# ELEC 3210
# Introduction to Mobile Robotics
# Lecture 10

## (Machine Learning and Infomation Processing for Robotics)


Huan YIN

Research Assistant Professor, Dept. of ECE

eehyin@ust.hk





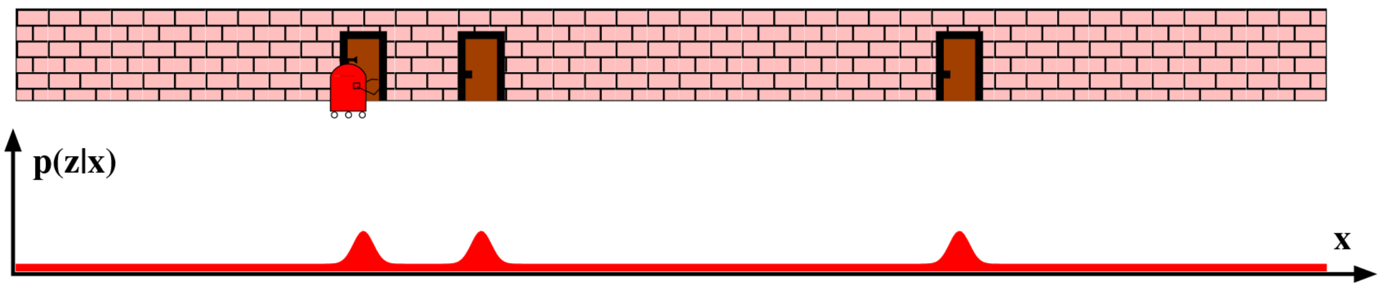
# Recap L7/L8/L9

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

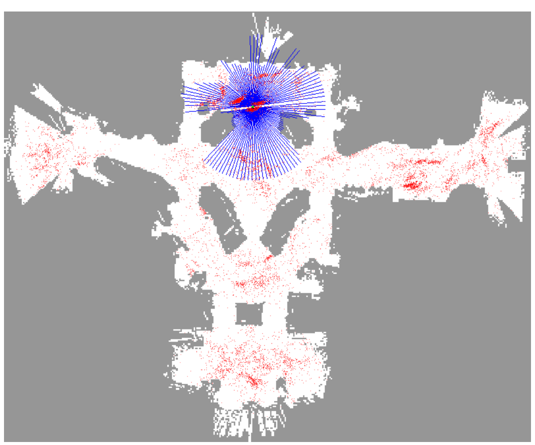
- L7 Bayes Filter
  - recursive filter
- L8 Particle Filter
  - draw particles
- L9 Kalman Filter and EKF
  - Everything stays Gaussian
  - Linearization of EKF

# Examples in L7/L8/L9

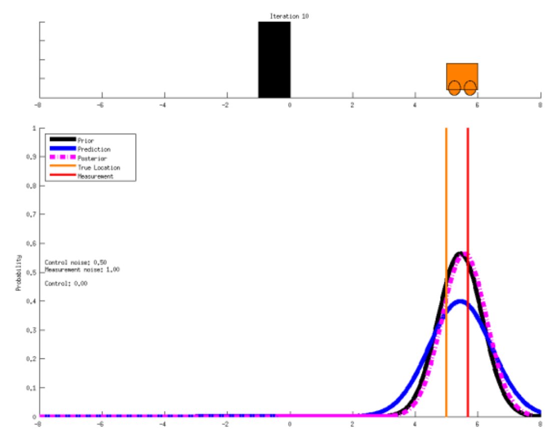
- All are map-based localization, Lecture 3
- **Question: How to obtain the map $m$? A hidden variable**




L7 Bayes Filter



L8 Particle Filter




L9 Kalman Filter and EKF

# EKF SLAM Today

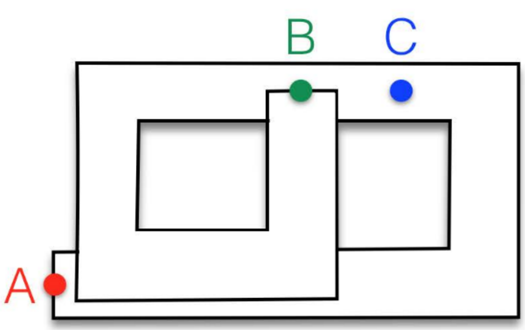
- Extended Kalman Filter-based Simultaneous Localization and Mapping (EKF SLAM)
- Goal: obtain both **feature map** and **robot poses** in **real time**

Courtesy: Victoria Dataset

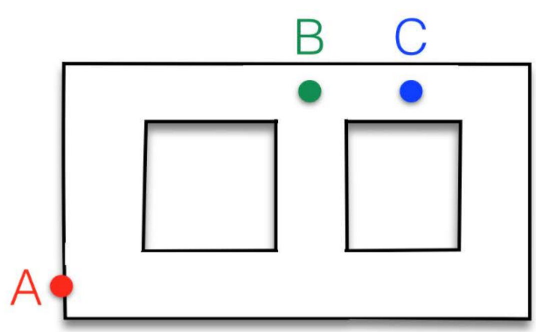
# Simultaneous Localization and Mapping

# Recap L3 - Robot Localization

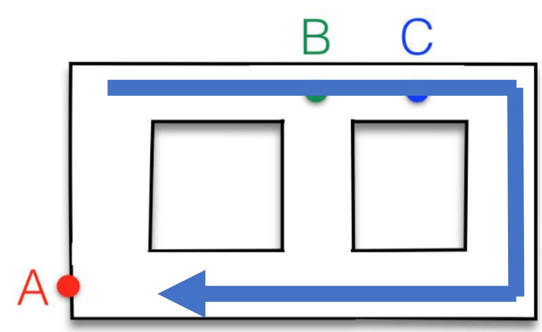
- **Odometry**
  - Wheel Odometry
  - Visual Odometry
  - LiDAR Odometry
  - etc




- **SLAM**
  - Simultaneous localization and mapping




- **Map-based Localization**
  - Localize on a given map



# Definition of SLAM Problem

- Given
  - The sent control commends (odometry)

$$u_{1:T} = \{u_1, u_2, u_3, \dots, u_T\}$$

  - Observations (measurements)

$$z_{1:T} = \{z_1, z_2, z_3, \dots, z_T\}$$

- Wanted
  - Map of the environment

$$m$$

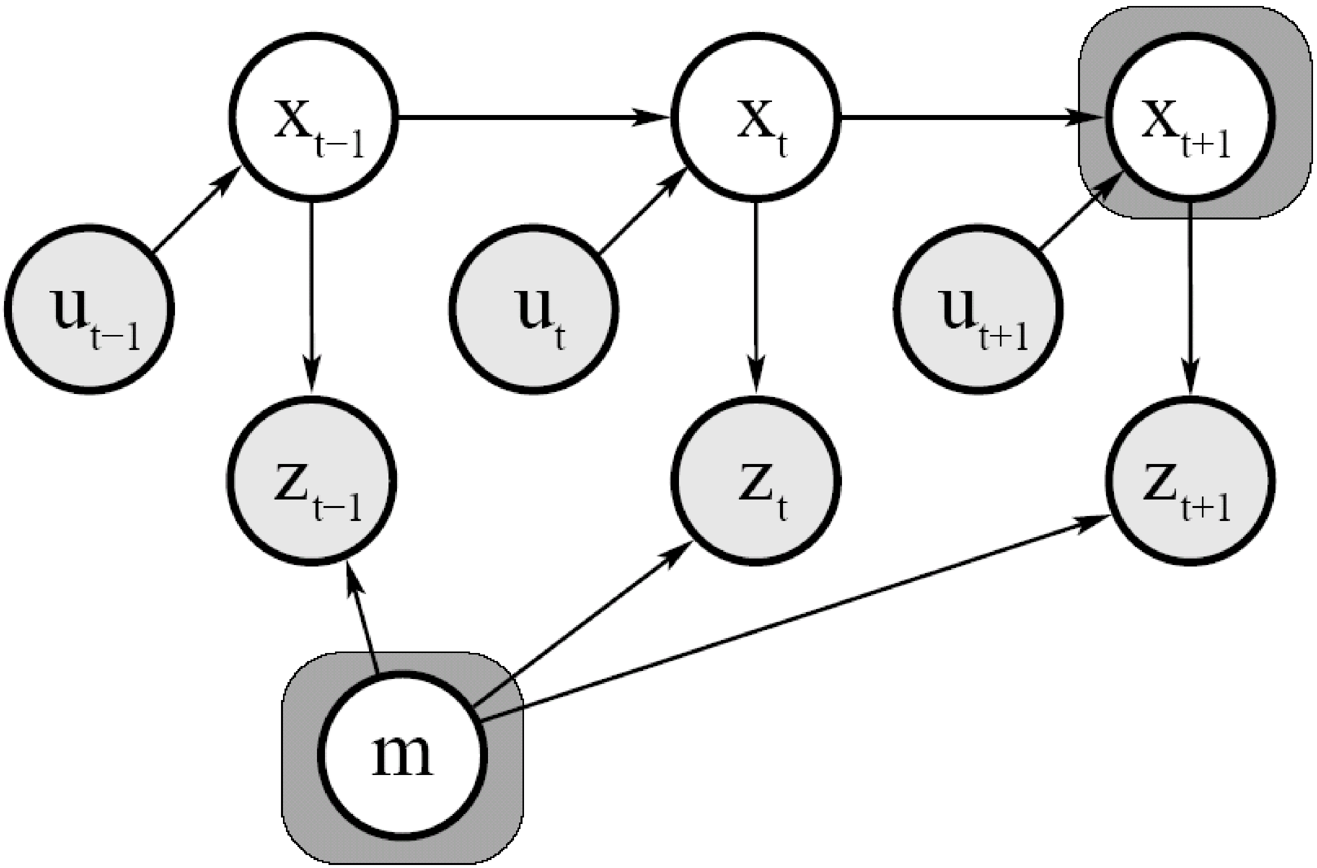
  - Path (or only current pose) of the robot

$$x_{0:T} = \{x_0, x_1, x_2, \dots, x_T\}$$
$$x_t$$

Courtesy: Cyrill Stachniss

# Graphical Model of Online SLAM

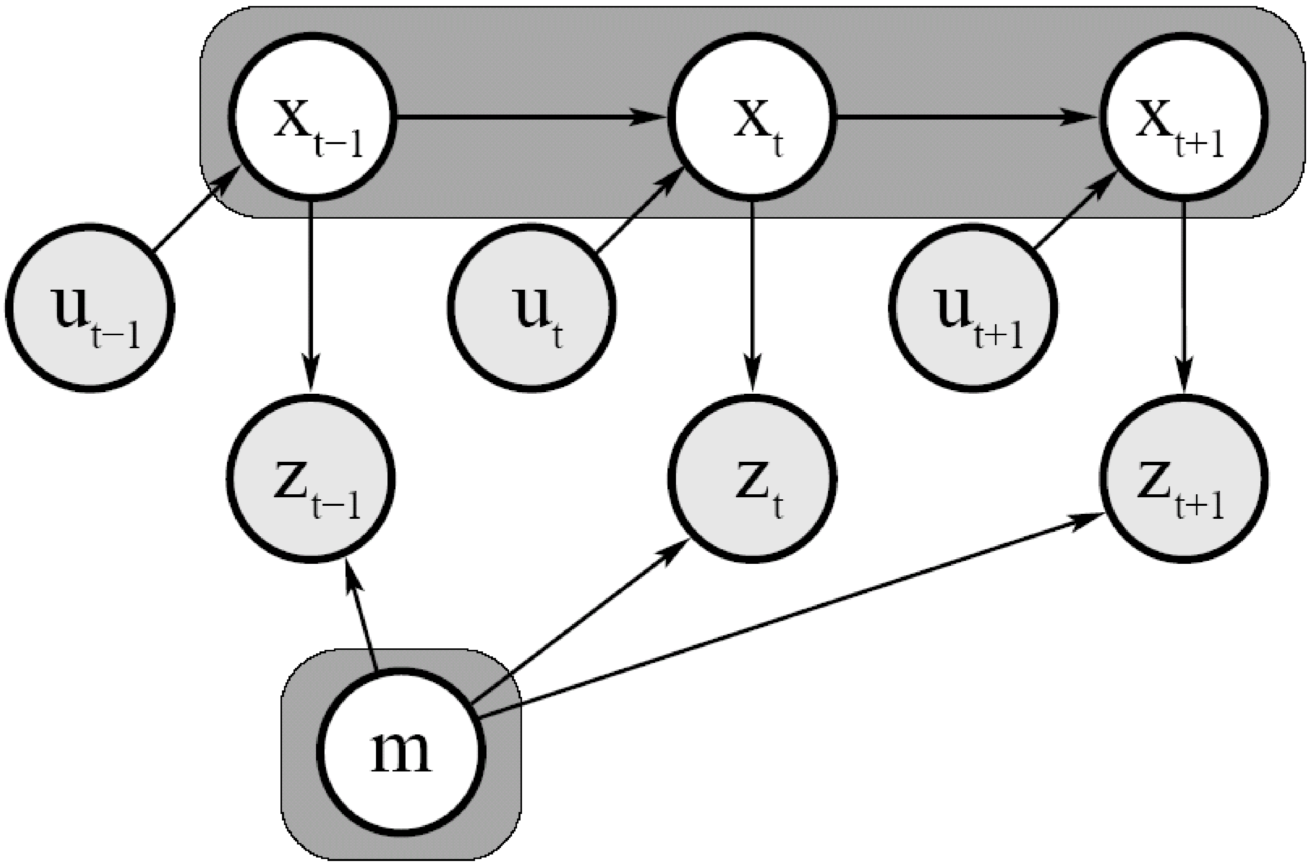
$$p(x_t, m \mid z_{1:t}, u_{1:t})$$




Courtesy: Thrun, Burgard, and Fox

# Graphical Model of Full SLAM

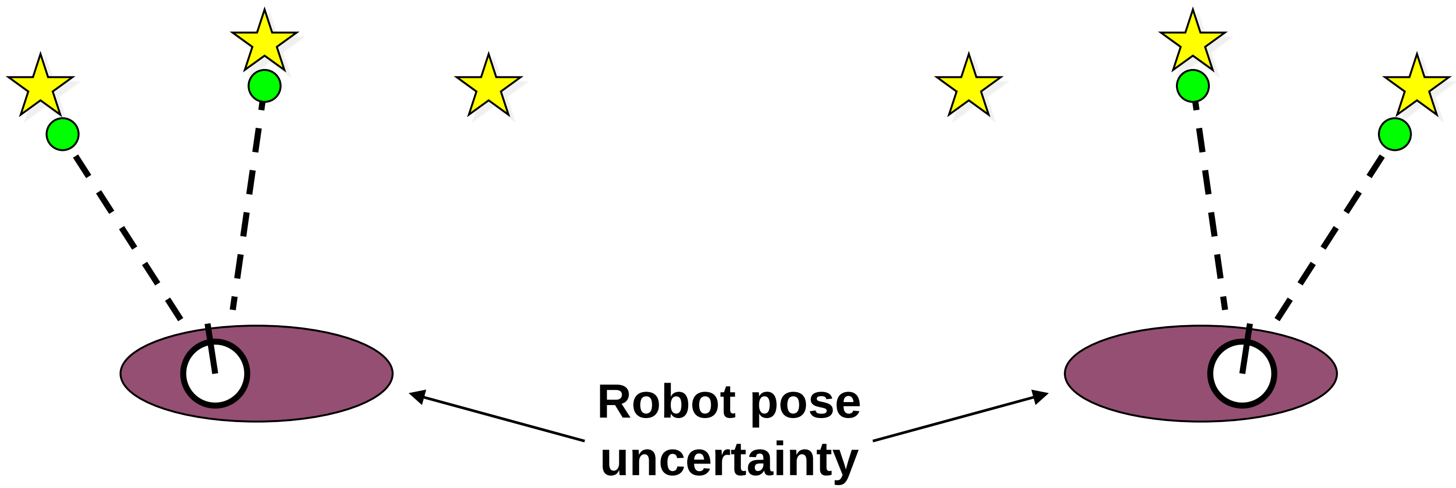
$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$




Courtesy: Thrun, Burgard, and Fox

# Why SLAM a hard Problem

- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations (Correspondece!) can have catastrophic consequences
- Chicken-or-Egg Problem
  - a map is needed for localization and
  - a pose estimate is needed for mapping




Courtesy: Thrun, Burgard, and Fox

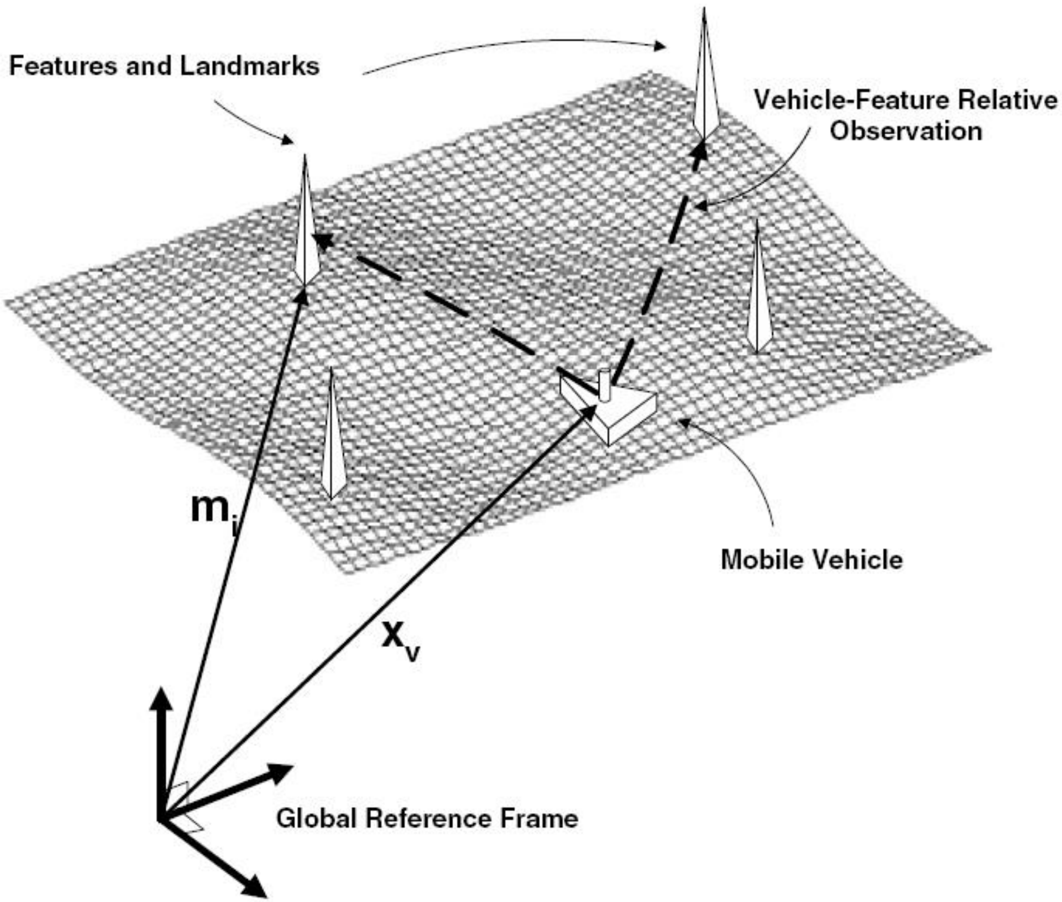
# SLAM Techniques

- EKF SLAM
- Fast SLAM (Particle Filter-based)
- Graph-based SLAM
- Topological SLAM (mainly place recognition)
- Scan Matching / Visual Odometry (only locally consistent maps)
- Approximations for SLAM: Local submaps, Sparse extended information filters etc.
- …

Courtesy: Wolfram Burgard

# EKF SLAM

# Problem




Courtesy: Thrun, Burgard, and Fox

# Recap L9 - EKF

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

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

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

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

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

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

$$\text{return } \mu_t, \Sigma_t$$

Courtesy: Cyrill Stachniss

# EKF for Online SLAM

- Application of the EKF to SLAM
- Estimate robot's pose and locations of landmarks in the environment
- Assumption: known correspondence
  - If unknown, search the nearest map landmark with observed one
  - Odometry is desired to be accurate enough
- State space for the 2D plane is

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

# State Representations

- Map with N landmarks: (3+2N)-dimensional Gaussian
- Belief is represented by

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

Courtesy: Cyrill Stachniss

# State Representations

- More compactly

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

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

Courtesy: Cyrill Stachniss

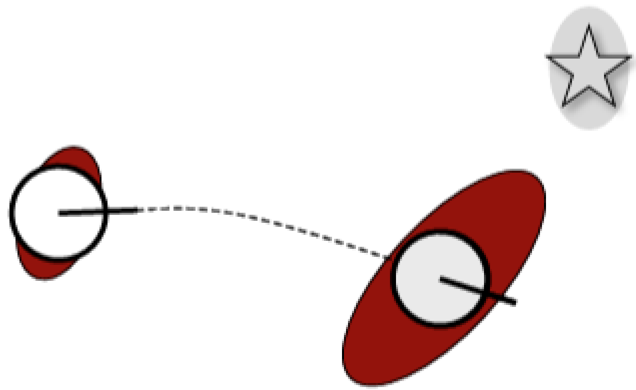
# EKF SLAM - Filter Cycle

1. State prediction
2. Measurement prediction
3. Measurement
4. Data association
5. Update

Courtesy: Cyrill Stachniss

# State Prediction

- State propagation with motion model

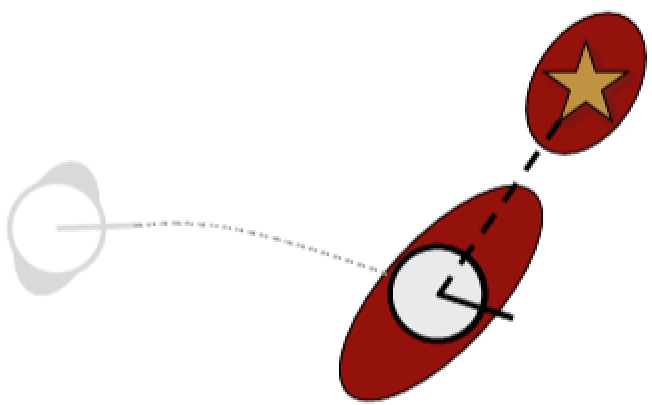


$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R}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

Courtesy: Cyrill Stachniss

# Measurement Prediction

- Predict the where the map landmark should be if it appears

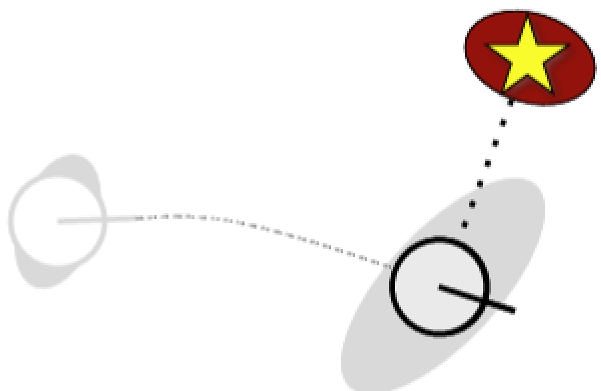


$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R}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

Courtesy: Cyrill Stachniss

# Obtained Measurement

- Observe a landmark using the sensor

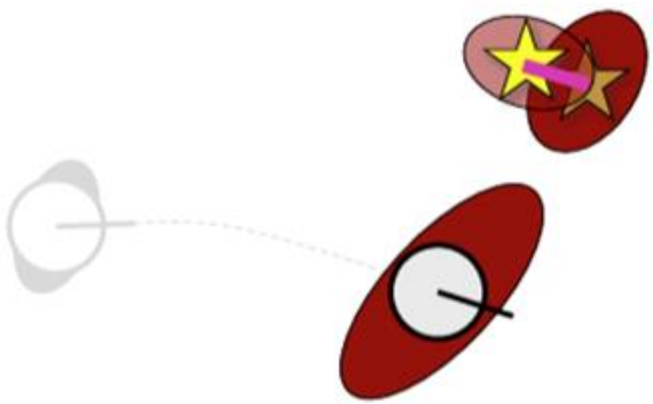


$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R}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

Courtesy: Cyrill Stachniss

# Difference Between h(x) and z

- Build the residual between as-predicted and as-observed

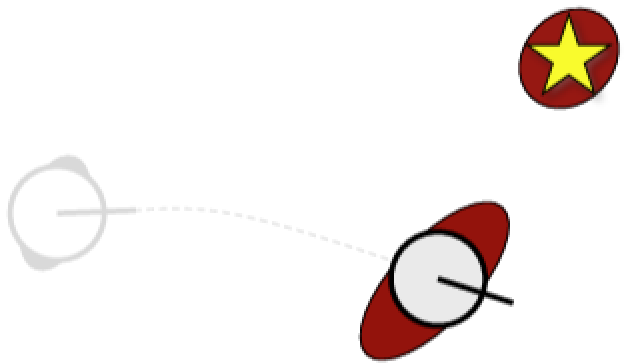


$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R}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

Courtesy: Cyrill Stachniss

# Update Step

- Update the map and the pose



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

Courtesy: Cyrill Stachniss

# EKF SLAM - Math Part

# EKF SLAM - Concrete Example

**Setup**

- Velocity-based motion model
  - Wheeled odometry, different from Kinematics in Lecture 3
- Observation of point landmarks
  - Feature Map, Lecture
- Range-bearing sensor
  - Laser scanner, Lecture 4
- Known data asscociation (correspondence)
  - If unknown, search the nearest map landmark with observed one
- Known number of landmarks
  - If unknown, set a number and increase the matrix size when needed

# Initialization

- Robot starts in its own reference frame
- All landmarks are unknown (set zero)
- 3 + 2N dimensions

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

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

Courtesy: Cyrill Stachniss

# EKF Algorithm

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

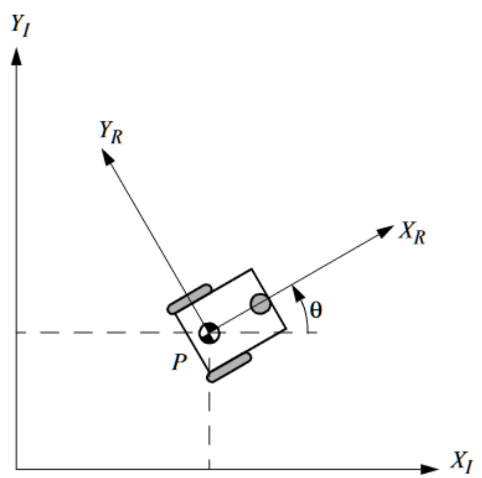
$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$$

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

Courtesy: Cyrill Stachniss

# Wheeled Motion

- Lecture 3
  - Differential drive robot
  - Predict the pose from motion
  - **Based on measured spinning wheel speeds**




$$\dot{\mathbf{X}}_I = R(\theta)^{-1}\dot{\mathbf{X}}_R$$

$$= R(\theta)^{-1}\begin{bmatrix} \frac{r\dot{\varphi}_1}{2} + \frac{r\dot{\varphi}_2}{2} \\ 0 \\ \frac{r\dot{\varphi}_1}{2l} + \frac{-r\dot{\varphi}_2}{2l} \end{bmatrix}$$

- Lecture 10
  - Differential drive robot
  - Predict the pose from motion
  - **Based on velocities (the sent control commands)**
  - ***Probabilistic Robotics*, Chapt. 5.3**
  - **Prime = Bar:** $x' = \bar{x}$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ y_c - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \theta + \omega\Delta t \end{pmatrix}$$

$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega}\sin\theta + \frac{v}{\omega}\sin(\theta + \omega\Delta t) \\ \frac{v}{\omega}\cos\theta - \frac{v}{\omega}\cos(\theta + \omega\Delta t) \\ \omega\Delta t \end{pmatrix}$$

# Prediction Step

- Goal: Update the state space based on the motion

- Motion in the plane

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

- How to map to the 3+2N dim state space in the EKF?

Courtesy: Cyrill Stachniss

# Update the State Space

- From the motion in the plane

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

- to the 3+2N dimensional space

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

Courtesy: Cyrill Stachniss

# EKF Algorithm

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

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

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

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

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

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

7: return $\mu_t, \Sigma_t$

Courtesy: Cyrill Stachniss

# Update Covariance

- The function $g$ only affects the motion not the landmarks

**Jacobian of the motion (3 x 3)**

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

**Identity (2N x 2N)**

Courtesy: Cyrill Stachniss

# Jacobian of the Motion

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

$$= I + \frac{\partial}{\partial(x,y,\theta)^T}\begin{pmatrix} -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ \frac{v_t}{\omega_t}\cos\theta - \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ \omega_t\Delta t \end{pmatrix}$$

$$= I + \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

Courtesy:

# This Leads to the Update

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

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

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

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

Courtesy: Cyrill Stachniss

# EKF SLAM - Prediction Step

**EKF_SLAM_Prediction**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, R_t)$:

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

Courtesy: Cyrill Stachniss

# EKF Algorithm

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

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

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

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

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

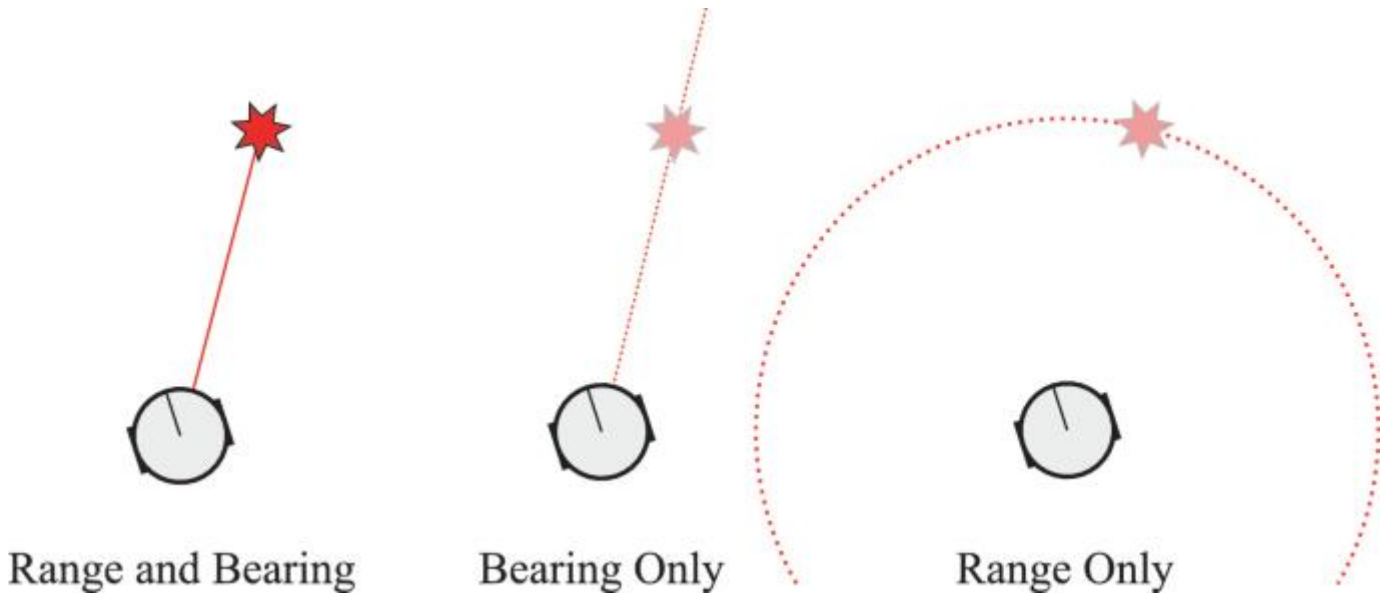
6: $\Sigma_t = (I - K_t\ H_t)\ \bar{\Sigma}_t$

7: return $\mu_t, \Sigma_t$

Courtesy: Cyrill Stachniss

# Range-Bearing Observation

- Detect a landmark with the laser scan
  - like clustering the laser points and detect with a classifier
- Range (meter) and orientation (degree/rad) respected to the robot
- Also work for stereo vision



# Range-Bearing Observation

- Range-Bearing observation: $z_t^i = (r_t^i, \phi_t^i)^T$
- If landmark has not been observed before,

  we can intilialize it with:

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

**Location of landmark j** | **Estimated Location of the Robot** | **Relative Measurement**

Courtesy: Cyrill Stachniss

# Expected Observation

- $c_t^i = j$ , $i$-th measurement at the timestamp $t$ correspondes to the landmark $j$
- Compute the expected observation according to the current state
- That's what we need for the residual

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

Courtesy: Cyrill Stachniss

# Jacobian for the Observation

- Based on

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

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

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

$$= h(\bar{\mu}_t)$$

- Compute the Jacobian

$$^{\text{low}}H_t^i = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t}$$

**Low-dimensional space** $(x, y, \theta, m_{j,x}, m_{j,y})$

Courtesy: Cyrill Stachniss

# Jacobian for the Observation

- Based on

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

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

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

- Compute the Jacobian

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

Courtesy: Cyrill Stachniss

# Jacobian for the Observation

- Based on

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

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

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

$$= h(\bar{\mu}_t)$$

- Applying the chain rule, we obtain

$$\frac{\partial \sqrt{q}}{\partial x} = \frac{1}{2} \frac{1}{\sqrt{q}} 2\delta_x(-1)$$

$$= \frac{1}{q}(-\sqrt{q}\delta_x)$$

Courtesy: Cyrill Stachniss

# Jacobian for the Observation

- Based on

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

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

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

$$= h(\bar{\mu}_t)$$

- Compute the Jacobian

$$\text{low } H_t^i = \frac{\partial h(\overline{\mu_t})}{\partial \bar{\mu}_t}$$

$$= \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix}$$

Courtesy: Cyrill Stachniss

# Jacobian for the Observation

- Use the computed Jacobian

$$ {}^{\text{low}}H_t^i = \frac{\partial h(\bar{\mu}_t)}{\partial \bar{\mu}_t} $$

$$ = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & \delta_x \end{pmatrix} $$

- Map it to the high dimensional space

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

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

Courtesy: Cyrill Stachniss

# EKF SLAM - Update Step (1/2)

**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*

$\quad j = c_t^i$

$\quad$ *if landmark* $j$ *never seen before*

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

$\quad$ *endif*

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

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

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

Courtesy: Cyrill Stachniss

# EKF SLAM - Update Step (2/2)

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

$$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x \end{pmatrix} F_{x,j}$$

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

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

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

endfor

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

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

return $\mu_t, \Sigma_t$

Courtesy: Cyrill Stachniss

# EKF Algorithm

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

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

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

Courtesy: Cyrill Stachniss

# EKF SLAM Summary

- The first SLAM solution
- Convergence results for the linear case
- Can diverge if nonlinearities are large!
- Have been applied successfully in medium-scale environments
- Approximations reduce the computational complexity

IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, VOL. 17, NO. 3, JUNE 2001 229

A Solution to the Simultaneous Localization and Map Building (SLAM) Problem

M. W. M. Gamini Dissanayake, *Member, IEEE*, Paul Newman, *Member, IEEE*, Steven Clark,
Hugh F. Durrant-Whyte, *Member, IEEE*, and M. Csorba

# EKF SLAM Application - 1

- Victoria Park, Sydney
- https://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm

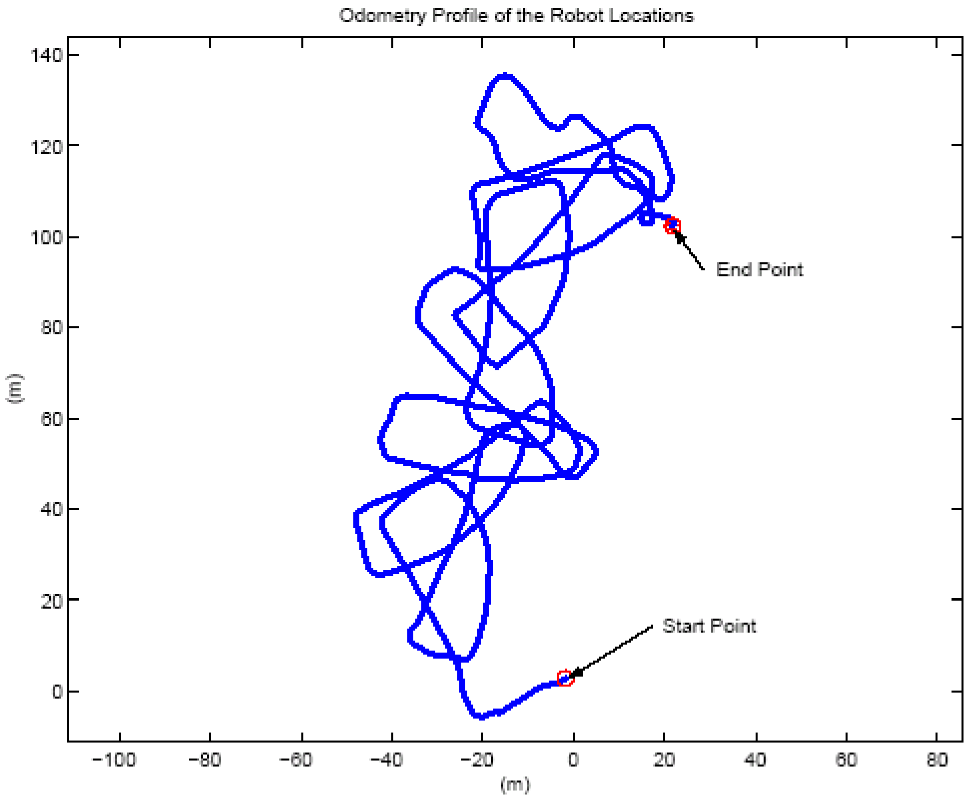
Courtesy: Victoria Dataset

# With Submap

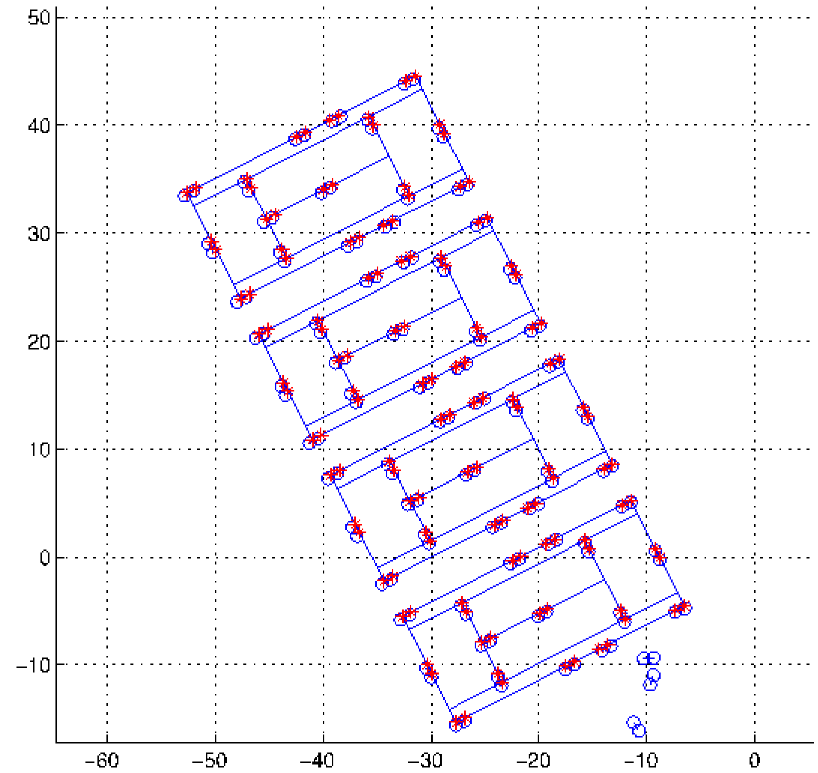
# EKF SLAM Application - 2


Courtesy: John Leonard

# EKF SLAM Application - 2




**Wheel Odometry**

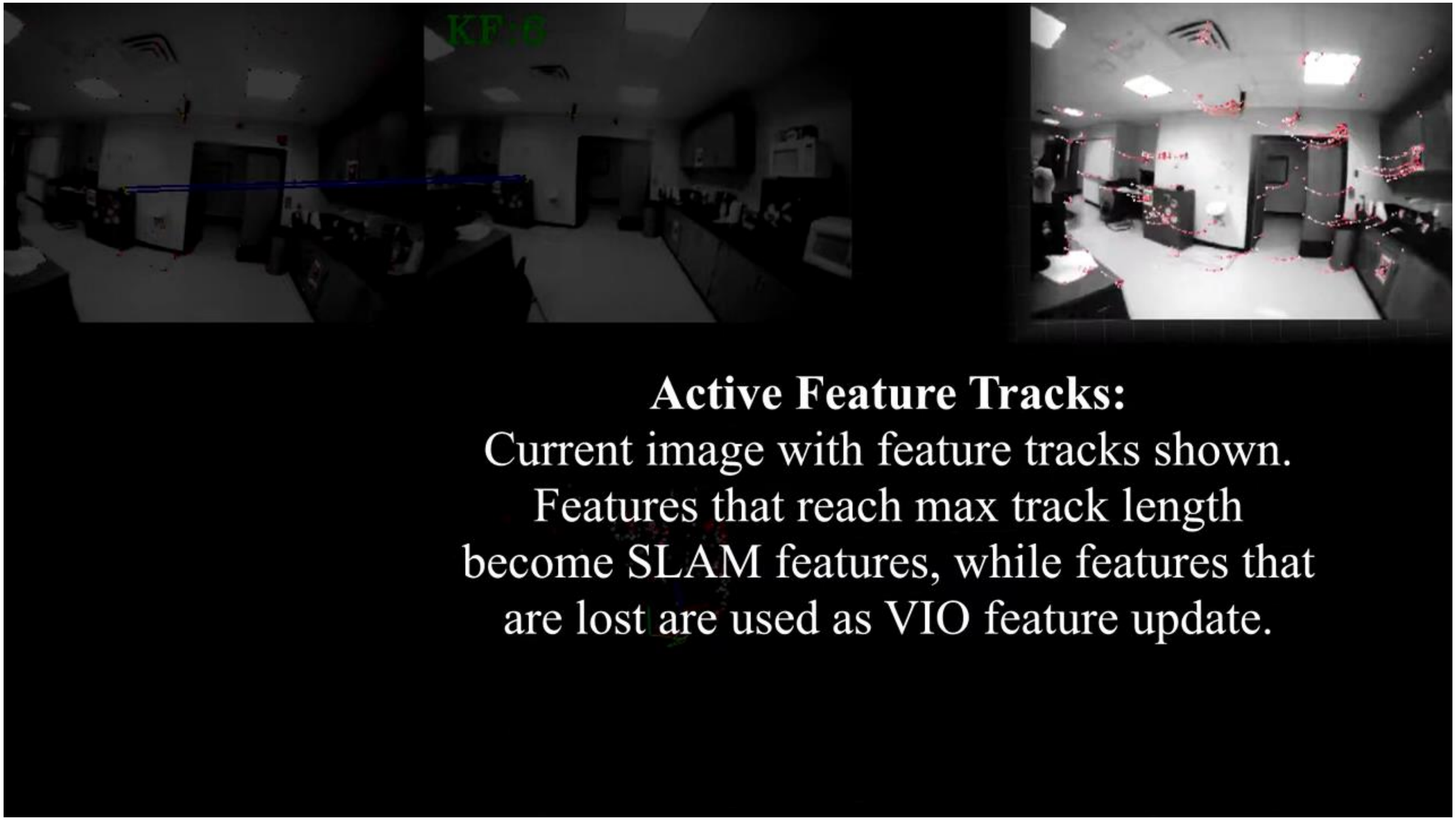



**EKF SLAM Trajectory**

Courtesy: John Leonard

# EKF Visual SLAM

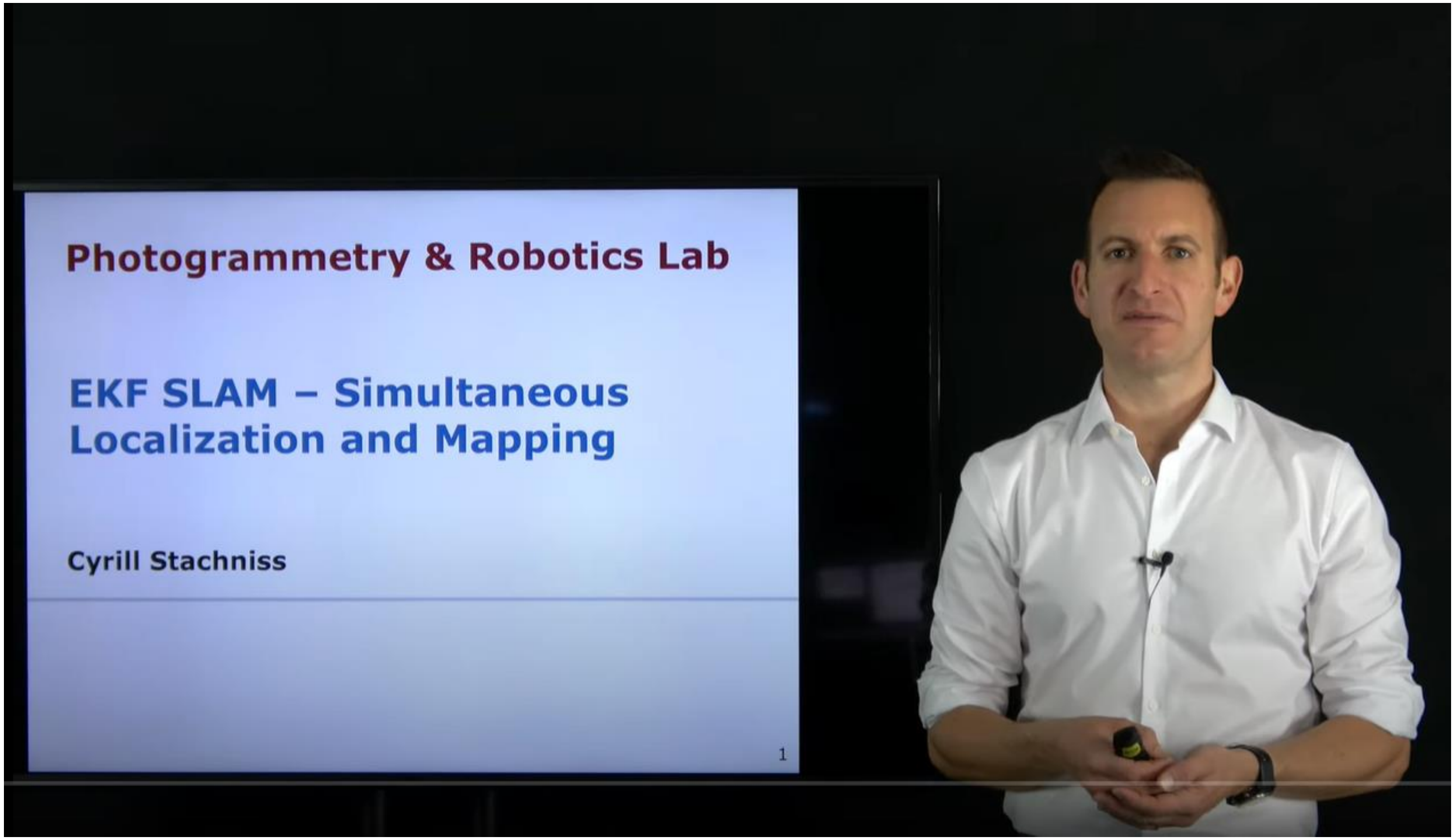
- Schmidt-EKF Visual-inertial SLAM




Courtesy: Robot Perception and Navigation Group

# Resources

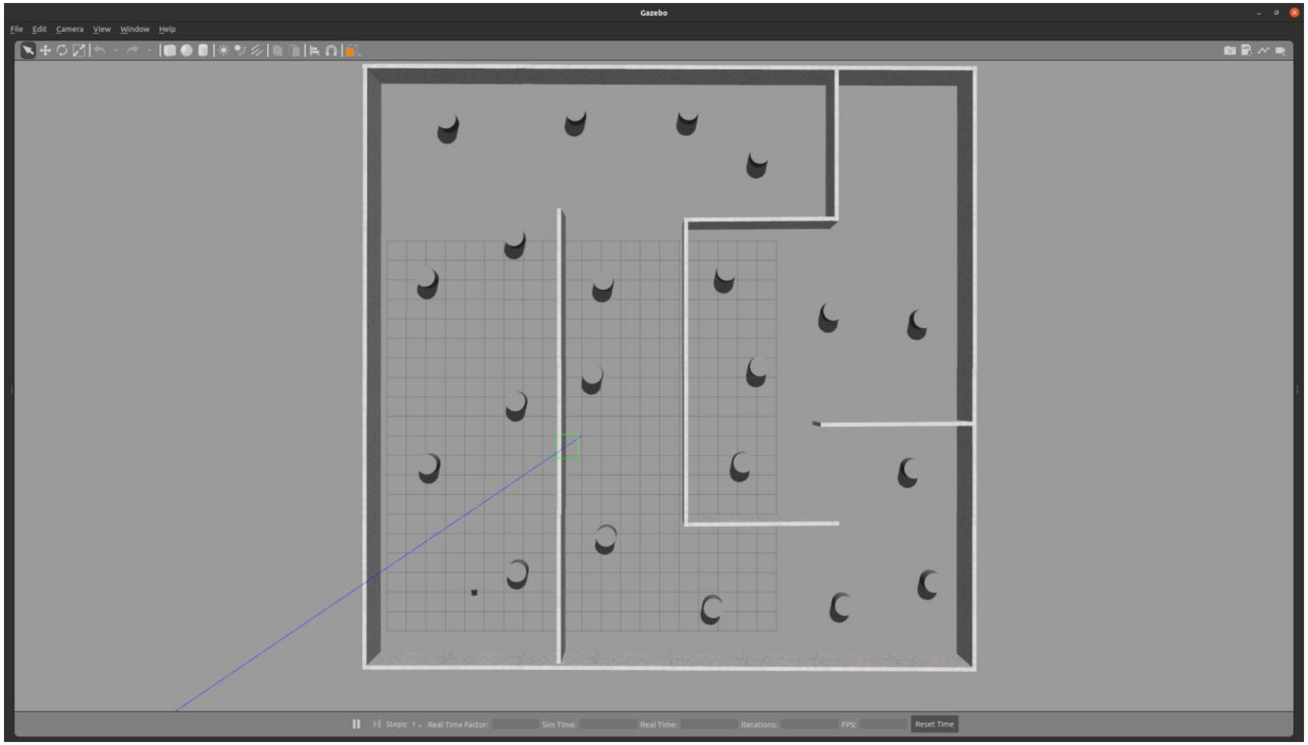
- Probabilistic Robotics. Chapter 10.3
- Prof. Cyrill Stachniss



# Project 2

# Virtual Lab

- A mobile robot with a LiDAR Scanner
- Landmark-based Laser EKF SLAM



# Video by TAs

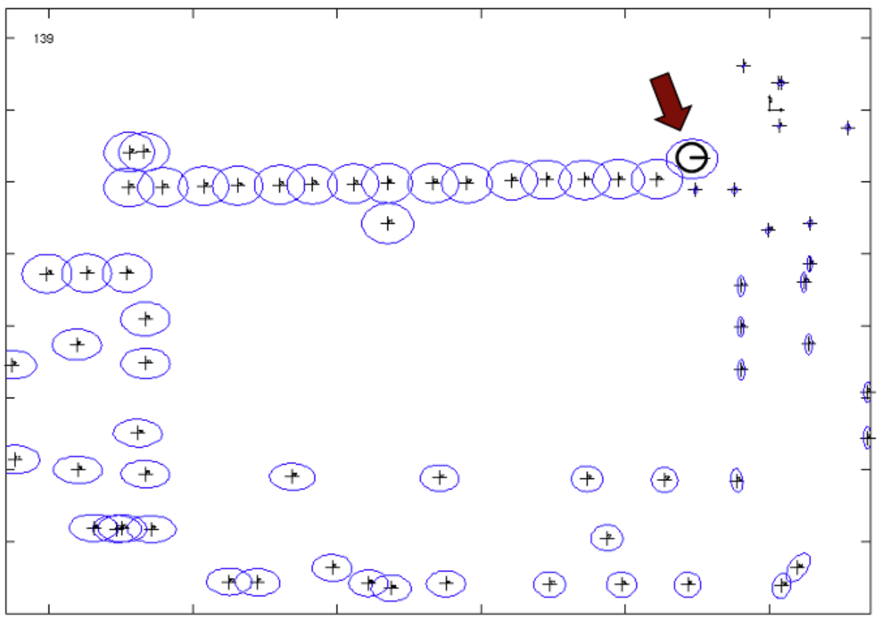
# Notes

- Manage the map and poses
- You may need to create the $F$ matrices explicitly
- Always normalize the angular components!
- Write and Debug!
    - Test the functions individually and separately
- Do not forget the deadline!
    - Three weeks for you
    - November 8th

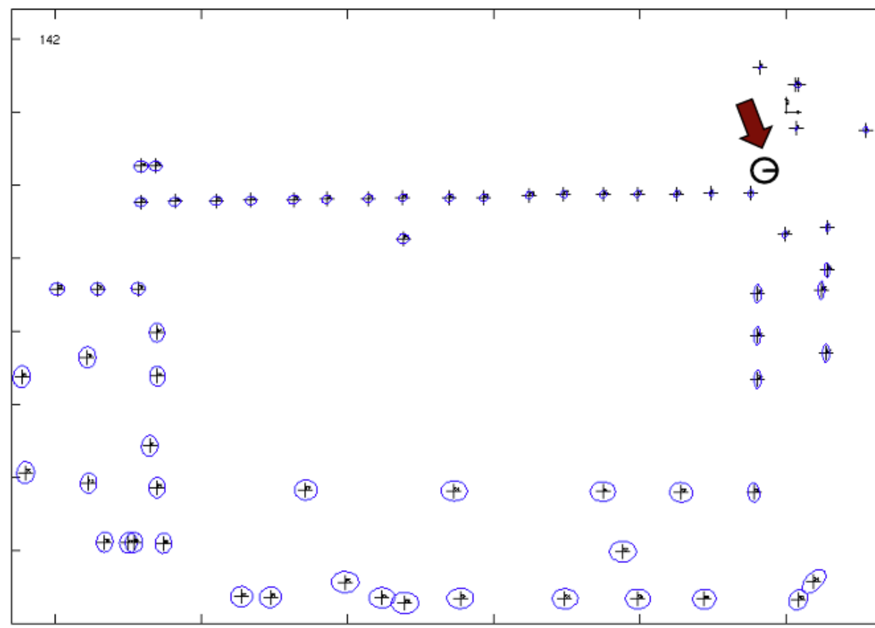
# Loop Closing

# Loop Closing

- Loop closing reduces the uncertainty in robot and landmark estimates
- This can be  exploited when exploring an environment for the sake of better (more accurate) maps
- Wrong loop closures lead to filter divergence




**Before Loop Closing**




**After Loop Closing**

Courtesy: Cyrill Stachniss and K. Arras

# Next Lecture

- Place Recognition
  - Needed in large-scale environments
  - A more general technique that includes loop closing
  - A prior condition for Lecture 12 Graph SLAM