



上海科技大学
ShanghaiTech University

CS283: Robotics Spring 2025: SLAM II

Sören Schwertfeger / 师泽仁

ShanghaiTech University

OVERVIEW: THREE SLAM PARADIGMS

The Three SLAM Paradigms

- Most of the SLAM algorithms are based on the following three different approaches:
 - Extended Kalman Filter SLAM: (called EKF SLAM)
 - Particle Filter SLAM: (called FAST SLAM)
 - Graph-Based SLAM

EKF SLAM: overview

- **Extended state vector** y_t : robot pose x_t + position of all the features m_i in the map:

$$y_t = [x_t, m_0, \dots, m_{n-1}]^T$$

- Example: 2D line-landmarks, size of $y_t = 3 + 2n$: three variables to represent the robot pose + $2n$ variables for the n line-landmarks having vector components

$$(\alpha_i, r_i)$$

$$y_t = [x_t, y_t, \theta_t, \alpha_0, r_0, \dots, \alpha_{n-1}, r_{n-1}]^T$$

- As the robot moves and takes measurements, the state vector and covariance matrix are updated using the standard equations of the extended Kalman filter
- Drawback: EKF SLAM is computationally very expensive.

Particle Filter SLAM: FastSLAM

- **FastSLAM approach**

- Using particle filters.
- Particle filters: mathematical models that represent probability distribution as a set of discrete particles that occupy the state space.

- **Particle filter update**

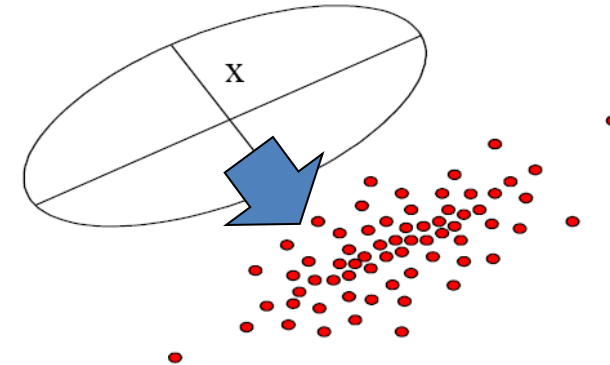
- Generate new particle distribution using motion model and controls

- a) For each particle:

1. Compare particle's prediction of measurements with actual measurements
2. Particles whose predictions match the measurements are given a high weight

- b) Filter resample:

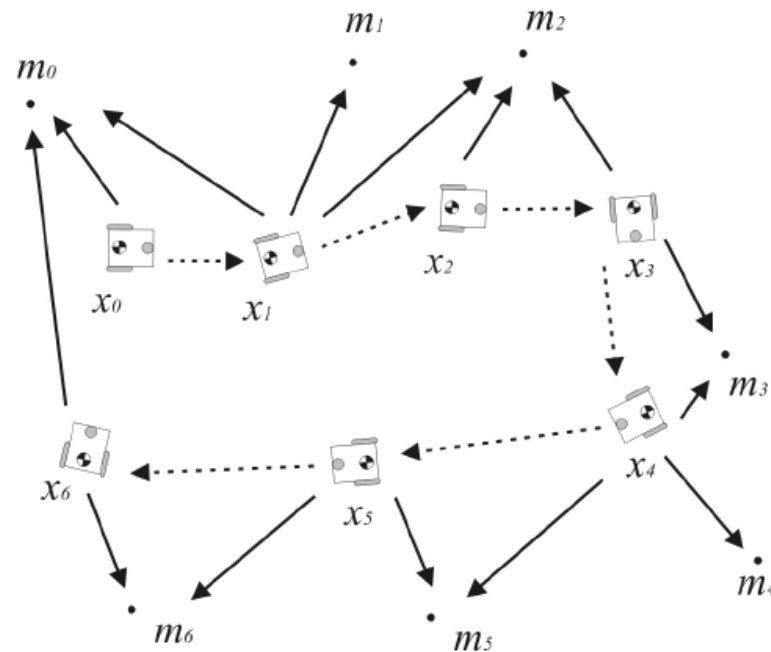
- Resample particles based on weight
- Filter resample
 - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.



probability distribution (ellipse) as particle set (red dots)

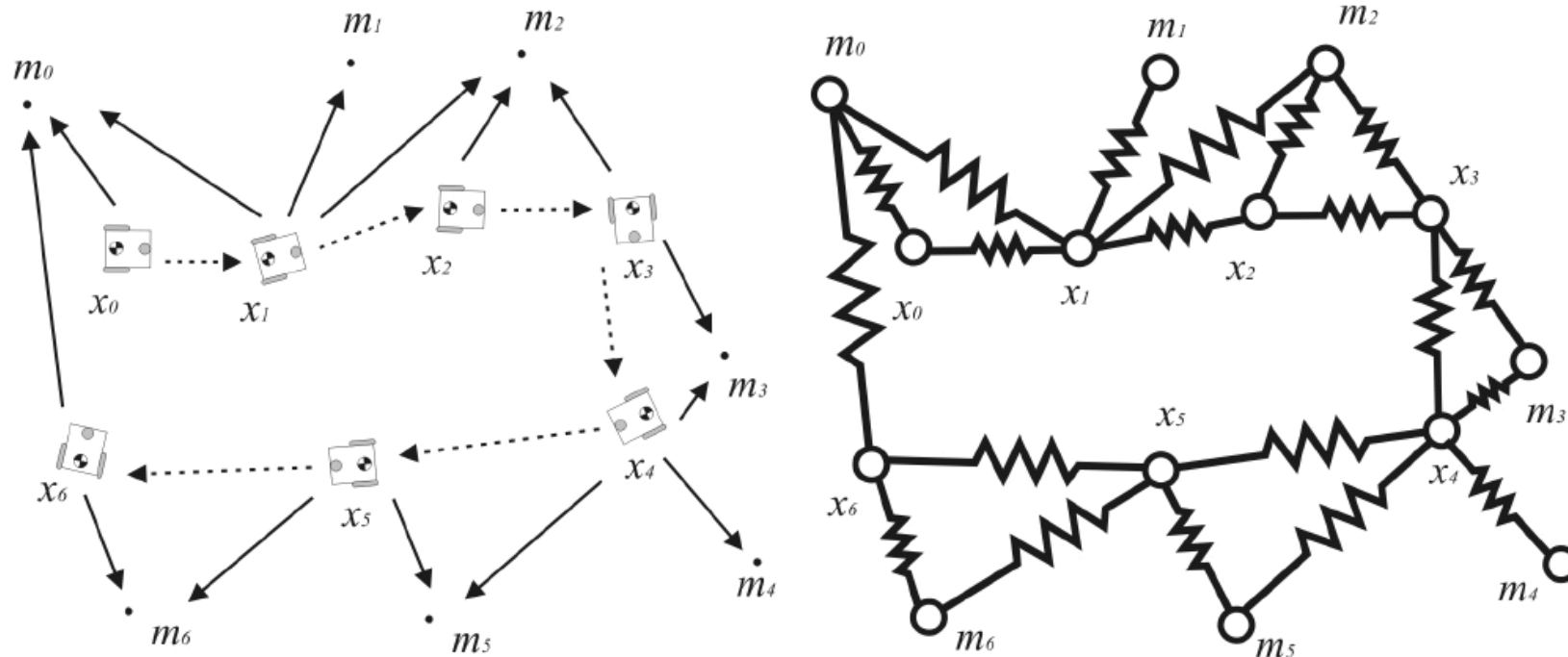
Graph-Based SLAM (1/3)

- SLAM problem can be interpreted as a sparse graph of nodes and constraints between nodes.
- The nodes of the graph are the robot locations and the features in the map.
- Constraints: relative position between consecutive robot poses, (given by the odometry input \mathbf{u}) and the relative position between the robot locations and the features observed from those locations.



Graph-Based SLAM (2/3)

- Constraints are not rigid but soft constraints!
- Relaxation: compute the solution to the full SLAM problem =>
 - Compute best estimate of the robot path and the environment map.
 - Graph-based SLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net



Graph-Based SLAM (3/3)

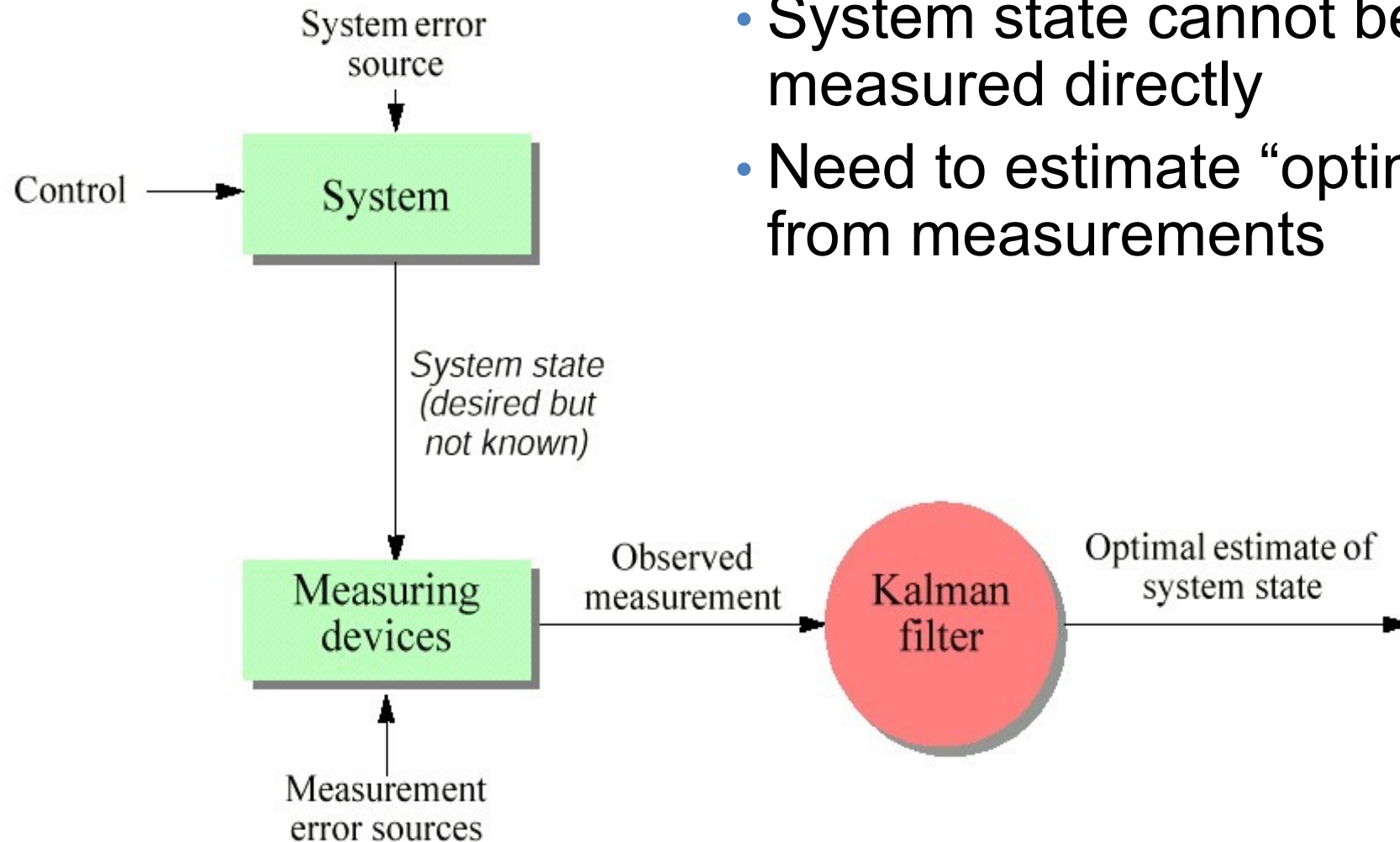
- Significant advantage of graph-based SLAM techniques over EKF SLAM:
 - EKF SLAM: computation and memory for to update and store the covariance matrix is quadratic with the number of features.
 - Graph-based SLAM: update time of the graph is constant and the required memory is linear in the number of features.
- However, the final graph optimization can become computationally costly if the robot path is long.
- Libraries for graph-based slam: g2o, ceres

KALMAN FILTER OVERVIEW

Following Material:

- Michael Williams, Australian National University
- Cornelia Fermüller, University of Maryland

The Problem

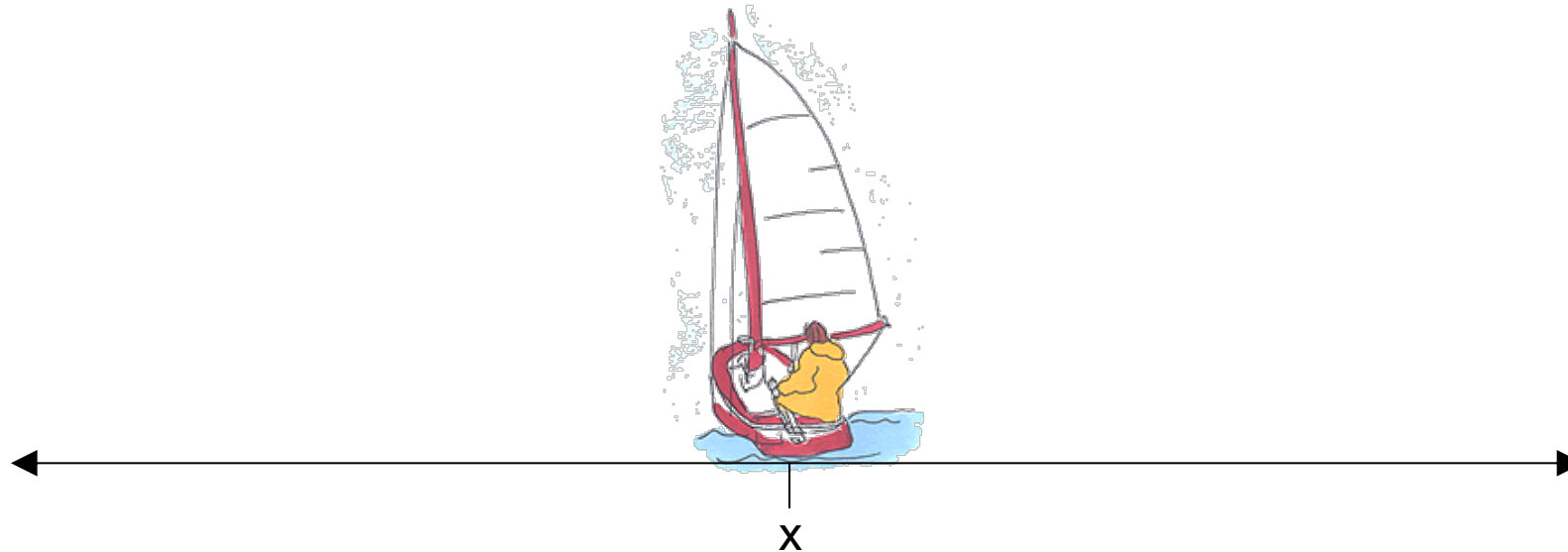


- System state cannot be measured directly
- Need to estimate “optimally” from measurements

What is a Kalman Filter?

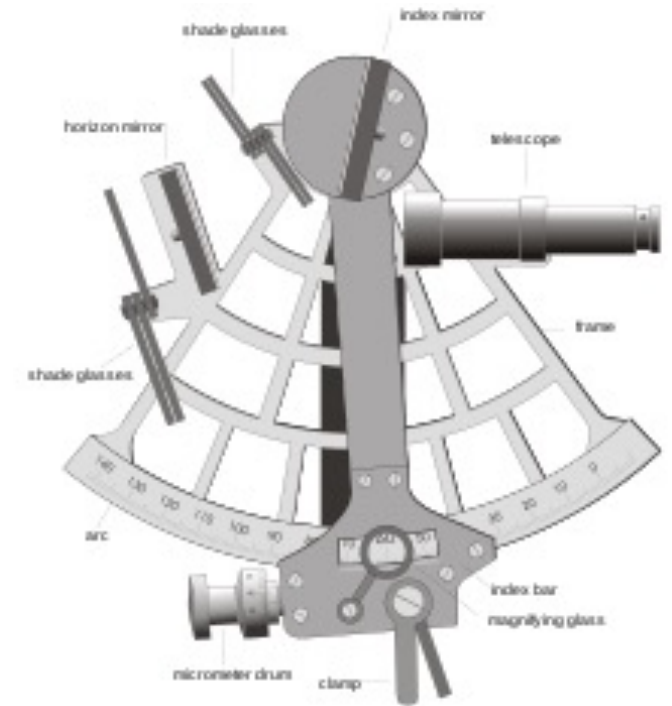
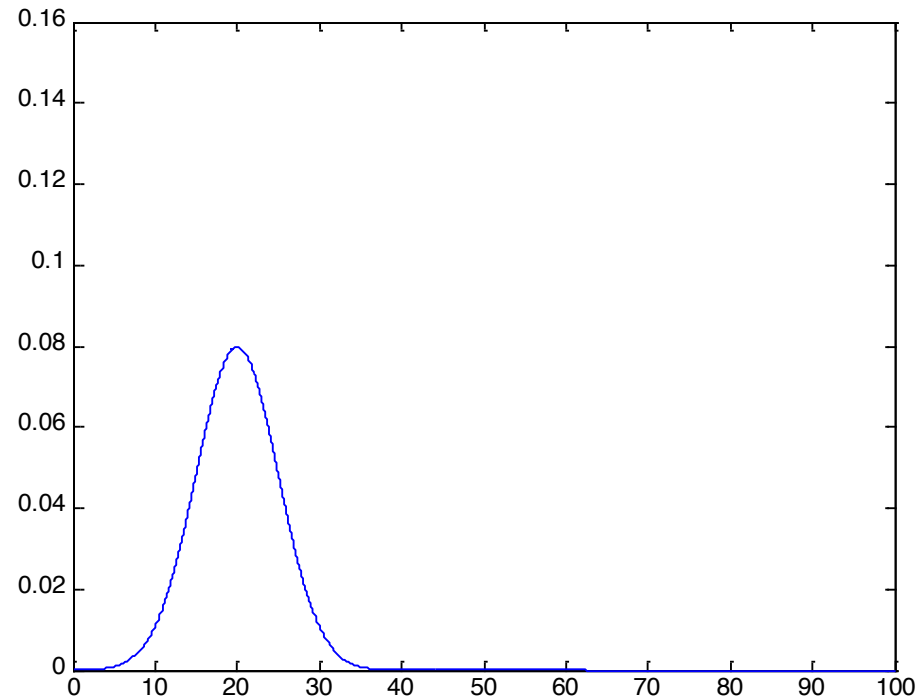
- Recursive data processing algorithm
- Generates optimal estimate of desired quantities given the set of measurements
- Optimal?
 - For linear system and white Gaussian errors, Kalman filter is “best” estimate based on all previous measurements
 - For non-linear system optimality is ‘qualified’
- Recursive?
 - Doesn't need to store all previous measurements and reprocess all data each time step

Conceptual Overview



- Lost on the 1-dimensional line
- Position – $x(t)$
- Assume Gaussian distributed measurements

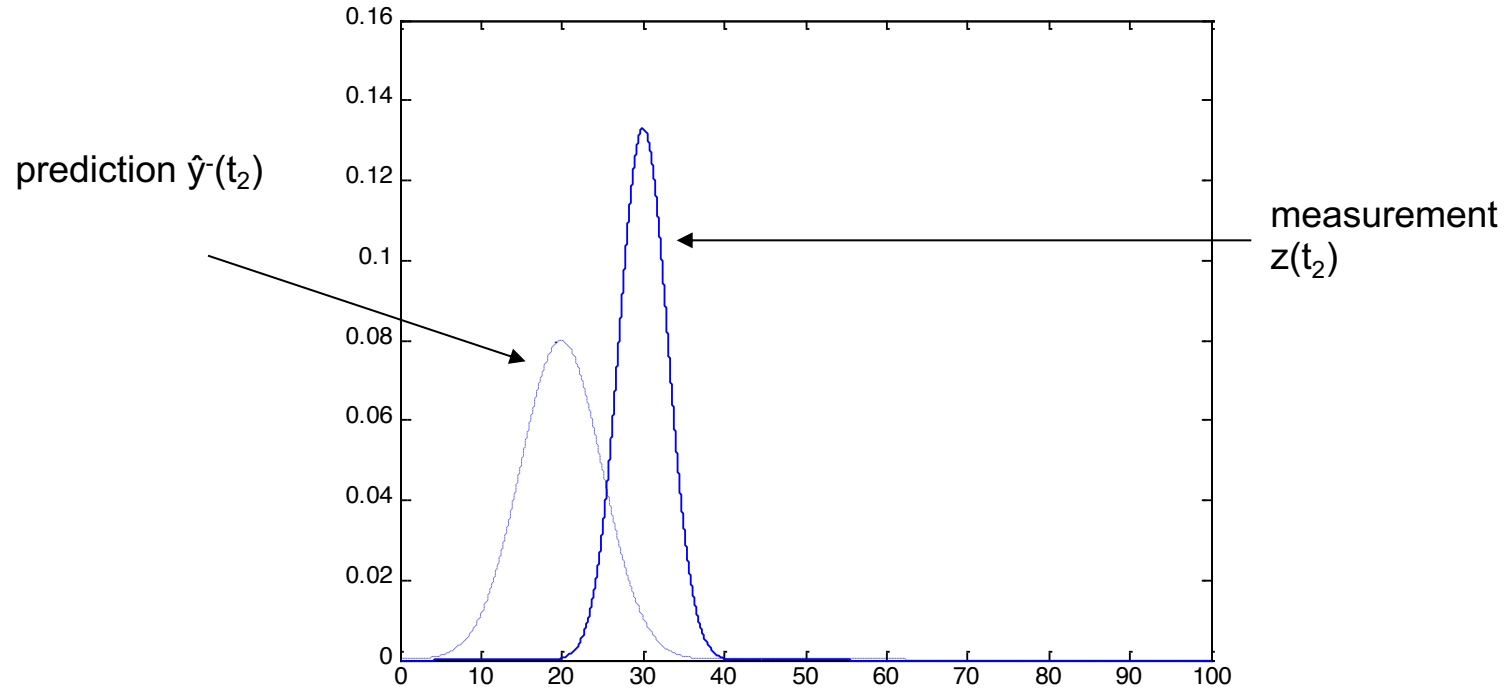
Conceptual Overview



Sextant

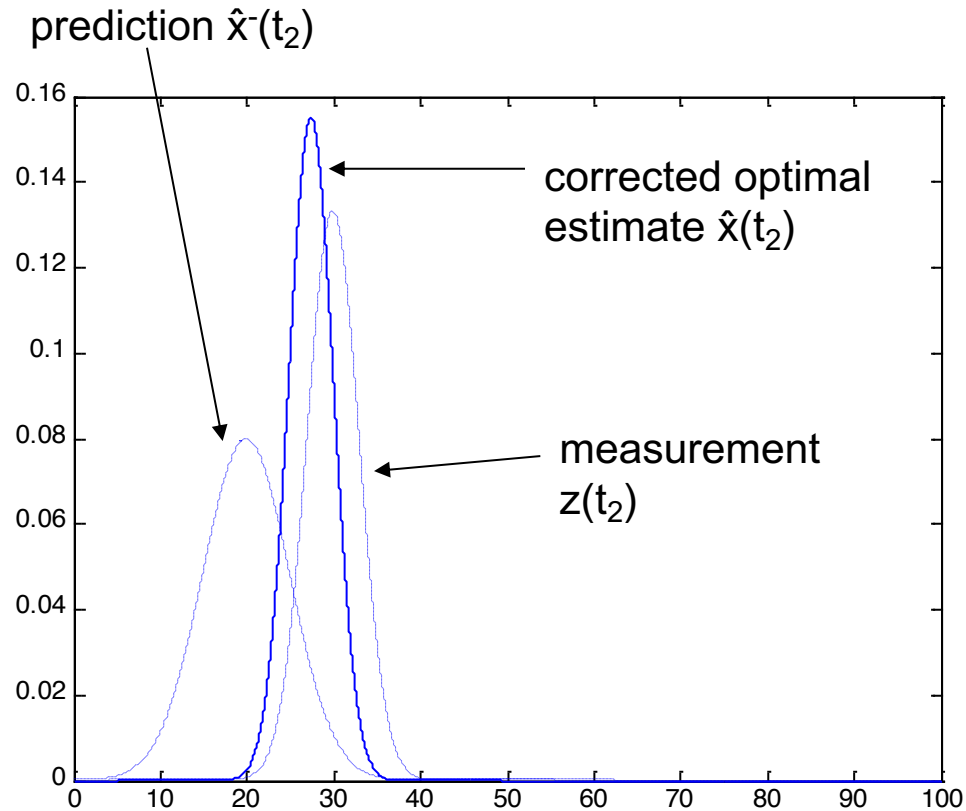
- Sextant Measurement at t_1 : Mean = z_1 and Variance = σ_{z_1}
- Optimal estimate of position is: $\hat{x}(t_1) = z_1$
- Variance of error in estimate: $\sigma_x^2(t_1) = \sigma_{z_1}^2$
- Boat in same position at time t_2 - Predicted position is z_1

Conceptual Overview



- So we have the prediction $\hat{x}^-(t_2)$
- GPS Measurement at t_2 : Mean = z_2 and Variance = σ_{z_2}
- Need to correct the prediction due to measurement to get $\hat{x}(t_2)$
- Closer to more trusted measurement – linear interpolation?

Conceptual Overview



- Corrected mean is the new optimal estimate of position
- New variance is smaller than either of the previous two variances

Conceptual Overview

- Lessons so far:

Make prediction based on previous data: \hat{x}^-, σ^-



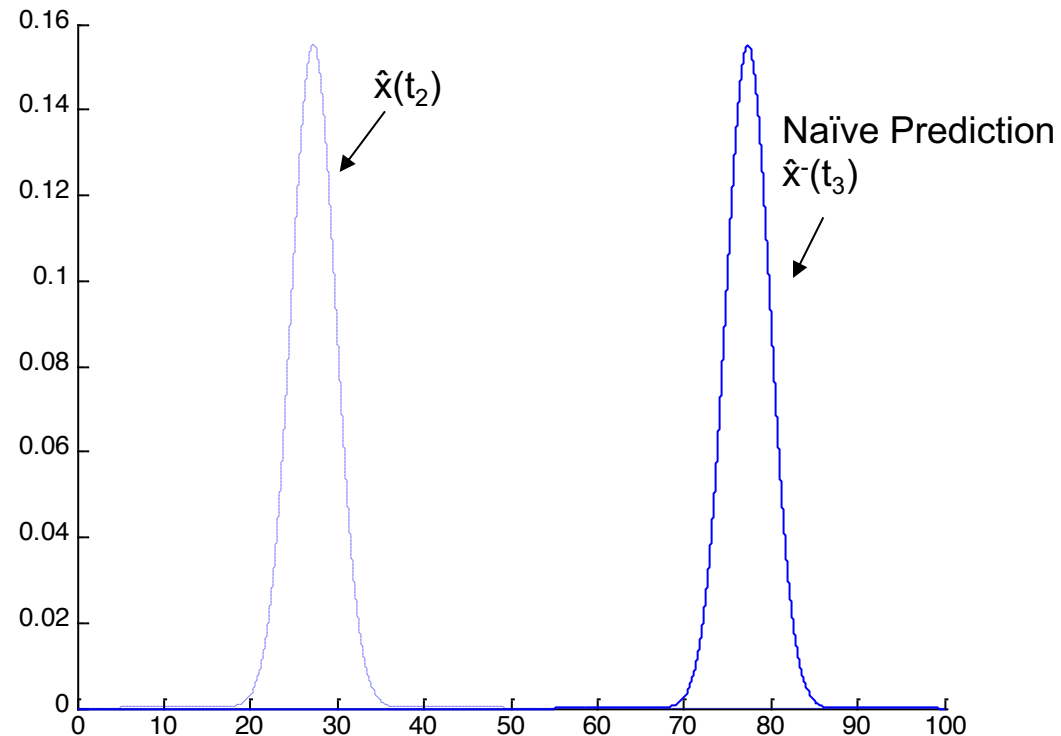
Take measurement: z_k, σ_z



Optimal estimate (\hat{x}) = Prediction + (Kalman Gain) * (Measurement - Prediction)

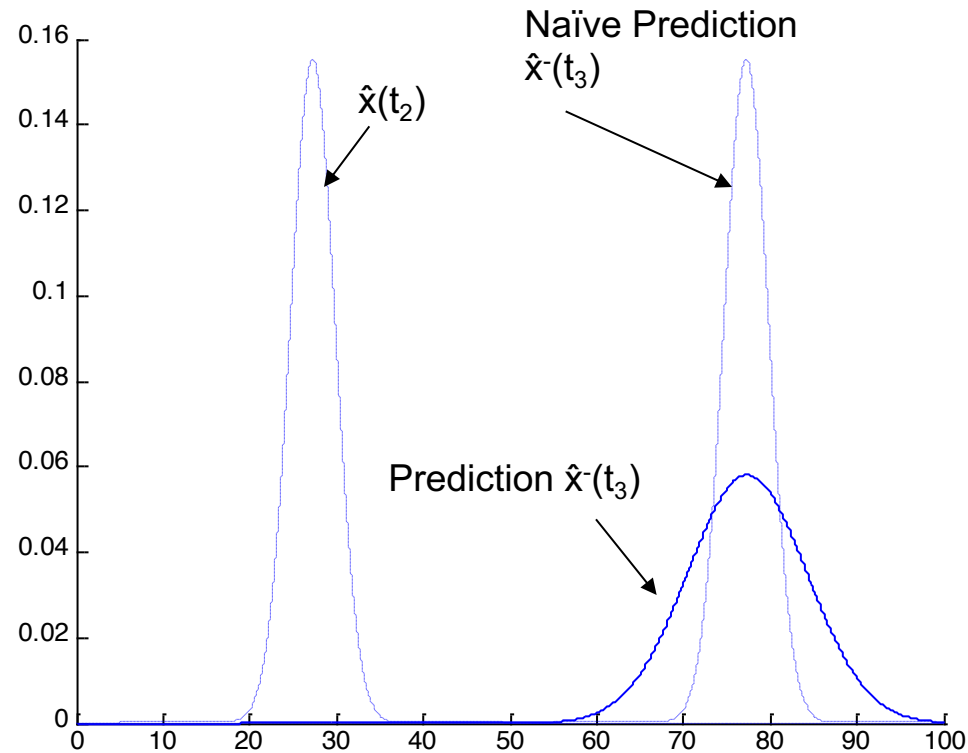
Variance of estimate = Variance of prediction * (1 - Kalman Gain)

Conceptual Overview



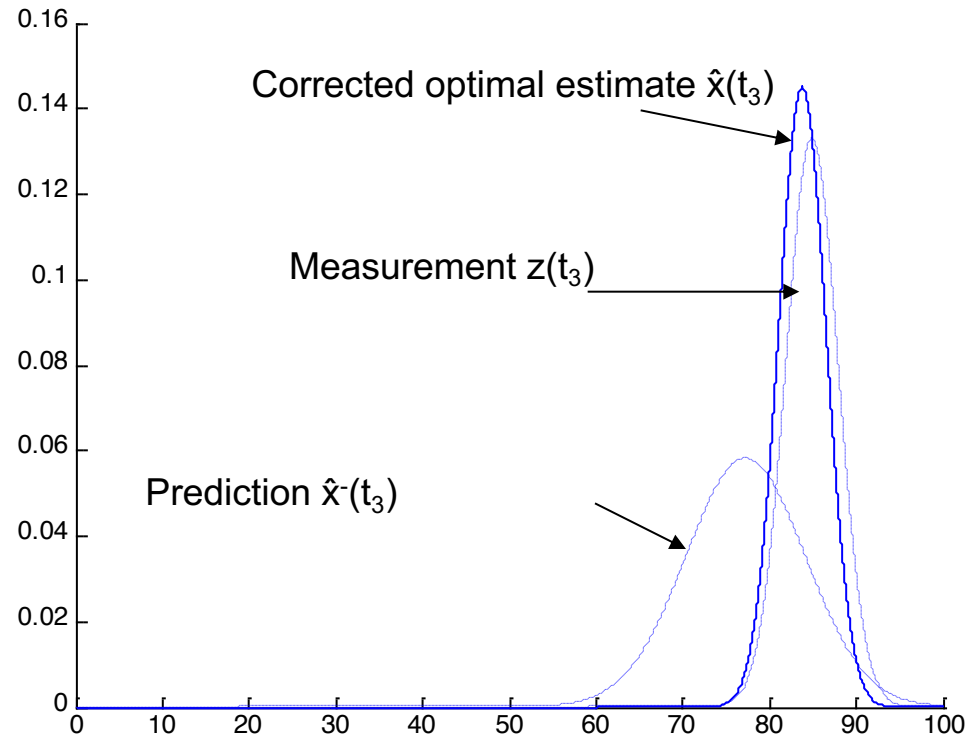
- At time t_3 , boat moves with velocity $dx/dt=u$
- Naïve approach: Shift probability to the right to predict
- This would work if we knew the velocity exactly (perfect model)

Conceptual Overview



- Better to assume imperfect model by adding Gaussian noise
- $dx/dt = u + w$
- Distribution for prediction moves and spreads out

Conceptual Overview



- Now we take a measurement at t_3
- Need to once again correct the prediction
- Same as before

Conceptual Overview

- Lessons learnt from conceptual overview:
 - Initial conditions (\hat{x}_{k-1} and σ_{k-1})
 - Prediction (\hat{x}_k^- , σ_k^-)
 - Use initial conditions and model (eg. constant velocity) to make prediction
 - Measurement (z_k)
 - Take measurement
 - Correction (\hat{x}_k , σ_k)
 - Use measurement to correct prediction by ‘blending’ prediction and residual – always a case of merging only two Gaussians
 - Optimal estimate with smaller variance

Theoretical Basis

- Process to be estimated:

$$\hat{x}_k = Ax_{k-1} + Bu_k + w_{k-1} \quad \text{Process Noise (w) with covariance Q}$$

$$z_k = Hx_k + v_k \quad \text{Measurement Noise (v) with covariance R}$$

- Kalman Filter

Predicted: \hat{y}_k^- is estimate based on measurements at previous time-steps

$$\hat{x}_k^- = Ax_{k-1} + Bu_k$$

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

Corrected: \hat{y}_k has additional information – the measurement at time k

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H \hat{x}_k^-)$$

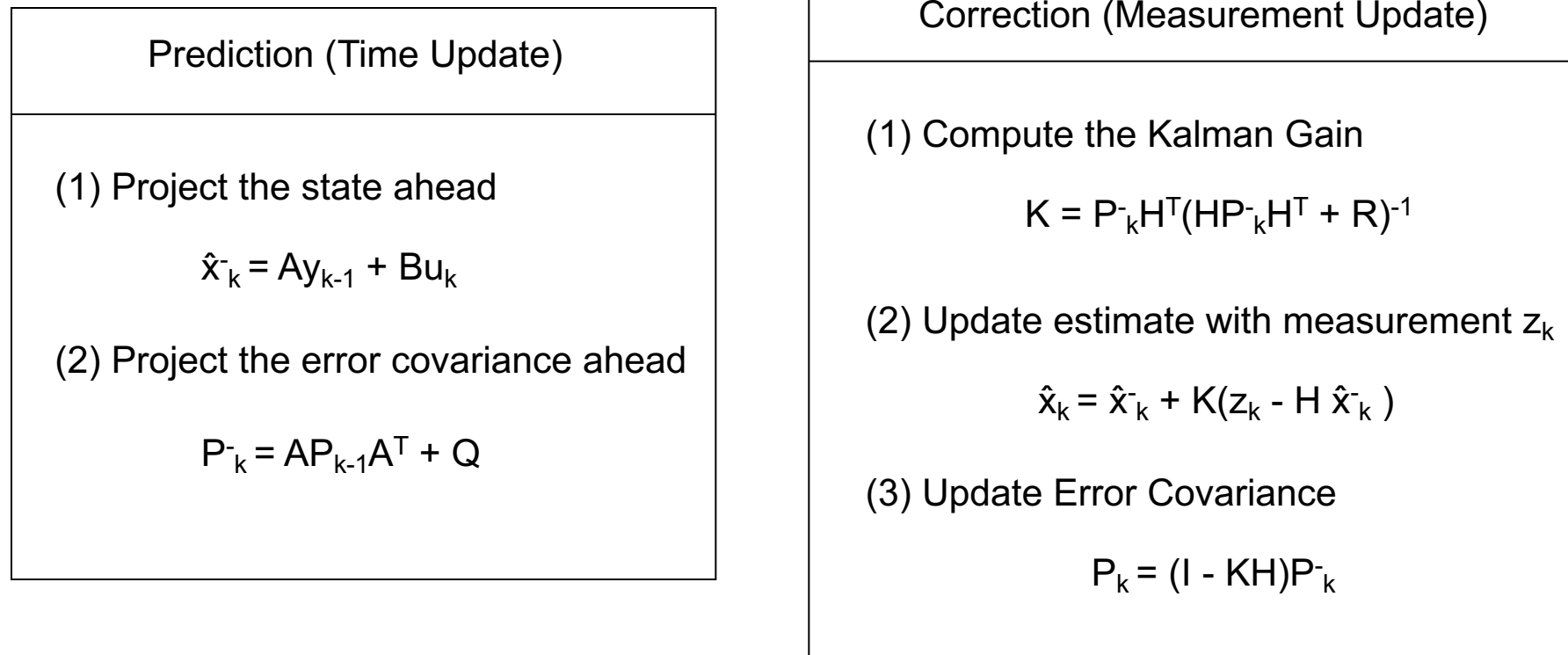
$$K = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

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

Blending Factor

- If we are sure about measurements:
 - Measurement error covariance (R) decreases to zero
 - K decreases and weights residual more heavily than prediction
- If we are sure about prediction
 - Prediction error covariance P_k^- decreases to zero
 - K increases and weights prediction more heavily than residual

Theoretical Basis



RECURSIVE STATE ESTIMATION

Following Material:

- Cyrill Stachniss, University of Bonn

State Estimation

- Estimate the state x of a system given observations z and controls u
- **Goal:**

$$p(x \mid z, u)$$

State Estimation

- Estimate the state x of a system given observations z and controls u
- **Goal:**

$$p(x_t | z_{1:t}, u_{1:t})$$

(reminder)

Tiny Reminder (Probability Theory)

(reminder)

Joint probability

$p(x, y)$ probability that x and y happen together

Conditional probability

$p(y \mid x)$ probability that x happens, given that y already happened

(reminder)

Bayes' Rule

$$p(x, y) = p(x | y) p(y)$$

$$p(x, y) = p(y | x) p(x)$$



$$p(x | y) = \frac{p(y | x) p(x)}{p(y)} = \frac{\text{likelihood prior}}{\text{evidence}}$$

Bayes' Rule with Background Knowledge z

$$p(x | y) = \frac{p(y | x) p(x)}{p(y)}$$



$$p(x | y, z) = \frac{p(y | x, z) p(x | z)}{p(y | z)}$$

(reminder)

Law of Total Probability and Marginalization

Law of Total Probability

$$p(x) = \sum_y p(x | y) p(y) \quad p(x) = \int p(x | y) p(y) dy$$

Marginalization

$$p(x) = \sum_y p(x, y) \quad p(x) = \int p(x, y) dy$$

Markov Property/ Assumption

- “The future is independent from the past - given the **current** state.”
- Markov property = the conditional **probability distribution** of future states depends only upon the present state, not on the sequence of events that preceded it.
- Such a process has no memory

State Estimation

- Estimate the state x of a system given observations z and controls u
- **Goal:**

$$p(x_t | z_{1:t}, u_{1:t})$$

- This gives the probability distribution of the robot state, with the mean of the distribution being the most likely (robot) state x_t and the uncertainty is the covariance of the distribution, describing the spread of the distribution and thus the confidence in the estimate.

Recursive Bayes Filter 1

$$bel(x_t) = \underline{p(x_t \mid z_{1:t}, u_{1:t})}$$

definition of the belief

$bel(x_t)$ Belief of (x_t): the probability distribution

Recursive Bayes Filter 2

$$\begin{aligned} \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\ &= \eta \underbrace{p(z_t \mid x_t, z_{1:t-1}, u_{1:t})}_{\text{Bayes' rule}} p(x_t \mid z_{1:t-1}, u_{1:t}) \end{aligned}$$

Bayes' rule

Recursive Bayes Filter 3

$$\begin{aligned} \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\ &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta \underline{p(z_t \mid x_t)} p(x_t \mid z_{1:t-1}, u_{1:t}) \end{aligned}$$

Markov assumption

Recursive Bayes Filter 4

$$\begin{aligned} \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\ &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) \int \underbrace{p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t})}_{\underbrace{p(x_{t-1} \mid z_{1:t-1}, u_{1:t})} dx_{t-1}} \end{aligned}$$

Law of total probability

Recursive Bayes Filter 5

$$\begin{aligned} \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\ &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\ &\quad p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\ &= \eta p(z_t \mid x_t) \int \underline{p(x_t \mid x_{t-1}, u_t)} p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \end{aligned}$$

Markov assumption

Recursive Bayes Filter 6

$$\begin{aligned}
 \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
 &\quad p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, \underline{u_{1:t-1}}) dx_{t-1}
 \end{aligned}$$

independence assumption

Recursive Bayes Filter 7

$$\begin{aligned}
 \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\
 &\quad p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) dx_{t-1} \\
 &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) \underline{\text{bel}(x_{t-1})} dx_{t-1}
 \end{aligned}$$

recursive term

Complete Derivation of the Recursive Bayes Filter

$$\begin{aligned} \text{bel}(x_t) &= p(x_t \mid z_{1:t}, u_{1:t}) \\ &= \eta p(z_t \mid x_t, z_{1:t-1}, u_{1:t}) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) p(x_t \mid z_{1:t-1}, u_{1:t}) \\ &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, z_{1:t-1}, u_{1:t}) \\ &\quad p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\ &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t}) dx_{t-1} \\ &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) p(x_{t-1} \mid z_{1:t-1}, u_{1:t-1}) dx_{t-1} \\ &= \eta p(z_t \mid x_t) \int p(x_t \mid x_{t-1}, u_t) \text{bel}(x_{t-1}) dx_{t-1} \end{aligned}$$

Prediction and Correction Step

- Bayes filter can be written as a two step process

$$\underline{bel(x_t) = \eta p(z_t | x_t) \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}}$$

- **Prediction step**

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

- **Correction step**

$$bel(x_t) = \eta p(z_t | x_t) \overline{bel}(x_t)$$

Motion and Observation Model

- Prediction step

$$\overline{bel}(x_t) = \int \underline{p(x_t | u_t, x_{t-1})} \overline{bel}(x_{t-1}) dx_{t-1}$$

motion model

- Correction step

$$bel(x_t) = \eta \underline{p(z_t | x_t)} \overline{bel}(x_t)$$

observation model

(also: measurement or sensor model)

Different Realizations

- The Bayes filter is a **framework** for recursive state estimation
- There are **different realizations**
- **Different properties**
 - Linear vs. non-linear models for motion and observation models
 - Gaussian distributions only?
 - Parametric vs. non-parametric filters
 - ...

Popular Filters

- **Kalman filter & EKF**
 - Gaussians
 - Linear or linearized models
- **Particle filter**
 - Non-parametric
 - Arbitrary models (sampling required)

KALMAN FILTER DETAILS

Following Material:

- Michael Williams, Australian National University
- Cornelia Fermüller, University of Maryland

Recursive Bayes Filter

$$Bel(x_t) = \eta P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

1. Algorithm **Bayes_filter**($Bel(x), d$):
2. $\eta = 0$
3. If d is a **perceptual** data item z then
4. For all x do
5. $Bel'(x) = P(z | x) Bel(x)$
6. $\eta = \eta + Bel'(x)$
7. For all x do
8. $Bel(x) = \eta^{-1} Bel'(x)$
9. Else if d is an **action** data item u then
10. For all x do
11. $Bel'(x) = \int P(x | u, x') Bel(x') dx'$
12. Return $Bel(x)$

Kalman Filter

- Bayes filter with **Gaussians**
- Developed in the late 1950's
- Most relevant Bayes filter variant in practice
- Applications range from economics, weather forecasting, satellite navigation to robotics and many more.
- The Kalman filter "algorithm" is a couple of **matrix multiplications!**

Gaussians

$$p(x) \sim N(\mu, \sigma^2):$$

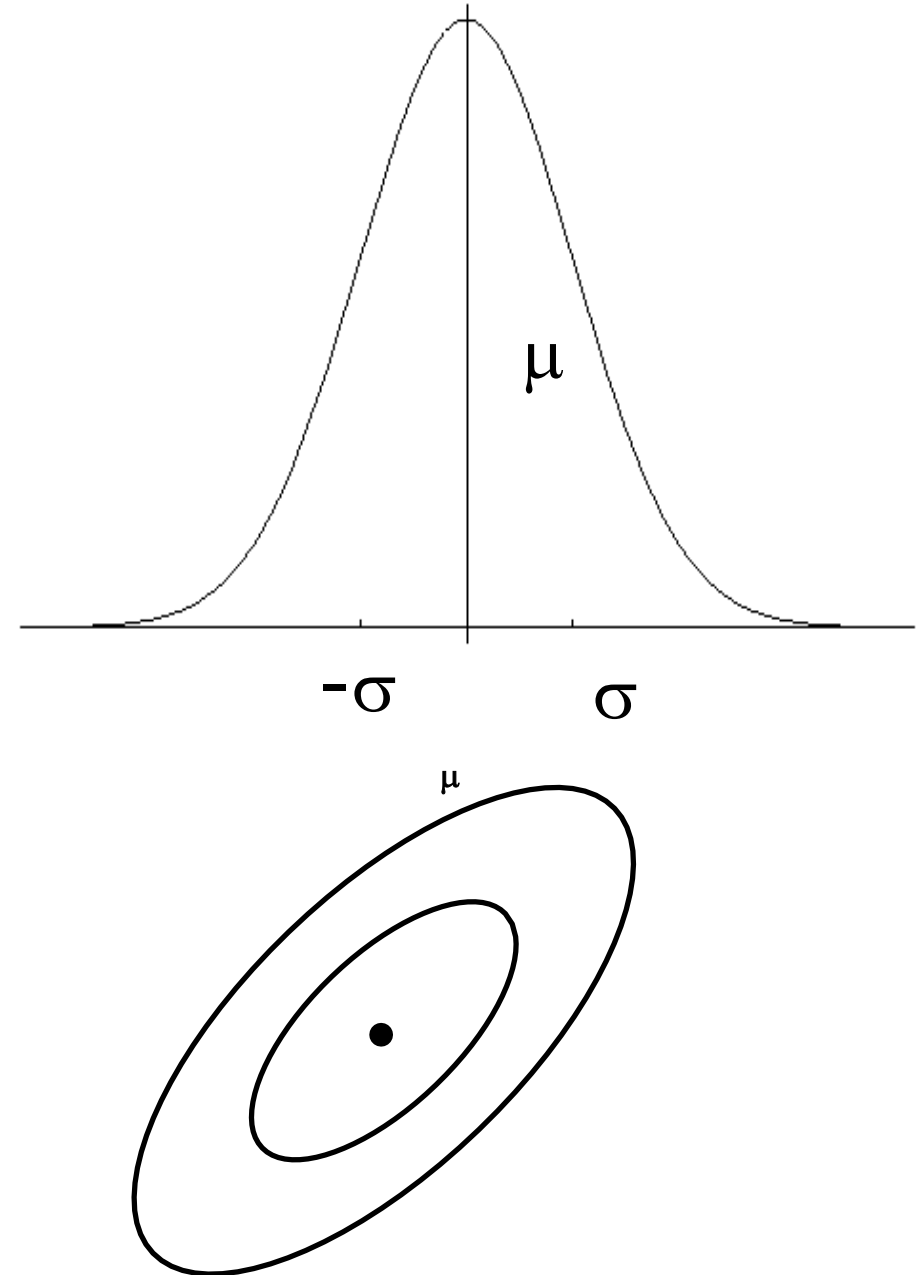
$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

Univariate

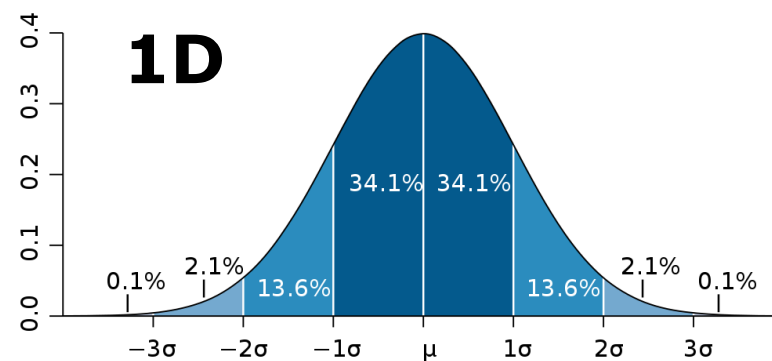
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

Multivariate



Gaussians



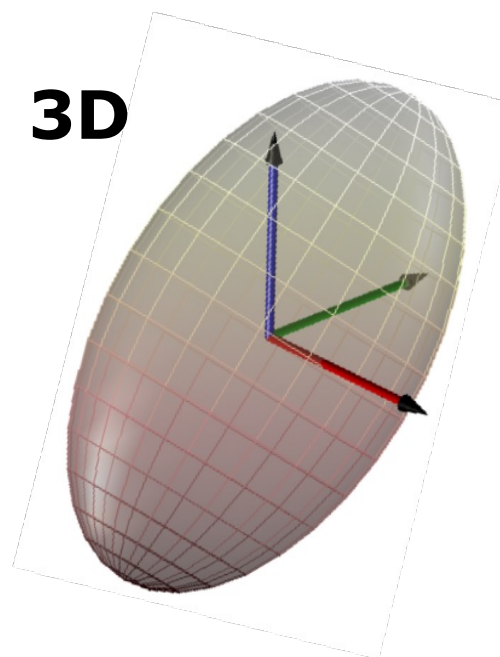
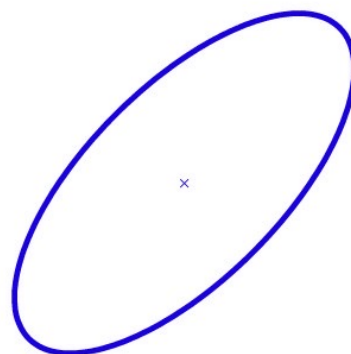
2D

$$C = \begin{bmatrix} 0.020 & 0.013 \\ 0.013 & 0.020 \end{bmatrix}$$

$$\lambda_1 = 0.007$$

$$\lambda_2 = 0.033$$

$$\rho = \sigma_{XY} / \sigma_X \sigma_Y = 0.673$$



Properties of Gaussians

- Univariate

$$\left. \begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array} \right\} \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

$$\left. \begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array} \right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \mu_2, \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}}\right)$$

- Multivariate

$$\left. \begin{array}{l} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array} \right\} \Rightarrow Y \sim N(A\mu + B, A\Sigma A^T)$$

$$\left. \begin{array}{l} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array} \right\} \Rightarrow p(X_1) \cdot p(X_2) \sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2} \mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2} \mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)$$

- We stay in the “Gaussian world” as long as we start with Gaussians and perform only linear transformations

Introduction to Kalman Filter

- Two measurements no dynamics

$$\hat{q}_1 = q_1 \text{ with variance } \sigma_1^2$$

$$\hat{q}_2 = q_2 \text{ with variance } \sigma_2^2$$

- Weighted least-square

$$S = \sum_{i=1}^n w_i (\hat{q} - q_i)^2$$

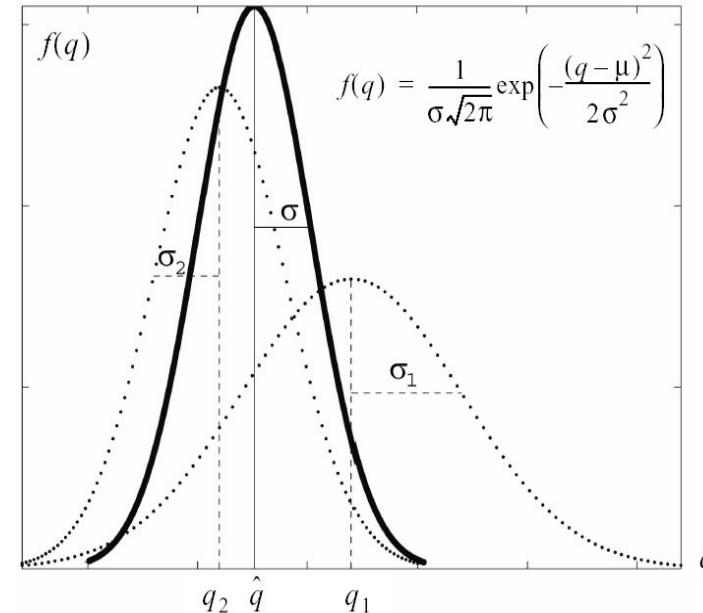
- Finding minimum error

$$\frac{\partial S}{\partial \hat{q}} = \frac{\partial}{\partial \hat{q}} \sum_{i=1}^n w_i (\hat{q} - q_i)^2 = 2 \sum_{i=1}^n w_i (\hat{q} - q_i) = 0$$

- After some calculation and rearrangements

$$\hat{q} = q_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} (q_2 - q_1)$$

- Another way to look at it – weighted mean



Discrete Kalman Filter

- Estimates the state x of a discrete-time controlled process that is governed by the linear stochastic difference equation

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \quad \leftarrow \text{Process dynamics}$$

- with a measurement

$$z_t = C_t x_t + \delta_t \quad \leftarrow \text{Observation model}$$

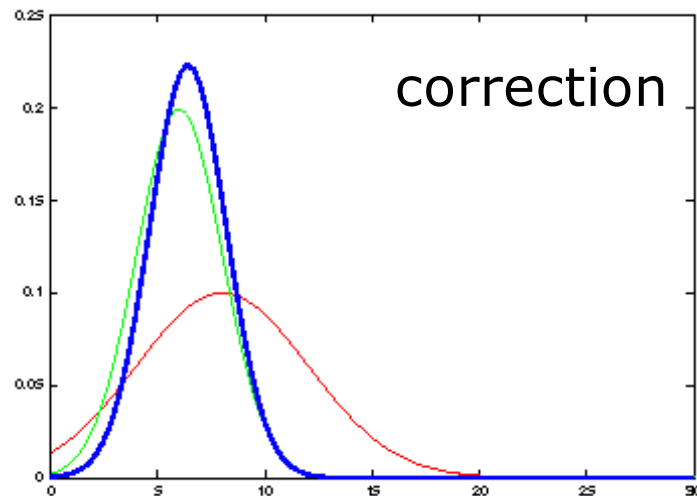
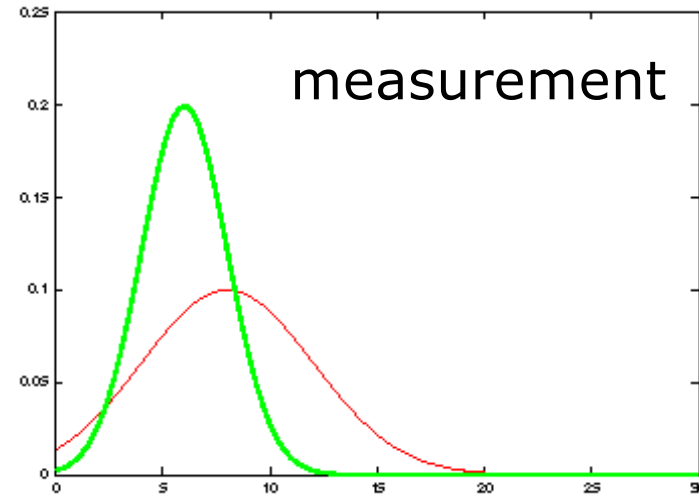
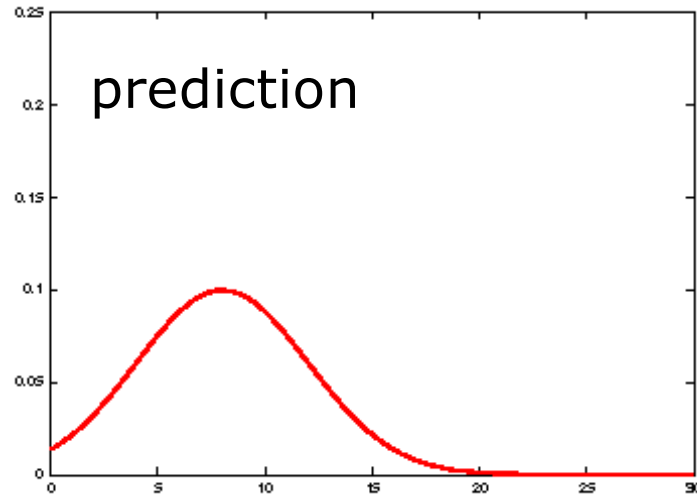
A_t Matrix ($n \times n$) that describes how the state evolves from $t-1$ to t without controls or noise.

B_t Matrix ($n \times l$) that describes how the control u_t changes the state from $t-1$ to t .

C_t Matrix ($k \times n$) that describes how to map the state x_t to an observation z_t .

ε_t Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t respectively.

Kalman Filter Updates in 1D

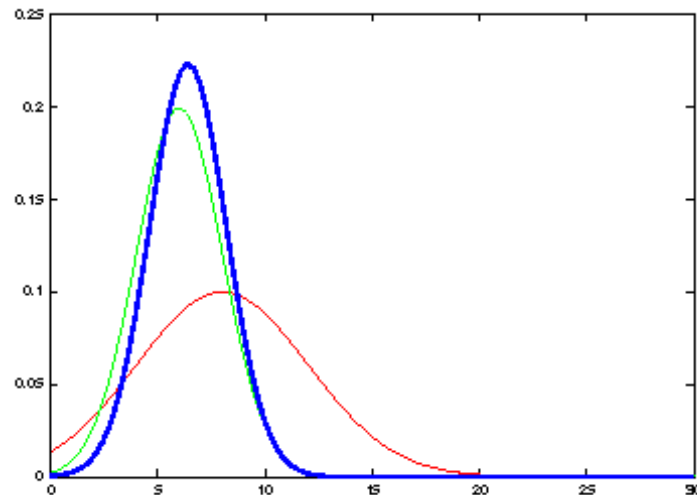


It's a weighted mean!

Kalman Filter Updates in 1D

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + \overset{\text{gain}}{K_t} (z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t) \bar{\sigma}_t^2 \end{cases} \quad \text{with} \quad K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

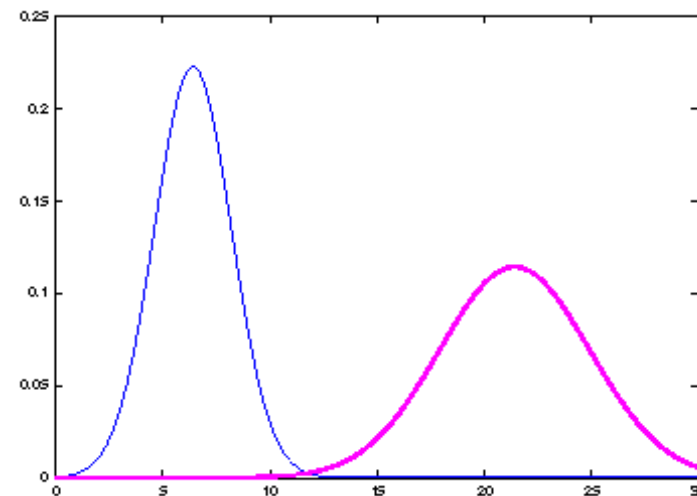
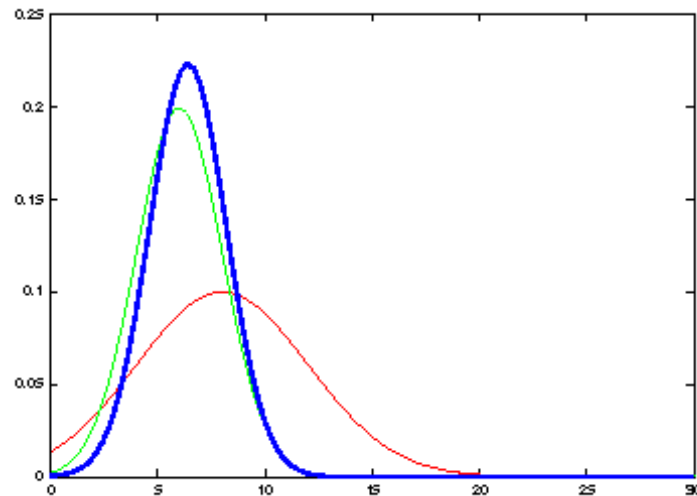
$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$



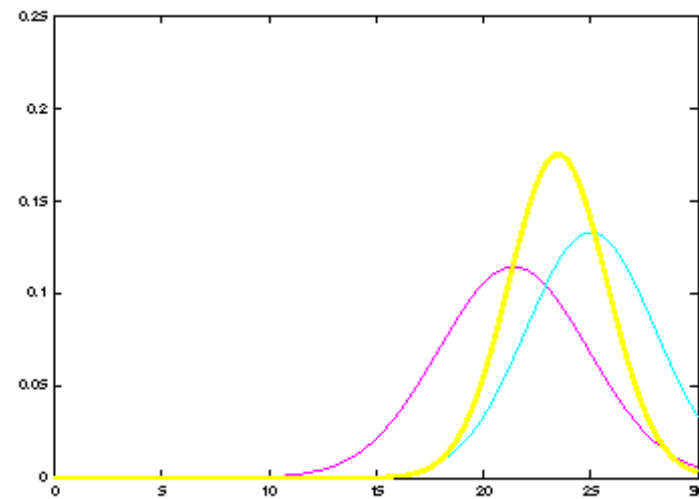
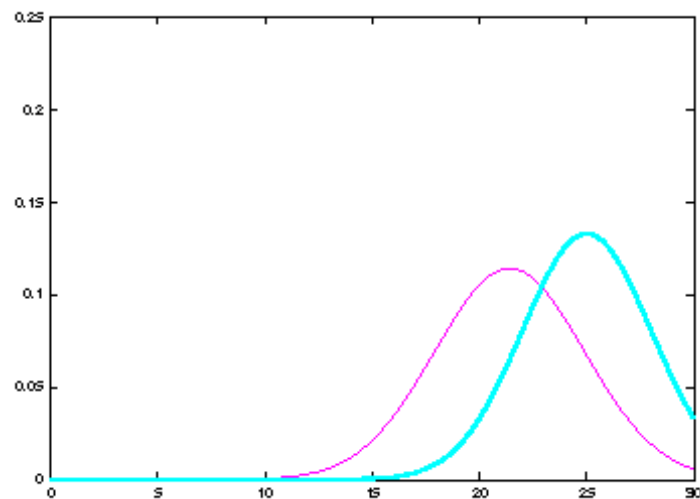
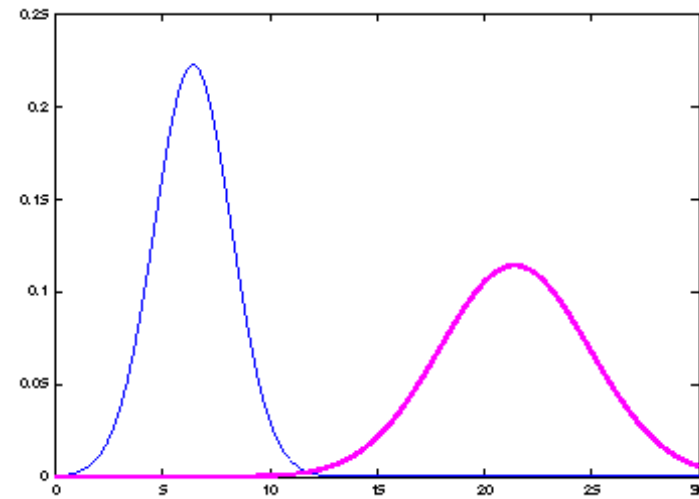
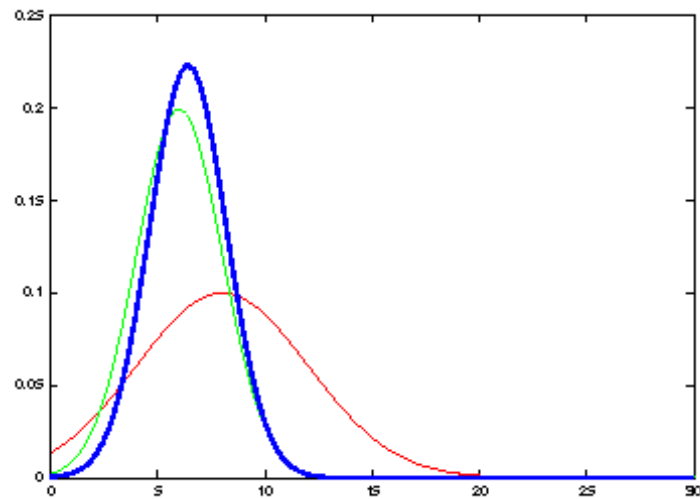
Kalman Filter Updates in 1D

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_{t-1}^2 + \sigma_{act,t}^2 \end{cases}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$



Kalman Filter Updates

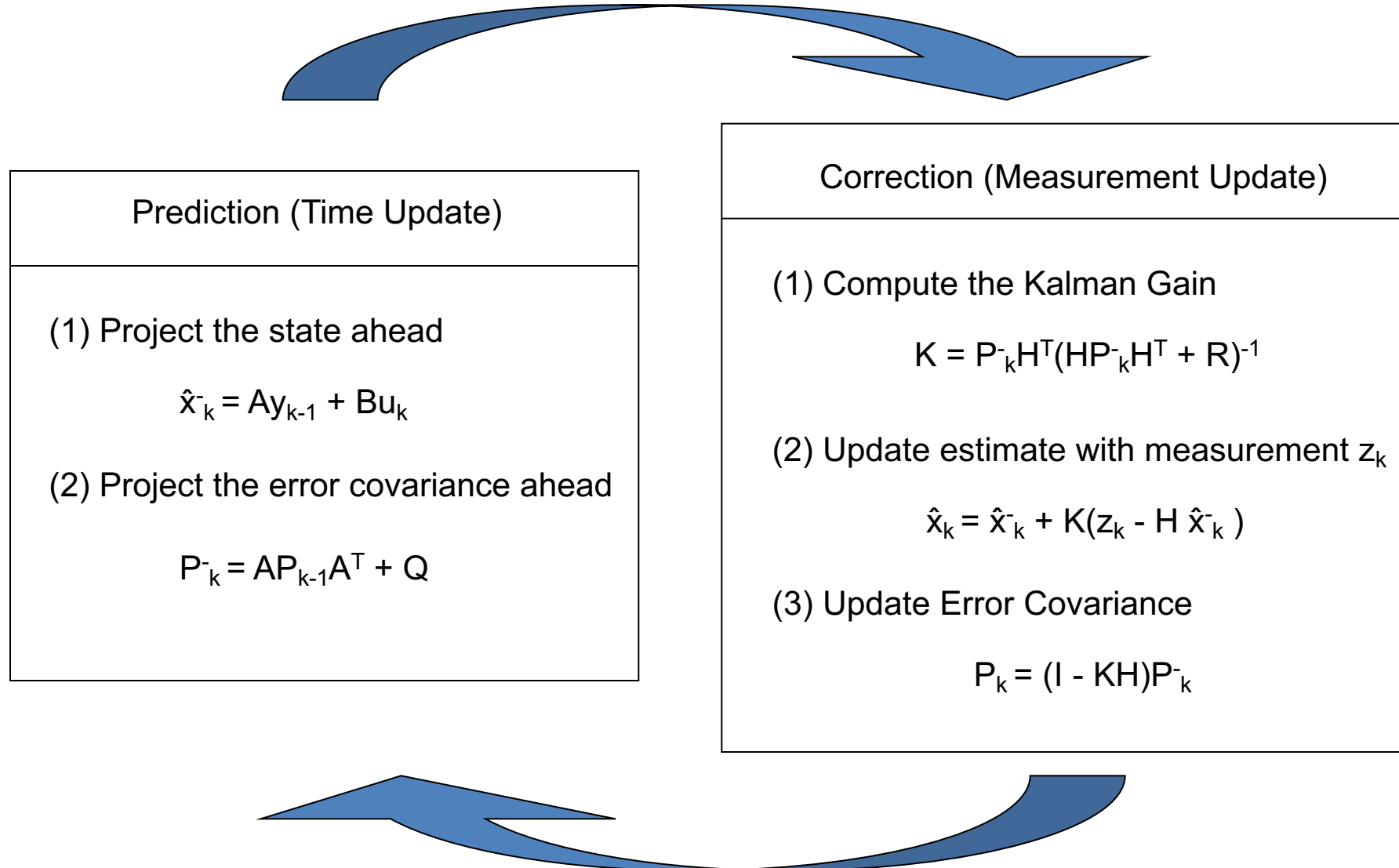


EXTENDED KALMAN FILTER (EKF)

Following Material:

- Cyrill Stachniss, University of Bonn

Kalman Filter



EKF: Non-linear Dynamic Systems

- Most realistic problems (in robotics) involve nonlinear functions

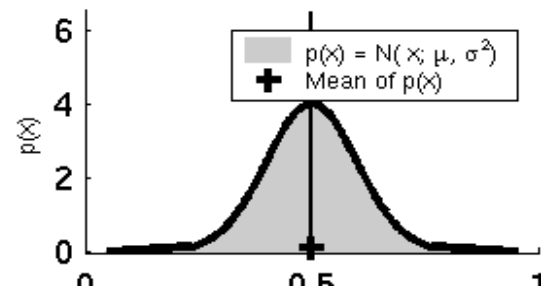
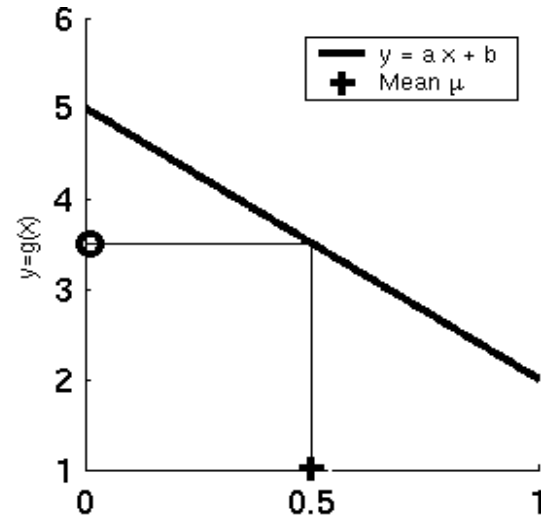
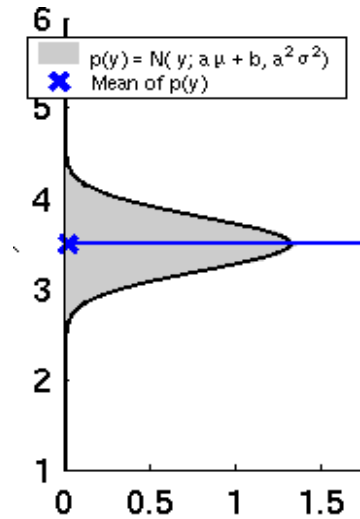
$$\cancel{x_t = A_t x_{t-1} + B_t u_t + \epsilon_t} \quad \cancel{z_t = C_t x_t + \delta_t}$$



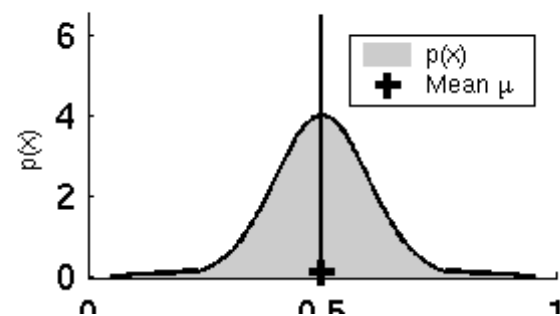
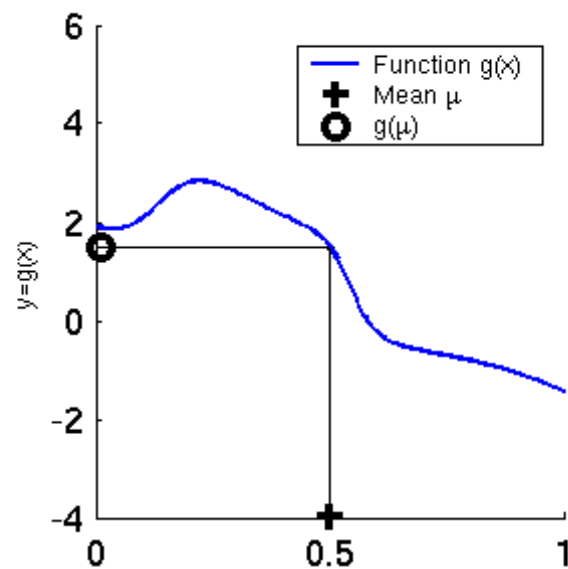
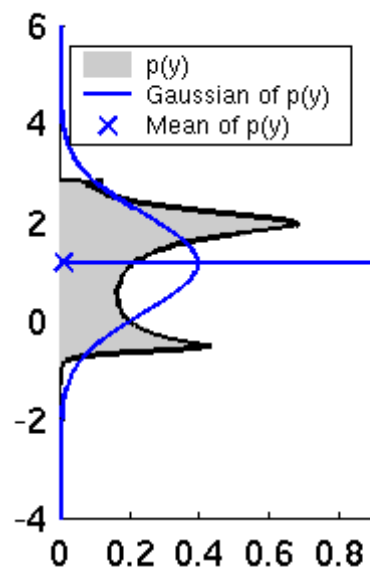
$$x_t = g(u_t, x_{t-1}) + \epsilon_t \quad z_t = h(x_t) + \delta_t$$

- Extended Kalman filter relaxes linearity assumption

Linearity Assumption Revisited



Non-linear system



Other Error Prop. Techniques

- **Second-Order Error Propagation**

Rarely used (complex expressions)

- **Monte-Carlo**

Non-parametric representation of uncertainties

1. Sampling from $p(X)$
2. Propagation of samples
3. Histogramming
4. Normalization

- **Extended Kalman Filter: EKF**

EKF Linearization: First Order Taylor Expansion

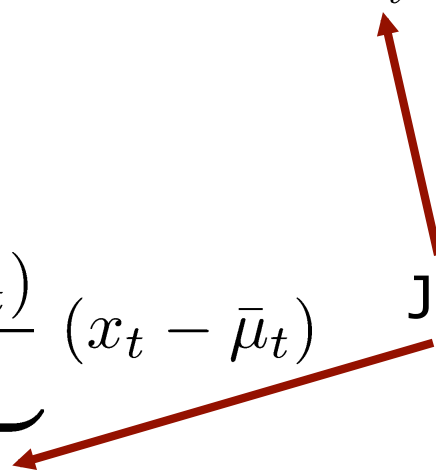
- Prediction:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=: G_t} (x_{t-1} - \mu_{t-1})$$

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=: H_t} (x_t - \bar{\mu}_t)$$

Jacobian matrices



Jacobian Matrix

- It's a **non-square matrix** $n \times m$ in general
- Suppose you have a vector-valued function $f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix}$
- Let the **gradient operator** be the vector of (first-order) partial derivatives

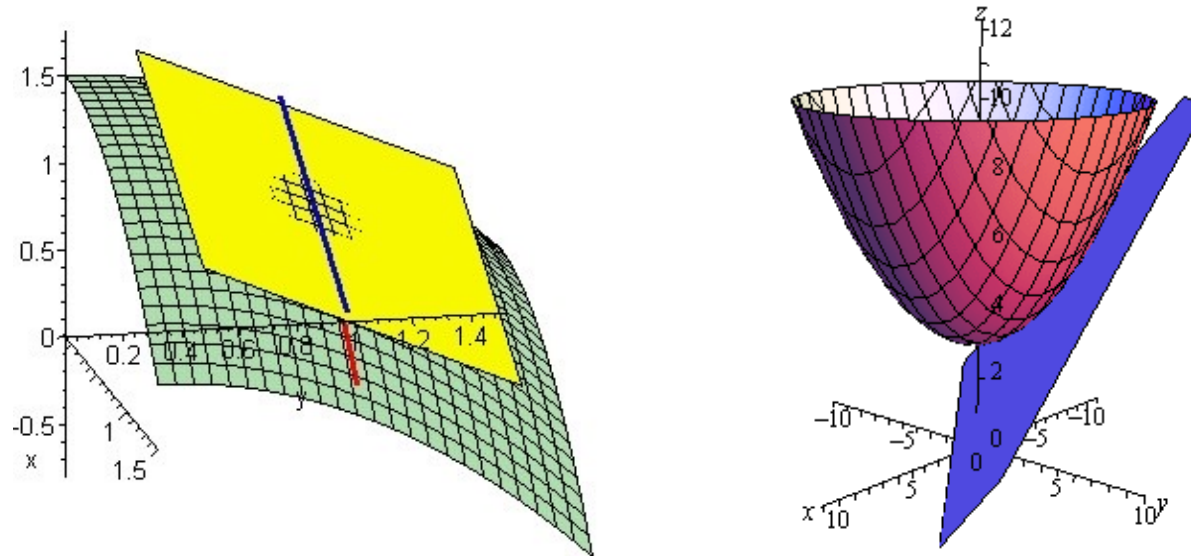
$$\nabla_{\mathbf{x}} = \left[\frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \cdots \quad \frac{\partial}{\partial x_n} \right]^T$$

- Then, the **Jacobian matrix** is defined as

$$\mathbf{F}_{\mathbf{x}} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \cdot \left[\frac{\partial}{\partial x_1} \quad \cdots \quad \frac{\partial}{\partial x_n} \right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_2}{\partial x_n} \end{bmatrix}$$

Jacobian Matrix

- It's the orientation of the **tangent plane** to the vector-valued function at a given point



- **Generalizes the gradient** of a scalar valued function
- Heavily used for **first-order error propagation...**

EKF Linearization: First Order Taylor Expansion

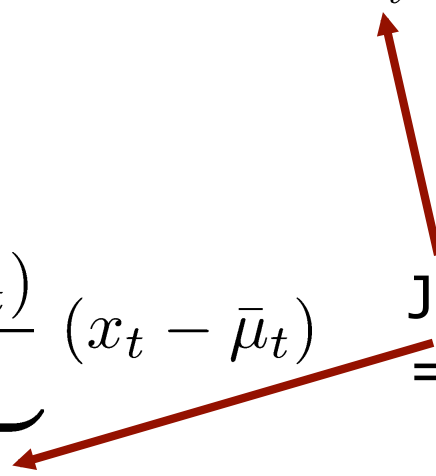
- Prediction:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \underbrace{\frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}}_{=: G_t} (x_{t-1} - \mu_{t-1})$$

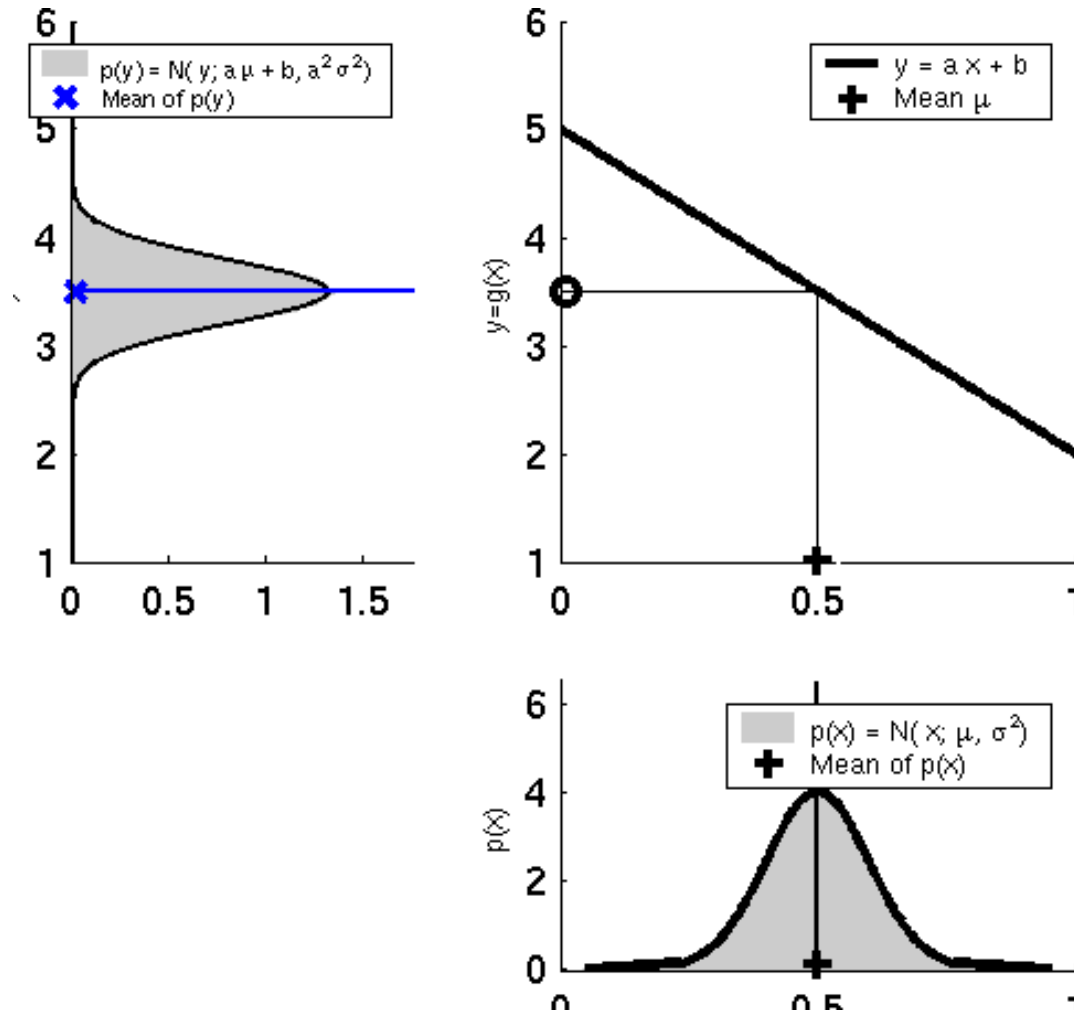
- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \underbrace{\frac{\partial h(\bar{\mu}_t)}{\partial x_t}}_{=: H_t} (x_t - \bar{\mu}_t)$$

Jacobian matrices
=> linear functions!

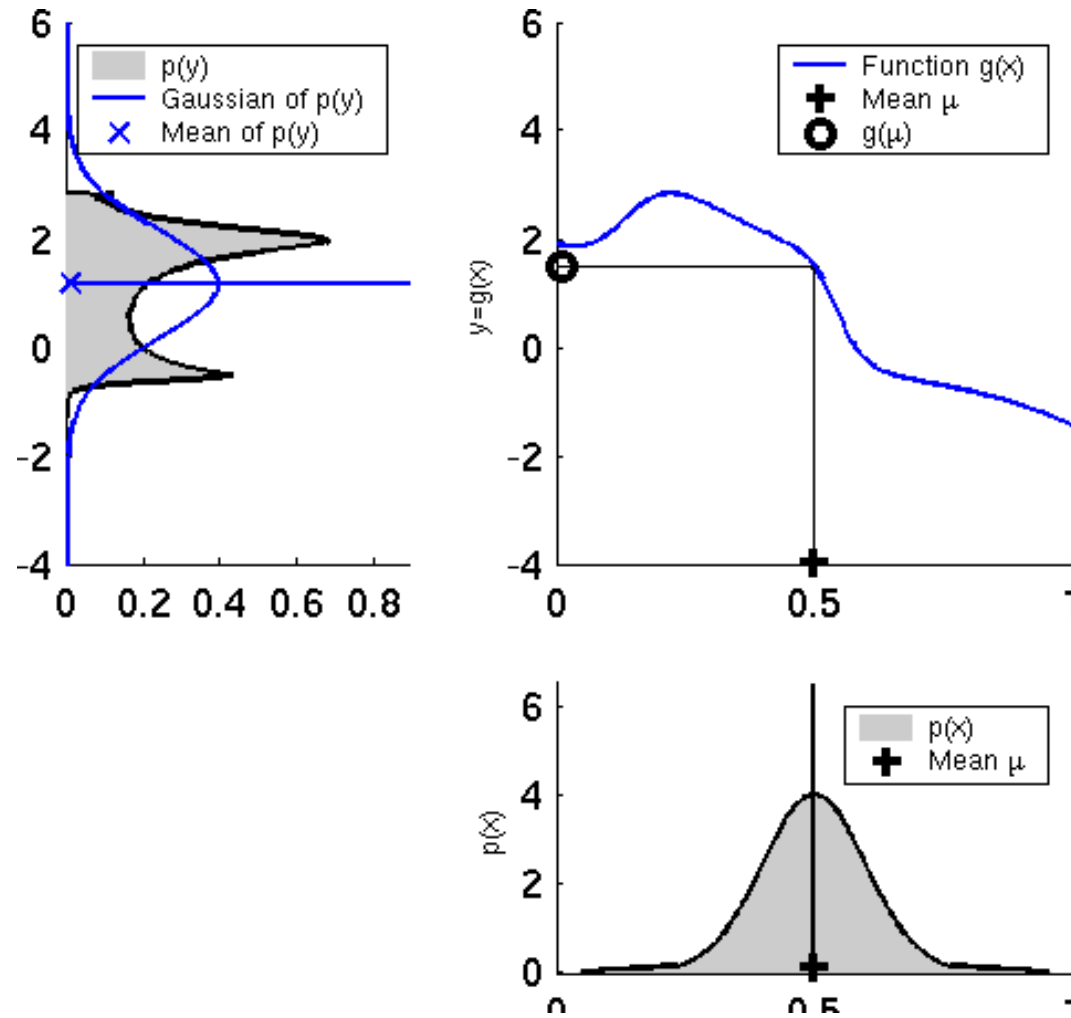


Linearity Assumption Revisited



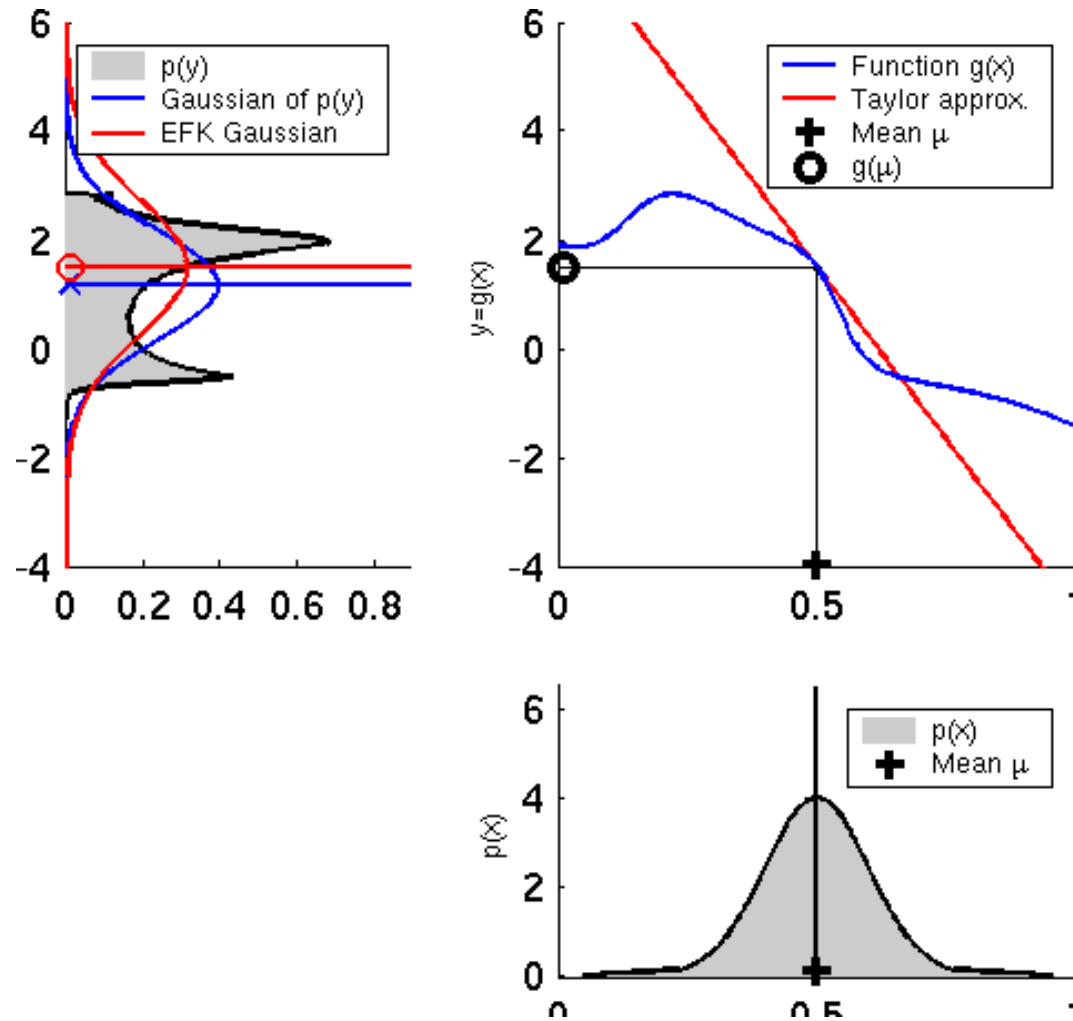
Courtesy: Thrun, Burgard, Fox

Non-Linear Function



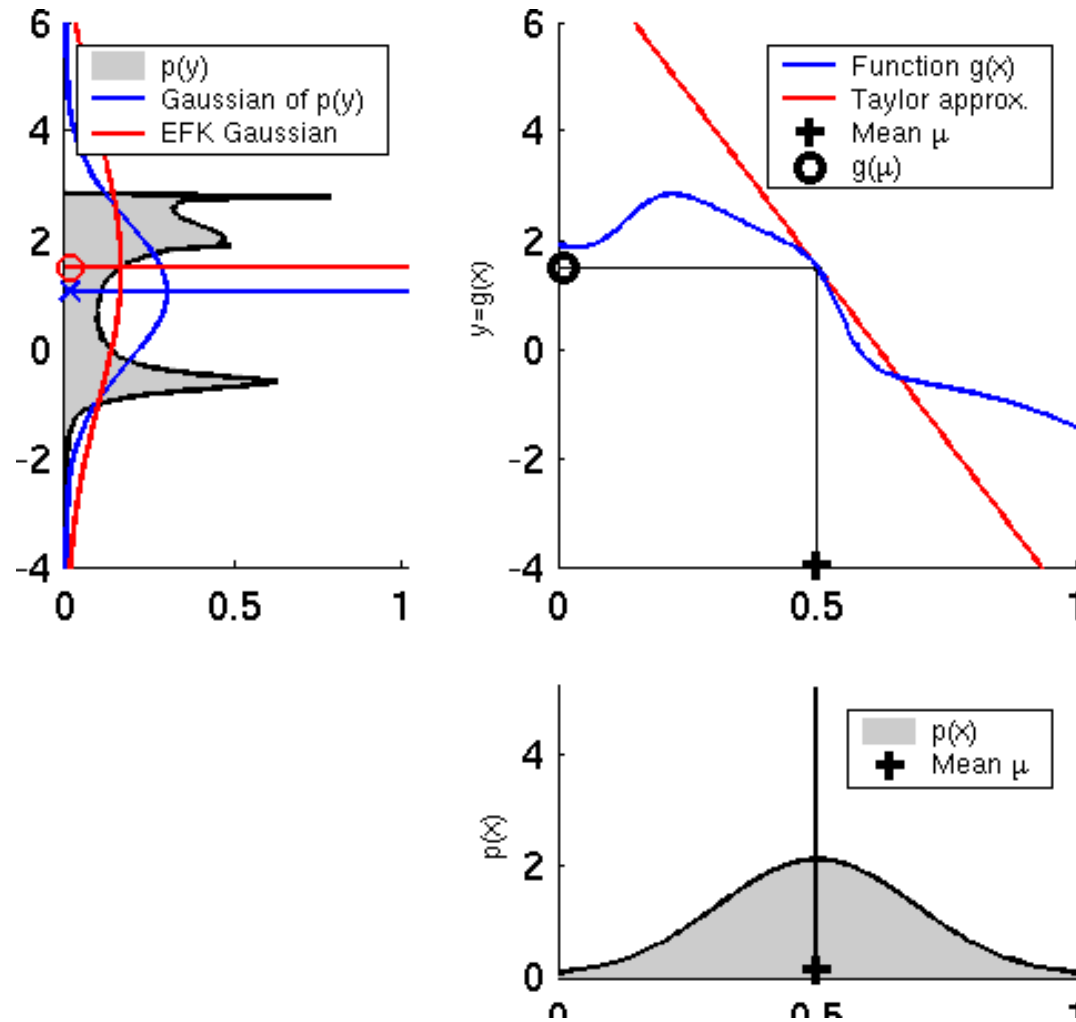
Courtesy: Thrun, Burgard, Fox

EKF Linearization (1)



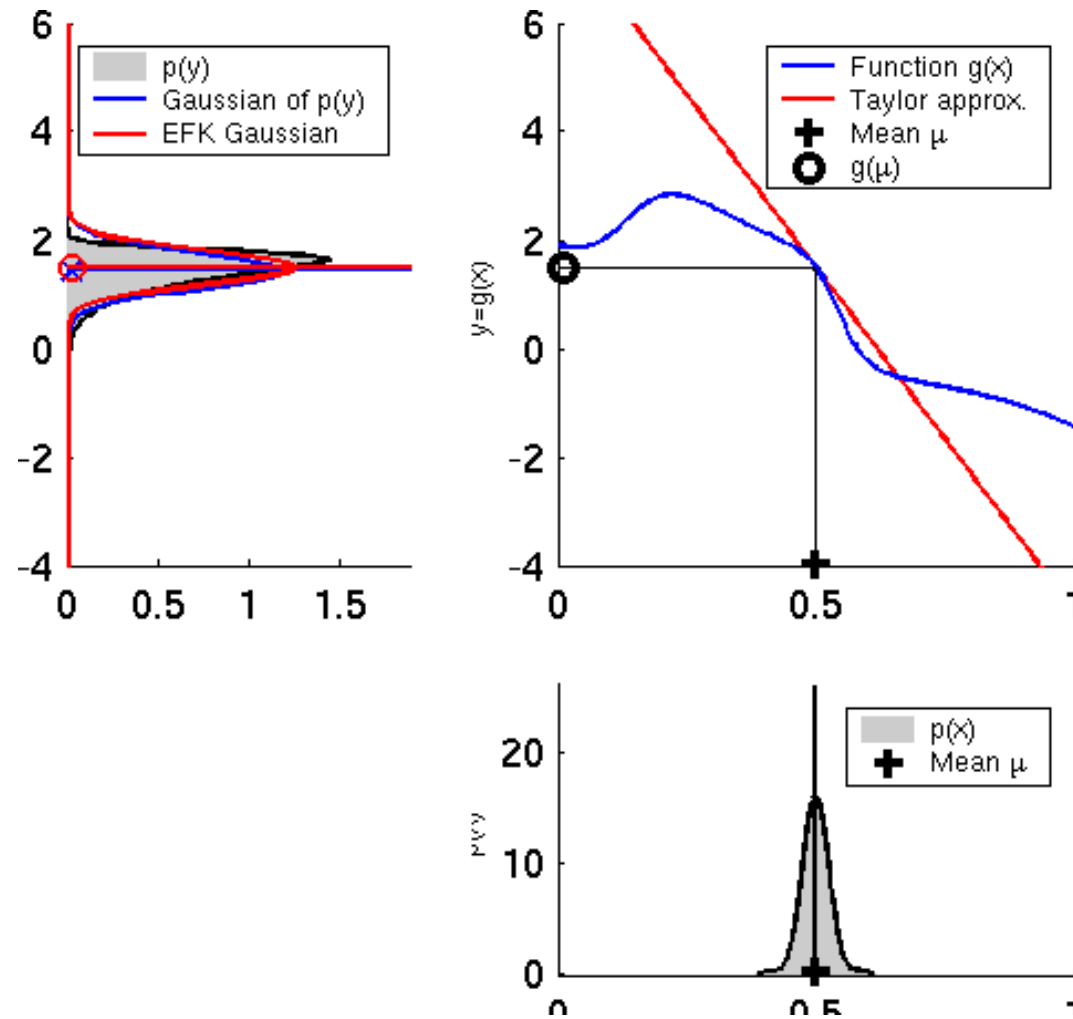
Courtesy: Thrun, Burgard, Fox

EKF Linearization (2)



Courtesy: Thrun, Burgard, Fox

EKF Linearization (3)



Extended Kalman Filter Algorithm

- 1: **Extended_Kalman_filter**($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):
- 2: $\bar{\mu}_t = \underline{g}(u_t, \mu_{t-1})$
- 3: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
- 4: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
- 5: $\mu_t = \bar{\mu}_t + K_t (z_t - \underline{h}(\bar{\mu}_t))$
- 6: $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
- 7: *return* μ_t, Σ_t

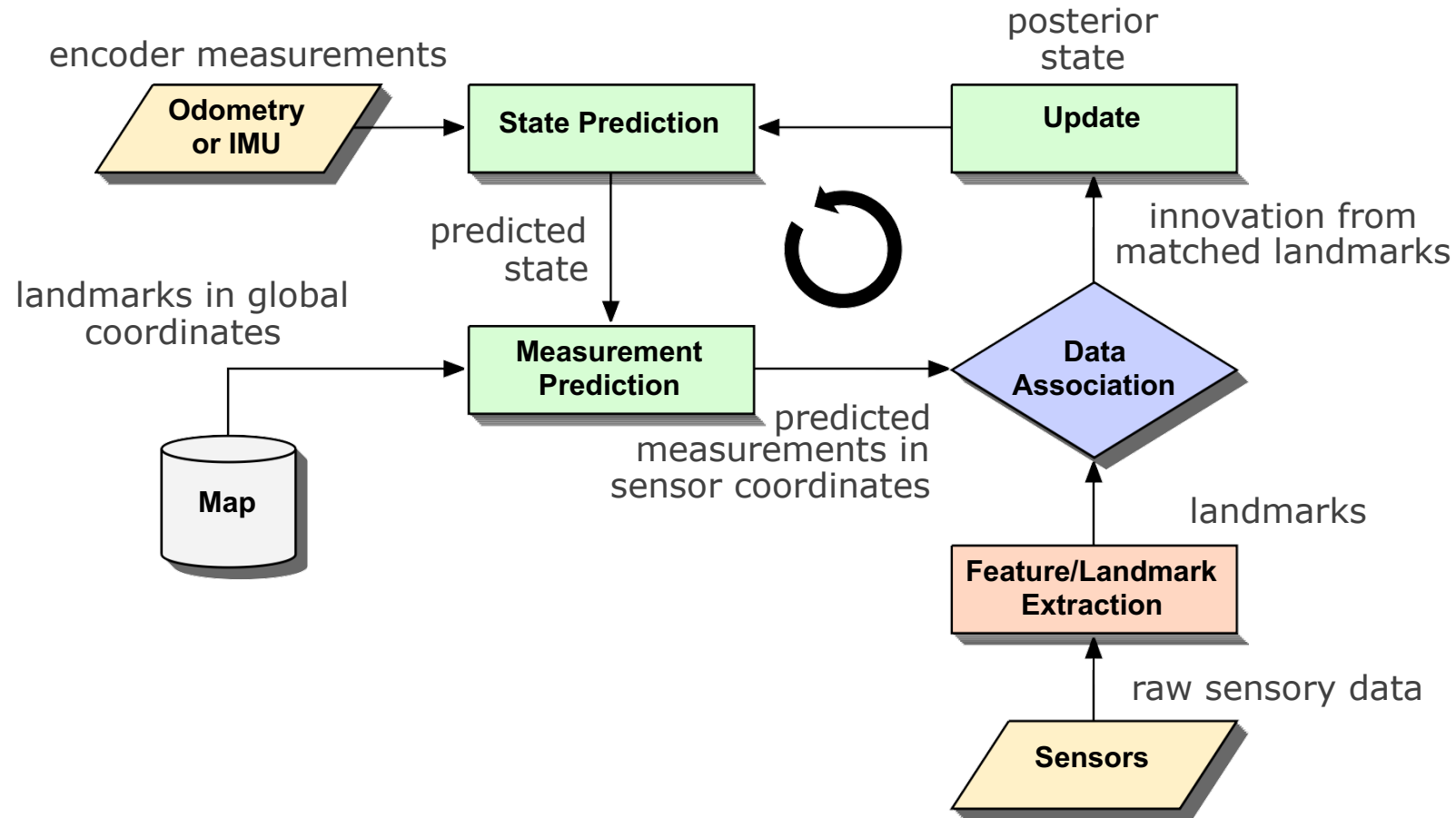
$$A_t \leftrightarrow G_t$$

$$C_t \leftrightarrow H_t$$

KF vs. EKF

Landmark-based Localization

EKF Localization: Basic Cycle



Landmark-based Localization

State Prediction (Odometry)

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

$$\hat{C}_k = F_x C_k F_x^T + F_u U_k F_u^T$$

Control \mathbf{u}_k : wheel displacements s_l, s_r

$$\mathbf{u}_k = (s_l \ s_r)^T \quad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

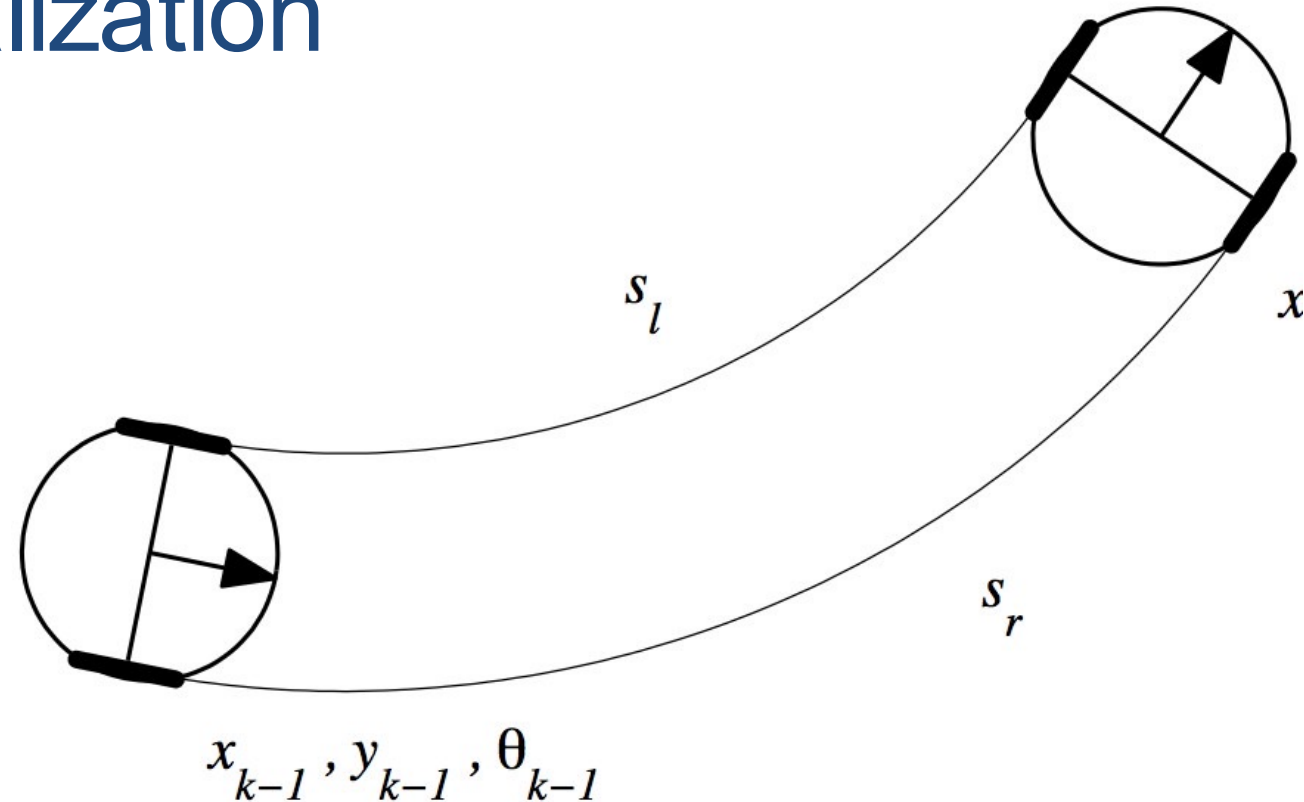
Error model: linear growth

$$\sigma_l = k_l |s_l|$$

$$\sigma_r = k_r |s_r|$$

Nonlinear process model f :

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left(-\sin \theta_{k-1} + \sin\left(\theta_{k-1} + \frac{s_r - s_l}{b}\right) \right) \\ \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left(\cos \theta_{k-1} - \cos\left(\theta_{k-1} + \frac{s_r - s_l}{b}\right) \right) \\ \frac{s_r - s_l}{b} \end{bmatrix}$$



Landmark-based Localization

State Prediction (Odometry)

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

$$\hat{C}_k = F_x C_k F_x^T + F_u U_k F_u^T$$

Control \mathbf{u}_k : wheel displacements s_l, s_r

$$\mathbf{u}_k = (s_l \ s_r)^T \quad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

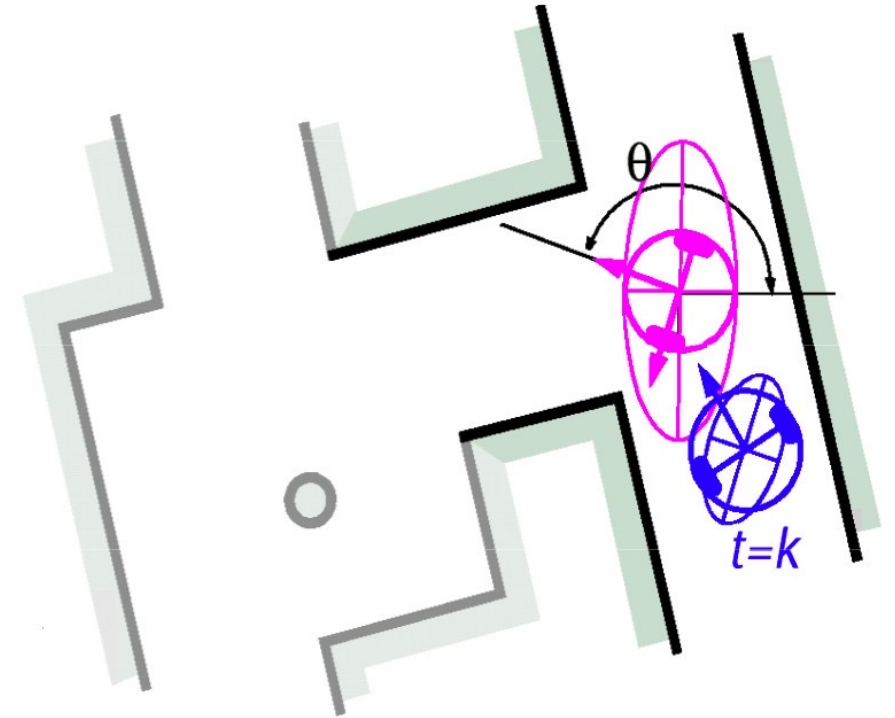
Error model: linear growth

$$\sigma_l = k_l |s_l|$$

$$\sigma_r = k_r |s_r|$$

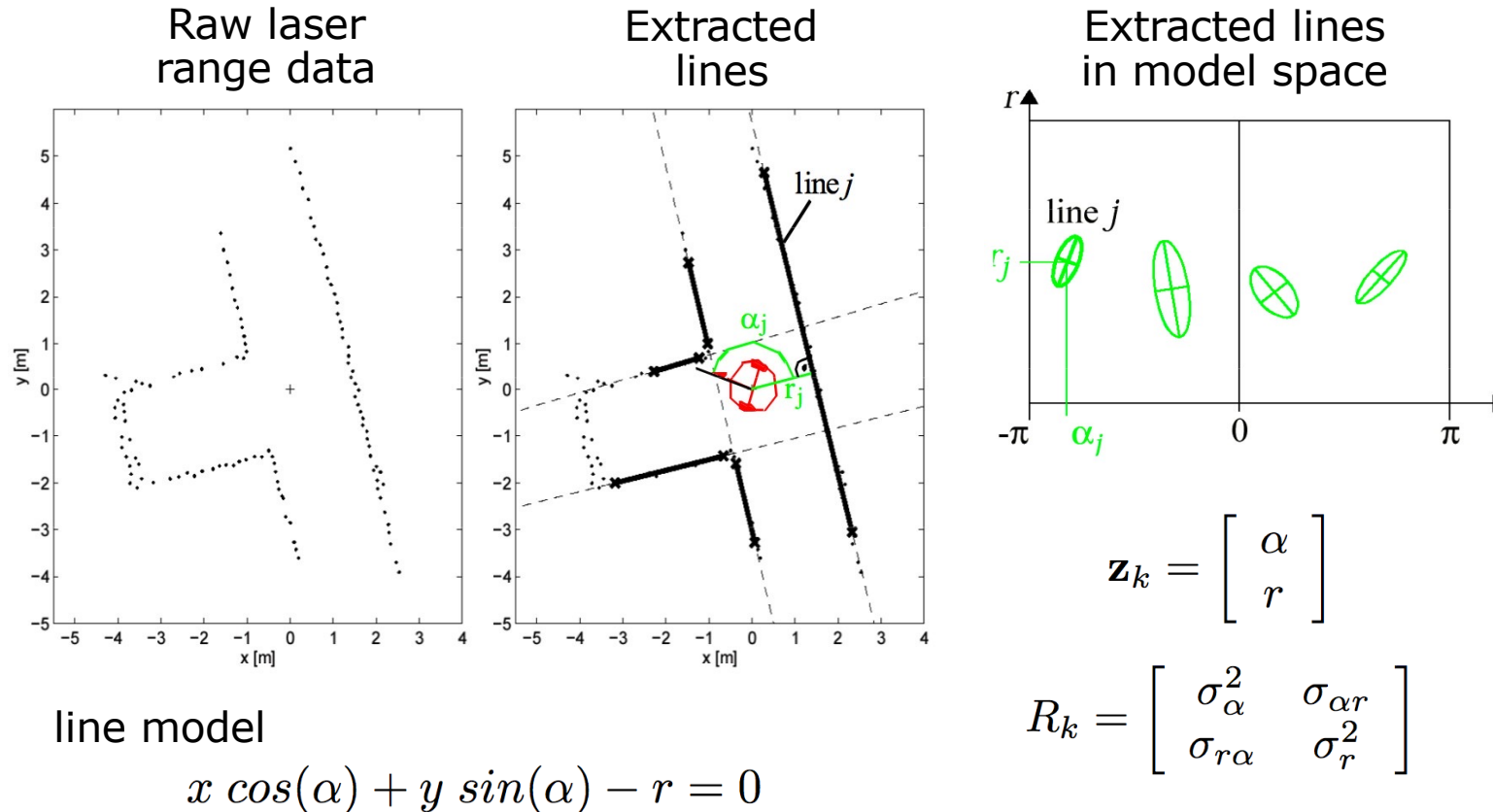
Nonlinear process model f :

$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left(-\sin \theta_{k-1} + \sin\left(\theta_{k-1} + \frac{s_r - s_l}{b}\right) \right) \\ \frac{b}{2} \frac{s_l + s_r}{s_r - s_l} \left(\cos \theta_{k-1} - \cos\left(\theta_{k-1} + \frac{s_r - s_l}{b}\right) \right) \\ \frac{s_r - s_l}{b} \end{bmatrix}$$



Landmark-based Localization

Landmark Extraction (Observation)

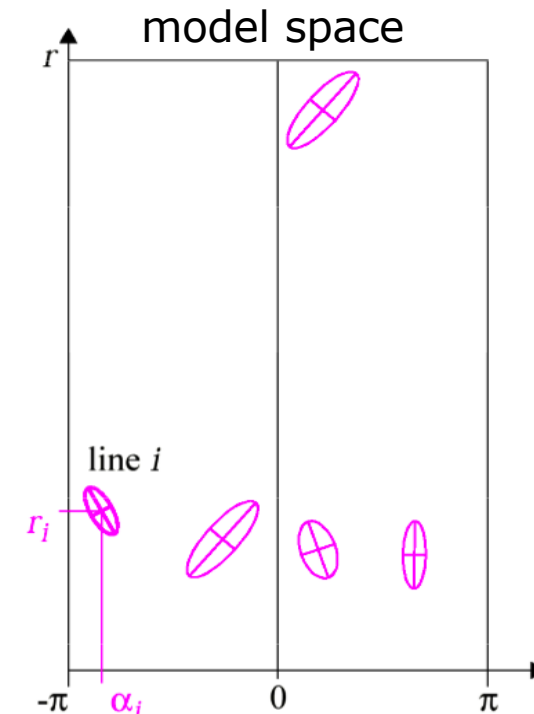
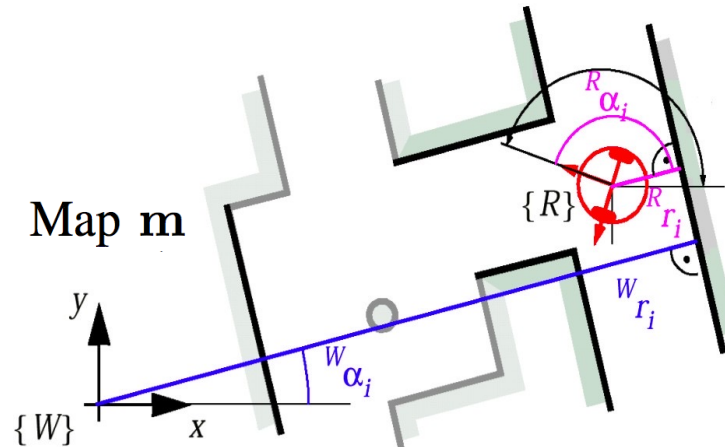


Landmark-based Localization

Measurement Prediction

- ...is a coordinate frame transform world-to-sensor
- Given the predicted state (robot pose), predicts the location $\hat{\mathbf{z}}_k$ and location uncertainty $H \hat{C}_k H^T$ of expected observations in sensor coordinates

$$\hat{\mathbf{z}}_k = h(\hat{\mathbf{x}}_k, \mathbf{m})$$



Landmark-based Localization

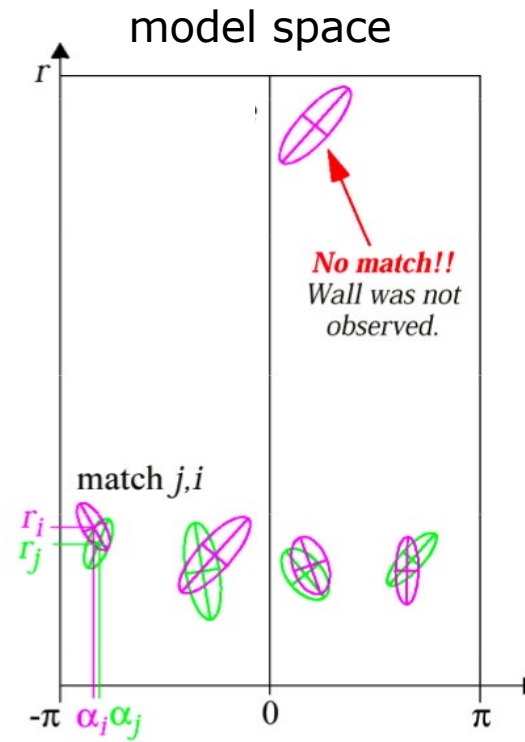
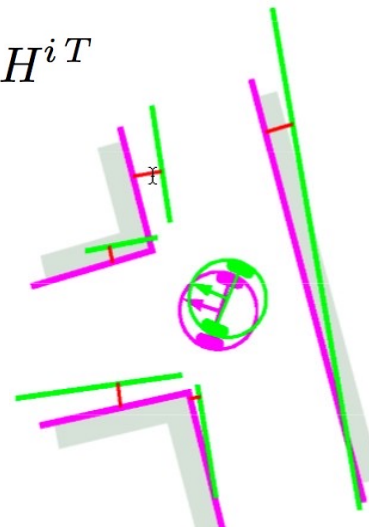
Data Association (Matching)

- Associates predicted measurements $\hat{\mathbf{z}}_k^i$ with observations \mathbf{z}_k^j

$$\nu_k^{ij} = \mathbf{z}_k^j - \hat{\mathbf{z}}_k^i$$

$$S_k^{ij} = R_k^j + H^i \hat{C}_k H^{iT}$$

- Innovation and innovation covariance ν_k^{ij}



Green: observation

Magenta: measurement prediction

Landmark-based Localization

Update

- Kalman gain

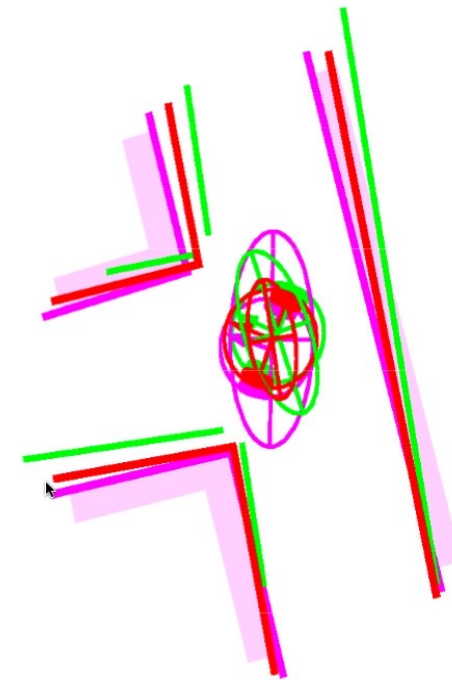
$$K_k = \hat{C}_k H^T S_k^{-1}$$

- State update (robot pose)

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + K_k \nu_k$$

- State covariance update

$$C_k = (I - K_k H) \hat{C}_k$$



Red: posterior estimate