EL2320 - Graphical Methods

John Folkesson

GraphSLAM
(Chap 11 in Thrun)
Also Square Root SAM, Dellaert et al. (SAMDellaert.pdf)

KTH
Centre for Autonomous Systems

Student Board

- We need at least two volunteers to be the student board (Kursnmden).
- This is a vital part of the course development and evaluation.
- Involves meeting (1 hour) sometime early next year to disscuss: how the course went and what could be improved, and so on,
- Please email me if you are able to help with this.

Full Slam

The Graphical SLAM approach is one in which the full SLAM problem is addressed. That is all the robot path is estimated based on all the measurements upto a given time.

This allows all critical steps of SLAM to be repeteded iteratively, Data Association, Linearization and estimation, driving the system to a local minimum.

Thus it allows for the most accurate estimates of any methods we studied so far.

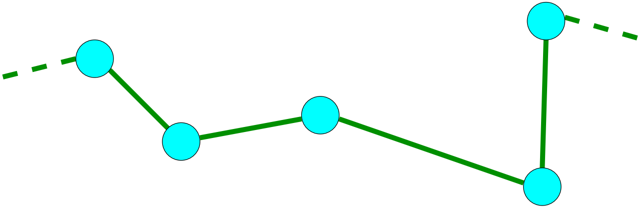
Full Slam

In its pure form the Graphical representation represents the true posteori over the map and robot path. It can thus provide a tool for making judgements (inference) about any point in the state space.

So for example questions like how much more likely is the robot to be near point $\mathbf{x}_1$ than near point $\mathbf{x}_2$. Or to compare two possible correspondences.

The Graph

The basic idea is to represent all robot pose states as nodes and measurements as edges between nodes.



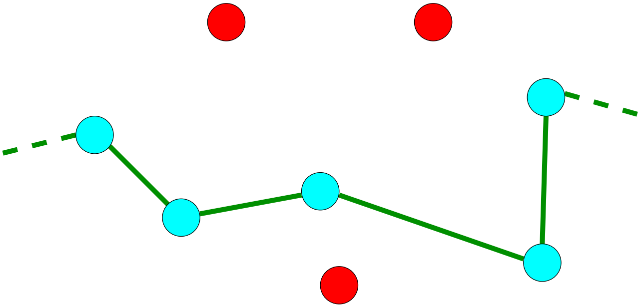
This can be thought of as a system of springs and masses. The 'energy' in the spring is then

$$\frac{1}{2}(\mathbf{x}_t - \mathbf{g}(u_t, \mathbf{x}_{t-1}))Q_t^{-1}(\mathbf{x}_t - \mathbf{g}(u_t, \mathbf{x}_{t-1}))$$

Compare to the exponent of the Gaussian. The minimum energy is then the maximum likelihood.

The Graph

Similarly we can represent the features as nodes (red).

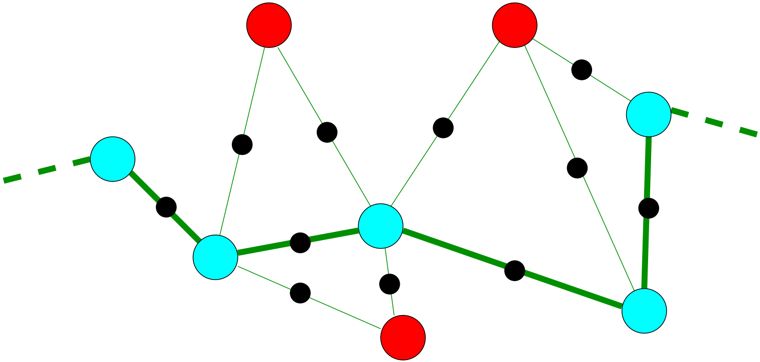


They also contribute to the energy with each measurement:

$$\frac{1}{2}(\mathbf{z}_t^i - \mathbf{h}^i(\mathbf{x}_t))R_t^{-1}(\mathbf{z}_t^i - \mathbf{h}^i(\mathbf{x}_t))$$

The Graph

We can emphisize the constraints by creating 'Energy Nodes'.



So computing the Posteori at any state (values pluged into the red and blue nodes) requires computing the energy at all energy nodes (black nodes) and adding up.

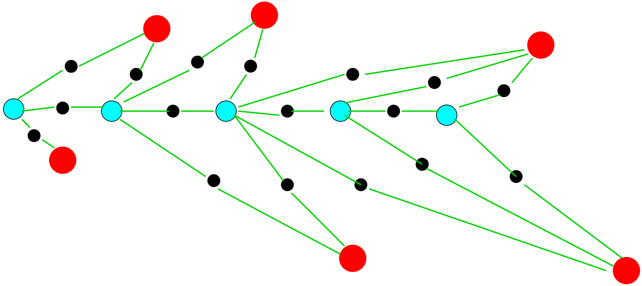
Each black node here depends on two state nodes which translates to a sparse matrix representation.

The Graph

= Features = Robot Poses

= Measurements or Energy Nodes

= Edge Showing dependancy



This is called a 'Factor Graph'.

The Measurement Matrix

We have energy contributions from energy node k that looks like:

$\frac{1}{2}\mathbf{f}_k(\mathbf{x}_i, \mathbf{x}_j) C_k^{-1} \mathbf{f}_k(\mathbf{x}_i, \mathbf{x}_j)$

Where $\mathbf{f}_k$ is this non-linear 'error'. If we take the 'squareroot' of the $C_k^{-1} = S_k^T S_k$ then we need to minimize a sum of terms like

$$
\begin{aligned}
E &= \sum_k \frac{1}{2} |S_k \mathbf{f}_k(\mathbf{x}_i, \mathbf{x}_j)|^2 \\
&\approx \sum_k \frac{1}{2} |A_k \begin{pmatrix} \Delta \mathbf{x}_i \\ \Delta \mathbf{x}_j \end{pmatrix} - \mathbf{b}_k|^2 \\
A_k &= S_k J_k \\
\mathbf{b}_k &= S_k \mathbf{f}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)
\end{aligned}
$$

Where J_k stands for Jacobian of $\mathbf{f}_k(\mathbf{x}_i, \mathbf{x}_j)$ evaluated at some linearization point $(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j)$ and $\Delta \mathbf{x} = \mathbf{x} - \bar{\mathbf{x}}$.

The Measurement Matrix

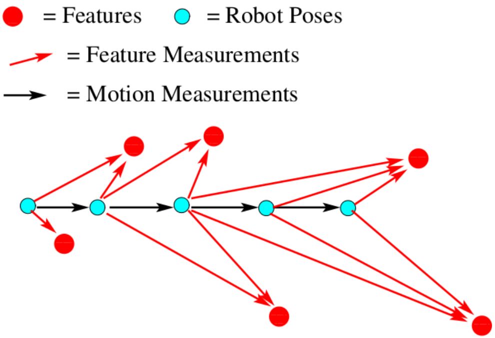
$$E \approx \sum_k \frac{1}{2} |A_k \begin{pmatrix} \Delta \mathbf{x}_i \\ \Delta \mathbf{x}_j \end{pmatrix} - \mathbf{b}_k|^2$$

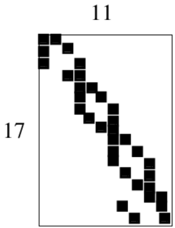
We can find the local min of this by setting its gradient to 0.

$\sum_k A_k^T (A_k \begin{pmatrix} \Delta \mathbf{x}_i \\ \Delta \mathbf{x}_j \end{pmatrix} - \mathbf{b}_k) = 0$

Stacking all the sparse A_k matricies creates the so called Measurement Matrix, A.

The Measurement Matrix

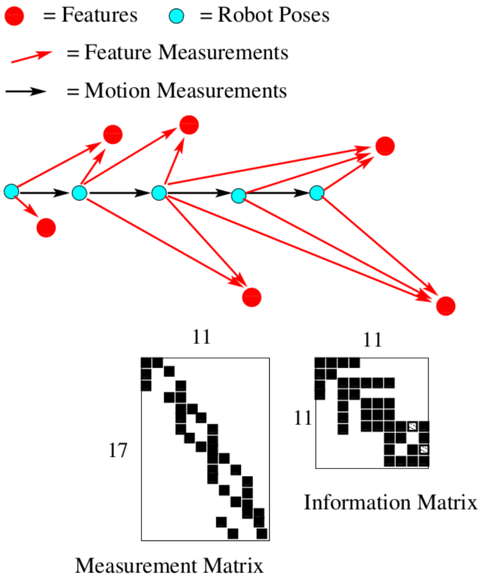




Measurement Matrix

$$A^T A \begin{pmatrix} \Delta \mathbf{x}_i \\ \Delta \mathbf{x}_j \end{pmatrix} = A^T \mathbf{b}_k$$

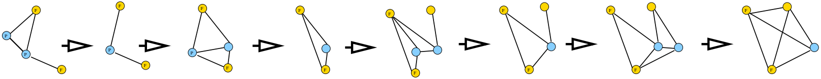
The Measurement Matrix



$$\Omega = A^T A$$

The Extended Information Filter

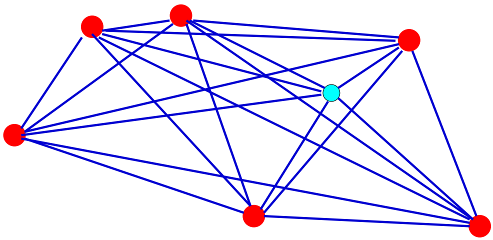
We can take the full SLAM Information Matrix and 'marginalize out' the past poses.

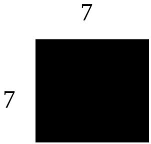


This is done by variable elimination. Essentially we solve for the pose as a linear function of the other varibles, plug in and collect terms to form a smaller system without the pose.

This however destroys our sparseness.

The Covariance Matrix, EKF

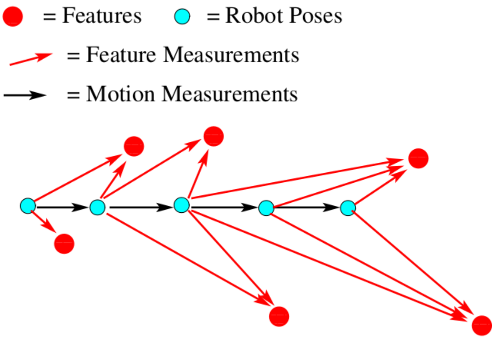


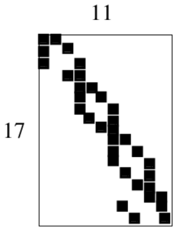


Marginalizing out poses

Covariance Matrix

The Measurement Matrix





Measurement Matrix

$$A^T A \begin{pmatrix} \Delta \mathbf{x}_i \\ \Delta \mathbf{x}_j \end{pmatrix} = A^T \mathbf{b}_k$$

If the robot only observes features for a while then moving on to new features extending the map and pose chain as it goes then the Measurement matrix remains sparse and the system can be 'solved' in linear time.

$$A^T A \Delta \mathbf{x} = A^T \mathbf{b}$$
$$R^T Q^T Q R \Delta \mathbf{x} = R^T Q^T \mathbf{b}$$
$$R^T R \Delta \mathbf{x} = R^T Q^T \mathbf{b}$$
$$R^T \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta \mathbf{x} = R^T \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix}$$

Where is did the so called QR decomposition of A into an orthogonal Matrix Q and an upper triangluar matrix R, (Not to be confused with our R_t and Q_t which were something else entirely).

$$R^T \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta \mathbf{x} = R^T \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix}$$
$$\tilde{R} \Delta \mathbf{x} = \mathbf{d}$$
$$residual = \mathbf{e}^T \mathbf{e}$$

As R is upper trianglular this can be solved in at worst $O(n^2)$ time but since it is also sparse this is typically $O(n)$.

The residual is important for comparing the energies at different linearization points.

The Residual

$$Q^T\mathbf{b} = \begin{bmatrix} \mathbf{d} \\ \mathbf{e} \end{bmatrix} ==> |\mathbf{b}|^2 = |\mathbf{d}|^2 + |\mathbf{e}|^2$$

$$R\Delta\mathbf{x} = \begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta\mathbf{x} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}$$

$$\begin{aligned}
residual &= (A\Delta\mathbf{x} - \mathbf{b})^T(A\Delta\mathbf{x} - \mathbf{b}) \\
&= (QR\Delta\mathbf{x} - \mathbf{b})^T(QR\Delta\mathbf{x} - \mathbf{b}) \\
&= (Q\begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} - \mathbf{b})^T(Q\begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} - \mathbf{b}) \\
&= |\mathbf{d}|^2 - \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}^T Q^T\mathbf{b} - \mathbf{b}^T Q \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix} + |\mathbf{b}|^2 \\
&= |\mathbf{d}|^2 - 2|\mathbf{d}|^2 + |\mathbf{b}|^2 = |\mathbf{e}|^2 \\
residual &= \mathbf{e}^T\mathbf{e}
\end{aligned}$$

The Residual

$$\begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta\mathbf{x} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}$$

$$residual = \mathbf{e}^T \mathbf{e}$$

The residual for instance can allow us to do ML data association. By computing the residual for different correspondances we can treat this residual exactly as the mahalanobis distance.

$$\begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta\mathbf{x} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}$$

$$\tilde{R}\Delta\mathbf{x} = \mathbf{d}$$

$$\Sigma = R^{-1}(R^{-1})^T$$

In practice it is easiler to compute only the submatricies of R^{-1} and Σ for the parts one is interested in.

$$\begin{bmatrix} \tilde{R} \\ 0 \end{bmatrix} \Delta \mathbf{x} = \begin{bmatrix} \mathbf{d} \\ \mathbf{0} \end{bmatrix}$$

$$\tilde{R}\Delta \mathbf{x} = \mathbf{d}$$

$$residual = \mathbf{e}^T \mathbf{e}$$

Adding a new measurement to a system already in this QR form can be done without having to start over from the begining.

We can use our previous decomposistion as a starting point and just 'fix' the new rows.

This leads to essentially constant time updates as long as we never return to a region we previously visited.

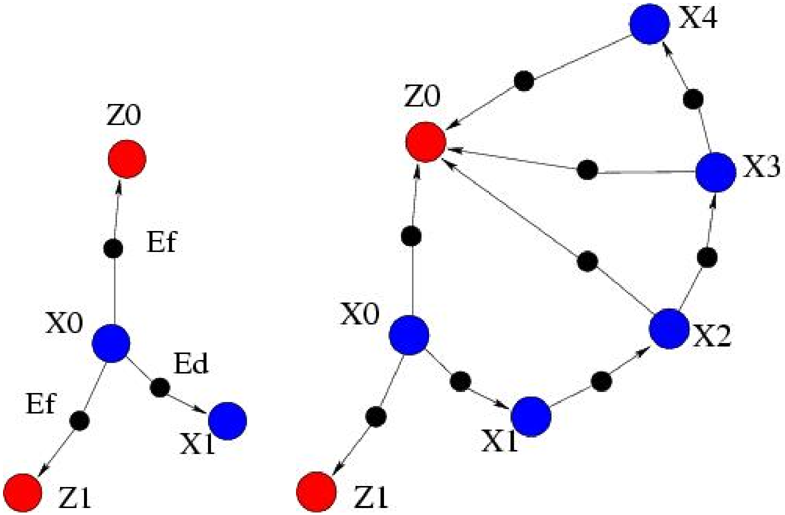
Anchoring Constraint

We typically add some sort of absolute initial position estimate

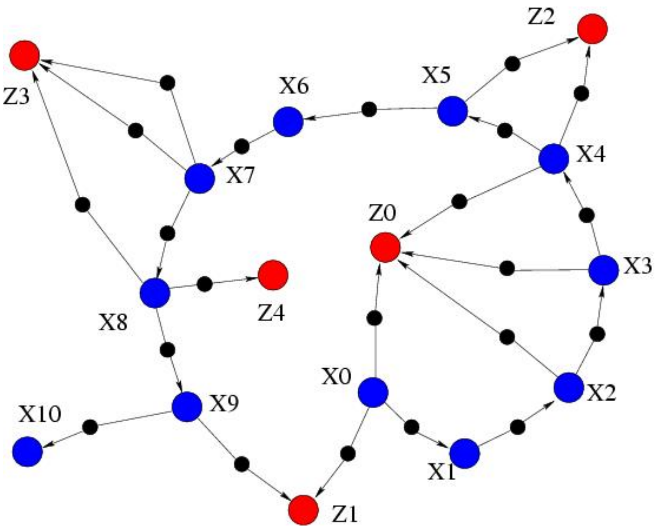
$\frac{1}{2}(\mathbf{x}_0 - \mu_0)^T \Omega_0 (\mathbf{x}_0 - \mu_0)$

to the graph in order to keep the system observable.
This might be real information at the start or all estimates might be understood as being relative to this assumed initial location.

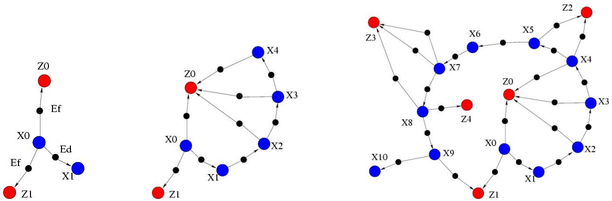
Loops



Loops



Loops

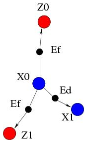


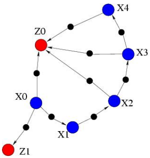
Loops are the end of the happy story of graphical SLAM.

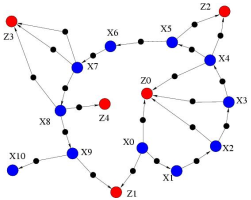
Loops will introduce constraints between the begining and end of the graph.

These make our sparse system dense and we can have very high complexity if we just drive ahead.

Loops







The most effiecient methods to solve graphs with loops use some sort of divide and conquer.

Above one can eliminate nodes one at a time creating fully connected subsystems.

By doing this in a good order really huge graphs can be solved (1,000,000 features).

There is no one Graph SLAM algorithm, but rather a collection of approaches to solving the full non-linear SLAM system using graphs.

These can give superior results compared to EKF and Particle filters both in terms of accuracy and computation time.

The drawback is messy linear algebra all over the place.

The algorithms are generally, relatively hard to implement (efficiently).