上海科技大学
ShanghaiTech University

CS283: Robotics Fall 2020: SLAM II

Qingwen Xu Sören Schwertfeger

ShanghaiTech University

Admin

- HW3 is postponed to October 18, Sunday, 10pm
- The demo is now a video:
  - Video upload: https://star-center.shanghaitech.edu.cn/seafile/u/d/922df19bd14d40b88af2/
  - Only upload once! Once it is uploaded you cannot see/ change it anymore.
  - Maximum file size: 50MB (use good compression - but not too low quality) - you will loose points if your video is too big.
  - Naming convention: hw3_<email>.mp4 (replace <email> with your email user name).
- Paper presentation due: Thursday, Oct 15

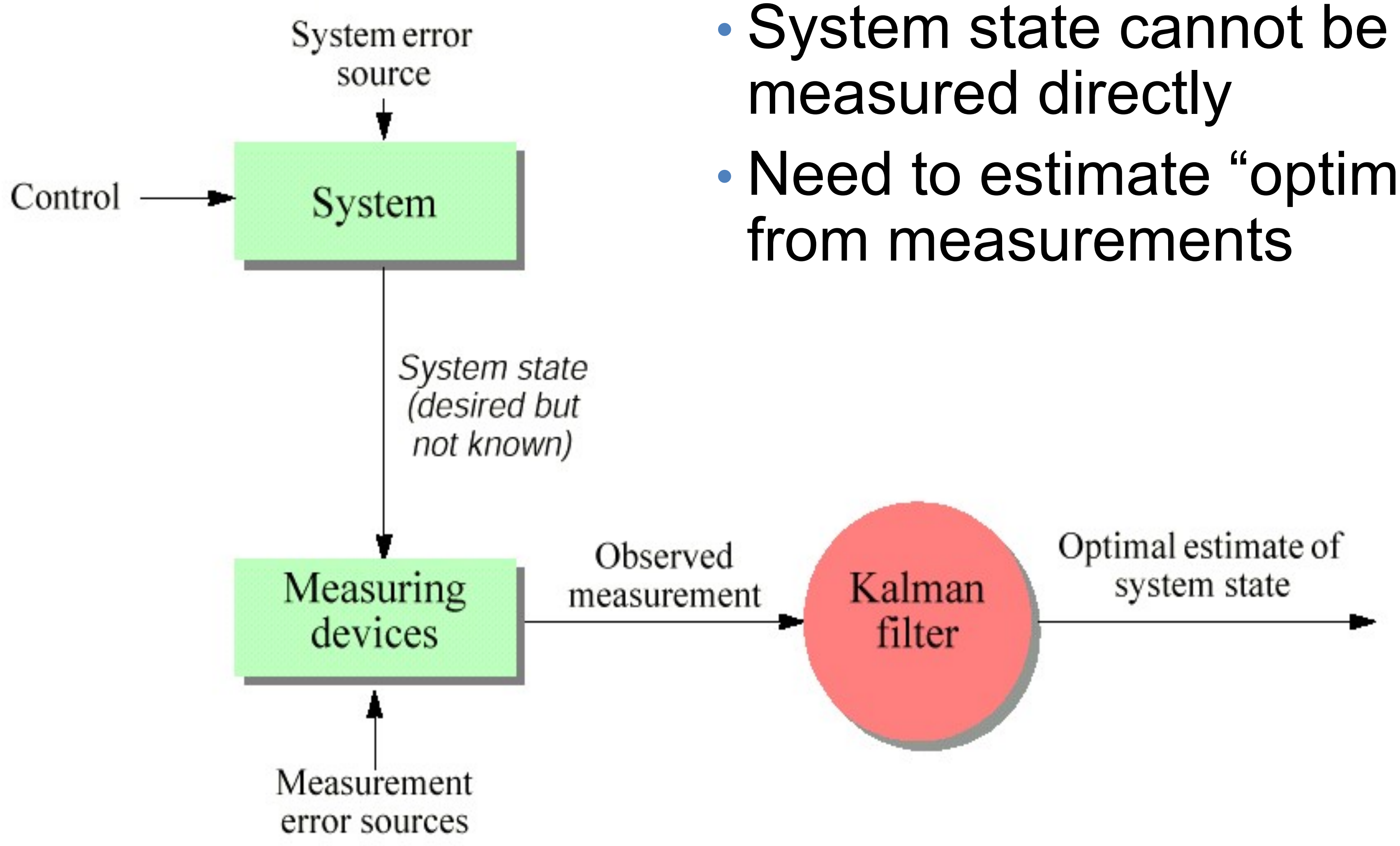
- Meet with your advisor this week! It is the groups responsibility to make the appointment! You lose points if the meeting is not documented in meetings.txt

KALMAN FILTER OVERVIEW

Following Material:

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

The Problem

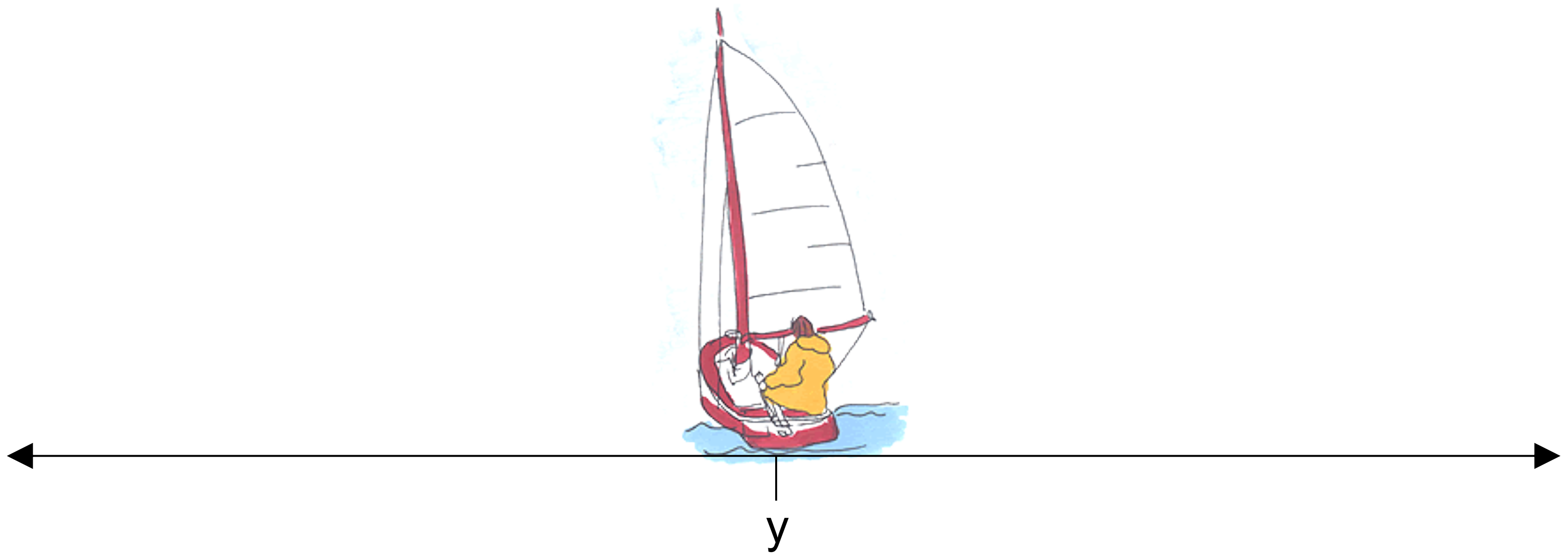

System error source
Control
System
System state (desired but not known)
Measuring devices
Observed measurement
Kalman filter
Optimal estimate of system state
Measurement error sources

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

What is a Kalman Filter?

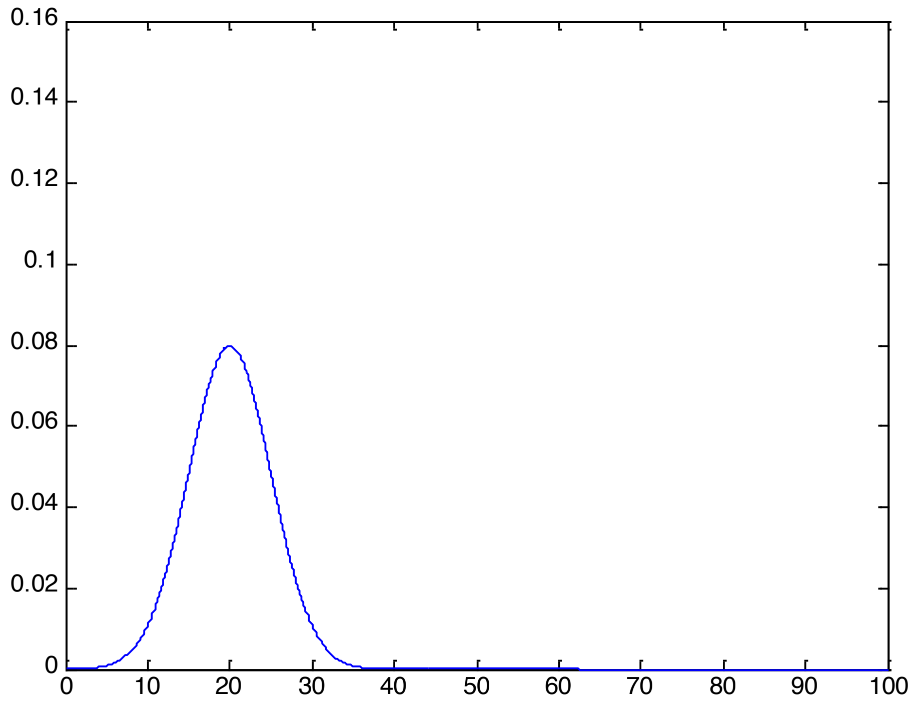
- Recursive data processing algorithm
- Generates optimal estimate of desired quantities given the set of measurements
- Optimal?
  - For linear system and white Gaussian errors, Kalman filter is “best” estimate based on all previous measurements
  - For non-linear system optimality is ‘qualified’
-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

Conceptual Overview



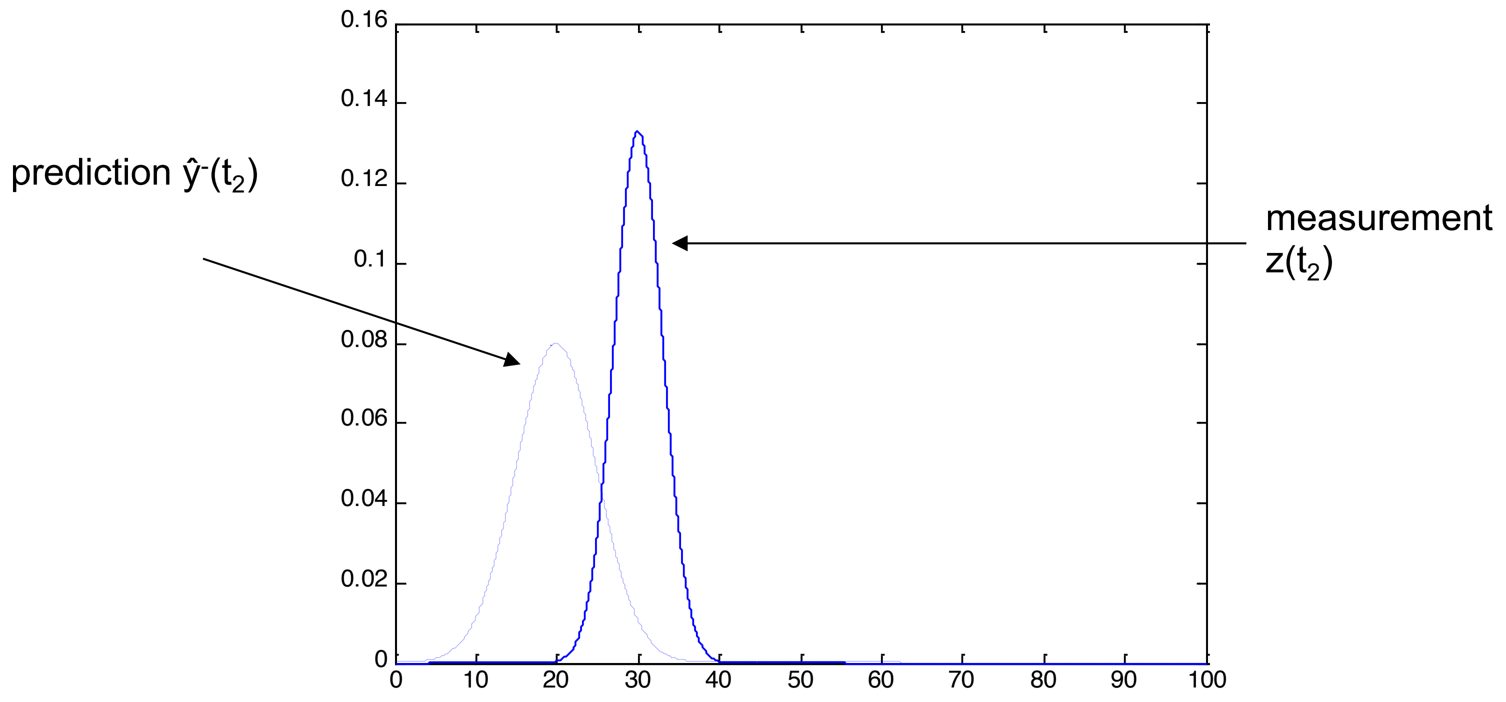
- Lost on the 1-dimensional line
- Position – y(t)
- Assume Gaussian distributed measurements

Conceptual Overview



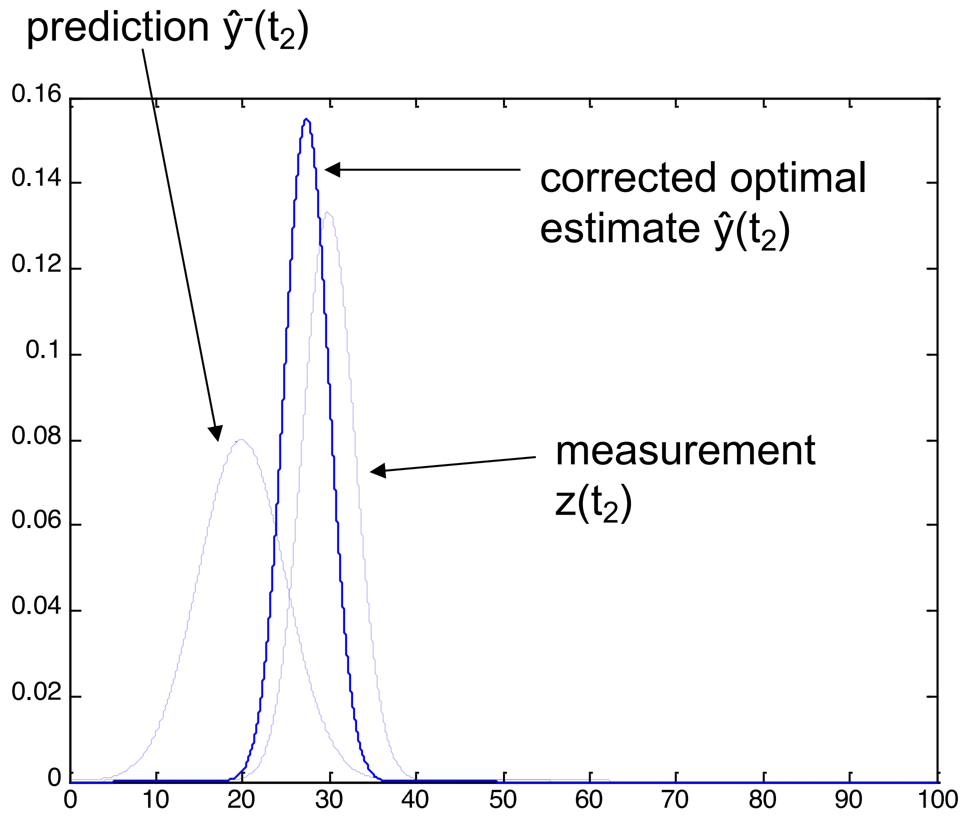
- Sextant Measurement at t_1: Mean = z_1 and Variance = σ_{z1}
- Optimal estimate of position is: $\hat{y}(t_1) = z_1$
- Variance of error in estimate: $\sigma^2_x (t_1) = \sigma^2_{z1}$
- Boat in same position at time t_2 - Predicted position is z_1

Conceptual Overview



- So we have the prediction $\hat{y}^{-}(t_2)$
- GPS Measurement at t_2: Mean = z_2 and Variance = σ_{z2}
- Need to <u>correct</u> the prediction due to measurement to get $\hat{y}(t_2)$
- 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 - $\hat{y}^-$, σ^-

$\Downarrow$

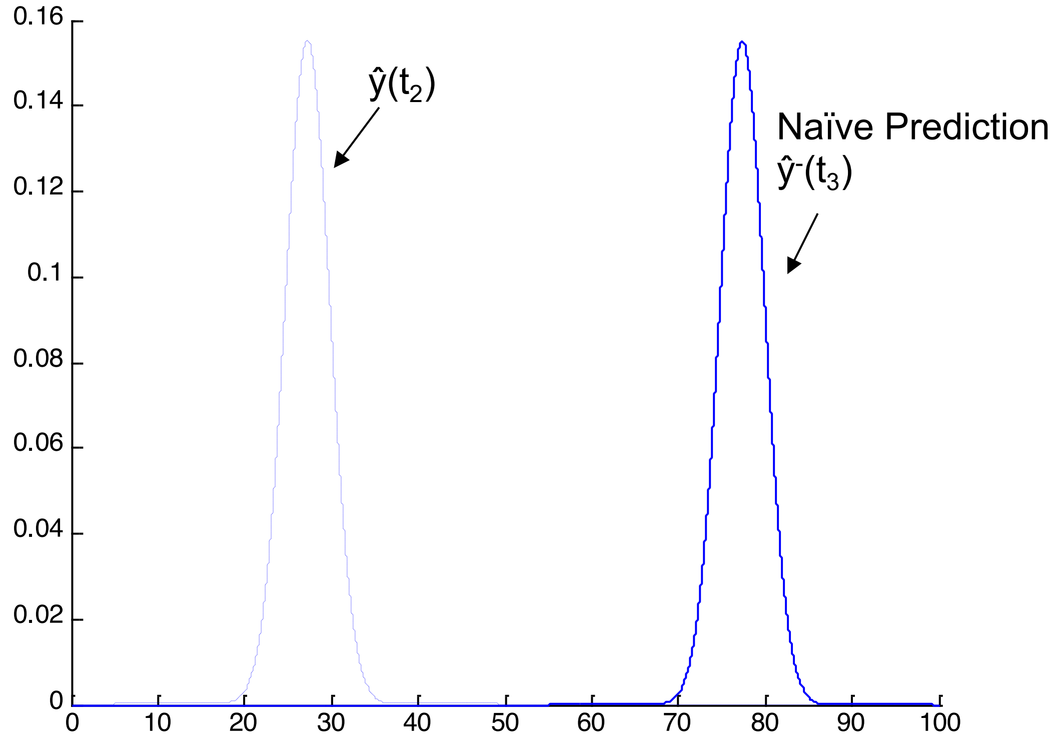
Take measurement – z_k, σ_z

$\Downarrow$

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

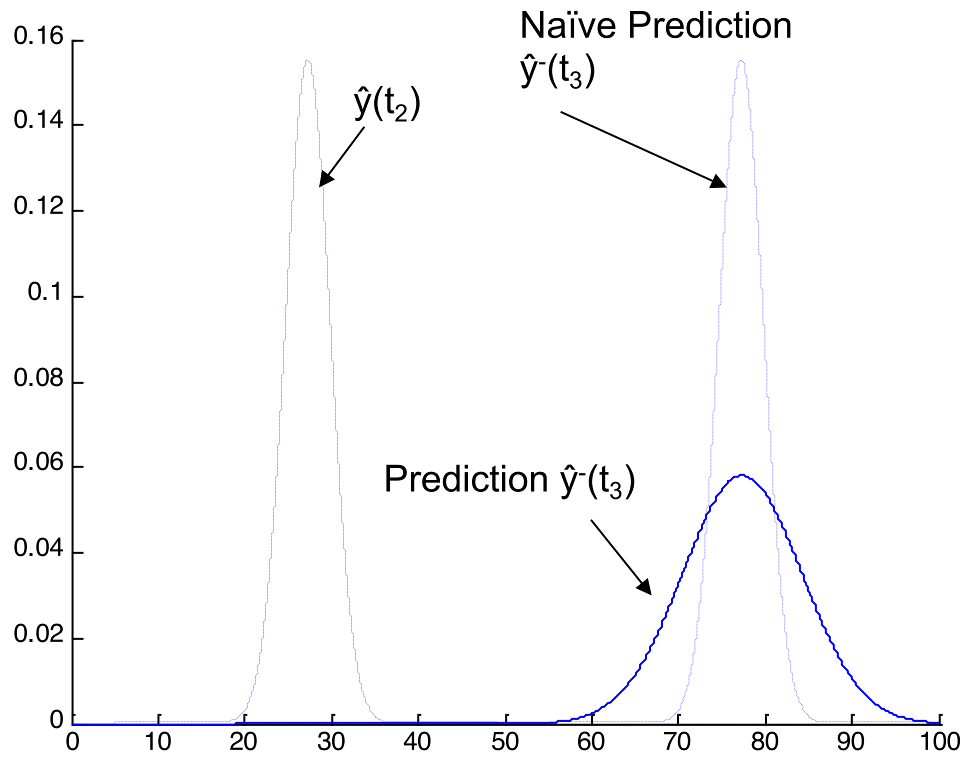
Variance of estimate = Variance of prediction * (1 – Kalman Gain)

Conceptual Overview



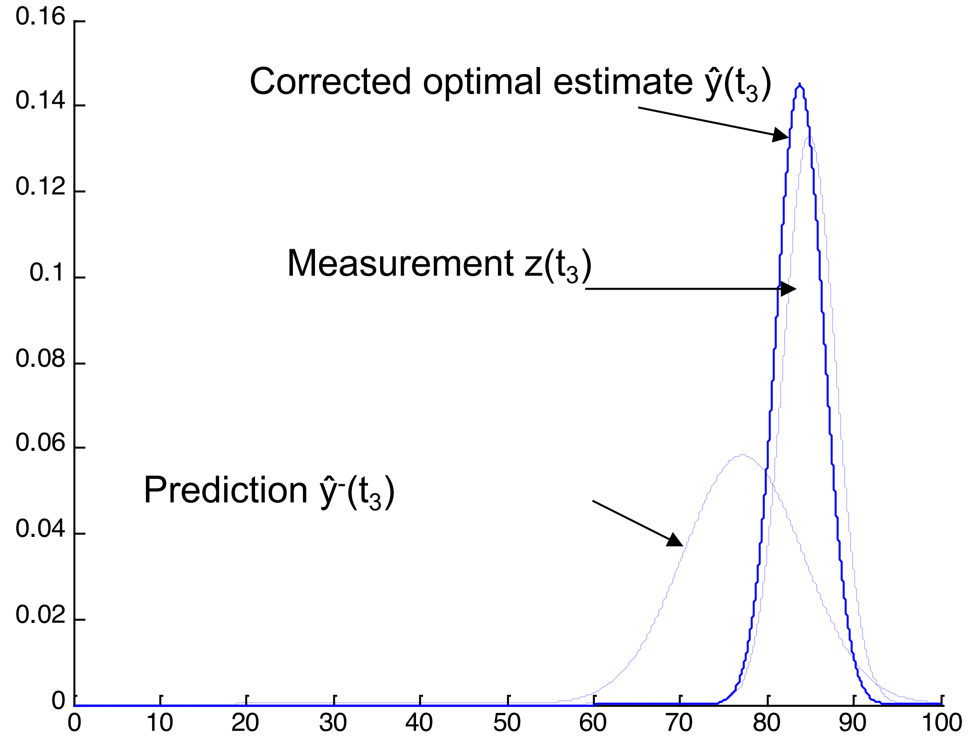
- At time t_3, boat moves with velocity dy/dt=u
- Naïve approach: Shift probability to the right to predict
- This would work if we knew the velocity exactly (perfect model)

Conceptual Overview



- Better to assume imperfect model by adding Gaussian noise
- dy/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 ($\hat{y}_{k-1}$ and σ_{k-1})
  - Prediction ($\hat{y}^-_k$, σ^-_k)
    - Use initial conditions and model (eg. constant velocity) to make prediction
  - Measurement (z_k)
    - Take measurement
  - Correction ($\hat{y}_k$, 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:

$y_k = Ay_{k-1} + Bu_k + w_{k-1}$ Process Noise (w) with covariance Q

$z_k = Hy_k + v_k$ Measurement Noise (v) with covariance R

- Kalman Filter

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

$$\hat{y}^-_k = Ay_{k-1} + Bu_k$$

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

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

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

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

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

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

Prediction (Time Update)

(1) Project the state ahead

$$\hat{y}^-_k = Ay_{k-1} + Bu_k$$

(2) Project the error covariance ahead

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

Correction (Measurement Update)

(1) Compute the Kalman Gain

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

(2) Update estimate with measurement z_k

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

(3) Update Error Covariance

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

KALMAN FILTER DETAILS

Following Material:

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

Bayes Filter

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

```
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.          $Bel'(x) = P(z \mid x) Bel(x)$
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.         $Bel'(x) = \int P(x \mid u, x') \ Bel(x') \ dx'$
12. Return Bel'(x)
```

Bayes Filter Reminder

- Prediction

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

- Correction

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

Kalman Filter

- Bayes filter with **Gaussians**
- Developed in the late 1950's
- Most relevant Bayes filter variant in practice
- Applications range from economics, wheather forecasting, satellite navigation to robotics and many more.

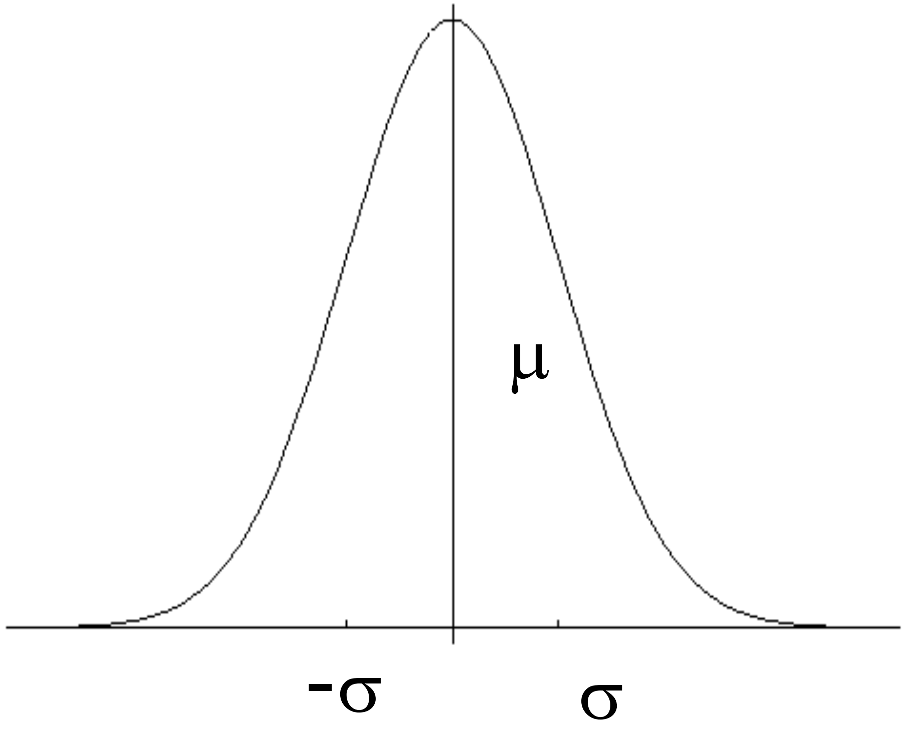
- The Kalman filter "algorithm" is a couple of **matrix multiplications**!

Gaussians

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

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

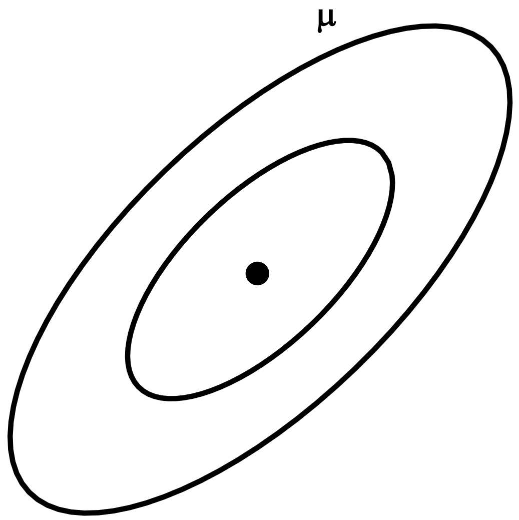
Univariate



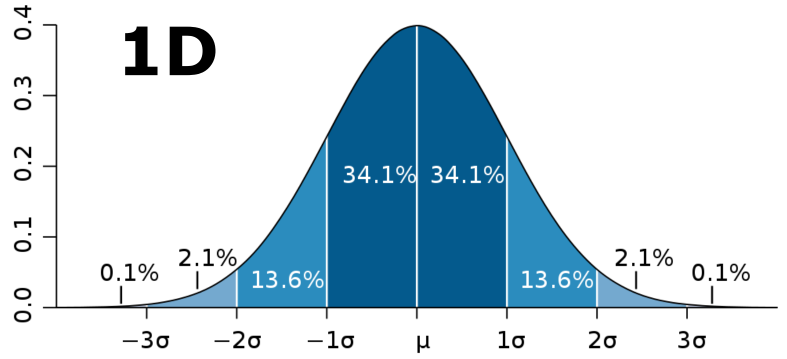
$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma}):$$

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

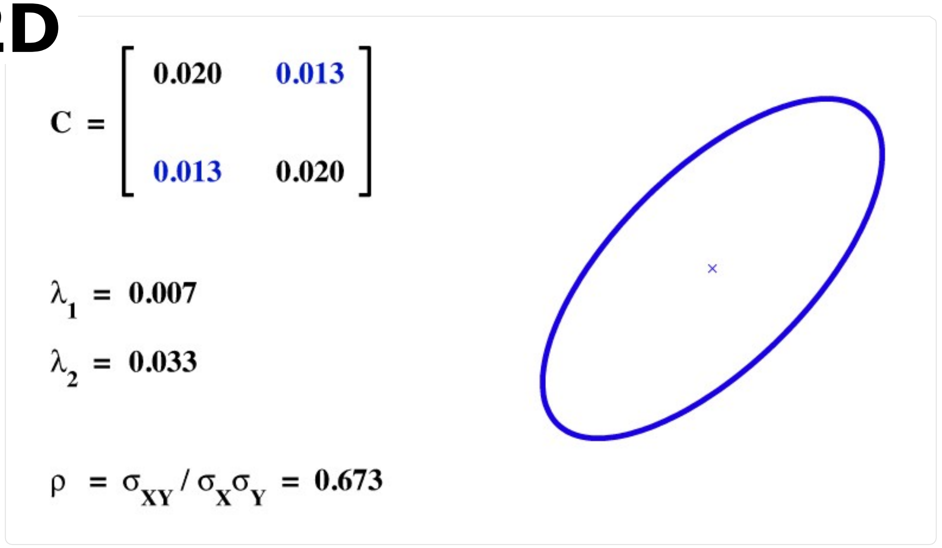
Multivariate



Gaussians



2D



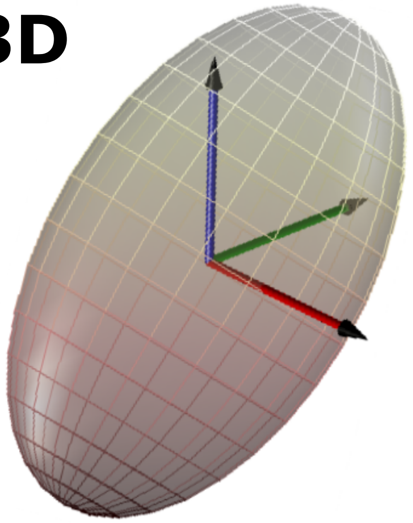
$$C = \begin{bmatrix} 0.020 & 0.013 \\ 0.013 & 0.020 \end{bmatrix}$$

$\lambda_1 = 0.007$

$\lambda_2 = 0.033$

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

3D



Properties of Gaussians

- Univariate

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

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

- Multivariate

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

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

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

Introduction to Kalman Filter (1)

- 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

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

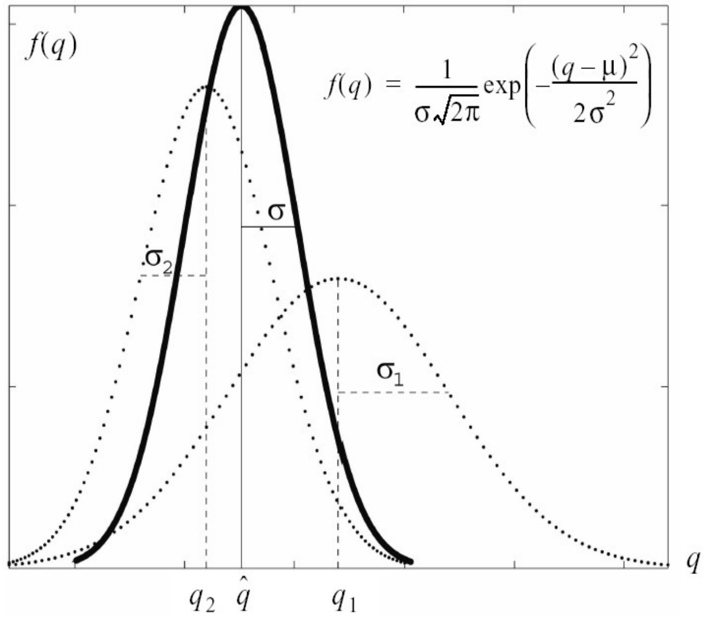
- Finding minimum error

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

- 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

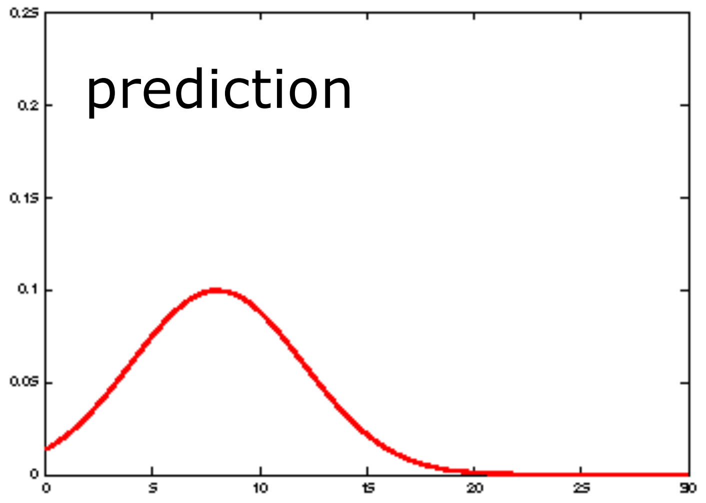
$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \quad \longleftarrow \quad \text{Process dynamics}$$

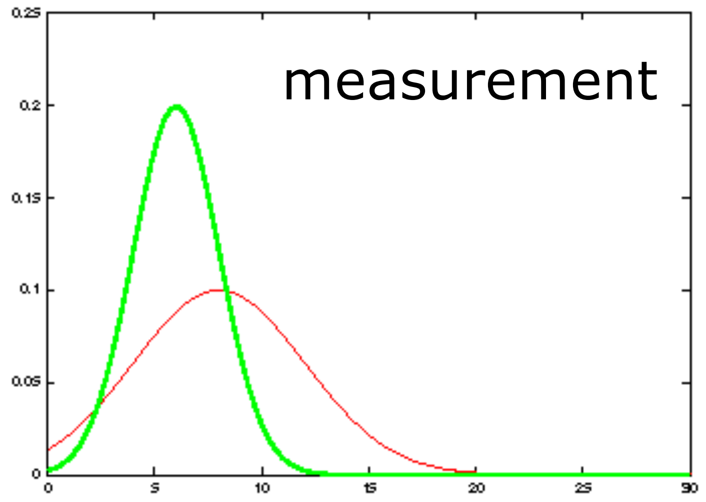
- with a measurement

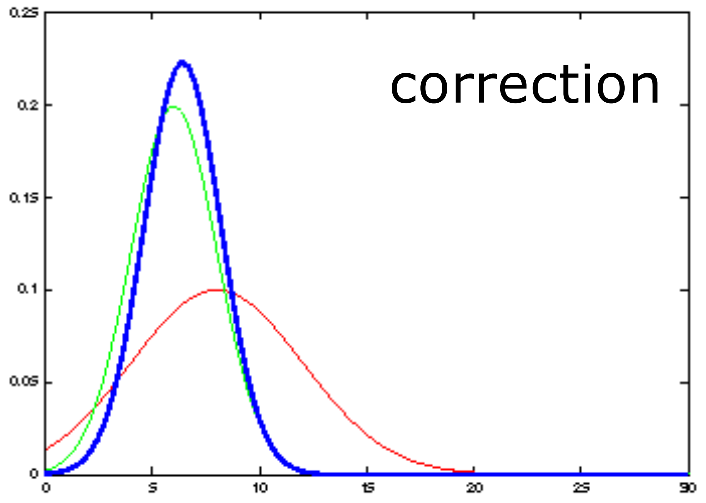
$$z_t = C_t x_t + \delta_t \quad \longleftarrow \quad \text{Observation model}$$

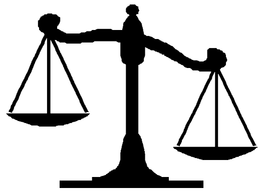
A_t	Matrix (nxn) that describes how the state evolves from t - 1 to t without controls or noise.
B_t	Matrix (nxl) that describes how the control u_t changes the state from t-1 to t.
C_t	Matrix (kxn) that describes how to map the state x_t to an observation z_t.
ε_t, δ_t	Random variables representing 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







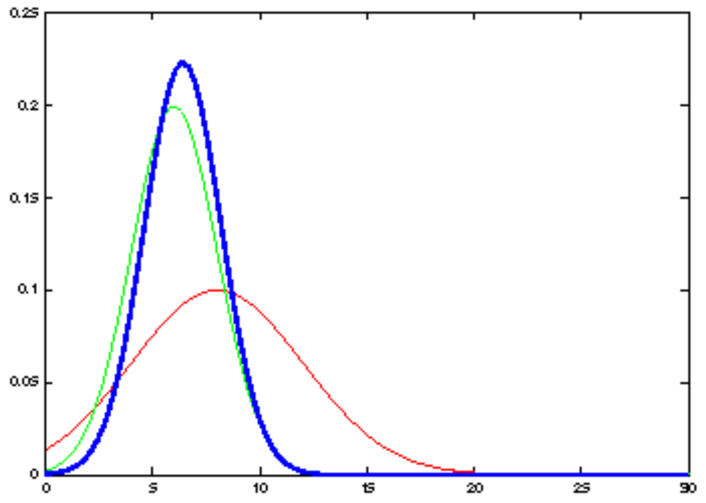


It's a weighted mean!

Kalman Filter Updates in 1D

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

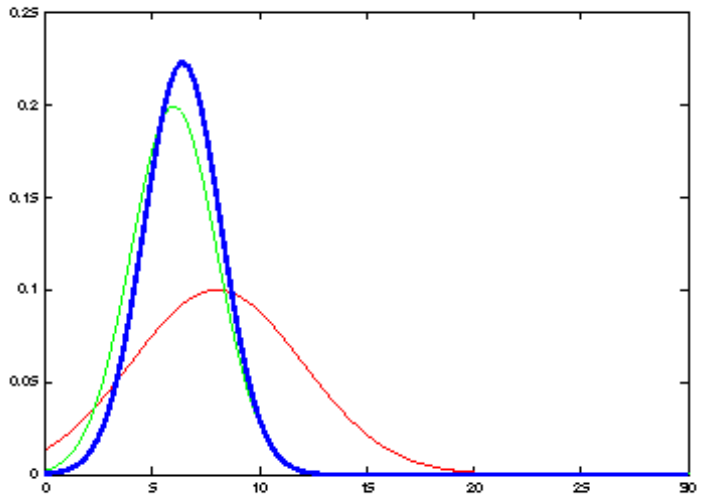
$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases} \quad \text{with} \quad K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

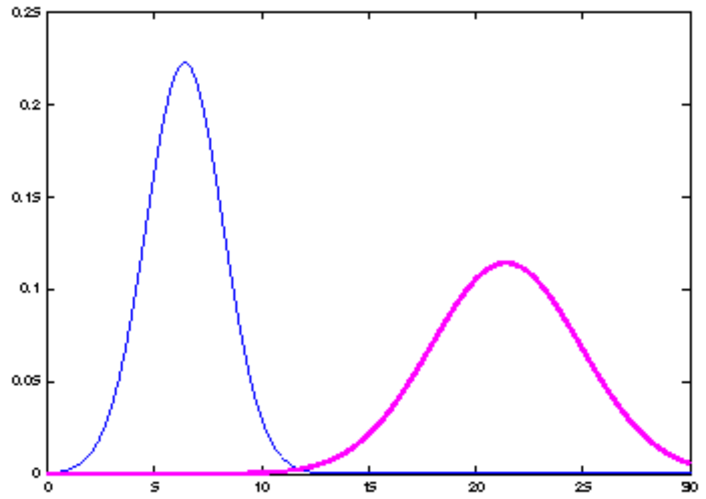


Kalman Filter Updates in 1D

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

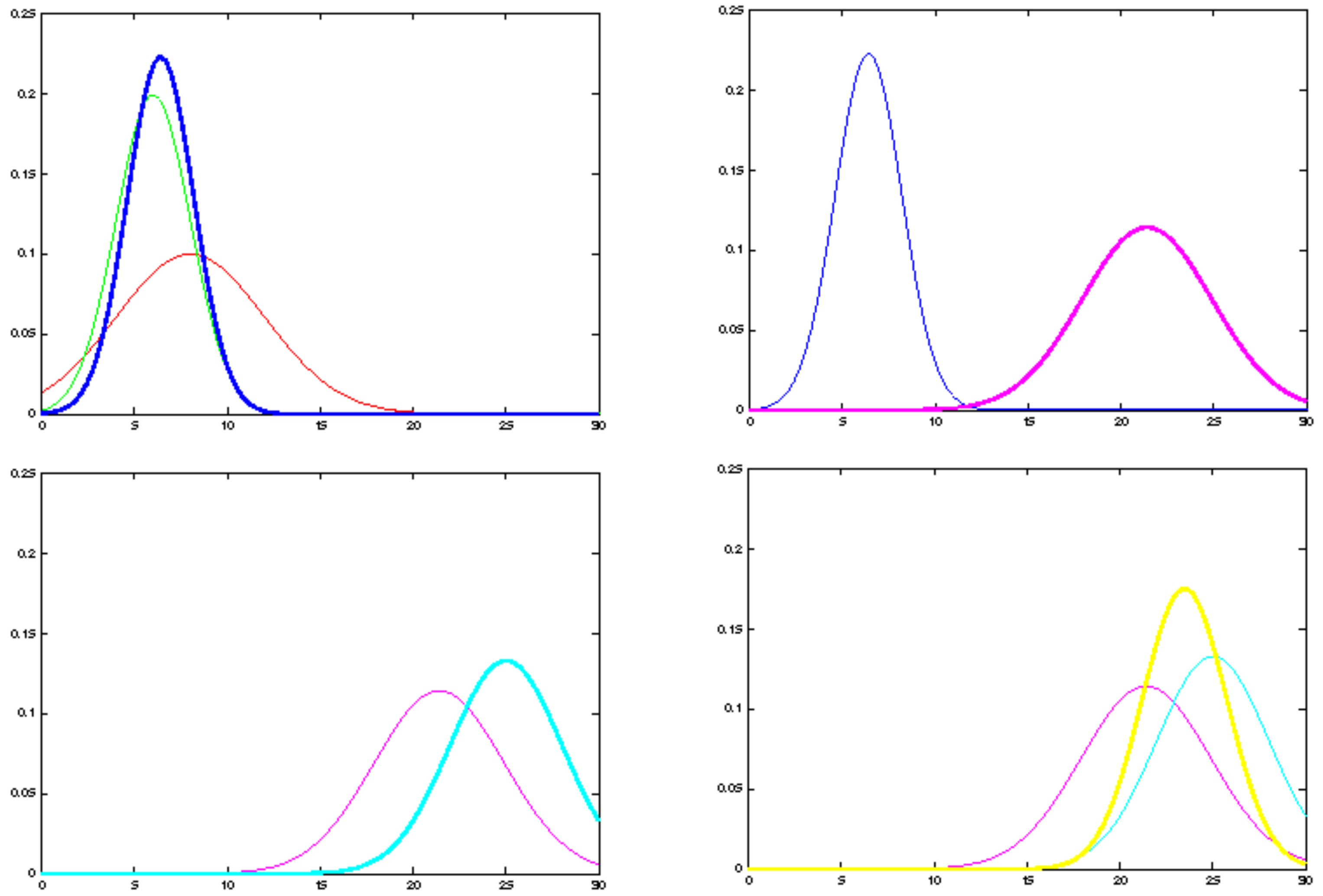
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$





4

Kalman Filter Updates



Linear Gaussian Systems: Initialization

- Initial belief is normally distributed:

$$bel(x_0) = N(x_0; \mu_0, \Sigma_0)$$

Linear Gaussian Systems: Dynamics

- Dynamics are linear function of state and control plus additive noise:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

$$p(x_t \mid u_t, x_{t-1}) = N\left(x_t ; A_t x_{t-1} + B_t u_t, R_t\right)$$

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

$$\Downarrow \qquad \Downarrow$$

$$\sim N\left(x_t ; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1} ; \mu_{t-1}, \Sigma_{t-1}\right)$$

Linear Gaussian Systems: Dynamics

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

$$\Downarrow \qquad\qquad \Downarrow$$

$$\sim N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right) \quad \sim N\left(x_{t-1}; \mu_{t-1}, \Sigma_{t-1}\right)$$

$$\Downarrow$$

$$\overline{bel}(x_t) = \eta \int \exp\left\{-\frac{1}{2}(x_t - A_t x_{t-1} - B_t u_t)^T R_t^{-1} (x_t - A_t x_{t-1} - B_t u_t)\right\}$$

$$\exp\left\{-\frac{1}{2}(x_{t-1} - \mu_{t-1})^T \Sigma_{t-1}^{-1} (x_{t-1} - \mu_{t-1})\right\} dx_{t-1}$$

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

Linear Gaussian Systems: Observations

- Observations are linear function of state plus additive noise:

$$z_t = C_t x_t + \delta_t$$

$$p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)$$

$$bel(x_t) = \eta \quad \underset{\sim N(z_t; C_t x_t, Q_t)}{\underset{\Downarrow}{p(z_t \mid x_t)}} \quad \underset{\sim N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)}{\underset{\Downarrow}{\overline{bel}(x_t)}}$$

Linear Gaussian Systems: Observations

$$bel(x_t) = \eta \quad \underset{\substack{\Downarrow \\ \sim N(z_t; C_t x_t, Q_t) \\ \Downarrow}}{p(z_t \mid x_t)} \quad \underset{\substack{\Downarrow \\ \sim N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)}}{\overline{bel}(x_t)}$$

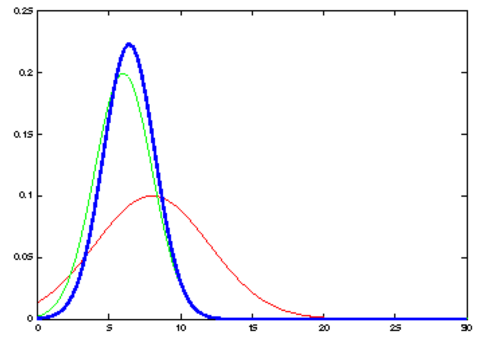
$$bel(x_t) = \eta \exp\left\{-\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\right\} \exp\left\{-\frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\right\}$$

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

Kalman Filter Algorithm

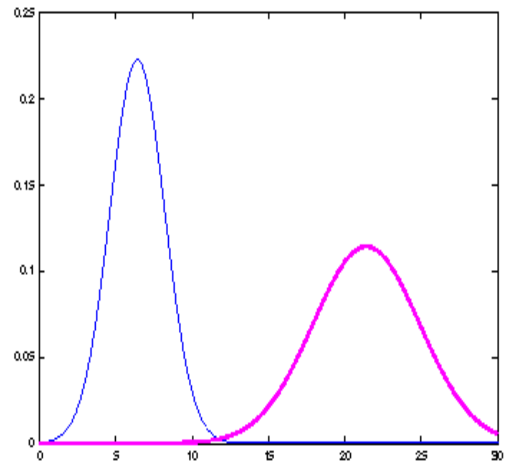
```
1.  Algorithm Kalman_filter( $\mu_{t-1}$, $\Sigma_{t-1}$, $u_t$, $z_t$):

2.  Prediction:
3.      $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4.      $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5.  Correction:
6.      $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$
7.      $\mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t)$
8.      $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$

9.  Return $\mu_t$, $\Sigma_t$
```

The Prediction-Correction-Cycle

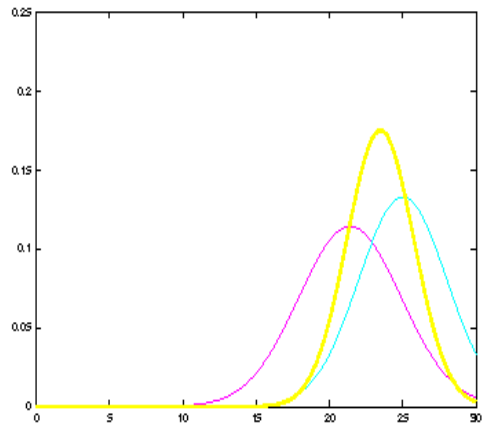


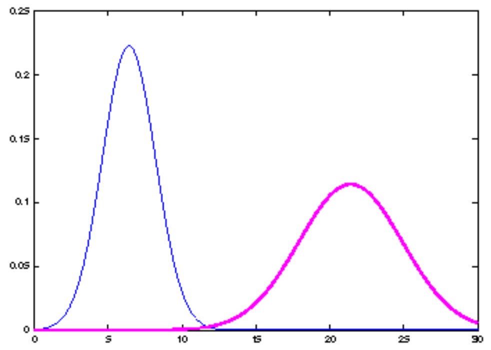
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

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



The Prediction-Correction-Cycle

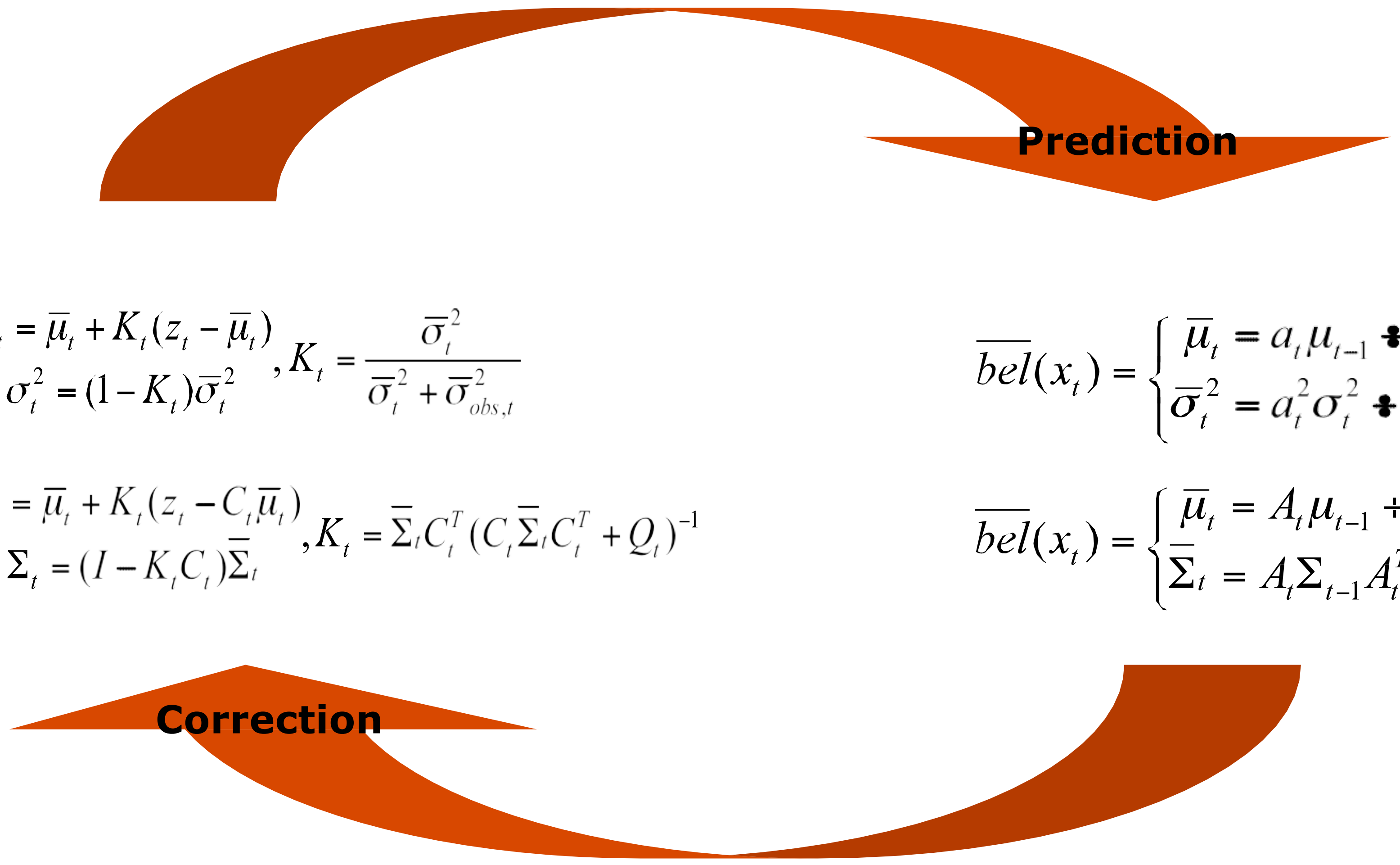




$$bel(x_t) = \begin{cases} \mu_t = \mu_t \quad K_t(z_t \quad \mu_t) \\ \sigma_t^2 = (1 \quad K_t)\sigma_t^2 \end{cases}, K_t = \frac{\overline{\sigma}_t^2}{\overline{\sigma}_t^2 + \overline{\sigma}_{obs,t}^2}$$

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

The Prediction-Correction-Cycle



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

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

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

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

Kalman Filter Summary

- Highly efficient: Polynomial in measurement dimensionality k and state dimensionality n:

  $$O(k^{2.376} + n^2)$$

- Optimal for linear Gaussian systems!

- Most robotics systems are nonlinear!

EXTENDED KALMAN FILTER (EKF)

Nonlinear Dynamic Systems

- Most realistic robotic problems involve nonlinear functions

$$x_t = g(u_t, x_{t-1})$$

$$z_t = h(x_t)$$

- Extended Kalman filter relaxes linearity assumption

Other Error Prop. Techniques

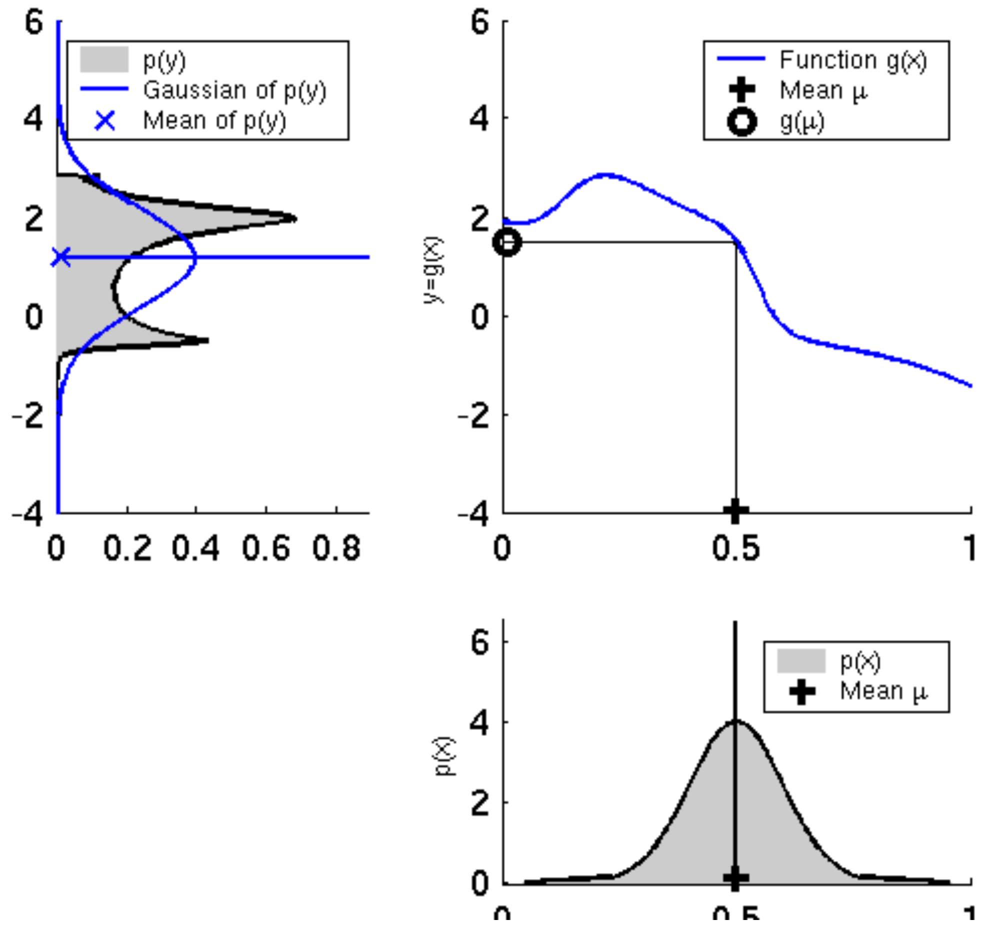
- **Second-Order Error Propagation**

Rarely used (complex expressions)

- **Monte-Carlo**

Non-parametric representation of uncertainties

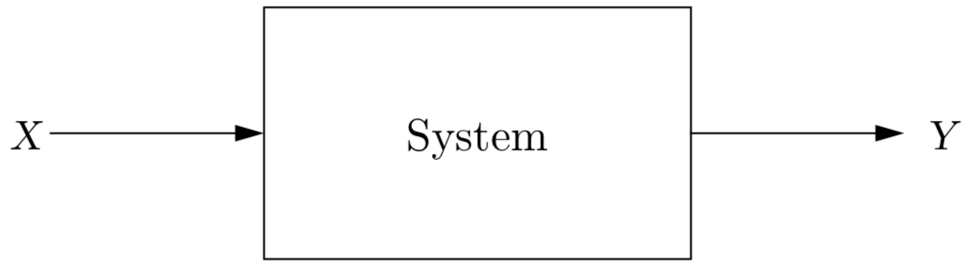
1. Sampling from $p(X)$
2. Propagation of samples
3. Histogramming
4. Normalization



First-Order Error Propagation

X,Y assumed to be Gaussian

$Y = f(X)$



Taylor series expansion

$$Y \approx f(\mu_X) + \left.\frac{\partial f}{\partial X}\right|_{X = \mu_X} (X - \mu_X)$$

Wanted: μ_Y , σ_Y^2

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

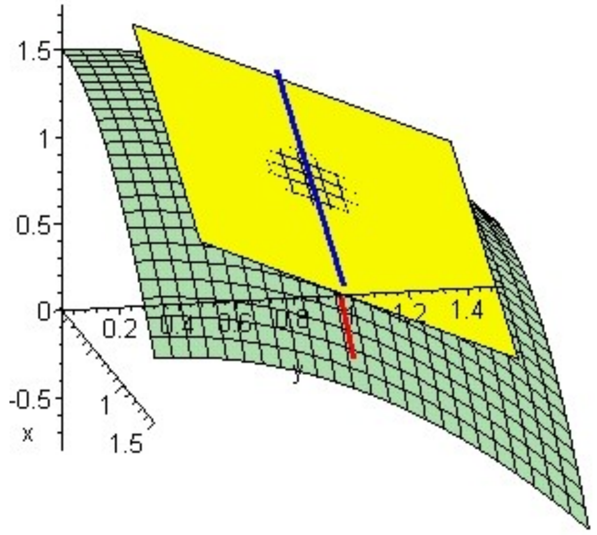
$$\nabla_{\mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial x_1} & \frac{\partial}{\partial x_2} & \cdots & \frac{\partial}{\partial x_n} \end{bmatrix}^T$$

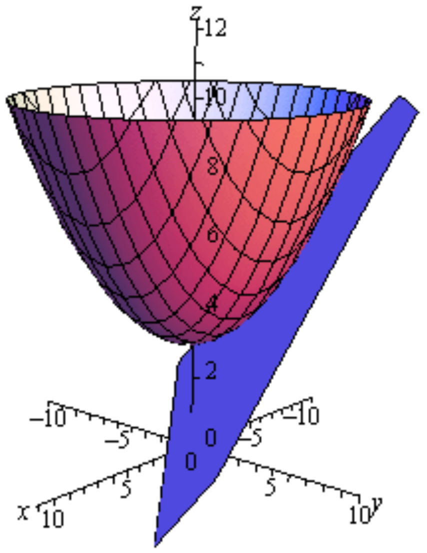
- Then, the **Jacobian matrix** is defined as

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

Jacobian Matrix

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



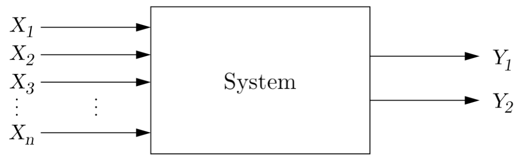


- **Generalizes the gradient** of a scalar valued function
- Heavily used for **first-order error propagation...**

First-Order Error Propagation

Putting things together...

$$C_X = \begin{bmatrix} \sigma_{X_1}^2 & \sigma_{X_1X_2} & \cdots & \sigma_{X_1X_n} \\ \sigma_{X_2X_1} & \sigma_{X_2}^2 & \cdots & \sigma_{X_2X_n} \\ \vdots & \vdots & & \vdots \\ \sigma_{X_nX_1} & \sigma_{X_nX_2} & \cdots & \sigma_{X_n}^2 \end{bmatrix}$$



$$C_Y = \begin{bmatrix} \sigma_{Y_1}^2 & \sigma_{Y_1Y_2} \\ \sigma_{Y_2Y_1} & \sigma_{Y_2}^2 \end{bmatrix}$$

with

$$\sigma_Y^2 = \sum_i \left(\frac{\partial f}{\partial X_i}\right)^2 \sigma_i^2 + \sum_{i \neq j}\sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial f}{\partial X_j}\right)\sigma_{ij}$$

$$\sigma_{YZ} = \sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial g}{\partial X_i}\right)\sigma_i^2 + \sum_{i \neq j}\sum \left(\frac{\partial f}{\partial X_i}\right)\left(\frac{\partial g}{\partial X_j}\right)\sigma_{ij}$$

$\longrightarrow$ "Is there a **compact form?...**"

First-Order Error Propagation

- Input covariance matrix C_X
- Jacobian matrix F_X

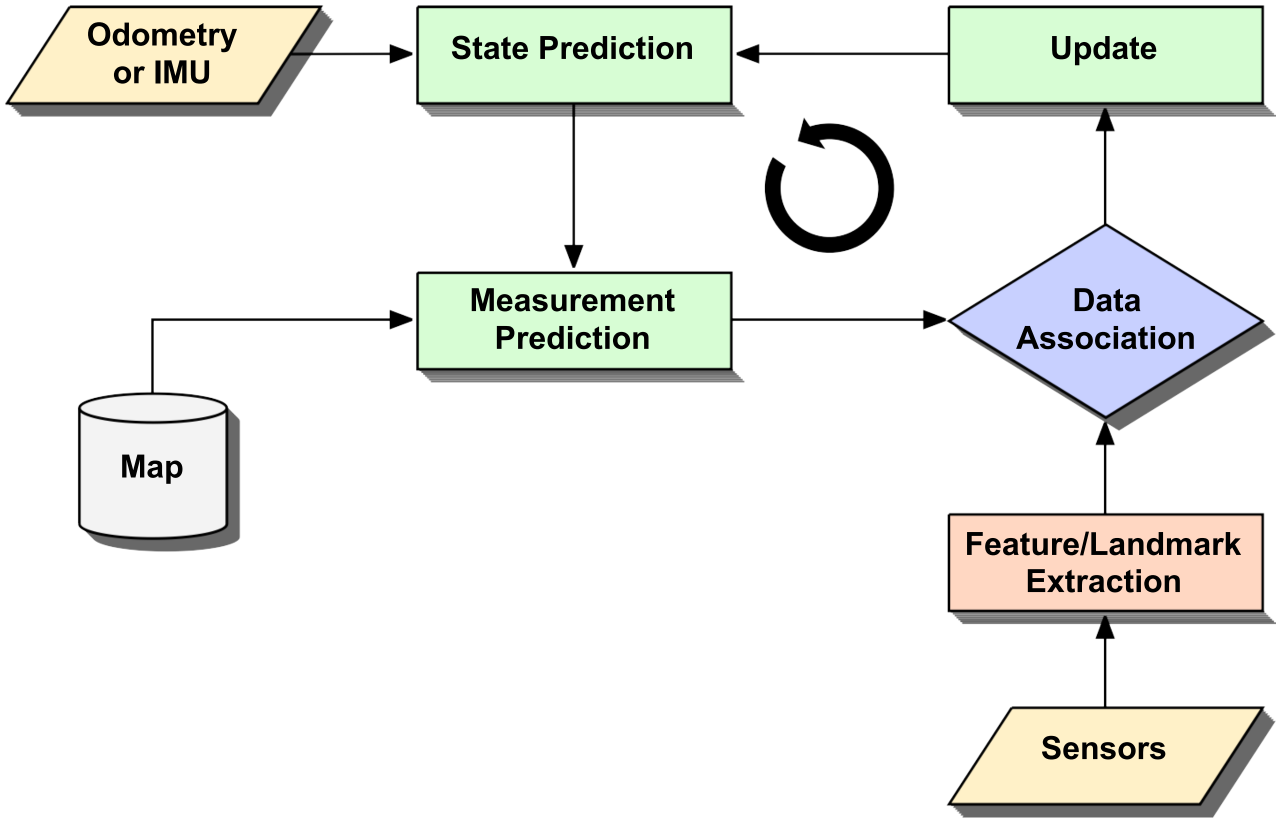
the **Error Propagation Law**

$$\boxed{C_Y = F_X C_X F_X^T}$$

computes the output covariance matrix C_Y

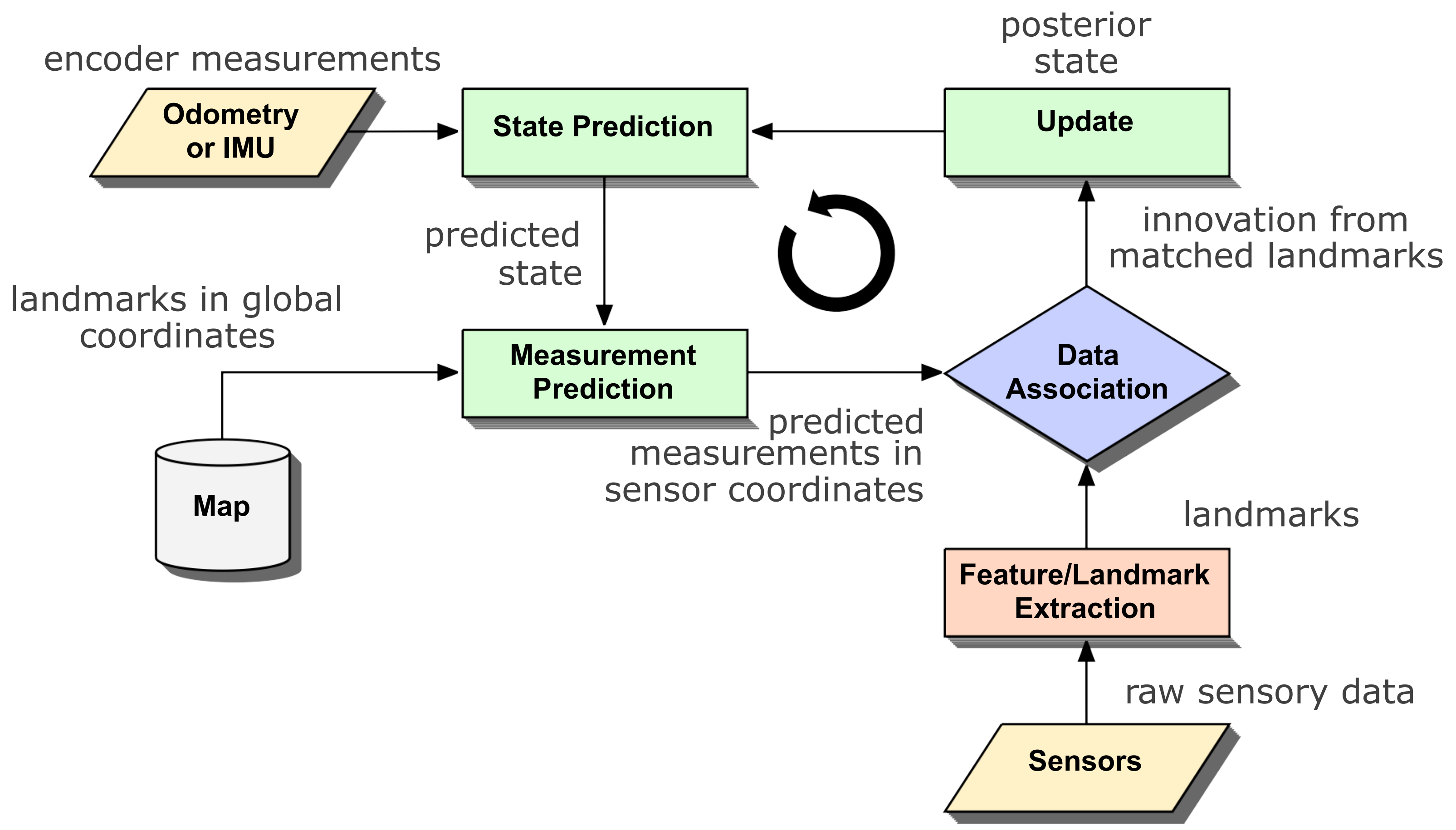
Landmark-based Localization

EKF Localization: Basic Cycle



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

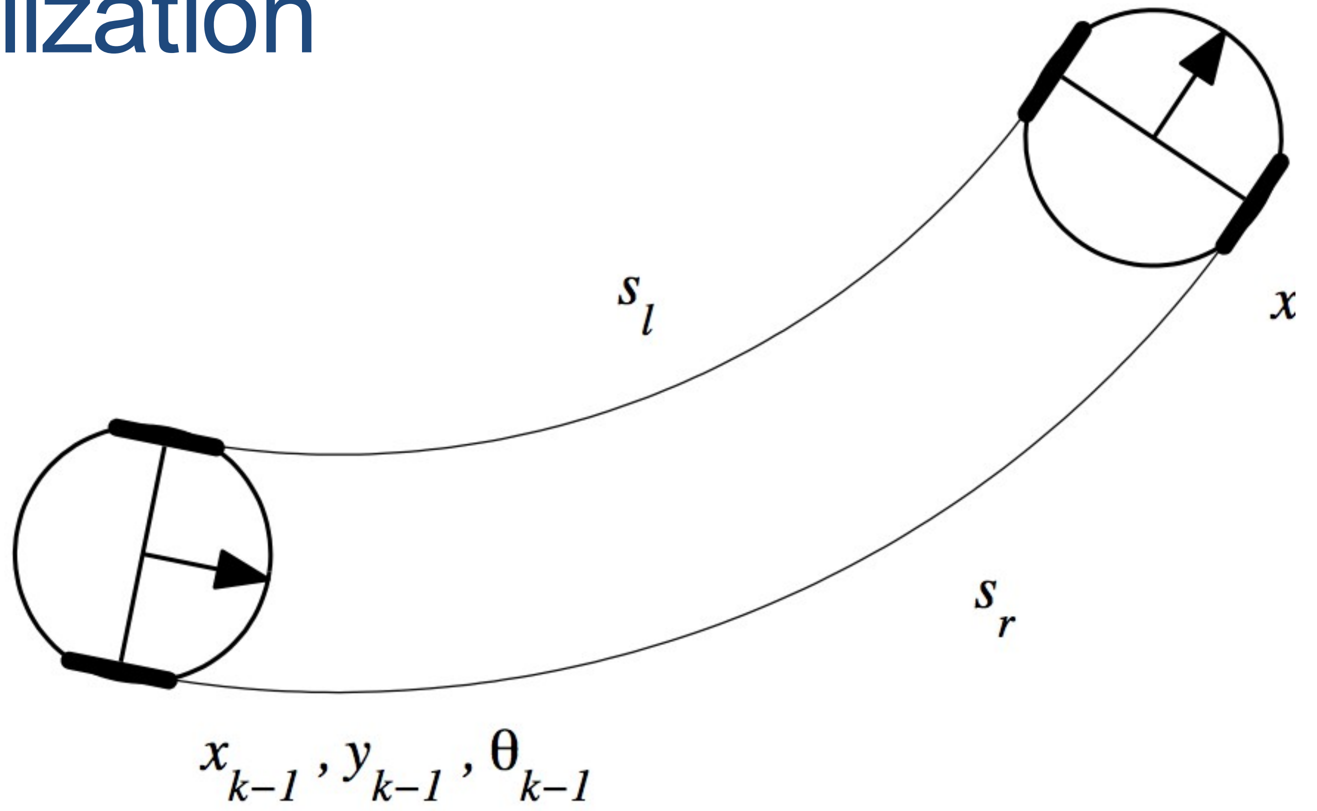
$$\mathbf{u}_k = (s_l \;\; s_r)^T \qquad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

Error model: linear growth

$$\begin{aligned} \sigma_l &= k_l\,|s_l| \\ \sigma_r &= k_r\,|s_r| \end{aligned}$$

Nonlinear process model f :

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



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

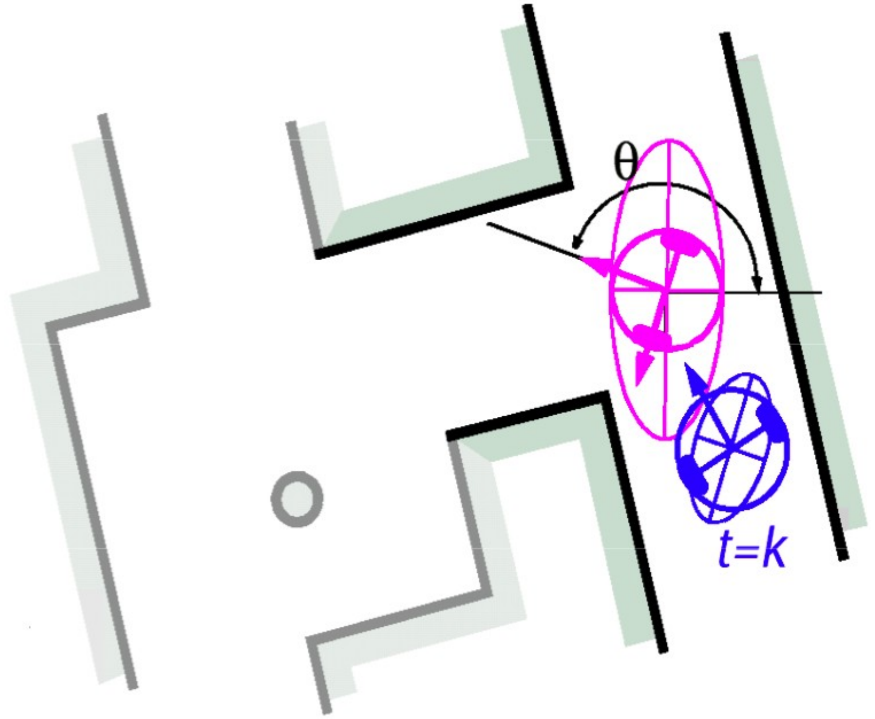
$$\mathbf{u}_k = (s_l \;\; s_r)^T \qquad U_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

Error model: linear growth

$$\begin{aligned} \sigma_l &= k_l\,|s_l| \\ \sigma_r &= k_r\,|s_r| \end{aligned}$$

Nonlinear process model f:

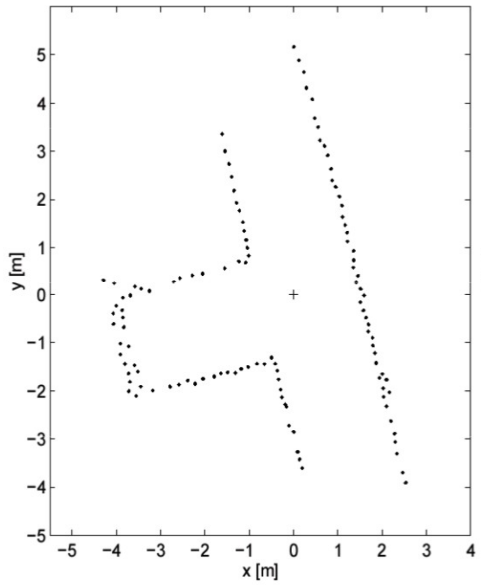
$$\mathbf{x}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(-\sin\theta_{k-1} + \sin(\theta_{k-1} + \frac{s_r-s_l}{b})\right) \\ \frac{b}{2}\frac{s_l+s_r}{s_r-s_l}\left(\cos\theta_{k-1} - \cos(\theta_{k-1} + \frac{s_r-s_l}{b})\right)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



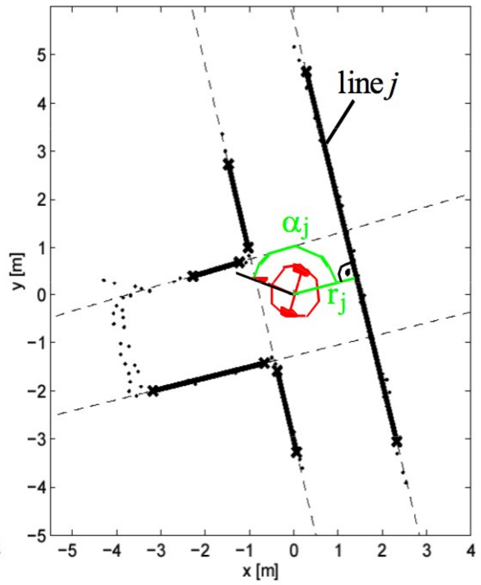
Landmark-based Localization

Landmark Extraction (Observation)

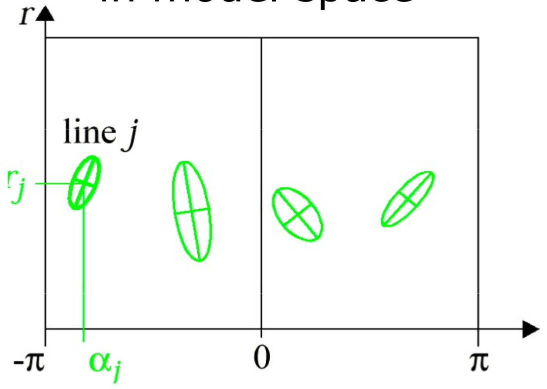
Raw laser range data



Extracted lines



Extracted lines in model space



Hessian line model

$$x\ cos(\alpha) + y\ sin(\alpha) - r = 0$$

$$\mathbf{z}_k = \begin{bmatrix} \alpha \\ r \end{bmatrix}$$

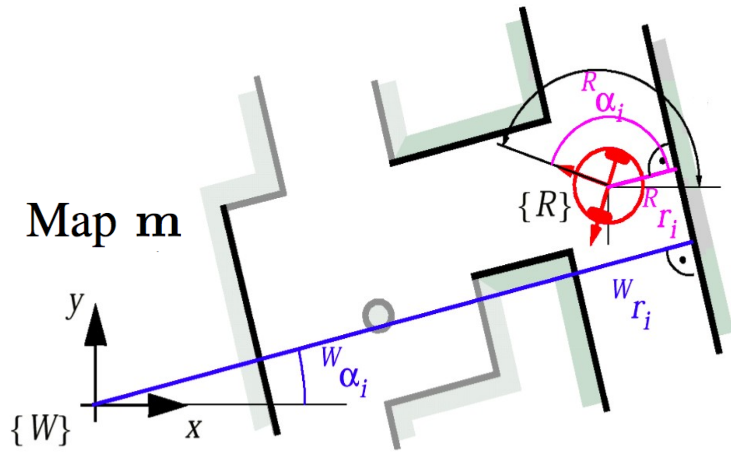
$$R_k = \begin{bmatrix} \sigma_\alpha^2 & \sigma_{\alpha r} \\ \sigma_{r\alpha} & \sigma_r^2 \end{bmatrix}$$

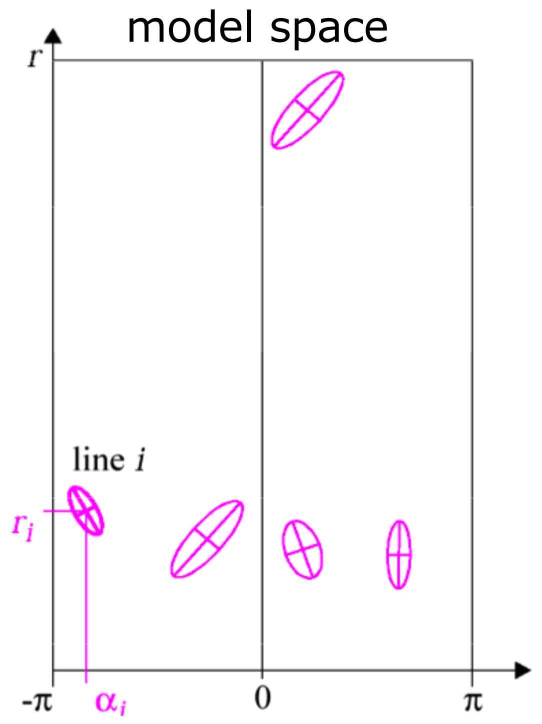
Landmark-based Localization

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

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





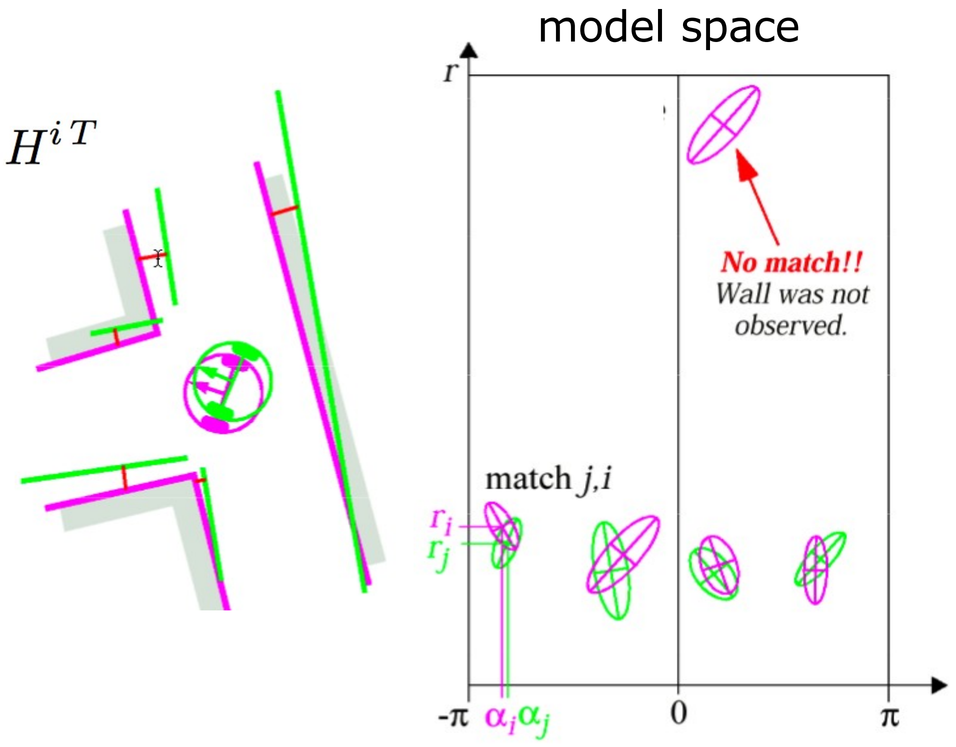
Landmark-based Localization

Data Association (Matching)

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

$$\begin{aligned} \nu_k^{ij} &= \mathbf{z}_k^j - \hat{\mathbf{z}}_k^i \\ S_k^{ij} &= R_k^j + H^i \, \hat{C}_k \, H^{i\,T} \end{aligned}$$

- Innovation and innovation covariance ν_k^{ij}



Green: observation

Magenta: measurement prediction

Landmark-based Localization

Update

- Kalman gain

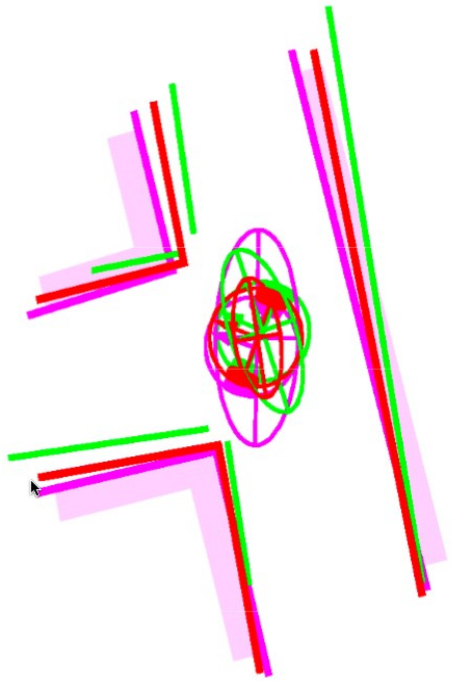
$$K_k = \hat{C}_k \, H^T S_k^{-1}$$

- 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

Material

- Kalman, R. E. 1960. “A New Approach to Linear Filtering and Prediction Problems”, Transaction of the ASME--Journal of Basic Engineering, pp. 35-45 (March 1960).
- Welch, G and Bishop, G. 2001. “An introduction to the Kalman Filter”, http://www.cs.unc.edu/~welch/kalman/
- Thrun, S. and Burgard, W. and Fox, D. “Probabilistic Robotics” MIT Press 2006

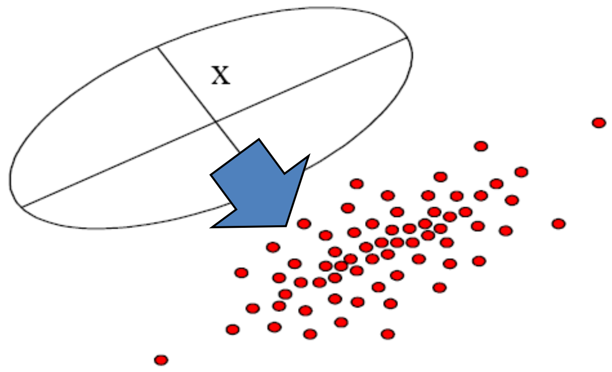
PARTICLE FILTER

Following Material:

- Wolfram Burgard, University of Freiburg

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.



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

- 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.

Motivation

- Particle filters are a way to efficiently represent non-Gaussian distribution
- Basic principle
  - Set of state hypotheses (“particles”)
  - Survival-of-the-fittest

Mathematical Description

- Set of weighted samples

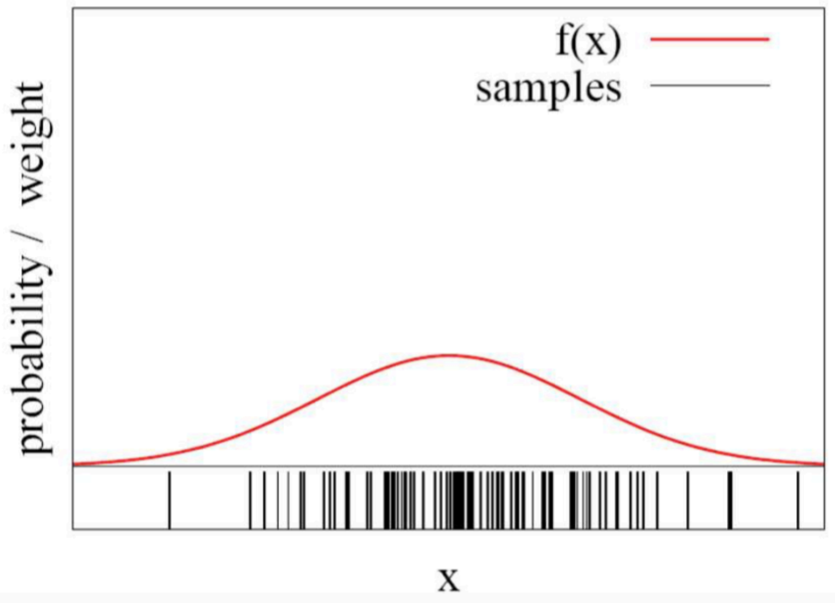
$$S = \{< \underbrace{s^{[i]}}_{\text{State hypothesis}}, \underbrace{w^{[i]}}_{\text{Importance Weight}} > \mid i = 1, \dots, N\}$$

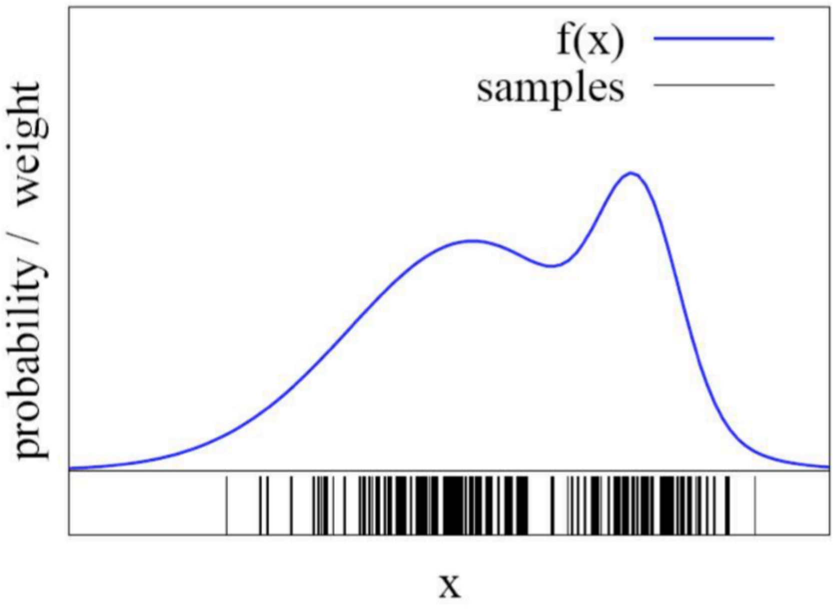
- The samples represent the posterior

$$p(x) = \sum_{i=1}^{N} w_i \cdot \delta_{s^{[i]}}(x)$$

Function Approximation

- Particle sets can be used to approximate functions

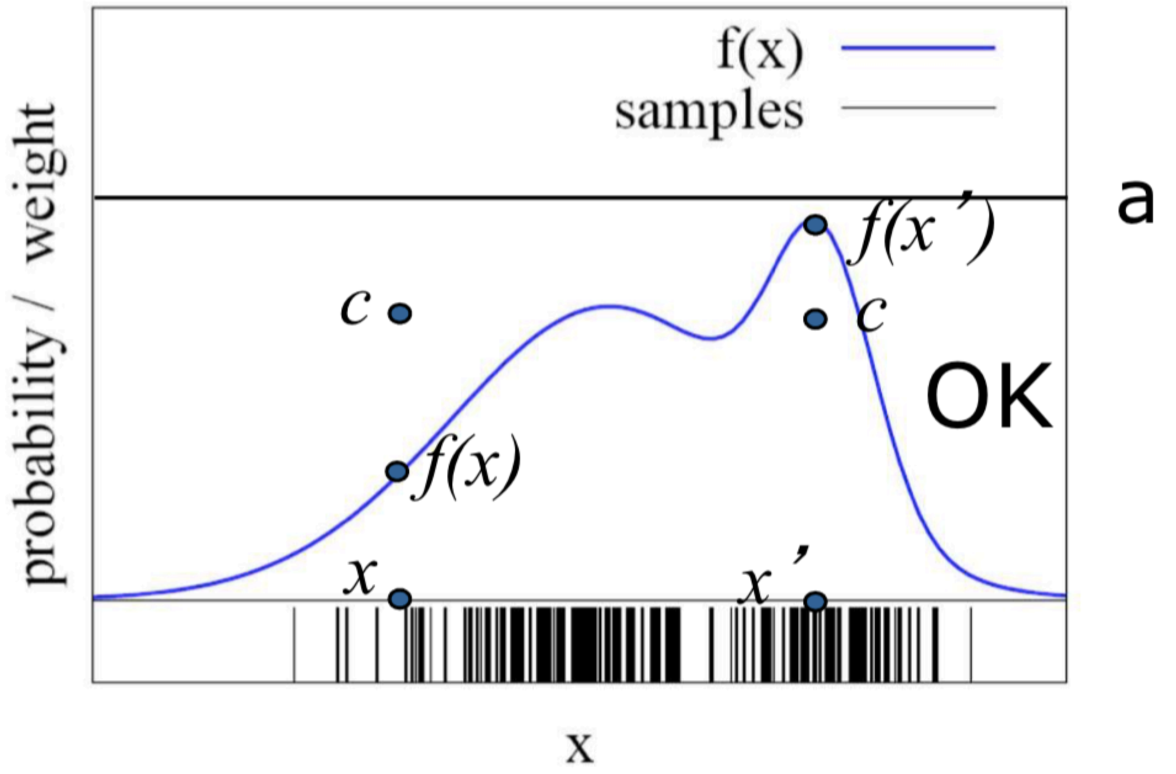




- The more particles fall into an interval, the higher the probability of that interval
- How to draw samples from a function/distribution?

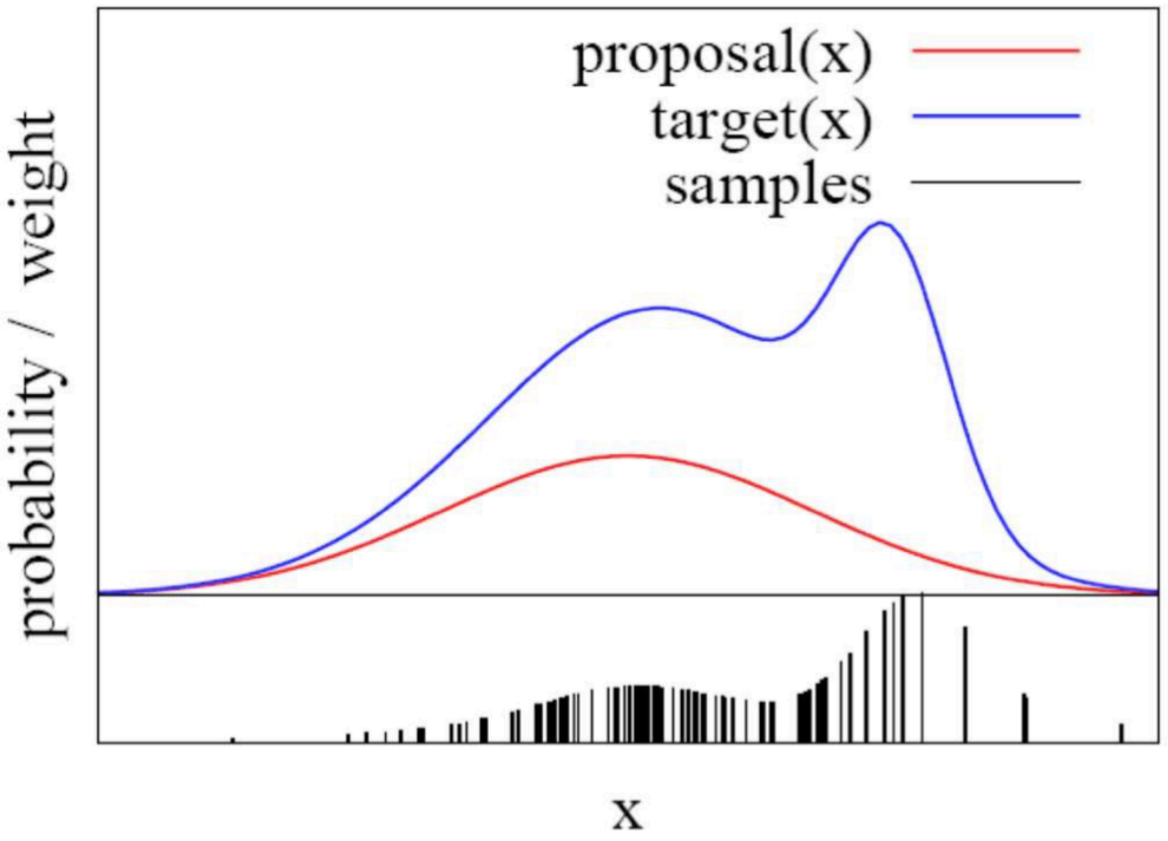
Rejection Sampling

- Let us assume that $f(x) < a$ for all x
- Sample x from a uniform distribution
- Sample c from $[0, a]$
- if $f(x) > c$ keep the sample
- otherwise reject the sample

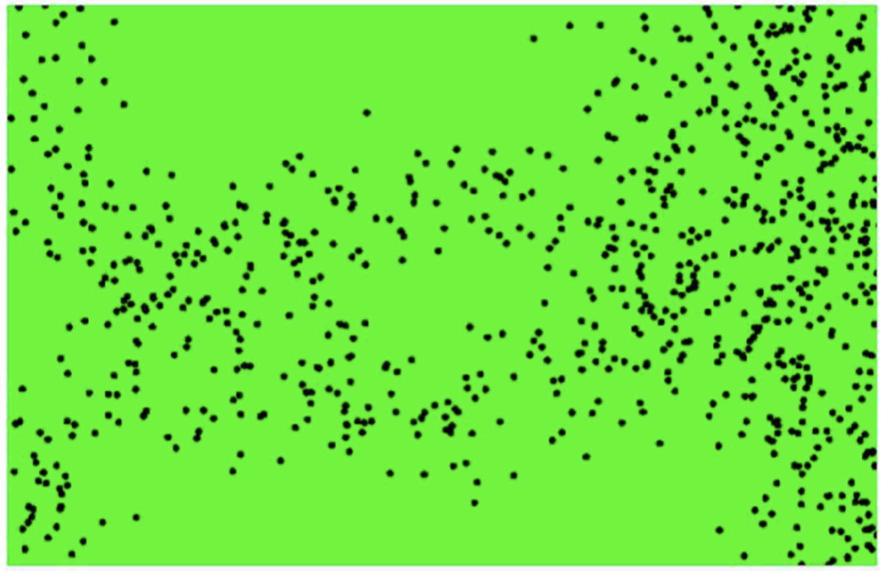


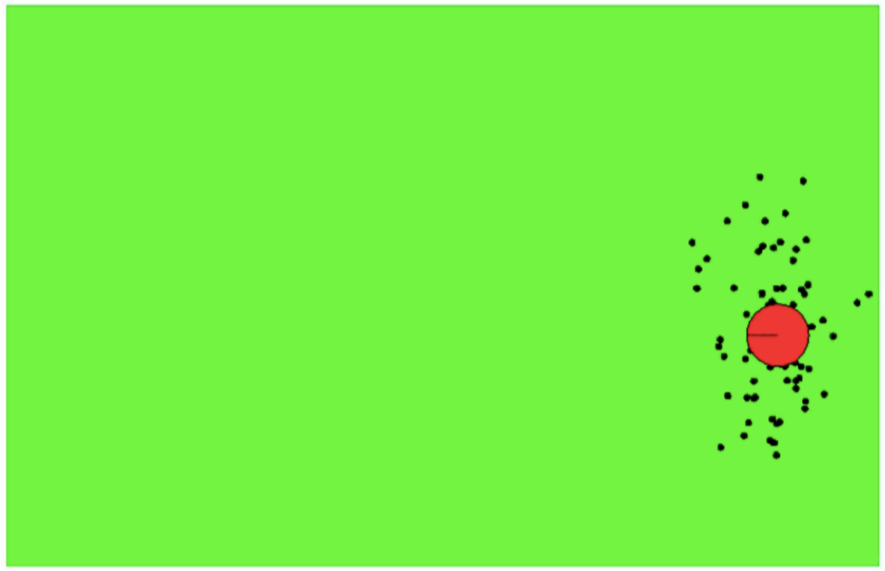
Importance Sampling Principle

- We can even use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the “differences between g and f ”
- $w = f/g$
- f is called target
- g is called proposal
- Pre-condition:
  - $f(x) > 0 \rightarrow g(x) > 0$



Importance Sampling with Resampling


Weighted Samples


After Resampling

Particle Filter Algorithm

- Sample the next generation for particles using the proposal distribution
- Compute the importance weights :

$$weight = target\ distribution\ /\ proposal\ distribution$$

- Resampling: "Replace unlikely samples by more likely ones"

Particle Filter Algorithm

```
1.  Algorithm particle_filter( $S_{t-1}$, $u_t$, $z_t$):
2.  $S_t = \emptyset, \eta = 0$
3.  For $i = 1, \dots, n$                                              Generate new samples
4.          Sample index $j(i)$ from the discrete distribution given by $w_{t-1}$
5.          Sample $x_t^i$ from $p(x_t | x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and $u_t$
6.          $w_t^i = p(z_t | x_t^i)$                                   Compute importance weight
7.          $\eta = \eta + w_t^i$                                      Update normalization factor
8.          $S_t = S_t \cup \{< x_t^i, w_t^i >\}$                      Add to new particle set
9.  For $i = 1, \dots, n$
10.         $w_t^i = w_t^i / \eta$                                     Normalize weights
```

Particle Filter Algorithm

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

Draw x_{t-1}^i from $bel(x_{t-1})$

Draw x_t^i from $p(x_t|x_{t-1}, u_t)$

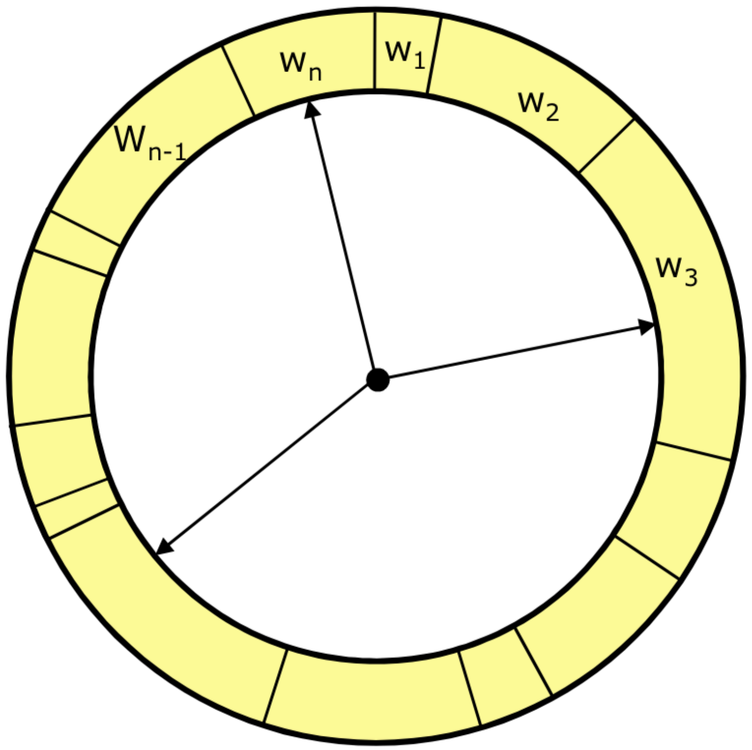
Importance factor for x_t^i

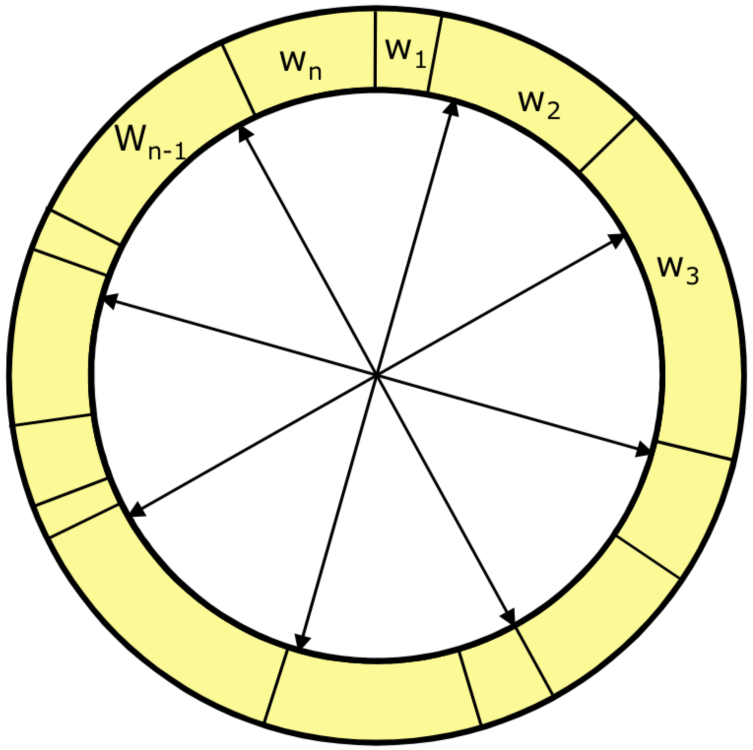
$$w_t^i = \frac{target\ distribution}{proposal\ distribution} = \frac{\eta p(z_t|x_t) p(x_t|x_{t-1}, u_t) bel(x_{t-1})}{p(x_t|x_{t-1}, u_t) bel(x_{t-1})} \propto \eta p(z_t|x_t)$$

Resampling

- Given: Set S of weighted samples.
- Wanted : Random sample, where the probability of drawing x_i is given by w_i.
- Typically done n times with replacement to generate new sample set S'.

Resampling





- Roulette wheel
- Binary search, $n \log n$

- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Mobile Robot Localization

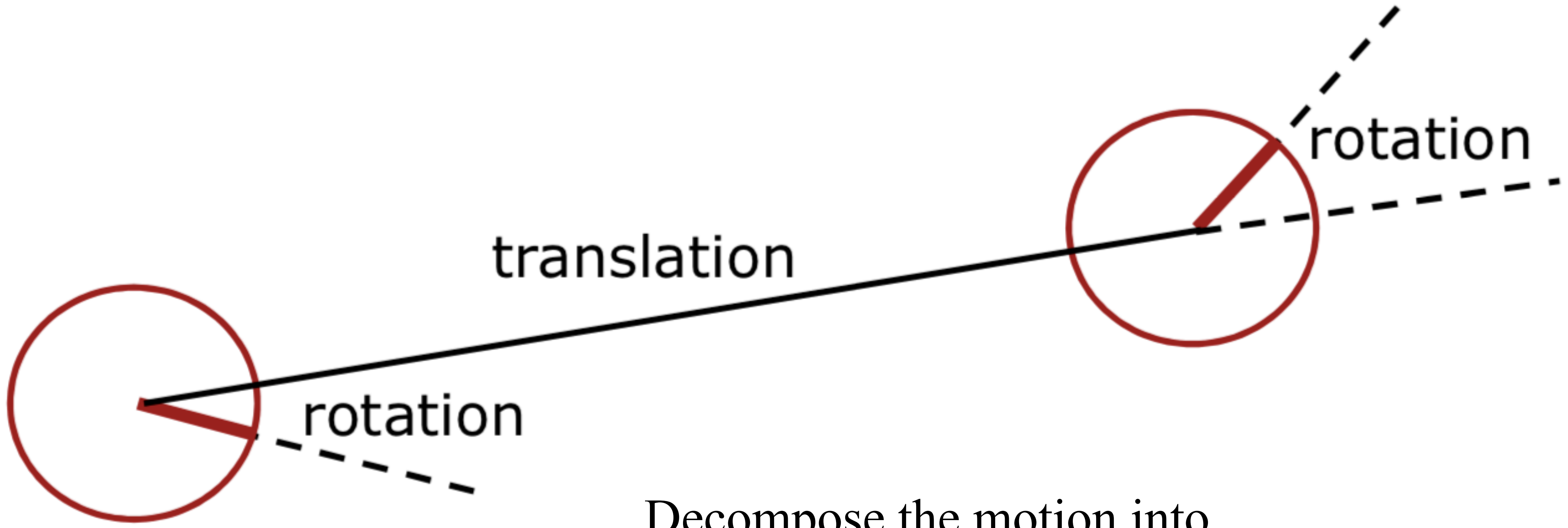
- Each particle is a potential pose of the robot
- Proposal distribution is the motion model of the robot (prediction step)
- The observation model is used to compute the importance weight (correction step)

Motion Model Reminder

End Pose
Start Pose

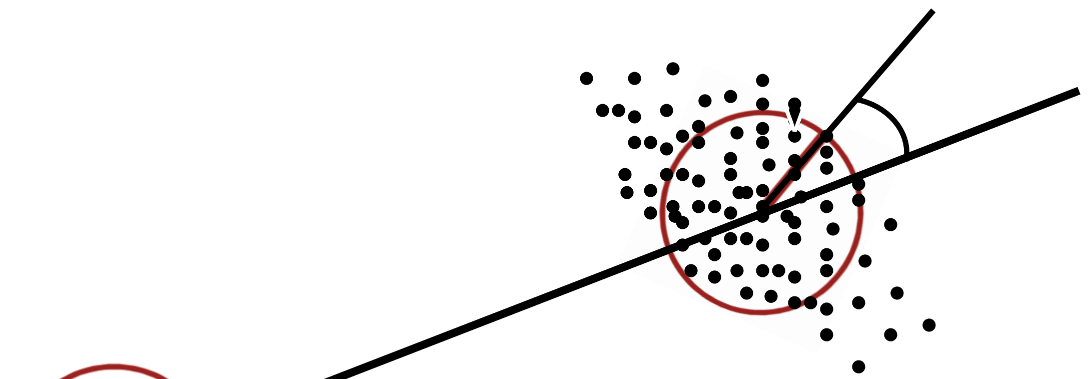
According to the estimated motion

Motion Model Reminder



Decompose the motion into
- Traveled distance
- Start rotation
- End rotation

Motion Model Reminder



- Uncertainty in the translation of the robot: Gaussian over the traveled distance
- Uncertainty in the rotation of the robot: Gaussians over start and end rotation
- For each particle, draw a new pose by sampling from these three individual normal distributions

Mobile Robot Localization Using Particle Filters (1)

- Each particle is a potential pose of the robot
- The set of weighted particles approximates the posterior belief about the robot's pose (target distribution)

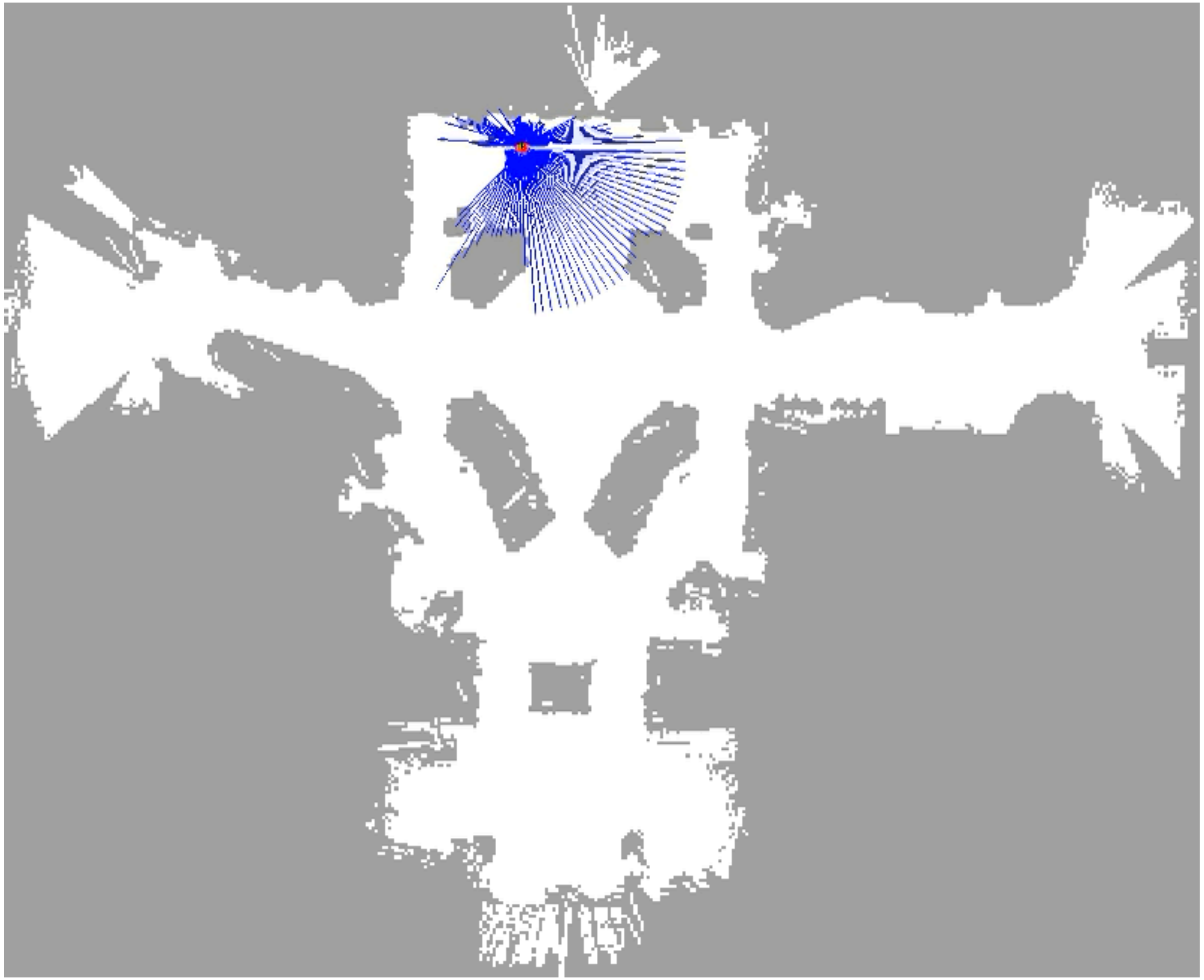
Mobile Robot Localization Using Particle Filters (2)

- Particles are drawn from the motion model (proposal distribution)
- Particles are weighted according to the observation model (sensor model)
- Particles are resampled according to the particle weights

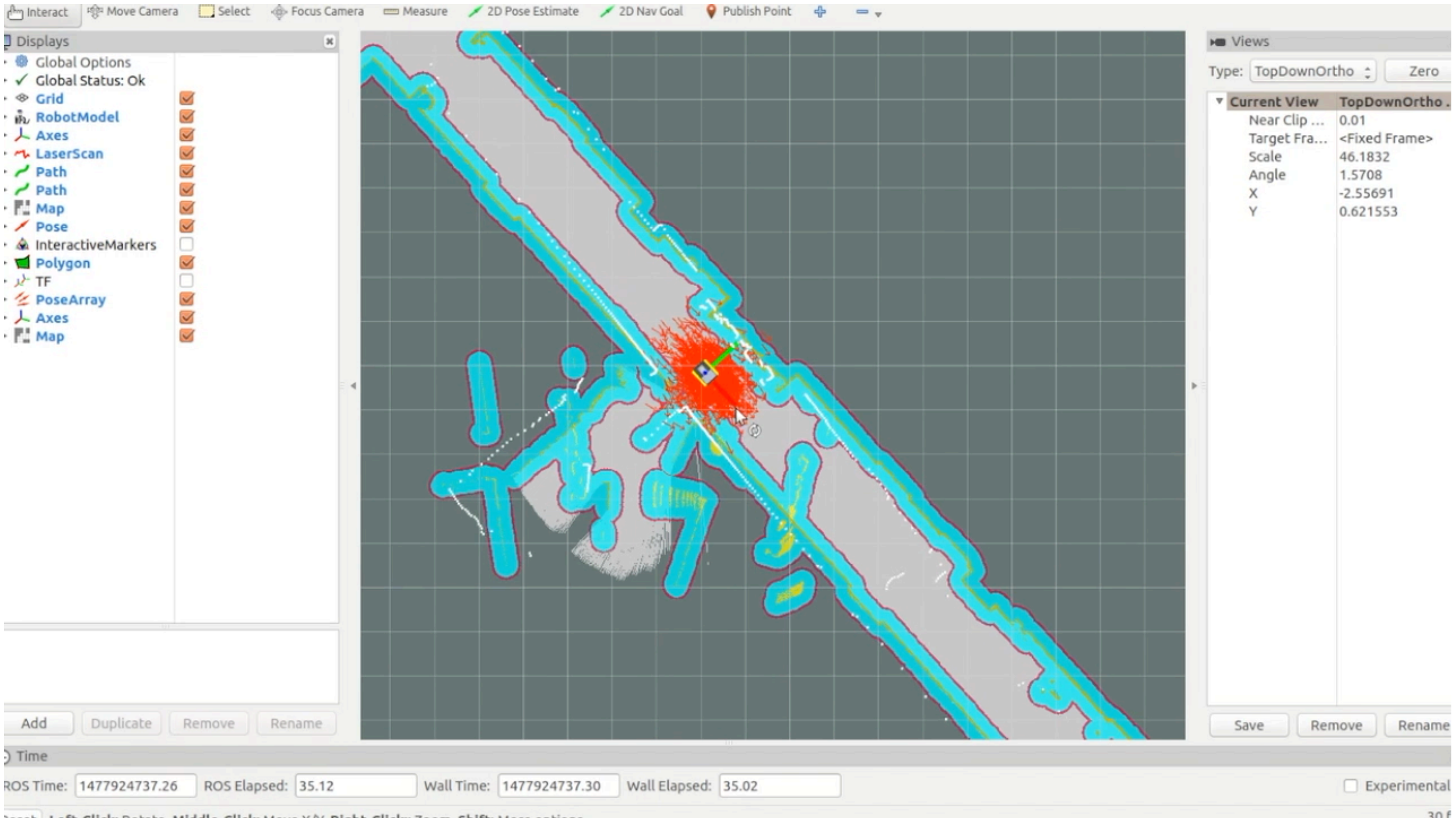
Mobile Robot Localization Using Particle Filters (3)

- Why is resampling needed?
  - We only have a finite number of particles
  - Without resampling: The filter is likely to loose track of the “good” hypotheses
  - Resampling ensures that particles stay in the meaningful area of the state space

MCL & Robot Kidnapping



AMCL in ROS – play with it in MoManTu!



Global Options	
Global Status: Ok	
Grid	✓
RobotModel	✓
Axes	✓
LaserScan	✓
Path	✓
Path	✓
Map	✓
Pose	✓
InteractiveMarkers	☐
Polygon	✓
TF	☐
PoseArray	✓
Axes	✓
Map	✓

Type: TopDownOrtho

Current View	TopDownOrtho
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Scale	46.1832
Angle	1.5708
X	-2.55691
Y	0.621553

ROS Time: 1477924737.26 ROS Elapsed: 35.12 Wall Time: 1477924737.30 Wall Elapsed: 35.02 ☐ Experimental

SLAM Using Particle Filters – Grid-based SLAM

- Can we solve the SLAM problem if no pre- defined landmarks are available?
- Can we use the ideas of FastSLAM to build grid maps?
- As with landmarks, the map depends on the poses of the robot during data acquisition
- If the poses are known, grid-based mapping is easy ("mapping with known poses")

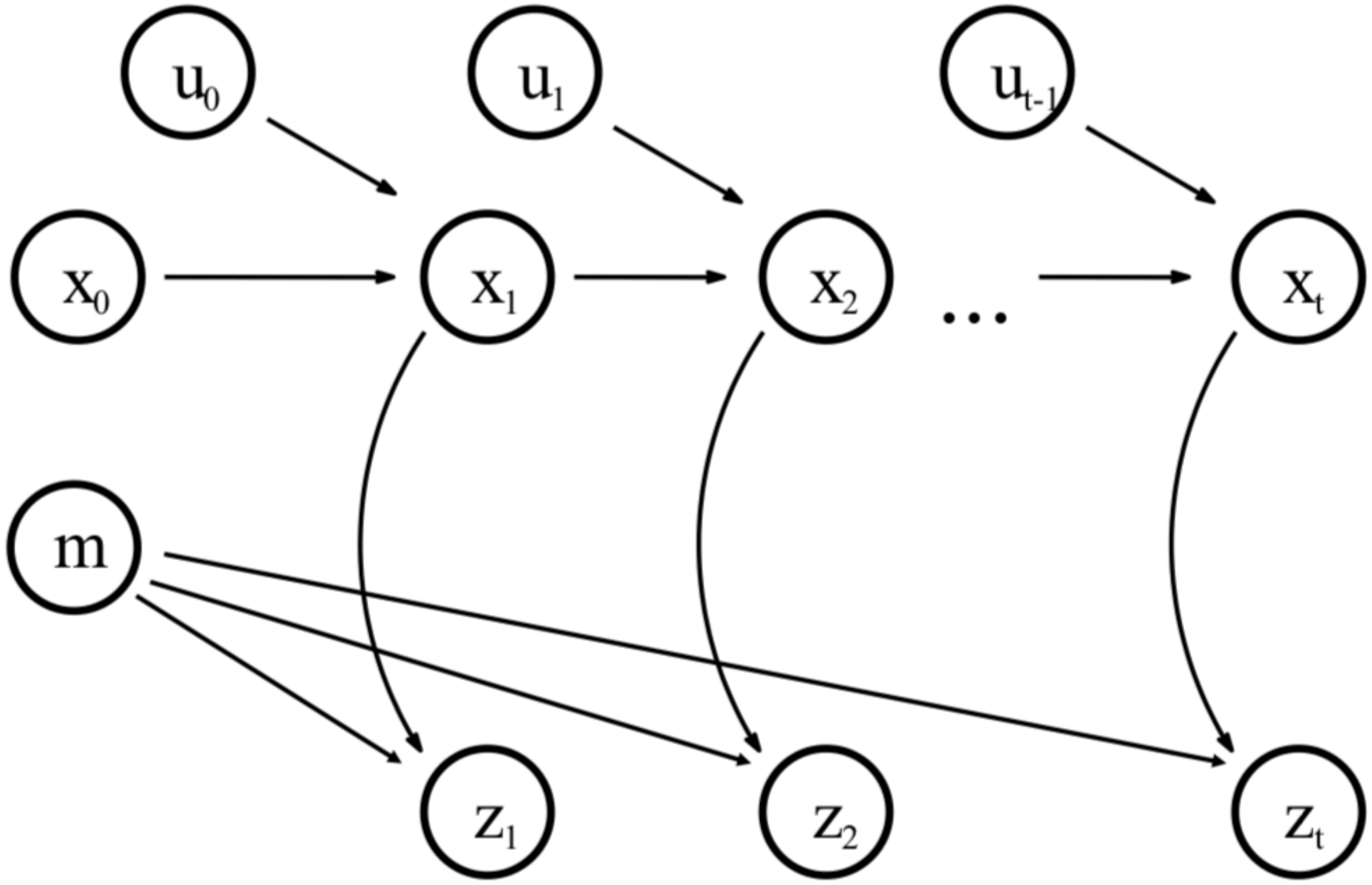
Rao-Blackwellization

- $$\underbrace{p(\overset{\text{Poses}}{x_{1:t}}, \overset{\text{Map}}{m} | \overset{\text{Observations}}{z_{1:t}}, \overset{\text{Movements}}{u_{0:t-1}})}_{\text{SLAM posterior}} = \underbrace{p(x_{1:t}, | z_{1:t}, u_{0:t-1})}_{\text{Robot path posterior}} \cdot \underbrace{p(m | x_{1:t}, z_{1:t})}_{\text{Mapping with known poses}}$$

Rao-Blackwellization

- $p(x_{1:t}, m | z_{1:t}, u_{0:t-1}) = \underbrace{p(x_{1:t}, | z_{1:t}, u_{0:t-1})}_{\text{This is Localization, use MCL}} \cdot \underbrace{p(m | x_{1:t}, z_{1:t})}_{\text{Use the pose estimate from the MCL and apply mapping with known poses}}$

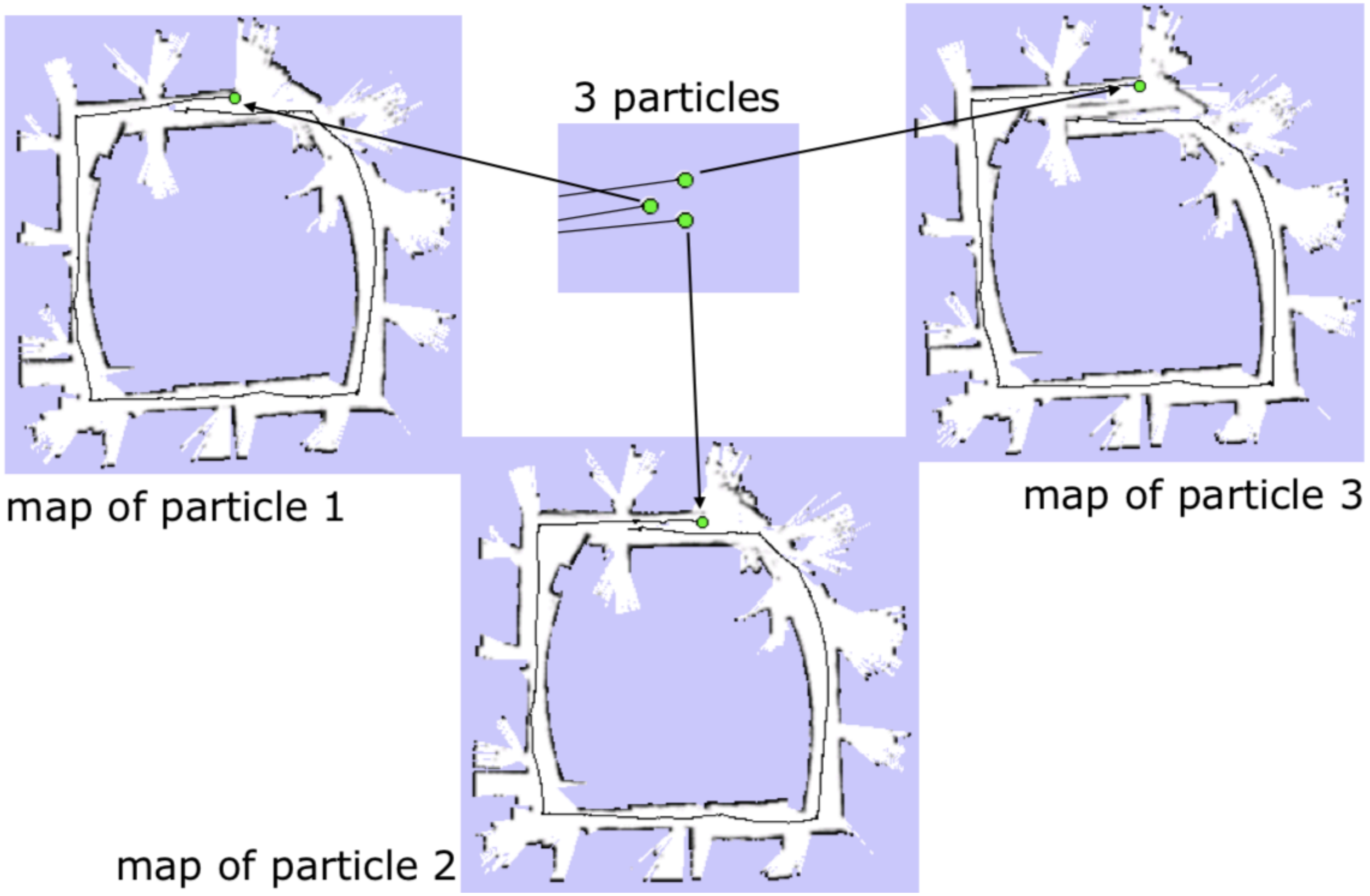
A Graphical Model of Mapping with Rao-Blackwellized PFs



Mapping with Rao- Blackwellized Particle Filters

- Each particle represents a possible trajectory of the robot
- Each particle
  - maintains its own map and
  - updates it upon "mapping with known poses"
- Each particle survives with a probability proportional to the likelihood of the observations relative to its own map

Particle Filter Example



Problem

- Each map is quite big in case of grid maps
- Each particle maintains its own map, therefore, one needs to keep the number of particles small
- **Solution:**
  Compute better proposal distributions!
- **Idea:**
  Improve the pose estimate before applying the particle filter

Pose Correction Using Scan Matching

- Maximize the likelihood of the i-th pose and map relative to the $(i-1)$-th pose and map

$$\hat{x}_t = argmax_{x_t}\{p(z_t|x_t, \hat{m}_{t-1}) \cdot p(x_t|u_{t-1}, \hat{x}_{t-1})\}$$

current measurement

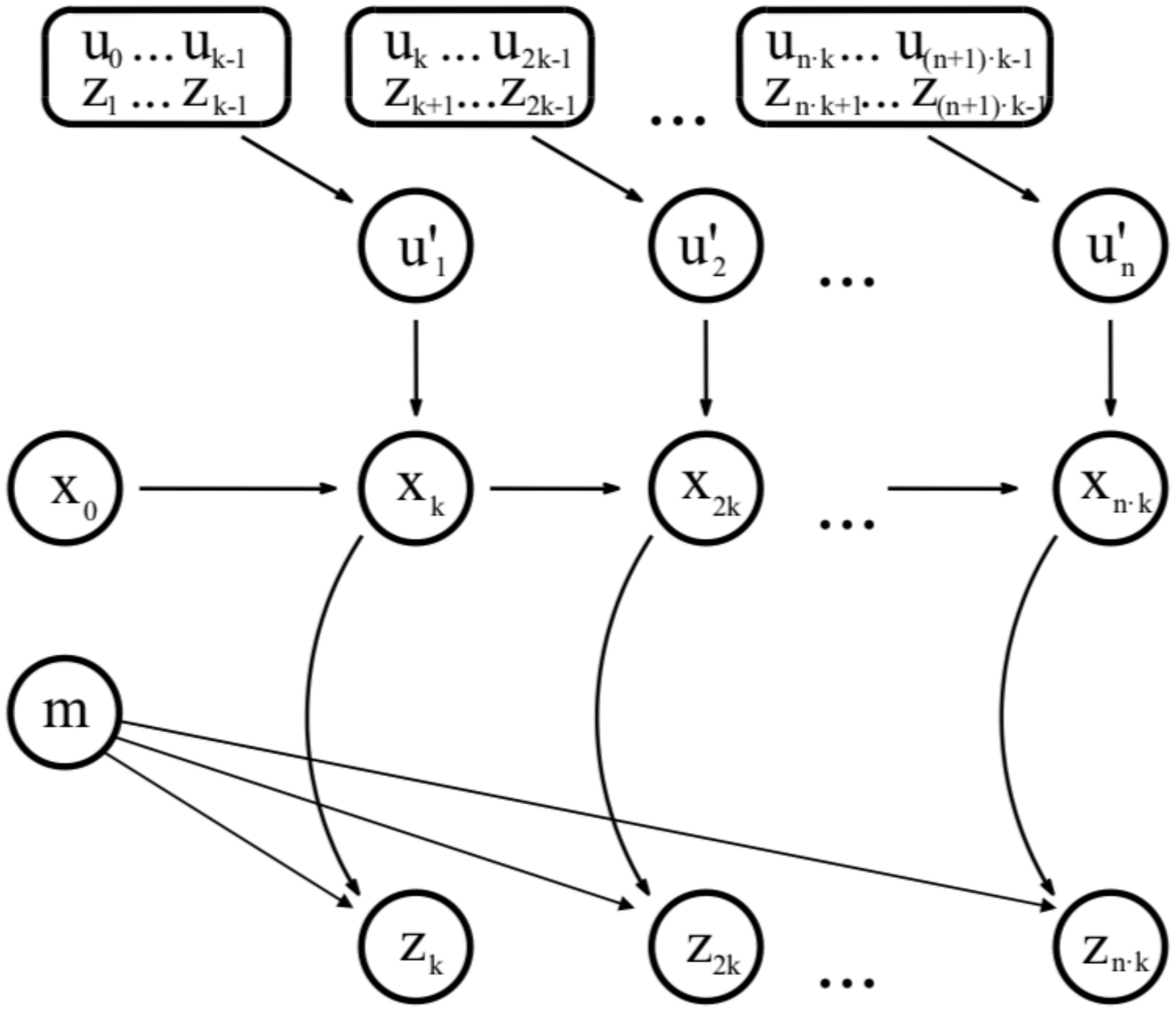
robot motion

map constructed so far

FastSLAM with Improved Odometry

- Scan-matching provides a locally consistent pose correction
- Pre-correct short odometry sequences using scan-matching and use them as input to FastSLAM
- Fewer particles are needed, since the error in the input is smaller

Graphical Model for Mapping with Improved Odometry

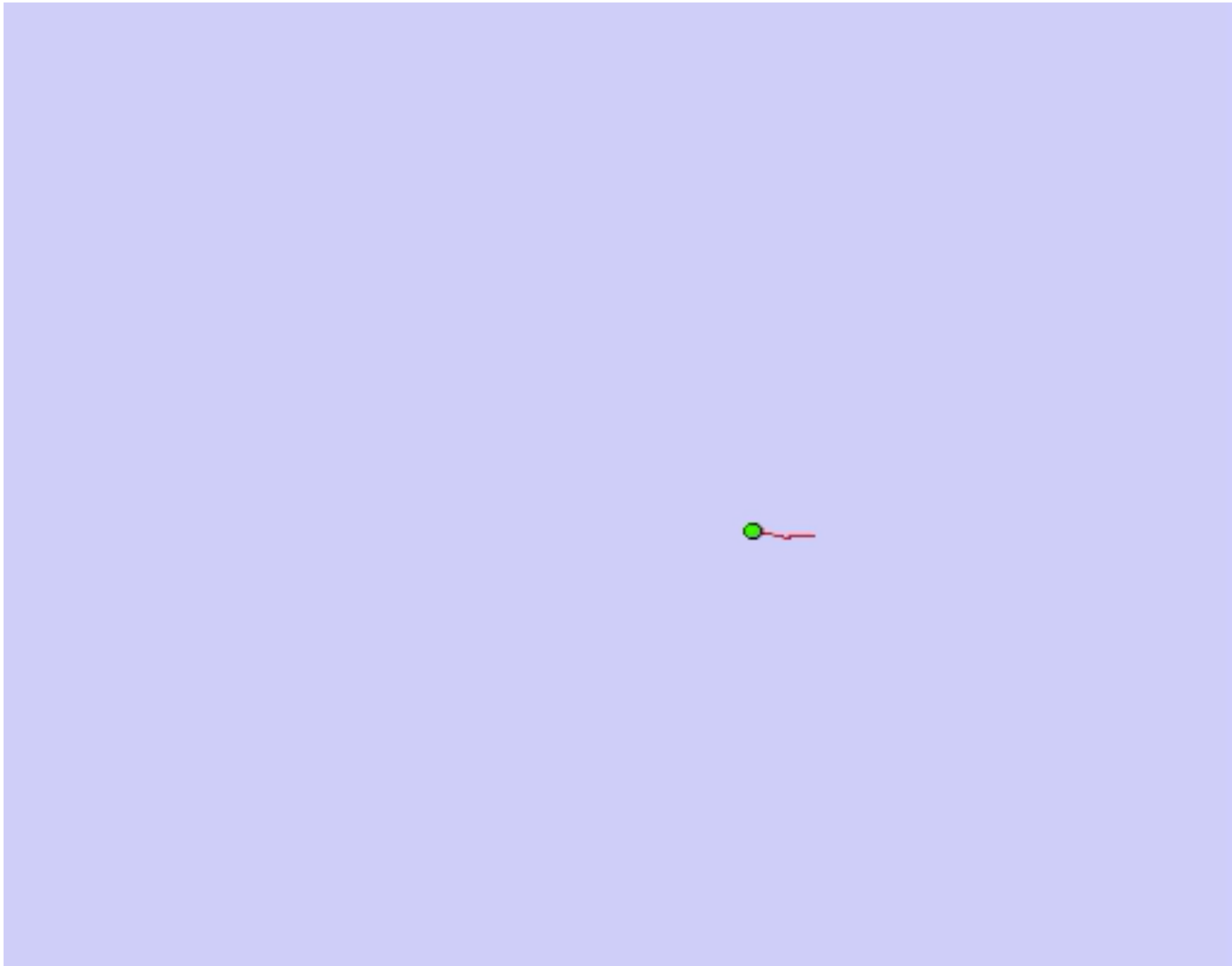


Raw Odometry

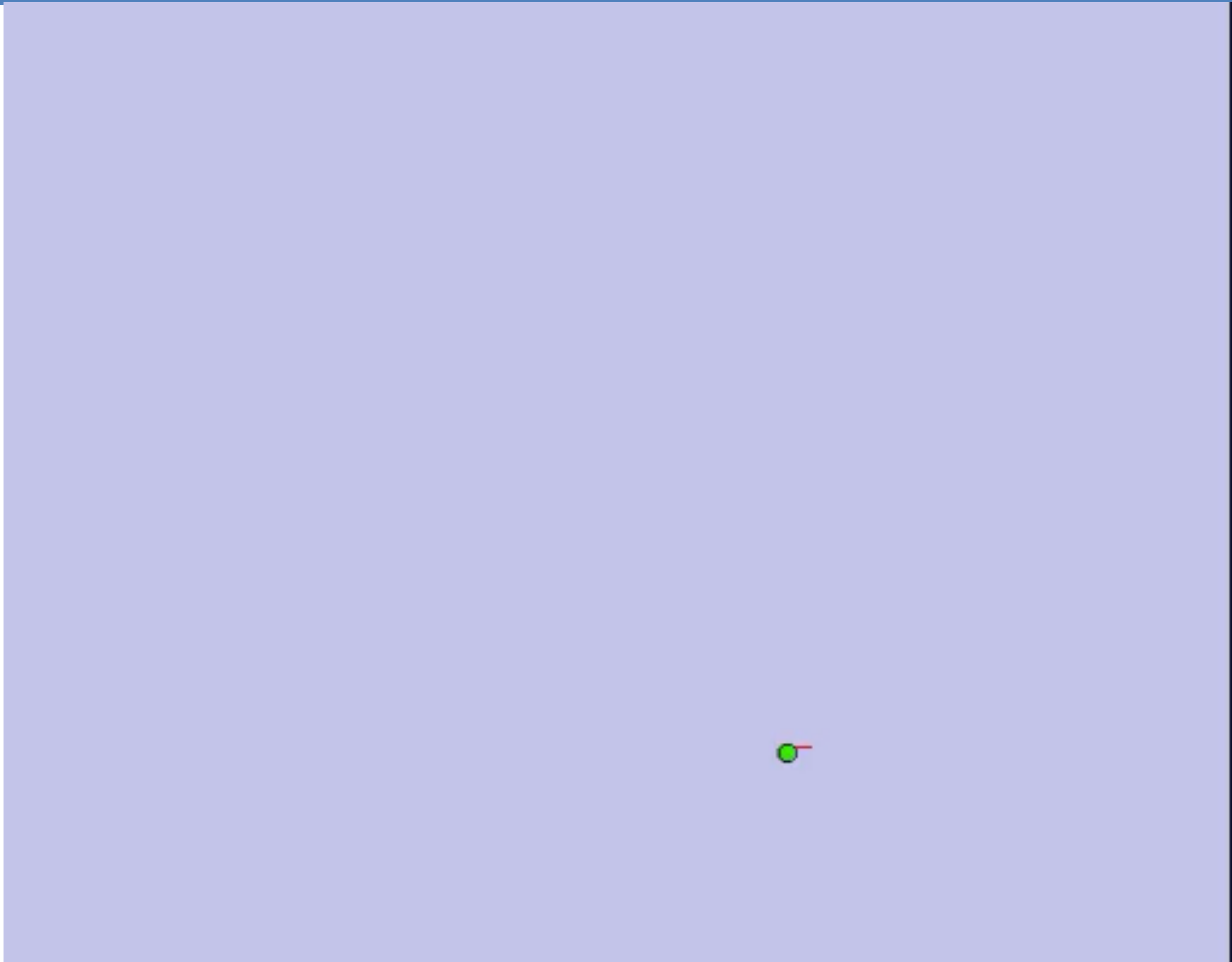
- Famous Intel Research Lab dataset (Seattle) by Dirk Hähnel

Courtesy of S. Thrun

http://robots.stanford.edu/videos.html

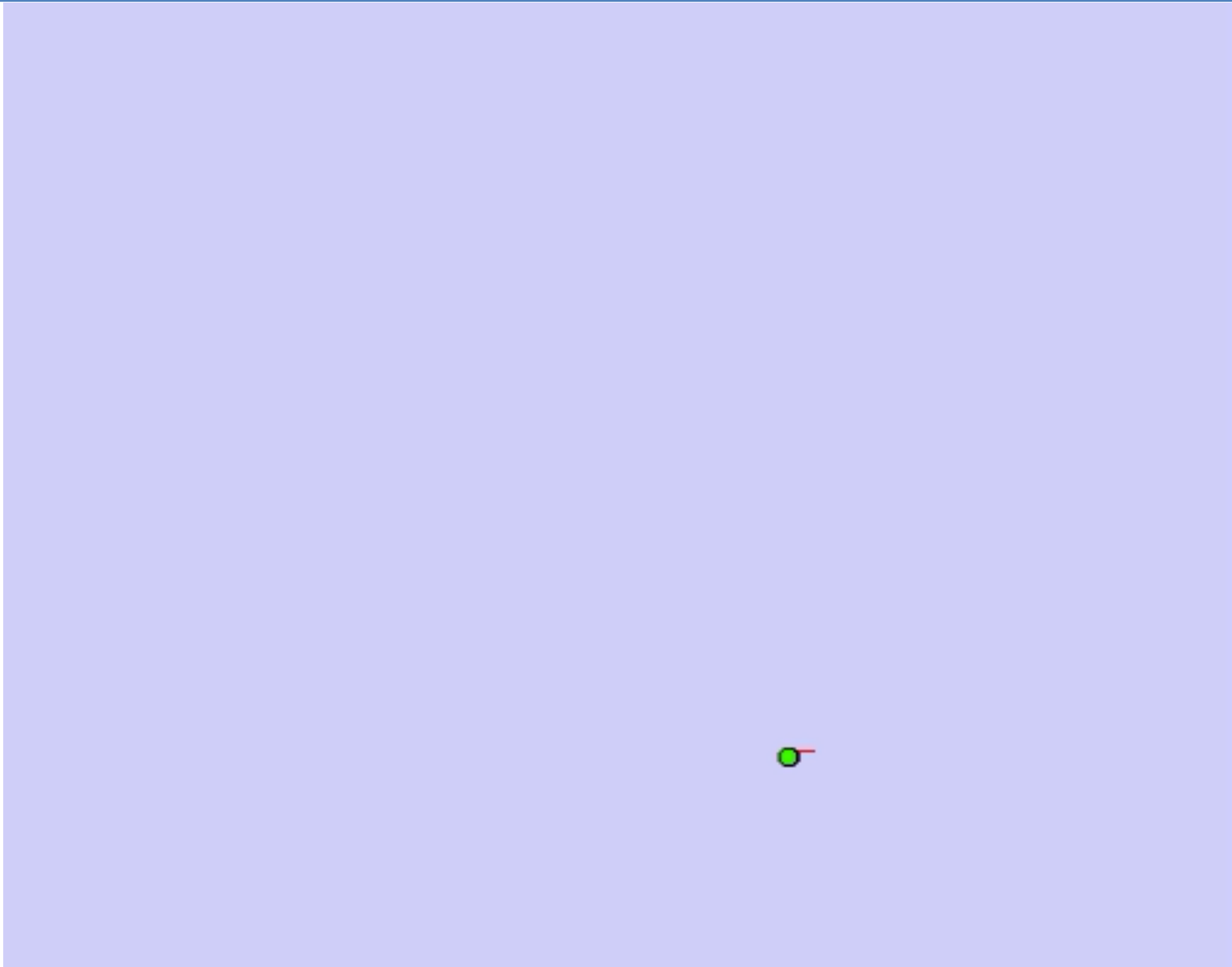


Scan Matching: compare to sensor data from previous scan



Courtesy of S. Thrun

FastSLAM: Particle-Filter SLAM



Courtesy of S. Thrun

Conclusion (thus far ...)

- The presented approach is a highly efficient algorithm for SLAM combining ideas of scan matching and FastSLAM
- Scan matching is used to transform sequences of laser measurements into odometry measurements
- This version of grid-based FastSLAM can handle larger environments than before in “real time”

What’s Next?

- Further reduce the number of particles
- Improved proposals will lead to more accurate maps
- Use the properties of our sensor when drawing the next generation of particles