Robotics 2

Graph Based SLAM using Least Squares

Giorgio Grisetti
Graph-Based SLAM in a Nutshell

- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes

KUKA Halle 22, courtesy of the Pfaffie
Graph-Based SLAM in a Nutshell

- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes

KUKA Halle 22, courtesy of the Pfaffie
Graph-Based SLAM in a Nutshell

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

- 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 render a map based on the known poses and we are all happy

KUKA Halle 22 mapped, courtesy of me
Graph Optimization

- In this lecture we will *not* address the how to construct the graph, but only how to retrieve the position of its nodes which is maximally consistent the observations in the edges.
- A general Graph-Based slam algorithm interleaves the two steps
  - Graph Construction
  - Graph Optimization
- A consistent map helps in determining the new constraints by reducing the search space.
How does the Graph look like?

- It has \( n \) nodes \( x = x_{1:n} \)
  - Each node \( x_i \) is a 2D or 3D transformation representing the pose of the robot at time \( t_i \).
- There is a constraint \( e_{ij} \) between the node \( x_i \) and the node \( x_j \) if
  - either
    1. The robot observed the same part of the environment from both \( x_i \) and \( x_j \) and,
    2. Via this common observation it constructs a “virtual measurement” about the position of \( x_j \) seen from.
  - Or
    1. The positions are subsequent in time and there is an odometry measurement between the two.
How does the Graph look like?

- It has $n$ nodes $\mathbf{x} = \mathbf{x}_{1:n}$
  - Each node $\mathbf{x}_i$ is a 2D or 3D transformation representing the pose of the robot at time $t_i$.
- There is a constraint $e_{ij}$ between the node $\mathbf{x}_i$ and the node $\mathbf{x}_j$ if
  - either
    - The robot observed the same part of the environment from both $\mathbf{x}_i$ and $\mathbf{x}_j$ and,
    - Via this common observation it constructs a “virtual measurement” about the position of $\mathbf{x}_j$ seen from.
  - Or
    - The positions are subsequent in time and there is an odometry measurement between the two.
How does the Graph look like?

- It has $n$ nodes $\mathbf{x} = \mathbf{x}_{1:n}$
  - Each node $\mathbf{x}_i$ is a 2D or 3D transformation representing the pose of the robot at time $t_i$.
- There is a constraint $e_{ij}$ between the node $\mathbf{x}_i$ and the node $\mathbf{x}_j$ if
  - either
    - The robot observed the same part of the environment from both $\mathbf{x}_i$ and $\mathbf{x}_j$ and,
    - Via this common observation it constructs a “virtual measurement” about the position of $\mathbf{x}_j$ seen from.
  - Or
    - The positions are subsequent in time and there is an odometry measurement between the two.

In the edge:
the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$, based on the **observations**
How does the Graph look like?

- It has $n$ nodes $\mathbf{x} = \mathbf{x}_{1:n}$
  - Each node $\mathbf{x}_i$ is a 2D or 3D transformation representing the pose of the robot at time $t_i$.

- There is a constraint $e_{ij}$ between the node $\mathbf{x}_i$ and the node $\mathbf{x}_j$ if
  - either
    - The robot observed the same part of the environment from both $\mathbf{x}_i$ and $\mathbf{x}_j$ and,
    - Via this common observation it constructs a "virtual measurement" about the position of $\mathbf{x}_j$ seen from.
  - Or
    - The positions are subsequent in time and there is an odometry measurement between the two.
How does the Graph look like?

- To account for the different nature of the observations we add to the edge an information matrix $\Omega_{ij}$ to encode the uncertainty of the edge.
- The “bigger” (in matrix sense) $\Omega_{ij}$ is, the more the edge “matters” in the optimization procedure.

- Any hint about the information matrices of the system in case we use scan-matching and odometry?
- How should these matrices look like in an endless corridor in the two cases?
Pose Graph

The input for the optimization procedure is a graph annotated as follows:

- **Nodes**: \( x_i \) and \( x_j \)
- **Edge**: \( \langle z_{ij}, \Omega_{ij} \rangle \)
- **Observation of** \( x_j \) **from** \( x_i \)
- **Error**: \( e_{ij}(x_i, x_j) \)

**Goal:**

- Find the assignment of poses to the nodes of the graph which minimizes the negative log likelihood of the observations:

\[
\hat{x} = \arg\min_x \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij}^T
\]

```
pose_graph
```
SLAM as a Least Square Problem

- The function to minimize looks suitable for least squares (see previous lecture)

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

\[
= \arg\min \sum_{k} e_{k}^T(x) \Omega_{k} e_{k}(x)
\]

- We can regard each edge as a measurement, and use what we already know.

Questions:
- What is the state vector?

- What is the error function?
SLAM as a Least Square Problem

- The function to minimize looks suitable for least squares (see previous lecture)

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

\[ = \arg\min \sum_{k} e_{k}^T(x) \Omega_{k} e_{k}(x) \]

- We can regard each edge as a measurement, and use what we already now.

- Questions:
  - What is the state vector?
    \[ x^T = \begin{pmatrix} x_1^T & x_2^T & \cdots & x_n^T \end{pmatrix} \]
  - What is the error function?

One block for each node of the graph
The Error Function

- The generic error function of a constraint characterized by a mean $z_{ij}$ and an information $\Omega_{ij}$ is vector of the same size of a pose $x_i$.
  
  \[ e_{ij}(x_i, x_j) = t2v(Z_{ij}^{-1}(X_i^{-1} \cdot X_j)) \]

- We can write the error as a function of all the state $x$.
  
  \[ e_{ij}(x) = t2v(Z_{ij}^{-1}(X_i^{-1} \cdot X_j)) \]

- Note that the error function is 0 when $Z_{ij} = (X_i^{-1} \cdot X_j)$
The 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 Jacobian?
  - Yes, it will be non-zero only in the rows corresponding to $x_i$ and $x_j$!
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 Jacobian?
The 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?
  - Yes, it will be non-zero only in the rows corresponding to $x_i$ and $x_j$!

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

\[
A_{ij} = \begin{pmatrix}
0 & \ldots & B_{ij} & \ldots & C_{ij} & \ldots & 0
\end{pmatrix}
\]
Consequences of the Sparsity

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

\[ b^T = \sum_{ij} b_{ij}^T = \sum_{ij} e_{ij}^T \Omega_{ij} A_{ij} \]

\[ H = \sum_{ij} H_{ij} = \sum_{ij} A_{ij}^T \Omega A_{ij}^T \]

- The sparse structure of \( A_{ij} \) will result in a sparse structure of the linear system.
- This structure will reflect the topology of the graph.
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:

\[
b_{ij}^T = e_{ij}^T \Omega_{ij} A_{ij} = e_{ij}^T \Omega_{ij} \left( \begin{array}{ccc} 0 & \cdots & B_{ij} & \cdots & C_{ij} & \cdots & 0 \\ \end{array} \right) = \left( \begin{array}{cccc} 0 & \cdots & e_{ij}^T \Omega_{ij} B_{ij} & \cdots & e_{ij}^T \Omega_{ij} C_{ij} & \cdots & 0 \end{array} \right)
\]

- It is non-zero only in correspondence of $x_i$ and $x_j$
Consequences of the Sparsity (cont.)

The coefficient matrix of an edge is:

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

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

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

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

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

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

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

- $x$: the initial guess
- While (! converged)
  - $<H,b> = \text{buildLinearSystem}(x)$;
  - $\Delta x = \text{solveSparse}(H \Delta x = b)$;
  - $x += \Delta x$;
Exercise(s)

- 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
  \[ B_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_i} \quad C_{ij} = \frac{\partial e(x_i, x_j)}{\partial x_j} \]

- 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

- A part of the SLAM problem can be effectively solved with least square optimization.
- The algorithm described in this lecture has been entirely implemented in octave. Get the package from the web-page of the course.
- Play with the example, and figure out the relation between
  - the connectivity of the graph and
  - The structure of the matrix $H$. 