

# **Cognitive Robotics**

---

## **08. Simultaneous Localization and Mapping (SLAM)**

---

AbdElMoniem Bayoumi, PhD

Fall 2022

# Acknowledgment

- These slides have been created by Wolfram Burgard, Dieter Fox, Cyrill Stachniss and Maren Bennewitz

# Recap: Mapping so far

- Mapping with **known poses** for a grid map representation (easy)
- **Occupancy grids**: each cell is a binary random variable estimating whether the cell is occupied
- Static state binary Bayes filter per cell
- **Reflection Maps**: store in each cell the probability that a beam is reflected by this cell
- Given the discussed sensor model, counting yields the maximum likelihood model

# Difference between Occupancy Grid Maps and Reflection Maps

- The **counting** model determines how often a cell reflects a beam
- The **occupancy** model represents whether or
  - not a cell is occupied by an object
- Although a cell might be occupied by an object, the **reflection probability of this object might be very small**

# Recap: Example Occupancy Map

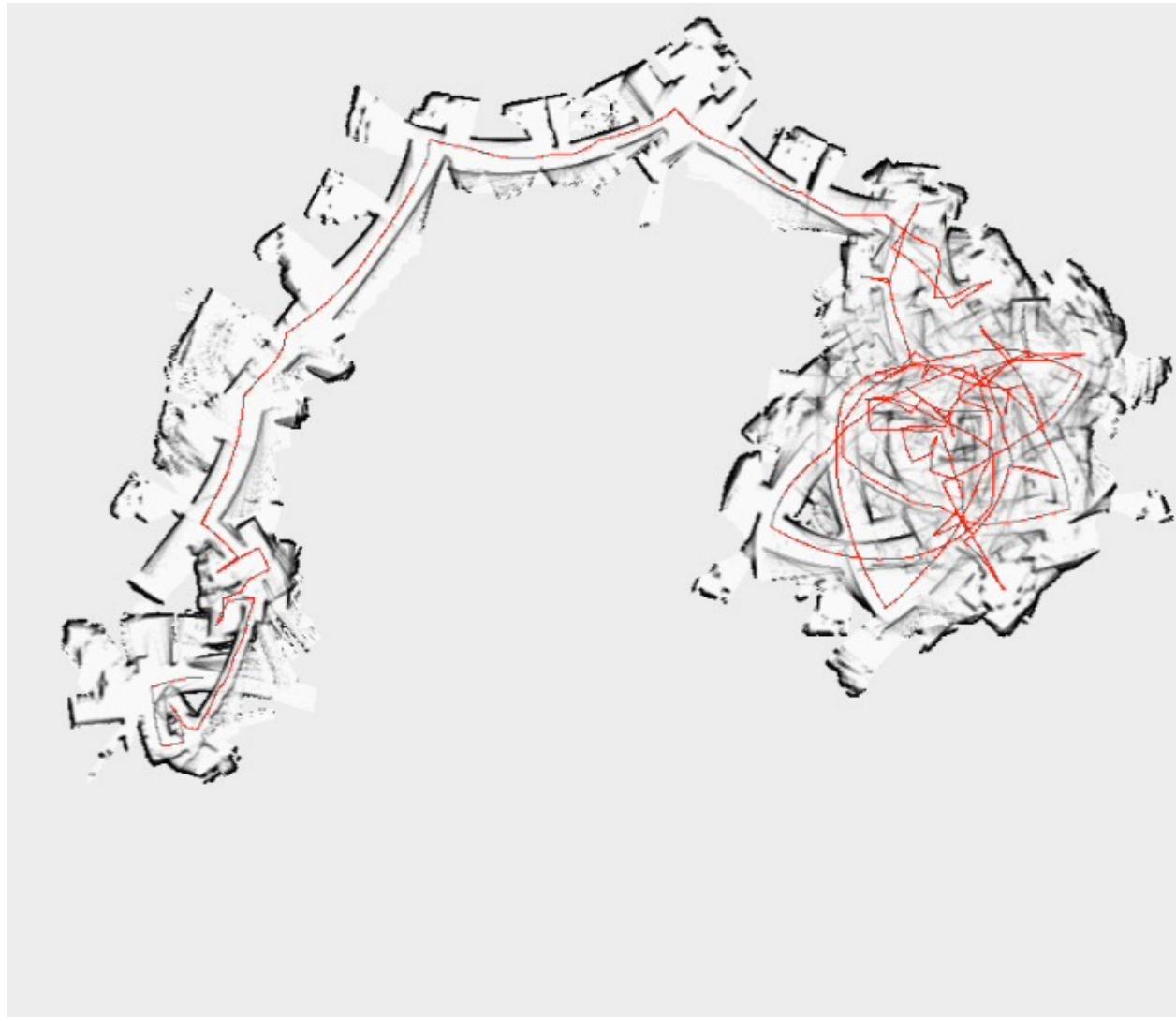


# Recap: Example Reflection Map



# **Grid Mapping Meets Reality...**

# Mapping With Raw Odometry



Courtesy: D. Hähnel

# Possible Solution: Incremental Scan Alignment

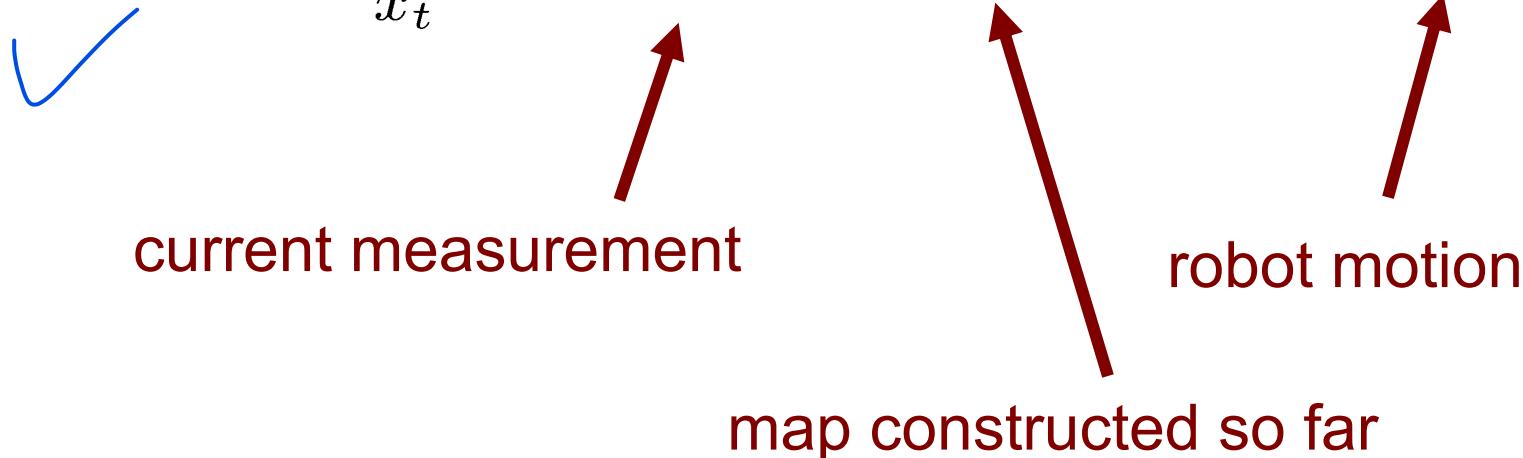
- Motion is noisy, we cannot ignore it
- In reality, the robot poses are not known
- Often, the sensor is rather precise (laser)
- Scan matching: **incrementally align two scans or a scan to a map**

# Pose Correction Using Scan Matching

Maximize the likelihood of the **current** pose relative to the **previous** pose and map

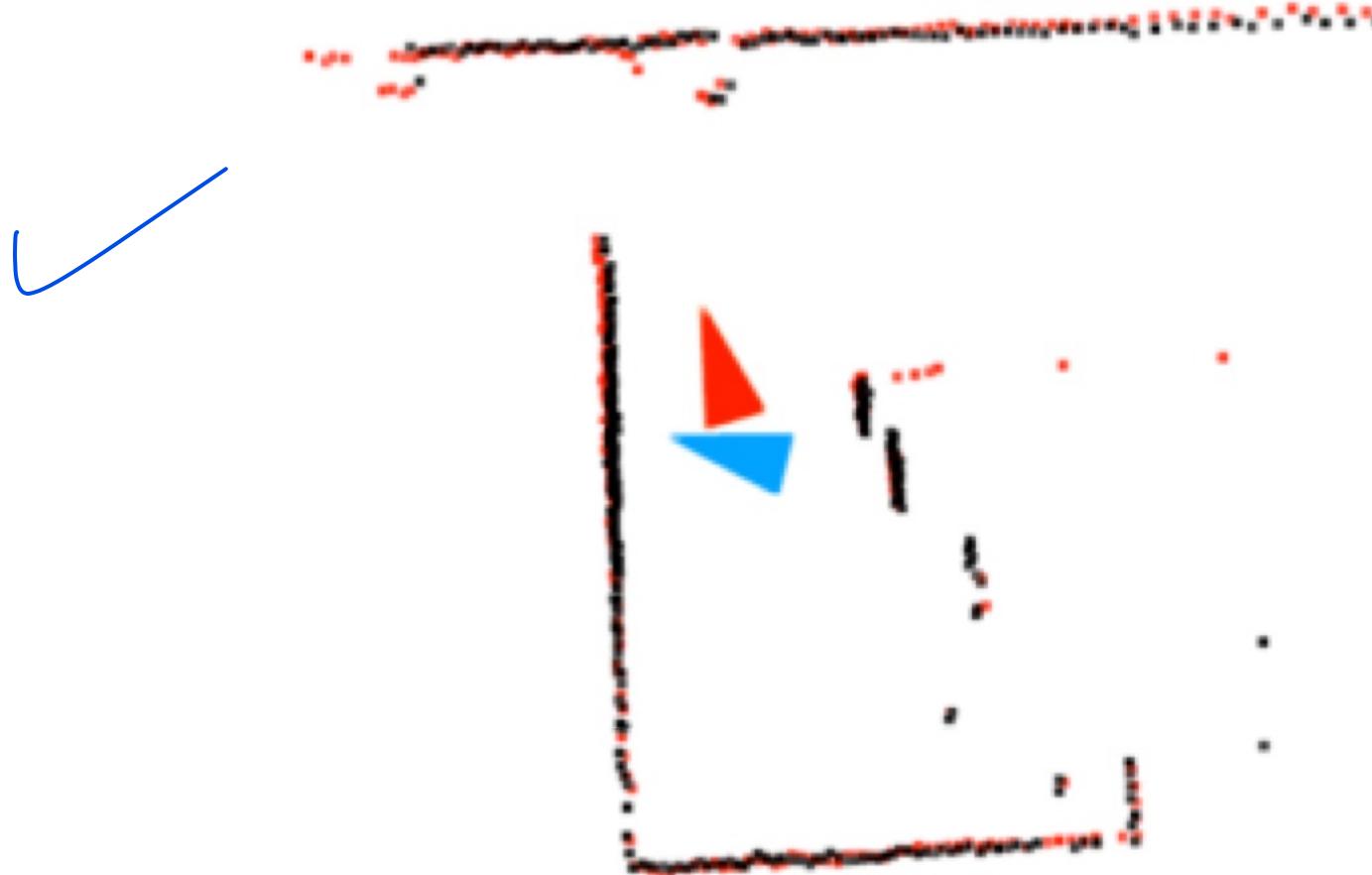
h5tar mkan el motion elly yemshy m3 erayt el sensor.

$$x_t^* = \operatorname{argmax}_{x_t} \left\{ p(z_t | x_t, m_{t-1}) p(x_t | u_{t-1}, x_{t-1}^*) \right\}$$



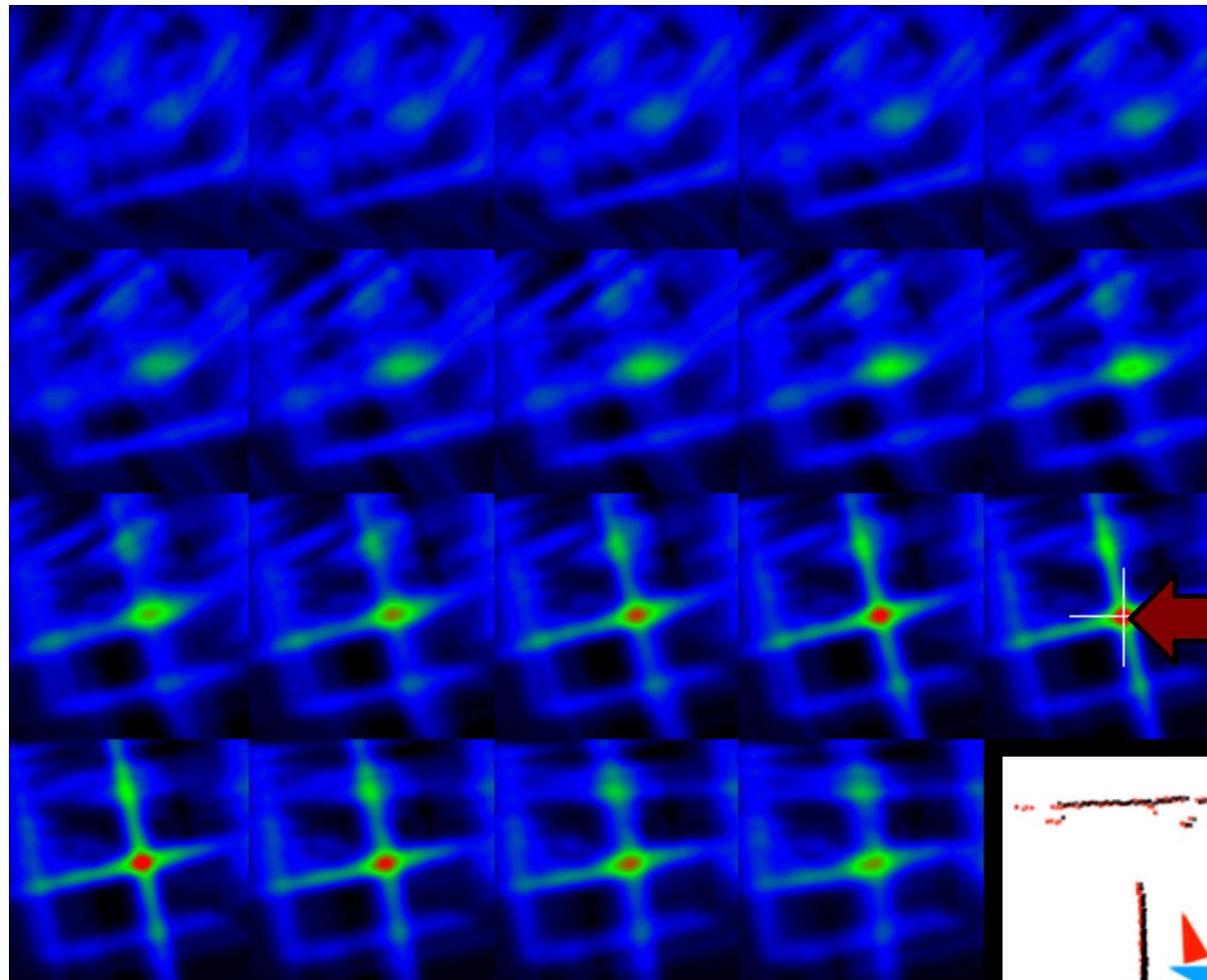
# Incremental Alignment

---



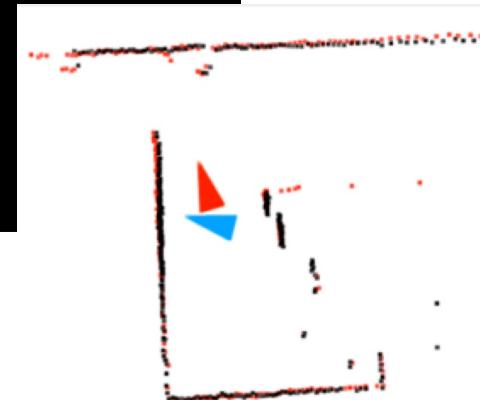
Courtesy: E. Olson

# Incremental Alignment



each tile shows the likelihood for the x/y translation for a fixed orientation

Courtesy: E. Olson

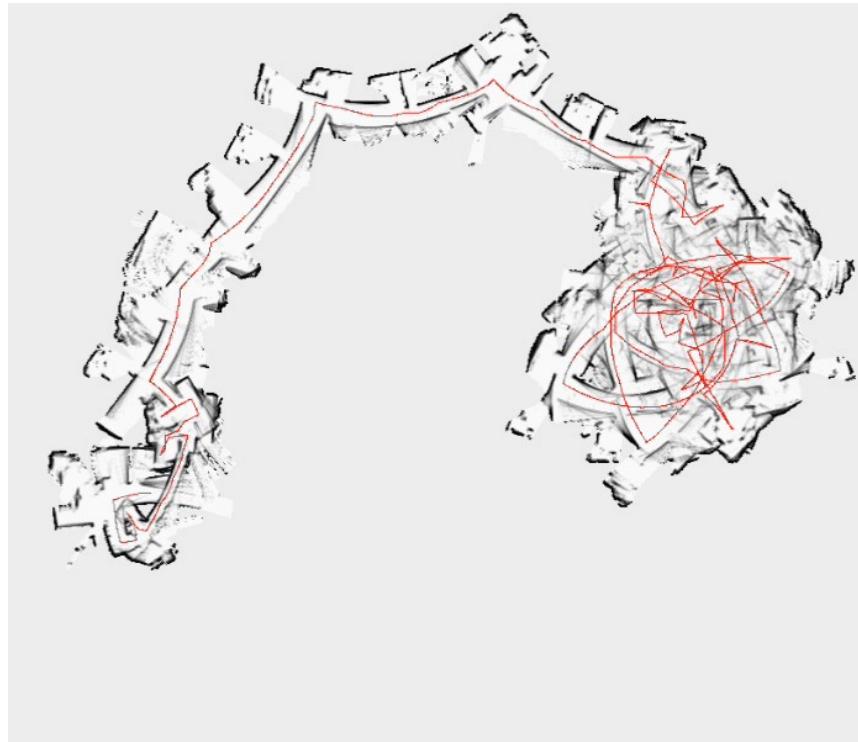


# Various Different Ways to Realize Scan Matching

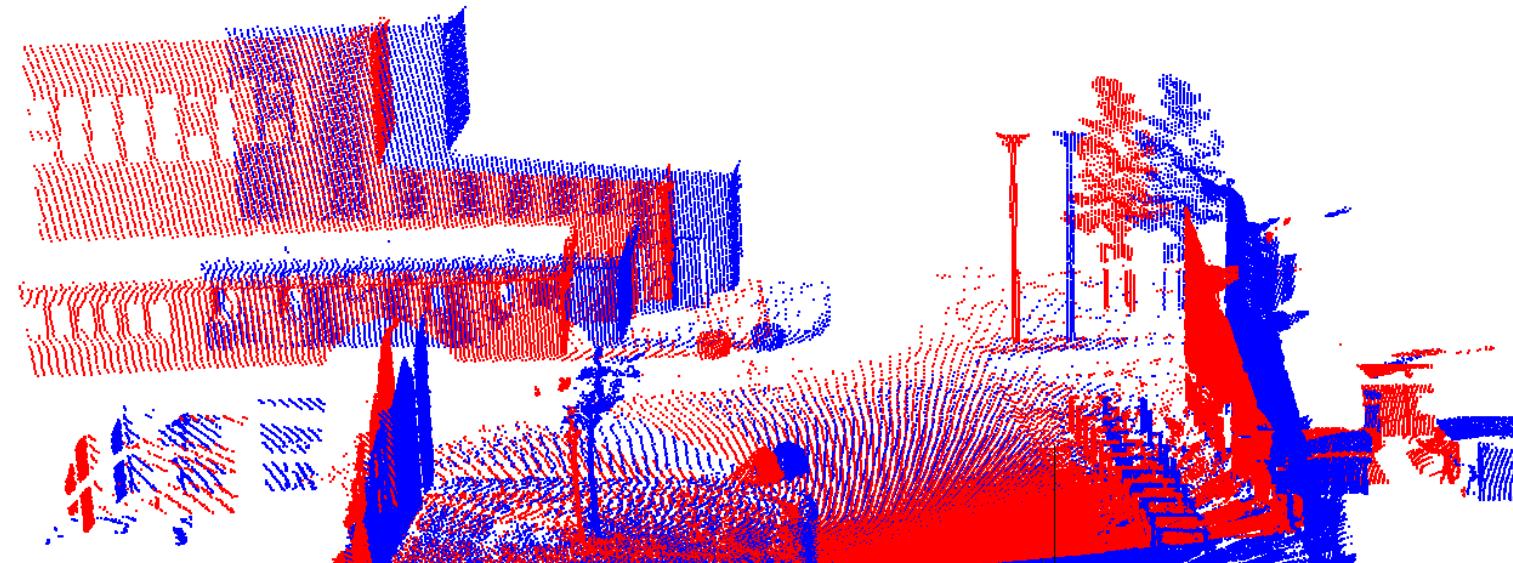
- Scan-to-scan
- Scan-to-map 
- Map-to-map
- Iterative closest point (ICP)
- Feature-based
- ...

# With and Without Scan Matching

hena el map andaf kter, bs lesa feha mshakel/

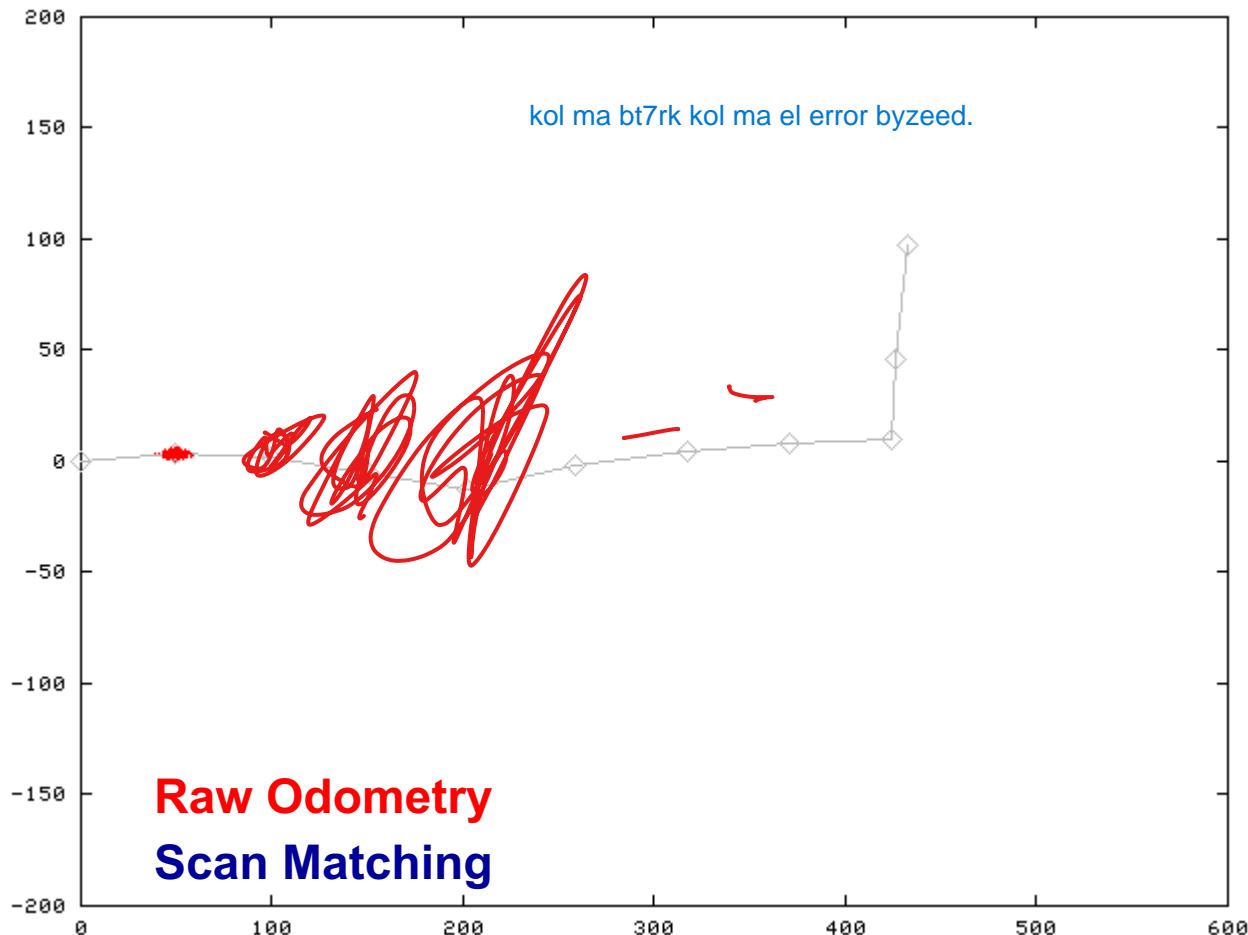


# Example: Aligning in 3D



Courtesy: P. Pfaff

# Motion Model for Scan Matching



# Summary: Scan Matching

- Scan matching improves the pose estimate (and thus mapping) substantially
- Locally consistent estimates
- But: Often scan matching is not sufficient to build large consistent maps



**SLAM**

# What is SLAM?

- Estimate the pose of a robot and the map of the environment at the same time
- SLAM is hard, because
  - a map is needed for localization and
  - a good pose estimate is needed for mapping

# What is SLAM?

- Localization: inferring the robot's location within a given map
- Mapping: inferring a map given sensor data from known robot locations
- **SLAM: learning a map and locating the robot simultaneously**

# SLAM Applications

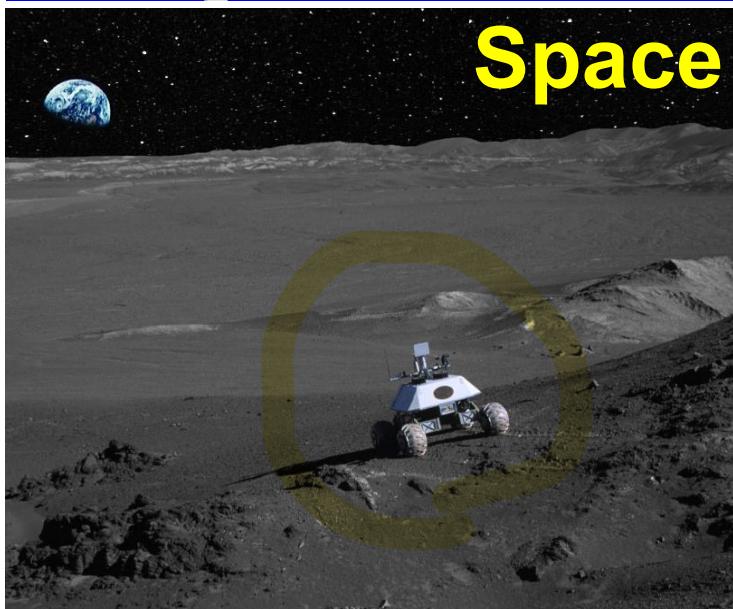
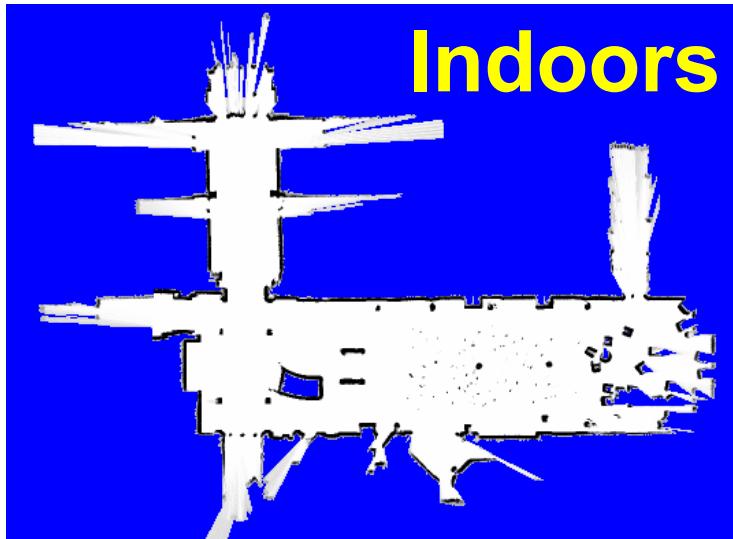
- SLAM is central to a range of indoor, outdoor, in-air, and underwater applications

## Examples:

- At home: vacuum cleaner, lawn mower
- Surveillance with unmanned air vehicles
- Underwater: reef monitoring
- Underground: exploration of mines
- Space: terrain mapping

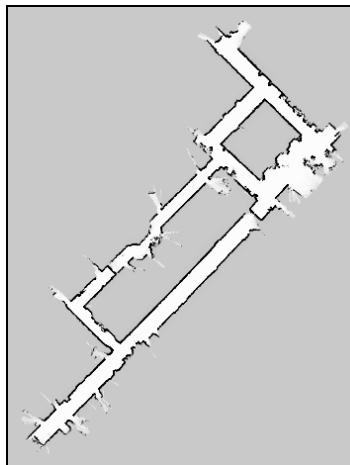
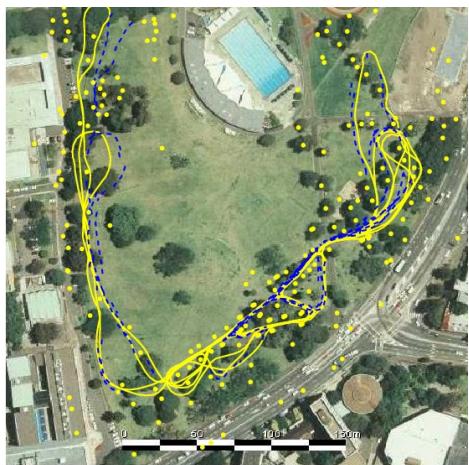


# SLAM Applications



# Typical Map Representations

Landmark-based or grid-based (2D or 3D) representations of the environment



# The SLAM Problem

- SLAM is considered a fundamental problem for robots to become truly autonomous
- Large variety of SLAM approaches have been developed
- The majority uses probabilistic concepts
- History of SLAM dates back to the mid-eighties

# Definition of the SLAM Problem

## Given

- The robot's controls

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

- Observations

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

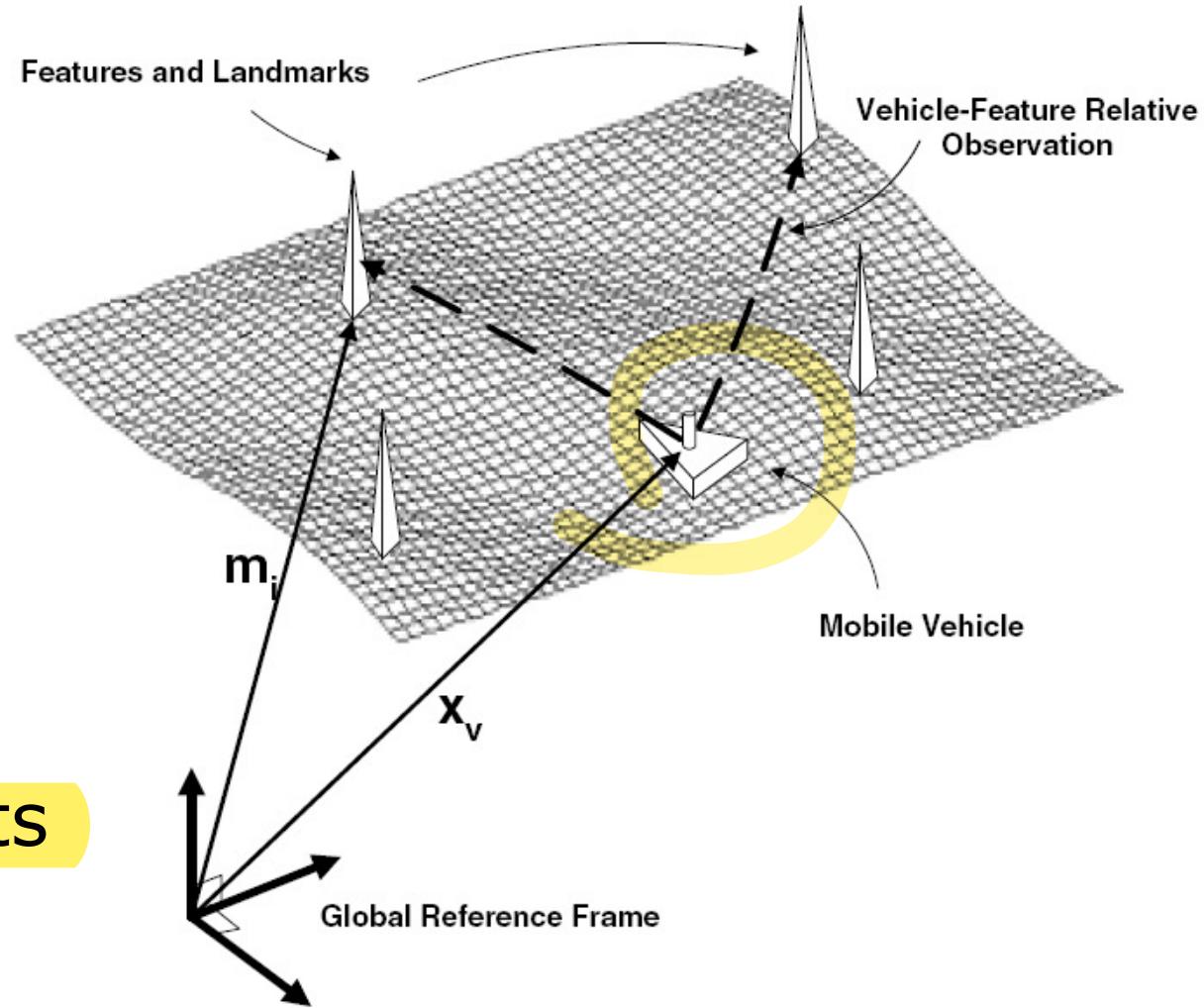
## Wanted

- Map of the environment  $m$
- Path of the robot

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

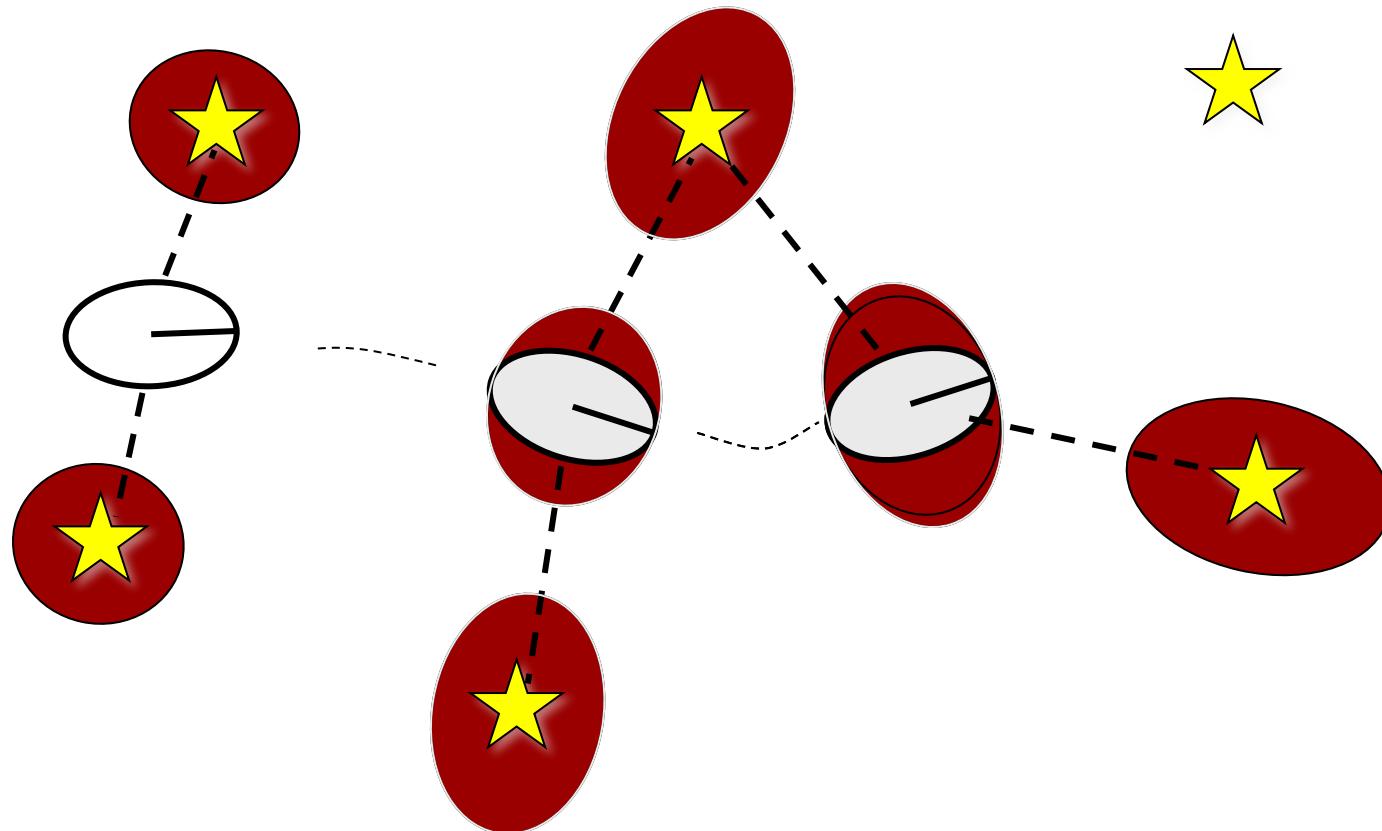
# Feature-Based SLAM

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



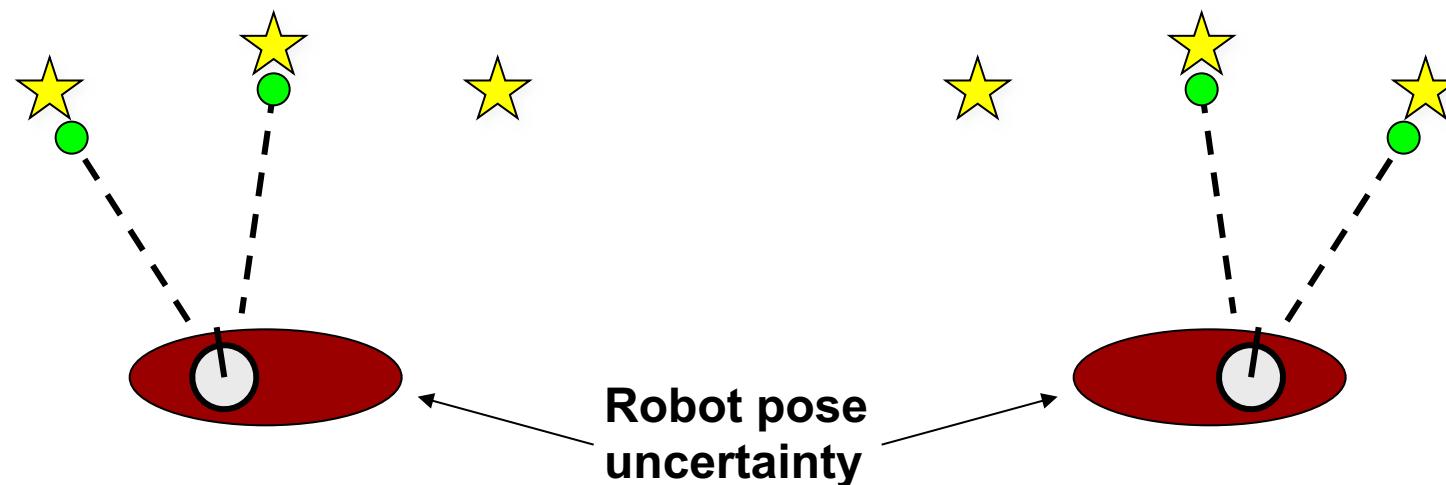
# Why is SLAM a Hard Problem?

- Robot path and map are both **unknown**
- Errors in map and pose estimates **correlated**



# Data Association

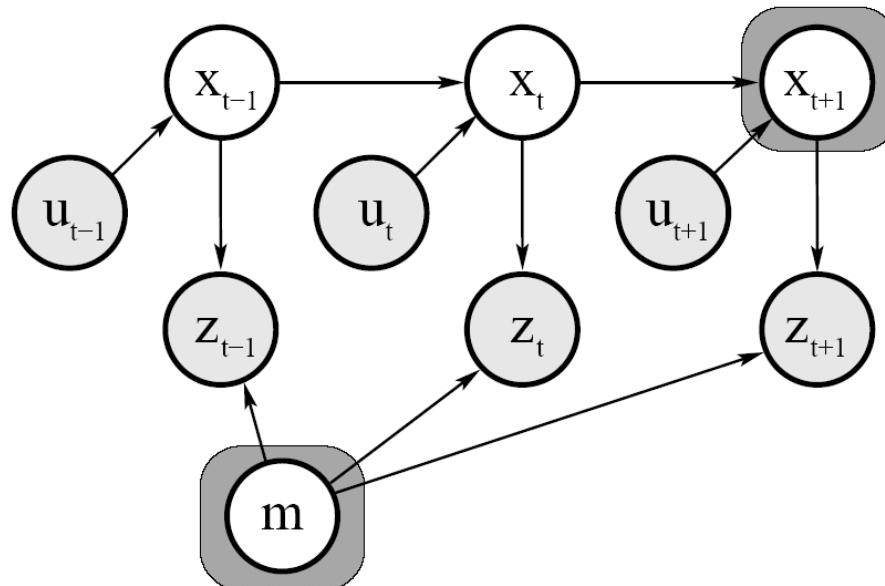
- The **mapping between observations and landmarks is unknown**
- Picking **wrong** data associations can have **catastrophic consequences** (divergence)



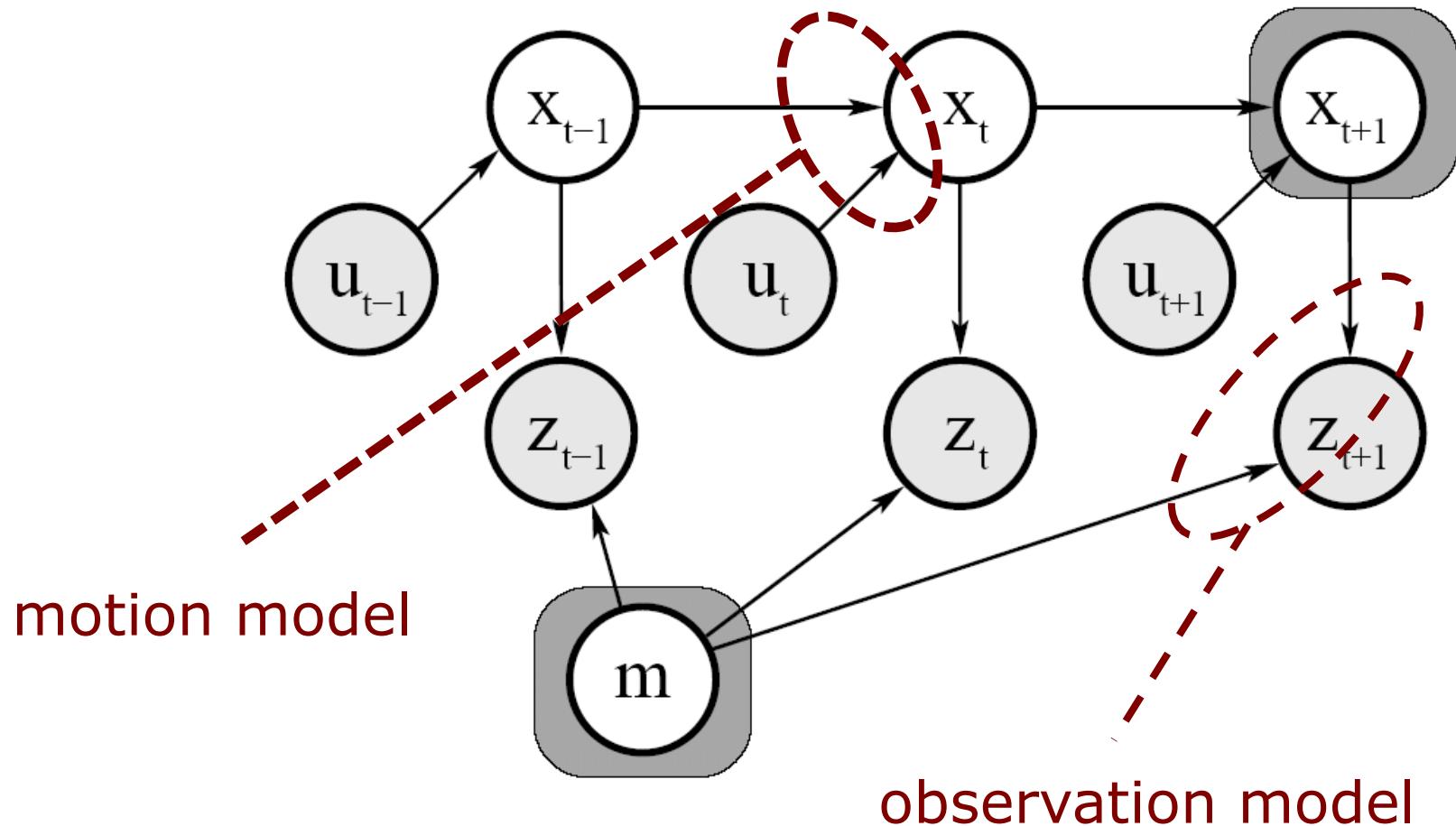
# EKF for Online SLAM

- Kalman filter as a solution to the online SLAM problem
- Estimate the most recent pose and map

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



# Motion and Observation Model



# Recap: KF Algorithm

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

2. Prediction:

$$3. \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \quad \text{motion model}$$

$$4. \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

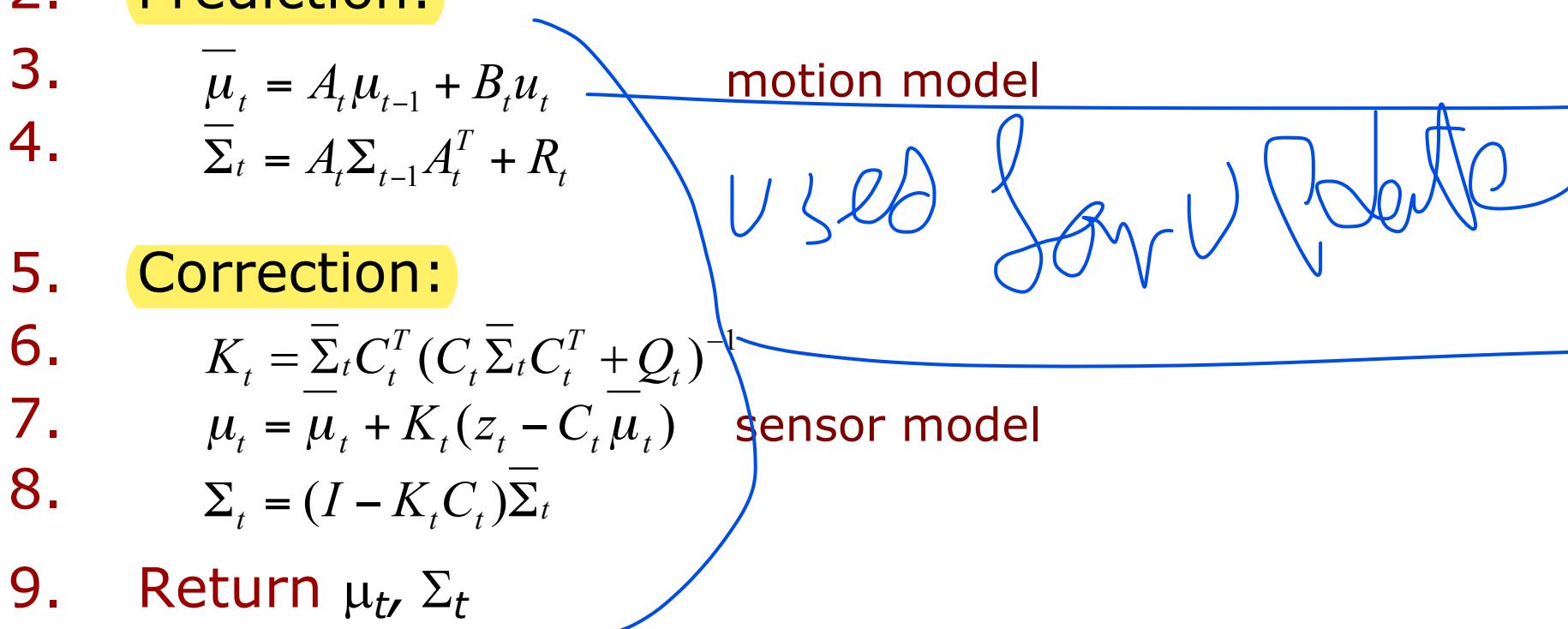
5. Correction:

$$6. K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

$$7. \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \quad \text{sensor model}$$

$$8. \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$$

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



# EKF SLAM

- Application of the EKF to SLAM
- Estimate robot's pose **and landmark locations**
- Assumption: **known correspondences**
- State space (for the 2D plane):

$$x_t = \left( \underbrace{\begin{array}{c} x, y, \theta \end{array}}_{\text{robot's pose}}, \underbrace{m_{1,x}, m_{1,y}, \dots}_{\text{landmark 1}}, \underbrace{m_{n,x}, m_{n,y}}_{\text{landmark n}} \right)^T$$

# EKF SLAM: State Representation

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

$$\left( \begin{array}{c} x \\ y \\ \theta \\ m_{1,x} \\ m_{1,y} \\ \vdots \\ m_{n,x} \\ m_{n,y} \end{array} \right) \underbrace{\left( \begin{array}{ccc} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \\ \hline \sigma_{m_{1,x}x} & \sigma_{m_{1,x}y} & \sigma_{\theta m_{1,x}} \\ \sigma_{m_{1,y}x} & \sigma_{m_{1,y}y} & \sigma_{\theta m_{1,y}} \\ \vdots & \vdots & \vdots \\ \hline \sigma_{m_{n,x}x} & \sigma_{m_{n,x}y} & \sigma_{\theta m_{n,x}} \\ \sigma_{m_{n,y}x} & \sigma_{m_{n,y}y} & \sigma_{\theta m_{n,y}} \end{array} \right)}_{\mu} \underbrace{\left( \begin{array}{cccc} \sigma_{xm_{1,x}} & \sigma_{xm_{1,y}} & \cdots & \sigma_{xm_{n,x}} & \sigma_{xm_{n,y}} \\ \sigma_{ym_{1,x}} & \sigma_{ym_{1,y}} & \cdots & \sigma_{m_{n,x}} & \sigma_{m_{n,y}} \\ \sigma_{\theta m_{1,x}} & \sigma_{\theta m_{1,y}} & \cdots & \sigma_{\theta m_{n,x}} & \sigma_{\theta m_{n,y}} \\ \hline \sigma_{m_{1,x}m_{1,x}} & \sigma_{m_{1,x}m_{1,y}} & \cdots & \sigma_{m_{1,x}m_{n,x}} & \sigma_{m_{1,x}m_{n,y}} \\ \sigma_{m_{1,y}m_{1,x}} & \sigma_{m_{1,y}m_{1,y}} & \cdots & \sigma_{m_{1,y}m_{n,x}} & \sigma_{m_{1,y}m_{n,y}} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline \sigma_{m_{n,x}m_{1,x}} & \sigma_{m_{n,x}m_{1,y}} & \cdots & \sigma_{m_{n,x}m_{n,x}} & \sigma_{m_{n,x}m_{n,y}} \\ \sigma_{m_{n,y}m_{1,x}} & \sigma_{m_{n,y}m_{1,y}} & \cdots & \sigma_{m_{n,y}m_{n,x}} & \sigma_{m_{n,y}m_{n,y}} \end{array} \right)}_{\Sigma}$$

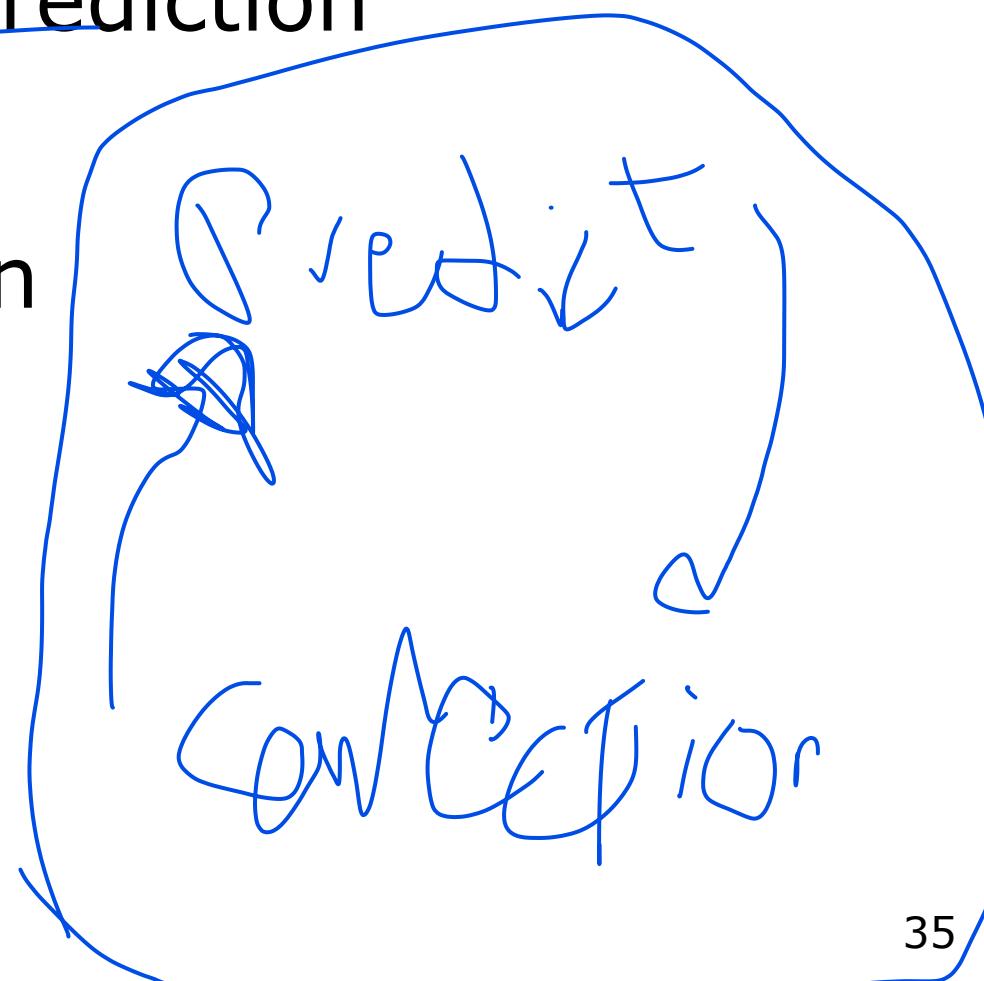
# EKF SLAM: State Representation

More compactly:

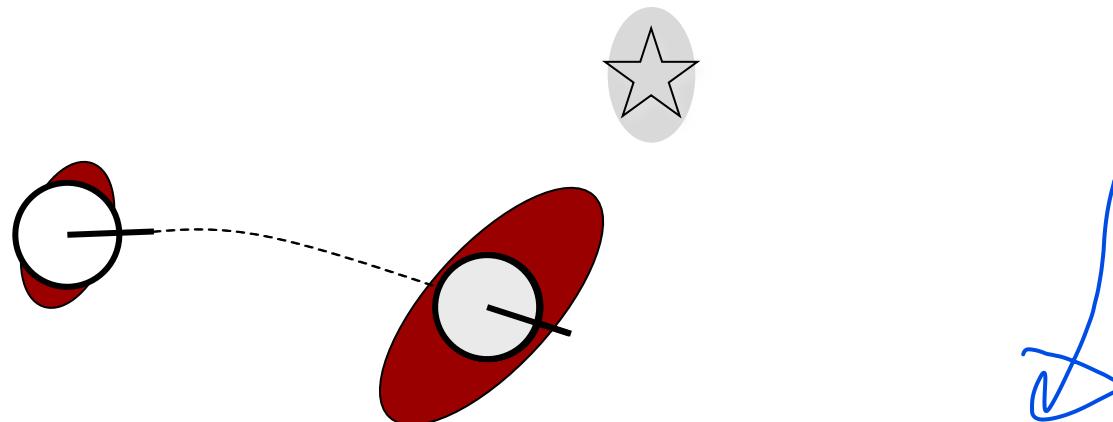
$$\left( \begin{array}{c} x_R \\ m_1 \\ \vdots \\ m_n \end{array} \right) \left( \begin{array}{cccc} \Sigma_{x_R x_R} & & & \\ \Sigma_{m_1 x_R} & \ddots & & \\ \vdots & & \ddots & \\ \Sigma_{m_n x_R} & & & \Sigma_{m_n m_n} \end{array} \right)$$

# EKF SLAM: Filter Cycle

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



# EKF SLAM: State Prediction

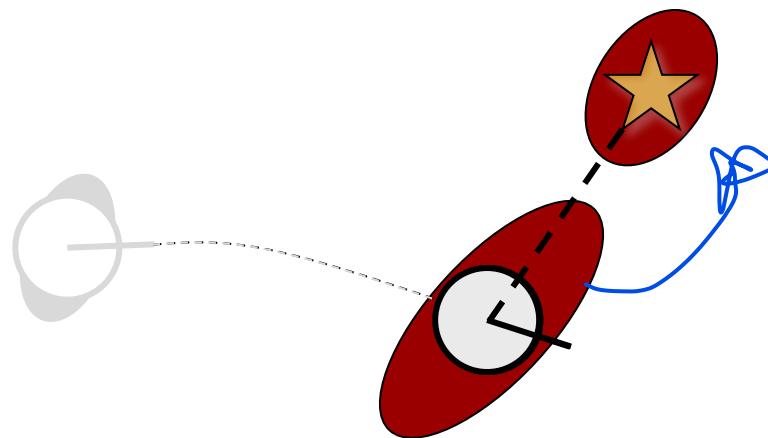


The diagram illustrates a robot's movement from a circular pose to an elliptical pose, with a star-shaped object in the background.

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

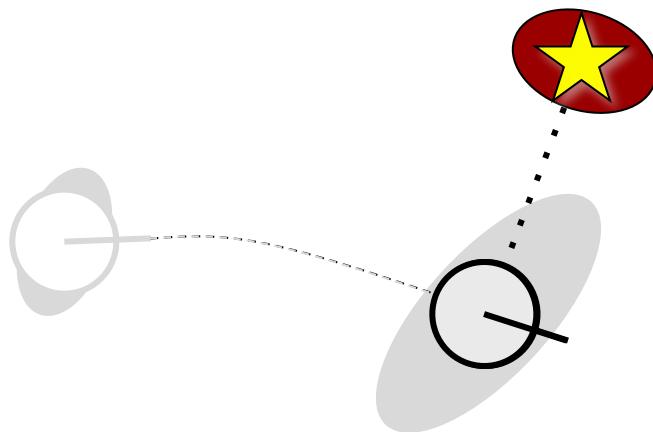
The state vector  $\mu$  contains the robot's position  $x_R$  and the weights  $m_1, m_2, \dots, m_n$ . The covariance matrix  $\Sigma$  is a symmetric matrix representing the uncertainty of the state. The diagonal elements represent the variance of the position and weights, while the off-diagonal elements represent the covariances between the position and weights, and between the weights themselves.

# EKF SLAM: Measurement Prediction



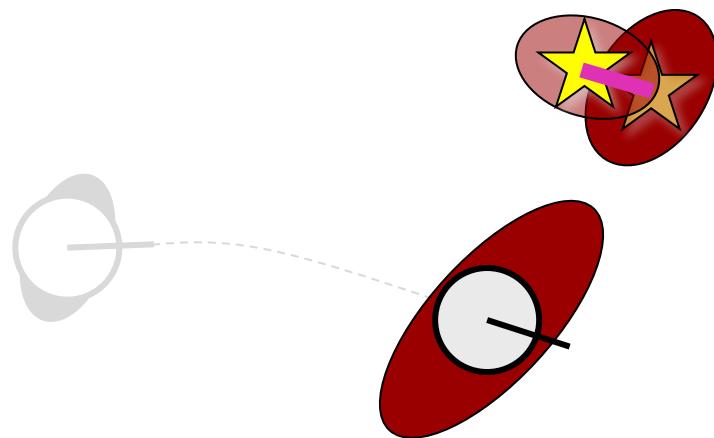
$$\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix} \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots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

# EKF SLAM: Obtained Measurement



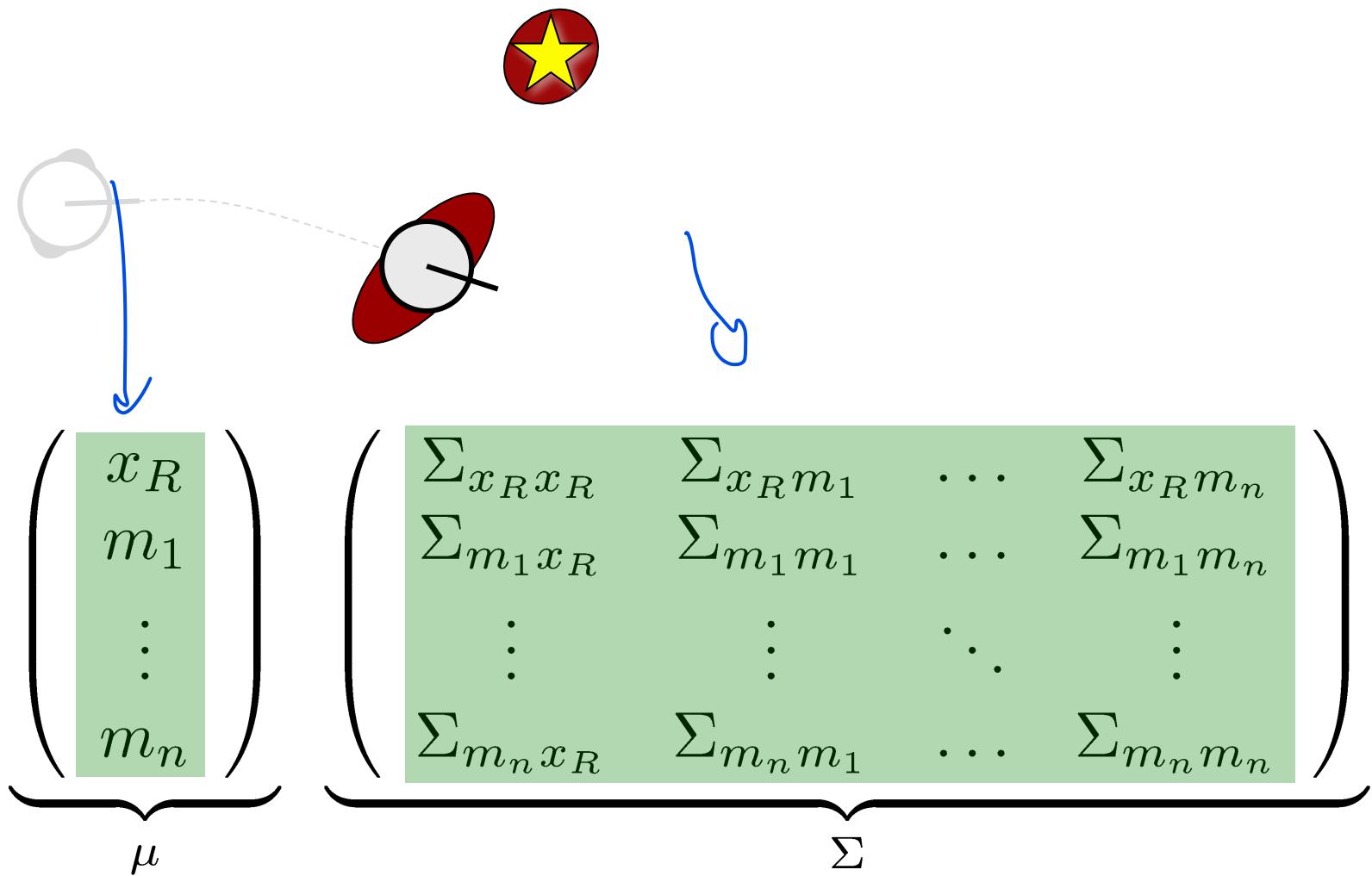
$$\underbrace{\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix}}_{\mu} \quad \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R}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

# EKF SLAM: Data Association and Difference to Predicted Observ.



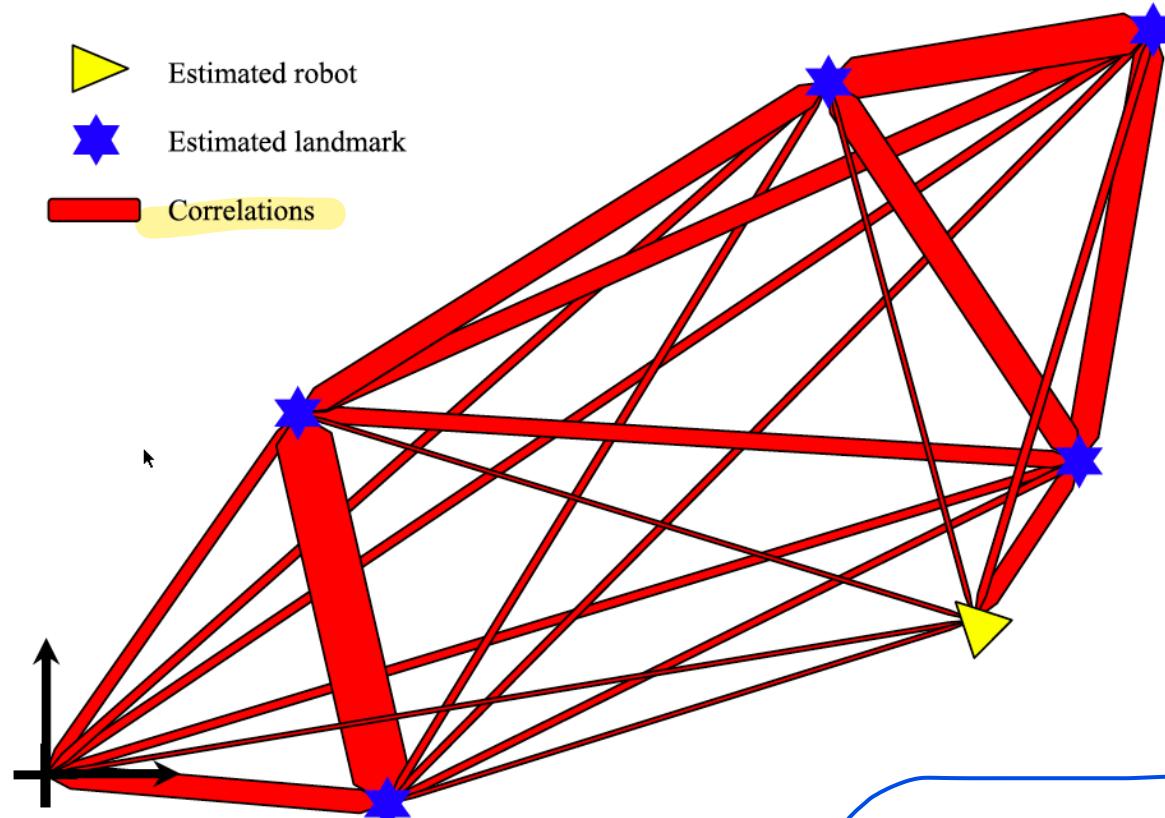
$$\begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix} \underbrace{\begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots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

# EKF SLAM: Update Step



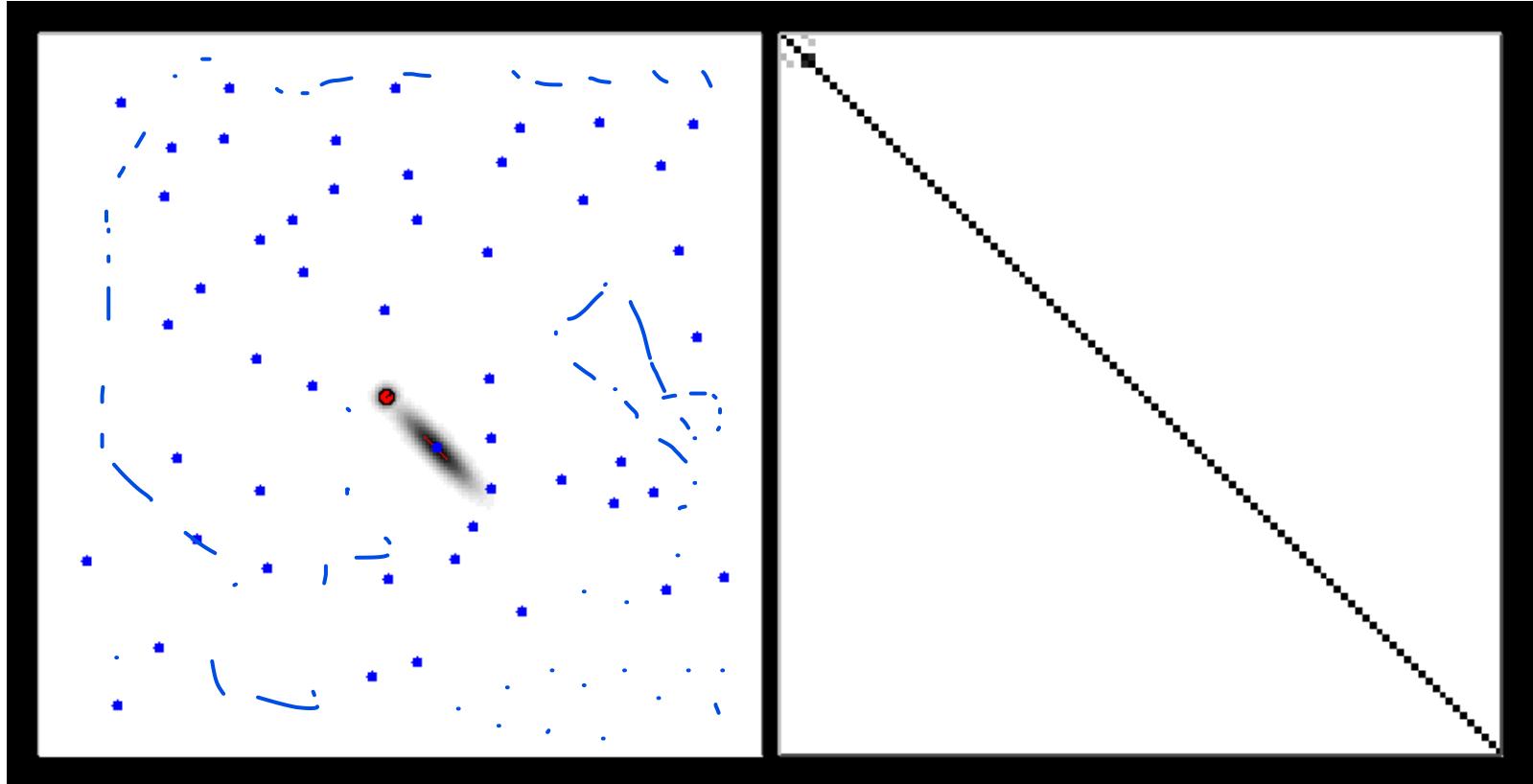
# EKF SLAM Correlations

Over time, the landmark estimates become **fully correlated**



Courtesy: Dissanayake

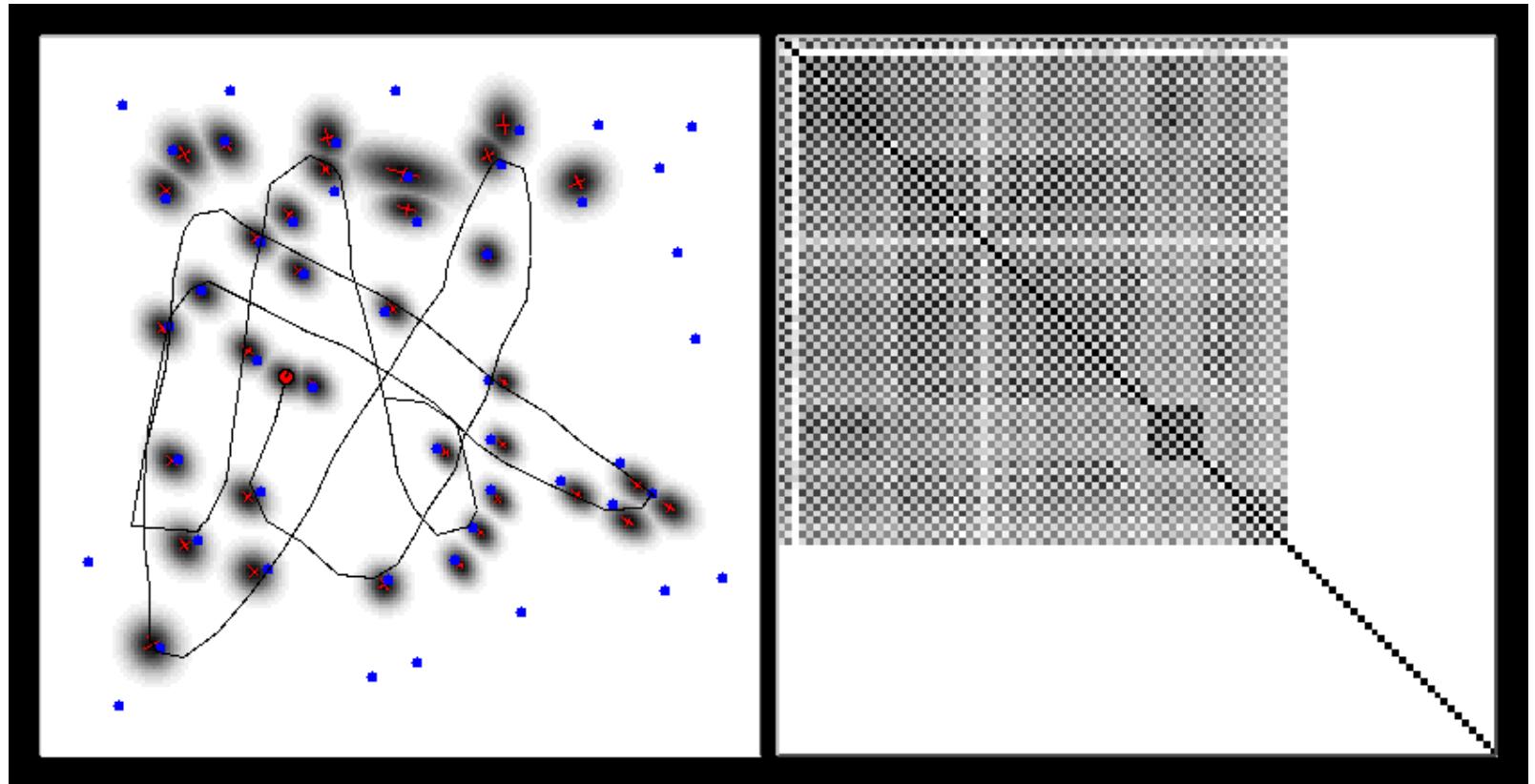
# EKF SLAM



Map

Correlation matrix

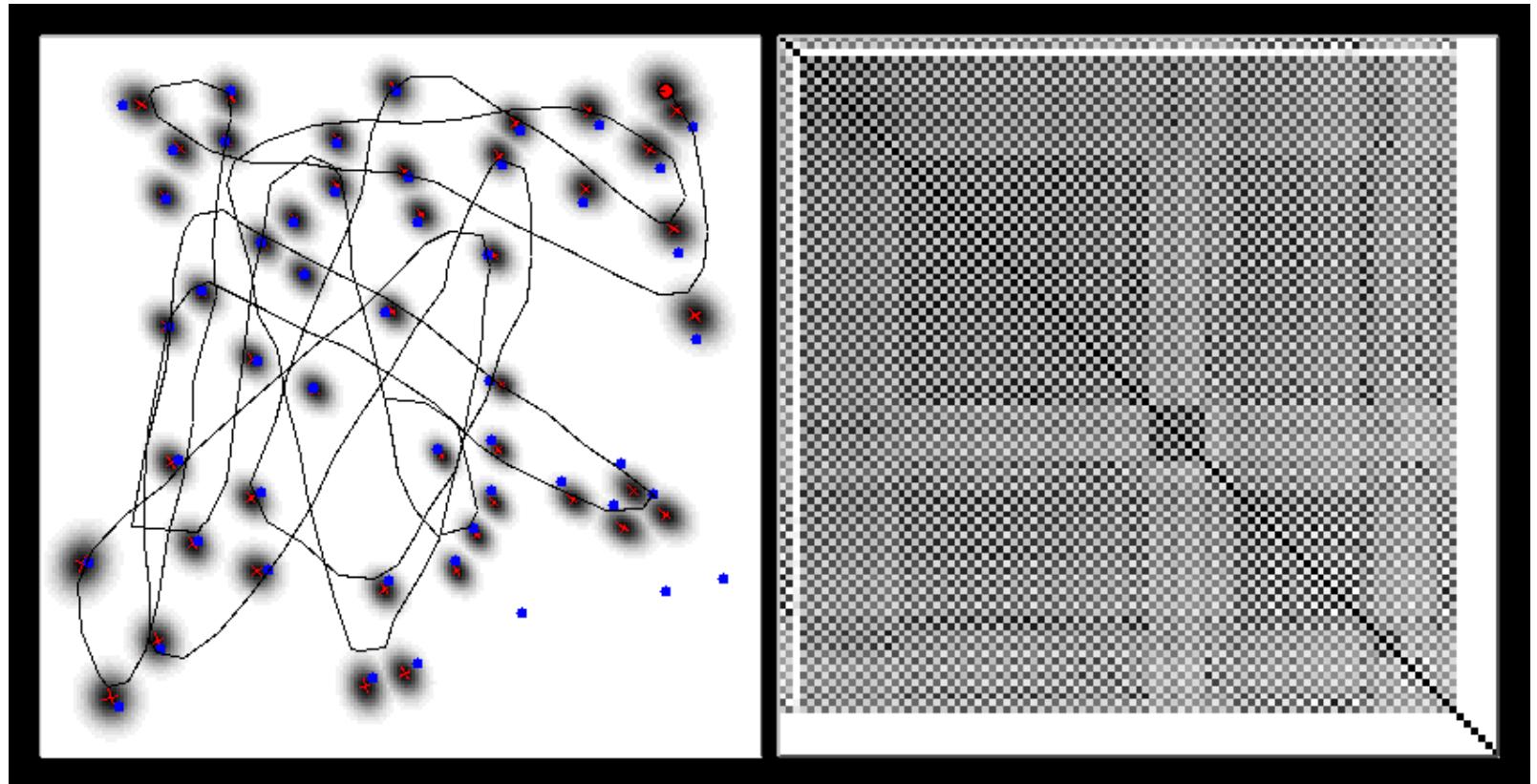
# EKF SLAM



Map

Correlation matrix

# EKF SLAM



Map

Correlation matrix

# EKF SLAM: Correlations Matter

- What if we neglected cross-correlations?

$$\Sigma_k = \begin{bmatrix} \Sigma_R & 0 & \cdots & 0 \\ 0 & \Sigma_{M_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{M_n} \end{bmatrix} \quad \begin{aligned} \Sigma_{RM_i} &= \mathbf{0}_{3 \times 2} \\ \Sigma_{M_i M_{i+1}} &= \mathbf{0}_{2 \times 2} \end{aligned}$$

# EKF SLAM: Correlations Matter

- What if we neglected cross-correlations?

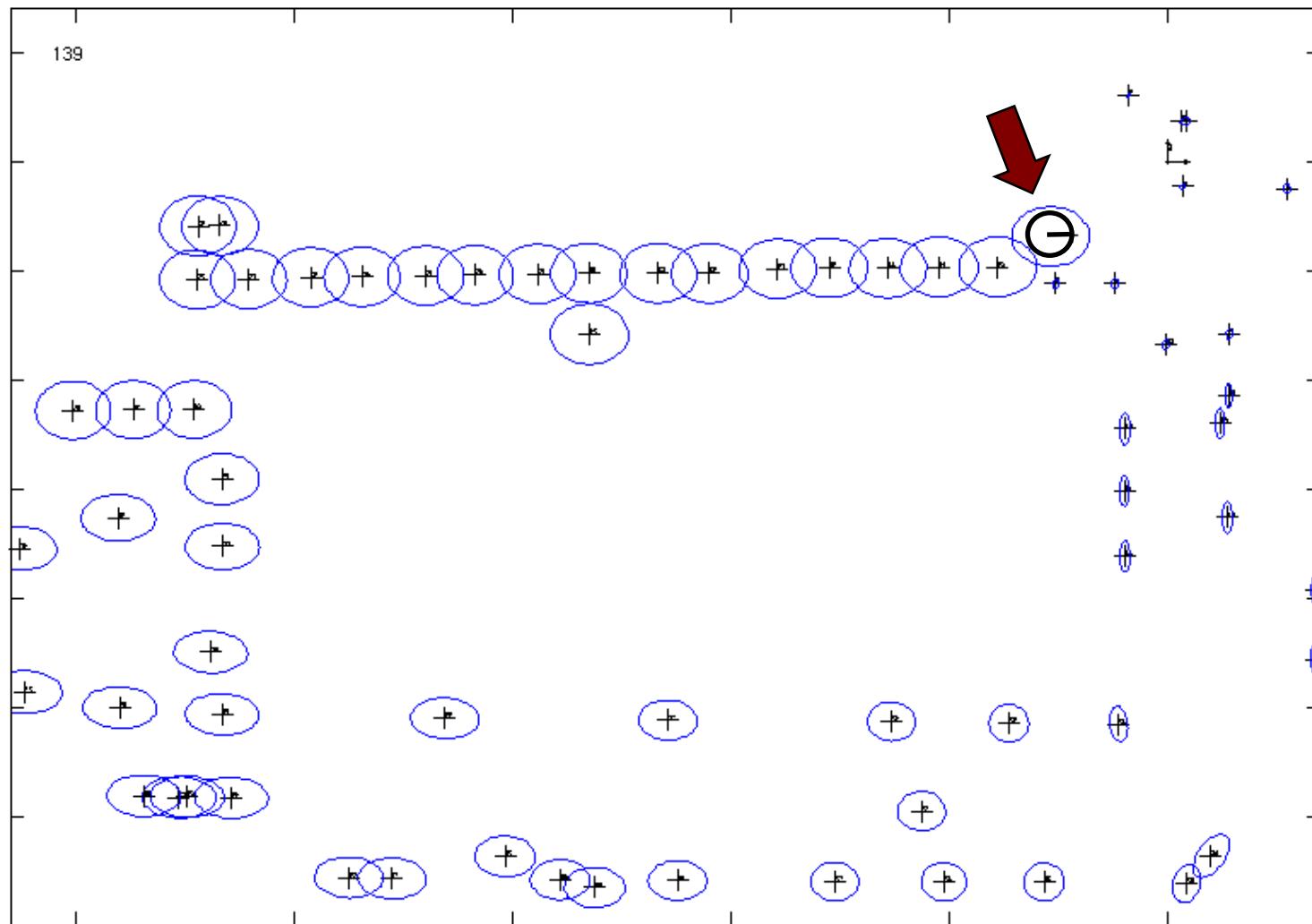
$$\Sigma_k = \begin{bmatrix} \Sigma_R & 0 & \cdots & 0 \\ 0 & \Sigma_{M_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{M_n} \end{bmatrix} \quad \Sigma_{RM_i} = \mathbf{0}_{3 \times 2}$$
$$\Sigma_{M_i M_{i+1}} = \mathbf{0}_{2 \times 2}$$

- Landmark and robot uncertainties would become overly optimistic
- Data association would fail
- As a result, multiple map entries of the same landmark
- Inconsistent map

# Loop Closing

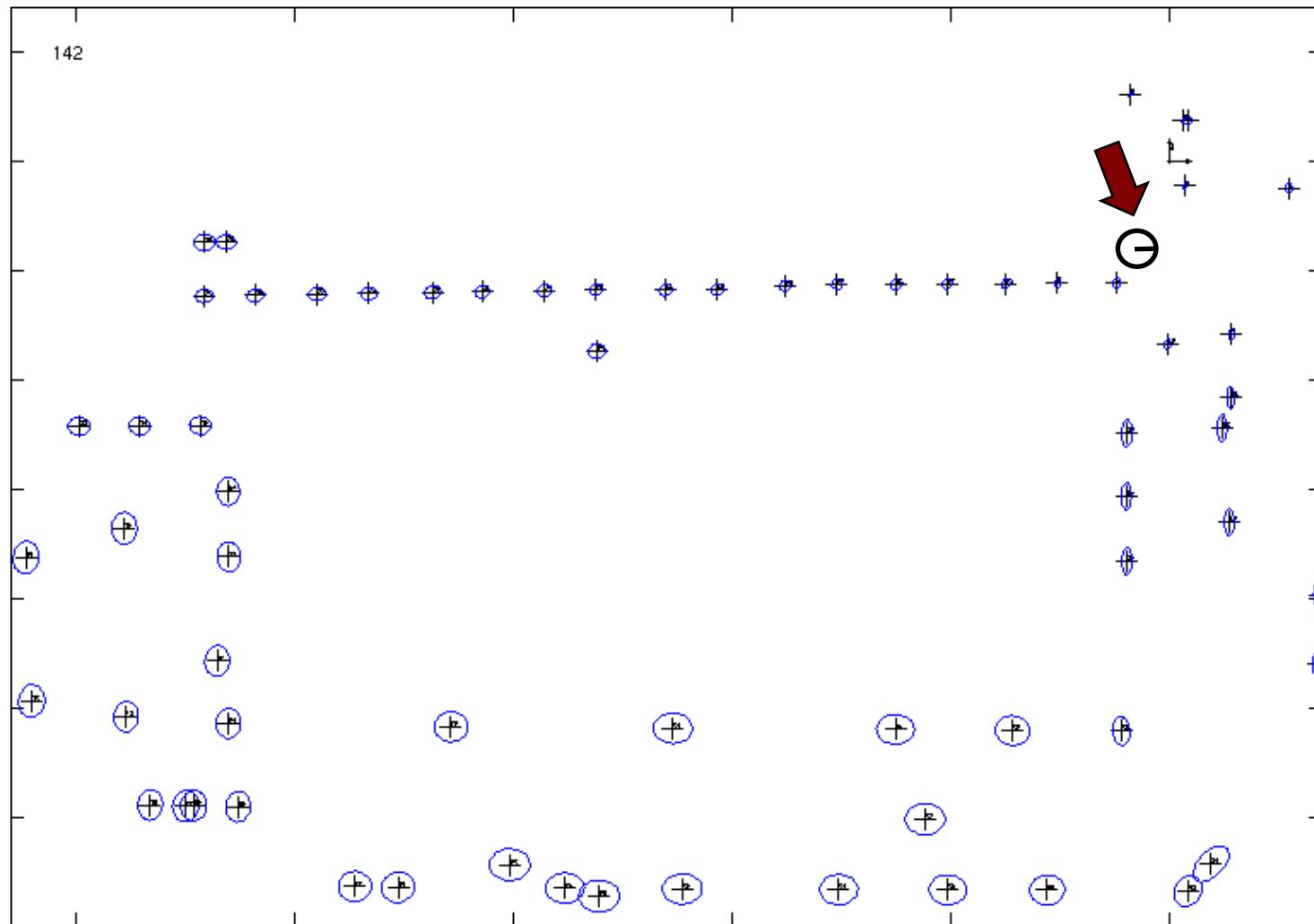
- Recognizing an already mapped area
- Data association under
  - high ambiguity
  - possible environment symmetries
- **Uncertainties collapse after a loop closure** (whether the closure was correct or not)

# Before the Loop Closure



Courtesy: K. Arras

# After the Loop Closure



Courtesy: K. Arras

# Loop Closures in SLAM

- Loop closing reduces the uncertainty in robot and landmark estimates
- This can be exploited when exploring an environment
- However, wrong loop closures lead to filter divergence

# Example: Victoria Park Dataset



Courtesy: E. Nebot

51

# Victoria Park: Data Acquisition



Courtesy: E. Nebot

52

# Victoria Park: Landmarks



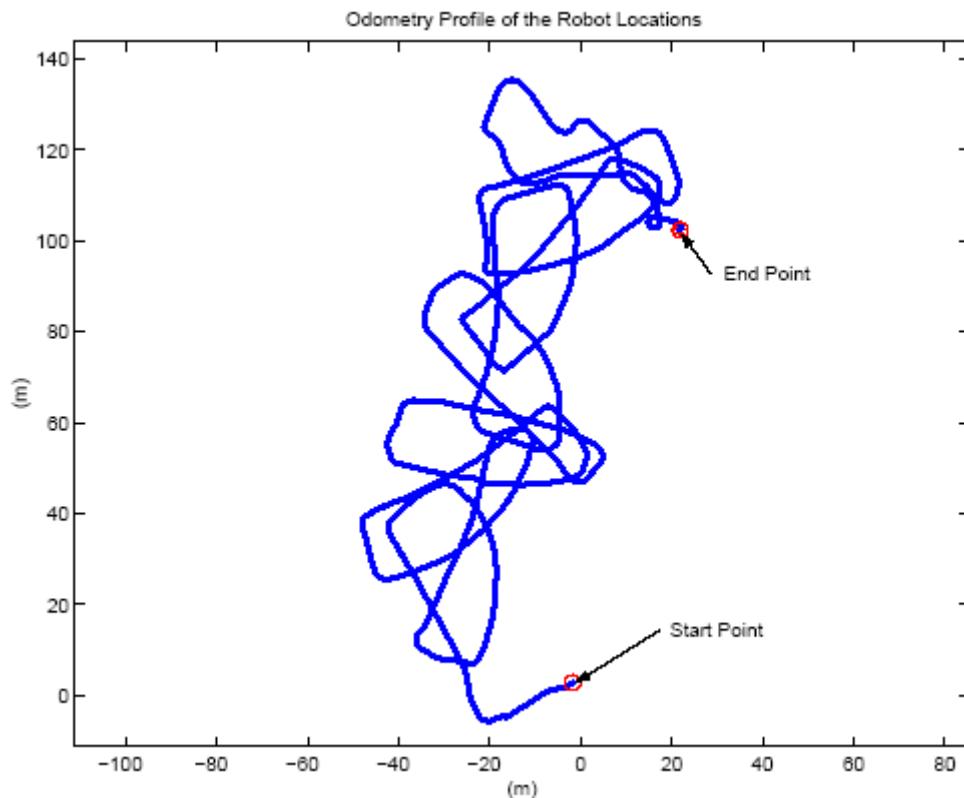
Courtesy: E. Nebot

# Example: Tennis Court Dataset

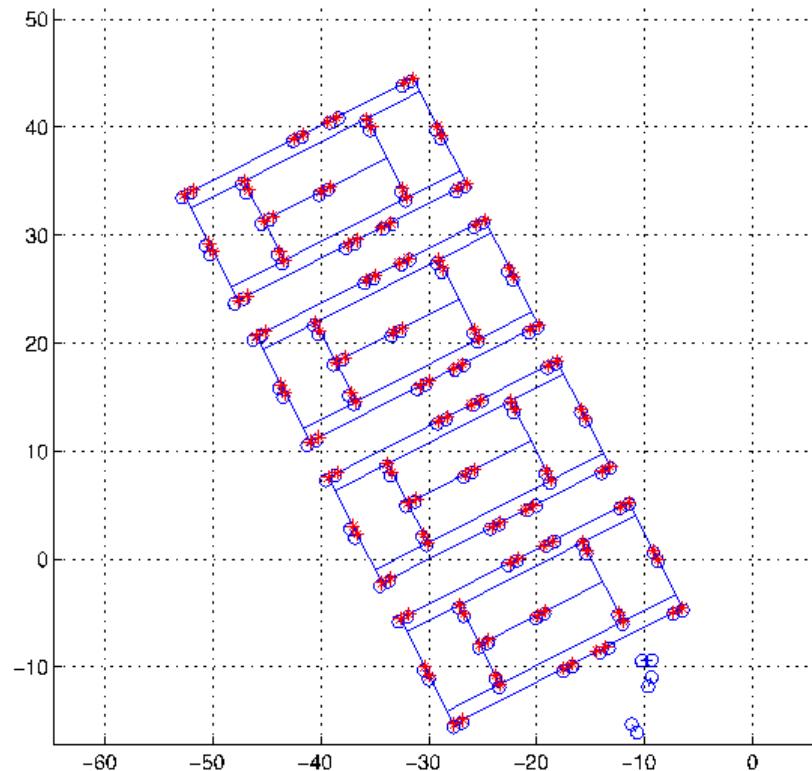


Courtesy: J. Leonard and M. Walter

# EKF SLAM on a Tennis Court



odometry



estimated map

# EKF SLAM Complexity

- Cubic complexity depends only on the measurement dimensionality
- Cost per step: dominated by the number of landmarks:  $O(n^2)$
- Memory consumption:  $O(n^2)$
- The EKF becomes computationally intractable for large maps!

# Summary: EKF SLAM

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

# Particle Filter MC

- Non-parametric recursive Bayes filter
- Posterior is represented by a set of weighted samples
- Can model arbitrary distributions
- Works well in low-dimensional spaces
- Three steps
  - Sampling from proposal
  - Importance weighting
  - Resampling

# Particle Representation

- A set of weighted samples

$$\mathcal{X} = \{\langle x^{[i]}, w^{[i]} \rangle\}_{i=1,\dots,N}$$

- Each sample is a hypothesis about the state
- For feature-based SLAM:

$$x = \underbrace{(x_{1:t}, m_{1,x}, m_{1,y}, \dots, m_{M,x}, m_{M,y})^T}_{\text{poses}} \quad \underbrace{\phantom{x_{1:t}, m_{1,x}, m_{1,y}, \dots, m_{M,x}, m_{M,y}}_T}_{\text{landmarks}}$$

# Dimensionality Problem

- Particle filters are effective in low-dimensional spaces
- The likely regions of the state space need to be covered with samples

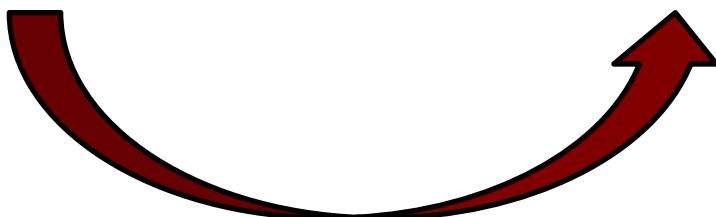
$$x = (x_{1:t}, m_{1,x}, m_{1,y}, \dots, m_{M,x}, m_{M,y})^T$$

**high-dimensional!**

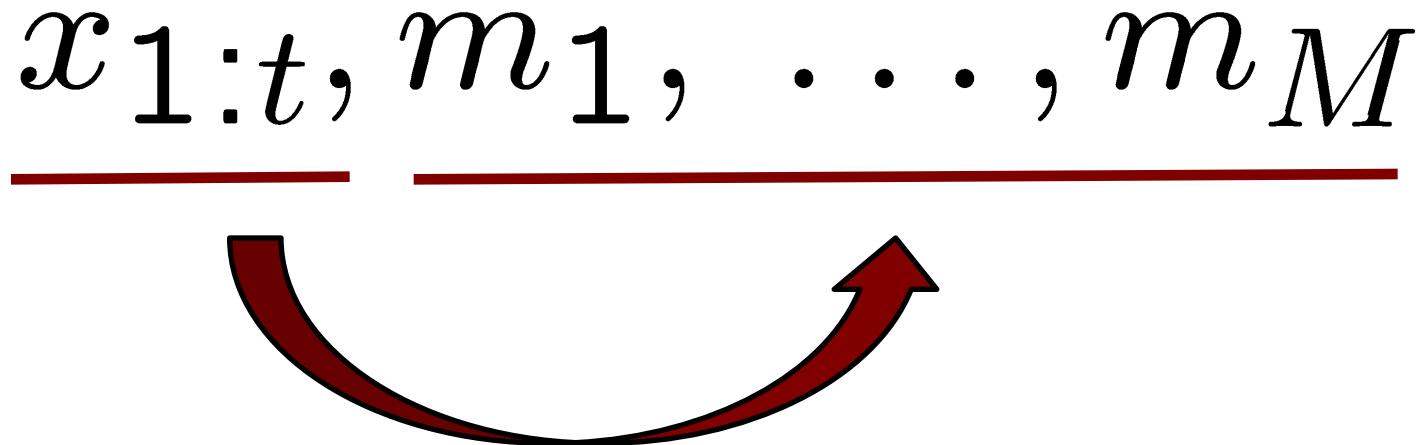
# Can We Exploit Dependencies Between the Different Dimensions of the State Space?

$x_{1:t}, m_1, \dots, m_M$

# If We Know the Poses of the Robot, Mapping is Easy!

$$\underline{x_{1:t}, m_1, \dots, m_M}$$


# Key Idea



- If we use the particle set **only** to model the robot's path, each sample is a path hypothesis
- For each sample, we can **compute** an individual map of landmarks

# Acknowledgment

- These slides have been created by Wolfram Burgard, Dieter Fox, Cyrill Stachniss and Maren Bennewitz