# Kalmanfilter und Nichtparametrische Filter

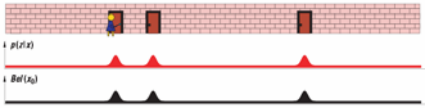
Daniel Göhring

06.11.2006

# Bayesfilter

- Zustand zur Zeit t ist Zufallvariable $x_t$
- Wahrscheinlichkeitsfunktion Bel (Belief) über $x_t$ : $Bel(x_t)$ repräsentiert die Unsicherheit

Quelle: Thrun, Burgard, Fox

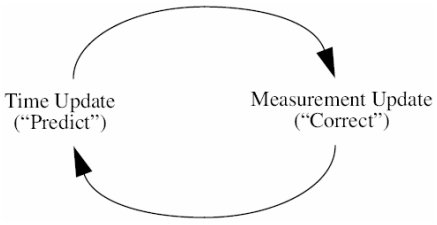


- $Bel(x_t) = p(z_1,z_2, \dots , z_t)$, aber es wird Markov angenommen, $x_{t-1}$ reicht für die Folgezustandsberechnung
- $Bel^-(x_t) \leftarrow \int p(x_t|x_{t-1})Bel(x_{t-1})dx_{t-1}$ (predict)
- $Bel(x_t) \leftarrow \alpha_t p(z_t|x_t)\, Bel^-(x_t)$ (correct)
  - $\alpha_t$ ist Normierungskonstante $z_t$ sind Sensordaten

# Kalmanfilter

- Parametrischer Filter
- Verwendet Gaußfunktion zur Repräsentation der
  - Objektposition sowie
  - Sensormodell p(z|x‘)
  - Zustandsübergangsfunktion p(x‘|u,x)

# Arbeitsweise Kalman




1. Schritt: Vorausberechnung der neuen Ballposition aus der alten Position (Modellierung)
2. Korrektur des berechneten Wertes durch die Messung

# Das Kalmanmodell

1. Vorausberechnung (Prediction):

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}$$

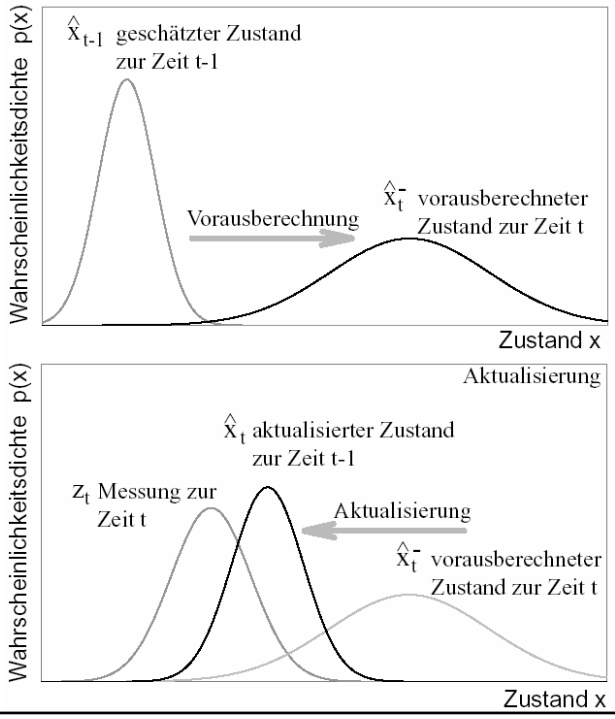
mit A gelangt man vom Zustand in k-1 zum Zustand in k
B repräsentiert die Kontrollinputmatrix

2. Messung (Measurement):

$$z_k = Hx_k + v_k$$

H beschreibt den Zusammenhang zwischen realem und gemessenem Zustand

$w_k$ , $v_k$ repräsentieren das Prozess- bzw. Messungsrauschen (weiß, normalverteilt)

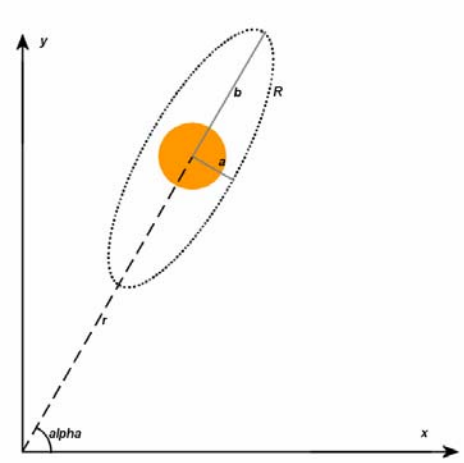


# Kalmanmatrizen

- **Q** repräsentiert die Prozessfehlerkovarianzen, je größer die Werte von Q, je stabiler arbeitet der Filter
- **R** repräsentiert die Messfehlerkovarianzen, je größer die Werte von R, je reaktiver arbeitet der Filter
- $P_k$ repräsentiert die Diskrepanz zwischen Messung und Berechnung in k, jeweils a priori und a posteriori, wird für jeden Schritt neu berechnet

Q, R müssen dem Modell mitgeteilt werden, beiben const., $x_k$ wird Anfangs auf $z_k$ initialisiert

# Beispiel für Messfehlerkovarianzmatrizen **R**



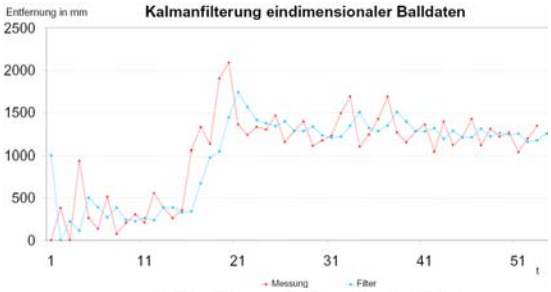

- Entfernungsmessung für den Roboter schwieriger als Abschätzung des Winkels

- Korrelation der einzelnen Messgrößen beachten

## Auswirkung von Q bzw. R auf die Filterwirkung

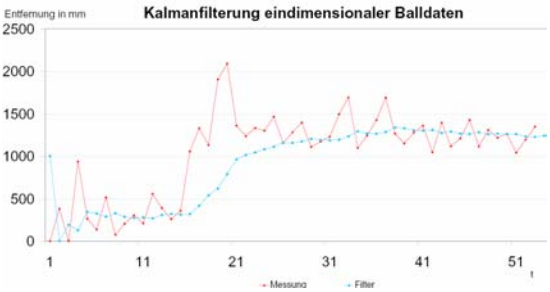
R klein

Q groß




R groß

Q klein




# Berechnungsschleife

Prediction:

$$x^-_k = Ax_{k-1} + Bu_{k-1}$$

$$P^-_k = AP_{k-1}A^T + Q$$ //a-priori Fehlerkovarianzmatr.

Update:

$$K_k = P^-_k H^T (HP^-_k H^T + R)^{-1}$$

$$x_k = x^-_k + K_k(z_k - Hx^-_k)$$

$$P_k = (I - K_k H)P^-_k$$ //a-post. Fehlerkovarianzmatr.

K wird gemeinhin als das Kalmangain bez.

## Probleme

- Messfehlerkov.-mat. R ist leicht aus Eingangsdaten zu ermitteln
- für verschiedene Ballentfernungen
- Prozessfehlerkov.-matrix hingegen schwierig
- Deckenkamera notwendig als Kontrolle von außen
- keine Repräsentation von Negativinformation möglich

## Probleme (2)

- Messfehler nicht konstant
  - Entfernungsfehler steigt quadratisch mit wachsender Ballentfernung
  - Winkelfehler steigt bei Kopfbewegung
- also Messfehlerkovarianzmatrix **R** variabel gestalten

Wdh.: Update: $K_k = P^-_k H^T (H P^-_k H^T + \mathbf{R})^{-1}$

## Implementationsbeispiel

- Eindimensionaler Kalmanfilter für x:

Bsp.: Für x:

$S_x^- = S_x + V_x$

$P_x^- = P_x + \mathbf{Q_x}$

$K_x = P_x^- * (P_x^- + \mathbf{R_x})^{-1}$

$S_x = S_x^- + K_x * (Z_x - S_x^-)$

$P_x = (I - K_x) * P_x^-$

## Stärken von KF

- Optimale Objektzustandsabschätzung für lineare Prozesse
- Schnelle Berechenbarkeit, auch für höherdimensionale Probleme (1000 Dimensionen)

## Schwächen von KF

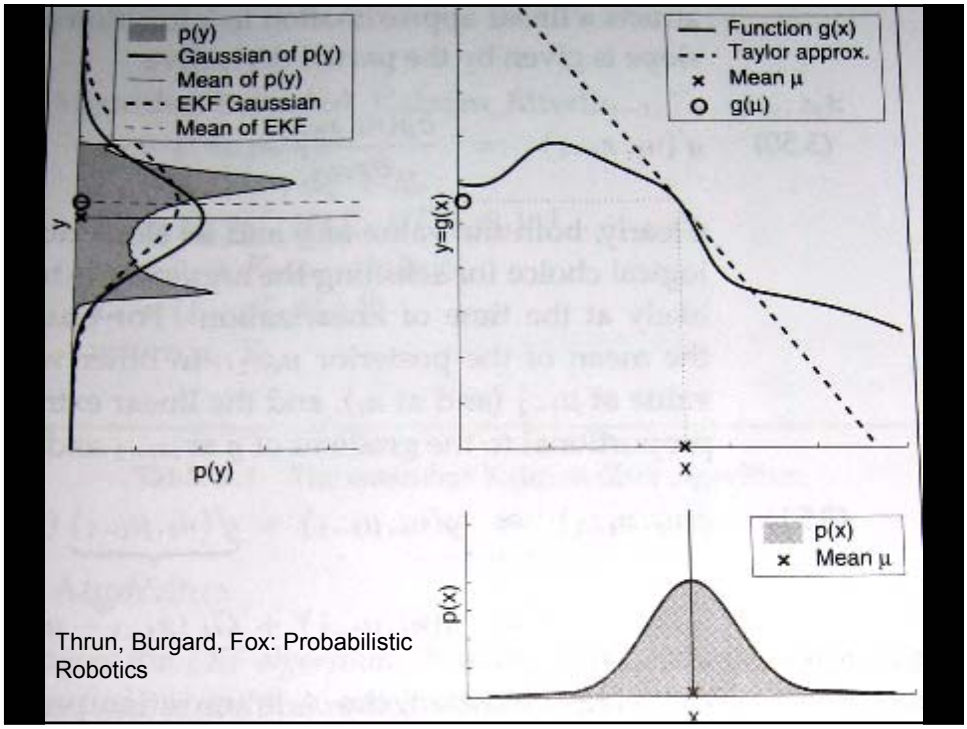
- Beschränkt Normalverteilungen
  - Nur einzelne Objekte können verfolgt werden
  - Gaussform der Objekte
- Beschränkt auf lineare Prozesse

Daher: Erweiterungen von Kalmanfiltern

## Extended Kalmanfilter

- sind eine Möglichkeit, Kalmanfilter zu erweitern
- Idee: Linearisierung von gegebenen Verteilungen
- Annahme bei EKF: Zustandsübergangsfunktion und Messungswahrscheinlichkeit sind nichtlineare Funktionen g und h, also kein linearer Zusammenhang zw. x und z:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t$$
$$z_t = h(x_t) + \delta_t$$




Thrun, Burgard, Fox: Probabilistic Robotics

# Linearisierung

- Die Linearisierung des Zusammenhangs bei gegebener Gleichung

$$z_t = h(x_t) + \delta_t$$

  ist gegeben durch die Tangente an h am Mittelwert $\mu_x$ der Gaußverteilung von x
- Entspricht Taylorapproximation 1. Grades von h

## Schwächen

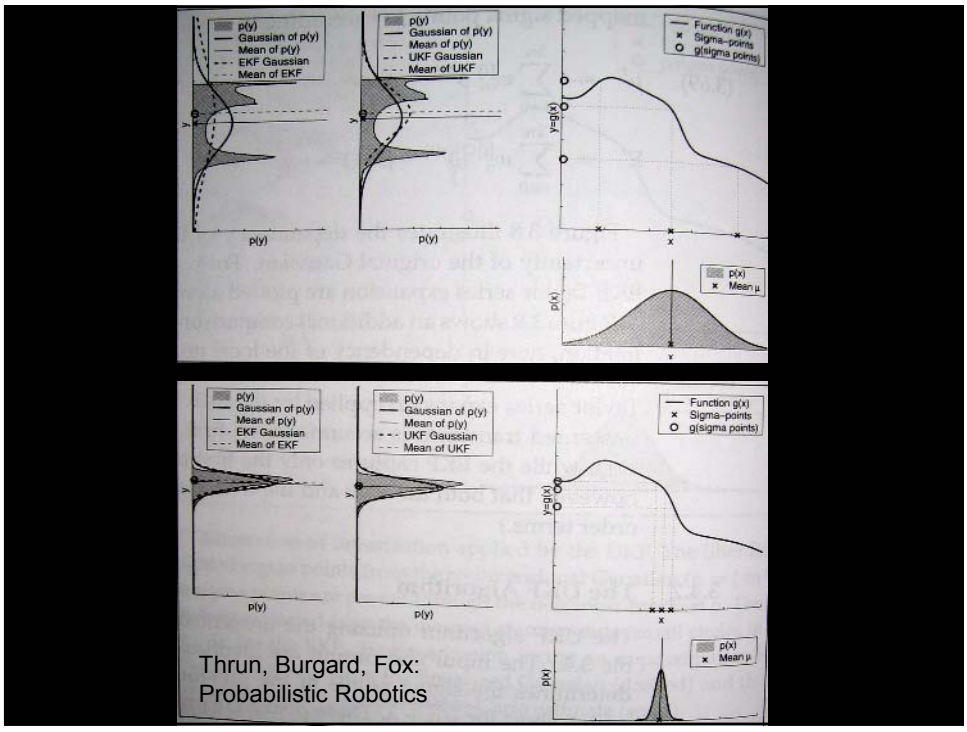
- Linearisierung hängt nur vom Wert von h in $\mu_x$ ab.
- Güte der Linearisierung hängt von der Nichtlinearität von g bzw. h ab

## Erweiterungen

- Verwendung von Summen von Gaußfunktionen, also mehreren EKFs
- Jeder EKF wird dabei gewichtet aufsummiert - MHEKF

# Uncented Kalmanfilter

- Statt der Verwendung der ersten zwei Terme der Taylorentwicklung an der Stelle $\mu_x$ werden werden die Standardabweichungen der Verteilung auf eine andere Art und Weise verrechnet
  - Höhere Genauigkeit der Abschätzung zur Wirklichen Verteilung



## Nichtparametrische Filter

- Nichtparametrische Filter verwenden Repräsentanten zur Darstellung der PDF
- Keine feste funktionale Form der Wahrscheinlichkeitsfunktion
- Repräsentanten können durch endlich viele Regionen (Histogramm) oder durch endlich viele Samples dargestellt werden

## Histogrammfilter

- Zustand wird in endlich viele, gemeinhin kongruente Bereiche unterteilt
- Jeder Bereich erhält nur noch eine Zahl, die die Aufenthaltswahrscheinlichkeit repräsentiert
- Bsp.: Occupancy-Grid
  - Jede Bereich im Raster ist entweder besetzt oder frei – Repräsentation durch Boolsche Variable

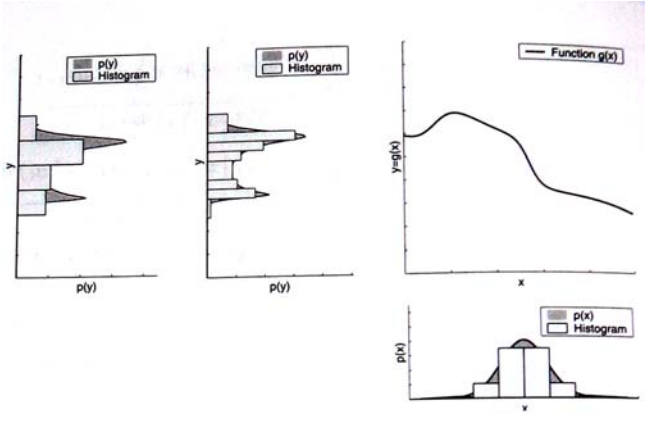
# Diskreter Bayesfilter

$$\underline{p}_{k,t} = \Sigma_i \, p(X_t = x_k \mid u_t, X_{t-1} = x_i) \, p_{i,t-1}$$

$$p_{k,t} = \eta \, p(z_t \mid X_t = x_k) \, \underline{p}_{k,t}$$

# Histogrammfilter

- Unterschiedliche Auflösungen möglich
- Beispiel:



## Partikelfilter

- Repräsentation der PDF durch Samples, oft auch Partikel genannt
- Partikel werden anfangs gleichmäßig in den Zustandsraum eingestreut
- Likelihood gibt an, wie gut die einzelnen Samples mit den Sensordaten (Evidence) übereinstimmen
- Danach Resampling

## Vorteil

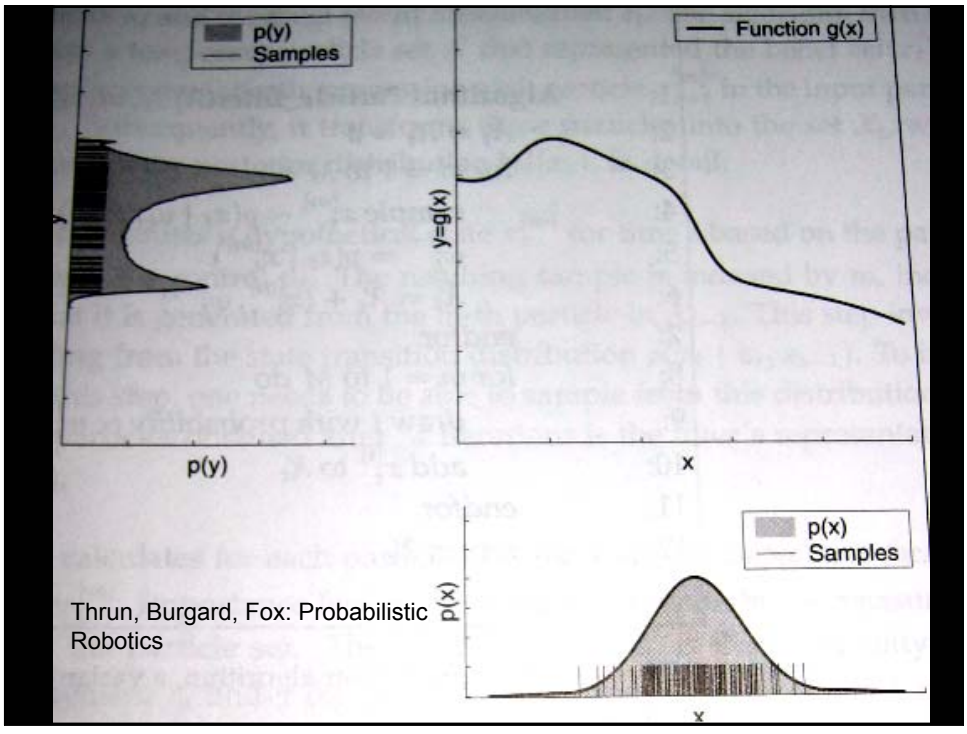
- Kann mehr Verteilungen repräsentieren als z.B. Gaußverteilungen
- Nichtlineare Transformationen der Zufallsvariablen können repräsentiert werden
- Samples der Zustandsabschätzung heißen Partikel

$$X_t := x_t^{[1]},\ x_t^{[2]}, \ldots, x_t^{[M]}$$

- Jeder Partikel $x_t^{[m]}$ ist eine Instanzierung eines Zustandes zur Zeit *t,* M ist die Partikelzahl

# Partikelfilteralgorithmus

1: Alg. Partikel Filter ($X_{t-1}$, $u_t$, $z_t$):
2: $\underline{X}_t = X_t = \{\}$
3: for m = 1 to M do
4: sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1})$
5: $w_t^{[m]} \sim p(z_t \mid x_t^{[m]})$
6: $\underline{X}_t = \underline{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for m = 1 to M
9: draw i with probability $\sim w_t^{[i]}$
10: add $x_t^{[i]}$ to $X_t$
11: endfor
12: return $X_t$



## Berechnungsmöglichkeit des Gewichts bei Sensordaten

$w_t^{[m]}$ = 1, falls Resampling stattfand

$w_t^{[m]}$ = $p(z_t \mid x_t^{[m]}) * w_{t-1}^{[m]}$ , falls Resampling stattfand

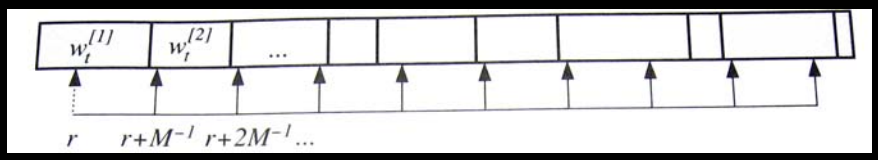
## Resampling

- Manchmal verändert sich der Objektzustand nicht – Roboter bewegt sich nicht, insbesondere, wenn keine Sensordaten vorliegen
- Bei Resampling würden Partikel gelöscht werden
- Motivation für Verfahren, die Varianzreduktionen unterbinden

# Resampling

- Zu viel Resampling kann Diversität reduzieren
- Zu wenig Resampling kann Partikel in Regionen geringen Interesses verschwenden
- Lösung: Resampling an der Varianz der Gewichte Orientieren – alle Gewichte gleich – nicht resamplen
- Andere Verfahren wie Low Variance Resampling

# Low Variance Resampling

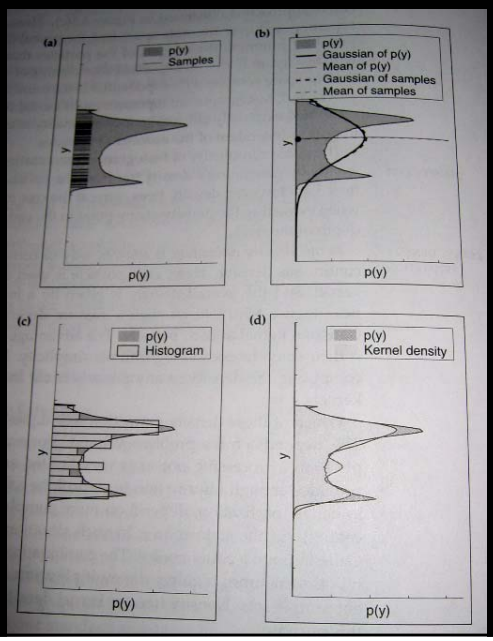



Thrun, Burgard, Fox: Probabilistic Robotics

## Wahrscheinlichkeitsdichteabschätzung

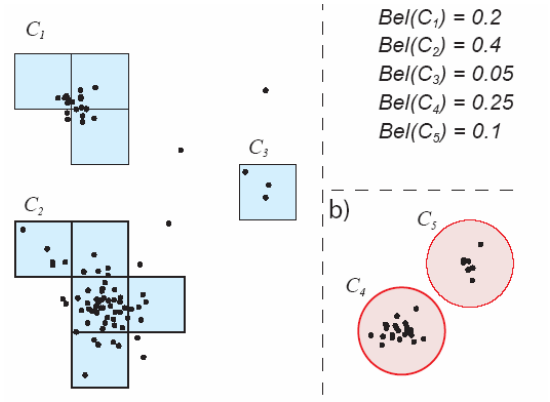
- Schwierigkeit, aus endlich vielen Repräsentanten, kontinuierliche Verteilung zu ermitteln
- Verschiedene Möglichkeiten dafür:
  - Gaußverteilung berechnen – nicht sinnvoll bei multimodalen Verteilungen
  - k-means clustering sinnvoller bei multim. Vert. (Menge von Gaußfunktionen)
    - Diskretes Histogramm über alle Partikel legen
  - Auch Dichtebäume möglich

## Möglichkeiten der Dichteabschätzung




Thrun, Burgard, Fox: Probabilistic Robotics

# Histogramm / Clustering



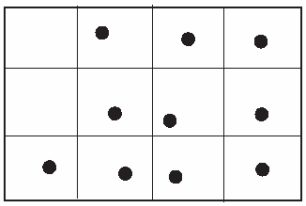

# Maß für die Konvergenz

- Entropie H gibt Auskunft über die Konvergenz der Partikelverteilung
- Vorgehen, Rasterung des Spielfeldes in n Unterbereiche
- $H = \sum -p_i * \log p_i$

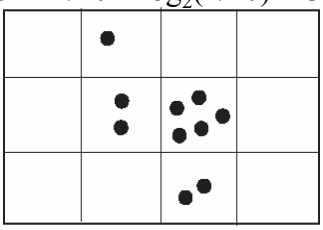
Wobei $p_i$ die Anzahl der Partikel im Bereich i dividiert durch Gesamtpartikelzahl ist (relative Häufigkeit)

# Entropie Beispiel

- Entropie maximal bei Gleichverteilung
- minimal, wenn alle Partikel in einem Raster liegen
  - 0 * log 0 $=_{def}$ 0



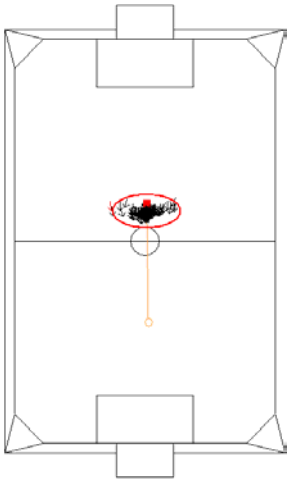
$= 10 * -1/10 * \log_2(1/10) \approx 3.32$



$= -1/10*\log_2(1/10) - 4/10*\log_2(2/10) - 5/10*\log_2(5/10) \approx 1.76$

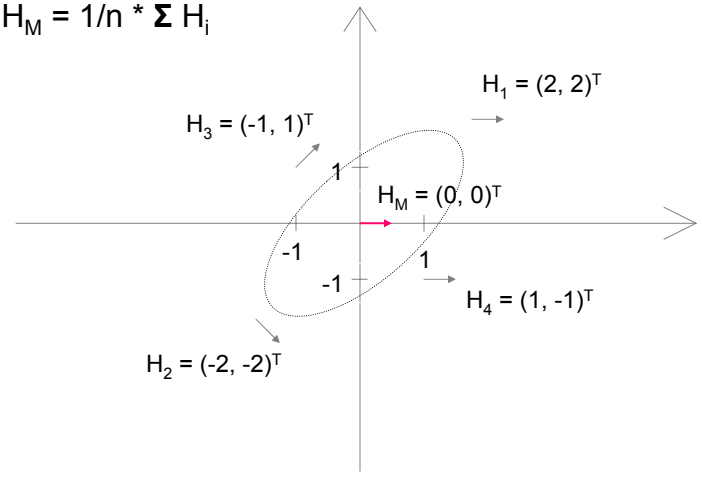
# Aktive Vision - Berechnung der Hauptachsen einer statistischen Verteilung

- Annahme – Normalverteilung
- Man moechte Betrag und Richtung der Hauptachsen wissen
- Ziel: Maximale Verringerung der Unsicherheit



# Mittelwert

$H_M = 1/n * \Sigma H_i$




# Berechnung der Eigenwerte für x und y

Ausgangsmatrix:

- $S = \begin{pmatrix} 2.5 & 1.5 \\ 1.5 & 2.5 \end{pmatrix}$

Gesucht: alle Eigenwerte λ,

wobei gilt: $\det (S - I \lambda) = 0$

# Berechnung der Eigenwerte

- $\det \begin{pmatrix} 2.5 - \lambda & 1.5 \\ 1.5 & 2.5 - \lambda \end{pmatrix} = 0,$

- Also $(2.5 - \lambda)^2 - 1.5^2 = 0$
  $0 = \lambda^2 - 5\lambda + 4$

- Eigenwerte: $\lambda_1 = 2.5 + 1.5 = 4$
  $\lambda_2 = 2.5 - 1.5 = 1$

# Berechnung der Eigenvektoren

- Einsetzen der Eigenwerte λ in die Kovarianzmatrix S, größte Eigenwerte zuerst.

$$(S - I\,\lambda) * \vec{V} = \vec{0}$$

## Berechnung der Eigenvektoren (2)

Berechnung des 1. V, $\lambda = 4$:

$$-1.5 * V_{1x} + 1.5 * V_{1y} = 0$$
$$1.5 * V_{1x} - 1.5 * V_{1y} = 0$$

$$V_{1x} = V_{1y}$$

Eigenvektor normiert (Betrag = 1): $V_1 = \begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix}$

## Normierung der Eigenvektoren

$$V_2 = \begin{pmatrix} -0.707 \\ 0.707 \end{pmatrix}$$

EVs sind Zeilen der Transf.-matrix

Transformationsmatrix $K = (V_1, V_2)^T$

$$K = \begin{pmatrix} 0.707 & 0.707 \\ -0.707 & 0.707 \end{pmatrix}$$

## Transformation (optional)

$$S' = K * S * K^T$$

(K ist orthonormal, $K^T = K^{-1}$)

$$S' = \begin{pmatrix} 4*0.7 & 4*0.7 \\ -1*0.7 & 1*0.7 \end{pmatrix} * \begin{pmatrix} 0.7 & -0.7 \\ 0.7 & 0.7 \end{pmatrix}$$

## Transformation (2)

- Nach der Transformation sind die Kovarianzen null

$$S' = \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}$$

# Zu Beachten

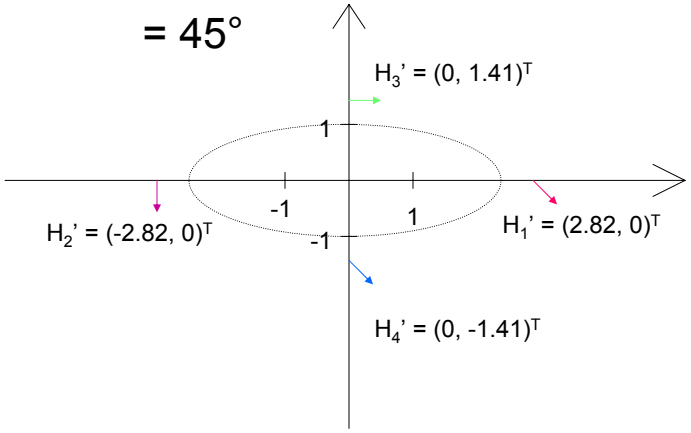
- Rotationsmatrix hat die Form

$$K = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

  Vorzeichen von K können zeilenweise vertauscht werden (Spiegelung der EVs)
- Ellipsenrotation von 200° = 20°, schnellere Erreichbarkeiten (Kosten) beachten

# Nach der Transformation

also: $\alpha$ = arccos(0.707) = arcsin(0.707)
= 45°



# Also

- 45 Grad nach rechts schauen