SLAM: Simultaneous Localization and Mapping

Dr Gaurav Pandey

Assistant Professor

IIT Kanpur

<u>NOTE: The slides have been adapted from the lecture slides of Cyrill Stachniss from University of Freiburg</u>

Localization

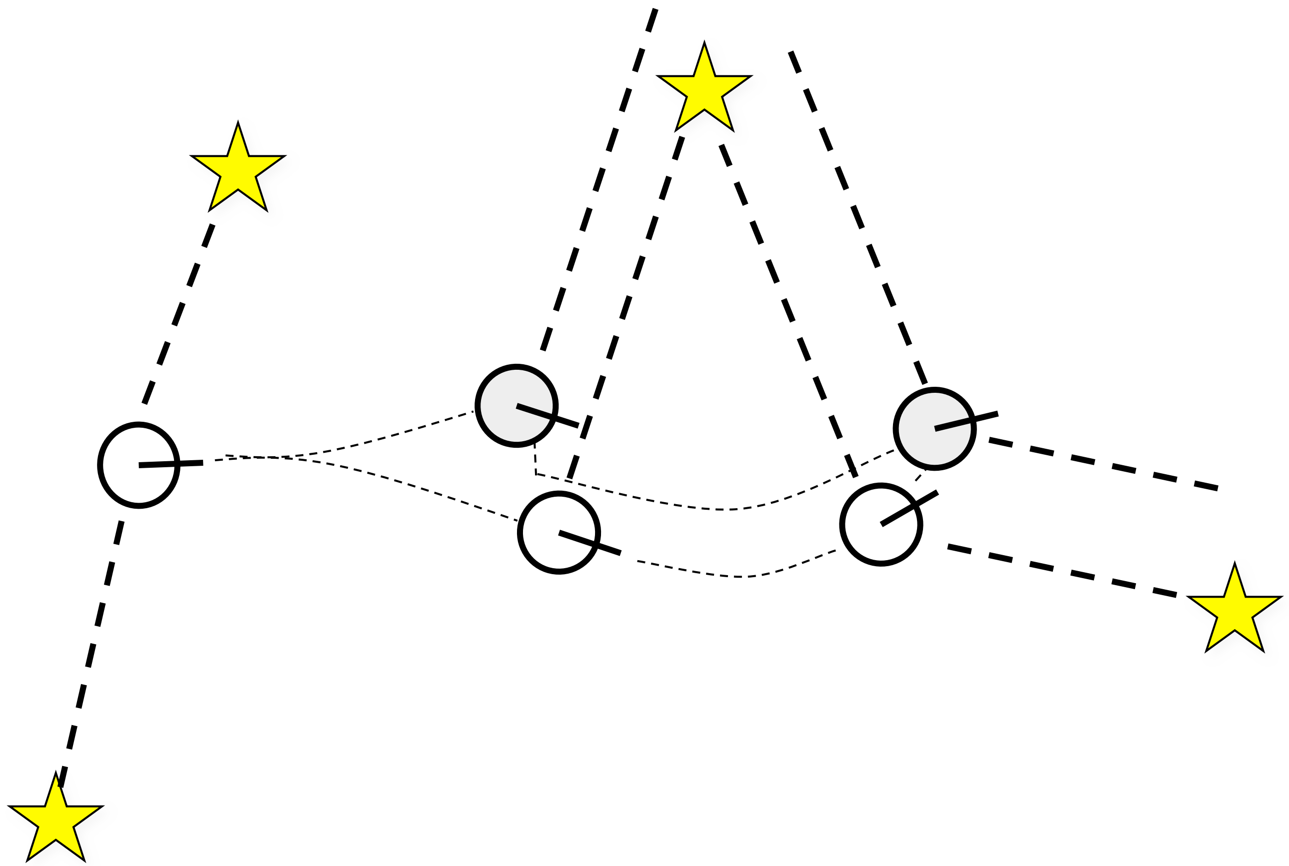


Figure Courtesy: Cyrill Stachniss

Mapping

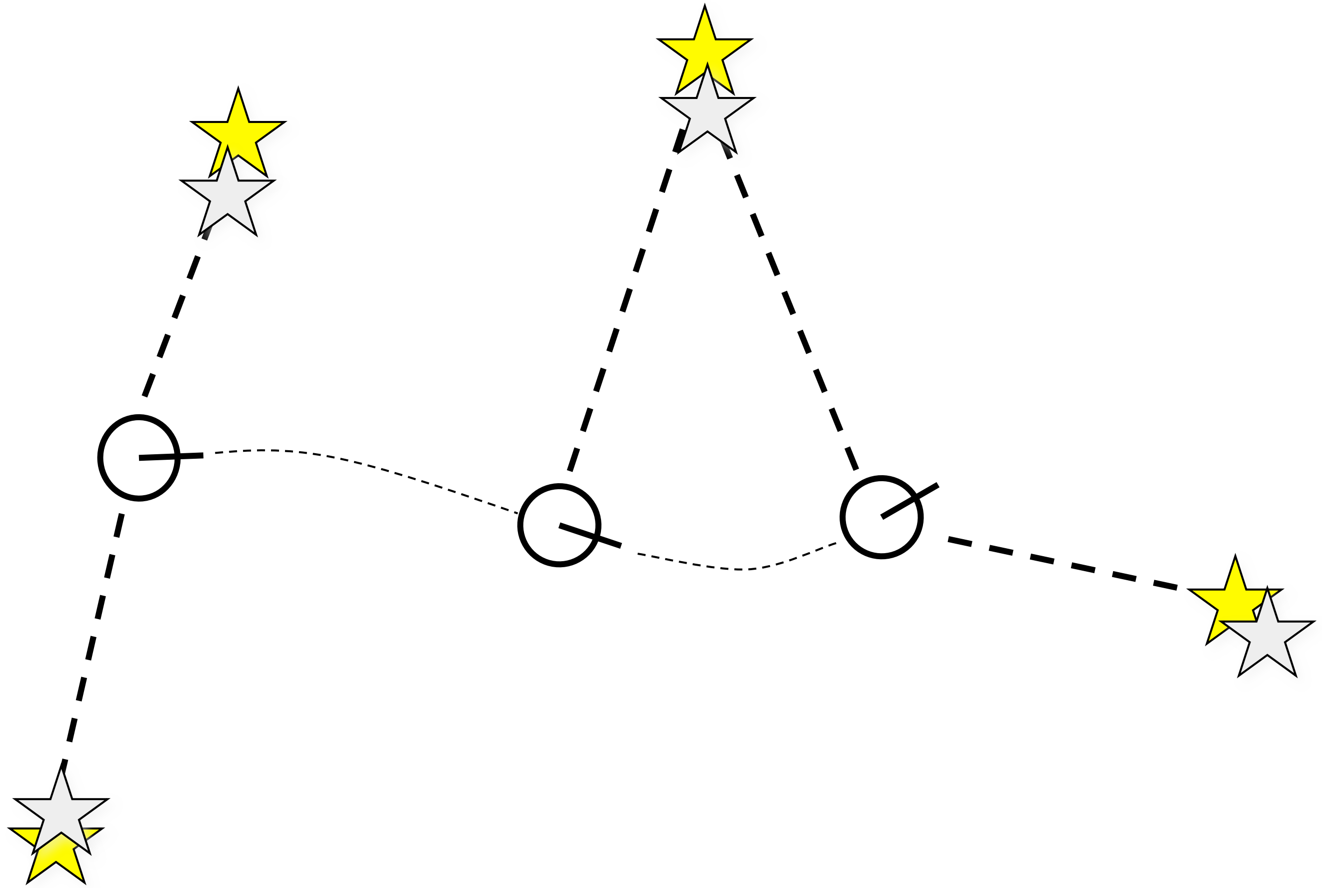


Figure Courtesy: Cyrill Stachniss

SLAM: Simultaneous Localization and Mapping

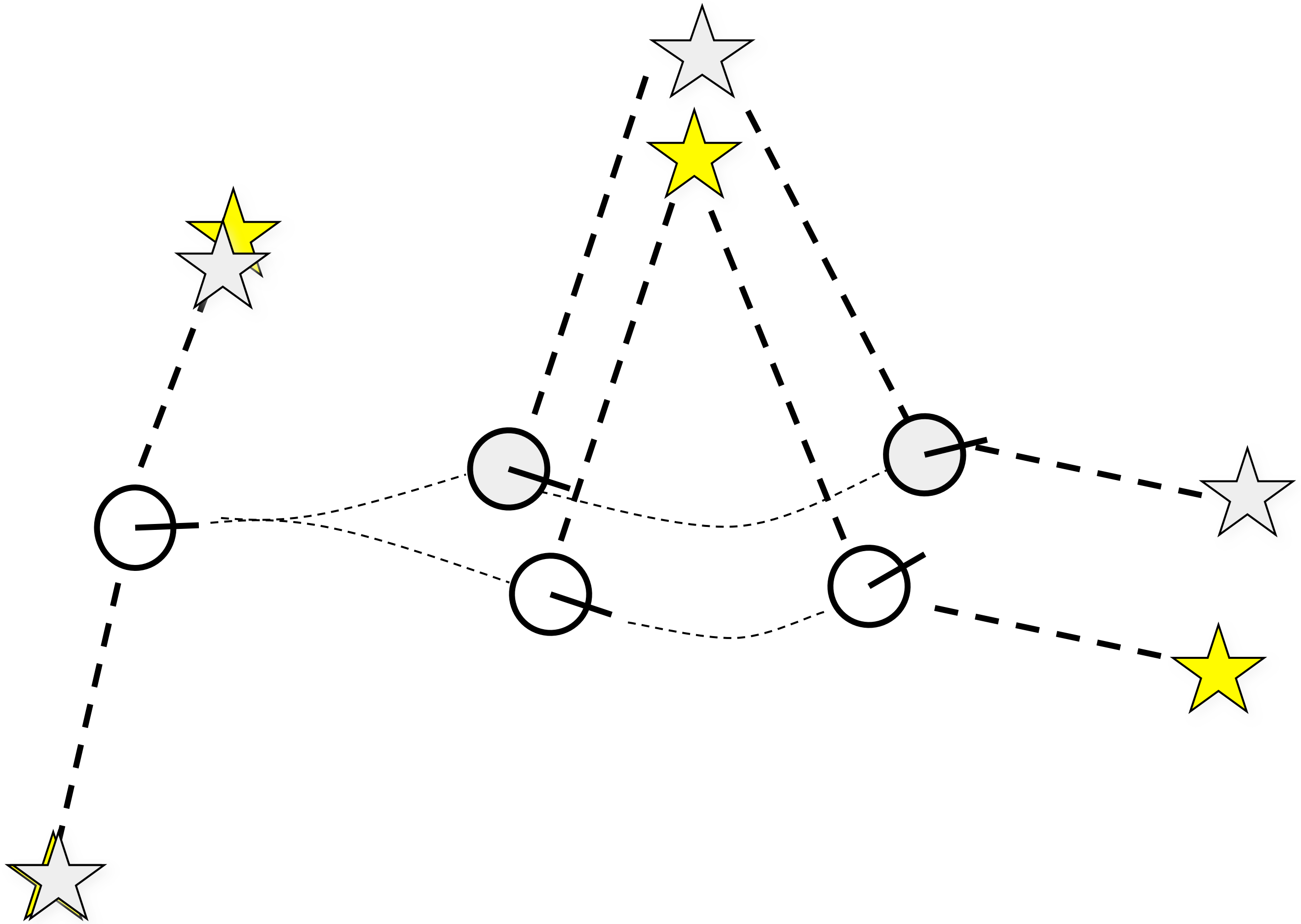


Figure Courtesy: Cyrill Stachniss

SLAM: Definition

- Given:
  - Robot Control
    $u_{1:T} = \{u_1, u_2, u_3 \ldots, u_T\}$
  - Observations / Measurements
    $z_{1:T} = \{z_1, z_2, z_3 \ldots, z_T\}$
- Estimate:
  - Map
    M
  - Robot Path / Trajectory
    $x_{0:T} = \{x_0, x_1, x_2 \ldots, x_T\}$

<u>Probabilistically</u>

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution path map given observations controls

Full/Offline SLAM vs Online SLAM

- Full/Offline SLAM estimates the map and complete robot trajectory. Often solved using graphical models and least squares optimization.

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

- Online SLAM only estimates the most recent pose and the map. Often solved using filtering techniques (EKF, EIF, Particle filter)

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

Why is SLAM a hard problem ?

- Robot path and Map both are unknown. Chicken-and-egg problem.

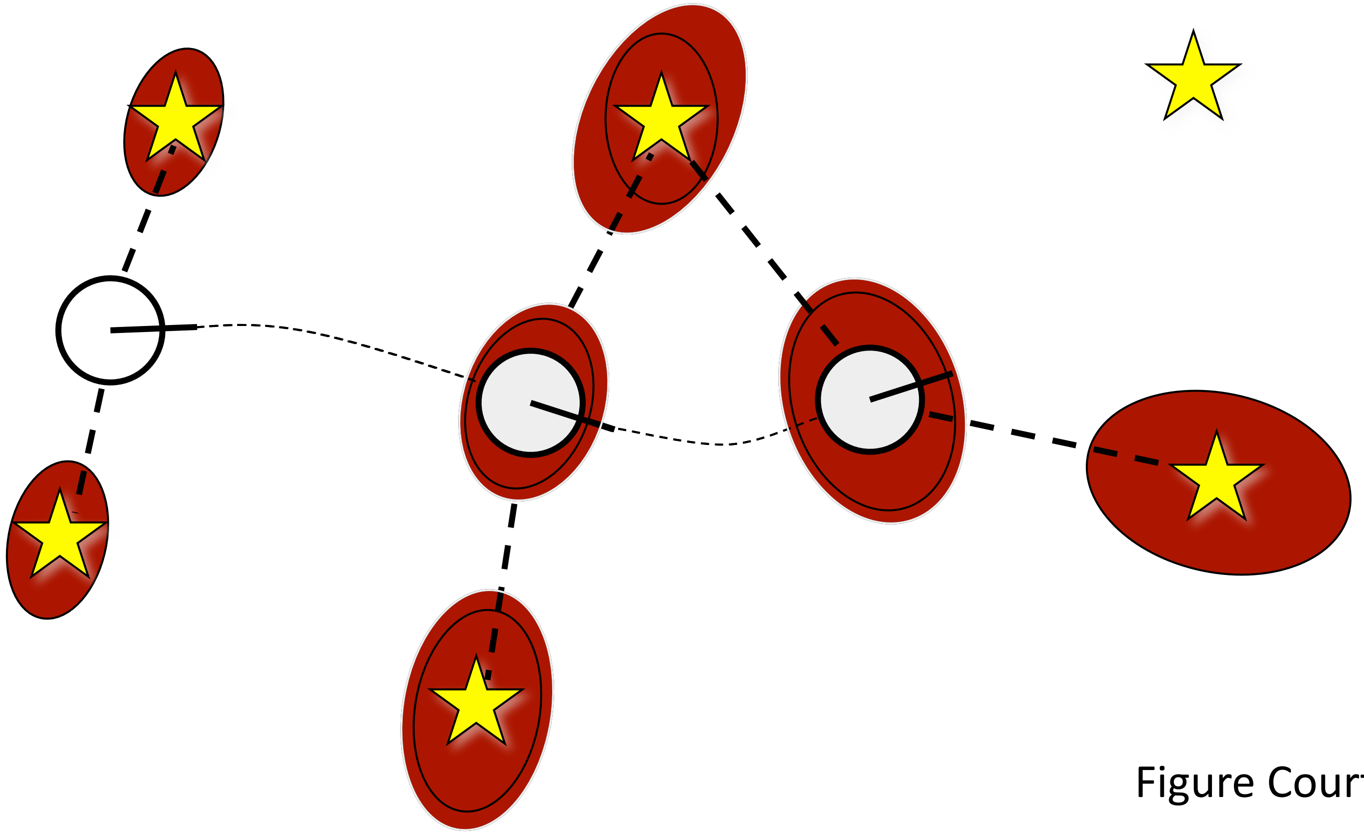


Figure Courtesy: Cyrill Stachniss

Why SLAM is a hard problem ?

- Data association is unkown.

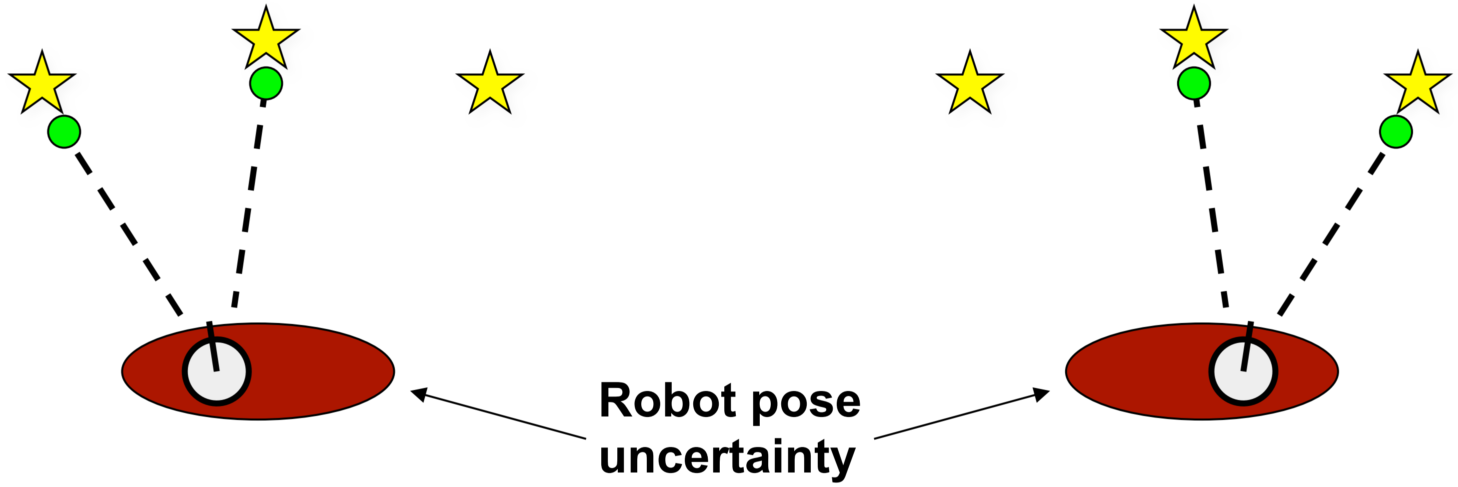
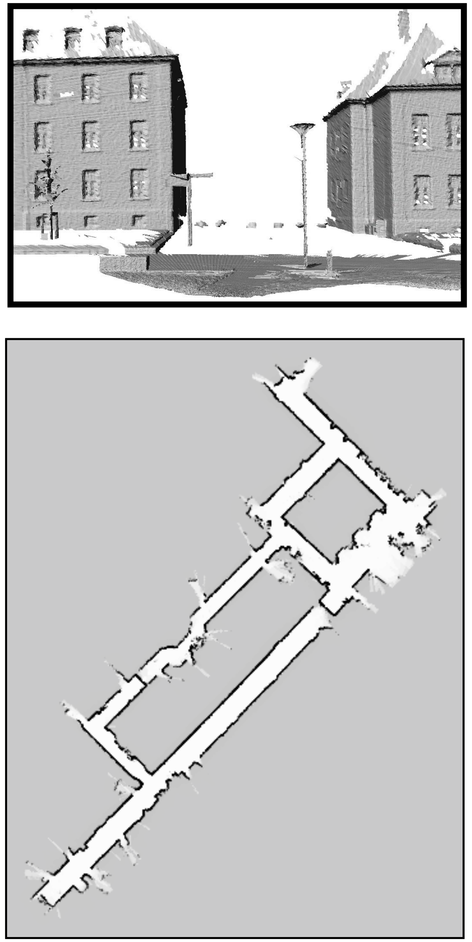
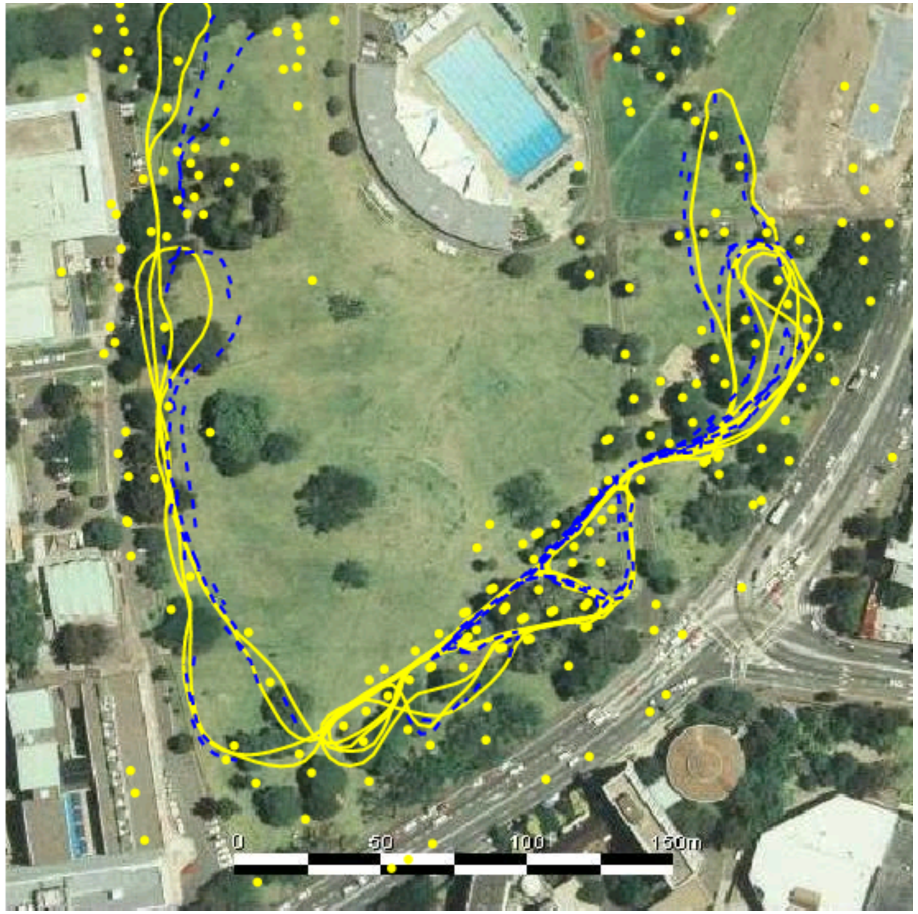


Figure Courtesy: Cyrill Stachniss

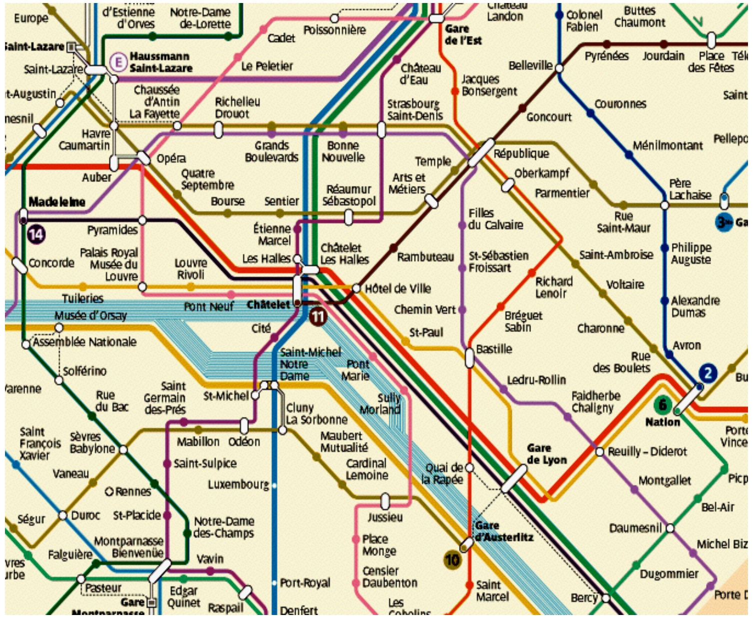
Types of Map in SLAM



Metric Maps



Feature-based Maps



Topological Maps

EKF-SLAM

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

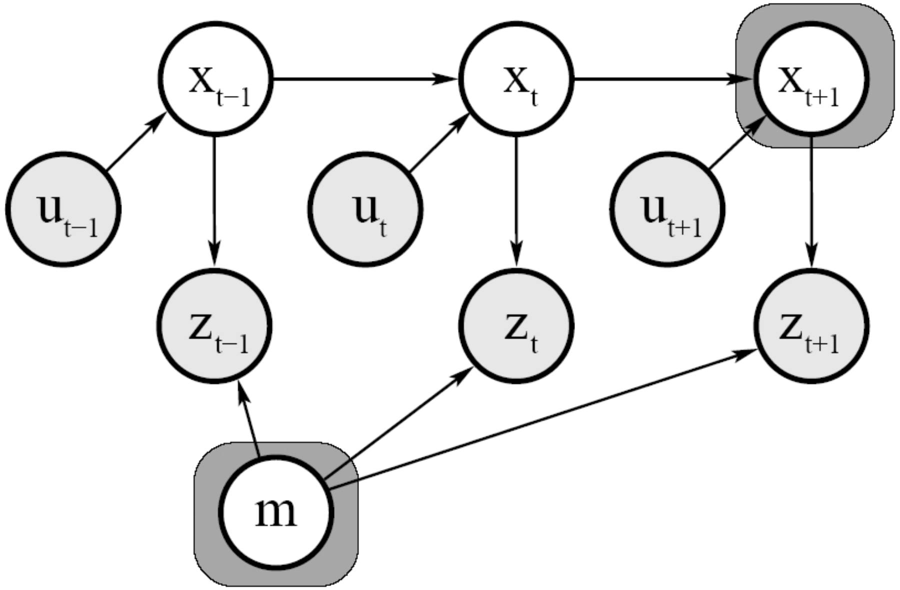


Figure Courtesy: Cyrill Stachniss

Recall EKF

$$\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\textbf{:}$$

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

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

EKF-SLAM

- Estimate pose of the robot and the landmark locations (map).
- Assumption: Known Correspondence
- State space in 2D plane

$$x_t = (\underbrace{x, y, \theta}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}}_{\text{landmark 1}}, \ldots, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}})^T$$

EKF-SLAM state representation

- Map with “n” landmarks => (3 + 2n) dimensional gaussian

$$\underbrace{\begin{pmatrix} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} & \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} & \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta} & \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta} & \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta} & \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta} & \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{pmatrix}}_{\Sigma}$$

Figure Courtesy: Cyrill Stachniss

EKF-SLAM: Compact state representation

$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \dots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \dots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \dots & \Sigma_{m_n m_n} \end{pmatrix}}_{\Sigma}$$

$$\Downarrow$$

$$\underbrace{\begin{pmatrix} x \\ m \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix}}_{\Sigma}$$

Figure Courtesy: Cyrill Stachniss

EKF-SLAM example

- Robot moves in 2D plane
- Number of landmarks are known “n”
- Dimension of state space (3+2n)
- Velocity motion model
- Range and bearing sensor
- Robot observes point landmarks
- Data association is known

Initialization

- Robot starts at (0,0) that becomes the global frame
- All landmarks are unknown

$$\mu_0 = \begin{pmatrix} 0 & 0 & 0 & \ldots & 0 \end{pmatrix}^T$$

$$\Sigma_0 = \begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \infty \end{pmatrix}$$

Motion Update: Prediction

- Update the position of robot

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\ \sin(\theta + \omega_t \Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\ \cos(\theta + \omega_t \Delta t) \\ \omega_t\ \Delta t \end{pmatrix}}_{g_{x,y,\theta}(u_t,(x,y,\theta)^T)}$$

- How to convert this to (3 + 2n) dimension state space ?

Recall EKF

$$\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\textbf{:}$$

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

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

- How to convert motion model to (3 + 2n) dimension state space ?

Motion update

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \underbrace{\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\ \sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\ \cos(\theta+\omega_t\Delta t) \\ \omega_t\ \Delta t \end{pmatrix}}_{g_{x,y,\theta}(u_t,(x,y,\theta)^T)}$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \\ \vdots \end{pmatrix} = \underbrace{\begin{pmatrix} x \\ y \\ \theta \\ \vdots \end{pmatrix} + \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0\ldots0 \\ 0 & 1 & 0 & 0\ldots0 \\ 0 & 0 & 1 & \underbrace{0\ldots0}_{2N cols} \end{pmatrix}^T}_{F_x^T} \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\ \sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\ \cos(\theta+\omega_t\Delta t) \\ \omega_t\ \Delta t \end{pmatrix}}_{g(u_t,x_t)}$$

EKF-SLAM

$$\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\textbf{:}$$

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

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

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

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

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

$$return\ \mu_t, \Sigma_t$$

Prediction: Update Covariance

- Motion model only affects robots position
- Landmarks remains unchanged

Jacobian of the motion (3x3)

$$G_t = \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix}$$

Identity (2N x 2N)

Jacobian

$$\begin{aligned}
G_t^x &= \frac{\partial}{\partial(x,y,\theta)^T}\left[\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\ \sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\ \cos(\theta+\omega_t\Delta t) \\ \omega_t\ \Delta t \end{pmatrix}\right] \\
&= I + \frac{\partial}{\partial(x,y,\theta)^T}\begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\ \sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\ \cos(\theta+\omega_t\Delta t) \\ \omega_t\ \Delta t \end{pmatrix} \\
&= I + \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}$$

Predicted Covariance

$$
\begin{aligned}
\bar{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + R_t \\
&= \begin{pmatrix} G_t^x & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{pmatrix} \begin{pmatrix} (G_t^x)^T & 0 \\ 0 & I \end{pmatrix} + R_t \\
&= \begin{pmatrix} G_t^x \Sigma_{xx} (G_t^x)^T & G_t^x \Sigma_{xm} \\ (G_t^x \Sigma_{xm})^T & \Sigma_{mm} \end{pmatrix} + R_t
\end{aligned}
$$

EKF-SLAM Prediction step

$$\textbf{EKF\_SLAM\_Prediction}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, R_t)\textbf{:}$$

2: $$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

3: $$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

4: $$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

5: $$\bar{\Sigma}_t = G_t \, \Sigma_{t-1} \, G_t^T + \underbrace{F_x^T \, R_t^x \, F_x}_{R_t}$$

EKF-SLAM

$$\textbf{Extended\_Kalman\_filter}(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\textbf{:}$$

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

$$\cancel{\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t}$$ **Apply & DONE**

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

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

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

$$return\ \mu_t, \Sigma_t$$

EKF-SLAM: Correction step

- Landmark observation $z_t^i = (r_t^i, \phi_t^i)^T$

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \ \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}$$

observed location of landmark j	estimated robot's location	relative measurement

- If the landmark is observed for the first time you will have to initialize the landmark.

Correction step

- Compute expected observation based on current estimate

$$
\begin{aligned}
\delta &= \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix} \\
q &= \delta^T \delta \\
\hat{z}_t^i &= \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix} \\
&= h(\bar{\mu}_t)
\end{aligned}
$$

Measurement Jacobian

$$
\begin{aligned}
{}^{\text{low}}H_t^i &= \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} \\
&= \begin{pmatrix} \frac{\partial \sqrt{q}}{\partial x} & \frac{\partial \sqrt{q}}{\partial y} & \dots \\ \frac{\partial \text{atan2}(\dots)}{\partial x} & \frac{\partial \text{atan2}(\dots)}{\partial y} & \dots \end{pmatrix}
\end{aligned}
$$

- Note that this Jacobian is low-dimensional and only depends on

$$(x, y, \theta, m_{j,x}, m_{j,y})$$

Measurement Jacobian

$$
\begin{aligned}
{}^{\text{low}}H_t^i &= \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} \\
&= \frac{1}{q}\begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix}
\end{aligned}
$$

- We can map this to the full state space

$$
H_t^i = {}^{\text{low}}H_t^i \, F_{x,j}
$$

$$
F_{x,j} = \begin{pmatrix}
1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\
0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\
0 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & 0 & 1 & \underbrace{0 \cdots 0}_{2N-2j}
\end{pmatrix}
$$

EKF-SLAM correction

EKF_SLAM_Correction

$$Q_t = \begin{pmatrix} {\sigma_r}^2 & 0 \\ 0 & {\sigma_\phi}^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i, \phi_t^i)^T$ *do*

$$j = c_t^i$$

if landmark j *never seen before*

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \ \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix}$$

EKF-SLAM correction

$$F_{x,j} = \left(\begin{array}{cccccccc} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & 0 & 1 & \underbrace{0 \cdots 0}_{2N-2j} \end{array} \right)$$

$$H_t^i = \frac{1}{q} \left(\begin{array}{ccccc} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x \end{array} \right) F_{x,j}$$

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

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i \; H_t^i) \; \bar{\Sigma}_t$$

endfor

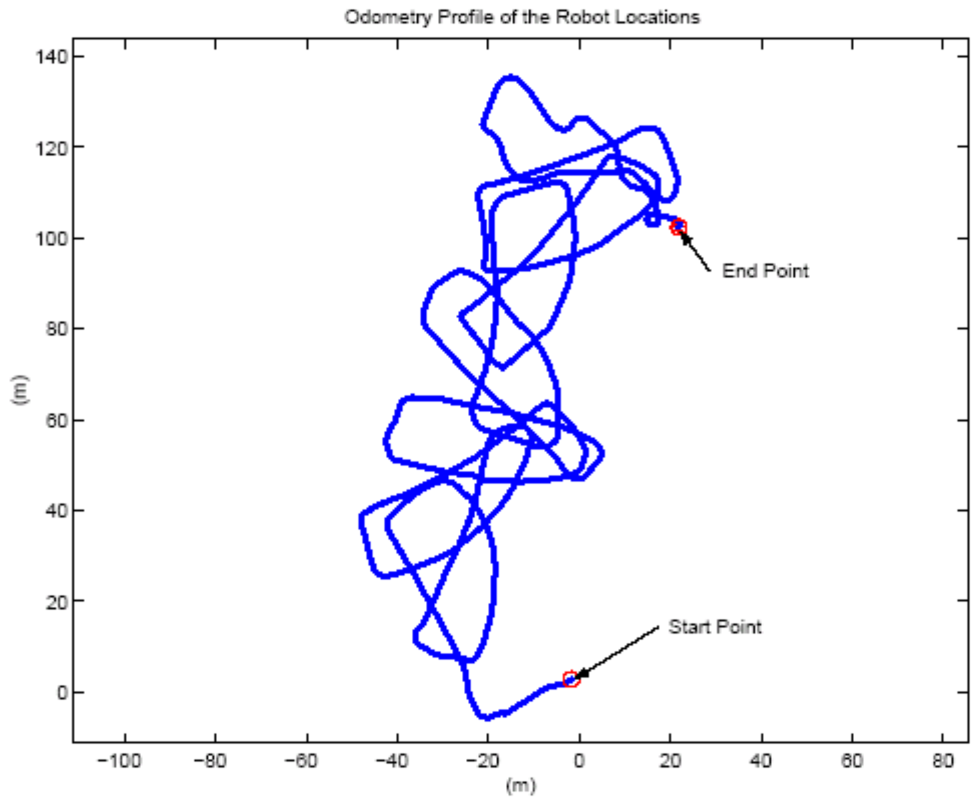
$$\mu_t = \bar{\mu}_t$$

$$\Sigma_t = \bar{\Sigma}_t$$

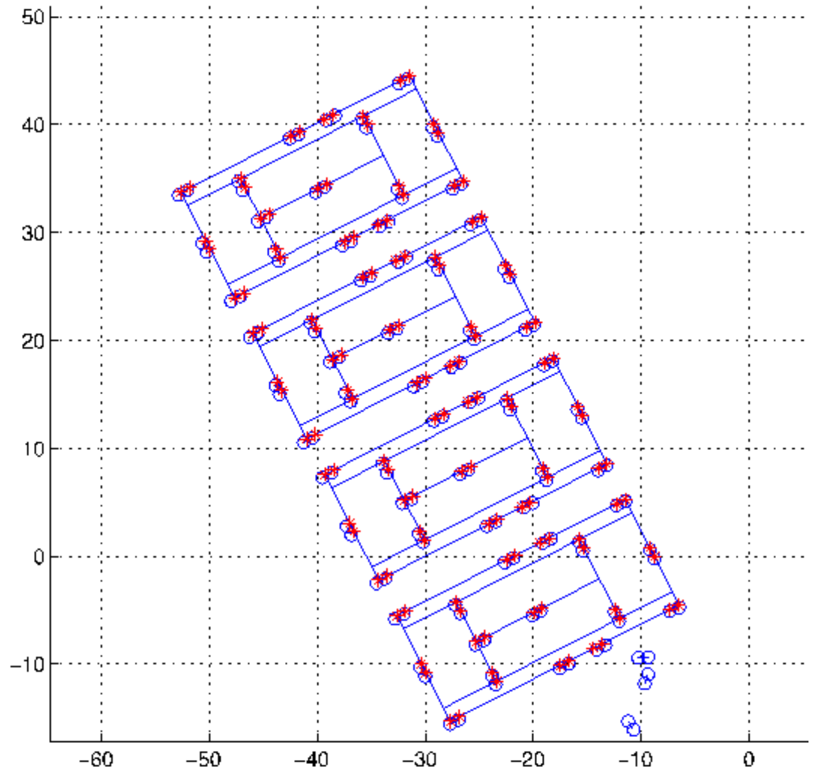
return μ_t, Σ_t

Example: MIT tennis court dataset

Example: MIT tennis court dataset

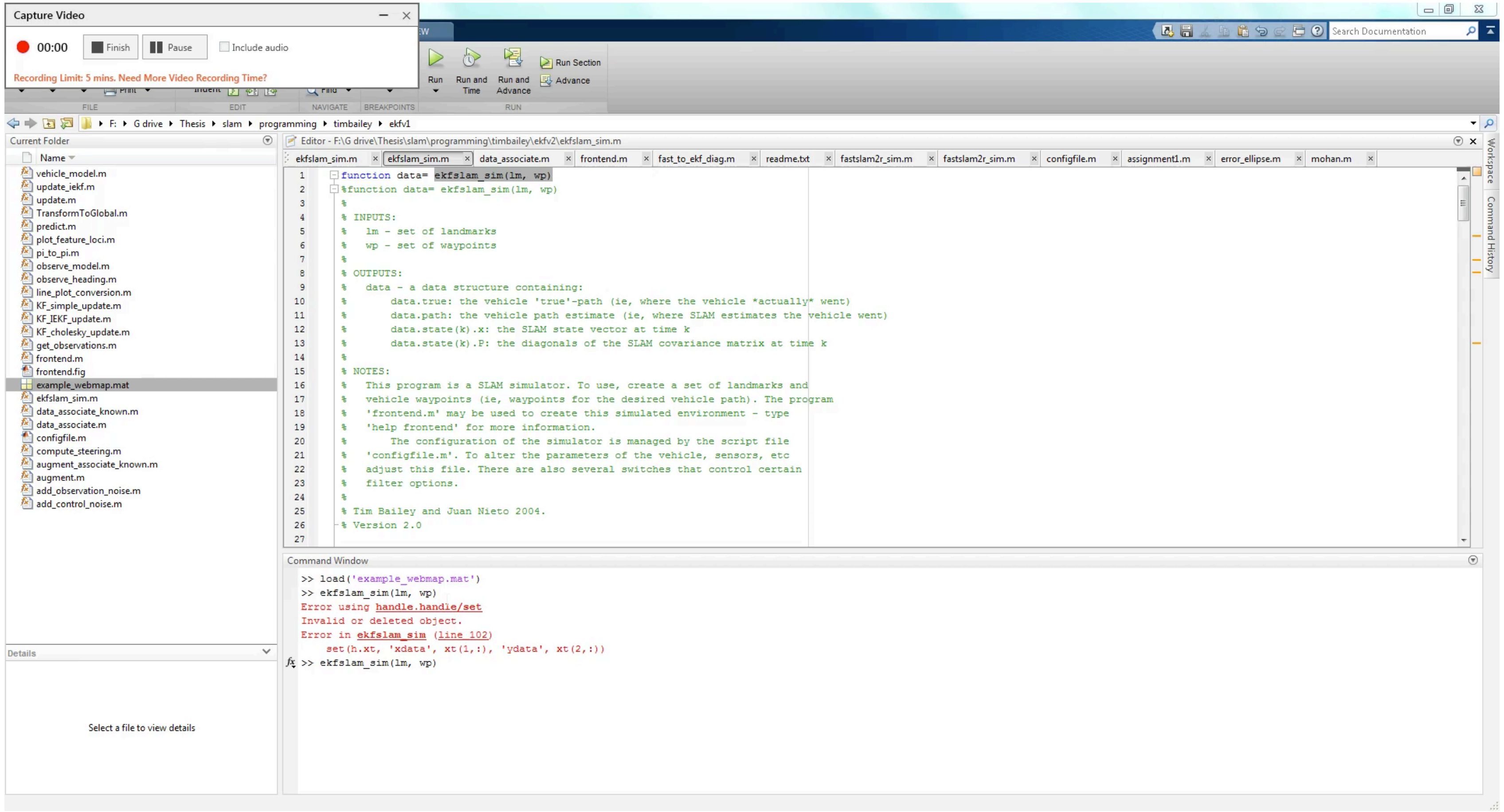


odometry



estimated trajectory

EKF-SLAM


Capture Video
00:00
Finish
Pause
Include audio
Recording Limit: 5 mins. Need More Video Recording Time?
Run
Run and Time
Run and Advance
Run Section
Advance
Search Documentation
F: ▸ G drive ▸ Thesis ▸ slam ▸ programming ▸ timbailey ▸ ekfv1
Current Folder
Name
vehicle_model.m
update_iekf.m
update.m
TransformToGlobal.m
predict.m
plot_feature_loci.m
pi_to_pi.m
observe_model.m
observe_heading.m
line_plot_conversion.m
KF_simple_update.m
KF_IEKF_update.m
KF_cholesky_update.m
get_observations.m
frontend.m
frontend.fig
example_webmap.mat
ekfslam_sim.m
data_associate_known.m
data_associate.m
configfile.m
compute_steering.m
augment_associate_known.m
augment.m
add_observation_noise.m
add_control_noise.m
Details
Select a file to view details
Editor - F:\G drive\Thesis\slam\programming\timbailey\ekfv2\ekfslam_sim.m
ekfslam_sim.m
ekfslam_sim.m
data_associate.m
frontend.m
fast_to_ekf_diag.m
readme.txt
fastslam2r_sim.m
fastslam2r_sim.m
configfile.m
assignment1.m
error_ellipse.m
mohan.m
function data= ekfslam_sim(lm, wp)
%function data= ekfslam_sim(lm, wp)
%
% INPUTS:
% lm - set of landmarks
% wp - set of waypoints
%
% OUTPUTS:
% data - a data structure containing:
% data.true: the vehicle 'true'-path (ie, where the vehicle *actually* went)
% data.path: the vehicle path estimate (ie, where SLAM estimates the vehicle went)
% data.state(k).x: the SLAM state vector at time k
% data.state(k).P: the diagonals of the SLAM covariance matrix at time k
%
% NOTES:
% This program is a SLAM simulator. To use, create a set of landmarks and
% vehicle waypoints (ie, waypoints for the desired vehicle path). The program
% 'frontend.m' may be used to create this simulated environment - type
% 'help frontend' for more information.
% The configuration of the simulator is managed by the script file
% 'configfile.m'. To alter the parameters of the vehicle, sensors, etc
% adjust this file. There are also several switches that control certain
% filter options.
%
% Tim Bailey and Juan Nieto 2004.
% Version 2.0
Command Window
>> load('example_webmap.mat')
>> ekfslam_sim(lm, wp)
Error using handle.handle/set
Invalid or deleted object.
Error in ekfslam_sim (line 102)
set(h.xt, 'xdata', xt(1,:), 'ydata', xt(2,:))
>> ekfslam_sim(lm, wp)

EKF-SLAM: Unknown Correspondence

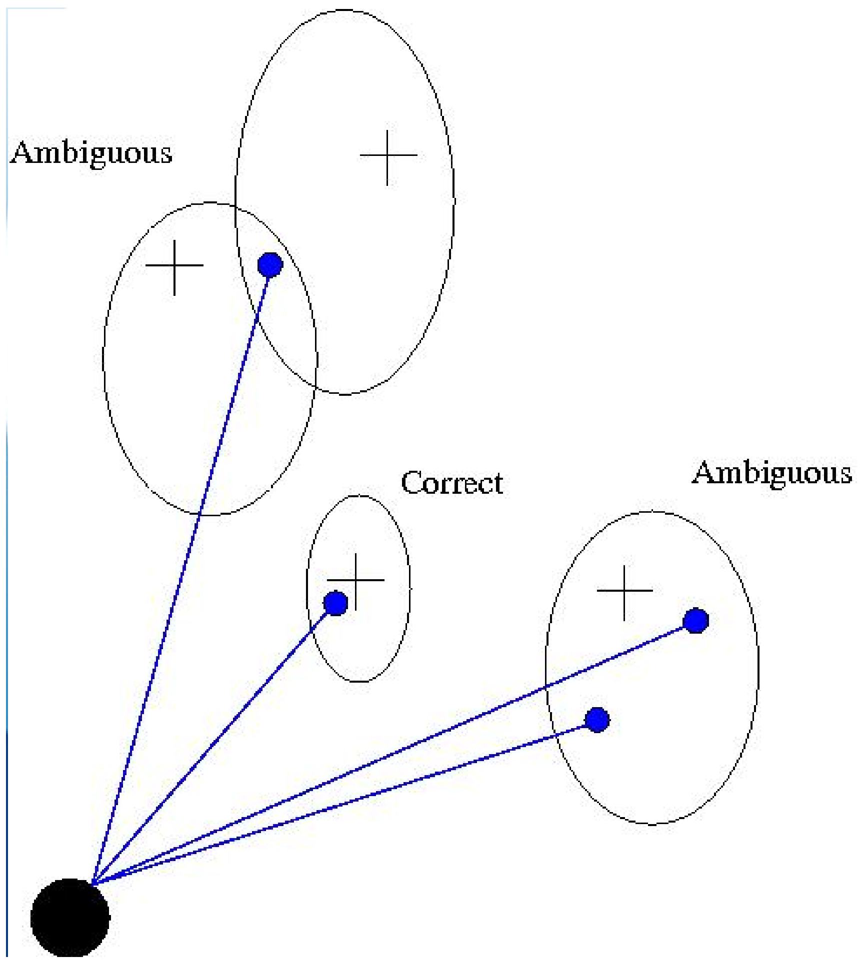


Image Credit: Jack Collier

Data Association: Maximum Likelihood / Mahalanobis Distance

- Maximum Likelihood (ML) data association !
- Find which landmark is most likely to give this measurement if the robot pose was correct.

$$c = \frac{1}{(2\pi)^{k/2}|S|^{1/2}} \exp\left(-\frac{1}{2}(z-\hat{z})^T S^{-1}(z-\hat{z}))\right)$$

$$c' = (z-\hat{z})^T S^{-1}(z-\hat{z})$$

$$(H_t\ \bar{\Sigma}_t\ H_t^T + Q_t)$$

ML Data Association
Z1 → L1
Z2 → L3
Z3 → L4

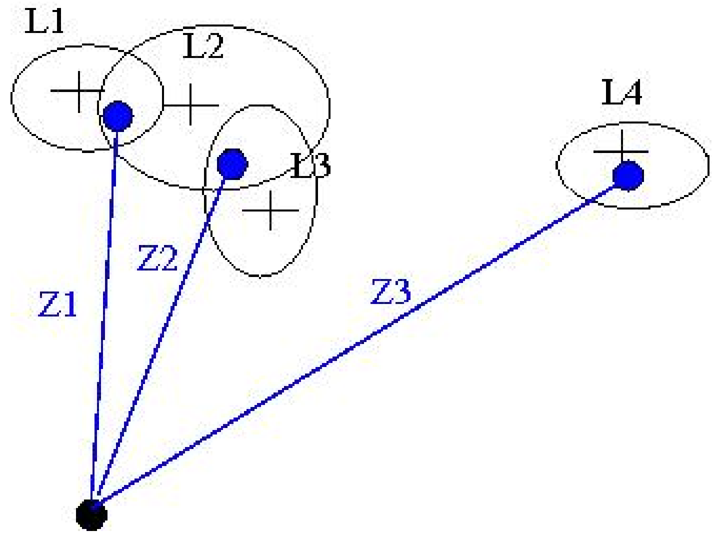
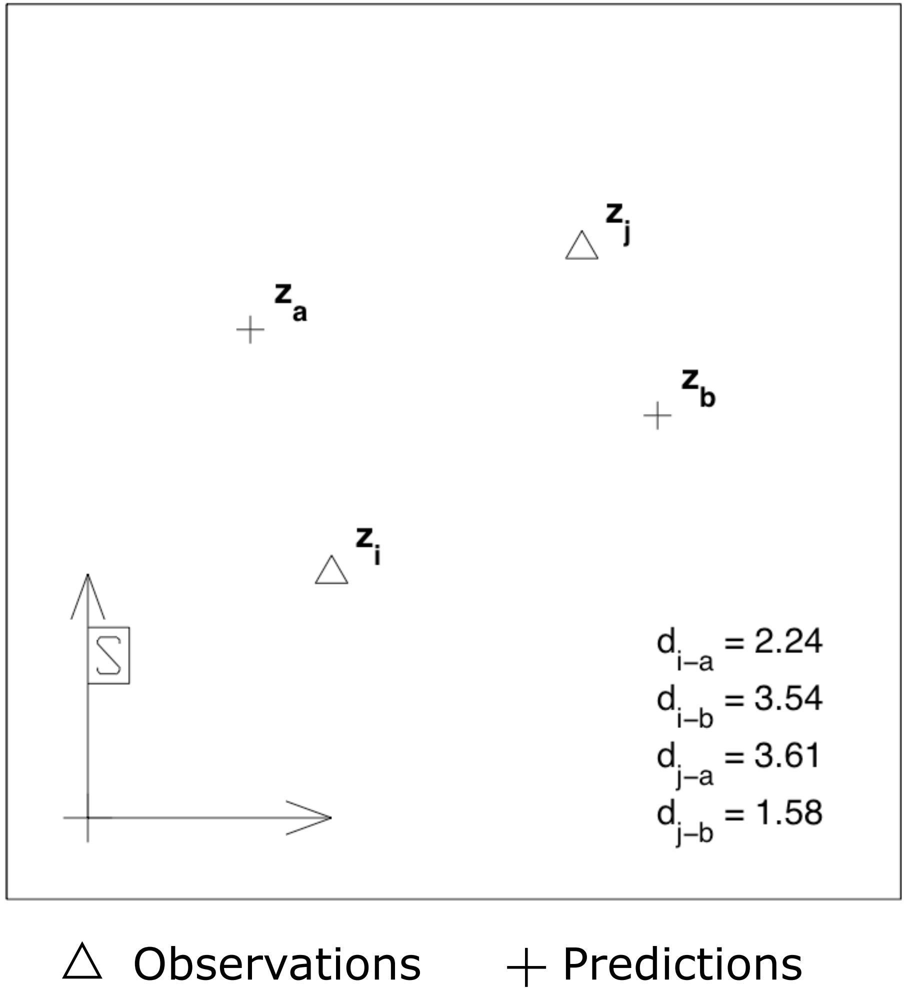


Image Credit: Jack Collier

Why MLE / Mahalonobis distance ?



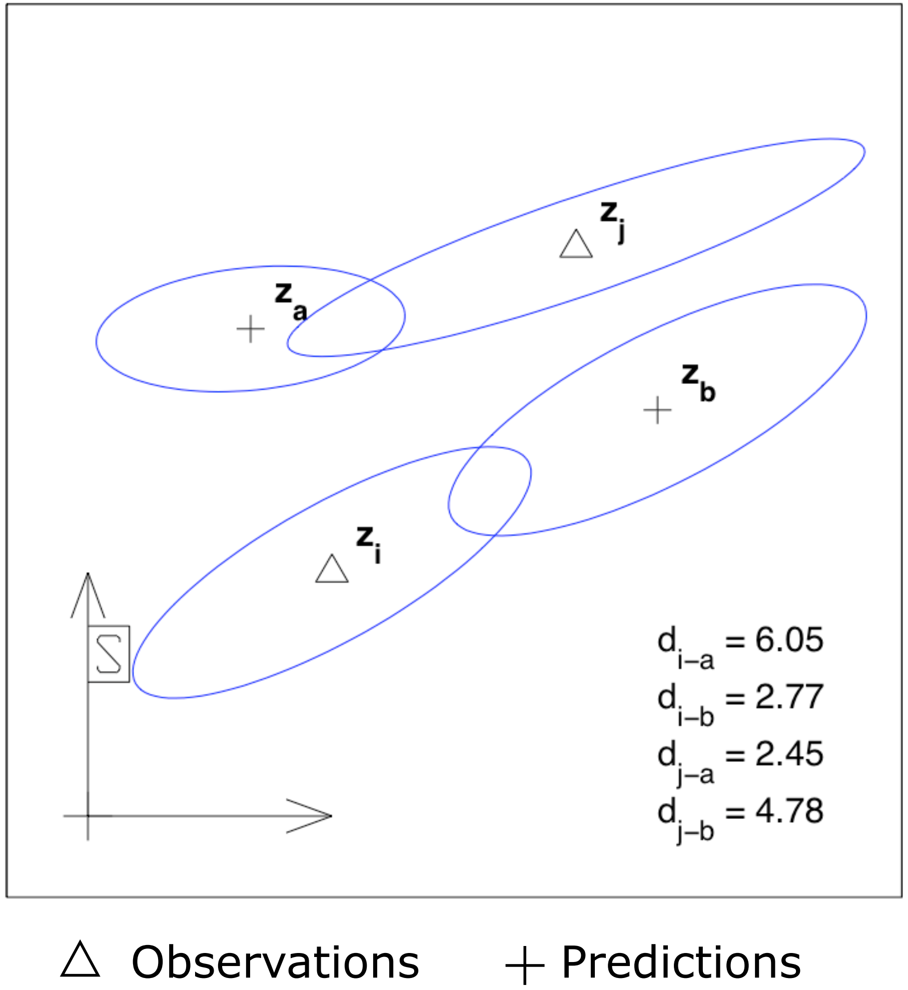


Image Credit: Cyrill Stachniss

EKF Correction: Unknown Correspondence

for all observed features $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ do

$$\begin{pmatrix} \bar{\mu}_{N_t+1,x} \\ \bar{\mu}_{N_t+1,y} \\ \bar{\mu}_{N_t+1,s} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \\ s_t^i \end{pmatrix} + r_t^i \begin{pmatrix} \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \\ 0 \end{pmatrix}$$

for $k = 1$ to N_t+1 do

$$\delta_k = \begin{pmatrix} \delta_{k,x} \\ \delta_{k,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{k,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{k,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q_k = \delta_k^T \delta_k$$

$$\hat{z}_t^k = \begin{pmatrix} \sqrt{q_k} \\ \operatorname{atan2}(\delta_{k,y}, \delta_{k,x}) - \bar{\mu}_{t,\theta} \\ \bar{\mu}_{k,s} \end{pmatrix}$$

$$F_{x,k} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 0 & 1 & 0\cdots0 \end{pmatrix}$$

$$H_t^k = \frac{1}{q_k} \begin{pmatrix} \sqrt{q}_k \delta_{k,x} & -\sqrt{q}_k \delta_{k,y} & 0 & -\sqrt{q}_k \delta_{k,x} & \sqrt{q}_k \delta_{k,y} & 0 \\ \delta_{k,y} & \delta_{k,x} & -1 & -\delta_{k,y} & -\delta_{k,x} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} F_{x,k}$$

$$\Psi_k = H_t^k \; \bar{\Sigma}_t \; [H_t^k]^T + Q_t$$

$$\pi_k = (z_t^i - \hat{z}_t^k)^T \; \Psi_k^{-1} \; (z_t^i - \hat{z}_t^k)$$

endfor

$$\pi_{N_t+1} = \alpha$$

$$j(i) = \underset{k}{\operatorname{argmin}} \;\; \pi_k$$

$$N_t = \max\{N_t, j(i)\}$$

$$K_t^i = \bar{\Sigma}_t \; [H_t^{j(i)}]^T \Psi_{j(i)}^{-1}$$

endfor

Data Association: JCBB

- Joint Compatibility Branch and Bound

Potential Matches
Z1 → L1 or L2
Z2 → L2 or L3
Z3 → L4

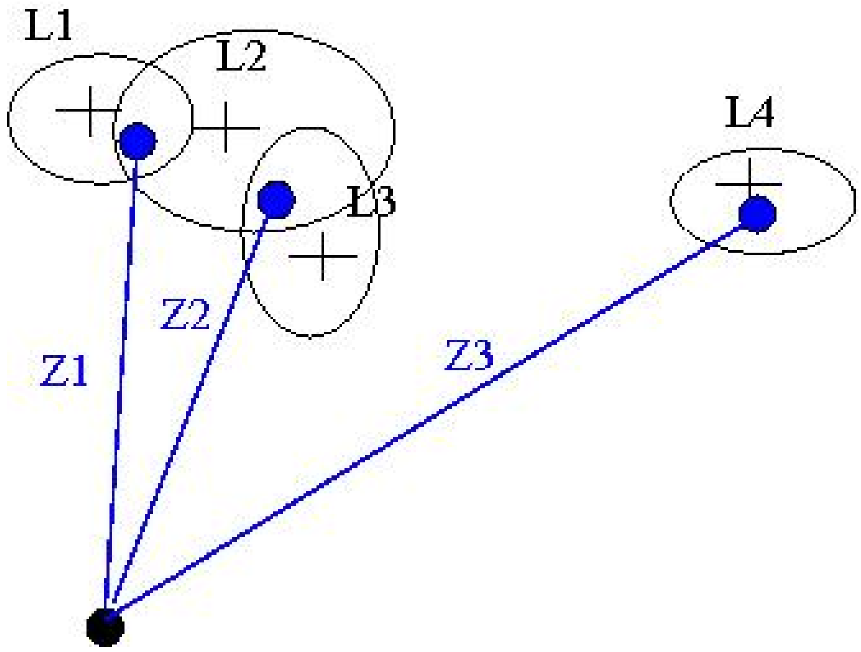


Image Credit: Jack Collier

Data Association: JCBB

- Joint Compatibility Branch and Bound
- Look at joint ML !

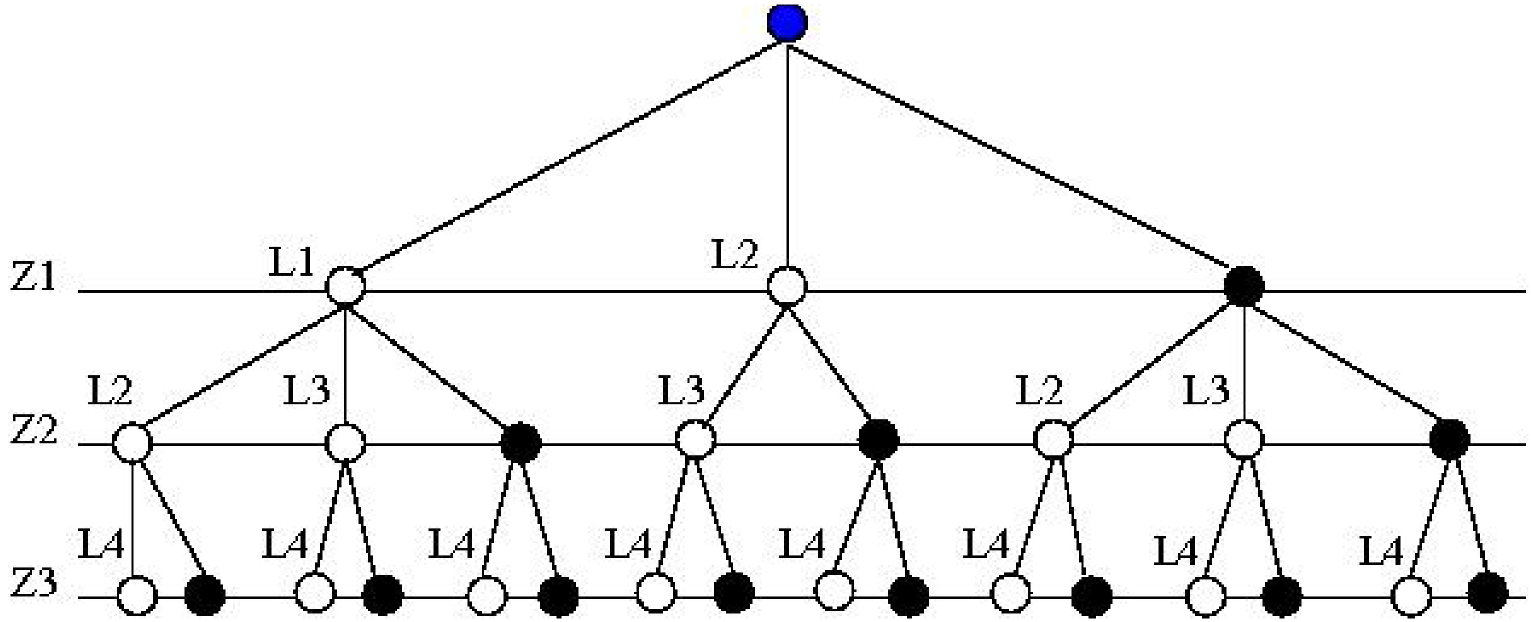
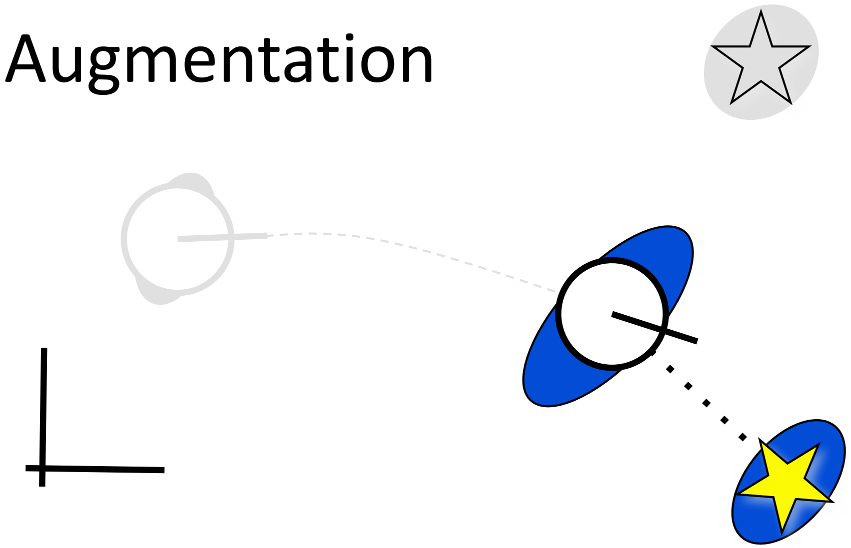


Image Credit: Jack Collier

Integrating New Landmark

- State Augmentation



State augmented by

$$\mathbf{m}_{n+1} = g(\mathbf{x}_R, \mathbf{z}_j)$$

$$C_{M_{n+1}} = G_R \, C_R \, G_R^T$$

Cross-covariances:

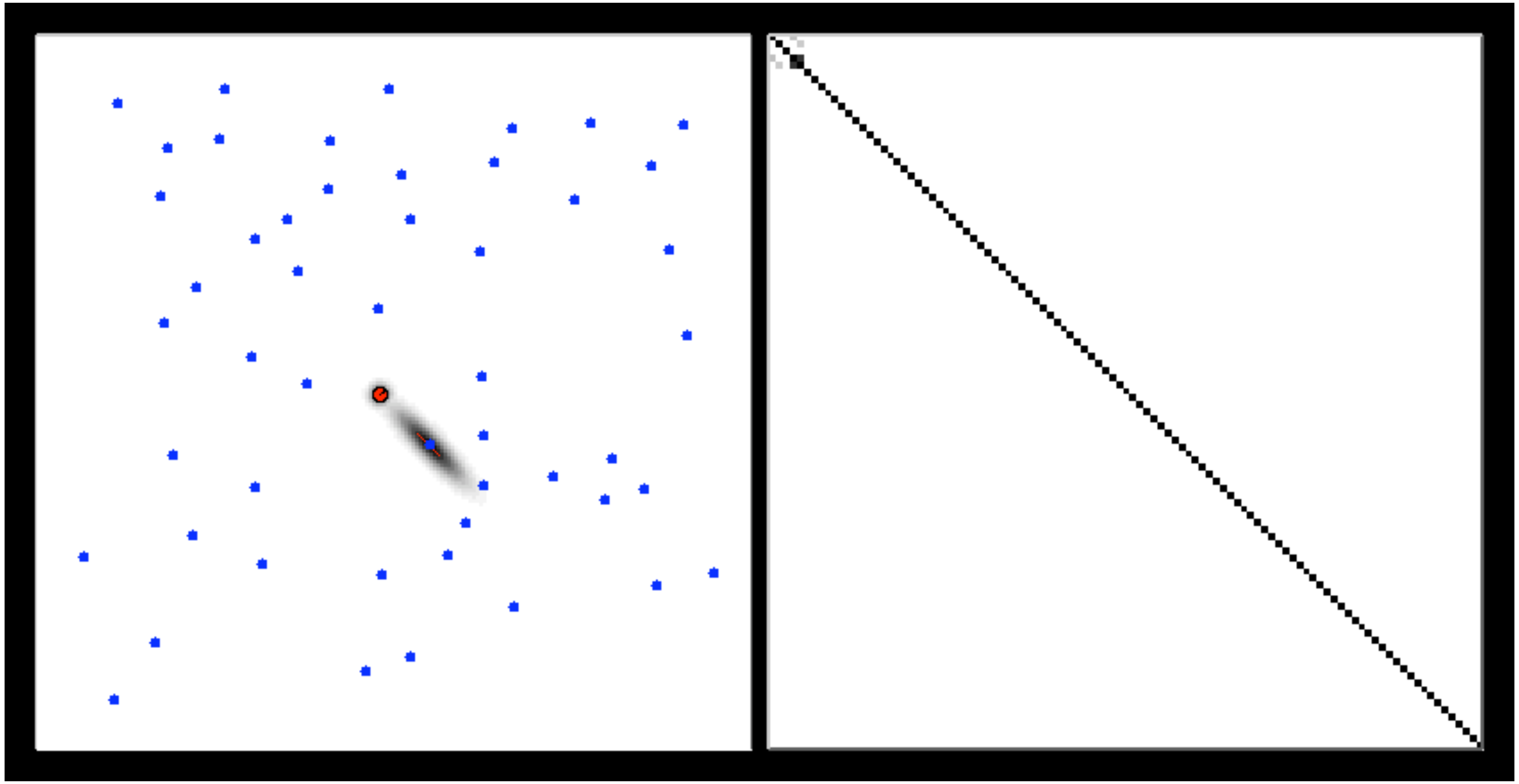
$$C_{M_{n+1}M_i} = G_R \, C_{RM_i}$$

$$C_{M_{n+1}R} = G_R \, C_R$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \\ \mathbf{m}_{n+1} \end{bmatrix}_k \quad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} & C_{RM_{n+1}} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} & C_{M_1M_{n+1}} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} & C_{M_2M_{n+1}} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} & C_{M_nM_{n+1}} \\ C_{M_{n+1}R} & C_{M_{n+1}M_1} & C_{M_{n+1}M_2} & \cdots & C_{M_{n+1}M_n} & C_{M_{n+1}} \end{bmatrix}_k$$

Slide Credit: Wolfram Burgard

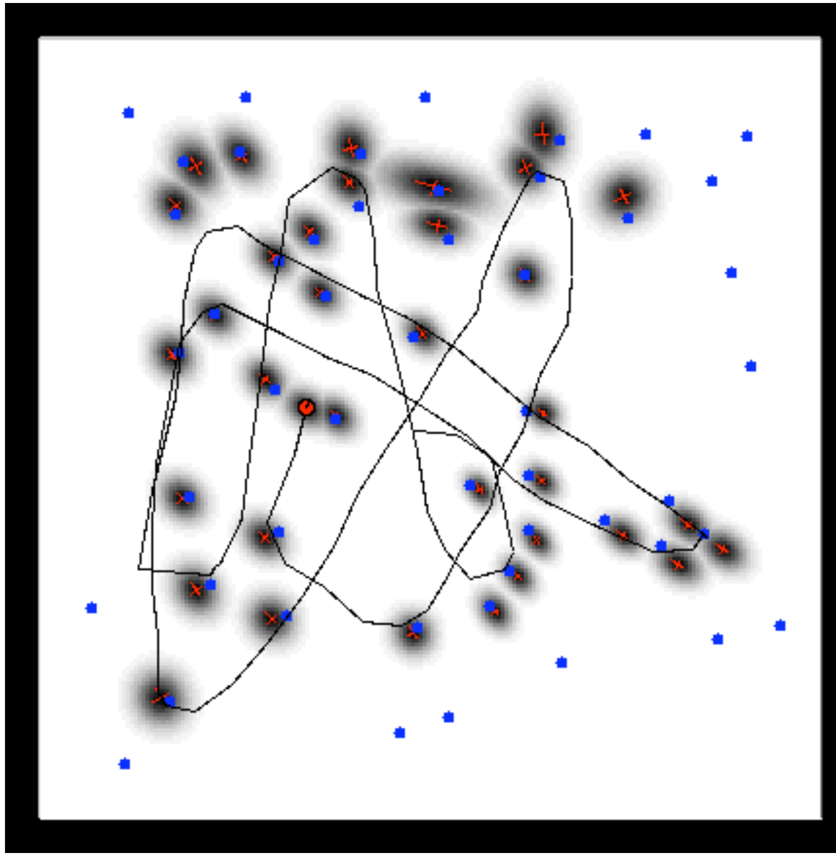
EKF-SLAM: Correlation Analysis



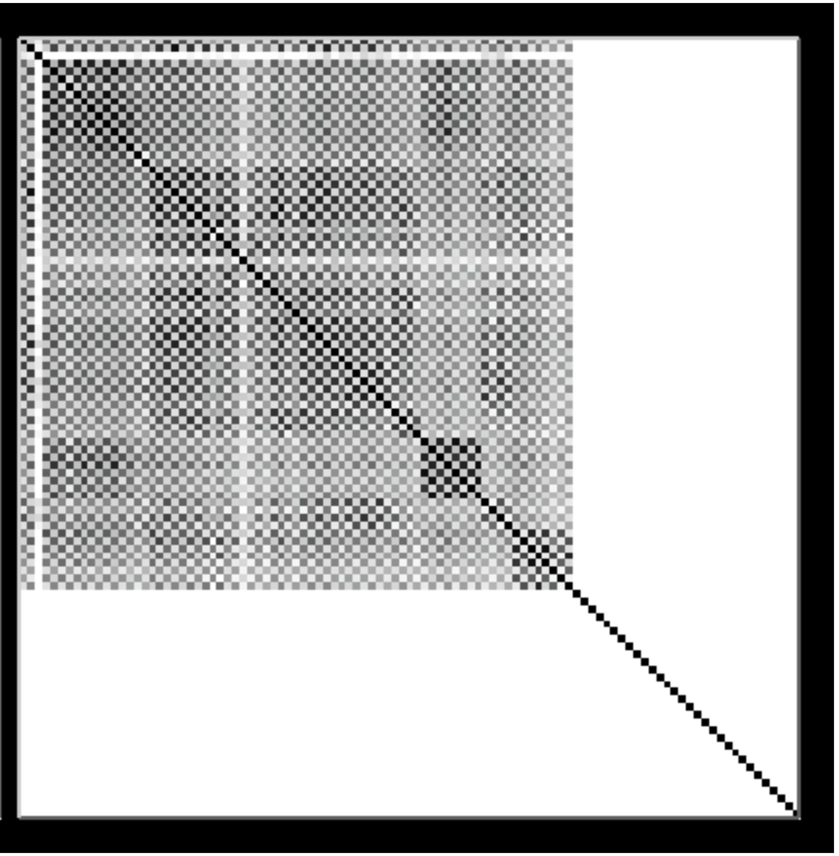
Map Correlation matrix

Slide Credit: Wolfram Burgard

EKF-SLAM: Correlation Analysis



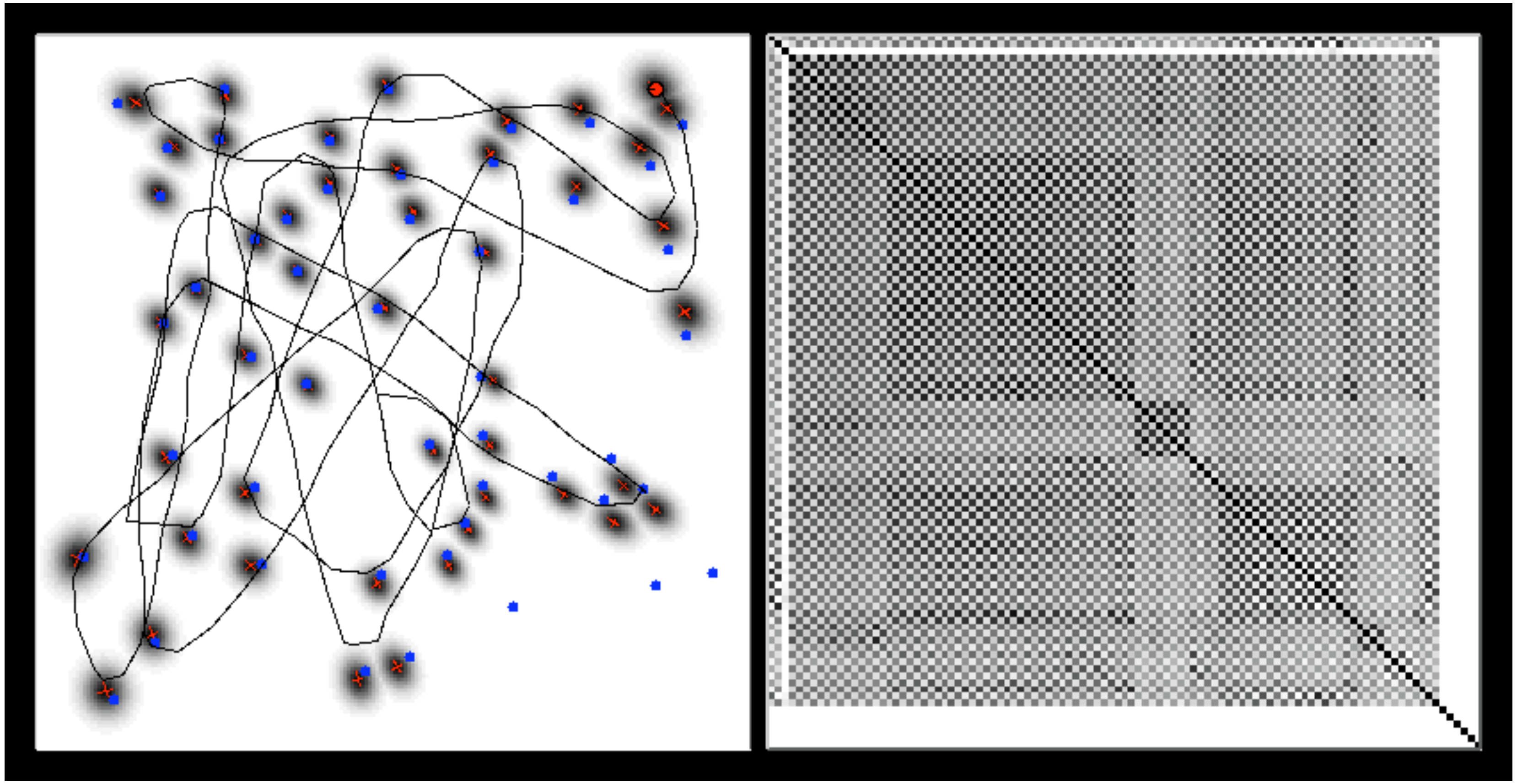
Map



Correlation matrix

Slide Credit: Wolfram Burgard

EKF-SLAM: Correlation Analysis



Map Correlation matrix

Slide Credit: Wolfram Burgard

Correlation Matters

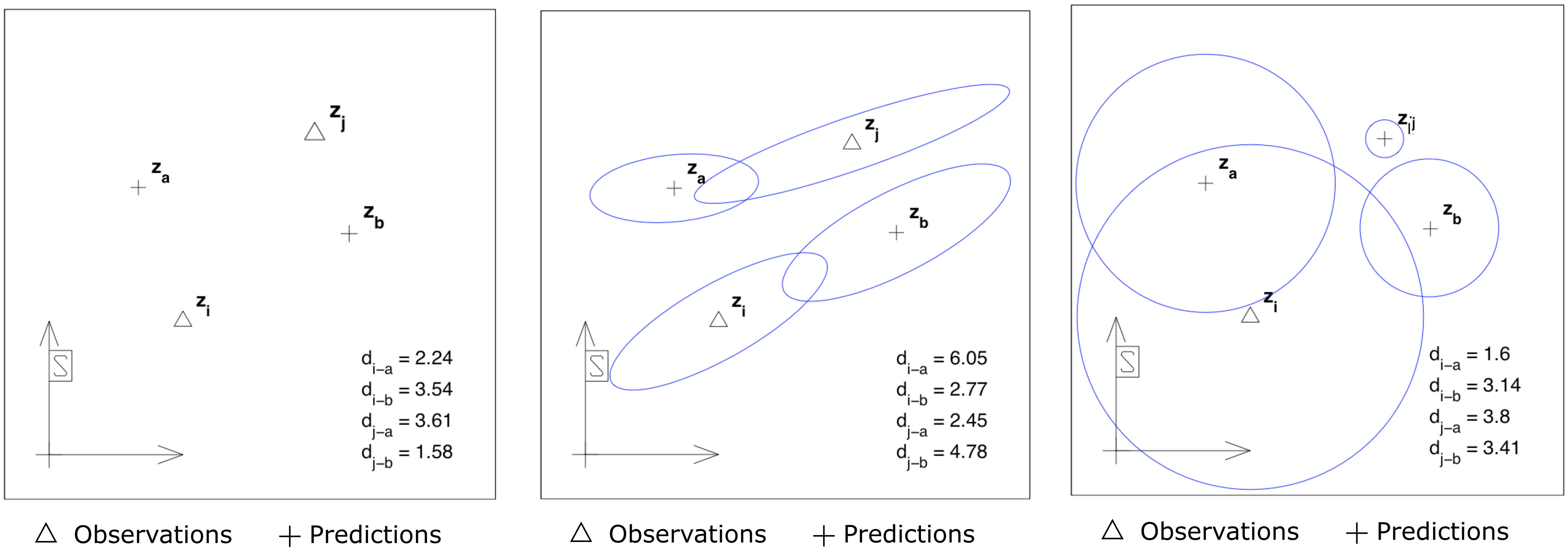
- What if we **neglected** cross-correlations?

$$C_k = \begin{bmatrix} C_R & 0 & \cdots & 0 \\ 0 & C_{M_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_{M_n} \end{bmatrix}_k$$

$$C_{RM_i} = \mathbf{0}_{3\times 2}$$

$$C_{M_iM_{i+1}} = \mathbf{0}_{2\times 2}$$

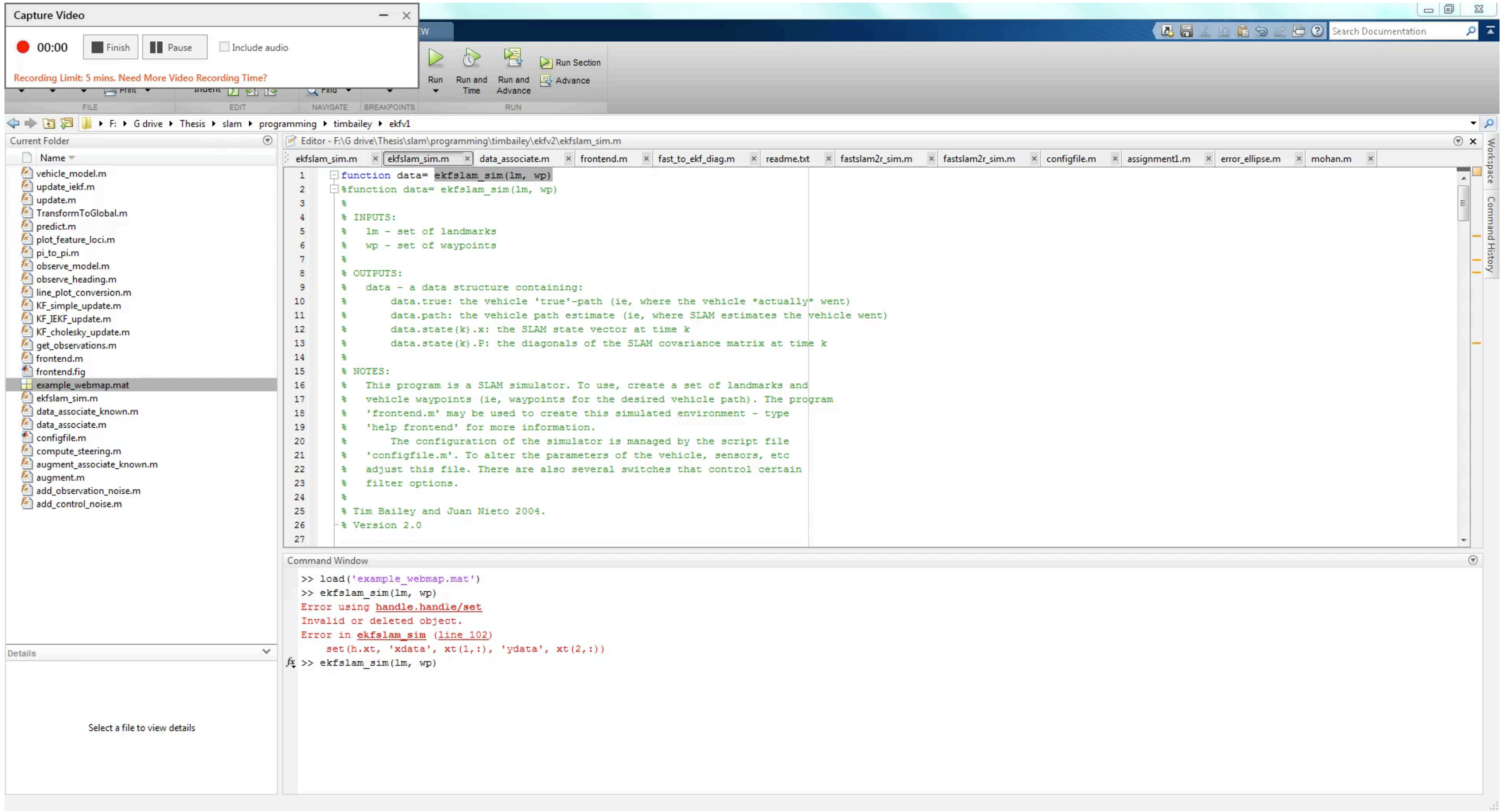
Correlation Matters



SLAM Loop Closures

- Loop Closure is the problem of recognizing an already mapped area, typically after a long exploration path (the robot “closes a loop”)
- Structurally identical to data association
- Uncertainties reduces significantly after the loop closure event

EKF-SLAM


Capture Video
00:00
Finish
Pause
Include audio
Recording Limit: 5 mins. Need More Video Recording Time?
Run
Run and Time
Run and Advance
Run Section
Advance
Search Documentation
F: G drive Thesis slam programming timbailey ekfv1
Current Folder
Name
vehicle_model.m
update_iekf.m
update.m
TransformToGlobal.m
predict.m
plot_feature_loci.m
pi_to_pi.m
observe_model.m
observe_heading.m
line_plot_conversion.m
KF_simple_update.m
KF_IEKF_update.m
KF_cholesky_update.m
get_observations.m
frontend.m
frontend.fig
example_webmap.mat
ekfslam_sim.m
data_associate_known.m
data_associate.m
configfile.m
compute_steering.m
augment_associate_known.m
augment.m
add_observation_noise.m
add_control_noise.m
Details
Select a file to view details
Editor - F:\G drive\Thesis\slam\programming\timbailey\ekfv2\ekfslam_sim.m
function data= ekfslam_sim(lm, wp)
%function data= ekfslam_sim(lm, wp)
% INPUTS:
% lm - set of landmarks
% wp - set of waypoints
% OUTPUTS:
% data - a data structure containing:
% data.true: the vehicle 'true'-path (ie, where the vehicle *actually* went)
% data.path: the vehicle path estimate (ie, where SLAM estimates the vehicle went)
% data.state(k).x: the SLAM state vector at time k
% data.state(k).P: the diagonals of the SLAM covariance matrix at time k
% NOTES:
% This program is a SLAM simulator. To use, create a set of landmarks and
% vehicle waypoints (ie, waypoints for the desired vehicle path). The program
% 'frontend.m' may be used to create this simulated environment - type
% 'help frontend' for more information.
% The configuration of the simulator is managed by the script file
% 'configfile.m'. To alter the parameters of the vehicle, sensors, etc
% adjust this file. There are also several switches that control certain
% filter options.
% Tim Bailey and Juan Nieto 2004.
% Version 2.0
Command Window
>> load('example_webmap.mat')
>> ekfslam_sim(lm, wp)
Error using handle.handle/set
Invalid or deleted object.
Error in ekfslam_sim (line 102)
set(h.xt, 'xdata', xt(1,:), 'ydata', xt(2,:))
>> ekfslam_sim(lm, wp)

EKF-SLAM Computational Complexity

- **Cost per step**: quadratic in n, the number of landmarks: $O(n^2)$
- **Total cost** to build a **map** with n landmarks: $O(n^3)$
- **Memory**: $O(n^2)$

Problem: becomes computationally intractable for large maps!

➔ Approaches exist that make EKF-SLAM amortized $O(n)$ / $O(n^2)$ / $O(n^2)$ D&C SLAM *[Paz et al., 2006]*

EKF-SLAM Summary

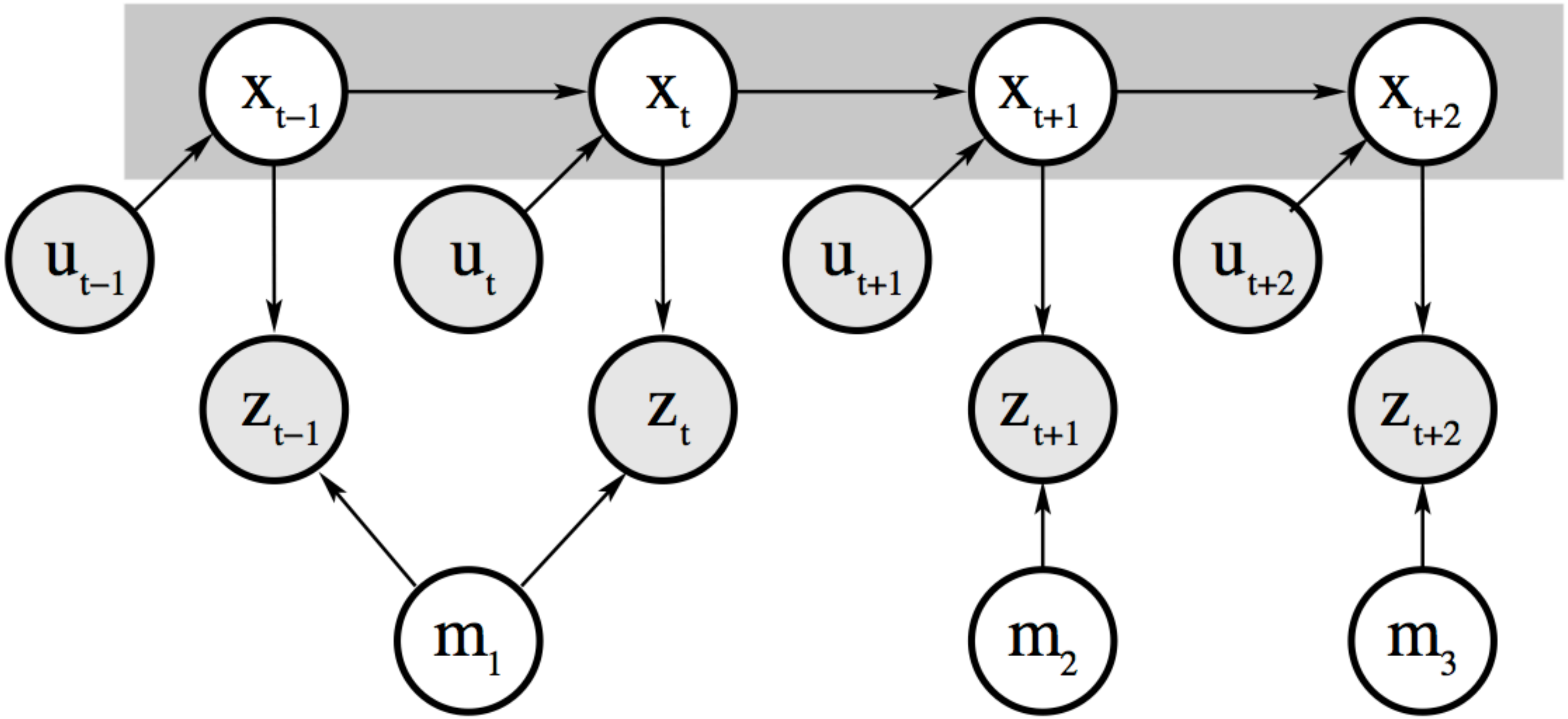
- First practical online implementation of SLAM
- Converges for linear problems
- Diverges for large non-linearities
- Works for small / medium environments
- Computationally intractable for large environments

Particle Filter SLAM

- Particle filter SLAM does not approximate but depends upon the number of particles used. For a high dimensional state space particle filter becomes computationally inefficient !

- Solution FastSLAM !

Landmarks are conditionally independent given the poses



Factorization of SLAM posterior

$$
\begin{aligned}
& p(x_{0:t}, l_{1:M} \mid z_{1:t}, u_{1:t}) \\
& \quad = \; p(x_{0:t} \mid z_{1:t}, u_{1:t}) \, p(l_{1:M} \mid x_{0:t}, z_{1:t}) \\
& \quad = \; \underline{p(x_{0:t} \mid z_{1:t}, u_{1:t})} \prod_{i=1}^{M} \underline{p(l_i \mid x_{0:t}, z_{1:t})}
\end{aligned}
$$

particle filter similar to MCL

2-dimensional EKFs!

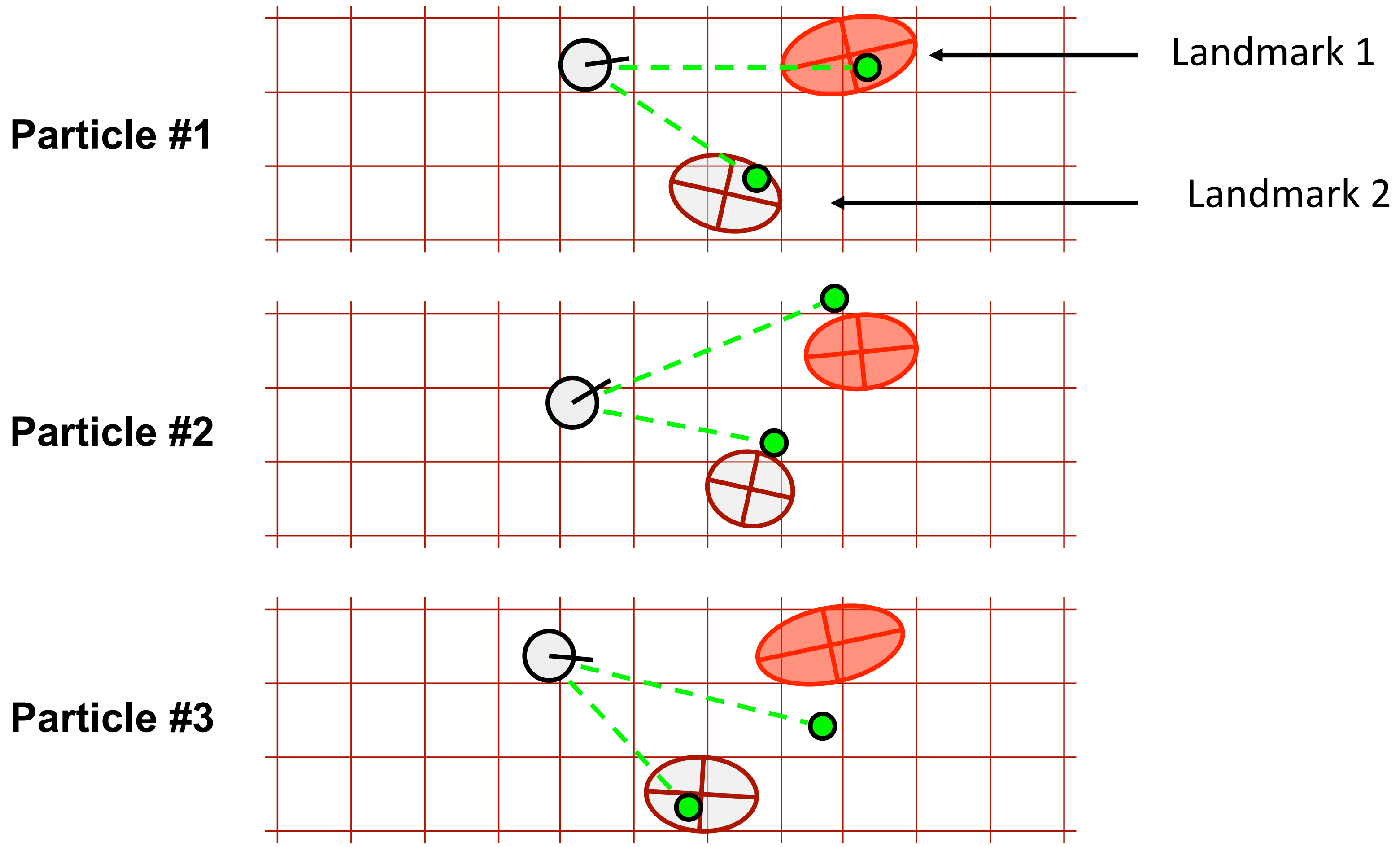
Fast-SLAM

- Robot pose is represented by particles (n).
- Each landmark is represented by a [2x2] EKF
- Each particle has to maintain (M) EKFs

Particle 1	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**
Particle 2	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**
⋮					
Particle N	x, y, θ	**Landmark 1**	**Landmark 2**	**…**	**Landmark M**

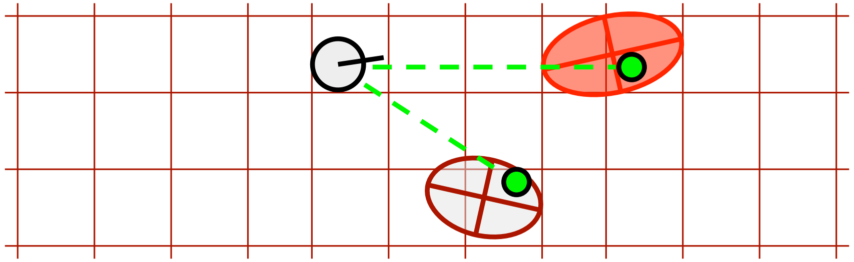
Slide Credit: Cyrill Stachniss

Fast-SLAM



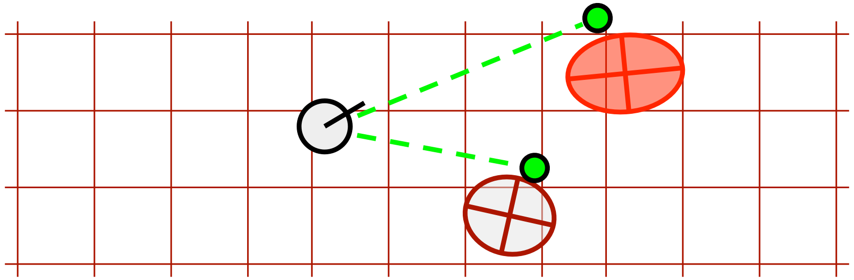
Fast-SLAM (Importance Weight)

Particle #1



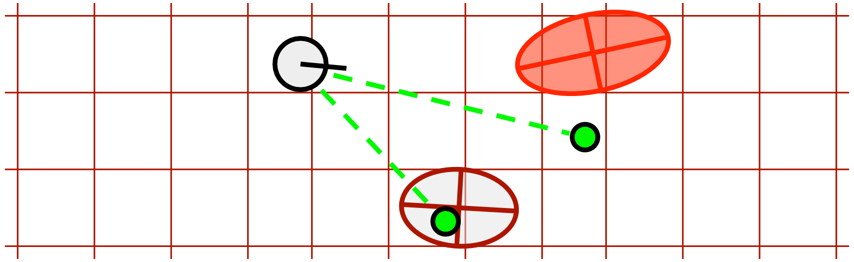
Weight = 0.8

Particle #2



Weight = 0.4

Particle #3



Weight = 0.1

Key Steps in FastSLAM

- Propagate each particle using the motion model

$$x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$$

- Compute particle weight

$$w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \ \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T Q^{-1} \ (z_t - \hat{z}^{[k]})\right\}$$

- EKF update of landmark locations for each particle
- Resample the particles

Recall: Derivation of Particle Filter

- Consider the proposal distribution

$$g(\mathrm{x_t}) = \overline{Bel}\,(\mathrm{x_t})$$

- Consider the target distribution

$$f(\mathrm{x_t}) = Bel\,(\mathrm{x_t})$$

- For any particle $\mathrm{x}_\mathrm{t}^{[m]}$

$$\mathrm{w}_\mathrm{t}^{[m]} = \frac{target\ distribution}{proposal\ distribution}$$

$$= \frac{Bel\,(x_t^{[m]})}{\overline{Bel}(x_t^{[m]})} = \frac{\eta P(z_t|x_t^{[m]})\,\overline{Bel}(x_t^{[m]})}{\overline{Bel}(x_t^{[m]})} = \eta P(z_t|x_t^{[m]})$$

FastSLAM

- See the complete algorithm here:
- “FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association” Sebastian Thrun, Michael Montemerlo, Daphne Koller, Ben Wegbreit, Juan Nieto, and Eduardo Nebot