Robot Mapping

Least Squares SLAM Revisited & Hierarchical Approach to Least Squares SLAM

Cyrill Stachniss
Graph-Based SLAM

- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain
Graph-Based SLAM

- Observing previously seen areas generates constraints 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 constraint between them
- **Graph-Based SLAM**: Build the graph and find a node configuration that minimize the error introduced by the constraints
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 constraint/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

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 $x_i$

![Diagram showing two points $x_i$ and $x_j$ with an edge between them. The 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
Pose Graph

- **Goal:** \( \mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij} \)
The Error Function

- Error function for a single constraint

\[ e_{ij}(x_i, x_j) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j)) \]

- 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

$$J_{ij} = \frac{\partial e_{ij}(x)}{\partial x}$$
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 $x_j$

$$J_{ij} = \begin{pmatrix} 0 & \frac{\partial e(x_i)}{\partial x_i} & 0 & \frac{\partial e(x_j)}{\partial x_j} & 0 & \ldots & 0 \\ \frac{\partial e(x_i)}{\partial x_i} & 0 & \frac{\partial e(x_j)}{\partial x_j} & 0 & \ldots & 0 \\ \frac{\partial e(x_i)}{\partial x_i} & 0 & \frac{\partial e(x_j)}{\partial x_j} & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e(x_i)}{\partial x_i} & 0 & \frac{\partial e(x_j)}{\partial x_j} & 0 & \ldots & 0 \\ 0 & \frac{\partial e(x_i)}{\partial x_i} & 0 & \frac{\partial e(x_j)}{\partial x_j} & 0 & \ldots & 0 \end{pmatrix}$$
Consequences of the Sparsity

- We need to compute the coefficient vector $\mathbf{b}$ and matrix $\mathbf{H}$:

$$\mathbf{b}^T = \sum_{ij} \mathbf{b}^T_{ij} = \sum_{ij} \mathbf{e}^T_{ij} \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$$

$$\mathbf{H} = \sum_{ij} \mathbf{H}_{ij} = \sum_{ij} \mathbf{J}^T_{ij} \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$$

- The sparse structure of $\mathbf{J}_{ij}$ will result in a sparse structure of $\mathbf{H}$

- 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

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

\[ H = \sum_{ij} H_{ij} \]
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:
  \[ H = \begin{pmatrix} H^{11} & H^{12} & \cdots & H^{1n} \\
H^{21} & H^{22} & \cdots & H^{2n} \\
\vdots & \vdots & \ddots & \vdots \\
H^{n1} & H^{n2} & \cdots & H^{nn} \end{pmatrix} \]
Building the Linear System

For each constraint:

- 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) = buildLinearSystem(x)
4:           Δx = solveSparse(HΔx = -b)
5:           x = x + Δx
6:       end
7:   return x
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 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”**

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

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

\[ \Delta x = (0 \ 1)^T \]
Role of the Prior

- We saw that the matrix $\mathbf{H}$ has not full rank (after adding the constraints).
- 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 constraint.
- 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><strong>INFO. FORM</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mu = \mu_{\alpha} )</td>
<td>( \eta = \eta_{\alpha} - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} \eta_{\beta} )</td>
</tr>
<tr>
<td>( \Sigma = \Sigma_{\alpha\alpha} )</td>
<td>( \Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} )</td>
</tr>
</tbody>
</table>

\[
\mu' = \mu_{\alpha} + \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1} (\beta - \mu_{\beta}) \]

\[
\Sigma' = \Sigma_{\alpha\alpha} - \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1} \Sigma_{\beta\alpha}
\]

\[
\eta' = \eta_{\alpha} - \Lambda_{\alpha\beta} \beta
\]

\[
\Lambda' = \Lambda_{\alpha\alpha}
\]
Uncertainty

- $\mathbf{H}$ is the information matrix (given the linearization point)
- Inverting $\mathbf{H}$ results in a (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 two nodes $x_i$ and $x_j$:

- Construct the matrix $H$
- Suppress the rows and the columns of $x_i$ (=“fixes” 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
Does all that run online?
Does all that run online?

... it depends on the size of the graph...
Hierarchical Pose-Graph

bottom layer (input data)  first layer  second layer  top layer

“There is no need to optimize the whole graph when a new observation is obtained”
Motivation

- Front-end seeks for loop-closures
- Requires to compare observations to all previously obtained ones
- In practice, limit search to areas in which the robot is likely to be
- This requires to know in which parts of the graph to search for data associations
Hierarchical Approach

- **Insight:** to find loop closures, one does not need the perfect global map
- **Idea:** correct only the core structure of the scene, not the overall graph
- The hierarchical pose-graph is a sparse approximation of the original problem
- It exploits the facts that in SLAM
  - Robot moved through the scene and it not “teleported” to locations
  - Sensors have a limited range
Key Idea of the Hierarchy

- Input is the dense graph
Key Idea of the Hierarchy

- Input is the dense graph
- Group the nodes of the graph based on their local connectivity
Key Idea of the Hierarchy

- Input is the dense graph
- Group the nodes of the graph based on their local connectivity
- For each group, select one node as a “representative”
Key Idea of the Hierarchy

- The representatives are the nodes in a new sparsified graph (upper level)
Key Idea of the Hierarchy

- The representatives are the nodes in a new sparsified graph (upper level)
- Edges of the sparse graph are determined by the connectivity of the groups of nodes
- The parameters of the sparse edges are estimated via local optimization
Key Idea of the Hierarchy

- The representatives are the nodes in a new sparsified graph (upper level)
- Edges of the sparse graph are determined by the connectivity of the groups of nodes
- The parameters of the sparse edges are estimated via local optimization

Process is repeated recursively
Key Idea of the Hierarchy

- Only the upper level of the hierarchy is optimized completely
Key Idea of the Hierarchy

- Only the upper level of the hierarchy is optimized completely.
- The changes are propagated to the bottom levels only close to the current robot position.
- Only this part of the graph is relevant for finding constraints.
Construction of the Hierarchy

- When and how to generate a new group?
  - A (simple) distance-based decision
  - The first node of a new group is the representative

- When to propagate information downwards?
  - Only when there are inconsistencies

- How to construct an edge in the sparsified graph?
  - Next slides

- How to propagate information downwards?
  - Next slides
Determining Edge Parameters

- Given two connected groups
- How to compute a virtual observation $Z$ and the information matrix $\Omega$ for the new edge?
Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest
Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest.
- The observation is the relative transformation between the two representatives.
Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest
- The observation is the relative transformation between the two representatives
- The information matrix is computed from the diagonal block of the matrix $H$

Inverse of the $[b,b]$ block of $H^{-1}$

$$\Omega_{ab} = (H_{[b,b]}^{-1})^{-1}$$
Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level
Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level
- Information is propagated downwards by transforming the group at the lower level using a rigid body transformation
Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level
- Information is propagated downwards by transforming the group at the lower level using a rigid body transformation
- Only if the lower level becomes inconsistent, optimize at the lower level
For the Best Possible Map...

- Run the optimization on the lowest level (at the end)
- For offline processing with all constraints, the hierarchy helps convergence faster in case of large errors
- In this case, one pass up the tree (to construct the edges) followed by one pass down the tree is sufficient
Stanford Garage

- Parking garage at Stanford University
- Nested loops, trajectory of ~7,000m
Stanford Garage Result

- Parking garage at Stanford University
- Nested loops, trajectory of ~7,000m
Stanford Garage Video
Intel Research Lab Video
Consistency

- How well does the top level in the hierarchy represent the original input?
Consistency

- How well does the top level in the hierarchy represent the original input?
- Probability mass of the marginal distribution in the highest level vs. the one of the true estimate (original problem, lowest level)

<table>
<thead>
<tr>
<th></th>
<th>Prob. mass not covered</th>
<th>Prob. mass outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel</td>
<td>0.10%</td>
<td>10.18%</td>
</tr>
<tr>
<td>W-10000</td>
<td>2.53%</td>
<td>24.05%</td>
</tr>
<tr>
<td>Stanford</td>
<td>0.01%</td>
<td>7.88%</td>
</tr>
<tr>
<td>Sphere</td>
<td>2.75%</td>
<td>10.21%</td>
</tr>
</tbody>
</table>

low risk of becoming overly confident

one does not ignore too much information
Consistency

- **Red**: overly confident (~0.1% prob. mass)
- **Blue**: under confident (~10% prob. mass)
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.
- Hierarchical pose-graph for computing approximate solutions online.
Literature

Least Squares SLAM

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

Hierarchical Approach

- Grisetti, Kümmerle, Stachniss, Frese, and Hertzberg: “Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping”
- Code: http://openslam.org/hog-man.html