



Kostenoptimale Industrieroboterprogrammierung am Beispiel
von Tricept-Robotern der Firmen Comau und SEF

Studienarbeit

Torben Beisch

**Universität der Bundeswehr
Fachbereich Maschinenbau
Laboratorium Fertigungstechnik**

Prof. Dr.-Ing. J. Wulfsberg

Hamburg
Dezember 2001

Ehrenworterklärung

Hiermit erkläre ich, daß die vorliegende Arbeit selbständig und nur unter Zuhilfenahme der angegebenen Mittel angefertigt wurde. Die Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt und ist nicht veröffentlicht worden.

Hamburg, Dezember 2001

Torben Beisch

Inhaltsverzeichnis

Inhaltsverzeichnis	III
Abkürzungsverzeichnis.....	IV
1. Einleitung	1
1.1 Thema.....	1
1.1 Aufgabenstellung	1
1.2 Vorgehensweise	2
2. Zielsetzung der Roboter-Offlineprogrammierung.....	3
3. Stand des Wissens und der Technik	6
4. Kriterienbildung und Vergleich verschiedener OLP-Systeme.....	8
5. Kriteriengewichtung und Auswahl eines OLP-Systems.....	12
6. Generierung des geforderten Robotermodells im OLP-System.....	15
6.1 Geometrische Modellierung	15
6.1.1 Überprüfung der geometrischen Einzelemente.....	15
6.1.2 Korrektur einzelner Bauteile.....	16
6.2 Kinematische Modellierung	18
6.2.1 Anpassung und Änderung der kinematischen Gleichungen.....	18
6.2.2 Implementierung der Kinematik in das OLP-System	20
6.3 Test der Simulation	21
6.3.1 Abgleich mit realen Achs- und Raumkoordinaten.....	21
7. Übertragung des TR600-Modells auf den SRT60.....	23
7.1 Anpassung der geometrischen Einzelemente an den SRT60	23
7.2 Anpassung der kinematischen Daten an den SRT60	24
7.3 Test der Simulation mit dem SRT60	25
8. Zusammenfassung	27
8.1 Ergebnis.....	27
8.2 Ausblick	27
9. Anhang	29
9.1 Struktogramm	29
9.3 Abbildungsverzeichnis.....	29
9.4 Literaturverzeichnis	30

9.5 CD mit Programm und dieser Arbeit als Word-Dokument.....	30
--	----

Abkürzungsverzeichnis

CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
DLL	Dynamic Link Library (Bibliotheksdatei)
GUI	Graphical User Interface
IGRIP	Interactive Graphics Robot Instruction Program
IR	Industrieroboter
OLP-System	Offline-Programmier-System
SRT60	Tricept-Roboter der Firma SEF
TCP	Tool Center Point
TR600	Tricept-Roboter der Firma Neos

1. Einleitung

1.1 Thema

Der Aufwand der Programmierung von Industrierobotern (IR) kann durch den Übergang von der Online-Programmierung zur Offline-Programmierung, auch indirekte Programmierung genannt, deutlich verringert werden, da der Roboter selbst nicht zur Programmierung benötigt wird. Die Offline-Programmierung kann dabei gleichzeitig zu einer Steigerung der Genauigkeit führen. Hierzu gibt es bereits eine Reihe von verschiedenen Offline-Programmier- und Simulationssystemen (OLP-Systemen) auf dem Markt, die sich aber hinsichtlich ihres Preises, der Leistung, dem Umfang und der Kompatibilität unterscheiden. Aus diesem Grunde wurde im Laboratorium für Fertigungstechnik an der Universität der Bundeswehr die Idee entwickelt, ein kostengünstiges und anwenderfreundliches OLP-System speziell für Tricept-Roboter zu schaffen, welches auf ein PC-basiertes Simulationssystem aufbaut und universell einsetzbar ist, d. h. sowohl Visualisierung als auch Simulation und Programmierung ermöglicht.

Im Rahmen dieser Studienarbeit soll die Auswahl und Ergänzung eines geeigneten Simulationssystems sowie der Test desselben erfolgen. So soll letztendlich eine Anwendung entstehen, mit der verschiedene Tricept-Roboter kostengünstig und hochgenau programmiert werden können.

1.1 Aufgabenstellung

Die Aufgabenstellung der Studienarbeit beinhaltet folgende Punkte:

- Bildung von Kriterien zum Vergleich von OLP-Systemen
- Kriteriengewichtung und Vergleich verschiedener OLP-Systeme
- Auswahl eines OLP-Systems anhand der gebildeten Kriterien
- Prüfung und Korrektur des vom Programm zur Verfügung gestellten geometrischen Modells
- Prüfung und Anpassung des vorgegebenen kinematischen Modells
- Abgleich mit realen Achs- und Raumkoordinaten
- Übertragung des TR600-Modells auf den SRT60
- Abschließende Verifikation und Validierung

1.2 Vorgehensweise

Gemäß der Aufgabenstellung war es bei dieser Arbeit zunächst notwendig, die entscheidenden Kriterien zur Auswahl eines OLP-Systems festzulegen und diese zu gewichten. Im zweiten Schritt mußten Daten über verschiedene OLP-Systeme gesammelt und diese den gebildeten Kriterien zugeordnet werden. Hierbei bestand das Problem, daß die verschiedenen Entwickler der Systeme ihre speziellen Stärken hervorheben und Schwächen erst bei genauer Prüfung auftraten.

Über die Kriterien und deren Gewichtung konnte eine Entscheidung bezüglich der Programmauswahl getroffen werden. Hierbei spielten vor allem die Punkte Preis, Kompatibilität und PC-Basis eine entscheidende Rolle.

Nach der Auswahl des OLP-Systems mußte das bestehende geometrische Modell überprüft und korrigiert werden. Anschließend mußte das kinematische Modell an das OLP-System angepaßt und implementiert werden. Dabei wurde besonders Wert darauf gelegt, daß das OLP-System den üblichen Konventionen entspricht und somit universell kompatibel ist.

Nach dem Test der Genauigkeit wurde ein zweiter Tricept-Roboter in dem System erstellt, um die Kompatibilität des OLP-Systems sicherzustellen und zu testen.

2. Zielsetzung der Roboter-Offlineprogrammierung

Es ist sicherlich unbestritten, daß das unternehmerische Ziel das Streben nach materiellem Gewinn ist. Damit ein Unternehmen dieses Ziel möglichst gut erfüllt und somit am Markt wettbewerbsfähig ist und bleibt, muß es zu möglichst geringen Kosten seine Erzeugnisse produzieren. Dies war unter anderem ein Grund dafür, daß in der Industrie in den letzten 20 Jahren die schon viel länger vorhandene Automatisierung durch Roboter erweitert wurde und den Menschen mehr und mehr aus seiner Rolle des Arbeiters gedrängt hat. Weitere Vorteile eines Roboters gegenüber einem menschlichen Arbeiter zeigen sich in den folgenden Faktoren:

- höhere Wiederhol- und Fertigungsgenauigkeit
- kontinuierlicher Einsatz
- flexible Einsatzfähigkeit durch freie Programmierung
- Ausführen von Arbeiten, die für den Menschen gesundheitsschädlich oder gefährlich sind
- Aufbringen höherer Kräfte

Die Programmierung dieser Roboter erfolgt klassisch direkt in seiner realen Fertigungsumgebung an realen Werkstücken. Dieses Programmierverfahren, auch Teach-In-Verfahren genannt, ist allerdings aufwendig und die Genauigkeit, die erreicht werden kann, ist von den Fähigkeiten des Arbeiters abhängig, der den Roboter programmiert. Bei geringer Änderung des Werkstücks ist häufig eine komplette, zeitaufwendige Neuprogrammierung des Roboters notwendig. Des weiteren ist die theoretische Genauigkeit eines Industrieroboters deutlich höher, und daher wird hier oft eine vorhandene Ressource nicht vollends ausgenutzt.

An dieser Stelle setzt die Offline-Programmierung an. In das OLP-System lädt man erstens das CAD-Modell des zu bearbeitenden Werkstücks, zweitens das CAD-Modell des Roboters mit seinen kinematischen Eigenschaften und drittens die Umgebung des Roboters. Nun ist es möglich, den Bahnverlauf des Roboters direkt an dem CAD-Modell des Werkstücks zu programmieren. Dabei können gleichzeitig der Aktionsradius des jeweiligen Roboters sowie eventuelle Kollisionen des Roboters mit seiner Umgebung oder dem Werkstück automatisch überprüft und bei Bedarf korrigiert werden. In dem OLP-System kann anschließend ein Roboterprogramm erstellt werden, welches nur noch in die Sprache des jeweils zu programmierenden Roboters übersetzt werden muß, um dann in die Robotersteuerung geladen zu werden.

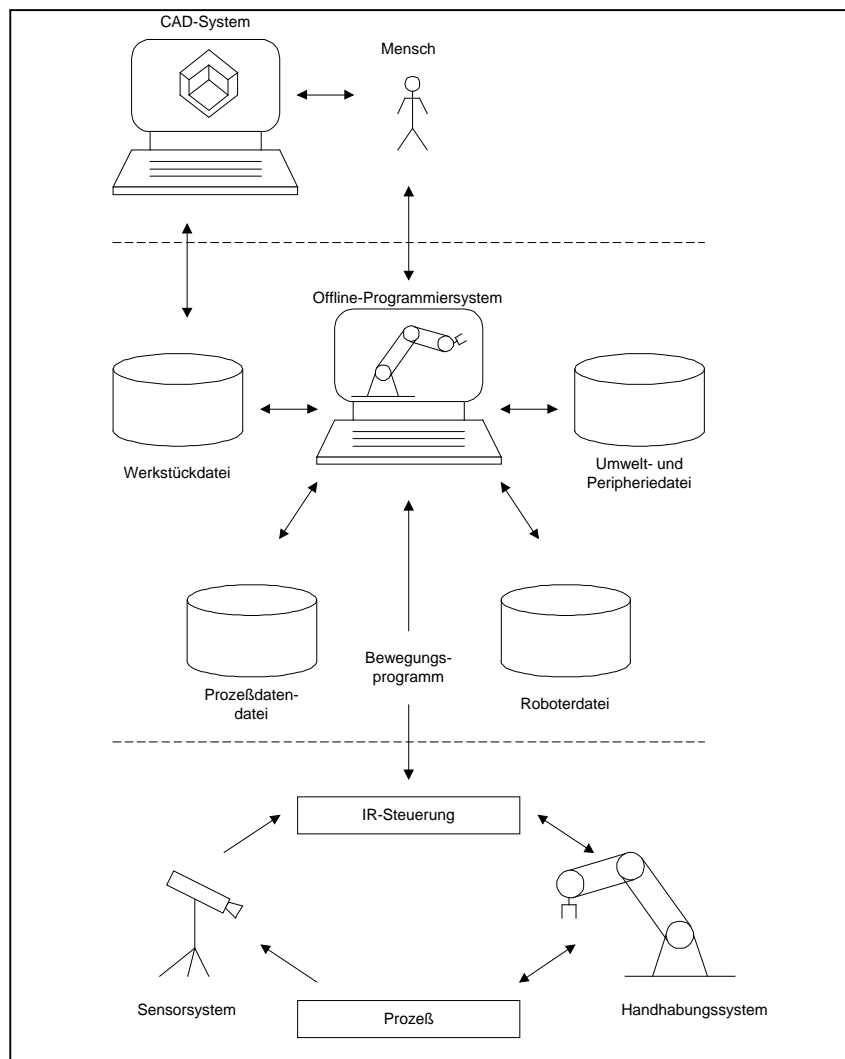


Abbildung 1 – Struktur eines CAD/CAM-gestützten Offline-Programmiersystems [1]

Hierbei treten jedoch teilweise deutliche Unterschiede zwischen der simulierten und der realen IR-Applikation auf, die sich auf folgende Einzeleinflüsse zurückführen lassen:

- ungenaue Werkstücklage (Position der Vorrichtung)
- Abweichungen der Werkstückformen (z.B. durch Verzug)
- Falsche Werkzeugdaten (ungenaue Vermessung, Verschleiß)
- Roboterungenauigkeiten (schlechte Absolutgenauigkeit)
- Übersetzungsfehler bei Programmerstellung

Für das Erstellen eines korrekten Programms ist es u.a. nötig, daß die kinematische Berechnung exakt dem Roboter entspricht, da man hier ein Modell direkt in die Wirklichkeit überträgt. Dabei ergibt sich allerdings ein Problem, das darin liegt, daß der reale Roboter nicht exakt seinem CAD-Modell entspricht, da es gewisse Fertigungstoleranzen gibt und der

Roboter auch nicht frei von Verschleiß oder Beschädigungen ist, die etwa durch Kollisionen hervorgerufen wurden. Aus diesem Grund besteht die Notwendigkeit, daß die Diskrepanzen des Roboters zwischen der Wirklichkeit und dem Modell mittels eines Kalibriersystems erkannt werden können. Das OLP-System muß nun in der Lage sein, diese geometrischen Unterschiede mittels Parametrisierung sowohl in das CAD-Modell als auch in die kinematischen Berechnungen einfließen zu lassen. So kann das Modell jederzeit nach einer Kalibrierung wieder an die Realität angepaßt werden. Dabei erhält man stets eine hochgenaue Anwendung, die nur noch durch die Wiederholgenauigkeit der Aktuatoren des Roboters beschränkt ist. Dadurch ist es vorstellbar, einen Roboter für Arbeiten im Genauigkeitsbereich von wenigen Zehntel Millimetern einzusetzen.

Der Zeitaufwand in der Fertigungszelle ist bei einer solchen Programmierung deutlich geringer, vor allem, wenn es häufig zu Änderungen des Programms kommt. Somit werden auch teure Stillstandszeiten des Roboters stark reduziert, da eine neue Programmierung bereits vorbereitet werden kann, während der Roboter, der dieses Programm später ausführen soll, noch andere Arbeiten verrichtet. Auch der Austausch eines Roboters bedarf lediglich einer Kalibrierung und somit Parametrisierung des OLP-Systems sowie der Generierung eines geänderten Verfahrsprogramms ohne weiteren Aufwand vor Ort.

So wird nicht nur die Genauigkeit und somit auch die Qualität herkömmlicher Produkte erhöht und der Zeitaufwand deutlich verringert, auf diese Art wird auch ein wesentlich größeres Einsatzspektrum des Roboters geschaffen. So können weitere Arbeiten von Robotern ausgeführt werden, in die die oben genannten Vorteile ebenfalls einfließen.

Anhand der aufgezeigten Punkte ist zu erkennen, daß sich die Effizienz und das Spektrum der automatisierten Produktion durch das Ersetzen der Online-Programmierung durch die Offline-Programmierung von Industrierobotern deutlich erhöhen läßt, wenn die genannten Kalibrierungen und Parametrisierungen stattfinden.

3. Stand des Wissens und der Technik

Die Offline-Programmierung von Robotern ermöglicht die rechnergestützte Programmierung zeit- und ortsunabhängig von dem zu programmierenden Roboter. Ziel ist die Generierung des Roboterprogramms parallel zum Robotereinsatz, so daß lange Stillstandszeiten vermieden werden. Dazu gibt es drei verschiedene Varianten der Offline-Programmierung. [2]

Erstens die textuelle Programmierung: Seit der Mitte der 70er Jahre wurden Roboterprogrammiersprachen entwickelt, in denen ein Roboterprogramm textuell beschrieben wurde. Diese Programme können mittels eines Computers bearbeitet werden. Allerdings war es lediglich möglich, bestehende Programme zu verändern oder neue Programmgerüste zu erstellen, da die Generierung eines kompletten Programms inklusive Raumkoordinaten einen hohen manuellen numerischen Rechenaufwand sowie gutes räumliches Vorstellungsvermögen des Programmierers erforderte. So wurden die Raumkoordinaten mittels Teach-In-Verfahren direkt in der Roboterzelle eingefügt. Dieses Verfahren verringerte zwar die Stillstandszeit der Roboter, allerdings noch nicht in einem akzeptablen Rahmen. Dafür waren die Ansprüche an Hard- und Software der Zeit angemessen.

Das zweite Verfahren ist eine CAD-basierte Makroprogrammierung:

Ein Makro ist eine Folge von Befehlen, die abgespeichert und jeweils bei Bedarf aufgerufen werden kann. Bei diesem Verfahren wird ein CAD-Programm mit der Makroprogrammierung verbunden. So können Befehlsketten schnell und bei exakter Makroerstellung fehlerfrei in ein Programm eingebettet werden. Dies gilt allerdings nur für eine begrenzte Zahl von häufig auftretenden Befehlsketten. Der Rest der Programmierung muß online in der Roboterzelle erfolgen. Nachteil dieser Programmierung ist eine geringe Flexibilität bei Änderungen des Werkstücks. Außerdem ist diese Makroprogrammierung relativ unübersichtlich und stellt daher auch hohe Ansprüche an den Programmierer.

Das dritte Verfahren, auf dem in dieser Arbeit das Hauptaugenmerk liegt, ist die grafisch-interaktive Programmierung. Dieses Verfahren hat ebenfalls eine CAD-basierte Bedienoberfläche, das OLP-System ermöglicht dem Benutzer aber die Modellierung des Roboters, seiner Kinematik, der Roboterzelle sowie des Werkstücks. Die einzelnen CAD-Elemente können aus anderen CAD-Programmen importiert werden. Dabei gibt es zwei wesentliche Vorteile. Erstens werden die Arbeitspunkte nicht textuell, sondern grafisch-interaktiv erzeugt. Zweitens ist auch eine Simulation des Programmablaufs möglich, in der festgestellt werden kann, ob etwaige Kollisionen des Roboters mit dem Werkstück oder

anderen Elementen seiner Umwelt vorliegen. In neueren Systemen besteht neben der Simulation der Kinematik auch die Möglichkeit der Simulation der Dynamik und somit des kompletten Arbeitsablaufes. Bei diesem Verfahren treten allerdings verschiedene Probleme auf. Während die Fertigungsumgebung ausreichend exakt dargestellt werden kann, entstehen Probleme bei der exakten Darstellung der Werkstücklage und des Roboters, da dieser Fertigungstoleranzen und Verschleißerscheinungen aufweist. Ein weiteres Problem ist die Berechnung der inversen Kinematik, die absolut exakt dem Roboter entsprechen muß. Diese wird ebenfalls durch kinematische Ungenauigkeiten verfälscht. Da ein Programm, das mit einem OLP-System erstellt wurde, häufig nicht in der erforderlichen Robotersprache vorliegt, muß dieses übersetzt werden, wodurch weitere Fehler entstehen können.

Dieses Verfahren, das einerseits sehr komfortabel ist und andererseits die Stillstandszeiten des Roboters minimiert, setzte allerdings über eine lange Zeit hinweg entweder einen Hochleistungs-PC oder eine Workstation voraus. Dazu wurde meistens auch ein speziell geschulter Programmierer benötigt. Durch die immer schnellere Entwicklung einfacher PCs halten mittlerweile OLP-Systeme für einen Standard-PC Einzug. Diese sind kostengünstiger und der Anspruch an einen Programmierer wird durch eine Steigerung des Komforts und den Umstieg auf eine intuitive Bedienung deutlich verringert. Damit wird diese Form der Programmierung auch für kleinere Unternehmen mit wenigen oder nur einem Roboter immer interessanter.

4. Kriterienbildung und Vergleich verschiedener OLP-Systeme

Um aus der Vielzahl von OLP-Systemen eines auszuwählen, war es notwendig Kriterien zu bilden, um die verschiedenen OLP-Systeme zu vergleichen. Es wurden sechs Hauptkriterien gebildet, die sich teilweise noch untergliedern. Diese Kriterien sollten bei allen Systemen vorhandene Merkmale beschreiben, so daß ein Vergleich überhaupt erst möglich wird.

Die ersten zwei Kriterien sind am objektivsten und einfach festzustellen.

Das erste Kriterium ist der Preis des Systems, da in der heutigen Zeit der Drang zur Kostenersparnis innerhalb von Unternehmen immer stärker wächst. Der Preisvergleich ist wichtig, da die einzelnen Systeme nicht zu einem homogenen Preis angeboten werden.

Das zweite Kriterium ist die Betriebssystembasis. Die Betriebssystembasis sagt indirekt auch etwas über die Anschaffungs- und Folgekosten des Systems aus. Es gibt im wesentlichen zwei Möglichkeiten für das Betriebssystem, auf dem das System läuft. Dies ist zum einen Windows und zum anderen Unix. Daraus ergibt sich, welche Art von Computer benötigt wird, um das System zu betreiben. Ein Unix-Betriebssystem setzt eine Workstation voraus, die ein Vielfaches von einem PC kostet, auf dem Windows als Betriebssystem läuft. Auch die Bedien- und Wartungskosten einer Workstation sind deutlich höher als die eines PCs. Der Grund, ein Unix-basiertes System zu wählen, ist entweder eine Frage der Performance oder ein Festhalten an vergangenen Standards.

Das dritte Kriterium ist die Kompatibilität des OLP-Systems. Zu diesem Kriterium gibt es zwei Unterpunkte: Zum einen die Kompatibilität zu verschiedenen Robotern und zum anderen die Kompatibilität zu CAD-Programmen. Einige Roboterhersteller bieten auf ihr Produkt zugeschnittene OLP-Systeme an, die häufig nur eine mangelnde Kompatibilität zu Robotern anderer Hersteller aufweisen. Die Kompatibilität zu anderen Robotern ist allerdings wichtig, da eine Firma durchaus verschiedene Robotertypen einsetzen kann, und der Aufwand verringert wird, wenn ein OLP-System für alle Roboter vorhanden ist. Die Kompatibilität zu CAD-Programmen ist wichtig, da die OLP-Systeme häufig nur begrenzte Möglichkeiten haben, komplexe CAD-Objekte zu erstellen. Außerdem werden Werkstücke von Konstrukteuren in komplexen CAD-Programmen erstellt, und es muß die Möglichkeit bestehen, diese Objekte zu importieren. Dabei dürfen allerdings keine Konvertierungsfehler auftreten, da die Anwendung sonst ungenau wird.

Das vierte Kriterium ist die Erweiterbarkeit des OLP-Systems. Auch dieses Kriterium gliedert sich in zwei Unterpunkte. Einerseits spielt die Möglichkeit des Einbindens eigener

Berechnungen des inversen kinematischen Problems eine wichtige Rolle, andererseits muß ein Roboter in dem OLP-System parametrisierbar sein, um die Realität detailgetreu nachbilden zu können. Das Einbinden eigener Berechnungen ist wichtig, da die Beschreibung der inversen Kinematik zum einen nicht zwangsläufig in einem OLP-System implementiert ist, zum anderen kann es vorkommen, daß die Notwendigkeit besteht, eine eigene Berechnung einzuführen, weil die vorgegebene nicht exakt genug ist oder neue Erkenntnisse zu einer Änderung der Berechnung führen. Die Parametrisierbarkeit des Robotermodells dient der korrekten Visualisierung des Roboters, von der zwar die korrekte Achswinkelberechnung nicht abhängt, jedoch die Kollisionserkennung.

Das fünfte Kriterium umfaßt die Fähigkeiten des OLP-Systems. Daher umfaßt dieses Kriterium eine größere Zahl an Punkten.

Erstens muß geprüft werden, in wie weit das Programm in der Lage ist, die CAD-Objekte richtig darzustellen, weil nur mittels korrekter Darstellung eine einfache Bedienung gewährleistet wird. Wenn der Benutzer sich Gedanken darüber machen muß, wie die Darstellung in dem OLP-System in der Wirklichkeit aussieht, besteht die Gefahr, daß sich Fehler einschleichen.

Zweitens ist es zweckmäßig, wenn die Berechnungen, die in dem Programm angestellt werden, weitestgehend korrekt sind. Dies ist zwar nicht unbedingt notwendig, wenn man eigene Berechnungen implementiert, jedoch kann man sich selbst so überprüfen, da die Berechnungen wenigsten auf einen Millimeter übereinstimmen sollten.

Drittens ist die mitgelieferte Bibliothek an Robotern, Werkzeugen und Umgebungsmodellen sowie deren Korrektheit interessant. Häufig werden sicherlich eigene Modelle benutzt, jedoch ist eine umfangreiche Bibliothek eine sinnvolle Ergänzung.

Viertens muß die CAD-Fähigkeit so weit gegeben sein, daß importierte CAD-Objekte leicht kombiniert und angeordnet werden können, um einen reibungsfreien Modellaufbau zu gewährleisten. Das Erstellen einfacher CAD-Objekte sowie geringfügige Änderungen bestehender Objekte sollten gegeben sein, damit nicht immer ein zusätzliches CAD-Programm benötigt wird. Die Möglichkeit komplexe Objekte zu erstellen, ist nicht von großer Bedeutung, da es bereits Programme gibt, die dies als Kernkompetenz haben. Das Übertragen dieser gesamten Bandbreite an Möglichkeiten im CAD Bereich wäre zu aufwendig und nur von einem Hersteller von CAD-Programmen realisierbar. Hier muß eine Aufteilung in CAD-Programm und OLP-System erfolgen.

Fünftens ist es besonders bei dieser Arbeit wünschenswert, daß das OLP-System in der Lage ist, eine geschlossene kinematische Kette zu realisieren, wie sie für einen Tricept-Roboter benötigt wird. Dies ist keine absolut notwendige Voraussetzung, aber die Berechnung vereinfacht sich, wenn eine geschlossene Kinematik realisierbar ist. Ansonsten können sich mehr Fehler in der Berechnung und Umsetzung ergeben.

Das sechste und letzte Kriterium ist Optik und Bedienung des OLP-Systems. Wünschenswert ist hier eine weitestgehend intuitive Bedienung, die allerdings bei einem so spezifischen Programm immer noch die Einarbeitung in die Materie selbst erfordert. Eng verbunden mit der intuitiven Bedienung ist auf der einen Seite das Design des OLP-System, die Anordnung der Schaltflächen sowie das Gesamterscheinungsbild. Auf der anderen Seite spielt der Bedienkomfort eine wesentliche Rolle. Hier ist zu prüfen, ob das jeweilige Programm dem gängigen GUI-Design entspricht. Des weiteren spielen Standardprogrammelemente eine wichtige Rolle, um die Bedienung eines Programms schnell zu erlernen.

Preis	Basissystem	Kompati- bilität	Erweiterbar- keit	Eigene Fähigkeiten	Optik und Bedienung
		zu Robotern	durch eigene Berechnungen	Darstellung	intuitive Bedienung
		zu CAD- Programmen	durch Para- metrisierbarkeit	Berechnung	Design
				Bibliltheken	Bedien- komfort
				CAD Mög- lichkeiten	
				geschlossene Kinematik	

Tabelle 1 – Kriterienübersicht

Im Folgenden sollen sechs verschiedene OLP-Systeme kurz vorgestellt werden und eine Bewertung dieser anhand der hier genannten Kriterien erfolgen.

Die sechs Programme sind:

Programm	Firma
Easy-Rob	Easy-Rob
eM-Workspace (RobCad)	Tecnomatrix
IGRIP	Delmia
KR-SIM	Kuka Roboter GmbH
Robot-Studio	ABB
Workspace	Eurobtec

Tabelle 2 – Übersicht über OLP-Systeme

Die Spannweite von Einsatzmöglichkeiten reicht hier vom einfachen PC-System zur Darstellung lediglich eines Roboters bis zu Hochleistungs-Programmsystemen, die eine komplette Fabrikplanung und Fertigungsliniensimulation erlauben, dafür aber auch eine High-End-Workstation in Preisregionen bis 70.000 € erfordern.

Die folgende Tabelle ist eine Zusammenfassung der Bewertung der einzelnen Programme. Dabei werden für die letzten vier Kriterien Bewertungen von 1 (mangelhaft) bis 6 (sehr gut) vergeben:

Programm	Preis	Basis-System	Kompatibilität	Erweiterbarkeit	Eigene Fähigkeiten	Optik und Bedienung
Easy-Rob	2.375 €	Windows	6	6	5	3
eM-Workspace	80.000 €	Unix	5	5	6	5
IGRIP	100.000 €	Windows Unix	5	5	6	5
KR-SIM	15.000 €	Kuka Terminal	1	1	6	5
Robot-Studio	12.000 €	Windows	1	1	6	4
Work-space	20.000 €	Windows	3	3	4	5

Tabelle 3 – Bewertungen der OLP-Systeme

5. Kriteriengewichtung und Auswahl eines OLP-Systems

Um eine Auswahl eines OLP-Systems treffen zu können, müssen die einzelnen Kriterien gewichtet werden. Anhand der Gewichtung ergibt sich für jedes Programm eine Maßzahl, durch die eines der Programme ausgewählt wird. Dazu sollen für jedes Kriterium Punkte vergeben werden.

Das Kriterium „Basissystem“ geht nicht mit in diese Maßzahl ein, da dieses Kriterium allen anderen vorgeschaltet ist. Die Einschränkung, ein Programm lediglich auf einer Workstation oder einem speziell angefertigten Computer laufen lassen zu können, schließt dieses Programm vollkommen in dieser Arbeit aus.

Das Kriterium Preis muß nun noch in eine Maßzahl umgesetzt werden. Dazu ist es zweckmäßig ebenfalls Noten für den Preis zu vergeben. Jede Note soll hier 15.000 € umfassen.

Nachdem für alle Kriterien eine Maßzahl gefunden wurde, muß abgewogen werden, ob es sinnvoll ist, den Kriterien verschiedene Gewichte beizumessen. Dies ist sinnvoll, da es eine Staffelung in der Wichtigkeit der Kriterien gibt.

Der Preis soll mit dem zweifachen Gewicht einfließen, da dieser durchaus die Auswahl beeinflusst, aber nicht die Hauptentscheidungsgrundlage bildet. Diese liegt sicherlich bei den Kriterien Kompatibilität und Erweiterbarkeit. Aus diesem Grund werden diese Kriterien vierfach gewichtet. Die eigenen Fähigkeiten des Programms sind wie der Preis wichtig, aber nicht Hauptgrund und werden daher ebenfalls zweifach gewichtet. Das letzte Kriterium, Optik und Bedienung, ist von geringerer Bedeutung für die Auswahl eines OLP-Systems. Daher wird dieses Kriterium einfach gewichtet.

So ergeben sich die folgenden Werte:

Programm	Preis (zweifach)	Kompatibilität (vierfach)	Erweiterbarkeit (vierfach)	Eigene Fähigkeiten (zweifach)	Optik und Bedienung (einfach)	Summe
Easy-Rob	12	24	24	20	12	92
eM- Workspace	4	20	20	24	20	88
IGRIP	2	20	20	24	20	86
KR-SIM	12	4	4	24	20	64
Robot- Studio	12	4	4	24	16	60
Work- space	10	12	12	16	20	70

Tabelle 4 – Gewichtete Bewertung der OLP-Systeme

Damit steht die Auswahl fest. Das Programm Easy-Rob wird nun zur Simulation zweier Tricept-Roboter getestet.

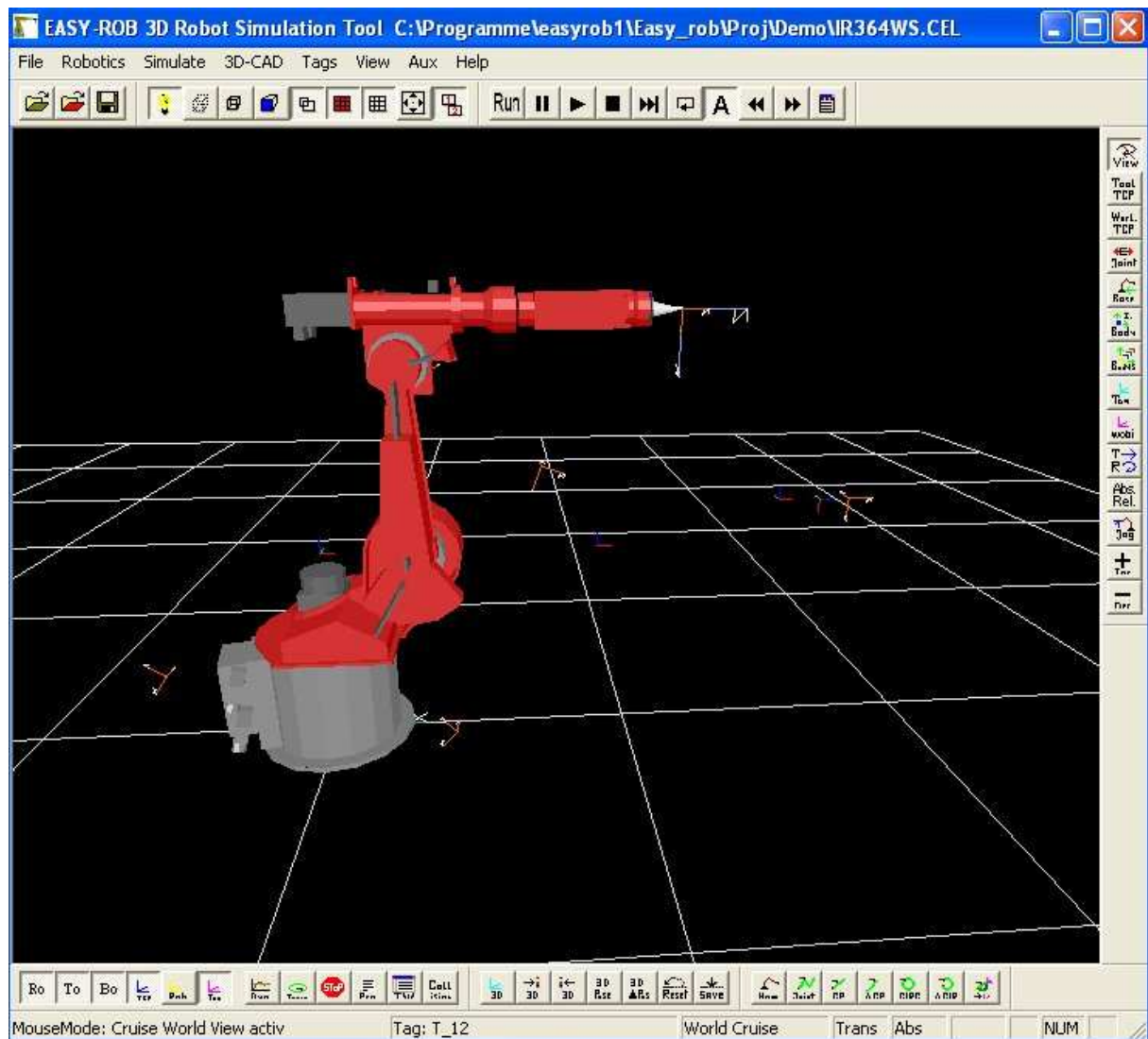


Abbildung 2 – Bedienoberfläche von Easy-Rob mit Beispielroboter

6. Generierung des geforderten Robotermodells im OLP-System

6.1 Geometrische Modellierung

6.1.1 Überprüfung der geometrischen Einzelemente

Im folgenden soll der Tricept-Roboter TR600 der Firma Neos Robotics modelliert werden.

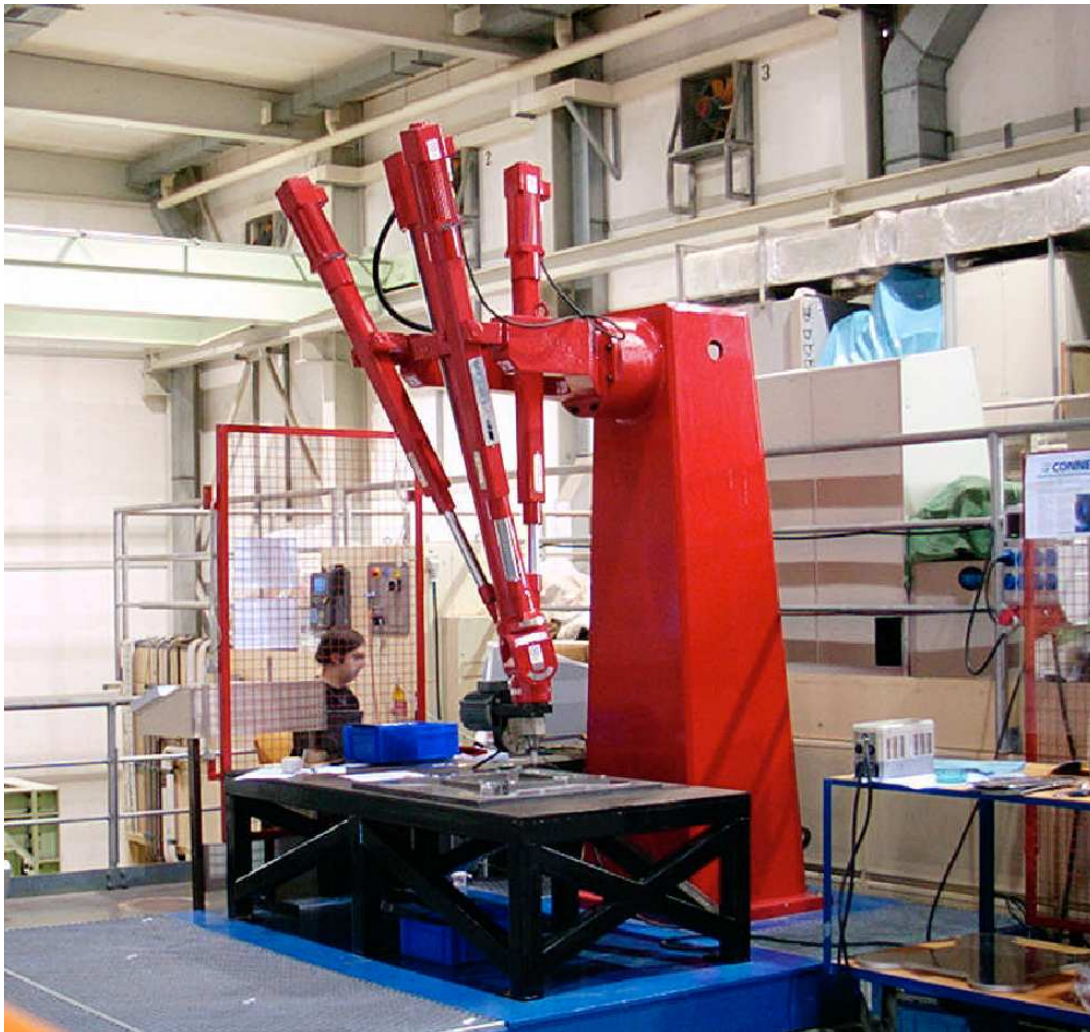


Abbildung 3 – TR600-Roboter

Da dies ein Standard-Tricept-Roboter ist, ist u.a. dieses Modell in dem Tricept-Paket des ausgewählten OLP-System EASY-ROB enthalten. Dieses Modell ist auch zu einem gewissen Maße parametrisierbar und daher als Grundlage sehr gut geeignet.

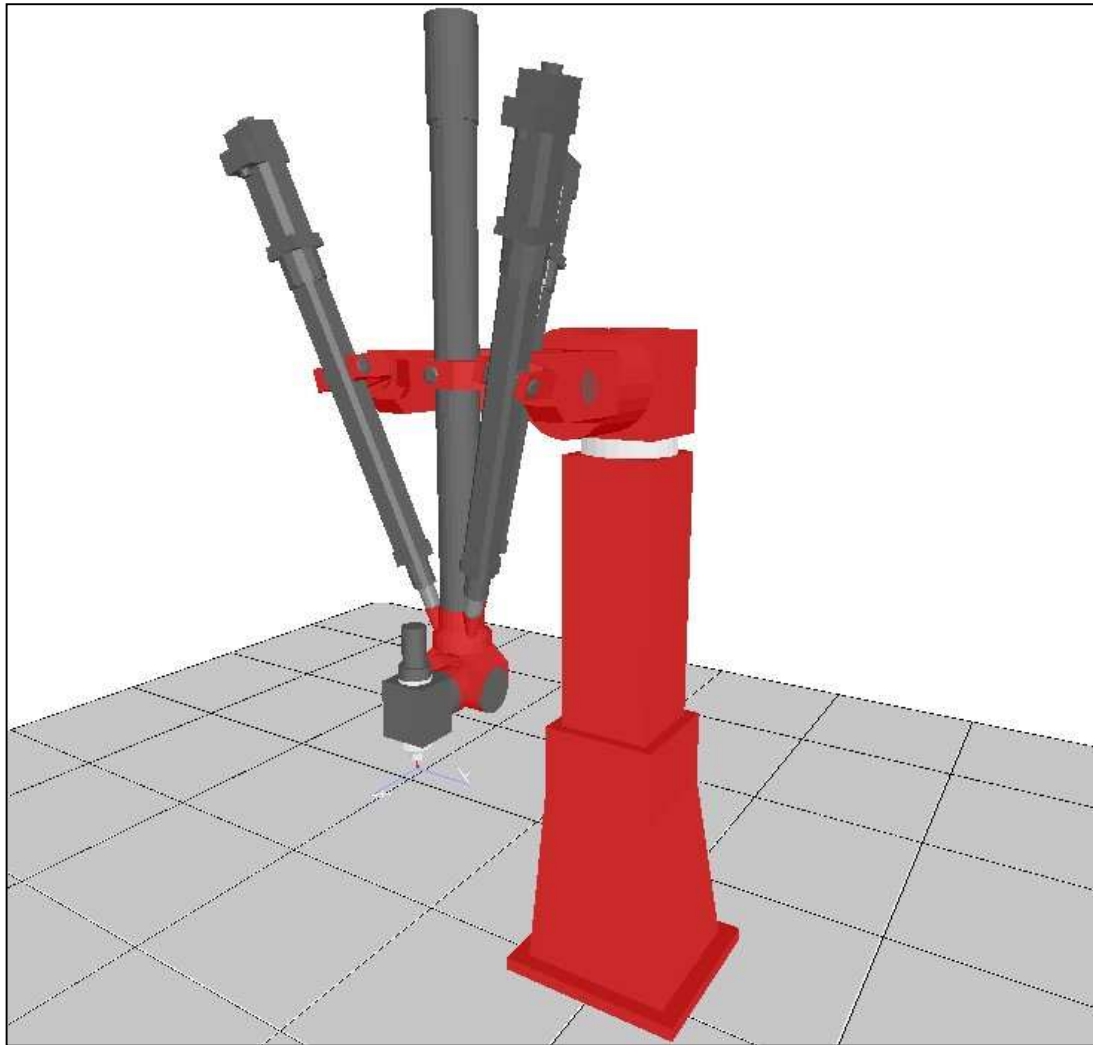


Abbildung 4 – Vorgegebenes TR600-Modell

Dieses Modell mußte zunächst mit den realen Daten des TR600 verglichen werden, um sicherzustellen, daß keine geometrische Ungenauigkeit im Modell einen Fehler in der Simulation und dem späteren Roboterprogramm hervorruft.

Ergebnis dieser Betrachtungen war, daß lediglich die Achslimits des Roboters mußten entsprechend der Realität festgelegt werden. Ansonsten war die Geometrie fehlerfrei in dem Modell beschrieben, eine weitere Korrektur war nicht erforderlich.

6.1.2 Korrektur einzelner Bauteile

Einzelne Bauteile stimmten optisch noch nicht mit der realen Farbgebung des TR600 überein. Nicht für alle zu bearbeitenden Teile war es möglich, diesen die korrekte Farbe mittels

EASY-ROB zu geben. Der Grund hierfür lag darin, daß sämtliche Teile in dem OLP-System IGRIP erstellt wurden. Dabei wurden Einzelteile bereits zu Baugruppen zusammengefügt. Jede dieser Baugruppen hat ein Hauptobjekt und eventuell weitere Unterobjekte. Diese Baugruppen wurden in Easy-Rob importiert und zu dem Roboter zusammengesetzt. In Easy-Rob konnten die Baugruppen nicht in die Einzelobjekte aufgeteilt werden. Beim Ändern der Farben einer Baugruppe in Easy-Rob wurde lediglich die Farbe des Hauptobjektes geändert. Die übrigen Objekte blieben unverändert. Das Problem trat bei einem Teil des Ständers, der zentralen Führungssäule und einem Element der Roboterhand auf. Diese Teile mußten zunächst wieder zurück in das OLP-System IGRIP exportiert werden. Hier war es nun möglich, den Elementen die gewünschte Farbe zu geben und sie wieder zu importieren. So konnte das Robotermodell komplett die korrekten Farben erhalten.

Anschließend wurde das Modell in dem OLP-System so verschoben und gedreht, daß der Ursprung des Weltkoordinatensystem im Roboterkopf liegt, damit bei dem späteren Roboterprogramm das Koordinatensystem mit dem Koordinatensystem der internen Steuerung des Roboters übereinstimmt.

Da der Boden in dem OLP-System genau durch den Ursprung des Weltkoordinatensystem in der x-y-Ebene verläuft, wurde dieser voreingestellte Boden ausgeblendet und ein eigener Boden als CAD-Objekt so eingefügt, daß der Roboter auf diesem steht.

Des weiteren wurde in dem vorgegebenen Modell das am Roboter befindliche Werkzeug entfernt, da zur Überprüfung von Achsstellungen Referenzdaten für einen Roboter ohne Werkzeug vorlagen. Das Koordinatensystem an der Roboterhand mußte nun um 90° um die y-Achse gedreht werden, damit auch dieser Punkt mit der realen Robotersteuerung übereinstimmt.

Damit waren alle Modifikationen abgeschlossen, um die notwendige Kongruenz zwischen Modell und realem Roboter zu erhalten.

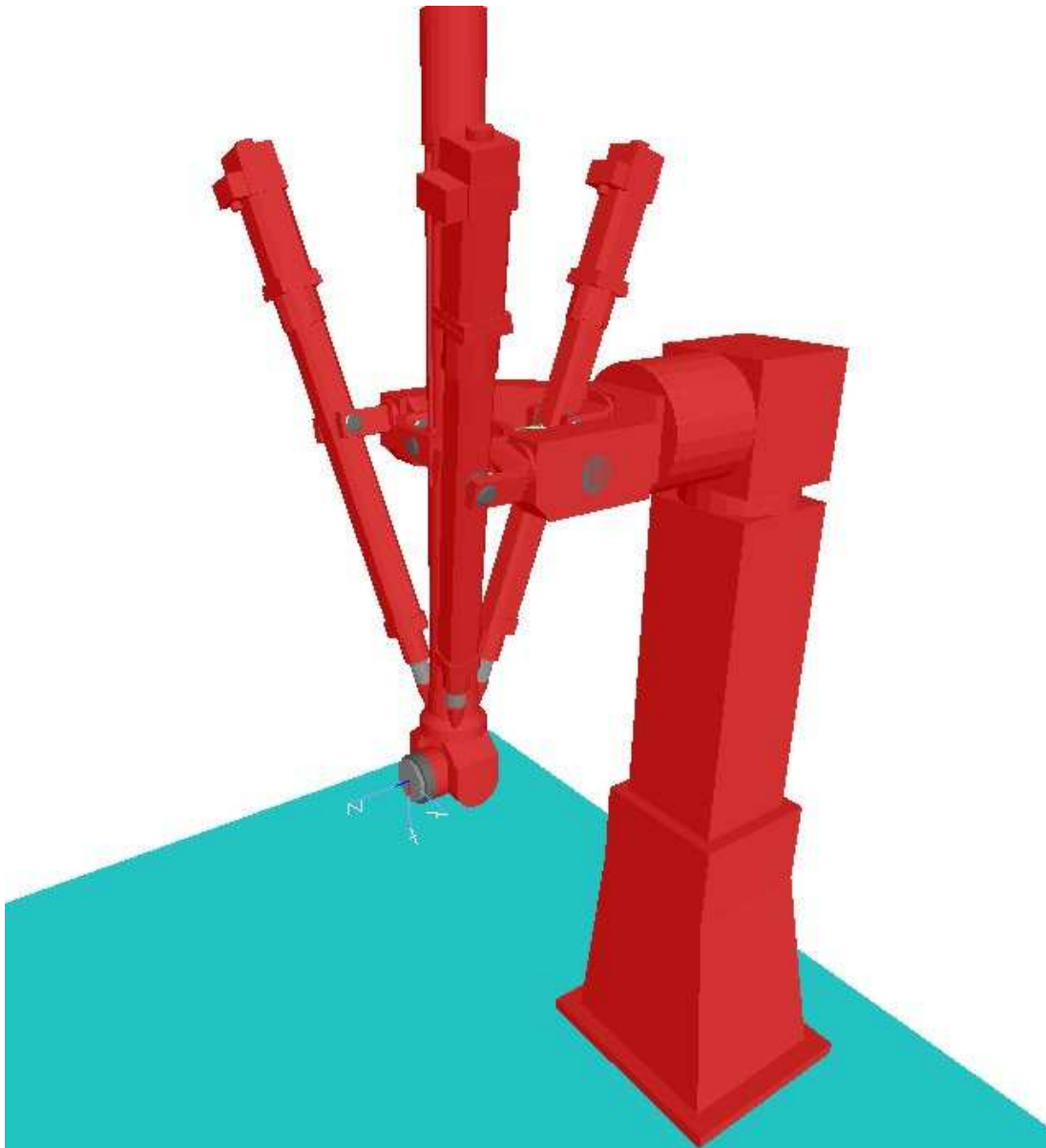


Abbildung 5 – Korrektes TR600-Modell

Durch diese Modifikationen ist es möglich, ein mit Easy-Rob generiertes Roboterprogramm nach seiner Übersetzung in die jeweilige Robotersprache direkt auf dem Roboter auszuführen, ohne weitere Umrechnungen während oder nach der Übersetzung.

6.2 Kinematische Modellierung

6.2.1 Anpassung und Änderung der kinematischen Gleichungen

Das nächste Ziel war, eine korrekte Kinematik des TR600 in das OLP-System zu implementieren. Es ist zwar eine Lösung des inversen kinematischen Problems¹ in dem Tricept-Paket des OLP-System inbegriffen. Da bei diesem OLP-System aber die Möglichkeit besteht, eine eigene Berechnung zu implementieren, hat man sich hier dazu entschieden, die mitgelieferte Berechnung nicht zu benutzen. Es gab drei Gründe, die zu dieser Entscheidung führten.

Erstens ist die Kinematik bei Tricept-Robotern eine kompliziertere als bei anderen Robotern. Daher kann man nicht davon ausgehen, daß die inverse Kinematik, bei der zu einem Zielpunkt des Roboters die entsprechenden Achswinkel berechnet werden, ausreichend und korrekt beschrieben ist. Bei dem Tricept-Roboter muß die so genannte Schraubenwirkung mit in die Berechnung einfließen, die ein gewisses Drehen zweier der Spindelmotoren beim Bewegen des dritten beschreibt. Zu dieser Bewegung kommt es dadurch, daß bei der Bewegung eines Motors der Angriffspunkt aller Achsen verdreht wird, wodurch die Spindeln der anderen beiden Motoren etwas zurückdrehen. Diese Berechnung sollte in die inverse Kinematik des OLP-Systems und somit in das spätere Roboterprogramm möglichst exakt eingebunden werden, um die Wirklichkeit möglichst genau zu modellieren.

Zweitens ist die mitgelieferte Berechnung von den mitgelieferten Roboterparametern abhängig. Diese Parameter sind zwar für die Visualisierung ausreichend, aber es sollten in die Berechnung noch eine Vielzahl weiterer Parameter einfließen, um die angesprochene Genauigkeit in der Berechnung auch zu gewährleisten, wenn die Parameter des realen Roboters z. B. durch eine Kollision oder Verschleiß anderer Art sich stark verändern. Dann ist es nicht mehr möglich, nur über die Änderung der Grundparameter des Roboters diesen korrekt zu beschreiben, da die verschiedensten Fehler in der Geometrie auftreten können.

Drittens sprach die Flexibilität einer eigenen Berechnung für sich, da die mitgelieferte Berechnung in einer „black box“ stattfindet und nicht selbständig durch neue Erkenntnisse verändert werden kann.

¹ Zu einer gegebenen Position und Orientierung (x, y, z, A, B, C) sollen durch Rückwärtstransformation die zugehörigen Gelenkwinkel $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ des Roboters bestimmt werden; im Gegensatz zu dem direkten kinematischen Problem, bei dem aus den gegebenen Winkeln $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ durch Vorwärtstransformation die kartesische Position und Orientierung (x, y, z, A, B, C) bestimmt werden soll. [3]

So war die Entscheidung gefallen, eine als C-Code zur Verfügung gestellte Berechnung der inversen Kinematik zu verwenden. Nach kleinen Korrekturen einzelner Variablendeklarationen war die Berechnung eigenständig lauffähig. Ein Test dieser mit exakten Referenzdaten zeigte deren Korrektheit.

Diese Berechnung erfolgt unter Zuhilfenahme einer Datei mit den Parametern für den Roboter selbst und einer für die Parameter des Robotertools. Zusätzlich enthielt eine Datei die anzufahrenden Punkte. Die Einbindung der Dateien mußte leicht verändert werden, da die Verzeichnis- und Namenskonventionen nicht für eine Einbindung in Easy-Rob geeignet erschienen. Die Datei, die die Berechnung mit anzufahrenden Punkten versorgte, war nun überflüssig, da dies direkt von Easy-Rob ohne den Umweg über eine Datei erfolgt.

6.2.2 Implementierung der Kinematik in das OLP-System

Bei dem OLP-System Easy-Rob besteht die Möglichkeit zwölf eigene Berechnungen der inversen Kinematik mittels einer Direct Link Library (DLL) zu implementieren. Dazu ist das Grundgerüst der DLL mit den Funktionen für die zwölf Berechnungen bereits als Visual C++-Arbeitsbereich mitgeliefert. Das OLP-System übergibt an die jeweils aufgerufene Funktion den anzufahrenden Punkt in Framedarstellung². Von der Funktion wird erwartet, daß sie die erforderlichen Achsdaten des Roboters wieder zurückgibt, damit diese im OLP-System visualisiert werden können. Zusätzlich können verschiedene Variablen, wie etwa Offsets, Basisposition des Roboters oder Vektor von der Basisposition zum ersten Gelenk ausgelesen und verarbeitet werden.

Da der Frame, den Easy-Rob übergibt nicht den Konventionen entspricht, die bei der vorgegebenen Berechnung zu Grunde liegen, mußte dieser verschiedenen Transformationen unterzogen werden.

Zunächst ist es notwendig, aus den übergebenen Daten tatsächlich einen Frame zu erstellen, indem man die 4. Dimension einführt.

Des weiteren müssen die in Metern übergebenen x-, y- und z-Koordinaten für die Berechnung in Millimeter umgewandelt werden. Außerdem ist die z-Koordinate mit einem Offset von 2750 mm versehen, was der Höhe des Roboterkoordinatensystems über Grund entspricht. Aufgrund des Unterschieds der Abbildung der Realität zwischen der Berechnung und dem

² Zusammenfassung der Orientierungsmatrix D und des Positionsvektors t erweitert um die 4. Dimension

OLP-System muß der erhaltene Punkt um 180° um die y-Achse gedreht werden. Dieser Frame kann nun zur Berechnung der Achsstellung benutzt werden. Das Ergebnis aus der Berechnung liefert die korrekten Achsstellungen, wobei die ersten Werte von Millimeter zurück in Meter umgerechnet werden müssen. Die drei Handwinkel werden, wie errechnet, im Bogenmaß zurückgegeben.

Die Fehlerbehandlung in der Berechnung der inversen Kinematik mußte ebenfalls an die Einbindung in Easy-Rob angepaßt werden. Während die ursprüngliche Berechnung in einem DOS-Fenster ablief und dort auch die Fehler ausgegeben werden konnten, müssen auftretende Fehler nun über Easy-Rob an den Benutzer gemeldet werden. Zu diesem Zweck gibt es eine Funktion, mit der Informationen in einem Infofenster von Easy-Rob angezeigt werden können. So wurden Fehlermeldungen, die das Öffnen der Parameterdateien sowie ein Überschreiten der Achslimits betreffen, an diese Funktion übergeben, die die Informationen an den Benutzer weiterleitet.

Diese Funktionen werden als DLL compiliert und gelinkt, so daß Easy-Rob die DLL beim Start einbezieht. Im Easy-Rob selbst kann man für den TR600 nun die benutzerspezifische Berechnung der inversen Kinematik auswählen.

6.3 Test der Simulation

6.3.1 Abgleich mit realen Achs- und Raumkoordinaten

Für den Test dieser Simulation wurden von dem GKSS Forschungsinstitut Geesthacht Roboterachswerte mit der zugehörigen kartesischen Position zu Verfügung gestellt.

Die Werte der Robotergelenke waren:

Gelenk A = -100 mm

Gelenk B = -200 mm

Gelenk C = -300 mm

Gelenk D = 0°

Gelenk E = 0°

Gelenk F = 0°

Zu diesen Achswerten errechnete die reale Robotersteuerung die kartesische Position, zeigte diese an und verfuhr den Roboter an diese Position mit den folgenden Werten:

X = 9.961 mm

$$Y = -448.654 \text{ mm}$$

$$Z = 1219.012 \text{ mm}$$

$$A = 20.21^\circ$$

$$B = -0.46^\circ$$

$$C = 0.01^\circ$$

Um nun die in Easy-Rob implementierte inverse Kinematik zu überprüfen, wurde ein Tag-Point³ mit den gegebenen kartesischen Koordinaten gesetzt und der Roboter an diese Position verfahren. Die sich dabei ergebenden Gelenkwinkel konnten nun mit den vorgegebenen Gelenkwinkeln verglichen werden.

$$\text{Gelenk A} = -99.75 \text{ mm}$$

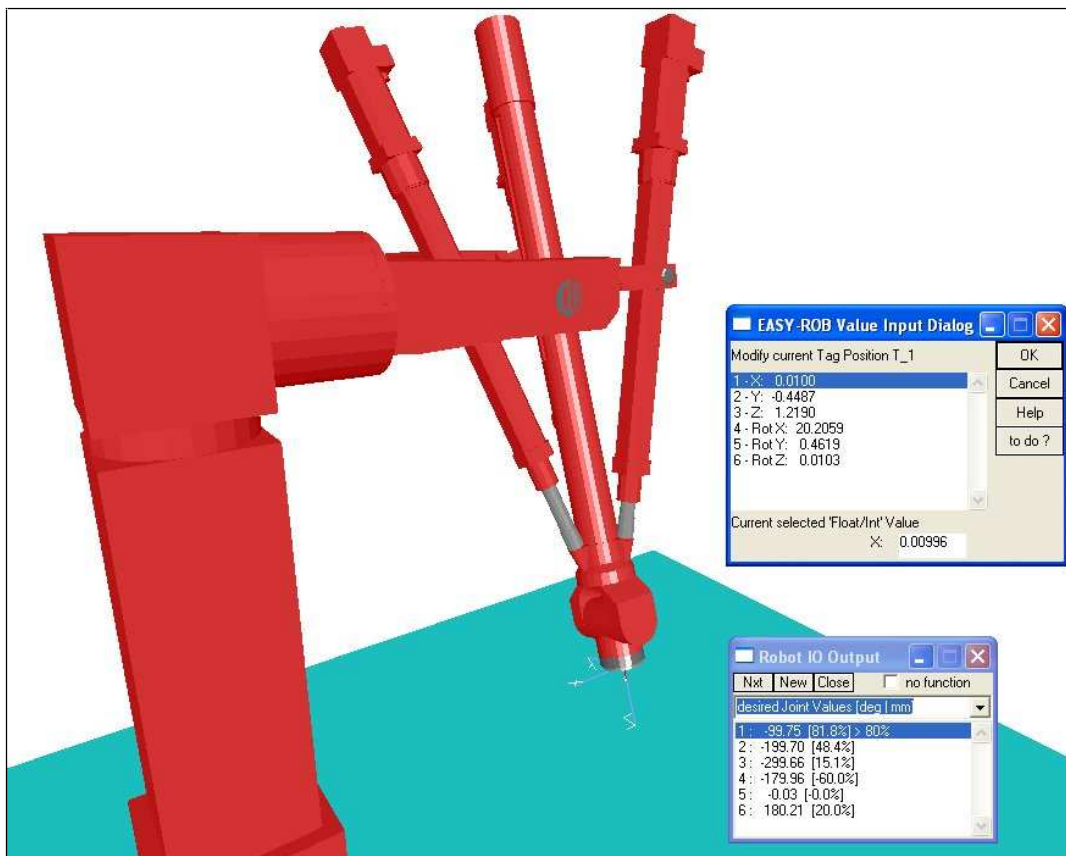
$$\text{Gelenk B} = -199.70 \text{ mm}$$

$$\text{Gelenk C} = -299.66 \text{ mm}$$

$$\text{Gelenk D} = -179.96^\circ$$

$$\text{Gelenk E} = -0.03^\circ$$

$$\text{Gelenk F} = 180.21^\circ$$



³ Arbeitspunkt des Roboters

Abbildung 6 – TR600 Modell an einem Testpunkt

Dabei ergab sich für die Zielsetzung der Genauigkeit ein relativ hoher Fehler. Dieser lag im Bereich von 0.2 – 0.35 mm pro Gelenk.

Bei einer Überprüfung des zur Verfügung gestellten Algorithmus in seiner ursprünglichen Umgebung ergaben sich die gleichen Werte und damit Abweichungen zu den von der Steuerung errechneten Werten. Es konnten allerdings keine Fehler in dem Algorithmus festgestellt werden. Auch die kinematischen Vorgaben waren fehlerfrei und konnten somit nicht für diese Abweichung verantwortlich sein. So entstand die Vermutung, daß die Steuerung des realen Roboters selbst die Fehlerquelle war. Dieser Verdacht war insofern begründet, da die Steuerung relativ alt ist und die Möglichkeit besteht, daß diese alte Steuerung die Schraubenverdrehung nicht mit berücksichtigt.

Um diese These zu bestätigen oder zu widerlegen, wurde das TR600 Modell auf den SRT60 übertragen und Tests mit realen Koordinaten und Achswerten durchgeführt. Diese Maßnahme wäre ohnehin notwendig gewesen, um auch die Kompatibilität des Algorithmus und des Programms zu verifizieren.

7. Übertragung des TR600-Modells auf den SRT60

7.1 Anpassung der geometrischen Einzelemente an den SRT60

Geometrisch gesehen gleicht der SRT60 dem TR600 fast vollständig. Die drei wesentlichen Unterschiede sind erstens die Farbe des Roboters, zweitens das Logo der Firma, drittens eine leichte Abweichung des Maßes der Roboterhand.

Beim Ändern der Farbe ergaben sich die gleichen Probleme wie bei der Bearbeitung des TR600, die sich auf die gleiche Weise lösen ließen.

Das Comau-Logo mußte entfernt und durch das SEF GmbH Logo ersetzt werden.

Eine Vermessung der Hand des SRT60 hat ergeben, daß das letzte aktive Gelenkt 0,1 Millimeter länger ist als bei dem TR600. Dieser Wert wurde als einziger an der Geometrie verändert.

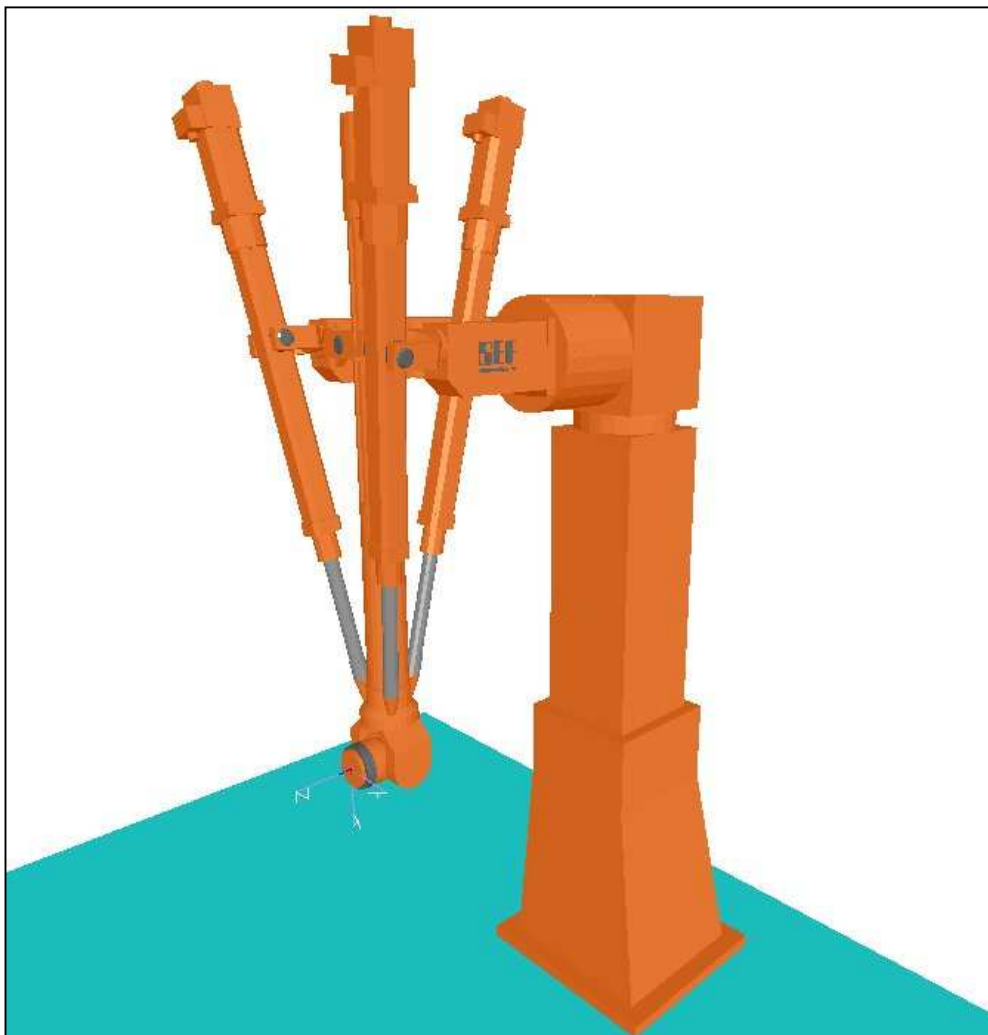


Abbildung 7 – Korrektes SRT60-Modell

7.2 Anpassung der kinematischen Daten an den SRT60

Zunächst wurden die Dateien für die Kinematik- und Toolparameter erstellt, wobei diese sich nicht nur in der Handlänge von dem TR600 unterscheiden. Ein weiterer Unterschied besteht in der Nullstellung der Roboter. Der TR600 hat seine Nullstellung ca. 40 cm höher als der SRT60. Außerdem unterscheiden sich beide Roboter in ihren Limits, so daß auch diese für den SRT60 verändert werden mußten. Während der TR600 die letzte Achse zehn Mal in jede Richtung drehen kann, ist der SRT60 auf knapp zwei Mal beschränkt. Diese Änderungen wurden auch für die Geometrie übernommen, um ein korrektes Abbild zu erhalten.

Bei der Berechnung der inversen Kinematik mußte eine neue Funktion für den SRT60 geschaffen werden, da bei der Berechnung die Kinematik- und Toolparameter gelesen werden und für jeden Roboter eine eigene Parameterdatei vorgesehen ist.

7.3 Test der Simulation mit dem SRT60

Für den Test des SRT60-Modells sollten die gleichen Achswerte wie bei dem TR600 zu Grunde gelegt werden. Dabei ergab sich aber das Problem, daß der SRT60 wegen seiner tieferen Nullstellung die drei „Stangen“ lediglich um 290 mm ausfahren kann. Als Referenzdaten waren allerdings für die Achse 300 mm vorgegeben. Dieser Wert konnte aber dennoch realisiert werden, indem die Begrenzungen softwareseitig außer Funktion gesetzt wurden.

Für die Achswerte:

$$A = -100\text{mm}$$

$$B = -200\text{mm}$$

$$C = -300\text{mm}$$

$$D = 0^\circ$$

$$E = 0^\circ$$

$$F = 0^\circ$$

ergaben sich in der Realität folgende kartesische Koordinaten:

$$X = 9.1\text{mm}$$

$$Y = -560.4\text{mm}$$

$$Z = 1552.4\text{mm}$$

$$A = 0^\circ$$

$$B = 0.33^\circ$$

$$C = 19.85^\circ$$

Bei dem Test des SRT60 wurde ebenfalls ein Tag-Point mit den gegebenen kartesischen Koordinaten festgelegt, der von dem Roboter angefahren werden sollte. Auch hier erkannte das System, daß der Punkt außerhalb der Achslimits liegt, er konnte aber dennoch angefahren werden.

Bei diesem Test ergaben sich die folgenden Achswerte:

$$A = -100.00\text{mm}$$

$$B = -199.99\text{mm}$$

$$C = -299.99\text{mm}$$

$$D = -176.24^\circ$$

$$E = -0.02^\circ$$

$$F = 183.87^\circ$$

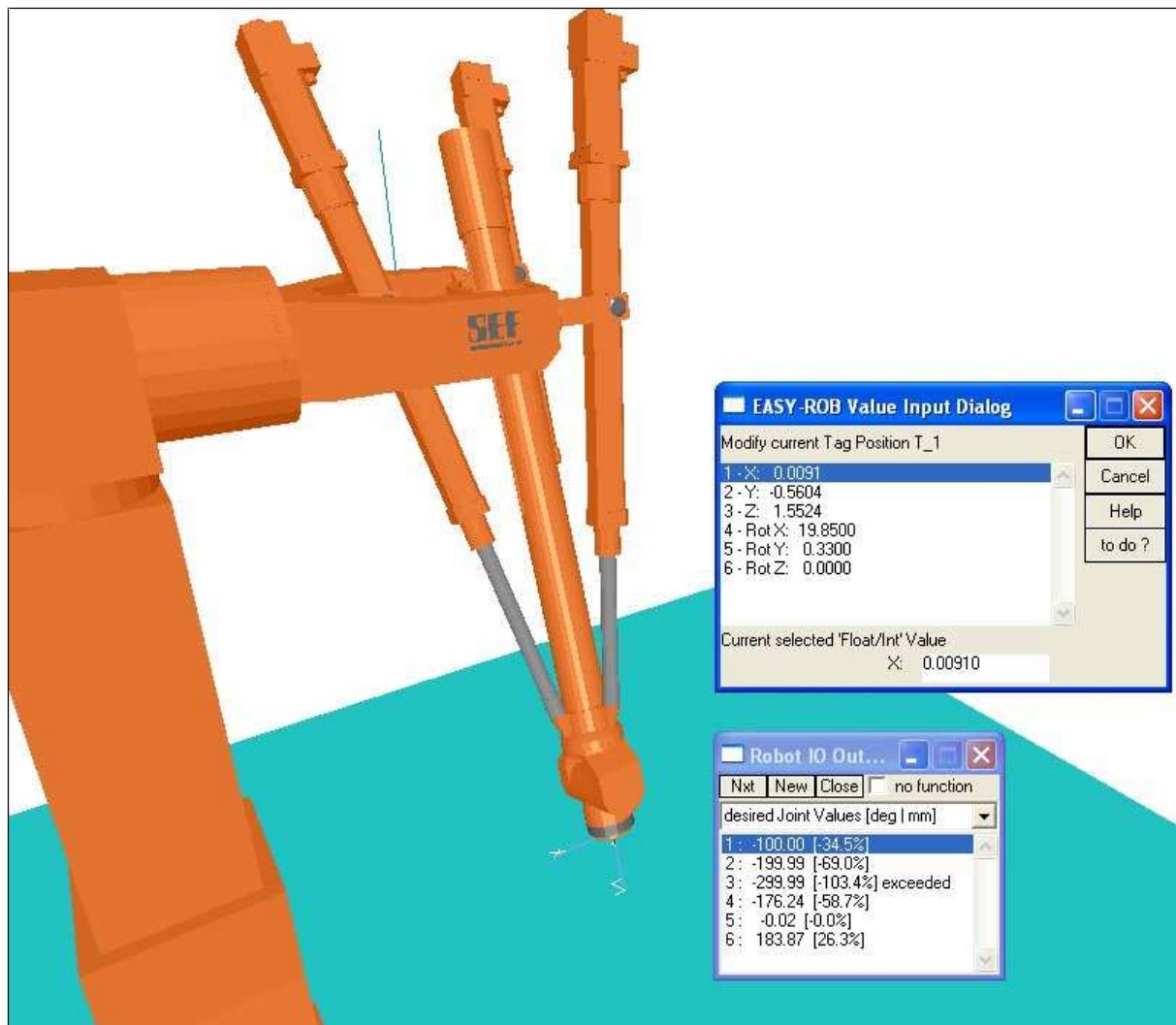


Abbildung 8 – SRT60 an einem Testpunkt

Bei diesen Werten war eine Abweichung von weniger als ein tausendstel Millimeter zu erkennen.

Die Abweichung der Werte des vierten und sechsten Gelenkes ergeben sich, da es unendlich viele verschiedene Möglichkeiten für den Roboter gibt, einen Punkt zu erreichen, bei dem die Roboterhand gestreckt ist (sog. Singularität). Die Summe des vierten und sechsten Gelenkes zusammen muß lediglich 0° ergeben. Die sonstigen Abweichungen kommen durch Rundungsfehler in der Berechnung selbst zustande. Dieser Wert übertraf die Anforderungen um mindestens eine Größenordnung. Es konnte also festgestellt werden, daß die Position des Modells mit der Position der Realität übereinstimmt. So lag nahe, daß die Fehler bei dem TR600 Modell tatsächlich in der eigentlichen Robotersteuerung und nicht in dem Modell zu suchen sind.

8. Zusammenfassung

8.1 Ergebnis

Bei dieser Arbeit hat sich gezeigt, daß es möglich ist, aus der Vielzahl von OLP-Systemen ein günstiges, PC-basiertes auszuwählen, das universell einsetzbar ist und mit dem hochgenaue Roboterprogramme generiert werden können. Dabei sind die Anforderungen an den Computer deutlich unterhalb des Standes der Technik, wodurch die Anschaffung eines Rechners speziell für diese Aufgabe nicht notwendig ist.

Obwohl eigentlich noch in der Entwicklung, zeigte sich das Programm Easy-Rob insgesamt sehr bedienerfreundlich und kompatibel sowohl zu verschiedenen selbstgenerierten Robotermodellen als auch der zugehörigen kinematischen Beschreibung und Berechnung der inversen Kinematik. Das Berücksichtigen von Daten, die durch eine Kalibrierung gewonnen werden, in der Berechnung erwies sich ebenfalls als problemlos.

Tests mit realen Achs- und Raumkoordinaten bestätigten die exakte Berechnung innerhalb des Programms.

8.2 Ausblick

Notwendig ist auf jeden Fall eine genaue Klärung der Unstimmigkeiten, die sich bei dem Test mit dem TR600 ergaben, mit der GKSS. So kann die korrekte Berechnung des Programms wiederholt verifiziert werden.

Eine weitere Verifizierung des Programms kann durch Tests mit anderen Tricpet-Modellen erfolgen. Dazu bieten sich Modelle an, die geometrisch stärker von dem TR600 abweichen, als es bei dem SRT60 der Fall ist. Es bietet sich hier zum Beispiel der TR805 an.

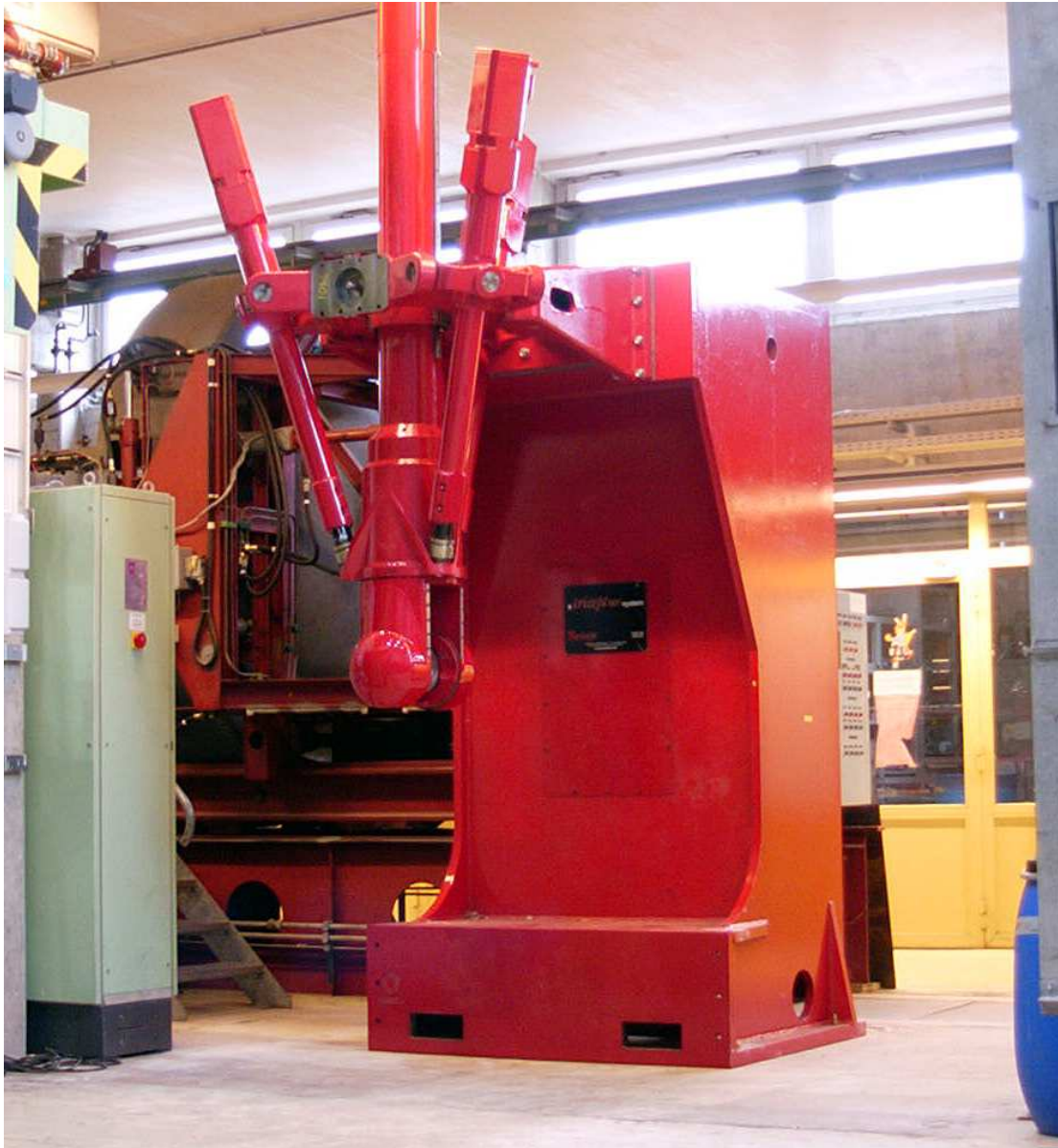


Abbildung 9 – TR805-Roboter

Anschließend muß ein Übersetzungsprogramm erstellt werden, das ein mit Easy-Rob erstelltes Programm in jede andere gewünschte Robotersprache übersetzt, da dieses Werkzeug ohne eine angeschlossene Übersetzungseinheit wertlos ist. Um dann den Kreis zu schließen und die Funktionalität des Gesamtsystems zu bestätigen, ist ein Test eines so erstellten Programms an einem realen Roboter notwendig.

Eine sinnvolle Erweiterung ist auch das Einbeziehen der Systemdynamik, bei der das Übertragungsverhalten der Lageregelkreise in den Achsen, der Antriebe und der Getriebe beeinflusst durch Größen wie Masse, Massenträgheitsmomente, Reibung oder Elastizität einbezogen wird.

9. Anhang

9.1 Struktogramm

Aufruf der Berechnung mit Übergabe der Positionsdaten von Easy-Rob
Konvertierung der Positionsdaten in Frame-Darstellung
Drehung der Position um 180°
Einlesen der Kinematik- und Tooldaten
Berechnung der Achswinkel
Limitenprüfung
Rückkonvertierung der Achswinkel
Übergabe der Achswinkel an Easy-Rob

9.3 Abbildungsverzeichnis

- 1 Struktur eines CAD/CAM-gestützten Offline-Programmiersystems
- 2 Bedienoberfläche von Easy-Rob mit Beispielroboter
- 3 TR600-Roboter
- 4 Vorgegebenes TR600-Modell
- 5 Korrektes TR600-Modell
- 6 TR600 an einem Testpunkt
- 7 Korrektes SRT60-Modell
- 8 SRT60 an einem Testpunkt
- 9 TR805-Roboter

9.4 Literaturverzeichnis

- [1] Harkort, R. Ein Beitrag zur Steigerung der Verfah- und Positioniergenauigkeit von Industrierobotern im Rahmen eines Offline-Programmiersystems, Dortmund, 1988
- [2] Roos, E. Anwendungsorientierte Meß- und Berechnungsverfahren zur Kalibrierung off-line programmierter Roboterapplikationen, Düsseldorf, VDI Verlag, 1998
- [3] Roos, E. Robotik, Vorlesungsskript, Universität der Bundeswehr, Hamburg 1996

9.5 CD mit Programm und dieser Arbeit als Word-Dokument