Reasoning and Decision Making under Uncertainty

Classification with Context and Sensor Fusion

Prof. Dr. Frank Deinzer
Technical University of Applied Sciences Würzburg-Schweinfurt
Faculty Computer Science and Business Information Systems

Summer 2024

Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

Classification with Context

- Basic idea of "context"
  - Intuitively, it is obvious that classification in the context of multiple patterns should be possible with smaller error than classifying each pattern individually
    - Speech signals
    - Handwritten text
    - Other patterns embedded in a temporal or spatial context

  - We now consider a sequence of features

$$\boldsymbol{C} = \left({}^{1}\boldsymbol{c}, {}^{2}\boldsymbol{c}, \ldots, {}^{N}\boldsymbol{c}\right)$$

and the corresponding sequence of classes

$$\boldsymbol{\Omega} = \left({}^{1}\Omega, {}^{2}\Omega, \ldots, {}^{N}\Omega\right) \quad \text{with} \quad {}^{i}\Omega \in \{\Omega_1, \Omega_2, \ldots, \Omega_k\}$$

  - We are looking for the best sequence of classes taking into account all N decisions made

Classification with Context

- Obviously the Bayes classifier can be applied to a sequence of features and classes

$$p(\boldsymbol{\Omega}|\boldsymbol{C}) = \frac{p(\boldsymbol{\Omega})p(\boldsymbol{C}|\boldsymbol{\Omega})}{p(\boldsymbol{C})}$$

- Problem here: computational complexity
  - There are k^N values of the a priori probabilities and k^N densities of nN-dimensional vectors
  - First simplification: Feature vectors are statistically independent

$$p(\boldsymbol{C}|\boldsymbol{\Omega}) = \prod_{\rho=1}^{N} p({}^{\rho}\boldsymbol{c}|{}^{\rho}\Omega = \Omega_{\kappa})$$

    Reduces to k densities of dimension n instead of k^N densities of nN-dimensional vectors
  - Second simplification: Class depends only on the direct predecessor (Markov property)

$$p(\boldsymbol{\Omega}) = p({}^1\Omega, {}^2\Omega, \ldots, {}^N\Omega) = \boxed{p({}^1\Omega)}\boxed{p({}^2\Omega|{}^1\Omega)p({}^3\Omega|{}^1\Omega{}^2\Omega)\ldots p({}^N\Omega|{}^1\Omega\ldots{}^{N-1}\Omega)}$$

$$\Rightarrow \quad p(\boldsymbol{\Omega}) = \boxed{p({}^1\Omega)}\boxed{p({}^2\Omega|{}^1\Omega)p({}^3\Omega|{}^2\Omega)\ldots p({}^N\Omega|{}^{N-1}\Omega)}$$

  - Only k^2 transition probabilities and k probabilities (instead of k^N) left

Classification with Context

- Finally, the initial equation “simplifies” to

$$p(\boldsymbol{\Omega}|\boldsymbol{C}) = \frac{1}{p(\boldsymbol{C})} p(\boldsymbol{\Omega}) p(\boldsymbol{C}|\boldsymbol{\Omega}) = \eta\, p({}^1\Omega) \left(\prod_{\rho=2}^{N} p({}^\rho\Omega|{}^{\rho-1}\Omega) \right) \left(\prod_{\rho=1}^{N} p({}^\rho\boldsymbol{c}|{}^\rho\Omega = \Omega_\kappa)) \right)$$

- Better numerical properties are obtained using the logarithmized version

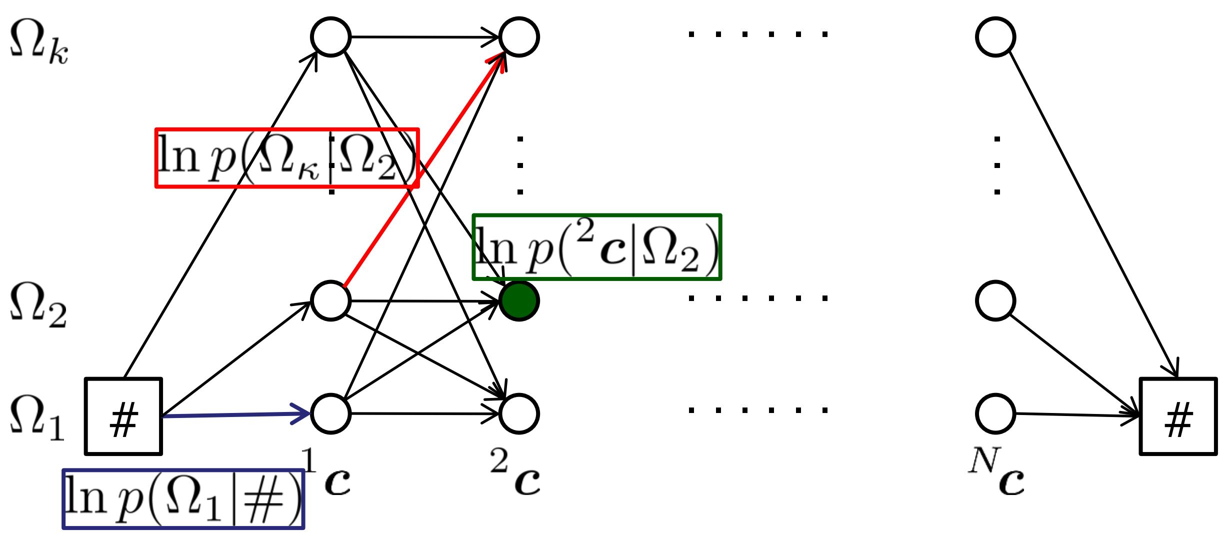
$$\ln p(\boldsymbol{\Omega}|\boldsymbol{C}) = \ln p({}^1\Omega) + \sum_{\rho=2}^{N} \ln p({}^\rho\Omega|{}^{\rho-1}\Omega) + \sum_{\rho=1}^{N} \ln p({}^\rho\boldsymbol{c}|{}^\rho\Omega = \Omega_\kappa))$$

- Do we know these densities and their parameters?

$$p({}^\rho\boldsymbol{c}|\Omega_\kappa) \qquad\qquad p(\Omega_\kappa|\Omega_\lambda)$$

Classification with Context

- Solution of the context problem: Viterbi algorithm
  - Set up network for all feature vectors and all possible classes: kN nodes
  - Each node represents observation o a feature vector under the assumption of a class
    - Assign weights to all nodes: $\ln p({}^{\rho}\boldsymbol{c}|\Omega_\kappa)$
  - Each edge between two nodes represents the assumption that one class follows another class
    - Assign weights to all edges: $\ln p(\Omega_\kappa|\Omega_\lambda)$
  - Sequence of features is delimited by a special symbol #
    - Can assign weights to start and end of sequence
  - Each sequence of classes for the sequence C corresponds to a path through the net
    - Weight of the path is sum of edge and node weights
    - Optimal path through the net is the one with maximum weight
  - Viterbi algorithm calculates the optimal path using principle of dynamic programming



Classification with Context

- Viterbi algorithm [Vit67]

FOR κ = 1 TO N

FOR λ = 1 TO k

Calculate $G_{\lambda,\rho} = \ln p({}^{\rho}\boldsymbol{c}|\Omega_\lambda) + \max\limits_{j\in\{1,\ldots,k\}} \left(G_{j,\rho-1} + \ln p(\Omega_\lambda|\Omega_j)\right)$

Special treatment for ρ = 1

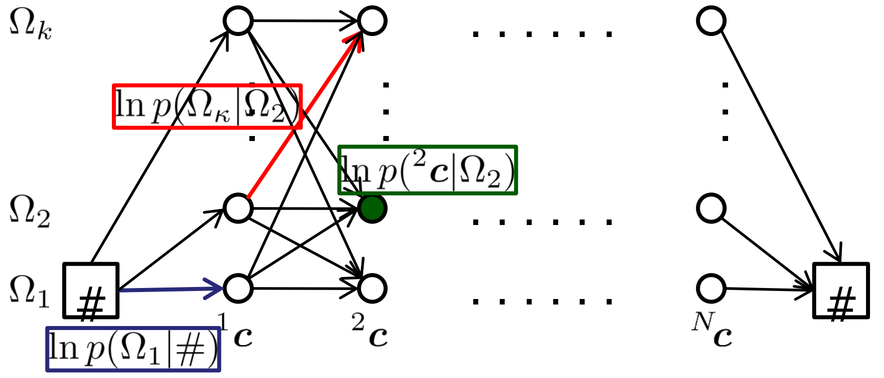
$$\max_j \left(G_{j,\rho-1} + \ln p(\Omega_\lambda|\Omega_j)\right) = \ln p(\Omega_\lambda|\#) = \ln p(\Omega_\lambda)$$

Store index of the predecessor node leading to the maximum of $G_{\lambda,\rho}$

The last column contains k weights. Calculate largest weight

$$G_{\kappa,N} = \max_\lambda G_{\lambda,N}$$

Determine path with maximum weight by tracing stored indexes from end node to start node: Sequence of these nodes is $\boldsymbol{\Omega}$



Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

Classification with Context – Hidden Markov Model

- What we came across with the Viterbi algorithm is better known under the term

  (Hidden) Markov Model (HMM)

  - Getting closer to the common view of HMMs we need to broaden the mathematical perspective
  - Change of notation from now on
    - We generalize our features to observations $\boldsymbol{c} \rightarrow \boldsymbol{o}$
    - We generalize classes to states (may be continuous) $\Omega \rightarrow \boldsymbol{q}$

    - States can only be estimated, they can never be observed, they are hidden (latent variables)
    - States differ in some important properties from classes
      - A state is usually multivariate and sometimes continuous
      - A state represents a more comprehensive description of the world

  - Notation of time-sequential states and observations

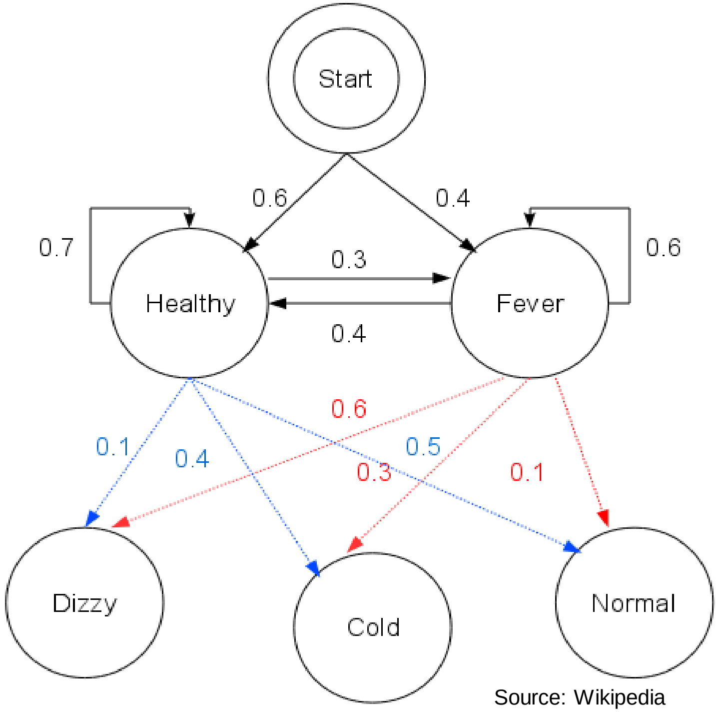
$$\langle \boldsymbol{q} \rangle_t = \boldsymbol{q}_t, \boldsymbol{q}_{t-1}, \ldots, \boldsymbol{q}_0$$
$$\langle \boldsymbol{o} \rangle_t = \boldsymbol{o}_t, \boldsymbol{o}_{t-1}, \ldots, \boldsymbol{o}_0$$

Classification with Context – Hidden Markov Model

- The context equation now changes to

$$\begin{aligned} p(\boldsymbol{q}_t, \boldsymbol{q}_{t-1}, \ldots, \boldsymbol{q}_0 | \boldsymbol{o}_t, \boldsymbol{o}_{t-1}, \ldots, \boldsymbol{o}_0) &= p(\langle \boldsymbol{q}_t \rangle \,|\, \langle \boldsymbol{o}_t \rangle) \\ &= \frac{p(\langle \boldsymbol{q} \rangle_t) p(\langle \boldsymbol{o} \rangle_t \,|\, \langle \boldsymbol{q} \rangle_t)}{p(\langle \boldsymbol{o} \rangle_t)} \\ &= \eta \prod_t p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) \prod_t p(\boldsymbol{o}_t | \boldsymbol{q}_t) \end{aligned}$$

  - What about $p(\boldsymbol{q}_0 | \boldsymbol{q}_{-1})$?



Source: Wikipedia

- In the context of HMM the "transition probabilities" are based mostly on discrete states
  - Just like our discrete classes from above
  - This leads to a transition matrix that contains all the transition probabilities
  - But: we do not want to restrict to discrete states
- In the context of HMM the observations are discrete most of the time
  - At each time any of the (finite) observations can occur
  - This leads to a observation matrix that contains the probabilities for a observation under the condition of a specific current state
  - But: we do not want to restrict to discrete observations

Classification with Context – Hidden Markov Model

- Known states: s_1 = Healthy, s_2 = Fever
- Known observations: v_1 = Dizzy, v_2 = Cold, v_3 = Normal

- Start probabilities

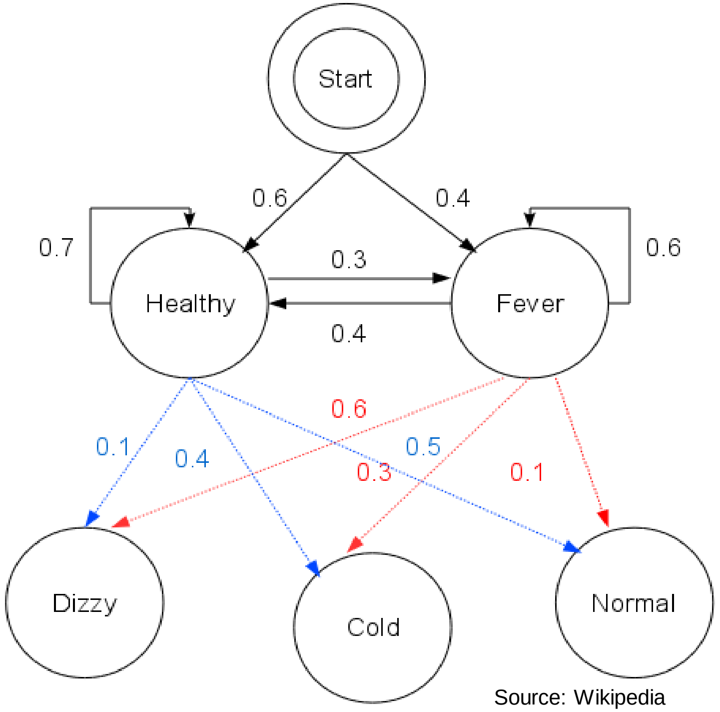
$$\boldsymbol{\pi} = \begin{pmatrix} 0.6 & 0.4 \end{pmatrix} \qquad \pi_i = p(q_0 = s_i)$$

- State transition probabilities written as matrix

$$\boldsymbol{T} = \begin{pmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{pmatrix} \qquad t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

- Output probabilities (emission probabilities) written as matrix

$$\boldsymbol{E} = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.6 & 0.3 & 0.1 \end{pmatrix} \qquad e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

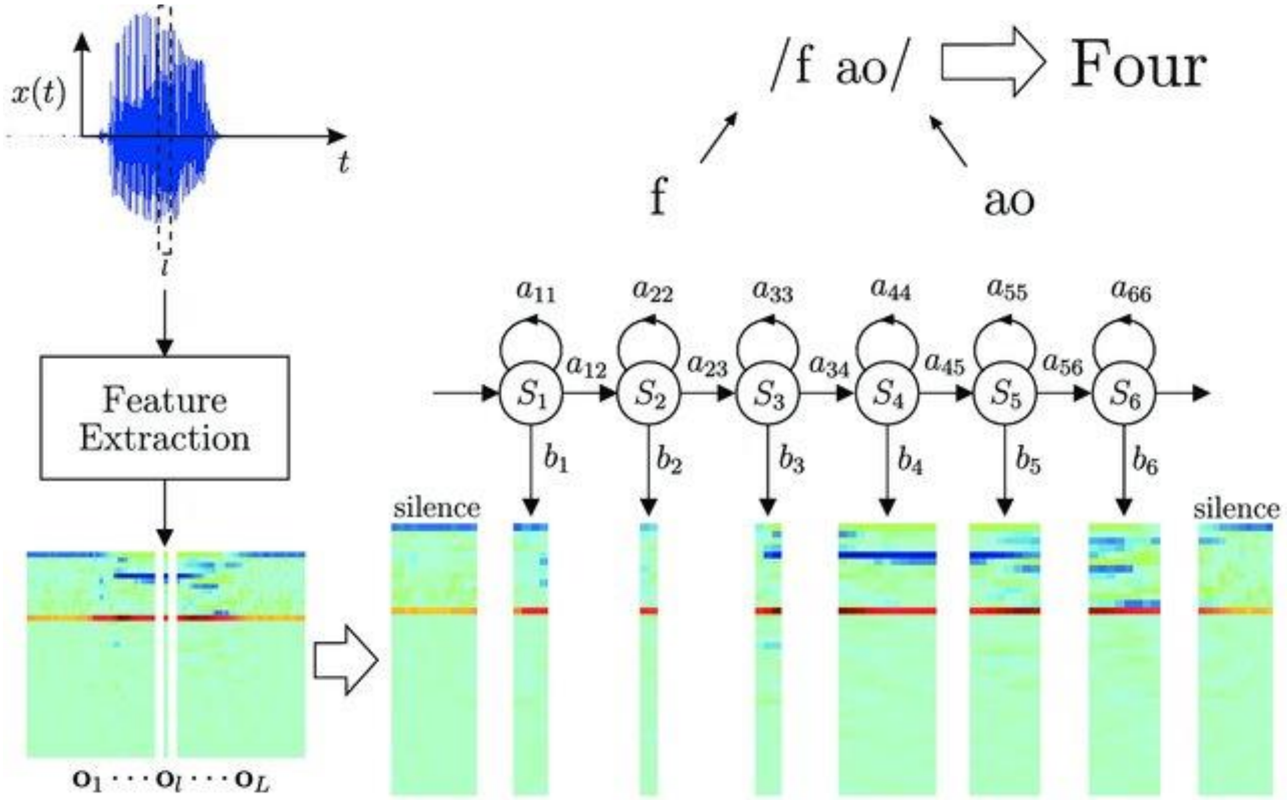


Source: Wikipedia

- Example state sequence $\langle q \rangle_4 =$ (Healthy, Healthy, Fever, Fever, Healthy)
- Example observation sequence $\langle o \rangle_4 =$ (Cold, Dizzy, Dizzy, Normal, Cold)

Classification with Context – Hidden Markov Model

- Classification with discrete observations and discrete states makes sense for many applications
  - Example: speech recognition
    - Observation in HMM model: short-term spectra of the speech signal
    - (Hidden) state in HMM model: semantic units (e.g. phonemes in speech recognition)
    - HMM model used to detect sequence of semantic units in the sequential data



Source: Ramon Fernandez Astudillo, Integration of Short-Time Fourier Domain Speech Enhancement and Observation Uncertainty Techniques for Robust Automatic Speech Recognition, DOI: 10.14279/depositonce-2483

Classification with Context – Hidden Markov Model

- Important HMM concepts
  - Filtering problem
    - Given a HMM and an observation sequence of length T
    - Looking for the probability that the instantaneous hidden state at the last time T is some specific state
    - An efficient method for solving the filtering problem is the **forward algorithm**
  - Prediction problem
    - Given a HMM and an observation sequence of length T
    - We are looking for the probability that the HMM will be in a specific hidden state at future time $T + t$ (t time-steps ahead in the future)
    - Prediction is repeated filtering without new observations
    - An efficient method for solving the prediction problem is the **forward algorithm**
  - Smoothing problem
    - Given a HMM and an observation sequence of length T
    - We are looking for the probability that the model was in a certain state at a previous time $T - t$
    - Smoothing uses future observations
    - An efficient method for solving the smoothing problem is the **forward-backward algorithm**

Classification with Context – Hidden Markov Model

- Decoding problem
  - Given an HMM and an observation sequence
  - Determine the most probable sequence of states that could have produced a given output sequence
  - An efficient method for solving the decoding problem is the **Viterbi algorithm**
- Learning problem
  - Given only an observation sequence
  - Determine the parameters of an HMM that are most likely to produce the output sequence
  - An efficient method for solving the decoding problem is the **Baum-Welch algorithm** (specialized EM-Algorithm)

Classification with Context – Hidden Markov Model

- Forward algorithm
  - Probability in the given HMM to make series of observations $\langle o \rangle_T$

$$\pi_i = p(q_0 = s_i)$$

$$t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

$$e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

1. Initialization for all known states j at first time step 0

$$\alpha_0(j) = \pi_j e_j(o_0)$$

2. Recursion for a all known states j and time steps t

$$\alpha_t(j) = e_j(o_t) \sum_i \alpha_{t-1}(i) t_{ij}$$

3. Termination

$$p(\langle o \rangle_T | \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E}) = \sum_i \alpha_T(i)$$

Classification with Context – Hidden Markov Model

- Backward algorithm
  - Probability of partial observation sequence from $t{+}1$ to end T, given state at time t

$$\pi_i = p(q_0 = s_i)$$

$$t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

$$e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

1. Initialization for all known states i at last time step T

$$\beta_T(i) = 1 \quad \forall i$$

2. Recursion for a all known states i and time steps t

$$\beta_t(i) = \sum_j t_{ij} e_j(o_{t+1}) \beta_{t+1}(j)$$

3. Termination

$$p(\langle o \rangle_T | \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E}) = \sum_j \pi_j e_j(o_0) \beta_0(j)$$

Classification with Context – Hidden Markov Model

- Using forward and backward variables for solving the smoothing problem
  - Probability that the model was in a certain state at a previous time if complete sequence of observations is available
  - Use combination of forward and backward variables
  - Matrix S aggregates these probabilities for all states and time steps

$$s_{it} = \alpha_t(i)\beta_t(i) = s_t(i)$$

Classification with Context – Hidden Markov Model

- Viterbi algorithm
  - Find the most probable sequence of states given a series of observations

$$\pi_i = p(q_0 = s_i)$$

$$t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

$$e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

1. Initialization for all known states j at first time step 0

$$v_0(j) = \pi_j e_j(o_0)$$

$$b_0(j) = 0$$

2. Recursion for a all known states j and time steps t

$$v_t(j) = \max_i v_{t-1}(i) t_{ij} e_j(o_t)$$

$$b_t(j) = \operatorname*{argmax}_i v_{t-1}(i) t_{ij} e_j(o_t)$$

3. Termination
   - Best possible score: $\max_i v_T(i)$
   - Start of backtrace: $\operatorname*{argmax}_i v_T(i)$

Classification with Context – Hidden Markov Model

- Baum-Welch Algorithm
  - Finds/calculates the parameters π, $\boldsymbol{T}$, $\boldsymbol{E}$ of the HMM

  - Initialization parameters π, $\boldsymbol{T}$, $\boldsymbol{E}$ of the HMM randomly or with prior knowledge
  - Compute forward and backward variables: $\alpha_t(j), \beta_t(i)$

  - Remember that

$$p(\langle o \rangle_T \,|\boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E}) = \sum_i \alpha_T(i) = \sum_j \pi_j e_j(o_0)\beta_0(j)$$

$$\pi_i = p(q_0 = s_i)$$

$$t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

$$e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

Classification with Context – Hidden Markov Model

- EM-Algorithm: Iterate until convergence
  - E-step
    - Probability of being in state s_i at time t, given observation sequence and model

$$\pi_i = p(q_0 = s_i)$$

$$t_{ij} = p(q_t = s_j | q_{t-1} = s_i)$$

$$e_{ij} = p(o_t = v_j | q_t = s_i) = e_i(j)$$

$$\gamma_t(i) = p(q_t = s_i | \langle o \rangle_T, \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E}) = \frac{\alpha_t(i)\beta_t(i)}{p(\langle o \rangle_T | \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E})}$$

    - Probability of being in state s_i at time t and state s_j at time $t+1$, given observation sequence and model

$$\xi_t(i,j) = p(q_t = s_i, q_{t+1} = s_j | \langle o \rangle_T, \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E}) = \frac{\alpha_t(i) t_{ij} e_j(o_{t+1}) \beta_{t+1}(j)}{p(\langle o \rangle_T | \boldsymbol{\pi}, \boldsymbol{T}, \boldsymbol{E})}$$

    - This satisfies the relationship

$$\gamma_t(i) = \sum_j \xi_t(i,j)$$

Classification with Context – Hidden Markov Model

- M-step
  - Update state transition probabilities

$$t_{ij} = \frac{\sum_{t=0}^{T-1} \xi_t(i,j)}{\sum_{t=0}^{T-1} \sum_k \xi_t(i,k)}$$

  - Update emission probabilities

$$e_j(k) = \frac{\sum_{t=0}^{T} f(o_t, v_k)\gamma_t(j)}{\sum_{t=0}^{T} \gamma_t(j)} \qquad f(a,b) = \begin{cases} 1 & \text{if a = b} \\ 0 & \text{otherwise} \end{cases}$$

- Convergence criteria: maximize likelihood

Classification with Context – Hidden Markov Model

- Baum-Welch algorithm so far estimates parameters from only **one** sequence of observations
  - Many observation sequences are used for practical implementations

  - E-step: Estimate γ and ξ for every single observation
  - M-step: Average over updates for transition and emission probabilities

- All terms in the estimation process are significantly less than 1.0
  - Products and summations converge to 0.0 very quickly (the larger T becomes, the faster the calculations will converge towards 0.0)
  - Precision of computation of forward/backward variables will exceed the floating-point accuracy of CPU

  - Solution is a proper scaling of forward and backward variables
    - See eLearning: Dawei Shen. “Some Mathematics for HMM”

Classification with Context – Sensor Fusion

- Disadvantages Forward Algorithm and Viterbi Algorithm
  - All observations must be available at the time of calculation
  - Disadvantage 1: Stepwise integration of new information not possible

$$p(\langle \boldsymbol{q}_t \rangle \mid \langle \boldsymbol{o}_t \rangle) = \eta \prod_t p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) \prod_t p(\boldsymbol{o}_t | \boldsymbol{q}_t)$$

    - Need to have all observations available
    - Calculation goes over complete sequence of observations
  - Disadvantage 2: Need to find proper family of densities that describes scenario
    - In common HMM applications this is often achieved by discrete transitions and discrete observations

Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

Recursive Density Estimation

- More general approach: recursive density estimation
  - Start again with same Bayes classifier as before
  - Limit our interest to an estimate of the state at the current time

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \, \langle \boldsymbol{o} \rangle_t)$$

    - Previous states have already been calculated

  - This can be rewritten as

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Recursive Density Estimation

- Components of the recursive formulation

$$p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

  - Estimate density for state at current time: Old information is not of interest in dynamic environment
  - Recursive part contains all information up to the last time step. Recursion until time step 0 is reached
  - State transition contains information about system dynamics
  - Observation integrates new information/observations/patterns

- Disadvantage 1 solved: Stepwise integration of new information now possible
  - Only density from last time step, system dynamics (known) and current observation is needed at any time

Recursive Density Estimation

- Disadvantage 2: There are different solutions

$$p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

  - It is always only a matter of implementing the above formula
  - We are looking for ways to describe these densities

  - Will discuss two techniques
    - Kalman Filter
    - Particle Filter

Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

Kalman Filter

- Realization of Recursive Density Estimation: Kalman Filter
  - Dynamic environments rely on interaction with the environment
    - Introduce (control) actions $\boldsymbol{a}$ in addition to observations
    - Extend recursive formula by actions

$$p(\boldsymbol{q}_t|\left\langle \boldsymbol{o}\right\rangle_t, \left\langle \boldsymbol{a}\right\rangle_{t-1}) = p(\boldsymbol{o}_t|\boldsymbol{q}_t)\int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1})p(\boldsymbol{q}_{t-1}|\left\langle \boldsymbol{o}\right\rangle_{t-1}, \left\langle \boldsymbol{a}\right\rangle_{t-2})d\boldsymbol{q}_{t-1}$$

    - State is thus dependent on
      - Observations: passive component, has no own influence on the environment, only observes the environment
      - Actions: our own commands for interaction with the environment, own decision, if no actions exist then omit them
  - Kalman filter has two major requirements
    - Linear state transition
    - Normally distributed densities

Kalman Filter

$$p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t, \langle \boldsymbol{a} \rangle_{t-1}) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}, \langle \boldsymbol{a} \rangle_{t-2}) d\boldsymbol{q}_{t-1}$$

- Kalman requirement: linear state transition
  - Realization of

$$p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1})$$

with linear state transition

$$\boldsymbol{q}_t = \boldsymbol{A}_t \boldsymbol{q}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1} + \boldsymbol{\epsilon}_t$$

Normally distributed random variable with covariance matrix $\boldsymbol{Q}$

and normally distributed uncertainty leads to density:

$$p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}) = \frac{1}{\sqrt{\det(2\pi \boldsymbol{Q}_t)}} \exp\left(- \frac{(\boldsymbol{q}_t - (\boldsymbol{A}_t \boldsymbol{q}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1}))^T \boldsymbol{Q}_t^{-1} (\boldsymbol{q}_t - (\boldsymbol{A}_t \boldsymbol{q}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1}))}{2} \right)$$

Kalman Filter

$$p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t , \langle \boldsymbol{a} \rangle_{t-1}) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1} , \langle \boldsymbol{a} \rangle_{t-2}) d\boldsymbol{q}_{t-1}$$

- Kalman requirement: linear observation model
  - Realization of

$$p(\boldsymbol{o}_t | \boldsymbol{q}_t)$$

with linear observation model

$$\boldsymbol{o}_t = \boldsymbol{C}_t \boldsymbol{q}_t + \boldsymbol{\delta}_t$$

Normally distributed
random variable with
covariance matrix $\boldsymbol{R}$

and normally distributed uncertainty leads to density:

$$p(\boldsymbol{o}_t | \boldsymbol{q}_t) = \frac{1}{\sqrt{\det(2\pi \boldsymbol{R}_t)}} \exp\left(-\frac{(\boldsymbol{o}_t - \boldsymbol{C}_t \boldsymbol{q}_t)^T \boldsymbol{R}_t^{-1} (\boldsymbol{o}_t - \boldsymbol{C}_t \boldsymbol{q}_t)}{2} \right)$$

Kalman Filter

- Famous Kalman algorithm [Kal60] is a realization of this linear state transition and linear observation model
  - Need mean vector and covariance matrix of a normal distribution representing

$$p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1})$$

  - Inputs for one iteration of the Kalman filter
    - Mean vector and covariance matrix of last iteration (last time step)
    - Current observation and action
  - State at time 0 must be normally distributed
    - Need initial mean vector and covariance matrix

Kalman Filter

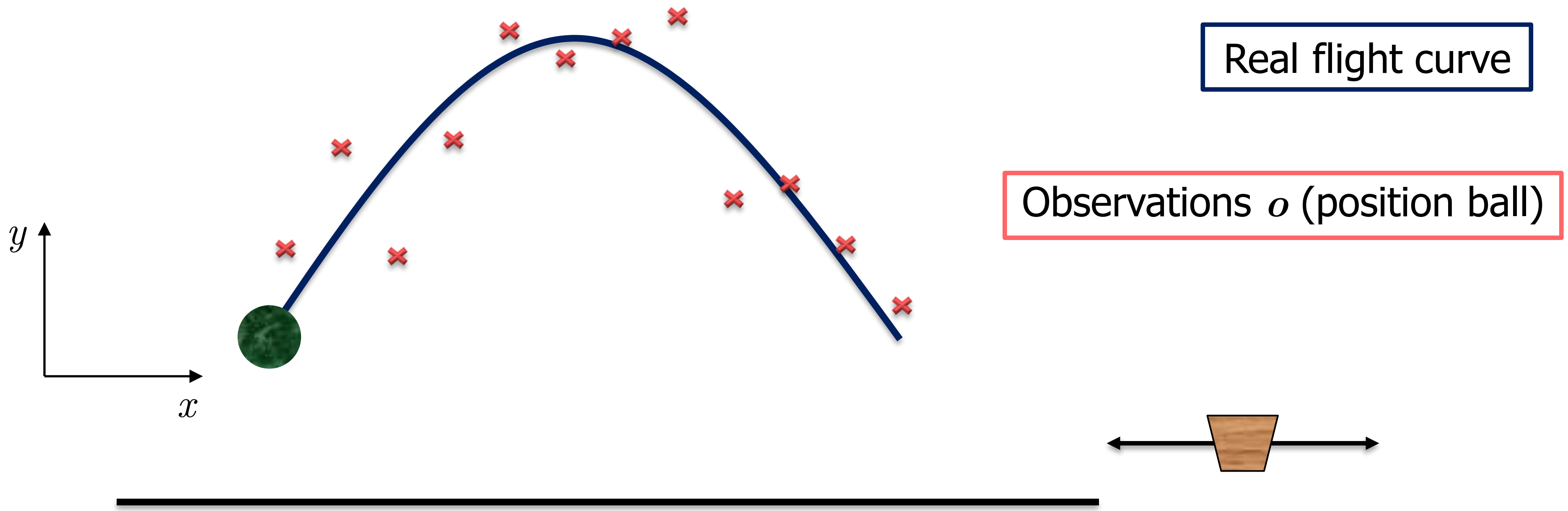
$$\texttt{Kalman}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{o}_t, \boldsymbol{a}_{t-1})$$

1. $\boldsymbol{\mu}'_t = \boldsymbol{A}_t \boldsymbol{\mu}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1}$
2. $\boldsymbol{\Sigma}'_t = \boldsymbol{A}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{Q}_t$
3. $\boldsymbol{K}_t = \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}$
4. $\boldsymbol{\mu}_t = \boldsymbol{\mu}'_t + \boldsymbol{K}_t (\boldsymbol{o}_t - \boldsymbol{C}_t \boldsymbol{\mu}'_t)$
5. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}_t) \boldsymbol{\Sigma}'_t$

- Important component: Kalman gain $\boldsymbol{K}$ [Thr05]
  - Degree/Weighting with which observation is integrated
  - Minimizes the expected, a posteriori error
  - Solves question: Greater confidence in prediction or in measurement of observation?

Kalman Filter

- Example Kalman filter: parabolic throw



  - Physical movement

$$x_{t+1} = x_t + v_x \Delta t$$

$$y_{t+1} = y_t + v_y \Delta t - 0.5 g \Delta t^2$$

Kalman Filter

- Required: transition and observation model

Transition model: $$\boldsymbol{q}_t = \boldsymbol{A}_t \boldsymbol{q}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1} + \boldsymbol{\epsilon}_t$$

$$\boldsymbol{q}_t = \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_t = \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_{t-1} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0.5\Delta t^2 & 0 \\ 0 & 0 & 0 \\ 0 & \Delta t & 0 \end{pmatrix} \begin{pmatrix} 0 \\ -g \\ 0 \end{pmatrix}_{t-1} + \boldsymbol{\epsilon}_{t-1}$$

Observation model: $$\boldsymbol{o}_t = \boldsymbol{C}_t \boldsymbol{q}_t + \boldsymbol{\delta}_t$$

$$\boldsymbol{o}_t = \begin{pmatrix} x \\ y \end{pmatrix}_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ v_x \\ v_y \end{pmatrix}_t + \boldsymbol{\delta}_t$$

Kalman Filter

$\texttt{Kalman}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{o}_t, \boldsymbol{a}_{t-1})$

1. $\boldsymbol{\mu}'_t = \boldsymbol{A}_t \boldsymbol{\mu}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1}$
2. $\boldsymbol{\Sigma}'_t = \boldsymbol{A}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{Q}_t$
3. $\boldsymbol{K}_t = \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}$
4. $\boldsymbol{\mu}_t = \boldsymbol{\mu}'_t + \boldsymbol{K}_t (\boldsymbol{o}_t - \boldsymbol{C}_t \boldsymbol{\mu}'_t)$
5. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}_t) \boldsymbol{\Sigma}'_t$

- Estimated mean vector corresponds to our state $\boldsymbol{\mu} = \boldsymbol{q}$
  - Arbitrary initial mean vector, e.g. $\boldsymbol{\mu}_0 = (0,0,0,0)^T$
  - Arbitrary initial covariance matrix, e.g. $\boldsymbol{\Sigma}_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - Action constant: acceleration due to gravity $\boldsymbol{a} = (0, -g, 0)^T$
  - Information about noise in $\boldsymbol{Q}$ and $\boldsymbol{R}$ depend on type of observation

Kalman Filter

- Limitations of the Kalman filter
  - Linear state transition and observation: unrealistic/wrong in most applications
    - Example: Robot moving on circular path cannot be described with linear motion model
  - Only unimodal, Gaussian distributed states can be modeled
  - Noise component limited

- Extended Kalman Filter (EKF)
  - Eliminates linearity requirement: nonlinear functions $\boldsymbol{g}$ and $\boldsymbol{h}$

Kalman

$$\boldsymbol{q}_t = \boldsymbol{A}_t \boldsymbol{q}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1} + \boldsymbol{\epsilon}_t$$

$$\boldsymbol{o}_t = \boldsymbol{C}_t \boldsymbol{q}_t + \boldsymbol{\delta}_t$$

$\Rightarrow$

EKF

$$\boldsymbol{q}_t = \boldsymbol{g}(\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1}) + \boldsymbol{\epsilon}_t$$

$$\boldsymbol{o}_t = \boldsymbol{h}(\boldsymbol{q}_t) + \boldsymbol{\delta}_t$$

Kalman Filter

- EKF idea: Linearization
  - Calculate a function that locally approximates a nonlinear function
  - There are many methods for linearization
    - Common idea: Take first both terms of the Taylor series expansion
    - Leads to usage of Jacobian $\boldsymbol{G}$ and $\boldsymbol{H}$ of the functions $\boldsymbol{g}$ and $\boldsymbol{h}$

Kalman$(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{o}_t, \boldsymbol{a}_{t-1})$

1. $\boldsymbol{\mu}'_t = \boldsymbol{A}_t \boldsymbol{\mu}_{t-1} + \boldsymbol{B}_t \boldsymbol{a}_{t-1}$
2. $\boldsymbol{\Sigma}'_t = \boldsymbol{A}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{A}_t^T + \boldsymbol{Q}_t$
3. $\boldsymbol{K}_t = \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T (\boldsymbol{C}_t \boldsymbol{\Sigma}'_t \boldsymbol{C}_t^T + \boldsymbol{R}_t)^{-1}$
4. $\boldsymbol{\mu}_t = \boldsymbol{\mu}'_t + \boldsymbol{K}_t (\boldsymbol{o}_t - \boldsymbol{C}_t \boldsymbol{\mu}'_t)$
5. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{C}_t) \boldsymbol{\Sigma}'_t$

$\Rightarrow$

EKF $(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}, \boldsymbol{o}_t, \boldsymbol{a}_{t-1})$

1. $\boldsymbol{\mu}'_t = \boldsymbol{g}(\boldsymbol{\mu}_{t-1}, \boldsymbol{a}_{t-1})$
2. $\boldsymbol{\Sigma}'_t = \boldsymbol{G}_t \boldsymbol{\Sigma}_{t-1} \boldsymbol{G}_t^T + \boldsymbol{Q}_t$
3. $\boldsymbol{K}_t = \boldsymbol{\Sigma}'_t \boldsymbol{H}_t^T (\boldsymbol{H}_t \boldsymbol{\Sigma}'_t \boldsymbol{H}_t^T + \boldsymbol{R}_t)^{-1}$
4. $\boldsymbol{\mu}_t = \boldsymbol{\mu}'_t + \boldsymbol{K}_t (\boldsymbol{o}_t - \boldsymbol{h}(\boldsymbol{\mu}_t)')$
5. $\boldsymbol{\Sigma}_t = (\boldsymbol{I} - \boldsymbol{K}_t \boldsymbol{H}_t) \boldsymbol{\Sigma}'_t$

  - EKF still has serious limitations
    - Only unimodal, Gaussian distributed states can be modeled
    - Noise component limited

Kalman Filter

- There are many other methods that deal with these constraints
  - Unscented Kalman filter
    - For highly nonlinear state transitions and observation
    - Samples around mean vector
  - Information filter
  - Histogram filter

  - **Particle filter**

Content and Overview

- Classification with Context
  - Viterbi Algorithm
- Hidden Markov Models
  - Concepts: Filtering, Prediction, Smoothing, Decoding, Learning
- Recursive Density Estimation
  - Kalman Filter
  - Particle Filter

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter

- Realization recursive density estimation: Particle Filter
  - Can approximate any kind of complex densities
    - Not restricted to normal distributions
    - Non-parametric
    - Suitable for nonlinear problems
    - Solution possibility for recursive density estimation

  - Need to look for solutions for two questions and find suitable components
    - Task 1: Unlimited possibility to model arbitrary densities
      - Solution: Particle Sets
      - Particle sets model densities

    - Task 2: Any system dynamics should be describable
      - Solution: Particle Filter
      - Particle filters model system dynamic

$p(\langle \boldsymbol{q}\rangle_t \,|\, \langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{q}_t|\, \langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t)\int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1})p(\boldsymbol{q}_{t-1}|\, \langle \boldsymbol{o}\rangle_{t-1})d\boldsymbol{q}_{t-1}$

Particle Filter

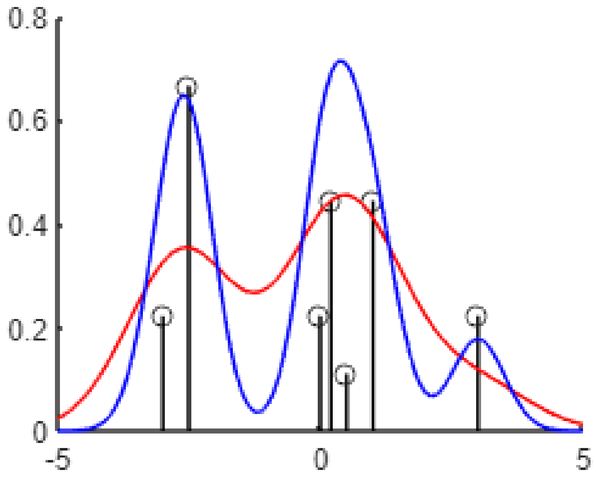
- Task 1: Approximate single density by particle set
  - Particle set consists of individual particles

$$S_t = \{s_t^1, s_t^2, \ldots, s_t^n\}$$

  - Each particle contains necessary data $\boldsymbol{s}$ for modeling the state $\boldsymbol{q}$, e.g. 2d position

$$\boldsymbol{s} = \boldsymbol{q} = (x,y)^T \quad s = \{\boldsymbol{s}, w\} = \{(x,y)^T, w\}$$

    - Each particle contains state information: application-dependent, usually same as $\boldsymbol{q}$
    - Each particle contains weighting w (think of it as probability of this particle)
  - Each particle describes an evaluation of the density at a location in the state space
  - Approximation of complete density from particle set



$$p(\boldsymbol{q}_t|\, \langle \boldsymbol{o}\rangle_t) \approx \sum_i w_t^i \delta\left(\boldsymbol{s}_t^i - \boldsymbol{q}_t\right) \qquad \sum_i w_t^i = 1$$

    - Very similar to Parzen estimate, only additional weighting for particle (like in mixture distributions)
    - If number of particles runs to infinity, window function δ can be arbitrary small and density can be approximated with arbitrary accuracy

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter

- Task 2: Particle filter that handles system dynamics over time
  - Most important particle filter: Condensation Algorithm [Isa98]

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from $S_{t\text{-}1}$ for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

  - There are many other particle filters [Dou01]

Particle Filter

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from S_{t-1} for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

- Initial particle set S_0
  - Mostly no prior knowledge about initial state available: Uniform distribution assumption for $p(\boldsymbol{q}_0)$
    - Create initial particle set by distributing n particles uniformly over state space
  - If prior knowledge available: Draw n particles from this distribution
  - Set particle weights $w_0^i = \frac{1}{n}$

$p(\langle \boldsymbol{q}\rangle_t \,|\, \langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{q}_t|\,\langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t) \int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1}|\,\langle \boldsymbol{o}\rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

- Sample n particles from $S_{t\text{-}1}$ and create new particle set S_t

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from $S_{t\text{-}1}$ for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

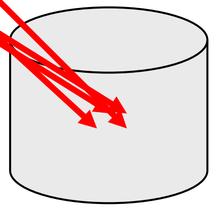
  - How to sample new particles
    - Create uniformly distributed random number [0.0 - 1.0]
    - Find particles with matching cumulative range (random number within range)
    - Copy particles into new particle set
  - This is the same as sampling from multinominal distribution

Particle	Weight	Cumulative Range
s_1	0.1	0.0 – 0.1
s_2	0.3	0.1 – 0.4
s_3	0.45	0.4 – 0.85
s_4	0.15	0.85 – 1.0

Example:

1. Created random number: 0.92
2. Created random number: 0.58
3. Created random number: 0.89
4. Created random number: 0.27

$S_t = \{s_4, s_3, s_4, s_2\}$



New particle set

$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t| \, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t) \int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1}| \, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

- Propagate particles
  - State transition model

$$p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{a}_{t-1})$$

or

$$p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1})$$

or

$$p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \boldsymbol{o}_{t-1})$$

or …

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from S_{t-1} for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

  - Goal: Adapt state data in particle due to (known or assumed) system dynamics
  - In general, state transition is not deterministic: sample from appropriate density

$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t) \int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from $S_{t\text{-}1}$ for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

- Evaluate particle

$$p(\boldsymbol{o}_t|\boldsymbol{q}_t)$$

$$S_t = \left\{s_t^1, s_t^2, \ldots, s_t^n\right\}$$
$$s_t^i = \left\{(x_t^i, y_t^i)^T, w_t^i\right\}$$

- Set weight of each particle
  - Calculate new particle weight with classifier
  - Example for simple normal distribution classifier

$$w_t^i = p\left(\boldsymbol{o}_t|(x_t^i, y_t^i)^T\right) = \mathcal{N}\left(\boldsymbol{o}_t|(x_t^i, y_t^i)^T, \boldsymbol{\Sigma}\right)$$

- Usually: Use your statistical classifier here
  - Calculate probability for each sample for the current, given observation
  - This is like evaluating density for feature vector $\boldsymbol{o}$

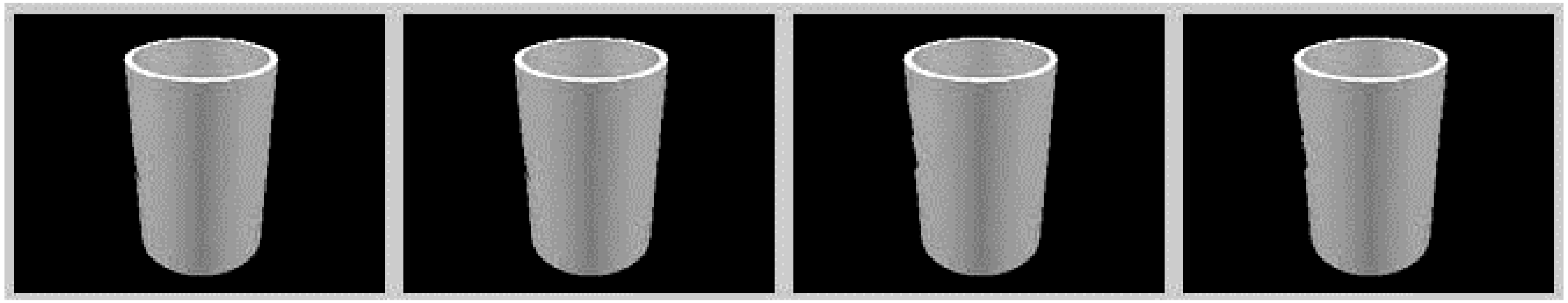
- After evaluating all particles: Normalize weighting

$$w_t^i = \frac{w_t^i}{\sum_j w_t^j}$$

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter – Classification/Localization Example

- Example particle filter: Classification
  - Recognize ambiguous objects [Dei01]



Cup A1: 50%

Cup A2: 50%

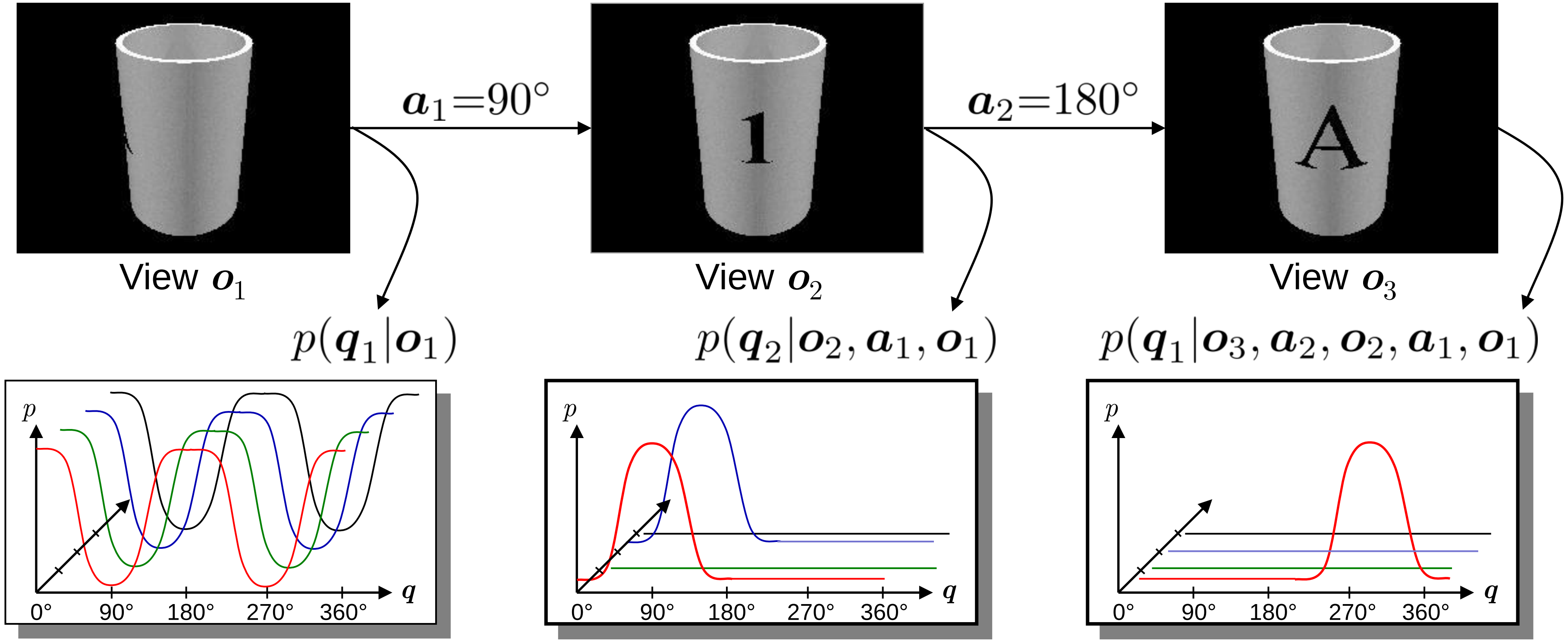
Cup A1: 50%

Cup B1: 50%

$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter – Classification/Localization Example

- Example particle filter: Classification
  - Fusion of information from 3 images

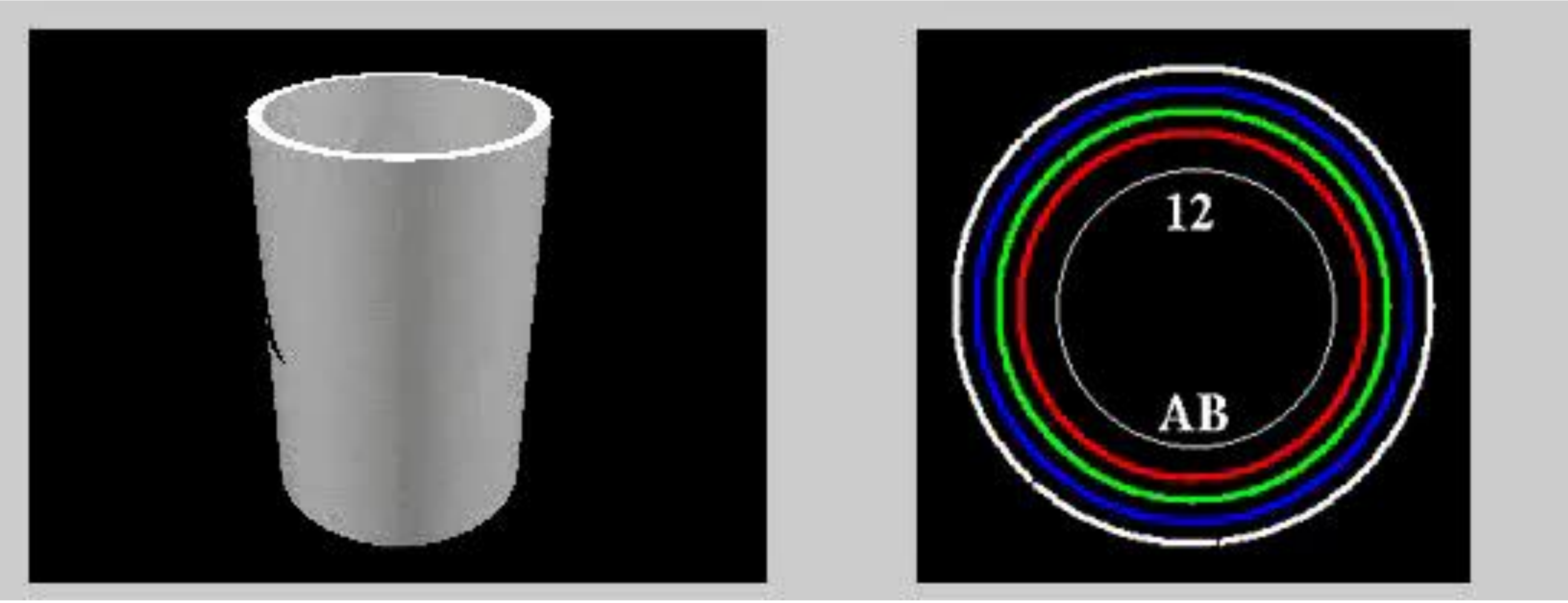


Cup A1, Cup A2, Cup B1, Cup B2

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

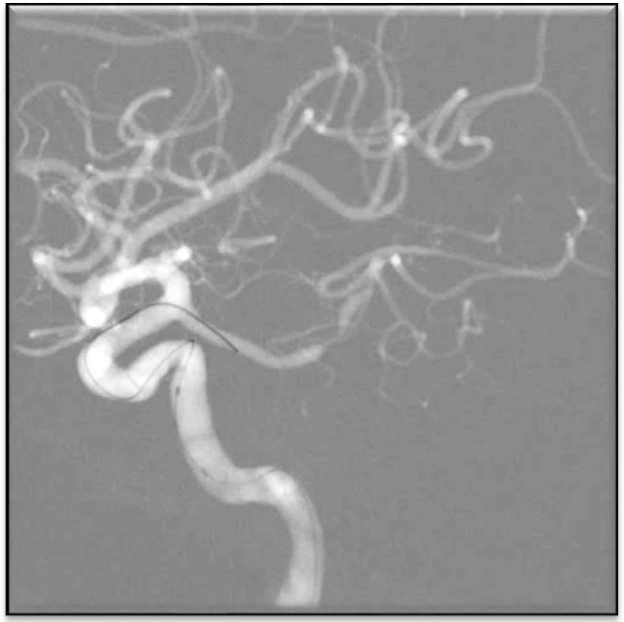
Particle Filter – Classification/Localization Example

- Example particle filter: Classification
  - Fusion of a sequence of many images [Dei01]



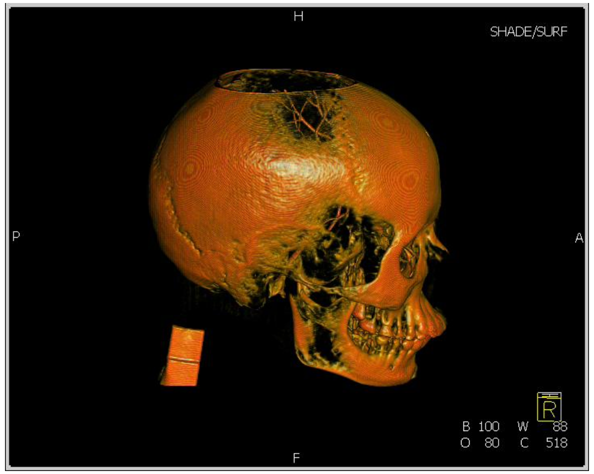
Particle Filter – Medical Example (Angiography)

- Example particle filter: Medical Application in Angiography
  - 2-D X-ray imaging basis of many applications in interventional radiology
  - Specialized **C-arm** systems used for treatment of strokes, heart attacks, tumors



Particle Filter – Medical Example (Angiography)

- Example particle filter: Medical Application in Angiography
  - 3-D interventional imaging essential technique nowadays
  - Information from 3D volume often not visible in 2D X-ray images
  - No real-time 3D imaging



3D volume

Particle Filter – Medical Example (Angiography)

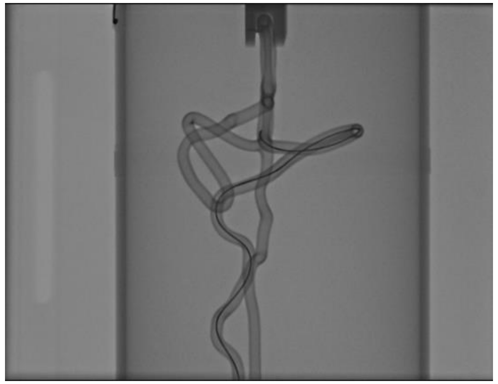
- Example particle filter: Medical Application in Angiography
  - Application: Overlay of rendered 3D volume
  - Preprocessing of 3D volume (e.g. remove
  - unwanted anatomic details) possible
  - Overlaid volume will follow any C-arm system parameter changes and movements

  - Here: Treatment of an aneurysm

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t) \int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1}| \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter – Medical Example (Angiography)

- Example particle filter: Medical Application in Angiography
  - Track and reconstruct medical guidewire in patient vessel system [Brü09]



Sequence of X-ray images

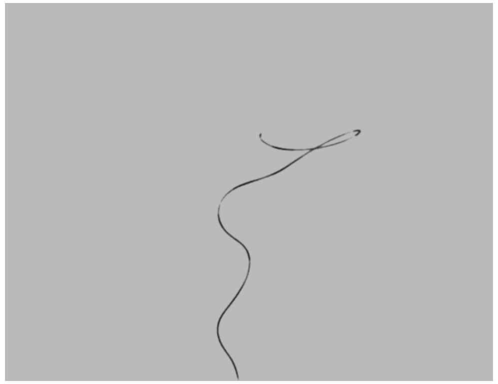
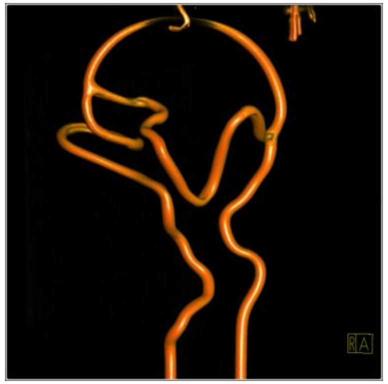


Image processing using mask images: only guidewire visible



3D reconstruction from CT-scanner

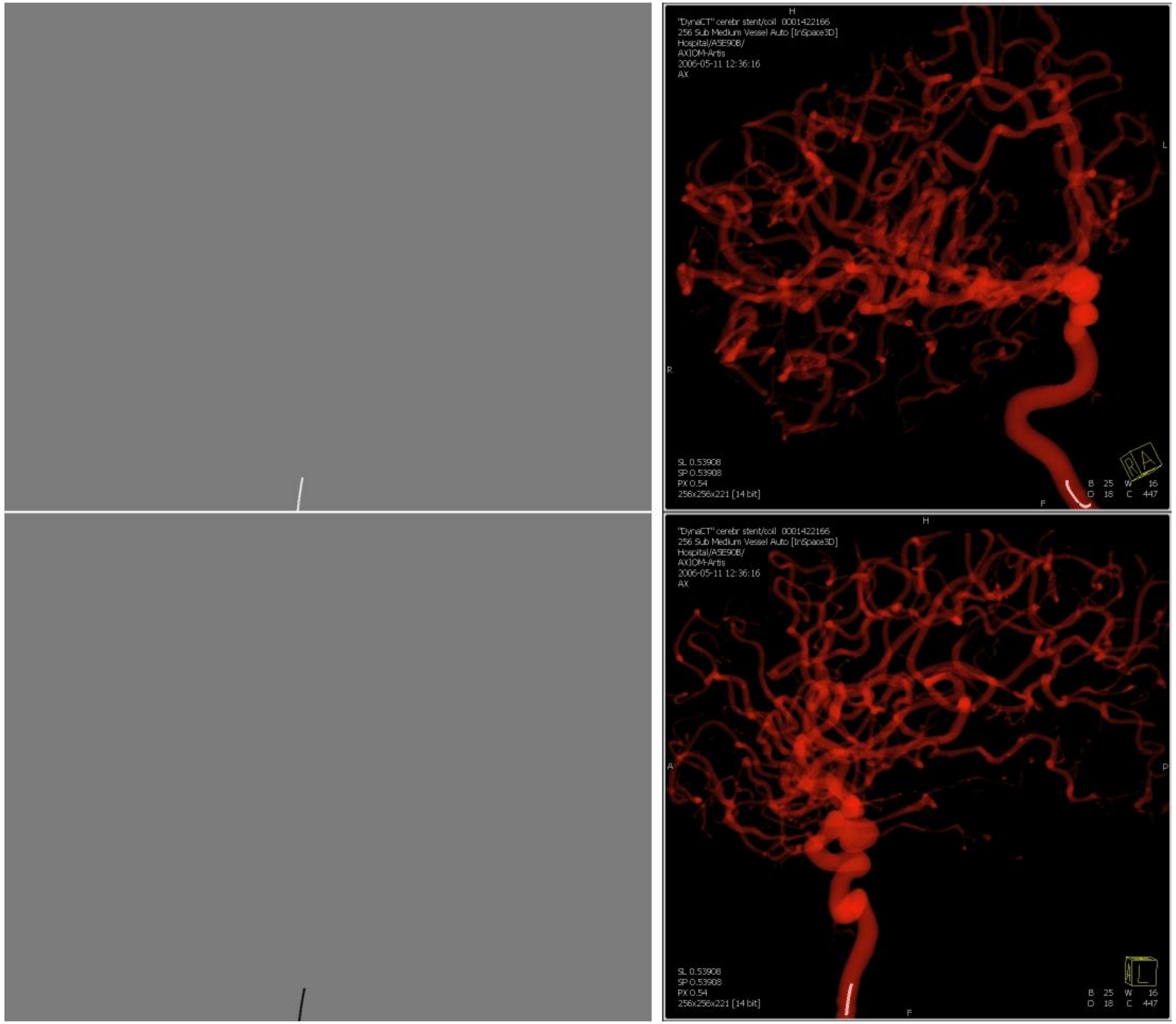
  - Difficult problem in case of complex vessel systems

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter – Medical Example (Angiography)

$$p(\langle \boldsymbol{q}\rangle_t \mid \langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{q}_t \mid \langle \boldsymbol{o}\rangle_t) = p(\boldsymbol{o}_t|\boldsymbol{q}_t) \int p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1})p(\boldsymbol{q}_{t-1}| \langle \boldsymbol{o}\rangle_{t-1})d\boldsymbol{q}_{t-1}$$

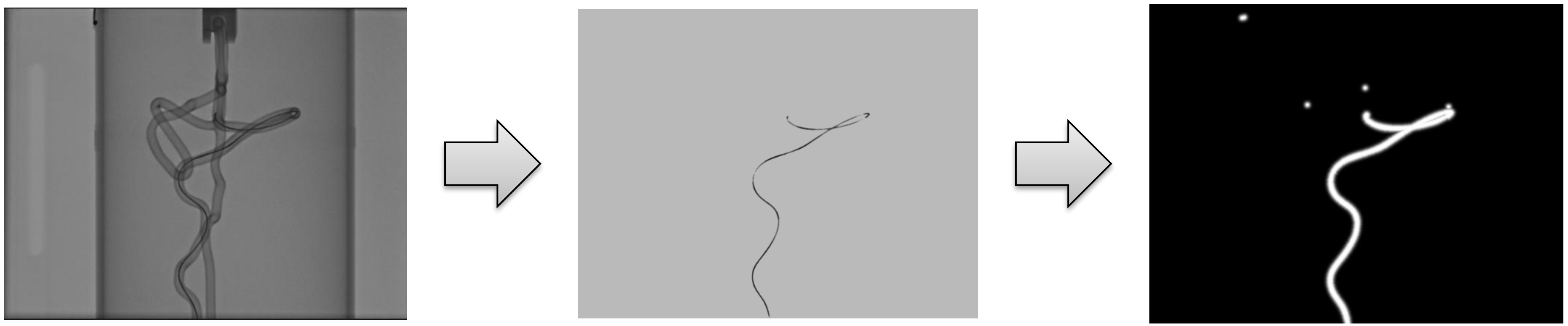
Particle Filter – Medical Example (Angiography)



Particle Filter – Medical Example (Angiography)

- Example particle filter: Catheter Tracking
  - Use 3D coordinates as state $\boldsymbol{q} = (x, y, z)^T$

  - Observation $p(\boldsymbol{o}_t | \boldsymbol{q}_t)$
  - Use image difference (live image – mask image) $\boldsymbol{o}_t$
  - Apply some kind of blurring ➔ model inaccuracies
  - Density $p(\boldsymbol{o}_t | \boldsymbol{q}_t)$ of observations depend on known projective geometry (just like for catheter tip)

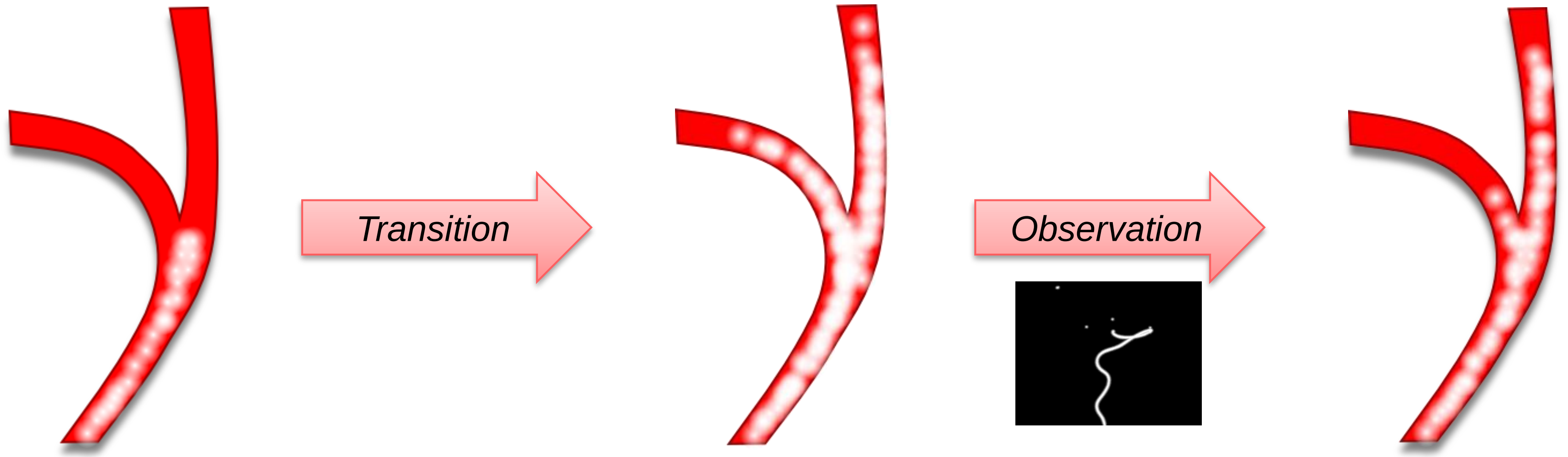


Particle Filter – Medical Example (Angiography)

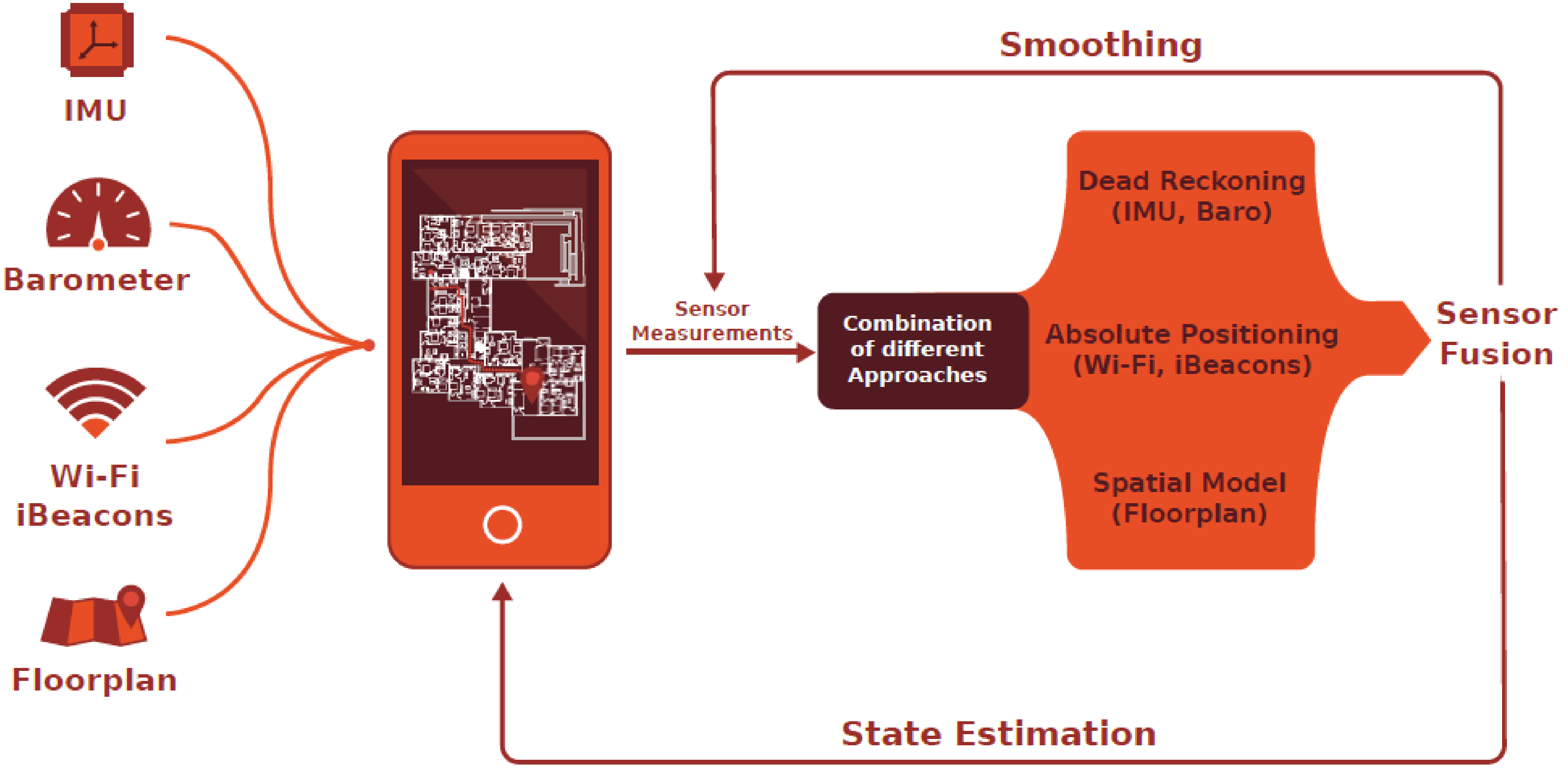
- Example particle filter: Catheter Tracking
  - Transition $p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \langle \boldsymbol{o} \rangle_{t-1})$
  - Along the way of the catheter probabilities shall be evenly distributed
  - Traditional transition without prior observations will avoid this
  - Integrating observations allows a formulation in the form

$$p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1}, \langle \boldsymbol{o} \rangle_{t-1}) \propto \frac{p(\boldsymbol{q}_t|\boldsymbol{q}_{t-1})}{\int\limits_{\boldsymbol{\epsilon} \in A} \int p(\boldsymbol{q}_t + \boldsymbol{\epsilon}|\boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1}|\boldsymbol{o}_{t-1}) d\boldsymbol{q}_{t-1} d\boldsymbol{\epsilon}}$$

    - A describes a local area around $\boldsymbol{q}_t$
    - Denominator acts as smoothing of transition probability

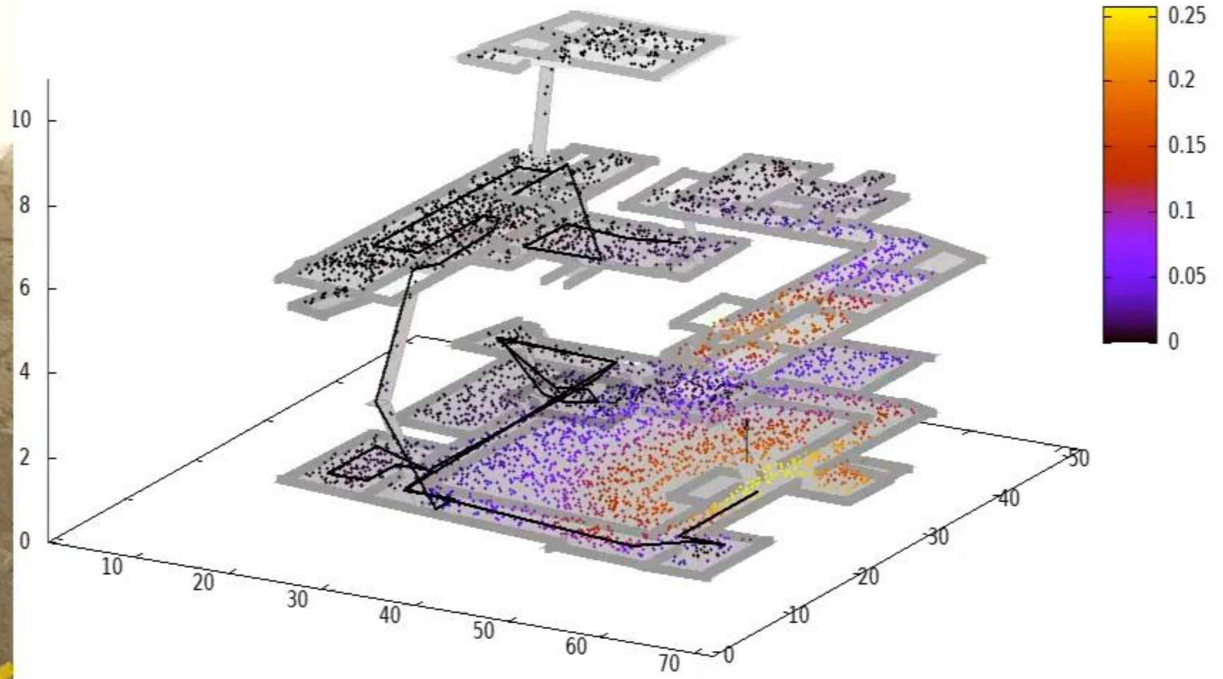


Particle Filter – Indoor Localization



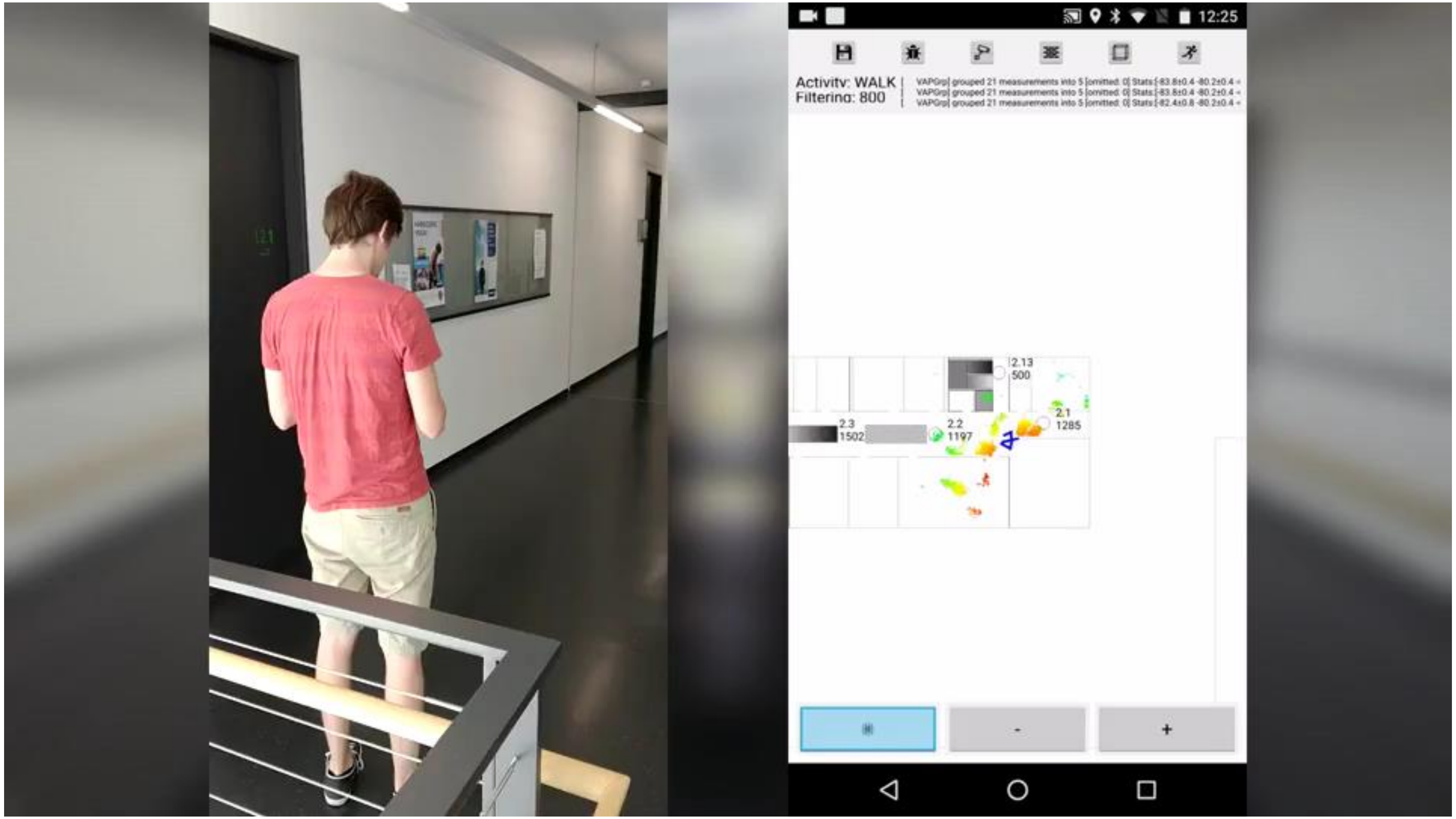
Particle Filter – Indoor Localization

- Example: Indoor Localization [Fet18], More Videos [SimpleLoc YouTube]



Particle Filter – Indoor Localization

- Example: Indoor Localization @ SHL, More Videos SimpleLoc YouTube

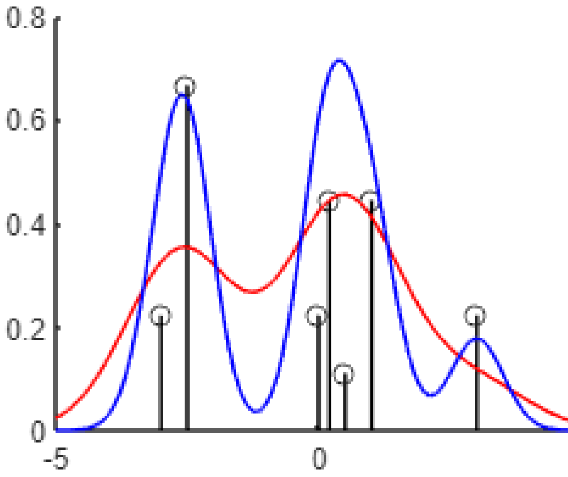


$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

- Particle Filtering Revisited
  - Approximate posterior by particle set (= set of weighted samples)

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) \approx \sum_i w_t^i \delta \left(\boldsymbol{s}_t^i - \boldsymbol{q}_t \right)$$



  - Sample-based approximation has an obvious difficulty
    - Posterior is unknown
    - Hence sampling from posterior is impossible
  - Solution: Importance Sampling
    - If we cannot sample from the desired posterior, then we sample from other density
    - We need to correct the error that we are making as a result of this
    - This other density is called *proposal density* or *importance density* and denoted by q

$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

- When sampling from a proposal density q we need to compute the weights without knowing the posterior
  - Can be done this way [Thr05]

$$w_t^i \propto \frac{p(\langle \boldsymbol{s}^i \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t)}{q(\langle \boldsymbol{s}^i \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t)}$$

  - Luckily, we are not interested in the full sequence of states over time, but only in the current state
  - Weighting can thus be reformulated similarly to the derivation of the recursive density estimate

$$w_t^i \propto w_{t-1}^i \frac{p(\boldsymbol{o}_t | \boldsymbol{s}_t^i) p(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i)}{q(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i, \boldsymbol{o}_t)}$$

  - These are the weights for our sampled density

$$p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) \approx \sum_i w_t^i \delta \left(\boldsymbol{s}_t^i - \boldsymbol{q}_t \right)$$

  - This is the **Sequential Importance Sampling** (SIS) algorithm. These weights are optimal

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t |\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} |\, \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter

- At first glance, it looks as if we now have two unknown densities instead of one

$$w_t^i \propto w_{t-1}^i \frac{p(\boldsymbol{o}_t | \boldsymbol{s}_t^i) p(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i)}{q(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i, \boldsymbol{o}_t)}$$

  - It is up to you to use a suitable proposal density
    - Known from importance sampling: you can choose any proposal density you want (if they fulfill some basic properties)
    - Most common choice (Condensation algorithm does it that way) is

$$q(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i, \boldsymbol{o}_t) = p(\boldsymbol{s}_t^i | \boldsymbol{s}_{t-1}^i)$$

    - The transition model shall be the proposal density, because with that choice the weight update simplifies to

$$w_t^i \propto w_{t-1}^i p(\boldsymbol{o}_t | \boldsymbol{s}_t^i)$$

  - Condensation algorithm
    - Sample from proposal density
    - Update weights

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from $S_{t\text{-}1}$ for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$

Particle Filter

Condensation Algorithm

1. Create particle set S_0 for initial density $p(\boldsymbol{q}_0)$
2. Sample particles from $S_{t\text{-}1}$ for new particle set S_t
3. Propagate each particle from S_t with state transition model
4. Evaluate each particle from S_t with observation model
5. Repeat from (2.)

- The Condensation algorithm incorporates a resampling step to solve the *degeneracy problem*
- Performing without step (2.)
  - After few iterations weight of **one** particle will be close to 1.0
  - All other particle weights will be almost zero
  - This is the degeneracy problem
- Resampling of the particle set leads to new, equal weights
  - Degeneracy problem solved
- This is the **Sequential Importance Resampling** (SIR) filter
  - Transition prior used as proposal function
  - SIR filters are commonly known as **Bootstrap Filter** and **Condensation Algorithm**

$$p(\langle \boldsymbol{q} \rangle_t \,|\, \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{q}_t | \langle \boldsymbol{o} \rangle_t) = p(\boldsymbol{o}_t | \boldsymbol{q}_t) \int p(\boldsymbol{q}_t | \boldsymbol{q}_{t-1}) p(\boldsymbol{q}_{t-1} | \langle \boldsymbol{o} \rangle_{t-1}) d\boldsymbol{q}_{t-1}$$

Particle Filter

- There are many more challenges in particle filtering
  - Sample Impoverishment: Transition step contains a deterministic and a stochastic part. Particles with same state will diversify during transition, but if noise variance is low, samples will not diversify enough. After some iterations particles will collapse into a single point in state space
  - Particle Filter Divergence: Poor tuning of filter, incorrect modeling assumptions, inconsistent measurements. Filter will diverge to "somewhere"
  - Proposal Density: Good proposal function is important. Transition as proposal is not always a good choice
  - Realtime Performance: There are many expensive components in a particle filter
    - Evaluation: Every particle must be evaluated
    - Transition: Every particle must be processed by transition model
    - Resampling: Particle set must be resampled. There are many different resampling strategies
    - Number of particles: Curse of dimensionality. Many parameters in state will lead to large particle sets
  - Good reading: [Elf21]

Bibliography

[Dem77] Dempster, A.P., Laird. N.M., Rubin, D.B.: *Maximum-Likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, 1977

[Dou01] Arnaud Doucet, Nando DeFreitas, Neil Gordon: Sequential Monte Carlo Methods in Practice. Springer 2001

[Elf21] Jos Elfring, Elena Torta, René van de Molengraft: *Particle Filters: A Hands-On Tutorial*. Sensors 2021, 21, 438. https://doi.org/10.3390/s21020438

[Isa98] Michael Isard , Andrew Blake. *CONDENSATION - conditional density propagation for visual tracking*. International Journal of Computer Vision, 29:5-28, 1998

[Kal60] Kalman, R. E: *A New Approach to Linear Filtering and Prediction Problems*, Transaction of the ASME, Journal of Basic Engineering, 35-45, 1960

[Nie83] H. Niemann: Klassifikation von Mustern. Springer, Berlin 1983. Online available http://www5.informatik.uni-erlangen.de/fileadmin/Persons/NiemannHeinrich/klassifikation-von-mustern/m00-www.pdf

[Thr05] Sebastian Thrun, Wolfram Burgard, Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), Mit Press, 2005

[Vit67] A. Viterbi: Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory. Vol. 13, Nr. 2, 1967, pp 260–269

Bibliography – Own Publications on Sensor Fusion

[Fet23b] Toni Fetzer, Markus Bullmann, Markus Ebner, Steffen Kastner, Frank Deinzer und Marcin Grzegorzek. „Interacting Multiple Model Particle Filter for Indoor Positioning Applications“. In: Proceedings of the 2023 International Technical Meeting of The Institute of Navigation. Long Beach, California, USA, 2023.

[Fet23] Toni Fetzer, Frank Ebner, Frank Deinzer und Marcin Grzegorzek. „Using Barometer for Floor Assignation within Statistical Indoor Localization“. In: Sensors 23.1 (2023). ISSN: 1424-8220.

[Bul22] Markus Bullmann, Toni Fetzer, Markus Ebner, Steffen Kastner, Frank Deinzer und Marcin Grzegorzek. „Data Driven Sensor Model for Wi-Fi Fine Timing Measurement“. In: International Conference on Indoor Positioning and Indoor Navigation (IPIN 2022). Beijing, China: IEEE, 2022.

[Ebn22] Markus Ebner, Toni Fetzer, Markus Bullmann, Steffen Kastner, Frank Deinzer und Marcin Grzegorzek. „PIPF: Proposal-Interpolating Particle Filter“. In: International Conference on Indoor Positioning and Indoor Navigation (IPIN 2022). Beijing, China: IEEE, 2022.

[Kas22] Steffen Kastner, Markus Ebner, Markus Bullmann, Toni Fetzer, Frank Deinzer und Marcin Grzegorzek. „Magnetic Signature Sensor Model for Accurate Short-Distance Localization“. In: 2022 IEEE Sensors. Dallas, TX, USA, 2022, S. 1–4. DOI: 10.1109/SENSORS52175.2022.9967176.

[Fet22] Toni Fetzer, Julian Maier, Markus Ebner, Markus Bullmann und Frank Deinzer. „Digitales Spaghetti-Diagramm zur Laufweganalyse“. In: wt Werkstattstechnik online 112.10/2022 (Okt. 2022), S. 727–731.

[Ebn20] Markus Ebner, Toni Fetzer, Markus Bullmann, Frank Deinzer, Marcin Grzegorzek. "Recognition of Typical Locomotion Activities Based on the Sensor Data of a Smartphone in Pocket or Hand". Sensors 2020, 20(22), 6559.

[Bull20] Markus Bullmann, Toni Fetzer, Frank Ebner, Markus Ebner, Frank Deinzer, Marcin Grzegorzek. "Comparison of 2.4 GHz WiFi FTM- and RSSI-Based Indoor Positioning Methods in Realistic Scenarios". Sensors 2020, 20(16), 4515. doi.org/10.3390/s20164515

[Bull18] Markus Bullmann, Toni Fetzer, Frank Ebner, Frank Deinzer und Marcin Grzegorzek. „Fast Kernel Density Estimation Using Gaussian Filter Approximation“. In: 21st International Conference on Information Fusion, FUSION 2018, Cambridge, UK, July 10-13, 2018. IEEE, 2018, S. 1233–1240. ISBN: 978-0-9964527-6-2. DOI: 10.23919/ICIF.2018.8455686. URL: doi. org/10.23919/ICIF.2018.8455686.

[Fet18] Toni Fetzer, Frank Ebner, Markus Bullmann, Frank Deinzer und Marcin Grzegorzek. „Smartphone-Based Indoor Localization within a 13th Century Historic Building“. In: Sensors 18.12 (2018). ISSN: 1424-8220. DOI: 10.3390/s18124095. URL: www.mdpi.com/1424-8220/18/12/4095. [Bul17]

Bibliography – Own Publications on Sensor Fusion

[Bul17] Markus Bullmann, Toni Fetzer, Frank Ebner, Frank Deinzer, Marcin Grzegorzek. Fast Kernel Density Estimation Using Gaussian Filter Approximation. 2018 International Conference on Information Fusion (FUSION). Oxford, UK. July 2017

[Ebn17] Frank Ebner, Toni Fetzer, Frank Deinzer, Marcin Grzegorzek. "On Wi-Fi Model Optimizations for Smartphone-Based Indoor Localization". ISPRS International Journal of Geo-Information, 2017, 6(8), 233, MDPI

[Tor17] Joaquín Torres-Sospedra, Antonio R. Jiménez, Stefan Knauth, Adriano Moreira, Yair Beer, Toni Fetzer, Viet-Cuong Ta, Raul Montoliu, Fernando Seco, Germán M. Mendoza-Silva, Oscar Belmonte, Athanasios Koukofikis, Maria João Nicolau, António Costa, Filipe Meneses, Frank Ebner, Frank Deinzer, Dominique Vaufreydaz, Trung-Kien Dao, Eric Castelli. The Smartphone-Based Offline Indoor Location Competition at IPIN 2016: Analysis and Future Work. Sensors, Vol. 17, No. 3, 2017

[Fet17] T. Fetzer, F. Ebner, L. Köping, M. Grzegorzek und F. Deinzer. Recovering From Sample Impoverishment in Context of Indoor Localisation. International Conference on Indoor Positioning and Indoor Navigation (IPIN 2017). Sapporo, Japan: IEEE, Sep. 2017, S. 1–8.

[Ebn16] F. Ebner, T. Fetzer, F. Deinzer, M. Grzegorzek On Prior Navigation Knowledge in Multi Sensor Indoor Localisation. International Conference on Information Fusion (FUSION), 2016, International Conference on. IEEE, July 2016

[Fet16] T. Fetzer, F. Ebner, L. Köping, M. Grzegorzek, F. Deinzer. On Monte Carlo Smoothing in Multi Sensor Indoor Localisation. International Conference on Indoor Positioning and Indoor Navigation (IPIN), 2016, International Conference on. IEEE, October 2016

[Ebn15] F. Ebner, T. Fetzer, L. Köping, M. Grzegorzek, F. Deinzer. Multi Sensor 3D Indoor Localisation. Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on. IEEE, Oktober 2015, S. 1-11

[Köp15] L. Köping, M. Grzegorzek, F. Deinzer, S. Bobek, M. Slazynski und G.J. Nalepa. Improving indoor localization by user feedback. Information Fusion (Fusion), 2015 18th International Conference on. IEEE, Juli 2015, S. 1053–1060

[Köp14] Lukas Köping, Frank Deinzer. „Realidades Virtuales/Aumentadas para el Desarrollo Social: Experiencias Entre Europa y Latinoamerica“. 2014. Kap. Real-Time Indoor Localization for Augmented and Virtual Reality Applications.

[Ser14] Francisco Serón, Carlos Vaz de Carvalho, Melanie Saul, Frank Deinzer, Roberto Guerrero, Juan Carlos Parra, Cristiano Costa und Sandro Rigo. „Realidades Virtuales/Aumentadas para el Desarrollo Social: Experiencias Entre Europa y Latinoamerica“. 2014. Kap. Virtual Reality, Augmented Reality and Ubiquitous Computing: Concepts, Technology and Practice.

[Fet14] Toni Fetzer, Frank Deinzer, Lukas Köping und Marcin Grzegorzek. Statistical indoor localization using fusion of depth-images and step detection. In: Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on. IEEE, Okt. 2014, S. 407–415

[Ebn14] F. Ebner, F. Deinzer, L. Köping und M. Grzegorzek. Robust self-localization using Wi-Fi, step/turn-detection and recursive density estimation. Indoor Positioning and Indoor Navigation (IPIN), 2014 International Conference on. IEEE, Okt. 2014, S. 627–635

[Köp14] L. Köping, F. Ebner, M. Grzegorzek, F. Deinzer. Indoor Localization Using Step and Turn Detection Together with Floor Map Information. In: FHWS Science Journal 1 (2014), S. 40–49

[Köp12b] L. Köping, T. Mühsam, C. Ofenberg, B. Czech, M. Bernard, J. Schmer, and F. Deinzer. Indoor Navigation Using Particle Filter and Sensor Fusion, Annual of Navigation, vol. 19, no. 2, pp. 31–40, December 2012.

[Köp12a] L. Köping, T. Mühsam, C. Ofenberg, B. Czech, M. Bernard, J. Schmer, and F. Deinzer, Indoor Navigation Using Particle Filter and Sensor Fusion, Proceedings of the European Navigation Conference 2012, April 2012.

[Brü09] Brückner, Deinzer, Denzler. Temporal Estimation of the 3d Guide-Wire Position Using 2d X-ray Images. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2009, LNCS 5761, 386–393

[Dei01] F. Deinzer, J. Denzler, H. Niemann. On Fusion of Multiple Views for Active Object Recognition. In B. Radig, S. Florczyk, (Herausgeber), Pattern Recognition - 23rd DAGM Symposium. Lecture Notes in Computer Science 2191, Seite 239-245, München, September 2001, Springer Berlin.

Bibliography – Own Publications on Sensor Fusion

Awards

- Winner of the "Indoor Localization Competition for Smartphone-Based Solutions" at the International Conference on Indoor Positioning and Indoor Navigation 2016 in Madrid, Spain
- Computer Science Award 2016 of the Fachbereichtag Informatik. Award for the master thesis Toni Fetzer "Smoothing and Prediction in Statistical Indoor Localization"
- Best-Paper Award of the International Conference on Indoor Positioning and Indoor Navigation 2015 (Frank Ebner, Toni Fetzer, Lukas Koeping, Marcin Grzegorzek, Frank Deinzer): Award for the paper "Multi Sensor 3D Indoor Localization".
- Innovationspreis 2008 der Gesellschaft für Informatik e.V. (Frank Deinzer, Esther Platzer): Auszeichnung für die Arbeit „Erstellung von 4-D-Angiogrammen in der interventionellen Radiologie“
- Scientific Award 2007 der BMW Group (1. Platz): Auszeichnung der Diplomarbeit Esther Platzer: „Visualierung von Blutfluss in 3-D-Datensätzen aus 2-D-Angiogrammen“
- DAGM-Preis 2001 (Frank Deinzer, Joachim Denzler, Heinrich Niemann): Auszeichnung für die Arbeit „On Fusion of Multiple Views for Active Object Recognition“