UNIVERSITÄT DORTMUND

Bildbasierte Regelung mit Momenten von SIFT-Merkmalen

Torsten Seyffarth

Algorithm Engineering Report
TR06-2-006
Mai 2006
ISSN 1864-4503

Universität Dortmund
Fachbereich Informatik
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

Bildbasierte Regelung mit Momenten von SIFT-Merkmalen

Diplomarbeit von
Torsten Seyffarth

30. März 2006

Betreut von:

1. Gutachter:

Dr. rer. nat. Frank Hoffmann

Lehrstuhl für Regelungssystemtechnik
Fakultät für Elektrotechnik und Informationstechnik

2. Gutachter:

Prof. Dr. Günter Rudolph

Lehrstuhl für Algorithm Engineering und Computational Intelligence
Fachbereich Informatik

Inhaltsverzeichnis

1 Einführung

Das Thema dieser Diplomarbeit stammt aus dem Bereich der Servicerobotik. Ziel ist es, einen Roboterarm mit Hilfe einer am Arm befestigten Kamera in eine Zielposition relativ zu einem Objekt zu regeln, unter Berücksichtigung der aus dem gegebenen Szenario resultierenden Einschränkungen. Die Aufgabenstellung wird in Abschnitt 1.2 detailliert erläutert, nachdem zunächst eine kurze Einführung in den Bereich der Robotik gegeben wird.

1.1 Historie

Einen guten Einstieg in den Bereich der Robotik bietet die Frage: „Was ist eigentlich ein Roboter?“. In der heutigen Zeit sind Roboter für jedermann ein Begriff und jeder verbindet eine gewisse konkrete Vorstellung damit. Diese ist in starkem Maße von Science-Fiktion-Werken in Literatur, Film und Fernsehen geprägt. Dabei reicht die Spanne der konstruierten Szenarien von einem Utopia, in dem kein Mensch mehr lästige oder gefährliche Arbeiten erledigen muss, bis hin zu Robotern, die ihren Schöpfer, den Menschen, unterjochen.

Tatsächlich stammt der Begriff „Roboter“ aus der Literatur. Der Autor Karel Capek gebrauchte ihn erstmals 1921 in seinem Theaterstück „Rossum´s Universalroboter“ und bezeichnete damit künstlich erschaffene, nicht organische Wesen, die dem Menschen dienen und alle schweren Arbeiten verrichten sollen. Das Wort „Roboter“ ergab sich aus der Kombination der slawischen Wörter „rabota“ für „verpflichende Arbeit“ und „robotnik“, was im deutschen „Sklave“ bedeutet.

In den fünfziger Jahren war Isaac Asimov der Vordenker im Bereich der Roboter. Er stellte sie erstmals als vom Menschen programmierte Maschinen dar, die ausschließlich nach den ihnen vorgegebenen Gesetzen handeln. Neben der Schaffung einer Vielzahl bedeutender Science-Fiktion-Werke prägte er den Begriff der „Robotik“. Heute wird damit die Wissenschaft und Technologie von Robotern, ihrem Design, ihrer Fertigung und ihrer Anwendung bezeichnet.

Außerhalb der Literatur ist es schwer, einen genauen Zeitpunkt für die Erfindung des Roboters festzulegen. Die grundsätzliche Idee einer Maschine, die dem Menschen in irgendeiner Form helfen kann, ist sicherlich sehr viel älter als das Stück von Karel Capek. Denn es ist anzunehmen, dass jeder gute Ingenieur der Vergangenheit zumindest einmal darüber nachgedacht hat. Bereits vor Christi Geburt soll es in Ägypten mit Dampfkraft bewegte Statuen gegeben haben. Im 18. Jahrhundert wurden mittels Feinmechanik aus dem Uhrenbau in Mitteleuropa erste „realistische“ Kreaturen gebaut, die malen, musizieren und sogar „atmen“ konnten.

Nach dem heutigen Verständnis erfüllen diese Automaten aber nicht die Anforderungen für einen Roboter. Die mit Abstand meisten Roboter heutzutage sind Industrieroboter.

Diese sind im ISO Standard 8373 [Int94] als automatisch geregelte, reprogrammierbare, Mehrzweck-Manipulatoren definiert, die in drei oder mehr Achsen programmierbar sind. Erfunden wurde der Industrieroboter 1954 von Georg Devol. Zusammen mit Joseph Engelberger gründete er die erste Robotikfirma, die 1961 den ersten Industrieroboter bei General Motors in der Produktionslinie aufstellte. 2004 gab es alleine in Japan 356.000 Industrieroboter. Deutschland folgte auf Platz zwei mit 121.000 und als Drittes die USA mit 115.000.

Allerdings haben die heutigen Industrieroboter wenig mit den flexiblen Alleskönnern aus den Visionen der Schriftsteller gemein, die sogar dem Menschen im Alltag Konkurrenz machen können. Sie sind vielmehr für eine schnelle und exakte Wiederholung von immer der gleichen Aufgabe in einer speziellen Umgebung entwickelt, wie zum Beispiel das Setzen von Schweißpunkten. Mit der Aufgabe, die Roboter in den Alltag der Menschen zu bringen, um dort Dienstleistungen zu erfüllen, befasst sich die Servicerobotik. Das Einsatzgebiet der Serviceroboter reicht dabei von monotonen Aufgaben, wie das Verteilen der Post in einem Bürogebäude, bis hin zu gefährlichen Aufgaben, wie die Suche nach Verletzten in einem einsturzgefährdeten Haus.

Dadurch ergeben sich je nach konkretem Einsatzgebiet besondere Bedürfnisse an den Roboter. Er muss in einer Umwelt agieren, die zeitlich veränderlich, unstrukturiert, also nicht speziell an den Roboter angepasst, und gegebenenfalls vor der Ausführung der Aufgabe unbekannt ist. Änderungen in der Umgebung können zum Beispiel eine Tür sein, die geschlossen wird, oder ein Mensch, der sich in dem gleichen Raum bewegt wie der Roboter. Letzteres stellt eine große Herausforderung für den Serviceroboter da, weil in keinem Fall ein Mensch zu Schaden kommen darf. Dabei sollen die Menschen in der Umgebung keine besondere Rücksicht auf den Roboter nehmen müssen. Außerdem müssen neue Wege in der Interaktion mit dem Roboter und des Roboters mit seiner Umwelt begangen werden. Zum einen muss im Allgemeinen davon ausgegangen werden, dass der Bediener keine besondere Schulung aufweist, zum anderen muss der Roboter Objekte verschiedenster Form, Größe, Farbe und Material ohne Änderungen in der Hard- und Software handhaben können.

Dies erfordert eine neue Definition des Roboter-Begriffs. Matarić [Matck] definiert den Roboter als autonomes System, dass in der physikalischen Welt existiert, seine Umgebung wahrnehmen und in ihr agieren kann, um bestimmte Ziele zu erreichen. Die Serviceroboter, die alle oben beschriebenen Schwierigkeiten meistern können, befinden sich zur Zeit noch im Entwicklungsstadium. Aber es gibt erste einfache Varianten wie ein Staubsaugerroboter oder ein Rasenmäherroboter, die bereits im Handel erhältlich sind.

1.2 Problemstellung

Ein typisches Szenario im Bereich der Servicerobotik sind Hol- und Bringdienste in unstrukturierten Umgebungen. Abbildung 1.1 zeigt einen experimentellen Serviceroboter, wie er für wissenschaftliche Zwecke eingesetzt wird. Er besteht aus einer mobilen Plattform, um seine Position zu verändern, einer Reihe von Sensoren, mit denen er seine Umgebung wahrnehmen kann, und einem Roboterarm, um Objekte in seiner Reichweite zu manipulieren.

Ein alltägliches Beispiel aus dem Bereich der Hol- und Bringdienste würde wie folgt aussehen, wenn es einen Roboter mit den entsprechenden Fähigkeiten gäbe. Dem achzigjährigen

Abb. 1.1: Experimenteller Serviceroboter

Anwender fällt auf der Wohnzimmercouch auf, dass er die Kaffeetasse auf dem Küchentisch vergessen hat. Da er sein Alter bereit in den Beinen spürt, ist dies ein weiter Weg für ihn. Deshalb teilt er seinen Wunsch nach der Kaffeetasse mit der Ortsangabe seinem Serviceroboter mit.

Auf Grund der Spracherkennung kann der Roboter diesen Befehl verstehen. Mit Hilfe seiner Sensoren und der mobilen Plattform ist er daraufhin in der Lage, selbstständig in die Küche zum Tisch zu navigieren, und den Roboterarm in die Nähe der Kaffeetasse zu fahren. Dieser positioniert nun unter Verwendung der am Arm angebrachten Kamera den Greifer in eine bestimmte Lage relativ zur Kaffeetasse, von der aus mit Hilfe einer Methodik zum Greifen von Objekten die Tasse erfasst wird. Nun bewegt sich der Roboterarm in eine sichere Stellung zurück, und der Roboter fährt wieder ins Wohnzimmer, um dort die Tasse abzugeben.

Diese Diplomarbeit befasst sich innerhalb dieses Szenarios mit der Teilaufgabe der Positionierung des Greifers relativ zu einem Objekt durch eine bildbasierte Regelung des Roboterarms. Diese Aufgabe wird dadurch erschwert, dass die Lage des Objekts relativ zur Roboterbasis nicht bekannt und veränderlich ist, da die Sensorik der mobilen Plattform für die großräumige Navigation ausgelegt und nicht für die Erkennung von kleinen Objekten und die exakte Positionierung bestimmt ist. Außerdem können bauliche Hindernisse die Bewegungsmöglichkeiten des Roboters einschränken.

Für die Alltagstauglichkeit eines solchen Roboters ist es wichtig, dass die Objekte, mit denen der Roboter hantieren muss, nicht besonders gekennzeichnet werden müssen. Der Anwender soll nicht dazu gezwungen sein, spezielle Tassen verwenden oder die eingekaufte Wasserflasche mit Markierungen versehen zu müssen. Alle für die Regelung benötigten Informationen müssen sich aus der natürlichen Textur der Objekte extrahieren lassen.

Darüber hinaus soll der Ansatz modellfrei sein. Es soll also nicht nötig sein, die dreidimensionale Form des Objektes und die Position von Merkmalen auf dessen Oberfläche anzugeben. Statt dessen muss der Roboter alle für die Regelung benötigten Informationen in einer für jeden Objekttyp einmaligen Lernphase ermitteln. Die Lernphase startet in der Ziellage relativ zum Objekt, die durch den Greifalgorithmus vorgegeben wird. In der Lernphase kann der Roboterarm sich beliebig um das Objekt bewegen, Bilder aufnehmen und so die benötigten Informationen extrahieren und sammeln.

Die Erkenntnisse dieser Arbeit wurden zunächst anhand eines in Matlab implementierten Simulationsmodells eines Roboterarms vom Typ „Katana HD5M“ der Firma „Neuronics“ entwickelt. Anschließend wird in Kapitel 9 die Übertragbarkeit des Hauptergebnisses auf Realbedingungen demonstriert. Dazu steht der oben genannte Roboterarm in der Realität zur Verfügung.

Diese Arbeit befasst sich nach einer kurzen Einführung in die Thematik der bildbasierten Regelung zunächst mit dem Problem der Extraktion von Merkmalen aus natürlichen Texturen. Dazu wird in Kapitel 3 der SIFT-Algorithmus von Lowe [Low04] vorgestellt, der in der Textur handelsüblicher Objekte stabile Merkmalspunkte detektiert. In der Regel findet dieser Algorithmus allerdings sehr viel mehr Merkmale, als für die bildbasierten Reglung benötigt werden. In Kapitel 4 wird deshalb ein Verfahren zur automatischen Reduzierung der Anzahl durch Auswahl einer „guten“ Teilmenge der Merkmale in der Lernphase erläutert. „Gut“ ergibt sich dabei aus den Bedürfnissen der bildbasierten Regelung.

Der Hauptteil dieser Arbeit befasst sich mit der Entwicklung eines bildbasierten Reglers, der trotz der Nachteile, die die Merkmale aus einer natürlichen Textur mit sich bringen, und der Bedingung der Modellfreiheit den Roboterarm zuverlässig in die Ziellage verfährt. Dazu wird zunächst in Kapitel 5 ein Lösungsversuch mit Hilfe der adaptiven Jacobi-Matrix von Jägersand [Jäg96] dargestellt und der Grund erläutert, warum dieser Ansatz nicht zum Ziel führt. Daraus ergibt sich die alternative Lösung mit momentenbasierten Merkmalsparametern, die in Kapitel 6 und 7 erläutert wird.

In Kapitel 8 wird darauf aufbauend eine Methodik vorgestellt, wie durch Aufnahme zusätzlicher Zielbilder in der Lernphase, so genannter Zwischenbilder, der Regler in der Lage ist, auch aus Positionen das Ziel zu erreichen, in denen kein Merkmal des eigentlichen Zielbildes zu erkennen ist.

Nach der Demonstration des Hauptergebnisses auf dem realen Roboterarm, werden abschließend in Kapitel 10 die zentralen Ergebnisse dieser Arbeit nochmal kurz zusammengefasst und einige Möglichkeiten für darauf aufbauende wissenschaftliche Tätigkeiten dargelegt.

2 Grundlagen der Bildbasierten Regelung

Wie in Abschnitt 1.1 erläutert, stellt das Themenfeld der Servicerobotik neue Herausforderungen an die Regelung von Roboter. Da die Umwelt nicht an die Bedürfnisse des Roboter angepasst werden kann, muss der Roboter seine Umgebung mit Sensoren wahrnehmen und mit Hilfe dieser Informationen seine Handlungen an diese adaptieren.

Ein besonders geeigneter Sensor ist dabei die Kamera, die eines der wichtigsten menschlichen Sinnesorgane, das Auge, nachbildet. Sie ist ein passiver Sensor, der im Vergleich zu ähnlich leistungsfähigen Sensoren, wie zum Beispiel Laserscanner, deutlich günstiger ist, eine hohe Bandbreite aufweist und die Umwelt berührungsfrei erfassen kann. Auf Grund der Fortschritte in der Computertechik kann die rechenintensive Bildverarbeitung heutzutage auf preisgünstiger PC-Hardware ausgeführt werden. Trotzdem bildet die Bildverarbeitung immer noch den Flaschenhals der Ausführung eines Regelschrittes.

„Visual Servo“ Systeme, wie die Regelungssysteme mit einer Kamera im Regelkreis genannt werden, werden bereits in einer Vielzahl von wissenschaftlichen Arbeiten erörtert und nutzen die Erkenntnisse aus verschiedenen Forschungsgebieten wie Bildverabeitung, Kinematik, Dynamik, Regelungstechnik und Informatik. Ein ausführliches Tutorial der grundlegenden Begrifflichkeiten und Methodiken bietet die Arbeit von Corke, Hutchinson und Hager [CHH96]. In den nachfolgenden Abschnitten wird eine kurze Zusammenfassung bis hin zum Beispiel eines bildbasierten Reglers gegeben.

2.1 Homogene Koordinaten-Transformation

Für die Regelung eines Roboterarms muss man die Lage von Objekten und Punkten im Raum beschreiben können. Die Lage setzt sich aus der Position und der Orientierung des Objektes zusammen. Die Menge aller Positionen und Orientierungen, die ein Werkzeug bzw. der Tool Center Point (TCP) einnehmen kann, bildet den Arbeitsraum A des Roboterarms. Im Allgemeinen hat dieser in einer dreidimensionalen Arbeitsumgebung sechs Freiheitsgrade. In manchen Anwendungen wird dieser Raum allerdings auf einen Unterraum beschränkt. So werden in der Objektverfolgung, bei der ein Ziel im Blickfeld der Kamera gehalten werden soll, oft nur die Rotationen berücksichtigt. Ein anderes Beispiel ist der hier verwendete Roboterarm „Katana HD5M“, der auf fünf Freiheitsgrade beschränkt ist (siehe Abschnitt 5.3.1), da er nur fünf Achsen besitzt.

Typischerweise werden in der Robotik Positionen und Lagen relativ zu verschiedenen Objekten benötigt. Zum Beispiel berechnet der Regler eine Bewegung im Koordinatensystem der Kamera, für die Rückwärtstransformation wird aber die neue Lage im Koordinatensystem der Roboterbasis benötigt. Wird die Position eines Punktes $\mathbf{P}$ in einem Koordinatensystem x angegeben, so wird dies durch die Notation ${}^{x}\mathbf{P}$ verdeutlicht. Die Position eines Koordinatensystems y im Koordinatensystem x wird durch die Position des Ursprungs ${}^{x}\mathbf{t}_{y}$ von y in x und die Orientierung wird durch die Rotationsmatrix ${}^{x}\mathbf{R}_{y}$ angegeben. Somit

ergibt sich also die Lage von y als Tupel ${}^{x}\mathbf{x}_{y} = ({}^{x}\mathbf{R}_{y}, {}^{x}\,\mathbf{t}_{y})$. Wenn x nicht angegeben ist, dann ist das Weltkoordinatensystem das Bezugssystem.

Es sei ein Punkt ${}^{y}\mathbf{P}$ und die Lage von y im Koordinatensystem x durch ${}^{x}\mathbf{x}_{y} = ({}^{x}\mathbf{R}_{y}, {}^{x}\,\mathbf{t}_{y})$ gegeben. Die Koordinaten von $\mathbf{P}$ im Koordinatensystem x ergeben sich aus folgender affinen Abbildung:

$$ {}^{x}\mathbf{P} = {}^{x}\mathbf{R}_{y} \cdot {}^{y}\mathbf{P} + {}^{x}\mathbf{t}_{y} $$

In der Robotik kommt es allerdings oft vor, dass zwischen dem Ausgangs- und Zielkoordinatensystem noch mehrere weitere Koordinatensysteme liegen. Sind zum Beispiel die Lagen ${}^{x}\mathbf{x}_{y}$ und ${}^{y}\mathbf{x}_{z}$ gegeben, dann kann ${}^{x}\mathbf{P}$ aus ${}^{z}\mathbf{P}$ wie folgt berechnet werden:

$$ \begin{aligned} {}^{x}\mathbf{P} &= {}^{x}\mathbf{R}_{y} \cdot {}^{y}\mathbf{P} + {}^{x}\mathbf{t}_{y} \\ &= {}^{x}\mathbf{R}_{y} \cdot ({}^{y}\mathbf{R}_{z} \cdot {}^{z}\mathbf{P} + {}^{y}\mathbf{t}_{z}) + {}^{x}\mathbf{t}_{y} \\ &= {}^{x}\mathbf{R}_{y} \cdot {}^{y}\mathbf{R}_{z} \cdot {}^{z}\mathbf{P} + {}^{x}\mathbf{R}_{y} \cdot {}^{y}\mathbf{t}_{z} + {}^{x}\mathbf{t}_{y} \end{aligned} $$

Je mehr Lagen berücksichtigt werden müssen, um so unübersichtlicher wird diese Art der Berechnung. Deshalb wird in der Robotik die Tatsache verwendet, dass eine affine Abbildung in einem m-dimensionalen Vektorraum durch eine homogene Transformation im $(m+1)$-dimensionalen Vektorraum ausgedrückt werden kann. Die Lage ${}^{x}\mathbf{x}_{y} = ({}^{x}\mathbf{R}_{y}, {}^{x}\,\mathbf{t}_{y})$ wird somit durch folgende homogene Transformation beschrieben:

$$ {}^{x}\mathbf{T}_{y} = \begin{pmatrix} & {}^{x}\mathbf{R}_{y} & & {}^{x}\mathbf{t}_{y} \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

Die Koordinaten eines Punktes werden entsprechend durch das Hinzufügen einer 1 um eine Dimension erweitert. Das obige Beispiel vereinfacht sich dann wie folgt:

$$ \begin{aligned} {}^{x}\mathbf{P} &= {}^{x}\mathbf{T}_{y} \cdot {}^{y}\mathbf{P} \\ &= {}^{x}\mathbf{T}_{y} \cdot {}^{y}\mathbf{T}_{z} \cdot {}^{z}\mathbf{P} \end{aligned} $$

Die Lage eines Objektes wird also entweder durch die homogene Koordinatentransformation angegeben, oder durch einen Vektor, der die Position des Ursprungs und die drei Rotationswinkel angibt, die das Referenzkoordinatensystem in das Objektkoordinatensystem überführen. Beide Darstellungen können ineinander umgerechnet werden.

2.2 Kameramodelle

Zum Verständnis von Visual Servo Systemen muss man die geometrischen Aspekte begreifen, wie aus einer Szene ein Bild entsteht, was als Projektion bezeichnet wird. Jede Kamera enthält eine Linse, die eine zweidimensionale Abbildung der Szene auf die Bildebene erzeugt, wo sich der Sensor befindet. Dadurch geht zwangsläufig die Tiefeninformation eines Objektpunktes im Raum verloren. Somit entspricht jeder Bildpunkt einem Strahl im dreidimensionalen Raum, auf dem der zugehörige Objektpunkt liegt. Durch zusätzliche Daten lässt sich diese Information allerdings zurückgewinnen, zum Beispiel durch mehr

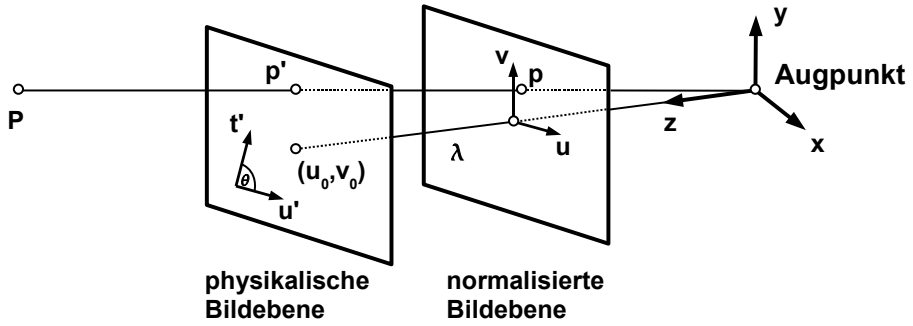


Abb. 2.1: Normalisierte und physikalische Bildebenen im Kamerakoordinatensystem

als eine Kamera, mehrere Blickwinkel der gleichen Kamera oder Informationen über die geometrische Beziehung der Merkmalspunkte.

Für diese Arbeit ist ausschließlich die perspektivische Projektion von Bedeutung, die im Folgenden erläutert wird. Zum Vergleich werden aber auch die Parallelprojektion und die affine Projektion vorgestellt. Für alle drei Projektionsmodelle gilt zunächst die Anordnung der normalisierten Bildebene aus Abbildung 2.1. Das heißt, dass die Bildebene parallel zur X-Y-Ebene des Kamerakoordinatensystems mit dem Abstand 1 liegt. Die Brennweite λ ist also 1. Die Z-Achse befindet sich senkrecht zur Bildebene und durchstößt diese im Mittelpunkt. Der Augpunkt bildet den Ursprung des Koordinatensystems der Kamera.

Perspektivische Projektion: Es sei $\mathbf{p} = [u, v]^T$ die Projektion des Punktes ${}^c\mathbf{P} = [x, y, z]^T$ auf die Bildebene, dessen Position im Koordinatensystem der Kamera angegeben ist. Wird die projektive Geometrie einer Kamera durch perspektivische Projektion modelliert (siehe zum Beispiel [Hor86]), dann ist $\mathbf{p}$ gegeben durch

$$\pi(x, y, z) = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \end{bmatrix}$$

Wird die Position von $\mathbf{P}$ in einem Koordinatensystem x angegeben, dann muss zunächst die Koordinatentransformation ${}^c\mathbf{P} = {}^c\mathbf{T}_x \cdot {}^x\mathbf{P}$ durchgeführt werden.

Parallelprojektion: Die perspektivische Projektion ist eine nichtlineare Abbildung von kartesischen Koordinaten auf Bildkoordinaten. In vielen Fällen kann diese durch eine lineare Parallelprojektion approximiert werden. In diesem Modell ergeben sich die Bildkoordinaten zu einem Punkt ${}^c\mathbf{P}$ durch

$$\begin{bmatrix} u \\ v \end{bmatrix} = s \begin{bmatrix} x \\ y \end{bmatrix}$$

Dabei ist s ein fester Skalierungsfaktor. Das Modell der Parallelprojektion ist gültig für Szenen, in denen die relative Tiefe der Punkte in der Szene klein ist im Vergleich zum Abstand der Kamera zu der Szene. Dies ist zum Beispiel bei einem Flugzeug hoch über der Erde oder einer Kamera mit großer Brennweite, die einige Meter vom Arbeitsraum entfernt ist, der Fall.

Affine Projektion: Eine andere lineare Approximation der perspektivischen Projektion ist die so genannte affine Projektion. In diesem Fall sind die Bildkoordinaten für die Projektion eines Punktes ${}^c\mathbf{P}$ gegeben durch

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{A}\,{}^c\mathbf{P} + \mathbf{c}$$

Dabei ist $\mathbf{A}$ eine beliebige 2×3 Matrix und $\mathbf{c}$ ein beliebiger zweidimensionaler Vektor. Man beachte, dass die Parallelprojektion ein Spezialfall der affinen Projektion ist. Allgemein entspricht die affine Projektion keiner speziellen bildgebenden Situation. Der primäre Vorteil liegt darin, dass es sich um ein lineares Kameramodell ohne Nebenbedingungen handelt. Somit können $\mathbf{A}$ und $\mathbf{c}$ relativ einfach aus einer Menge von korrespondierenden Paaren $\left\{({}^c\mathbf{P}_i, [u_i, v_i]^T)\right\}$ durch lineare Regressionstechniken berechnet werden. Damit ist das Kalibrierungsproblem in diesem Modell deutlich vereinfacht.

Im Allgemeinen gibt es zwei Arten von Kalibrierungsfaktoren, die in [FP02] ausführlich erklärt werden. Die extrinsischen Kameraparameter geben die Lage der Kamera in einem Referenzkoordinatensystem x an. Aus diesem Grund werden diese meistens durch eine homogene Koordinatentransformation ${}^cT_x$ angegeben.

Wie in Abbildung 2.1 zu sehen ist, unterscheidet sich im Allgemeinen die physikalische Bildebene von der Normalisierten. Diese Abweichung wird durch die intrinsischen Kameraparameter beschrieben. Dies beinhaltet folgende Faktoren:

λ: Abstand der Bildebene vom Ursprung, die so genannte Brennweite der Linse

k,l: Skalierungsfaktoren zum Umrechnen von Metern in Pixel; da die Pixel meistens nicht quadratisch sind, jeweils einen für u- und v- Richtung

u_0,v_0: Abweichung des Ursprungs der Bildebene in u- und v-Richtung (Principal Point)

θ: Winkel zwischen den physikalischen Bildachsen

Meist werden λ, k und l zu $\alpha = k \cdot \lambda$ und $\beta = l \cdot \lambda$ zusammengefasst. Dann ergibt sich folgende Matrix mit den intrinsischen Kameraparametern

$$\mathbf{K} = \begin{pmatrix} \alpha & -\alpha \cdot cot\theta & u_0 \\ 0 & \frac{\beta}{sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Wenn nicht anders angegeben, so wird im folgenden immer von einer normalisierten Bildebene ausgegangen. $\mathbf{K}$ ist dann die Einheitsmatrix.

2.3 Epipolargeometrie

Wie im vorherigen Abschnitt beschrieben, kann der zu einem Bildpunkt gehörige Objektpunkt auf einem Strahl im dreidimensionalen Raum liegen, der im Augpunkt beginnt und

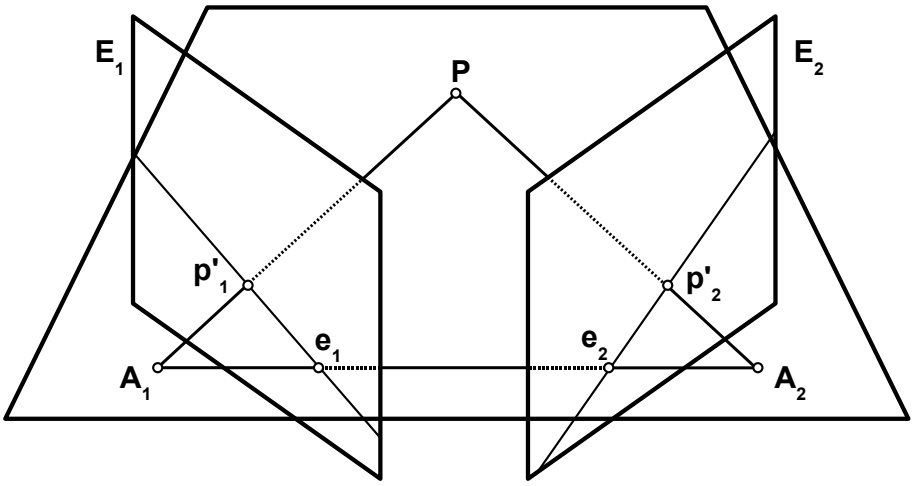


Abb. 2.2: Darstellung der Größen der Epipolargeometrie

durch den Bildpunkt geht. Wird nun der Objektpunkt durch eine zweite Kamera (oder von der selben Kamera) aus einer anderen Lage betrachtet, dann muss der zugehörige Bildpunkt in der zweiten Bildebene auf der Projektion dieses Strahles liegen. Sind die intrinsischen Kameraparameter der beiden Kameras und die relative Lage zueinander bekannt, dann kann mit Hilfe der Epipolargeometrie diese sogenannte Epipolarlinie berechnet werden.

Abbildung 2.2 illustriert die folgenden Größen. Sei $P = [x, y, z]$ der betrachtete Objektpunkt, $p'_1 = [u_1, v_1]^T$ und $p'_2 = [u_2, v_2]^T$ die Projektionen von P auf die jeweiligen Bildebenen E_1 und E_2, zu denen die beiden Augpunkte A_1 und A_2 gehören. P, A_1 und A_2 definieren die so genannte Epipolarebene, deren Schnittgeraden mit den Bildebenen die Epipolarlinien sind. Die Projektionen der Augpunkte auf die jeweils andere Bildebene werden als Epipole e_1 und e_2 bezeichnet.

Zur Berechnung der Epipolarlinie werden die homogenen Koordinaten $p_1 = [u_1, v_1, 1]^T$ und $p_2 = [u_2, v_2, 1]^T$ der Bildpunkte verwendet. Auf Grund der epipolaren Zwangsbedingung gilt folgende lineare Beziehung:

$$p_1^T \cdot F \cdot p_2 = 0$$

Dabei ist die sogenannte Fundamentalmatrix F eine 3×3 Matrix, die von der relativen Lage der Augpunkte zueinander und den intrinsischen Kameraparametern der beiden Kameras abhängt. Da sie gewissen Beschränkungen unterliegt, hat sie nur sieben freie Parameter. [FLP01] beschreibt genau, wie die Fundamentalmatrix berechnet wird.

Es seien nun p_1 und F bekannt und es wird daraus die Epipolarlinie in der zweiten Bildebene berechnet. Wenn $s = [s_1, s_2, s_3] = p_1^T \cdot F$ ist, dann muss $s \cdot p_2 = 0$ gelten. Beachtet man noch, dass die dritte Komponente von p_2 gleich 1 ist, dann bleibt folgende Gleichung übrig:

$$s_1 \cdot u_2 + s_2 \cdot v_2 + s_3 = 0$$

Die möglichen Lösungen für u_2 und v_2 bilden im Allgemeinen einen eindimensionalen Untervektorraum, der die Epipolarlinie in der zweiten Bildebene bildet.

2.4 Bildmerkmale und Bildmerkmalsraum

Um die große Informationsmenge eines Bildes für eine Regelung nutzen zu können, muss das Bild auf einige Bildmerkmale reduziert werden. In der entsprechenden Literatur ist ein Bildmerkmal als ein Strukturmerkmal definiert, das aus einem Bild extrahiert werden kann (zum Beispiel eine Ecke oder eine Kante). Oft korrespondiert ein solches Bildmerkmal mit der Projektion eines physikalischen Merkmals des betrachteten Objektes (zum Beispiel mit der Ecke einer Box).

Ein Bildmerkmalsparameter sei nun als eine reellwertige Größe definiert, die sich aus einem oder mehreren Bildmerkmalen berechnen lässt. Beispiele dafür sind Momente, Beziehungen zwischen Regionen oder Knoten und Polygonflächen. [JKCB91] gibt eine formale Definition von Bildfunktionalen an, die im folgenden als Bildmerkmalsparameter bezeichnet werden.

Am häufigsten werden die Koordinaten von einem Merkmalspunkt oder vom Schwerpunkt einer Region genutzt. Ein guter Merkmalspunkt kann dabei aus verschiedenen Ansichten stabil und eindeutig lokalisiert werden, wie zum Beispiel das Loch in einem Dichtungsring [FDD94] oder ein künstliches Muster [ECR92, CSS91].

Es wurden bereits eine Vielzahl von verschiedenen Bildmerkmalsparametern für die bildbasierte Regelung verwendet:

- Koordinaten eines Bildpunktes in der Bildebene [CH94, ECR92, KH94, PK93, PKK93, SBJ90]
- Distanz zwischen zwei Bildpunkten und die Orientierung der Verbindungslinie [FDD94, HN94]
- wahrgenommene Kantenlänge [SWN87]
- Größe einer projizierten Oberfläche und die relative Größe zweier projizierten Oberflächen [And88, LG92, SWN87, YA94]
- Schwerpunkt und Momente höherer Ordnung einer projizierten Oberfläche [SWN87, TC05]
- Parameter einer Linie in der Bildebene [ECR92]
- Parameter einer Ellipse in der Bildebene [ECR92]

Für die Regelung muss also eine Menge von k Bildmerkmalsparametern ausgewählt werden, die einen Merkmalsvektor $\mathbf{f} = [f_1 \cdots f_k]^T$ bilden. Da jedes f_i ein (möglicherweise begrenzter) reellwertiger Parameter ist, gilt $\mathbf{f} = [f_1 \cdots f_k]^T \in M \subseteq \Re^k$, wobei M den Bildmerkmalsraum repräsentiert.

Die Abbildung von der Position und Orientierung des End-Effektors auf die korrespondierenden Bildmerkmalsparameter kann durch die projektive Geometrie der Kamera berechnet werden, wenn die räumliche Lage der zu den verwendeten Bildmerkmalen gehörenden physikalischen Merkmale bekannt ist. Die Abbildung sei als $\mathbf{B} : T \rightarrow M$ bezeichnet.

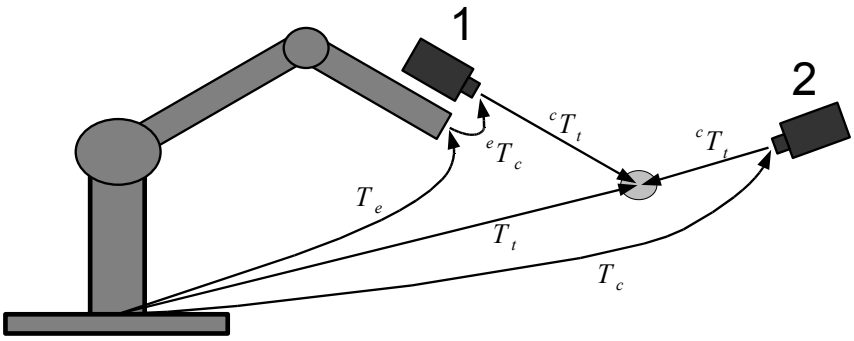


Abb. 2.3: Kamerakonfigurationen mit relevanten Koordinatentransformationen (Bild aus [CHH96] mit kleinen Modifikationen)

Wenn zum Beispiel $M \subseteq \Re^2$ der Raum der Bildkoordinaten u, v der Projektion eines Punktes $\mathbf{P}$ auf die Bildebene ist, dann gilt unter Annahme der perspektivischen Projektion

$$\mathbf{B} = \begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \end{bmatrix}$$

2.5 Kamerakonfigurationen

Bei Visual Servo Systemen wird typischer Weise eine von zwei möglichen Konfigurationen verwendet:

1. Die Kamera ist am End-Effektor befestigt.
2. Die Kamera befindet sich an einem festen Ort im Raum.

Die erste Konfiguration wird oft als „Auge-In-Hand“ Konfiguration bezeichnet. Kamera 1 in Abbildung 2.3 entspricht dieser Konfiguration. Hierbei gibt es eine bekannte, oft konstante, Transformation $^{e}\mathbf{T}_{c}$ zwischen der Lage der Kamera und des End-Effektors. Die Lage zwischen dem relevanten Objekt („Target“) und der Kamera $^{c}\mathbf{T}_{t}$ ändert sich mit der Bewegung des Roboters und gegebenenfalls der Bewegung des Objekts.

Kamera 2 in Abbildung 2.3 ist an einem festen Ort im Raum montiert und beobachtet sowohl den Roboter als auch das relevante Objekt. Hier sind die Transformationen $^{0}\mathbf{T}_{c}$ von der Roboterbasis und $^{c}\mathbf{T}_{t}$ von der Kamera zum Objekt meistens bekannt und konstant. Als eine Variante dieser Konfiguration ist die Kamera auf einem zweiten Roboter angebracht und somit beweglich, um den zu regelnden Roboter immer aus der besten Perspektive betrachten zu können [NK94].

In beiden Fällen müssen je nach gewählter Architektur für eine erfolgreiche Regelung die Kameraparameter bestimmt werden. Das sind, neben den intrisischen Parametern für die verwendete Kamera und Linse, im ersten Fall die Transformation $^{e}\mathbf{T}_{c}$ und im zweiten $^{0}\mathbf{T}_{c}$. Einige Beispiele zur Lösung des Kalibrierungsproblems finden sich in [Sut74, TL89, Tsa87].

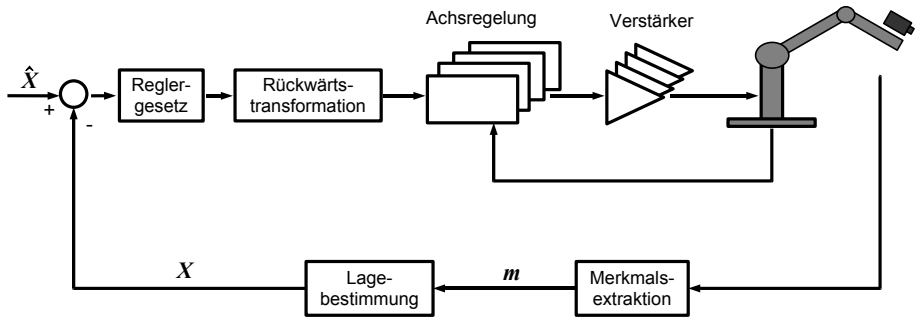


Abb. 2.4: Positionsbasierte Look-And-Move Struktur (Bild aus [CHH96] mit kleinen Modifikationen)

2.6 Regelungs-Architekturen

Im Jahr 1980 führten Sanderson und Weiss [SW80] eine Kategorisierung der Visual Servo Systeme ein. Die Zugehörigkeit eines Systems zu einer der Kategorien ergibt sich aus der Antwort auf zwei Fragen:

1. Ist die Regelungsstruktur hierarchisch? Gibt also das Visual Servo System Zielpunkte für die untergelagerte Achsregelung vor, oder werden die Achsen direkt gesteuert?
2. Ist das Fehlersignal in kartesischen Koordinaten oder in Bildmerkmalen definiert?

Die möglichen Antworten auf diese Fragen führen zu vier Hauptkategorien, die im folgenden beschrieben werden und in den Abbildungen 2.4 bis 2.7 schematisch dargestellt sind.

Die Abbildungen 2.4 und 2.5 zeigen eine hierarchische Regelungsarchitektur. Das heißt, die untergelagerte Achsregelung nutzt die Messwerte der Achsen, um den Roboter intern zu stabilisieren, und das Visual Servo System gibt nur die Zielpunkte vor. Dies wird als „Look-And-Move“ Struktur bezeichnet. Im Gegensatz dazu wird beim „Direct Visual Servo“* der interne Regler des Roboters komplett ersetzt, wie in den Abbildungen 2.6 und 2.7 zu sehen. Somit wird das gesamte System ausschließlich durch die visuelle Wahrnehmung stabilisiert.

Aus mehreren Gründen nutzen die meisten Implementierungen den Look-And-Move Ansatz. Zum ersten ist die Abtastrate des Kamerasystems normalerweise deutlich geringer als die der Achsregelung, was die direkte Regelung eines End-Effektors mit komplexer, nichtlinearer Dynamik extrem herausfordernd macht. Benutzt man hingegen die internen Messwerte mit einer hohen Abtastrate, so steht dem Visual Servo System eine idealisierte Achsdynamik zur Verfügung [And88]. Zum zweiten besitzen viele Roboter bereits eine Schnittstelle, über die kartesische Geschwindigkeiten oder Positionsschritte angegeben werden können. Dadurch wird die Implementierung deutlich vereinfacht und macht die entwickelten Methoden portierbarer. Zum dritten versteckt die Look-And-Move Struktur

*Sanderson und Weiss benutzten den Ausdruck „Visual Servo“ für diese Art der Architektur, aber seitdem hat sich dieser Begriff als generelle Bezeichnung für die Regelung eines Roboters mit Hilfe einer Kamera etabliert. Deshalb wird hier der Ausdruck „Direct Visual Servo“ verwendet, um Missverständnisse zu vermeiden.

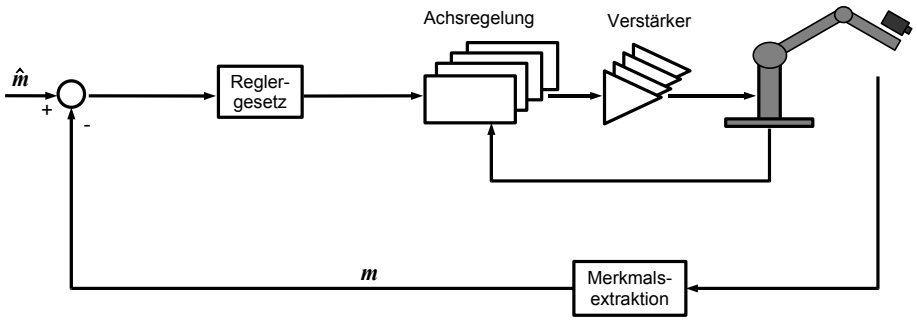


Abb. 2.5: Bildbasierte Look-And-Move Struktur (Bild aus [CHH96] mit kleinen Modifikationen)

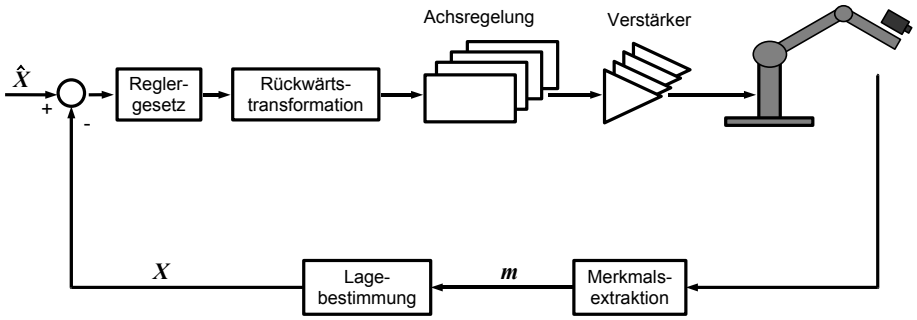


Abb. 2.6: Positionsbasierte Direct Visual Servo Struktur (Bild aus [CHH96] mit kleinen Modifikationen)

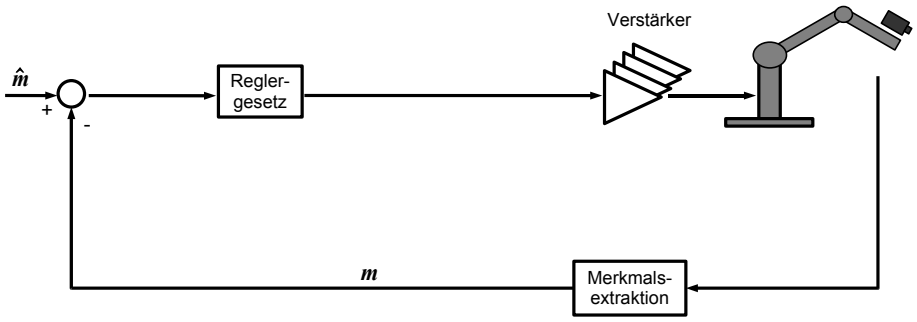


Abb. 2.7: Bildbasierte Direct Visual Servo Struktur (Bild aus [CHH96] mit kleinen Modifikationen)

die kinematischen Singularitäten vor dem Visual Servo System, weil viele Achsregelungen [Whi72] bereits spezielle Mechanismen zum Umgang mit Singularitäten besitzen [CSS91]. Dadurch wird das Systemdesign weiter vereinfacht.

Eine besonderer Fall der Look-And-Move Struktur ist die so genannte „Look-Then-Move“ Struktur. Dabei wird das Visual Servo System so implementiert, dass es erst dann mit dem nächsten Regelungsschritt beginnt, wenn die untergelagerte Achsregelung die vorgegebene Zielposition ausgeregelt hat. Dies kann je nach Regelverfahren die Imlementierung des

Systems deutlich vereinfachen, was durch eine erhöhte Regeldauer und eine abgehackte Bewegung erkauft wird.

Die beiden oben vorgestellten Klassifizierungen werden jeweils in „positionsbasiert“ (Abbildungen 2.4 bzw. 2.6) und „bildbasiert“ (Abbildungen 2.5 bzw. 2.7) unterteilt. Bei der positionsbasierten Regelung werden Merkmale aus dem Bild extrahiert, woraus zusammen mit dem geometrischen Modell des Objektes und dem Modell der Kamera, die relative Lage des Objektes zur Kamera approximiert wird. Mit dieser Schätzung werden die Stellgrößen so gewählt, dass der Fehler in kartesischen Koordinaten verkleinert wird.

Bei der bildbasierten Regelung werden die Stellgrößen hingegen direkt aus den Bildmerkmalen berechnet, so dass die Abweichung von den Sollmerkmalen im Bild reduziert wird. Durch die geeignete Wahl der Bildmerkmale ergibt sich dadurch mittelbar eine Bewegung des Roboters zu seiner Zielposition. Der bildbasierte Ansatz kann die Rechenverzögerung reduzieren, benötigt keine Lageschätzung und somit kein geometrisches Modell des Objektes, was den Aufwand der Modellierung einspart und robuster gegen Rauschen in den Merkmalspositionen ist. Ausserdem wird der Fehler aus der Sensormodellierung und Kamerakalibrierung reduziert. Er stellt allerdings eine größere Herausforderung an das Reglerdesign.

2.7 Jacobi-Matrix

Abschließend soll nun eine weit verbreitete Methodik für einen bildbasierten Regler vorgestellt und auf einige damit verbundene Probleme eingegangen werden. In Abschnitt 2.4 wurde gezeigt, dass es eine Abbildung $\mathbf{B} : T \rightarrow M$ gibt, die die Position und Orientierung des End-Effektors auf die korrespondierenden Bildmerkmalsparameter abbildet. Sei $\mathbf{X}$ die Kameraposition in kartesischen oder Achskoordinaten und $\mathbf{m}$ der Merkmalsvektor, dann gilt $\mathbf{m} = \mathbf{B}(\mathbf{X})$.

Als Regelziel wird eine Zielposition für die Kamera $\hat{\mathbf{X}}$ gewählt und die Sollmerkmale $\hat{\mathbf{m}} = \mathbf{B}(\hat{\mathbf{X}})$ in dieser Position ermittelt. Als Eingabe erhält der Regler also den Merkmalsfehler $\Delta\mathbf{m} = \hat{\mathbf{m}} - \mathbf{m}$ als aktuelle Differenz zwischen den Sollmerkmalen und den Istmerkmalen. Die Ausgabe des Reglers sind Kamerabewegungen $\Delta\mathbf{X}$, die das Ziel haben, den Merkmalsfehler zu eliminieren. Bei der richtigen Wahl der Merkmale befindet sich die Kamera dann in ihrer Zielposition $\hat{\mathbf{X}}$.

Für den Reglerentwurf wird nun die Taylorentwicklung in der aktuellen Kameraposition $\mathbf{X}_{akt}$ bis zum ersten Glied betrachtet

$$\mathbf{B}(\mathbf{X}) = \mathbf{B}(\mathbf{X}_{akt}) + \mathbf{J}(\mathbf{X} - \mathbf{X}_{akt}) + \mathbf{R}_2(\mathbf{X}, \mathbf{X}_{akt})$$

Gilt $\mathbf{X} \approx \mathbf{X}_{akt}$ dann kann $\mathbf{R}_2$ vernachlässigt werden. Die so genannte Jacobi-Matrix $\mathbf{J}$ ist gegeben durch $J_{j,i} = \frac{\partial B_j}{\partial x_i}$ und gibt für kleine Bewegungen der Kamera die zugehörige Bewegung der Bildmerkmale an. Es gilt also in einer gewissen Umgebung um die aktuelle Position der Kamera

$$\Delta\mathbf{m} \approx \mathbf{J} \cdot \Delta\mathbf{X}$$

Daraus ergibt sich das Reglergesetz

$$\Delta\mathbf{X} = -K \cdot \mathbf{J}^{\dagger} \cdot \Delta\mathbf{m} \tag{2.1}$$

Dabei ist K ein konstanter Verstärkungsfaktor, der die Schrittweite in einem Regelschritt vorgibt. Je besser die lineare Approximation mit der Jacobi-Matrix ist, um so größer kann K gewählt werden, und um so schneller wird das Ziel erreicht. Da meist mehr Merkmale verwendet werden, als der Roboter Freiheitsgrade hat ($dim(\Delta\mathbf{X}) < dim(\Delta\mathbf{m})$), ist die Jacobi-Matrix in der Regel nicht quadratisch. $\Delta\mathbf{m} = \mathbf{J} \cdot \Delta\mathbf{X}$ bildet also ein überbestimmtes Gleichungssystem. Von den möglichen Lösungswegen wird in der bildbasierten Regelung meist das Verfahren der Pseudoinversen $\mathbf{J}^{\dagger}$ verwendet.

Wie die Jacobi-Matrix berechnet wird, ist abhängig von der Wahl der Merkmalsparameter. Oft werden die Koordinaten von Merkmalspunkten in der Bildebene benutzt. Für einen Merkmalspunkt an der Stelle $[u, v]^T$ wird die Jacobi-Matrix bei der Verwendung von kartesischen Koordinaten berechnet durch

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & \frac{-u}{z} & \frac{-uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{z} & \frac{-v}{z} & \frac{-\lambda^2-v^2}{\lambda} & \frac{uv}{\lambda} & u \end{bmatrix} \cdot \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.2}$$

z ist dabei die Z-Komponente der Position des zugehörigen physikalischen Merkmals im Koordinatensystem der Kamera und λ die Brennweite der benutzen Linse. T_x , T_y und T_z sind die translatorischen und ω_x , ω_y und ω_z die rotatorischen Geschwindigkeiten der Kamera. Die gesamte Jacobimatrix für n Merkmalspunkte besteht aus $2 \cdot n$ Zeilen, wobei die Zeilen $2 \cdot i - 1$ und $2 \cdot i$ dem i-ten Merkmalspunkt zugeordnet sind und entsprechend 2.2 berechnet werden.

Diese weit verbreitete Methodik für einen bildbasierten Regler hat vor allem zwei Schwächen in der Auge-In-Hand Konfiguration. Zum einen kann der Informationsverlust bei der Projektion der Merkmale dazu führen, dass zwei Unterschiedliche Bewegungen der Kamera in kartesischen Koordinaten zu fast den gleichen Bewegungen der Merkmale in Bildkoordinaten führt. Das kann lokale Minima ergeben, in denen der Merkmalsfehler trotz einer relativ großen Auslenkung der Kamera sehr klein ist. Auf dieses Problem wird im Abschnitt 5.4 am konkreten Regler genauer eingegangen.

Das zweite Problem ergibt sich aus der gewählten Methodik. Bei dieser werden die Kamerabewegungen so gewählt, dass möglichst alle Merkmale gleichzeitig auf direktem Weg in ihre Sollwerte überführt werden. Dies entspricht im Allgemeinen aber nicht dem optimalen Weg der Kamera im Raum zu ihrer Zielposition. Auf dem besten Weg der Kamera beschreiben die Merkmalsparameter beliebige Kurven im Merkmalsraum. Gegebenenfalls müssen sich mehrere oder alle Merkmale zwischenzeitlich sogar von ihrer Zielposition entfernen.

Oft führt dieses Problem lediglich dazu, dass die von der Kamera beschriebene Trajektorie nicht optimal ist. In manchen Situationen erreicht die Kamera ihre Zielposition aber auch gar nicht. Das klassische Beispiel für diesen Fall ist das so genannte Kamera-Rückzugs-Problem, das in Abbildung 2.8 illustriert wird. Bei diesem Problem ist die Kamera um

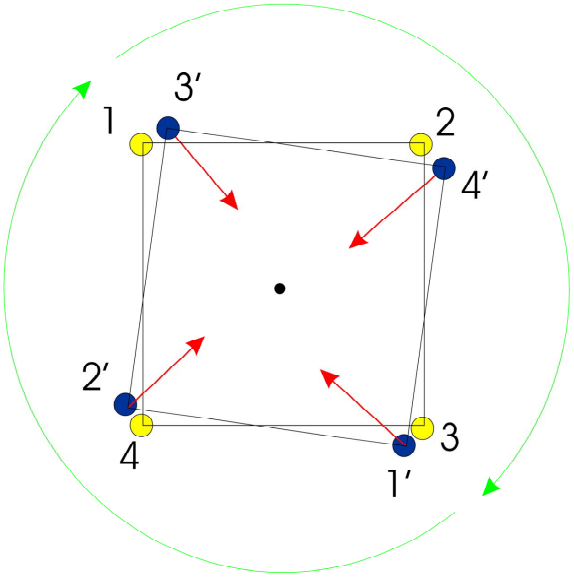


Abb. 2.8: Darstellung der Merkmale im Bild beim Kamera-Rückzugs-Problem: gelbe Punkte = Sollpositionen, graue Punkte = Istpositionen beim Start, rote Pfeile = optimale Bewegung der Merkmale im Bildraum, grüne Pfeile = optimale Bewegung der Kamera

etwa 180° um die Tiefenachse gedreht, ansonsten befindet sie sich in ihrer Zielposition. Die Merkmale sind so gewählt, dass sie auf den Ecken eines Quadrates liegen, dessen Mittelpunkt genau in der optischen Achse der Kamera fällt und die eine Ebene aufspannen, die sich parallel zur Bildebene befindet.

Durch die Rotation fällt jedes Merkmal ungefähr auf die Sollposition des diagonal gegenüberliegenden Merkmals. Der direkte Weg der Merkmalspunkte zu ihren Sollpositionen führt also über die Diagonalen. Dies entspricht im euklidischen Raum einer Distanzierung der Kamera von den Merkmalen weg entlang der optischen Achse, bis sich, wenn sich die Kamera unendlich weit entfernt hat, alle Merkmale im Mittelpunkt treffen. Die Kamera entfernt sich also mit diesem Regler beliebig weit von ihrer Sollposition weg, anstatt sich einfach um etwa 180° um die Tiefenachse zu drehen, was die optimale Bewegung im euklidischen Raum wäre. Für den Roboter bedeutet dies, dass die Bewegung in einer Singularität endet.

3 Merkmalsextraktion

Bevor im Folgenden der bildbasierte Regler entwickelt wird, wird erstmal ein zentrales Problem des im Abschnitt 1.2 beschriebenen Szenarios behandelt. Wie dort dargelegt, ist es notwendig, dass der Algorithmus zur Merkmalsextraktion ohne künstliche Merkmale auskommt. Das heißt auf dem relevanten Objekt dürfen keine zusätzlichen, leicht zu extrahierenden Farbflächen oder Ähnliches angebracht werden.

Es wird also ein Algorithmus benötigt, der aus den meisten, handelsüblichen Objekten mit Texturen* Merkmalspunkte robust extrahiert. Dies bedeutet, dass der Algorithmus die Punkte auch dann als markante Punkte im Bild wiedererkennt, wenn sich der Abstand der Kamera zum Objekt erhöht hat (Skalierung), die Kamera um die Tiefenachse rotiert wurde, die Position der Kamera anderweitig verändert wurde (affine Transformation) oder sich die Beleuchtung geändert hat. Darüber hinaus muss der Algorithmus eine Beschreibung des Merkmalspunktes erzeugen, die sich bei den oben beschriebenen Veränderungen möglichst wenig verändert, damit die selben Punkte in unterschiedlichen Bildern auch als solche erkannt werden. Dies wird als Korrespondenzproblem bezeichnet.

Diese Anforderungen erfüllt die „Scale-Invariant Feature Transformation“, kurz „SIFT“, von Lowe [Low04], die in der Objekterkennung eingesetzt wird. Im Abschnitt 3.1 wird die Funktionsweise des Algorithmus vorgestellt und danach kurz auf die Anwendung eingegangen. Im letzten Abschnitt wird demonstriert, wie mit Hilfe des perspektivischen Kameramodells in der Simulation frei im Raum positionierte Merkmale auf die Bildebene abgebildet werden. Damit kann zu Testzwecken die Merkmalsposition mit einem geringeren Rechenaufwand ermittelt werden.

Der SIFT-Algorithmus stellt allerdings erst den Beginn der Auswahl der Merkmalspunkte dar, weil er in der Regel sehr viel mehr Punkte erzeugt, als für die bildbasierte Regelung benötigt werden. Da nicht alle Merkmalspunkte gleich gute Kandidaten sind, wird im folgenden Kapitel darauf eingegangen, was gute Merkmalspunkte sind und wie diese für die bildbasierten Regelung automatisch selektiert werden.

3.1 Scale-Invariant Feature Transformation (SIFT)

Der SIFT-Algorithmus ermittelt die stabilen Merkmalspunkte, hier Keypoints genannt, in vier Schritten:

1. Extraktion von skalierungsinvarianten Keypoint-Kandidaten durch Extremasuche im Skalenraum†

*Besonders geeignet sind unstrukturierte Objekte mit unregelmäßiger Textur.

†Auf die genaue Erläuterung des Begriffs „Skalenraum“ und der dahinter stehenden Theorie soll in dieser Zusammenfassung verzichtet werden. Für ein tiefer gehendes Verständnis der SIFT-Merkmale ist dies aber hilfreich. Dafür sei auf [Wit83, Low04] verwiesen.

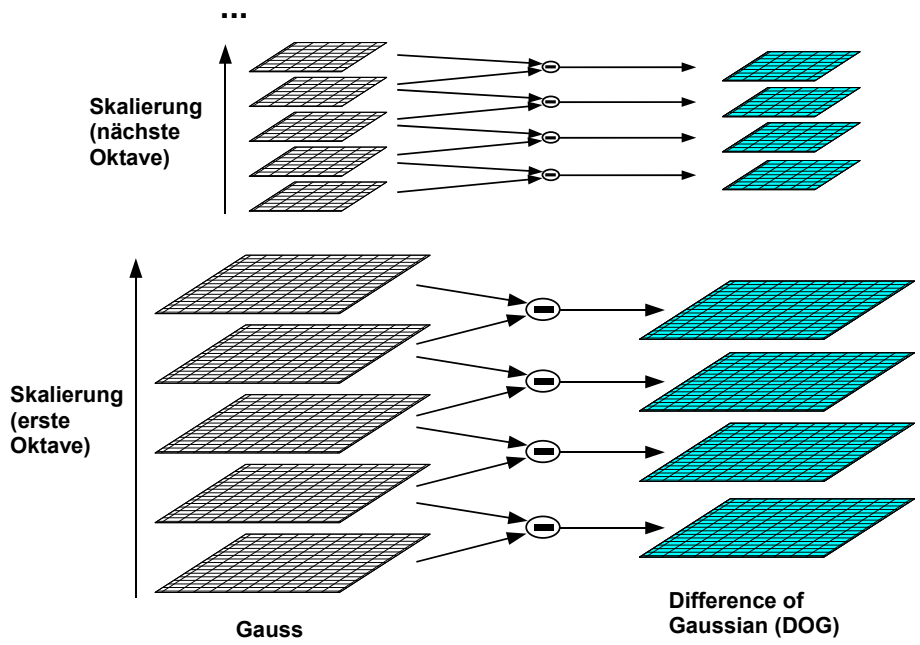


Abb. 3.1: Berechnung der DOG Bilder (Bild aus [Low04])

2. Elimination von Keypoint-Kandidaten auf Grund von fehlender Stabilität oder Kontrast

3. Zuweisung einer Hauptorientierung zu jedem Keypoint

4. Berechnung eines SIFT-Vektors* zur Beschreibung des Keypoints

Diese Schritte werden im Folgenden einzeln erläutert.

3.1.1 Extraktion von Keypoint-Kandidaten

Im ersten Schritt der Merkmalsextraktion werden Orte und Skalierungen ermittelt, die aus verschiedenen Ansichten des Objektes wiedergefunden werden können. Dazu wird die Difference-of-Gaussian (DOG) Funktion gefaltet mit dem Bild genutzt, da sie besonders effizient berechnet werden kann und eine gute Approximation der skalierungsnormalisierten Laplacian-of-Gaussian $\sigma^2\nabla^2 G$ ist. Lindeberg [Lin94] zeigte, dass damit echte Skalierungsinvarianz erreicht werden kann. Darüber hinaus bewies Mikolajczyk [Mik02], dass dies die stabilsten Merkmale erzeugt, im Vergleich zu Funktionen wie Gradient, Hessematrix oder Harris-Ecken-Funktion.

Zur Berechnung wird zunächst das Ausgangsbild sukzessive mit einer Gaussfunktion gefaltet, was dem unteren linken Teil der Abbildung 3.1 entspricht. Dies ist eine approximierte Tiefpassfilterung, wobei in jedem gefalteten Bild die höchsten Frequenzen aus dem

*Um den Vektor von Merkmalen und den Vektor zur Beschreibung eines Merkmals unterscheiden zu können, wird letzter hier nicht Merkmalsvektor sondern SIFT-Vektor genannt.

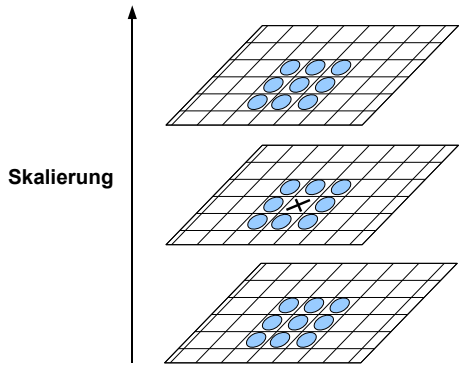


Abb. 3.2: Dreidimensionale Extremasuche in den DOG Bildern (Bild aus [Low04])

vorherigen Bild wegfallen. Diese bilden die erste Oktave des Skalenraums. Die Difference-of-Gaussian Funktion gefaltet mit dem Bild ergibt sich nun aus der Diffenenz zweier aufeinander folgender Bilder in dieser Oktave, wie in Abbildung 3.1 unten rechts zu sehen ist. Dies entspricht der Approximation eines Bandpassfilters.

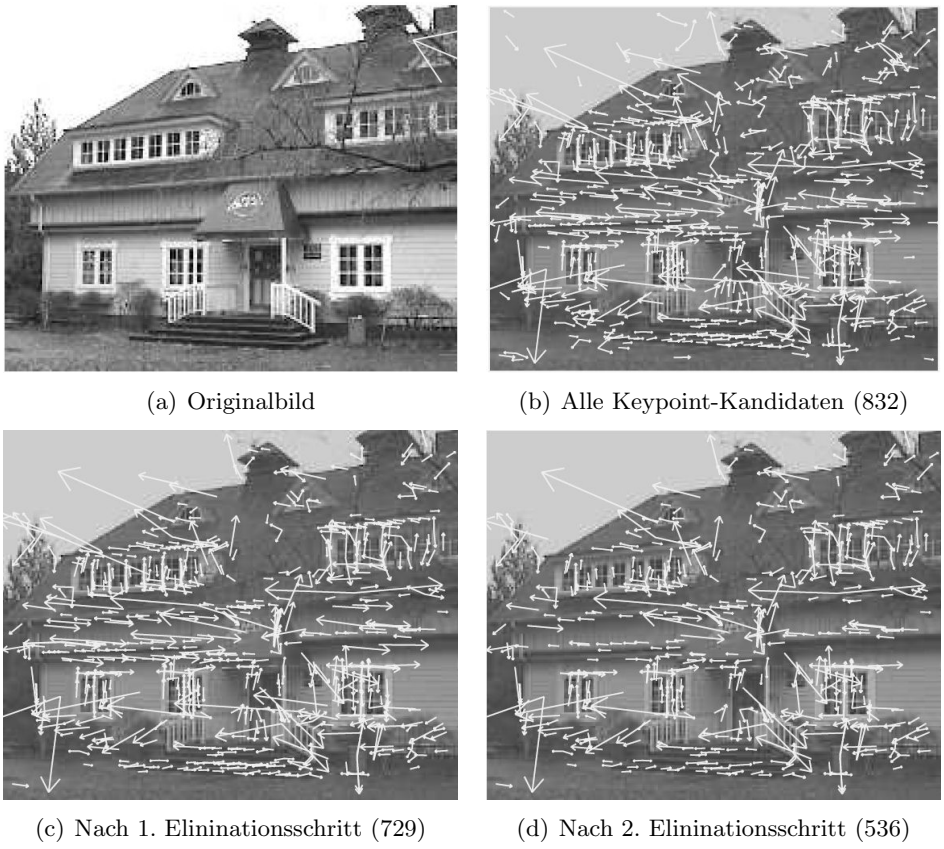
Die stabilen Keypoint-Kandidaten für diese Oktave werden detektiert, indem dreidimensionale Extrema in den DOG Bildern ermittelt werden. Dies bedeutet, dass ein Punkt mit den acht Nachbarn in seinem eigenen Bild und mit den jeweils neun Nachbarn in dem DOG Bild oberhalb und unterhalb verglichen (siehe Abbildung 3.2) und nur dann ausgewählt wird, wenn er größer oder kleiner als alle Nachbarn ist. Der Rechenaufwand dieses Tests ist relativ gering, da die meisten Punkte bereits nach dem Vergleich mit wenigen Nachbarn herausfallen.

Das Startbild für die nächste Oktave ergibt sich dann aus dem einmal mit der Gaussfunktion gefalteten Bild, indem nur jeder zweite Pixel in jeder Spalte und Zeile genommen wird. Die Größe des Bildes wird also in beiden Dimensionen halbiert. Mit diesem Bild wird dann analog zu oben verfahren und somit sukzessive weitere Keypoint-Kandidaten generiert, was der obere Teil der Abbildung 3.1 andeuten soll. Die Bildgröße wird so lange verkleinert, bis keine ausreichende Umgebung für die Definition eines SIFT-Vektors mehr vorhanden ist.

Die Anzahl der stabilen Merkmale kann nochmals um etwa den Faktor vier erhöht werden, wenn die Größe des Bildes am Anfang in beiden Dimensionen durch ein Interpolationsverfahren verdoppelt wird. Diesem Gewinn an stabilen Merkmalen steht allerdings die ungefähre Vervierfachung der Rechenzeit gegenüber. Eine genaue Komplexitätsanalyse ist noch Gegenstand der Forschung. Weitere Erhöhungen der Pixelzahl bringen keinen signifikanten Vorteil mehr.

3.1.2 Elimination von Keypoint-Kandidaten

Es hat sich gezeigt, dass das im vorherigen Abschnitt beschriebene Verfahren sehr viele stabile Keypoints erzeugt. In der Menge der Kandidaten befinden sich aber immer noch eine ganze Reihe von wenig robusten Merkmalen. In den nachfolgenden Schritten werden viele dieser instabilen Keypoint-Kandidaten erkannt und eliminiert.



(a) Originalbild (b) Alle Keypoint-Kandidaten (832)

(c) Nach 1. Elininationsschritt (729) (d) Nach 2. Elininationsschritt (536)

Abb. 3.3: Schrittweise Elimination der Keypoint-Kandidaten (Bild aus [Low04])

Zunächst werden die Kandidaten aussortiert, die einen zu geringen Kontrast zu ihrer Umgebung haben und somit anfällig gegen Rauschen sind. Wie Brown und Lowe [BL02] gezeigt haben, ist ein Keypoint sehr viel stabiler, wenn er nicht am mittleren Abtastpunkt positioniert wird, sondern im Maximum einer dreidimensionalen, quadratischen Funktion, die die umgebenen Abtastpunkte interpoliert. Die so ermittelten Merkmalspunkte sind also subpixel und subskalen genau. Darüber hinaus bildet der Wert dieser Funktion im Maximum ein gutes Maß dafür, ob ein Keypoint zu geringen Kontrast besitzt. Ein Beispiel für die Reduktion der Anzahl der Merkmale ist in Abbildung 3.3 von (b) nach (c) zu sehen.

Im zweiten Schritt werden Keypoints eliminiert, die auf einer Kante liegen. Die DOG Funktion hat große Werte entlang von Kanten, auch wenn die Position eines Merkmals auf der Kante nur schlecht bestimmt ist und sich schon bei geringen Änderungen deutlich verschieben kann. Ein solches schlechtes Extremum in der DOG Funktion wird dadurch erkannt, dass hier die Hauptkrümmung entlang der Kante groß ist im Vergleich zur Hauptkrümmung im rechten Winkel dazu. Da die Hauptkrümmungen proportional zu den Eigenwerten der Hessematrix sind, wird das Verhältnis der Hauptkrümmungen aus dem Verhältnis der Eigenwerte berechnet. Die Auswirkung dieses Schrittes ist in Abbildung 3.3 von (c) nach (d) zu sehen.

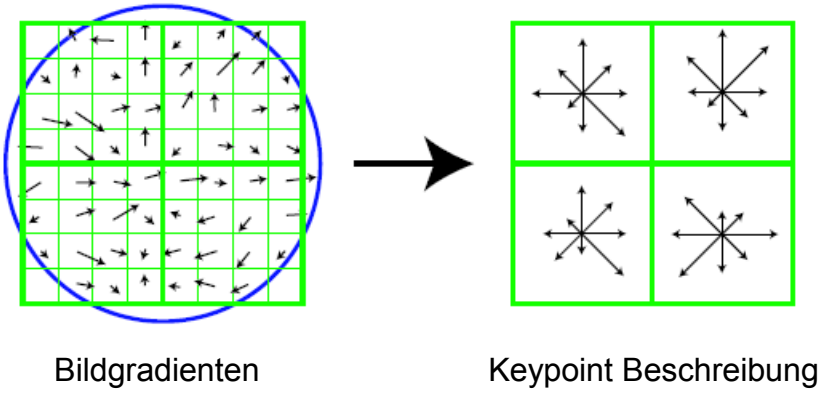


Abb. 3.4: Bestimmung des SIFT-Vektors (Bild aus [Low04])

3.1.3 Zuweisung der Hauptorientierung

Die noch verbliebenen Keypoint-Kandidaten bilden die vom SIFT-Algorithmus extrahierten Merkmale. Nun wird für jedes Merkmal ein SIFT-Vektor bestimmt, durch den das gleiche Merkmal in verschiedenen Bildern als identisch erkannt werden kann.

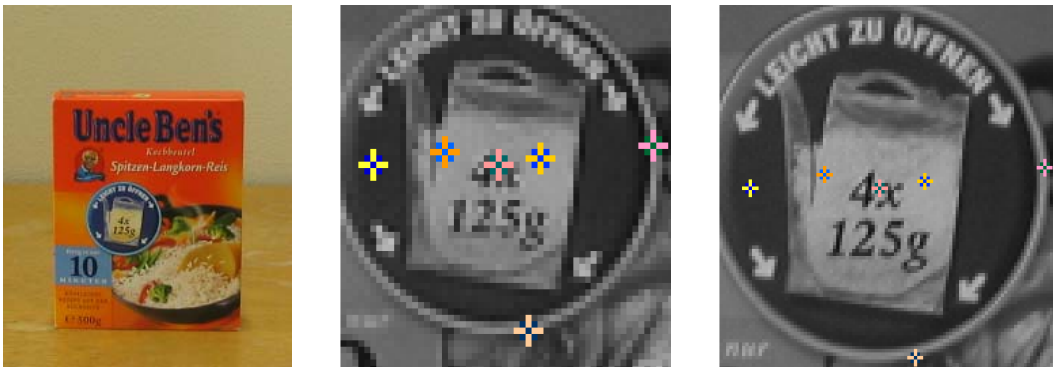
Dazu wird zunächst die Hauptorientierung eines Merkmals aus der lokalen Umgebung berechnet. Der SIFT-Vektor wird dann relativ zu dieser Hauptorientierung ausgedrückt und ist somit invariant gegen eine Rotation des Bildes. Die Skalierungsinvarianz wird durch die Berechnung der Hauptorientierung in dem gaussgeglätteten Bild erreicht, welches der Skalierung des Merkmals am nächsten kommt.

Zur Ermittlung der Hauptorientierung wird in der Umgebung um den Keypoint in jedem Abtastpunkt der Betrag und die Orientierung des Gradienten berechnet. Aus diesen wird ein Orientierungshistogramm gebildet, dass aus 36 diskreten Winkelbereichen besteht, die den gesamten Umfang, also 360°, abdecken. Dabei wird der Betrag des Gradienten von jedem hinzugefügten Abstastpunkt abhängig vom Abstand zum Keypoint mit einer Gaussfunktion gewichtet, dessen σ 1,5-mal so groß ist wie die Skalierung des Keypoints.

Maxima in diesem Histogramm entsprechen also dominanten Richtungen der lokalen Gradienten. Zunächst wird das globale Maximum im Histogramm ermittelt. Für jedes lokale Maximum, dass mindestens 80% der Höhe des globalen Maximums erreicht, wird ein Keypoint mit der entsprechenden Orientierung angelegt. Die Orientierung eines Keypoints ergibt sich, indem eine Parabel durch die drei Werte am nächsten zum jeweiligen Maximum gelegt und so die genaue Position des Maximums interpoliert wird.

3.1.4 Bestimmung des SIFT-Vektors

Abschließend wird nun für jedes Merkmal der SIFT-Vektor berechnet. Dazu wird das gaussgeglättete Bild verwendet, das der Skalierung des Keypoints am nächsten kommt, um Skalierungsinvarianz zu gewährleisten. Wie in Abbildung 3.4 auf der linken Seite dargestellt, werden zunächst wieder die lokalen Gradienten mit Betrag und Richtung in einer Umgebung um den Keypoint berechnet, wobei die Beträge mit einem runden Gaussfenster



(a) Objektansicht (b) Ist-Merkmale (c) Soll-Merkale

Abb. 3.5: Skalierungsinvarianz

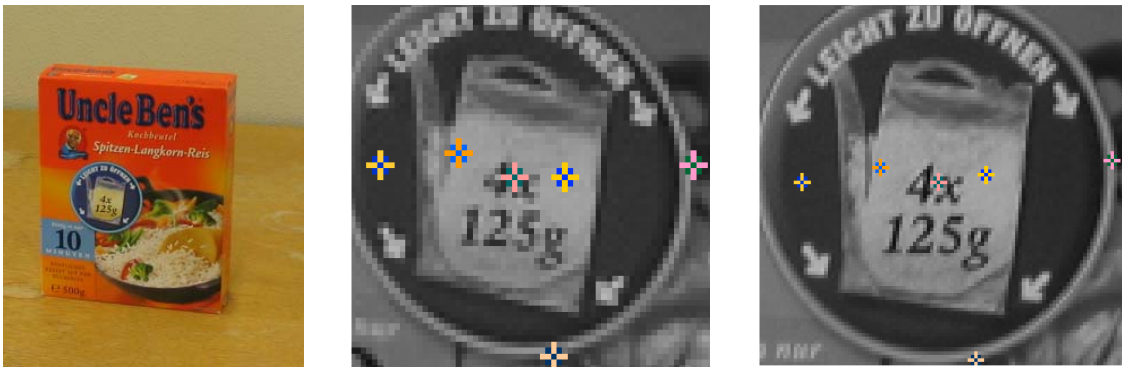
(a) Objektansicht (b) Ist-Merkmale (c) Soll-Merkale

Abb. 3.6: Rotationsinvarianz

gewichtet werden (der Kreis in der Abbildung). Für die Rotationsinvarianz werden die Positionen der Punkte und die Orientierungen der Gradienten entsprechend der Orientierung des Keypoints rotiert.

Dieser Bildausschnitt wird nun in gleich große Bereiche unterteilt. In Abbildung 3.4 ist dies für eine 2×2 Matrix vereinfacht gezeigt. In der Praxis wird eine 4×4 Matrix verwendet, wobei jedes Feld 4×4 Abtastpunkte enthält. Für jeden Bereich wird ein Histogramm mit acht möglichen Richtungen erstellt. Nach einer trilinearen Interpolationen bilden die Einträge in den Histogrammen den SIFT-Vektor, der somit eine Größe von $4 \cdot 4 \cdot 8 = 128$ Elementen hat. Am Ende wird dieser Vektor normalisiert und die Größe der Einträge begrenzt, um die Invarianz gegen Beleuchtungsänderungen zu verbessern.

Das Ergebnis ist in den Abbildungen 3.5 bis 3.8 an Beispielen verdeutlicht. Trotz der Änderungen in der Ansicht des Objektes können die Soll-Merkmale im aktuellen Kamerabild wiedergefunden und korrekt zugeordnet werden, was an der Farbcodierung der Makierungen zu erkennen ist. Das Verfahren zur Zuordnung der Merkmale im Ist- und Sollbild wird im Abschitt 4.1 erläutert.



(a) Objektansicht (b) Ist-Merkmale (c) Soll-Merkale

Abb. 3.7: Invarianz gegen affine Transformationen

(a) Objektansicht (b) Ist-Merkmale (c) Soll-Merkale

Abb. 3.8: Invarianz gegen Beleuchtungsänderungen

3.2 Anwendung

Als Implementierung wird die von Lowe für wissenschaftliche Zwecke zur Verfügung gestellt Umsetzung in C genutzt. Durch das Hinzufügen einer „mexFunction“ kann der Code in Matlab zu einer „mex-Datei“ compiliert werden. Dadurch steht die Funktion zur Berechnung der Merkmale in gleicher Art und Weise wie jede andere Matlab-Funktion zur Verfügung [Mat05]. Dies hat den Vorteil, dass der existierende Quellcode nicht als „m-Datei“ umgeschrieben werden muss und die Laufzeit deutlich reduziert wird.

Das konkrete Verhalten dieser Implementierung ist von einer Vielzahl von Parametern abhängig, die sowohl die Anzahl als auch die Qualität der gefunden Keypoints. So können einzelne Teile des Algorithmus wie das Interpolieren des Bildes zu Beginn des Prozesses auf die vierfache Pixelanzahl an- beziehungsweise abgeschaltet oder verschiedene Grenzwerte und Größen verändert werden. [Low04] beinhaltet bereits ausführliche Tests, in denen sinnvolle Werte für die Parameter als Defaulteinstellungen ermittelt werden.

Die meisten dieser Defaultwerte erwiesen sich auch für diese Arbeit als sinnvoll. Nur die Verdoppelung des Bildes in beiden Dimensionen zu Beginn der Merkmalsextraktion wurde abgeschaltet, da auch ohne dieses rechenintensive Verfahren in den durchgeführten Tests genügend Merkmale detektiert wurden. Des weiteren wurde die Interpolation der

Abb. 3.9: Dreidimensionales Modell des Roboterarms und eines Objektes mit realistischer Textur in der VR-Toolbox

Keypoint-Positionen eingeschaltet, um Subpixelgenauigkeit zu erreichen.

3.3 Simulation

Wie im Abschnitt 1.2 beschrieben, werden die Ergebnisse zunächst in einer Simulation erarbeitet. Dazu wird Matlab mit verschiedenen Toolboxen verwendet. Die so genannte „Virtual-Reality-Toolbox“ ermöglicht das Nachbilden einer dreidimensionalen Umgebung einschließlich realistischer Texturen wie in Abbildung 3.9 zu sehen. Damit werden die Bewegungen des Roboterarms und die Bilderzeugung der Kamera simuliert und der gesamte Regelkreis einschließlich der oben beschriebenen Bildverarbeitung in der Simulation nachgebildet.

Diese relativ genaue Abbildung der realen Situation lässt gute Schlüsse auf das Verhalten des Roboters in der Realität zu, ist aber andererseits sehr rechenintensiv. Gerade in den ersten Entwicklungs- und Implementierungsschritten sowie bei ausführlichen Tests wirkt sich dieser hohe Zeitaufwand gravierend aus. Dabei nimmt die Bilderzeugung und die Bildverarbeitung die meiste Zeit in Anspruch. Deshalb wird zusätzlich eine alternative Möglichkeit implementiert, um die Merkmalspositionen im Bild zu bestimmen. Diese Methodik kann nur in der Simulation verwendet werden, da die Positionen der Merkmale im Raum bekannt sein müssen. Mit dieser Information und der aktuellen Lage der Kamera werden die Merkmalspositionen im Bild mit wenig mathematischen Operationen bestimmt, wie in Abschnitt 2.2 beschrieben. Die „Epipolar Geometry Toolbox“ [MP05, MP06] stellt eine entsprechende Funktion zur Verfügung.

Diese Methodik hat neben dem um mehrere Größenordnungen gesenkten Zeitaufwand

noch weiter Vorteile im Entwicklungsprozess.

Zum ersten werden die Positionen bis auf kleine Rechenungenauigkeiten exakt berechnet und unterliegen somit nicht dem Rauschen wie bei der Bestimmung mit Hilfe der Bildverarbeitung. Somit können Aussagen über das Verhalten des Reglers ohne Merkmalsrauschen getroffen werden. Soll hingegen gerade die Robustheit gegen Rauschen untersucht werden, kann ein künstlicher Positionsfehler genau in der gewünschten Größenordnung aufaddiert werden.

Zum zweiten müssen die Merkmale nicht wie bei dem SIFT-Algorithmus auf Objekten an bestimmten Stellen der Textur liegen, sondern können frei im Raum platziert werden.

Zum dritten kann bei diesem Verfahren ein Merkmalspunkte nicht verloren gehen. Es kann im Gegensatz zur Bildverarbeitung nicht vorkommen, dass ein Merkmal zeitweise nicht erkannt, nicht korrekt zugeordnet oder von einem Objektteil verdeckt wird. Außerdem wird bei dieser Projektion eine unendlich große Bildebene angenommen, so dass die Merkmale das Bild nicht verlassen können. Die einzige Ausnahme ist, dass ein Merkmalspunkt nicht auf die Bildebene projiziert werden kann, weil die Linie zwischen Merkmal und Augpunkt parallel zur Bildebene verläuft, also $z = 0$ ist.

4 Automatische Merkmalsauswahl

Der SIFT-Algorithmus ermittelt im Kamerabild eine Menge von stabilen Merkmalen und für jedes Merkmal einen beschreibenden Vektor, um das Korrespondenzproblem zu lösen. Abhängig von der Textur und der Wahl der Parameter des Algorithmus beinhaltet ein typisches Bild der Größe 500×500 Pixel bis zu 2000 stabile SIFT-Merkmale ([Low04]). Ziel der automatischen Merkmalsauswahl ist, aus diesen Merkmalen eine Teilmenge auszuwählen, die für die bildbasierte Regelung am besten geeignet ist, zum Beispiel bezüglich der Eindeutigkeit.

Die in den nächsten beiden Kapiteln vorgestellten Reglerkonzepte besitzen zwar keine Obergrenze für die Anzahl der verwendbaren Merkmale. Aber ab einer bestimmten Menge erhöht sich nur noch der Rechen- und Speicheraufwand, ohne dass die Qualität des Reglers signifikant steigt. Zudem wächst die Wahrscheinlichkeit von weniger eindeutigen Merkmalen und somit von Fehlern bei der Lösung des Korrespondenzproblems. Folglich wird für eine stabile Regelung nur einen Teil der Merkmale verwendet.

Shademan und Janabi-Sharifi [SJS04] beschreiben dazu ein Verfahren, dass gute SIFT-Merkmale entlang einer vorgegeben Trajektorie ermittelt. Dazu gehen sie davon aus, dass sich ein Merkmal in einem Schritt nur eine maximale Distanz im Bild bewegt, wenn es richtig zugeordnet wird. Dieses Verfahren eignet sich aber wenig, um Merkmale für einen ganzen Arbeitsbereich auszuwählen. Außerdem lässt sich ohne genaues Wissen über die Entfernung zum betrachteten Objekt die maximal mögliche Distanz, die sich ein Merkmal im Bild innerhalb eines Schrittes bewegen kann, nur schlecht abschätzen.

Im folgenden wird eine alternative Methodik vorgestellt, die in der Lernphase die Sollmerkmale automatisch ermittelt. Dazu wird zunächst erläutert, wie das Korrespondenzproblem gelöst wird. In Abschnitt 4.2 ergeben sich daraus die Kriterien für die Merkmalsauswahl und der Algorithmus, der eine entsprechende Teilmenge der Merkmale bestimmt.

4.1 Korrespondenzproblem

Mit Hilfe des SIFT-Algorithmus werden im Zielbild Sollmerkmale extrahiert und deren SIFT-Vektoren ermittelt. Analog wird zu Beginn jedes Regelschritts mit dem aktuellen Kamerabild verfahren. Beim Korrespondenzproblem muss entschieden werden, welches Merkmal im aktuellen Bild welchem Merkmal im Zielbild entspricht. Dazu werden die SIFT-Vektoren verwendet. Allerdings ist der SIFT-Vektor des gleichen Merkmals in verschieden Bilder im Allgemeinen zwar sehr ähnlich aber nicht identisch. Außerdem muss nicht jedes Merkmal aus dem Zielbild im aktuellen Bild auftauchen und umgekehrt.

Die SIFT-Vektoren sind reellwertige Vektoren der Dimension 128. Dabei sind sich zwei Vektoren um so ähnlicher, je kleiner der Winkel zwischen ihnen ist. Der Winkel wird aus dem Skalarprodukt der normalisierten Vektoren berechnet und definiert die Ähnlichkeit

zweier SIFT-Vektoren:

$$\angle(\mathbf{v}_1, \mathbf{v}_2) = arccos\left(\frac{\mathbf{v}_1 \cdot \mathbf{v}_2^T}{|\mathbf{v}_1| \cdot |\mathbf{v}_2|}\right)$$

Alternativ kann die Ähnlichkeit über den euklidischen Abstand der Vektoren definiert werden. Doch das Skalarprodukt kann in Matlab deutlich effizienter berechnet werden und weißt die gleiche Qualität wie der euklidische Abstand auf.

Auf Grund der Dimension des Vektor ist es unwahrscheinlich, dass ein zufälliger Vektor einen kleinen Winkel zu einem gegeben Vektor aufweist. Wird also ein SIFT-Vektor aus dem aktuellen Bild mit allen SIFT-Vektoren aus dem Zielbild verglichen, dann ist es sehr wahrscheinlich, dass diejenigen mit der größten Ähnlichkeit, also dem kleinsten Winkel, dasselbe Merkmal beschreiben, sofern das Merkmal auch im Zielbild vorhanden ist.

Die Entscheidung, ob ein Merkmal zugeordnet wird oder nicht, wird dabei nicht auf Grund der absoluten Ähnlichkeit zweier Merkmale getroffen, sondern ergibt sich aus einer relativen Ähnlichkeit. Die beste Zuordnung wird akzeptiert, wenn die Ähnlichkeit deutlich größer ist als die Ähnlichkeit zum zweitbesten SIFT-Vektor im Referenzbild. Experimente haben gezeigt, dass dieses Kriterium eine zuverlässigere Zuordnung bietet als ein simpler Grenzwert ([Low04]).

Zusätzlich wird die Zuverlässigkeit erhöht,in dem das zugeordnet Merkmal im Zielbild mit allen Merkmalen im aktuellen Bild verglichen und nach dem gleichen Algorithmus einem oder gegebenenfalls keinem Merkmal zugeordnet wird. Nur für den Fall, dass hier die gleiche Übereinstimmung gefunden wird, wie in der umgekehrten Richtung, gelten die beiden Merkmale als identisch. Dies verhindert vor allem in dem Fall, dass im aktuellen Bild zufällig zwei sehr ähnliche Vektoren vorhanden sind, dass diese beide dem gleichen Merkmal im Zielbild zugeordnet werden.

4.2 Algorithmus

Zunächst werden die Hauptkriterien festgelegt, die die Auswahl der Merkmale erfüllen soll. Diese ergeben sich aus den Bedürfnissen der bildbasierten Regler und den Beschränkungen, die sich aus der Verwendung von natürlichen Texturen und des SIFT-Algorithmus ergeben.

Als erstes ist entscheidend, dass die Merkmale im Zielbild vorhanden sind. Sonst können keine Sollpositionen bestimmt werden. Darüber hinaus müssen die ausgewählten Merkmale über einen möglichst großen Bereich stabil wieder gefunden und zugeordnet werden können, wobei nicht verhindert werden kann, dass einige Merkmale in Teilbereichen des Arbeitsraums nicht erkannt werden. Besonders wichtig ist dabei, dass es zu keinen falschen Korrespondenz kommt.

Der SIFT-Algorithmus gewährleistet für jedes Merkmal einzeln betrachtet, dass dieses Merkmal stabil über einen großen Bereich von Objektansichten wiedererkannt wird und immer eine ähnliche Beschreibung erhält. Die korrekte Zuordnung eines Merkmals im aktuellen Bild zu seinem Referenzmerkmal ist aber auch von den anderen Merkmalen in der momentanen Ansicht und im Zielbild abhängig. Gibt es zum Beispiel ein zweites Merkmale mit einem sehr ähnlichen SIFT-Vektor, dann werden in vielen Fällen beide auf Grund des oben beschriebenen Kriteriums der relativen Ähnlichkeit nicht zugeordnet. Für eine Verbesserung muss also die gesamte Merkmalsmenge zusammen betrachtet werden.

Dazu wird zunächst die Menge der Referenzmerkmale mit den SIFT-Merkmalen des Zielbildes initialisiert. Da jedes Sollmerkmal im Zielbild vorhanden sein muss, ist dies die größte mögliche Referenzmenge. Im folgenden werden zwei Schritte vorgestellt, die aus dieser Menge automatisch die Merkmale entfernen, die die obigen Kriterien nicht erfüllen.

Im ersten Schritt werden nur die Merkmale aus dem Zielbild betrachtet. Jeder SIFT-Vektor wird mit einem kleinen zufälligen Fehler versehen und dann mit der Referenzdatenbank verglichen. Wird ein Merkmal dabei nicht korrekt sich selber zugeordnet, dann wird es nach diesem Schritt aus der Datenbank entfernt. Somit werden alle Merkmale eliminiert, deren Umgebungen ähnliche SIFT-Vektoren ergeben haben. Dies tritt zum Beispiel bei den Ecken eines Schachbrettmusters auf. Es ist nicht zu erwarten, dass diese Merkmale während der Regelung häufig detektiert werden, wenn schon ein kleiner Fehler im SIFT-Vektor Auswirkungen auf die korrekte Zuordnung hat. Viel mehr besteht die Möglichkeit, dass diese Vektoren korrekte Zuordnungen anderen Vektoren verhindern oder sogar für falsche Zuordnungen sorgen.

Es ist dabei wichtig, dass die Elimination eines Merkmals erst am Ende des Schritts erfolgt. Andernfalls würde bei einer Gruppe von ähnlichen SIFT-Vektoren der zuletzt betrachtete in der Datenbank verbleiben, da kein Ähnlicher mehr vorhanden ist. In einem aktuellen Kamerabild kann es dann vorkommen, dass aus dieser Gruppe nur ein Merkmal gefunden wird, das aber nicht in der Datenbank ist. Die Wahrscheinlichkeit ist hoch, dass dieses Merkmal fälschlicherweise dem in der Datenbank verbliebenen Merkmal aus der Gruppe zugeordnet wird. Nun beinhaltet die Datenbank nur noch Merkmale, deren Beschreibungen paarweise unähnlich sind.

Im nächsten Schritt werden die Merkmale entfernt, die über den Arbeitsbereich weniger stabil in ihrer Position und Beschreibung sind und somit zu falschen Zuordnungen führen. Dazu werden gleichverteilt im Arbeitsraum um die Zielpositon zusätzliche Bilder aufgenommen. Darin werden die SIFT-Merkmale bestimmt und den Merkmalen in der aktuellen Datenbank zugeordnet. Zwei Methoden werden angewendet, ein Winkelkriterium und Epipolargeometrie, um automatisch falsche Zuordnungen zu erkennen. In diesem Fall wird das zugeordnete Referenzmerkmal anschließend aus der Datenbank eliminiert.

Für das Winkelkriterium wird die Hauptorientierung der Merkmale aus dem SIFT-Algorithmus benutzt (siehe Abschnitt 3.1.3). Bei einer affinen Transformation ändert sich die Hauptorientierung der Merkmale im Bild um etwa den gleichen Winkel. Bei einer korrekten Zuordnung aller Merkmale bilden die Winkeldifferenzen zwischen Ist- und Sollorientierung eine Gaussverteilung mit einer geringen Varianz (siehe Abbildung 6.1 auf Seite 58). Weicht also die Winkeldifferenz einer Zuordnung mehr als ein erlaubten Grenzwert von dem Mittelwert ab, dann wird diese Zuordnung als falsch klassifiziert.

Der Vorteil dieses Verfahrens ist, dass es keine zusätzlichen Informationen benötigt und somit auch während der Regelung eingesetzt werden kann, um zur Laufzeit des Reglers falsche Zuordnungen zu erkennen. Andererseits hat es den Nachteil, dass falsche Zuordnungen schlecht erkannt werden, die sich aus ähnlichen Mustern innerhalb der Textur ergeben (zum Beispiel der gleiche Buchstabe in einem Schriftzug). In diesem Fall ist die Wahrscheinlichkeit hoch, dass beide Merkmale in etwa die gleiche Hauptorientierung besitzen. Diese Merkmale wurden aber entweder bereits im ersten Schritt entfernt, da das ähnliche Muster zu einem ähnlichen SIFT-Vektor geführt hat, oder die falsche Zuordnung wird durch das zweite Kriterium erkannt.

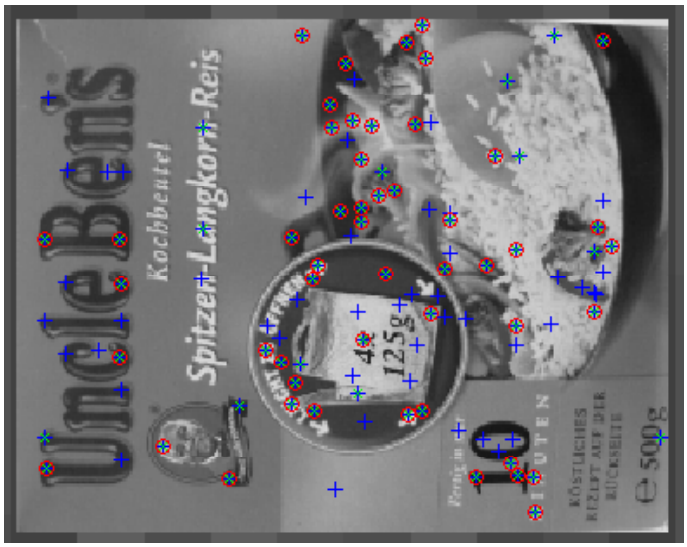


Abb. 4.1: Automatische Merkmalsauswahl

Das zweite Verfahren, um falsche Zuordnungen zu erkennen, verwendet die Epipolargeometrie. Dazu werden zusätzlich die Lagen der Kamera, in denen die Bilder aufgenommen wurden, und die intrinsischen Kameraparameter benötigt. Auf Grund dieser Zusatzinformationen kann dieses Kriterium allerdings nicht während der Regelung angewendet werden, und es ist abhängig von der Güte der Kamerakalibrierung.

Wie in Abschnitt 2.3 beschrieben kann aus den gegebenen Informationen für ein Merkmal im aktuellen Bild die Epipolarlinie im Zielbild bestimmt werden, auf der das zugehörige Referenzmerkmal liegen muss. Auf Grund von Kalibrierungsfehlern, Rechenungenauigkeiten und leichten Verschiebungen des Merkmals kann der Referenzpunkt auch bei einer korrekten Zuordnung neben der Epipolarlinie liegen. Somit wird ein Merkmal als falsch zugeordnet eingestuft, wenn sein Abstand zur Epipolarlinie einen Grenzwert übersteigt. Es ist möglich, dass falsch zugeordnete Referenzmerkmale nahe genug an der Epipolarlinie liegen und somit nicht als falsch erkannt werden. Eine falsche Zuordnung tritt aber für gewöhnlich in mehreren Bildern mit unterschiedlicher Lage auf, so dass sich deutlich unterschiedliche Epipolarlinien ergeben, und die falsche Zuordnung mit hoher Wahrscheinlichkeit erkannt wird. Zusätzlich wird die Zuverlässigkeit erhöht, indem der Test ebenfalls in der anderen Richtung vom Zielbild zum aktuellen Kamerabild durchgeführt wird.

Die in der Datenbank verbliebenen Merkmale bilden nun die Sollmerkmale für die Regelung. Abbildung 4.1 zeigt das Ergebnis der automatischen Merkmalsauswahl an einem Beispiel. Die blauen Plus-Zeichen zeigen alle 188 Merkmale die der Sift-Algorithmus in diesem Bild ermittelt hat. Der Unterschied zu den oben angegeben 2000 Merkmalen ergibt sich dabei aus der fehlenden Verdoppelung der Auflösung im SIFT-Algorihtmus und der geringeren Bildgröße. Nach dem ersten Schritt bleiben die 82 mit einem grünen Kreuz markierten Merkmale übrig. Am Ende sind es noch 61 Merkmale (mit einem roten Kreis markiert), die für die Regelung genutzt werden.

Die im Folgenden vorgestellten Regler besitzen keine Obergrenze für die Anzahl der verwendbaren Merkmale und in den durchgeführten Experimenten blieb nach der obigen

Elimination eine vernünftige Anzahl von Merkmalen übrig. Trotzdem kann es sinnvoll sein, die Anzahl der Merkmale zu reduzieren, um zum Beispiel Rechenzeit zu sparen. Dazu stehen drei Kriterien zur Verfügung, die je nach Anwendung gegeneinander abgewogen werden müssen. Ein Kriterium ist die Fläche, die die Merkmale aufspannen. Diese muss für eine erfolgreiche Regelung möglichst groß sein. Somit werden besser Merkmale entfernt, die im Inneren der aufgespannten Fläche liegen. Das zweite Kriterium betrifft die Skalierung der Merkmale. Da in der Praxis der Roboterarm meist auf das Objekt zu bewegt wird, bekommen die Merkmale mit einer großen Skalierung den Vorzug. Diese sind auch noch in einem größeren Abstand vom Objekt zu erkennen. Als drittes Kriterium wird die Anzahl der Bilder genommen, in denen ein Merkmal erkannt wurde.

Andersherum bedeutet eine zu kleine Anzahl von Merkmalen, die am Ende übrig bleiben, dass entweder die Parameter zu restriktiv für die Güte der Berechnung und Kalibrierung eingestellt sind, oder dass der SIFT-Algorithmus nicht für die verwendete Textur geeignet ist. Aber auch wenn die Anzahl der Merkmale in der Datenbank ausreichend groß ist, dann gibt die Anzahl der gefundenen Merkmale in den einzelnen Bildern Aufschluss darüber, ob auch am Rand des Arbeitsraums genügend Merkmale für die Regelung zur Verfügung stehen. Wenn dies nicht der Fall ist muss der Arbeitsraum verkleinert und somit mehr Zwischenbilder aufgenommen werden (siehe Kapitel 8).

5 Adaptiver, Bildbasierter Regler

Nachdem das Problem der Merkmalsextraktion aus natürlichen Texturen durch den SIFT-Algorithmus und die automatische Merkmalsauswahl gelöst wurde, wird im folgenden ein Regler entwickelt, der unter Verwendung der SIFT-Merkmale die weiteren Nebenbedingungen erfüllt.

Diese Nebenbedingungen sind zum ersten, das der Regler robust gegen den zeitweisen Verlust von Referenzmerkmalen im aktuellen Kamerbild sein muss. Dies kann bei der Verwendung von SIFT-Merkmalen trotz sorgfältiger Auswahl der Merkmalen nicht verhindert werden. Darüber hinaus kommt es gelegentlich vor, dass ein Merkmal falsch zugeordnet wird. Dies darf die Robustheit des Reglers nicht signifikant beeinflussen. Auf Grund der Merkmalsauswahl tritt dies aber nur sehr selten auf, so dass dies eine untergeordnete Rolle beim Reglerdesign spielt.

Zum zweiten haben viele der verwendeten Objekte die Eigenschaft, dass sie annähernd quaderförmig sind. Dies hat zur Folge, dass die physikalischen Merkmale, die zu den verwendeten Bildmerkmalen gehören, häufig in einer Ebene liegen, weil nur Merkmale von einer Seite des Quaders gefunden werden. Zusammen mit der Tatsache, dass hier eine Auge-In-Hand Konfiguration verwendet wird, erschwert dies die Unterscheidung zwischen bestimmten Rotationen und Translationen deutlich, was in Abschnitt 5.4 genau erläutert wird.

Das zunächst schwerwiegendste Problem ist allerdings, dass über das Zielobjekt, relativ zu dem der Greifer positioniert werden soll, kein zusätzliches Modellwissen vorhanden ist. Dies bedeutet, da die geometrische Form des Objektes nicht bekannt ist, kann aus den Merkmalen nicht die relative Lage der Kamera zum Objekt berechnet werden. Somit müsste für die Anwendbarkeit eines positionsbasierten Reglers die geometrische Form des Objektes in der Lernphase geschätzt werden. Ein darauf aufbauender Regler ist im allgemeinen allerdings weniger robust als ein bildbasierter Regler. Deshalb wird dieser Ansatz in dieser Arbeit nicht verfolgt.

Für den bildbasierten Regler wird die Look-Then-Move Struktur verwendet. Dadurch werden zum einen Probleme mit der Totzeit durch die Bildverarbeitung vermieden. Zum anderen vereinfacht dies die Synchronisation zwischen der Bestimmung der TCP-Lage im Referenzkoordinatensystem und dem Aufnahmezeitpunkt der Kamera beziehungsweise der „Virtual-Reality-Toolbox“ von Matlab.

Das in Abschnitt 2.7 vorgestellte Reglerdesign wird im folgenden als Ausgangpunkt für die Entwicklung des Reglers gewählt. Allerdings kann dieser Regler nicht identisch übernommen werden, weil zur Berechnung der Jacobi-Matrix, wie in Formel 2.2 zu sehen, der Tiefenabstand der Merkmale von der Kamera benötigt wird. Auf Grund des fehlenden Modells des Objektes und der unbekannten relativen Lage des Objektes zur Roboterbasis, kann diese Information aber nicht bestimmt werden.

1. Initialisiere Jacobi−Matrix
2. Führe einen Schritt $\Delta\mathbf{X}$ mit der bisherigen Jacobi−Matrix aus
3. Beobachte die Merkmalsbewegung $\Delta\mathbf{m}$ im Bild
4. Adaptiere die Jacobi−Matrix nach der Vorschrift:
   $\mathbf{J}_{k+1} = \mathbf{J}_k + a \cdot \frac{(\Delta\mathbf{m} - \mathbf{J}_k\Delta\mathbf{X})\Delta\mathbf{X}^T}{\Delta\mathbf{X}^T\Delta\mathbf{X}}$
5. Beginne wieder mit Schritt 2 bis die Merkmalspositionen konvergiert sind

Pseudocode 5.1: Adaption der Jacobi-Matrix

In diesem Kapitel wird deshalb ein Verfahren von Jägersand [Jäg96] vorgestellt, in dem die Jacobi-Matrix während der Bewegung adaptiert und somit Fehler korrigiert werden. Dazu wird zunächst im folgenden Abschnitt die Adaption an sich erläutert. In Abschnitt 5.2 wird die Trust-Region-Methode erläutert, die die Konvergenz des Reglers verbessert. Danach werden in 5.3 einige Probleme mit der Kinematik des Roboterarms und Schwierigkeiten in der Anwendung des bildbasierten Reglers aufgezeigt, bevor in Abschnitt 5.4 auf die Problematik mit lokalen Minima eingegangen wird.

5.1 Adaptive Jacobi-Matrix

Das folgende vorgestellte Reglerdesign unterscheidet sich von dem im Abschnitt 2.7 nur in der Bestimmung der Jacobi-Matrix. Während im obigen Fall in jedem Arbeitspunkt die Jacobi-Matrix neu analytisch berechnet wird, wird hier die Jacobi-Matrix aus dem vorherigen Schritt, auf Grundlage der Bewegung des Manipulators und der beobachteten Bewegung der Merkmale im Bild, nur korrigiert. Dies geschieht also während der Erfüllung der Regelaufgabe mit den Informationen, die sowieso zur Verfügung stehen.

Pseudocode 5.1 zeigt den Ablauf des Algorithmus. Vor dem ersten Regelschritt muss die Jacobi-Matrix initialisiert werden. Dies geschieht entweder rein zufällig oder mit einer sinnvollen Schätzung, wenn entsprechende Zusatzinformationen bekannt sind. Je mehr sich die initiale Jacobi-Matrix von der tatsächlichen in diesem Arbeitspunkt unterscheidet, um so mehr Lernschritt werden benötigt, um diesen Fehler zu korrigieren, und um so schlechter sind die ersten Bewegungen des Roboterarms.

Als nächstes wird mit der aktuellen Jacobi-Matrix $\mathbf{J}_k$ nach dem Reglergesetz 2.1 eine Kamerabewegung $\Delta\mathbf{X}$ bestimmt. Bei Ausführung dieser Bewegung haben sich die Merkmale im Bild um $\Delta\mathbf{m}$ bewegt. Ziel ist es nun, dass Modell der Jacobi-Matrix $\mathbf{J}$ in Einklang mit der Messung $\Delta\mathbf{m}$ zu bringen, es soll also gelten $\Delta\mathbf{m} = \mathbf{J}_{k+1} \cdot \Delta\mathbf{X}$. Für $dim(\mathbf{J}) > [1, 1]$ hat dieses Problem allerdings keine eindeutige Lösung. Hier wird die Lösung gewählt, bei der die geringste Änderung an der Jacobi-Matrix nötig ist.

Dazu wird aus der Abweichung der vorhergesagten Bewegung der Bildmerkmale von der tatsächlichen Bewegung $\Delta\mathbf{m} - \mathbf{J}_k \cdot \Delta\mathbf{X}$ ein Korrekturterm für die Jacobi-Matrix berechnet:

$$\Delta\mathbf{J}_k = \frac{(\Delta\mathbf{m} - \mathbf{J}_k\Delta\mathbf{X})\Delta\mathbf{X}^T}{\Delta\mathbf{X}^T\Delta\mathbf{X}} \tag{5.1}$$

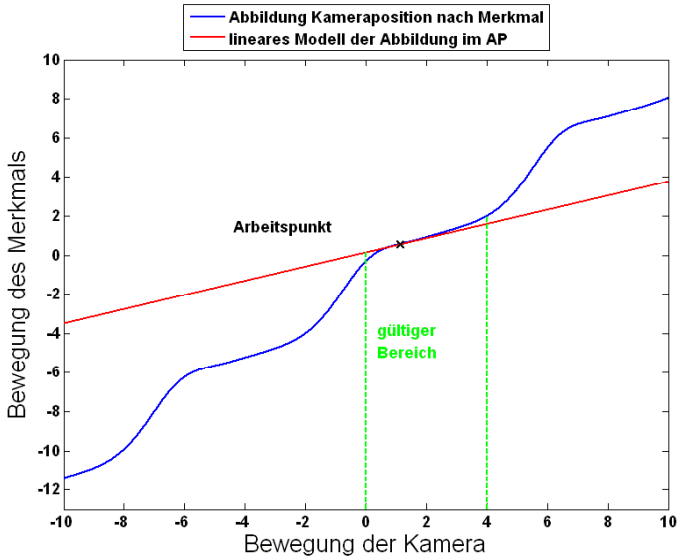


Abb. 5.1: Linearisierung im Arbeitspunkt

Es ist einfach zu sehen, dass $\mathbf{J}_{k+1} = \mathbf{J}_k + \Delta\mathbf{J}_k$ das Modell mit der Messung in Einklang bringt. In der Praxis wird allerdings ein Lernfaktor $a \in [0, 1]$ gewählt, um Störungen zu unterdrücken. Das neue Modell ergibt sich also zu $\mathbf{J}_{k+1} = \mathbf{J}_k + a \cdot \Delta\mathbf{J}_k$, wobei $a = min(1, \frac{\|\Delta\mathbf{m}\|}{Ndi})$ gilt. Ndi repräsentiert das aktuelle Vertrauen in das Modell angegeben durch eine maximal erlaubte Schrittlänge. Die hier beschriebene Technik fällt in eine Klasse genannt „Broyden Methoden“ ([Fle87]), die in der nichtlinearen Optimierung eingesetzt werden.

Ein weiterer Vorteil dieses Ansatzes ist, dass hier keine Modellierung des Systems vorgenommen werden muss. Für eine andere Anzahl oder Typ von Freiheitsgraden, für andere Merkmale oder für eine direkte Steuerung der Achsgelenke des Roboterarms muss also nur die Größe der Jacobi-Matrix entsprechend angepasst werden.

5.2 Trust-Region-Methode

Bei dem vorgestellten Reglergesetz wird ein lineares Modell der Abbildung von der Bewegung der Kamera auf die Bewegung der Merkmale benutzt, was Abbildung 5.1 beispielhaft an einem fiktiven Zusammenhang zwischen einer eindimensionalen Bewegung der Kamera und einem einzelnen Merkmal darstellt. Dieses lineare Modell besitzt aber nur in einem gewissen Bereich um den Arbeitspunkt herum eine hinreichende Gültigkeit. Das bedeutet, dass in diesem Bereich das lineare Modell nur wenig vom tatsächlichen funktionalen Zusammenhang abweicht. Die Größe dieses Bereiches ist dabei abhängig vom aktuellen Arbeitspunkt.

Vor allem bei einem großen Abstand zum Ziel ist dieses Problem von Bedeutung, da hier der Bildfehler relativ groß ist, und somit große Stellgrößen berechnet werden. Abweichungen des Modells von der Wirklichkeit wirken sich also stärker aus. Im obigen Reglergesetz wird dieses Problem mit einem konstanten Reduktionsfaktor $K \in [0, 1]$ berücksichtigt,

der die Schrittweite so reduziert, dass der Regler auch im Worst-Case im gültigen Bereich bleibt und somit konvergiert.

Bei der Verwendung der adaptiven Jacobi-Matrix wird dieses Problem noch verstärkt. Insbesondere am Anfang der Bewegung, wenn die Jacobi-Matrix zufällig initialisiert wurde, stimmt die Schätzung nicht mit dem tatsächlichen Modell überein. Aber auch während der Bewegung wird die Jacobi-Matrix in jedem Schritt für den vorherigen Arbeitspunkt gelernt. Das adaptierte Modell hängt also dem tatsächlichen Modell immer ein wenig hinterher.

Um also die Konvergenz in den Zielpunkt in einem ausreichend großen* Bereich um den Zielpunkt herum gewährleisten zu können, muss K relativ klein gewählt werden, was zu sehr kleinen Bewegungen der Kamera in der Nähe der Ziellage führt. Dadurch wird die Regelzeit sehr lang. Dabei können in den meisten Arbeitspunkten größere Schritte ausgeführt werden ohne den gültigen Bereich des linearen Modells zu verlassen.

Um dieses Problem zu beheben, haben Siebel et al. [SLWG99] eine Trust-Region-Methode vorgestellt. Dabei wird die Konstante K durch einen variablen Reduktionsfaktor λ_k ersetzt. Ziel ist es, λ_k in jedem Schritt so zu bestimmen, dass zum einen möglichst große Stellgrößen berechnet werden, um eine schnelle Regelung zu erzielen. Zum anderen wird die Bewegung der Kamera so beschränkt, dass sie nicht aus einer Region heraus führt, in der man dem linearen Modell $\mathbf{J}_k$ vertrauen kann. Diese Region wird „trust region“ oder „model trust region“ genannt [Fle87].

Zur Bestimmung des Reduktionsfaktors λ_k wird eine Schranke α_k für die prognostizierte Bewegung der M Merkmale im Bild definiert. Für gegebenes α_k wird bei der Stellgrößenbestimmung zunächst die Länge l_k der maximalen Bewegung auf dem Sensor bei Ausführung der Kamerabewegung vorhergesagt:

$$l_k = \max_{i=1,\ldots,M} \left\| \begin{bmatrix} (\mathbf{J}_k \cdot \Delta\mathbf{X}_k)_{2i-1} \\ (\mathbf{J}_k \cdot \Delta\mathbf{X}_k)_{2i} \end{bmatrix} \right\|_2$$

Daraufhin wird im eigentlichen Stellgesetz die Stellgröße derart beschränkt, dass die vorhergesagte Bewegung keiner der abgebildeten Objektmarkierungen auf dem Sensor α_k überschreitet:

$$\begin{aligned} \Delta\mathbf{X}_k &= \lambda_k \cdot \mathbf{J}_k^{\dagger} \cdot (-\Delta\mathbf{m}_k) \\ &= min\left\{1, \frac{\alpha_k}{l_k}\right\} \cdot \mathbf{J}_k^{\dagger} \cdot (-\Delta\mathbf{m}_k) \end{aligned}$$

Es gilt nun, α_k in jedem Schritt so einzustellen, dass eine möglichst große Stellgröße zugelassen, andererseits aber der Bereich, in dem das Modell hinreichend gültig ist, nicht verlassen wird.

Dazu wird ein Maß für die Abweichung des Modells $\mathbf{J}_k$ vom tatsächlichen Verhalten der Regelstrecke bestimmt. Zu den Bildmerkmalen $\mathbf{m}_k$ im k-ten Abtastschritt, $k > 0$, ergeben sich die entsprechenden vorhergesagten Bildmerkmale durch

$$\tilde{\mathbf{m}}_k = \mathbf{m}_{k-1} + \mathbf{J}_{k-1} \cdot \Delta\mathbf{X}_{k-1}$$

*„Ausreichend groß“ ist hier abhängig von der Entfernung der Lagen zueinander, in denen die Zwischenbilder aufgenommen werden (siehe Kapitel 8).

Damit wird im Abtastschritt $k > 0$ der bei Ausführung von $\Delta\mathbf{X}_{k-1}$ aufgetretene momentane Modellfehler d_k für M beobachtete Merkmale als das Maximum der Abweichung zwischen vorhergesagten und tatsächlichen Merkmalspositionen im Bild definiert:

$$d_k = \max_{i=1,\ldots,M} \left\| \begin{bmatrix} (\tilde{\mathbf{m}}_k)_{2i-1} \\ (\tilde{\mathbf{m}}_k)_{2i} \end{bmatrix} - \begin{bmatrix} (\mathbf{m}_k)_{2i-1} \\ (\mathbf{m}_k)_{2i} \end{bmatrix} \right\|_2$$

Zur Bewertung des Modellfehlers wird zusätzlich zum momentanen Modellfehler d_k ein zulässiger (akzeptierter) Modellfehler d_{soll} als Reglerparameter vorgegeben. Daraus ergibt sich der relative Modellfehler

$$r_k = \frac{d_k}{d_{soll}}$$

Ein kleiner Wert stellt dabei eine gut Übereinstimmung zwischen Modell und Realität dar. Angestrebt zur Abwägung von Modellfehler und einer raschen Regelung ist der Wert $r_k = 1$. Das Ziel ist es also, α_k in jedem Schritt so zu wählen, dass im nächsten Schritt d_{k+1} den Wert d_{soll} nach Möglichkeit genau erreicht. Es gilt:

$$\alpha'_k = \alpha_{k-1} \cdot \frac{d_{soll}}{d_k} = \frac{\alpha_{k-1}}{r_k}$$

Der Wert von α_k ergibt sich aus α'_k unter Beachtung weiterer Reglerparameter:

- Minimal- und Maximalwert α_{min} und α_{max} für α_k. Mit Hilfe dieser Schranken soll sichergestellt werden, dass die zulässige Schrittweite nicht zu große oder zu kleine Werte annimmt.
- Ein Faktor μ, der den Anstieg von α_k in einem Regelschritt beschränkt, um die Empfindlichkeit gegen Rauschen in den Messwerten und Nichtlinearitäten in der Regelstrecke zu verringern.

Für $d_k = 0$ wird α_k unter Beachtung der obigen Reglerparameter maximal erhöht.

Abbildung 5.2 illustriert das Verfahren und die verwendeten Größen anhand von einem einzelnen Merkmalspunkt. In diesem Beispiel ist die tatsächliche Abweichung kleiner als die zulässige Abweichung. Für die Berechnung der nächsten Stellgröße wird also α_{k+1} größer als α_k gewählt werden.

5.3 Anwendung

In diesem Abschnitt wird auf einige grundsätzliche Probleme und Fragestellungen eingegangen, die sich bei der Umsetzung des obigen Reglers ergeben. Auf die Darstellung der konkreten Implementierung wird hierbei verzichtet.

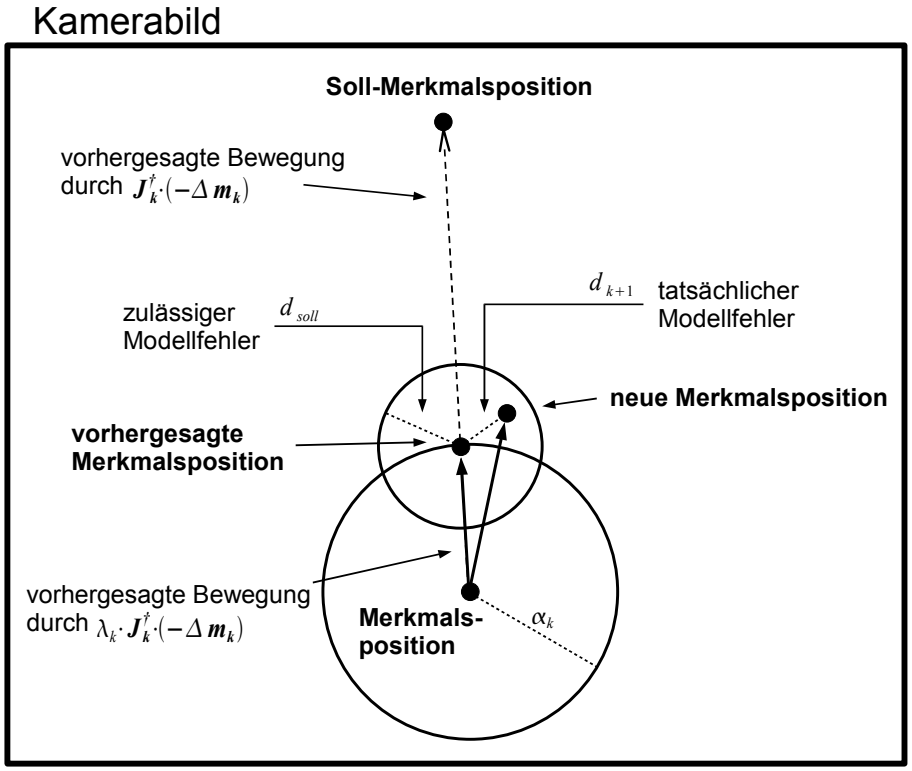


Abb. 5.2: Trust-Region-Methode am Beispiel eines einzelnen Merkmalpunktes (Bild aus [SLWG99] mit kleinen Modifikationen)

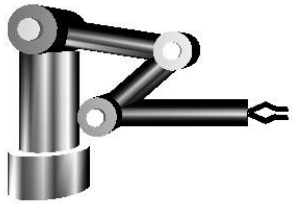
5.3.1 Darstellung der Lage des Roboterarms

Besitz ein Roboterarm alle sechs Freiheitsgrade, dann kann die Lage des TCPs im Referenzkoordinatensystem des Roboters durch sechs Parameter angegeben werden. Dazu wird oft die Position des TCPs in euklidischen Koordinaten $[x, y, z]$ und die Orientierung zum Beispiel in Eulerwinkeln oder Roll-Pitch-Yaw Winkeln angegeben [Cra04].

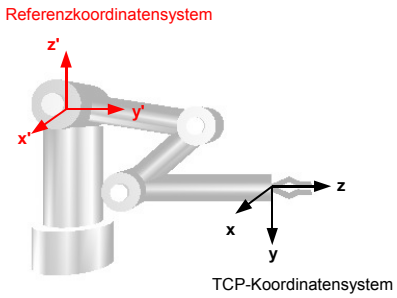
Der hier verwendete Roboterarm besitzt allerdings nur fünf Achsen und somit nur fünf Freiheitsgrade. Gesucht ist also eine Darstellung der möglichen Lagen mit fünf frei wählbaren Parametern. Auf Grund der Geometrie des Roboterarms und der Wahl der Rotationsachsen bei Eulerwinkeln und Roll-Pitch-Yaw Winkeln sind diese dafür nicht geeignet. Statt dessen werden die Rotationsachsen des Roboters verwendet.

Zur Herleitung müssen die Bewegungsmöglichkeiten des Roboterarms genau betrachtet werden. Zunächst gibt es keine Einschränkung der möglichen Positionen des TCPs. Daher können weiterhin die drei euklidischen Koordinaten für die Beschreibung der Position verwendet werden. Bei der Betrachtung von Rotationen spielt im Allgemeinen die Reihenfolge eine Rolle für das Ergebnis. Da hier die Rotationachsen entsprechend dem Roboterarm gewählt werden, gilt dies nicht, weil sie entsprechend mitrotieren. Es wird deshalb die Reihenfolge gewählt, die das entscheidende Problem am besten verdeutlicht.

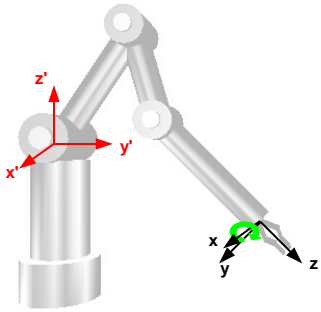
In Abbildung 5.3 (a) ist der TCP in der Position $[0, 290, -120]$ im Referenzkoordinatensystem zu sehen, wobei die Koordinaten in mm angegeben sind. Die Orientierung des



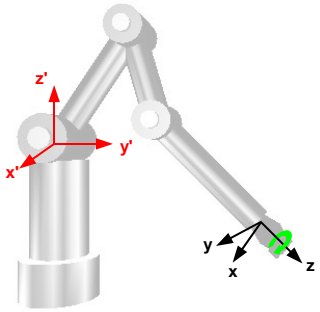
(a) Roboterarm



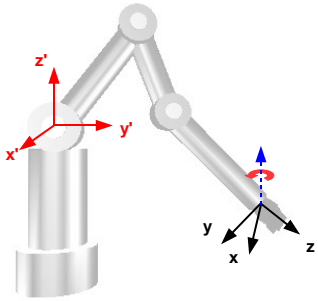
(b) Koordinatensysteme



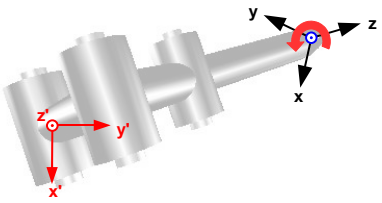
(c) α-Rotation



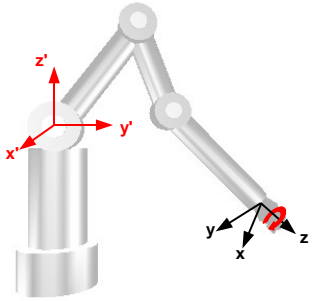
(d) γ-Rotation



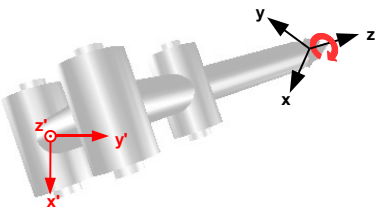
(e) Abhängige β-Roation



(f) Abhängige β-Roation



(g) Korrektur durch γ-Rotation



(h) Korrektur durch γ-Rotation

Abb. 5.3: Herleitung der Darstellung der Lage des Roboterarms

TCP-Koordinatensystems ist dabei identisch zum Koordinatensystem der Kamera gewählt worden (b), die mittig auf dem letzten Gelenkkörper angebracht wird (siehe zum Beispiel Abbildung 5.5 (a)). Somit unterscheiden sich die beiden Koordinatensysteme nur durch eine Verschiebung, was die Umrechnung zwischen den Koordinatensystemen erleichtert. In dieser Lage seien die gesuchten Parameter der Rotation gleich Null.

Die erste Achse um die der TCP gedreht wird, ist die X-Achse des TCP-Koordinatensystem beziehungsweise des Referenzkoordinatensystems (c). Dieser Parameter wird mit α bezeichnet. Die zweite Rotationsachse ist die neue Z-Achse des TCP-Koordinatensystems, dessen Rotation durch den Parameter γ ausgedrückt wird (d).

Ein Problem ergibt sich, wenn die Position des TCPs nicht in der Y-Z-Ebene des Referenzkoordinatensystems liegt. Um eine solche Position zu erreichen, muss der Roboterarm das erste Achsgelenk drehen, wodurch es unmittelbar zu einer Rotation des TCP-Koordinatensystems um die erste Y-Achse des TCP-Koordinatensystems kommt (von vorne (e), von oben (f)). Der Winkel β ergibt sich dabei aus der Position durch $\beta = atan2(x, y)$.

Die frei wählbaren Parameter $[x, y, z, \alpha, \gamma]$ und der abhängige Parameter β beschreiben also eine eindeutige, erreichbare Lage des TCPs und jede erreichbare Lage kann mit diesen fünf Parametern dargestellt werden. Der Nachteil dieser Wahl der Parameter ist aber, dass sie die translatorische Bewegung nicht vollständig von der Rotation entkoppelt. Befindet sich zum Beispiel der TCP in einer Position in der Y-Z-Ebene und es gilt $\alpha = 90°$, also dass der Greifer senkrecht auf den Tisch zeigt, dann führt eine Bewegung in X-Richtung zu einer Veränderung der Orientierung des TCPs. Dabei ist in diesem Fall die Position auch mit der alten Orientierung für den Roboterarm zu erreichen, indem die Orientierung durch eine Rotation des letzten Achsgelenkes ausgeglichen wird.

Allgemein muss die zwangsläufige Rotation um die β-Achse*, durch die Rotationen um die anderen Achsen ausgeglichen werden. Da die α-Achse immer senkrecht auf der β-Achse steht, kann diese dazu nicht benutzt werden. Wenn $\alpha \notin \{0°, 180°\}$ dann gilt dies allerdings nicht für die γ-Achse. Für $\alpha \in \{90°, 270°\}$ kann die Rotation sogar komplett kompensiert werden, da die γ- und die β-Achse in diesen Fällen parallel sind. Allgemein muss das TCP-Koordinatensystem um die γ-Achse um den Wert

$$\gamma - \beta \cdot sin(\alpha)$$

rotiert werden, um die β-Rotation so gut wie möglich auszugleichen (von vorne (g), von oben (h)).

5.3.2 Bewegungen der Kamera

Bei der Implementierung des Reglers muss man beachten, dass der Ausgabevektor die Bewegungen der Kamera $\Delta\mathbf{X} = [\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\gamma]$ im Kamerakoordinatensystem beschreibt. Die Beschränkung des Roboterarms müssen also auf dieses übertragen werden.

Zunächst werde angenommen, dass sich die Kamera genau im TCP befindet, dass also die beiden Koordinatensystem identisch sind†. Die γ-Achse ist dabei mit der optischen

*Zum besseren Verständnis werden die Rotationsachsen durch die mit ihnen assoziierten Parameter bezeichnet.

†Dies macht praktisch keinen Sinn, da die Kamera so dem Greifer und dem Objekt im Wege ist, stellt aber einen guten Ausgangspunkt für die nachfolgenden Überlegungen dar.

Achse identisch. Die α-Achse liegt in der X-Y-Ebene des Kamerakoordinatensystems und die Richtung hängt von der Drehung des letzten Achsgelenkes ab.

Um das Reglerdesign zu vereinfachen und die allgemeine Verwendbarkeit des Reglers zu gewährleisten, insbesondere für Roboterarme mit sechs Freiheitsgraden, wird die α-Rotation aus der Rotation um die statischen X- und Y-Achse berechnet. Der Regler ermittel also zunächst die Bewegungen für alle sechs Freiheitsgrade, darunter die Rotationen $\Delta\alpha'$, $\Delta\beta'$ und $\Delta\gamma$ um die X-, Y- und Z-Achse des Kamerakoordinatensystems. Befindet sich der TCP beziehungsweise die Kamera in der Lage $[x, y, z, \alpha, \gamma]$, dann ergibt sich die Rotation um die α-Achse $\Delta\alpha$ zu:

$$\begin{aligned} \theta &= \gamma - atan2(x, y) \cdot sin(\alpha) \\ \Delta\alpha &= cos(\theta) \cdot \Delta\alpha' + sin(\theta) \cdot \Delta\beta' \end{aligned}$$

Aus dieser Bewegung und der alten Lage der Kamera lässt sich mit Hilfe von homogenen Koordinatentransformationen die neue Lage der Kamera berechnen. Da diese identisch ist mit der Lage des TCPs, ist dies auch dessen neue Lage, die durch die Konstruktion auch erreichbar ist. An dieser Stelle sei darauf hingewiesen, dass die umgekehrte Rechnung vor jedem Adaptionsschritt gemacht werden muss, da zur Berechnung der Korrekturmatrix 5.1 die Bewegung der Kamera im alten Kamerakoordinatensystem benötigt wird. Diese muss aus der alten und der neuen Lage der Kamera berechnet werden, weil sich die tatsächliche Bewegung auf Grund von Ungenauigkeiten in der Positionierung des Roboterarms von der Ausgabe des Reglers im vorherigen Schritt unterscheiden kann.

Befindet sich die Kamera allerdings nicht genau im TCP, dann muss aus der neuen Lage der Kamera, die neue Lage des TCPs berechnet werden. Ist die relative Lage von TCP und Kamera bekannt, ist dies mit homogenen Koordinatentransformationen zwar möglich, die neue Lage des TCPs muss aber für den Roboterarm nicht erreichbar sein.

Gelöst werden kann dieses Problem, indem der Regler so beschränkt wird, dass er nur Bewegungen ausführt, die zu Kameralagen führen, zu denen erreichbare TCP-Lagen gehören. Dies gefährdet allerdings die Güte und die allgemeine Einsetzbarkeit des Reglers in anderen Systemen.

Die zweite Lösungsmöglichkeit ist, wie oben beschrieben, eine neue Lage der Kamera zu berechnen. Dazu wird dann die Lage des TCPs ermittelt, die die Kamera möglichst nah an die vorgesehene Lage heranbringt. Dies ist allerdings ein nicht triviales multikriterielles Optimierungsproblem, dass abhängig vom Verhalten des Reglers Abweichungen in bestimmten Richtungen den Vorzug geben muss. Die allgemeine Lösung dieses Problems soll hier nicht weiter erörtert werden.

Stattdessen wird für den oben vorgestellten adaptiven Regler eine Methodik verwendet, die sich für den Fall, dass α in der Nähe von 90° ist, als sehr gut herausgestellt hat. Die Beschränkung fällt hier nicht so sehr ins Gewicht, so dass dies für die Tests des adaptiven Reglers ausreicht. Bei dieser Methodik wird die Position des TCP entsprechender der neuen Lage der Kamera und der relativen Lage des TCPs und der Kamera berechnet und die Rotationen α und γ einfach übernommen. Da die Position des TCPs allerdings anders ist als die der Kamera und somit die abhängige Rotation β sich verändert hat, unterscheidet sich die tatsächliche Lage der Kamera von der Gewünschten, wie in Abbildung 5.4 an einer Beispiellage zu sehen ist. Bei der obigen Einschränkung ist der Fehler aber zu vernachlässigen.

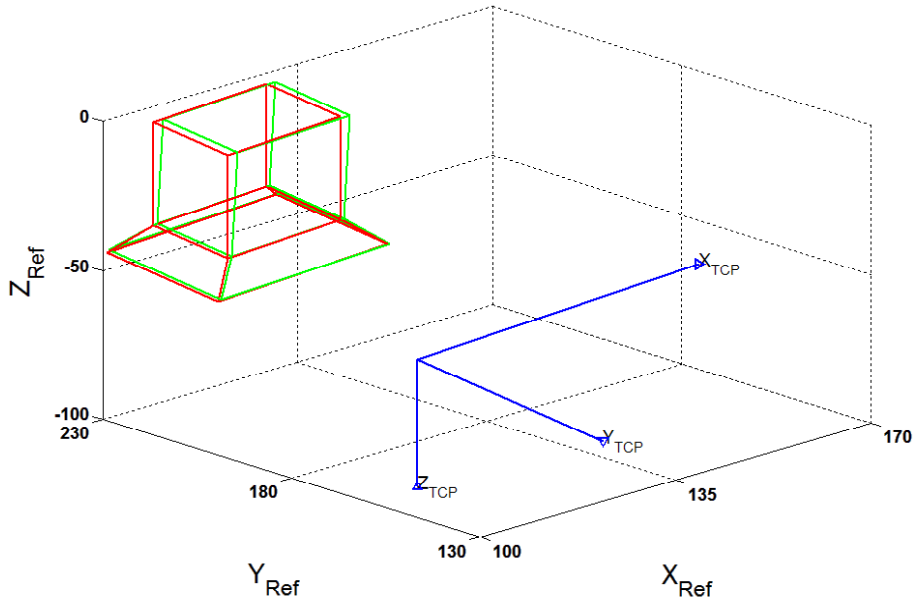


Abb. 5.4: Unterschied zwischen der tatsächlichen (rot) und gewünschten (grün) Lage der Kamera

Bei dem in Kapitel 7 entwickelten alternativen Regler kann dies jedoch nicht eingesetzt werden, da dieser in jeder Situation funktionieren soll. In diesem Fall wird vereinfachend angenommen, dass sich die Kamera im TCP befindet. Dies führt dazu, dass bei einer gewünschten Rotation der Kamera diese tatsächlich zusätzlich eine Bewegung auf einer Kreisbahn um den TCP ausübt. Diese zusätzliche Bewegung muss dann vom Regler im nächsten Schritt ausgeglichen werden. Dabei zeigt sich der vorgestellte Regler als äußerst robust gegen diesen „Kalibrierungsfehler“, so dass dies besser ist als eine schlechte Wahl der Lage des TCPs in der vorherigen Lösung. Allerdings sollte die Kamera nicht zu weit vom TCP entfernt sein, was bei einer Montierung auf dem Roboterarm zwangsläufig vermieden wird. Auf der anderen Seite hat dieses Verfahren den Vorteil, das auf die Angabe der relativen Lage der Kamera zum TCP verzichtet werden kann.

5.3.3 Verlust von Merkmalen

Wie in Kapitel 3 beschrieben, wird für die Merkmalsextraktion der SIFT-Algorithmus benutzt. Da keine künstlichen Markierungen auf dem zu fassenden Objekt genutzt werden, besteht allerdings eine relativ hohe Wahrscheinlichkeit, dass in einem Schritt ein Teil der Merkmale, die für die Regelung verwendet werden, von dem Algorithmus nicht erkannt wird. Dies kann daran liegen, dass der entsprechende Teil des Objektes zeitweise nicht zu sehen ist, dass auf Grund der Bewegung diese Punkte nicht mehr als Merkmalspunkte erkannt werden, oder dass die Merkmalspunkte den entsprechenden Sollpunkten nicht zugeordnet werden konnten. Der Regler darf selbst dann nicht versagen, wenn ihm nur ein Teil der Merkmale zur Verfügung steht.

Bei dem hier verwendeten adaptiven Regler kann dieses Problem mit geringen Änderungen gelöst werden. Zu jedem Merkmalspunkt mit seiner u- und v- Komponente gehören genau zwei Zeilen der Jacobi-Matrix. Diese zwei Zeilen können für jeden Merkmalspunkt getrennt adaptiert werden, da jede Zeile des Korrekturterms 5.1 nur von der Bewegung des zugehörigen Merkmals, der Bewegung der Kamera und der entsprechenden Zeile in der aktuellen Jakobi-Matrix abhängt. In einem Regelschritt werden also nur die Zeilen der Jacobimatrix adaptiert und im Reglergesetz 2.1 verwendet, die zu Merkmalen gehören, die im aktuellen Bild erkannt wurden. Ähnliches gilt für die Bestimmung des Reduktionsfaktors bei der Trust-Region-Methode. Hier werden nur die Merkmale verwendet, die im aktuellen und im vorherigen Schritt erfasst wurden.

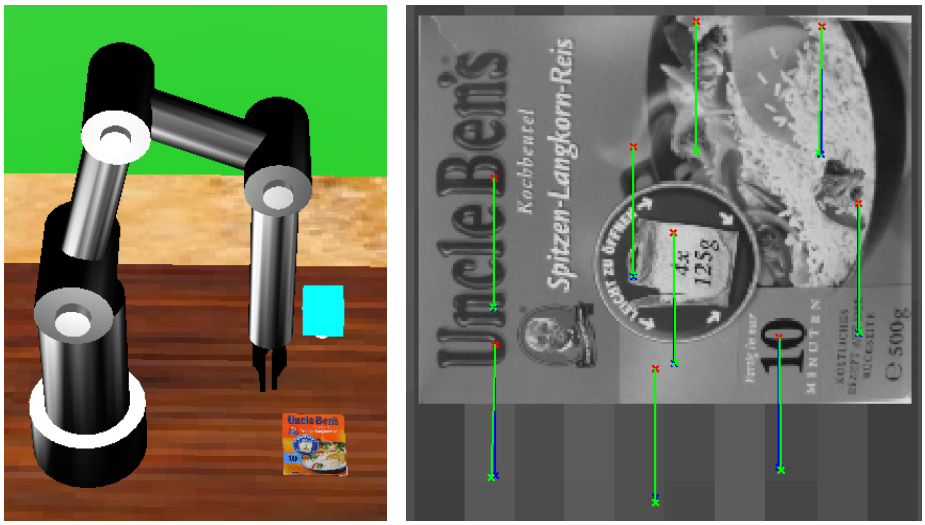
Nachteil dieses Verfahrens ist ein relativ große Aufwand zur Datenverwaltung. Für jeden Merkmalspunkt muss einzeln die Position der Kamera, in der der Merkmalspunkt zuletzt gefunden wurde und ob der Merkmalspunkt im letzten Bild zu sehen war, abgespeichert werden. Außerdem führt ein längerer Verlust eines Merkmalspunktes dazu, dass die entsprechenden Zeilen der Jacobi-Matrix nicht mehr mit dem aktuellen linearen Modell übereinstimmen. Auf Grund der Merkmalsauswahl ist ein längerer Verlust eines Merkmals allerdings relativ selten, und bei einer ausreichend großen Anzahl von Merkmalen kompensieren die anderen Punkte den Fehler dieses Merkmals, bis die Jacobi-Matrix auch hier wieder richtig gelernt wurde. Analoges gilt für eine gelegentliche, falsche Zuordnung eines Merkmalspunktes und der daraus folgenden, falschen Adaption der entsprechenden Zeilen der Jacobi-Matrix.

5.3.4 Variable Adaptionsschritte

Bei der praktischen Anwendung der adaptiven Jacobi-Matrix stellt sich ein negative Effekt durch das Rauschen in der Position der Bildmerkmale ein. Führt der Roboterarm nur relativ kleine Bewegungen aus, dann wird der Nenner im Korrekturterm 5.1 sehr klein. Bei korrekten Merkmalspositionen wird dies durch entsprechend kleine Werte in der Matrix im Zähler und die kleine Lernrate ausgeglichen. Durch das Rauschen in der Position wird dieses Verhältnis aber gestört, was zu sehr großen Einträgen im Korrekturterm und somit in der Jacobi-Matrix führt, die bis zu fünfzehn Größenordnungen über den normalen Werten liegen.

Daraus ergeben sich wiederum kleine Einträge in der Pseudoinversen und somit kleine Stellgrößen des Reglers, was zu noch kleineren Bewegungen des Roboterarms führt, was das Problem immer weiter verstärkt. Führt der Roboterarm also in Abhängigkeit von der Stärke des Rauschen einmal eine zu geringe Bewegung aus, dann wächst die Jacobi-Matrix innerhalb weniger Regelschritte so stark an, dass der Roboter praktisch still steht.

Zur Lösung des Problems wird ausgenutzt, dass sich die Jacobi-Matrix bei geringen Bewegungen nur wenig verändert, also nicht unbedingt adaptiert werden muss. Die Jacobi-Matrix wird nicht mehr in jedem Regelschritt adaptiert, sondern nur dann, wenn sich der Roboterarm seit der letzten Adaption ausreichend weit bewegt hat, was von der Stärke des Rauschen abhängt. Dadurch hat der Nenner im Korrekturterm immer eine gewisse Mindestgröße und große Einträge in der Jacobi-Matrix werden vermieden. Außerdem wird so die Information aus der Bewegung des Roboterarms und der zugehörigen Bewegung der Merkmalspunkte besser ausgenutzt, da sie sich besser vom Rauschen abhebt.



(a) Ausgangslage (b) Bewegung der Merkmale

Abb. 5.5: Vergleich der Bewegungen in α-Richtung (grün) und Y-Richtung (blau)

In der Nähe vom Ziel wird durch diese Methodik die Jacobi-Matrix nicht mehr adaptiert, da die Mindestdistanz für eine Adaption nicht mehr erreicht wird. Wird diese hinreichend klein gewählt, dann ist die Jacobi-Matrix im Bereich des Ziels so gut gelernt, dass dieses erreicht wird.

5.4 α-Y-Problem

Ein großes Problem des adaptiven Regler ergibt sich aus der Auge-In-Hand Struktur. Zur Verdeutlichung befinde sich der Roboterarm in der in Abbildung 5.5 (a) gezeigten Stellung und die Kamera betrachtet ein auf dem Tisch liegendes Objekt (b). Führt der Roboterarm in dieser Lage zum einen eine Translation in Y-Richtung im Referenzkoordinatensystem aus oder zum anderen eine Rotation um die α-Achse, so ist festzustellen, dass die Merkmale in beide Fällen eine nahezu identische Bewegung im Bildraum ausführen. Das heißt, dass sich die beiden Bewegungen bezüglich der Abweichungen der Merkmale von der Sollposition fast genau kompensieren können, wenn sie entsprechend gegenläufig gerichtet sind. Die Kamera ist in diesem Fall dann deutlich von ihrer Zielposition entfernt, aber die Merkmale im Bild weisen fast keinen Fehler auf.

Erschwert wird dieses Problem dadurch, dass sich die Merkmale alle in einer Ebene befinden, also alle die gleiche Entfernung zum Augpunkt in Z-Richtung haben. Wie man an der analytischen Jacobimatrix 2.2 sehen kann, ist der Zusammenhang zwischen der Bewegung des Bildmerkmals und der Bewegung der Kamera in Y-Richtung abhängig von der Tiefe z im Gegensatz zur α-Richtung. Das bedeutet, dass die Größe der Bewegung in Y-Richtung, die eine bestimmte Rotation um die α-Achse ausgleicht, abhängig ist von z. Befinden sich also im Bild Merkmale mit deutlich unterschiedlicher Tiefe, dann erreichen diese nur in der Ziellage der Kamera alle gleichzeitig ihre Zielposition.

Die Auswirkungen dieses Problems werden an Hand eines Testlaufs verdeutlicht. Dabei gilt die in Abbildung 5.5 gezeigte Situation. Das betrachtet Objekt liegt flach auf dem

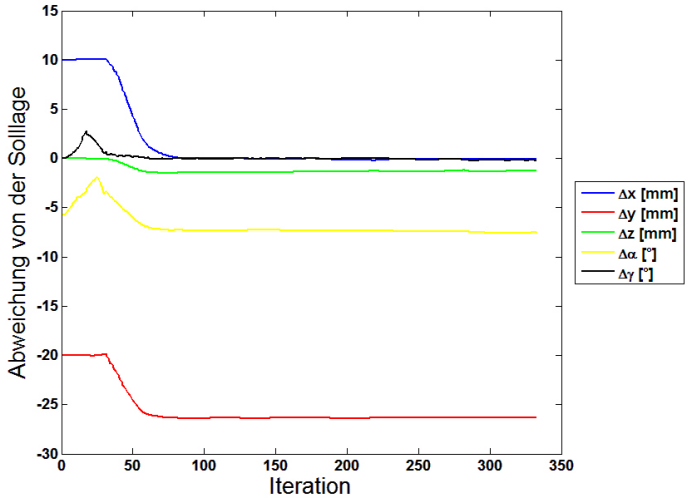


Abb. 5.6: Abweichung der Lage der Kamera von der Solllage in den fünf Dimensionen

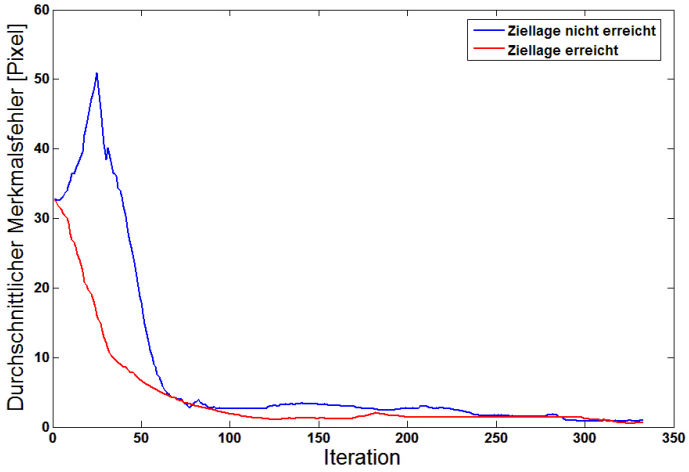
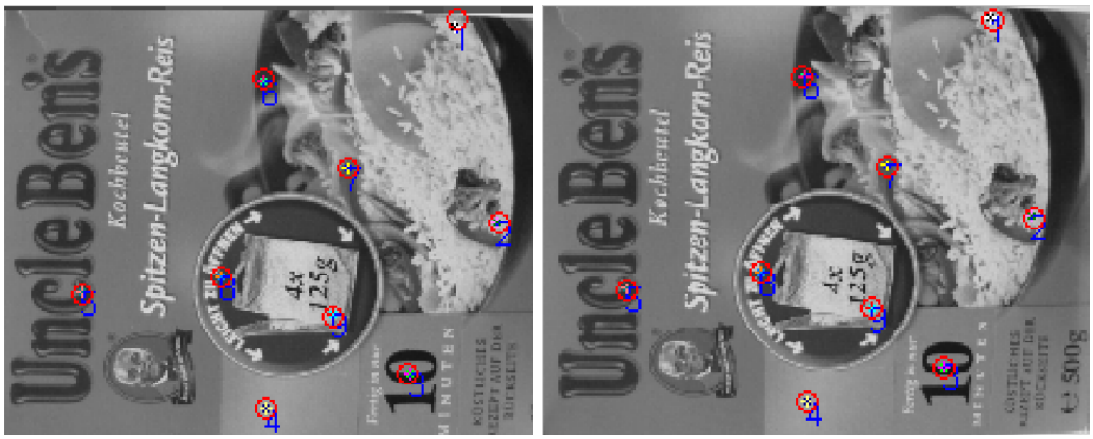


Abb. 5.7: Durchschnittlicher Abstand der Merkmale von den Sollpositionen für einen erfolgreichen und einen erfolglosen Regelverlauf

Tisch und die Solllage ist so gewählt, dass sich die Kamera genau oberhalb des Objektes befindet und die Bildebene parallel zum Tisch liegt. Von dieser Solllage wird die Kamera 1 cm in X-, 2 cm in Y-Richtung und ca. 6° um die α-Achse ausgelenkt und die Regelung gestartet. Abbildung 5.6 zeigt die Trajektorie der Kamera in den fünf Freiheitsgrade als Abweichung von den Sollwerten. Es ist zu erkennen, dass die Kamera eine stabile Lage erreicht, die mit $\Delta x = 1,3$ mm , $\Delta y = 26,3$ mm und $\Delta\alpha = 7,5°$ deutlich von der Ziellage entfernt ist. Die Lage bleibt für 250 Iterationen unverändert, der Regler ist also in einem lokalen Minimum konvergiert.

Dabei ergibt sich der Fehler in Z-Richtung durch die Fehler in Y- und α-Richtung, die aus dem oben beschriebenen Problem resultieren. Dies wird in Abbildung 5.7 deutlich. Hier ist der durchschnittliche Abstand der Merkmale von ihren Sollpositionen für den obigen



(a) erfolglos (b) erfolgreich

Abb. 5.8: Ausschnitte der Kamerabilder am Ende vom erfolgreichen und erfolglosen Regelverlauf

Testlauf in Blau dargestellt. In Rot ist zum Vergleich der gleiche Wert für eine erfolgreiche Regelung aufgetragen. Der unterschiedliche Regelverlauf ergibt sich dabei durch die Verwendung von verschiedenen initialen Jacobi-Matrizen. Es ist zu erkennen, dass die durchschnittliche Abweichung in beiden Fälle gegen Ende der Durchläufe ähnliche Werte aufweist. Die beiden Situation sind also schwer zu unterscheiden. Dies zeigen auch die beiden Ausschnitte der Kamerabilder mit den Sollmerkmalen (rote Kreise) und den Istmerkmale (bunte Makierung, Nummer) am Ende der erfolgreichen und der erfolglosen Regelung in Abbildung 5.8, die fast identisch sind.

Die beschriebene Situation stellt für jeden Regler eine Schwierigkeit dar. Für den adaptiven Regler ist das Problem noch größer, da er aus den gegeben Informationen zusätzlich die Jacobi-Matrix adaptieren muss. Befindet sich die Kamera in dieser Situation und führt auf Grund der geringen Abweichungen der Merkmalspositionen vom Soll kleine Bewegungen aus, dann gibt es nur wenige Kombinationen von Bewegungen in den fünf Dimensionen, die den Fehler zumindest nicht verschlechtern. Da die Abweichung der Merkmale aber bereits die Größenordnung von der Ziellage erreicht hat, kann der Fehler auch nicht mehr deutlich verkleinert werden. Vielmehr führen bereits geringe Abweichungen von diesen Bewegungen zu einer deutlichen Verschlechterung des Fehlers. Auf Grund von Abweichungen des linearen Modells von der tatsächlichen Abbildung ist es aber wahrscheinlicher, dass eine Bewegung ausgeführt wird, die den Fehler vergrößert. Daraufhin wird die Jacobi-Matrix so adaptiert, dass der Fehler wieder vermindert wird, was die Kamera wieder zurück in die ursprüngliche Lage führt. Da die Bewegungen relativ klein sind, verweilt der Roboterarm also stabil in diesem lokalen Minimum.

Dieses Problem kann normalerweise umgangen werden, indem anstatt der Auge-In-Hand Konfiguration eine externe Kamera verwendet wird, die sowohl den Roboterarm als auch das Objekt von außerhalb der Szene beobachtet. Dadurch kann die Rotation einfach von der Translation unterschieden werden. Um dies in dem hier vorgestellten Szenario anwenden zu können, muss die Kamera auf der mobilen Plattform angebracht werden. Die Ziellage des Roboterarms wird allerdings relativ zum Objekt definiert, dessen Lage zur Kamera veränderlich ist. Zur Ermittlung der Ziellage des Roboterarms im Bild der externen

Startlage	Δx [mm]	Δy [mm]	Δz [mm]	$\Delta\alpha$ [°]	$\Delta\gamma$ [°]
1	10	0	0	0	0
2	0	-10	0	0	0
3	0	0	10	0	0
4	0	0	0	-5,7296	0
5	0	0	0	0	5,7296
6	-6,1752	-2,4205	7,4838	3,1627	4,7785
7	2,5399	3,3930	19,5458	-9,6715	-6,1478
8	-20,9365	1,7505	21,4149	-8,4168	7,7808
9	18,3058	10,7762	27,8167	2,5325	11,1784
10	-34,7535	-16,4039	11,0954	-7,9641	8,2391

Tab. 5.8: Abweichungen der Startlagen vom Ziel

Kamera muss also die Lage des Objektes durch ein modellbasiertes Verfahren geschätzt werden. Auf Grund der Ungenauigkeiten und geringeren Robustheit eines solchen Verfahrens im Vergleich zum bildbasierten Ansatz, soll dies hier nicht angewendet werden.

5.4.1 Testszenario

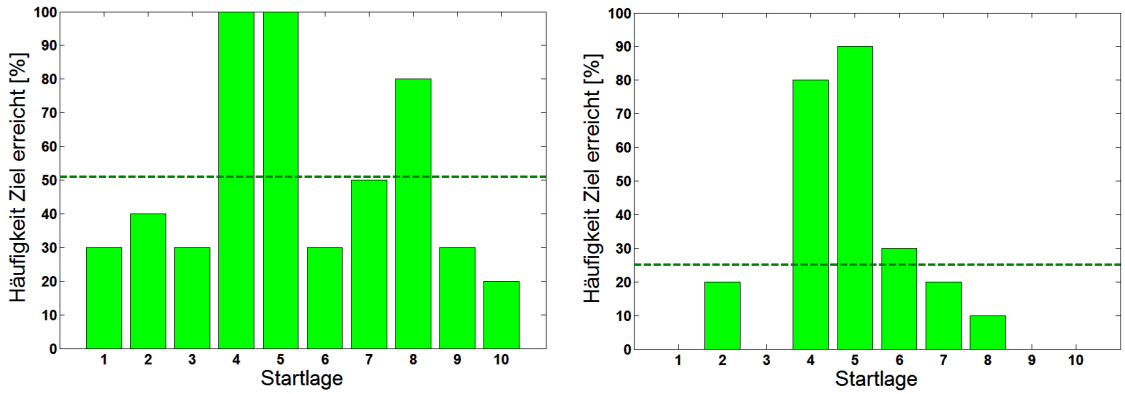
Im folgenden werden mehrere Ideen betrachtet, um das oben beschriebene Problem zu lösen. Da der genaue Verlauf und somit der Erfolg der Regelung von mehreren Faktoren abhängt, wie zum Beispiel die Startlage, die initiale Jacobi-Matrix und das Rauschen der Merkmale, wird zunächst ein Testszenario bestimmt, dass eine Aussage über den Erfolg oder Misserfolg einer Maßnahme erlaubt.

Da vor allem die relative Lage der Kamera vom Start zum Ziel für die Schwierigkeit der Regelaufgabe verantwortlich ist, wird für jeden Durchlauf die gleiche, oben beschriebene Ziellage mit dem Objekt auf dem Tisch verwendet. Die relativen Startpostionen und -orientierungen dazu sind in der Tabelle 5.4.1 angegeben. Die ersten fünf Lagen sind so gewählt, dass sich der Start nur in einer Dimension leicht vom Ziel unterscheidet. Die anderen fünf sind zufällige Werte, wobei sich die Distanz zum Ziel immer weiter erhöht.

Um die Abhängigikeit des Ergebnisses von den anderen Einflussfaktoren zu minimieren, wird jede der zehn Aufgaben zehn mal wiederholt. Dabei wird jedes Mal eine andere zufällige initiale Jacobi-Matrix verwendet. Dazu werden einmalig zehn zufällige Jacobi-Matrizen erzeugt und jede Regelaufgabe wird mit den gleichen zehn Jacobi-Matrizen getestet. Dadurch werden die Ergebnisse vergleichbar.

Für die Bewertung jedes Reglers werden also 100 Durchläufe durchgeführt. Mit der Benutzung der SIFT-Merkmale nimmt dies zu viel Zeit in Anspruch. Deshalb wird hier die in Abschnitt 3.3 vorgestellte Alternative verwendet. Jeder Regler wird einmal ohne und einmal mit Merkmalsrauschen getestet, um sowohl das Verhalten des Reglers ohne Störung als auch seine Robustheit gegen Fehler in der Merkmalsposition zu prüfen. Das Rauschen wird durch die Addition einer gauss-verteilte Zufallszahl auf die Merkmalsposition in u- und v-Richtung simuliert.

Es werden fünfzehn Merkmale verwendet, die in einer Ebene im Raum plaziert werden. Die Ebene befindet sich parallel zu Bildebene in der Zielposition mit dem Abstand 7 cm.



(a) ohne Rauschen (b) mit Rauschen

Abb. 5.9: Testergebnis des ursprünglichen Reglers

In allen Durchläufen werden dabei die gleichen Merkmalspositionen benutzt, die einmalig zufällig ermittelt werden.

Abbildung 5.9 zeigt die Testergebnisse des ursprünglichen Reglers. Links befindet sich das Diagramm ohne und rechts mit Merkmalsrauschen. Die Balken in den Diagrammen geben für jede Startposition die prozentuale Häufigkeit einer erfolgreichen Regelung an. Dabei gilt eine Regelung als erfolgreich, wenn die Differenz zwischen der Ist- und Solllage nach der Konvergenz des Reglers in allen Translationsrichtungen weniger als 1 mm und bei den Rotationen weniger als $0,573°$ beträgt. Die gestrichelte Linie befindet sich auf Höhe der Erfolgsquote aller 100 Durchläufe.

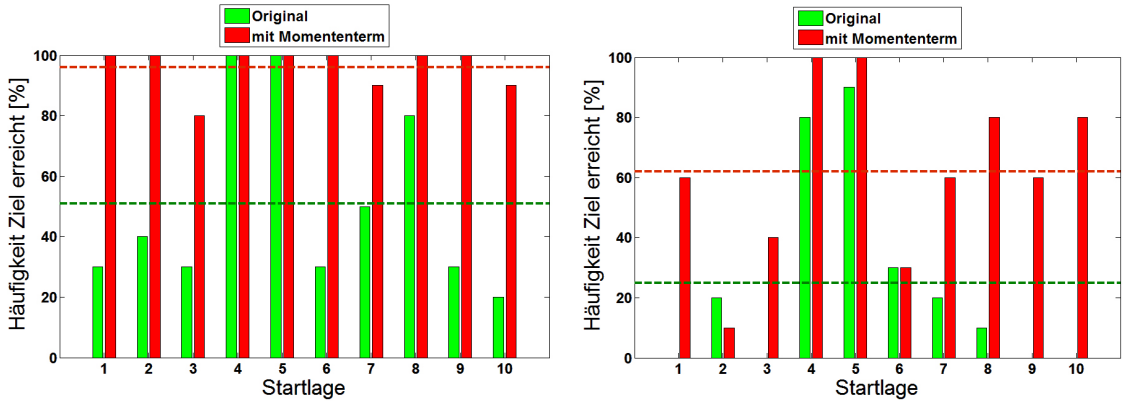
Für einen praktisch anwendbaren Regler ist eine hunderpozentige Erfolgsquote gefordert. Insbesondere wenn man bedenkt, dass hier eher ein einfaches Testszenario gewählt wurde. Die Startpositionen sind maximal $4cm$ vom Ziel entfernt und der Winkelfehler ist immer unter 12°. Außerdem sind die Merkmale relativ dicht an der Kamera, so dass sie eine große Fläche im Bild aufspannen und sehr stark auf Bewegungen der Kamera reagieren.

Die Diagramme in Abbildung 5.9 zeigen aber, dass der Regler das geforderte Ergebnis bei weitem nicht erfüllt. Vor allem mit Merkmalsrausch, bei dem der Regler erwartungsgemäß ein schlechteres Ergebnis erzielt, ist nur gut jeder vierte Durchlauf erfolgreich. Außerdem zeigt sich ein deutlicher Unterschied bei der Erfolgsquote der einzelnen Startpositionen. Daraus wird ersichtlich, dass sich bei diesem Regler einfache und schwierige Startlagen nicht einfach aus dem Abstand zwischen Start und Ziel ergeben. Dies macht Aussagen über den Regler schwierig, da Erkenntnisse für eine Startlage nicht automatisch auf alle Startlagen mit geringerem Abstand zum Ziel übertragen werden können.

Im Folgenden werden nun drei Maßnahmen vorgestellt und bewertet, die das Verhalten des Reglers verbessern. Abschließend wird das Gesamtergebnis interpretiert.

5.4.2 Momententerm

Das α-Y-Problem tritt dadurch auf, dass sich die Kamera in der Nähe einer Lage befindet, in der die Merkmalsfehlerfunktion bezüglich der meisten Bewegungsrichtungen ein lokales Minimum besitzt. Wie oben beschrieben ist es bei der Verwendung des adaptiven Reglers unwahrscheinlich, dass die Kamera dieses wieder verlässt.



(a) ohne Rauschen (b) mit Rauschen

Abb. 5.10: Testergebnis des Reglers mit Momententerm im Vergleich zum Ursprünglichen

Ziel der hier beschriebenen Verbesserung ist, zu verhindern, dass der Regler in ein solches Minimum gerät und dort bleibt. Dazu wird ein Momententerm für die Stellgrößen in Y- und α-Richtung benutzt. Der ursprüngliche Regler erzeugt im k-ten Schritt die Stellgröße $[\Delta y_k, \Delta\alpha_k]^T$ in diesen beiden Richtungen. Die Ausgabe des Reglers mit Momententerm für diese beiden Richtungen ergibt sich dann zu

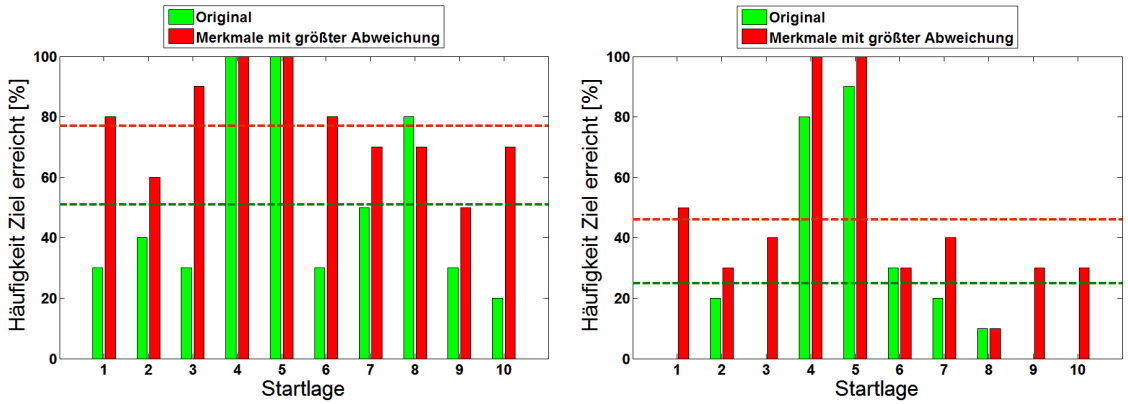
$$\begin{bmatrix} \Delta y_k^m \\ \Delta\alpha_k^m \end{bmatrix} = a \cdot \begin{bmatrix} \Delta y_{k-1}^m \\ \Delta\alpha_{k-1}^m \end{bmatrix} + (1-a) \cdot \begin{bmatrix} \Delta y_k \\ \Delta\alpha_k \end{bmatrix}$$

Die Stellgröße wird also nur noch zum Teil durch das aktuelle Ergebnis des Reglergesetzes bestimmt und zum Teil durch die vorherige Ausgabe, wobei der Reglerparameter $a \in [0,1]$ die Gewichtung bestimmt. Die übrigen Stellgrößen bleiben wie vorher. Dadurch können sich die beiden Stellgrößen in einem Regelschritt nicht so stark verändern. Auf Grund dieser Trägheit wird die Wahrscheinlichkeit erhöht, dass der Regler ein lokales Minimum sofort wieder verlässt, anstatt darin hängen zu bleiben. Allerdings führt dieses Verfahren zu einem verstärken Überschwingen im Ziel und allgemein zu einer langsameren Konvergenz.

Abbildung 5.10 zeigt das Testergebnis für dieses Verfahren. Sowohl mit Rauschen als auch ohne konnte das Ergebnis um über vierzig Prozentpunkte verbessert werden. Allerdings ist bei Startposition 2 mit Merkmalsrauschen zu erkennen, dass auf Grund der schlechteren Konvergenz auch ursprünglich erfolgreiche Läufe nun nicht mehr schnell genug ins Ziel führen.

5.4.3 Merkmale mit Größter Abweichung

In Abbildung 5.8 (a) ist zu erkennen, dass sich die einzelnen Merkmale in ihrem Abstand zur Sollposition unterscheiden. Dabei enthalten vor allem diejenigen mit einem großen Abstand die entscheidenden Informationen, um den Fehler der Kameralage zu verkleinern. Bei der Bestimmung der nötigen Bewegung durch das überbestimmte Gleichungssystem aus dem Reglergesetz 2.1 können die Merkmale mit der geringen Abweichung hinderlich sein. Deshalb werden bei der hier vorgestellten Verbesserung in jedem Regelschritt von



(a) ohne Rauschen (b) mit Rauschen

Abb. 5.11: Testergebnis des Reglers mit Auswahl der Merkmale mit der größten Abweichung von den Sollpositionen im Vergleich zum ursprünglichen Regler

den n Merkmalen nur die m Merkmale mit dem größten Abstand zu ihrer Sollposition für die Bestimmung der Stellgrößen verwendet.

Allerdings kann es bei diesem Verfahren vor allem in der Nähe der Ziellage dazu kommen, dass hauptsächlich die Merkmale mit dem größten Rauschen ausgewählt werden. Denn in der Nähe der Ziellage haben die Merkmale alle den gleichen geringen Abstand zu den Zielpositionen und Unterschiede ergeben sich nur noch durch das Rauschen in der Merkmalsposition. Somit ist es sehr wahrscheinlich, dass die Merkmale mit dem größten Abstand auch den größten Fehler in der Merkmalsposition haben. Dies hat negative Auswirkungen auf die Positioniergenauigkeit des Reglers.

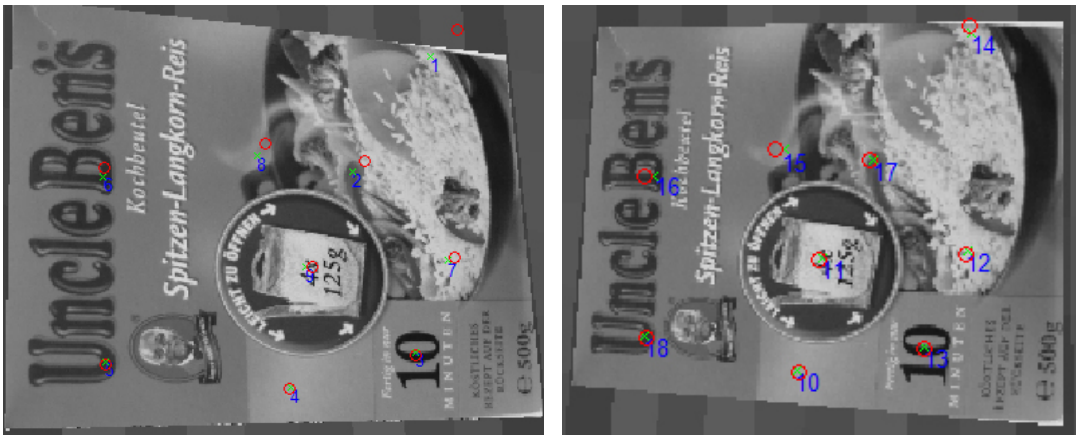
Die Diagramme in Abbildung 5.11 zeigen, dass diese Maßnahme in beiden Fällen eine Verbesserung von gut zwanzig Prozentpunkten im Vergleich zum ursprünglichen Regler erzielt. Auch hier fällt bei einer Startposition (in diesem Fall die Nummer 8 ohne Rauschen) auf, dass im Einzelfall die Erfolgsquote sinken kann. Dies liegt daran, dass hier unter Umständen auch brauchbare Informationen weggelassen werden.

5.4.4 Zwei Kameras

Abhängig von der Position der Kamera tritt das oben beschriebene Problem bei unterschiedlichen α-Winkeln auf. Werden zwei Kameras benutzt, deren Position sich hinreichend weit unterscheidet, dann befindet sich immer nur höchstens eine von beiden im lokalen Minimum. Da die Kameras am End-Effektor montiert werden müssen, ist der Unterschied in den Positionen durch bauliche Möglichkeiten begrenzt.

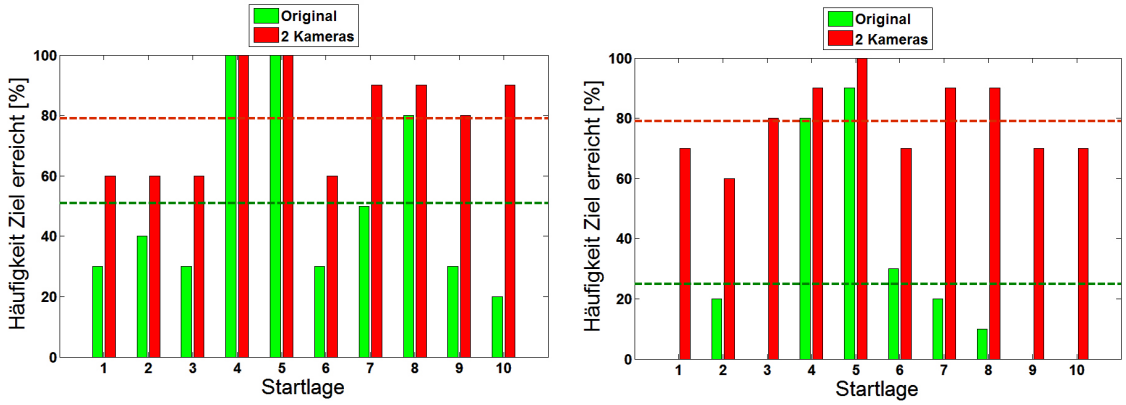
Die beiden Kameras werden entlang der Y-Achse nach innen gekippt und damit der Bereich vergrößert, in dem sich die Bilder der beiden Kameras überschneiden. Außerdem erhöht dies den Unterschied in der Information der beiden Kameras, was den Nutzen dieser Maßnahme erhöht. Ein Beispiel der Ansichten der beiden Kameras mit einem realistischen Abstand voneinander zeigt Abbildung 5.12.

Um den Test möglichst identisch zu den Fällen mit einer Kamera zu halten, werden hier sieben der ursprünglichen fünfzehn Merkmale für die eine und die restlichen acht für die



(a) Kamera 1 (b) Kamera 2

Abb. 5.12: Ausschnitte der Bilder der beiden Kameras in der Nähe eines lokalen Minimums



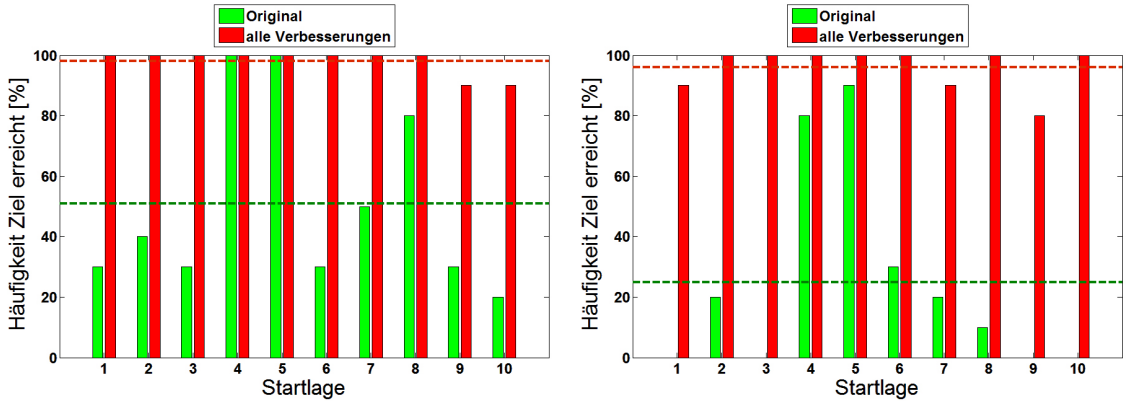
(a) ohne Rauschen (b) mit Rauschen

Abb. 5.13: Testergebnis des Reglers mit zwei Kameras im Vergleich zum Ursprünglichen

andere Kamera verwendet. So stehen dem Regler insgesamt auch nur fünfzehn Merkmale zur Verfügung. Beim tatsächlichen Einsatz stellt sich dies als ein zusätzlicher Vorteil von zwei Kameras heraus, da jedes stabil erkennbare Merkmal zweimal genutzt werden kann.

Wie die Diagramme in Abbildung 5.13 zeigen, wirkt sich diese Maßnahme vor allem auf den Fall mit den verrauschten Merkmalspositionen positiv aus. Hier wird das Ergebnis um über 50 Prozentpunkte verbessert, was das beste Ergebnis aller drei Maßnahmen ist. Der Unterschied zwischen den Fällen mit und ohne Rauschen ist bei dieser Verbesserung sehr gering. Die Robustheit des Reglers wurde also deutlich erhöht. Der Abstand zwischen den beiden Kamerapositionen reicht aber nicht aus, um alle Regelaufgaben erfolgreich zu absolvieren.

Außerdem sind einige Nachteile mit dieser Maßnahme verbunden. Erstens erhöhen sich durch den Kauf einer zweiten Kamera die Kosten. Zweitens vergrößert sich das Gewicht, dass der Roboterarm bereits ohne Objekt im Greifer tragen muss. Da der Roboterarm eine Obergrenze für die Nutzlast von 500 Gramm hat, können selbst bei relativ leichten Kameras nur noch sehr leichte Objekte gegriffen werden. Bei größeren Roboterarmen ist



(a) ohne Rauschen (b) mit Rauschen

Abb. 5.14: Testergebnis des Reglers mit allen Verbesserungen im Vergleich zum Ursprünglichen

dies ein geringeres Problem. Drittens muss die Bildverarbeitung für beide Kamerabilder durchgeführt werden. Die Bildverarbeitung wird somit noch mehr zum Flaschenhals der Berechnungen des Reglers. Viertens ist der Bereich, in dem ein Objekt von beiden Kameras erfasst wird kleiner als der Blickwinkel einer einzelnen Kamera.

5.4.5 Fazit

Keine der Verbesserungen für sich kann das geforderte Ergebnis erzielen. Weil sie an unterschiedlichen Stellen ansetzen, können sie aber auch gleichzeitig eingesetzt werden. In diesem Fall ergibt sich das Testergebnis in Abbildung 5.14. Mit einer Erfolgsquote von 98% ohne Rauschen und 96% mit Rauschen wurde das geforderte Ergebnis somit fast erreicht.

Wie in Abschnitt 5.4.1 beschrieben, wurde das Testszenario allerdings relativ einfach gewählt, so dass alles unter 100% nicht akzeptabel ist. Wird der Test etwas erschwert, zum Beispiel indem die Merkmale weiter von der Zielposition entfernt werden, wird das Ergebnis deutlich schlechter. Besonders negativ fällt auf, dass in der Startposition 1 im Fall mit Merkmalsrauschen das Ziel einmal nicht erreicht wurde. Dabei kommt diese Situation mit einer dominaten Abweichung vom Ziel in einer Dimension bei der äquidistanten Verteilung der Zwischenbilder (siehe Kapitel 8) relativ häufig vor.

In den beiden nächsten Kapitel wird deshalb ein alternativer Regler ohne Adaption vorgestellt, der die einzelnen Bewegungsrichtungen von einander entkoppelt.

6 Momentenbasierte Merkmalsparameter

Das entscheidende Problem des Reglers mit einer Auge-In-Hand Konfiguration ist die Unterscheidung zwischen der Translation in Y-Richtung und der Rotation um die α-Achse. Bei einem Roboterarm mit sechs Freiheitsgraden ergibt sich das gleiche Problem bei der X-Translation und der β-Rotation.

Um dieses Problem zu lösen, werden im folgenden Merkmalsparameter entwickelt, die die gegebenen Informationen besser ausnutzen. Dies bedeutet zum einen, dass mit der Orientierung und Skalierung der Merkmale zusätzliche Informationen des SIFT-Algorithmus ausgenutzt werden. Zum anderen wird für jede der sechs Dimensionen eine Statistik über die Merkmale als Merkmalsparameter definiert, die besonders sensitiv für die Bewegung des Roboterarms in dieser Dimension ist. Dabei ist entscheidend, dass der Merkmalsparameter bei einer auf diese Dimension beschränkten Bewegung streng monoton ist um die Konvergenz des Reglers zu gewährleisten. Allgemein werden diese Art von Merkmalsparameter als „momentenbasiert“ bezeichnet.

Solche generischen Merkmalparameter, die sich aus einer Statistik über alle im aktuellen Bild wiedererkannten Merkmalspunkte ergeben, haben den Vorteil, dass der darauf aufbauende Regler im aktuellen Kamerabild nur eine ausreichend große Teilmenge der Referenzmerkmale detektieren muss. Er ist somit nicht darauf angewiesen, bestimmte Merkmalspunkte in jeden Regelschritt wiederzufinden. Die generischen Merkmalsparameter bieten die Freiheit aus dem Satz detektierter SIFT-Merkmale eine Untermenge auszuwählen, die optimal bezüglich der Regelgüte ist.

In den folgenden Abschnitten werden die in dieser Arbeit entwickelten Merkmalsparameter für die einzelnen Dimensionen vorgestellt. Prinzipiell gibt es beliebig viele Möglichkeiten Merkmalsparameter zu definieren. Dabei ergibt sich die Güte der Merkmalsparameter aus experimentellen Ergebnissen. Die Merkmalsparameter werden also nicht exakt hergeleitet, sondern es werden lediglich diejenigen vorgestellt und motiviert, die in den Tests die gewünschten Eigenschaften aufwiesen.

Dazu wird angenommen, dass im aktuellen Bild n Merkmalspunkte erkannt und den entsprechenden Sollmerkmalen zugeordnet werden. Zu jedem Merkmal $i = 1 \dots n$ ermittelt der SIFT-Algorithmus neben der Position $\mathbf{p}_i = [u_i, v_i]^T$ die Hauptorientierung φ_i und die Skalierung σ_i. Die gleichen Informationen $(\hat{u}_i, \hat{v}_i, \hat{\varphi}_i, \hat{\sigma}_i)$ stehen für die zugehörigen Sollmerkmale zur Verfügung. Wenn nicht anders angegeben, dann wird die Berechnung der Merkmalsparameter an den Istmerkmalen demonstriert. Für die Sollmerkmalsparameter erfolgt die Vorgehensweise analog.

Die Betrachtung der Merkmalsparameter wird mit der m_γ begonnen, weil dieser genutzt werden kann, um die Rotation der Kamera um die Tiefenachse aus den Merkmalspositionen herauszurechnen. Die folgenden Merkmalsparameter werden mit diesen korrigierten Merkmalspositionen ermittelt und sind somit unempfindlich für die γ-Rotation.

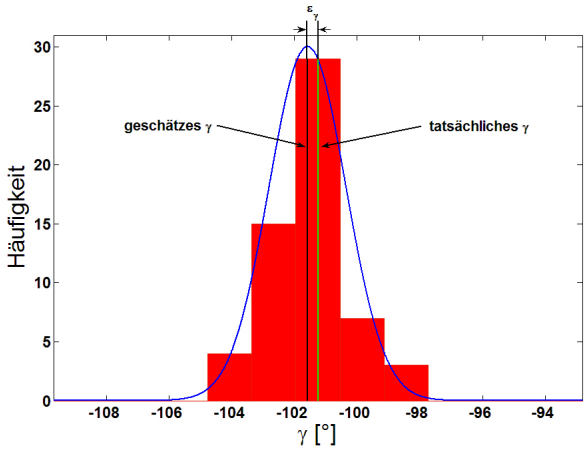


Abb. 6.1: Orientierungshistogramm für ein Bild

6.1 Merkmalsparameter m_γ

Zur Schätzung der Rotation der Kamera um die Tiefenachse wird die Orientierung der Merkmale benutzt, die der SIFT-Algorithmus aus den Gradienten der lokalen Umgebung berechnet. Dies hat den Vorteil, dass der Merkmalsparameter wenig sensitiv gegen die Bewegungen in den anderen Dimensionen ist und eine absolute Schätzung des Orientierungsfehlers möglich wird. Dadurch kann dieser Fehler aus den Merkmalspositionen herausgerechnet werden.

In der klassischen Reglerstruktur wird ein Istmerkmalsparameter und ein Sollmerkmalsparameter bestimmt, und der Merkmalsfehler ergibt sich aus ihrer Differenz. Dies führt allerdings zu praktischen Problemen auf Grund der Periodizität der Winkel. Da im Fall von γ keine analytische Form der Jacobi-Matrix im nächsten Kapitel berechnet werden muss, kann dieses Problem dadurch umgangen werden, dass aus den einzelnen Ist- und Sollwinkeln direkt das Fehlersignal berechnet wird.

$$\Delta m_\gamma = \frac{\sum_{i=1}^{n} (\hat{\varphi}_i - \varphi_i)}{n}$$

Δm_γ gibt also eine Schätzung des absoluten Fehlers in der Rotation der Kamera um die Z-Achse an. Abbildung 6.1 zeigt die annähernd gaussförmige Verteilung der $\gamma_i = \hat{\varphi}_i - \varphi_i$ der einzelnen Merkmale in ein Bild. Die durch den Mittelwert geschätzte Abweichung der Kamera vom Sollwert beträgt $101,25°$ und hat somit einen Fehler zur tatsächlichen Abweichung von ungefähr $\varepsilon_\gamma = 0,3°$.

Um die Robustheit der Schätzung zu bewerten, wird die Kamera vor einem Objekt platziert und von $-180°$ bis $180°$ in $\frac{180°}{64}$ Schritten rotiert. In Abbildung 6.3 ist der dabei aufgetretene Fehler in der Schätzung ε_γ in Abhängigkeit von der tatsächlichen Rotation γ dargestellt, wobei $0°$ die Sollposition bildet. Es ist zu erkennen, dass der Fehler betragsmäßig nie größer als $0,6°$ ist. Wie das Histogramm in Abbildung 6.2 zeigt, ist der Fehler meistens sogar deutlich kleiner. Der mittlere Fehler $\bar{\varepsilon}_\gamma$ in diesem Test beträgt nur ungefähr $-0,1°$ mit einer Standartabweichung von $0,18°$. Damit ist eine zuverlässige Regelung möglich.

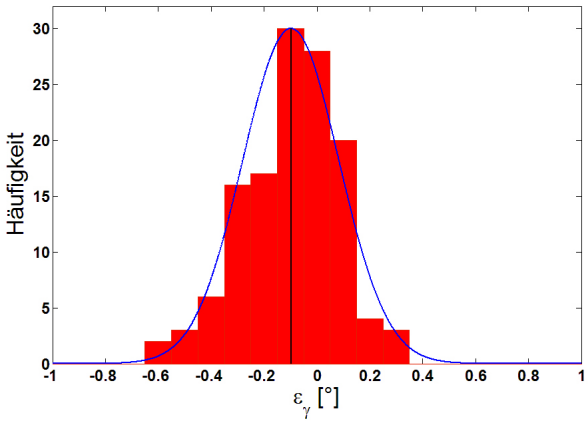


Abb. 6.2: Histogramm des Fehlers ε_γ

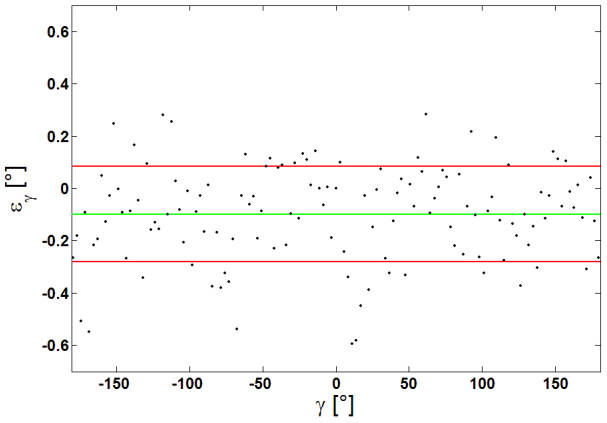


Abb. 6.3: Verteilung der Fehler ε_γ in Abbhängigkeit von der tatsächlichen Rotation

$\Delta\alpha$ [°]	0	5	10	15	20	25	30
$\lvert\bar{\varepsilon}_\gamma\rvert$ [°]	0,05	0,18	0,25	0,27	0,56	1,17	1,18

Tab. 6.3: Durchschnittlicher Fehler der Orientierungschätzung in Abbhängigkeit von $\Delta\alpha$

Im Folgenden wird die Sensitivität dieses Merkmalsparameters für Bewegungen in den anderen Dimensionen betrachtet. Eine Translation der Kamera in X- oder Y-Richtung verändert die lokalen Gradienten nicht, somit ist das Merkmal unabhängig von diesen Bewegungen. Gleiches gilt für die Z-Richtung, weil für die Berechnung der Hauptorientierung die passende Skalierungsebene gewählt wird. Allerdings hat eine α-Rotation Auswirkungen auf die Umgebung des Merkmals. Wie Tabelle 6.1 zeigt, wird dadurch die Schätzung nicht entscheidend verschlechtert. Selbst bei einem α-Fehler von 30° bleib der Fehler der Schätzung im Schnitt deutlich unter 2°. Dies ist für ist für eine erfolgreiche Regelung mehr als ausreichend. Bei größeren α-Fehlern ist das Objekt entweder nicht mehr zu sehen oder es wird ein Zwischenbild (siehe Kapitel 8) verwendet.

Abb. 6.4: Korrektur der Merkmalsposition

Der besondere Vorteil dieses Merkmalsparameters ist, dass er eine sehr genaue Schätzung des Rotationsfehlers um die γ-Achse ermöglicht. Somit kann damit nicht nur ein Regelschritt berechnet werden, der diesen Fehler in der Kameralage verkleinert, sondern auch aus den aktuellen Merkmalspositionen die Auswirkungen dieses Fehlers herausgerechnet werden. Dazu werden die Koordinaten entsprechend rotiert. Die neuen Merkmalspositionen ergeben sich zu:

$$\begin{bmatrix} u_i' \\ v_i' \end{bmatrix} = \begin{bmatrix} cos(\Delta m_\gamma) & -sin(\Delta m_\gamma) \\ sin(\Delta m_\gamma) & cos(\Delta m_\gamma) \end{bmatrix} \cdot \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Somit können die weiteren Merkmalsparameter ohne den Einfluss des γ-Fehlers bestimmt werden. Durch diese Methode wird das Kamerarückzugsproblem gelöst. In Abbildung 6.4 ist diese Situation dargestellt. Die blauen Kreuze markieren die tatsächlichen Positionen der Merkmale, die in diesem Bild wiedergefunden wurden. Nach der Korrektur befinden sie sich an den grünen Plus-Zeichen, die sich fast exakt an den Sollpositionen (rote Kreise) befinden. Da die neuen Merkmalspositionen für die Berechnung der übrigen Parameter genutzt werden, weisen diese nur einen geringen Fehler auf. Somit kann ein darauf aufbauender Regler erkennen, dass der optimale Weg ins Ziel eine Rotation um die Tiefenachse ist.

Bei der Anwendung muss noch beachtet werden, das die Korrektur der Merkmalspositionen Auswirkung auf die Bestimmung der Lage der α-Rotationsachse im Kamerakoordinatensystem hat. In diesem Fall ergibt sich $\Delta\alpha$ aus $\Delta\alpha'$ und $\Delta\beta'$ (siehe Abschnitt 5.3.2) wie folgt:

$$\begin{aligned} \theta &= \gamma - atan2(x, y) \cdot sin(\alpha) + \Delta m_\gamma \\ \Delta\alpha &= cos(\theta) \cdot \Delta\alpha' + sin(\theta) \cdot \Delta\beta' \end{aligned}$$

6.2 Merkmalsparameter m_x, m_y

Wenn die Ebene, die durch die Merkmale aufgespannt wird, mit der Bildebene parallel ist, dann führt eine Bewegung der Kamera in X-, Y-Richtung zu einer einheitlichen Positionsänderung der Merkmale im Bild. Diese Positionsänderung ist proportional zur Bewegung der Kamera, wobei der Proportionalitätsfaktor von der Entfernung der Merkmale zur Kamera und deren Brennweite abhängt. Diese Merkmalsparameter erfüllen somit die Anforderung der Monotonie:

$$\begin{aligned} m_x &= \frac{\sum_{i=1}^{n} u'_i}{n} \\ m_y &= \frac{\sum_{i=1}^{n} v'_i}{n} \end{aligned}$$

Es wird also ein neuer Merkmalspunkt $[m_x, m_y]^T$ bestimmt, wobei durch die Mittelung über alle Merkmale die Ungenauigkeit in der Position reduziert wird. Ansonsten gilt für diesen Merkmalspunkt das gleich wie für jeden einzelnen. Die Sensitivität kann also aus der Jacobi-Matrix in Gleichung 2.2 abgelesen werden. Die Merkmalsparameter für X- und Y-Richtung sind also unempfindlich für die jeweils andere Richtung. Dafür sind sie sensitiv für eine Z-Translation und die drei Rotationen, wobei der Fehler in der γ-Rotation bereits aus den Merkmalspositionen herausgerechnet wurde und somit hier keine Rolle mehr spielt.

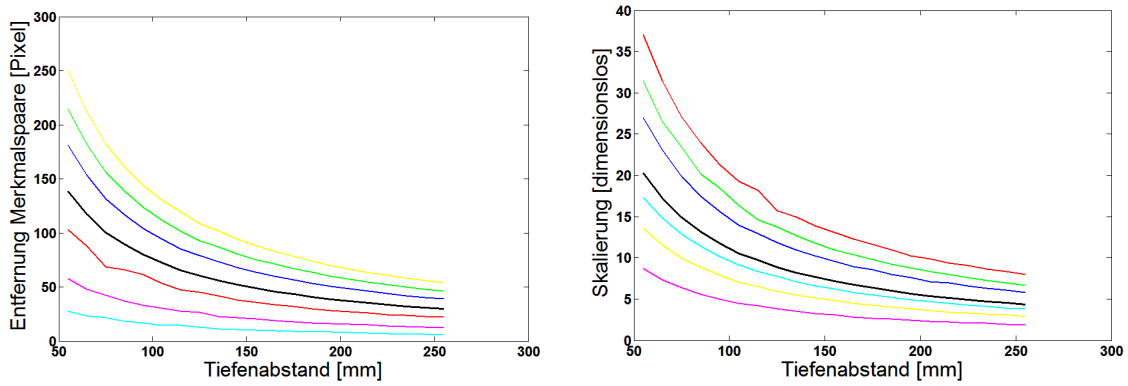
Die Abhängigkeit dieser Merkmalsparameter von den anderen Bewegungsrichtungen kann dazu führen, dass ein Fehler in diesen Merkmalsparametern zu einer Bewegung in X- beziehungsweise Y-Richtung führt, obwohl der Fehler durch eine Abweichung in den anderen Dimensionen verursacht wird. Die Kamera entfernt sich also in X- beziehungsweise Y-Richtung vom Ziel. Da der Fehler in den anderen Dimensionen verkleinert wird, wird diese falsche Bewegung im weiteren Regelverlauf wieder korrigiert, so dass das Ziel letztendlich erreicht wird. Die zwischenzeitliche Verschlechterung der Kamerapositionen in diesen beiden Dimensionen hat allerdings den Vorteil, dass dadurch der Schwerpunkt der Merkmale in Richtung seiner Sollposition bewegt wird. Diese befindet sich meistens in der Nähe der Bildmitte, so dass durch diese Bewegung die Sichtbarkeit der Merkmale verbessert wird.

6.3 Merkmalsparameter m_{ze}, m_{zs}

Für die Z-Richtung werden hier zwei alternative Varianten des Merkmalsparameters vorgestellt. Die Vor- und Nachteile werden allerdings erst im Zusammenhang mit dem Reglerdesign deutlich. Deshalb wird darauf erst im nächsten Kapitel eingegangen.

Entfernt sich die Kamera von einem Objekt entlang der optischen Achse, dann bewegen sich die Merkmale im Bild in Richtung des Principal Point, in dem sie sich bei einer unendlich großen Entfernung treffen. Durch die Bewegung verändert sich somit auch die relative Position der Merkmale. Als erstes Kriterium wird deshalb die Entfernung der Merkmalspunkte zueinander genutzt:

$$m_{ze} = \frac{\sum_{i=1}^{n} \sum_{j=i+1}^{n} \left\| \mathbf{p}'_j - \mathbf{p}'_i \right\|_2}{\frac{n}{2} \cdot (n-1)}$$



(a) Entfernungsvariante (b) Skalierungsvariante

Abb. 6.5: Verlauf des Merkmalsparameters in Abhängigkeit von der Entfernung

Wie die schwarze Linie in Abbildung 6.5 (a) zeigt, fällt der Merkmalsparameter streng monoton ab und da Bewegungen in X- und Y-Richtung die Entfernungen zwischen den Merkmalen nicht beeinflusst, ist der Merkmalsparameter für diese Bewegungen unempfindlich. Allerdings wird er durch α- und β-Rotationen verändert. Wie der nächste Unterabschnitt allerdings zeigt, bleibt der Fehler aber hinreichend klein.

Alternativ wird die Skalierungsinformation der SIFT-Merkmale für den Merkmalsparameter in Z-Richtung verwendet:

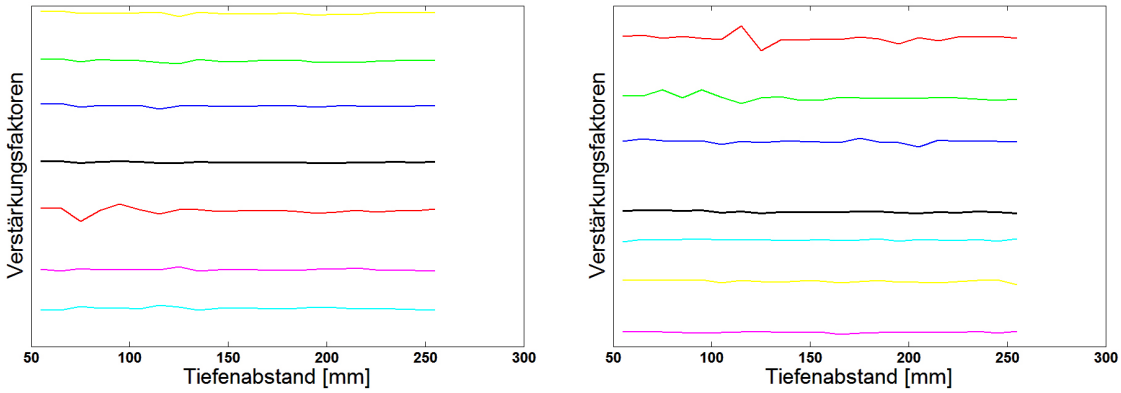
$$m_{zs} = \frac{\sum_{i=1}^{n} \sigma_i}{n}$$

In Abbildung 6.5 (b) ist zu erkennen, dass dieses Merkmal einen ähnlichen Verlauf aufweist. Auch für die Sensitivität des Merkmals gilt das gleiche wie oben. In den beiden Abbildungen sind zusätzlich in verschiedenen Farben die Verläufe für einzelne Merkmale beziehungsweise Merkmalspaare eingezeichnet. Es ist zu erkennen, dass jedes Merkmal für sich bereits die nötige Eigenschaft für den Merkmalsparameter erfüllt. Durch die Mittelung über alle Merkmale beziehungsweise Merkmalspaare wird aber die Empfindlichkeit gegen Rauschen verbessert.

6.3.1 Z-Schätzung

Die Abbildung 6.5 legt die Vermutung nahe, dass die Verläufe proportional zu $1/z$ sind. Dies wird verifiziert, indem die Graphen mit z multipliziert werden. Wie in Abbildung 6.6 zu sehen, bleibt dann nur noch der Proportionalitätsfaktor k_i übrig. Die Vermutung kann also experimentell verifiziert werden. Dabei hängt der Proportionalitätsfaktor von der Brennweite der Linse und der initialen Skalierung beziehungsweise Entfernung der Merkmale ab.

Ist die Entfernung zum Objekt in einem Kalibrierungspunkt bekannt, kann somit die Entfernung zum Ziel anhand der im aktuellen Bild wiedergefundenen Merkmale geschätzt werden. In dem hier gegebenen Szenario wird als Kalibirierungspunkt die Zielposition genommen. Für die Anwendung des Greifalgorithmus muss der Roboterarm in eine bestimmte Position relativ zum Objekt gebracht werden, somit ist die Entfernung zum Ziel



(a) Entfernungsvariante (b) Skalierungsvariante

Abb. 6.6: Verifikation der $1/z$ Proportionalität

in dieser Position relativ genau bekannt. Die Z-Schätzung wird im Kapitel 7 für eine der vorgestellten Reglervarianten benötigt.

Für die Berechnung wird angenommen, dass $\hat{z}$ die Entfernung zum Objekt in der Zielposition und z die gesuchte Entfernung im aktuellen Bild ist. Die Entfernung kann einzeln für jedes Merkmal geschätzt und anschließend gemittelt oder direkt mit den Merkmalsparametern berechnet werden. Das Ergebnis ist mathematisch identisch. Wie oben beschrieben gilt:

$$\hat{m}_{ze} = \frac{k_e}{\hat{z}} \quad \text{bzw.} \quad \hat{m}_{zs} = \frac{k_s}{\hat{z}} \quad \text{und}$$

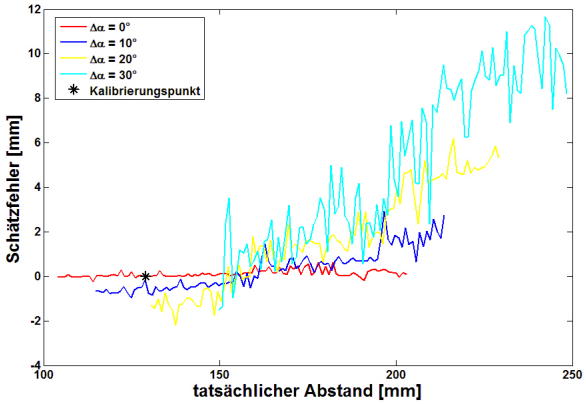
$$m_{ze} = \frac{k_e}{z} \quad \text{bzw.} \quad m_{zs} = \frac{k_s}{z}$$

wobei k_e beziehungsweise k_s der unbekannte Proportionalitätsfaktor des Merkmalsparameters ist. Durch Umstellen und Einsetzen ergibt sich:

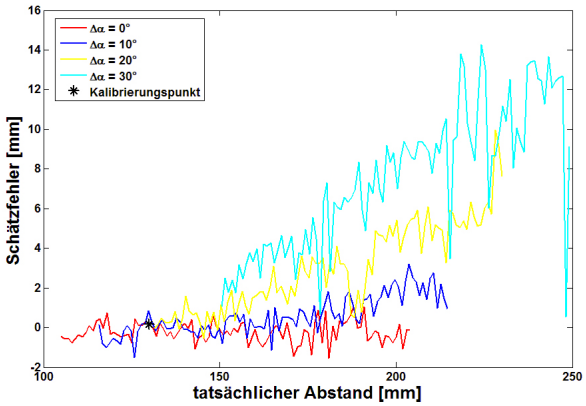
$$z = \frac{\hat{m}_{ze} \cdot \hat{z}}{m_{ze}} \quad bzw. \quad z = \frac{\hat{m}_{zs} \cdot \hat{z}}{m_{zs}}$$

Abbildung 6.7 zeigt den Fehler in der Schätzung in Abbhängigkeit vom Abstand für verschiedene Fehler in der α-Rotation. Der Kalibrierungspunkt ist dabei mit einem schwarzen Stern markiert. Beide Varianten weisen auch im schlimmsten Fall einen Schätzfehler von unter 15 mm auf, der in der Nähe der Zielposition und bei geringem α-Fehler nochmal deutlich geringer ist. Somit sind beide Varianten für die Regelung geeignet, wobei sich die genauen Vor- und Nachteile aus der Anwendung in einem Regler ergeben. Bei den jeweils 4 Testläufen, die in Abbildung 6.7 dargestellt sind, wird ein leicht unterschiedlicher Entfernungsbereich abgedeckt, um den Roboteram im Arbeitsbereich und die Merkmale im Bild zu halten.

Der Abstand zu den Merkmalen könnte ähnlich zur γ-Rotation dazu verwendet werden, um den Z-Fehler aus den Merkmalspositionen herauszurechnen und so die Entkopplung weiter zu verbessern. Davon wird hier aber abgesehen, da ein Fehler in Z-Richtung deutlich



(a) Entfernungsvariante



(b) Skalierungsvariante

Abb. 6.7: Fehler in der Z-Schätzung in Abhängigkeit vom Abstand bei verschiedenen Abweichungen in α-Richtung

geringere Probleme für den Regler verursacht als ein γ-Fehler, und somit der mögliche Gewinn sehr viel kleiner ist. Außerdem wird die Abhängigkeit des Reglers von der korrekten Einstellung des Kalibrierungspunktes erhöht. Des weiteren werden für die Korrekturberechnungen die intrinsischen Kameraparameter benötigt, also zusätzliche Kalibrierungsparameter, was hier so weit es geht vermieden werden soll.

6.4 Merkmalsparameter $m_{\alpha'}$, $m_{\beta'}$

Wie im Abschnitt 5.3.2 erläutert, werden auch bei dem hier verwendeten Roboterarm Merkmalsparameter für alle sechs Freiheitsgrade benötigt. Außerdem wird der entwickelte Regler so auch einsetzbar bei Roboterarmen mit allen sechs Freiheitsgraden. Die Rotationen um die statische X- und Y- Achse des Kamerakoordinatensystems werden analog zu Abschnitt 5.3.2 mit α' und β' bezeichnet. Im folgenden wird der Merkmalsparameter für α' motiviert. Derjenige für β' ergibt sich analog.

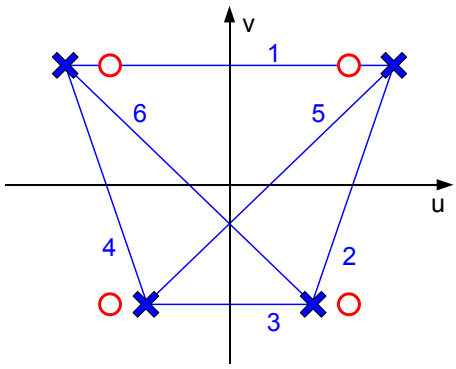


Abb. 6.8: Motivation des α'-Merkmalsparameters

Abbildung 6.8 zeigt ein Beispiel von vier Merkmalen, die im Sollbild ein Quadrat bilden (rote Kreise). Bei einem Fehler der Istlage in der α'-Rotation, bilden die Istmerkmale (blaue Kreuze) ein Trapez, wobei in dem Bild davon ausgegangen wird, dass durch eine Translation der Kamera die Merkmale in der Bildmitte bleiben. Diese Veränderung der Merkmalspositionen muss in abstrahierter Form durch den Merkmalsparameter ausgedrückt werden, da die tatsächlich verwendeten Merkmale normalerweise nicht in dieser strukturierten Art vorliegen. Als erfolgreich hat sich dabei ein Merkmalsparameter herausgestellt, der auf dem durchschnittlichen Abstand der Merkmale zueinander aufbaut:

$$\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}'_j-\mathbf{p}'_i\right\|_2}{\frac{n}{2}\cdot(n-1)}$$

Allerdings gibt es bei einer α'-Rotation Abstände die kürzer und Abstände die länger werden, wie in Abbildung 6.8 zum Beispiel an Linie 1 und 3 zu erkennen ist. Diese Entfernungen müssen also einen entgegengesetzten Beitrag zum Merkmalsparameter liefern. Dies wird erreicht, in dem die Abstände mit der negativen Summe der beiden v-Koordinaten der zugehörigen Sollpositionen gewichtet werden, weil die v-Komponenten der beiden Linien ein unterschiedliches Vorzeichen besitzen. Außerdem ist diese Summe für die anderen Entfernungen (2,4,5,6), die wenig Information über die α-Rotation beinhalten und deshalb möglichst wenig in den Merkmalsparameter einfließen sollen, sehr klein:

$$\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i-\hat{v}_j)\cdot\left\|\mathbf{p}'_j-\mathbf{p}'_i\right\|_2}{\frac{n}{2}\cdot(n-1)}$$

Der Merkmalsparameter in dieser Form ist ähnlich sensitiv für Translationen entlang der Z-Achse wie m_{ze} oder m_{zs}. Um die Unabhängigkeit des Merkmalsparameter von dieser Bewegung zu erzielen, muss der Anteil der Längenänderung verursacht durch den Z-Fehler von dem hervorgerufen durch den α-Fehler unterschieden werden. Dies ist mit Hilfe der Z-Schätzung prinzipiell möglich, soll aus den oben genannten Gründen hier aber nicht angewendet werden. Stattdessen wird der Anteil aus dem Z-Fehler durch die durchschnittliche Längenänderung zwischen den Merkmalen approximiert. Beim Merkmalsparameter

muss also der durchschnittliche Abstand aller Merkmale m_{ze} vom Abstand der Merkmale abgezogen werden, damit bei der Differenz aus Soll- und Istmerkmalsparameter der gewünschte Effekt eintritt:

$$m_{\alpha'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i - \hat{v}_j)\cdot\left(\left\|\mathbf{p}'_j - \mathbf{p}'_i\right\|_2 - m_{ze}\right)}{\frac{n}{2}\cdot(n-1)}$$

Analog ergibt sich der Merkmalsparameter für β':

$$m_{\beta'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{u}_i - \hat{u}_j)\cdot\left(\left\|\mathbf{p}'_j - \mathbf{p}'_i\right\|_2 - m_{ze}\right)}{\frac{n}{2}\cdot(n-1)}$$

Entsprechend gilt für die Sollmerkmalsparameter:

$$\hat{m}_{\alpha'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i - \hat{v}_j)\cdot\left(\left\|\hat{\mathbf{p}}'_j - \hat{\mathbf{p}}'_i\right\|_2 - \hat{m}_{ze}\right)}{\frac{n}{2}\cdot(n-1)}$$

$$\hat{m}_{\beta'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{u}_i - \hat{u}_j)\cdot\left(\left\|\hat{\mathbf{p}}'_j - \hat{\mathbf{p}}'_i\right\|_2 - \hat{m}_{ze}\right)}{\frac{n}{2}\cdot(n-1)}$$

Auf Grund der Approximation bleiben die Merkmalsparameter sensitiv für Bewegungen in Z-Richtung, die Abhängigkeit wird aber reduziert. Außerdem sind sie empfindlich für Rotationen um die jeweils andere Achse, weil dadurch die Entfernung zwischen den Merkmalspunkten verändert wird. Durch die Benutzung der Sollpositionen als Gewichtung sind die Merkmalsparameter allerdings unempfindlich gegen Translationen in X- und Y-Richtung.

7 Momentenbasierte Regler

Aufbauend auf den im vorherigen Kapitel entwickelten Merkmalsparamtern werden in diesem Kapitel zunächst zwei Regleransätze vorgestellt und anschließend in Abschnitt 7.3 getestet. Für beide Relger gilt, dass der γ-Fehler getrennt von den anderen Dimensionen durch einen einfachen Proportionalregler ausgeregelt wird. Als Eingabe erhalten die Regler also den Merkmalsfehler in der Entfernungsvariante

$$\begin{aligned}\Delta\mathbf{m}_e &= \hat{\mathbf{m}}_e - \mathbf{m}_e \\ &= [\hat{m}_x, \hat{m}_y, \hat{m}_{ze}, \hat{m}_{\alpha'}, \hat{m}_{\beta'}]^T - [m_x, m_y, m_{ze}, m_{\alpha'}, m_{\beta'}]^T\end{aligned}$$

beziehungsweise der Skalierungsvariante

$$\begin{aligned}\Delta\mathbf{m}_s &= \hat{\mathbf{m}}_s - \mathbf{m}_s \\ &= [\hat{m}_x, \hat{m}_y, \hat{m}_{zs}, \hat{m}_{\alpha'}, \hat{m}_{\beta'}]^T - [m_x, m_y, m_{zs}, m_{\alpha'}, m_{\beta'}]^T\end{aligned}$$

des Z-Merkmalsparameters. Die Ausgabe ist zunächst eine Kamerabewegung in fünf Dimensionen $\Delta\mathbf{X} = [\Delta x, \Delta y, \Delta z, \Delta\alpha', \Delta\beta']^T$. $\Delta\alpha'$ und $\Delta\beta'$ werden dann, wie in Abschnitt 6.1 beschrieben, zu $\Delta\alpha$ umgerechnet.

7.1 Analytische Herleitung der Jacobi-Matrix

Die erste Reglervariante setzt das Reglergesetz 2.1 für diese Merkmalsparameter um. Dazu muss die analytische Form der Jacobi-Matrix berechnet werden. Dies ist nur für die Entfernungsvariante der Merkmalsparameter möglich, so dass dieser Regler nur mit diesem Merkmalsparameter angewendet werden kann.

$[m_x, m_y]^T$ verhält sich wie ein einzelner Merkmalspunkt. Entsprechend können die ersten beiden Zeilen der Jacobi-Matrix aus Gleichung 2.2 übernommen werden, nur das auch hier über alle Merkmale gemittelt werden muss:

$$\mathbf{J}_{x,y} = \frac{\sum_{i=1}^{n} \begin{bmatrix} \frac{\lambda}{z} & 0 & \frac{-u_i}{z} & \frac{-u_i v_i}{\lambda} & \frac{\lambda^2+u_i^2}{\lambda} \\ 0 & \frac{\lambda}{z} & \frac{-v_i}{z} & \frac{-\lambda^2-v_i^2}{\lambda} & \frac{u_i v_i}{\lambda} \end{bmatrix}}{n}$$

Dabei wird angenommen, dass alle Merkmale die gleiche Tiefe z haben, die sich aus einer der beiden in Abschnitt 6.3.1 beschriebenen Varianten der Z-Schätzung ergibt. Diese Annahme ist sinnvoll, da der Unterschied in der Tiefe der einzelnen Merkmale im Vergleich zum Abstand zur Kamera klein ist und dadurch die nachfolgende Rechnung deutlich vereinfacht wird.

Die übrigen drei Zeilen der Jacobi-Matrix müssen nun für die entsprechenden Merkmalsparameter einzeln hergeleitet werden. Dazu wird der Merkmalsparameter zunächst umgeformt und dann nach der Zeit abgeleitet, um die Abbhängigkeit der Änderung des Merkmalsparamters von den Kamerageschwindigkeiten $[T_x, T_y, T_y, \omega_{\alpha'}, \omega_{\beta'}]^T$ zu erhalten. Diese werden im Reglergesetzt durch die gesuchte Bewegung $[\Delta x, \Delta y, \Delta z, \Delta\alpha', \Delta\beta']^T$ ersetzt. Für

$$m_{ze} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} \left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2}{\frac{n}{2}\cdot(n-1)}$$

gilt nach Auflösen der euklidischen Norm:

$$m_{ze} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} \sqrt{(u_j-u_i)^2+(v_j-v_i)^2}}{\frac{n}{2}\cdot(n-1)}$$

Wie in Abschnitt 2.2 beschrieben gilt $u_i = \lambda \cdot x_i/z$ und für u_j, v_i und v_j entsprechendes. Durch Einsetzen ergibt sich:

$$m_{ze} = \frac{\lambda}{z}\frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n} \sqrt{(x_j-x_i)^2+(y_j-y_i)^2}}{\frac{n}{2}\cdot(n-1)}$$

Die Ableitung der Summe wird zur besseren Übersicht an einem einzelnen Summanden demonstriert:

$$\begin{aligned}
\dot{m}'_{ze} &= \left(\frac{\lambda}{z}\dot{\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}}\right)\\
&= \lambda\frac{z\cdot\dot{\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}} - \sqrt{(x_j-x_i)^2+(y_j-y_i)^2}\cdot\dot{z}}{z^2}\\
&= \lambda\frac{z\cdot\frac{1}{2}(\dot{(x_j-x_i)^2+(y_j-y_i)^2}) - ((x_j-x_i)^2+(y_j-y_i)^2)\cdot\dot{z}}{z^2\cdot\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}}\\
&= \lambda\frac{z\cdot((x_j-x_i)\cdot(\dot{x_j}-\dot{x_i})+(y_j-y_i)\cdot(\dot{y_j}-\dot{y_i})) - ((x_j-x_i)^2+(y_j-y_i)^2)\cdot\dot{z}}{z^2\cdot\sqrt{(x_j-x_i)^2+(y_j-y_i)^2}}
\end{aligned}$$

Umgekehrt zu oben wird nun $x_i = z\cdot u_i/\lambda$ und x_j, y_i und y_j entsprechend eingesetzt:

$$\begin{aligned}
\dot{m}'_{ze} &= \lambda\frac{\frac{z^2}{\lambda}\cdot((u_j-u_i)\cdot(\dot{x_j}-\dot{x_i})+(v_j-v_i)\cdot(\dot{y_j}-\dot{y_i})) - \frac{z^2}{\lambda^2}((u_j-u_i)^2+(v_j-v_i)^2)\cdot\dot{z}}{\frac{z^3}{\lambda}\cdot\sqrt{(u_j-u_i)^2+(v_j-v_i)^2}}\\
&= \frac{\lambda\cdot((u_j-u_i)\cdot(\dot{x_j}-\dot{x_i})+(v_j-v_i)\cdot(\dot{y_j}-\dot{y_i})) - ((u_j-u_i)^2+(v_j-v_i)^2)\cdot\dot{z}}{z\cdot\sqrt{(u_j-u_i)^2+(v_j-v_i)^2}}
\end{aligned}$$

Nach [CHH96] gilt $\dot{x}_j - \dot{x}_i = -\omega_z(y_j-y_i)$ und $\dot{y}_j - \dot{y}_i = \omega_z(x_j-x_i)$. Nach dem Einsetzen ergibt sich:

$$\begin{aligned}
\dot{m}'_{ze} &= \frac{-\dot{z}}{z}\cdot\sqrt{(u_j-u_i)^2+(v_j-v_i)^2}\\
&= \frac{-\dot{z}}{z}\cdot\left\|\mathbf{p}_j-\mathbf{p}_i\right\|_2
\end{aligned}$$

Außerdem gilt $\dot{z} = T_z + \omega_x \cdot y - \omega_y \cdot x$, wobei hier $x = (x_i + x_j)/2$ und $y = (y_i + y_j)/2$ gesetzt wird. Für die gesamte Summe ergibt sich dann:

$$\begin{aligned}\dot{m}_{ze} &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\left(\frac{-T_z}{z} - \frac{\omega_x(y_i+y_j)}{2\cdot z} + \frac{\omega_y(x_i+x_j)}{2\cdot z}\right)}{\frac{n}{2}\cdot(n-1)} \\ &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\left(\frac{-T_z}{z} - \frac{\omega_x(v_i+v_j)}{2\cdot\lambda} + \frac{\omega_y(u_i+u_j)}{2\cdot\lambda}\right)}{\frac{n}{2}\cdot(n-1)}\end{aligned}$$

Die Ableitung für $m_{\alpha'}$ wird auf ähnliche Weise berechnet. Durch Umformen gilt:

$$\begin{aligned}m_{\alpha'} &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i - \hat{v}_j)\cdot\left(\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2 - m_{ze}\right)}{\frac{n}{2}\cdot(n-1)} \\ &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i - \hat{v}_j)\cdot\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2}{\frac{n}{2}\cdot(n-1)} + \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\hat{v}_i + \hat{v}_j)}{\frac{n}{2}\cdot(n-1)}\cdot m_{ze}\end{aligned}$$

Da $(-\hat{v}_i - \hat{v}_j)$ eine Konstante ist, ergibt sich die Ableitung unmittelbar aus der Berechnung von $\dot{m}_{ze}$:

$$\begin{aligned}\dot{m}_{\alpha'} &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{v}_i - \hat{v}_j)\cdot\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\left(\frac{-T_z}{z} - \frac{\omega_x(v_i+v_j)}{2\cdot\lambda} + \frac{\omega_y(u_i+u_j)}{2\cdot\lambda}\right)}{\frac{n}{2}\cdot(n-1)} \\ &\quad + \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\hat{v}_i + \hat{v}_j)}{\frac{n}{2}\cdot(n-1)}\cdot\dot{m}_{ze}\end{aligned}$$

Analoges gilt für $m_{\beta'}$:

$$\begin{aligned}\dot{m}_{\beta'} &= \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(-\hat{u}_i - \hat{u}_j)\cdot\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\left(\frac{-T_z}{z} - \frac{\omega_x(v_i+v_j)}{2\cdot\lambda} + \frac{\omega_y(u_i+u_j)}{2\cdot\lambda}\right)}{\frac{n}{2}\cdot(n-1)} \\ &\quad + \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\hat{u}_i + \hat{u}_j)}{\frac{n}{2}\cdot(n-1)}\cdot\dot{m}_{ze}\end{aligned}$$

Somit ergeben sich die letzten drei Zeilen der Jacobi-Matrix:

$$\mathbf{J}_{ze} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\cdot\begin{bmatrix}0 & 0 & \frac{1}{z} & -\frac{v_i+v_j}{2\cdot\lambda} & \frac{u_i+u_j}{2\cdot\lambda}\end{bmatrix}}{\frac{n}{2}\cdot(n-1)}$$

$$\mathbf{J}_{\alpha'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\cdot\begin{bmatrix}0 & 0 & -\frac{(\hat{v}_i+\hat{v}_j)}{z} & \frac{(\hat{v}_i+\hat{v}_j)\cdot(v_i+v_j)}{2\cdot\lambda} & -\frac{(\hat{v}_i+\hat{v}_j)\cdot(u_i+u_j)}{2\cdot\lambda}\end{bmatrix}}{\frac{n}{2}\cdot(n-1)} + \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\hat{v}_i+\hat{v}_j)}{\frac{n}{2}\cdot(n-1)}\cdot\mathbf{J}_{ze}$$

$$\mathbf{J}_{\beta'} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\left\|\mathbf{p}_j - \mathbf{p}_i\right\|_2\cdot\begin{bmatrix}0 & 0 & -\frac{(\hat{u}_i+\hat{u}_j)}{z} & \frac{(\hat{u}_i+\hat{u}_j)\cdot(v_i+v_j)}{2\cdot\lambda} & -\frac{(\hat{u}_i+\hat{u}_j)\cdot(u_i+u_j)}{2\cdot\lambda}\end{bmatrix}}{\frac{n}{2}\cdot(n-1)} + \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}(\hat{u}_i+\hat{u}_j)}{\frac{n}{2}\cdot(n-1)}\cdot\mathbf{J}_{ze}$$

Die Ausgabe des Reglers ergibt sich dann entsprechend dem Reglergesetz, wobei hier die Inverse der Jacobi-Matrix genutzt werden kann, da die Matrix quadratisch ist:

$$\Delta\mathbf{X} = -\mathbf{K} \cdot \mathbf{J}^{-1} \cdot \Delta\mathbf{m}_e$$

Dabei muss beachtet werden, dass durch geeignete Verstärkungsfaktoren die unterschiedlichen Größenordnungen der Merkmalsparameter ausgeglichen werden.

Die Performance des Reglers wird im Abschnitt 7.3 im Vergleich zur folgenden Variante getestet.

7.2 Diagonalmatrix

Vereinfacht dargestellt sieht die oben hergeleitete Jacobi-Matrix wie folgt aus:

$$\mathbf{J} = \begin{bmatrix} j_x & 0 & \tilde{j}_{xz} & \tilde{j}_{x\alpha'} & \tilde{j}_{x\beta'} \\ 0 & j_y & \tilde{j}_{yz} & \tilde{j}_{y\alpha'} & \tilde{j}_{y\beta'} \\ 0 & 0 & j_z & \tilde{j}_{z\alpha'} & \tilde{j}_{z\beta'} \\ 0 & 0 & \tilde{j}_{\alpha' z} & j_{\alpha'} & \tilde{j}_{\alpha'\beta'} \\ 0 & 0 & \tilde{j}_{\beta' z} & \tilde{j}_{\beta'\alpha'} & j_{\beta'} \end{bmatrix}$$

Dabei geben die mit ˜ gekennzeichneten Elemente die unerwünschten Abhängigkeiten zwischen Merkmalsparametern und Bewegungen der Kamera an. Ideal ist also eine Diagonalmatrix.

Die Merkmalsparameter wurden so gewählt, dass sie besonders sensitiv für eine Bewegungsrichtung sind. Die meisten Nicht-Diagonal-Elemente sind also vernachlässigbar. Die Abhängigkeiten von m_x und m_y von den entsprechenden Rotationen ist zwar signifikant, aber wie in Abschnitt 6.2 führt die Vernachlässigung dazu, dass die Merkmale besser im Bild gehalten werden.

In der zweiten Reglervariante wird somit angenommen, dass es eine reine Eins-Zu-Eins-Beziehung zwischen den Merkmalsparametern und den Bewegungsrichtungen der Kamera gibt. Die Inverse der Jacobi-Matrix $\mathbf{J}^{-1}$ wird somit durch eine konstante Diagonalmatrix

$$\mathbf{D} = \begin{bmatrix} k_x & 0 & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 & 0 \\ 0 & 0 & k_{ze} \text{ bzw. } k_{zs} & 0 & 0 \\ 0 & 0 & 0 & k_{\alpha'} & 0 \\ 0 & 0 & 0 & 0 & k_{\beta'} \end{bmatrix}$$

ersetzt. Dabei gleichen die verschiedenen Konstanten die unterschiedlichen Größenordnungen der Merkmalsparameter aus. Das Reglergesetz lautet dann wie folgt:

$$\Delta\mathbf{X} = -K \cdot \mathbf{D} \cdot \Delta\mathbf{m}_e$$

Der zusätzliche Vorteil von diesem Reglergesetz ist, dass auch die alternative Variante des Z-Merkmalsparameters genutzt werden kann. Das Reglergesetz lautet dann:

$$\Delta\mathbf{X} = -K \cdot \mathbf{D} \cdot \Delta\mathbf{m}_s$$

Mit diesem Reglergesetz kann der Roboterarm, wenn er auf die Freiheitsgrade X, Y, Z und γ beschränkt wird, mit nur einem einzigen Merkmal ausgeregelt werden. Dies ist möglich, da neben der Position auch die Skalierung und Orientierung der Merkmals genutzt wird. Es ist zwar nicht sinnvoll, nur ein Merkmal zu verwenden, wenn im Bild sehr viel mehr detektiert werden, es zeigt aber, dass der Regler im gesamten Arbeitsbereich um die Zielposition herum funktioniert, in dem noch zumindest ein einziges Merkmal wiedergefunden werden kann.

7.3 Simulatorische Verifikation

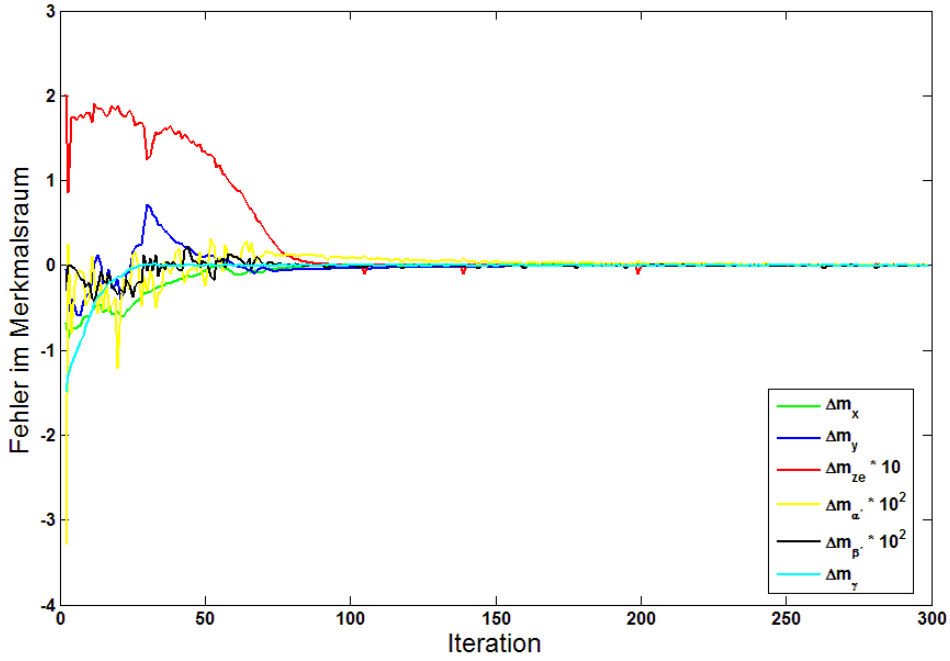
In den vorherigen Abschnitten wurden vier Reglervarianten vorgestellt. Zum einen die Versionen mit der analytischen Jacobi-Matrix, für die zwei verschiedene Methoden der Z-Schätzung verwendet werden können, und zum anderen die Varianten mit der Diagonalmatrix, die mit den beiden unterschiedlichen Merkmalsparametern m_{ze} oder m_{zs} genutzt werden können. Diese werden im Folgenden in der virtuellen Realität getestet.

Dabei zeigen alle Regler ein deutlich besseres Verhalten als der adaptive Regler. So lange sich der Roboterarm nicht zu dicht am Rand seines Arbeitsraumes in der Start- und Ziellage befindet, dass die Abweichungen von der optimalen Trajektorie den Roboterarm aus seinem Arbeitsraum herausführen, und noch genügend Merkmale vom SIFT-Algorithmus detektiert werden können, weisen die Regler eine nahezu hundertprozentige Erfolgsquote auf.

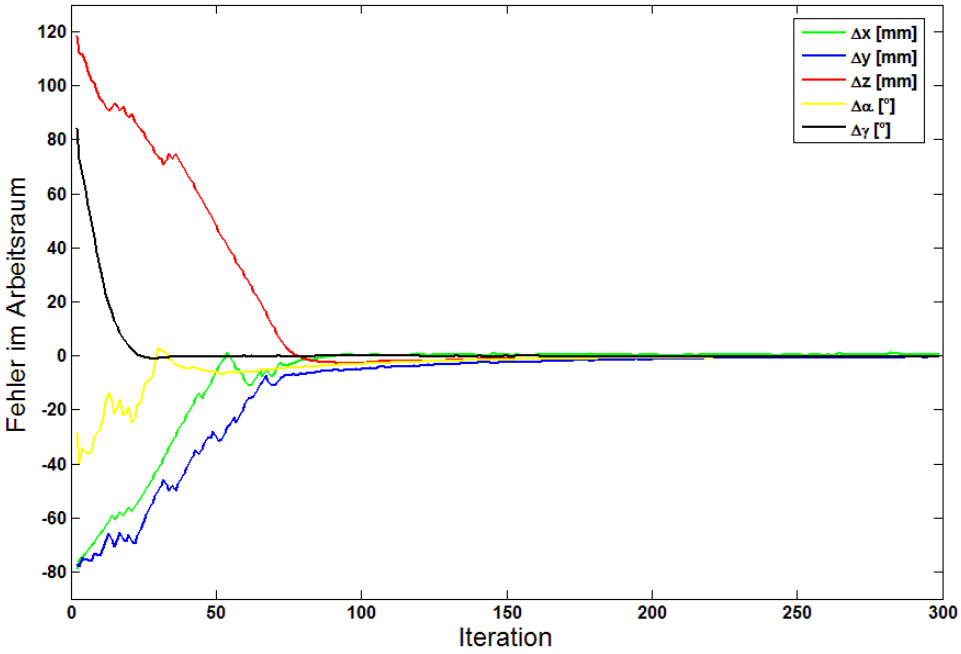
Die Unterschiede der einzelnen Varianten werden an Hand eines Testlaufs in der Simulation demonstriert. Dabei liegt das Zielobjekt flach auf dem Tisch und das Zielbild entspricht in etwa Abbildung 4.1 auf Seite 34. Der Roboterarm wird mit einer Abweichung von $\Delta x = -80$ mm, $\Delta y = -80$ mm, $\Delta z = 120$ mm, $\Delta\alpha = -28,65°$ und $\Delta\gamma = 90°$ von der Ziellage gestartet. Auf Grund des beschränkten Arbeitsraums und der Randbedingung, das Objekt im Blickfeld der Kamera zu halten, ist eine größere Auslenkung nicht möglich. Diese Situation ist also eine der Schwierigsten, die bei diesem Roboterarm auftreten kann.

Abbildung 7.1 bis 7.4 zeigen die Entwicklung des Fehlers im Merkmals- und Arbeitsraum für die verschiedenen Reglervarianten. Es zeigt sich, dass beide Fehler bei allen Reglern gegen einen ähnlich kleinen Restfehler konvergieren, der durch das Rauschen der Merkmale im Bild verursacht wird. Der Betrag des Restfehlers im Arbeitsraum beträgt in den Translationsrichtungen weniger als $0,6$ mm und bei den Rotationen unter $0,3°$. Dieser Grad der Präzision ist mehr als ausreichend für Greif- und Handhabungsaufgaben in der Servicerobotik und ermöglicht sogar die genaue Positionierung von Werkzeugen und Objekten in industriellen Anwendungen.

Durch die Abhängigkeit der Merkmalsparameter von mehreren Bewegungenrichtungen ist bei allen Testläufen zu erkennen, dass der Fehler in Arbeitsraum nicht monoton abfällt. Da die Varianten mit Jacobi-Matrix diese Abhängigkeit berücksichtigen, fällt dieser Effekt hier geringer aus. Der anfängliche deutliche Anstieg des Fehlers in einzelnen Dimensionen bei den anderen Varianten des Reglers kann dazu führen, dass der Roboterarm den Arbeitsbereich verlässt. Deshalb muss hier der Sicherheitsabstand zum Rand des Arbeitsraums größer gewählt werden. Außerdem kann das Überschwingen über die Zielposition zu

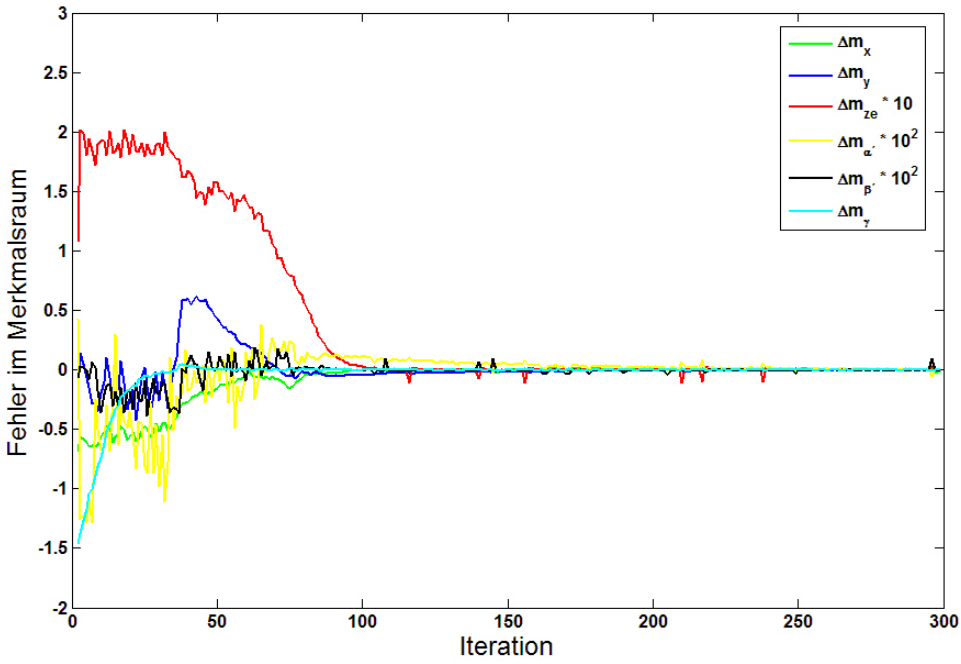


(a) Merkmalsraum

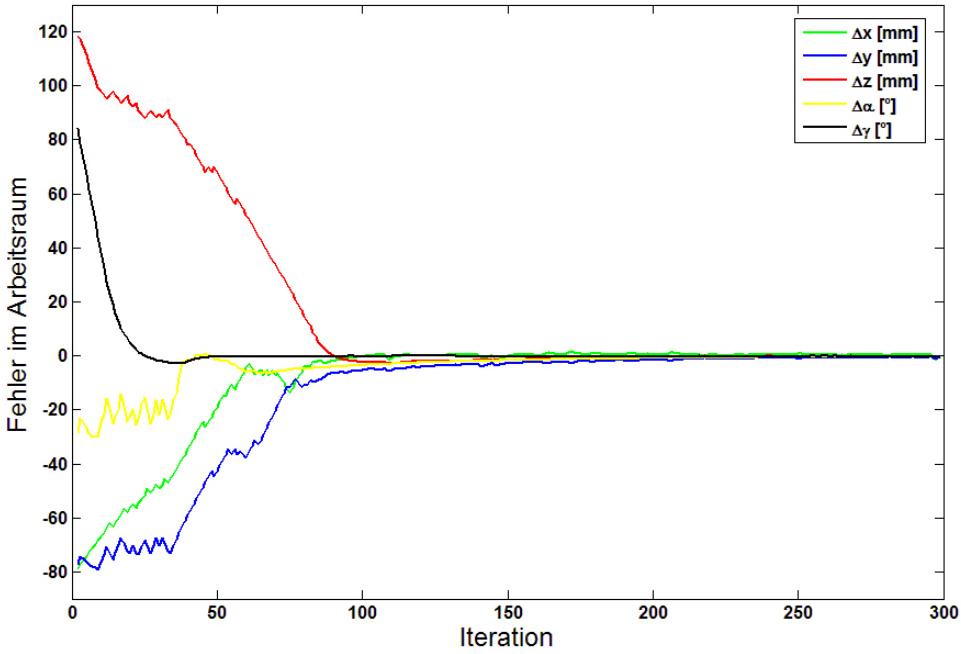


(b) Arbeitsraum

Abb. 7.1: Testlauf des Reglers mit analytischer Jacobi-Matrix und Z-Schätzung über die Abstände der Merkmale

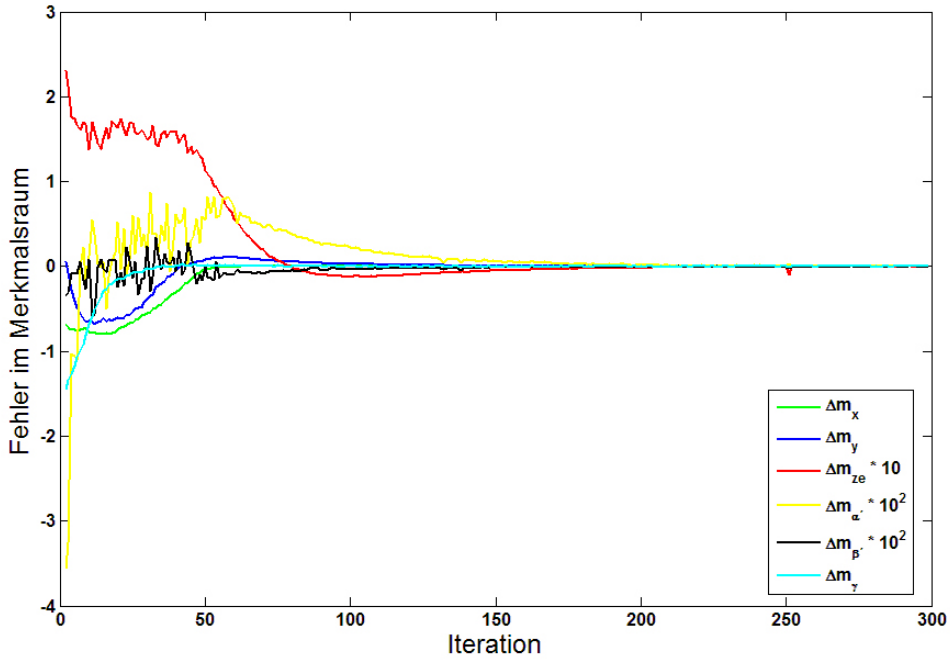


(a) Merkmalsraum

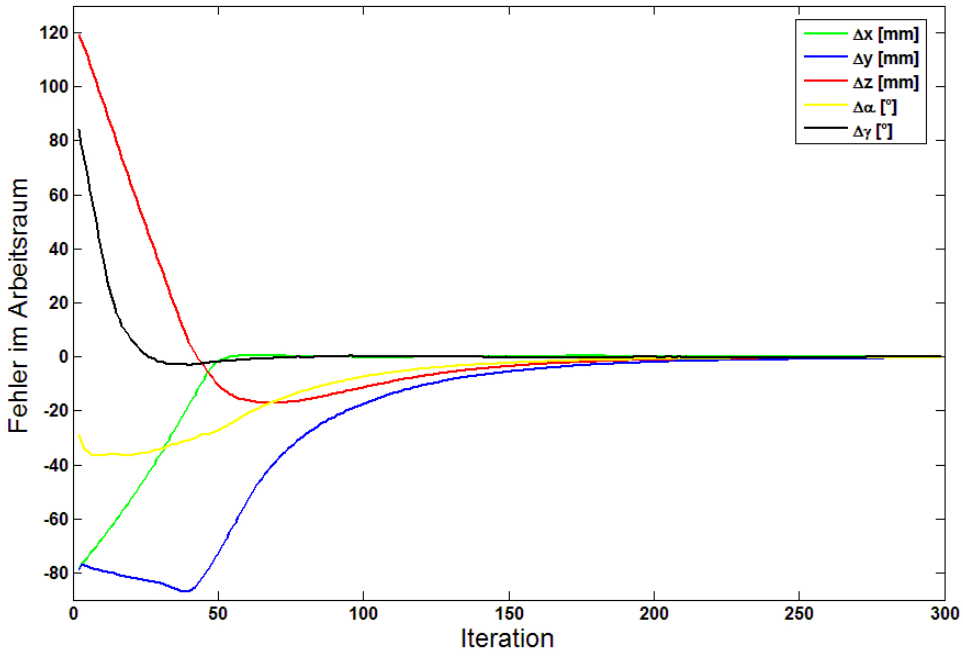


(b) Arbeitsraum

Abb. 7.2: Testlauf des Reglers mit analytischer Jacobi-Matrix und Z-Schätzung über die Skalierung der Merkmale

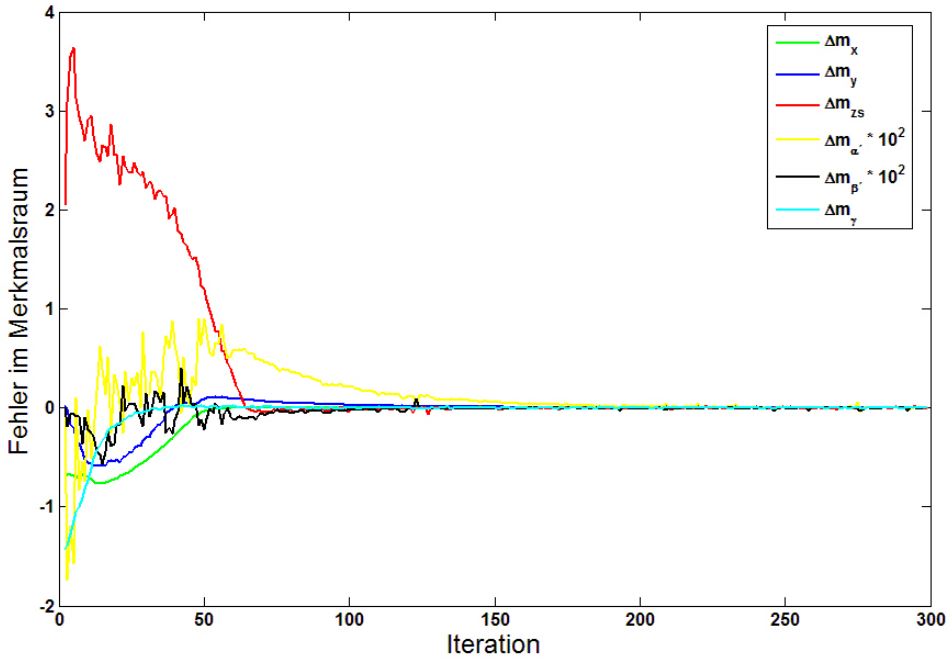


(a) Merkmalsraum

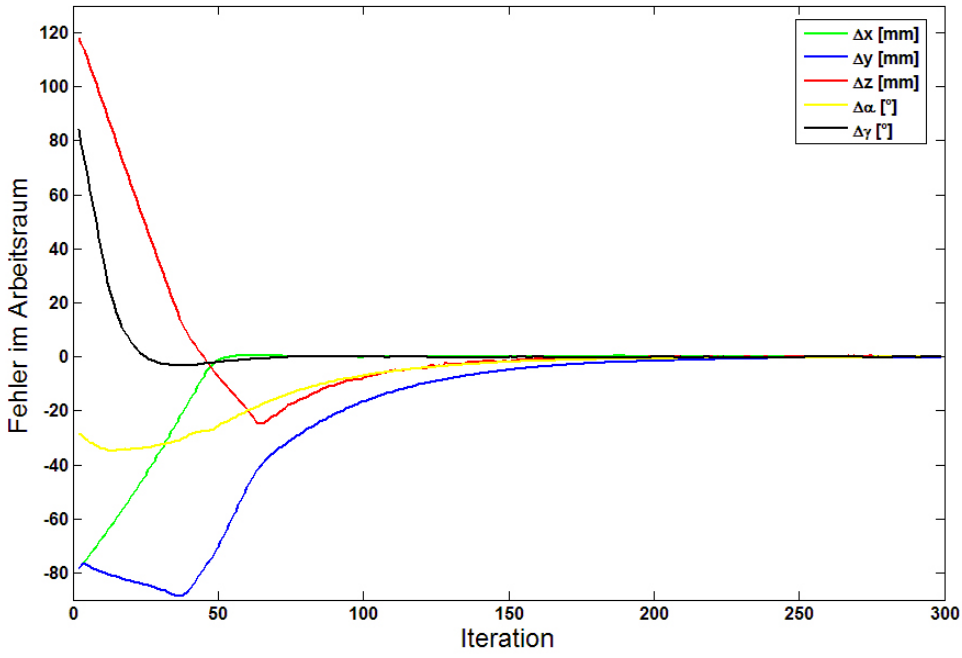


(b) Arbeitsraum

Abb. 7.3: Testlauf des Reglers mit Diagonalmatrix und $\mathbf{m}_e$



(a) Merkmalsraum



(b) Arbeitsraum

Abb. 7.4: Testlauf des Reglers mit Diagonalmatrix und $\mathbf{m}_s$

einer Kollision mit dem Objekt führen. Durch das Hinzufügen von Zwischenbilder (siehe Kapitel 8) können diese Probleme kompensiert werden, indem durch die Lagen, in denen die Zwischenbilder aufgenommen wurden, eine bestimmte Trajektorie vorgegeben wird.

Beim Vergleich der Verläufe des Fehlers im Arbeitsraum fällt besonders ins Auge, dass die Graphen in Abbidlung 7.3 und 7.4 deutlich glatter sind als die anderen beiden. Dies liegt darin begründet, dass bei der Variante mit der Diagonalmatrix jede Bewegungsrichtung nur an einem einzelnen Merkmalsparameter gekoppelt ist. Dieser Vorteil ist bei der hier genutzten Look-Then-Move Struktur nicht so entscheidend, sorgt aber bei der Verwendung einer differentiellen Kinematik in einer Look-And-Move Sturktur für ein ansprechenderes Ergebnis.

Weiter Experimente haben gezeigt, dass die Regler mit Diagonalmatrix weniger empfindlich für Merkmalsrauschen und gelegentliche falsche Zuordnungen von Merkmalen sind. Außerdem wurde das Konvergenzverhalten der Regler für den Fall untersucht, dass die Ebene, die von den Merkmalen aufgespannt wird, und die Bildebene in der Ziellage nicht parallel sind. Für die Berechnung der Jacobi-Matrix wurde angenommen, dass alle Merkmale die gleiche Tiefe z haben. Somit versagen die Regler mit Jacobi-Matrix bei einem Neigungswinkel von mehr als 30°. Die Regler mit Diagonalmatrix sind hingegen unabhängig von der Z-Schätzung und konvergieren erfolgreich gegen die Ziellage auch im nicht parallelen Fall. Diese Eigenschaft ist entscheidend für die Anwendung des Reglers mit Zwischenbildern im nächsten Kapitel, da insbesondere bei den Zwischenbildern die Parallelität der Ebenen nicht gegeben ist.

Für die beiden Möglichkeiten der Z-Schätzung beziehungsweise die beiden Varianten der Merkmalsparameter konnten in den Experimenten keine signifikanten Unterschiede festgestellt werden.

Die experimentellen Ergebnisse zeigen, dass es möglich ist, einen Roboterarm in fünf Freiheitsgraden mit einer monokularen Kamera unter Annahme einer Eins-Zu-Eins Beziehung zwischen Merkmalsparametern und Bewegungsrichtungen zu regeln. Dabei erreicht der Ansatz die gleiche Positionierungsgenauigkeit wie die Regler mit der exakten, analytischen Jacobi-Matirx, und ist dieser Methode bezüglich Robustheit und Gleichmäßigkeit der Bewegung im Arbeitsraum sogar überlegen.

8 Zwischenbilder

Der im vorherigen Kapitel vorgestellt Regler ist fähig, mit den in der Ziellage ermittelten Sollmerkmalen die Kamera aus jeder Position und Orientierung wieder in die gleiche relative Lage zum Objekt auszuregeln, sofern in jedem Kamerabild genügend Merkmale erkannt werden können. Allerdings müssen bei Hol- und Bringdiensten dreidimensionale Objekte gegriffen werden und es kann nicht garantiert werden, aus welcher Richtung die Kamera das Objekt beim Start wahrnimmt. Soll zum Beispiel ein quaderförmiges Objekt, wie in Abbildung 8.1 (a) zu sehen, vom Roboterarm von oben gegriffen werden, dann ist im Zielbild nur die obere Seite zu sehen (b) und somit stammen alle Sollmerkmale von dessen Textur. Befindet sich die Startposition für die Regelung allerdings seitlich vom Objekt, so ist die obere Seite im aktuellen Kamerabild nicht wahrzunehmen (c). Somit kann keines der Sollmerkmale erkannt werden und es ist keine Regelung möglich.

Dieses Problem wird dadurch behoben, dass zwischen der Start- und der Ziellage ein weiteres Zwischenbild aufgenommen wurde, in dem beide Seiten von dem Objekt zu erkennen sind. Der Regler nutzt zunächst die Merkmale aus dem Zwischenbild als Sollmerkmale um näher zum Ziel zu kommen. Wenn der Regler die Zwischenposition erreicht hat, stehen dem Regler wieder die Merkmale aus dem Zielbild zur Verfügung und er kann den Roboterarm in die Ziellage regeln.

Im Allgemeinen können beliebig viele Zwischenbilder vor dem Zielbild genutzt werden, zum Beispiel wenn sich der Roboterarm einmal um das gesamte Objekt herum bewegen muss. Dabei werden die Zwischenbilder in der Lernphase aufgenommen. Um also jede

(a) Quaderförmiges Objekt

(b) Zielbild

(c) Startbild

Abb. 8.1: Ansichten eines Beispielobjektes

beliebige Startlage ausregeln zu können, müssen hinreichend viele Zwischenbilder in der Lernphase generiert werden. Durch die Zwischenbilder wird somit der Raum um das Objekt in Teilräume zerlegt, in denen jeweils auf ein anderes Zwischenbild geregelt wird.

Im folgenden Abschnitt wird zur Auswahl der Zwischenbilder eine für viele Körper geeignete Heuristik vorgestellt. Anschließend wird in Abschnitt 8.2 anhand einer Simulation verifiziert, dass mit der Methode der Zwischenbilder das obige Problem gelöst wird.

8.1 Heuristik

Die Auswahl der Kameralagen, in denen die Zwischenbilder aufgenommen werden, ergibt sich aus den Eigenschaften der Merkmalsextraktion und des Reglers. Dabei wird die Heuristik zunächst für sechs Freiheitsgrade vorgestellt, da für fünf Freiheitsgrade eine Teilmenge dieser Zwischenbilder verwendet werden kann.

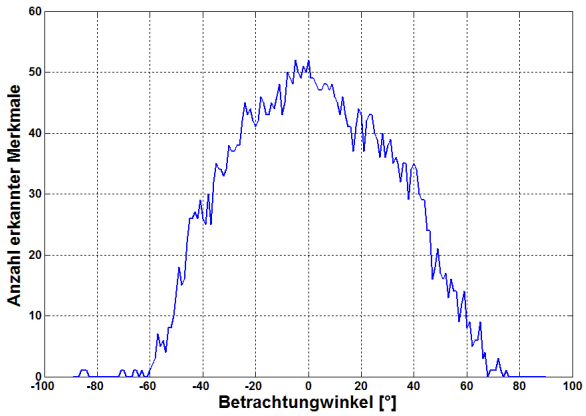
Die erste Eigenschaft der SIFT-Merkmale ist, dass sie skalierungsinvariant sind. Es müssen also keine Zwischenbilder generiert werden, die sich nur in ihrem Abstand zum Objekt unterscheiden. Die Positionen befinden sich also alle auf einer Sphäre mit dem Objekt im Mittelpunkt.

Die zweite Eigenschaft der Merkmale ist die Rotationsinvarianz. Jedes Zwischenbild wird mit der γ-Orientierung der Kamera aufgenommen, die auch in der Zielorientierung benötigt wird.

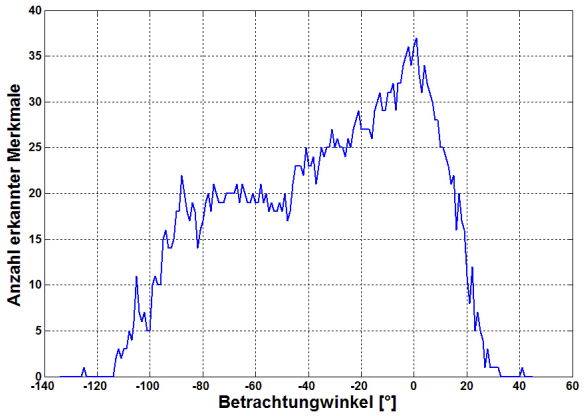
Die letzte entscheidende Eigenschaft ist die Invarianz gegen affine Verzerrungen, zumindest bist zu einem gewissen Maße. Lowe [Low04] konnte experimentell ermitteln, dass bei einer Abweichung von 50° noch etwa 50% der Merkmale wiedergefunden werden können. Dabei bezieht sich die Abweichung auf den Winkel zwischen der Bildebene und der Ebene, in der sich die Merkmale befinden. Als Referenz wird das Bild bei 0° Abweichung verwendet, also ohne jegliche Verzerrung. Abbildung 8.2 (a) zeigt die Anzahl der detektierten Merkmale in Abbhängigkeit von dem Winkel zwischen den beiden Ebenen. Es ist zu erkennen, dass unter den hier gegebenen Bedingungen das Ergebnis von Lowe in etwa bestätigt werden kann, wobei die 50% Grenze hier bei ca. 45° liegt.

Bei der Verwendung von Zwischenbilder ist es unvermeidlich, dass einige der Zwischenbilder eine deutliche Verzerrung aufweisen. Zum Beispiel bei einem quaderförmigen Objekt weisen im schlimmsten Fall alle Flächen, die im Bild zu sehen sind, einen Winkel von 45° zur Bildebene auf. Abbildung 8.2 (b) zeigt das entsprechende Ergebnis für den Fall, dass das Referenzbild mit einer Verzerrung von 45° aufgenommen wurde, wobei die Winkel relativ zur Verzerrung des Referenzbildes angegeben sind. Dies zeigt, dass in diesem Fall relativ zur Referenz nur noch eine Abweichung von ca. 20° in eine Richtung möglich ist, bis nicht mehr genügend Merkmale für die Regelung zur Verfügung stehen. Dieser Fall kann für viele alltägliche Objekte als Worst-Case angenommen werden.

Der Richtwert von 20° wird nun für die Bestimmung der Kamerapositionen der Zwischenbilder auf der Sphäre genutzt. In jeder Positionen wird die Kamera so orientiert, dass sie auf die Objektmitte gerichtet ist. So wird das Kamerabild am besten ausgenutzt. Verschiedene Orientierungen der Kamera in einer Position sind unnötig, da bei einer α-Rotation um 20° das Objekt kaum noch im Blickfeld der Kamera ist. Außerdem ergeben sich durch die verschiedenen Kamerapositionen unterschiedliche Orientierungen. Die Positionen der



(a) Referenz ohne Verzerrung



(b) Referenz mit 45° Verzerrung

Abb. 8.2: Anzahl der erkannten Merkmale in Abhängigkeit von der affinen Verzerrung

Kamera werden auf den Schnittpunkten von gedachten Längen- und Breitenkreisen auf der Sphäre platziert. Dabei wird der „Nordpol“ so gewählt, dass er auf der Geraden durch den Mittelpunkt der Sphäre und der Zielposition liegt. Sollte also der Zielpunkt auf der Sphäre liegen, dann bildet dieser den „Nordpol“.

Somit werden für ein Objekt 128 Zwischenbilder benötigt. Geht man davon aus, das in einem Bild im Schnitt 50 Sollmerkmale abgespeichert werden müssen, wobei jedes Merkmal einen Speicherbedarf von 132 Byte hat, dann benötigt ein Objekt ungefähr $0,8$ MB Speicherplatz. Dies ist sehr gering im Vergleich zu Verfahren der erscheinungsbasierten Lageschätzung, die bis zu 200 MB Platz benötigen.

Am Beginn der Regelung muss zunächst des erste Zwischenbild ermittelt werden, auf das geregelt werden soll. Unter Umständen kann dies natürlich schon das Zielbild sein, wenn die Startlage nur eine geringe Abweichung vom Ziel aufweist. Allgemein wird das erste Bild dadurch ermittelt, das für jedes Zwischenbild der prozentuale Anteil der Merkmale bestimmt wird, die im aktuellen Kamerabild wiedergefunden werden. Das Zwischenbild, für das der Anteil am höchsten ist, ist wahrscheinlich das nächstgelegenste, und wird somit als erstes Zwischenziel verwendet.

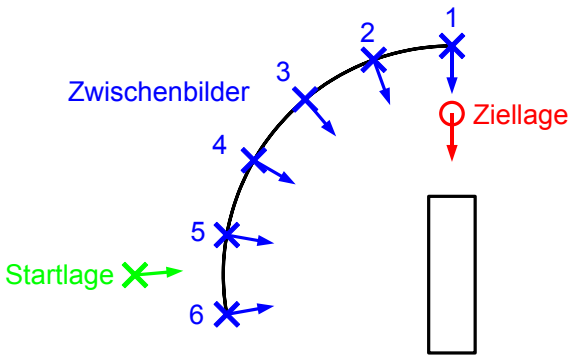


Abb. 8.3: Zweidimensionale, schematische Darstellung der Lage der Zwischenbilder

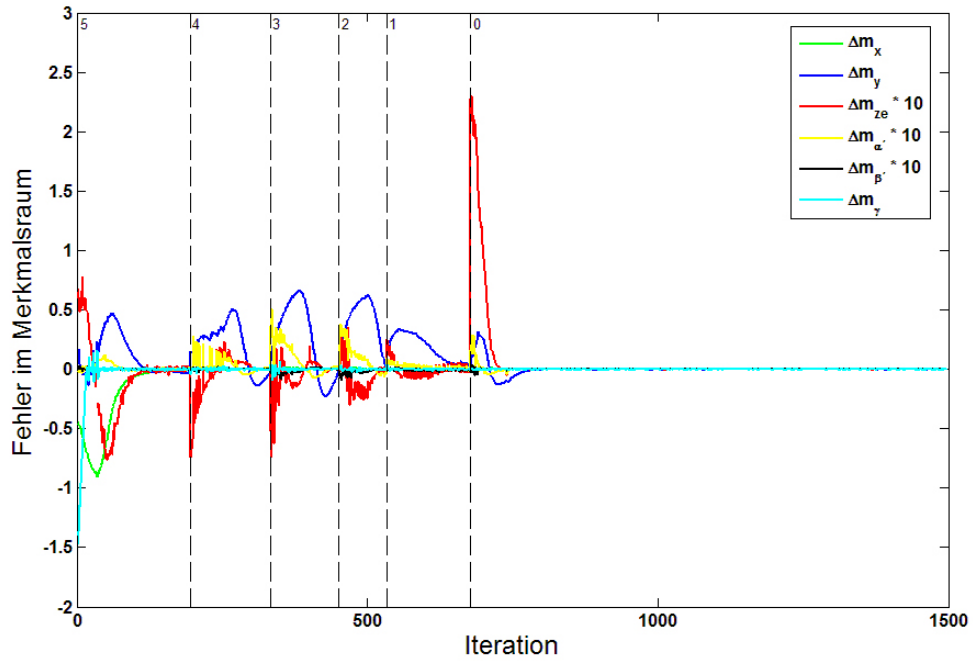
Als nächstes muss entschieden werden, wann auf welches Zwischenbild gewechselt wird. Als Zeitpunkt wird hier das ungefähre Erreichen der Zwischenbildposition gewählt, wenn die Werte der Stellgrößen des Reglers unterhalb eines Schwellwertes sinken. Die Wahl des nächsten Zwischenbildes wird statisch festgelegt. Bei der Generierung der Zwischenbilder ist bekannt, welches das nächste Zwischenbild auf dem Weg zum Ziel ist. Die Zwischenbilder können somit in einer Baumstruktur angeordnet werden, mit dem Zielbild als Wurzel, dem „Nordpol“ als dessen Kind (sofern es das nicht selber ist). Jedes weitere Zwischenbild ist das Kind vom nächstnäheren zum Ziel. Beim Erreichen einer Zwischenlage wird also als nächstes das Bild im Elternknoten für die Regelung genutzt.

8.2 Simulatorische Verifikation

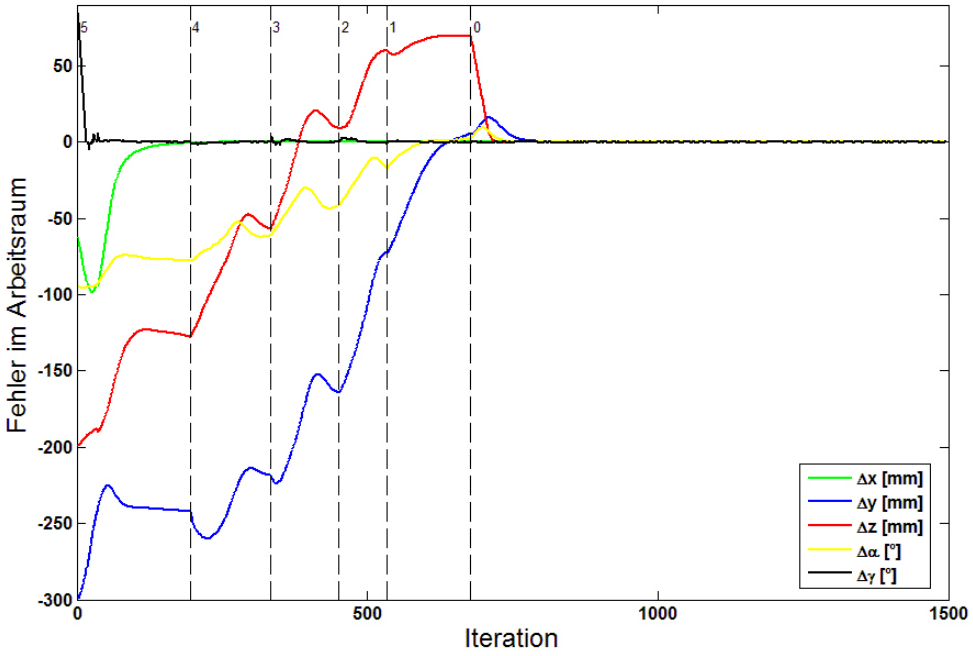
Bei dem in dieser Arbeit verwendete Roboterarm mit fünf Freiheitsgraden können allerdings die Lagen der Zwischenbilder nicht erreichen werden, die eine Rotation der Kamera um die β'-Achse erfordern. Für den Test werden deshalb nur die benötigten Zwischenbilder entlang des 0° Längenkreises aufgenommen und davon ausgegangen, dass das Objekt nicht entlang seiner Hoch- oder Längsachse rotiert ist. Dann wäre die Ziellage für den Roboterarm nicht mehr erreichbar.

Die Regelsituation ist in Abbildung 8.3 schematisch dargestellt. Die Zielposition befindet sich oberhalb von einem quaderförmigen Objekt mit Blickrichtung zum Objektmittelpunkt (roter Kreis). Sechs Zwischenbilder (blaue Kreuze) werden wie oben beschrieben entlang einer Kreisbahn aufgenommen, wobei die Zahlen die statische Reihenfolge der Zwischenbilder in absteigender Folge angeben. Für jedes Zwischenbild wird das in Kapitel 4 vorgestellte Verfahren zur automatischen Auswahl robuster Merkmale verwendet. Die Abweichung beim Start von der Ziellage beträgt $\Delta x = -60$ mm, $\Delta y = -300$ mm, $\Delta z = -200$ mm, $\Delta\alpha = -95°$ und $\Delta\gamma = 90°$ (grünes Kreuz). Somit ist beim Start die Oberseite vom Objekt nicht im Blickfeld der Kamera und keines der Merkmale aus dem Zielbild kann detektiert werden.

Der Arbeitsbereich des Roboterarms ist allerdings so eingeschränkt, dass damit keine Auslenkung möglich ist, mit der man die Fähigkeiten dieser Methode demonstrieren kann. Deshalb wird für die Demonstration in der Simulation der Roboterarm entfernt und eine „schwebende“ Kamera verwendet.

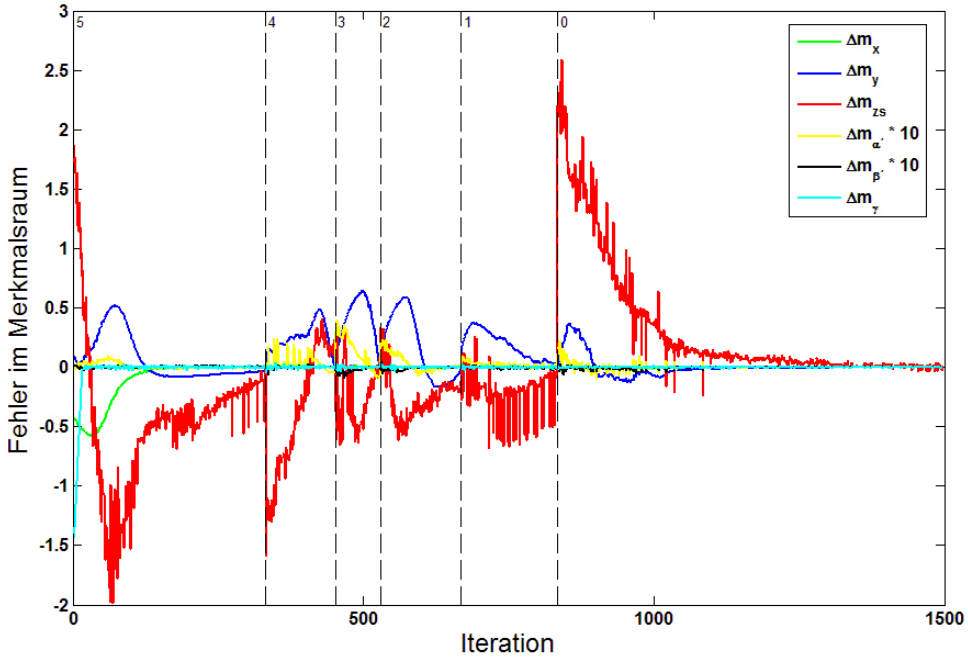


(a) Merkmalsraum

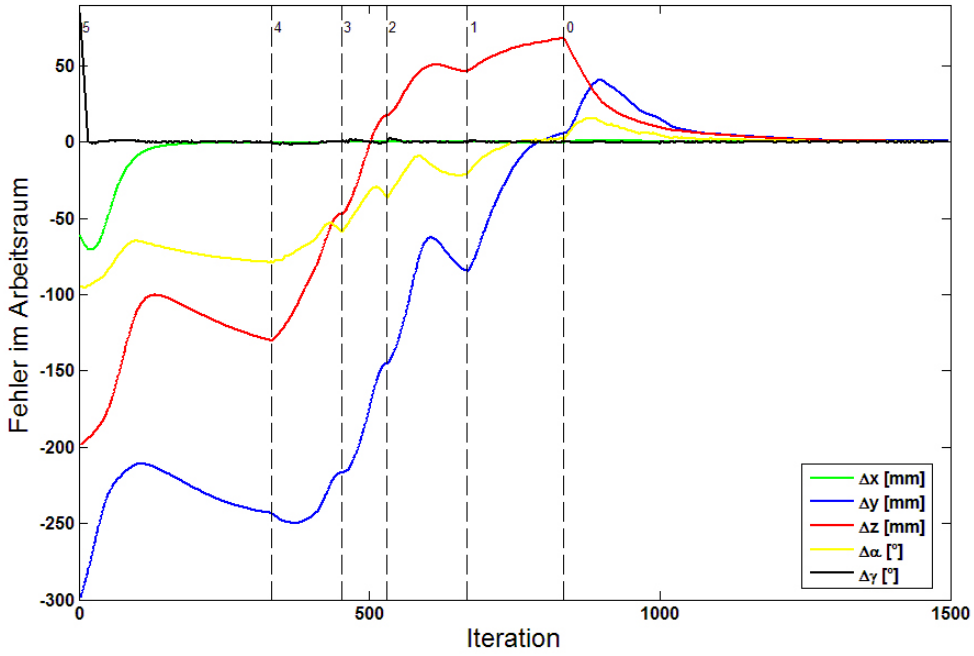


(b) Arbeitsraum

Abb. 8.4: Regelverlauf mit Zwischenbildern und Entfernungvariante der Merkmalsparameter



(a) Merkmalsraum



(b) Arbeitsraum

Abb. 8.5: Regelverlauf mit Zwischenbildern und Skalierungsvariante der Merkmalsparameter

Die Abbildungen 8.4 und 8.5 zeigen den Verlauf des Fehlers im Bild- und im Arbeitsraum bei diesem Test für den Regler mit der Diagonalmatrix jeweils für die beiden Varianten des Merkmalsparameters. Dabei bezieht sich der Fehler im Bildraum auf das nächste Zwischenbild, während der Fehler im Arbeitsraum die Abweichung von der Ziellage angibt. Die senkrechten, gestrichelten Linien befinden sich an den Regelschritten, in denen auf das nächste Zwischenbild gewechselt wird, dessen Nummer aus Abbildung 8.3 oben an der Linie annotiert ist. Die Null steht dabei für das Zielbild.

Das Zwischenbild Nummer 5 wird mit $41,9\%$ wiedergefundener Merkmale als nächstgelegenes Zwischenbild erkannt. Danach werden die Zwischenbilder in der festgelegten Reihenfolge abgefahren, bis das Ziel erreicht ist. Dabei ergibt sich bei jedem Umschalten durch den Wechsel der Sollmerkmale ein Sprung in den Merkmalsparametern. Die Positionierungsgenauigkeit der beiden Regler in der Ziellage entspricht der im vorherigen Kapitel, was zu erwarten war, da das abschließende Ausregeln des Zielbildes der Situation im vorherigen Kapitel entspricht.

9 Experimentelle Ergebnisse am Realen Roboterarm

Abschließend werden im Folgenden die in der Simulation erarbeiteten Ergebnisse in Experimenten am realen Roboterarm verifiziert. Dazu wird der Roboterarm „Katana HD5M“ der Firma „Neuronics“ verwendet, der in Abbildung 9.1 zu sehen ist. Die Bilder für die Reglung werden durch die am Roboterarm montierte Kamera vom Typ „DFK 21F04“ der Firma „The Imaging Source“ aufgenommen. Bei der gewählten Auflösung 320×240 Pixeln ermöglicht der Rechenaufwand für die Merkmalsextraktion und -zuordnung eine Regelfrequenz von ungefähr 7 Hz. Dies ist prinzipiell ausreichend für eine kontinuierliche Bewegung in einer Look-And-Move Struktur. Allerdings hat die aktuelle Implementierung der differentiellen Kinematik des verwendeten Roboterarm das Problem einer Kommunikationsverzögerung zwischen dem Host und dem Mikrocontroller im Roboterarm von über 300 ms. Deshalb muss hier auch eine Look-Then-Move Struktur verwendet werden, die eine Regelfrequenz von ungefähr 3 Hz erreicht.

Für die Experimente kann die für die Simulation entwickelte Implementierung durch weni-

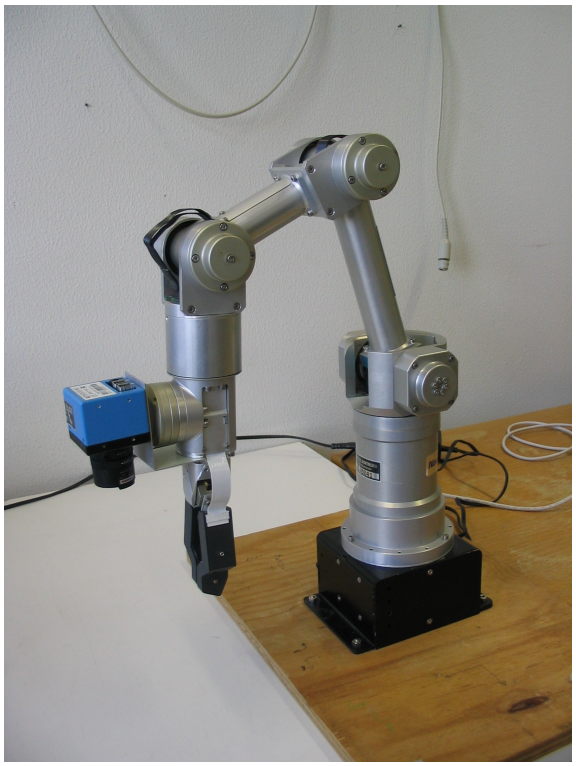


Abb. 9.1: Roboterarm „Katana HD5M“ mit Kamera

Startlage	Δx [mm]	Δy [mm]	Δz [mm]	$\Delta\alpha$ [°]	$\Delta\gamma$ [°]
1	40	-30	40	20	110
2	-50	-60	50	23	-60

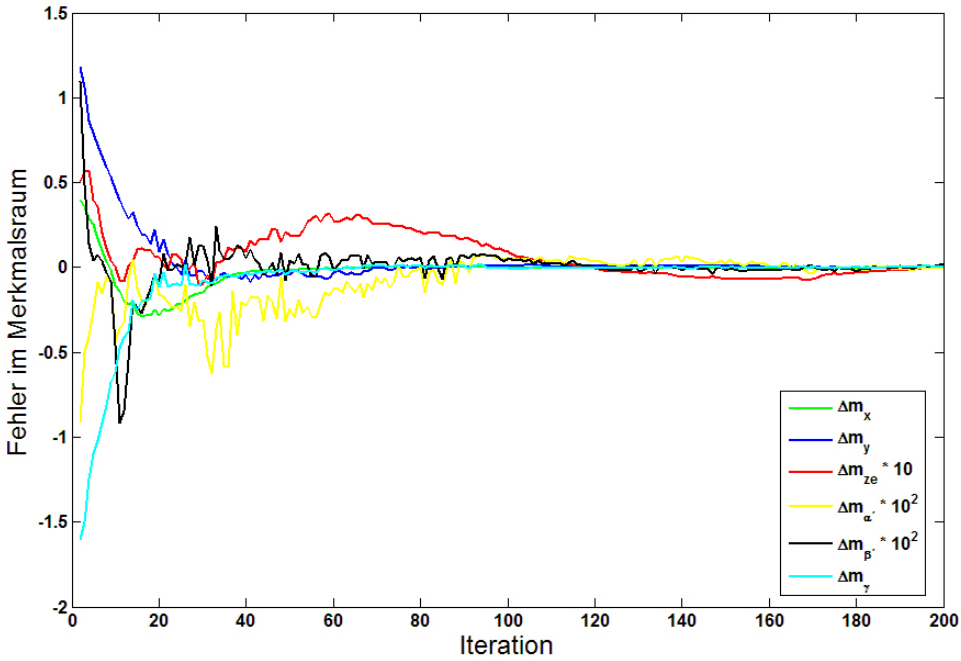
Tab. 9.1: Abweichungen der Startlagen vom Ziel

ge Änderungen an den Schnittstellen beibehalten werden. Dabei wird mit Hilfe der „Image Acquisition Toolbox“ das per „FireWire“ übertragene Bild der Kamera in Matlab zur Verfügung gestellt. Die Ansteuerung des Roboterarms erfolgt über die von „Neuronics“ bereitgestellte Bibliothek, die durch eine „dll“ mit „mexFunction“ in Matlab verwendet werden kann. Die entsprechenden Steuerbefehle werden über eine „USB-To-Serial“ Verbindung an den Roboterarm übertragen.

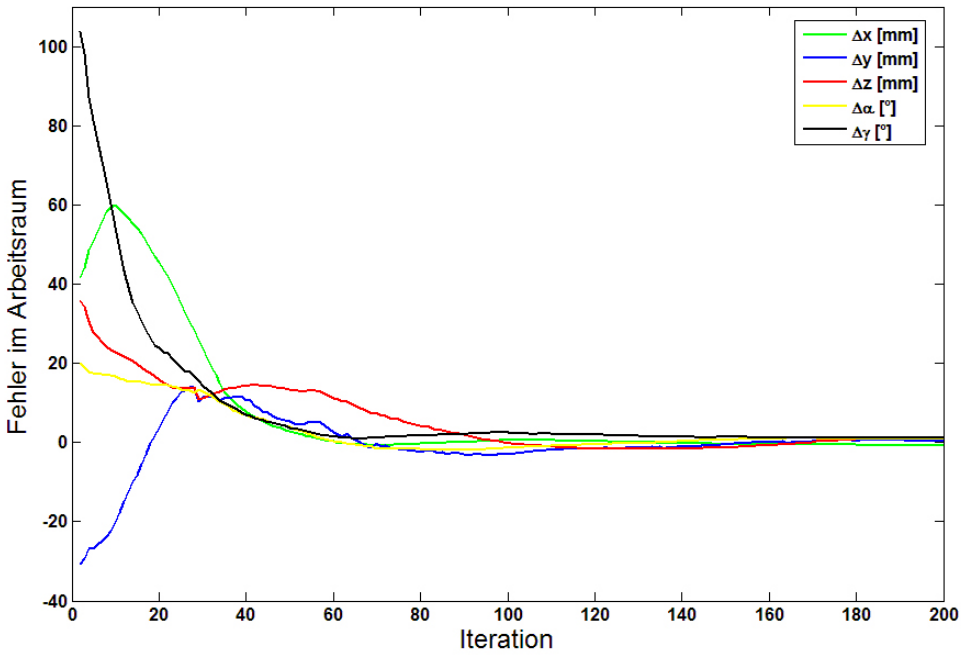
Bei dem Testaufbau in der Realität wird der Aufbau in der Simulation, der in Abschnitt 7.3 auf Seite 71 erläutert wurde, möglichst genau nachgebildet. Die Experimente werden für den Regler mit der Diagonalmatrix und den beiden Varianten des Merkmalsparameters durchgeführt. Für beide wird der Regelungsverlauf exemplarisch an zwei Startlagen demonstriert, die in Tabelle 9.1 relativ zu Ziellage angegeben sind. Zum Schutz des Roboterarm vor Bewegungen, die zu Hardwareschäden führen können, ist der Arbeitsbereich im Vergleich zur Simulation noch weiter eingeschränkt. Deshalb wurden die Abweichungen zwischen Start und Ziel im realen Experiment deutlich kleiner gewählt.

Die Abbildungen 9.2 bis 9.5 zeigen die Entwicklung des Fehler im Merkmals- und Arbeitsraum für die unterschiedlichen Merkmalsparameter und Startlagen. In allen Fällen wird die Abweichung vom Ziel erfolgreich ausgeregelt, mit einem zur Simulation vergleichbaren Regelverlauf und Anzahl von Regelschritten, bis ein gewisser Restfehler erreicht ist. Allerdings ist dieser Restfehler etwas größer im Vergleich zur Simulation. In den Translationsrichtungen liegt der Fehler maximal bei $1mm$ und bei den Rotationen ca. $0,5°$ in α-Richtung und $1,5°$ bei γ. Dies liegt an der geringeren Auflösung der Kamera und Ungenauigkeiten bei der Positionierung des Roboterarms. Außerdem ist die Kamera nicht ganz starr am Roboterarm befestigt und kann durch dessen Bewegungen in Schwingungen versetzt werden. Dies führt zu unscharfen Aufnahmen und ständig wechselnder relativer Lage der Kamera zum TCP.

Auffällig ist der sprunghaft Anstieg Merkmalsparameterfehlers Δm_{zs} in Abbildung 9.5 (a) bei ungefähr 60 Iterationen. Wie der Verlauf im Arbeitsraum in (b) zeigt, wird dieser Anstieg nicht durch den Regler verursacht, da alle Fehler im Arbeitsraum kleiner werden. Der Anstieg wird durch SIFT-Merkmale mit hohen Frequenzanteilen verursacht, die durch das Annähern der Kamera an das Objekt zum ersten Mal im Bild detektiert werden. Durch diese zusätzlichen Informationen wird der Fehler im Arbeitsraum in den folgenden Iterationen deutlich schneller reduziert.

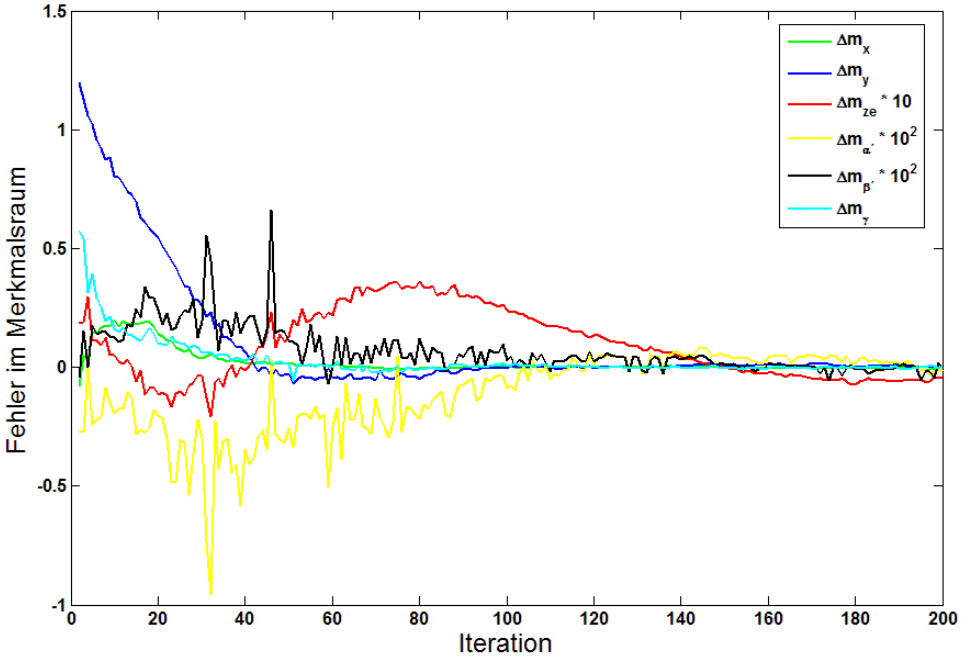


(a) Merkmalsraum

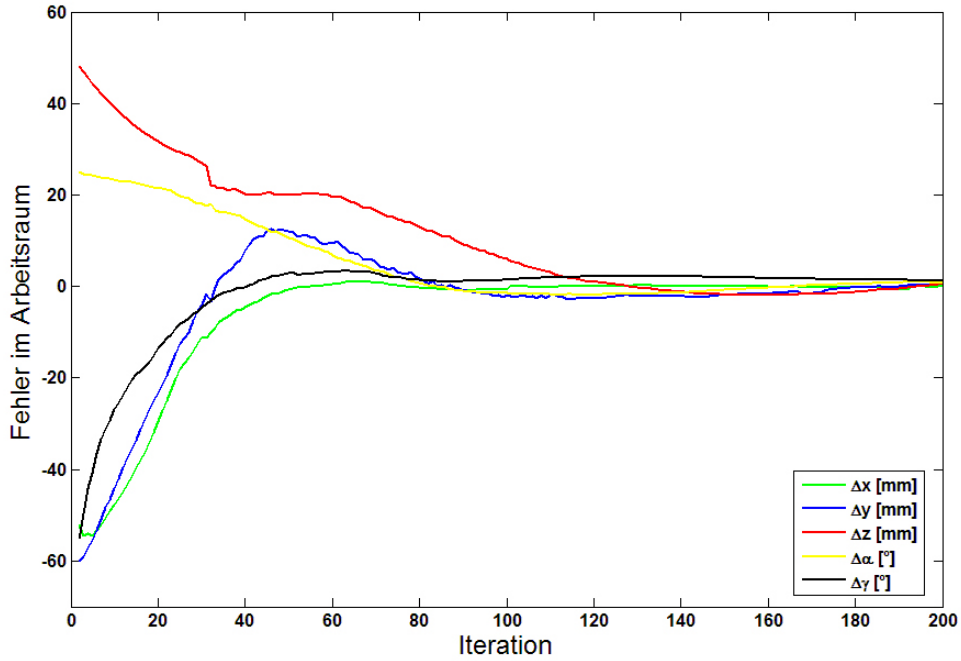


(b) Arbeitsraum

Abb. 9.2: Testlauf des Realen Roboterams von der ersten Startlage mit der Entfernungsvariante

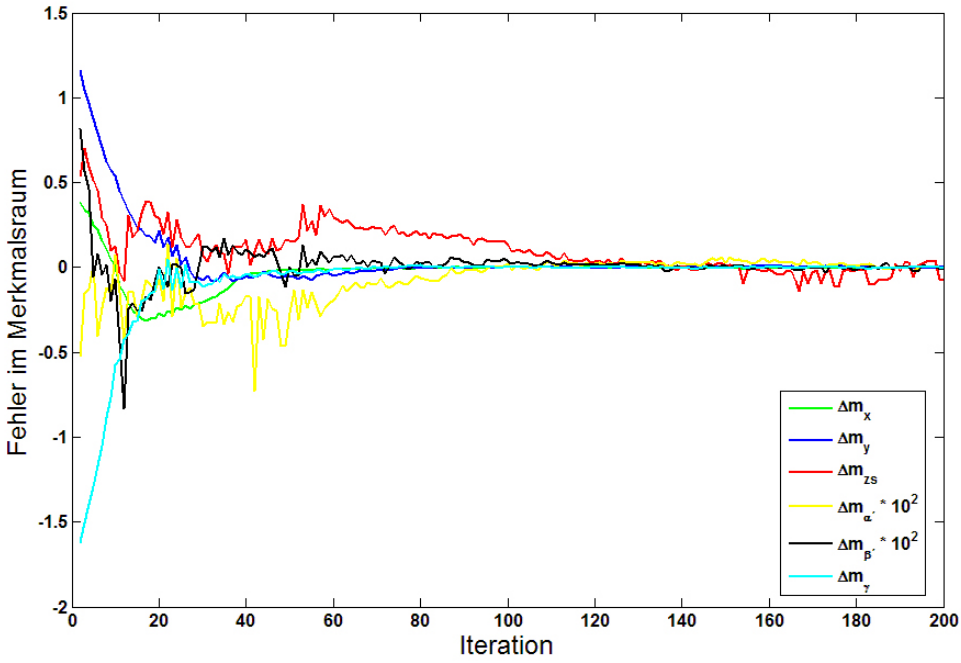


(a) Merkmalsraum

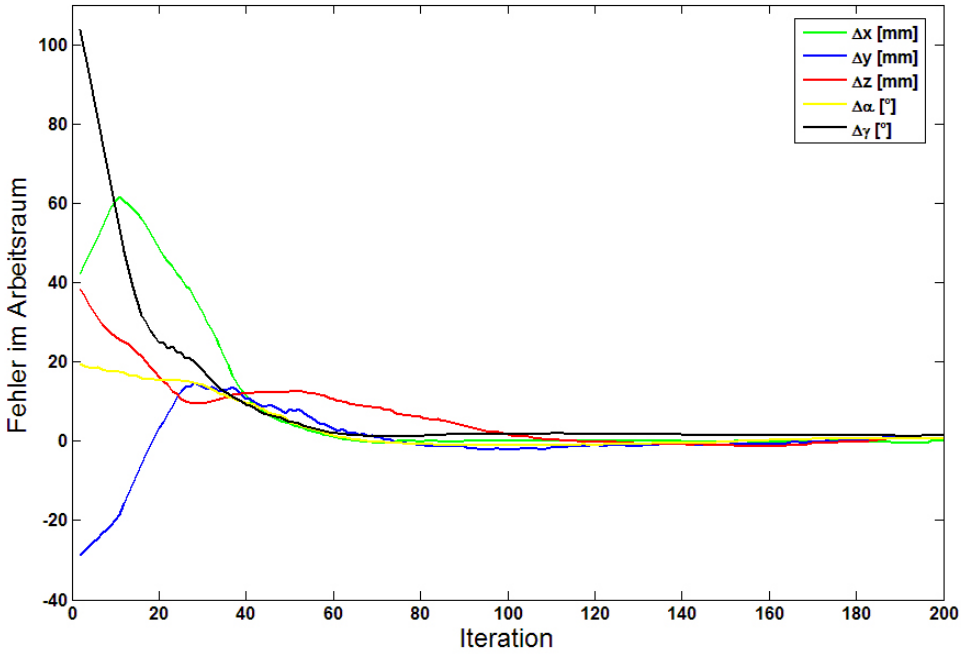


(b) Arbeitsraum

Abb. 9.3: Testlauf des Realen Roboterams von der zweiten Startlage mit der Entfernungsvariante

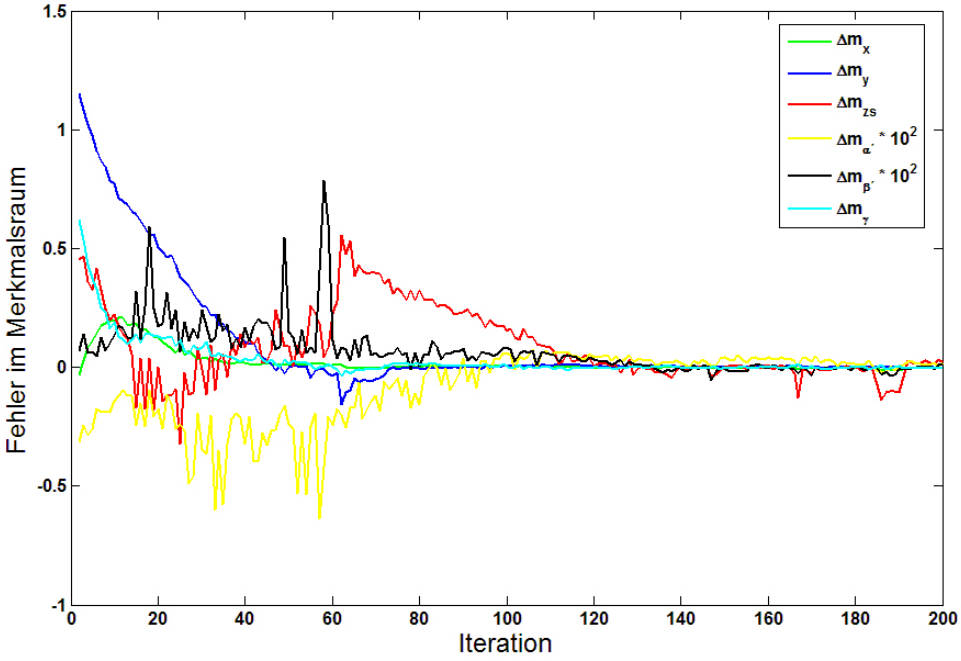


(a) Merkmalsraum

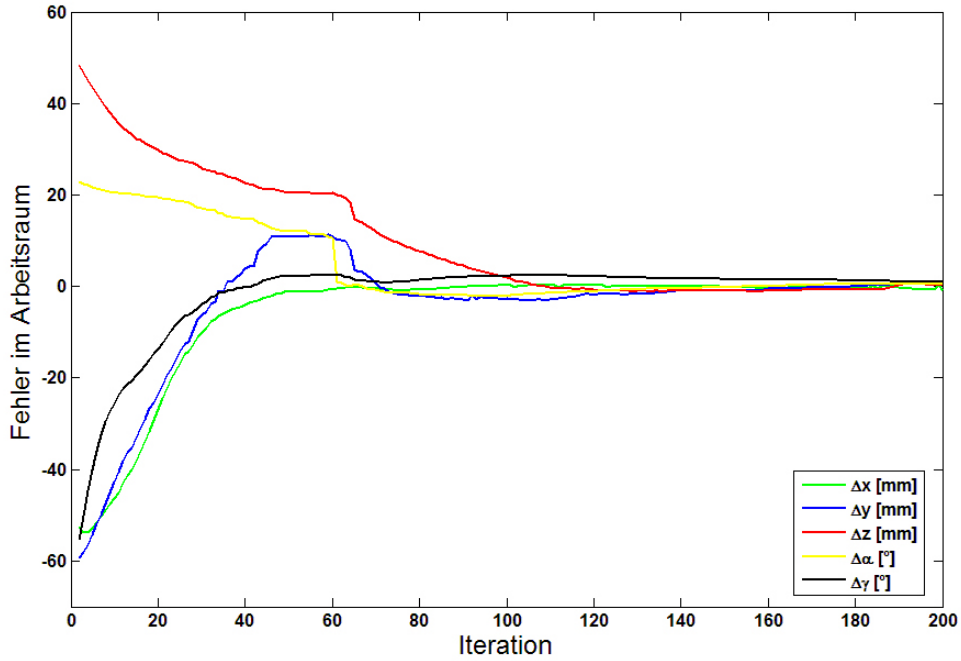


(b) Arbeitsraum

Abb. 9.4: Testlauf des Realen Roboterams von der ersten Startlage mit der Skalierungsvariante



(a) Merkmalsraum



(b) Arbeitsraum

Abb. 9.5: Testlauf des Realen Roboterams von der zweiten Startlage mit der Skalierungsvariante

10 Zusammenfassung und Ausblick

In dieser Arbeit wird gezeigt, dass sich die Scale-Invariant Feature Transformation, die ursprünglich aus dem Forschungsgebiet der Objekterkennung stammt, auch für eine bildbasierte Regelung eignet und somit Referenzobjekte ohne spezielle Makierungen verwendet werden können. Für eine robustere Regelung wird ein automatisches Auswahlverfahren vorgestellt, das aus der Menge der SIFT-Merkmale die beste Teilmenge zur zuverlässigen Lösung des Korrespondenzproblems ermittelt.

Basierend auf Momenten von SIFT-Merkmalen wird ein neues Reglerdesigns vorgestellt, dass die klassischen Probleme der bildbasierten Regelung in der Auge-In-Hand Konfiguration löst. Die Hauptorientierung der SIFT-Merkmale ermöglicht dabei eine direkte Regelung der Kamerarotation um die optische Achse und die vollständige Entkopplung der übrigen Merkmalsparameter von dieser Rotation. Die definierten Merkmalsparameter sind generisch, da sie aus Momenten von Positionen, Orientierungen und Skalierungen über eine Menge von SIFT-Merkmalen berechnet werden. Dadurch ist der Regler unabhängig von der Sichtbarkeit bestimmter Merkmale, was die Robustheit des Reglers erhöht.

Die zu den Merkmalsparametern hergeleitete Jacobi-Matrix beschreibt den Zusammenhang zwischen der Bewegung der Kamera und der Änderung der Merkmalsparameter. Auf Grund der Vielzahl vernachlässigbar kleiner Einträge in der Jacobi-Matrix kann das Reglerdesign vereinfacht werden, indem eine Eins-Zu-Eins Beziehung zwischen den Merkmalsparameter und den Bewegungsrichtungen angenommen wird. Dieser Regler ist der Variante mit der exakten Jacobi-Matrix in Bezug auf Robustheit und Gleichmäßigkeit der Bewegung im Arbeitsraum überlegen.

Der Rechenaufwand für die Extraktion und Zuordnung der SIFT-Merkmale ermöglicht einen echtzeitfähigen, bildbasierten Regler mit einer Regelfrequenz von 7 Hz. Die experimentellen Untersuchungen zeigen, dass der entwickelte Regler eine Positionierungsgenauigkeit im Submillimeterbereich aufweist. Die erreichbare Genauigkeit beim realen Roboterarm ist vielmehr durch die Kinematik des Roboterarms als durch die Präzision der bildbasierten Regelung begrenzt.

Zur Erweiterung des Arbeitsbereiches des Reglers wird basierend auf dem vereinfachten Reglerdesign die Methode der Zwischenbilder vorgestellt. Durch das Aufnehmen zusätzlicher Referenzbilder ist es möglich, den Roboterarm aus einer Startlage ins Ziel zu verfahren, in der er die Sollmerkmale des Zielbildes nicht detektieren kann.

Zukünftige Arbeiten werden sich damit beschäftigen, die Entkopplung weiter zu verbessern. Ziel ist es Merkmalsparameter zu finden, die tatsächlich eine Eins-Zu-Eins Beziehung mit den Bewegungsrichtungen aufweisen. Im Bereich der Zwischenbilder sollte die Möglichkeit untersucht werden, ob von den Zwischenbildern auf einzelne Merkmale abstrahiert werden kann. Damit kann eine Zeit- oder Bahnoptimalität erreichbar werden.

Darüber hinaus sollte das Problem des beschränkten Arbeitsbreiches des Roboterams gelöst werden. Eine Möglichkeit dazu ist, dass die mobile Plattform mit in die Regelung des

Roboterarm einbezogen wird. Bestimmte Translationen des Roboterarms würden nicht durch die Bewegung der Achsen sondern durch die Bewegung der mobilen Plattform also der Basis des Roboterarms durchgeführt werden. Hierdurch erhält der Roboterarm einen größeren effektiven Arbeitsbereich.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[And88] Andersson, R. L.: *A Robot Ping-Pong Player. Experiment in Real-Time Intelligent Control*. MIT Press, Cambridge, MA, 1988.

[BL02] Brown, M. und D.G. Lowe: *Invariant features from interest point groups*. In: *British Machine Vision Conference*, Seiten 656–665, Cardiff, Wales, 2002.

[CH94] Castano, A. und S. A. Hutchinson: *Visual compliance: Task-directed visual servo control*. IEEE Transactions on Robotics and Automation, 10:334–342, Juni 1994.

[CHH96] Corke, P. I., S. A. Hutchinson und G.D. Hager: *A tutorial on visual servo control*. IEEE Transactions on Robotics and Automation, 12(5):651–670, Oktober 1996.

[Cra04] Craig, J. J.: *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 3 Auflage, July 2004.

[CSS91] Chieaverini, S., L. Sciavicco und B. Siciliano: *Control of robotic systems through singularities*. In: *Proc. Int. Workshop on Nonlinear and Adaptive Control: Issues in Robotics (C. C. de Wit, ed.)*. Springer-Verlag, 1991.

[ECR92] Espiau, B., F. Chaumette und P. Rives: *A new approach to visual servoing in robotics*. IEEE Transactions on Robotics and Automation, 8:313–326, Juni 1992.

[FDD94] Fagerer, C., D. Dickmanns und E. D. Dickmanns: *Visual grasping with long delay time of a free floating object in orbit*. Autonomous Robots, 1(1):53–68, 1994.

[Fle87] Fletcher, R.: *Practical methods of optimization*. Wiley-Interscience, New York, NY, USA, 2nd Auflage, 1987.

[FLP01] Faugeras, O., Q.-T. Luong und T. Papadopoulou: *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. MIT Press, Cambridge, MA, USA, 2001.

[FP02] Forsyth, D. A. und J Ponce: *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[HN94] Huang, T. S. und A. N. Netravali: *Motion and structure from feature correspondences: A review*. IEEE Proceeding, 82(2):252–268, 1994.

[Hor86] Horn, B. K.: *Robot Vision*. McGraw-Hill Higher Education, 1986.

[Int94] International Organization for Standardization: *ISO 8373: Manipulating industrial robots – Vocabulary*, 1994.

[Jäg96] Jägersand, M.: *Visual Servoing using Trust Region Methods and Estimation of the Full Coupled.* In: *IASTED Applications of Control and Robotics*, Seiten 105–108, Rochester, NY 14627, 1996. Department of Computer Science, University of Rochester.

[JKCB91] Jang, W., K. Kim, M. Chung und Z. Bien: *Concepts of augmented image space and transformed feature space for efficient visual servoing of an eye-in-hand robot.* Robotica, 9:203–212, 1991.

[KH94] Kumar, R. und A. R. Hanson: *Robust methods for estimating pose and a sensitivity analysis.* CVGIP: Image Understanding, 60(3):313–342, 1994.

[LG92] Lei, M. und B.K. Ghosh: *Visually-Guided Robotic Motion Tracking.* In: *Thirtieth Annual Allerton Conference on Communication, Control and Computing*, Seiten 712–721, Allerton, Illinois, USA, Oktober 1992.

[Lin94] Lindeberg, T.: *Scale-space theory: A basic tool for analysing structures at different scales.* Journal of Applied Statistics, 21(2):224–270, 1994.

[Low04] Lowe, D. G.: *Distinctive Image Features from Scale-Invariant Keypoints.* Int. J. Comput. Vision, 60(2):91–110, 2004.

[Mat05] *Matlab User Guides: External Interfaces*, 2005.

[Matck] Matarić, M. J.: *The Robotics Primer.* The MIT Press, Vorabdruck.

[Mik02] Mikolajczyk, K.: *Detection of local features invariant to affines transformations.* Doktorarbeit, Institut National Polytechnique de Grenoble, France, 2002.

[MP05] Mariottini, G.L. und D. Prattichizzo: *EGT: a Toolbox for Multiple View Geometry and Visual Servoing.* IEEE Robotics and Automation Magazine, 3(12), Dezember 2005.

[MP06] Mariottini, G. L. und D. Prattichizzo: *Epipolar Geometry Toolbox: User Guide*, Januar 2006.

[NK94] Nelson, B. und P. K. Khosla: *Integrating Sensor Placement and Visual Tracking Strategies.* In: *The 3rd International Symposium on Experimental Robotics III*, Seite 169.181, London, UK, 1994. Springer-Verlag.

[PK93] Papanikolopoulos, N. P. und P. K. Khosla: *Adaptive robotic visual tracking: theory and experiments.* IEEE Transactions on Automatic Control, 38(3):429–445, März 1993.

[PKK93] Papanikolopoulos, N.P., P.K. Khosla und T. Kanade: *Visual tracking of a moving target by a camera mounted on a robot:a combination of control and vision.* IEEE Transactions on Robotics and Automation, 9(1):14–35, Februar 1993.

[SBJ90] Skaar, S. B., W. H. Brockman und W. S. Jang: *Three-Dimensional Camera Space Manipulation.* International Journal of Robotics Research, 9(4):22–39, 1990.

[SJS04] Shademan, A. und F. Janabi-Sharifi: *Using scale-invariant feature points in visual servoing.* In: *SPIE: OpticsEast 2004: Machine Vision and Its Optomechatronic Applications Conference*, Band 5603, Seiten 63–70, Philadelphia, Pennsylvania, USA, Oktober 2004.

[SLWG99] Siebel, N. T., O. Lang, F. Wirth und A. Gräser: *Robust Positioning of a Robot by Visual Servoing using a Trust-Region Method.* Forschungsbericht der Deutschen Forschungsvereinigung für Meß-, Regelungs- und Systemtechnik (DFMRS) e.V, 99(1):23–39, November 1999.

[Sut74] Sutherland, I. E.: *Three-dimensional data input by tablet.* In: *IEEE on Computer Graphics*, Seiten 453–461, April 1974.

[SW80] Sanderson, A. C. und L. E. Weiss: *Image-based visual servo control using relational graph error signals.* In: *IEEE International Conference on Robotics and Automation*, Seiten 1074–1077, 1980.

[SWN87] Sanderson, A. C., L. E. Weiss und C. P. Neuman: *Dynamic sensor-based control of robots with visual feedback.* IEEE Transaction Robotics and Automation, RA-3:404–417, Oktober 1987.

[TC05] Tahri, O. und F. Chaumette: *Point-Based and Region-Based Image Moments for Visual Servoing of Planar Objects.* IEEE Transactions on Robotics, 21(6):1116–1127, Dezember 2005.

[TL89] Tsai, R. und R. Lenz: *A new technique for fully autonomous and efficient 3D robotics hand/eye calibration.* IEEE Transactions on Robotics and Automation, 5(3):345–358, Juni 1989.

[Tsa87] Tsai, R.: *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses.* IEEE Transactions on Robotics and Automation, 3:323–344, August 1987.

[Whi72] Whitney, D. E.: *The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators.* Journal of Dynamic Systems, Measurement, and Control, 122:303–309, Dezember 1972.

[Wit83] Witkin, A.P.: *Scale-space filtering.* In: *International Joint Conerence on Artificial Intelligence*, Seiten 1019–1022, Karlsruhe, Germany, 1983.

[YA94] Yoshimi, B. und P. K. Allen: *Active, uncalibrated visual servoing.* In: *IEEE International Conference on Robotics and Automation*, Seiten 156–161, San Diego, CA, USA, Mai 1994.