Xiaoqing Cheng, Tim Lueth: Reaktive Planung und Planausführung in dem intelligenten Robotersystem KAMRO. Künstliche Intelligenz, KI 8 (3): 13-22 (1994)

Reaktive Planung und Planausführung in dem intelligenten Robotersystem KAMRO

Xiaoqing Cheng und Tim Lüth

Institut für Prozeßrechentechnik und Robotik, Universität Karlsruhe,
76128 Karlsruhe, e-mail: t.lueth@ieee.org

Roboter, die nicht nur ein reaktives Verhalten zeigen, sondern darüberhinaus komplizierte Aufgabenstellungen autonom bearbeiten können, benötigen hierfür ein geeignetes Planungs- und Planausführungssystem. Dieses System muß in eine entsprechende intelligente Planungs- und Steuerungsarchitektur des Roboters eingebettet werden. Am Beispiel des im Rahmen des SFB 314 entstandenen Robotersystems KAMRO werden Lösungsansätze für die Planung und Ausführung von Serviceaufgaben erarbeitet und erprobt.

Einleitung

Die Robotik ist seit langer Zeit eine Herausforderung für die Entwicklung und Erprobung von Verfahren der Künstlichen Intelligenz. Die ersten Anregungen zur Verknüpfung der beiden Themen wurden vor 25 Jahren in [Nilsson 69] vorgeschlagen und viele der bekannten Klassiker im Bereich der Planung wie STRIPE [Fikes 71,72], SIPE [Wilkins 83], DEVISER [Vere 83], IPEM [Ambros 83] oder PRS [Georgeff 87] hatten entweder direkt Roboter als Anwendungsgebiet vor Augen oder teilweise Systeme, die das menschliche Verhalten in einer Umwelt nachbilden sollten wie OPM [Hayes 79].

Reale Systeme sind aber auch darauf angewiesen, über Sensoren die Umwelt zu erfassen, Situationen zu interpretieren und darauf zu reagieren. Prinzipiell müssen alle diese Prozesse parallel zu Aktionen in einer sich dynamisch verändernden Welt ablaufen. Gerade die Interpretations- und Interaktionsprozesse sind dabei zusätzlich fehlerbehaftet. Zur Bewältigung der Echtzeitanforderungen und zur Trennung der unterschiedlichen Abstraktionsstufen wurde schon früh die Idee der anfänglich streng hierarchischen Planungs- und Steuerungsarchitekturen eingeführt [Albus 81]. Dieses Architekturprinzip beschreibt die funktionale Trennung von einzelnen getrennt durchführbaren Planungs- und Steuerungsaufgaben, deren Zusammenwirken über fest definierte Kommunikationsmechanismen sichergestellt wird. In dem hierarchischen Ansatz steigt der Abstraktionsgrad der Informationsrepräsentation und der lösbaren Aufgabe mit jeder Ebene.

Bis Mitte der achziger Jahre war dennoch weltweit kein Roboter in der Lage, eine ihm gestellte Aufgabe wie beispielsweise das Erfassen einer Montagesituation und einer autonomen Montageausführung unter realistischen Bedingungen zu bewältigen. Dies führte zu der Kritik, daß planenden Systeme prinzipiell nicht in der Lage wären, den Echtzeitanforderungen realer Umgebungen gerecht zu werden [Brooks 91] und statt dessen künstliche intelligente Systeme nur durch eine künstliche Evolution einfacher reaktiver Systeme [Brooks 86] erreicht werden könnten. Diese verhaltensorientierten Systeme verfügen über keine explizite

Planungskomponente, die den Sensor- und Effektoreinsatz steuert, sondern das Ein-/Ausgangsverhalten wird durch Wechselwirkung und Überlagerung grundsätzlicher Reaktionsverhalten bestimmt.

Die Erfahrungen mit verhaltensorientierten Systemen haben gezeigt, daß diese Systeme ein unübertroffenes reaktives und robustes Verhalten bei speziellen Problemen wie Kollisionsvermeidung während der Navigation zeigen. Sie sind jedoch nicht in der Lage, Handlungsketten zu erzeugen, die in komplizierten Situationen zur Problemlösung, wie in dem oben genannten Montageproblem, notwendig sind. Der verhaltensbasierte Ansatz bringt daher im Bereich des reaktiven Handelns große Vorteile, hat jedoch in Bereichen Defizite, die mit planenden Systeme problemlos bewältigt werden können. Dennoch regt die Idee der verhaltensbasierten Systeme dazu an, bestimmte Problemklassen mit einfachen Mechanismen anstatt mit aufwendigen Planern zu lösen. Eine Entwurfssystematik fehlt jedoch bisher.

Im Projekt "Karlsruher Autonomer Mobiler Roboter" KAMRO wird im Rahmen des SFB 314 seit 1985 versucht, die Vorteile eines planenden Systems mit den Vorteilen eines reaktiven Systems zu verbinden, um ein robustes autonomes Robotersystem für Service-Anwendungen (Transport, Montage, Inspektion, Reparatur) in teilstrukturierten Umgebungen zu schaffen. Ein wesentlicher Schwerpunkt hierbei ist die Realisierung der Fähigkeit, aus nicht vollständig spezifizierten Aufgabenstellungen selbständig Ausführungsstrategien abzuleiten und die Aufgaben in einer realer Umgebung ohne menschliche Eingriffe auszuführen. Vor allem in der Sensorintegration wurden im KAMRO-Projekt die meisten Erfahrungen gesammelt und die Planungsarchitektur kontinuierlich verbessert. Der Artikel beschreibt im weiteren die relevanten Probleme bei der autonomen Handhabung und die im Rahmen des KAMRO-Projektes erarbeiteten Lösungsansätze.

Das Robotersystem

Der KAMRO (Bild 1) besteht aus einer mobilen Plattform und zwei, an Galgen montierten, Manipulator-Greifer-Einheiten. Als Antriebe werden Elektromotoren verwendet. Die Plattform hat eine holonome Kinematik mit drei Freiheitsgraden. Zum aktuellen Zeitpunkt werden nur Sensoren zur direkten Bestimmung von Abständen zu Hindernissen eingesetzt. Dies sind 16 um die Plattform ringförmig angeordnete Ultraschallsensoren, ein mittig in Fahrtrichtung ausgerichteter Laser-Scanner, und ein versetzt in Fahrtrichtung ausgerichtetes passives Stereo-Sichtsystem, das über eine spezielle Optik und Kreuzkorrelationsmechanismen [Takeno 92] direkt ein Abstandsraster erzeugt. Ein taktiler Sensorring (Bumper) dient zur Notabschaltung im Kollisionsfall. Die Manipulatoren sind in einem Abstand montiert, der einen überlappenden Arbeitsbereich vorsieht. Sie haben je sechs Freiheitsgrade, und die Zweibackengreifer haben je einen. In die Greifer wurden Sensoren integriert , die jeweils drei orthogonale Kräfte und drei Momente messen. Die Greifer selbst haben keine taktilen Sensorflächen. Sowohl bei den Greifern als auch bei den Manipulatoren ist es darüberhinaus möglich, durch Messung der Motorströme zusätzliche Sensorinformation zu erhalten. Zwischen den beiden Manipulatoraufhängungen ist eine CCD-Grauwertkamera montiert, die auf den überlappenden Arbeitsraum der Manipulatoren ausgerichtet ist. Zusätzlich sind an den Greifern Miniaturkameras (Handkameras) mit einem Beleuchtungsring angebracht, der Schlagschattenprobleme ausschließt.

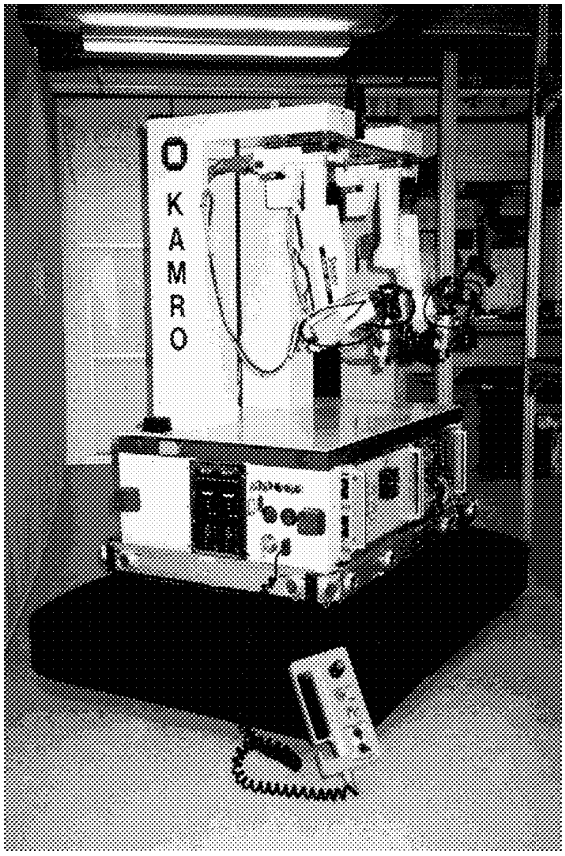

Bild 1: Das Robotersystem KAMRO

Insgesamt sind drei auf VMEbus basierte Multi-Prozessorsysteme im Einsatz, die über Bus-Koppler miteinander verbunden sind. Ein System dient zur Navigation der Plattform, eines zur Steuerung der Manipulatoren und das dritte zur Bedienung der vier Kameras. Die Kommunikation zwischen den Systemen ist für die Stereo-Navigation, die Kamera-Manipulator-Kopplung und die mobile Manipulation notwendig. Die Steuerungen sind für die Sensorvorverarbeitung und das reaktive und echtzeitfähige Verhalten der Subsysteme verantwortlich. Sie erlauben die lokale Ausführung elementarer sensorgestützter Roboteroperationen, die operationsspezifisches Wissen verwenden. Diese expliziten Elementaroperationen (EEOs) liefern nach ihrer Ausführung Informationen über den Terminierungszustand und aktuelle Sensordaten und bilden so die Schnittstelle zum eigentlichen Planungssystem des KAMRO. Das Planungssystem wird durch die Steuerungen von den Echtzeitanforderungen entlastet. Planungsentscheidungen müssen daher nur noch im echtzeitnahen Bereich (on-line) getroffen werden. Im weiteren Text wird nur auf das on-line Planungssystem eingegangen, das über ein lokales Netz mit den echtzeitfähigen, reaktiven Robotersteuerungen kommuniziert. Die Schnittstelle zwischen dem on-line Planungssystem und dem Robotersteuerungssystem ist definiert durch die folgenden EEOs:

transfer*(M: Manipulator; p: Zielposition; T: Trajektorie; v: Geschwindigkeit; a: Beschleunigung)*

fine_motion*(M: Manipulator; p: Zielposition; s: Steifigkeit)*

join*(M: Manipulator; p: Zielposition; K: Fügekonfiguration; s: Steifigkeit)*

disjoin*(M: Manipulator; p: Zielposition; K: Trennkonfiguration; s: Steifigkeit)*

dock*(p: Zielposition; K: Dockkonfiguration)*

move*(p: Zielposition; T: Trajektorie;v: Geschwindigkeit; a: Beschleunigung)*

undock*(p: Zielposition; K: Trennkonfiguration)*

set_gripper*(M: Manipulator; d: Abstand)*

grasp*(M: Manipulator; d: Abstand; f: Greifkraft)*

detach*(M: Manipulator; d: Abstand)*

Aufgabenstellung

Die Aufgabe des Robotersystems besteht darin, gleichzeitig mehrere priorisierte Aufgabenstellungen (Dienstleistungen) wie Transport, Montage, Überwachung und

Fehlerbehebung in einer industriellen Fertigungsumgebung durchzuführen. Dies kann beispielsweise bedeuten, an einem Arbeitsplatz eine Montageaufgabe durchzuführen und bei Bedarf an einem anderen Ort der Fertigungsumgebung Bestückungsfehler zu beheben. Weiterhin hat der Roboter die Aufgabe, in festen Zeitabständen Instrumente abzulesen und über Bedienelemente einen modellierten Prozeß zu steuern.

Bei der autonomen Montage [Hörmann 91] muß der Roboter eine Menge beliebig auf einem Arbeitstisch verteilter Montageteile selbständig zu einem Produkt zusammenbauen. Dabei müssen zuerst mit Hilfe der Sichtsystems die Montageteile erkannt werden. Danach müssen für die gegebene Situation vor allem die Entscheidungen, welche Montageteile zu welchem Zeitpunkt von welchen Manipulatoren bearbeitet werden, ob Teile vor dem Montieren wegen ihren aktuellen Positionen und stabilen Lagen durch Zweiarm-Kooperation umgegriffen bzw. ausgetauscht werden sollen, on-line getroffen werden. Ein weiteres Kernproblem bildet die kollisionsfreie Bewegungsplanung für das Zweiarm-Manipulatorsystem. Sie soll einen kollisionsfreien und weitergehend parallelen Einsatz der beiden Manipulatoren ermöglichen. Ebenso muß der Roboter in der Lage sein, sich kollisionsfrei zwischen einzelnen Einsatzorten zu bewegen, worauf hier jedoch nicht weiter eingegangen wird.

Da die Behebung von Bestückungsfehlern auf einem Werkstückträger durch Pick- und Place-Operationen ausgeführt werden kann, fällt sie direkt in die Kategorie der autonomen Montage. Eine Klasse der Reparaturaufgaben für Maschinenkomponenten, die durch Austausch von defekten Bauteilen ausgeführt werden können, kann ebenfalls von KAMRO durchgeführt werden.

Bei der Durchführung von Serviceaufgaben müssen die Anforderungen einer dynamischen Einsatzumgebung berücksichtigt werden. Da hier die Aufgaben mit unterschiedlichen Dringlichkeiten (mittelfristigen Echtzeitanforderungen) zu beliebigen Zeitpunkten gestellt werden können, muß das on-line Planungssystem in der Lage sein, mehrere solche asynchrone Aufgabenstellungen gleichzeitig zu verfolgen und die eventuellen Konflikte bezüglich der gestellten mittelfristigen Echtzeitanforderungen aufzulösen.

Wissensrepräsentation

Eine intelligente Planung setzt geeignete Formalismen zur Weltmodellierung und Aufgabendarstellung voraus. Basierend auf der Weltmodellierung muß die Aufgabendarstellung einerseits das zu erreichende Ziel eindeutig spezifizieren und andererseits möglichst viel Spielraum lassen, damit das Planungssystem unter Benutzung aktueller Sensorinformation und des Modellwissens situationsgerechte Ausführungsstrategien ableiten und geeignete Roboteraktionen planen und ausführen kann.

Aufgabendarstellung

Im Hinblick auf die Zielsetzung bietet sich ein impliziter Ansatz für die Aufgabendarstellung an, der nur den zu erreichenden Zielzustand einer Aufgabe, jedoch nicht den Weg beschreibt, wie dieser Zielzustand hergestellt werde kann. Beispielsweise wird eine Montageaufgabe ausschließlich durch die Manipulationseffekte an Montageobjekten spezifiziert. Es ist außerdem möglich, Aufgaben auf verschiedenen Abstraktionsebenen darzustellen. Z. B. läßt sich eine komplexe Montageaufgabe allein durch die Beschreibung der räumlichen Relationen der

Bauteile untereinander jeweils im Anfangszustand vor der Montage und im montierten Endzustand darstellen. Auf einer weniger abstrakten Ebene kann eine Montageaufgabe durch eine Sequenz (allgemeiner eine Halbordnung) von einfachen Montageoperationen dargestellt werden. Eine einfache Montageoperation kann z. B. das Montieren eines einzigen Montageteils darstellen. Weiterhin läßt sich eine einfache Montageoperation durch einen Zyklus von Objekthandhabungen *nimm_objekt_auf* (pick) und *verbinde_objekt* (place) beschreiben. Je abstrakter eine Aufgabendarstellung ist, desto mehr Planungsaufwand ist erforderlich, um die Roboteroperationen zur Aufgabenausführung zu planen. Für die autonome Montage muß man das Abstraktionsniveau der impliziten Aufgabendarstellung so wählen, daß die Abbildung auf explizite Roboteroperationen in solchen Zeitintervallen durchgeführt werden kann, die in vertretbarer Relation zu den Ausführungszeiten der geplanten Roboteroperationen stehen.

Die Grundlage für die implizite Aufgabendarstellung für KAMRO bildet der Montagevorranggraph. Ein Montagevorranggraph stellt eine Halbordnung über einfachen Montageoperationen dar. Die Halbordnung läßt sich interpretieren als die zeitlichen Vorrangrelationen unter den einfachen Montageoperationen. Eine Kante $A \rightarrow B$ in der Halbordnung bedeutet, daß die Montageoperation *B* erst dann ausgeführt werden darf, nachdem die Montageoperation *A* bereits durchgeführt worden ist Die Knoten eines Montagevorranggraphen sind die einfachen Montageoperationen. Sie beschreiben jeweils den Zielzustand einer Objekthandhabung, der durch räumliche Kontakte zwischen dem manipulierten Objekt und den Basisobjekten dargestellt wird. Als räumliche Kontakte werden die Verbindungen zwischen montagerelevanten Objektmerkmalen wie Stiften und Löchern betrachtet. Durch Angabe von Transformationen zwischen solchen Objektmerkmalen lassen sich die absoluten Positionen der manipulierten Objekte im Raum eindeutig bestimmen.

Ein Montagevorranggraph mit seinen einfachen Montageoperationen und seiner Netzstruktur repräsentiert die statischen Aspekte einer Montageaufgabe. Um die dynamischen Abläufe der Aufgabenausführung darstellen zu können, wird ein Montagevorranggraph vor der Ausführung auf ein Bedingung/Ereignis-Petrinetz abgebildet. Die Abbildung wird in zwei Schritten durchgeführt, wie es in Bild 2 gezeigt wird.

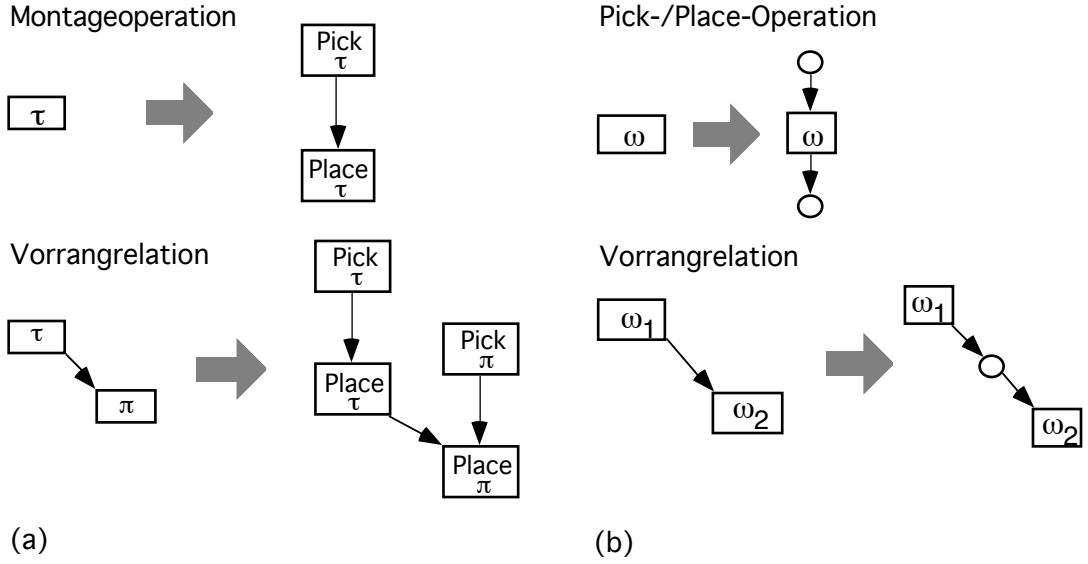

Bild 2: Abbildung von Montagevorranggraphen auf B/E-Petrinetz. (a) Aufspalten einfacher Montageoperationen in Pick- und Place-Operationen; (b) Abbildung von Pick- und Place-Operationen und ihrer Vorrangrelation in ein B/E-Petrinetz.

Weltmodellierung

Zur Weltmodellierung gehört die Darstellung der Umgebung, in der Aufgaben geplant und ausgeführt werden, der zu manipulierenden Werkstücke und des Robotersystems selbst. Neben geometrischer und kinematischer Beschreibung steht hier die funktionale Modellierung im Vordergrund. Das umfaßt unter anderem die funktionale Beschreibung der montagerelevanten Objektmerkmale, der Ausführungsstrategien impliziter Elementaroperationen und die Fähigkeiten der verschiedenen Sensoren.

Das Hauptproblem bei der Weltmodellierung ist die Beherrschung extrem hoher Komplexität. Um diese Anforderung zu erfüllen, wurde ein Prolog-basiertes System zur objektorientierten Wissensdarstellung auf der Grundlage von Frames [Minsky 75] entwickelt. In einem Frame lassen sich die Aussagen über ein Objekt in einer gemeinsamen Datenstruktur zusammenfassen. Durch eine beliebige Anzahl von Slots in einem Frame können neben aktuellen Wertebelegungen für Objekteigenschaften auch Defaultwerte und prozedurale Anhängsel (procedural attachments) spezifiziert werden. Prozedurale Anhängsel sind kleine Programme, die bei einem Zugriff auf das entsprechende Slot ausgeführt werden, beispielsweise für die Aktualisierung des Umweltzustandes nach der Ausführung einer Elementaroperation (post condition).

Zusätzlich werden die Mechanismen der Objektorientierten Programmierung in die Wissensdarstellung eingebaut. Mit Frames werden sowohl Objektklassen als auch Objektinstanzen in Hierarchien dargestellt. Eine Objektklasse definiert die Eigenschaften aller zu ihr gehörenden Unterklassen und Instanzen. Jede Objektklasse kann mehrere Objektinstanzen besitzen, die die eigentlichen Individuen mit gleicher Eigenschaftstruktur, jedoch mit unterschiedlichen Wertebelegungen repräsentieren. In solchen Vererbungshierarchien werden die allgemeineren Eigenschaften und Prozeduren den Vorgängern zugeordnet und an die Nachfolger vererbt, während die individuellen Ausprägungen von den Nachfolgern und letztendlich von den Objektinstanzen getragen werden.

Die Wissensbasis für die on-line Planung von KAMRO enthält fünf Klassenhierarchien, die im folgenden kurz erläutert werden:

- Die Klassenhierarchie *actor* stellt neben den geometrischen und kinematischen Eigenschaften der mobilen Plattform, der Manipulatoren und der Greifer vor allem die individuellen Basiskoordinatensysteme, die aktuellen Positionen und die verfügbaren Sensorinformationen zur Repräsentation der aktuellen Zustände dieser Aktuatoren dar.
- In der Klassenhierarchie *eo* werden die Parameter der Elementaroperationen (Pick- und Place-Operation und EEOs) und die Routinen zur dynamischen Berechnung dieser Parameter spezifiziert. Zur Laufzeit werden die geplanten Elementaroperationen als Objektinstanzen in dieser Klassenhierarchie erzeugt.
- Die Fähigkeiten und die geometrischen Parameter der Sensoren werden in der Klassenhierarchie *sensor* dargestellt. Zur Sensoreinsatzplanung werden zusätzlich die objektspezifischen Merkmale der Werkstücke beschrieben.
- In der Klassenhierarchie *workcell* werden die Arbeitsstationen in der Einsatzumgebung des KAMROs modelliert. Unter anderem werden hier die Dockpositionen des KAMROs an den jeweiligen Arbeitsstationen beschrieben.
- Die zu manipulierenden Werkstücke werden in der Klassenhierarchie *workpiece*

dargestellt. Neben der Objektgeometrie werden die möglichen stabilen Lagen der Werkstücke beschrieben. In Bezug auf die jeweilige stabile Lage werden die montagerelevanten Merkmale wie Stifte, Löcher und mögliche Greifpositionen spezifiziert. Die eigentlichen Werkstückteile werden als Objektinstanzen generiert, nachdem sie von dem Sichtsystem erkannt wurden.

Planungsarchitektur

Die Planungsarchitektur spielt eine entscheidende Rolle bei der Entwicklung des Planungssystems. Sie definiert nicht nur die funktionale Struktur, sondern bietet vor allem auch einen Rahmen von Kooperationsmethoden an, in dem die verschiedenen Lösungen der Funktionskomponenten zu einem funktionsfähigen Planungssystem integriert werden können. Um komplexe Planungssysteme für die Robotersteuerung zu entwickeln, bieten sich hierarchische Planungsarchitekturen an.

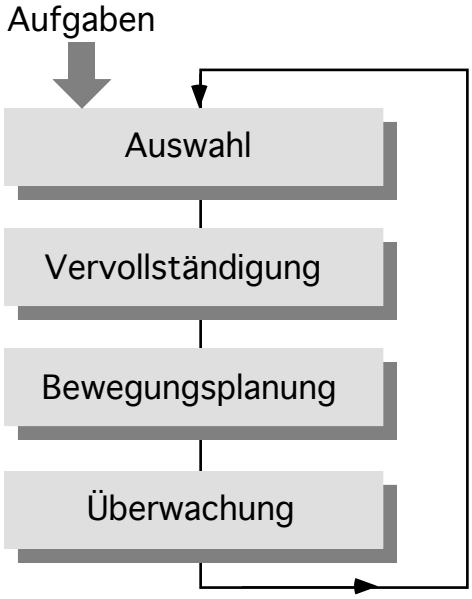

Bild 3: Ablauf der on-line Planung für die autonome Montage

Für die autonome Montage können die folgenden Hierarchieebenen identifiziert werden (Bild 3):

- Auswahl von ausführbaren Montageoperationen,
- Vervollständigung der impliziten Montageoperationen,
- Planung expliziter Roboterbewegungen, und
- Überwachung und Synchronisation der Ausführung.

Charakteristisch an der hierarchischen Planungsarchitektur ist ihr zyklischer Ablauf. Auf jeder Planungsebene wird eine Aufgabe nicht vollständig durchgeplant, sondern jeweils nur die Entscheidung für den nächsten Schritt getroffen. Zur Ausführung dieser Entscheidung wird die nächste unterliegende Planungsebene beauftragt. Nach der Ausführung wird abhängig vom jeweils erreichten Zustand eine weitere Entscheidung getroffen. Der Informationsaustausch geschieht immer nur zwischen zwei direkt benachbarten Planungsebenen. Dadurch wird ein eng gekoppelter zyklischer Ablauf von Planung, Ausführung und Wahrnehmung und damit das reaktive Systemverhalten realisiert.

Auswahl ausführbarer Montageoperationen

In einem B/E-Petrinetz als Spezifikation einer Montageaufgabe stellen die Eingabenstellen einer Transition die Vorrangbedingungen dar. Das Feuern einer Transition entspricht der Aktivierung einer Pick- oder Place-Operation. Nach der Ausführung einer Pick- oder Place-Operation werden Marken in den entsprechenden Ausgabenstellen erzeugt. Basierend auf diesem Darstellungsprinzip läßt sich die aktuelle Menge der ausführbaren Pick- und Place-Operationen in Bezug auf die Vorrangrestriktionen dem momentanen Montagezustand entsprechend berechnen. Diese berechnete Menge wird dann an die nächste Planungsebene der Vervollständigung weitergegeben.

Vervollständigung impliziter Montageoperationen

Die Vervollständigung versucht, einer impliziten Pick- oder Place-Operation aus der Menge der ausführbaren durch Klassifikation verfügbare Manipulatoren zuzuordnen und diese implizite Operation durch roboterspezifische Parameter zu vervollständigen. Dieser Prozeß wird immer wieder gestartet, sobald ein Manipulator für Aufgabenausführungen zur Verfügung steht. Bei der Vervollständigung sollen vor allem die Verklemmungssituationen bezüglich der Betriebsmittelzuordnung vermieden werden.

Die Vervollständigung wird in zwei aufeinanderfolgenden Klassifikationsschritten durchgeführt. In der ersten, ablaufspezifischen Klassifikation wird jeweils ein Pick-und-Place-Zyklus einer Objekthandhabung betrachtet. Dabei werden nur diejenigen Pick-Operationen ausgewählt, deren ihre nachfolgende Place-Operationen bezüglich der Vorrangrelationen des B/E-Petrinetzes unmittelbar ausgeführt werden können. Da eine Pick-Operation (mindestens) einen Manipulator belegt und die nachfolgende Place-Operation den Manipulator wieder freigibt, soll diese Strategie die Wirkung haben, daß ein belegter Manipulator möglichst bald wieder freigegeben wird. Auch aus diesem Grund wird jede Place-Operation in der ablaufspezifischen Klassifikation ausgewählt.

In der zweiten, objektspezifischen Klassifikation werden nur die im ersten Schritt ausgewählten Operationen betrachtet. Dabei ist jede Place-Operation direkt ausführbar, da ihre entsprechende Pick-Operation den zweiten Klassifikationsschritt bereits bestanden hat und ihre Parameter, wie ausführender Manipulator, gegriffene Objektinstanz und Greifposition, übernommen werden können. Einer Pick-Operation wird in der objektspezifischen Klassifikation zuerst eine Objektinstanz zugeordnet. Abhängig von der jeweiligen Aufnahme- und Ablageposition der zugeordneten Objektinstanz und deren relativen Lagen zu den Arbeitsräumen der beiden Manipulatoren wird festgestellt, ob eine Austausch- bzw. Umgreifoperation durch Zweiarm-Kooperation erforderlich ist oder die Pick- und die nachfolgende Place-Operation von einem einzigen Manipulator ausgeführt werden kann. Auf der Basis dieser Entscheidung wird ein momentan verfügbarer Manipulator zugeteilt bzw. werden die unterschiedlichen Funktionen der beiden Manipulatoren bei einer Zweiarm-Kooperation bestimmt. Anschließend wird die kollisionsfreie Greifposition an der Objektinstanz bestimmt. Falls hierbei wegen in der Nähe liegenden Objekten keine Objektinstanz gefunden wird, die kollisionsfrei gegriffen werden kann, wird eine Kollisionsbehandlung gestartet. Dafür werden, gegebenenfalls rekursiv, Pick- und Place-Operation zum Entfernen eines direkt benachbarten Objektes generiert und in das B/E-Petrinetz eingefügt.

Für die objektspezifische Klassifikation wird ein graphenbasierter Algorithmus entwickelt [Hörmann 92], der für eine implizite Pick-Operation durch Best-first-Suche in einem Graphen aller möglichen Klassifikationsergebnissen eine der folgenden drei vervollständigten Elementaroperationen generiert:

- ***pick_up***$(O_I$*: Objektinstanz; O.S: Stabile Lage; M: Manipulator; g(M): Greifposition)*
- ***exchange***$(O_I$*: Objektinstanz;* $O.S_1$, $O.S_2$*: Stabile Lage;* M_1, M_2*: Manipulator;* $g(M_1)$, $g(M_2)$*: Greifposition)*
- ***regrasp***$(O_I$*: Objektinstanz;* $O.S_1$, $O.S_2$, $O.S_3$*: Stabile Lage;* M_1, M_2*: Manipulator;* $g_1(M_1)$, $g(M_2)$, $g_2(M_1)$*: Greifposition)*

Eine Place-Operation ***place***$(O_1, B_1, ..., B_n$*: Montageobjekt; ρ: Verbindungsrelation)* wird dabei wie folgt vervollständigt:

- ***put_down***$(O_1, B_1, ..., B_n$*: Objektinstanz; ρ: Verbindungsrelation; M: Manipulator; g(M): Greifposition)*

Planung expliziter Bewegungen

Für jede vervollständigte Elementaroperation wird auf dieser Planungsebene eine Sequenz von expliziten Elementaroperationen (EEOs) generiert. Dabei werden die aufgabenspezifischen Parameter, wie stabile Lagen der Objektinstanzen und Greifpositionen relativ zu Objektinstanzen, in TCP(Tool Center Point)-spezifische Bewegungsparameter transformiert. Beispielsweise wird für die vervollständigte Elementaroperation ***pick_up*** die EEO-Sequenz zur Aufnahme eines Montageobjektes generiert:

set_gripper → transfer → fine_motion → grasp → fine_motion → transfer

Eine wesentliche Aufgabe dieser Planungsebene ist die kollisionsfreie Bewegungsplanung für das Zweiarm-Manipulatorsystem. Die Zielsetzung ist, möglichst ohne Verzögerung die kollisionsfreien Bewegungsbahnen der EEOs ***transfer*** sowohl für unabhängige Einarm-Operationen als auch für Zweiarm-Kooperationen zu planen. Die Hauptschwierigkeit hierbei ist, daß sich die beiden Manipulatoren zusammen mit den gegriffenen Objekten gegenseitig als dynamische Hindernisse darstellen. Dazu kommt es noch, daß für unabhängige Einarm-Operationen keine zeitliche Korrespondenz der Transfer-Bewegungen vorgegeben wird.

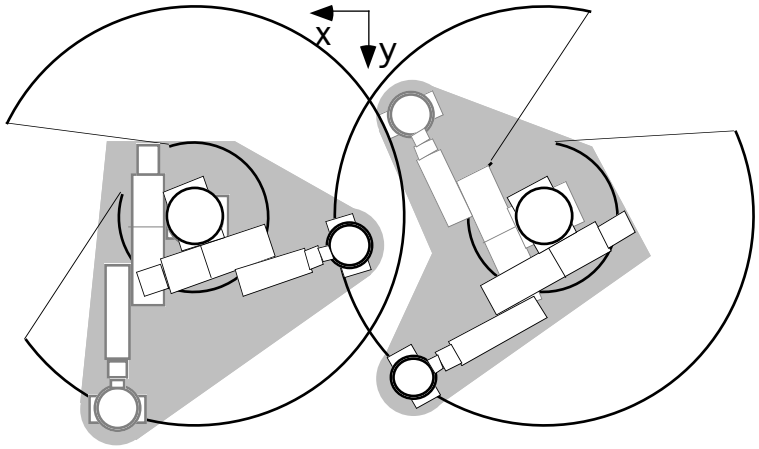

Bild 4: Swept-Regionen der Manipulatoren für die on-line kollisionsfreie Bewegungsplanung

Um Kollisionen zwischen einem Manipulator und Montageobjekten auf dem Arbeitstisch zu vermeiden, werden Bewegungsbahnen nur mit senkrechten und waagrechten Segmenten jeweils zur Arbeitstischoberfläche geplant. Die senkrechten Bewegungen werden ausgeführt, um über die Montageteile auf dem Arbeitstisch anzurücken und sie zwischen der Arbeitstischoberfläche und einer horizontalen Bewegungsebene zu transportieren, die hoch genug über dem Arbeitstisch liegt. Der Algorithmus für die on-line kollisionsfreie Bewegungsplanung benutzt deshalb ein zweidimensionales Modell und basiert auf sogenannten Swept-Regionen der Manipulatoren (Bild 4) [Cheng 93]. Die Bewegungsbahnen der Manipulatoren werden durch

Polygonzüge dargestellt. Die Swept-Region eines Manipulators für eine Transfer-Bewegung *SR(M, PayLoad, Path)* ist definiert als ein Polygon im zweidimensionalen Modell, der von dem Manipulator *M* zusammen mit dem gegriffenen Objekt *PayLoad* bei der Bewegung des TCP vom Startposition P_S entlang der Bewegungsbahn *Path* = (P_S, ..., P_g) zur Zielposition P_g aufgespannt wird.Für eine Transfer-Bewegung ist die Swept-Region für die Zeitdauer zwischen ihrer Aktivierung und dem Empfangen der Fertigmeldung vom Echtzeitsteuerungssystem definiert. Danach wird die Swept-Region dem neuen Manipulatorzustand entsprechend aktualisiert.

Die Bewegungen der Manipulatoren sind garantiert kollisionsfrei, wenn sich die Swept-Regionen der beiden Manipulatoren nicht überschneiden. Ausgehend von einer Bewegungsaufforderung an Manipulator *M* von P_S nach P_g und der aktuellen Swept-Region des anderen Manipulators SR_O wird zunächst eine "direkte" Bahn generiert, die entweder aus einer geraden Linie oder einem Polygonzug $(P_S, ..., P_g)$ um die innere Arbeitsraumgrenze besteht. Darauf basierend wird dann die Swept-Region $SR(M, P, (P_S, ..., P_g))$ berechnet. Durch Überschneidungstests [Preparata 85] zwischen den beiden Swept-Regionen SR_O und *SR(M)* wird die direkte Bahn so modifiziert, daß die beiden Swept-Regionen sich nicht mehr überschneiden. Für Situationen, in denen eine kollisionsfreie Bahn für den Manipulator *M* im Moment nicht gefunden werden kann, ist eine Koordinierungsstrategie notwendig, die im nächsten Abschnitt vorgestellt wird.

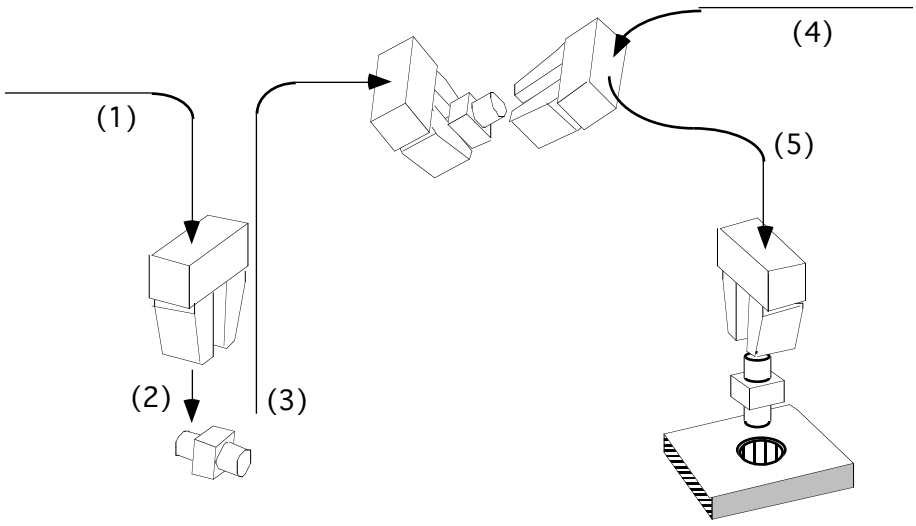

Bild 5: Austausch und Umgreifen eines Montageteils durch Zweiarm-Kooperation

Für Zweiarm-Kooperationen müssen nicht nur die kollisionsfreien Bewegungen der Manipulatoren, sondern auch die nominale Position und Orientierung eines Objektes während des Umgreifens bzw. des Austausches on-line geplant werden (Bild 5). Hierbei werden hauptsächlich die Bewegungseinschränkungen der beiden Manipulatoren in einer geschlossenen kinematischen Kette berücksichtigt. Unterstützt durch eine off-line kinematische Analyse kann on-line mit geringem Aufwand jeweils abhängig von der aktuellen Umgreifkonfiguration der beiden Greifer relativ zum Objekt eine günstige Position im gemeinsamen Arbeitsraum bestimmt werden, so daß die beiden Manipulatoren ohne kinematische Restriktion an ihre Umgreifpositionen in einer symmetrischen Orientierung anfahren können [Cheng 93]. Basiert

auf diesem Ergebnis werden die Bewegungen der Manipulatoren geplant. Nachdem das auszutauschende bzw. umzugreifende Objekt aufgenommen wurde, erreichen die beiden Manipulatoren jeweils zuerst eine ausgezeichnete Position außerhalb des gemeinsamen Arbeitsraums, die als Rest-Position des jeweiligen Manipulators genannt wird. An dieser Stelle wird das auf Swept-Region basierte Verfahren zur kollisionsfreien Bewegungsplanung ausgeschaltet. Aus diesen Rest-Positionen werden die Umgreifpositionen auf direkten Bahnen angefahren und die eigentliche Zweiarm-Operation durchgeführt. Anschließend kehren die beiden Manipulatoren an die Rest-Positionen zurück, danach werden die Bewegungen der Manipulatoren wieder als unabhängige Einarm-Operationen betrachtet.

Überwachung und Synchronisation

Zur Überwachung und Synchronisation der Ausführung von geplanten expliziten Elementaroperationen beim Echtzeit-Robotersteuerungssystem werden Operations-Warteschlangen jeweils für die Manipulatoren und die mobile Plattform verwaltet. Bei Zweiarm-Kooperationen müssen manche EEOs, die von unterschiedlichen Manipulatoren ausgeführt werden, in ihrer Ausführungsreihenfolge synchronisiert werden. Beispielsweise darf der erste Manipulator das Objekt bei einer Umgreifoperation erst dann loslassen, nachdem der zweite Manipulator das Objekt bereits gegriffen hat. Für diesen Zweck werden Synchronisationsmarken zwischen den EEOs in die Warteschlangen eingefügt. Eine Synchronisationsmarke definiert ein Prädikat in der Regel über die erfolgreiche Durchführung einer bestimmten EEO. Eine EEO hinter einer Synhronisationsmarke wird so lange von ihrer Ausführung verzögert, bis das entsprechende Prädikat erfüllt ist.

Mit der Ausnahme von ***transfer*** werden die Parameter aller anderen EEOs zum Zeitpunkt des Einfügens in die Warteschlange vollständig bestimmt. Für eine ***transfer***-EEO wird bei der Sequenzgenerierung nur die zu erreichende Zielposition berechnet. Unmittelbar vor ihrer Aktivierung wird die kollisionsfreie Bewegungsbahn nach dem im letzten Abschnitt beschriebenen Verfahren bestimmt. In manchen Situationen, beispielsweise wegen gleichzeitigen Einsatzes der beiden Manipulatoren an zu nahe liegenden Arbeitspositionen, kann eine kollisionsfreie Bewegungsbahn nicht gefunden werden. In diesem Fall wird durch eine Koordinierungsstrategie der Einsatz der Manipulatoren senquentialisiert. Zu diesem Zweck wird jedem Manipulator eine Zustandsvariable zugeordnet. Sie kann einen der folgenden drei Zustände einnehmen:

- *idle*: in diesem Zustand führt der Manipulator keine EEO aus.
- *waiting*: die Ausführung einer transfer-EEO wird suspendiert, da eine kollisionsfreie Bewegungsbahn im Moment nicht gefunden werden kann.
- *busy*: eine EEO wird gerade von dem Manipulator ausgeführt.

Für einen Manipulator M im Zustand *idle* oder *waiting* wird aufgrund einer transfer-EEO der Bewegungsplanungsalgorithmus vom letzten Abschnitt aufgerufen. Falls dabei eine kollisionsfreie Bewegungsbahn gefunden wird, wird die Swept-Region von M entsprechend aktualisiert, die transfer-EEO durch ein Aktivierungskommando an das Robotersteuerungssystem gestartet, und der Manipulator M bekommt den Zustand *busy*. Andernfalls geht der Manipulator M in den Zustand *waiting* über, und danach wird der Zustand des anderen Manipulators M_O analysiert. Für den Fall, daß M_O sich im Zustand *idle* befindet, wird eine direkte Bahn für M_O generiert, um ihn aus dem gemeinsamen Arbeitsraum zu bringen.

Falls M_O sich im Zustand *busy* befindet, muß der Manipulator M jetzt auf M_O warten, und der Bewegungsplanungsalgorithmus wird wieder für M aufgerufen, sobald sich die Swept-Region von M_O geändert hat. Anhand aller möglichen Zustandsübergängen läßt sich die Verklemmungsfreiheit dieser Koordinierungsstrategie nachweisen.

Integration erweiterter Planungsfähigkeiten

Bis jetzt wurde die grundlegende Planungsarchitektur des KAMROs am Beispiel der autonomen Montage vorgestellt. Auf der Basis der autonomen Montage wird die Erweiterung der Fähigkeit des on-line Planungssystems in Bezug auf die Fehlertoleranz und den Einsatz des KAMROs für Serviceaufgaben in einer industriellen Fertugungsumgebung beschriebn.

Handkamera-Einsatzplanung

Zur Erhöhung der Zuverlässigkeit von Fügeoperationen werden bei KAMRO Handkameras für die Manipulatoren eingesetzt. Das zweidimentionale Sichtsystem für die Handkameras erkennt statt Objekte nur die Objekmerkmale wie Ecken und Löcher von Objekten, da Aufnahmen relativ nahe an Objekten gemacht werden und deshalb ein relativ kleines Sichtfeld besitzen. Nachdem eine Handkamera über ein Objektmerkmal positioniert ist, kann sie die Position und Orientierung dieses Merkmals relativ zur aktuellen TCP-Position sehr schnell und mit einer sehr hohen Genauigkeit bestimmen.

Die Handkameras werden kurz vor einer Greifoperation im Rahmen einer Pick-Operation und kurz vor einer Fügeoperation für eine Place-Operation zur Feinbestimmung der Greifposition bzw. der Fügeposition eingesetzt. Der Handkamera-Einsatz wird im Rahmen einer Transfer-Bewegung (***trans_he***) integriert. Für die Einsatzplanung der Hand-Kameras wird bezüglich einer stabilen Lage eines Objektes eine Menge von sogenannten objektspezifischen Merkmalen OSFs (Object Specific Features) definiert. Ein OSF besteht aus einem oder mehreren Objektmerkmalen, die von einer Hand-Kamera erkannt werden können, und ist mit einem eignen Koordinatensystem vorgesehen. Charakteristisch am einem OSF ist, daß sich die Position eines Objektes eindeutig aus der Position eines OSFs bestimmen läßt. Für eine handkamera-gestützte Transfer-Bewegung wählt die Einsatzplanung zunächst ein günstiges OSF aus der Menge der OSFs des zu greifenden Objektes bzw. des Basisobjektes einer Fügeoperation bezüglich der aktuellen Objektlage aus, positioniert die Handkamera über das OSF und aktiviert das Sichtsystem zur Merkmalerkennung. Die TCP-Position, an der die Handkamera eine Aufnahme macht, wird Schnappschuß-Position genannt. Aus der durch die Handkamera bestimmten relativen Position des OSFs zur aktuellen TCP-Position wird die Zielposition ebenfalls im aktuellen TCP-Koordinatensystem neu berechnet and anschließend angefahren.

Da eine Handkamera ein OSF in einer beliebigen Orientierung erkennen kann, wird dieser rotatorische Freiheitsgrad bei der Bestimmung der Schnappschuß-Position zur Optimierung ausgenutzt. Mögliche Kriterien dafür sind:

- Der TCP an der Schnappschuß-Position soll einen minimalen translatorischen Abstand zur Zielposition haben.
- Der TCP an der Schnappschuß-Position soll die gleiche Orientierung einnehmen wie an der Zielposition, so daß nach der Handkamera-Aufnahme die Zielposition durch eine

möglichst rein translatorische Bewegung erreicht werden kann.

Durch Experimente stellt es sich heraus, daß unter Anwendung des letzteren Kriteriums eine bessere Positioniergenauigkeit an der Zielposition erreicht werden kann. Somit steht die Orientierung des TCPs vor der Bestimmung der Schnappschuß-Position fest. Deshalb wird sie auch für die Auswahl des günstigen OSFs herangezogen. Mit dieser vorgegebenen Orientierung wird das günstigeste OSF so aus der Menge bestimmt, daß der TCP an der entsprechenden Schnappschuß-Position einen möglichst kurzen Abstand zur Zielposition hat.

Bei der Bestimmung des günstigesten OSFs muß ein weiteres Problem gelöst werden. Wegen der einfachen Binär-Bildverarbeitung der Handkameras können beispielsweise manche Montageteile bzw. ihre Objektmerkmale in einem montierten Zustand wegen zu geringem Kontrast zu ihren Basisobjekten nicht erkannt werden, diese Montageteile dienen aber selbst als Basisobjekte für Fügeoperationen von weiteren Montageteilen. Zur Lösung dieses Problems werden die Mengen der OSFs der Objekte dynamisch verwaltet. Nach der Montage eines Objektes, das in seinem montierten Zustand nicht mehr von einer Handkamera erkannt wird, wird die Menge der OSFs als leer gekennzeichnet. In diesem Fall versucht die Einsatzplanung dann das günstigeste OSF aus der OSF-Menge eines seiner Basisobjekte auszuwählen. Die Suche nach den jeweiligen Basisobjekten wird in der Darstellung des aktuellen Montagezustandes in der Wissensbasis (Klasse *workpiece*) durchgeführt, sie kann im allgemeinen rekursiv verlaufen.

Automatische Behandlung von Ausnahmesituationen

Unerwartete Einflüsse der Realwelt auf die Montageumgebung können Ausnahmesituationen verursachen, die während der Durchführung einer Montageaufgabe vom intendierten Ablauf abweichen und in der Regel zu einer Störung führen. Beispiele dafür sind das Herausrutschen eines Teils aus dem Greifer aufgrund zu geringer Kontaktkräfte oder die ungewollte Verschiebung von Teilen aufgrund von Kollisionen. Eine Fehlerbehandlung auf der Planungs- und Planausführungsebene kann Fehlersituationen meistern, die durch ein pur reaktives Verhalten bezüglich Parameterabweichung nicht bewältigt werden können. Beispielsweise wird ein bereits passierter Fehler erst zu einem späteren Zeitpunkt durch die Sensorik erkannt.

Das entwickelte Verfahren zur automatischen Behandlung von Ausnahmesituationen bei Montagevorgängen umfaßt einen sequentiellen Ablauf von modellbasierter und sensorunterstützter *Fehlererkennung*, *Fehlerdiagnose* und *Fehlerbehebung*. Als Sensor wird die Overheadkamera eingesetzt. Die aktuellen TCP-Positionen und die aktuellen Greiferöffnungen werden als Sensorinformation interpretiert.

Die Fehlererkennung ist EEO-spezifisch und basiert auf dem Prinzip eines Soll-Ist-Vergleichs zwischen den Modelldaten, die als Parameter einer EEO ihre Nachbedingungen darstellen, und den Umgebungsdaten, die mit Hilfe von Sensoren extrahiert werden. Ausnahmesituationen liegen vor, wenn die Abweichungen dieser Wertepaare außerhalb eines gewissen Toleranzbereichs liegen. Für die realisierten EEOs lassen sich operationsspezifische Fehlerklassen als Zielmenge einer Fehlererkennung definieren.

Die Fehlerdiagnose versucht nun, auf der Grundlage dieser Fehlerklassen die Ursachen sowie die Auswirkungen von Fehlerzuständen zu bestimmen. Im ersten Schritt wird der kausale

Zusammenhang in Form eines Fehlerursachenmodells der betroffenen EEO innerhalb des Montagevorgangs bestimmt. Ein *Fehlerursachenmodell* beschreibt alle Bedingungen, die für eine erfolgreiche Ausführung einer EEO relevant sind, und besteht aus drei Komponenten: einer Anzahl von *Fehlerknoten*, einem *Aktionsknoten* und einer Anzahl von *Zielknoten*. Die Fehlerknoten beschreiben die Vorbedingungen, die vor der Aktivierung einer EEO erfüllt sein müssen und charakterisieren somit mögliche Ursachen von Ausführungsfehlern. Im Gegensatz dazu beschreiben die Zielknoten die jeweiligen Nachbedingungen und stellen mögliche Auswirkungen von Ausführungsfehlern auf das Ergebnis einer EEO dar. Der Aktionsknoten verbindet die Fehlerknoten mit den Zielknoten und repräsentiert damit die Kausalität.

Mit Hilfe der Fehlerursachenmodelle der bisher ausgeführten Roboteraktionen (EEOs) läßt sich ein globaler *Fehlerursachenbaum* erstellen. Die Wurzel dieses Baums ist dabei durch den Aktionsknoten der EEO definiert, bei der die Echtzeitsteuerung eine Fehlersituation signalisiert hat. Der eigentliche Aufbau des Baums beruht auf dem Prinzip, daß manche Fehlerursachen (Fehlerknoten) sich auf die fehlerhaften Nachbedingungen (Zielknoten) seiner Vorgänger-EEOs zurückführen lassen. Durch Verknüpfen der Fehlerknoten mit den entsprechenden Vorgänger-Zielknoten entsteht ein globaler Fehlerursachenbaum. Das Finden einer Erklärung für einen Fehler entspricht nun der Suche nach einem plausiblen Weg von der Wurzel des Baums zu einem Blattknoten, der die eigentliche Fehlerursache repräsentiert. Für jeden Weg von der Wurzel zu einem Blattknoten läßt sich ein Wahrscheinlichkeitswert berechnen, der angibt, wie plausibel diese Kausalkette den erkannten Fehler erklären kann. Dieser Wahrscheinlichkeitswert bestimmt sich als Produkt aus der Fehlerwahrscheinlichkeit W des Blattknotens und den Reduktionsfaktoren R_i der n Zielknoten (n≥0), die im Weg enthalten sind:

$$\text{P(Fehlerursache)} = W \cdot \prod_{i=1}^{n} R_i \quad .$$

Im Fall von mehreren diagnostizierten Fehlerursachen kann somit eine Reihenfolge festgelegt werden, so daß die nachgeschaltete Fehlerbehebung immer die wahrscheinlichste Fehlerursache zuerst bearbeitet.

Ausgehend von den Daten der Fehlerdiagnose versucht die Fehlerbehebung, einen Fehlerzustand durch eine lokale Neuplanung zu beseitigen, so daß die ursprüngliche Montageaufgabe wieder fortgesetzt werden kann. Die Neuplanung heißt deshalb lokal, da sie nur den Zielzustand des laufenden Pick-und-Place-Zyklus herzustellen versucht (forward recovery). Zu diesem Zweck existiert eine Bibliothek von Fehlerbehebungsplänen, die durch eine entsprechende Parameterisierung sämtliche von der Fehlerdiagnose gelieferten Fehlerursachen abdeckt. Die Fehlerbehebungspläne dienen unter anderem dazu, noch laufende Aktionen definiert abzubrechen, einen Einsatz der Overheadkamera vorzubereiten und die Objektmanipulationen zur Fehlerbehebung durch Pick- und Place-Operationen durchzuführen. Kommt es bei der Ausführung von Fehlerbehebungsaktionen (Pick-und-Place-Zyklen) selbst wieder zu Fehlerzuständen, so werden diese rekursiv durch das hier beschriebene Verfahren bearbeitet. Zu einem Zeitpunkt können demnach mehrere Fehlerbehandlungen ineinander geschachtelt ablaufen (recoverable recovery).

Einsatz von KAMRO für Serviceaufgaben

Um den KAMRO als Serviceroboter in einer industriellen Fertigungsumgebung einsetzen zu

können, wird das on-line Planungssystem um die Fähigkeit der reaktiven Einsatzplanung und der automatischen Generierung von Aktionsplänen für Wartungs- und Reparaturaufgaben erweitert.

Die Zielsetzung der reaktiven Einsatzplanung ist die gleichzeitige Verfolgung von mehreren asynchronen Aufgabenstellungen. Dabei müssen neue Aufgaben während des laufenden Betriebs entgegengenommen und die Ausführungsreihenfolge dynamisch bestimmt bzw. verändert werden, damit das Robotersystem flexibel auf dringlichere Aufgaben reagieren und die Zeitvorgaben für deren Durchführung einhalten kann. Als Lösungsansatz dient ein neues Konzept der Unterbrechungsbehandlung und der Zeitabschätzung für die Ausführung von strukturierten Aufgaben.

Die Unterbrechungsbehandlung hat die Aufgabe, den Planungs- und Planausführungsprozeß zu einem beliebigen Zeitpunkt in einem sicheren Zwischenzustand zu unterbrechen, so daß das Robotersystem für die Ausführung einer anderen Aufgabe zur Verfügung steht. Die unterbrochene Aufgabe muß zu einem späteren Zeitpunkt wieder fortgesetzt werden können. Die Unterbrechungsbehandlung umfaßt ein definiertes Abbrechen der laufenden Roboteraktion, eine lokale Neuplanung zum Erreichen eines sicheren Zwischenzustandes und schließlich die Fortsetzung der unterbrochenen Aufgabenausführung zu einem späteren Zeitpunkt.

Nachdem die laufende Roboteraktion definiert unterbrochen wurde, hat die lokale Neuplanung einen *sicheren* Zwischenzustand zu erreichen. Ein sicherer Zwischenzustand bedeutet in diesem Zusammenhang erstens, daß die zu handhabenden Bauteile sich in stabilen Konfigurationen entweder auf der Arbeitsfläche oder in der Baugruppe befinden. Zweitens müssen die Greifer der beiden Manipulatoren frei sein, damit sie für die Ausführung einer anderen Aufgabe zur Verfügung stehen. Darüber hinaus muß die unterbrochene Aufgabenausführung zu einem späteren Zeitpunkt nach der ursprünglichen Aufgabenspezifikation fortgesetzt werden. Für diesen Zweck existieren zwei Klassen von lokalen Aktionsplänen, die entweder den laufenden Pick-und-Place-Zyklus bis zum Ende durchführen, oder, gegebenenfalls durch Ablegen eines bereits gegriffenen Objekts auf den Arbeitstisch, den Zustand vor dem aktuellen Pick-und-Place-Zyklus einnehmen. Für die Entscheidung, die lokale Neuplanung vorwärts oder rückwärts durchzuführen, ist eine möglichst kurze Ausführungszeit des jeweiligen lokalen Aktionsplans ausschlaggebend. Die Neuplanung heißt hier lokal, weil sie den laufenden Pick-und-Place-Zyklus nicht verläßt. Der durch die lokale Neuplanung erreichte sichere Zwischenzustand läßt sich in der B/E-Petrinetz-Darstellung einer Aufgabenspezifikation festhalten. Aus diesem Grund kann das Robotersystem nach der Abarbeitung einer dringlichen Aufgabe die unterbrochene Aufgabenausführung wie gewohnt fortsetzen.

Um die Entscheidung für eine Änderung der Ausführungsreihenfolge treffen zu können, muß zuerst die Zeit, die für die Durchführung momentaner Aufgabe benötigt wird, abgeschätzt und mit der noch zur Verfügung stehenden Zeit verglichen werden. Auf Grund der abstrakten Aufgabenspezifikation kann die Ausführungszeit einer Aufgabe wegen der Möglichkeit der parallelen Ausführung (Halbordnung) und der indeterministischen Einflüsse auf die sensorintegrierten Elementaroperationen nicht vorher exakt bestimmt werden. Für die Zeitabschätzung wird die Ausführungszeit einer impliziten Pick- oder Place-Operation approximativ durch ein Intervall von minimalem und maximalem Erfahrungswert dargestellt [Freedmann 91]. Basiert auf dieser Zeitdarstellung wird die Ausführungszeit einer Aufgabe unter Berücksichtigung der beiden Extremfälle, nämlich die Sequentialisierung und die maximale Parallelisierung (durch n Roboterarme) der Operationsausführung, und der

Scheduling-Strategien der Vervollständigung abgeschätzt.

Exemplarisch für strukturierte Serviceaufgaben betrachten wir die Klasse von Wartungs- und Reparaturaufgaben, die durch Bearbeitung von Verschleißteilen bzw. Austausch von defekten Teilen ausgeführt werden können. Der Ablauf umfaßt die drei aufeinander folgenden Phasen: Demontage, bis die Verschleißteile bzw. die defekten Teile zugänglich sind, Bearbeitung bzw. Austausch, und anschließend wieder Zusammenbau der demontierten Teile. Durch die Erweiterung von Montagevorranggraphen um implizite Demontageoperationen läßt sich diese Klasse von Serviceaufgaben durch einen allgemeinen Vorranggraphen mit impliziten Montage- und Demontageoperationen darstellen. Analog zu einer Montageoperation läßt sich eine Demontageoperation ebenfalls in eine implizite Pick- und Place-Operation aufspalten.

Ein wichtiges Problem bei den Reparaturaufgaben ist, daß man im Allgemeinen nicht vorhersagen kann, welche Teile defekt sind und wann dieser Fehler passieren wird. Nachdem eine Diagnose über die defekten Teile vorliegt, kann erst dann eine Reparaturaufgabe bestimmt und der Vorranggraph als Aktionsplan dafür generiert werden. Wegen der Vielzahl der Fehlersituationen und der Nichtvorhersagbarkeit ist es weder sinnvoll noch machbar, für jede mögliche Fehlersituation einen Aktionsplan off-line vorzubereiten. Deshalb wurde ein on-line Verfahren für die automatische Erzeugung von Spezifikationen für Reparaturaufgaben entwickelt. Der Ausgangspunkt ist ein Montagevorranggraph, der den Zusammenbau einer Maschinenkomponete beschreibt. Abhängig von der aktuellen Diagnose für die defekten Teile kann ein allgemeiner Vorranggraph für die Reparaturaufgabe automatisch generiert werden. Der Teilvorranggraph für die Demontagephase wird durch Umkehrung der Montageoperationen in notwendige Demontageoperationen und Invertierung der entsprechenden Vorrangrelationen des zugrundliegenden Montagvorranggraphen erzeugt. Die Austauschphase besteht aus den Demontageoperationen für die defekten Teile und darauf folgenden Montageoperationen der neuen Ersatzteile. Für die anschließende Montagephase wird der Teilgraph der demontierten Werkstückteile des Montagevorranggraphen übernommen. Dabei wird die Umkehrbarkeit der zugrundeliegenden Montageoperationen vorausgesetzt. Die Umkehrbarkeit einer Montageoperation besteht darin, daß ein Montageteil, statt von der Startposition zur Zielposition, umgekehrt von der Zielposition zur Startposition durch Pick- und Place-Operationen zurückbewegt werden kann.

Falls nicht jede zugrundliegende Montageoperation umkehrbar ist, wird eine Menge von Montagevorranggraphen als Ausgangspunkt benutzt, die die verschiedenen Möglichkeiten für den Zusammenbau einer Maschinenkomponente darstellen. In diesen Montagevorranggraphen werden die Montageoperationen jeweils als umkehrbar oder nicht umkehrbar markiert. Die Teilgraphen für die drei Ausführungsphasen werden unter Berücksichtigung dieser Information generiert.

Verteilte Planung und Steuerung

Die im bisherigen Teil vorgestelle Planungs- und Steuerungsarchitektur des Robotersystems KAMRO besteht aus der reaktiven Robotersteuerung zur Ausführung der Elementaroperationen und dem oben beschriebenen echtzeitnahen Planungssystem, die wie in Bild 2 hierachisch angeordnet sind. Die Planungs- bzw. Steuerungssysteme sind zentral für alle beteiligten Komponenten (Fahrzeug, Manipulatoren, Kameras etc.) des gesamten Robotersystems

zuständig. Dies bringt hinsichtlich der zeitoptimierten Einsatzplanung oder des Einsatzes beider Manipulatoren deutliche Vorteile. Auf der anderen Seite hat ein zentrales Planungssystem auch Nachteile wie beispielsweise die Geschlossenheit eines solchen Systems. Aus diesem Grund wird momentan an der Entwicklung einer neuen verteilten Planungs- und Steuerungsarchitektur KAMARA gearbeitet. Diese Architekturform sieht für jede aktive Einheit (Agent) im Robotersystem eine eigene Planungs- und Steuerungsarchitektur vor, die miteinander kommunizieren und kooperieren sowie ihre Pläne koordinieren. Neben den Vorteilen des zentralen Systems soll KAMARA folgende Eigenschaften besitzen [Lüth 94]:

- *Modularität:* Durch einen vorgebenen Rahmen für die Agenteninteraktion können einzelne Komponenten weitgehend getrennt entwickelt und getestet werden.
- *Verteiltes Wissen:* Jeder Agent (Fahrzeug, Manipulatoren, aktive Sichtsysteme) besitzt das seinen Fähigkeiten angemessene Wissen. Eine globale Wissensbasis kann bei Bedarf zusätzliche Information zur Verfügung stellen.
- *Fehlertoleranz:* Solange systeminhärente Redundanz durch mehrere ähnliche Agenten vorliegt, soll das Gesamtsystem auch im Falle von funktionsuntüchtigen Komponenten eine gestellte Aufgabe lösen.
- *Integrierbarkeit:* Ohne Änderungen in der Steuerungsarchitektur sind neue Kooperationsformen zwischen Agenten implementierbar.
- *Erweiterbarkeit:* Neue Systemkomponenten (Agenten) können hinzukommen, ohne daß Änderungen der Systemarchitektur oder der Protokolle vorgenommen werden müssen.

Ein wichtiger Aspekt bei den verteilten Steuerungsarchitekturen ist darüberhinaus noch das Realzeitverhalten bei kritischen Kooperationen wie die Handhabung von schweren Objekten mit zwei Manipulatoren. In diesem Umfeld müssen noch neue Konzepte wie [Längle 94] entwickelt werden. Bisher gibt es nur sehr wenige Arbeiten die sich mit der verteilten Planung aus Ausführung mit realen Robotersystemen beschäftigen. Offene Fragen gibt es dagegen noch viele.

Bewertung

Im KAMRO-Projekt wurde ein hoch integriertes Robotersystem erfolgreich für autonome Montage und Serviceaufgaben in einer industriellen Fertigungsumgebung entwickelt. Mit seinen reaktiven Planungsfähigkeiten kann das Robotersystem unter Berücksichtigung spezieller Anforderungen seiner Einsatzumgebung komplexe Aufgaben mit einem hohen Grad an Selbständigkeit, Flexibilität und Zuverlässigkeit durchführen.

Bei der Entwicklung des on-line Planungssystems spielt die Integration der Lösungskonzepte für die einzelnen Systemkomponenten eher eine wichtigere Rolle als die eigentliche Entwicklung dieser Lösungen. Als Resultat ergibt sich die Planungsarchitektur aus der Systemintegration. Erst mit umfassender Erfahrung über die Aufgabenzerlegung und die gegenseitigen Wechselwirkungen zwischen den Teilaufgaben lassen sich effiziente, domänenspezifische Planungsarchitekturen entwickeln. Dabei müssen geeignete hierarchische Planungsebenen definiert werden, so daß eine praxisrelevante Aufgabe stufenweise und weitergehend unabhängig voneinander auf diesen Hierarchieebenen geplant werden kann. Im Zentralpunkt steht hierbei die Möglichkeit der Integration von Sensorinformation auf verschiedenen Abstraktionsebenen.

Die Planung für Serviceaufgaben in einer dynamischen Umgebung kann nur in einem on-line Betrieb durchgeführt werden. Dafür ist eine zyklisch ablaufende Struktur mit eng gekoppelter Wahrnehmung, Planung und Ausführung erforderlich. Dabei spielt das Zeitverhältnis zwischen der Planung und der Ausführung eine zentrale Rolle. In einer on-line Planung ist die Planungszeit nur dann akzeptabel, wenn sie in der Regel kürzer und im schlimmsten Fall vergleichbar mit der Ausführungszeit für die geplante Roboteroperation ist. Erst unter dieser Voraussetzung kann ein Robotersystem sinnvoll auf nicht vorhersehbare Ereignisse reagieren.

Durch den verteilten Ansatz für Planung und Steuerung können zukünftige komplexe Robotersysteme nicht nur mit viel höherer Qualität und Produktivität entwickelt werden, diese Systeme können auch den Menschen durch intelligente Mensch-Maschine-Kooperation bei der Aufgabenausführung mit einbeziehen. Dadurch werden die Effizienz, Flexibilität und Zuverlässigkeit dieser Systeme bedeutend erhöht und somit auch völlig neue Anwendungsgebiete erschlossen werden können.

Literatur

[Albus 81] Albus, J.S., Barbera,A.J., Nagel,R.N. (1981): Theory and practice of hierarchical control. IEEE Comp. Soc. Int. Conf., 1981, pp. 18-39.

[Albus 91] Albus, J.S. (1991): Outline for a Theory of Intelligence. IEEE Trans. on System Man and Cybernetics, 21, 3, pp. 473-509.

[Ambros 83] Ambros-Ingerson, J.A.; Steel, S. (1983): Integrating Planning, Execution and Monitoring. AAAI National Conf. on Artificial Intelligence. Reprinted in Allen, J.; Hendler, J.; Tate, A. (Ed.), Readings in Planning (1990), Morgan Kaufmann Publishers, pp. 735-740.

[Antsaklis 94] Antsaklis, P. (Ed.) (1994): Intelligent Control • Final Report of the IEEE TF on Intelligent Control. IEEE Control System Magazine, June.

[Brooks 86] Brooks, R.A. (1986): A Robust Layered Control System for a Mobile Robot. IEEE Trans. on Robotics and Automation, 2, pp. 14-23.

[Brooks 91] Brooks, R.A. (1991): Intelligence without reason. International Joint Conference on Artificial Intelligence IJCAI ´91, pp. 569-595.

[Cheng 93] Cheng, X.(1993): Executing elementary assembly operations by a two-arm robot. JMSE International Conference on Advanced Mechatronics, Tokyo, Japan, 2.-4. August 1993. pp. 396-401.

[Fikes 71] Fikes, R.E.; Nilsson, N.J: (1971): STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 2, pp. 189-208.

[Fikes 72] Fikes, R.E.; Hart, P.E.; Nilsson, N,J, (1972): Learning and Executing Generalized Robot Plans. Artificial Intelligence, 3(4). Reprinted in Allen, J.; Hendler, J.; Tate, A. (Ed.), Readings in Planning (1990), Morgan Kaufmann Publishers, pp. 251-288.

[Freedman 91] Freedman, P.(1991): Time, Petri nets, and robotics. IEEE Trans. on Robotics and Automation, Vol. 7, No. 4, August 1991. pp. 417-433.

[Georgeff 87] Georgeff, M.P.; Lansky, A.L. (1987): Reactive Reasoning and Planning. National Conference on Artificial Intelligence, Menlo Park, CA. Reprinted in Allen, J.; Hendler, J.; Tate, A. (Ed.), Readings in Planning (1990), Morgan Kaufmann Publishers, pp. 729-734.

[Hayes 79] Hayes-Roth, B.; Hayes-Roth, F. (1979): A Cognitive Model of Planning. Cognitive Science, . Reprinted in Allen, J.; Hendler, J.; Tate, A. (Ed.), Readings in Planning, Morgan Kaufmann Publishers, pp. 245-262.

[Hörmann 91] Hörmann, A., Rembold, U.(1991): Development of an advanced robot for autonomous assembly. IEEE International Conference on Robotics and Automtion, Sacramento, USA, 1991. pp.2452-2457.

[Hörmann 92] Hörmann, A.(1992): On-line planning of action sequences for two-arm manipulator systems. IEEE International Conference on Robotics and Automtion, Nice, France, 1992. pp.1109-1114.

[Laengle 94] Laengle, Th; T.C. Lueth (1994): Decentralized Control of Distributed Intelligent Robots and Subsystems. AIRTC Symp. on Artificial Intelligence in Real-Time Control, Valencia, Spain, October, 3-5, pp. to appear.

[Lueth 94] Lueth, T.C.; Laengle, Th. (1994): Task Description, Decomposition, and Allocation in a Distributed Autonomous Multi-Agent Robot System. IROS IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, Munich, Germany, Sep. 12-16, pp. to appear.

[Minsky 75] Minsky, M.(1975): A framework for representing knowledge. In: Winston, P.(Ed.) The psychology of computer vision. McGraw-Hill, New York, 1975. pp. 211-277.

[Nilsson 69] Nilsson, N.J. (1969): A Mobile Automaton: An Application of Artificial Intelligence Techniques. IJCAI Int. Joint Conf. on Artificial Intelligence, Washington D.C., USA.

[Preparata 85] Preparata, F. P.; Shamos, M. I.(1985): Computational geometry. Springer Verlag, New York, 1985.

[Takeno 92] Takeno, J.; Mizuguchi, N.; Sorimachi, K.(1992): Realisation of a 3D vision mobile robot that can avoid collision with moving obstacles. Robotersysteme 8, 1-12 (1992).

[Vere 83] Vere, S.A. (1983): Planning in Time: Windows and Durations for Activities and Goals. IEEE Trans. on Pattern Recognition and Machine Intelligence, 5(3). Reprinted in Allen, J.; Hendler, J.; Tate, A. (Ed.), Readings in Planning (1990), Morgan Kaufmann Publishers, pp. 297-318.

[Wilkins 83] Wilkins, D.E. (1983): Representation in a Domain-Independent Planner. IJCAI Int. Joint Conf. on Artificial Intelligence, Menlo Park, CA, pp. 733-740.