ANIRUDH SETH
aniset@kth.se

Tutorial 4
Factor Graph SLAM using Matlab

2020-02-05

1. The figures below represent the SLAM problem using different probabilistic graphical models. Here $x_{0:N}$ denotes the robot poses, $m_{1:M}$ the map landmarks , $z_{1:K}$ the set of measurements and $u_{1:N}$ represent the controls.

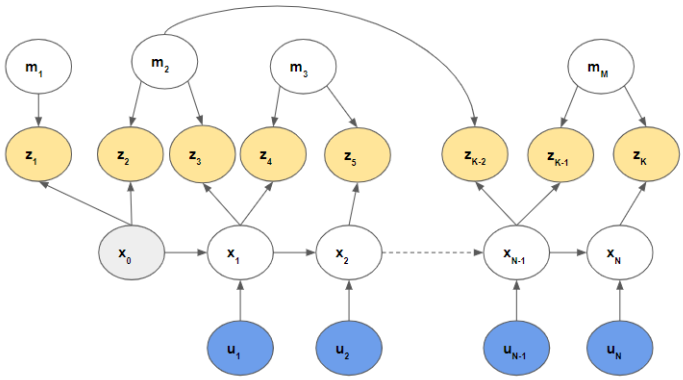


Figure 1: SLAM problem as a Bayesian belief network

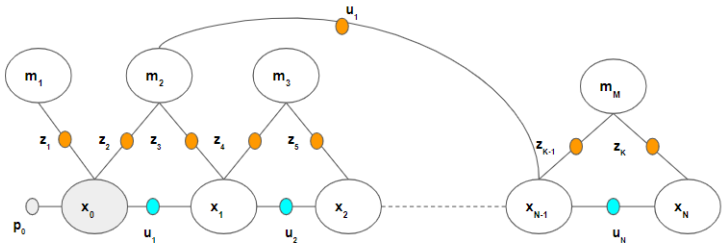


Figure 2: SLAM problem as a factor graph

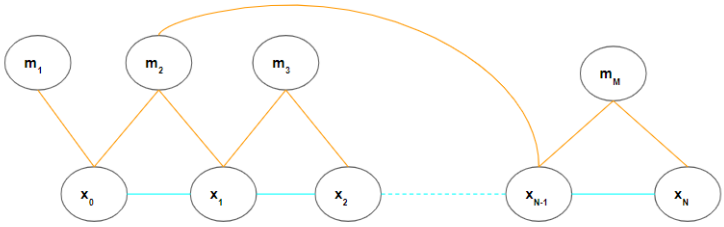


Figure 3: SLAM problem as a Markov random Field

2. In the equation 1 , the state of robot at the i^{th} time time step is denoted by x_i, with i $\in 0..N$, a landmark is represented by m_j , with j $\in 0..M$ and a measurement is represented with z_k with $k \in 1..K$. $P(x_0)$ represents the initial state. $p(x_i|x_{i-1}, u_i)$ represents the motion model, parameterised by a control input u_i and $p(z|x, m)$ is the landmark measurement model. The above measurement assumes a uniform prior over the landmarks m. Furthermore , it assumes that the data association problem has been solved, i.e that the indices i_k and j_k corresponding to each measurement z_k are known.

$$p(x_{0:N}, m_{1:M}, z_{1:K}) = p(x_0) \prod_{i=1}^{N} p(x_i|x_{i-1}, u_i) \prod_{k=1}^{K} p(z_k|x_{i_k}, m_{j_k}) \tag{1}$$

3. If each feature is seen only locally in time, the graph represented by the constraints is linear. Thus, Ω can be reordered so that it becomes a band-diagonal matrix, that is, all non-zero values occur near its diagonal. The equation $\mu = \Omega^{-1}\xi$ can then be computed in linear time. This intuition carries over to a cycle-free world that is traversed once, so that each feature is seen for a short, consecutive period of time. However, if some features are observed multiple times, with time delays in between(because the robot goes back and forth through a corridor, or because the world possesses cycles), there will exist features m_j that are seen at drastically different time steps x_{t1} and x_{t2} , with $t2 >> t1$. In our constraint graph, this introduces a cyclic dependence: x_{t1} and x_{t2} are linked through the sequence of controls u_{t1+1}, u_{t1+2}, . . . , u_{t2} and through the joint observation links between x_{t1} and m_j , and x_{t2} and m_j , respectively. Such links make our variable reordering trick inapplicable, and recovering the map becomes more complex.

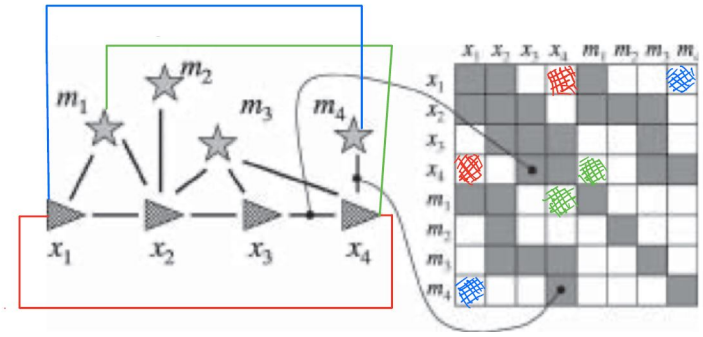


Figure 4: Structure of Ω in case of loop closure.

4. The log SLAM posterior can be written as follows:

$$\log p\left(y_{0t}|z_{1t},u_{1t},c_{1t}\right) \quad = \text{const.} + \log p\left(x_0\right) \quad + \sum_t \left[\log p\left(x_t|x_{t-1},u_t\right) + \sum_i \log p\left(z_t^i|y_t,c_t^i\right)\right]$$

The prior $p(x_0)$ anchors the initial pose of the robot to the origin of the global coordinate system : $x_0 : (0,0,0)$. The negative log probability can be written as:

$$\log p\left(y_{0:t}|z_{1:t},u_{1:t},c_{1:t}\right) = \text{const.} - \tfrac{1}{2} \quad \Big\{x_0^T\Omega_0 x_0 + \sum_t \left[x_t - g\left(u_t,\mu_{t-1}\right) - G_t\left(x_{t-1}-\mu_{t-1}\right)\right]^T$$

$$R_t^{-1}\left[x_t - g\left(u_t,\mu_{t-1}\right) - G_t\left(x_{t-1}-\mu_{t-1}\right)\right]$$
$$+ \sum_i \left[z_t^i - h\left(\mu_t,c_t^i\right) - H_t^i\left(y_t-\mu_t\right)\right]^T Q_t^{-1}\left[z_t^i - h\left(\mu_t,c_t^i\right) - H_t^i\left(y_t-\mu_t\right)\right]\Big\}$$

We can collect all quadratic terms into the matrix Ω, and all linear terms into a vector ξ (check equation 2):

$$\log p\left(y_{0:t}|z_{1:t},u_{1:t},c_{1:t}\right) = \text{const} - \tfrac{1}{2} \underbrace{x_0^T\Omega_0 x_0}_{\text{quadratic in } x_0} \underbrace{-\frac{1}{2}\sum_t x_{t-1:t}^T \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1}\left(1-G_t\right) x_{t-1:t}}_{\text{quadratic in } x_{t-1:t}}$$

$$\underbrace{+x_{t-1:t}^T \begin{pmatrix} 1 \\ -G_t \end{pmatrix} R_t^{-1}\left[g\left(u_t,\mu_{t-1}\right) + G_t\mu_{t-1}\right]}_{\text{linear in } x_{t-1:t}} - \tfrac{1}{2}\sum_i \underbrace{y_t^T H_t^{iT} Q_t^{-1} H_t^i y_t}_{\text{quadratic in } y_i} + \underbrace{y_t^T H_t^{iT} Q_t^{-1}\left[z_t^i - h\left(\mu_t,c_t^i,i\right) - H_t^i\mu_t\right]}_{\text{linear in } y_t}$$

Here $x_{t-1:t}$ denotes the vector concatenating x_{t-1} and x_t; hence we can write $\left(x_t - \ G_t x_{t-1}\right)^T = x_{t-1:t}^T\left(1-G_t\right)$

$$\log p\left(y_{0:t}|z_{1:t},u_{1:t},c_{1:t}\right) = \ \text{const.} \ - \frac{1}{2} y_{0:t}^T \Omega y_{0:t} + y_{0:t}^T\xi \tag{2}$$

Note: The matrices are rearranged so as to be of matching dimensions and the final results are calculated. Linearizing the motion equation for one pose to pose factor gives rise to the following terms to be added to Ω and ξ at the appropriate places:

$\Delta\Omega_{i-1,i-1} = G^T R^{-1} G$
$\Delta\Omega_{i,i} = R^{-1}$
$\Delta\Omega_{i,i-1} = -R^{-1}G$
$\Delta\Omega_{i-1,i} = -G^T R^{-1}$
$\Delta\xi_{i-1} = G^T R^{-1}(G\bar{\mu}_{i-1} - g(\bar{\mu}_{i-1}, u_i)$

$\Delta\xi_i = R^{-1}(g(\bar{\mu}_{i-1}, u_i) - G\bar{\mu}_{i-1})$

Using the Taylor expansion the kinematic model function g and measurement model function h are linearized. G and H represent the corresponding Jacobians.

$$\begin{aligned} g\left(u_t, x_{t-1}\right) &\approx g\left(u_t, \mu_{t-1}\right) + \underbrace{g'\left(u_t, \mu_{t-1}\right)}_{=G_t}\left(x_{t-1} - \mu_{t-1}\right) \\ h\left(y_t, c_t^i, i\right) &\approx h\left(\mu_t, c_t^i, i\right) + \underbrace{h'\left(\mu_t\right)}_{=H_i^i}\left(y_t - \mu_t\right) \end{aligned} \tag{3}$$

μ are the pose estimates. The pose estimates are initialized by assigning the first pose by zero, and then recursively applying the velocity motion model. μ_{i-1} represents the estimates from the last time step.

5. Each iteration takes as an input an estimated mean vector $\mu_{0:t}$ from the previous iteration, and outputs a new, improved estimate. A convergence test i.e absolute difference between the current and previous estimate can be compared with a set threshold (in tutorial the default threshold 1e-03 was used) to stop the algorithm.

6. The number of iterations of the GraphSLAM optimization can vary with the error estimates of initial poses of the robot. A higher error (e.g. more than 20 degrees orientation error) may require more iterations A small number of iterations (e.g. 3) is usually sufficient.[**1**]

7. The convergence of the algorithm is affected by the initial estimates of the poses.Also as the total number of time steps increases, the accuracy of the odometry-based initial guess will degrade, leading to an increased number of data association errors. GraphSLAM, as formulated in the paper [**1**], will eventually diverge because of this initial pose estimation step. This can be avoided by breaking estimate into parts which was implemented for the dataset 3,4 and 5. Another limitation pertains to the matrix inversion which can be slow and numerically unstable for larger data. It was also observed in dataset 3, that the algorithm got stuck at a local minima.

8. - The first and the second dataset are relatively simple , the first one is a small map with four landmarks and no noise. The second dataset is a relatively large map but since it is denser ,has several landmarks with very little noise, it is reconstructed easily. The third dataset is a small map with 5 landmarks but has a loop closure with the same landmarks being observed at different time steps. The fourth dataset has large noise in the measurements due to outliers. The large dataset has no odometry details so lacks initial pose estimates.
   - For the first three datasets, we can observed the final reconstructed map is the closest to the true map signifying that the solution has converged. For the fourth dataset, it was observed that the computation was often getting stuck at local minima resulting in an poor reconstruction of the true map. Modifying the covariance matrix Q for the measurement model altered the solution, but it suffered from the same problem.
   - Dataset 2 shows the effect of a dense map with several landmarks. In absence or little noise , the results converge to the true solution.
   - Noise in the measurements affect convergence. Figure 5 and 6 and show the convergence achieved on a noise free / little noise map . Dataset 4 in figure 7(left) has high noise resulting in a poor reconstruction in absence of outlier detection.
   - Origin has been used as the starting point as it anchor the initial pose of the robot. This is done for convenience. However, it is possible to have a differnt starting point. The formulas and the notation will have to be slightly modified to accommodate the same.
   - As mentioned before ,If each feature is seen only locally in time, the graph represented by the constraints is linear. Thus,Ω can be reordered so that it becomes a band-diagonal matrix, that is, all non-zero values occur near its diagonal. The equation $\mu = \Omega^{-1}\xi$ can then be computed in linear time. In practice this is often the case , the maps involves loop closure , with same landmarks being observed at different time steps. In such cases we can exploit the optimization perspective of SLAM by applying gradient descent [**2**] and [**3**] or conjugate gradient[**4**] without explicitly computing the full inverse matrix.
   - The result can be seen figure 4 , which is also explained in task 3.

9. The best final maps for each data set can be seen below :

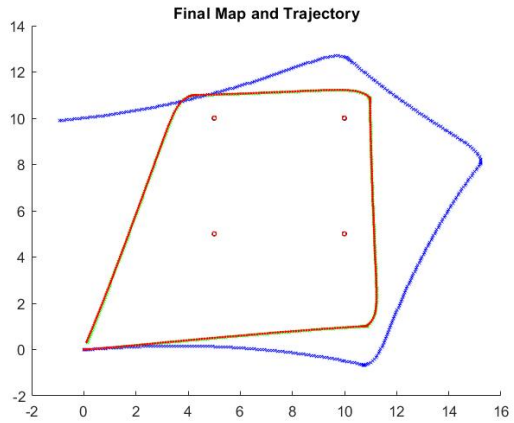


Figure 5: Dataset : 1

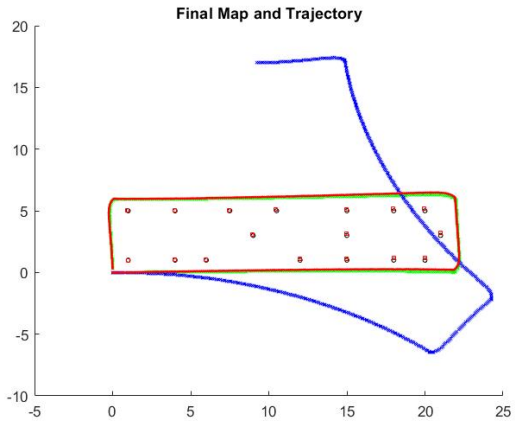
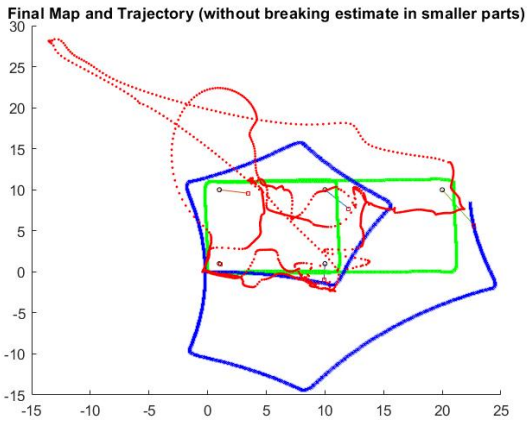


Figure 6: Dataset : 2



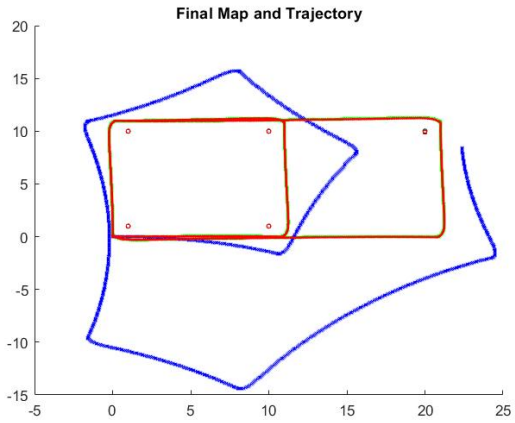
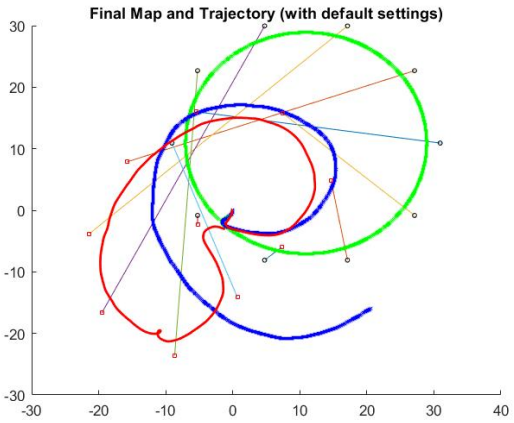


Figure 7: Dataset : 3



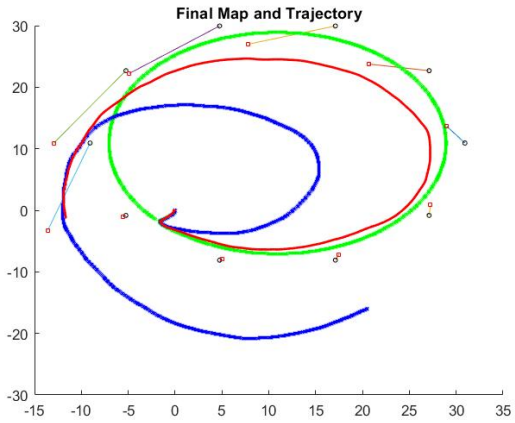


Figure 8: Dataset : 4

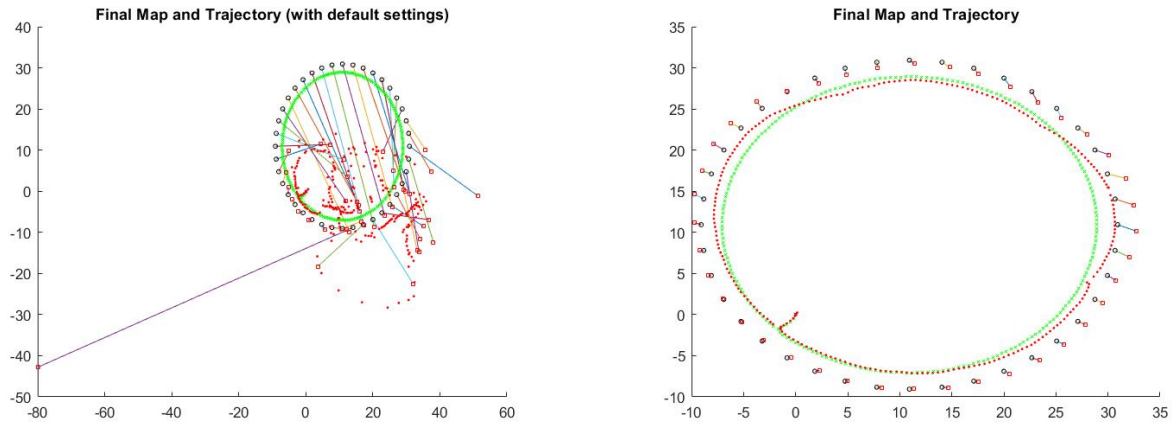


Figure 9: Dataset : 5

10. The initial algorithm calculates the entire estimate at one go. A simple improvement would be to breakdown the estimate into a number of parts, then using the estimate of the previous step as the starting point of the subsequent part. In case of outliers , a weighting strategy can be implemented which assign low weights to spurious measurements. In case of missing correspondence data , an algorithm in [**1**] can be implemented. For each pair of pair of features, we calculate probability of correspondence (data association). If this value is greater than a threshold , the correspondence vectors are set to the same value and the algorithm is repeated. Some alternative improvements could be to use the optimization perspective of SLAM by applying gradient descent [**2**] and [**3**] or conjugate gradient[**4**] without explicitly computing the full inverse matrix.

11. To get the algorithm the converge , some of the above mentioned improvements were implemented.

    - In Dataset 3, in figure 7 , the landmarks were observed after some time steps. The results from the initial approach can be seen in the figure on the left. To account for this estimate was broken into smaller estimates (with N=14 in this case) The previous estimate was then used as starting point of the subsequent steps. This improved the convergence significantly which can be seen in figure 7 on the right.
    - The Dataset 4, in figure 8 , the same approach was used along with strict convergence test over several iterations. There was improvement in the results , but the solution often converged to a local minima.
    - The Dataset 5 , in figure 9 was reconstructed by breaking the estimate in very small parts and assigning a very high covaraince to the measurement model. A significant improvement from the initial estimate was observed.

References

[1] Sebastian Thrun and Michael Montemerlo. 2006. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. Int. J. Rob. Res. 25, 5–6 (May 2006), 403–429. DOI:https://doi.org/10.1177/0278364906065387

[2] Folkesson, J. and Christensen, H. I. 2004a. Graphical SLAM: A self-correcting map. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), New Orleans, USA.

[3] Folkesson, J. and Christensen, H. I. 2004b. Robust SLAM. In Proceedings of the International Symposium on Autonomous Vehicles, Lisboa, PT.

[4] Konolige, K. 2004. Large-scale map-making. In Proceedings of the AAAI National Conference on Artificial Intelligence, pp. 457–463, San Jose, CA. AAAI