Labor für CIM & Robotik
Prof. Dipl.-Ing. Georg Stark

Modellbasierte und komponentenorientierte Programmierung von Steuerungen

1. Entwicklungsprozess Industriesteuerung
2. Programmierparadigmen
   - objektorientiert
   - komponentenorientiert
   - modellbasiert
3. Beispiel – Entwicklung der Steuerung **MRobot**
   - Synchrone Ausführung und Simulation
   - Sensorintegration

Entwicklungsprozess - Industriesteuerung

Ziele

- Gute Wartbarkeit der Software
  - Fehlerbeseitigung
  - Erweiterungen
- Kosteneffizienz
- Optimaler Informationsfluss zwischen allen beteiligten Personen
- Hohe Funktionalität der Software

Ansatz: Verbesserte Softwaretechnologie

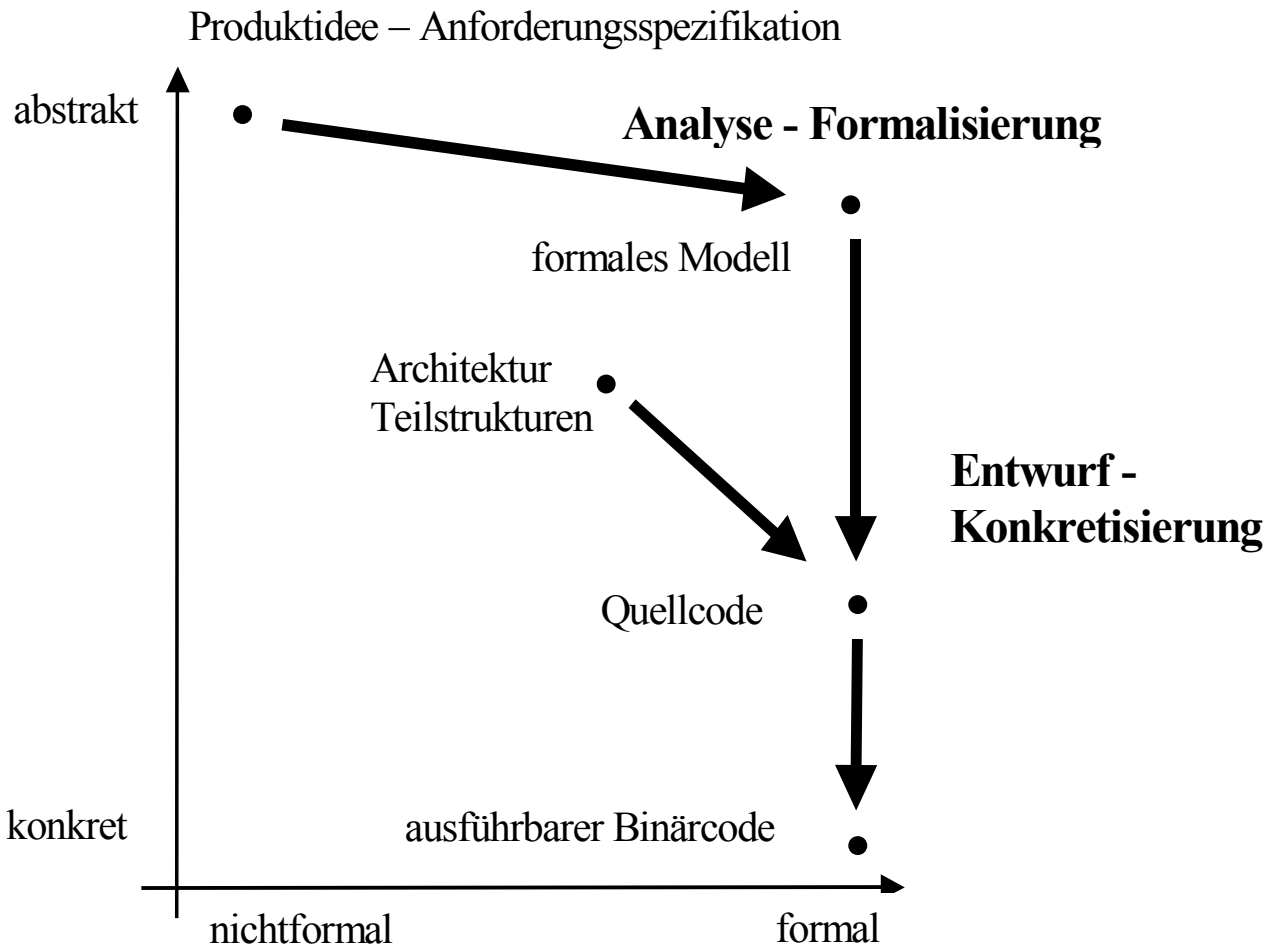
- Kombination aus modellbasierter, komponentenorientierter und objektorientierter Programmierung

Entwicklungsprozess - Phasen

1. Planung
   - Anforderungsspezifikation
2. **Analyse**
   - Beschaffung, Formalisierung des Wissens
3. **Entwurf**
   - Architektur
   - Teilstrukturen
   - Test
4. Implementierung und Test
5. Verifikation

Entwicklungsprozess – Analyse und Entwurf

Darstellung von Information in der Entwicklungsebene



Objektorientierte Programmierung

Leitgedanken

- Konzept der **vererbbaren Klassen**, **Kapselung**
- Klassen sind Urkopien für Objekte
- Bildung von Softwarevarianten
- **Softwareschnittstellen**, abstrakte Klassen
- Klassen stellen Wissenselemente dar
- Keine domänenspezifische Programmiersprache
- Unterstützung für die Darstellung der **Struktur des Wissens**, nicht des Wissens selbst

Komponentenorientierte Programmierung

Leitgedanken

- **Softwarekomponenten** sind ausführbare Softwareteile, die über standardisierte Schnittstellen benutzt werden
- Mit Hilfe von Komponenten kann eine **Framework-Plugin**-Architektur realisiert werden
- Das Framework stellt insbesondere die kritische **Echtzeit-funktionalität** bereit. Das spezifische **Robotikwissen** wird mit Hilfe von Plugins implementiert
- Vorteile der komponentenorientierten Programmierung:
  - Komponenten können mit **unterschiedlichen Sprachen** implementiert sein
  - Die **Wartung** der Software wird erleichtert
- Beispiele für **standardisierte Schnittstellen** für Komponenten:
  - **COM** von Microsoft,
  - CORBA von der OMG (www.omg.org),
  - JavaBeans.

Modellbasierte Programmierung

Leitgedanken

- Programme implementieren **Wissen** aus unterschiedlichen Bereichen, z.B. Bediendialoge, Eigenschaften von Objekten, Bewegungsverhalten von Maschinen
- **Formale Modelle** stellen das Wissen mit Hilfe von formalen Sprachen dar
- Als Modell wird eine hinreichend genaue, **zusammenhängende Darstellung** eines bestimmten Bereichs der realen Welt bezeichnet
- Für eine effiziente, direkte Implementierung von formalen Modellen braucht es geeignete **domänenspezifische Sprachen**
- Die technische Software **MATLAB** umfasst eine Programmiersprache, welche eine direkte Implementierung von formalen Modellen unterstützt

Vergleich Programmierparadigmen

- **Objektorientierte Programmierung**

  Entwurf: Darstellung der Struktur des Wissen durch Klassen,
  Sicherheit, Wiederverwendbarkeit der Software

  Implementierung: Allgemeine Sprachen

- **Komponentenorientierte Programmierung**

  Entwurf: Definition von ausführbaren Strukturen

  Implementierung: Austauschbarkeit (Plugins),
  unterschiedliche Sprachen möglich

- **Modellbasierte Programmierung**

  Analyse: Darstellung des Wissens durch formale Modelle

  Entwurf: Modelle bestimmen die Struktur der Software.

  Implementierung: Domänenspezifische Sprachen

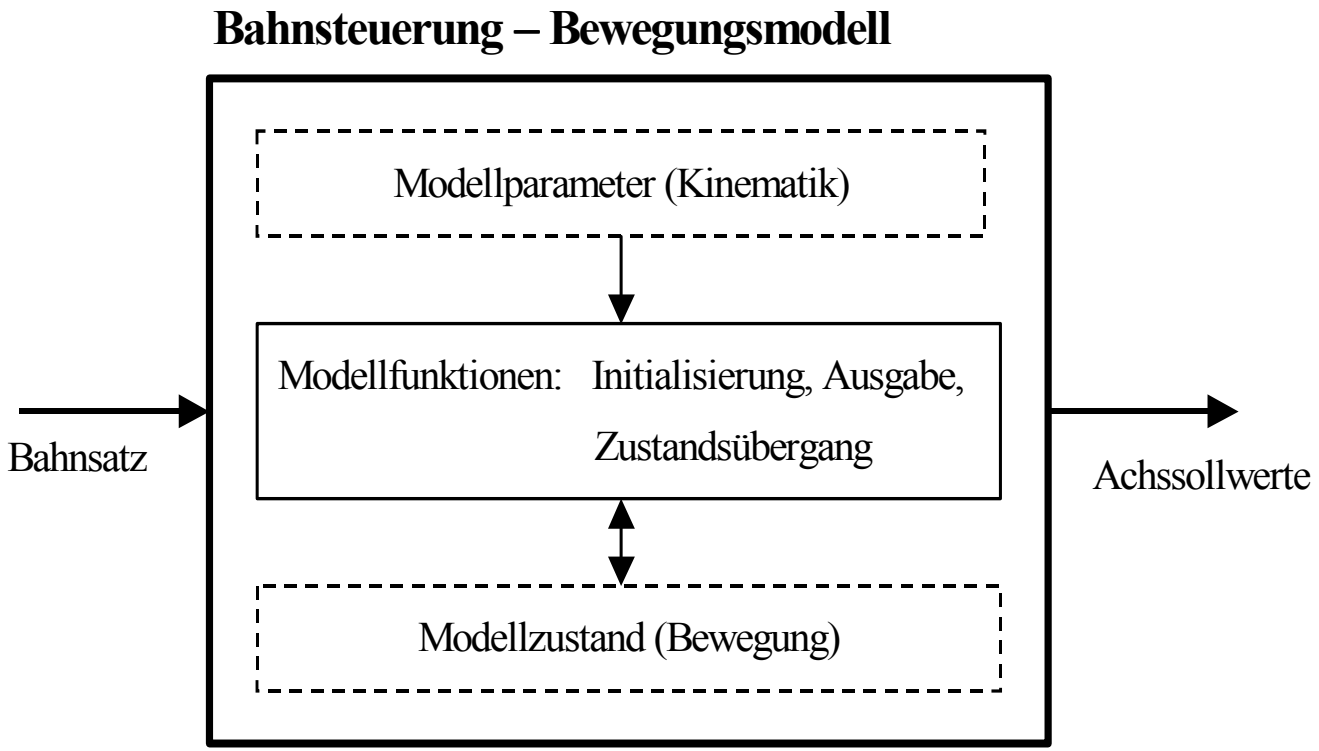
Beispiel – Robotersteuerung MRobot

Funktionsumfang der Software

- 1 – 12 Bewegungsachsen
- Interpolationsarten
  - Punkt zu Punkt (PTP)
  - Linear mit Polynom-Überschleifen
  - Kreis
  - Spline
- **Sensorsteuerung**
- Offline-Programmierung mit **Grafiksimulation in Echtzeit**
- Mächtige Anwender-Programmiersprache

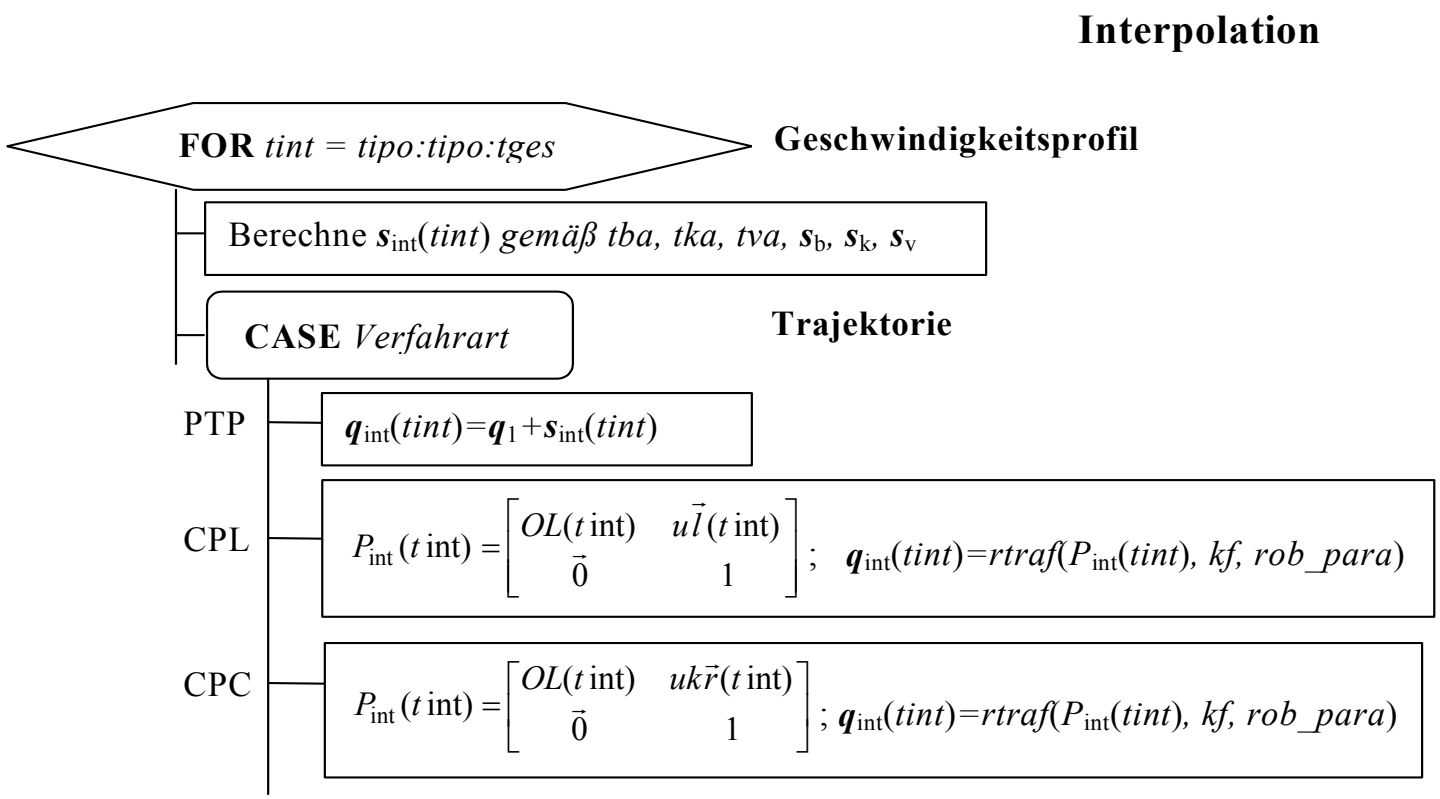
Robotersteuerung MRobot

Modellbasierter Entwurf - Bahnsteuerung



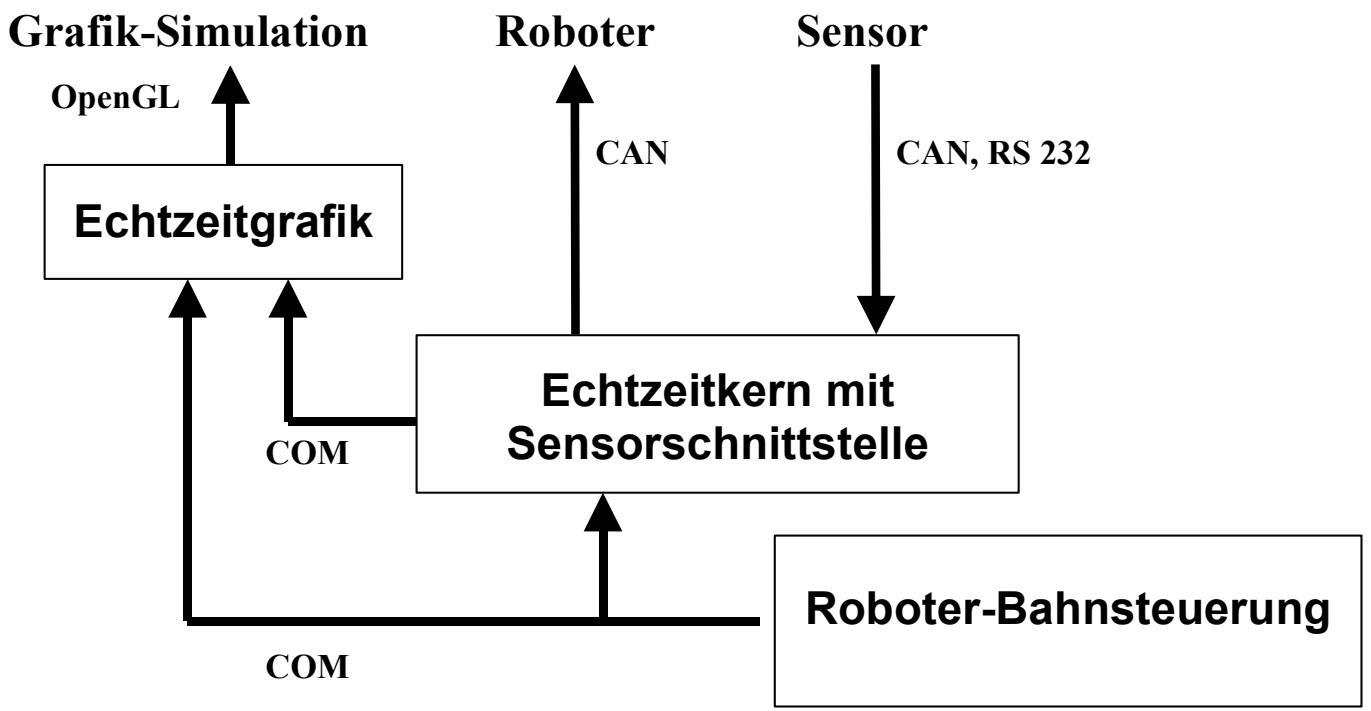
Robotersteuerung MRobot

Modellbasierte Entwurf – Zustandsübergang Bewegung



Robotersteuerung MRobot

Softwarestruktur – Komponenten, Schnittstellen



Robotersteuerung MRobot

Vorteile des gewählten Realisierungsansatzes

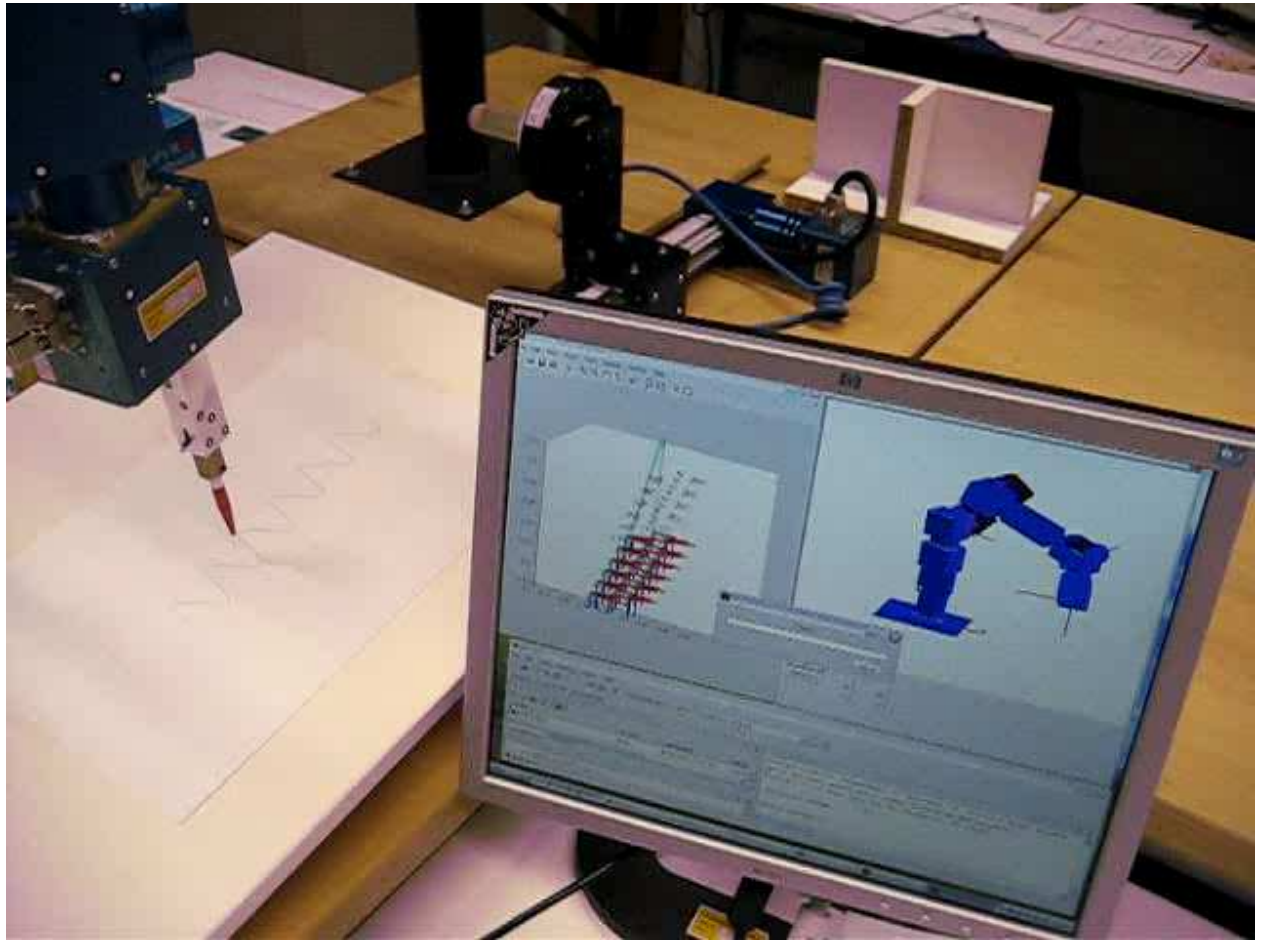
- Nur die Programmierung des **roboterunabhängigen Echtzeitkerns** erfordert intensive C++ - Programmierkenntnisse.
- Alle roboterspezifischen Programmteile sind in einer problemnahen, einfachen Skriptsprache implementiert.
- Die Programmierung und Wartung der umfangreichen, roboterspezischen Software kann deshalb von **Roboterfachleuten ohne umfangreiche Programmierkenntnisse** durchgeführt werden.
- Die **Kosten** für Entwicklung und Wartung der Software werden gegenüber dem konventionellen Realisierungsansatz wesentlich **gesenkt.**

!

!

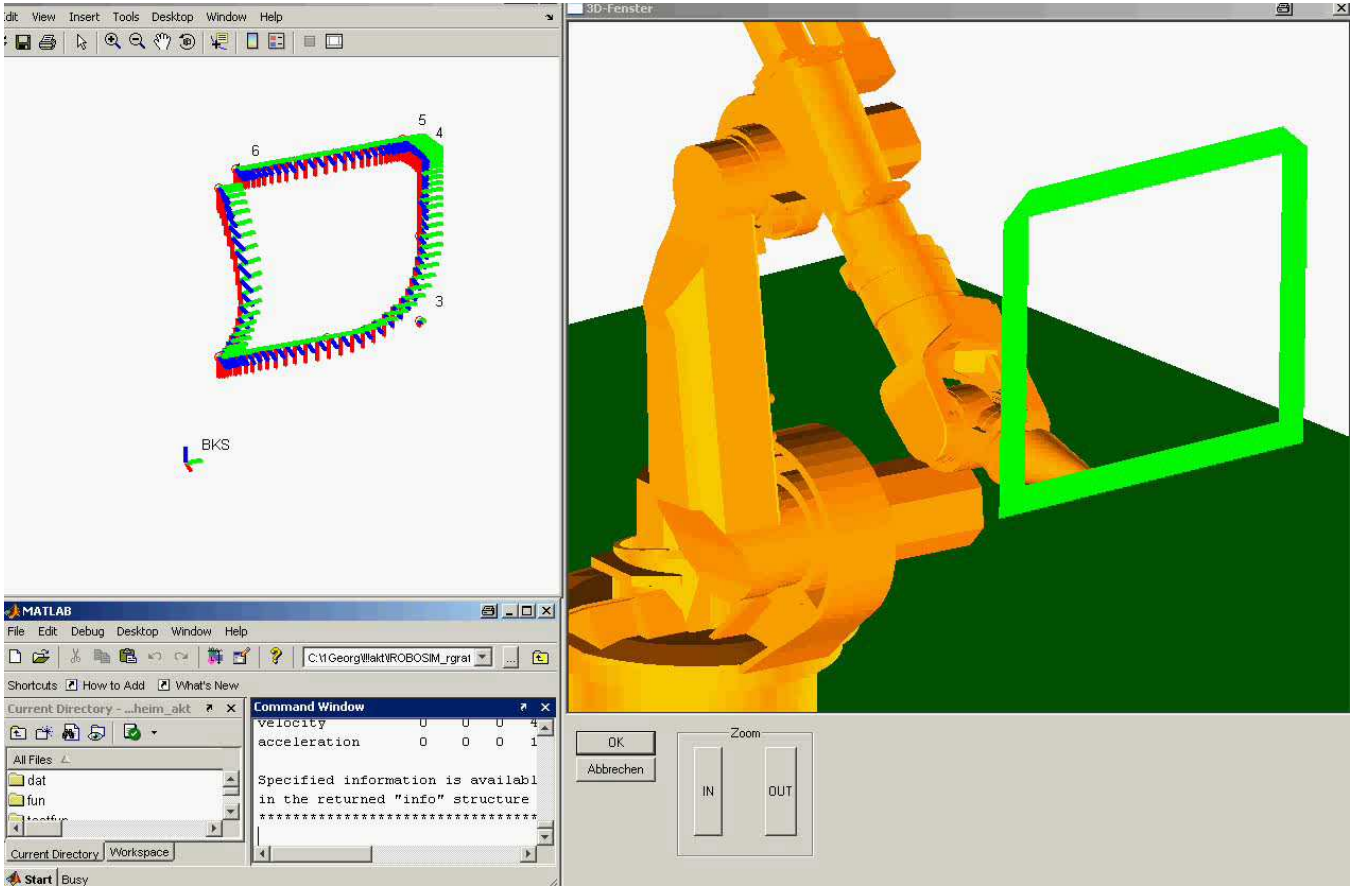
Robotersteuerung MRobot

Synchrone Ausführung und Simulation



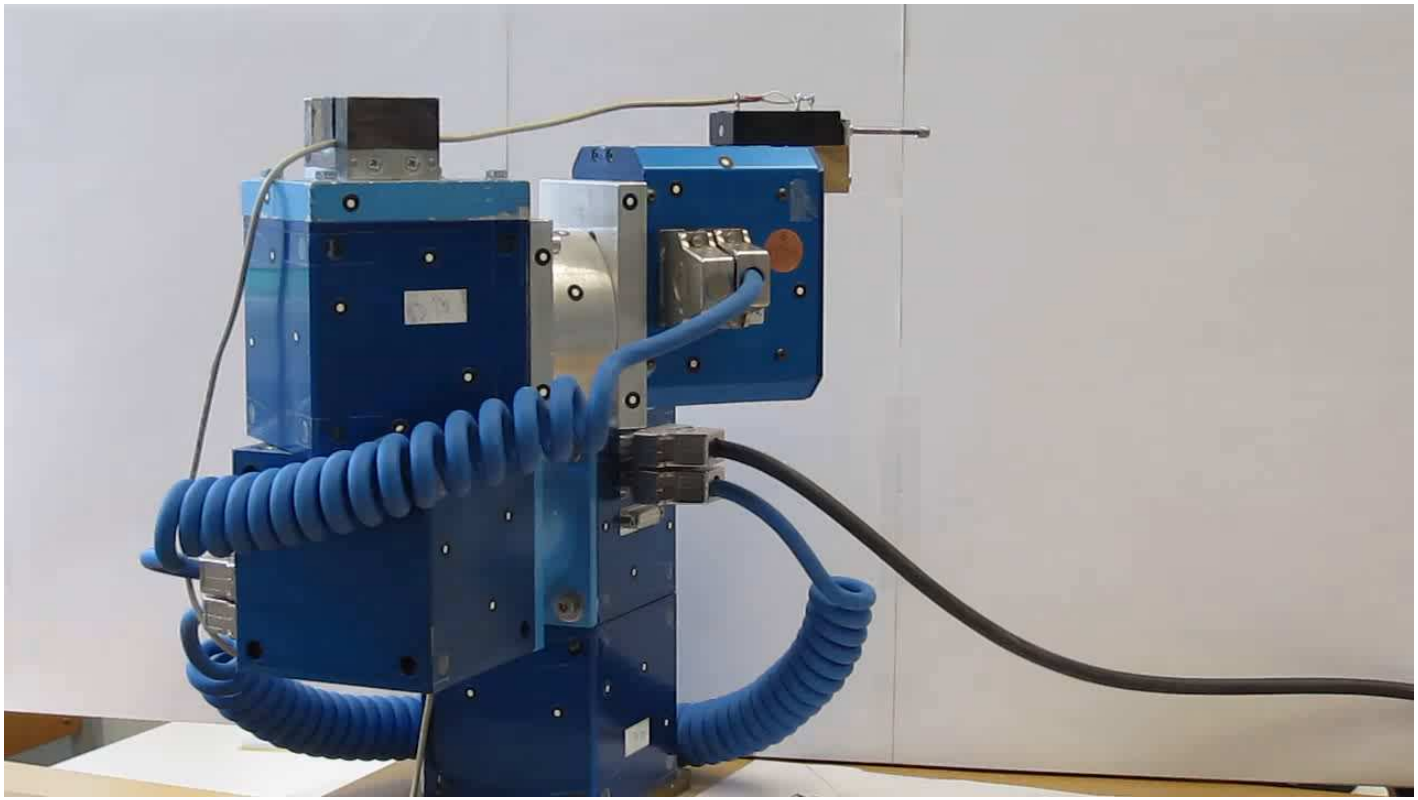
Robotersteuerung MRobot

Simulation KUKA KR15



Robotersteuerung MRobot

Bahnsteuerung durch Abstandssensor (Eigenentwicklung)



Robotersteuerung MRobot

Einfaches Überwachungssystem mit 3D-Webcam

Robotersteuerung MRobot

Objektverfolgung durch 3D-Webcam (Eigenentwicklung)

Labor CIM & Robotik

Darstellung im Internet

- Buch: Robotik mit MATLAB:

  http://www.hs-augsburg.de/stark/robotik_mit_matlab/

- MATLAB User Story:

  http://www.mathworks.de/company/user_stories/userstory20581.html

- Labor CIM & Robotik:

  http://www.hs-augsburg.de/campus/rotes_tor/j-bau/j3/j307/index.html