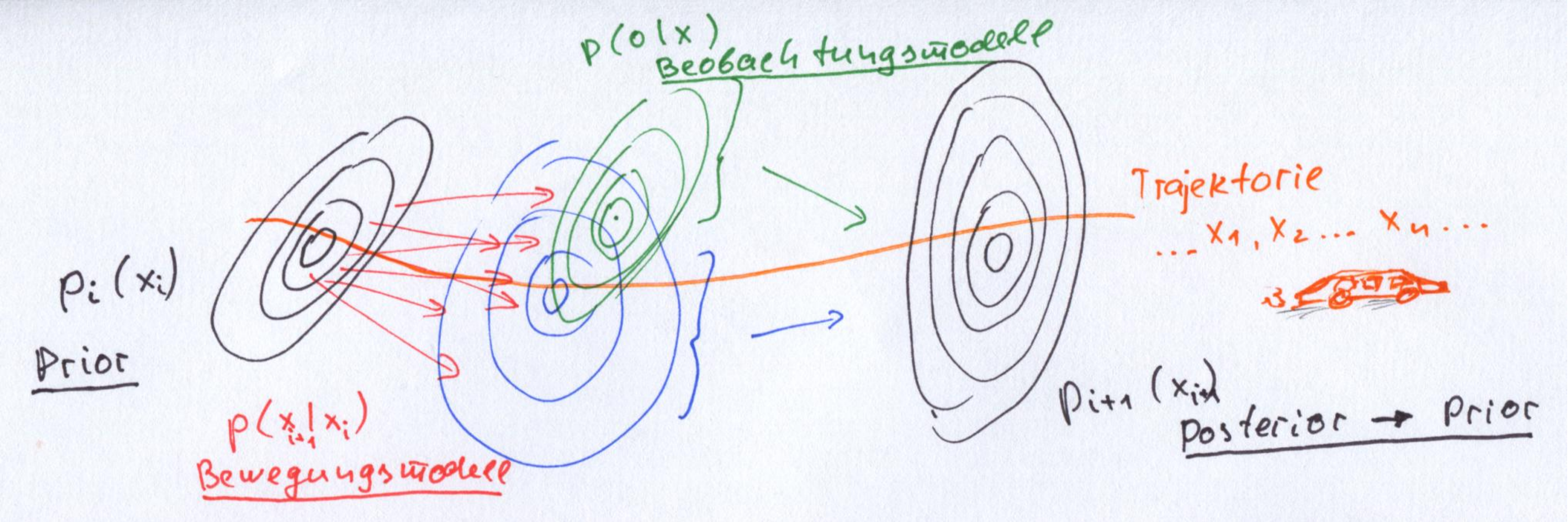
Image Processing

Tracking

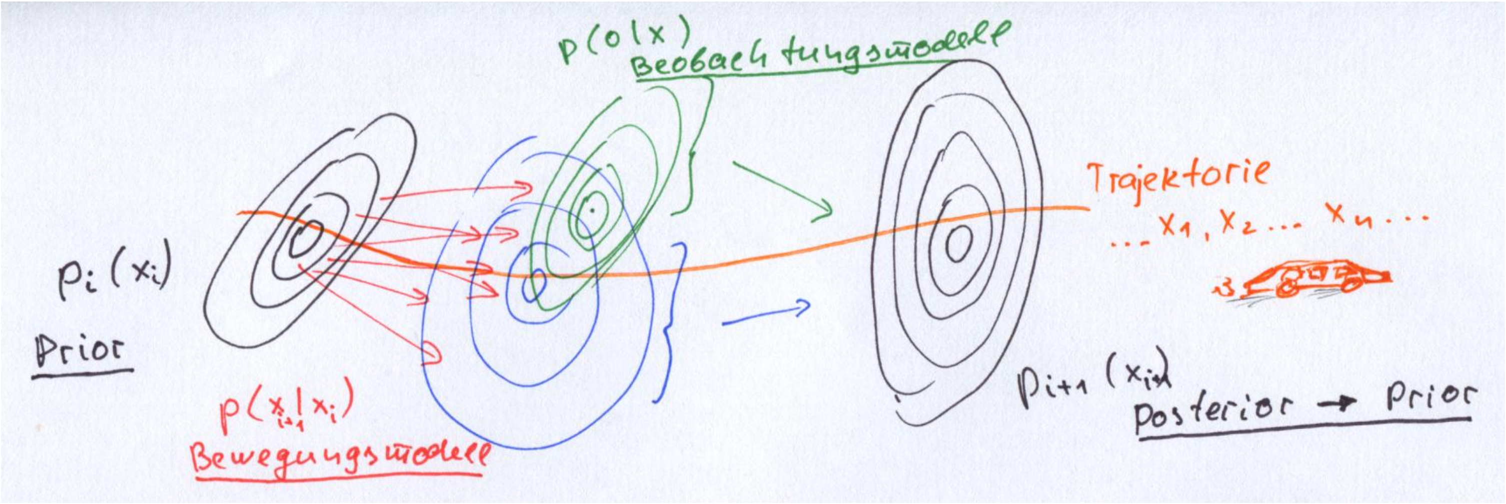
Bayesian Filtering



There is a set of states $\mathcal{X}$, in which an object can stay, i.e. $x \in \mathcal{X}$

There are:

1. Transition model – how the state at the next time is obtained from the state at the previous one
2. Observation (Measurement) model – what and how is observed



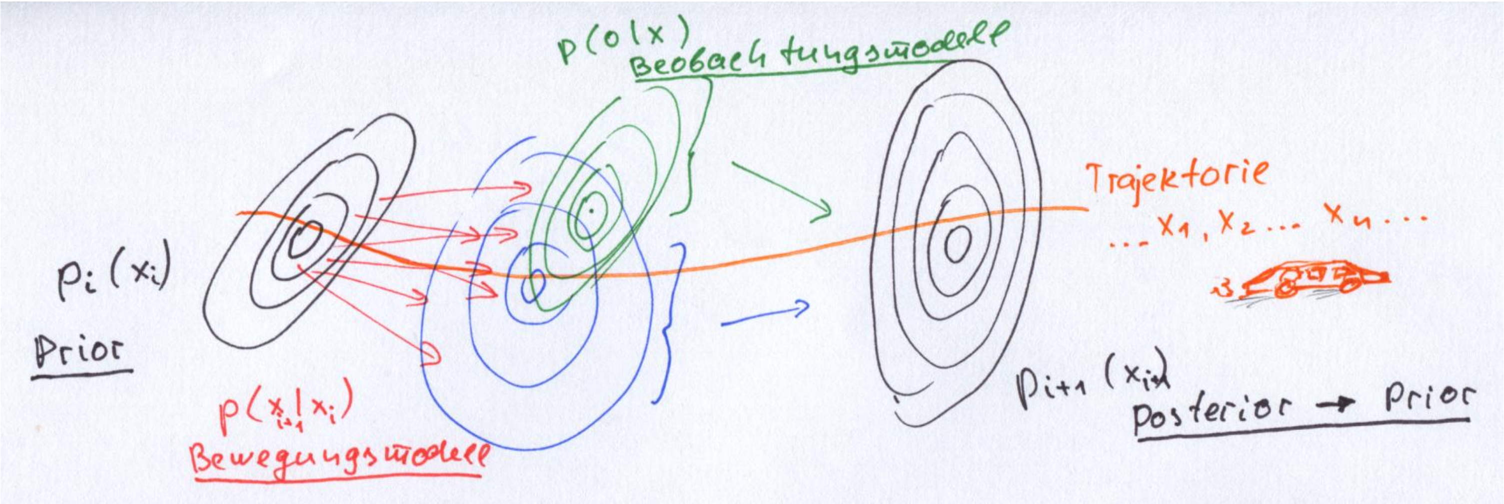
Let at the time i the probability distribution of states $p_i(x_i)$ be known. Note: not the state x_i but the probability distribution !!!

The **prior** probability distribution of states for the next time is obtained by

$$p_{i+1}(x_{i+1}) = \sum_{x_i} p_i(x_i) \cdot p(x_{i+1}|x_i)$$

This is called **Prediction**

Bayesian Filtering



Let a **measurement** be done, i.e. we have an observation o_{i+1}

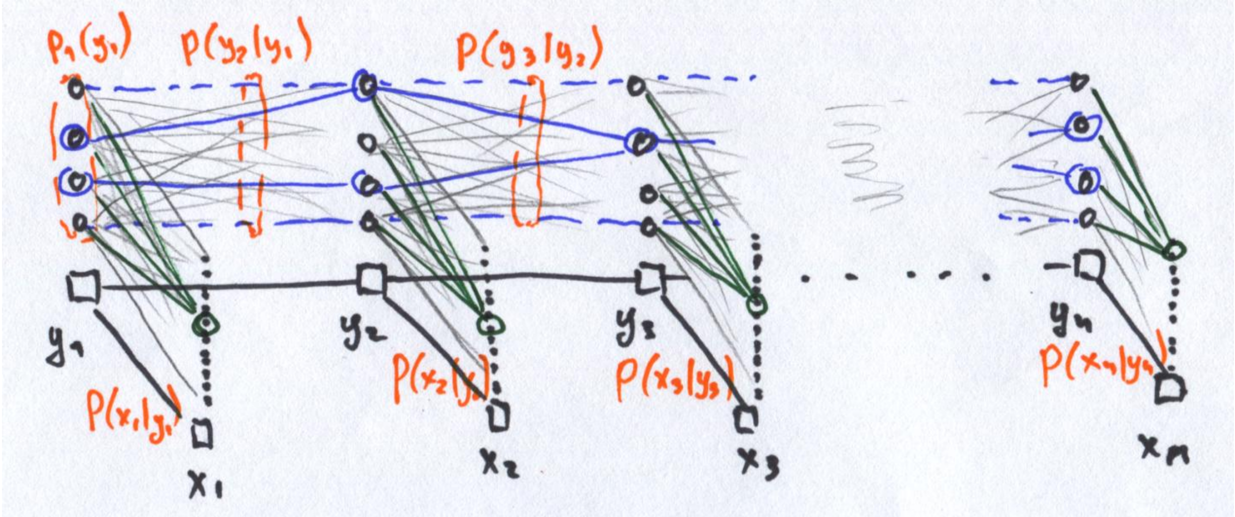
The **posterior** probability distribution of states is obtained by

$$p_{i+1}(x_{i+1}) \equiv p_{i+1}(x_{i+1}|o_{i+1}) \sim {\color{blue}p_{i+1}(x_{i+1})} \cdot {\color{green}p(o_{i+1}|x_{i+1})}$$

and serves as the **prior** for the next step.

This is called **Correction**

Markovian Chains



The set of states is discrete. The transition model is given (mainly) explicitly in form of transitional matrix.

"+" – quite general, i.e. general discrete probability distributions can be modeled

"-" – not appropriate for large state sets (mainly due to the time complexity)

Applications: speech recognition, network traffic analysis, CV …

Kalman filter [Kalman, 1960]

States and observations are vectors $x \in \mathbb{R}^n$ and $o \in \mathbb{R}^m$

Both transition model and measurement model are **linear**

$$x_{i+1} = A \cdot x_i + \epsilon, \quad o_i = B \cdot x_i + \delta$$

A and B are $n \times n$ and $n \times m$ matrices

$\epsilon \in \mathbb{R}^n$ and $\delta \in \mathbb{R}^m$ are process noise and observation **noises**

Noises are **normally distributed**

$$p(\epsilon) = \mathcal{N}(0, \Sigma_\epsilon) \sim \exp(-\epsilon^T \Sigma_\epsilon^{-1} \epsilon), \quad p(\delta) = \mathcal{N}(0, \Sigma_\delta) \sim \exp(-\delta^T \Sigma_\delta^{-1} \delta)$$

with mean values =0 and covariance matrices Σ_ϵ and Σ_δ

Kalman filter, example

The state $x = [x, y, v_x, v_y]$ describes a position (x, y) and the speed (v_x, v_y) of an object in $\mathbb{R}^2$.

For "almost uniform" motion it holds ($\triangle t$ is a time step)

$$
\begin{aligned}
x_{i+1} &= x_i + \triangle t \cdot v_{x,i} + O(\triangle t^2) \\
y_{i+1} &= y_i + \triangle t \cdot v_{y,i} + O(\triangle t^2) \\
v_{x,i+1} &= v_{x,i} + O(\triangle t) \\
v_{y,i+1} &= v_{y,i} + O(\triangle t)
\end{aligned}
$$

The state at the $i+1$-th time step is a **linear mapping** of the state at the i-th one with the "noises" $O(\triangle t)$ and $O(\triangle t^2)$

Kalman filter, example

In the matrix form:

$$\begin{bmatrix} x_{i+1} \\ y_{i+1} \\ v_{x,i+1} \\ v_{y,i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \triangle t & 0 \\ 0 & 1 & 0 & \triangle t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ v_{x,i} \\ v_{y,i} \end{bmatrix} + \epsilon$$

For measurements (only position is observed):

$$\begin{bmatrix} o_{x,i+1} \\ o_{y,i+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_i \\ y_i \\ v_{x,i} \\ v_{y,i} \end{bmatrix} + \delta$$

Extensions: $2\mathrm{D} \to 3\mathrm{D}$ ($\mathbb{R}^6$), with angles and angular speeds $\mathbb{R}^{12}$

Non-rigid objects – …

Kalman filter

Assumption: at the first time point $p(x_0) = \mathcal{N}(\bar{x}_0, \Sigma_0)$

Prediction is a **convolution** of two Gaussians:

$$p_{i+1}(x_{i+1}) = \int p_i(x_i) \cdot p(x_{i+1}|x_i)\; dx_i \sim$$

$$\sim \int \exp\Big[-(x_i - \bar{x}_i)^T \Sigma_i^{-1} (x_i - \bar{x}_i)\Big] \cdot$$

$$\cdot \exp\Big[-(x_{i+1} - Ax_i)^T \Sigma_\epsilon^{-1} (x_{i+1} - Ax_i)\Big]\, dx_i$$

→ the result is again a Gaussian $\mathcal{N}(\bar{x}'_{i+1}, \Sigma'_{i+1})$

Kalman filter

The correction is a component-vise **multiplication** of two Gaussians

$$p_{i+1}(x_{i+1}|o_{i+1}) = p_{i+1}(x_{i+1}) \cdot p(o_{i+1}|x_{i+1}) \sim$$
$$\sim \exp\Big[-(x_{i+1} - \bar{x}'_{i+1})^T \Sigma'^{-1}_{i+1} (x_{i+1} - \bar{x}'_{i+1})\Big] \cdot$$
$$\cdot \exp\Big[-(o_{i+1} - Bx_{i+1})^T \Sigma_\delta^{-1} (o_{i+1} - Bx_{i+1})\Big]$$

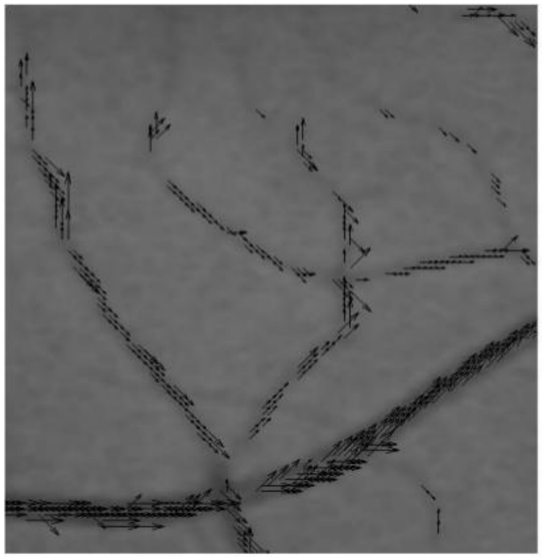
→ the result is a Gaussian again $\mathcal{N}(\bar{x}_{i+1}, \Sigma_{i+1})$

It is not necessary to propagate the probability distributions explicitly (i.e. to compute it for all x_{i+1}).

Only the parameters need to be re-computed (i.e. the mean und the covariance matrix).

Kalman filter, an application

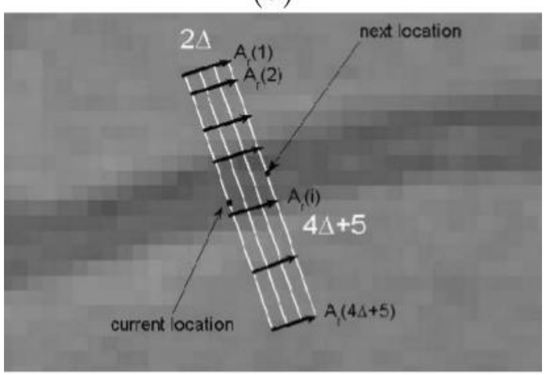
Tracking of blood vessels [Yedidya, Hartley, 2008]



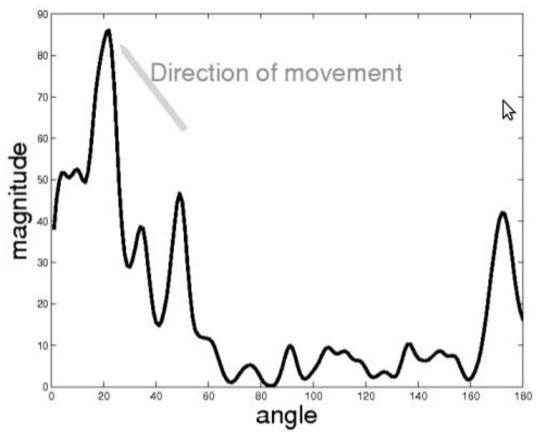
(a)

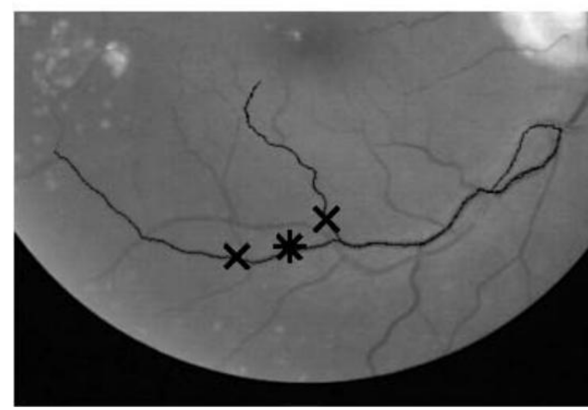
An “object” moves along the blood vessels

Its state is composed of the position, speed, thickness, gray-values observed so far etc.



(b)





Extensions

Shortcoming: Gaussian noise

A better choice – Gaussian mixtures $p(\epsilon) = \sum_k w_k \mathcal{N}(\mu_k, \Sigma_{\epsilon k})$

The problem – the number of Gaussians increases
(even for linear models)
$\sum_i \cdot \sum_j = \sum_{ij} \neq \sum_i$ the parameters can not be propagated ☹

The way out – permanently approximate the posteriors by Gaussian mixtures with a fixed number of components.

Another extension – use non-linear transition and observation models, i.e. Ax becomes $a(x)$.

How to parameterize the state distribution, how to propagate it ?

Particle filter

The Idea:
represent distributions by the density of data samples (particles).

It is possible to propagate such representation (i.e. implicitly).

Let $p_i(x_i)$ be “known”. Do many times:

1. Draw a sample x' from $p_i(x_i)$
2. Propagate it, i.e. $x'' = a(x')$
3. Compute $p(o_{i+1}|x'')$ (compare o_{i+1} with $b(x'')$)
4. Accept/reject/weight x''

Such a set of samples is distributed according to $p_{i+1}(x_{i+1})$

It is not necessary to specify the probability distributions explicitly – only sample sets are propagated (i.e. implicit non-parametric representation of the target probability distribution)

Some trackers

CONDENSATION – a particular kind of particle filtering.

Michael Isard and Andrew Blake (1998),
Condensation – conditional density propagation for visual tracking

http://www.robots.ox.ac.uk/~misard/condensation.html

TLD (Track, Learn, Detect)

Zdenek Kalal, Jiri Matas, Krystian Mikolajczyk (2009),
Online learning of robust object detectors during unstable tracking

http://kahlan.eps.surrey.ac.uk/featurespace/tld/