Karlsruher Institut für Technologie

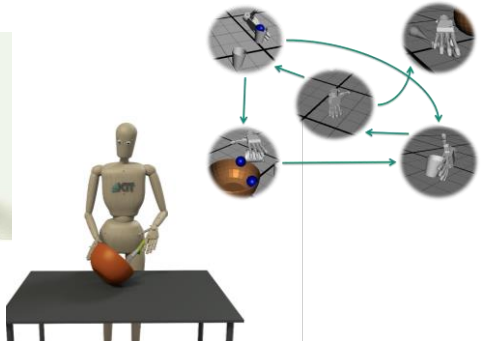
Hochperformante Humanoide Technologien

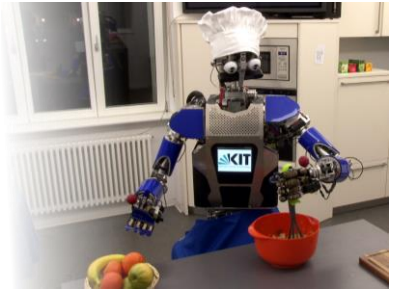
Robotik I: Einführung in die Robotik

Kapitel 10 – Roboterprogrammierung und Programmieren durch Vormachen (PdV)

Tamim Asfour

http://www.humanoids.kit.edu





Inhalt

- **Motivation**
- Klassische Roboterprogrammierung (Übersicht)
- Graphische Programmierung (Statecharts)
- Programmieren durch Vormachen

Motivation - Neue Anforderungen in der Produktion

- Klein- & Kleinstserienfertigung
- Unikatfertigung (z.B. Prototyp)
- Produkte mit:
  - Vielen Ausstattungsvarianten
  - Hoher Rekonfigurierbarkeit

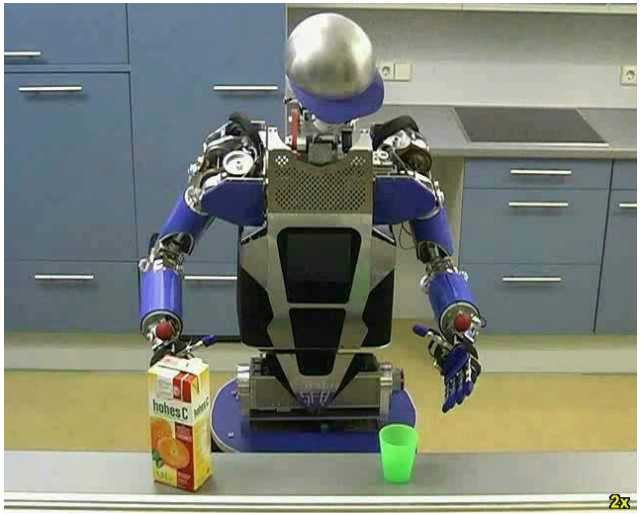
Flexible Fertigung

Motivation - Neue Anforderungen im Servicebereich

- Handel:
  - Kommissionierung und Palettierung von Waren
  - Bestücken von Regalen
- Qualitätssicherung
- Pflege:
  - Unterstützung von in der Rehabilitation und Pflege
- Handwerk:
  - Handhabungen in Schreinereien und Schlossereien

Motivation - Anforderungen an die humanoide Robotik

- Manipulation beliebiger Objekte
- Selbstständiges Lösen komplexer Aufgaben
- Einsatz im menschlichen Umfeld



Komplexe Umgebung!

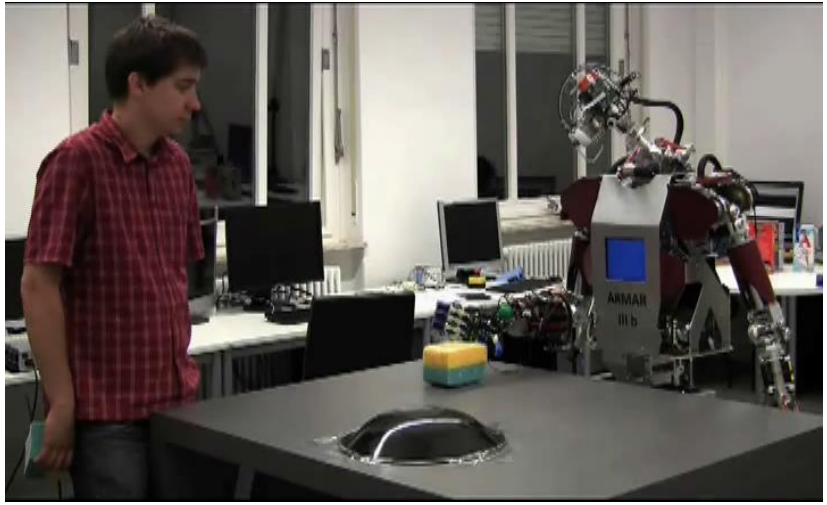
&

Viele Bewegungsfreiheitsgrade!

Methoden zu Programmierung von Robotern?

Programmieren durch Vormachen (PdV)

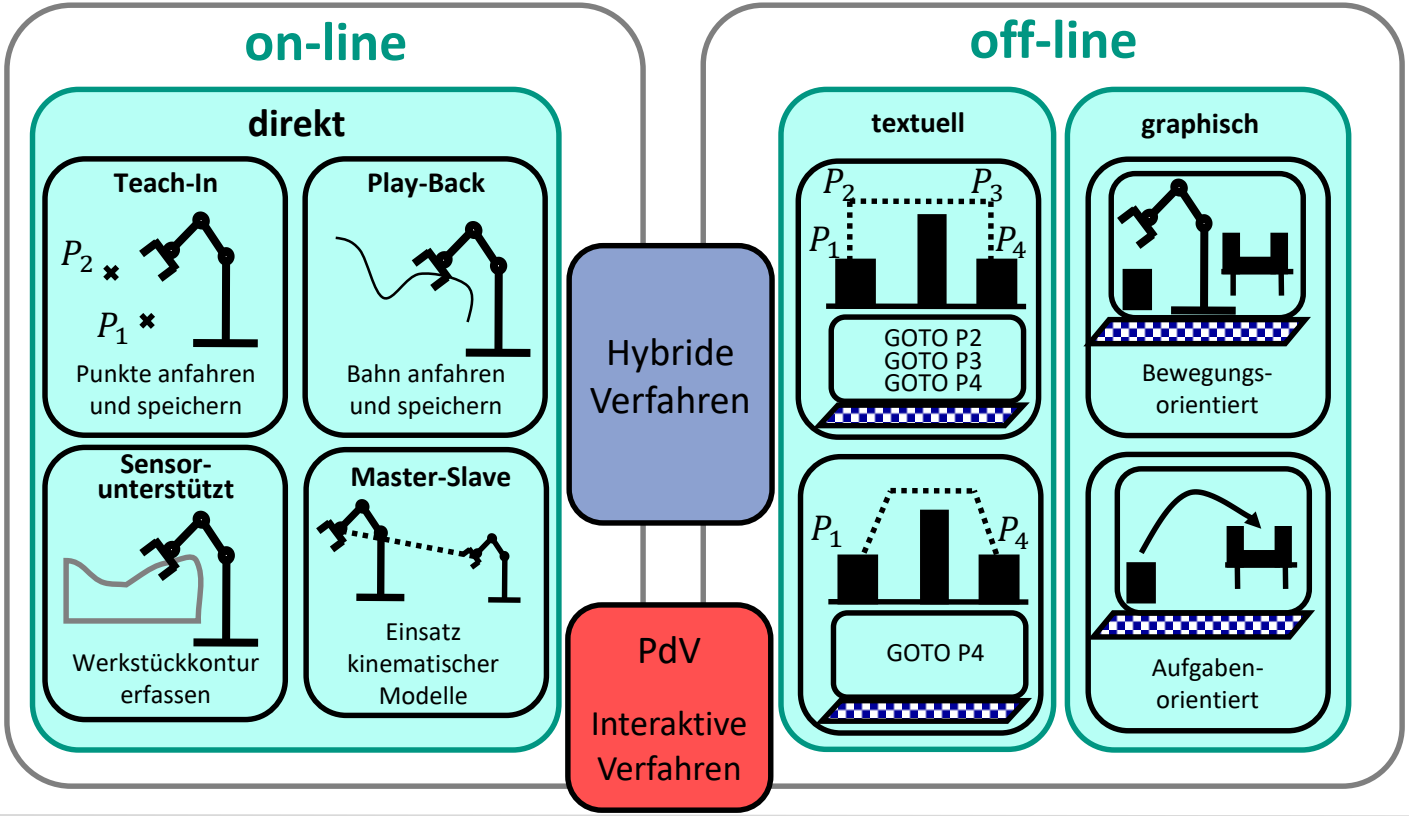
- Beobachtung des Menschen
- Interpretation menschlicher Demonstration
- Abbildung auf Roboter



Inhalt

- Motivation
- **Klassische Roboterprogrammierung (Übersicht)**
- Graphische Programmierung (Statecharts)
- Programmieren durch Vormachen

Roboterprogrammierverfahren



Klassifizierung der Roboterprogrammierverfahren

Kriterien:

I. **Programmierort**
   - **Direkte Programmierung**
   - **Indirekte Programmierung**

II. Abstraktionsgrad der Programmierung
   - Implizite Programmierung
   - Explizite Programmierung

III. Art der Programmierung
   - Direkte Programmierung
   - Textuelle Verfahren
   - Graphische Verfahren
   - Gemischte Verfahren

Roboterprogrammierverfahren

on-line

Die Programmierung erfolgt **direkt am Roboter** (an der Robotersteuerung).

In der Literatur auch **direkte** oder prozessnahe Programmierung genannt.

Hybride Verfahren

off-line

Die Programmierung erfolgt **ohne den Roboter** mit Hilfe textueller, graphischer, interaktiver Methoden.

In der Literatur auch **indirekte** oder prozessferne Programmierung genannt.

Klassifizierung der Roboterprogrammierverfahren

Kriterien:

I. Programmierort
- Direkte Programmierung
- Indirekte Programmierung

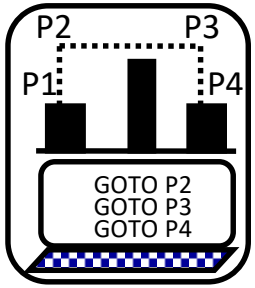
II. Abstraktionsgrad der Programmierung
- **Implizite Programmierung**
- **Explizite Programmierung**

III. Art der Programmierung
- Direkte Programmierung
- Textuelle Verfahren
- Graphische Verfahren
- Gemischte Verfahren

Abstraktionsgrad der Programmierung

Explizite oder roboterorientierte Programmierung (imperativ)

Bewegungen und Greiferbefehle sind direkt in eine Programmiersprache eingebunden.

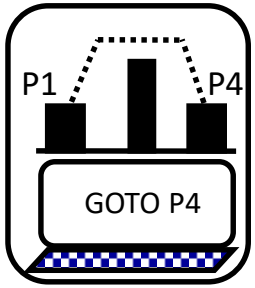


„Wie ist etwas zu tun?“

Abstraktionsgrad der Programmierung

Implizite oder aufgabenorientierte Programmierung (deklarativ)

Die Aufgabe, die der Roboter durchführen soll, wird beschrieben, z.B. in Form von Zuständen.



„Was ist zu tun?“

- Abstrakte Form der Programmierung erfolgt in den Phasen
  1. Modellierung der Umwelt
  2. Spezifikation der Aufgabe
  3. Erzeugung der Roboterprogramme
- u. U. erfolgt vor der Ausführung eine Überprüfung des Roboterprogramms (Simulation)

Klassifizierung der Roboterprogrammierverfahren

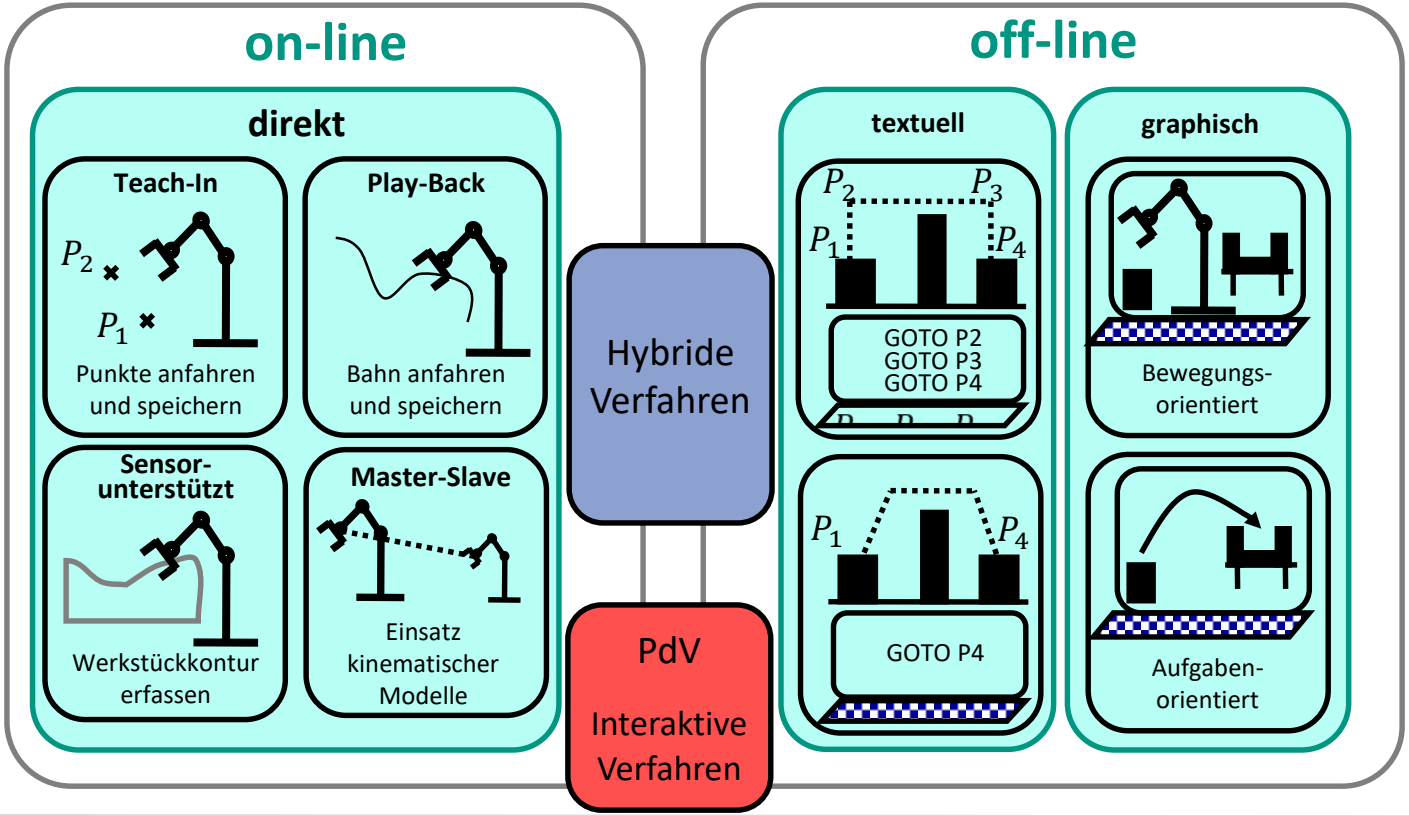
Kriterien:

I. Programmierort
- Direkte Programmierung
- Indirekte Programmierung

II. Abstraktionsgrad der Programmierung
- Implizite Programmierung
- Explizite Programmierung

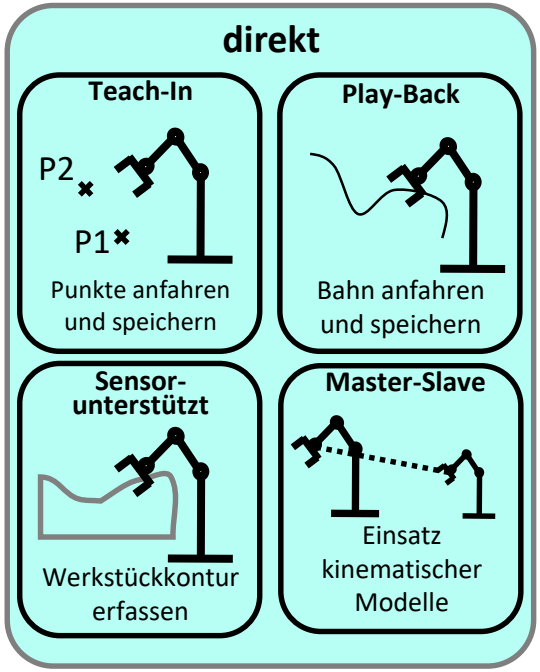
III. Art der Programmierung
- **Direkte Programmierung**
- **Textuelle Verfahren**
- **Graphische Verfahren**
- **Gemischte Verfahren**

Roboterprogrammierverfahren



Direkte Programmierung

- Teach-In Programmierung
- Play-Back Programmierung
- Master-Slave Programmierung (Sonderfall Teleoperation)
- Sensorunterstützte Programmierung

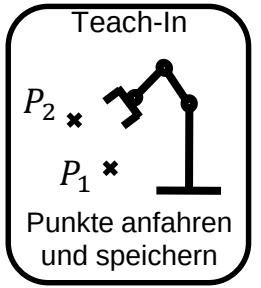


Direkte Programmierung : Teach-In

Teach-In Programmierung

- Anfahren markanter Punkte der Bahn mit manueller Steuerung (Teach Box, Teach Panel)

- Funktionalität einer Teach Box:
  - Einzelbewegung der Gelenke
  - Bewegung des Effektors in 6 Freiheitsgraden
  - Speichern / Löschen von Anfahrpunkten
  - Eingabe von Geschwindigkeiten
  - Eingabe von Befehlen zur Bedienung des Greifers
  - Starten / Stoppen ganzer Programme



Beipiel: Teachbox des Mitsubishi RM-501

5 DoF + Greifer
1,2 kg Nutzlast
Reichweite 45cm

Bewegung:

- Basis
- Schulter
- Ellenbogen
- Handgelenk
- Handdrehung
- Greifer

Direkte Programmierung: Teach-In

Vorgehensweise beim Teach-In

- Anfahren markanter Punkte der Bahn
- Bahn = Folge von Zwischenpunkten
- Speichern der Gelenkwinkelwerte
- Nachträgliche Ergänzung der gespeicherten Werte um weitere Parameter wie Geschwindigkeit, Beschleunigung usw.
- Anwendung:
  - In der Fertigungsindustrie (Punktschweißen, Nieten)
  - Handhabungsaufgaben (Pakete vom Fließband nehmen)

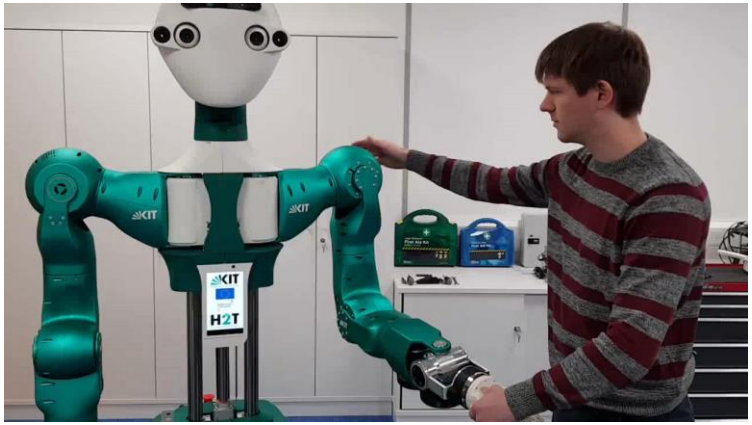
Direkte Programmierung: Play-Back

Play-Back- (manuelle) Programmierung

- Einstellung des Roboters auf Zero-Force-Control (Roboter kann durch den Bediener bewegt werden)
- Auch **kinästhetisches Teaching** genannt
- Abfahren der gewünschten Bahn
- Speichern der Gelenkwerte:
  - Automatisch (definierte Abtastfrequenz) oder
  - Manuell (durch Tastendruck)
- Anwendung:
  - Mathematisch schwer beschreibbare Bewegungsabläufe
  - Integration der handwerklichen Erfahrung des Bedieners
  - Typische Einsatzbereiche sind: Lackieren, Kleben

Direkte Programmierung: Play-Back

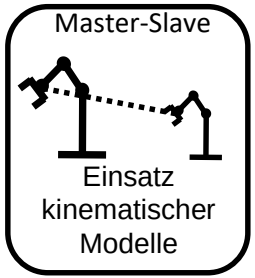
Play-Back- (manuelle) Programmierung



Direkte Programmierung: Play-Back

- Nachteile der Play-Back-Programmierung:
  - Direkter Kontakt mit Roboter → dadurch Sicherheitsrisiko
  - Hoher Speicherbedarf bei hoher Abtastrate (heute kein Problem mehr)
  - Schlechte Korrekturmöglichkeiten

Direkte Programmierung: Master-Slave

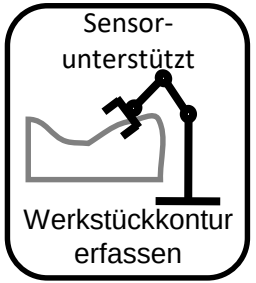


Master-Slave-Programmierung:

- Bediener führt einen kleinen, leicht bewegbaren Master-Roboter (entspricht einem kinematischen Modell des Slave-Roboters)
- Bewegung wird auf den Slave-Roboter übertragen
- Bewegungen werden synchron ausgeführt
- Slave-Roboter wirkt als Kraftverstärker
- Anwendung:
  - Handhabung großer Lasten bzw. großer Roboter
- Vor- und Nachteile:
  - teuer, da zwei Roboter benötigt werden
  - Möglichkeit, auch schwerste Roboter zu programmieren

Direkte Programmierung: Sensorunterstützt

Sensorunterstützte Programmierung



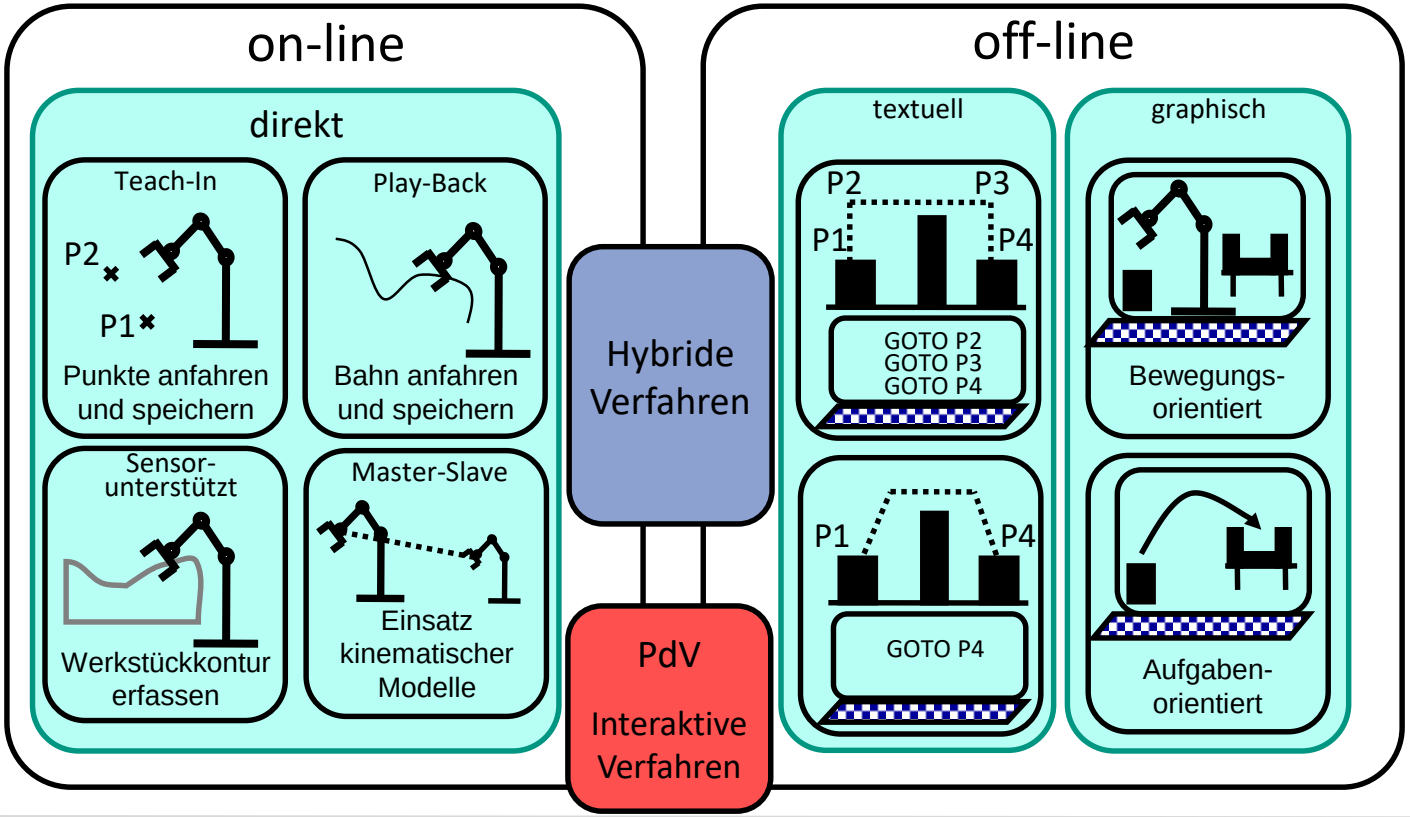
- Manuell
  - Bediener führt Programmiergriffel (Leuchtstift, Laserstift) entlang der abzufahrenden Bahn
  - Erfassung der Bewegung durch externe Sensoren (z.B. Kameras, Laserscanner)
  - Berechnung der inversen Kinematik
  - Abspeichern der Bahn als Folge der Gelenkwinkel
- Automatisch
  - Vorgabe des Start- und Zielpunktes
  - Sensorische Abtastung der Sollkontur (z.B. über Kraft-Momenten-Sensor)
- Nachteile:
  - Fehler bei Erfassung der Bahn; Verdeckung von Teilen der Bahn
  - Kein Einbeziehen der handwerklichen Erfahrung des Bedieners
- Anwendung:
  - Schleifen, Entgraten von Werkstücken

Direkte Programmierung: Zusammenfassung

- **Vorteile:**
  - Schnell bei einfachen Trajektorien
  - Sofort anwendbar
  - Geringe Fehleranfälligkeit
  - Bediener benötigt keine Programmierkenntnisse
  - Kein Modell der Umwelt erforderlich

- **Nachteile:**
  - Hoher Aufwand bei komplexen Trajektorien
  - Nur mit und am Roboter möglich
  - Spezifisch für einen Robotertyp
  - Verletzungsgefahr durch Roboter
  - Keine Adaption an neue Gegebenheiten

Textuelle Programmierverfahren



Textuelle Programmierverfahren

- Programmierung erfolgt mittels erweiterter, höherer Programmiersprachen wie PASRO, VAL, V+ (Unimation/Stäubli), RAPID (ABB), KRL (KUKA), …

  → **Robotersteuerprogramm**

- Vorteile:
  - Programmierung kann unabhängig vom Roboter erfolgen
  - Strukturierte, übersichtliche Programmierlogik
  - Erstellung komplexer Programme
    (Einbezug von Wissensbasis, Weltmodell, Auswertung von Sensoren)

- Nachteile:
  - Bediener benötigt Programmierkenntnisse

Textuelle Programmierverfahren: DIN 66025

- Befehlscodierung nach **DIN 66025**
- Programm = Menge numerierter Sätze

  Beispiel:

  `N70 G00 X20 Z12`

  Werkzeug im Eilgang (`G00`) an Position `X=20` und `Z=12` bewegen.
  (`N` = Satznummer)

- Sprachen:
  - APT (Automatically Programmed Tools) — 1961 MIT
  - EXAPT (Extended Subset of APT) — 1966 TH Aachen

- Beispiel

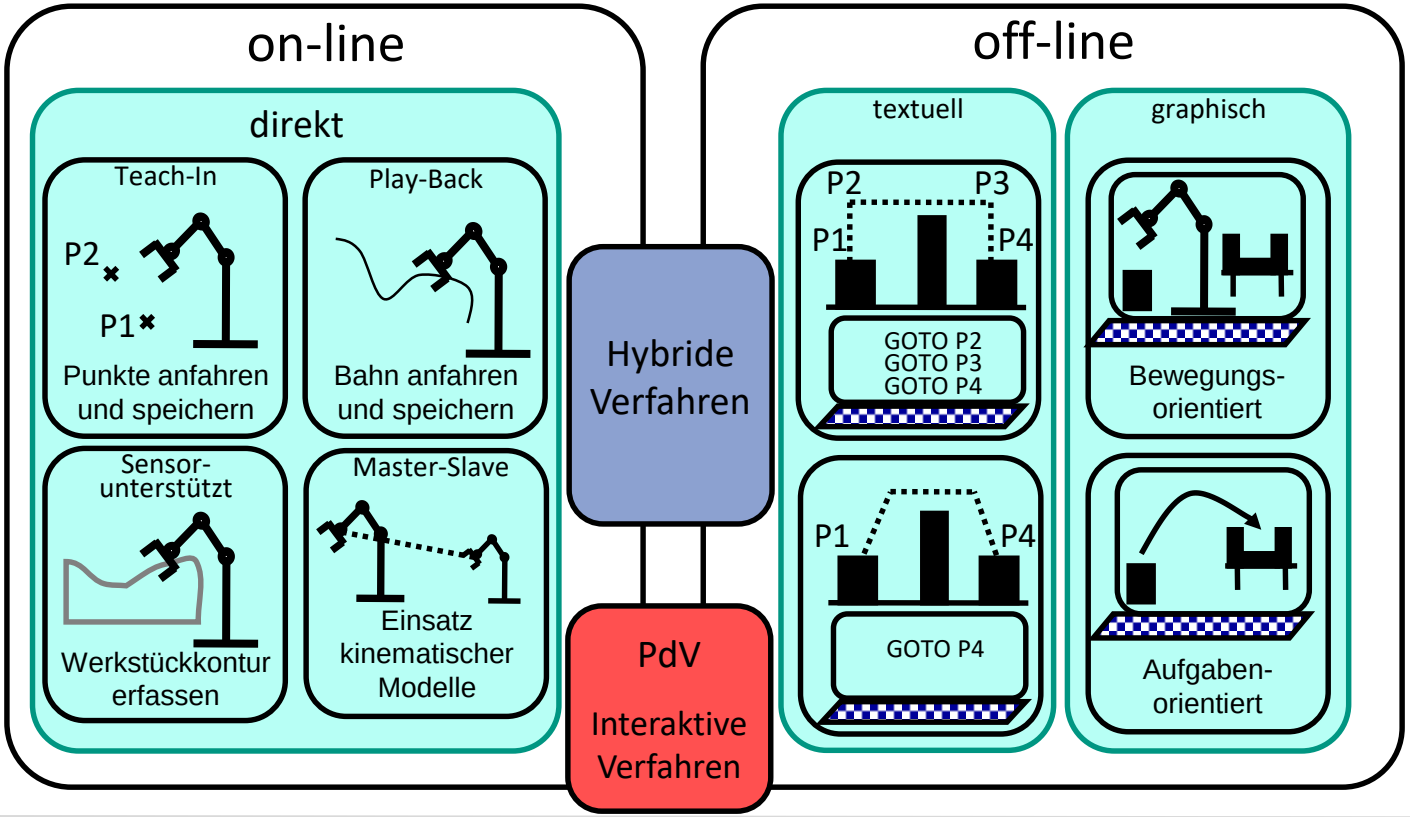
```
P1=POINT/20,12      Variablendefinition
Rapid               Eilgang
GOTO/P1             Positionierung auf P1
```

Textuelle Programmierverfahren: SPS

- **VPS: Verbindungsprogrammierte Steuerung (historisch)**
  - Steuerung erfolgt über Hardware
  - „Programmänderung“ = Hardwareänderung

- **SPS: Speicherprogrammierbare Steuerung (englisch: PLC, Programmable Logic Controller)**
  - Steuerungs- und Regelungsablauf werden programmiert
  - Große Flexibilität

Hybride Programmierverfahren



Hybride Verfahren

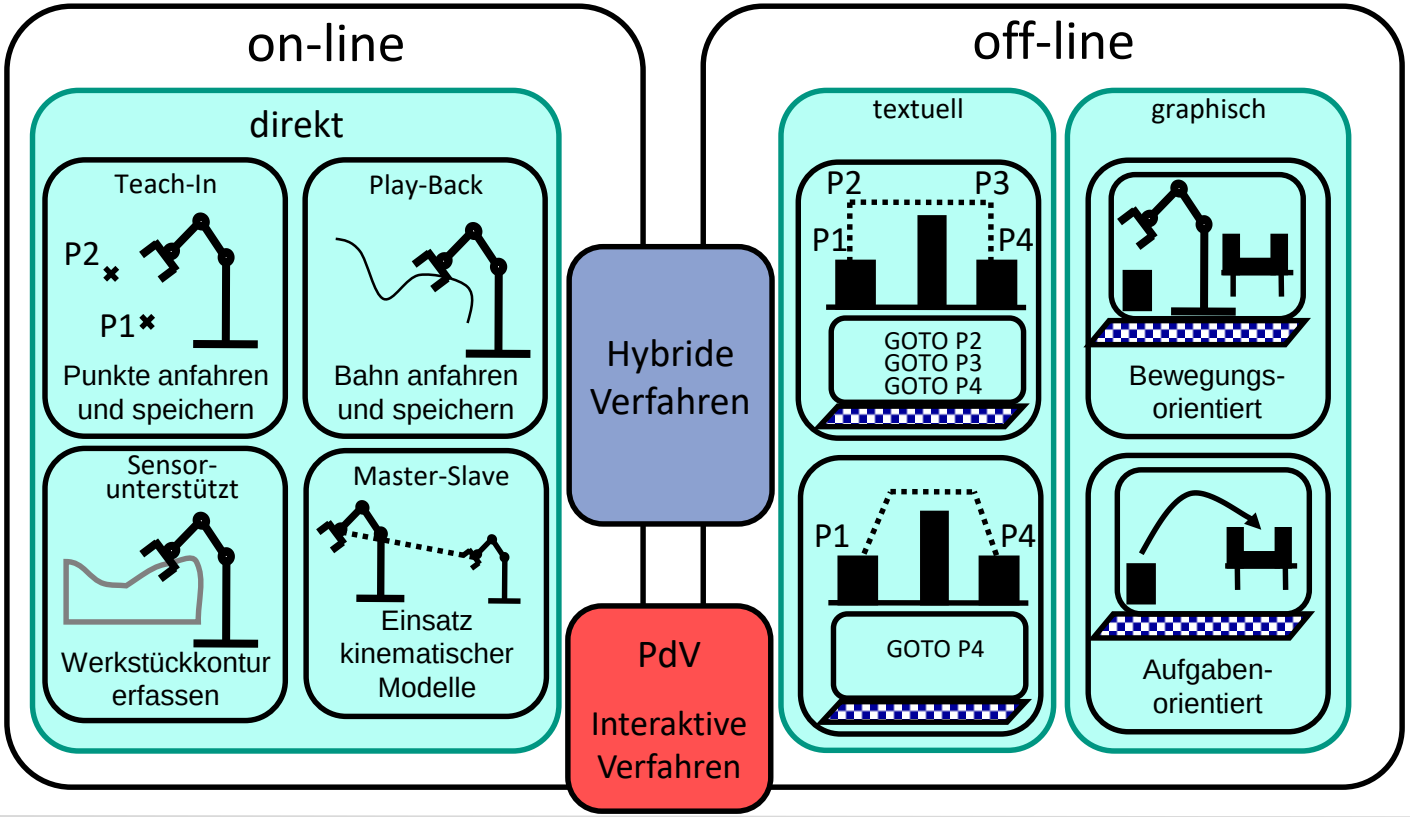
- Graphische Programmierung basierend auf **sensorieller Erfassung der Benutzervorführung**

  → Simulation der Roboterprogramme

- Vorteile:
  - Programmierer benötigt weniger Programmierkenntnisse als bei textueller Programmierung
  - Einfache Programmierung, leichte Fehlererkennung
  - Schnelles Erstellen komplexer Programme (rapid prototyping)

- Nachteile:
  - sensorielle Benutzererfassung noch zu ungenau (wie bei allen online Verfahren)
  - Leistungsfähige Hardware für Signalanalyse, Modellierung, ...
  - Komplexe Modelle werden benötigt

Graphische Programmierverfahren

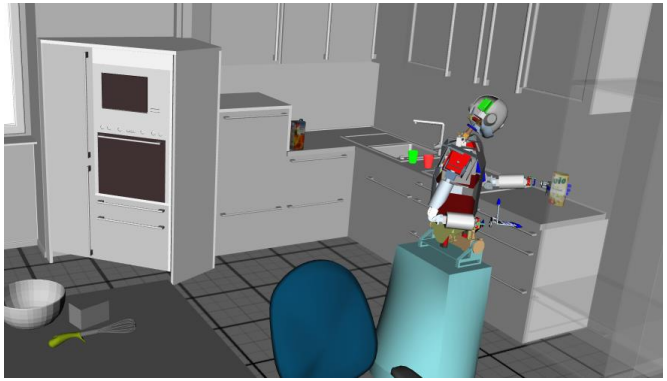


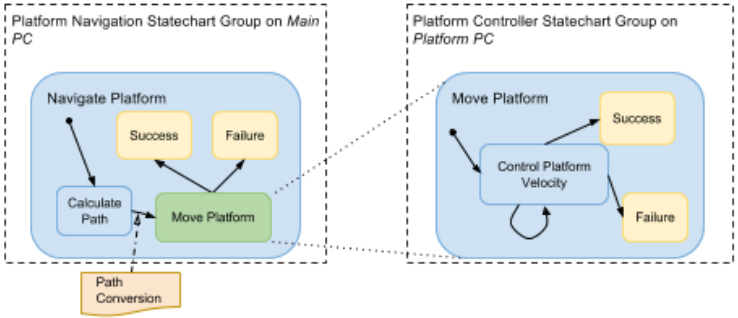
Inhalt

- Motivation
- Klassische Roboterprogrammierung (Übersicht)
- **Graphische Programmierung (Statecharts)**
- Programmieren durch Vormachen

Graphische Roboterprogrammierung

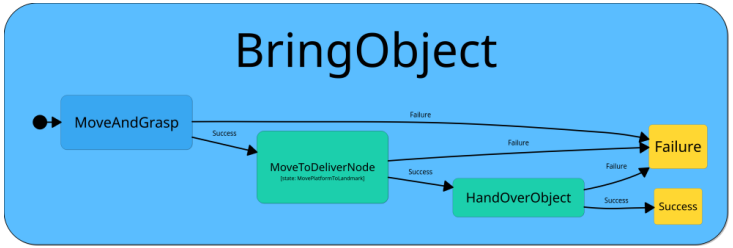
- Zwei grundsätzlich unterschiedliche Varianten
  - **Virtuelles Teach-In**
    - Manipulation von Roboter und Umgebung in 3D Visualisierung
    - Abspeichern der Bewegungen
    - Benötigt exaktes 3D Modell von Roboter und Umgebung
  - **Graphische Modellierungsformalismen**
    - Endliche Zustandsautomaten
    - Petri-Netze
    - Statecharts





Repräsentation von Roboteraktionen mittels Statecharts

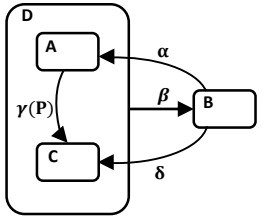
- Warum Statecharts zur Programmierung von Robotern?
- **Lernen** von Aktionen aus Beobachtung ist ein schwieriges Problem
  - Wahrnehmung
  - Embodiment
  - Ausführungsunsicherheiten
- **Textuelle Aktionsprogrammierung** schwierig
  - Systemkomplexität
  - Roboterfähigkeiten stark zustandsbehaftet
  - Fähigkeiten bestehen zumeist aus Subfähigkeiten
  - Textuelle Programmierung unübersichtlich



→Graphische Programmierung von Roboteraktionen: **Statecharts**

Graphische Modellierungsformalismen

- Harel Statechart Formalismus (Harel, 1987)
  - **Graphischer Formalismus** zum Entwurf komplexer Systeme
  - Wichtigste Features:
    - Hierarchisch
    - Interlevel-Transitionen
    - Orthogonalität
    - Zustandsaktionsphasen: Entry, Exit, Throughout
  - **Limitation:** Keine Datenflussspezifikation

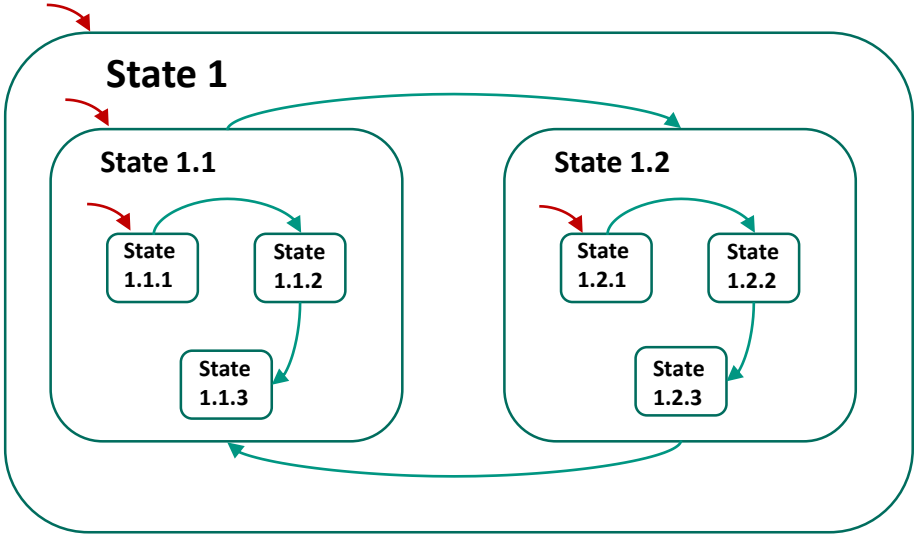


Harel,1987

- A, B, C und D sind Zustände
- Buchstaben auf den Kanten kennzeichnen Ereignisse; in Klammern werden die Bedingungen angegeben

D. Harel, *Statecharts: A visual formalism for complex systems*, Science of computer programming 8.3, pp. 231-274, 1987

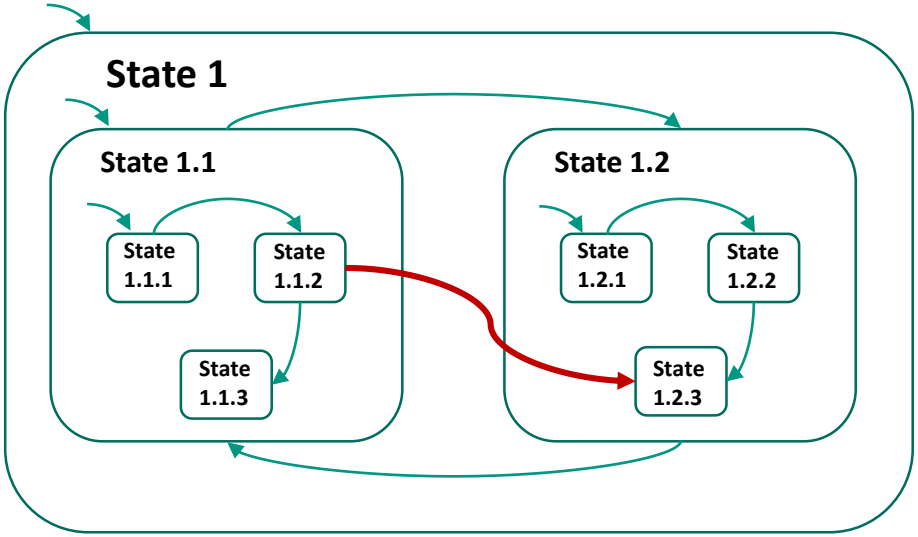
Statecharts (1)



Hierarchie

- State 1 ist Top-Level Zustand
- State 1.1 und State 1.2 sind Unterzustände von State 1
- State 1.1 und State 1.2 enthalten weitere Kind-Zustände
- Beim Betreten eines Zustands wird dessen **initialer Kind-Zustand** betreten

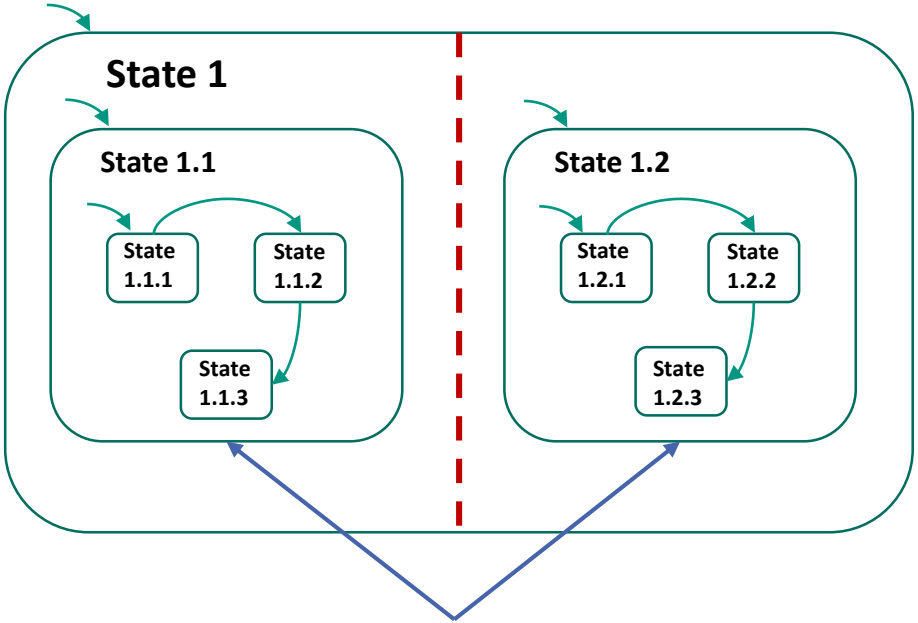
Statecharts (2)



Interlevel Transitionen

- Transitionen können zwischen Hierarchie-Ebenen stattfinden

Statecharts (3)

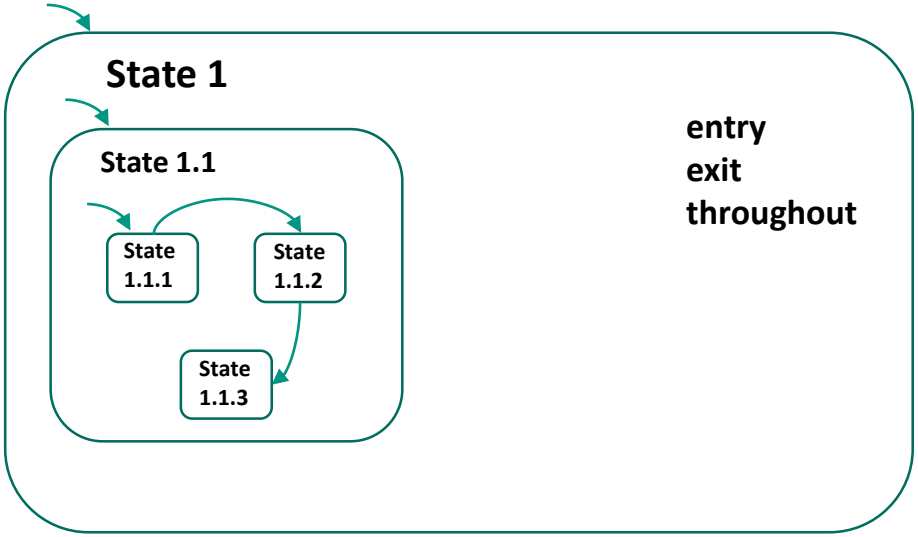


Parallele Ausführung dieser beiden Zustände

Orthogonalität

- Eine **gestrichelte Linie** markiert die **parallele Ausführung** der Zustände State 1.1 und State 1.2

Statecharts (4)



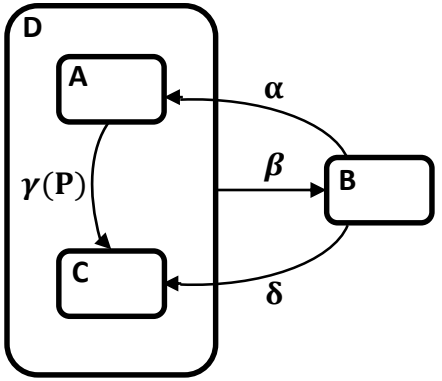
Zustandsphasen

- Beim **Betreten** von State 1 wird **zuerst** **Aktivität *entry*** ausgeführt, bevor State 1.1 betreten wird
- **Aktivität *throughout*** wird ausgeführt, **während** State 1.1 ausgeführt wird
- Nach Beendigung von State 1.1 und **vor dem Verlassen** von State 1 wird **Aktivität *exit*** ausgeführt

Limitation des Harel Statchart Formalismus

- Keine **Datenflussspezifikation**
  - Datenfluss in der Robotik ist notwendig; Ergebnisse von Zuständen werden in Folgezuständen benötigt
  - Beispiel: Objektlokalisierung wird für Visual Servoing oder inverse Kinematik benötig

- Wiederverwendbarkeit erfordert eine **Adaption von Parametern**
  - Kontrollparameter
  - Kinematikparameter
  - Objektparameter
  - ….

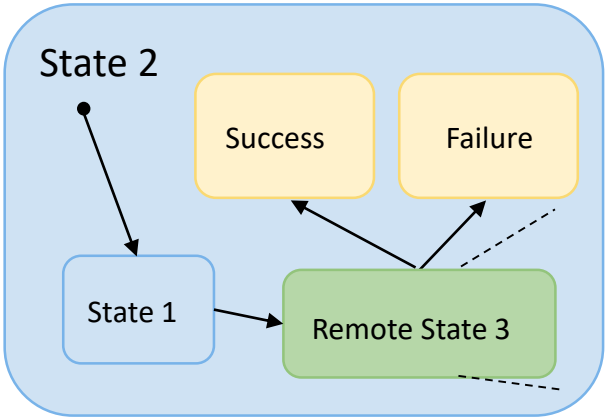


Harel,1987

Erweiterung des Harel Statechart Formalismus am H²T

ArmarX Statechart Erweiterung: https://armarx.humanoids.kit.edu

- **Datenflussspezifikation**: Transitionsbasierter Datenfluss
- **Keine Inter-level Transitionen**, da sie die Wiederverwendbarkeit verhindern
- **Erfolgs- und Misserfolgszustände** in jedem Zustand
- Dynamische Struktur
- Verteilung über mehrere Host-PCs
- Verbindung von graphischer Kontroll- und Datenflussspezifikation mit C++ Benutzer-Code

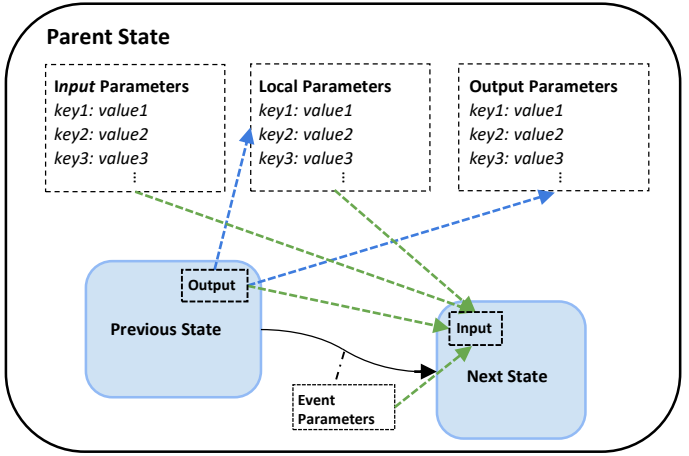


M. Wächter, S. Ottenhaus, M. Kröhnert, N. Vahrenkamp and T. Asfour, *The ArmarX Statechart Concept: Graphical Programming of Robot Behaviour*, Frontiers - Software Architectures for Humanoid Robotics, 2016

Statechart Erweiterung

Transitionsbasierter Datenfluss

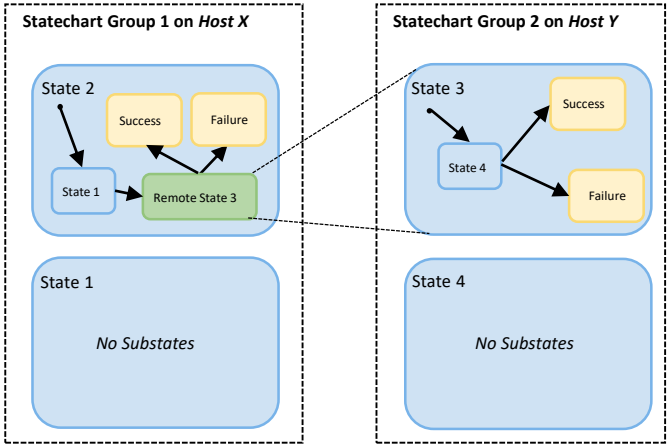
- Beliebige Datentypen
  - Grundlegende Datentypen: int, float, string, …
  - Komplexe Datentypen: Position, 6D-Pose, Matrizen, Listen, …
- Drei Parametersätze pro Zustand
  - Eingabe (Input), Lokale Parameter (Local), Ausgabe (Output)
- Parameterabbildung pro Transition
  - von Ausgabeparameter des **Quellzustandes**
  - auf Eingabeparameter des **Zielzustandes**
- Spezifizierter Datenfluss
  → Keine Seiteneffekte durch globale Variablen



Parameterabbildungen

Statechart Erweiterungen (1)

- **Verteilung über mehrere Host-PCs**
  - Transparente Verteilung von Statecharts
    - Netzwerk Middleware *ZeroC Ice* (*https://zeroc.com*)
  - Kind-Zustände können Zeiger auf entfernt liegende Zustände sein (grüner Zustand)
  - **Vorteile:**
    - Lastverteilung
    - Erhöhte Robustheit und Fehlertoleranz
    - Näher an eingesetzter Hardware (Sensorik, Aktorik)

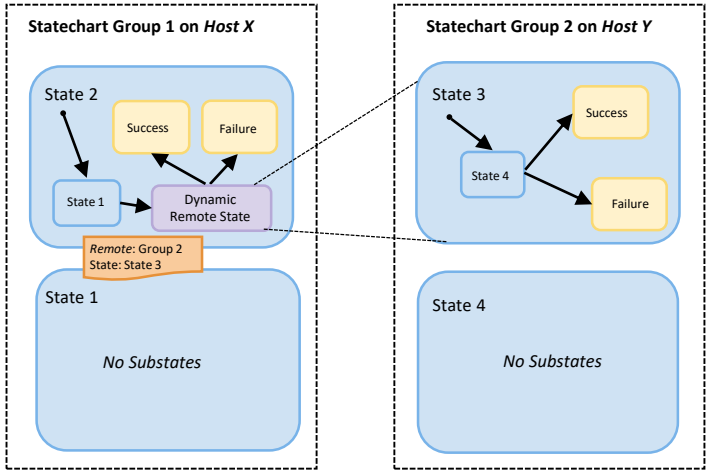


Verteilte Statecharts

Statechart Erweiterungen (2)

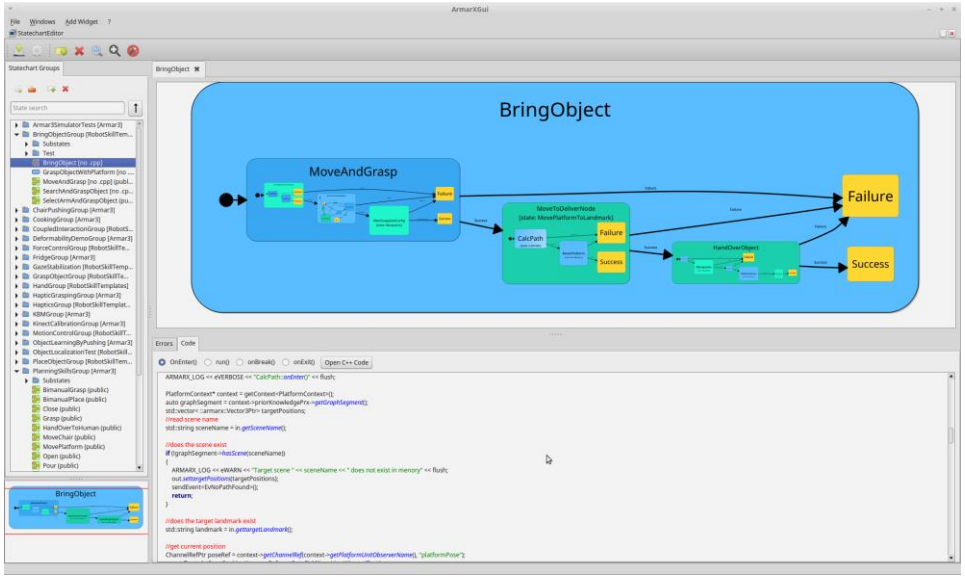
- **Dynamische Struktur**
  - Austausch von Unterzuständen zur Laufzeit
  - Transparente Verbindung zu beliebigem entfernten Zustand
  - Unterzustand durch Parameterabbildung spezifiziert
  - Einsatz: Ausführung von generierten Plänen

- **Vollständig integriert in das Roboter-Entwicklungsframework *ArmarX***
  - Graphischer Editor
  - Verbindung zu allen Roboterkomponenten
  - Online Inspektion der aktuellen Ausführung

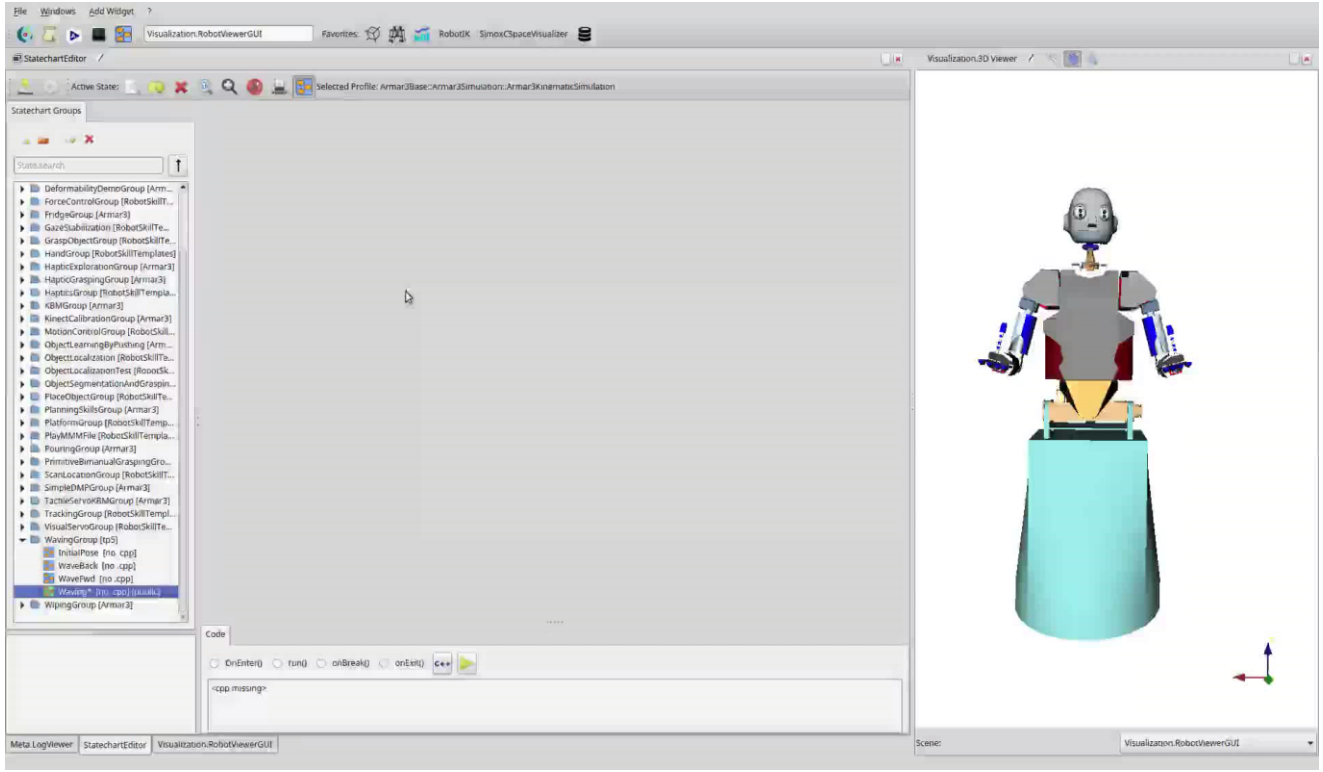


Graphischer Statechart Editor in ArmarX

- Tool zur graphischen Spezifikation von
  - Kontrollfluss
  - Datenfluss
  - Abhängigkeiten zu externen Roboterkomponenten
  - Direkt verlinkt mit C++ Code

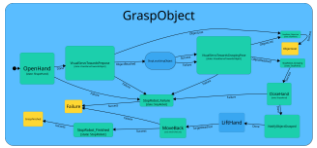


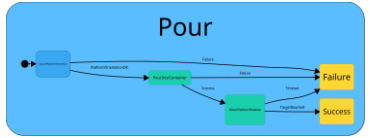
Graphischer Statechart Editor in ArmarX: Beispiel

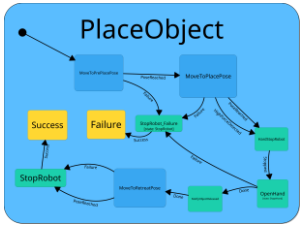


Manipulationsfähigkeiten mittels Statecharts

- ARMAR-III: Zur Zeit ~330 Zustände implementiert







Inhalt

- Motivation
- Klassische Roboterprogrammierung (Übersicht)
- Graphische Programmierung (Statecharts)
- **Programmieren durch Vormachen**
  - Kernfragen von Programmieren durch Vormachen
  - Wahrnehmung menschlicher Demonstrationen
  - Lernen von Task-Modellen
  - Ausführung auf einem Roboter

Literatur

- Detaillierte Behandlung in der Vorlesung Robotik-II
- Hier: Grundlagen
- Literatur

A. Billard, S. Calinon, R. Dillmann and S. Schaal, Robot Programming by Demonstration, In *Handbook of Robotics*, Springer, pp. 1371-1394, 2008.

B. Argall, S. Chernova, M. Veloso and B. Browning, *A survey of robot learning from demonstration*, Robotics and Autonomous Systems, 2009.

Grundidee

- Lernen aus Beobachtung des Menschen

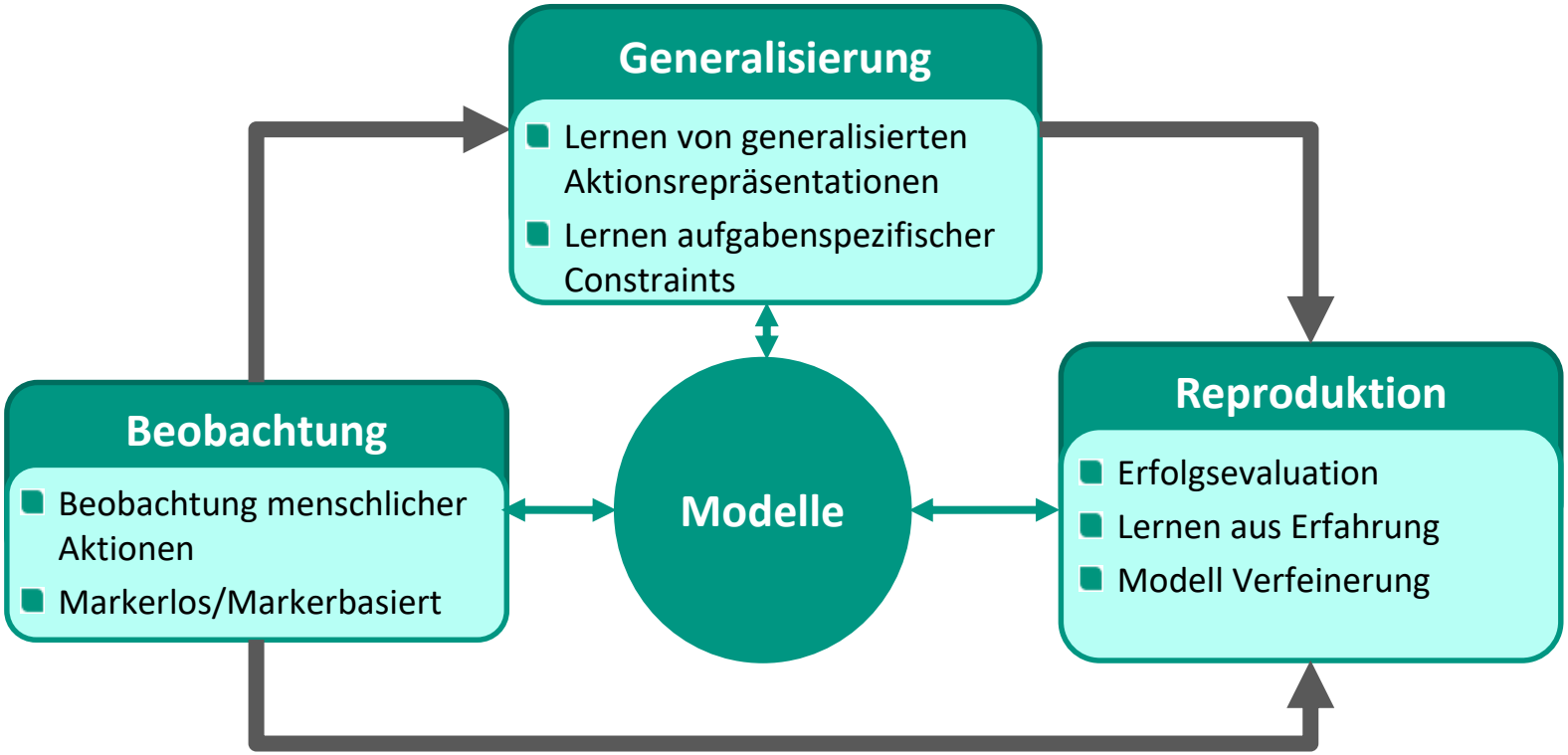
Programmieren durch Vormachen (PdV)

- Hauptziel von Programmieren durch Vormachen
  - **Intuitive** Roboterprogrammierung **ohne Expertenwissen**
  - Lernen von Task-Modellen aus Demonstrationen des Menschen

- PdV impliziert Lernen und Generalisierung und ist deshalb kein Playback Verfahren.

- Das Thema ist auch bekannt als
  - Programming by Demonstration (PbD)
  - Learning from/by demonstration (LfD)
  - Learning from human demonstration (LfD)
  - Apprenticeship learning
  - Imitation learning (IL)

Lernen durch Beobachtung des Menschen



Drei Gründe für PdV / Imiltationslernen

- Mächtiger Mechanismus zur **Komplexitätsreduktion** des Suchraums während des Lernens (im Gegensatz zum Ausprobieren aller Möglichkeiten).
  - Gute Demonstrationen können als Startpunkt des Lernens einer Fähigkeit gewählt werden
  - Schlechte Demonstrationen können entweder aus dem Suchraum eliminiert werden oder auch zum Lernen aus negativen Beispielen verwendet werden
- Impliziter Mechanismus zum **Trainieren von Robotern**, der das explizite und mühsame händische Programmieren reduziert oder sogar ganz eliminiert.
- Verständnis der Kopplung und Lernen relevanter Beziehungen zwischen **Perzeption und Aktion**.

Herausforderungen in PdV

1. Wer soll imitiert werden?

- **Lehrerauswahl:** wähle einen Lehrer von dessen Verhalten der Imitierende am meisten profitiert.
- Problem wird bei den meisten Methoden durch Auswahl eines dedizierten Lehrers vermieden.

Herausforderungen in PdV / Imiltationslernen

2. Wann soll imitiert werden?

- Der Imitierende muss die Bewegung **segmentieren** und Start und Ende der demonstrierten Aufgabe/Aktion ermitteln.
- Der Imitierende muss entscheiden, ob es angemessen ist das Beobachtete im **aktuellen Kontext** auszuführen und wie oft es wiederholt werden muss.
- Problem wird bei den meisten Methoden durch explizites Kennzeichnen von Start und Ende der Demonstration vermieden.

Herausforderungen in PdV / Imiltationslernen

3. Was soll imitiert werden?

- Wie findet man heraus, **welche Aspekte der Demonstration von Interesse sind**?
  Bei bestimmten Aufgaben sind es die Bewegungen (Trajektorien) des Menschen.
  In anderen Situationen sind das Ergebnis und die Effekte von Aktionen wichtig.

- Manche beobachtbaren **Eigenschaften sind irrelevant** und können **ignoriert** werden.
  Ist es zum Beispiel wichtig, dass der Vorführende immer von Norden her zum Tisch läuft?

- Bei kontinuierlichen Regelungsaufgaben entspricht diese Aufgabe der Bestimmung des **Merkmalraums** für das Lernen, sowie der **Constraints** und der **Kostenfunktion.**

- Bei diskreten Regelungsaufgaben, die z.B. mit Reinforcement Learning und symbolischem Reasoning behandelt werden, entspricht diese Aufgabe der **Definition** des **Zustands-** und **Aktionsraums** und dem automatischen Lernen von **Vor- und Nachbedingungen (Constraints)**.

Herausforderungen in PdV / Imiltationslernen

4. Wie soll imitiert werden?

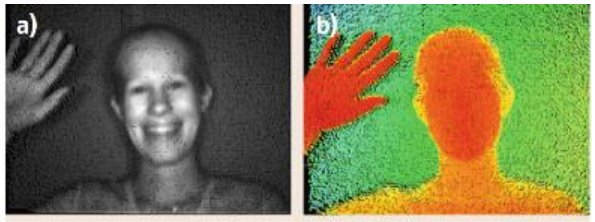
- Bestimme, wie der Roboter das **Gelernte** (Aktionen und deren Sequenzen) **ausführen** wird, um die Metrik zu maximieren, die beim „*Was soll imitiert werden?*“ Problem gefunden wurde.
  - Auf Grund **unterschiedlicher Kinematik** kann z.B. ein Roboter menschliche Bewegungen nicht auf menschenähnliche Art und Weise reproduzieren.
  - **Beipsiel:** Darf z.B. ein Roboter mit Rädern ein Objekt anfahren, wenn der Lehrer es mit dem Fuß angestoßen hat, oder muss er stattdessen seinen Arm einsetzen?
- Der Roboter muss eine **Abbildung** von der Perzeption zu einer Sequenz eigener Bewegungen lernen. Deshalb bestimmen das **Embodiment** und die Körpereinschränkungen, wie beobachtete Aktionen imitiert werden können (**Korrespondenzproblem**).

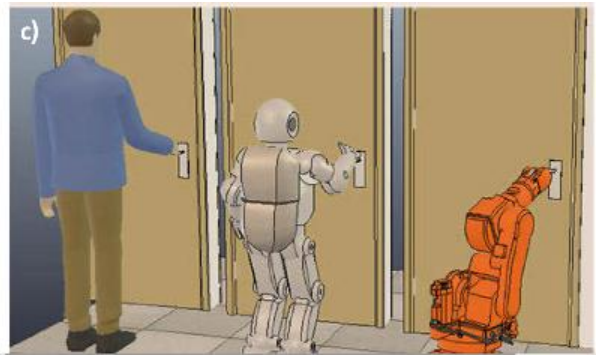
Perzeptive und Physikalische (Nicht)Äquivalenz

Zwei unterschiedliche Arten wie bestimmt werden kann, ob der Zustand des Lehrers (Mensch) und des Imitators (Roboter) korrelieren.

- **Perzeptive Äquivalenz**: Auf Grund von Unterschieden zwischen menschlicher und robotischer **Sensorfähigkeiten** kann die gleiche Szene sehr unterschiedlich aussehen. Ein Mensch erkennt beispielsweise andere Menschen und Gesten anhand von Farbe und Intensität, während ein Roboter **Tiefeninformation** dazu verwendet.

- **Physikalische Äquivalenz**: Auf Grund unterschiedlicher **Embodiements**, führen Mensch und Roboter z.B. unterschiedliche Aktionen aus, um den gleichen physikalischen Effekt zu erzielen.





[Billard et al. 2008]

Herausforderungen in PdV Imiltationslernen

- Wer soll imitiert werden?
- Wann soll imitiert werden?

Größtenteils unerforscht!

- Was soll imitiert werden?
- Wie soll imitiert werden?

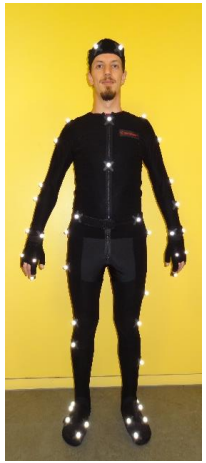
Lernen einer Fähigkeit oder einer Aufgabe!

Inhalt

- Motivation
- Klassische Roboterprogrammierung (Übersicht)
- Graphische Programmierung (Statecharts)
- **Programmieren durch Vormachen**
  - Kernfragen von Programmieren durch Vormachen
  - Wahrnehmung menschlicher Demonstrationen
  - Lernen von Task-Modellen
  - Ausführung auf einem Roboter

Erfassung der menschlichen Bewegung

- Die **Erfassung der menschlichen Bewegung** ist Ausgangspunkt für die Programmierung durch Vormachen



Zwei Möglichkeiten

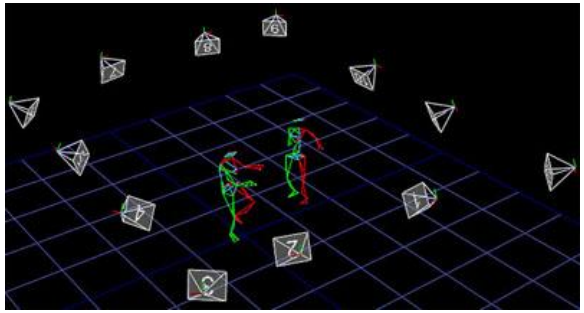
- **Marker-basierte Verfahren:**
  - Marker werden an vorher festgelegten anatomischen Landmarken des menschlichen Körpers angebracht
  - Marker sind entweder **aktiv** (z.B.: Codierung mit LED) oder **passiv** (z.B.: reflektierend)
  - Mensch muss zur Erfassung vorbereitet („**vermarkert**“) werden
- **Marker-lose Verfahren:**
  - Direkte Rekonstruktion der menschlichen Pose aus Kamera-Daten (RGB-/Tiefendaten)
  - Keine Vermarkerung des Menschen erforderlich

Marker-basierte optisch-passive Bewegungserfassung

- Lokalisierung von infrarot-reflektierenden passiven Markern mit Hilfe eines **Mehr-Kamera-Systems**
- Weit verbreitete Lösung, verschiedene kommerzielle Anbieter (z.B. **VICON** und **OptiTrack**)

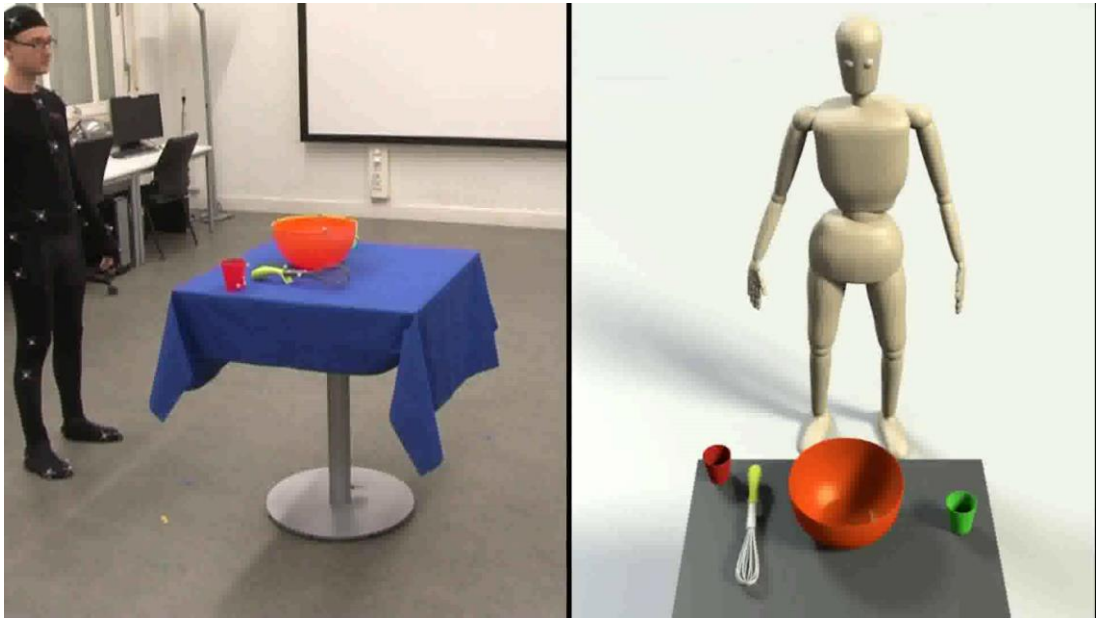
- **Vorteile**:
  - Hohe räumliche Genauigkeit (Submillimeter-Bereich)
  - Hohe zeitliche Auflösung (1-2 kHz möglich)

- **Nachteile**:
  - Teuer und hoher technischer Aufwand
  - Aufwand für Nachbearbeitung durch Verdeckungen und Labeling der Marker
  - Anforderungen an Aufnahmebereich (z.B. nur indoor möglich)



Beisipiel

- KIT Whole-Body Human Motion Database



https://motion-database.humanoids.kit.edu

Marker-basierte optisch-aktive Bewegungserfassung

- Marker übertragen eine **codierte Kennung**, z.B. mit Infrarot-LED

- **Vor-/Nachteile** ähnlich zu optisch-passiven Systemen, aber
  - Vereinfache Handhabung des Marker Labelings
  - Dafür: Batterien/Kabel zur Stromversorgung jedes Markers notwendig

© Qualisys

IMU-basierte Bewegungserfassung

- Anbringung von **inertialen Messeinheiten (IMUs)** an anatomisch definierten Punkten des menschlichen Probanden (z.B. 17 beim Xsens MVN)

- **Vorteile**:
  - Geringe Anforderungen an Aufnahme-Umgebung (z.B. auch outdoor)
  - Genaue Bestimmung der menschlichen Pose

- **Nachteile**:
  - Exakte Platzierung der IMUs erforderlich
  - Keine exakte Positionsbestimmung des Menschen im globalen Koordinatensystem möglich (IMU-Drift)

© Xsens

Mechanische Bewegungserfassung

- Direkte Messung von Gelenkwinkeln (z.B. Potentiometer an Exoskeletten)

- **Vorteile**:
  - Geringe Anforderungen an Aufnahme-Umgebung (z.B. outdoor)

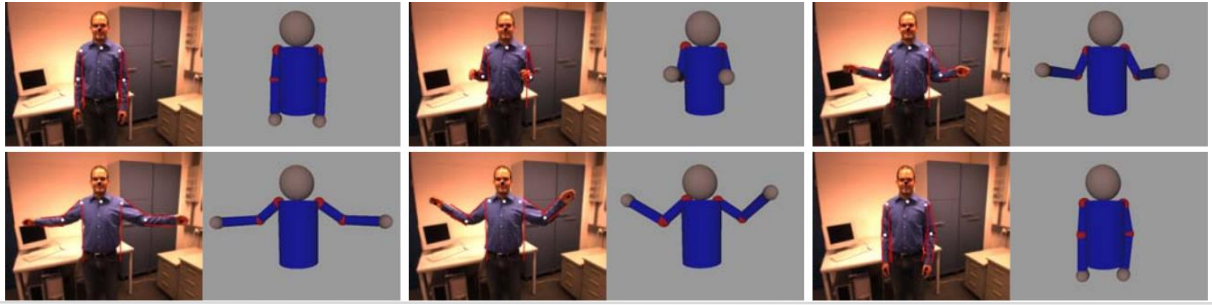
- **Nachteile**:
  - Einschränkung der menschlichen Bewegung durch Exoskelett → Siehe Vorlesung „Anziehbare Robotertechnologien“ im SS
  - Hoher technischer Aufwand
  - Keine Positionsbestimmung des Menschen im globalen Koordinatensystem möglich

Alexander Gmiterko, Tomáš Lipták, Motion Capture of Human for Interaction with Service Robot, *American Journal of Mechanical Engineering*. 2013, 1(7), 212-216 doi:10.12691/ajme-1-7-12
http://pubs.sciepub.com/ajme/1/7/12/index.html

https://metamotion.com

Marker-lose optische Bewegungserfassung

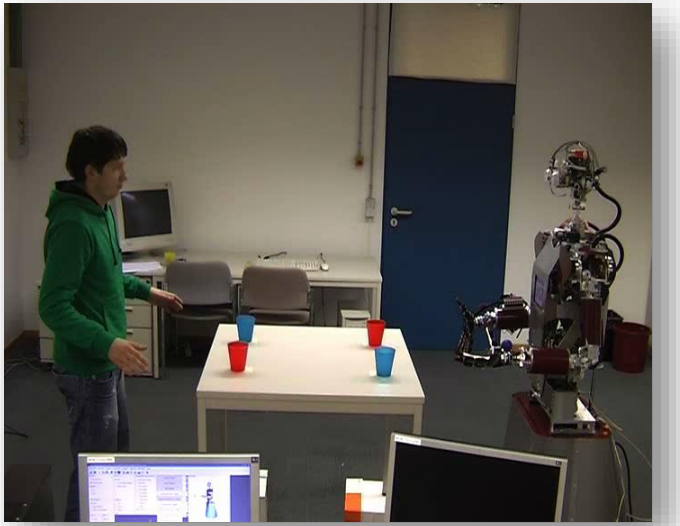
- Rekonstruktion der menschlichen Pose aus **RGB- und/oder Tiefen-Daten**
- **Vorteile**:
  - Sehr günstig, geringer Hardware-Aufwand
  - Geringe Anforderungen an Aufnahme-Umgebung
- **Nachteile**:
  - Komplexe Algorithmen und hohe Fehlerrate (aktives Forschungsgebiet)
  - Niedrige zeitliche Auflösung, vor allem bei Online-Einsatz



Azad et al. (2008)

Marker-lose optische Bewegungserfassung

- Stereovision



Azad et al. (2008)

Marker-lose optische Bewegungserfassung

- Skeleton Tracking mit Tiefenkameras
- Skelettdaten werden aus jedem Einzelbild berechnet
- Genauigkeit zwischen RGB-Tracking und markerbasiert Tracking

Kinect (RGB-D)

Captured depth image with background removal

Inferred body parts per pixel

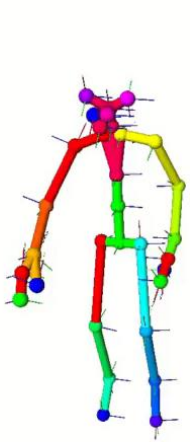
Estimated body joint positions

Fitted skeleton model

J. Shotton et al., *Real-Time Human Pose Recognition in Parts from a Single Depth Image, CVPR, 2011*

Deep Learning basierte Ansätze

- Azure Kinect body tracking (in real-time) https://learn.microsoft.com/en-us/azure/kinect-dk/body-sdk-download
  - Deep Neural Network for human pose estimation
  - Nonlinear optimization for motion retargeting to MMM model



MMM

RGB | Depth | Human pose | Motion retargeting

Motion Capture Studio at H2T

14 Vicon Motion Capture Cameras

3 Blue Trident IMU Sensors

Force Torque Sensors

Sensorized Exoskeleton

3 Azure Kinect RGB-D Cameras

3 Kinect V2 RGB-D Cameras

Mikrophone Array

GoPro Hero 8

Pair of CyberGlove II

Inhalt

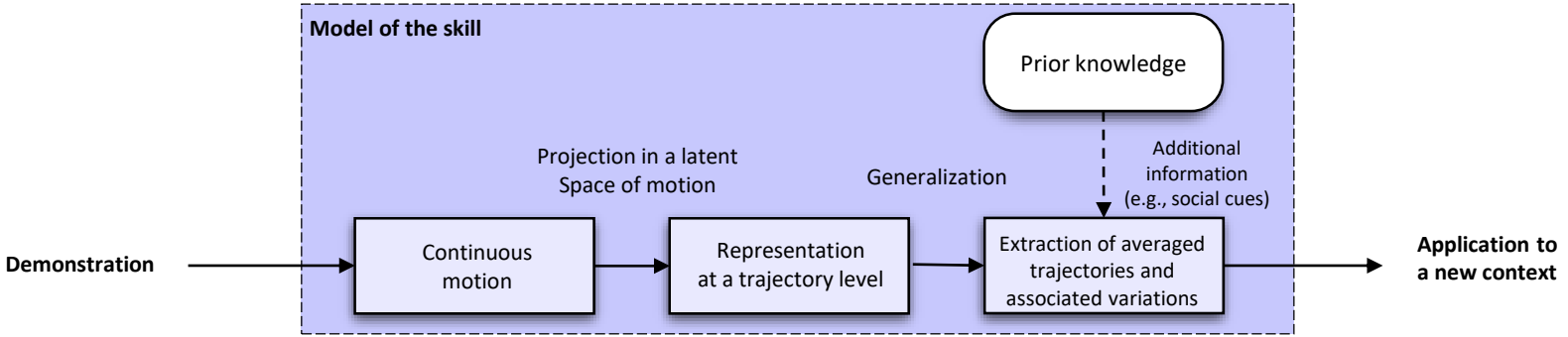
- Motivation
- Klassische Roboterprogrammierung (Übersicht)
- Graphische Programmierung (Statecharts)
- **Programmieren durch Vormachen**
  - Kernfragen von Programmieren durch Vormachen
  - Wahrnehmung menschlicher Demonstrationen
  - Lernen von Task-Modellen
  - Ausführung auf einem Roboter

Wichtige Fragen

- **Lernen auf zwei Ebenen**
  - **Subsymbolisch bzw. sensomotorisch** (Trajektorien-Ebene)
    → Lernen einer Fähigkeit (Skill)
  - **Symbolische bzw. semantische** (Symbolische Ebene)
    → Lernen von Task-Modellen

PdV – Lernen einer Fähigkeit

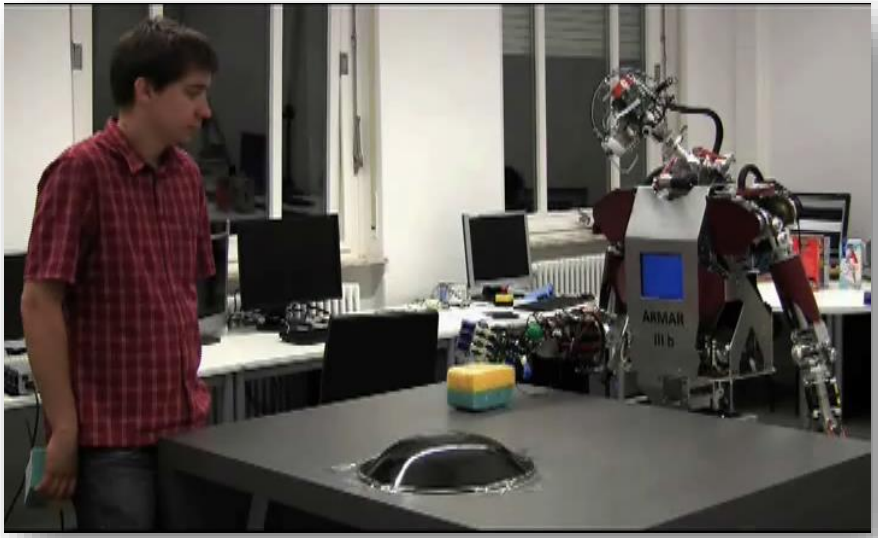
- **Subsymbolisch bzw. sensomotorisch (Trajektorien-Ebene)**
  Lernen einer nicht-lineare Abbildung zwischen Sensor- und Motorinformation. Diese stellt eine Repräsentation auf niedriger Ebene dar
- **Generalisierung auf Trajektorien-Ebene**

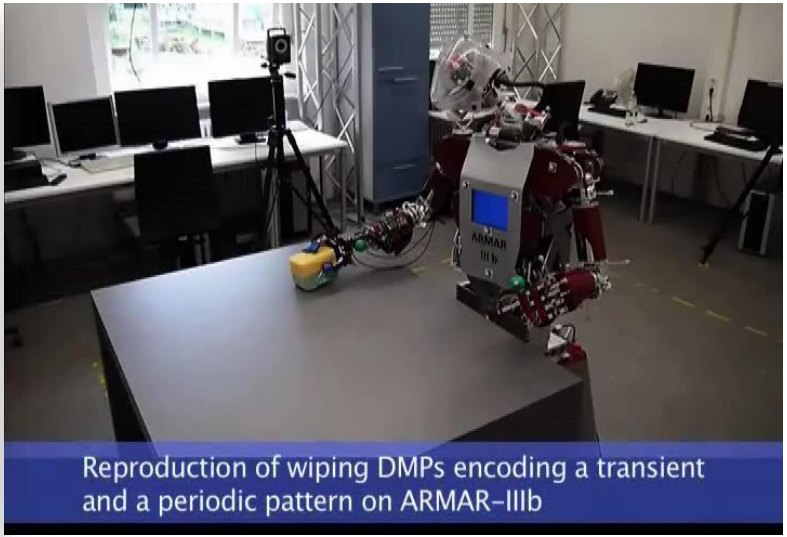


Billard, A., Calinon, S. and Dillmann, R. (2016). Learning From Humans. Siciliano, B. and Khatib, O. (eds.). Handbook of Robotics, 2nd Edition, Chapter 74, pp. 1995-2014. Springer

Beispiel: Trajektorien-Level

- Lernen einer Wischbewegung





Segmentierung der Demonstration

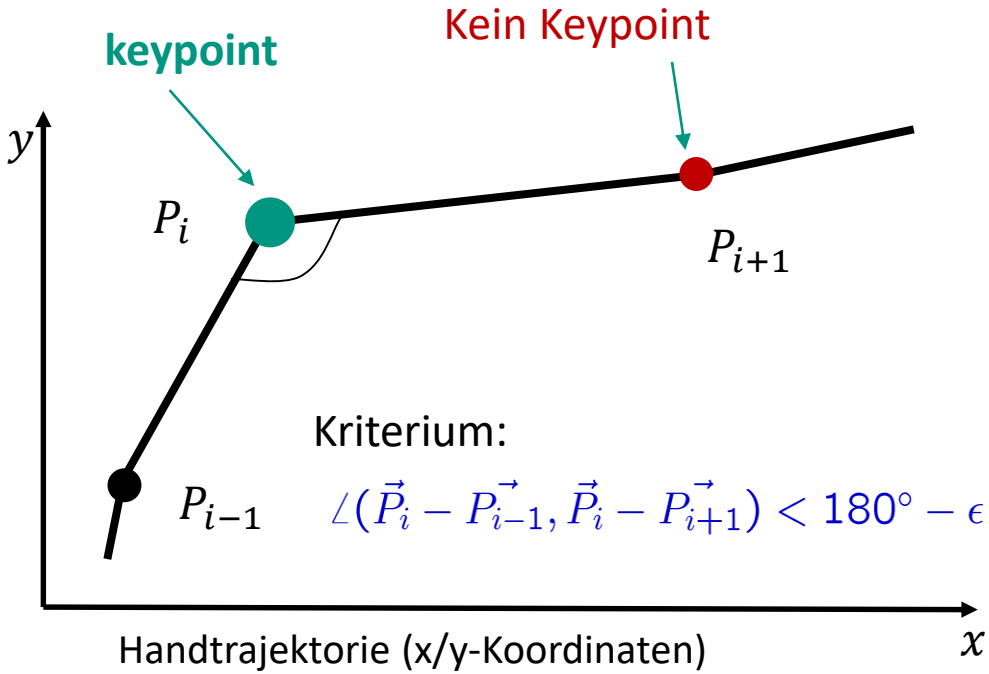
- **Bewegungssegmentierung:** Unterteilung der Bewegungstrajektorie(n) in mathematische einfach darstellbare Teile

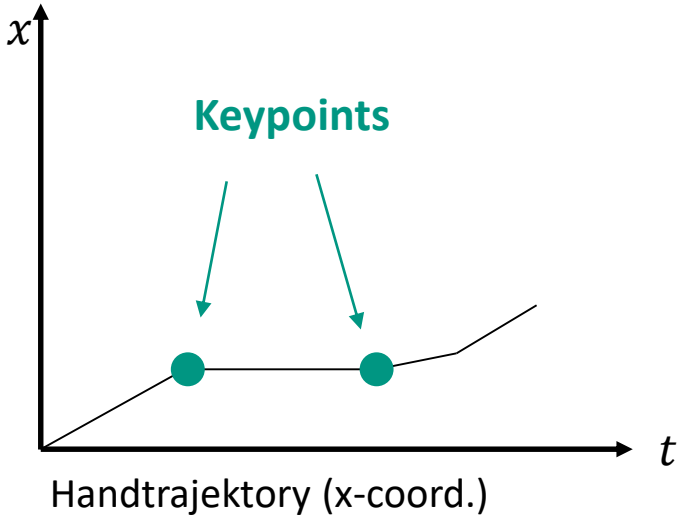
- Hierzu müssen **Schlüsselpunkten (key points)** der Demonstration bestimmt werden

- Kriterien zur Segmentierung notwendig
  - Im Konfigurationsraum und im Aufgabenraum
  - Unterschiedliche Kriterien: Änderung der Bewegung, Suchen nach bestimmten Mustern, ..

Kriterien zur Segmentierung

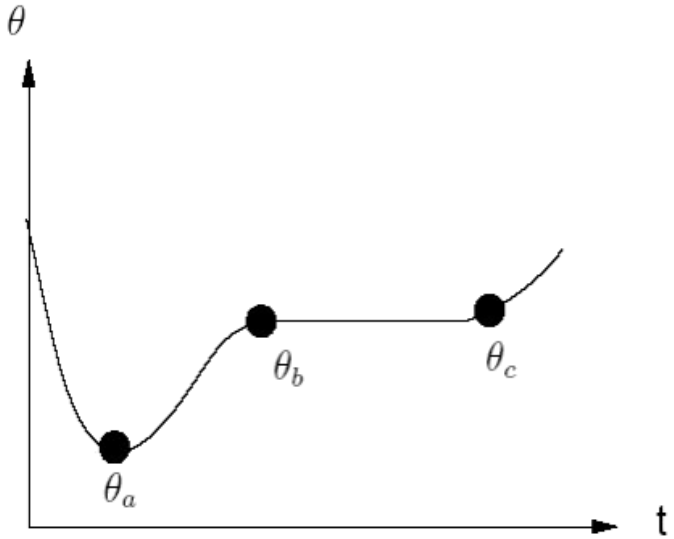
- Lokale Minima und Maxima sowie Pausen im Aufgabenraum





Kriterien zur Segmentierung

- Lokale Minima und Maxima sowie Pausen im Konfigurationsraum

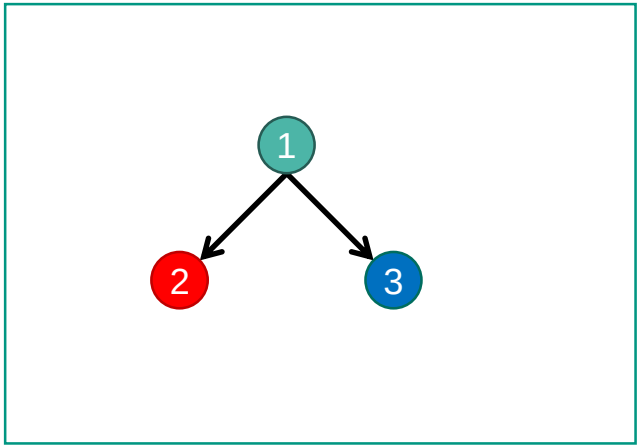


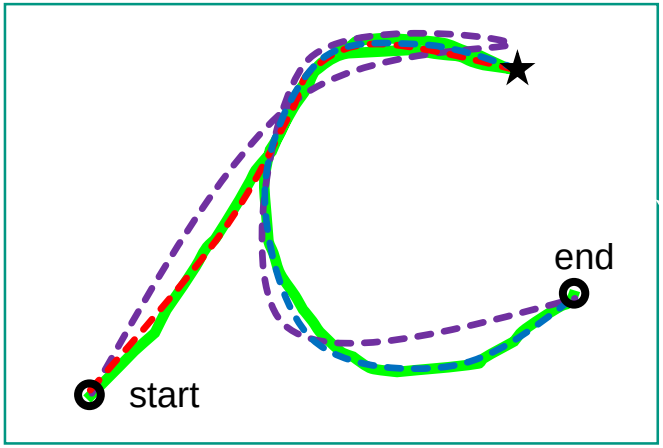
Joint trajectory

Kriterien zur Segmentierung

- Minimierung des Approximationsfehlers zwischen der Demonstration und einer bekannten „Grundfunktion“

$$error(y, \hat{y}) = \sum_{t=1}^{T} |y(t) - \hat{y}(t)|^2$$





$$\arg\min_t \sqrt{\sum_{i=1}^{n} \dot{y}_i(t)^2}$$

$$\arg\min_t \sqrt{\sum_{i=1}^{n} \ddot{y}_i(t)^2}$$

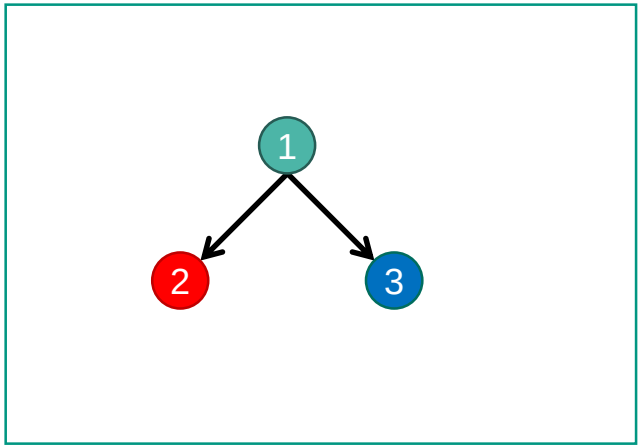
$$\arg\max_t \sum_{i=1}^{n} |y_i(t) - \hat{y}_i(t)|$$

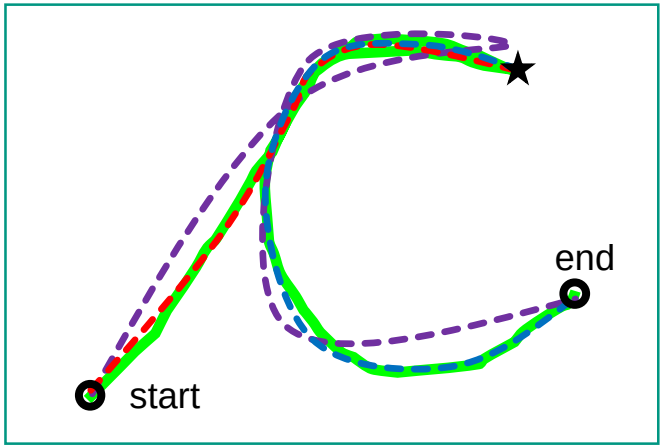
$$\vdots$$

Kriterien zur Segmentierung

- Minimierung des Approximationsfehlers

$$error(y, \hat{y}) = \sum_{t=1}^{T} |y(t) - \hat{y}(t)|^2$$





$$\arg\min_t \sqrt{\sum_{i=1}^{n} \dot{y}_i(t)^2}$$

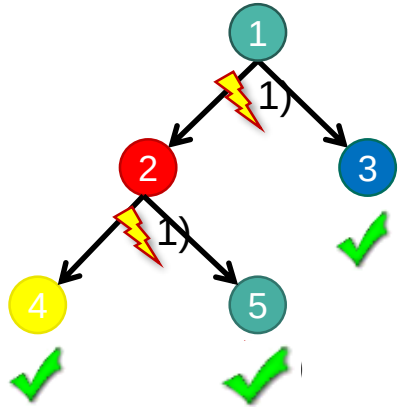
$$\arg\min_t \sqrt{\sum_{i=1}^{n} \ddot{y}_i(t)^2}$$

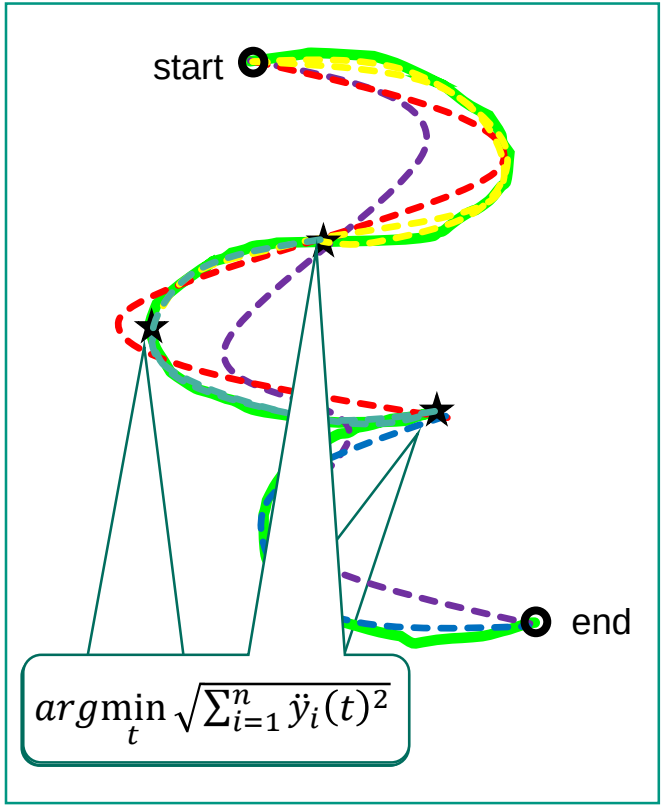
$$\arg\max_t \sum_{i=1}^{n} |y_i(t) - \hat{y}_i(t)|$$

$$\vdots$$

Kriterien zur Segmentierung

1) Approximationsfehler
2) Dauer der Bewegung



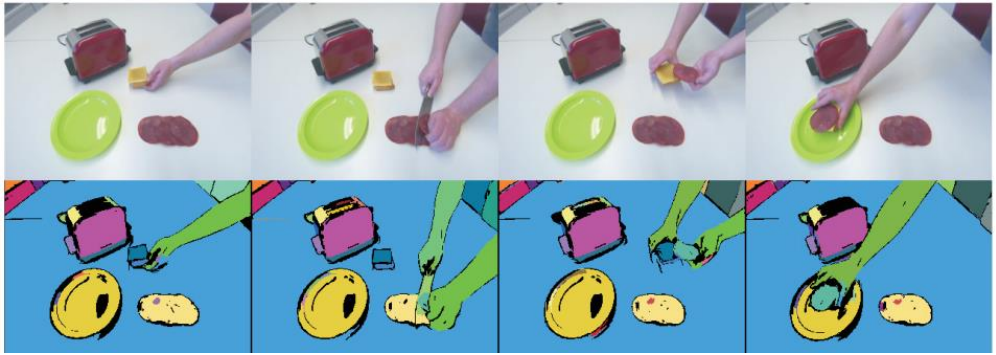


Beispiel: Lernen einer Trajektorie aus RGB-D Daten

- Tracking von Lehrer und Objekten durch Farb- und Tiefenregionen
- Optischer Fluss (Bewegungsmuster) wird benutzt, um Bewegungen zwischen 2 Frames zu tracken
- Segmentmittelpunkte pro Frame stellen die Trajektorie dar

Kamera Bilder

Getrackte Bildregionen



A. Abramov, E.E. Aksoy, J. Dörr, K. Pauwels, F. Wörgötter, *Real-Time Human Pose Recognition in Parts from a Single Depth Image, 3DPVT, 2010*

PdV – Lernen einer Fähigkeit

- Lernen von Aktionen und Repräsentation geschieht oft zusammen
  - Repräsentation ist stark an der Lernverfahren gekoppelt
- Algorithmen zur Bewegungssegmentierung: HMM, PCA, Clustering, Template Matching, Klassifikatoren
- Methoden zum Lernen einer Fähigkeit
  - Hidden Markov Models (HMM)
  - Dynamic Movement Primitives (DMP)
  - Gaussian Mixture Models (GMM)
- Verfeinerung von gelernten Trajektorien
  - Reinforcement Learning
- **Mehr dazu in der Vorlesung „Robotik-II: Humanoide Robotik“ im SS**

PdV – Lernen einer Fähigkeit

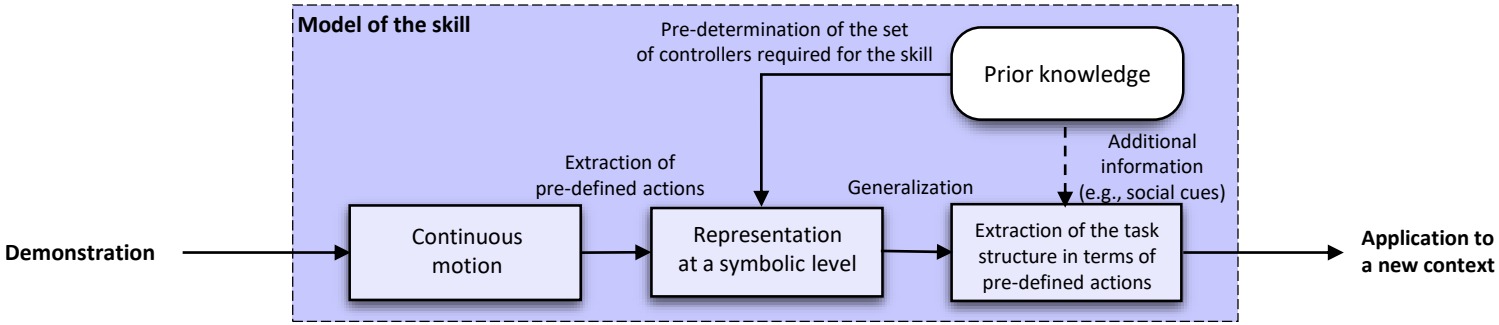
- **Subsymbolisch:**
  - Segmentierung von Bewegungstrajektorie, d.h. die Identifikation von Schlüsselpunkten (Key Points) der Demonstration
  - Beschreibung der resultierenden Segmente in einer generalisierten Form (Funktionsapproximation)

- **Vorteil:** Effizientes Lernen einer Bewegung

- **Nachteil:** Keine Berücksichtigung von semantischer Information, z.B. Vor- und Nachbedingungen

PdV – Lernen von Task-Modellen

- **Semantischer bzw. symbolisch:**
  Lernen einer Sequenz von Aktionen, die die Demonstration repräsentiert. Diese stellt eine Repräsentation auf höherer Ebene dar.
- **Generalisierung auf symbolischem (Task) Level**



Billard, A., Calinon, S. and Dillmann, R. (2016). Learning From Humans. Siciliano, B. and Khatib, O. (eds.). Handbook of Robotics, 2nd Edition, Chapter 74, pp. 1995-2014. Springer

Beispiel: Task-Level

- Lernen einer Sequenz von Aktionen, d.h. eines Plans (Teig vorbereiten)

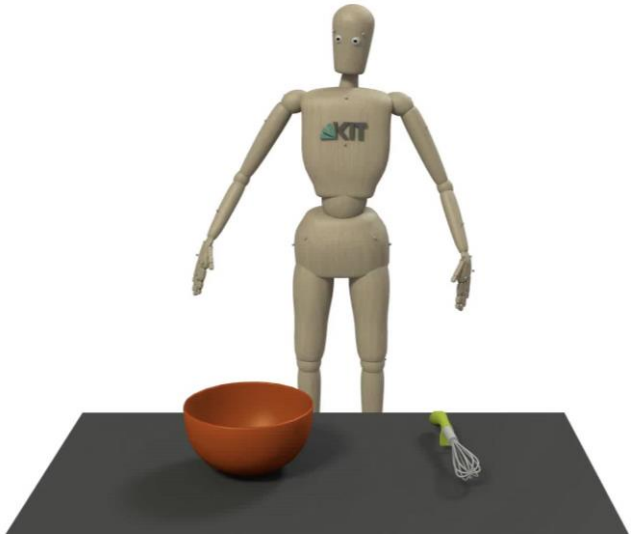
PdV – Lernen von Task-Modellen - Vorgehensweise

- Beobachtung des Lehrers und der Umgebung
- Extraktion von semantischen Merkmalen: Zustände, Objekt-Relationen, Object-Hand-Relation, Regeln
- Der Roboter muss eine geeignete Aktion auf Basis des **aktuellen Weltzustands** bestimmen oder abschätzen
- Der Weltzustand muss aus **Beobachtungen des Roboters** bestimmt werden
- **Ziel:** Eine **Zuordnung** zwischen Weltzustand und Aktion → Strategie zur Aktionswahl

Aufgabensegmentierung „Task Segmentation“

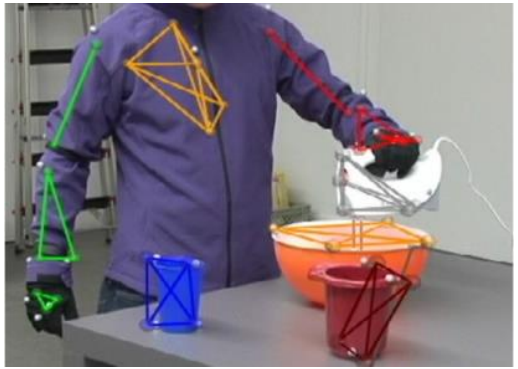
- Demonstrationen bestehen meist aus Aktionssequenzen
- **Verständnis** von Demonstration **einfacher**, wenn diese in **kleinere Segmente** oder sogar **bekannte Aktionen** zerlegt sind

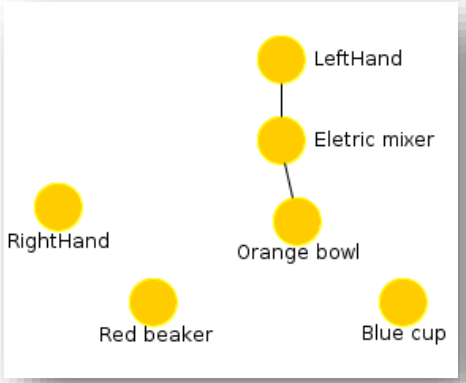
- Aber:
  - Welche Aktionen kommen vor und wie viele Aktionen sind unbekannt?
  - Start und Ende von Aktionen ist unbekannt
  - Aktionen gehen nahtlos ineinander



Aufgabensegmentierung „Task Segmentation“

- **Kontaktrelationen zwischen Objekten** zur Repräsentation des Weltzustandes
- **Keyframes** bei **Änderungen** der Kontaktrelationen
- Jedes Segment repräsentiert eine **Manipulationsaktion**,
  - Teig umrühren enthält z.B. greifen, rühren, abstellen, …





Aufgabensegmentierung „Task Segmentation“

- **Objektrelationen** ermöglichen die Extraktion von Vorbedingungen und Effekten (Nachbedingungen) von Aktionen für **symbolische Planer**

- **Beispiel:**
  - Zustand am Segmentanfang: `LeftHand berührt nichts`
  - Zustand am Segmentende: `LeftHand berührt RedCup`

  -> *Vorbedingung:* `empty(LeftHand)`

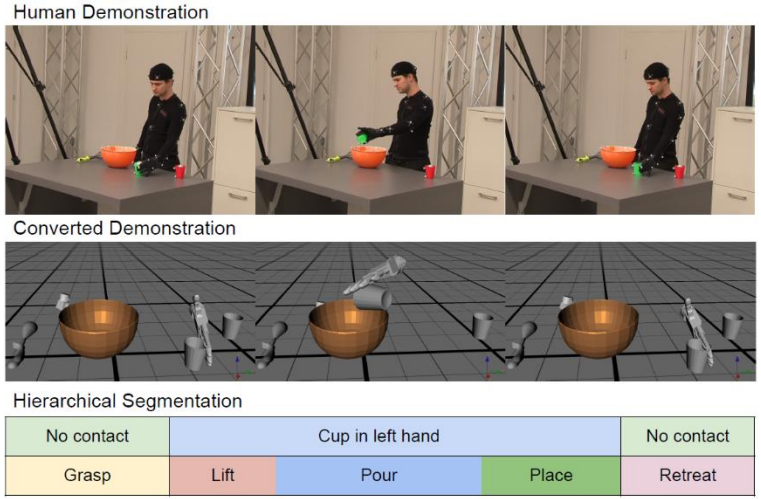
  -> *Nachbedingung:* `in(RedCup,LeftHand), !empty(LeftHand)`

Hierarchische Aufgabensegmentierung

- **Hierarchische Aufgabensegmentierung**
  unter Berücksichtigung von Bewegung und relevanten Objekten

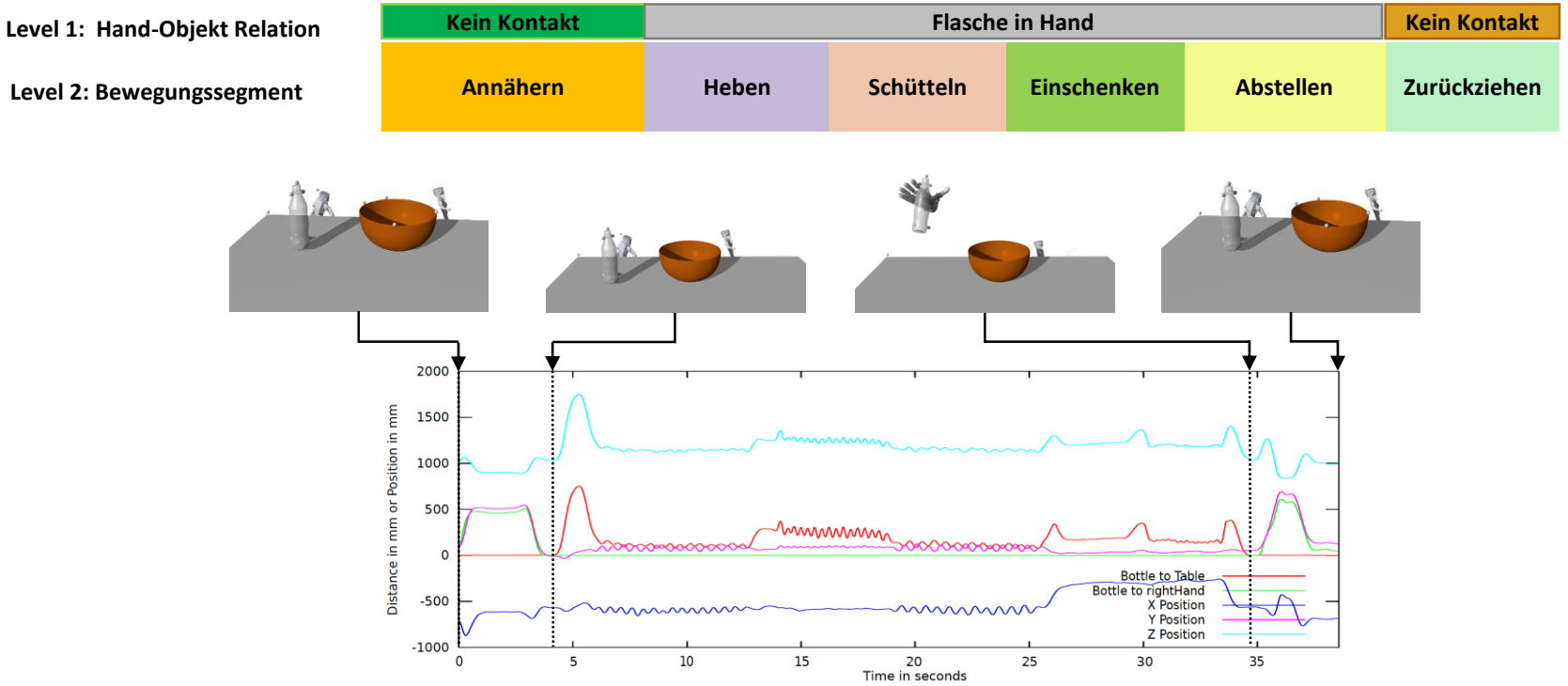
  Level 1: Semantische Segmentierung
  basierend auf **Kontaktrelationen zwischen Objekten**

  Level 2: Bewegungssegmentierung
  basierend auf **Charakteristiken** von Trajektorien (Bewegungsdynamik)



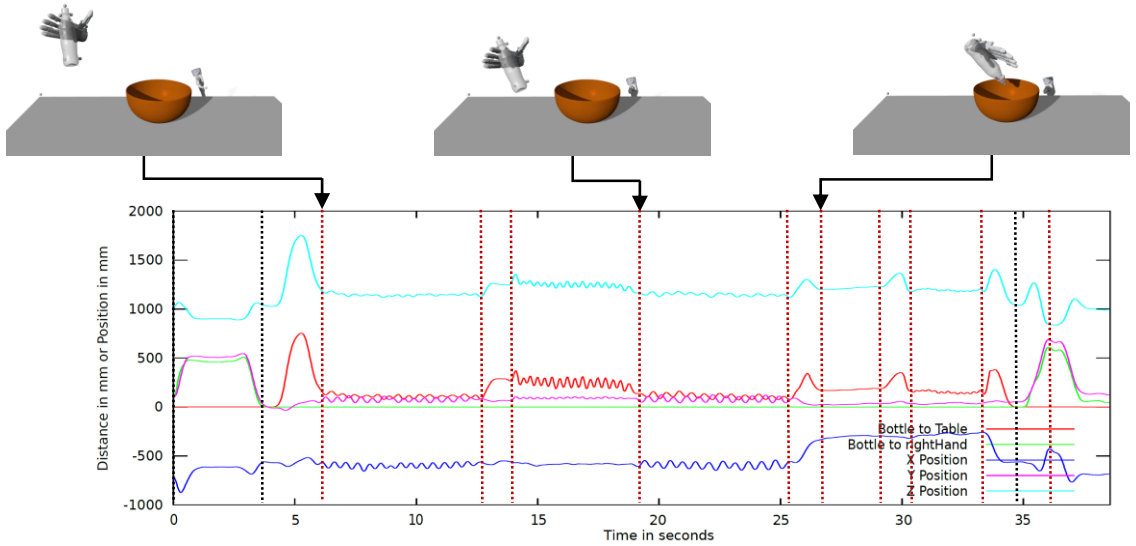
M. Wächter and T. Asfour, Hierarchical Segmentation of Manipulation Actions based on Object Relations and Motion Characteristics, International Conference on Advanced Robotics (ICAR), July, 2015

Level 1: Semantische Segmentierung



Level 2: Bewegungssegmentierung

Level 1: Hand-Objekt Relation	**Kein Kontakt**	**Flasche in Hand**				**Kein Kontakt**
Level 2: Bewegungssegment	**Annähern**	**Heben**	**Schütteln**	**Einschenken**	**Abstellen**	**Zurückziehen**

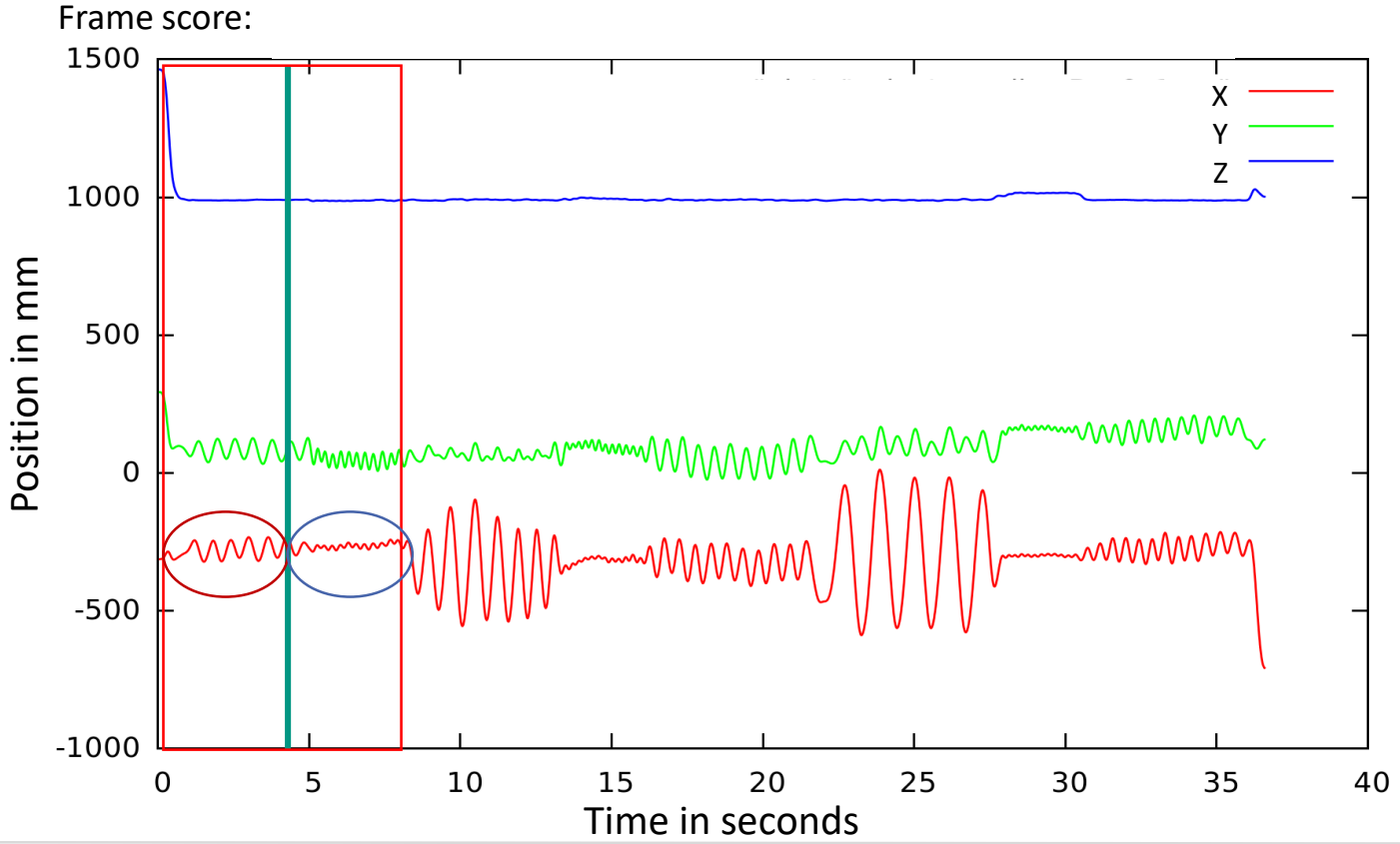


Karlsruher Institut für Technologie

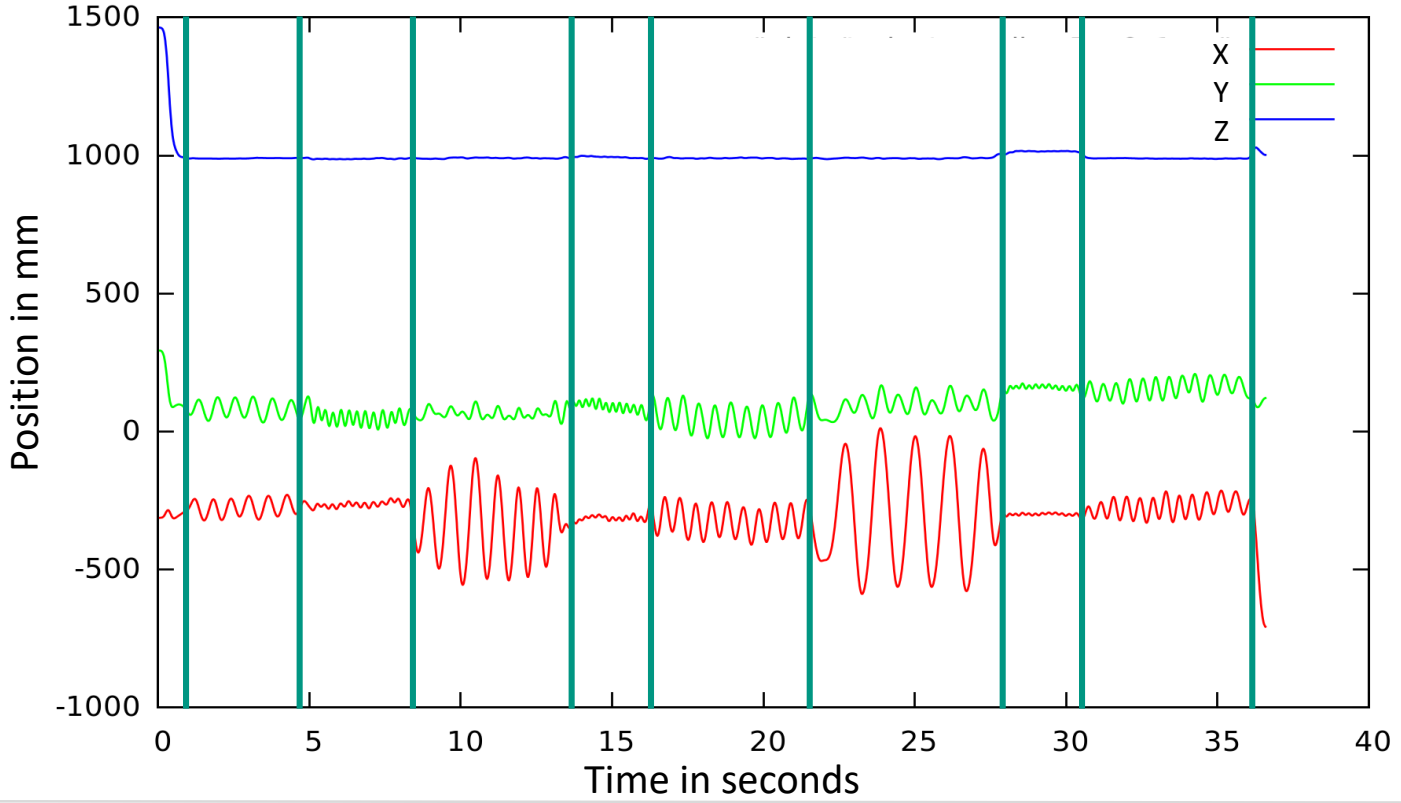
Stoff nur bisher relevant für die Klausur

H2T

Bewegungscharakteristik Heuristik: Gleitendes Fenster



Bewegungscharakteristik Heuristik: Rekursive Suche



Bewegungscharakteristik Heuristik

- Heuristik basierend auf dem Beschleunigungsprofil
  - Messen der Länge der Beschleunigungskurve

$$s_{l,d}(t_c) = \int_{t_c-\frac{w}{2}}^{t_c-1} \sqrt{1 + a'_d(t)^2}\,dt \left(\frac{\hat{\mathrm{U}}_l}{\hat{\mathrm{U}}_r}\right)^2$$

$$s_{r,d}(t_c) = \int_{t_c}^{t_c+\frac{w}{2}-1} \sqrt{1 + a'_d(t)^2}\,dt \left(\frac{\hat{\mathrm{U}}_r}{\hat{\mathrm{U}}_l}\right)^2$$

- Iterative Suche des besten Keyframe Kandidaten mittels gleitendem Fenster
  - Vergleiche der Segmente links und rechts des Keyframe Kandidaten mit der Auswertfunktion *s*
  - Keyframe Qualität: $q_d = \begin{cases} \frac{s_{l,d}}{s_{r,d}} & s_{l,d} > s_{r,d} \\ \frac{s_{r,d}}{s_{l,d}} & s_{l,d} \leq s_{r,d} \end{cases}$

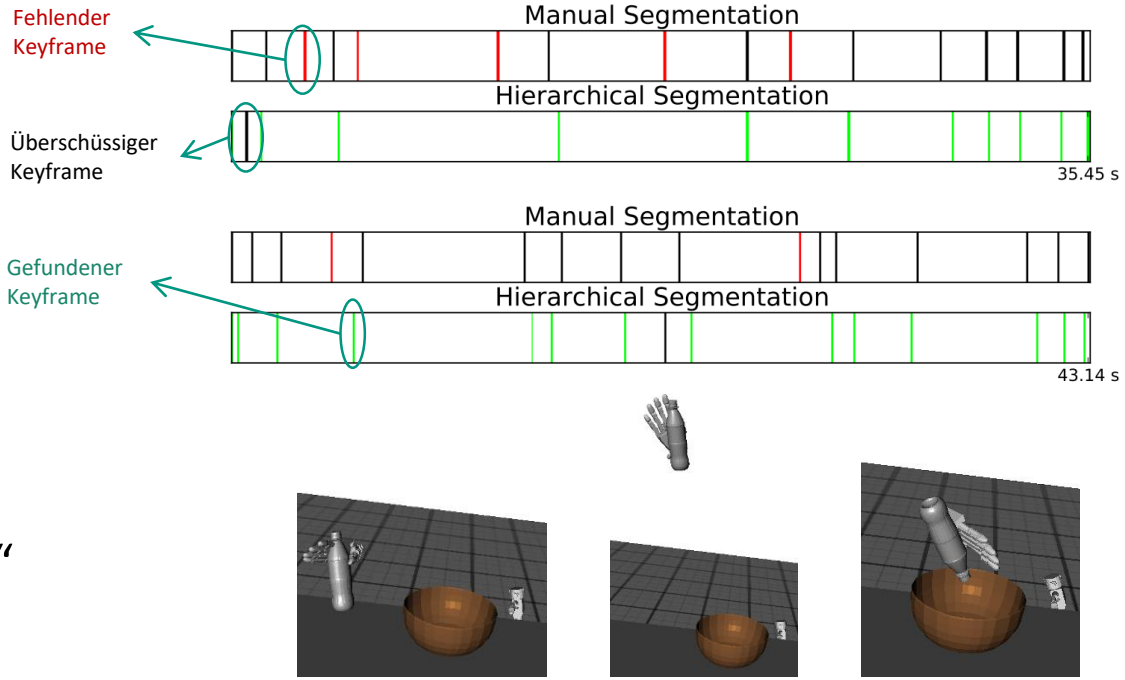
- Rekursive Unterteilung bis die Segmentgröße oder Qualität zu klein/niedrig ist

Evaluation der Segmentierung

- Metrik für die Segmentierungsergebnisse: **Bestrafung** für fehlende und zusätzliche Keyframes

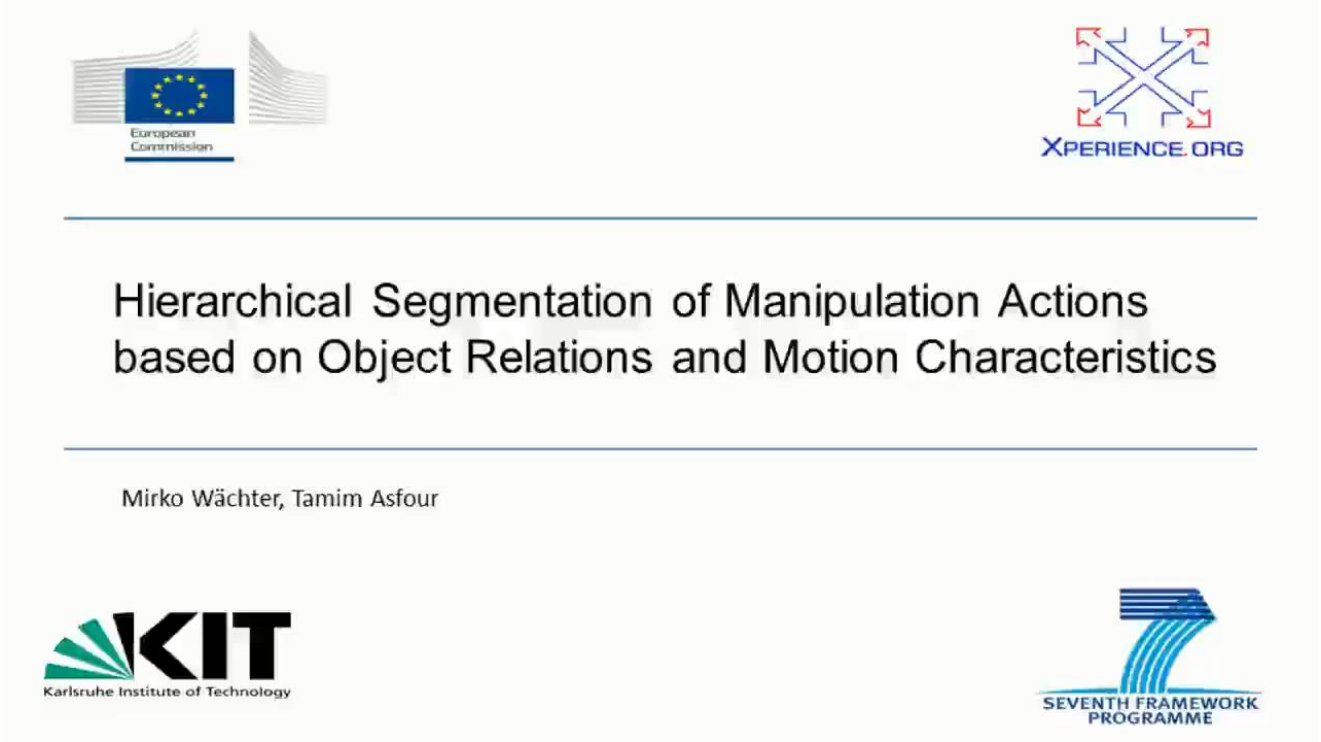
$$e = \underbrace{(m + f) * p}_{\text{Bestrafung für fehlende und zusätzliche Keyframes}} + \underbrace{\sum_i \min_j (k_{r,i} - k_{f,j})^2}_{\text{Mittlerer quadratischer Fehler zum nächsten Keyframe}}$$

Evaluation: Vergleich zu einer Referenzsegmentierung

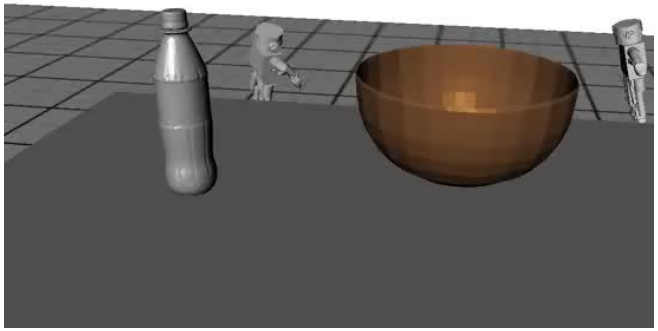


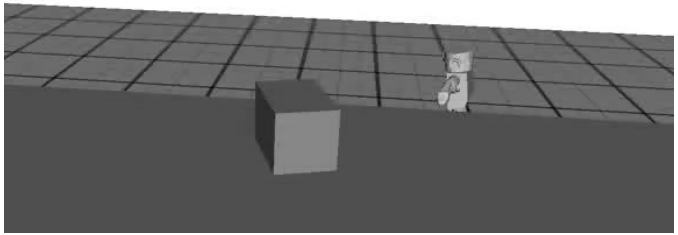
2 Wiederholungen von „Schütteln-und-Einschenken“

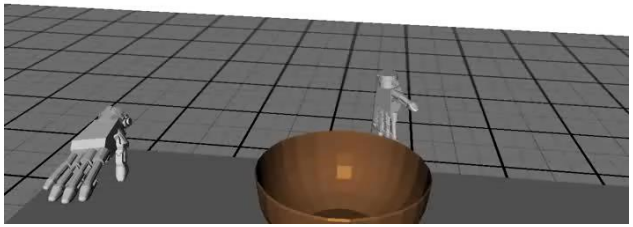
Beispiel: Bewegungssegmentierung

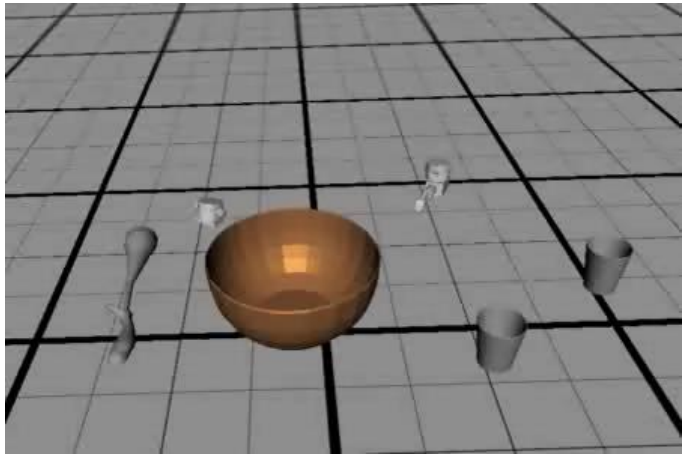


Verschiedene Aktionssequenzen



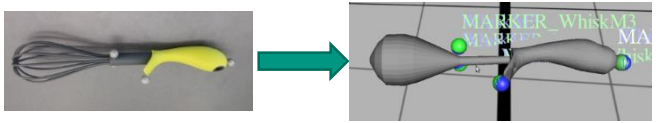






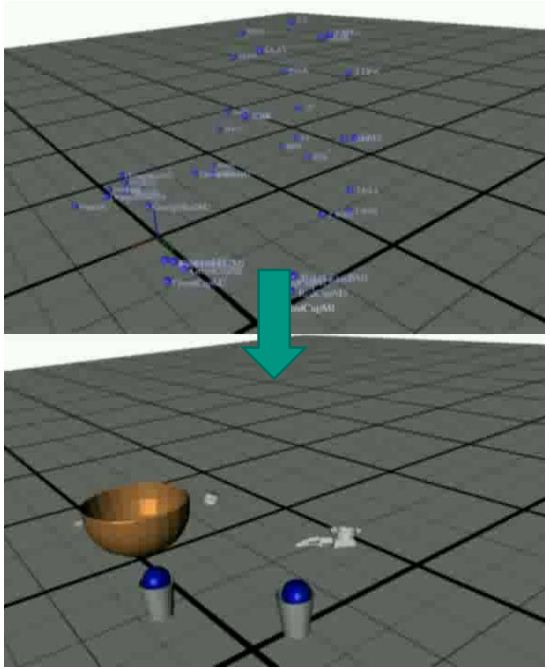
Marker-basierte Erfassung der Demonstration

- Marker-basierte Erfassung der Demonstration von
  - Mensch (Lehrer)
  - Objekten
  - Umgebung



- Transformation zu Trajektorien von 6D-Posen mit Hilfe von 3D Modelle mit virtuellen Markern

- Referenzmodell des menschlichen Körpers in der Master Motor Map (MMM) Datenformat[1]



[1] https://mmm.humanoids.kit.edu/

PdV – Lernen von Task-Modellen

- **Symbolisch:**
  - Ein üblicher Ansatz segmentiert und repräsentiert die Demonstration anhand von vordefinierter Aktionen bzw. deren Folgen, die symbolisch beschrieben sind.
  - Die Repräsentation und Reproduktion der Aktionsfolgen kann mit Methoden des maschinellen Lernens erfolgen.

- **Vorteil:** Komplexe Aufgaben können effizient über einen interaktiven Prozess gelernt werden

- **Nachteil:** Vorwissen wird benötigt, um die wichtigen „Aspekte“ in der Demonstration erkennen zu können

Zusammenfassung - Lernen auf zwei Ebenen

Repräsentation	Generalisierung	Vorteile	Nachteile
Trajektorie	Generalisierung von Bewegungen	Allgemeine Repräsentation von Bewegungen, die es erlaubt verschiedenste Arten von Signalen/Funktionen zu kodieren	Kann komplexe Fertigkeiten nicht reproduzieren
Symbolisch	Organisation von vordefinierten Bewegungselementen	Ermöglicht das Lernen von hierarchischen Aufgaben-beschreibungen	Benötigt eine vordefinierte Menge von Controllern zur Reproduktion

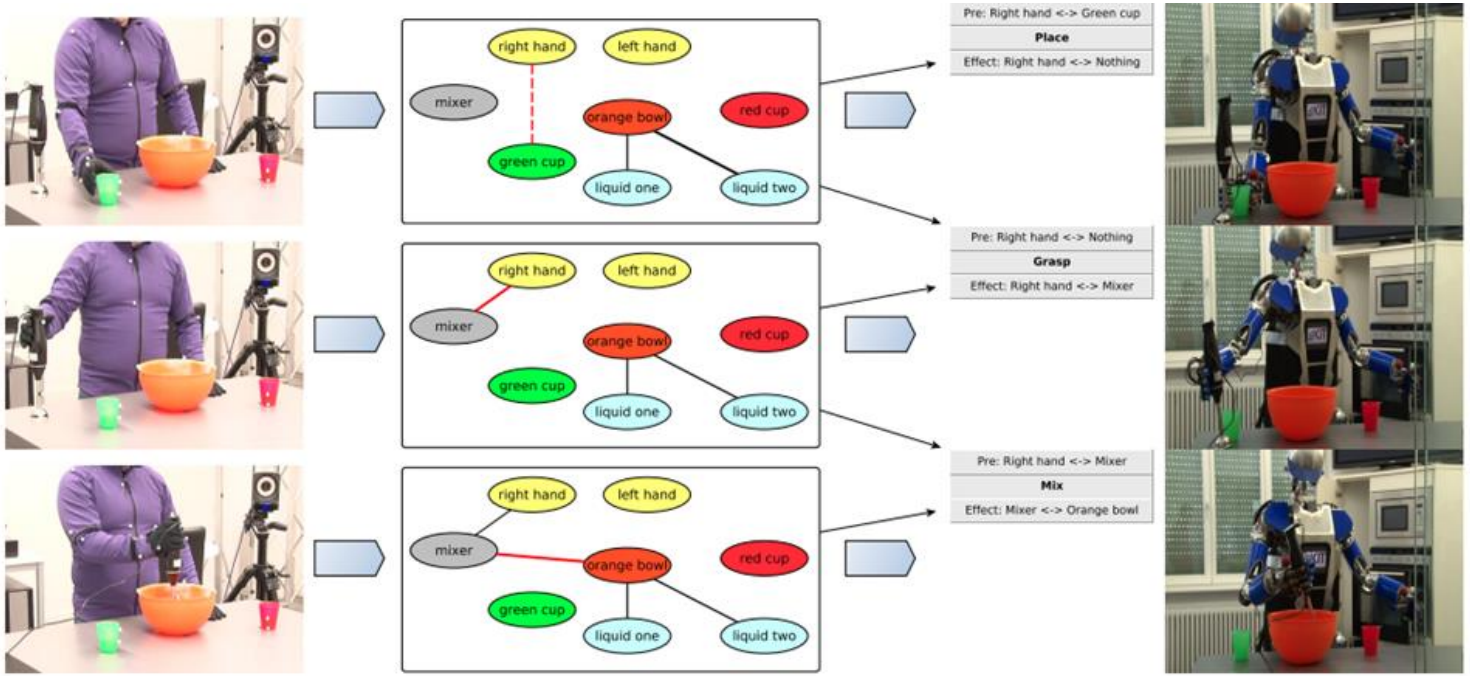
Inhalt

- Motivation
- Klassische Roboterprogrammierung (Übersicht)
- Graphische Programmierung (Statecharts)
- **Programmieren durch Vormachen**
  - Kernfragen von Programmieren durch Vormachen
  - Wahrnehmung menschlicher Demonstrationen
  - Lernen von Task-Modellen
  - Ausführung auf einem Roboter

Ausführung auf einem Roboter

- Abbildung des gelernten Wissens (Aktionen, Aktionssequenzen, …) auf den Roboter
- **Lösung des Korrespondenzproblems**: Mensch und Roboter haben unterschiedliche Kinematik und Dynamik
- Resultierende Bewegungen als Sollvorgaben für die Regelung
- Ausführung mit online Adaption and dynamischen Umgebungen

Beispiel – Teig vorbereiten



dotted lines for fading touching relations

Beispiel – Teig vorbereiten

Zusammenfassung

- PdV ermöglicht es, Fähigkeiten vom Menschen zu lernen und auf beliebige Roboter zu transferieren
- PdV kann den Lernprozess beschleunigen (im Vergleich zu Reinforcement Learning bzw. trial-and-error Methoden)
- Der Roboter muss über eine Bibliothek an generischen Fähigkeiten (skills) verfügen und in der Lage sein diese an den Kontext anzupassen

Forschungsfragen

- **Art des Lernens**
  - **Batch Lernen:** Die Aktion wird gelernt wenn alle Demonstrationen/Beispiele aufgenommen sind
  - **Inkrementelles Lernen:** Die Aktionsrepräsentation wird nach jeder Demonstration gelernt/aktualisiert
- **Anzahl und Art der Demonstrationen/Beispielen**
  - Lernen aus **vielen** Demonstrationen
  - Lernen aus **wenigen** Demonstrationen (aus einer **einzigen** Demonstration)
  - Lernen aus **positiven** und **negativen** Demonstrationen
- **Interaktion mit dem Menschen während des Lernvorgangs**
  - **Natürliche Sprache:** Sprachliche Anweisungen (hier, dort, jetzt, schneller, ..), Klärungsdialoge zur Vervollständigung von Wissen über die Aufgabe, Feedback, …
  - Blickrichtung und Zeige-Gesten des Lehrers
  - Einbindung weiterer Modalitäten wie Haptik, Audio, ….

Mehr zum Thema Programmieren durch Vormachen in der **Vorlesung „Robotik-II: Humanoide Robotik“** im SS