Programmiersprachen & Programmierung von Robotersystemen

Eine Ausarbeitung von Hannes Ahrens
zum Vortrag von Hannes Ahrens ;-)

Proseminar: Anwendungen und Methoden der modernen Robotik
Proseminarleiter: Prof. Dr. Jianwei ZHANG

Behandelt wird: Seite

Einleitung

Da ich mich generell sehr für die Programmierung interessiere (Welch' Wunder als Informatikstudent), mich Roboter schon immer fasziniert haben, jemand anderes bereits den Vortrag über die Geschichte der Robotik halten wollte - welcher mich ebenso interessiert hätte - und ich zudem sehr erfreut über den als früh angesetzten Termin war, habe ich mich für den Vortrag über Programmiersprachen für Roboter entschieden.
Um meinem Interesse für die Geschichte dennoch gerecht zu werden, habe ich mich entschieden außerdem auf die Entwicklung der Roboterprogrammierung einzugehen.

Während der Erarbeitung zum Vortrag ist mir dann aufgefallen, dass unter der Roboterprogrammierung noch ganz andere Dinge zu verstehen sind als nur die Implementierung von Quellcode, weshalb ich mich entschied zuerst die verschiedenen Programmierverfahren zu erläutern und mein Thema mit „Programmierung & Programmiersprachen von Robotersystemen“ zu benennen.

Auf Anraten des Proseminarleiters Dr. Jianwei Zhang entschied ich in meinem Vortrag zudem auf die in u.a. unserer Uni angewendete Roboterprogrammiersprache, bzw. den Simulator, RCCL einzugehen.

Entsprechend der recht geringen Resonanz, die mir in den Teilaspekten der Geschichte und Programmierverfahren noch am höchsten erschien, und den kaum Vorhandenen Nachfragen und Anregungen, will ich in dieser Ausarbeitung auch vor allem auf die im Vortrag selbst behandelten Aspekte ohne besondere Vertiefung von RCCL eingehen.

Die Programmierverfahren

– Was man unter Roboterprogrammierung versteht

In der Roboterprogrammierung gibt es viele verschiedene Programmierverfahren, die sich jedoch alle in die beiden Unterkategorien der Online- und Offlineprogrammierung unterteilen lassen.
Dabei versteht man unter Onlineprogrammierung die Programmierung eines Roboters unter Nutzung des selbigen – daraus zu verstehen ist, dass der Roboter während der Programmierung aktiv ist und nicht abgeschaltet, bzw. neu gestartet wird – und unter Offlineprogrammierung entsprechend die Programmierung ohne Verwendung des Roboters selbst.

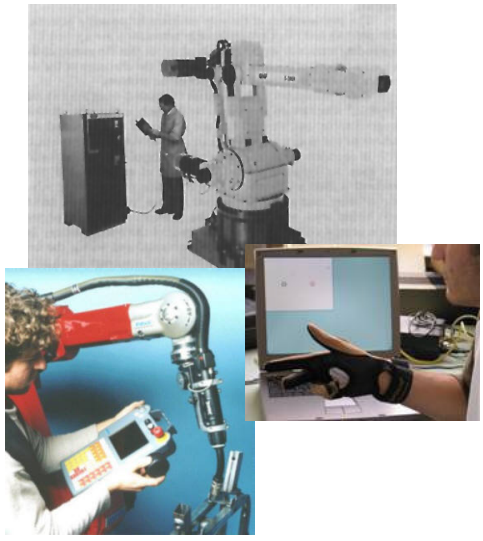
Zur Onlineprogrammierung zählt man insbesondere das „Teach-In“ -Verfahren, zu dem wiederum das „Play-Back“ -Verfahren zugehörig ist. Außerdem wird auch die Parametereingabe per Softwareschnittstelle über zumeist verwendete, so genannte „Control Panels“ dazugezählt.
Zur Offlineprogrammierung hingegen zählen die Programmierung per 3D-Simulation, die Erstellung und Kompilierung von Quellcode, sowie das Editieren von Quelldateien die von der Robotersteuerungssoftware geladen werden müssen.

All diese Verfahren werden im Folgenden Vorgeführt, veranschaulicht und erläutert:

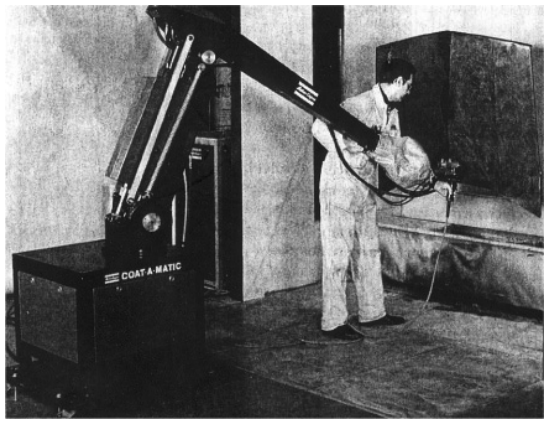
Onlineprogrammierung

Die Onlineprogrammierung ist sehr nützlich um einem Roboter schnell und einfach, mittels von der Steuerungssoftware gegebener Operationen und Schnittstellen, ein neues Verhalten einzuprogrammieren.
Dabei sehr hilfreich ist es, den Roboter bei seinem Verhalten direkt beobachten zu können. Damit bietet diese Variante zudem den Vorzug ohne lange und aufwändige Schulungen auszukommen („Learning by Doing“), was für Unternehmen von großer Bedeutung sein kann.

- **„Teach-In“ -Verfahren:**
  Unter dem „Teach-In“ -Verfahren versteht man alle Verfahren, bei denen einem Roboter sein späteres Verhalten „angelernt“ wird, bis dass das gewünschte Ergebnis erzielt ist. Dazu zählt wie bereits erwähnt auch das „Play-Back“ -Verfahren.
  Ebenso zählen jedoch auch dazu einem Roboter über z.B. ein Touchpanel oder durch Gesten oder Sprache seine Aufgaben zuzuteilen.



- **„Play-Back“ -Verfahren:**
  Beim diesem Verfahren wird der Roboter programmiert, indem er entweder direkt, oder indirekt bewegt wird. Später führt er diese Bewegung dann immer wieder wie einmal vorgeführt aus, dabei wird jedoch über die Steuerungssoftware häufig noch durch Menschen bei der Vorführung verursachtes Zittern herausinterpoliert.



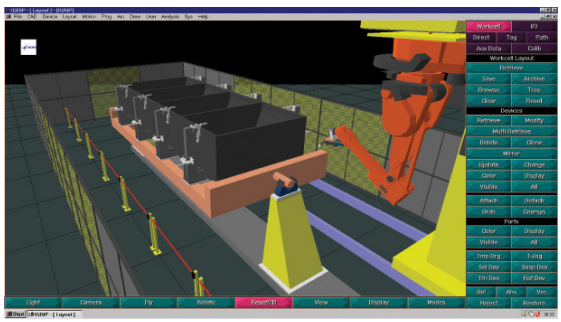
  Bei Industrierobotern war es lange Zeit Gang und Gebe, dass es für jeden Typspezifischen Roboter ein leichteres, besser zu bewegendes 1:1 Vorführmodell gab.

- **Parametereingabe per Softwareschnittstelle:**
  Die direkte Parametereingabe ist auch Bestandteil der Onlineprogrammierung. Dabei werden einfach alle für die Gelenkbewegung benötigten Parameter spezifiziert und der Roboter führt diese Bewegung dann exakt aus.
  Dies kann sowohl über einen PC, als auch über speziell angepasste Control Panels geschehen.

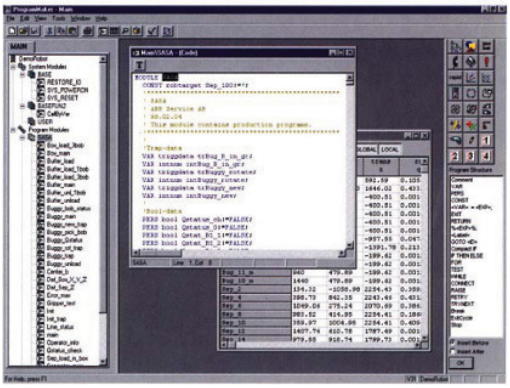
Offlineprogrammierung

Da bei der Onlineprogrammierung oft kostbare Zeit in der Produktion verloren geht, insbesondere wenn der gesamte Produktionsprozess für die Umprogrammierung eines einzelnen Roboters (um ihn bspw. an ein neues, noch nicht gefertigtes Produkt anzupassen) angehalten werden muss, wird auch die Offlineprogrammierung häufig verwendet.
Außerdem wird diese benötigt um tiefere Eingriffe in ein Robotersystem und dessen Verhalten vornehmen und dann zur Sicherheit vor der Anwendung simulieren zu können.
Wenn dabei die Steuerungssoftware abzuändern oder gar neu zu implementieren ist, dann ist die Offlineprogrammierung unumgänglich.

- **3D-Simulation:**
  Wichtiger Bestandteil der Offlineprogrammierung ist die Simulation um spätere Probleme wie Kollisionen frühzeitig zu erkennen und zu vermeiden. Je nach Simulator kann dabei auch auf das Verhalten des simulierten Roboters eingewirkt werden. Später braucht dann nur noch die Ausführung auf den echten Roboter übertragen werden.
  Ggf. wird dann dabei die komplette Steuerungssoftware ersetzt.

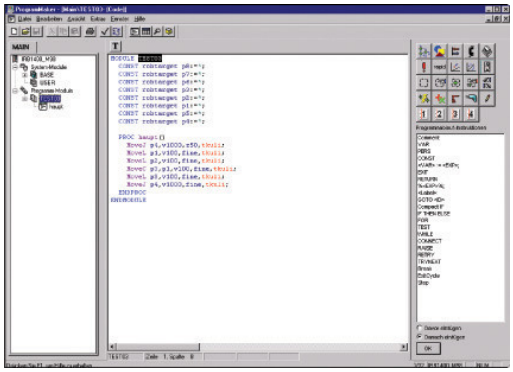


- **Softwarekodierung und -kompilierung:**



„Wenn nichts mehr hilft, dann muss die alte Softwarekodierung und -kompilierung ran“ mag sich manch einer sagen.
Und dies ist für wahr auch sicher immer wieder mal der Fall (Wenn z.B. ein Bug in der Steuerungssoftware entdeckt wurde).
Hierfür wird extra angelerntes Personal benötigt, oder es wird auf den Service des Roboterherstellers oder einer Fremdfirma zurückgegriffen.
Die Kodierung und Kompilierung mag hierbei jedoch auch nützlich für manche Optimierungen, oder das Einbringen neuer Möglichkeiten für z.B. die Onlineprogrammierung sein, wofür nicht Zwangsweise ein Problem vorausgegangen sein muss.
Zudem muss eine Steuerungssoftware zumindest einmalig geschrieben und kompiliert worden sein – nur auf Basis dieser sind andere Programmierverfahren erst möglich!

- **Editieren von Quelldateien:**



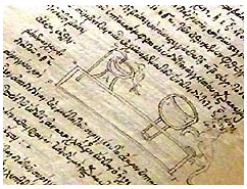
Zu guter letzt hier Aufgeführt gehört auch das Abändern von Quelldateien, die von der Steuerungssoftware geladen werden zur Offlineprogrammierung.
Nützlich sind solche Quelldateien um verschiedene Verhaltensmodelle zu erstellen und diese dann von der Steuerungssoftware je nach Bedarf laden zu können. Die Speicherung und das Laden solcher Quelldateien mag – je nach Softwareimplementierung – auch über Onlineverfahren möglich sein, hier jedoch ist das Editieren einer solchen Datei gemeint.
Dabei werden feste Parameter in eine Datei geschrieben, die von der Steuerungssoftware später eingelesen und verwendet werden.

Die Entwicklung

Ab wann von Roboterprogrammierung die Rede sein kann:

Schon vor langer Zeit entwickelten Ägypter und Griechen hoch komplizierte Maschinen, die teilweise wie Roboter fungierten. Diese hatten dann zumeist eine feste Programmierung, die nicht abgeändert werden konnte, ohne die Maschine selbst abzuändern („fixed automation“).
Dabei braucht man nur einmal an die vielen Fallen in Grabstätten, die in allen wohl nicht mit allzu vielen Effekten aufwarten können.

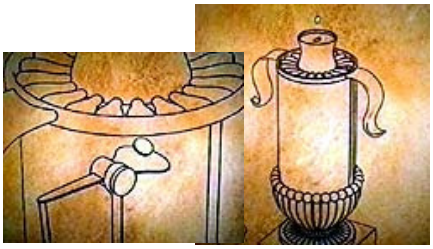
Genannt sei hier einmal der Tüftler Heron, der vor etwa 2000 Jahren, um 60 nach Christus in Alexandria (in Ägypten) lebte. Dieser lehrte als Mathematiker und Physiker und auf ihn gehen nach heutigen Erkenntnissen eine Menge Erfindungen zurück.



Dabei ist die Zuordnung manches Mal nicht eindeutig, da oft nur abgeschriebene Schriften aus dem Mittelalter die Zeit überdauert haben, oder nicht klar ist ob Heron selbst auch nur wieder von anderen die Ideen erhalten hat.

Heron soll, so ist es auf den Seiten

http://www.zdf.de/ZDFde/inhalt/7/0,1872,2051719,00.html und http://www.zdf.de/ZDFde/inhalt/0/0,1872,2051776,00.html nachzulesen, unter anderem sich automatisch schließende und öffnende Türen, spezielle Trinkbehälter wie Sprühflaschen, den ersten Automaten



(der Weihwasser auf Münzeinwurf spendete), die erste heute bekannte Dampfturbine, verschiedene schwere Katapulte wie auch eine Maschinenarmbrust erfunden haben.

Nicht zuletzt soll er jedoch auch die ersten per Vorläufer eines Trommelspeichers programmierbaren Roboter entwickelt haben.

Die meisten seiner Erfindungen wurden allerdings kaum eingesetzt und nur von ihm selbst für Vorführungen genutzt, so auch seine Roboterholzfiguren, die er für Theaterstücke nutze.

Diese Figuren, wie auch Bühnengegenstände, bewegten sich, durch Schnüre gelenkt, auf einer kleinen gebastelten Bühne umher und er erzählte dazu vermutlich Geschichten.

Dem ganzen fügte er dann noch Effekte hinzu indem er z.B. kleine Kugeln auf Bleche fallen ließ um es Donnern zu lassen, oder Figuren mit Spiegeln erscheinen und verschwinden ließ.
Die über zahlreiche Rollen geführten Schnüre, durch welche die Figuren gesteuert wurden, wickelte er wiederum um eine mit Noppen versehene Stange. Je nach Wickelung bewegten sich die Figuren und Gegenstände nun anders beim Drehen der Stange.

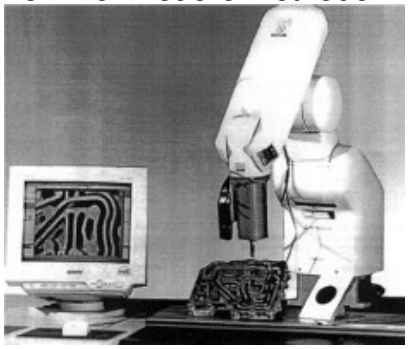
Was heute wäre, wenn all diese Erfindungen sich bereits zu jener Zeit, vor 2000 Jahren, durchgesetzt hätten, ist nicht zu sagen, aber dies sollte zeigen, dass solch' Erfindungen und die Ideen dazu nicht allein in den letzten 100 Jahren und im Zuge der Entwicklung von Computern entstanden sind. Wirklich nutzbringend im industriellen Maße eingesetzt wurden diese natürlich erst in jüngerer Zeit.

Wo wir heute mit der Programmierung sind:

und was sie geschaffen hat

Heutzutage gibt es eine Vielzahl verschiedener speziell angepasster Roboterprogrammiersprachen zur Quellcodekodierung. Dabei sind viele dieser Sprachen sogar speziell auf bestimmte Robotertypen/-familien ausgerichtet und von den Herstellern der Roboter extra für eben genau diese entwickelt worden.
Es gibt aber auch einige freier Programmiersprachen oder Erweiterungen für gängige Programmiersprachen wie C++ und JAVA.
Die diversen Sprachen bieten alle speziell für Roboter benötigte Funktionen und Features wie z.B. dass Roboter in Echtzeit angesprochen werden können, oder dass verschiedene Steuerungssysteme unterstützt werden. Auch Methoden, die Raumbewegungen in z.B. pneumatische Gelenkbewegungen umwandeln werden geboten um einem Programmierer die Arbeit zu erleichtern.

Dennoch ist die Programmierung auf diese Weise oft sehr aufwändig und teuer, weshalb außerdem eine Vielzahl anderer Verfahren entwickelt wurde.
Davon wurden bereits die Hauptkategorien der Online- und Offlineprogrammierung vorgestellt. Im Sinne der Aufwands- und Kostenreduzierung nehmen die Teach-In - Verfahren heute sicher einen bedeutenden Bestandteil ein. Auch die Programmierung per Control Panel und mit graphischen Benutzeroberflächen, wie auch Simulatoren ist nicht zu vernachlässigen, aber der Bereich der Roboter-programmierung befindet sich in stetigem Wandel. So kommen neuere Methoden wie die Sensorgestützte Programmierung und Virtual-Reality heute auch immer mehr zum Tragen.



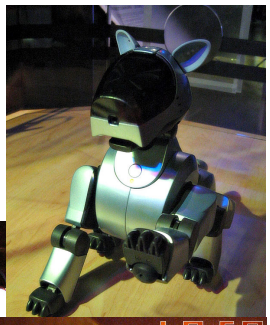
Bei der Sensorgestützten Programmierung ist eine Software z.B. in der Lage, wie im Bild gezeigt, scharfe, raue Kanten eines Bauteils zu identifizieren und die Programmierung des Roboters so abzuändern, dass dieser die entsprechenden Stellen entgratet.

Und im Bereich „Virtual-Reality“ darf sich der Programmierer wie der Roboter selbst fühlen. Er steuert ihn, z.B. mit einem elektronischen Handschuh direkt, als sei er selbst der Roboter.
Zudem bekommt er visuelle dreidimensionale Daten aus Sicht des Roboters übertragen – wie in einem Computerspiel.
Eine 3D-Brille, ggf. etwas passende Musik und eine Tüte Popkorn dürfen dabei natürlich nicht fehlen! ;-)

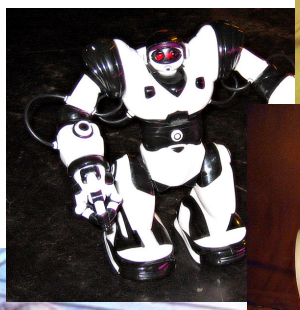
Wohin sich die Roboterprogrammierung fortentwickelt:

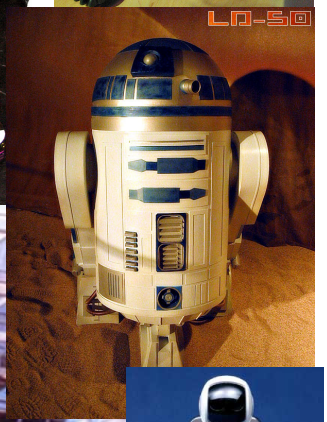
und möglicherweise „schon bald“ entwickeln wird

Momentan wird in viele verschiedene Richtungen entwickelt und geforscht.
So versucht man den Robotern die natürliche Sprache des Menschen, das Reden, Deuten, Zeigen, verständlich zu machen, oder ihnen beizubringen eigene Entscheidungen zu fällen und Reflexe zu besitzen.
Sie sollen eigenständig Dinge ausprobieren, lernen, und gelerntes anwenden. Dabei sollen sie dann natürlich noch Sinn und Zweck der Ausführungen bewerten können.
Auch ist die Kommunikation und die Kooperation miteinander ein wichtiges Forschungsfeld.
Alles in allem sind diese Gebiete alle sehr komplex und es ist nicht klar wann und wo sie später nutzbringend Anwendung finden.
Bisher werden die auf diesen Gebieten geführten Entwicklungen vor allem in Entertainment und für weitere Forschungen eingesetzt.



Wohin uns die Roboterprogrammierung und die KI in Zukunft bringen werden ist nur noch spekulativer.
VII. Werden die Ideen aus Science-Fiction-Filmen wie A.I. in der einen oder anderen Weise eines Tages zutreffen, wie es sich in der Geschichte zu Heute auch in manchen Fällen zugetragen hat, aber ob Roboter wirklich eines Tages so intelligent, oder gar noch intelligenter als Menschen werden können, ob es überhaupt möglich ist einen Roboter oder ein Programm intelligent werden zu lassen, sei einmal offen gelassen.

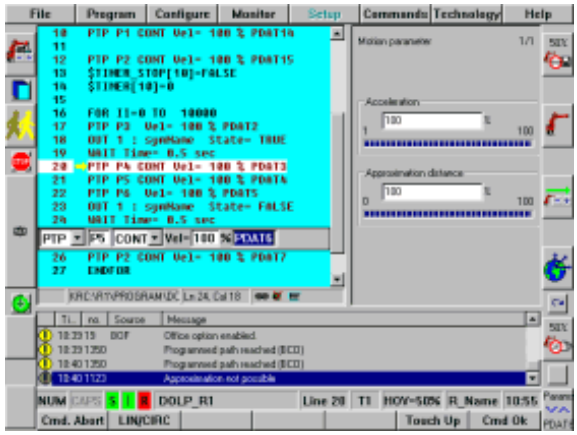


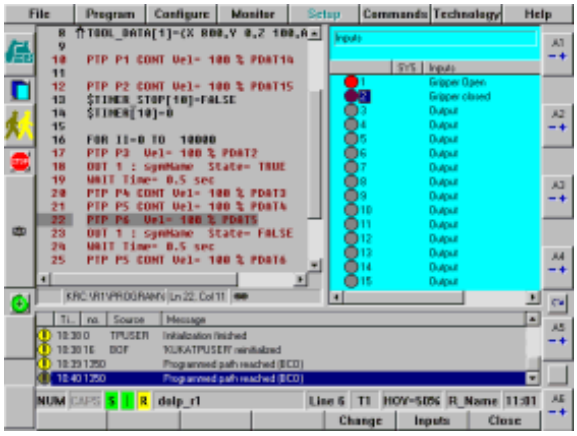


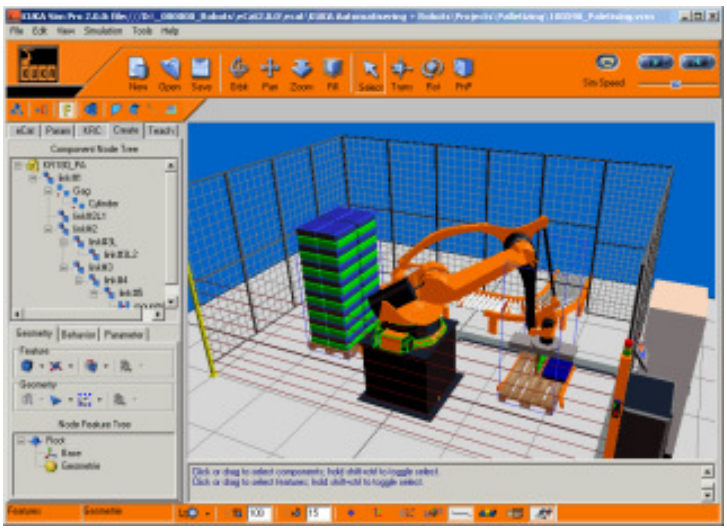
Programmiersprachen

Im Internet und in der Literatur ist eine Vielzahl verschiedener Roboterprogrammiersprachen zu finden.
Dazu zählen z.B. AL, AML, AR-BASIC, ARLA, BAPS, COSIMIR, IRL, JARS, KAREL, PA-LIB, RAPID, ROBEX, SRL und TPE, die hier keine weitere Erläuterung finden sollen.
NQC, VAL/VAL2/VAL3, KRL und RCCL sind ebenso Roboterprogrammiersprachen, wobei RCCL später noch genauer betrachtet wird.
NQC, stehend für „Not Quite C“ ist, wie es der Name schon vermuten lässt, an C / C++ angelehnt. Diese Sprache wurde von Dave Baum für Kleinroboter entwickelt und ist als Open-Source erhältlich. Sie bietet zwar weniger Funktionen als C und C++, ist dafür aber sehr viel kompakter und der für einen Roboter benötigte Betriebskern, sowie die darauf aufbauenden Programme beanspruchen die Hardwareressourcen relativ gering.
Diese Eigenschaften macht die Sprache beispielsweise für die Lego-Mindstorm Roboter sehr nützlich.
NQC wird in Verbindung mit Lego-Robotern häufig zum praxisbegleiteten Erlernen von Grundkenntnissen in der Roboterprogrammierung eingesetzt.
VAL, VAL2, VAL3 und KRL hingegen sind Sprachen, die für Industrieroboter entwickelt wurden. Die VAL-Sprachfamilie wurde von dem Industrieroboterhersteller Stäubli und KRL von, wohl Europas größtem Roboterhersteller, KUKA entwickelt.
Diese Sprachen sind entsprechend extra auf die produzierten Roboter der beiden Hersteller zugeschnitten.
Das macht sie zwar ungeeignet für Roboter fremder Hersteller, aber die Roboter aus eigener Fabrikation finden dafür entsprechend bessere Unterstützung in all ihren Funktionen.

Auf den Bilder zu sehen:
Die Programmierumgebung KRL
und der zugehörige Simulator von KUKA.



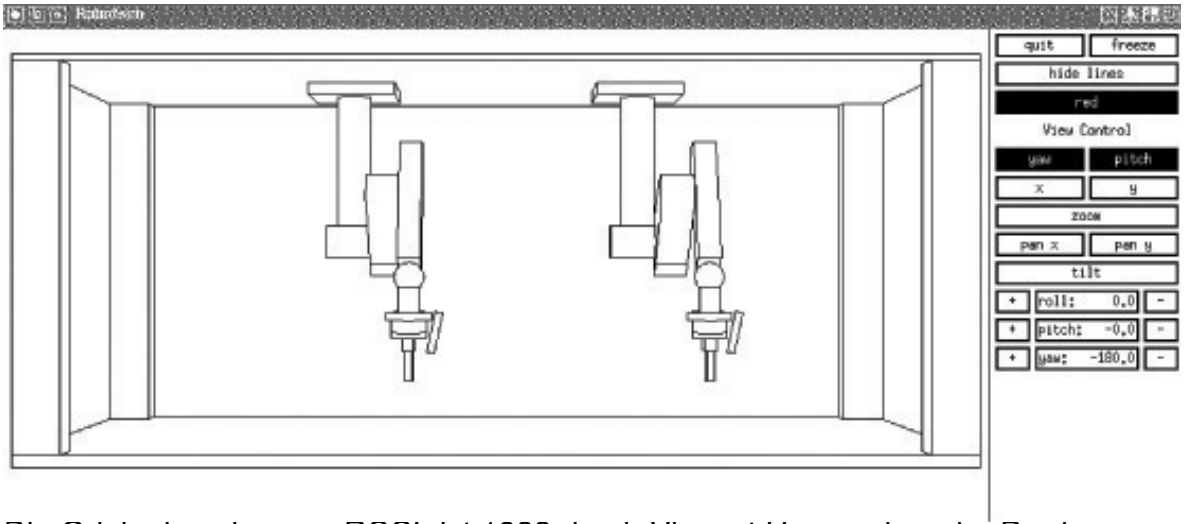




RCCL (Multi-RCCL)

Was RCCL ist und wo RCCL Anwendung findet:

RCCL, stehend für „Robot Control C-Library", ist eine Bibliothek für die Programmiersprache C/C++ und ist zur Erstellung von Aufgabenorientierter Software zur Robotersteuerung unter Unix gedacht.
RCCL beinhaltet außerdem noch einen Simulator zur 3D-Simulation der späteren Ausführungen.



Die Originalversion von RCCL ist 1983 durch Vincent Hayward an der Purdue Universität in den USA entstanden. Seitdem durchlebte RCCL eine starke Weiterentwicklung. So wurde zum Beispiel 1988 von Lloyd, Parker und McClain an der McGill Universität in Zusammenarbeit mit dem General Electric Advanced Technologie Laboratory in New Jersey die Multi-Roboter- und Multi-CPU-Unterstützung eingeführt. Infolge änderte sich die offizielle Bezeichnung von RCCL zu „Multi-RCCL".
RCCL gilt inzwischen zwar als veraltet und findet in der Industrie keine Anwendung mehr, aber sie ist für den nichtkommerziellen Einsatz frei erhältlich und wird aufgrund ihrer breiten Unterstützung und oftmals vorhandenen Weiterentwicklungen noch in Universitäten und teils auch Forschungseinrichtungen eingesetzt.
So kommt es, dass z.B. die Steuerungssoftware für den Arm mit integrierter Baretthand des mobilen Roboters am Informatikum der Universität Hamburg mit Hilfe von RCCL entwickelt wurde.

Wie RCCL aufgebaut ist und funktioniert:

Auf RCCL aufbauende Programme teilen sich in einen Hintergrund-Task und einen „Planungs"-Task.
Der Hintergrund-Task ist in der RCCL-C-Bibliothek verborgen und für den Programmierer nicht weiter sichtbar.
Der in ihm integrierte Bewegungsgenerator (trajektory generator) erstellt pro Roboter einen Bewegungs-Task (trajektory task) und versucht diese über alle vorhandenen CPUs aufzuteilen. Dabei werden auch mehrere Arme als einzelne Roboter gesehen.
Dem Programmierer werden zwar spezielle Funktionen und Datenstrukturen zur Verfügung gestellt um Koordinaten und Vektoren festzulegen und damit die Roboter zu steuern, aber auf die in der Bibliothek enthaltenen Servo-Level-Algorithmen hat er

keinen direkten Einfluss. Eine eigene Implementierung hierfür ist nicht gedacht und so kann es zu Kollisions- und Verrenkungsproblemen der Roboter führen, wenn die Algorithmen eine für die Bewegung ungünstige Bahn errechnen.
Allerdings gibt es dafür einige „Workarrounds“ um beispielsweise Bewegungen mit theoretisch unendlicher Beschleunigung zu entgehen, die unter anderem auftreten können, wenn ein Roboter für die Fortführung seiner Bewegung erst einmal seine Gelenke in eine andere Position oder Stellung bringen müsste.
Die eigentliche Programmierarbeit besteht darin den „Planungs“-Task zu implementieren. Dazu werden die von RCCL zur Verfügung gestellten Funktionen und Datenstrukturen genutzt um mittels Shared Memory dem Hintergrundtask mit seinen Bewegungs-Tasks mitzuteilen was die Roboter machen sollen.
Dabei ist dem Programmierer offen gelassen, ob er Informationen einliest, eine Benutzeroberfläche zur Verfügung stellt, oder alle Befehle selbst fest implementiert.

Ein Code-Beispiel:

Es folgt ein Codebeispiel für eine geradlinige Bewegung zu einem Zielpunkt und eine vom Bewegungsgenerator als „günstig“ errechnete, geschwungene Bewegung zurück zum Startpunkt. Dabei wird zuerst sichergestellt, dass sich der Roboter in einer bekannten Initialposition befindet, um Verrenkungen bei der geradlinigen Bewegung zu vermeiden. Zudem werden zwei kleine Pausen eingelegt.
Das Programm kann mithilfe der RCCL-Bibliothek in C für Unix kompiliert und im RCCL-Simulator ausgetestet werden. Die einzelnen Codezeilen sind zum besseren Verständnis kommentiert.

```
#include <rccl.h>                                   // RCCL-Library
#include “manex.560.h“                              // spezielle Deklarationen für den
                                                    // PUMA 560 - Roboter
main()                                              // Beginn der Hauptroutine
{
        TRSF_PTR p, t;                              // Transformationsmatrizen
        POS_PTR pos;                                // Zielposition
        MANIP *mnp;                                 // aktuelle Roboterposition/-eigenschaften
        JNTS rcclpark;                              // Initialposition und -parameter
        char *robotName                             // Roboterbezeichnung
        rcclSetOptions(RCCL_ERROR_EXIT);            // Option damit das Programm nur bei
                                                    // Laufzeitfehlern abgebrochen wird
        robotName = getDefaultRobot();
        if(!getRobotPosition(rcclpark.v, “rcclpark“, robotName))  // Falls für den Standardroboter keine
        {    printf(“position ‘rcclpark‘ not defined for robot\n“);    // Initialposition ermittelt werden kann,
             exit(-1);    }                                          // gib Fehlermeldung aus und beende!
        t = allocTransXyz(“T“, UNDEF, -300.0, 0.0, 75.0);           // relative Zielkoordinaten
        p = allocTransRot(“P“, UNDEF, P_X, P_Y, P_Z, xunit, 180.0); // Rotationsmatrix auf Starteinst.
        pos = makePosition(“pos“, T6, EQ, p, t, NULL);              // „kinematische Positionsgleichung“
        mnp = rcclCreate(robotName, 0);                             // initialisiere Trajektory Generator

        rcclStart();                    // startet Trajektory-Task
        movej(mnp, &rcclpark);          // Bewegung zur bekannten Initialposition
        setMod(mnp, ‘c‘);               // geradlinige kartesische Bewegung
        move(mnp, pos);                 // Bewegung zur Zielposition
        stop(mnp, 1000.0);              // mache eine Sekunde Pause
        movej(mnp, &rcclpark);          // Bewegung zurück zur Startposition
        stop(mnp, 1000.0);              // mache eine Sekunde Pause
        waitForCompleted(mnp);          // sicherstellen, daß alle Bewegungen abgeschlossen sind
        rcclRelease(YES);               // beende Trajektory Generator und schalte Roboter ab
}
```

Quellen

- **Internet:**
  - http://tams-www.informatik.uni-hamburg.de/lehre/ws2004/vorlesungen/einfuehrung_in_die_robotik/folien/48.pdf
  - www.iof.mw.tu-dresden.de/vorlesung/000301020401/05_mrb03_Programmierung_Flemming.pdf
  - www.ba-eisenach.de/fileadmin/_temp_/dokumente/stura/scripte/sem4/rob07.pdf
  - http://de.geocities.com/gehnio/Robotik.pdf
  - http://www.kuka.com/germany/de
  - http://www.staeubli.de

  im Weiteren für Bilder verwendet:
  - http://www.spiegel.de/wissenschaft/mensch/0,1518,164435,00.html
  - http://www.ld50.hu/article/ld50/reports/roboterkommen
  - http://www.smpp.northwestern.edu/robotLab/Cyberglovew.jpg
  - http://ai3.inf.uni-bayreuth.de/teaching/vl.grundlagenderrobotik/files/phgroboter.jpg
  - http://www.iwka.de/images/fotoarchiv/schweissanlagen.jpg

  (gefunden über die Google-Bildersuche)

  nicht weiter verwendet:
  - für Lego-Mindstorm-Interessierte:
    http://www.fh-landshut.de/~gschied/robotic/index.html
    http://bricxcc.sourceforge.net/nqc/
  - außerdem interessant: http://www.orocos.org/ **O**pen **Ro**bot **Co**ntrol **S**ystem

- **Literatur:**
  - Introduction to Robotics (John J. Craig)
  - Multi-RCCL User‘s Guide (John Lloyd und Vincent Hayward 1992)