# Principles of Robot Autonomy I

Extended Kalman Filter, EKF Mapping, Localization, and EKF SLAM,
Unscented Kalman Filter

Stanford University

MSL

IPRL

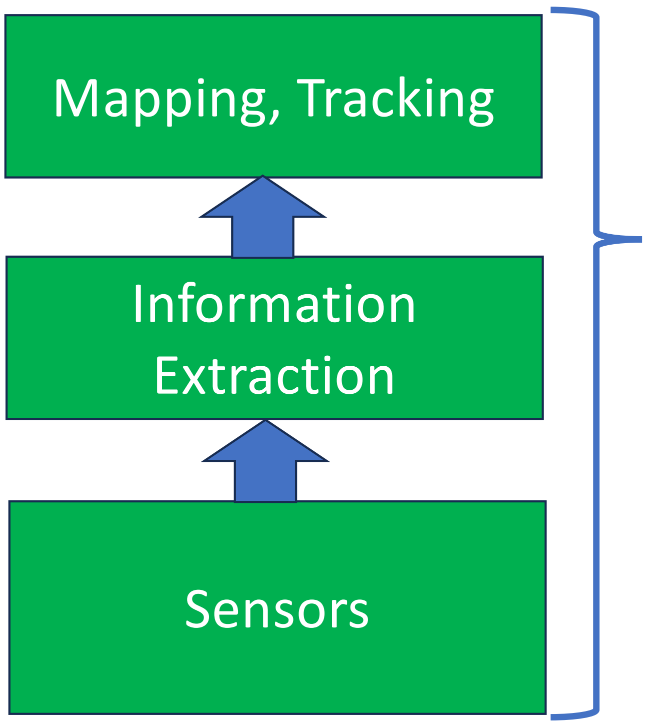
ASL
Autonomous Systems Lab

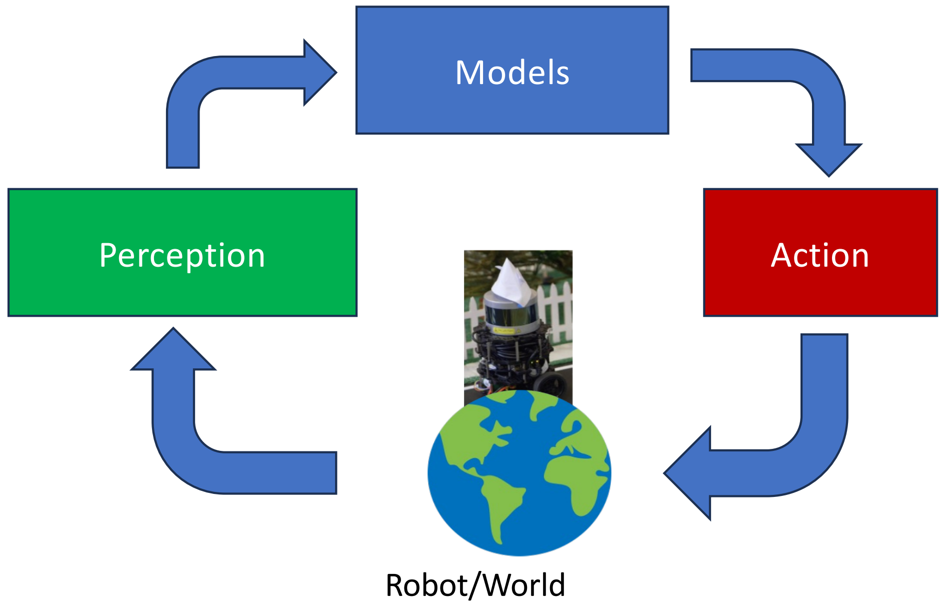
# Logistics

- Homework 5: Due 12/2 (last one!)
- Guest Lecture: Dr. Vincent Vanhoucke, Distinguished Engineer, Waymo, Thurs 11/20
- Final exam, window Sat 12/6, 6:30pm – Tues 12/9, 6:30pm:
    - Take home, 72 hour window
    - Check out exam on gradescope
    - You will have personal 5 hour time slot
    - Open notes, book, HW solutions
    - No internet, no GenAI, no working with others
- Lecture 15:
    - Extended Kalman Filter (EKF)
    - EKF localization, mapping, SLAM
    - Unscented Kalman Filter (UKF)

# Robot Perception

Perception Stack



# Perception Stack: Kalman Filtering

## Use sensor data to update the models

### Localization, Mapping, Tracking:

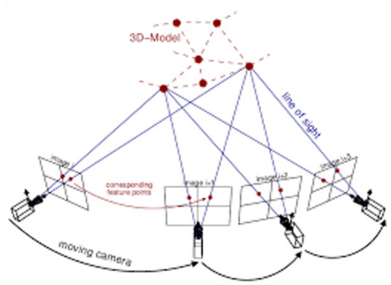
- EKF/Monte Carlo localization
- Occupancy grid mapping
- Factor graphs/SLAM
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**

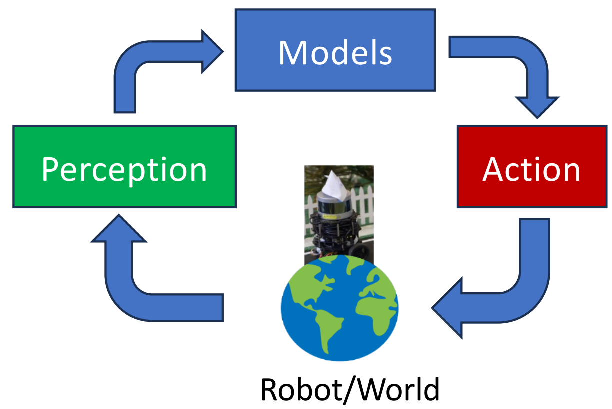
### Information extraction

- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

### Sensors:

- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders



# Recall: Kalman filter (KF)

- Assumption #1: linear dynamics

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{GWN}(0, R_t)$$

- Assumption #2: linear measurement model

$$z_t = C_t x_t + \delta_t \qquad \delta_t \sim \mathcal{GWN}(0, Q_t)$$

- Assumption #3: Gaussian Prior

$$bel(x_0) = \mathcal{N}(\mu_0, \Sigma_0)$$

# Kalman filter: algorithm

**Prediction**

Project state ahead

$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$$

Project covariance ahead

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

**Correction**

Compute Kalman gain

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$$

Update estimate with new measurement

$$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t)$$

Update covariance

$$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$$

**Data:** $\underbrace{(\mu_{t-1}, \Sigma_{t-1})}_{bel(x_{t-1})}, u_t, z_t$

**Result:** $(\mu_t, \Sigma_t)$

Prediction: $\overline{bel}(x_t)$

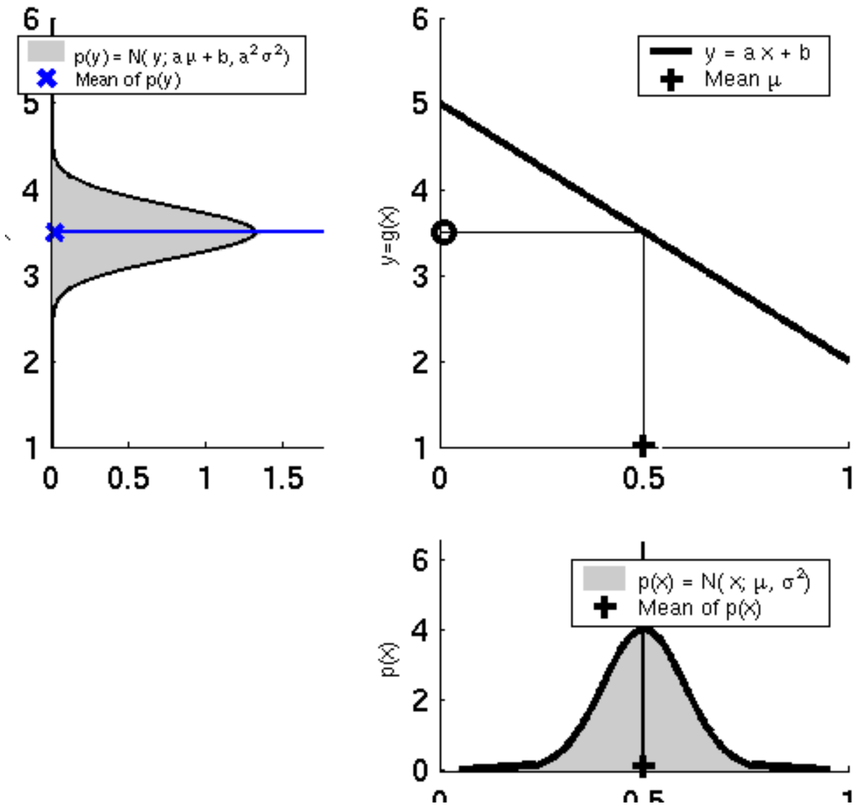
$$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \ ;$$
$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t;$$

Correction: $bel(x_t)$

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1};$$
$$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t);$$
$$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t;$$

Return $\underbrace{(\mu_t, \Sigma_t)}_{bel(x_t)}$

# Revisiting linearity assumption




- linear transformation of a Gaussian RV results in a Gaussian RV
- However, linearity assumptions are severely restrictive for robotics applications

# Extended Kalman filter (EKF)

- Goal: relax the linearity assumption
- The dynamics are now given by

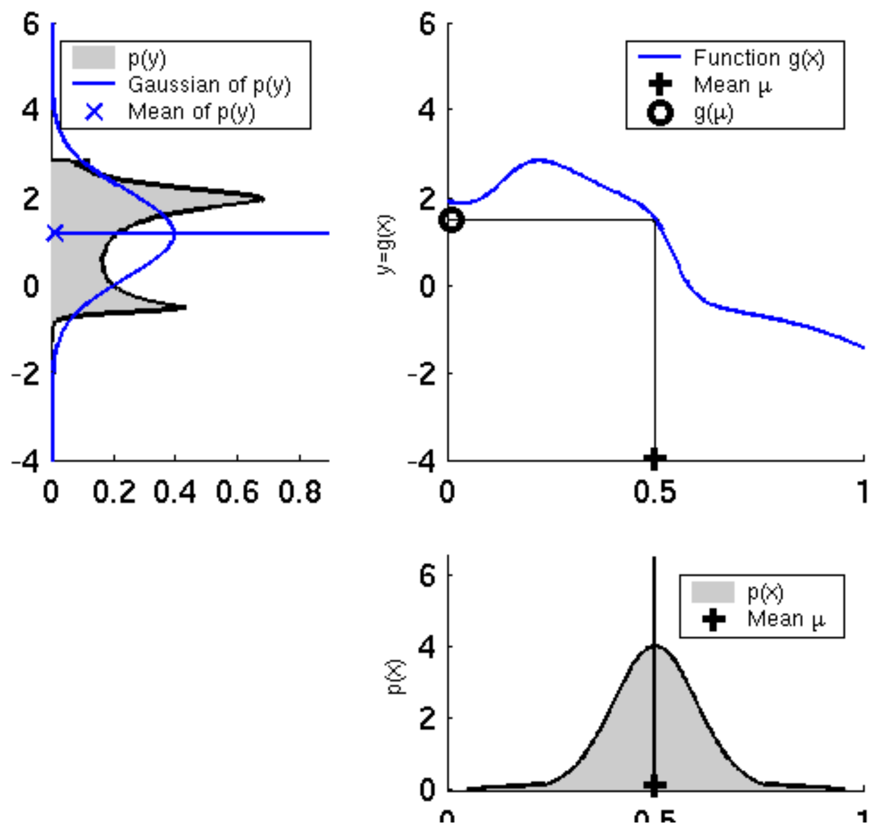
$$x_t = g(u_t, x_{t-1}) + \epsilon_t \qquad \epsilon_t \sim \mathcal{GWN}(0, R_t)$$

- And the measurement model is now given by

$$z_t = h(x_t) + \delta_t \qquad \delta_t \sim \mathcal{GWN}(0, Q_t)$$

- Key idea: shift focus from computing exact posterior to efficiently compute a Gaussian approximation

# Goal of EKF



p(y)
Gaussian of p(y)
Mean of p(y)
Function g(x)
Mean μ
g(μ)
y=g(x)
p(x)
Mean μ
p(x)

# EKF: key idea

- Key idea: linearize *g* and *h* around the most likely state and transform beliefs according to such linear approximations
- For the dynamics equation

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

Jacobian of *g*

- Accordingly

$$p(x_t \,|u_t,\, x_{t-1}) = \det(2\pi R_t)^{-1/2}$$
$$\exp\left(-\frac{1}{2}[x_t - g(u_t,\, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]^T R_t^{-1}[x_t - g(u_t,\, \mu_{t-1}) - G_t(x_{t-1} - \mu_{t-1})]\right)$$

# EKF: key idea

- For the measurement model

$$h(x_t) \approx h(\overline{\mu}_t) + \underbrace{J_h(\overline{\mu}_t)}_{:=H_t}(x_t - \overline{\mu}_t)$$

- Accordingly,

$$p(z_t \mid x_t) = \det(2\pi Q_t)^{-1/2} \exp\left(-\frac{1}{2}[z_t - h(\overline{\mu}_t) - H_t(x_t - \overline{\mu}_t)]Q_t^{-1}[z_t - h(\overline{\mu}_t) - H_t(x_t - \overline{\mu}_t)]\right)$$

# Jacobians

Given:

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$g(u_t, \mu_{t-1}) = \begin{bmatrix} g_1(u_t, \mu_{t-1}) \\ \vdots \\ g_n(u_t, \mu_{t-1}) \end{bmatrix}$$

$$h(\bar{\mu}_t) = \begin{bmatrix} h_1(\bar{\mu}_t) \\ \vdots \\ h_p(\bar{\mu}_t) \end{bmatrix}$$

We have:

$$G_t = J_g(u_t, \mu_{t-1}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_j} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial g_i}{\partial x_1} & \cdots & \frac{\partial g_i}{\partial x_j} & \cdots & \frac{\partial g_i}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial x_1} & \cdots & \frac{\partial g_n}{\partial x_j} & \cdots & \frac{\partial g_n}{\partial x_n} \end{bmatrix} \Big|_{x=\mu_{t-1}}$$

$$H_t = J_h(\bar{\mu}_t) = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_j} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial h_i}{\partial x_1} & \cdots & \frac{\partial h_i}{\partial x_j} & \cdots & \frac{\partial h_i}{\partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial h_p}{\partial x_1} & \cdots & \frac{\partial h_p}{\partial x_j} & \cdots & \frac{\partial h_p}{\partial x_n} \end{bmatrix} \Big|_{x=\bar{\mu}_t}$$

# EKF: algorithm

- Main differences:
    1. Linear predictions are replaced by their nonlinear generalizations
    2. EKF uses Jacobians instead of linear system matrices
    3. Mathematical derivation of EKF parallels that of KF

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

$$J_g(u_t, \mu_{t-1}) := G_t$$

$$z_t = h(x_t) + \delta_t$$

$$J_h(\bar{\mu}_{t-1}) := H_t$$

**Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$

**Result:** $(\mu_t, \Sigma_t)$

$\overline{\mu}_t = g(u_t, \mu_{t-1})$ ;

$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t{}^T + R_t$;

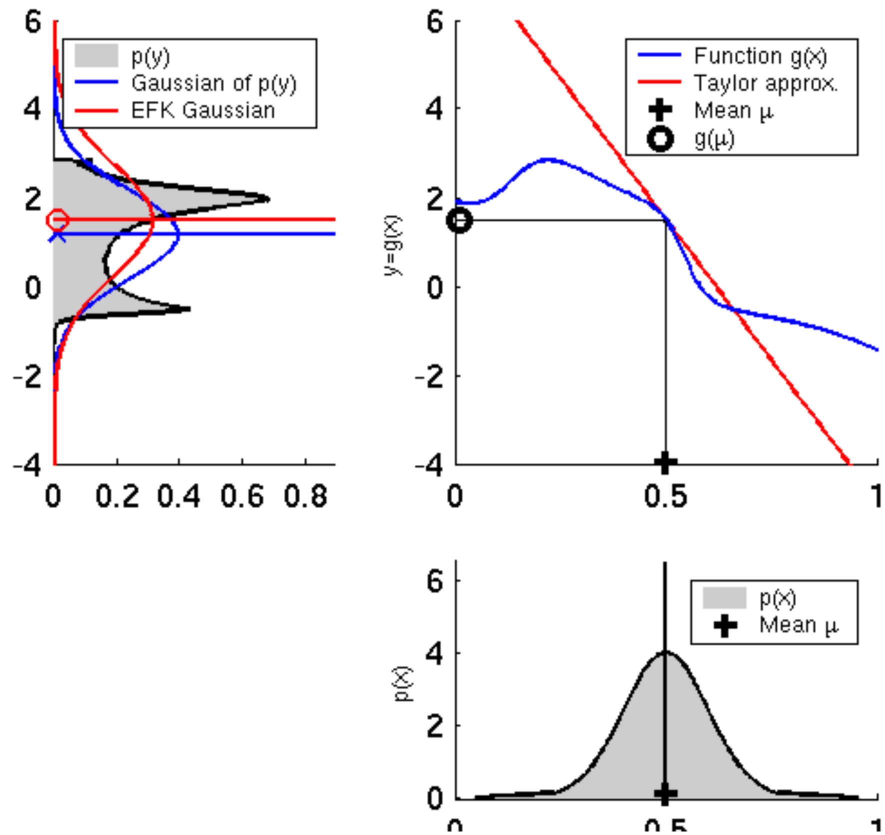
$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$;

$\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$;

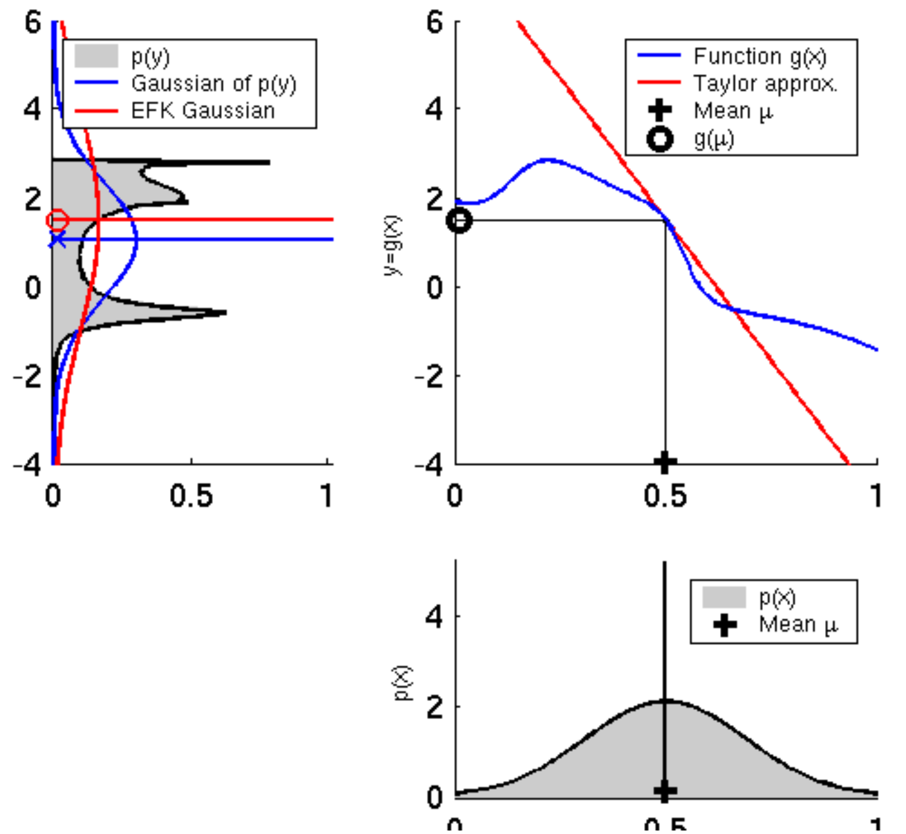
$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$;

Return $(\mu_t, \Sigma_t)$

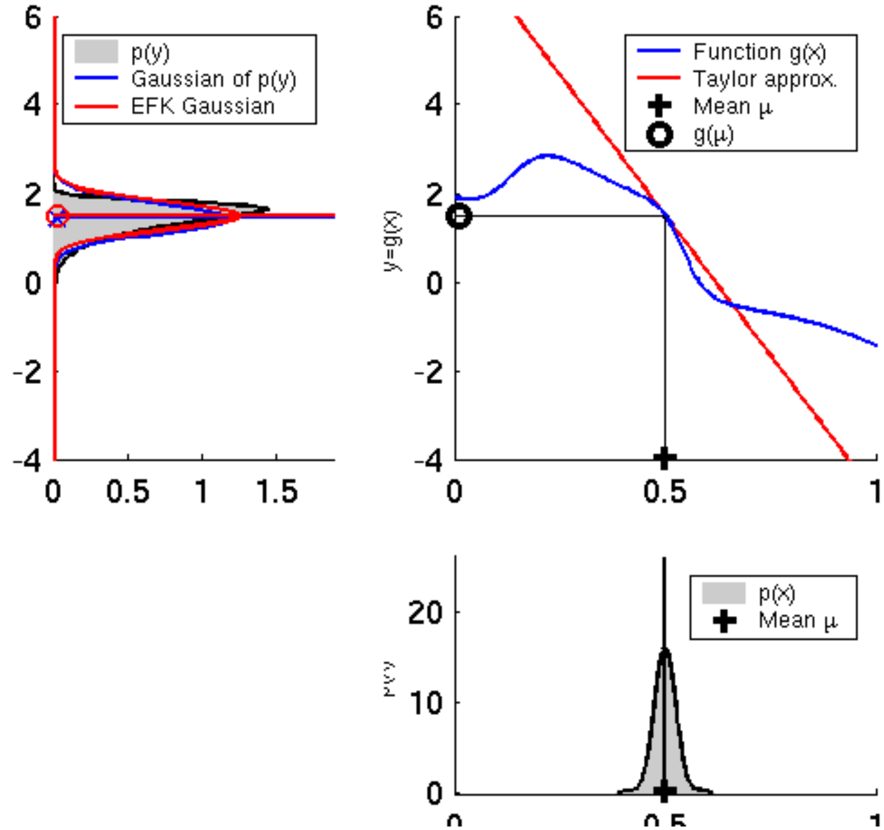
# EKF: intuition



# EKF: intuition



# EKF: intuition

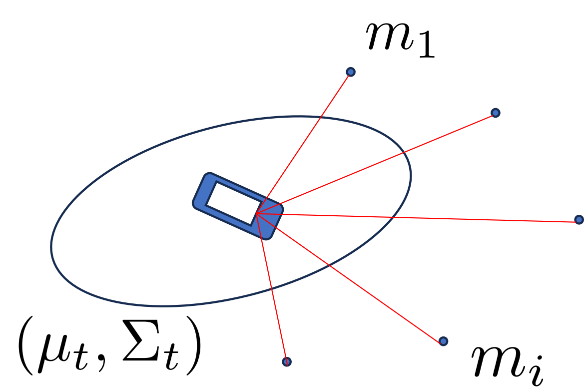


# EKF Example: Robot localization in a feature map

- Map represented as a set of known feature locations (a point cloud)

- We want to estimate robot's pose

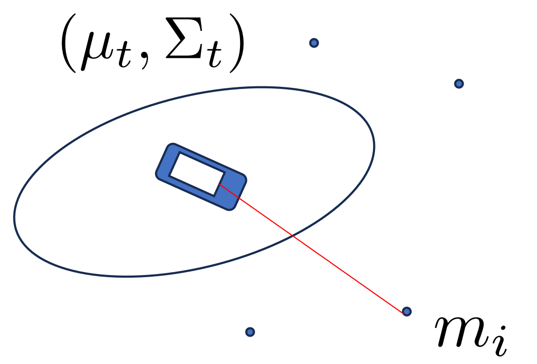
$$[x_t \;\; y_t \;\; \theta_t]^T = q_t \sim \mathcal{N}(\mu_t, \Sigma_t)$$



# EKF Example: Robot localization in a feature map

Known map feature locations: $m_i, \quad i = 1, \ldots, N$

Robot pose prior: $q_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$




Possible sensor models (same as before!):

Find Jacobian with respect to $q_t$

Linear: $z_t^i = R(\theta_t)(m_t^i - p_t) + \delta_t^i, \quad \delta_t^i \sim \mathcal{N}(0, Q_t^i)$

Range: $z_t^i = \|m_t^i - p_t^i\| + \delta_t^i$

Bearing: $z_t^i = \mathrm{atan}\left(\frac{m_t^i(2) - y_t}{m_t^i(1) - x_y}\right) - \theta_t + \delta_t^i$

$$z_t = [z_t^1 \quad \ldots \quad z_t^N]^T$$

Robot dynamics model:

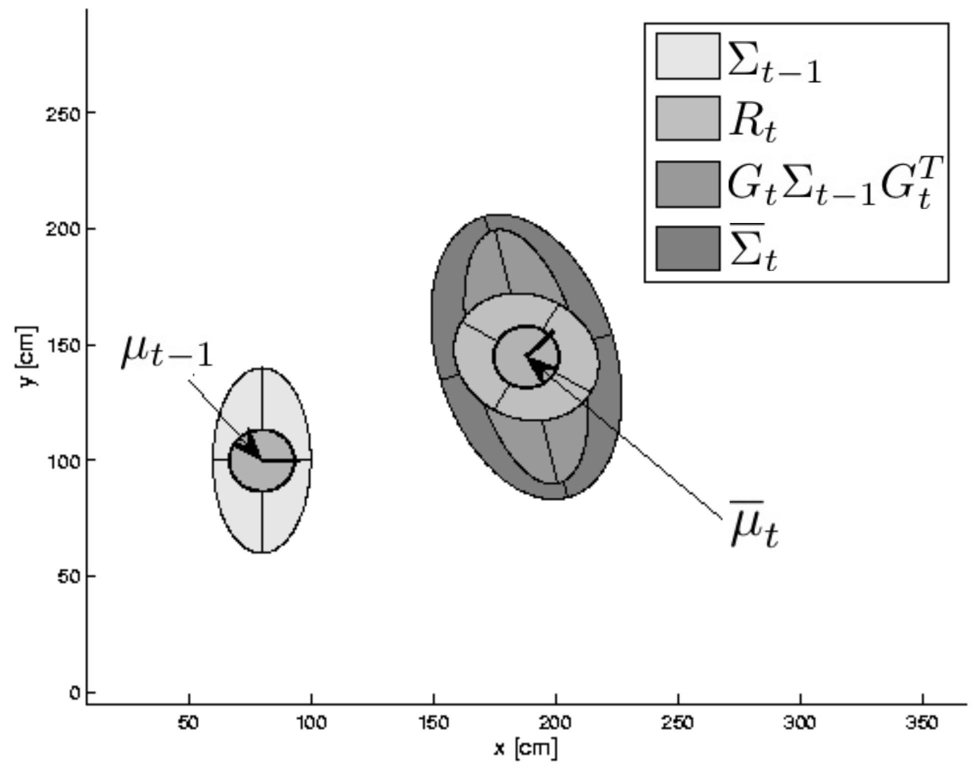
Find Jacobian with respect to $q_t$

$$\begin{aligned} x_{t+1} &= x_t + v_t \delta_t \cos(\theta_t) + \epsilon_t^x \\ y_{t+1} &= y_t + v_t \delta_t \sin(\theta_t) + \epsilon_t^y \\ \theta_{t+1} &= \theta_t + \omega_t \delta_t + \epsilon_t^\theta \end{aligned}$$

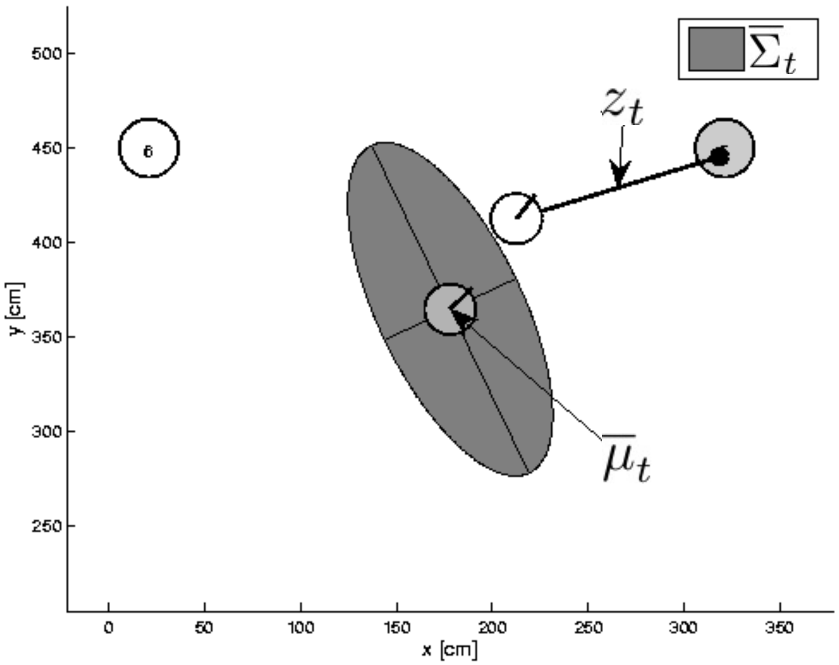
$$[\epsilon_t^x \;\; \epsilon_t^y \;\; \epsilon_t^\theta]^T = \epsilon_t \sim \mathcal{N}(0, R_t)$$

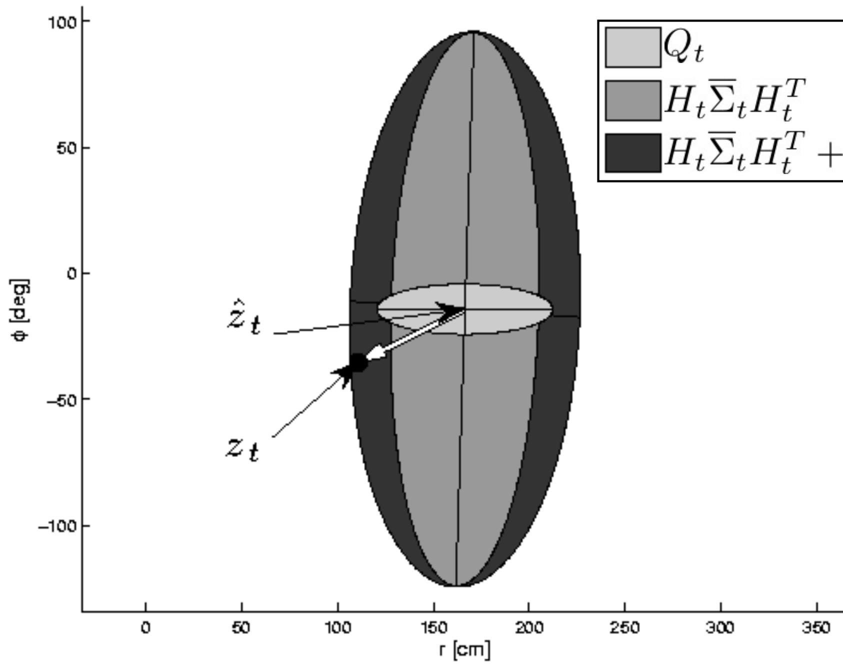
# Example of EKF-localization: prediction step

- Observations measure relative distance and bearing to a marker
- For simplicity, we assume that the robot detects only one marker at a time

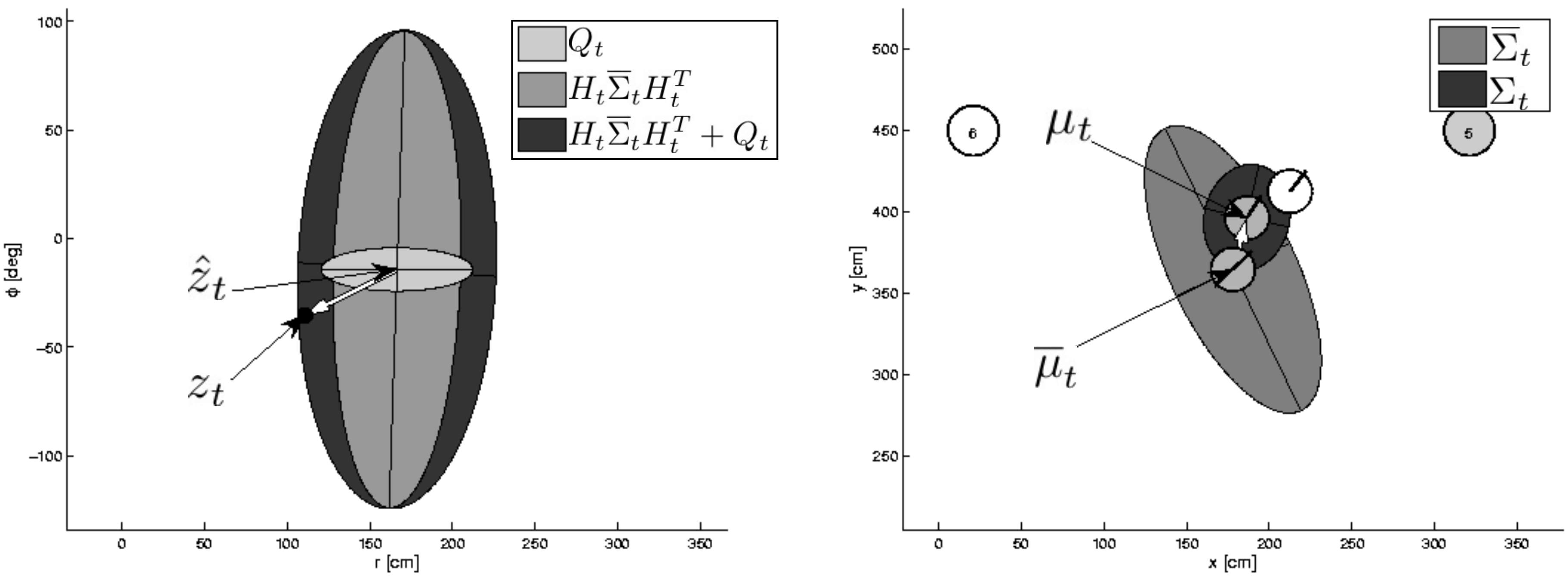


# Example of EKF-localization: measurement prediction step



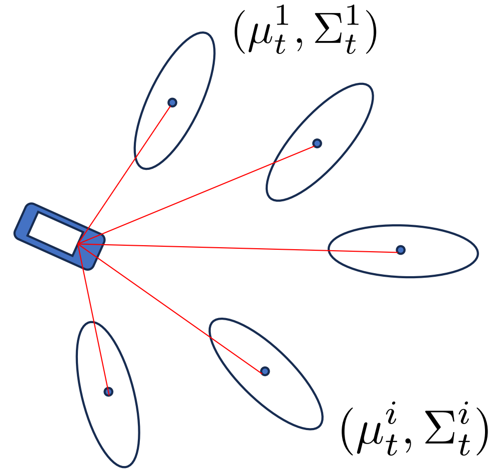
# Example of EKF-localization: correction step



# EKF Example: Feature-Based Mapping

- Map represented as a set of feature locations (a point cloud)

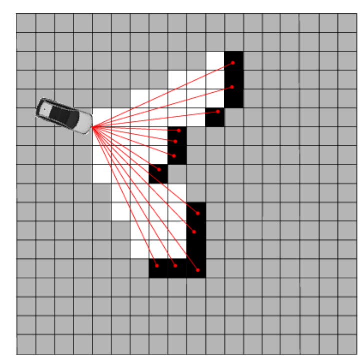
$$p(m_i \mid z^i_{1:t}) \sim \mathcal{N}(\mu^i_t, \Sigma^i_t)$$




- In contrast to occupancy mapping

(e.g. with histogram filter)

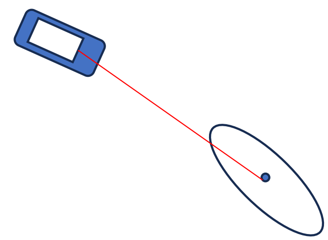
$$p(m(i,j) \mid z_{1:t}(i,j)) \in [0,1]$$



# EKF Example: Feature-Based Mapping

Known robot pose: $q_t = [x_t \;\; y_t \;\; \theta_t]^T \quad p_t = [x_t \;\; y_t]^T$

Feature location prior: $m_0^i \sim \mathcal{N}(\mu_0^i, \Sigma_0^i)$



$(\mu_t^i, \Sigma_t^i)$

Possible sensor models ( $z_t = h(x_t, u_t) + \delta_t$ ):

Linear (use KF): $z_t^i = R(\theta_t)(m_t^i - p_t) + \delta_t^i, \quad \delta_t^i \sim \mathcal{N}(0, Q_t^i)$

Range (EKF): $z_t^i = \|m_t^i - p_t^i\| + \delta_t^i$

Bearing (EKF): $z_t^i = \operatorname{atan}\left(\frac{m_t^i(2) - y_t}{m_t^i(1) - x_y}\right) - \theta_t + \delta_t^i$

Find Jacobian with respect to $m_t^i$

"Static" Dynamics model ( $x_{t+1} = g(x_t, u_t) + \epsilon_t$ ):

$$m_{t+1}^i = m_t^i + \epsilon_t^i, \quad \epsilon_t^i \sim \mathcal{N}(0, R_t^i) \qquad R_i \text{ very small}$$

# Plug into Kalman filter/EKF

**Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
**Result:** $(\mu_t, \Sigma_t)$
$\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$ ;
$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$;

$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$;
$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t)$;
$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$;

Return $(\mu_t, \Sigma_t)$

**Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
**Result:** $(\mu_t, \Sigma_t)$
$\overline{\mu}_t = g(u_t, \mu_{t-1})$ ;
$\overline{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + R_t$;

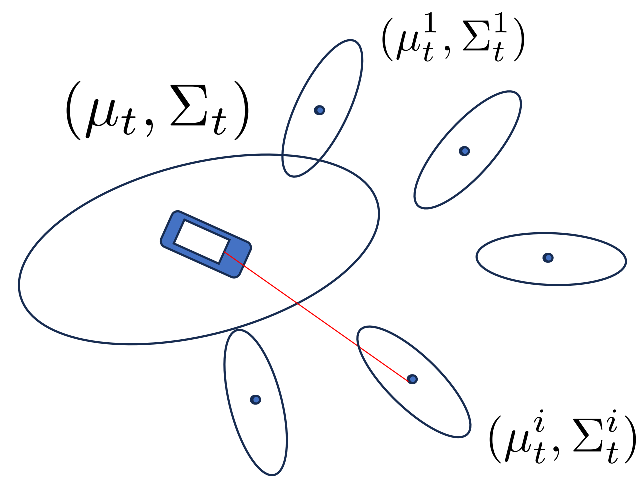
$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$;
$\mu_t = \overline{\mu}_t + K_t (z_t - h(\overline{\mu}_t))$;
$\Sigma_t = (I - K_t H_t) \overline{\Sigma}_t$;

Return $(\mu_t, \Sigma_t)$

# EKF Example: EKF SLAM

Known map feature locations: $m_i, \quad i = 1, \ldots, N$

SLAM prior: $q_0 \sim \mathcal{N}(\mu_0, \Sigma_0) \;\; m_0^i \sim \mathcal{N}(\mu_0^i, \Sigma_0^i)$




Possible sensor models (same as before!):

Find Jacobian with respect to $q_t, m_t^i$

Linear: $z_t^i = R(\theta_t)(m_t^i - p_t) + \delta_t^i, \quad \delta_t^i \sim \mathcal{N}(0, Q_t^i)$

Range: $z_t^i = \|m_t^i - p_t^i\| + \delta_t^i$

Bearing: $z_t^i = \operatorname{atan}\left(\dfrac{m_t^i(2) - y_t}{m_t^i(1) - x_y}\right) - \theta_t + \delta_t^i$

$$z_t = [z_t^1 \quad \ldots \quad z_t^N]^T$$

Robot dynamics model:

$$\begin{aligned} x_{t+1} &= x_t + v_t \delta_t \cos(\theta_t) + \epsilon_t^x \\ y_{t+1} &= y_t + v_t \delta_t \sin(\theta_t) + \epsilon_t^y \\ \theta_{t+1} &= \theta_t + \omega_t \delta_t + \epsilon_t^\theta \end{aligned}$$

$$[\epsilon_t^x \;\; \epsilon_t^y \;\; \epsilon_t^\theta]^T = \epsilon_t \sim \mathcal{N}(0, R_t)$$

# Plug into EKF

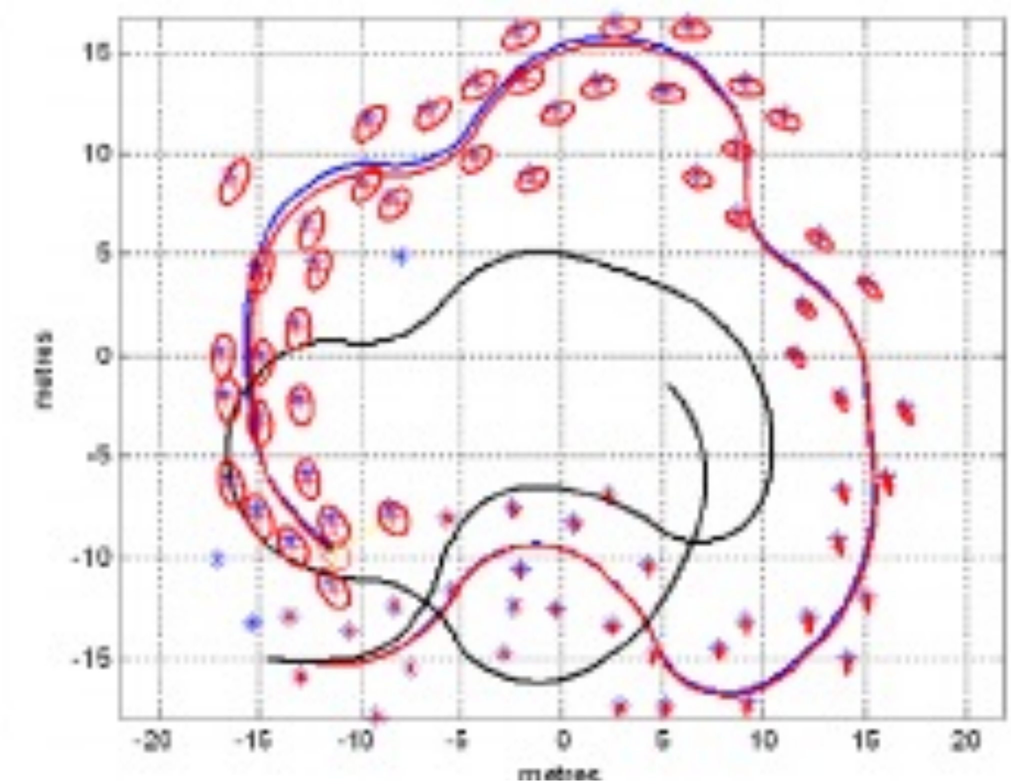
**Data:** $(\mu_{t-1}, \Sigma_{t-1}), u_t, z_t$
**Result:** $(\mu_t, \Sigma_t)$

$$\overline{\mu}_t = g(u_t, \mu_{t-1}) \ ;$$
$$\overline{\Sigma}_t = G_t \Sigma_{t-1} {G_t}^T + R_t;$$

$$K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1};$$
$$\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t));$$
$$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t;$$

Return $(\mu_t, \Sigma_t)$

# EKF Example: EKF SLAM



# The issue of data association

- Data association problem: uncertainty may exist regarding the identity of a landmark
- Formally, we define a *correspondence variable* between measurement $z_t^i$ and landmark $m_j$ in the map as (assume $N$ landmarks)

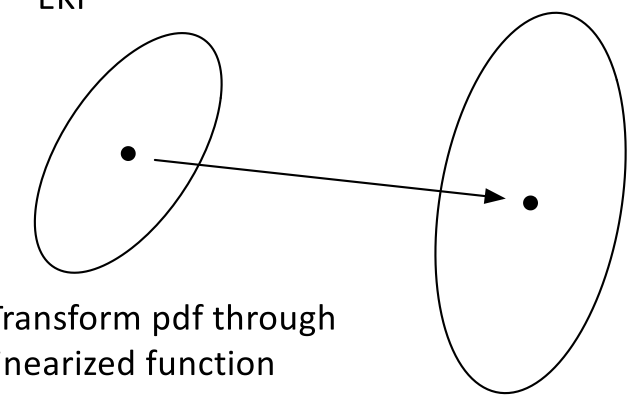
$$c_t^i \in \{1, \ldots, N+1\}$$

  - $c_t^i = j \leq N$ if *i*-th measurement at time *t* corresponds to *j*-th landmark
  - $c_t^i = N+1$ if a measurement does not correspond to any landmark
- Two versions of the localization problem
  1. Correspondence variables are known
  2. Correspondence variables are not known (usual case)

# Unscented Kalman filter (UKF) – basic idea

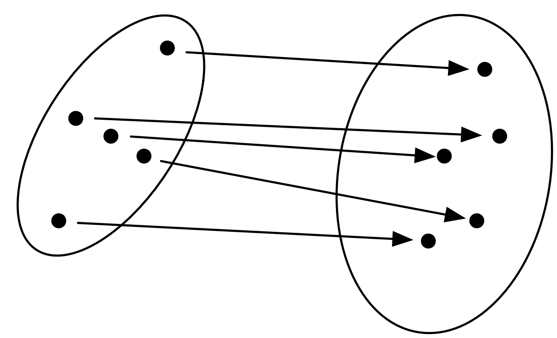
- Taylor series expansion applied by EKF is not the only way to approximate the transformation of a Gaussian;
- Unscented transform/sigma-point transform

EKF



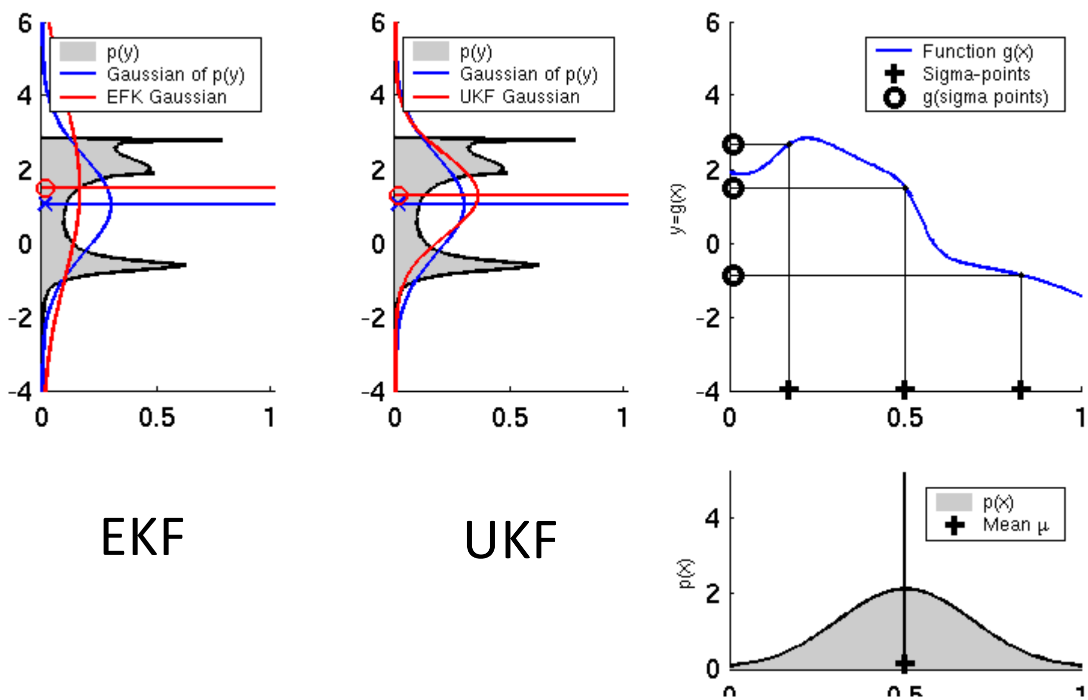
Transform pdf through linearized function

UKF

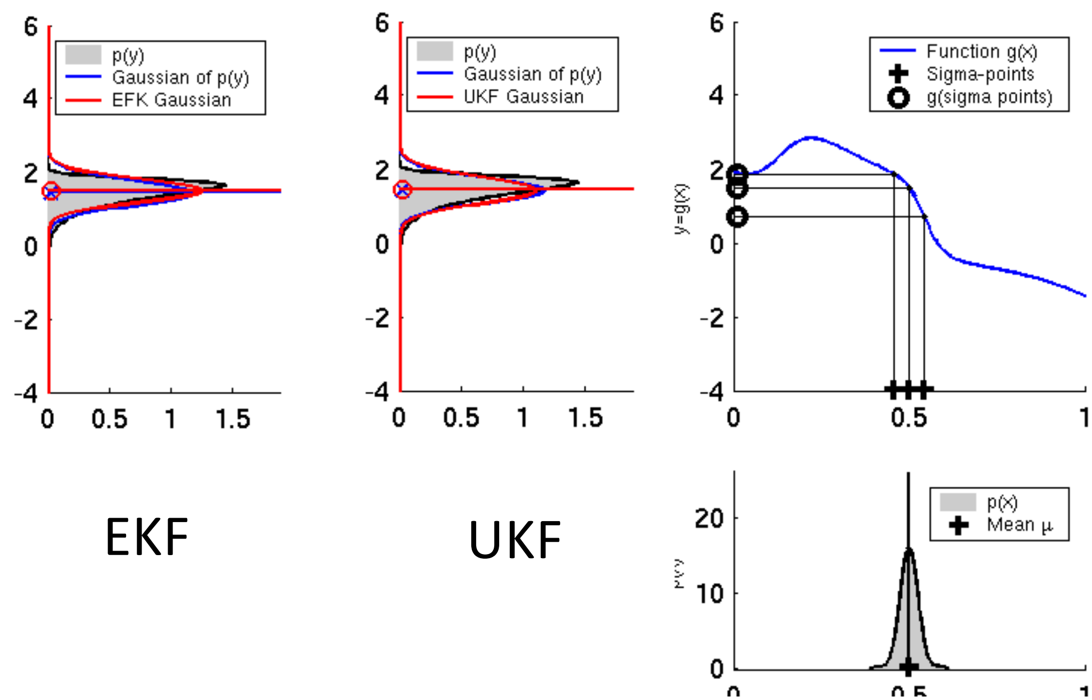


1. Compute sigma-points
2. Transform each sigma point through nonlinear function
3. Compute Gaussian from the transformed and weighted sigma-points

# UKF: example


p(y)
Gaussian of p(y)
EFK Gaussian
p(y)
Gaussian of p(y)
UKF Gaussian
Function g(x)
Sigma-points
g(sigma points)
y=g(x)
p(x)
Mean μ
p(x)
EKF
UKF

# UKF: example




EKF

UKF

# Next time