Robot Localization

Javier González Jiménez

Reference Books:

- Probabilistic Robotics. S. Thrun, W. Burgard, D. Fox. MIT Press. 2001
- Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods. Juan-Antonio Fernández-Madrigal and José Luis Blanco Claraco. IGI-Global. 2013.

Content

- Least Squares Positioning
- Map registration
  - Landmark points (known correspondence)
  - Scan points: ICP (unknown correspondences)
- Filtering
  - Kalman Filter
  - Extended Kalman Filter
  - Particle Filter

Robot Localization

"Using sensory information to locate the robot in its environment is the most fundamental problem to providing a mobile robot with autonomous capabilities." [Cox '91]

Given

Map of the environment

Sequence of sensor measurements

Wanted

Robot pose

Types of localization problems

- ***Position tracking***: the robot knows approximately where it is
- ***Global localization***: The robot has no clue where it is
- ***Kidnapped robot problem***: It thinks where it is but is wrong

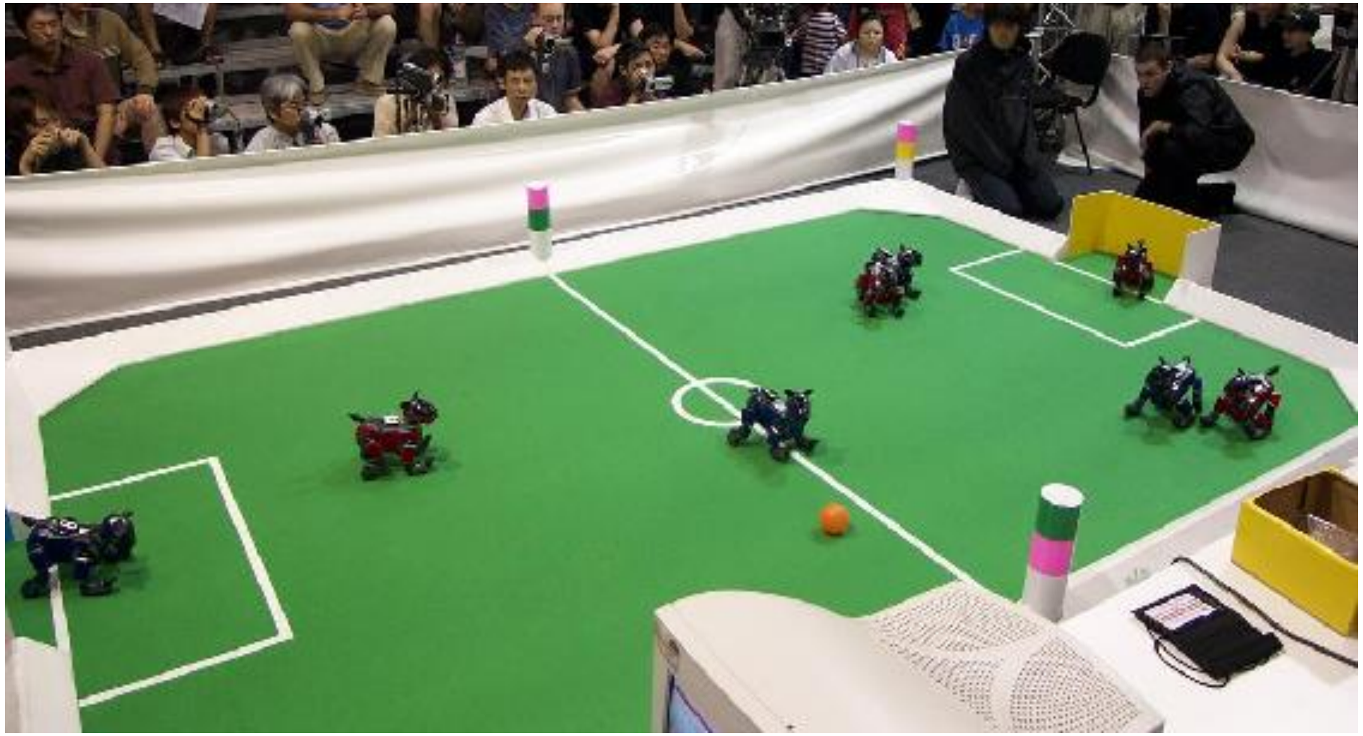
Least Squares Positioning

Problem: Given a vector of measurements z that is related to the unknown pose x by the linear observation model (o function)

$$z = Hx$$

we want to find the "best" pose $\hat{x}$

Example:



Least Squares Positioning: Given $z = Hx$, find the (best) pose $\hat{x}$

$$z_{\mathrm{mx1}} = H_{\mathrm{mxn}} x_{\mathrm{nx1}}$$

n: pose, typically 3 o 6 unknowns
m: #observation constraints (equations)

H
m=3 
n=3

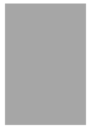
if m=n: unknowns (n) equals (independent) equations (m). For example, having 3 independent observations to compute the pose

- H (3x3) is invertible, direct solution for x: $x = H^{-1}z$

H
m 
n=3

if m<n: more unknowns (n) than equations (m)

- The pose is Non-observable: infinitely many solutions for x ($H^T H$ not invertible)

H
m 
n=3

if m>n: less unknowns (n) than equations (m)

- No exact solution since measurements z are affected by error: $z = Hx + e$
- Find x that is closest to the ideal → minimum square error

$$\hat{x} = \arg\min_x \overset{\text{Cost function } \|e\|^2}{\boxed{e^T e}} = \arg\min_x \left[(z - Hx)^T (z - Hx)\right] = \arg\min_x \overset{\text{Cost function}}{\boxed{\|z - Hx\|^2}}$$

Solution: $\hat{x} = (H^T H)^{-1} H^T z$

Weighted Least Squares

- Measurements are not equally reliable: different "importance" of the noise error for each measurement $e_i = z_i - [Hx]_i$ Element i of the vector Hx

- This is modelled by a diagonal covariance matrix

  (no correlation between the error measurements, since they are assumed to be independent one to another)

$$Q = \begin{bmatrix} \sigma_{z_1}^2 & 0 & \cdots \\ 0 & \sigma_{z_2}^2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \qquad Q^{-1} = \begin{bmatrix} \frac{1}{\sigma_{z_1}^2} & 0 & \cdots \\ 0 & \frac{1}{\sigma_{z_1}^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Each element e_i of the error vector e is inversely weighted by the uncertainty in measurement z, i.e.$1/\sigma_{z_i}^2$

Cost function

$$\hat{x} = \arg\min_x \boxed{e^T Q^{-1} e} = \arg\min_x \left[(z - Hx)^T Q^{-1} (z - Hx)\right] = \arg\min_x \sum_{i=1}^{m} \frac{e_i^2}{\sigma_{z_i}^2}$$

Solution:

[Closed form]

$$\hat{x} = (H^T Q^{-1} H)^{-1} H^T Q^{-1} z$$

Best estimation

$$\Sigma_{\hat{x}} = (H^T Q^{-1} H)^{-1}$$

Uncertainty of the estimation

LS equivalent to ML Estimation if error is Gaussian

Maximum Likelihood (ML) Estimation [*from lecture 2*]:

Find x that maximizes the likelihood function $\mathcal{L}(x) \triangleq p(z|x)$

$$x_{ML} = \arg\max_{x} p(z|x)$$

If the error in the measurements is Gaussian, i.e. $e \sim N(0, Q)$

$$z = Hx + e \Rightarrow z \sim N(Hx, Q)$$

$$p(z|x) \;\; = K \cdot \exp\{-\frac{1}{2}[z - Hx]^T Q^{-1} [z - Hx]\}$$

$$\hat{x}_{ML} = \arg\max_{x} p(z|x) \;\; = \arg\max_{x} \{-[z - Hx)]^T Q^{-1} [z - Hx]\}$$

$$= \arg\min_{x} [(z - Hx)^T Q^{-1} (z - Hx)] = \arg\min_{x} e^T Q^{-1} e = \hat{x}_{LS}$$

$$\hat{x}_{ML} = \hat{x}_{LS}$$

The LS and ML solutions are the same

Non-linear Least Squares

Non-linear observation (sensor) model: $z = h(x)$ (instead of $z = Hx$)

$$\hat{x} = \arg\min_{x} \|z - h(x)\|^2$$

These is a vector of m numbers (observations), not RVs!

No closed-form solution exists, but iterative:

Jacobian of h evaluated at x_0

Taylor expansion: $h(x) = h(x_0 + \delta) \cong h(x_0) + J_{h_0}\delta$

Error vector e_0

$$\|z - h(x)\|^2 \cong \left\|z - h(x_0) - J_{h_0}\delta\right\|^2 = \left\|\boxed{(z - h(x_0))} - J_{h_0}\delta\right\|^2 = \left\|e_0 - J_{h_0}\delta\right\|^2$$

Equivalent optimization problem

$$\delta = \arg\min_{\delta} \|e_0 + J_e\delta\|^2 \Longrightarrow \delta = -(J_e^T J_e)^{-1} J_e^T e_0$$

3x1 3xn 3xm mx1

δ that makes this squared norm minimum

$$J_{h_0} = \left.\frac{dh}{dx}\right|_{x_0} = \begin{bmatrix} \frac{dh_1}{dx_1} & \cdots & \frac{dh_1}{dx_n} \\ \vdots & & \vdots \\ \frac{dh_m}{dx_1} & \cdots & \frac{dh_m}{dx_n} \end{bmatrix}_{x_0} = -J_e$$

mxn

Jacobian of the error vector $e = z - h(x)$ is the minus JACOBIAN of the sensor model $h(x)$

Weighted Non-linear Least Squares

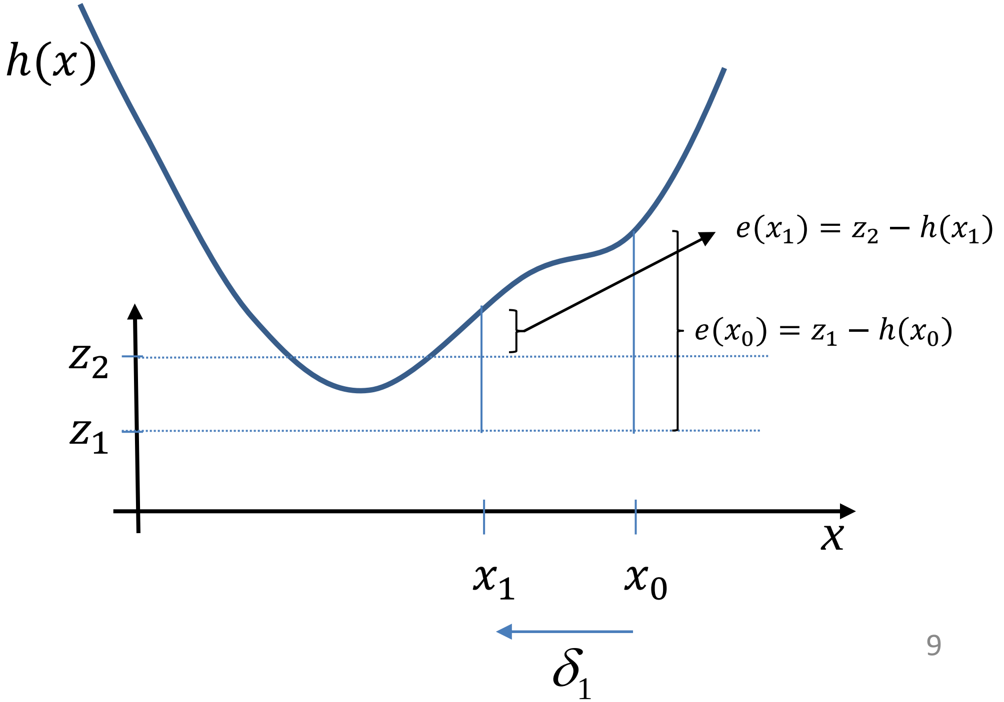
Hessian and gradient of the cost function $f = \frac{1}{2} e(x)^T Q^{-1} e(x)$

$$\delta = -\underbrace{(J_e^T Q^{-1} J_e)}_{H_f}{}^{-1} \underbrace{J_e^T Q^{-1} e}_{\nabla_f} = H_f^{-1} \nabla f$$

$$\begin{cases} J_e^T Q^{-1} J_e = H_f & \text{Hessian matrix of } f \text{ (Fisher Information matrix)} \\ J_e^T Q^{-1} e = \nabla f & \text{Gradient vector of } f \end{cases}$$

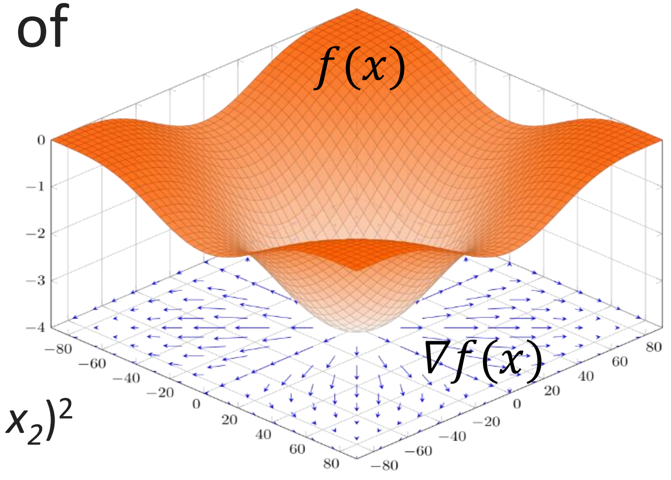
Algorithm (known as Gauss-Newton):

1. Begin with an initial guess $\hat{x}$
2. Evaluate
   $$\delta = -\left(J_e^T Q^{-1} J_e\right)^{-1} J_e^T Q^{-1}\left[z - h(\hat{x})\right]$$
3. Set $\hat{x} = \hat{x} - \delta$
4. If $\delta > tolerance$ goto 1
   else stop



A note on Jacobian and Gradient

- The **gradient** is the vector formed by the partial derivatives of a scalar (or real-value) function ($f: \mathbb{R}^n \rightarrow \boxed{\mathbb{R}}$)



GRADIENT of f: $\nabla f(x) = \left[\frac{\partial f}{\partial x_1} \quad \cdots \quad \frac{\partial f}{\partial x_n}\right] \in \mathbb{R}^n$

Example (Wikipedia): Gradient of the function $f(x_1,x_2) = -(\cos^2 x_1 + \cos^2 x_2)^2$ depicted as a projected vector field on the bottom plane

- The **Jacobian** matrix is
  - the matrix formed by the partial derivatives of a vector function ($\boldsymbol{f}: \mathbb{R}^n \rightarrow \boxed{\mathbb{R}^m}$)
  - a generalization of the **gradient** to vector functions.

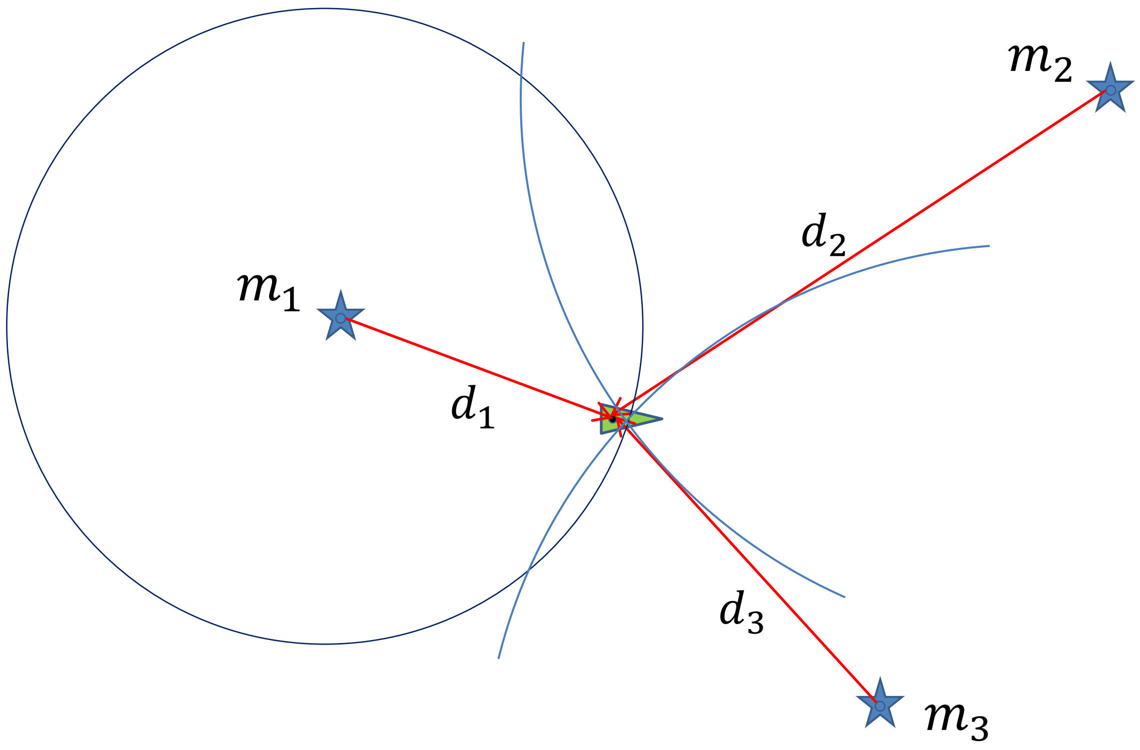
$\nabla f_1(x)$: each row is a gradient

JACOBIAN of $\boldsymbol{f}$: $$J = \left[\frac{\partial \boldsymbol{f}}{\partial x_1} \quad \cdots \quad \frac{\partial \boldsymbol{f}}{\partial x_n}\right] = \begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_n} \\ \vdots & & \vdots \\ \frac{df_m}{dx_1} & \cdots & \frac{df_m}{dx_n} \end{bmatrix}_{x_0} \qquad J = \left[\frac{\partial f_i}{\partial x_j}\right]$$

Least Squares Positioning

Example: Multi-lateration(also called *Range-only positioning*) in 2D

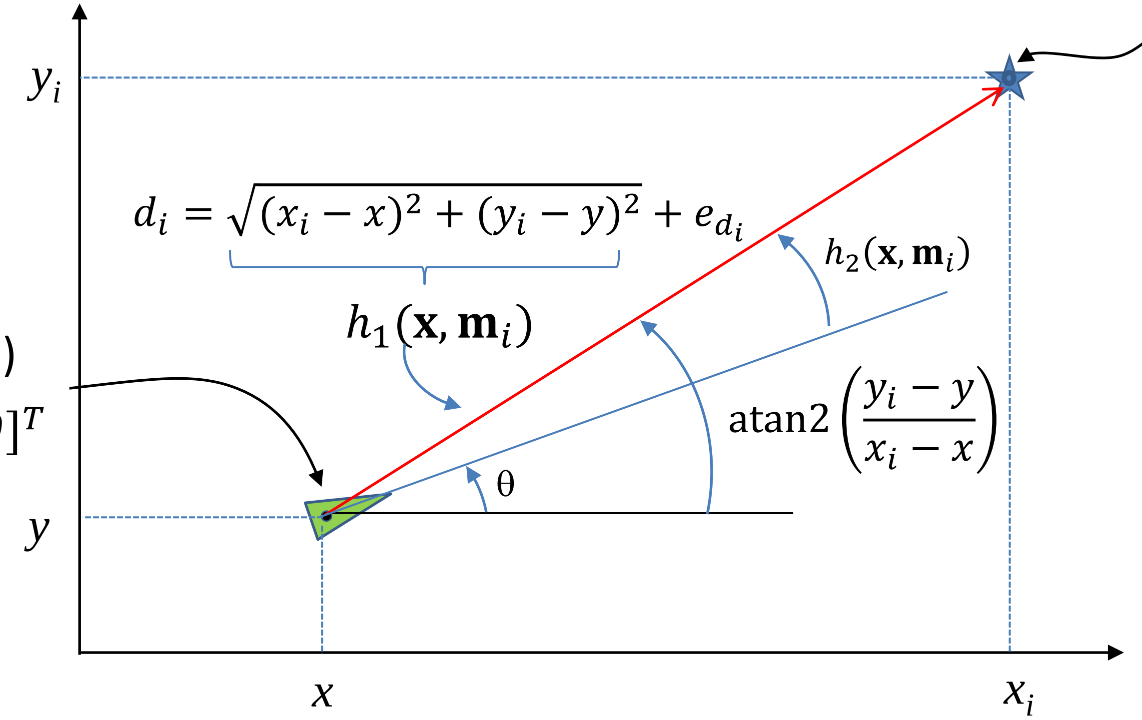
Estimate the position p of a vehicle in 2D, given distances d_i to *n* beacons m_i (sort of indoor GPS)



m_i: position of beacon i

$p=[x,y]^{\mathrm{T}}$: robot position

The range-bearing observation model (revisited)



Known position of the landmark $\boldsymbol{m}_i$

True robot pose (unknown)

$$\boldsymbol{x} = [\underbrace{x, y}_{p}, \theta]^T = [p, \theta]^T$$

Sensor error: $\mathbf{e}_i = \begin{bmatrix} e_{d_i} \\ e_{\theta_i} \end{bmatrix}$

Observation model

$$\mathbf{z}_i \equiv \begin{bmatrix} d_i \\ \theta_i \end{bmatrix} = h(\mathbf{x}, \mathbf{m}_i) + \mathbf{e}_i = \begin{bmatrix} \sqrt{(x_i - x)^2 + (y_i - y)^2} \\ \text{atan}(\frac{y_i - y}{x_i - x}) - \theta \end{bmatrix} + \mathbf{e}_i$$

The observations are the true values $h(\mathbf{x}, \mathbf{m}_i)$ (from the true pose to the true landmark position) plus a sensor error $\mathbf{e}_i$

Jacobian (wrt the robot pose $\boldsymbol{x}$)

$$\mathrm{J}_{h,\boldsymbol{x}} = \frac{\partial h}{\partial\{x, y, \theta\}} = \begin{bmatrix} -\frac{x_i - x}{d} & -\frac{y_i - y}{d} & 0 \\ \frac{y_i - y}{d^2} & -\frac{x_i - x}{d^2} & -1 \end{bmatrix}_{2x3}$$

Example 3 beacons: Trilateration (Range-only positioning) in 2D

$$\mathbf{z} = h(\mathbf{x}) + \boldsymbol{e} \xrightarrow{m=3} \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \end{bmatrix} = \underbrace{\begin{bmatrix} \|m_1 - \mathbf{x}\| \\ \|m_2 - \mathbf{x}\| \\ \|m_3 - \mathbf{x}\| \end{bmatrix}}_{\text{Distance of } \mathbf{x} \text{ to each landmark } m_i} = \begin{bmatrix} [(x_1 - x)^2 + (y_1 - y)^2]^{1/2} \\ [(x_2 - x)^2 + (y_2 - y)^2]^{1/2} \\ [(x_3 - x)^2 + (y_3 - y)^2]^{1/2} \end{bmatrix}$$

$$\boldsymbol{e} = \mathbf{z} - \boldsymbol{h}(\mathbf{x}) = \begin{bmatrix} d_1 - h_1(x) \\ d_2 - h_2(x) \\ d_3 - h_3(x) \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$$

Jacobian of $\boldsymbol{h}$ = - Jacobian of $\boldsymbol{e}$

$$J_h = -J_e$$

Least square solution:

$$\hat{x} = \arg\min_x e^T e = \arg\min_x [e_1^2 + e_2^2 + e_3^2]$$

Example 3 beacons: Range-only positioning in 2D (cont.)

$$\begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \end{bmatrix} = \begin{bmatrix} \|m_1 - \mathbf{x}\| \\ \|m_2 - \mathbf{x}\| \\ \|m_3 - \mathbf{x}\| \end{bmatrix} = \begin{bmatrix} [(x_1 - x)^2 + (y_1 - y)^2]^{1/2} \\ [(x_2 - x)^2 + (y_2 - y)^2]^{1/2} \\ [(x_3 - x)^2 + (y_3 - y)^2]^{1/2} \end{bmatrix}$$

Do not depend on the robot heading θ

Gradient of h_i:

$$\frac{\partial}{\partial x}[(x_i - x)^2 + (y_i - y)^2]^{1/2} = -\frac{1}{d_i}(x_i - x)$$

$$\frac{\partial}{\partial y}[(x_i - x)^2 + (y_i - y)^2]^{1/2} = -\frac{1}{d_i}(y_i - x)$$

$$\frac{\partial}{\partial \theta}[(x_i - x)^2 + (y_i - y)^2]^{1/2} = 0$$

Gradient of h_1

$$J_h = \nabla h = \begin{bmatrix} -\frac{1}{d_1}(x_1 - x) & -\frac{1}{d_1}(y_1 - y) & 0 \\ -\frac{1}{d_2}(x_2 - x) & -\frac{1}{d_2}(y_2 - y) & 0 \\ -\frac{1}{d_3}(x_3 - x) & -\frac{1}{d_3}(y_3 - y) & 0 \end{bmatrix} = -J_e$$

$J_h^T J_h$ is **not invertible** (rank =2) → as expected, the heading of the robot (θ) is not observable (can not be estimated)

Solution: Drop the unknown θ in the formulation

$$J_h = \begin{bmatrix} -\frac{1}{d_1}(x_1 - x) & -\frac{1}{d_1}(y_1 - y) \\ -\frac{1}{d_2}(x_2 - x) & -\frac{1}{d_2}(y_2 - y) \\ -\frac{1}{d_3}(x_3 - x) & -\frac{1}{d_3}(y_3 - y) \end{bmatrix} \qquad \delta = \left(J_h^T J_h\right)^{-1} J_h^T [z - h(\hat{x})]$$

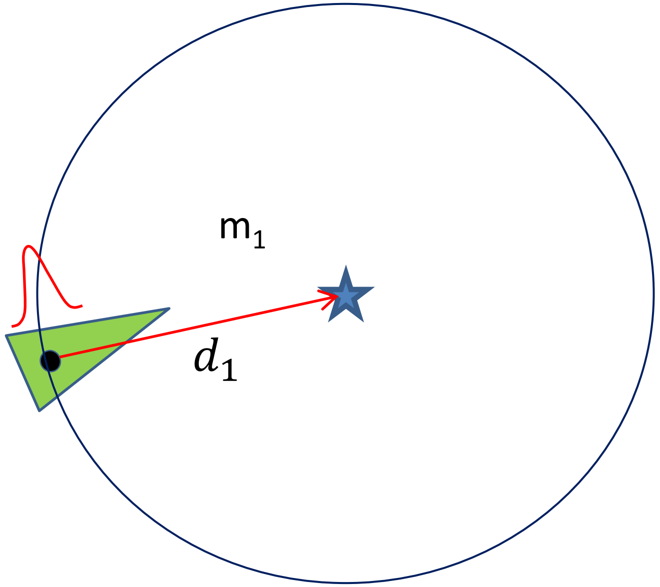
Example: Range-only positioning in 2D (cont.)

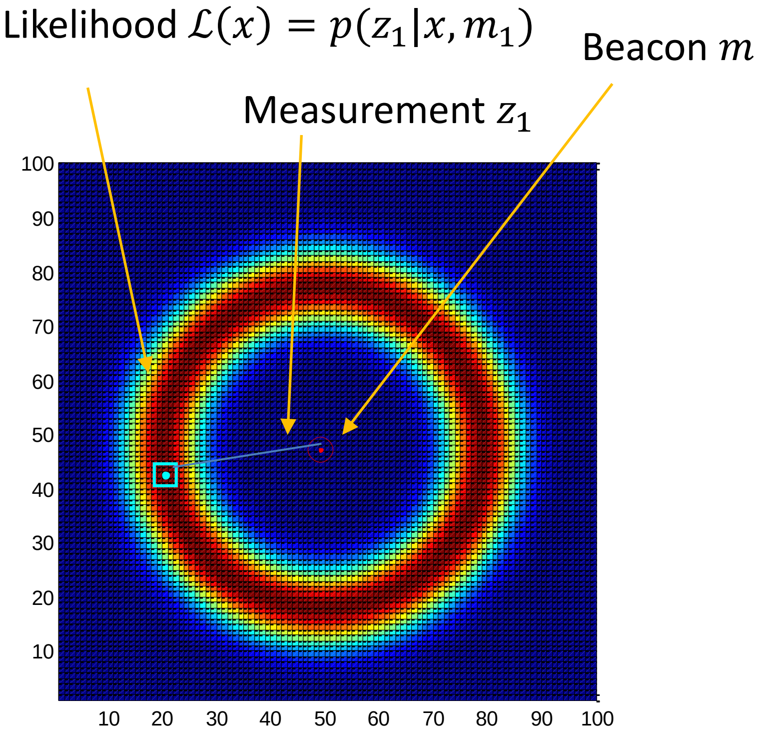
If just one landmark is observed, estimating $p = (x, y)$ has infinitely many solutions!

$$z_1 = d_1 = [(x_1 - x)^2 + (y_1 - y)^2]^{1/2} + e_d$$

The robot position will be on a circumference with radius d_1

With $e_d \sim N(0, \sigma^2)$, the robot position uncertainty will be given by a "gaussian donut" → **It is not a Gaussian distribution**





Bayes: $p(x|z_1, m_1) = kp(z_1|x, m_1)p(x|m_1)$

$if\, p(x|m_1)\ uniform$ → $p(x, |z_1, m_1) = k'p(z_1|x, m_1)$

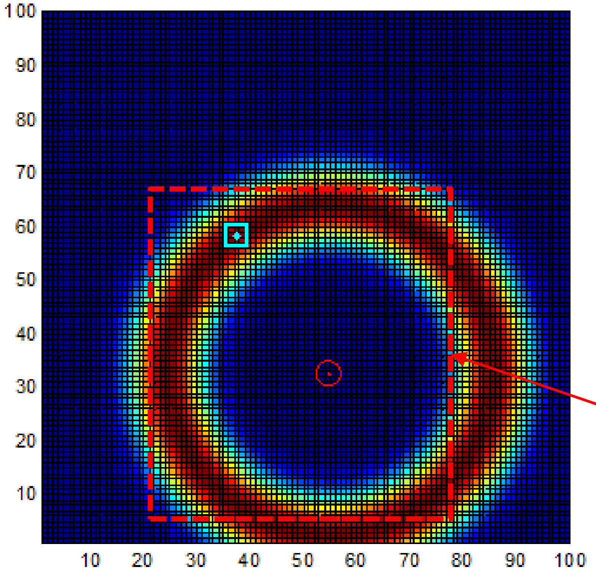
Notice: $p(z|x, m_1)$ is Gaussian BUT $\mathcal{L}(x) = p(z_1|x, m_1)$ IS NOT ("donut shape")

Two observed range combined

Robot pose (unknown)

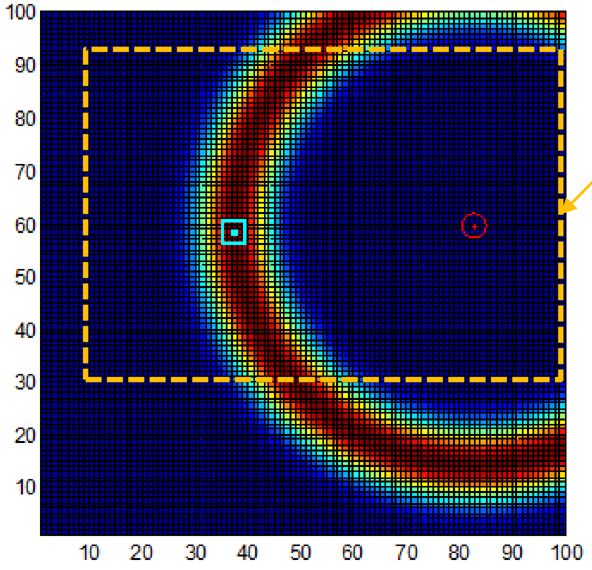
Beacons $m = \{m_1, m_2\}$

First range observation (to beacon B1)

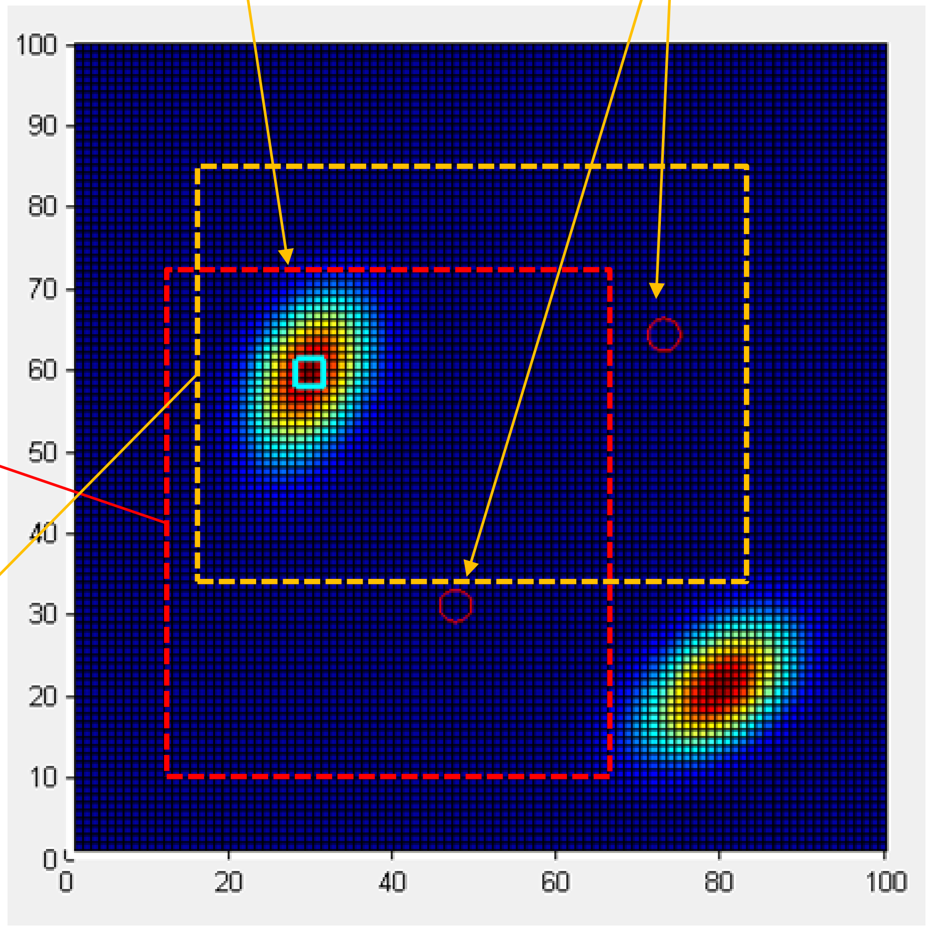


$p(z_1|x, m_1)$

Second range observation (to beacon B2)



$p(z_2|x, m_1)$



$if\, p(x|m)$ $(prior)$ $uniform$

$$p(x|\, z_1, z_2, m) = k' p(z_1, z_2|x, m) =$$

$$= k' p(z_1|x, m)\, p(z_2|x, m)$$

Conditional independence

Least Squares Positioning

Characteristics:

- Needs full observability of the state (pose) to give an estimation.
  - For example, in range-only positioning (2 unknowns: x, y), with only one observed landmark the problem becomes undetermined
  - However, using filtering techniques (i.e. Kalman filter) we can have one solution because a prior exists (explained later)
- If closed-form (non-iterative) solution exists:
  - Global minimum guaranteed → Very effective Global Localization

  otherwise

  - Must be solved iteratively
  - Global minimum NOT guaranteed
  - Requires a good initial guess (e.g. from the motion model)
- Recall: Least Squares is equivalent to ML Estimation if $p(z|x, m)$ is Gaussian

Map registration

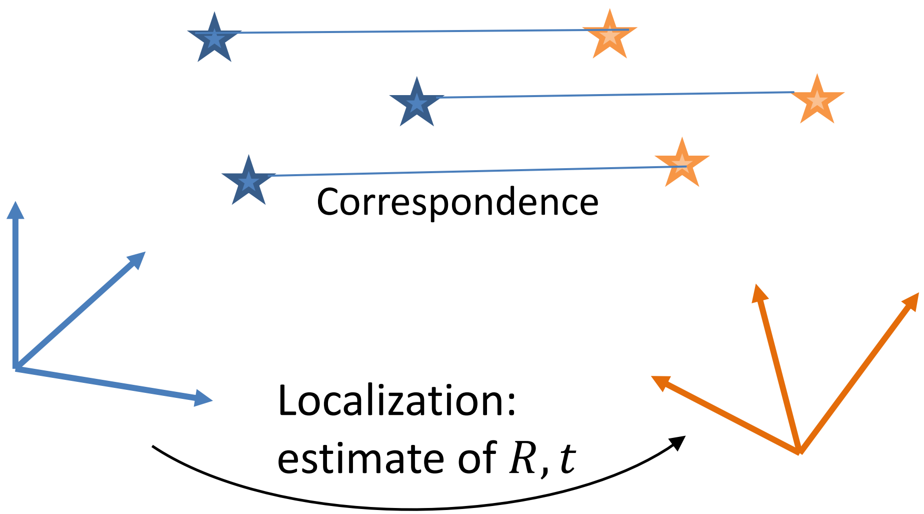
A set of observed world points are **registered (overlapped) against …**

- A global map → gives us the global pose of the observation (global localization)
- A previously observed point set → gives us their relative pose (odometry)

Two types of points:

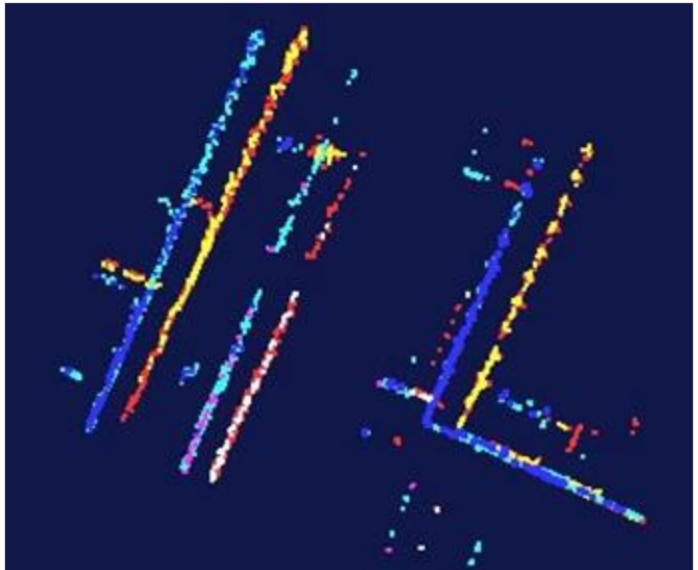
- Landmark points:

Correspondences are typically known (from landmarks descriptors)



- Scan points:

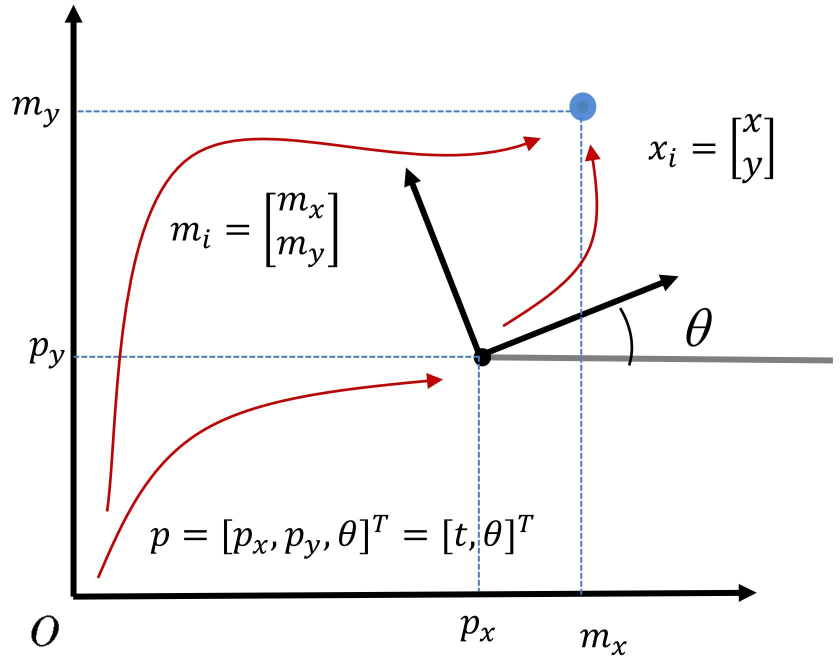
Correspondence between points are unknown



For **landmark points:** known correspondences assumed

Given two sets of points $M = \{m_1, \cdots, m_n\}$ and $X = \{x_1, \cdots, x_n\}$

where the pair (m_i, x_i) are corresponding points, i.e. points of the same physical entity, observed from different robot poses



$$p \oplus x = \begin{bmatrix} p_x + x\cos\theta - y\sin\theta \\ p_y + x\sin\theta + y\cos\theta \end{bmatrix} = t + Rx$$

Error for one point:

$$E_i(R,t) = \|m_i - p \oplus x_i\|^2 = \|m_i - Rx_i - t_i\|^2$$

Error for n points: $E(R,t) = \sum_{i=1}^{n} \|m_i - Rx_i - t\|^2$

Find the translation t and rotation R that minimize the sum of the squared error $E(R,t)$

R, t can be calculated iteratively (e.g. Gauss-Newton) or in closed-form (analytically)

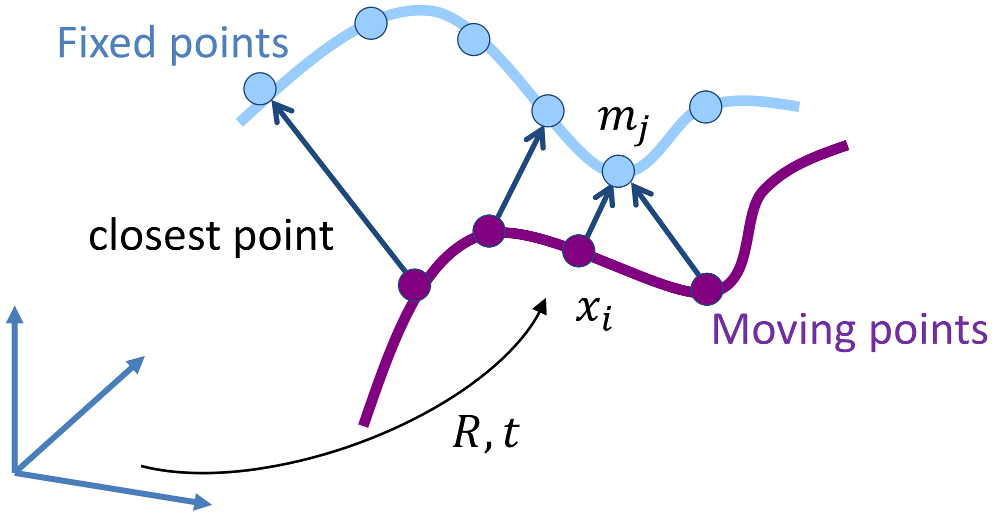
Scan Points Matching: Iterated Closest Point (**ICP**) Algorithm

Iterate until convergence:

1. For each point x_i in one set (e.g. scan points), make a pair with the **closest point** m_j in the other point set
2. Solve one step in the Gauss-Newton method for the minimization of:

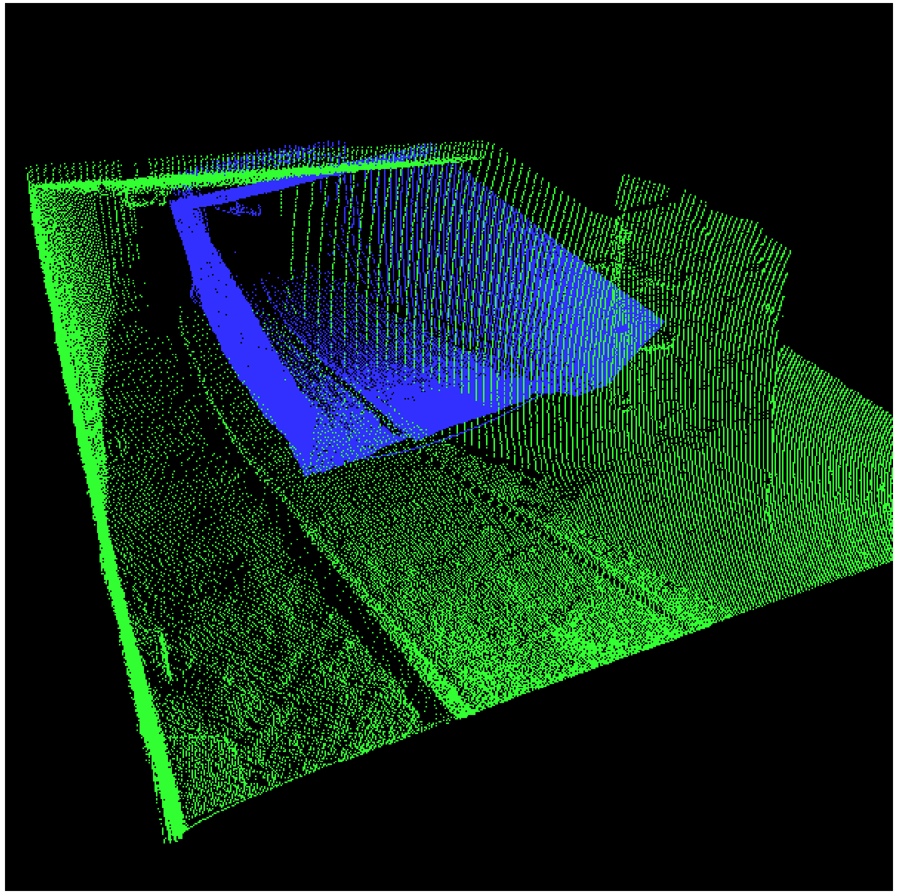
$$E(R,t) = \sum_{i=1}^{n} \left\| m_j - Rx_i - t_i \right\|^2$$

Convergence: computed R,t in step 2 are negligible or the pairs in step 1 do not change

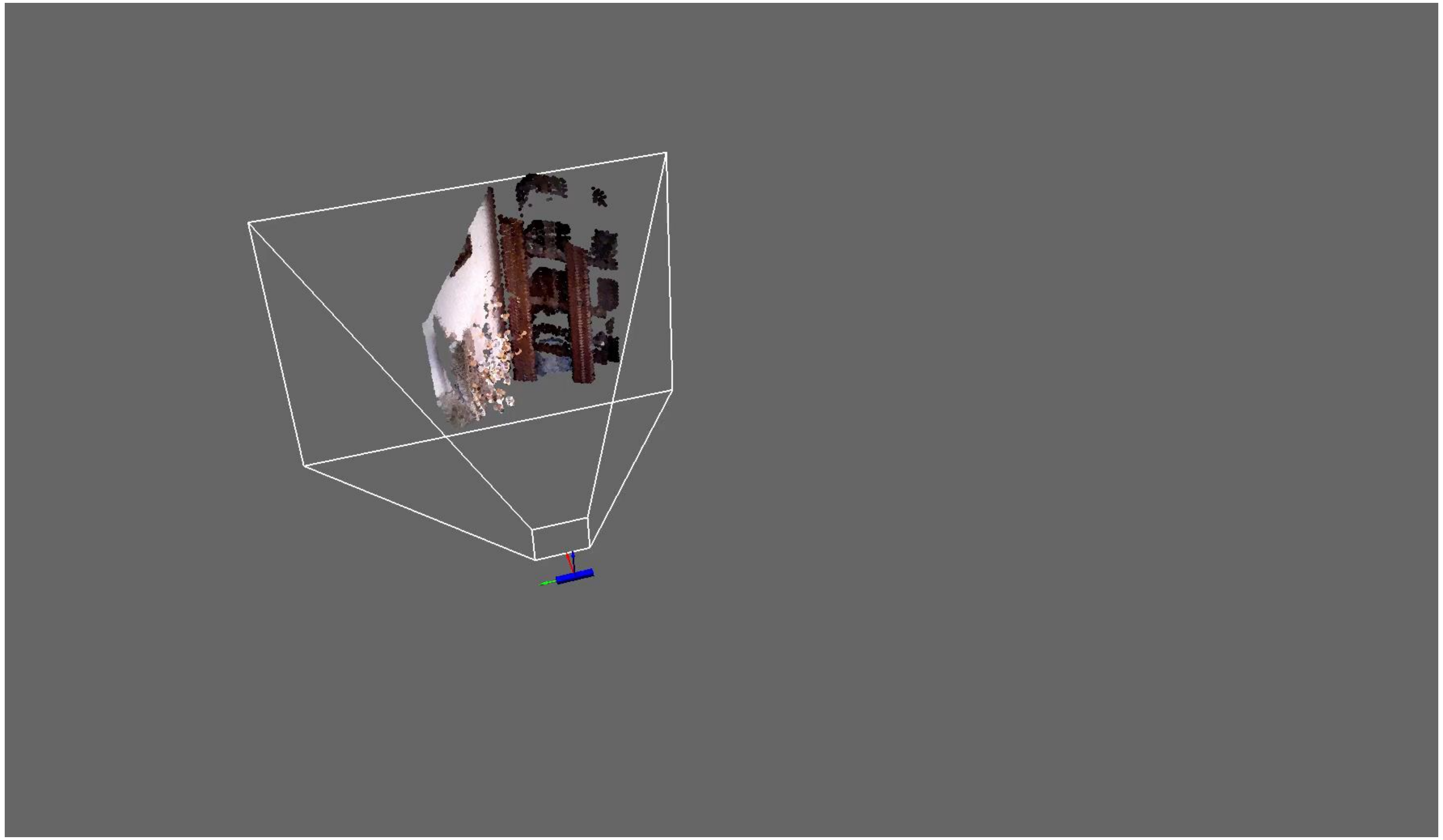


Closest-point matching is generally stable, but slow and requires preprocessing (KD-tree technique)

ICP: Also for 3D point clouds



[Nuechter et al., 2004]



[Jaimez & Gonzalez-Jimenez, 2015]

Discrete Kalman Filter (KF)

Kalman Filter is a method to overcome the occasional un-observability problem of the Least Squares approach thanks to a model of how the state changes

Based on the Bayes Filter (called **Markov localization**):

$$Belief(x_t) = p(x_t|u_1, z_1 \ \ldots, u_t, z_t) = p(x_t|u_{1:t}, z_{1:t})$$

Correction *Prediction* (prior)

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

KF computes this equation alternating between two steps::

- **Prediction** (computes prior): $\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) \, bel(x_{t-1}) \, dx_{t-1}$ Total probability (sum rule)
- **Correction** (computes posterior): $bel(x_t) = \eta p(z_t|x_t)\overline{bel}(x_t)$ Bayes rule

where:

1) The two process equations, $p(x_t|u_t, x_{t-1})$ (motion) and $p(z_t|x_t)$ (sensing) are **linear**

2) All pdfs involved are **Gaussians**: $p(x_t|u_t, x_{t-1}), p(z_t|x_t) \, , bel(x_0)$

RECALL 1 Bayes filter:

$$Belief(x_t) = p(x_t \mid u_1, z_1 \ldots, u_t, z_t) = p(x_t \mid u_{1:t}, z_{1:t})$$

$$z_{1:t} = z_1, \ldots, z_t$$
$$u_{1:t} = u_1, \ldots, u_t$$

Bayes $$= \eta \; p(z_t \mid x_t, \cancel{u_{1:t}, z_{1:t-1}}) \; p(x_t \mid u_{1:t}, z_{1:t-1})$$ ← without z_t

Markov $$= \eta \; p(z_t \mid x_t) \; p(x_t \mid u_{1:t}, z_{1:t-1})$$

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

Total prob. $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid \underline{u_{1:t}, z_{1:t-1}}, x_{t-1}) p(x_{t-1} \mid u_{1:t}, z_{1:t-1}) \, dx_{t-1}$$

Markov $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \; p(x_{t-1} \mid u_{1:t}, z_{1:t-1}) \, dx_{t-1}$$

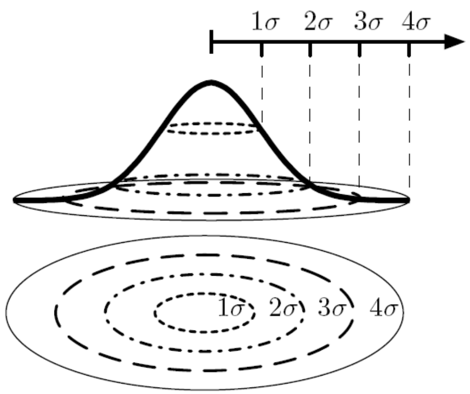
Markov $$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \boxed{p(x_{t-1} \mid u_{1:t-1}, z_{1:t-1})} \, dx_{t-1}$$

We don't apply Markov here because it is not needed. This is the definition of $Belief(x_{t-1})$

$$= \eta \; p(z_t \mid x_t) \int p(x_t \mid u_t, x_{t-1}) \; Belief(x_{t-1}) \, dx_{t-1}$$

RECALL 2: How Gaussians look like

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



RECALL 3:

- Product
- Marginalization/conditioning
- Linear transformation

of Gaussians pdf's is Gaussian

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

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

$$\left.\begin{array}{r} X_1 \sim N(\mu_1,\Sigma_1) \\ X_2 \sim N(\mu_2,\Sigma_2) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\big(\Sigma_{12}(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2),\quad \Sigma_{12} = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}\big)$$

KF: Let's go into the details

Kalman Filter is a general tool to estimate the state x of a process that is governed by the linear stochastic **model**:

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

$\varepsilon_t \sim N(0, R_t)$

Prediction phase

covariance of the linear motion model $B_t u_t$

with a linear measurement equation

$$z_t = C_t x_t + \delta_t$$

$\delta_t \sim N(0, Q_t)$

Correction phase

covariance of the linear measurement model $C_t x_t$

A_t Matrix (nxn) that describes how the state x evolves from t-1 to t without motion command (u_t)

B_t Matrix (nxl) that describes how the motion command u_t (lx1) changes the state x from t-1 to t

C_t Matrix (kxn) that describes how the state x_t is related to an observation z_t.

ε_t δ_t Random variables representing the motion and measurement noise. They are assumed to be independent and normally distributed with covariance R_t and Q_t, respectively.

Kalman Filter from Bayes filter

Posterior — *Likelihood* — $\overline{Bel}(x_t)$ *or prior*

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

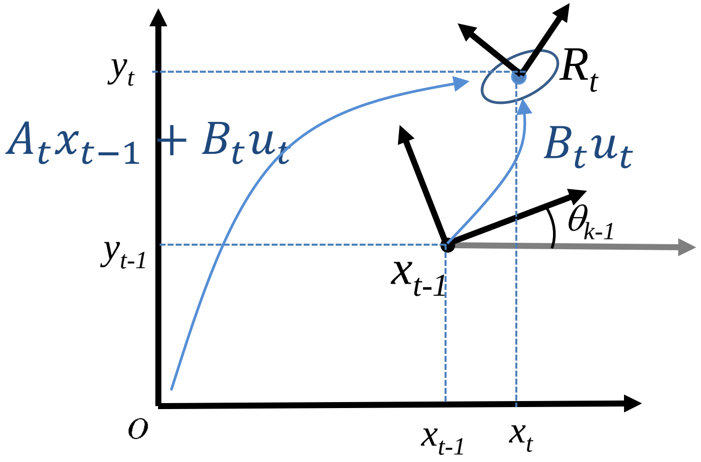
$N(x_t; \mu_t, \Sigma_t)$ — $N(z_t; C_t x_t, Q_t)$ — $N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)$

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

Motion model: Distribution over poses when executing the noisy motion command u_t and its pose is x_{t-1}

$\sim N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$

Distribution over poses x_{t-1}



$$\underbrace{Bel(x_t)}_{Posterior} = \eta \; \underbrace{p(z_t \mid x_t)}_{Likelihood} \underbrace{\int p(x_t \mid u_t, x_{t-1}) \, Bel(x_{t-1}) \, dx_{t-1}}_{\overline{Bel}(x_t) \text{ or prior}}$$

$N(x_t; A_t x_{t-1} + B_t u_t, R_t)$

Motion model: Distribution over poses when executing the noisy motion command u_t and its pose is x_{t-1}

$N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$

Distribution over poses x_{t-1}

Prediction phase (Prior computation):

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

This is another Gaussian, since marginalization and product of Gaussians is Gaussian:

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

u_t omitted for clarity

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

$$\underbrace{Bel(x_t)}_{\substack{Posterior}} = \eta \underbrace{p(z_t \mid x_t)}_{Likelihood} \int \underbrace{p(x_t \mid u_t, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}}_{\overline{Bel}(x_t) \text{ or prior}}$$

$$N(x_t; \mu_t, \Sigma_t) \qquad N(z_t; C_t x_t, Q_t) \qquad N(x_t; \bar{\mu}_t, \bar{\Sigma}_t)$$

Correction phase (posterior computation):

$$z_t = C_t x_t + \delta_t \quad \Rightarrow \quad p(z_t \mid x_t) = N(z_t; C_t x_t, Q_t)$$ Observation model

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

This is the multiplication of two Gaussians, which is also Gaussian

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

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

Prediction:

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

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

Correction:

$$K_t = \bar{\Sigma}_t C_t^T \left(\boxed{C_t \bar{\Sigma}_t C_t^T + Q_t}\right)^{-1}$$

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

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

Return μ_t, Σ_t

More uncertainty: The predicted covariance is the sum of those of the two RVs involved: R_t for $B_t u_t$ and $A_t \Sigma_{t-1} A_t^T$ for x_{t-1}

Covariance of the **Innovation**

This is called: **Innovation:** error between the observation and the prediction of the landmark

Less uncertainty: $\det(\Sigma_t) < \det(\bar{\Sigma}_t)$

Discrete Kalman Filter

Summary

- Highly efficient: Polynomial complexity in dimensionality of the measurement *k* and state dimensionality *n* (*n*=3 for 2D pose):

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

- Gives an optimal estimate for linear Gaussian systems
- Unfortunately, most robotics models (motion and sensing) are nonlinear ☹, and hence, KF can not be applied …

BUT we can linearize the motion and sensing model → Extended KF

Extended Kalman Filter

- EKF is an adaptation of KF to Nonlinear Dynamic Systems

Model transition (**prediction**): $\bar{x}_t = x_{t-1} \oplus u_t + \varepsilon_t$ (Instead of $\bar{x}_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$)

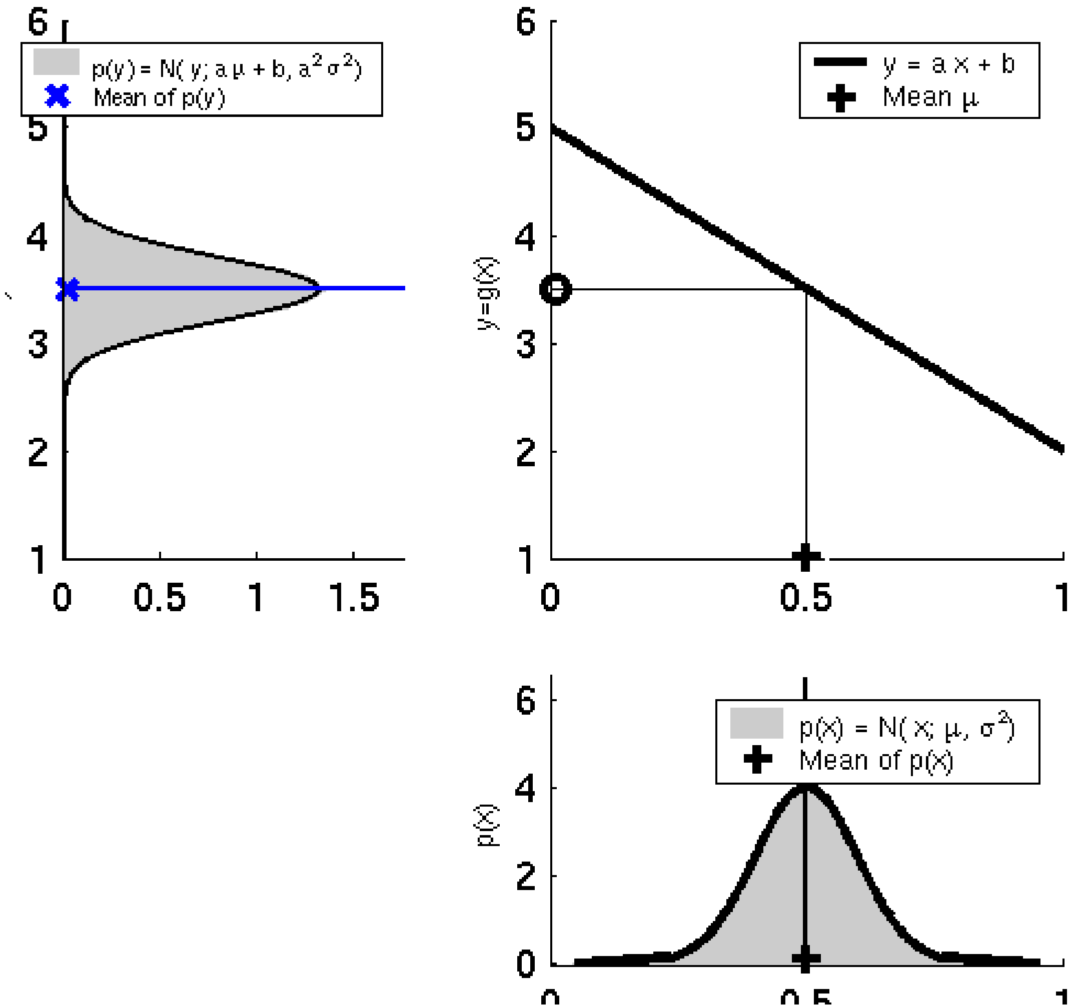
Observation function (**correction**): $z_t = h(x_t) + \delta_t$ (Instead of $z_t = C_t x_t + \delta_t$)

PROBLEM: Transformed Gaussians are no longer Gaussians

The robot pose estimate can not be represented in a parametric form (mean, covariance)

Watch out: $\bar{x}_t$ is NOT the mean of x_t, but the **predicted** x_t (before the observation z_t)

Linearity Assumption Revisited (lecture 2)

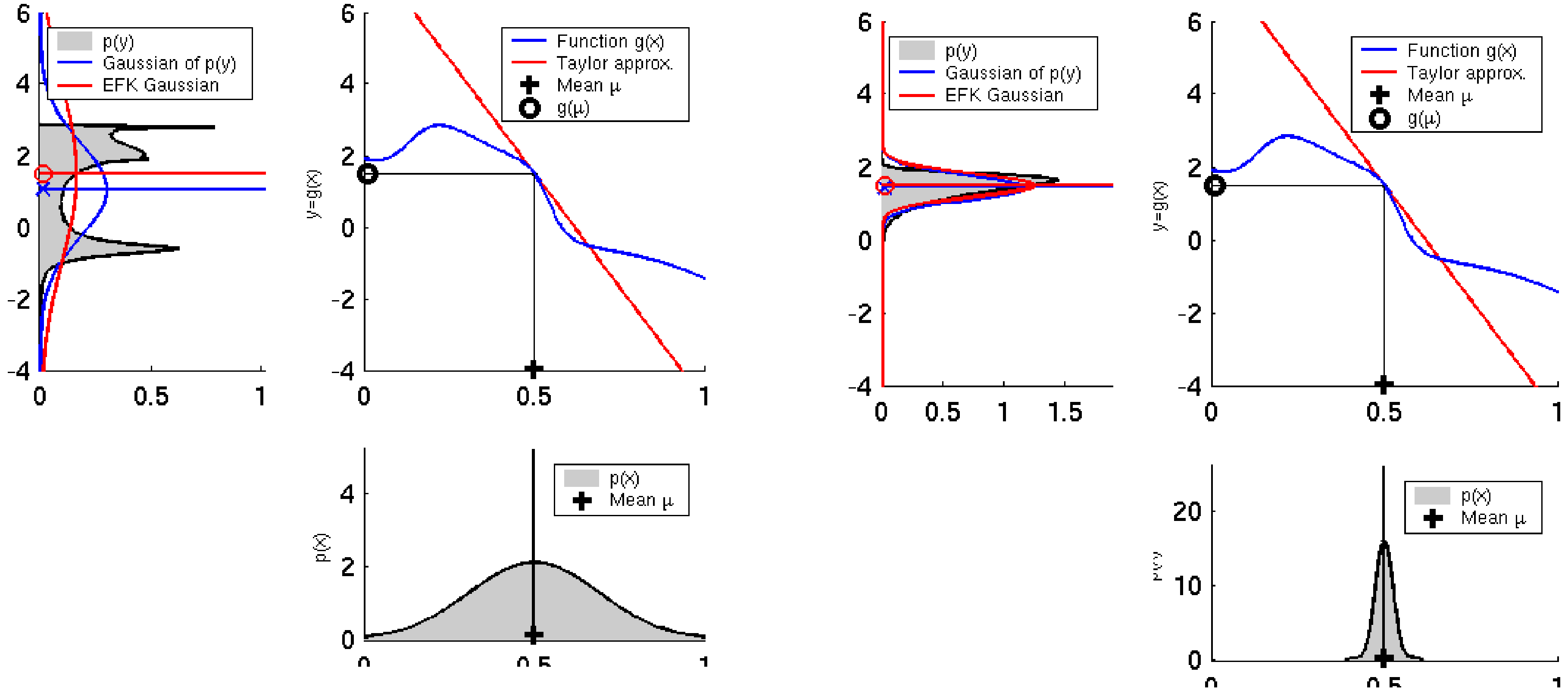


$$\left.\begin{aligned} X &\sim N(\mu,\Sigma) \\ Y &= AX + B \end{aligned}\right\} \Rightarrow \quad Y \sim N(A\mu + B, A\Sigma A^T)$$

From: Thrun *et.al.*

Linearity Assumption Revisited (lecture 2)

The pdf of y=f(x) is not well approximated by a Gaussian



Large variance of x

Small variance of x

For the same linearization of the function f(x), **much more error in the gaussian approximation when covariance large**!!

From: Thrun *et.al.*

EKF Linearization: First Order Taylor Expansion

- **Prediction:** $\bar{x}_t = x_{t-1} \oplus u_t + \varepsilon_t = g(x_{t-1}, u_t) + \varepsilon_t$

mean of x_{t-1}

The Jacobian is evaluated here, the mean of x_{t-1}

$$g(x_{t-1}, u_t) \approx g(\mu_{t-1}, u_t) + \underbrace{\frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}}_{\text{Jacobian G}} \underbrace{(x_{t-1} - \mu_{t-1})}_{\Delta x_{t-1}} = g(\mu_{t-1}, u_t) + G \; \Delta x_{t-1}$$

Linearization for x_{t-1} at μ_{t-1}

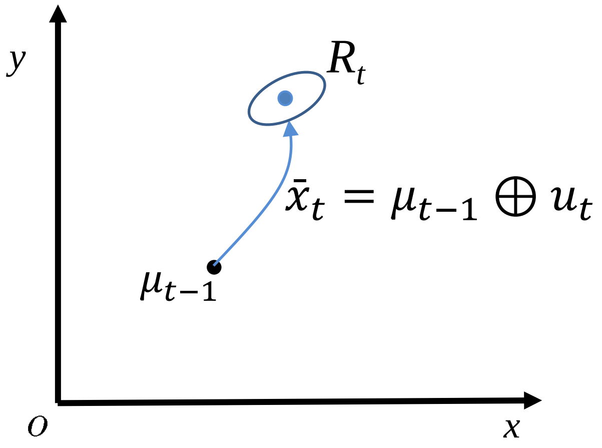
$$\bar{x}_t = g(x_{t-1}, u_t) + \varepsilon_t \approx \underbrace{g(\mu_{t-1}, u_t)}_{\mu_{t-1} \oplus u_t} + G \; \Delta x_{t-1} + \varepsilon_t$$

Random variables $\Delta x_{t-1} \sim N(0, \Sigma_{t-1})$ $\varepsilon_t \sim N(0, R_t)$

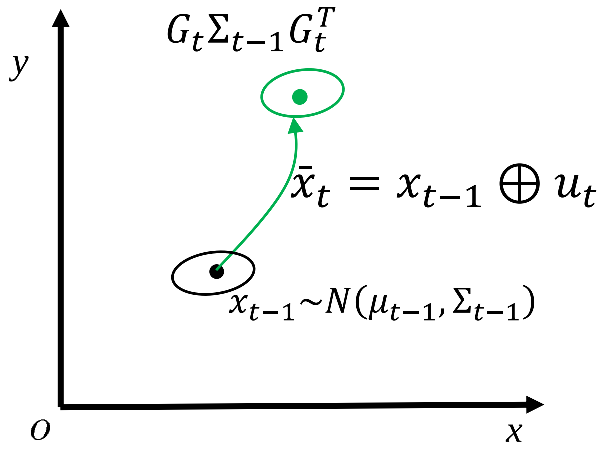
Parameters of the Normal distribution of $\bar{x}_t$:

$$\bar{x}_t \sim N(\bar{\mu}_t, \bar{\Sigma}_t) = N(\mu_{t-1} \oplus u_t, G_t \Sigma_{t-1} G_t^T + R_t)$$

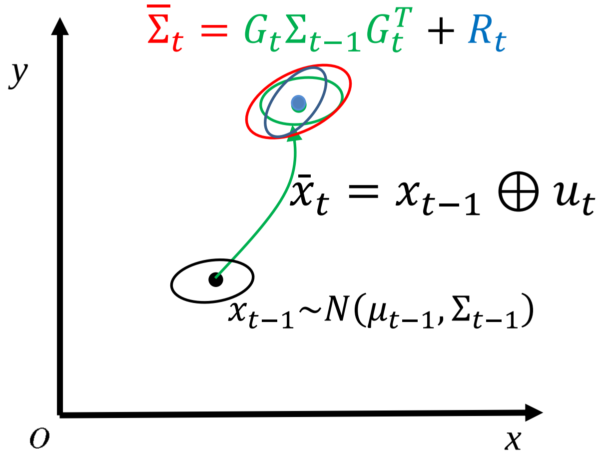
EKF Prediction Step



Motion of a **known pose $\boldsymbol{\mu_{t-1}}$** by $u_t \sim N(\bar{u}_t, R_t)$



Motion of the pose $x_{t-1} \sim N(\mu_{t-1}, \Sigma_{t-1})$ by a **known motion** u_t



Motion of the pose $x_{t-1} \sim N(\mu_{t-1}, \Sigma_{t-1})$ by $u_t \sim N(\bar{u}_t, R_t)$

EKF Linearization: First Order Taylor Expansion

- **Correction:**

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$$

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

Predicted mean (computed in the prediction step: $\mu_{t-1} \oplus u_t$)

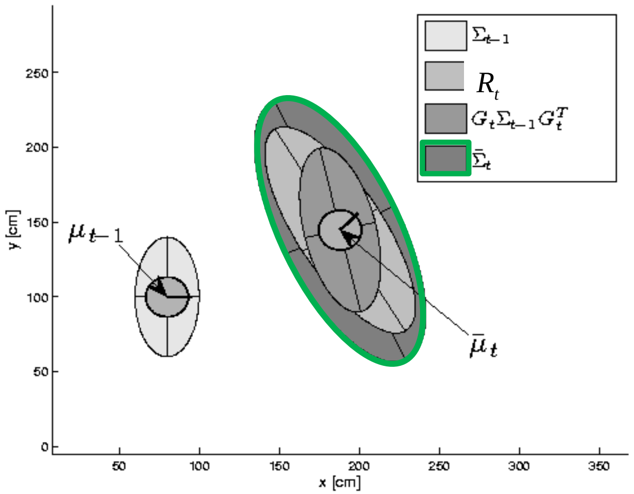
$$Bel(x_t) = \eta \underbrace{\exp\left\{-\frac{1}{2}(z_t - h(x_t))^T Q_t^{-1}(z_t - h(x_t))\right\}}_{\textit{Likelihood}} \underbrace{\exp\left\{-\frac{1}{2}(x_t - \bar{\mu}_t)^T \bar{\Sigma}_t^{-1}(x_t - \bar{\mu}_t)\right\}}_{\textit{Prior (Prediction)}}$$

Multiplication of two Gaussians = another Gaussian with a smaller Covariance

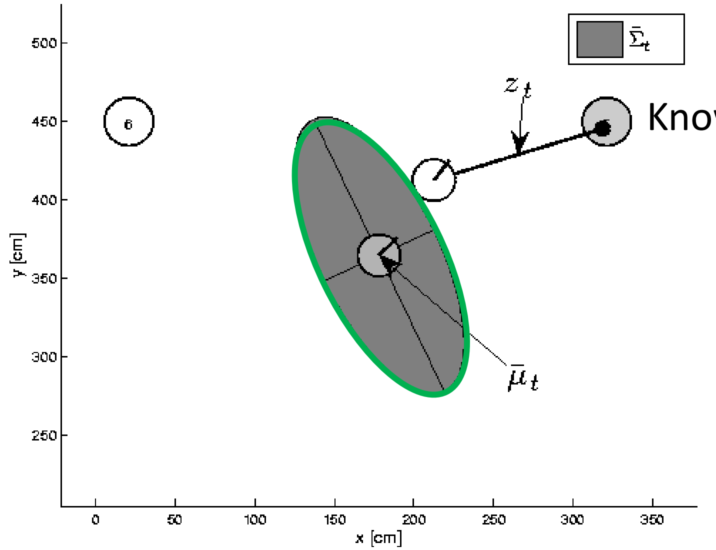
$$Bel(x_t) = N\left(x_t; \mu_t, \Sigma_t\right) \quad \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \\ \Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \\ K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \end{cases}$$

EKF Correction Step

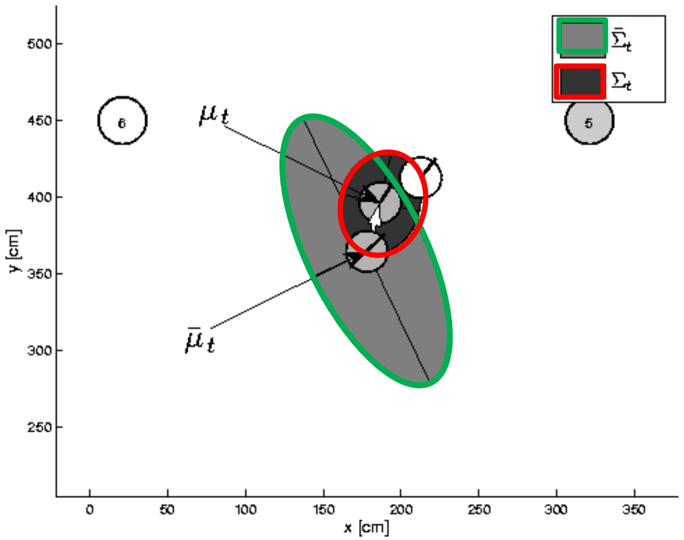
Prediction estimate



Take observation z_t



Known landmark



Correction estimate

Correction

From: Thrun *et.al.*

Extended Kalman Filter

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

Prediction:

1. $\bar{\mu}_t = g(\mu_{t-1}, u_t) = \mu_{t-1} \oplus u_t$
2. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$

Correction:

1. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
2. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
3. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$

Return μ_t, Σ_t

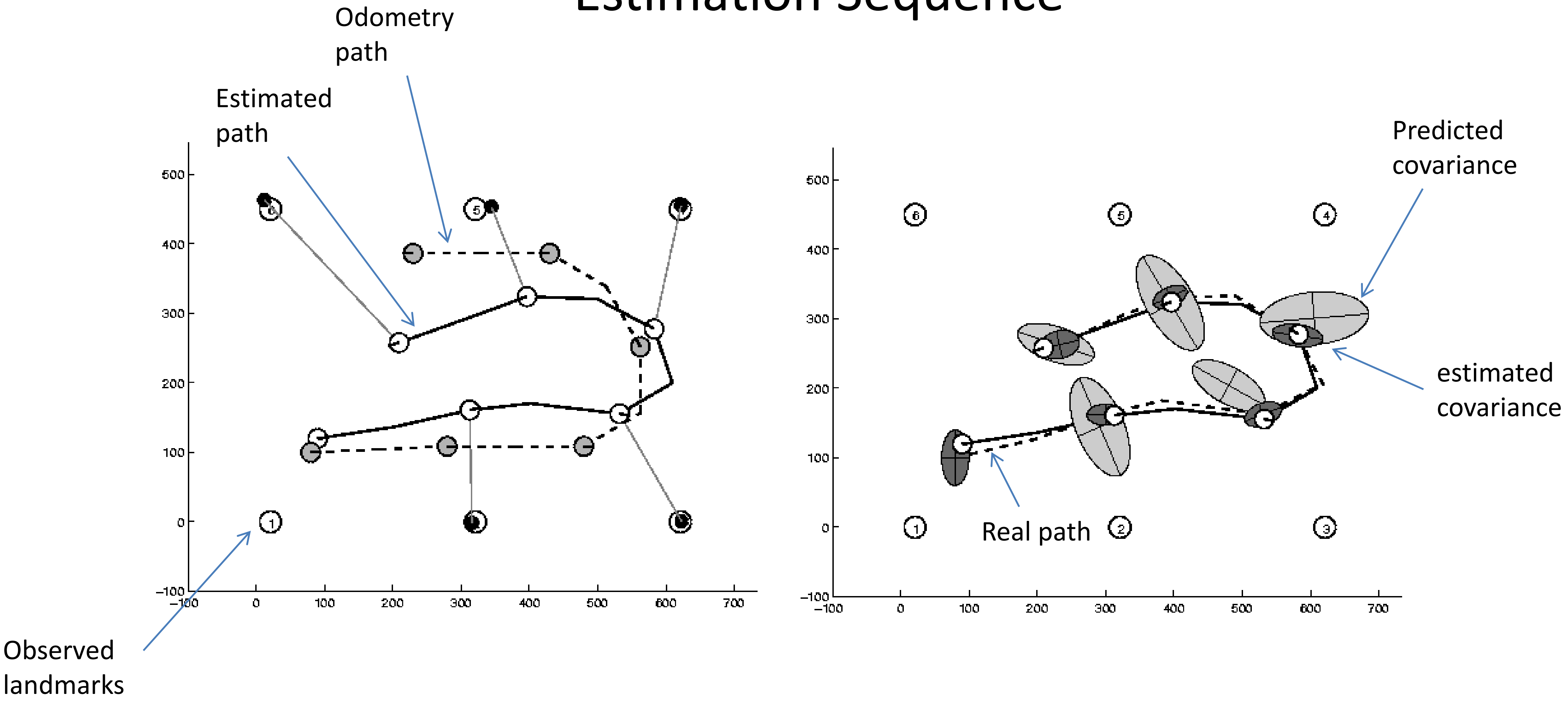
Jacobians

$$G_t = \frac{\partial g(\mu_{t-1}, u_t)}{\partial x_{t-1}}$$

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$$

Extended Kalman Filter

Estimation Sequence



We'll program this example in the practical sessions

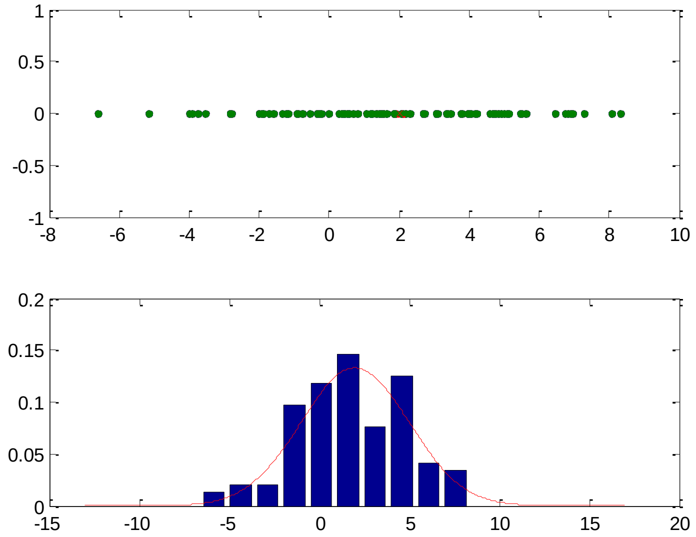
Particle Filter

- Technique for implementing recursive Bayesian filter by Monte Carlo sampling

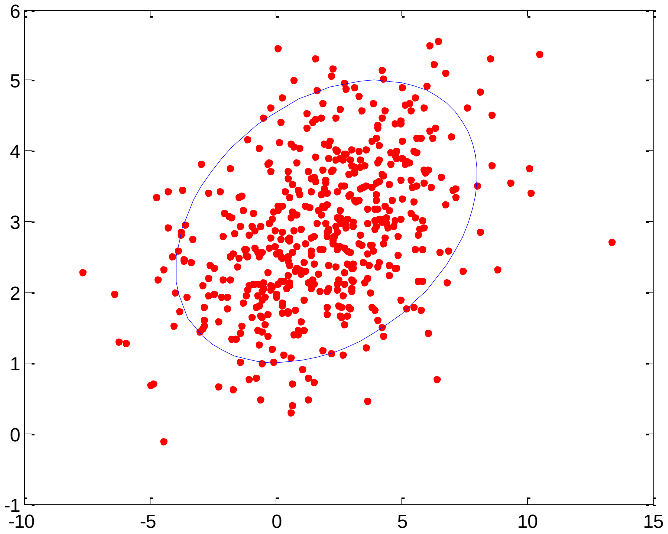
- **Idea**: to represent the posterior density by a set of k random samples (particles). The robot pose follows the distribution given by the samples

Example for a Gaussian pdf:

1D



2D



Particle Filter

- Still an implementation of the Bayes Filter

$$Bel(x_t) = \eta p(z_t|x_t)\overline{Bel}(x_t) = \eta p(z_t|x_t)\overline{Bel}(g(u_t, x_{t-1}))$$

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

Particles x^i_t drawn from *prior*

Importance factor (weight) for x^i_t: $w^i_t \propto p(z_t \mid x_t)$

- Bayes Filter for pdf given by samples

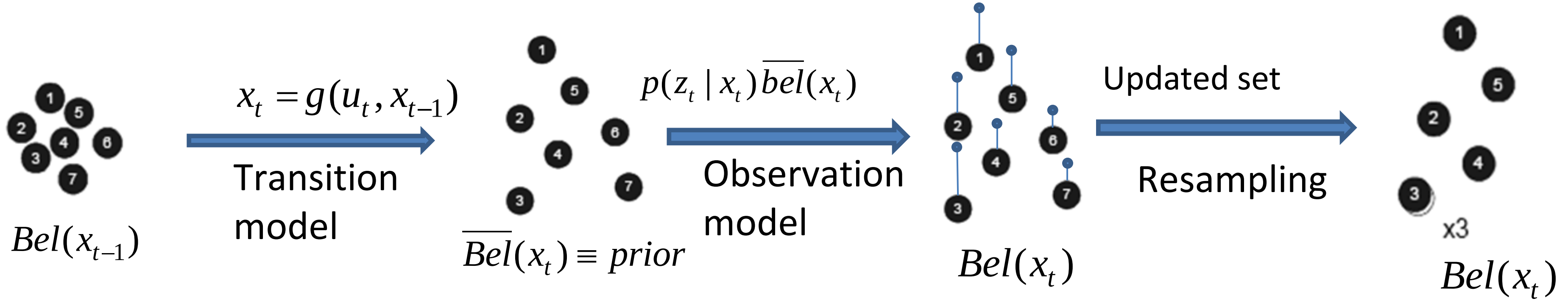
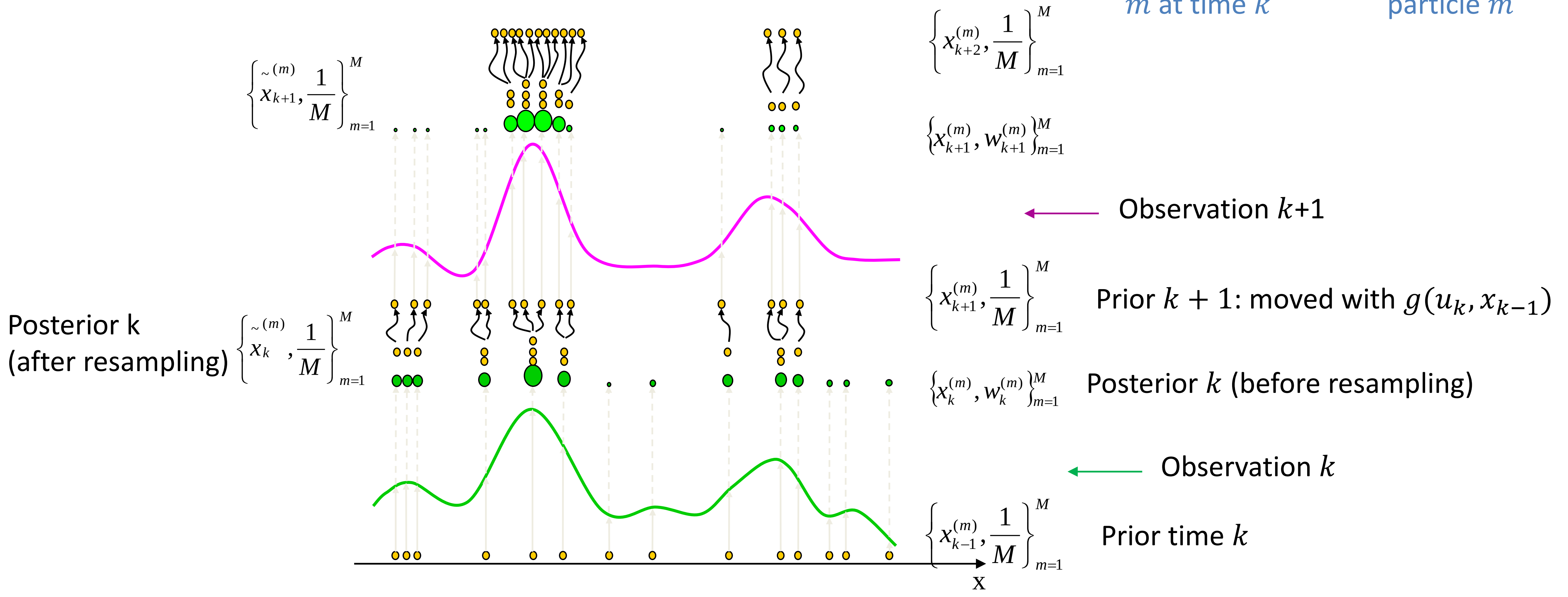


Illustration of the PF in 1D: SIR implementation, M particles: $\left\{x_{k-1}^{(m)}, w^{(m)}\right\}_{m=1}^{M}$

Pose of the particle m at time k

Weight of the particle m

Weights here are represented by the area of the circle

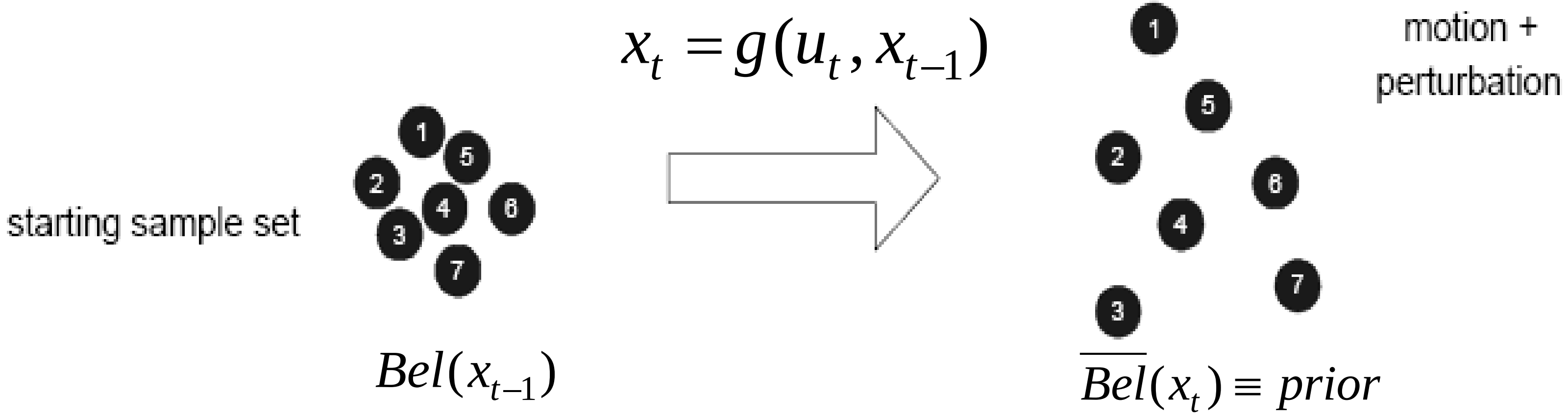


SIR: Sequential Importance Resampling

Particle Filter

1. Prediction

Implements the prediction $\overline{Bel}\ (g(u_t, x_{t-1}))$ in a sample-based form

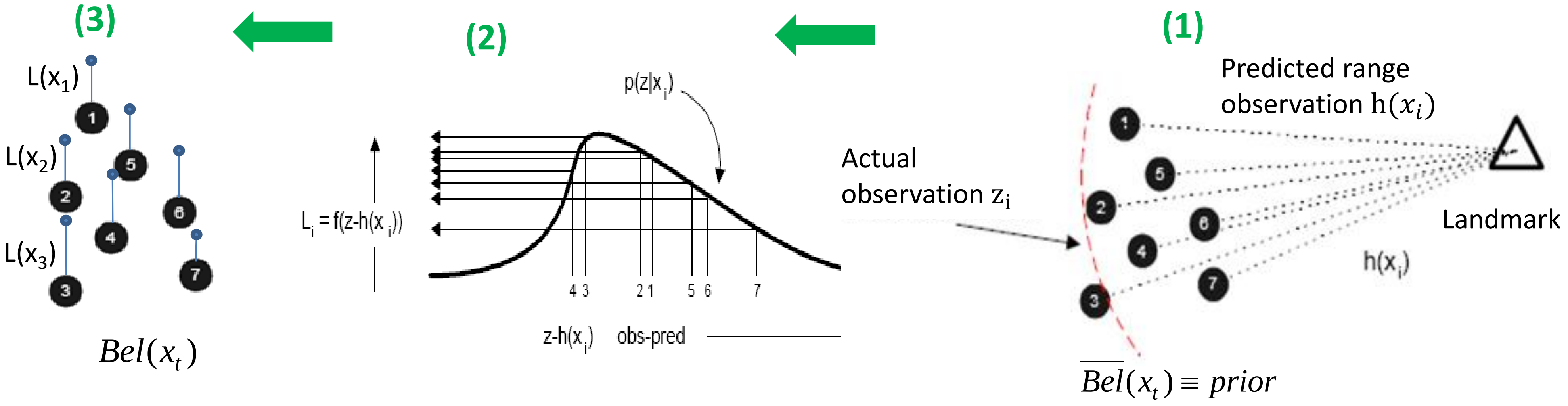



```
% xP (3xn): Vector with the pose (x,y,θ) of the n particles
% U: covariance matrix 3x3 of the error motion
% Each particle p moves with the motion vector plus noise
   for(p = 1:nParticles)
       xP(:,p) = compose(xP(:,p),u+sqrt(U)*randn(3,1));
   end;
```


Courtesy: P. Newman

Particle Filter

2. Update. a) Observation prediction




```
% (1)Compute predicted observation for each particle x
      zPred = DoObservationModel(xP(:,x),iFeature,Map);
% (2)Compute innovation
      Innov = z-zPred;
% (3)Get likelihood (new weight)
      L(x) = exp(-0.5*Innov'*inv(REst)*Innov)+0.001;
```


Courtesy: P. Newman

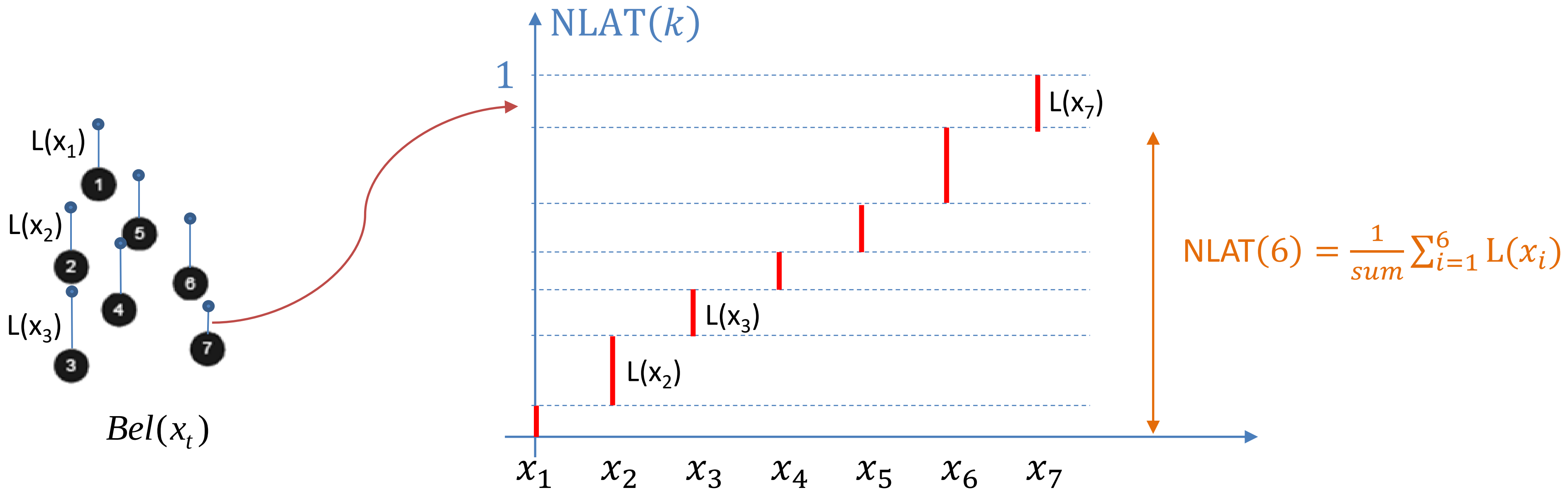
2. Update. b) Resampling based on weights (survival of the fittest)

IDEA: Particles with big weights will occupy a greater percentage of the y axis in a Likelihood accumulated Look-up table (LAT)

b1) Build a Look-up table for the Likelihood accumulation

Normalized Likelihood accumulated table

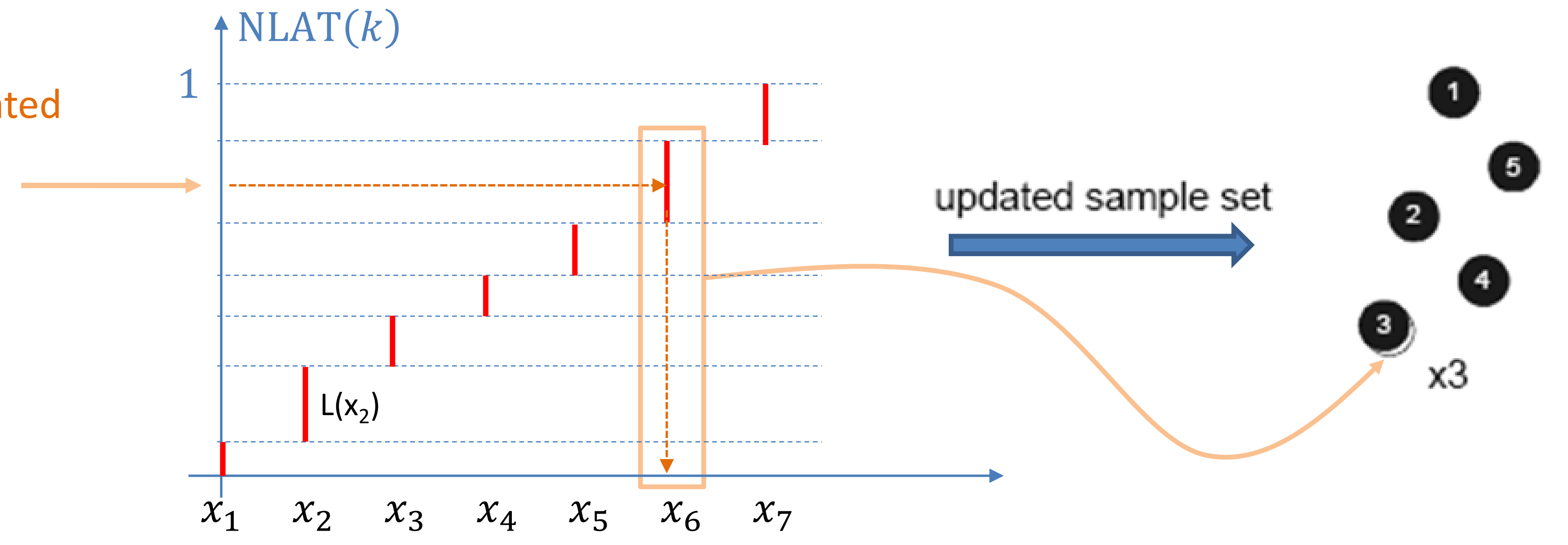
$$\mathrm{NLAT}(k) = \frac{1}{sum} \sum_{i=1}^{k} \mathrm{L}(x_i) \qquad sum = \sum_{i=1}^{np} \mathrm{L}(x_i)$$



2. Update. b) Resampling based on weights (survival of the fittest)

b2) Randomly generate new particles from the weights

Sample generated from U[0,1]




```
%LAT(k), table with accumulated likelihood
        NLAT = LAT/sum;
%Randomly (uniform) choose a value y in [0,1] for all the particles
        iSelect  = rand(nParticles,1);
%Find particle that corresponds to each y value (just a look up)
    iNextGeneration = interp1(NLAT,1:nParticles,iSelect,'nearest','extrap');
%copy selected particles for next generation..
    xP = xP(:,iNextGeneration);
```

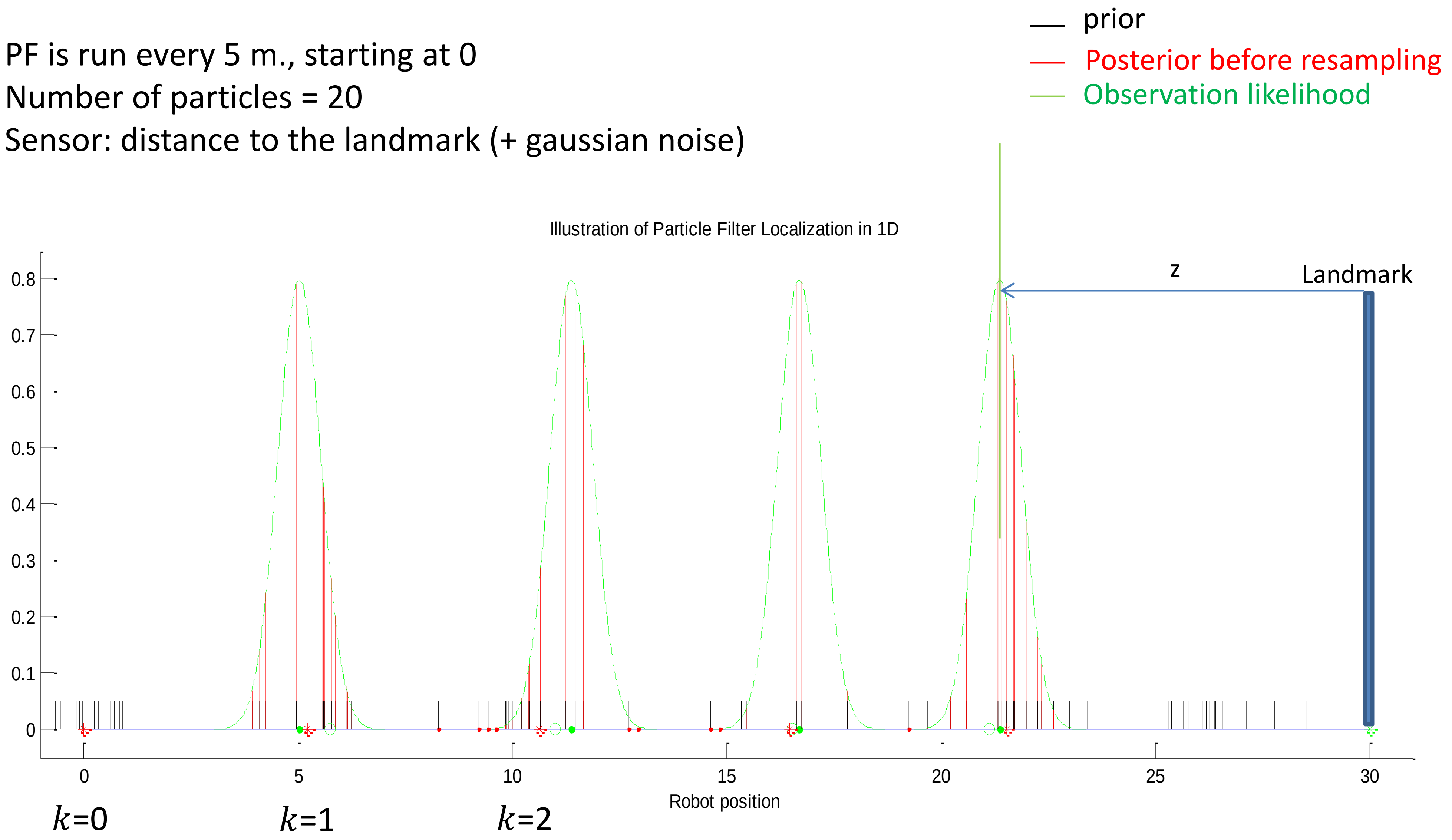
Courtesy: P. Newman

Example in 1D: Position tracking. A robot moves in a corridor with a range sensor that gets the distance to ONE landmark at the end of the corridor

PF is run every 5 m., starting at 0

Number of particles = 20

Sensor: distance to the landmark (+ gaussian noise)



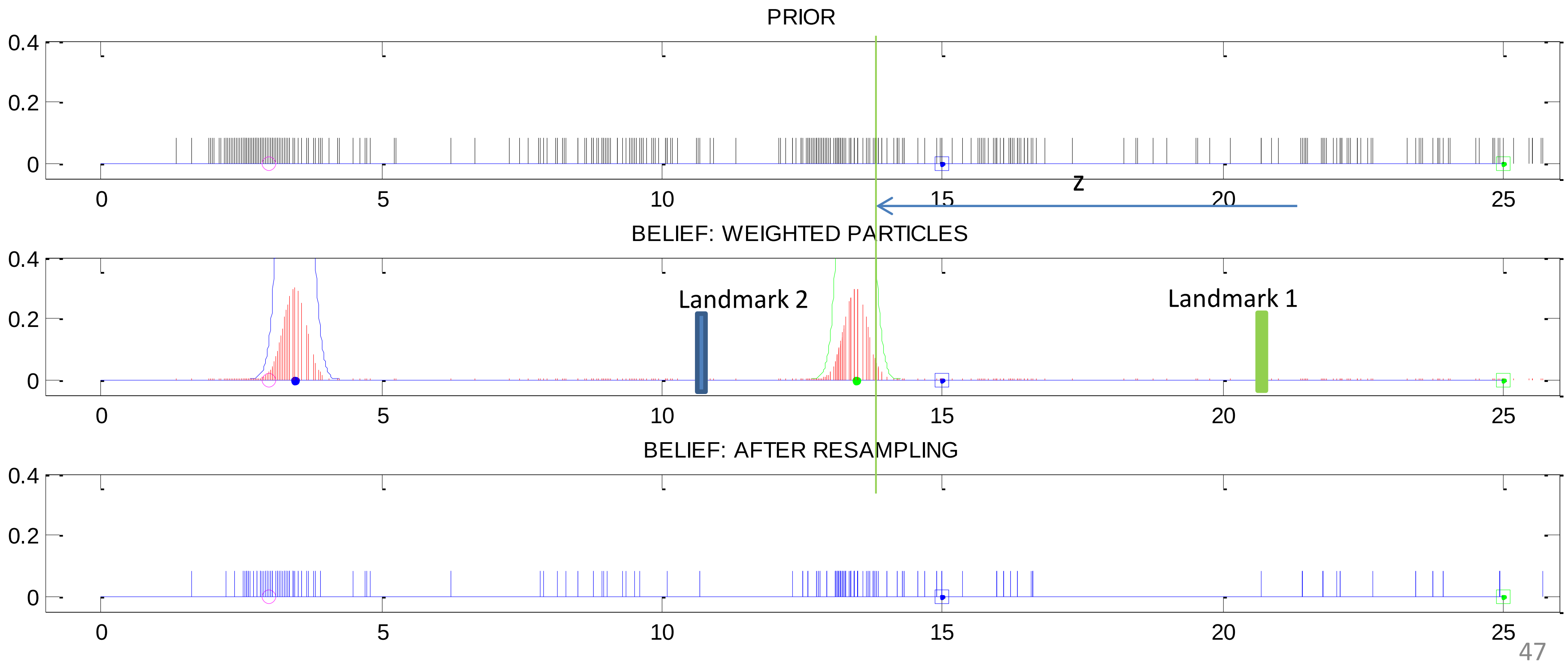
Example in 1D: Global localization

2 Landmarks, 1 observation

Number of particles = 500

— prior
— posterior
— Gaussian Observation likelihood to landmark 1
— Gaussian Observation likelihood to landmark 2

Given a measurement of z = 12 m we have to main hypothesis of the pose



Particle Filter

Advantages

- Model any probability distribution
- Model multimodal distributions (multiple hypothesis)
- Very simple implementation

Disadvantages

- Not as accurate as continuous random variable (it’s an approximation). The more particles the more accurate
- Sometimes the PF fails: degeneracy problems
- Computational complexity: assumable for states of small dimension (3D), e.g. localization in 2D