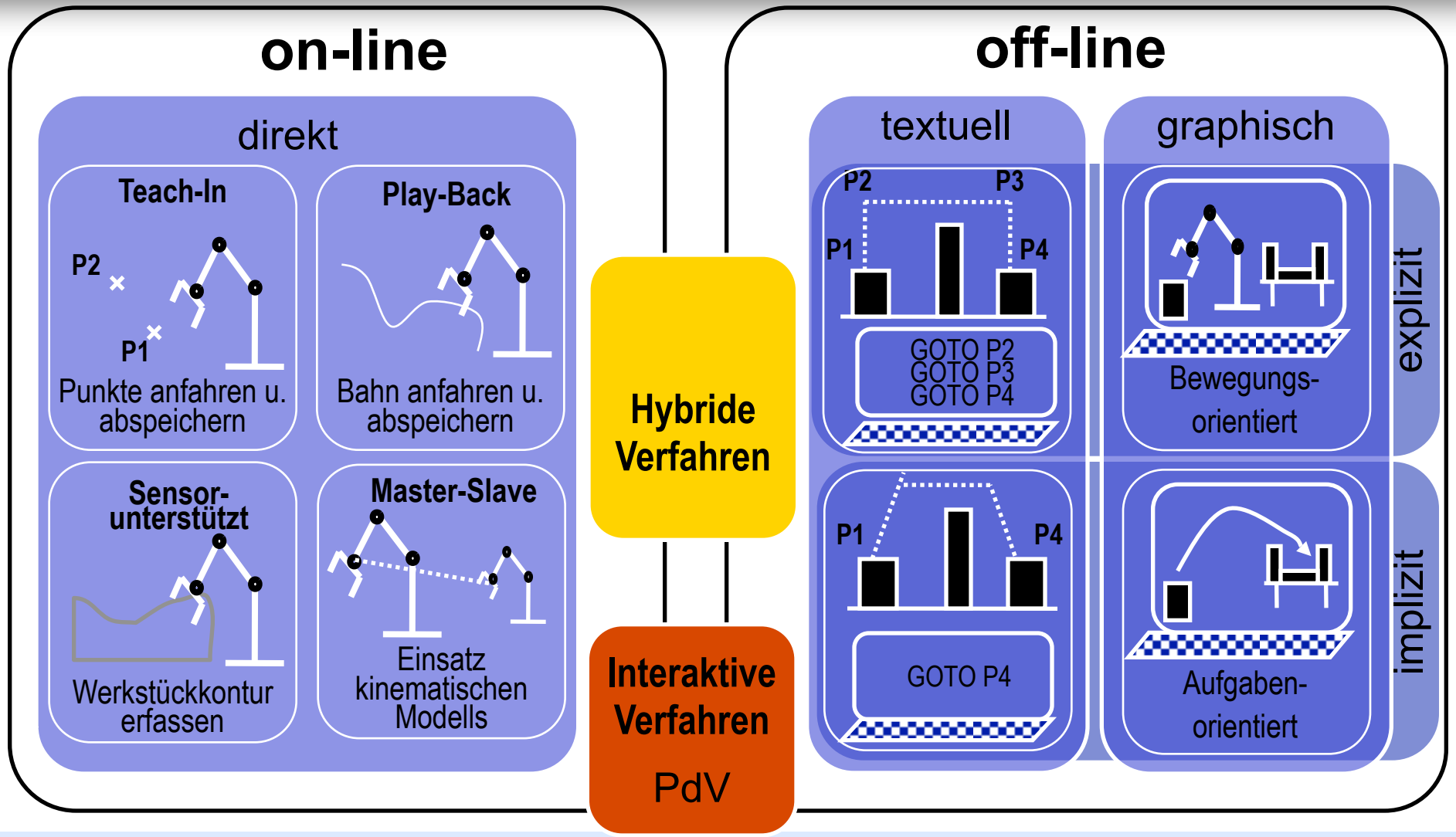
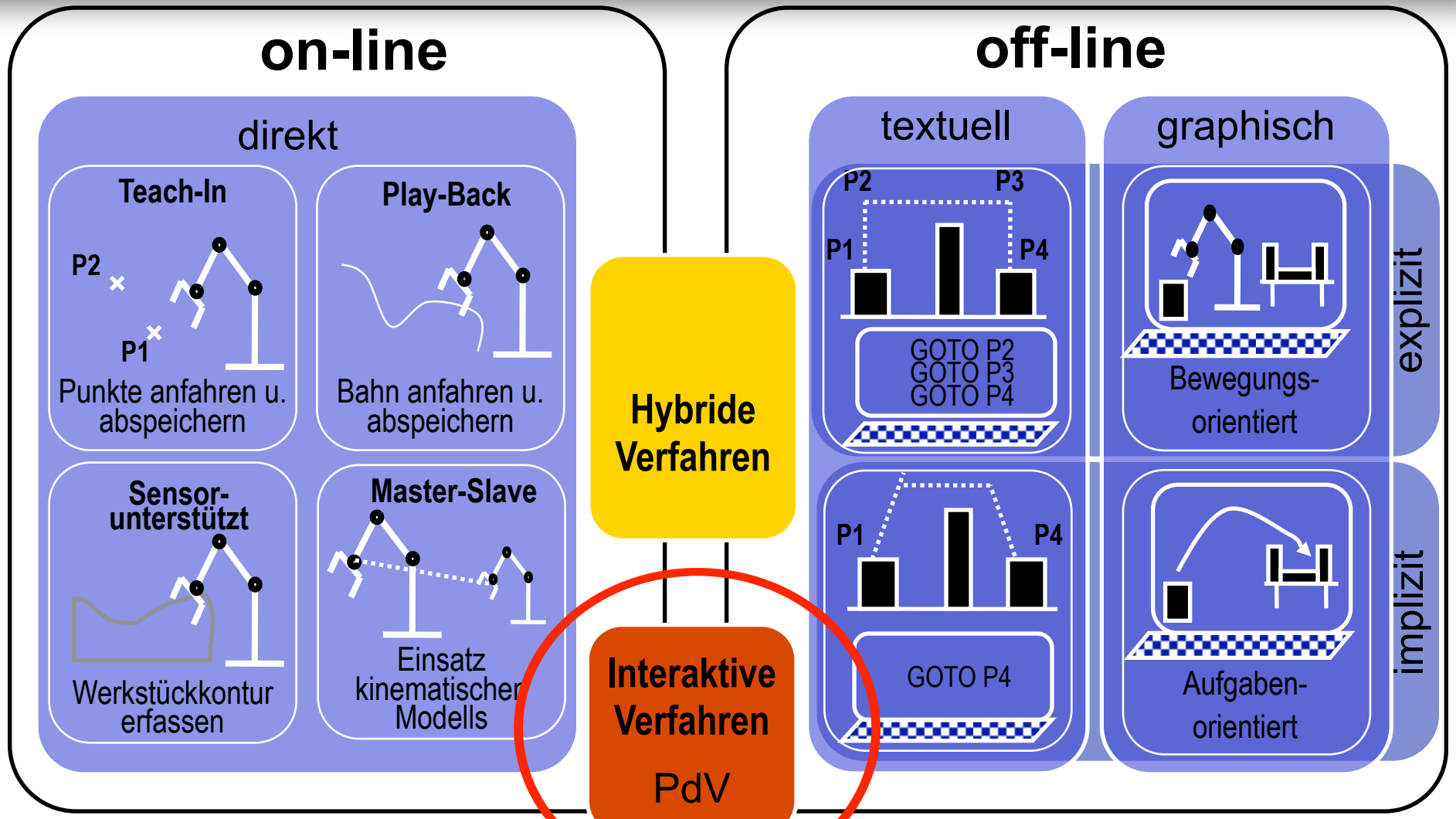
IV
Interaktive Programmierung von Robotern II

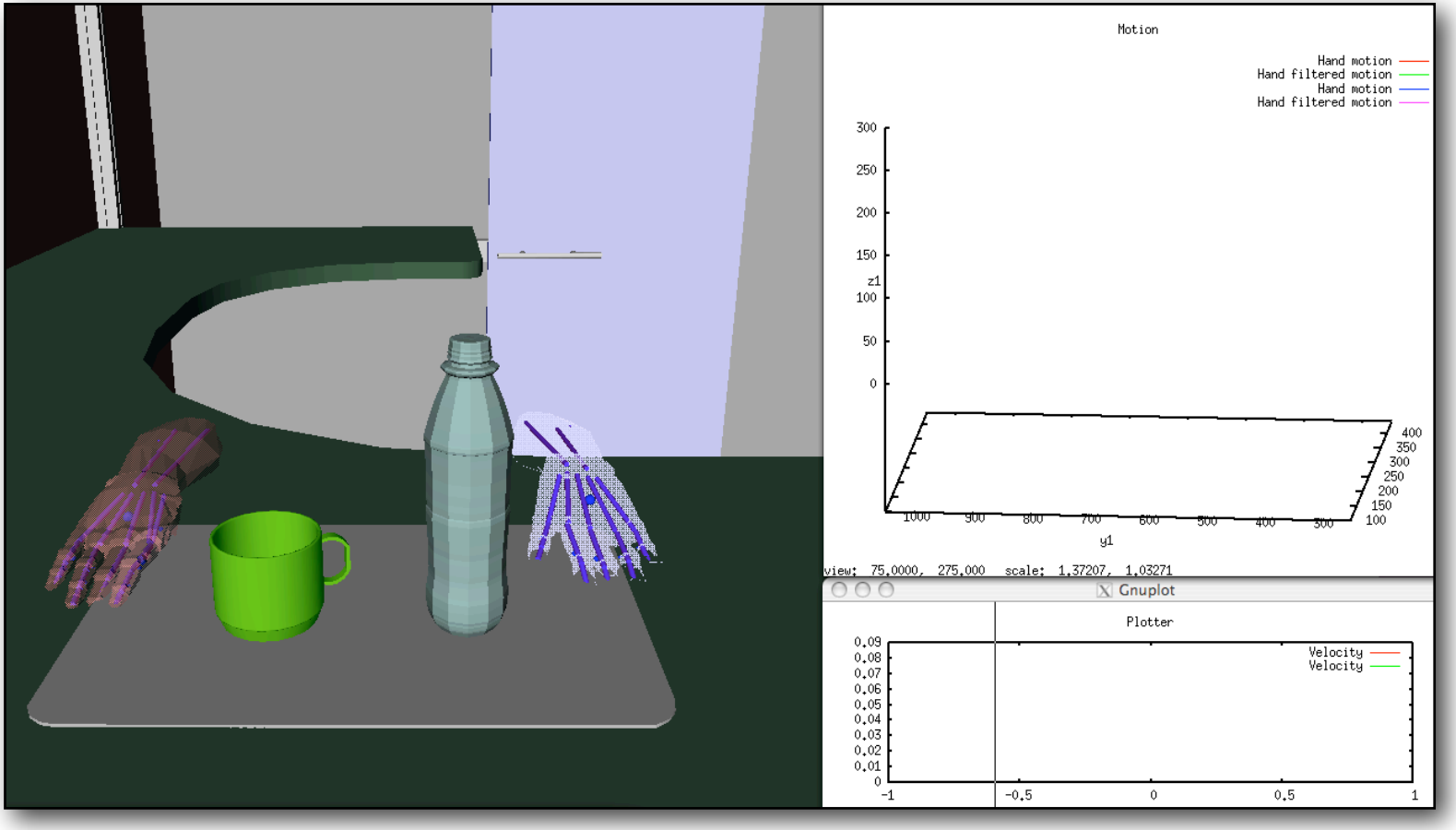
Roboterprogrammierverfahren



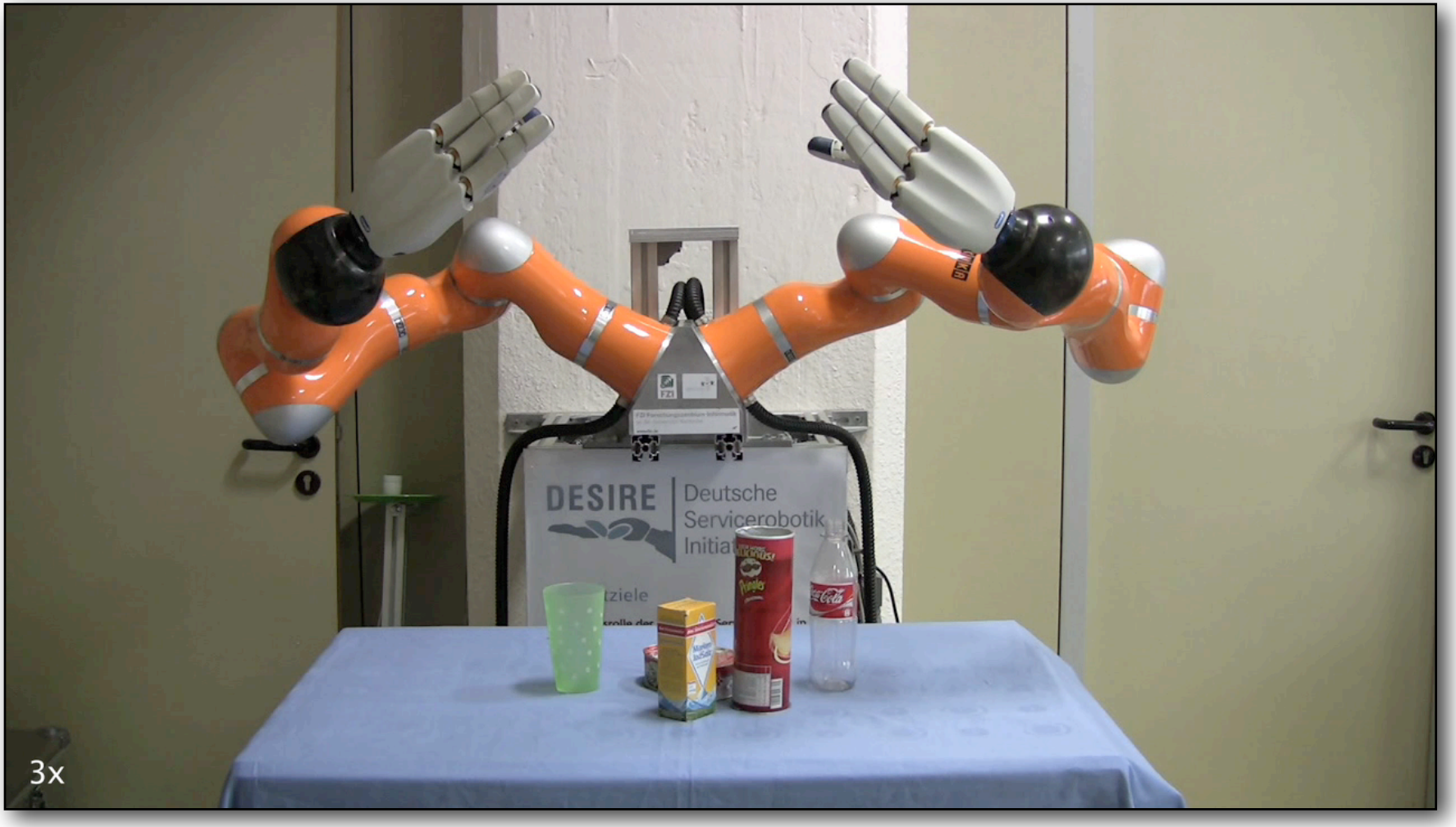
Roboterprogrammierverfahren



Rückblick: Lernen aus Beobachtung des Menschen



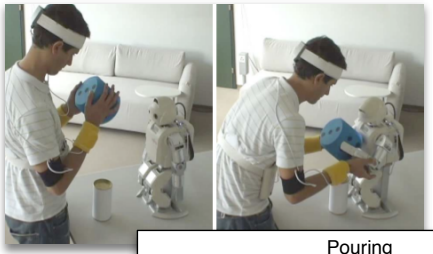
Rückblick: Ausführung



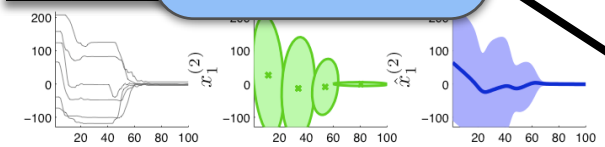
Rückblick: Komponenten

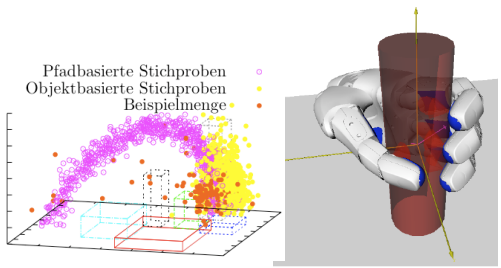
Benutzer

Sensoren und Interaktionsformen



Lernsystem





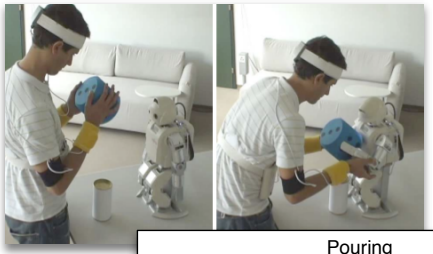
Transformation

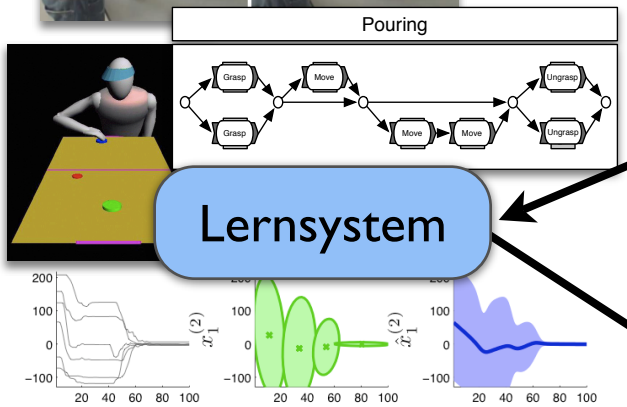
Ausführung

Rückblick: Komponenten

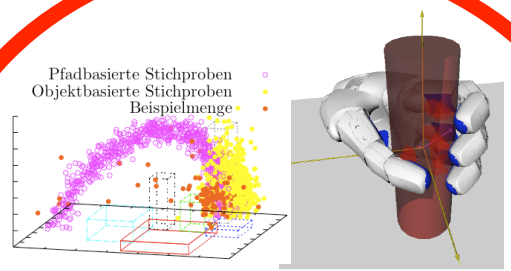
Benutzer

Sensoren und Interaktionsformen





Lernsystem



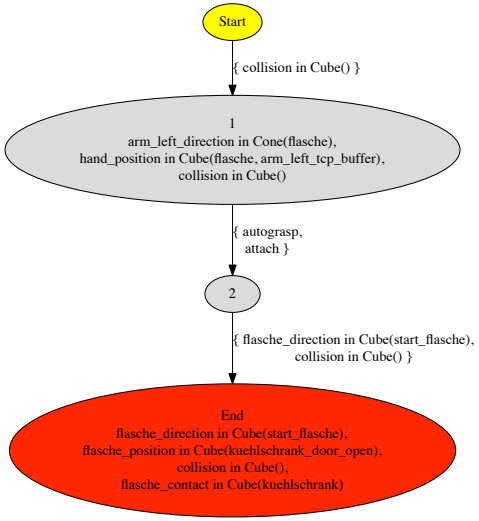
Transformation

Ausführung

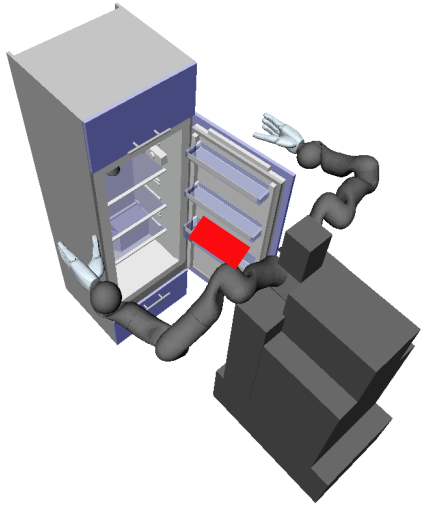
Rückblick: Repräsentation des Planungsproblems

„Strategiegraph“:

- Tupel ($X, C^t_n, C^t_e, C^b_n, C^b_e$)
- Knoten X, zeitliche Einschränkungen der Knoten C^t_n und Kanten C^t_e, Bewegungseinschränkungen der Knoten C^b_n und Kanten C^b_e



Strategiegraph



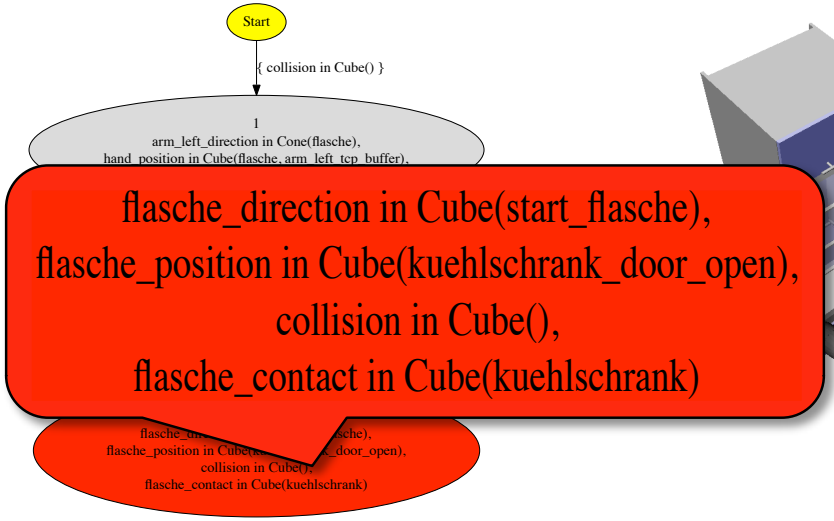
Bewegungseinschränkungen (rot)

Planung / Ausführung

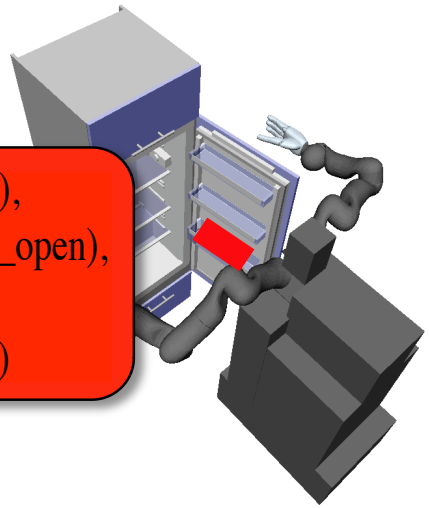
Rückblick: Repräsentation des Planungsproblems

„Strategiegraph“:

- Tupel (X, C^t_n, C^t_e, C^b_n, C^b_e)
- Knoten X, zeitliche Einschränkungen der Knoten C^t_n und Kanten C^t_e, Bewegungseinschränkungen der Knoten C^b_n und Kanten C^b_e



Strategiegraph



Bewegungseinschränkungen (rot)

Planung / Ausführung

Rückblick: Repräsentation des Planungsproblems

„Strategiegraph“:

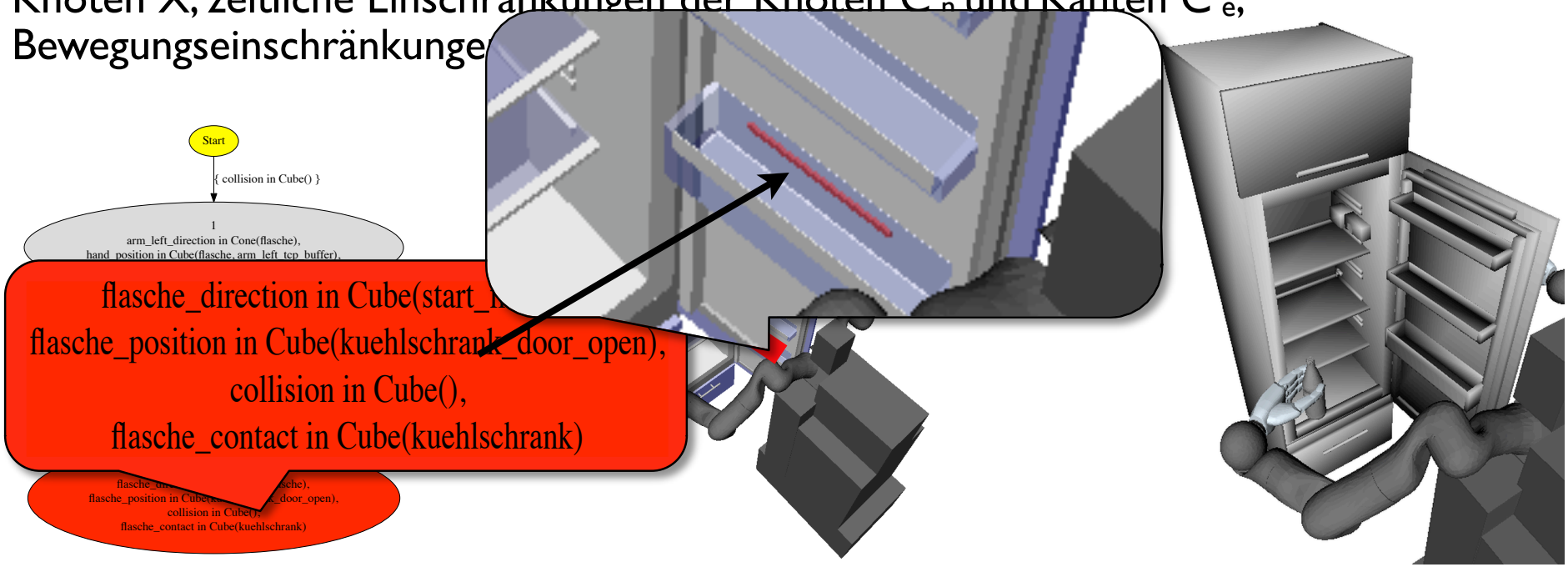
- Tupel (X, C^t_n, C^t_e, C^b_n, C^b_e)
- Knoten X, zeitliche Einschränkungen der Knoten C^t_n und Kanten C^t_e, Bewegungseinschränkunge

Strategiegraph | Bewegungseinschränkungen (rot) | Planung / Ausführung

Rückblick: Repräsentation des Planungsproblems

„Strategiegraph“:

- Tupel $(X, C^t_n, C^t_e, C^b_n, C^b_e)$
- Knoten X, zeitliche Einschränkungen der Knoten C^t_n und Kanten C^t_e, Bewegungseinschränkunge

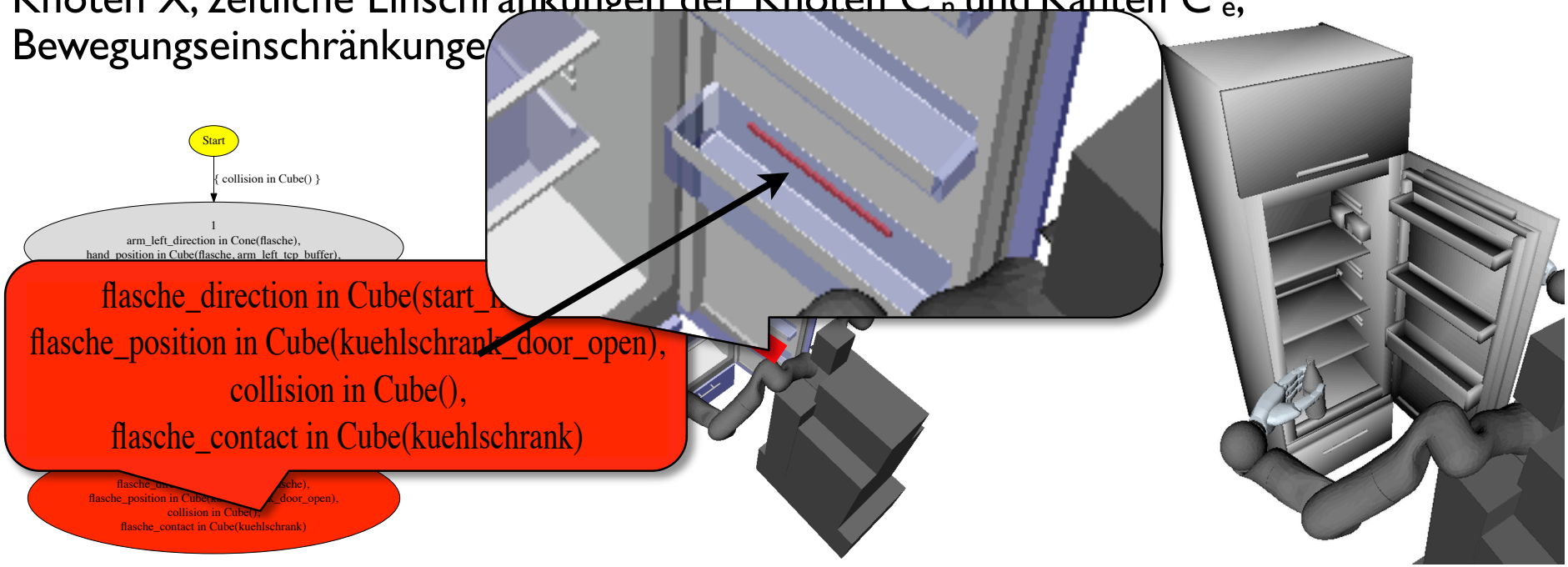


Strategiegraph | Bewegungseinschränkungen (rot) | Planung / Ausführung

Rückblick: Repräsentation des Planungsproblems

„Strategiegraph“:

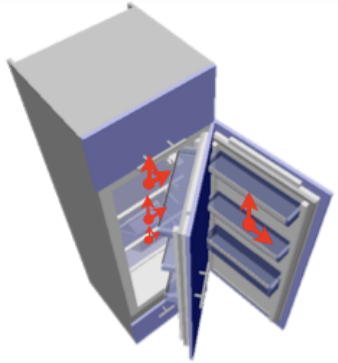
- Tupel $(X, C^t_n, C^t_e, C^b_n, C^b_e)$
- Knoten X, zeitliche Einschränkungen der Knoten C^t_n und Kanten C^t_e, Bewegungseinschränkunge



Strategiegraph | Bewegungseinschränkungen (rot) | Planung / Ausführung

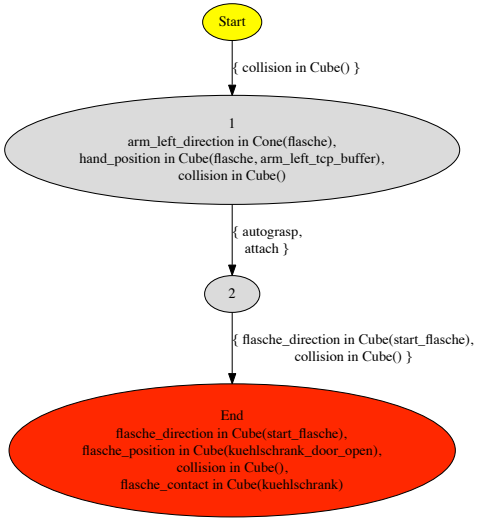
IPoR II: Transformation

- Ausführungsumgebung = Unterschiede in Objekten, Objektposen, Hindernissen (vgl. mit Demonstrationen)
- Einschränkungen im Strategiegraphen referenzieren Koordinatensysteme
- Lage der Koordinatensysteme automatisch adaptiert



- Ausführung unter Verwendung von Bahnplanung
  - Knoten: Teilziele der Manipulationsaufgabe
  - Kanten: Übergänge zwischen den Teilzielen

- Lösen eines Planungsproblems für jedes Teilziel



Strategiegraph

Übersicht

- Bahnplanung:
  - Definitionen
  - Simpler Rapidly-exploring Random Tree (RRT) Planer
  - Hinderniserweiterung und Pfadglättung
  - TC-RRT: Planung mit Task Constraints
- Griffklassifikation:
  - Cutkosky-Hierarchie
- Griffplanung:
  - Definitionen
  - Vorwärtsplaner: Grasplt!

Bahnplanung: Definitionen (I)

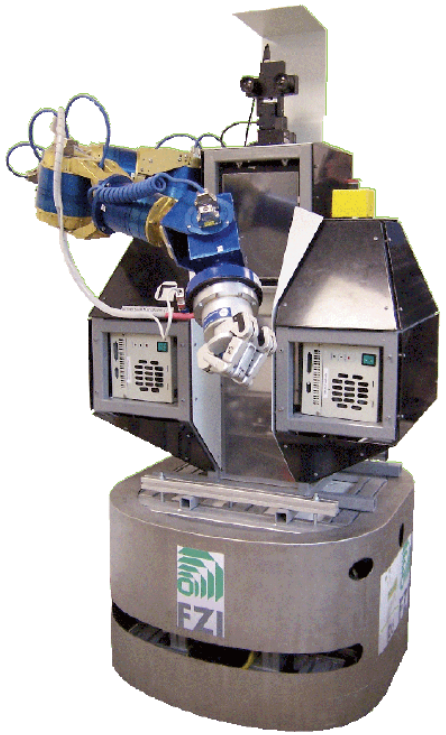
Wiederholung (siehe Robotik I):

- Eine Konfiguration K ist eine vollständige, eindeutige Beschreibung des Zustands eines Roboters A, z.B.
  - im euklidischen Raum durch seine Position und Orientierung
  - im Gelenkwinkelzustandsraum durch die Werte der Gelenke
- Der Konfigurationsraum des Roboters A ist der Raum IK aller möglichen Konfigurationen von A
- Ein Weg für den Roboter A von der Konfiguration K_{Start} zur Konfiguration K_{Ziel} ist eine stetige Abbildung:

$$\tau: [0, 1] \rightarrow IK$$
$$\tau(0) = K_{Start}, \tau(1) = K_{Ziel}$$

- Eine Einschränkung für den Roboter A ist eine Abbildung:

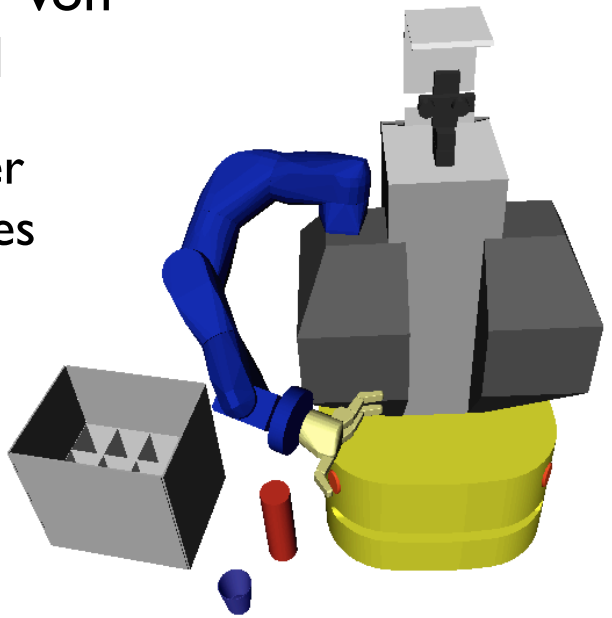
$$\lambda : IK \rightarrow [0, 1]$$



Beispiel: Albert II
(p_x p_y p_α
a_1 a_2 a_3 a_4 a_5 a_6 a_7
h_1 h_2 h_3 h_4)

Bahnplanung: Definitionen (II)

- Ein Arbeitsraumhindernis H ist der Raum, welcher von einem Objekt im Arbeitsraum eingenommen wird



- Ein Konfigurationsraumhindernis IK_{H_i} ist die Menge aller Punkte des Konfigurationraumes, welche innerhalb eines Arbeitsraumhindernisses H_i liegen:

$$IK_{H_i} = \{ K \in IK \mid K \in Hi\}$$

- Der Hindernisraum IK_H ist die Menge aller Konfigurationsraumhindernisse:

$$IK_H = \bigcup_i IK_{H_i}$$

- Der Freiraum IK_F ist die Menge aller Punkte aus IK, welche nicht im Hindernisraum liegen:

$$IK_F = IK \setminus IK_H$$

- Ein kollisionsfreier Weg ist ein Weg τ mit $Bild(\tau) \subseteq IK_F$

Bahnplanung im Konfigurationsraum (I)

Bewegungen eines Roboters werden als Zustandsänderungen über die Zeit relativ zu einem stationären Koordinatensystem (kartesischer Raum, Gelenkwinkelraum) aufgefasst = Trajektorie im Konfigurationsraum

- Gegeben:
  - K_{Start} = Startkonfiguration
  - Λ_{Ziel} = Menge der Zieleinschränkungen
- Gesucht:
  - Kollisionsfreier Weg τ von K_{Start} nach K mit $\lambda(K)=1 \ \forall\lambda \in \Lambda_{Ziel}$
- Bedingungen:
  - Im Allgemeinen: Gütekriterien, Neben-,Rand- sowie Zwangsbedingungen

Bahnplanung im Konfigurationsraum (II)

Anmerkungen:

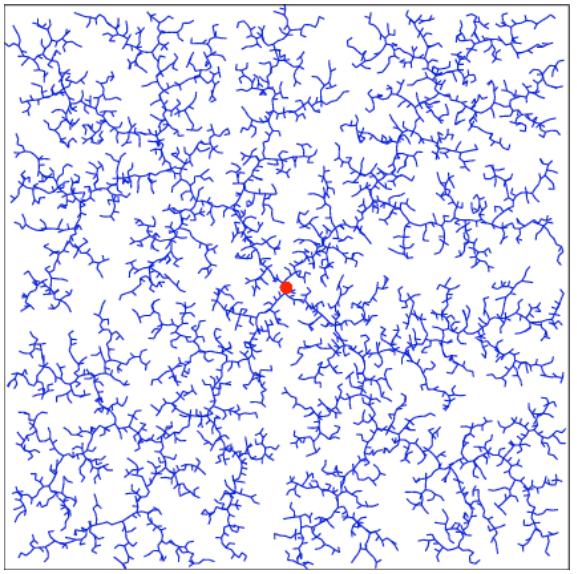
- Abstraktion von Roboter und Umwelt zu einem Punkt im Konfigurationsraum
- Kollisions- bzw. Einschränkungsüberprüfung ist Blackbox $f: IK \rightarrow \{0, 1\}$:

  Beispiel: $f(K) = \bigwedge_{\nu \in \Lambda_{Weg}} \nu(K) \geq \varepsilon(\nu)$
- Entwicklung allgemeiner Planungsverfahren auf Basis dieser Abstraktion:
  - Bahnplanung entspricht Suche nach stetiger Verbindung zweier Punkte im Konfigurationsraum
  - Keine explizite Beschreibung des Freiraums notwendig, d.h. Suchverfahren ist unabhängig von der Struktur und Repräsentation des Freiraums

Bahnplanung: RRT

RRT = Rapidly-exploring Random Tree

- [LaValle/Kuffner99]: Randomized Kinodynamic Planning
- Entwickelt zur effizienten Durchsuchung hoch-dimensionaler Räume
- Geeignet für holonome und nicht-holonome Problemstellungen mit Einschränkungen
- Inkrementeller Aufbau einer Baumstruktur



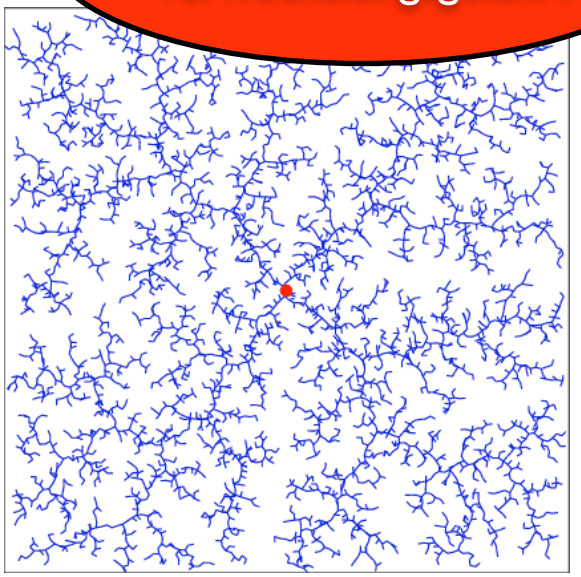
- Minimiert erwarteten Abstand eines Punkts zu einem Knoten im Baum

Bahnplanung: RRT

Viele Varianten
Verwechslungsgefahr!

RRT = Rapidly-exploring Random Tree

- [LaValle/Kuffner99]: Randomized Kinodynamic Planning
- Entwickelt zur effizienten Durchsuchung hoch-dimensionaler Räume
- Geeignet für holonome und nicht-holonome Problemstellungen mit Einschränkungen
- Inkrementeller Aufbau einer Baumstruktur



- Minimiert erwarteten Abstand eines Punkts zu einem Knoten im Baum

RRT: Algorithmus (I)

BUILD_RRT(K_{Start}, n, ε)

1. T.init(K_{Start})
2. for k = 1 to n
3. K_{Zuf} ← RAND_CONF()
4. K_{Nahe} ← NEAREST_VERTEX(K_{Zuf}, T)
5. K_{Neu} ← EXTEND(K_{Nahe}, K_{Zuf}, ε)
6. T.add_vertex(K_{Neu})
7. T.add_edge(K_{Nahe}, K_{Neu})
8. Return T

Kommentare:

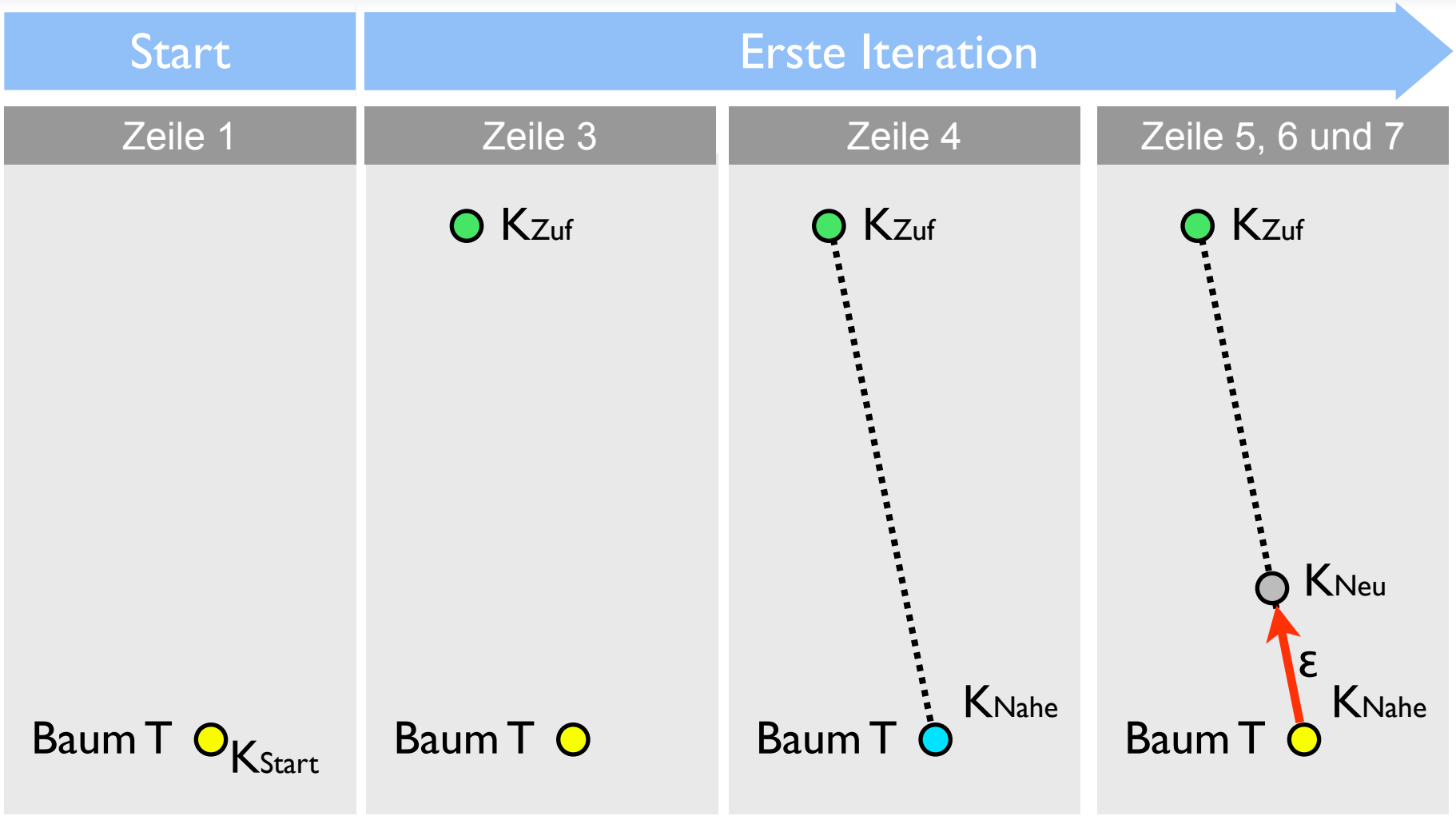
Neuer Baum mit Startkonfiguration in der Wurzel

Gleichverteilt zufällige Erzeugung einer Konfiguration

Bestimmung des nächsten Knotens

Erzeugung einer neuen Konfiguration

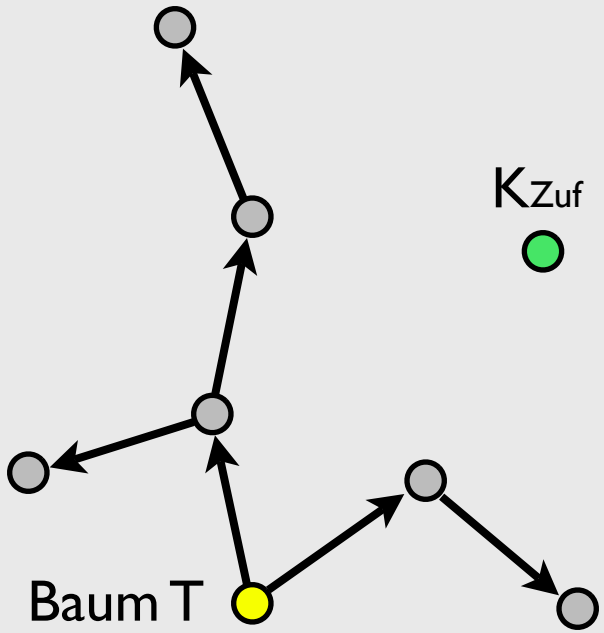
RRT: Algorithmus (II)



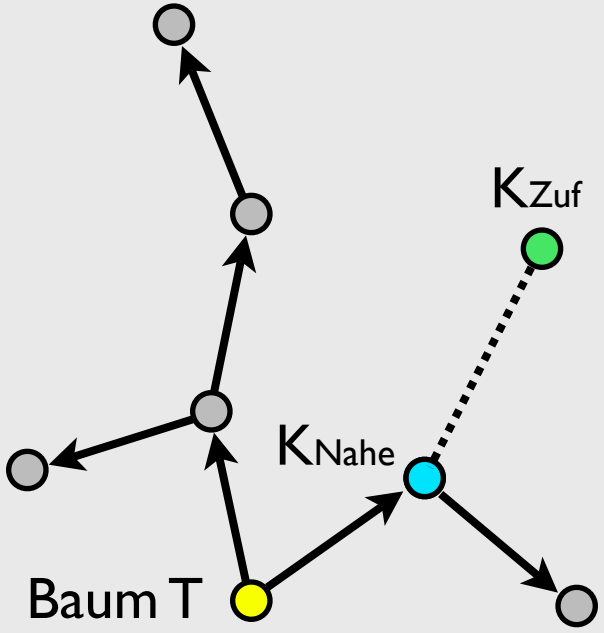
RRT: Algorithmus (III)

7te Iteration

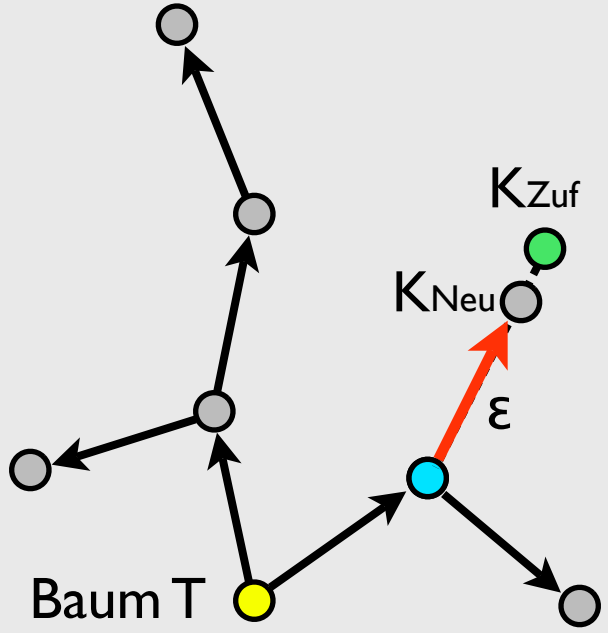
Zeile 3



Zeile 4



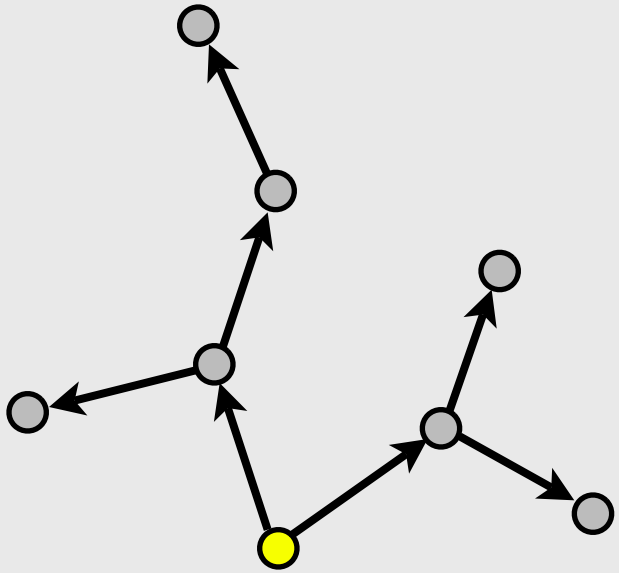
Zeile 5, 6 und 7



RRT:Voronoi-Bias (I)

Welcher Knoten wird am wahrscheinlichsten erweitert?

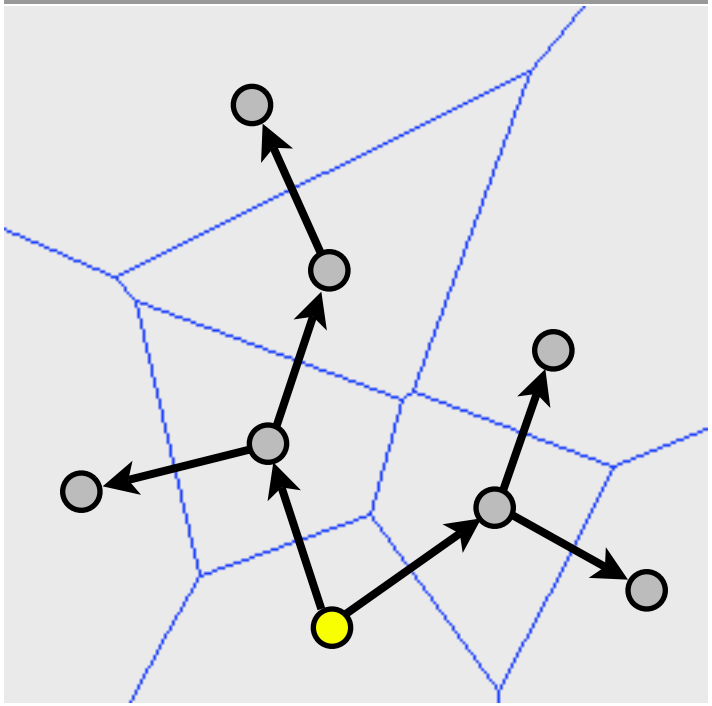
Baum nach 8. Iteration



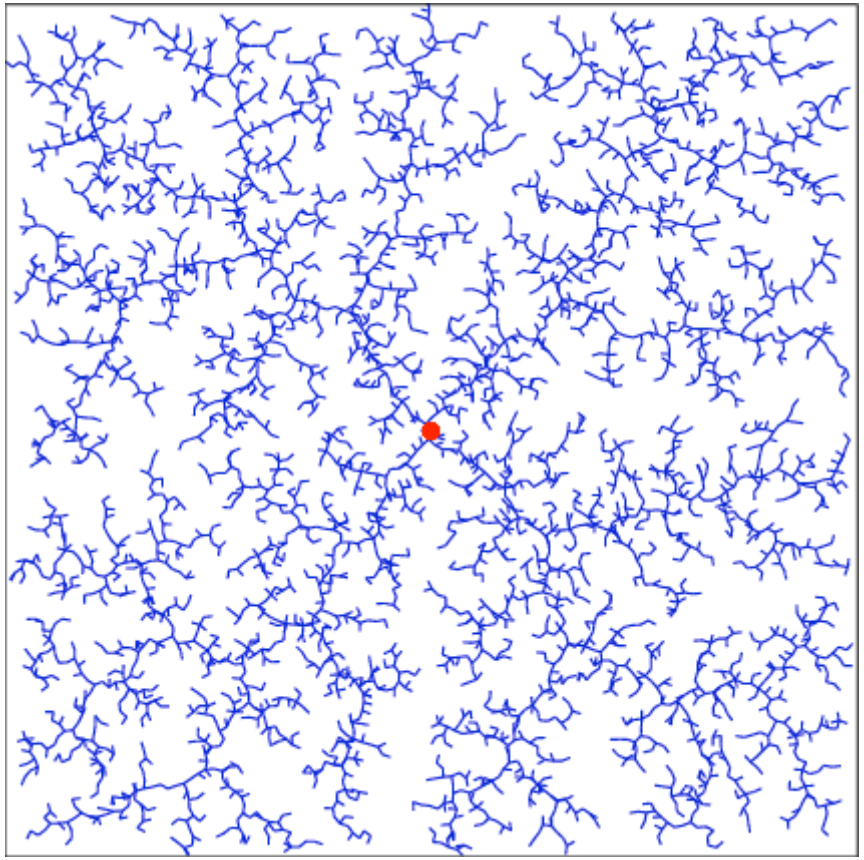
Knoten mit größter Voronoi-Region (Voronoi-Bias)

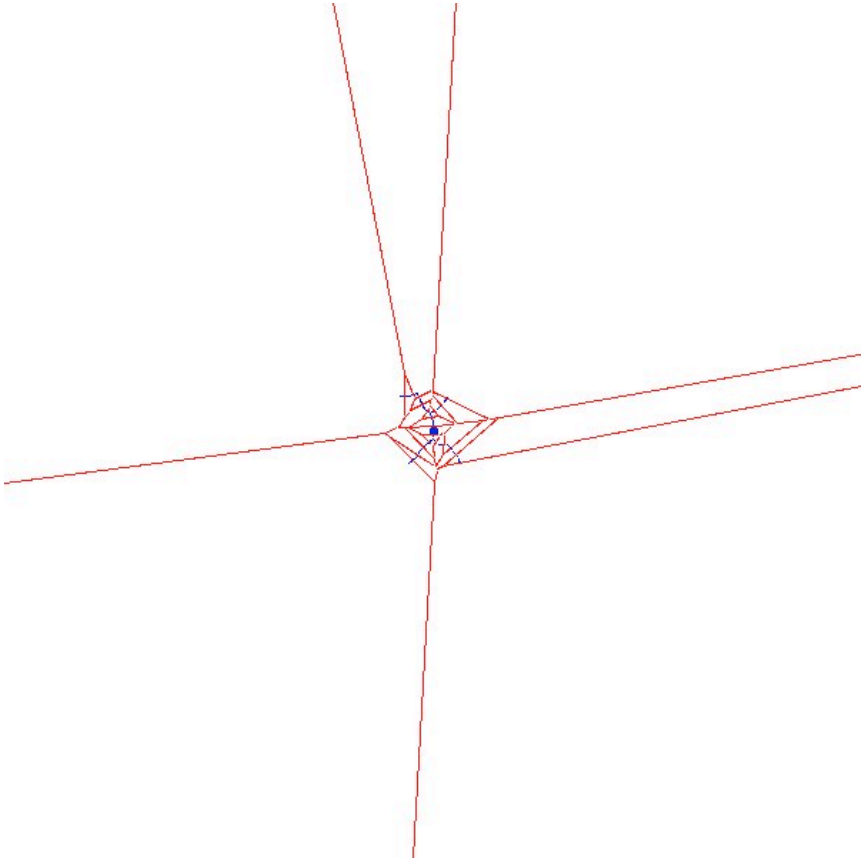
Voronoi-Gebiete am Anfang im Randbereich groß
→ Rasche Exploration
→ Verfeinerung

Baum nach 8. Iteration

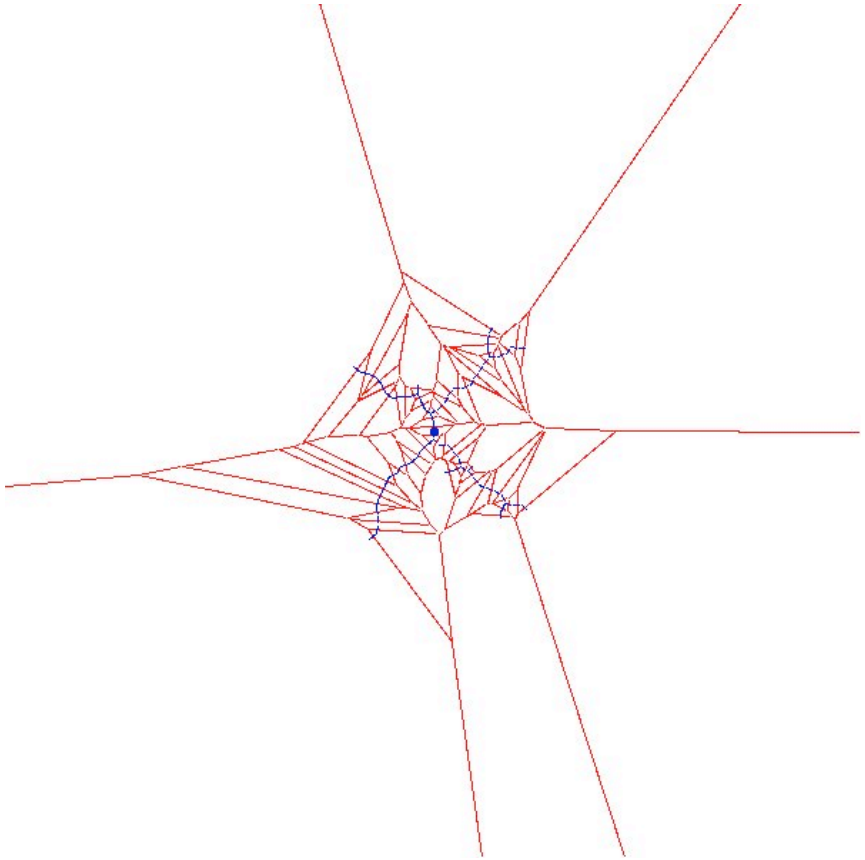


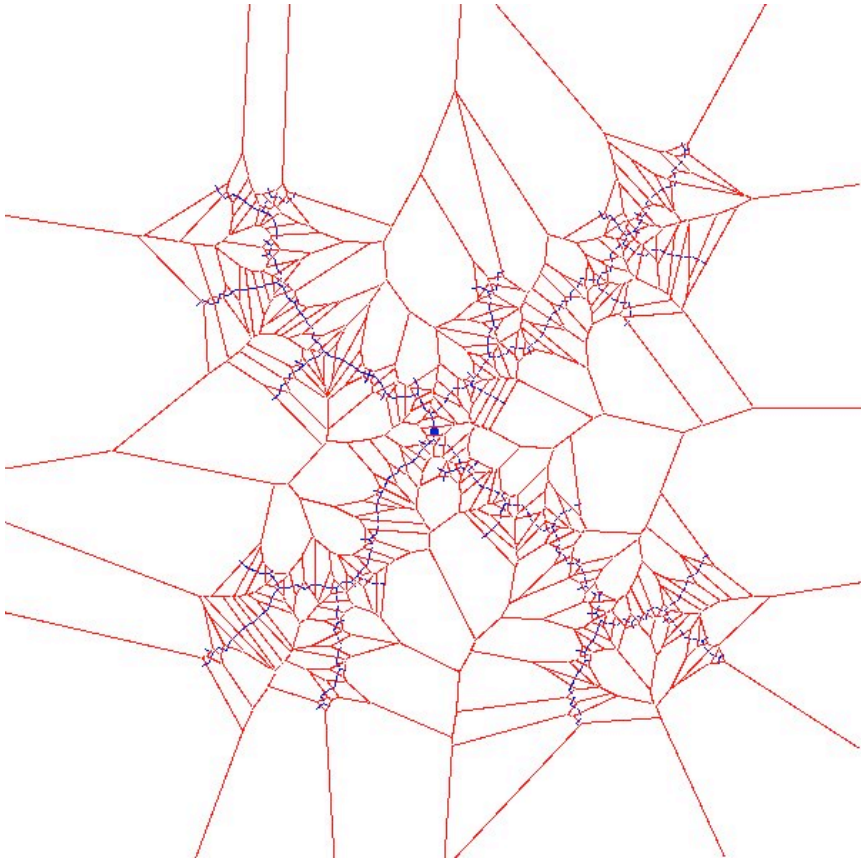
RRT: Voronoi-Bias (II)



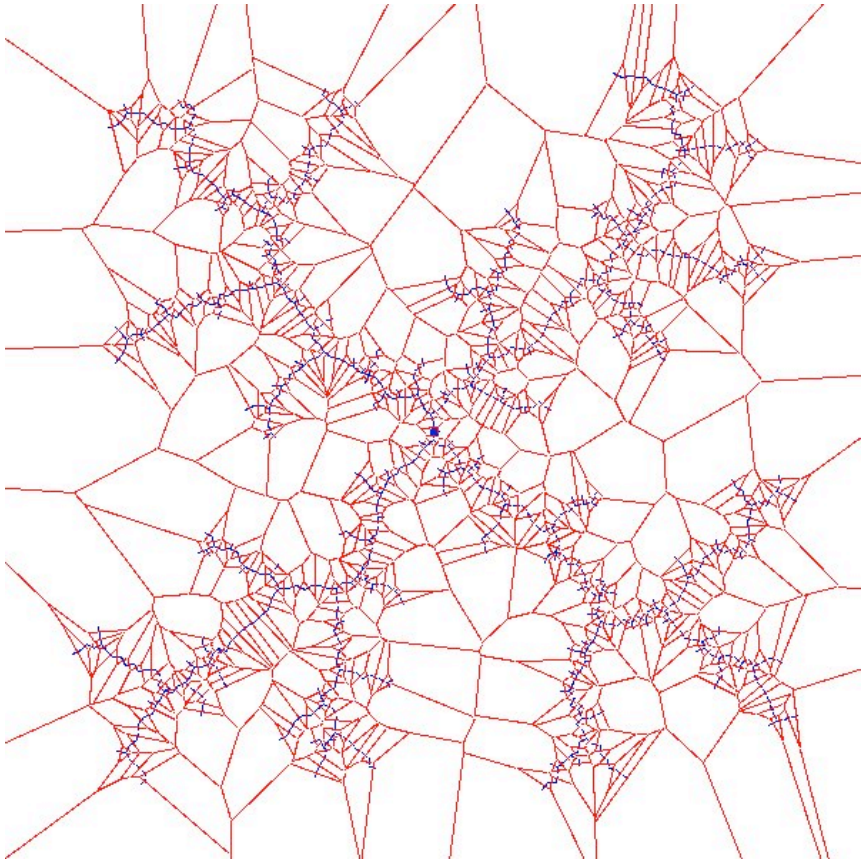


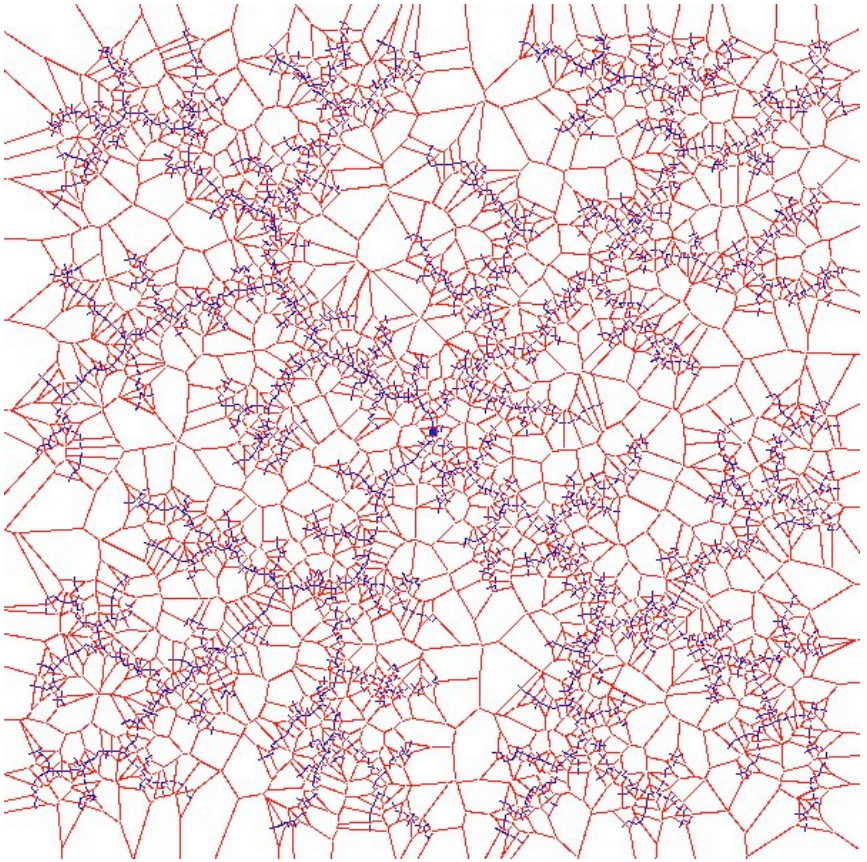
RRT: Voronoi-Bias (III)





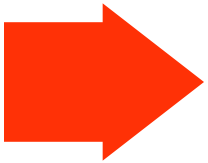
RRT:Voronoi-Bias (IV)





RRT: Zusammenfassung

- Allgemeines Verfahren zur Durchsuchung hoch-dim. Räume
- Online-Verfahren
- Approximation des Suchraums durch eindimensionale Struktur: Baum
- Rasche Exploration des Suchraums: Voronoi-Bias
- Probabilistische (oder deterministische) Stichprobenerzeugung
- Einfach zu implementieren, nur wenige Parameter (ε, Distanzfunktion auf IK)



Anwendung in der Bahnplanung:
- Wie zielgerichtet suchen?
- Wie kollisionsfreie Wege erzeugen?
- Wie Einschränkungen berücksichtigen?

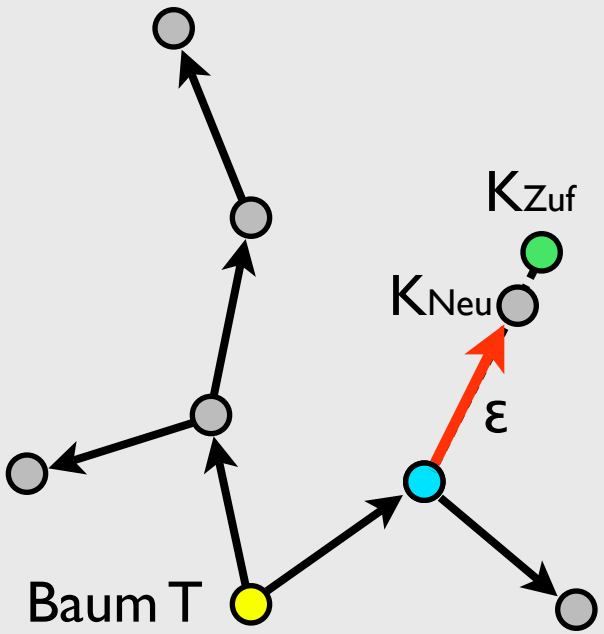
Simpler RRT Planer

Erweiterung des Basisalgorithmus:

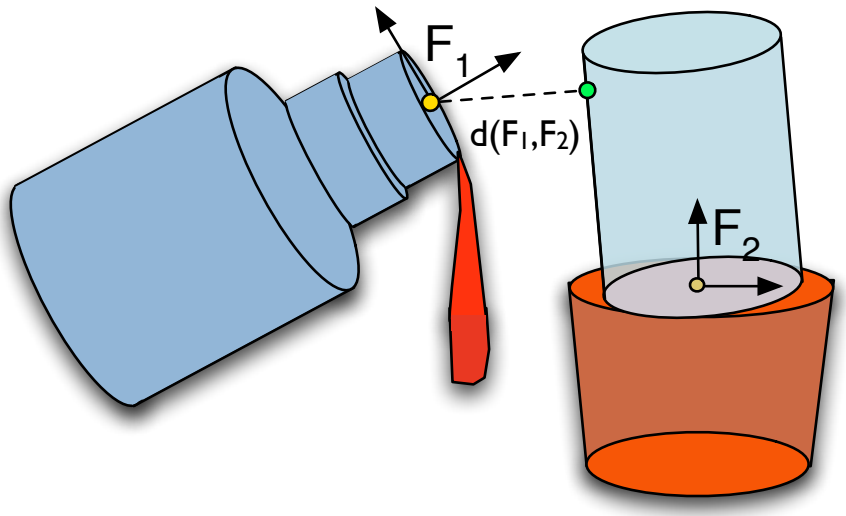
- Es werden nur Konfigurationen hinzugefügt, die alle Einschränkungen erfüllen
- Bei der Stichprobenerzeugung werden mit einer bestimmten Wahrscheinlichkeit Zielkonfigurationen generiert
- Der Planungsprozess ist beendet, wenn die letzte Konfiguration die Zieleinschränkungen erfüllt

RRT: Einbeziehung von Einschränkungen (I)

Zeile 5, 6 und 7



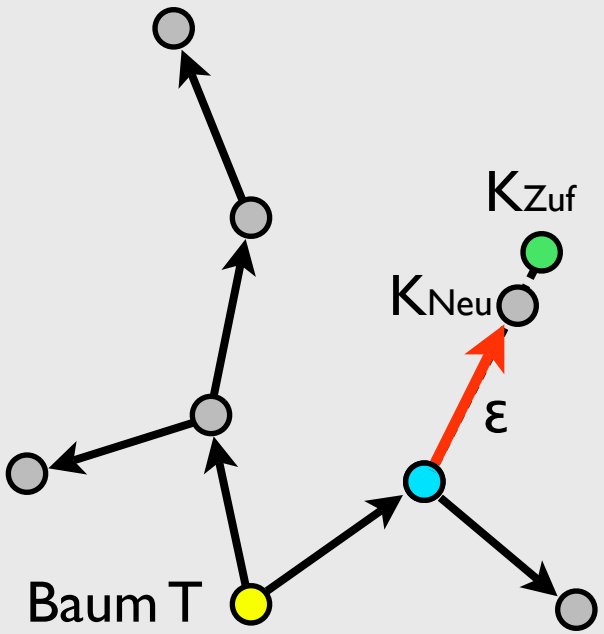
- K_{Neu} wird übernommen, wenn alle Einschränkungen zwischen K_{Nahe} nach K_{Neu} erfüllt sind
- Keine Distanz, nur Ja / Nein



Gelernte Einschränkungen
Distanz $d(F_1,F_2) > \delta$
schnell

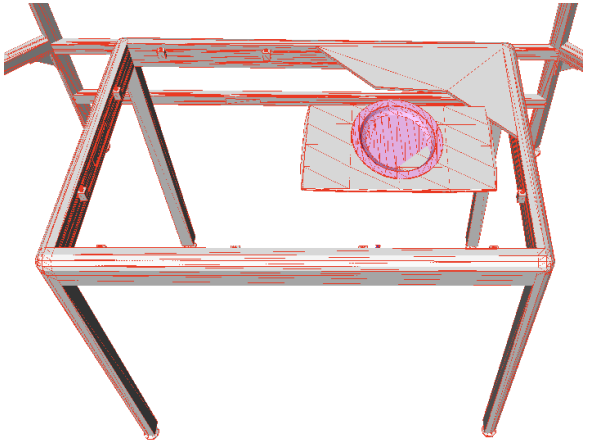
RRT: Einbeziehung von Einschränkungen (II)

Zeile 5, 6 und 7



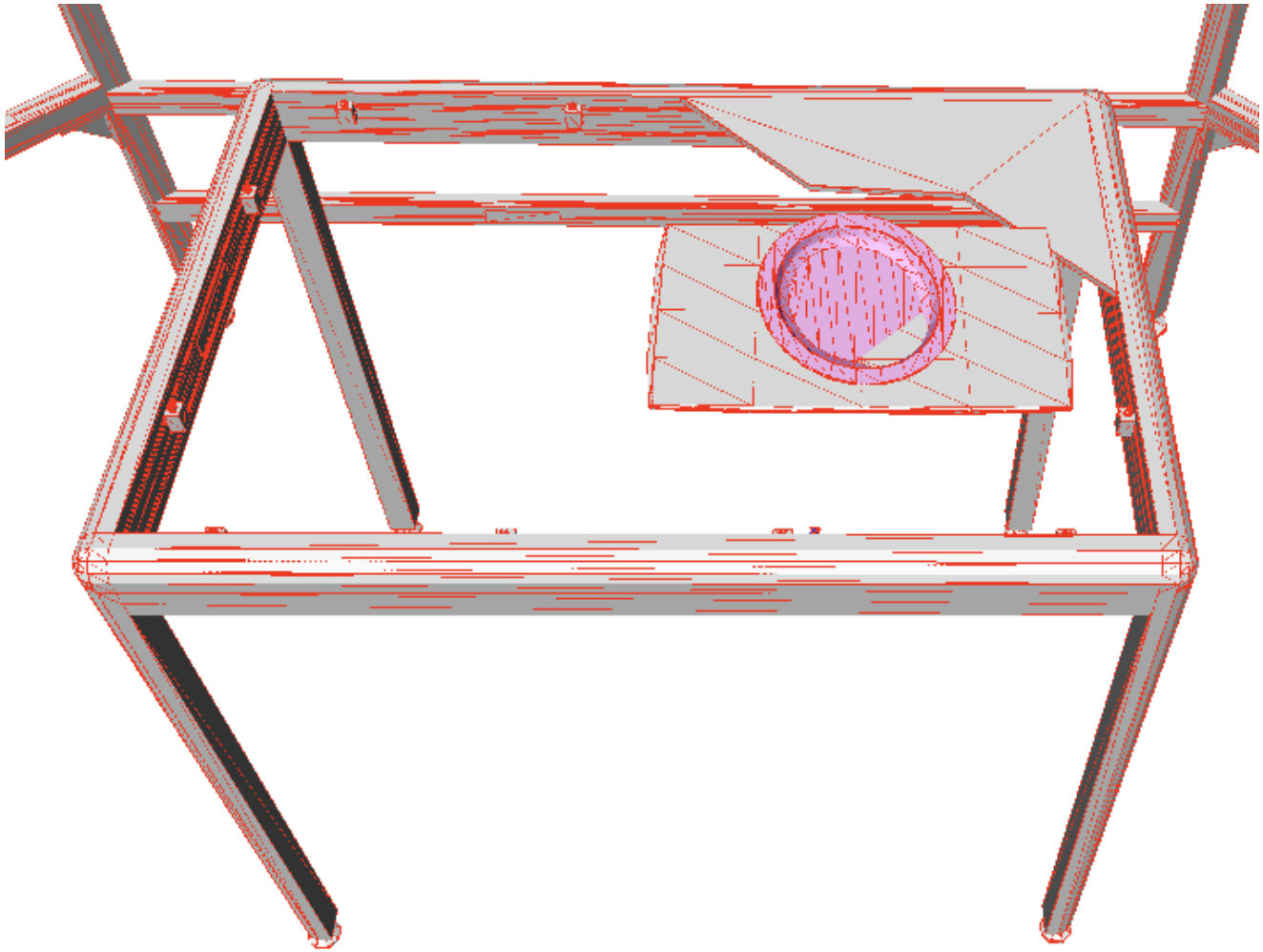
Kollisionen:

- Grundlage: Geometriemodell der Objekte
- Standard: 3d-Dreiecksnetze
- Kollisionsüberprüfung: mindestens zwei Dreiecke schneiden sich*
- Optimierte Algorithmen
- Vergleichsweise langsam

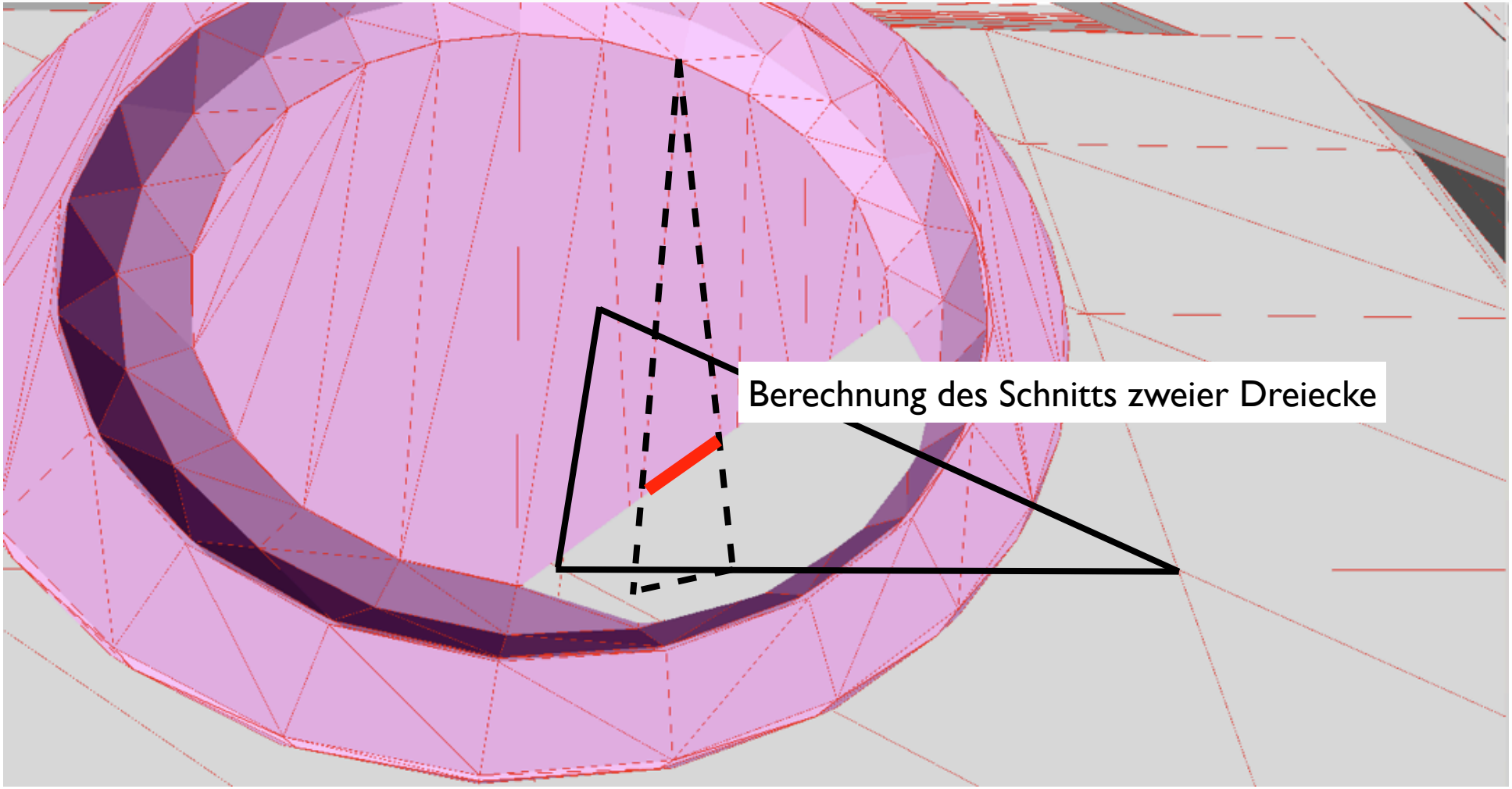


*oder 1. Objekt in 2. komplett enthalten

RRT: Kollisionsberechnung (I)



RRT: Kollisionsberechnung (II)



Wie zielgerichtet planen? (I)

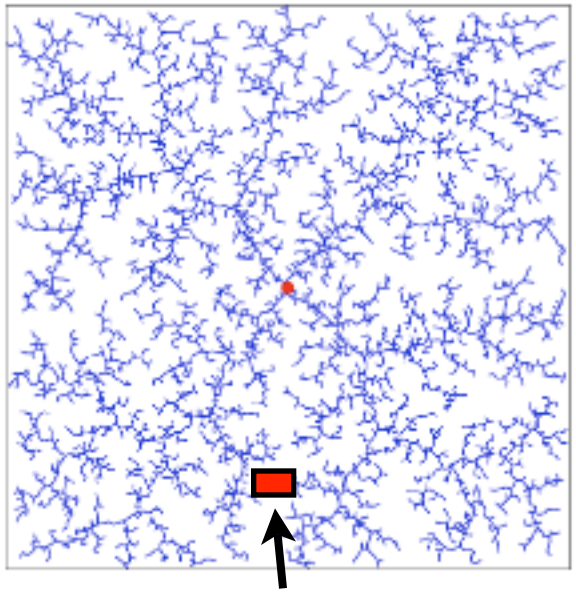
Zieleinschränkungen definieren i.d.R. stark begrenztes Gebiet in IK, z.B. endliche Menge von gültigen Zielkonfigurationen

➡ Wahrscheinlichkeit eine Zielkonfiguration zufällig zu erzeugen ist minimal

➡ Modifikation der Stichprobenerzeugung (z.B. $\delta = 0.01$):

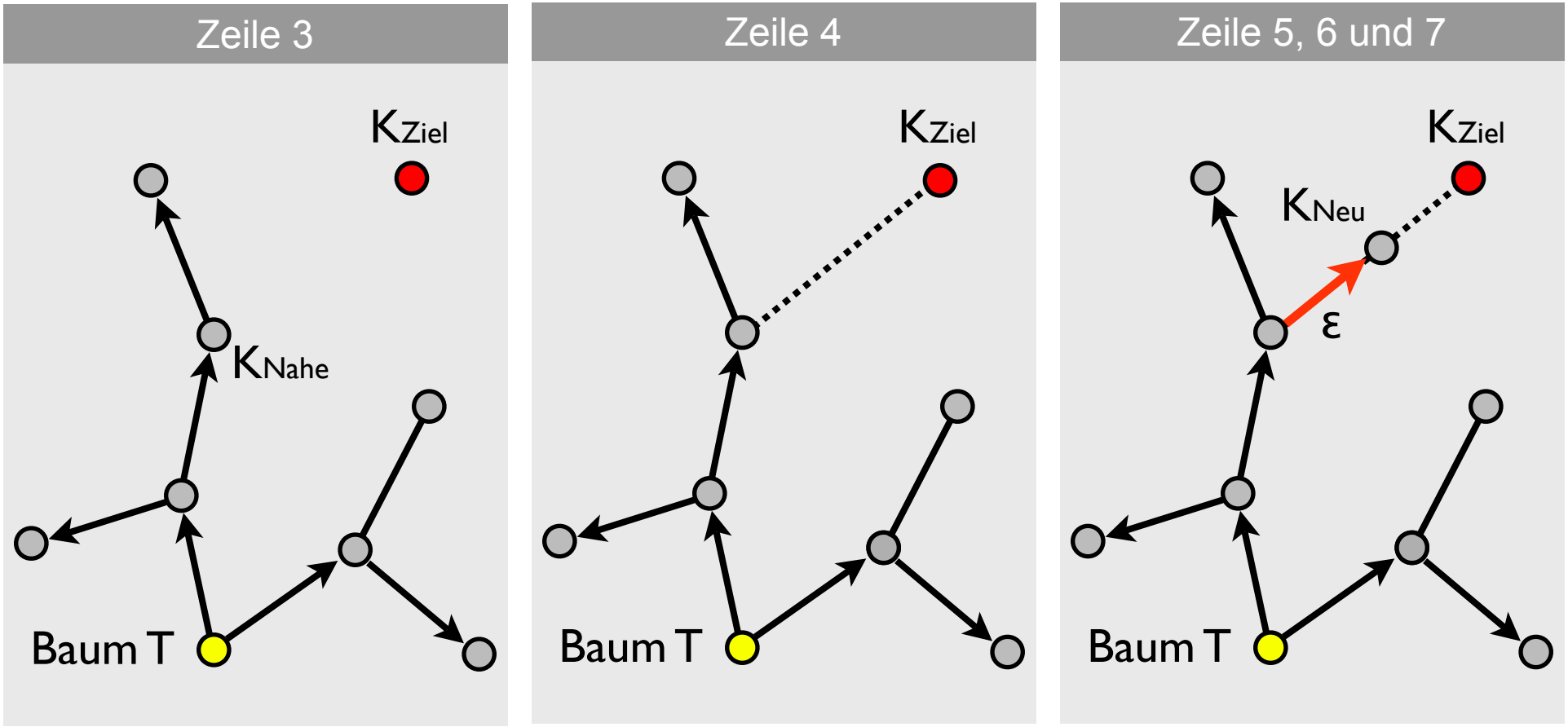
RAND_CONF(f_{Ziel}, δ)

1. $P \sim U([0,1])$
2. If $P < \delta$
3. Return $K \in IK : f_{Ziel}(K) = 1$
4. Else
5. Return $K \sim U(IK)$



Wie zielgerichtet planen (II)

$K_{Ziel} = RAND\_CONF(f_{Ziel}, \delta)$



Simpler RRT Planer

RRT_SIMPLE(K_{Start}, f_{Ziel}, f_{Weg}, ε, δ)

1. T.init(K_{Start})
2. For k = 1 to MAX
3. K_{Zuf} ← RAND_CONF(f_{Ziel}, δ)
4. K_{Nahe} ← NEAREST_VERTEX(K_{Zuf}, T)
5. K_{Neu} ← EXTEND(K_{Nahe}, K_{Zuf}, ε)
6. If f_{Weg}(K_{Neu}) = 1
7. T.add_vertex(K_{Neu})
8. T.add_edge(K_{Nahe}, K_{Neu})
9. If f_{Ziel}(K_{Neu}) = 1
10. Return (T, Gefunden)
11. Return (T, Nicht gefunden)

Kommentare:

f_{Ziel} ist Blackbox für Zieleinschränkungen

f_{Weg} ist Blackbox für Einschränkungen

Lösungspfad: Rückverfolgung der Elternknoten beginnend mit dem letzten Knoten

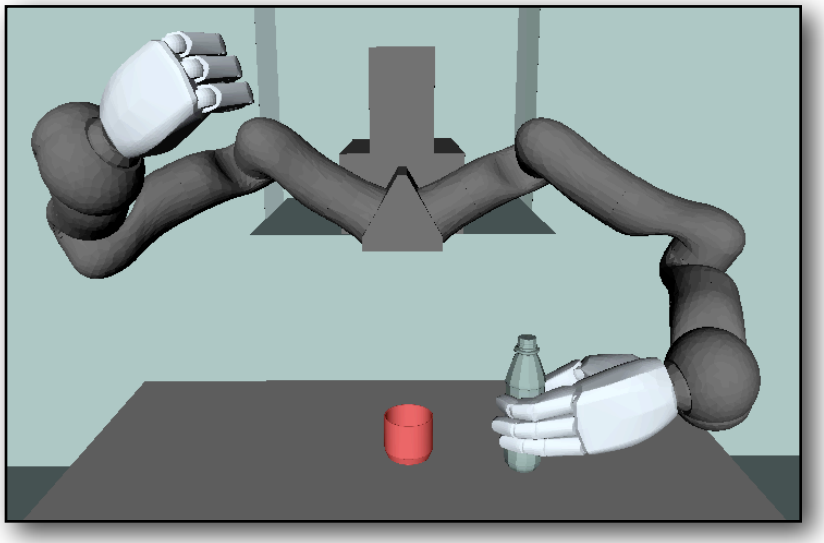
Simpler RRT Planer: Beispiel (I)

Beispiel:
Holonome Planung für einen Roboterarm mit 7 Freiheitsgraden

- Konfigurationsraum IK = $[0,1]^7$
- Distanzfunktion auf IK:

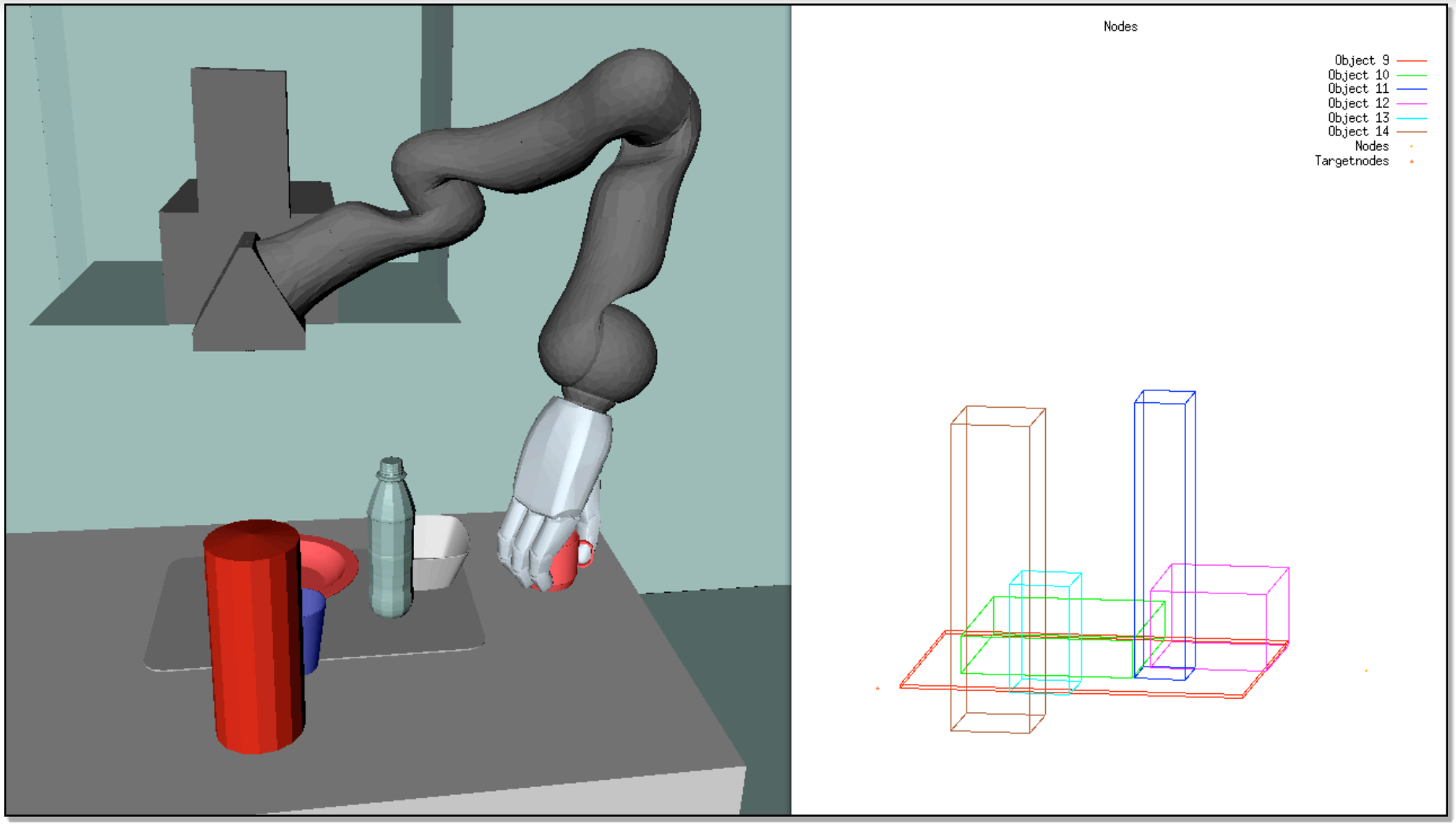
$$d(k,s) = \sum_{i=1..7} w_i \, (k_i - s_i)$$

- Gewichtsvektor w=(1.5,1.5,1.5,1.25,1,1,1)
- $\varepsilon = 0.001, \delta = 0.01$
- $f_{Ziel}(K) = 1 \Leftrightarrow K = K_{Ziel}$
- $f_{Weg}(K) = 1 \Leftrightarrow$ K ist kollisionsfrei



Planung für linken Arm ohne Hand

Simpler RRT Planer: Beispiel (II)



Reale Anwendung ...

Kann der Planer so direkt eingesetzt werden?

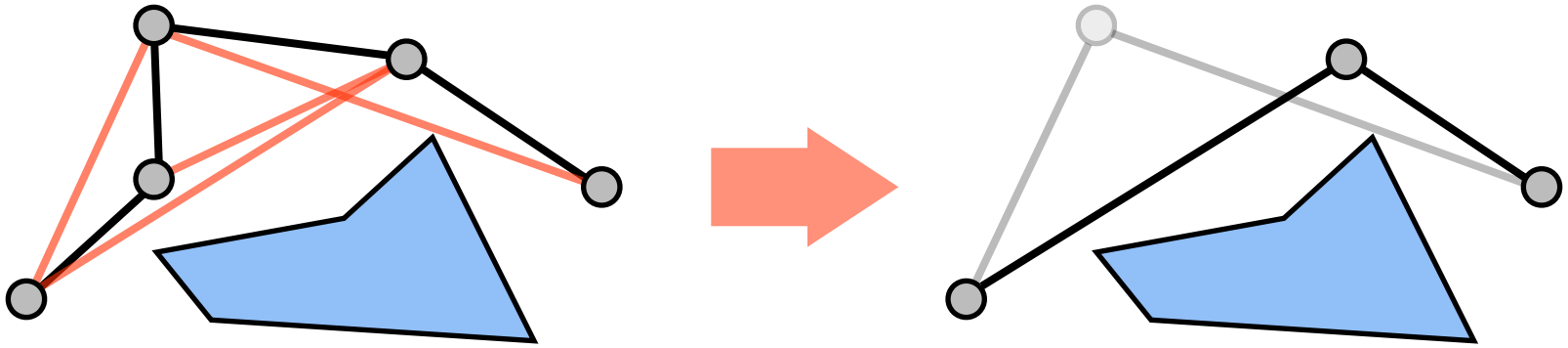
Nein:

- Glättung notwendig, da Weg nicht glatt
- Hinderniserweiterung notwendig, da kleine Abweichungen von der Bahn zu Kollisionen führen können

RRT: Glättung von Lösungswegen

Verringerung der Länge des Lösungswegs:

- Zufällige Wahl zweier Knoten im Lösungsweg
- Überprüfung der Verbindung beider Knoten auf Einhaltung der Einschränkungen
- Falls die Verbindung geringere Kosten aufweist als der entsprechende Teillösungspfad, Verbindung der beiden Knoten und Löschen der dazwischenliegenden Knoten aus dem Lösungspfad



Bahnplanung: Hinderniserweiterung

Zweck: Berücksichtigung eines Toleranzbereichs im Planungsprozess durch Vergrößerung der Hindernisse

Gründe:

- Diskretisierung des Konfigurationsraums bzw. Abtastung von Trajektorien
- Fehler der Objektlokalisierung: Kalibrierung des Roboters, Objekterkennung/verfolgung
- Abweichungen in der Bewegung: Rutschen, lockere Seilzüge, Modellierung, Impedanzregelung, Interpolationsart

Probleme:

- Künstliche Einschränkung des Arbeitsraums
- Erzeugung von Narrow-Passages

Simpler RRT Planer: Zusammenfassung

- Probabilistisch vollständig
- Uniforme Stichprobenverteilung → Bereiche im Suchraum mit geringem Lebesgue-Maß, z.B. Narrow Passages, werden nicht abgedeckt
- Hohe Laufzeitvarianz
- Kein problemspezifisches Wissen
- Ineffiziente Ergebnispfade → Glättung notwendig

Manipulation

Einschränkungen können Unterräume mit Lebesgue-Maß 0 erzeugen:
→ Modifikation notwendig
→ Blackbox-Formulierung der Einschränkungen problematisch

Bahnplanung unter Einschränkungen (I)

Zweck: Berücksichtigung von Einschränkungen im Planungsprozess

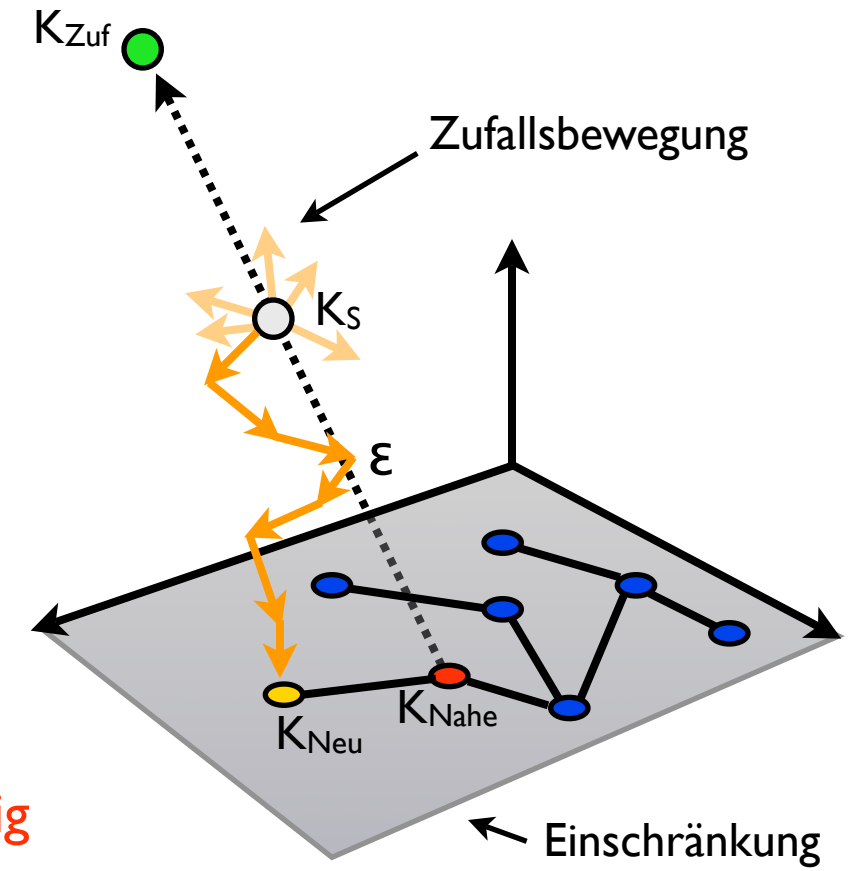
Wiederholung:

- Ziele haben gleiche Struktur wie Einschränkungen
- Kollisionsfreiheit ist eine Einschränkung
- Erweiterung des Blackbox-Konzepts auf Einschränkungen möglich.
  Problem: Narrow-Passages und Nullmengen treten sehr viel häufiger als bei Kollisionen auf
  → spezielle Methoden zur Berücksichtigung von Einschränkungen nötig

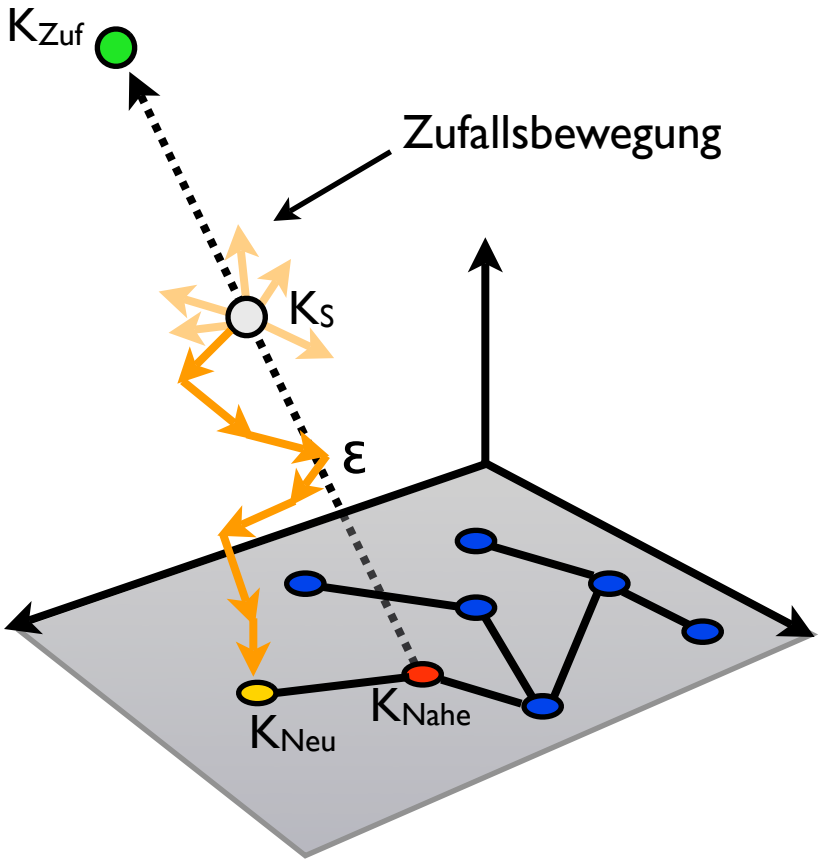
Randomized Gradient Descent (RGD)

- Toleranzwert für Einschränkungen: α
- Zufällige Bestimmung von n Nachbarn von K_S (in Hyperkugel mit Radius d_{max})
- Falls minimale Distanz der Nachbarn kleiner als die Distanz von K_S, ersetze K_S mit dem Nachbarn mit kleinster Distanz
- Wiederholen bis maximale Iterationszahl erreicht oder die Distanz von $K_S < \alpha$ ist

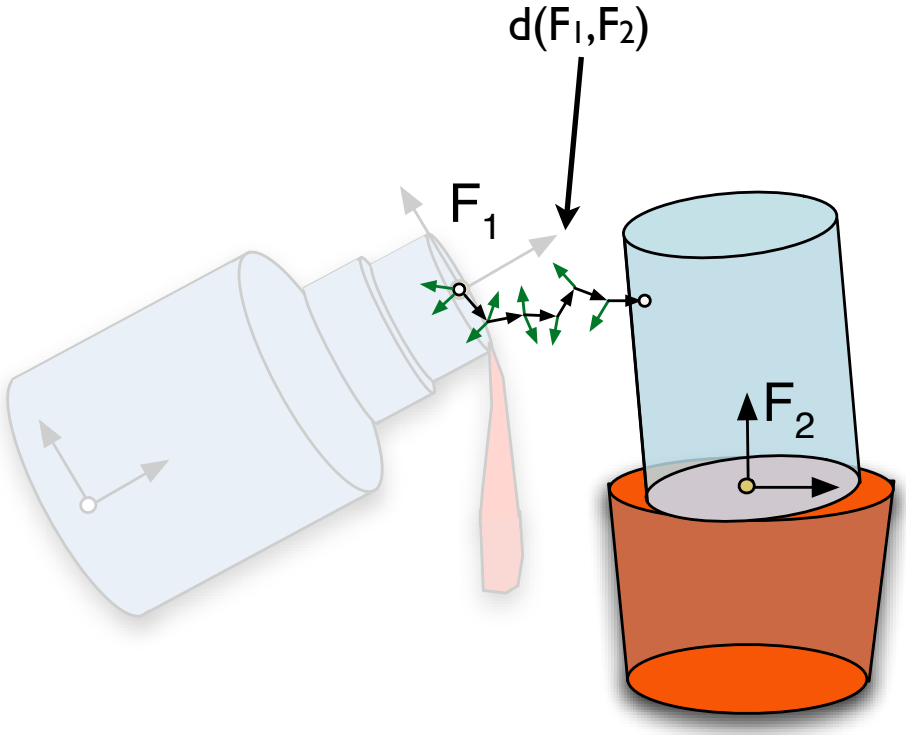
→ Keine Richtungsinformation notwendig



IPoR II: Randomized Gradient Descent

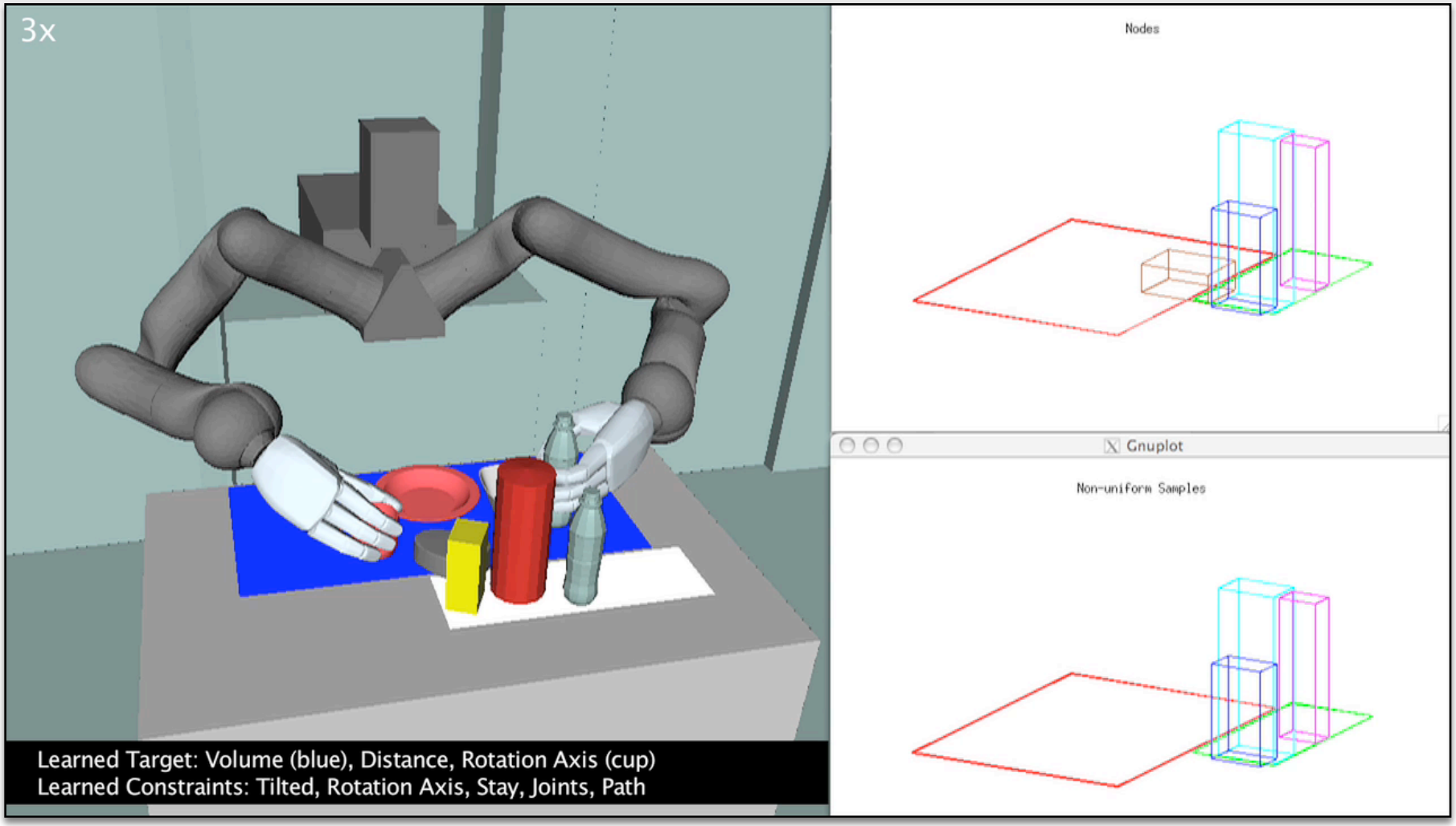


RGD

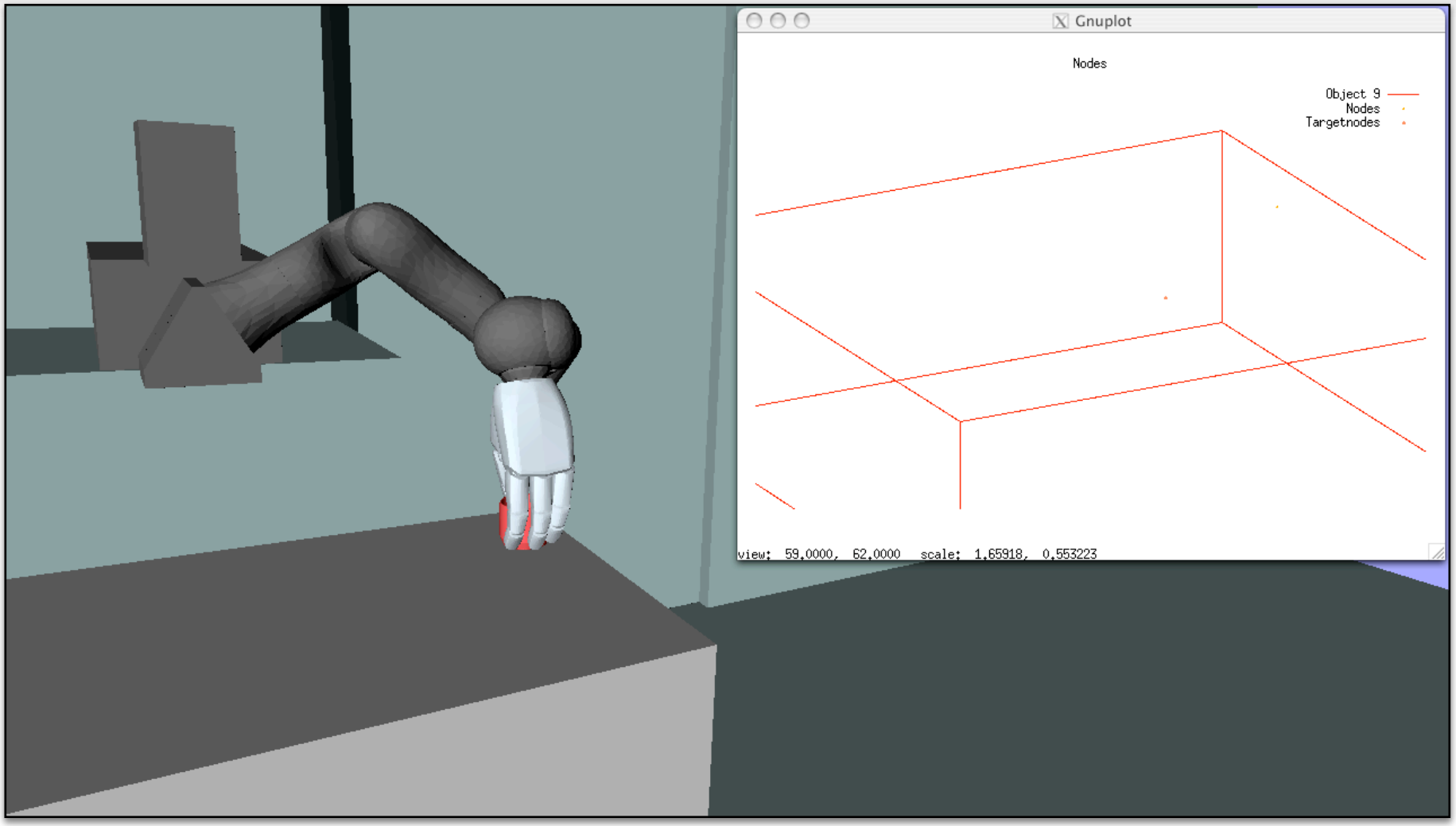


RGD mit Distanz $d(F_1,F_2)$

IPoR II: Beispiel

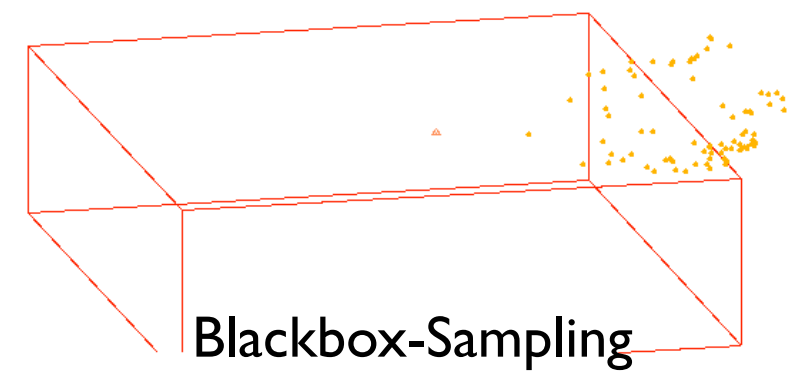


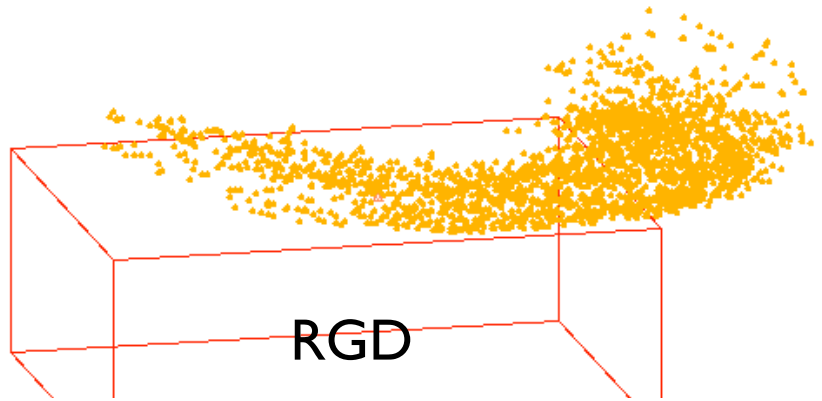
Bahnplanung: Constraints (Beispiel)

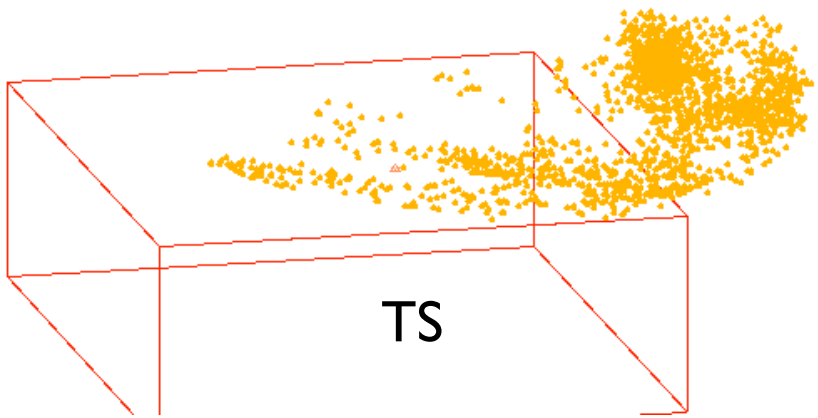


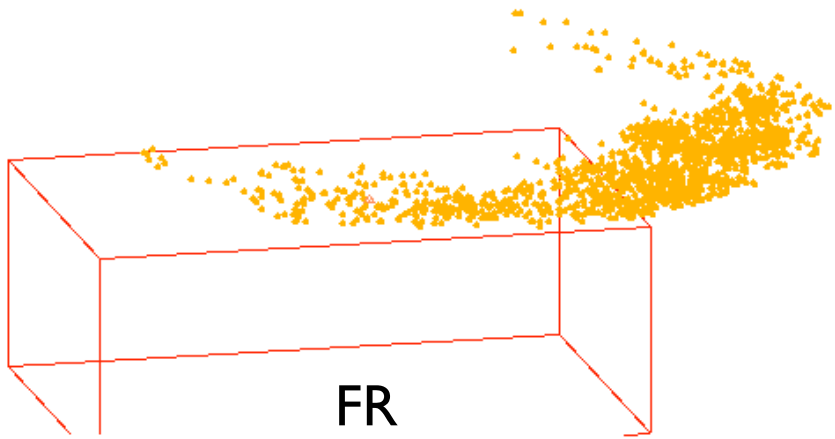
Vergleich der Stichprobenerzeugung

Jacobimatrix-basierte Ansätze bei komplexeren Einschränkungen im Vorteil









RRT: Erweiterungen

- Connect-Heuristik:
  - Multiple Extend-Schritte in einer Iteration
- Bidirektional:
  - Wachsen eines RRTs in der Start- und Zielkonfiguration
  - Bäume wachsen aufeinander zu
  - Planungsproblem gelöst, wenn Bäume verbunden
  - Balanciert: gleiche Anzahl Knoten in beiden Bäumen
- dd-RRT (Jaillet05):
  - Verwendung einer nicht-uniformen Stichprobenverteilung auf der Basis des aktuellen Suchbaums → Besseres Verhalten in eingeschränkten Regionen

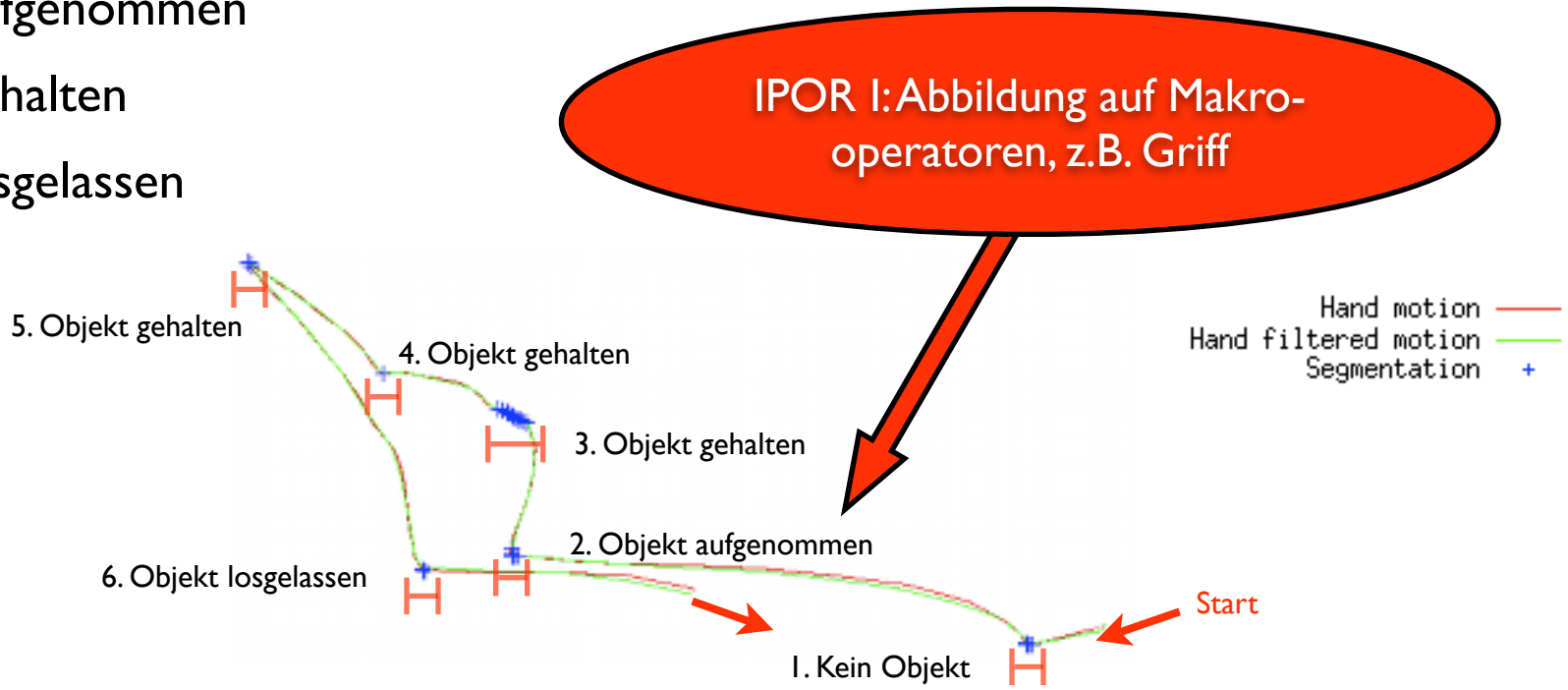
Übersicht

- Bahnplanung:
  - Definitionen
  - Simpler Rapidly-exploring Random Tree (RRT) Planer
  - Hinderniserweiterung und Pfadglättung
  - TC-RRT: Planung mit Task Constraints
- Griffklassifikation:
  - Cutkosky-Hierarchie
- Griffplanung:
  - Definitionen
  - Vorwärtsplaner: GraspIt!

Rückblick - *IPoR II*: Interpretation

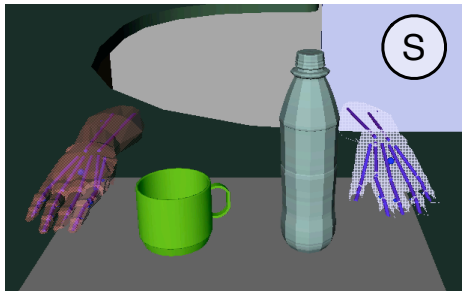
Klassifikation der Segmentierungspunkte in 4 Typen auf Basis des Weltzustands an den Intervallgrenzen:

- Kein Objekt
- Objekt aufgenommen
- Objekt gehalten
- Objekt losgelassen

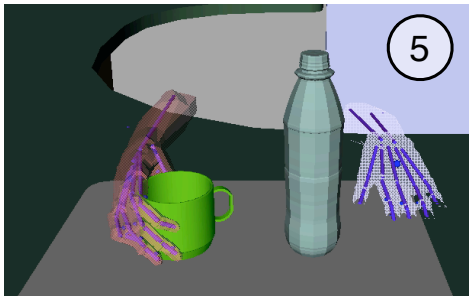


IPoR II: Simple Abbildung auf Greifaktionen

- Klassifikation des menschlichen Griffs im Segmentierungspunkt
- Abbildung auf vordefinierten Griff der gleichen Klasse für Roboterhand
- Bestimmung eines optimalen Griffs in Simulation
- Ausführung auf Roboter



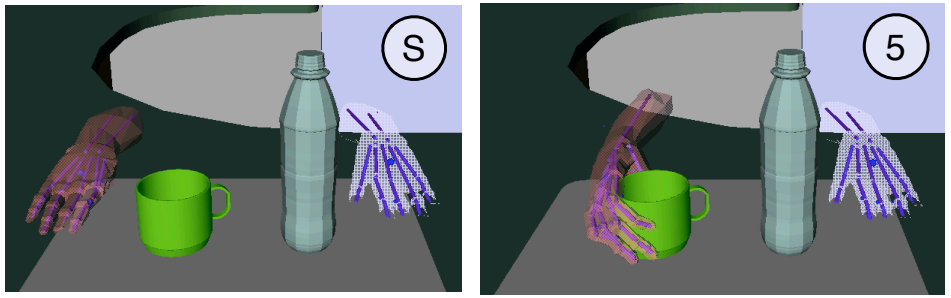
Klassifikation des menschlichen Griffs



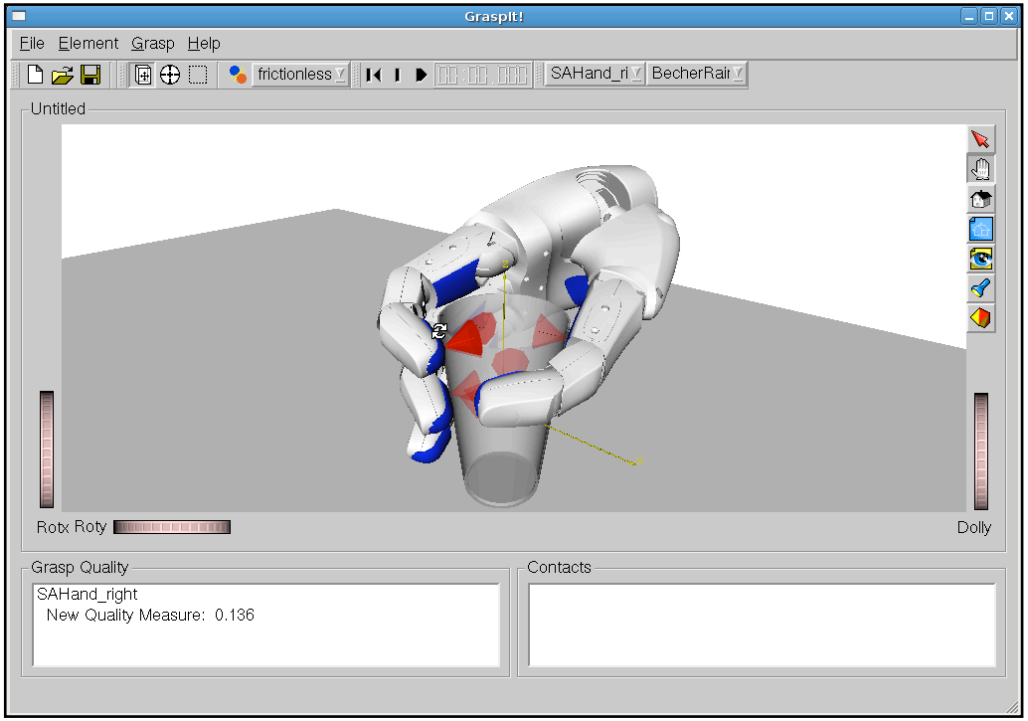
Griffplanung in Simulation

IPoR II: Simple Abbildung auf Greifaktionen

- Klassifikation des menschlichen Griffs im Segmentierungspunkt
- Abbildung auf vordefinierten Griff der gleichen Klasse für Roboterhand
- Bestimmung eines optimalen Griffs in Simulation
- Ausführung auf Roboter



Klassifikation des menschlichen Griffs



Griffplanung in Simulation

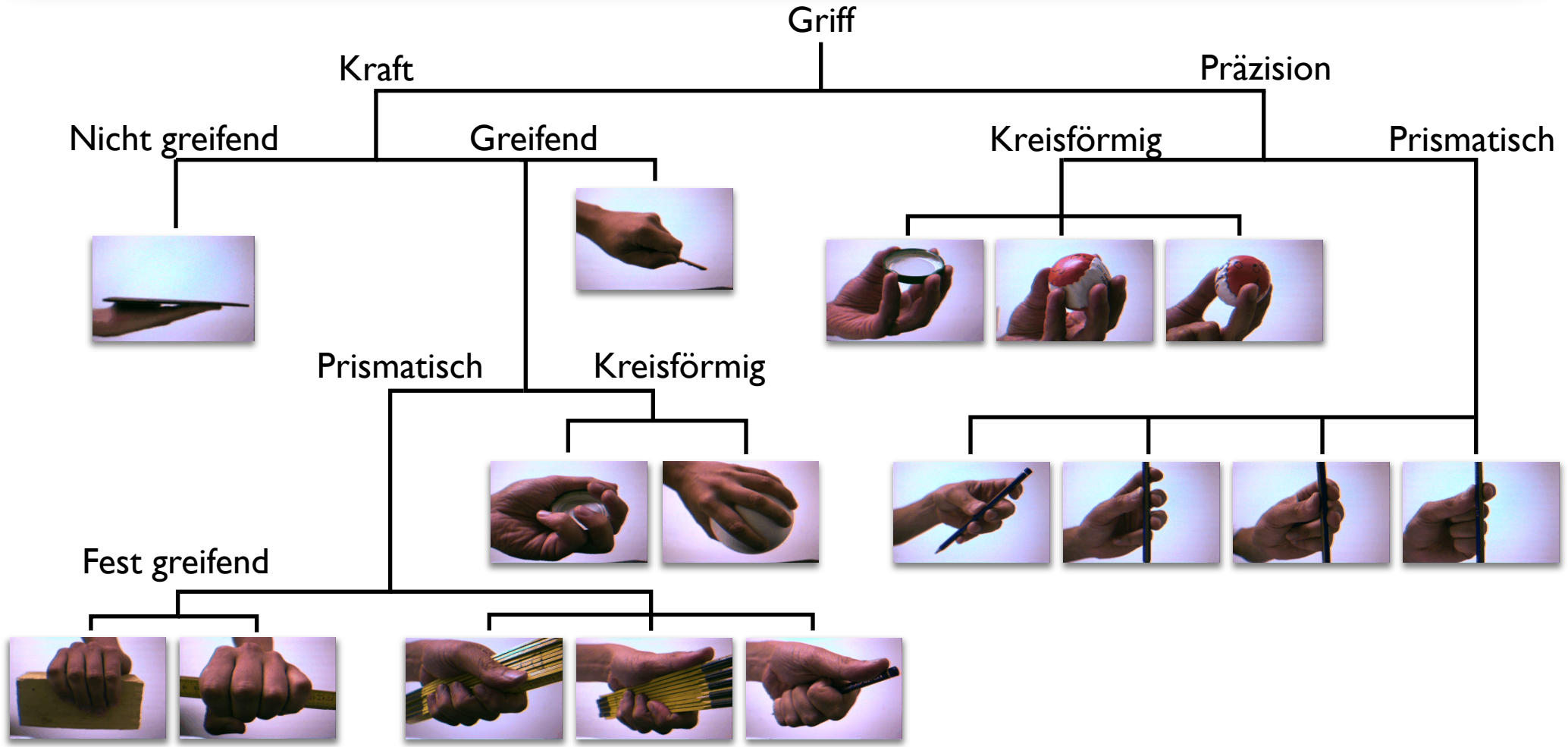
Übersicht

- Bahnplanung:
  - Definitionen
  - Simpler Rapidly-exploring Random Tree (RRT) Planer
  - Hinderniserweiterung
  - TC-RRT: Planung mit Task Constraints
- Griffklassifikation:
  - Cutkosky-Hierarchie
- Griffplanung:
  - Definitionen
  - Vorwärtsplaner: GraspIt!

Griffklassifikation: Problembeschreibung

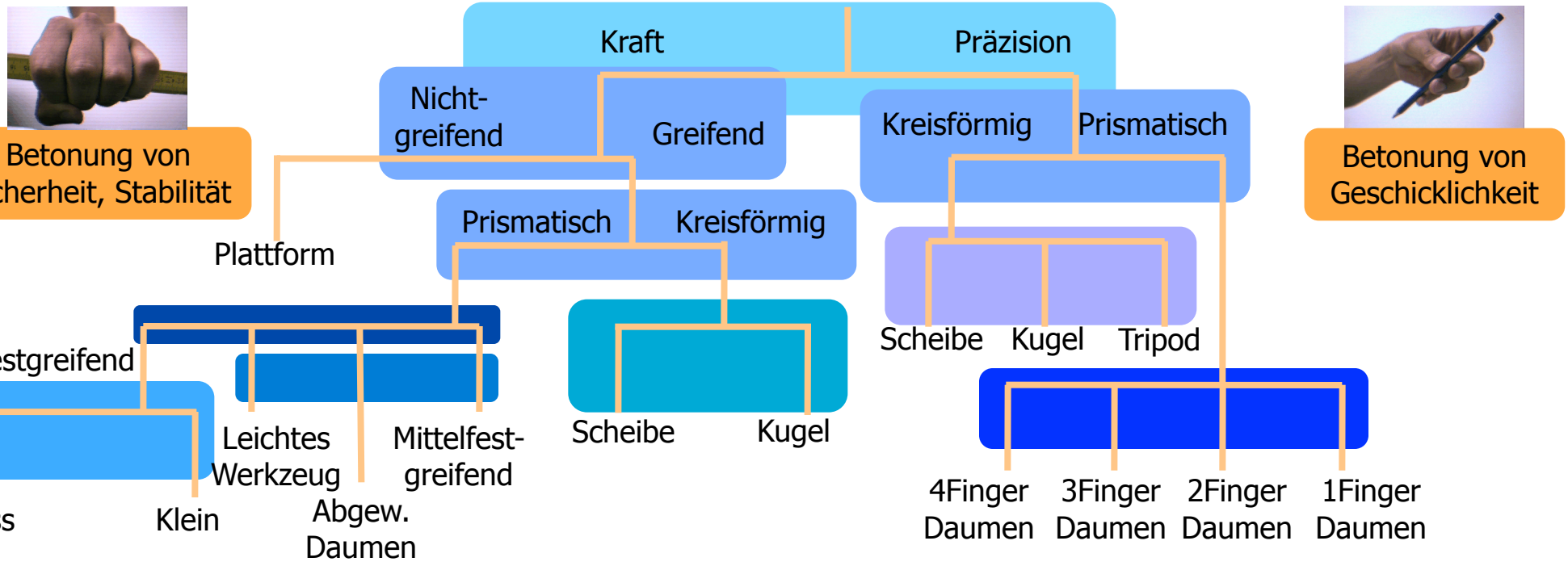
- Ziel: Klassifikation des menschlichen Griffs
- Ansatz:
  - Klassenhierarchie nach Cutkosky
  - Training einer Support Vector Machine in jeder Hierarchieebene (10)
  - Hierarchische Auswertung der Support Vector Machines

Griffklassifikation: Cutkosky Hierarchie



Griffklassifikation: Netztopologie

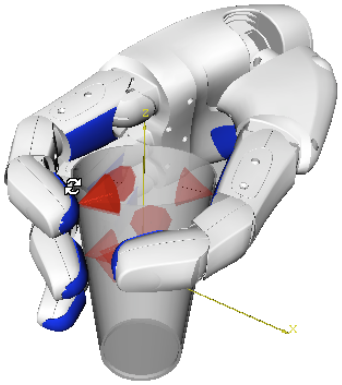
- Hierarchische Netztopologie reflektiert Griffklassifikation nach Cutkosky
- Jedes Netz wird mit entsprechenden Beispielen trainiert



Übersicht

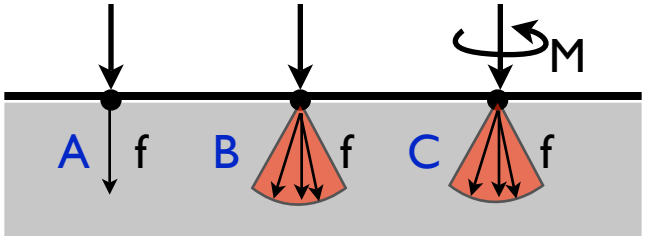
- Bahnplanung:
  - Definitionen
  - Simpler Rapidly-exploring Random Tree (RRT) Planer
  - Hinderniserweiterung
  - TC-RRT: Planung mit Task Constraints
- Griffklassifikation:
  - Cutkosky-Hierarchie
- Griffplanung:
  - Definitionen
  - Vorwärtsplaner: GraspIt!

Griffplanung: Übersicht

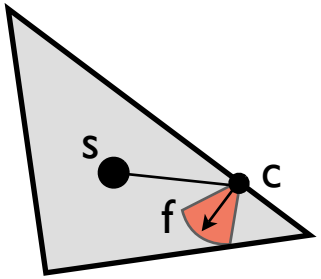


- Ziel: Berechnung eines Griffs, d.h. Menge von Kontaktstellen zwischen Roboter und Objekt

- Kontaktstellen:
  - Punktkontakt ohne Reibung (A)
  - Starrer Punktkontakt mit Reibung (B)
  - Nicht-starrer Punktkontakt mit Reibung („soft finger contact“) (C)
  - Flächenkontakte auf Basis von Punktkontakten



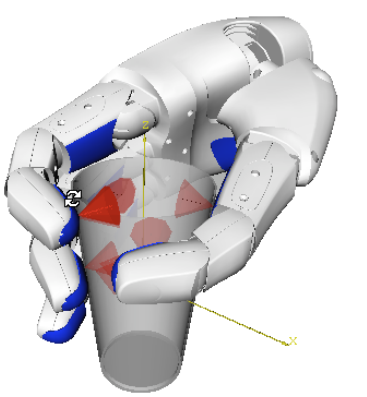
- Wirkung auf Objekt
  - Wrenchvektor: Kraft + Moment auf Objekt: $\begin{pmatrix} f \\ (c\text{-}s) \times f \end{pmatrix}$



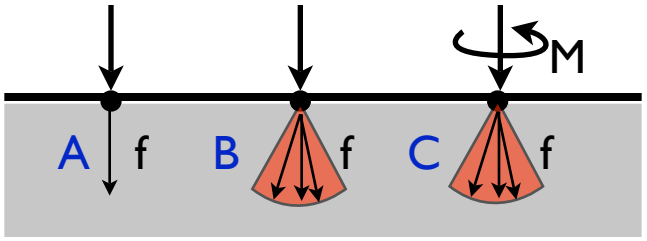
Griffplanung: Übersicht

geometrisches Objektmodell

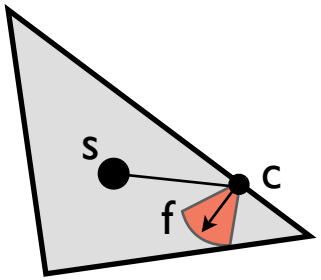
- Ziel: Berechnung eines Griffs, d.h. Menge von Kontaktstellen zwischen Roboter und Objekt



- Kontaktstellen:
  - Punktkontakt ohne Reibung (A)
  - Starrer Punktkontakt mit Reibung (B)
  - Nicht-starrer Punktkontakt mit Reibung („soft finger contact“) (C)
  - Flächenkontakte auf Basis von Punktkontakten

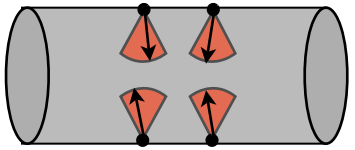


- Wirkung auf Objekt
  - Wrenchvektor: Kraft + Moment auf Objekt: $\begin{pmatrix} f \\ (c\text{-}s) \times f \end{pmatrix}$

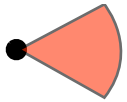


Griffplanung: Definitionen

- Beschreibung der Griffqualität:
  - Cone Wrench Space
  - Grasp Wrench Space
  - Task Wrench Space: aufgabenabhängig



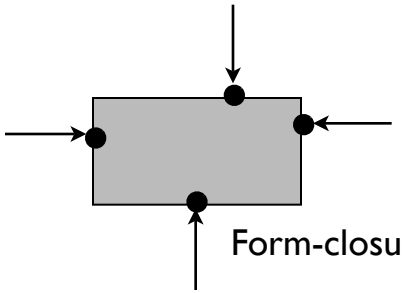
GWS: alle möglichen Summen aus jeweils einem Wrenchvektor jedes einzelnen Kegels



CWS: Kegel

TWS:
Knopf drücken

- Eigenschaften eines Griffs:
  - Widerstand gegen Stöße (beliebiger externer Wrench w):
    - Force-closure: -w liegt im GWS
    - Form-closure: geometrische Einschränkung
- Qualitätsmaße für Griffe (Grasp quality measure)
  - Beispiel: größte Hyperkugel um 0, die im GWS liegt

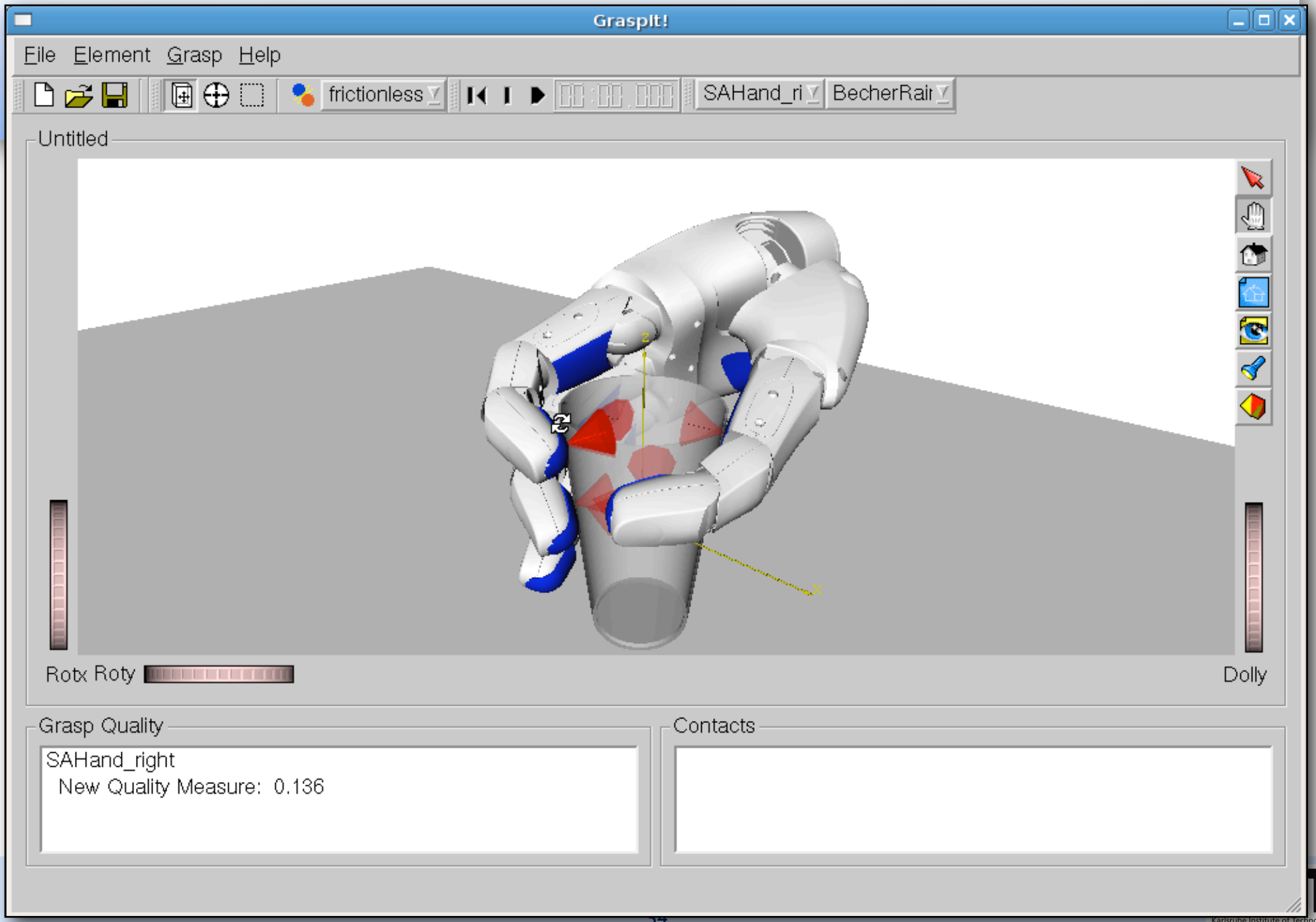


Form-closure

Wie werden Griffe berechnet?

Vorwärtsgriffplanung

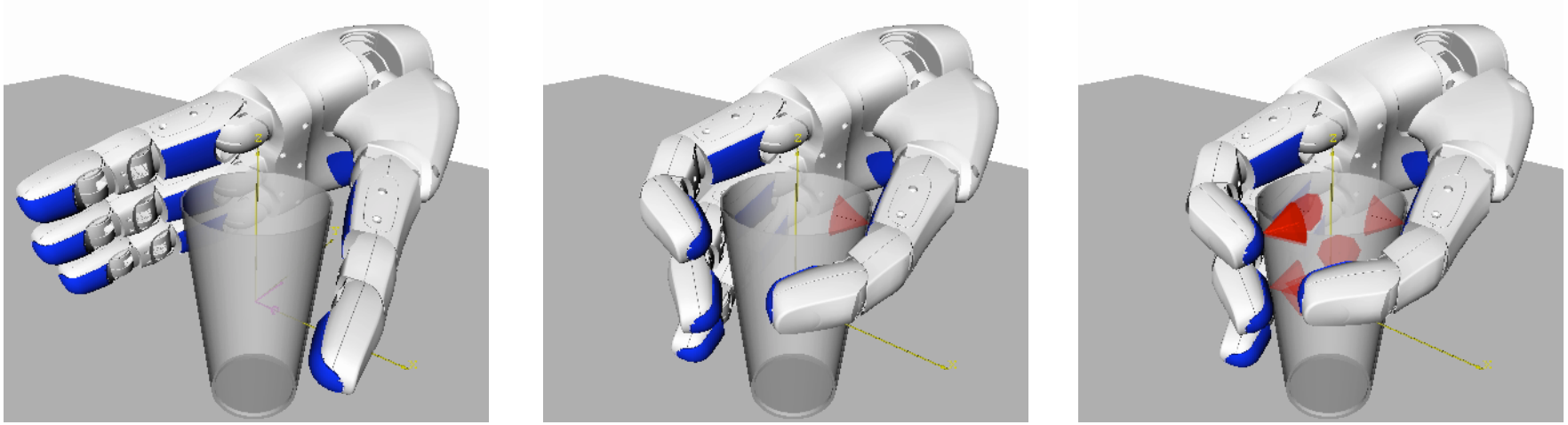
1. Setze Gelenkwinkel des Handmodells vor dem Zugreifen, z.B. prismatischer Kraftgriff
2. Setze 3d-Handmodell relativ zu Objekt vor dem Anrücken
3. Bewege Hand auf das Objekt zu
4. Schließe jeden einzelnen Finger bis auf Kontakt
5. Bestimme Kraftkegel in allen Kontaktpunkten
6. Berechne Griffqualität
7. * Iteriere bis Griff mit hoher Griffqualität gefunden


GraspIt!
File Element Grasp Help
frictionless
SAHand_ri
BecherRair
Untitled
Rotx Roty
Dolly
Grasp Quality
SAHand_right
New Quality Measure: 0.136
Contacts

Finger schließen bis Kontakt

Einfacher Algorithmus

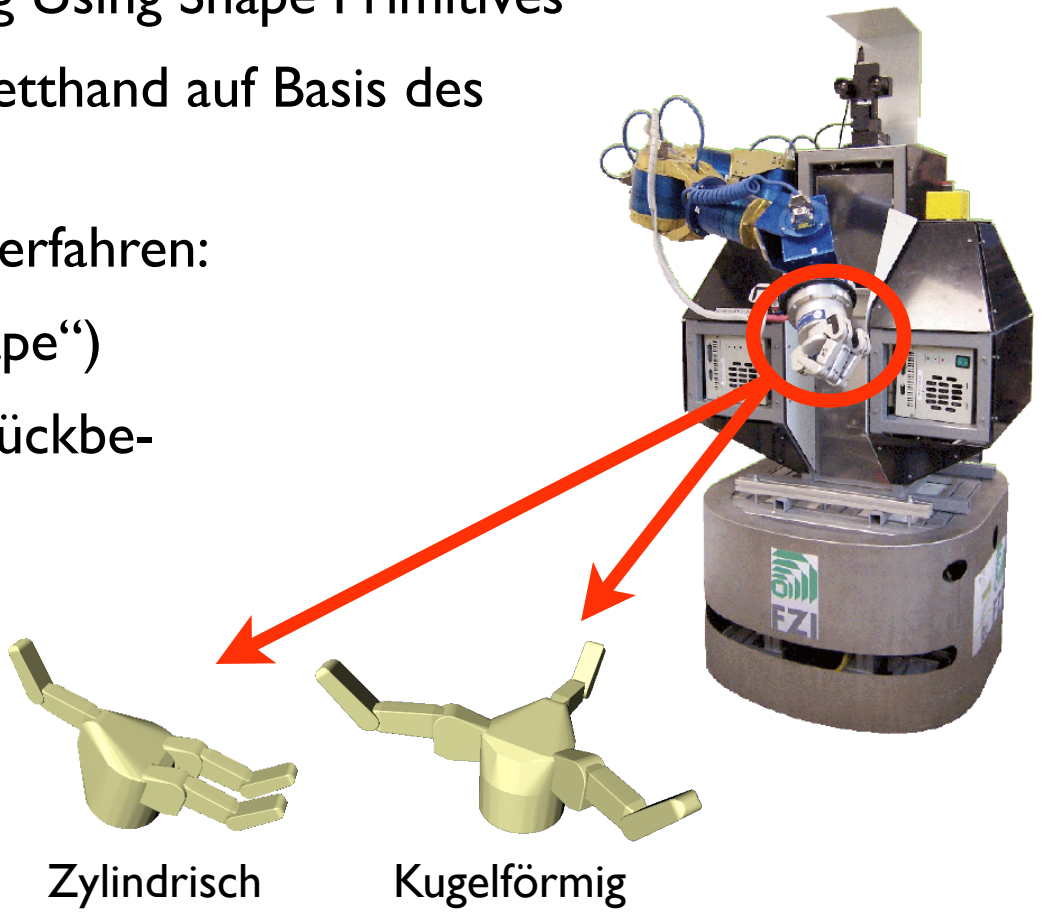
1. Schrittweise Änderung der Gelenkwinkel bis Hand komplett geschlossen
2. Überprüfung der Kollisionen in jedem Schritt → geom. Objektmodell
3. Bei Kollision: gebe vorherige Gelenkwinkel zurück



Spezialisierte Algorithmen basierend auf Distanz: C2A [Tang2009]

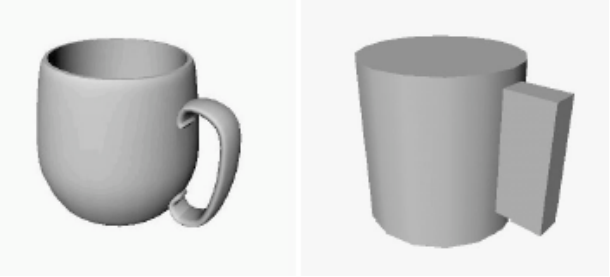
Greifplanungsverfahren mit GraspIt!

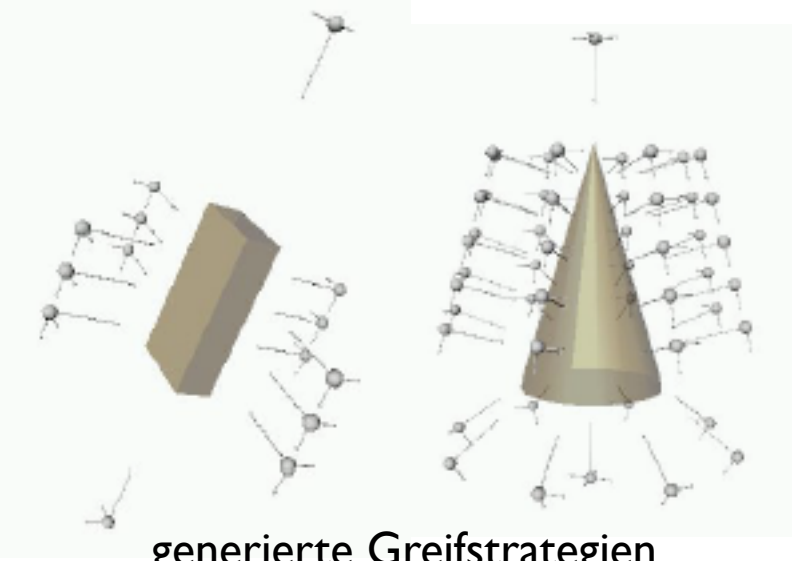
- Miller03: „Automatic Grasp Planning Using Shape Primitives“
- Greifplanungsverfahren für die Barretthand auf Basis des GraspIt! Simulators
- Vorwärtsgerichtetes Greifplanungsverfahren:
  - Vorgabe einer Griffform („preshape“)
  - Vorgabe einer Startpose und Anrückbewegung
  - Durchführung des Griffs
  - Evaluation mit Qualitätsmaß
- Zwei vordefinierte Preshapes:



Greifplanungsverfahren für die Barretthand

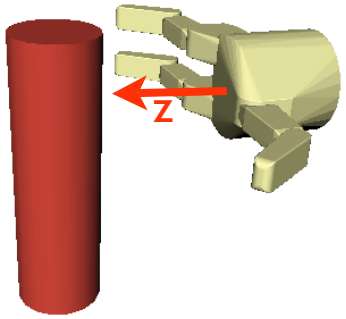
- Bestimmung der Greifstrategie (Preshape und Startlage) auf Basis einer Zerlegung des Objekts in Objektprimitive:
  - Kugeln, Zylinder, Kegel, Quader



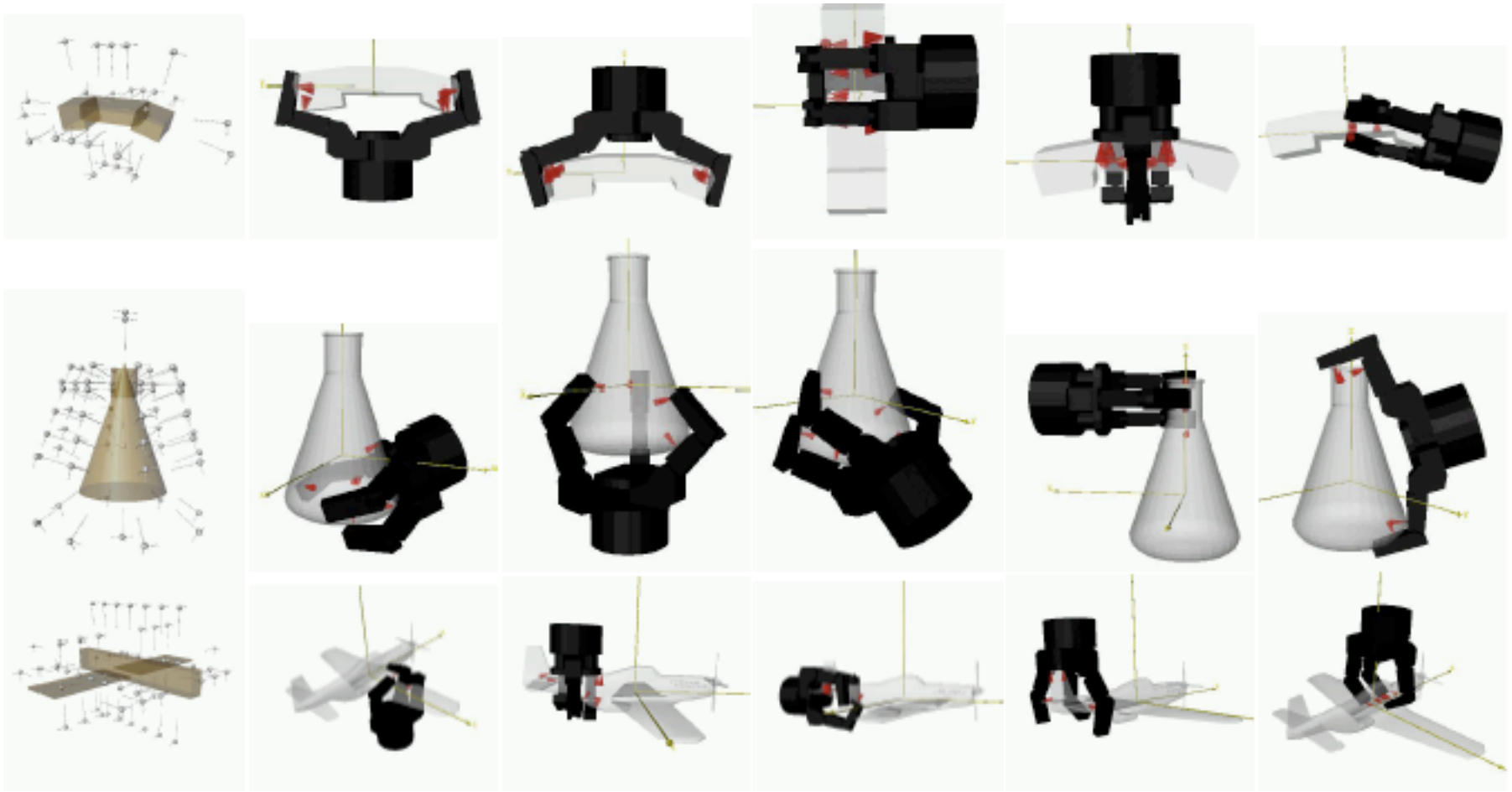


generierte Greifstrategien

- Vorgabe einer Menge von Greifstrategien für jedes Objektprimitive
- Anrückbewegung: linear in z-Richtung bis zu einem Kontakt und danach Backtracking



Greifplanungsverfahren für die Barretthand



PdV: Lernen von Griffen (1/3)

1. Beobachtung des Menschen

- Mensch demonstriert Griffbewegung
- Bestimmung der Griffform basierend auf Cutkosky-Hierarchie, z.B. sphärischer Präzisionsgriff
- Bestimmung der Anrückbewegung auf das Objekt

PdV: Lernen von Griffen (1/3)

1. Beobachtung des Menschen

- Mensch demonstriert Griffbewegung
- Bestimmung der Griffform basierend auf Cutkosky-Hierarchie, z.B. sphärischer Präzisionsgriff
- Bestimmung der Anrückbewegung auf das Objekt

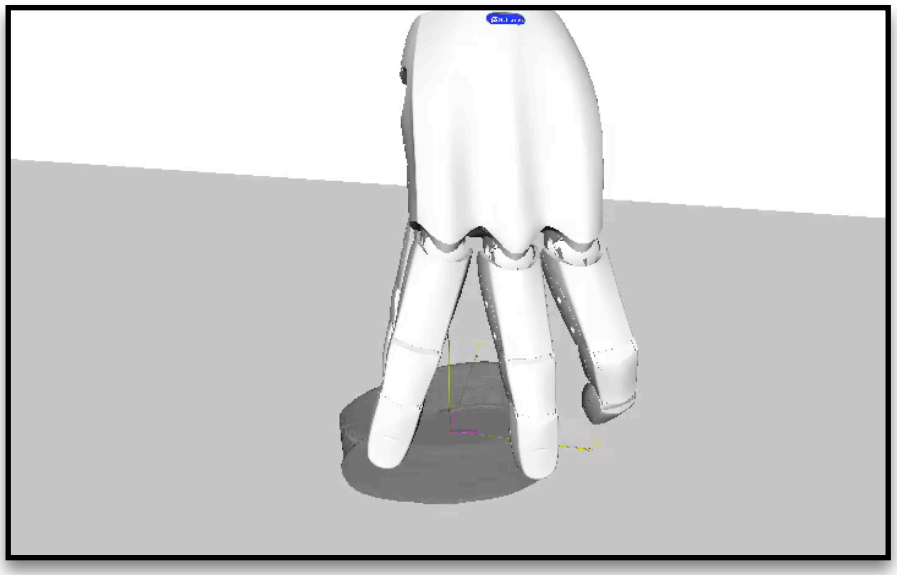
PdV: Lernen von Griffen (2/3)

2. (Vorwärtsgerichtete) Griffplanung zur Bestimmung von Kontaktstellen auf dem Objekt
   - Abbildung der menschlichen auf eine Roboter-Griffform
   - Abbildung der menschlichen auf eine Roboteranrückbewegung
   - Anrücken an Objekt und Schließen der Hand
   - Bestimmung der Kontaktstellen mit geometrischem Objektmodell
   - Berechnung des gewählten Qualitätsmaßes für Griffe

PdV: Lernen von Griffen (2/3)

2. (Vorwärtsgerichtete) Griffplanung zur Bestimmung von Kontaktstellen auf dem Objekt
   - Abbildung der menschlichen auf eine Roboter-Griffform
   - Abbildung der menschlichen auf eine Roboteranrückbewegung

   - Anrücken an Objekt und Schließen der Hand
   - Bestimmung der Kontaktstellen mit geometrischem Objektmodell
   - Berechnung des gewählten Qualitäts-maßes für Griffe



PdV: Lernen von Griffen (3/3)

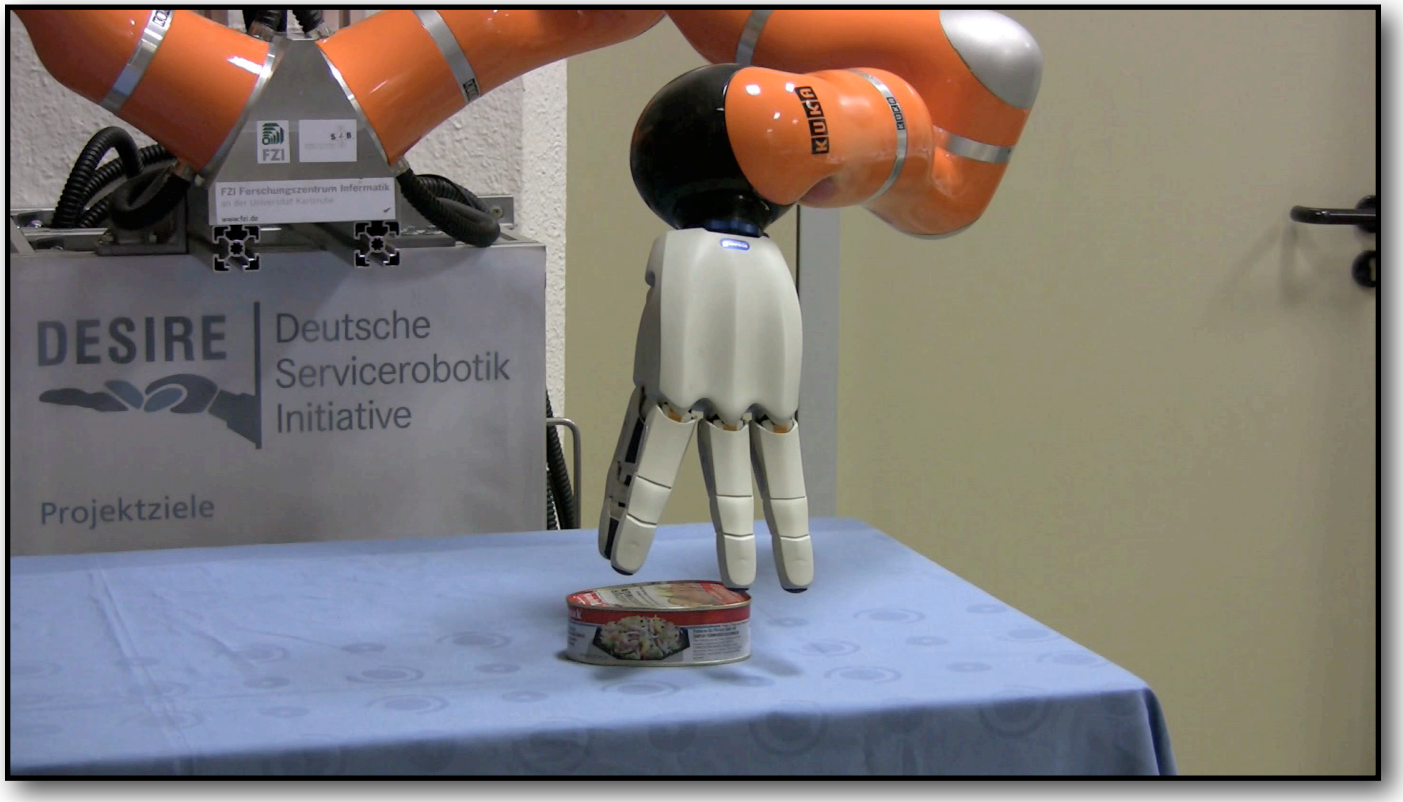
3. Ausführung auf dem Robotersystem

- Griffkraft manuell nachjustiert

PdV: Lernen von Griffen (3/3)

3. Ausführung auf dem Robotersystem

- Griffkraft manuell nachjustiert



Vielen Dank!