Studienarbeit im Fach Systemtechnik und technische Kybernetik

Entwicklung einer Programmierschnittstelle zum Roboterarm „Katana“

vorgelegt am

7. Januar 2010

von

Ingo Schalk-Schupp

geb. am 24.12.1979

in Karlsruhe

STK-A 168791

Prüfer: Prof. Dr. rer. nat. Andreas Wendemuth

Betreuer: Dipl.-Inf. Ronald Böck

Lehrstuhl für kognitive Systeme

OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

EIT FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK

Selbständigkeitserklärung

Ich erkläre hiermit an Eides Statt, daß ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind ausnahmelos als solche gekennzeichnet.

Inhaltsverzeichnis

1. Thema, Ziele und Aufgaben

Ziel der Studienarbeit ist es, eine Programmierschnittstelle für zukünftige Arbeiten mit dem Katana zur Verfügung zu stellen, über welche die Nutzung des Roboterarms mit einfachen Programmierkenntnissen ermöglicht wird.

Der am Lehrstuhl vorhandene Roboterarm sowie die mitgelieferte Software sind auf Vollständigkeit und Funktionalität zu prüfen. Falls erforderlich, sind Reparaturmaßnahmen an der Hardware vorzunehmen.

Mit Hilfe der vorhandenen Dokumentation ist eine geeignete elektronische Schnittstelle zu wählen. Es ist eine Kommunikation zwischen PC und Katana aufzubauen, um die im Katana implementierten Befehle nutzen zu können.

Daraufhin ist zunächst eine grafische Oberfläche zu entwerfen, die die Steuerung der einzelnen Motoren durch die Maus ermöglicht. Da der Katana über einen flüchtigen Speicher verfügt, ist eine robuste Routine zur Justierung nach jedem Einschalten zu entwickeln, die den Arm in einen definierten Ausgangszustand versetzt.

Die Funktionalität des Katanas soll in Form einer C++-Klasse wiedergegeben werden, über deren Member-Funktionen die Katana-Befehle zur Verfügung stehen. Diese Klasse ist zur weiteren Verwendung in eine Bibliothek zu exportieren.

Die Ansteuerung der Motoren soll unter Angabe von Winkeln im Grad- oder Bogenmaß erfolgen. Gegebenenfalls sind diese Angaben auf die reell verwendeten Koordinaten abzubilden. Darüber hinaus soll eine inverse Kinematik implementiert werden, die Raumkoordinaten in die erforderlichen Steuerbefehle umsetzt bzw. prüft, ob der Arbeitsraum des Armes verlassen würde.

2. Überblick

Der in dieser Arbeit behandelte „Katana K200“ ist ein Knickarmroboter mit vier Freiheitsgraden und einem annähernd halbkugelförmigen Arbeitsraum mit einem Radius von etwa 50 cm.

Basis des Roboterarms ist ein um die aufrechte Achse drehbarer Turm, an dessen Schulter der Oberarm beweglich angebracht ist. Dieser ist wiederum über ein zum Schultergelenk paralleles Drehgelenk mit dem Unterarm verbunden, an dessen Ende ein Greifer befestigt werden kann.

Der Arm ist mit einem Winkelgreifer ausgerüstet, der es dem Roboter erlaubt, kleine Gegenstände zu greifen, anzuheben und zu bewegen. Zudem verfügt er über mehrere Kraft- und Infrarotsensoren zur Erkennung von Gegenständen und Hindernissen.

Für die Stromversorgung ist eine auf 12 V geregelte Gleichspannung vorgesehen. Diese Spannung kann über eine netzbetriebene Gleichstromquelle oder über eine Batterie zur Verfügung gestellt werden, so daß der Roboterarm auch für den mobilen Einsatz geeignet ist.

Über eine serielle Schnittstelle kann der Host mit dem Katana kommunizieren. Für tiefergehende Systemeingriffe, wie zum Beispiel die Installation einer neuen Firmware, sind spezielle Anschlüsse auf der Systemplatine des Katanas vorgesehen.

Für eingehendere Informationen zu den Eigenschaften des Katana K200 sei auf die Seminararbeit von Geue, Schuster und Walter [3] verwiesen.

3. Kommunikation

Es ist möglich, den Katana von verschiendenen Host-Systemen aus zu steuern. Neben dem PC kommen dafür auch passend programmierte Microcontroller und mobile Host-Roboter in Frage. Besonders zu nennen ist hier der „Labo 3“, ein tragfähiger Transportroboter, der zusammen mit dem Katana ausgeliefert wurde.

Der Katana verfügt über ein integriertes, örtlich verteiltes Betriebssystem: Auf der Hauptplatine, die im Turm untergebracht ist, befindet sich der Master-Controller. Dieser kommuniziert einerseits über ein Bus-System namens „Katabus“ mit den Slave-Controllern und nimmt andererseits über eine serielle RS-232-Schnittstelle Befehle von einem Host entgegen.

3.1. Physikalische Schnittstelle

Im Rahmen dieser Studienarbeit ist der Host des Katanas stets ein PC mit Windows-Betriebssystem. Ältere PCs verfügen über eine serielle COM-Schnittstelle, die direkt an den Steckverbinder des Katanas angeschlossen werden kann. Über einen Adapter ist es möglich, einen USB-Anschluß zu verwenden, über den unter Windows ein COM-Anschluß emuliert wird.

Seitens des Katanas wird das RS-232-Protokoll (siehe hierzu `http://de.wikipedia.org/wiki/RS-232`) nur teilweise unterstützt. Anwendung finden lediglich die Leitungen RX (Datenempfang), TX (Datenübertragung) und GND (Masse); eine Datenflußkontrolle findet auf elektronischer Ebene nicht statt.

Tabelle 3.1.: Aufbau eines Datenwortes. Die acht Datentransportierenden Bits x_i werden vom Startbit eingeleitet. Das Ende des Wortes wird durch das Stopbit markiert.

1	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	0
Start	Daten								Stop

Das Protokoll sieht für die Datenübertragung folgende Einstellungen vor:

- Datenrate: 57600 baud
- Synchronisierung: 1 Start-Bit und 1 oder 2 Stop-Bits.
- Paritätsprüfung: keine

Um den Datendurchsatz zu optimieren wurde für die Klasse `CProtocol` nur 1 Stop-Bit gewählt. So beträgt die Wortbreite insgesamt 10 bit. Der Informationsgehalt eines Symbols (Spannungspegel in einem bestimmten Zeitraum) beträgt in diesem Fall genau ein Bit (High/Low bzw. 1/0), womit die Symbolübertragungsrate identisch mit der Bitüberragungsrate ist. Mit einem Nutzdatenanteil von $\frac{8bit}{10bit} = 0,8$ beträgt die maximale Nutzdatenübertragungsrate $57600\frac{bit}{s} \cdot 0,8 \cdot \frac{1byte}{8bit} = 5,76\frac{kbyte}{s}$.

Der Katana ist nicht in der Lage, die zur Verfügung stehende Bandbreite voll auszuschöpfen. Zwischen den seinerseits gesendeten Worten tritt stets eine kurze Pause von mehreren Millisekunden auf. Vermutlich ist dies auf die um Rechenzeit konkurrierende Ausführung der internen und externen Kommunikationsaufgaben zurückzuführen: Nachdem der Katana einen Befehl empfangen hat, wird er ihn intern auswerten und gegebenenfalls Befehle an die Motorcontroller schicken, deren Antwort abwarten und erst danach dem Host eine Antwort übermitteln.

3.2. Sicherung durch Prüfsumme

3.2.1. Befehlsformat

Das im Katana implementierte Serial-Zero-Protocol (SZP) legt wie in Tabelle 3.2 dargestellt fest, welche Bedeutung die Worte eines zu übertragenden Paketes haben. Ein Datenwort trägt hier genau ein Byte, daher wird im Folgenden von Bytes die Rede sein.

Nicht dokumentiert [4] ist die notwendige Prüfsumme, die zu jedem Befehl zu erstellen ist. Hierbei handelt es sich um eine zyklische Redundanzprüfung mit 16 bit Wortbreite. Die Berechnung der Prüfsumme auf dem PC ist in der mitgelieferten Bibliothek Katana Native Interface (KNI) implementiert und wurde für die entwickelte Programmierschnittstelle in vereinfachter Form übernommen.

Die Prüfsumme (im Folgenden auch CRC) wird über den Befehlsteil des Paketes, also über Opcode und Daten, gebildet. Die zwei entstehenden Bytes werden an das Paket angehängt.

Die führenden Nullen im Header sind entgegen der Dokumentation [4] für eine reibungslose Kommunikation nicht erforderlich. Vermutlich dienen sie dazu, dem Katana Zeit für die Verarbeitung noch anstehender Aufgaben zu verschaffen. Da aber ohnehin auf jeden Befehl eine Antwort abgewartet werden soll und der Master-Controller zwischen Antwort und dem nächsten Befehl keine Aufgaben mehr hat, da er an die Slaves delegiert, ist diese Sicherheitsmaßnahme hinfällig. Zudem wird durch die verwendete Sendefunktion unter Windows die Programmausführung solange unterbrochen, bis der Katana alle Zeichen aus dem Kommunikationspuffer abgeholt hat. Sollte es also dennoch zu einer Verzögerung kommen, so wird diese automatisch berücksichtigt.

Möglicherweise dienen die Nullen auch der Synchronisation der Datenübertragungsgeschwindigkeit. Da der Katana aber auf 57600 baud voreingestellt ist und davon nicht abweicht, bliebe allenfalls die Anzahl der Stop-Bits zu ermitteln. Dafür wären aber keine sechzehn Wiederholungen nötig, noch müßte man diese Information vor jedem Befehl neu ermitteln.

Schließlich zeigt sich im Experiment, daß die führenden Nullen bis auf Geschwindigkeitseinbußen keine Auswirkung zu haben scheinen.

Tabelle 3.2.: Befehlspaket vom Host zum Katana gemäß dokumentiertem Serial-Zero-Protocol

	Katana-Befehlspaket					
Abschnitt	Header			Befehlslänge	Befehl	
Inhalt	16x0	1	Adresse	Länge	Opcode	Daten
Länge in Bytes	16	1	1	1	1	n(0..255)

Das aus der angehängten Prüfsumme resultierende Paket ohne die führenden Nullen hat die in Tabelle 3.3 dargestellte Struktur.

Tabelle 3.3.: Befehlspaket vom Host zum Katana einschließlich Prüfsumme (CRC)

	Katana-Befehlspaket mit Prüfsumme					
Abschnitt	Header		Befehlslänge	Befehl		Prüfsumme
Inhalt	1	Adresse	Länge	Opcode	Daten	CRC16
Länge in Bytes	1	1	1	1	n(0..255)	2

3.2.2. Antwortformat

Auf jeden Befehl antwortet der Katana mit einem Antwortpaket. Das SZP gibt die Struktur vor, jedoch wird den in der Dokumentation [4] angegebenen Antwortdaten wiederum eine nicht dokumentierte Prüfsumme angehängt, so daß sich die Struktur nach Tabelle 3.4 ergibt.

Tabelle 3.4.: Antwortpaket vom Katana zum Host einschließlich Prüfsumme (CRC)

	Katana-Antwortpaket		
Abschnitt	Antwort		Prüfsumme
Inhalt	Ack-Tag	Daten	CRC
Länge in Bytes	1	n(0..255)	2

3.3. Anwendung

Aufbauend auf dem modifizierten Serial-Zero-Protocol stehen dem Katana – je nachdem, ob man Befehlsvarianten als eigene Befehle auffaßt – 11 bis 18 Befehle zur Verfügung, darunter solche zur Ansteuerung der Motorregler, zum Auslesen von Positionen oder Sensordaten oder zum Liefern von Informationen über Eigenschaften des Roboterarmes.

3.3.1. Kommunikationsprotokoll

Jeder Befehl an den Katana wird mit einem Opcode eingeleitet. Dieser besteht stets in einem menschenlesbaren Großbuchstaben nach dem American Standard Code for Information Interchange (ASCII). Dieser Opcode definiert sowohl den Befehl als auch die nachfolgende Datenmenge. Zu jedem Befehl gehört eine vorher bekannte Anzahl von Bytes, die auf den Opcode folgt.

Grundsätzlich reagiert der Katana auf jeden Befehl mit der Übermittlung eines Antwortpaketes. Mit dem in Tabelle 3.4 erwähnten „Ack-Tag“ (Acknowledge-Tag) wird der erfolgreich übermittelte Befehl quittiert und zugleich die Antwort eingeleitet. Zu jedem Befehl existiert ein Ack-Tag, und zwar der dem Opcode entsprechende ASCII-Kleinbuchstabe. Darauf folgen gegebenenfalls Daten, die wie für das Befehlspaket auch für das Antwortpaket in der Länge vorherbestimmt sind.

Die eindeutige Zuordnung von Opcodes zu Ack-Tags sowie die bekannte Länge jeder Antwort ermöglicht eine einfache Fehlererkennung: Antwortet der Katana nicht oder entspricht das Ack-Tag nicht dem Opcode oder fällt die Antwort zu kurz aus, so kann dies erkannt und auf den Fehler reagiert werden.

Auch seitens des Katanas ist eine Fehlererkennung eingerichtet: entspricht die dem Befehl angehängte Prüfsumme nicht den Daten, so wird das offenbar fehlerhaft übermittelte Befehlspaket mit der ASCII-Meldung ’`err`’ quittiert. Da kein Befehls-Opcode ’`E`’ lautet, stellt das erste Zeichen dieser Fehlermeldung niemals ein gültiges Ack-Tag dar, so daß nicht fälschlicherweise angenommen werden kann, der Katana liefere eine gültige Antwort.

Die erfolgreiche Kommunikation erfordert, daß der PC die den Befehlen zugeordneten Daten- und Antwortlängen kennt. Hierfür kommen zwei Methoden in Frage:

1. Die Befehls- und die Antwortlänge zu jedem Befehl werden der Dokumentation entnommen und zur Kompilierzeit fest vorgegeben.

2. Der Befehl ’`X`’ weist den Katana an, Informationen über all seine Befehle zu übermitteln. Um die Antwort auszuwerten, müssen nur die Länge der Antwort dieses einzigen Befehls und deren Format bekannt sein. Daraus lassen sich die Befehlsdaten- und Antwortlängen für alle anderen Befehle ableiten. Das Programm speichert die Daten zur Laufzeit.

Es stellte sich heraus, daß die Dokumentation [4] des Katanas nicht zu der vorliegenden Hardware paßt. Die angegebenen Daten haben sich in mehreren Fällen als unzuverlässig erwiesen. Daher wurde auf die zweite Methode zurückgegriffen.

3.3.2. Befehlssatz

Die meisten Katana-Befehle scheinen über die Firmware-Versionen hinweg unverändert geblieben zu sein. Die vorliegende Dokumentation [4] enthält eine Beschreibung der meisten Befehle, einige vom Katana K200 angebotene werden jedoch nicht angeführt, und einige dokumentierte werden wiederum vom Katana nicht unterstützt.

Sämtliche tatsächlich zur Verfügung stehenden Befehle sind im Anhang A aufgeführt.

Nach dem Einschalten des Roboters sowie nach jedem Rücksetzen sollte zunächst die Methode `Adjust()` aufgerufen werden, die den Katana in die Ausgangsstellung versetzt und anhand der Anschlagpositionen die absoluten Winkelpositionen setzt. Danach können direkt und ohne die Gefahr eines Überlaufes (siehe Abschnitt 4.3.1) die Motoren angesteuert werden. Hierzu dient die Methode

```
void setMotorPosition(
byte        motor   ,
TPosition   position,
TMotorState state   =CKatana::MS_MOVE)
```

Der Parameter `motor` gibt die Nummer des zu bewegenden Motors an. `position` ist der anzusteuernde Winkel. Der Typ `TPosition` ist in der Klasse `CKatana` als katana-interne Winkeleinheit definiert, während `CKatanaKinematic` mit Grad-Winkeleinheiten arbeitet. `state` gibt den gewünschten Motorzustand an. Dieser Parameter kann zum Bewegen des Motors weggelassen werden.

Die Befehle

```
CKatana* pKatana;
pKatana=new CKatana(1);  // Katana an COM1
pKatana->purge();
pKatana->Adjust();
pKatana->setMotorPosition(0, 1000);
// 0=Turm: 1000 Winkeleinheiten
```

bewegen den Turm zu der Position, die 1000 Einheiten im Uhrzeigersinn von der Ausgangslage entfernt liegt.
Dagegen veranlassen die Befehle

```
CKatanaKinematic* pKatana;
pKatana=new CKatanaKinematic(1);  // Katana an COM1
pKatana->purge();
pKatana->Adjust();
pKatana->setMotorPosition(0, 90); // 0=Turm: 90°
```

den Turm, sich positiv rechtwinklig zur Ausgangsposition auszurichten.
Mit Hilfe der dazu komplementären Routine `TPosition getMotorPosition(byte motor)` können die Motorpositionen abgefragt und gespeichert werden. Auch hier gilt, daß die Winkeleinheit von der verwendeten Klasse abhängig ist. Speichert man die Positionen sämtlicher Motoren in einem Feld, so lassen sich die Werte später abrufen, um beispielsweise ein Teaching von Positionen zu realisieren.

4. Kinematik

Die Kinematik beschäftigt sich mit der Beschreibung des physischen Aufbaus eines Roboters und der durch seine Struktur vorgegebenen Bewegungsmöglichkeiten und -grenzen. Dabei wird stets von Gelenken ausgegangen, die über starre Verbindungen aneinander gekoppelt sind.

Mit Einteilung der Gelenke in rotatorische und translatorische mit jeweils nur einer Bewegungsachse lassen sich alle anderen Gelenkarten, auch solche mit mehreren Freiheitsgraden, modellieren. In letzterem Fall geht man dazu von mehreren einachsigen Gelenken aus, die miteinander starr verbunden sind. Einer solchen fiktiven Verbindung muß dabei kein realer Körper entsprechen.

4.1. Theoretische Grundlagen der Kinematik

4.1.1. Kinematische Ketten

Angeles [1] beschreibt kinematische Ketten als einen Satz starrer Körper, den Verbindungen, die über kinematische Paare (im Folgenden: Kopplungen) miteinander verbunden seien. Kinematische Kopplungen seien demnach die Einschränkungen, denen die relative Bewegung der beiden Verbindungen unterworfen sei.

Beispiele für solche Kopplungen seien die rotatorische Kopplungen, die eine relative Drehbewegung um eine wohldefinierte Achse zwischen den Verbindungen zuließe, und die prismatische, deren Kontaktfläche einem Prisma beliebigen Querschnitts entspreche und die eine Verschiebebewegung definierter Richtung ermögliche. Diese kinematischen Kopplungen schlössen jegliche andersartige Relativbewegung der beteiligten Verbindungen zueinander aus. Eine rotatorische Kopplung besitze eine Achse, die genau der Rotationsachse entspreche, und die prismatische Kopplung besitze zwar eine Richtung, jedoch

keine verortete Achse, da für eine Verschiebebewegung sämtliche Achsen gleicher Orientierung äquivalent seien.

Eine *einfache* kinematische Kette sei eine kinematische Kette, deren Verbindungen mit jeweils maximal zwei anderen Verbindungen in Kontakt stünden, also keine Verzweigungen aufweise. Eine *offene* einfache kinematische Kette sei eine einfache kinematische Kette, von der genau zwei Verbindungen mit jeweils nur einer Verbindung gekoppelt seien. Die erste Verbindung heiße Basis, die letzte heiße Endeffektor.

Dem Aufbau des Katanas entsprechend werden im folgenden nur offene einfache kinematische Ketten mit rotatorischen Kopplungen betrachtet.

4.1.2. Koordinatensysteme am Roboter

Aufgrund der im allgemeinen komplizierten Bewegungsmöglichkeiten eines Roboterarmes bietet es sich an, zur mathematisch einfacheren Beschreibung zunächst nur die Relativbewegungen der Verbindungen zueinander zu betrachten, die sich stets auf eine Basisbewegung mit einem einzigen variablen Parameter reduzieren lassen. Im Falle einer Rotation wäre dies der Drehwinkel, im Falle einer Translation die Distanz. Später lassen sich die einzelnen Relativbewegungen zu einer Gesamtbewegung zusammenfassen, um so die Position des Endeffektors in einem Referenzsystem zu erhalten. Dies geschieht, wie später gezeigt werden wird, durch einfache Matrizenmultiplikation.

Um eine Bewegung zweier Verbindungen zueinander angeben zu können, benötigt man ein Bezugssystem (Anmerkung: „Referenzsystem“ und „Bezugssystem“ werden hier synonym verwendet). Zweckmäßigerweise wird dieses orts- und orientierungsfest an eine der beiden angesprochenen Verbindungen gekoppelt. So entsteht eine mathematisches Repräsentation der physischen Verbindungen der kinematischen Kette. Offen bleibt derweil noch die Modellierung der kinematischen Kopplungen, also der Bewegungsmöglichkeiten. Diese werden im folgenden Abschnitt behandelt.

Der Ort und die Orientierung eines Bezugssystems sind zwar willkürlich wählbar, jedoch ergeben sich durch geschickte Wahl Vereinfachungen für verschiedene Ziele. So ist es für die Kinetik, also die Untersuchung dynamischer Eigenschaften im Zusammenhang mit den wirkenden Kräften, sinnvoll, den Ursprung jedes Koordinatensystems in den Massenschwerpunkt der zugehörigen Verbindung zu legen und eine oder zwei Koordinatenachsen an den Hauptträgheitsachsen zu orientieren [6]. Für die Kinematik hat sich dagegen das

Verfahren nach Denavit und Hartenberg durchgesetzt, das in Abschnitt 4.2.1 besprochen wird.

In dieser Arbeit wird von orthonormalen, kartesischen, rechtshändigen Koordinatensystemen ausgegangen. Diese reichen für die Beschreibung und Berechnung der Roboterbewegungen aus, ohne deren Allgemeinheit zu beschränken. Benötigt man in der Anwendung andere Koordinaten, so können die unterschiedlichen Systeme nach Belieben mit Hilfe bekannter Verfahren ineinander überführt werden.

4.1.3. Transformationsbeziehungen zwischen Koordinatensystemen und Frames

Definition 1 *Ein Koordinatensystem sei durch $\{X\}$ bezeichnet. Ein an einem starren Körper verankertes Koordinatensystem, das sich mit dem Körper bewegt, heiße Frame. Auch hierfür sei die Notation $\{X\}$ eingeführt.*

Die relative Lage eines Koordinatensystems $\{B\}$ zu einem System $\{A\}$ läßt sich beschreiben durch Angabe einer Transformation, die $\{A\}$ in $\{B\}$ überführt. Das Verwenden unterschiedlicher Bezugssysteme hat dabei unterschiedliche Werte für die Transformationen zur Folge, jedoch sind sie prinzipiell äquivalent. Für die Kinematik bietet es sich an, für jede Transformation das System $\{A\}$ als Bezugssystem zu wählen.

Definition 2 *Eine Transformation, die System $\{A\}$ in System $\{B\}$ überführt und dabei $\{A\}$ als Referenzkoordinatensystem verwendet, werde durch ${}^A_B T$ bezeichnet.*

Definition 3 *Ein Vektor* $\mathbf{r}$*, beschrieben in den Koordinaten des Systems $\{X\}$ sei bezeichnet durch ${}^X\mathbf{r}$. Der Vektor, der die Lage eines Punktes P in den Koordinaten des Systems $\{X\}$ ausdrückt, heiße ${}^X\mathbf{r}_P$.*

Satz 1 *Gegeben seien ein Vektor* $\mathbf{r}$ *und zwei Systeme $\{A\}$ und $\{B\}$. Bei bekannter Transformation ${}^A_B T$ läßt sich die Darstellung von* $\mathbf{r}$ *in $\{A\}$, nämlich ${}^A\mathbf{r}$, auch darstellen als auf ${}^B\mathbf{r}$ angewandte Transformation: ${}^A\mathbf{r} = {}^A_B T \circ {}^B\mathbf{r}$.*

Satz 2 *Gegeben seien drei Systeme $\{A\}$, $\{B\}$ und $\{C\}$. Bei bekannten Transformationen ${}^A_B T$ und ${}^B_C T$ ergibt sich die Transformation ${}^A_C T$ transitiv aus der Verknüpfung der einzelnen Transformationen: ${}^A_C T = {}^A_B T \circ {}^B_C T$.*

4.1.4. Homogene Koordinaten

Jede Transformation ${}^A_B T$ zweier orthonormaler Systeme $\{A\}$ und $\{B\}$ gleicher Händigkeit läßt sich durch die Kombination zweier Operationen erreichen: Zuerst wird der Ursprung von $\{A\}$ auf den Ursprung von $\{B\}$ verschoben. Diese Verschiebung läßt sich durch einen Vektor $\mathbf{q}$ beschreiben. Es folgt eine Rotation, die das verschobene System mit $\{B\}$ zur Deckung bringt. Diese Rotation ist im dreidimensionalen Raum durch eine orthogonale Matrix $R_{3\times 3}$ vollständig beschrieben.

Während die durch R beschriebene Rotation an sich eine streng lineare Abbildung ist, wird die Gesamttransformation durch die Verschiebung um $\mathbf{q}$ zur affinen Abbildung degradiert.

Wie in der Dokumentation zu „Cinderella“ [5] anschaulich beschrieben, wird durch Einführen einer neuen Dimension und Abbildung jedes Punktes (x, y, z) auf die Hyperebene $(x, y, z, 1)$ die homogenen Koordinaten eingeführt. Jeder Punkt $(x, y, z, w \neq 1)$ im vierdimensionalen Raum sei äquivalent zu dem auf der Hyperebene befindlichen Punkt $(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1)$.

Die affinen Abbildungen des zugrundeliegenden dreidimensionalen Raumes werden auf die linearen Abbildungen des vierdimensionalen abgebildet, so daß sich jede Transformation zwischen zwei Frames in homogenen Koordinaten als 4×4-Matrix darstellen läßt:

$$ {}^A_B T = \begin{bmatrix} {}^A_B R & {}^A_B \mathbf{q} \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{4.1} $$

Es sind weiterhin ${}^A_B R$ die besprochene Rotationsmatrix und ${}^A_B \mathbf{q}$ der Verschiebungsvektor, die $\{A\}$ auf $\{B\}$ abbilden. ${}^A_B T$ werde nun Transformationsmatrix von $\{A\}$ zu $\{B\}$ genannt.

Die Verknüpfungen zwischen Transformationen und die einer Transformation mit einem Vektor bilden sich in den homogenen Koordinaten auf die Multiplikation ab:

$$ {}^A_C T = {}^A_B T \cdot {}^B_C T \tag{4.2} $$

$$ {}^A \mathbf{r} = {}^A_B T \cdot {}^B \mathbf{r} \tag{4.3} $$

4.2. Vorwärtskinematik

Zuvor wurden Transformationsmatrizen, homogene Koordinaten und Frames besprochen. Diese finden bei der numerischen Modellierung des Katanas Anwendung.

Eine voll besetzte Transformationsmatrix in homogenen Koordinaten für den dreidimensionalen Raum enthält bis zu 12 Fließkommawerte, wenn man nur Verschiebungen und Rotationen, nicht aber Scherungen, Spiegelungen oder Projektionen berücksichtigt. Diese Daten sind jedoch redundant. Jede denkbare dieser speziellen Transformationen läßt sich durch vier einparametrige Transformationen darstellen, so daß sich auch die vollständige Information in vier Fließkommazahlen darstellen läßt.

4.2.1. Modellierung mit Denavit-Hartenberg oder dem Starrkörperverfahren

Eine geeignete formale Methode, eine beliebige Transformation ohne Scherung, Spiegelung oder Projektion darzustellen ist die von Denavit und Hartenberg 1955 beschriebene. [2]

Die zu beschreibende Transformation zwischen zwei Frames $\{i-1\}$ und $\{i\}$ wird äquivalent ersetzt durch (X_i, Y_i, Z_i seien die Achsen des Frames $\{i\}$, O_i der Ursprung):

- eine Rotation um Z_{i-1} mit Parameter Θ_i, so daß X_{i-1} und X_i parallel liegen,
- eine Translation entlang Z_{i-1} mit Parameter d_i, um X_{i-1} und X_i zur Deckung zu bringen,
- eine Translation entlang X_i mit Parameter a_i, so daß sich die Ursprünge O_{i-1} und O_i decken und
- eine Rotation um X_i mit Parameter α_i, so daß die beiden Frames $\{i-1\}$ und $\{i\}$ sich völlig decken.

Um die Parameter eindeutig bestimmen zu können, gehören zur Denavit-Hartenberg (DH)-Konvention zudem einige Festlegungen: Alle Gelenkachsen werden numeriert. Einem bestimmten Schema zufolge werden an diesen Achsen die Frames ausgerichtet.

Nach dieser Vorbereitung können die Transformationsmatrizen zwischen den Frames bestimmt werden, indem man das Produkt der vier obengenannten Einzeloperationen bildet.

Während die Gelenkachsen in der Beschreibung der DH-Transformationen im allgemeinen schief zueinander stehen und sich nicht schneiden, treten beim Katana (wie auch bei den meisten anderen realen Roboterarmen) stets Sonderfälle auf, für die die allgemeine Vorgehensweise nur mehrdeutig beschrieben ist. Dies tritt dann ein, wenn zwei aufeinanderfolgende Gelenkachsen sich schneiden oder parallel verlaufen. Für diese Fälle müssen praktikable und notwendigerweise einheitliche Festlegungen getroffen werden.

Es gibt grundsätzlich zwei unterschiedliche Konventionen, die vorschreiben, wie die Frames zu plazieren sind: Bei der proximalen Methode, die ursprünglich von Denavit und Hartenberg vorgeschlagen wurde, und bei der der Koordinatenursprung jedes Frames am nahegelegenen (proximalen) Ende einer Verbindung zu liegen kommt, wird jede Achsvariable q_n an der Koordinatenachse Z_{n-1} gemessen.

Eine Modifikation des DH-Verfahrens ist das sogenannte Starrkörperverfahren, auch distales DH-Verfahren genannt. Hier werden die Ursprünge der Frames jeweils an das entfernte Ende einer Verbindung gelegt, so daß sich jede Bewegungsachse L_n mit der entsprechenden Koordinatenachse Z_n deckt.

Die für den Katana gebrauchten Konventionen lassen sich schrittweise darstellen und stellen einen Spezialfall des Starrkörperverfahrens dar:

- Numeriere die Verbindungen beginnend mit der Nummer 1 für die Basis! Nenne sie J_1 bis J_n!
- Errichte ein Basiskoordinatensystem $\{0\}$, dessen Z_0-Achse parallel zur Drehachse J_1 ist!
- Richte alle Z_i entlang der Drehachsen den Verbindungen J_{i+1} aus!
- Lege jeden Ursprung O_i in den Schnittpunkt von Z_{i-1} und Z_i! Besitzen diese keinen Schnittpunkt, dann errichte die Normale zwischen Z_{i-1} und Z_i, und lege O_i auf den Schnittpunkt dieser Normalen mit Z_i!
- Setze jedes $X_i := \pm\frac{(Z_{i-1}\times Z_i)}{\|Z_{i-1}\times Z_i\|}$! Wähle das Vorzeichen so, daß X_i der mechanischen Drehachse nach der Rechte-Hand-Regel entspricht! Sind Z_{i-1} und Z_i parallel, so setze X_i gleich einer gemeinsamen Normalen, die in Richtung des nächsten Ursprungs O_{i+1} weist!

Tabelle 4.1.: DH-Parameter für den Katana nach Starrkörpermodell. h_1 ist die Höhe der Basis, l_2 die Länge des Unterarmes und l_1 die Länge des Oberarmes vom Ellenbogen bis zum Greifpunkt. q_i sind die freien Achsvariablen jedes Gelenkes. Anmerkung: Die im DH-Schema verwendeten Indizes entsprechen nicht den Motornummern in der Bibliothek. Diese beginnen bei 0.

		DH-Parameter			
i	Gelenk	θ_i	d_i	a_i	α_i
1	Turm	$0° + q_1$	h_1	0	$-90°$
2	Schulter	$0° + q_2$	0	l_1	$180°$
3	Ellenbogen	$90° + q_3$	0	0	$90°$
4	Unterarm	$90° + q_4$	l_2	0	$0°$

- Wähle jedes Y_i so, daß es das zugehörige System zu einem rechtshändigen System ergänzt: $Y_I := \frac{(Z_i \times X_i)}{\|Z_i \times X_i\|}$!
- Für die letzte Verbindung, den Endeffektor, ist kein nächster Ursprung vorhanden. Richte daher Z_n an Z_{n-1} aus! X_n stehe senkrecht auf Z_n und zeige in die Bewegungsrichtung beim Öffnen des Greifers, von oben gesehen nach rechts. Y_n ergänze wiederum zu einem rechtshändigen System.

Danach können wie oben beschrieben die DH-Parameter bestimmt werden. Für den Katana sind diese in Tabelle 4.1 aufgeführt.

4.2.2. Implementierung des Modells

Zum bequemen Implementieren jeder offenen kinematischen Kette steht die im Rahmen der Studienarbeit entwickelte Bibliothek `ISSKinematics` zur Verfügung. Diese definiert eine allgemeine Matrix-Klasse mit vielen Operatoren, Funktionen zum Erzeugen verschiedener Transformationsmatrizen (Translation und Rotation) in homogenen Koordinaten sowie eine Funktion zum Erzeugen einer konstanten DH-Matrix unter Angabe aller vier DH-Parameter. Zusätzlich sind die jeweiligen Inversen verfügbar, so daß ohne Rechenaufwand oder Fehleranfälligkeit wegen nichtkommutativer Operationen Vorwärts- und Rückwärtstransformationen zwischen den Frames errechnet werden können.

Darüberhinaus definiert `ISSKinematics` die Klasse `DHRotation`, die keine konstante Matrix liefert, sondern variable Platzhalter für den Drehwinkel um die betreffende Ach-

se anbietet. So ist es möglich, nicht nur feste Konfigurationen des Katanas zu betrachten, indem seine Freiheitsgrade mitmodelliert werden. Konkrete Werte für den Winkel eines Gelenkes können so später ergänzt werden, ohne daß die restlichen Modellparameter erneut angeführt werden müssen. Auch zu `DHRotation` exisitiert die entsprechende Inverse `DHInvRotation`.

Ein Analogon für Translationsachsen, `DHTranslations` und `DHInvTranslation` wurde der Vollständigkeit halber ebenfalls implementiert, wird aber für den Katana, der nur über Drehgelenke verfügt, nicht benötigt.

Um beispielsweise die in Tabelle 4.1 dargestellten Transformationen und ihre Komplemente mittels `ISSKinematics` zu modellieren, kann auf bequeme Weise der Klassenkonstruktor genutzt werden. So sind die einzelnen Transformationen leicht in einer Tabelle zusammenzufassen, und die Klasse kann für beliebige nach DH modellierbare Roboter eingesetzt werden. Die katana-spezifischen Maße `h`, `l1` und `l2` (Turmhöhe sowie Länge der beidem Armabschnitte) seien in diesem Beispiel bereits mit Zahlenwerten belegt:

```
DHRotation DHKatana[] = {
 DHRotation( 0    ,  h,  0,-90*deg)
,DHRotation( 0    ,  0, l1,180*deg)
,DHRotation(90*deg,  0,  0, 90*deg)
,DHRotation(90*deg, l2,  0,  0    )
};

DHInvRotation DHInvKatana[] = {
 DHInvRotation( 0    ,  h,  0,-90*deg)
,DHInvRotation( 0    ,  0, l1,180*deg)
,DHInvRotation(90*deg,  0,  0, 90*deg)
,DHInvRotation(90*deg, l2,  0,  0    )
};
```

Durch die Definition eine Referenzkoordinatensystems kann der Ursprung beliebig plaziert und orientiert werden.

```
Matrix  // Ref.-System beschreibt die rel. Lage der Basis
TR0(Translate(225,96,0)*RotateZ(145*deg)),
T0R(InvRotateZ(145*deg)*InvTranslate(225,96,0));
```

```
Matrix Origin(4,1);  // Dimension 4*1, also ein
Origin.clear().set(3,1.0); // Vektor in homogenen Koordinaten

Matrix  // Die verschiedenen Transformationen im Beispiel:
T01(    DHKatana[0].get(  50*deg)),  // 0:Basis -> 1:Turm
T02(T01*DHKatana[1].get( -44*deg)),  // 0:Basis -> 2:Oberarm
T03(T02*DHKatana[2].get(-115*deg)),  // 0:Basis -> 3:Unterarm
T04(T03*DHKatana[3].get(   0*deg));  // 0:Basis -> 4:Greifer

Matrix TR4(TR0*T04);  // Trafo von R:Referenz zum 4:Greifer
```

Die Anwendung für die Vorwärtskinematik besteht darin, die zuvor als Modell des Roboters erstellten Matrizen auszuwerten. Dazu müssen sämtliche Gelenkpositionen bekannt sein.

Jede Gelenkposition wird als Eingabeparameter der Methode `DHRotation::get(double theta)` übergeben, die eine konstante Matrix zurückliefert. Diese stellt das transformierte Frame dar.

Um die gesamte Kette von Basis bis Endeffektor zu berechnen, müssen die konstanten Matrizen lediglich miteinander in aufsteigender Reihenfolge multipliziert werden. Man erhält den Greiferframe für die zugrundeliegenden Gelenkstellungen. Dieser enthält sowohl Ort als auch Orientierung des Greifers und hat analog zu Gleichung 4.1 den Aufbau:

$$\{G\} = \left\{ \begin{array}{cc} {}^{1}_{G}Rot & {}^{1}_{G}\mathbf{q} \\ \mathbf{0}^{T} & 1 \end{array} \right\} \tag{4.4}$$

Hier stellt Rot die Orientierung des Greifers dar, und $\mathbf{q}$ beschreibt die Verschiebung des Ursprungs gegenüber dem Referenzframe.

4.3. Inverse Kinematik

Während die Vorwärtskinematik eine Vorschrift zur Berechnung der Greiferposition aus den Gelenkwinkeln liefert, ist es das Ziel der inversen Kinematik, zu einem gegebenen

Frame, der Soll-Lage und Soll-Orientierung des Greifers beschreibt, diejenigen Gelenkstellungen, die zum Erreichen dieses Zielframes führen, zu berechnen.

Im allgemeinen ist die inverse Kinematik aufgrund von Nichtlinearitäten, Polen und Mehrdeutigkeiten numerisch bzw. mathematisch komplizierter als die stets eindeutige Vorwärtsberechnung.

Satz 3 *Geschlossene Lösungen existieren nur für kinematische Ketten, deren Achsen parallel verlaufen oder sich in einem Winkel von 90° schneiden [6].*

Numerische Lösungen existieren bereits für alle offenen kinematischen Ketten mit sechs Freiheitsgraden [6].

Der Aufbau des Katanas stellt eine offene kinematische Kette dar. Diese weist nur senkrecht aufeinanderstehende oder zueinander parallele aufeinanderfolgende Achspaare auf. Somit ist die notwendige Bedingung aus Satz 3 für die Existenz einer analytischen Lösung erfüllt.

4.3.1. Arbeitsraum des Katanas

Um zunächst einen Überblick über den Arbeitsraum des Katanas zu erhalten, waren die Anschlagwinkel der einzelnen Gelenke möglichst genau zu bestimmen. Die Bemaßungen der Roboterarmes sind in keiner verfügbaren Dokumentation zu finden und sind wegen der Unzugänglichkeit der Achsmitten nur indirekt über trigonometrische Zusammenhänge zu ermitteln.

Der Katana ist kein gelenkiger Roboter, das heißt, daß er an keinem Punkt seines Arbeitsraumes den Endeffektor beliebig im Raum orientieren kann. Der Greifer weist stets von der Basis fort und kann beispielsweise nicht dazu eingesetzt werden, einen Gegenstand auf der Stelle um seine aufrechte Achse zu drehen.

Der Turm dreht sich um die senkrechte Achse und bestimmt so die „Blickrichtung“ des Armes. Durch mechanische Anschläge beträgt der Öffnungswinkel dieses Gelenkes $347,84°$. Die Winkelhalbierende wird aus numerischen und praktischen Gründen zu $0°$ definiert. Praktisch wird hierdurch festgelegt, daß der Katana sich nach links und rechts gleich weit drehen kann.

Hintergrund der numerischen Betrachtung ist die interne Repräsentation des Winkels im Katana: Hier dient ein aus zwei Bytes bestehendes, vorzeichenbehaftetes Wort als Speicher für die Position. Der Öffnungswinkel in diesen internen Koordinaten beträgt etwa 49702 Einheiten. Wieviel Grad dies entspricht, soll im folgenden geklärt werden. Wählt man den Nullpunkt ungeschickt, so kann der Katana den Wertebereich zwischen -32768 und 32767 verlassen. Solch ein Über- oder Unterlauf hätte zur Folge, daß der Regler des Gelenks sofort versucht, den Arm in die aktuelle Bewegungsrichtung zu bewegen, bis er auf den Anschlag trifft, oder in entgegengesetzte Richtung, was ein Zittern des Armes („ugly mechanical vibrations“ [4]) bewirkt, da der Über- oder Unterlauf ausgeglichen würde. Analoges gilt auch für die folgenden Gelenke; allerdings hat der Turm den numerisch größten Öffnungswinkel, so daß sich der für die Wahl des Nullpunktes sichere Bereich stark verkleinert.

Eine vollständige Umdrehung entspräche nach den gemessenen Werten $49702 \cdot \frac{360^\circ}{347{,}84^\circ} \approx 51440$ Einheiten. Es ist jedoch üblich, für Winkelgeber Vielfache üblicher Winkeleinheiten (Grad oder Neugrad) zu wählen, um die Umrechnung zu vereinfachen. Plausibler erscheinen daher aufgrund ihrer Primfaktorenzerlegung die dem Schätzwert nahegelegenen Zahlen $51450 = 2 \cdot 3 \cdot 5^2 \cdot 7^3$ und $51480 = 2^3 \cdot 3^2 \cdot 5 \cdot 11 \cdot 13 = 360 \cdot 143$. Untersucht wurden die Primfaktoren der Zahlen 51400 bis 51500, weitere Zahlen in diesem Bereich, deren Primfaktoren kleiner als 20 sind, lauten $51425 = 5^2 \cdot 11^2 \cdot 17$ und $51408 = 2^4 \cdot 3^3 \cdot 7 \cdot 17$. Die Teilbarkeit von 51480 durch 360 legt jedoch die Vermutung nahe, daß dies der genaue Wert ist. Die Dokumentation beziffert eine Umdrehung mit $29120 = 2^6 \cdot 5 \cdot 7 \cdot 13$ Einheiten, weist jedoch auf unterschiedliche Werte zwischen verschiedenen Versionen des Katanas hin.

Ein Grad entspricht damit 143 Winkeleinheiten.

Die Schulter hat einen Öffnungswinkel von $151,72^\circ$. Dies wurde durch Messen der Lage des Ellenbogens in der Nullstellung und den Anschlagpositionen ermittelt. Nullposition ist hier die waagerechte Lage, in der der Arm vorwärts ausgestreckt ist. Siehe Abbildung 4.1.

$$\alpha = 180^\circ - \arcsin\left(\frac{h_O - h_0}{l_1}\right) \approx 180^\circ - 48,36^\circ = 131,64^\circ \tag{4.5}$$

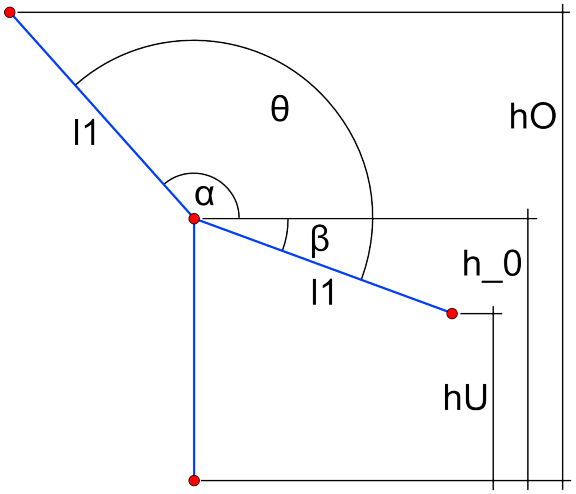

Abbildung 4.1.: Anschlagwinkel des Schultergelenks: In den Anschlägen wurde die jeweilige Höhe des Ellenbogens gemessen, um die Anschlagwinkel zu bestimmen.

$$\beta = \arcsin\left(\frac{h_U - h_0}{l_1}\right) \approx -20,09° \tag{4.6}$$

$$\theta = \alpha - \beta \approx 151,72° \tag{4.7}$$

Die internen Koordinaten liefern die Werte $M_{1,\max} = 17286$ und $M_{1,\min} = -22567$, wodurch sich eine Spanne von 39853 Einheiten ergibt. Pro Umdrehung entspricht das etwa 94560 Einheiten.

Ein Grad entspräche demnach $262\frac{2}{3}$ Motoreinheiten.

Der Ellenbogen kann sich vom ausgestreckten Zustand nach hinten oder vorne drehen. Der mechanische Anschlag ist in beiden Richtungen gleich beschaffen. Daher wird der Nullpunkt über die Messungen des Anschlags bestimmt. Dies verspricht eine höhere Genauigkeit, als den gestreckten Zustand so exakt wie mechanisch möglich einzustellen und von ihm als Nullpunkt auszugehen.

Das in Abbildung 4.2 dargestellte Problem reduziert sich auf die Bestimmung und Addition zweier Winkel in rechtwinkligen Dreiecken wie in Abbildung 4.3 zu sehen.

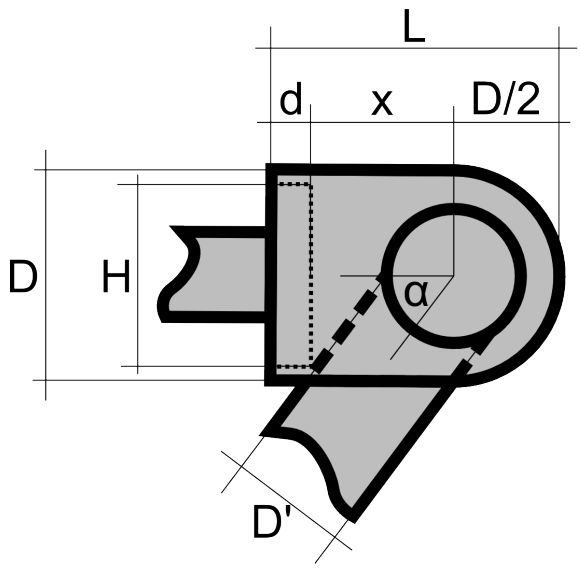

Abbildung 4.2.: Abmessungen des Ellenbogens: Der mechanische Anschlag befindet sich zwischen dem Teil mit der Höhe H und dem Oberarm der Dicke D'. Zu bestimmen war der Winkel α. Ausgenommen α konnten sämtliche hier angeführten Größen direkt gemessen oder berechnet werden.

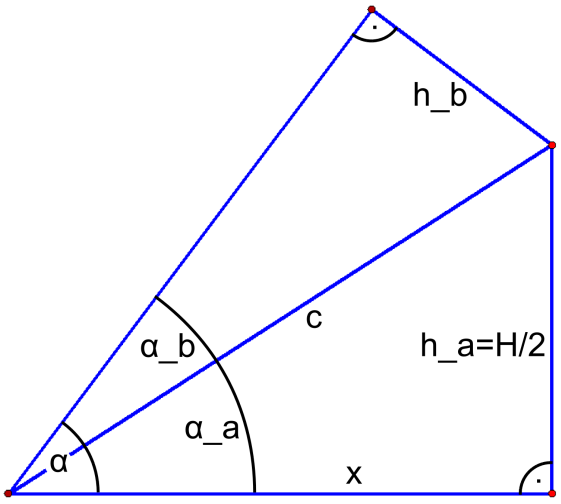

Abbildung 4.3.: Geometrie des Ellenbogengelenks: Zu bestimmen war die Summe der beiden Winkel α_a und α_b.

$$c = \sqrt{x^2 + h_a^2} \tag{4.8}$$

$$\alpha_a = \arctan \frac{h_a}{x} \tag{4.9}$$

$$\alpha_b = \arcsin \frac{h_b}{c} \overset{(4.8)}{=} \arcsin \frac{h_b}{\sqrt{x^2 + h_a^2}} \tag{4.10}$$

$$\alpha = \alpha_a + \alpha_b = \arctan \frac{h_a}{x} + \arcsin \frac{h_b}{\sqrt{x^2 + h_a^2}} \tag{4.11}$$

$$\alpha^* = 180^\circ - \alpha \approx 126,96^\circ \tag{4.12}$$

Die durch Vermessen der Ellbogenmechanik mit der Schiebelehre gemessenen Größen bestimmen den theoretischen Anschlagwinkel beider Richtungen zu einem Betrag von $\frac{\Theta}{2} = 126,96^\circ$.

Bei einmal gesetztem Nullpunkt lassen sich die Anschlagwinkel in Motoreinheiten sehr gut reproduzieren: In zehn Messungen ergab sich für hinteren und vorderen Anschlag ein Mittelwert von 17806 bzw. -17806 Einheiten bei einer maximalen Einzelabweichung von fünf Einheiten. Dies entspricht ca. 100980 Einheiten pro Umdrehung.

Ein Grad kommt also 280,5 Motoreinheiten gleich.

Die Darstellung des Arbeitsraumes. Mit Hilfe des Programmes „Cinderella“ [5], einer Anwendung zur Untersuchung dynamischer Geometrie, war es möglich, anhand der Abmessungen der Armglieder und der Anschlagwinkel den planaren Arbeitsraum graphisch darzustellen und diejenigen Bereiche zu identifizieren, in denen keine oder eine Lösung existiert oder in denen mehrere Lösungen existieren.

Der sich aus den zuvor ermittelten Bewegungsmöglichkeiten ergebende Arbeitsraum entspricht dem Sektor eines Rotationskörpers, der aus der in Abbildung 4.4 dargestellten Fläche entsteht.

4.3.2. Zerlegung des Problems

Laut Angeles wird zum Lösen des „inverse kinematic problem“ (IKP) im allgemeinen auf approximative und numerisch aufwendige Methoden zurückgegriffen. Auch analytische

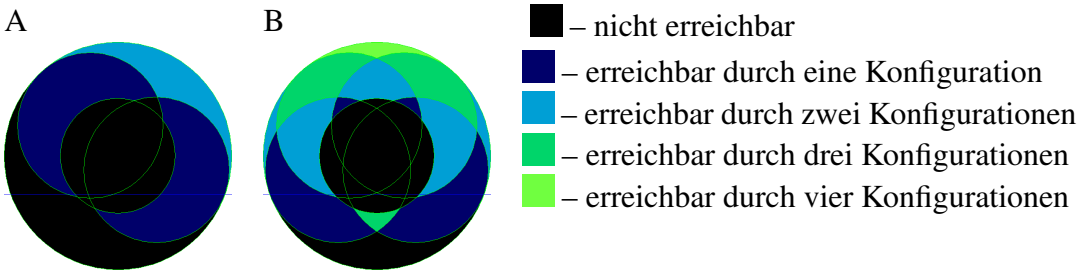

Abbildung 4.4.: Arbeitsraum des Katanas: Die verschiedenfarbigen Flächen symbolisieren die Anzahl der Konfigurationen, durch die jeder Punkt in diesen Flächen erreicht werden kann. Die Flächen sind vertikale Schnitte durch den Arbeitsraum, die die senkrechte Achse des Turmes enthalten. Die blaue waagerechte Linie stellt die Höhe der Montagefläche des Katanas dar.
A Schnitt durch den Sektor, den der begrenzte Öffnungswinkel der Turmdrehung einschränkt.
B Schnitt durch den restlichen Sektor. Die Anzahl der Konfigurationen jedes Punktes ergibt sich aus der Summe von A und seiner Spiegelung, da hier jeweils zwei Turmwinkel in Frage kommen.

Methoden stehen zur Verfügung. Die meisten beschäftigen sich mit dem inversen kinematischen Problem eines sechsgelenkigen Knickarmroboters. Die analytischen Methoden beschäftigen sich mit der Transformation eines Systems aus 14 Gleichungen in bivariate oder univariate Form und Lösung der resultierenden Gleichung. Probleme treten auch hier an Polstellen auf. [1]

Wie weiter oben bereits erwähnt, ist der Katana kein gelenkiger Roboter. Ferner verfügt der Katana über nur vier Gelenke. Dadurch wird das Lösen des IKP zu einem Spezialfall des sechsgelenkigen Knickarmroboters. Es ist aufwendig, in diese Gleichungen die Restriktionen des Katanas einzufügen und anschließend die allgemeinen Lösungsmethoden anzusetzen.

Stattdessen lohnt es sich, die beschränkten Bewegungsmöglichkeiten des vorliegenden Armes genauer zu untersuchen. Hierbei wird deutlich, daß das IKP in drei Unterprobleme zerfällt, die sich getrennt lösen und dann zusammenführen lassen, um zu jedem gegebenen Punkt im Raum sämtliche Konfigurationen wiederzugeben, die der Katana einnehmen kann, um den Zielpunkt zu erreichen:

1. Turmdrehung,
2. planarer Zweiarm-Manipulator und
3. Orientierung des Greifers.

4.3.2.1. Turmdrehung

Der Turm des Armes läßt sich um ca. $348°$, also etwa 97% eines Kreises, drehen. Dadurch sind viele Punkte sowohl durch Hinweisen des Turmes auf den Punkt und Vorwärtsgriff des Armes als auch durch Wegweisen des Turmes vom Punkt und Rückwärtsgriff des Armes erreichbar.

Die Auswahl zwischen beiden Möglichkeiten kann der Benutzer treffen, oder die Entscheidung kann der Bibliothek überlassen werden. Siehe hierzu Abschnitt 4.3.5: „Bewertungsfunktion“.

4.3.2.2. Stellung der Schulter und des Ellenbogens

Gibt man die Ausrichtung des Turmes vor und läßt die Orientierung des Greifers außer acht, so liegen die beiden verbleibenden Gelenkachsen parallel zueinander, also normal zu einer Ebene, in der auch die Turmachse liegt. Diese Ebene ist weiter oben in Abbildung 4.4 A dargestellt.

Die weiteren Betrachtungen in diesem Abschnitt beziehen sich auf die Ebene.

Zur Lösung des Problems wird zunächst die relative Lage des Zielpunktes zur Schulter bestimmt. Ist die Entfernung zwischen beiden größer als die gesamte Armlänge, so ist keine Lösung zu finden. Dies ist nicht nur anschaulich klar, es wird auch durch die allgemein bekannte Dreiecksungleichung belegt, die besagt, daß in euklidischer Geometrie zwei Seiten eines Dreiecks zusammen nicht kürzer sein können als die dritte Seite.

Ist die Bedingung erfüllt, so wird der Ellenbogenwinkel ermittelt, der nötig ist, um den Punkt zu erreichen. Das Knicken des Armes führt zu einer Verkürzung der Strecke zwischen Schulter und Greifer gegenüber dem gestreckten Arm, so daß auch nähergelegene Punkt angefahren werden können. Die benötigten Winkel ergeben sich aus dem Kosinussatz. Siehe Abbildung 4.5.

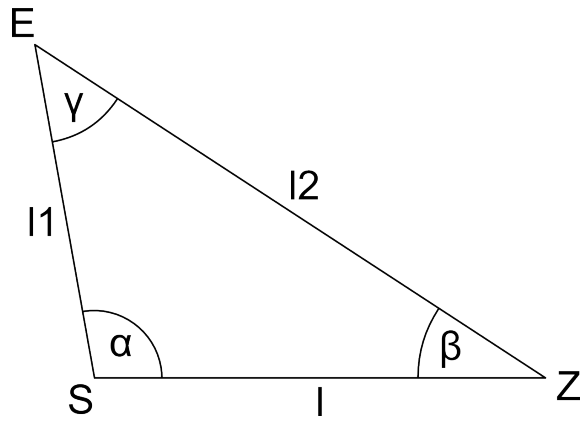

Abbildung 4.5.: Trigonometrie der inversen Kinematik: Der Oberarm $\overline{SE}$ mit der Länge l_1 und der Unterarm $\overline{EZ}$ der Länge l_2 sollen über γ so angewinkelt werden, daß die Strecke $\overline{SZ}$ die Länge l hat. Dies ist die Entfernung des Zielpunktes von der Schulter.

Hierzu wird ohne Beschränkung der Allgemeinheit angenommen, daß gilt: $l < l_2$. Ein dritter Winkel, der dabei der Berechnung durch den Kosinussatz vorenthalten bleibt, läßt sich durch die Winkelsumme bestimmen. Zu diesem Teilproblem gibt es stets zwei potentielle Lösungen (spiegelsymmetrisch an der Strecke SZ aus Abbildung 4.5), die aber jeweils später möglicherweise durch überschrittene Anschlagwinkel ausgeschlossen werden.

Es gilt:

$$\beta = \arccos\left(\frac{l^2 + l_2^2 - l_1^2}{2 \cdot l \cdot l_2}\right) \tag{4.13}$$

und mit $l < l_2$:

$$\gamma = \arccos\left(\frac{l_1^2 + l_2^2 - l^2}{2 \cdot l_1 \cdot l_2}\right) \tag{4.14}$$

und

$$\alpha = 180^\circ - \beta - \gamma \tag{4.15}$$

Das gesamte Dreieck muß nun noch in der Greiferebene rotiert werden, und Schulter- und Ellenbogenwinkel müssen an die im jeweiligen Gelenk verwendeten Winkelkoordinatensysteme angepaßt werden, um die Sollwinkel für beide zu erhalten. Dasselbe geschieht mit der Spiegelung des Dreiecks.

Erreichen die so ermittelten Winkel nicht die Anschlagwinkel der Gelenke, so ist eine mögliche Lösung für dieses Teilproblem gefunden.

4.3.2.3. Orientierung des Greifers

Die Orientierung des Greifers kann als vollständig isoliertes Problem betrachtet werden, da er das letzte Glied der offenen kinematischen Kette ist und nur einen Freiheitsgrad besitzt, der die Lage des Greifpunktes nicht beeinflußt.

So ist es ebenfalls zweckmäßig, diese Orientierung direkt zu übergeben und den Winkel in der Methode `CKatanaKinematic::setXYZ` lediglich auf Gültigkeit zu prüfen.

Leider war es nicht möglich, die Anschlagwinkel für die Greiferdrehung zu bestimmen, da dieses Gelenk defekt ist und sich der Katana zum Zeitpunkt der Abgabe dieser Arbeit noch in der Werkstatt befand.

4.3.3. Zusammenführung der Probleme

Nachdem die geometrisch möglichen Lösungen bestimmt wurden, ohne die Beschränkungen der Gelenke zu beachten, werden diese zunächst miteinander kombiniert. Aus zwei möglichen Turmorientierungen und zwei möglichen Ellbogenknickrichtungen ergeben sich stets vier Lösungen. Für all diese Lösungen werden nun die theoretisch erforderlichen Gelenkwinkel auf Zulässigkeit geprüft. Die nach dieser Filterung verbleibenden Lösungen stehen dem Benutzer zur Verfügung. Die Entscheidung zwischen diesen verschiedenen Armkonfigurationen wird in den nächsten Abschnitten behandelt.

4.3.4. Konfigurationen

Da viele Punkte des Arbeitsraumes durch mehrere Konfigurationen erreicht werden, ist es erforderlich, diese zu unterscheiden und sich für eine zu entscheiden.

Da es pro Punkt maximal vier verschiedene Lösungen gibt, die sich gegenseitig ausschließen, sind 2 bit für eine Beschreibung der diskreten Zustände ausreichend. Hierfür gibt es mehrere Möglichkeiten. Die im folgenden beschriebene wurde aus Gründen der Anschaulichkeit gewählt.

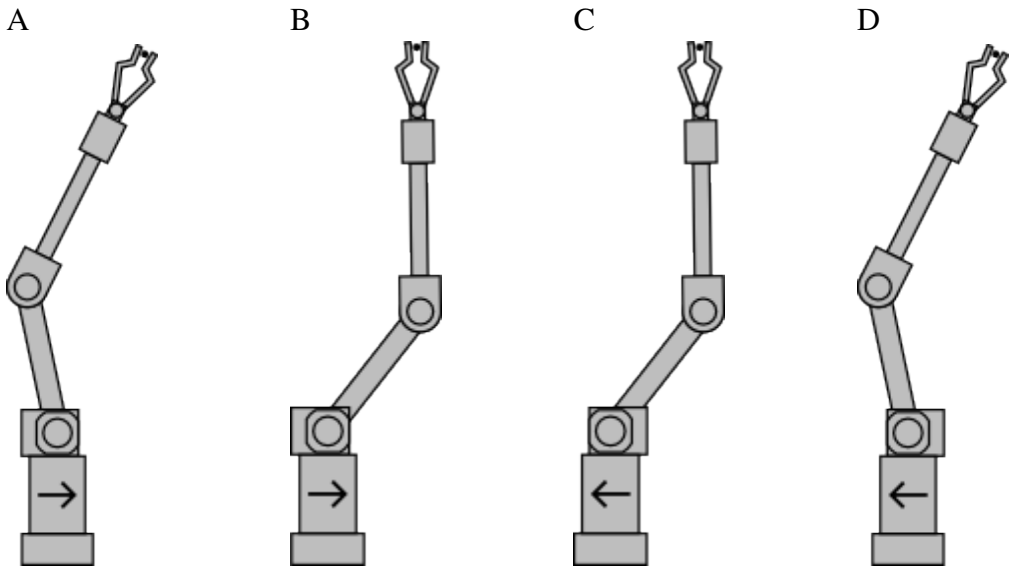

Abbildung 4.6.: Maximal vier Konfigurationen führen zum selben Greifpunkt: Der Punkt liegt in oder entgegen der Vorwärtsrichtung des Turmes, und der Ellenbogen ist aus Sicht des Turmes entweder nach oben bzw. hinten oder nach unten bzw. vorne weisend angewinkelt.
A vorwärts von hinten: Der Turm weist zum Greifpunkt, und der Ellenbogen ist nach hinten angewinkelt.
B vorwärts von vorne: Der Turm weist zum Greifpunkt, und der Ellenbogen ist nach vorne angewinkelt.
C rückwärts von hinten: Der Turm weist vom Greifpunkt fort, und der Ellenbogen ist nach hinten angewinkelt.
D rückwärts von vorne: Der Turm weist vom Greifpunkt fort, und der Ellenbogen ist nach vorne angewinkelt.

- Turm: wendet sich der Turm dem Zielpunkt zu, so heißt die Konfiguration „vorwärts“, ansonsten „rückwärts“ ausgerichtet.
- Ellenbogen: Ist der Ellenbogen gegenüber dem gestreckten Zustand in negative Richtung ausgelenkt, so greift der Arm „von oben“ oder „von hinten“, andernfalls „von unten“ oder „von vorn“.

Diese beiden Unterscheidungen sind voneinander unabhängig und müssen kombiniert werden, wodurch sich die maximale Lösungszahl von $2 \times 2 = 4$ ergibt. Diese sind in Abbildung 4.6 dargestellt.

Es werden bei Aufruf der Methode `CKatanaKinematic::setXYZ` alle Konfigurationen in Erwägung gezogen und geprüft. Für alle gefundenen Lösungen werden die Gelenkstellungen für die Weitergabe an die Stellroutine zurückgegeben.

4.3.5. Bewertungsfunktion

Eine andere Option stellt eine Variante derselben Methode dar, die als Argument die gewünschte Konfiguration entgegennimmt und nach der Prüfung entweder diese ansteuert oder scheitert. Kernbestandteil ist hier die Möglichkeit, die „nächste“ Stellung zu wählen.

Muß unter mehreren möglichen Lösungen eine automatische ausgewählt werden, so ist die Bewertung jedes Kandidaten und eine Auswahl durch Gegenüberstellung nötig. Hierzu ruft `CKatanaKinematic::setXYZ` zu jeder gefundenen Lösung eine Bewertungsfunktion auf, der sie die entsprechenden Gelenkstellungen übermittelt. Die Funktion gibt einen Kostenwert für die zum Erreichen der Zielposition notwendige Bewegung zurück, den `setXYZ` zwischenspeichert, um schließlich den Lösungskandidaten mit dem geringsten beigeordneten Kosten zu bestimmen und diesen anzusteuern.

Für eine zweckmäßige Bewertung wurde eine Funktion eingesetzt, die über alle Gelenke das Maximum der gewichteten Differenz zwischen Ist- und Sollwinkel zurückgibt. Eine Wichtung ist nötig, da die Gelenke sowohl über unterschiedliche Winkelmaße als auch über unterschiedliche typische Winkelgeschwindigkeiten verfügen. Der Anwendung der Maximum-Funktion liegt zugrunde, daß üblicherweise alle Motoren gleichzeitig in Gang gesetzt werden und der größte zu überstreichende Winkel die benötigte Gesamtzeit bestimmt.

Die Vorteile dieser Bewertung kommen vor allem dann zum Tragen, wenn Ursprungs- und Zielpunkt beide hoch über dem Katana in der Nähe der senkrechten Achse liegen, da eine Turmdrehung um beispielsweise 180° viel mehr Zeit in Anspruch nimmt (Faktor in der Größenordnung 10 bis 20), als den Arm über den Pol hinwegzubewegen, obwohl die vom Greifpunkt überstrichene Strecke, die andere Bewertungsfunktionen zur Beurteilung heranziehen, sich zwischen den beiden Fällen um einen Faktor von nur etwa π unterscheidet.

Es seien $\mathbf{a}$ der Vektor der gegenwärtigen Motorpositionen, $\mathbf{g}$ der Vektor der Zielpositionen und $\mathbf{w}$ der Vektor der Wichtungsfaktoren. Alle Vektoren haben der Anzahl der Motoren

entsprechend 5 Elemente. c sei die Bewertungsfunktion, die die Kosten der Bewegung von $\mathbf{a}$ in die Position $\mathbf{g}$ liefert. Dann gilt:

$$c(\mathbf{a}, \mathbf{g}) := \max_{i=1..5} ((\mathbf{g}_i - \mathbf{a}_i) \cdot \mathbf{w}_i) \tag{4.16}$$

Da die Geschwindigkeit einiger Gelenke von der Stellung des Armes abhängt, ist eine genaue Schätzung der Dauer des Bewegungsablaufs nicht ohne weiteres möglich und im Falle dieses Roboterarmes zu aufwendig, da der Katana über keine Geschwindigkeitsregelung verfügt. Dieser Umstand verhindert auch eine Trajektorienplanung, für die dann eine Funktion zur Energieminimierung sinnvoll einzusetzen wäre.

Die Bewertungsfunktion läßt sich dennoch leicht ersetzen, so daß eine Beurteilung nach verschiedenen Maßstäben ermöglicht wird.

5. Zusammenfassung

Der Funktionsumfang des Katana K200 entspricht in weiten Teilen dem in der Dokumentation [4] angegebenen. Wo Funktionen nicht zur Verfügung stehen, ist dies im Quellcode kenntlich gemacht.

Das Greifergelenk konnte mit einfachen Mitteln repariert werden. Das Unterarmgelenk befindet sich zur Reparatur in der Werkstatt.

Es wurde eine Programmierschnittstelle in Form einer kompilierten Bibliothek entwickelt, die es ermöglicht, den Funktionsumfang des Katanas auf einfache Weise zu nutzen. Darüberhinaus kann bei Bedarf der durchweg kommentierte Quellcode angepaßt und erweitert werden, um weitere Anwendungen zu implementieren.

Die entworfene Routine zum Justieren des Armes nach dem Einschalten liefert zuverlässige, reproduzierbare Winkel für die Gelenke Turm, Schulter und Ellenbogen. Die Justageparameter für die beiden übrigen Gelenke sind nach der Reparatur einzustellen.

Der Aufbau des Katanas wurde mittels eines angepaßten Denavit-Hartenberg-Schemas modelliert, um eine Vorwärtskinematik zu implementieren. Das Problem der inversen Kinematik wurde exakt durch trigonometrische Betrachtungen gelöst, die das allgemeine Problem stark vereinfachen.

Zentrales Element der Katana-Klasse ist die inverse Kinematik, die es ermöglicht, nach dem Festlegen eines kartesischen Koordinatensystems jeden beliebigen Punkt im Arbeitsraum des Katanas durch Angabe seiner Koordinaten anzufahren. Ungültige Aufträge werden abgewiesen. Bei Entscheidungsfreiheit über die zum Zielpunkt führenden Konfigurationen findet die Entscheidung aufgrund einer ersetzbaren Bewertungsfunktion statt.

Somit wurden die Hauptziele der Arbeit erreicht. Leider fand die Reparatur des Unterarmgelenkes nicht rechtzeitig statt, so daß manche Funktionen des Katanas nicht zur Verfügung stehen und das Verfassen einer Schritt-für-Schritt-Anleitung noch nicht sinn-

voll wäre. Dies betrifft insbesondere das Unterarmgelenk und den Greifer. Auch über die Aufgabenstellung hinausgehende Untersuchungen der Genauigkeit konnten somit nicht durchgeführt werden.

Diese Arbeiten werden nach dem Einreichen der Studienarbeit und Reparatur des Katanas beendet.

6. Ausblick

Der Umstand, daß es sich in dieser Arbeit um eine Grundlagenarbeit handelt, zieht viele Erweiterungsmöglichkeiten nach sich, insbesondere auf der Software-Ebene. Im Hinblick auf diese Erweiterungen war es wichtig, die Bibliothek für die zukünftige Arbeit nicht nur mit dem Roboter, sondern auch mit dem Quellcode, besonders übersichtlich zu gestalten.

6.1. Software

Die implementierte Schnittstelle ist so konzipiert, daß sie leicht durch weitere Funktionen ergänzt werden kann, so zum Beispiel durch Teach-in von Positionen und Bewegungen, Kollisionserkennung oder Kollisionsvermeidung.

Teach-in von Positionen

Über die bereitgestellten Methoden `getRawMotorPosition(byte motor)` und `setRawMotorPosition(byte motor)` können sämtliche Motorpositionen nacheinander abgefragt bzw. gesetzt werden. Die von `getRawMotorPosition()` gelieferten Werte können in beliebiger Form verarbeitet und gespeichert werden, um sie später mit `setRawMotorPosition()` wieder zu setzen. So können auch ganze Abfolgen von Positionen gespeichert und abgespielt werden.

Teach-in von Bewegungen

Zum Erfassen von Bewegungsabläufen ist eine ständig wiederholte Abfrage der Motorpositionen vonnöten, während der Roboterarm von Hand geführt wird. Die Meßdaten

können nachträglich interpoliert und vereinfacht werden, zum Beispiel in Form von Splines.

Da der Katana seine Motoren nur über Positionsregler steuert, jedoch nicht über Geschwindigkeitsregler verfügt, hängt die tatsächliche Geschwindigkeit der einzelnen Drehbewegungen stark von der aktuellen Haltung des Armes ab. Zudem sind die Motoren nicht in der Lage, das für einige Bewegungen erforderliche Moment aufzubringen. Es ist beispielsweise nicht möglich, den waagerecht ausgestreckten Arm ohne vorheriges Anwinkeln in die Senkrechte zu bringen.

Eine weitere Einschränkung ergibt sich aus der fehlenden Implementierung des 'P'-Befehls, der alle Motorpositionen auf einmal übertragen soll. Mit diesem Befehl wäre es bei nahezu gleichbleibender zu übertragender Datenmenge gegenüber dem einzelnen Setzen der Motoren möglich, auch die Geschwindigkeiten festzulegen. Legte man auch diese einzeln fest, so wäre für dieselbe Funktion die doppelte Datenmenge erforderlich. Zudem ergibt sich beim einzelnen Setzen der Motoren ein minimaler zeitlicher Versatz.

Auf Grund dieser Einschränkungen ist es zweifelhaft, ob der Katana einer Solltrajektorie in akzeptablem Rahmen folgen kann.

Kollisionsvermeidung

Die Klasse `ISSKinematics` mit der zugehörigen Denavit-Hartenberg-Funktionalität ist modular konzipiert, so daß sie auf einfache Weise zur vorherigen Überprüfung anzufahrender Zielpositionen herangezogen werden kann.

Wie in Gleichung 4.2.2: „Implementierung des Modells“ beschrieben wird dieses Konzept bereits in der Klasse `ISSKatana` genutzt, um vor dem Anfahren einer Position zu prüfen, ob diese überhaupt physikalisch erreichbar ist.

Mit Hilfe derselben Methoden können auch verbotene Räumen implementiert werden. Die Vorwärtskinematik bietet die Möglichkeit, zu einer gegebenen Konfiguration, wie sie zum Beispiel durch `setXYZ()` geliefert wird, jede Gelenkposition zu berechnen. So kann zusätzlich zur Greiferposition auch die Ellenbogenposition vor dem Anfahrbefehl überprüft werden.

Eine weitere denkbare Anwendung ist die Überprüfung der tatsächlichen Roboterposition und die Ausgabe eines Alarmes, wenn der Arm in einen kritischen Bereich eindringt.

Kollisionserkennung

Nach dem Senden eines Befehls zum Verfahren des Robotersaarmes kann eine ständige Abfrage der jeweils aktuellen Motorpositionen wie beim Teach-in von Bewegungen dazu verwendet werden, den unbeabsichtigten Stillstand eines oder mehrerer Gelenke zu erkennen und entsprechend darauf zu reagieren, beispielsweise durch das einfrieren sämtlicher Bewegungen, das Erschlaffenlassen aller Gelenke oder ein Zurückverfahren an die letzte bekanntermaßen ungefährliche Position.

Hierbei kann die Möglichkeit, die dynamischen Parameter des Katanas einzustellen, effektiv genutzt werden: Durch Reduzieren der Reglerkonstanten läßt sich die Kraft, mit der der Arm einer Kollision begegnet, begrenzen, wobei der Arm noch immer bei voller Winkelgeschwindigkeit verfahren würde.

6.2. Hardware

In seinem derzeitigen Zustand ist der Katana nicht voll funktionstüchtig. Eingeschränkt werden seine Bewegungsmöglichkeiten insbesondere durch das Gelenk Nr. 4, welches den Unterarm um seine Längsachse dreht. Zudem ist für eine höhere Anfahrgenauigkeit eine präzisere Vermessung des Roboterarmes notwendig.

Unterarmgelenk

Der Roboter befand sich wegen des Unterarmgelenkes bis zur Fertigstellung des schriftlichen Teils dieser Arbeit in der Werkstatt. Sollte die Reparatur gelingen, so steht danach der volle Funktionsumfang zur Verfügung. Andernfalls wird der Arm nicht um seine Längsachse gedreht werden können. In diesem Fall wäre es sinnvoll, den Unterarm so am Ellenbogen zu fixieren, daß der Greifer sich seitwärts öffnet. So könnten zumindest einfache Greifaufgaben durchgeführt werden, die nicht erfordern, daß der gegriffene Gegenstand auf den Kopf gestellt oder seitlich abgelegt wird.

Sensoren

Die in der Hand befindlichen Drucksensoren müssen hin und wieder justiert werden. Die Innenfläche des Greifers ist mit dem starren Teil der Hand nur verklebt, und die Schaumstoffzwischenschicht kann sich davon ablösen. In der Aluminium-Innenfläche ist eine Madenschraube versenkt, die direkt auf den Drucksensor drücken kann. Sie läßt sich so einstellen, daß der Sensor schon bei der kleinsten Berührung ausschlägt, anstatt erst oberhalb eines minimalen Druckes. Möglicherweise läßt sich hierfür eine bessere Hardware-Lösung finden. Denkbar wäre eine gelenkige mechanische Aufhängung der Innenfläche mit einer einstellbaren Verspannung, so daß die Fläche selbst auf den Sensor drückt.

A. Befehlssatz des Katanas

In diesem Abschnitt werden die in der im Rahmen dieser Studienarbeit entwickelten Bibliothek `ISSKatana` umgesetzen Katana-Befehle aufgeführt. Jeder Befehl ist in Form einer Tabelle des folgenden Formats dargestellt:

`Syntax der C-Funktion mit Parameters`		
'TAG'	Kurzbeschreibung des Befehls	
IN	`Parameter 1`	Beschreibung des Eingabeparameters
OUT	`Parameter 2`	Beschreibung des Ausgabeparameters
IN (OPT)	`Parameter 3`	Beschreibung des optionalen Eingabeparameters
Hier stehen gegebenenfalls Anmerkungen zum Befehl.		

Das TAG ist der vom Serial-Zero-Protocol verwendete Buchstabe, der den Befehl auszeichnet. Antwort-Tag ist der entsprechende Kleinbuchstabe nach ASCII. Mit Hilfe dieses Tags können die den C-Methoden zugrundeliegenden Katana-Befehle in der Katana-Dokumentation nechgeschlagen werden.

`void getMasterFirmware(byte *Version, byte *Revision)`		
'B'	Fordert die Version der Firmware an.	
OUT	`Version`	Zeiger auf die Variable, in der die Hauptversionsnummer gespeichert werden soll
OUT	`Revision`	Zeiger auf die Variable, in der die Unterversionsnummer gespeichert werden soll

`void setRawMotorPosition(byte motor, TPosition position, TMotorState state)`		
'C'	Setzt die Zielposition eines Motors.	
IN	`motor`	Die Nummer des anzusprechenden Motors, angefangen mit 0
IN	`position`	Die anzufahrende Position in Winkeleinheiten des entsprechenden Motors
IN (OPT)	`state`	Der Modus des Motors. Sinnvolle, gültige Werte sind `MS_OFF` (das Gelenk läßt locker), `MS_FREEZE` (Der Motor hält das Gelenk in der aktuellen Position fest) und `MS_MOVE` (Der Motor versucht, die Zielposition anzufahren). Wird `state` nicht angegeben, so wird `MS_MOVE` angenommen.

`void setRawActualMotorPosition(byte motor, TPosition position, TMotorState state)`		
'C'	Nimmt für die gegenwärtige Position eines Motors den angegebenen Wert an.	
IN	`motor`	Die Nummer des anzusprechenden Motors, angefangen mit 0
IN	`position`	Die anzunehmende Position in Winkeleinheiten des entsprechenden Motors
IN (OPT)	`state`	Der Modus des Motors. Sinnvolle, gültige Werte sind `MS_OFF` (das Gelenk läßt locker), `MS_FREEZE` (Der Motor hält das Gelenk in der aktuellen Position fest) und `MS_MOVE` (Der Motor versucht, die Zielposition anzufahren). Wird `state` nicht angegeben, so wird `MS_FREEZE` angenommen.

Der Katana arbeitet mit einer gemessenen Position und einer Zielposition. Dieser Befehl bewirkt, daß die aktuelle Position mit dem angegebenen Wert `position` belegt wird. Sollte diese von der Zielposition abweichen, versucht der Katana bei aktiviertem Motor sofort, die Differenz auszugleichen.
Da dieser Befehl das Koordinatensystem des Motors verschiebt, ist der Standardwert für diesen Befehl `MS_FREEZE`.
Es ist nicht empfehlenswert, diesen Befehl bei gleichzeitiger Nutzung von `ISSKinematics` zu verwenden.

`TDynamicData getRawDynamicData(byte motor)`		
'D'	Erfragt Informationen zur Dynamik eines Motors.	
IN	`motor`	Die Nummer des Motors, über den Informationen eingeholt werden, angefangen mit 0
OUT	`return`	Die Funktion liefert eine Struktur vom Typ `TDynamicData` zurück, deren Elemente Angaben zur Position, zur Geschwindigkeit, zum Tastverhältnis des PWM-Modulators und zum Status des Motors enthalten.

Der Rückgabetyp `TDynamicData` hat die folgenden Elemente:

- `TPosition Position` – die Position des Motorsin Winkeleinheiten
- `TVelocity Velocity` – die Geschwindigkeit des Motors in Geschwindigkeitseinheiten
- `TPWMDuty PWMDuty` – das Tastverhältnis des Motors, 70 entspricht 100%
- `TMotorState MotorState` – der Zustand des Motors

Zu den einzelnen Elementen siehe die folgenden Befehle.

`TPosition getRawMotorPosition(byte motor)`		
'D'	Erfragt die gegenwärtige Position eines Motors in Katana-Winkeleinheiten	
IN	`motor`	Die Nummer des Motors, über den Informationen eingeholt werden, angefangen mit 0
OUT	`return`	Die Funktion liefert einen Wert vom Typ `TPosition`, der die Position des Motors in dessen Winkeleinheiten enthält. Jeder Motor benutzt ein eigenes Einheitensystem.

Achtung! Die Einheiten der Katana-Winkelgeschwindigkeiten entsprechen nicht den Katana-Winkeleinheiten pro Sekunde.

`TVelocity getRawMotorVelocity(byte motor)`		
'D'	Erfragt die gegenwärtige Geschwindigkeit eines Motors in Katana-Geschwindigkeitseinheiten.	
IN	`motor`	Die Nummer des Motors, über den Informationen eingeholt werden, angefangen mit 0
OUT	`return`	Die Funktion liefert einen Wert vom Typ `TVelocity`, der die Geschwindigkeit des Motors in dessen Geschwindigkeitseinheiten enthält. Jeder Motor benutzt ein eigenes Geschwindigkeitensystem.

Achtung! Die Einheiten der Katana-Winkelgeschwindigkeiten entsprechen nicht den Katana-Winkeleinheiten pro Sekunde.

TPWMDuty getRawMotorPWMDutyCycle(byte motor)		
'D'	Erfragt das gegenwärtige Tastverhältnis eines Motors in Katana-Tastverhältniseinheiten.	
IN	motor	Die Nummer des Motors, über den Informationen eingeholt werden, angefangen mit 0
OUT	return	Die Funktion liefert einen Wert vom Typ TPWMDuty, der das Tastverhältnis des Motors in Katana-Tastverhältniseinheiten enthält. Dabei entspricht ein Wert von 70 voller Leistung.

Das Tastverhältnis entspricht einer Ganzzahl im Bereich von -70 bis +70 und ist proportional zur Mittel der am Motor anliegenden Spannung. Ein Wert von 70 entspricht 100% der zur Verfügung stehenden Spannung, der Wert -70 steht für 100% der Spannung für die entgegengesetzte Drehrichtung. Eine Null steht für keine Spannung. Das heißt, daß der Motor effektiv nicht in Betrieb ist.

TMotorState getMotorState(byte motor)		
'D'	SHORT	
IN	PARAM	DESC
OUT	PARAM	DESC
IN (OPT)	PARAM	DESC

Der Motorzustand entspricht dem Element MotorState ist vom Typ TMotorState. Er kann die folgenden Werte annehmen:

- MS_MECHSTOP – Der mechanische Anschlag wurde erreicht. Dieser Wert scheint vom Katana K200 nicht verwendet zu werden, da er über keinen Anschlagsensor verfügt.
- MS_MAXPOS – Die maximal anfahrbare Position wurde erreicht. Auch dieser Wert wird nicht verwendet.
- MS_MINPOS – Die minimal anfahrbare Position wurde erreicht. Auch dieser Wert wird nicht verwendet.
- MS_INPOSITION – Das Gelenk befindet sich in der Ziellage.
- MS_NORMAL – Der Motor ist gerade dabei, die Zielposition anzufahren, hat diese aber noch nicht erreicht.
- MS_INVALID – Ein internes Kommunikationsproblem ist im Katana aufgetreten.

void getSensorData(byte data[16])		
'E'	Liefert die aktuellen Meßwerte aller Sensoren	
IN/OUT	data	Feld, in dem die 16 Meßwerte gespeichert werden sollen.

Für Informationen zu den Sensoren siehe [4].

`void setCompensator(byte motor, TCompensator c)`		
'K'	Konfiguriert einen Positionsregler	
IN	`motor`	Nummer des zu konfigurierenden Positionsreglers, angefangen mit 0
IN	`c`	

Der Typ `TCompensator` ist eine Struktur mit den folgenden Komponenten:

- `TPWMDuty MaxPositivePWM` – das maximal zulässige Tastverhältnis für die positive Drehrichtung, maximal 70.
- `TPWMDuty MaxNegativePWM` – das maximal zulässige Tastverhältnis für die negative Drehrichtung, maximal 70.
- `byte kP` – Proportionalanteil des Reglers. Werte zwischen 0 und 63 sind zulässig. Standard ist 16.
- `byte kD` – Differentialanteil des Reglers.
- `byte kSpeed` –

Ein integrierender Anteil ist im K200 nicht implementiert.

`void getMotorState(TMotorState states[5])`		
'N'	Liefert die Status sämtlicher Motoren	
OUT	`states`	Die Elemente dieses Feldes enthalten Die Statusflags der Motoren. Diese sind eine logische Und-Kombination aus den möglichen Werten: - `MS_MECHSTOP` – Das Gelenk hat einen mechanischen Anschlag erreicht (wird vom Katana nicht unterstützt). - `MS_MAXPOS` – Das Gelenk hat den Anschlag in positivem Drehsinn erreicht (wird vom Katana nicht unterstützt). - `MS_MINPOS` – Das Gelenk hat den Anschlag in negativem Drehsinn erreicht (wird vom Katana nicht unterstützt). - `MS_INPOSITION` – Das Gelenk hat die Zielposition bis auf 50 Einheiten genau erreicht. - `MS_NORMAL` – Das Gelenk versucht, die Zielposition zu erreichen. Möglicherweise befindet es sich jedoch nicht in Bewegung, da es zuvor auf `MS_FREEZE` oder `MS_OFF` gesetzt wurde. - `MS_INVALID` – Die Bedeutung ist nicht weiter dokumentiert.

`void setRawDynamicLimits(byte motor, TDynamicLimits limits)`		
'O'	Setzt die dynamischen Grenzen eines Motors	
IN	`motor`	Die Nummer des Motors, angefangen mit 0
IN	`limits`	Eine Struktur mit den dynamischen Grenzen. `TDynamicLimits` enthält die Felder: - `TAcceleration MaxAcceleration` – die maximal zulässige Beschleunigung - `TAcceleration MaxDeceleration` – die maximal zulässige Verzögerung - `TVelocity MaxPositiveVelocity` – die maximal zulässige Geschwindigkeit in positive Richtung - `TVelocity MaxNegativeVelocity` – die maximal zulässige Geschwindigkeit in negative Richtung
Die Einheiten für die Beschleunigung sind nicht dokumentiert. Die Geschwindigkeitseinheiten dieses Befehls stimmen laut Dokumentation [4] nicht mit denen der Geschwindigkeitsmessung überein. Dies sei ein Fehler, der in Zukunft behoben würde.		

`void getSlaveFirmware(byte Slave, byte *Version, byte *SubVersion, byte *Revision, byte *FWType, byte *FWSubType)`		
'V'		
IN	`Slave`	Nummer des Motortreibers, über den Informationen eingeholt werden sollen, angefangen mit 0
OUT	`Version`	Hauptversionsnummer des Slave-Treibers
OUT	`SubVersion`	Unterversionsnummer des Slave-Treibers
OUT	`Revision`	Überarbeitungssnummer des Slave-Treibers
OUT	`FWType`	Art der Firmware: 4 steht für Motoren mit Positionsregler, 5 für experimentelle Treiber.
OUT	`FWSubType`	Unterart der Firmware. Wird vom Katana nicht unterstützt.

`TDynamicLimits getRawDynamicLimits(byte motor)`		
'V'	Fragt die derzeit eingestellten dynamischen Grenzen eines Motors ab.	
IN	`motor`	Nummer des Motors, dessen dynamische Grenzen angefragt werden sollen, angefangen mit 0
OUT	`return`	Eine Struktur mit den dynamischen Grenzen. Für die einzelnen Felder siehe `setRawDynamicLimits`.

`TCompensator getCompensator(byte motor)`		
'V'	Erfragt die Konfiguration eines Positionsreglers	
IN	`motor`	Nummer des Motors, dessen Regler abgefragt werden soll, angefangen mit 0
OUT	`return`	Eine Struktur mit den Parametern des Positionsreglers. Für die einzelnen Felder siehe `setCompensator`.

`void getIDString(char* ID)`		
'Y'	Liefert den Erkennungsstring des Katanas	
OUT	`ID`	Der Erkennungsstring des Katanas. Das vorliegende Modell liefert `Katana 24 exp Katana 6M Arm, Neuronics AG $Revision: 40.0 $UX`
Dies ist dieselbe Zeichenkette, die der Katana nach dem Einschalten oder nach einem Rücksetzen ungefragt sendet.		

`bool recvEcho()`		
'Z'	Testet die Kommunikation zwischen Host und Katana	
OUT	`return`	Liefert `true`, wenn die Kommunikation erfolgt ist, andernfalls `false`.

B. Abkürzungsverzeichnis

IKP „inverse kinematic problem“

ASCII American Standard Code for Information Interchange

KNI Katana Native Interface

SZP Serial-Zero-Protocol

DH Denavit-Hartenberg

C. Literaturverzeichnis

[1] ANGELES, JORGE: *Fundamentals of Robotic Mechanical Systems*. Springer, Second Auflage, 2003.

[2] DENAVIT, J. und R.S. HARTENBERG: *A kinematic notation for lower-pair mechanisms based on matrices*. ASME Journal Applied Mechanics, 22:215–221, 1955.

[3] GEUE, S., S. SCHUSTER und T. WALTER: *Ansteuerung des Katana Roboterarms*. Diplomarbeit, Otto-von-Guericke-Universität Magdeburg, 2004.

[4] NEURONICS-AG: *Katana user Manual and Technical Description*. MS Word document, 2002. Version 1.30.

[5] RICHTER-GEBERT, J. und U.H. KORTENKAMP: *Die interaktive Geometrie-Software Cinderella*. Springer, Heidelberg, 1999. URL: http://cinderella.de/.

[6] WLOKA, DIETER W.: *Robotersysteme*, Band 1. Springer, 1992.