Graph-Based SLAM and Sparsity

Cyrill Stachniss
Graph-Based SLAM ??
Graph-Based SLAM

SLAM = simultaneous localization and mapping
Graph-Based SLAM

SLAM = simultaneous localization and mapping

graph = representation of a set of objects where pairs of objects are connected by links encoding relations between the objects
What is my goal for today?
Graph-Based SLAM

- Nodes represent poses or locations
- 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
Graph-SLAM and Least Squares

- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive
- We have **real observations** relating the nodes with each other
Graph-SLAM and Least Squares

- The nodes represent the state
- Given a state, we can compute what we expect to perceive

Giorgio’s lecture

Find a configuration of the nodes so that the real and predicted observations are as similar as possible
Error Function

\[ h_1(x) = \hat{z}_1 \quad z_1 \]
\[ h_2(x) = \hat{z}_2 \quad z_2 \]
\[ h_n(x) = \hat{z}_n \quad z_n \]

minimize the differences!

\[ e_i(x) = e_i(x)^T \Omega_i e_i(x) \]
\[ e_i(x) = z_i - h_i(x) \]
Procedure in Brief

Iterate the following steps:

- Linearize around $x$ and compute for each measurement
  \[ e_i(x + \Delta x) \approx e_i(x) + J_i \Delta x \]

- Compute the terms for the linear system
  \[ b = \sum_i J_i^T \Omega_i e_i \quad H = \sum_i J_i^T \Omega_i J_i \]

- Solve the linear system
  \[ \Delta x^* = -H^{-1}b \]

- Updating state $x \leftarrow x + \Delta x^*$
Let’s use that for SLAM
Pose-Graph-Based SLAM

- Nodes represent poses or locations
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain
Pose-Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses
The Pose-Graph

- It consists of $n$ nodes $x = x_{1:n}$
- Each $x_i$ is a 2D or 3D pose (position and orientation 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 $x_i$

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

- **How to express** $x_j$ **relative to** $x_i$?
- Express this through transformations
- Let $X_i$ be transformation of the origin into $x_i$
- Let $X_i^{-1}$ be the inverse transformation
- We can express relative transformation $X_i^{-1}X_j$
Transformations

- **How to express** \( x_j \) **relative to** \( x_i \)?
- Express this through transformations
- Let \( X_i \) be transformation of the origin into \( x_i \)
- Let \( X_i^{-1} \) be the inverse transformation
- We can express relative transformation \( X_i^{-1}X_j \)
- Transformations can be expressed using **homogenous coordinates**
Transformations

- Transformations can be expressed using **homogenous coordinates**
- Odometry-Based edge
  \[
  (X_{i}^{-1}X_{i+1})
  \]
- Observation-Based edge
  \[
  (X_{i}^{-1}X_{j})
  \]
  describes “how node i sees node j”
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

Question

- What should these matrices look like when moving in a long, featureless corridor?
Pose-Graph

observation of $x_j$ from $x_i$ →

$e_{ij}(x_i, x_j)$ → error

nodes according to the graph

$z_{ij}, \Omega_{ij}$ ← edge
Pose-Graph

Goal: \( x^* = \arg\min_x \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij} \)
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 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
  \[ Z_{ij} = (X_i^{-1}X_j) \]
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 $\mathbf{x}$ via Taylor expansion

$$e_{ij}(\mathbf{x} + \Delta \mathbf{x}) \simeq e_{ij}(\mathbf{x}) + \mathbf{J}_{ij} \Delta \mathbf{x}$$

with

$$\mathbf{J}_{ij} = \frac{\partial e_{ij}(\mathbf{x})}{\partial \mathbf{x}}$$
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 $x_j$

\[
\frac{\partial e_{ij}(x)}{\partial x} = \begin{pmatrix} 0 & \ldots & \frac{\partial e_{ij}(x_i)}{\partial x_i} & \ldots & \frac{\partial e_{ij}(x_j)}{\partial x_j} & \ldots & 0 \end{pmatrix}
\]

\[
J_{ij} = \begin{pmatrix} 0 & \ldots & A_{ij} & \ldots & B_{ij} & \ldots & 0 \end{pmatrix}
\]
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 $x_j$

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

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

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

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

- 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

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

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

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

\[ H = \sum_{ij} H_{ij} \]
Sparsity Effect on $b$

- An edge contributes to the linear system via $b_{ij}$ and $H_{ij}$
- The coefficient vector is:

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

$$= e_{ij}^T \Omega_{ij} \left( \begin{array}{c} 0 \cdots A_{ij} \cdots B_{ij} \cdots 0 \end{array} \right)$$

$$= \left( \begin{array}{c} 0 \cdots e_{ij}^T \Omega_{ij} A_{ij} \cdots e_{ij}^T \Omega_{ij} B_{ij} \cdots 0 \end{array} \right)$$

- It is non-zero only at the indices corresponding to $x_i$ and $x_j$
Sparsity Effect on $H$

- The coefficient matrix of an edge is:

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

$$= \begin{pmatrix}
\vdots \\
A_{ij}^T \\
\vdots \\
B_{ij}^T \\
\vdots 
\end{pmatrix} \Omega_{ij} \begin{pmatrix}
\cdots A_{ij} \cdots B_{ij} \cdots 
\end{pmatrix}$$

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

- 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
All We Need...

- Vector of the states increments:
  \[ \Delta x^T = \left( \begin{array}{c} \Delta x_1^T \\ \Delta x_2^T \\ \vdots \\ \Delta x_n^T \end{array} \right) \]

- Coefficient vector:
  \[ b^T = \left( \begin{array}{c} b_1^T \\ b_2^T \\ \vdots \\ b_n^T \end{array} \right) \]

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

  small blocks (or vectors) corresponding to the individual constraints
... for 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:         \Delta x = solveSparse(H\Delta x = -b)

5:         x = x + \Delta x

6:     end

7:     return x
Real World Examples
The Graph with Landmarks
The Graph with Landmarks

- **Nodes** can represent:
  - Robot poses
  - Landmark locations

- **Edges** can represent:
  - Landmark observations
  - Odometry measurements

- The minimization optimizes the **landmark locations and robot poses**
Landmarks Observation

- Expected observation (x-y sensor)

\[
\hat{z}_{il}(x_i, x_l) = X_i^{-1} \begin{pmatrix} x_l \\ 1 \end{pmatrix}
\]

robot     landmark
Landmarks Observation

- Expected observation (x-y sensor)
  \[ \hat{z}_{il}(x_i, x_l) = X_i^{-1} \begin{pmatrix} x_l \\ 1 \end{pmatrix} \]
  robot  landmark

- Error function (in Euclidian space)
  \[ e_{il}(x_i, x_l) = \hat{z}_{il} - z_{il} \]
Bearing Only Observations

- A landmark is still a 2D point
- The robot observes only the bearing towards the landmark
- 1D Observation function

\[
\hat{z}_{il}(x_i, x_l) = \arctan\left(\frac{(x_l - t_i) \cdot y}{(x_l - t_i) \cdot x}\right) - \theta_i
\]
Bearing Only Observations

- **Observation function**

\[
\hat{z}_{il}(x_i, x_l) = \arctan\left(\frac{(x_l - t_i)_y}{(x_l - t_i)_x}\right) - \theta_i
\]

- **Error function**

\[
e_{il}(x_i, x_l) = \arctan\left(\frac{(x_l - t_i)_y}{(x_l - t_i)_x}\right) - \theta_i - z_{il}
\]
The Rank of the Matrix $H$

- What is the rank of $H_{ij}$ for a 2D landmark-pose constraint?
The Rank of the Matrix $H$

- What is the rank of $H_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $J_{ij}$ are 2x3 matrices
  - $H_{ij}$ cannot have more than rank 2

\[
\text{rank}(A^T A) = \text{rank}(A^T) = \text{rank}(A)
\]
The Rank of the Matrix $H$

- What is the rank of $H_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $J_{ij}$ are a 2x3 matrices
  - $H_{ij}$ cannot have more than rank 2
    $$\text{rank}(A^T A) = \text{rank}(A^T) = \text{rank}(A)$$

- What is the rank of $H_{ij}$ for a bearing-only constraint?
The Rank of the Matrix $H$

- What is the rank of $H_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $J_{ij}$ are $2 \times 3$ matrices
  - $H_{ij}$ cannot have more than rank 2
    $$\text{rank}(A^TA) = \text{rank}(A^T) = \text{rank}(A)$$

- What is the rank of $H_{ij}$ for a bearing-only constraint?
  - The blocks of $J_{ij}$ are $1 \times 3$ matrices
  - $H_{ij}$ has rank 1
Rank

- In landmark-based SLAM, the system can be under-determined
- The rank of $H$ is *less or equal* to the sum of the ranks of the constraints
- To determine a *unique solution*, the system must have *full rank*
Under-Determined Systems

- No guarantee for a full rank system
  - Landmarks may be observed only once
  - Robot might have no odometry
- We can still deal with these situations by adding a “damping” factor to $H$
- Instead of solving $H\Delta x = -b$, we solve
\[
(H + \lambda I)\Delta x = -b
\]

Levenberg Marquardt
UAV Example
UAV Example
Summary

- The back-end part of the SLAM problem can be solved with GN or LM
- The $H$ matrix is typically sparse
- This sparsity allows for efficiently solving the linear system
- There are several extensions (online, robust methods wrt outliers or initialization, hierarchical approaches, exploiting sparsity, multiple sensors)
YouTube Lectures

SLAM Course - WS13/14

Lecture Recordings from my winter 2013/14 course on SLAM taught in Freiburg.

Lecture material can be found here:
http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/

https://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_
Thank you for your attention!
Slide Information

- These slides have been created by Cyrill Stachniss, Giorgio Grisetti and Wolfram Burgard evolving from different courses and tutorials that we taught over the years between 2010 and 2016.
- I tried to acknowledge all people that contributed image or video material. In case I missed something, please let me know. If you adapt this course material, please make sure you keep the acknowledgements.
- Feel free to use and change the slides. If you use them, I would appreciate an acknowledgement as well. To satisfy my own curiosity, I appreciate a short email notice in case you use the material in your course.
- My video recordings of my lectures on robot mapping are available through YouTube:
  http://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_&feature=g-list

  Cyrill Stachniss, cyrill.stachniss@igg.uni-bonn.de