The SLAM problem
Autonomous Mobile Robots

Margarita Chli – University of Edinburgh

Paul Furgale, Marco Hutter, Martin Rufli, Davide Scaramuzza, Roland Siegwart

SLAM | Simultaneous Localization and Mapping

The SLAM problem:

How can a body **navigate** in a previously unknown environment while constantly building and updating a **map** of its workspace using on board sensors only?

- When is SLAM necessary?
  - When a robot must be truly **autonomous** (no human input)
  - When there is **no prior** knowledge about the environment
  - When we cannot place **beacons** and cannot use **external positioning systems** (e.g. GPS)
  - When the robot needs to know where it is

SLAM | the chicken and egg problem

- An unbiased **map** is necessary for localizing the robot

  Pure localization with a known map.

  SLAM: no *a priori* knowledge of the robot's workspace

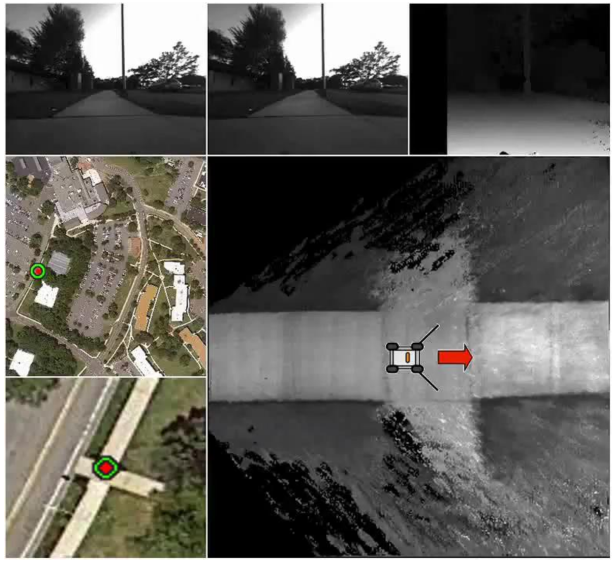
- An accurate **pose estimate** is necessary for building a map of the environment

  Mapping with known robot poses.

  SLAM: the robot poses have to be estimated along the way

- SLAM: one of the greatest challenges in probabilistic robotics

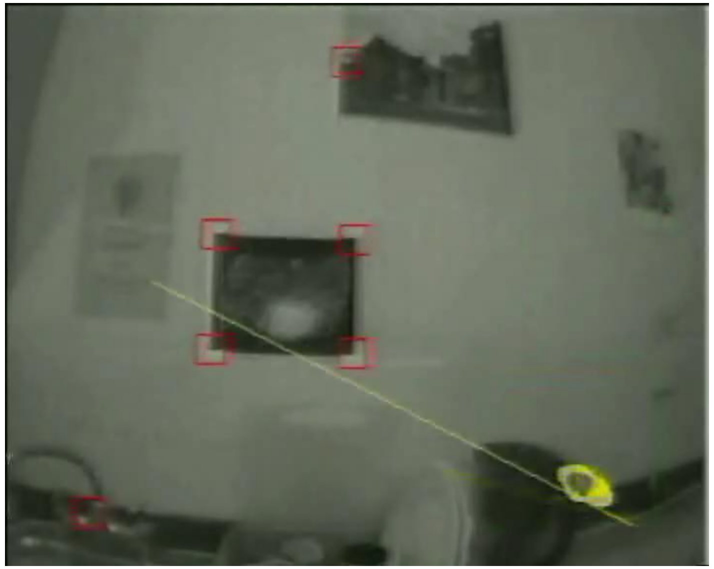
Localization using satellite images
[Senlet and Elgammal, ICRA 2012]



Helicopter pose given by the Leica tracker
Video courtesy of Simon Lynen

SLAM | perceiving motion

- Can we track the motion of a camera/robot while it is moving?



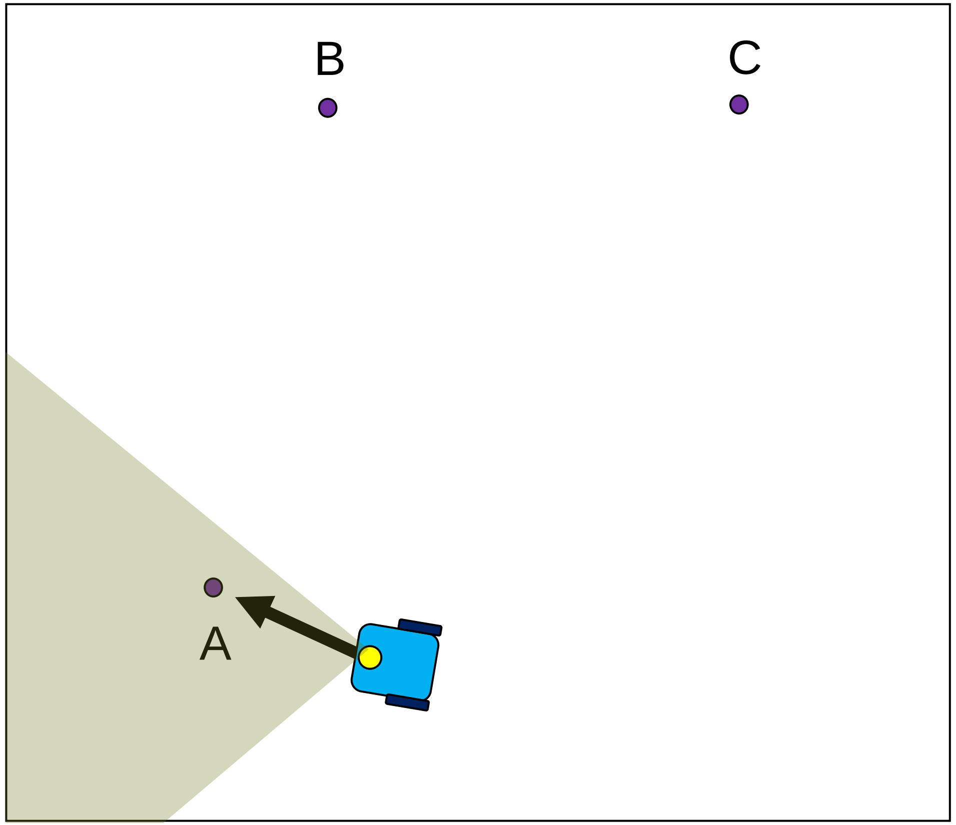
Courtesy of Andrew J. Davison

- Pick natural scene features to serve as landmarks
  (in most modern SLAM systems)
- Range sensing (laser/sonar): line segments, 3D planes,…
- Vision: point features, lines, textured surfaces.

- **Key**: features must be distinctive & recognizable from different viewpoints

SLAM | how to do SLAM

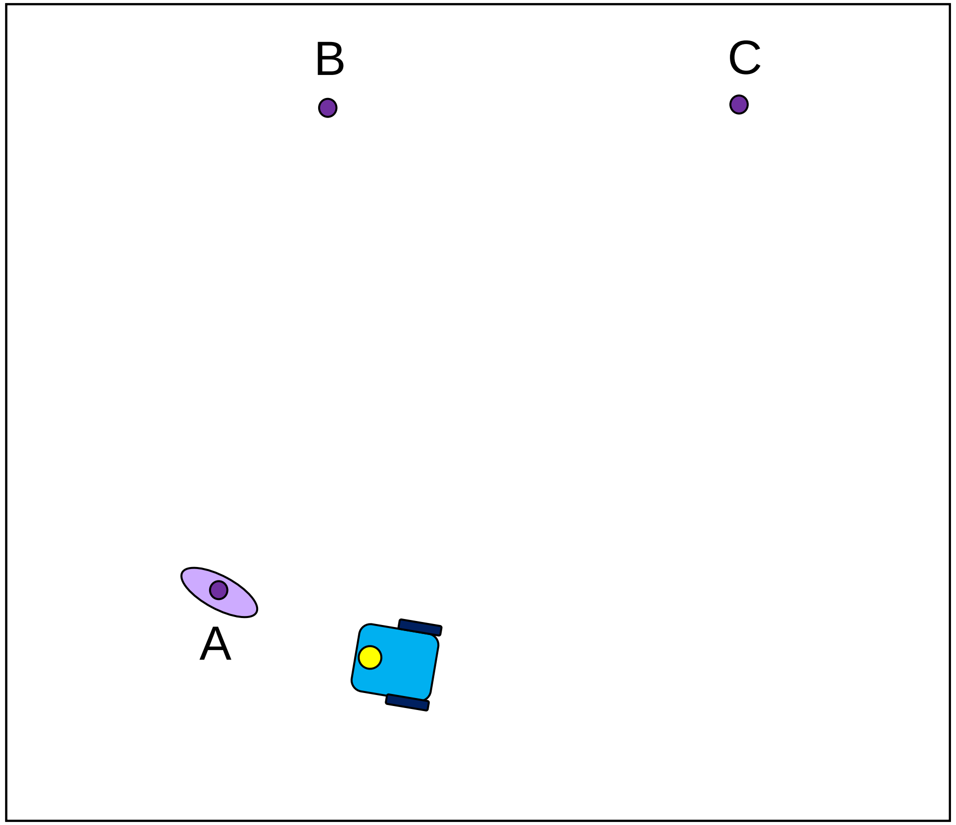
On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



First measurement of feature A

SLAM | how to do SLAM

- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**

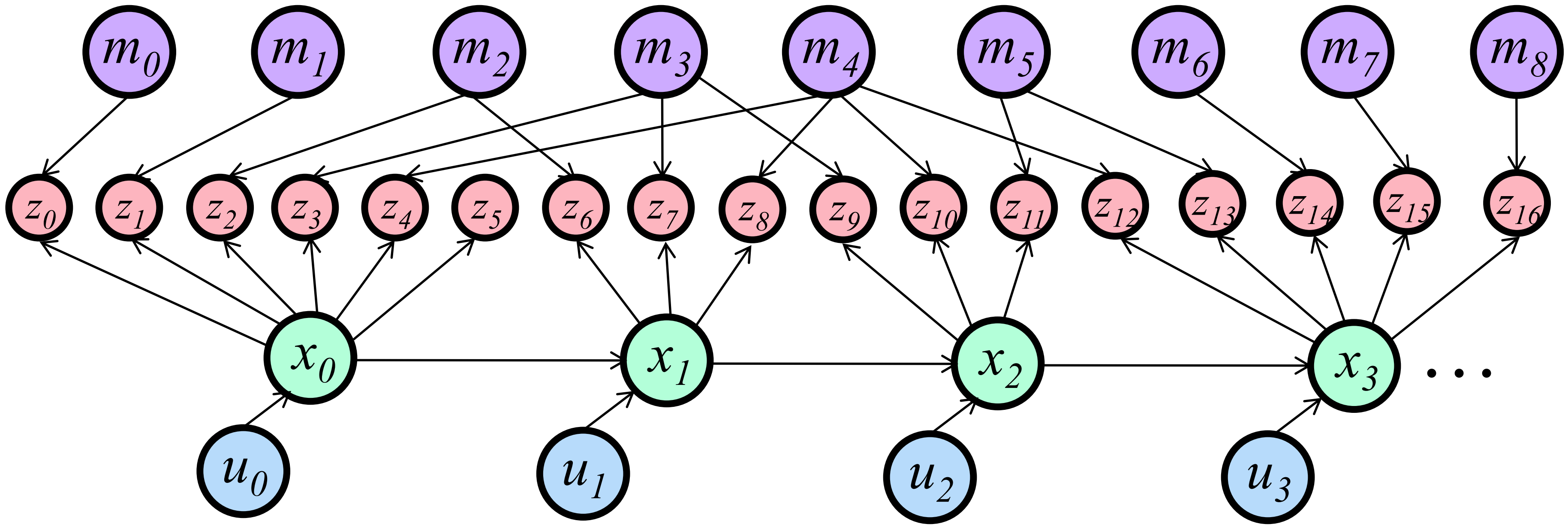


On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

SLAM | probabilistic formulation

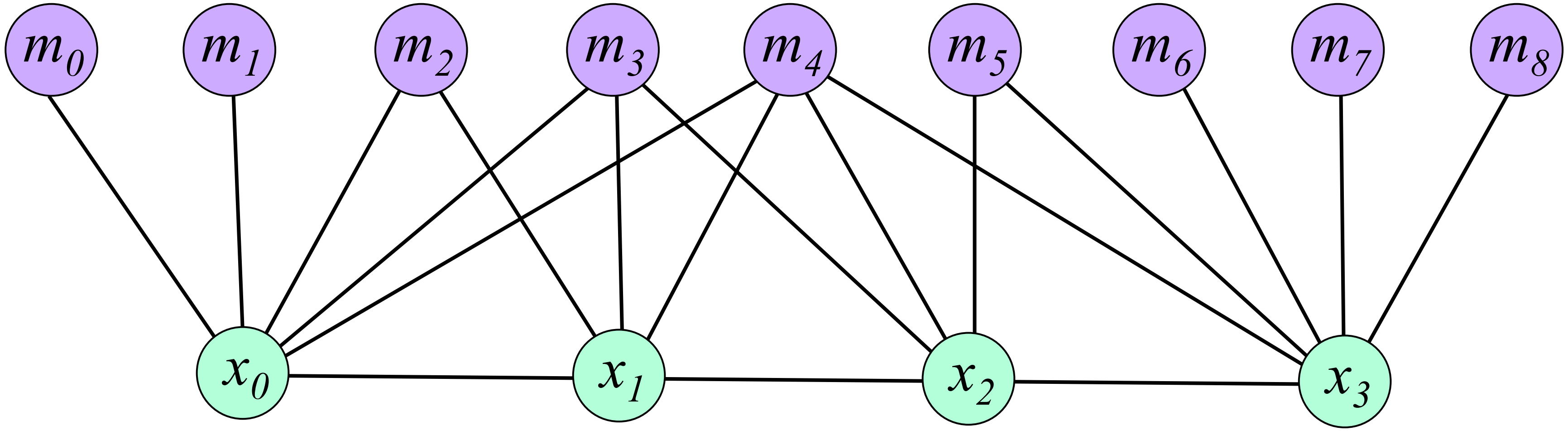
- Robot **pose** at time t : x_t ⇨ Robot **path** up to this time: $\{x_0, x_1, \ldots, x_t\}$
- Robot **motion** between time t-1 and t : u_t (control inputs / proprioceptive sensor readings)
  ⇨ Sequence of robot relative motions: $\{u_0, u_1, \ldots, u_t\}$
- The **true map** of the environment: $\{m_0, m_1, \ldots, m_N\}$
- At each time t the robot makes measurements z_i
  ⇨ Set of all measurements (observations): $\{z_0, z_1, \ldots, z_k\}$
- The Full SLAM problem: estimate the posterior $p(x_{0:t}, m_{0:n} \mid z_{0:k}, u_{0:t})$
- The Online SLAM problem: estimate the posterior $p(x_t, m_{0:n} \mid z_{0:k}, u_{0:t})$

SLAM | graphical representation



SLAM | approaches

- **Full graph optimization** (bundle adjustment)

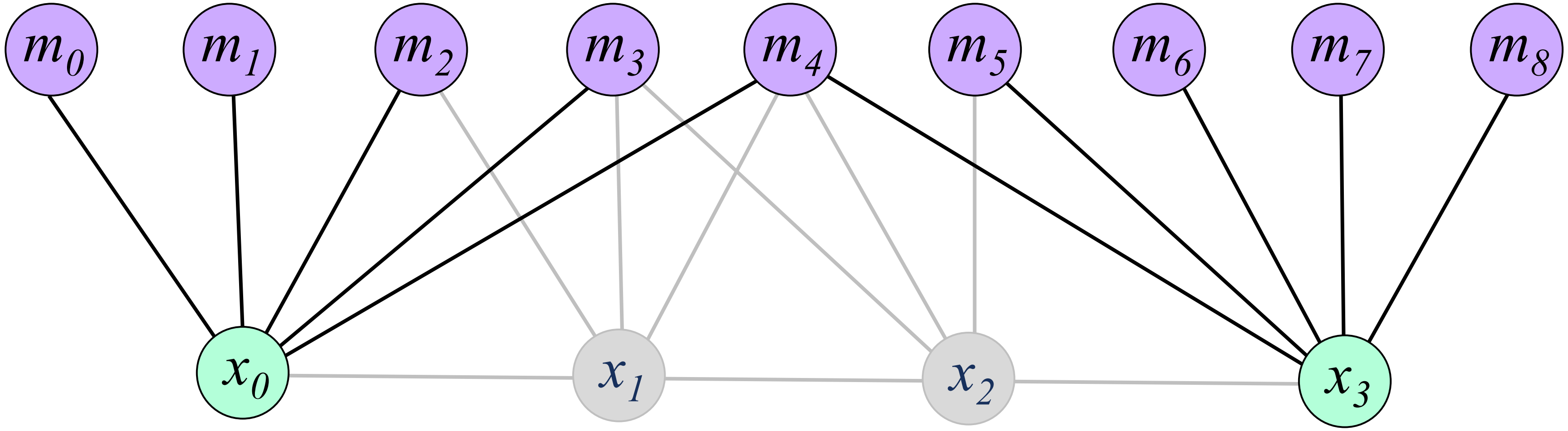


- Eliminate observations & control-input nodes and solve for the constraints between poses and landmarks.
- Globally consistent solution, but infeasible for large-scale SLAM

⇨ If real-time is a requirement, we need to **sparsify** this graph

SLAM | approaches

- **Key-frames**

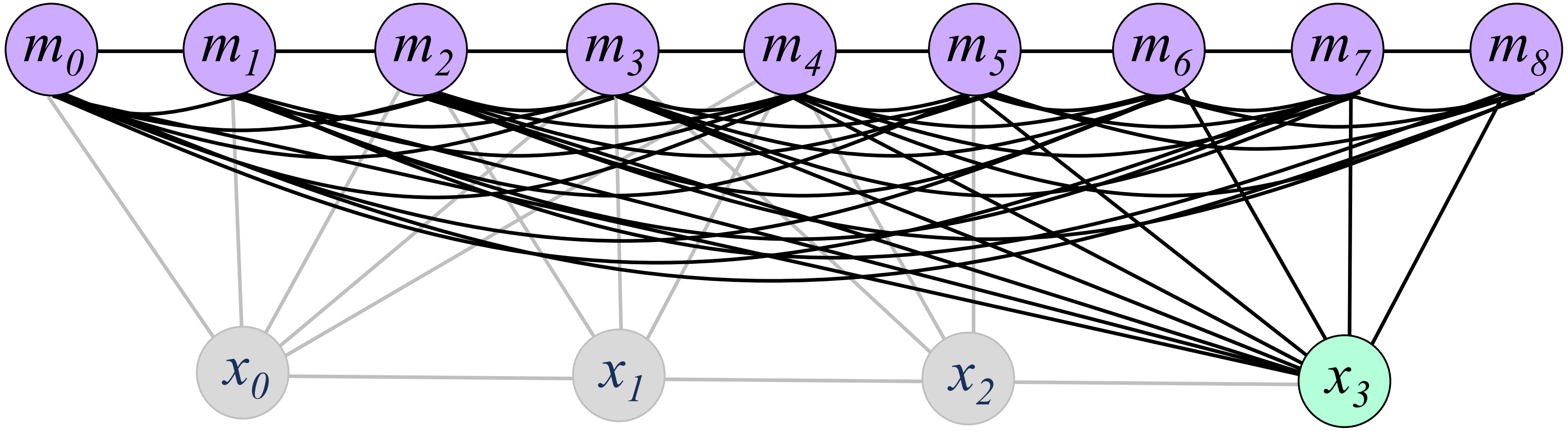


- Retain the most ‘representative’ poses (key-frames) and their dependency links ⇨ optimize the resulting graph

- Example: PTAM [Klein & Murray, ISMAR 2007]

SLAM | approaches

- **Filtering**



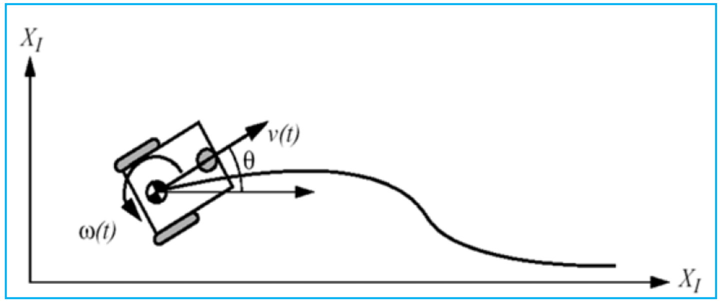
- Eliminate all past poses: ‘summarize’ all experience with respect to the last pose, using a **state vector** and the associated **covariance matrix**

- Example: [MonoSLAM [Davison et al., PAMI 2007]]

EKF SLAM | overview

- All past experience is summarized an **extended state vector** $\boldsymbol{y_t}$ comprising of the robot pose $\boldsymbol{x_t}$ and the position of all the features $\boldsymbol{m_i}$ in the map, and an associated **covariance matrix** $\boldsymbol{P_{yt}}$:

$$y_t = \begin{bmatrix} x_t \\ m_1 \\ \dots \\ m_{n-1} \end{bmatrix} \quad , \quad P_{y_t} = \begin{bmatrix} P_{xx} & P_{xm_1} & .. & P_{xm_{n-1}} \\ P_{m_1x} & P_{m_1m_1} & .. & P_{m_1m_{n-1}} \\ .. & .. & .. & .. \\ P_{m_{n-1}x} & P_{m_{n-1}m_1} & .. & P_{m_{n-1}m_{n-1}} \end{bmatrix}$$



- If we sense 2D line-landmarks, the size of $\boldsymbol{y_t}$ is $3+2n$
  (and size of $\boldsymbol{P_t}$: $(3+2n)(3+2n)$)
  - 3 variables to represent the robot pose and
  - $2n$ variables for the n line-landmarks with state components (α_i, r_i)

  Hence, $y_t = [X_t, Y_t, \theta_t, \alpha_0, r_0, \dots, \alpha_{n-1}, r_{n-1}]^T$

- As the robot moves and makes measurements, $\boldsymbol{y_t}$ and $\boldsymbol{P_{yt}}$ are updated using the **standard EKF equations**

EKF SLAM | prediction

- The predicted robot pose $\hat{x}_t$ at time-stamp t is computed using the estimated pose x_{t-1} at time-stamp t-1 and the odometric control input $u_t = \{\Delta S_l, \Delta S_r\}$

$$\hat{x}_t = f(x_{t-1}, u_t) = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} X_{t-1} \\ Y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2}\cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2}\sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

Odometry function

$\Delta S_l ; \Delta S_r$: distance travelled by the left and right wheels resp.
b : distance between the two robot wheels
(based on the example of Section 5.8.4 of the AMR book)

- During this step, the position of the features remains unchanged. EKF Prediction Equations:

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

Jacobians of f

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

Covariance at previous time-stamp

Covariance of noise associated to the motion

EKF SLAM | comparison with EKF localization

EKF LOCALIZATION

- The state x_t is **only** the robot configuration:

$$x_t = [X_t, Y_t, \theta_t]^T$$

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

$$\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} X_{t-1} \\ Y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

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

EKF SLAM

- The state y_t comprises of the robot configuration x_t **and** that of each feature m_i :

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

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

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

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

EKF SLAM | measurement prediction & update

- The application of the **measurement model** is the same as in EKF localization. The predicted observation of each feature m_i is:

$$\hat{z}_i = \begin{bmatrix} \hat{\alpha}_i \\ \hat{r}_i \end{bmatrix} = h_i(\hat{x}_t, m_i)$$

The predicted new pose is used to predict where each feature lies in measurement space

- After obtaining the set of **actual** observations $z_{0:n-1}$ the EKF state gets updated:

$$y_t = \hat{y}_t + K_t(z_{0:n-1} - h_{0:n-1}(\hat{x}_t, m_{0:n-1}))$$

$$P_{y_t} = \hat{P}_{y_t} - K_t \Sigma_{IN} K_t^T$$

where

$$\Sigma_{IN} = \underbrace{H}_{\text{Jacobian of } h} \hat{P}_{y_t} H^T + \underbrace{R}_{\text{Measurement noise}}$$

$$\underbrace{K_t}_{\text{Kalman Gain}} = \hat{P}_{y_t} H (\underbrace{\Sigma_{IN}}_{\text{Innovation Covariance}})^{-1}$$