Sensor Fusion

Fredrik Gustafsson

Lecture	Content	Chapters
1	Course overview. Estimation theory for linear models.	1–2
2	Estimation theory for nonlinear models and sensor networks.	3–4
3	Detection theory with sensor network applications.	5
4	Nonlinear filter theory. The Kalman filter. Filter banks.	6–7, 10
5	Kalman filter approximation for nonlinear models (EKF, UKF).	8
6	The point-mass filter and the particle filter.	9
7	The particle filter theory. The marginalized particle filter.	9
8	Simultaneous localization and mapping (SLAM).	11
9	Modeling and motion models.	12–13
10	Sensors and filter validation.	14–15

Literature: Statistical Sensor Fusion. Studentlitteratur, 2010.

Exercises: compendium. Software: *Signals and Systems Lab* for Matlab.

Lecture 1: Estimation theory in linear models

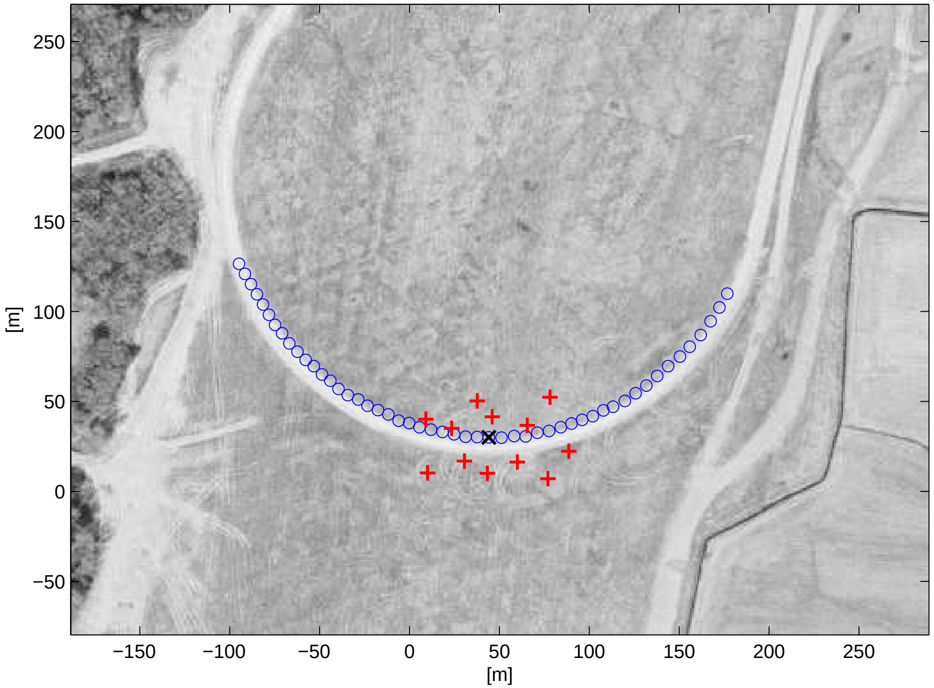
Whiteboard:

- The weighted least squares (WLS) method.
- The maximum likelihood (ML) method.
- The Cramér-Rao lower bound (CRLB).
- Fusion algorithms.

Slides:

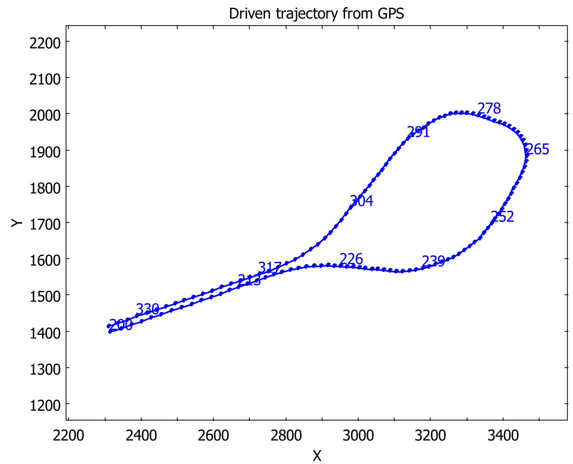
- Examples
- Code examples
- Algorithms

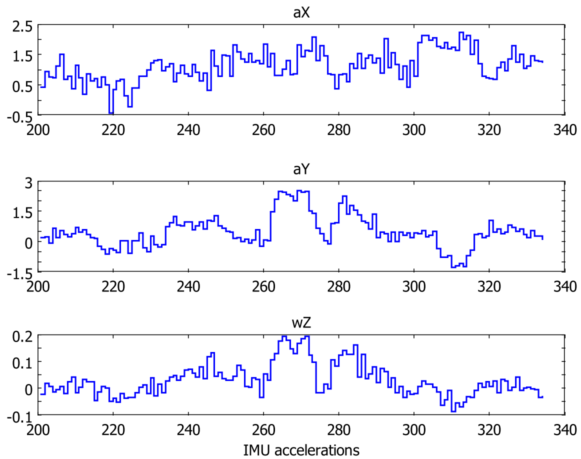
Example 1: sensor network



12 sensor nodes, each one with microphone, geophone and magnetometer.
One moving target.
Detect, localize and track/predict the target.

Example 2: fusion of GPS and IMU





GPS gives good position. IMU gives accurate accelerations.

Combine these to get even better position, velocity and acceleration.

Example 3: fusion of camera and radar images

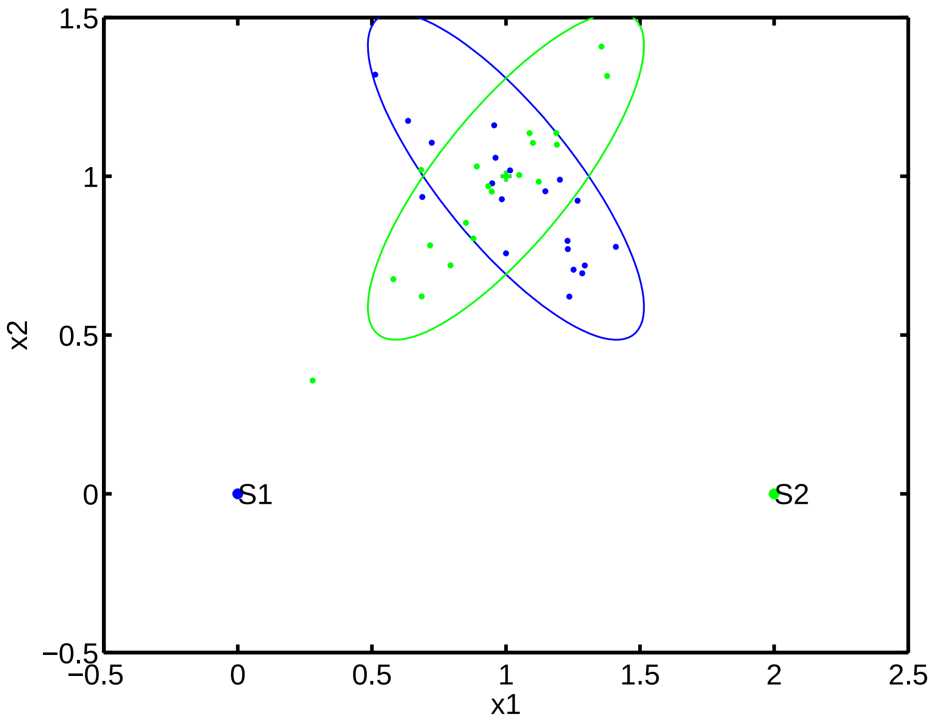
Radar gives range and range rate with good horizontal angle resolution, but no vertical resolution.
Camera gives very good angular resolution, and color, but no range.
Combined, they have a great potential for situation awareness.

Chapter 2: estimation for linear models

- Least squares and likelihood methods.
- Sensor network example.
- Fusion and safe fusion in distributed algorithms

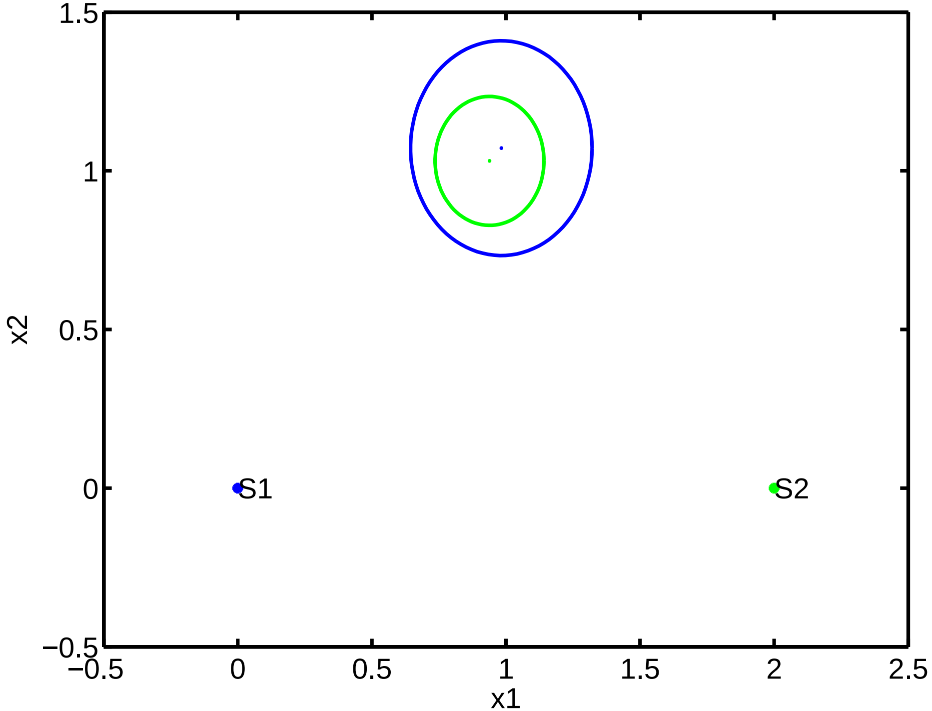
Code for Signals and Systems Lab:

```
p1=[0;0];
p2=[2;0];
x=[1;1];
X1=ndist(x,0.1*[1 -0.8;-0.8 1]);
X2=ndist(x,0.1*[1 0.8;0.8 1]);
plot2(X1,X2)
```



Sensor network example, cont'd

```
X3=fusion(X1,X2);   % WLS
X4=0.5*X1+0.5*X2;   % LS
plot2(X3,X4)
```



Information loops in sensor networks

- Information and sufficient statistics should be communicated in sensor networks.
- In sensor networks with untagged observations, our own observations may be included in the information we receive.
- Information loops (updating with the same sensor reading several times) give rise to optimistic covariances.
- Safe fusion algorithms (or *covariance intersection techniques*) give conservative covariances, using a worst case way of reasoning.

Safe fusion

Given two unbiased estimates $\hat{x}_1, \hat{x}_2$ with information $\mathcal{I}_1 = P_1^{-1}$ and $\mathcal{I}_2 = P_2^{-1}$ (pseudo-inverses if singular covariances), respectively. Compute the following:

1. SVD: $\mathcal{I}_1 = U_1 D_1 U_1^T$.
2. SVD: $D_1^{-1/2} U_1^T \mathcal{I}_2 U_1 D_1^{-1/2} = U_2 D_2 U_2^T$.
3. Transformation matrix: $T = U_2^T D_1^{1/2} U_1$.
4. State transformation: $\hat{\hat{x}}_1 = T\hat{x}_1$ and $\hat{\hat{x}}_2 = T\hat{x}_2$. The covariances of these are $\text{Cov}(\hat{\hat{x}}_1) = I$ and $\text{Cov}(\hat{\hat{x}}_2) = D_2^{-1}$, respectively.
5. For each component $i = 1, 2, \ldots, n_x$, let

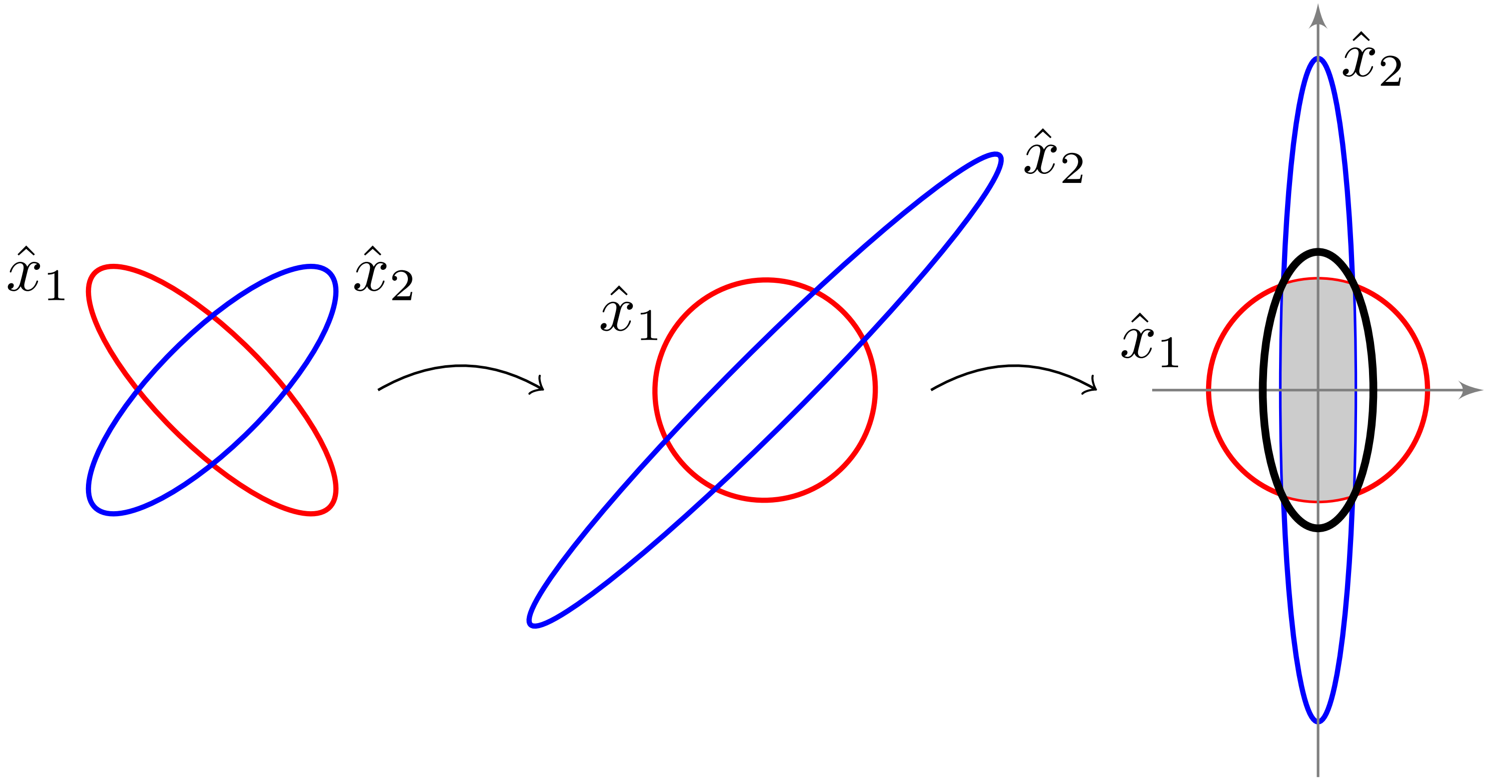
$$\hat{\hat{x}}^i = \hat{\hat{x}}_1^i, \quad D^{ii} = 1 \quad \text{if} \quad D_2^{ii} < 1,$$

$$\hat{\hat{x}}^i = \hat{\hat{x}}_2^i, \quad D^{ii} = D_2^{ii} \quad \text{if} \quad D_2^{ii} > 1.$$

6. Inverse state transformation:

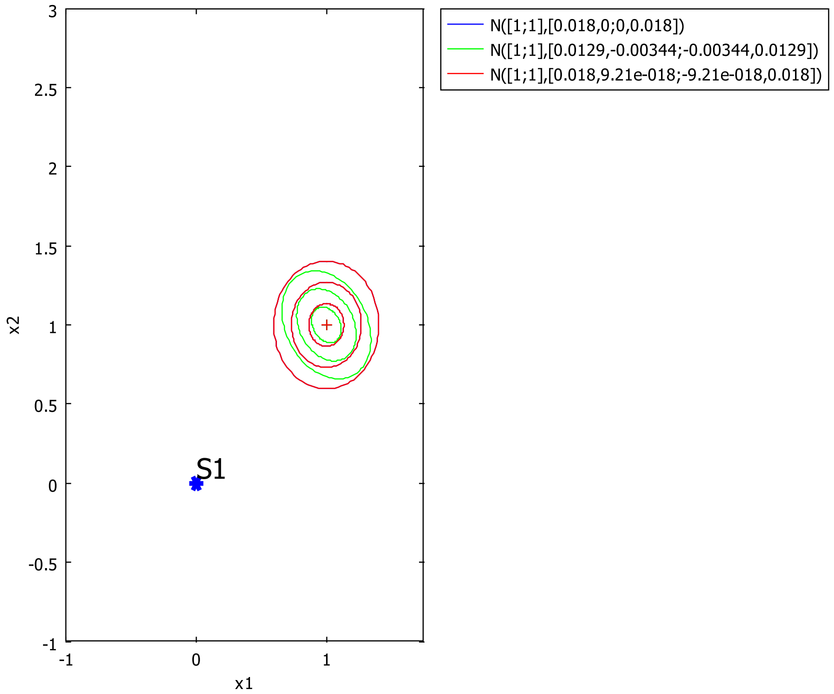
$$\hat{x} = T^{-1}\hat{\hat{x}}, \quad P = T^{-1} D^{-1} T^{-T}$$

Transformation steps



Sensor network example, cont'd

```
X3=fusion(X1,X2);      % WLS
X4=fusion(X1,X3);      % X1 used twice
X5=safefusion(X1,X3);
plot2(X3,X4,X5)
```



Sequential WLS

The WLS estimate can be computed recursively in the space/time sequence y_k. Suppose the estimate $\hat{x}_{k-1}$ with covariance P_k based on observations $y_{1:k-1}$. A new observation is fused using

$$\hat{x}_k = \hat{x}_{k-1} + P_{k-1}H_k^T \left(H_k P_{k-1} H_k^T + R_k\right)^{-1} (y_k - H_k \hat{x}_{k-1}),$$
$$P_k = P_{k-1} - P_{k-1}H_k^T \left(H_k P_{k-1} H_k^T + R_k\right)^{-1} H_k P_{k-1}.$$

Note that the fusion formula can be used alternatively. In fact, the derivation is based on the information fusion formula applying the matrix inversion lemma.

Batch vs sequential evaluation of loss function

The minimizing loss function can be computed in two ways using batch and sequential computations, respectively,

$$
\begin{aligned}
V^{WLS}(\hat{x}_N) &= \sum_{k=1}^{N} (y_k - H_k \hat{x}_N)^T R_k^{-1} (y_k - H_k \hat{x}_N) \\
&= \sum_{k=1}^{N} (y_k - H_k \hat{x}_{k-1})^T (H_k P_{k-1} H_k^T + R_k)^{-1} (y_k - H_k \hat{x}_{k-1}) \\
&- (\hat{x}_0 - \hat{x}_N)^T P_0^{-1} (\hat{x}_0 - \hat{x}_N)
\end{aligned}
$$

See Theorem 6.2!

The second expression should be used in decentralized sensor network implementations and on-line algorithms.

The last correction term to de-fuse the influence of the initial values is needed only when this initialization is used.

Batch vs sequential evaluation of likelihood

The Gaussian likelihood of data is important in model validation, change detection and diagnosis. Generally, Bayes formula gives

$$p(y_{1:N}) = p(y_1) \prod_{k=2}^{N} p(y_k | y_{1:k-1}).$$

For Gaussian noise and using the sequential algorithm, this is

$$p(y_{1:N}) = \prod_{k=1}^{N} \gamma(y_k; H_k \hat{x}_{k-1}, H_k P_{k-1} H_k^T + R_k)$$

Using (5.98) in *Adaptive Filtering and Change Detection*:

$$p(y_{1:N}) = \gamma(\hat{x}_N; x_0, P_0) \sqrt{\det(P_N)} \prod_{k=1}^{N} \gamma(y_k; H_k \hat{x}_N, R_k).$$

Prefered for de-centralized computation in sensor networks or on-line algorithms.