle Ansätze durchlaufen jedoch folgende Schritte [175]:

1. Vorverarbeitung
   (a) Thresholding
   (b) Entzerrung
   (c) Linienerkennung (line detection/line fitting)
   (d) Detektion von Eckpunkten
2. Ermittlung potentieller Marker
   (a) Aussortieren nicht-markerähnlicher Objekte
   (b) Schneller Test potentieller Marker
3. Identifikation und Dekodierung der Marker
   (a) Template-Matching oder Dekodierung

A.1.1 Ermittlung der 3D-Markerposition

Die ebenen Marker bilden durch die komplanaren Eckpunkte ein festes Welt-KS für die AR-Darstellung. Der Ursprung dieses KS liegt typischerweise in der Mitte oder an einem Eckpunkt des Markers. Die Marker-Ebene stellt typischerweise die XY-Ebene dar. Die 3D-Koordinaten der Eckpunkte des Markers im Marker-KS $\boldsymbol{p}_M$ lassen sich aus den Maßen des Markers berechnen und werden als bekannt vorausgesetzt. Aus dem vorangegangenen Schritt der Bildverarbeitung sind die Koordinaten der Eckpunkte $\boldsymbol{p}_C$ im Kamera-KS bekannt. Abbildung A.2 veranschaulicht die räumliche Beziehung der KS.
Eine Zuweisung übereinstimmender Markereckpunkte in beiden KS erfolgt mithilfe des Musters im Inneren des Markers. Anhand der Positionskorrespondenzen in Marker-KS und projektiver Bildebene kann nun die Ermittlung der Transformationsmatrix zwischen Kamera-KS und Marker-KS durchgeführt werden (s. Formeln A.1 und A.2). Zu diesem Zweck bestehen verschiedene Ansätze, um die extrinsischen Kameraparameter in Form der homogenen Transformationsmatrix $\boldsymbol{T}_E$ zu schätzen, unter anderem der Orthogonal-Iteration-Algorithmus [186] oder die Homographie [187].

$$\boldsymbol{p}_C = \boldsymbol{T}_E \cdot \boldsymbol{p}_M \tag{A.1}$$

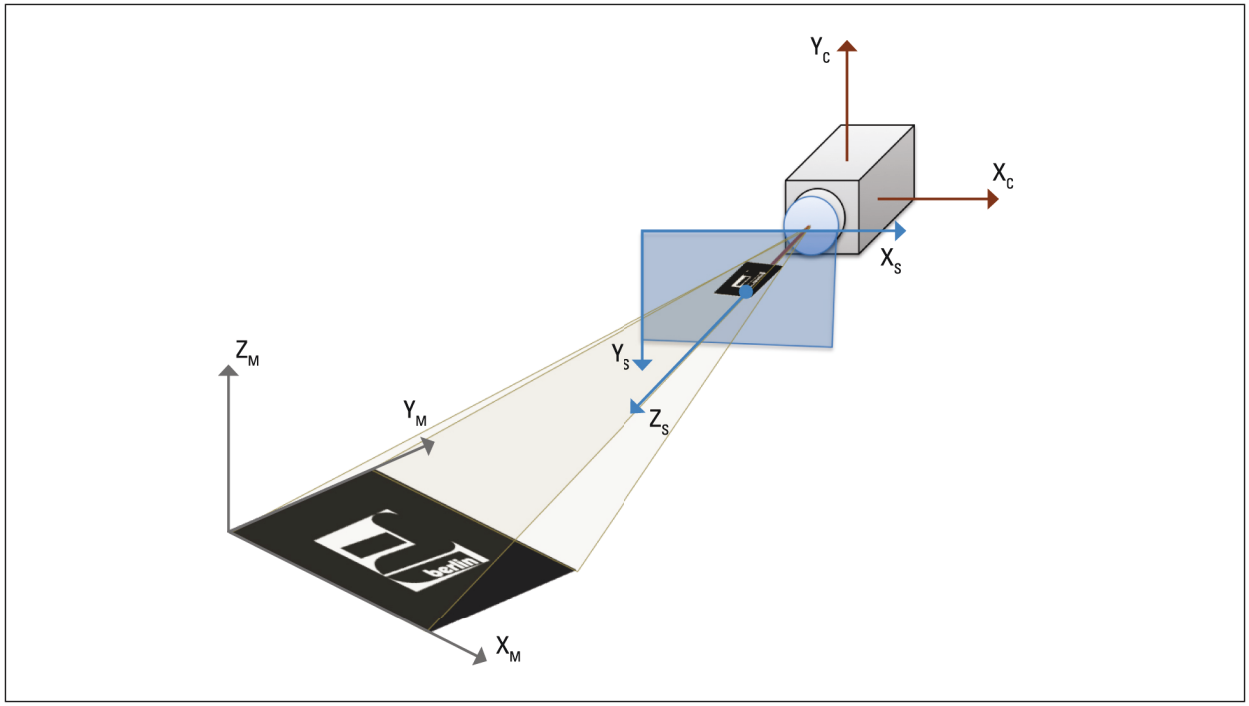

Abbildung A.2: Zusammenhang der Koordinatensysteme für die AR-Darstellung in Anlehnung an die Hochschule RheinMain[2].

In Matrixschreibweise entspricht dies:

$$\begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_M \\ y_M \\ z_M \\ 1 \end{pmatrix} \tag{A.2}$$

A.1.2 Darstellung der 3D-Objekte im Kamerabild

Mit bekannter Kamerapose im Kamera-KS lassen sich für jeden Frame 3D-Objekte im Marker-KS platzieren und im Kamerabild darstellen. Um dies zu realisieren, wird eine transparente Zeichenebene über den Videostream der Kamera gelegt. Die Darstellung der virtuellen 3D-Objekte in der 2D-Ebene erfolgt über eine Projektion anhand der intrinsischen Kameramatrix $\boldsymbol{P}$. Mithilfe dieses Rechenschritts können die Kamerakoordinaten in Bildschirmkoordinaten transformiert werden:

$$\begin{pmatrix} hx_S \\ hy_S \\ h \\ 1 \end{pmatrix} = \begin{pmatrix} f_X & 0 & c_X & 0 \\ 0 & f_Y & c_Y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_C \\ y_C \\ z_C \\ 1 \end{pmatrix} \tag{A.3}$$

f_X und f_Y beschreiben die Brennweite in Pixeln. c_X und c_Y beschreiben einen Referenzpunkt in der Bildmitte in Pixeln. h ist ein Skalierungsfaktor. Somit kann ein beliebiger 3D-Punkt, welcher

[2] Vgl. http://www.mi.hs-rm.de/~schwan/Projects/CG/CarreraCV/doku/extrinsisch/extrinsisch.htm - 20.01.2014.

im Marker-KS vorliegt, in die Bildschirmebene übertragen und dort auf der entsprechenden Zeichenebene über dem Kamerabild gerendert werden.

B Positionsbasierte Definition der Rotation

Der folgende Ansatz stellt die positionsbasierte Definition der Rotation des Endeffektors in Form der Quaternionendarstellung vor[1]. Abbildung B.1 veranschaulicht die Ableitung der Basisvektoren des Endeffektor-KS aus drei Positionen.

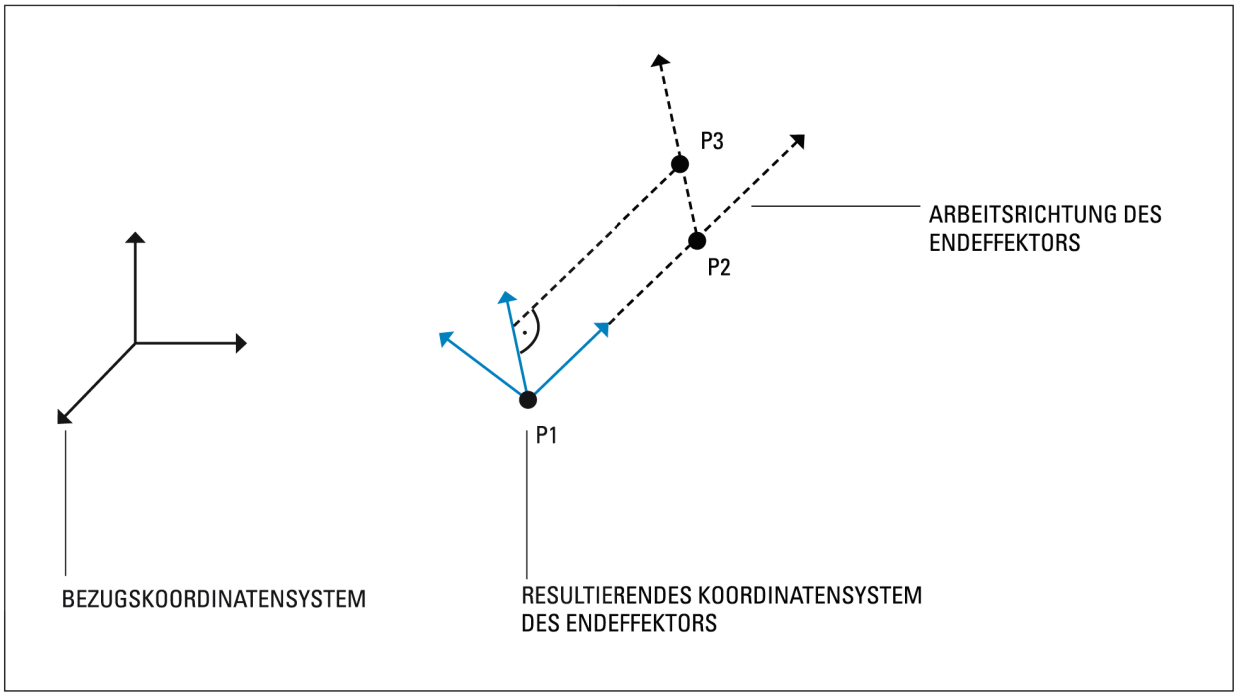

Abbildung B.1: Definition des Endeffektor-Koordinatensystems über drei Positionen.

Unter der Annahme, dass die Koordinaten der Punkte $P1$, $P2$, $P3$ im Bezugs-KS als bekannt vorausgesetzt werden können, lässt sich eine Lösung zur Ermittlung der Rotation zwischen Bezugs-KS und Effektor-KS analytisch vornehmen. An dieser Stelle wird ein Ansatz unter Verwendung der Quaterniondarstellung vorgestellt. Dabei wird die Rotation über zwei Drehungen Q_1 und Q_2 bestimmt. Die erste Drehung Q_1 dreht das Referenzkoordinatensystem zunächst in Richtung der Arbeitsrichtung des Endeffektors (hier Z-Achse). Die zweite Drehung rotiert das resultierende Koordinatensystem um die Arbeitsrichtung. Zur Ermittlung von Q_1 wird zunächst die Arbeitsrichtung des Endeffektors $\boldsymbol{v}$ anhand der Positionsvektoren der Punkte $\boldsymbol{p}_1$ und $\boldsymbol{p}_2$ bestimmt:

[1] Vgl. Kapitel 4.2.2.

$$\boldsymbol{v} = \boldsymbol{p}_2 - \boldsymbol{p}_1 \tag{B.1}$$

Anschließend wird die Rotationsachse $\boldsymbol{r}$ der Drehung bestimmt. Dabei handelt es sich um das Kreuzprodukt der Z-Achse des Referenzkoordinatensystems $\boldsymbol{z}_{ref}$ und der Arbeitsrichtung des Endeffektors $\boldsymbol{v}$:

$$\boldsymbol{r} = \boldsymbol{z}_{ref} \times \boldsymbol{v} \tag{B.2}$$

Der Drehwinkel α_1 um die Rotationsachse $\boldsymbol{r}$ wird wie folgt ermittelt:

$$\alpha_1 = \cos^{-1}\left(\frac{\boldsymbol{z}_{ref} \cdot \boldsymbol{v}}{|\boldsymbol{z}_{ref}| \cdot |\boldsymbol{v}|}\right) \tag{B.3}$$

Aus dem normierten Rotationsvektor $\boldsymbol{r}$ und Rotationswinkel lässt sich die erste Quaternion bilden als $Q_1(r_X, r_Y, r_Z, \alpha_1)$.
Anhand der Einheitsquaternion kann die neue X-Achse des Hilfskoordinatensystems $\boldsymbol{x}_{Hilf}$ folgendermaßen berechnet werden:

$$\boldsymbol{x}_{Hilf} = \boldsymbol{Q}_1 \cdot \boldsymbol{x}_{Ref} \cdot \boldsymbol{Q}_1^{-1} \tag{B.4}$$

Die Rotationsachse der zweiten Drehung Q_2 stellt der Vektor $\boldsymbol{v}$ dar. Zur Ermittlung des Drehwinkels wird die resultierende X-Achse $\boldsymbol{x}_{Ziel}$ anhand einer Orthogonalprojektion des Verbindungsvektors $\boldsymbol{p}$ zwischen den Punkten $P1$ und $P3$ berechnet:

$$\boldsymbol{x}_{Ziel} = \boldsymbol{p} - \left[\left(\boldsymbol{p} \cdot \frac{\boldsymbol{v}}{|\boldsymbol{v}|}\right) \cdot \frac{\boldsymbol{v}}{|\boldsymbol{v}|}\right] \tag{B.5}$$

Der Drehwinkel ergibt sich aus dem Winkel zwischen der X-Achse des Hilfskoordinatensystems $\boldsymbol{x}_{Hilf}$ und der X-Achse des Zielkoordinatensystems $\boldsymbol{x}_{Ziel}$:

$$\alpha_2 = \cos^{-1}\left(\frac{\boldsymbol{x}_{Hilf} \cdot \boldsymbol{x}_{Ziel}}{|\boldsymbol{x}_{Hilf}| \cdot |\boldsymbol{x}_{Ziel}|}\right) \tag{B.6}$$

Die Quaternion der zweiten Drehung bildet sich analog als $Q_2(v_x, v_y, v_z, \alpha_2)$.

Die gesuchte Gesamtrotation Q_{ges} ergibt sich in Quaterniondarstellung durch die Multiplikation der beiden Quaternionen:

$$Q_{ges} = Q_1 \cdot Q_2 \tag{B.7}$$

C Definition der vereinheitlichten Roboterschnittstelle

C.1 Funktionsübersicht

Funktion	**Beschreibung**	**Antwort**
Stop	Stoppen von Achsen/Bewegungen	Done
Grip	Zustandsänderung des Greifers	Done
DriveAxes	Verfahren bestimmter Achsen starten	-
HomeAxes	Home-Position von Achsen anfahren	Done
SetPose	Eine bestimmte Pose mit weiteren Parametern ausstatten	Done
SetFrame	Eine Pose senden, um sie über movePath anzusprechen	Done
GetCurrentPosition	Abfragen der aktuellen Pose	GetCurrentPosition
MovePath	Alle Posen in best. Intervall anfahren	Done
ChangeTool	Wechseln des Greifers	Done
Init	Initialisierung	Init
Done	Befehl abgeschlossen, Bestätigung erfolgt in der gleichen Reihenfolge, wie die Befehle gesendet wurden.	-
DriveVector	Verfahren im Welt-KS starten/ändern	-
ReadInOut	Einlesen von Ein- und Ausgängen	ReadInOut
SetOut	Setzen von Ausgängen	Done

Tabelle C.1: Funktionsübersicht.

C.2 Funktionsbeschreibung und Paketaufbau

Typ	Name	Beschreibung
uint8	MAX_ACHSEN	Maximalzahl der unterstützten Achsen

Tabelle C.2: Konstante.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	0	ID
2	int8[MAX_ACHSEN]	axes	[0,1]	1: Stoppen der jeweiligen Achse 0: Zustand unverändert

Tabelle C.3: Stop.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	1	ID
2	uint8	grip		0: Greifer unverändert 1: Greifer zu n: diskrete Greiferstellung oder Kraft 255: Greifer auf

Tabelle C.4: Grip.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	2	ID
2	int16[MAX_ACHSEN]	axes		n: $mrad/s$

Tabelle C.5: MoveAxes.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	3	ID
2	int8[MAX_ACHSEN]	axes	[0,1]	1: Homeposition der Achse anfahren 0: Zustand unverändert

Tabelle C.6: HomeAxes.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	4	ID
2	uint16	posId		Eindeutige Nummer dieser Pose (s. movePath)
4	int16	frame		Referenzierte Pose beschrieben durch einen Frame (s. SetFrame) $frame < 0$: Referenz-Frame aus der Steuerung $frame > 0$: Selbstdefinierte Koordinaten
6	uint16	speed		Geschwindigkeit mm/s
8	uint16	acc		Beschleunigung mm/s^2
10	uint16	flags		Bit [2,1,0]:Bewegungsart 0: PTP [000] 1: LIN [001] 2: CIRC [010] 3: Spline [011]
12	uint16	dist		Überschleifabstand [mm], ab dem ein Überschleifen zur Sollpose möglich wird
14	uint16	waitMs		Wartezeit [ms] bis der Greiferzustand geändert wird, nachdem die Pose erreicht ist.
16	uint8	gripper		Greiferstatus an dieser Pose (s. Grip)

Tabelle C.7: SetPose.

Adresse	**Typ**	**Name**	**Wert**	**Beschreibung**
0	uint16	id	5	ID
2	int16	frameId		Eindeutige Nummer des Frames
4	int32[6]	pos		kartesisch: [x,y,z,a,b,c] x,y,z: Position 10^{-6} m a,b,c: Orientierung 10^{-6} rad Achskoordinaten: [MAX_ACHSEN], Orientierung 10^{-6} rad
28	int16	baseFrame		Referenz-Frame
30	uint8	type		Bit 0: Bewegungsart 0: Absolut 1: Relativ Bit [2,1]: Eulerwinkel (a,b,c) 00: RX, RY, RZ 01: RZ, RY, RX 10: RZ,RX',RZ" (Eulerwinkel) Bit [3]: Posenangaben 0: Kartesisch 1: Achskoordinaten

Tabelle C.8: SetFrame.

Adresse	**Typ**	**Name**	**Wert**	**Beschreibung**
0	uint16	id	6	ID
2	int16	frameId		Eindeutige Nummer des Referenz-Frames
4	uint8	type		Bit 0: 0 = Antwort 1 = Anforderung Bit 1: 0 = Kartesisch 1 = Achswerte
5	uint8	length		Anzahl der Rückgabewerte
6	int32[length]	pos		
6+4 *length				

Tabelle C.9: GetCurrentPose.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	7	ID
2	uint16	length		0: [points[0]...points[1]] => n=2 n: Anzahl der folgenden Punkte
4	uint16[n]	points		Nummer des Punkts aus SetPose

Tabelle C.10: MovePath.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	8	ID
2	uint8	tool		Ablegen des aktuellen Werkzeugs und Aufnehmen des Werkzeugs mit der Nummer „tool“, Ändern des TCP

Tabelle C.11: ChangeTool.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	9	ID
2	uint8	version		Protokollversion

Tabelle C.12: Init.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	10	ID
2	uint16	requestId		PaketTyp des Pakets, das bestätigt wird
4	int16	status		-1 : Fehler 0 : Ok 1 : Not Implemented

Tabelle C.13: Done.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	11	ID
2	uint16	speed		Geschwindigkeit [mm/s]
4	int32[6]	vector		Vektor zum Verfahren aus X,Y,Z, a,b,c: [-1,00; 1,00]*10^3

Tabelle C.14: MoveVector/DriveVector.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	12	ID
2	uint16	port		Eingangs/Ausgangs-Id
4	uint8	type		Bit 0: 0 = Antwort 1 = Anforderung Bit 1: 0 = Binär 1 = Analog Bit 2: 0 = Eingang 1 = Ausgang
5	int16	data		bei binär: 0 oder 1 bei analog zwischen -1,00 und 1,00 $*10^6$ (normiert auf Ausgangsspannung $\pm 10\ V$)

Tabelle C.15: ReadInOut.

Adresse	Typ	Name	Wert	Beschreibung
0	uint16	id	13	ID
2	uint16	port		Ausgangs-ID
4	uint8	type		Bit 1: 0 = Binär 1 = Analog
5	int16	data		bei binär: 0 oder 1 bei analog zwischen -1,00 und 1,00 $*10^6$ (normiert auf Ausgangsspannung $\pm 10\ V$)

Tabelle C.16: SetOut.

D Datenbankmodell

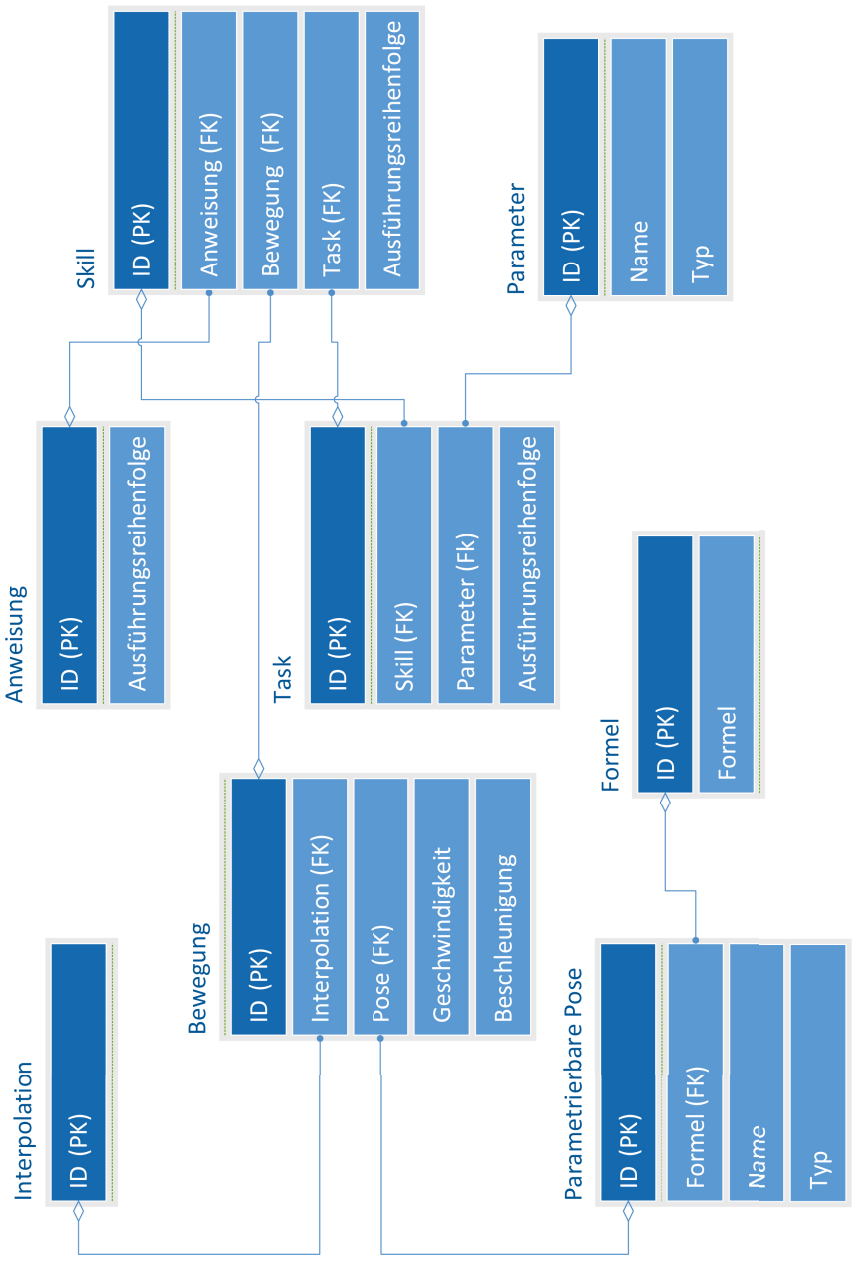

Abbildung D.1: Datenbankmodell für die aufgabenorientierte Programmierung, modelliert in IDEF1X-Notation.

E Berechnungen zur Rotation

Mithilfe der folgenden Berechnungen lässt sich in Anlehnung an [188] auf Basis verschiedener Konventionen zur Darstellung der Orientierung die 3×3 Rotationsmatrix aufstellen und umgekehrt. Die verwendete Funktion $atan2$ ist eine Arcustangensfunktion mit zwei Argumenten, welche eine Fallunterscheidung für unterschiedliche Quadranten überflüssig macht.

E.1 Vektor zu Rotationsmatrix

E.1.1 ZY'Z"-Euler

Die Z'Y"Z-Euler-Konvention $\boldsymbol{R}_{ZY'Z''}(\alpha, \beta, \gamma)$ bezeichnet die Rotation um ein mitdrehendes Koordinatensystem mit der Reihenfolge $\boldsymbol{R}_Z(\alpha)$, $\boldsymbol{R}_Y(\beta)$ und $\boldsymbol{R}_Z(\gamma)$.

$$\text{Rotation (ZY'Z"-Euler):} \begin{pmatrix} \alpha \\ \beta \\ \gamma \end{pmatrix} \rightarrow \boldsymbol{R}_{ZY'Z''}(\alpha, \beta, \gamma) \tag{E.1}$$

$$\begin{aligned} \boldsymbol{R}_{ZY'Z''}(\alpha, \beta, \gamma) =& \boldsymbol{R}_Z(\alpha) \cdot \boldsymbol{R}_Y(\beta) \cdot \boldsymbol{R}_Z(\gamma) \\ \boldsymbol{R}_{ZY'Z''}(\alpha, \beta, \gamma) =& \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix} \\ & \begin{pmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{E.2}$$

E.1.2 Roll-Pich-Yaw

Die RPY-Konvention $\boldsymbol{R}_{XYZ}(\gamma, \beta, \alpha)$ bezeichnet die Rotation um ein raumfestes Referenzkoordinatensystem mit der Reihenfolge $\boldsymbol{R}_X(\gamma)$ (roll), $\boldsymbol{R}_Y(\beta)$ (pitch) und $\boldsymbol{R}_Z(\alpha)$ (yaw).

$$\text{Rotation (RPY):} \begin{pmatrix} \gamma \\ \beta \\ \alpha \end{pmatrix} \rightarrow \boldsymbol{R}_{XYZ}(\gamma, \beta, \alpha) \tag{E.3}$$

$$\begin{aligned}\boldsymbol{R}_{XYZ}(\gamma,\beta,\alpha) =& \boldsymbol{R}_Z(\alpha)\cdot\boldsymbol{R}_Y(\beta)\cdot\boldsymbol{R}_X(\gamma)\\ \boldsymbol{R}_{XYZ}(\gamma,\beta,\alpha) =& \begin{pmatrix}\cos(\alpha) & -\sin(\alpha) & 0\\ \sin(\alpha) & \cos(\alpha) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\cos(\beta) & 0 & \sin(\beta)\\ 0 & 1 & 0\\ -\sin(\beta) & 0 & \cos(\beta)\end{pmatrix}\\ & \begin{pmatrix}1 & 0 & 0\\ 0 & \cos(\gamma) & -\sin(\gamma)\\ 0 & \sin(\gamma) & \cos(\gamma)\end{pmatrix}\end{aligned} \tag{E.4}$$

Zu beachten ist, dass bei der Drehung um ein raumfestes KS die Matrixmultiplikationen von rechts nach links durchgeführt werden müssen.

E.1.3 Quaternionen

Quaternionen bestehen aus vier Skalarwerten: einem realen und drei imaginären Anteilen. Durch eine Quaternion (Q) lassen sich Rotationen im 3D-Raum durch einen Vektor $\boldsymbol{r}$ und einen Rotationswinkel ϕ um diesen Vektor repräsentieren. Aus den beiden Anteilen lässt sich mit geringem Rechenaufwand die entsprechende Einheitsquaternion $Q(q_X, q_Y, q_Z, q_W)$ ableiten.

$$\text{Rotation (Q):} \begin{pmatrix}q_X\\ q_Y\\ q_Z\\ q_W\end{pmatrix} \rightarrow \boldsymbol{R}(Q) \tag{E.5}$$

$$\boldsymbol{R}(Q) = \begin{pmatrix}1-2\cdot(q_Y^2+q_Z^2) & 2\cdot(q_X\cdot q_Y - q_W\cdot q_Z) & 2\cdot(q_X\cdot q_Z - q_Y\cdot q_W)\\ 2\cdot(q_X\cdot q_Y + q_Z\cdot q_W) & 1-2\cdot(q_X^2+q_Z^2) & 2\cdot(q_Y\cdot q_Z - q_X\cdot q_W)\\ 2\cdot(q_X\cdot q_Z + q_Y\cdot q_W) & 2\cdot(q_Y\cdot q_Z - q_X\cdot q_W) & 1-2\cdot(q_X^2+q_Y^2)\end{pmatrix} \tag{E.6}$$

E.2 Rotationsmatrix zu Vektor

E.2.1 ZY'Z"-Euler

$$\text{Rotation (ZY'Z"-Euler):} \boldsymbol{R} = \begin{pmatrix}r_{1,1} & r_{1,2} & r_{1,3}\\ r_{2,1} & r_{2,2} & r_{2,3}\\ r_{3,1} & r_{3,2} & r_{3,3}\end{pmatrix} \rightarrow \begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix} \tag{E.7}$$

Allgemein gilt:

$$\beta = atan2\left(\sqrt{r_{3,1}^2 + r_{3,2}^2}, r_{3,3}\right) \tag{E.8}$$

Zur Behandlung von Singularitäten wird bei der Ermittlung der Euler-Winkel aus der Rotationsmatrix eine Fallunterscheidung durchgeführt. Wenn β gleich null ist, gilt:

$$\begin{aligned} \alpha &= 0 \\ \gamma &= atan2\left(-r_{1,2}, r_{1,1}\right) \end{aligned} \tag{E.9}$$

Wenn β gleich 180° ist, gilt:

$$\begin{aligned} \alpha &= 0 \\ \gamma &= atan2\left(r_{1,2}, -r_{1,1}\right) \end{aligned} \tag{E.10}$$

Für alle anderen Fälle gilt:

$$\begin{aligned} \alpha &= atan2\left(\frac{r_{2,3}}{\sin\left(\beta\right)}, \frac{r_{1,3}}{\sin\left(\beta\right)}\right) \\ \gamma &= atan2\left(\frac{r_{3,2}}{\sin\left(\beta\right)}, \frac{-r_{3,1}}{\sin\left(\beta\right)}\right) \end{aligned} \tag{E.11}$$

E.2.2 Roll-Pich-Yaw

$$\text{Rotation (RPY): } \boldsymbol{R} = \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} \gamma \\ \beta \\ \alpha \end{pmatrix} = \begin{pmatrix} roll \\ pitch \\ yaw \end{pmatrix} \tag{E.12}$$

Allgemein gilt:

$$\beta = atan2\left(-r_{3,1}, \sqrt{r_{1,1}^2 + r_{2,1}^2}\right) \tag{E.13}$$

Zur Behandlung von Singularitäten wird bei der Ermittlung der RPY-Winkel ebenfalls eine Fallunterscheidung durchgeführt. Eine Singularität liegt vor, wenn $\cos\left(\beta\right)$ gleich null ist.

Wenn β gleich $\pm 90°$ ist, gilt:

$$\begin{aligned} \alpha &= 0 \\ \gamma &= \pm atan2\left(r_{1,2}, r_{2,2}\right) \end{aligned} \tag{E.14}$$

Für alle anderen Fälle gilt:

$$\begin{aligned} \alpha &= atan2\left(\frac{r_{2,1}}{\cos\left(\beta\right)}, \frac{r_{1,1}}{\cos\left(\beta\right)}\right) \\ \gamma &= atan2\left(\frac{r_{3,2}}{\cos\left(\beta\right)}, \frac{r_{3,3}}{\cos\left(\beta\right)}\right) \end{aligned} \tag{E.15}$$

E.2.3 Quaternionen

Bei der Ermittlung der Quaternion aus einer Rotationsmatrix wird zunächst die Spur der Rotationsmatrix berechnet. Da diese für die weitere Berechnung positiv sein muss, werden abhängig von den Vorzeichen der Diagonaleinträge der Matrix Fallunterscheidungen vorgenommen.

Ermittle die Spur der Matrix mit verschiedenen Vorzeichenkombinationen und ermittle das Maximum j:

$$\begin{aligned} tq_1 &= 1 + r_{1,1} + r_{2,2} + r_{3,3} \\ tq_2 &= 1 + r_{1,1} - r_{2,2} - r_{3,3} \\ tq_3 &= 1 - r_{1,1} + r_{2,2} - r_{3,3} \\ tq_4 &= 1 - r_{1,1} - r_{2,2} - r_{3,3} \end{aligned} \tag{E.16}$$

$$j = \max(tq_1, tq_2, tq_3, tq_4) \tag{E.17}$$

$$s = \sqrt{\frac{0,25}{j}} \tag{E.18}$$

Führe Fallunterscheidungen abhängig von j durch. Wenn $j = tq_1$:

$$\begin{aligned} q_w &= tq_1 \cdot s \\ q_x &= (r_{3,2} - r_{2,3}) \cdot s \\ q_y &= (r_{1,3} - r_{3,1}) \cdot s \\ q_z &= (r_{2,1} - r_{1,2}) \cdot s \end{aligned} \tag{E.19}$$

andernfalls, wenn $j = tq_2$:

$$\begin{aligned} q_w &= (r_{3,2} - r_{2,3}) \cdot s \\ q_x &= tq_2 \cdot s \\ q_y &= (r_{2,1} + r_{1,2}) \cdot s \\ q_z &= (r_{1,3} + r_{3,1}) \cdot s \end{aligned}$$

andernfalls, wenn $j = tq_3$:

$$\begin{aligned} q_w &= (r_{1,3} - r_{3,1}) \cdot s \\ q_x &= (r_{2,1} + r_{1,2}) \cdot s \\ q_y &= tq_3 \cdot s \\ q_z &= (r_{3,2} + r_{2,3}) \cdot s \end{aligned} \tag{E.21}$$

andernfalls, wenn $j = tq_4$:

$$\begin{aligned} q_w &= (r_{2,1} - r_{1,2}) \cdot s \\ q_x &= (r_{3,1} + r_{1,3}) \cdot s \\ q_y &= (r_{2,3} + r_{3,2}) \cdot s \\ q_z &= tq_4 \cdot s \end{aligned} \tag{E.22}$$

F Spline-Interpolation

F.1 Konturbeschreibung mit B-Splines

F.1.1 Allgemeine Berechnungsvorschriften

B-Splines dienen in der vorliegenden Arbeit zur Darstellung der Handkontur auf Basis vorgegebener Kontrollpunkte. Für einen beliebigen Punkt auf der B-Spline-Kurve gilt allgemein:

$$\begin{pmatrix} x(s) \\ y(s) \end{pmatrix} = \boldsymbol{B}(s) \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \tag{F.1}$$

$\boldsymbol{x}$ und $\boldsymbol{y}$ sind $n \times 1$ Spaltenvektoren, die jeweils die X- und Y-Koordinaten der Kontrollpunkte beinhalten. $\boldsymbol{B}(s)$ ist eine 2×2 Matrix, deren Einträge die B-Spline-Basisfunktionen darstellen. Diese Basisfunktionen werden an der Stelle s ausgewertet. Bei den B-Splines entsteht eine mehrsegmentige Kurve [178], welche sich folgendermaßen beschreiben lässt:

$$\boldsymbol{C}(s) = \sum_{i=0}^{n} \boldsymbol{p}_i \cdot N_{i,k}(s) \text{ mit } s \in [0, n-k+2] \tag{F.2}$$

$\boldsymbol{p}_i$ sind die $n+1$ Stützpunkte und $\boldsymbol{C}(s)$ der interpolierte Punkt an der Stelle s. $N_{i,k}(s)$ sind segmentweise aufgebaute B-Spline-Basisfunktionen der Ordnung k bzw. Polynome vom Grad $k-1$. Zumeist werden für die B-Splines kubische Polynome vom Grad vier genutzt. Zu den einzelnen Segmenten gehören Parameterwerte, welche den Trägervektor $\boldsymbol{t}$ bilden. Für $N_{i,k}(s)$ gilt folgende rekursive Berechnungsvorschrift:

$$\begin{aligned} N_{i,k}(s) &= \frac{s - t(i)}{t(i+k-1) - t(i)} \cdot N_{i,k-1}(s) + \frac{t(i+k) - s}{t(i+k) - t(i+1)} \cdot N_{i+1,k+1}(s) \\ \text{mit } t : [0, n+k] \subset \mathbb{N} \rightarrow [0, n-k+2] \subset \mathbb{N}, T(i) &= \begin{cases} 0, & i-k \\ i-k+1, & k \leq i \leq n \\ n-k+2, & i > n \end{cases} \\ \text{und } N_{i,1}(s) &= \begin{cases} 1, & t(i) \leq s \leq t(i+1), t(i) < t(i+1) \\ 0, & \text{sonst} \end{cases} \end{aligned} \tag{F.3}$$

Wird die Gleichung der Formel F.3 auf die 2D-Konturbeschreibung des Hand-Tracking übertragen, resultiert $\boldsymbol{C}(S) = \begin{pmatrix} x(s) & y(s) \end{pmatrix}^T$ und $\boldsymbol{p}_i = \begin{pmatrix} x_i & y_i \end{pmatrix}^T$. Es gilt:

$$\begin{aligned} \begin{pmatrix} x(s) \\ y(s) \end{pmatrix} &= \sum_{i=0}^{n} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \cdot N_{i,k}(s) \\ &= \begin{pmatrix} x_0 \cdot N_{0,k}(s) + x_1 \cdot N_{1,k}(s) + \cdots + x_n \cdot N_{n,k}(s) \\ y_0 \cdot N_{0,k}(s) + y_1 \cdot N_{1,k}(s) + \cdots + y_n \cdot N_{n,k}(s) \end{pmatrix} \\ &= \begin{pmatrix} N_{0,k}(s) & N_{1,k}(s) & \cdots & N_{n,k}(s) \\ N_{0,k}(s) & N_{1,k}(s) & \cdots & N_{n,k}(s) \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \\ y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix} \\ &= \boldsymbol{B}(s) \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} \end{aligned} \tag{F.4}$$

Für die Ermittlung der senkrecht zur Kontur verlaufenden Messlinien des Tracking-Algorithmus lässt sich die Ableitung der B-Spline-Kurve an dem zugehörigen Messpunkt bestimmen [189]. Es gilt:

$$\boldsymbol{C}(s)' = \sum_{i=0}^{n} \boldsymbol{p}_i \cdot N_{i,k}(s)' \text{ mit } s \in [0, n-k+2] \tag{F.5}$$

Zur Berechnung der Ableitung des B-Splines lässt sich wiederum eine rekursive Berechnungsvorschrift anwenden:

$$N_{i,k}(s)' = \frac{k-1}{t(i+k-1) - T(i)} \cdot N_{i,k-1}(s) - \frac{k-1}{t(i+k) - t(i+1)} \cdot N_{i+1,k-1}(s) \tag{F.6}$$

F.1.2 Initialisierung der B-Splines

Die Verwendung der B-Splines dient im Programmiersystem der Darstellung der Handkontur auf dem Bildschirm sowie der Bestimmung der senkrecht zur Kontur verlaufenden Messlinien. Formel F.2 erfordert die rekursive Berechnung der Basisfunktionen $N_{i,k}(s)$. Der Parameter k ist als Grad der B-Splines eine Konstante. i beschreibt den aktuellen Kontrollpunkt, dessen Werte von i zwischen 0 und n_p liegen.
Zur Erzeugung der Kontur muss s mit diskreten Werten beschrieben werden. Deren Anzahl n_s bestimmt den Detaillierungsgrad der dargestellten Spline-Kurve. Dies entspricht der Abbildung $S : [0, n_s - 1] \subset \mathbb{N} \rightarrow [0, n_p - k + 2] \subset \mathbb{R}$. Für den Definitionsbereich gilt $j \in [0, n_s] \subset \mathbb{N}$. Wird s als eindimensionale Datenstruktur umgesetzt, ergibt sich für die rekursive Berechnung der

Basisfunktionen:

$$N_{i,k}(j) = \frac{S(j) - t(i)}{T(i+k-1) - t(i)} \cdot N_{i,k-1}(j) + \frac{t(i+k) - s(j)}{t(i+k) - t(i+1)} \cdot N_{i+1,k-1}(j)$$
$$\text{und} \quad N_{i,1}(j) = \begin{cases} 1, & t(i) \leq S(j) \leq t(i+1), t(i) < t(i+1) \\ 0, & \text{sonst} \end{cases} \tag{F.7}$$

und deren Abbildung:

$$N_{i,k}(j)' = \frac{k-1}{t(i+k-1) - t(i)} \cdot N_{i,k-1}(j) - \frac{k-1}{t(i+k) - t(i+1)} \cdot N_{i+1,k-1}(j) \tag{F.8}$$

i und j sind ganzzahlige Variablen, deren Werte als bekannt vorausgesetzt werden. Derart lassen sich die Gewichtungsfaktoren der Basisfunktionen einmalig initialisieren und in einer zweidimensionalen Struktur $(N[j][i])$ speichern.
Die Abbildung des Trägervektors $\boldsymbol{t} : [0, n_p + k] \rightarrow [0, n_p - k + 2]$ wird ebenfalls mithilfe einer eindimensionalen Datenstruktur beschrieben. Es folgt eine Darstellung des Algorithmus zur Initialisierung der B-Splines basierend auf den in F.7 und F.8 eingeführten Formeln:

Algorithmus zur Initialisierung der B-Splines

Für alle j in N[j][i] und N'[j][i]:

- Einträge in Zeilen N[j][...] und N'[j][...] zu 0 setzen
- Finde t, für das gilt: $t(t) \leq s(j)$ mit $t(t) < t(t+1)$
- N[j][t] zu 1 setzen
- Für alle k von 2 bis K (Grad der B-Splines):
  - Index des ersten Eintrags in k-ter Stufe bestimmen, der nicht 0 ist: $iStart = max(0, t - k + 1)$
  - Index des letzten Eintrags in k-ter Stufe bestimmen, der nicht 0 ist: $iStop = min(t, n_p + K - k)$
  - Speicher $iStart$ und $iStop$
  - Für alle i von $iStart$ bis $iStop$:
    - Ermittle Brüche der Gleichungen in den Formeln F.7 und F.8
    - Bestimme N'[j][...] und im Anschluss N[j][...]

F.2 Beschreibung von Freiformkurven mit NURBS

Zur Beschreibung und Übertragung von freien Bewegungen der Hand des Anwenders auf den Industrieroboter werden NURBS eingesetzt. Aufbauend auf den B-Splines ist bei den NURBS eine weitere Anpassung des Kurvenverlaufs (Kurvenpunkte C) durch die Nutzung variabler Intervalllängen möglich. Weiterführende Möglichkeiten zur Anpassung des Kurvenverlaufs bietet ein Gewichtungsfaktor. Durch den Gewichtungsfaktor w lässt sich der Einfluss einzelner Kontrollpunkte p parametrieren, sodass sich nach [179] für den Kurvenverlauf folgende Formel ergibt:

$$\boldsymbol{C}(s) = \frac{\sum_{i=0}^{n} \boldsymbol{p}_i \cdot N_{i,k}(s) \cdot w_i}{\sum_{i=0}^{n} N_{i,k}(s) \cdot w_i} \text{ mit } a \leq s \leq b \tag{F.9}$$

Unter der Voraussetzung $a = 0$, $b = 1$ und $w_i > 0$ ergibt sich:

$$\begin{aligned} \boldsymbol{C}(s) &= \sum_{i=0}^{n} R_{i,k}(s) \cdot \boldsymbol{p}_i \\ \text{mit: } R_{i,k}(s) &= \frac{N_{i,k}(s) \cdot w_i}{\sum_{j=0}^{n} N_{j,k}(s) \cdot w_j} \end{aligned} \tag{F.10}$$

Mithilfe dieser Gleichung lässt sich der B-Spline bei Kenntnis der Kontrollpunkte individuell über die zusätzlichen Parameter anpassen. Im vorliegenden Programmiersystem ergibt sich im Rahmen der Interpolation jedoch das Problem, dass der Kurvenverlauf anhand von gegebenen Bahnpunkten abstrahiert werden soll. Zu diesem Zweck wird eine Methode aus [179] gewählt, welche im ersten Schritt die Bildung eines Hilfsvektors zur Ermittlung der Kontrollpunkte vorschlägt. Der sogenannte „Hilfsknotenvektor“ berücksichtigt die Lage und Abstände der Kurvenpunkte. Auf deren Basis erfolgt im nächsten Schritt eine Gleichverteilung der Kontrollpunkte. Nach [179] werden im Folgenden die benötigten Algorithmen derartig kombiniert, dass sich eine Matrixdarstellung der Basisfunktionen ergibt. Die entsprechende Umformung der Gleichung in Formel F.10 mit auszuwertenden Hilfsknotenpunkten führt in Matrixschreibweise zu Formel F.11. Eine Lösung des daraus resultierenden Gleichungssystems (s. Formel F.11) zur Beschreibung des Kurvenverlaufs kann durch den Gaußalgorithmus ohne Pivotisierung erfolgen.

$$\begin{pmatrix} R_{0,k}(s_0) & \cdots & R_{i,k}(s_0) \\ \vdots & \ddots & \vdots \\ R_{0,k}(s_k) & \cdots & R_{i,k}(s_k) \end{pmatrix} \cdot \begin{pmatrix} p_0 \\ \vdots \\ p_i \end{pmatrix} = \begin{pmatrix} C_0 \\ \vdots \\ C_i \end{pmatrix} \tag{F.11}$$

Durch Invertierung der Matrix $R_{i,p}$ lassen sich die Kontrollpunkte $\boldsymbol{p}$ ermitteln. Zu diesem Zweck erfolgt eine LR-Zerlegung ohne Pivotisierung.
Zur Definition des Geschwindigkeitsprofils des Splines wird zunächst die Kurvenlänge bestimmt. Darauf wird der Bahnverlauf in eine ganzzahlige Anzahl von Abschnitten aufgeteilt. Für diese werden abschließend mithilfe des De-Casteljau-Algorithmus einzelne Bahnpunkte in festen Interpolationsschritten zur Realisierung des Geschwindigkeitsprofils ermittelt.

Literaturverzeichnis

[1] E. Geisberger und M. Broy, Hrsgg., *agendaCPS: Integrierte Forschungsagenda Cyber-Physical Systems*, Serie acatech Studie, Berlin und Heidelberg: Springer Verlag, 2012.

[2] VDI/VDE-Gesellschaft. (2009) Automation 2020 Bedeutung und Entwicklung der Automation bis zum Jahr 2020: Thesen und Handlungsfelder. Online verfügbar unter: http://www.vdi.de/nc/technik/fachthemen/mess-und-automatisierungstechnik/fachbereiche/automation-2020/?cid=62458&did=10435&sechash=dadd8ff6 - 20.01.2014.

[3] IFR Statistical Department, *World Robotics 2011 Industrial Robots*, Frankfurt: VDMA Robotik, 2011.

[4] B. Denkena, H. Wörn, R. Apitz, R. Bischoff, B. Hein, P. Kowalski, D. Mages und H. Schuler, „Roboterprogrammierung in der Fertigung: Einfache Roboterprogrammierung für die Produktion von morgen (Ergebnisse des IRoProg Projekts)," *wt Werkstattstechnik online*, Vol. 2005, Nr. 09-2005, S. 656–660, 2005.

[5] H. Armbruster, E. Kirner und S. Kinkel, „Neue Kundengruppen für Industrieroboter: wo liegen unausgeschöpfte Anwendungspotenziale für Roboter im deutschen Verarbeitenden Gewerbe?" Fraunhofer Institut für System- und Innovationsforschung (ISI), Mitteilungen aus der Produktionsinnovationserhebung 38, 2006.

[6] S. Forge und C. Blackman. (2010) A helping hand for Europe: the competitive outlook for the EU robotics industry. Online verfügbar unter: http://ipts.jrc.ec.europa.eu/publications/pub.cfm?id=3781 - 20.01.2014.

[7] Bremer Institut für Produktion und Logistik und ISEIC Pfeffermann Consulting. (2012) RoboScan'12 - Studienergebnisse der Onlinebefragung zum Markt der Robotik-Logistik - Kurzreprot. Online verfügbar unter: http://www.robotik-logistik.de/fileadmin/user_upload/services/news/2012/Kurzreport_de_web.pdf - 01.03.2014.

[8] M. Prensky, „Digital Natives, Digital Immigrants," *On the Horizon*, Vol. 9, Nr. 5, 2001.

[9] BITKOM. (2012) Der App-Boom geht weiter, Presseinformation zur App-Verbreitung. Berlin. Online verfügbar unter: http://www.bitkom.org/files/documents/BITKOM-Presseinfo_App-Verbreitung_10_10_2012.pdf - 20.01.2014.

[10] K. Khoshelham und S. O. Elberink, „Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications," *Sensors*, Vol. 12, Nr. 12, S. 1437–1454, 2012.

[11] BITKOM. (2013) Tablet-Nutzung in Unternehmen, Presseinformation. Berlin. Online verfügbar unter: http://www.bitkom.org/files/documents/BITKOM_Presseinfo_Tablet-Nutzung_in_Unternehmen_19_04_2013.pdf - 20.01.2014.

[12] U. W. Schamari, „Smartphones und Tablet-PC erobern den Maschinenbau," *MM MaschinenMarkt – Das Industriemagazin*, Vol. 2012, Nr. 47, 2012.

[13] M. Dahm, *Grundlagen der Mensch-Computer-Interaktion*, 1. Aufl., München: Pearson Studium, 2006.

[14] K.-P. Timpe, „Mensch-Maschine-Interaktion in der Fertigungstechnik," in *Handbuch der Mess- und Automatisierungstechnik in der Produktion*, Serie VDI-Buch, H.-J. Gevatter und U. Grünhaupt, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2006, S. 3–8.

[15] T. Stiedl, „Multimodale Interaktion mit Automatisierungssystemen," Dissertation, Universität Stuttgart, 2010.

[16] Norm DIN EN ISO 9241 „Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability," 1999-01.

[17] Norm DIN EN ISO 9241 „Ergonomics of human-system interaction Part 210: Human-centred design for interactive systems," 2011-08.

[18] Norm DIN EN ISO 14915 „Software ergonomics for multimedia user interfaces - Part 2: Multimedia navigation and control," 2003-11.

[19] Norm DIN EN ISO 6385 „Ergonomic principles in the design of work systems," 2004-05.

[20] Technische Richlinie VDI/VDE 3850 Blatt 1 „Gebrauchstaugliche Gestaltung von Benutzungsschnittstellen für technische Anlagen - Konzepte, Prinzipien und grundsätzliche Empfehlungen," 2012-08.

[21] Technische Richtlinie VDI/VDE 3850 Blatt 2 „Nutzergerechte Gestaltung von Bediensystemen für Maschinen - Interaktionsgeräte für Bildschirme," 2002-11.

[22] Technische Richtlinie VDI/VDE 3850 Blatt 3 „Nutzergerechte Gestaltung von Bediensystemen für Maschinen - Dialoggestaltung für Touchscreens," 2004-03.

[23] Norm DIN EN ISO 9241 „Ergonomic requirements for office work with visual display terminals (VDTs) - Part 16: Direct-manipulation dialogues," 2000-03.

[24] K. Renaud und R. Cooper, „Feedback in human-computer interaction - characteristics and recommendations," *South African Computer Journal*, Vol. 26, S. 105–114, 2000.

[25] M. Schudnagis und C. Womser-Hacker, „Feedback als Kernelement der benutzerfreundlichen Mensch-Maschine-Interaktion bei Lernsystemen," in *DeLFI 2003: Die 1. e-Learning Fachtagung Informatik*, A. Bode, J. Desel, S. Rathmeyer und M. Wessner, Hrsgg., Gesellschaft für Informatik, 2003, S. 173–182.

[26] C. Malerczyk, „Intuitive Interaktion durch videobasierte Gestenerkennung," Dissertation, Universität Rostock, 2009.

[27] S.-K. Ong und Z. B. Wang, „Augmented assembly technologies based on 3D bare-hand interaction," *CIRP Annals - Manufacturing Technology*, Vol. 60, Nr. 1, S. 1–4, 2011.

[28] J. Blake, *Natural User Interfaces in . Net*, 1. Aufl., Serie Manning Pubs Co, Shelter Island (NY): Manning Publications Company, 2012.

[29] D. Wigdor und D. Wixon, *Brave NUI world: Designing natural user interfaces for touch and gesture*, 1. Aufl., Burlington (MA): Morgan Kaufmann, 2010.

[30] R. Marin, P. Sanz, P. Nebot und R. Wirz, „A multimodal interface to control a robot arm via the web: a case study on remote programming," *IEEE Transactions on Industrial Electronics*, Vol. 52, Nr. 6, S. 1506–1520, 2005.

[31] J. N. Pires, „Robot-by-voice: experiments on commanding an industrial robot using the human voice," *Industrial Robot: An International Journal*, Vol. 32, Nr. 6, S. 505–511, 2005.

[32] I. Wechsung, J. Hurtienne und A. Naumann, „Multimodale Interaktion: Intuitiv, robust, bevorzugt und altersgerecht?" in *Konferenz Mensch & Computer 2009: Grenzenlos frei!?*, Oldenbourg Verlag, 2009, S. 213–222.

[33] P. Boudoin, C. Domingues, S. Otmane, N. Ouramdane und M. Mallem, „Towards multimodal human-robot interaction in large scale virtual environment," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, Serie HRI '08, New York, NY, USA: ACM, 2008, S. 359–366.

[34] Fraunhofer IPA. (2007) ImRoNet: Internetbasierte multimediale/multimodale Nutzerschnittstellen zur Teleoperation von Robotern. Online verfügbar unter: http://www.ipa.fraunhofer.de/ImRoNet.568.0.html - 20.01.2014.

[35] B. Akan, A. Ameri, B. Cürüklü und L. Asplund, „Intuitive industrial robot programming through incremental multimodal language and augmented reality," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, S. 3934–3939.

[36] V. I. Pavlovic, R. Sharma und T. S. Huang, „Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 19, Nr. 7, S. 677–695, 1997.

[37] D. McNeill, *Gesture and thought*, 1. Aufl., Chicago: University of Chicago Press, 2005.

[38] S. Berman und H. Stern, „Sensors for Gesture Recognition Systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Vol. 42, Nr. 3, S. 277–290, 2012.

[39] S. S. Rautaray und A. Agrawal, „Vision based hand gesture recognition for human computer interaction: a survey," *Artificial Intelligence Review*, 2012.

[40] R. Azuma, „A Survey of Augmented Reality," *Presence: Teleoperators and Virtual Environments*, Vol. 6, S. 355–385, 1997.

[41] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier und B. MacIntyre, „Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, Vol. 21, Nr. 6, S. 34–47, 2001.

[42] D. W. F. van Krevelen und R. Poelman, „A Survey of Augmented Reality Technologies, Applications and Limitations," *The International Journal of Virtual Reality*, Vol. 9, Nr. 2, S. 1–20, 2010.

[43] P. Milgram und H. Colquoun, „A Taxonomy of Real and Virtual World Display Integration," in *Proceedings of 1st International Symposium on Mixed Reality (ISMAR)*, J. Otah und T. H., Hrsgg., Berlin und Heidelberg: Springer Verlag, 1999, S. 5–30.

[44] A. Tegtmeier, „Augmented Reality als Anwendungstechnologie in der Automobilindustrie," Dissertation, Otto von Guericke Universität Magdeburg, 2007.

[45] G. Bleser, C. Wohlleber, M. Becker und D. Stricker, „Fast and Stable Tracking for AR fusing Video and Inertial Sensor Data," in *Proceedings of the 14th International Conference on in Central Europe on Computer Graphics, Visualization ans Computer Vision*, 2006, S. 109–115.

[46] G. Klein, „Visual Tracking for Augmented Reality," Dissertation, University of Oxford, 2006.

[47] J. Köhler, A. Pagani und D. Stricker, „Detection and Identification Techniques for Markers Used in Computer Vision," in *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop)*, Serie OpenAccess Series in Informatics (OASIcs), A. Middel, I. Scheler und H. Hagen, Hrsgg., Vol. 19, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, S. 36–44.

[48] D. Abawi, J. Bienwald und R. Dorner, „Accuracy in optical tracking with fiducial markers: an accuracy function for ARToolKit," in *Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2004, S. 260–261.

[49] J. Pilet, V. Lepetit und P. Fua, „Fast Non-Rigid Surface Detection, Registration and Realistic Augmentation," *International Journal of Computer Vision*, Vol. 76, Nr. 2, S. 109–122, 2008.

[50] A. Placitelli und L. Gallo, „Low-Cost Augmented Reality Systems via 3D Point Cloud Sensors," in *Seventh International Conference on Signal-Image Technology and Internet-Based Systems (SITIS)*, 2011, S. 188–192.

[51] S. Izad, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison und A. Fitzgibbon, „Kinectfusion: real-time 3D reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST)*, 2011, S. 559–568.

[52] D. Zufferey, „Device based gesture recognition," *ACM Second International Conference on Tangible and Embedded Interaction*, Vol. 2008, 2008.

[53] Norm DIN EN 60529 „Degrees of protection provided by enclosures (IP code) (IEC 60529:1989 + A1:1999)," 2000-09.

[54] E. Riedenklau, D. Petker, T. Hermann und H. Ritter, „Embodied Social Networking with Gesture-enabled Tangible Active Objects," in *Proceedings of AMiRE*, U. Rückert, J. Sitte und F. Werner, Hrsgg., 2011.

[55] J. Fischer, D. Flohr und W. Straßer, „Selective stylization for visually uniform tangible AR," in *Proceedings of the 14th Eurographics conference on Virtual Environments*, 2008, S. 1–8.

[56] D. Schmalstieg, A. Fuhrmann und G. Hesina, „Bridging multiple user interface dimensions with augmented reality," in *Proceedings of the IEEE and ACM International Symposium on Augmented Reality (ISAR)*, 2000, S. 20–29.

[57] J. W. S. Chong, S.-K. Ong und A. Y. C. Nee, „Methodologies for Immersive Robot Programming in an Augmented Reality Environment," *The International Journal of Virtual Reality (IJVR)*, Vol. 6, Nr. 1, S. 69–79, 2007.

[58] B. Schwald und B. De Laval, „An Augmented Reality System for Training and Assistance to Maintenance in the Industrial Context," in *Proceedings of the 11th International Conference*

in Central Europe on Computer Graphics, Visualization and Computer Vision, 2003, S. 425–432.

[59] R. Reif, „Entwicklung und Evaluierung eines Augmented Reality unterstützten Kommissioniersystems," Dissertation, Technische Universität München, 2009.

[60] Y. Shen, S.-K. Ong und A. Y. C. Nee, „Vision-Based Hand Interaction in Augmented Reality Environment," *International Journal of Human-Computer Interaction*, Vol. 27, Nr. 6, S. 523–544, 2011.

[61] Technische Richtlinie VDI 2860 „Montage- und Handhabungstechnik; Handhabungsfunktionen, Handhabungseinrichtungen; Begriffe, Definitionen, Symbole," 1990-05.

[62] G. Spur und E. Uhlmann, „Industrieroboter," in *Dubbel*, K.-H. Grote und J. Feldhusen, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2005, S. T106–T112.

[63] W. Weber, *Industrieroboter: Methoden der Steuerung und Regelung*, 2. Aufl., München: Fachbuchverl. Leipzig im Carl-Hanser-Verl., 2009.

[64] W. Mahnke, S.-H. Leitner und M. Damm, *OPC unified architecture*, 1. Aufl., Berlin und Heidelberg: Springer Verlag, 2009.

[65] R. Bernhardt, G. Schreck und C. Willnow, „A cost effective control system: CEROS," Proceedings of the International Symposium on Industrial Robots 27, 1996.

[66] Norm ISO 6983 „Automation systems and integration - Numerical control of machines - Program format and definitions of address words - Part 1: Data format for positioning, line motion and contouring control systems," 2009-12.

[67] M. Weck und C. Brecher, „Robotersteuerungen," in *Werkzeugmaschinen 4*, 6. Aufl., Serie VDI-Buch, Berlin und Heidelberg: Springer Verlag, 2006, S. 353–418.

[68] KUKA Roboter GmbH. (2013) Controller KR C4 Specification. Ausgsburg. Online verfügbar unter: http://www.kuka-robotics.com/res/sps/94de4d6a-e810-4505-9f90-b2d3865077b6_Spez_KR_C4_NA_de.pdf - 20.01.2014.

[69] J. Bickendorf, „Roboter-Schweißen von Stahlbauprofilen mit Losgröße 1 - „Schweißbaugruppenschnittstelle Stahlbau" und Offline-Programmiersystem ermöglichen wirtschaftliche Automatisierung," in *VDI-Berichte Nr. 2012*, VDI Verlag, 2008.

[70] E. Freund, B. Ludemann-Ravit, O. Stern und T. Koch, „Creating the architecture of a translator framework for robot programming languages," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 1, 2001, S. 187–192.

[71] P. Neto, J. N. Pires und A. P. Moreira, „High-level programming and control for industrial robotics: using a hand-held accelerometer-based input device for gesture and posture recognition," *Industrial Robot: An International Journal*, Vol. 37, Nr. 2, S. 137–147, 2010.

[72] M. Hägele, K. Nilsson und J. Pires, „Industrial Robotics," in *Springer Handbook of Robotics*, B. Siciliano und O. Khatib, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2008, S. 963–986.

[73] G. Biggs und B. MacDonald, „A Survey of Robot Programming Systems," in *Proceedings of the Australasian Conference on Robotics and Automation (CSIRO)*, 2003, S. 27–36.

[74] C. Brecher, M. Göbel, W. Herfs und G. Pohlmann, „Roboterprogrammierung durch Demonstration: Ein ganzheitliches Verfahren zur intuitiven Erzeugung und Optimierung von Roboterprogrammen," *wt Werkstattstechnik online*, Vol. 2009, Nr. 9, S. 655–660, 2009.

[75] B. D. Argall, S. Chernova, M. Veloso und B. Browning, „A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, Vol. 57, Nr. 5, S. 469–483, 2009.

[76] P. Prassler, D. Stopp, M. Hägele, I. Iossifidis, D. Lawitzky, D. Grunwald und P.-I. Dillmann, „Learning, Programming and Instructing," in *Advances in Human-Robot Interaction*, Serie Springer tracts in advanced robotics, E. Prassler, G. Lawitzky, A. Stopp, G. Grunwald, M. Hägele, R. Dillmann und I. Iossifidis, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2005, Vol. 14, S. 117–118.

[77] V. Schmirgel, U. E. Zimmermann, T. Hulin und C. Preusche, „Position Paper: Human Skills for Programming-by-Demonstration of Robots," *Beyond Movement. Skills - scientific workshop*, 2007.

[78] R. Schraft, „The need for an intuitive teaching method for small and medium enterprises," in *Proceedings of the Joint Conference on Robotics. ISR 2006, 37th International Symposium on Robotics and 4th German Conference on Robotics*, 2006.

[79] J. N. Pires, „New challenges for industrial robotic cell programming," *Industrial Robot: An International Journal*, Vol. 36, Nr. 1, 2009.

[80] C. Brecher, J. Roßmann, C. Schlette, W. Herfs, H. Ruf und M. Göbel, „Intuitive Roboterprogrammierung in der automatisierten Montage," *wt Werkstattstechnik online*, Nr. 9, S. 681–686, 2010.

[81] S. Estable, I. Ahms, H. G. Backhaus, O. El Zubi und R. Muenstermann, „Intuitive teaching and surveillance for production assistants," in *Proceedings of the 11th IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, Berlin, 2002, S. 474–481.

[82] E. Uhlmann, T. Friedrich und N. Bayat, „Development of a technology orientated programming system for industrial robots," *Production Engineering*, Vol. 2, Nr. 1, S. 103–108, 2008.

[83] O. Rogalla, M. Ehrenmann, R. Zollner, R. Becher und R. Dillmann, „Using gesture and speech control for commanding a robot assistant," in *Proceedings of the11th IEEE International Workshop on Robot and Human Interactive Communication*, Sep.2002, S. 454–459.

[84] G. Grunwald, G. Schreiber, A. Albu-Schäffer und G. Hirzinger, „Touch: The direct type of human interaction with a redundant service robot," in *Proceedings of the 10th IEEE International Workshop on Robot and Human Interactive Communication*, 2001, S. 347–352.

[85] A. Albu-Schäffer und G. Hirzinger, „Cartesian impedance control techniques for torque controlled light-weight robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 1, 2002, S. 657–663.

[86] R. Dillmann, R. Zöllner, M. Ehrenmann und O. Rogalla, „Interactive natural programming of robots: Introductory overview," in *Proceedings of the 2nd IARP/IEEE-RAS Joint Workshop on Technical Challenge for Dependable Robots in Human Environments*, 2002.

[87] R. Zollner, T. Asfour und R. Dillmann, „Programming by demonstration: dual-arm manipulation tasks for humanoid robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 1, 2004, S. 479–484.

[88] T. Asfour, P. Azad, F. Gyarfas und R. Dillmann, „Imitation Learning of Dual-Arm Manipulation Tasks in Humanoid Robots," *International Journal of Humanoid Robotics*, Vol. 5, Nr. 2, S. 183–202, 2008.

[89] R. Bischoff, A. Kazi und M. Seyfarth, „The MORPHA style guide for icon-based programming," in *Proceedings of the11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002, S. 482–487.

[90] A. Kazi, J. Bunsendal, D. Haag, R. Baum und R. Bischoff, „Next Generation Teach Pendants for Industrial Robots," in *Advances in Human-Robot Interaction*, Serie Springer Tracts in Advanced Robotics, E. Prassler, G. Lawitzky, A. Stopp, G. Grunwald, M. Hägele, R. Dillmann und I. Iossifidis, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2005, Nr. 14, S. 47–66.

[91] J. N. Pires, „Robotics for small and medium enterprises: control and programming challenges," *Industrial Robot: An International Journal*, Vol. 33, Nr. 6, 2006.

[92] C. Meyer, R. Hollmann, C. Parlitz und M. Hägele, „Programmieren durch Vormachen für Assistenzsysteme – Schweiß- und Klebebahnen intuitiv programmieren (Programming by Demonstration for Assistive Systems – Intuitive Programming of Welding and Gluing Trajectories)," *it - Information Technology*, Vol. 49, Nr. 4, S. 238–246, 2007.

[93] C. Meyer, „Aufnahme und Nachbearbeitung von Bahnen bei der Programmierung durch Vormachen von Industrierobotern," Dissertation, Universität Stuttgart, 2011.

[94] J. Pires, T. Godinho und R. Araújo, „Using digital pens to program welding tasks," *Industrial Robot: An International Journal*, Vol. 34, Nr. 6, S. 476–486, 2007.

[95] J. Roßmann, C. Schlette und H. Ruf, „Modellbasierte Roboterprogrammierung 'by Demonstration' (ProDemo)," *Augmented & Virtual Reality in der Produktentstehung, 8. Paderborner Workshop*, Nr. 252, S. 65–76, 2009.

[96] W. Vogl, „Eine interaktive räumliche Benutzerschnittstelle für die Programmierung von Industrierobotern," Dissertation, Technische Universität München, 2009.

[97] G. Reinhart, U. Munzert und W. Vogl, „A programming system for robot-based remote-laser-welding with conventional optics," *CIRP Annals - Manufacturing Technology*, Vol. 57, Nr. 1, S. 37–40, 2008.

[98] G. Arbeiter, A. Bubeck und J. Fischer, „Teleoperierte mobile Roboter warten Prozessanlagen," *atp edition*, Vol. 52, Nr. 05, S. 44–51, 2010.

[99] A. Steiger, „Interaktive Projektionsbasierte Erweiterte Realität in der Robotik." Dissertation, Karlsruhe Institute of Technology, 2011.

[100] R. Bischoff und J. Kurth, „Concepts, Tools and Devices for Facilitating Human-Robot Interaction with Industrial Robots through Augmented Reality," in *Proceedings of the Fifth IEEE and ACM International Symposium on Mixed and Ausgemented Reality (ISMAR)*, 2006.

[101] R. Bischoff, Y. Kogan und J. Kurth, „Lebenszyklusorientierte Betrachtung der Einsatzmöglichkeiten von Augmented Reality in der Industrierobotik," *atp - Automatisierungstechnische Praxis, Praxis der Mess-, Steuerungs-, Regelungs- und Informationstechnik*, Vol. 49, Nr. 7, 2007.

[102] T. Alt, W. Schreiber, W. Wohlgemuth und P. Zimmermann, „Das Verbundprojekt AVILUS," in *Virtuelle Techniken im industriellen Umfeld*, W. Schreiber und P. Zimmermann, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2011, S. 4–18.

[103] U. Zimmermann, „AR basiertes Trainingstool in der Robotik," in *Virtuelle Techniken im industriellen Umfeld*, W. Schreiber, Hrsg., Berlin and Heidelberg, 2011, S. 4–18, 694–697.

[104] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze und A. Wortmann, „A new skill based robot programming language using UML/P Statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, S. 461–466.

[105] T. Dietz, U. Schneider, M. Barho, S. Oberer-Treitz, M. Drust, R. Hollmann und M. Haegele, „Programming System for Efficient Use of Industrial Robots for Deburring in SME Environments," in *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, 2012, S. 1–6.

[106] A. Gaschler, M. Springer, M. Rickert und A. Knoll, „Intuitive Robot Tasks with Augmented Reality and Virtual Obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[107] KUKA Laboratories GmbH. TAPAS: Robotics-enabled Logistics and Assistive Services for the Transformable Factory of the Future. Online verfügbar unter: http://www.tapas-project.eu/files/TAPAS_flyer.pdf - 20.01.2014.

[108] M. Hvilshøj, S. Bøgh, S. O. Nielsen und O. Madsen, „Multiple part feeding - real-world application for mobile manipulators," *Assembly Automation*, Vol. 32, Nr. 1, S. 62–71, 2012.

[109] J. Aleotti, A. Skoglund und T. Duckett, „Position Teaching of a Robot Arm by Demonstration with a Wearable Input Device," in *Proceedings of the International Conference on Intelligent Manipulation and Grasping (IMG)*, 2004.

[110] K. Dixon, S. Iba, J. Dolan und P. Khosla, „Gesture-based Programming for Robotic Arc Welding," Technical report, Carnegie Mellon University, Institute for Complex Engineered Systems, 2002.

[111] B. Hein und H. Wörn, „Intuitive and model-based on-line programming of industrial robots: New input devices," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, S. 3064–3069.

[112] B. Hein, M. Hensel und H. Wörn, „Intuitive and model-based on-line programming of industrial robots: A modular on-line programming environment," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2008, S. 3952–3957.

[113] T. Stipancic, B. Jerbic, A. Bucebic und P. Curkovic, „Programming an Industrial Robot by Demonstration," in *Annals of DAAAM for 2012 & Proceedings of the 23rd International DAAAM Symposium*, Vol. 23, 2012, S. 15–18.

[114] A. Léon, E. F. Morales, L. Altamirano und J. R. Ruiz, „Teaching a Robot to Perform Task through Imitation and On-line Feedback," in *Progress in Pattern Recognition, Image*

Analysis, Computer Vision, and Applications, Serie Lecture Notes in Computer Science, C. S. Martin und S.-W. Kim, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2011, Nr. 7042, S. 549–556.

[115] G. Burdea, „Invited review: the synergy between virtual reality and robotics," *IEEE Transactions on Robotics and Automation*, Vol. 15, Nr. 3, S. 400–410, 1999.

[116] Z. Pan, J. Polden, N. Larkin, S. Van Duin und J. Norrish, „Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, Vol. 28, Nr. 2, S. 87–94, 2012.

[117] A. Rastogi, P. Milgram und D. Drascic, „Telerobotic control with stereoscopic augmented reality," in *SPIE Proceedings*, Vol. 2653, 1996, S. 135–146.

[118] R. Marin, P. Sanz und J. Sánchez, „A very high level interface to teleoperate a robot via Web including augmented reality," in *proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 3, 2002, S. 2725–2730.

[119] J. Bravo, C. Alberto, F. A. C. Herás, M. Fernández und F. Torres Medina, „An augmented reality interface for training robotics through the web," *Communication*, S. 189–194, 2005.

[120] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal und T. Lokstad, „Augmented reality for programming industrial robots," in *Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2003, S. 319–320.

[121] J. W. S. Chong, S.-K. Ong, A. Y. C. Nee und K. Youcef-Youmi, „Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, Vol. 25, Nr. 3, S. 689–701, 2009.

[122] J. W. S. Chong, A. Y. C. Nee, K. Youcef-Toumi und S.-K. Ong, „An Application of Augmented Reality (AR) in the Teaching of an Arc Welding Robot," *Innovation in Manufacturing Systems and Technology (IMST)*, 2005.

[123] H. C. Fang, S.-K. Ong und A. Y. C. Nee, „Interactive robot trajectory planning and simulation using Augmented Reality," *Robotics and Computer-Integrated Manufacturing*, Vol. 28, Nr. 2, S. 227–237, 2012.

[124] B. Akan, B. Cürüklü und L. Asplund, „Interacting with industrial robots through a multi-modal language and sensory systems," in *Proceedings of the 39th International Symposium on Robotics*, 2008, S. 66–69.

[125] E. A. Ameri, B. Akan und B. Cürüklü, „Augmented Reality Meets Industry: Interactive Robot Programming," in *Proceedings of SIGRAD*, Västerås, Sweden, 2010, S. 55–58.

[126] B. Akan, B. Cürüklü, G. Spampinato und L. Asplund, „Object selection using a spatial language for flexible assembly," in *Proceedings of the IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2009, S. 1–6.

[127] S. Abbas, S. Hassan und J. Yun, „Augmented Reality Based Teaching Pendant for Industrial Robot," in *Proceedings of the 12th International Conference on Control, Automation and Systems (ICCAS)*, 2012, S. 2210–2213.

[128] J. A. Neuhöfer, „System zur Simulation der Mensch-Roboter-Kooperation in virtuellen und erweiterten Umgebungen." Dissertation, RWTH Aachen, 2011.

[129] Norm DIN EN ISO 10218 „Robots and robotic devices - Safety requirements for industrial robots - Part 1: Robots," 2012-01.

[130] J.-G. Neugebauer, „Einsatz neuer Mensch-Maschine-Schnittstellen für Robotersimulation und -programmierung," Dissertation, Universität Stuttgart, 1997.

[131] Technische Richtlinie VDI/VDE 4499 Blatt1 „Digitale Fabrik - Grundlagen," 2008-02.

[132] C. Brecher und B. Schröter, „Inbetriebnahme und Programmierung portabler Robotersysteme," in *BMBF-Abschlussveranstaltung Robotik und Handhabungstechnik in der Produktion*, Stuttgart, 2005.

[133] N. Ahlbehrendt, B. Matthias, A. Neumann, O. Görnemann, S. Prinz, C. Almeida und J. Hümmler, „Sensortechnik für den ortsflexiblen Robotereinsatz," in *Tagungsband Robotik 2004: Leistungsstand, Anwendungen, Visionen, Trends*, Serie VDI-Berichte 1841, München, 2004, S. 559–566.

[134] J. Lambrecht, M. Kleinsorge und J. Krüger, „Markerless gesture-based motion control and programming of industrial robots," in *Proceedings of the IEEE 16th Conference on Emerging Technologies Factory Automation (ETFA)*, 2011, S. 1–4.

[135] J. Krüger, T. Lien und A. Verl, „Cooperation of human and machines in assembly lines," *CIRP Annals - Manufacturing Technology*, Vol. 58, Nr. 2, S. 628–646, 2009.

[136] Norm DIN EN ISO 10218 „Robots and robotic devices - Safety requirements for industrial robots - Part 2: Robot systems and integration," 2012-06.

[137] S. Kurkovsky, R. Koshy, V. Novak und P. Szul, „Current issues in handheld augmented reality," in *Proceedings of the International Conference on Communications and Information Technology (ICCIT)*, 2012, S. 68–72.

[138] D. Baricevic, C. Lee, M. Turk, T. Hollerer und D. Bowman, „A hand-held AR magic lens with user-perspective rendering," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2012, S. 197–206.

[139] Y. Tokusho und S. Feiner, „Prototyping an outdoor mobile augmented reality street view application," in *Proceedings of the 8th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009.

[140] O. Cakmakci und J. Rolland, „Head-Worn Displays: A Review," *Journal of Display Technology*, Vol. 2, Nr. 3, S. 199–216, 2006.

[141] A. State, K. Keller und H. Fuchs, „Simulation-based design and rapid prototyping of a parallax-free, orthoscopic video see-through head-mounted display," in *Proceedings of the Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2005, S. 28–31.

[142] D. Wagner, „Handheld augmented reality," Dissertation, Graz University of Technology, 2007.

[143] S. Malik, G. Roth und C. McDonald, „Robust corner tracking for real-time augmented reality," *Vision Interface*, S. 399–406, 2002.

[144] M. Tönnis, *Augmented Reality: Einblicke In Die Erweiterte Realität*, 1. Aufl., Berlin und Heidelberg: Springer Verlag, 2010.

[145] C. Owen, F. Xiao und P. Middlin, „What is the best fiducial?" in *Proceedings of the the First IEEE International Workshop on Augmented Reality Toolkit*, 2002, S. 98–105.

[146] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond und D. Schmalstieg, „Pose tracking from natural features on mobile phones," in *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, 2008, S. 125–134.

[147] J. Lambrecht und J. Krüger, „Spatial programming for industrial robots based on gestures and Augmented Reality," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, S. 466–472.

[148] A. B. Postawa, M. Kleinsorge und J. Krüger, „Automated Image Based Recognition of Manual Work Steps in the Remanufacturing of Alternators," in *Advances in Sustainable Manufacturing*, G. Seliger, M. M. K. Khraisheh und I. S. Jawahir, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2011, S. 209–214.

[149] T. Miyasato und R. Nakatsu, „Allowable delay between images and tactile information in a haptic interface," in *Proceedings of the International Conference on Virtual Systems and MultiMedia (VSMM)*, 1997, S. 84–89.

[150] B. Watson, N. Walker, P. Woytiuk und W. Ribarsky, „Maintaining usability during 3D placement despite delay," in *Proceedings of IEEE Virtual Reality*, 2003, S. 133–140.

[151] Norm DIN EN ISO 9241 „Ergonomie der Mensch-System-Interaktion - Teil 171: Leitlinien für die Zugänglichkeit von Software," 2008-10.

[152] C. D. Mutto, P. Zanuttigh und G. Cortelazzo, *Time-of-Flight Cameras and Microsoft Kinect*, New York: Springer Verlag US, 2012.

[153] J. Radmer, „Depth data based determination of gait parameters of subjects after stroke for the use in clinical gait rehabilitation," Dissertation, Technische Universität Berlin, 2011.

[154] J.-M. Schares, L. Hoegner und U. Stilla, „Geometrische Untersuchung zur Tiefengenauigkeit des Kinect-Sensorsystems," *Tagungsband 32. Wissenschaftlich-Technische Jahrestagung der DGPF*, Nr. 21, S. 372–380, 2012.

[155] M. Livingston, J. Sebastian, Z. Ai und J. Decker, „Performance measurements for the Microsoft Kinect skeleton," in *IEEE Virtual Reality Short Papers and Posters (VRW)*, 2012, S. 119–120.

[156] I. Oikonomidis, N. Kyriazis und A. Argyros, „Efficient model-based 3D tracking of hand articulations using Kinect," in *Proceedings of the 22nd British Machine Vision Conference*, British Machine Vision Association, 2011, S. 101.1–101.11.

[157] J. Lambrecht, M. Kleinsorge, M. Rosenstrauch und J. Krüger, „Spatial Programming for Industrial Robots Through Task Demonstration," *International Journal of Advanced Robotic Systems*, S. 1–10, 2013.

[158] M. Tosas, „Visual articulated hand tracking for interactive surfaces," Dissertation, University of Nottingham, 2006.

[159] J. Lambrecht, H. Walzel und J. Krüger, „Robust finger gesture recognition on handheld devices for spatial programming of industrial robots," in *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2013, S. 99–106.

[160] J. Krüger, B. Nickolay, P. Heyer und G. Seliger, „Image based 3D Surveillance for flexible Man-Robot-Cooperation," *CIRP Annals - Manufacturing Technology*, Vol. 54, Nr. 1, S. 19–22, 2005.

[161] V. Vezhnevets, V. Sazonov und A. Andreeva, „A Survey on Pixel-Based Skin Color Detection Techniques," in *Proceedings of the GraphiCon*, 2003, S. 85–92.

[162] G. Gomez, „On selecting colour components for skin detection," in *Proceedings of the 16th International Conference on Pattern Recognition*, Vol. 2, 2002, S. 961–964.

[163] J. Brand und J. Mason, „A comparative assessment of three approaches to pixel-level human skin-detection," in *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 1, 2000, S. 1056–1059.

[164] G. Gomez und E. F. Morales, „Automatic Feature Construction and a Simple Rule Induction Algorithm for Skin Detection," in *Proceedings of the ICML Workshop on Machine Learning in Computer Vision*, 2002, S. 31–38.

[165] A. Blake und M. Isard, *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion*, New York: Springer Verlag (US), 1998.

[166] M. Isard und J. MacCormick, „Hand tracking for vision-based drawing," Visual Dynamics Group, Department of Engineering Science, University of Oxford, Technical report, 2000.

[167] D. Zhou, Y. Wang und X. Chen, „Hand Contour Tracking Using Condensation and Partitioned Sampling," in *Technologies for E-Learning and Digital Entertainment*, Serie Lecture Notes in Computer Science, Z. Pan, X. Zhang, A. E. Rhalibi, W. Woo und Y. Li, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2008, Nr. 5093, S. 343–352.

[168] M. Isard und A. Blake, „CONDENSATION - conditional density propagation for visual tracking," *International Journal of Computer Vision*, Vol. 29, S. 5–28, 1998.

[169] J. MacCormick und M. Isard, „Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking," in *Computer Vision - ECCV 2000*, Serie Lecture Notes in Computer Science, D. Vernon, Hrsg., Berlin und Heidelberg: Springer Verlag, 2000, Nr. 1843, S. 3–19.

[170] A. Billard, S. Calinon, R. Dillmann und S. Schaal, „Robot Programming by Demonstration," in *Springer Handbook of Robotics*, B. Siciliano und O. Khatib, Hrsgg., Berlin und Heidelberg: Springer Verlag, 2008, S. 1371–1394.

[171] G. Schreck, R. Bernhardt und C. Willnow, *Virtual Robot Controller (VRC) interface: Leistungsstand - Anwendungen - Visionen - Trends*, Serie VDI-Berichte, Düsseldorf: VDI-Verlag, 2000, Vol. 1552.

[172] J. Lambrecht, M. Chemnitz und J. Krüger, „Control layer for multi-vendor industrial robot interaction providing integration of supervisory process control and multifunctional control units," in *Proceedings of the IEEE Conference on Technologies for Practical Robot Applications (TePRA)*, 2011, S. 115–120.

[173] E. Roos, „Anwendungsorientierte Meß- und Berechnungsverfahren zur Kalibrierung offline- programmierter Roboterapplikationen," Dissertation, Universität der Bundeswehr Hamburg, 1998.

[174] VisionMobile Ltd. (2012) Developer Economics 2012 - The Mobile App Economy: Marktstudie. Online verfügbar unter: http://www.visionmobile.com/product/developer-economics-2012/ - 20.01.2014.

[175] S. Siltanen, „Theory and applications of marker-based augmented reality," *VTT Technical Research Centre of Finland*, Nr. 3, 2012.

[176] M. M. Wloka und B. G. Anderson, „Resolving occlusion in augmented reality," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, Serie I3D '95, New York, NY: ACM, 1995, S. 5–12.

[177] G. Stark, *Robotik mit MATLAB:*, 1. Aufl., Serie Lehrbücher zur Informatik, München: Fachbuchverl. Leipzig im Carl-Hanser-Verl, 2009.

[178] P. Bonitz, *Freiformflächen in der rechnerunterstützten Karosseriekonstruktion und im Industriedesign: Grundlagen und Anwendungen*, Berlin und Heidelberg: Springer Verlag, 2009.

[179] L. Piegle und W. Tiller, *The NURBS book*, 2. Aufl., Berlin und Heidelberg: Springer Verlag, 1997.

[180] E. Gamma, R. Johnson, R. Helm und J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Amsterdam: Addison-Wesley Longman, 1995.

[181] J. MacCormick, *Stochastic Algorithms for Visual Tracking - Probabilistic Modelling and Stochastic Algorithms for Visual Localisation and Tracking*, London: Springer (UK), 2002.

[182] G. Kitagawa, „Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models," *Journal of Computational and Graphical Statistics*, Vol. 5, Nr. 1, S. 1–25, 1996.

[183] J. Aleotti, S. Caselli und M. Reggiani, „Leveraging on a virtual environment for robot programming by demonstration," *Robotics and Autonomous Systems*, Vol. 47, Nr. 2–3, S. 153–161, 2004.

[184] J. Krüger, G. Schreck und D. Surdilovic, „Dual arm robot for flexible and cooperative assembly," *CIRP Annals - Manufacturing Technology*, Vol. 60, Nr. 1, S. 5–8, 2011.

[185] H. Kagermann, W. Wahlster und J. Helbig, Hrsgg., *Umsetzungsempfehlungen für das Zukunftsprojekt Industrie 4.0*, Berlin: Forschungsunion im Stifterverband für die Deutsche Wirtschaft e.V., 2012.

[186] A. Fakhreddine und M. Mallem, „Robust camera pose estimation using 2d fiducials tracking for real-time augmented reality systems," in *Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry (VRCAI)*, 2004, S. 431–435.

[187] S. Prince, K. Xu und A. Cheok, „Augmented reality camera tracking with homographies," *IEEE Computer Graphics and Applications*, Vol. 22, Nr. 6, S. 39–45, 2002.

[188] J. J. Craig, *Introduction to robotics: Mechanics and control*, 3. Aufl., Upper Saddle (N.J.): Pearson/Prentice Hall, 2005.

[189] J. Prochazkova und D. Prochazka, „An Implementation of NURBS Curve Derivatives in Engineering Practice," in *Proceedings of the International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2007, S. 5–8.

Beitragende studentische Arbeiten

In Zusammenhang mit den Forschungs- und Entwicklungsarbeiten zur natürlich-räumlichen Industrieroboterprogrammierung entstanden am Fachgebiet Industrielle Automatisierungstechnik, Institut für Werkzeugmaschinen und Fabrikbetrieb (IWF) der Technischen Universität Berlin studentische Arbeiten, deren Ergebnisse partiell in das vorliegende Dokument eingeflossen sind. Die Arbeiten entstanden unter wissenschaftlicher, fachlicher und inhaltlicher Anleitung des Autors und werden im Folgenden aufgelistet. Der Dank des Autors gilt allen Studierenden für ihr Engagement bei der Unterstützung der vorliegenden Arbeit.

S. Albrecht, R. Chakrabarti und H. Walzel, „Steuerung eines virtuellen Industrieroboters auf Basis markerbasierter Augmented Reality“, Projektarbeit, 2011.

M. Bederke und M. Liskow, „Potentialanalyse der räumlichen Industrieroboterprogrammierung auf Basis von Gesten und Augmented Reality“, Projektarbeit, 2014.

M. Brendike, M. Meitzner und M. Sigmund, „Entwicklung einer Steuerungs- und Programmierumgebung für Industrieroboter auf Basis von Augmented Reality“, Projektarbeit, 2012.

F. Heinemann, „Entwicklung und Implementierung von 3D-Zirkular- und 3D-Spline-Interpolationen für eine Industrierobotersteuerung“, Bachelorarbeit, 2011.

S. Kind, „Visualisierung von gestenbasierten Interaktionsprozessen zur Programmierung von Industrierobotern“, Masterarbeit, 2014.

M. Lübbers, „Entwicklung einer gestenbasierten Steuerung für Industrieroboter“, Studienarbeit, 2012.

M. Ruhmann, „Entwicklung und Implementierung von Spline-Interpolationen zur Abfahrt von Freiformkurven durch einen Industrieroboter“, Projektarbeit, 2012.

B. Seeger, „Anpassung und Erweiterung von Kommunikationsschnittstellen und Sicherheitsfunktionen zur Steuerung von Industrierobotern mit einem Smartphone“, Diplomarbeit, 2011.

H. Walzel, „Entwicklung und Implementierung von robusten Methoden der Mensch-Roboter-Interaktion auf Basis von Handgesten und Augmented Reality“, Masterarbeit, 2013.

S. Weiß, „Entwicklung und Implementierung einer Schnittstelle zum Austausch von Industrieroboterprogrammen zwischen Digitaler Fabrik und einem multimodalen Programmiersystem auf mobilen Endgeräten“, Masterarbeit, 2013.

T. Westphal, „Entwicklung eines aufgabenorientierten Programmiersystems für robotergestützte Montageaufgaben auf einem mobilen Handheld-Gerät“, Bachelorarbeit, 2013.