Probabilistische Entscheidungsverfahren II

Wiederholung: MDPs

- Markov Decision Processes sind ein theoretisches Rahmenwerk zur Modellierung von Entscheidungsfindung in
  - Dynamischen
  - Sequentiellen
  - Diskreten (oder auch kontinuierlichen)
  - Stochastischen
  - Vollständig beobachtbaren

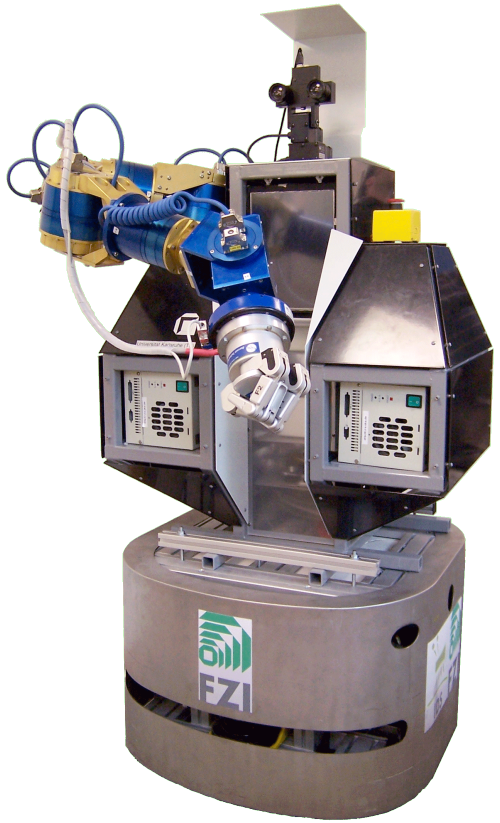
  Domänen

Wiederholung: MDPs Value Iteration

- Mittels MDP Value Iteration können optimale bzw. annähernd optimale Entscheidungsfunktionen berechnet werden
- Ein rationaler Agent kombiniert Risikoabschätzung und Gewinnmaximierung in einem Szenario mittels seiner Entscheidungsfunktion (*policy*)

Szenario Serviceroboter

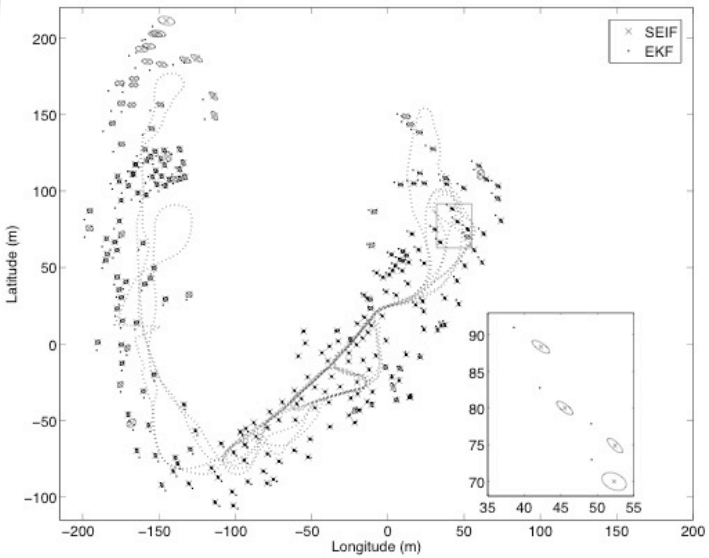
- Dynamisch
- Sequentiell
- Kontinuierlich
- Stochastisch
- ***Unvollständig*** beobachtbar
- Multiagent

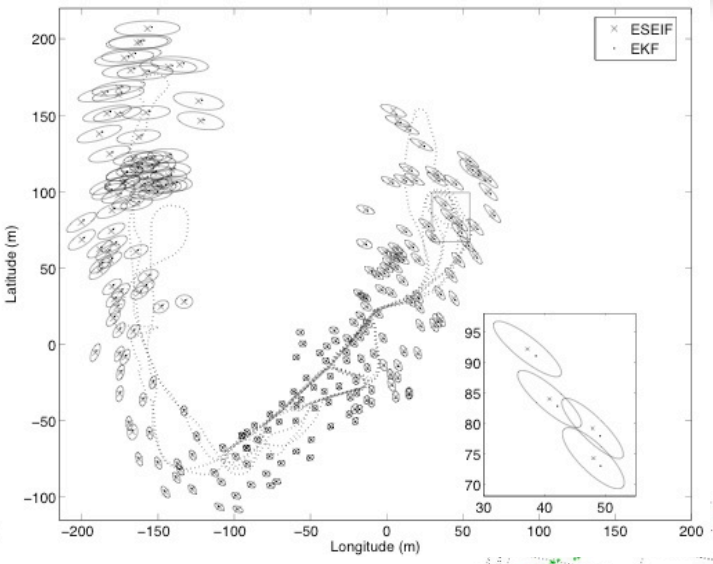


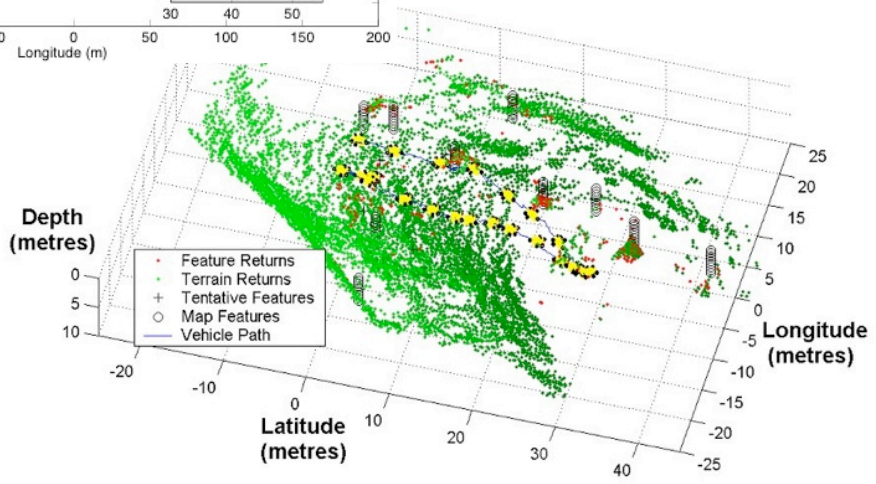
Eine unvollständig beobachtbare Umwelt

- Die Eigenschaft *unvollständig beobachtbar* wird nicht von reinen MDPs modelliert
- Bei autonomen Robotern ist diese Eigenschaft der Umwelt stark ausgeprägt
- Die Perzeption ist üblicherweise verrauscht und deckt nur einen kleinen Teil des Szenarios ab (z.B. Verdeckungen)

Beispiel verrauschte Perzeption: SLAM

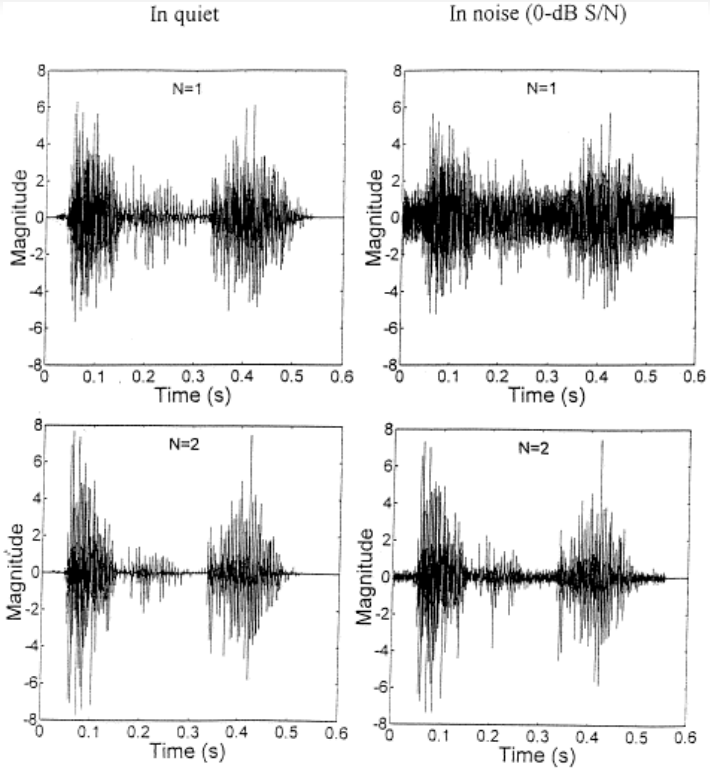


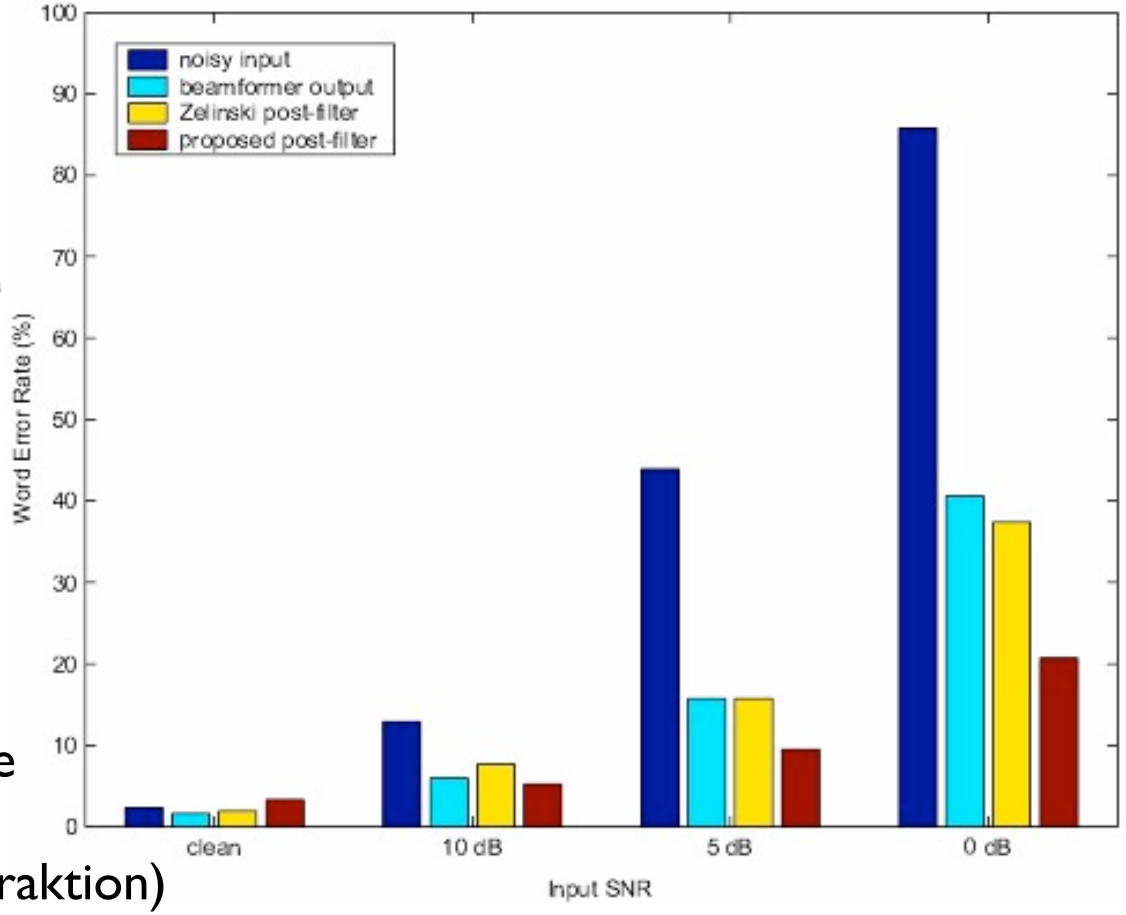




Simultaneous Localization And Mapping (SLAM) ist eine zentrale Fähigkeit für autonome, mobile Roboter, also auch Serviceroboter

Beispiel verrauschte Perzeption: Sprache





Spracherkennung ist eine zentrale Fähigkeit für Serviceroboter (natürliche Mensch-Roboter-Interaktion)

Zustandsschätzung: Mehr als Perzeption

- Die Schätzung des aktuellen Zustands muss nicht nur auf der Perzeption basieren
- Grundprinzip rationaler Agent in Domäne mit Markov Eigenschaft:
  - Aktueller Zustand kann über unvollkommene Perzeption **beobachtet** werden (*measurement*)
  - Aktueller Zustand kann auf Basis des vorherigen Zustands und der ausgeführten Handlung **vorhergesagt** werden (*prediction*)

Zustandsschätzung: Beobachtung

- Die Beobachtung wird durch ein auf die Eigenschaften des physischen oder logischen Sensors abgestimmtes Unsicherheitsmodell abgebildet

- Die Unsicherheit kann so explizit, numerisch angegeben werden

- Diese Unsicherheit kann verschieden modelliert werden:
  - kontinuierlich oder diskret
  - parametrisch oder nicht-parametrisch

Zustandsschätzung: Vorhersage

- Die Vorhersage wird über ein Transitionsmodell realisiert
- Das Transitionsmodell führt zu einer Wahrscheinlichkeitsverteilung über allen Zuständen wie im MDP
- Eine Vorhersage funktioniert nur iterativ, ausgehend von einem aktuellen Zustand (bzw. begrenzten Historie: Markov Eigenschaft)

Vorhersage + Beobachtung: Bayes Filter

- Vorhersage und Beobachtung können nun fusioniert werden, um zusammen eine höhere Genauigkeit zu erreichen:

Bayes Filter

Bayes Filter: Idee

- Eine gute Vorhersage kann eine sehr ungenaue Beobachtung verbessern
- Eine gute Beobachtung präzisiert jedoch eine ungenaue Vorhersage
- Da die Vorhersage nur iterativ funktioniert, arbeitet der Bayes Filter ebenfalls iterativ von Zeit t-1 nach t

Bayes Filter: Zustandsvermutung

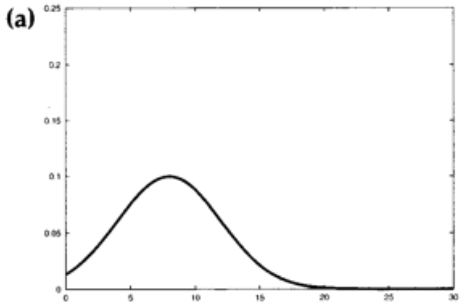
- Die Ausgabe einer Vorhersage und auch einer Beobachtung ist eine Wahrscheinlichkeitsverteilung
- Da der Bayes Filter iterativ läuft, ist die Eingabe also auch eine Wahrscheinlichkeitsverteilung
- Der Agent besitzt daher zu jedem Zeitpunkt eine **Zustandsvermutung** (*belief*)

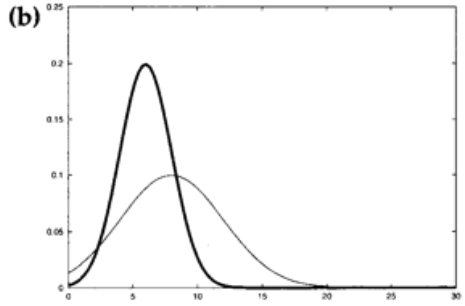
Zustandsvermutung: Repräsentation

- Die Zustandsvermutung repräsentiert die subjektive Wahrnehmung des Agenten bezüglich der nur unvollständig beobachtbaren Welt (Zustandsraum)
- Die Zustandsvermutung ist eine
  - diskrete Wahrscheinlichkeitsverteilung über einem diskreten Zustandsraum
  - Oder eine kontinuierliche Verteilung über einem kontinuierlichen Zustandsraum

Beispiel: kontinuierlicher Bayesfilter

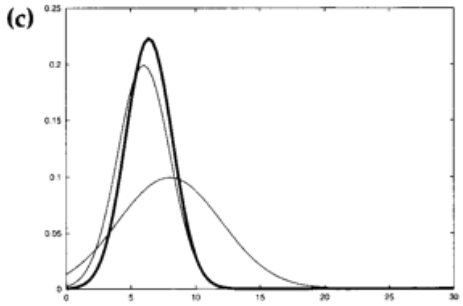
Initiale Vorhersage (t_0)

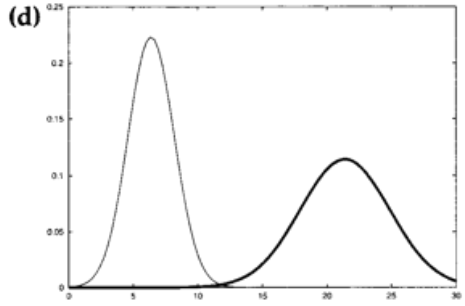




Beobachtung (t_0)

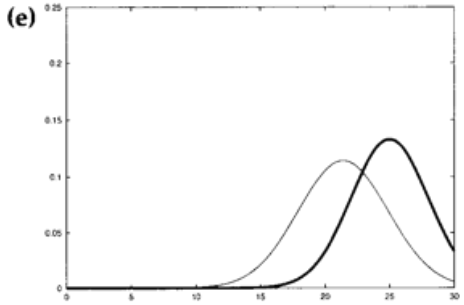
belief (t_0)

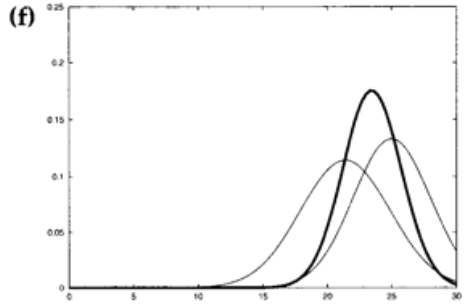




Vorhersage (t_1)

Beobachtung (t_1)





belief (t_1)

Bayes Filter: Vorhersage

- Zustände x, Aktion u
- Diskreter Fall:

$$\overline{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t-1}$$

- Kontinuierlicher Fall:

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

Bayes Filter: Beobachtung

- Zustände x, Beobachtung z
- Diskreter Fall:

$$p_{k,t} = \eta\, p(z_t | X_t = x_k)\, \overline{p}(k,t)$$

- Kontinuierlicher Fall:

$$bel(x_t) = \eta\, p(z_t | x_t)\, \overline{bel}(x_t)$$

Diskreter Bayes Filter

- Der vollständige, diskrete Bayes Filter mit Zuständen *x*, Aktion *u* und Beobachtung *z*:

$$
\begin{array}{l}
\textbf{discreteBayesFilter}(P_{t-1}, u_t, z_t): \\
\quad \textbf{for each } p_k \textbf{ in } P \textbf{ do} \\
\quad\quad \overline{p}_{k,t} = \sum_i p(X_t = x_k | u_t, X_{t-1} = x_i) p_{i,t-1} \\
\quad\quad p_{k,t} = \eta\, p(z_t | X_t = x_k)\, \overline{p}(k,t) \\
\quad \textbf{endfor} \\
\quad \textbf{return } P_t
\end{array}
$$

Diskreter Bayes Filter: Alternativ

- Alternative Darstellung für den diskreten Bayes Filter mit Normalisierer α, Zuständen s, explizitem Transitionsmodell *T* und Beobachtungsmodell *O*:

$$\textbf{discreteBayesFilter}(S_{t-1}, u_t, z_t, T, O):$$
$$\quad \textbf{for each } s_t \textbf{ in } S \textbf{ do}$$
$$\quad\quad p(s_t) \;=\; \alpha\, O(z_t, s_t) \sum_{s_{t-1}} T(s_t, u_t, s_{t-1})\, p(s_{t-1})$$
$$\quad \textbf{endfor}$$
$$\quad \textbf{return } s_t$$

Allgemeiner kontinuierlicher Bayes Filter

- Der allgemeine, kontinuierliche Bayes Filter:

$$
\begin{aligned}
&\textbf{generalBayesFilter}(bel_{t-1}, u_t, z_t):\\
&\quad \textbf{for each } x_t \textbf{ in } bel \textbf{ do}\\
&\qquad \overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}\\
&\qquad bel(x_t) = \eta\, p(z_t|x_t)\, \overline{bel}(x_t)\\
&\quad \textbf{endfor}\\
&\quad \textbf{return } bel(x_t)
\end{aligned}
$$

Spezielle kontinuierliche Bayes Filter

- Kontinuierliche Bayes Filter werden durch spezielle Methoden realisiert
- Parametrische Filter:
  - Kalman Filter (siehe Robotik III)
  - Extended Kalman Filter
  - Unscented Kalman Filter
  - Information Filter
  - Extended Information Filter
- Nicht-Parametrische Filter:
  - Partikelfilter

MDP + Bayes Filter

- Um MDPs in einer unvollständig beobachtbaren Umgebung nutzen zu können: Bayes Filter mit MDPs verbinden
- Dies erfordert die Einführung der Zustandsvermutung in den MDP

Zustandsvermutung im MDP

- Durch die Zustandsvermutung hat der MDP nun zwei Komponenten:
  - Realer Zustand der Welt
  - Subjektive Zustandsvermutung des Agenten

⇒ **Partially Observable Markov Decision Processes (POMDPs)**

Unvollständig beobachtbare MDPs

- POMDPs ermöglichen die Entscheidungsfindung in Domänen mit den Eigenschaften
  - Dynamisch
  - Sequentiell
  - Diskret oder Kontinuierlich
  - Stochastisch
  - ***Unvollständig beobachtbar***

POMDP: Struktur

- Die Struktur eines diskreten POMDPs wird durch folgendes Modell beschrieben:
  - Eine Menge von symbolischen **Zuständen** der Welt ***S***
  - Eine Menge von symbolischen **Handlungen** des Agenten ***U***
  - Eine Menge von symbolischen **Beobachtungen** des Agenten ***M***
  - Ein **Übergangsmodell** (*transition model*) *T(s', u, s)* welches die stochastischen Übergänge modelliert
  - Ein **Belohnungsmodell** (*reward model*) *R(s, u)* welches die Ziele und Absichten des Agenten modelliert
  - Ein **Beobachtungsmodell** (*observation model*) O(m, s') welches die Ungenauigkeit der Wahrnehmung des Agenten modelliert

POMDP: Statische Eigenschaften

- Die statischen Eigenschaften des reinen MDP gelten auch im POMDP
- Zusätzlich kommen hinzu:
  - Der Agent kann in jedem Agentenzyklus genau eine Beobachtung machen
  - In einem diskreten POMDP ist dies eine symbolische Beobachtung
- Im Folgenden wird von diskreten POMDPs ausgegangen

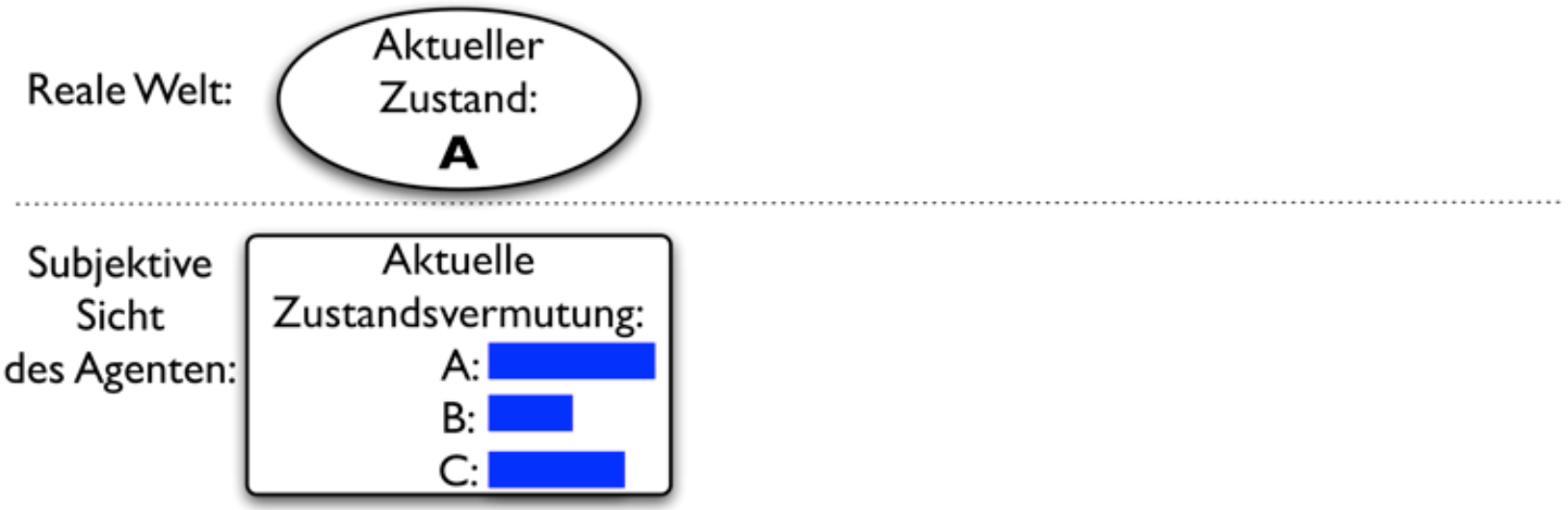
POMDP: Dynamikmodelle

- Transitions- und Belohnungsmodell funktionieren wie beim MDP
- Hinzu kommt das Beobachtungsmodell:
  - O(m, s‘) = Wahrscheinlichkeit
    - Wenn die Beobachtung m gemacht wurde, dann ist die Wahrscheinlichkeit, dass der wirkliche Weltzustand s‘ ist, vom Modell an O(m, s‘) gegeben
  - Das Beobachtungsmodell ist eine Matrix

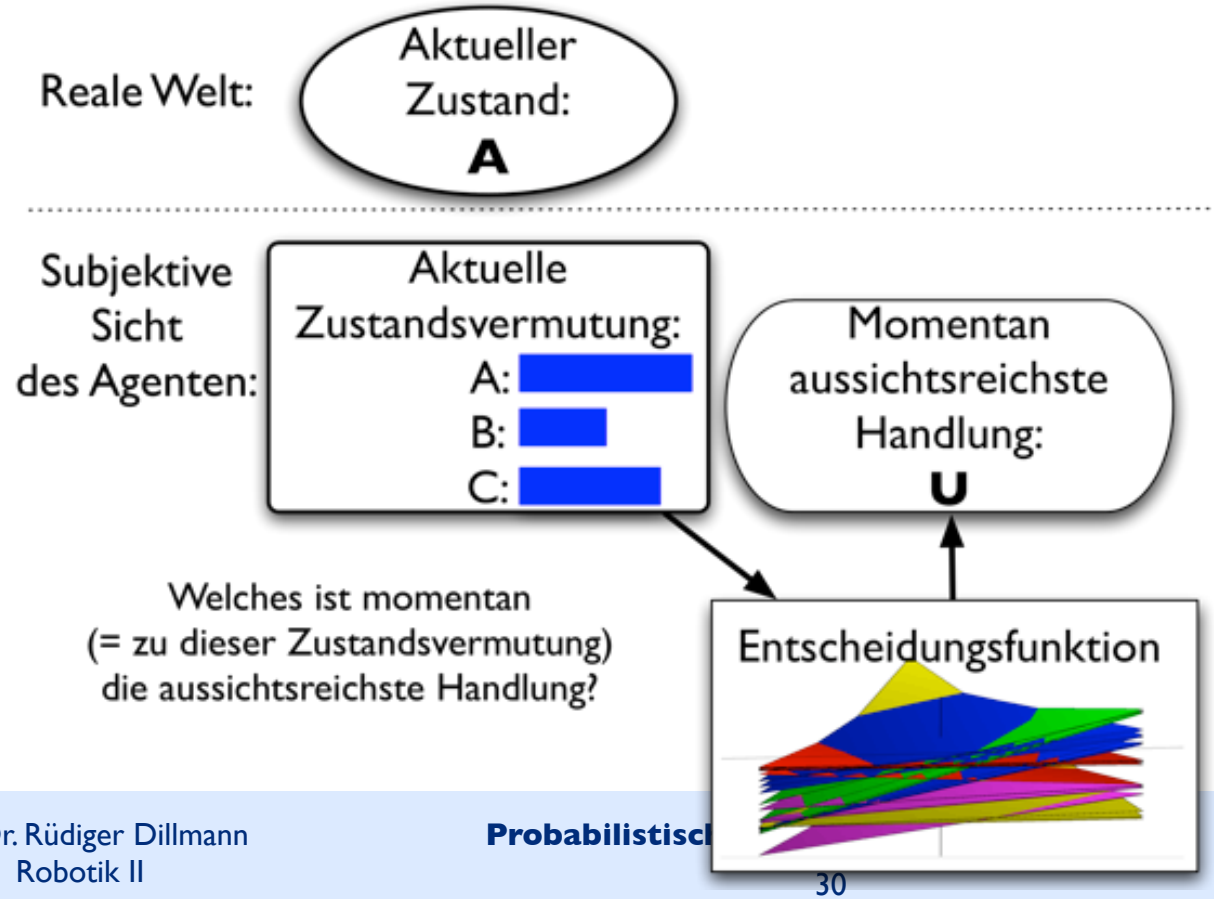
POMDP: Sekundäre Eigenschaften

- Die Entscheidungsfunktion (*policy*) existiert auch beim POMDP, wenn auch in anderer Form
- Hinzu kommt die Zustandsvermutung
  - Im diskreten Fall eine diskrete Wahrscheinlichkeitsverteilung über alle Zustände
  - Die Zustandsvermutung wird zu jedem Zeitschritt durch Bayes Filterung neu berechnet
  - Führt zu einer klaren Trennung zwischen objektiver Realität (echter Weltzustand) und der subjektiven Sicht des Agenten

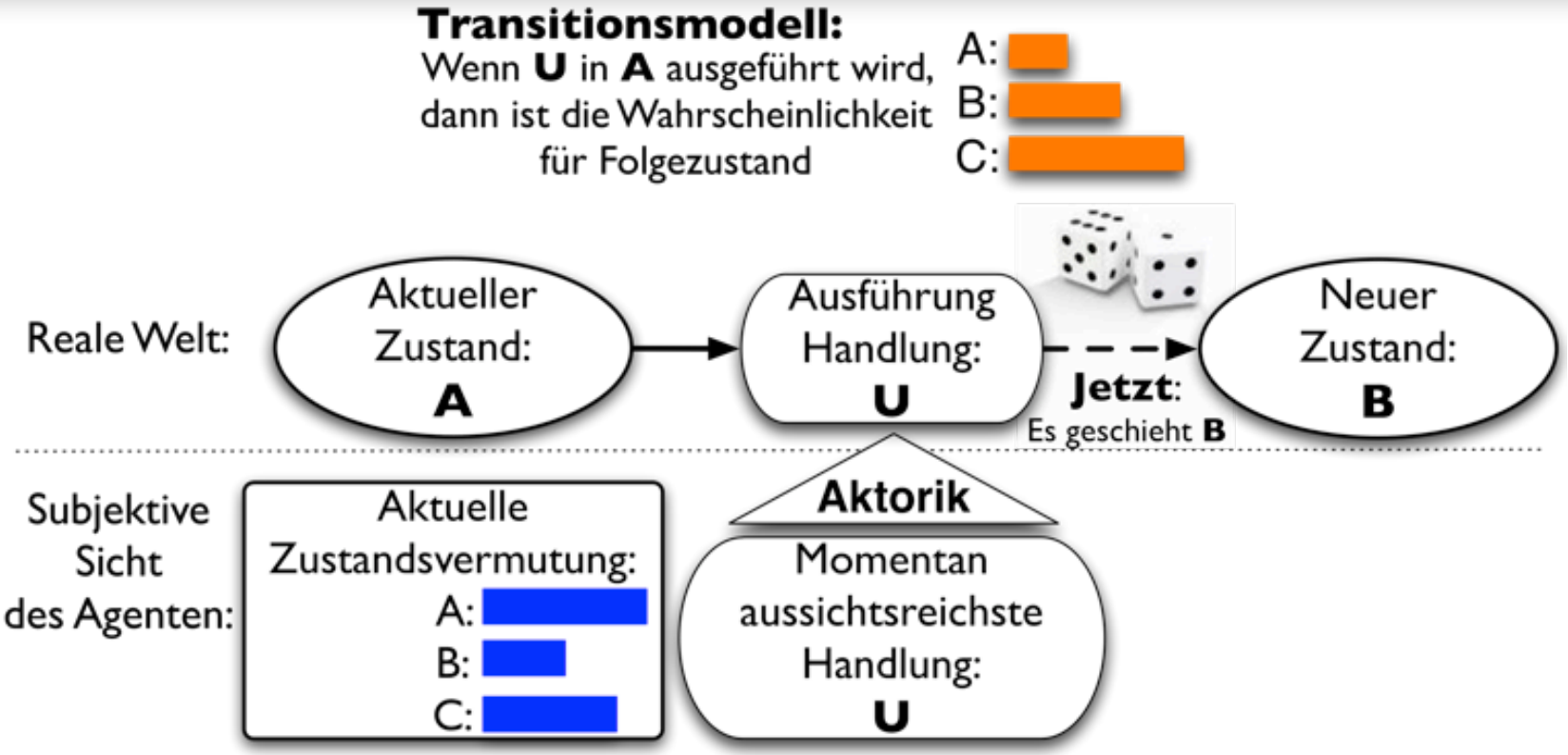
POMDP Schritt: Beginn



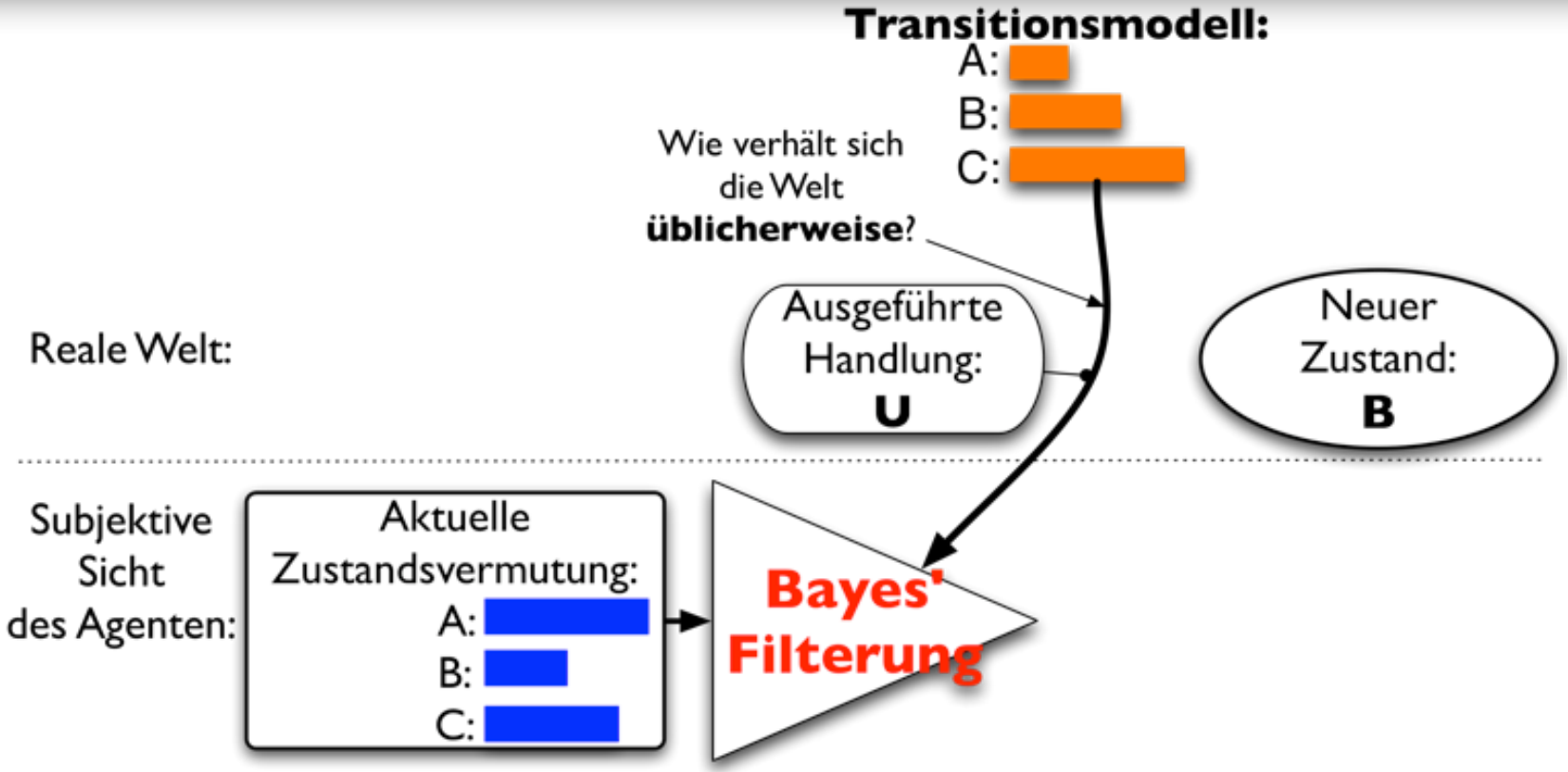
POMDP Schritt: Entscheidung



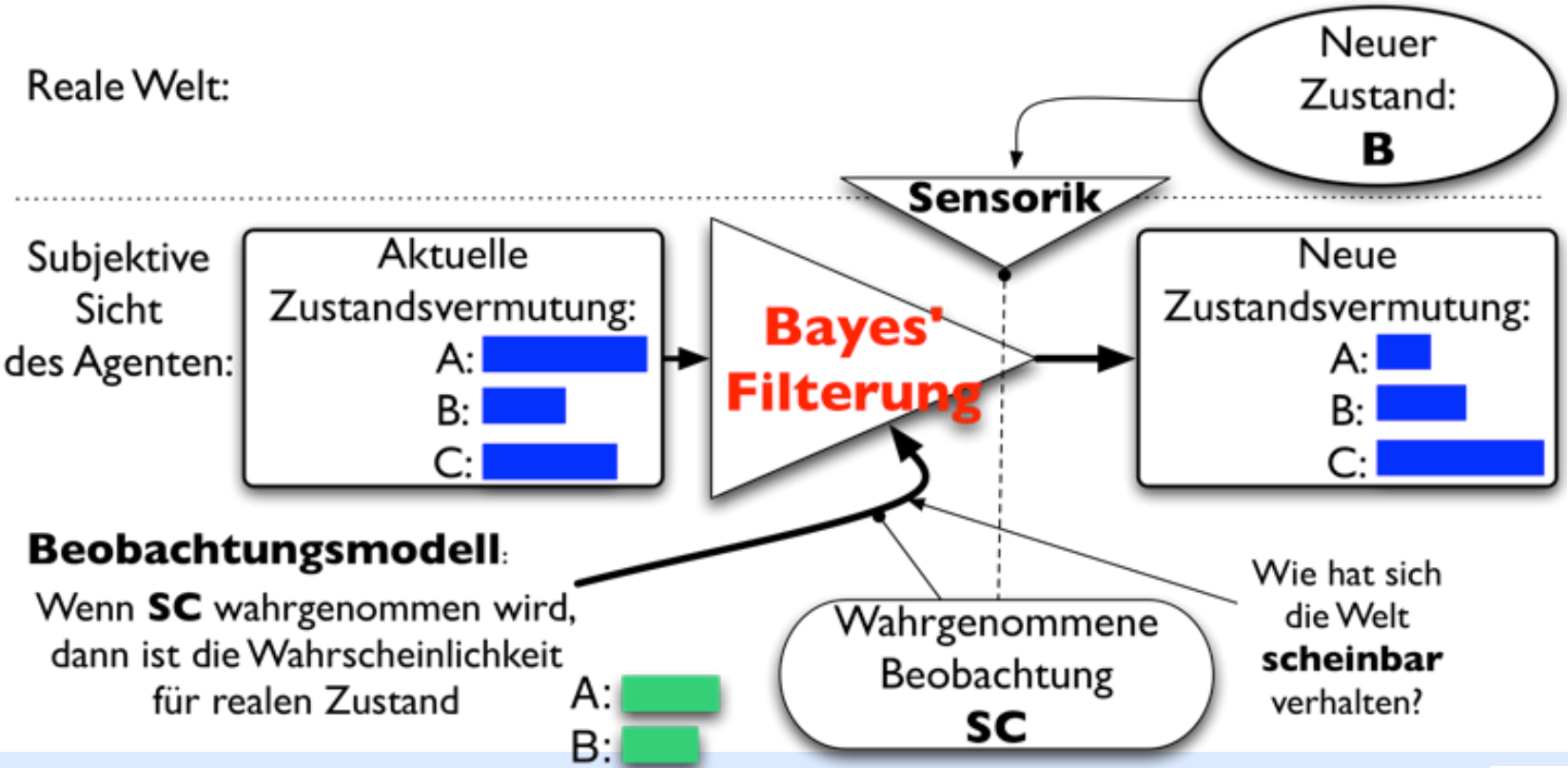
POMDP Schritt: Handlung



POMDP Schritt: Vorhersage



POMDP Schritt: Beobachtung



POMDP: Entscheidungsfunktion

- Bei MDPs enthält die Entscheidungsfunktion eine Handlung zu jedem Zustand
- Im Gegensatz zu MDPs muss die Entscheidungsfunktion bei POMDPs über **allen möglichen Zustandsvermutungen** definiert sein
- Struktur der Entscheidungsfunktion
  - Bei diskreten POMDPs:

    Über dem |S|-1 dimensionalen Raum aller möglichen Zustandsvermutungen definiert
  - Bei kontinuierlichen POMDPs:

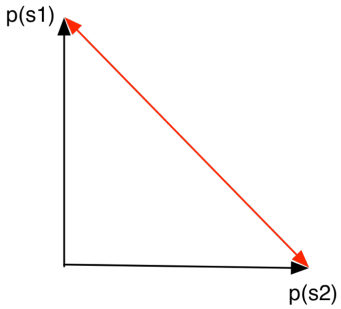
    Über einem unendlich-dimensionalen Raum definiert, daher speziell über endlich-dimensionalen Teilräumen angegeben

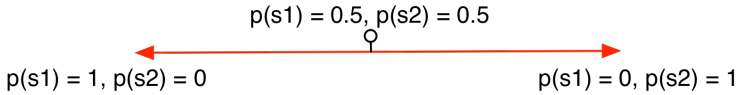
Diskreter POMDP: Raum der Vermutungen

- Alle möglichen Wahrscheinlichkeiten über allen Zuständen s eines Szenarios spannen den Raum der Zustandsvermutungen auf
- Da $\sum p(s_i) = 1$, ist der Raum begrenzt, also ein Simplex (= konvexe Hülle von n+1 Punkten in einem n-dimensionalen Raum)
- Wegen der Summe $|S|-1$ Dimensionen

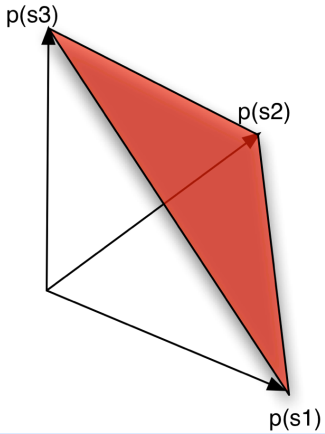
Beispiele: Raum der Vermutungen

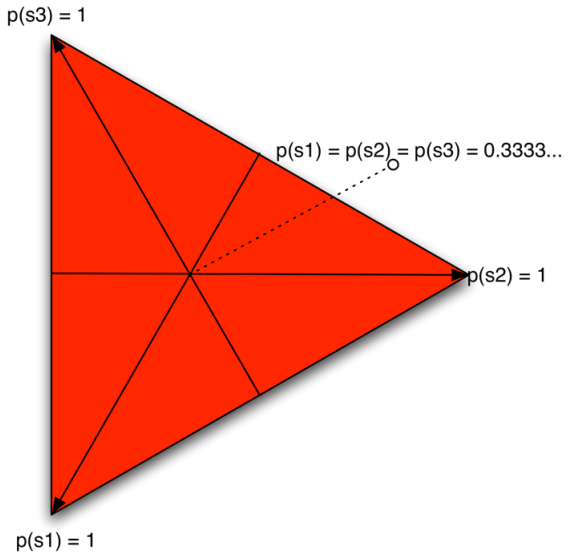
- 2 Zustände:





- 3 Zustände:





POMDP Entscheidungsfunktion: Berechnung

- Aufgrund der komplexeren Struktur, ist die Berechnung deutlich komplizierter als bei MDPs

Value Iteration existiert auch für POMDPs, funktioniert jedoch anders

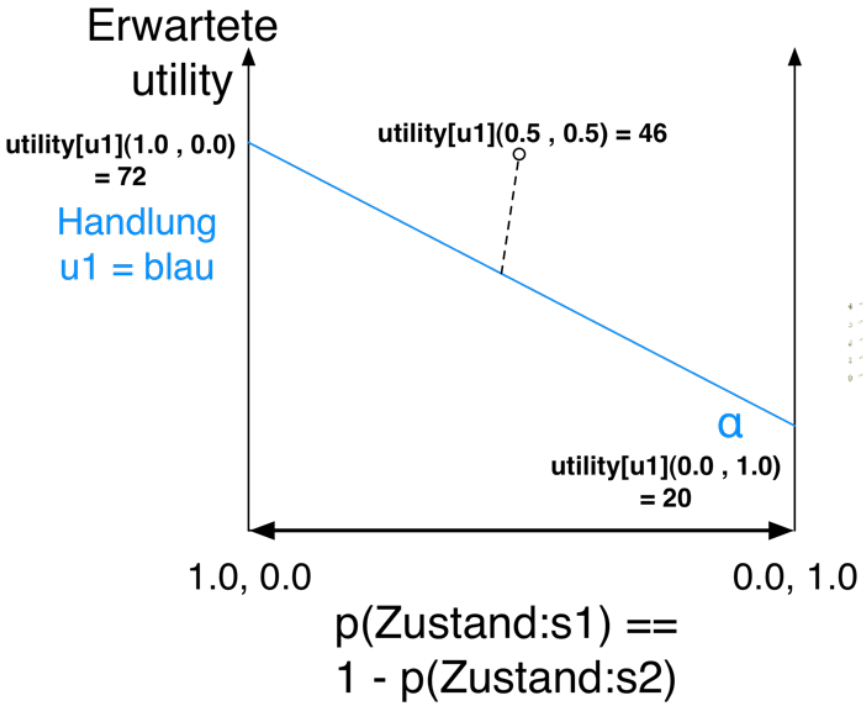
- Der iterative Ansatz ist eine Gemeinsamkeit
- Ein **endlicher Horizont** wird vorausgesetzt
- Ein schrittweises Vorausschauen wird von einer initialen Entscheidungsfunktion aus durchgeführt
- Das Vorausschauen muss die stochastische Natur der Weltdynamik, als auch die verschieden genauen Beobachtungen eben dieser Dynamik in die Abwägung einfließen lassen

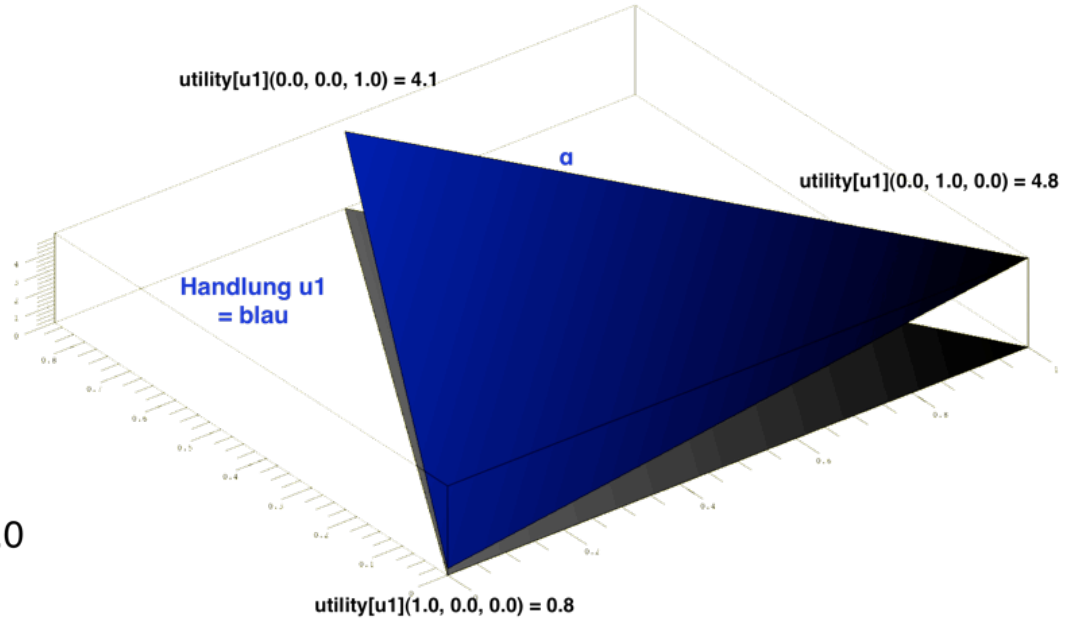
POMDP Value Iteration: Struktur

- Lineare Funktionen α repräsentieren die utility kontinuierlich über dem gesamten Zustandsvermutungs-Simplex bezogen auf eine Handlung (nicht Zustand, wie im MDP)
- Für jede Zustandsvermutung kann aus den Funktionen eine utility berechnet werden

Beispiel: Value Iteration Struktur

- Beispiele für 2 und 3 Zustände





POMDP Value Iteration: Beginn

- Der Algorithmus beginnt mit dem Pseudohorizont 0
- Nun wird bis zu einem Horizont T in die Zukunft iteriert
- Dabei wird eine Menge Γ von linearen Funktionen α erzeugt, die Gewinnerwartungswerte für Mengen möglicher Handlungsketten darstellen

POMDP Value Iteration: Schritt

- Ein Iterationsschritt besteht aus zwei wesentlichen Teilen:
  - Temporäre Koeffizienten werden durch die Bayes' Vorwärtsfilterung (Projektion) jeder vorigen linearen Funktion über jede mögliche Handlung und Wahrnehmung erzeugt
  - Die temporären Koeffizienten werden für die Bildung der Erwartung über alle möglichen Beobachtungskombinationen verwendet, woraus die neuen linearen Funktionen erzeugt werden

POMDP Value Iteration Schritt: Teil 1

- Die Projektion jeder linearen Funktion α aus Γ in t-1 durch jede Kombination von Aktion *u* und Beobachtung *m* führt zu den temporären Koeffizienten *v{k}*:

$$
\begin{aligned}
&\textbf{for each } \alpha^k \textbf{ in } \Gamma \textbf{ do} \\
&\quad \textbf{for each } u \textbf{ in } \text{Actions} \textbf{ do} \\
&\quad\quad \textbf{for each } m \textbf{ in } \text{Measurements} \textbf{ do} \\
&\quad\quad\quad \textbf{for } j = 1 \textbf{ to } |States| \textbf{ do} \\
&\quad\quad\quad\quad v^k_{a,m,j} = \textstyle\sum_i^{|States|} \alpha^k_i * O[m,i] * T[i,u,j] \\
&\quad\quad\quad \textbf{endfor} \\
&\quad\quad \textbf{endfor} \\
&\quad \textbf{endfor} \\
&\textbf{endfor}
\end{aligned}
$$

POMDP Value Iteration Schritt: Teil 2

- Aus den temporären Koeffizienten wird für jede mögliche Kombination von Beobachtungen die Gewinnerwartung berechnet und eine neue lineare Funktion α‘ erzeugt:

for each a **in** Actions **do**
  for each $mset[*] = (1_1, 1_2, ..., 1_M)$ **to** $(|\Gamma|_1, ..., |\Gamma|_M)$ **do**
    for $j = 1$ **to** $|States|$ **do**
      for each m **in** Measurements
        $v''_j += v^{mset[m]}_{a,m,j}$
      endfor
      $\alpha'_a[j] = \gamma * (R(j,a) + v''_j)$
    endfor
    $\Gamma' <= \alpha'_a$
  endfor
endfor

Die linearen Constraints Alpha entstehen durch Bildung des Erwartungswerts über die Beobachtungen. Für jede Aktion u generiert der Algorithmus K^M solcher Constraints Alpha.

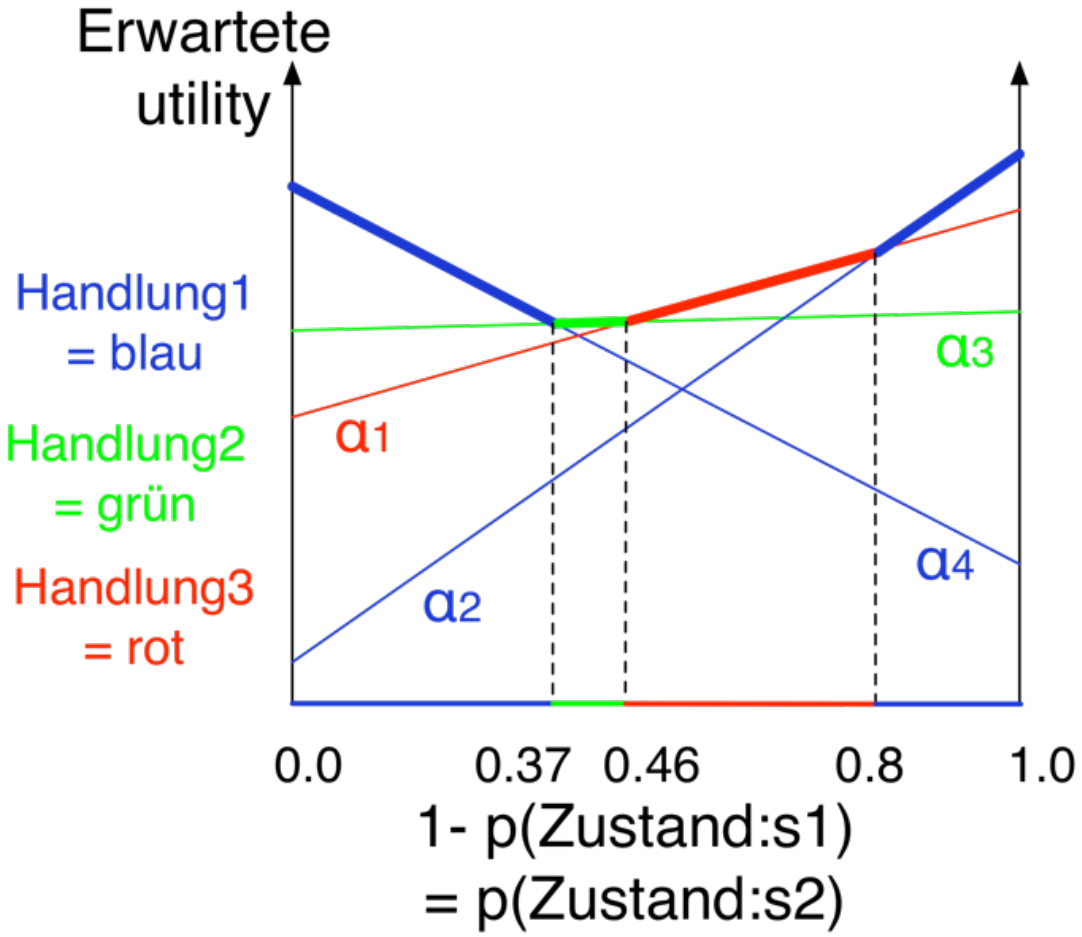
Jede Erwartung wird über M mögliche Beobachtungen gebildet, welche mit jeder der K bisherigen Constraints Alpha kombiniert werden kann

POMDP VI Entscheidungsfunktion

- Die Entscheidungsfunktion ist nun das Maximum der utility in der Menge von linearen Funktionen über dem jeweiligen Bereich des Zustandsvermutungsraums
- Diese Menge ist immer konvex über dem Zustandsvermutungs-Simplex

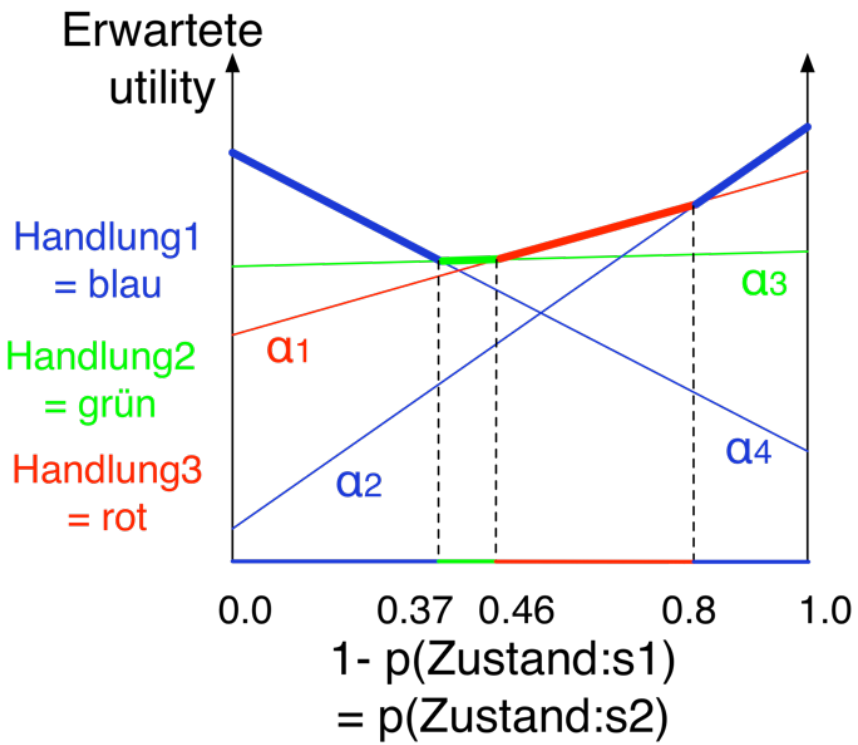
Beispiel: VI Entscheidungsfunktion

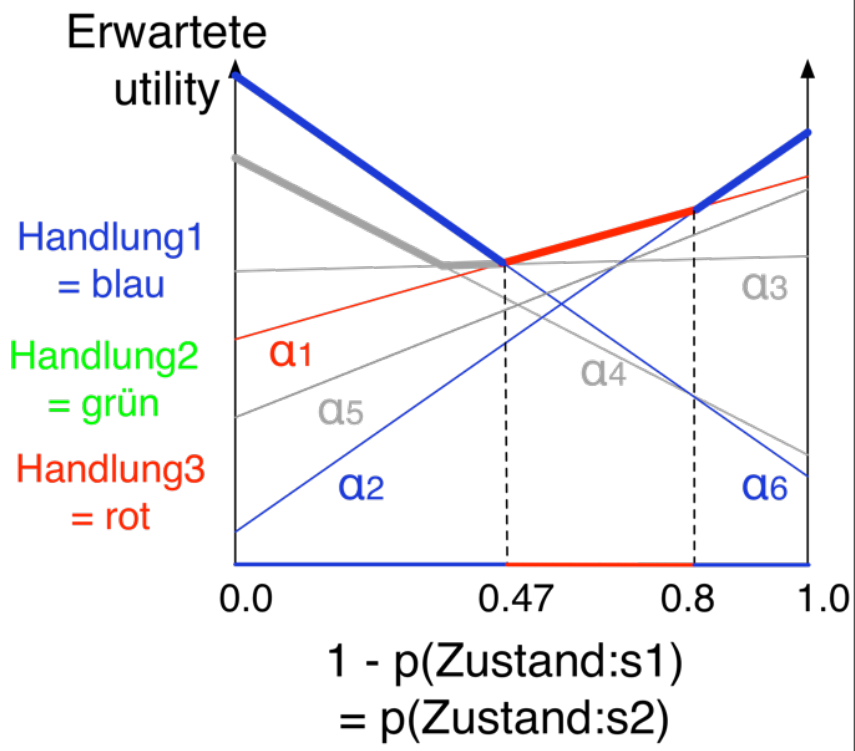
- 2 Zustände, 3 Handlungen



Schema: VI Schritt

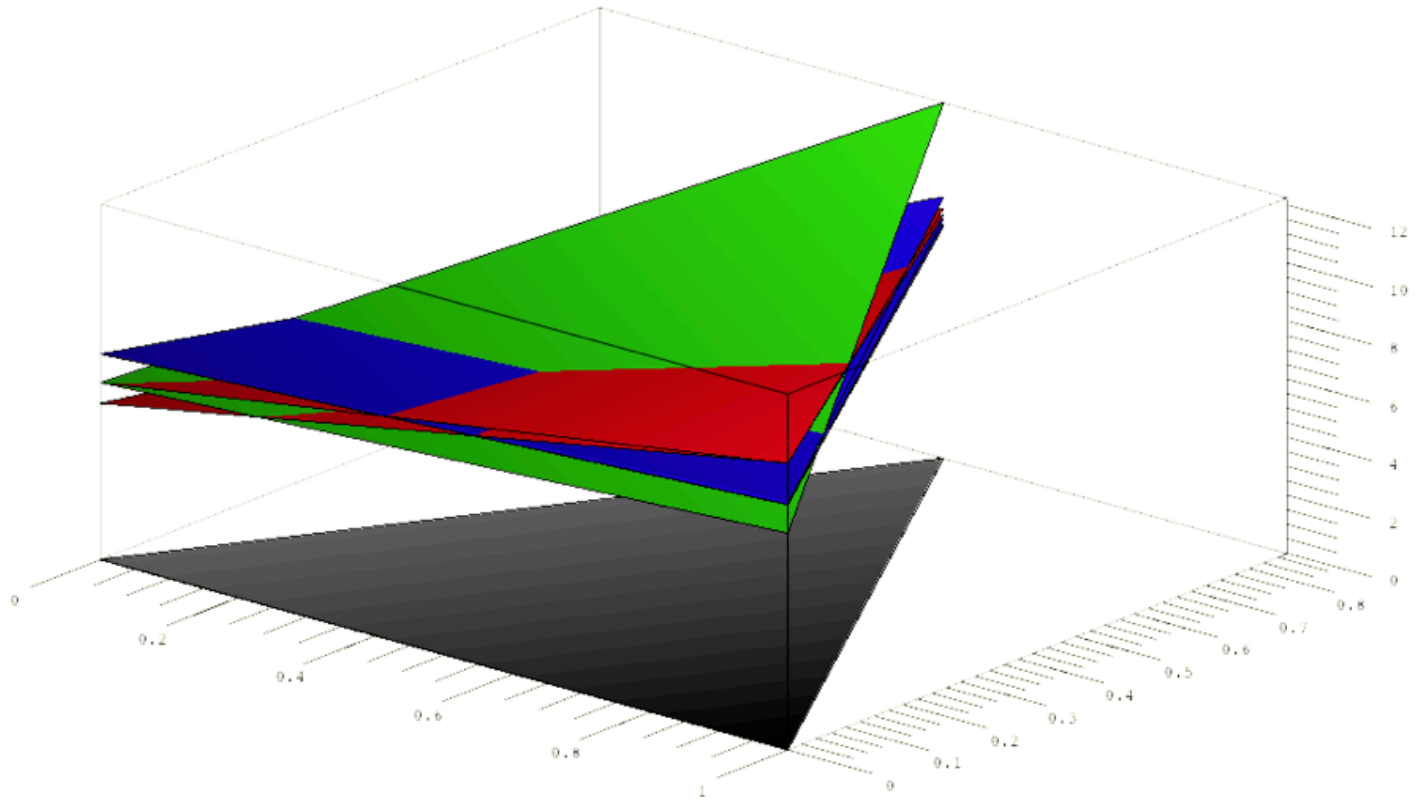
- 2 Zustände, 3 Handlungen





Beispiel: VI Entscheidungsfunktion

- 3 Zustände, 3 Handlungen



POMDP Value Iteration: Komplexität

- Schritt Teil 2 ist aus komplexitätstheoretischer Sicht bedeutend:

$$\text{for each } mset[*] = (1_1, 1_2, ..., 1_M) \textbf{ to } (|\Gamma|_1, ..., |\Gamma|_M) \textbf{ do}$$

- Hier geschieht eine kombinatorische Explosion bezüglich der Anzahl der erzeugten α:

$$|\Gamma_t| = |Actions| * |\Gamma_{t-1}|^{|Observations|}$$

- Was zu einer doppelt-exponentiellen Komplexität (2-EXPSPACE) führt:

$$|\Gamma_t| = |Actions|^{(\sum_{i=0}^{t-1} |Observations|^i)}$$

Beispiel: Value Iteration Komplexität

- Beispiel 3 Aktionen, 3 Beobachtungen, $|\Gamma|$:
  - Horizont 1: 3
  - Horizont 2: 81
  - Horizont 3: 1594323
  - Horizont 4: 1.2 x 10^19
  - Horizont 5: 5.4 x 10^57

⟹ Exakte POMDP Value Iteration unbrauchbar

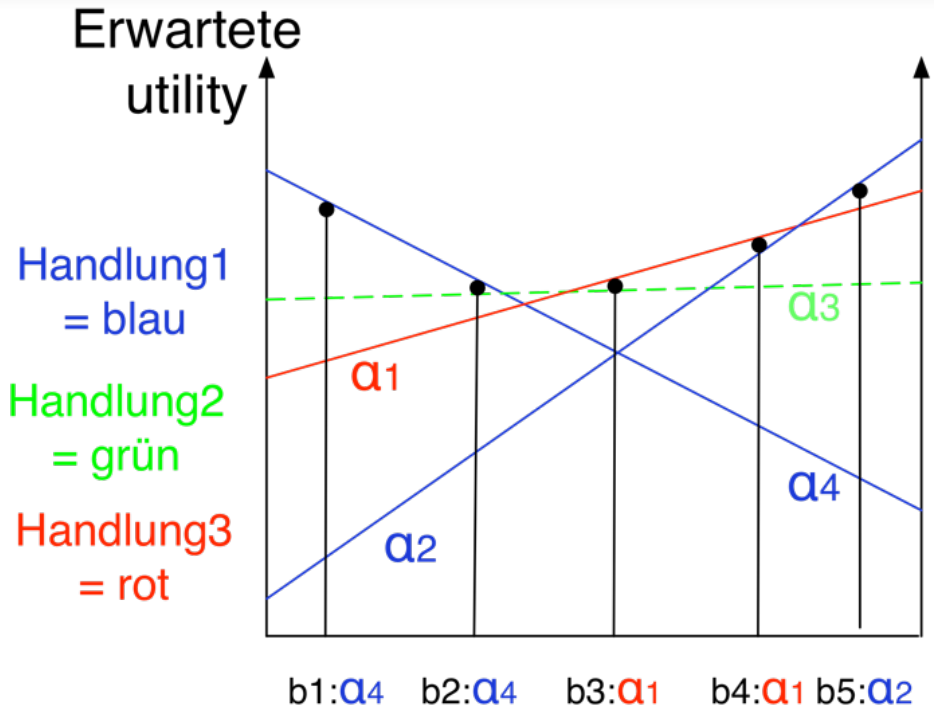
Approximative Value Iteration

- Um die Komplexität handhabbar zu machen, wurden approximative Value Iteration Verfahren entwickelt
- Dadurch wird mindestens der Teil 2 des Iterationsschritts entschärft
- Alle relevanten Verfahren haben garantierte Fehlerschranken und zeichnen sich durch sehr gute Approximation aus

Point Based Value Iteration

- Point Based Value Iteration (PBVI) [2003] nutzt ein dynamisches Raster zur Berechnung der Entscheidungsfunktion
- Die Entscheidungsfunktion wird nur für eine dynamische Menge von Stellvertreterpunkten im Zustandsvermutungsraum berechnet
- Sie gilt jedoch für den gesamten Raum

PBVI: Schema



- Die Genauigkeit hängt in der Praxis von der Anzahl der Vertreterpunkte und der Änderungshäufigkeit optimaler Handlungen über dem Zustandsraum ab

Weitere approximative VI

- PERSEUS [2004]
- HSVI2 [2006]
- PERSEUS für kontinuierliche POMDPs [2006]
- SARSOP [2008]

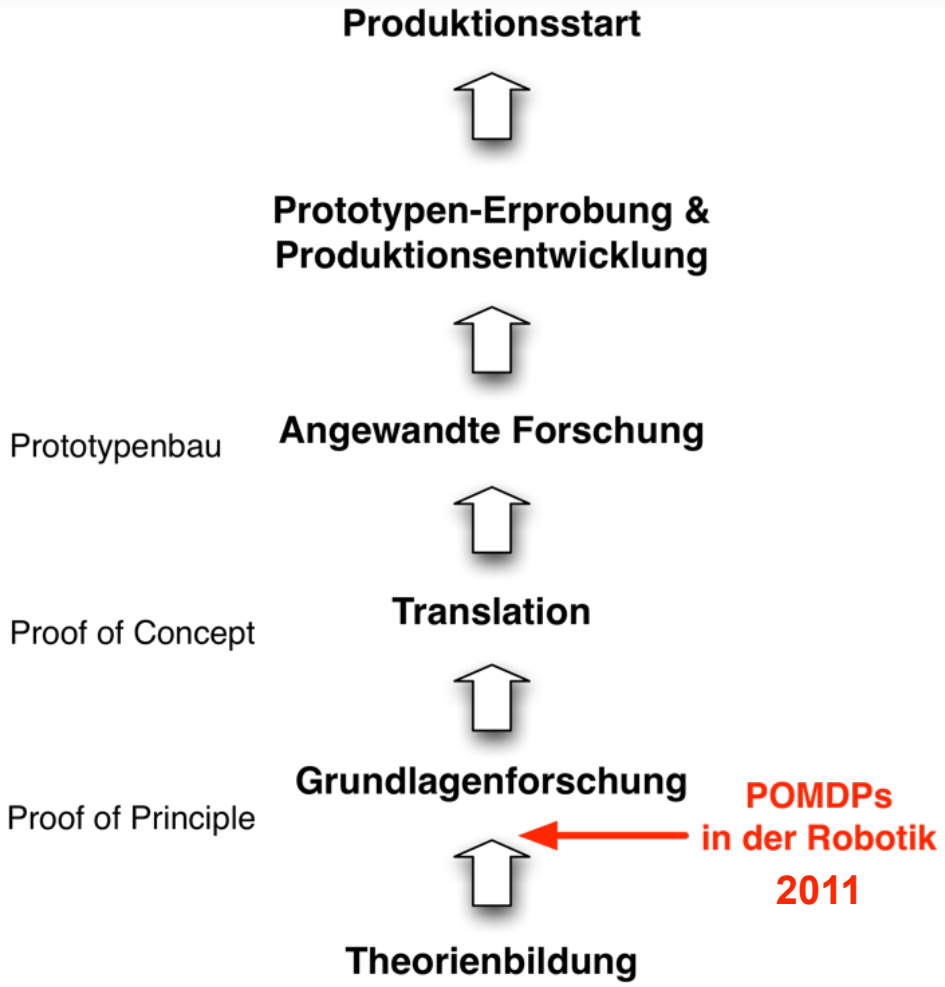
- Theoretische Grundlagen noch ganz aktuelles Forschungsthema

POMDPs und Serviceroboter

- Die Nutzung von POMDPs in autonomen Servicerobotern steht noch ganz am Anfang
- Momentan fast nur in Simulationen und virtuellen Benchmarkszenarien im Einsatz
- Mono-Modale SLAM-Szenarien stehen dort noch im Mittelpunkt

POMDPs: Technologieeinordnung

- Die Übertragung der Theorie auf Robotersteuerungen wird gerade erforscht

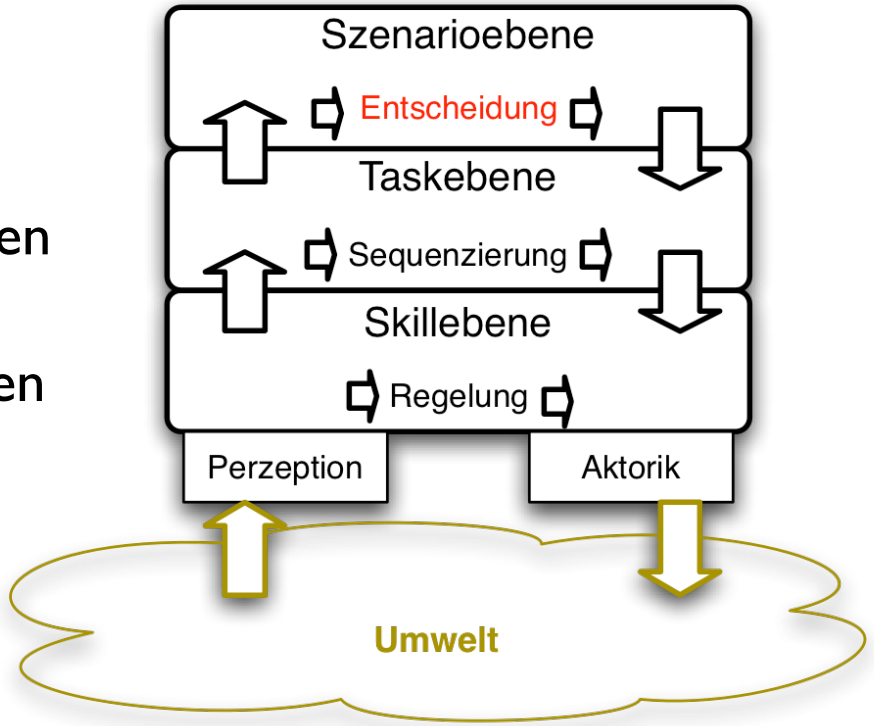


POMDPs auf Servicerobotern

- Herausforderungen:
  - Überwindung der Abstraktionslücke zwischen Roboter Sensorik/Aktorik und symbolischen POMDPs
  - Stochastische Modellierung von Szenarien
  - Strukturierung der stochastischen Modelle
  - Integration der Stärken klassischer Planung/ Ausführung
  - Integration von maschinellem Lernen

POMDPs in einer Robotersteuerung

- POMDPs werden mit anderen Steuerungsmethoden zusammen eingesetzt
- Schichtung trennt die Methoden



POMDP Grundlagenliteratur

- Artificial Intelligence: A Modern Approach
  2nd Edition; S. Russel, P. Norvig; 2003
  Kapitel 17
- Probabilistic Robotics
  S. Thrun, W. Burgard, D. Fox; 2005
  Kapitel 15
- Planning Algorithms
  S. M. LaValle; 2006
  Kapitel III

POMDP weiterführende Literatur

[1] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for pomdps," in *International Joint Conference on Artificial Intelligence (IJCAI)*, August 2003, pp. 1025 – 1032.

[2] P. Poupart and C. Boutilier, "Vdcbpi: an approximate scalable algorithm for large scale pomdps," in *In Advances in Neural Information Processing Systems 17 (NIPS)*, 2004, pp. 1081–1088.

[3] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for pomdps," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[4] P. Poupart, N. Vlassis, J. Hoey, and K. Regan, "An analytic solution to discrete bayesian reinforcement learning," in *In Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

[5] T. Smith and R. Simmons, "Focused real-time dynamic programming for mdps: Squeezing more out of a heuristic," in *Nat. Conf. on Artificial Intelligence (AAAI)*, 2006.