# C280, Computer Vision

Prof. Trevor Darrell
trevor@eecs.berkeley.edu

Lecture 19: Tracking

# Tracking scenarios

- Follow a point
- Follow a template
- Follow a changing template
- Follow all the elements of a moving person, fit a model to it.

# Things to consider in tracking

What are the dynamics of the thing being tracked?

How is it observed?

# Three main issues in tracking

- **Prediction:** we have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$ — what state does this set of measurements predict for the $i$'th frame? to solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.
- **Data association:** Some of the measurements obtained from the $i$-th frame may tell us about the object's state. Typically, we use $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$ to identify these measurements.
- **Correction:** now that we have $\boldsymbol{y}_i$ — the relevant measurements — we need to compute a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$.

# Simplifying Assumptions

- **Only the immediate past matters:** formally, we require

$$P(\boldsymbol{X}_i|\boldsymbol{X}_1,\ldots,\boldsymbol{X}_{i-1}) = P(\boldsymbol{X}_i|\boldsymbol{X}_{i-1})$$

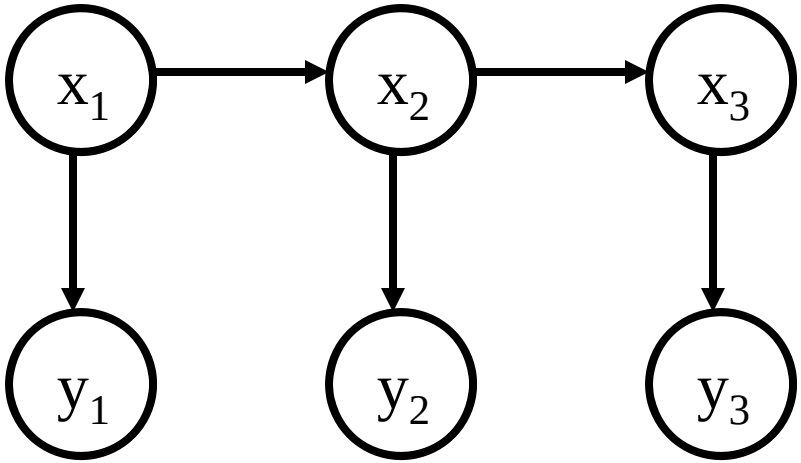
  This assumption hugely simplifies the design of algorithms, as we shall see; furthermore, it isn't terribly restrictive if we're clever about interpreting $\boldsymbol{X}_i$ as we shall show in the next section.

- **Measurements depend only on the current state:** we assume that $\boldsymbol{Y}_i$ is conditionally independent of all other measurements given $\boldsymbol{X}_i$. This means that

$$P(\boldsymbol{Y}_i,\boldsymbol{Y}_j,\ldots\boldsymbol{Y}_k|\boldsymbol{X}_i) = P(\boldsymbol{Y}_i|\boldsymbol{X}_i)P(\boldsymbol{Y}_j,\ldots,\boldsymbol{Y}_k|\boldsymbol{X}_i)$$

  Again, this isn't a particularly restrictive or controversial assumption, but it yields important simplifications.

# Kalman filter graphical model and corresponding factorized joint probability




$$P(x_1, x_2, x_3, y_1, y_2, y_3) =$$

$$P(x_1)P(y_1 \mid x_1)P(x_2 \mid x_1)P(y_2 \mid x_2)P(x_3 \mid x_2)P(y_3 \mid x_3)$$

# Tracking as induction

- Make a measurement starting in the $0^{th}$ frame
- Then:  assume you have an estimate at the ith frame, after the measurement step.
- Show that you can do prediction for the i+1th frame, and measurement for the i+1th frame.

# Base case

Firstly, we assume that we have $P(\boldsymbol{X}_0)$

$$P(\boldsymbol{X}_0|\boldsymbol{Y}_0 = \boldsymbol{y}_0) = \frac{P(\boldsymbol{y}_0|\boldsymbol{X}_0)P(\boldsymbol{X}_0)}{P(\boldsymbol{y}_0)}$$

$$\propto P(\boldsymbol{y}_0|\boldsymbol{X}_0)P(\boldsymbol{X}_0)$$

# Prediction step

**Prediction**

Prediction involves representing

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})$$

given

$$P(\boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1}).$$

Our independence assumptions make it possible to write

$$\begin{aligned} P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1}) &= \int P(\boldsymbol{X}_i,\boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})d\boldsymbol{X}_{i-1} \\ &= \int P(\boldsymbol{X}_i|\boldsymbol{X}_{i-1},\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})P(\boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})d\boldsymbol{X}_{i-1} \\ &= \int P(\boldsymbol{X}_i|\boldsymbol{X}_{i-1})P(\boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})d\boldsymbol{X}_{i-1} \end{aligned}$$

# Update step

**Correction**

Correction involves obtaining a representation of

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)$$

given

$$P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})$$

Our independence assumptions make it possible to write

$$\begin{aligned}
P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i) &= \frac{P(\boldsymbol{X}_i,\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i,\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\
&= P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})\frac{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{P(\boldsymbol{y}_0,\ldots,\boldsymbol{y}_i)} \\
&= \frac{P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})}{\int P(\boldsymbol{y}_i|\boldsymbol{X}_i)P(\boldsymbol{X}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})d\boldsymbol{X}_i}
\end{aligned}$$

# The Kalman Filter

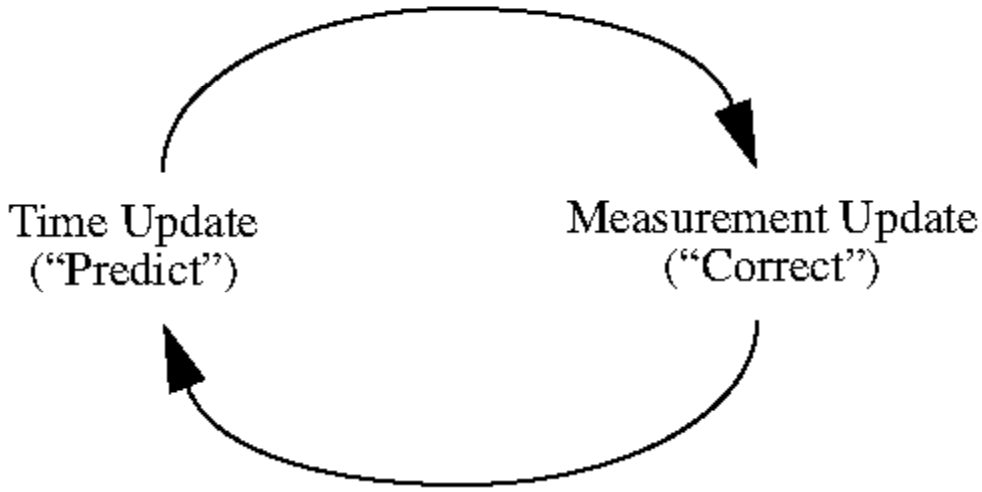
- Key ideas:
  - Linear models interact uniquely well with Gaussian noise - make the prior Gaussian, everything else Gaussian and the calculations are easy
  - Gaussians are really easy to represent --- once you know the mean and covariance, you’re done

# Recall the three main issues in tracking

- **Prediction:** we have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$ — what state does this set of measurements predict for the $i$'th frame? to solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.
- **Data association:** Some of the measurements obtained from the $i$-th frame may tell us about the object's state. Typically, we use $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$ to identify these measurements.
- **Correction:** now that we have $\boldsymbol{y}_i$ — the relevant measurements — we need to compute a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_i = \boldsymbol{y}_i)$.

*(Ignore data association for now)*

# The Kalman Filter






[figure from http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html]

# The Kalman Filter in 1D

- Dynamic Model

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2)$$

$$y_i \sim N(m_i x_i, \sigma_{m_i}^2)$$

- Notation

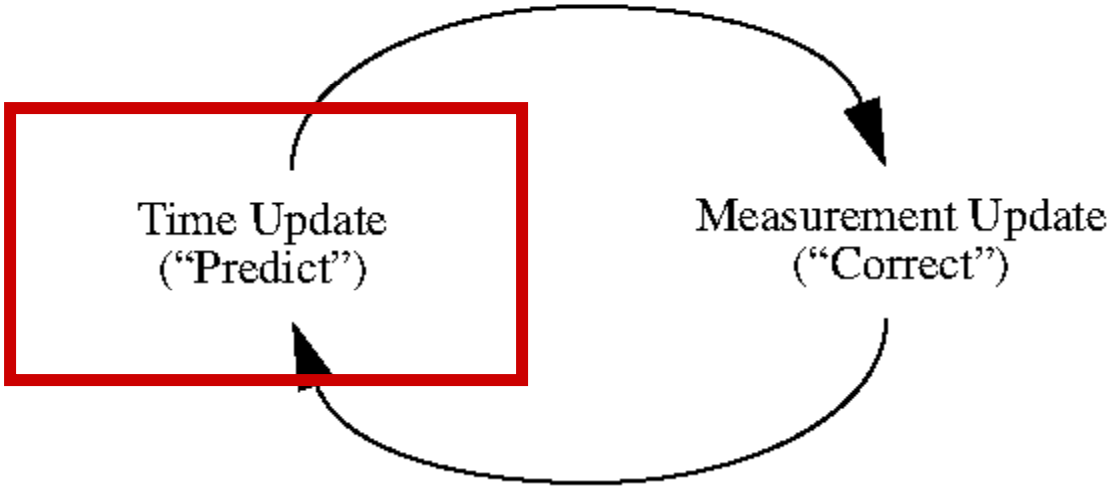
mean of $P(X_i|y_0, \ldots, y_{i-1})$ as $\overline{X}_i^-$ ← Predicted mean

mean of $P(X_i|y_0, \ldots, y_i)$ as $\overline{X}_i^+$ ← Corrected mean

the standard deviation of $P(X_i|y_0, \ldots, y_{i-1})$ as $\sigma_i^-$

of $P(X_i|y_0, \ldots, y_i)$ as $\sigma_i^+$

# The Kalman Filter



# Prediction for 1D Kalman filter

- The new state is obtained by
  - multiplying old state by known constant
  - adding zero-mean noise

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i}^2)$$

- Therefore, predicted mean for new state is
  - constant times mean for old state
- Old variance is normal random variable
  - variance is multiplied by square of constant
  - and variance of noise is added.

$$\overline{X}_i^- = d_i \overline{X}_{i-1}^+ \qquad (\sigma_i^-)^2 = \sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2$$

**Dynamic Model:**

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

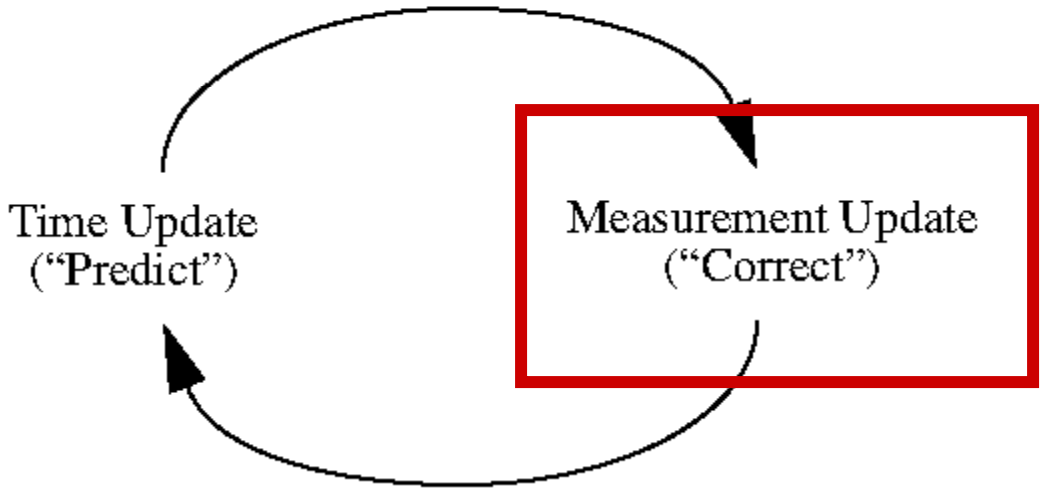
$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

**Start Assumptions:** $\overline{x}_0^-$ and $\sigma_0^-$ are known
**Update Equations:** Prediction

$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

# The Kalman Filter



# Measurement update for 1D Kalman filter

$$x_i^+ = \left( \frac{\overline{x_i^-}\,\sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left( \frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

Notice:

- if measurement noise is small,

  we rely mainly on the measurement,
- if it's large, mainly on the prediction
- $\sigma$ does not depend on y

**Dynamic Model:**

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

**Start Assumptions:** $\overline{x_0}^-$ and $\sigma_0^-$ are known
**Update Equations:** Prediction

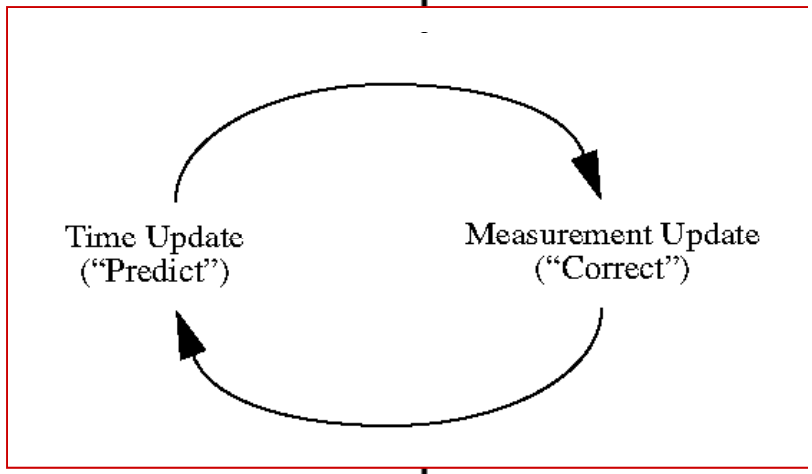
$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

**Update Equations:** Correction

$$x_i^+ = \left( \frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left( \frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$



# Kalman filter for computing an on-line average

- What Kalman filter parameters and initial conditions should we pick so that the optimal estimate for x at each iteration is just the average of all the observations seen so far?

## Kalman filter model

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

$$d_i = 1,\ m_i = 1,\ \sigma_{d_i} = 0,\ \sigma_{m_i} = 1$$

## Initial conditions

$$\overline{x}_0^- = 0 \qquad \sigma_0^- = \infty$$

$\overline{x}_0^-$ and $\sigma_0^-$ are known

Prediction

$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Correction

$$x_i^+ = \left( \frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left( \frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

| Iteration | 0 | 1 | 2 |
|---|---|---|---|
| $\overline{x}_i^-$ | 0 | $y_0$ | $\frac{y_0 + y_1}{2}$ |
| $\overline{x}_i^+$ | $y_0$ | $\frac{y_0 + y_1}{2}$ | $\frac{y_0 + y_1 + y_2}{3}$ |
| $\sigma_i^-$ | $\infty$ | 1 | $\frac{1}{\sqrt{2}}$ |
| $\sigma_i^+$ | 1 | $\frac{1}{\sqrt{2}}$ | $\frac{1}{\sqrt{3}}$ |

What happens if the x dynamics are given a non-zero variance?

## Kalman filter model

$$x_i \sim N(d_i x_{i-1}, \sigma_{d_i})$$

$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

$$d_i = 1,\ m_i = 1,\ \sigma_{d_i} = 1,\ \sigma_{m_i} = 1$$

Initial conditions

$$\overline{x}_0^- = 0 \qquad \sigma_0^- = \infty$$

$\overline{x}_0^-$ and $\sigma_0^-$ are known

Prediction

$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Correction

$$x_i^+ = \left( \frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

$$\sigma_i^+ = \sqrt{\left( \frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$

| Iteration | 0 | 1 | 2 |
|---|---|---|---|
| $\overline{x}_i^-$ | 0 | $y_0$ | $\frac{y_0 + 2y_1}{3}$ |
| $\overline{x}_i^+$ | $y_0$ | $\frac{y_0 + 2y_1}{3}$ | $\frac{y_0 + 2y_1 + 5y_2}{8}$ |
| $\sigma_i^-$ | $\infty$ | $\sqrt{2}$ | $\sqrt{\frac{5}{3}}$ |
| $\sigma_i^+$ | 1 | $\sqrt{\frac{2}{3}}$ | $\sqrt{\frac{5}{8}}$ |

# Linear dynamic models

- A linear dynamic model has the form

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- This is much, much more general than it looks, and extremely powerful

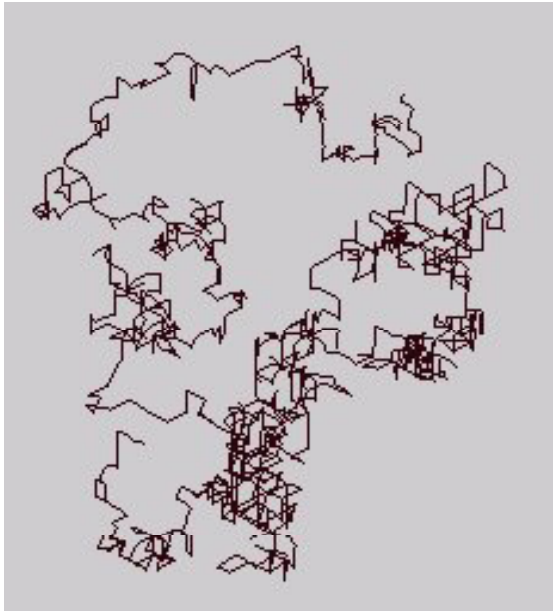
# Examples of linear state space models

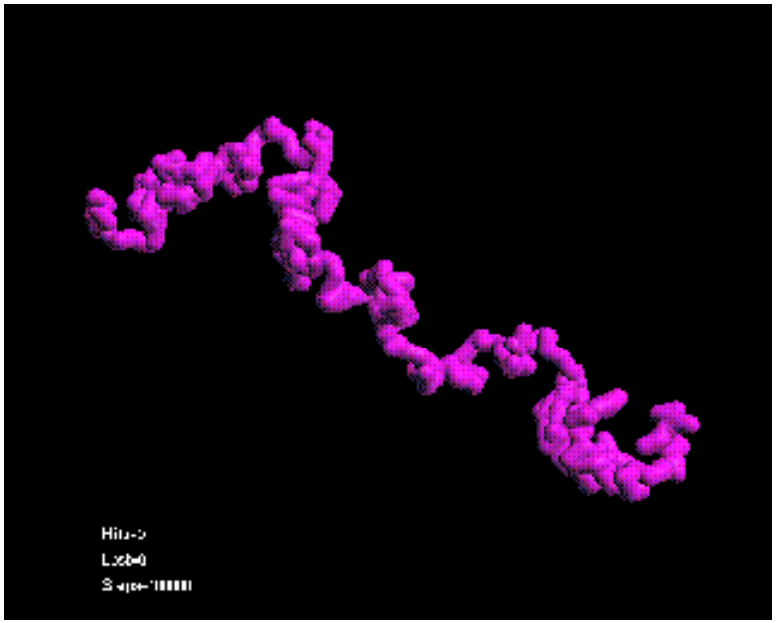
$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- Drifting points
  - assume that the new position of the point is the old one, plus noise

**D = Identity**





cic.nist.gov/lipman/sciviz/images/random3.gif
http://www.grunch.net/synergetics/images/random3.jpg

# Constant velocity

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- We have

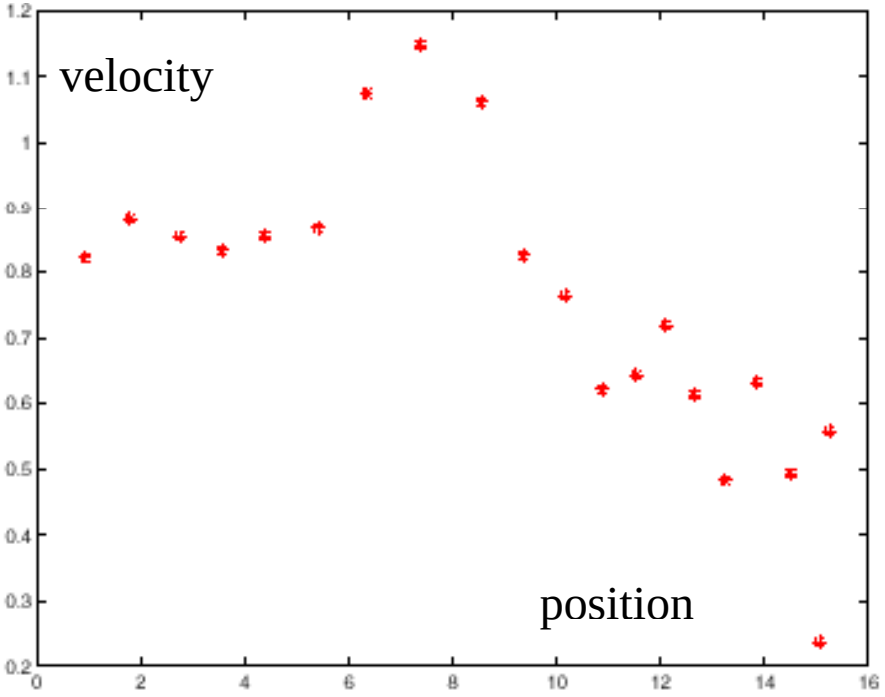
$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$
$$v_i = v_{i-1} + \varsigma_i$$

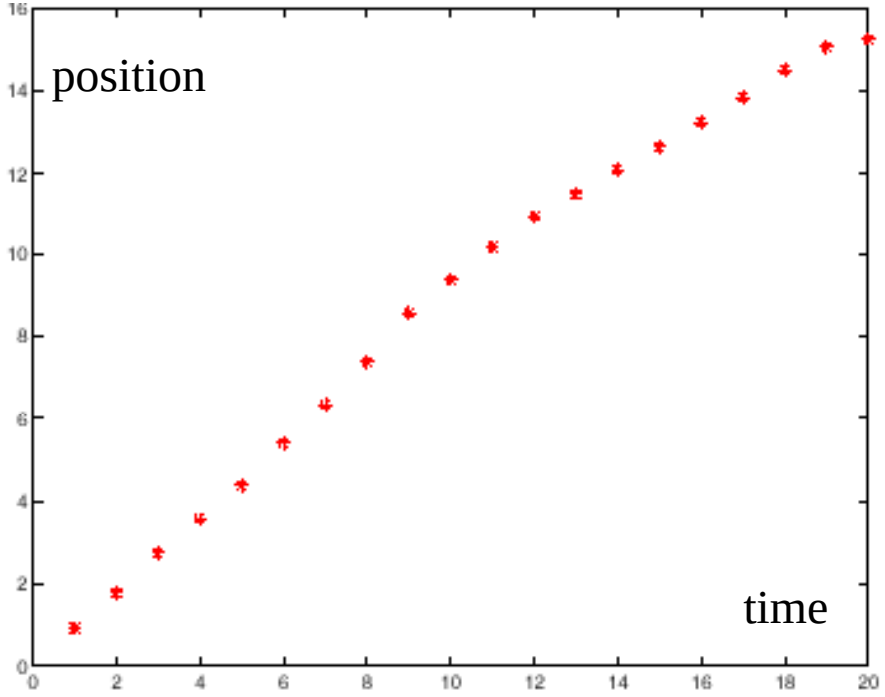
  - (the Greek letters denote noise terms)
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_{i-1} + \text{noise}$$

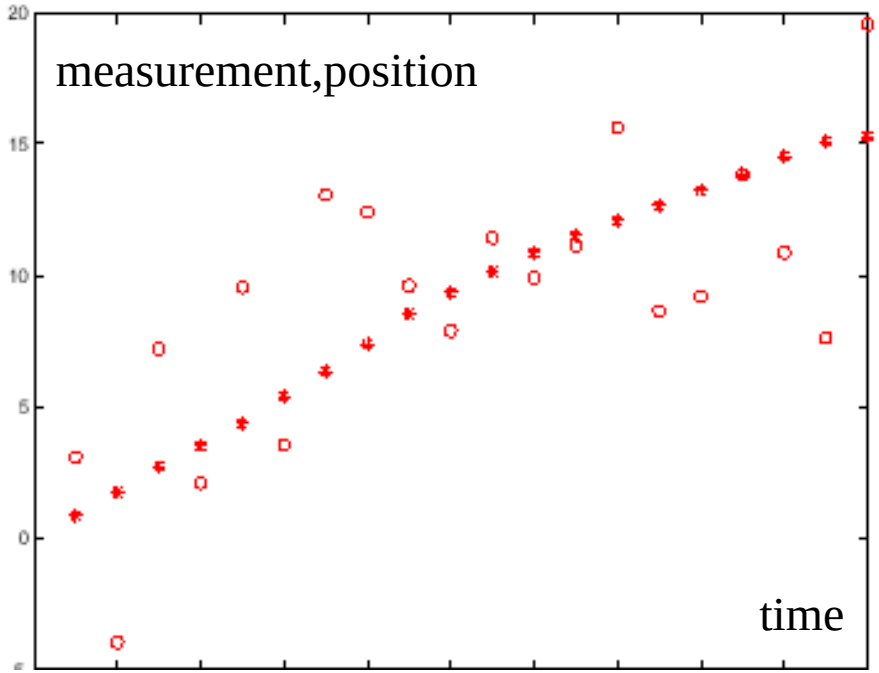
  - which is the form we had above

$\mathbf{x_i}$ $\mathbf{D_{i-1}}$ $\mathbf{x_{i-1}}$




Constant Velocity Model



# Constant acceleration

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

- We have

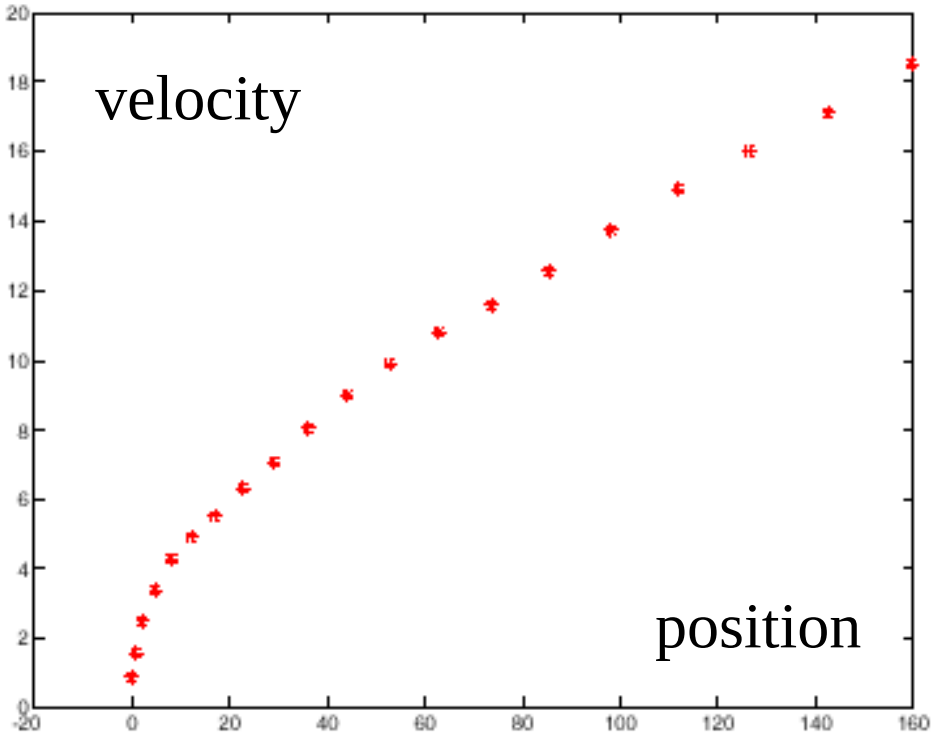
$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$
$$v_i = v_{i-1} + \Delta t a_{i-1} + \varsigma_i$$
$$a_i = a_{i-1} + \xi_i$$

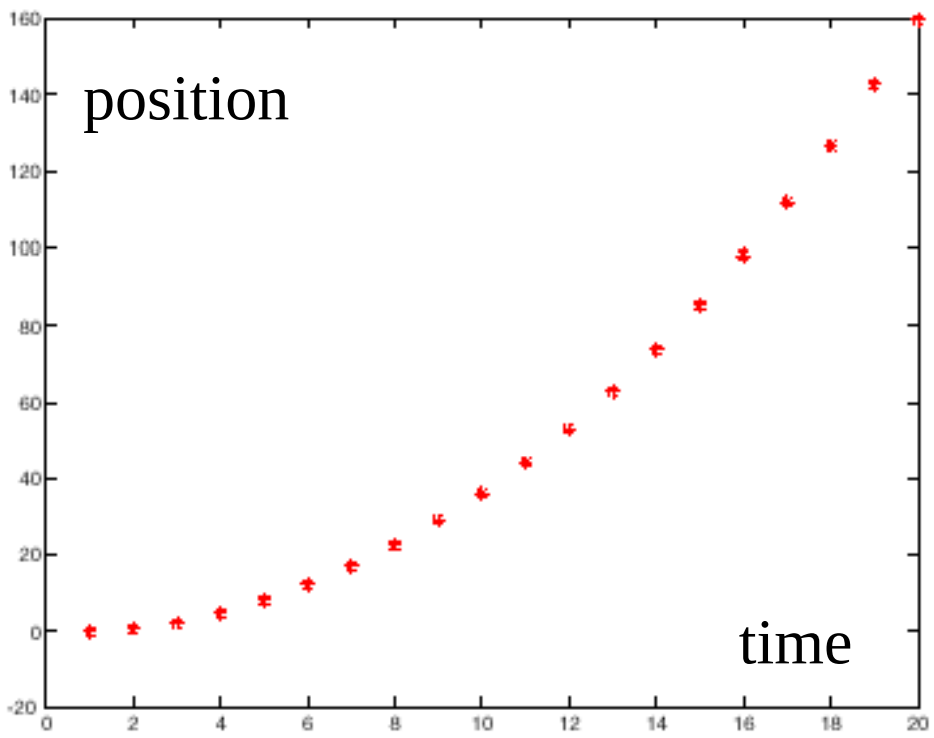
  - (the Greek letters denote noise terms)
- Stack (u, v) into a single state vector

$$\begin{pmatrix} u \\ v \\ a \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ a \end{pmatrix}_{i-1} + \text{noise}$$

  - which is the form we had above

$\mathbf{x_i}$ $\mathbf{D_{i-1}}$ $\mathbf{x_{i-1}}$




Constant Acceleration Model

# Periodic motion

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$
$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

Assume we have a point, moving on a line with a periodic movement defined with a differential eq:

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

can be defined as

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{u} = \mathcal{S}\boldsymbol{u}$$

with state defined as stacked position and velocity u=(p, v)

# Periodic motion

$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1}; \Sigma_{d_i}\right)$$

$$\mathbf{y}_i = N\left(\mathbf{M}_i\mathbf{x}_i; \Sigma_{m_i}\right)$$

$$\frac{d\boldsymbol{u}}{dt} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \boldsymbol{u} = \mathcal{S}\boldsymbol{u}$$

Take discrete approximation….(e.g., forward Euler integration with $\Delta t$ stepsize.)

$$\begin{aligned} \boldsymbol{u}_i &= \boldsymbol{u}_{i-1} + \Delta t \frac{d\boldsymbol{u}}{dt} \\ &= \boldsymbol{u}_{i-1} + \Delta t \mathcal{S}\boldsymbol{u}_{i-1} \\ &= \begin{pmatrix} 1 & \Delta t \\ -\Delta t & 1 \end{pmatrix} \boldsymbol{u}_{i-1} \end{aligned}$$

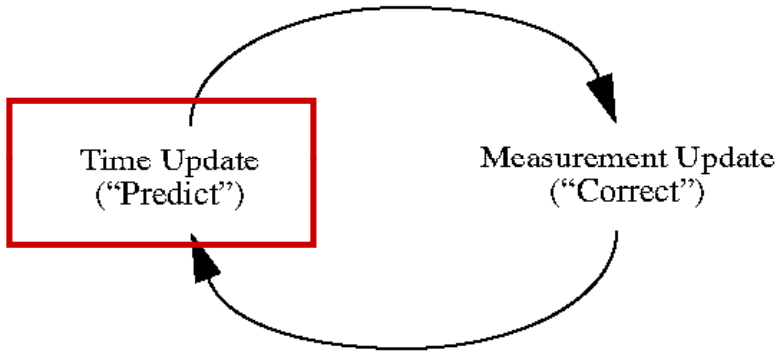
$\mathbf{x_i}$ $\mathbf{D_{i-1}}$ $\mathbf{x_{i-1}}$

# n-D

Generalization to n-D is straightforward but more complex.

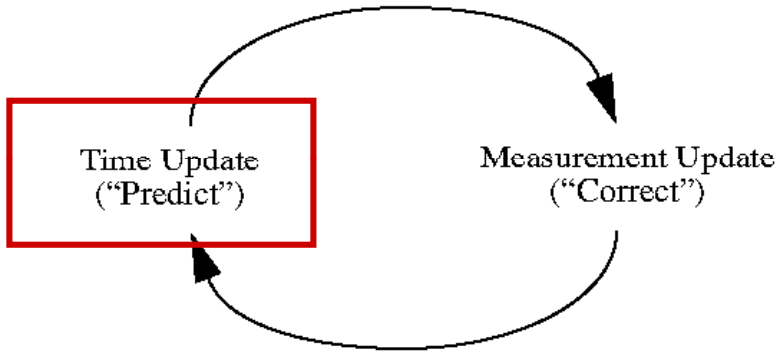
# n-D

Generalization to n-D is straightforward but more complex.



# n-D Prediction

Generalization to n-D is straightforward but more complex.




Prediction:

- Multiply estimate at prior time with forward model:

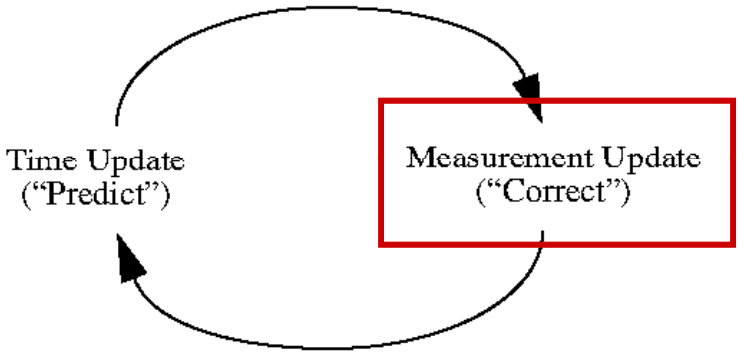
$$\overline{\boldsymbol{x}}_i^- = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

- Propagate covariance through model and add new noise:

$$\Sigma_i^- = \Sigma_{d_i} + \mathcal{D}_i \sigma_{i-1}^+ \mathcal{D}_i$$

# n-D Correction

Generalization to n-D is straightforward but more complex.




Correction:

- Update *a priori* estimate with measurement to form *a posteriori*

# n-D correction

Find linear filter on innovations

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[ \boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^- \right]$$

which minimizes *a posteriori* error covariance:

$$E\left[\left(x-\overline{x^+}\right)^T\left(x-\overline{x^+}\right)\right]$$

K is the *Kalman Gain* matrix. A solution is

$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[ \mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i} \right]^{-1}$$

# Kalman Gain Matrix

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[ \boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^- \right]$$

$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[ \mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i} \right]^{-1}$$

As measurement becomes more reliable, K weights residual more heavily,

$$\lim_{\Sigma_m \to 0} K_i = M^{-1}$$

As prior covariance approaches 0, measurements are ignored:

$$\lim_{\Sigma_i^- \to 0} K_i = 0$$

**Dynamic Model:**

$$\boldsymbol{x}_i \sim N(\mathcal{D}_i \boldsymbol{x}_{i-1}, \Sigma_{d_i})$$

$$\boldsymbol{y}_i \sim N(\mathcal{M}_i \boldsymbol{x}_i, \Sigma_{m_i})$$

**Start Assumptions:** $\overline{\boldsymbol{x}}_0^-$ and $\Sigma_0^-$ are known
**Update Equations:** Prediction

$$\overline{\boldsymbol{x}}_i^- = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

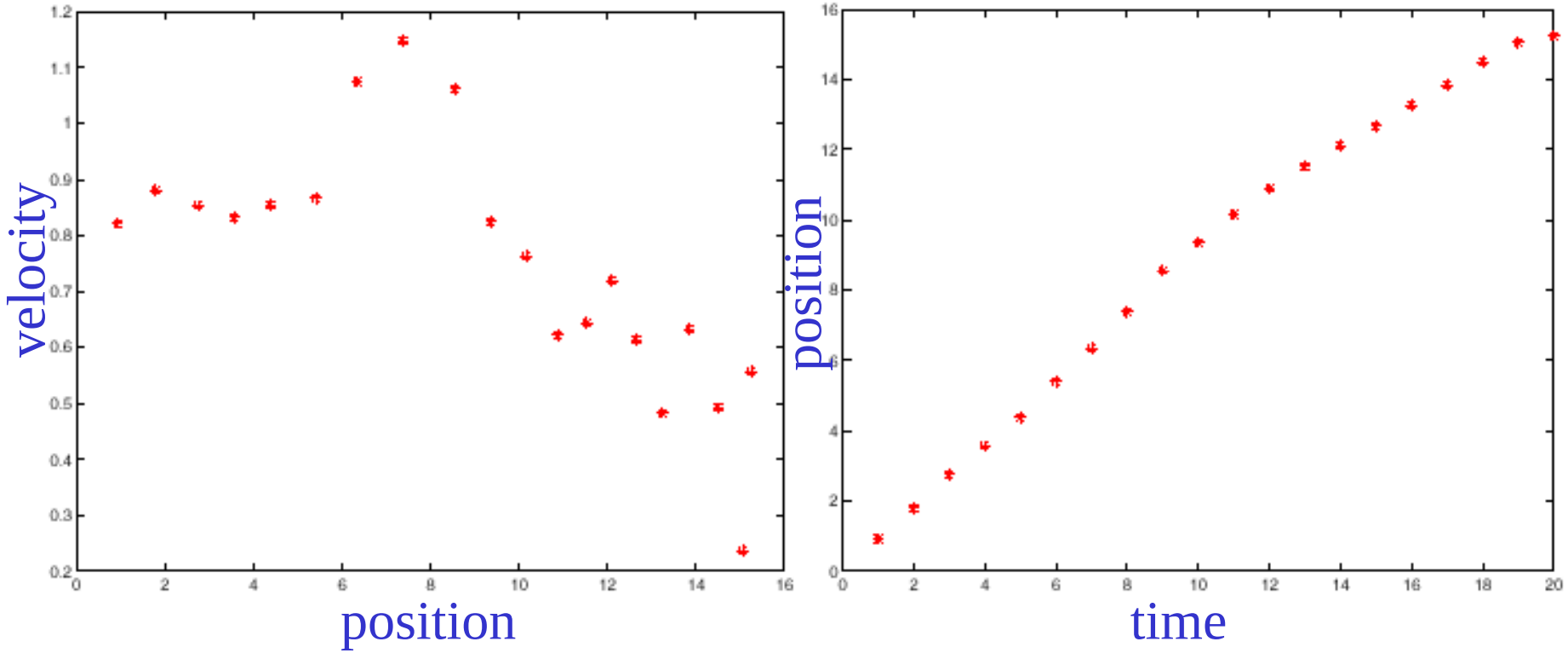
$$\Sigma_i^- = \Sigma_{d_i} + \mathcal{D}_i \sigma_{i-1}^+ \mathcal{D}_i$$

**Update Equations:** Correction

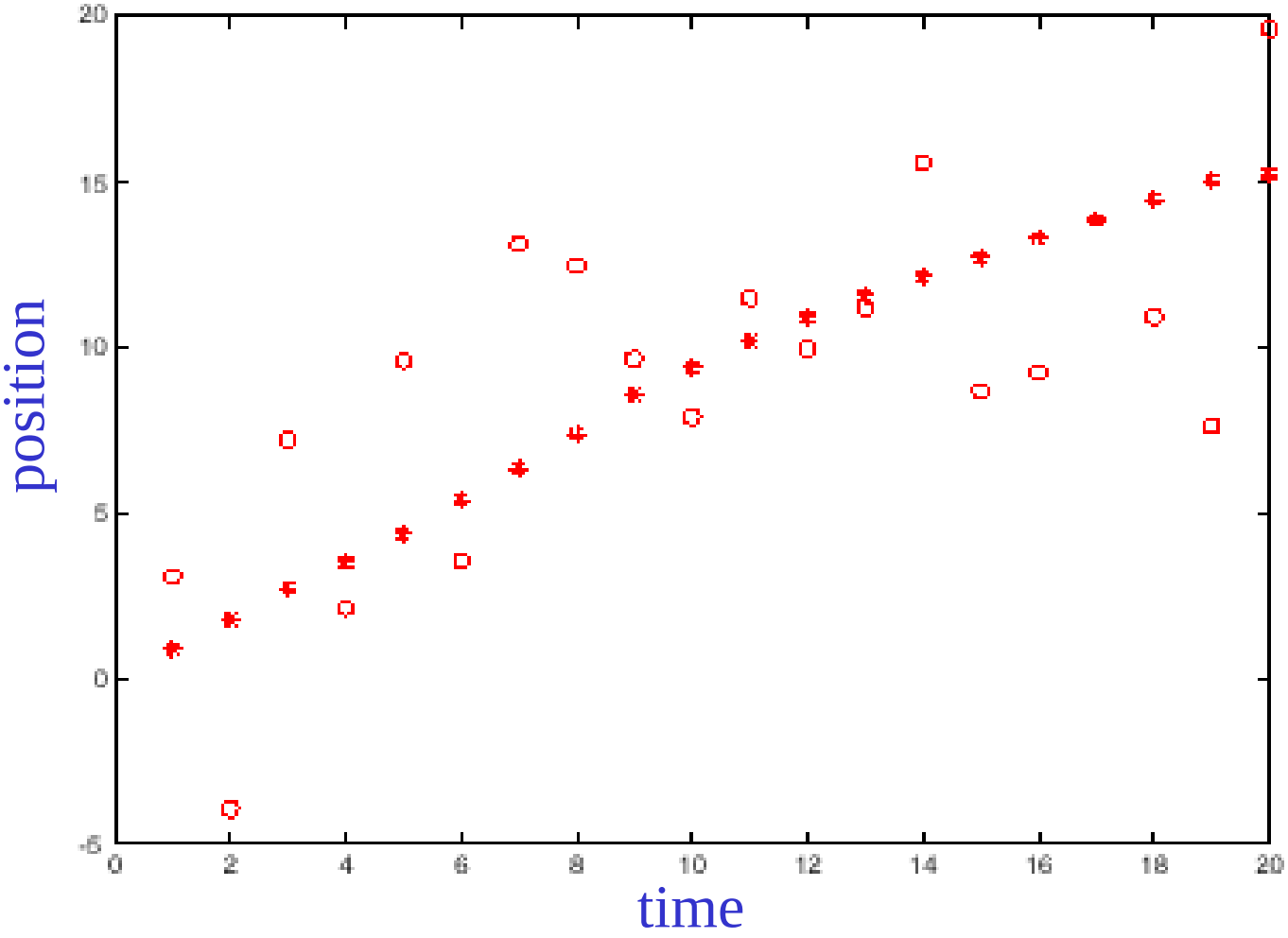
$$\mathcal{K}_i = \Sigma_i^- \mathcal{M}_i^T \left[\mathcal{M}_i \Sigma_i^- \mathcal{M}_i^T + \Sigma_{m_i}\right]^{-1}$$

$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^-\right]$$

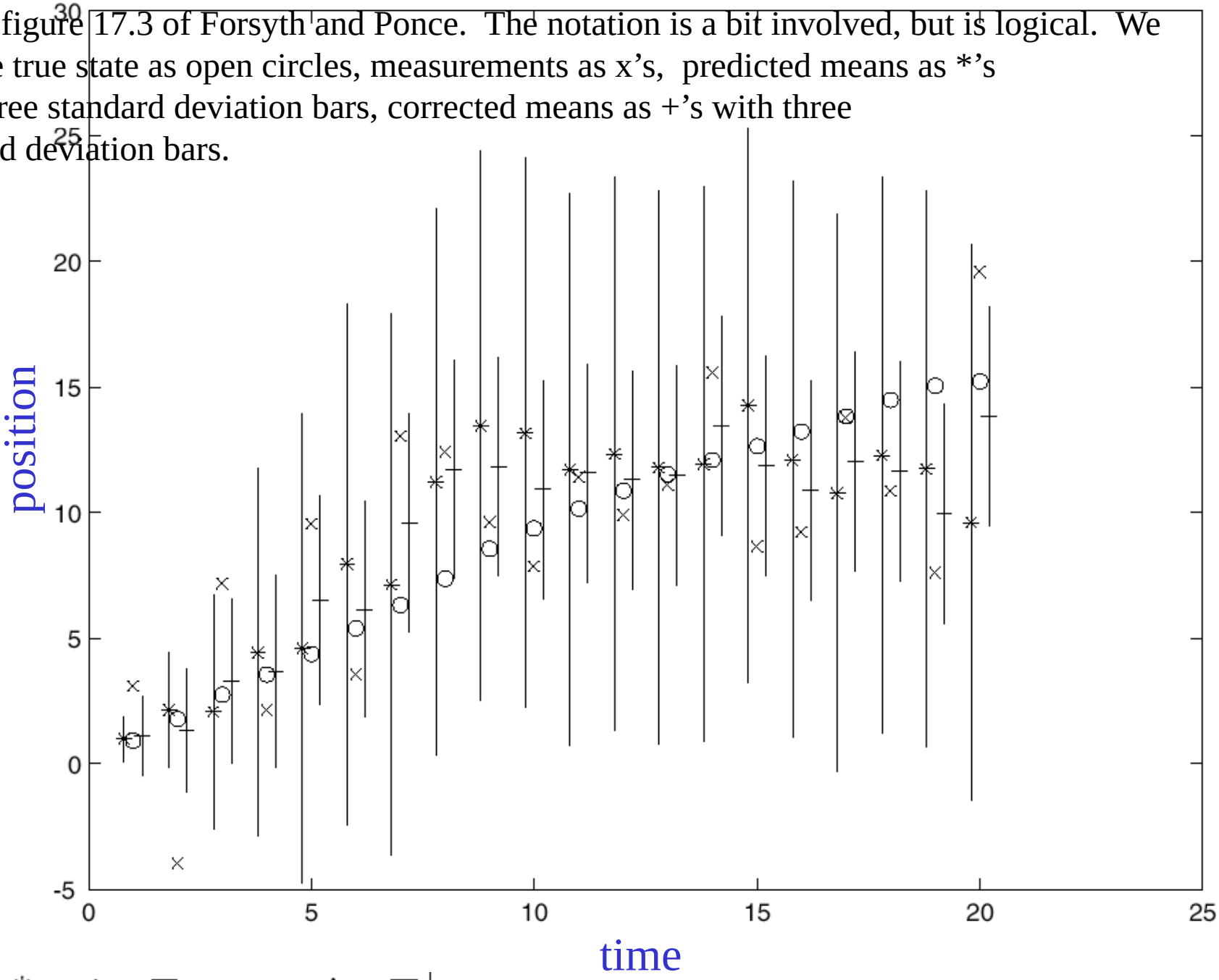
$$\Sigma_i^+ = \left[Id - \mathcal{K}_i \mathcal{M}_i\right] \Sigma_i^-$$



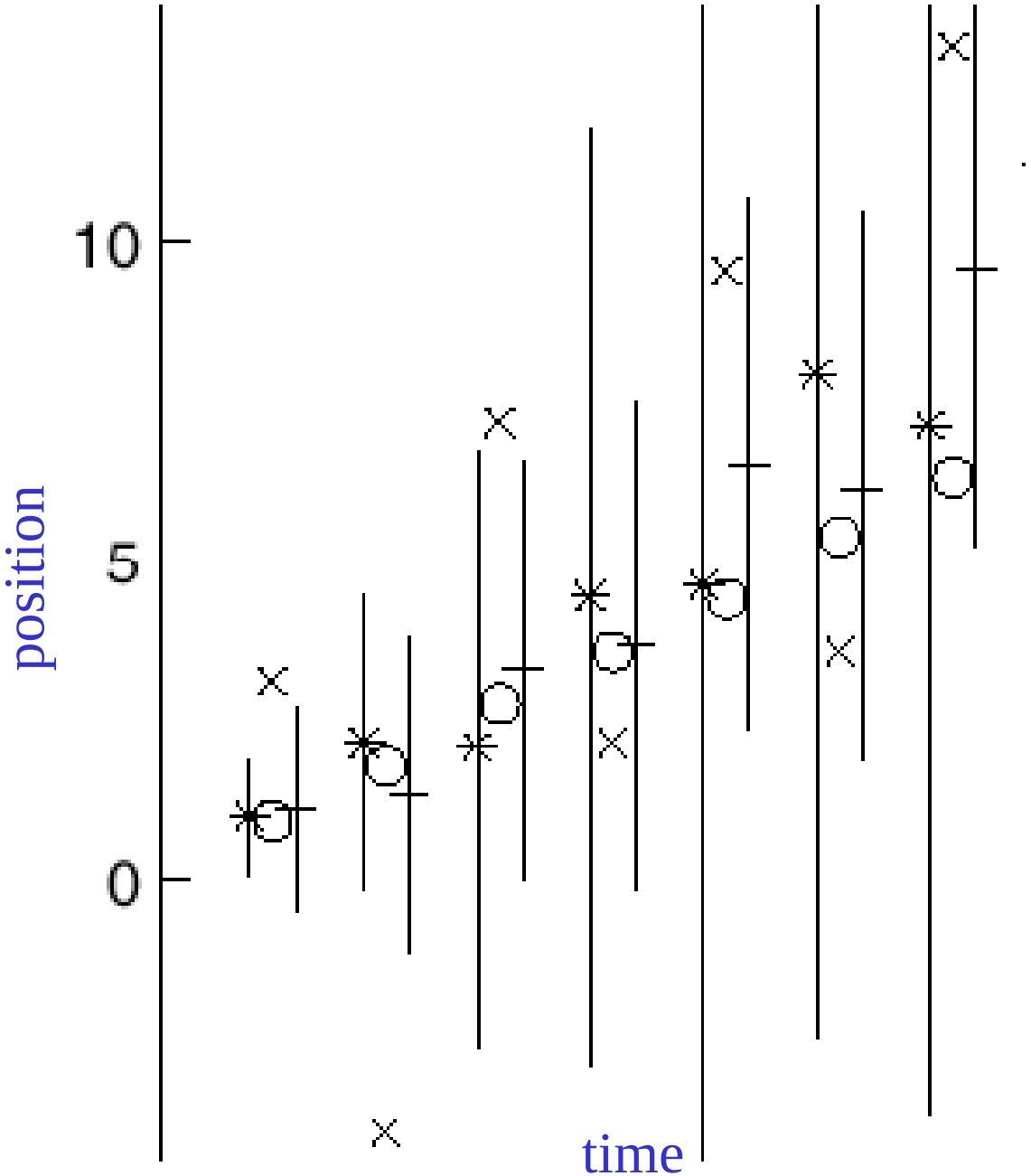

Constant Velocity Model


position
time
20
15
10
5
0
-5
0
2
4
6
8
10
12
14
16
18
20

This is figure 17.3 of Forsyth and Ponce. The notation is a bit involved, but is logical. We plot the true state as open circles, measurements as x's, predicted means as *'s with three standard deviation bars, corrected means as +'s with three standard deviation bars.




The *-s give $\overline{\boldsymbol{x}}_i^-$ , +-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars




The o-s give state, x-s measurement.

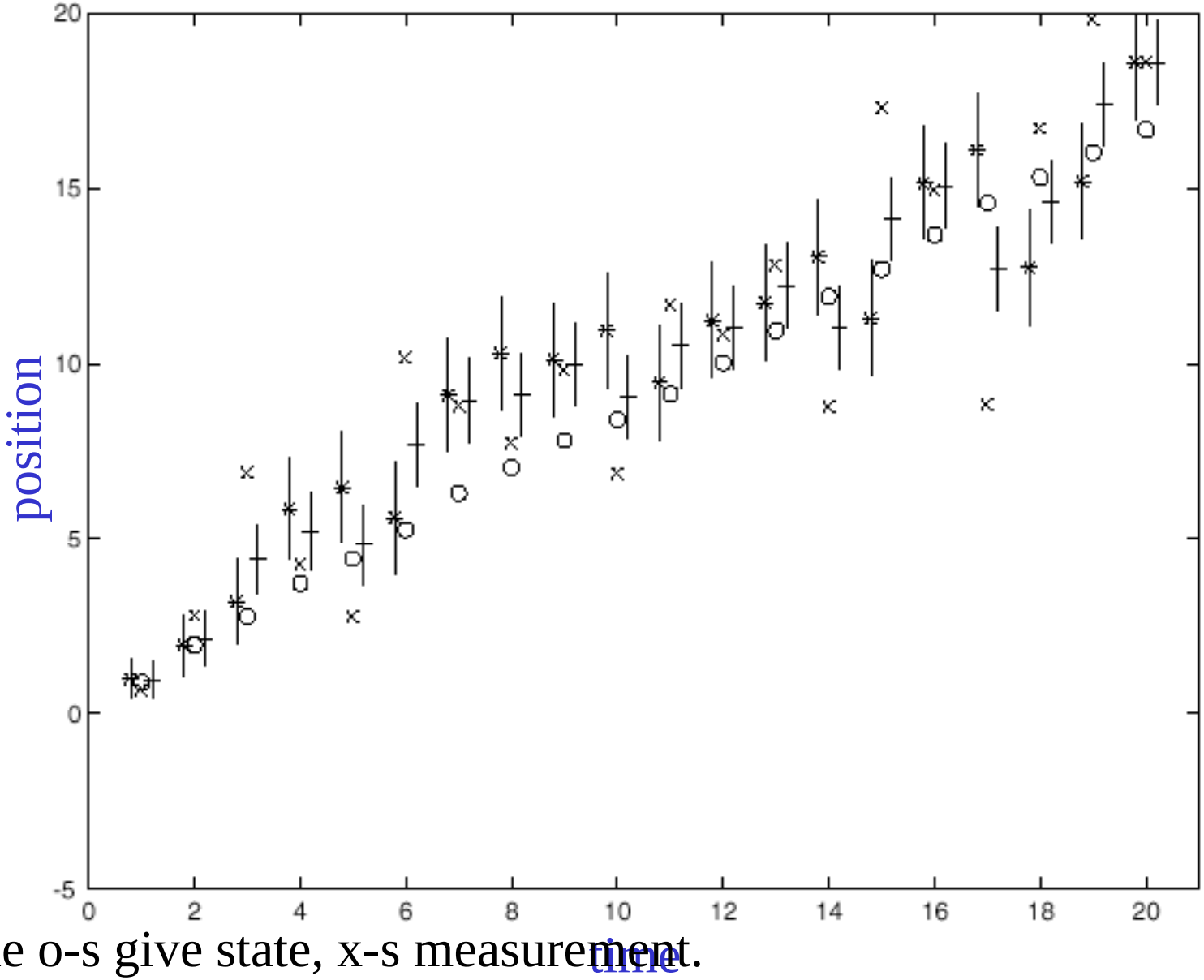


The *-s give $\overline{\boldsymbol{x}}_i^-$, +-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars

# Smoothing

- Idea
  - We don’t have the best estimate of state - what about the future?
  - Run two filters, one moving forward, the other backward in time.
  - Now combine state estimates
    - The crucial point here is that we can obtain a smoothed estimate by viewing the backward filter’s prediction as yet another measurement for the forward filter

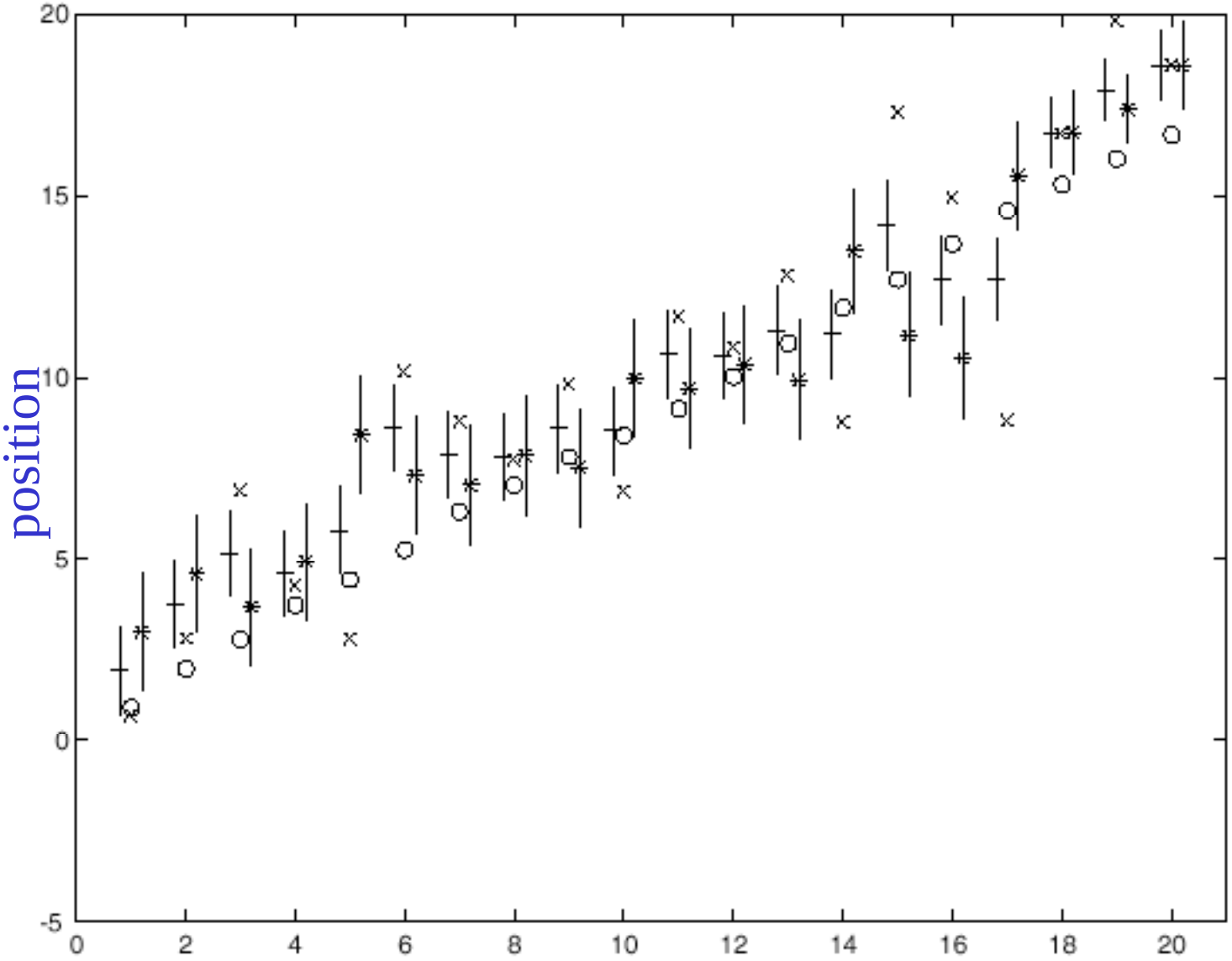
Forward estimates.




The o-s give state, x-s measurement.

The *-s give $\overline{\boldsymbol{x}}_i^-$ , +-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars

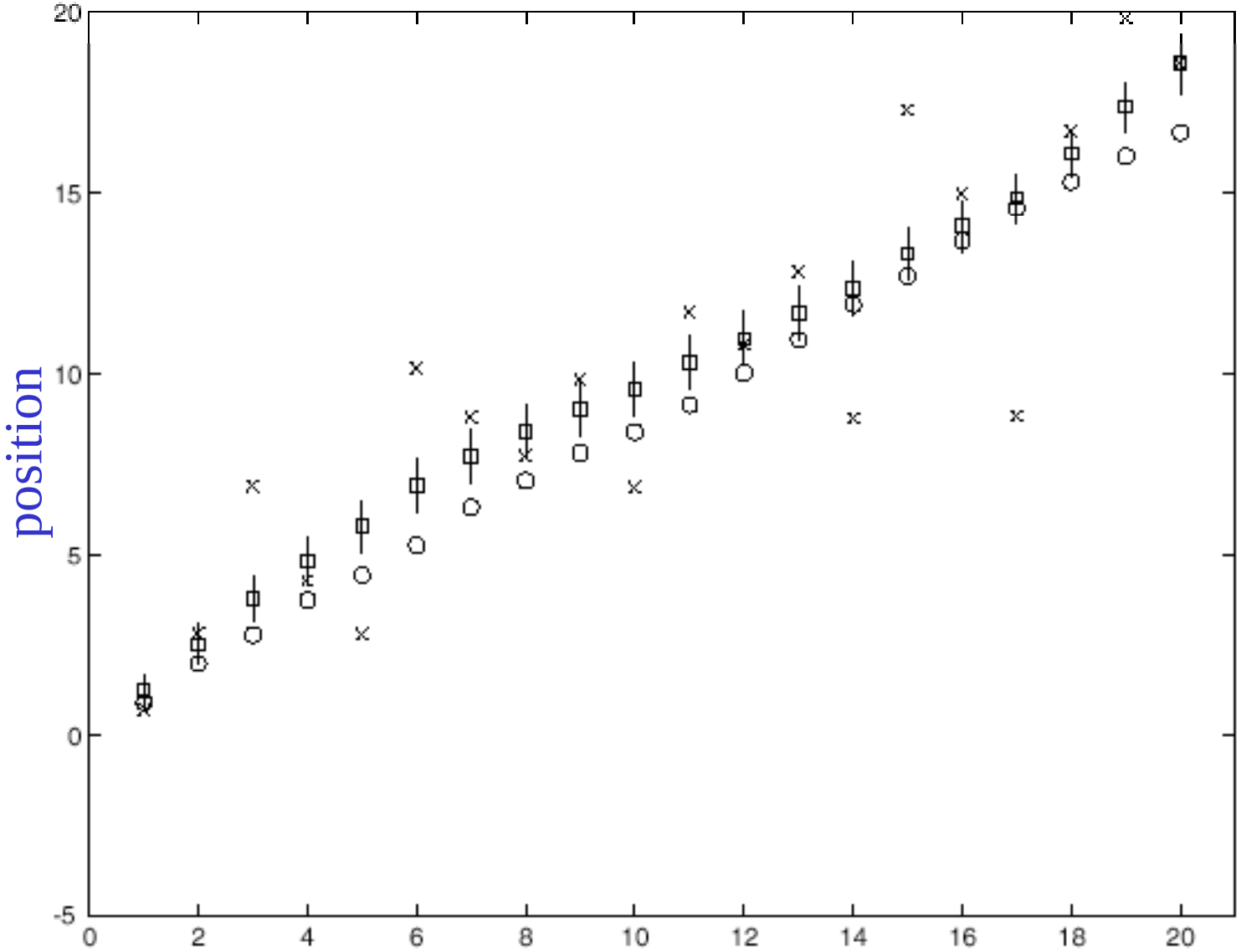
Backward estimates.




The o-s give state, x-s measurement.

The *-s give $\overline{x}_i^-$, +-s give $\overline{x}_i^+$, vertical bars are 3 standard deviation bars

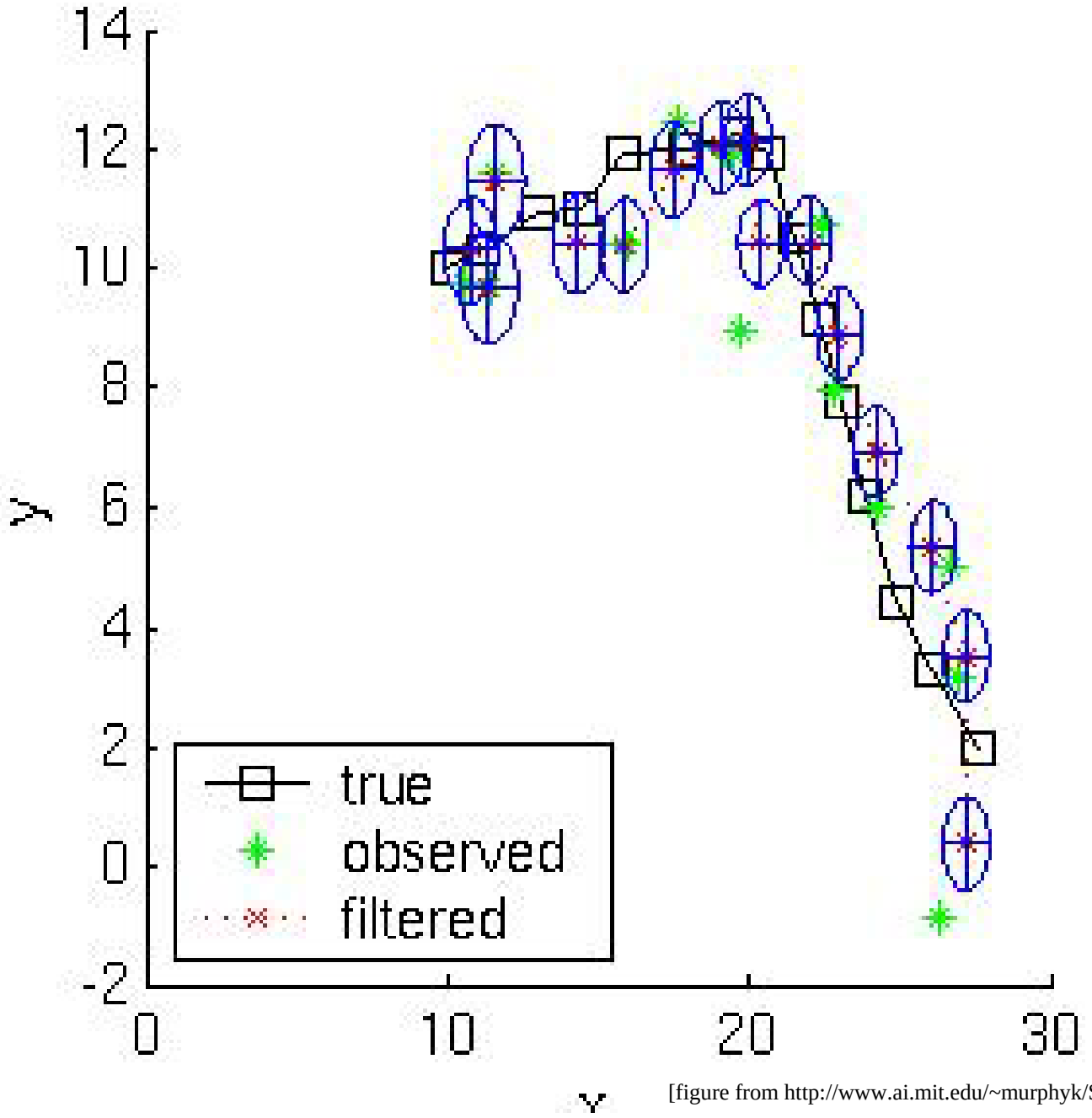
# Combined forward-backward estimates.



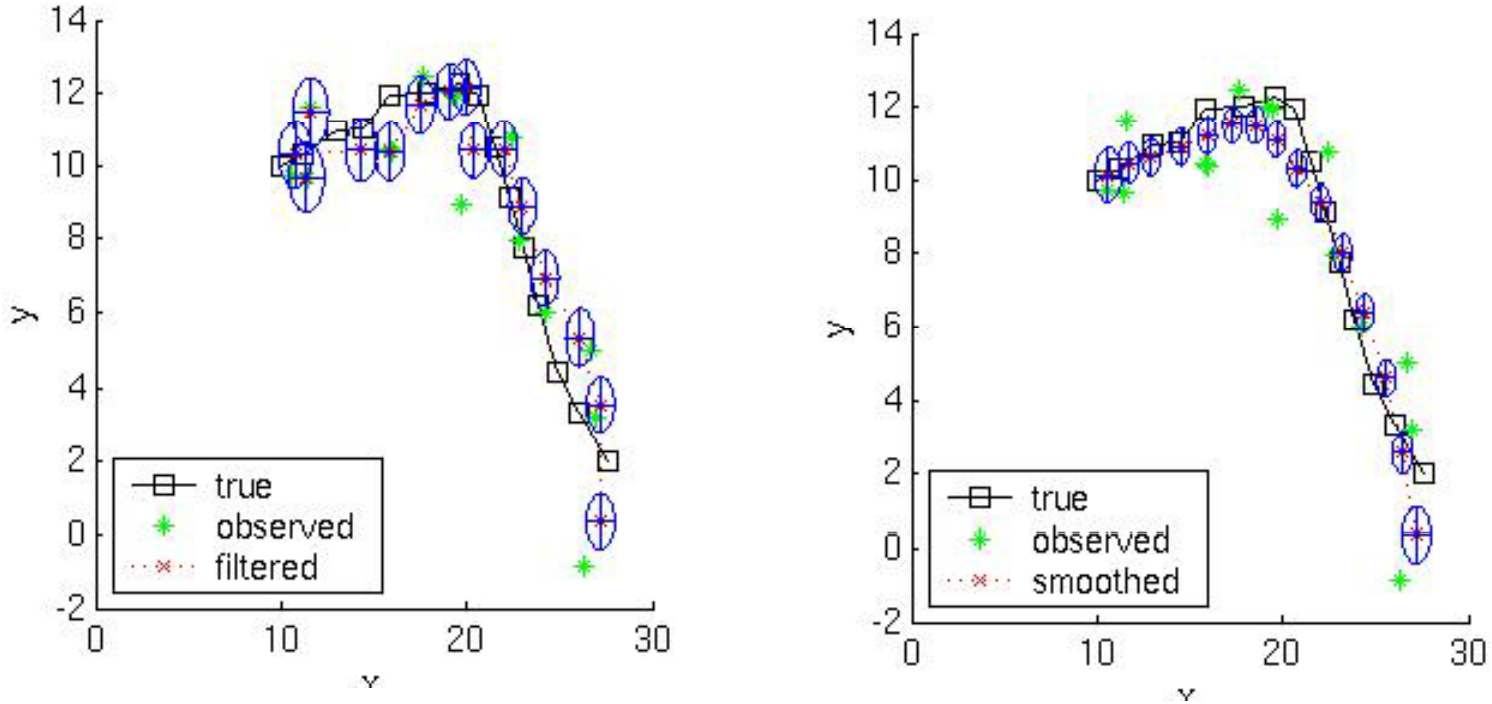

The o-s give state, x-s measurement.

The *-s give $\overline{x}_i^-$ , +-s give $\overline{x}_i^+$, vertical bars are 3 standard deviation bars

2-D constant velocity example from Kevin Murphy's Matlab toolbox




[figure from http://www.ai.mit.edu/~murphyk/Software/Kalman/kalman.html]




2-D constant velocity example from Kevin Murphy's Matlab toolbox

- MSE of filtered estimate is 4.9; of smoothed estimate. 3.2.
- Not only is the smoothed estimate better, but we know that it is better, as illustrated by the smaller uncertainty ellipses
- Note how the smoothed ellipses are larger at the ends, because these points have seen less data.
- Also, note how rapidly the filtered ellipses reach their steady-state ("Ricatti") values.



[figure from http://www.ai.mit.edu/~murphyk/Software/Kalman/kalman.html]

# Resources

- Kalman filter homepage

http://www.cs.unc.edu/~welch/kalman/

(kalman filter demo applet)

- Kevin Murphy's Matlab toolbox:

http://www.ai.mit.edu/~murphyk/Software/Kalman/kalman.html

# Embellishments for tracking

- Richer models of $P(x_n|x_{n-1})$
- Richer models of $P(y_n|x_n)$
- Richer model of probability distributions

# Abrupt changes

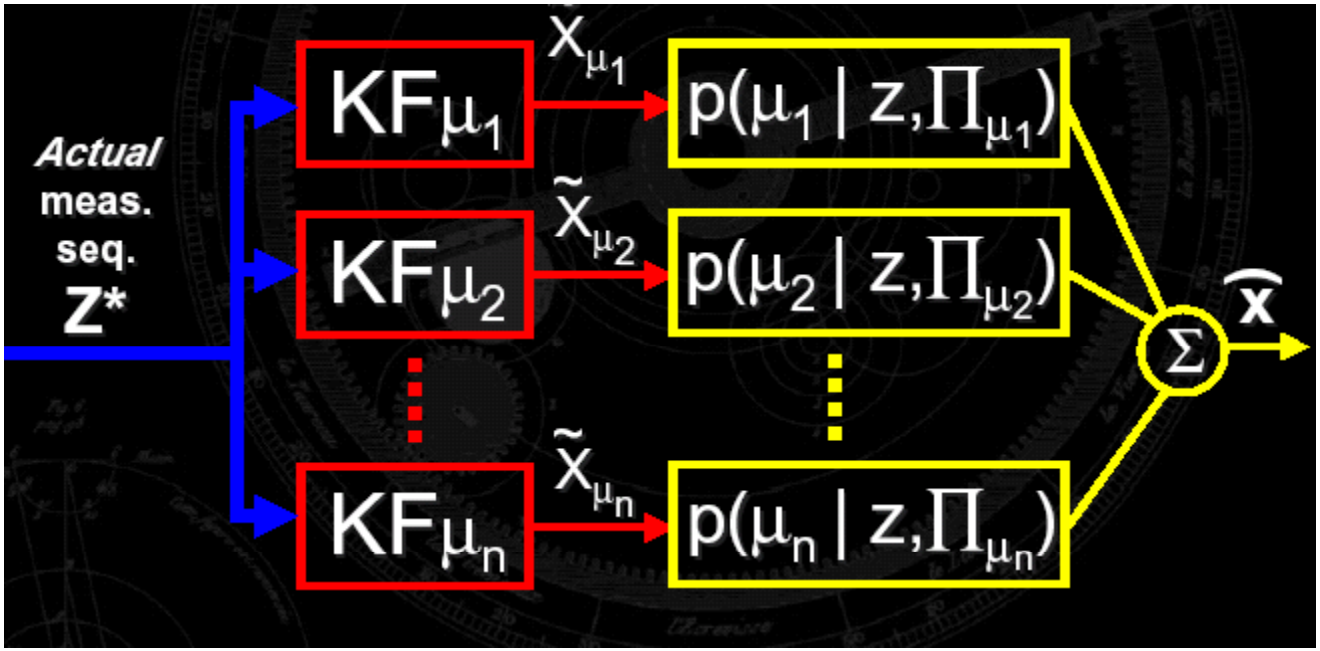
What if environment is sometimes unpredictable?

Do people move with constant velocity?

Test several models of assumed dynamics, use the best.

# Multiple model filters

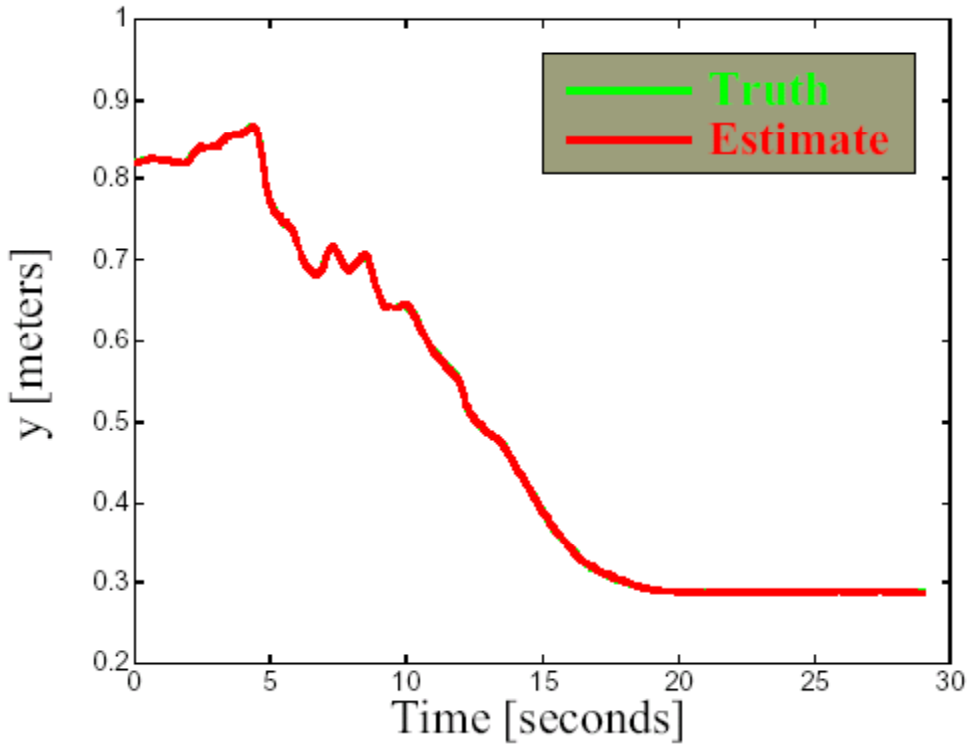
Test several models of assumed dynamics




[figure from Welsh and Bishop 2001]

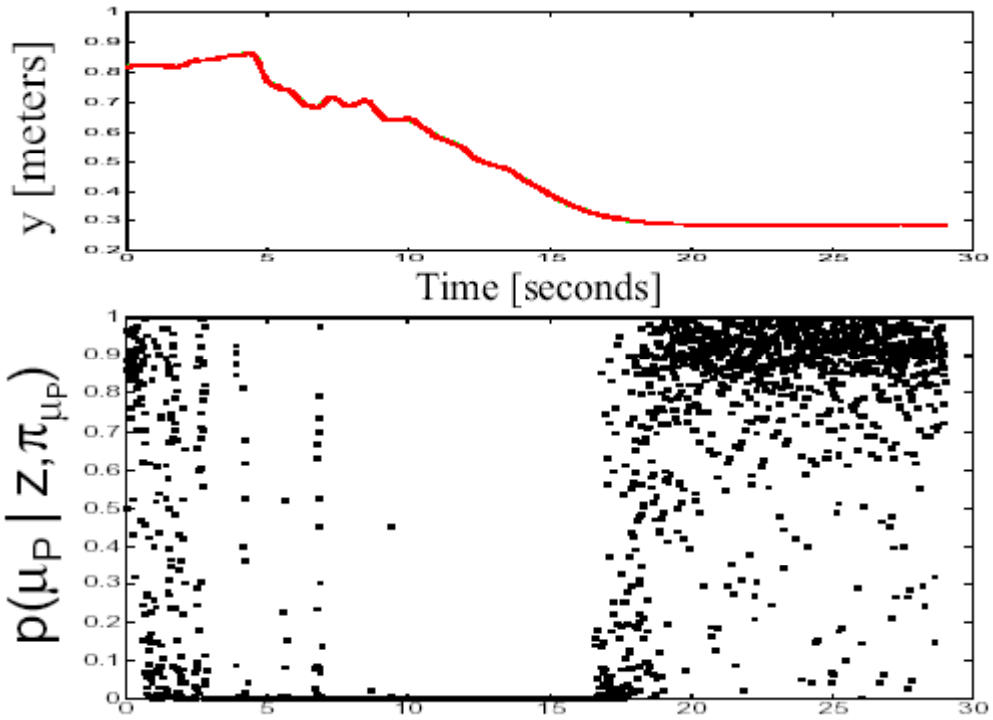
# MM estimate

Two models: Position (P), Position+Velocity (PV)



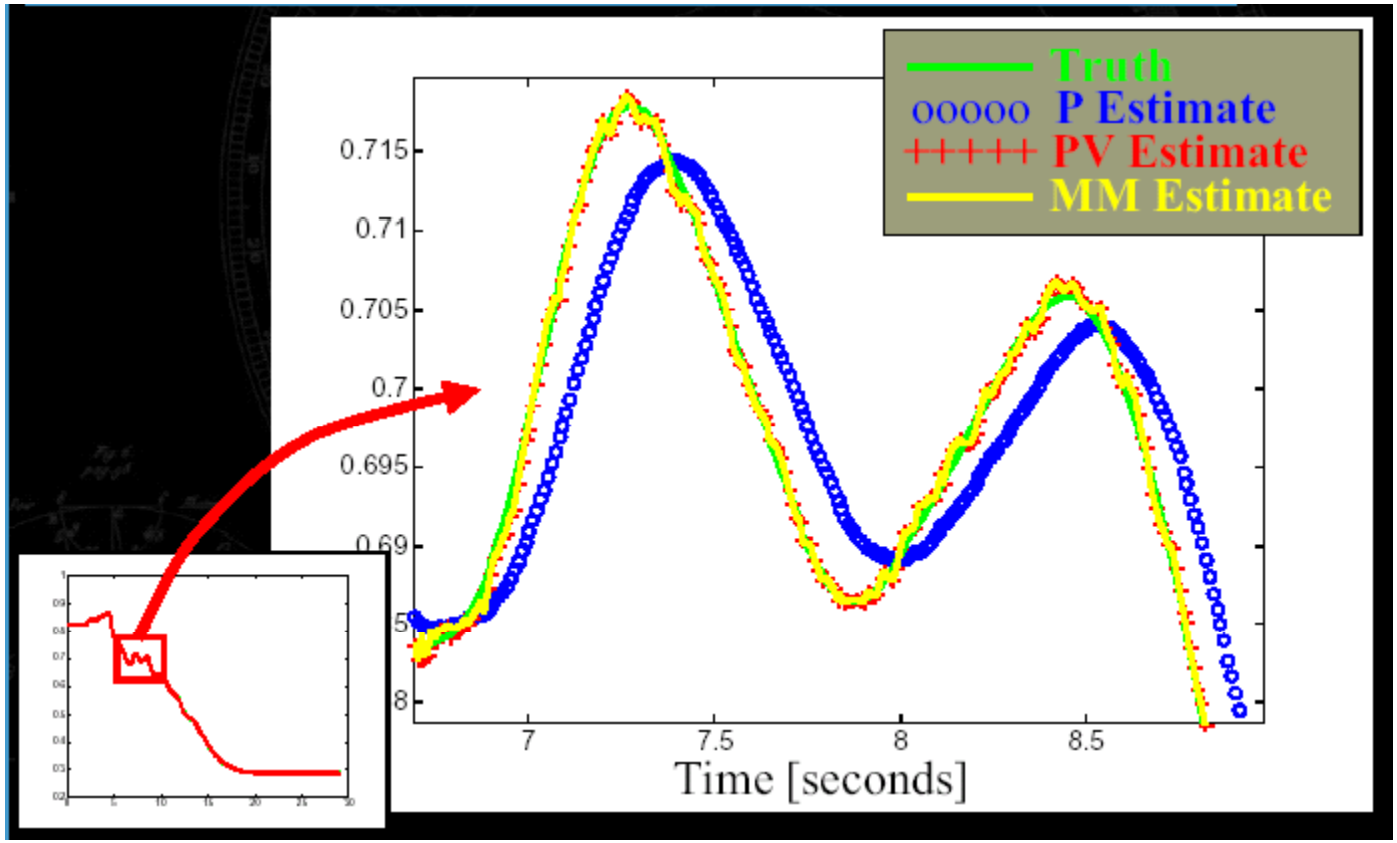

[figure from Welsh and Bishop 2001]

# P likelihood



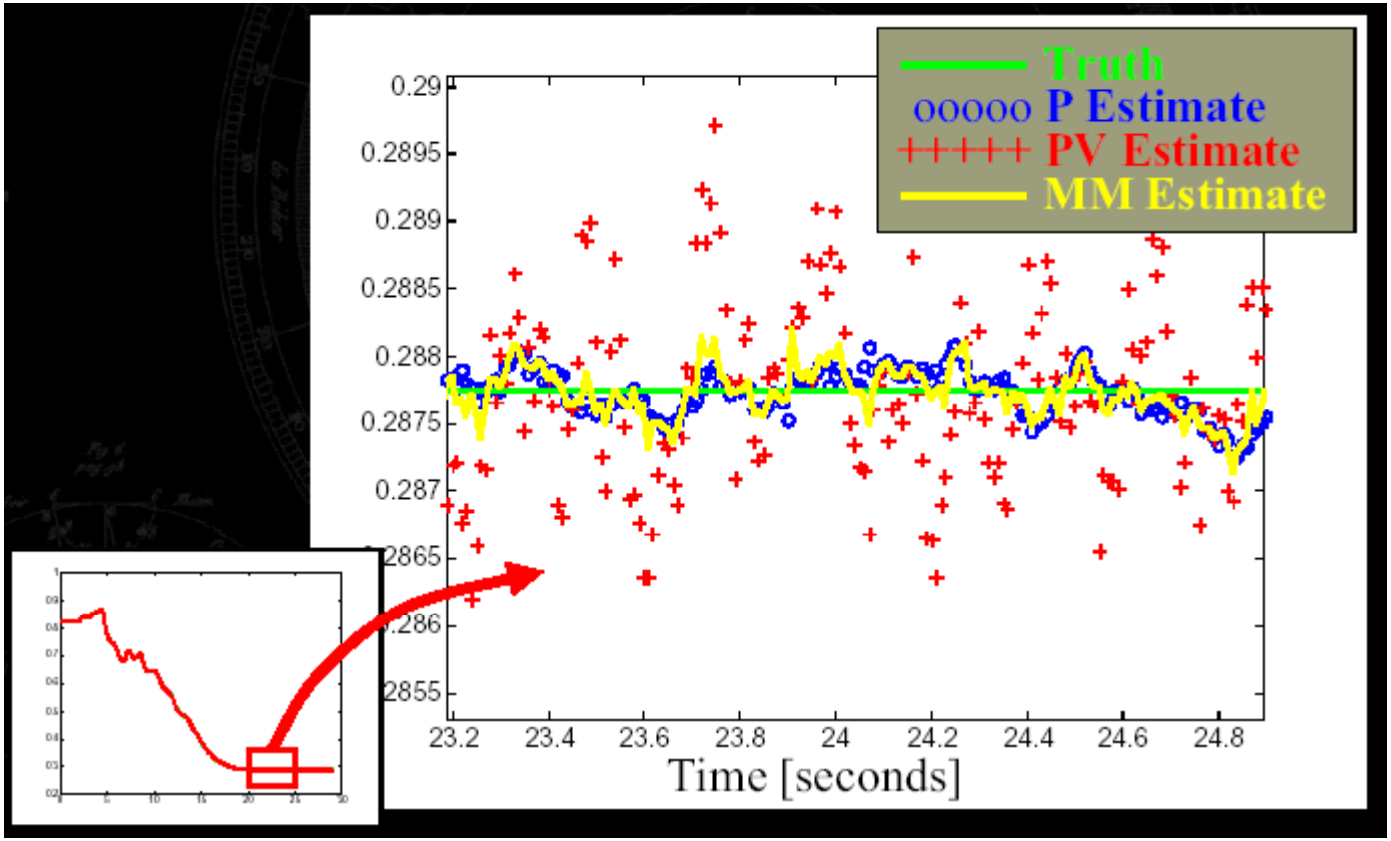

[figure from Welsh and Bishop 2001]

# No lag




[figure from Welsh and Bishop 2001]

# Smooth when still




[figure from Welsh and Bishop 2001]

# Embellishments for tracking

- Richer models of $P(y_n|x_n)$
- Richer model of probability distributions

# Jepson, Fleet, and El-Maraghi tracker

*IEEE Conference on Computer Vision and and Pattern Recognition, Kauai, 2001, Vol. I, pp. 415–422*

## Robust Online Appearance Models for Visual Tracking

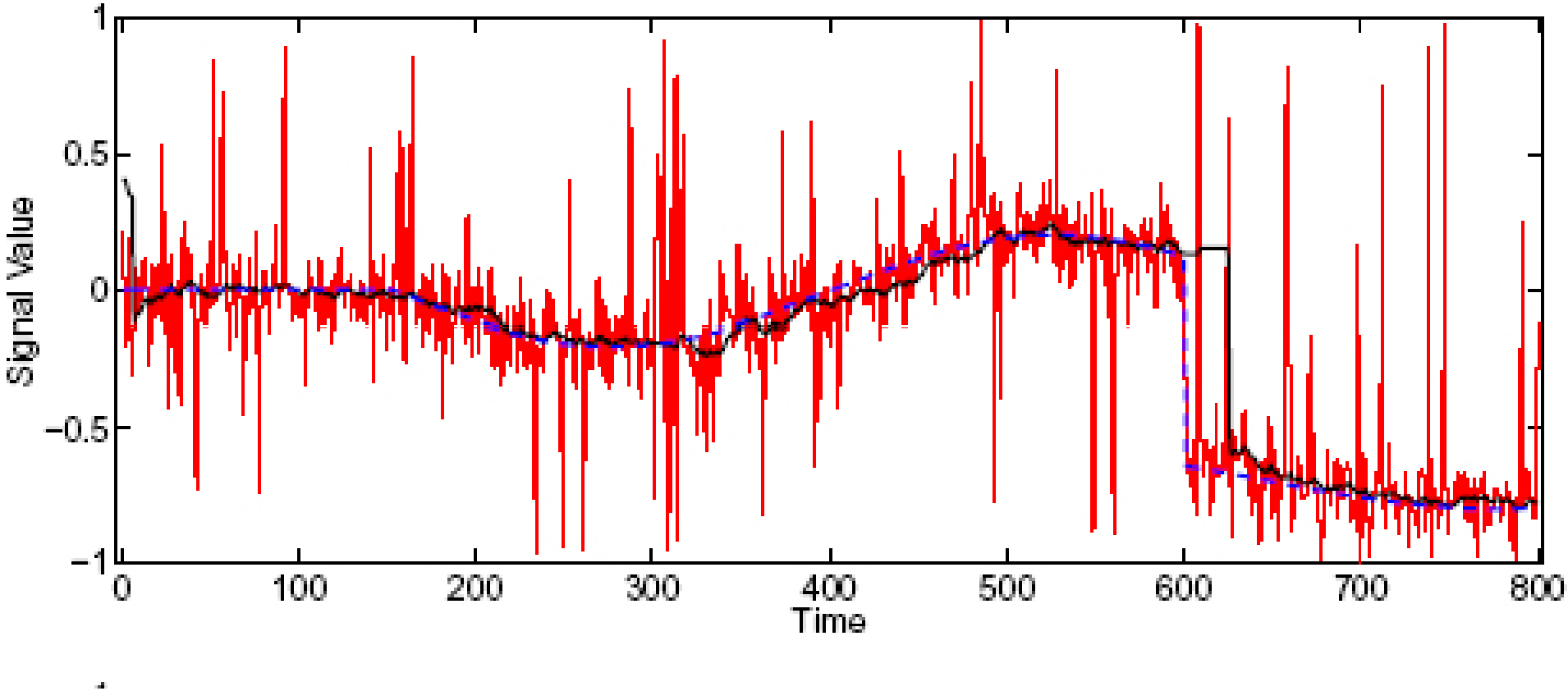
**Allan D. Jepson*** **David J. Fleet†** **Thomas F. El-Maraghi*†**

* Department of Computer Science, University of Toronto, Toronto, M5S 1A4
† Xerox Palo Alto Research Center, 3333 Coyote Hill Rd, Palo Alto, CA 94304

# Wandering, Stable, and Lost appearance model

- Introduce 3 competing models to explain the appearance of the tracked region:
  - A stable model—Gaussian with some mean and covariance.
  - A 2-frame motion tracker appearance model, to rebuild the stable model when it gets lost
  - An outlier model—uniform probability over all appearances.
- Use an on-line EM algorithm to fit the (changing) model parameters to the recent appearance data.

# Jepson, Fleet, and El-Maraghi tracker for toy 1-pixel image model




Red line: observations

Blue line: true appearance state

Black line: mean of the stable process

Mixing probabilities for Stable (black), Wandering (red) and Lost (green) processes

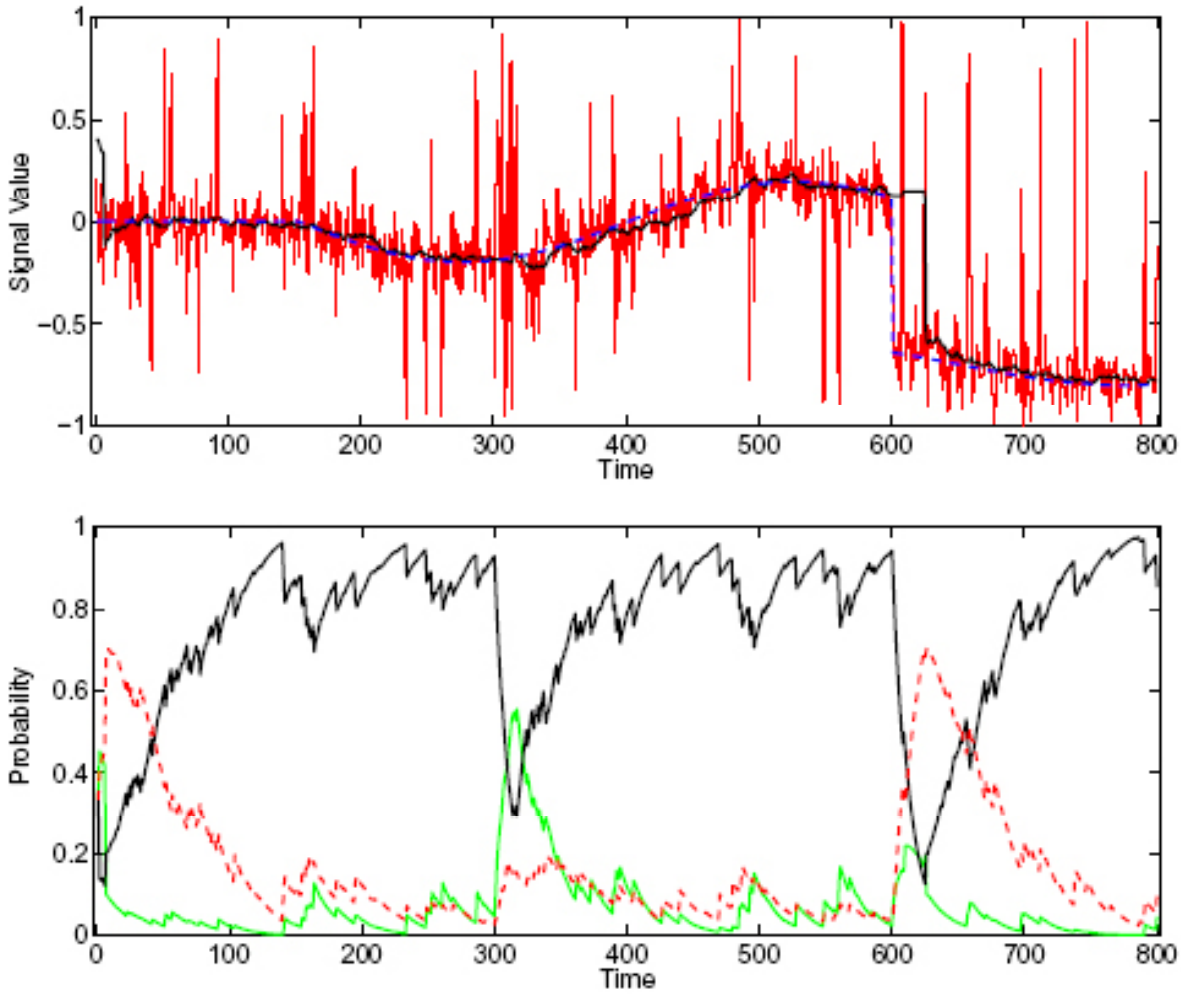



Figure 2. Estimation using on-line EM. (top) The original data (thin red) with true state (dashed blue) and the estimated mean of the stable process (thick black). The noise is a mixture of Gaussian and uniform densities, with mixing probabilities (0.9, 0.1), except for 15 frames at 300 which are pure outliers. (bottom) Mixing probabilities for $\mathcal{S}$ (black), $\mathcal{W}$ (dashed red), and the $\mathcal{L}$ (light green).

# Non-toy image representation

Phase of a steerable quadrature pair (G2, H2).
Steered to 4 different orientations, at 2 scales.

# The motion tracker

- Motion prior, $P(x_n \mid x_{n-1})$, prefers slow velocities and small accelerations.
- The WSL appearance model gives a likelihood for each possible new position, orientation, and scale of the tracked region.
- They combine that with the motion prior to find the most probable position, orientation, and scale of the tracked region in the next frame.
- Gives state-of-the-art tracking results.

# Jepson, Fleet, and El-Maraghi tracker

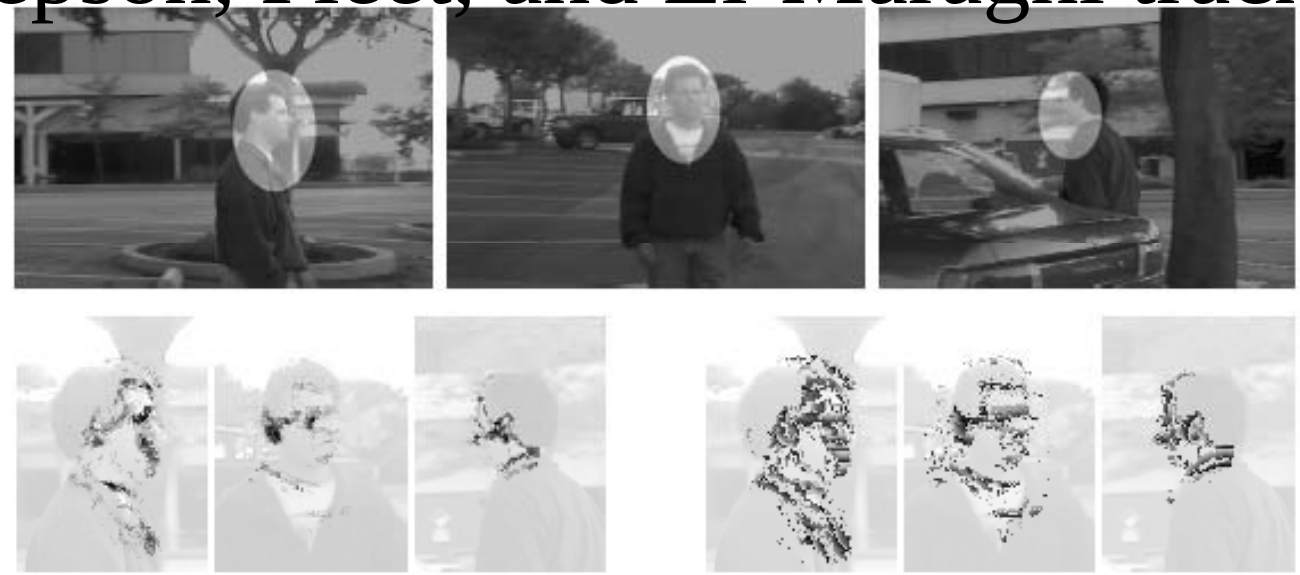
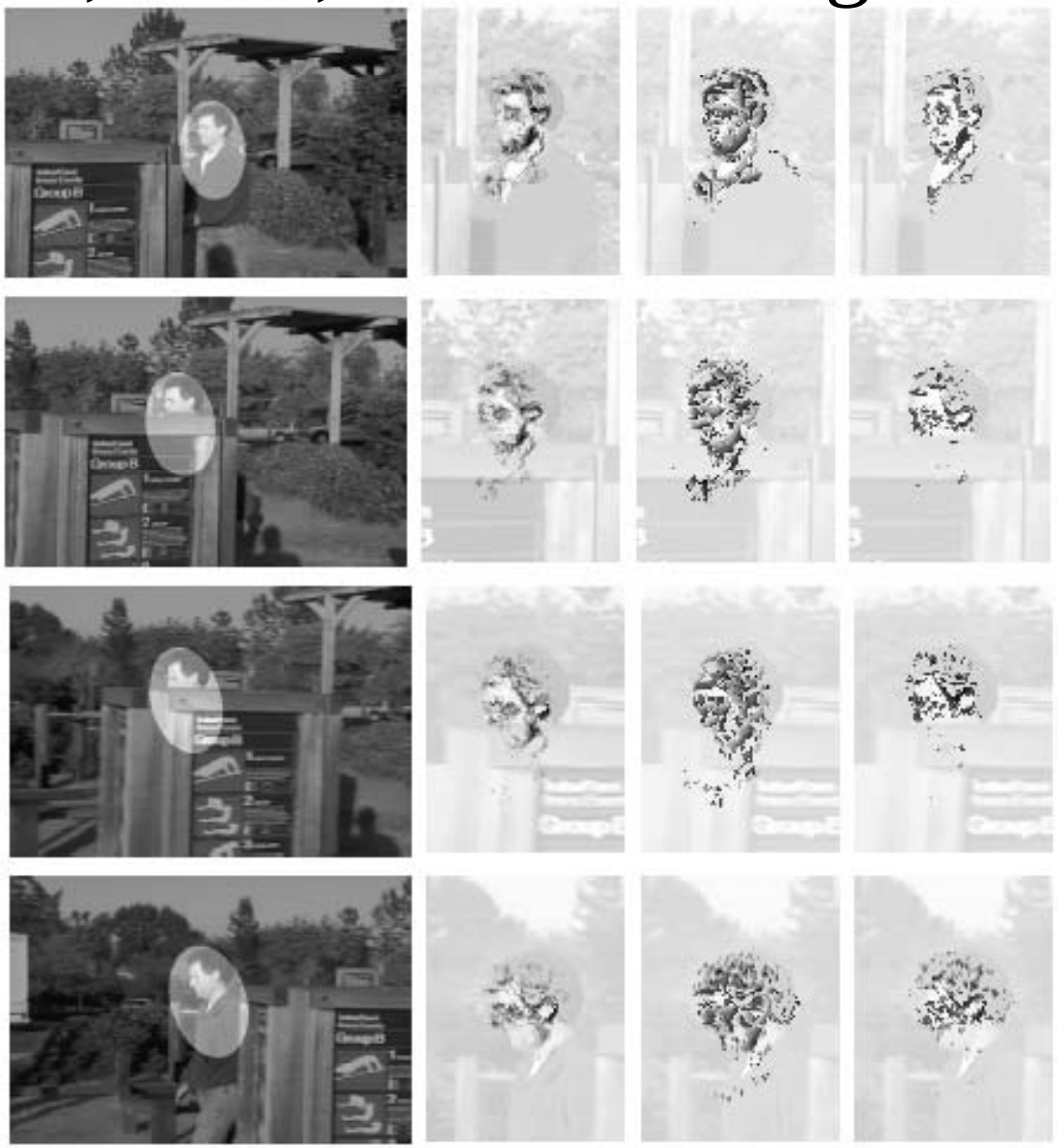


Figure 4. The adaptation of the model during tracking. (top) The target region in selected frames 200, 300, 480. (bottom) The stable component's mixing probability (left) and mean (right) for the selected frames.

# Jepson, Fleet, and El-Maraghi tracker



Far right column: the stable component's mixing probability. Note its behavior at occlusions.

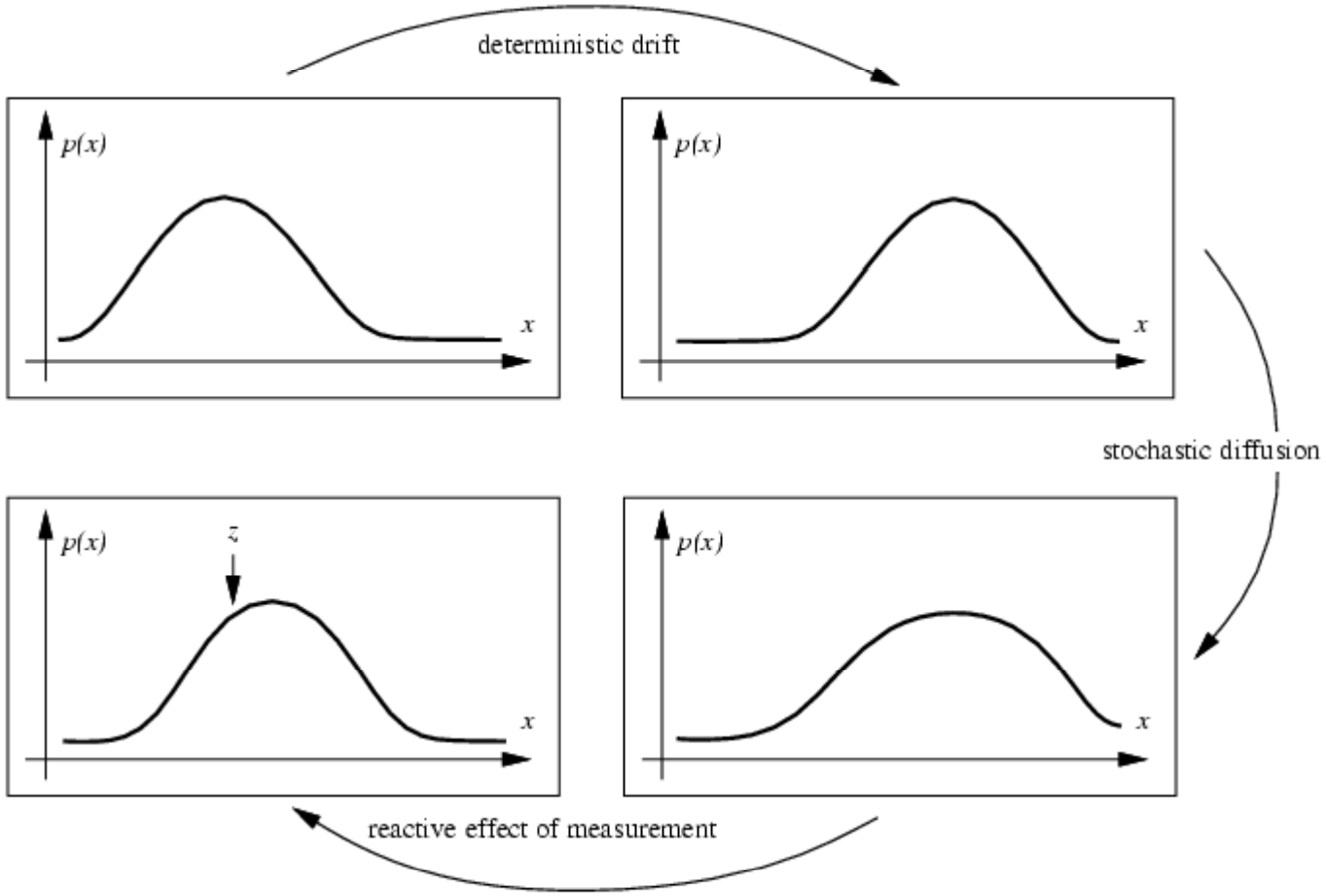
Figure 3. Each row shows, from left to right, the tracking region, the stable component's mixing probability $m_s(\mathbf{x}, t)$, mean $\mu_s(\mathbf{x}, t)$, and ownership probability $o_s(\mathbf{x}, t)$. The rows correspond to frames 244, 259, 274, and 289, top to bottom. Note the model persistence and the drop in data ownership within the occluded region.

# Embellishments for tracking

- Richer model of $P(y_n|x_n)$
- Richer model of probability distributions
  - Particle filter models, applied to tracking humans

# (KF) Distribution propagation

prediction from previous time frame



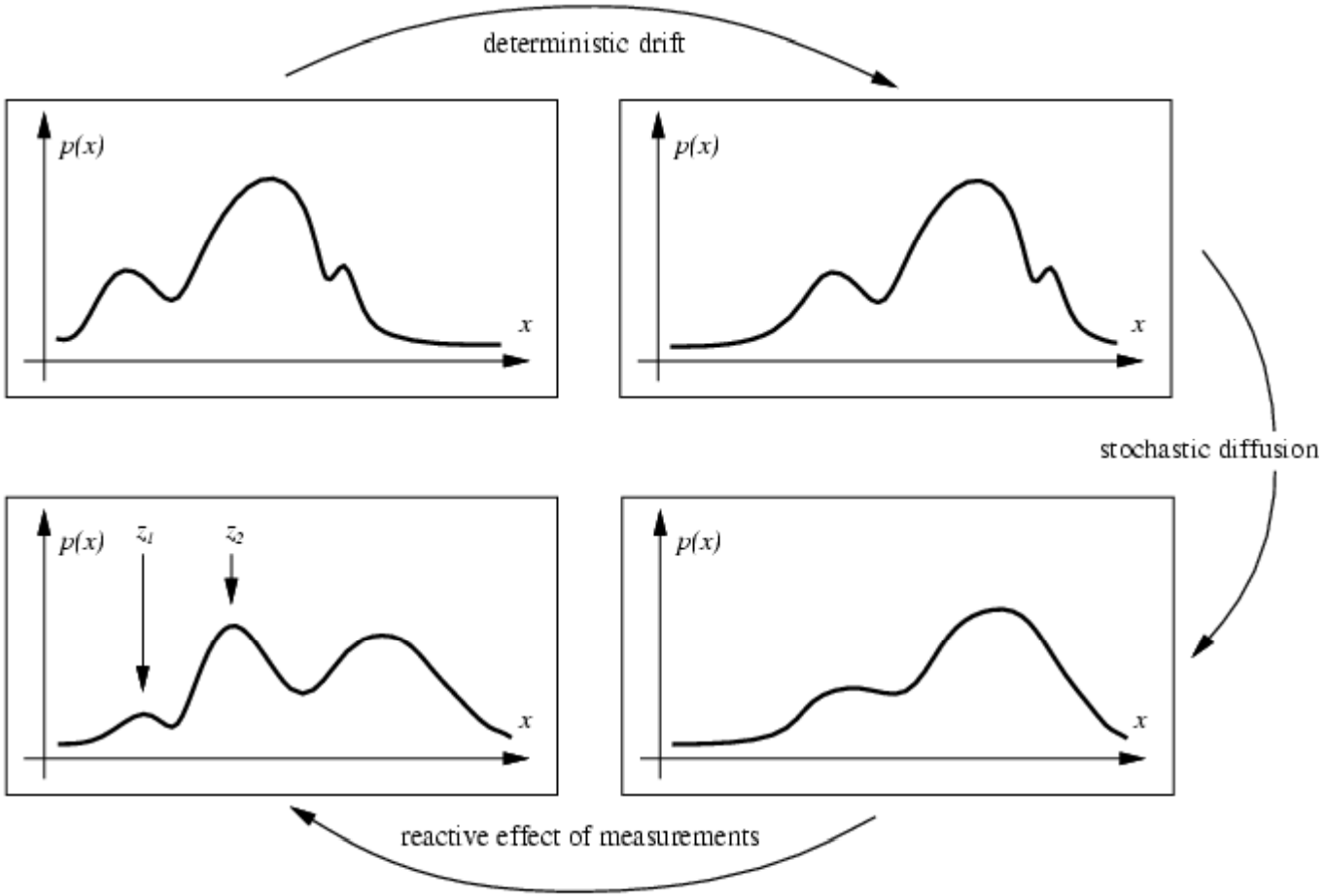

Noise added to that prediction

Make new measurement at next time frame



[Isard 1998]

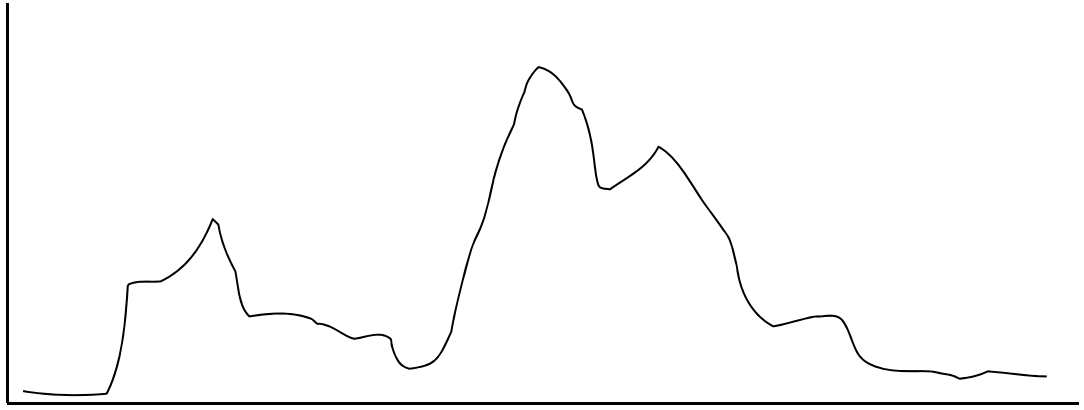
# Distribution propagation





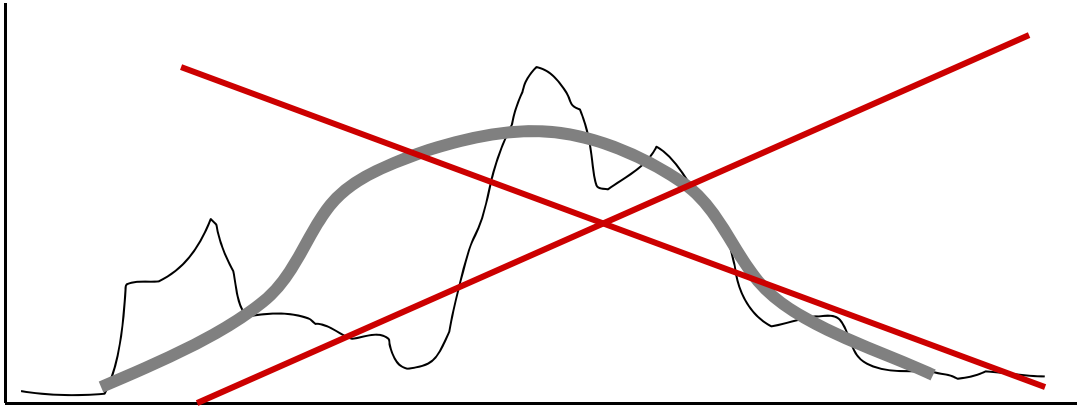

[Isard 1998]

# Representing non-linear Distributions



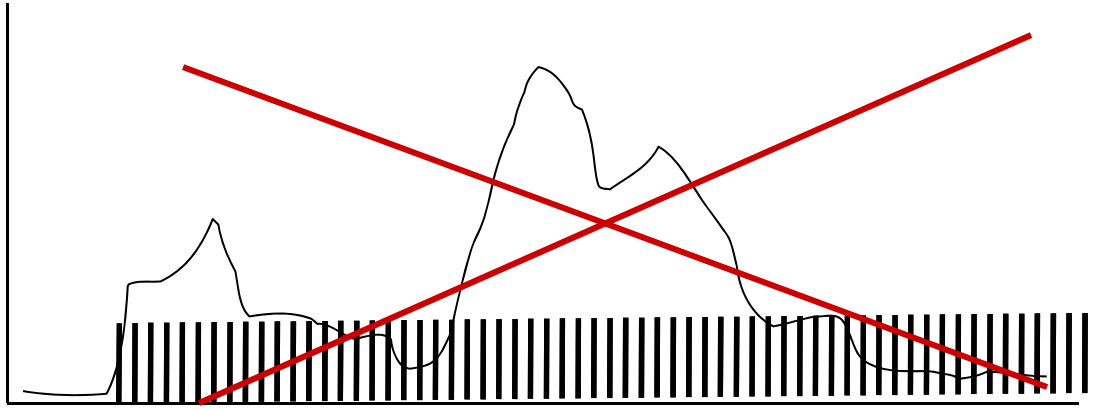
# Representing non-linear Distributions

Unimodal parametric models fail to capture real-world densities…



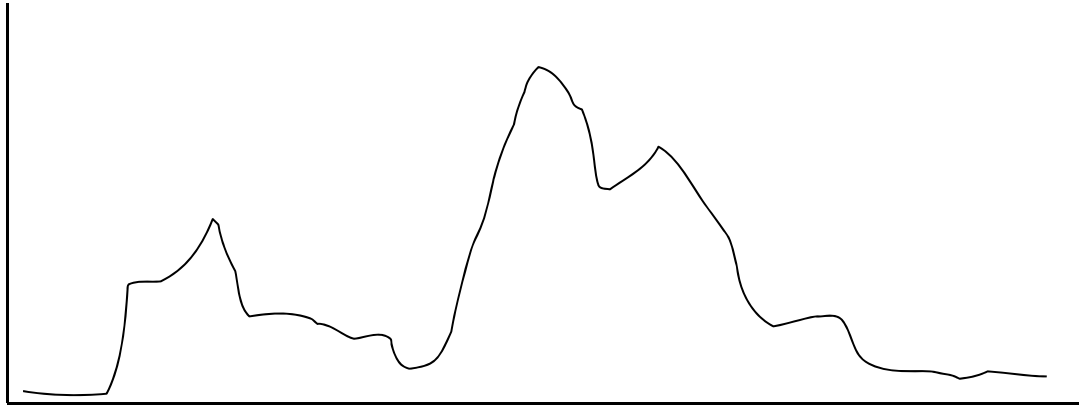
# Discretize by evenly sampling over the entire state space

Tractable for 1-d problems like stereo, but not for high-dimensional problems.



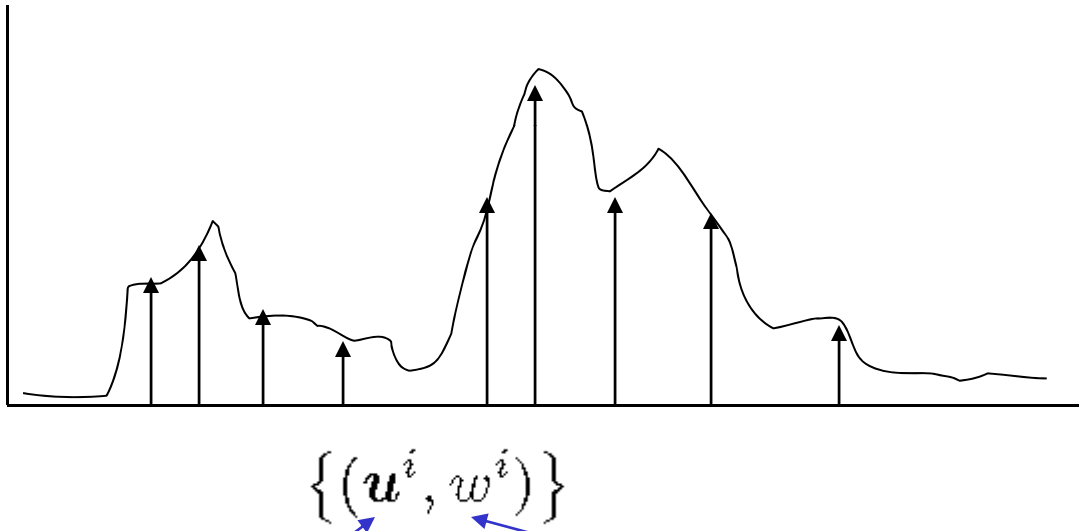
# Representing Distributions using Weighted Samples

Rather than a parametric form, use a set of samples to represent a density:



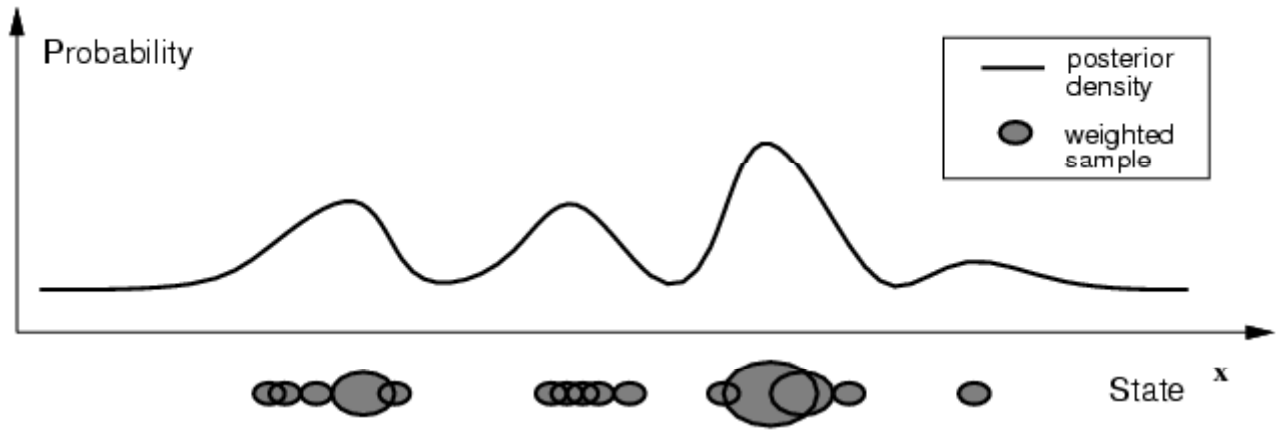
# Representing Distributions using Weighted Samples

Rather than a parametric form, use a set of samples to represent a density:




This gives us two knobs to adjust when representing a probability density by samples: the locations of the samples, and the probability weight on each sample.

# Representing distributions using weighted samples, another picture






[Isard 1998]

# Sampled representation of a probability distribution

Represent a probability distribution

$$p_f(\boldsymbol{X}) = \frac{f(\boldsymbol{X})}{\int f(\boldsymbol{U})d\boldsymbol{U}}$$

by a set of $N$ weighted samples

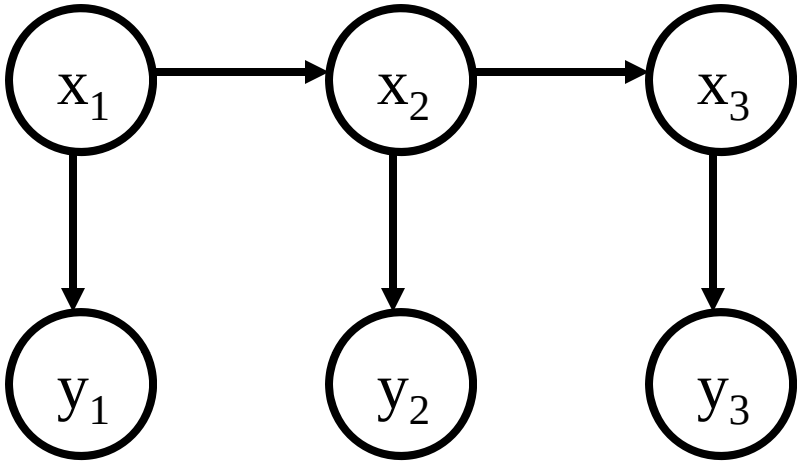
$$\left\{(\boldsymbol{u}^i, w^i)\right\}$$

where $\boldsymbol{u}^i \sim s(\boldsymbol{u})$ and $w^i = f(\boldsymbol{u}^i)/s(\boldsymbol{u}^i)$.

You can also think of this as a sum of dirac delta functions, each of weight w:

$$p_f(x) = \sum_i w^i \delta(x - u^i)$$

# Tracking, in particle filter representation




$$p_f(x) = \sum_i w^i \delta(x - u^i)$$

$$P(x_n \mid y_1 \ldots y_n) = k\, P(y_n \mid x_n) \int dx_{n-1} P(x_n \mid x_{n-1}) P(x_{n-1} \mid y_1 \ldots y_{n-1})$$

Update step

Prediction step

# Particle filter

- Let’s apply this sampled probability density machinery to generalize the Kalman filtering framework.
- More general probability density representation than uni-modal Gaussian.
- Allows for general state dynamics, f(x) + noise

# Sampled Prediction

$$P(\boldsymbol{x}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1}) \;=\; ?$$

$$p(\boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})$$

$$\left\{(\boldsymbol{u}_{i-1}^k, w_{i-1}^k)\right\} \longrightarrow \boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{x}_{i-1}) + \xi_i \longrightarrow$$

$$\left\{((f(\boldsymbol{u}_{i-1}^k) + \xi_i^l, \boldsymbol{u}_{i-1}^k), w_{i-1}^k)\right\}$$

$$p(\boldsymbol{X}_i, \boldsymbol{X}_{i-1}|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1})$$

Drop elements to marginalize to get

$$P(\boldsymbol{x}_i|\boldsymbol{y}_0,\ldots,\boldsymbol{y}_{i-1}) \;\sim=$$

$$\left\{(f(\boldsymbol{u}_{i-1}^k) + \xi_i^l, w_{i-1}^k)\right\}$$

# Sampled Correction (Bayes rule)

Prior → posterior

Reweight every sample with the likelihood of the observations, given that sample:

$$p(\boldsymbol{Y}_i = \boldsymbol{y}_i | \boldsymbol{X}_i = \boldsymbol{s}_i^{k,-}) w_i^{k,-}$$

yielding a set of samples describing the probability distribution after the correction (update) step:

$$\left\{ (\boldsymbol{s}_i^{k,-}, p(\boldsymbol{Y}_i = \boldsymbol{y}_i | \boldsymbol{X}_i = \boldsymbol{s}_i^{k,-}) w_i^{k,-}) \right\}$$

# Naïve PF Tracking

- Start with samples from something simple (Gaussian)
- Repeat
  - Correct

Take each particle from the prediction step and modify the old weight by multiplying by the new likelihood

$$\left\{(\boldsymbol{s}_i^{k,-}, p(\boldsymbol{Y}_i = \boldsymbol{y}_i | \boldsymbol{X}_i = \boldsymbol{s}_i^{k,-}) w_i^{k,-})\right\}$$

  - Predict

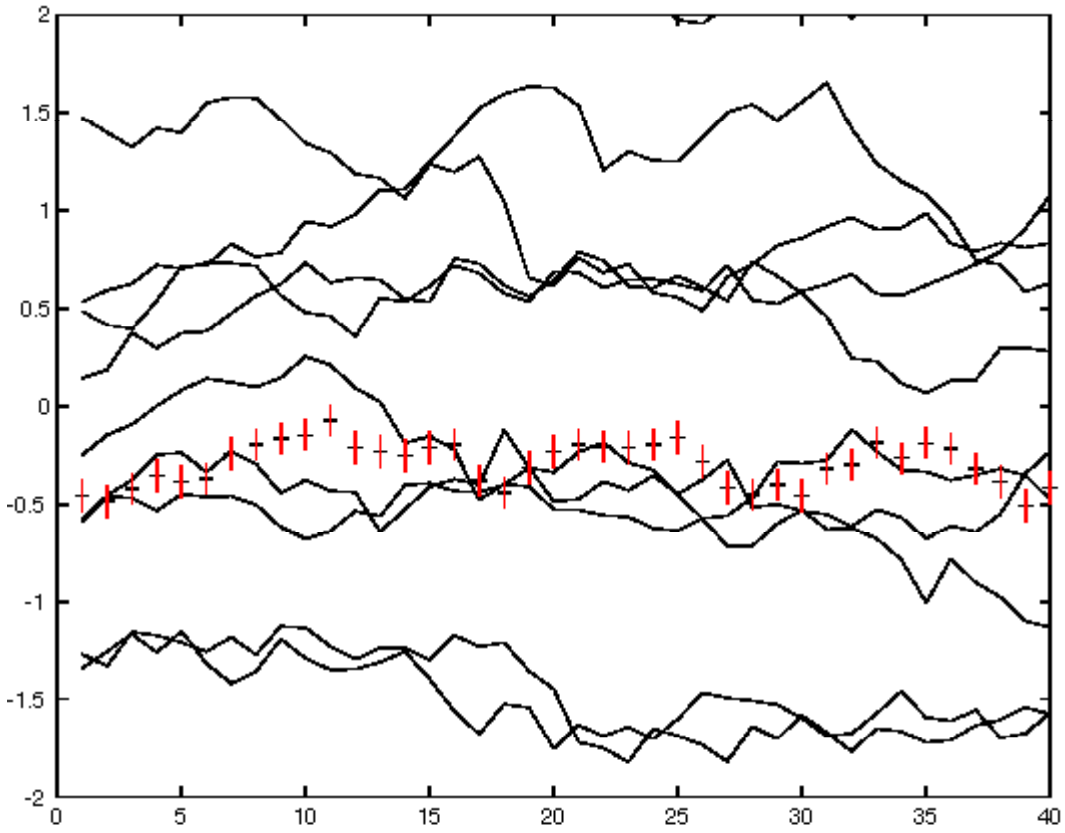
$$\left\{(f(\mathbf{s}_{i-1}^k) + \xi_i^l, w_{i-1}^k)\right\}$$

Run every particle through the dynamics function and add noise.

But doesn't work that well because of sample impoverishment…

# Sample impoverishment

10 of the 100 particles, along with the true Kalman filter track, with variance:



# Resample the prior

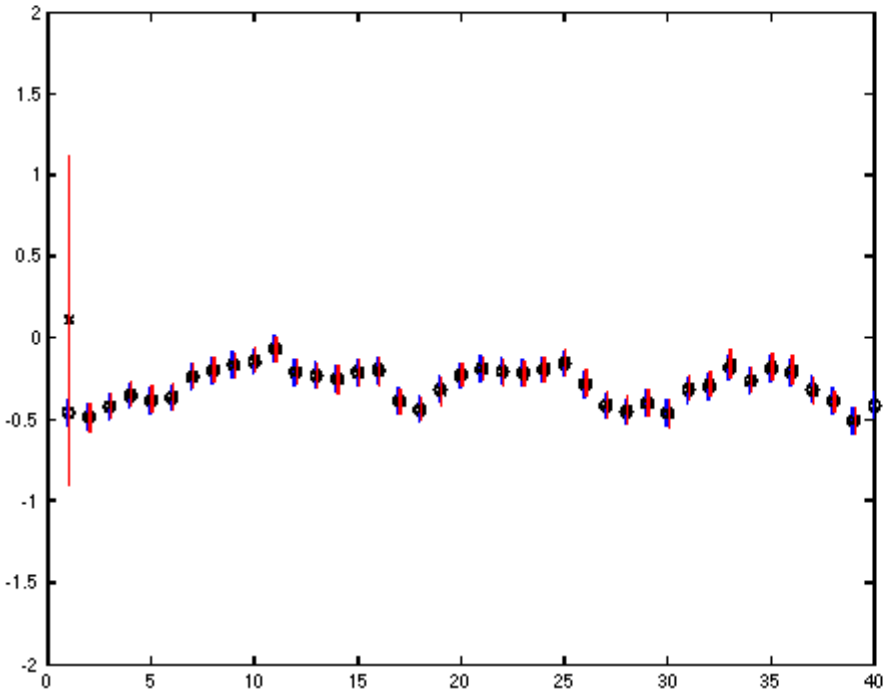
In a sampled density representation, the frequency of samples can be traded off against weight:

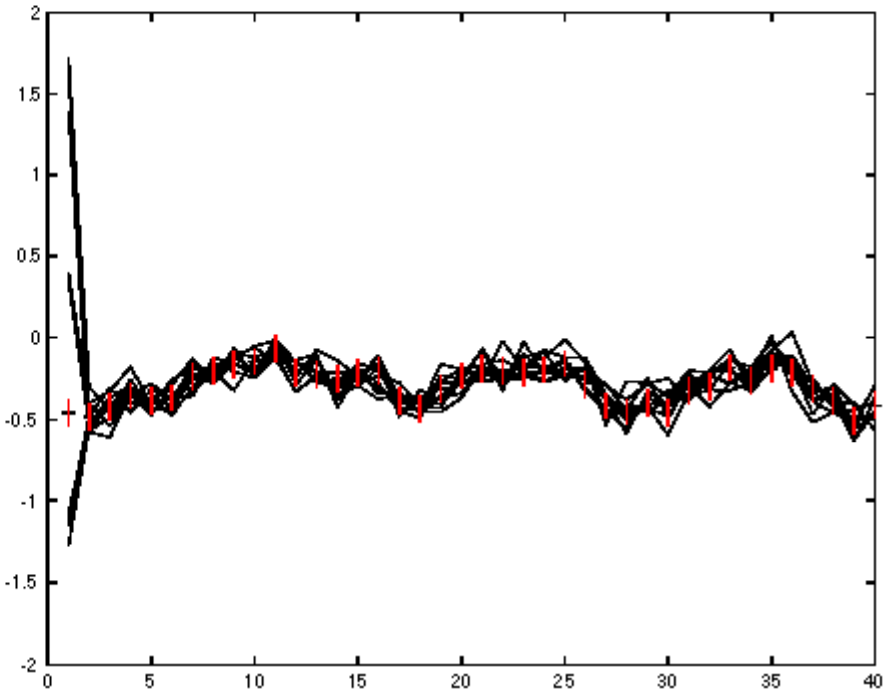
$$(\boldsymbol{s}_k, w_k) \longrightarrow \begin{matrix}(\boldsymbol{s}_k, 1)\\ (\boldsymbol{s}_k, 1)\\ (\boldsymbol{s}_k, 1)\\ \vdots\end{matrix} \; N_k \text{ copies} \quad \text{s.t.} \quad \frac{N_k}{\sum_k N_k} = w_k$$

These new samples are a representation of the same density.

I.e., make N draws with replacement from the original set of samples, using the weights as the probability of drawing a sample.

# Resampling concentrates samples



# A practical particle filter with resampling

**Initialization:** Represent $P(X_0)$ by a set of $N$ samples

$$\left\{(s_0^{k,-}, w_0^{k,-})\right\}$$

where

$$s_0^{k,-} \sim P_s(S) \text{ and } w_0^{k,-} = P(s_0^{k,-})/P_s(S = s_0^{k,-})$$

Ideally, $P(X_0)$ has a simple form and $s_0^{k,-} \sim P(X_0)$ and $w_0^{k,-} = 1$.
**Prediction:** Represent $P(X_i|y_0, y_{i-1})$ by

$$\left\{(s_i^{k,-}, w_i^{k,-})\right\}$$

where

$$s_i^{k,-} = f(s_{i-1}^{k,+}) + \xi_i^k \text{ and } w_i^{k,-} = w_{i-1}^{k,+} \text{ and } \xi_i^k \sim N(0, \Sigma_{d_i})$$

**Correction:** Represent $P(X_i|y_0, y_i)$ by

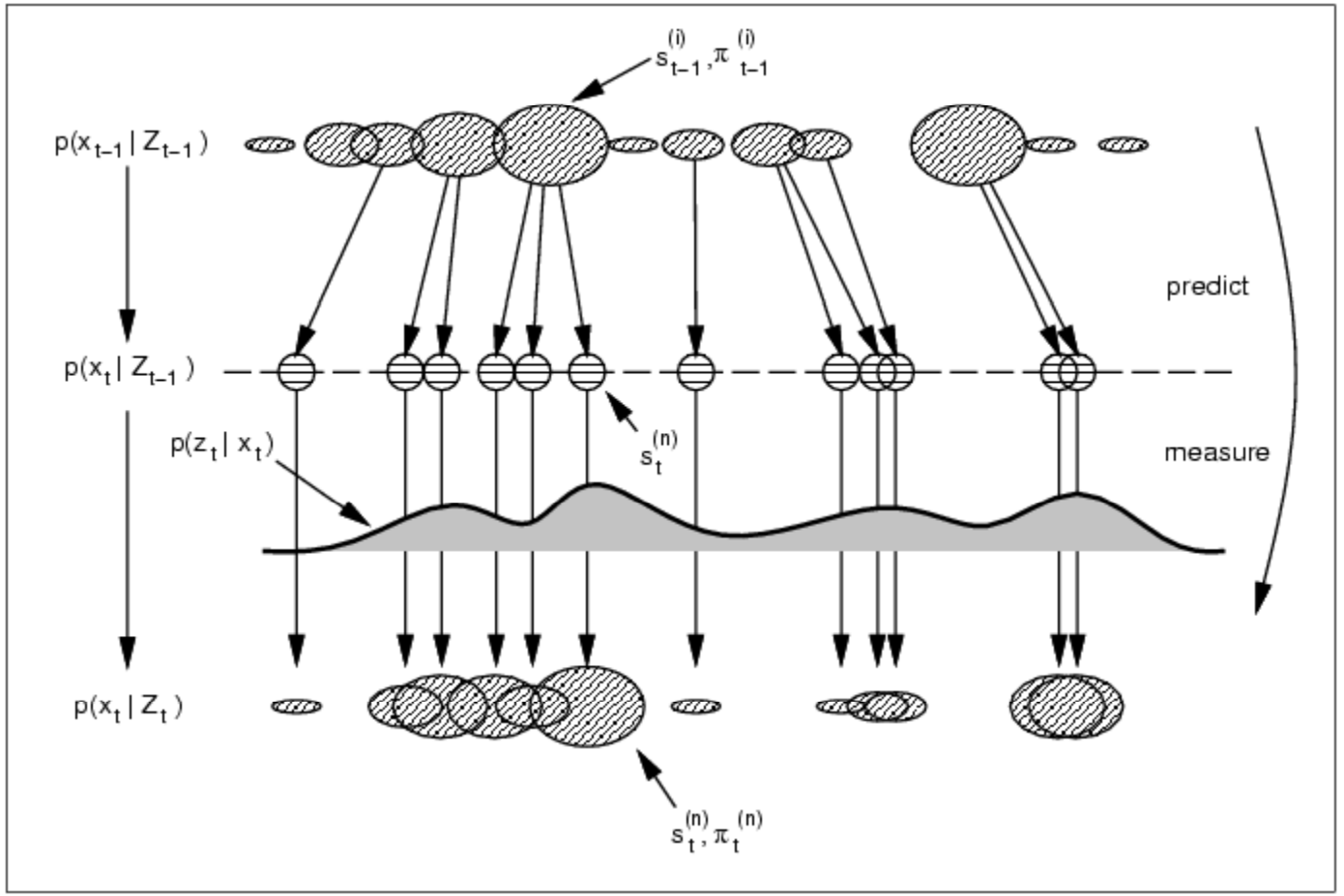
$$\left\{(s_i^{k,+}, w_i^{k,+})\right\}$$

where

$$s_i^{k,+} = s_i^{k,-} \text{ and } w_i^{k,+} = P(Y_i = y_i|X_i = s_i^{k,-})w_i^{k,-}$$

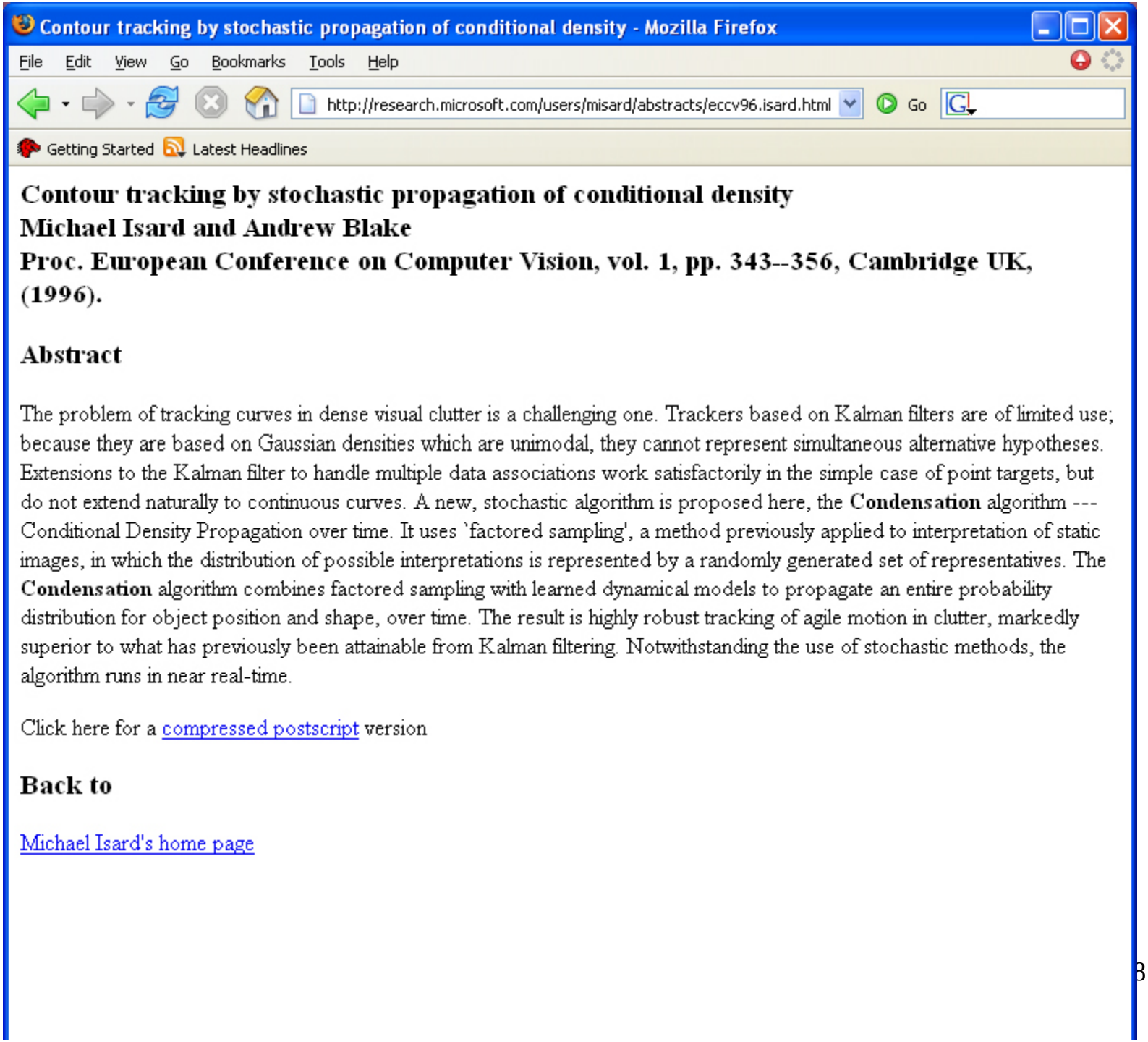
**Resampling:** Normalise the weights so that $\sum_i w_i^{k,+} = 1$ and compute the variance of the normalised weights. If this variance exceeds some threshold, then construct a new set of samples by drawing, with replacement, $N$ samples from the old set, using the weights as the probability that a sample will be drawn. The weight of each sample is now $1/N$.

# Pictorial view






[Isard 1998]


Contour tracking by stochastic propagation of conditional density - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://research.microsoft.com/users/misard/abstracts/eccv96.isard.html
Go
Getting Started
Latest Headlines
Contour tracking by stochastic propagation of conditional density
Michael Isard and Andrew Blake
Proc. European Conference on Computer Vision, vol. 1, pp. 343--356, Cambridge UK, (1996).
Abstract
The problem of tracking curves in dense visual clutter is a challenging one. Trackers based on Kalman filters are of limited use; because they are based on Gaussian densities which are unimodal, they cannot represent simultaneous alternative hypotheses. Extensions to the Kalman filter to handle multiple data associations work satisfactorily in the simple case of point targets, but do not extend naturally to continuous curves. A new, stochastic algorithm is proposed here, the Condensation algorithm --- Conditional Density Propagation over time. It uses 'factored sampling', a method previously applied to interpretation of static images, in which the distribution of possible interpretations is represented by a randomly generated set of representatives. The Condensation algorithm combines factored sampling with learned dynamical models to propagate an entire probability distribution for object position and shape, over time. The result is highly robust tracking of agile motion in clutter, markedly superior to what has previously been attainable from Kalman filtering. Notwithstanding the use of stochastic methods, the algorithm runs in near real-time.
Click here for a compressed postscript version
Back to
Michael Isard's home page

# The Condensation Algorithm

## Background

Tracking objects through highly cluttered scenes is difficult. We believe that for tracking to be robust when following agile moving objects, in the presence of dense background clutter, probabilistic algorithms are essential. Previous algorithms, for example the Kalman filter, have been limited in the range of probability distributions they represent. We have developed a new algorithm, the **Condensation** algorithm (**Con**ditional **Den**sity Propag**ation**) which allows quite general representations of probability. Experimental results show that this increased generality does indeed lead to a marked improvement in tracking performance. In addition to permitting high-quality tracking in clutter, the simplicity of the **Condensation** algorithm also allows the use of non-linear motion models more complex than those commonly used in Kalman filters. We have implemented a mixed discrete/continuous tracker in the **Condensation** framework which switches between multiple continuous Auto-Regressive Process motion models according to a discrete transition matrix. Also, by using the statistical technique of *importance sampling* it is possible to build a **Condensation** tracker which runs in real time, and we have implemented a real-time hand-tracker on a low-end SGI workstation. My D.Phil. thesis gives a thorough description of the algorithm and some applications.

## Sample Code

Download source code of a simple implementation of the **Condensation** algorithm.

## Results

Here is an MPEG (2.3Mb) showing the **Condensation** algorithm tracking a leaf blowing in the wind, against a background of

# Animation of condensation algorithm

[Isard 1998]

# Applications

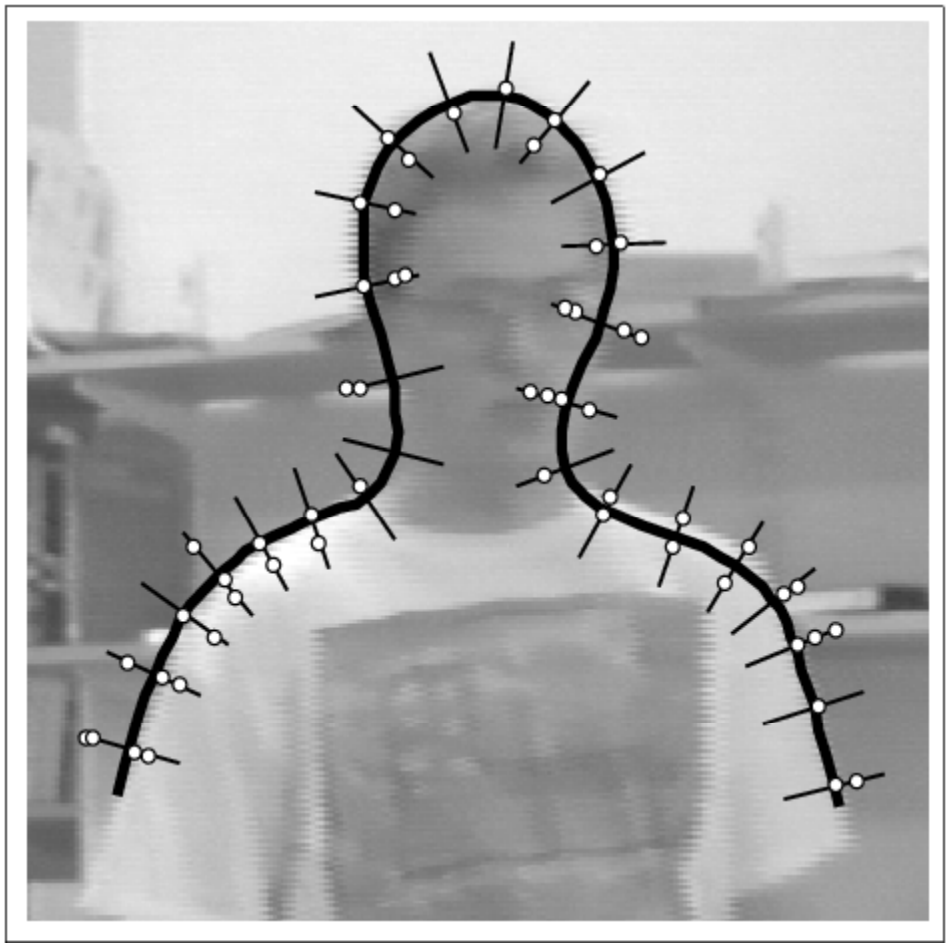
Tracking

- hands
- bodies
- Leaves

What might we expect?

Reliable, robust, slow

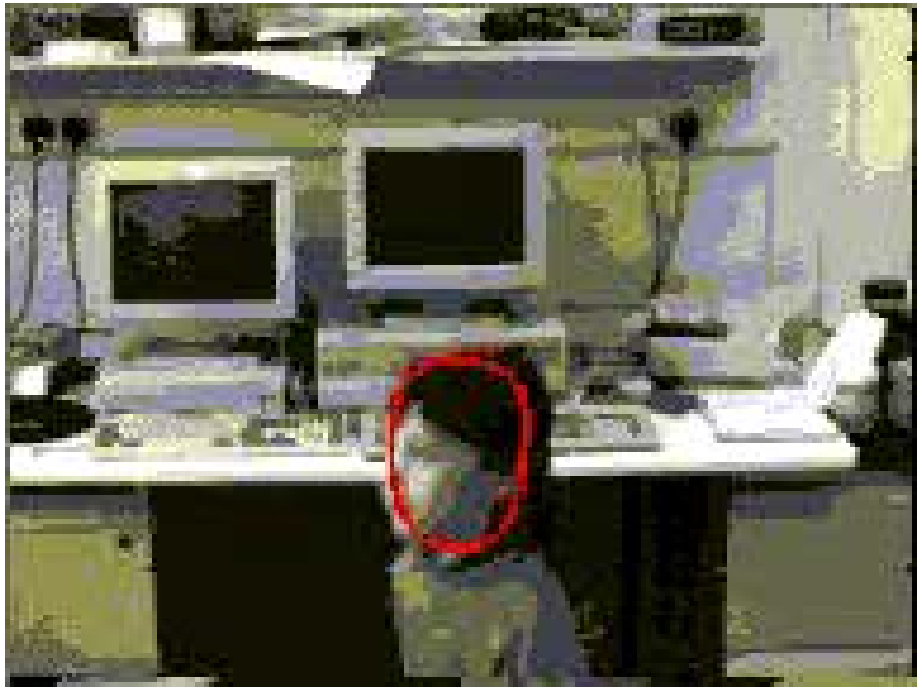
# Contour tracking

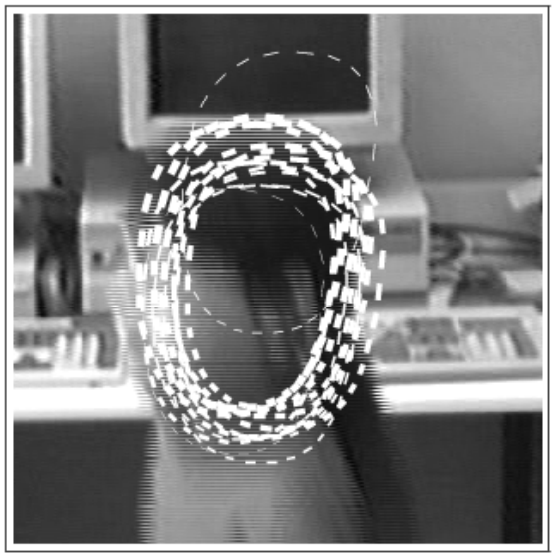




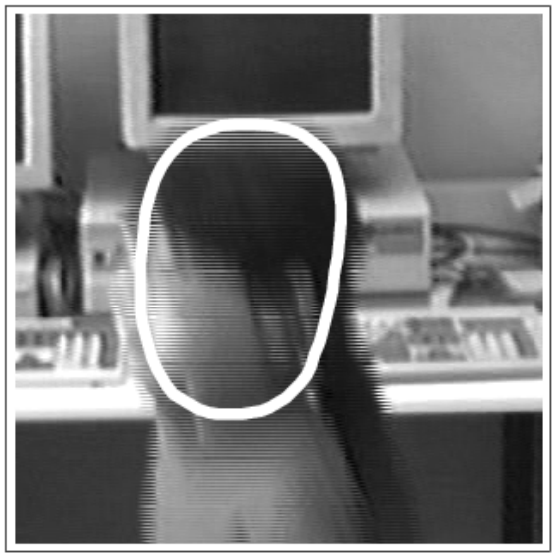
[Isard 1998]

# Head tracking





(a)



(b)

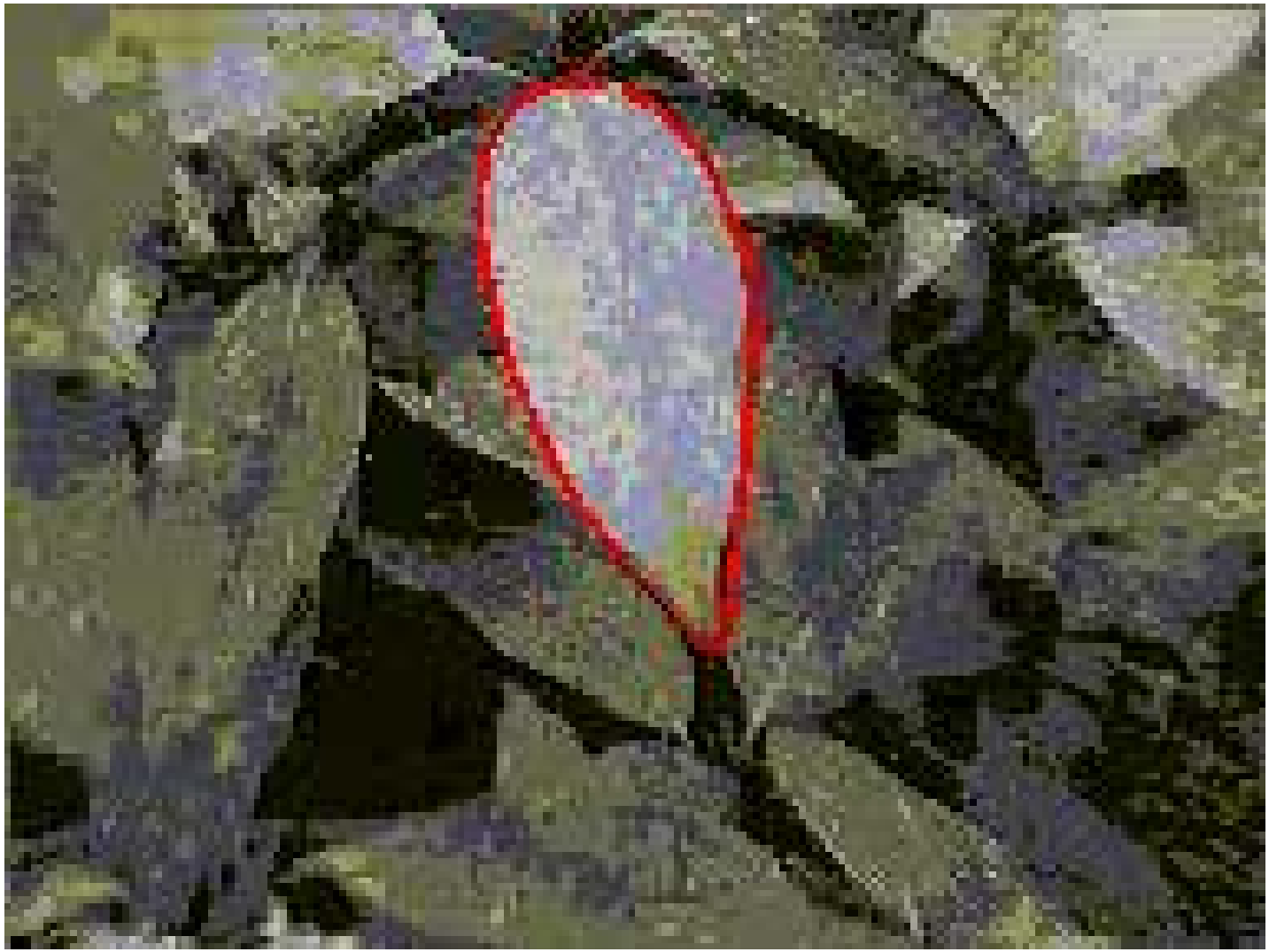
Picture of the states represented by the top weighted particles

The mean state



[Isard 1998]

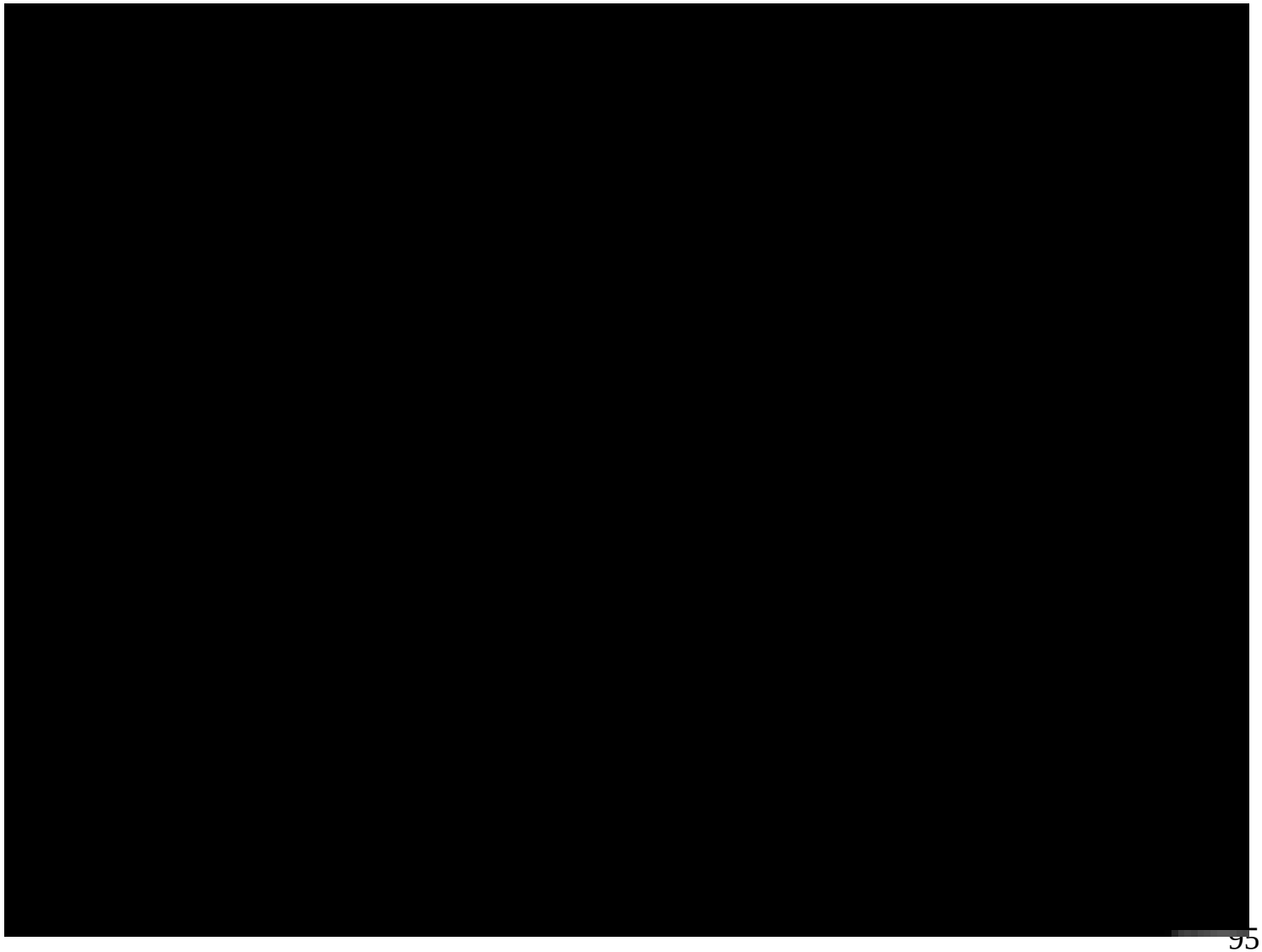
# Leaf tracking





[Isard 1998]

# Hand tracking





[Isard 1998]

# Desired operations with a probability density

- Expected value
- Marginalization
- Bayes rule

# Computing an expectation using sampled representation

$$\mathrm{E}_{p_f}[g] = \int g(\boldsymbol{U}) p_f(\boldsymbol{U}) d\boldsymbol{U}$$

using

$$p_f(x) = \sum_i w^i \delta(x - u^i)$$

$$\approx \frac{\sum_i g(\boldsymbol{u}_i) w_i}{\sum_i w_i}$$

# Marginalizing a sampled density

If we have a sampled representation of a joint density

$$\left\{((\boldsymbol{m}^i, \boldsymbol{n}^i), w^i)\right\}$$

and we wish to marginalize over one variable:

$$p_f(\boldsymbol{M}) = \int p_f(\boldsymbol{M}, \boldsymbol{N}) d\boldsymbol{N}$$

we can simply ignore the corresponding components of the samples (!):

$$\begin{aligned}
\int g(\boldsymbol{M}) p_f(\boldsymbol{M}) d\boldsymbol{M} &= \int g(\boldsymbol{M}) \int p_f(\boldsymbol{M}, \boldsymbol{N}) d\boldsymbol{N} d\boldsymbol{M} \\
&= \int \int g(\boldsymbol{M}) p_f(\boldsymbol{M}, \boldsymbol{N}) d\boldsymbol{N} d\boldsymbol{M} \\
&\approx \frac{\sum_{i=1}^{N} g(\boldsymbol{m}^i) w^i}{\sum_{i=1}^{N} w^i}
\end{aligned}$$

# Marginalizing a sampled density

Assume we have a sampled representation of a distribution

$$p_f(\boldsymbol{M}, \boldsymbol{N})$$

given by

$$\left\{((\boldsymbol{m}^i, \boldsymbol{n}^i), w^i)\right\}$$

Then

$$\left\{(\boldsymbol{m}^i, w^i)\right\}$$

is a representation of the marginal,

$$\int p_f(\boldsymbol{M}, \boldsymbol{N}) d\boldsymbol{N}$$

# Sampled Bayes rule

Assume we have a representation of $p(\boldsymbol{U})$ as

$$\{(\boldsymbol{u}^i, w^i)\}$$

Assume we have an observation $\boldsymbol{V} = \boldsymbol{v}_0$,

and a likelihood model $p(\boldsymbol{V}|\boldsymbol{U})$.

The posterior, $p(\boldsymbol{U}|\boldsymbol{V} = \boldsymbol{v}_0)$ is represented by

k $p(V=v_0|U)p(U)$

$$\{(\boldsymbol{u}^i, w'^i)\}$$

where

$$w'^i = p(\boldsymbol{V} = \boldsymbol{v}_0|\boldsymbol{u}^i)w^i$$