



Robotics - Localization - E.K.F.

Simone Ceriani

ceriani@elet.polimi.it

Dipartimento di Elettronica e Informazione
Politecnico di Milano

7 June 2012

Outline

- 1 Kalman Filter
- 2 K.F. Example
- 3 E.K.F.
- 4 EKF Loc. - Prediction
- 5 EKF Loc. - Update
- 6 Correspondences
- 7 Monte Carlo Localization - Particle



Outline

1 Kalman Filter

2 K.F. Example

3 E.K.F.

4 EKF Loc. - Prediction

5 EKF Loc. - Update

6 Correspondences

7 Monte Carlo Localization - Particle

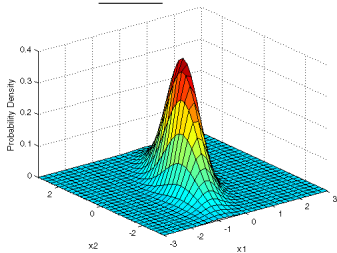


Gaussian Distribution Reminder

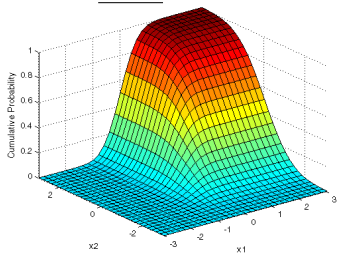
MULTIVARIATE GAUSSIAN DISTRIBUTION

- \mathbf{x} is a vector
- $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with
 - $\boldsymbol{\mu}$: $n \times 1$, mean vector
 - $\boldsymbol{\Sigma}$: $n \times n$, covariance matrix
- Linear transformation:
 - $X \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
 - $Y = \mathbf{A}X + \mathbf{b} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$

P.D.F. - $n = 2$



C.D.F. - $n = 2$



Kalman Filter - Introduction

KALMAN FILTER

- Was introduced by R. E. Kálmán in 1960
- It is a technique for optimal filtering and *prediction* in *Linear Gaussian System*
- Implements belief computation
 - for continuous state
 - distributed as a multivariate Gaussian
 - i.e., belief is represented by μ and Σ
- The best studied technique for implementing Bayes Filters

Kalman Filter - Algorithm

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

$$\begin{aligned} \textcircled{1} \quad & \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \\ K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1} \end{cases} \\ \textcircled{2} \quad & \begin{cases} \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases} \\ & \text{return } \mu_t, \Sigma_t \end{aligned}$$

TWO STEP ALGORITHM**①** First Step - Prediction

- Calculate the $\overline{bel}(x_t)$:
mean ($\bar{\mu}$) and covariance ($\bar{\Sigma}$)
- Application of Gaussian properties

② Second Step - Update

- Calculate the $bel(x_t)$:
mean (μ) and covariance (Σ)
- Using the *Kalman Gain* (K_t)
on the *innovation*,
i.e. difference of
 - Measure: z_t
 - Expected Measure: $C_t \mu_t$

Outline

1 Kalman Filter

2 **K.F. Example**

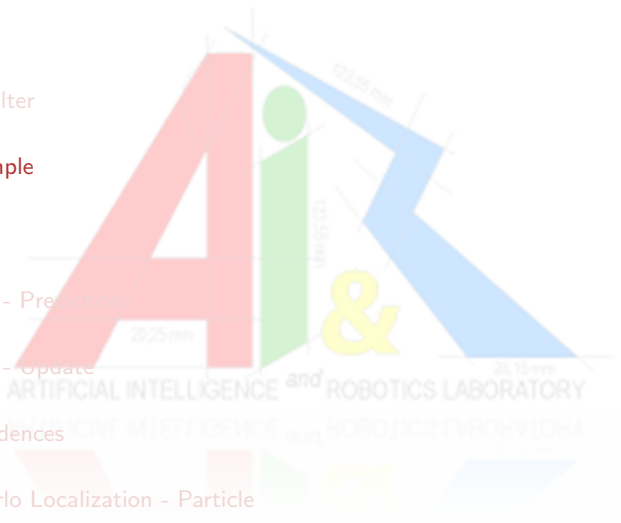
3 E.K.F.

4 EKF Loc. - Prediction

5 EKF Loc. - Update

6 Correspondences

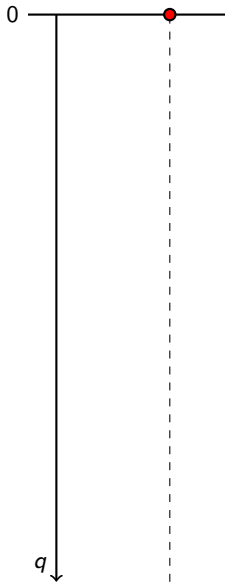
7 Monte Carlo Localization - Particle



Kalman Filter - Example - Problem

FALLING BODY

- Start at 0m with 0 m/s speed
- Standard deviation on position is 0[m]
- Standard deviation on speed is 0[m/s]
- A sensor measure the altitude in millimeters with a stochastic error of 100mm
- During the fall, stochastic errors affect
 - altitude, with std. dev. of 0.01m
 - speed, with std. dev. of 0.005m/s



Kalman Filter - Example - Definitions

DEFINITIONS

- State: $x_t = [q_t, v_t]^T$, altitude and speed
- Input: $u_t = g$, gravity
- Constants
 - $b = 0.0025$, friction coefficient
 - $\Delta t = 0.001s$, period of discrete time step

INITIAL STATE

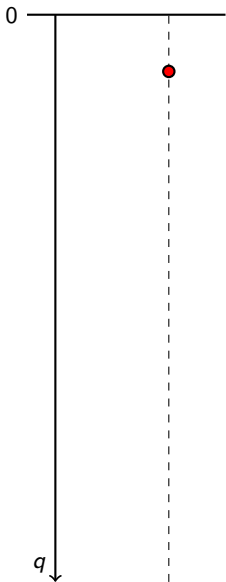
- $\mu_0 = [0, 0]^T$, $\Sigma_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$

STATE TRANSITION

- $q_t = q_{t-1} + v_{t-1}\Delta t + \epsilon_{1t}$
- $v_t = v_{t-1} + g\Delta t - bv_{t-1} + \epsilon_{2t}$
- Matrix form:

$$\underbrace{\begin{bmatrix} q_t \\ v_t \end{bmatrix}}_{x_t} = \underbrace{\begin{bmatrix} 1 & \Delta t \\ 0 & 1 - b \end{bmatrix}}_{A_t} \underbrace{\begin{bmatrix} q_{t-1} \\ v_{t-1} \end{bmatrix}}_{x_{t-1}} + \underbrace{\begin{bmatrix} 0 \\ \Delta t \end{bmatrix}}_{B_t} \underbrace{g}_{u_t} + \underbrace{\begin{bmatrix} \epsilon_{1t} \\ \epsilon_{2t} \end{bmatrix}}_{\epsilon_t}$$

- $R_t = \begin{bmatrix} 0.01^2 & 0 \\ 0 & 0.005^2 \end{bmatrix}$



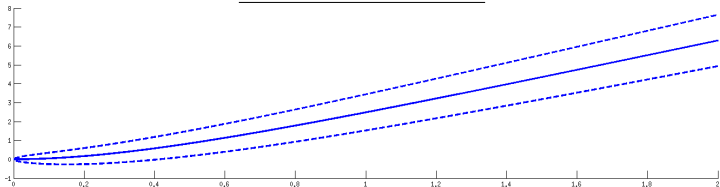
Kalman Filter - Example - Prediction

ONLY PREDICTIONS EFFECTS

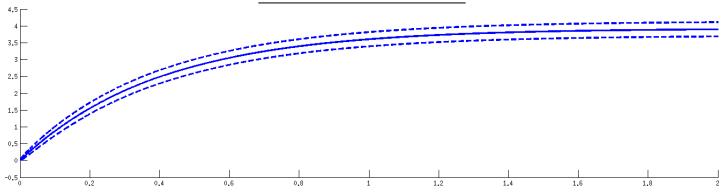
$$\bullet \bar{\mu}_t = \mathbf{A}\mu_{t-1} + \mathbf{B}u_1$$

$$\bullet \bar{\Sigma}_t = \mathbf{A}\Sigma_{t-1}\mathbf{A}^T + \mathbf{R}$$

ALTITUDE PREDICTION



SPEED PREDICTION

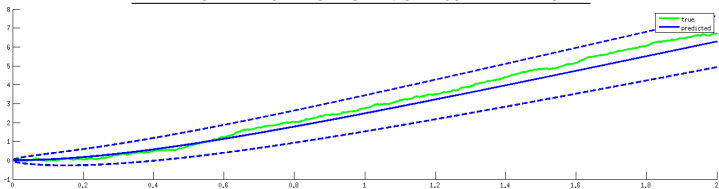


Dashed lines are $\bar{\mu}_t \pm 3 * \sqrt{\text{diag}(\bar{\Sigma}_t)}$

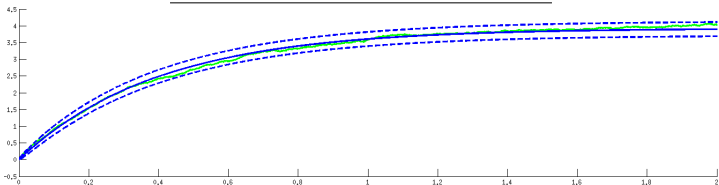
Kalman Filter - Example - Prediction vs True State

WHAT'S HAPPEN IN THE REAL WORLD - STATE TRANSITION

ALTITUDE PREDICTION VS TRUE ALTITUDE



SPEED PREDICTION VS TRUE SPEED

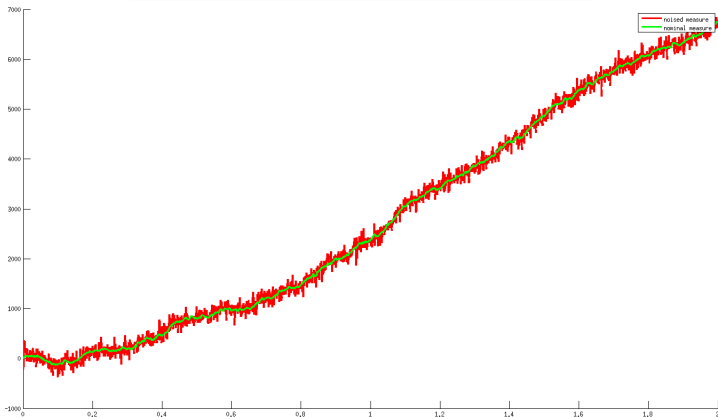


Real Values is contained in the 3σ confidence interval

Kalman Filter - Example - Sensor Measurement vs True Measure

WHAT'S HAPPEN IN THE REAL WORLD - MEASUREMENT

SENSOR MEASUREMENT VS TRUE MEASURE



Kalman Filter - Example - Update Step

MEASUREMENT MODEL

- $z_t = \mathbf{C}_t \mathbf{x}_t + \delta_t$
- $z_t = 1000 \cdot q_t + \delta_t$
- $\mathbf{C}_t = \begin{bmatrix} 1000 & 0 \end{bmatrix}$
- $\mathbf{Q}_t = \begin{bmatrix} 100^2 \end{bmatrix}$

THE UPDATE STEP

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

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

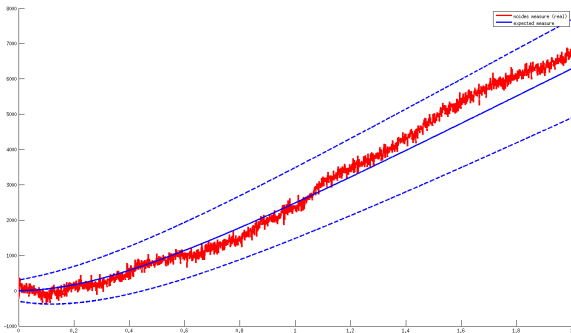
$$\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

AN INTERMEDIATE STEP

- Let's define
 - $\mathbf{y}_t = \mathbf{C}_t \bar{\mu}_t$
 - $\mathbf{S}_t = \mathbf{C}_t \bar{\Sigma}_t \mathbf{C}_t^T + \mathbf{Q}_t$
- They represent a *measurement prediction*
- i.e., the (Gaussian) probability of the expected measurement $\sim \mathcal{N}(\mathbf{y}_t, \mathbf{S}_t)$

Kalman Filter - Example - Measurement Step

MEASUREMENT PREDICTION



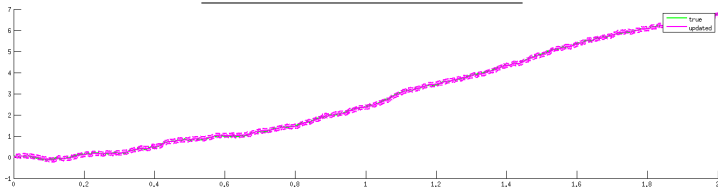
Dashed lines are $y_t \pm 3 * \sqrt{\text{diag}(\mathbf{S})_t}$

Up to now, no Update Step!!

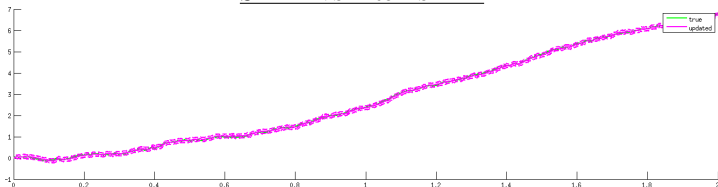
Kalman Filter - Example - Run the update

PREDICTION + UPDATE

ALTITUDE VS TRUE ALTITUDE



SPEED VS TRUE SPEED

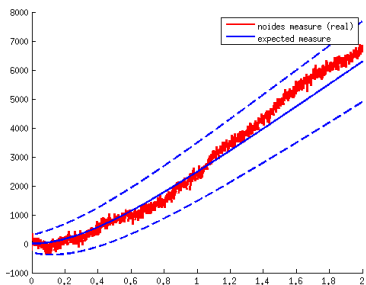


The filter state track the real value!

Kalman Filter - Example - Comparisons

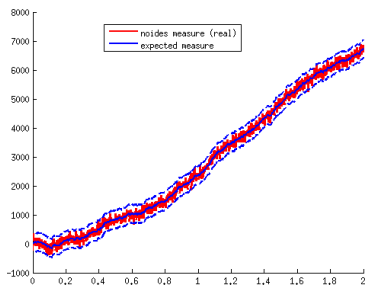
MEASUREMENT PREDICTION

NO UPDATE



MEASUREMENT PREDICTION

WITH UPDATE

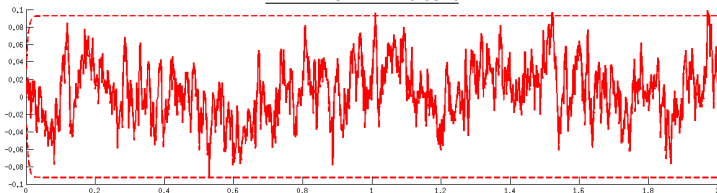


Kalman Filter - Example - Errors

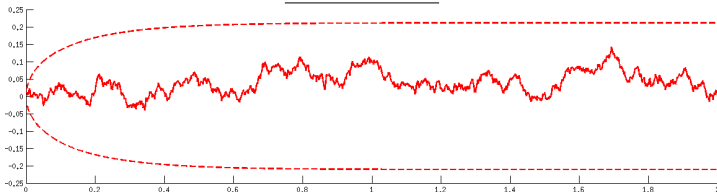
ERROR ON STATE

- $\mathbf{x}_t^* - \mu_t$: difference between true value and estimated

ALTITUDE ERROR

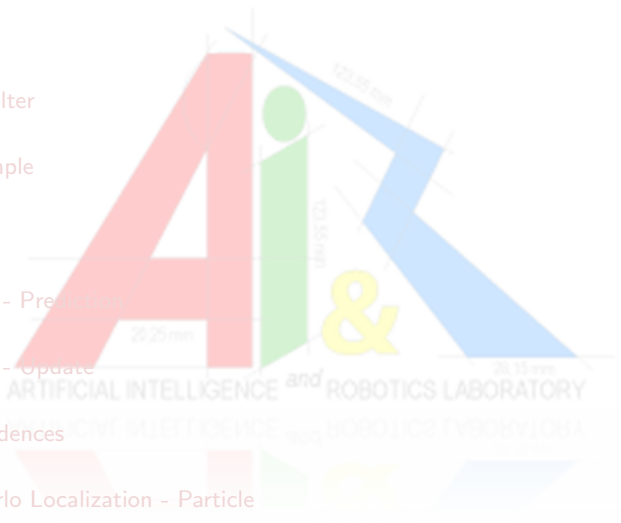


SPEED ERROR



Outline

- 1 Kalman Filter
- 2 K.F. Example
- 3 E.K.F.**
- 4 EKF Loc. - Prediction
- 5 EKF Loc. - Update
- 6 Correspondences
- 7 Monte Carlo Localization - Particle



Extended Kalman Filter

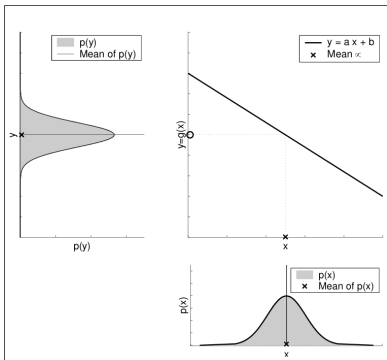
KALMAN FILTER MAIN ISSUE

- Works with linear systems
- Most real systems are modelled by nonlinear functions
- Can we “extend” it to treat non linear system?
- Yes, but under some hypothesis and with some drawbacks
- How to extend? → *local linearization*

Gaussian Variable Transformation - 1

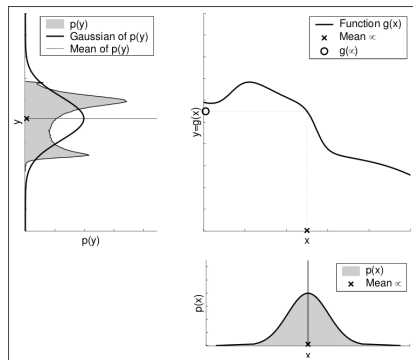
GIVEN $x \sim \mathcal{N}(\mu, \sigma^2)$

$$y = ax + b$$



$$y \sim \mathcal{N}(a\mu + b, a^2\sigma^2)$$

$$y = g(x)$$

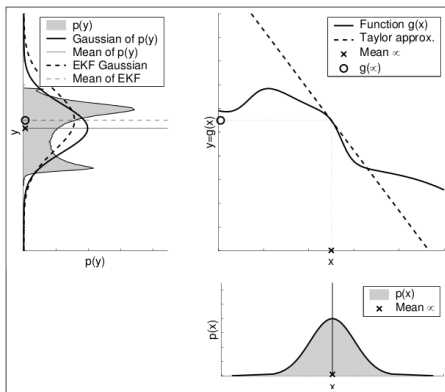


$$y \approx \mathcal{N}(\dots)$$

The maintenance of a good Gaussian approximation depends on the shape of $g(x)$

Gaussian Variable Transformation - 2

Linearization via Taylor Expansion of $g(x)$

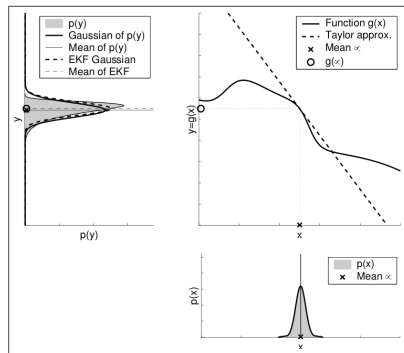
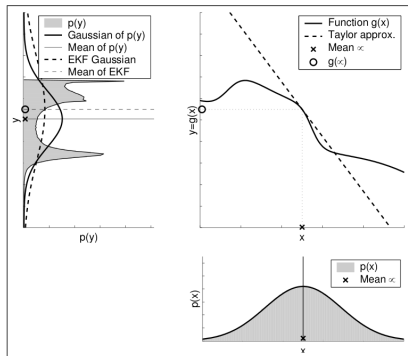


$$y \sim \mathcal{N}(\dots)$$

The maintenance of a good Gaussian approximation depends on the linearization point

Gaussian Variable Transformation - 3

Linearization via Taylor Expansion of $g(x)$



The maintenance of a good Gaussian approximation depends on the variance of x

KF to EKF

KF

STATE TRANSITION

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \epsilon_t$$

MEASUREMENT PROBABILITY

$$\mathbf{z}_t = \mathbf{C}_t \mathbf{x}_t + \delta_t$$

PREDICTION STEP

$$\begin{aligned}\bar{\boldsymbol{\mu}}_t &= \mathbf{A}_t \boldsymbol{\mu}_{t-1} + \mathbf{B}_t \mathbf{u}_t \\ \bar{\boldsymbol{\Sigma}}_t &= \mathbf{A}_t \boldsymbol{\Sigma}_{t-1} \mathbf{A}_t^T + \mathbf{R}_t\end{aligned}$$

UPDATE STEP

$$\begin{aligned}\mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T (\mathbf{C}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^T + \mathbf{Q}_t)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t - \mathbf{K}_t (\mathbf{z}_t + \mathbf{C}_t \bar{\boldsymbol{\mu}}_t) \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{C}_t) \bar{\boldsymbol{\Sigma}}_t\end{aligned}$$

EKF

STATE TRANSITION

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t, \epsilon_t)$$

MEASUREMENT PROBABILITY

$$\mathbf{z}_t = h(\mathbf{x}_t, \delta_t)$$

PREDICTION STEP

$$\begin{aligned}\bar{\boldsymbol{\mu}}_t &= g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0) \\ \bar{\boldsymbol{\Sigma}}_t &= \mathbf{G}_t \boldsymbol{\Sigma}_{t-1} \mathbf{G}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T\end{aligned}$$

UPDATE STEP

$$\begin{aligned}\mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t (\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t, 0)) \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\boldsymbol{\Sigma}}_t\end{aligned}$$

EKF - Jacobians

EKF

STATE TRANSITION

$$\mathbf{x}_t = \mathbf{g}(\mathbf{x}_{t-1}, \mathbf{u}_t, \epsilon_t)$$

MEASUREMENT PROBABILITY

$$\mathbf{z}_t = h(\mathbf{x}_t, \delta_t)$$

PREDICTION STEP

$$\begin{aligned} \bar{\boldsymbol{\mu}}_t &= \mathbf{g}(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0) \\ \bar{\boldsymbol{\Sigma}}_t &= \mathbf{G}_t \boldsymbol{\Sigma}_{t-1} \mathbf{G}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T \end{aligned}$$

UPDATE STEP

$$\begin{aligned} \mathbf{K}_t &= \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T (\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^T + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T)^{-1} \\ \boldsymbol{\mu}_t &= \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t (\mathbf{z}_t - h(\bar{\boldsymbol{\mu}}_t, 0)) \\ \boldsymbol{\Sigma}_t &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \bar{\boldsymbol{\Sigma}}_t \end{aligned}$$

JACOBIANS

- $$\mathbf{G}_t = \left. \frac{\partial \mathbf{g}(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, \epsilon=0}$$
 derivate of the state transition function
w.r.t. state variables
- $$\mathbf{N}_t = \left. \frac{\partial \mathbf{g}(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \epsilon} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, \epsilon=0}$$
 derivate of the state transition function
w.r.t. noise variables
- $$\mathbf{H}_t = \left. \frac{\partial h(\mathbf{x}, \delta)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \delta=0}$$
 derivate of the measurement function
w.r.t. state variables
- $$\mathbf{M}_t = \left. \frac{\partial h(\mathbf{x}, \delta)}{\partial \delta} \right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \delta=0}$$
 derivate of the measurement function
w.r.t. noise variables

Outline

- 1 Kalman Filter
 - 2 K.F. Example
 - 3 E.K.F.
 - 4 **EKF Loc. - Prediction**
 - 5 EKF Loc. - Update
 - 6 Correspondences
 - 7 Monte Carlo Localization - Particle
- 

Localization with E.K.F.

LOCALIZATION

- EKF can treat the *pose tracking* problem
- We can consider *uncertainty* (or *belief*) locally gaussian

STATE TRANSITION - PREDICTION

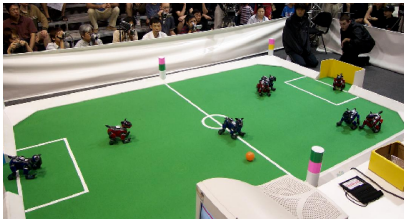
- The next robot position
using motion information

MEASUREMENT - UPDATE

- Sense of a *map landmark*
e.g., sense distance and angle

MAP AND CORRESPONDENCES

- Position of landmarks in world coordinates
- Landmarks are uniquely identifiable
e.g., different colors



Prediction step - Robot motion - 1

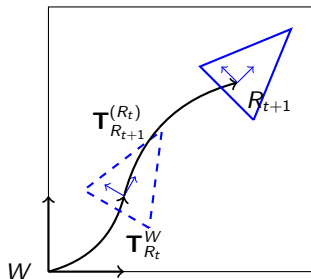
STATE

- The robot position and orientation (2D)
 - $[x, y]$: the robot position
 - θ : orientation

in world reference frame: $\mathbf{T}_{R_t}^W$

STATE PREDICTION

- Input \mathbf{u} : $[\Delta x, \Delta y, \Delta \theta] = [v_x \Delta t, v_y \Delta t, \omega \Delta t]$
- Relative motion: $\mathbf{T}_{R_{t+1}}^{R_t}(\mathbf{u})$
- Prediction: $\mathbf{T}_{R_{t+1}}^W = \mathbf{T}_{R_t}^W \cdot \mathbf{T}_{R_{t+1}}^{R_t}$



$$= \begin{bmatrix} \cos(\theta_t) & -\sin(\theta_t) & x_t \\ \sin(\theta_t) & \cos(\theta_t) & y_t \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & \Delta x \\ \sin(\Delta\theta) & \cos(\Delta\theta) & \Delta y \\ 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} \cos(\theta_t + \Delta\theta) & -\sin(\theta_t + \Delta\theta) & \cos(\theta_t)\Delta x - \sin(\theta_t)\Delta y + x_t \\ \sin(\theta_t + \Delta\theta) & \cos(\theta_t + \Delta\theta) & \sin(\theta_t)\Delta x + \cos(\theta_t)\Delta y + y_t \\ 0 & 0 & 1 \end{bmatrix}$$

Prediction step - Robot motion - 2

STATE PREDICTION

- $\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t, 0)$ (no noise)
- $$\begin{cases} x_{t+1} = \cos(\theta_t)\Delta x - \sin(\theta_t)\Delta y + x_t \\ y_{t+1} = \sin(\theta_t)\Delta x + \cos(\theta_t)\Delta y + y_t \\ \theta_{t+1} = \theta_t + \Delta\theta \end{cases}$$

NOISE INTRODUCTION

- Suppose that noise affects inputs

- $$\begin{cases} \tilde{\Delta}x = \Delta x + \epsilon_x \\ \tilde{\Delta}y = \Delta y + \epsilon_y \\ \tilde{\Delta}\theta = \Delta\theta + \epsilon_\theta \end{cases}$$
- $\epsilon = [\epsilon_x, \epsilon_y, \epsilon_\theta] \sim \mathcal{N}(0, \Sigma_\epsilon)$
- $$\begin{cases} x_{t+1} = \cos(\theta_t)\tilde{\Delta}x - \sin(\theta_t)\tilde{\Delta}y + x_t \\ y_{t+1} = \sin(\theta_t)\tilde{\Delta}x + \cos(\theta_t)\tilde{\Delta}y + y_t \\ \theta_{t+1} = \tilde{\theta}_t + \Delta\theta \end{cases}$$

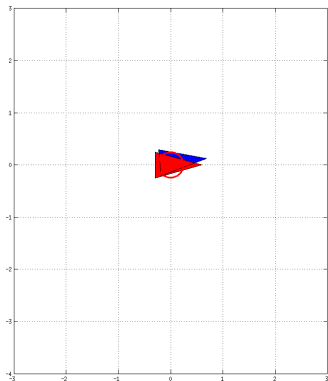
JACOBIANS

- $$\mathbf{G}_t = \left. \frac{\partial g(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, \epsilon=0}$$
- $$= \begin{bmatrix} \frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial x} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial y} & \frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \theta} \\ \frac{\partial g_2(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial x} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial y} & \frac{\partial g_2(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \theta} \\ \frac{\partial g_3(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial x} & \frac{\partial g_3(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial y} & \frac{\partial g_3(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \theta} \end{bmatrix}$$
- $\frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial x} = 1, \quad \frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial y} = 0$
- $\frac{\partial g_1(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \theta} = -\sin(\theta)\Delta x - \cos(\theta)\Delta y$
- $$= \begin{bmatrix} 1 & 0 & -\sin(\theta)\Delta x - \cos(\theta)\Delta y \\ 0 & 1 & \cos(\theta)\Delta x - \sin(\theta)\Delta y \\ 0 & 0 & 1 \end{bmatrix}$$
- $$\mathbf{N}_t = \left. \frac{\partial g(\mathbf{x}, \mathbf{u}, \epsilon)}{\partial \epsilon} \right|_{\mathbf{x}=\boldsymbol{\mu}_{t-1}, \mathbf{u}=\mathbf{u}_t, \epsilon=0}$$
- $$= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Initial state

EKF STATE

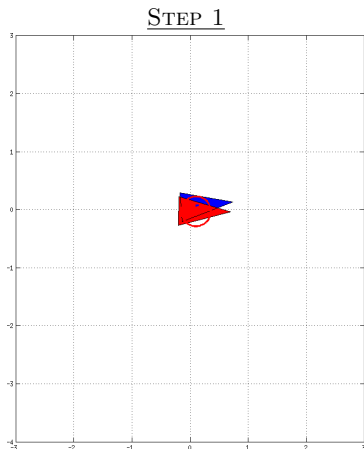
- $\mu_0 = [0, 0, 0]$
- $\Sigma_0 = \begin{bmatrix} 0.1^2 & 0 & 0 \\ 0 & 0.1^2 & 0 \\ 0 & 0 & \text{deg2rad}(10)^2 \end{bmatrix}$
- Robot is in the origin
- But we have some uncertainty on its position and orientation
- e.g., Robot real pose is $[0.094, 0.069, 5.2769^\circ]$
- $\mathbf{C} = k \cdot \Sigma_{0[1:2,1:2]}^{-1}$ confidence ellipse



blue: true position

red: estimated position (μ)

Only Prediction - 1



a (noisy) input arrives, the prediction step is performed

$$\bar{\boldsymbol{\mu}}_t = g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t, 0)$$

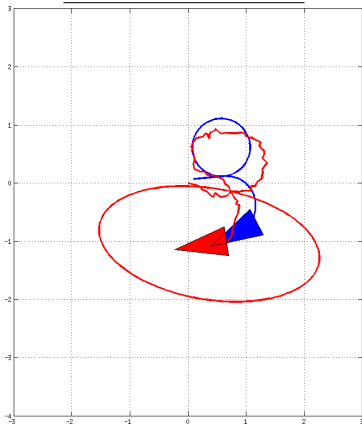
$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{H}_t \boldsymbol{\Sigma}_{t-1} \mathbf{H}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T$$

blue: true position

red: estimated position ($\boldsymbol{\mu}$)

Only Prediction - 2

AFTER SOME STEPS...



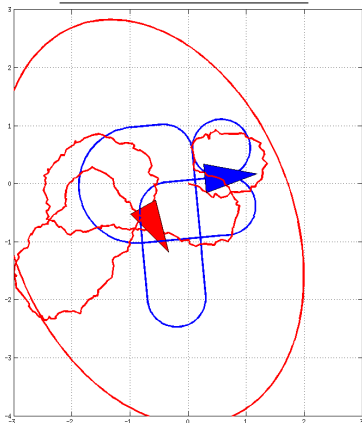
$$\bar{\mu}_t = g(\mu_{t-1}, \mathbf{u}_t, 0)$$

$$\bar{\Sigma}_t = \mathbf{H}_t \Sigma_{t-1} \mathbf{H}_t^T + \mathbf{N}_t \mathbf{R}_t \mathbf{N}_t^T$$

blue: true position and trajectory

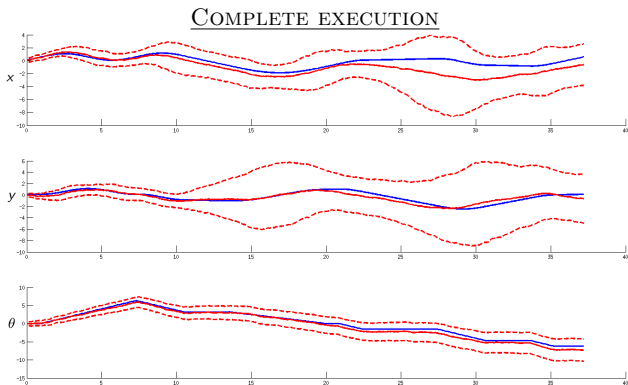
red: estimated position and trajectory (μ_t)

Only Prediction - 3

COMPLETE EXECUTION

The complete path with the prediction step
Notice that the true position is inside the ellipse

Only Prediction - Graphs



The complete path with the prediction step
 Notice that the true position is inside the ellipse

Prediction - Code

A SNAPSHOT OF CODE

```
G = [ 1, 0, - dy*cos(th) - dx*sin(th);
      0, 1,  dx*cos(th) - dy*sin(th);
      0, 0,                               1];

N = [ cos(th), -sin(th), 0;
      sin(th),  cos(th), 0;
      0,        0,       1];

x1 = cos(th)*(dx) - sin(th)*(dy) + x;
y1 = sin(th)*(dx) + cos(th)*(dy) + y;
th1 = th + dth;

ekf.mu = [x1,y1,th1]';
ekf.sigma = G * ekf.sigma * G' + N * R * N';
```

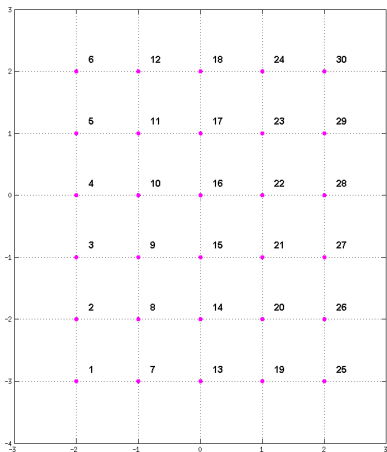
Outline

-
- ① Kalman Filter
 - ② K.F. Example
 - ③ E.K.F.
 - ④ EKF Loc. - Prediction
 - ⑤ EKF Loc. - Update
 - ⑥ Correspondences
 - ⑦ Monte Carlo Localization - Particle

Measurement & Update Step - Map

THE MAP

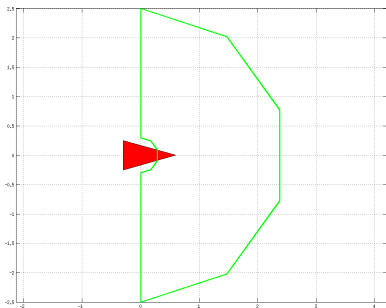
- $\mathbf{m} : \{\mathbf{p}_1^{(W)}, \mathbf{p}_1^{(W)}, \dots, \mathbf{p}_m^{(W)}\}$
- i.e., a set of points in world coordinates
- Known with *absolute* precision



Measurement & Update Step - The sensor

THE SENSOR

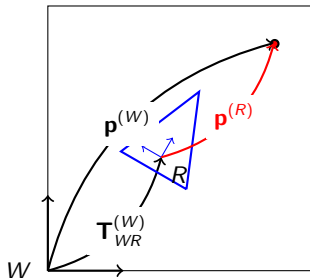
- Measure points in *polar coordinates*
i.e., ρ , θ values
- w.r.t. robot reference frame
- It recognize the ID of the landmark
 - i.e., Landmarks uniquely identifiable
 - Correspondences are known
 - No data association issues
- Physical limits:
 - Min and max distance
 - Min and max angle
 - Additive zero mean noise on measures
both for distance and angle



Measurement & Update Step - The equation

MEASUREMENT

- $\mathbf{x} = [x, y, \theta]$ is the EKF state
i.e., the robot complete pose
- Measure: $h_i(\mathbf{x}, \mathbf{p}_i^{(W)}, \delta)$
 - It express what we expect from the sensor
 - Given a single map point $\mathbf{p}_i^{(W)}$
 - Given the estimate robot pose $\mathbf{x} \rightarrow \mathbf{T}_{WR}^{(W)}$
 - i.e., $\mathbf{p}_i^{(R)}$ in polar coordinates wrt



MEASUREMENT

- $\mathbf{p}_i^{(R)} = (\mathbf{T}_{WR}^{(W)})^{-1} \mathbf{p}_i^{(W)}$
- $\rho_i = \sqrt{\mathbf{p}_{i_x}^{(R)2} + \mathbf{p}_{i_y}^{(R)2}}$
- $\theta_i = \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)})$

MEASUREMENT WITH NOISE

- $h_i(\mathbf{x}, \mathbf{p}_i^{(W)}, \delta_i) = \begin{cases} \tilde{\rho}_i = \sqrt{\mathbf{p}_{i_x}^{(R)2} + \mathbf{p}_{i_y}^{(R)2}} + \delta_{\rho_i} \\ \tilde{\theta}_i = \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)}) + \delta_{\theta_i} \end{cases}$
- $\delta_i = [\delta_{\rho_i}, \delta_{\theta_i}]^T \sim \mathcal{N}(0, \mathbf{Q}_i)$

Measurement & Update Step - Jacobians

MEASUREMENT EQUATION

$$h_i(\mathbf{x}, \mathbf{p}_i^{(W)}, \delta_i) = \begin{cases} \tilde{\rho}_i = \sqrt{\mathbf{p}_{i_x}^{(R)^2} + \mathbf{p}_{i_y}^{(R)^2}} + \delta_{\rho_i} \\ \tilde{\theta}_i = \text{atan2}(\mathbf{p}_{i_y}^{(R)}, \mathbf{p}_{i_x}^{(R)}) + \delta_{\theta_i} \end{cases}$$

$$\mathbf{p}_i^{(R)} = (\mathbf{T}_{WR}^{(W)})^{-1} \mathbf{p}_i^{(W)}$$

EKF JACOBIANS

- $\mathbf{H}_i = \left. \frac{\partial h_i(\mathbf{x}, \mathbf{p}, \delta_i)}{\partial \mathbf{x}} \right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \mathbf{p}=\mathbf{p}_i^{(W)}, \delta_i=0}$
derivate of the measurement function
w.r.t. state variables
- $\mathbf{M}_i = \left. \frac{\partial h_i(\mathbf{x}, \mathbf{p}, \delta_i)}{\partial \delta_i} \right|_{\mathbf{x}=\bar{\boldsymbol{\mu}}_{t-1}, \mathbf{p}=\mathbf{p}_i^{(W)}, \delta_i=0}$
derivate of the measurement function
w.r.t. noise variables

JACOBIANS

$$\begin{aligned} \bullet \mathbf{H}_i &= \begin{bmatrix} \frac{\partial h_{i1}(\mathbf{x}, \mathbf{p}, \delta)}{\partial x} & \frac{\partial h_{i1}(\mathbf{x}, \mathbf{p}, \delta)}{\partial y} & \frac{\partial h_{i1}(\mathbf{x}, \mathbf{p}, \delta)}{\partial \theta} \\ \frac{\partial h_{i2}(\mathbf{x}, \mathbf{p}, \delta)}{\partial x} & \frac{\partial h_{i2}(\mathbf{x}, \mathbf{p}, \delta)}{\partial y} & \frac{\partial h_{i2}(\mathbf{x}, \mathbf{p}, \delta)}{\partial \theta} \end{bmatrix} \\ &= \dots \end{aligned}$$

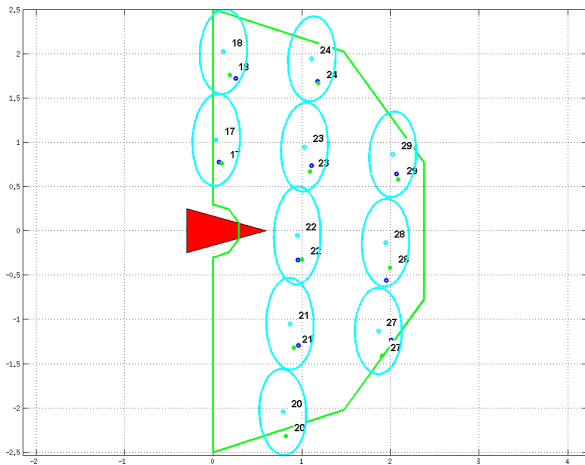
$$\bullet \mathbf{M}_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

SOME NOTES

- Not all $h_i(\cdot)$ are *valid*
e.g., $\rho_i \notin [\rho_{min}, \rho_{max}]$
e.g., $\theta_i \notin [\theta_{min}, \theta_{max}]$
- We select a subset of $h_i(\cdot)$

Measurement & Update Step - Measurement Details

MEASUREMENT IN ROBOT FRAME



- **Cyan:** the *predicted measure*, $h_i(\cdot)$
- **Green:** the real map point in robot coordinates
- **Blue:** the noisy sensor measurement z_i

- Ellipses: given by covariance

$$\mathbf{S}_t = \mathbf{H}_t \bar{\Sigma}_t \mathbf{H}_t^T + \mathbf{M}_t \mathbf{Q}_t \mathbf{M}_t^T$$
- Innovation: $\mathbf{z}_i - h_i(\cdot)$

Measurement & Update Step - Unique Update

THE MEASUREMENTS

- $h_i(\cdot)$, $z_i(\cdot)$, \mathbf{H}_i , $\mathbf{M}_i(\cdot)$, $\mathbf{Q}_i(\cdot)$
feasible measurements and Jacobians
- How to update?

THE COMPLETE MEASUREMENTS

- $h(\mathbf{x}, \mathbf{m}, \delta) = \begin{cases} h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1) \\ h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2) \\ \dots \\ h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m) \end{cases}$
- $\delta = [\delta_1^T \quad \delta_2^T \quad \dots \quad \delta_m^T]^T$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \mathbf{x}} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \mathbf{x}} \\ \dots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \mathbf{x}} \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} \frac{\partial h_1(\mathbf{x}, \mathbf{p}_1^{(W)}, \delta_1)}{\partial \delta_1} \\ \frac{\partial h_2(\mathbf{x}, \mathbf{p}_2^{(W)}, \delta_2)}{\partial \delta_2} \\ \dots \\ \frac{\partial h_m(\mathbf{x}, \mathbf{p}_m^{(W)}, \delta_m)}{\partial \delta_m} \end{bmatrix}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Q}_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 \\ \dots & \dots & \mathbf{Q}_{m-1} & \dots \\ 0 & \dots & 0 & \mathbf{Q}_m \end{bmatrix}$$

THE UPDATE

$$\mathbf{h} = [h_1^T \quad h_2^T \quad \dots \quad h_m^T]^T$$

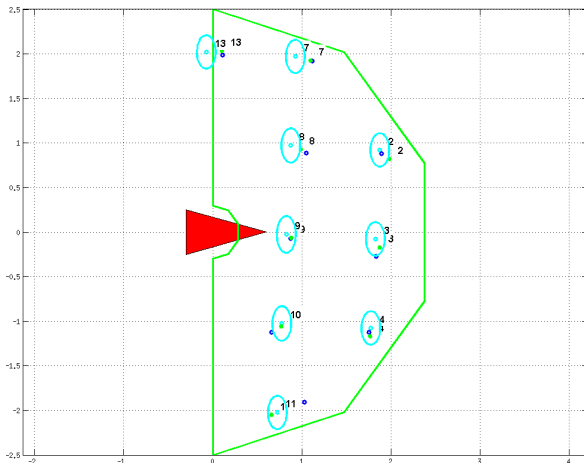
$$\mathbf{H} = [\mathbf{H}_1^T \quad \mathbf{H}_2^T \quad \dots \quad \mathbf{H}_m^T]^T$$

$$\mathbf{z} = [z_1^T \quad z_2^T \quad \dots \quad z_m^T]^T$$

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & 0 & \dots & 0 \\ 0 & \mathbf{M}_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & \mathbf{M}_{m-1} & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & \mathbf{M}_m \end{bmatrix}$$

Measurement & Update Step - Some Steps - 2

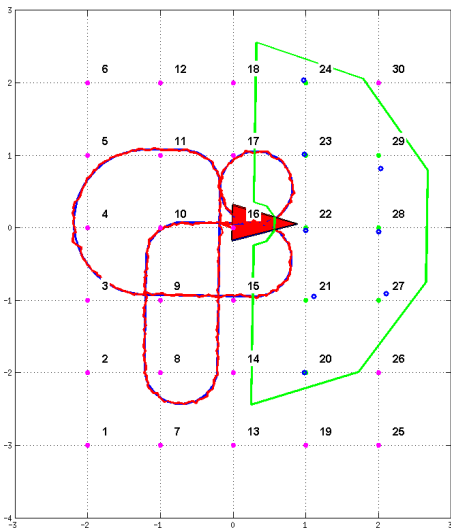
IN THE ROBOT



Covariance on measurement is reduced due to the minor uncertainty in the pose

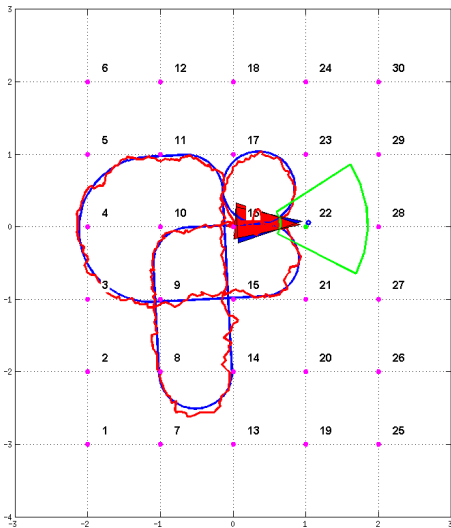
Measurement & Update Step - Complete Execution

THE COMPLETE PATH



Measurement & Update Step - Worse Sensor

THE COMPLETE PATH - WORSE SENSOR



Estimated trajectory is less precise, covariance on robot pose is bigger

Outline

1 Kalman Filter

2 K.F. Example

3 E.K.F.

4 EKF Loc. - Prediction

5 EKF Loc. - Update

6 Correspondences

7 Monte Carlo Localization - Particle



Correspondences

CORRESPONDENCES

- Correspondences are known \rightarrow this is uncommon in real environments
- If correspondences are unknown we have to perform the *data association*

DATA ASSOCIATION

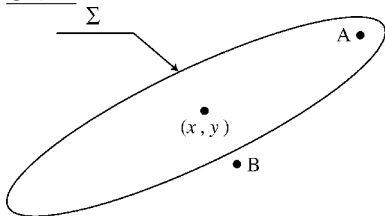
- Given a set of measurements $\{\mathbf{z}_i\}, i = 1 : m$
- Given a set of *measurements prediction* $\{\mathbf{h}_j\}, j = 1 : w$
- We have to select correspondences c_{ij}

“DUMMY” APPROACH

- 1 $k = 1$
- 2 Select w such that \mathbf{z}_w closest to \mathbf{h}_k
- 3 Remove \mathbf{z}_w from $\{\mathbf{z}_i\}$
- 4 Repeat from 2

Mahalanobis Distance

GIVEN



- Given A, B coordinates
- Distance to (x, y)
- Suppose to know covariance Σ
- i.e., $\sim \mathcal{N}(\mu = [x, y], \Sigma)$

EUCLIDEAN DISTANCE

- A is closest to x, y
- B is far

MAHALANOBIS DISTANCE

- $D^2 = (x - \mu)^T \Sigma^{-1} (x - \mu)$
- Squared distance weighted for the inverse of covariance
- $D^2(A) < D^2(B)$,
 A is inside the covariance ellipse
- It is a scaled and rotated distance
- Same probability = same distance
- D^2 is distributed as a $\chi^2(n)$

Data association

DATA ASSOCIATION WITH D^2

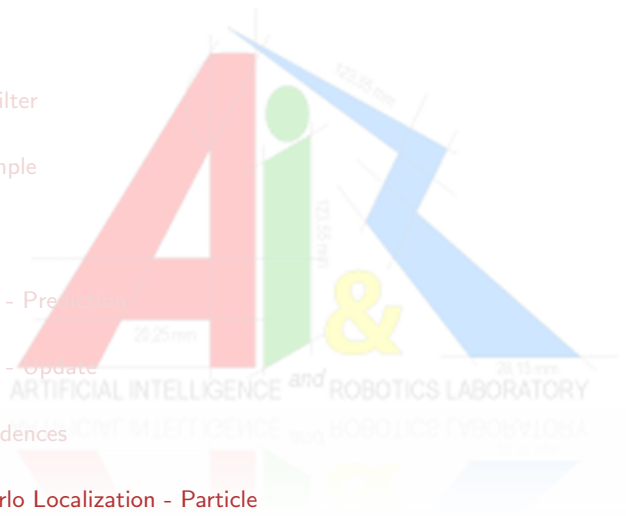
- ➊ $k = 1$
- ➋ Select w such that \mathbf{z}_w closest to \mathbf{h}_k in $D^2(\mathbf{z}_w, \mathbf{h}_k)$
- ➌ Remove \mathbf{z}_w from $\{\mathbf{z}_i\}$
- ➍ Repeat from 2
 - Further, when minimum distance is too high, stop association algorithm

IS IT THE RIGHT APPROACH?

- Is better than the Euclidean distance data association
- Could performs wrong associations
- It consider only *individual compatibility*
- Resulting in a bad localization
- Better approach will consider *joint compatibility* or performs *multi hypothesis*

Outline

- 1 Kalman Filter
- 2 K.F. Example
- 3 E.K.F.
- 4 EKF Loc. - Prediction
- 5 EKF Loc. - Update
- 6 Correspondences
- 7 Monte Carlo Localization - Particle**



Towards non parametric filters

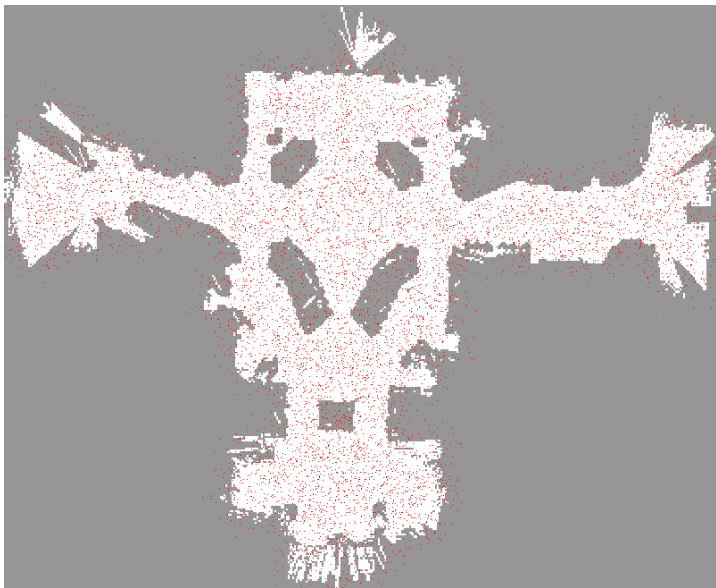
EKF SUMMARY

- Efficient: $< O(n^3)$
- Not optimal, suffers of linearizations
- Can diverge with huge nonlinearities
- Works surprisingly well even when all assumptions are violated
- Not suitable for global localization or kidnapped robot problem

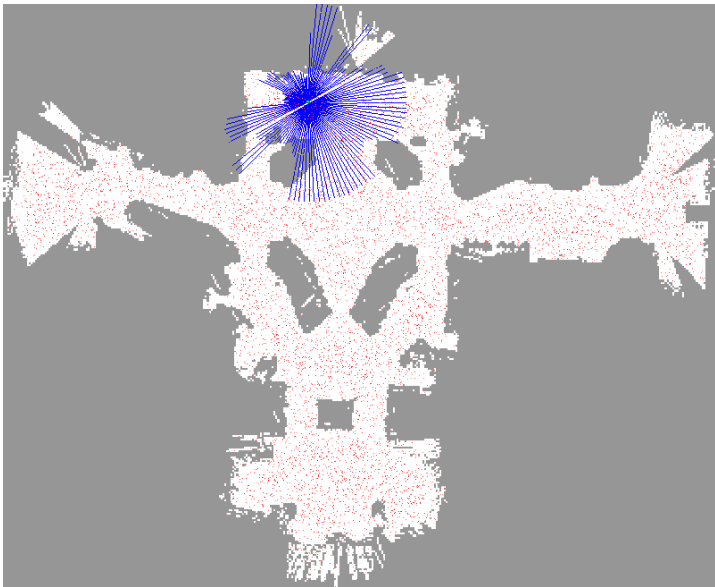
PARTICLE FILTER

- Represents belief by *random samples* from distributions
- Can treat nonlinear and non gaussian problems
- a.k.a. *Sequential Monte Carlo* (SMC) method
- Performs Monte Carlo Localization (MCL)

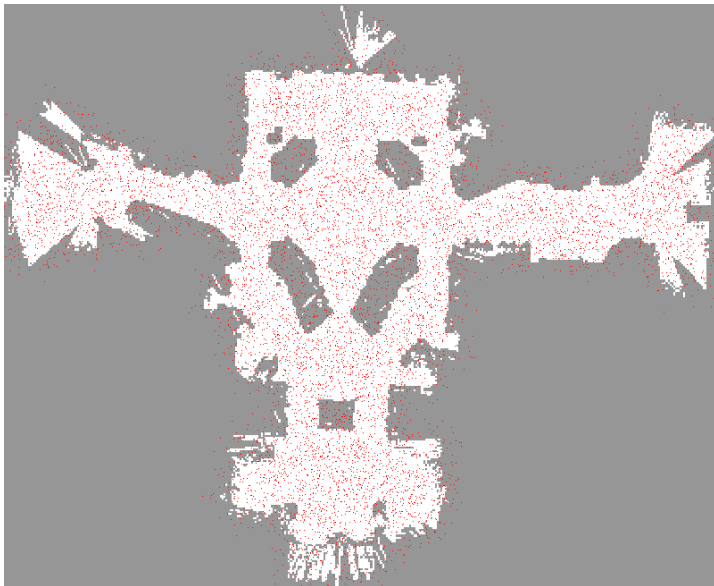
MCL - 1



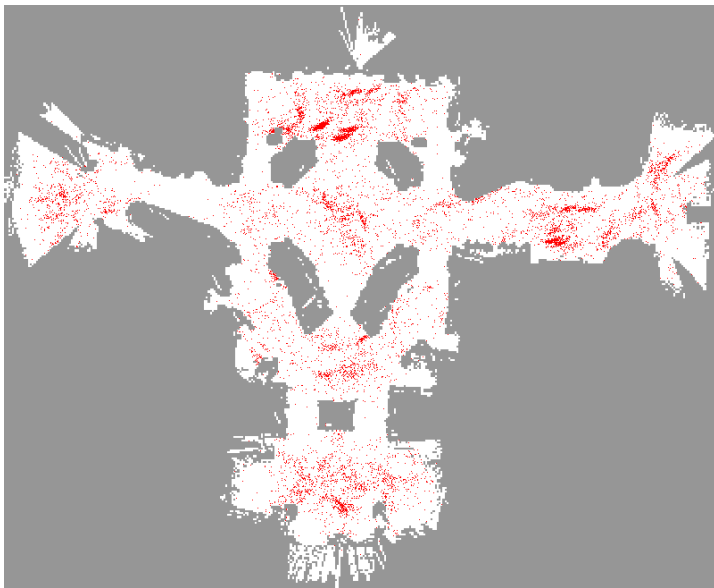
MCL - 2



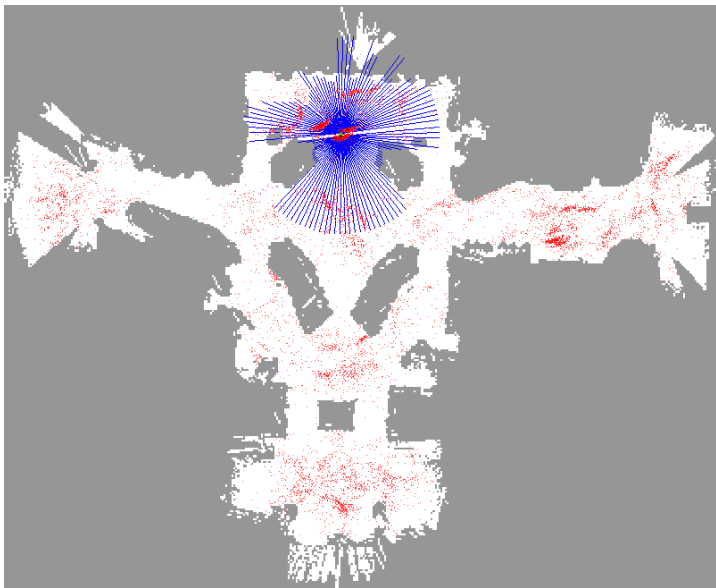
MCL - 3



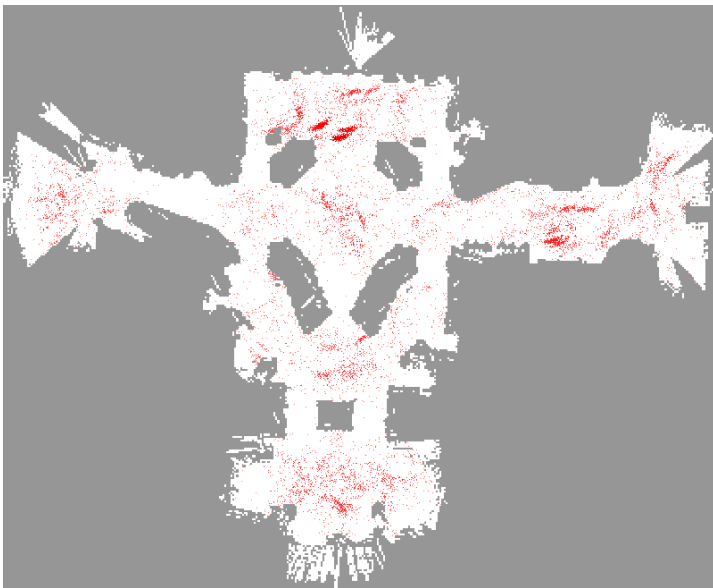
MCL - 4



MCL - 5



MCL - 6



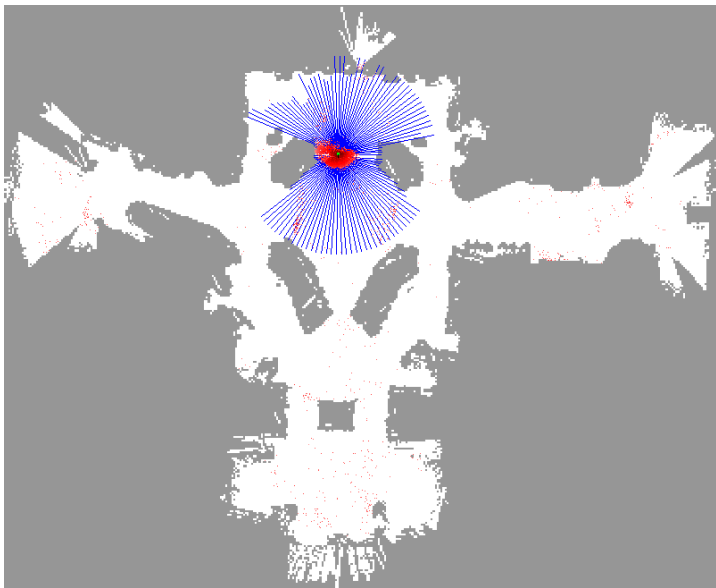
MCL - 7



MCL - 8



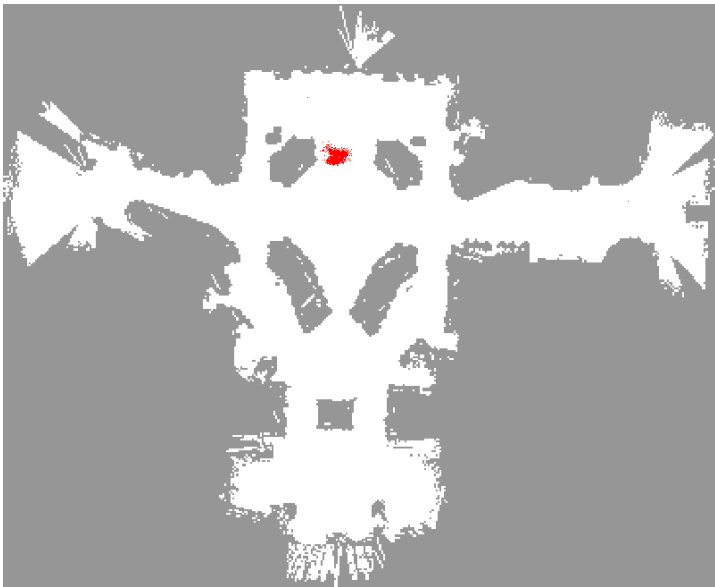
MCL - 9



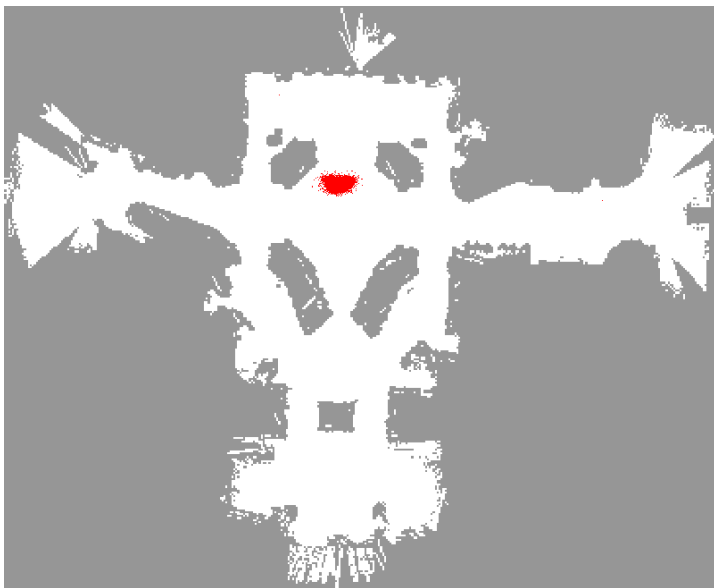
MCL - 10



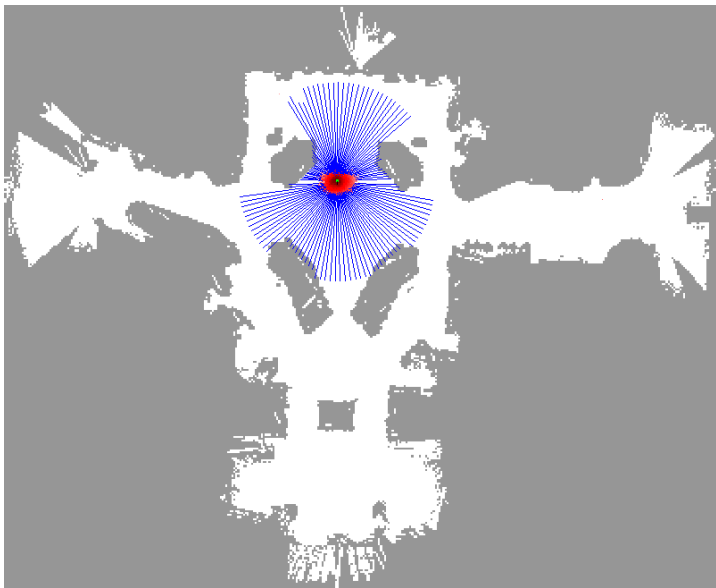
MCL - 11



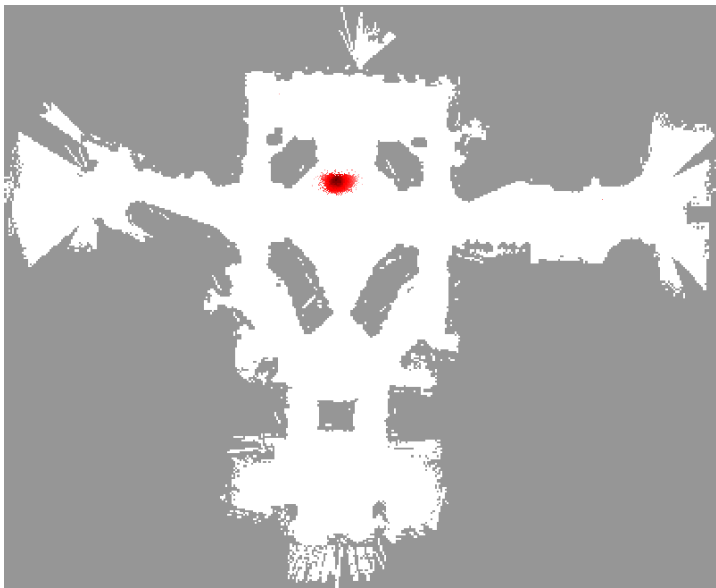
MCL - 12



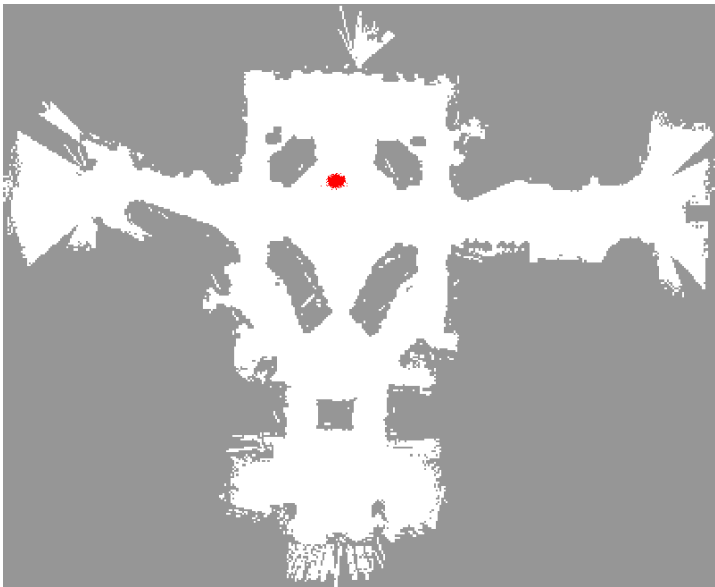
MCL - 13



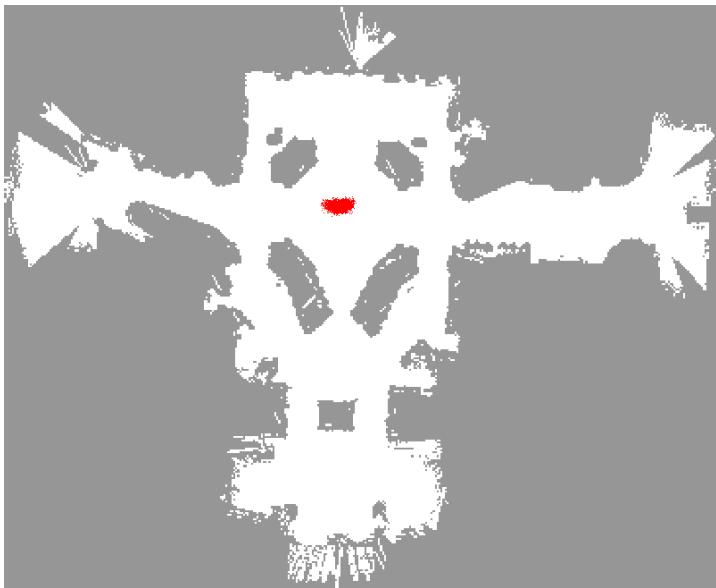
MCL - 14



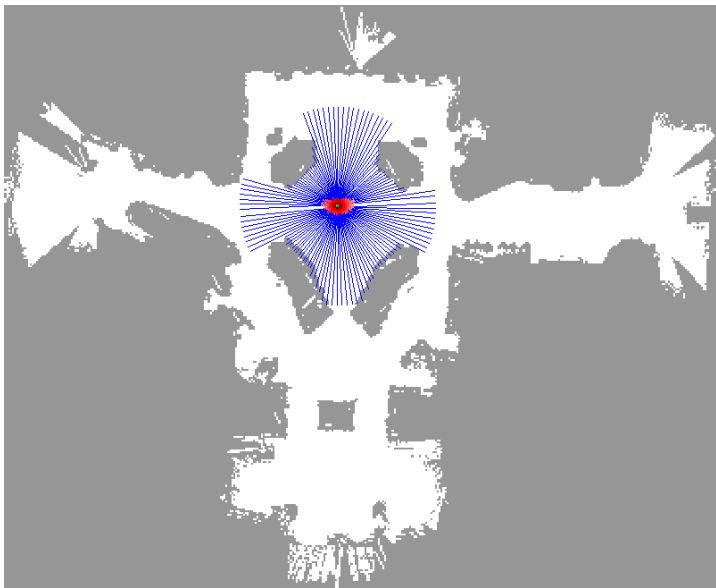
MCL - 15



MCL - 16



MCL - 17



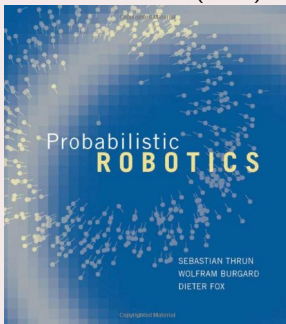
References

Reference

"Probabilistic Robotics"

(Intelligent Robotics and Autonomous Agents series)

The MIT Press (2005)



Chapters 2, 3, 4, 7.