



# Programmieren durch Vormachen für Assistenzsysteme – Schweiß- und Klebebahnen intuitiv programmieren

Programming by Demonstration for Assistive Systems – Intuitive Programming of Welding and Gluing Trajectories

Christian Meyer, Rebecca Hollmann, Christopher Parlitz, Martin Hägele, Fraunhofer-Institut für Produktionstechnik und Automatisierung, Stuttgart

**Zusammenfassung** Assistenzsysteme, seien sie mobil oder stationär, unterstützen den Werker im Produktionsprozess, beispielsweise bei Schweißarbeiten in kleinen Losgrößen. Für die Programmierung dieser Prozesse existiert keine etablierte Lösung, da die Vorgehensweisen der Industrierobotik nicht übertragbar sind. Das Konzept des Programmierens durch Vormachen ist ein Lösungsansatz, der mit der Programmierumgebung InTeach verfolgt wird. Die Komponenten des Systems und die damit mögliche schnelle und intuitive Programmierung werden am Beispiel von zwei As-

sistenzsystemen vorgestellt. ▶▶▶ **Summary** Assistant systems, both mobile or stationary, support the worker in his production process, e.g., when welding in small lot sizes. For programming these tasks no viable solution exists, the methods of industrial robotics cannot be transferred. The concept of programming by demonstration is an approach that is followed with the programming environment InTeach. The fast and intuitive programming possible with this environment is presented with respect to two assistant systems.

**KEYWORDS** J.7 [Computer Applications: Computers in other Systems: Industrial Control] robot programming, walk through programming, human machine interface, human robot interface, assistant system / Roboterprogrammierung, Walk Through Programmierung, Mensch-Maschine Schnittstelle, Assistenzsystem

## 1 Einführung

Assistenzsysteme, die den Werker bei industriellen Prozessen in kleinen Stückzahlen unterstützen, sind keine klassischen Serviceroboter nach der Definition der International Federation of Robotics [1]. Ebenso wenig aber sind die für Industrieroboter etablierten Techniken, etwa zur Programmierung, auf sie übertragbar. Wie Serviceroboter müssen sie auf eine schnell erlernbare, intuitive Weise programmiert werden können. Für Serviceroboter existieren bereits viele solcher

Schnittstellen, etwa zur Definition von Zielpunkten mobiler Systeme über eine grafische Oberfläche.

Assistenzsysteme im industriellen Bereich, bspw. der in Bild 1 dargestellte rob@work [2], benötigen zur Ausführung ihrer Aufgaben aber darüber hinausgehende Informationen; das Reparaturschweißen einer Schiffsschraube etwa bedarf exakter Bahninformationen in Referenz zum Werkstück. Die Programmierung dieser Teilaufgabe ist sehr ähnlich zur Programmierung von Industrierobotern. Da aber Losgrö-

ßen, Infrastruktur und Personal sich grundlegend von der Situation in der Industrierobotik unterscheiden, müssen neue Lösungen gefunden werden.

Ziel der hier dargestellten Arbeiten ist, den Prozessexperten in die Lage zu versetzen, die Schweiß- oder Klebeanwendungen eines Assistenzsystems intuitiv zu programmieren. Dazu wird in Abschnitt 2 der Stand der Technik zur Programmierung von Robotersystemen aufgearbeitet, Abschnitt 3 stellt die Komponenten der intuitiven Pro-



**Bild 1** Assistenzroboter rob@work beim assistierten Lichtbogenschweißen.

grammierungsumgebung InTeach dar, neben der reinen Bewegungsführung und -aufnahme auch Nutzerschnittstellen über Sprache oder PDA. Die Integration in das Steuerungsframework Go wird skizziert. In Abschnitt 4 werden Anwendungen an zwei Robotersystemen dargestellt. Abschnitt 5 bietet einen Ausblick.

## 2 Stand der Technik – Programmierung von Robotersystemen

Assistenzsysteme im industriellen Umfeld sind häufig mit einem Manipulatorarm ausgestattet, neben Handlings- und Transportaufgaben können sie für einfache Klebe- oder Schweißprozesse eingesetzt werden. Während für symbolische Ansätze etwa zur Programmierung von Pick-and-Place Aufgaben Vorgehensweisen bekannt sind [3], ist für die Definition der Schweiß- oder Klebetrajektorie des Roboterarms bisher keine optimal Lösung gefunden.

### 2.1 Bahnprogrammierung im industriellen Umfeld

Für die Programmierung von Industrierobotern gibt es drei vorwiegend genutzte Ansätze, definiert etwa von der amerikanischen Occupational Safety and Health Administration [4]:

- „Lead-Through Programming“ mittels eines Programmierhandgerätes,
- „Walk-Through Programming“, bekannt von Lackierrobotern, und
- „Offline Programming“.

In größeren Anlagen werden Roboterprogramme über Offline-Programmiersysteme wie em-Workplace oder IGRIP definiert und über Programmierhandgeräte angepasst. Kleinere Robotersysteme werden komplett über Programmierhandgeräte ‚geteacht‘.

Bei der Walk-Through Programmierung wird der Roboter nicht über ein Bediengerät in X, Y und Z dirigiert, sondern direkt am TCP (Tool Center Point) geführt. Diese Methode der Programmierung wurde bei frühen Lackierrobotern genutzt, um fließende Bewegungen zu erzeugen: die ausbalancierten Roboter wurden bei deaktivierten Motoren von den Programmierern durch Ihre Bahn gezogen, diese wieder abgespielt. Heute ist diese Programmierung eine Nischenlösung.

Dazu kann beispielsweise der Whole Arm Manipulator der Fa. Barrett Technologies [5] leicht geführt werden, Hilfsfunktionen wie virtuelle Wände lassen sich nut-

zen. Die Bahnen können anschließend wieder abgespielt werden. KUKA Roboter können mit der Option Safe Handling geordert werden, mit einem Joystick ausgestattet, der das sichere Verfahren ermöglicht [6]. Für Manutec Roboter kann eine Steuerung der Fa. mz robotlab verwendet werden, diese ist in der Lage, kraftsensitive Prozesse auszuführen, etwa zum Schleifen oder Entgraten von Bauteilen [7]. Der Roboter kann über das Führen programmiert werden. Für die Messung der auftretenden Prozess- und Führungskräfte wird ein Kraft-Momenten-Sensor verwendet.

### 2.2 Defizite

Wenn es um die Bahndefinition von Schweiß- und Klebeaufgaben geht, müssen auch die Manipulatoren von Assistenzsystemen programmiert werden. Die Standard-Methoden der Lead-Through Programmierung oder der Offline-Programmierung sind nicht einsetzbar, kleine Stückzahlen, fehlende Infrastruktur sowie unstrukturierte Umgebung kennzeichnen die neuen Anwendungsfelder.

Die markterhältlichen Lösungen zur Walk-Through Programmierung behandeln zwei Punkte nicht: zum einen können Genauigkeiten von bis zu 1/10 mm etwa bei Schweißprozessen nicht erreicht werden, des Weiteren ist eine intuitive Adaption der Bahn nach der Programmierung nicht möglich. Die aktuell vorgestellte Programmierungsumgebung InTeach stellt Lösungen für diesen zweiten Punkt dar.

## 3 Komponenten der intuitiven Programmierungsumgebung InTeach

Um eine schnelle und intuitive Programmierung von Assistenzsystemen für Klebe- und Schweißprozesse zu ermöglichen, wurde die Programmierungsumgebung InTeach realisiert. Sie besteht aus einer Anwendung zur Programmierung durch Vormachen und einer An-

wendung zur Adaption der aufgenommenen Bahn.

### 3.1 Gesamtkonzept und Datenfluss

Die InTeach Programmierumgebung besteht aus einem Zellenrechner, der über Verbindungen einerseits mit dem Robotersystem, andererseits mit einzelnen Modulen kommuniziert. Wie in Bild 2 dargestellt sind über TCP/IP-Verbindungen einerseits ein Sprachdialogsystem, andererseits ein indu-

strietauglicher PDA angeschlossen. Über den Monitor kann mittels einer 3-D Schnittstelle interagiert werden, ein Kraft-Momenten-Sensor ist über eine AD-Wandlerkarte integriert. Die Verbindung zur Robotersteuerung ist je nach Typ mittels einer seriellen, TCP/IP- oder bus-basierten Kommunikation aufgebaut.

Der Datenfluss in der Anwendung wird in Bild 3 dargestellt. Die während des Vormachens aufgenommenen Bahndaten werden zu-

nächst automatisch zu Meta-Daten verarbeitet. Auf diesen Daten können im Anschluss ein Prozess-Expertensystem und der Nutzer arbeiten. Die näheren Möglichkeiten werden im folgenden Abschnitt beschrieben.

### 3.2 Programmierung durch Vormachen

Zur Definition der Bahn bewegt der Nutzer den Manipulator durch Ziehen, Schieben und Drücken am Tool Center Point des Roboters, dieser bewegt sich entsprechend. Realisiert wird diese Bewegung in der InTeach-Programmierungsumgebung nach dem Admittanz-Verfahren über einen Kraft-Momenten-Sensor [8], die gemessenen Kräfte des Nutzers werden in zurückzulegende Wege umgerechnet. Dieses einfache Verfahren ist aufgrund der Randbedingungen ausreichend; programmiert werden können Assistenzsysteme auch mit Standardkinematiken, was zu sehr geringen Regelungstakten (bis hinab zu 10Hz) führt. Die damit nicht realisierbare Dynamik der Regelung etwa bei Werkstückkontakt muss auf anderem Wege erreicht werden, etwa durch passiv nachgiebige Elemente am Roboter.

Notwendig aber ist weiterhin die Kompensation der Werkstückmasse, die bei Änderung der Orientierung Momente aufbringt. Dazu ist eine Messequenz vorgesehen, die ohne Nutzerinteraktion Masse und Schwerpunkt des Werkstücks berechnet [9]. Weiter unterstützend bei der Bahnprogrammierung wirken Assistenzfunktionen wie virtuelle Wände, Gitter oder Tastfunktionen.

Dieses Bewegungsverfahren wurde auf Assistenzsystemen verschiedener Größen realisiert und weiterentwickelt. Angefangen beim Mitsubishi PA-10 des rob@work, siehe Abschnitt 4.1, der mit 100 Hz über ein Controller-Board angesteuert wird, bis zu Systemen mit Industrierobotern, etwa einem Reis RV40 oder einem KUKA KR210. Der Reis-Roboter wird dabei mittels einer TCP/IP Schnittstelle im

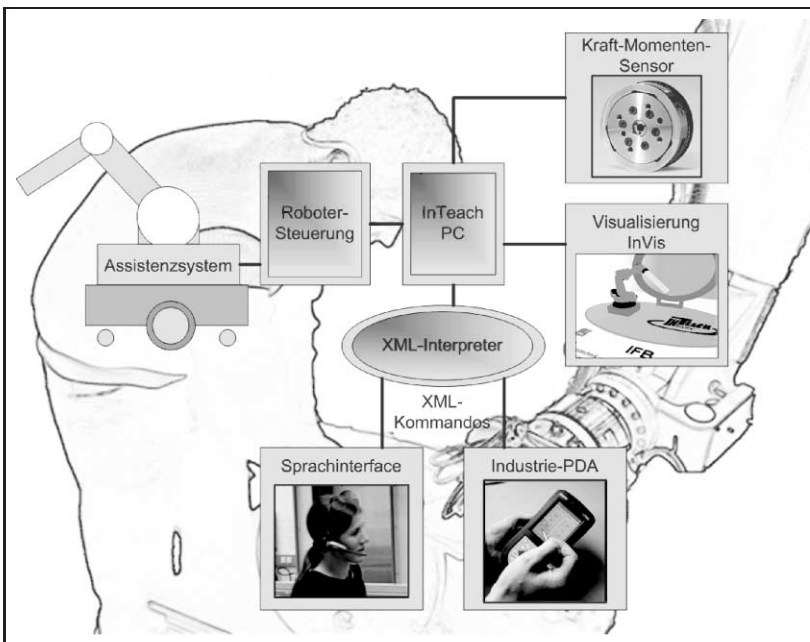


Bild 2 Komponenten der Programmierumgebung (nur Armsteuerung).

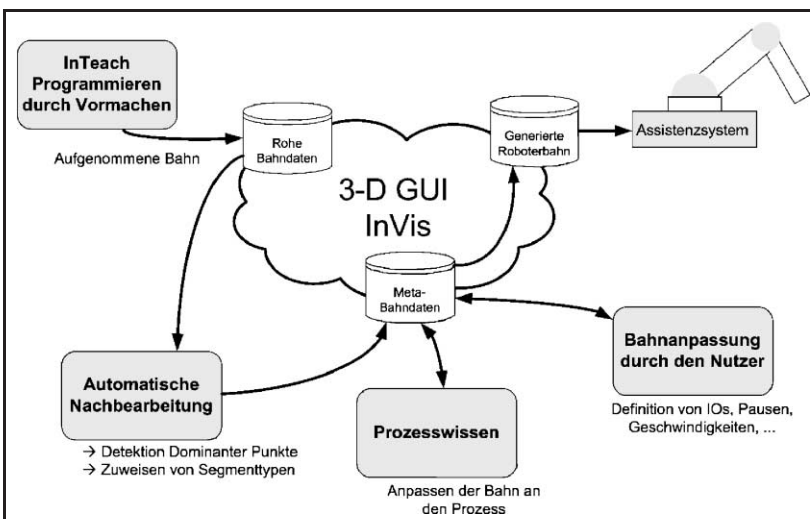


Bild 3 Datenfluss in der InTeach Programmierumgebung.

XML-Format bei 50 Hz angesteuert, der KUKA seriell mit etwa 10 Hz. Schnellere Schnittstellen von beiden Herstellern werden aktuell implementiert.

### 3.3 Nachbearbeitung der Bahn

Im Anschluss an die Aufnahme der Bahn muss eine Nachbearbeitung stattfinden. Gründe dieser Notwendigkeit sind:

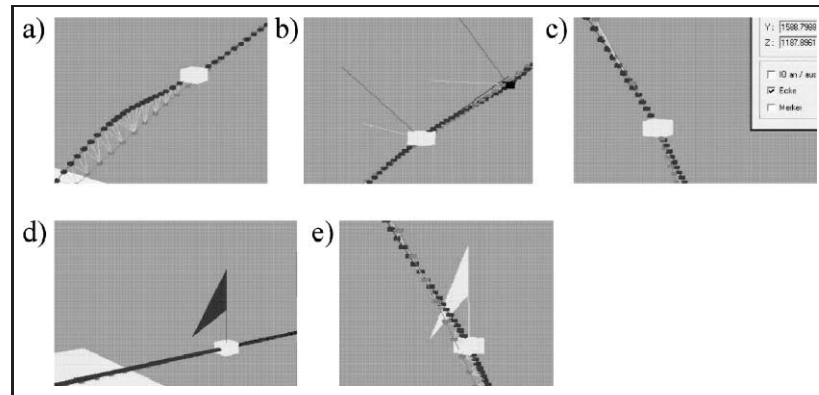
- Genauigkeiten: Die Bahn muss an das Werkstück angepasst werden, um geforderte Toleranzen einzuhalten.
- Ein- und Ausgänge: Der Nutzer muss, abhängig von der Bahnposition, den Zustand der angeschlossenen Geräte definieren, etwa Klebepistole oder Schweißanlage.
- Geschwindigkeit: Ein Geschwindigkeitsprofil muss definiert werden.

Der Prozess, um diese Parameter zu definieren, ist in einen automatischen und einen manuellen Teil separiert. Nach einer Kompression der Bahn werden Segmente generiert, Geraden und Splines (NURBS, non-uniform rational b-splines). Durch diese Abstraktion wird der Pfad zu einem vom Nutzer vorzugebenen Grad geglättet.

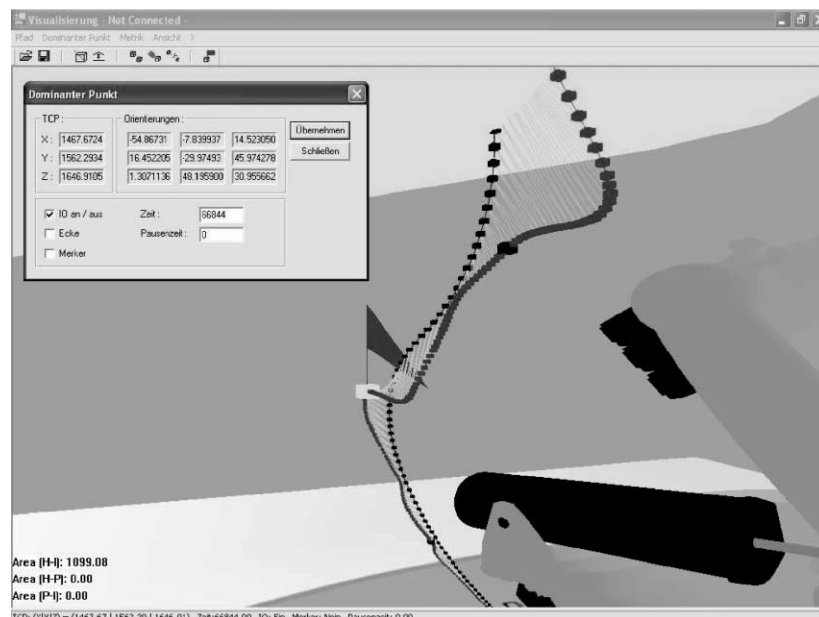
Im Anschluss kann der Nutzer den Pfad auf verschiedene Arten anpassen, wie in Bild 4 beschrieben.

Diese Funktionalitäten werden dem Nutzer durch die 3-D Nachbearbeitungssoftware InVis, siehe Bild 5, zur Verfügung gestellt. Das Interface beruht auf der OpenGL Grafikbibliothek und dient neben der Bahnanpassung der Verifikation der in Simulation dargestellten Bahn.

**Abweichungsmetrik** Zwischen der anfangs aufgenommenen Bahn und der durch Kompression, Segmentierung und Nachbearbeitung entstandenen Trajektorie werden Abweichungen bestehen. Genauso, und noch weitaus interessanter, werden Abweichungen zu der eigentlich gewünschten Bahn, etwa mit herge-



**Bild 4** Verschiedene Möglichkeiten der Bahnnachbearbeitung: a) *Translation*: Durch Drag and Drop von Dominanten Punkten wird die Bahn abgepasst. Punkte können gelöscht und hinzugefügt werden. b) *Orientierung*: Über die GUI kann die Orientierung verändert werden, ebenso können Inertialsensoren genutzt werden. c) *Ausprägung des Segments*: Die Ausprägung eines Segments kann zwischen Linie und Spline umgeschaltet werden. d) *Digitale Ausgänge*: Der Nutzer kann an beliebigen Dominanten Punkten die digitalen Ausgänge schalten. e) *Pausen*: An Dominanten Punkten können Pausen definiert werden.



**Bild 5** Grafische Schnittstelle InVis zur Bahnnachbearbeitung.

brachten Methoden programmiert, bestehen. Als Indikator für das Ausmaß dieser Abweichungen wurde eine Metrik in InVis integriert.

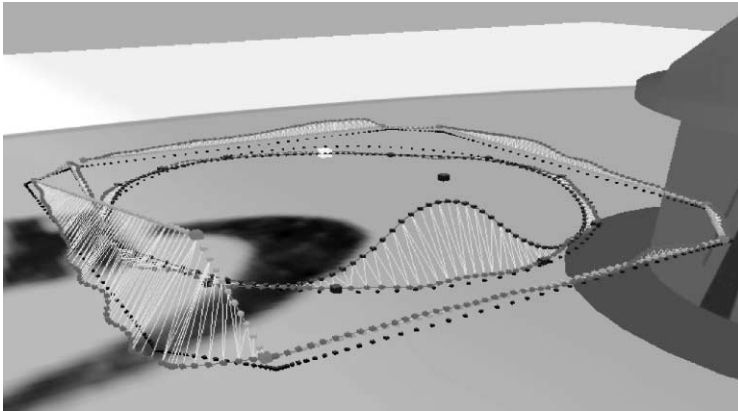
Die einfachste Form der Abweichungsmetrik ist die Summe über alle Abweichungen zwischen zwei Bahnpunkten auf der neuen und alten Bahn:

$$g = 1/e = 1/\sum \|frame_{new} - frame_{old}\|$$

Diese Metrik kann nicht angewandt werden, da Listen mit unterschiedli-

cher Punktezahl verglichen werden. Wir schlagen alternativ die Berechnung der minimalen Fläche zwischen den beiden Trajektorien vor. Diese Fläche wird über Triangulation zwischen den Bahnpunkten berechnet. Natürlich ist dieses Ergebnis aber noch kein eindeutiger Indikator für die Qualität der Trajektorie:

- Um einen eindeutigen Qualitätsindikator nutzen zu können, muss eine Referenztrajektorie bekannt sein. Dies ist nur in wenigen Trainingsfällen der Fall.



**Bild 6** Drei Trajektorien für die Klebebahn am Gehäuse einer Ölpumpe: herkömmlich aufgenommen, per Hand geführt und angepasst (übertrieben zur Visualisierung der Metrik).

- In dieser grafisch begründeten Metrik werden nur die drei translatorischen Freiheitsgrade berücksichtigt, Orientierungen gehen nicht ein.
- Der Nutzer weiß nicht notwendigerweise, welche Trajektorie die Beste für die Aufgabe ist.
- unabhängig von einem stationären Arbeitsplatz zu agieren und
- Informationen über den Systemzustand bzw. zeitnahe Warnungen zu erhalten, ohne den Blick auf den Monitor richten zu müssen.

Bild 6 zeigt einen Messversuch mit den relevanten Trajektorien und der vorgeschlagenen Metrik. Künstliche Abweichungen wurden in die Bahn eingefügt, um die Metrik zu visualisieren.

Eine bessere Einschätzung der Güte der Bahn ergibt sich durch die Verwendung von lokalen Sensoren, über die die Werkstückgeometrie aufgenommen werden kann. Mittels eines Laser-Triangulationssensors beispielsweise kann die programmierte Bahn weiter an das Werkstück angepasst werden. Diese Alternativen werden in aktuellen Arbeiten untersucht.

### 3.4 Sprachdialogsystem

Die Erweiterung der Eingabeschnittstelle um ein Sprachmodul bietet dem Nutzer eine intuitive Interaktionsmöglichkeit mit dem System. Mithilfe eines Sprachdialogsystems ist es dem Benutzer möglich,

- den Roboter zu steuern, auch wenn zeitgleich ausgeführte Arbeiten den Einsatz beider Hände erfordern,

Grundsätzlich bieten sich zwei unterschiedliche Möglichkeiten zur Realisierung eines Spracherkennungssystems. Die erste besteht darin, vollständige Sätze zu analysieren und die darin enthaltenen Informationen zu extrahieren. In einem zweiten Ansatz ist die Erkennung selektiv auf eine Menge vordefinierter Befehle beschränkt, wobei jeder Befehl direkt mit eindeutigen Systembefehlen verknüpft ist. Aus Gründen der Zuverlässigkeit eignet sich dieser zweite Ansatz speziell für Anwendungen im industriellen Bereich besser als die kontinuierliche Spracherkennung.

In der InTeach Programmierumgebung wird ein DECT Headset zur Ein- und Ausgabe der Sprachdaten eingesetzt. Die realisierte Softwarelösung zur Verarbeitung der Sprachkommandos nutzt die von Microsoft bereitgestellte Schnittstelle SAPI5.1 und erlaubt die Ausführung einfacher Steuerungsbefehle, wie das Starten und Stoppen der Aufnahme und Wiedergabe sowie das Einstellen der Geschwindigkeit und der Bewegungsrichtung. Die Grammatik der hierzu

notwendigen Befehle lässt sich mithilfe der Extensible Markup Language (XML) definieren [10].

Für den Einsatz einer Sprachsteuerung im industriellen Umfeld ist eine hohe Robustheit gegenüber Umgebungsgeräuschen notwendig. Voraussetzungen hierzu sind eine gute Tonqualität des verwendeten Mikrophons sowie eine hinreichende Genauigkeit des Spracherkenners. Doch auch die Struktur der Grammatik trägt einen entscheidenden Teil zur Zuverlässigkeit des Systems bei. Um das System bedienerfreundlich zu gestalten, sollte auf lange und umständliche Befehle verzichtet werden. Andererseits führen speziell kurze und einsilbige Kommandos bei starken Störgeräuschen leicht zu unbeabsichtigten Erkennungsergebnissen.

Der ungewollten Erkennung von Befehlen wird in der Sprachschnittstelle der InTeach Programmierumgebung durch verschiedene Mechanismen entgegengewirkt. Zum einen werden zwei Systemzustände unterschieden, welche die spezifische Aktivierung des Erkenners erlauben. Im „Standby“-Zustand reagiert das System nur auf einen einzigen Befehl, welcher aufgrund seiner Länge eine unbeabsichtigte Erkennung unwahrscheinlich macht. Wird das Kommando „start communication“ erkannt, geht das Programm in den aktiven Zustand über, in dem die Interpretation einer Reihe zwei- bis dreiteiliger Befehle möglich ist, siehe Bild 7.

Wird nur der erste Teil eines Befehls erkannt, so bietet die Sprachsoftware dem Nutzer Hilfestellung, indem mögliche Ergänzungen aufgezeigt werden. Auf diese Weise fordert das System – etwa von unerfahrenen Benutzern – den vollständigen Befehl schrittweise ein. Kritische Kommandos, welche beispielsweise eine Bewegung des Roboters auslösen, werden durch Nachfrage des Systems zusätzlich abgesichert (in Bild 7 durch <confirm> gekennzeichnet).

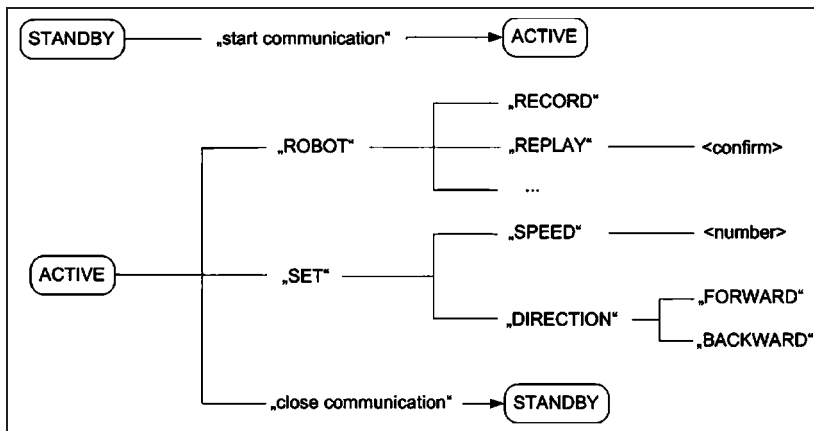


Bild 7 Befehlsstruktur.

### 3.5 Kommandieren und Visualisieren über einen PDA

Die Nutzung eines PDA (Portable Digital Assistant) macht als Ergänzung des Sprachdialogsystems Sinn, wenn die Nutzung eines stationären Arbeitsplatzes nicht immer möglich ist. Im industriellen Umfeld ist darauf zu achten, dass entsprechende Richtlinien und Anforderungen eingehalten werden, angefangen von IP-Schutzklassen bis hin zu ATEX-Kategorien. Durch die dem Massenmarkt entnommene Hard- und Software sind einerseits die Kosten für die Geräte und die Software-Entwicklung überschaubar, andererseits finden die Nutzer ein bekanntes Look-and-Feel vor.

Für die InTeach Programmierumgebung wird ein i.roc 410 der Fa. Exloc Instruments verwendet, siehe Bild 8. Das Gerät ist unter anderem sturzsicher und entspricht der Schutzklasse IP65. Es verfügt über Bluetooth- und WLAN-Konnektivität und verwendet das Windows Mobile 2003 Betriebssystem.

Mit dem Gerät wird eine Visualisierung des Roboters in 3-D ermöglicht, die Grafik wird mittels der OpenGL ES Grafikbibliothek erzeugt. Der Nutzer ist so in der Lage, sich die Roboterbewegungen in einer Simulation anzuschauen und auf Fehler zu überprüfen. Weiterhin ermöglicht der PDA die Kommandierung des Roboters, also Start und Stop der Aufnahme und der Wiedergabe. Ebenso ist aber auch die De-

finition von komplexen Parametern, etwa eines Geschwindigkeitsprofiles über eine grafische Schnittstelle möglich.

### 3.6 Steuerungsarchitektur

Während die aktuellen Implementierungen der InTeach Programmierumgebung monolithisch reali-

siert sind, bietet die Integration in ein Framework Vorteile. Am Fraunhofer IPA wird dazu das auf der Skriptsprache Python [11] basierende Steuerungsframework Go entwickelt.

Gerade wenn Komponenten sich in der Entwicklung befinden und neue Module laufend hinzu kommen, ist eine nahtlose und leichte Integration in ein modulares Framework von Vorteil. Mögliche Module sind in Bild 9 und in diesem Abschnitt dargestellt.

Für unterschiedlichste Hard- und Softwarekomponenten und deren Interaktion und Kommunikation innerhalb eines Gesamtsystems wurden bereits viele Middleware- und Integrationsframeworks vorgestellt. Zu nennen wären hier z. B. ORCA [12], GenoM [13] oder MCA2 [14]. Meistens jedoch sind sie nur für spezielle Einsatzgebiete geeignet oder erfordern einen ho-



Bild 8 PDA mit 3-D Visualisierung.

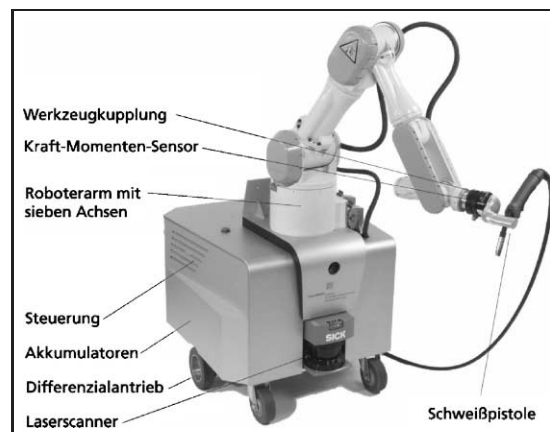


Bild 9 Komponenten des rob@work.

hen Aufwand bei der Einrichtung des Frameworks. Durch die langjährige Entwicklung von Steuerungen am Fraunhofer IPA haben sich u. a. folgende Kriterien für ein gutes Framework herauskristallisiert:

- Modularität,
- Plattformunabhängigkeit,
- Kommunikation zwischen Prozessen,
- Ressourcen-Management,
- Robustheit und
- Echtzeitfähigkeit.

Das am Fraunhofer IPA entwickelte Framework Go auf Basis der Skriptsprache Python versucht, diese Kriterien möglichst abzudecken. Es bietet die synchrone und asynchrone Ausführung von zyklischen oder einmaligen Aktivitäten, die Möglichkeit zur Interprozesskommunikation über Sockets und diverse Entwicklungs- und Optimierungstools.

Die Echtzeitfähigkeit, die besonders bei kritischen Prozessen notwendig ist, lässt sich über das Go-Framework nicht sicherstellen, hier ist eine Entkopplung der entsprechenden Planungs- und Ausführungsebenen notwendig. Für die Implementierung von Funktionen nach weichen Echtzeit-Anforderungen (etwa > 50 ms) wurde ein Katalog von Echtzeit-Entwurfsmustern definiert.

Es gibt bereits eine Beispielumsetzung einer dreischichtigen Steuerungsarchitektur in Go. Hierbei werden die hardwarenahen, meist in C/C++ geschriebenen Komponenten der Robotersteuerung mittels des zentral positionierten Go kontrolliert. Durch die immer komplexer werdenden Roboter müssen dabei auch die Prozesse unterschiedlicher Rechner gesteuert werden. Dank Go spielt es aber bei der Programmierung keine Rolle, ob eine Funktion lokal oder über Netzwerk auf einem externen Rechner ausgeführt wird.

## 4 Anwendungsbeispiele

Im folgenden Abschnitt werden zwei Assitenzsysteme mit der InTeach

Programmierungsumgebung vorgestellt. Eine Industrieroboterzelle ermöglicht das Kleben von Werkstücken, der Assistenzroboter rob@work unterstützt den Nutzer bei Schweißarbeiten im Kleinserien- und Reparaturbereich.

Die Sicherheit des Nutzers wird im Industrieroboter-Beispiel über einen in die Robotersteuerung integrierten Safety Controller garantiert [15]. Er überwacht Geschwindigkeiten und Verfahrbereiche entsprechend der aktuellen Sicherheitsvorschriften [16].

### 4.1 Assistenzroboter rob@work

Der rob@work ist als Assistenzroboter konzipiert, um den Nutzer in industriellen Umgebungen bei seinen Aufgaben zu unterstützen. Er besteht aus einer mobilen Plattform mit Differenzialantrieb, Energieversorgung für neun Arbeitsstunden und Steuerrechner. An einem siebenachsigen Manipulator werden über eine Kupplung Werkzeuge wie Schweißgeräte oder Bohrmaschinen angebracht.

Der rob@work nimmt seine Umwelt durch vielfältige Sensoren wahr. Ein Laserscanner erlaubt die autonome Navigation. Über einen am Roboterarm angebrachten Kraft-Momenten-Sensor werden durch Werker oder Prozess aufgebrauchte Kräfte und Momente detektiert. Mit einem an der Kupplung montierten Laserlinienscanner werden Werkstücke vermessen, jeweils im relevanten Roboterkoordinatensystem.

Der Roboter kann durch Ziehen am Werkzeug geführt und sein Arm positioniert werden. Die notwendige Umrechnung der Sensordaten in Bewegungen erfolgt in Analogie zu einem aus Federn, Dämpfern und Masselementen zusammengesetzten Modell. Die notwendigen Parameter werden personen- und aufgabenspezifisch ermittelt.

Einsatzszenarien des rob@work finden sich in den Bereichen der Kleinserien- und Einzelstückproduktion, aber auch im Instandhaltungs- und Reparaturbereich. Der

rob@work ist durch seine Assistenz in der Lage, die Qualität der Schweißnaht zu verbessern, indem er wichtige Prozessgrößen wie Geschwindigkeit oder Winkel konstant hält. Ebenso kann er aber, ausgerüstet mit der vorgestellten Programmierungsumgebung, sehr effizient Kleinserien schweißen.

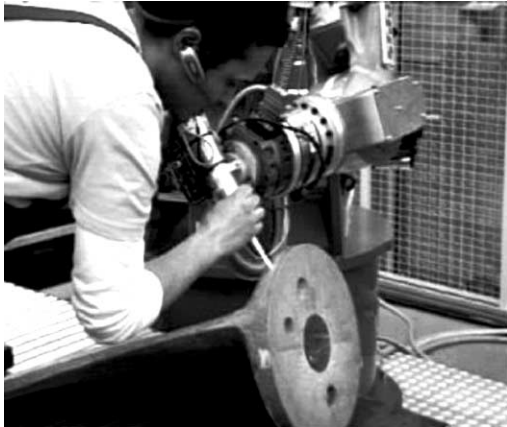
Mit dem rob@work wurden verschiedene einfache Schweißaufgaben gelöst. Die Verbindung von Blechen mit Wannennähten und das Füllen einer Kehlnaht mit mehreren Schweißlagen wurden erfolgreich vorgenommen. Besonders positiv fällt dabei die Leichtigkeit auf, mit der der Roboterarm verfahren werden kann. Ebenso gelingt die Positionierung der mobilen Plattform auf Antrieb. Negativ sind die Toleranzen zu bemerken, die sich durch die Räder und den nicht besonders steifen Arm ergeben. Für den späteren Einsatz müssen hier konstruktive Lösungen gefunden werden, etwa ausfahrbare Stützen.

### 4.2 Industrieroboterzelle

Eine weitere Testumgebung für die Programmierungsumgebung InTeach wurde mit einem Reis RV40 Industrieroboter in einem Klebeszenario aufgebaut. Eine Silikonkartusche wird, über den Roboter gesteuert, mittels Druckluft ausgepresst. Ziel ist es, der Freiformkontur einer Schiffsschraube zu folgen. In dem Szenario werden PDA, Sprachdialogsystem und 3-D Interface zur Nachbearbeitung der Bahn genutzt.

Die Programmierung der Bahn ist in Bild 10 dargestellt, der Werker führt den Roboter am Klebewerkzeug, das System detektiert diese Kräfte und setzt sie in Bewegungen um.

Nach der Aufnahme der Bahn wird im Nachbearbeitungsschritt ein Geschwindigkeitsprofil definiert, ebenso die Bahnpunkte, an denen das Werkzeug aktiv sein soll. Eine Glättung und Segmentierung der Bahn findet durch die automatische Vorverarbeitung statt. Im Anschluss wird die Bahn mit aktivem



**Bild 10** Führen des Roboters über den KMS entlang der Kontur der Schiffschraube.



**Bild 11** Wiedergabe der aufgenommenen Bahn mit aktivem Prozess.

Werkzeug wieder abgefahren, siehe Bild 11. Die Programmierung der Bearbeitungsaufgabe an einem Freiform-Werkstück nimmt nur wenige Minuten in Anspruch.

## 5 Zusammenfassung und Ausblick

In diesem Beitrag wurde eine Möglichkeit zur intuitiven Programmierung von Assistenzsystemen im Produktionsprozess, etwa dem Schweißen oder Kleben, dargestellt. Basierend auf der Programmierung durch Vormachen kann der Roboter an der Naht entlang geführt werden, die Bahn kann aufgenommen, nachbearbeitet und beliebig wieder abgespielt werden. Die Interaktion wird taktil oder über grafische und verbale Schnittstellen realisiert. Die Integration dieser Programmierumgebung kann über eine skript-basierte Steuerungsarchitektur erfolgen.

Der Nutzen für den Anwender besteht in einer niedrigeren Schwelle der Produktionsstückzahlen für eine wirtschaftliche Au-

tomatisierung. Dadurch, dass sich das Verhältnis von Programmierzeit zu Prozesszeit verbessert, können auch niedrigere Losgrößen automatisiert werden. Die Produktivität und damit die Wettbewerbsfähigkeit der Unternehmen steigt.

Ziel der weiteren Arbeiten am Fraunhofer IPA ist die Erhöhung der Flexibilität der Programmierumgebung InTeach; durch die Integration von Sensoren wird die Bahnprogrammierung tolerant gegenüber kleinen Abweichungen. Weiterentwickelt werden ebenso die Schnittstellen zum Nutzer. Besonders die Spracheingabe bietet viele Möglichkeiten zur Weiterentwicklung, angefangen bei einer Adaption an den Nutzer über die Fehlerbehandlung bis hin zu natürlichsprachlichen Dialogen.

## Danksagung

Diese Arbeit wurde von der EU im 6. Rahmenprogramm (FP6) durch die Projekte 011838 SMERobot<sup>TM</sup> und 002020 COGNIRON gefördert.

## Literatur

- [1] World Robotics 2005. UNCE, IFR. United Nations, Genf, 2005.
- [2] Schraft, R. D.; Helms, E.; Meyer, C.: Assistenzfunktionen des ASSISTOR-Demonstrators rob@work. In: Tagungsband der Robotik 2004, München 2004.
- [3] Hans, M.: Eine modulare Kontrollarchitektur für den Hol- und Bringdienst von Roboterassistenten. IPA-IAO-Bericht 412, Heimsheim: Jost-Jetter Verlag, 2005, 122 S. Zugl.: Stuttgart, Univ., Diss.
- [4] [http://www.osha.gov/dts/osta/otm/otm\\_iv/otm\\_iv\\_4.html](http://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_4.html). Homepage der Occupational Health and Safety Administration, U.S. Department of Labor. April 2006.
- [5] Leeser, K., Donoghue, J., Townsend, W.: Computer Assisted Teachand Play. In: Proc. of Fifth World Conf. on Robotics Research. Cambridge, MA, 1994.
- [6] Heiligensetzer, P.: Safe Operation – Safe Handling. 4. OTSWorkshop. FpF – Verein zur Förderung produktionstechnischer Forschung, Stuttgart, 2005.
- [7] <http://www.robolab.de>. Homepage der mz robolab GmbH, March 2006.
- [8] Albu-Schaefer, A.; Hirzinger, G.: Cartesian Impedance Control Techniques for Torque Controlled Light-Weight Robots. In: Proc. of IEEE Int'l Conference on Robotics and Automation (ICRA) 2002, Washington, USA.
- [9] Schraft, R., Dr; Meyer, C.: The Need for an Intuitive Teaching Method for Small and Medium Enterprises. In: Tagungsband ISR 2006 – ROBOTIK 2006. Mai 2006, München.
- [10] Pires, J. N.: Robot-by-voice: Experiments on Commanding an Industrial Robot using the Human Voice. Industrial Robot, an International Journal, Vol. 32, No. 6, Emerald Publishing, 2005.
- [11] <http://www.python.org/>, Juni 2006.
- [12] <http://orca-robotics.sourceforge.net/>, Juni 2006.
- [13] <http://softs.laas.fr/openrobots/tools/genom.php>, Juni 2006.
- [14] <http://mca2.sourceforge.net/>, Juni 2006.
- [15] Schraft, R. D.; Meyer, C.; Parlitz, C.: PowerMate – A Safe and Intuitive Robot Assistant for Handling and



Assembly Tasks. In: Proc. IEEE Int'l Conference on Robotics and Automation (ICRA) 2005, Florida, USA.

- [16] ISO-Norm 10218-1, Roboter für Industrieumgebungen – Sicherheit – Teil 1: Roboter. Beuth-Verlag, Juni 2006.



1



2



3



4

**1 Dipl.-Systemwiss. Christian Meyer** studierte Angewandte Systemwissenschaft an der Universität Osnabrück und ist wissenschaftlicher Mitarbeiter in der Abteilung Robotersysteme des Fraunhofer IPA. Arbeitsgebiete sind Roboterprogrammierung, Assistenzsysteme und Sicherheitskonzepte.

Adresse: Fraunhofer-Institut für Produktionstechnik und Automatisierung IPA, Nobelstrasse 12, 70569 Stuttgart, E-Mail: christian.meyer@ipa.fraunhofer.de

**2 Dipl.-Inf. (Bio) Rebecca Hollmann** studierte Bioinformatik in Tübingen und arbeitet seitdem als wissenschaftliche Mitarbeiterin in der Abteilung Robotersysteme des Fraunhofer IPA. Ihre Arbeitsschwerpunkte sind die Mensch-Maschine-Schnittstelle und Steuerungsarchitekturen.

Adresse: s. o.

**3 MSc Dipl.-Math. (FH) Christopher Parlitz** studierte Mathematik und Software Technology in Stuttgart. Seitdem arbeitet er mit dem Schwerpunkt Servicerobotik am Fraunhofer IPA. Er ist dort in mehreren internationalen und nationalen Projekten tätig.

Adresse: s. o.

**4 Dipl.-Ing. Martin Hägele, M.S.** studierte Maschinenbau an der Universität Stuttgart und Engineering Science an der George Washington University (Washington DC, USA). Seitdem arbeitet Martin Hägele am Fraunhofer IPA. Er leitet seit 1993 die Abteilung Robotersysteme. Er veröffentlichte bisher über 30 Publikationen aus den Bereichen Industrielle Automation, Service Robotik und Mechatronik-Entwicklung. Martin Hägele ist Koordinator des EU-IP SMERobot.

Adresse: s. o.