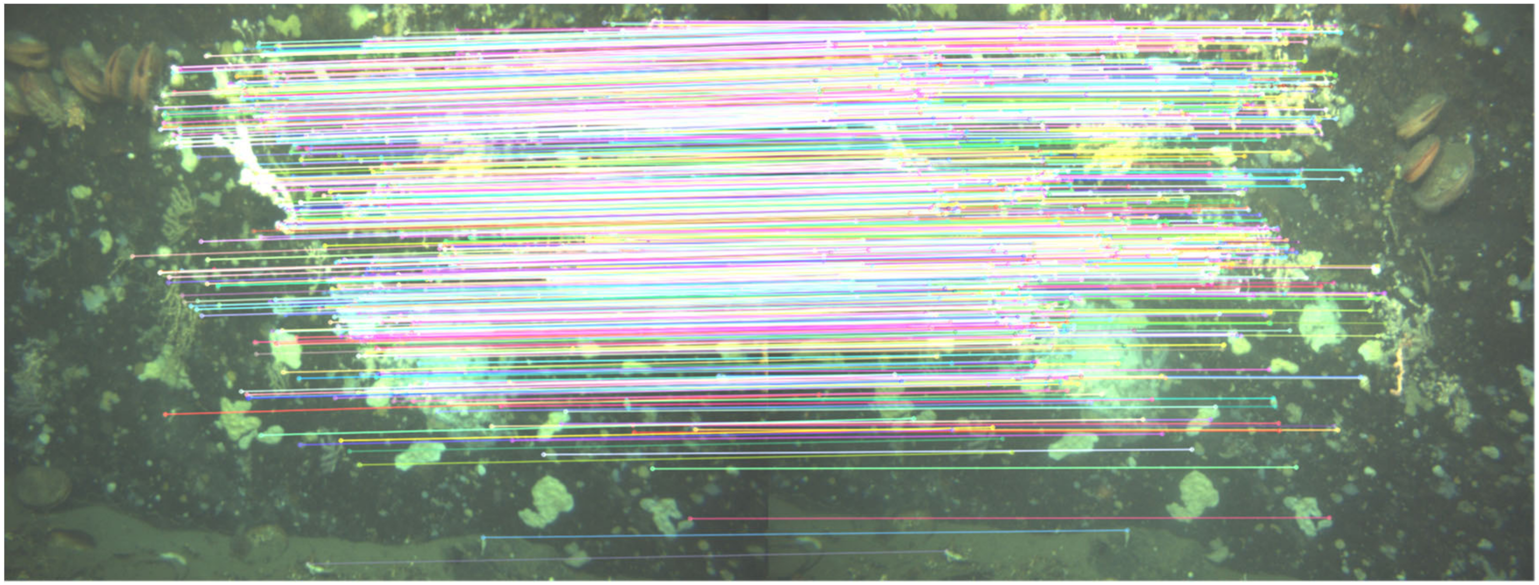
TMR4585

Simultaneous Localization And Mapping

2019

Martin Ludvigsen
Applied Underwater Robotics Laboratory (AUR-Lab)
Department of Marine Technology
Norwegian University of Science and Technology
E-mail: martinl@ntnu.no

Kahoot



Learning objectives SLAM

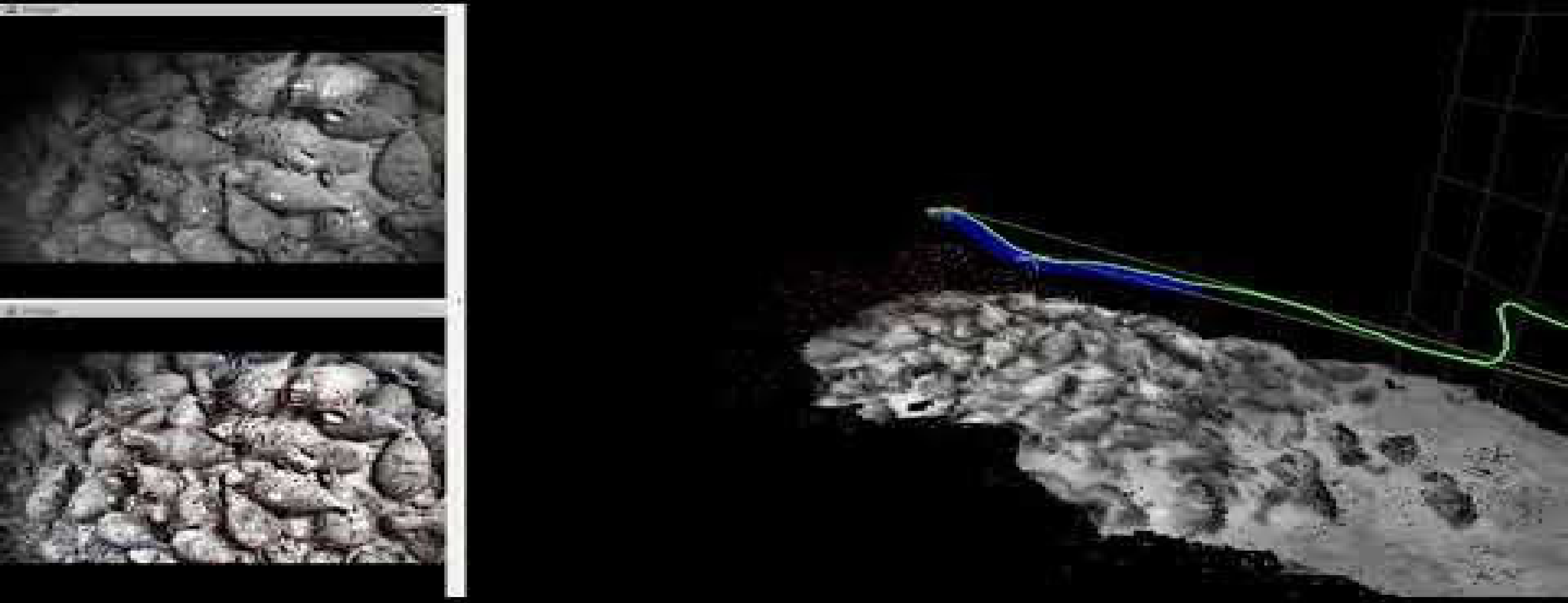
- Whats is SLAM
- Terms
- Observation model - motion model
- Full SLAM vs Online SLAM
- SLAM approaches
  - EKF
  - Particle filter
  - Graph based

What is SLAM?

- Computing the robot’s poses and the map of the environment at the same time

- **Localization**: estimating the robot’s location
- **Mapping**: building a map
- **SLAM**: building a map and localizing the robot simultaneously

Visual-Inertial-Pressure SLAM with online dense 3D mapping on an underwater archeological site.



Related Terms

State Estimation

Localization

Mapping

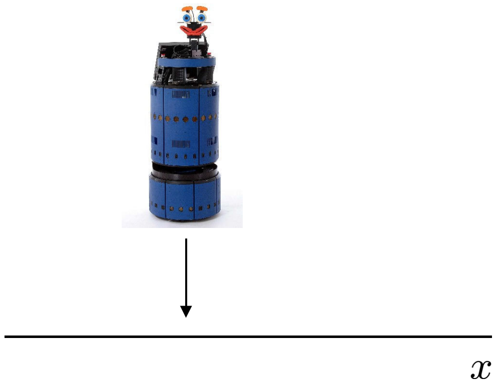
SLAM

Navigation

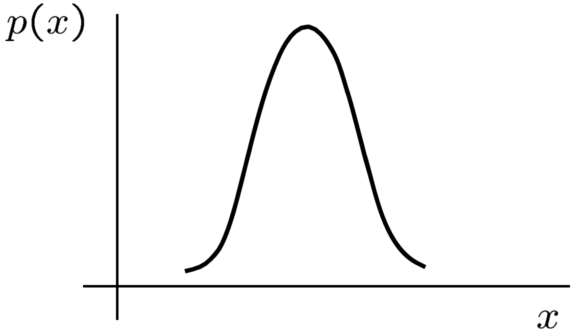
Motion Planning

Probabilistic Approaches

- Uncertainty in the robot’s motions and observations
- Use the probability theory to explicitly represent the uncertainty



“The robot is exactly here”



“The robot is somewhere here”

SLAM History by Durrant-Whyte

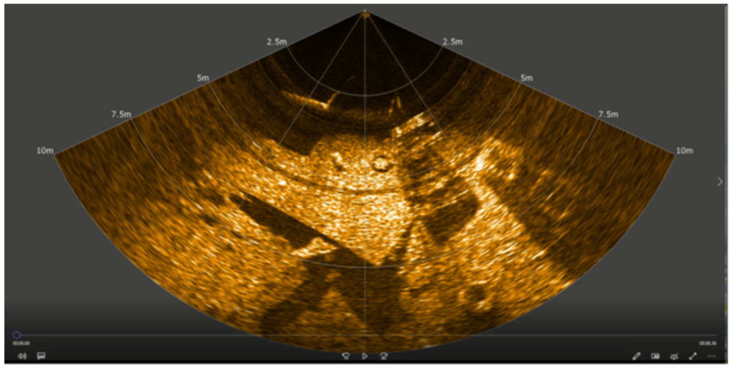
- 1985/86: Smith et al. and Durrant-Whyte describe geometric uncertainty and relationships between features or landmarks
- 1986: Discussions at ICRA on how to solve the SLAM problem followed by the key paper by Smith, Self and Cheeseman
- 1990-95: Kalman-filter based approaches
- 1995: SLAM acronym coined at ISRR'95
- 1995-1999: Convergence proofs & first demonstrations of real systems
- 2000: Wide interest in SLAM started

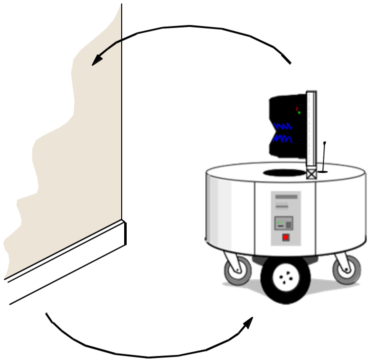
The SLAM Problem

SLAM is the process by which a robot **builds a map** of the environment and, at the same time, uses this map to **compute its location**

- Localization: inferring location given a map
- Mapping: inferring a map given a location
- SLAM: learning a map and locating the robot simultaneously

The SLAM Problem





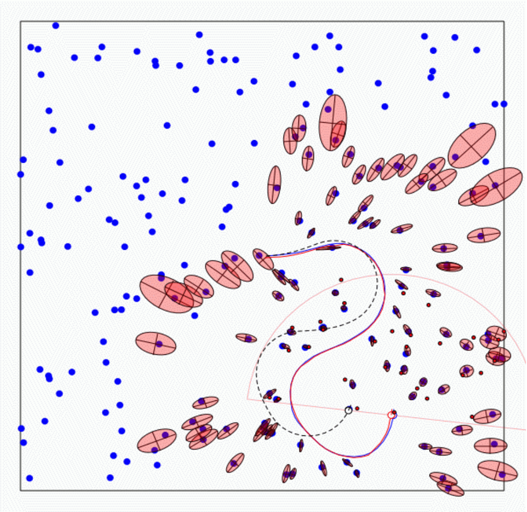
- SLAM is a **chicken-or-egg problem**:
  → A map is needed for localizing a robot
  → A pose estimate is needed to build a map

- Thus, SLAM is (regarded as) a **hard problem** in robotics

The SLAM Problem

Given:

- The robot’s controls
  $\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\}$
- Relative observations

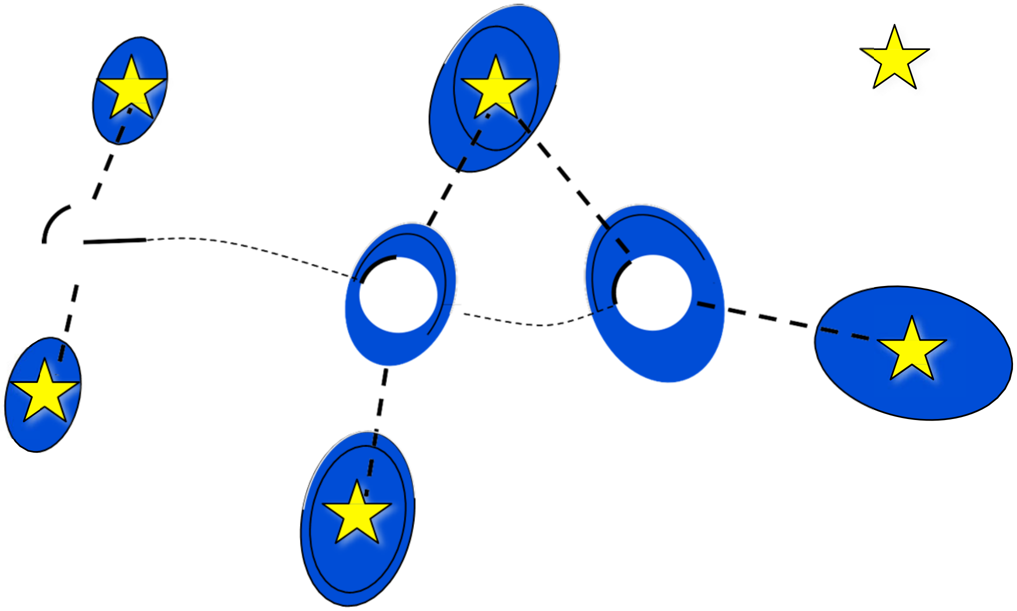


Wanted:

- Map of features
  $\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$
- Path of the robot
  $\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\}$

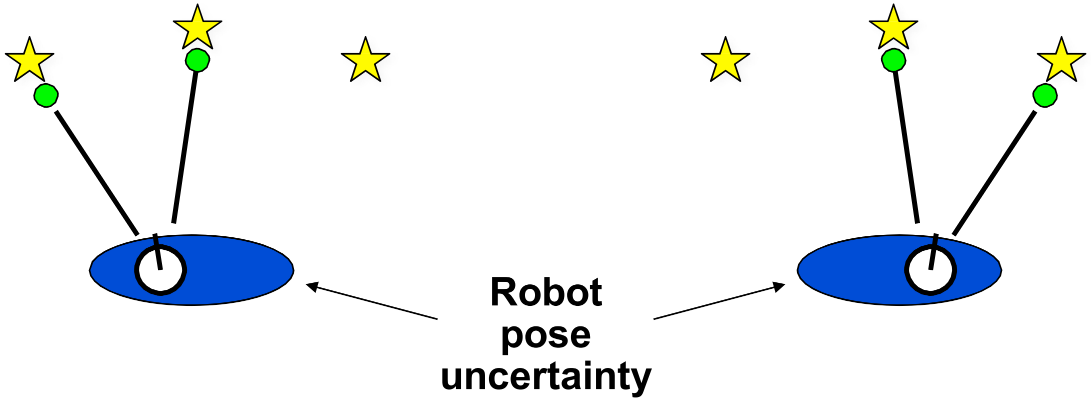
Why is SLAM a hard problem?

SLAM: robot path and map are both **unknown**



Robot path error correlates errors in the map

Why is SLAM a hard problem?

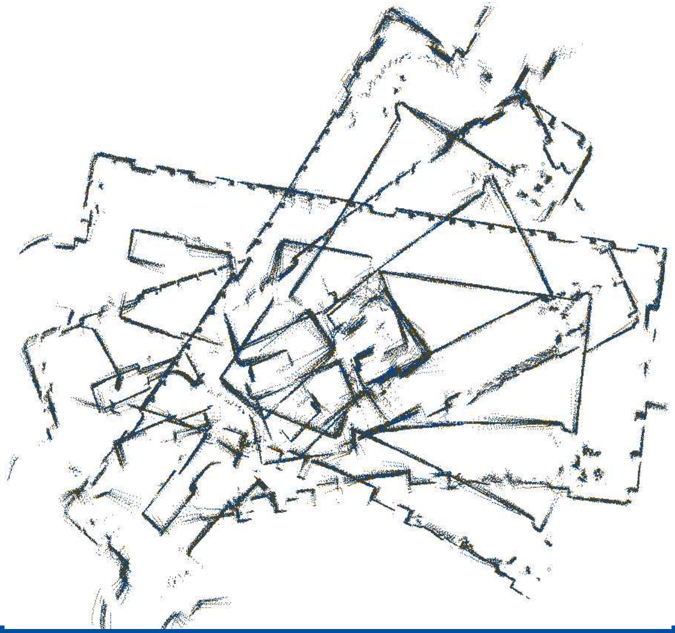


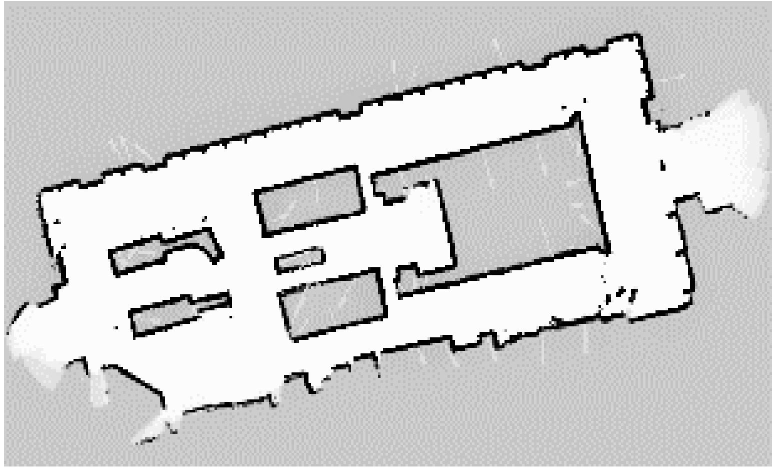
Robot pose uncertainty

- In the real world, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

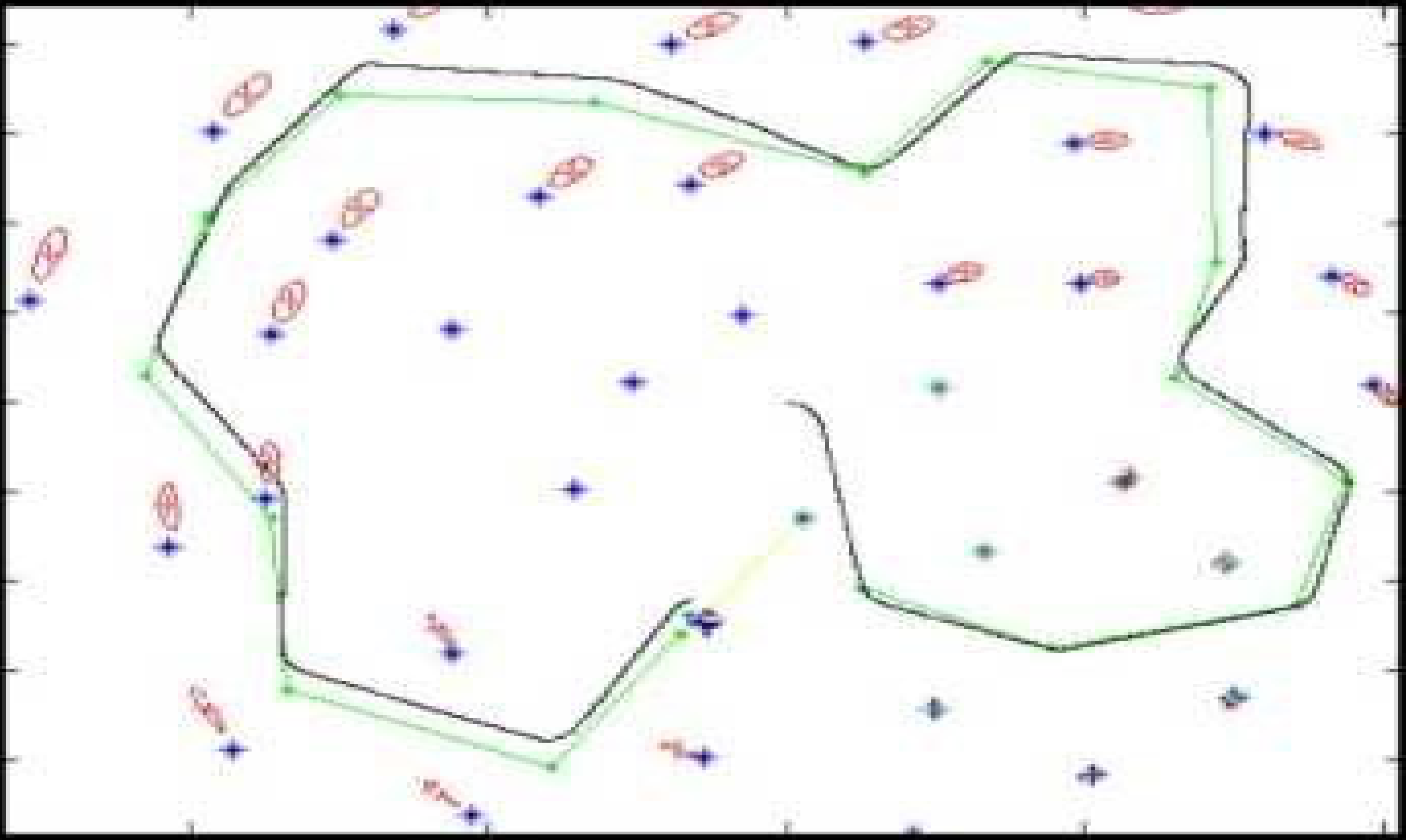
Cyclic Environments

- Small local error accumulate to arbitrary large global errors!
- This is usually irrelevant for navigation
- However, when closing loops, global error does matter





Loop Closure



SLAM:

Simultaneous Localization and Mapping

- Full SLAM:

Estimates entire path and map!

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

- Online SLAM:

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

Integrations (marginalization) typically done one at a time

Estimates most recent pose and map!

Terminology

- Observation model: $P(z_t \mid x_t)$ or $P(z_t \mid x_t, m)$
  - The probability of a measurement z_t given that the robot is at position x_t *and map m*.
- Motion Model: $P(x_t \mid x_{t-1}, u_t)$
  - The posterior probability that action u_t carries the robot from x_{t-1} to x_t.

SLAM algorithm

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

- Prediction

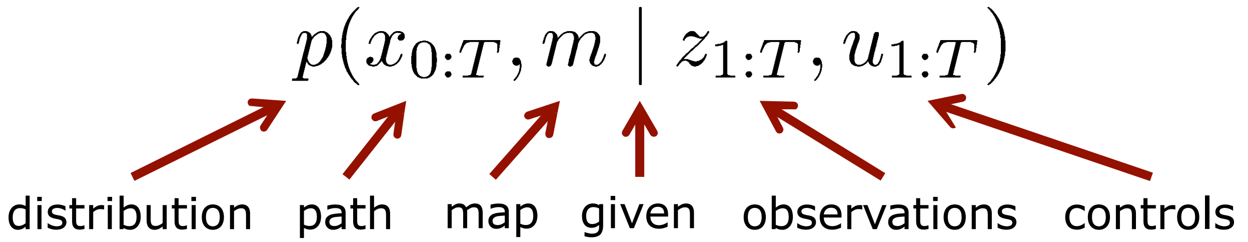
$$\overline{bel}(x_t, m) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1}, m)\, dx_{t-1}$$

- Update

$$bel(x_t, m) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t, m)$$

In the Probabilistic World

Estimate the robot’s path and the map

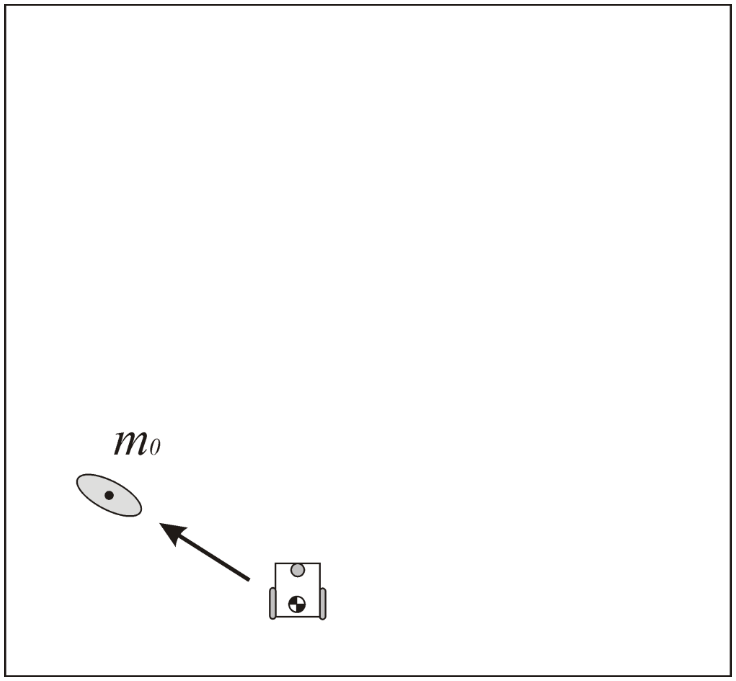


$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution path map given observations controls

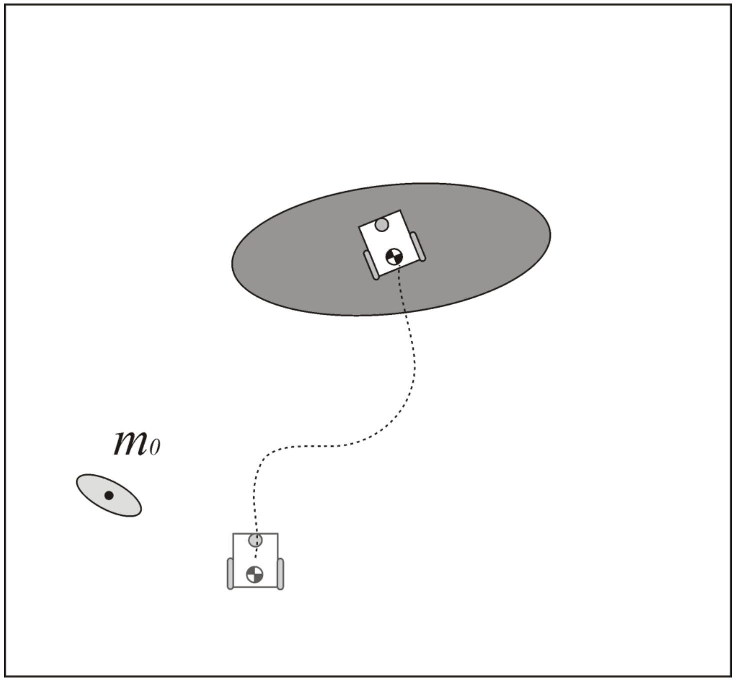
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



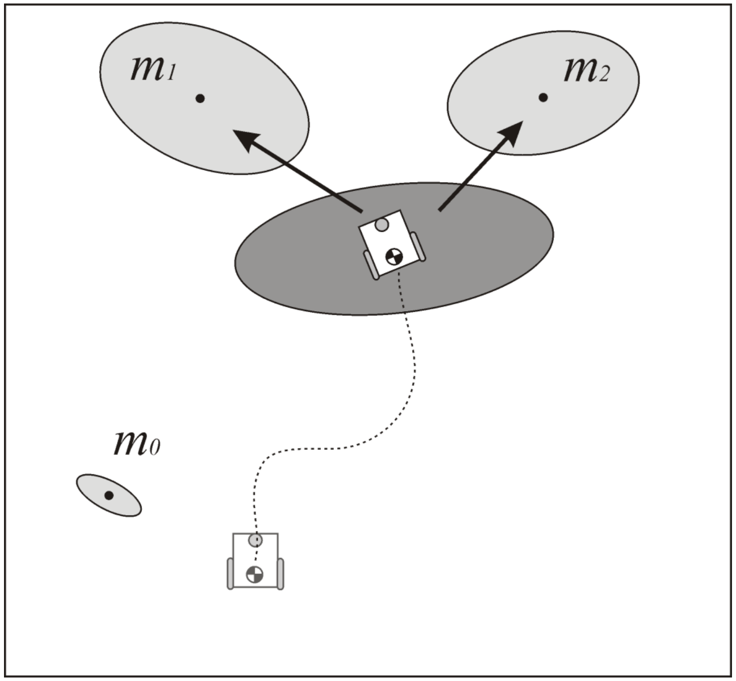
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry



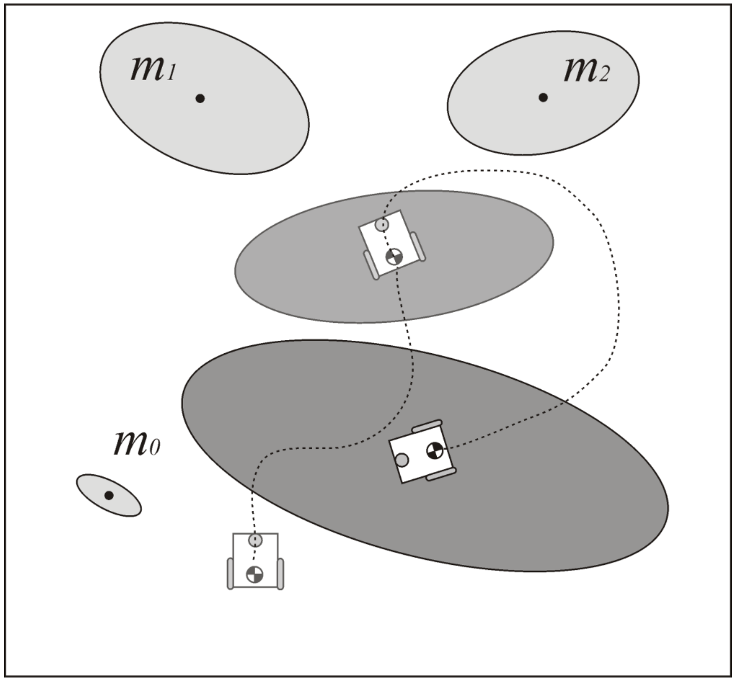
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty
- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



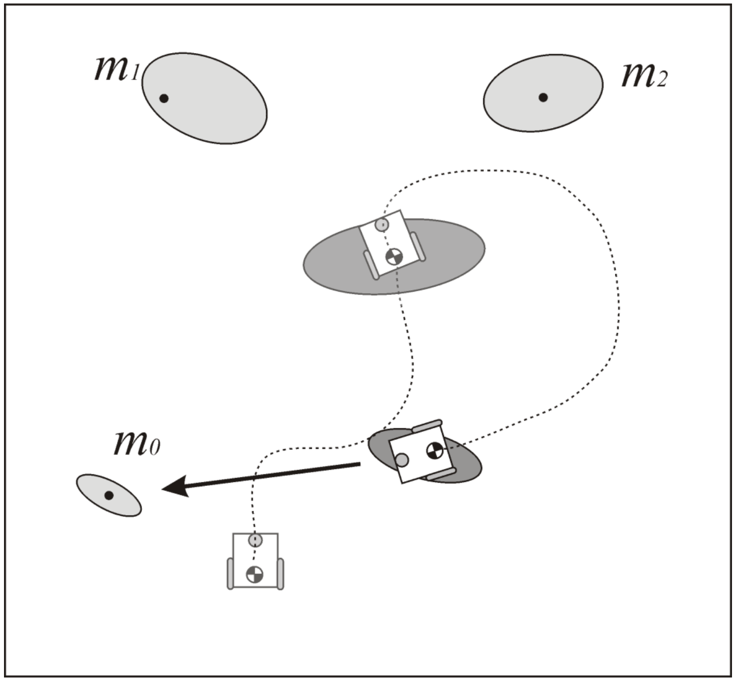
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry



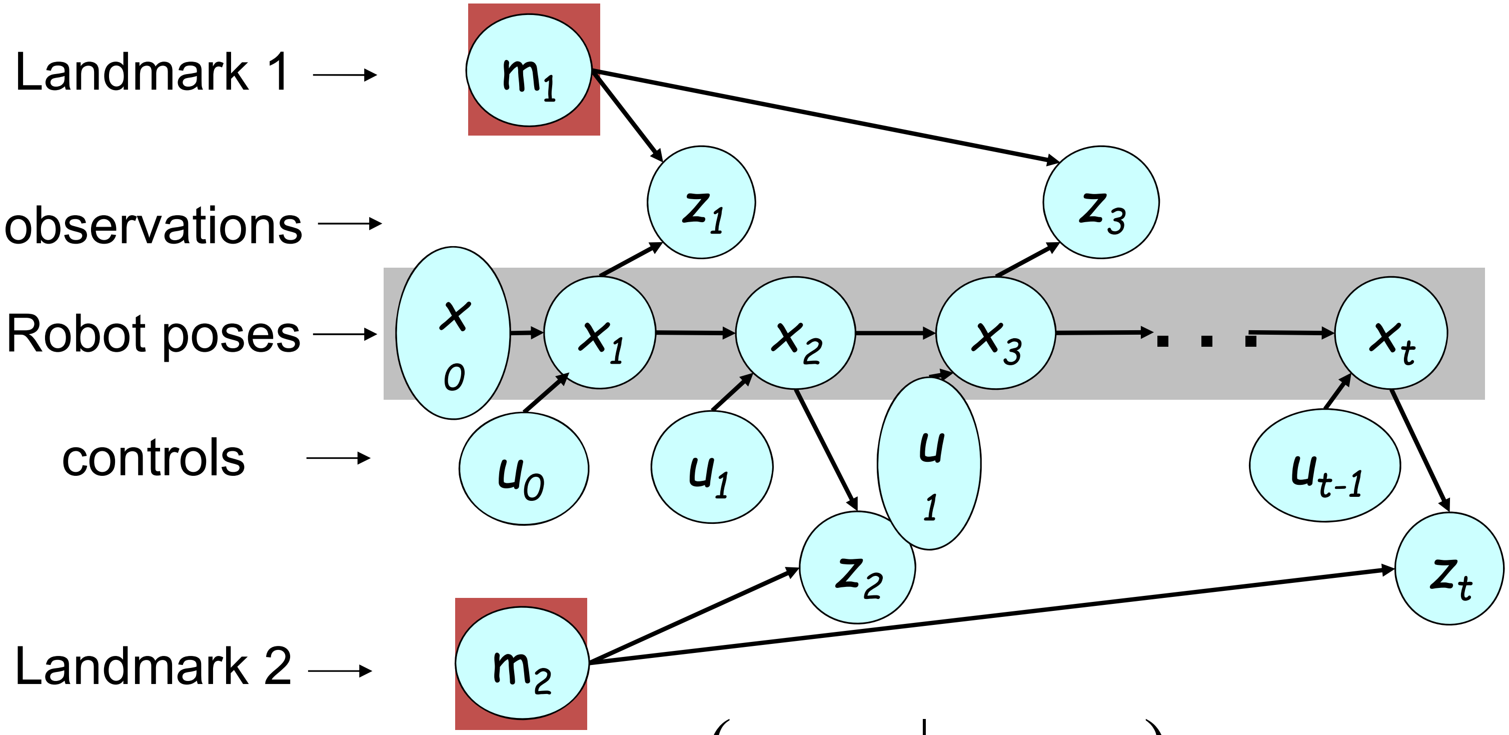
SLAM overview

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.
- In this case, the observation is called ***loop closure detection***.
- When a loop closure is detected, the robot pose uncertainty shrinks.
- At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce



SLAM

- No Map Available and No Pose Info



Landmark 1 →

observations →

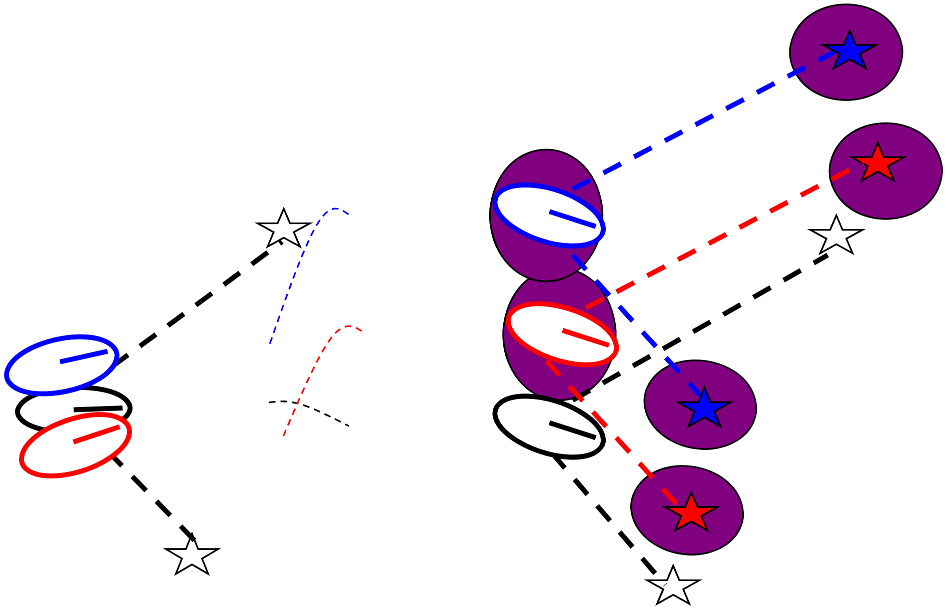
Robot poses →

controls →

Landmark 2 →

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

SLAM



- The observation model $p(z_t \mid x_t, m)$ makes the dependence between observations and robot locations.
- The joint posterior cannot be partitioned because there is dependence between observations and robot locations.

$$p(x_t, m \mid z_t) \neq P(x_t \mid z_t) P(m \mid z_t)$$

Three SLAM paradigms

- Most of the SLAM algorithms are based on the following three different approaches:
  - Extended Kalman Filter SLAM: (called EKF SLAM)
  - Particle Filter SLAM: (called FAST SLAM)
  - Graph-Based SLAM

EKF SLAM: overview

- The EKF SLAM proceeds exactly like the standard EKF that we have seen for robot localization, with the only difference that it uses an **extended state vector** $\boldsymbol{y_t}$ which comprises both the robot pose $\boldsymbol{x_t}$ and the position of all the features $\boldsymbol{m_i}$ in the map, that is:

$$y_t = [x_t, m_1, \ldots, m_{n-1}]^T$$

- If we sense 2D line-landmarks, the size of $\boldsymbol{y_t}$ would be $3+2n$, since we need three variables to represent the robot pose and $2n$ variables for the n line-landmarks having vector components

$$(\alpha_i, r_i)$$

$$y_t = [x_t, y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$$

- As the robot moves and takes measurements, the state vector and covariance matrix are updated using the standard equations of the extended Kalman filter

EKF-SLAM

- Extended Kalman Filter approximates the posterior as a Gaussian

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

- Mean (state vector) contains robot location and map points

$$\mu_t = \{x_t, m_1, \ldots, m_M\}$$

- Covariance Matrix estimates uncertainties and relationships between each element in state vector

$$\Sigma_t = [\mu_t \mu_t^T]$$

EKF based SLAM: prediction phase

- During the prediction phase, the robot pose is updated using the odometric position update formula

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

- The position of the features, will conversely remain unchanged.
- Therefore, we can write the prediction model of the EKF SLAM as

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \\ \hat{\alpha}_0 \\ \hat{r}_0 \\ \cdots \\ \hat{\alpha}_{n-1} \\ \hat{r}_{n-1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \alpha_0 \\ r_0 \\ \cdots \\ \hat{\alpha}_{n-1} \\ r_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \\ 0 \\ 0 \\ \cdots \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{P}_t = F_y P_{t-1} F_y^{\ T} + F_u Q_t F_u^{\ T}$$

EKF based SLAM: Comparison with EKF localization

LOCALIZATION

- The state X is ONLY the robot configuration, that is:

$$x_t = [x_t, y_t, \theta_t]^T$$

- The prediction function is:

$$\hat{x}_t = f(x_{t-1}, u_t)$$

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

$$\hat{P}_t = F_x P_{t-1} F_x{}^T + F_u Q_t F_u{}^T$$

SLAM

- The state Y is the robot configuration X **plus** the features $m_i=(r_i,\varphi_i)$ that is:

$$y_t = [x_t, y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$$

- The prediction function is:

$$\hat{y}_t = f(y_{t-1}, u_t)$$

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \\ \hat{\alpha}_0 \\ \hat{r}_0 \\ \ldots \\ \hat{\alpha}_{n-1} \\ \hat{r}_{n-1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \alpha_0 \\ r_0 \\ \ldots \\ \hat{\alpha}_{n-1} \\ r_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \\ 0 \\ 0 \\ \ldots \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{P}_t = F_y P_{t-1} F_y{}^T + F_u Q_t F_u{}^T$$

EKF based SLAM: estimation phase

- The observation model is the same as the EKF localization:

$$\hat{z}_i = \begin{bmatrix} \hat{\alpha}_i \\ \hat{r}_i \end{bmatrix} = h(x)$$

- The estimation proceeds in the same manner as the conventional EKF:

$$y_t = \hat{y}_t + K_t(z - h(x))$$

$$P_t = \hat{P}_t - K_t \Sigma_{IN} K_t^T$$

where

$$\Sigma_{IN} = HPH^T + R$$

$$K_t = PH(\Sigma_{IN})^{-1}$$

EKF SLAM: considerations

- At start, when the robot takes the first measurements, the covariance matrix is populated by assuming that these (initial) features are uncorrelated, which implies that the off diagonal elements are set to zero.

$$P_0 = \begin{bmatrix} P_x & 0 & 0 & \dots & 0 & 0 \\ 0 & P_{m_0} & 0 & \dots & 0 & 0 \\ 0 & 0 & P_{m_1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & P_{m_{n-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & P_{m_{n-1}} \end{bmatrix}$$

EKF SLAM: considerations

- However, when the robot starts moving and takes new measurements, both the robot pose and features start becoming correlated.

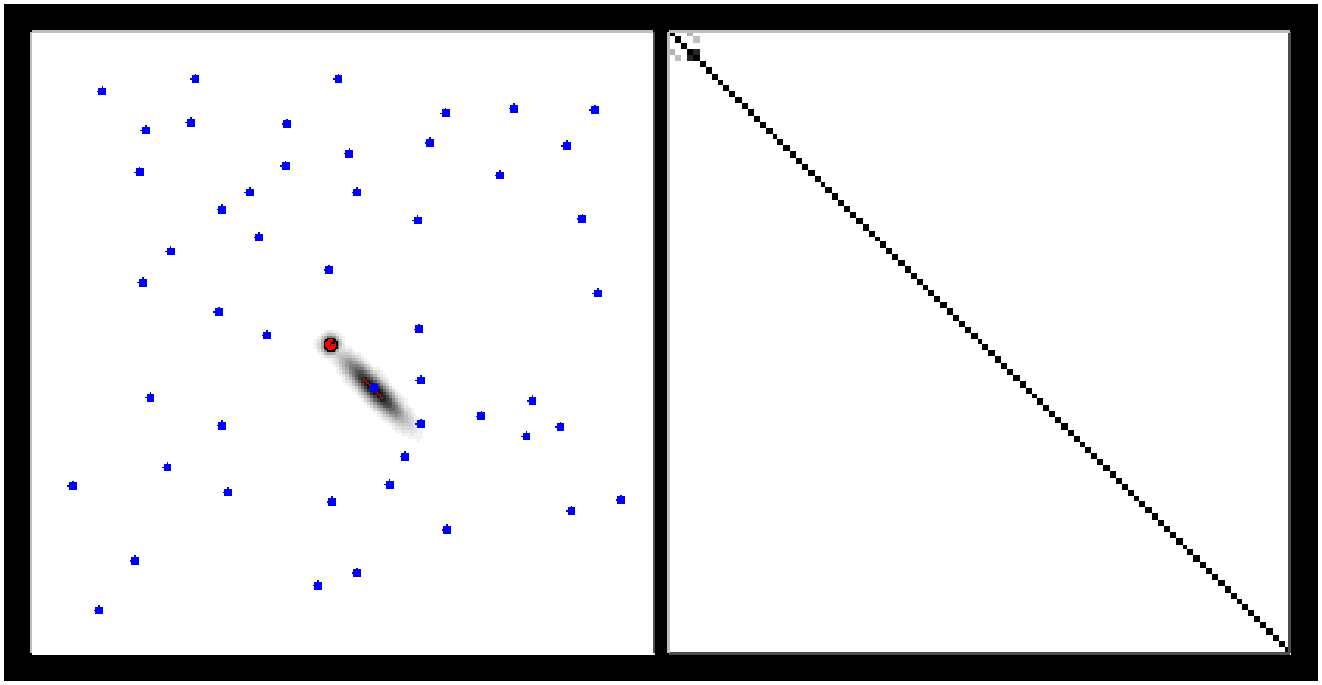
$$\hat{P}_t = F_y P_{t-1} F_y^{\,T} + F_u Q_t F_u^{\,T}$$

- Accordingly, the covariance matrix becomes *non-sparse*.

$$P_0 = \begin{bmatrix} P_x & P_{xm_0} & P_{xm_1} & \dots & P_{xm_{n-2}} & P_{xm_{n-1}} \\ P_{xm_0} & P_{m_0} & P_{m_0m_1} & \dots & P_{m_0m_{n-2}} & P_{m_0m_{n-1}} \\ P_{xm_1} & P_{m_0m_1} & P_{m_1} & \dots & P_{m_1m_{n-2}} & P_{m_1m_{n-1}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{xm_{n-2}} & P_{m_0m_{n-2}} & P_{m_1m_{n-2}} & \dots & P_{m_{n-2}} & P_{m_1m_{n-1}} \\ P_{xm_{n-1}} & P_{m_0m_{n-1}} & P_{m_1m_{n-1}} & \dots & P_{m_{n-2}m_{n-1}} & P_{m_{n-1}} \end{bmatrix}$$

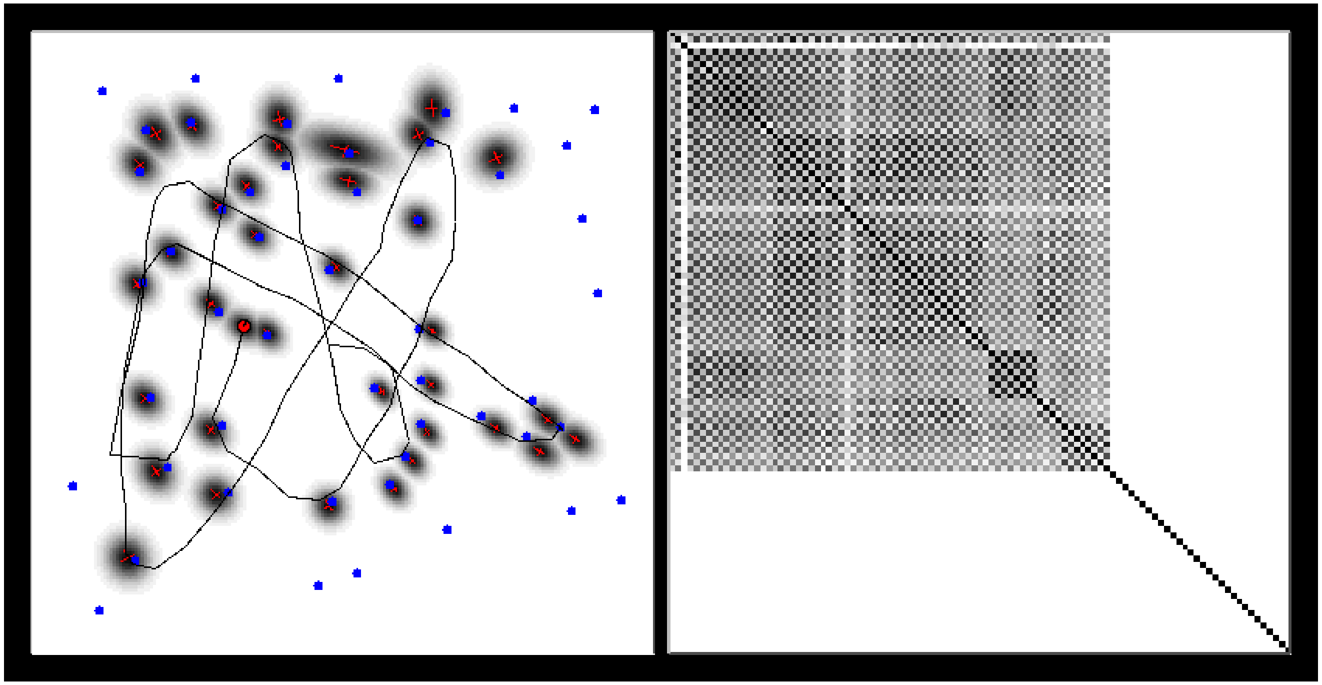
- The existence of this correlation can be explained by recalling that the uncertainty of the features in the map depends on the uncertainty associated to the robot pose. But it also depends on the uncertainty of other features that have been used to update the robot pose. This means that when a new feature is observed this contributes to correct not only the estimate of the robot pose but also that of the other features as well. The more observations are made, the more the correlations between the features will grow, the better the solution to SLAM.

EKF-SLAM



Map Correlation matrix

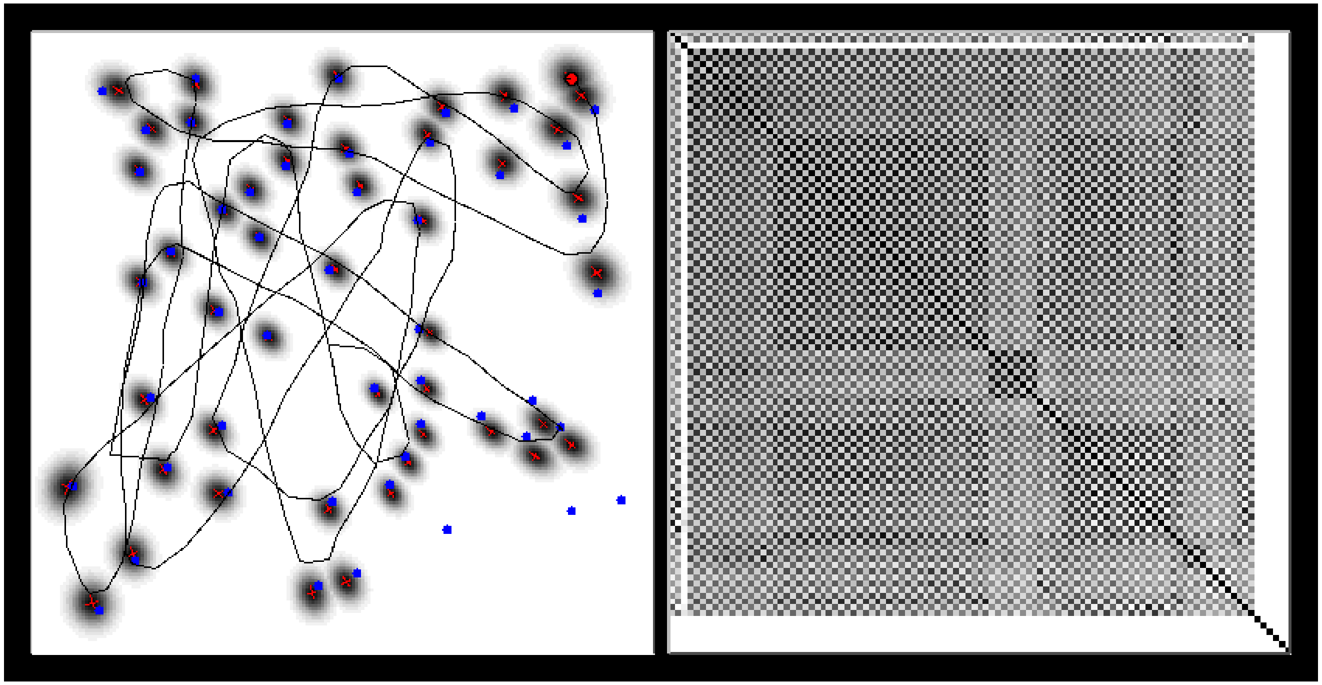
EKF-SLAM



Map

Correlation matrix

EKF-SLAM



Map Correlation matrix

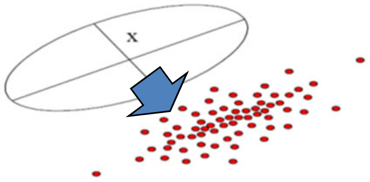
Drawbacks of EKF SLAM

- Clearly, the state vector in EKF SLAM is much larger than the state vector in EKF localization where only the robot pose was being updated. This makes EKF SLAM computationally very expensive.
- Notice that, because of its formulation, maps in EKF SLAM are supposed to be feature based (i.e. points, lines, planes). As new features are observed, they are added to the state vector. Thus, the noise covariance matrix grows quadratically, with size (3+2n)x(3+2n). For computational reasons, the size of the map is therefore usually limited to less than 1,000 features.

Particle Filter SLAM: FastSLAM

- **FastSLAM approach**
  - It solves the SLAM problem using particle filters.
  - Each particle k :estimate of robot path and mean, and covariance of each of the n features: $P^{[k]}$ $(X_t^{[k]}$ $\mu^{[k];}$ $\Sigma_1^{[k]}$ … $\mu^{[k]}$ $\Sigma_n^{[k]}$)

- Particle filter update
  - In the update step a new particle distribution, given motion model and controls applied is generated.



probability distribution (ellipse) as particle set (red dots)

a) For each particle:

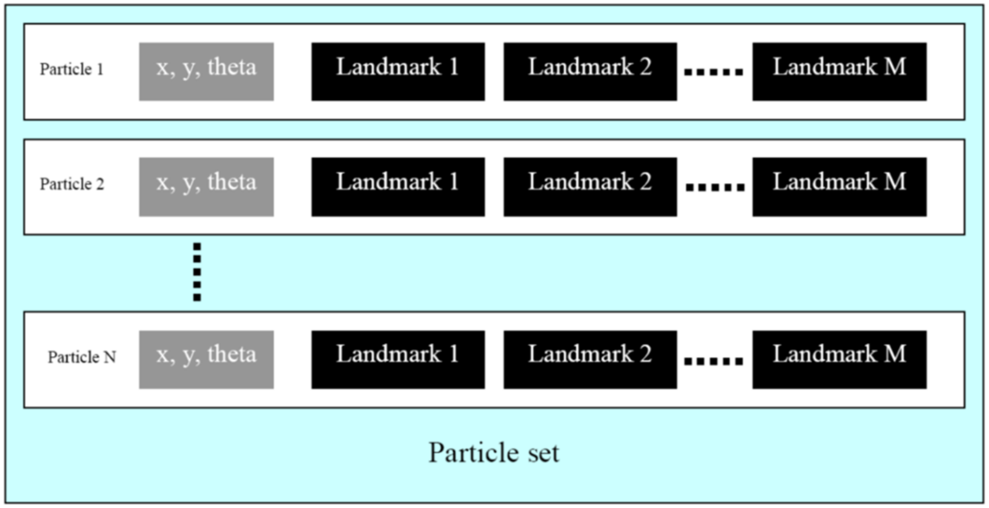
1. Compare particle’s prediction of measurements with actual measurements
2. Particles whose predictions match the measurements are given a high weight

b) Filter resample:

- Resample particles based on weight
- Filter resample
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.

Particle Filter SLAM

- FastSLAM approach
  - Particle set
    - The robot posterior is solved using a Rao Blackwellized particle filtering using landmarks. Each landmark estimation is represented by a 2x2 EKF (Extended Kalman Filter). Each particle is “independent” (due the factorization) from the others and maintains the estimate of M landmark positions.



SLAM algorithm

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

- Prediction

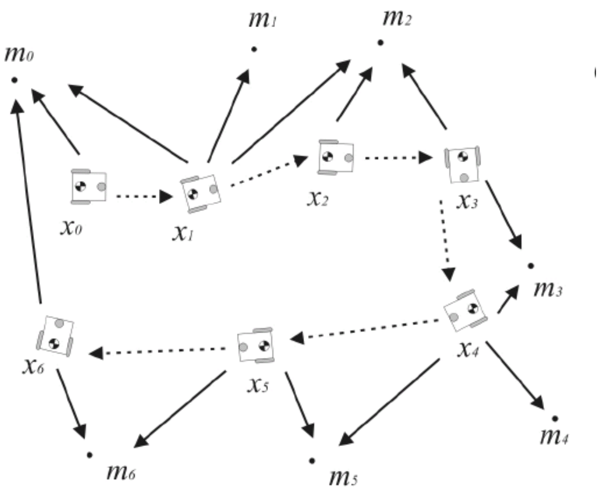
$$\overline{bel}(x_t, m) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1}, m)\, dx_{t-1}$$

- Update

$$bel(x_t, m) = \eta\, p(z_t \mid x_t, m)\, \overline{bel}(x_t, m)$$

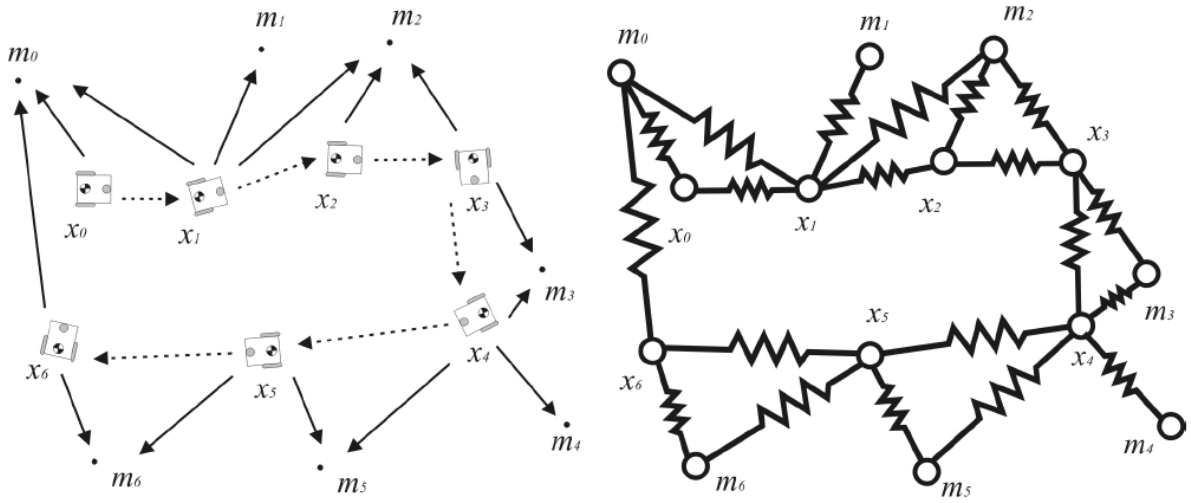
Graph-Based SLAM

- Graph-based SLAM is born from the intuition that the SLAM problem can be interpreted as a sparse graph of nodes and constraints between nodes.
- The nodes of the graph are the robot locations and the features in the map.
- The constraints are the relative position between consecutive robot poses , (given by the odometry input ***u***) and the relative position between the robot locations and the features observed from those locations.



Graph-Based SLAM

- The key property to remember about graph-based SLAM is that the constraints are not to be thought as rigid constraints but as soft constraints.
- By relaxing these constraints that we can compute the solution to the full SLAM problem, that is, the best estimate of the robot path and the environment map. By saying this in other words, graph-based SLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net



Graph-Based SLAM

- There is a significant advantage of graph-based SLAM techniques over EKF SLAM. As we have seen, in EKF SLAM the amount of computation and memory requirement to update and store the covariance matrix grows quadratically in the number of features. Conversely, in graph-based SLAM the update time of the graph is constant and the required memory is linear in the number of features.
- However, the final graph optimization can become computationally costly if the robot path is long.

Learning objectives SLAM

- Whats is SLAM
- Terms
- Observation model - motion model
- Full SLAM vs Online SLAM
- SLAM approaches
  - EKF
  - Particle filter
  - Graph based

More reading

- Simultaneous Localization and Mapping: Part I - Hugh Durrant-Whyte and Tim Bailey
- Simultaneous Localization and Mapping: Part II - Hugh Durrant-Whyte and Tim Bailey
- Review of Underwater SLAM Techniques, Hidalgo, Bräunl
- Feature extraction for underwater visual SLAM, Aulinas, Carreras, Llado, Salvi, Garcia and Ricard Prados, Petillot