BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

7 – Simultaneous Localization and Mapping

Advanced Methods for Mapping and Self-localization in Robotics (MPC-MAP)

Course supervisor: Ing. Petr Gábrlík, Ph.D.

Adam Ligocki

Brno University of Technology
2024

Robotics and AI

Profile

Robotics and AI

Ing. Adam Ligocki, Ph.D.

Position: Research Staff

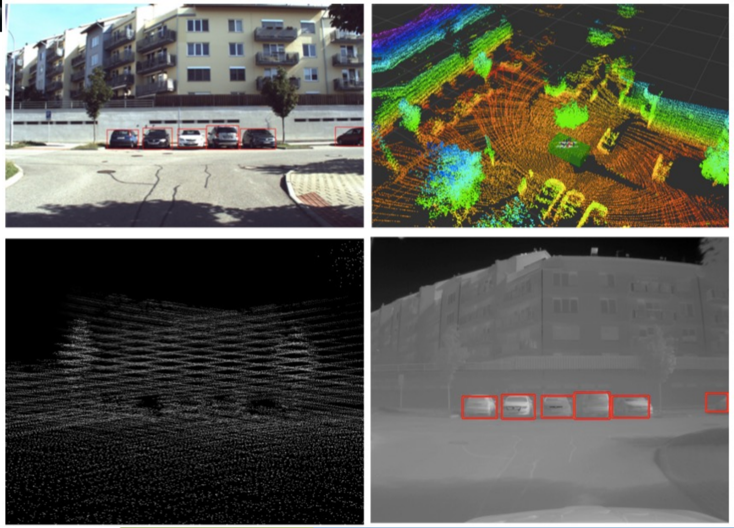
Research: Data Fusion

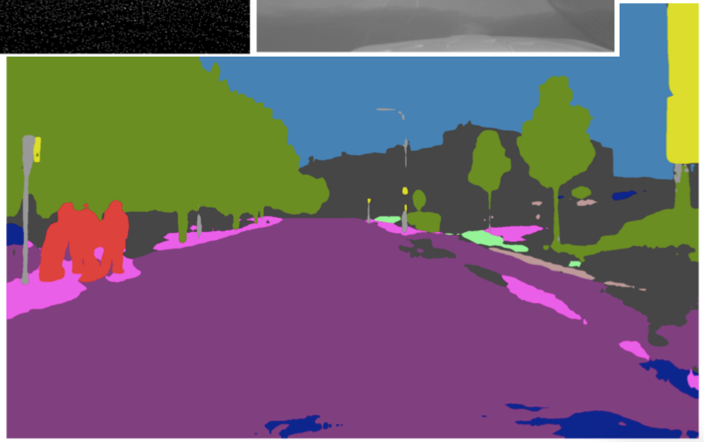
Room: SE1.102

Web: https://www.vut.cz/lide/adam-ligocki-154791

Background:

- Artificial Intelligence
- Neural Networks
- Software Development





Simultaneous Localization and Mapping (SLAM)

Motivation

Localization:

Collision Avoidance

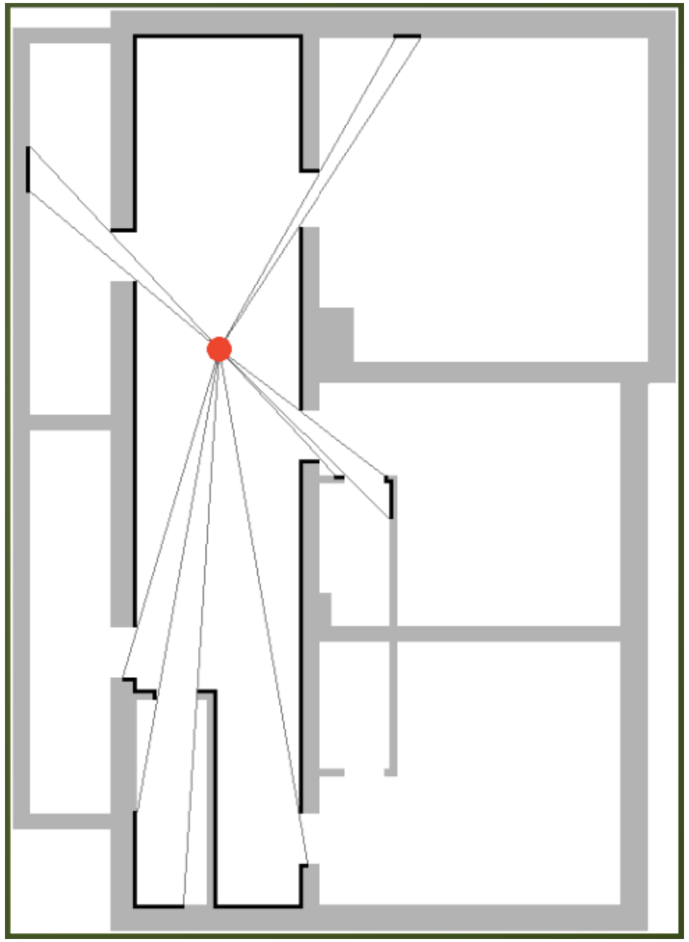
Orientation

Navigation

Mapping:

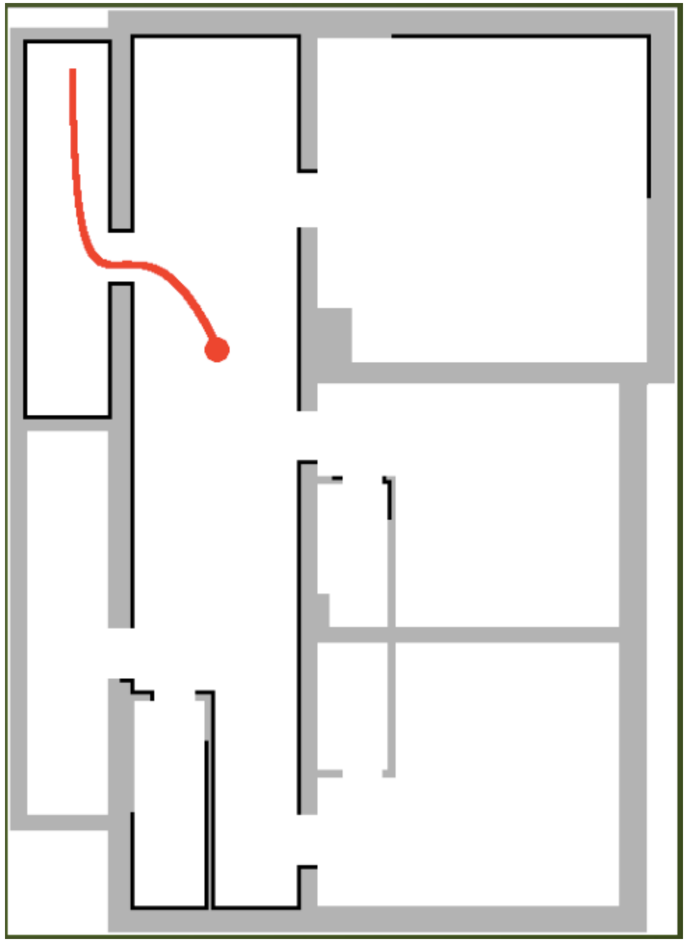
Exploring Unknown Environment

Mission Planning



Self-localization in known environment

Mission Planning



[Jelínek Aleš, SLAM-MRBT-2015]

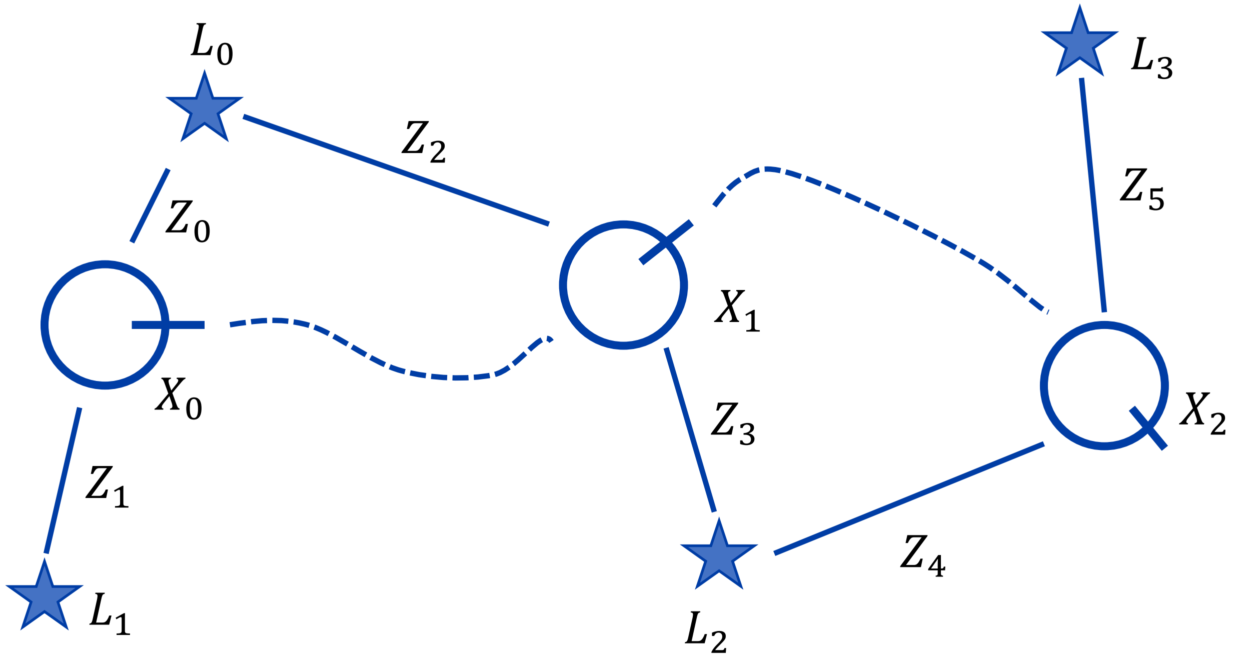
Simultaneous Localization and Mapping - SLAM

Mapping – Building model of the surrounding environment

Localization – Estimating robot‘s pose

SLAM – Build a model of the environment and estimate self position in it at the same time

Map Example:



X_n ... Pose
L_n ... Landmark
Z_n ... Measurement

Simultaneous Localization and Mapping - SLAM

Given:

x_0 ... initial position
u_t ... robot's control
z_t ... robot's observation

Wanted:

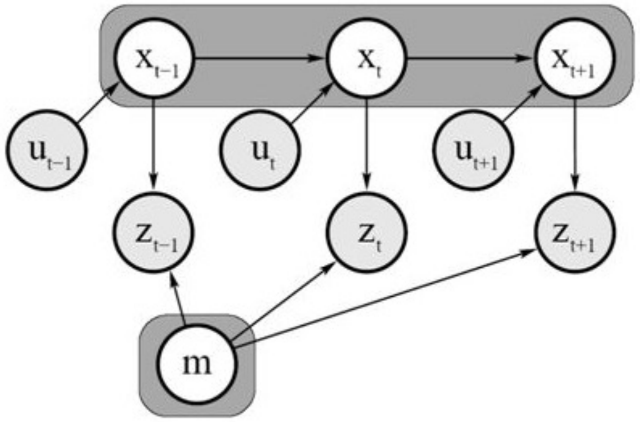
$x_{1:T}$ ... trajectory (vector of poses)
m ... Map

Data are corrupted by errors -> not able to estimate *x* and *m* precisely.

Generic SLAM Usage Examples:

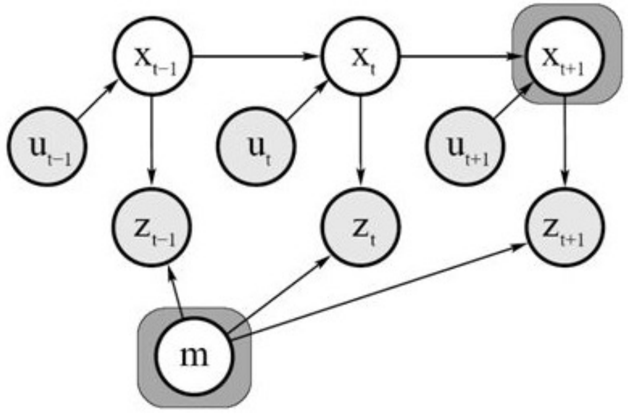
Vacuum cleaners
UGV, UAV
mining, space, sea explorations
AR (mobile phones), VR headsets

Full vs. Online SLAM



Full SLAM

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$



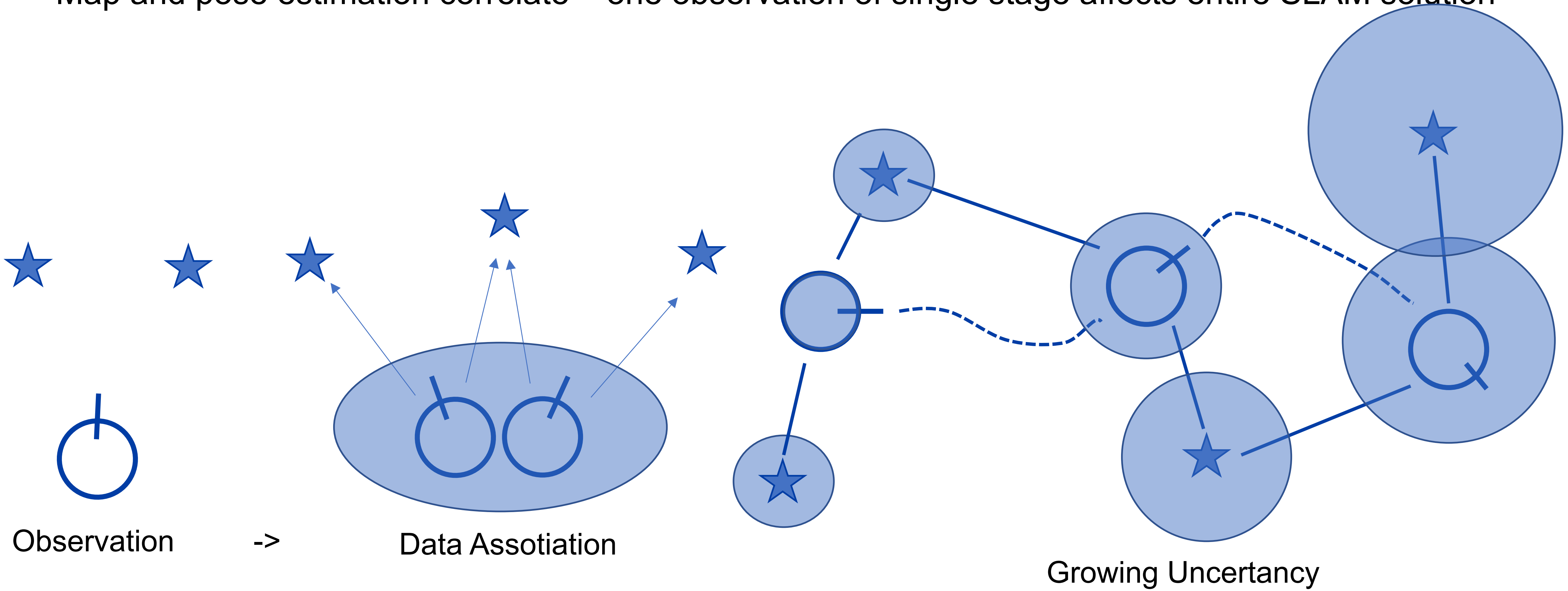
Online SLAM

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \dots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \dots dx_{t-1}$$

$$\underbrace{p}_{\text{Probability Distribution}}(\underbrace{x_{0:T}}_{\text{Path}}, \overbrace{m}^{\text{Map}} \mid \underbrace{z_{1:T}}_{\text{Observations}}, \overbrace{u_{1:T}}^{\text{Control}})$$

Why is SLAM Hard?

Robot‘s path and map of the environment are both unknown
Map and pose estimation correlate – one observation of single stage affects entire SLAM solution



Human SLAM vs Machine SLAM

Human SLAM

Brain evolved in way, it is good in solving highly abstract tasks, like:

- **Complex pattern matching**
- Extracting feature points (remember important marks in the environment)
- Constructing topological representation of the environment (building map in our head)

Human perception has **high uncertainty**.

Machine SLAM

Very fast and precise calculations and **measurements**

Limited computational power

Fast, but very specialized **pattern matching** and **feature point extraction**

Localization and “Mapping“ on Mars - Ingenuity

Visual Odometry Technique

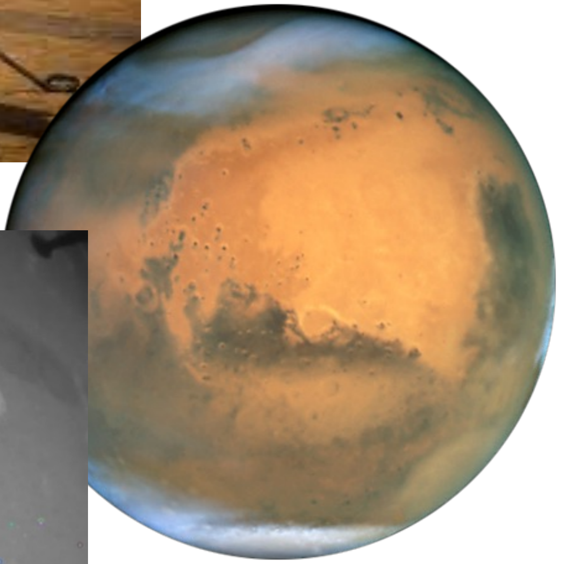
Sensory Equipment:

IMU: Bosch Sensortec BMI-160
Camera: Omnivision OV7251
Laser rangefinder: Garmin Lidar-Lite-V3

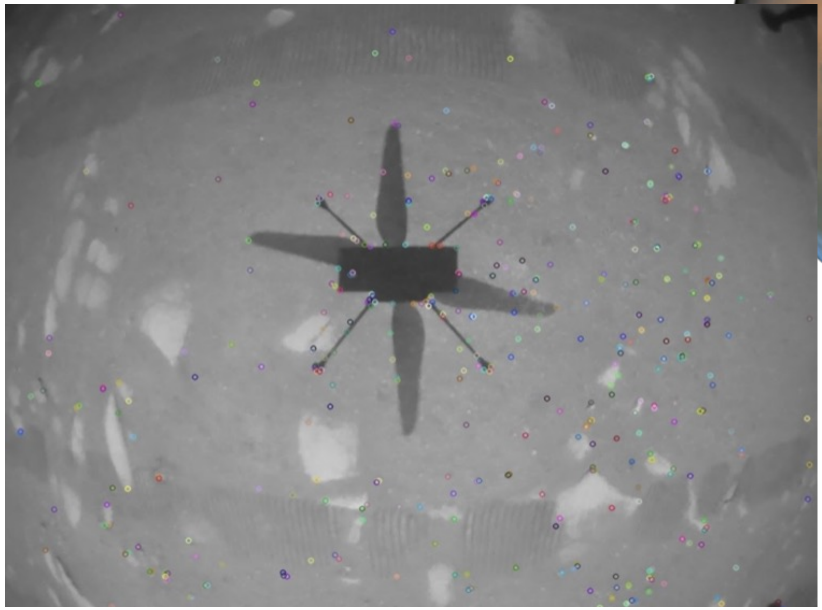
NO GNSS
NO radio navigation

Navigation Principle:

Detecting FAST featurepoints in images.
Matching featurepoints between images.
Considering the motion of the featurepoints
between the images robot estimates
the change of the position.
Combined with IMU data.



www.kosmonautix.cz



https://www.therobotreport.com/ingenuity-helicopter-state-estimation-localization/

Taxonomy of SLAM

Volumetric models full env	**vs**	**Featured** models only important points of the env
Topological models struct. of important points	**vs**	**Geometric** models exact geometry of the env.
Known	**vs**	**Unknown Data Association**
Static	**vs**	**Dynamic Environment**
Active Robot takes its own decisions how to build a map	**vs**	**Passive SLAM** Robot is controlled by other entity
Single Robot	**vs**	**Swarm SLAM**

Bayes Filter

Bayes Filter is **generic mathematical framework** to estimate inner states of the system (world).

Prediction:

Motion Model

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

Correction:

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

Observation Model

Most Common Paradigmas in SLAM

	Kalman Filter	Particle Filter	Graph Based
Approaches:	**Kalman Filter** **Extended Kalman Filter** **Unscented Kalman Filter** **Information Filter** **Sparse Extended Information Filter**	**Particle Filter**	**Least Squares**
SLAMs:	**EKF SLAM** **SEIF SLAM**	**FAST SLAM** **Grid-based SLAM with PF**	**Least Squares SLAM**

General Kalman Filter

Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

$$\bar{\mu}_t = A\mu_{t-1} + B\,u_t$$

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

Prediction

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

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

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

Correction

return μ_t, Σ_t

Kalman vs Information Filter

- Kalman is cheap for states update, but very expensive for states correction (matrix inversion)

- Information filter is a different implementation of the bayes filter, that makes states update expensive, but states correction is very cheap

$$\Sigma = \Omega^{-1} \qquad \Omega = \Sigma^{-1} \qquad \Omega \ldots \text{ Information matrix}$$

$$\mu = \Omega^{-1}\varepsilon \qquad \varepsilon = \Sigma^{-1}\mu \qquad \varepsilon \ldots \text{ Information vector}$$

- Information matrix is usually very sparse (majority of components are 0 or close to 0). It allows to ignore them and make matrix multiplication and inversion very fast

$$\mu_{t-1} = \Omega_{t-1}^{-1}\varepsilon_{t-1}$$

$$\bar{\Omega}_t = (G_t \Omega_{t-1}^{-1} G_t^T + R_t)^{-1}$$

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

$$\bar{\varepsilon}_t = \bar{\Omega}_t \bar{\mu}_t$$

$$\Omega_t = \bar{\Omega}_t + H_t^T Q_t^{-1} H_t$$

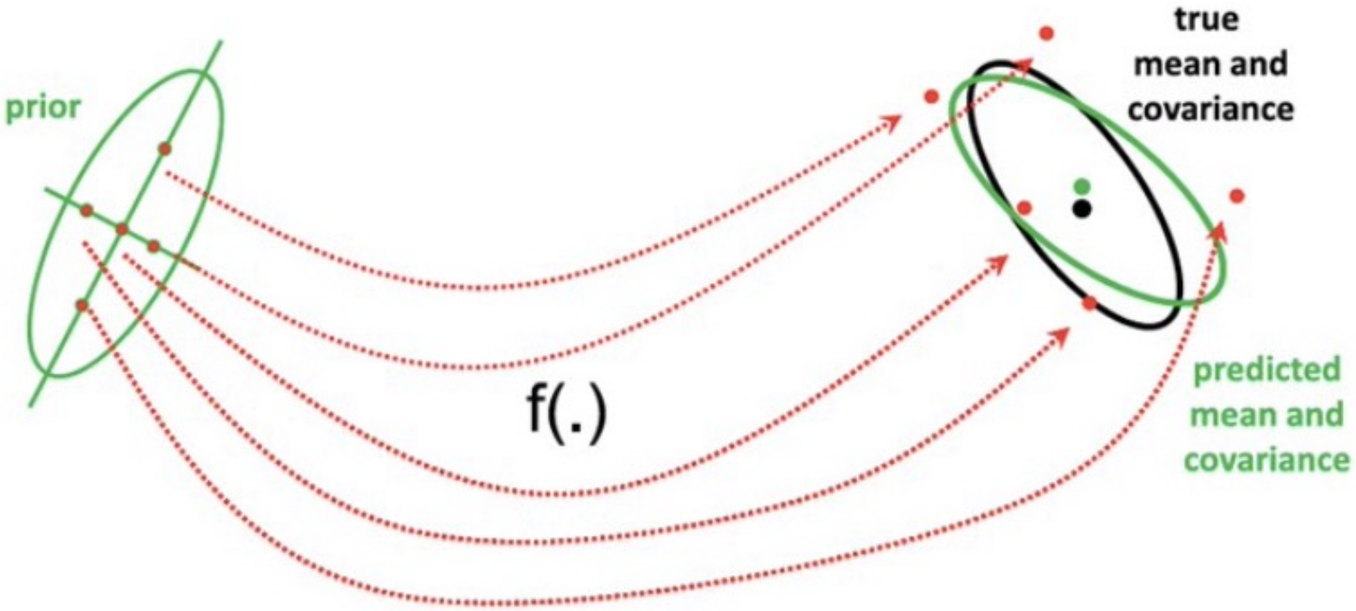
$$\varepsilon_t = \bar{\varepsilon}_t + H_t^T Q_t^{-1}(z_t - h(\bar{\mu}_t) + H_t \bar{\mu}_t)$$

Unscented Kalman Filter

Kalman Filter – linear models
EKF – linearization via Taylor expansion

Unscented Kalman Filter (UKF) – applying non-linear transformation on gaussian variable.

- Based on state vector and covariance matrix generate "sigma points"
- Apply non linear transformation on sigma points
- Recover state vector and covariance matrix from transformed sigma points



[UKF; Gabriel Terejanu - https://slideplayer.com/slide/9280356/]

Extended Kalman Filter SLAM (EKF SLAM)

Extended Kalman Filter SLAM (EKF SLAM)

Basic principle:

Using the Extended Kalman Filter (EKF) to model the robot‘s pose and pose of all observerd landmarks.

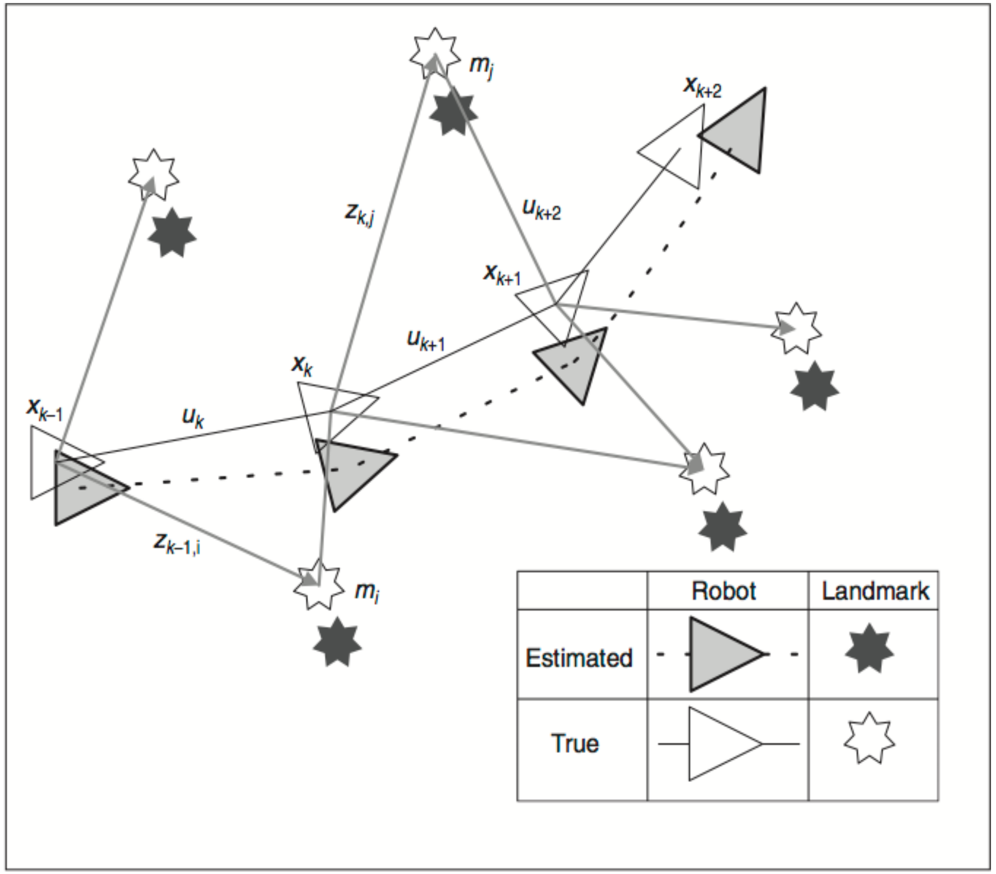
$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$

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

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

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

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



http://gopalmenon.github.io/Raspberri-Pi-GoPiGo-Robot-EKF-SLAM-Manuscript/images/EssSlam.png

Handy overview available at:
https://www.iri.upc.edu/people/jsola/JoanSola/objectes/curs_SLAM/SLAM2D/SLAM%20course.pdf

Extended Kalman Filter SLAM (EKF SLAM)

$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \cdots \\ m_{nx} \\ m_{ny} \end{bmatrix} \qquad \mathbf{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \cdots & \Sigma_{xr,m1y} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \Sigma_{m1yr,xr} & \cdots & \cdots & \Sigma_{m1y,m1y} \end{bmatrix}$$

Motion Model

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

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

Observation Model

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

Prediction

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

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

Correction

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

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

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

return μ_t, Σ_t

EKF SLAM – Filter Cycle Phases

1) Motion Prediction

2) Measurement Prediction

3) Measurement

4) Data Association

5) Correction

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

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

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

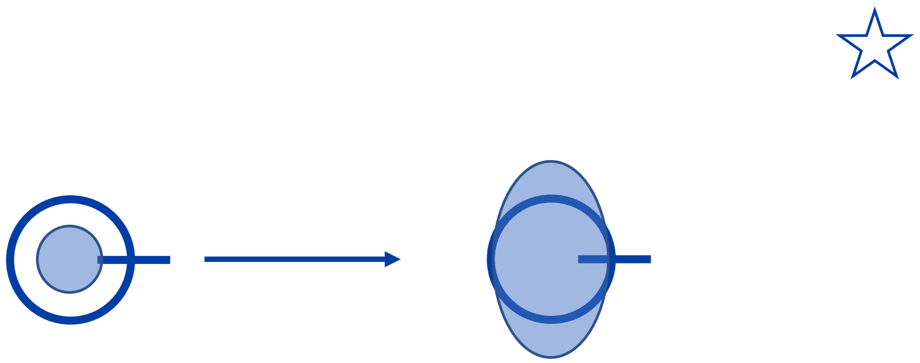
$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$

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

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

return μ_t, Σ_t

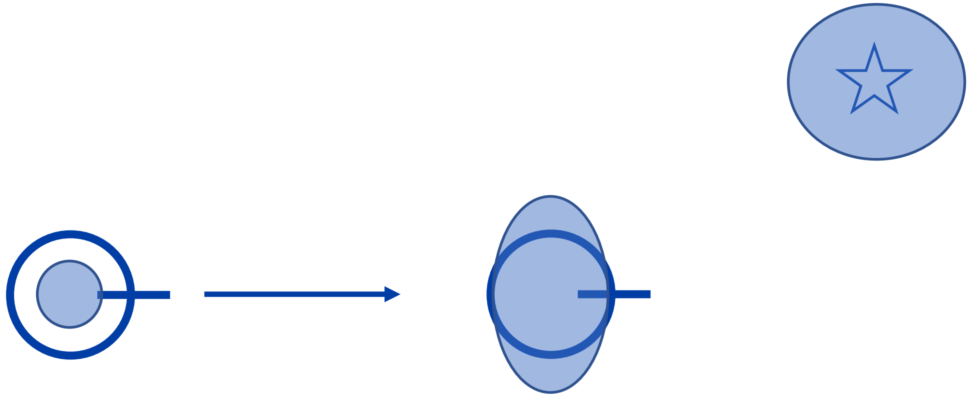
EKF SLAM Phases: Motion Preiction (State Prediction)



$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \dots \\ m_{nx} \\ m_{ny} \end{bmatrix}$$

$$\mathbf{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \dots & \Sigma_{xr,mny} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \Sigma_{mnyr,xr} & \dots & \dots & \Sigma_{mny,mny} \end{bmatrix}$$

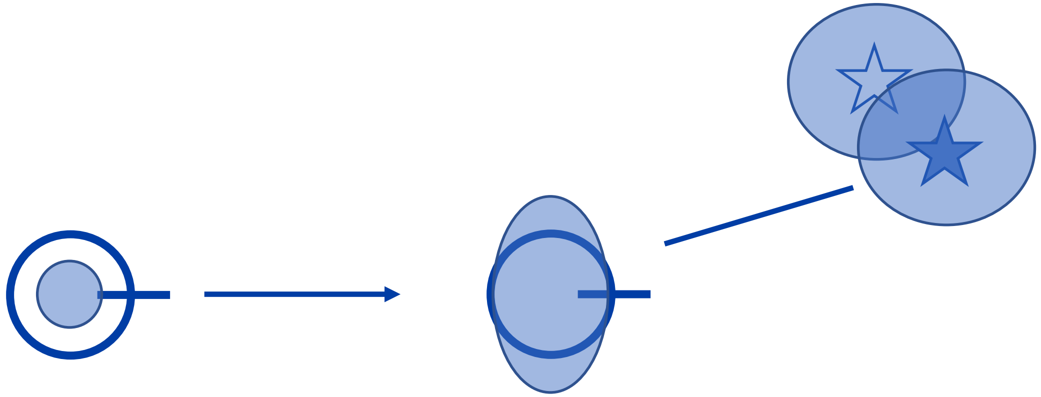
EKF SLAM Phases: Predict Measurement



$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \dots \\ m_{nx} \\ m_{ny} \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \dots & \Sigma_{xr,mny} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \Sigma_{mnyr,xr} & \dots & \dots & \Sigma_{mny,mny} \end{bmatrix}$$

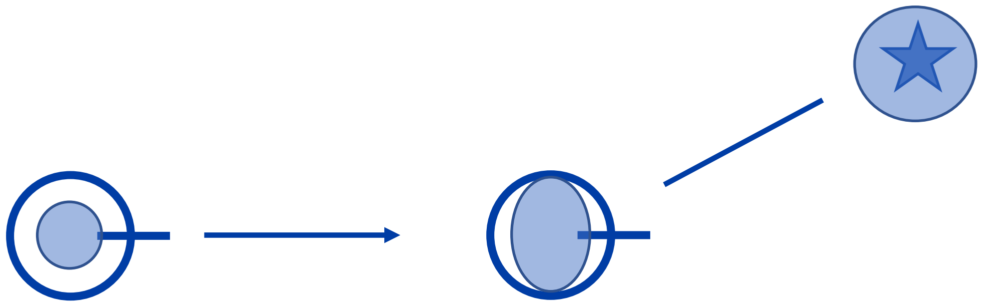
EKF SLAM Phases: Measurement



$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \dots \\ m_{nx} \\ m_{ny} \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \dots & \Sigma_{xr,mny} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \Sigma_{mnyr,xr} & \dots & \dots & \Sigma_{mny,mny} \end{bmatrix}$$

EKF SLAM Phases: Correction



$$\mu = \begin{bmatrix} x_{robot} \\ y_{robot} \\ \theta_{robot} \\ m_{1x} \\ m_{1y} \\ \dots \\ m_{nx} \\ m_{ny} \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \Sigma_{xr,xr} & \Sigma_{xr,yr} & \dots & \Sigma_{xr,mny} \\ \Sigma_{yr,xr} & \Sigma_{yr,yr} & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \Sigma_{mnyr,xr} & \dots & \dots & \Sigma_{mny,mny} \end{bmatrix}$$

Robotic setup for our example:

- Platform moving in 2D
- Velocity based motion model
- Observation of point landmarks
- Range-bearing sensor
- Known data association
- Known number of landmarks

EKF SLAM - Initialization

- The robot takes it‘s original position as a origin of the entire map
- Robot know its position with absolute certancy
- Robot know nothing about landmarks
- Robot does not know any relation between itself, lendmarks, and relation between landmarks
- $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ dims: 3 + 2 * num_of_landmarks num_of_landmarks -> N

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \infty & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & \infty \end{bmatrix}$$

1) Motion Prediction
2) Measurement Prediction
3) Measurement
4) Data Association
5) Correction

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

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

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

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

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

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

return μ_t, Σ_t

EKF SLAM – Motion Model

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}v * \cos(\theta)\\ v * sin(\theta)\\ \omega\end{bmatrix} * \Delta t$$

$$\begin{bmatrix}x'\\ y'\\ \theta'\end{bmatrix} = \begin{bmatrix}x\\ y\\ \theta\end{bmatrix} + \begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}x\\ y\\ \theta\end{bmatrix} + \begin{bmatrix}v * \cos(\theta)\\ v * sin(\theta)\\ \omega\end{bmatrix} * \Delta t$$

$$\bar{\mu}_t = \begin{bmatrix}x'\\ y'\\ \theta'\end{bmatrix} = g(\boldsymbol{u_t}, \mu_{\boldsymbol{t-1}}) = \begin{bmatrix}x + v * \cos(\theta) * \Delta t\\ y + v * sin(\theta) * \Delta t\\ \theta + \omega * \Delta t\end{bmatrix}$$

$$\bar{\mu}_t = \begin{bmatrix}x'\\ y'\\ \theta'\\ \dots\end{bmatrix} = \begin{bmatrix}x\\ y\\ \theta\\ \dots\end{bmatrix} + \boldsymbol{F} * \begin{bmatrix}v * \cos(\theta)\\ v * \sin(\theta)\\ \omega\end{bmatrix} * \Delta t = \begin{bmatrix}x\\ y\\ \theta\\ \dots\end{bmatrix} + \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\\ \dots & \dots & \dots\\ 0 & 0 & 0\end{bmatrix} * \begin{bmatrix}v * \cos(\theta)\\ v * \sin(\theta)\\ \omega\end{bmatrix} * \Delta t$$

1) Motion Prediction

2) Measurement Prediction

3) Measurement

4) Data Association

5) Correction

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

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

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

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

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

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

return μ_t, Σ_t

EKF SLAM – Jacobian of Motion

Function $g(\boldsymbol{u_t}, \mu_{\boldsymbol{t-1}})$ affects only robot's states, not the landmarks

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

$$\boldsymbol{G_t} = \begin{bmatrix} \boldsymbol{G_t^r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}$$

$\boldsymbol{G_t^r}$ -> 3x3

$\boldsymbol{I}$ -> 2N x 2N

$$\boldsymbol{G_t^r} = \frac{\delta}{\delta(x,y,\theta)}[g(u_t, \mu_{t-1})] = \frac{\delta}{\delta(x,y,\theta)}\left(\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} v * \cos(\theta) \\ v * sin(\theta) \\ \omega \end{bmatrix} * \Delta t\right)$$

$$\boldsymbol{G_t^r} = \boldsymbol{I} + \begin{bmatrix} 0 & 0 & -v * \sin(\theta) * \Delta t \\ 0 & 0 & v * cos(\theta) * \Delta t \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -v * \sin(\theta) * \Delta t \\ 0 & 1 & v * cos(\theta) * \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

R_t ... User defined matrix of motion uncertancy, usually diagonal matrix

EKF SLAM – Update Covariance

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

$$\bar{\Sigma}_t = \begin{bmatrix} \boldsymbol{G}_t^r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{bmatrix} \begin{bmatrix} (\boldsymbol{G}_t^r)^T & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} + \boldsymbol{R_t}$$

$$\bar{\Sigma}_t = \begin{bmatrix} \boldsymbol{G}_t^r \Sigma_{xx} (\boldsymbol{G}_t^r)^T & \boldsymbol{G}_t^r \Sigma_{xm} \\ (\boldsymbol{G}_t^r \Sigma_{xm})^T & \Sigma_{mm} \end{bmatrix} + \boldsymbol{R_t}$$

R_t … User defined matrix of motion uncertancy, usually diagonal matrix (robot pose covariance only)

Updating covariance affects only robot’s pose uncertainty and robot-landmark pose covariance.

Landmarks’ pose uncertainty and landmark-landmark pose covariance stays unchanged.

EKF SLAM - Correction

Assumptions:

Known data association
Range-bearing sensor

Components:

$\bar{\mu}_t$	...	predicted states
z_t	...	measurement
$h(\bar{\mu}_t)$	...	predicted measurement
H_t	...	measurement Jacobian
K_t	...	Kalman gain
Q_t	...	measurement noise
μ_t	...	corrected states
Σ_t	...	corrected covariance

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

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

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

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

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

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

return μ_t, Σ_t

EKF SLAM – Range-bearing sensor model

$$z_i = [r_i \quad \phi_i]^T$$

$$\begin{array}{cccc} & \text{Robot‘s pose} & & \\ \begin{bmatrix} \mu_{ix} \\ \mu_{iy} \end{bmatrix} = & \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} & + & \begin{bmatrix} r_i * \cos(\phi_i + \mu_\theta) \\ r_i * \sin(\phi_i + \mu_\theta) \end{bmatrix} \\ \text{Landmark‘s pose} & & & \text{Robot’s measurement} \end{array}$$

$$\delta = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} \mu_{ix} - \mu_x \\ \mu_{iy} - \mu_y \end{bmatrix} \qquad q = \delta^T \delta$$

$$h_i(\bar{\mu}_t) = \begin{bmatrix} \sqrt{q} \\ atan2(\delta_y, \delta_x) - \mu_\theta \end{bmatrix} = \bar{z}_i$$

$$h_i(\bar{\mu}_t) = \begin{bmatrix} \sqrt{q} \\ atan2\left(\delta_y, \delta_x\right) - \mu_\theta \end{bmatrix} = \bar{z}_i$$

$$H_{i-low} = \frac{\delta h(\bar{\mu}_t)}{\delta \bar{\mu}_t} = \begin{bmatrix} \dfrac{\delta\sqrt{q}}{\delta x} & \dfrac{\delta\sqrt{q}}{\delta y} & \ldots \\ \dfrac{\delta atan2(\ldots)}{\delta x} & \dfrac{\delta atan2(\ldots)}{\delta y} & \ldots \end{bmatrix}$$

$$H_{i-low} = \frac{1}{q}\begin{bmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & \sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{bmatrix}$$

$$H_i = H_{i-low} * F$$

$(x, y, \theta, \mu_{ix}, \mu_{iy})$

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 1 & 0 & \ldots & 0 \end{bmatrix}$$

EKF SLAM - Implementation

Implementation tips:

Always normalize the angular components between π and –π

F matrix is not necessary – it is just helps to understand topic

Landmarks association: usually nearest neighbour

Extended Kalman Filter$(\mu_{t-1}, \Sigma_{t-1}, \mu_{t-1}, z_t)$

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

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

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

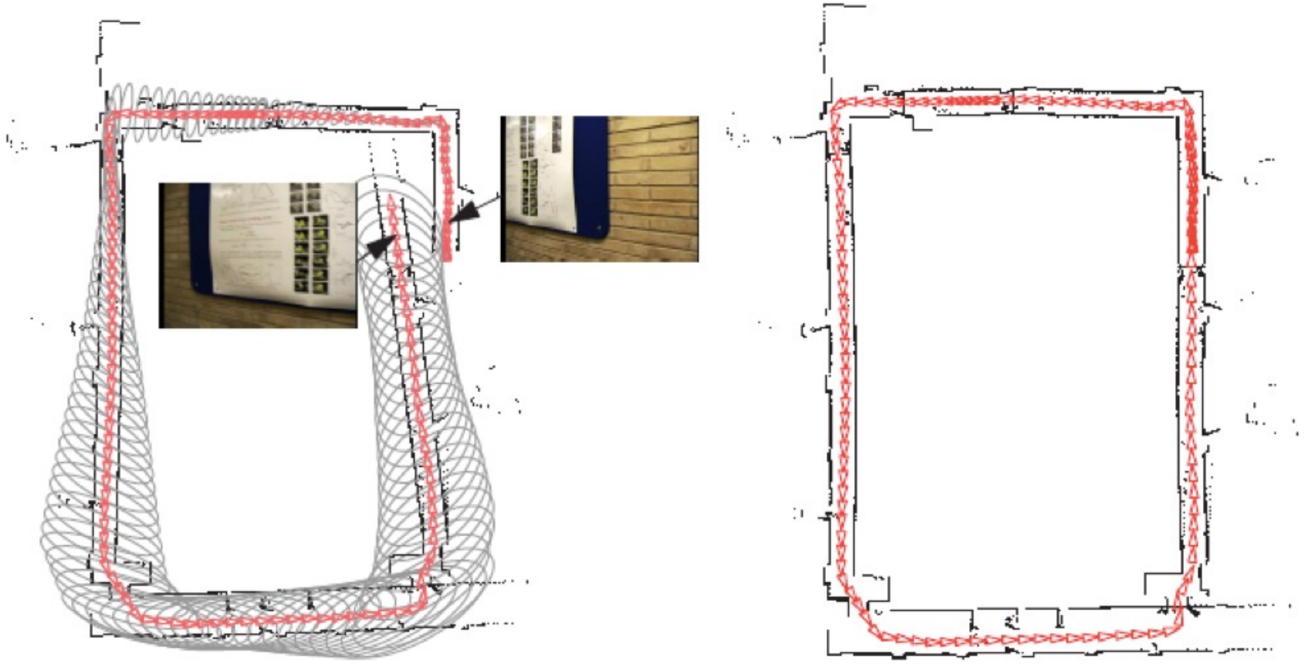
$$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$$

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

return μ_t, Σ_t

EKF SLAM – Loop Closure

- Loop closing allows distributing low uncertainty through the map
- Helps to reduce cumulated positioning error by creating relations between observations with high and low error
- Wrong loop closure assumption leads to map divergence

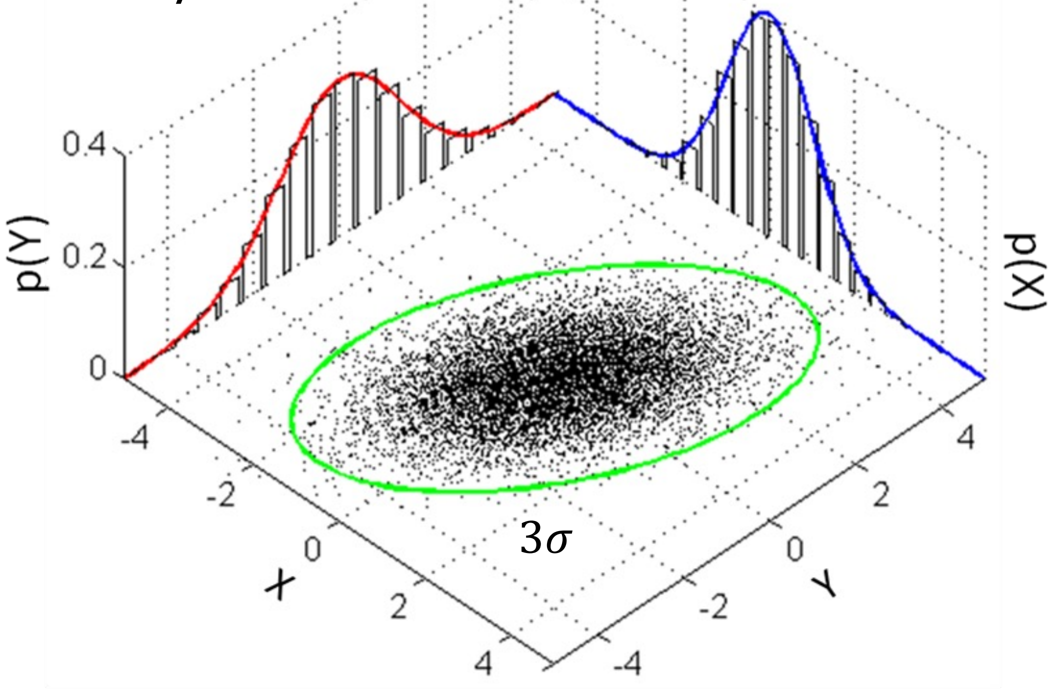


Elasmar, S.R.A.G., 2012. Information-theoretic environment modeling for mobile robot localization.

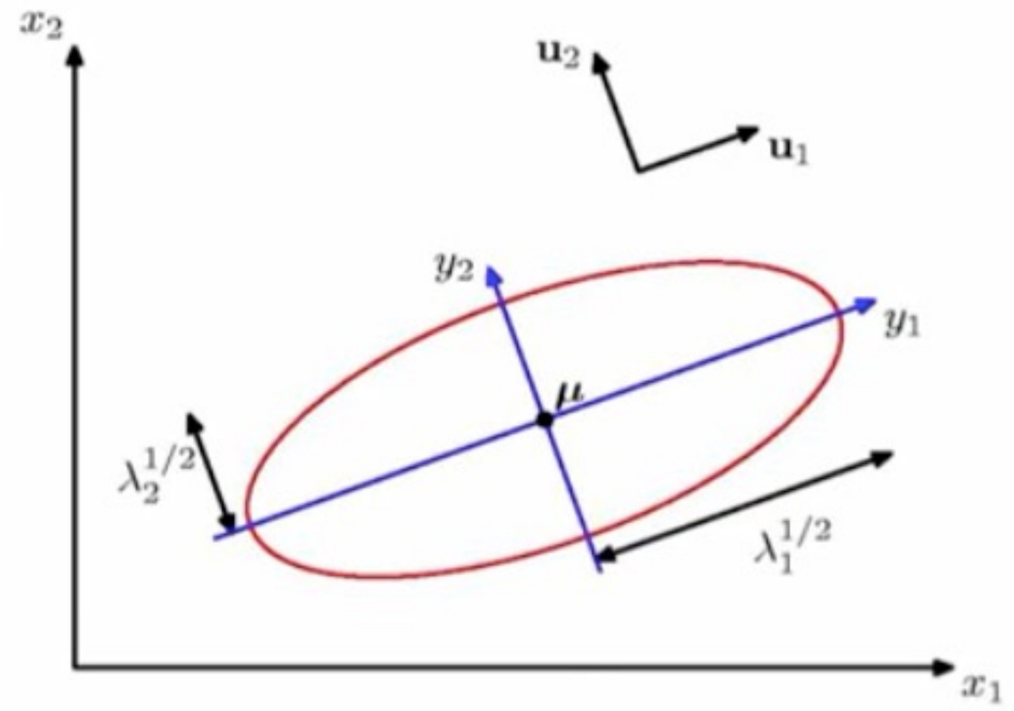
EKF SLAM – Covariance Matrix

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 3/5 \\ 3/5 & 2 \end{bmatrix}$$



$$\Sigma = U\Lambda U^T = \begin{bmatrix} u_1 & u_2 \\ u_1 & u_2 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u_1 & u_1 \\ u_2 & u_2 \end{bmatrix}$$

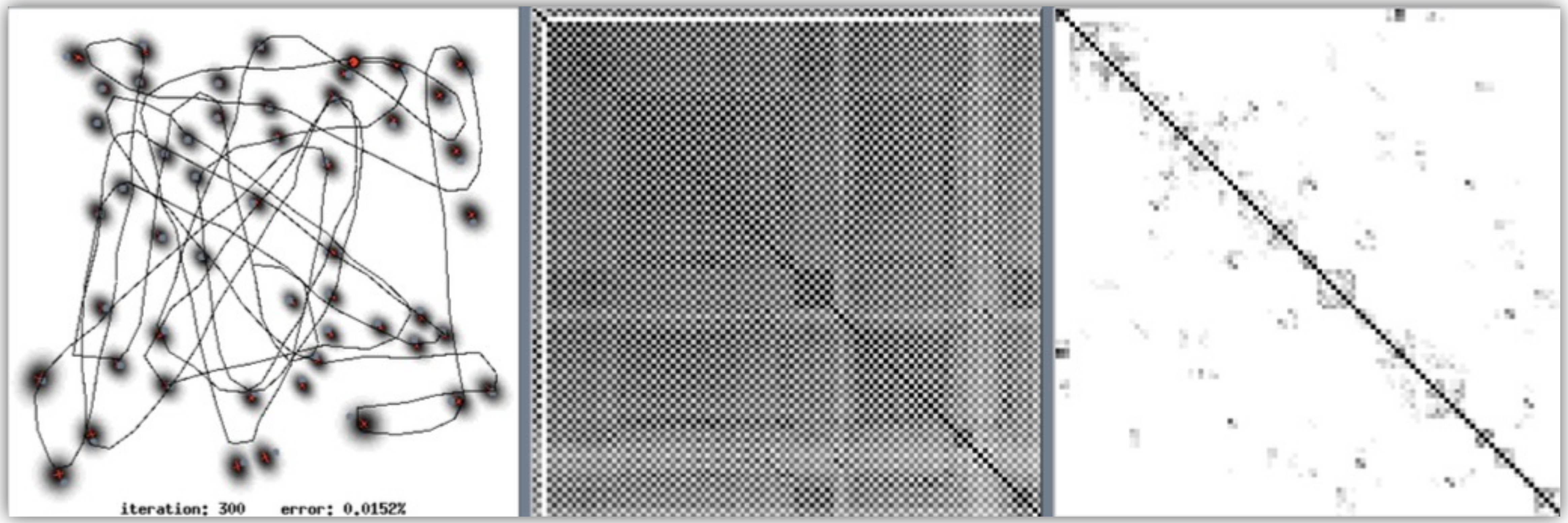


[1] https://en.wikipedia.org/wiki/Multivariate_normal_distribution
[2] https://www.youtube.com/watch?v=eho8xH3E6mE

EKF SLAM – Covariance Matrix

Map with landmarks' uncertainty

Information matrix



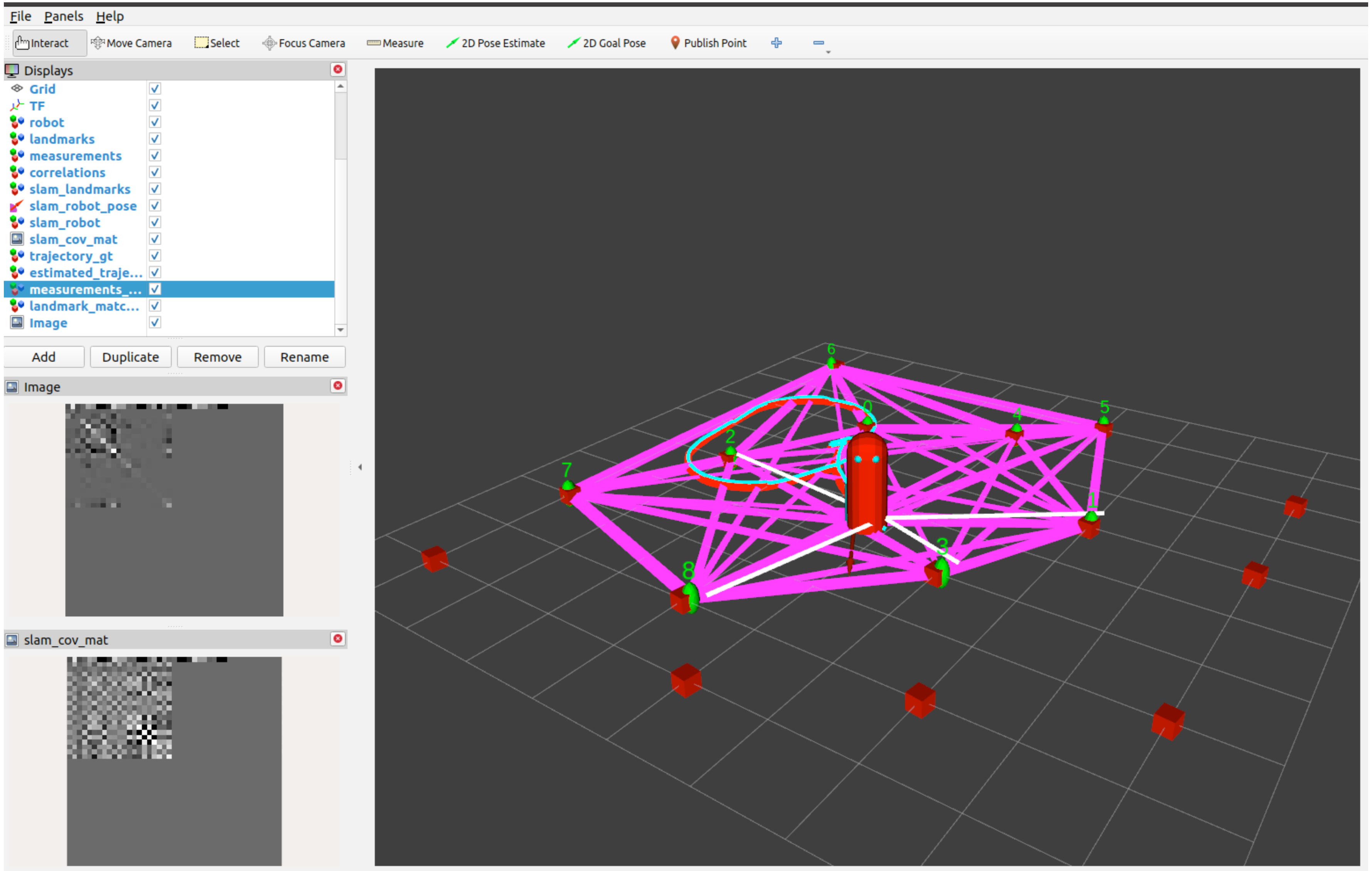
Covariance matrix

[Probabilistic Robotics]

EKF SLAM - Summary

- Algorithm converges for linear Gaussian cases
- Diverges for large non-linearities
- Unimodal states estimation
- Works for small up to mid-size scenes
- Used for short term estimations (visual odometry)
- Complexity: computational – $O(n^2)$, memory - $O(n^2)$

EKF SLAM - Example

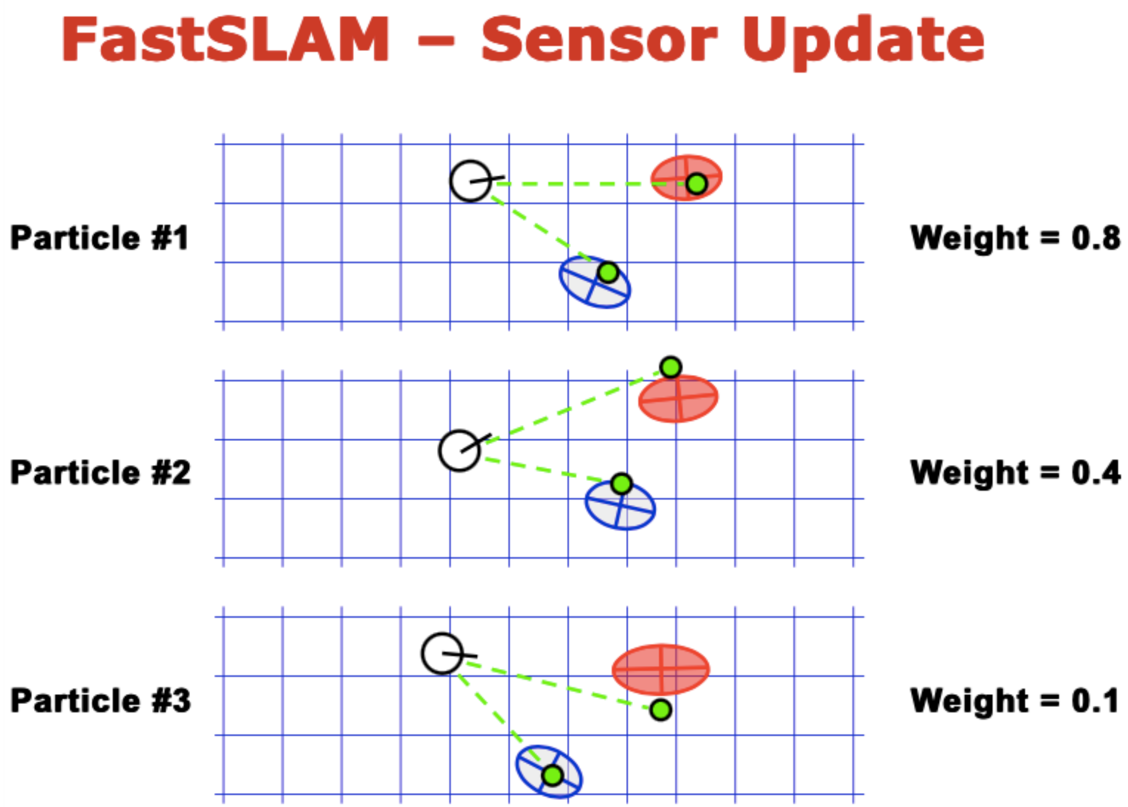


https://github.com/adamek727/EKF-SLAM-Example

FastSLAM

FastSLAM

- Combination of Particle Filter and Kalman Filter
- FastSLAM keeps large number of particles. Each particle represents entire map.
- Each particle uses multiple Kalman Filters to represent previously observer landmarks
- The better particle corresponds with current observation, the higher chance it has to survive
- Each particle represents different hypotesis, how the world looks like
- Lower complexity compared to Kalman Filter
  KF: $O(m^2)$
  FastSLAM: $O(n\log(m))$, n ... no of particles
  m ... no of landmarks



[Thrun, Probabilistic Robotics, https://slideplayer.com/slide/10143612/]

Least Squares (Graph) SLAM

Least Squares (Graph) SLAM

SLAM is based on building graph and numerical optimization

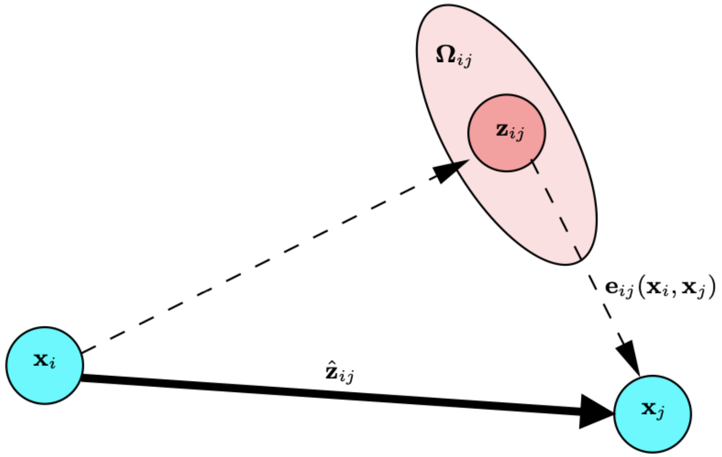
- **Graph Nodes**
  - List of robot poses
  - Important parts of the map
- **Graph Edges** – Relation between nodes
  - Odometry
  - Environment observations (measurements)

$$\bar{e}_i(x) = z_i - f_i(x)$$

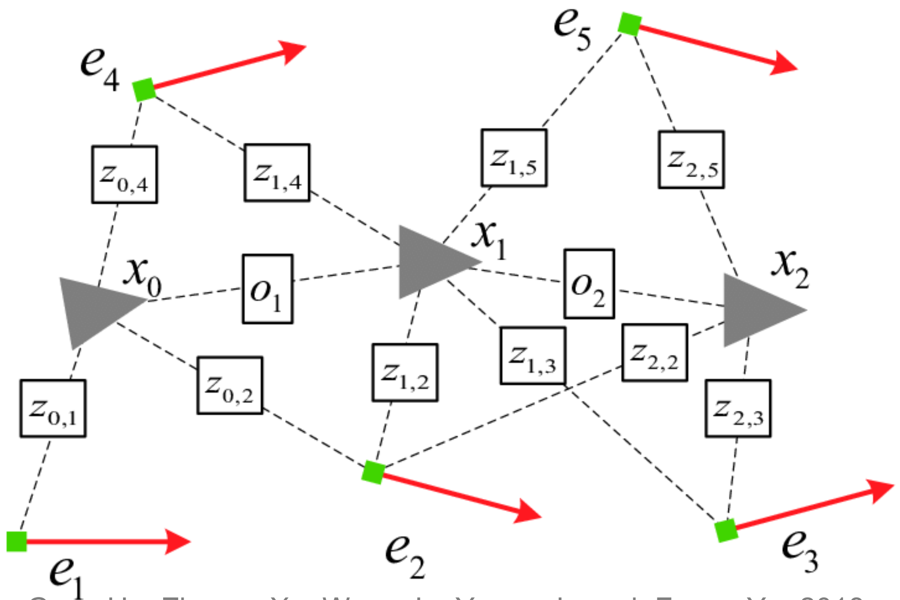
$$e_i(x) = \bar{e}_i(x)^T \Omega^{-1} \bar{e}_i(x)$$

$$x^* = \underset{x}{\operatorname{argmin}} \sum_i e_i(x)$$

Graph SLAM allows to create hierarchical strutures



http://www2.informatik.uni-freiburg.de/~stachnis/pdf/grisetti10titsmag.pdf



Gao, H., Zhang, X., Wen, J., Yuan, J. and Fang, Y., 2018. Autonomous indoor exploration via polygon map construction and graph-based SLAM using directional endpoint features.

SLAM Comparison

	KF	EKF	SEIF	PF	Graph
Complexity	n^2	n^2	const	M lon(n) *	edges-related
Assumpt. Dist.	gauss	gauss	gauss	pose: any landmrk: gauss	gauss + outlayers
Linearization	all linear	once	once	no needed	re-linearization**
Flexibility	medium	medium	medium	good	good
Large Scalse SLAM	bad	bad	ok	ok	good

* ... M – number of particles; N – number of landmarks

** ... closer to ground truth solution -> better linearization

Adam Ligocki

adam.ligocki@vutbr.cz

Brno University of Technology
Faculty of Electrical Engineering and Communication
Department of Control and Instrumentation

Robotics and AI
Research Group