SLAM: Simultaneous Localization and Mapping

Javier González Jiménez

Reference Books:

- Probabilistic Robotics. S. Thrun, W. Burgard, D. Fox. MIT Press. 2001
- Simultaneous Localization and Mapping for Mobile Robots: Introduction and Methods. Juan-Antonio Fernández-Madrigal and José Luis Blanco Claraco. IGI-Global. 2013.

Content

SLAM

Landmarks and poses unknown → The state to estimate comprises poses+map

Two possibilities:

- The Full SLAM problem: estimates the posterior of the complete **robot path**

$$p(x_{1:k}, m_{1:L} | z_{1:k}, u_{1:k})$$

All the poses until instant k (the complete robot path)!

- The Online SLAM problem: estimates the posterior of the **most recent robot pose**

$$p(x_k, m_{1:L} | z_{1:k}, u_{1:k})$$

Only the last robot pose!

Assumption: Data association is given, that is, we know which feature is being seen by each observation

EKF-SLAM (for online SLAM)

State:

Like in the mapping case, but including the robot pose

$$s_k = [x_k \mid m_{x1}\, m_{y1} \cdots m_{xL}\, m_{yL}]^T = [x_k \mid \mathbf{m}]^T$$

dim(s_k) = 3+2L

Robot pose at instant k
(x_k, y_k, θ_k)

m: Landmarks of the maps (x, y)

L: number of landmarks

Every time a new landmark is observed s_k augments in (m_x, m_y)

We assume $s_k \sim N(\mu_{s_k}, \Sigma_k)$

Covariance of the pose (3x3)

Correlation between pose and landmarks

$$\Sigma_k = \begin{bmatrix} \Sigma_{x_k} & \Sigma_{xm_k} \\ \Sigma_{xm_k}^T & \Sigma_{m_k} \end{bmatrix}_{(3+2L)x(3+2L)}$$

Covariance of the Landmarks (2Lx2L)

Recall: Correlation means that error in x_k affects error in $\mathbf{m}$ (and the opposite)

EKF-SLAM

Prediction: MEAN

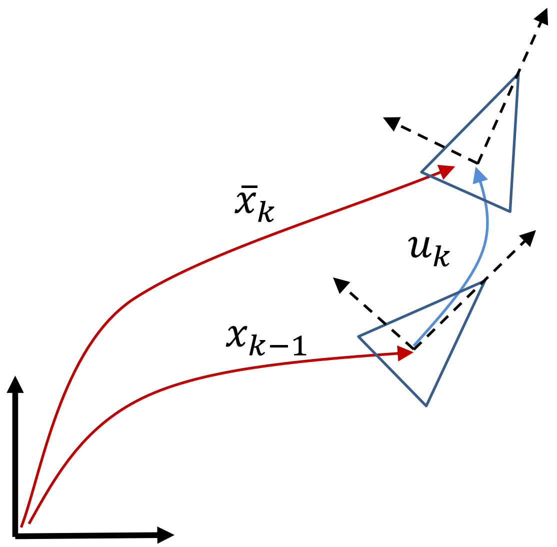
$$\bar{s}_k = \begin{bmatrix} \bar{x}_k \\ \bar{m}_k \end{bmatrix} = g(s_{k-1}, u_k) = \begin{bmatrix} x_{k-1} \oplus u_k \\ m_{k-1} \end{bmatrix}$$

We take the mean as the best estimate avaliable

This means **"predicted"** (not mean!)

Only the robot pose changes, the **map is static**

Note: u_k is not velocity here, but pose increment $u_k = \Delta \boldsymbol{x}_t$



RECALL: $u_k = \Delta \boldsymbol{x}_t = [\Delta x_t, \Delta y_t, \Delta \theta_t]^T$

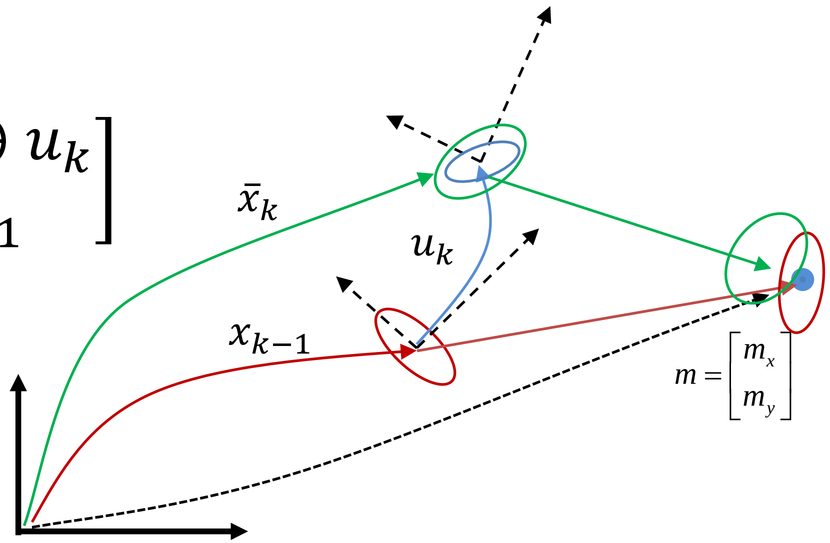
$$\bar{x}_k = x_{k-1} \oplus u_k = \begin{bmatrix} x_{k-1} + \Delta x_t \cos\theta_{k-1} - \Delta y_t \sin\theta_{k-1} \\ y_{k-1} + \Delta x_t \sin\theta_{k-1} + \Delta y_t \cos\theta_{k-1} \\ \Delta\theta_t + \theta_{k-1} \end{bmatrix}$$

```
xVehicle = xEst(1:3); %Robot pose: the first 3 elements of the state vector
xMap = xEst(4:end); %Map the remaining elements
xVehiclePred = tcomp(xVehicle,u); %Predictive mean pose
xPred = [xVehiclePred;xMap]; %Predictive mean state vector
```

EKF-SLAM

Prediction: COVARIANCE

$$\bar{s}_k = \begin{bmatrix} \bar{x}_k \\ \bar{m}_k \end{bmatrix} = g(s_{k-1}, u_k) = \begin{bmatrix} x_{k-1} \oplus u_k \\ m_{k-1} \end{bmatrix}$$



RECALL: if $Z=f(X,Y)$

$$\boxed{\Sigma_Z = \frac{\partial f}{\partial X}\Sigma_X\left(\frac{\partial f}{\partial X}\right)^T + \frac{\partial f}{\partial Y}\Sigma_Y\left(\frac{\partial f}{\partial Y}\right)^T}$$

$$\bar{\Sigma}_k \approx \underbrace{\frac{\partial g}{\partial s_{k-1}}\Sigma_{k-1}\frac{\partial g}{\partial s_{k-1}}^T}_{\text{Covariance due to the uncertainty in the previous (k-1) state}} + \underbrace{\frac{\partial g}{\partial u_k}\Sigma_{u_k}\frac{\partial g}{\partial u_k}^T}_{\text{Covariance due to the uncertainty of the robot motion}}$$

$\bar{\Sigma}_k$ is the sum of two covariance matrices → Increase of uncertainty!

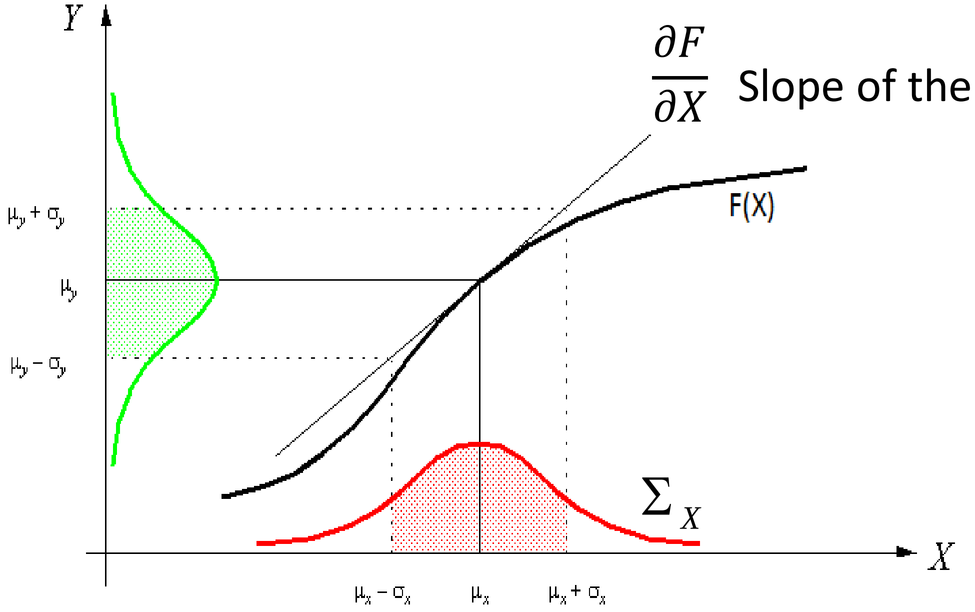
RECALL 1: **Jacobian of a vector function**

$$\boldsymbol{F}(x_1, x_2) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{bmatrix}$$

$$J_{F(x_1, x_2)} = \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{x}} = \frac{\partial \{f_1, f_2\}}{\partial \{x_1, x_2\}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}$$

RECALL 2: **Covariance of a RV** *Y=F(X)*

$$\Sigma_Y \approx \frac{\partial F}{\partial X} \Sigma_X \left(\frac{\partial F}{\partial X}\right)^T$$



Notice that X and Y may be of different dimesions: $X \in \mathbb{R}^m$, $Y \in \mathbb{R}^n$

$$\underbrace{\Sigma_Y}_{\text{nxn}} \approx \underbrace{\frac{\partial F}{\partial X}}_{\text{nxm}} \underbrace{\Sigma_X}_{\text{mxm}} \underbrace{\left(\frac{\partial F}{\partial X}\right)^T}_{\text{mxn}}$$

$$\bar{\Sigma}_k \approx \boxed{\frac{\partial g}{\partial s_{k-1}} \Sigma_{k-1} \frac{\partial g}{\partial s_{k-1}}^T} + \frac{\partial g}{\partial u_k} \Sigma_{u_k} \frac{\partial g}{\partial u_k}^T$$

$$\bar{s}_k = \begin{bmatrix} \bar{x}_k \\ \bar{m}_k \end{bmatrix} = g(s_{k-1}, u_k) = \begin{bmatrix} x_{k-1} \oplus u_k \\ m_{k-1} \end{bmatrix}$$

Let's compute the two additive terms:

$$\frac{\partial \bar{s}_k}{\partial s_{k-1}} = \frac{\partial g(s_{k-1}, u_k)}{\partial s_{k-1}} = \frac{\partial \{\bar{x}_k, \bar{m}_k\}}{\partial \{x_{k-1}, m_{k-1}\}} = \begin{bmatrix} \frac{\partial \bar{x}_k}{\partial x_{k-1}} & \frac{\partial \bar{x}_k}{\partial m_{k-1}} \\ \frac{\partial \bar{m}_k}{\partial x_{k-1}} & \frac{\partial \bar{m}_k}{\partial m_{k-1}} \end{bmatrix}_{(3+2L)x(3+2L)} = \begin{bmatrix} \frac{\partial \bar{x}_k}{\partial x_{k-1}} & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} J_1 & 0 \\ 0 & I \end{bmatrix}$$

$$\frac{\partial \bar{s}_k}{\partial s_{k-1}} \Sigma_{k-1} \frac{\partial \bar{s}_k}{\partial s_{k-1}}^T = \begin{bmatrix} J_1 & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} \Sigma_{x_{k-1}} & \Sigma_{xm_{k-1}} \\ \Sigma_{xm_{k-1}}^T & \Sigma_{m_{k-1}} \end{bmatrix} \begin{bmatrix} J_1^T & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} J_1 \Sigma_{x_{k-1}} J_1^T & J_1 \Sigma_{xm_{k-1}} \\ \Sigma_{xm_{k-1}}^T J_1^T & \Sigma_{x_{m-1}} \end{bmatrix}$$

$$\boxed{\bar{\Sigma}_k \approx \frac{\partial g}{\partial s_{k-1}} \Sigma_{k-1} \frac{\partial g}{\partial s_{k-1}}^T + \boxed{\frac{\partial g}{\partial u_k} \Sigma_{u_k} \frac{\partial g}{\partial u_k}^T}}$$

$$\bar{s}_k = \begin{bmatrix} \bar{x}_k \\ \bar{m}_k \end{bmatrix} = g(s_{k-1}, u_k) = \begin{bmatrix} x_{k-1} \oplus u_k \\ m_{k-1} \end{bmatrix}$$

Now the second additive term:

$$\frac{\partial g}{\partial u_k} = \frac{\partial g(s_{k-1}, u_k)}{\partial u_k} = \frac{\partial \{\bar{x}_k, \bar{m}_k\}}{\partial u_k}$$

with $u_t = [\Delta x_t, \Delta y_t, \Delta \theta_t]^T$

$$= \begin{bmatrix} \frac{\partial \bar{x}_k}{\partial u_k} \\ \frac{\partial \bar{m}_k}{\partial u_k} \end{bmatrix} = \begin{bmatrix} \frac{\partial \bar{x}_k}{\partial \Delta x_t} & \frac{\partial \bar{x}_k}{\partial \Delta y_t} & \frac{\partial \bar{x}_k}{\partial \Delta \theta_t} \\ \frac{\partial \bar{m}_k}{\partial \Delta x_t} & \frac{\partial \bar{m}_k}{\partial \Delta y_t} & \frac{\partial \bar{m}_k}{\partial \Delta \theta_t} \end{bmatrix}_{(3+2L)x3} = \begin{bmatrix} \frac{\partial \bar{x}_k}{\partial \Delta x_t} & \frac{\partial \bar{x}_k}{\partial \Delta y_t} & \frac{\partial \bar{x}_k}{\partial \Delta \theta_t} \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} J_{2\,3x3} \\ \mathbf{0}_{\mathbf{2x3L}} \end{bmatrix}$$

$$\frac{\partial g}{\partial u_k} \Sigma_{u_k} \frac{\partial g}{\partial u_k}^T = \begin{bmatrix} J_2 \\ 0 \end{bmatrix} \begin{bmatrix} \sigma^2_{\Delta x_t} & 0 & 0 \\ 0 & \sigma^2_{\Delta x_t} & 0 \\ 0 & 0 & \sigma^2_{\Delta \theta_t} \end{bmatrix} [J_2 \quad 0] = \begin{bmatrix} J_2 \Sigma_{u_k} J_2^T & 0 \\ 0 & 0 \end{bmatrix}_{(3+2L)x(3+2L)}$$

Prediction: COVARIANCE (recap)

$$\bar{\Sigma}_k = \begin{bmatrix} \Sigma_{x_k} & \Sigma_{xm_k} \\ \Sigma_{xm_k} & \Sigma_{m_k} \end{bmatrix} \approx \frac{\partial g}{\partial s_{k-1}} \Sigma_{k-1} \frac{\partial g}{\partial s_{k-1}}^T + \frac{\partial g}{\partial u_k} \Sigma_{u_k} \frac{\partial g}{\partial u_k}^T$$

$$J_1 = \frac{\partial \bar{x}_k}{\partial x_{k-1}} = \left. \frac{\partial g(x_{k-1}, u_k)}{\partial x_{k-1}} \right|_{s_{k-1}}$$

$$J_2 = \frac{\partial \bar{x}_k}{\partial u_k} = \left. \frac{\partial g(x_{k-1}, u_k)}{\partial u_k} \right|_{s_{k-1}}$$

$$\bar{\Sigma}_k = \underbrace{\begin{bmatrix} J_1 & 0_{3x2L} \\ 0_{2Lx3} & I_{2Lx2L} \end{bmatrix} \begin{bmatrix} \Sigma_{x_{k-1}} & \Sigma_{xm_{k-1}} \\ (\Sigma_{xm_{k-1}})^T & \Sigma_{m_{k-1}} \end{bmatrix} \begin{bmatrix} J_1 & 0_{3x2L} \\ 0_{2Lx3} & I_{2Lx2L} \end{bmatrix}^T}_{\begin{bmatrix} J_1 \Sigma_{x_{k-1}} J_1^T & J_1 \Sigma_{xm_{k-1}} \\ (J_1 \Sigma_{xm_{k-1}})^T & \Sigma_{m_{k-1}} \end{bmatrix}} + \begin{bmatrix} J_2 \Sigma_{u_k} J_2^T & 0_{3x2L} \\ 0_{2Lx3} & 0_{2Lx2L} \end{bmatrix}$$

u_k does not change the uncertainty of the landmarks

EKF Update (correction): $z_k = h(s_k) + w \qquad w \sim N(0, Q_k)$

Covariance matrix of the sensor noise

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{Q}_{k,1} & 0 & \cdots & 0 \\ 0 & \mathbf{Q}_{k,2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{Q}_{k,M} \end{pmatrix}$$

M observed landmarks in each iteration k

For range-bearing observations:

$$h(x, m) = \begin{bmatrix} h_r \\ h_\theta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i - x)^2 + (y_i - y)^2} \\ atan2\left(\frac{y_i - y}{x_i - x}\right) - \theta \end{bmatrix}$$

Jacobian for each observation $k\ (r, \theta)$:

$$\begin{array}{cccc} & \text{Pose 2x3} & \text{Landmark 1} \quad \text{2x2L} & \text{Landmark L} \\ H_k = \frac{\partial h(s_k)}{\partial s_k} = & \left[\begin{matrix} \frac{\partial h_r}{\partial x_k} & \frac{\partial h_r}{\partial y_k} & \frac{\partial h_r}{\partial \theta_k} \\ \frac{\partial h_\theta}{\partial x_k} & \frac{\partial h_\theta}{\partial y_k} & \frac{\partial h_\theta}{\partial \theta_k} \end{matrix}\right. & \begin{matrix} \frac{\partial h_r}{\partial m_{x1}} & \frac{\partial h_r}{\partial m_{y1}} \\ \frac{\partial h_\theta}{\partial m_{x1}} & \frac{\partial h_\theta}{\partial m_{y1}} \end{matrix} & \left.\begin{matrix} \cdots \frac{\partial h_r}{\partial m_{xL}} & \frac{\partial h_r}{\partial m_{yL}} \\ \cdots \frac{\partial h_\theta}{\partial m_{xL}} & \frac{\partial h_\theta}{\partial m_{yL}} \end{matrix}\right]_{2x(3+2L)} \end{array}$$

jHxv

jHxf if observed

if not observed

- This Jacobian matrix tells us how the landmark observation $[h_r \quad h_\theta]^T$ changes with changes of the pose.
- If an element is non-zero, it means there is a dependency → introduces **correlation between landmarks and pose:** **error in the pose leads to error in the landmark**

$$\Rightarrow \begin{bmatrix} -\frac{x_i - x}{r} & -\frac{y_i - y}{r} & 0 \\ \frac{y_i - y}{r^2} & -\frac{x_i - x}{r^2} & -1 \end{bmatrix} \begin{bmatrix} \frac{x_i - x}{r} & \frac{y_i - y}{r} \\ -\frac{y_i - y}{r^2} & \frac{x_i - x}{r^2} \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

For M observed landmarks: $\text{size}(H_k) = 2\text{Mx}(3\text{x}2\text{L})$

EKF-SLAM

Update (correction)

Though $\bar{\Sigma}_k$ can be sparse, its propagation through H_k will eventuallly derive in a dense 2Mx2M matrix.

Gain of the EKF: Assuming M observed landmarks in each iteration

EKF update equations

$$\underbrace{K_k}_{(3+2L)x2M} = \underbrace{\bar{\Sigma}_k H_k^T}_{(3+2L)x2M} \underbrace{(H_k \bar{\Sigma}_k H_k^T + Q_k)}_{2Mx2M}{}^{-1}$$

$$\mu_k = \bar{\mu}_k + K_k (z_k - h(\bar{\mu}_k))$$

Innovation

$$\Sigma_k = (I - K_k H_k)\bar{\Sigma}_k$$

$$\mathbf{Q}_k = \begin{pmatrix} \mathbf{Q}_{k,1} & 0 & \cdots & 0 \\ 0 & \mathbf{Q}_{k,2} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{Q}_{k,M} \end{pmatrix}$$

2x2

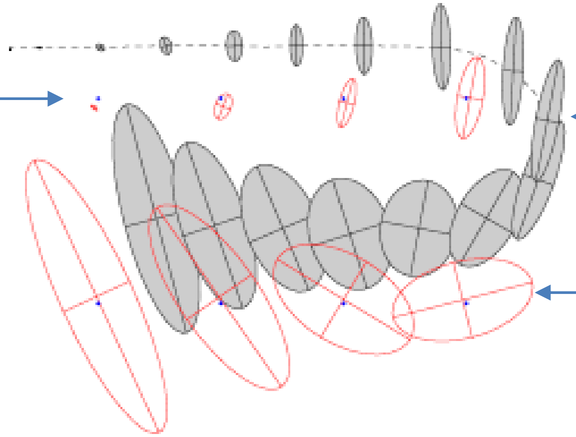
$$Q_{k,i} = \Sigma_{r_i\theta_i} = \begin{bmatrix} \sigma_{r_i}^2 & 0 \\ 0 & \sigma_{\theta_i}^2 \end{bmatrix}$$

```
%do Kalman update
   Innov = z-zPred;
   Innov(2) = AngleWrap(Innov(2));
   S = jH*PPred*jH'+REst;
   K = PPred*jH'*inv(S); %Gain K of the EKF
   xEst = xPred+ K*Innov;
   PEst = PPred-W*S*W';
```

EKF-SLAM

In **Online-Slam**, the uncertainty of both Poses and Landmarks grows as the robot moves

First landmark observed



Uncertainty in the pose

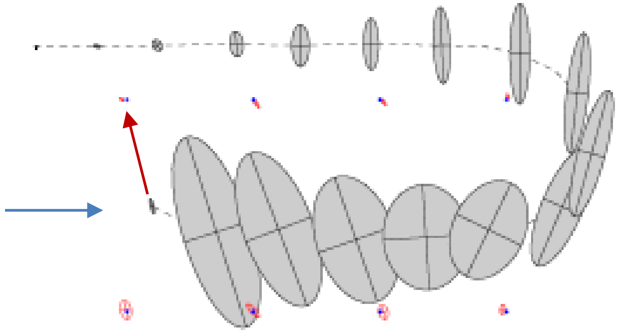
Uncertainty in the landmark

Get bigger and bigger

Until landmarks already in the map are re-observed

This is called: **Loop Closure**

From this pose, the first landmark is re-observed



The uncertaninty of the vector state $s_k = \{x_k, m_{1:L}\}$ decreases:

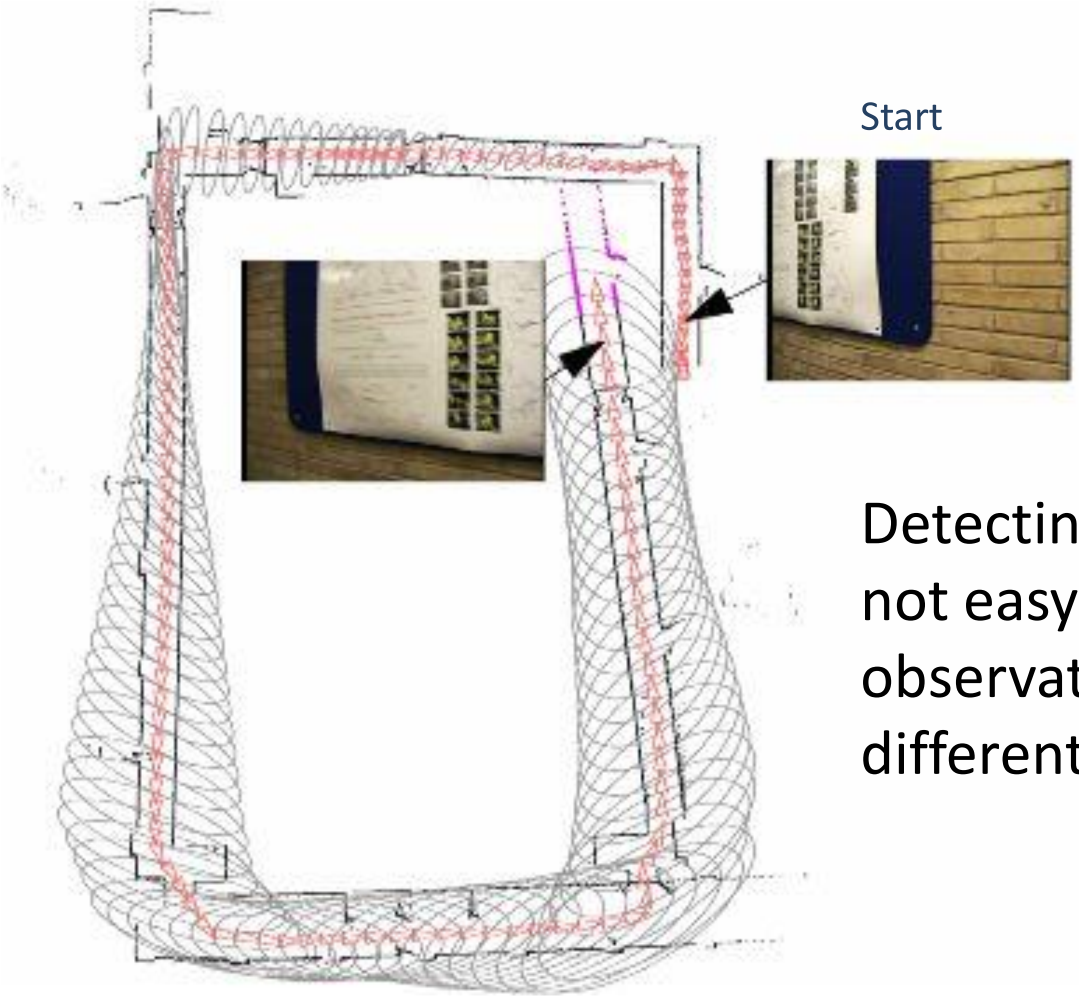
- The current pose x_k
- All the landmarks

Previous poses $x_{1:k-1}$ are not modified (they are not in the state vector!)

EKF-SLAM

Loop Closure (in Online EKF-SLAM)

Route along a rectangular corridor



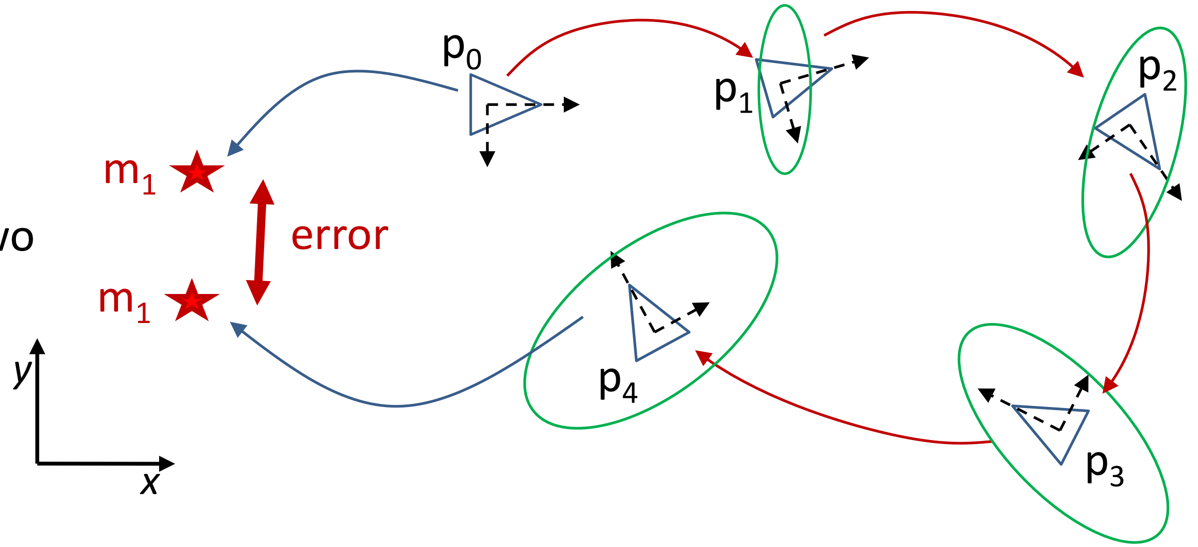
Detecting loop closure is not easy since the observations come from a different viewpoint.

GraphSLAM (for full SLAM)

- Landmarks and robot poses (all the trajectory) are represented as **nodes** in a graph
- **Arcs** are the sensed information relating them: odometry and/or the common observations
- **General idea:** Arcs are constraints for the free movement of nodes (landmarks and poses)

This creates a loop, which makes a huge difference to correct errors

Same landmark observed from two poses!



Why the global position of m_1 is different from p_0 and from p_4?

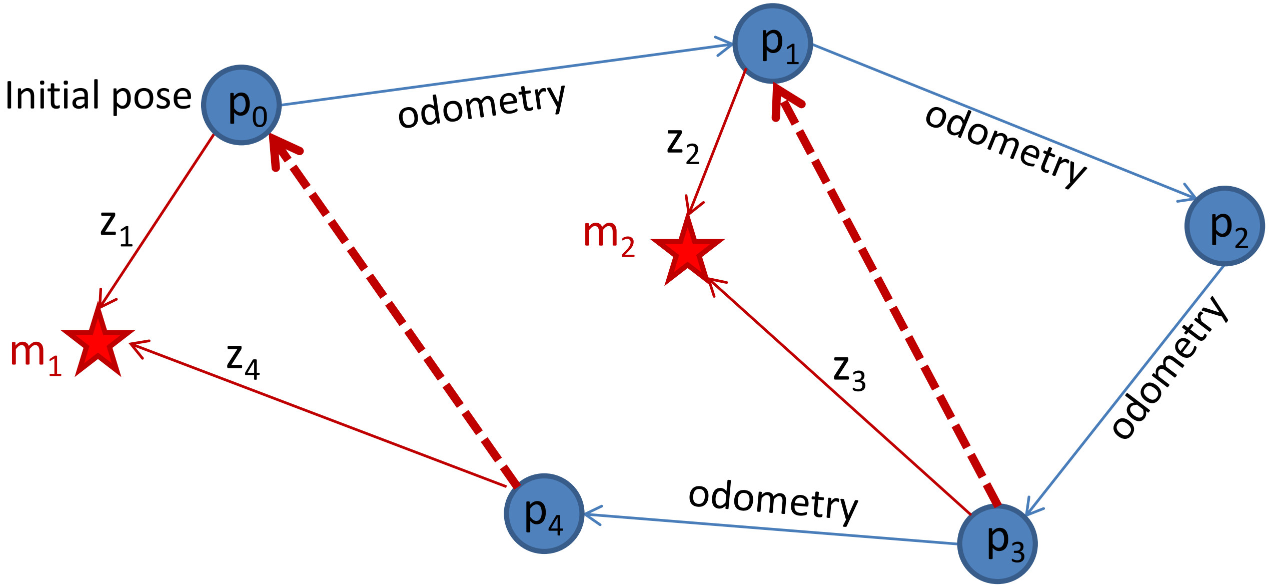
Because the estimated robot poses and landmark positions are not the right ones!

Solution: Move the nodes (Landmarks and Poses) to minimize the overall error in the observations (Square Error minimization)

Pose GraphSLAM

A simplification of GraphSlam: Only the poses are optimized

- **Nodes**: unknown robot poses (not the landmarks)
- **Arcs**: known constraints between robot poses given by :
  - Common observed landmarks (e.g. <z_1, z_4>, <z_2, z_3>) create a constraint (arc) between the poses
  - Odometry (visual odometry or wheel odometry). Constraint between consecutive nodes

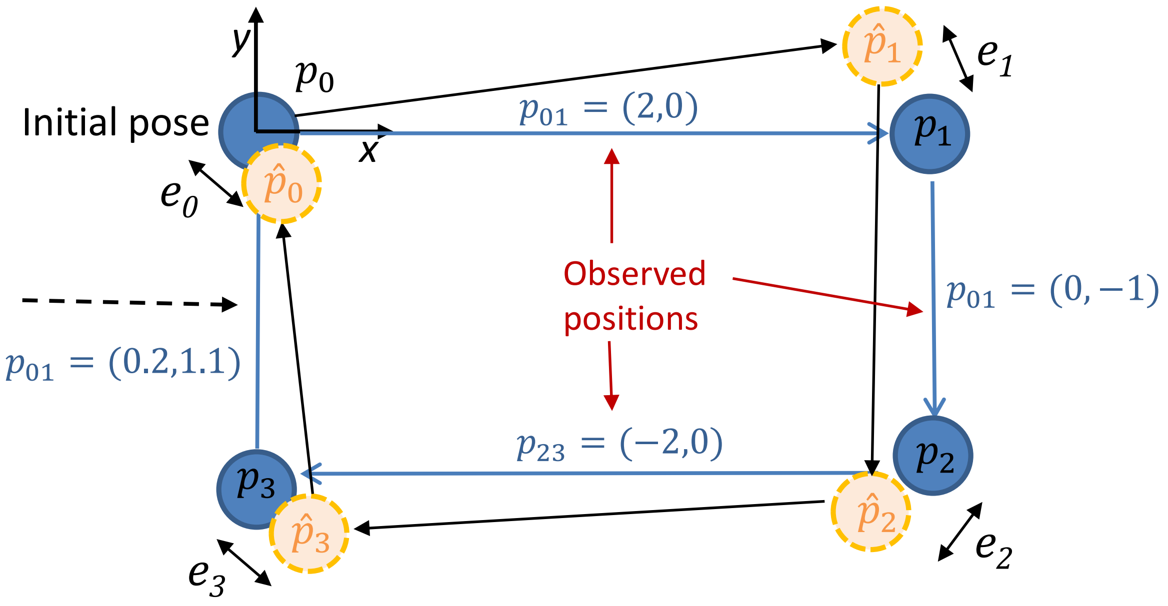


Loops in the Graph creates inconsistencies

<u>Example</u>: What is wrong with these 4 robot positions?

This observation should be (0,1) instead if $(0.2,1.1)$!

square error $(e^T e)$ = 0.2² + 0.1²



Solution: Move all the poses (except the initial one p_0 which is known) to minimize the overall square error between the new poses $\hat{p}_i$ and the observed ones

$$\{\hat{p}_1, \hat{p}_2, \hat{p}_3\} = \arg\min_{\{p_i\}} (e_0^2 + e_1^2 + e_2^2 + e_3^2) = \arg\min_{\{p_i\}} (e^T e)$$

Pose GraphSLAM

Constraint between poses created by a common observation: 2D landmark

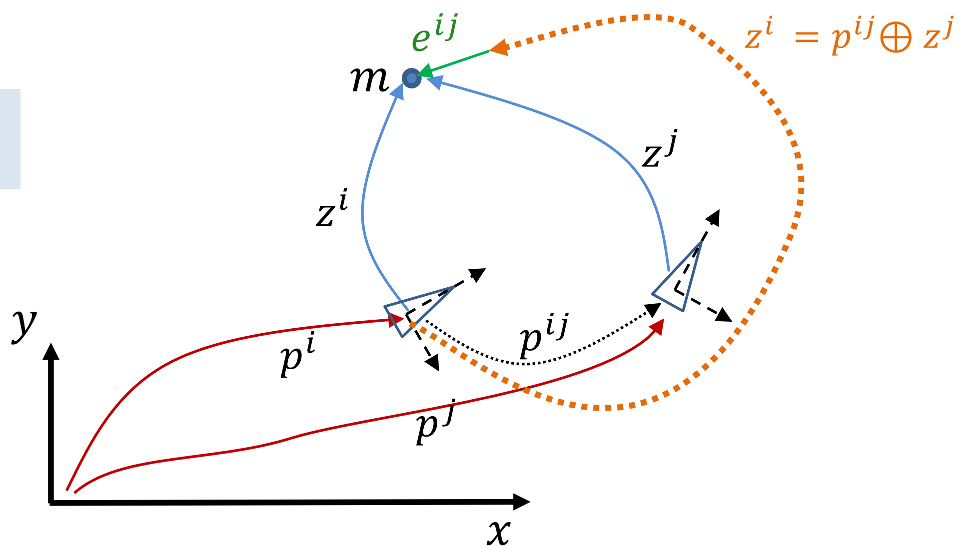
For convenience we change from subscript to superscripts

Let's consider a landmark m seen from two robot poses p^i and p^j

Constraint on p^{ij} given by z^i and z^j: $z^i = p^{ij} \oplus z^j$

$$p^j = p^i \oplus p^{ij} \Rightarrow p^{ij} = \ominus p^i \oplus p^j = p^j \ominus p^i$$

$$z^i = p^{ij} \oplus z^j = p^j \ominus p^i \oplus z^j$$



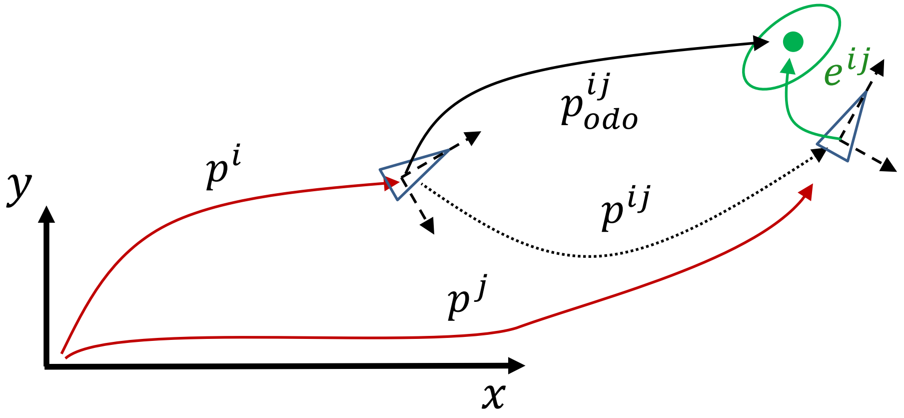
Not fulfilling this constraint gives us an error that depends on the poses p^i and p^j :

$$e^{ij}_{land}(p^i, p^j) = \left[e^{ij}_x, e^{ij}_y\right]^T = z^i \ominus p^{ij} \oplus z^j = z^i \ominus p^j \ominus p^i \oplus z^j$$

$e^{ij}_{land}(p^i, p^j) \backsim N(0, \Sigma_{e^{ij}_{land}})$ $\Sigma_{e^{ij}_{land}}$ computed by propagating the covariance Σ_{z_i} and Σ_{z_j}

Constraint between poses created by the odometry

$p^{ij}_{odo} \backsim N(\bar{p}_{odo}{}^{ij}, U_{ij})$ Uncertainty due to error in the odometry motion



Here, the error e^{ij} is a pose vector (not a point)

Constraint: $p^{ij}_{odo} = p^{ij} = \ominus p^i \oplus p^j = p^j \ominus p^i$

$$e^{ij}_{odo}(p^i, p^j) = [e^{ij}_x, e^{ij}_y, e^{ij}_\theta]^T = \ominus p^{ij} \oplus p^{ij}_{odo} = p^{ij}_{odo} \ominus p^{ij}$$

$$e^{ij}_{odo} \backsim N(0, \Sigma_{e^{ij}_{odo}})$$

$\Sigma_{e^{ij}_{odo}}$ computed by propagating the covariance U_{ij}

Optimization of the Graph

Taking into account all the arcs (errors):

$$\hat{P} = \arg \min_{P=\{p^i\}} [e^T \Sigma_e^{-1} e]$$

$$e = \begin{bmatrix} \vdots \\ e_{odo}^{ij} \\ \vdots \\ e_{land}^{ij} \\ \vdots \end{bmatrix}$$

Remember that each e_{odo}^{ij}, e_{land}^{ij} are function of $P = \{p^i\}$

$$\Sigma_e = \begin{bmatrix} \ddots & & & \\ & \Sigma_{e_{odo}^{ij}} & & \\ & & \Sigma_{e_{land}^{ij}} & \\ & & & \ddots \end{bmatrix}$$

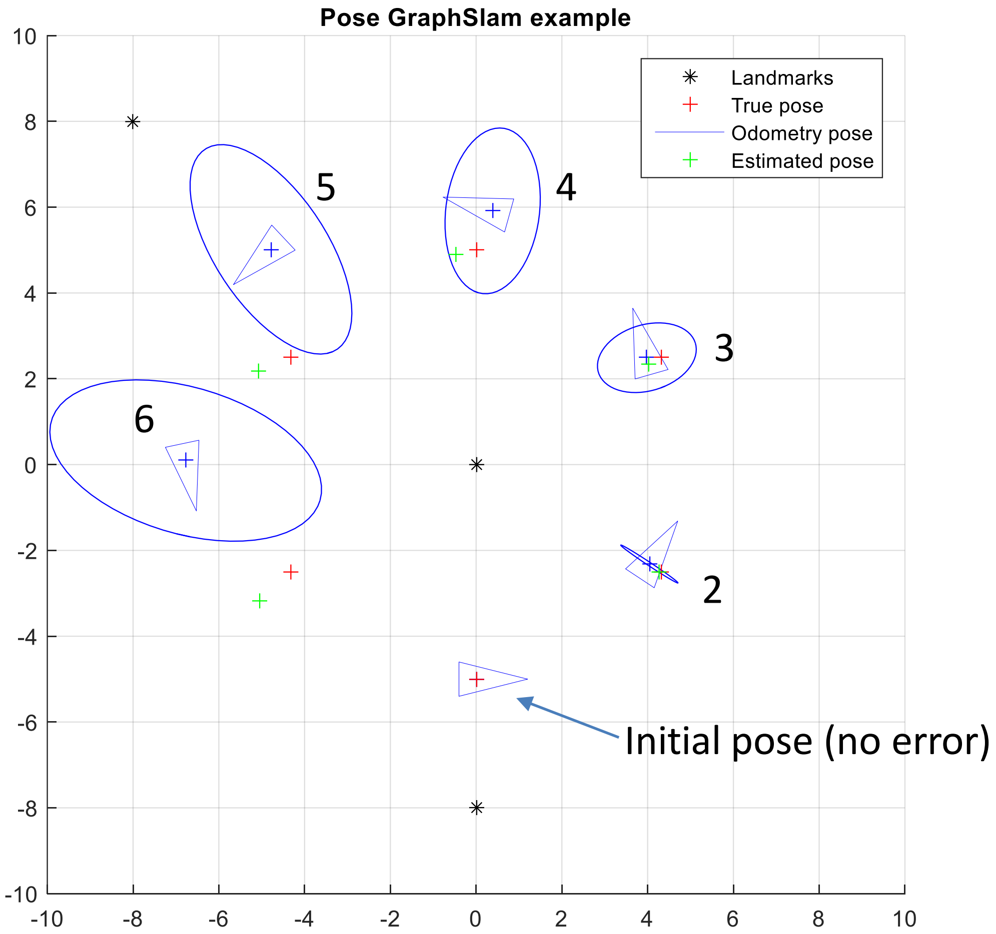
Solved iteratively with **Gauss-Newton:**

initial guess: $P_0 = P_{odo}$

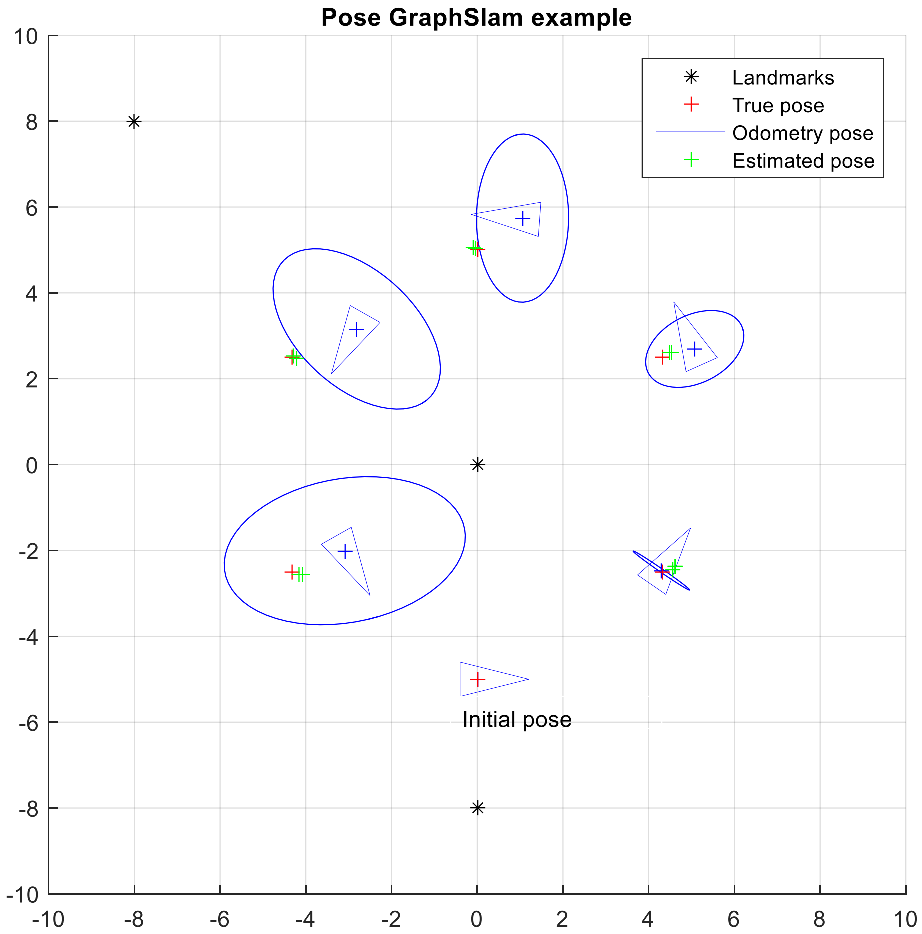
$\delta_P = (J_e^T \Sigma_e^{-1} J_e)^{-1} J_e^T \Sigma_e^{-1} e$

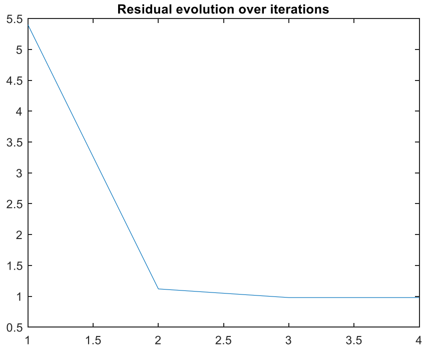
$P = P - \delta_P$

Example: Robot moving 5 times along a circular path observing 3 landmarks



After the first Gauss-Newton iteration





Convergence at iteration 4

Conclussions

- **SLAM**: Landmarks and poses unknown
- The Full SLAM: estimate the map and the full path of the robot
- The Online SLAM: estimate the map and the last robot pose
- **Two types**:
  - EKF-SLAM → Online SLAM
  - Pose GraphSLAM → Full SLAM