

Particle filters for tracking

March 2004

MARCH 2004

What is tracking ?

Visual tracking = recursive localisation of moving objects in video sequences.

use info from previous frame(s) to determine the position of the object in the current frame.

Overview

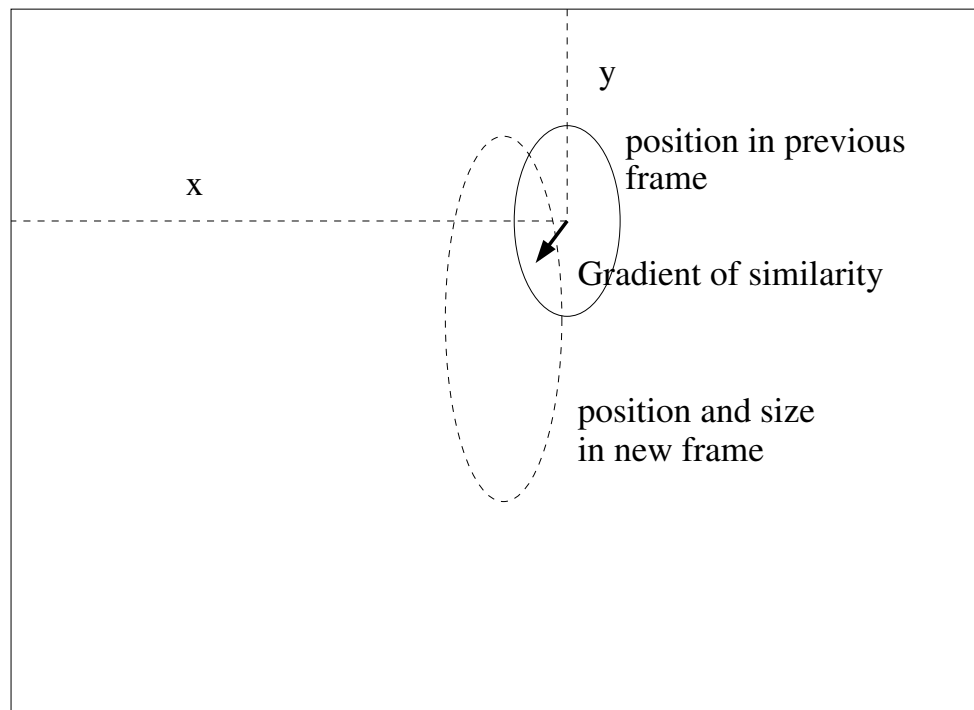
1. An intuitive example: tracking using colour
2. Formalisation
 - State-space models
 - Bayesian Recursive estimation/filtering (BRF)
 - Particle solution to BRF
3. Again the same example
4. Summary

Example: tracking using colour

- **Target model** to track described by its colour distribution $q(u)$ inside an ellipse
- Target position \vec{x} known in previous frame
- At point \vec{x} , $p_{\vec{x}}$ is the **measured** colour distribution in the current frame
- Similarity between the 2 distr. $\rho_{\vec{x}}(p, q) = \sum_u \sqrt{p_{\vec{x}}(u)q(u)}$

Example: tracking using colour

- **Traditional approach** (Lucas-Kanade style): compute ρ gradient and move in this direction until ρ does not change.



Example: tracking using colour II



- **Instead:** use N ellipses $\mathbf{x}^{(i)}$, $i = 1, \dots, N$ ($= N$ particles) and measure $\rho^{(i)} = \rho_{\vec{x}(\mathbf{x}^{(i)})}$

Example: tracking using colour III

- Particles: $\mathbf{x} = (x, y, \dot{x}, \dot{y}, H_x, H_y, \dot{a})^T$
- Dynamical model for \mathbf{x}_t (for frame t)

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$$

where A is (7 x 7) matrix, \mathbf{w}_{t-1} Gaussian 7-dim. noise vector. Gives a **prediction** on \mathbf{x} for next frame.

$$\begin{pmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \\ H_{xt} \\ H_{yt} \\ \dot{a}_t \end{pmatrix} = \begin{pmatrix} 1 & 0 & \Delta & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \Delta \\ 0 & 0 & 0 & 0 & 0 & 1 & \Delta \end{pmatrix} \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ H_{xt-1} \\ H_{yt-1} \\ \dot{a}_{t-1} \end{pmatrix} + \begin{pmatrix} w_{1t-1} \\ w_{2t-1} \\ w_{3t-1} \\ w_{4t-1} \\ w_{5t-1} \\ w_{6t-1} \\ w_{7t-1} \end{pmatrix}$$

Example: tracking using colour IV

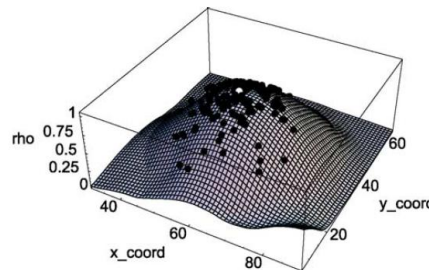
After prediction, **weight** the particle $\mathbf{x}_t^{(i)}$ with

$$\pi_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(1 - \rho^{(i)})^2}{2\sigma^2}\right)$$

i.e. the particles close to target are **reinforced**, particles far are **diminished** ($\rho \approx 1$ for similar colour distrib.)

The mean of the weighted particles gives the target position in current frame t

$$E[\mathbf{x}_t] = \sum_i \pi^{(i)} \mathbf{x}_t^{(i)}$$



Example: tracking using colour V

- Particles with small probabilities tend to disappear, they will however survive for some frames.
- Not only the “most probable” hypothesis is kept
- The tracker is able to maintain **multi-hypotheses** at the same time, but this is done naturally through the model.

Example: tracking using colour VI

Summary: One step of the PF algorithm

- From

1. the particles from previous frame $\mathbf{x}_{t-1}^{(i)}$
2. System equation $\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$

Predict next positions of particles $\mathbf{x}_t^{(i)}$

Note: need to draw from \mathbf{w}_{t-1} otherwise particles evolve identically

- Then **Weight** the particle $\mathbf{x}_t^{(i)}$ with $\pi_t^{(i)} \propto \exp(-(\rho^{(i)} - 1)^2 / 2\sigma^2)$.
- **Estimate** the mean state $E[\mathbf{x}_t] = \sum_i \pi^{(i)} \mathbf{x}_t^{(i)}$
- **Resample**: particles with big weight generate many particles (Prepare N particles for next frame).

Overview

1. An intuitive example: tracking using colour
2. **Formalisation**
 - State-space models
 - Bayesian Recursive estimation/filtering (BRF)
 - Particle solution to BRF
3. Again the same example
4. Summary

Formalisation: state-space model

Tracking can be written in the Bayesian Recursive estimation/filtering framework

The model:

- State vector \mathbf{x}_t contains (internal) info about the system at frame t
- Observation vector \mathbf{z}_t contains features of interest extracted from frame t
- State evolution $\mathbf{x}_t = \mathbf{f}_t(\mathbf{x}_{t-1}) + \mathbf{w}_{t-1}$
- Observation/measurement model: relates the state to the noisy measurement $\mathbf{z}_t = \mathbf{h}_t(\mathbf{x}_t) + \mathbf{n}_t$
- state evolution $\Leftrightarrow p(\mathbf{x}_t | \mathbf{x}_{t-1})$
- observ. model $\Leftrightarrow p(\mathbf{z}_t | \mathbf{x}_t)$

Our purpose

- Find $p(\mathbf{x}_t | \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_t) = p(\mathbf{x}_t | \mathbf{z}_{1:t})$
- $p(\mathbf{x}_t | \mathbf{z}_{1:t})$ give estimates (MAP, MMSE) of state and accuracy.
e.g.

$$\bar{\mathbf{x}}_t = E[\mathbf{x}_t] = \int p(\mathbf{x}_t | \mathbf{z}_{1:t}) d\mathbf{x}_t$$

Bayesian Recursive Filtering

Under hypotheses that observations are mutually independent and \mathbf{x}_t is a markov process, solution to our problem is given by

- Prediction step:

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t}) = \int p(\mathbf{x}_{t+1}|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{z}_{1:t})d\mathbf{x}_t$$

- Update step: measurement \mathbf{z}_t becomes available

$$p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t+1}) = k_t p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})p(\mathbf{x}_{t+1}|\mathbf{z}_{1:t})$$

The repetition of the two steps **propagates** $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ in time

Particle filter

- Particle filters give a numerical solution to BRF
- $p(\mathbf{x}_t | \mathbf{z}_{1:t})$ is represented by N particles

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}) \approx 1/N \sum_i \delta(\mathbf{x}_t - \mathbf{x}_t^{(i)})$$

- **Prediction step**: apply the state evolution equation to each particle $\mathbf{x}_t^{(i)}$
 $\mathbf{x}_t^{(i)} = \mathbf{f}_t(\mathbf{x}_{t-1}^{(i)}) + \mathbf{w}_{t-1} \Leftrightarrow$ drawing from $p(\mathbf{x}_t | \mathbf{x}_{t-1})$
 - Deterministic drift coming from $\mathbf{f}_t()$ – same for all particles
 - Stochastic drift coming from \mathbf{w}_{t-1} – different for each particles
- **Update step**: we must update our $\mathbf{x}_t^{(i)}$'s with the new measurement \mathbf{z}_t^*
 \Rightarrow **Importance Sampling**

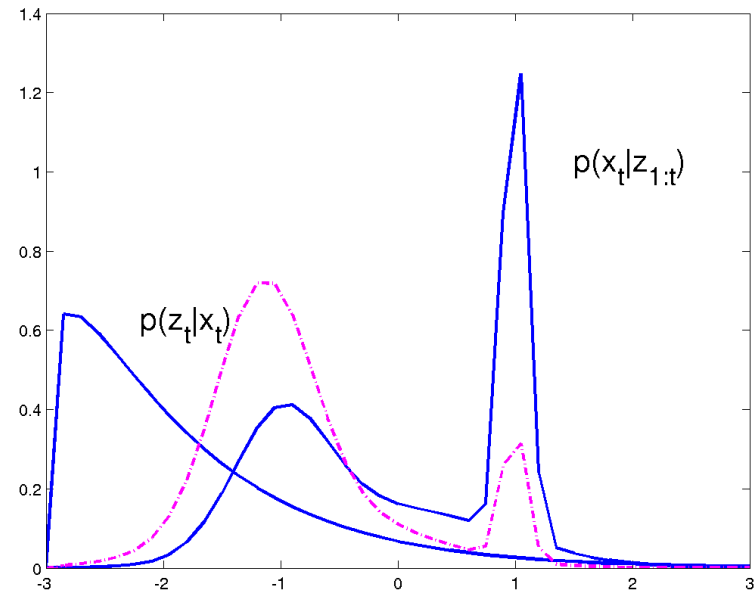
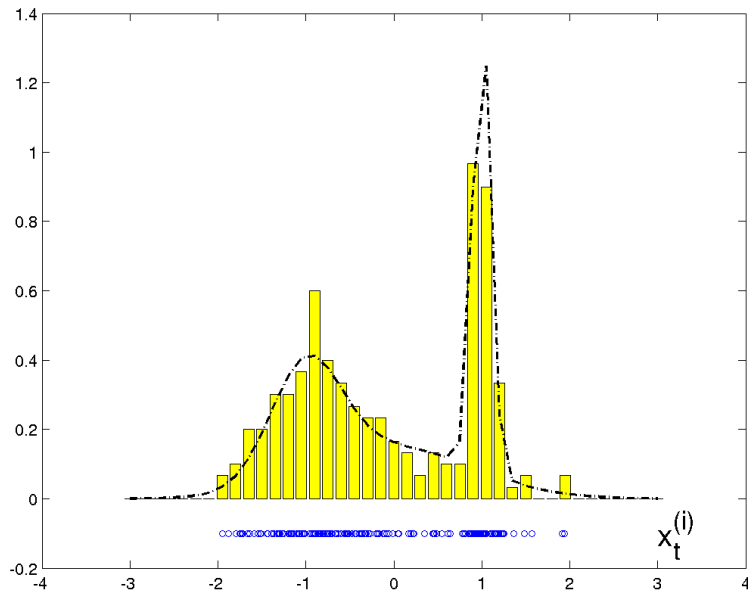
Importance Sampling I

- We have $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$
- We know $p(\mathbf{z}_t | \mathbf{x}_t)$ (measurement model)
- We know that $p(\mathbf{x}_t | \mathbf{z}_{1:t}) = k_t p(\mathbf{z}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$
- We have a realisation of \mathbf{z}_t
- We want $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{z}_{1:t})$

Importance Sampling II

We weight the particles $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$ with $\pi_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_t^{(i)})$

In 1-dimension: x axis = \mathbf{x}_t

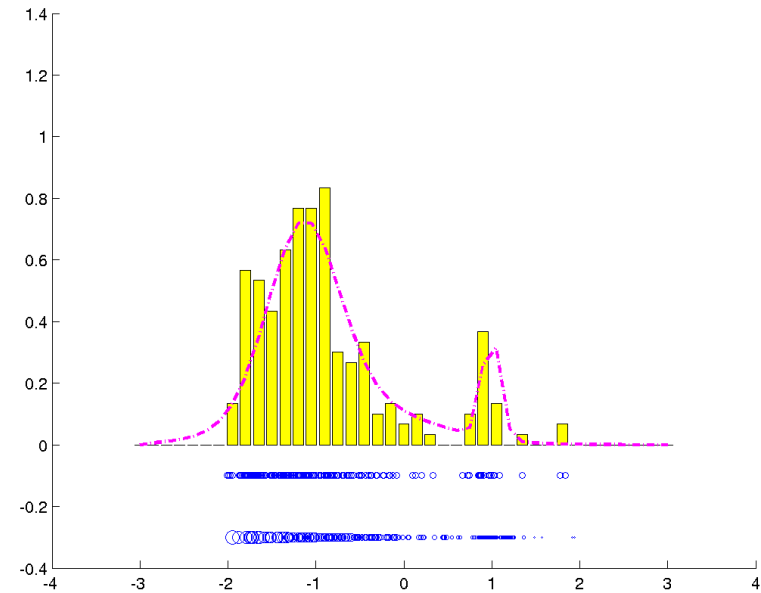
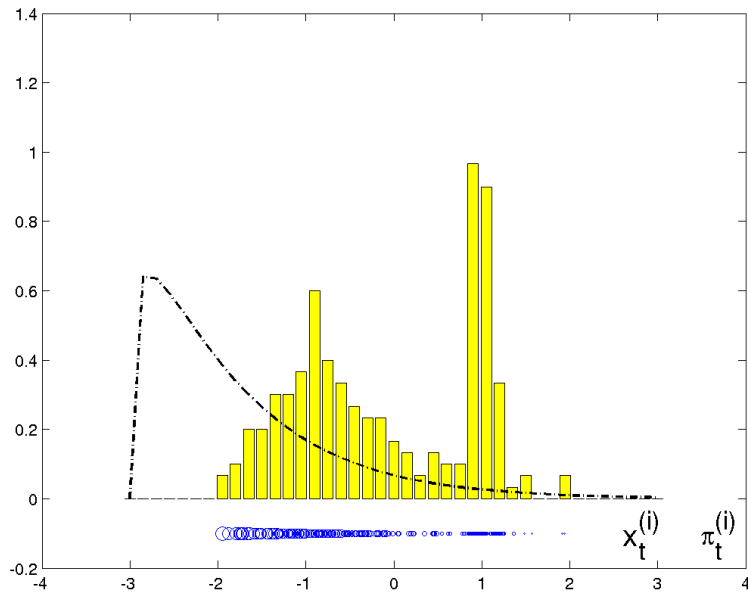


$$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$$

$$p(\mathbf{x}_t | \mathbf{z}_{1:t}) = k_t p(\mathbf{z}_t | \mathbf{x}_t) p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$$

Importance Sampling III

We weight the particles $\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{z}_{1:t-1})$ with $\pi_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_t^{(i)})$



Resampling \Leftrightarrow Big weights generate many particles

Particle filter: summary

- We have $p(\mathbf{x}_t | \mathbf{z}_{1:t})$ represented by N particles from the previous frame.
- **Prediction step:** apply the state evolution equation to each particle $\mathbf{x}_t^{(i)}$
 $\mathbf{x}_t^{(i)} = \mathbf{f}_t(\mathbf{x}_{t-1}^{(i)}) + \mathbf{w}_{t-1}$ (= draw from $p(\mathbf{x}_t | \mathbf{x}_{t-1})$)
- **Update step:** importance sampling* = weight part. $\mathbf{x}_t^{(i)}$ with
 $\pi_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_t^{(i)})$
- **Estimate** the mean state $E[\mathbf{x}_t] = \sum_i \pi^{(i)} \mathbf{x}_t^{(i)}$
- **Resampling:** big weights generate many particles
Then $p(\mathbf{x}_{t+1} | \mathbf{z}_{1:t+1})$ represented by N particles

Overview

1. An intuitive example: tracking using colour
2. Formalisation
 - State-space models
 - Bayesian Recursive estimation/filtering (BRF)
 - Particle solution to BRF
3. **Again the same example: colour based particle filter**
4. Summary

Colour based particle filter

- State vector = position, size and speed of ellipse containing the object to track $\mathbf{x}_t = (x, y, \dot{x}, \dot{y}, H_x, H_y, \dot{a})^T$
- Dynamical model $\mathbf{x}_t = A\mathbf{x}_{t-1} + \mathbf{w}_{t-1}$ i.e. simple 1st order linear system and Gaussian multivariate noise.
- N particles = N ellipses (+ size & speed) represent $p(\mathbf{x}_t | \mathbf{z}_{1:t})$
- **Prediction step**: apply the state evolution equation to each particle



Colour based particle filter

- **Update step:** a new observation (frame) comes up weight part. $\mathbf{x}_t^{(i)}$ with $\pi_t^{(i)} = p(\mathbf{z}_t | \mathbf{x}_t = \mathbf{x}_t^{(i)})$

$$\pi_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(d^{(i)})^2}{2\sigma^2}\right)$$

where $d^{(i)} = \sqrt{1 - \rho^{(i)}}$ and $\rho^{(i)} = \sum_u \sqrt{p_{\mathbf{x}^i}(u)q(u)}$
i.e. $d \sim N(0, \sigma)$

- **Resample.**

Particle filter: issues

PF's give framework for solving the BRF problem.
The BRF problem must be determined first

- Determine what should contain the state \mathbf{x}
- The state equation must reflect the dynamics of the object to track: i.e. we have to know (learn) $p(\mathbf{x}_t|\mathbf{x}_{t-1})$: difficult
- We have to know (learn) $p(\mathbf{z}_t|\mathbf{x}_t)$: difficult
- We have to know the prior $p_0(\mathbf{x})$ (initialisation)

Particle filter: Summary

- PF's offer a general framework for solving Bayesian Recursive Estimation in the state-space formalism (Kalman).
- PF's estimate $p(\mathbf{x}_t | \mathbf{z}_{1:t})$
- PF's represent $p(\mathbf{x} | \mathbf{z}_{1:t})$ as a set of particles $\mathbf{x}^{(i)}$
- The problem is solved in three steps
 - Prediction: use the state equation to compute state vector one time step ahead wrt current position
 - Update: use **importance sampling** to give more weight to $\mathbf{x}^{(i)}$'s that correspond to observation model $p(\mathbf{z}_t | \mathbf{x}_t)$ and current observation
 - Resample to get equal weights
- iterate the three steps as more observations become available