6.891

Computer Vision and Applications

Prof. Trevor. Darrell

Lecture 16: Tracking
- Density propagation
- Linear Dynamic models / Kalman filter
- Data association
- Multiple models

Readings: F&P Ch 17

Syllabus

15	4/1	Face Detection and Recognition II		Project proposal due
16	4/6	Segmentation and Clustering	Req: FP 14, 15.1-15.2	
17	4/8	Segmentation and Fitting	Req: FP 15.3-15.5, 16	PS3 out (4/7)
18	4/13	Tracking I	Req: FP 17	
19	4/15	Medical Imaging		
20	4/20			PS3 due, PS4 out
21	4/22	Darrell ILP Event Talk - Kresge		
22	4/27	Image-Based Rendering		
23	4/29	Example-based Parameter Estimation		PS4 due
24	5/4	Articulated Tracking and Shape Inference		EX2 out
25	5/6	Project Presentations 11-2pm		EX2 due
26	5/11	Project week--no class		
27	5/13	Projects week--no class		Project final report due (extension to 5/16 on request)

Tracking Applications

- Motion capture
- Recognition from motion
- Surveillance
- Targeting

Things to consider in tracking

What are the

- Real world dynamics
- Approximate / assumed model
- Observation / measurement process

Density propogation

- Tracking == Inference over time
- Much simplification is possible with linear dynamics and Gaussian probability models

Outline

- Recursive filters
- State abstraction
- Density propagation
- Linear Dynamic models / Kalman filter
- Data association
- Multiple models

Tracking and Recursive estimation

- Real-time / interactive imperative.
- Task: At each time point, re-compute estimate of position or pose.
  - At time n, fit model to data using time 0…n
  - At time n+1, fit model to data using time 0…n+1
- Repeat batch fit every time?

Recursive estimation

- Decompose estimation problem
  - part that depends on new observation
  - part that can be computed from previous history

- E.g., running average:

$$a_t = \alpha\, a_{t-1} + (1-\alpha)\, y_t$$

- Linear Gaussian models: Kalman Filter
- First, general framework…

Tracking

- Very general model:
  - We assume there are moving objects, which have an underlying state X
  - There are measurements Y, some of which are functions of this state
  - There is a clock
    - at each tick, the state changes
    - at each tick, we get a new observation
- Examples
  - object is ball, state is 3D position+velocity, measurements are stereo pairs
  - object is person, state is body configuration, measurements are frames, clock is in camera (30 fps)

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.

Tracking as induction

- Assume data association is done
  - we'll talk about this later; a dangerous assumption
- Do correction for the 0'th frame
- Assume we have corrected estimate for i'th frame
  - show we can do prediction for i+1, correction for i+1

Base case

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

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

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

Induction 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

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

Induction 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

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

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

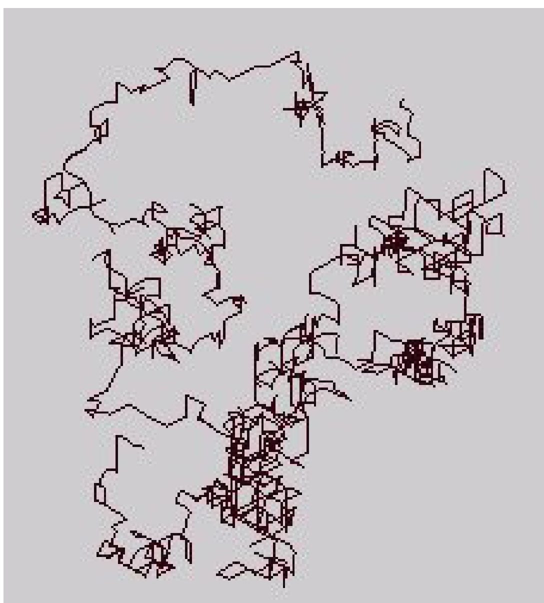
Examples

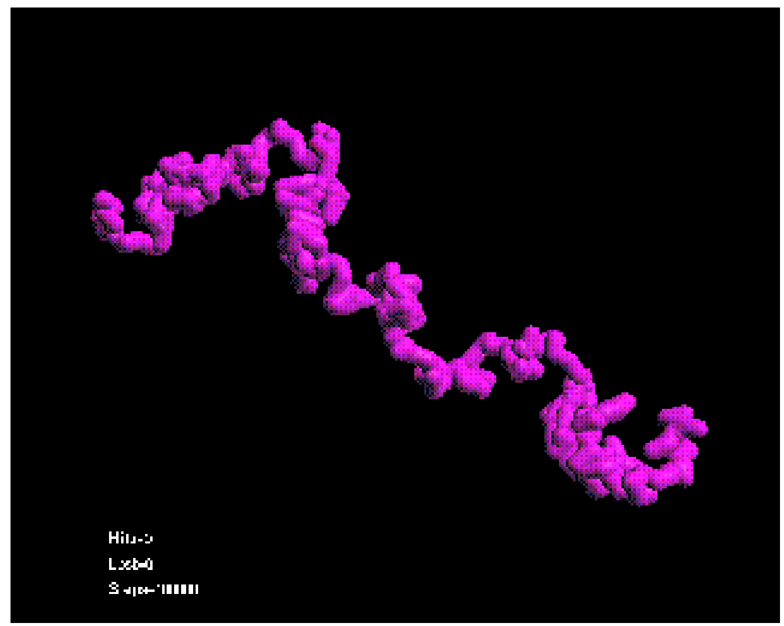
$$\mathbf{x}_i = N\left(\mathbf{D}_{i-1}\mathbf{x}_{i-1};\Sigma_{d_i}\right)$$

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

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

$$\mathbf{D} = \mathbf{Id}$$





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

Constant velocity

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

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

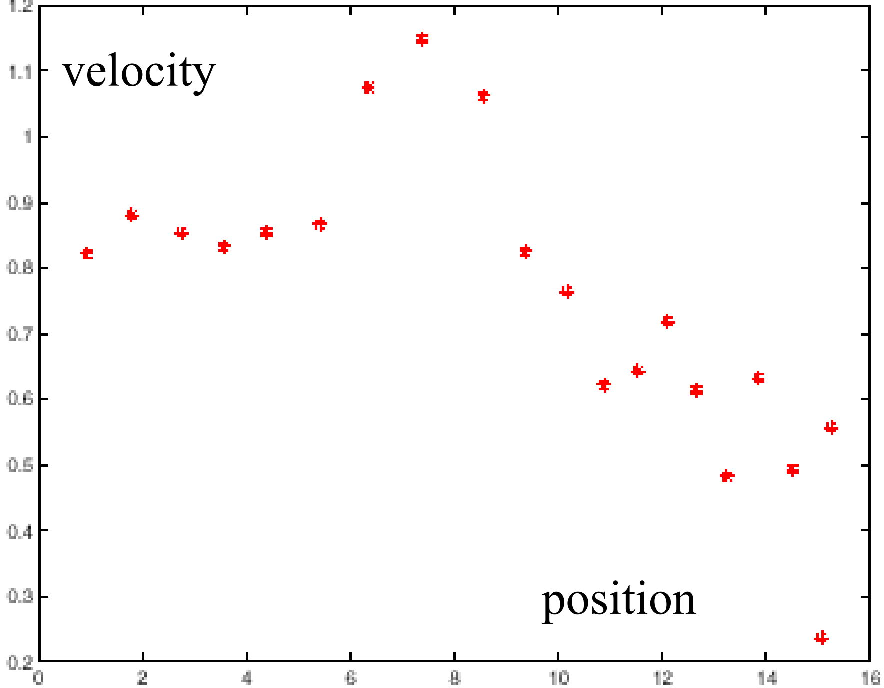
- We have

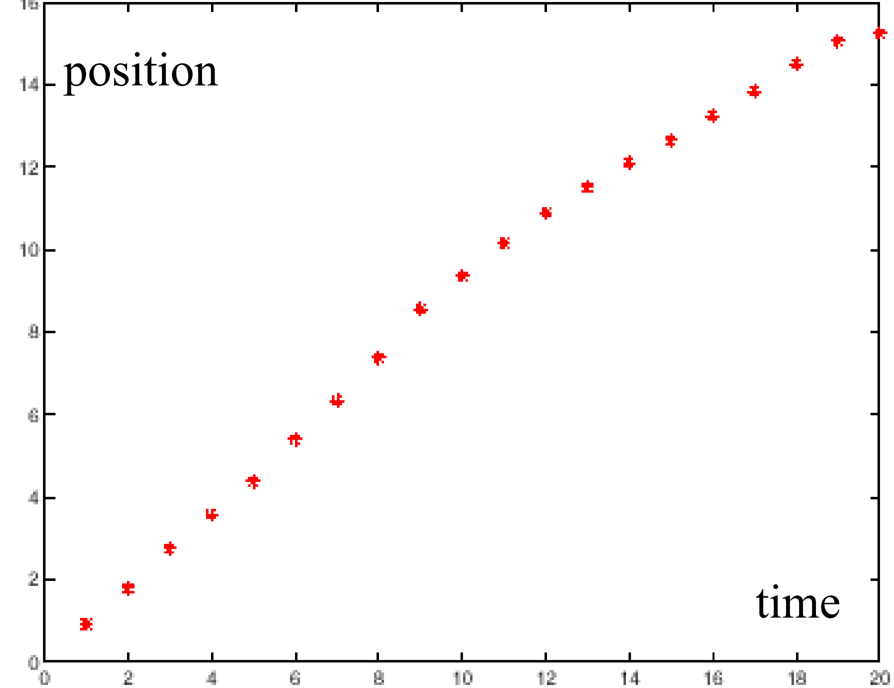
$$u_i = u_{i-1} + \Delta t v_{i-1} + \varepsilon_i$$
$$v_i = v_{i-1} + \varsigma_i$$

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

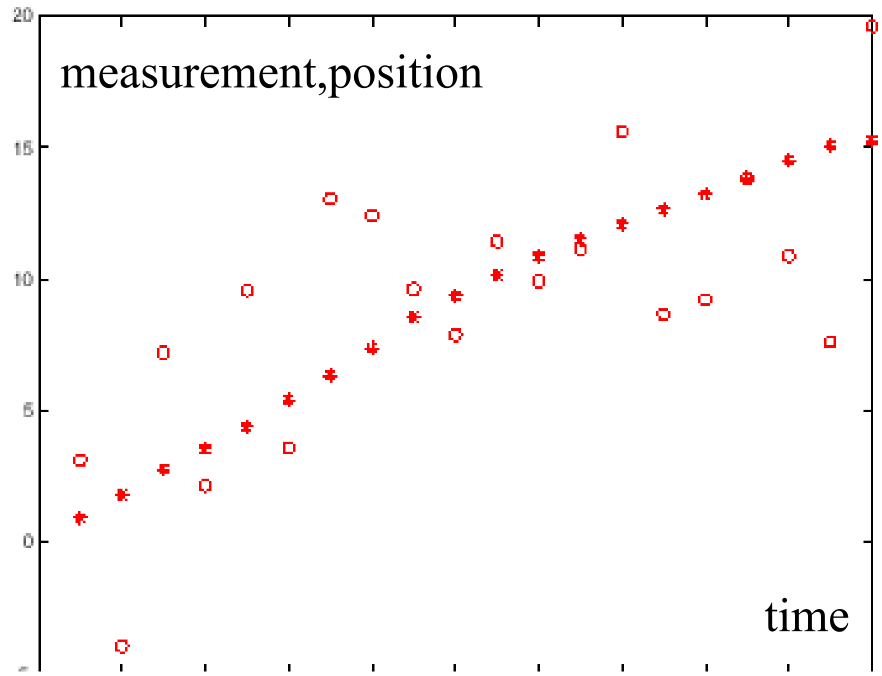
$$\begin{pmatrix} u \\ v \end{pmatrix}_i = \begin{pmatrix} 1 & \Delta t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}_{i-1} + \text{noise}$$

  – which is the form we had above





Constant
Velocity
Model



Constant acceleration

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

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

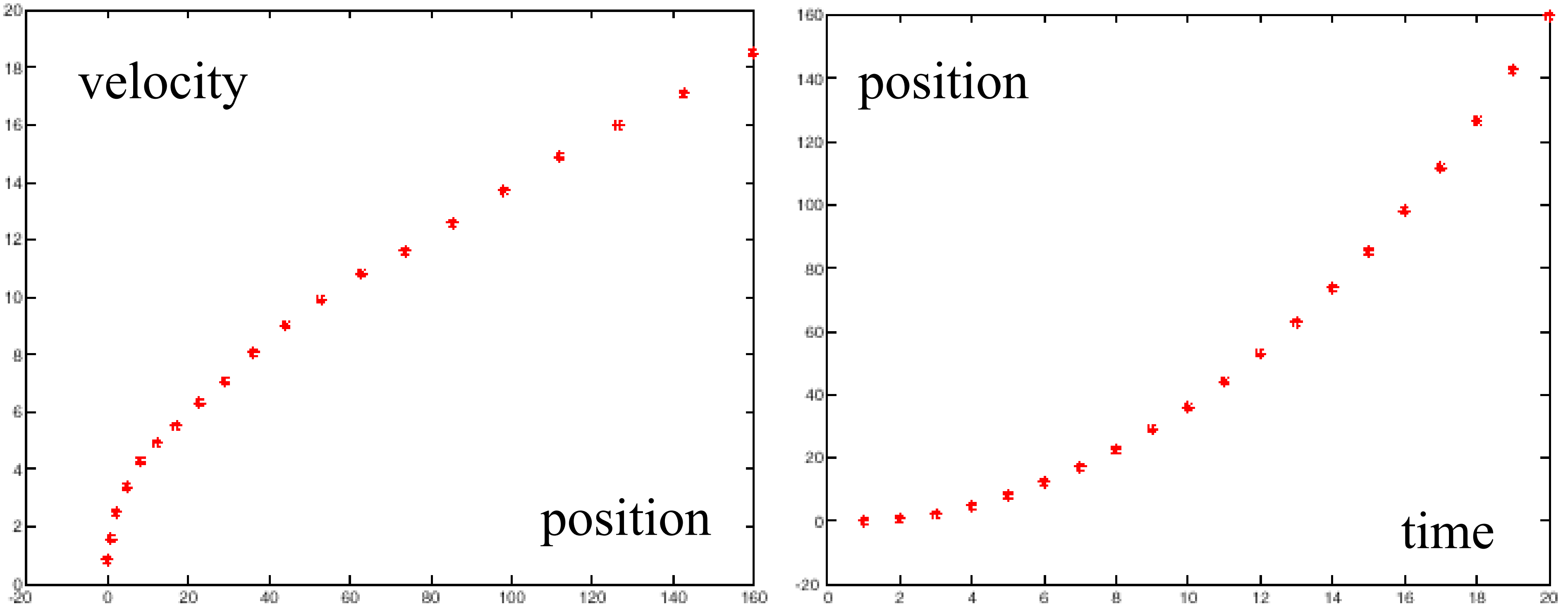
- We have

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

  - (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



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 Δt stepsize.)

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

Higher order models

- Independence assumption

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

- Velocity and/or acceleration augmented position
- Constant velocity model equivalent to

$$P(\boldsymbol{p}_i|\boldsymbol{p}_1,\ldots,\boldsymbol{p}_{i-1}) = N(\boldsymbol{p}_{i-1} + (\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}), \Sigma_{d_i})$$

  – velocity == $\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}$
  – acceleration == $(\boldsymbol{p}_{i-1} - \boldsymbol{p}_{i-2}) - (\boldsymbol{p}_{i-2} - \boldsymbol{p}_{i-3})$
  – could also use $\boldsymbol{p}_{i-4}$, etc.

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

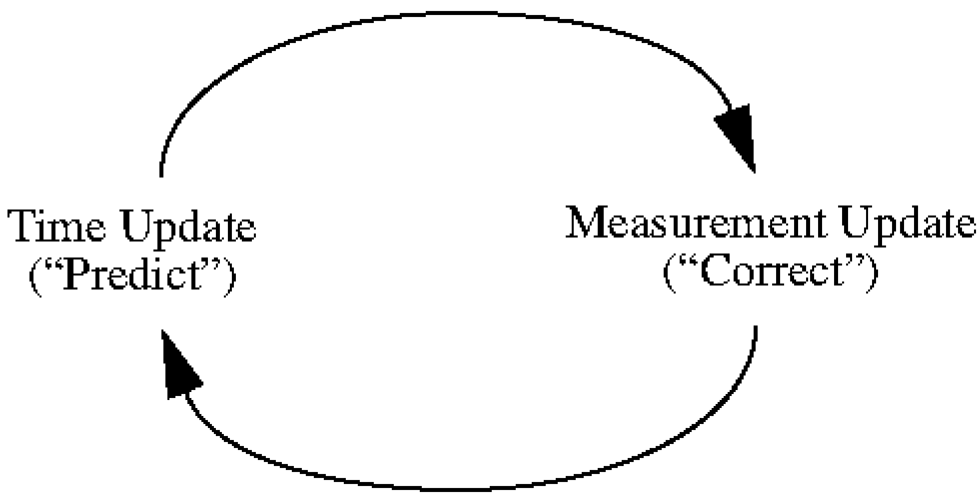
- **Prediction:** we have seen $\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{i-1}$ — what state does this set of measurements predict for the i'th frame? to solve this problem, we need to obtain a representation of $P(\boldsymbol{X}_i | \boldsymbol{Y}_0 = \boldsymbol{y}_0, \ldots, \boldsymbol{Y}_{i-1} = \boldsymbol{y}_{i-1})$.

- **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

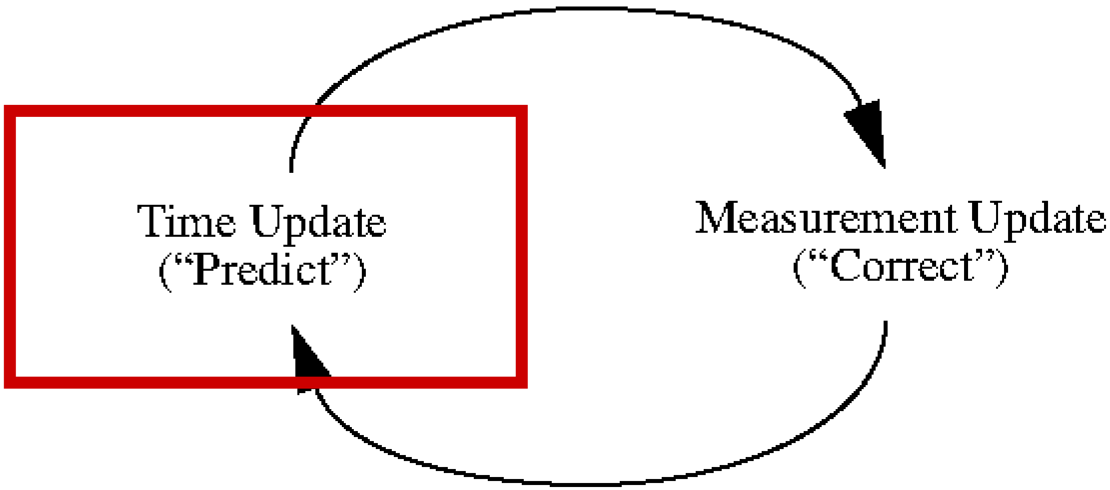
mean of $P(X_i | y_0, \ldots, y_{i-1})$ as $\overline{X}_i^-$ ← Predicted mean

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 σ_i^-

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

The Kalman Filter



Prediction for 1D Kalman filter

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

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

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

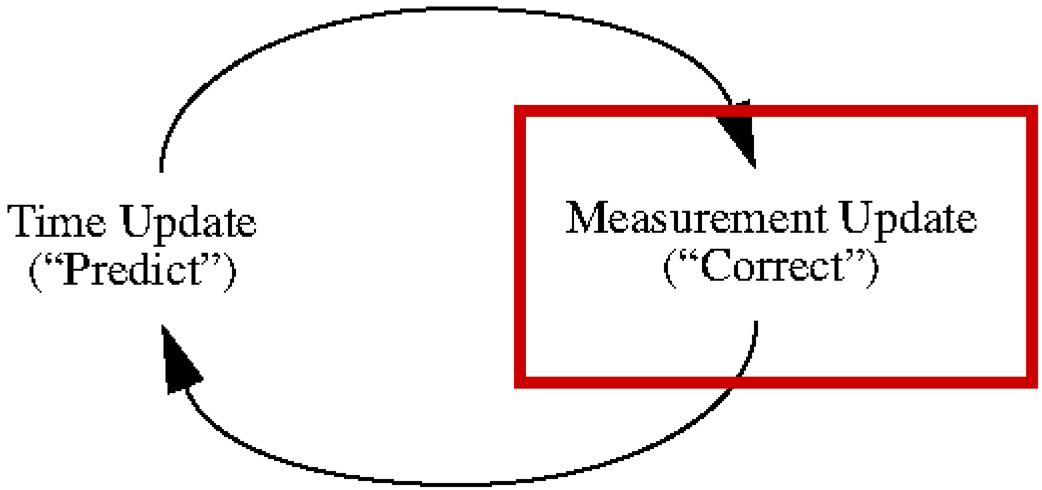
$$y_i \sim N(m_i x_i, \sigma_{m_i})$$

Start Assumptions: $\overline{x}_0^-$ and σ_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



Correction 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
– σ 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 σ_0^- are known
Update Equations: Prediction

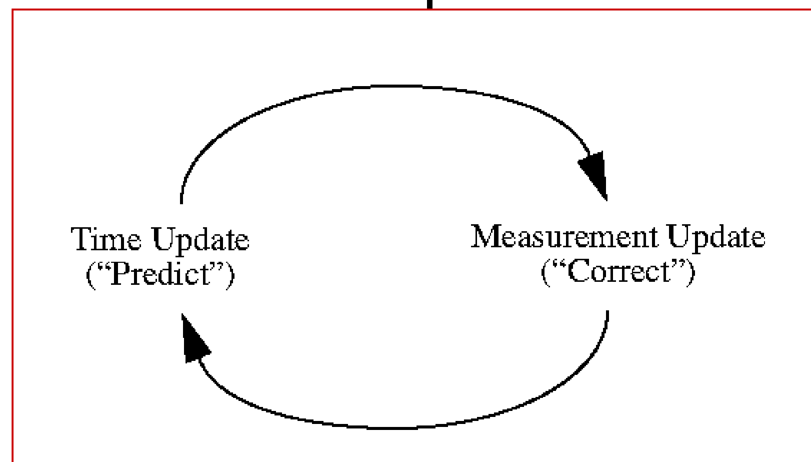
$$\overline{x}_i^- = d_i \overline{x}_{i-1}^+$$

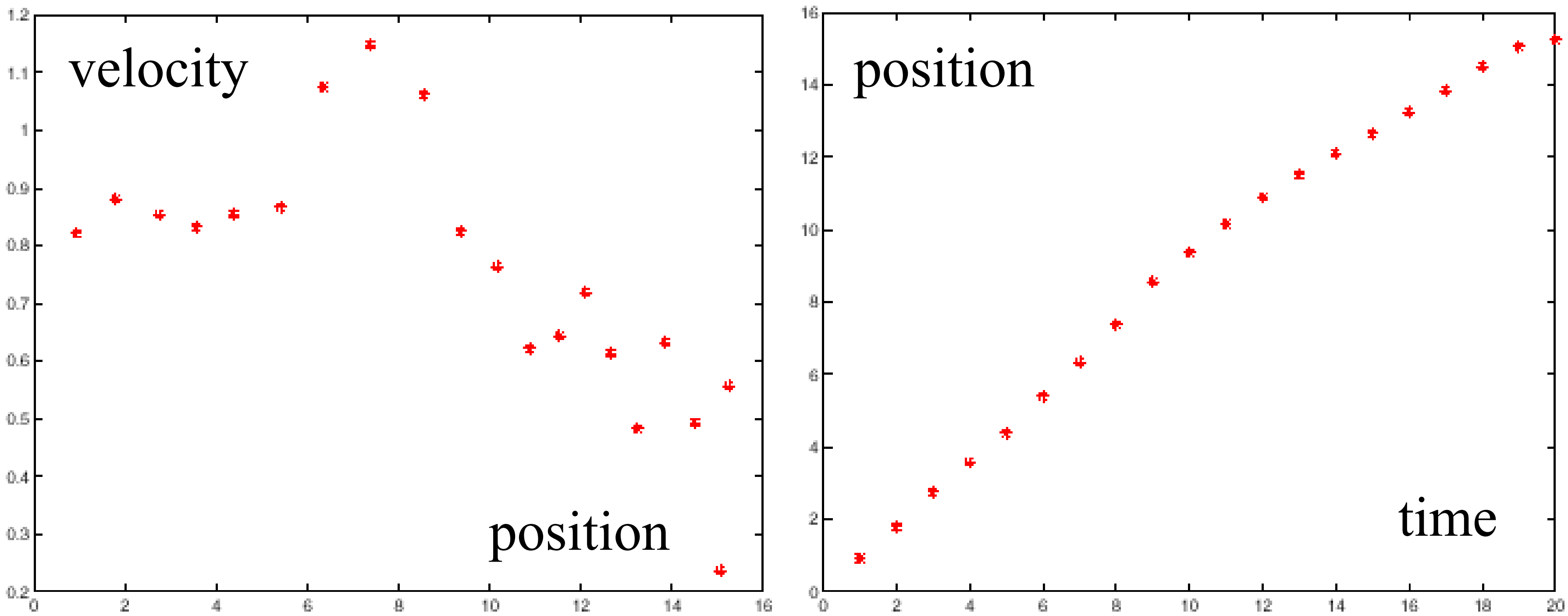
$$\sigma_i^- = \sqrt{\sigma_{d_i}^2 + (d_i \sigma_{i-1}^+)^2}$$

Update Equations: Correction

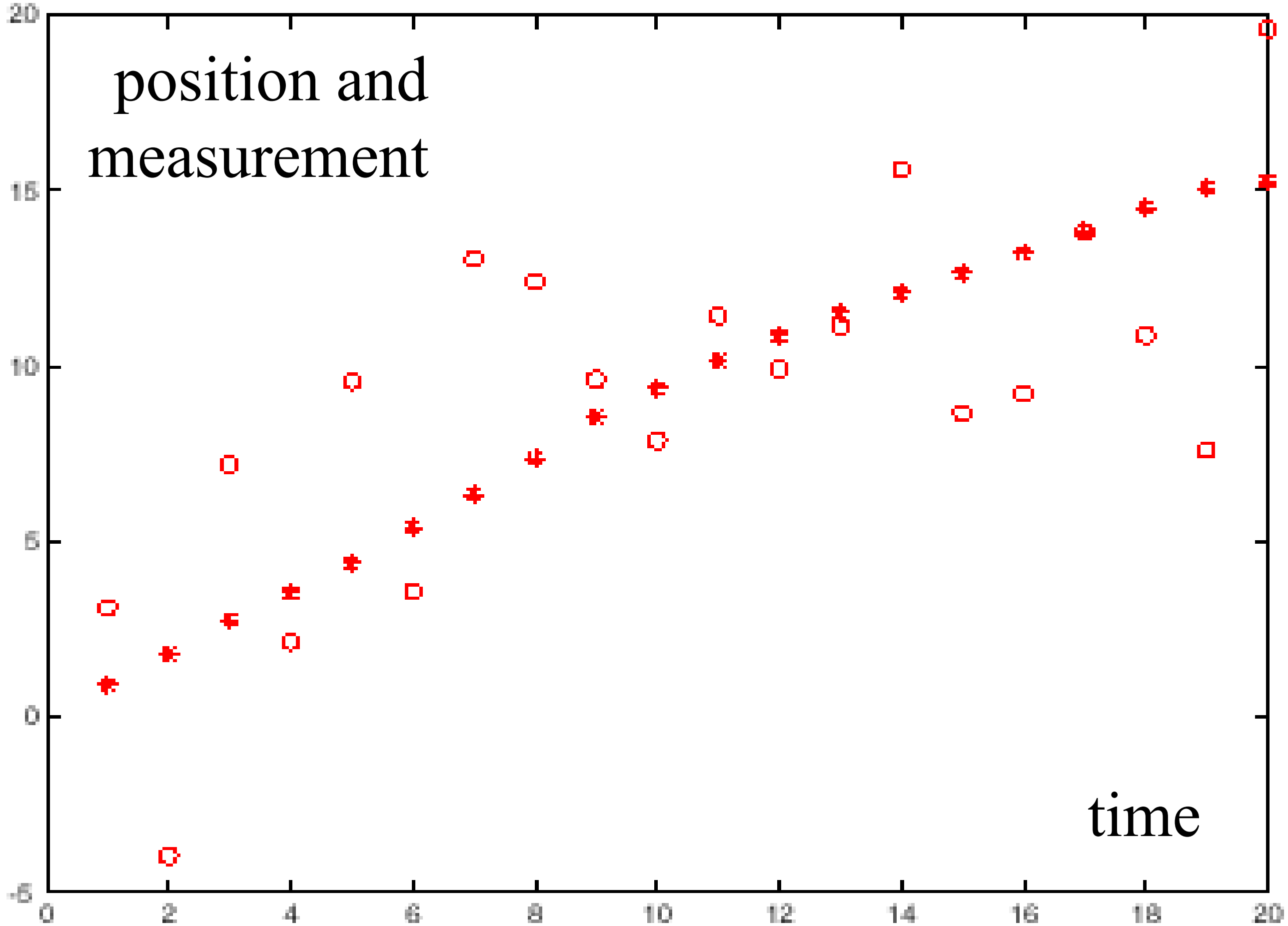
$$x_i^+ = \left(\frac{\overline{x}_i^- \sigma_{m_i}^2 + m_i y_i (\sigma_i^-)^2}{\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2} \right)$$

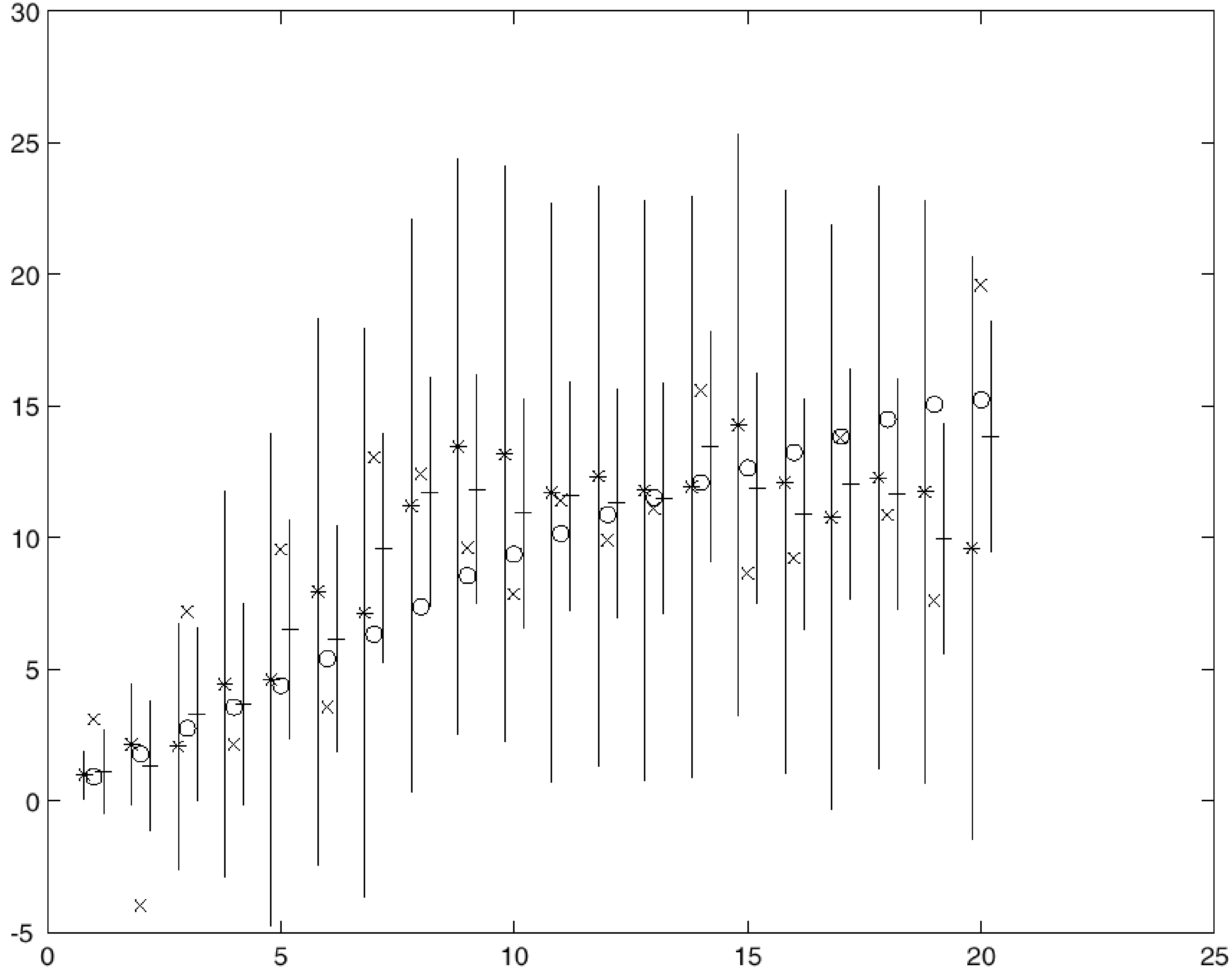
$$\sigma_i^+ = \sqrt{\left(\frac{\sigma_{m_i}^2 (\sigma_i^-)^2}{(\sigma_{m_i}^2 + m_i^2 (\sigma_i^-)^2)} \right)}$$



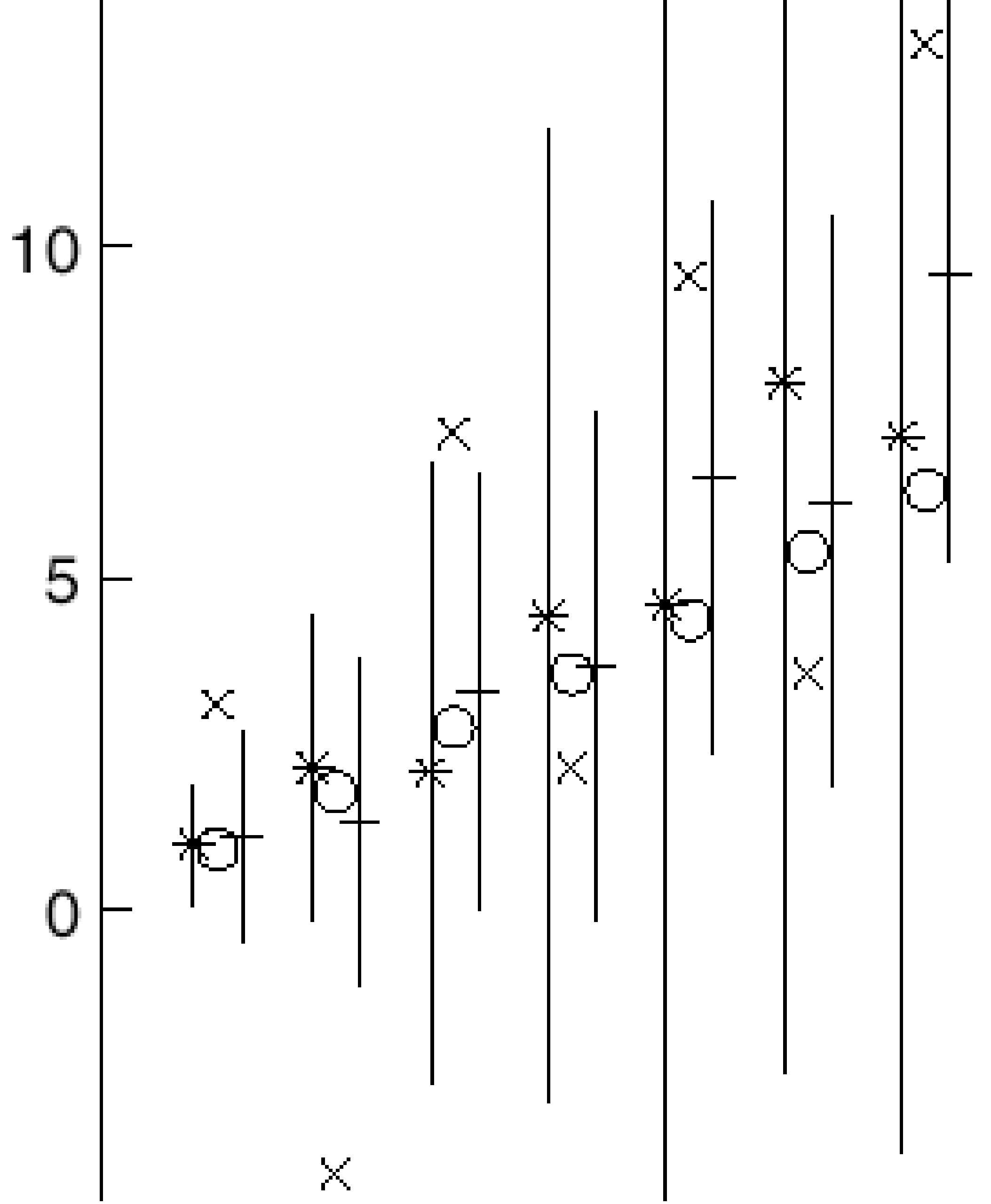


Constant
Velocity
Model


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

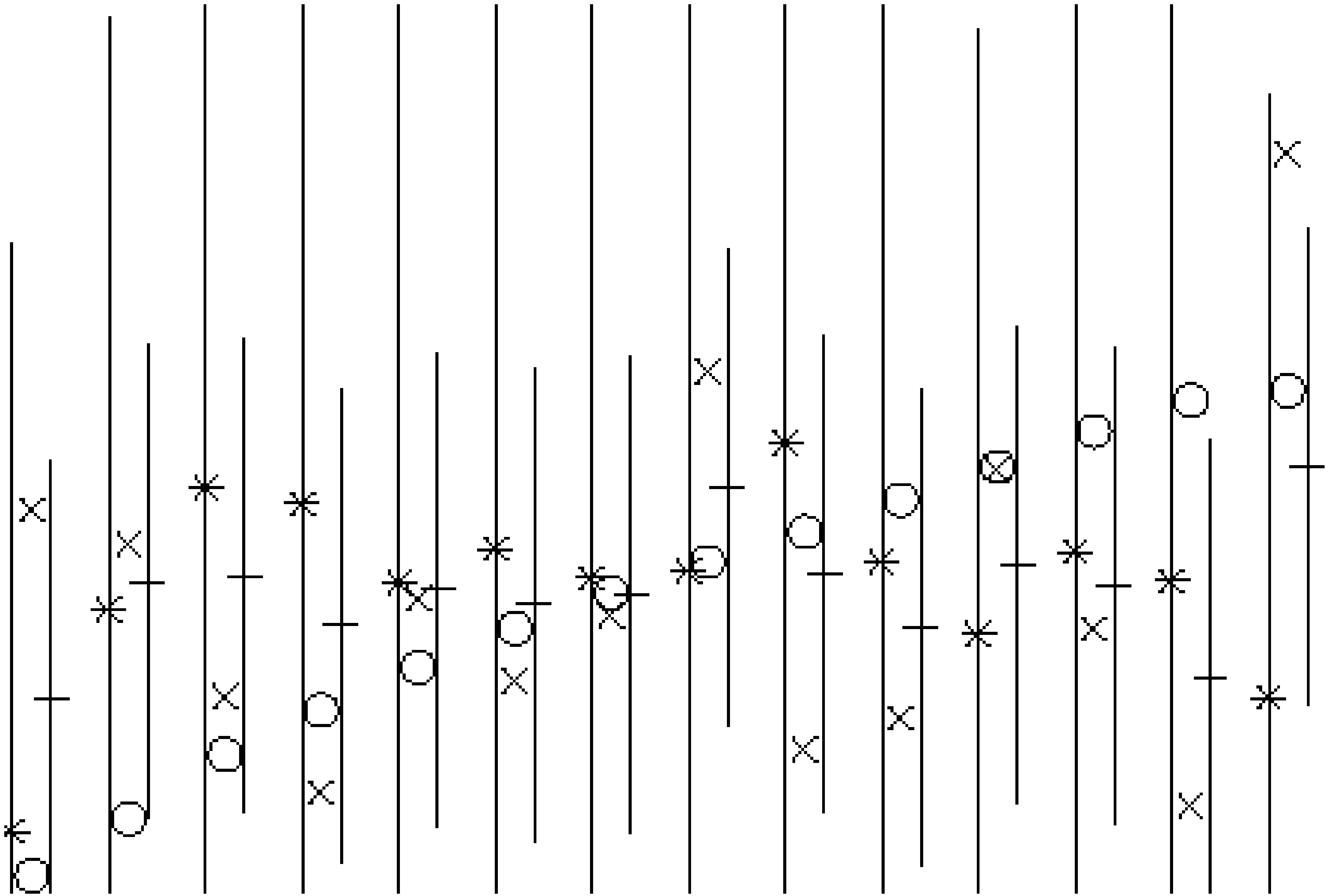


The *-s give $\overline{\boldsymbol{x}}_i^-$, +-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars



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

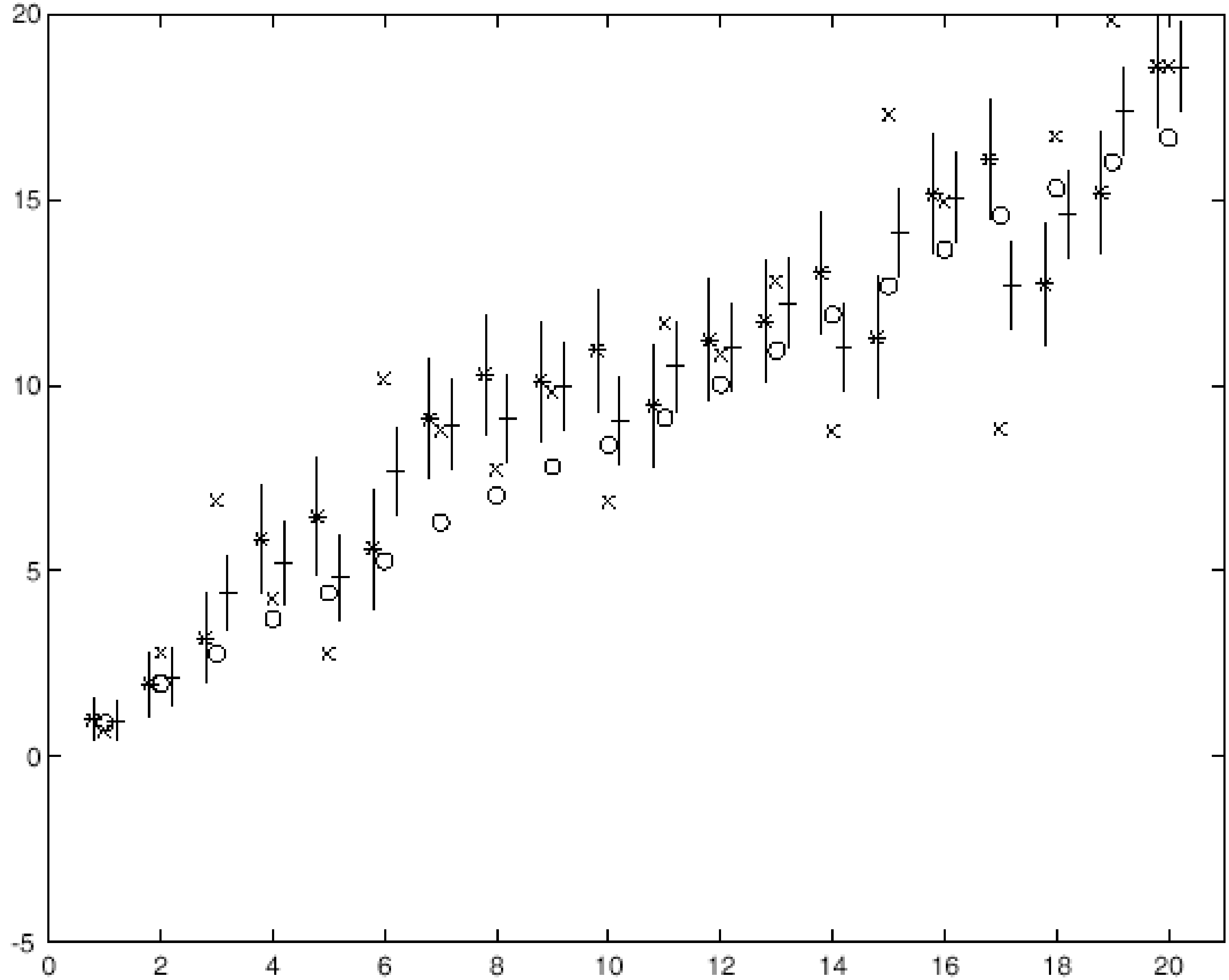


The o-s give state, x-s measurement.

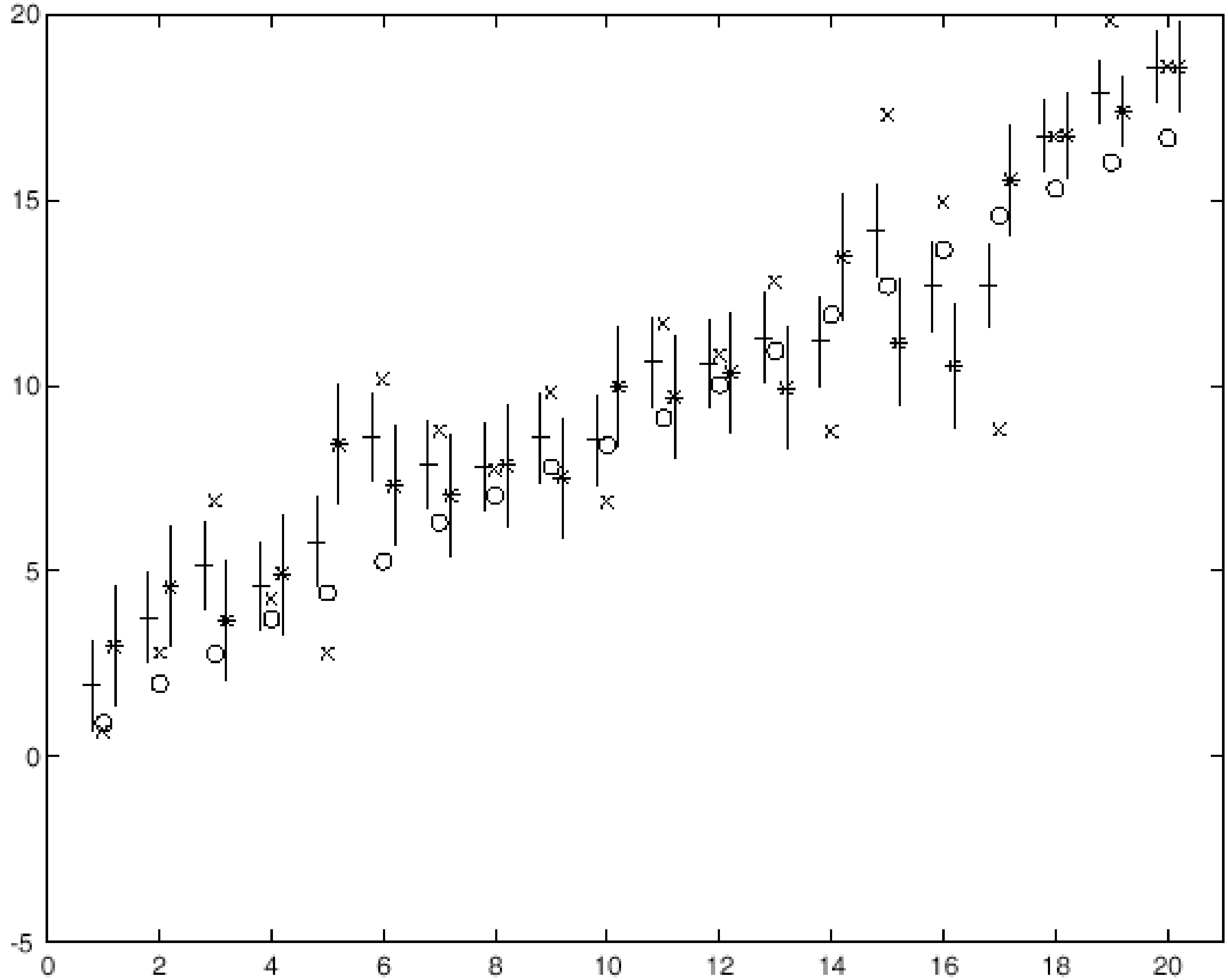
The $*$-s give $\overline{\boldsymbol{x}}_i^-$, $+$-s give $\overline{\boldsymbol{x}}_i^+$, vertical bars are 3 standard deviation bars

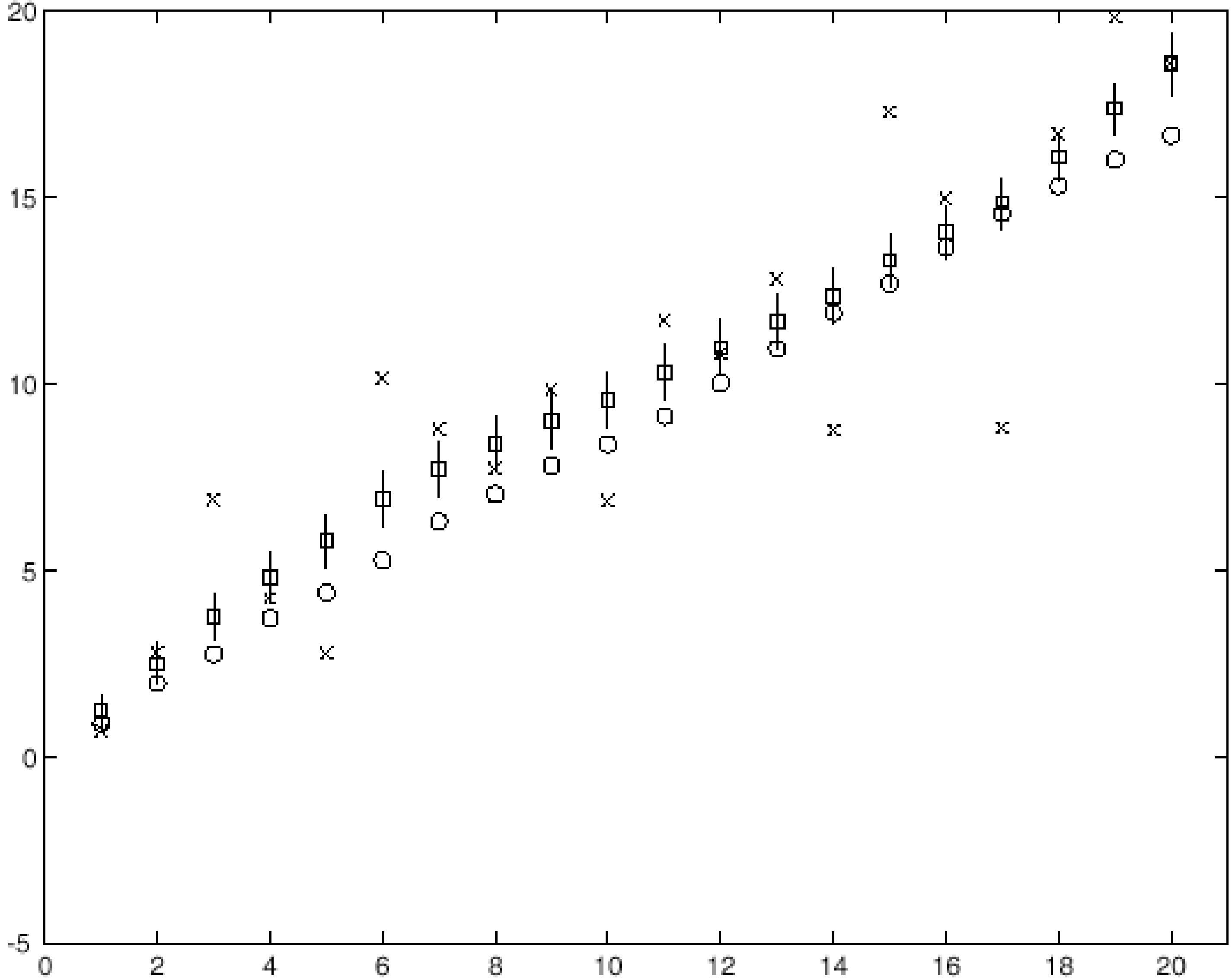
Smoothing

- 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



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


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



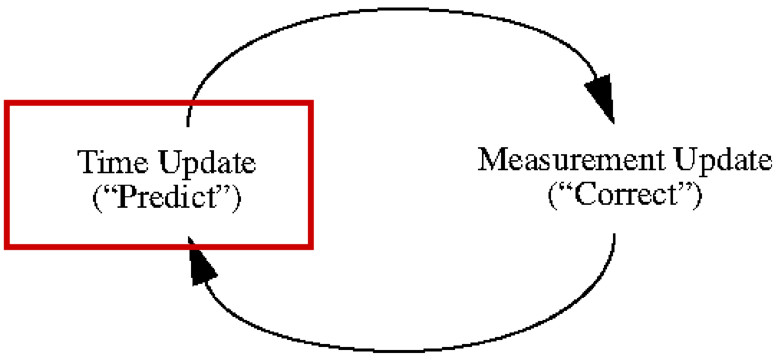
20
15
10
5
0
-5
0
2
4
6
8
10
12
14
16
18
20

n-D

Generalization to n-D is straightforward but more complex.

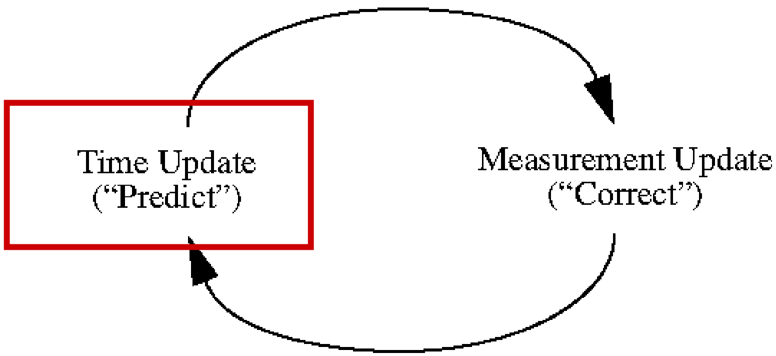
n-D

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:

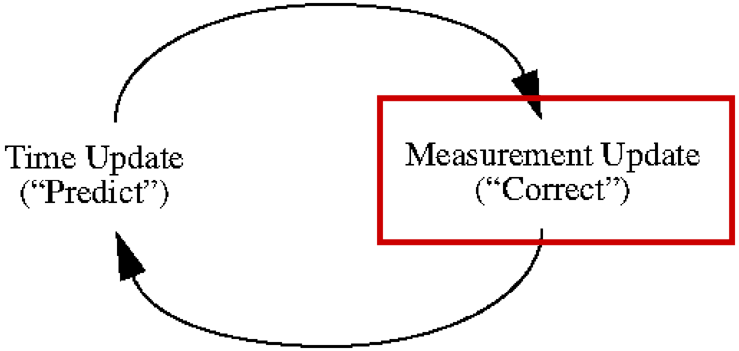
$$\overline{\boldsymbol{x}}_i^- = \mathcal{D}_i \overline{\boldsymbol{x}}_{i-1}^+$$

- 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

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

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

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 Σ_0^- are known
Update Equations: Prediction

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

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

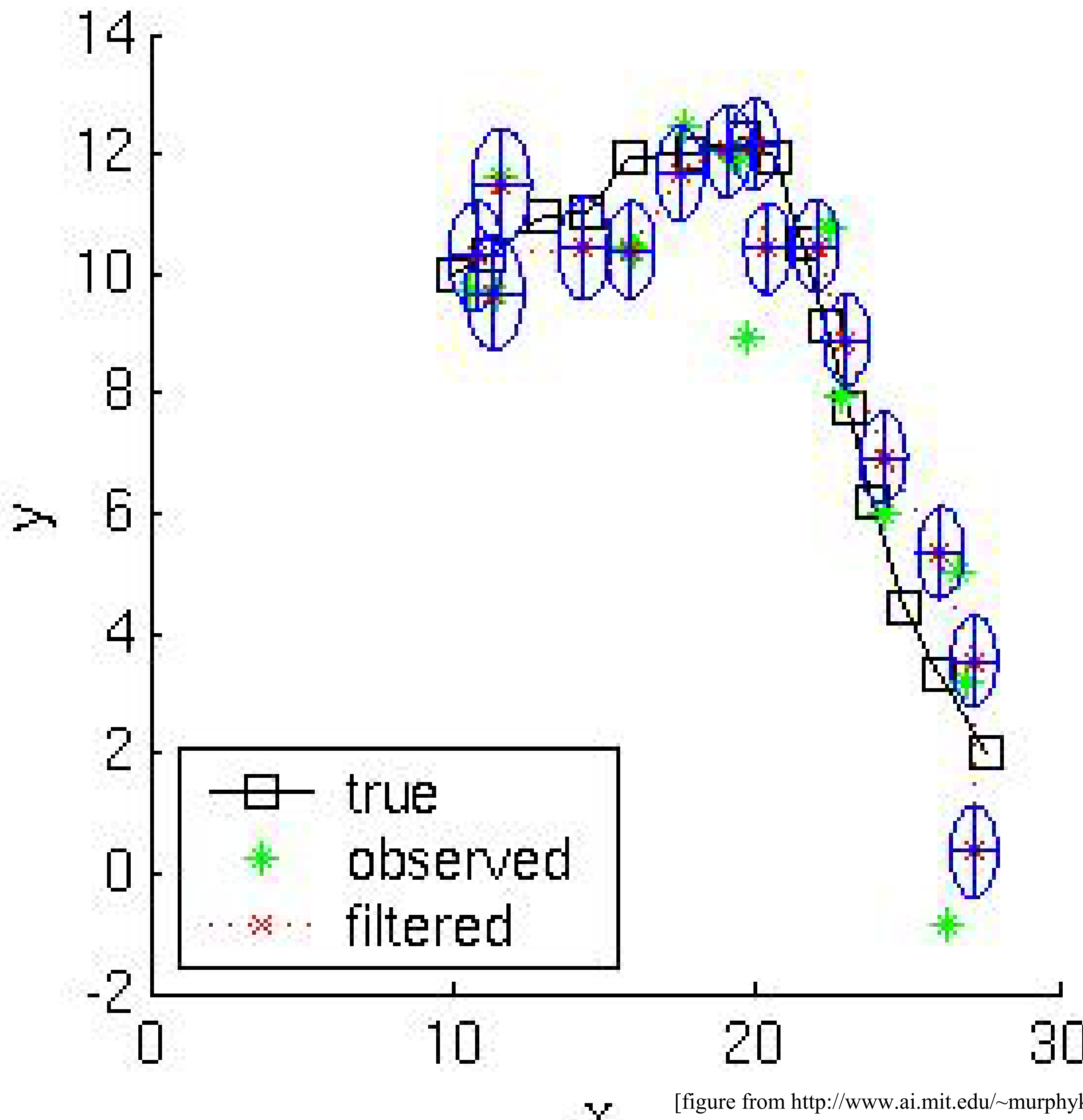
Update Equations: Correction

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

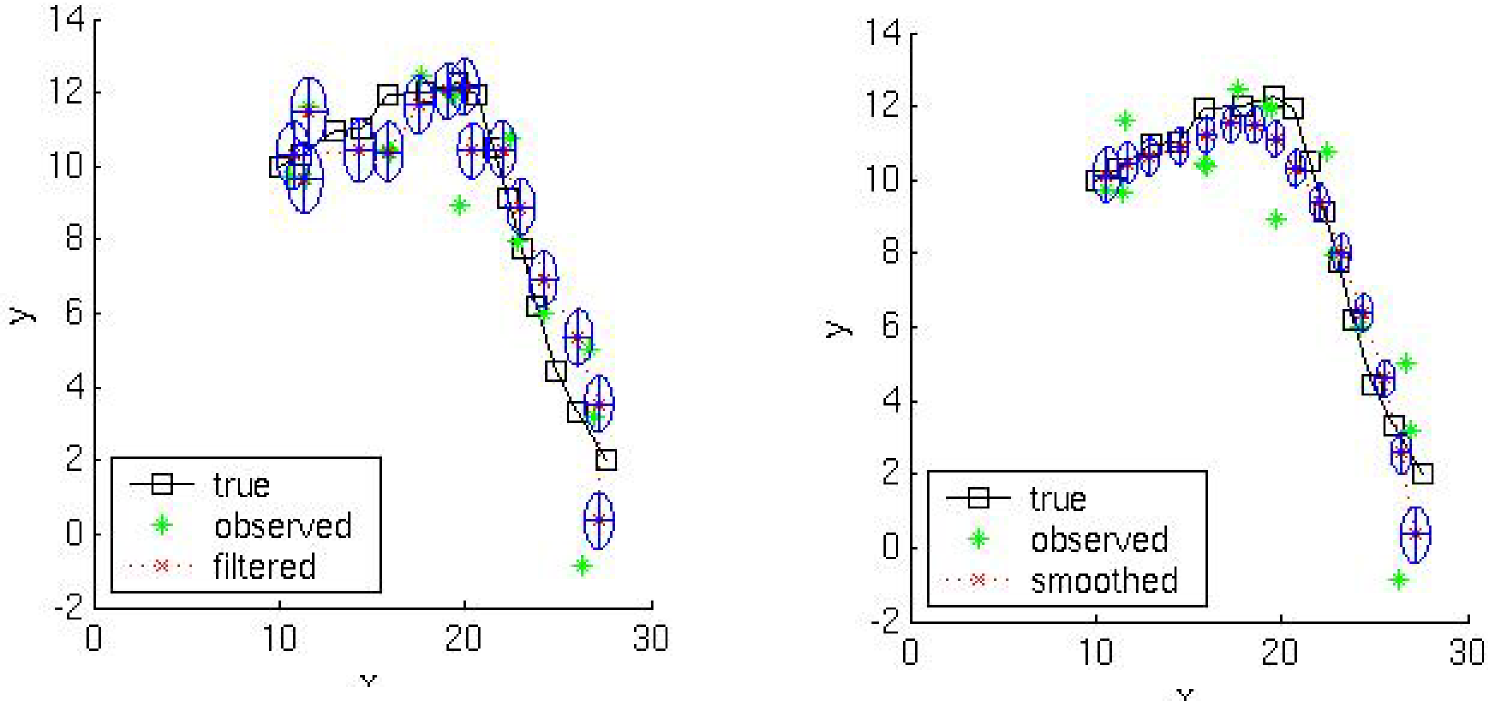
$$\overline{\boldsymbol{x}}_i^+ = \overline{\boldsymbol{x}}_i^- + \mathcal{K}_i \left[\boldsymbol{y}_i - \mathcal{M}_i \overline{\boldsymbol{x}}_i^- \right]$$

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

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]

Data Association

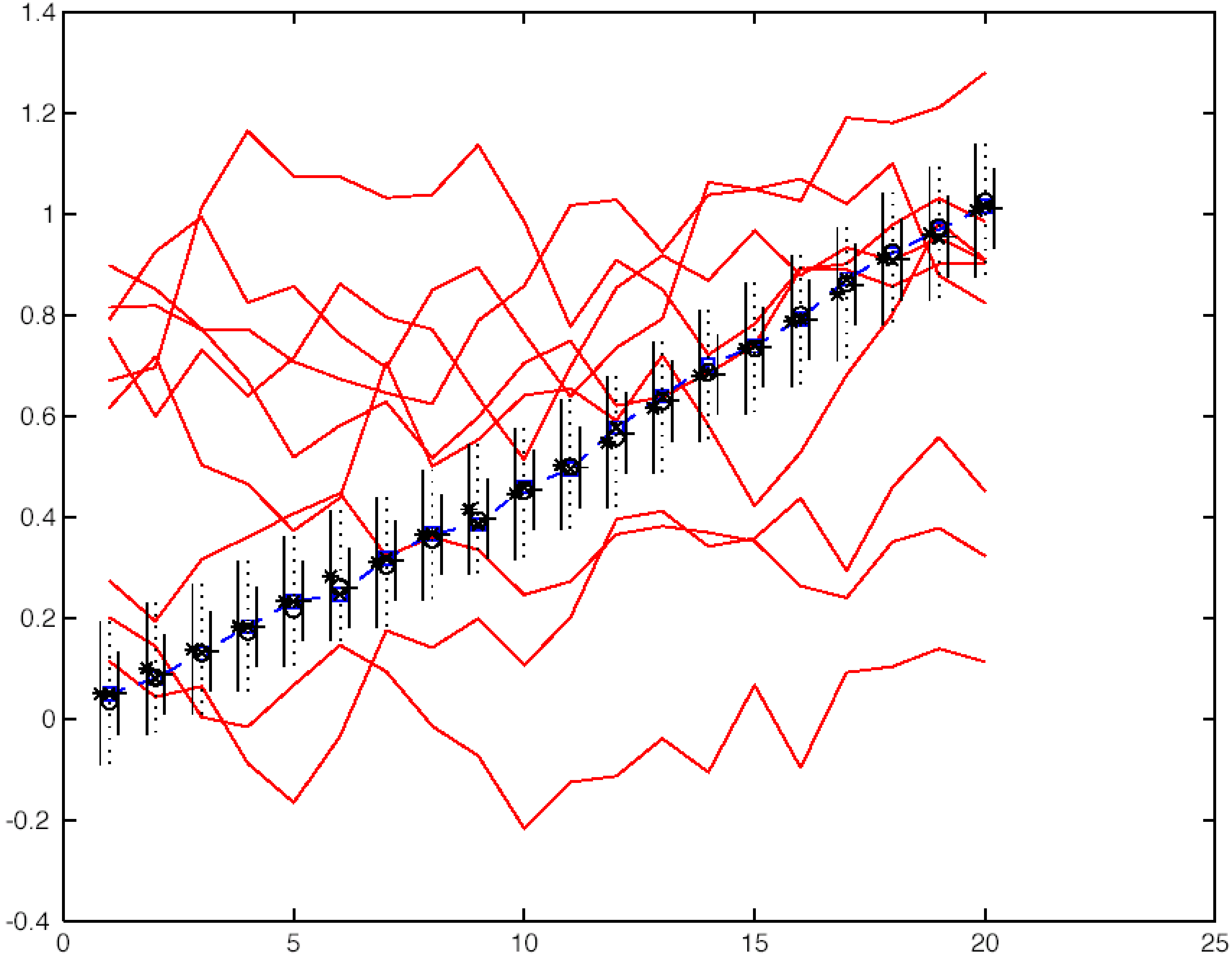
In real world y_i have clutter as well as data…

E.g., match radar returns to set of aircraft trajectories.

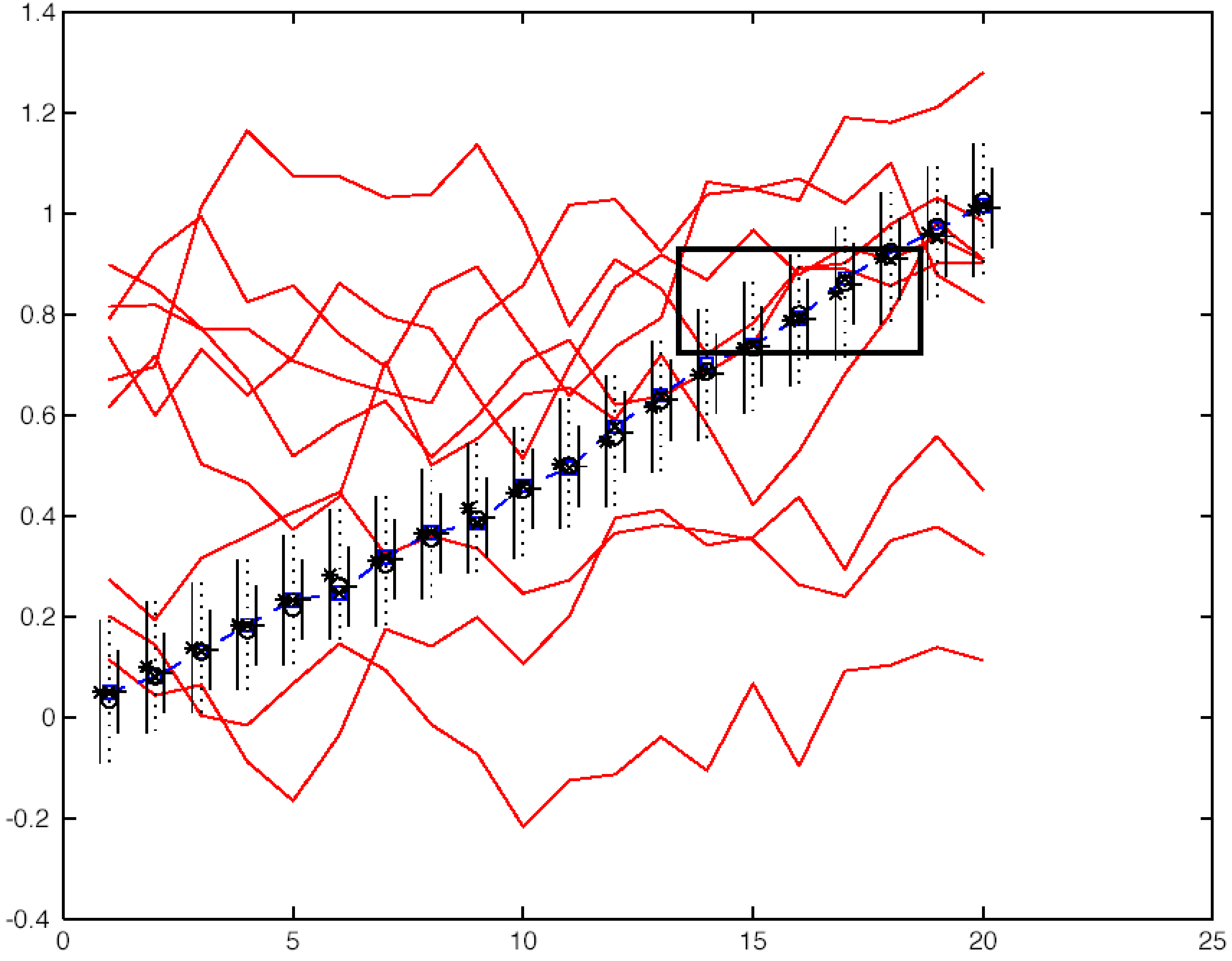
Data Association

Approaches:

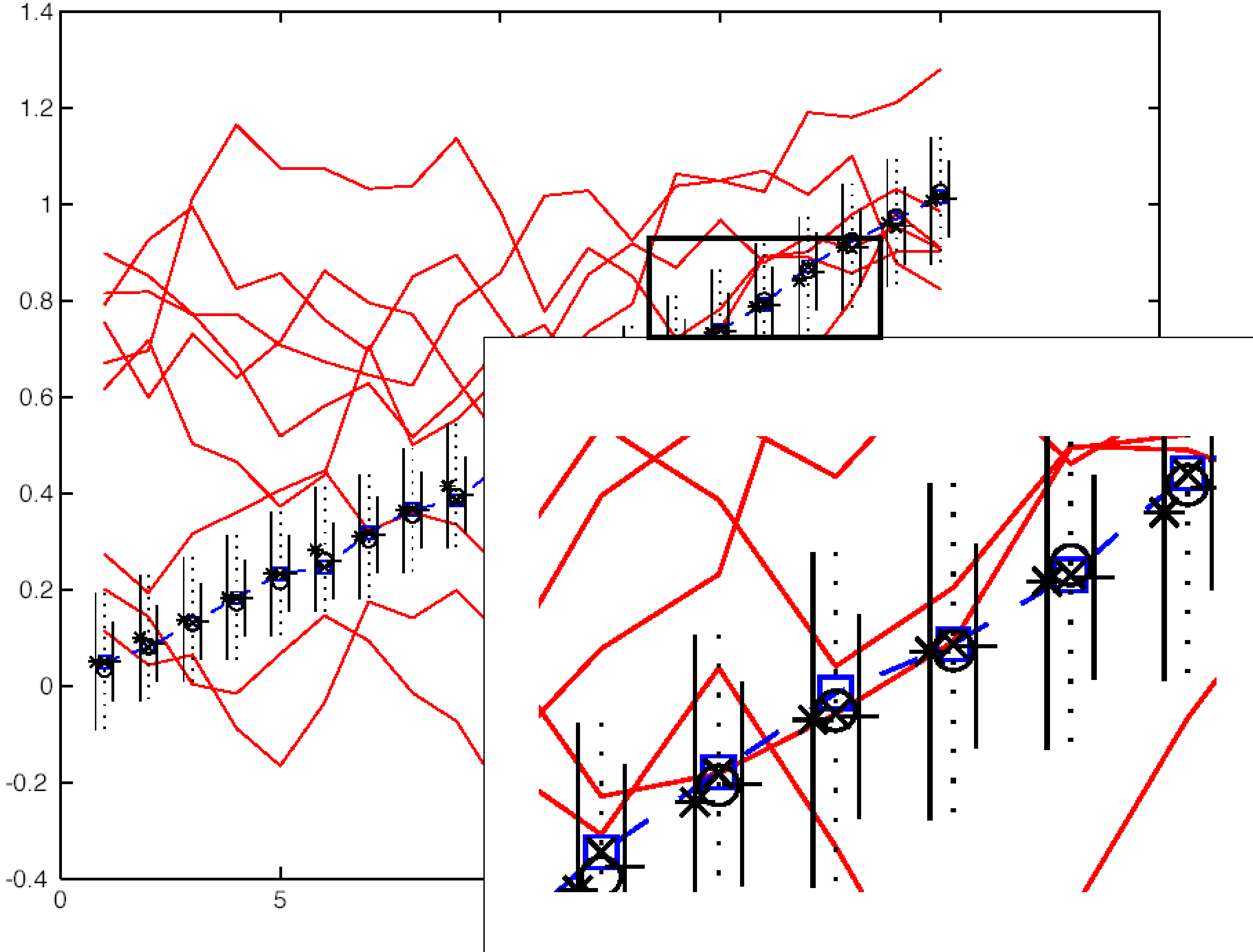
- Nearest neighbours
  - choose the measurement with highest probability given predicted state
  - popular, but can lead to catastrophe
- Probabilistic Data Association
  - combine measurements, weighting by probability given predicted state
  - gate using predicted state



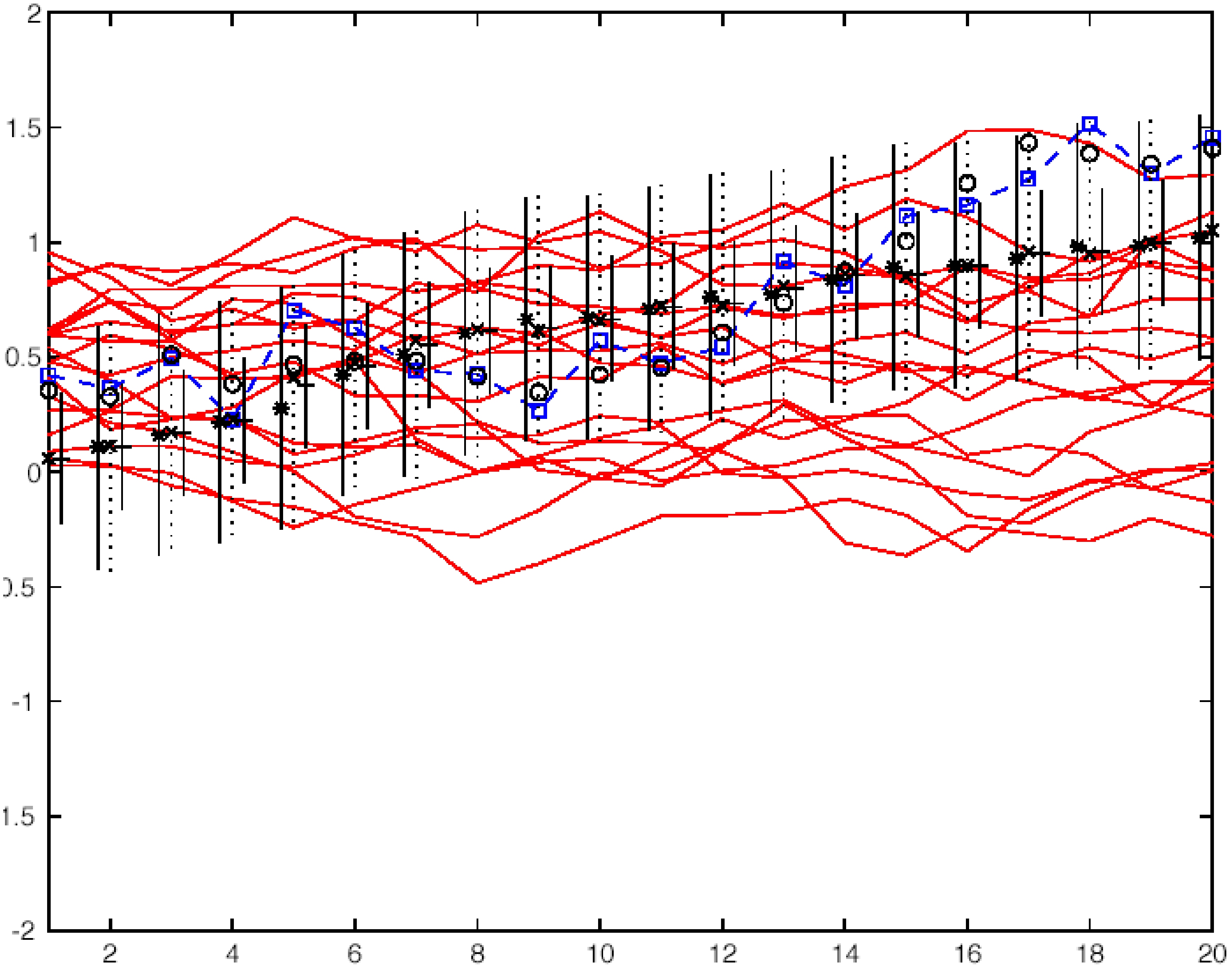
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25



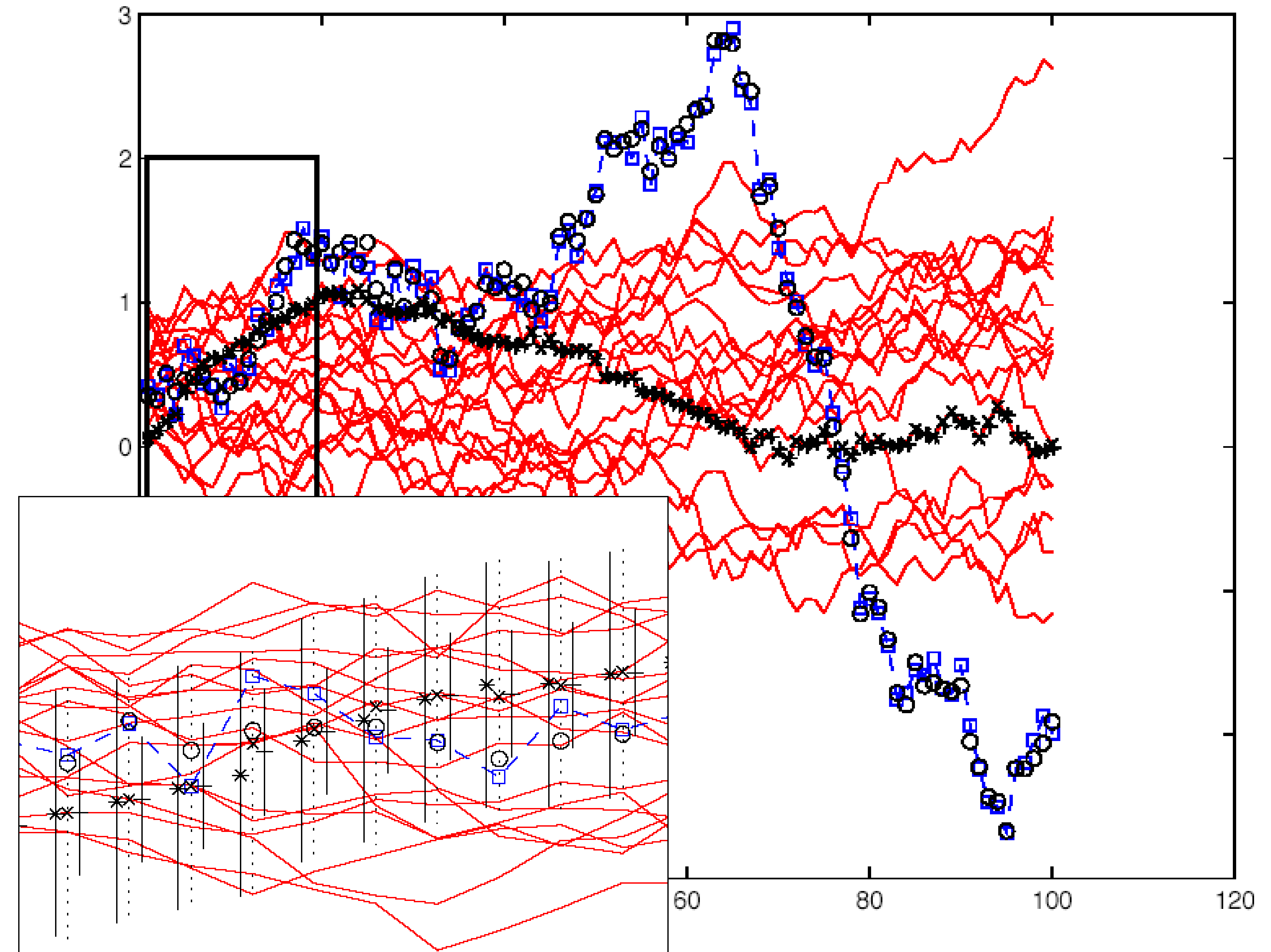
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25



1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5



2
1.5
1
0.5
0
0.5
-1
1.5
-2
2
4
6
8
10
12
14
16
18
20


3
2
1
0
60
80
100
120

Abrupt changes

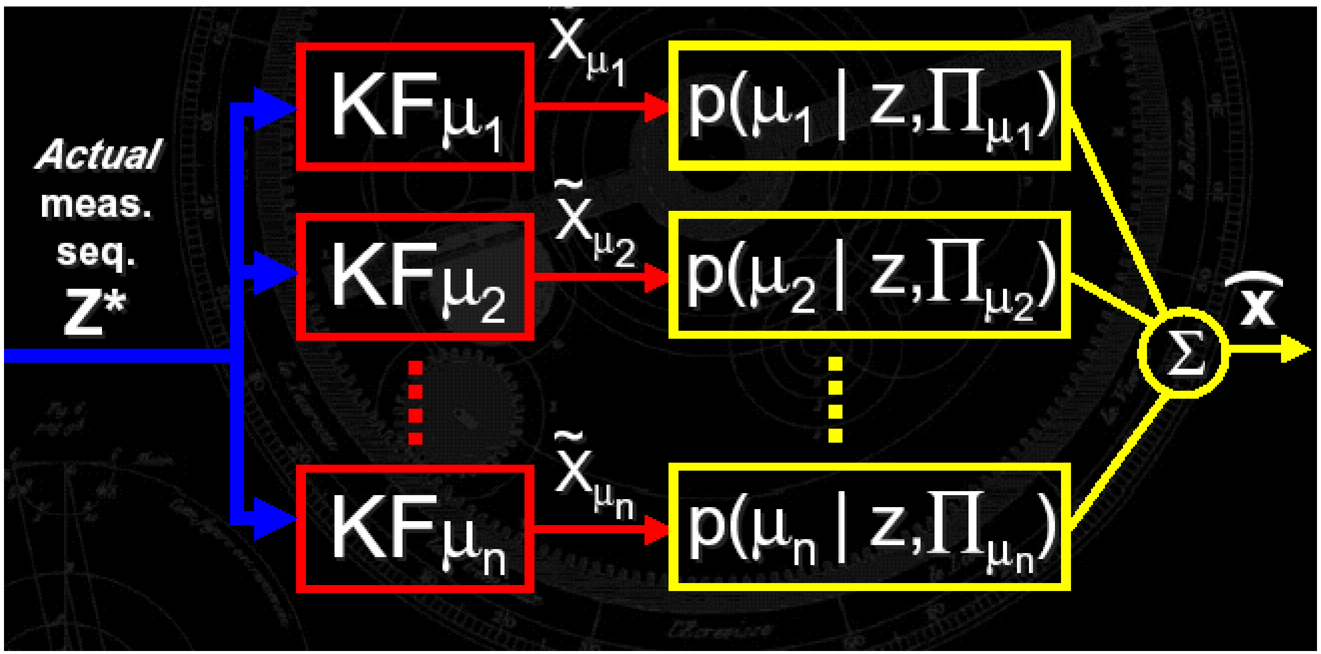
What if environment is sometimes unpredictable?

Do people move with constant velocity?

Test several models of assumed dynamics, use the best.

Multiple model filters

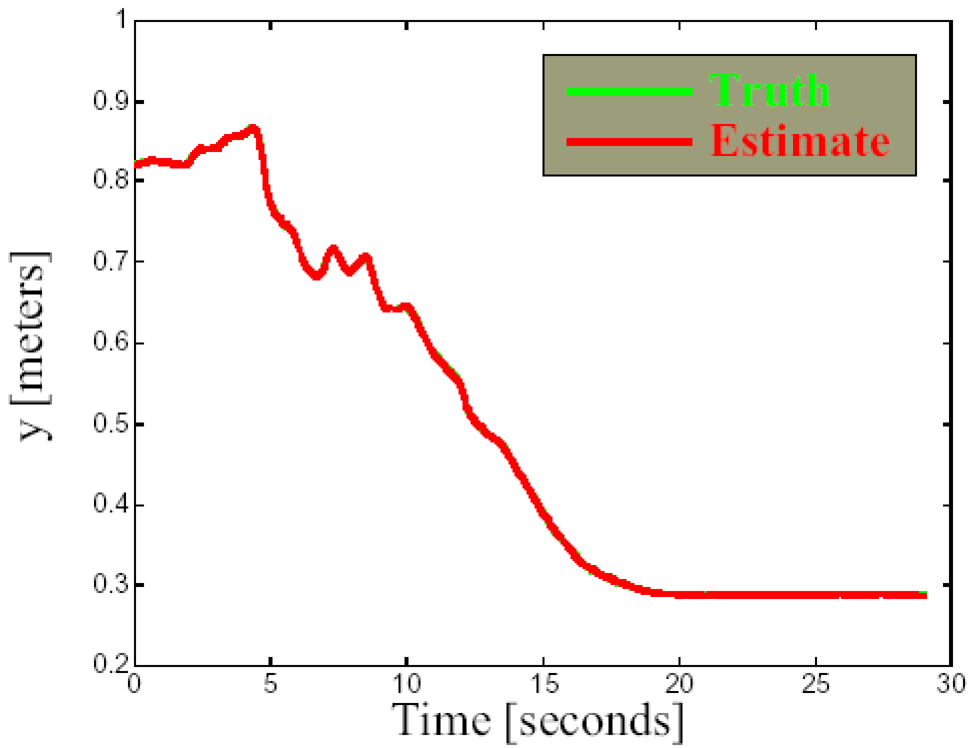
Test several models of assumed dynamics



[figure from Welsh and Bishop 2001]

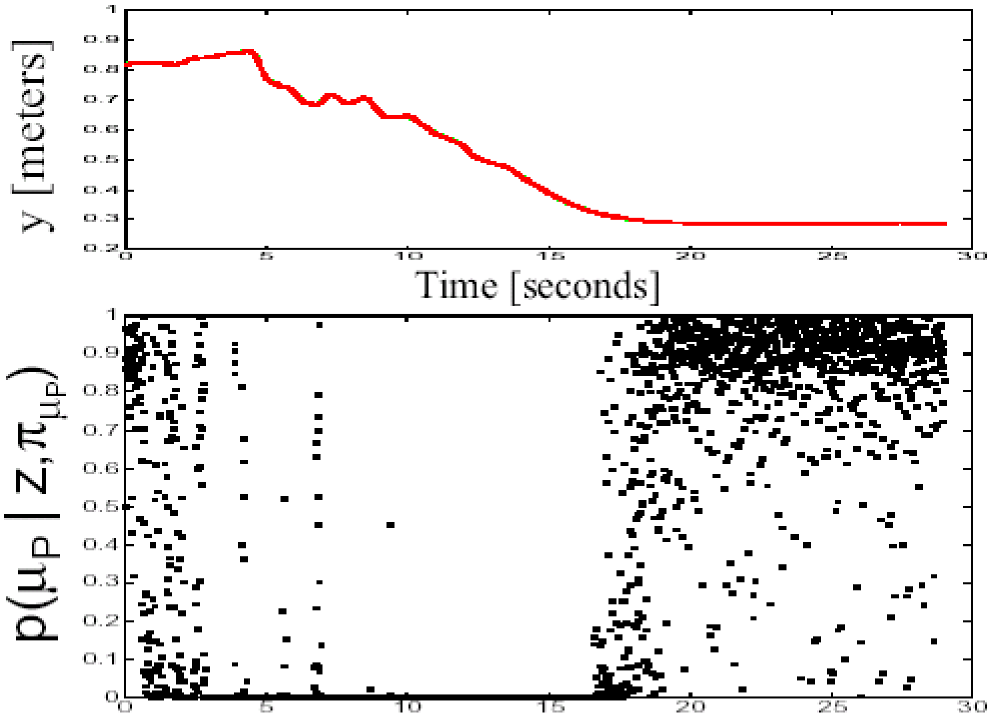
MM estimate

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



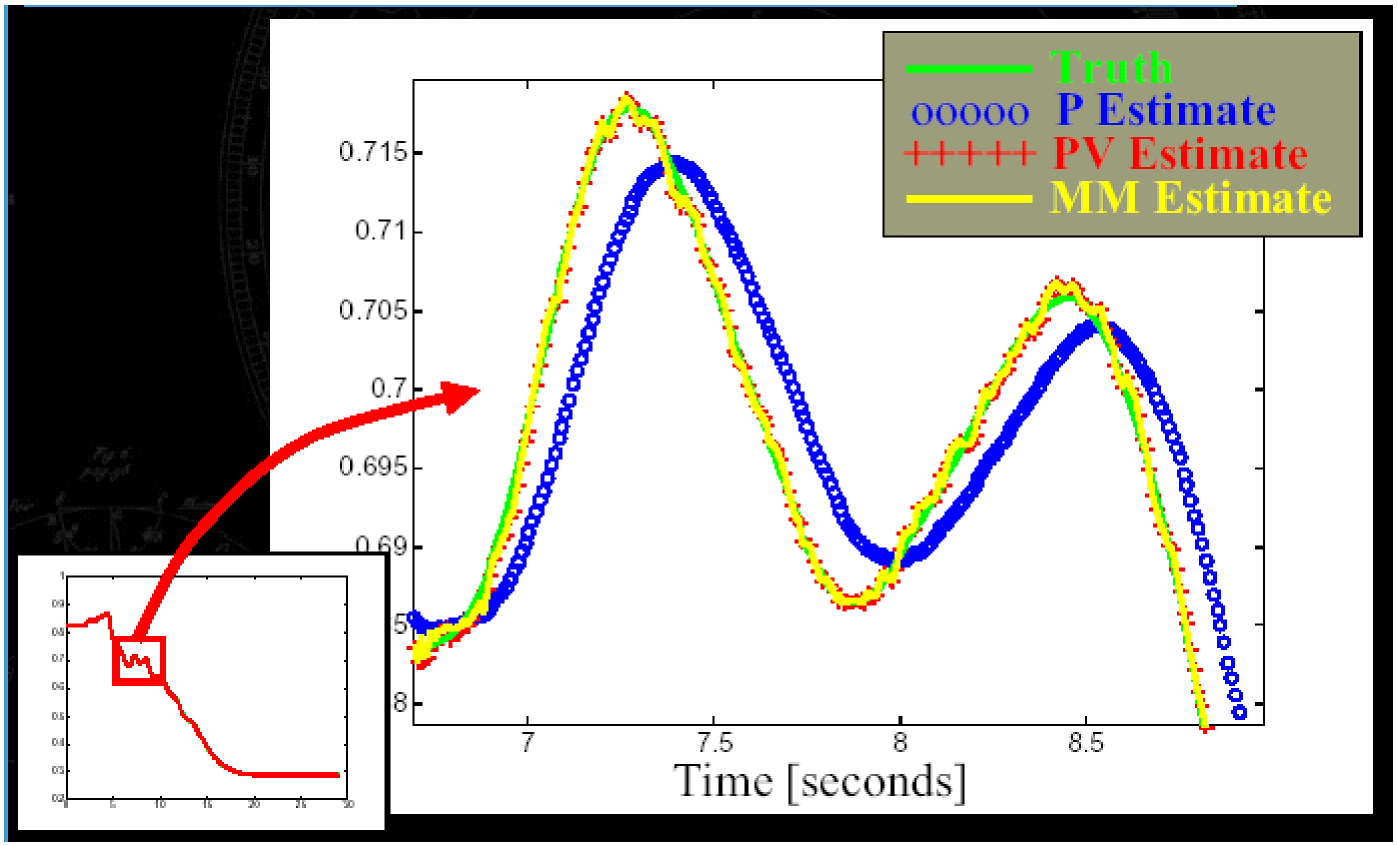
[figure from Welsh and Bishop 2001]

P likelihood



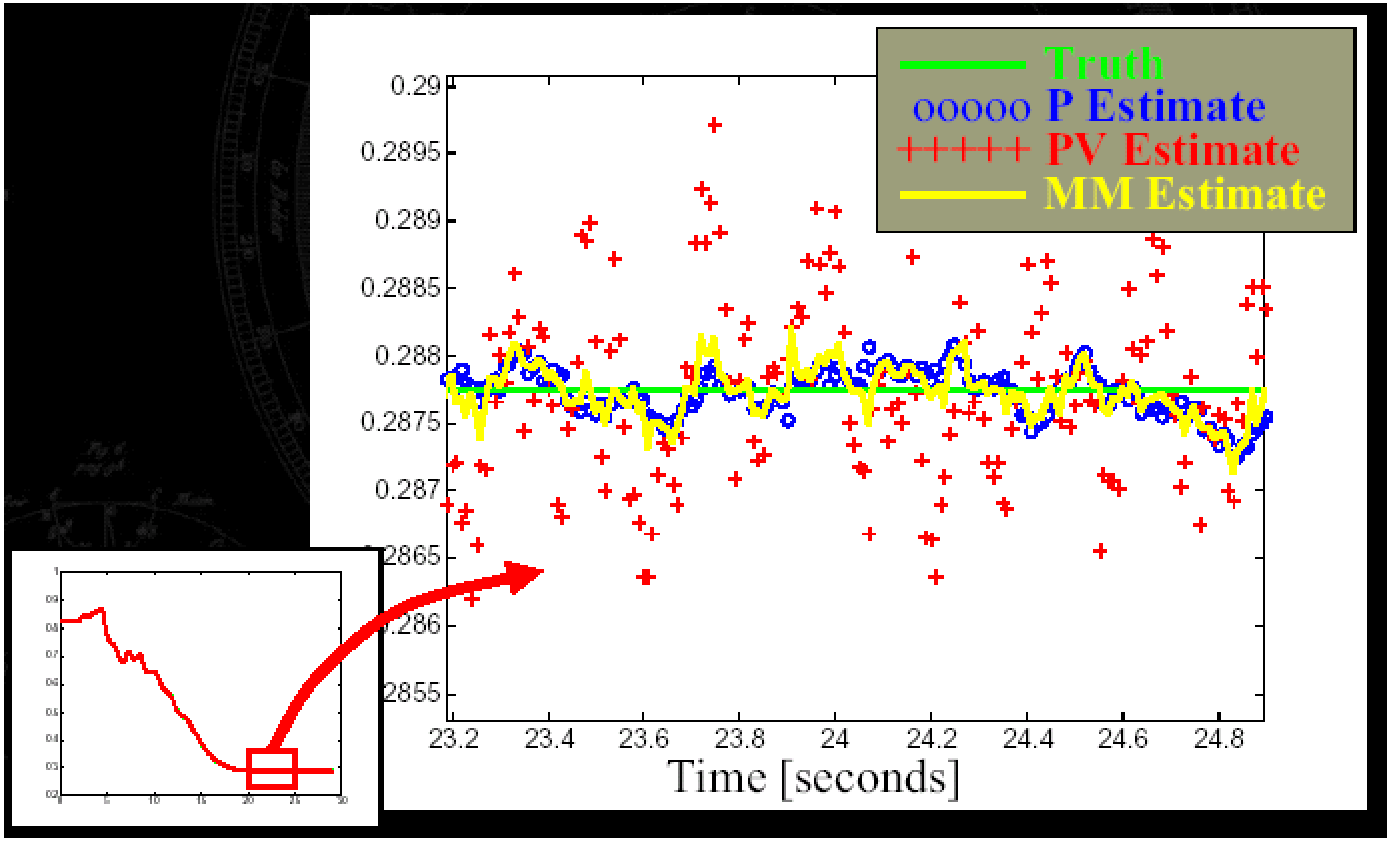
[figure from Welsh and Bishop 2001]

No lag



[figure from Welsh and Bishop 2001]

Smooth when still



[figure from Welsh and Bishop 2001]

Resources

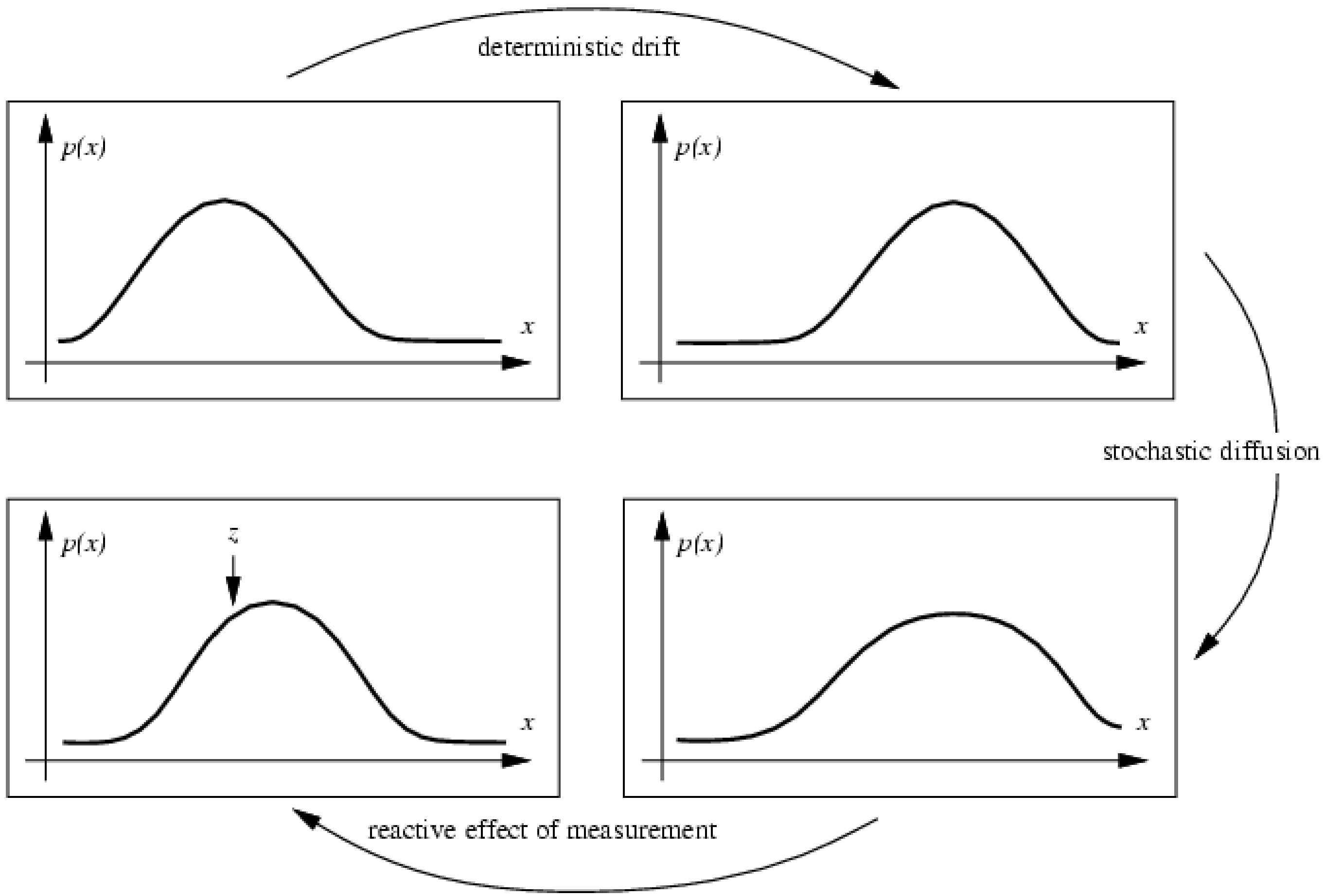
- Kalman filter homepage

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

- Kevin Murphy's Matlab toolbox:

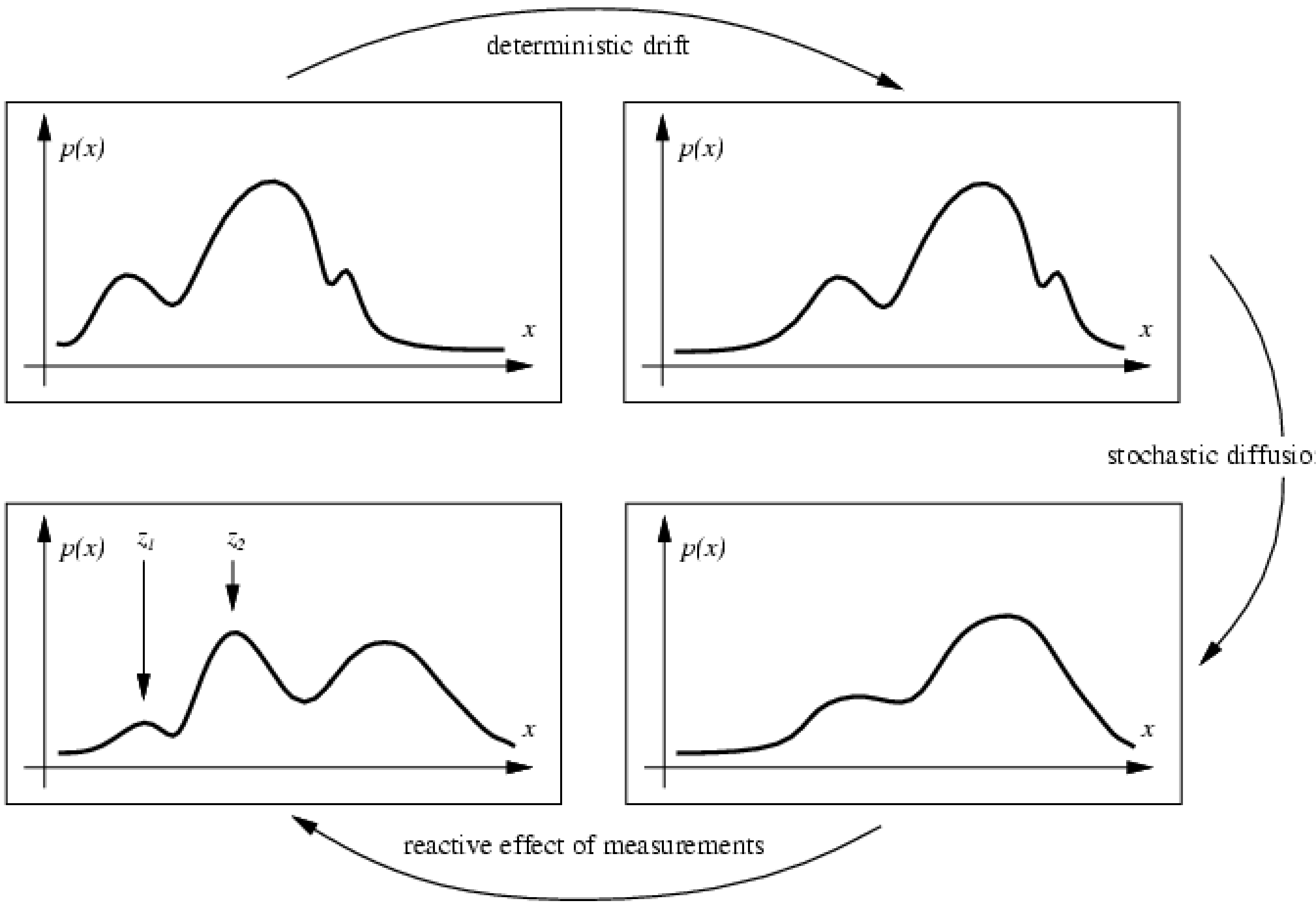
http://www.ai.mit.edu/~murphyk/Software/Kalman/kalman.html

(KF) Distribution propogation



[Isard 1998]

Distribution propogation



[Isard 1998]

EKF

Linearize system at each time point to form an Extended Kalman Filter (EKF)

– Compute Jacobian matrix

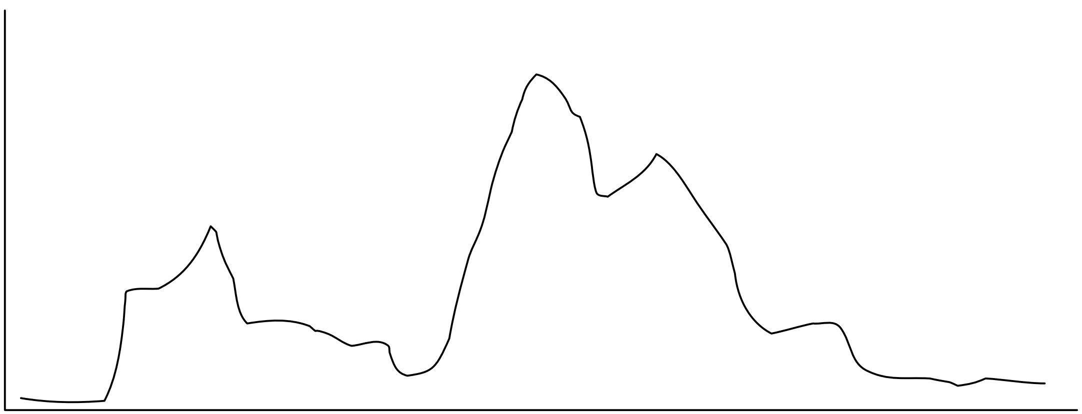
$\mathcal{J}(\boldsymbol{g}; \boldsymbol{x}_j)$

whose (l,m)'th value is $\dfrac{\partial f_l}{\partial x_m}$ evaluated at $\boldsymbol{x}_j$

– use this for forward model at each step in KF

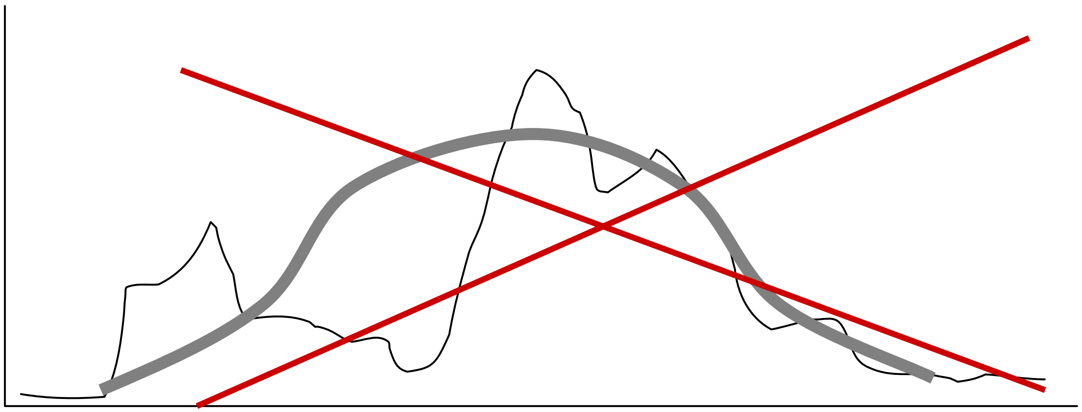
Useful in many engineering applications, but not as successful in computer vision….

Representing non-linear Distributions



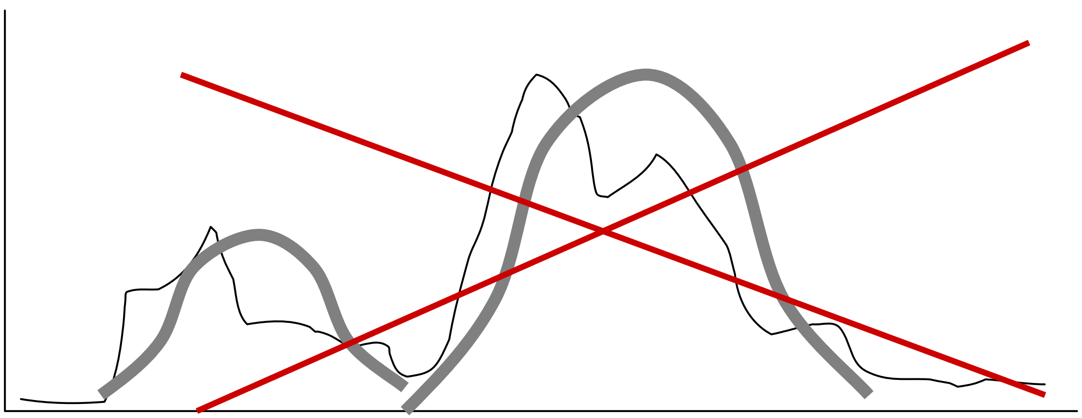
Representing non-linear Distributions

Unimodal parametric models fail to capture real-world densities…



Representing non-linear Distributions

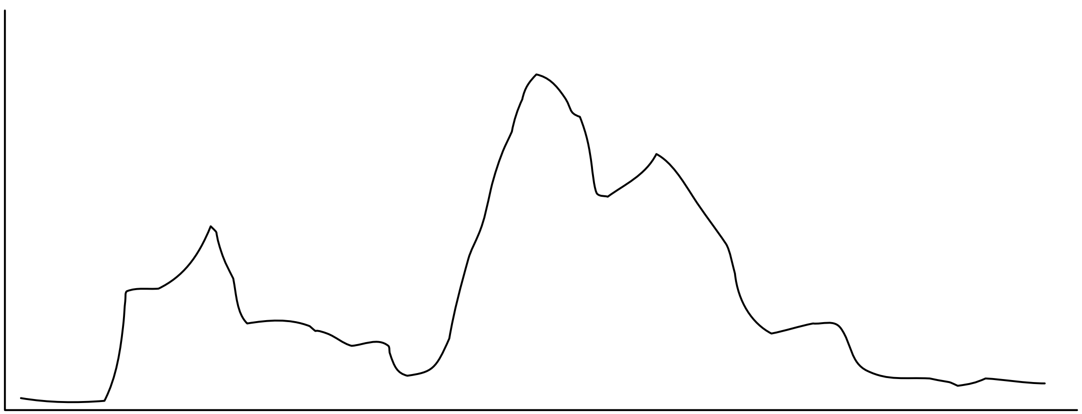
Mixture models are appealing, but very hard to propagate analytically!



[but see Cham and Rehg's MHT approach]

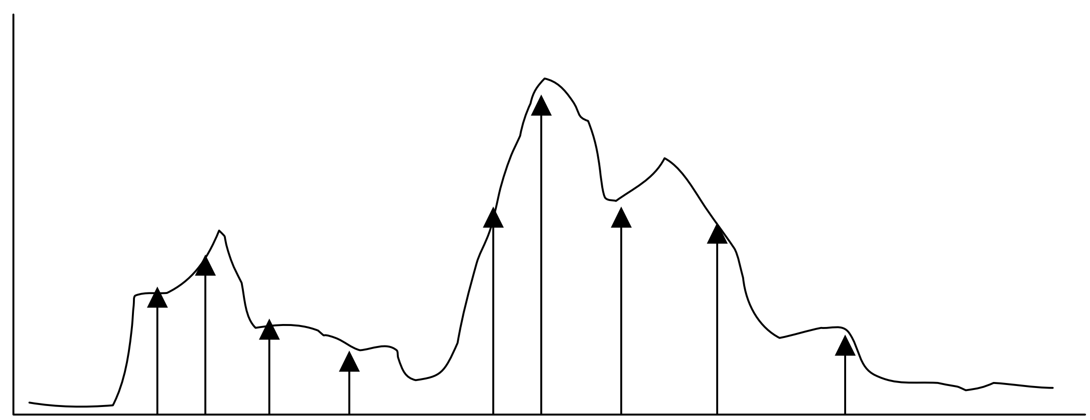
Representing Distributions using Weighted Samples

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:



Outline

- Recursive filters
- State abstraction
- Density propagation
- Linear Dynamic models / Kalman filter
- Data association
- Multiple models

- Next time:
  - Sampling densities
  - Particle filtering

[Figures from F&P except as noted]