

6. Zustands-Automaten

Bisher:

- **R-S-Agenten mit unmittelbarer Reaktion auf Sensor Reize**

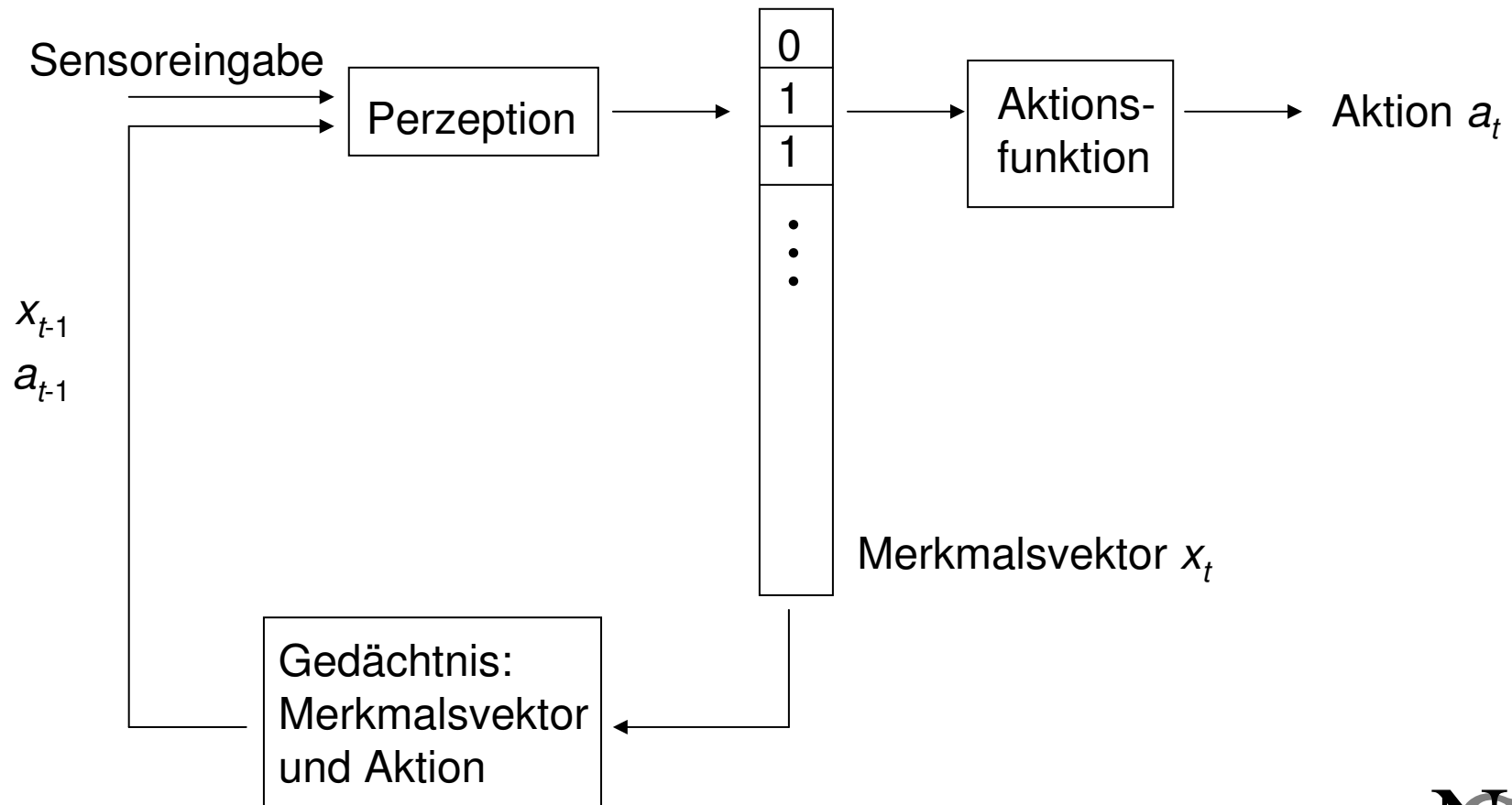
Jetzt:

- **Ausnutzung von Sensorinformationen aus der Vergangenheit**

Zustands-Automaten

6.1 Darstellung der Umgebung mit Merkmalsvektoren

Beispiel:



6.1 Merkmalsvektoren

Beispiel: **Roboter in Gitterwelt mit begrenzter Sensorinformation**

■ **Sensoreingabe zum Zeitpunkt t:**

- $s_2^t, s_4^t, s_6^t, s_8^t$ (d.h. nur 4 statt bisher 8 Sensoren)
- $s_i^t = 1 \Leftrightarrow$ Feld s_i^t ist nicht frei

■ **Aufgabe: Wand folgen**

■ **Idee: den Merkmalsvektor vom Zeitpunkt davor nutzen**

Beispiel: Roboter in Gitterwelt (2)

■ Definition der Merkmalsvektoren:

$$w_i^t = s_i^t \text{ für } i = 2, 4, 6, 8$$

$$w_1^t = 1 \Leftrightarrow w_2^{t-1} = 1 \text{ and } a_{t-1} = \text{east}$$

$$w_3^t = 1 \Leftrightarrow w_4^{t-1} = 1 \text{ and } a_{t-1} = \text{south}$$

$$w_5^t = 1 \Leftrightarrow w_6^{t-1} = 1 \text{ and } a_{t-1} = \text{west}$$

$$w_7^t = 1 \Leftrightarrow w_8^{t-1} = 1 \text{ and } a_{t-1} = \text{north}$$

→ **Eingeschränkte Sensorinformationen können (teilweise) ausgeglichen werden!**

Beispiel: Roboter in Gitterwelt (3)

■ Sinnvolle Aktionen zur Wandverfolgung:

$$w_2^t \wedge \overline{w_4^t} \rightarrow \text{east} \quad w_1^t \wedge \overline{w_2^t} \rightarrow \text{north}$$

$$w_4^t \wedge \overline{w_6^t} \rightarrow \text{south} \quad w_3^t \wedge \overline{w_4^t} \rightarrow \text{east}$$

$$w_6^t \wedge \overline{w_8^t} \rightarrow \text{west} \quad w_5^t \wedge \overline{w_6^t} \rightarrow \text{south}$$

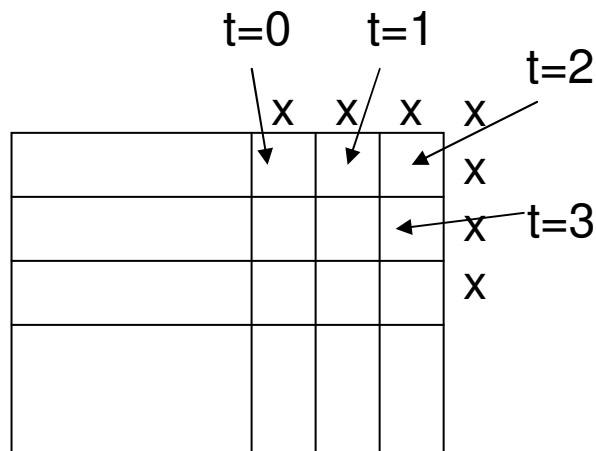
$$w_8^t \wedge \overline{w_2^t} \rightarrow \text{north} \quad w_7^t \wedge \overline{w_8^t} \rightarrow \text{west}$$

$$\text{alle } w_i = 0 \rightarrow \text{north}$$

Beispiel: Roboter in Gitterwelt (4)

■ Implementierung (Beispiel einer Bewegung)

	Sensoreingabe				Merkmalsvektor								Aktion
t	s_2^t	s_4^t	s_6^t	s_8^t	w_1^t	w_2^t	w_3^t	w_4^t	w_5^t	w_6^t	w_7^t	w_8^t	a_t
0	1	0	0	0	0	1	0	0	0	0	0	0	east
1	1	0	0	0	1	1	0	0	0	0	0	0	east
2	1	1	0	0	1	1	0	1	0	0	0	0	south
2'	0	0	0	0	1	0	0	0	0	0	0	0	north



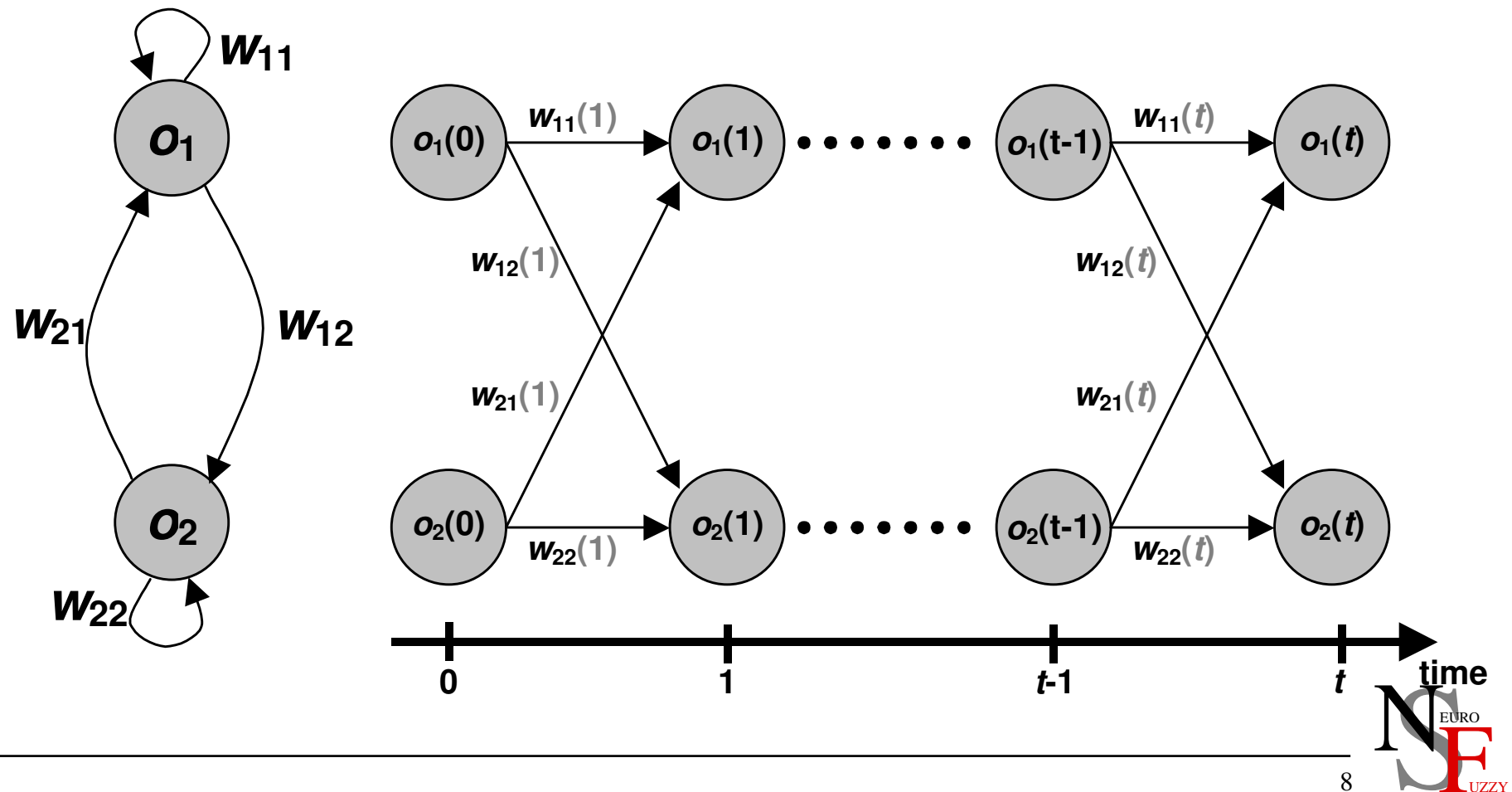
Anmerkung: Situation t=2' z.B. bei Sensorstörung (alle Sensoren $s_i=0$)

6.2 Rekurrent Neuronale Netze

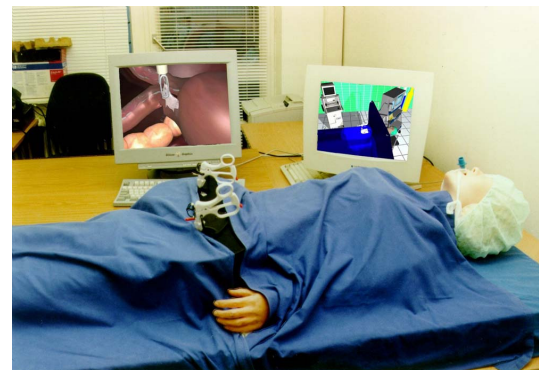
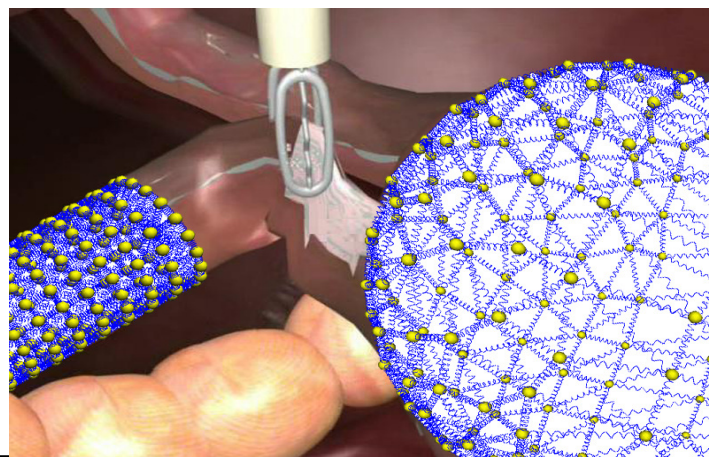
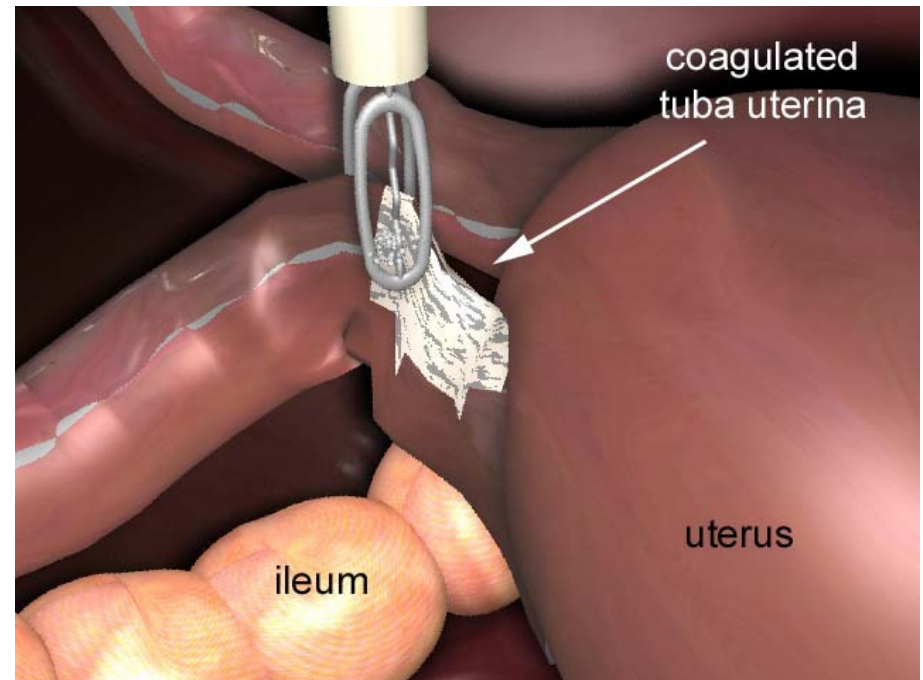
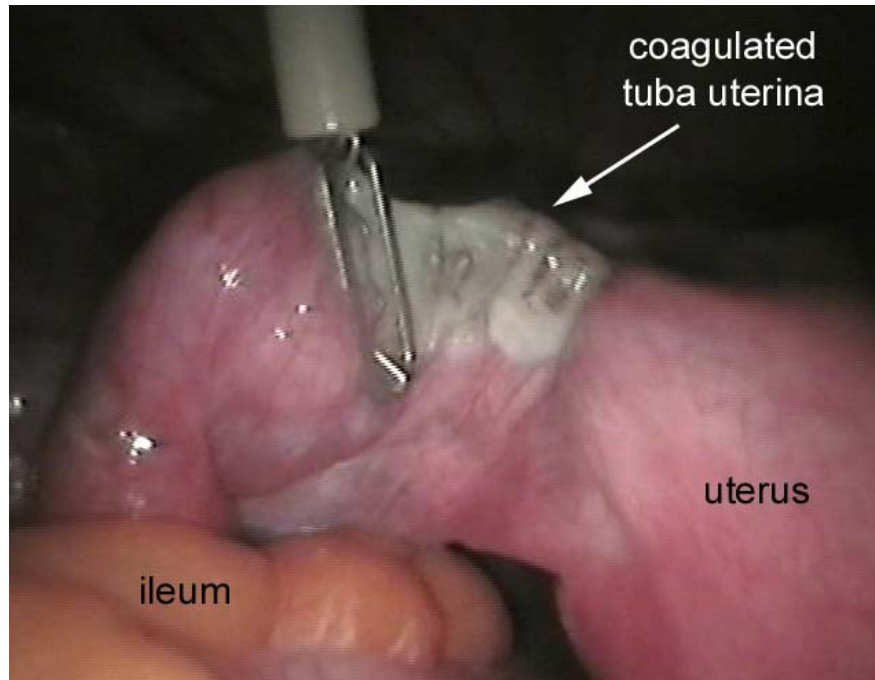
- In rückgekoppelten neuronalen Netzen sind Verbindungen von oberen zu unteren Schichten erlaubt (Multilayerperzeptron: nur Verbindungen von „unten“ nach „oben“).
- Durch die Rückkopplung kann Information gespeichert werden.
- Rückgekoppelte Netze können durch Modifikationen von Backpropagation-Verfahren trainiert werden.
- Rückgekoppelte Netze können komplexe Differentialgleichungssystem approximieren.

Rekurrente Netze: Lernverfahren

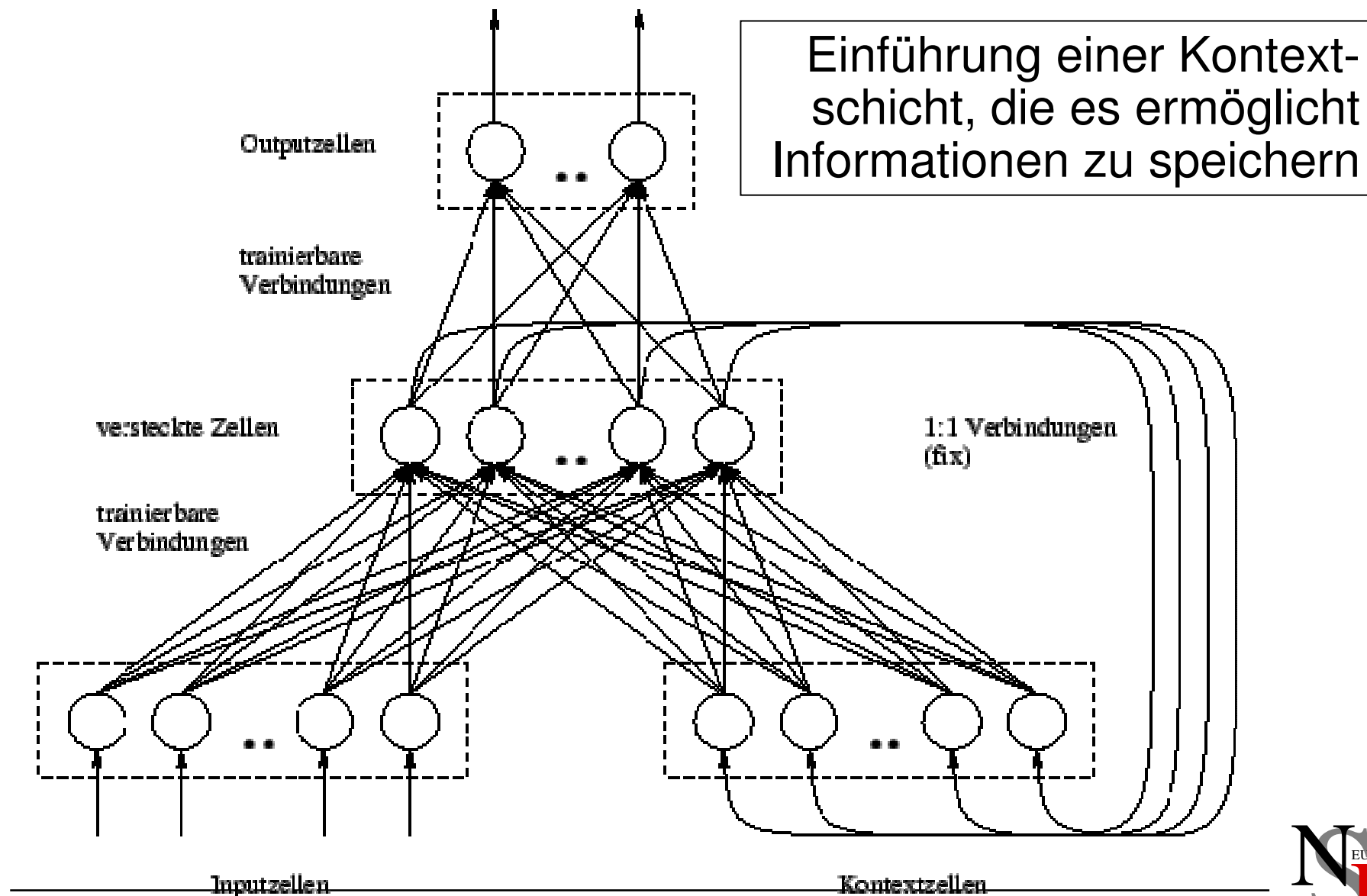
- Idee der Lernverfahren: Entfalten des Netzes über die Zeit



SUSILAP-G (SURgical Simulator for LAParoscopy in Gynaecology)



Einfache RNNs: Elman Netze

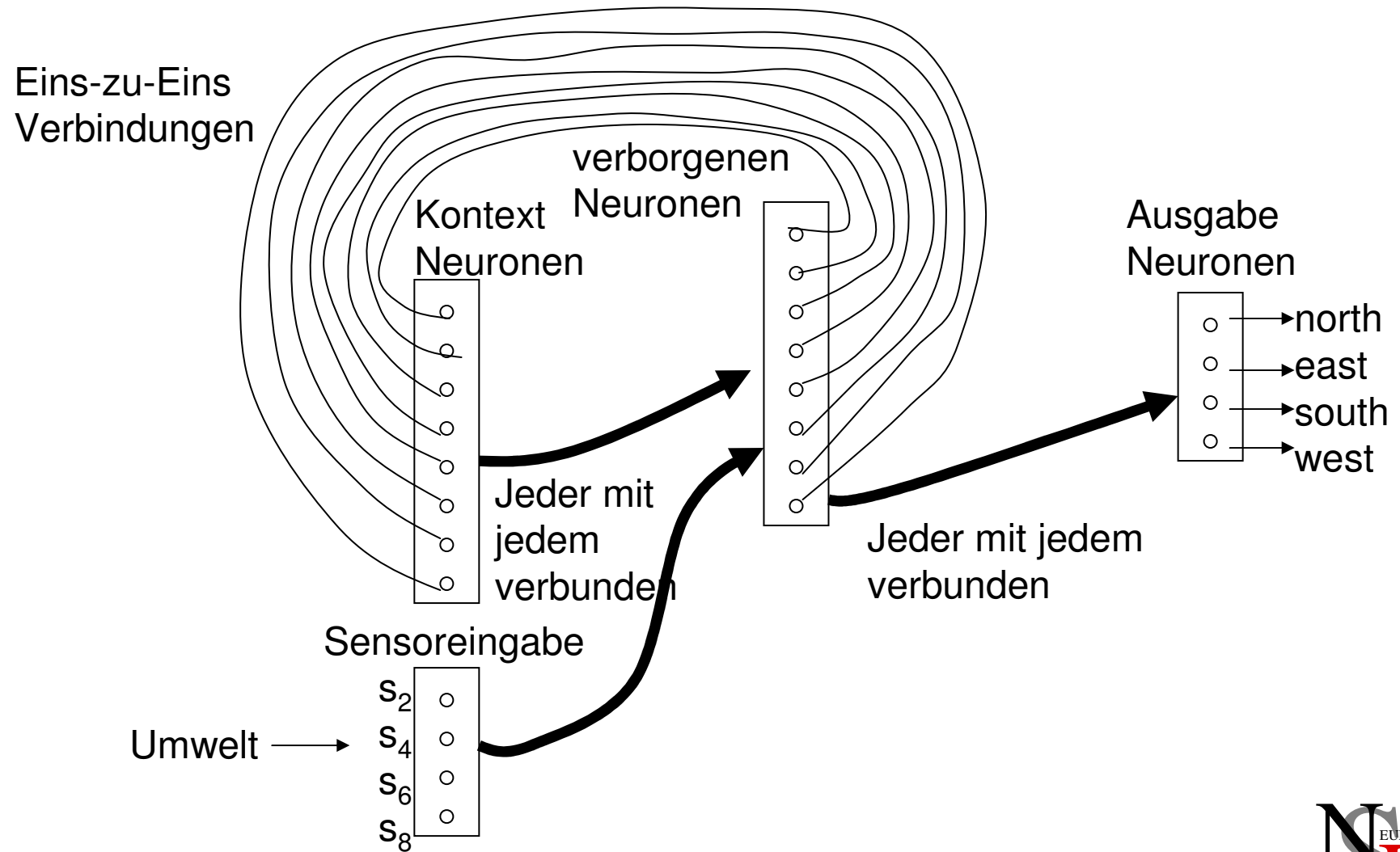


Beispiel: Elman Netz für Roboter

Elman Netz für Roboter in Gitterwelt:

- **8 dimensionale Merkmalsvektoren (im allgemeinen ist die Anzahl der Dimensionen unbekannt)**
- **4 Eingaben (Sensoren)**
- **4 Ausgaben (Richtungen; Ausgabe mit größtem Wert wird gewählt)**
- **Dieses rückgekoppelte Netz kann auch mit Backpropagation trainiert werden.**
- **Man kann diese Netze als „lernfähige“ Automaten auffassen.**

Beispiel: Elman Netz (2)



6.3 Nutzung von Umgebungsmodellen

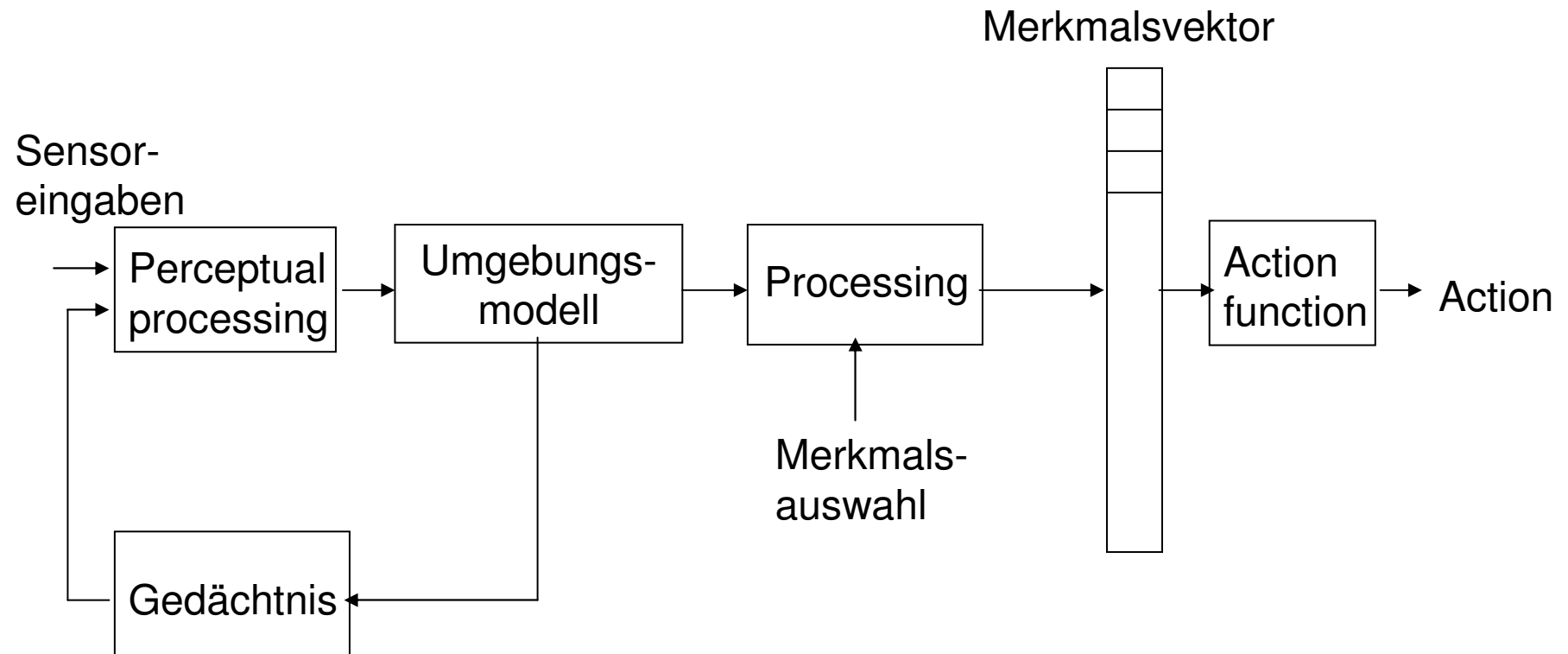
- **Bisher hatte der Roboter nur Informationen über einen sehr kleinen Ausschnitt seiner Umgebung:**
 - unmittelbare Nachbarschaft
 - gespeichert in Merkmalsvektor

- **Idee der Umgebungsmodelle:**
 - Speicherung möglichst aller bereits gesammelter Informationen über die Umgebung
 - Nutzung geeigneter Datenstrukturen wie z.B. Landkarten, etc.

Nutzung von Umgebungsmodellen

Beispiel einer Implementierung für die Gitterwelt

(i) Architektur:



Umgebungsmodelle: Beispiel (2)

(ii) Modell der Umgebung:

1	1	1	1	1	?	?
1	0	0	0	0	0	?
1	0	0	0	0	0	?
1	0	0	R	0	0	?
1	0	0	0	0	0	?
1	?	?	?	?	?	?
?	?	?	?	?	?	?

1: belegt
0: frei
?: unbekannt
R: Roboter

Mögliche Aktion basierend auf diesen Informationen:
Go west (or north) and follow wall

Umgebungsmodelle: Beispiel (3)

(iii) Aktionen

- Aktionen können z.B. über zwei-dimensionale Potential-Felder bestimmt werden.
- Potential-Felder werden aus der Überlagerung von anziehenden (,attractive‘) und abstoßende (,repulsive‘) Komponenten gebildet.
- Die Bewegung des Roboters erfolgt in abstei-gende Richtung des Gradienten (lokale Minima!).
- Die Bewegungsrichtung kann vorberechnet oder online (in z.B. sich ändernden Umgebungen) bestimmt werden.

Umgebungsmodelle: Beispiel (4)

- Potentialfeld für das Gridworld-Beispiel:

- Anziehende Komponente:

- Durch das Zielfeld erzeugt: $p_a(\vec{x}) = k_1 \cdot d(\vec{x})^2$

wobei k_1 ein konstanter Faktor und d der Abstand zum Zielfeld ist.

- Abstoßende Komponente(n):

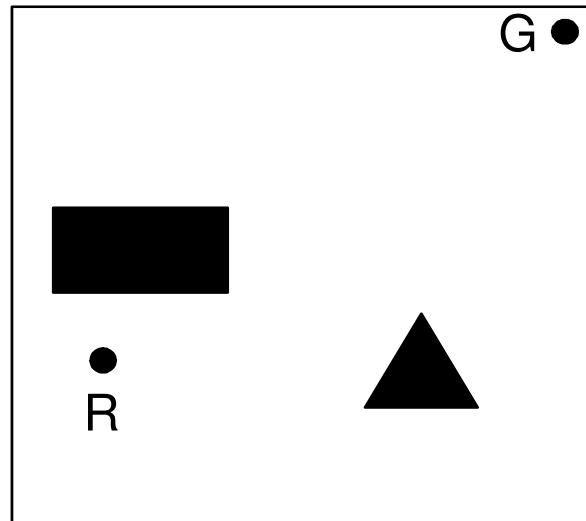
- Durch Hindernisse erzeugt: $p_r(\vec{x}) = \frac{k_2}{d(\vec{x})^2}$

wobei k_2 ein konstanter Faktor und d der Abstand zum Hindernis ist.

- Insgesamt: $p = p_a + p_r$

Potentialfelder: Beispiel

■ Beispiel einer Umgebung eines Roboters:



Roboter (R)

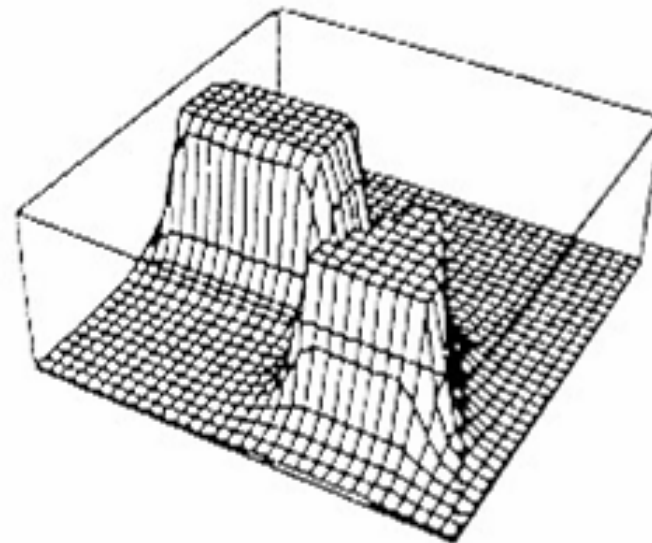
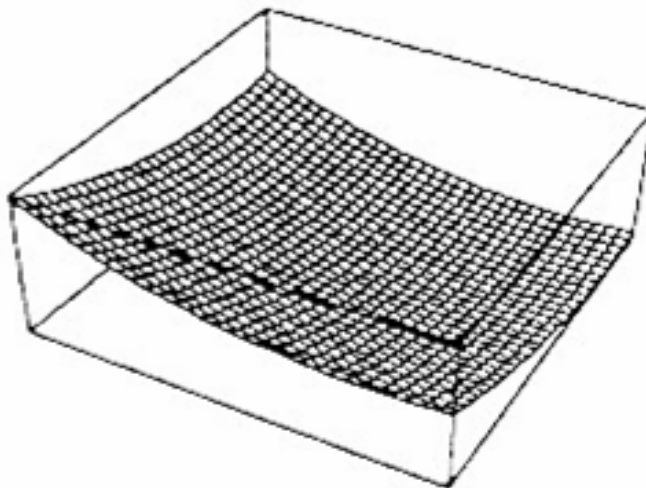
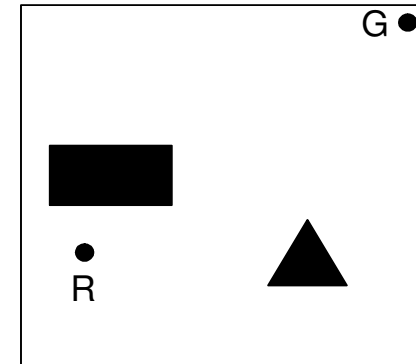
Ziel (G)

Hindernisse (◆)

Potentialfelder: Beispiel (2)

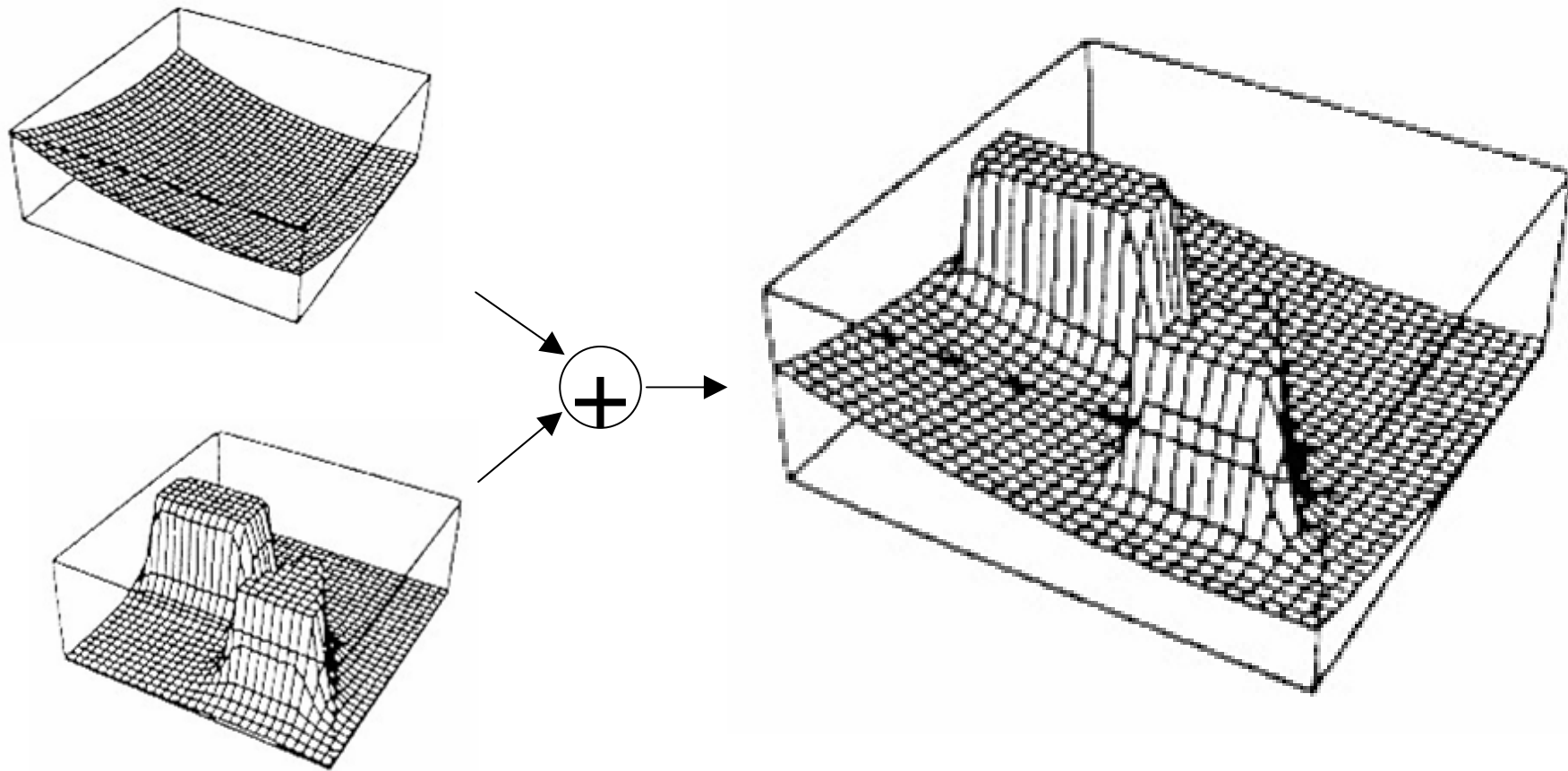
Potentialfeldkomponenten:

- Ziels (links)
- Hindernisse (rechts)



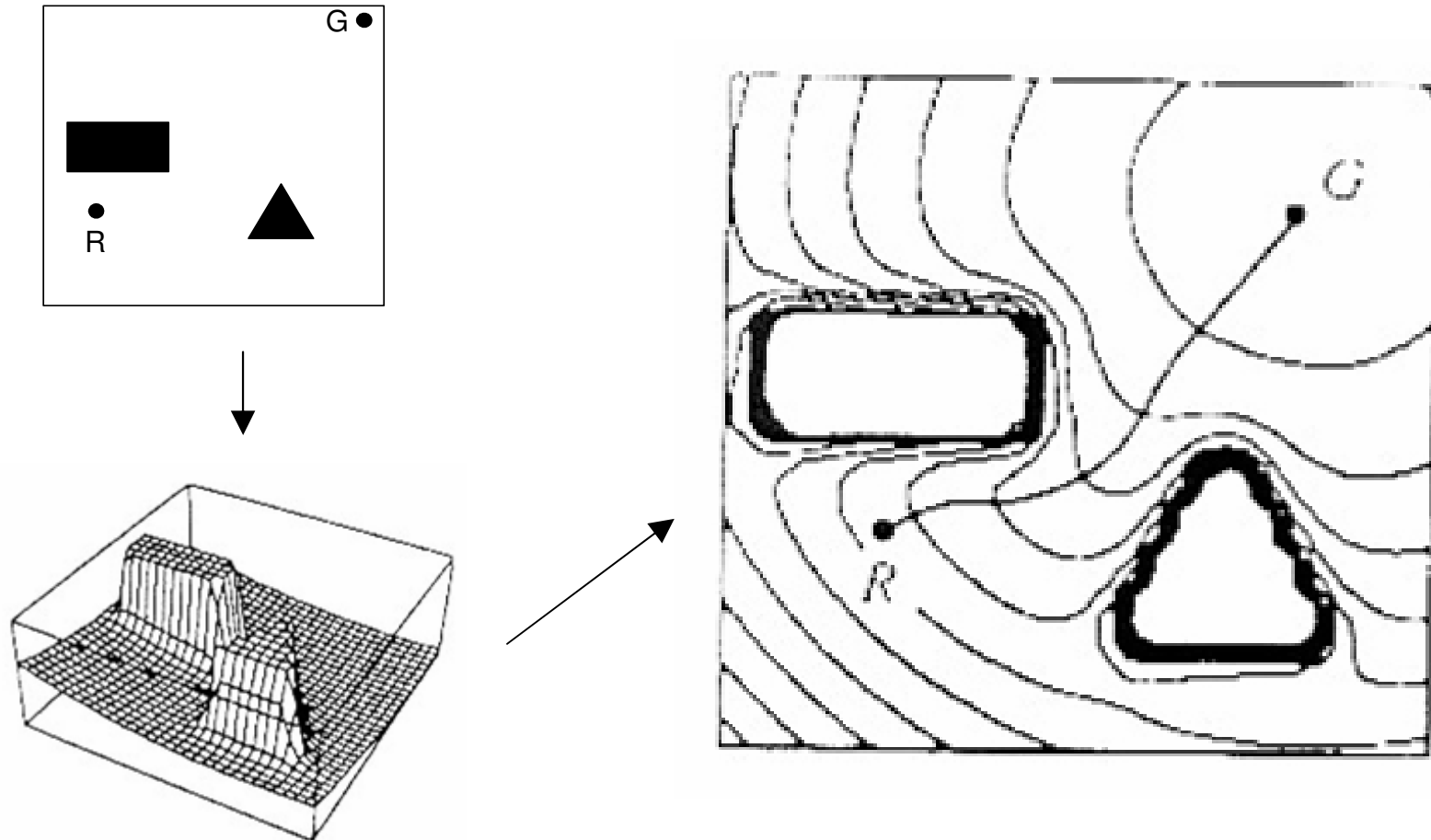
Potentialfelder: Beispiel (3)

■ Gesamtes Potentialfeld:



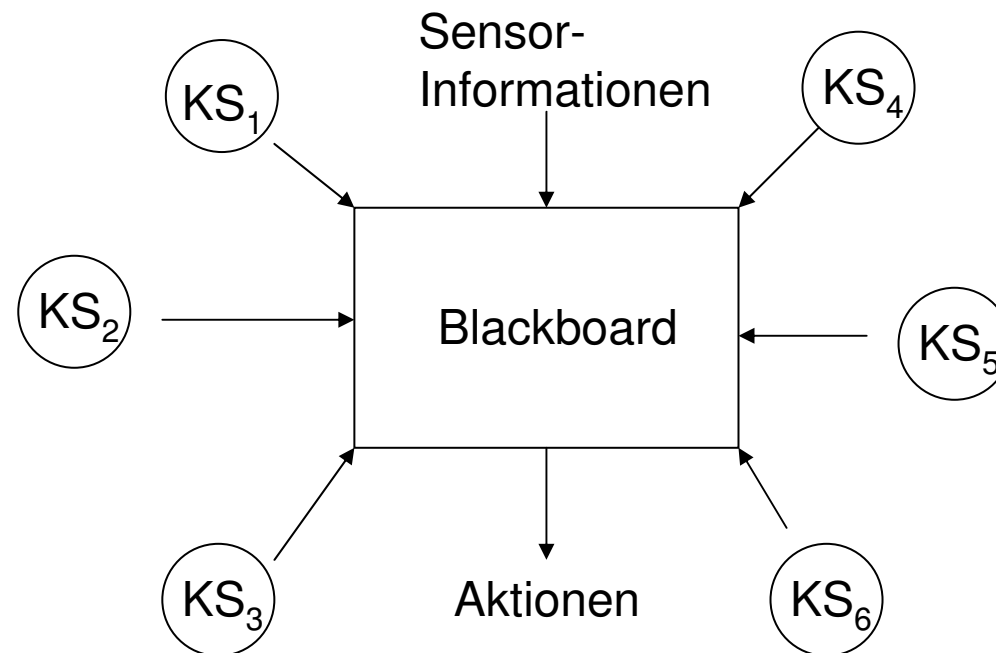
Potentialfelder: Beispiel (4)

■ Äquipotentiallinien (für Gradientenverfahren):



6.4 Blackboard-Systeme

- Blackboard: **Spezielle Datenstruktur**
- Knowledge Source (KS): **Programm, das das Blackboard lesen und verändern kann**



Blackboard-Systeme

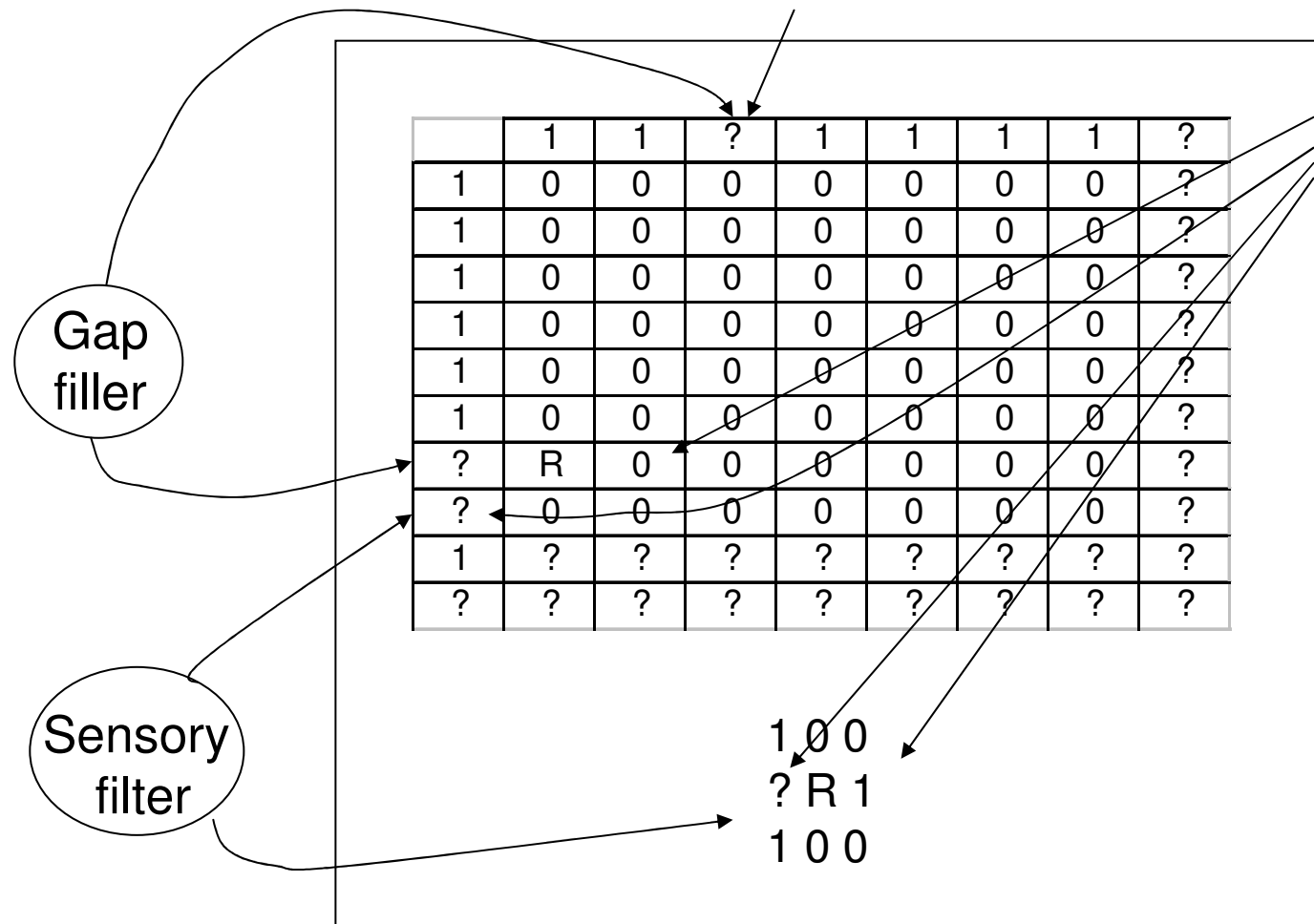
- Jede Knowledge Source hat einen
 - Bedingungsteil (berechnet Wert eines Merkmals) und einen
 - Aktionsteil (Programm, das die Datenstruktur ändert und/oder externe Aktionen durchführt)
- Wenn zwei Knowledge Sourcen ausgeführt werden können, entscheidet ein Konfliktlöser, welche KS gewählt wird.
- Jede Knowledge Source dient als „Experte“ des Blackboardteils, den sie überwacht.

Blackboard-Systeme: Beispiel

- **Das Weltmodell im Roboter kann unvollständig oder falsch sein (wegen Sensorfehlern).**
- **Mögliche Knowledge Sourcen**
 - **Lückenfüller (Gap-Filler):**
Sucht nach engen Räumen (tight spaces) im gelernten Umgebungsmodell und markiert das Feld bzw. korrigiert ggf. vorhandene Fehler.
 - **Sensorfilter (Sensory-Filter):**
Vergleicht Sensorinformationen mit dem gelernten Umgebungsmodell (Karte) und versucht Fehler zu beseitigen.

Blackboard-Systeme: Beispiel (2)

Enger Raum? Hier wird eine 1 eingefügt!



Widerspruch
Modell-Sensor-
Informationen?

Mögliche Aktio-
nen der KSs:

- Sensor setzt 1
- Gap Filler
- $s_4=1$: fehlerhaft

Blackboard