Advanced Techniques for Mobile Robotics

Graph-based SLAM using Least Squares

Wolfram Burgard, Cyrill Stachniss, Kai Arras, Maren Bennewitz
SLAM

- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain
Observing previously seen areas generates constraints between non-successive poses.

Constraints are inherently uncertain.
Idea of Graph-Based SLAM

- Use a graph to represent the problem
- Every **node** in the graph corresponds to a pose of the robot during mapping
- Every **edge** between two nodes corresponds to a spatial constraint between them

Graph-Based SLAM: **Build the graph** and find a node configuration that **minimize the error** introduced by the constraints
Graph-Based SLAM in a Nutshell

- Every node in the graph corresponds to a robot position and a laser measurement.
- An edge between two nodes represents a spatial constraint between the nodes.

KUKA Halle 22, courtesy of P. Pfaff
Graph-Based SLAM in a Nutshell

- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes
Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by “moving” the nodes
Once we have the graph, we determine the most likely map by “moving” the nodes... like this
Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by “moving” the nodes
- ... like this
- Then, we can render a map based on the known poses
The Overall SLAM System

- Interleaving process of front-end and back-end
- A consistent map helps to determine new constraints by reducing the search space
- This lecture focuses only on the optimization part
The Graph

- It consists of \( n \) nodes \( x = x_{1:n} \)
- Each node \( x_i \) is a 2D or 3D transformation (the pose of the robot at time \( t_i \))
- A constraint \( e_{ij} \) exists between the nodes \( x_i \) and \( x_j \) if
  - the robot observed the same part of the environment from \( x_i \) and \( x_j \) and constructs a “virtual measurement” about the position of \( x_j \) seen from \( x_i \)
  - an odometry measurement connects both poses.
The Graph

- It consists of $n$ nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each node $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time $t_i$)
- A constraint $e_{ij}$ exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if
  - the robot observed the same part of the environment from $\mathbf{x}_i$ and $\mathbf{x}_j$ and constructs a “virtual measurement” about the position of $\mathbf{x}_j$ seen from or
  - an odometry measurement connects both poses.
The Graph

- It consists of $n$ nodes $x=x_{1:n}$
- Each node $x_i$ is a 2D or 3D transformation (the pose of the robot at time $t_i$)
- A constraint $e_{ij}$ exists between the nodes $x_i$ and $x_j$ if
  - the robot observed the same part of the environment from $x_i$ and $x_j$ and constructs a “virtual measurement” about the position of $x_j$ seen from or
  - an odometry measurement connects both poses.
The Graph

- It consists of $n$ nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each node $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time $t_i$)
- A constraint $e_{ij}$ exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if
  - the robot observed the same part of the environment from $\mathbf{x}_i$ and $\mathbf{x}_j$ and constructs a “virtual measurement” about the position of $\mathbf{x}_j$ seen from or
  - an odometry measurement connects both poses.
The Edge Information Matrices

- Observations are affected by noise
- We use an information matrix $\Omega_{ij}$ for each edge to encode the uncertainty of the edge
- The “bigger” $\Omega_{ij}$, the more the edge “matters” in the optimization procedure

Questions:
- What do the information matrices look like in case of scan-matching vs. odometry?
- What should these matrices look like in a long, featureless corridor?
Pose Graph

observation of $x_j$ from $x_i$

nodes according to the graph

$\langle z_{ij}, \Omega_{ij} \rangle$

e$_{ij}(x_i, x_j)$

edge

error
Pose Graph

observation of $x_j$ from $x_i$

nodes according to the graph

$\langle z_{ij}, \Omega_{ij} \rangle$

edge

e_{ij}(x_i, x_j)$

error

Goal:

$$\hat{x} = \arg\min_x \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij}$$
SLAM as a Least Squares Problem

- The error function looks suitable for least squares error minimization

\[ \hat{x} = \arg\min_x \sum_{i,j} e_{ij}^T(x_i, x_j) \Omega_{ij} e_{ij}(x_i, x_j) \]

\[ = \arg\min_x \sum_k e_k^T(x) \Omega_k e_k(x) \]
The error function looks suitable for least squares error minimization

$$\hat{x} = \arg\min_x \sum_{ij} e_{ij}^T(x_i, x_j) \Omega_{ij} e_{ij}(x_i, x_j)$$

$$= \arg\min_x \sum_k e_k^T(x) \Omega_k e_k(x)$$

Questions:
- What is the state vector?
- Specify the error function!
SLAM as a Least Squares Problem

- The error function looks suitable for least squares error minimization

\[
\hat{x} = \arg\min_x \sum_{i,j} e_{ij}^T(x_i, x_j) \Omega_{ij} e_{ij}(x_i, x_j)
= \arg\min_x \sum_k e_k^T(x) \Omega_k e_k(x)
\]

Questions:
- What is the state vector?

\[
x^T = \begin{pmatrix} x_1^T & x_2^T & \cdots & x_n^T \end{pmatrix}
\]

- Specify the error function!
The Error Function

- The generic error function of a constraint characterized by a mean $z_{ij}$ and an information matrix $\Omega_{ij}$ is a vector of the same size as $x_i$.

$$e_{ij}(x_i, x_j) = t2v(Z^{-1}_{ij}(X_i^{-1}X_j))$$

- The error as a function of all the state $x$:

$$e_{ij}(x) = t2v(Z^{-1}_{ij}(X_i^{-1}X_j))$$

- The error function is 0 when

$$Z_{ij} = (X_i^{-1}X_j)$$
The Overall Error Minimization Procedure

- Define the error function
- Linearize the error function
- Compute its derivative
- Set the derivative to zero
- Solve the linear system
- Iterate this procedure until convergence
Linearizing the Error Function

- We can approximate the error functions around an initial guess $x$ via Taylor expansion

$$ e_{ij}(x + \Delta x) = e_{ij}(x) + J_{ij} \Delta x $$

$$ J_{ij} = \frac{\partial e_{ij}(x)}{\partial x} $$
Derivative of the Error Function

- Does one error function $e_{ij}(x)$ depend on all state variables?
Derivative of the Error Function

- Does one error function \( e_{ij}(x) \) depend on all state variables?
  - No, only on \( x_i \) and \( x_j \)
- Is there any consequence on the *structure* of the Jacobian?
Does one error function $e_{ij}(x)$ depend on all state variables?
- No, only on $x_i$ and $x_j$

Is there any consequence on the structure of the Jacobian?
- Yes, it will be non-zero only in the rows corresponding to $x_i$ and $x_j$!

$$\frac{\partial e_{ij}(x)}{\partial x} = \begin{pmatrix} 0 & \ldots & \frac{\partial e_{ij}(x_i)}{\partial x_i} & \ldots & \frac{\partial e_{ij}(x_j)}{\partial x_j} & \ldots & 0 \\ 0 & \ldots & A_{ij} & \ldots & B_{ij} & \ldots & 0 \end{pmatrix}$$
The error function $e_{ij}$ of one constraint depends only on the two parameter blocks $x_i$ and $x_j$

$$e_{ij}(x) = e_{ij}(x_i, x_j)$$

Thus, the Jacobian will be 0 everywhere but in the columns of $x_i$ and $x_j$.

$$J_{ij} = \begin{pmatrix} 0 \cdots 0 & \frac{\partial e(x_i)}{\partial x_i} & 0 \cdots 0 & \frac{\partial e(x_j)}{\partial x_j} \\ A_{ij} & B_{ij} & 0 \cdots 0 \end{pmatrix}$$
Consequences of the Sparsity

- To apply least squares, we need to compute the coefficient vectors and the coefficient matrices:

\[ b^T = \sum_{ij} b^T_{ij} = \sum_{ij} e^T_{ij} \Omega_{ij} J_{ij} \]

\[ H = \sum_{ij} H_{ij} = \sum_{ij} J^T_{ij} \Omega J_{ij} \]

- The sparse structure of \( J_{ij} \) will result in a sparse structure of \( H \)
- This structure reflects the adjacency matrix of the graph
Illustration of the Structure

\[ b_{ij} = J_{ij}^T \Omega_{ij} e_{ij} \]

Non-zero only at \( x_i \) and \( x_j \)
Illustration of the Structure

\[ b_{ij} = J_{ij}^T \Omega_{ij} e_{ij} \]

Non-zero only at \( x_i \) and \( x_j \)

\[ H_{ij} = J_{ij}^T \Omega_{ij} J_{ij} \]

Non-zero on the main diagonal at \( x_i \) and \( x_j \)
Illustration of the Structure

$$b_{ij} = J_{ij}^T \Omega_{ij} e_{ij}$$

Non-zero only at $x_i$ and $x_j$

$$H_{ij} = J_{ij}^T \Omega_{ij} J_{ij}$$

Non-zero on the main diagonal at $x_i$ and $x_j$

... and at the blocks $ij, ji$
Illustration of the Structure

\[ b = \sum_{ij} b_{ij} \]

\[ H = \sum_{ij} H_{ij} \]
Consequences of the Sparsity

- An edge of the graph contributes to the linear system via its coefficient vector $b_{ij}$ and its coefficient matrix $H_{ij}$.

- The coefficient vector is:

  $$b_{ij}^T = e_{ij}^T \Omega_{ij} J_{ij}$$
  $$= e_{ij}^T \Omega_{ij} \begin{pmatrix} 0 \cdots A_{ij} \cdots B_{ij} \cdots 0 \end{pmatrix}$$
  $$= \begin{pmatrix} 0 \cdots e_{ij}^T \Omega_{ij} A_{ij} \cdots e_{ij}^T \Omega_{ij} B_{ij} \cdots 0 \end{pmatrix}$$

- It is non-zero only at the indices corresponding to $x_i$ and $x_j$
Consequences of the Sparsity

- The coefficient matrix of an edge is:

\[
H_{ij} = J_{ij}^T \Omega_{ij} J_{ij}
\]

\[
= \left( \begin{array}{c}
A_{ij}^T \\
B_{ij}^T \\
\vdots
\end{array} \right) \Omega_{ij} \left( \begin{array}{c}
\cdots A_{ij} \\
\cdots B_{ij} \\
\vdots
\end{array} \right)
\]

- Is non zero only in the blocks \(i,j\).
Sparsity Summary

- An edge between $x_i$ and $x_j$ in the graph contributes only to the
  - $i^{th}$ and the $j^{th}$ blocks of the coefficient vector,
  - blocks $ii$, $jj$, $ij$ and $ji$ of the coefficient matrix.

- The resulting system is sparse and can be computed by iteratively “accumulating” the contribution of each edge

- Efficient solvers can be used
  - Sparse Cholesky decomposition (with COLAMD)
  - Conjugate Gradients
  - ... many others
The Linear System

- Vector of the states increments:
  \[ \Delta x^T = \begin{pmatrix} \Delta x_1^T & \Delta x_2^T & \cdots & \Delta x_n^T \end{pmatrix} \]

- Coefficient vector:
  \[ b^T = \begin{pmatrix} b_1^T & b_2^T & \cdots & b_n^T \end{pmatrix} \]

- System Matrix:
  \[ H = \begin{pmatrix}
    H^{11} & H^{12} & \cdots & H^{1n} \\
    H^{21} & H^{22} & \cdots & H^{2n} \\
    \vdots & \ddots & \ddots & \vdots \\
    H^{n1} & H^{n2} & \cdots & H^{nn}
  \end{pmatrix} \]

The linear system is a block system with \( n \) blocks, one for each node of the graph.
Building the Linear System

- $x$ is the current linearization point
- Initialization $b = 0$, $H = 0$
- For each constraint:
  - Compute the error $e_{ij} = t2v(Z_{ij}^{-1}(X_i^{-1} \cdot X_j))$
  - Compute the blocks of the Jacobian:
    $$A_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_i} \quad B_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_j}$$
- Update the coefficient vector:
  $$\bar{b}_i^T + = e_{ij}^T \Omega_{ij} A_{ij} \quad \bar{b}_j^T + = e_{ij}^T \Omega_{ij} B_{ij}$$
- Update the system matrix:
  $$\bar{H}_{ii}^{ij} + = A_{ij}^T \Omega_{ij} A_{ij} \quad \bar{H}_{ij}^{ij} + = A_{ij}^T \Omega_{ij} B_{ij}$$
Algorithm

- \( \mathbf{x} \): the initial guess
- While (!converged)
  - \( \langle \mathbf{H}, \mathbf{b} \rangle = \text{buildLinearSystem}(\mathbf{x}) \);
  - \( \Delta \mathbf{x} = \text{solveSparse}(\mathbf{H} \Delta \mathbf{x} = -\mathbf{b}) \);
  - \( \mathbf{x} += \Delta \mathbf{x} \);
How to Solve the Linear System?

- Linear system $H \Delta x = -b$
- Can be solved by matrix inversion (in theory)
- In practice:
  - Cholesky factorization
  - QR decomposition
  - Iterative methods such as conjugate gradients (for large systems)
- In Octave, use the backslash operator
  
  $\delta_x = -H \backslash b$
Example on the Blackboard…
Trivial 1D Example

- Two nodes and one observation

\[ x = (x_1, x_2)^T = (0, 0) \]
\[ z_{12} = 1 \]
\[ \Omega = 2 \]
\[ e_{12} = z_{12} - (x_2 - x_1) = 1 - (0 - 0) = 1 \]
\[ J_{12} = (1, -1) \]
\[ b_{12}^T = e_{12}^T \Omega_{12} J_{12} = (2, -2) \]
\[ H_{12} = J_{12}^T \Omega J_{12} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} \]
\[ \Delta x = -H_{12}^{-1} b_{12} \]

**BUT** \( \text{det}(H) = 0 \) ??
What Went Wrong?

- The constraint only specifies a **relative constraint** between both nodes
- Any poses for the nodes would be fine as long as their relative coordinates fit
- **One node needs to be fixed**

\[
\begin{align*}
H &= \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \\
\Delta x &= -H^{-1}b_{12} \\
\Delta x &= (0 1)^T
\end{align*}
\]
Exercise

- Consider a 2D graph where each pose $x_i$ is parameterized as $x_i^T = (x_i, y_i, \theta_i)$.

- Consider the error function $e_{ij} = t2v(Z_{ij}^{-1}(X_i^{-1} \cdot X_j))$.

- Compute the blocks of the Jacobian $J$:

$$A_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_i}, \qquad B_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_j}$$

- Hint: write the error function by using rotation matrices and translation vectors

$$e_{ij}(x_i, x_j) = Z_{ij}^{-1} \begin{pmatrix} R_i^T(t_j - t_i) \\ \theta_j - \theta_i \end{pmatrix}$$
Conclusions

- The back-end part of the SLAM problem can be effectively solved with least squares error minimization.
- The $H$ matrix is typically sparse.
- This sparsity allows for efficiently solving the linear system.
- One of the state-of-the-art solutions to compute the maximum likelihood estimate.