Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

AW1 Ausarbeitung

Andreas Dubs

Entwicklung eines impliziten Planers mit Echtzeit-Kollisionsprüfung für einen mobilen Assistenzroboter Scitos G5

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einblick

1.1 Motivation

Die mobile Roboterplattform »SCITOS« G5 der Firma MetraLabs mit einem darauf montierten Greifarm mit 6 Freiheitsgraden der Firma Schunk (Abbildung 1.1) soll effizient die gestellte Aufgaben lösen. Es sind mehrere Komponente vorhanden, wie Sensoren, Antrieb der Plattform, Rotationsgelenke des Roboterarms und Greifer, die mit einem in der Plattform sich befindenden Rechner synchronisiert werden sollen. Eine Batterie dient als Energiequelle und begrenzt damit die Laufzeit. Um die Laufzeit zu verlängern, soll auf Energieeffizienz geachtet werden.

Weiterer Aspekt ist ein sicherer Betrieb in einer Umgebung mit Menschen. Die statische Objekte lassen sich in einer 3D-Weltmodell abbilden, die dynamische Objekte dagegen können ihre Positionen ändern und müssen bei der Bewegungen ständig betrachtet werden um mögliche Kollisionen zu vermeiden. Die Häufigkeit der Prüfungen ist dabei entscheidend für die Genauigkeit. Zu große Häufigkeit bringt mit sich mehr Rechenaufwand, so kann benötigte Rechenzeit den eigentlichen Zeitschritt übersteigern. Es muss eine optimale Lösung gefunden werden, um eine Echtzeit zu gewährleisten.

Abbildung 1.1: Scitos G5 Plattform

1.2 Aufgabenfelder

Die gesamte Ansteuerung teilt sich auf mehrere Aufgabenfelder die auf einander abgestimmt und verzahnt werden müssen.

- **Aktorik** - steuern und koordinieren ist eine komplexe Echtzeitaufgabe. Die Ansteuerung der mobilen Plattform fällt mehr in die Navigation, muss aber für die Koordination mit dem Roboterarm betrachtet werden. So kann ein Beispiel mit gleichzeitig fahrender Plattform und bewegendem Roboterarm angesehen werden, um Problematik näher zu kommen.
- **Sensorik** wird für Objektidentifikation benötigt. Die Sensordaten müssen verarbeitet und mit einem 3D-Weltmodell verglichen werden. Es können die Abweichungen und Unsicherheiten auftreten, die erst verarbeitet werden müssen, bevor die gewonnene Daten im eigentlichem Sinne benutzt werden können.
- **Navigation** ist für Ortsbestimmung zuständig. Auch Wegfindung für die mobile Plattform wird in diesem Aufgabenfeld bearbeitet. Es gibt mehrere Paper, die sich mit Lösung dieser Aufgabe beschäftigen.
- **Kognition** ist für zielgerichtetes Handeln und Aktionsplanung entscheidend. Dieser Bereich ist weitgehend nicht erforscht und lässt viele Fragen offen. Die Kognition lässt sich grob auf zwei Bereiche teilen: Reflexe und Aktionsplanung. Die Reflexe sind sofortige Aktionen, die nach einem auftretenden Ereignis ausgeführt werden sollen. So kann ein Ereignis wie Kollisionsgefahr zu der Reduzierung der Geschwindigkeit und/oder einem Ausweichmanöver führen. Die Aktionsplanung beschäftigt sich dagegen mit Erstellung einer Lösung zu gestellten Aufgabe und ist damit eigentliche Künstliche Intelligenz.

1.3 Aktueller Stand

1.3.1 Roboterarm

Es wurde eine bewegungsorientierte (explizite) Steuerung des Roboterarm realisiert. Sie erlaubt dem Anwender Erstellung eigener Bahnen mit Angabe alle für die Bewegung notwendigen Informationen:

- Endpunkt der Bewegung
- Art der Trajektorie (z.B. linear, zirkular)

- Hilfspunkte, falls benötigt (z.B.für Kreisbahnen)
- Parameter wie Geschwindigkeit, Anzahl der Interpolationspunkte oder Überschleifradius können global und lokal für jede Trajektorie einzeln eingestellt werden

Für die Berechnungen benötigter Anfangspunkt wird der aktuellen Position entnommen. Die erstellten Listen lassen sich abspeichern und später laden. Es gibt auch die Möglichkeit aktuelle Position in die Liste übernehmen oder die Positionen durch direkte Angabe von Gelenkwerten anfahren/abspeichern (Abbildung 1.2).

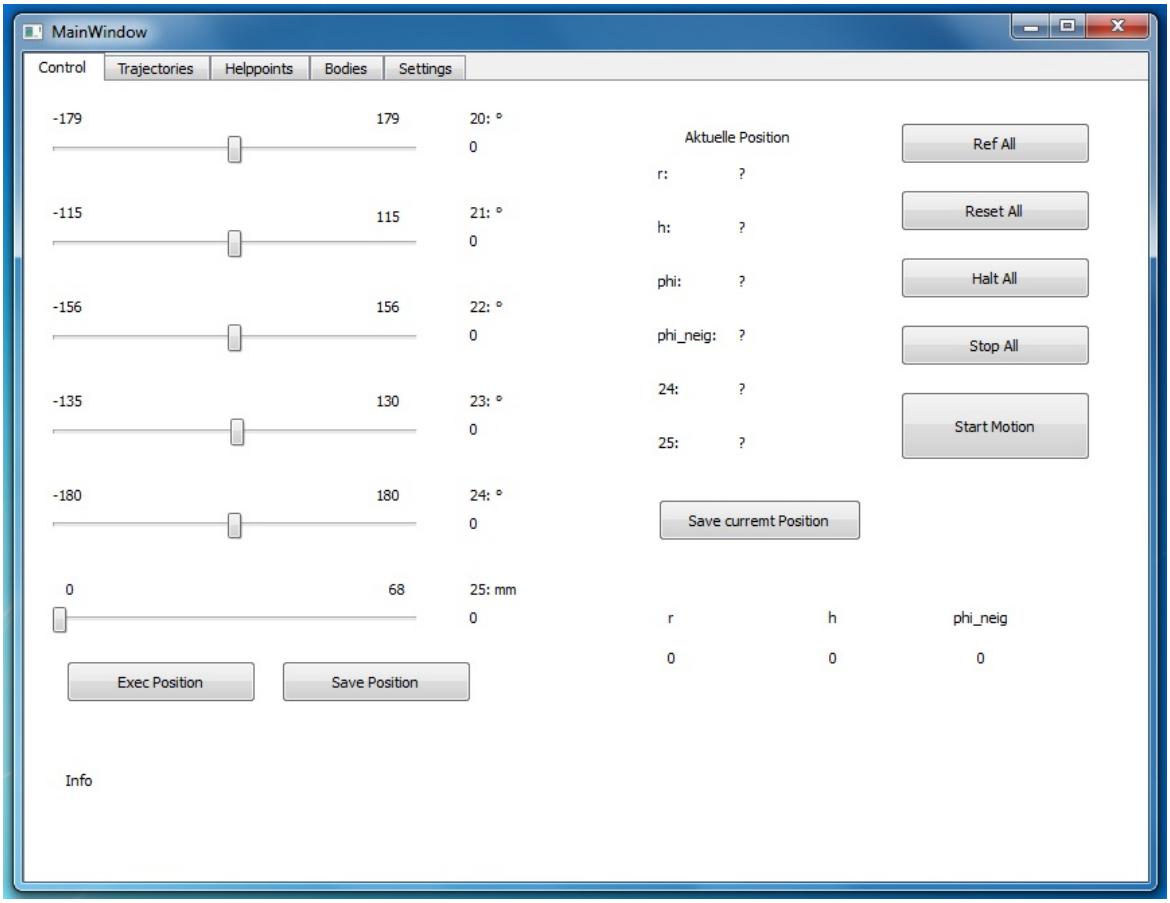

Abbildung 1.2: Anwenderprogramm [Dubs (2011)]

Es können auch einfache Körper für eine Kollisionsprüfung eingegeben werden. Die Überprüfung wirkt sich aber nur auf den Effektor aus. Die explizite Steuerung bewirkt, dass es bei Kollisionen und Fehlern wie unerreichbare Stellen oder unzulässige Gelenkpositionen der Anwender selbst eingreifen und die Trajektorien dementsprechend anpassen muss. Dies erweist sich als sehr mühsam, vor allem weil die Suche nach geeigneten Hilfspunkten viel Verständnis benötigt und nicht alle Koordinaten für eine Trajektorie gleich ersichtlich sind.

1.3.2 ROS

Für die weitere Anwendung im Roboterprojekt Scitos@HAW wurde das Framework ROS (Robot Operating System) gewählt. Den verfügbaren Dokumentationen nach bietet es, verglichen mit anderen Frameworks, den größten Funktionsumfang bei hochgradiger Flexibilität, um als Steuersoftware eines autonomen, mobilen Assistenzroboters zu fungieren. Weiterhin sollen die Funktionen der bestehenden Entwicklungen in zur ROS kompatible Plugins migriert werden.

ROS ist ein open-source Framework für Roboter und wurde 2007 im Projekt STAIR2 des Artificial Intelligence Laboratory der Standford University entworfen. Seitdem wird ROS vom Robotikunternehmen Willow Garage weiterentwickelt.

ROS (Robot Operating System) bietet Bibliotheken und Werkzeuge, um Software-Entwickler bei der Erarbeitung von Roboter-Steuerungen zu unterstützen. Diese Bibliotheken werden in der eigenen Programmentwicklung eingebunden und übernehmen etwa die Parameterverwaltung, die Steuerung periodischer Aufgaben sowie den Versand und Empfang von Nachrichten. Sie werden primär für C++ und Python entwickelt, weiterhin gibt es experimentelle Bibliotheken für Lisp, Java und Lua [Kolbe (2011)].

1.3.3 Umweltmodell

Aktuell wird eine Kinect-Kamera dazu genutzt, den Raum und Objekte dreidimensional zu erfassen. Diese Arbeit wird bereits mit Hilfe des ROS-Frameworks und den enthaltenen verfügbaren Paketen für Kinect und räumliche Karten entwickelt. Aktuell besteht das Problem, dass manche der für die Kinect benötigten Pakete nicht unter ROS-*diamondback*, oder unter Fedora lauffähig sind. Mittelfristig ist jedoch eine Aktualisierung auf ROS-*electric* ohnehin zu empfehlen. Weiterhin würde eine Umstellung des Roboter-Steuercomputers auf das Betriebssystem Ubuntu diverse Probleme vermeiden [Kolbe (2011)].

1.3.4 Kamera

An der Roboterhand wurde eine Kamera vom Typ *Imaging Source DMK 41BF02* montiert, welche bislang zur manuellen Sichtkontrolle der Bodenumgebung während ferngesteuerter oder autonomer Bewegungen genutzt wird. Im RobotVision-Labor wurden diverse Bildverarbeitungsverfahren für die Objekterkennung unter MatLab entwickelt. Der dabei oft in Kombination verwendete Katana-Greifarm ist wesentlich kleiner, hat aber dieselbe Kinematik wie der Scitos@HAW-Roboterarm. Es wäre daher möglich, diese Funktionen für ohne größeren Aufwand auf den SCITOS G5-Roboter übertragen zu können.

1.3.5 Laserscanner

Mit dem Laserscanner vom Typ *Leuze Rotoscan RS4*, der auf der mobilen Plattform montiert ist, wird eine 2D-Karte der Umgebung erstellt. Die Karte eignet sich für die Navigation und Ortsbestimmung in Laborräumen (Abbildung 1.3). Bei der Navigation wird versucht eine globale sowie eine lokale Pfadplanung zu einer gegebenen Zielkoordinate zu berechnen und diese anzufahren. Dazu werden die verfügbare Karte als auch aktuelle Sensorinformationen beachtet.

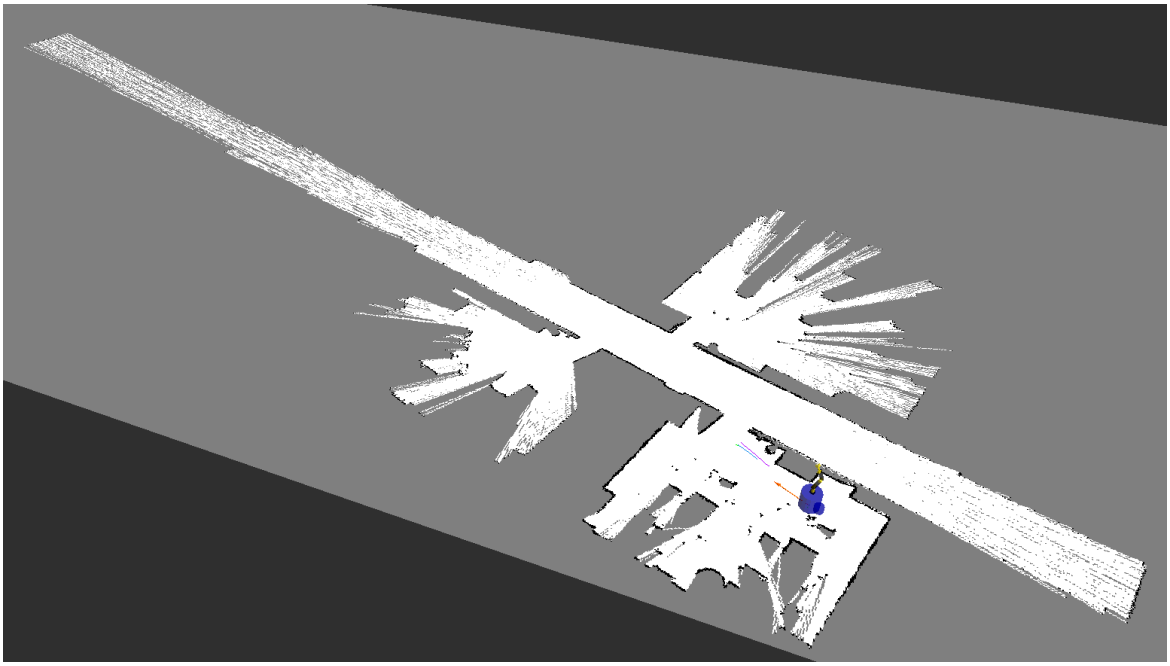

Abbildung 1.3: Kartierung [Kolbe (2011)]

1.3.6 Sprachsteuerung

Wenn die in Entwicklung befindliche Sprachsteuerung einen Befehl herausgearbeitet hat, muss dieser in passende ROS-Kommandos umgesetzt werden. Die Sprachsteuerung selbst wurde unter Linux entwickelt und sollte daher auch auf den Roboter-Steuercomputer portiert werden können.

2 Mein Einsatz

2.1 Vorbereitung

Der erste Schritt meiner Arbeit ist die Umstrukturierung und Anpassung entwickelter Algorithmen für den Einsatz mit ROS. Dabei soll die vorhandene Interpolation mit resultierenden gleichen Abständen durch eine Zeitinterpolation ersetzt werden. Für die Testzwecke wurde mit MatLab eine 3D-Simulation erstellt, die eine Interpolation mit einstellbaren Zeitschritten erlaubt (Abbildung 2.1).

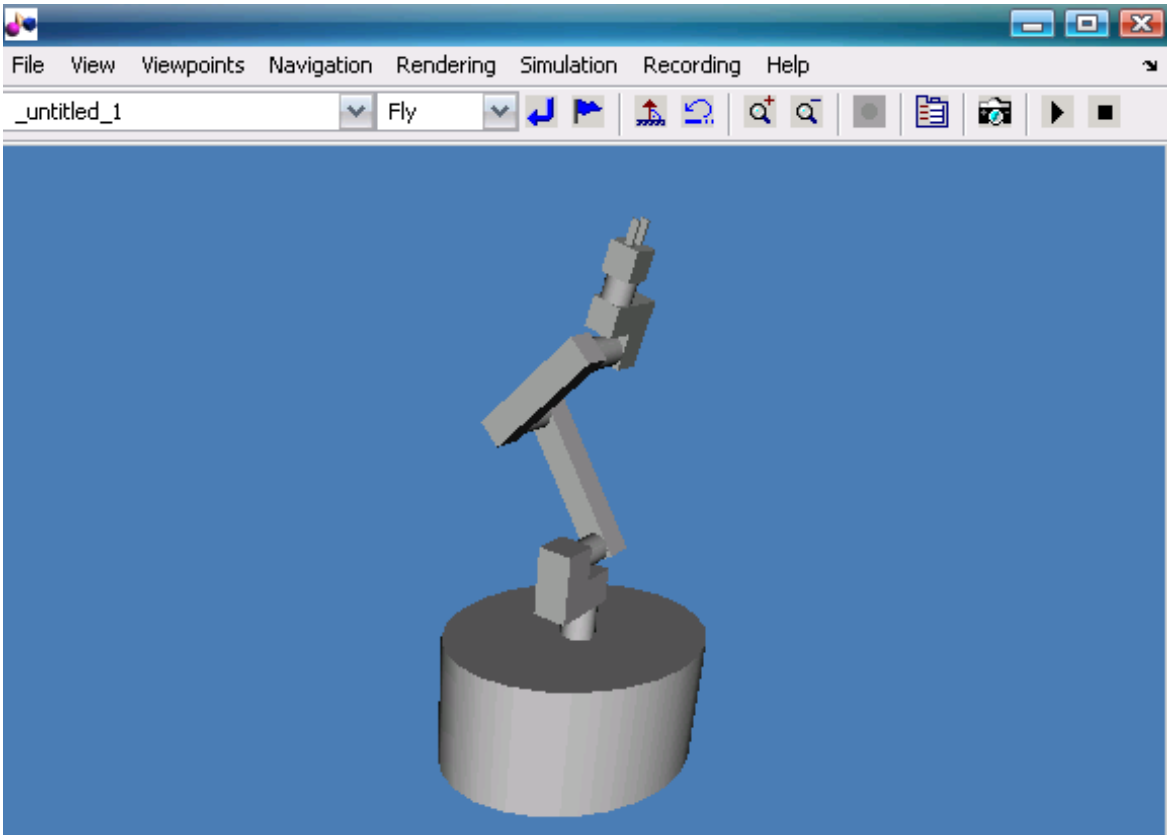

Abbildung 2.1: MatLab 3D-Simulation [Dubs (2012)]

Die Simulation soll erweitert werden, damit auch die Energieeffizienz erkundigt werden kann. Dafür müssen wirkende Kräfte und Drehmomente in die Simulation einbezogen werden. Dies erfordert einen tiefen Einsicht in die Modellierung technischer Systeme für Erstellung dazugehöriger mathematischer Gleichungen und Benutzung dieser in der Modellbildung. Die dadurch entstandene Algorithmen können weiter benutzt und in ROS eingebunden werden, was ein Verständnis des ROS-Frameworks erfordert.

Es können nicht alle vorhandene Algorithmen benutzt werden, besonders auf Hinsicht auf die neue Interpolationsart. Diese Algorithmen müssen neu geschrieben werden. Zwar bring ROS-Framework schon viele Lösungen mit sich, aber es ist noch zu erkundigen, ob die für den Einsatz gut eignen. So besitzt Framework eingebaute Pakete zur Koordinatentransformation, aber eine speziell für einen bestimmten Roboter entwickelte Rückwärtstransformation ist immer effizienter. Es existiert bereits eine solche inverse Kinematik, die sich gut in dem Anwenderprogramm und der 3D-Simulation bewährt hat.

2.2 Kollisionsprüfung

Der Kern liegt aber bei der Erstellung des ungefährlichen Assistenten des Menschen. Dies bedeutet Einbau einer umfangreichen Kollisionsprüfung, die permanent während der Bewegungen erfolgen soll. Es muss ein Echtzeitverhalten sichergestellt werden, damit eine Kollision nicht zu spät erkannt wird und die Ergebnisse einer Prüfung rechtzeitig für die Weiterverarbeitung bereit stehen.

Zwar ist eine Kollisionsprüfung vorhanden, aber sie beschäftigt sich nur mit dem Effektor. Außer dem Greifer müssen auch andere Komponente des Roboters nach Kollisionen geprüft werden, wie mit sich selbst, so auch mit den Objekten in der Umgebung. Ein 3D-Umweltmodell ist dafür von großer Bedeutung. ROS-Framework bietet schon einige Pakete an, ob die sich für eine Echtzeitprüfung eignen ist noch fraglich. So muss eventuell aus dem gesamten Umweltmodell ein Umgebungsmodell extrahiert werden, um die Anzahl zu prüfender Objekte zu verringern. Das Umgebungsmodell kann dabei einen Laborraum oder Arbeitsraum des Roboters (Greifreichweite) repräsentieren. Für eine weitere Effizienzsteigerung können komplexe Objekte in einfache Grundkörper wie Quader oder Zylinder eingeschlossen werden (Abbildung 2.2) um schnell eine grobe Prüfung durchzuführen.

Eine besondere Stelle nehmen dynamische Objekte, die ihre Position ständig ändern können. Es können andere Roboter, aber auch Menschen sein. Wenn eine permanente Kommunikation zwischen Robotern, wobei sie ihre Positionen einander mitteilen, möglich ist, so ist es zwischen einem Roboter und dem Menschen nicht ohne weiteres realisierbar. Die Sensoren spielen eine wichtige Rolle bei der Feststellung einer Bewegung der Objekte. In einem Vergleich mit der Umweltmodell können diese Objekte identifiziert und ihre mögliche weitere

Bahn vorhergesagt. Zur Zeit vorhandene Sensoren sind: Ultraschallsensoren rund um der Plattform, Laserscanner vorne, auf dem Effektor montierte Kamera und Kinect-Sensor, der seinen Platz noch nicht gefunden hat.

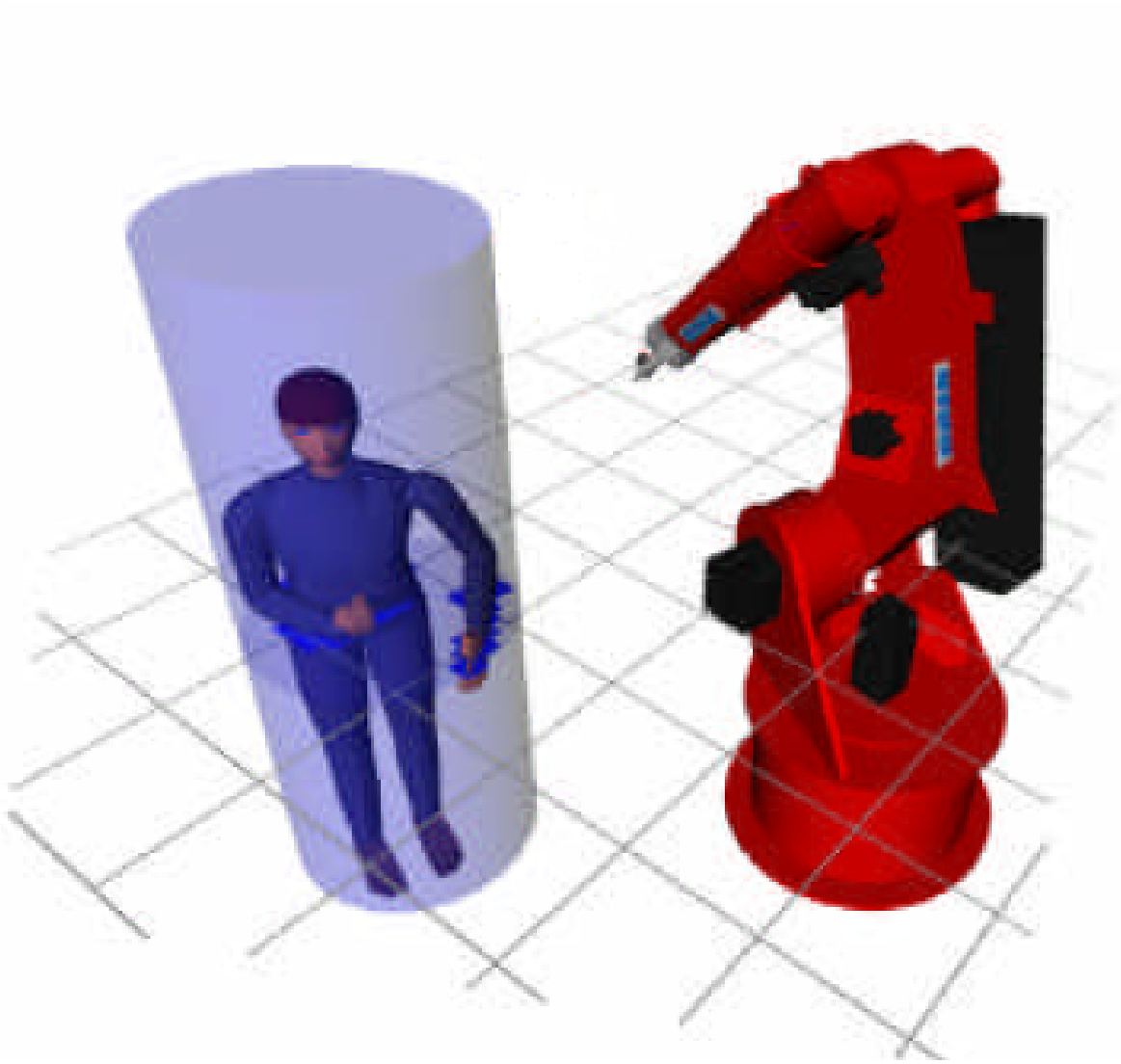

Abbildung 2.2: Kollisionsprüfung [Dresselhaus (2006)]

Da es mehrere Sensoren gibt, so können die ermittelten Objektpostionen sich unterscheiden, auch Ungenauigkeit jedes Sensors trägt dazu bei. Das Programm muss diese Unschärfen bearbeiten können und die Erkenntnisse daraus schlissen. Entsprechend zu der Veränderungen muss das Umweltmodell auf den neuesten stand aktualisiert werden.

2.3 Aktionsplaner

Den letzten Punkt in meiner Agenda belegt die Erstellung eines Aktionsplaners. Diese Aufgabe beinhaltet Ausführung selbstständiger Operationen mit Anwendung geeigneter Lösungssuchstrategien und Einbindung von Reflexen mit dazugehörigen Planänderungen, was zu einem autonomen Verhalten führen soll. Diese Herausforderung stellt sich als äußerst schwierig und bedarf einen tiefen Einsicht in den Bereich der Künstlichen Intelligenz.

Ein Planer soll eine bestimmte Problemstellung lösen und daraus resultierende Aktionen ableiten, so das eine Kette von Aktionen entsteht, die den Roboter aus einem Anfangs- zu dem Ziel-Zustand überführt. Dabei werden mehrere Aufgabenlösungsphasen benötigt [Gattringer (2011)]:

- **Zielformulierung**: Definition einer Menge von Zuständen, in denen das Ziel als erfüllt angesehen werden kann.
- **Problemformulierung**: Definition von Zuständen und Aktionen, die für die Lösung des Problems zur Verfügung stehen.
  - Singuläres Zustands-Problem, eigener Zustand und die Folgen der Aktionen sind bekannt.
  - Plurales Zustands-Problem, die Folgen der Aktionen sind bekannt, eigener Zustand aber nicht.
  - Kontingenz-Problem, eigener Zustand ist bekannt, Unsicherheit von Aktionen.
  - Explorations-Problem, Zustand und die Folgen der Aktionen sind nicht bekannt.
- **Problemlösung** mit Verwendung von geeigneter Suchstrategie.
  - Suchorientierte
  - Regelbasierte
  - Planungsbasierte
  - Konnektionistische
- **Ausführungsphase** des erstellten Aktionsplans.

Auf die Leistungsfähigkeit einer Suchstrategie wirken mehrere Kriterien:

- Vollständigkeit - wird eine Lösung immer gefunden?
- Optimalität - wird die beste Lösung (Pfadkosten) gefunden?
- Suchkosten: Zeitbedarf, Speicherplatzbedarf

3 Risiken

In allgemeinem ist der Zeitaufwand schwer vorher zu sehen. Die Einarbeitung in das ROS-Framework für Umstrukturierung vorhandener Komponente und in die Künstliche Intelligenz für Aktionsplaner kann einiges an Zeit in Anspruch nehmen. Die Umsetzung kann erst nach diesen Forschungen statt finden. Als besonders kostspielig kann sich die Auswahl geeigneter Lösungssuchstrategien für Aktionsplaner ergeben, weil der Bereich der Künstlichen Intelligenz noch relativ unerforscht ist und keine direkte Lösungswege bietet. Die einzelne Suchstrategien müssen daher erarbeitet und getestet werden.

Wegen seiner Komplexität ist der Aktionsplaner mehr unter Aussichten einzuordnen. Die tatsächliche Arbeit wird sich in erster Linie mit einfacheren Befehlen wie Bewegen oder Greifen beschäftigen, wobei die Trajekotrie nicht explizit von einem Anwender eingegeben werden soll, sondern mit dem Hinsicht auf Energieeffizienz und Kollisionsvermeidung von der Recheneinheit geplant und ausgeführt wird. Als Ziel kann eine Transportaufgabe angesehen werden, die keine exakten Bahnen wie z.B. beim Punktschweißen benötigt. Somit kann die Planung von der Recheneinheit übernommen werden.

Weil das Umweltmodell noch nicht vorliegt, ist es mit der Arbeit für Erstellung der Kollisionsprüfung abzuwarten. Dabei muss noch erforscht werden, ob die in die ROS-Framework eingebundene Pakete für eine Echtzeit-Kollisionsprüfung eignen, sonst müssen andere Wege zur Lösung dieses Problems gesucht werden. Der Hauptkriterium einer Prüfung liegt dabei bei einem Echtzeitverhalten, damit dynamisch auf die Veränderungen in der Umgebung reagiert werden kann. Der Rechenaufwand für einen Durchlauf der Prüfung legt die Zeitschrittweite und Reaktionsverzögerung fest. Es ist derzeit noch fraglich, ob die Zeitschrittweite statisch gesetzt oder dynamisch vom System berechnet werden soll. Dies könnte Auswirkungen auf die Synchronisation einzelner Komponente haben.

Literaturverzeichnis

[Dresselhaus 2006] DRESSELHAUS, Manfred: Sichere Steuerungstechnik für die Mensch-Roboter Kooperation REIS ROBOTICS (Veranst.), 2006

[Dubs 2011] DUBS, Andreas: *Entwicklung einer industriellen Robotersteuerung mit Trajektorienplanung*, HAW Hamburg, Bachelorarbeit, 2011

[Dubs 2012] DUBS, Andreas: *Zeitliche Interpolation einer linearen Robotertrajektorie mit inverser Kinematik*, HAW Hamburg, MT Ausarbeitung, 2012

[Gattringer 2011] GATTRINGER, Hubert: *Starr-elastische Robotersysteme : Theorie und Anwendungen*. 2011. – ISBN 978-3-642-22827-8

[Haun 2007] HAUN, Matthias: *Handbuch Robotik*. 2007. – ISBN 978-3-540-25508-6

[Kolbe 2011] KOLBE, Felix: *Einsatz des ROS-Frameworks zur Umgebungskartierung und Navigation des SCITOS G5-Service-Roboters*. 2011. – Ausarbeitung PJ2

[Stark 2009] STARK, Georg: *Robotik mit MATLAB*. Carl-Hanser-Verl., 2009. – ISBN 978-3-446-41962-9