# ME 599/699 Robot Modeling & Control
# Fall 2021

## Simultaneous Localization and Mapping

Hasan A. Poonawala

Department of Mechanical Engineering
University of Kentucky

Email: hasan.poonawala@uky.edu
Web: https://www.engr.uky.edu/~hap

# Simultaneous Localization and Mapping

A robot is typically unable to measure its position.

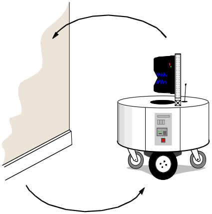
Instead, it can measure its position in relation to objects in the world.

Can't use KF directly to estimate location when we don't have a map of objects in the the world.

Without a map, we don't know what to expect as measurement in a state (no sensor model).

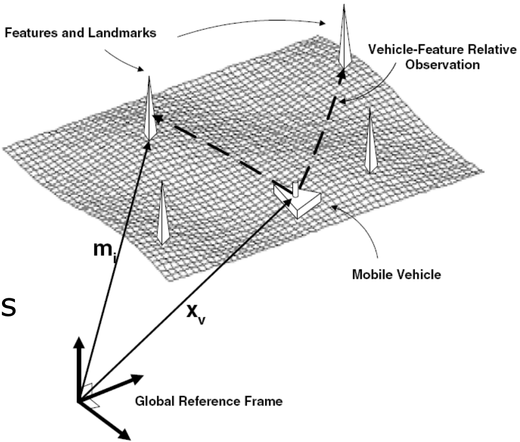
# The SLAM Problem

- SLAM is a **chicken-or-egg** problem:
  - → a map is needed for localization and
  - → a pose estimate is needed for mapping



# Feature-Based SLAM

- **Absolute** robot poses
- **Absolute** landmark positions
- But only **relative** measurements of landmarks



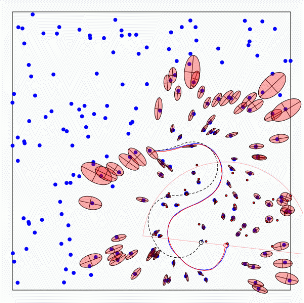
# Feature-Based SLAM

## Given:

- The robot's controls
  $$U_{1:k} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k\}$$
- Relative observations
  $$Z_{1:k} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_k\}$$

## Wanted:

- Map of features
  $$\boldsymbol{m} = \{\boldsymbol{m}_1, \boldsymbol{m}_2, \ldots, \boldsymbol{m}_n\}$$
- Path of the robot
  $$\boldsymbol{X}_{1:k} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k\}$$



# Simultaneous Localization and Mapping

SLAM tries to simultaneously make sense of where we think we are (localization) and what we think we should be seeing (mapping).

Basic idea: Make the map a part of the state.

We can build measurement and motion models for this expanded state.

# Example: 2D SLAM Non-rotating robot

The robot at $(x_r, y_r)$ measures the location of landmarks $l_i = (l_{ix}, l_{iy})$ in its frame of reference, which is aligned with the world axis.

State is $x = \begin{bmatrix} x_r \\ y_r \\ l_{1x} \\ l_{1y} \\ l_{2x} \\ l_{2y} \\ \vdots \\ l_{n_l x} \\ l_{n_l y} \end{bmatrix}$ where there are $n_l$ landmarks.

# Example: 2D SLAM Non-rotating robot

The landmarks are assumed stationary, with robot moving according to

$$\begin{bmatrix} x_r \\ y_r \end{bmatrix}_{t+1} = \begin{bmatrix} x_r \\ y_r \end{bmatrix}_t + u_t + v_t$$

Therefore, $A$ is $(2 + 2n_l) \times (2 + 2n_l)$, the only non-zero elements being

$$A_{1,1} = 1, \quad A_{2,2} = 1$$

.

$B$ is $(2 + 2n_l) \times 2$, the only non-zero elements being

$$B_{1,1} = 1, \quad B_{2,2} = 1$$

.

# Example: 2D SLAM Non-rotating robot

If the robot sees all landmarks in its frame, its measurement is

$$y = \begin{bmatrix} l_{1x} - x_r \\ l_{1y} - y_r \\ l_{2x} - x_r \\ l_{2y} - y_r \\ \vdots \end{bmatrix}$$

Therefore, $C$ is $2n_l \times (2 + 2n_l)$:

$$C = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & \dots & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & & & & & \vdots & & \\ -1 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

# Example: 2D SLAM Non-rotating robot

Performance when robot always sees all landmarks (Run `Julia` code on Canvas):

- The estimated mean position of each landmark is quite close to the actual landmark position, relative to other landmarks
- The uncertainty in the landmark position is identical to the uncertainty in initial position, in the long run
  - The robot can treat the landmark as lying within the $3\sigma$ ellipse of the landmark mean. To avoid collision with high probability, the $3\sigma$ ellipse corresponding to its position should lie outside the landmark's ellipse.
- The difference between the robot's true and estimated location depends on sensor accuracy

# Extended Kalman Filter

For linear systems with gaussian additive noise, the Kalman filter provides the least uncertain estimate.
When the system is nonlinear:

$$x_{t+1} = f(x_t, u_t) + v_t \tag{1}$$
$$y_t = h(x_t) + w_t, \tag{2}$$

use linearization:

$$A_t = \left.\frac{\partial f}{\partial x}\right|_{x=\mu_t}, \quad B_t = \left.\frac{\partial f}{\partial u}\right|_{u=u_t}, \quad C_t = \left.\frac{\partial h}{\partial x}\right|_{x=\mu_t^{pred}}.$$

A blind application of Kalman filter updates using $A_t$, $B_t$, and $C_t$ is not guaranteed to work but often does well in practice.

# Example: 2D SLAM Rotating robot

The inputs are the speed $s$ of the robot along its current heading direction, and the angular velocity $\omega$.

$$u_t = \begin{bmatrix} s_t \\ \omega_t \end{bmatrix}$$

The motion model is now nonlinear:

$$\begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_{t+1} = \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_t + \begin{bmatrix} s_t \cos\theta_r \\ s_t \sin\theta_r \\ \omega_t \end{bmatrix} + v_t = f(x_t, u_t)$$

We linearize to get part of the $A$ and $B$ matrices

$$\begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 & -\sin\theta_r s_t \\ 0 & 1 & -\cos\theta_r s_t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ \theta_r \end{bmatrix}_t + \begin{bmatrix} \cos\theta_r & 0 \\ \sin\theta_r & 0 \\ 0 & 1 \end{bmatrix} u_t + v_t$$

# Example: 2D SLAM Rotating robot

The position of a landmark $l_i$ in the robot's frame becomes

$$y_i = \begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix} \begin{bmatrix} l_{ix} - x_r \\ l_{iy} - y_r \end{bmatrix}.$$

The full measurement $y$ corresponds to stacking the $y_i$s.

The measurement model is nonlinear: $y = h(x) \neq Cx$ for any matrix $C$.

As mentioned earlier, we linearize

$$C = \left.\frac{\partial h}{\partial x}\right|_{x=\mu_t^{pred}}$$

# Limited Sensing

We've assumed that we can see all landmarks at once.

In practice, we can see nearby landmarks.

This limitation creates a data association problem when landmarks look similar to each other. For example, the entry to an office cubicle.

# EKF SLAM: Building the Map

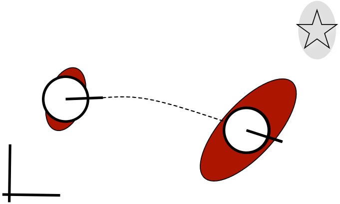
**Filter Cycle**, Overview:

1. State prediction (odometry)
2. Measurement prediction
3. Observation
4. Data Association 

5. Update
6. Integration of new landmarks

# EKF SLAM: Building the Map

- State Prediction



Odometry:

$$\hat{\mathbf{x}}_R = f(\mathbf{x}_R, \mathbf{u})$$

$$\hat{C}_R = F_x \, C_R \, F_x^T + F_u \, U \, F_u^T$$

Robot-landmark cross-covariance prediction:

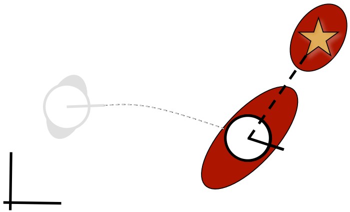
$$\hat{C}_{RM_i} = F_x \, C_{RM_i}$$

(skipping time index $k$)

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

# EKF SLAM: Building the Map

- Measurement Prediction



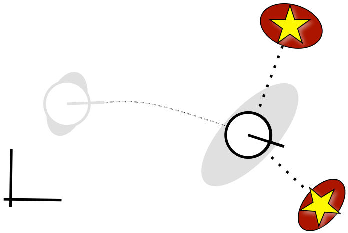
Global-to-local frame transform $h$

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

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

# EKF SLAM: Building the Map

- Observation



*(x,y)*-point landmarks

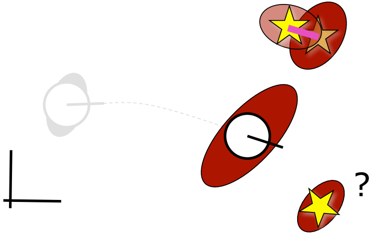
$$\mathbf{z}_k = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix}$$

$$R_k = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

# EKF SLAM: Building the Map

- Data Association



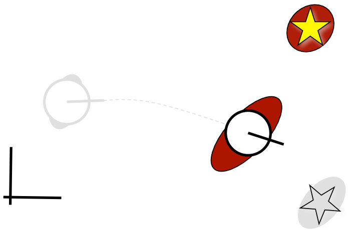

Associates predicted measurements $\hat{\mathbf{z}}_k^i$ with observation $\mathbf{z}_k^j$

$$\begin{aligned} \nu_k^{ij} &= \mathbf{z}_k^j - \hat{\mathbf{z}}_k^i \\ S_k^{ij} &= R_k^j + H^i \, \hat{C}_k \, H^{i\,T} \end{aligned}$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

# EKF SLAM: Building the Map

- Filter Update



The usual Kalman filter expressions

$$K_k = \hat{C}_k \, H^T S_k^{-1}$$

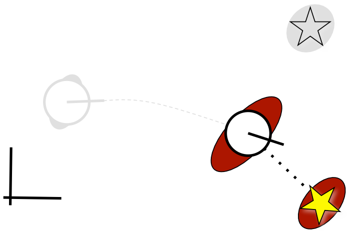
$$\mathbf{x}_k = \hat{\mathbf{x}}_k + K_k \, \nu_k$$

$$C_k = (I - K_k \, H) \, \hat{C}_k$$

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} \\ C_{M_2R} & C_{M_2M_1} & C_{M_2} & \cdots & C_{M_2M_n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{M_nR} & C_{M_nM_1} & C_{M_nM_2} & \cdots & C_{M_n} \end{bmatrix}_k$$

# EKF SLAM: Building the Map

- Integrating New Landmarks



State augmented by

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

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

Cross-covariances:

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

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

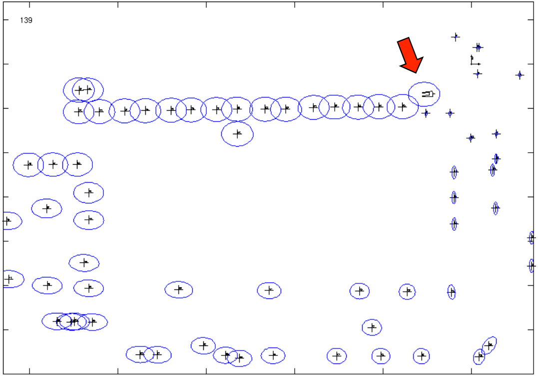
$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_n \\ \mathbf{m}_{n+1} \end{bmatrix}_k \qquad C_k = \begin{bmatrix} C_R & C_{RM_1} & C_{RM_2} & \cdots & C_{RM_n} & C_{RM_{n+1}} \\ C_{M_1R} & C_{M_1} & C_{M_1M_2} & \cdots & C_{M_1M_n} & C_{M_1M_{n+1}}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

# SLAM: Loop Closure

- **Recognizing an already mapped area**, typically after a long exploration path (the robot "closes a loop")
- Structurally identical to data association, but
  - high levels of ambiguity
  - possibly useless validation gates
  - environment symmetries
- Uncertainties **collapse** after a loop closure (whether the closure was correct or not)

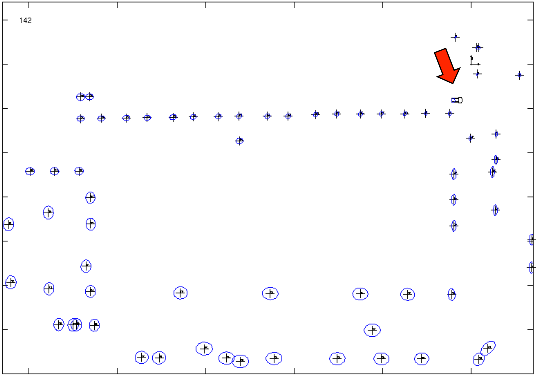
# SLAM: Loop Closure

- Before loop closure



# SLAM: Loop Closure

- After loop closure



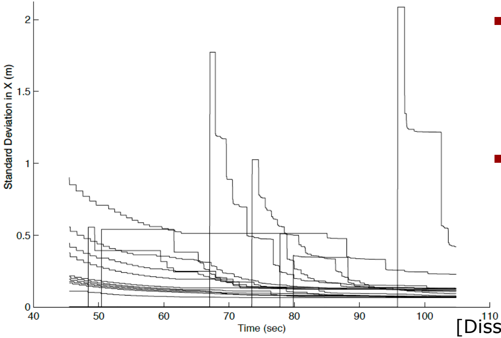
# SLAM: Loop Closure

- By revisiting already mapped areas, uncertainties in robot and landmark estimates can be **reduced**
- This can be exploited when **exploring** an environment for the sake of better (e.g. more accurate) maps
- Exploration: the problem of ***where* to acquire new information**

→ See separate chapter on exploration

# KF-SLAM Properties (Linear Case)

- The **determinant** of any sub-matrix of the map covariance matrix **decreases monotonically** as successive observations are made

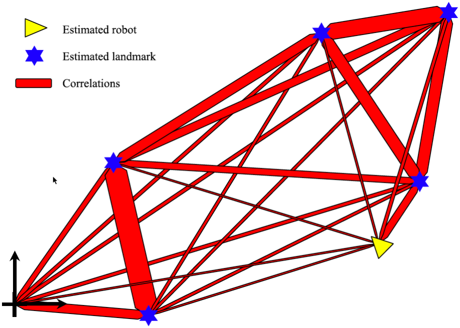



- When a new land-mark is initialized, its **uncertainty** is **maximal**
- Landmark uncer-tainty **decreases monotonically** with each new observation

[Dissanayake et al., 2001]

# KF-SLAM Properties (Linear Case)

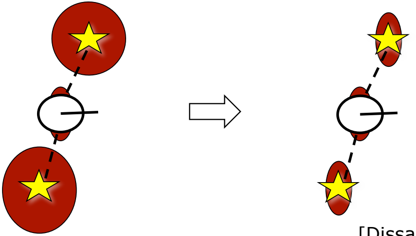
- In the limit, the landmark estimates become **fully correlated**




[Dissanayake et al., 2001]

# KF-SLAM Properties (Linear Case)

- In the limit, the **covariance** associated with any single landmark location estimate is determined only by the **initial covariance in the vehicle location estimate**.



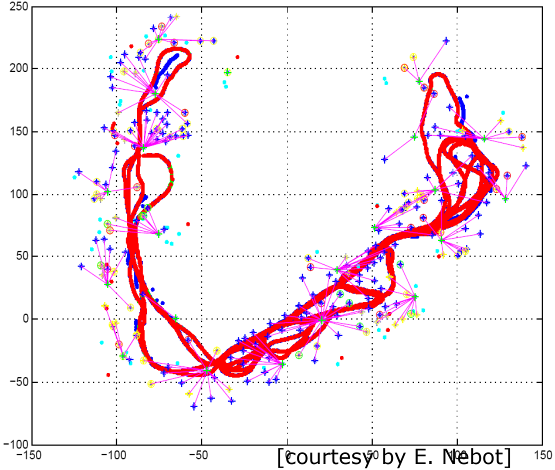
[Dissanayake et al., 2001]

# EKF SLAM Example:
# Victoria Park Dataset

# Victoria Park: Data Acquisition

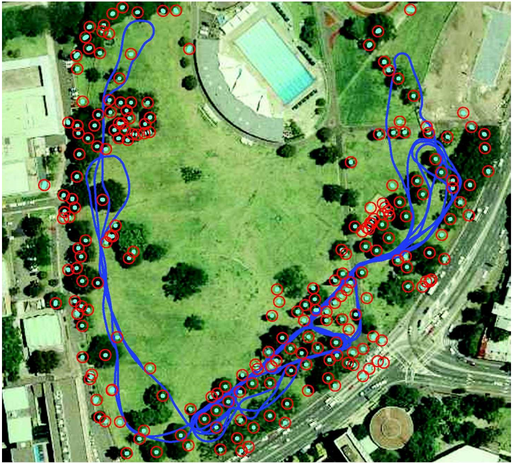
[courtesy by E. Nebot]

# Victoria Park: Estimated Trajectory




[courtesy by E. Nebot]

# Victoria Park: Landmarks

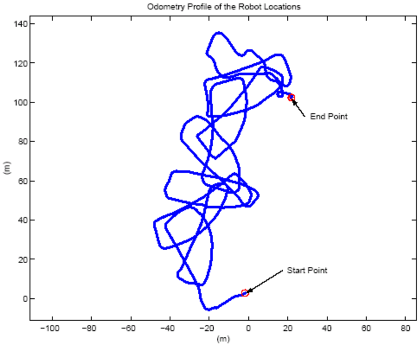


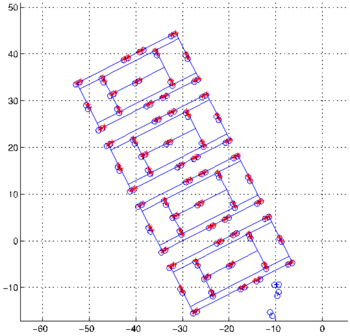
[courtesy by E. Nebot]

# EKF SLAM Example: Tennis Court


[courtesy by J. Leonard]

# EKF SLAM Example: Tennis Court



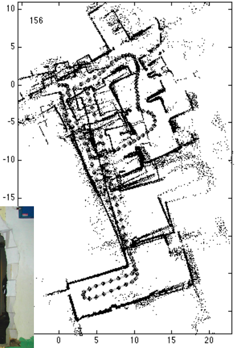

odometry

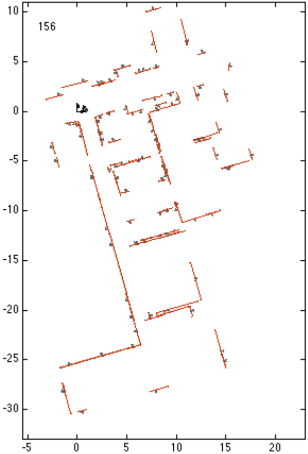
estimated trajectory

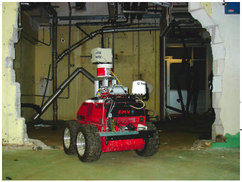
[courtesy by John Leonard]

# EKF SLAM Example: Line Features

- KTH Bakery Data Set



[Wulf et al., ICRA 04]

# EKF-SLAM: 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 consumption: $O(n^2)$
- Problem: becomes computationally intractable for large maps!
- There exists variants to circumvent these problems

# SLAM Techniques

- EKF SLAM
- FastSLAM
- 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, Sparse links, Thin junction tree filters, etc.
- …

# EKF-SLAM: Summary

- The first SLAM solution
- Convergence proof for linear Gaussian case
- Can diverge if nonlinearities are large (and the reality is nonlinear...)
- Can deal only with a single mode
- Successful in medium-scale scenes
- Approximations exists to reduce the computational complexity