Technical Cognitive Systems

Michael Beetz

Intelligent Autonomous Systems
Technische Universität München

Summer Term 2012
Part IV

State Estimation
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
Where are we?

Diagram showing controlled and controlling processes involving plant, environment, sensors, robot behavior, and state estimators and monitors.
Motivation
Probabilistic Robotics

- Key idea: explicit representation of uncertainty
- Using the calculus of probability theory
- perception = state estimation
- Action = selecting actions with the highest expected utility
Demonstrations of State Estimation

- Global localization
- Adaptive global localization
- Fast SLAM
- Distributed Mapping
- People Tracking
- Inferring Transportation Routines
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
The axioms of probability theory

- $P(A)$ denotes the probability that proposition $A$ is true
- $0 \leq P(A) \leq 1$
- $P(True) = 1$
- $P(False) = 0$
- $P(A \text{ or } B) = P(A) + P(B) - P(A \text{ and } B)$
Discrete random variables

• $X$ denotes a random variable
• $X$ can take a countable number of values in $\{x_1, ..., x_n\}$
• $P(X = x_i)$, or $P(x_i)$, is the probability that the random variable $X$ has the value $x_i$
• $P(\text{Room}) = \langle 0.7, 0.2, 0.08, 0.02 \rangle$
Continuous random variables

- \( X \) takes on values in the continuum
- \( P(X = x) \) or \( P(x) \) is a probability density function
Joint and conditional probability

- \( P(X = x \land Y = y) = P(x, y) \)
- If \( X \) and \( Y \) are independent then \( P(x, y) = P(x) \ast P(y) \)
- \( P(x \mid y) \) is the probability of \( x \) given \( y \)
- \( P(x \mid y) = \frac{P(x, y)}{P(y)} \)
- \( P(x, y) = P(x \mid y)P(y) \)
- If \( X \) and \( Y \) are independent then \( P(x \mid y) = P(x) \)
Law of total probability, marginals

Discrete case
- \( \sum_x P(x) = 1 \)
- \( P(x) = \sum_y P(x, y) \)
- \( P(x) = \sum_y P(x | y) P(y) \)

Continuous case
- \( \int P(x) = 1 \)
- \( P(x) = \int P(x, y) dy \)
- \( P(x) = \int P(x | y) P(y) dy \)
Bayes formula

- \( P(x, y) = P(x \mid y)P(y) = P(y \mid x)P(x) \)
- \( P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)} \)
- Bayes rule and background knowledge:
  \[
P(x \mid y, z) = \frac{P(y \mid x, z)P(x \mid z)}{P(y \mid z)}
  \]
State

• state = all aspects of the environment that can have impact the future
• state may change over time such as the location of people
• state that changes will be called *dynamic state*
• contrast: *stationary state*
• notation:
  – state \( x \)
  – state at time \( t \): \( x_t \)
Typical state variables

- robot pose: location and orientation relative to a global coordinate frame; often called *kinematic state*
- configuration of the robot’s actuators such as the joints of robotic manipulators. Each degree of freedom in a robot arm is characterized by a one-dimensional configuration
- robot velocities and velocities of its joints. Velocities are commonly referred to as *dynamic state*
- location and features of surrounding objects in the environment
- location and velocities of moving objects and people
- huge number of other state variables: battery charge, ...
Completeness of States

- A state $x_t$ will be *complete* if it is the best predictor of the future, in that knowledge of past state measurements, or controls carry no additional information that would help us to predict the future more accurately.
  - The future may be stochastic but no variables prior to $x_t$ may influence the stochastic evolution of future states, unless the dependence is mediated through $x_t$.
  - Temporal processes that meet these conditions are known as *Markov chains*.

- *Incomplete state* for real robots state cannot be complete.
Environment Interaction

• two types of interactions
  – sensor measurements: measurements, observations, percept
  – control actions

• two kinds of data:
  – measurement data provides information about the momentary state of the environment
    measurement data \( z_t \)
    \[ z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2} \]
  – control data: change of state
    • odometers are sensors that measure the revolution of a robot’s wheels. Even though odometers are sensors we treat them as control data because they are directly tight to the robots’ actions
    control data: \( u_t, u_{t_1:t_2} \)
Probabilistic Generative Laws

- The evolution of state and measurements is governed by probabilistic laws. In general, the state $x_t$ is generated probabilistically. Thus it makes sense to specify the probability distribution from which $x_t$ is generated:
  $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t})$

- if state $x$ is complete: conditional independence
  $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$

- $p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$
Probabilistic State Estimation as a Bayes Net

- map
- observations
- robot poses
- controls

Diagram:

- Map \( m \)
- Observations \( z_1, z_2, z_3 \)
- Robot poses \( x_1, x_2, x_3, x_T \)
- Controls \( u_2, u_3, u_T \)

Basics  Bayesian Filtering  Self Localization  State  Models  RoboCup

Michael Beetz
Summer Term 2012

Technical Cognitive Systems
Belief Distributions

- belief = the robot’s internal knowledge about the state of the environment
- \( bel(x_t) = p(x_t \mid z_{1:t}, u_{1:t}) \)
- \( \overline{bel}(x_t) = p(x_t \mid z_{1:t-1}, u_{1:t}) \)
Example

Situation
Example
Measurement and state transition probabilities

• Characterization of the robot’s sensors:
  - \( p(Z_t = \text{senseOpen} \mid X_t = \text{isOpen}) = 0.6 \)
  - \( p(Z_t = \text{senseClosed} \mid X_t = \text{isOpen}) = 0.4 \)
  - \( p(Z_t = \text{senseOpen} \mid X_t = \text{isClosed}) = 0.2 \)
  - \( p(Z_t = \text{senseClosed} \mid X_t = \text{isClosed}) = 0.8 \)

• Characterization of the action pushOpen
  - \( p(X_t = \text{isOpen} \mid U_t = \text{push}, X_{t-1} = \text{isOpen}) = 1 \)
  - \( p(X_t = \text{isClosed} \mid U_t = \text{push}, X_{t-1} = \text{isOpen}) = 0 \)
  - \( p(X_t = \text{isOpen} \mid U_t = \text{push}, X_{t-1} = \text{isClosed}) = 0.8 \)
  - \( p(X_t = \text{isClosed} \mid U_t = \text{push}, X_{t-1} = \text{isClosed}) = 0.2 \)

• Characterization of the action doNothing
  - \( p(X_t = \text{isOpen} \mid U_t = \text{doNothing}, X_{t-1} = \text{isOpen}) = 1 \)
  - \( p(X_t = \text{isClosed} \mid U_t = \text{doNothing}, X_{t-1} = \text{isOpen}) = 0 \)
  - \( p(X_t = \text{isOpen} \mid U_t = \text{doNothing}, X_{t-1} = \text{isClosed}) = 0 \)
  - \( p(X_t = \text{isClosed} \mid U_t = \text{doNothing}, X_{t-1} = \text{isClosed}) = 1 \)
Example

The robot’s prior belief

\[ bel(X_0 = \text{open}) = bel(X_0 = \text{closed}) = 0.5 \]
Example
Belief after *doNothing and senseOpen*

- \( \overline{bel}(x_1) \)
  \[
  \begin{align*}
  \overline{bel}(x_1) &= \int p(x_1 \mid u_1, x_0) \overline{bel}(x_0) \, dx_0 \\
  &= \sum_{x_0} p(x_1 \mid u_1, x_0) \overline{bel}(x_0) \, dx_0 \\
  &= p(x_1 \mid U_1 = \text{doNothing}, X_0 = \text{isOpen}) \cdot \overline{bel}(X_0 = \text{isOpen}) \\
  &\quad + p(x_1 \mid U_1 = \text{doNothing}, X_0 = \text{isClosed}) \cdot \overline{bel}(X_0 = \text{isClosed})
  \end{align*}
  \]

- \( \overline{bel}(X_1 = \text{isOpen}) \)
  \[
  \begin{align*}
  \overline{bel}(X_1 = \text{isOpen}) &= p(X_1 = \text{isOpen} \mid U_1 = \text{doNothing}, X_0 = \text{isOpen}) \cdot \overline{bel}(X_0 = \text{isOpen}) \\
  &\quad + p(X_1 = \text{isOpen} \mid U_1 = \text{doNothing}, X_0 = \text{isClosed}) \cdot \overline{bel}(X_0 = \text{isClosed}) \\
  &= 1 \cdot 0.5 + 0 \cdot 0.5 = 0.5
  \end{align*}
  \]

- \( \overline{bel}(X_1 = \text{isClosed}) = ... = 0.5 \)
Example
Belief \(\text{bel}(x_1)\)

- \(\text{bel}(x_1) = \eta p(Z_1 = \text{senseOpen} | x_1) \overline{\text{bel}}(x_1)\)
- \(\text{bel}(X_1 = \text{isOpen})\)
  \[= \eta p(Z_1 = \text{senseOpen} | X_1 = \text{isOpen}) \overline{\text{bel}}(X_1 = \text{isOpen})\]
  \[= \eta 0.6 \times 0.5 = 0.3\eta\]
- \(\text{bel}(X_1 = \text{isClosed})\)
  \[= \eta p(Z_1 = \text{senseOpen} | X_1 = \text{isClosed}) \overline{\text{bel}}(X_1 = \text{isClosed})\]
  \[= \eta 0.2 \times 0.5 = 0.1\]
- \(\eta = (0.3 + 0.1)^{-1} = 2.5\)
- \(\text{bel}(X_1 = \text{isOpen}) = 0.75\)
- \(\text{bel}(X_1 = \text{isClosed}) = 0.25\)
Example
Belief $bel(x_1)$ for $u_2 = push$ and $z_2 = senseOpen$

- $\overline{bel}(X_2 = isOpen) = 1 \times 0.75 + 0.8 \times 0.25 = 0.95$
- $\overline{bel}(X_2 = isClosed) = 0 \times 0.75 + 0.2 \times 0.25 = 0.05$
- $\overline{bel}(X_2 = isOpen) = \eta \times 0.6 \times 0.95 \approx 0.983$
- $\overline{bel}(X_2 = isClosed) = \eta \times 0.2 \times 0.05 \approx 0.017$
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
Bayesian Filter

• Why should you care?
  – Robot and environmental state estimation is a fundamental problem!

• Many algorithms that exist for spatial reasoning make use of this approach
  – If you’re working in mobile robotics, you’ll see it over and over!
  – Very important to understand and appreciate

• Efficient state estimator
  – Recursively compute the robots current state based on the previous state of the robot
Bayesian Filter

- Estimate state $x$ from data $d$
  - What is the probability of the robot being at $x$?
- $x$ could be robot location, map information, locations of targets, etc
- $d$ could be sensor readings such as range, actions, odometry from encoders, etc.
- This is a general formalism that does not depend on the particular probability representation
- Bayes filter recursively computes the posterior distribution:
  $$\text{Bel}(X_t) = P(X_t | Z_t)$$
Bayesian Filtering: two phases

- Two phases:
  1. Prediction Phase
  2. Measurement Phase
1. Prediction Phase

\[ P(x_t) = \sum P(x_t|x_{t-1}, u) P(x_{t-1}) \]

**Motion Model**
2. Measurement Phase

\[ P(x_t|z) = k \, P(z|x_t) \, P(x_t) \]

Sensor Model
Derivation of the Bayes filter (1)

• let $d_t$ have the form $d_{t-1}, u_{t-1}, z_t$ where $d_{t-1}$

• applying Bayes rule with $d_{t-1}$ and $u_{t-1}$ as background knowledge

$$P(A \mid B, C) = \frac{P(B \mid A, C) \cdot P(A \mid C)}{P(P \mid C)}$$

$$P(x_t \mid d_{t-1}, u_{t-1}, z_t) = \frac{P(z_t \mid d_{t-1}, u_{t-1}, x_t) \cdot P(x_t \mid d_{t-1}, u_{t-1})}{P(z_t \mid d_{t-1}, u_{t-1})}.$$  \hfill (1)

• applying the Markov assumption:

$$P(z_t \mid d_{t-1}, u_{t-1}, x_t) = P(z_t \mid x_t)$$

$$P(x_t \mid d_{t-1}, u_{t-1}, z_t) = \frac{P(z_t \mid x_t) \cdot P(x_t \mid d_{t-1}, u_{t-1})}{P(z_t \mid d_{t-1}, u_{t-1})}.$$  \hfill (2)
Derivation of the Bayes filter (2)

- the $P(z_t \mid d_{t-1}, u_{t-1})$ does not depend on the robot’s pose
  replace it by a normalization constant $\alpha_t$:

\[
P(x_t \mid d_{t-1}, u_{t-1}, z_t) = \alpha_t P(z_t \mid x_t) P(x_t \mid d_{t-1}, u_{t-1}).
\]
Derivation of the Bayes filter (3)

- consider $P(x_t \mid d_{t-1}, u_{t-1})$
  
  If we use the law of total probability we derive

  $$P(x_t \mid d_{t-1}, u_{t-1}) = \sum_{x_{t-1} \in X} P(x_t \mid d_{t-1}, u_{t-1}, x_{t-1})P(x_{t-1} \mid d_{t-1}, u_{t-1}). \quad (4)$$

- expand $P(x_t \mid d_{t-1}, u_{t-1})$

  $$P(x_t \mid d_{t-1}, u_{t-1}, x_{t-1}) = P(x_t \mid u_0, z_1, \ldots, z_{t-1}, u_{t-1}, x_{t-1}) \quad (5)$$

- apply Markov assumption:

  $$P(x_t \mid d_{t-1}, u_{t-1}, x_{t-1}) = P(x_t \mid u_{t-1}, x_{t-1}) \quad (6)$$
Derivation of the Bayes filter (4)

• simplify the right-hand side of Equation 4

\[
P(x_t | d_{t-1}, u_{t-1}) = \sum_{x_{t-1} \in X} P(x_t | u_{t-1}, x_{t-1}) P(x_{t-1} | d_{t-1}, u_{t-1}).
\]  

(7)

• \(u_{t-1}\) causes the transition from \(x_{t-1}\) to \(x_t\)  
  thus, \(x_{t-1}\) is independent of \(u_{t-1}\)

\[
P(x_{t-1} | d_{t-1}, u_{t-1}) = P(x_{t-1} | d_{t-1})
\]  

(8)

• Substituting Equation 8 into Equation 7 leads to

\[
P(x_t | d_{t-1}, u_{t-1}) = \sum_{x_{t-1} \in X} P(x_t | u_{t-1}, x_{t-1}) P(x_{t-1} | u_{t-1})
\]  

(9)
Derivation of the Bayes filter (5)

If we now substitute this equation into Equation 3 we obtain

$$P(x_t \mid d_t) = \alpha \, P(z_t \mid x_t) \sum_{x'_{t-1} \in X} P(x_t \mid u_{t-1}, x'_{t-1}) \, P(x'_{t-1} \mid d_{t-1}).$$  \hspace{1cm} (10)
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
Localization: the problem
Localization: the basic problem
Demonstrations of Self Localization

- Global localization
- Adaptive global localization
- Kidnapped robot problem
- Localization with adaptive sample size
Probabilistic Localization through recursive filtering

- \( x = \text{state, ie } \text{pose} \) given as its position and orientation
- \( P(x_t) = \text{robot’s probabilistic belief about its pose } x_t \) at time \( t \)
- \( z_t = \text{sensor measurements} \)
- \( u_t = \text{odometry measurements} \)
- \( d_t = d_t = u_0, z_1, \ldots, u_{t-1}, z_t \)
- \( P(x_t \mid d_t) = \text{posterior probability: conditional probability over all possible poses given all information gathered so far.} \)

\[
P(x_t \mid d_t) = P(x_t \mid u_0, z_1, \ldots, u_{t-1}, z_t) \\
= \alpha P(z_t \mid x_t) \sum_{x'_{t-1} \in X} P(x_t \mid u_{t-1}, x'_{t-1}) P(x'_{t-1} \mid u_0, z_1, \ldots, u_{t-2}, z_{t-1}) \\
= \alpha P(z_t \mid x_t) \sum_{x'_{t-1} \in X} P(x_t \mid u_{t-1}, x'_{t-1}) P(x'_{t-1} \mid d_{t-1}) \tag{11}
\]
Probabilistic Localization through recursive filtering (2)

\[ P(x_t \mid d_t) = P(x_t \mid u_0, z_1, \ldots, u_{t-1}, z_t) \]
\[ = \alpha P(z_t \mid x_t) \sum_{x'_{t-1} \in X} P(x_t \mid u_{t-1}, x'_{t-1}) P(x'_{t-1} \mid u_0, z_1, \ldots, u_{t-2}, z_{t-1}) \]
\[ = \alpha P(z_t \mid x_t) \sum_{x'_{t-1} \in X} P(x_t \mid u_{t-1}, x'_{t-1}) P(x'_{t-1} \mid d_{t-1}) \]  

(12)

- \(P(x_t \mid d_t)\) is the *posterior*
- \(P(z_t \mid x_t)\) = *observation model* which specifies the likelihood of the measurement \(z_t\) given the robot is at location \(x_t\)
- \(P(x_t \mid u_{t-1}, x'_{t-1})\) = *motion model* aka transition probability. likelihood that the movement action \(u_{t-1}\) carried out at location \(x'_{t-1}\) carries the robot to the location \(x_t\)
- \(P(x'_{t-1} \mid d_{t-1})\) = *prior* as it quantifies the probability that the robot is at location \(x'_{t-1}\) before the integration of \(z_t\) and \(u_{t-1}\)
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
Open points

1. how the robot’s belief $P(x)$ is represented and
2. how the conditional probabilities
   $P(x_t | u_{t-1}, x'_{t-1})$ and
   $P(z_t | x_t)$ are computed.
State Space Representations

Kalman filter

Multi-hypothesis

Histograms (metric, topological)

Particles

References:
- [Schiele et al. 94], [Weiβ et al. 94], [Borenstein 96],
  [Gutmann et al. 96, 98], [Arras 98]
- [Weckesser et al. 98], [Jensfelt et al. 99]
- [Nourbakhsh et al. 95], [Simmons et al. 95], [Kaelbling et al. 96],
  [Burgard et al. 96], [Konolige et al. 99]
- [Kanazawa et al 95] [de Freitas 98]
  [Isard/Blake 98] [Doucet 98]
Kalman Filter: Expected Value & Variance of the State

- The *expected value* for a random variable $X$ is \( (i.e. \text{ the mean}) \) defined as

\[
\mu = E(X) = \sum_{i=1}^{n} p_i x_i \quad \text{for discrete } X
\]

\[
\mu = E(X) = \int_{-\infty}^{\infty} x p_X(x) \, dx \quad \text{for continuous } X
\]

- The *variance* of $X$ about the mean is defined as

\[
\sigma^2 = E[(x-\mu)^2] = \sum_{i=1}^{n} p_i (x_i - \mu)^2 \quad \text{for discrete } X
\]

\[
\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p_X(x) \, dx \quad \text{for continuous } X
\]
Kalman Filter: The Gaussian Distribution

- A 1-D Gaussian distribution is defined as

\[ p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

- In 2-D (assuming uncorrelated variables) this becomes

\[ p(\vec{x}) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{1}{2} \left( \frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} \right)} \]

- In n dimensions, it generalizes to

\[ p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n|C|}} e^{-\frac{1}{2}(x-\mu)^T C^{-1} (x-\mu)} \]

The Normal (Gaussian) distribution is completely parameterized by its first and second moments.
Kalman Filter: What is Covariance?

• When $X$ is a vector, the variance is expressed in terms of a covariance matrix $C$ where

\[ c_{ij} = E[(\mathbf{x}_i - \mathbf{\mu}_i)^T (\mathbf{x}_j - \mathbf{\mu}_j)] \]

• The resulting matrix has the form

\[
C = \begin{bmatrix}
\sigma_1^2 & \rho_{12} \sigma_1 \sigma_2 & \cdots & \rho_{1n} \sigma_1 \sigma_n \\
\rho_{12} \sigma_1 \sigma_2 & \sigma_2^2 & \cdots & \rho_{2n} \sigma_2 \sigma_n \\
\vdots & \vdots & \ddots & \vdots \\
\rho_{1n} \sigma_1 \sigma_n & \rho_{2n} \sigma_2 \sigma_n & \cdots & \sigma_n^2
\end{bmatrix}
\]

where $\rho_{ij}$ corresponds to the degree of correlation between variables $X_i$ and $X_j$. 
Kalman Filter: Example

- **QUESTION:** Given a normal distribution with covariance matrix

\[
C = \begin{bmatrix}
5.4729 & 3.9719 \\
3.9719 & 4.5271
\end{bmatrix}
\]

characterize the PDF

- **SOLUTION:** Solving for the eigenvalues, we get

\[
(\lambda - 5.47)(\lambda - 4.53) - 3.97^2 = 0
\]

\[\Rightarrow \lambda_1 = 9, \lambda_2 = 1 \Rightarrow \sigma_1^2 = 9, \sigma_2^2 = 1\]

and solving for the first EVE we obtain

\[
x_1 = \begin{bmatrix}
0.747 \\
0.664
\end{bmatrix}
\Rightarrow \tan^{-1} \left( \frac{0.664}{0.747} \right) = 41.6^\circ
\]
Kalman Filter: Algorithm

\[(x_0^-, P_0^-)\]

**Time Update**

1. Project the state forward
   \[x_{k+1}^- = A \hat{x}_k + B \hat{u}_k\]

2. Project the covariance forward
   \[P_{k+1}^- = AP_k A^T + Q\]

**Measurement Update**

1. Compute Kalman Gain
   \[K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}\]

2. Update state estimate with measurement \(z_k\)
   \[x_k = x_k^- + K (z_k - H x_k^-)\]

3. Update error covariance
   \[P_k = (I - K_k H) P_k^-\]
Discrete Approximations

- discretize the environment into a grid of cells
typical size: 10-20cm (indoor)
- the probability distribution for the robot’s position is represented by a set of random variables $c_{x,y}$
representing the event that the robot is located in the grid cell $c_{x,y}$
- $P(c_{x,y})$ then is the belief of the robot being in the grid cell $c_{x,y}$
Probabilistic localization with discrete state space

for each state $x$ do
    initialize $P(x)$
forever do
    if new sensory input $z_t$ is received do
        $\alpha_t = 0$
        for each state $x$ do
            /* apply the perception model */
            $P(x) = P(z_t \mid x) \cdot P(x)$
            $\alpha_t = \alpha_t + P(x)$
        for each state $x$ do
            /* normalize the belief */
            $P(x) = \alpha_t^{-1} \cdot P(x)$
    if an odometry reading $u_t$ is received do
        for each state $x$ do
            /* apply the motion model */
            $P'(x) = \sum_{x' \in X} P(x \mid x', u_t) \cdot P(x)$
            $P(x) = P'(x)$
Occupancy grid map and data set
Density plots ...

after incorporating 5, 18, and 24 sonar scans (darker positions are more likely)
Particle Filter

• key idea:
  – represent the posterior be a set $M$ of $N$ samples
• sample $m = \langle x, \omega \rangle$
  – $x =$ state vector
  – $\omega =$ weighing factor
    $\omega$ used to store the importance of the corresponding particle
• posterior $=$ distribution of samples and their importance factors.
Particle Filter: 1. Prediction Phase

\[ P(x_t | \bullet, u) \]

Motion Model
Particle Filter: 2. Measurement Phase

\[ P(z|x_t) \]

Sensor Model
Particle Filter: 3. Resampling

O(N)
Particle Localization: the example revisited

Basics  Bayesian Filtering  Self Localization  State  Models  RoboCup
Particle Filter-based Localization (1)
Particle Filter-based Localization (2)

\[ p(o_t | s_t, m) \]
Particle Filter-based Localization (3)

\[ b(s_{t+1}) \leftarrow \int p(s_{t+1} | a_t, s_t, m) b(s_t) \, ds_t \]
Particle Filter-based Localization (4)

\[ b(s_{t+1}) \leftarrow \eta \ p(o_{t+1} | s_{t+1}, m) \ b(s_{t+1}) \]
Sequential importance sampling algorithm

Initialize $M$ according to $Bel(X_0 = x)$

forever do
  if new sensory input $z_t$ is received do
    $\alpha = 0$
    for each sample $\langle x_i, \omega_i \rangle \in M$ do
      $w_i = P(z_t \mid x_i)$
      $\alpha = \alpha + w_i$
    end for

  foreach sample $\langle x_i, \omega_i \rangle \in M$ do
    $w_i = \alpha^{-1} \cdot w_i$
  end foreach

  $M = \text{resample}(M)$

if an odometry reading $u_t$ is received do
  foreach sample $\langle x_i, \omega_i \rangle \in M$ do
    compute a new state $x'$ by sampling from $P(x' \mid x_i, u_t)$.
    $x_i = x'$
  end foreach
end if
Let $M$ consist of $N$ samples $\langle x_i, \omega_i \rangle$ and $M'$ consist of $N$ samples $\langle x'_i, \omega'_i \rangle$.

$$\Delta = \text{rand}(]0; N^{-1}])$$ /* draw random number from $]0; N^{-1}] */

$c = \omega_0$

$i = 0$

for $j = 0$ to $N - 1$ do

$u = \Delta + j \cdot N^{-1}$

while $u > c$ do

$i = i + 1$

$c = c + \omega_i$

$\langle x'_j, \omega'_j \rangle = \langle x_i, \omega_i \rangle$

return $M'$
Outline

Basic Concepts
Bayesian Filtering
Self Localization
State Representations

Sensor and Motion Models
State Estimation in RoboCup
Sensor and Motion Models

Motion Models

Sensor Models
Motion Models

Parameters $\alpha, \beta$, and $d$ specifying any incremental motion of a robot in the $(x, y, \theta)$-space.

Sample-based approximation of the belief of the robot after a repeatedly executing motion commands.
Sensor Models

Ultrasound scan perceived with a B21 robot in an office environment

Laser scan perceived in a populated museum
Beam-based Sensor Models

- A scan consists of $k$ measurements
  \[ Z = \{z_1, ..., z_k\} \]
- Individual measurements are independent given the robot position
  \[ P(z \mid x, m) = \sum_{k=1}^{K} P(z_k \mid x, m) \]
Uncertainty in proximity measurements

• Measurements can be caused by:
  – A known obstacle
  – Cross talk
  – An unexpected obstacle (persons, chairs, ...)
  – Missing all obstacles (reflection, glass, ...)

• Noise is due to uncertainty
  – In measuring distance to known obstacle
  – In position of known obstacle
  – In position of additional obstacles
  – Whether obstacle is missed
Components of the beam-based sensor models

Measurement noise

\[ P_{hit}(z \mid x, m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2b}(z-z_{\text{exp}})^2} \]

Unexpected obstacles

\[ P_{\text{unexp}}(z \mid x, m) = \begin{cases} \eta \lambda e^{-\lambda z} & z < z_{\text{exp}} \\ 0 & \text{otherwise} \end{cases} \]
Mixture model for beam-based sensors
Sensor Models: from Data sets

Sonar (distance 1)

Sonar (distance 2)

Laser (distance 1)

Laser (distance 2)
Approximation results

Measured and approximated probability of sonar (a,b) and laser (c,d) measurements:
Outline

Basic Concepts

Bayesian Filtering

Self Localization

State Representations

Sensor and Motion Models

State Estimation in RoboCup
Vision-based Cooperative State Estimation in Robot Soccer

joint work with: Thorsten Schmitt and Robert Hanek

AAAI 2002, AAMAS 2002
Transactions of Robotics and Automation 2003
Autonomous Robots 2004
Game State Estimation in Robot Soccer

Computational Problem:
Game State Estimation in Robot Soccer

Computational Problem:
Game State Estimation in Robot Soccer

Computational Problem:

- Basics
- Bayesian Filtering
- Self Localization
- State
- Models
- RoboCup
Game State Estimation in Robot Soccer

Computational Problem:
State Estimation in Dynamical Systems

Environment State:
Robot 1, ..., Robot 4, Ball, Opponent 1, ..., Opponent n

with:
Robot i = ⟨x_i, y_i, θ_i, ˙x_i, ˙y_i, ˙θ_i⟩

Belief State:
Bel(s_t) = p(s_t | d_t, ..., d_0; m)

Basics
Bayesian Filtering
Self Localization
State Models
RoboCup
State Estimation in Dynamical Systems

- Environment State:
  $Robot^1, \ldots, Robot^4, Ball, Opponent^1, \ldots, Opponent^n$
  with: $Robot^i = \langle x^i, y^i, \theta^i, \dot{x}^i, \dot{y}^i, \dot{\theta}^i \rangle$
**State Estimation in Dynamical Systems**

- **Environment State:**
  \( Robot^1, \ldots, Robot^4, Ball, Opponent^1, \ldots, Opponent^n \)
  with: \( Robot^i = \langle x^i, y^i, \theta^i, \dot{x}^i, \dot{y}^i, \dot{\theta}^i \rangle \)

- **Belief State:**
  \( Bel_t(s_t) = p(s_t | d_t, \ldots, d_0; m) \)
The AGILO State Estimation Module (Version 1)

Perception Module: VISBOT 1

Sensor-data Processing

Own Sensor-data
- Odometry
- Camera Image
- Environment Feature Detector
- Environment Features
- Ball Feature Detector
- Ball Features
- Opponents Feature Detector
- Opponents Features

State Estimators of Teammates
- Ball State
- Opponents State
- Team Mate State

State Estimators
- Self Localization (incremental)
- Ball Localization (incremental)
- Opponents Localization (MHT)

Belief State
- Own State
- Ball State
- Opponents State
- Team Mate State

Basics  Bayesian Filtering  Self Localization  State  Models
RobbyCup
The AGILO State Estimation Module (Version 2)

Perception Module: VISBOT 2

Sensor–data Processing

Own Sensor–data
- Odometry
- Camera Image
- Environment Features
- CCD Tracker
- Opponents Feature Detector

State Estimators of Teammates
- Ball State
- Opponents State
- Team Mate State

State Estimators
- Self Localization
  - incremental Kalman–filter–based Monitor
  - active Self–localization Particle–filter–based
- Ball Localization
  - increm. Monitor
  - Particle–filter–based Ball–local.
- Opponents Localization
  - learned Observation filtering
  - learned Perception model

Belief State
- Own State
- Ball State
- Opponents State
- Situation Class
- Team Mate State

Basics  Bayesian Filtering  Self Localization  State  Models

Michael Beetz
Summer Term 2012
Color Clustering

Color Clustering

Last reg. : 30 - 54
undefined : 128 - 127
red      : 40 - 210
yellow  : 111 - 189
green    : 40 - 130
cyan     : 108 - 194
blue     : 23 - 127
magenta : 128 - 127
black    : 30 - 127
gray     : 128 - 127
white    : 128 - 127

Select a "Color" to learn, "Next Image" or "Quit"!
Color Segmentation
AGILO Field Model
Model Projection and Matching

Projection of 3D feature into the image (depends on last pose $\Phi$).

Color change in image

(white)

(green)
Curve Fitting in Images: The Contracting Curve Density Algorithm

joint work with: Robert Hanek

IJCV 2004
RoboCup Engineering Challenge Award
Best Paper DAGM 2001
Image Segmentation

edge-based

region-based

graph-based
Image Segmentation

Our Approach:
Acquire, use, and update models of the region and the background
Curve Fitting: the computational problem

• Given:

  ![Image](image)
  ![Blurred Curve Model](blurred_curve_model)

• Compute:

  \[ \arg\max_{\Phi} p(I \mid \Phi)p(\Phi) \]
Curve Fitting: the algorithm

**Input:** image data $I^*$, curve function $c$, mean $m_\phi^*$ and covariance $\Sigma_\phi^*$

**Output:** estimate $m_\phi$ of model parameters and covariance $\Sigma_\phi$

**Initialization:** mean $m_\phi = m_\phi^*$, covariance $\Sigma_\phi = c_1 \cdot \Sigma_\phi^*$

repeat

1. **learn local statistics** of image data from the vicinity of the curve
   1.1 determine pixels $v$ in vicinity $V$ of curve from $c$, $m_\phi$ and $\Sigma_\phi$
   1.2 $\forall v \in V$ compute local statistics $S_v$ of image data $I^*_V$

   which characterize the two sides of the curve

2. **refine estimate** of model parameter vector
   2.1 update mean $m_\phi$ through MAP estimation:
      
      $$m_\phi = \arg \min_{m_\phi} \chi^2(m_\phi)$$

   2.2 updated covariance $\Sigma_\phi$ from the Hessian of $\chi^2(m_\phi)$

until changes of $m_\phi$ and $\Sigma_\phi$ are small enough

return mean $m_\phi$ and covariance $\Sigma_\phi$
Curve Fitting: Animation (1)

CCD Algorithm

repeat

1. **learn local statistics**
   1.1 determine vicinity
   1.2 compute local statistics

2. **refine parameter estimate**
   2.1 update mean
   2.2 update covariance

until accurate enough

return mean and covariance
Curve Fitting: Animation (1)

CCD Algorithm

repeat

1. learn local statistics
   1.1 determine vicinity
   1.2 compute local statistics

2. refine parameter estimate
   2.1 update mean
   2.2 update covariance

until accurate enough

return mean and covariance
Curve Fitting: Animation (2)

CCD Algorithm

repeat

1. **learn local statistics**
   1.1 determine vicinity
   1.2 compute local statistics

2. **refine parameter estimate**
   2.1 update mean
   2.2 update covariance

until accurate enough

return mean and covariance
Curve Fitting: Animation (3)

CCD Algorithm

repeat
  1. learn local statistics
     1.1 determine vicinity
     1.2 compute local statistics
  2. refine parameter estimate
     2.1 update mean
     2.2 update covariance
until accurate enough
return mean and covariance
Curve Fitting: Animation (4)

CCD Algorithm

repeat

1. learn local statistics
   1.1 determine vicinity
   1.2 compute local statistics

2. refine parameter estimate
   2.1 update mean
   2.2 update covariance

until accurate enough

return mean and covariance
Curve Fitting: Animation (5)

CCD Algorithm

repeat
  1. learn local statistics
     1.1 determine vicinity
     1.2 compute local statistics
  2. refine parameter estimate
     2.1 update mean
     2.2 update covariance
until accurate enough
return mean and covariance
Curve Fitting: Animation (6)

CCD Algorithm

repeat
  1. learn local statistics
     1.1 determine vicinity
     1.2 compute local statistics
  2. refine parameter estimate
     2.1 update mean
     2.2 update covariance
until accurate enough
return mean and covariance
Curve Fitting: Animation (7)

CCD Algorithm

repeat

1. learn local statistics
   1.1 determine vicinity
   1.2 compute local statistics

2. refine parameter estimate
   2.1 update mean
   2.2 update covariance

until accurate enough

return mean and covariance
Curve Fitting: Animation (8)

CCD Algorithm

repeat
1. learn local statistics
   1.1 determine vicinity
   1.2 compute local statistics
2. refine parameter estimate
   2.1 update mean
   2.2 update covariance
until accurate enough
return mean and covariance
Curve Fitting: Local Statistics

<table>
<thead>
<tr>
<th>iteration:</th>
<th>0</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.) Input image with superimposed curve</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image1.png" alt="Input Image" /></td>
<td><img src="image2.png" alt="Input Image" /></td>
<td><img src="image3.png" alt="Input Image" /></td>
<td></td>
</tr>
<tr>
<td>b.) Expected image according to the local statistics</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image4.png" alt="Expected Image" /></td>
<td><img src="image5.png" alt="Expected Image" /></td>
<td><img src="image6.png" alt="Expected Image" /></td>
<td></td>
</tr>
</tbody>
</table>
CCD Tracking: Results and Next Steps

Results:

- CCD tracking
- Condensation tracking

Next Steps:

- CCD-based tracking of human motions
- Recognizing complex activities
- Application: Basics Bayesian Filtering Self Localization State Models RoboCup
CCD Tracking: Results and Next Steps

Results:
- CCD tracking
- Condensation tracking

Next Steps:
- CCD-based tracking of human motions
- Recognizing complex activities
- Application:
Opponent Tracking

algorithm **Bayes Filter** $(Bel(x_{t-1}), data)$

1. **switch** $(data)$
2. **case** $(data$ is a control signal vector $u)$:
   3. $Bel(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) Bel(x_{t-1}) \, dx_{t-1}$
4. **case** $(data$ is a sensor data vector $z)$:
   5. $Bel(x_t) = \nu \, p(z_t|x_t) \, Bel(x_{t-1})$
Opponent Tracking

algorithm Bayes Filter $(Bel(x_{t-1}), data)$

1. $switch (data)$
2. $case (data$ is a control signal vector $u)$:
3. $Bel(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) \cdot Bel(x_{t-1}) \; dx_{t-1}$
4. $case (data$ is a sensor data vector $z)$:
5. $Bel(x_t) = \nu \cdot p(z_t|x_t) \cdot Bel(x_{t-1})$
Opponent Tracking

algorithm **Bayes Filter** \((\text{Bel}(x_{t-1}), \text{data})\)

1. **switch** \((\text{data})\)
2. **case** \((\text{data} \text{ is a control signal vector } u)\):  
3. \[ \text{Bel}(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) \text{Bel}(x_{t-1}) \, dx_{t-1} \]
4. **case** \((\text{data} \text{ is a sensor data vector } z)\):  
5. \[ \text{Bel}(x_t) = \nu \, p(z_t|x_t) \text{Bel}(x_{t-1}) \]
Opponent Tracking

**Algorithm** Bayes Filter \((Bel(x_{t-1}), data)\)

1. switch \((data)\)
2. case \((data\ is\ a\ control\ signal\ vector\ u)\) :
3. \(Bel(x_t) = \int p(x_t|x_{t-1}, u_{t-1}) \ Bel(x_{t-1}) \ d \ x_{t-1}\)
4. case \((data\ is\ a\ sensor\ data\ vector\ z)\) :
5. \(Bel(x_t) = \nu \ p(z_t|x_t) \ Bel(x_{t-1})\)

- represent \(Bel(x_t)\) as a set of weighted hypotheses:
  \(H_t = \bigcup_{i=1}^{n} \{h_t^i, p_t^i\}\)
- Belief is a mixture of Gaussians
  \(Bel(x_t) = \sum_{i=1}^{n} p_t^i \ast N(x_t; \bar{h}_t^i, \Sigma_h^i)\)
- compute \(p_t^i\) from \(P(h_t^i|Z_{t...0})\)
  - the form of \(P(h_t^i|Z_{t...0})\) is highly task specific with parameters:
    - termination likelihood \(\lambda_T = 2s\)
    - false observations \(\lambda_F = 0.0002\)
    - new tracks \(\lambda_N = 0.04\)
    - detection probability \(P_D = 0.9\)
    - ...

---

Michael Beetz  
Summer Term 2012  
Technical Cognitive Systems
The Multi Hypotheses Tracking Algorithm for Opponent Localization

- Hypotheses are organized in trees
- Every node represents a hypothesis
  - (mean + covariance + probability)
The Multi Hypotheses Tracking Algorithm for Opponent Localization

• Hypotheses are organized in trees
• Every node represents a hypothesis
  – (mean + covariance + probability)
The Multi Hypotheses Tracking Algorithm
for Opponent Localization

- Hypotheses are organized in trees
- Every node represents a hypothesis
  - (mean + covariance + probability)
The Multi Hypotheses Tracking Algorithm for Opponent Localization

```
algorithm MULTIPLEHYPOTHESISTRACKING()
  1  for k ← 1 to ∞
  2    do Z(k) ← INTERPRETSENSORDATA();
  3      ˜H^k ← APPLYMOTIONMODEL(H^{k-1}, M);
  4    for i ← 1 to n_k
  5      do for j ← 1 to m_k
  6          do h_{ij}^k ← ASSOCIATE(˜h_j^k, z_i(k));
  7                  COMPUTE(P(h_{ij}^k | Z(k)));
  8    for j ← 1 to n_k
  9      do H^k ← H^k ∪ {GENNEWHYP(z_j(k))};
 10     PRUNEHYPOTHESES(H^{k-N});
 11    X^{k-N} ← {h_1^{k-N}, ..., h_\delta_{k-N}};
```
Assessing Hypotheses

\[ P(h_i^k | Z^k) = P(\theta(k), h_{i-1}^k | Z(k), Z_{k-1}) \]  
\[ = P(\theta(k), h_{i-1}^k | Z(k), H^k) \]  
\[ = \alpha \ast p(Z(k) | \theta(k), h_{j-1}^k, Z_{k-1}) \]

\[ P(\theta(k) | h_{j-1}^k, Z_{k-1}) P(h_{j-1}^k | Z_{k-1}) \]
Learning Perception Models

- Classes: use or discard observation
- Feature Language:
  \[ F = \{ R, T, V, W, A, D, U \} \]
  - \( R \) = observing Robot
  - \( T \) = observation type
    (unknown, opponent)
  - \( V \) = translational velocity
  - \( W \) = rotational velocity
  - \( A \) = observational angle
  - \( D \) = distance
  - \( U \) = area of uncertainty
- Training examples from off-line MHT run with tree depth \( N = 20 \)
- Learned rules:
  - if \( W < 41 \) then use observation
  - if \( W > 41 \land D > 2.4 \) then use observation
  - if \( W > 41 \land D < 2.4 \) then discard observation
- other features irrelevant
More Informative Perception Models

- belief state:
  \[ H_t = \bigcup_{i=1}^n \{ h_t^i, p_t^i \} = \bigcup_{i=1}^n \{ \langle \bar{h}_t^i, \Sigma_{h_t^i} \rangle, p_t^i \} \]
  with \( h_t^i \sim N(\bar{h}_t^i, \Sigma_{h_t^i}) \)

- perception model:
  A hypothesis \( h \) is in \( H_t \) if
  - \( h \) corresponds to a robot and has not been pruned
  - \( h \) resulted from a hallucination

estimate expected accuracy

- simulator
- conditional probability distributions in estimation algorithm
State Estimation

What?

Input

Output

Why?

• cooperative
• vision-based

state estimation

How?

Results

estimated game state
State Estimation

**Algorithm**  
`SELFLOCALIZATION()`

```plaintext
let (\(\tilde{\Phi}, \tilde{C}\Phi\)), (\(\hat{\Phi}, \hat{C}\Phi\)), (\(\Phi^l, C^l\Phi\)), projections, corr

for all data
do (\(\tilde{\Phi}, \tilde{C}\Phi\)) ← APPLYMOTIONMODEL(\(\Phi^l, C^l\Phi\)); (\(\hat{\Phi}, \hat{C}\Phi\)) ← (\(\tilde{\Phi}, \tilde{C}\Phi\));
loop

switch (data)

case (data is an image):
  projections ← PROJECTCURVEPOINTS(\(\hat{\Phi}\));
  corr ← SEARCHCORRESP(projections, data);
  \(\chi^2_i\) ← QUALITYOFFITIMAGE(corr, \(\Phi\));

case (data is odometric data):
  \(\chi^2_o\) ← QUALITYOFFITOMOD(data, \(\Phi\));

case (data is ball observation from team-mate):
  \(\chi^2_t\) ← QUALITYOFFITTEAM(data, \(\Phi\));
  \(\chi^2\) ← \(\chi^2_i + \chi^2_t\);
  \(\hat{\Phi}\) ← \(\text{arg min}_{\Phi} \chi^2\); % one Newton step.

until (change of \(\hat{\Phi}\) is small or data is not an image)

\(\hat{C}\Phi = 2/\text{Hessian of } \chi^2\);
(\(\Phi^l, C^l\Phi\)) ← (\(\hat{\Phi}, \hat{C}\Phi\));
```

---

Michael Beetz  
Summer Term 2012  
Technical Cognitive Systems  
RoboCup
Detecting Opponents

algorithm INTERPRETSENSORDATA($\hat{\Phi}, \hat{C}_\phi$)
1 $I \leftarrow$ GetSensorData();
2 $R \leftarrow$ EXTRACTBLACKREGIONS($I$);
3 $R \leftarrow$ CHECKCONSTRAINTS($R$);
4 $R \leftarrow$ EXTRACTCASCADEDROBOTS($R$);
5 for $i \leftarrow 1$ to $|R|$ do
6   $(row, col, width) \leftarrow$ EXTRACTFEATURES($r_i$);
7   $\omega \leftarrow \left[ \hat{\Phi}, row, col, width \right]^T$;
8   $C_\omega \leftarrow \begin{pmatrix}
\hat{C}_\phi & 0 & 0 & 0 & 0 \\
0 & \sigma_{row} & 0 & 0 & 0 \\
0 & 0 & \sigma_{col} & 0 & 0 \\
0 & 0 & 0 & \sigma_{width} & 0
\end{pmatrix}$;
9  $z_i(k) \leftarrow$ UNSCENTEDTRANSFORM($\omega, C_\omega, opp$);
10 return ($Z(k)$);
MHT Algorithm

algorithm MultipleHypothesisTracking()
1 for k ← 1 to ∞
2 do Z(k) ← INTERPRETSENSORDATA();
3 \(\tilde{H}^k\) ← APPLYMOTIONMODEL(\(H^{k-1}\), \(M\));
4 for i ← 1 to \(n_k\)
5 do for j ← 1 to \(m_k\)
6 \(\text{do } h_{ij}^k \leftarrow \text{ASSOCIATE}(\tilde{h}_{ij}^k, z_i(k));\)
7 \(\text{compute } P(h_{ij}^k|Z(k));\)
8 for j ← 1 to \(n_k\)
9 do \(H^k \leftarrow H^k \cup \{\text{GENERATENEWHYP}(z_j(k))\};\)
10 \(\text{PRUNEHYPOTHESES}(H^{k-N});\)
11 \(X^{k-N} \leftarrow \{h_{1-N}^k, \ldots, h_{o_k-N}^k\};\)