# SLAM概述及机器人领域中的应用

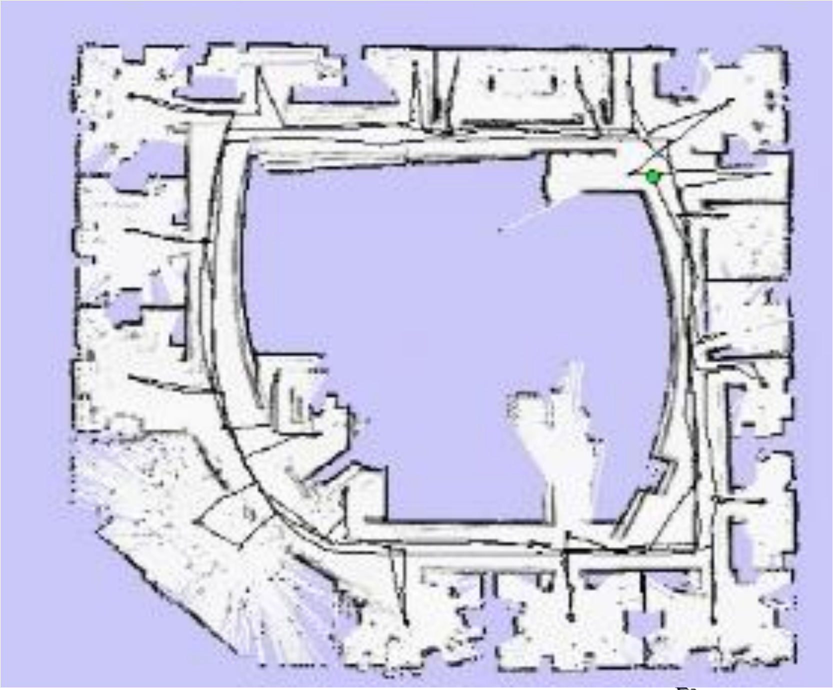
俞毓锋

2016年11月17日



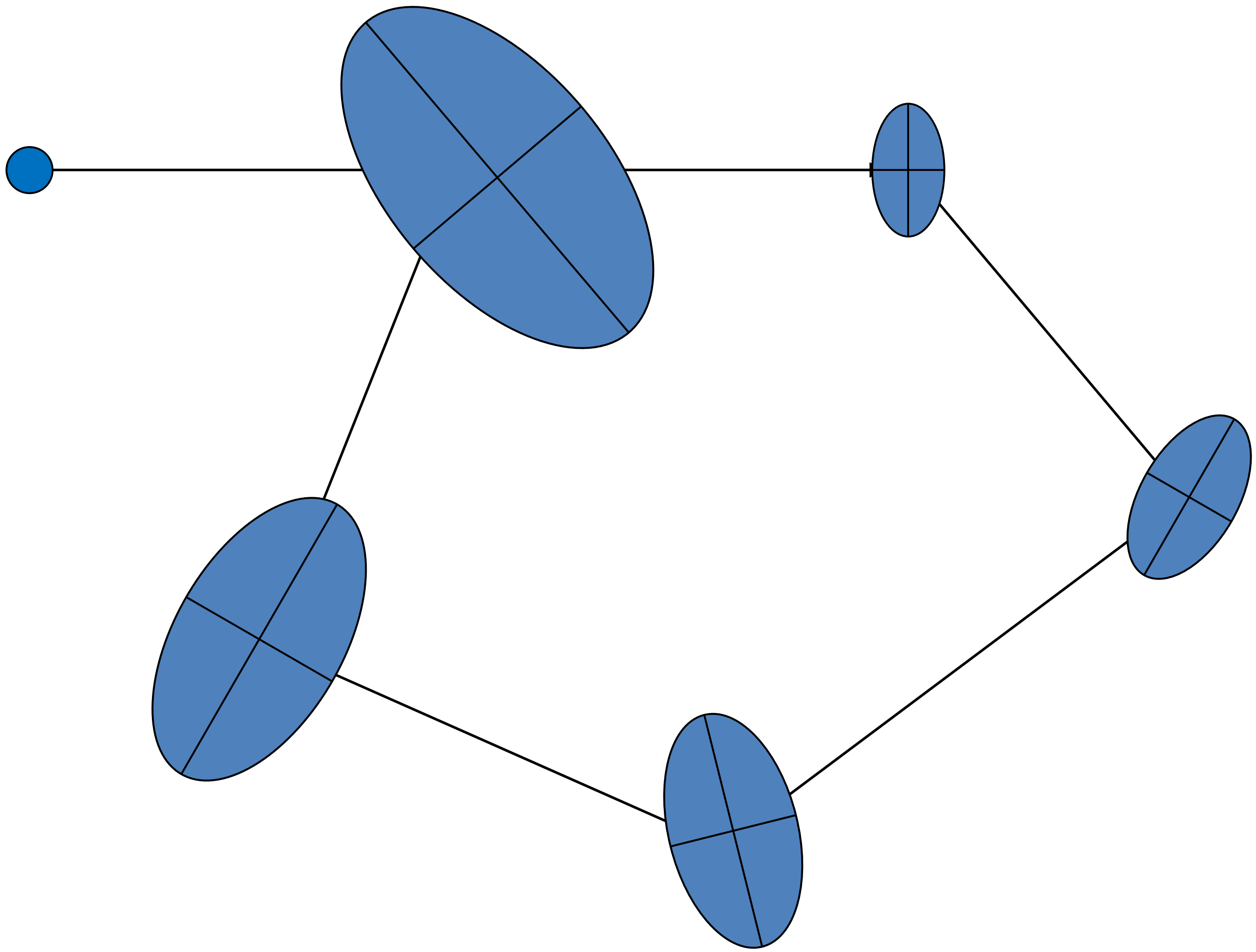
# The SLAM Problem

- Simultaneous localization and mapping
  - A robot is exploring an unknown, **static environment**
  - Doing localization and mapping Simultaneously
- **Given:**
  - The robot's controls
  - Observations of nearby features
- **Estimate:**
  - Map of features
  - Path of the robot



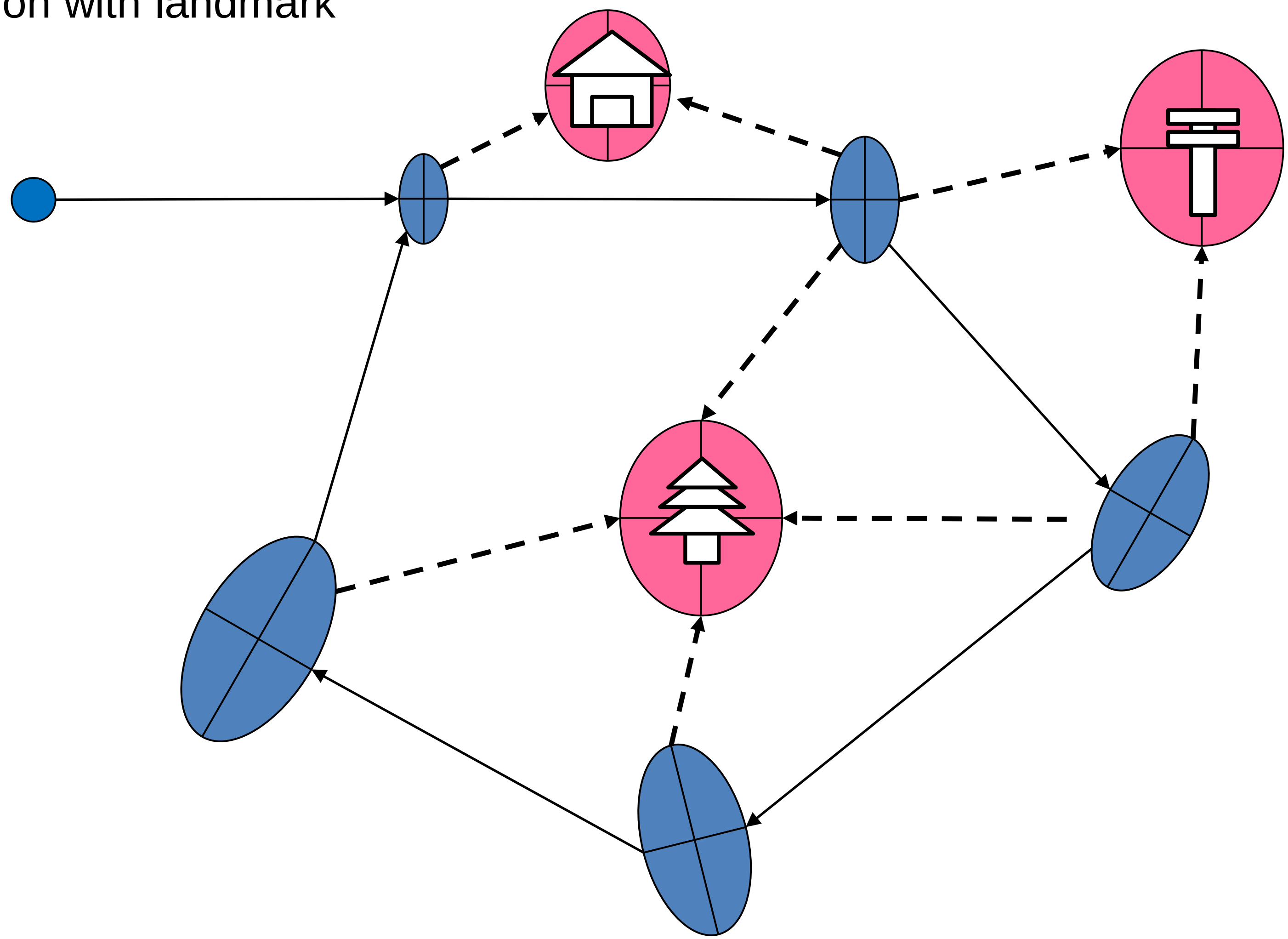
# How people do SLAM with a compass?

Localization in desert



# How people do SLAM with a compass?

Localization with landmark

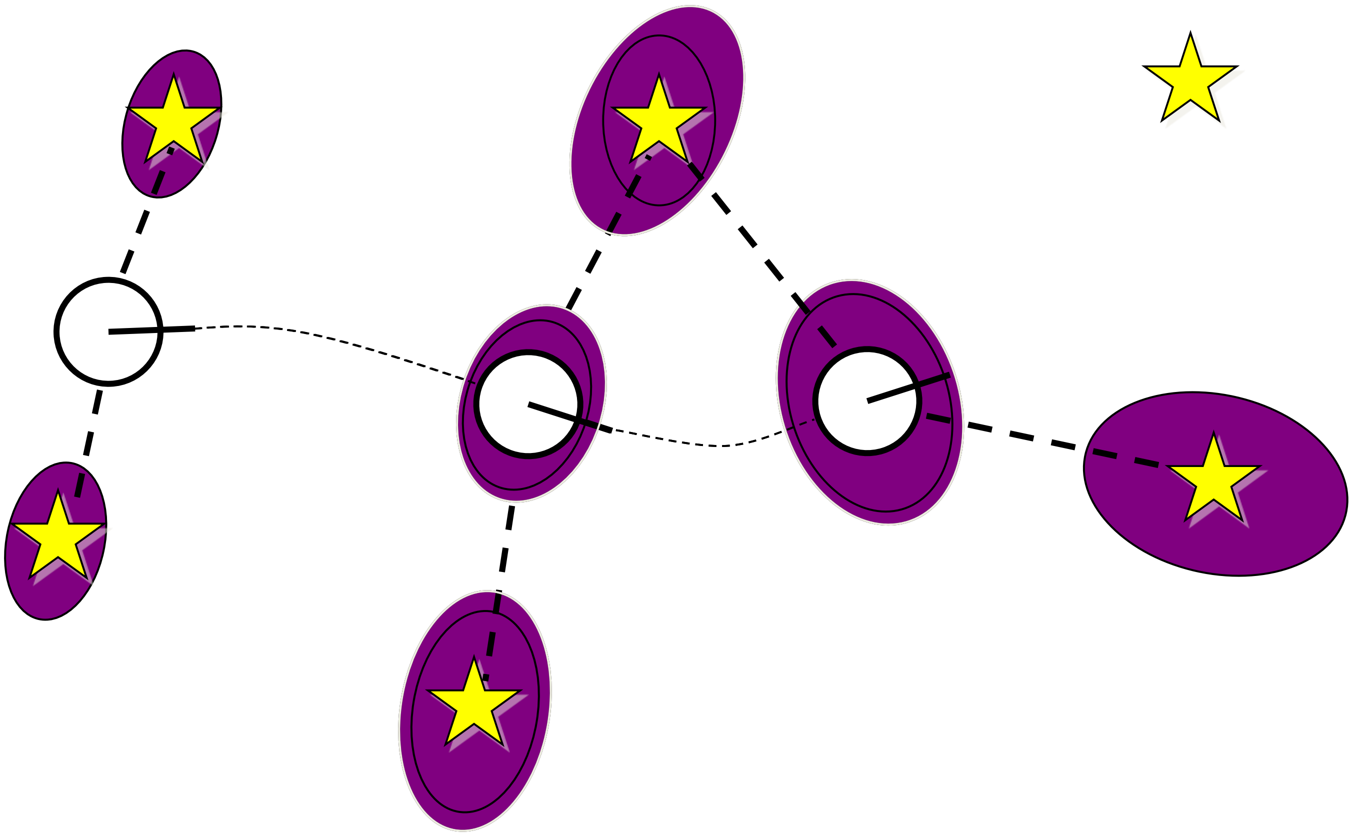


# How people do SLAM with a compass?

- Prediction
  - Predict Our location using compass and steps
- Observation
  - See all the landmarks we can see
  - Remember the location
- Correction
  - Use what wee have seen before to correct our location
- Mapping
  - Renew the landmarks we have seen before
  - Draw new landmarks on map



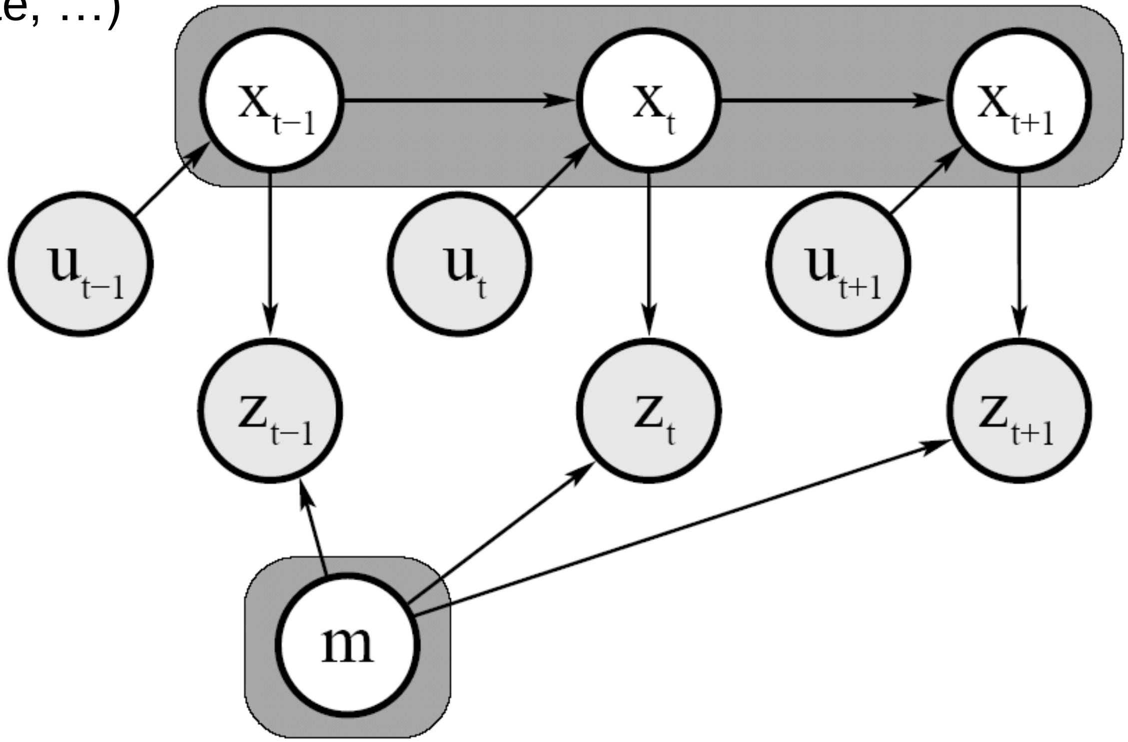
# SLAM for robots



http://robots.stanford.edu/probabilistic-robotics/

# Mathmatical definition for SLAM problem

- **X**: Pose (position and orientation)
- **U**: Control (velocity, yawrate, …)
- **Z**: Observation
- **M**: Map



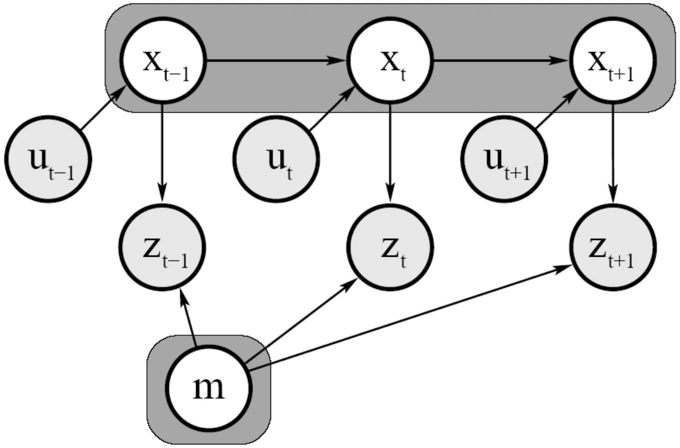

- What we have
  - U,Z
- What to do
  - X,M

# Full SLAM and Online SLAM

- Full SLAM:

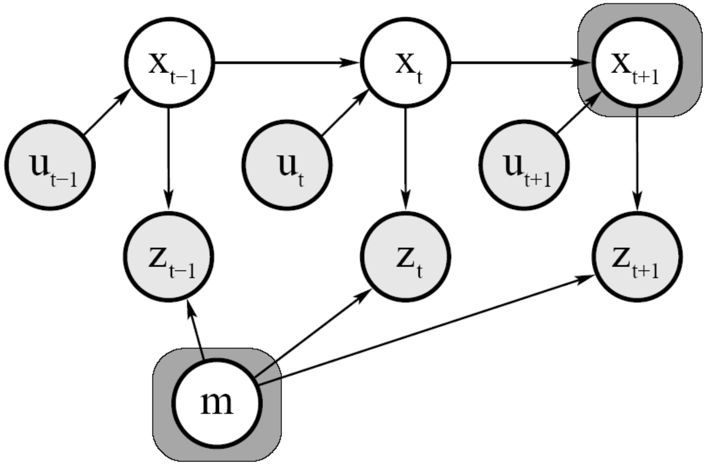
Estimates entire path and map!

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




- Online SLAM:

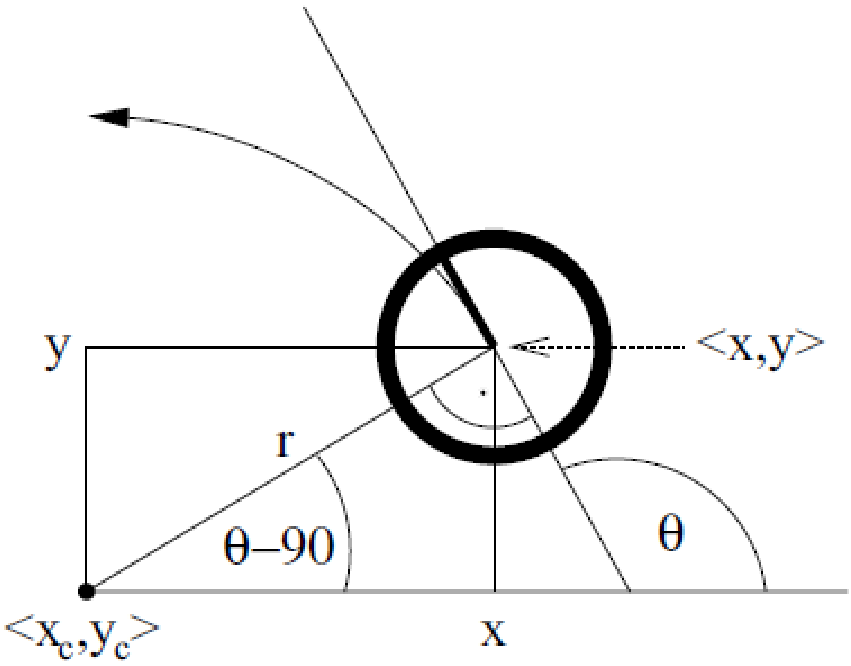
Estimates most recent pose and map!




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

# Motion Model

- State: x,y,θ
- Control: v,ω

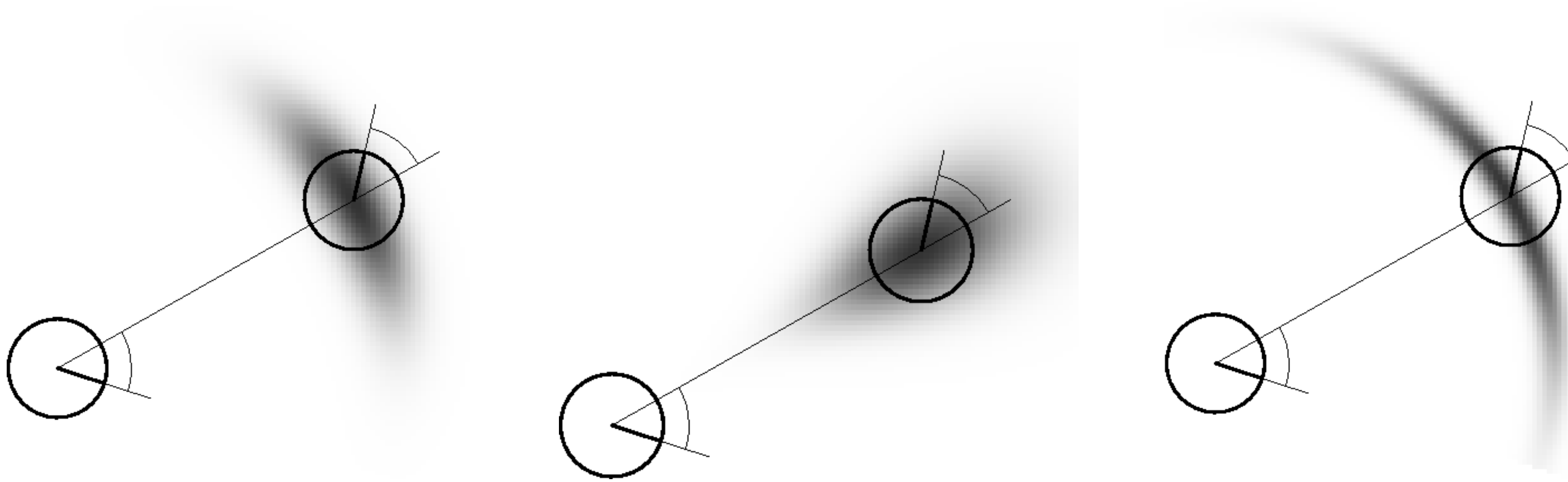



$$r = \left|\frac{v}{\omega}\right|$$

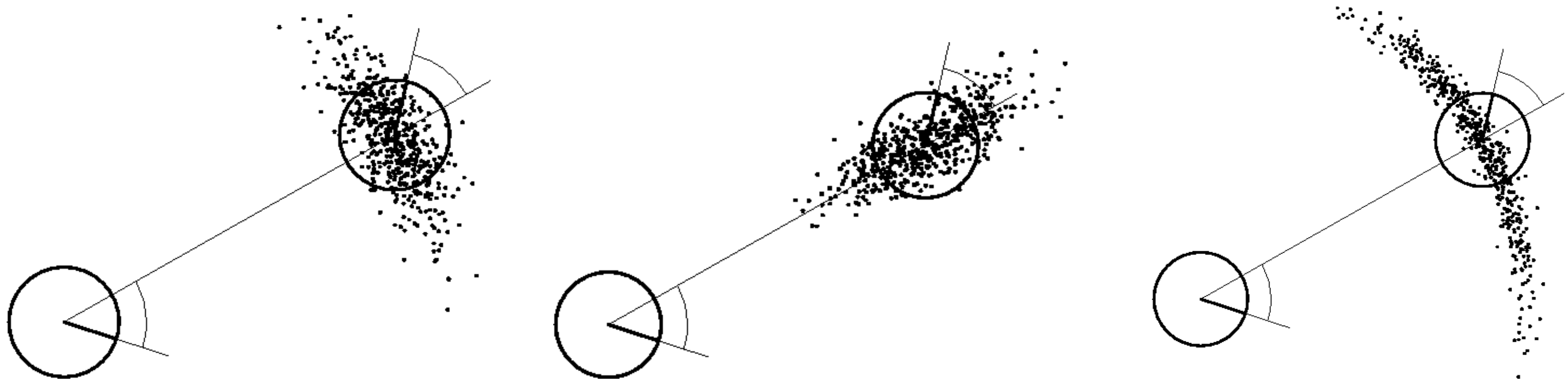
$$\begin{aligned} x_c &= x - \frac{v}{\omega}\sin\theta \\ y_c &= y + \frac{v}{\omega}\cos\theta \end{aligned}$$

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

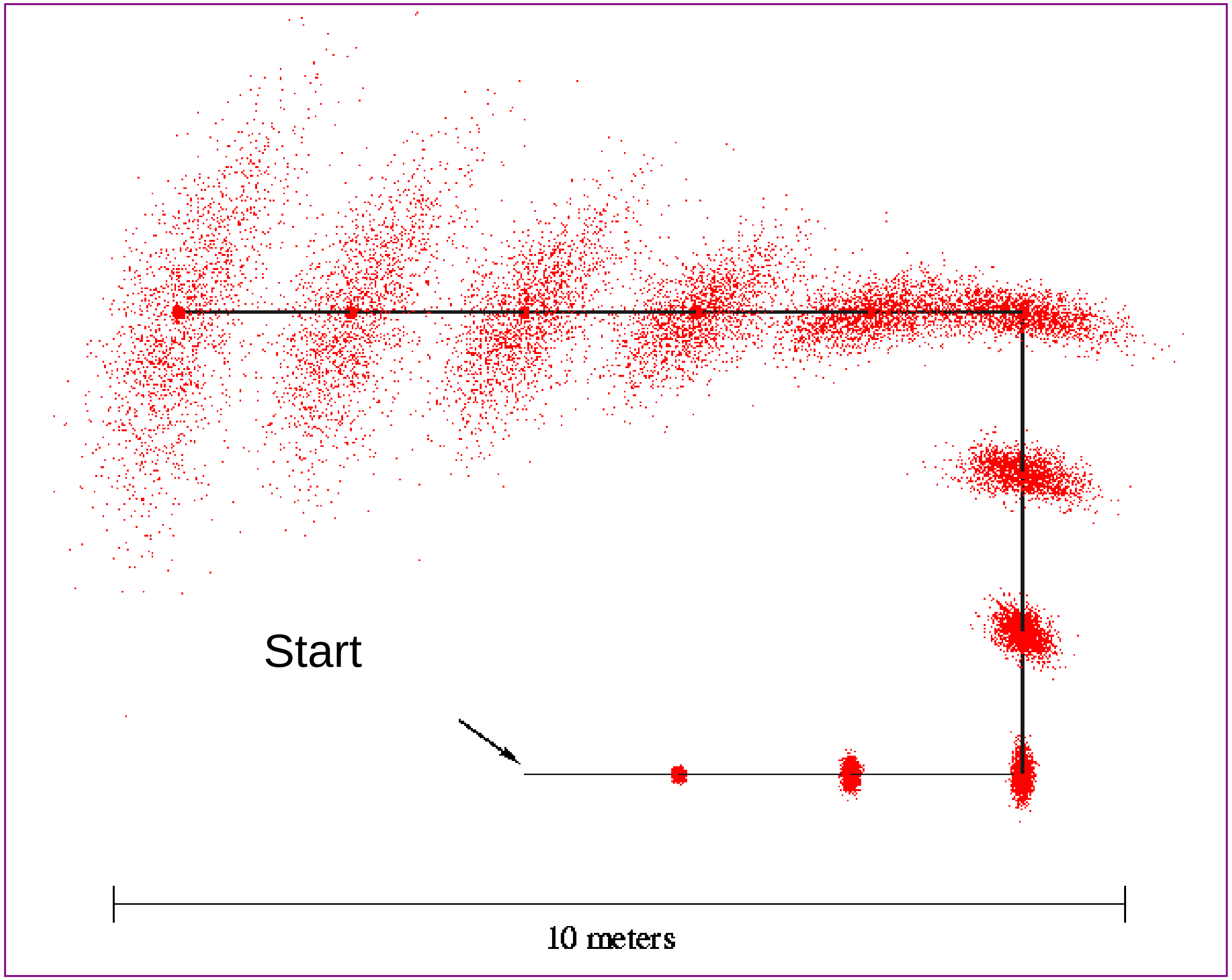
# Motion Model



$e_v >> e_\omega$ $e_\omega >> e_v$



# Motion Model



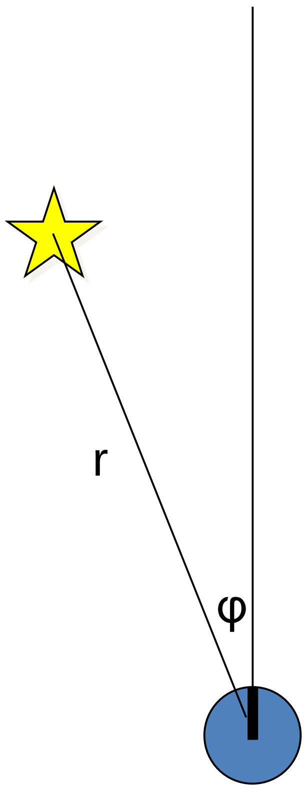
# Observation Model

- Observe: r, φ

$$\boldsymbol{\delta} = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = \begin{bmatrix} m_{j,x} - \mu_{t,x} \\ m_{j,y} - \mu_{t,y} \end{bmatrix}$$

$$r = \boldsymbol{\delta}^T \boldsymbol{\delta}$$

$$\varphi = \operatorname{atan2}\left(\delta_y, \delta_x\right) - \mu_{t,\theta}$$



# Basic Algorithm: Kalman Filter

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

2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$

5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t (z_t - H_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$

9. Return $\mu_t, \Sigma_t$

Linear Gaussian Dynamics:

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

$$\boxed{p(x_t \mid u_t, x_{t-1}) = N\left(x_t; A_t x_{t-1} + B_t u_t, R_t\right)}$$

Linear Gaussian Observations:

$$\boxed{z_t = H_t x_t + \delta_t}$$

$$\boxed{p(z_t \mid x_t) = N\left(z_t; H_t x_t, Q_t\right)}$$



# Basic Algorithm: Extended Kalman Filter

- Prediction:

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}} (x_{t-1} - \mu_{t-1})$$

$$g(u_t, x_{t-1}) \approx g(u_t, \mu_{t-1}) + G_t \, (x_{t-1} - \mu_{t-1})$$

- Correction:

$$h(x_t) \approx h(\bar{\mu}_t) + \frac{\partial h(\bar{\mu}_t)}{\partial x_t} (x_t - \bar{\mu}_t)$$

$$h(x_t) \approx h(\bar{\mu}_t) + H_t \, (x_t - \bar{\mu}_t)$$



# Basic Algorithm: Extended Kalman Filter

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

2. Prediction:
3. $$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad \longleftarrow \quad \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
4. $$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad \longleftarrow \quad \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$
5. Correction:

6. $$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1} \quad \longleftarrow \quad K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$
7. $$\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \quad \longleftarrow \quad \mu_t = \bar{\mu}_t + K_t(z_t - H_t \bar{\mu}_t)$$
8. $$\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \quad \longleftarrow \quad \Sigma_t = (I - K_t H_t)\bar{\Sigma}_t$$

9. Return $\mu_t$, $\Sigma_t$

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$



# Extended Kalman Filter for SLAM

- State:
  - Map with N landmarks:(3+2N)-dimensional Gaussian

$$
Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle
$$

# Extended Kalman Filter for SLAM

- Prediction:

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

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & \underbrace{0 \cdots 0}_{3N} \end{pmatrix}$$

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

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

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



# Extended Kalman Filter for SLAM

- Observation:

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

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

$$j = c_t^i$$

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

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

*endif*

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

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

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



# Extended Kalman Filter for SLAM

- Correction:

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 1 & 0 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 1 & 0\cdots0 & 0 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 1 & 0 & 0 & 0\cdots0 \\ 0 & 0 & 0 & 0\cdots0 & 0 & 1 & 0 & 0\cdots0 \\ 0 & 0 & 0 & \underbrace{0\cdots0}_{3j-3} & 0 & 0 & 1 & \underbrace{0\cdots0}_{3N-3j} \end{pmatrix}$$

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

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

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

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

endfor

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

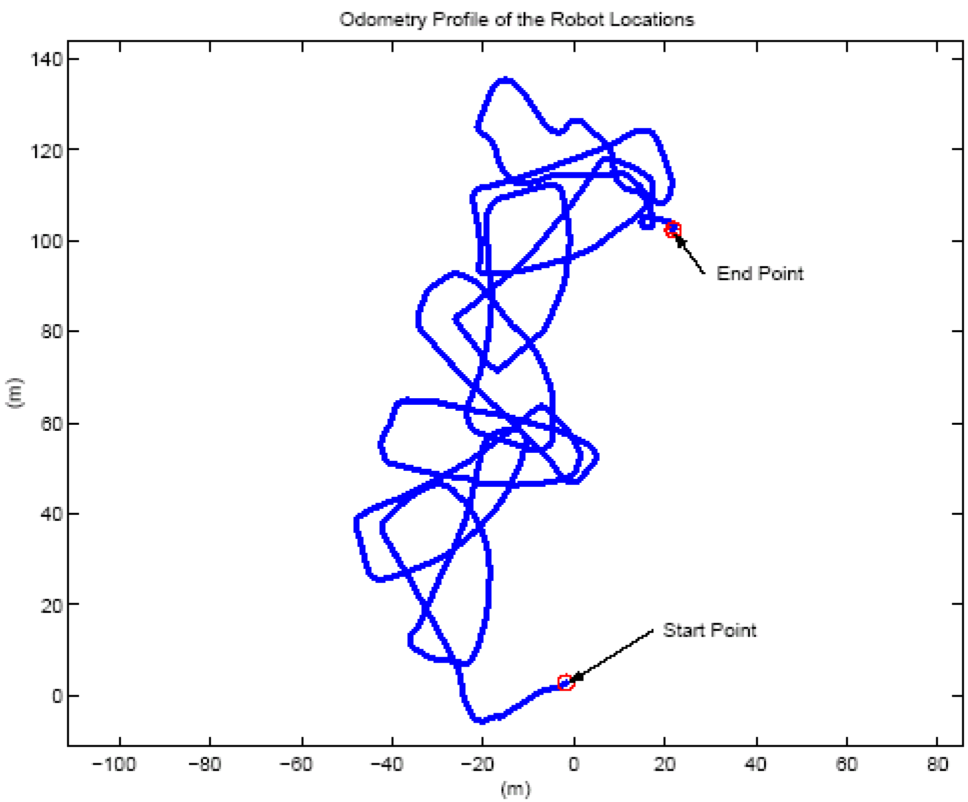
$$\Sigma_t = \bar{\Sigma}_t$$

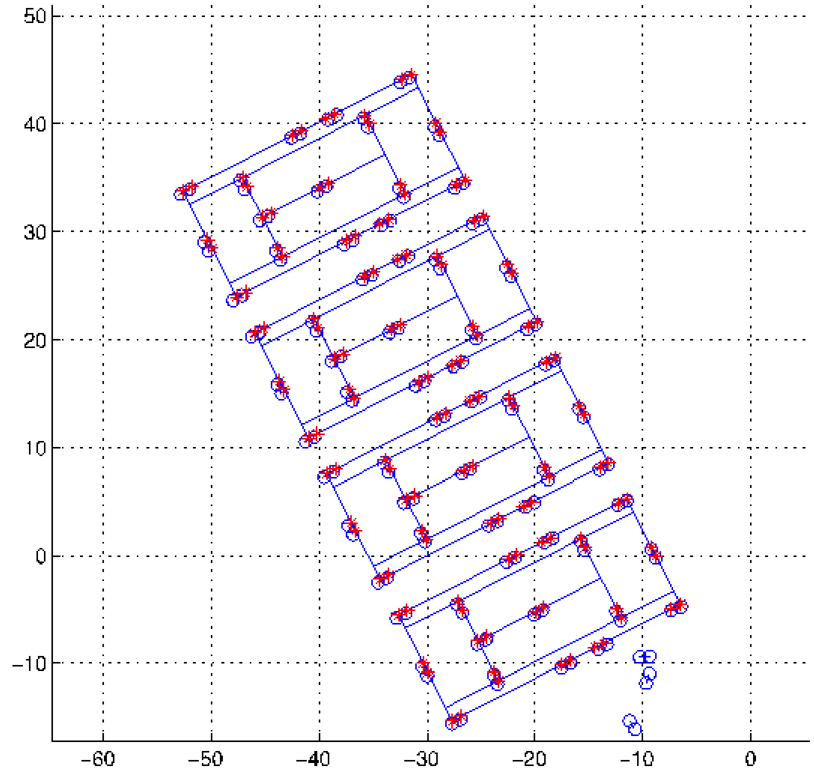


# EKF SLAM Application


[courtesy by John Leonard]

# EKF SLAM Application



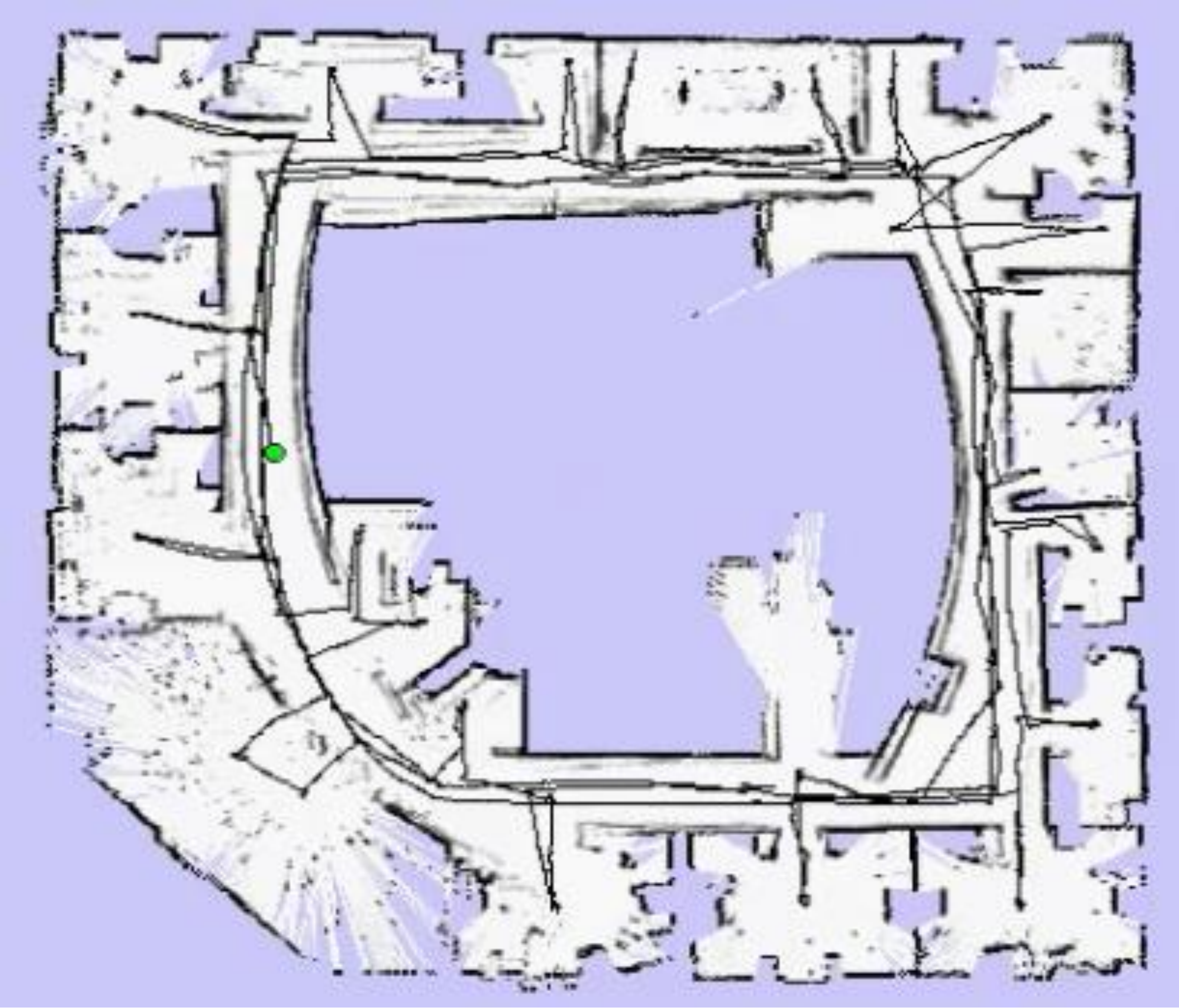




odometry

estimated trajectory

[courtesy by John Leonard]

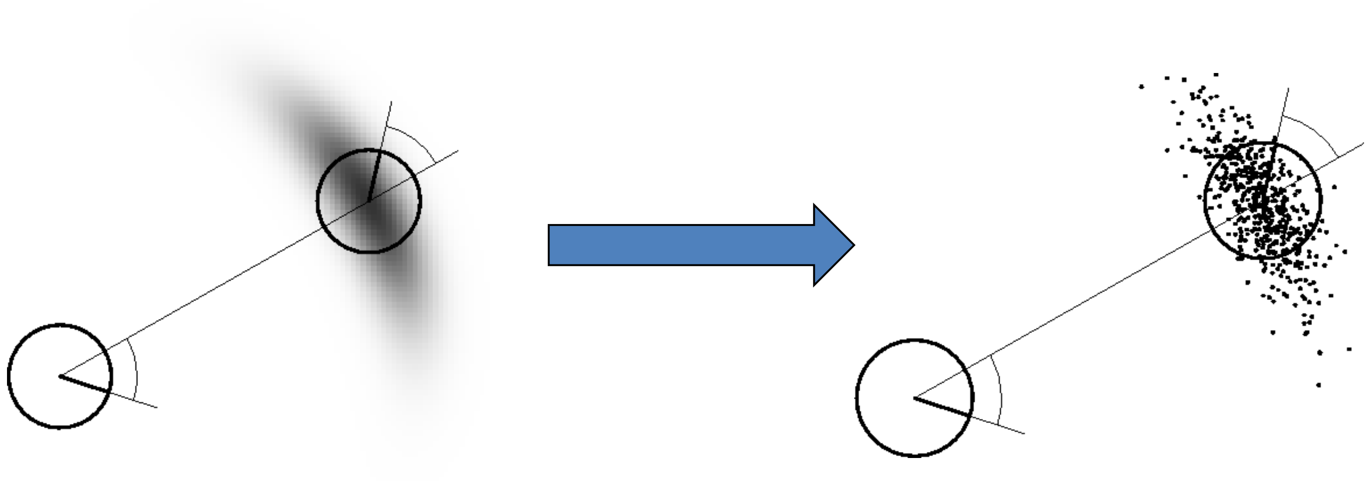
# EKF SLAM Application



# Problem of EKF-SLAM

- Too many linear Gaussian assumptions
- Only Online SLAM

- Solution
  - Sampling based on the probability distribution

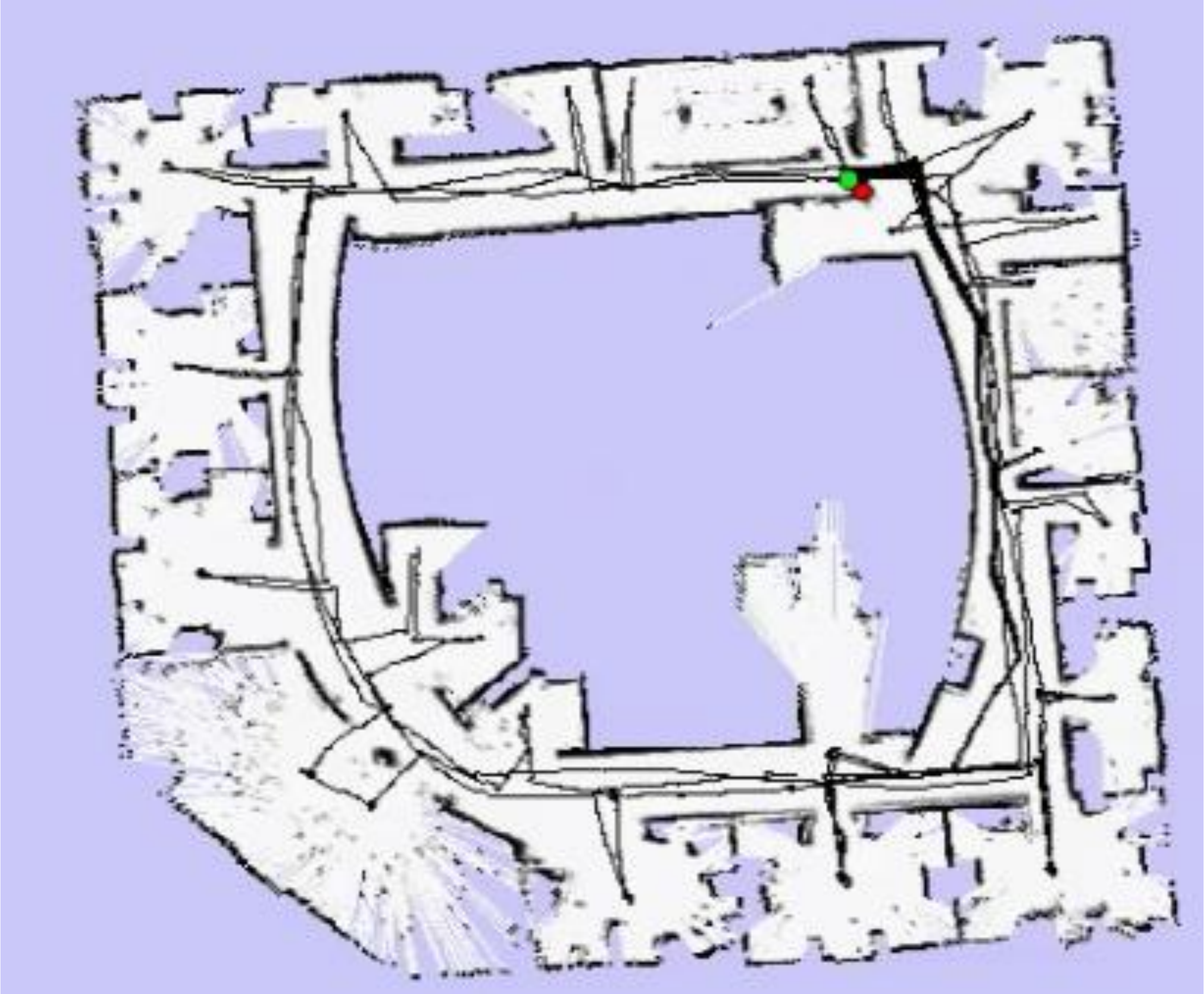


# Particle Filter SLAM

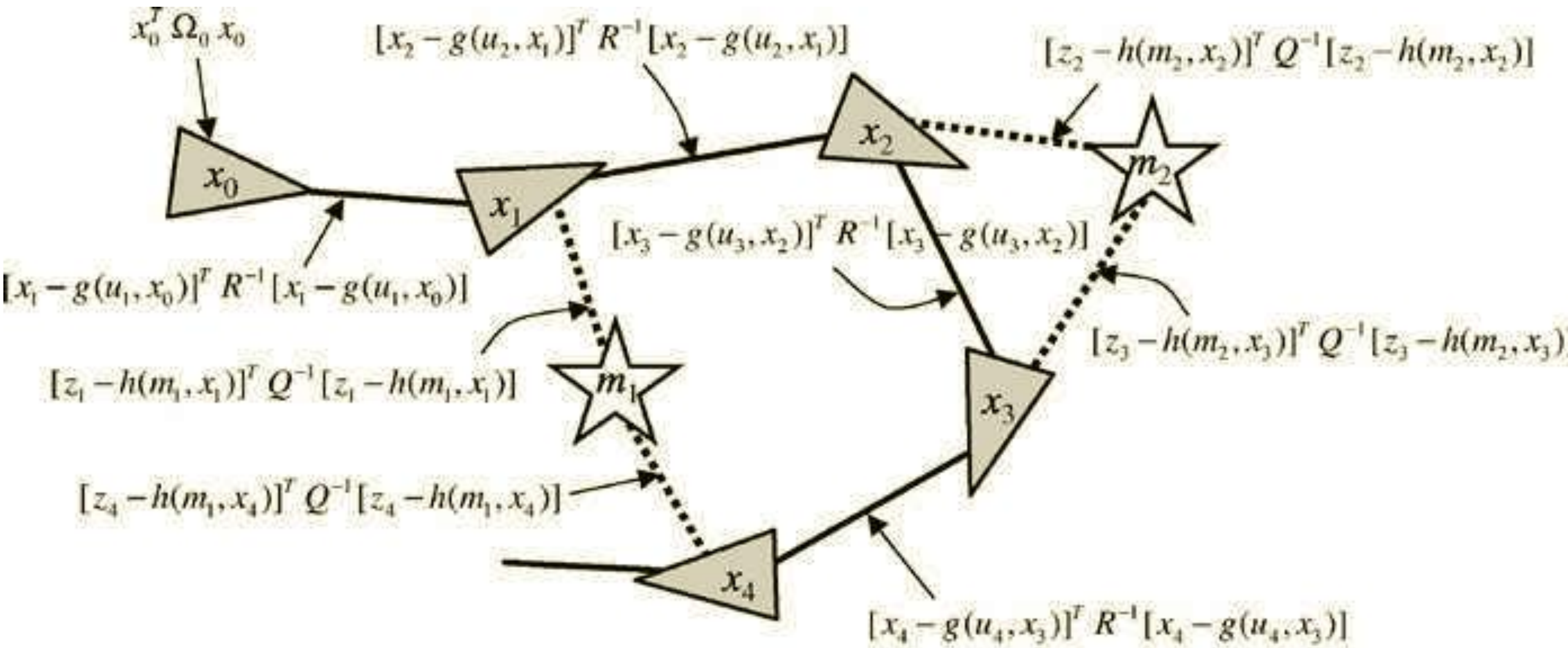
- Rao-Blackwellized particle filtering based on landmarks [Montemerlo et al., 2002]
- Each landmark is represented by a 2x2 Extended Kalman Filter (EKF)
- Each particle therefore has to maintain *M* EKFs

| | | | | | |
|---|---|---|---|---|---|
| **Particle #1** | x, y, $\theta$ | **Landmark 1** | **Landmark 2** | **…** | **Landmark M** |
| **Particle #2** | x, y, $\theta$ | **Landmark 1** | **Landmark 2** | **…** | **Landmark M** |
| **⋮** | | | | | |
| **Particle N** | x, y, $\theta$ | **Landmark 1** | **Landmark 2** | **…** | **Landmark M** |

# Particle Filter SLAM



# Graph SLAM




<u>Sum of all constraints:</u>

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

# Graph SLAM

- For the full SLAM problem

- Minimizing

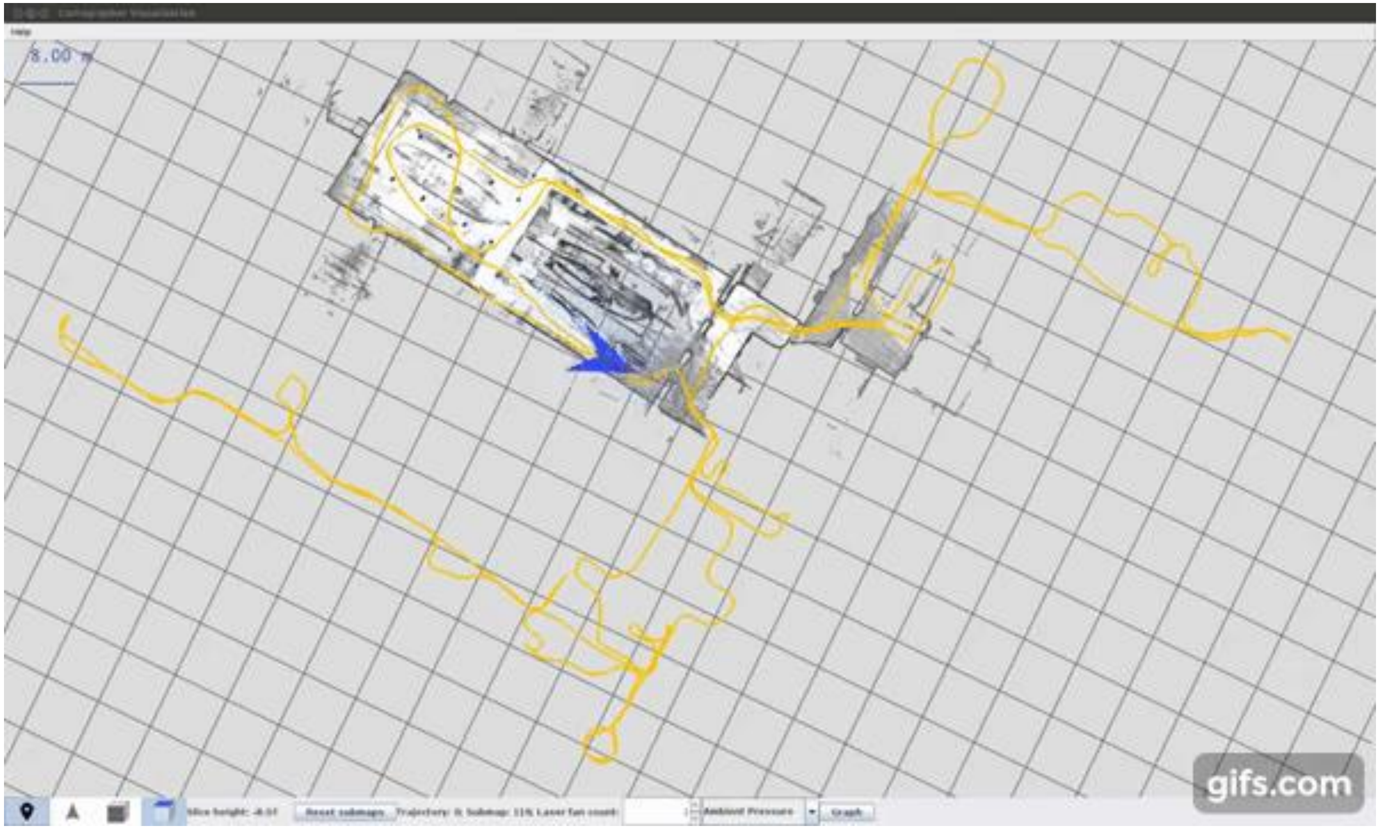
$$
\begin{aligned}
J_{\text{GraphSLAM}} \quad &= \quad x_0^T\, \Omega_0\, x_0 + \sum_t (x_t - g(u_t, x_{t-1}))^T\, R_t^{-1}\, (x_t - g(u_t, x_{t-1})) \\
&\quad + \sum_t \sum_i (z_t^i - h(y_t, c_t^i))^T\, Q_t^{-1}\, (z_t^i - h(y_t, c_t^i))
\end{aligned}
$$

- Easy to build up graph
- Hard to optimize: time consuming



# Open source1: Google Cartographer

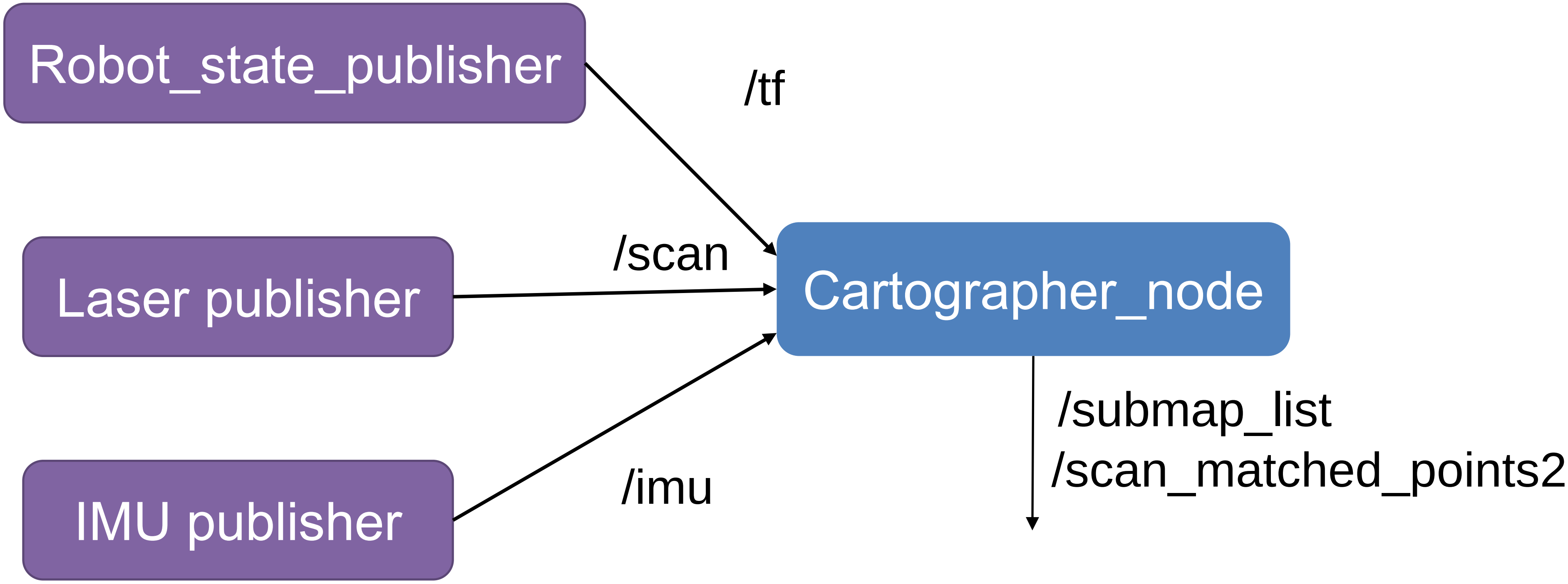
- Indoor 2D or 3D SLAM using LiDAR and IMU
- Provide ROS integration




https://github.com/googlecartographer/

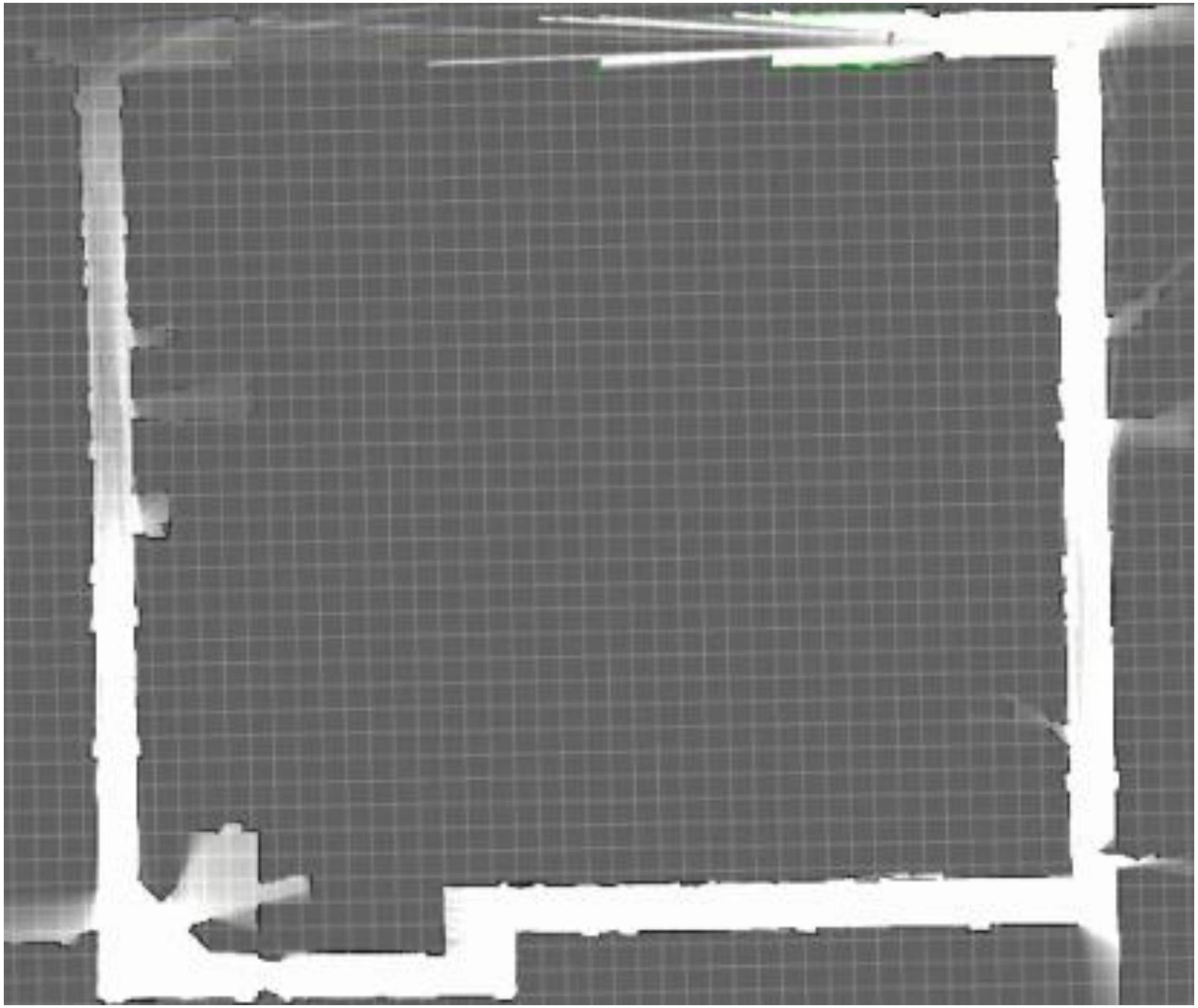
# Usage in ROS

- What we need to provide:
  - /tf: calibration between IMU and LiDAR
  - /scan: LiDAR data
  - /imu: imu data






# Results



# Algorithm

- Prediction
  - Use IMU

- Scan to submap (Online SLAM)
  - Use ***Ceres*** to optimize

$$\underset{\xi}{\operatorname{argmin}} \quad \sum_{k=1}^{K} \left(1 - M_{\text{smooth}}(T_\xi h_k)\right)^2$$

- Loop closure (Global optimization)
  - might be time consuming

$$\underset{\Xi^{\text{m}},\Xi^{\text{s}}}{\operatorname{argmin}} \quad \frac{1}{2}\sum_{ij} \rho\left(E^2(\xi_i^{\text{m}}, \xi_j^{\text{s}}; \Sigma_{ij}, \xi_{ij})\right)$$

W. Hess, D. Kohler, H. Rapp, and D. Andor, ***Real-Time Loop Closure in 2D LIDAR SLAM***, in Robotics and Automation (ICRA), 2016

# Loop Closure

- Search in window $W_x \times W_y \times W_\theta$（7m×7m×30°）

$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, \quad w_y = \left\lceil \frac{W_y}{r} \right\rceil, \quad w_\theta = \left\lceil \frac{W_\theta}{\delta_\theta} \right\rceil$$

- Naive algorithm

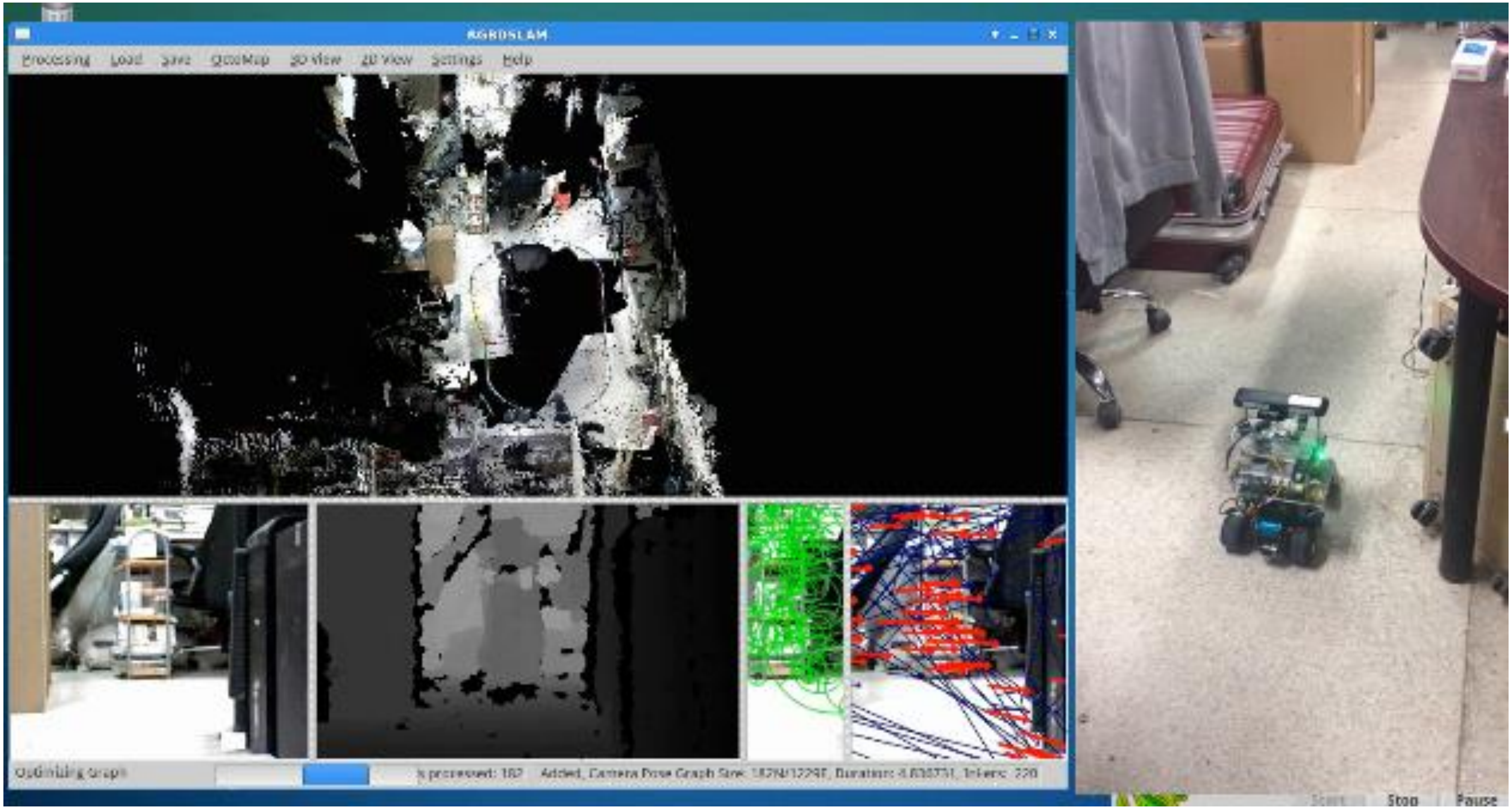
**Algorithm 1** Naive algorithm for (BBS)

$best\_score \leftarrow -\infty$<br>
**for** $j_x = -w_x$ **to** $w_x$ **do**<br>
  **for** $j_y = -w_y$ **to** $w_y$ **do**<br>
    **for** $j_\theta = -w_\theta$ **to** $w_\theta$ **do**<br>
      $score \leftarrow \sum_{k=1}^{K} M_{\text{nearest}}(T_{\xi_0 + (rj_x, rj_y, \delta_\theta j_\theta)} h_k)$<br>
      **if** $score > best\_score$ **then**<br>
        $match \leftarrow \xi_0 + (rj_x, rj_y, \delta_\theta j_\theta)$<br>
        $best\_score \leftarrow score$<br>
      **end if**<br>
    **end for**<br>
  **end for**<br>
**end for**<br>
**return** $best\_score$ and $match$ when set.

**Branch-and-Bound**
**分支定界**



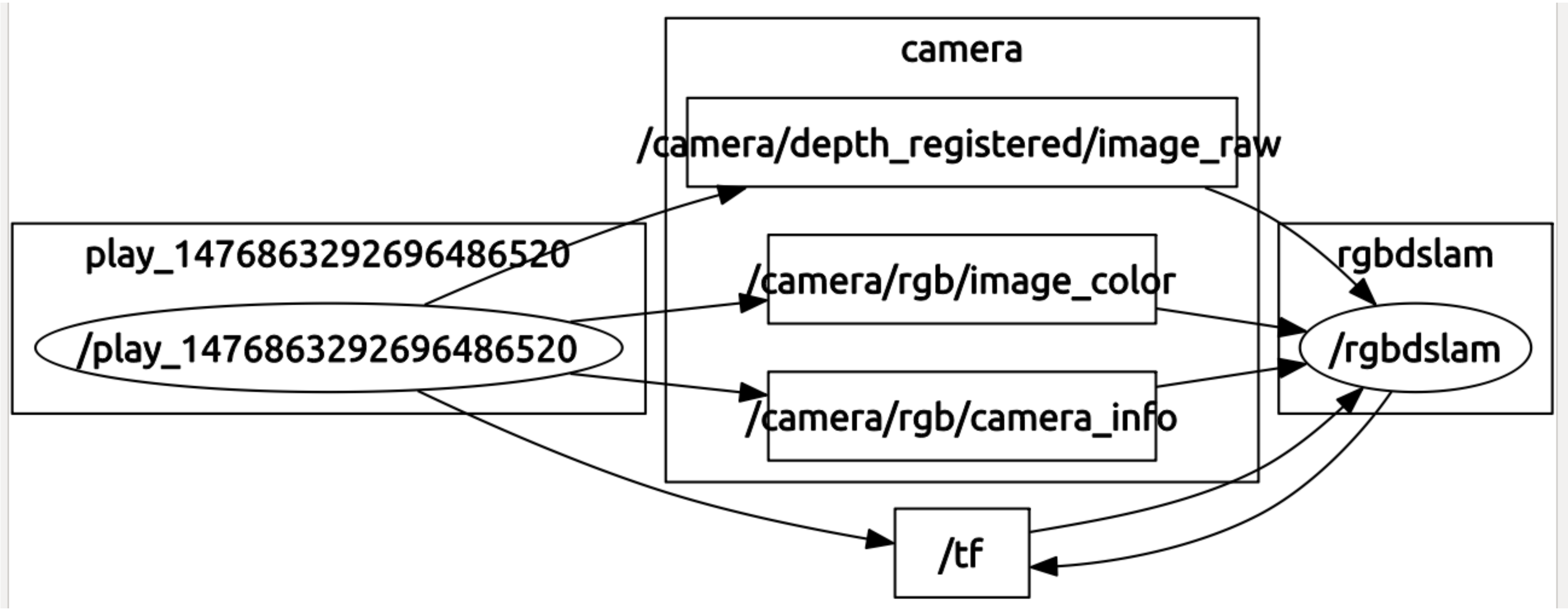
# Open Source2: RGBD-SLAM



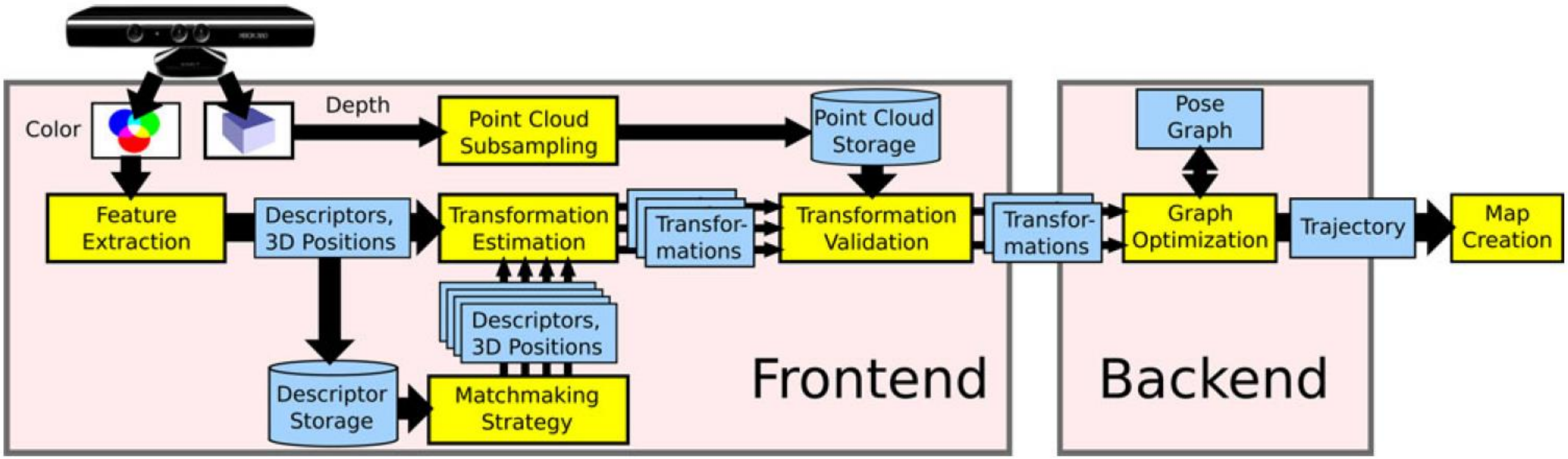

https://github.com/felixendres/rgbdslam_v2

# Usage in ROS

- What we need to provide:
  - /tf: calibration between IMU and RGBD camera + IMU data
  - /depth_registered/image_raw: depth image
  - /rgb/image_color: RGB image
  - /rgb/camera_info: camera calibration



# Algorithm




Endres F, Hess J, Sturm J, et al.
***3-D Mapping With an RGB-D Camera[J].***
Robotics IEEE Transactions on, 2014, 30(1):177-187.

# Problem for SLAM

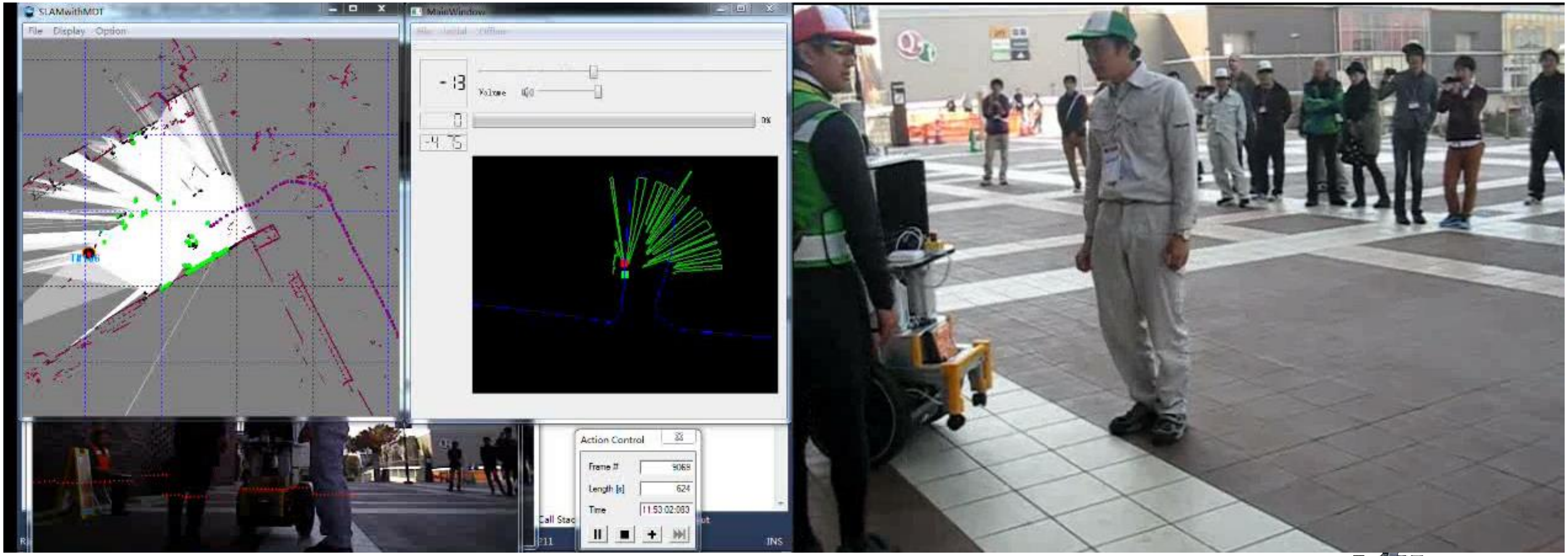
- Landmark correspondence
  - Visual based: feature matching (descriptor distance)
  - Laser based: nearest matching (like ICP, PL-ICP)

- Moving objects
  - Not included in the whole system
  - Need to add a object detection module
    - Directly detect and tracking moving objects
    - Treat them as outlier (eg. RANSAC)
  - SLAM-with-MODT



# What we have done in tsukuba chanllenge

- Open area
- Pre-built map using LiDAR based SLAM (with little manual correction)
- Online map-based localization with moving object detection
- Special people detection





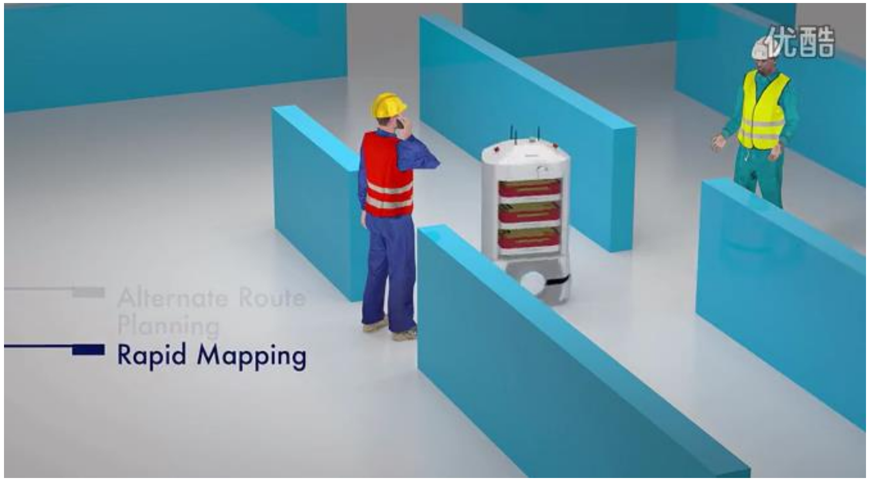

# Applications of SLAM

- Key: explore and mapping
  - Mine field explore

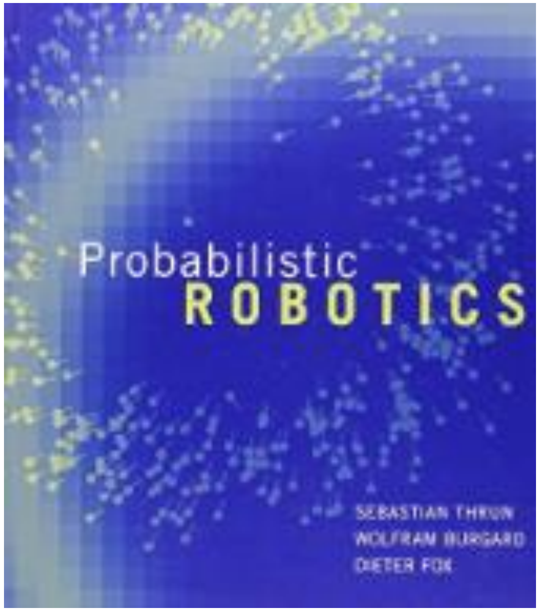
  - Sweeping robots

# Applications of SLAM

- Agv (Automated Guided Vehicle) [might be map-based localization]








# 谢谢！

特别感谢高飚、徐东昊帮我一起准备素材

强烈推荐《概率机器人》，部分视频和材料引自该书和网站
http://robots.stanford.edu/probabilistic-robotics/

本次直播由极视角组织。

# 极市平台

（极视角旗下产品）

最专业的视觉算法开发与分发平台

做好算法的同时，赚取一个亿

每周定期分享请关注