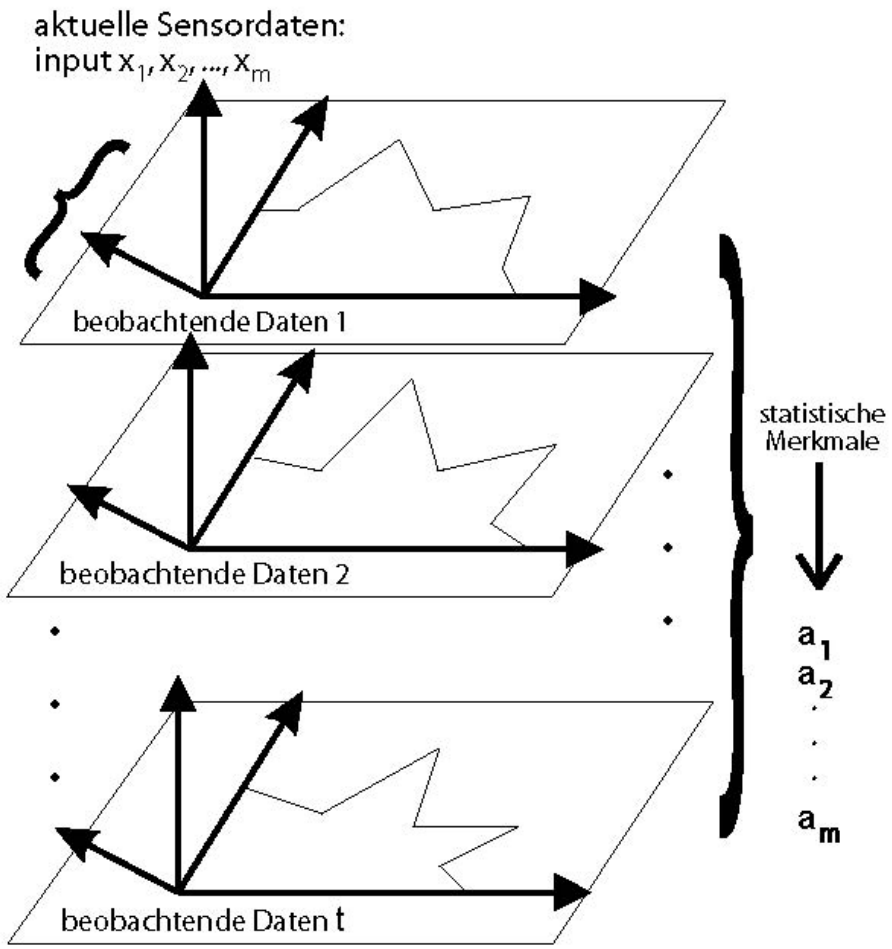
Methode der Dimensionsreduzierung

- **Merkmals-Extraktion:** Mustererkennungsmethoden, üblich, aber die Wahl der Merkmale ist immer problemspezifisch, und der Vorgang ist rechenaufwendig.
- **Input Selection:** eine experimentelle Methode zur Findung der wichtigsten Eingangsvariable aus einer grossen Anzahl.
- **Hierarchie:** Die Eingänge werden für Gruppen bestimmt, Zwischenvariablen werden benutzt für die nächste Hierarchie.
- **PCA und Variante(**$Principal\ Component\ Analysis$**):** Nutzung von statistischen Merkmalen zur Findung der effektivsten Repräsentation der Eingangsinformation.

Vertikale gegen Horizontale Merkmale



Nutzung von statistischen Merkmalen

Sei eine Serie von Sensordaten über eine Szene kontinuierlich in einem eingeschränkten Zeitintervall erfasst, werden die Projektionen dieser Sensordaten im Eigenraum eine geschlossene Mannigfaltigkeit bilden.

Darauf basierend kann eine Dimensionsreduzierung erreicht werden.

Statistische Merkmale

Eine Sammlung von hochdimensionalen Sensordaten in der Beobachtungsphase: $\mathbf{x} = [x_1, x_2, \ldots, x_t]^T$

Die Daten werden normalisiert: $\hat{\mathbf{x}}_j = \mathbf{x}_j / \|\mathbf{x}_j\|$

$$\{\hat{\mathbf{x}}_1, \ldots\ldots, \hat{\mathbf{x}}_2, \ldots\ldots, \hat{\mathbf{x}}_M\}$$

Das Durchschnittbild μ wird abgezogen und

eine Bildmatrix kann konstruiert werden:

$$\mathbf{P} \stackrel{\triangle}{=} \{\hat{\mathbf{x}}_1 - \mu, \hat{\mathbf{x}}_2 - \mu, \ldots\ldots, \hat{\mathbf{x}}_M - \mu\}$$

$$\mathbf{Q} \stackrel{\triangle}{=} \mathbf{P}\mathbf{P}^T$$

$$\lambda_k \mathbf{e}_k = \mathbf{Q}\mathbf{e}_k$$

Ein Punkt im Eigenraum, der mit K Eigenvektoren konstruiert wird, kann folgendermassen dargestellt werden:

$$\mathbf{f}_j = [\mathbf{e}_1, \mathbf{e}_2, \ldots\ldots, \mathbf{e}_K]^T (\hat{\mathbf{x}}_j - \mu)$$

Eigenraum und PCA

Jedes Sensorbild kann mit seiner Projektion so dargestellt werden:

$$\hat{\mathbf{x}}_m = \sum_{i=1}^{t} f_{mx}\mathbf{e}_x + \mu$$

$\hat{\mathbf{x}}_m$ kann approximiert werden durch:

$$\hat{\mathbf{x}}_m \approx \sum_{i=1}^{K} f_{mx}\mathbf{e}_x + \mu$$

Für die Summe der quadratischen Differenz zwischen zwei Sensorbildern gilt:

$$\begin{aligned}\|\hat{\mathbf{x}}_m - \hat{\mathbf{x}}_n\|^2 &= (\hat{\mathbf{x}}_m - \hat{\mathbf{x}}_n)^T(\hat{\mathbf{x}}_m - \hat{\mathbf{x}}_n) \\ &= 2 - 2\hat{\mathbf{x}}_m^T\hat{\mathbf{x}}_n\end{aligned}$$

$$\|\hat{\mathbf{x}}_m - \hat{\mathbf{x}}_n\|^2 \approx \|\sum_{i=1}^{K} f_{mx}\mathbf{e}_x - \sum_{i=1}^{K} f_{nx}\mathbf{e}_x\|^2$$

$$\begin{aligned}\|\sum_{i=1}^{K} f_{mx}\mathbf{e}_x - \sum_{i=1}^{K} f_{nx}\mathbf{e}_x\|^2 &= \|\sum_{i=1}^{K}(f_{mx} - f_{nx})\mathbf{e}_x\|^2 \\ &= \|\mathbf{f}_m - \mathbf{f}_n\|^2\end{aligned}$$

$$\|\hat{\mathbf{x}}_m - \hat{\mathbf{x}}_n\|^2 \approx \|\mathbf{f}_m - \mathbf{f}_n\|^2$$

Implementierung von PCA am Beispiel von Imagedaten

Jedes der k Trainingsbilder wird als ein Vektor $\vec{x^i}$ interpretiert, wobei die Pixelspalten gestapelten werden.

Sei k Eingangsvektoren $\vec{x^1}, \ldots, \vec{x^k}$ mit $\vec{x^i} = (x_1^i, \ldots, x_m^i)$ die gestapelten Bildervektoren.

Die PCA-Methode kann wie folgt eingesetzt werden:

Zuerst wird der Mittelwert $\vec{\mu}$ und die Kovarianz-Matrix $\mathbf{Q}$ dieser Vektoren wie folgt berechnet:

$$\mathbf{Q} = \frac{1}{k}\sum_{i=1}^{k}(\vec{x^i} - \vec{\mu})(\vec{x^i} - \vec{\mu})^T, mit \;\; \vec{\mu} = \frac{1}{k}\sum_{i=1}^{k}\vec{x^i}e$$

Die Eigenvektoren und die Eigenwerte können berechnet werden durch:

$$\lambda_i \vec{a}_i = \mathbf{Q}\vec{a}_i$$

wobei λ_i die m Eigenwerte sind und $\vec{a}_i$ die m-dimensionale Eigenvektoren der $\mathbf{Q}$ sind.

Eine Transformationsmatrix kann wie folgt definiert werden:

$$\mathbf{A} = (\vec{a}_1 \ldots \vec{a}_n)^T$$

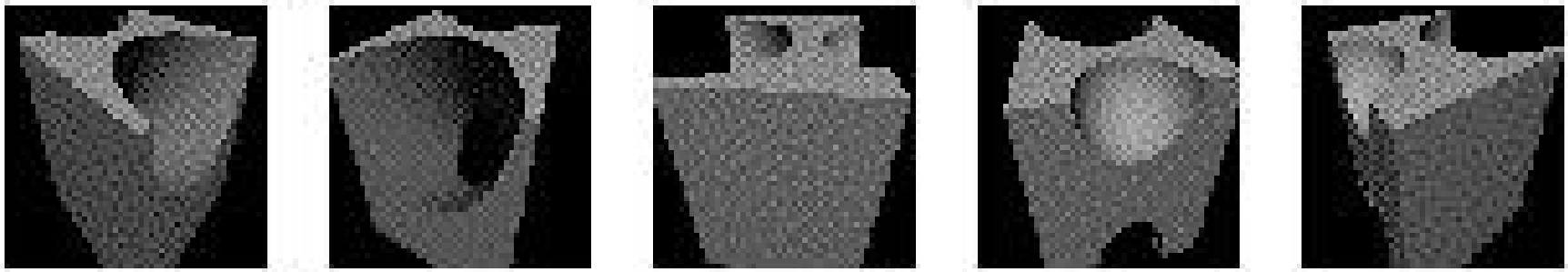
Die Dimension der $\vec{x^i}$ kann reduziert werden durch:

$$\vec{p^i} = \mathbf{A} \cdot \vec{x^i}; \;\; dim(\vec{p^i}) = l$$

Die Dimension l soll mit Berücksichtigung der Genauigkeit und der Rechenkomplexität bestimmt werden.

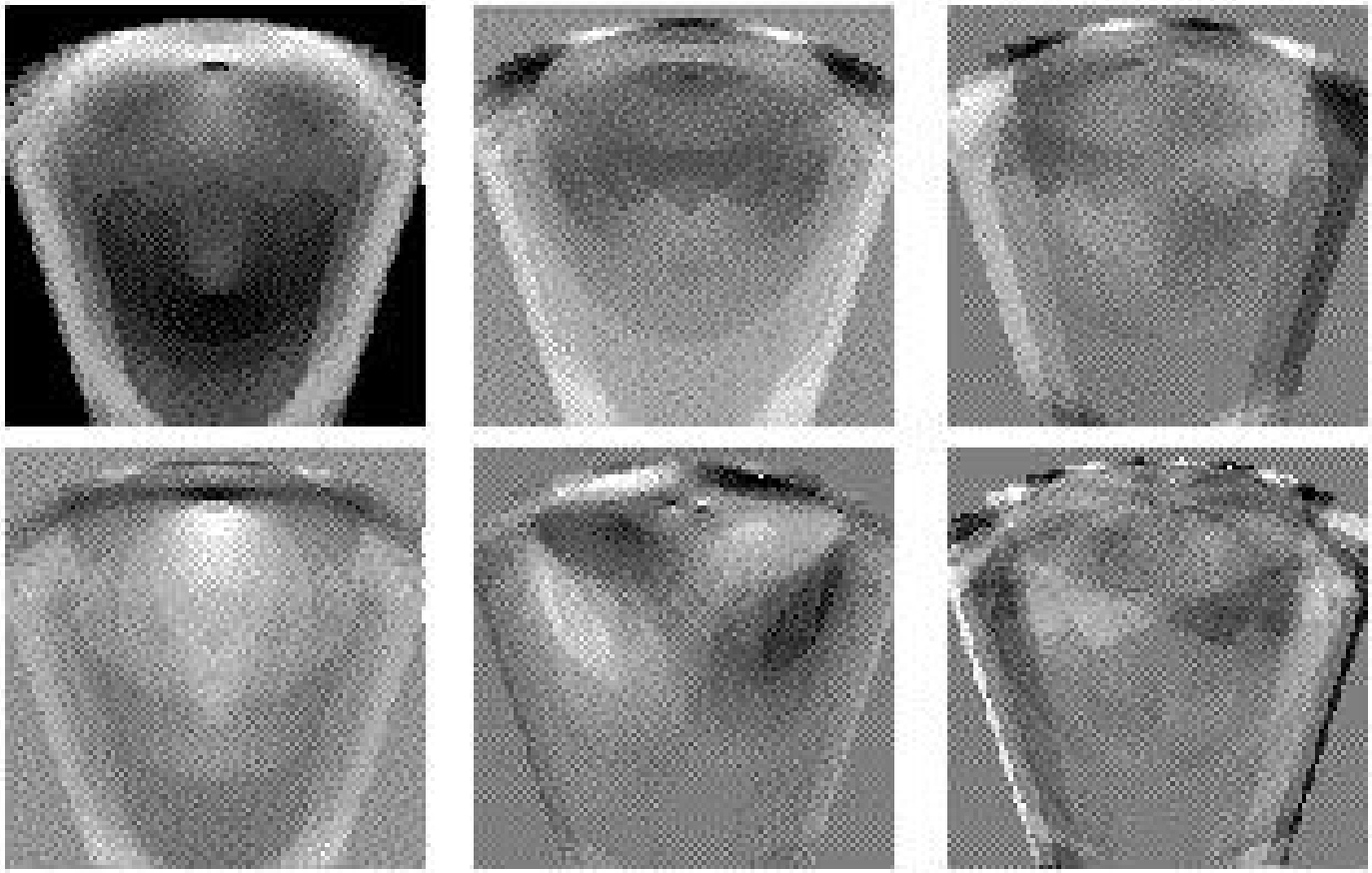
Beispiel: Bilderbeobachtung in lokalen Szenarien

Eine Sequenz von Grauwertbildern über ein Objekt wird durch die Bewegung entlang einer festen Lage erzeugt:

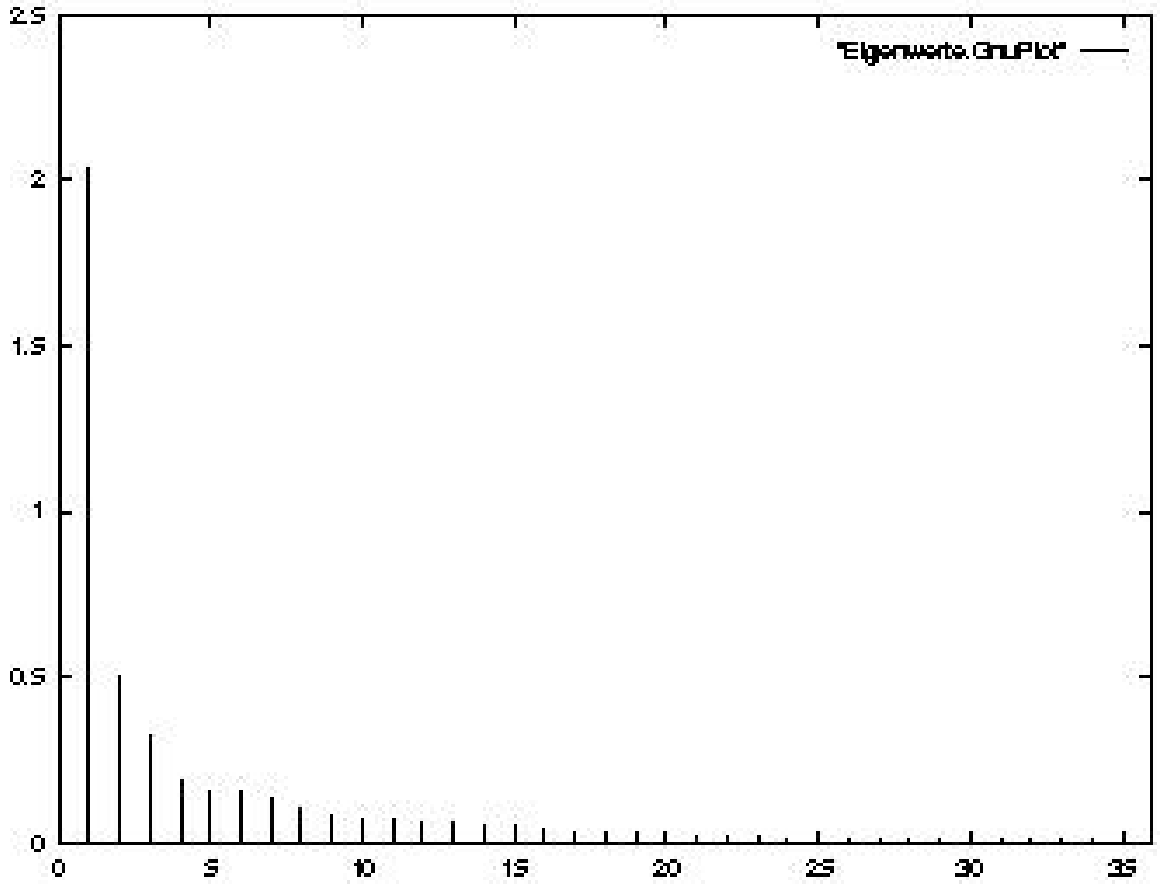


Beispiel: Extrahierung von Eigenvektoren

Die ersten 6 Eigenvektoren:



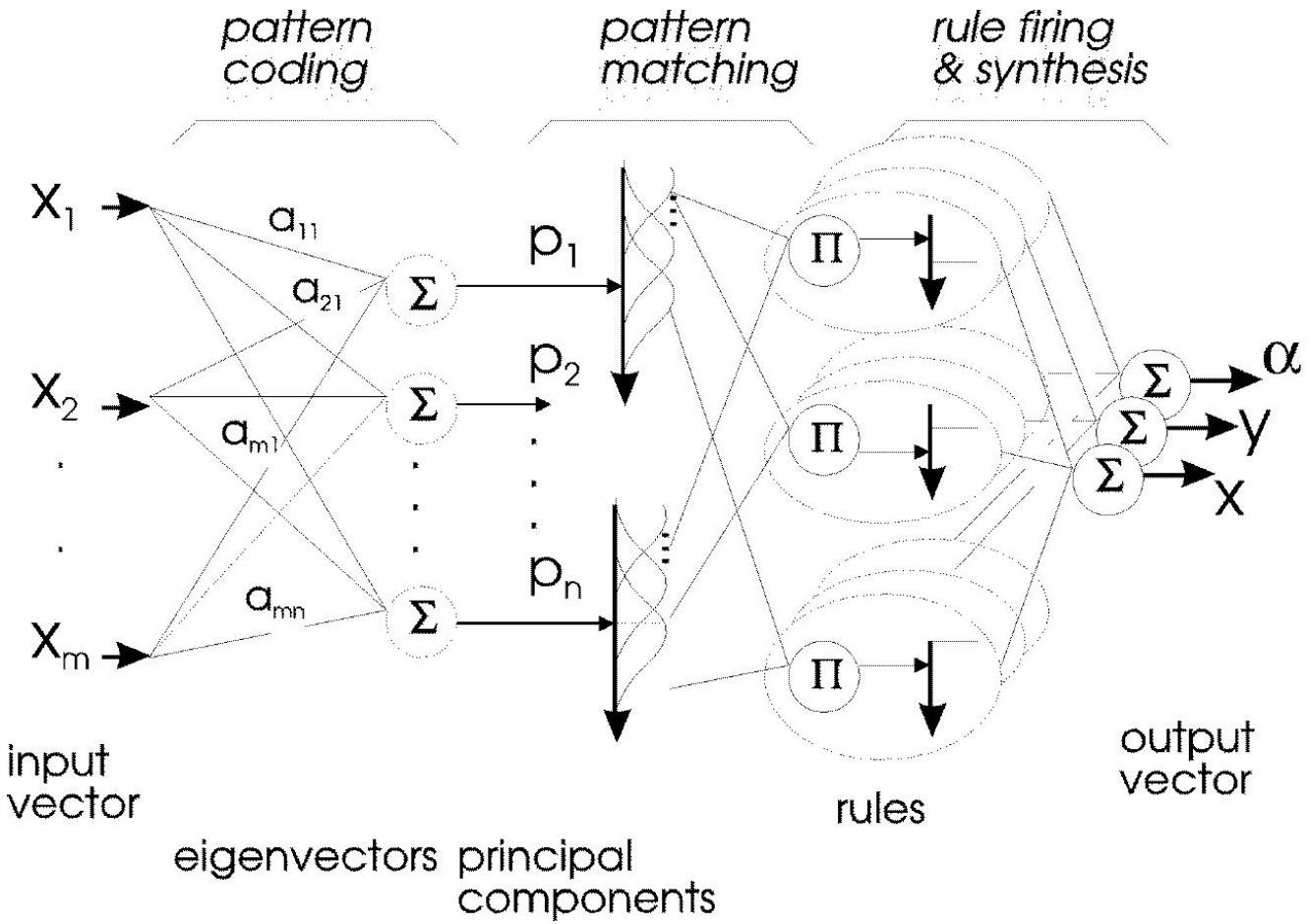
Beispiel: Eigenvektoren und Eigenwerte



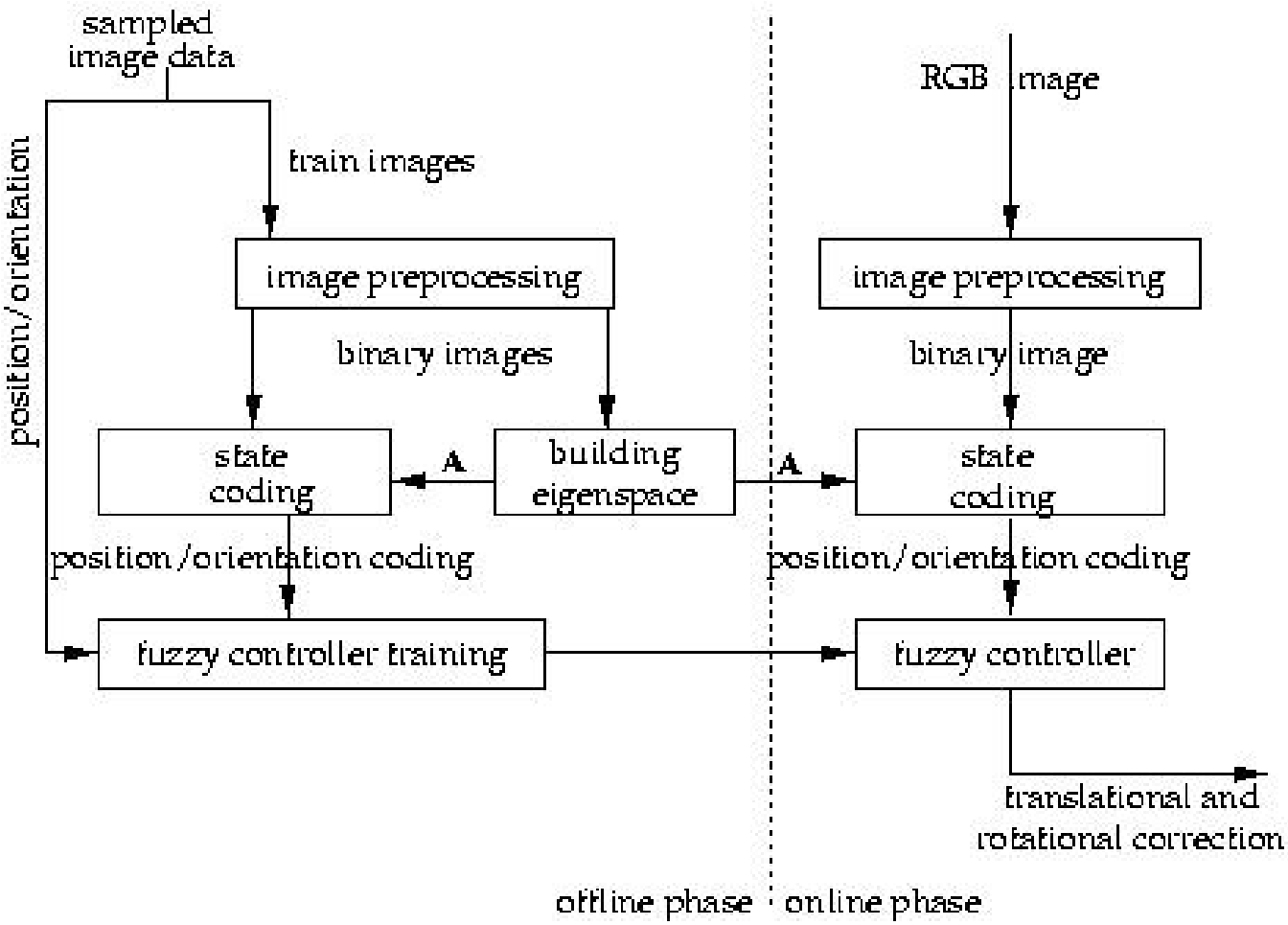
Kombination von Dimensionsreduzierung mit einem B-Spline-Model

Eigenvektoren können durch Linguistische Terme partitioniert werden.

Solche eine Kombination von PCA und B-Spline-Modell kann als ein Neuro-Fuzzy-Modell betrachtet werden.


pattern coding
pattern matching
rule firing & synthesis
x_1
x_2
x_m
a_{11}
a_{21}
a_{m1}
a_{mn}
p_1
p_2
p_n
α
y
x
input vector
eigenvectors
principal components
rules
output vector

Die Trainierungs- und Anwendungsphasen



Ausgangsrelevante Merkmale

Probleme können auftauchen wenn die Eingangsvariablen eine kleine Varianz besitzen aber für das System nicht signifikant sind.

Dann werden eine grosse Anzahl von Eigenvektoren benötigt.

Eine Lösung ist die Nutzung von einer Menge von Vektoren, die direkt mit dem Ein- und Ausgang korreliert sind.

Diejenige Merkmale, die den Ausgang beeinflussen, werden $Output\ Related\ Features$ (ORF) benannt.

Die ORFs können mit Hilfe von einer Trainierung mit $Hebbian\ learning\ rules$ extrahiert werden.

Seien die Trainingsdaten x_j $(j = 1, \ldots, k)$.

Sei ein ORF Gewichtvektor trainiert und als $\vec{a}$ bezeichnet, dann ist der Netzwerkausgang P:

$$P = \sum a_j x_j = \vec{a}^T \vec{x} = \vec{x}^T \vec{a}.$$

Wenn sowohl Eingangsdaten x als auch die Solldaten Y_S vorhanden sind, dann kann ein Element a_j des Gewichtsvektors $\vec{a}$ wie folgt modifiziert werden:

$$\Delta a_j = \eta(Y_S - P)x_j \tag{1}$$

wobei η der Lernschritt ist.

Die Berechnung mehrerer ORF Gewichtsvektoren $\vec{a_i}, (i = 1, 2, \dots)$ basiert auf dem ersten ORF weight vector (i=1) Gleichung (**??**).

Zur Berechnung weiterer $\vec{a_i}(i > 1)$, werden alle Eingangsdaten auf den letzten ORF-Vektor projeziert, d.h. i.e. $\vec{a}_1, \ldots, \vec{a}_{i-1}$, womit die Komponenten des Eingangsvektors, die parallel zum ORF-Vektor liegen, berechnet werden.

Diese Komponente werden aus dem Eingang abgezogen.

Das Element a_{ij} des Vektors $\vec{a_i}$ kann dann gelernt werden durch:

$$\Delta a_{ij} = \eta \left(Y_S - P_i\right) \left(x_j - \sum_{k=1}^{i-1} P_k a_{kj} \right) .$$