64-424
Intelligent Robotics

https://tams.informatik.uni-hamburg.de/lectures/2018ws/vorlesung/ir

Marc Bestmann / Michael Görner / Jianwei Zhang

University of Hamburg
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics
Technical Aspects of Multimodal Systems

Winterterm 2018/2019

Outline

Outline

State estimation

State estimation addresses the issue of recovery of state information from noisy sensor measurement data

- *Issue*: State variables cannot be measured directly
- *Idea*: Estimation of state variables through a probabilistic approach

- **Example:** Mobile robot localization

- Probabilistic state estimation algorithms calculate a belief distribution over possible states
- The belief describes the knowledge of a system about the state of its environment

Basic concepts

Sensor measurements, control variables and the state of a system and its environment can be modeled as a random variable

- Let X be a random variable and x a value which can be assigned to X
- If the value range of X is discrete, one writes

$$p(X = x)$$

to express the probability of X taking on the value x

Basic concepts (cont.)

For the sake of simplicity, we can write $p(x)$ instead of $p(X = x)$

- The sum of discrete probabilities is 1:

$$\sum_x p(x) = 1$$

- Probabilities are always non-negative, that means

$$p(x) \geq 0$$

Basic concepts (cont.)

If the value range of a random variable is continuous, the variable is said to possess a probability density function (PDF)

- A typical density function is the normal distribution with mean value μ and variance σ^2:

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\}$$

- If x is a multi-dimensional vector
  - μ becomes a mean *vector*
  - σ^2 is replaced by Σ, a *covariance matrix*

Basic concepts (cont.)

- Similar to the discrete probability distribution, a PDF integrates to 1

$$\int p(x)dx = 1$$

- Unlike discrete probabilities, the value of a PDF does not have an upper bound of 1

Basic concepts (cont.)

- The joint probability of X having the value x and Y having the value y is given by

$$p(x, y) = p(X = x \text{ and } Y = y)$$

- If both random variables X and Y are *independent* of each other, one has

$$p(x, y) = p(x)p(y)$$

- If it is known that Y has the value y, the probability for X under the condition $Y = y$ is given by

$$p(x|y) = p(X = x|Y = y)$$

Basic concepts (cont.)

- If one has $p(y) > 0$ for this conditional probability, the following applies

$$p(x|y) = \frac{p(x,y)}{p(y)}$$

- If X and Y are *independent* variables, one has:

$$p(x|y) = \frac{p(x)p(y)}{p(y)} = p(x)$$

- Thus, if X and Y are independent variables, Y doesn't tell us anything about X

Basic concepts (cont.)

The theorem of total probability relates outcome probabilities to conditional probabilities

$$
\begin{aligned}
p(x) &= \textstyle\sum_y p(x|y)p(y) && \text{(discrete)} \\
p(x) &= \textstyle\int p(x|y)p(y)dy && \text{(continuous)}
\end{aligned}
$$

Basic concepts (cont.)

The Bayes rule [1] relates the conditional probability $p(x|y)$ to its "inverse" $p(y|x)$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')} \quad (discrete)$$

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\int p(y|x')p(x')dx} \quad (continuous)$$

- Bayes rule describes the reversion of conclusions
  - The calculation of $p(effect|cause)$ is usually simple
  - But $p(cause|effect)$ carries more information

[1] The rule requires $p(y) > 0$

Basic concepts (cont.)

The Bayes rule plays a fundamental role in state estimation

- If x is the quantity which we want to infer from y, then $p(x)$ is called the prior probability distribution and y is called data (e.g. sensor measurements)
- The distribution $p(x)$ describes the knowledge about X before taking the measurement y into consideration
- The distribution $p(x|y)$ is referred to as the posterior probability distribution of X

- It becomes possible to determine the posterior $p(x|y)$ using the conditional probability $p(y|x)$ and the prior probability $p(x)$

Basic concepts (cont.)

- In Bayes rule, $p(y)$ does not depend on x
- Therefore, the factor $p(y)^{-1}$ is equal for all values x in $p(x|y)$
- Bayes rule calls this factor the normalization factor:

$$p(x|y) = \eta p(y|x)p(x)$$

- This notation describes the normalization of the result to 1

Basic concepts (cont.)

All previous rules may be conditioned on an additional random variable Z

- Conditioning the Bayes rule on $Z = z$ gives us:

$$p(x|y, z) = \frac{p(y|x, z)p(x|z)}{p(y|z)}$$

  as long as $p(y|z) > 0$ is true

- Similar to the rule of combination of independent random variables, the following applies:

$$p(x, y|z) = p(x|z)p(y|z)$$

Basic concepts (cont.)

- Previous formula describes a conditional independence and is equivalent to

$$\begin{aligned} p(x|z) &= p(x|z,y) \\ p(y|z) &= p(y|z,x) \end{aligned}$$

- The formula implies that y carries no information about x, if z is known
- It does **not** imply, that X is independent of Y:

$$p(x,y|z) = p(x|z)p(y|z) \quad \not\Rightarrow \quad p(x,y) = p(x)p(y)$$

The converse generally does not apply as well:

$$p(x,y) = p(x)p(y) \quad \not\Rightarrow \quad p(x,y|z) = p(x|z)p(y|z)$$

Outline

1. State estimation
   - Fundamentals
   - State and belief
   - Bayes filter
   - Mobile robot localization

State

The state of a system can be described through a probability distribution

$$p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$$

which depends on:

- All previous states $x_{0:t-1}$
- All previous measurements $z_{1:t-1}$ and
- All previous control variables (control commands) $u_{1:t}$

State (cont.)

A state x is said to be complete, if knowledge of past states does not carry any information that would improve the estimate of the future state

- Assuming a complete state only the control variable u_t is important if state x_{t-1} is known ($\rightarrow$ conditional independence)

$$p(x_t | x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t)$$

- The measurement probability distribution is specified in a similar way

$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t)$$

- In other words: The state x_t is sufficient to predict the measurement z_t

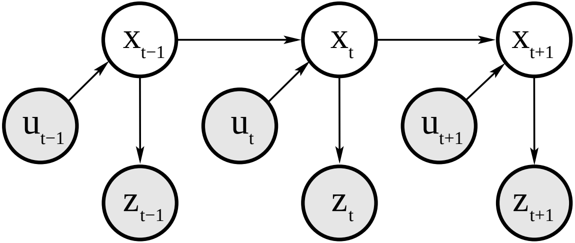
State (cont.)

- The conditional probability $p(x_t | x_{t-1}, u_t)$ is called state transition probability
- It describes how the state of the environment changes depending on the control variables

- The probability $p(z_t | x_t)$ is called measurement probability

- Both probabilities together describe a dynamic stochastic system
- Such as system description is also known as Hidden Markov Model (HMM) or Dynamic Bayes Network (DBN)

State (cont.)



A dynamic Bayes network describing the development of states, measurements and controls

Belief

The knowledge of a system about its state is called belief

- The *true state* of a system is **not equal** to the *belief*
- The *belief* is the posterior probability of the state variable based on previous measurement data

$$bel(x_t) = p(x_t | z_{1:t}, u_{1:t})$$

- This definition defines the *belief* as probability after measurement

Belief (cont.)

- The *belief* before incorporation of measurements is called the prediction

$$\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{1:t})$$

- The step of calculating $bel(x_t)$ from the prediction $\overline{bel}(x_t)$ is called correction or measurement update

Outline

1. State estimation
   - Fundamentals
   - State and belief
   - Bayes filter
   - Mobile robot localization

Bayes filter

The most fundamental algorithm to calculate *beliefs* is the Bayes filter algorithm

- The algorithm is recursive and calculates the belief distribution $bel(x_t)$ at time t from the following quantities
  - $bel(x_{t-1})$ at the time of $t-1$
  - The measurement data z_t
  - The control data u_t

Bayes filter (cont.)

The general Bayes filter algorithm

Algorithm Bayes_Filter($bel(x_{t-1})$, u_t, z_t):

1. **for all** x_t **do**
2. $\qquad \overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$
3. $\qquad bel(x_t) = \eta\, p(z_t | x_t) \overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

Bayes filter (cont.)

The Bayes filter algorithm has two essential steps

- In line 2, it processes the control variable u_t
- $\overline{bel}(x_t)$ is the integral (sum) of the product of two probability distributions:
  - The prior for state x_{t-1} and
  - The probability of switching to state x_t when u_t occurs
- That is the prediction step

- In line 3, the correction step is executed
- $\overline{bel}(x_t)$ is multiplied with the probability of detection of the measurement z_t in this state

Bayes filter algorithm (cont.)

- Due to its recursive nature the Bayes filter requires an initial belief $bel(x_0)$ at time $t = 0$ as a boundary condition

- If the initial state x_0 is known with certainty, $bel(x_0)$ should be initialized with a *point mass distribution* focused on x_0
- If the initial state is completely unknown, $bel(x_0)$ should be initialized with a *uniform distribution*

Bayes filter algorithm (cont.)

- In the presented form, the algorithm can only be implemented for very simple problems

- Either the integration in line 2 and the multiplication in line 3 need to have a closed form solution, ...
- ...or a finite state space must be given, so that the integral in line 2 becomes a sum

Bayes filter - an example

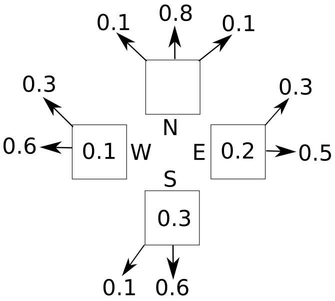
- Assume an agent in this small grid world

a	b	c
d	e	f

- The agent's *state* is $x \in \{a, b, c, d, e, f\}$
- The agent's belief is a 6-dimensional distribution $bel(x)$
- The agent can aim to move (*transition*) North, East, South, and West
- It can measure its *longitude* (i.e. column)

Bayes filter - an example (cont.)

- The agent can choose $u \in \{N, E, S, W\}$
- It might end up somewhere else though:



- The agent can measure its current column $z \in \{-1, 0, 1\}$
- The measurement might be faulty

-1	0	1
0.25	0.5	0.25

- When the agent would hit a wall, it moves along the wall instead

Bayes filter - an example (cont.)

- Assume some distribution as the initial belief $bel(x_0)$
- Choose an action u_1 and compute $\overline{bel}(x_1)$
- Assume a measurement z_1 and compute $bel(x_1)$

Bayes filter - Example

x_0 :

0	0	0
0	1	0

u_1 : N

$\overline{bel}_1$:

$1 * 0.1$	$1 * 0.8$	$1 * 0.1$
0	0	0

=

0.1	0.8	0.1
0	0	0

z_1 : 0

bel_1 :

$0.1 * 0.25$	$0.8 * 0.5$	$0.1 * 0.25$
0	0	0

=

0.025	0.4	0.025
0	0	0

normalized

0.056	0.89	0.056
0	0	0

Bayes filter - Example

bel_1

0.056	0.89	0.056
0	0	0

$u_2 : S$

$\overline{bel_2}$:

$0.056 * 0.3$	$0.89 * 0.3$
$0.056 * 0.1 + 0.056 * 0.6 + 0.89 * 0.1$	$0.89 * 0.6 + 0.05 * 0.1$

$0.056 * 0.3$
$0.056 * 0.3 + 0.056 * 0.6$

=

0.0168	0.267	0.0168
0.1282	0.539	0.0504

normalize it again

do measurement

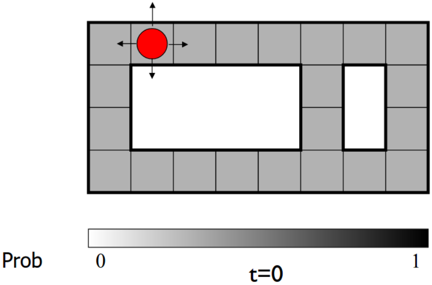
...

University of Hamburg

MIN Faculty
Department of Informatics

Bayes filter - example 2

Example from Michael Pfeiffer



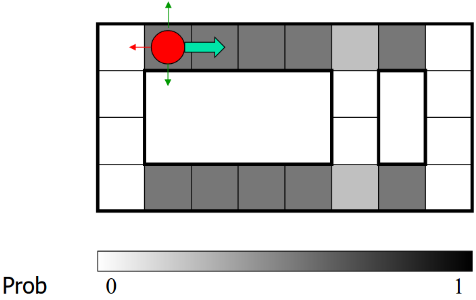
Sensor model: never more than 1 mistake

Know the heading (North, East, South or West)

Motion model: may not execute action with small prob.

https://people.eecs.berkeley.edu/ pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes filter - example 2

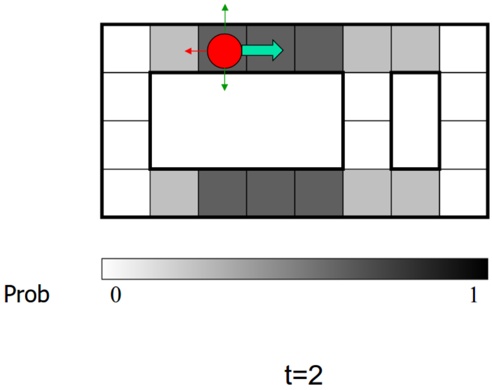


t=1

Lighter grey: was possible to get the reading, but less likely b/c required 1 mistake

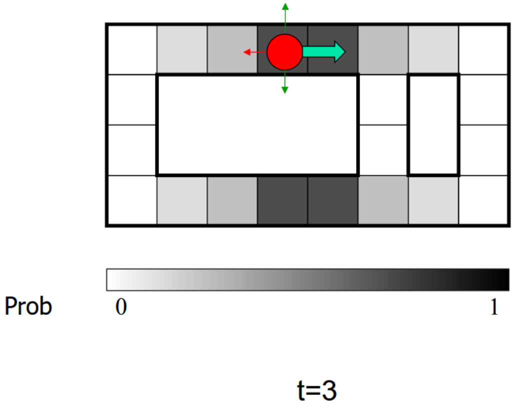
https://people.eecs.berkeley.edu/ pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes filter - example 2



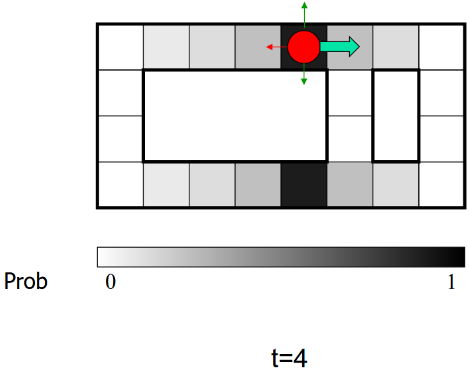
https://people.eecs.berkeley.edu/ pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes filter - example 2



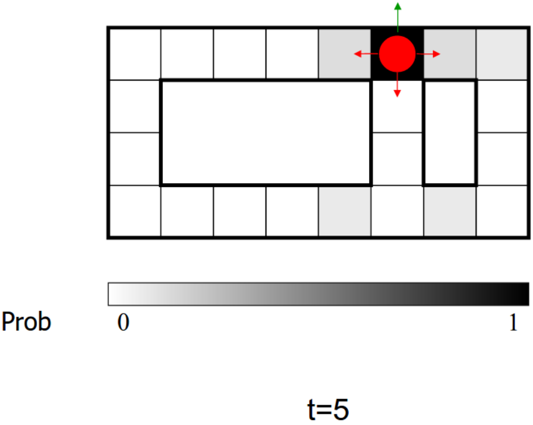
https://people.eecs.berkeley.edu/ pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes filter - example 2



https://people.eecs.berkeley.edu/ pabbeel/cs287-fa13/slides/bayes-filters.pdf

Bayes filter - example 2



Markov assumption

The assumption of a state being complete is called Markov assumption

- The assumption states independence of past and future data, if the current state x_t is known

The following is meant to illustrate, how tough this assumption is:

- Assuming that Bayes filters are used for localization of mobile robots, . . .
- . . . and x_t is the *pose* of the robot in relation to a static map

Markov assumption (cont.)

There are effects which falsify sensor measurements systematically and therefore render the Markov assumption void:

- Inaccuracies in the probabilistic models $p(x_t | u_t, x_{t-1})$ and $p(z_t | x_t)$
- Rounding errors, if approximations for the representation of the *belief* are used
- Variables within the software, which affect several control variables
- Influence of moving persons on sensor measurements

Some of these variables could be included in the state, but are often abandoned in order to reduce computational effort

Bayes filters

Bayes filters (based on the general filter itself) can be implemented in different ways

- The techniques are based on varying assumptions regarding the probability of the measurements, the state transitions and the *belief*
- In most cases the *beliefs* need to be approximated
- This affects the complexity of the algorithms
- Generally none of these techniques should be favored of the others

Bayes filters (cont.)

Various Bayes filter implementations express different runtime behavior

- Some approximations require a polynomial runtime, depending on the dimensionality of the state (e.g. Kalman filter)
- Some filters have an exponential runtime

- The runtime of particle based procedures depends on the desired accuracy

Bayes filters (cont.)

Some approximations are better suited to approximate a range of probability distributions

- For uni-modal probability distributions, for example, normal distributions qualify
- Histograms can approximate multi-modal distributions, at the cost of accuracy and computational load
- Particle techniques can approximate a wide range of distributions, possibly resulting in a large number of particles

Summary

Interaction between a robot and its environment is modeled as a coupled dynamic system. For this purpose, the robot sets control variables to manipulate the environment and perceives the environment through sensor measurements

- System dynamics are characterized through two laws of probability theory
  - Probability distribution for the state transition
  - Probability distribution for the measurements

  The first one describes how the state changes over time, the second one describes how measurements are perceived

Summary (cont.)

- The *belief* is the posterior probability of the state, given all previous measurements and control variables
- The *Bayes filter* is a general (recursive) algorithm for calculation of the *belief*
- The Bayes filter works based on the *Markov assumption* $\rightarrow$ The state is a complete summary of the past. In practice, this assumption is usually not true.
- Usually, the Bayes filter can not be applied directly. Implementations can be evaluated based on certain criteria, such as accuracy, efficiency and simplicity.

Outline

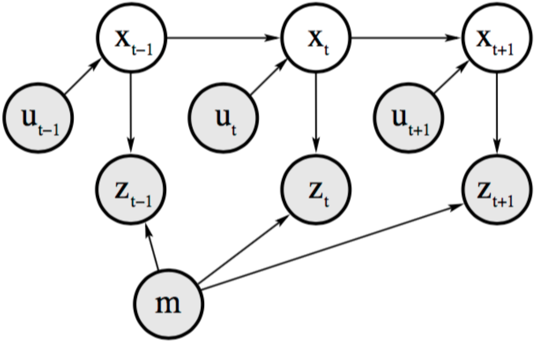
1. State estimation
   - Fundamentals
   - State and belief
   - Bayes filter
   - Mobile robot localization

Localization

A robot's ability to determine its location relative to a map of the environment

- Position tracking
  - Initial robot pose is *known*
  - Localization after control command
  - Pose uncertainty often approximated by a uni-modal distribution
  - Position tracking is a local problem (relative localization)
- Global localization
  - Initial robot pose is *unknown*
  - Uni-modal distributions are no longer appropriate
  - Absolute localization approach
  - Variant: Kidnapped Robot Problem

Localization (cont.)



Map m, measurements z and controls u are known, robot pose x must be inferred

Localization (cont.)

Maps are usually specified in one of two forms

- Location-based
  - Planar map with $m_{x,y}$ representing coordinate points
  - Maps are *volumetric*, every point is *labeled*
  - Information about objects in the environment and free space
- Feature-based
  - Map with m_n representing features (objects) in the environment
  - Loss of information, shape of environment known at feature locations only
  - Compact and efficient representation

Markov localization

Probabilistic localization approaches are variants of the Bayes filter

- The Bayes filter approach can be applied directly → Markov localization

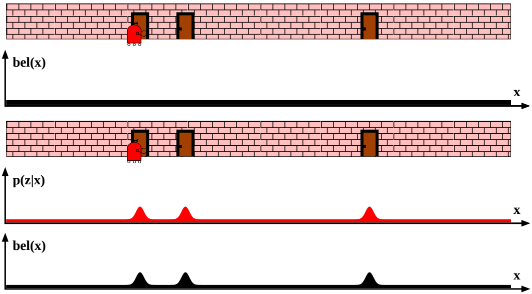
- Markov localization requires a map m of the environment
- The map plays a role in the motion and measurement models
- Markov localization is suitable for position tracking and global localization problems in static environments

Markov localization (cont.)

Algorithm Markov_Localization($bel(x_{t-1})$, u_t, z_t, m):

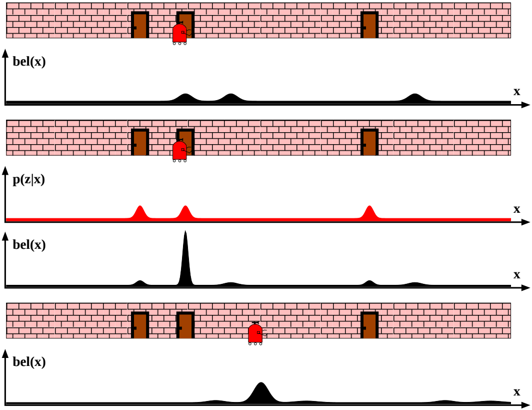
1. **for all** x_t **do**
2. $\quad \overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\, bel(x_{t-1})\, dx_{t-1}$
3. $\quad bel(x_t) = \eta\, p(z_t \mid x_t, m) \overline{bel}(x_t)$
4. **endfor**
5. **return** $bel(x_t)$

Markov localization (cont.)



Convolution of prior with motion model followed by incorporation of the measurement model.

Markov localization (cont.)



Localization (cont.)

Kalman filter based localization approaches

- Belief $bel(x_t)$ represented by uni-modal Gaussian $\mathcal{N}(\mu_t, \Sigma_t)$
- Suitable for pose tracking
- Efficient means for integration of multiple sensors
- Map-based localization requires uniquely identifiable features

Particle filter based localization approaches

- Belief $bel(x_t)$ represented by particles
- Particles are discrete samples of the state probability distribution
- Suitable for pose tracking and global localization problems

Kalman filter

The Kalman filter assumes linear system dynamics

- The state transition probability must be a linear function with added Gaussian noise

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t$$

- ϵ_t models the uncertainty introduced by the state transition, with its covariance denoted by R_t
- The measurement probability must also be a linear function with added Gaussian noise

$$z_t = C_t x_t + \delta_t$$

- C_t is the measurement matrix and δ_t is a zero mean Gaussian with covariance denoted by Q_t

Kalman filter (cont.)

- K_t represents the Kalman gain, a specification of the degree to which the measurement is incorporated into the new state estimate

Kalman filter (cont.)

Algorithm Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
5. $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$
6. **return** μ_t, Σ_t

Kalman filter (cont.)

Advantages:

- Highly efficient (prediction and correction steps in closed form)
- **Optimal for linear Gaussian systems**

The correctness of the Kalman filter crucially depends on the assumptions that the measurements are a linear function of the state and that the next state is a linear function of the current state

- Most problems in robotics are non-linear
  - State transitions and measurements are usually non-linear
  - **So the Kalman filter is not directly applicable!**

Extended Kalman filter

The Extended Kalman filter (EKF) relaxes the *linearity* assumption

- State transition probability and measurement probability

$$
\begin{aligned}
x_t &= g(u_t, x_{t-1}) + \epsilon_t \\
z_t &= h(x_t) + \delta_t
\end{aligned}
$$

- However, the belief is no longer a Gaussian
- EKF calculates a *Gaussian approximation* to the true belief
- The approximation is determined through linearization
  - Non-linear functions g and h are approximated by linear functions that are tangent to g or h at the mean of the Gaussian
  - This makes use of their Jacobian matrices G_t and H_t

Jacobian Matrix

- The Jacobian Matrix J_f of a function $f : \mathcal{R}^n \rightarrow \mathcal{R}^m$ is the matrix of all first-order partial derivatives of a vector-valued function.

$$(J_f)_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$J_f = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Extended Kalman filter (cont.)

Algorithm Extended_Kalman_Filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:

1. $\bar{\mu}_t = g(u_t, \mu_{t-1})$
2. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
4. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
5. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$
6. **return** μ_t, Σ_t

Extended Kalman filter (cont.)

Kalman filter vs. Extended Kalman filter

- The algorithms are quite similar and share several properties
- Most important difference concerns *state prediction* (line 1) and *measurement prediction* (line 4)
  - Linear predictions $\rightarrow$ Non-linear generalizations
- Additionally, EKF uses Jacobians G_t and H_t instead of the corresponding linear system matrices A_t, B_t and C_t

Extended Kalman filter - an example

- Let a robot's state be characterized by $X = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$
- The robot can move forward by d meter and turn by φrad, but only turns after moving. This can be represented by $u = \begin{pmatrix} d \\ \varphi \end{pmatrix}$
- It can measure its absolute orientation θ (by IMU)
- Define the transition and the measurement model g and h and the covariance matrices of their noise terms, and compute their Jacobian Matrices G and H
- Assume some initial belief $bel(x_0)$, an action u_1, and a measurement z_1 and compute $bel(x_1)$

Extended Kalman filter - an example

$$g(\begin{pmatrix} d \\ \varphi \end{pmatrix}, \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}) = \begin{pmatrix} x + d * cos(\theta) \\ y + d * sin(\theta) \\ \theta + \varphi \end{pmatrix}; \epsilon_t \sim \mathcal{N}(0, R_u)$$

$$R_u = \begin{pmatrix} 0.01 * d & 0 & 0 \\ 0 & 0.01 * d & 0 \\ 0 & 0 & 0.01 * \varphi \end{pmatrix}$$

$$h(\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}) = \theta; \delta_t \sim \mathcal{N}(0, Q); Q = 0.01$$

$$G_t = \begin{bmatrix} \frac{dg}{dx} & \frac{dg}{dy} & \frac{dg}{d\theta} \end{bmatrix} = \begin{pmatrix} 1 & 0 & -d * sin(\theta) \\ 0 & 1 & d * cos(\theta) \\ 0 & 0 & 1 \end{pmatrix}$$

$$H = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

Extended Kalman filter - an example

$$bel(X_0) = \mathcal{N}(\mu_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \Sigma_0 = 0)$$

$$u_1 = \begin{pmatrix} 1.0 \\ 1.6 \end{pmatrix}$$

$\overline{bel}(X_1)$:

$$\overline{\mu_1} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1.0 * cos(0) \\ 1 * sin(0) \\ 0 + 1.6 \end{pmatrix} = \begin{pmatrix} 1.0 \\ 0 \\ 1.6 \end{pmatrix}$$

$$\overline{\Sigma_1} = G_1 * \Sigma_0 * G_1^T + R_1 = \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.16 \end{pmatrix}$$

Extended Kalman filter - an example

$$K_1 = \begin{pmatrix} 0 \\ 0 \\ \frac{16}{17} \end{pmatrix}, z_1 = 2.0$$

$bel(X_1)$:

$$\mu_1 = \overline{\mu_1} + K_1(z_1 - h(\overline{\mu_1})) = \begin{pmatrix} 1.0 \\ 0 \\ 1.6 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \frac{16}{17} \end{pmatrix} * 0.4 = \begin{pmatrix} 1.0 \\ 0 \\ 1.98 \end{pmatrix}$$

$$\Sigma_1 = (1 - K_1 * H) * \overline{\Sigma_1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1}{17} \end{pmatrix} * \overline{\Sigma_1} \approx \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$$

Extended Kalman filter - an example

$$bel(X_0) = \mathcal{N}(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix})$$

$$u_1 = \begin{pmatrix} 1.0 \\ 1.6 \end{pmatrix}$$

$$\overline{bel(X_1)} = \mathcal{N}(\begin{pmatrix} 1.0 \\ 0 \\ 1.6 \end{pmatrix}, \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.16 \end{pmatrix})$$

$$z_1 = 2.0$$

$$bel(X_1) = \mathcal{N}(\begin{pmatrix} 1.0 \\ 0.0 \\ 1.98 \end{pmatrix}, \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix})$$

Kalman filter online demo

https://www.cs.utexas.edu/ teammco/misc/kalman_filter/

Extended Kalman filter (cont.)

Advantages:

- Highly efficient
- Useful for multi-sensor fusion
- Once non-linear functions g and h are linearized, the prediction and update procedures are equivalent to those of the Kalman filter

Disadvantages:

- Not optimal $\rightarrow$ Belief is approximated
- Can diverge if non-linearities are large

EKF localization

The *Extended Kalman filter* localization is a special case of Markov localization

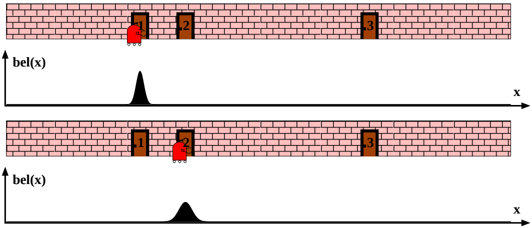
- **Assumption:** The map of the environment is represented as a collection of features

At any point in time the robot observes a vector of ranges to nearby features

- Features can be assumed to be *uniquely identifiable*

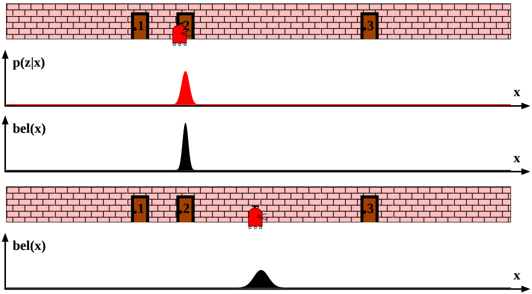
$$z_t = (z_t^1, z_t^2, \ldots, z_t^m)$$

EKF localization (cont.)



Uniquely identifiable features. Good knowledge about initial pose followed by convolution with motion model.

EKF localization (cont.)



- Belief remains Gaussian at any point in time
- If unique feature identification is not given, maximum likelihood estimation can provide correspondances

Unscented Kalman filter

The Unscented Kalman filter (UKF) is a variant of the Kalman filter that improves the belief estimate through a stochastic linearization method: the unscented transform

- It uses a weighted statistical linear regression process

Prediction and correction steps are preceeded with a sigma-point extraction step

1. Deterministic extraction of *sigma-points* [2]
2. Assignment of weights to extracted points
3. Transform of points through non-linear functions g and h
4. Computation of Gaussian from weighted points

[2] Located at the mean and along the axes of the covariance

Unscented Kalman filter (cont.)

- Highly efficient: Same complexity as EKF (constant factor slower in typical practical applications)
- Better linearization than EKF
- For purely linear problems belief estimate is *equal* to that generated by a Kalman filter
- For non-linear problems the estimate is *equal or better* than that generated by EKF
- UKF is a *derivative-free filter*: No Jacobians needed

- Still not optimal

KF based localization

- EKF and UKF localization are only applicable to *pose tracking* problems
- Linearized Gaussian approaches work well only if the pose uncertainty is small
- Linearization is usually only good in close proximity to the linearization point

- EKF and UKF localization process only a subset of all information in the sensor measurement data
- On the other hand it allows the efficient integration of measurements from multiple sources

Why do I need a Kalman filter?

39

20

I am designing an unmanned aerial vehicle, which will include several types of sensors:

- 3-axis accelerometer
- 3-axis gyroscope
- 3-axis magnetometer
- horizon sensor
- GPS
- downward facing ultrasound.

A friend of mine told me that I will need to put all of this sensor data through a Kalman filter, but I don't understand why. Why can't I just put this straight into my micro controller. How does the Kalman filter help me about my sensor data?

kalman-filter uav

share improve this question

edited Nov 11 '12 at 16:14
Atilla Ozgur
103 • 3

asked Nov 5 '12 at 23:18
Rocketmagnet
4,133 • 1 • 16 • 43

Grid localization

Grid localization approximates the *belief* using a Histogram filter applied to the grid decomposition of the state space

- Discretization of the state space through grid cells x
- Allows multimodal distributions
- This discrete Bayes filter handles a multitude of discrete probabilities

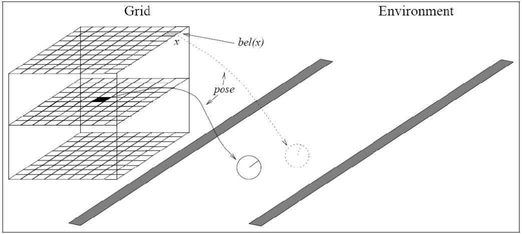
$$bel(x_t) = \{p_{k,t}\}$$

  where each $p_{k,t}$ belongs to a grid cell x_k
- The union of all cells at time t represents the state space X_t
- Two typical grid decomposition approaches exist

Grid localization (cont.)

(1) Metric grid decomposition

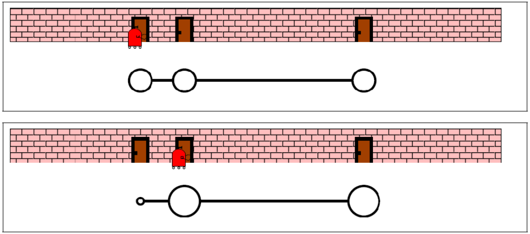
- Grid cells of equal size
- Typical cell sizes have about 15*cm* depth resolution at about 5° angular resolution
- Higher resolution compared to the topological grid at the cost of an increased computational effort



Grid localization (cont.)

(2) Topological grid decomposition

- Cell represents a significant location/feature on the map (Example: Door, Junction . . .)
- Resulting grid is usually very coarse
- Grid depends on local map structure/conditions/data



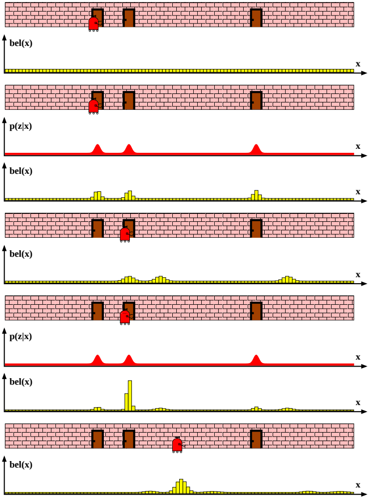
Grid localization (cont.)

Grid_Localization($\{p_{k,t-1}, u_t, z_t, m\}$):

1. **for all** k **do**
2. $\quad \bar{p}_{k,t} = \sum_i \left[p_{i,t-1} \cdot motion\_model(mean(x_k), u_t, mean(x_i)) \right]$
3. $\quad p_{k,t} = \eta \cdot measurement\_model(z_t, mean(x_k), m) \cdot \bar{p}_{k,t}$
4. **endfor**
5. **return** $p_{k,t}$

The function *mean* determines the center of mass of a cell x_i

Grid localization



Particle filter based localization

- Representation of belief by random samples (particles)
- Instead of representing *parameterized distributions* one can also reason with *samples from the distribution*
- Estimation of multi-modal, non-Gaussian, non-linear processes

- Monte Carlo filter is the most popular particle based technique
- Applicable to position tracking and global localization problems
- Naive versions of the algorithm are simple to implement

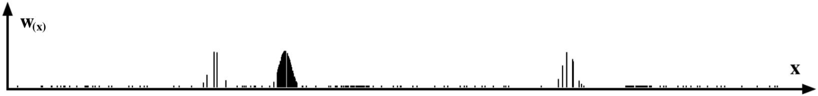
Monte Carlo localization

- *Monte Carlo localization* (MCL) approximates the belief $bel(x_t)$ through a set of M particles χ_t

$$\chi_t = \{\langle x_t^i, w_t^i \rangle \| x_t^i \in X_t, w_t^i \in \mathcal{R}^+\}$$

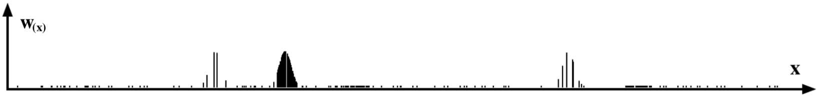
with $i = 1 \ldots M$ and state space X_t at time t

- Each sample is assigned an *importance weight* w_t^i



Monte Carlo localization (cont.)

- Discrete approximation of a probability distribution
- More particles can represent more complex distributions
- Approximation of any distribution is possible *in theory*
- Algorithm is structurally similar to Markov localization, intertwining motion model and sensor model updates



Monte Carlo localization (cont.)

- To focus particles on *important regions* of the state space, Monte Carlo methods apply a *resampling* step
  - **Resampling**: Selection of a new set of samples χ_t ...
    - ... from elements of the old sample set χ_{t-1} ...
    - ... generating new samples if necessary
- This ensures that samples with low weights get replaced by more important samples
- It might add alternative hypotheses that were not represented
- Resampling was a major breakthrough for particle filters and made them feasible in practice

Monte Carlo localization (cont.)

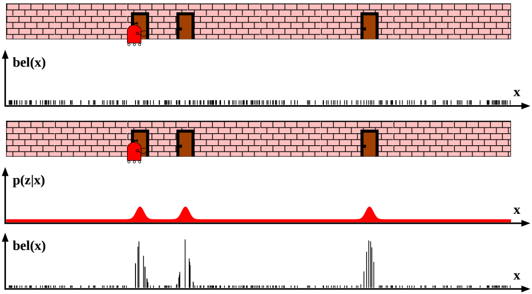
Algorithm Monte_Carlo_Localization(χ_{t-1}, u_t, z_t, m):

1. $\bar{\chi}_t = \chi_t = \emptyset$
2. $\#$*update step*
3. **for** $m = 1$ **to** M **do**
4. $\quad x_t^{[m]} = sample\_motion\_model(u_t, x_{t-1}^{[m]})$
5. $\quad w_t^{[m]} = measurement\_model(z_t, x_t^{[m]}, m)$
6. $\quad \bar{\chi}_t = \bar{\chi}_t \cup \{\langle x_t^{[m]}, w_t^{[m]} \rangle\}$
7. **endfor**

Monte Carlo localization (cont.)

8. #*resampling step*
9. **for** $i = 1$ **to** M **do**
10. *draw* $x_t^{[i]}$ *favoring larger* $w_t^{[i]}$
11. *add* $x_t^{[i]}$ *to* χ_t
12. **endfor**
13. **return** χ_t

Monte Carlo localization (cont.)

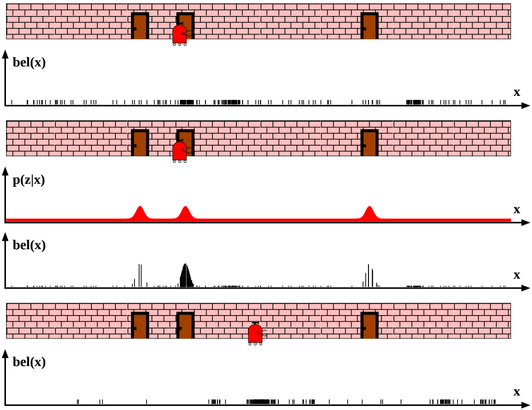


Random initialization. Incorporation of the motion model with weighting of the samples.

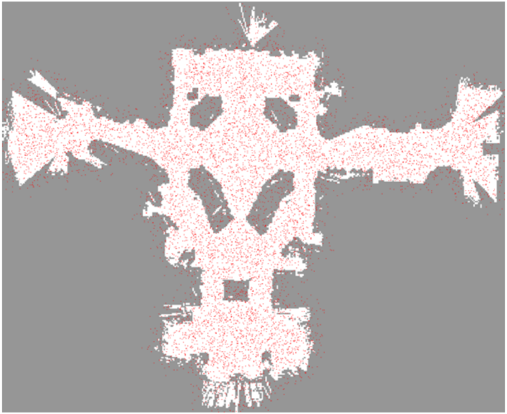
Adaptive Sample Size

- The number of considered particles can be altered online
- If the distribution of the current belief changes its complexity at runtime, the number of particles can be adjusted accordingly
- This is not easy to detect! Common attempts:
  - *Likelihood-based adaptation*:
    If measurements agree with most particles, fewer particles are needed
  - *KLD-sampling*:
    If the expected area of important regions changes, sample size can be adjusted to bound the error in terms of its KL-distance

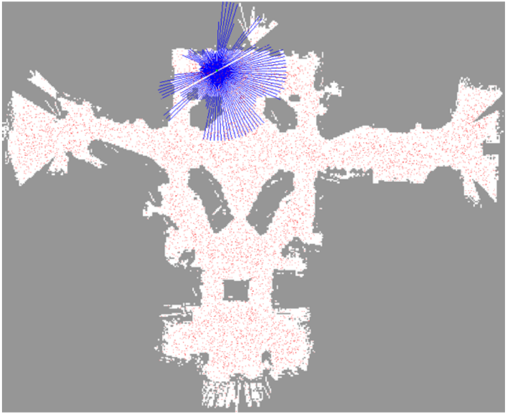
Monte Carlo localization (cont.)



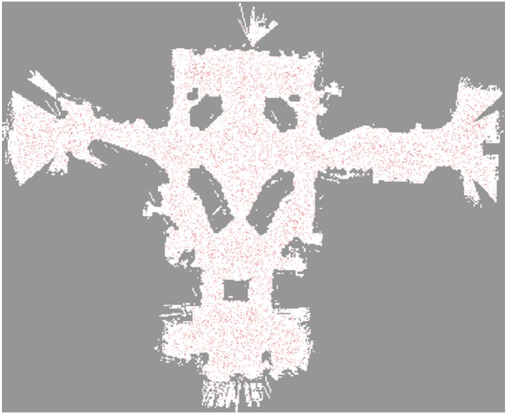
Monte Carlo localization (cont.)



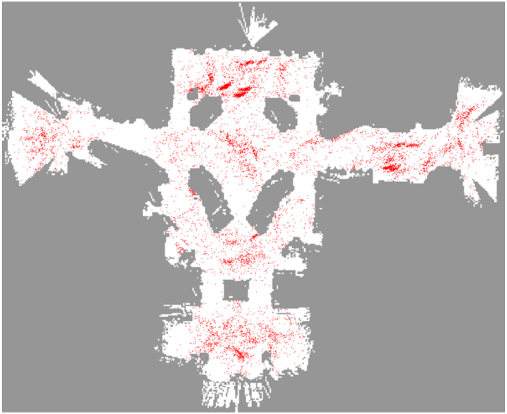
Monte Carlo localization (cont.)



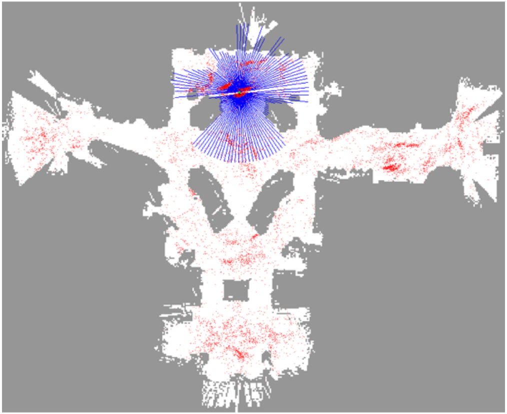
Monte Carlo localization (cont.)



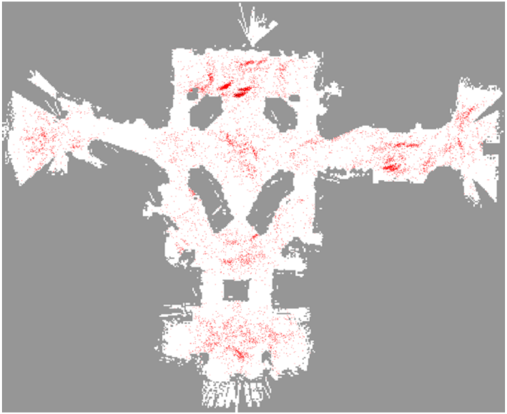
Monte Carlo localization (cont.)



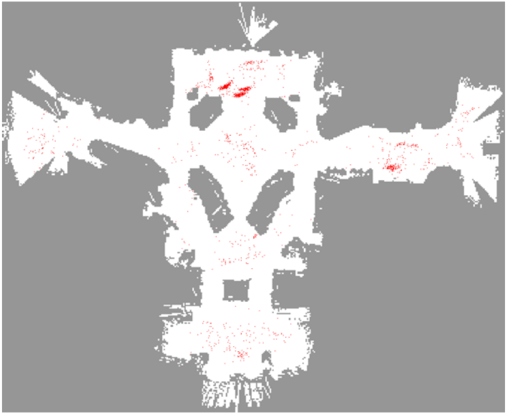
Monte Carlo localization (cont.)



Monte Carlo localization (cont.)

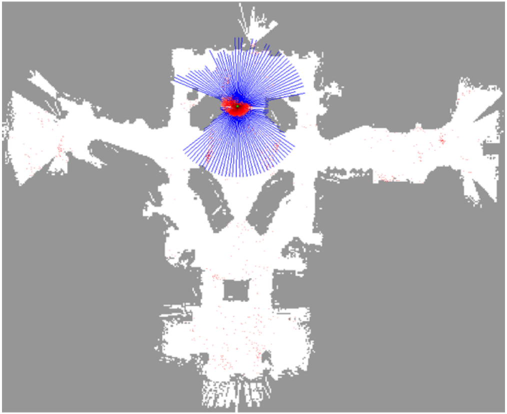


Monte Carlo localization (cont.)



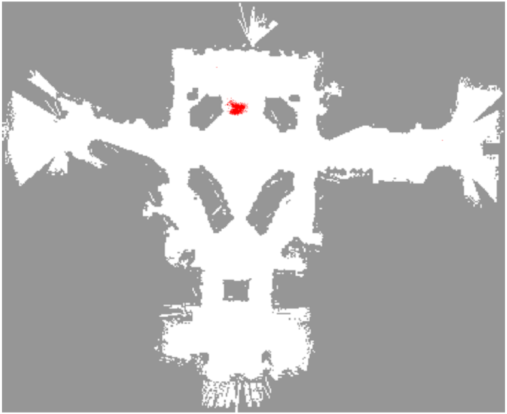
Monte Carlo localization (cont.)

Monte Carlo localization (cont.)

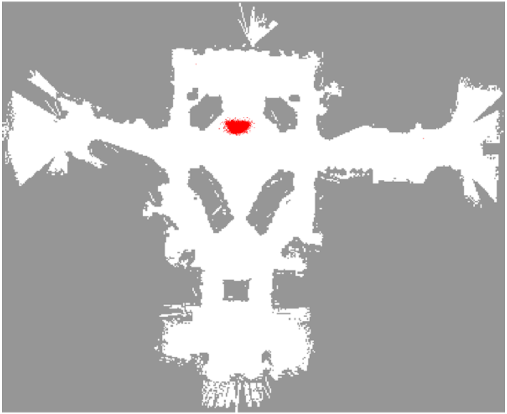


Monte Carlo localization (cont.)

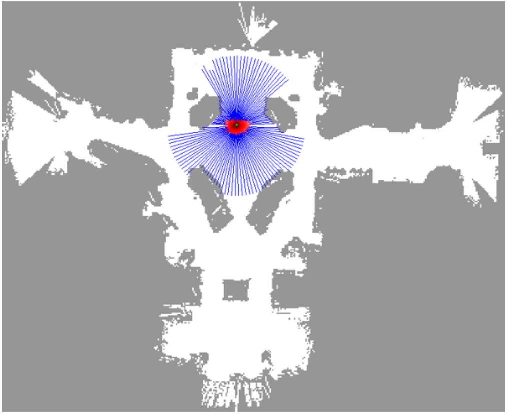
Monte Carlo localization (cont.)



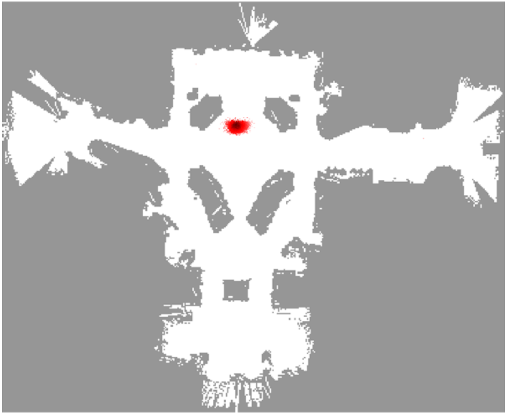
Monte Carlo localization (cont.)



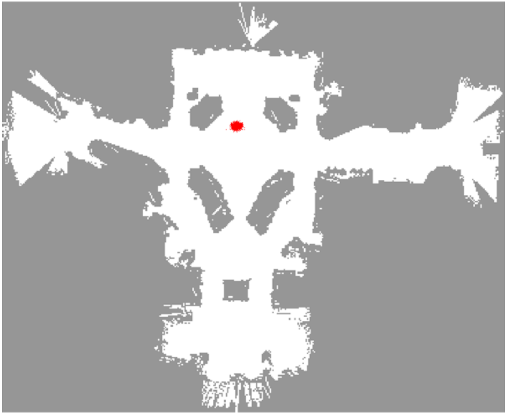
Monte Carlo localization (cont.)



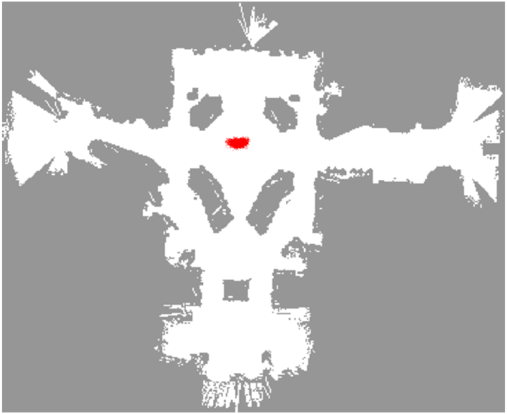
Monte Carlo localization (cont.)



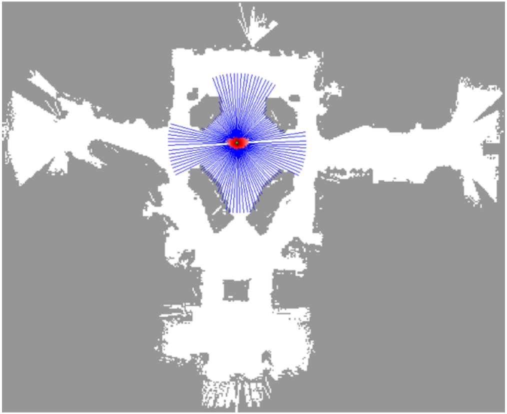
Monte Carlo localization (cont.)



Monte Carlo localization (cont.)



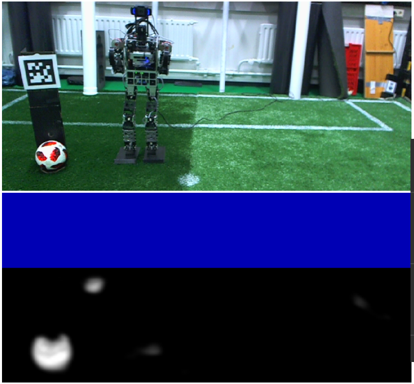
Monte Carlo localization (cont.)



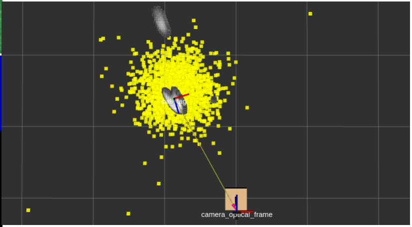
Particle Systems: Other Applications

- Particle-based inference is not restricted to a Pose state space
- Example Particle-based SLAM (*gmapping*)
  - Particles model the robot's pose *and an occupancy grid*, i.e. a probabilistic 2D map
  - Measurements weight *and update* particles
- Example FastSLAM
  - Each particle encapsulates the robot's pose *and extended Kalman filters for each landmark*
- Particle-based Inverse Kinematics
  - Particles represent joint angles of robotic manipulators
  - Optimization w.r.t. target pose and secondary objectives
- ...

Application Example



- Non gaussian, multi-modal
- Filtering of ball position
- Direct use of FCNN output



Literature list

[1] Sebastian Thrun, Wolfram Burgard, and Dieter Fox.
Probabilistic Robotics, chapter 2-4; 7-8, pages 13–116; 191–278.
MIT Press, 1. edition, 2005.