Sensor Data Fusion

Silvio Simani

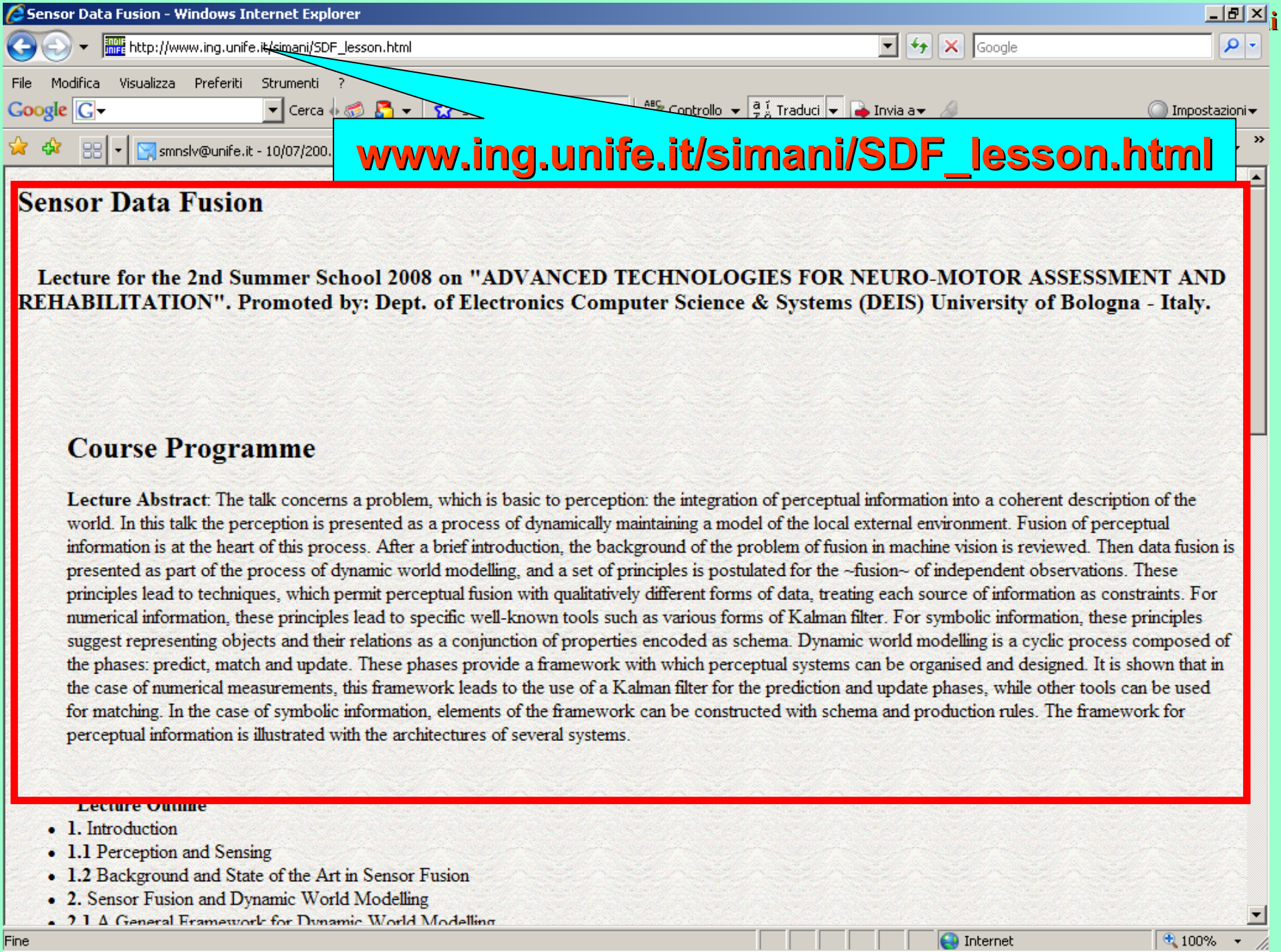
Dept. of Engineering, University of Ferrara

V. Saragat, 1/E. I-44100, Ferrara, Italy.

Ph.: +39 0532 97 4844, fax: +39 0532 97 4870

E-MAIL: silvio.simani@unife.it

HOME PAGE: http://www.ing.unife.it/simani/


Sensor Data Fusion - Windows Internet Explorer
http://www.ing.unife.it/simani/SDF_lesson.html
Google
File Modifica Visualizza Preferiti Strumenti ?
Cerca
Controllo
Traduci
Invia a
Impostazioni
www.ing.unife.it/simani/SDF_lesson.html
Sensor Data Fusion
Lecture for the 2nd Summer School 2008 on "ADVANCED TECHNOLOGIES FOR NEURO-MOTOR ASSESSMENT AND REHABILITATION". Promoted by: Dept. of Electronics Computer Science & Systems (DEIS) University of Bologna - Italy.
Course Programme
Lecture Abstract: The talk concerns a problem, which is basic to perception: the integration of perceptual information into a coherent description of the world. In this talk the perception is presented as a process of dynamically maintaining a model of the local external environment. Fusion of perceptual information is at the heart of this process. After a brief introduction, the background of the problem of fusion in machine vision is reviewed. Then data fusion is presented as part of the process of dynamic world modelling, and a set of principles is postulated for the ~fusion~ of independent observations. These principles lead to techniques, which permit perceptual fusion with qualitatively different forms of data, treating each source of information as constraints. For numerical information, these principles lead to specific well-known tools such as various forms of Kalman filter. For symbolic information, these principles suggest representing objects and their relations as a conjunction of properties encoded as schema. Dynamic world modelling is a cyclic process composed of the phases: predict, match and update. These phases provide a framework with which perceptual systems can be organised and designed. It is shown that in the case of numerical measurements, this framework leads to the use of a Kalman filter for the prediction and update phases, while other tools can be used for matching. In the case of symbolic information, elements of the framework can be constructed with schema and production rules. The framework for perceptual information is illustrated with the architectures of several systems.
Lecture Outline
1. Introduction
1.1 Perception and Sensing
1.2 Background and State of the Art in Sensor Fusion
2. Sensor Fusion and Dynamic World Modelling
Fine
Internet
100%


Sensor Data Fusion - Windows Internet Explorer
http://www.ing.unife.it/docenti/SilvioSimani/www/SDF_lesson.html
www.ing.unife.it/simani/SDF_lesson.html
Lecture Outline
1. Introduction
1.1 Perception and Sensing
1.2 Background and State of the Art in Sensor Fusion
2. Sensor Fusion and Dynamic World Modelling
2.1 A General Framework for Dynamic World Modelling
2.2 Principles for Integrating Perceptual Information
3. Techniques for Fusion of Numerical Properties
3.1 State Representation
3.2 Prediction: Discrete State Transition Equations
3.3 Matching Observation to Prediction
3.4 The Kalman Filter Update Equations
4. Example Systems and Practical Applications
4.1 Dynamic World Modelling
4.2 An Integrated Supervision System
5. Conclusions
Bibliographic References
K. Brammer, G. Siffling, Kalman Bucy Filters, Artech House Inc., Norwood MA, USA, 1989.
J. Jazwinski, Stochastic Processes and Filtering Theory. Academic Press, New York, 1970.
R. Kalman, A new approach to Linear Filtering and Prediction Problems, Transactions of the ASME, Series D. J. Basic Eng., Vol 82, 1960.
R. Kalman, R. Bucy, New Results in Linear Filtering and Prediction Theory, Transaction of the ASME, Series D. J. Basic Eng., Vol 83, 1961.
A. Papoulis, Probability, random variables, and stochastic processes. New York: McGraw-Hill, 1991.
D. L. Hall, Mathematical techniques in multisensor data fusion. Boston: Artech House, 1992.
Brian D. O. Anderson, John B. Moore, Optimal Filtering. Information and System Science Series. Thomas Kailath Editor, 2005.


Sensor Data Fusion - Windows Internet Explorer
http://www.ing.unife.it/docenti/SilvioSimani/www/SDF_lesson.html
www.ing.unife.it/simani/SDF_lesson.html
Download
"Lecture Slides" (in English): (PDF file, KB). (PDF file, KB, 2 slides per page). (PDF file, KB, 4 slides per page).
Movies used for the Presentation: (WMF and MPG files in zipped format, 24MB).
Matlab and Simulink files and software used for the presentation: (Matlab and Simulink files in zipped format, 185KB).
James L. Crowley and Yves Demazeau, "Principles and Techniques for Sensor Data Fusion". LIFIA (IMAG) 46 avenue Felix Viallet. F-38031 Grenoble Cédex, FRANCE. (PDF file, 90KB).
RON ROTH, "Trends in Sensor and Data Fusion". 'Photogrammetric Week 05', Dieter Fritsch, Ed. Wichmann Verlag, Heidelberg 2005. (PDF file, 3.74MB).
G. GIRIJA, J. R. RAOL, R. APPAVU RAJ and SUDESH KASHYAP, "Tracking filter and multi-sensor data fusion". Sadhana, Vol. 25, Part 2, April 2000, pp. 159-167. Printed in India, 2000. (PDF file, 117KB).
SHRABANI BHATTACHARYA and R APPAVU RAJ, "Performance evaluation of multi-sensor data fusion technique for test range application", Sadhana Vol. 29, Part 2, April 2004, pp. 237-247. Printed in India, 2004. (PDF file, 257KB).
Introduction paper: "Kalman Filtering: Whence, What and Whither?" by B. D. O. Anderson and J. B. Moore. (PDF file).
The book by BRIAN D. O. ANDERSON and JOHN B. MOORE, "Optimal Filtering". INFORMATION AND SYSTEM SCIENCES SERIES. Thomas Kailath Editor. (PDF file).
R. E. KALMAN, Research Institute for Advanced Study, Baltimore, Md. "A New Approach to Linear Filtering and Prediction Problems" (1960). Transactions of the ASME Journal of Basic Engineering, 82 (Series D): 35-45. (PDF file).
"An Introduction to the Kalman Filter", by Greg Welch and Gary Bishop. Department of Computer Science University of North Carolina at Chapel Hill Chapel Hill, NC 27599-3175. April 5, 2004 (PDF file).
Pattern Recognition and Machine Vision: Kalman Filter. By Dr. Simon J.D. Prince Computer Science University College London. Gower Street, London, WC1E 6BT: (PDF file).
"Least-Squares Estimation: From Gauss to Kalman". By H.W. Sorenson. University of California. San Diego. IEEE Spectrum, vol. 7, pp. 63-68. July, 1970. (PDF file).
Kalman filter Main Home Page: . Some tutorials, references, and research on the Kalman filter. The site is maintained by Greg Welch and Gary Bishop, faculty members of the Department of Computer Science at the University of North Carolina at Chapel Hill.
Internet
100%

Lecture Topics

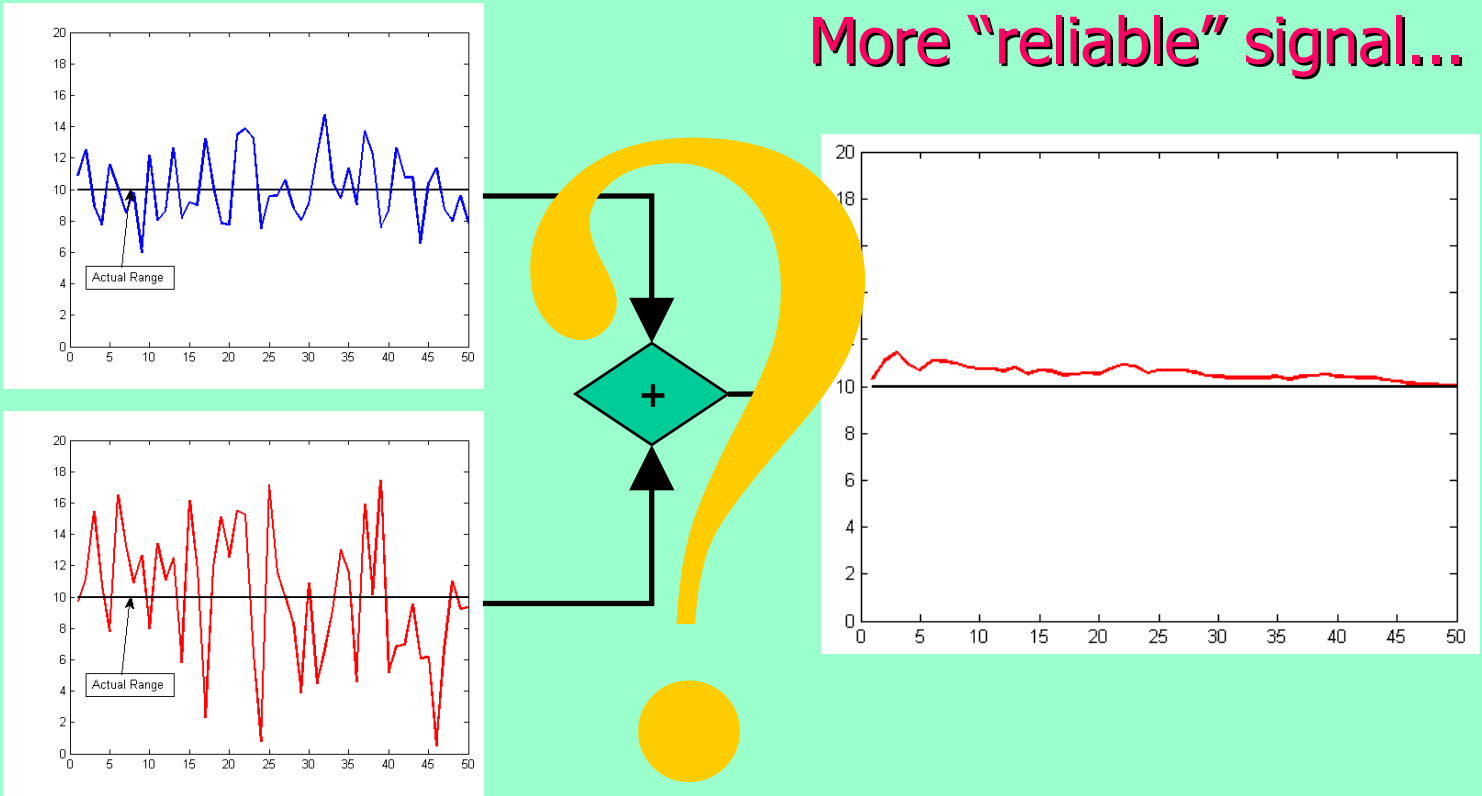
- Introduction
- Sensor Fusion and Dynamic World Modelling
- Techniques for Fusion of Numerical Properties
  - The Kalman Filter
- Example Systems and Practical Applications
- Conclusions

Bibliographical References

- K. Brammer, G. Siffling, *Kalman Bucy Filters*, Artech House Inc., Norwood MA, USA, 1989.
- J. Jazwinski, *Stochastic Processes and Filtering Theory*. Academic Press, New York, 1970.
- R. Kalman, A new approach to Linear Filtering and Prediction Problems, *Transactions of the ASME*, Series D. J. Basic Eng., Vol 82, 1960.
- R. Kalman, R. Bucy, New Results in Linear Filtering and Prediction Theory, *Transaction of the ASME*, Series D. J. Basic Eng., Vol 83, 1961.
- A. Papoulis, ***Probability, random variables, and stochastic processes***. New York: McGraw-Hill, 1991.
- D. L. Hall, *Mathematical techniques in multisensor data fusion*. Boston: Artech House, 1992.
- Brian D. O. Anderson, John B. Moore, *Optimal Filtering*. Information and System Science Series. Thomas Kailath Editor, 2005.

Sensor Fusion: Example…

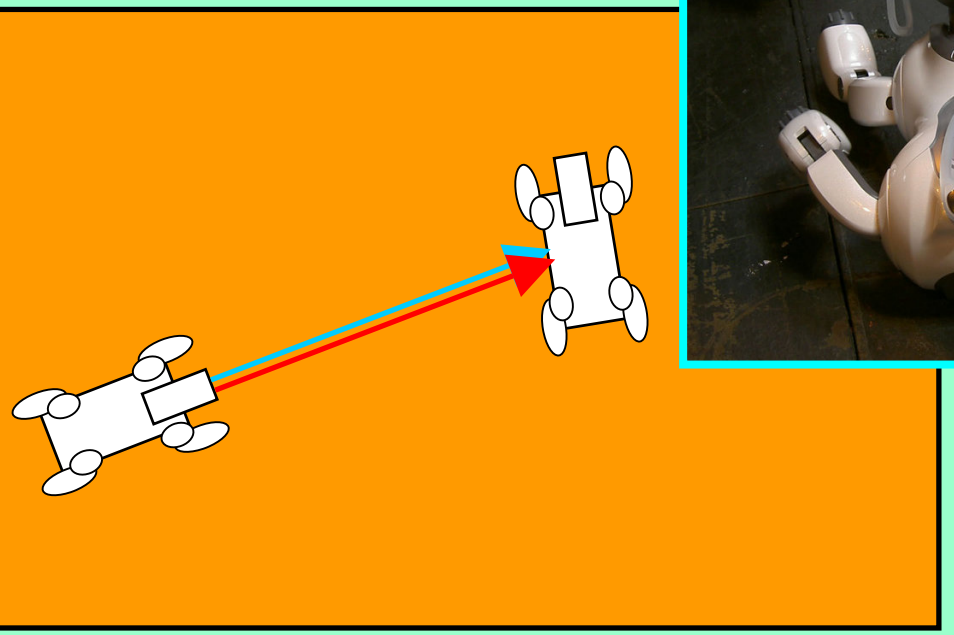
Measures from sensors…



More "reliable" signal…

Motivation: Example

- Let's assume that we have an AIBO. It detects an opponent at a distance of 100 cm with a sensor on its nose, but the same opponent reads at a distance of 80 cm from its chest sensor.
- What is our estimate of the distance?

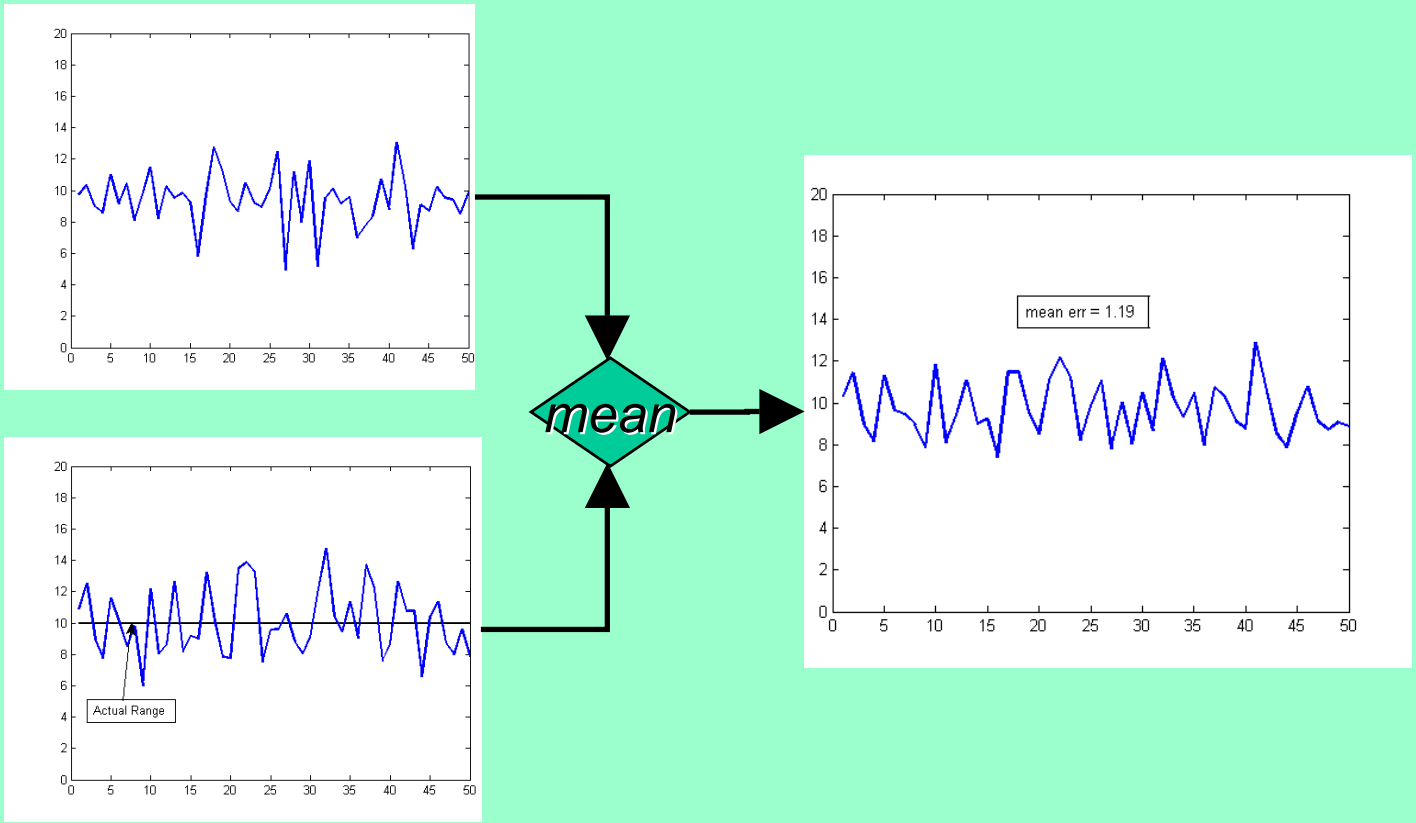


Why aren't the Sensor Measurements Correct?

- What about the accuracy of the (IR) sensors on the AIBO?
  - How was the ambient light?
  - What color was the target?
  - Are the errors the same for all ranges?
  - What about the orientation of the target surface?
  - Is it sensor noise?
  - What about bias?
- There are many reasons…
- These are often lumped (incorrectly) into the term "sensor noise"

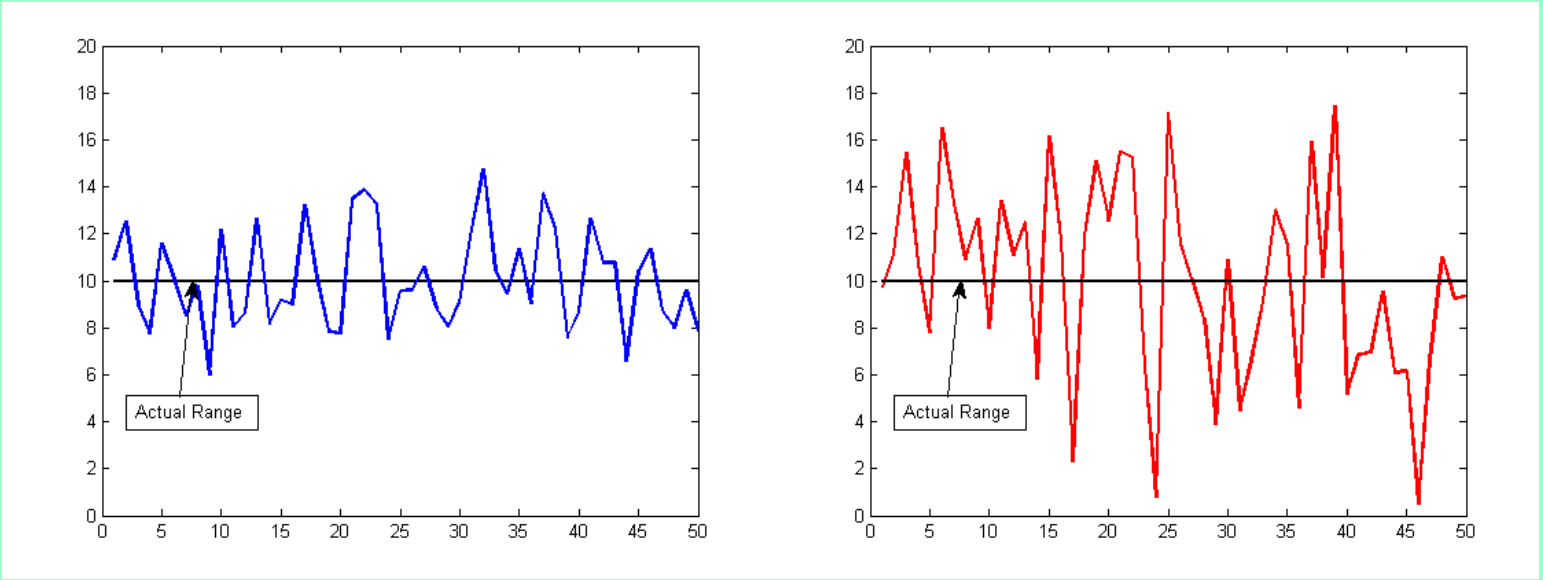
How should we Merge the Measurements?

- Can't we just average them?



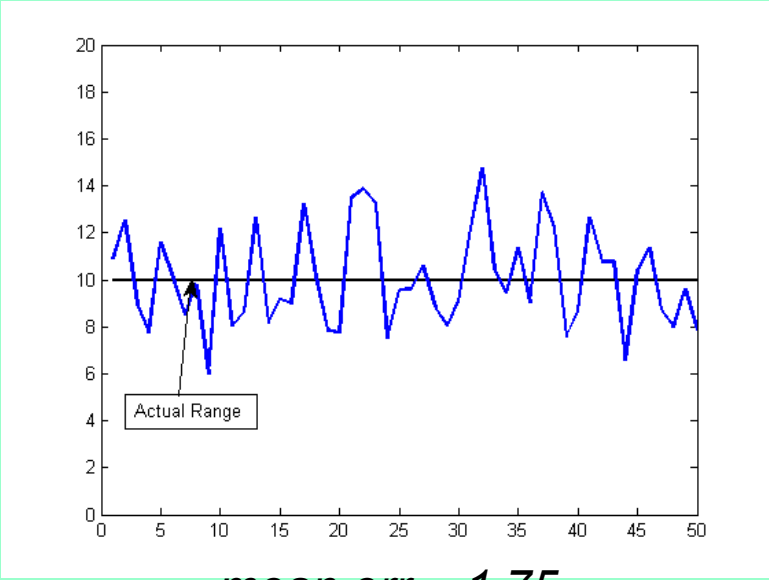
Merging Data from Heterogeneous Sensors

- Let's say that instead you have 2 different (not very accurate) sensor measuring the distance to a target over time
- How could you combine the data to get the best estimate possible for the true range to target?

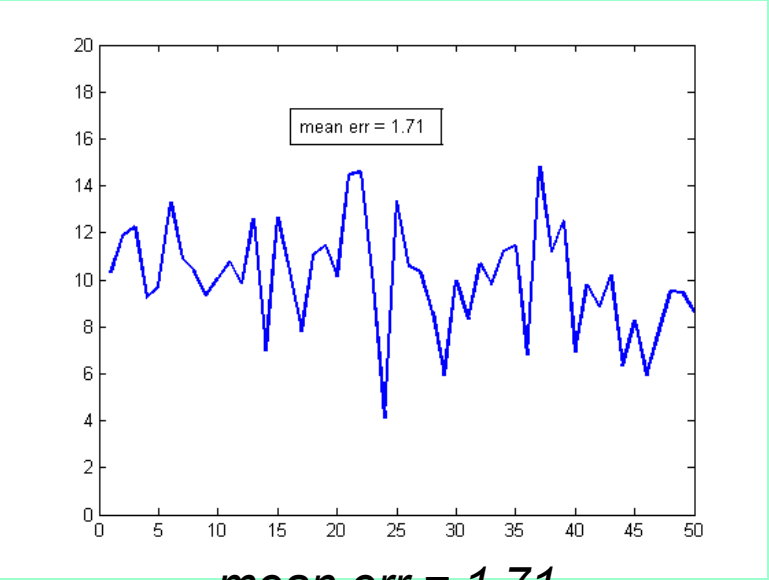


Merging Data from Heterogeneous Sensors (cont'd)

- Why not just average them?
- That works OK, but not really well...
- Can we do more?



mean err = 1.75



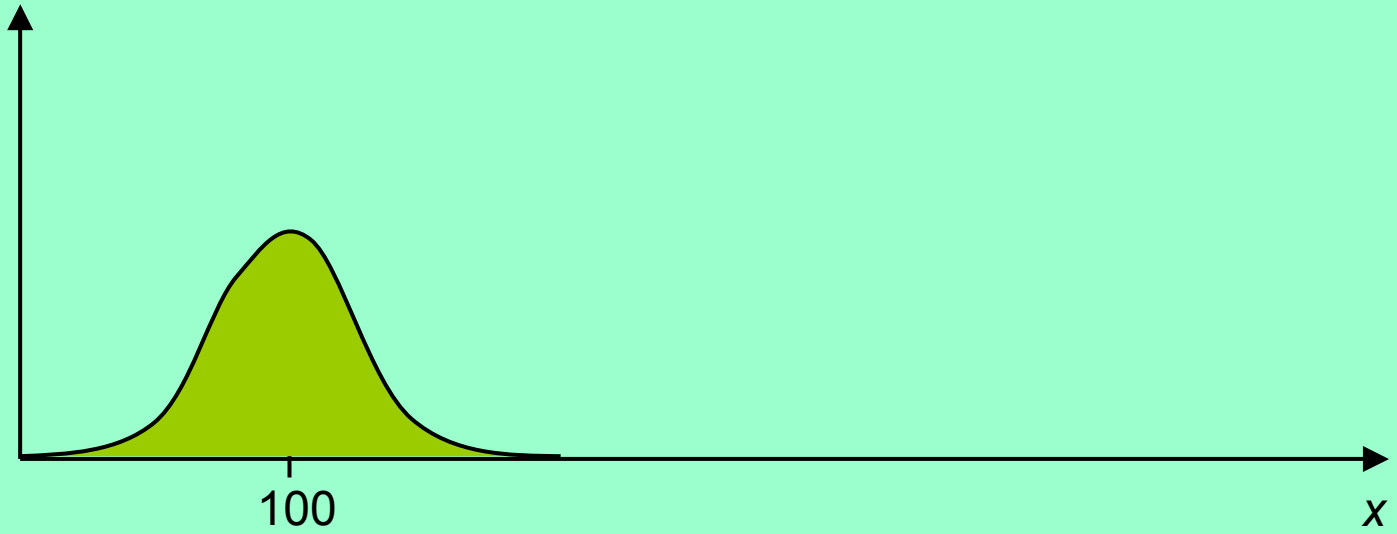
mean err = 1.71

Fusing Sensor Data

- Our ability to do something "intelligent" with the sensor data is a function of what we know about the *sensors* themselves
- The more accurate sensor model we have, the better our estimation performance will be
- These models are typically generated in a time consuming, empirical fashion
- Sensor models can vary dramatically as a function of the operational environment (recall the AIBO example)
- Often sensor models are represented compactly in the terms of probability density functions/distributions

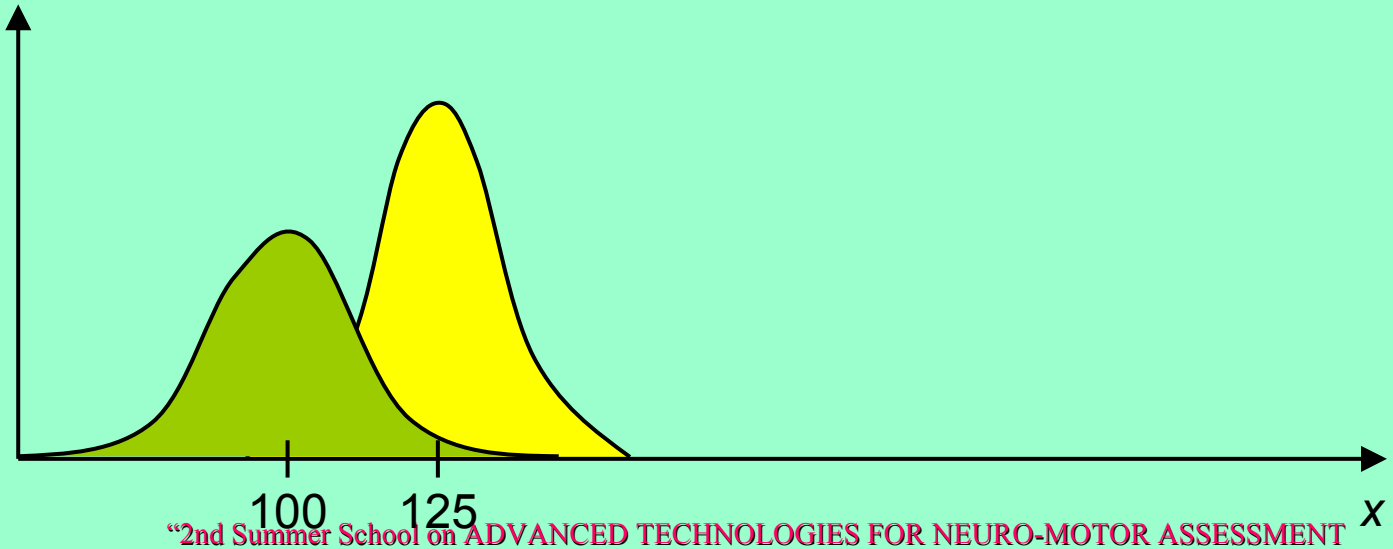
A Simple Sensor Fusion Example with a Gaussian Noise Model

- Consider a ship sailing east with a perfect compass trying to estimate its position.
- You estimate the position *x* from the stars as $z_1=100$ with a precision of $\sigma_x=4$ miles



A Simple Example (cont'd)

- Along comes a more experienced navigator, and she takes her own sighting z_2
- She estimates the position $x = z_2 = 125$ with a precision of $\sigma_x = 3$ miles
- How do you merge her estimate with your own?



A Simple Example (cont'd)

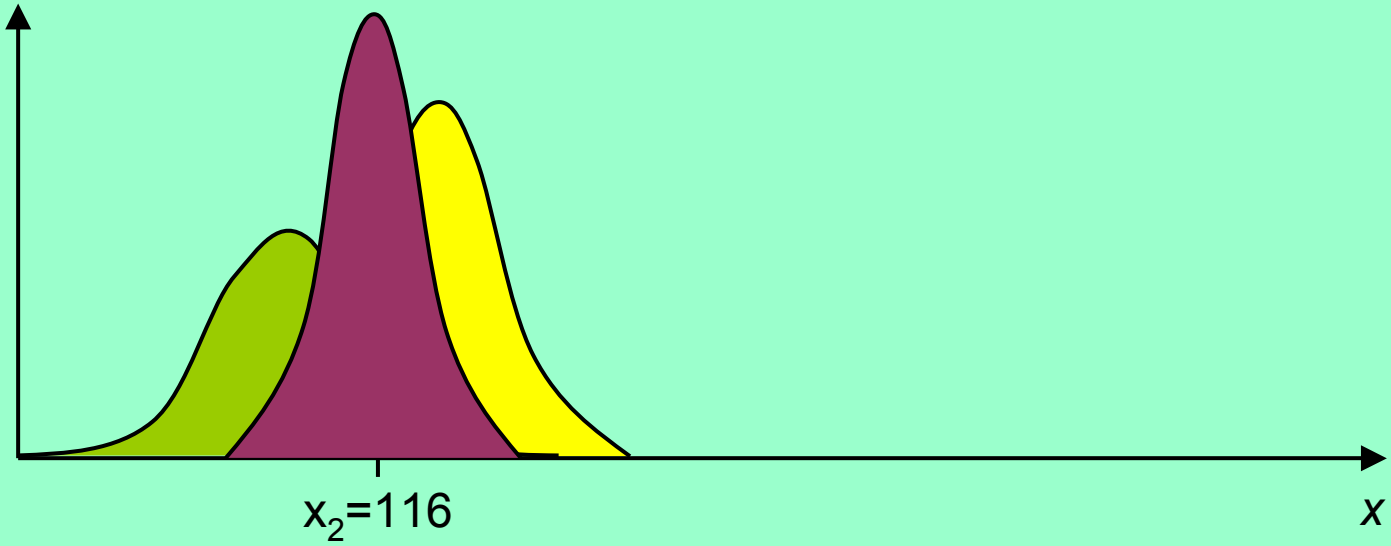
$$\mu = \left[\frac{\sigma_{z_2}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_2$$

$$= \left[\frac{9}{16+9}\right] 100 + \left[\frac{16}{16+9}\right] 125 = 116$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2}$$

$$\frac{1}{\sigma^2} = \frac{1}{9} + \frac{1}{16} = \frac{25}{144}$$

$$\Rightarrow \sigma = 2.4$$



A Simple Example (cont'd)

- With the distributions being Gaussian, the best estimate for the state is the mean of the distribution, so...

$$x_2 = \left[\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_2$$

or alternately

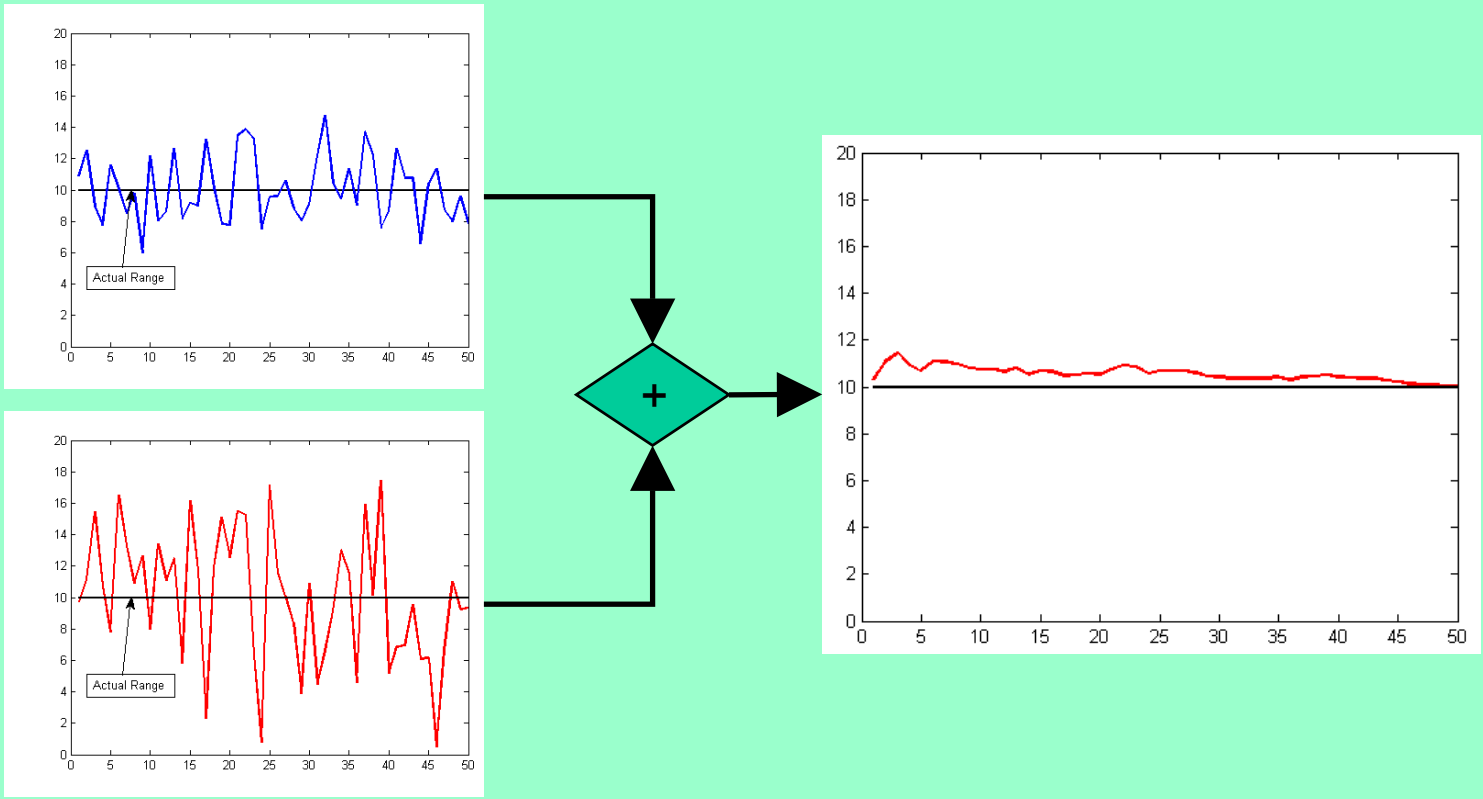
$$= z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right](z_2 - z_1)$$

$$= z_1 + K_2(z_2 - z_1)$$

Correction Term

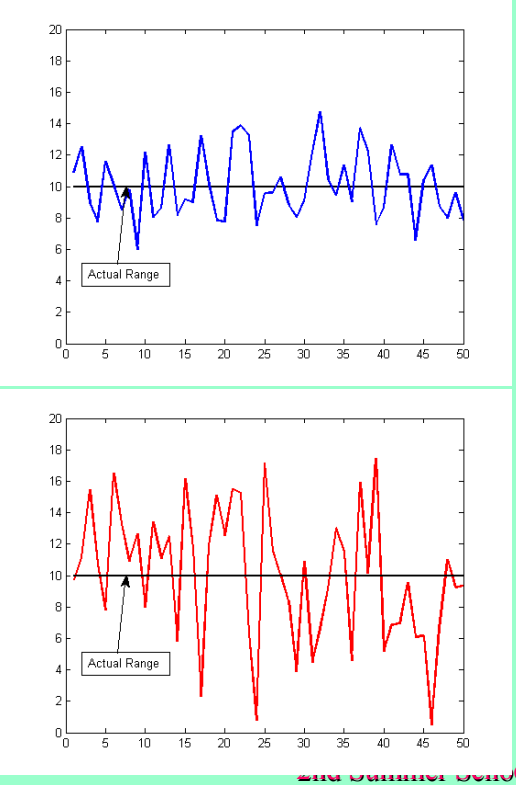
Let's Apply this to Our Example

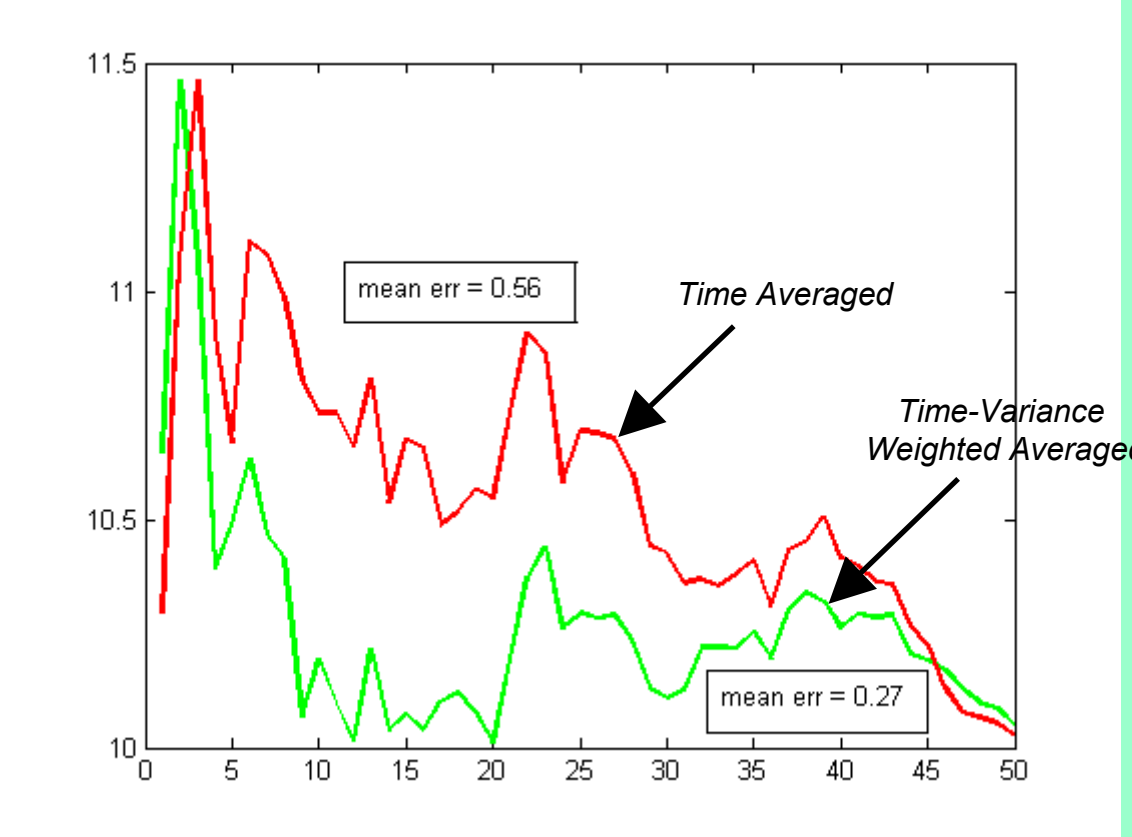
- Let's assume that we knew that the standard deviation of our sensors was 2 meters (blue) and 4 meters (red)



What if we had Merely Averaged the Data Over Time Instead

- The time average is in effect assuming that the variance of the two sensors are the same
- If we know better, we can use this to obtain a more accurate estimate





Sensor Fusion Principle

- ✓ Sensor fusion refers to the techniques applied in (intelligently) merging multiple sensor measurements across different sensors and/or time in order to obtain a more accurate estimate of some parameter of interest (e.g. the position of a target)
- ✓ The quality of your fused estimate will be a direct function of the accuracy of your sensor model
- ✓ In practice, the best models are often obtained empirically
- ✓ These empirical data are often fit to a Gaussian model when possible for mathematical convenience
- ✓ The primary motivation for this is the Kalman Filter algorithms that rely upon the Gaussian assumption, and work extremely well in theory and practice

Overview
The (Discrete) Kalman Filter

What is a Kalman Filter?

- *Optimal recursive* data fusion algorithm
- Predictor-Corrector style algorithm
- Processes all available sensor measurements in estimating the value of parameters of interest using:
  - Knowledge of system and sensor dynamics
  - Statistical models reflecting uncertainty in system noise and sensor dynamics
  - Any information regarding initial conditions

What is a Kalman Filter (cont'd)?

- *Optimal* in the sense that for systems which can be described by a *linear model* – *e.g.*

$$x_{k+1} = Ax_k + Bu_k + w_k$$
$$z_k = Hx_k + v_k$$

and for which the process and measurement noises w_k and v_k are *normally distributed*, the Kalman filter is the provably optimal estimator (estimate has minimum error variance)

- In our case, "process noise" corresponds to uncertainty in the motion model, measurement noise is from uncertainty in the sensing model, ***x*** denotes the state being estimated and ***z*** the sensor measurements

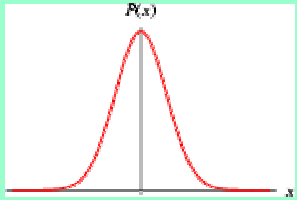
What is a Kalman Filter (cont'd)?

- *Recursive* in the sense that it is "memory-less"
  - Does not require all previous data to be maintained in memory and reprocessed at each time step
  - Propagates first and second order statistics only (*i.e.* mean and variance/covariance)

The Gaussian Distribution

- A 1-D Gaussian distribution is defined as

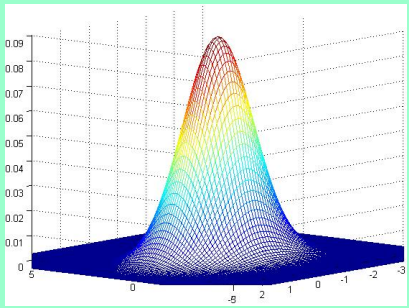
$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$



Gaussian p(x)

- In 2-D (assuming uncorrelated variables) this becomes

$$p(\vec{x}) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\left[\frac{(x_1-\mu_1)^2}{2\sigma_1^2}+\frac{(x_2-\mu_2)^2}{2\sigma_2^2}\right]}$$



- In *n* dimensions, it generalizes to

$$p(\vec{x}) = \frac{1}{\sqrt{(2\pi)^n |C|}} e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)}$$

In the Kalman filter, the process and measurement noise MUST be normally distributed for optimality to hold!

A Quick Primer/Refresher

- The *expected value* for a random variable *X* is (*i.e.* the mean) defined as

$$\mu = E(X) = \sum_{i=1}^{n} p_i x_i \quad \text{for discrete } X$$

$$\mu = E(X) = \int_{-\infty}^{\infty} x p_X(x) dx \quad \text{for continuous } X$$

- The *variance* of *X* about the mean is defined as

$$\sigma^2 = E[(x-\mu)^2] = \sum_{i=1}^{n} p_i (x_i - \mu)^2 \quad \text{for discrete } X$$

$$\sigma^2 = E[(x-\mu)^2] = \int_{-\infty}^{\infty} (x-\mu)^2 p_X(x) dx \quad \text{for continuous } X$$

A Quick Primer/Refresher (cont'd)

- When *X* is a vector, the variance is expressed in terms of a *covariance matrix C*, where its entries c_{ij} are:

$$c_{ij} = E[(\vec{x}_i - \vec{\mu}_i)^T (\vec{x}_j - \vec{\mu}_j)]$$

- The resulting matrix has the form:

$$C = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \cdots & \rho_{1n}\sigma_1\sigma_n \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \cdots & \rho_{2n}\sigma_2\sigma_n \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{1n}\sigma_1\sigma_n & \rho_{2n}\sigma_2\sigma_n & \cdots & \sigma_n^2 \end{bmatrix}$$

where ρ_{ij} corresponds to the degree of correlation between variables X_i and X_j

The Correlation Coefficient

- Correlation is a means to estimate how two functions/series are correlated. For a discrete series, it is defined as

$$\rho = \frac{\sum_i \left[(x_i - \mu_x)(y_i - \mu_y)\right]}{\sqrt{\sum_i (x_i - \mu_x)^2}\sqrt{\sum_i (y_i - \mu_y)^2}} \equiv \frac{C_{xy}}{\sqrt{C_{xx}}\sqrt{C_{yy}}} \equiv \frac{C_{xy}}{\sigma_x \sigma_y}$$

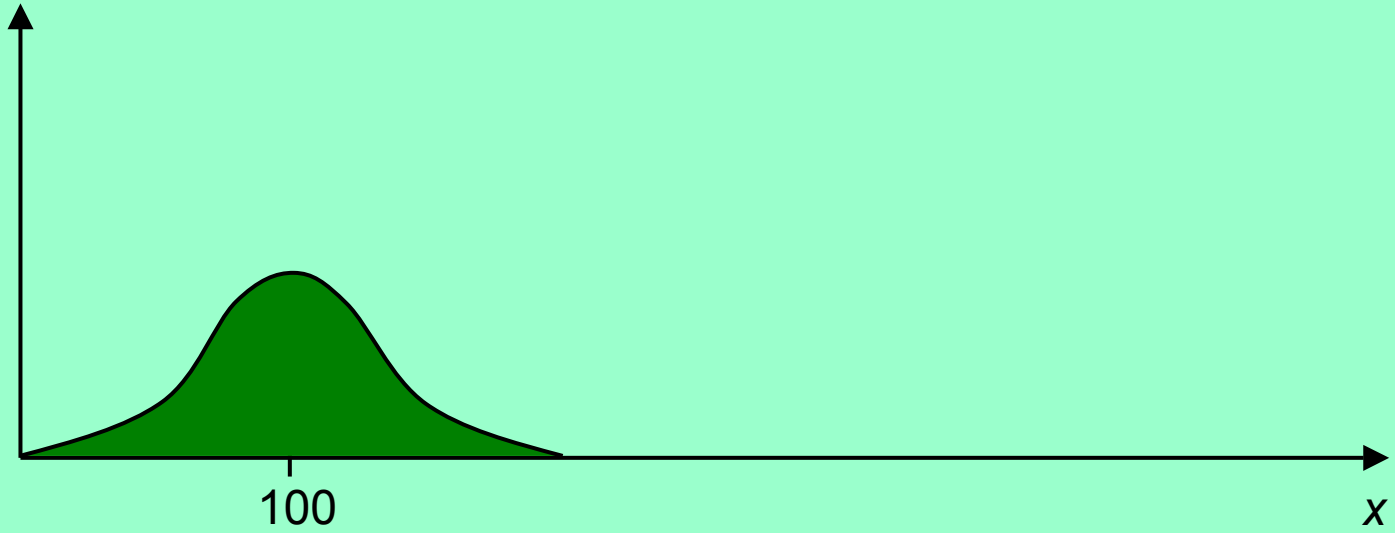
where ρ denotes the *correlation coefficient*

- The denominator normalizes the correlation coefficient such that

$$\rho \in [-1,1]$$

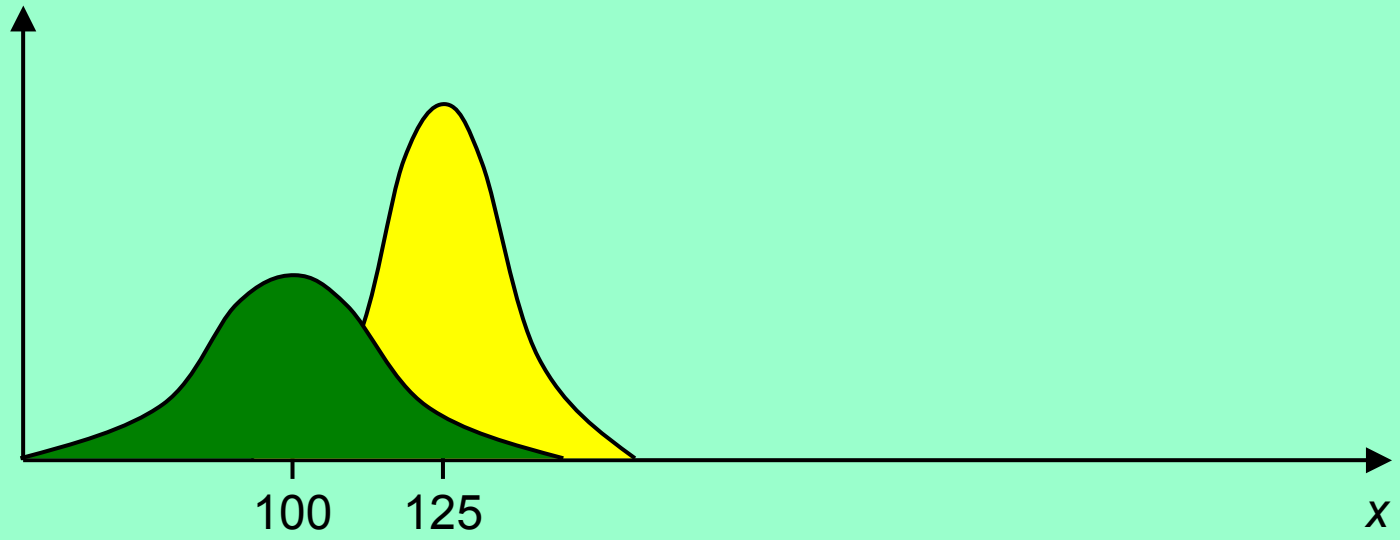
Our Simple Example Revisited

- Consider again the ship trying to estimate its position.
- The position *x* from the stars was estimated as $z_1 = 100$ with a precision of $\sigma_x = 4$ miles



The Simple Example (cont'd)

- The second and more experienced navigator (btw, a woman…) took her own sighting z_2
- She estimated the position $x = z_2 = 125$ with a precision of $\sigma_x = 3$ miles



A Simple Example (cont'd)

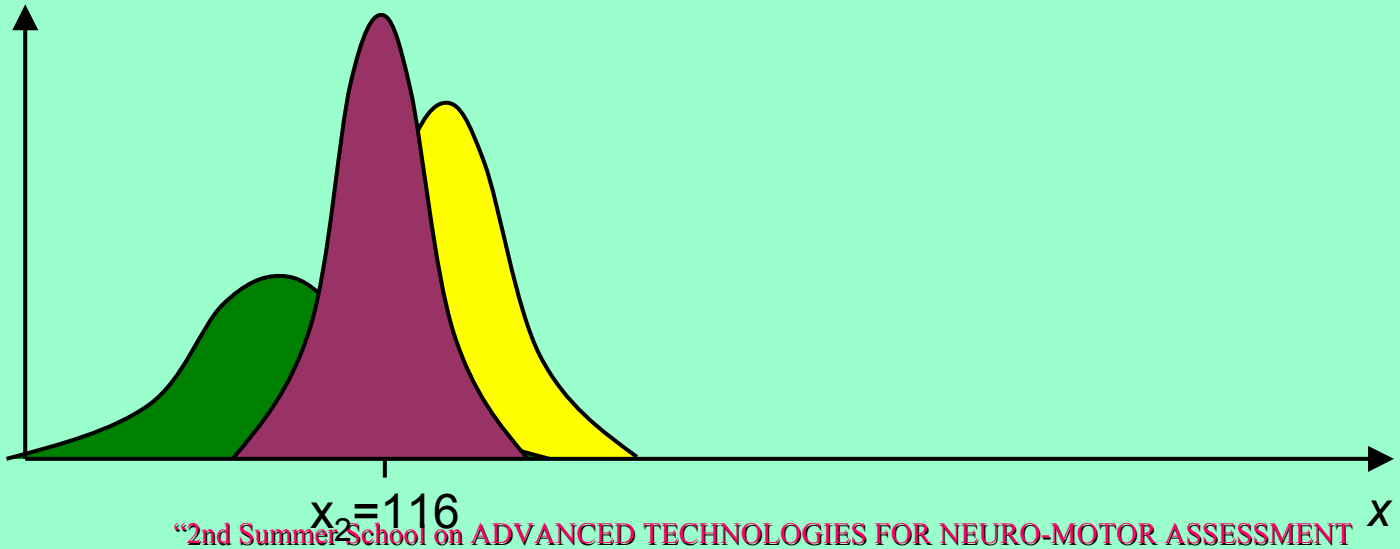
$$\mu = \left[\frac{\sigma_{z_2}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_2$$

$$= \left[\frac{9}{16+9}\right] 100 + \left[\frac{16}{16+9}\right] 125 = 116$$

$$\frac{1}{\sigma^2} = \frac{1}{\sigma_{z_1}^2} + \frac{1}{\sigma_{z_2}^2}$$

$$\frac{1}{\sigma^2} = \frac{1}{9} + \frac{1}{16} = \frac{25}{144}$$

$$\Rightarrow \sigma = 2.4$$



The Simple Example (cont'd)

- ✓ With the distributions being Gaussian, the best estimate for the state is the mean of the distribution, so...

$$x_2 = \left[\frac{\sigma_{z_2}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right] z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z1}^2 + \sigma_{z_2}^2}\right] z_2$$

or alternately

$$= z_1 + \left[\frac{\sigma_{z_1}^2}{\sigma_{z_1}^2 + \sigma_{z_2}^2}\right](z_2 - z_1)$$

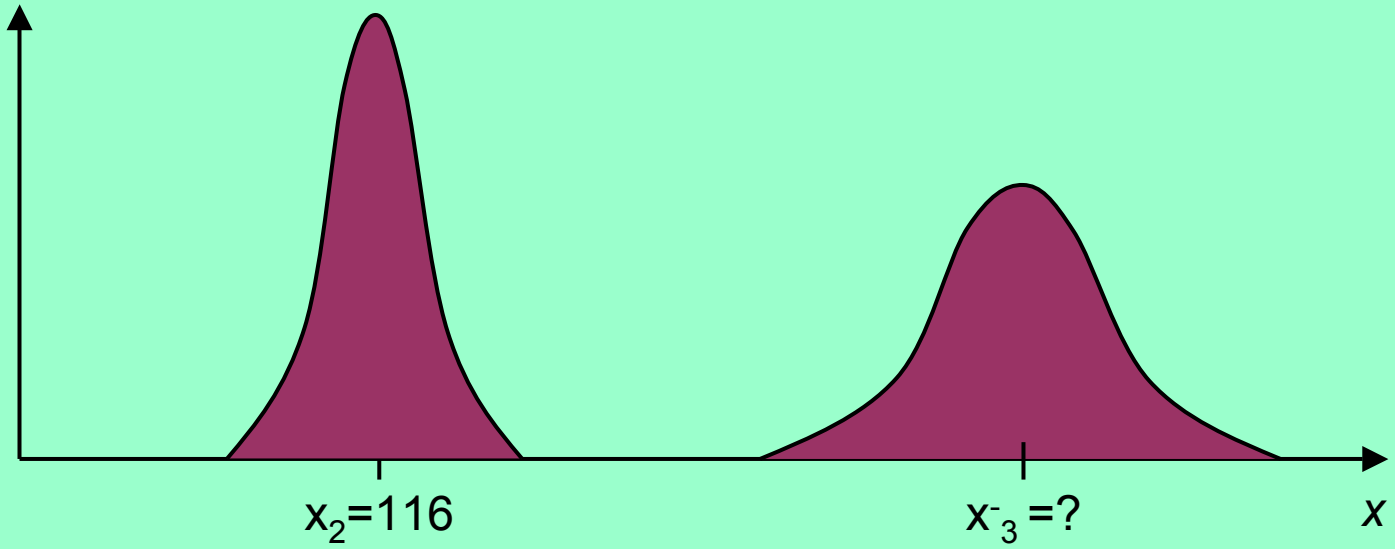
Correction Term

$$= z_1 + K_2(z_2 - z_1)$$

where K_t is referred to as the *Kalman gain*, and must be computed at each time step

A Simple Example: "Part 2"

- OK, now you fall asleep on your watch. You wake up after 2 hours, and you now have to re-estimate your position
- Let the velocity of the boat be nominally 20 miles/hour, but with a variance of σ^2_w=4 miles^2/hour
- What is the best estimate of your current position?

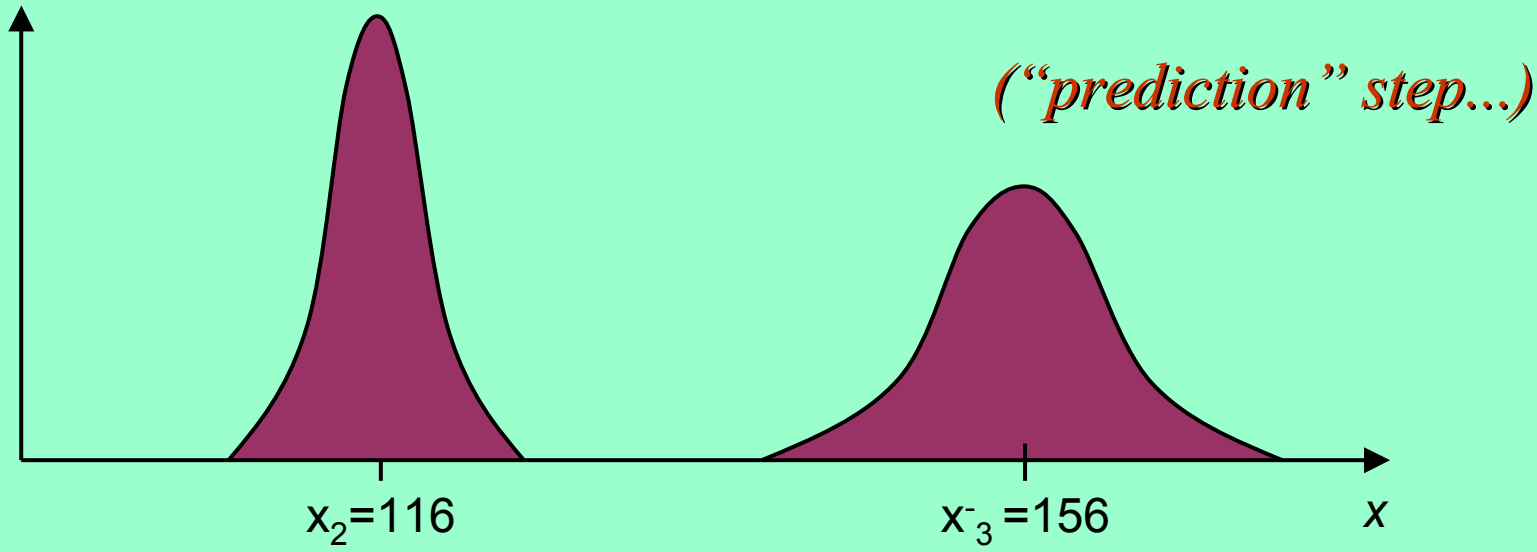


A Simple Example

- The next effect is that the Gaussian is translated by a distance and the variance of the distribution is increased to account for the uncertainty in dynamics

$$x_3^- = x_2 + v\Delta t \qquad \Rightarrow x_3^- = 116 + 40 = 156$$

$$\sigma_3^{2-} = \sigma_2^2 + \sigma_w^{2-}\Delta t \qquad \Rightarrow \sigma_3^{2-} = 5.76 + 8 = 13.76$$



A Simple Example (cont'd)

- OK, this is not a very accurate estimate. So, since you've had your nap you decide to take another measurement and you get z_3=*165* miles
- Using the same update procedure as the first update, we obtain:

$$x_3 = x_3^- + K_3(z_3 - x_3^-)$$

("correction" step...)

$$\sigma_3^2 = \sigma_3^{2-} - K_3\sigma_3^{2-}$$
$$= 13.76 - \left[\frac{13.76}{13.76+16}\right]13.76 = 7.40$$

and so on...

The Predictor-Corrector Approach

- In this example, prediction came from using knowledge of the vehicle dynamics to estimate its change in position
- The AIBO example would be integrating information from the robot kinematics (*i.e.* you give it a desired [x, y, α] velocities for a time Δt) to estimate changed in position
- The correction is accomplished through making exteroceptive observations and then fusing this with your current estimate
- This is akin to updating position estimates using landmark information, etc.

Process Model

- Describes how the state changes over time
- The state for the first example was scalar
- The *process model* was "nothing changes"

A better model might be:

- State is a 2-vector i.e. [position, velocity]
- $position_{n+1} = position_n + velocity_n * time$
- $velocity_{n+1} = velocity_n$

The Discrete Kalman Filter

- The Kalman filter addresses the problem of estimating the state $x \in R^n$ of a discrete-time controlled process governed by the linear difference equation

$$\vec{x}_{k+1} = A\vec{x}_k + B\vec{u}_k + \vec{w}_k$$

and with a measurement $z \in R^m$ that is

$$\vec{z}_k = H\vec{x}_k + \vec{v}_k$$

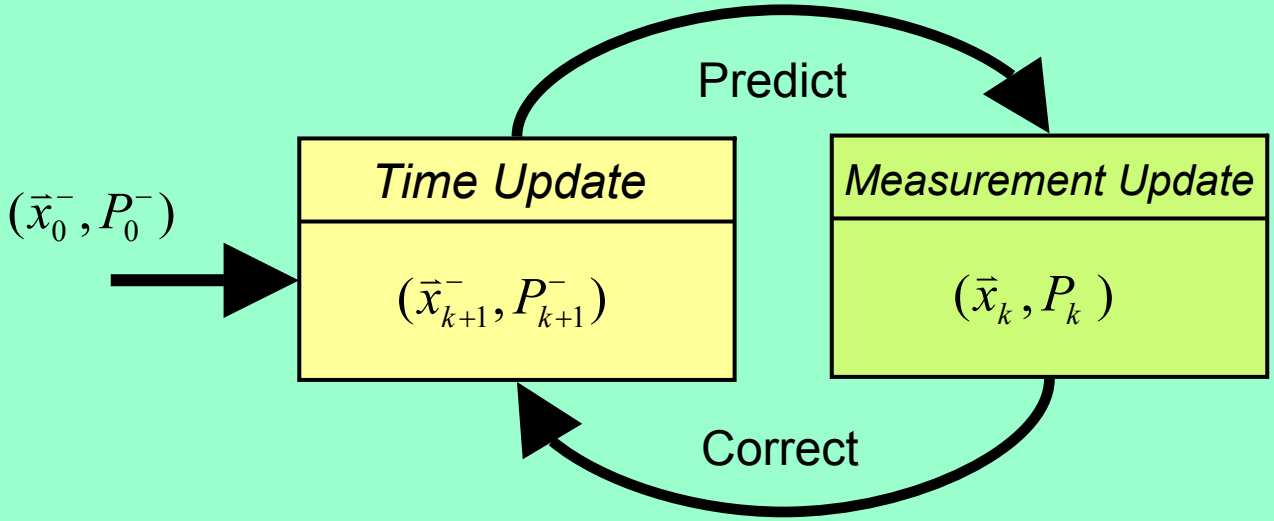
NOTE: The matrices A,B,H,Q & R may be time varying

where w_k and v_k represent the process and measurement noise. They are assumed independent, white, and with Gaussian PDFs

$$p(w) \sim N(0,Q) \qquad p(v) \sim N(0,R)$$

The Discrete Kalman Filter (cont'd)

- At each time step, the KF propagates both a *state estimate* x_k and an estimate for the *error covariance* P_k. The latter provides an indication of the uncertainty associated with the state estimate
- As mentioned previously, the KF is a predictor-corrector algorithm. Prediction comes in the *time update* phase, and correction in the *measurement update* phase



The Time Update Phase

1. Predict the state ahead

$$\vec{x}_{k+1}^{-} = A\vec{x}_k + B\vec{u}_k$$

2. Project the error covariance ahead

$$P_{k+1}^{-} = AP_kA^T + Q$$

(... i.e. from model dynamics)

Estimate Errors

- Let x_k^* denote the true state at time k
- Let x_k^- denote the *prior* estimate for x_k^*
- Let x_k denote the *posterior* estimate for x_k^*
- The corresponding estimate errors are then

$$e_k^- = x_k^* - x_k^-, \quad e_k = x_k^* - x_k$$

- The corresponding *error* covariances are then

$$P_k^- = [e_k^- e_k^{-T}], \quad P_k = [e_k e_k^T]$$

- To minimize the *posterior* error covariance, the *Kalman gain* takes the form

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

The Measurement Update Phase

1. Compute the Kalman Gain K_k

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

2. Update the estimate based on the new measurement z_k

$$\vec{x}_k = \vec{x}_k^- + K(\vec{z}_k - H\vec{x}_k^-)$$

3. Update the error covariance

$$P_k = (I - K_k H) P_k^-$$

The Kalman Gain

- The optimal Kalman gain $\mathbf{K}_k$ is:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^T (\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R})^{-1}$$

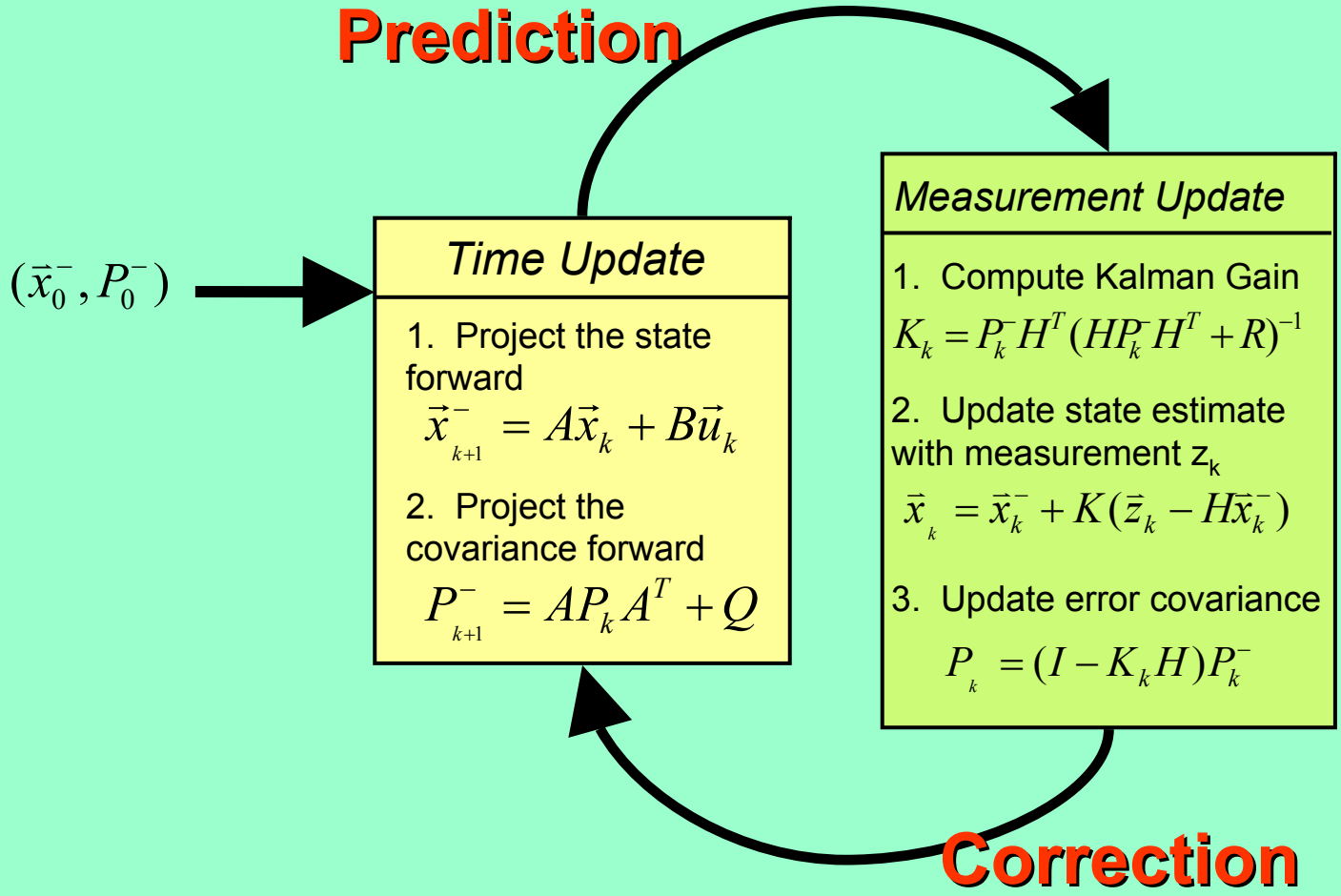
$$= \frac{\mathbf{P}_k^- \mathbf{H}^T}{\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}}$$

real values

- Compare with previous form:

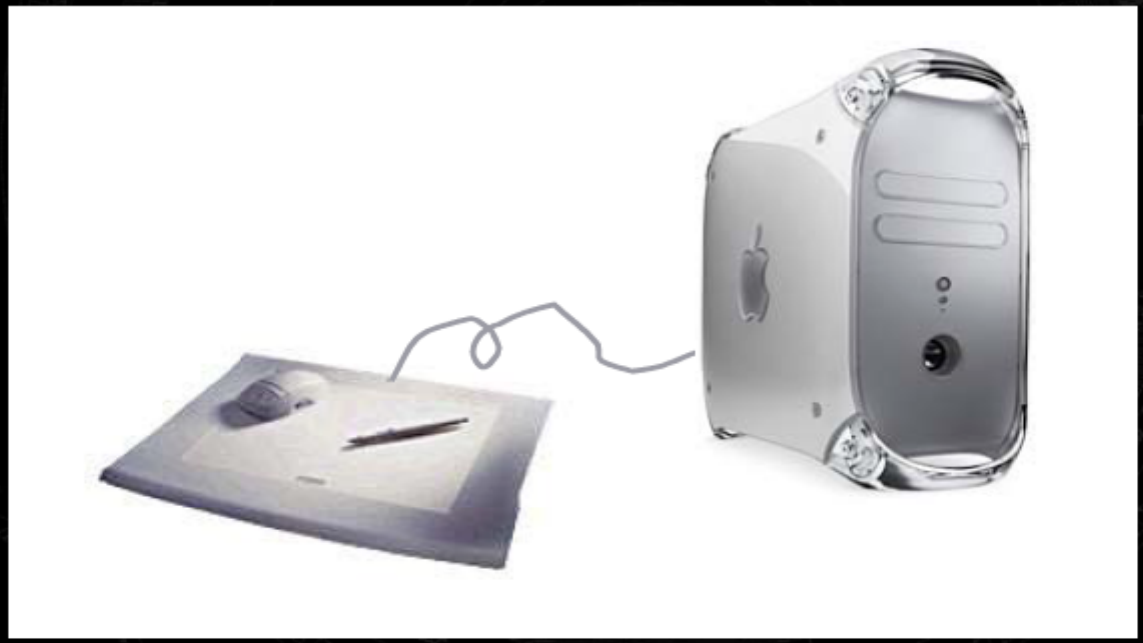
$$K(t_3) = \frac{\sigma^2(t_3^-)}{\sigma^2(t_3^-) + \sigma_3^2}$$

The Discrete Kalman Filter



One More Example: 2-D Position Only

Apparatus: 2D Tablet



Process Model

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + \begin{bmatrix} \sim x_{k-1} \\ \sim y_{k-1} \end{bmatrix}$$

state $\bar{x}_k$ — state transition A — state $\bar{x}_{k-1}$ — noise $\bar{w}_{k-1}$

$$\bar{x}_k = A\bar{x}_{k-1} + \bar{w}_{k-1}$$

Measurement Model

$$\begin{bmatrix} u_k \\ v_k \end{bmatrix} = \begin{bmatrix} H_x & 0 \\ 0 & H_y \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} \sim u_k \\ \sim v_k \end{bmatrix}$$

measurement $\bar{z}_k$ | measurement matrix H | state $\bar{x}_k$ | noise $\bar{v}_k$

$$\bar{z}_k = H\bar{x}_k + \bar{v}_k$$

Preparation

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

State Transition

$$Q = E\{\overline{w} * \overline{w}^T\} = \begin{bmatrix} Q_{xx} & 0 \\ 0 & Q_{yy} \end{bmatrix}$$

Process Noise Covariance

$$R = E\{\overline{v} * \overline{v}^T\} = \begin{bmatrix} R_{xx} & 0 \\ 0 & R_{yy} \end{bmatrix}$$

Measurement Noise Covariance

Initialization

$$\bar{x}_0 = H\bar{z}_0$$

$$P_0 = \begin{bmatrix} \varepsilon & 0 \\ 0 & \varepsilon \end{bmatrix}$$

PREDICT

$$\bar{x}_k^- = A\bar{x}_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

transition

uncertainty

CORRECT

$$\bar{x}_k = \bar{x}_k^- + K\left(\bar{z}_k - H\bar{x}_k^-\right)$$

$$P_k = (I - KH)P_k^-$$

actual

predicted

$$K = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1}$$

"denominator"
(measurement space)

Recall:

$$\left(K = \frac{P_k^- H^T}{H P_k^- H^T + R}\right)$$

Summary

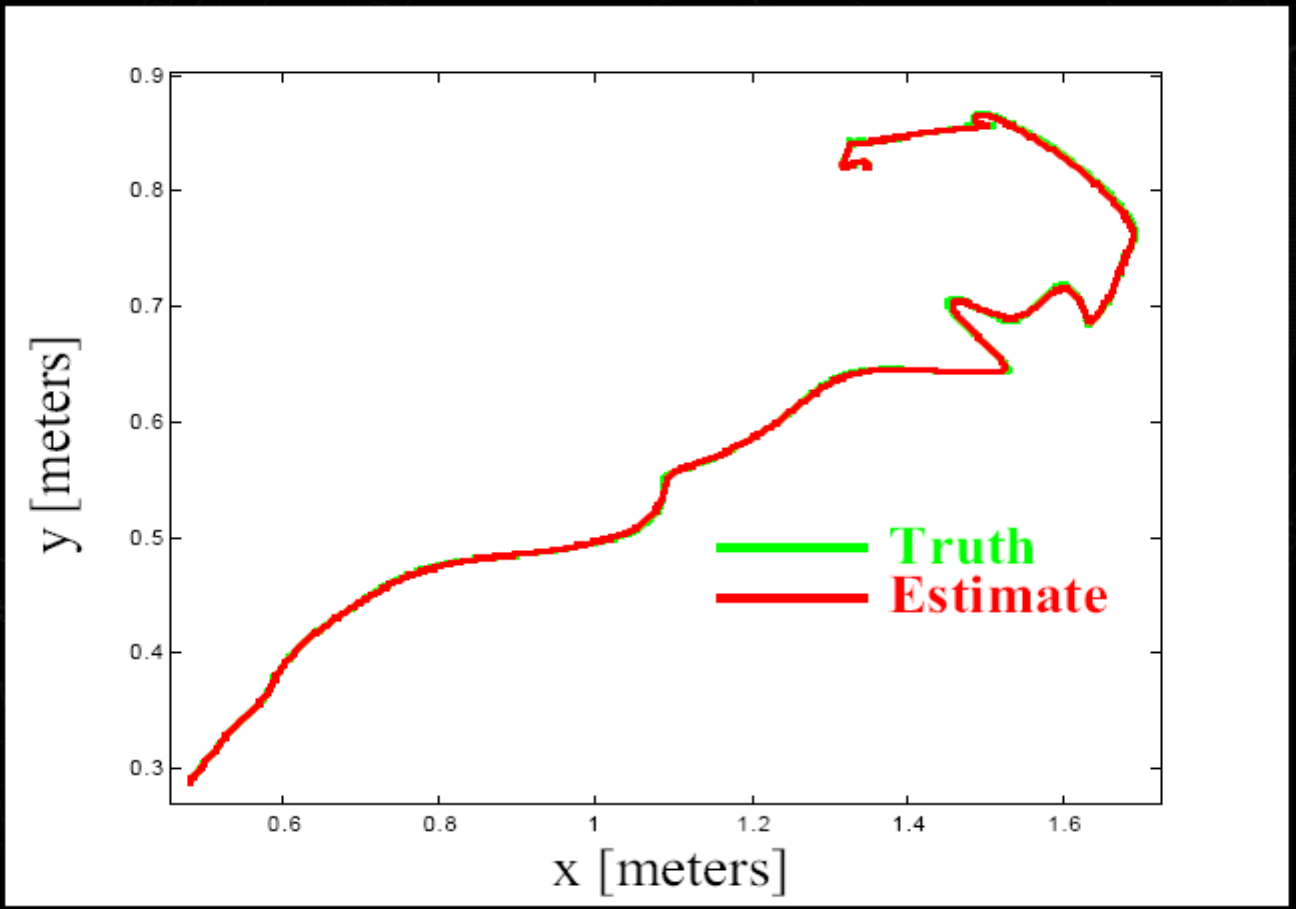
Prediction

$$\bar{x}_k^- = A\bar{x}_{k-1}$$
$$P_k^- = AP_{k-1}A^T + Q$$

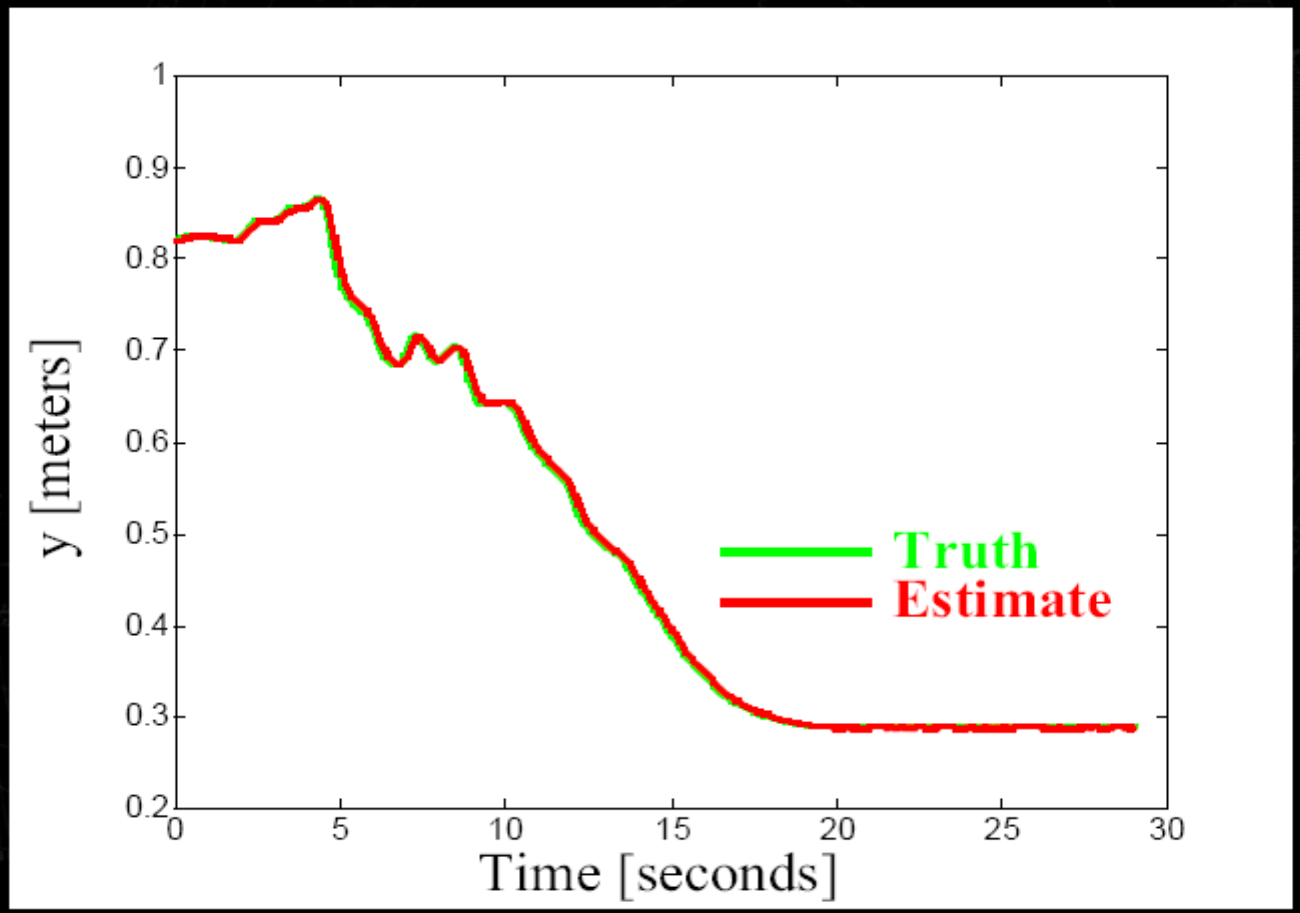
$$K = P_k^- H^T \left(HP_k^- H^T + R\right)^{-1}$$
$$\bar{x}_k = \bar{x}_k^- + K\left(\bar{z}_k - H\bar{x}_k^-\right)$$
$$P_k = (I - KH)P_k^-$$

Correction

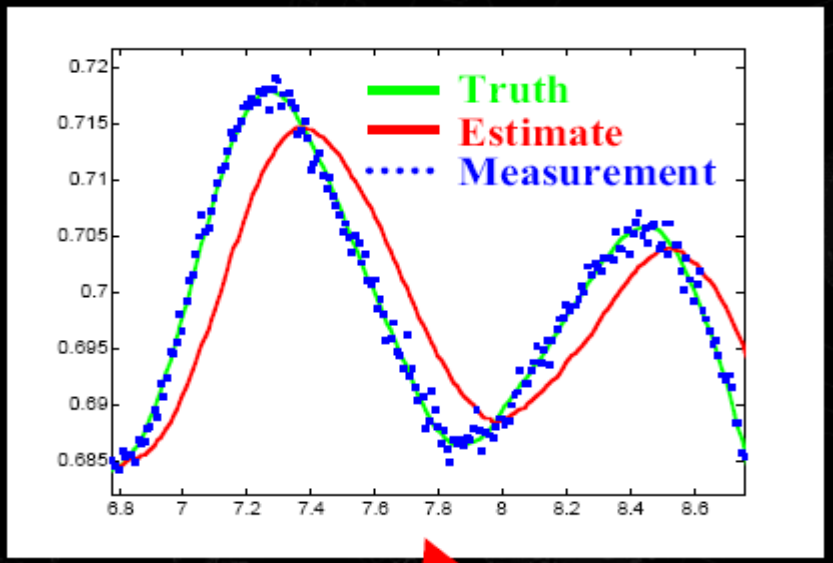
Results: XY Track

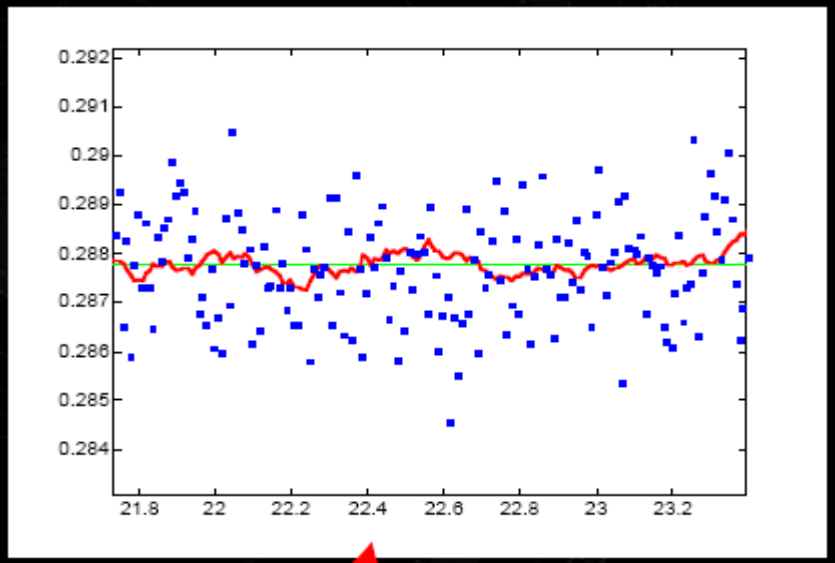


Y Track: Moving then Still

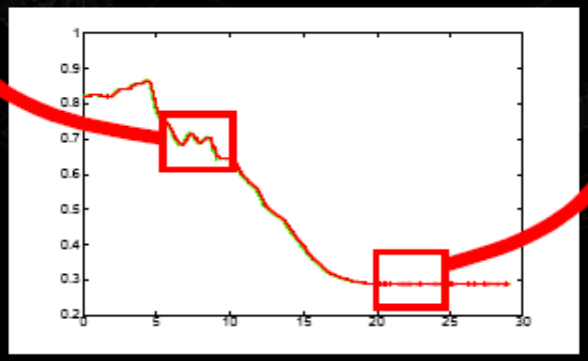


Motion-Dependent Performance





significant *latency* when moving…



…relatively *smooth* when not

Another Example
2-D Position-Velocity

Process Model (PV)

state transition

$$\begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

state

$$\begin{bmatrix} x \\ y \\ dx/dt \\ dy/dt \end{bmatrix}$$

Measurement Model (Same)

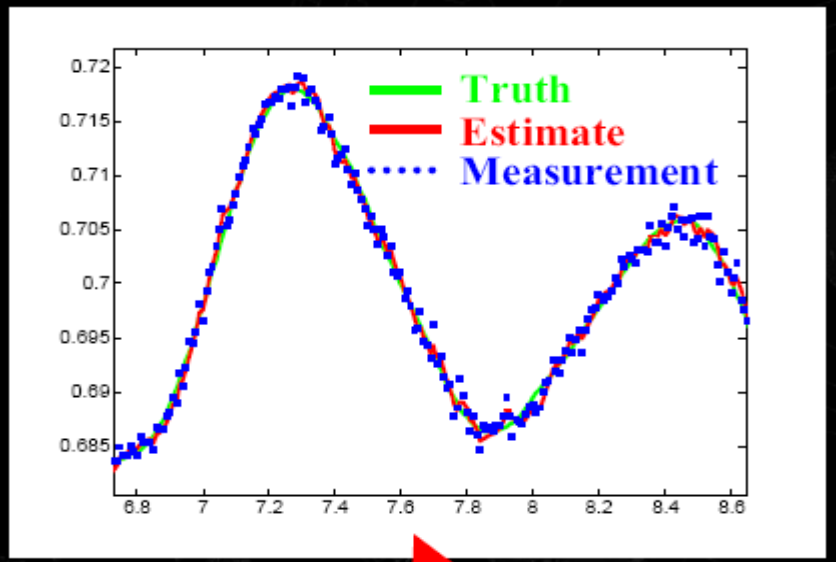
measurement matrix

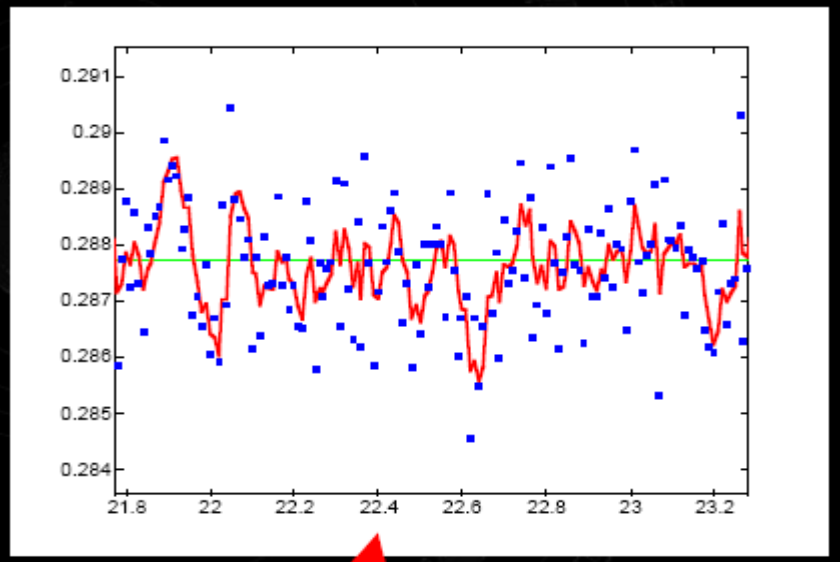
$$\begin{bmatrix} H_x & 0 & 0 & 0 \\ 0 & H_y & 0 & 0 \end{bmatrix}$$

state

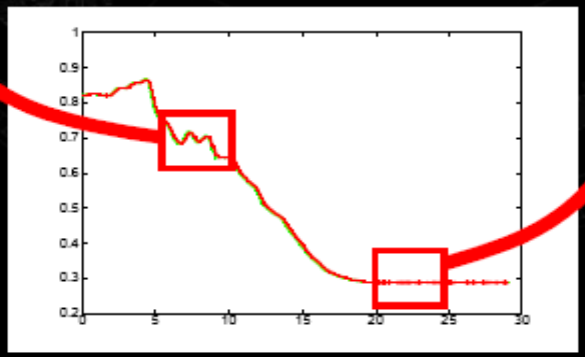
$$\begin{bmatrix} x \\ y \\ dx/dt \\ dy/dt \end{bmatrix}$$

Different Performance





improved latency when moving…



…relatively *noisy* when not

KF Summary & Remarks

- For *linear processes* with *white Gaussian noise*, the KF provides a provably optimal means for fusing measurement information
- In practice, the filter is "tuned" empirically by finding "optimal" estimates for ***Q*** and ***R***
- The linear constraints can be lifted, but the optimality of the filter is lost
- This is the basis for the *Extended Kalman Filter (EKF)*

Nonlinear Systems
Few Details...

Kalman Filter assumes linearity

- Only matrix operations allowed
- Measurement is a linear function of state
- Next state is linear function of previous state
- Can't estimate gain
- Can't handle rotations (angles in state)
- Can't handle projection

Extended Kalman Filter

Nonlinear Process (Model)

- Process dynamics: A becomes $a(x)$
- Measurement: H becomes $h(x)$

Filter Reformulation

- Use functions instead of matrices
- Use Jacobians to project forward, and to relate measurement to state

Jacobian?

- Partial derivative of measurement with respect to state
- If measurement is a vector of length M
- And state has length N
- Jacobian of measurement function will be MxN matrix of numbers (not equations)
- Often evaluating h(x) and Jacobian(h(x)) at the same time cost only a little extra

Extended Kalman Filter

- Suppose the state-evolution and measurement equations are non-linear:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \mathbf{w}_{k-1}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k$$

  - process noise **w** is drawn from $N(0,\mathbf{Q})$, with covariance matrix **Q**.
  - measurement noise **v** is drawn from $N(0,\mathbf{R})$, with covariance matrix **R**.

The Jacobian Matrix

- For a scalar function $y=f(x)$,

$$\Delta y = f'(x)\Delta x$$

- For a vector function $\mathbf{y}=f(\mathbf{x})$,

$$\Delta \mathbf{y} = \mathbf{J}\,\Delta \mathbf{x} = \begin{bmatrix} \Delta y_1 \\ \vdots \\ \Delta y_n \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1}(\mathbf{x}) & \cdots & \frac{\partial f_n}{\partial x_n}(\mathbf{x}) \end{bmatrix} \cdot \begin{bmatrix} \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

Linearise the Non-Linear

- Let **A** be the Jacobian of *f* with respect to **x**.

$$\mathbf{A}_{ij} = \frac{\partial f_i}{\partial x_j}(\mathbf{x}_{k-1}, \mathbf{u}_k)$$

- Let **H** be the Jacobian of *h* with respect to **x**.

$$\mathbf{H}_{ij} = \frac{\partial h_i}{\partial x_j}(\mathbf{x}_k)$$

- Then the Kalman Filter equations are almost the same as before!

EKF Update Equations

- Predictor step: $\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_k)$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$$

- Kalman gain: $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1}$

- Corrector step: $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-))$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$$

Some comments regarding Kalman filtering...

Why Kalman "Filter"?

- The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.
- In contrast to batch estimation techniques, no history of observations and/or estimates is required.
- It is unusual in being purely a time domain filter; most filters (for example, a low-pass filter) are formulated in the frequency domain and then transformed back to the time domain for implementation.

Why Kalman "Filter"? (Cont'd)

- The Kalman filter can be regarded as an adaptive low-pass infinite impulse response digital filter, with cut-off frequency depending on the ratio between the process- and measurement (or observation) noise, as well as the estimate covariance.
- Frequency response is, however, rarely of interest when designing state estimators such as the Kalman Filter, whereas for digital filters such as IIR and FIR filters, frequency response is usually of primary concern.
- For the Kalman Filter, the important goal is how accurate the filter is, and this is most often decided based on empirical Monte Carlo simulations, where "truth" (the true state) is known.

1-D Example: Estimating a Random Constant

- Suppose we are trying to estimate the value of a 1D constant from corrupted sensor measurements. Our process model is then

$$x_{k+1} = Ax_k + Bu_k + w_k = x_k + w_k$$

$$z_k = Hx_k + v_k = x_k + v_k$$

- The KF equations then are

Variance of our measurement device

Variance of our state estimate

Variance of our signal level

Time Update

$$x_{k+1}^- = x_k$$

$$P_{k+1}^- = P_k + Q$$

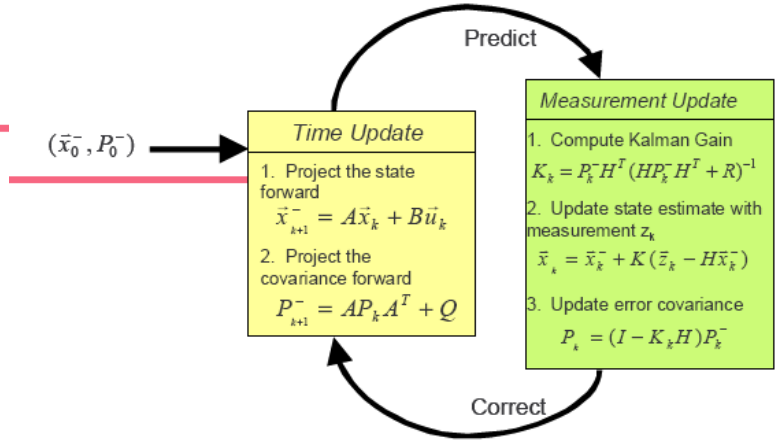
Measurement Update

$$K_k = P_k (P_k + R)^{-1}$$

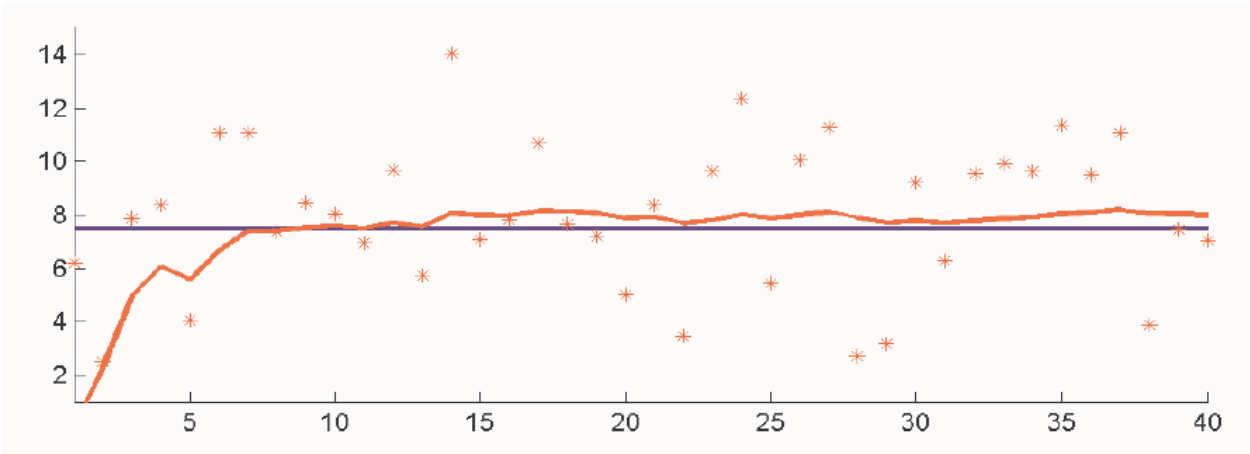
$$x_k = x_k^- + K(z_k - x_k^-)$$

$$P_k = (I - K_k) P_k^-$$

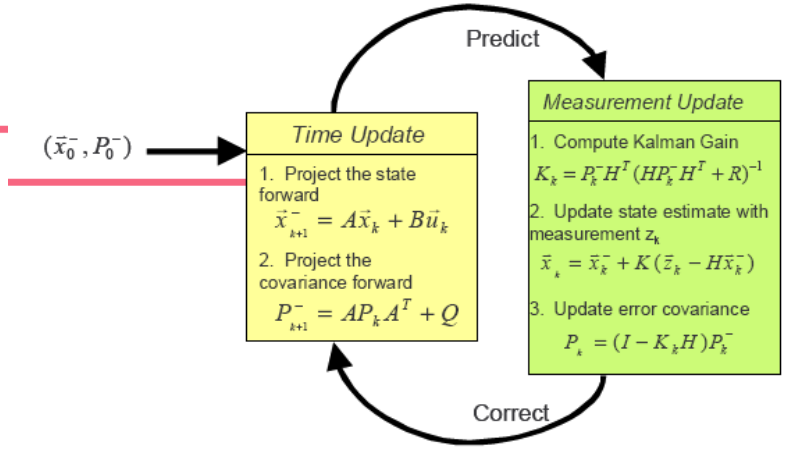
Simulation Results (1)



- Let us assume that x^*=7.5, **Q**=0.01, **R**=9
- With perfect knowledge of the process and sensor covariance model, we obtain

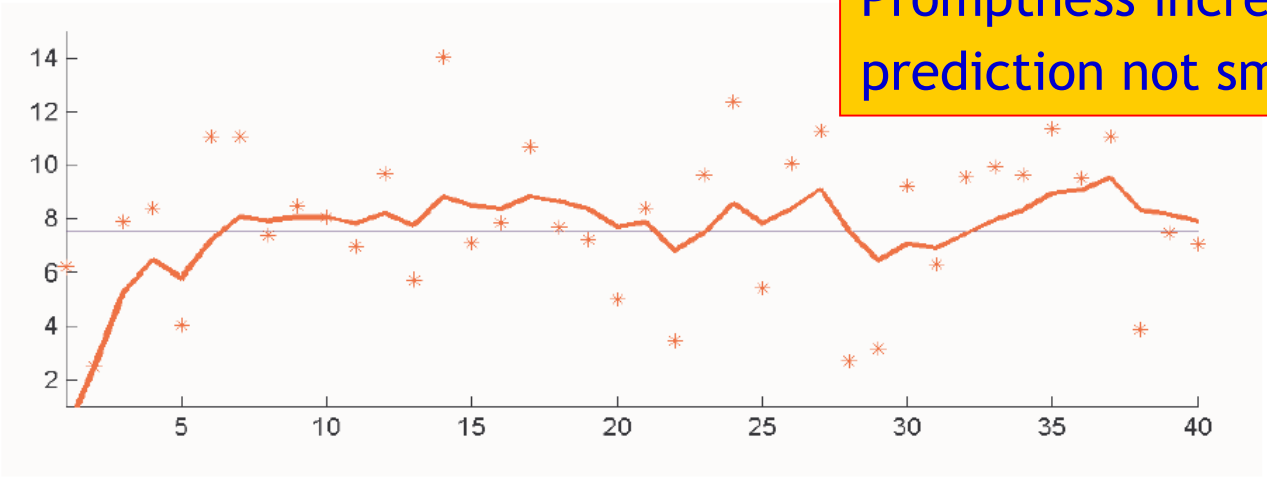


Simulation Results (2)

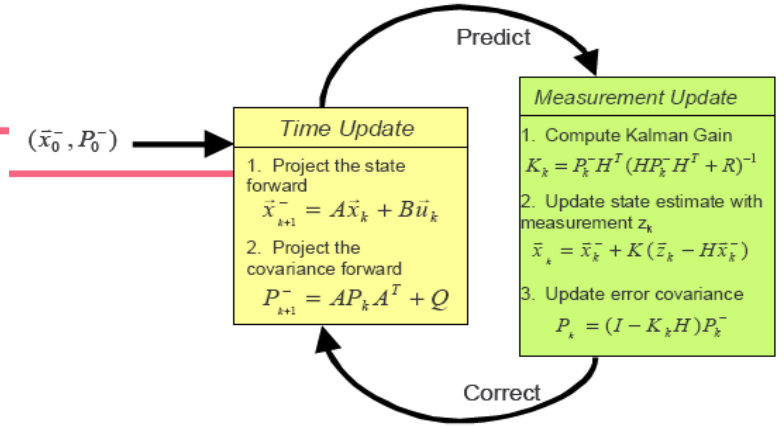


- Let us assume that x^*=7.5, Q=0.01, R=9
- Let us further assume that the user believes that the sensor covariance R= 0.09

Promptness increased but prediction not smooth!

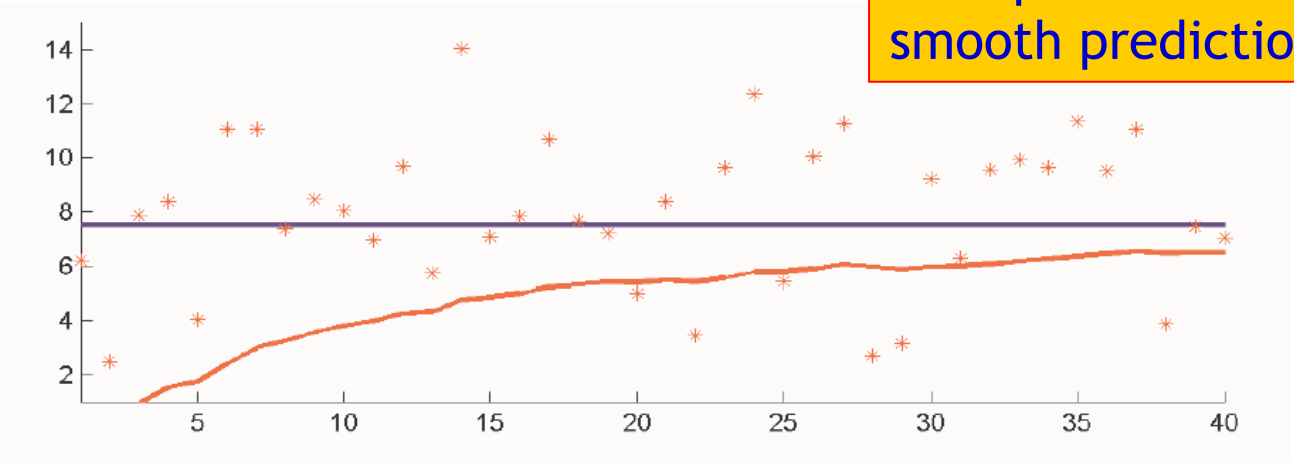


Simulation Results (3)



- Let us assume that x^*=7.5, Q=0.01, R=9
- Let us further assume that the user believes that the sensor covariance R= 900

Promptness decreased but smooth prediction



Common Applications

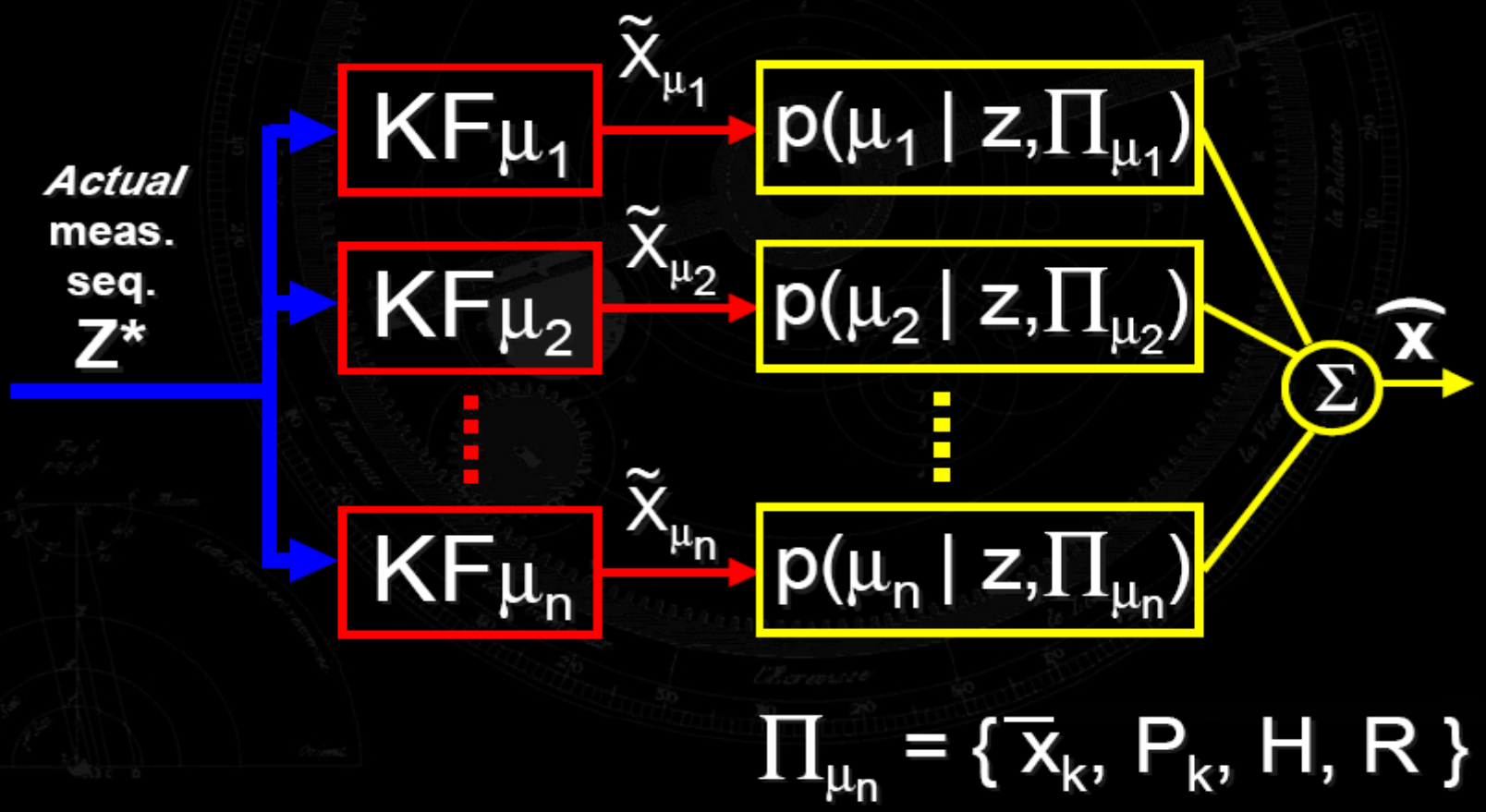
- Tracking missiles
  - Tracking moving objects
- GPS
- Computer vision
  - Extracting lip motion from video
- Data fusion/integration
  - Integration of spatio-temporal video segments
- Robotics
  - Robust estimation and sensor data noise reduction
- State and parameter estimation for monitoring, fault diagnosis and control
- Data smoothing and curve fitting

Kalman Filters for Sensor Fusion Implementation

Multiple Sensor Implementations

On-line Model/Sensor Selection

On-Line Multiple-Model Estimation



Probability of Model μ

For model μ with $\Pi_\mu = \{x, P, H, R\}$

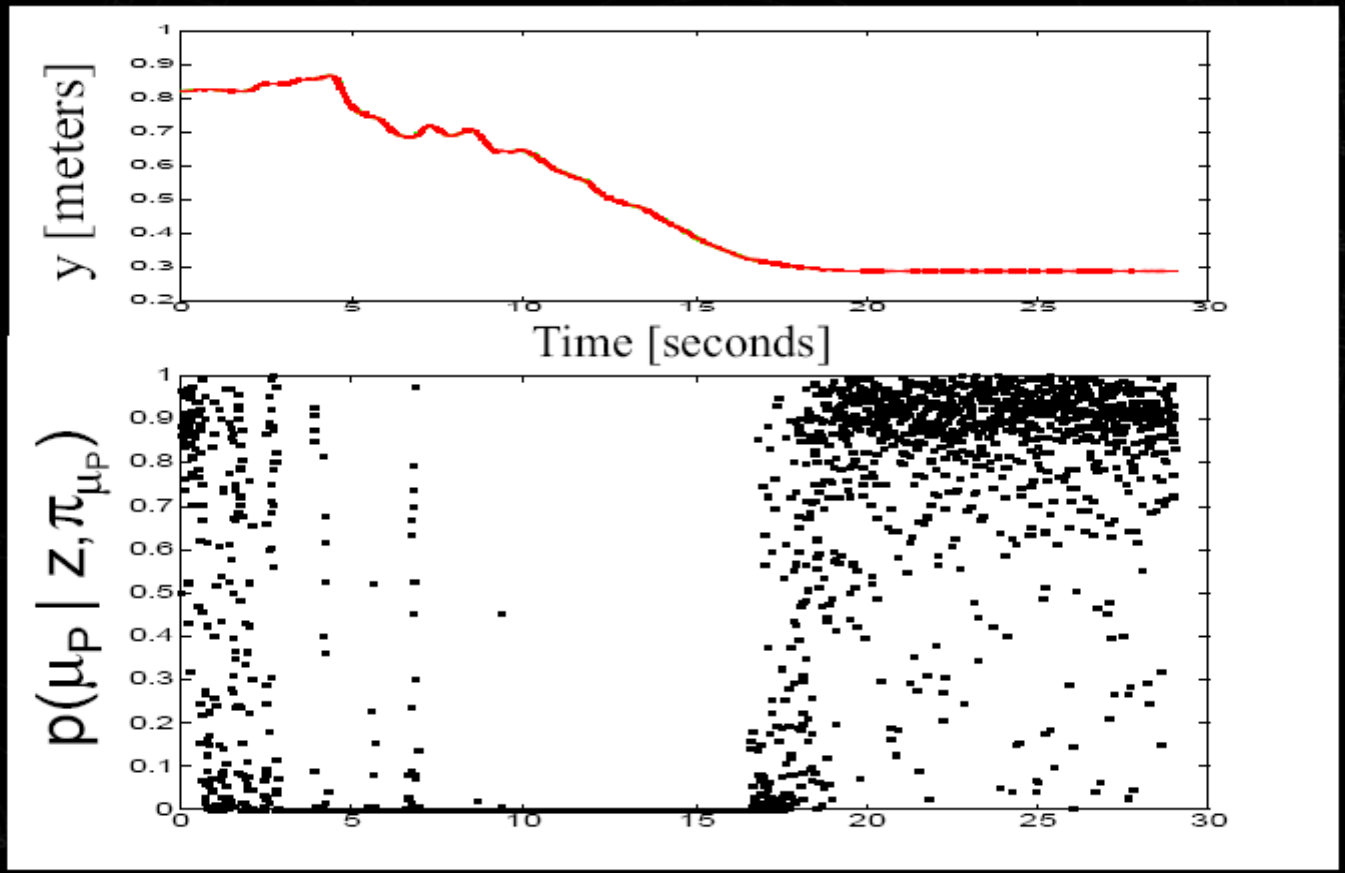
$$p(\mu|z,\Pi_\mu) = \frac{1}{(2\pi|C|)^{\frac{n}{2}}} e^{-\frac{1}{2}(z-Hx)^T C^{-1}(z-Hx)}$$

where $C = HPH^T + R$

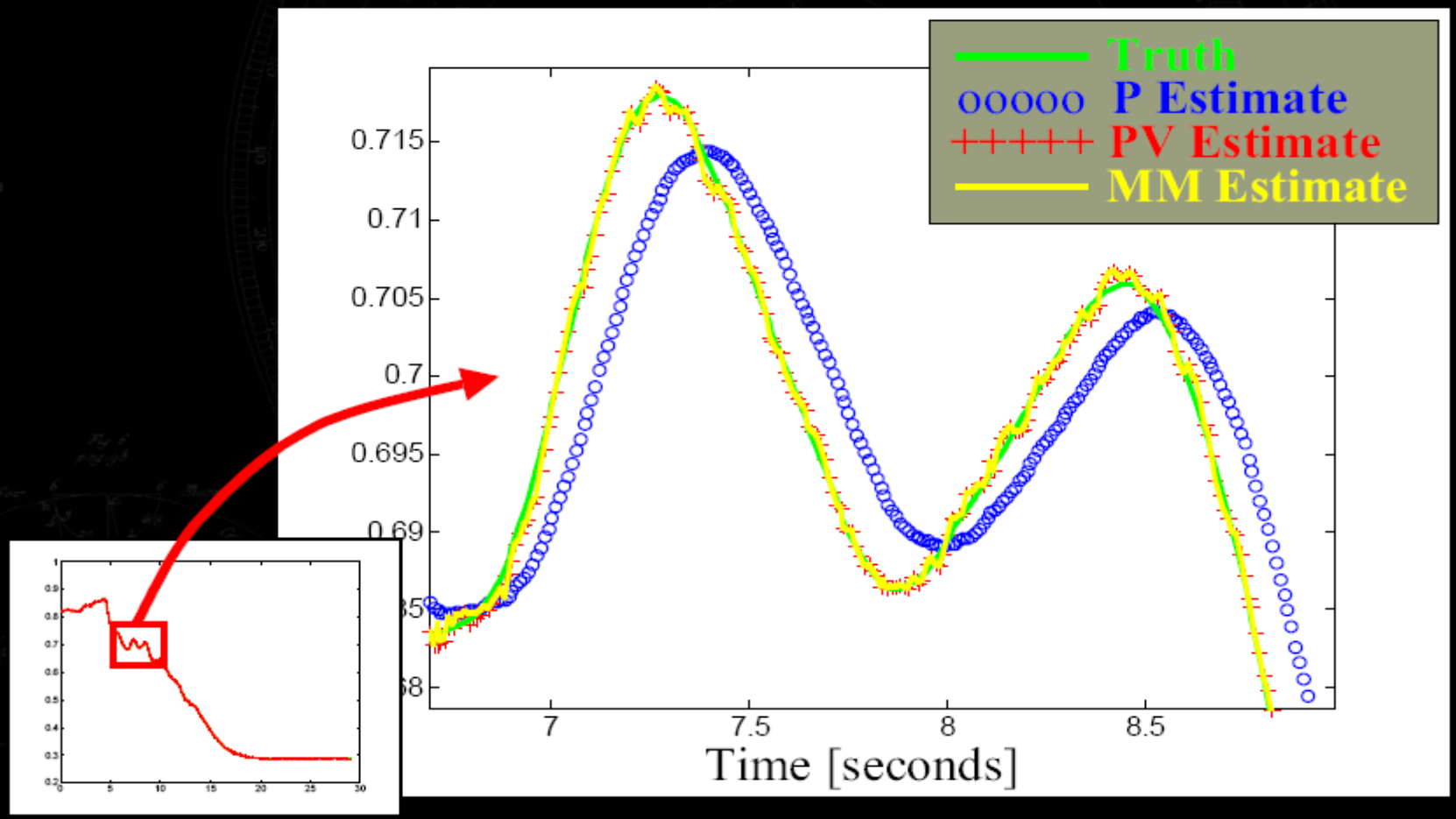
Final Combined Estimate

$$\hat{x} = \sum_{\mu} x_{f\mu} \frac{p(\mu | z, \Pi_{\mu})}{\sum_{\nu} p(\nu | z, \Pi_{\nu})}$$

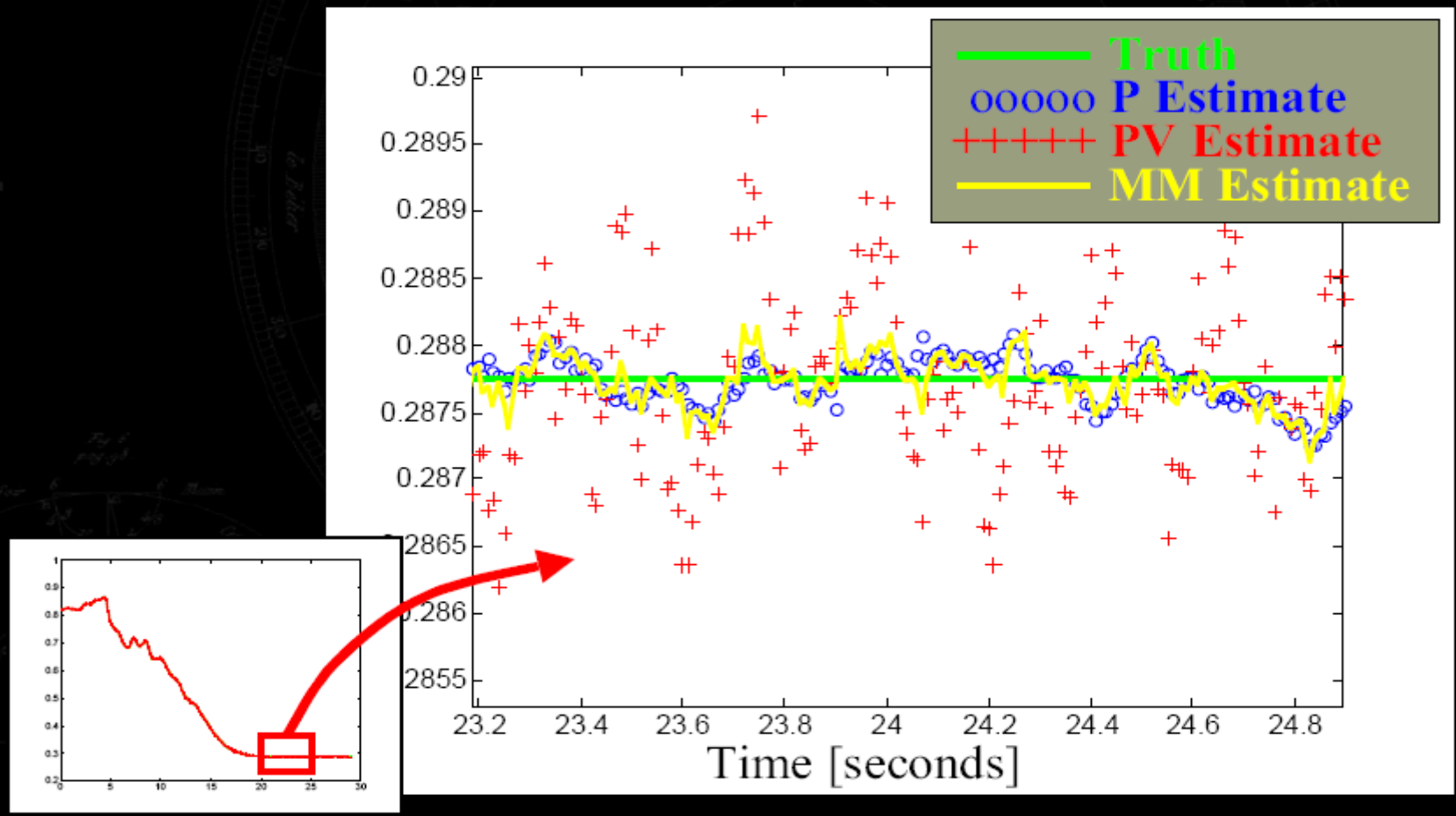
MME Weighting



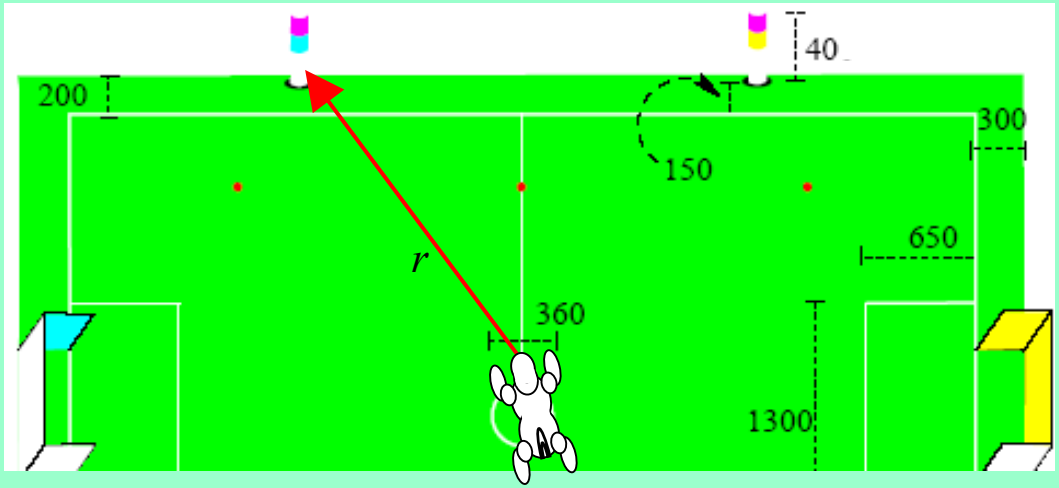
Low-Latency During Motion



Smooth When Still

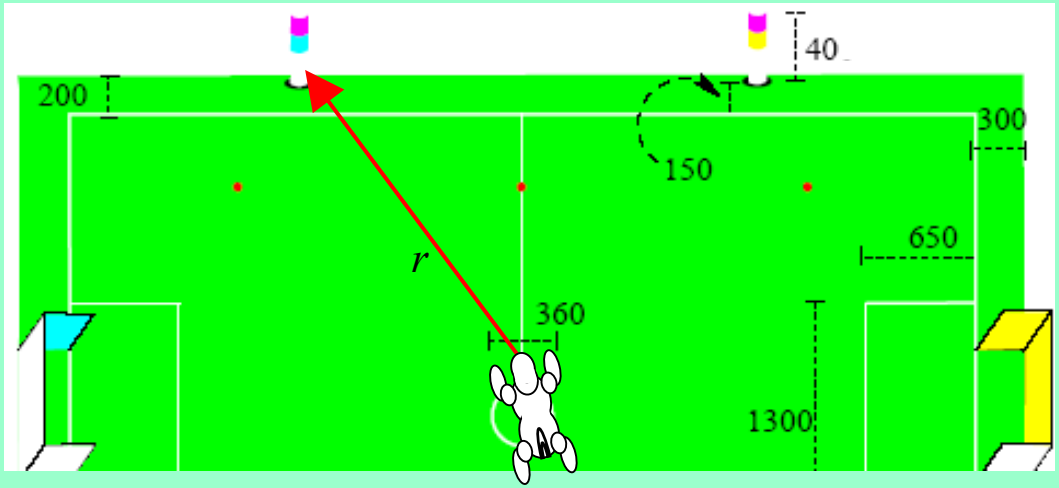


AIBO Navigation System Example



- Let's say your AIBO takes 3 measurements of the distance to a beacon as $Z = [2000, 1900, 2100]^T$
- What would be your estimate of the beacon distance?
- Well, a good estimate might be the mean of the 3 sensor values:

$$r = E(Z) = \sum_{i=1}^{3} p_i z_i = \frac{2000 + 1900 + 2100}{3} = 2000$$



- Now let's say your AIBO takes 3 measurements of the distance to a beacon using another group's shoddy code and you get Z = $[2000, 900, 3100]^T$
- We could again use the mean as the range estimate and obtain

$$r = E(Z) = \sum_{i=1}^{3} p_i z_i = \frac{2000 + 900 + 3100}{3} = 2000$$

- Would you have as much confidence in this estimate as the first?

Kalman Filter for Data Fusion

- The main idea behind the Kalman filter is that you do not just have an estimate for a parameter *x* but also have some estimate for the uncertainty in your value for *x*
- This is represented by the variance/covariance of the estimate P_x
- There are many advantages to this, as it allows you a means for estimating the confidence in your robot's ability to execute a task (*e.g.* navigating through a tight doorway)
- In the case of the KF, it also provides a nice mechanism for *optimally* combining data over time
- This optimality condition assumes we have linear models, and the error characteristics of our sensors can be modeled as zero-mean, Gaussian noise.

Data Fusion Example

- OK, let's say we use code from Team 1 and Team 2 to obtain two different measurements $Z = [z_1, z_2]^T$ for the range r to a beacon
- Let us further assume that the variance in each of these sensor measurements is R_1 and R_2, respectively
- How should we fuse these measurements in order to obtain the "best" possible resulting estimate for r ?
- We'll define "best" from a least-squares perspective…
- We have 2 measurements that are equal to r plus some additive zero-mean Gaussian noise v_1 and v_2

$$z_1 = r + N(0, R_1) = r + v_1$$
$$z_2 = r + N(0, R_2) = r + v_2$$

A Least-Squares Approach

$$z_1 = r + N(0, R_1) = r + v_1$$
$$z_2 = r + N(0, R_2) = r + v_2$$

- We want to fuse these measurements to obtain a new estimate for the range $\hat{r}$
- Using a weighted least-squares approach, the resulting sum of squares error will be

$$e = \sum_{i=1}^{n} w_i (\hat{r} - z_i)^2$$

- Minimizing this error with respect to $\hat{r}$ yields

$$\frac{\partial e}{\partial \hat{r}} = \frac{\partial}{\partial \hat{r}} \sum_{i=1}^{n} w_i (\hat{r} - z_i)^2 = 2 \sum_{i=1}^{n} w_i (\hat{r} - z_i) = 0$$

A Least-Squares Approach (cont'd)

- Rearranging we have

$$\sum_{i=1}^{n} w_i \hat{r} - \sum_{i=1}^{n} w_i z_i = 0$$

$$\Rightarrow \hat{r} = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i}$$

- If we choose the weight to be

$$w_i = \frac{1}{\sigma_i^2} = \frac{1}{R_i}$$

we obtain

$$\hat{r} = \frac{\frac{z_1}{R_1} + \frac{z_2}{R_2}}{\frac{1}{R_1} + \frac{1}{R_2}} = \frac{R_2}{R_1 + R_2} z_1 + \frac{R_1}{R_1 + R_2} z_2$$

A Least-Squares Approach

- This can be rewritten as

$$\hat{r} = z_1 + \frac{R_1}{R_1 + R_2}(z_2 - z_1)$$

Kalman Gain

or if we think of this as adding a new measurement to our current estimate of the state we would get

$$\hat{r}_{k+1} = \hat{r}_{k+1}^{-} + \frac{P_{k+1}^{-}}{P_{k+1}^{-} + R}(z_{k+1} - \hat{r}_{k+1}^{-}) \qquad \Rightarrow \hat{r}_{k+1} = \hat{r}_{k+1}^{-} + K_{k+1}(z_{k+1} - \hat{r}_{k+1}^{-})$$

- For merging Gaussian distributions, the update rule is

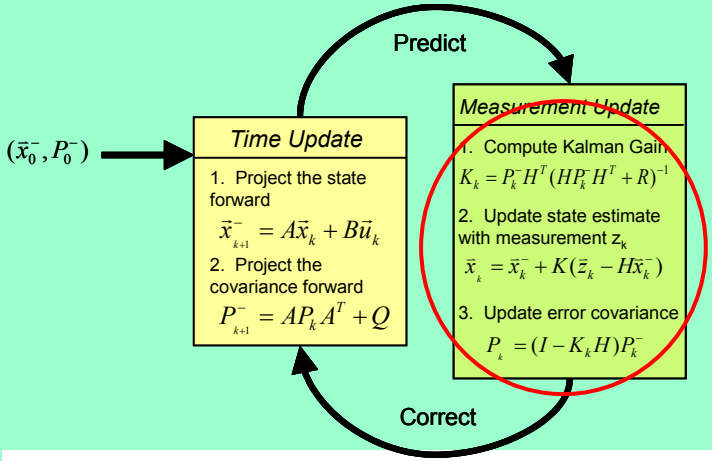
$$\frac{1}{\sigma_3^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2} = \frac{\sigma_1^2 + \sigma_2^2}{\sigma_1^2 \sigma_2^2} \quad \Rightarrow \quad \sigma_3^2 = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

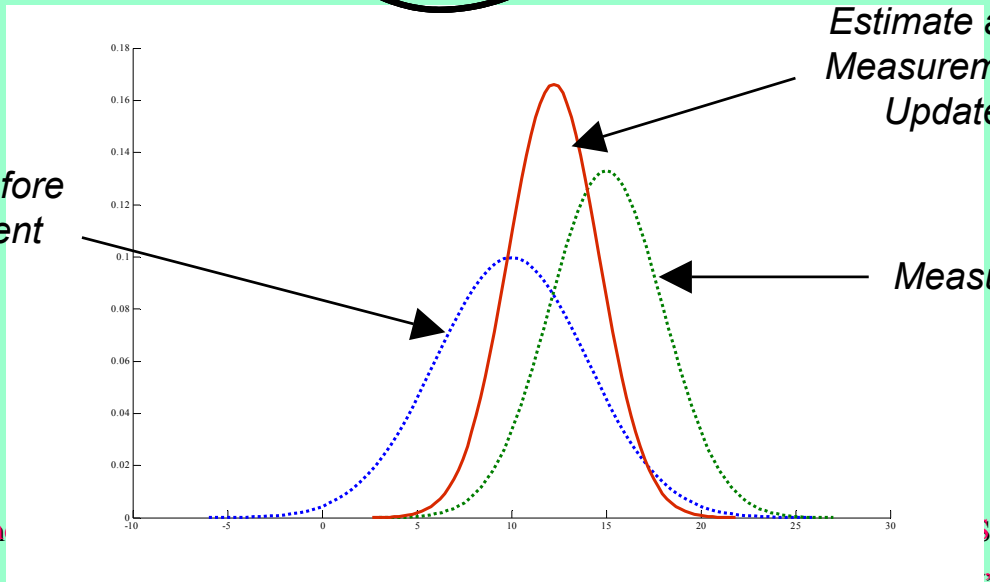
which if we write in our measurement update equation form we get

$$P_{k+1} = \frac{P_{k+1}^{-} R_{k+1}}{P_{k+1}^{-} + R_{k+1}} \equiv P_{k+1}^{-} - K_{k+1} P_{k+1}^{-}$$

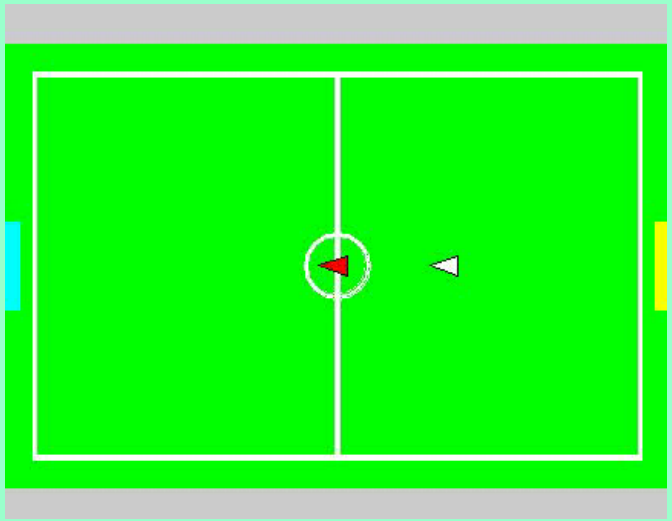
The Measurement Update Phase

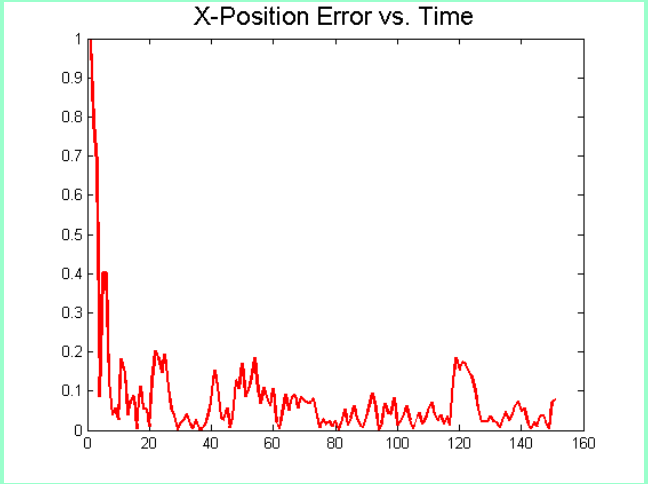
- These are the measurement update equations for the discrete Kalman filter

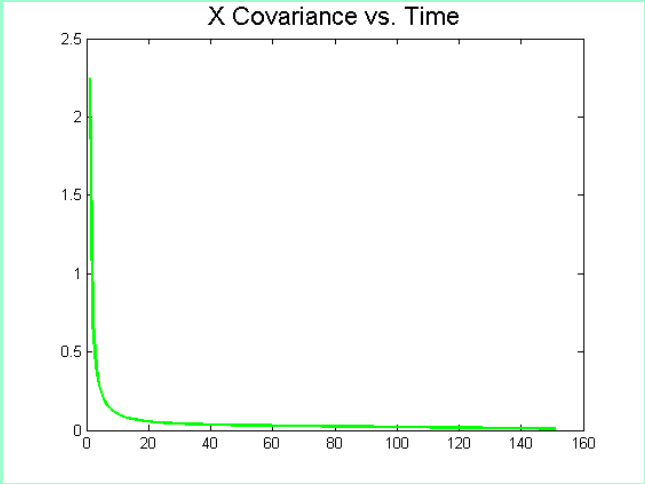




Filter Simulation in Action







Kalman Filter Localization Example

- Let's say that we are going to use a Kalman filter to localize our AIBO based upon taking range measurement(s) to beacons
- We could write our state update equation as

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k} + \begin{bmatrix} V_x \Delta t \\ V_y \Delta t \\ \omega \Delta t \end{bmatrix}_{k}$$

- This looks great as it's nice and linear. Now let's look at our measurement equations taking range to a beacon at (x_b, y_b)

$$r_k = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2}$$

- But, we have a problem…

Kalman Filter & EKF

- Time update and measurement equations are *NOT* linear
- The KF is not applicable
- We linearise around the nonlinearities, we can still get good performance in practice
- Development of the Extended Kalman Filter (EKF)
- By relaxing the linear assumptions, the use of the KF is extended dramatically
- We can no longer use the word "optimal" with the EKF

EKF Time Update Phase

- So the covariance is projected ahead as

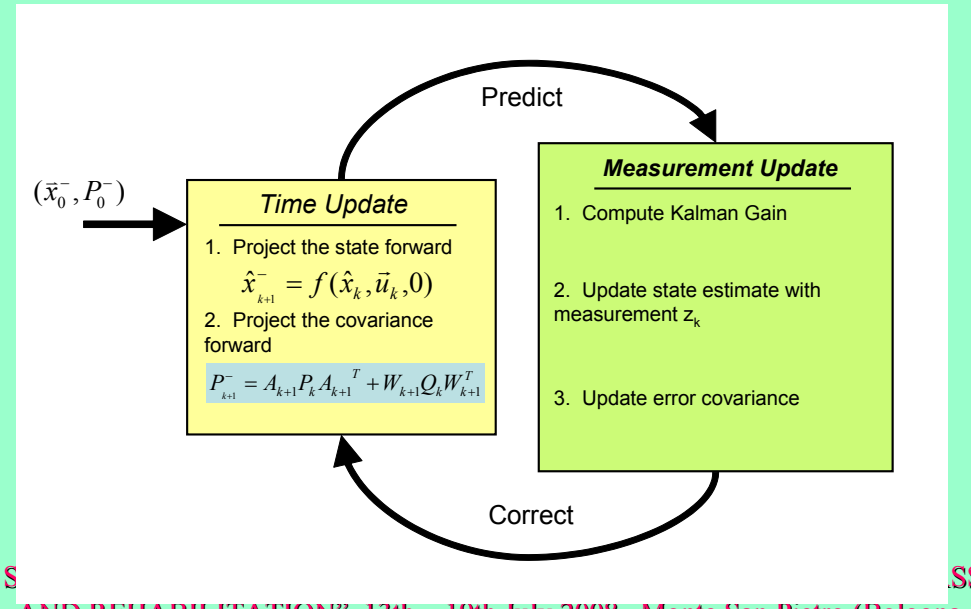
$$P_{k+1}^{-} = AP_kA^T + Q$$

Kalman Filter

$$P_{k+1}^{-} = AP_kA^T + WQW^T$$

Extended Kalman Filter

where *A* is now the Jacobian of *f* with respect to x and *W* is the Jacobian of *f* with respect to *w*



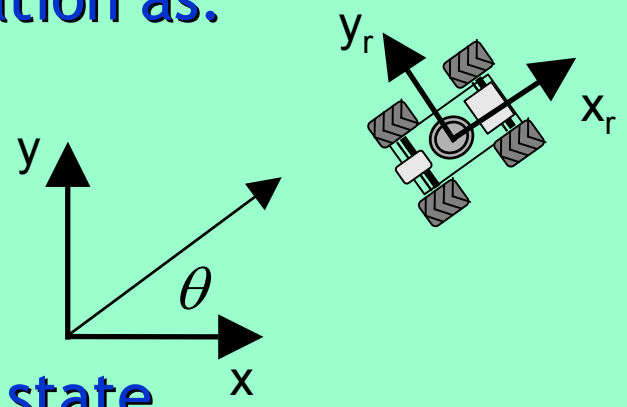
EKF Robot Implementation Example

- Assume that we have a mobile robot using odometer and range measurements to landmark to estimate its position and orientation

$$\vec{x} = [x, y, \theta]^T$$

- Assume that the odometer provides a velocity estimate *V* and an angular velocity estimate ω that are both corrupted by Gaussian noise
- We can write the state update equation as:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k+1} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_{k} + \begin{bmatrix} V\cos\theta\Delta t \\ V\sin\theta\Delta t \\ \omega\Delta t \end{bmatrix}$$



which is obviously non-linear in the state

EKF Measurement Update Phase

- Again, in the measurement update we can have a non-linear relationship between our measurements and state

$$\vec{z}_k = h(\vec{x}_k, \vec{v}_k)$$

and once again we will assume that the noise is zero

$$\vec{z}_k = h(\vec{x}_k, 0)$$

- To propagate uncertainty, we shall again have to calculate the appropriate Jacobians
  - *H* is the Jacobian relating changes in *h* to changes in our state *x*
  - *V* is the Jacobian relating changes in *h* to changes in the measurement noise *v*
- These are then substituted into the original KF as appropriate

EKF Measurement Update Phase (cont'd)

KF	EKF

- **Computing the Kalman Gain:**

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

$$K_k = P_k^- H^T (H P_k^- H^T + V R V^T)^{-1}$$

- **State Update:**

$$\vec{x}_k = \vec{x}_k^- + K_k(\vec{z}_k - H\vec{x}_k^-)$$

$$\vec{x}_k = \vec{x}_k^- + K_k(\vec{z}_k - h(\vec{x}_k^-, 0))$$

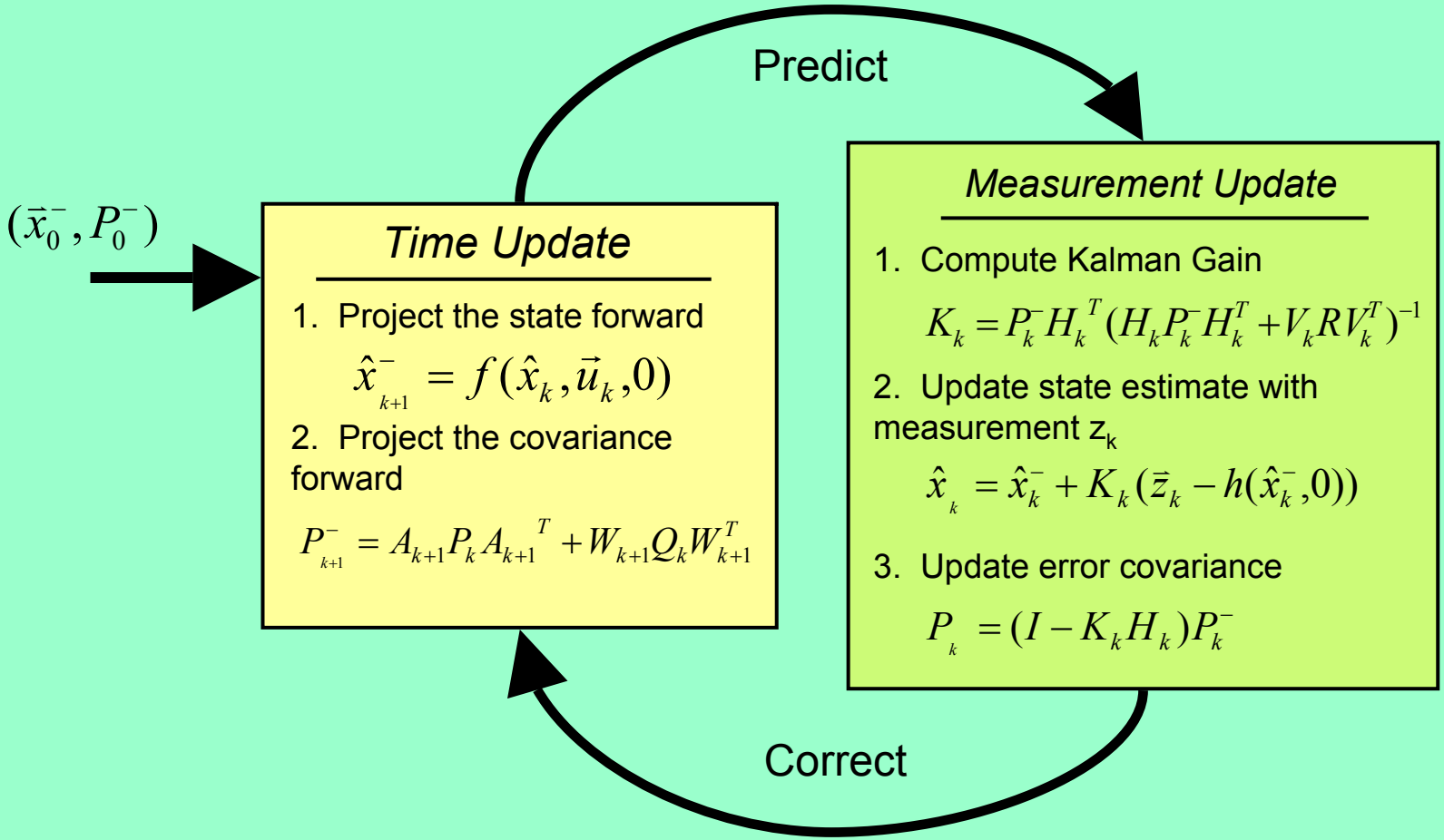
- **Covariance Update:**

$$P_k = (I - K_k H) P_k^-$$

$$P_k = (I - K_k H_k) P_k^-$$

NOTE: Some derivations will write this as Hx as well.

The Discrete Extended Kalman Filter



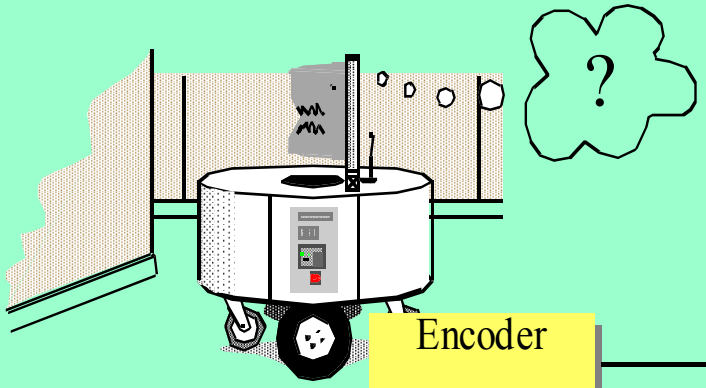
Robot Localisation & Mapping Example

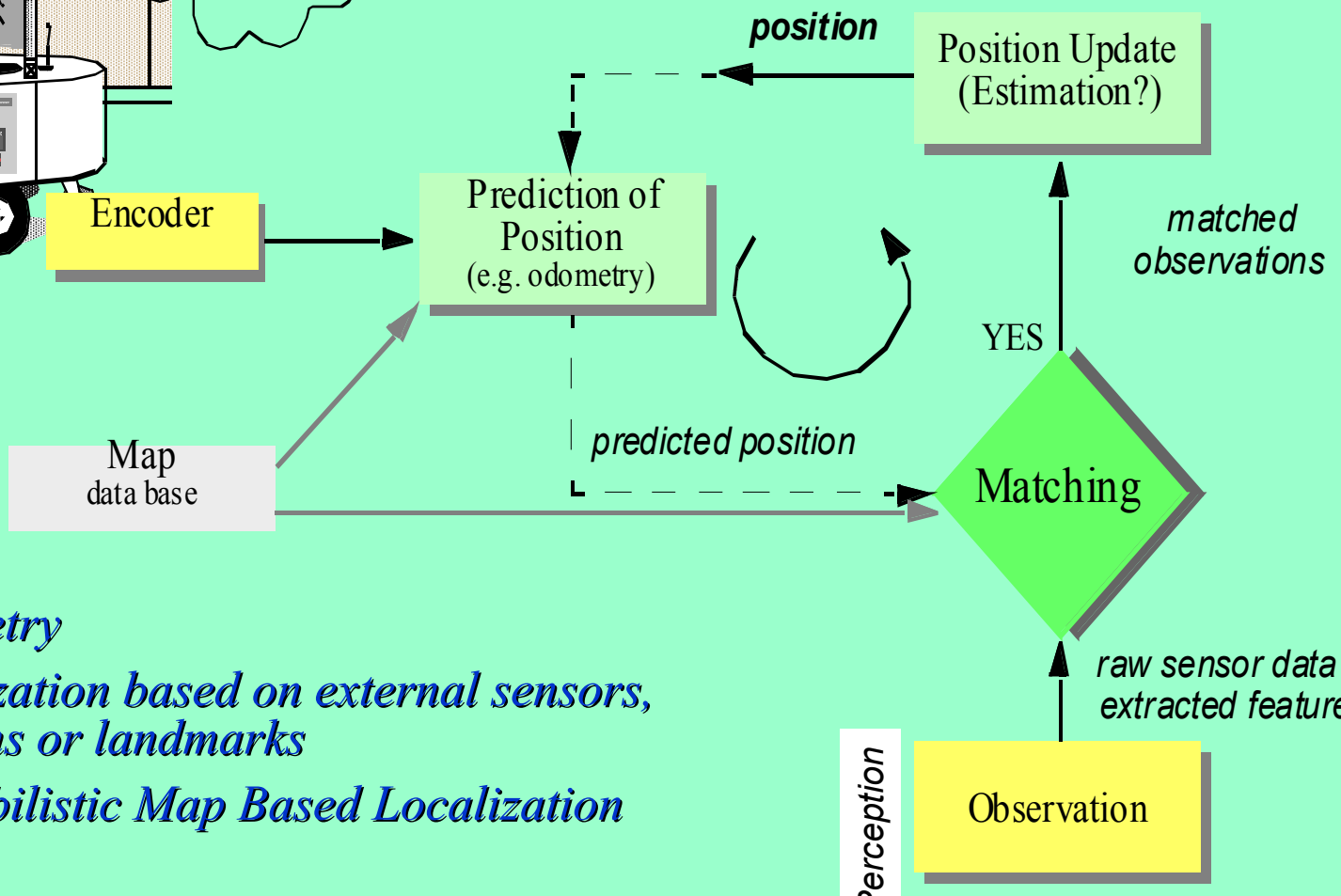
SLAM Overview

Fundamental problems to provide a mobile robot with autonomous capabilities:

- how to create an environmental map with imperfect sensors? ← mapping
- how a robot can tell where it is on a map? ← localization
- what if you're lost and don't have a map? ← robot SLAM

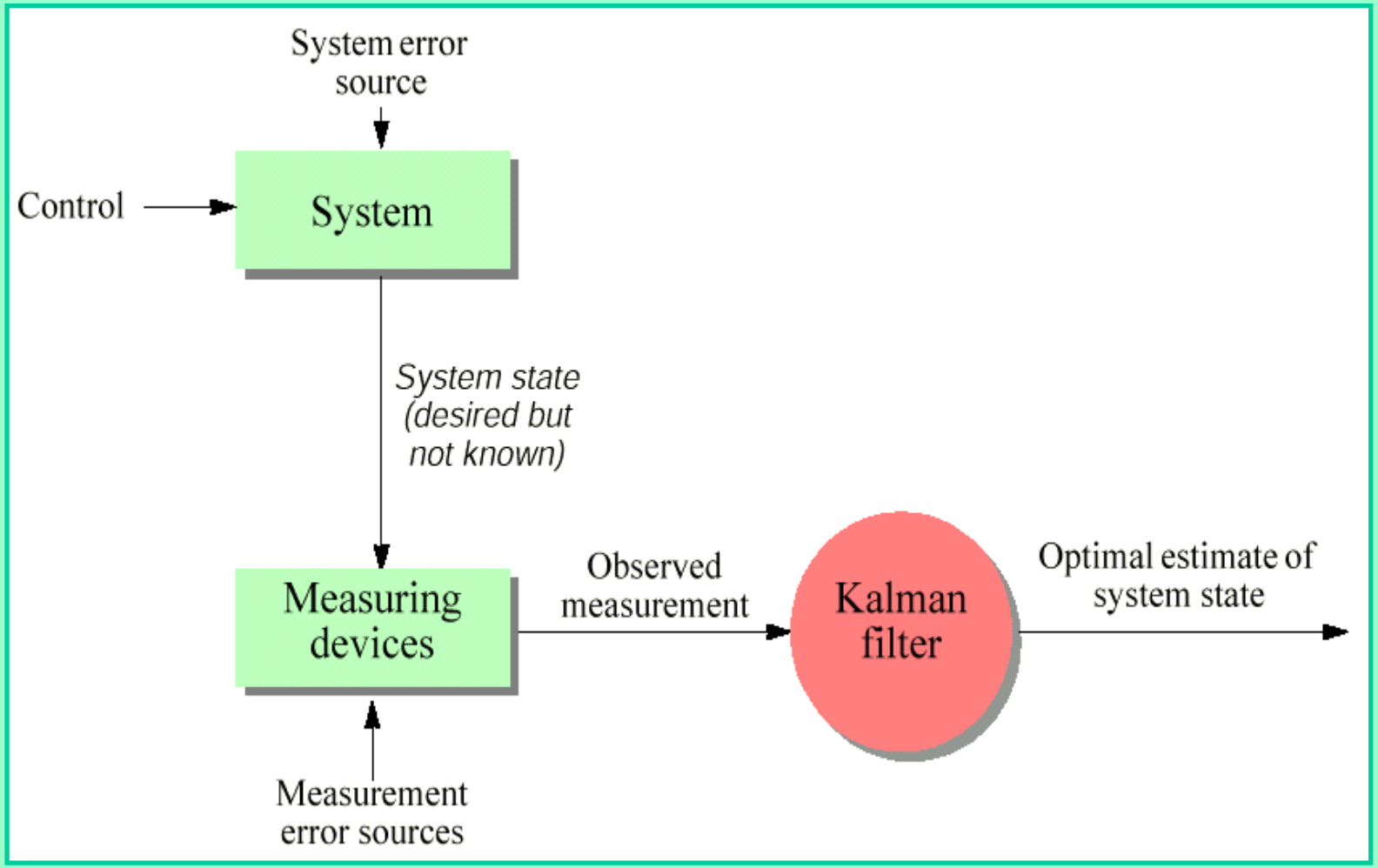
Localization, Where am I?



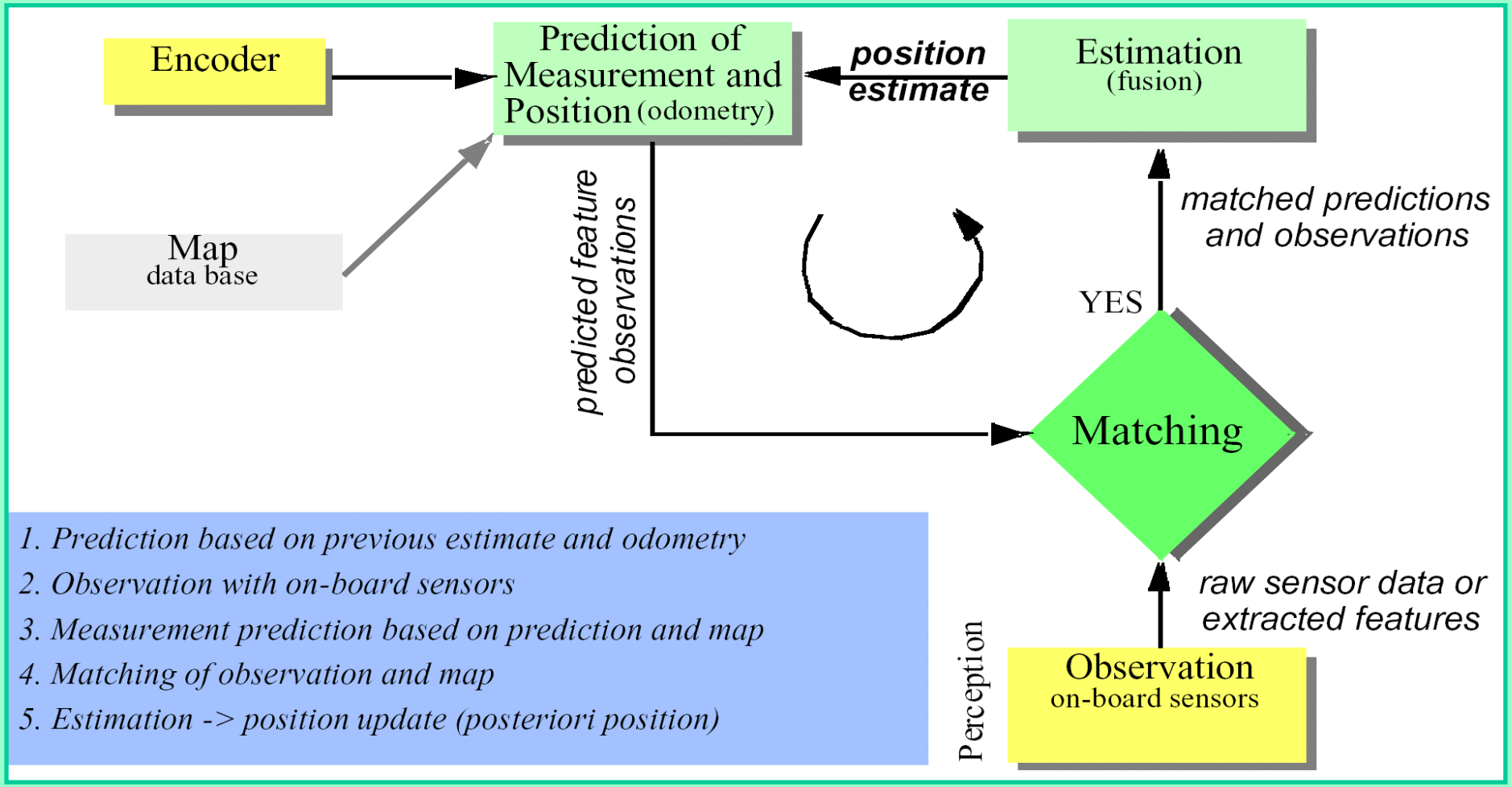


- *Odometry*
- *Localization based on external sensors, beacons or landmarks*
- *Probabilistic Map Based Localization*

Kalman Filter Localization



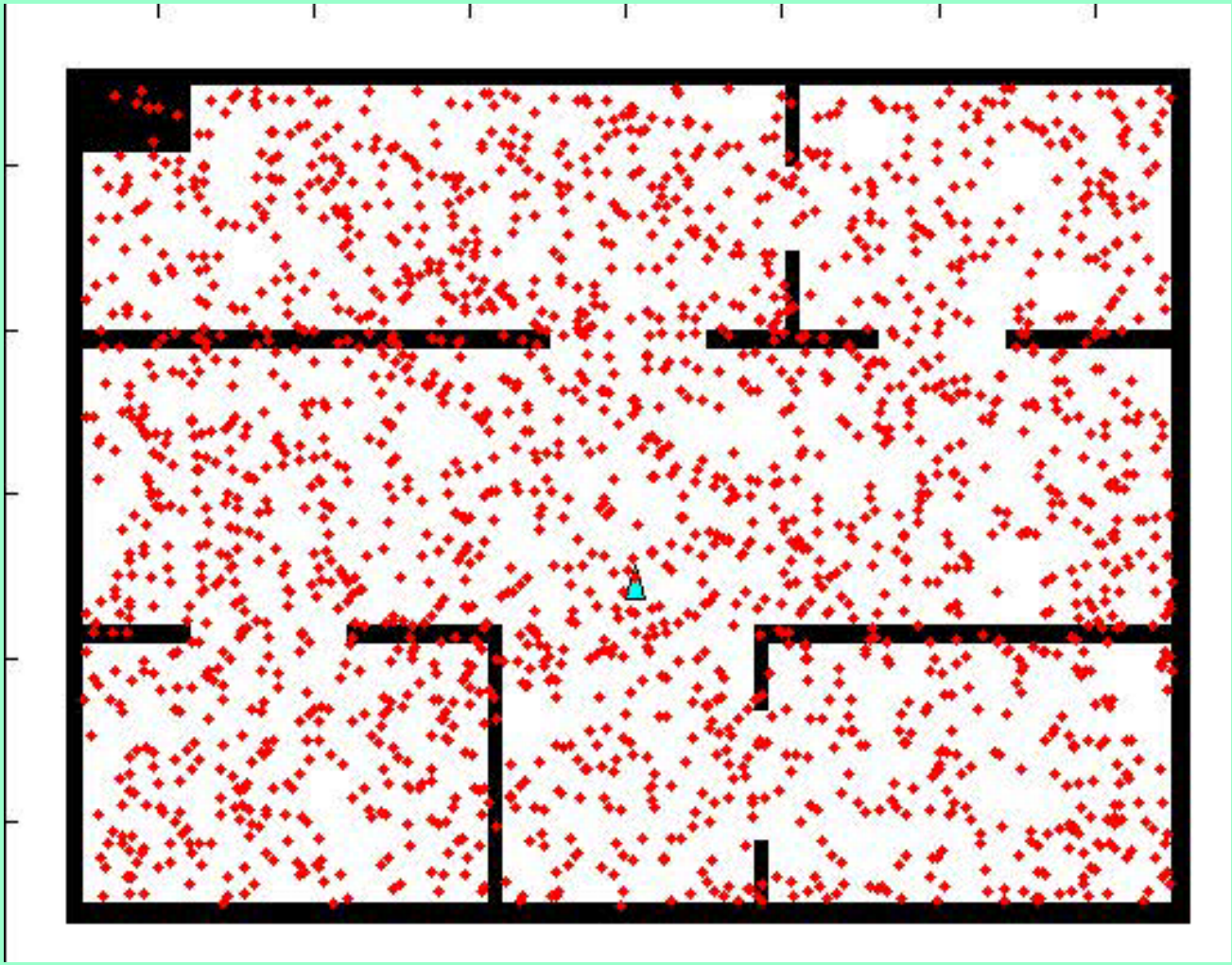
Kalman Filter for Mobile Robot Localization



Five steps: 1) Position Prediction, 2) Observation, 3) Measurement prediction, 4) Matching, 5) Estimation

Practical Application Examples (1)

Practical Application Examples (2)



Silvio Simani

Practical Application Examples (3)

Matlab®/Simulink® Software Examples

1. Estimation example of a constant signal with noise
   - Matlab script & Simulink file
2. Estimation of an acceleration signal
   - Simulink file
3. Estimation of velocity and acceleration signals
   - Matlab script file

References & Lecture Notes

Final Remarks & Conclusion

- http://www.cs.unc.edu/~welch/kalman
- ✓ Electronic and printed references
  - Book lists and recommendations
  - Research papers
  - Links to other sites
  - Some software
- http://www.ing.unife.it/simani/SDF_lesson.html
- ✓ Sensor Data Fusion
- ❖ Practical applications & software example