Mobile Robotics

Prof. Dr.-Ing. Gavin Kane

Kalman Filters

Lecture Content

- Probability Revision
- Introduction to the Kalman Filter
- Introduction to SLAM Concepts

Based on Slides from Uni Freiburg, Wolfram Burgard et al.

Probability

Generalised Concept - Actions and Sensors

Given a stream of sensor data together with known actions, estimate the system state.
Using:

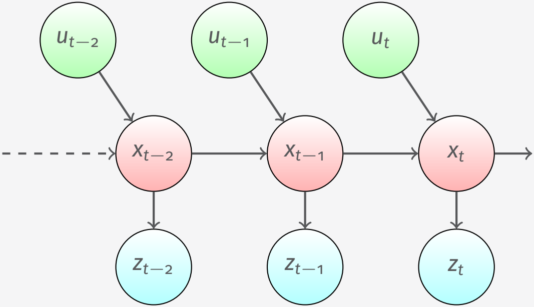
- Previous State Estimate x
- The probability of the previous estimate P(x),
- The Sensor Model $P(x|z)$, and
- The Action Model $P(x|u, x')$

Estimate the new state of the System x.
The posterior of the state is also called the Belief:

$$Bel(x_t) = P(x_t|u_1, z_1, \cdots, u_t, z_t) \tag{1}$$

Probability

Markov Assumption



$$p(z_t | x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t | x_t) \tag{2}$$

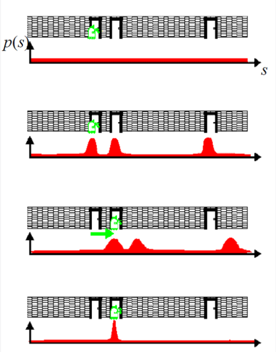
$$p(x_t | x_{1:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t | x_{t-1}, u_t) \tag{3}$$

Probability - Bayes Filter

Calculating the Belief

$$
\begin{aligned}
Bel(x_t) &= P(x_t|u_1, z_1, \cdots, u_t, z_t) \\
\text{(Bayes)} &= \eta P(z_t|x_t, u_1, z_1, \cdots, u_t) P(x_t|u_1, z_1, \cdots, u_t) \\
\text{(Markov)} &= \eta P(z_t|x_t) P(x_t|u_1, z_1, \cdots, u_t) \\
\text{(Total prob.)} &= \eta P(z_t|x_t) \int P(x_t|u_1, z_1, \cdots, u_t, x_{t-1}) P(x_{t-1}|u_1, z_1, \cdots, u_t) dx_{t-1} \\
\text{(Markov)} &= \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) P(x_{t-1}|u_1, z_1, \cdots, u_t) dx_{t-1} \\
\text{(Markov)} &= \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) P(x_{t-1}|u_1, z_1, \cdots, u_t, z_t) dx_{t-1} \\
&= \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1}
\end{aligned}
\tag{4}
$$

Position Estimation



Bayes Filter

$$Bel(x_t) = \eta P(z_t|x_t) \int P(x_t|u_t, x_{t-1}) Bel(x_{t-1}) dx_{t-1} \quad (5)$$

Prediction

$$\overline{bel}(x_t) = \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1} \quad (6)$$

Correction:

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t) \quad (7)$$

Kalman Filter

Position Estimation

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

Probability

Normal Distribution

A normal distribution is one of the most important distributions in Probability theorem. The definition of the distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{8}$$

In the following picture, three different probability density functions are shown:

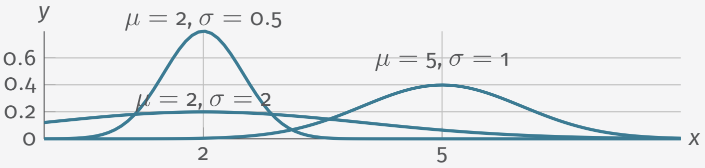


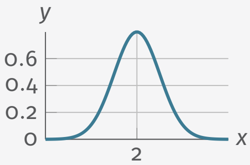
Figure: Examples of Normal Distribution Density Functions

Path Planning

One Dimensional Gaussian

$$p(x) \sim N(\mu, \sigma^2) \tag{9}$$

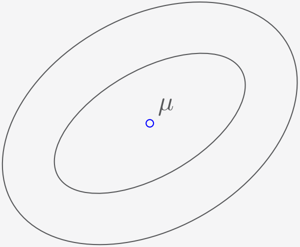
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{10}$$



Two Dimensional Gaussian

$$p(\mathbf{x}) \sim N(\mu, \Sigma) \tag{11}$$

$$p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu)^t\Sigma^{-1}(x-\mu)^2} \tag{12}$$



Position Estimation

Properties of Gaussians

Univariante Case

$$\left.\begin{array}{r} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array}\right\} \Rightarrow Y \sim N(a\mu + b, a^2\mu^2) \tag{13}$$

$$\left.\begin{array}{r} X_1 \sim N(\mu_1, \sigma_1^2) \\ X_2 \sim N(\mu_2, \sigma_2^2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \tag{14}$$

$$\sim N\left(\frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2}\mu_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2}\mu_2, \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}}\right)$$

Position Estimation

Properties of Gaussians

Multivariate Case (State Space)

$$\left.\begin{array}{r} X \sim N(\mu, \Sigma) \\ Y = AX + B \end{array}\right\} \Rightarrow Y \sim N(A\mu + B, A\Sigma A^T) \tag{15}$$

$$\left.\begin{array}{r} X_1 \sim N(\mu_1, \Sigma_1) \\ X_2 \sim N(\mu_2, \Sigma_2) \end{array}\right\} \Rightarrow p(X_1) \cdot p(X_2) \tag{16}$$

$$\sim N\left(\frac{\Sigma_2}{\Sigma_1 + \Sigma_2}\mu_1 + \frac{\Sigma_1}{\Sigma_1 + \Sigma_2}\mu_2, \frac{1}{\Sigma_1^{-1} + \Sigma_2^{-1}}\right)$$

where division is performed through matrix inversion

Position Estimation

Discrete Kalman Filter

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

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \tag{17}$$

with a measurement

$$z_t = C_t x_t + \delta_t \tag{18}$$

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Position Estimation

Components of a Kalman Filter

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

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

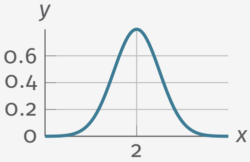
$\mathbf{C_t} \Rightarrow$ Matrix $(k \times n)$ that describes how to map the state x_t to an observation z_t .

$\varepsilon_\mathbf{t}$ & $\delta_\mathbf{t} \Rightarrow$ Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance Q_t and R_t respectively.

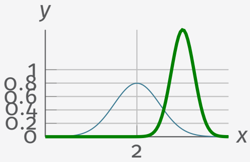
Position Estimation

Kalman Filter Updates in 1D

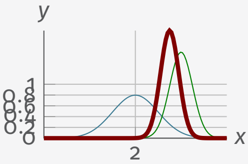
Prediction



Measurement



Correction

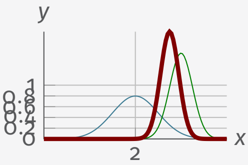


The Correction is calculated from a weighted mean of the prediction and measurement.

Position Estimation

How to get the red one?

Kalman correction step

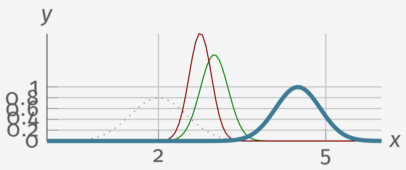


$$bel(x_t) = \begin{cases} \mu_t = \overline{\mu_t} + K_t(z_t - \overline{\mu_t}) \\ \sigma_t^2 = (1 - K_t)\overline{\sigma_t}^2 \end{cases} \text{ with } K_t = \frac{\overline{\sigma_t}^2}{\overline{\sigma_t}^2 + \overline{\sigma}^2_{obs,t}} \tag{19}$$

$$bel(x_t) = \begin{cases} \mu_t = \overline{\mu_t} + K_t(z_t - C_t\overline{\mu_t}) \\ \Sigma_t = (I - K_t C_t)\overline{\Sigma_t} \end{cases} \text{ with } K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1} \tag{20}$$

Position Estimation

How to get the State Prediction Step?



$$bel(x_t) = \begin{cases} \overline{\mu_t} = a_t \mu_t + b_t u_t \\ \overline{\sigma_t}^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases} \tag{21}$$

$$bel(x_t) = \begin{cases} \overline{\mu_t} = A_t \mu_{t-1} + B_t u_t \\ \overline{\Sigma_t} = A_t \Sigma_{t-1} A_t^T + Q^T \end{cases} \tag{22}$$

Position Estimation

Kalman Filter Summary

- Only two parameters describe belief about the state of the system
- Highly efficient
- Optimal for linear Gaussian systems!
- However: Most robotics systems are nonlinear!

Position Estimation

What is SLAM?

- Estimate the pose of a robot and the map of the environment at the same time
- SLAM is hard, because
  - a map is needed for localization and
  - a good pose estimate is needed for mapping
- Localization: inferring location given a map
- Mapping: inferring a map given locations
- SLAM: learning a map and locating the robot simultaneously

Position Estimation

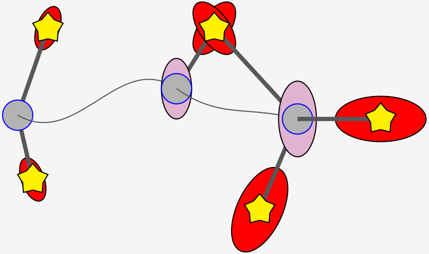
The SLAM Problem

- SLAM is considered a fundamental problem for robots in order to become truly autonomous
- Large variety of different SLAM approaches have been developed
- The majority uses probabilistic concepts
- History of SLAM dates back to the mid-eighties

Position Estimation

Why is SLAM such a hard problem?

1. Both path and map are unknown

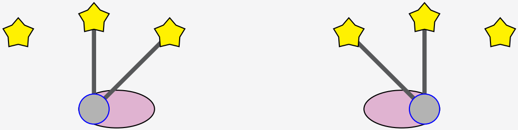


2. Errors in map and pose estimates correlated

Position Estimation

Why is SLAM such a hard problem?

- The mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences (divergence)



Robot pose uncertainty

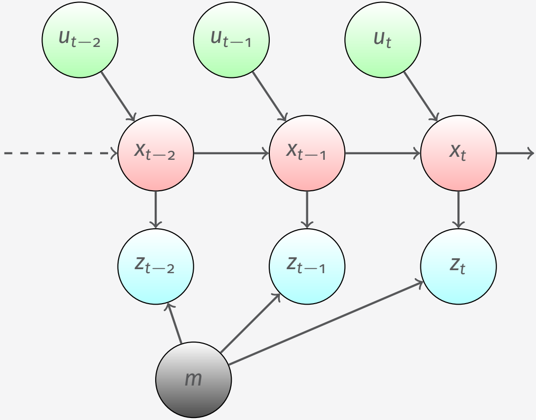
Position Estimation

SLAM: Simultaneous Localisation and Mapping

- Full SLAM:
$$p(x_{0:t}, m | z_{1:t}, u_{1:t}) \tag{23}$$
Estimates entire path and map
- Online SLAM:
$$p(x_t, m | z_{1:t}, u_{1:t}) \tag{24}$$
Estimates most recent pose and map!
- Integrations (marginalization) typically done recursively, one measurement and one action at a time

SLAM

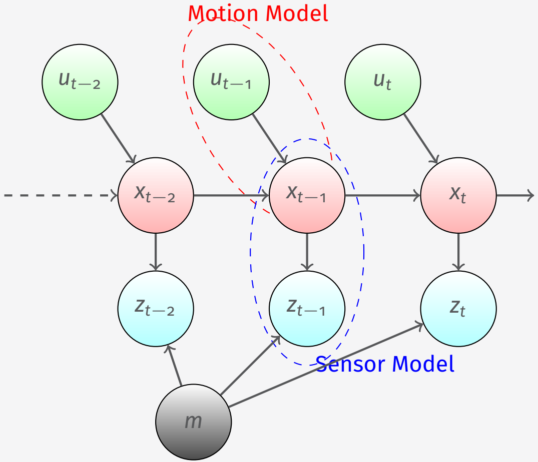
Graphical Model of Full SLAM



$$p(x_{1:t+1}, m | z_{1:t+1}, u_{1:t+1}) \tag{25}$$

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Motion and Observation Model



$$p(x_{1:t+1}, m | z_{1:t+1}, u_{1:t+1}) \tag{26}$$

Position Estimation

Remember the KF Algorithm

1. Algorithm Kalman filter $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$:
2. Prediction
   1. $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
   2. $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3. Correction
   1. $K_t = \Sigma_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
   2. $\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t)$
   3. $\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$
4. Return μ_t, Σ_t

University of Applied Sciences
HOCHSCHULE EMDEN·LEER

Position Estimation

EXF SLAM: State Representation

(EKF = Extended Kalman Filter)

Localization

3×1 pose vector

3×3 cov. matrix

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} \qquad \Sigma_k = \begin{bmatrix} \sigma_x^2 & \sigma_{xy}^2 & \sigma_{x\theta}^2 \\ \sigma_{yx}^2 & \sigma_y^2 & \sigma_{y\theta}^2 \\ \sigma_{\theta x}^2 & \sigma_{\theta y}^2 & \sigma_\theta^2 \end{bmatrix} \tag{27}$$

SLAM Landmarks simply extend the state. Growing state vector and covariance matrix!

$$x_k = \begin{bmatrix} x_R \\ m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \qquad \Sigma_k = \begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \Sigma_{RM_2} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \Sigma_{M_1M_2} & \cdots & \Sigma_{M_1M_n} \\ \Sigma_{M_2R} & \Sigma_{M_2M_1} & \Sigma_{M_2} & \cdots & \Sigma_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \Sigma_{M_nM_2} & \cdots & \Sigma_{M_n} \end{bmatrix} \tag{28}$$

Position Estimation

EKF SLAM: Filter Cycle

1. State prediction (odometry)
2. Measurement prediction
3. Measurement
4. Data association
5. Update
6. Integration of new landmarks

Position Estimation

EKF SLAM: State Prediction

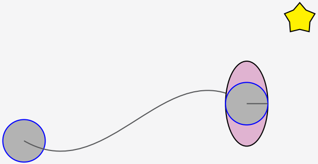
Odometry:

$$\hat{x_R} = f(x_R, u) \tag{29}$$

$$\hat{\Sigma_R} = F_x \Sigma_R F_x^T + F_u U F_u^T \tag{30}$$

Robot-landmark cross-covariance prediction:

$$\hat{\Sigma_{RM_i}} = F_x \Sigma_{RM_i} \tag{31}$$



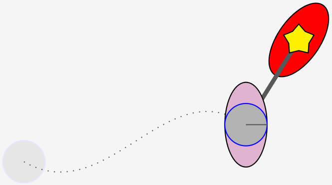
$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \qquad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1 R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1 M_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_n R} & \Sigma_{M_n M_1} & \cdots & \Sigma_{M_n} \end{bmatrix}}_{\Sigma} \tag{32}$$

Position Estimation

EKF SLAM: Measurement Prediction

Global-to-local frame transform h

$$\hat{z}_k = h(\hat{x}_k) \tag{33}$$



$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \qquad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1M_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \cdots & \Sigma_{M_n} \end{bmatrix}}_{\Sigma} \tag{34}$$

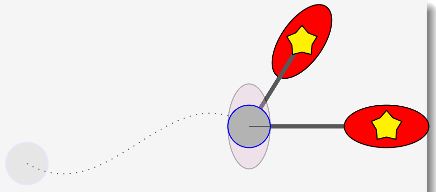
Position Estimation

EKF SLAM: Obtained Measurement

(x, y)-point landmarks

$$z_k = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad (35)$$

$$R_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \quad (36)$$



$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \quad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1M_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \cdots & \Sigma_{M_n} \end{bmatrix}}_{\Sigma} \quad (37)$$

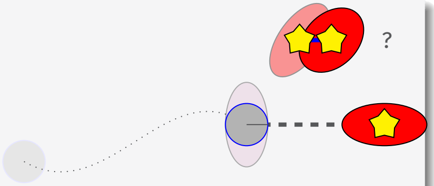
Position Estimation

EKF SLAM: Data Association

Associates predicted measurements $\hat{z}_k^i$ with measurement z_k^j

$$v_k^{ij} = z_k^j - \hat{z}_k^i \tag{38}$$

$$S_k^{ij} = R_k^j + H^i \hat{\Sigma}_k H^{iT} \tag{39}$$



$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \quad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1M_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \cdots & \Sigma_{M_n} \end{bmatrix}}_{\Sigma} \tag{40}$$

Position Estimation

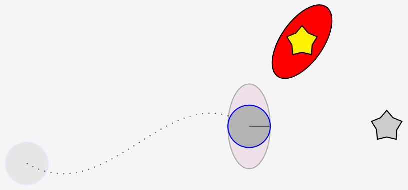
EKF SLAM: Update Step

The usual Kalman filter expressions

$$K_k = \hat{\Sigma}_k H^T S_k^{-1} \tag{41}$$

$$x_k = \hat{x}_k + K_k v_k \tag{42}$$

$$C_k = (I - K_k H)\hat{\Sigma}_k \tag{43}$$



$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \qquad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} \\ \Sigma_{M_1R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1M_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{M_nR} & \Sigma_{M_nM_1} & \cdots & \Sigma_{M_n} \end{bmatrix}}_{\Sigma} \tag{44}$$

Position Estimation

EKF SLAM: New Landmarks

State augmented by

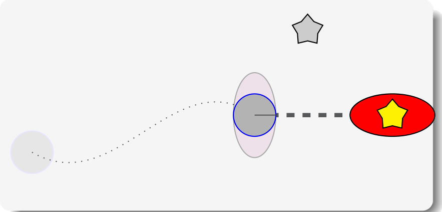
$$m_{n+1} = g(x_R, z_j) \tag{45}$$

$$\Sigma_{M_{n+1}} = G_R \Sigma_R G_R^T + G_z R_j G_z^T \tag{46}$$

Cross-covariances:

$$\Sigma_{M_{n+1} M_i} = G_R \Sigma_{R M_i} \tag{47}$$

$$\Sigma_{M_{n+1} R} = G_R \Sigma_R \tag{48}$$

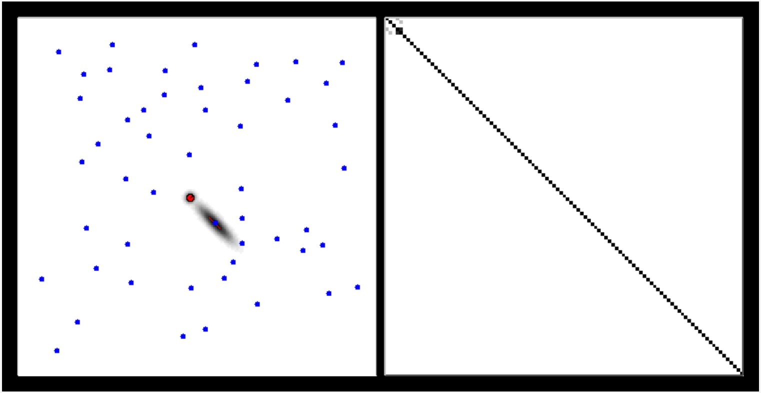


$$\underbrace{\begin{bmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{bmatrix}}_{\mu} \quad \underbrace{\begin{bmatrix} \Sigma_R & \Sigma_{RM_1} & \cdots & \Sigma_{RM_n} & \Sigma_{RM_{n+1}} \\ \Sigma_{M_1 R} & \Sigma_{M_1} & \cdots & \Sigma_{M_1 M_n} & \Sigma_{M_1 M_{n+1}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Sigma_{M_n R} & \Sigma_{M_n M_1} & \cdots & \Sigma_{M_n} & \Sigma_{M_{n+1}} \\ \Sigma_{M_{n+1} R} & \Sigma_{M_{n+1} M_1} & \cdots & \Sigma_{M_{n+1} M_n} & \Sigma_{M_{n+1}} \end{bmatrix}}_{\Sigma} \tag{49}$$

Position Estimation

EKF SLAM

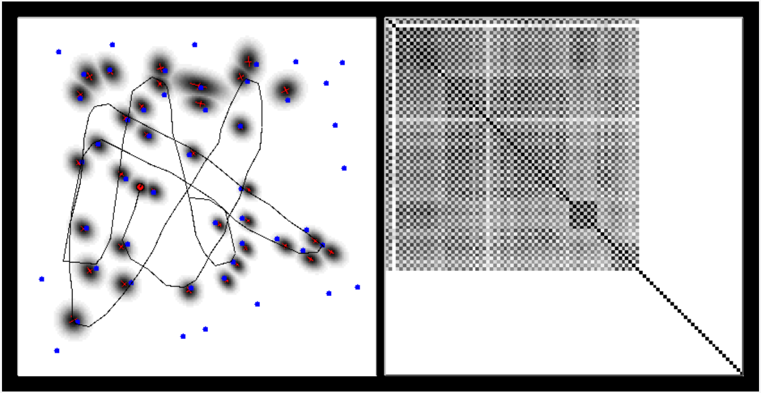
Left Picture: Map of Landmarks with Robot and measurement gaussian.
Right Picture: Σ Matrix. Greycoding shows values (white - 0, Black 1).



Position Estimation

EKF SLAM

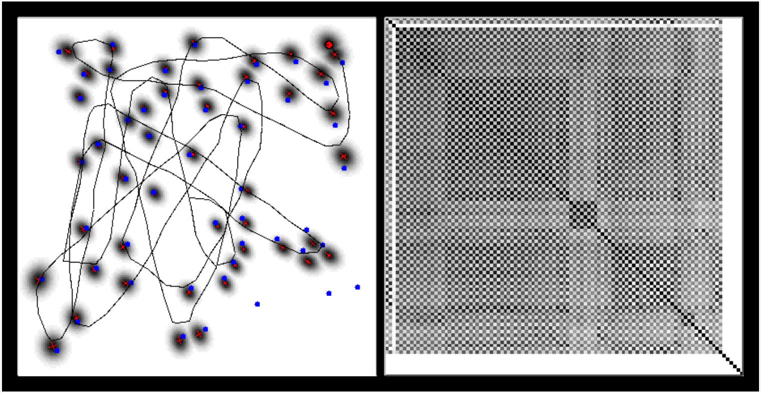
Left Picture: Map of Landmarks with Robot and measurement gaussian.
Right Picture: Σ Matrix. Greycoding shows values (white - 0, Black 1).



Position Estimation

EKF SLAM

Left Picture: Map of Landmarks with Robot and measurement gaussian.
Right Picture: Σ Matrix. Greycoding shows values (white - 0, Black 1).



Position Estimation

EKF-SLAM: Summary

- The first SLAM solution
- Convergence proof for linear Gaussian case
- Can diverge if nonlinearities are large (and the real world is nonlinear ...)
- Can deal only with a single mode
- Successful in medium-scale scenes
- Approximations exists to reduce the computational complexity
- Various alternates include:
  - EKF SLAM
  - FastSLAM
  - Graph-based SLAM
  - Topological SLAM

Position Estimation

Graph-based SLAM