Robot Mapping

Least Squares Approach to SLAM

Gian Diego Tipaldi, Wolfram Burgard
Three Main SLAM Paradigms

Kalman filter  
Particle filter  
Graph-based

least squares approach to SLAM
Least Squares in General

- Approach for computing a solution for an *overdetermined system*
- “More equations than unknowns”
- Minimizes the *sum of the squared errors* in the equations
- Standard approach to a large set of problems

Today: Application to SLAM
**Graph-Based SLAM**

- Odometry measurements connect the poses of the robot while it is moving.
- Measurements are uncertain.
Graph-Based SLAM

- Observing previously seen areas generates measurements between non-successive poses
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 measurement between them
- **Graph-Based SLAM**: Build the graph and find a node configuration that minimize the measurement error
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 measurement 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 measurement between the nodes.
Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes
Graph-Based SLAM in a Nutshell

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

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

- Interplay of front-end and back-end
- Map helps data association by reducing the search space
- Topic today: optimization

Graph Construction *(Front-End)*

raw data

Graph Optimization *(Back-End)*

node positions

graph (nodes & edges)
The Graph

- It consists of n nodes $x = x_1:n$
- Each $x_i$ is a 2D or 3D transformation (the pose of the robot at time $t_i$)
- A measurement/edge exists between the nodes $x_i$ and $x_j$ if...
Create an Edge If... (1)

- ...the robot moves from $x_i$ to $x_{i+1}$
- Edge corresponds to odometry

\[ x_i \rightarrow x_{i+1} \]

The edge represents the odometry measurement
Create an Edge If... (2)

- ...the robot observes the same part of the environment from $x_i$ and from $x_j$
Create an Edge If... (2)

- ...the robot observes the same part of the environment from $x_i$ and from $x_j$
- Construct a **virtual measurement** about the position of $x_j$ seen from $x_i$

Edge represents the position of $x_j$ seen from $x_i$ based on the **observation**
Transformations

- Transformations can be expressed using **homogenous coordinates**
- Odometry-Based edge

\[(X_{i}^{-1}X_{i+1})\]

- Observation-Based edge

\[(X_{i}^{-1}X_{j})\]

How node i sees node j
Homogenous Coordinates

- H.C. are a system of coordinates used in projective geometry
- Projective geometry is an alternative algebraic representation of geometric objects and transformations
- Formulas involving H.C. are often simpler than in the Cartesian world
- A single matrix can represent affine transformations and projective transformations
Homogenous Coordinates

- H.C. are a system of coordinates used in projective geometry
- Projective geometry is an alternative algebraic representation of geometric objects and transformations
- Formulas involving H.C. are often simpler than in the Cartesian world
- A single matrix can represent affine transformations and projective transformations
Homogenous Coordinates

- N-dim space expressed in N+1 dim
- 4 dim. for modeling the 3D space
- To HC: \((x, y, z)^T \rightarrow (x, y, z, 1)^T\)
- Backwards: \((x, y, z, w)^T \rightarrow \left(\frac{x}{w}, \frac{y}{w}, \frac{z}{w}\right)^T\)
- Vector in HC: \(v = (x, y, z, w)^T\)
- Translation:
  \[
  T = \begin{pmatrix}
  1 & 0 & 0 & t_x \\
  0 & 1 & 0 & t_y \\
  0 & 0 & 1 & t_z \\
  0 & 0 & 0 & 1
  \end{pmatrix}
  \]
- Rotation:
  \[
  R = \begin{pmatrix}
  R^{3D} & 0 \\
  0 & 1
  \end{pmatrix}
  \]
The Edge Information Matrices

- Observations are affected by noise
- Information matrix $\Omega_{ij}$ for each edge to encode its uncertainty
- The “bigger” $\Omega_{ij}$, the more the edge “matters” in the optimization

Questions

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

observation of $x_j$ from $x_i$

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

edge

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

error

nodes according to the graph
Pose Graph

- **Goal:** \( x^* = \arg\min_x \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij} \)
Least Squares SLAM

- This error function looks suitable for least squares error minimization

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

\[ = \arg\min_x \sum_k e_k^T(x) \Omega_k e_k(x) \]
Least Squares SLAM

- This error function looks suitable for least squares error minimization

\[ x^* = \arg\min_x \sum_k e_k^T(x)\Omega_k e_k(x) \]

**Question:**
- What is the state vector?
Least Squares SLAM

- This error function looks suitable for least squares error minimization

\[ x^* = \arg\min_x \sum_k e_k^T(x) \Omega_k e_k(x) \]

Question:

- 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!

One block for each node of the graph
The Error Function

- Error function for a single measurement
  \[ e_{ij}(x_i, x_j) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \]

  - measurement
  - \( x_j \) referenced w.r.t. \( x_i \)

- Error as a function of the whole state vector
  \[ e_{ij}(x) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \]

- Error takes a value of zero if
  \[ Z_{ij} = (X_i^{-1}X_j) \]
Gauss-Newton: 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

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

with

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

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

- Does one error term $e_{ij}(x)$ depend on all state variables?

  No, only on $x_i$ and $x_j$
Derivative of the Error Function

- Does one error term \( e_{ij}(x) \) depend on all state variables?
  
  No, only on \( x_i \) and \( x_j \)

- Is there any consequence on the structure of the Jacobian?
Derivative of the Error Function

- Does one error term $e_{ij}(x)$ 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 $x_j$

$$\frac{\partial e_{ij}(x)}{\partial x} = \begin{pmatrix} 0 & \cdots & \frac{\partial e_{ij}(x_i)}{\partial x_i} & \cdots & \frac{\partial e_{ij}(x_j)}{\partial x_j} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{pmatrix}$$

$$J_{ij} = \begin{pmatrix} 0 & \cdots & A_{ij} & \cdots & B_{ij} & \cdots & 0 \end{pmatrix}$$
Jacobians and Sparsity

- Error $e_{ij}(x)$ depends only on the two parameter blocks $x_i$ and $x_j$

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

- The Jacobian will be zero everywhere except in the columns of $x_i$ and $x_j$
Consequences of the Sparsity

- We need to compute the coefficient vector $b$ and matrix $H$:

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

$$H = \sum_{ij} H_{ij} = \sum_{ij} J^T_{ij} \Omega_{ij} J_{ij}$$

- 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

\[ b_{ij} = J^T_{ij} \Omega_{ij} e_{ij} \]

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 contributes to the linear system via $b_{ij}$ and $H_{ij}$
- 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:

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

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

\[ = \begin{pmatrix} A_{ij}^T \Omega_{ij} A_{ij} & A_{ij}^T \Omega_{ij} B_{ij} \\ B_{ij}^T \Omega_{ij} A_{ij} & B_{ij}^T \Omega_{ij} B_{ij} \end{pmatrix} \]

- Non-zero only in the blocks relating i, j
Sparsity Summary

- An edge $ij$ contributes only to the
  - $i^{th}$ and the $j^{th}$ block of $b_{ij}$
  - to the blocks $ii$, $jj$, $ij$ and $ji$ of $H_{ij}$
- Resulting system is sparse
- System can be computed by summing up the contribution of each edge
- Efficient solvers can be used
  - Sparse Cholesky decomposition
  - 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:**
  \[ H = \begin{pmatrix} \bar{H}^{11} & \bar{H}^{12} & \cdots & \bar{H}^{1n} \\ \bar{H}^{21} & \bar{H}^{22} & \cdots & \bar{H}^{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{H}^{n1} & \bar{H}^{n2} & \cdots & \bar{H}^{nn} \end{pmatrix} \]
Building the Linear System

For each measurement:

- Compute error \( e_{ij} = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \)

- 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:
  \[
  \bar{H}_{ii}^+ = A_{ij}^T \Omega_{ij} A_{ij} \quad \bar{H}_{ij}^+ = A_{ij}^T \Omega_{ij} B_{ij} \\
  \bar{H}_{ji}^+ = B_{ij}^T \Omega_{ij} A_{ij} \quad \bar{H}_{jj}^+ = B_{ij}^T \Omega_{ij} B_{ij}
  \]
Algorithm

1: optimize(x):

2: while (!converged)

3:     \((H, b) = \text{buildLinearSystem}(x)\)

4:     \(\Delta x = \text{solveSparse}(H\Delta x = -b)\)

5:     \(x = x + \Delta x\)

6: end

7: return x
Example on the Blackboard
Trivial 1D Example

- Two nodes and one observation

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

BUT \( \det(H) = 0 \) ???
What Went Wrong?

- The observation specifies a \textit{relative measurement} between the nodes.
- Any poses for the nodes would be fine as long as their relative coordinates fit.
- One node needs to be “fixed”

\[ H = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \]

\[ \Delta x = -H^{-1}b_{12} \]

\[ \Delta x = (0 \ 1)^T \]
Fixing the Global Frame

- We saw that the matrix $\mathbf{H}$ has not full rank (after adding the measurements)
- The global frame had not been fixed
- Fixing the global reference frame is strongly related to the prior $p(x_0)$
- A Gaussian estimate about $x_0$ results in an additional measurement
- E.g., first pose in the origin:

$$e(x_0) = t2v(X_0)$$
Real World Examples
Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori
- We may want to optimize all others and keep these fixed
- How?
Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori
- We may want to optimize all others and keep these fixed
- How?
- If a variable is not optimized, it should “disappears” from the linear system
Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori.
- We may want to optimize all others and keep these fixed.
- How?
- If a variable is not optimized, it should “disappears” from the linear system.
- Construct the full system.
- Suppress the rows and the columns corresponding to the variables to fix.
Why Can We Simply Suppress the Rows and Columns of the Corresponding Variables?

\[ p(\alpha, \beta) = \mathcal{N}(\begin{bmatrix} \mu_\alpha \\ \mu_\beta \end{bmatrix}, \begin{bmatrix} \Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta} \\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta} \end{bmatrix}) = \mathcal{N}^{-1}(\begin{bmatrix} \eta_\alpha \\ \eta_\beta \end{bmatrix}, \begin{bmatrix} \Lambda_{\alpha\alpha} & \Lambda_{\alpha\beta} \\ \Lambda_{\beta\alpha} & \Lambda_{\beta\beta} \end{bmatrix}) \]

**Marginalization**

\[
p(\alpha) = \int p(\alpha, \beta) d\beta
\]

**Conditioning**

\[
p(\alpha | \beta) = \frac{p(\alpha, \beta)}{p(\beta)}
\]

<table>
<thead>
<tr>
<th><strong>Cov. Form</strong></th>
<th>( \mu = \mu_\alpha )</th>
<th>( \mu' = \mu_\alpha + \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1}(\beta - \mu_\beta) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Sigma = \Sigma_{\alpha\alpha} )</td>
<td>( \Sigma' = \Sigma_{\alpha\alpha} - \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1} \Sigma_{\beta\alpha} )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Info. Form</strong></th>
<th>( \eta = \eta_\alpha - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} \eta_\beta )</th>
<th>( \eta' = \eta_\alpha - \Lambda_{\alpha\beta} \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} )</td>
<td>( \Lambda' = \Lambda_{\alpha\alpha} )</td>
<td></td>
</tr>
</tbody>
</table>

Courtesy: R. Eustice
Uncertainty

- $\mathbf{H}$ represents the information matrix given the linearization point
- Inverting $\mathbf{H}$ gives the (dense) covariance matrix
- The diagonal blocks of the covariance matrix represent the (absolute) uncertainties of the corresponding variables
Relative Uncertainty

To determine the relative uncertainty between \( x_i \) and \( x_j \):

- Construct the full matrix \( \mathbf{H} \)
- Suppress the rows and the columns of \( x_i \) (= do not optimize/fix this variable)
- Compute the block \( j,j \) of the inverse
- This block will contain the covariance matrix of \( x_j \) w.r.t. \( x_i \), which has been fixed
Example

robot
Conclusions

- The back-end part of the SLAM problem can be effectively solved with Gauss-Newton
- The $H$ matrix is typically sparse
- This sparsity allows for efficiently solving the linear system
- One of the state-of-the-art solutions for computing maps
Literature

Least Squares SLAM

- Grisetti, Kümerle, Stachniss, Burgard: “A Tutorial on Graph-based SLAM”, 2010