Ad-hoc Explorationsstrategie

- Zu ausgiebiges Erforschen bedeutet, daß der Agent auch nach langem Lernen noch ziellos im meist sehr großen Zustandsraum umherwandert. Dadurch werden auch Bereiche intensiv untersucht, die für die Lösung der Aufgabe gar nicht relevant sind.
- Zu frühes Ausbeuten der gelernten Approximation der Q-Funktion bewirkt möglicherweise, daß sich ein suboptimaler, d.h. längerer Pfad durch den Zustandsraum, der zufällig zuerst gefunden wurde, etabliert und die optimale Lösung nicht mehr gefunden wird.

Es existieren:

- "Greedy strategies"
- "randomized strategies"
- "interval-based techniques"

Beschleunigung des Lernens

Ad-hoc Techniken für

- Experience Replay
- Backstep Punishment
- Reward Distance Reduction
- Lerner-Kaskade

Experience Replay - I

Ein Pfad durch den Zustandsraum gilt als beendet, sobald G erreicht wurde.

Nun sei angenommen, im Zuge des Q-Learning werde dieser Pfad wiederholt durchlaufen.

Oftmals sind echt neue Lernschritte sehr viel kostenintensiver und/oder zeitaufwendiger als interne Wiederholungen bereits gespeicherter Lernschritte. Aus den genannten Gründen bietet es sich an, die Lernschritte abzuspeichern und intern zu wiederholen. Das Verfahren wird **Experience Replay** genannt.

Eine **Erfahrung** e (engl.: *experience*) ist ein Tupel

$$e = (s, s', a, r)$$

mit $s, s' \in S$, $a \in A$, $r \in \mathbb{R}$. e repräsentiert einen Lernschritt, d.h. einen Zustandsübergang, wobei s der Ausgangszustand, s' der Zielzustand, a die Aktion, die zu dem Zustandsübergang führte, und r das dabei erhaltene Reinforcement-Signal ist.

Experience Replay - II

Ein **Lernpfad** ist dann eine Serie $e_1 \ldots e_{L_k}$ von Erfahrungen (L_k ist die Länge des k-ten Lernpfades).

Der ER-Algorithmus:

```
for k = 1 to N
    for i = L_k to 1
        update(e_i aus Serie k)
    end for
end for
```

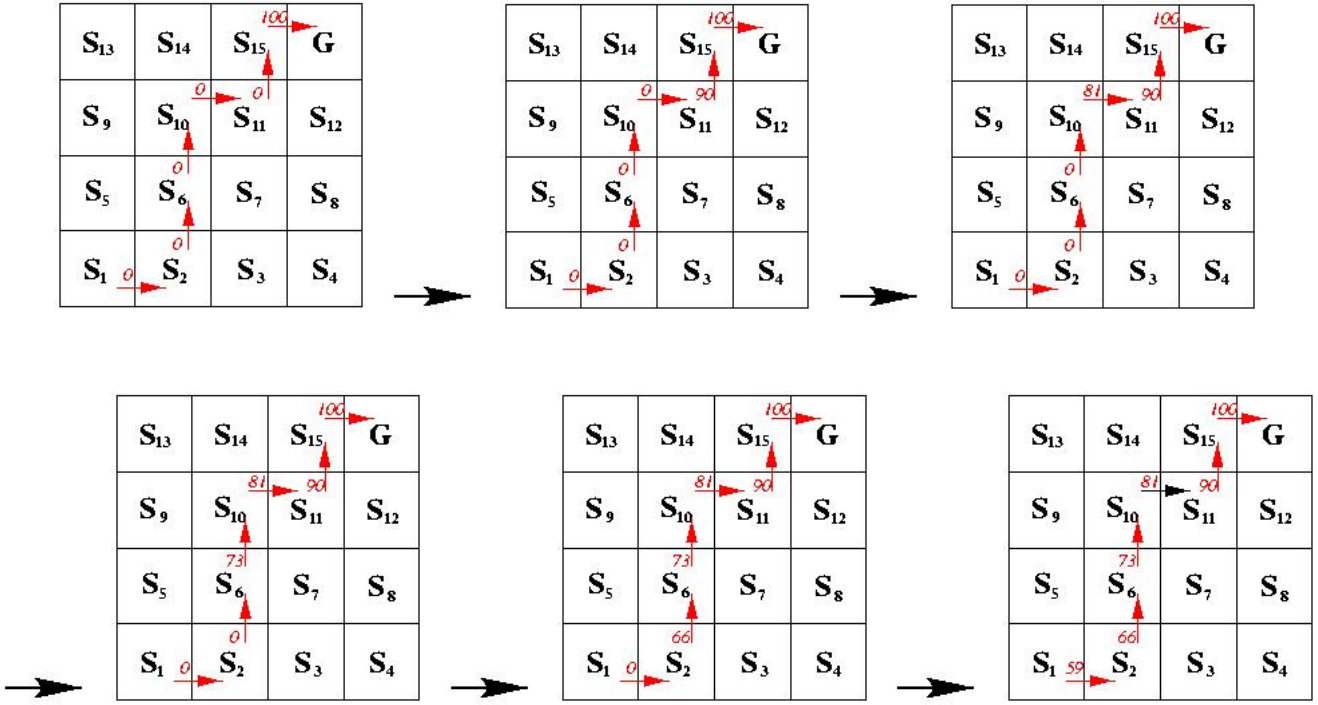
Experience Replay - III

Vorteile:

- Interne Wiederholungen gespeicherter Lernschritte verursachen meist weit weniger Kosten als echt neue Lernschritte.
- Intern läßt sich ein Lernpfad in umgekehrter Reihenfolge durchlaufen, was die Informationsausbreitung beschleunigt.
- Wenn sich Lernpfade kreuzen, können sie sozusagen „voneinander lernen“, d.h. Information austauschen. ER macht diesen Austausch unabhängig von der Reihenfolge, in der die Lernpfade erstmals ausgeführt wurden.

Experience Replay - Beispiel

Entwicklung der $\hat{Q}$-Werte bei wiederholtem Durchlaufen eines Lernpfades:



Backstep Punishment

Eine Explorationsstrategie wird benötigt, die für eine etwas „geradlinigere“ Bewegung des Agenten durch den Zustandsraum sorgt.

Dazu müssen vor allem Rückschritte vermieden werden.

Eine sinnvollere Methode scheint es zu sein, für den Fall, daß der Agent einen Rückschritt auswählt, ihn diesen ausführen zu lassen, aber ein *künstliches, negatives Reinforcement-Signal* zu generieren.

Kompromiß zwischen “Sackgasse-Vermeidung” und “schnellem Lernen”.

Beim zielorientierten Lernen könnte eine entsprechend erweiterte Reward-Funktion wie folgt aussehen:

$$r_{\mathrm{BP}} = \begin{cases} 100 & \text{falls Übergang in einen Zielzustand erfolgte} \\ -1 & \text{falls ein Rückschritt gemacht wurde} \\ 0 & \text{sonst} \end{cases}$$

Reward Distance Reduction

Die Reward-Funktion kann eine intelligentere Bewertung der Aktionen des Agenten vornehmen. Dies setzt aber Kenntnisse über die Struktur des Zustandsraumes voraus.

Wenn man zusätzlich die Kodierung des Zielzustandes kennt, dann kann es eine gute Strategie sein, die euklidische Distanz zwischen dem aktuellen Zustand und dem Zielzustand zu verringern.

Man kann die Rewardfunktion so erweitern, daß Aktionen, die die euklidische Distanz zum Zielzustand verringern, belohnt werden (**reward distance reduction**, **RDR**):

$$r_{\text{RDR}} = \begin{cases} 100 & \text{falls } \vec{s}' = \vec{s}_g \\ 50 & \text{falls } |\vec{s}' - \vec{s}_g| < |\vec{s} - \vec{s}_g| \\ 0 & \text{sonst} \end{cases}$$

wobei $\vec{s}$, $\vec{s}'$ und $\vec{s}_g$ die den Ausgangszustand, den Endzustand und den Zielzustand kodierenden Vektoren sind.

Lerner-Kaskade - I

Es ist für die Genauigkeit der Positionierung entscheidend, wie fein der Zustandsraum eingeteilt ist.

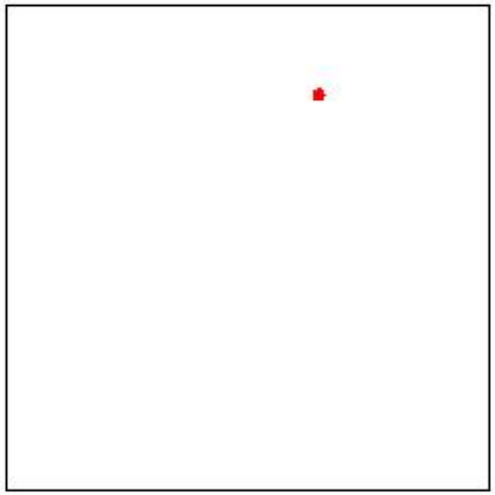
Andererseits steigt mit wachsender Feinheit der Diskretisierung auch die Anzahl der Zustände und damit der Aufwand für das Lernen.

Man muß sich vor dem Lernen für eine Diskretisierung entscheiden und dabei abwägen zwischen Lernaufwand und Genauigkeit der Positionierung.

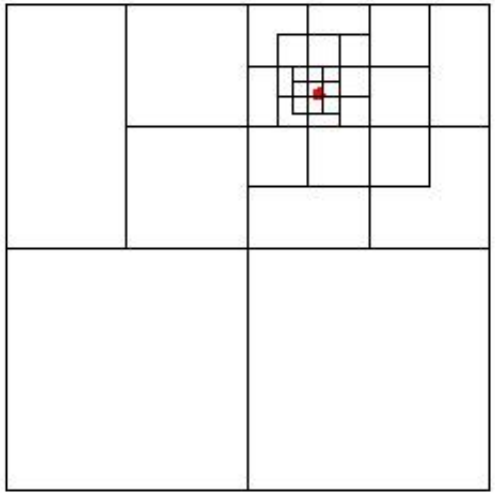
Lerner-Kaskade - Variable Diskretisierung

Ein Beispiel-Zustandsraum a) ohne Diskretisierung b) mit variabler Diskretisierung

a)



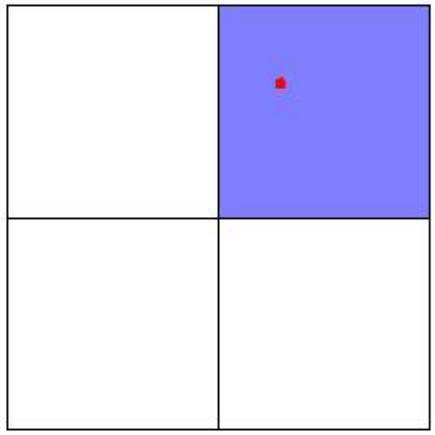
b)

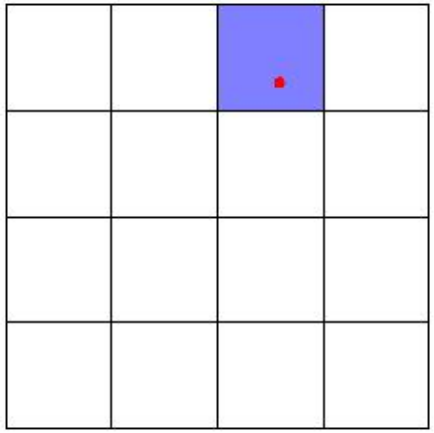


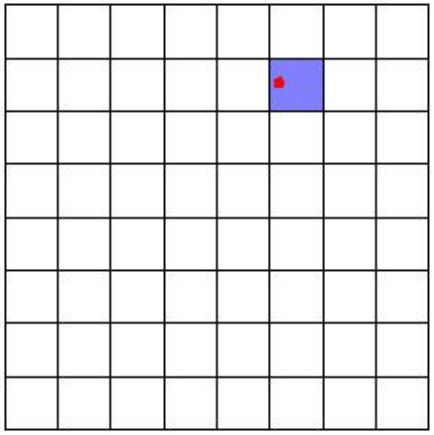
Dies setzt aber Kenntnisse über die Struktur des Zustandsraumes voraus.

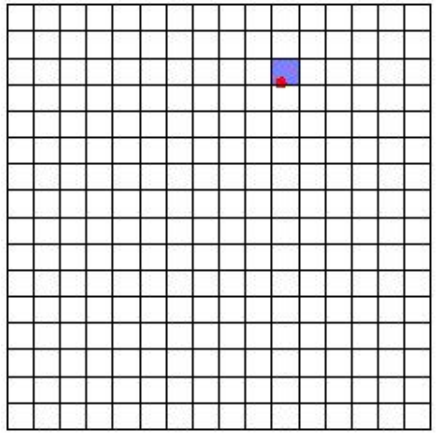
Lerner-Kaskade - n-stufige Lerner-Kaskade

Einteilungen des Zustandsraumes einer vierstufigen Lerner-Kaskade:

1. Lerner: 

2. Lerner: 

3. Lerner: 

4. Lerner: 

Bahnplanung mit DP - I

Zum finden des Pfades zwischen Start- und Zielkonfiguration wird die Policy Iteration benutzt.

Folgende Größen müßen dementsprechend beim vorliegenden Planungsproblem definiert werden:

- Als Zustandsmenge S dient der diskrete Konfigurationsraum, dh. jedes mögliche Gelenkwinkeltupel $(\theta_1, \theta_2, \ldots, \theta_{Dim})$ ist genau ein Zustand der Menge S mit Ausnahme der Zielkonfiguration

- Die Menge $A(s)$ aller möglichen Aktionen für einen Zustand s sind die Übergänge zu Nachbarkonfigurationen im K-Raum, also für einen oder mehrere Gelenkwinkel θ_i die Änderung um $\pm Dis$. Dabei sind nur Aktionen a in $A(s)$ enthalten, die nicht zu K-Hindernissen führen, dh. die nicht mit Hindernissen kollidieren, und nicht die Grenzen des K-Raumes verletzen

Bahnplanung mit DP - II

- Für die Rewardfunktion $R^a_{ss'}$ gilt:

  $R^a_{ss'} =$ **Reward_nicht_im_Ziel** $\forall s \in S, a \in A(s), s' \in S$ und
  $R^a_{ss'} =$ **Reward_im_Ziel** $\forall s \in S, a \in A(s), s' = s_t$.

  Also nur für das Erreichen des Zielzustandes wird ein anderer Rewardwert geliefert. Für alle anderen Zustände gibt es einen einheitlichen Betrag

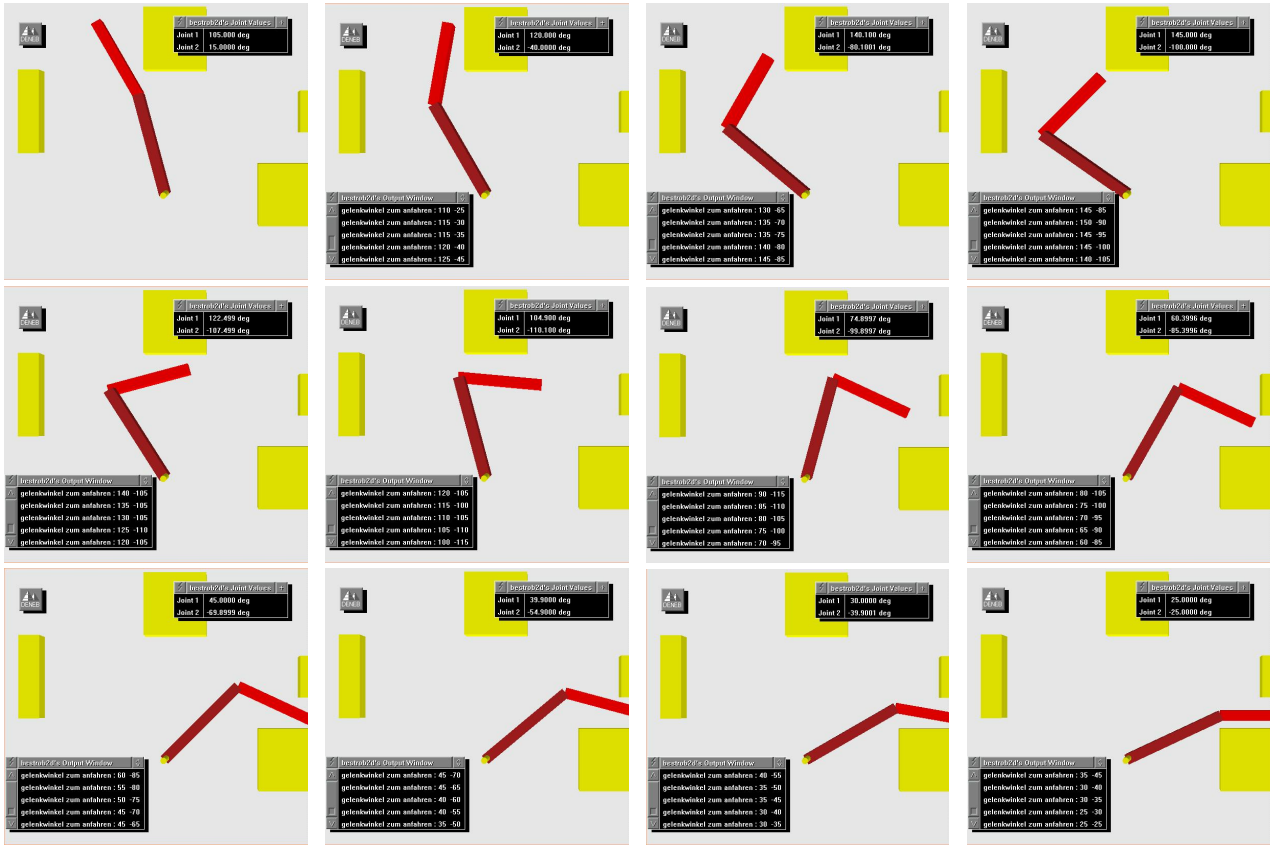
- Die Policy $\pi(s, a)$ sei ebenfalls deterministisch, dh. es gibt genau ein a mit $\pi(s, a) = 1$

- Eine Schranke Θ, bei der die Policy Evaluation beendet wird, muss gewählt werden

- Für das Problem ist das Infinite-Horizon Discounted Model am sinnvollsten, deshalb muss γ entsprechend gesetzt werden

2-gelenkiger Roboter

Die gefundene Bewegungssequenz des 2-gelenkigen Roboters (von links nach rechts, von oben nach unten):



Grasping mit RL: Aufteilung der DOFs

Üblicherweise braucht ein Roboterarm 6 DOFs um ein Objekt aus irgendeiner Position und Orientierung zu greifen.

Um quasi-planare Objekte zu greifen nehmen wir an, dass der Greifer senkrecht zum Tisch steht und dass sein Gewicht bekannt ist.

Es bleiben immer noch drei DOFs zur Kontrolle des Roboterarms: Bewegungen parallel zur Tischebene (x, y) und die Rotation um die senkrecht zum Tisch stehende Achse (θ).

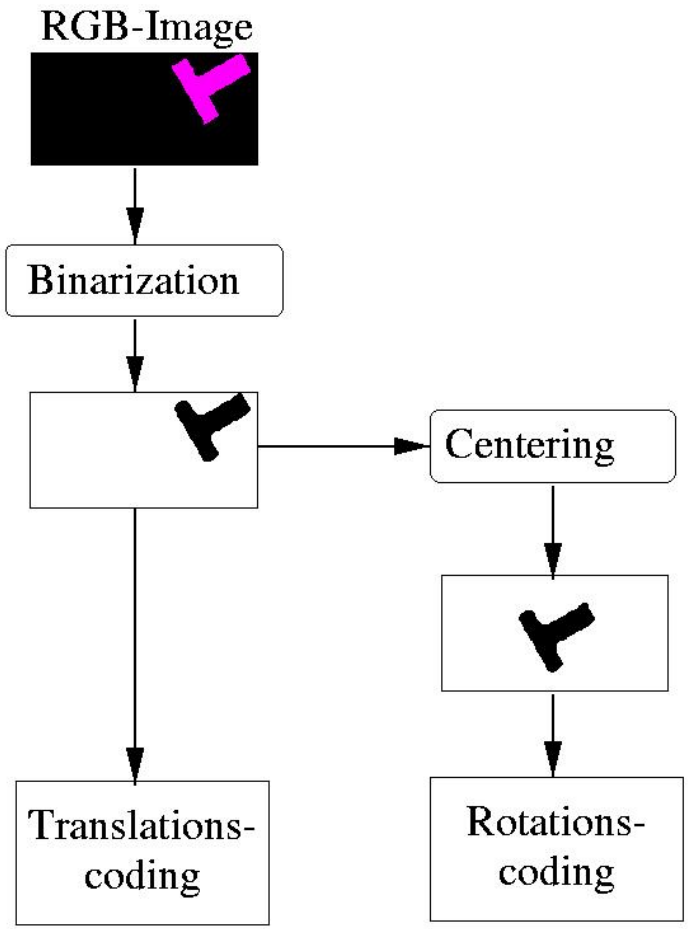


Abbildung 1: Controlling x, y, θ. Die vor-prozessierten Bilder werden für die Rotations-Kodierung zusätzlich zentriert.

Um einen kleinen $Zustandsraum$ zu erzeugen, wird das Lernen zwischen zwei Lernern aufgeteilt:

einer für die $translationale$ Bewegung auf der Ebene,

der andere für die Rotationsbewegung.

Der $Translations\text{-}Lerner$ kann dann vier Aktionen auswählen (zwei in x- und zwei in y-Richtung).

Der $Rotations\text{-}Lerner$ kann zwei Aktionen auswählen (Rotation im und gegen den Uhrzeigersinn).

Diese Aufteilung hat folgende Vorteile gegenüber einem monolithischem Lerner:

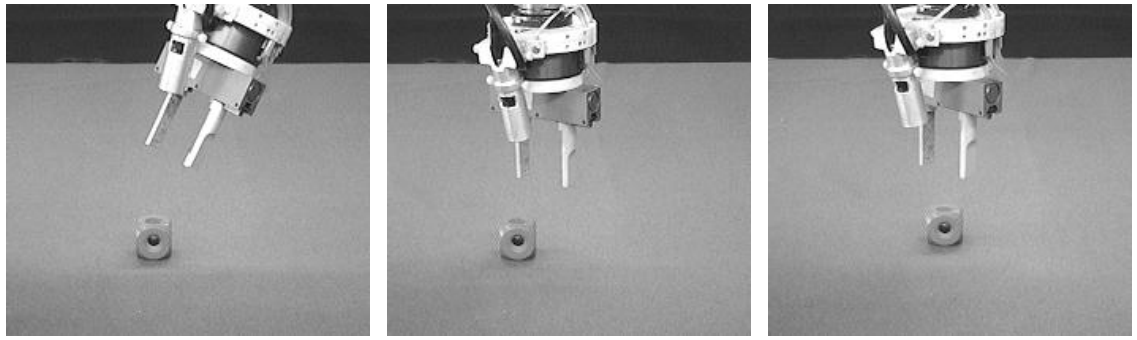
- Der Zustandsraum ist wesentlich kleiner.
- Die Zustandskodierung ist so entworfen, dass die Zustandsvektoren nur die relevanten Informationen für den entsprechenden Lerner enthalten.

In der Praxis werden die beiden Lerner abwechselnd angewandt.

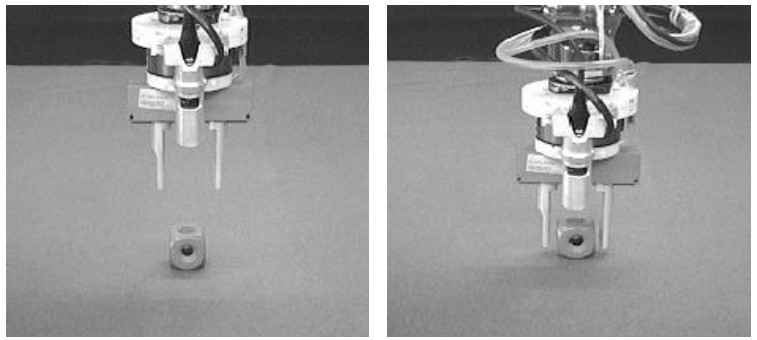
Zuerst wird der Translations-Lerner in langen Lern-Schritten angewandt, bis er das in seiner Zustands-Kodierung definierte Ziel erreicht hat.

Dann wird der Translations-Lerner durch den Rotations-Lerner ersetzt, der ebenfalls in langen Lern-Schritten angewandt wird bis sein Ziel erreicht ist.

Zu diesem Zeitpunkt kann es passieren, dass der Translations-Zielzustand vom Rotations-Lerner gestört wird. Daher wird der Translations-Lerner noch einmal aktiviert. Diese Prozeduren werden wiederholt, bis beide Lerner berichten, dass sie alle den Zielzustand erreichen. Dieser Zustand ist daher der gemeinsame Zielzustand.



(a) (b) (c)



(d) (e)

Abbildung 2: Positionierung und Orientierungs-Kontrolle mit 6 DOFs in vier Schritten.

Um alle sechs DOFs zu benutzen sollten zusätzliche Lerner eingeführt werden.

1. Der erste Lerner besitzt zwei DOFs und seine Aufgabe ist dass das Objekt von einer bestimmten Perspektive aus betrachtet wird. Für einen ebenen Tisch bedeutet dies typischerweise, dass der Greifer senkrecht zur Tischoberfläche positioniert werden muss (a $\rightarrow$ b).

2. Wende den x/y Lerner an (b $\rightarrow$ c).

3. Wende den θ Rotations-Lerner an (c $\rightarrow$ d).

4. Der letzte Höhen-Lerner wird die z-Koordinate korrigieren, die Höhe über dem Tisch (d $\rightarrow$ e).

Visuell geführtes Greifen mittels selbstbewertendem Lernen

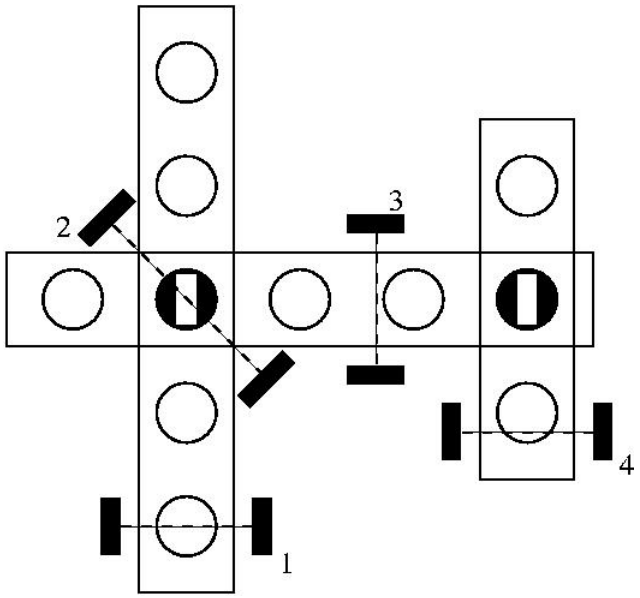
Griff ist optimal bzgl. lokaler Kriterien:

- Die Finger des Greifers können das Objekt am Greifpunkt umschließen
- Keine Reibung tritt zwischen Fingern und Objekt auf

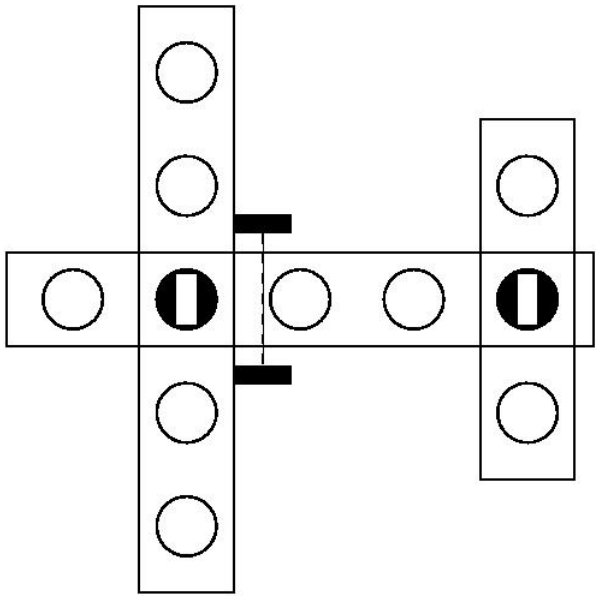
Griff ist optimal bzgl. globaler Kriterien:

- Kein bzw. minimales Drehmoment an den Fingern
- Objekt rutscht nicht aus dem Greifer
- Der Griff ist stabil, d.h. Objekt sitzt fest zwischen den Fingern

Bsp. Lokale und globale Kriterien



(a)



(b)

Zwei Ansätze

Ein Lerner:

- Die Zustände bestehen aus einem Satz $m + n$ lokaler und globaler Eigenschaften: $s = (f_{l_1}, \ldots, f_{l_m}, f_{g_1}, \ldots, f_{g_n})$.
- Der Lerner versucht sie auf die Aktionen $a = (x, y, \phi)$ abzubilden, wo x und y in Richtung x und y translationale Komponenten sind und ϕ die Rotation um den Annäherungsvektor des Greifers ist.

Zwei Lerner:

- Die Zustände für den ersten Lerner liefern nur die lokalen Eignschaften $s = (f_{l_1}, \ldots, f_{l_m})$.
- Der Lerner versucht sie auf Aktionen abzubilden, die nur aus der Rotationskomponente $a = (\phi)$ bestehen.
- Der zweite Lerner versucht, Zustände globaler Eigenschaften $s = (f_{g_1}, \ldots, f_{g_n})$ auf Aktionen translationaler Komponenten abzubilden: $a = (x, y)$.

Aufbau

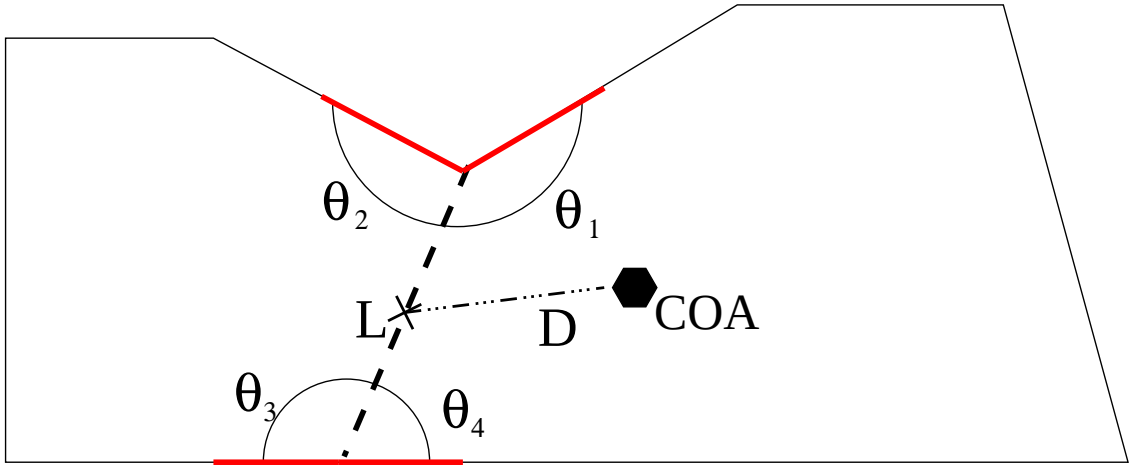
- Zweikomponentiges Lernsystem:

1. Orientierungs-Lerner	**2. Orts-Lerner**
Operiert auf lokalen Kriterien	Operiert auf globalen Kriterien
Gleich für jedes Objekt	Unterschiedlich für jedes neue Objekt

- Einsatz von Multisensorik:
  - Kamera
  - Kraft-Moment-Sensor

- Beide Lerner arbeiten aufeinanderfolgend im Perzeptions-Aktions-Zyklus

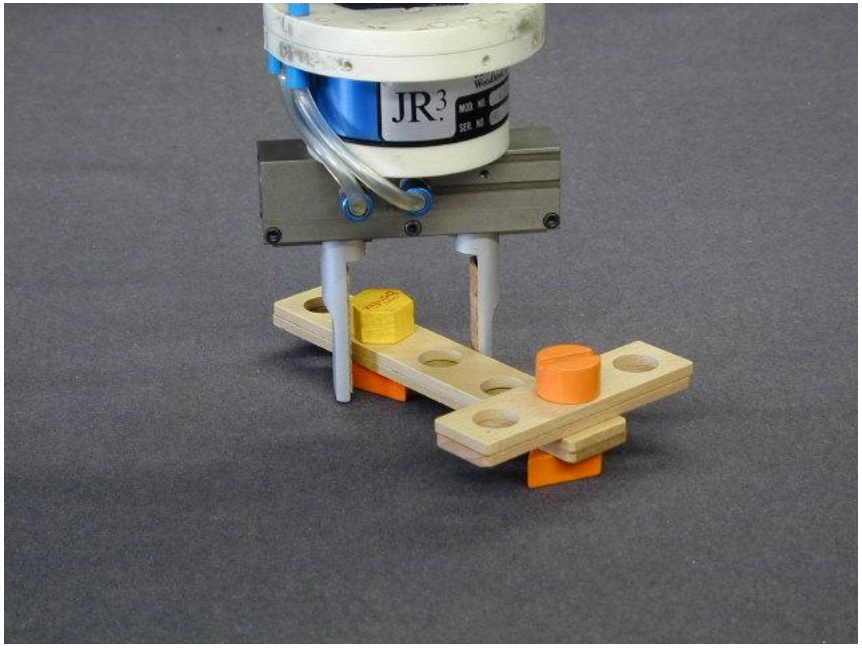
Zustandskodierung

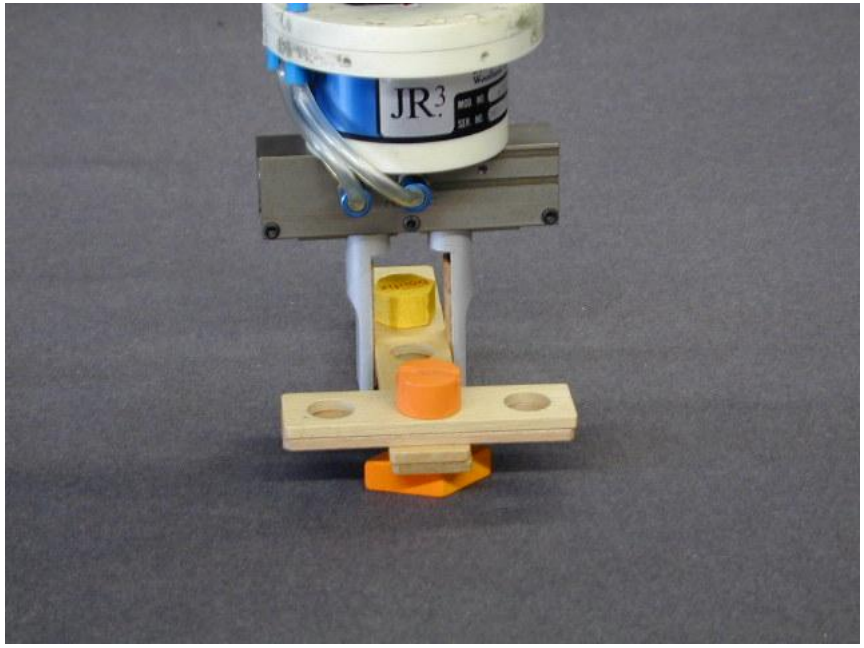


Der Orientierungs-Lerner benutzt Länge L sowie Winkel $\Theta_1, \ldots, \Theta_4$ wärend der Orts-Lerner die Distanz D zwischen Mittelpunkt der Greiflinie und Bildschwerpunkt des Objektes nutzt.

Selbstbewertungsmaßnahmen im Orientierungs-Lerner

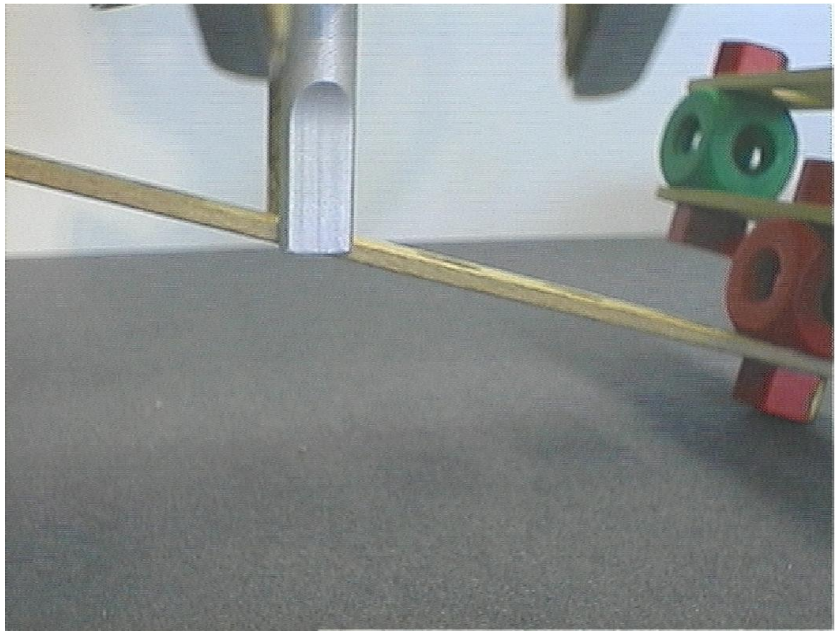
Visuelle Bewertung des Grifferfolges:
Reibung resultiert in Rotation oder Versatz des Objektes

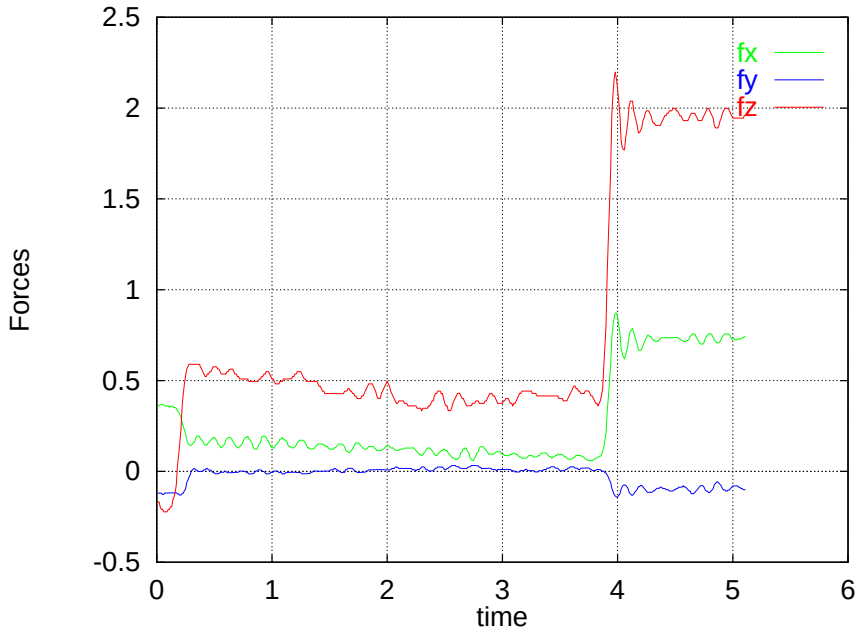




Selbstbewertungsmaßnahmen im Orts-Lerner I

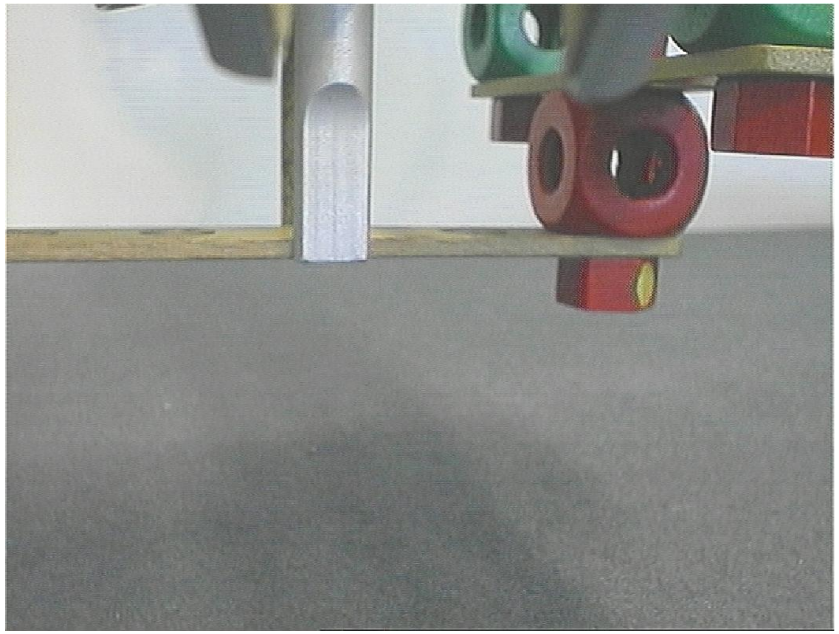
Bewertung mittels Kraftmomenten-Sensors:
Instabiler Griff – analysiert durch Kraftmessung

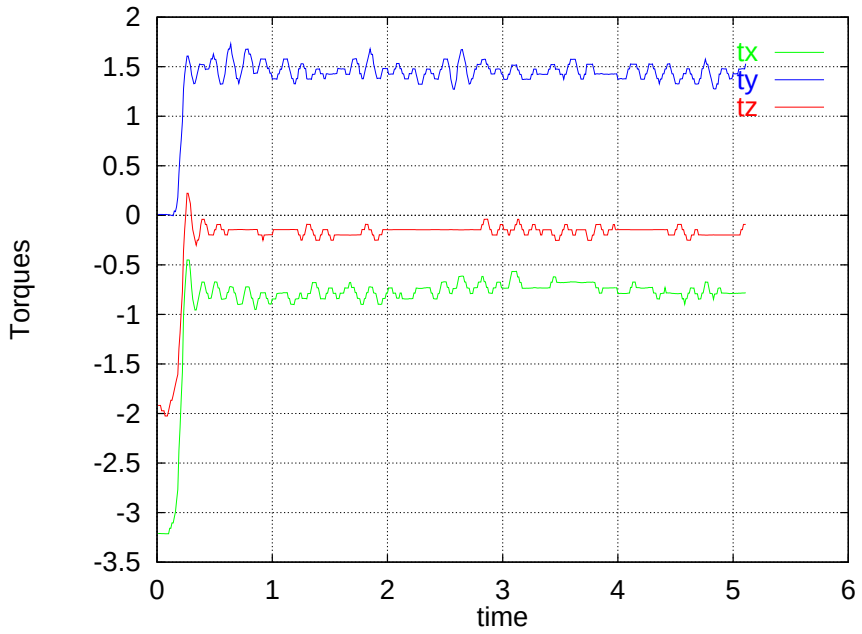




Selbstbewertungsmaßnahmen im Orts-Lerner II

Suboptimaler Griff – analysiert über Drehmomente

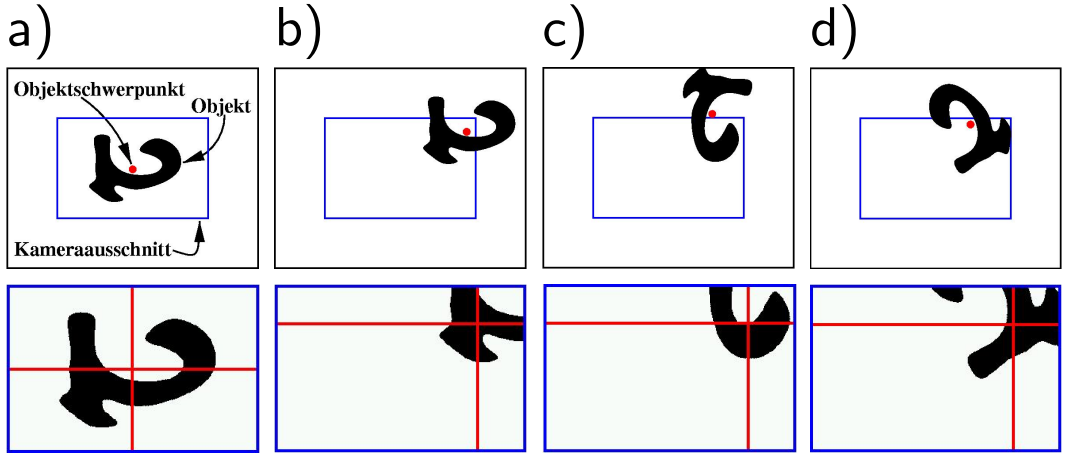




Das Problem der unvollständigen Zustandsinformation

Man spricht auch von **verborgenen Zuständen** (engl.: *hidden states*).

Beispiel für unvollständige Zustandsinformation:



Q-Lernen - offene Fragen

- oft nicht vorhanden: Markov-Annahme, Beobachbarkeit
- kontinuierliche Zustand-Aktion-Räume
- Generalisierung über Zustand und Aktion
- Kompromiß zwischen “Exploration” und “Exploitation”
- Generalisierung der automatischen Bewertung (Kredit-Vergabe)