Tracking

Goal: Fundamentals of model-based tracking with emphasis on probabilistic formulations. Examples include the Kalman filter for linear-Gaussian problems, and maximum likelihood and particle filters for nonlinear/nonGaussian problems.

Outline

- Introduction
- Bayesian Filtering / Smoothing
- Likelihood Functions and Dynamical Models
- Kalman Filter
- Nonlinear/NonGaussian Processes
- Hill Climbing (Eigen-Tracking)
- Particle Filters

Readings: Chapter 17 of Forsyth and Ponce.
Matlab Tutorials: motionTutorial.m

Challenges in Tracking

Tracking is the inference object shape, appearance, and motion as a function of time.

Main players:

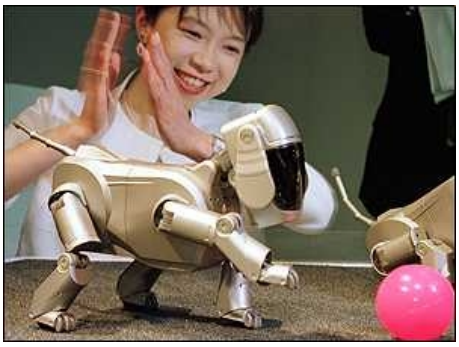
- what to model or estimate: shape (2D/3D), appearance, dynamics
- what to measure: color histograms, edges, feature points, flow, ...

Some of the main challenges:

- objects with many degrees of freedom,
  affecting shape, appearance, and motion;
- impoverished information due to occlusion or scale;
- multiple objects and background clutter

space / aerospace

surveillance



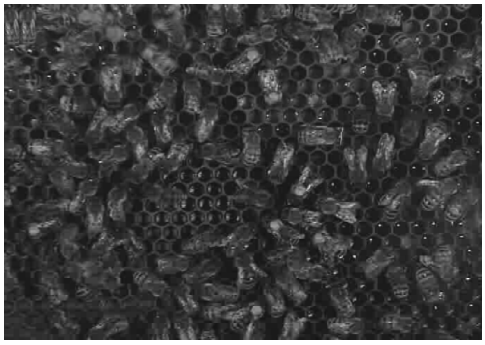
smart toys

automatic control

sports / kinesiology

human motion capture

non-rigid motions



biology (animal/cell/molecular)

Probability and Random Variables

A few basic properties of probability distributions will be used often:

- Conditioning (factorization):

$$p(a, b) \;=\; p(a|b)\, p(b) \;=\; p(b|a)\, p(a)$$

- Bayes' rule:

$$p(a|b) = \frac{p(b|a)\, p(a)}{p(b)}$$

- Independence: a and b are independent if and only if

$$p(a, b) \;=\; p(a)\, p(b)$$

- Marginalization:

$$p(b) \;=\; \int p(a,\, b)\, da \quad , \qquad p(b) \;=\; \sum_{a} p(a,\, b)$$

Probabilistic Formulation

We assume a state space representation in which time is discretized, and a *state* vector comprises all variables one wishes to estimate.

State: denoted $\mathbf{x}_t$ at time t, with the state history $\mathbf{x}_{1:t} = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$
– continuous variables (position, velocity, shape, size, ...)
– discrete variables (number of objects, gender, activity, ...)

Observations: the data measurements (images) with which we constrain state estimates, based on observation equation $\mathbf{z}_t = f(\mathbf{x}_t)$. The observations history is denoted $\mathbf{z}_{1:t} = (\mathbf{z}_1, \ldots, \mathbf{z}_t)$

Posterior Distribution: the conditional probability distribution over states specifies all we can possibly know (according to the model) about the state history from the observations.

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) \tag{1}$$

Filtering Distribution: often we only really want the marginal posterior distribution over the state at the current time given the observation history. This is called the filtering distribution:

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) \;=\; \int_{\mathbf{x}_1} \cdots \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) \tag{2}$$

Likelihood and Prior: using Bayes' rule we write the posterior in terms of a *likelihood*, $p(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t})$, and a *prior*, $p(\mathbf{x}_{1:t})$:

$$p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) \;=\; \frac{p(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t})\, p(\mathbf{x}_{1:t})}{p(\mathbf{z}_{1:t})}$$

Model Simplifications

The distribution $p(\mathbf{x}_{1:t})$, called a *prior* distribution, represents our prior beliefs about which state sequences (e.g., motions) are likely.

First-order Markov model for temporal dependence (dynamics):

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{1:t-1}) \;=\; p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \tag{3}$$

The order of a Markov model is the duration of temporal dependence (a first-order model requires past states up to a lag of one time step).

With a first-order Markov model one can write the distribution over the state history as a product of transitions from one time to the next:

$$\begin{aligned} p(\mathbf{x}_{1:t}) \;&=\; p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})\, p(\mathbf{x}_{1:t-1}) \\ &=\; p(\mathbf{x}_1) \prod_{j=2}^{t} p(\mathbf{x}_j \,|\, \mathbf{x}_{j-1}) \end{aligned} \tag{4}$$

The distribution $p(\mathbf{z}_{1:t} \,|\, \mathbf{x}_{1:t})$, often called a *likelihood* function, represents the likelihood that the state generated the observed data.

Conditional independence of observations:

$$\begin{aligned} p(\mathbf{z}_{1:t} \,|\, \mathbf{x}_{1:t}) \;&=\; p(\mathbf{z}_t \,|\, \mathbf{x}_t)\, p(\mathbf{z}_{1:t-1} \,|\, \mathbf{x}_{1:t-1}) \\ &=\; \prod_{\tau=1}^{t} p(\mathbf{z}_\tau \,|\, \mathbf{x}_\tau) \end{aligned} \tag{5}$$

That is, we assume that the observations at different times are independent when we know the true underlying states (or causes).

Filtering and Prediction Distributions

With the above model assumptions, one can express the posterior distribution recursively:

$$\begin{aligned} p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) &\propto p(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t})\, p(\mathbf{x}_{1:t}) \\ &= \prod_{\tau=1}^{t} p(\mathbf{z}_\tau \mid \mathbf{x}_\tau)\ p(\mathbf{x}_1) \prod_{j=2}^{t} p(\mathbf{x}_j \mid \mathbf{x}_{j-1}) \\ &\propto p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{1:t-1} \mid \mathbf{z}_{1:t-1}) \end{aligned} \tag{6}$$

The *filtering distribution* can also be written recursively:

$$\begin{aligned} p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) &= \int_{\mathbf{x}_1} \cdots \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) \\ &= c\, p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) \end{aligned} \tag{7}$$

with a *prediction distribution* defined as

$$p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) = \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}) \tag{8}$$

Recursion is important:

- it allows us to express the filtering distribution at time t in terms of the filtering distribution at time $t-1$ and the evidence at time t.
- all useful information from the past is summarized in the previous posterior (and hence in the prediction distribution).

Without recursion one may have to store all previous images to compute the the filtering distribution at time t.

Derivation of Filtering and Prediction Distributions

Filtering Distribution: Given the model assumptions in Equations (4) and (5), along with Bayes' rule, we can derive Equation (7) as follows:

$$\begin{aligned}
p(\mathbf{x}_t \mid \mathbf{z}_{1:t}) &= \int_{\mathbf{X}_1} \cdots \int_{\mathbf{X}_{t-1}} p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}) \\
&= \frac{1}{p(\mathbf{z}_{1:t})} \int_{\mathbf{X}_1} \cdots \int_{\mathbf{X}_{t-1}} p(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t})\, p(\mathbf{x}_{1:t}) \\
&= c \int_{\mathbf{X}_1} \cdots \int_{\mathbf{X}_{t-1}} p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{z}_{1:t-1} \mid \mathbf{x}_{1:t-1})\, p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{1:t-1}) \\
&= c\, p(\mathbf{z}_t \mid \mathbf{x}_t) \int_{\mathbf{X}_1} \cdots \int_{\mathbf{X}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{1:t-1},\, \mathbf{z}_{1:t-1}) \\
&= c\, p(\mathbf{z}_t \mid \mathbf{x}_t) \int_{\mathbf{X}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \int_{\mathbf{X}_1} \cdots \int_{\mathbf{X}_{t-2}} p(\mathbf{x}_{1:t-1},\, \mathbf{z}_{1:t-1}) \\
&= c\, p(\mathbf{z}_t \mid \mathbf{x}_t) \int_{\mathbf{X}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1},\, \mathbf{z}_{1:t-1}) \\
&= c\, p(\mathbf{z}_{1:t-1})\, p(\mathbf{z}_t \mid \mathbf{x}_t) \int_{\mathbf{X}_{t-1}} p(\mathbf{x}_t \mid \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1}) \\
&= c'\, p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) \ .
\end{aligned}$$

Batch Filter-Smoother (Forward-Backward Belief Propagation): We can derive the filter-smoother equation (9), for $1 < \tau \leq t$, as follows:

$$\begin{aligned}
p(\mathbf{x}_\tau \mid \mathbf{z}_{1:t}) &= \frac{1}{p(\mathbf{z}_{1:t})} \int_{\mathbf{X}_{1:\tau-1}} \int_{\mathbf{X}_{\tau+1:t}} p(\mathbf{x}_{1:t})\, p(\mathbf{z}_{1:t} \mid \mathbf{x}_{1:t}) \\
&= c \int_{\mathbf{X}_{1:\tau-1}} \int_{\mathbf{X}_{\tau+1:t}} p(\mathbf{x}_{1:\tau})\, p(\mathbf{x}_{\tau+1:t} \mid \mathbf{x}_\tau)\, p(\mathbf{z}_{1:\tau-1} \mid \mathbf{x}_{1:\tau-1})\, p(\mathbf{z}_\tau \mid \mathbf{x}_\tau)\, p(\mathbf{z}_{\tau+1:t} \mid \mathbf{x}_{\tau+1:t}) \\
&= c\, p(\mathbf{z}_\tau \mid \mathbf{x}_\tau) \int_{\mathbf{X}_{1:\tau-1}} p(\mathbf{x}_{1:\tau})\, p(\mathbf{z}_{1:\tau-1} \mid \mathbf{x}_{1:\tau-1}) \int_{\mathbf{X}_{\tau+1:t}} p(\mathbf{x}_{\tau+1:t} \mid \mathbf{x}_\tau)\, p(\mathbf{z}_{\tau+1:t} \mid \mathbf{x}_{\tau+1:t}) \\
&= c\, p(\mathbf{z}_\tau \mid \mathbf{x}_\tau)\, p(\mathbf{x}_\tau \mid \mathbf{z}_{1:\tau-1}) \int_{\mathbf{X}_{\tau+1:t}} p(\mathbf{x}_\tau \mid \mathbf{x}_{\tau+1:t}) \frac{p(\mathbf{x}_{\tau+1:t})}{p(\mathbf{x}_\tau)}\, p(\mathbf{x}_{\tau+1:t} \mid \mathbf{z}_{\tau+1:t}) \frac{p(\mathbf{z}_{\tau+1:t})}{p(\mathbf{x}_{\tau+1:t})} \\
&= c\, p(\mathbf{z}_\tau \mid \mathbf{x}_\tau)\, p(\mathbf{x}_\tau \mid \mathbf{z}_{1:\tau-1}) \frac{p(\mathbf{z}_{\tau+1:t})}{p(\mathbf{x}_\tau)} \int_{\mathbf{X}_{\tau+1:t}} p(\mathbf{x}_\tau \mid \mathbf{x}_{\tau+1:t})\, p(\mathbf{x}_{\tau+1:t} \mid \mathbf{z}_{\tau+1:t}) \\
&= \frac{c'}{p(\mathbf{x}_\tau)}\, p(\mathbf{z}_\tau \mid \mathbf{x}_\tau)\, p(\mathbf{x}_\tau \mid \mathbf{z}_{1:\tau-1})\, p(\mathbf{x}_\tau \mid \mathbf{z}_{\tau+1:t}) \ .
\end{aligned}$$

Filtering and Smoothing

Provided one can invert the dynamics equation, one can also perform inference (recursively) backwards in time:

$$\begin{aligned} p(\mathbf{x}_\tau \,|\, \mathbf{z}_{\tau:t}) &= c\, p(\mathbf{z}_\tau \,|\, \mathbf{x}_\tau) \int_{\mathbf{x}_{\tau+1}} p(\mathbf{x}_\tau \,|\, \mathbf{x}_{\tau+1})\, p(\mathbf{x}_{\tau+1} \,|\, \mathbf{z}_{\tau+1:t}) \\ &= c\, p(\mathbf{z}_\tau \,|\, \mathbf{x}_\tau)\, p(\mathbf{x}_\tau \,|\, \mathbf{z}_{\tau+1:t}) \end{aligned}$$

That is, the distribution depends on the likelihood with the current data, and the prediction based on the future data, through the (backward) filtering distribution at time $t+1$.

Smoothing distribution (forward-backward belief propagation):

$$p(\mathbf{x}_\tau \,|\, \mathbf{z}_{1:t}) = \frac{c}{p(\mathbf{x}_\tau)}\, p(\mathbf{z}_\tau \,|\, \mathbf{x}_\tau)\, p(\mathbf{x}_\tau \,|\, \mathbf{z}_{1:\tau-1})\, p(\mathbf{x}_\tau \,|\, \mathbf{z}_{\tau+1:t}) \qquad (9)$$

The smoothing distribution therefore accumulates information from past, present, and future data.

Batch Algorithms: Estimation of state sequences using the entire observation sequence (i.e., using all past, present & future data):

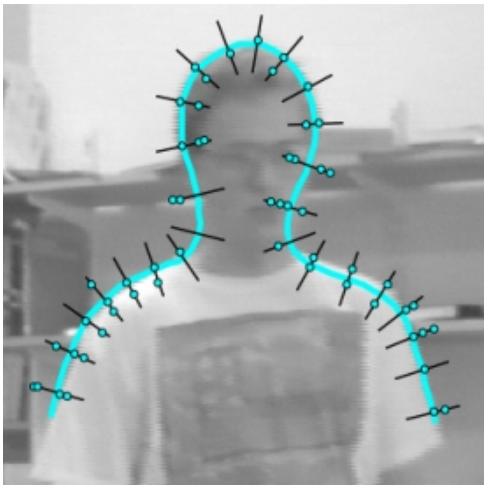
- the filter-smoother algorithm is efficient, when applicable.
- storage/delays make this unsuitable for many tracking domains.

Online Algorithms: Recursive inference (7) is causal. Estimation of $\mathbf{x}_t$ occurs as soon as observations at time t are available, thereby using present and past data only.

Likelihood Functions

There are myriad ways in which image measurements have been used for tracking. Some of the most common include:

- *Feature points:* E.g., Gaussian noise in the measured feature point locations. The points might be specified *a priori*, or learned when the object is first imaged at time 0.
- *Image templates:* E.g., subspace models learned prior to tracking, or brightness constancy as used in flow estimation.
- *Color histograms*: E.g., mean-shift to track modes of local color distribution, for robustness to deformations.
- *Gradient histograms*: E.g., histograms of oriented gradients (HOG) in local patches to capture image orientation structure as a function of spatial position over target.
- *Image curves (or edges)*: E.g., with Gaussian noise in measured location normal to the contour.



Temporal Dynamics

We often assume a combination of deterministic and stochastic dynamics. Common linear models include:

- Random walk with zero velocity and IID Gaussian *process noise*:

$$\mathbf{x}_t \;=\; \mathbf{x}_{t-1} + \vec{\eta}_d \;\;, \quad \vec{\eta}_d \;\sim\; \mathcal{N}(0,\, C)$$

- Random walk with zero acceleration and Gaussian *process noise*

$$\begin{pmatrix} \mathbf{x}_t \\ \vec{\mathbf{v}}_t \end{pmatrix} \;=\; \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x}_{t-1} \\ \vec{\mathbf{v}}_{t-1} \end{pmatrix} + \begin{pmatrix} \vec{\eta}_d \\ \vec{\epsilon}_d \end{pmatrix}$$

  where $\vec{\eta}_d \sim \mathcal{N}(0,\, C_x)$ and $\vec{\epsilon}_d \sim \mathcal{N}(0,\, C_v)$.

- For higher-order models we define an augmented state vector, and then use a first-order formulation. E.g., for a second-order model:

$$\mathbf{x}_t \;=\; A\mathbf{x}_{t-1} + B\mathbf{x}_{t-2} + \vec{\eta}_d$$

  one can define

$$\vec{\mathbf{y}}_t \equiv \begin{pmatrix} \mathbf{x}_t \\ \\ \mathbf{x}_{t-1} \end{pmatrix}$$

  for which the equivalent first-order augmented-state model is

$$\vec{\mathbf{y}}_t \;=\; \begin{pmatrix} A & B \\ I & 0 \end{pmatrix} \vec{\mathbf{y}}_{t-1} + \begin{pmatrix} \vec{\eta}_d \\ 0 \end{pmatrix}$$

  Typical observation model: $\mathbf{z}_t = f(\,[I\;0] \cdot \vec{\mathbf{y}}_t)$ plus Gaussian noise.

Dynamical Models

There are many other useful dynamical models. For example, harmonic oscillation can be expressed as

$$\frac{d^2x}{dt^2} = -x$$

or as a first order system with

$$\frac{d\vec{\mathbf{u}}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \vec{\mathbf{u}} \quad , \quad \text{where} \quad \vec{\mathbf{u}} \equiv \begin{pmatrix} x \\ v \end{pmatrix}$$

A first-order approximation yields:

$$\begin{aligned} \vec{\mathbf{u}}_t &= \vec{\mathbf{u}}_{t-1} + \Delta t \frac{d\vec{\mathbf{u}}}{dt} + \begin{pmatrix} \eta \\ \epsilon \end{pmatrix} \\ &= \begin{pmatrix} 1 & \Delta t \\ -\Delta t & 1 \end{pmatrix} \vec{\mathbf{u}}_{t-1} + \begin{pmatrix} \eta \\ \epsilon \end{pmatrix} \end{aligned}$$

In many cases it is useful to learn a suitable model of state dynamics. There are well-know algorithms for learning linear auto-regressive models of variable order.

Kalman Filter

Assume a linear dynamical model with Gaussian *process noise*, and a linear observation model with Gaussian *observation noise*:

$$\mathbf{x}_t \;=\; A\,\mathbf{x}_{t-1} + \vec{\eta}_d\;, \quad \vec{\eta}_d \;\sim\; \mathcal{N}(0,\, C_d)\;. \tag{10}$$

$$\mathbf{z}_t \;=\; M\,\mathbf{x}_t + \vec{\eta}_m\;, \quad \vec{\eta}_m \;\sim\; \mathcal{N}(0,\, C_m) \tag{11}$$

The *transition density* is therefore Gaussian, centred at mean $A\,\mathbf{x}_{t-1}$, with covariance C_d :

$$p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) \;=\; G(\mathbf{x}_t;\, A\,\mathbf{x}_{t-1},\, C_d)\;. \tag{12}$$

The *observation density* is also Gaussian:

$$p(\mathbf{z}_t \,|\, \mathbf{x}_t) \;=\; G(\mathbf{z}_t;\, M\,\mathbf{x}_t,\, C_m)\;. \tag{13}$$

Because the product of Gaussians is Gaussian, and the marginals of a Gaussian are Gaussian, it is straightforward (but tedious) to show that the prediction and filtering distributions are both Gaussian:

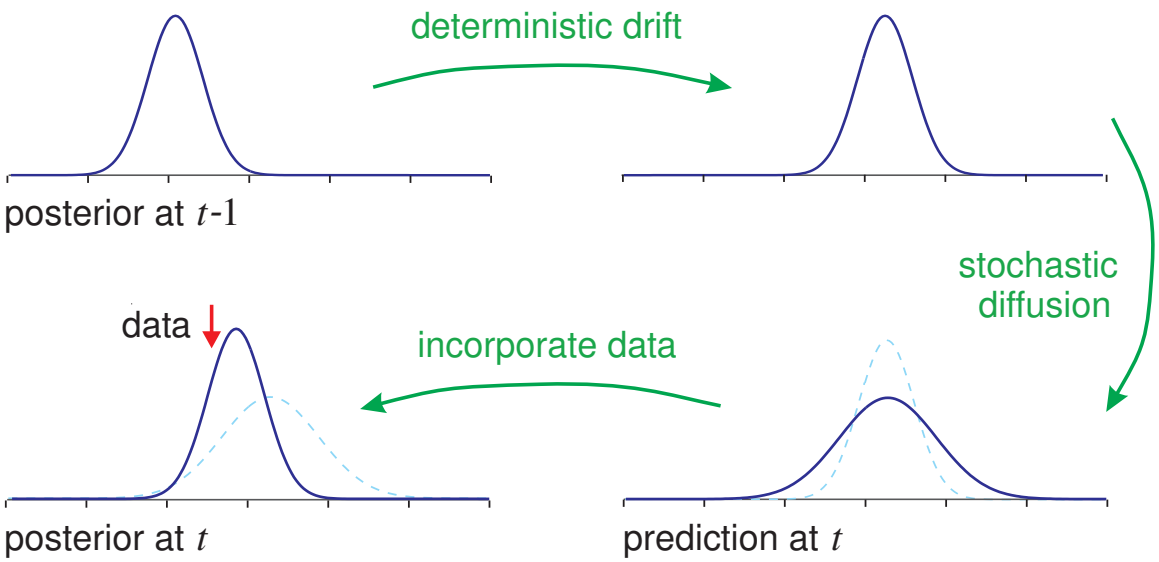
$$p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}) \;=\; \int_{\mathbf{x}_{t-1}} p(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})\, p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}) \;=\; G(\mathbf{x}_t;\, \mathbf{x}_t^-,\, C_t^-) \tag{14}$$

$$p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t}) \;=\; c\, p(\mathbf{z}_t \,|\, \mathbf{x}_t)\, p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}) \;=\; G(\mathbf{x}_t;\, \mathbf{x}_t^+,\, C_t^+) \tag{15}$$

with closed-form expressions for the means $\mathbf{x}_t^-$, $\mathbf{x}_t^+$ and covariances C_t^-, C_t^+.

Kalman Filter

Depiction of Kalman updates:



Kalman Filter (details)

To begin, suppose we know that $\mathbf{x} \sim \mathcal{N}(\vec{\mathbf{0}},\, C)$, and let $\vec{\mathbf{y}} = A\mathbf{x}$. Since $\mathbf{x}$ is zero-mean, it is clear that $\vec{\mathbf{y}}$ will also be zero-mean. Further, the covariance of $\vec{\mathbf{y}}$ is given by

$$\mathrm{E}[\,\vec{\mathbf{y}}\vec{\mathbf{y}}^T] \;=\; \mathrm{E}[\,A\,\mathbf{x}\mathbf{x}^T A^T] \;=\; A\,\mathrm{E}[\,\mathbf{x}\mathbf{x}^T]\,A^T \;=\; A\,C\,A^T \tag{16}$$

Now, let's use this to derive the form of the prediction distribution. Let's say that we know the filtering distribution from the previous time instant, $t-1$, and let's say it is Gaussian with mean $\mathbf{x}^+_{t-1}$ with covariance C^+_{t-1}.

$$p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}) \;=\; G(\mathbf{x}_{t-1};\, \mathbf{x}^+_{t-1},\, C^+_{t-1})\;. \tag{17}$$

And, as above, we assume a linear-Gaussian dynamical model,

$$\mathbf{x}_t \;=\; A\,\mathbf{x}_{t-1} + \vec{\eta}_d \quad , \quad \vec{\eta}_d \;\sim\; N(0,\, C_d)\;. \tag{18}$$

From above we know that $A\,\mathbf{x}_{t-1}$ is Gaussian. And we'll assume that the Gaussian process noise $\vec{\eta}_d$ is independent of the previous posterior distribution. So, 18 is the sum of two independent Gaussian random variable, and hence the corresponding density is just the convolution of their individual densities. Remember that the convolution of two Gaussians with covariances C_1 and C_2 is Gaussian with covariance $C_1 + C_2$. With this, it follows from (17) and (18) that the prediction mean and covariance of $p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1})$ in (14) are given by

$$\mathbf{x}^-_t \;=\; A\,\mathbf{x}^+_{t-1} \;\;,\;\; C^-_t \;=\; A\,C^+_{t-1}A^T + C_d\;.$$

This gives us the form of the prediction density.

Now, let's turn to the filtering distribution. That is, we wish to combine the prediction distribution with the observation density for the current observation, $\mathbf{z}_t$, in order to form the filtering distribution at time t. In particular, using (15), with (13) and the results above, it is straightforward to see that

$$\begin{aligned} p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t}) \;&\propto\; p(\mathbf{z}_t \,|\, \mathbf{x}_t)\; p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1}) && (19)\\ &=\; G(\mathbf{z}_t;\, M\mathbf{x}_t,\, C_m)\; G(\mathbf{x}_t;\, \mathbf{x}^-_t,\, C^-_t)\;. && (20) \end{aligned}$$

Of course the product of two Gaussians is Gaussian; and it remains to work out expressions for its mean and covariance. This requires somewhat tedious algebraic manipulation.

While there are many ways to express the posterior mean and covariance, the conventional solution defines an intermediate quantity called the *Kalman Gain*, K_t, given by

$$K_t \;=\; C_{t-1}^{-} M^T \left(M C_{t-1}^{-} M^T + C_m \right)^{-1} \;.$$

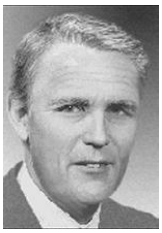
Using the Kalman gain, one can express the posterior mean and variance, $\mathbf{x}_t^+$ and C_t^+, as follows:

$$\begin{aligned} \mathbf{x}_t^+ &= \mathbf{x}_t^- + K_t \left(\mathbf{z}_t - M\mathbf{x}_t^- \right) \;, \\ C_t^+ &= \left(\mathbf{I} - K_t M \right) C_t^- \\ &= (\mathbf{I} - K_t) C_t^- (\mathbf{I} - K_t)^T + K_t C_m K_t^T \end{aligned}$$

The Kalman filter began to appear in computer vision papers in the late 1980s. The first two main applications were for (1) automated road following where lane markers on the highway were tracking to keep a car on the road; and (2) the estimation of the 3D struction and motion of a rigid object (or scene) with respect to a camera, given a sequences of point tracks through time.

Dickmanns & Graefe, "Dynamic monocular machine vision." *Machine Vision and Appl.*, 1988.

Broida, Chandrashekhar & Chellappa, "Rigid structure from feature tracks under perspective projection." *IEEE Trans. Aerosp. & Elec. Sys.*, 1990.

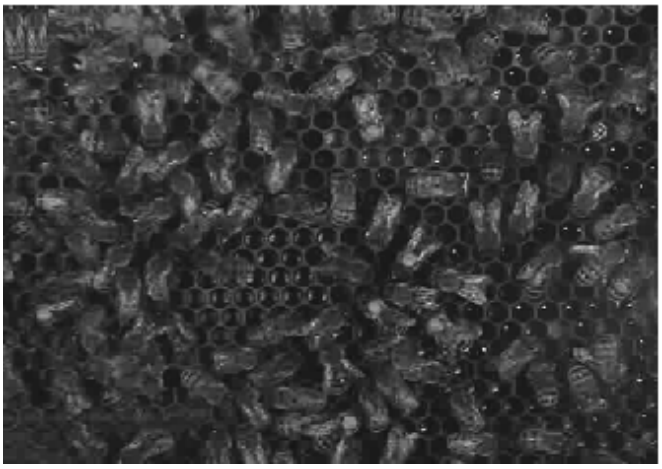
R.E. Kalman 

Non-linear / Non-Gaussian Systems

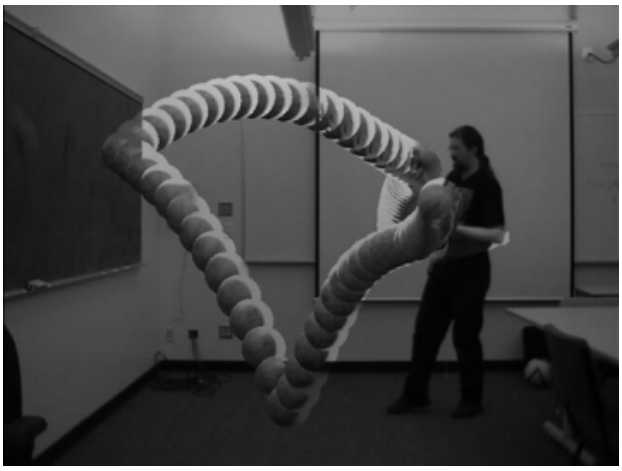
Tracking problems are rarely linear/Gaussian. The posterior, filtering and prediction distributions are usually nonGaussian, and often they are multimodal. The reasons for this include, among other things,

- scene clutter and occlusion, where many parts of the scene may appear similar to parts of the object being tracking
- image observation models are often nonlinear with heavy-tailed noise so that we can cope with outliers, complex appearance changes, and the nonlinearity of perspective projection.
- temporal dynamics are often nonlinear (e.g., human motion)

For example:



Background clutter and distractors.



Nonlinear dynamics.

Extended and Unscented Kalman Filters

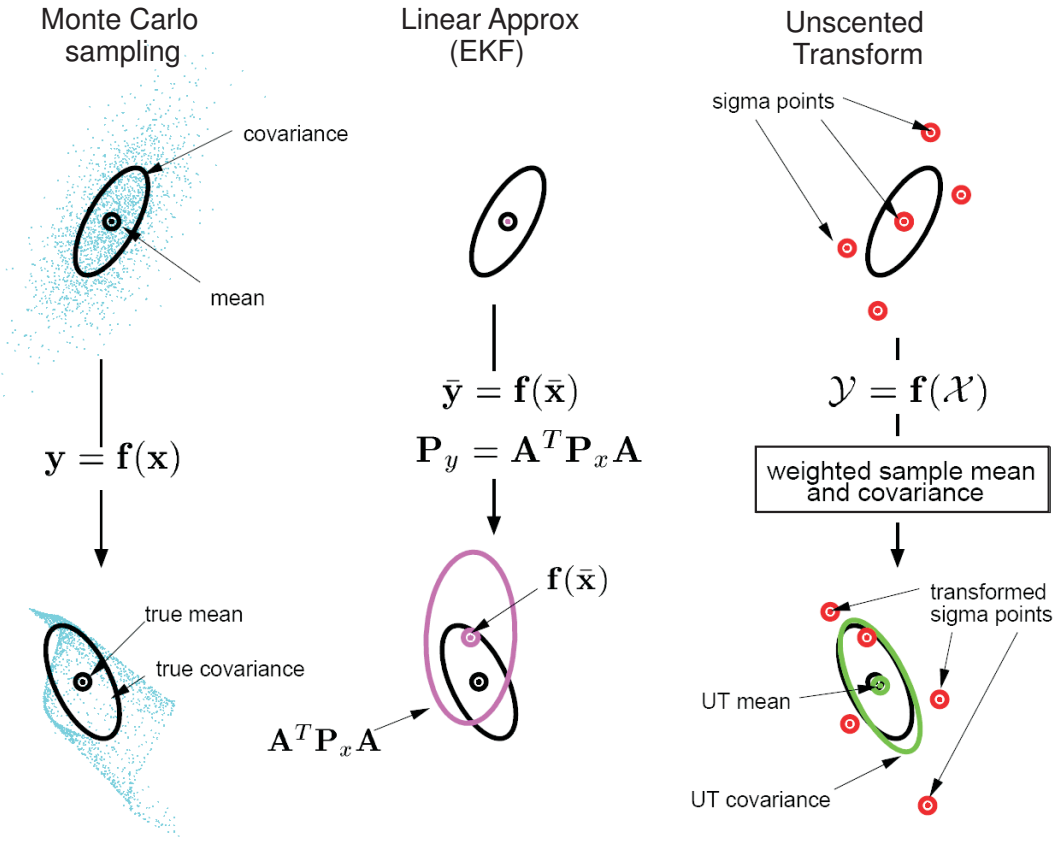
Extended Kalman Filter (EKF): For nonlinear dynamical models one can linearize the dynamics at current state; that is,

$$\vec{\mathbf{x}}_t \;=\; f(\vec{\mathbf{x}}_{t-1}) + \vec{\eta}_d \;\approx\; A\,\vec{\mathbf{x}}_{t-1} + \vec{\eta}_d\,,$$

where $A = \nabla f(\vec{\mathbf{x}})\,|_{\vec{\mathbf{x}}=\vec{\mathbf{x}}_{t-1}}$ and $\vec{\eta}_d \sim \mathcal{N}(0, C_d)$. One can also iterate the approximation to obtain the *Iterated Extended Kalman Filter (IEKF)*. In practice the EKF and IEKF have problems unless the dynamics are close to linear.

Unscented Kalman Filter (UKF): Estimate posterior mean and variance to second-order with arbitrary dynamics *[Julier & Uhlmann, 2004]*. Rather than linearize the dynamics to ensure Gaussian predictions, use exact 1^{st} and 2^{nd} moments of the prediction density under the nonlinear dynamics:

- Choose *sigma points* $\mathbf{x}_j$ whose sample mean and covariance equal the mean and variance of the Gaussian posterior at t
- Apply nonlinear dynamics to each sigma point, $\mathbf{y}_j = f(\mathbf{x}_j)$, and then compute the sample mean and covariances of the $\mathbf{y}_j$.



Hill-Climbing

Rather than approximating the posterior or filtering distributions (fully), just find local maxima of the filtering distribution at each time step. By also computing the curvature of the log filtering distribution at the maxima, one obtains local Gaussian approximations.

E.g., Eigen-Tracking *[Black & Jepson, '96]*:

- Assume we have learned (offline) a subspace appearance model for an object under varying pose, articulation, and lighting:

$$B(\mathbf{x},\, \mathbf{c}) \;=\; \sum_k c_k\, B_k(\mathbf{x})$$

- During tracking, we seek the image warp parameters $\mathbf{a}_t$, at each time t, and the subspace coefficients $\mathbf{c}_t$ such that the warped image is *explained* by the subspace; i.e.,

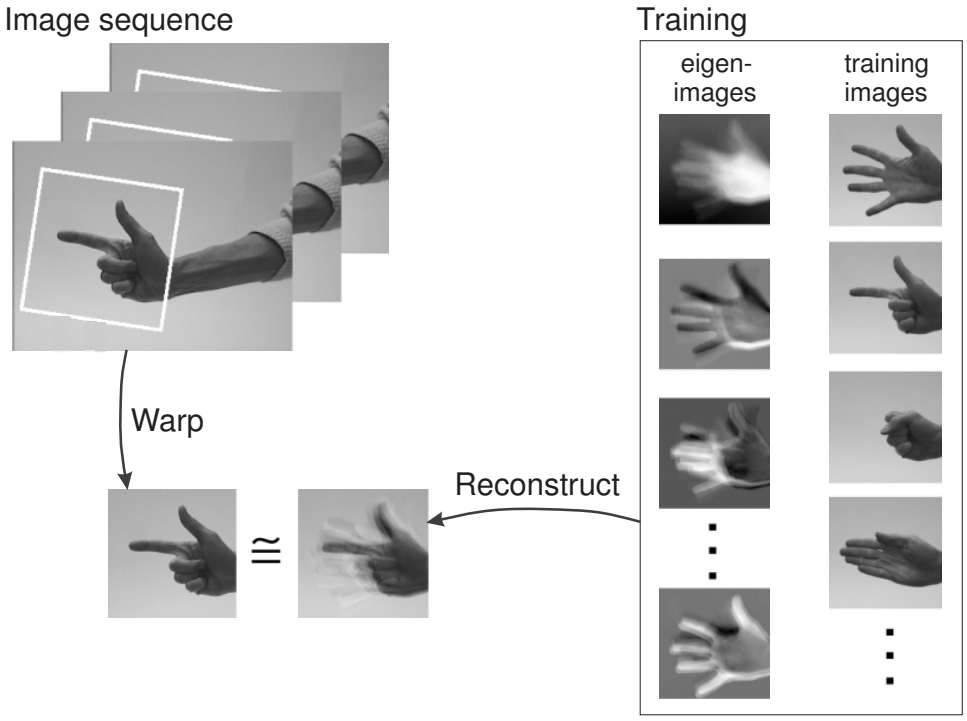
$$I(\mathbf{w}(\mathbf{x}, \mathbf{a}_t),\, t) \;\approx\; B(\mathbf{x},\, \mathbf{c}_t)$$

- A robust objective function helps cope with modeling errors, occlusions and other outliers:

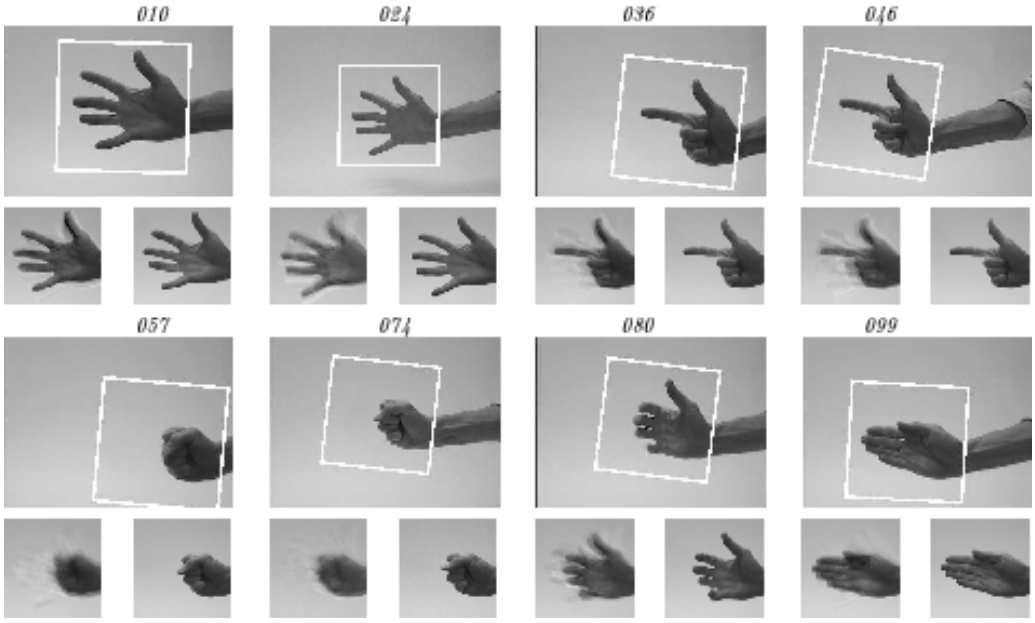
$$E(\mathbf{a}_t,\, \mathbf{c}_t) \;=\; \sum_{\mathbf{x}} \rho(\, I(\mathbf{w}(\mathbf{x}, \mathbf{a}_t),\, t) - B(\mathbf{x},\, \mathbf{c}_t)\,)$$

- Initialize the estimation at time t with ML estimate from time t–1.

Eigen-Tracking



Test Results:

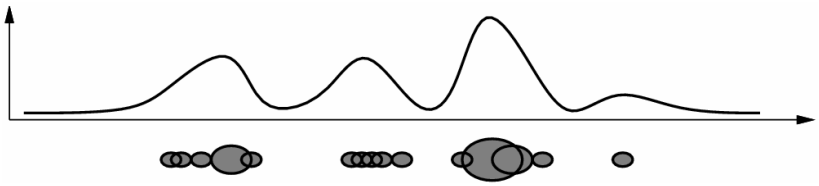


Figures show superimposed tracking region (top), the best reconstructed pose (bottom left), and the closest training image (bottom right).

Sequential Monte Carlo

For many problems ambiguity is sufficiently problematic that we must maintain a better representation of the filtering distribution.

Approximate the filtering distribution, $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$, using a weighted sample set, $\mathcal{S} = \{\mathbf{x}_t^{(j)}, w_t^{(j)}\}$; i.e., with a collection of point probability masses at locations $\mathbf{x}_t^{(j)}$ with weights $w_t^{(j)} = w(\mathbf{x}_t^{(j)})$ for some weight function $w(\mathbf{x})$.



Let's consider this in more detail below.

Monte Carlo: Approximate the filtering distribution $\mathcal{P}$ with samples drawn from it, $\mathcal{S} = \{\mathbf{x}^{(j)}\}_{j=1}^{N}$. Then, use sample statistics to approximate expectations under $\mathcal{P}$; i.e., for functions $f(\mathbf{x})$,

$$E_{\mathcal{S}}\left[\, f(\mathbf{x}) \,\right] \;\equiv\; \frac{1}{N}\sum_{j=1}^{N} f(\mathbf{x}^{(j)}) \;\stackrel{N\to\infty}{\longrightarrow}\; \int f(\mathbf{x})\,\mathcal{P}(\mathbf{x})\,d\mathbf{x} \;\equiv\; E_{\mathcal{P}}\left[\, f(\mathbf{x}) \,\right]$$

But, we don't know how to draw samples from our distribution $p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$.

Particle Filter

Importance Sampling: If one draws samples $\mathbf{x}^{(j)}$ from a *proposal distribution*, $\mathcal{Q}(\mathbf{x})$, with weights $w^{(j)}$, then

$$E_{\mathcal{S}}\,[\,f(\mathbf{x})\,] \;\equiv\; \sum_{j=1}^{N} w^{(j)} f(\mathbf{x}^{(j)}) \;\stackrel{N\to\infty}{\longrightarrow}\; E_{\mathcal{Q}}\,[\,w(\mathbf{x})\,f(\mathbf{x})\,]$$

If $w(\mathbf{x}) = \mathcal{P}(\mathbf{x})/\mathcal{Q}(\mathbf{x})$, then the weighted sample statistics approximate the desired expectations under $\mathcal{P}(\mathbf{x})$; i.e.,

$$\begin{aligned} E_{\mathcal{Q}}\,[\,w(\mathbf{x})\,f(\mathbf{x})\,] &= \int w(\mathbf{x})\,f(\mathbf{x})\,\mathcal{Q}(\mathbf{x})\,d\mathbf{x} \\ &= \int f(\mathbf{x})\,\mathcal{P}(\mathbf{x})\,d\mathbf{x} \\ &= E_{\mathcal{P}}[\,f(\mathbf{x})\,] \end{aligned}$$

Sequential Monte Carlo *[Arulampalam et al, 2002]:* The sample set is updated each time instant, incorporating new data, and possibly re-sampling the set of state samples $\mathbf{x}^{(j)}$. Key idea: exploit the form of the filtering distribution for importance sampling,

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

and the facts that it is often easy to evaluate the likelihood, and one can usually draw samples from the prediction distribution

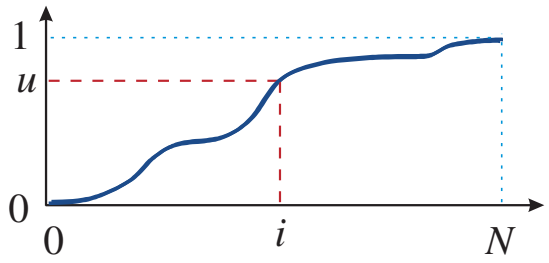
Simple Particle Filter: If we sample from the prediction distribution

$$\mathcal{Q} \;=\; p(\mathbf{x}_t \,|\, \mathbf{z}_{1:t-1})$$

then the weights must be $w(\mathbf{x}) = c\,p(\mathbf{z}_t \,|\, \mathbf{x}_t)$, with $c = 1/p(\mathbf{z}_t \,|\, \mathbf{z}_{1:t-1})$.

Simple Particle Filter

Step 1: Sample the approximate filtering distribution, $p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1})$, given the weighted sample set $\mathcal{S}_{t-1} = \{\mathbf{x}_{t-1}^{(j)}, w_{t-1}^{(j)}\}$. To do this, treat the N weights as probabilities and sample from the cumulative weight distribution; i.e., draw sample $u \sim \mathcal{U}(0, 1)$ to sample an index i.



Step 2: With sample $\mathbf{x}_{t-1}^{(i)}$, the dynamics provides a distribution over states at time t, i.e., $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$. A fair sample from the dynamics,

$$\mathbf{x}_t^{(j)} \sim p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)})$$

is then a fair sample from the prediction distribution $p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})$.

Step 3: To complete one iteration to find the new filtering distribution, given the samples $\mathbf{x}_t^{(j)}$, we compute the weights $w_t^{(j)} = c\, p(\mathbf{z}_t \mid \mathbf{x}_t^{(j)})$:

- $p(\mathbf{z}_t \mid \mathbf{x}_t^{(j)})$ is the data likelihood that we know how to evaluate.
- Using Bayes' rule one can show that c satisfies

$$\frac{1}{c} = p(\mathbf{z}_t \mid \mathbf{z}_{1:t-1}) = \int p(\mathbf{z}_t \mid \mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})\, d\mathbf{x}_t \approx \sum_j p(\mathbf{z}_t \mid \mathbf{x}_t^{(j)})$$

Using the approximation, the weights become normalized likelihoods (so they sum to 1).

Particle Filter Remarks

One can think of a sampled approximation as a sum of Dirac delta functions. A weighted sample set $\mathcal{S}_{t-1} = \{\mathbf{x}_{t-1}^{(j)},\, w_{t-1}^{(j)}\}$, is just a weighted set of delta functions::

$$p(\mathbf{x}_{t-1} \,|\, \mathbf{z}_{1:t-1}) \;=\; \sum_{j=1}^{N} w^{(j)}\, \delta(\mathbf{x}_{t-1} - \mathbf{x}_{t-1}^{(j)})$$

Sometimes people smooth the delta functions to create smoothed approximations (called Parzen window density estimates).
If one considers the prediction distribution, and uses the properties of delta functions under integration, then one obtains a mixture model for the prediction distribution. That is, given a weighted sample set $\mathcal{S}_{t-1}$ as above the prediction distribution in (8) is a linear mixture model

$$p(\mathbf{x}_{t} \,|\, \mathbf{z}_{1:t-1}) \;=\; \sum_{j=1}^{N} w^{(j)}\, p(\mathbf{x}_{t} \,|\, \mathbf{x}_{t-1}^{(j)})$$

The sampling method on the previous page is just a fair sampling method for linear mixture models.

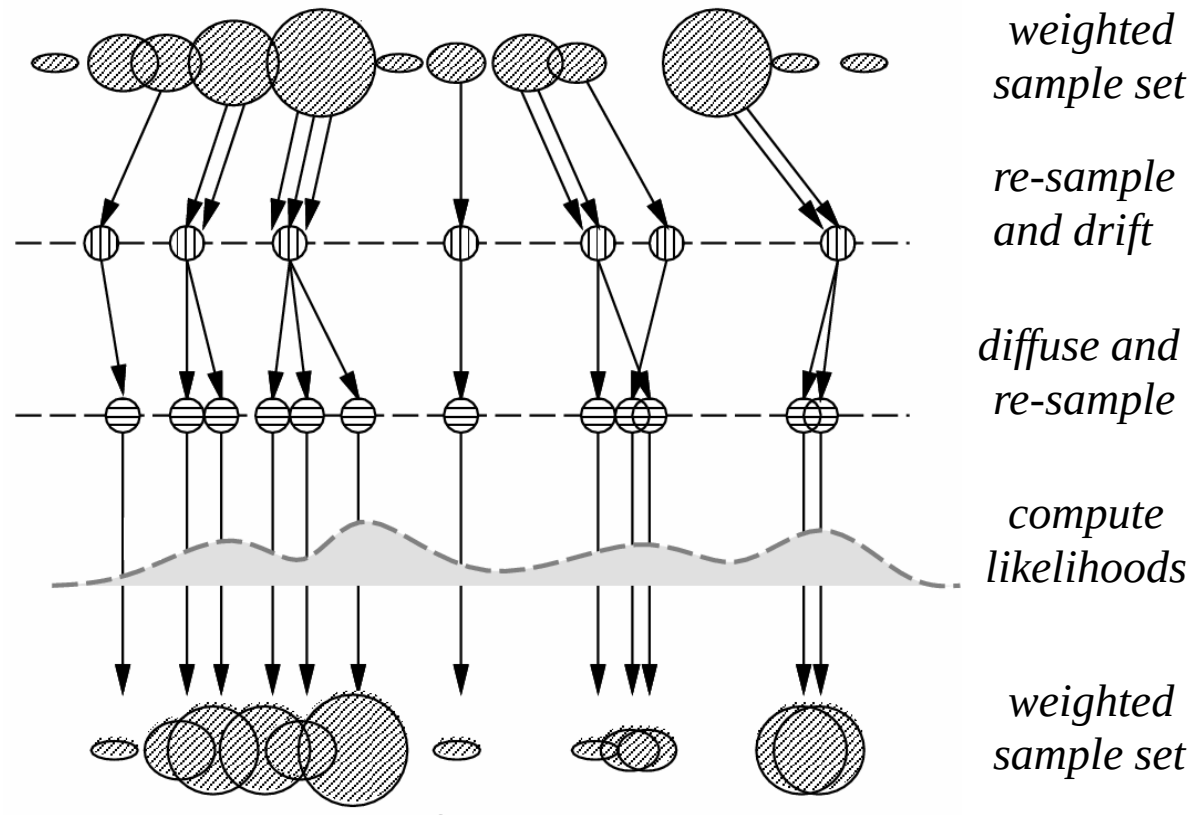
For more background on particle filters see papers by Gordon et al (1998), Isard and Blake (IJCV, 1998), and by Fearnhead (Phd) and Liu and Chen (JASA, 1998).

Particle Filter Steps

Summary of main steps in basic particle filter:

	sample		*sample*		*normalize*	
$p(\mathbf{x}_{t-1} \mid \mathbf{z}_{1:t-1})$	$\longrightarrow$	$p(\mathbf{x}_t \mid \mathbf{x}_{t-1})$	$\longrightarrow$	$p(\mathbf{z}_t \mid \mathbf{x}_t)$	$\longrightarrow$	$p(\mathbf{x}_t \mid \mathbf{z}_{1:t})$
filtering distribution		temporal dynamics		likelihood evaluation		filtering distribution

Depiction of the particle filter process (after *[Isard and Blake, '98]*):



Particle Filter Example

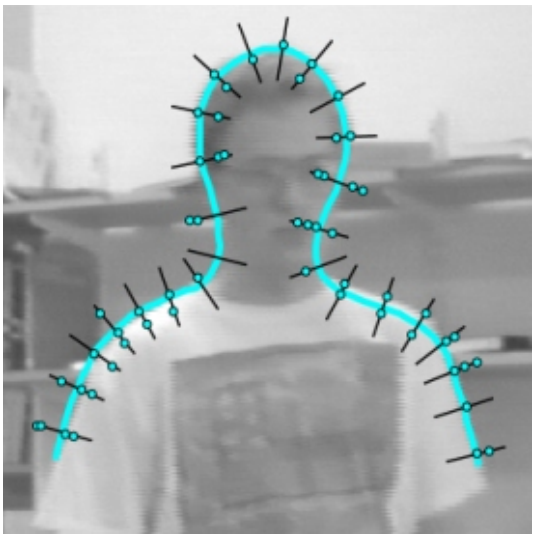
State: 6 DOF affine motion

Measurement: edge locations normal to contour

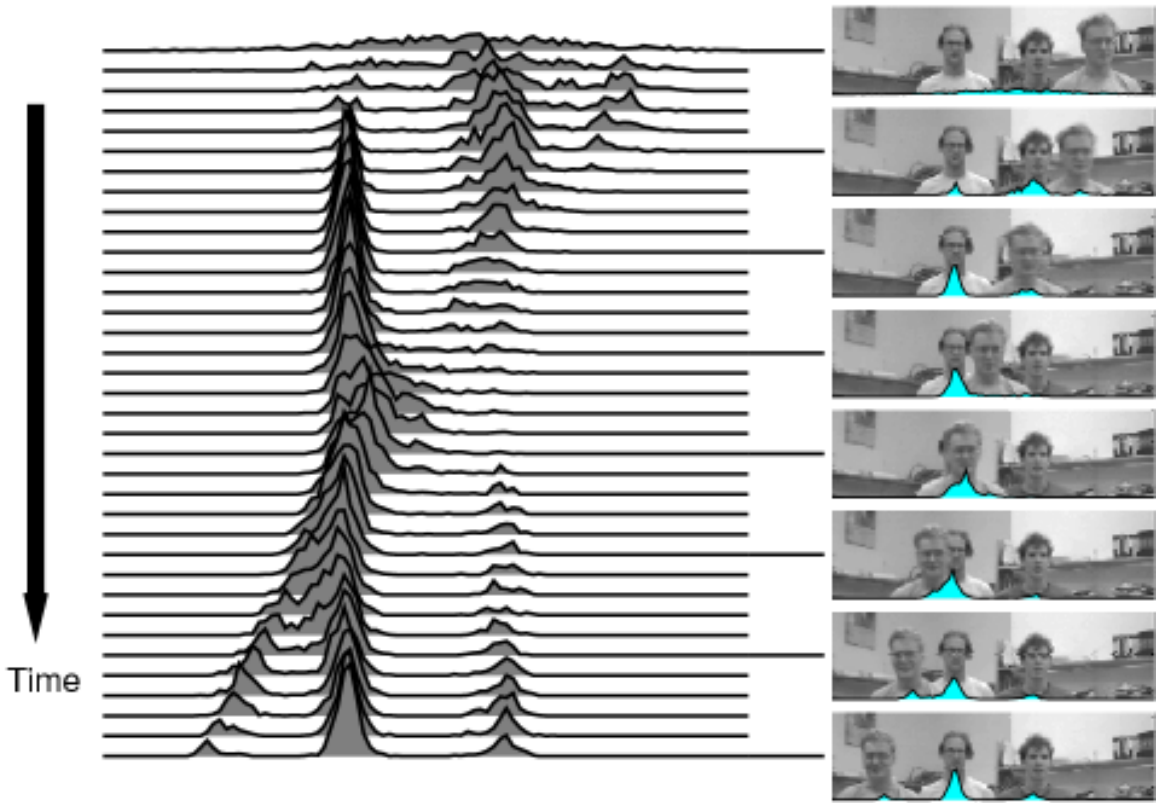
Dynamics: second-order Markov

Computation: 1000 particles with re-sampling every frame

[Isard and Blake, '98]



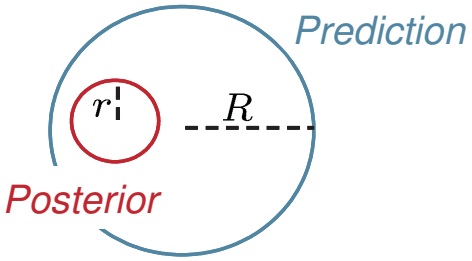
Depiction of the filtering distribution evolving through time.



Particle Explosion in High Dimensions

The number of samples N required by our simple particle filter depends on the effective volumes (entropies) of the prediction and posterior distributions.

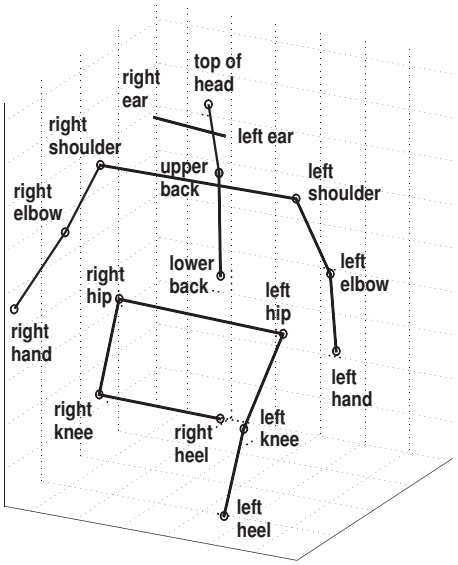
With random sampling from the prediction density, N must grow exponentially in state dimension D if we expect enough samples to fall on states with high posterior probability.



E.g., for D-dimensional spheres, with radii R and r, $N \gg \left(\frac{R}{r}\right)^D$

Example: 3D People Tracking

Goal: Estimate human pose and motion from monocular video.



Model State: 3D kinematic tree with 6 global degrees of freedom and 22 joint angles

Likelihood & Dynamics:

Given the state, **s**, and camera model, 3D marker positions $\mathbf{X}_j$ project onto the 2D image plane to locations

$$\mathbf{d}_j(\mathbf{s}) \;=\; T(\mathbf{X}_j;\, \mathbf{s}) \;.$$

Observation model:

$$\hat{\mathbf{d}}_j \;=\; \mathbf{d}_j + \eta_j \;, \quad \eta_j \;\sim\; \mathcal{N}(0,\, \sigma_m^2 \mathbf{I}_2) \;.$$

Likelihood of observed 2D locations, $\mathbf{D} = \{\hat{\mathbf{d}}_\mathbf{j}\}$:

$$p(\mathbf{D} \,|\, \mathbf{s}) \;\propto\; \exp(-\frac{1}{2\sigma_m^2} \sum_j || \,\hat{\mathbf{d}}_j - \mathbf{d}_j(\mathbf{s}) \,||^2) \;.$$

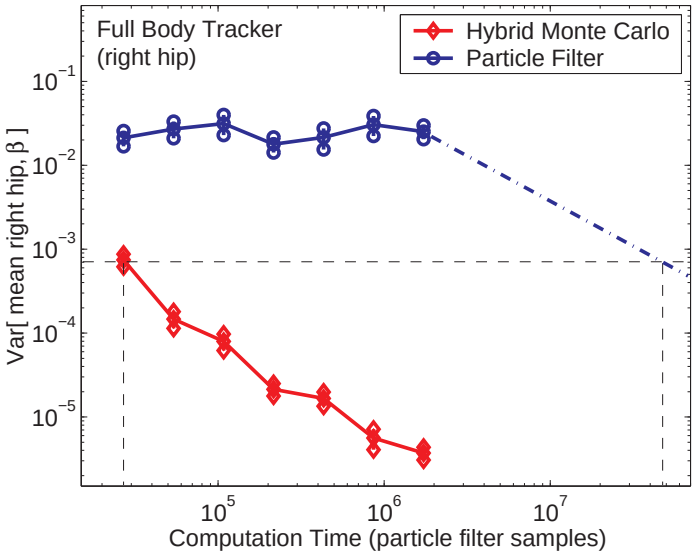
Smooth dynamics:

$$\mathbf{s}_t \;=\; \mathbf{s}_{t-1} \;+\; \epsilon_t \;.$$

where ϵ_t is isotropic Gaussian for translational and angular variables.

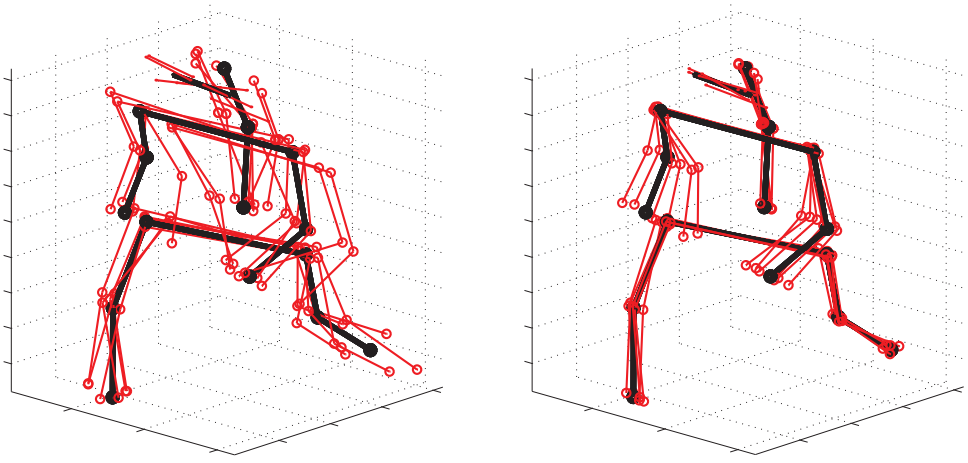
Example: 3D People Tracking (cont)

Estimator Variance: expected squared error (from posterior mean), computed over multiple runs with independent noise. With N fair posterior samples the estimator variance will decrease like $1/N$.



The estimator variance does not decrease anything like $1/N$. Better samplers are necessary.

E.g., *Hybrid Monte Carlo Filter [Choo & Fleet 01]:* A particle filter with Markov chain Monte Carlo updates is more efficient.



Particle Filter HMC Filter
(black: ground truth; red: mean states from 6 trials)

Effective Sample Size

If we drew N fair samples from the posterior, then *estimator variance* decreases like $1/N$.

We can approximate the number of "independent" samples (called the effective sample size) as follows:

$$N_e \approx \frac{1}{\sum_j (w^{(j)})^2}$$

In the worst case, when only one weight is significantly non-zero, N_e is close to one. In the best case, with N fair samples, all weights are $1/N$ so $N_e = N$.

In practice,

- when N_e is large (e.g., > 100, but this depends on the task), one should not necessarily re-sample the posterior. You may just propagate each sample forward by sampling from the transition density.
- when N_e is small (e.g., < 10), you likely have an unreliable posterior approximation, and you may lose track of the target. You need more or better samples!

Residual Sampling

Given N samples $\{\mathbf{x}_k\}_{k=1}^N$ with weights $\{w_k\}_{k=1}^N$, where $\sum_k w_k = 1$, and assume that we wish to draw N new samples (with replacement).

Rather than treating the weights are probabilities of a multinomial distribution and drawing N independent samples, one can greatly reduce sampling variability by using *residual sampling*.

- The expected number of times we expect to draw $\mathbf{x}_k$ is $n_k = Nw_k$.
- So first place $\lfloor n_k \rfloor$ copies of $\mathbf{x}_k$ in the new sample set, and let the *residual weights* be $a_k = n_k - \lfloor n_k \rfloor$.
- Then, draw $N - \sum_k \lfloor n_k \rfloor$ samples (with replacement) according to the probabilities $p_k = a_k / \sum_k a_k$.

Use the Current Observation to Improve Proposals

The proposal distribution $\mathcal{Q}$ should be as close as possible to the filtering distribution $\mathcal{P}$ that we wish to approximate. Otherwise,

- many particles will have weights near zero, contributing very little to the approximation to the filtering disitribution;
- we may even fail to sample significant regions of the state space, so the normalization constant c can be wildly wrong.

For visual tracking, the prediction distribution $\mathcal{Q} = p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1})$ often yields very poor proposals, because dynamics are often very uncertain, and likelihoods are often very peaked by comparison.

One way to greatly improve proposals is to use the current observation. For example, imagine that you are tracking faces and you have a low-level face detector.

- Let $\mathcal{D}(\mathbf{x}_t)$ be a continuous distribution obtained from some low-level detector which indicates where faces might be (e.g., Gaussian modes at locations of classifier hits).
- Then, just modify the proposal density and importance weights:

$$\mathcal{Q}(\mathbf{x}_t) \;=\; \mathcal{D}(\mathbf{x}_t)\, p(\mathbf{x}_t \mid \mathbf{z}_{1:t-1}) \;, \quad \text{with} \quad w(\mathbf{x}_t) \;=\; \frac{c\, p(\mathbf{z}_t \mid \mathbf{x}_t)}{\mathcal{D}(\mathbf{x}_t)}$$

Explain All Observations

Don't compare different states based on different sets of observations.

- If one hypothesizes two target locations, $\mathbf{s}_1$ and $\mathbf{s}_2$, and extracts target-sized image regions centered at both locations, I_1 and I_2, it makes no sense to say $\mathbf{s}_1$ is more likely if $p(I_1 \,|\, \mathbf{s}_1) > p(I_2 \,|\, \mathbf{s}_2)$.
- Rather, use $p(I \,|\, \mathbf{s}_1)$ and $p(I \,|\, \mathbf{s}_2)$ where I is the entire image.

Explain the entire image, or use likelihood ratios (for efficiency).
E.g., assume that pixels $I(\mathbf{y})$, given the state, are independent, so

$$p(I \,|\, \mathbf{x}) \;=\; \prod_{\mathbf{y} \in D_f} p_f(I(\mathbf{y}) \,|\, \mathbf{s}) \prod_{\mathbf{y} \in D_b} p_b(I(\mathbf{y}))$$

where D_f and D_b are disjoint sets of foreground and background pixels, and p_f and p_b are the respective likelihood functions.

Divide $p(I \,|\, \mathbf{s})$ by the background likelihood of all pixels (i.e., as if no target is present):

$$\begin{aligned} p(I \,|\, \mathbf{s}) \;&\propto\; \frac{\prod_{\mathbf{y} \in D_f} p_f(I(\mathbf{y}) \,|\, \mathbf{s}) \;\prod_{\mathbf{y} \in D_b} p_b(I(\mathbf{y}))}{\prod_{\mathbf{y}} p_b(I(\mathbf{y}))} \\ &=\; \frac{\prod_{\mathbf{y} \in D_f} p_f(I(\mathbf{y}) \,|\, \mathbf{s}) \;\prod_{\mathbf{y} \in D_b} p_b(I(\mathbf{y}))}{\prod_{\mathbf{y} \in D_f} p_b(I(\mathbf{y}) \,|\, \mathbf{s}) \;\prod_{\mathbf{y} \in D_b} p_b(I(\mathbf{y}))} \\ &=\; \prod_{\mathbf{y} \in D_f} \frac{p_f(I(\mathbf{y}) \,|\, \mathbf{x})}{p_b(I(\mathbf{y}))} \end{aligned}$$

Conditional Observation Independence

What about independence of measurements?

Pixels have correlated (dependent) noise because of several factors:

- models are wrong (most noise is model error)
- failures in feature tracking are not independent for different features.
- overlapping windows are often used to extract measurements

Consequence: Likelihood functions are often more sharply peaked than they ought to be.

Summary and Further Remarks on Filtering

- posteriors must be sufficiently constrained, with some combinations of posterior factorization, dynamics, measurements, ..
- proposal distributions should be non-zero wherever the posterior distribution is non-zero (usually heavy-tailed)
- proposals should exploit current observations in addition to prediction distribution
- likelihoods should be compared against the same observations
- sampling variability can be a problem
  - must have enough samples in regions of high probability for normalization to be useful
  - too many samples needed for high dimensional problems (esp. when samples drawn independently from prediction dist)
  - samples tend to migrate to a single mode (don't design a particle filter to track multiple objects with a state that represents only one such object)
  - sample deterministically where possible
  - exploit diagnostics to monitor effective numbers of samples

Further Readings

Arulampalam, M.S., Maskell, S., Gordon, N. and Clapp, T. (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Proc.* 50(2):174–188

Black, M.J. and Jepson, A.D. (1996) Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *Int. J. Computer Vision*, 26:63–84.

Blake, A. (2005) Visual tracking. In *Mathematical models for Computer Vision: The Handbook*. N. Paragios, Y. Chen, and O. Faugeras (eds.), Springer, 2005.

Carpenter, Clifford and Fearnhead (1999) An improved particle filter for nonlinear problems. IEE Proc. Radar Sonar Navig., 146(1)

Doucet, A., Godsill, S., and Andrieu, C. (2000) On sequential Monte Carlo sampling methods for Bayesian filtering. *Stats and Computing*, 10:197–208.

Gordon, N.J., Salmond, D.J. and Smith, A.F.M. (1993) Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113.

Isard, M. and Blake, A. (1998) Condensation: Conditional density propagation. *Int. J. Computer Vision*, 29(1):2–28.

Jepson, A.D., Fleet, D.J. and El-Maraghi, T. (2003) Robust, on-line appearance models for visual tracking. *IEEE Trans. on PAMI*, 25(10):1296–1311

Julier, S.J. and Uhlmann, J.K. (2004) Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422

Khan, Z., Balch, T. and Dellaert, F. (2004) A Rao-Blackwellized particle filter for EigenTracking. *Proc. IEEE CVPR*.

Sidenbladh, H., Black, M.J. and Fleet, D.J. (2000) Stochastic tracking of 3D human figures using 2D image motion. *Proc. ECCV*, pp. 702–718, Dublin, Springer.

Wan, E.A. and van der Merwe, R. (2000) The unscented Kalman filter for nonlinear estimation. (see van der Merwe's web site for the TR)