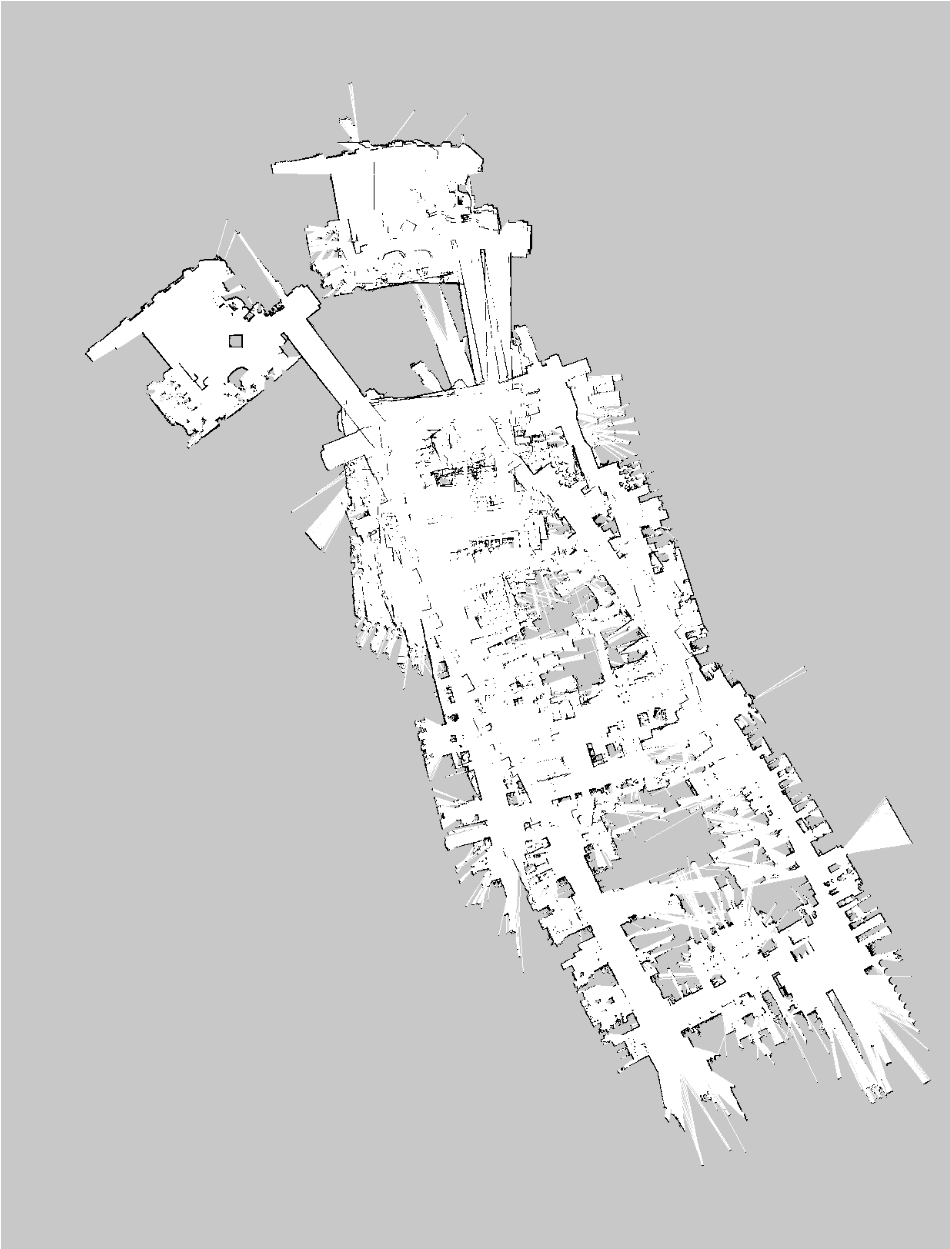
Robotics 2

Graph Based SLAM using Least Squares

Giorgio Grisetti

Graph-Based SLAM in a Nutshell

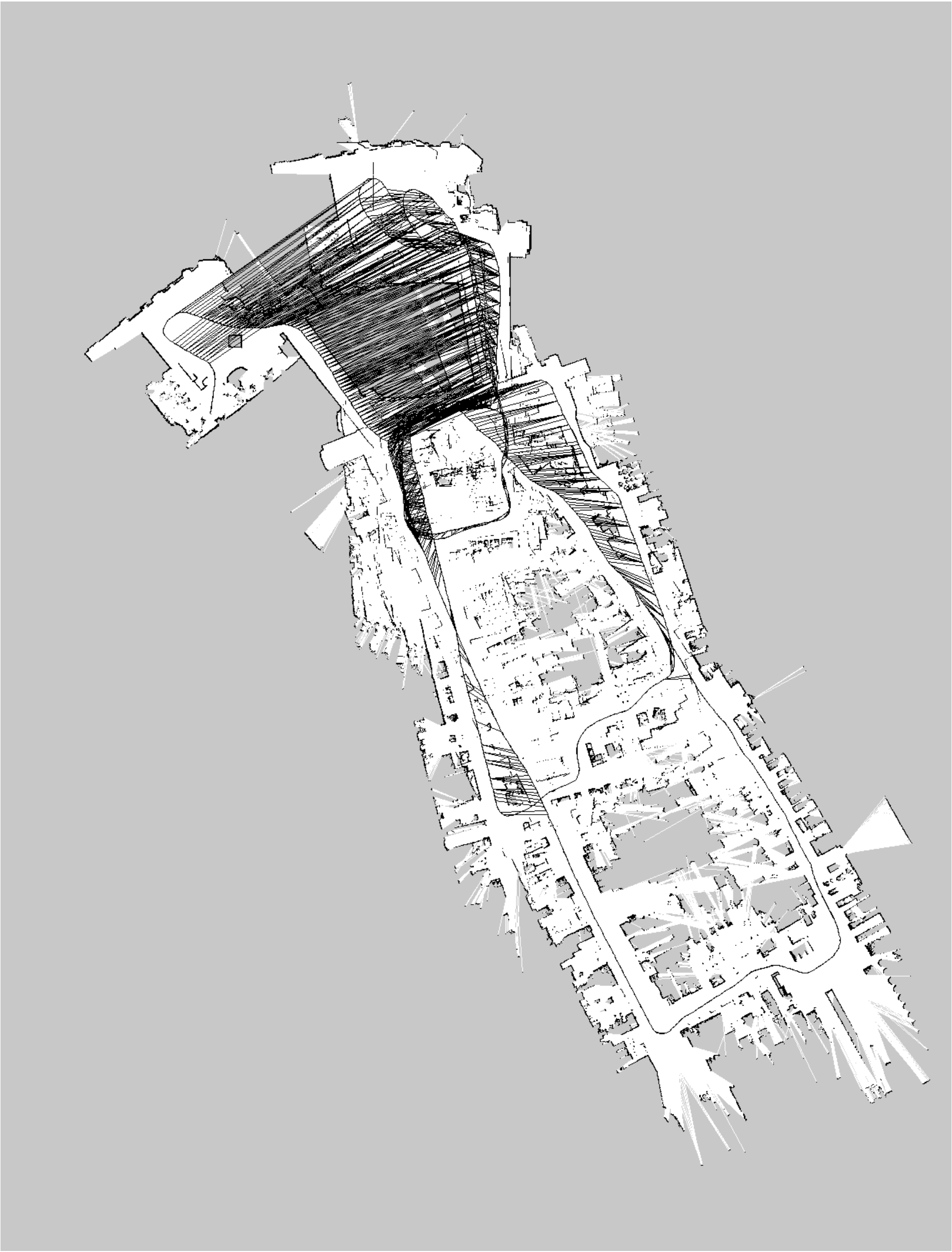
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



KUKA Halle 22, courtesy of the Pfaffie

Graph-Based SLAM in a Nutshell

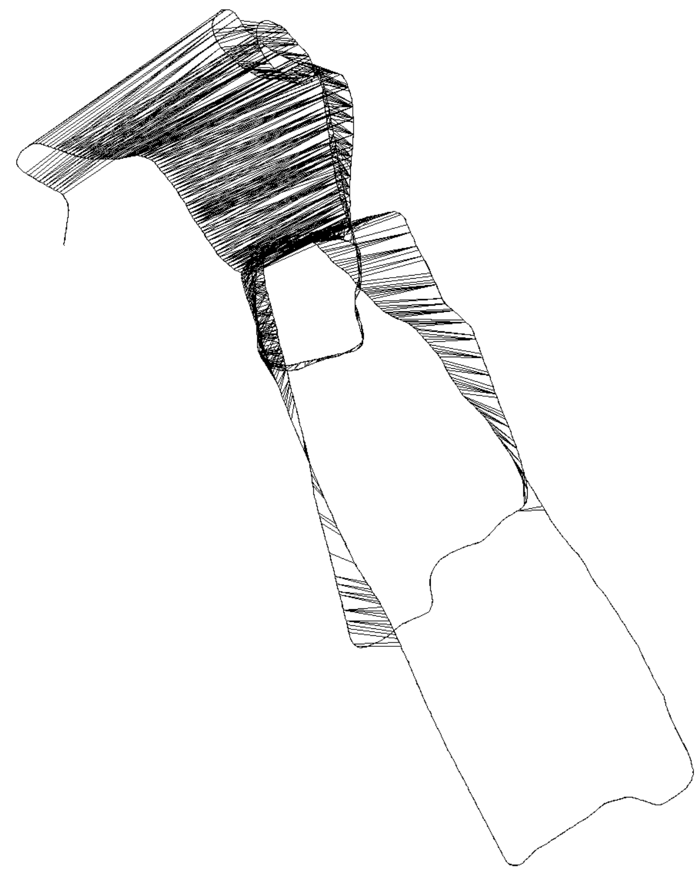
- Problem described as a graph
  - Every node corresponds to a robot position and to a laser measurement
  - An edge between two nodes represents a data-dependent spatial constraint between the nodes



KUKA Halle 22, courtesy of the Pfaffie

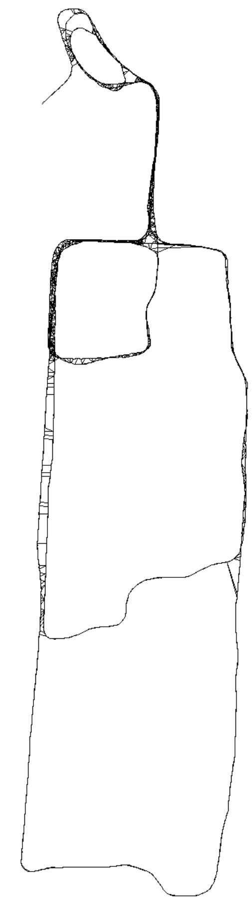
Graph-Based SLAM in a Nutshell

- Once we have the graph we determine the most likely map by “moving” the nodes



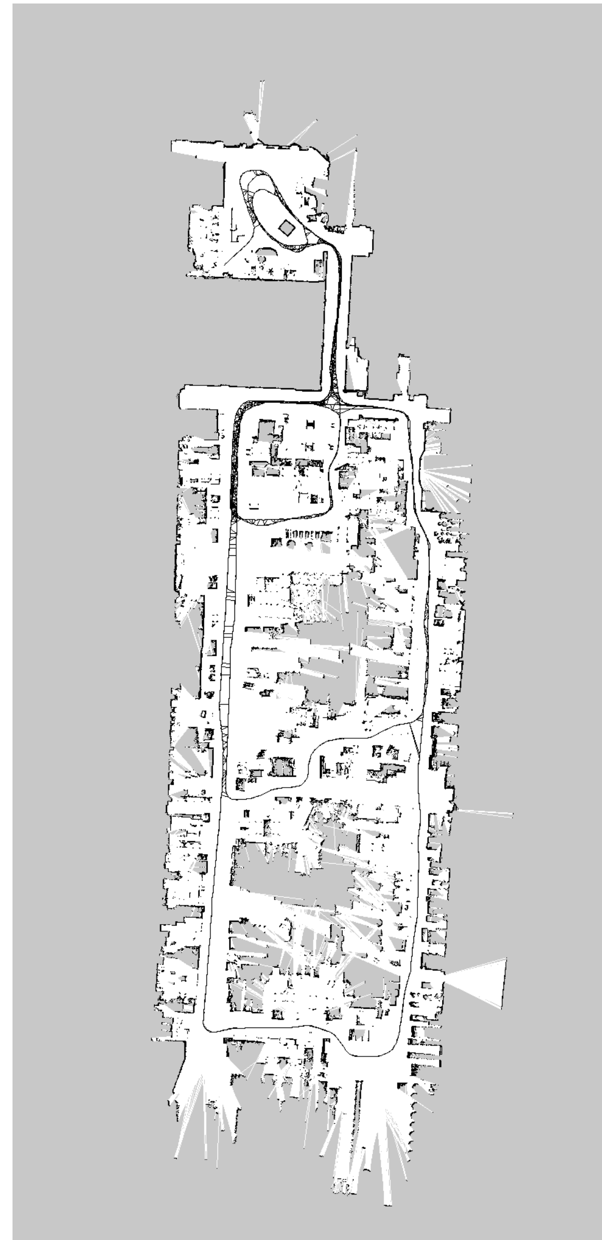
Graph-Based SLAM in a Nutshell

- Once we have the graph we determine the most likely map by “moving” the nodes
- … like this



Graph-Based SLAM in a Nutshell

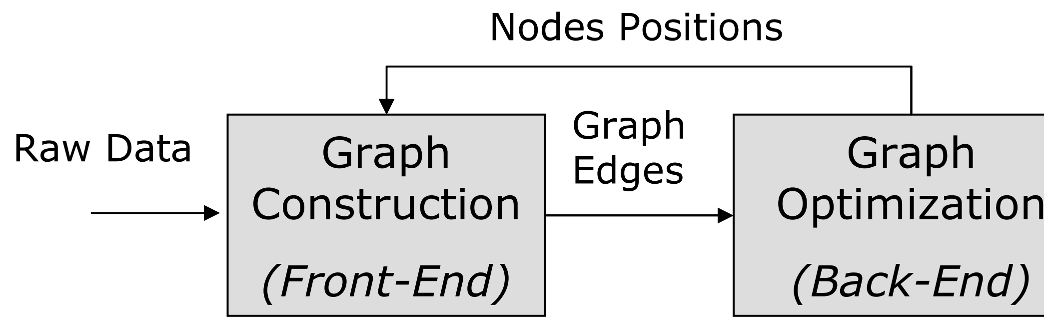
- Once we have the graph we determine the most likely map by “moving” the nodes
- … like this
- Then we render a map based on the known poses and we are all happy



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.

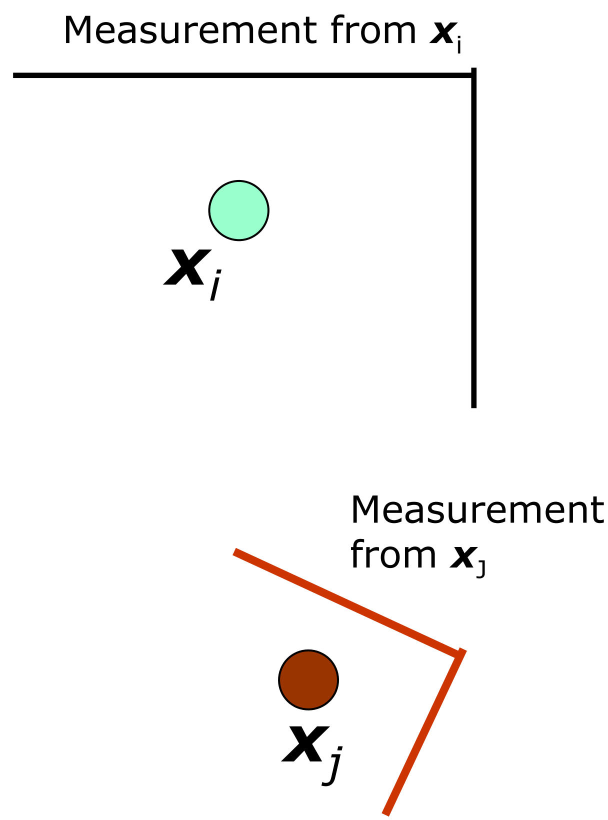


How does the Graph look like?

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

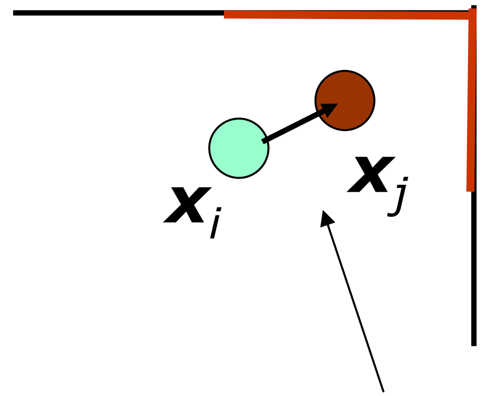
How does the Graph look like?

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



How does the Graph look like?

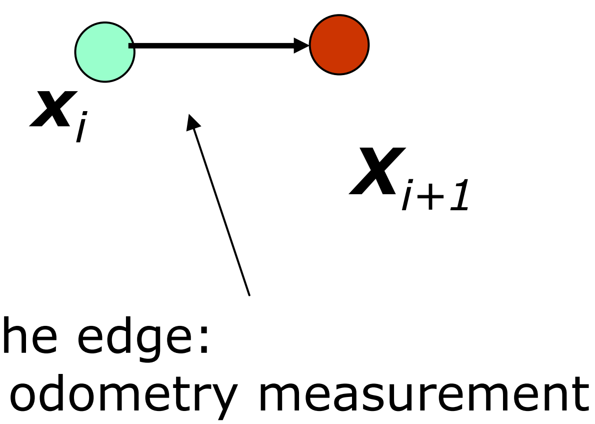
- It has ***n*** nodes $\boldsymbol{x}=\boldsymbol{x}_{1:n}$
  - Each node $\boldsymbol{x}_i$ is a 2D or 3D transformation representing the pose of the robot at time $\boldsymbol{t}_i$.
- There is a constraint e_{ij} between the node $\boldsymbol{x}_i$ and the node $\boldsymbol{x}_j$ if
  - either
 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



In the edge:
the position of $\boldsymbol{x}_j$ seen from $\boldsymbol{x}_i$, based on the ***observations***

How does the Graph look like?

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



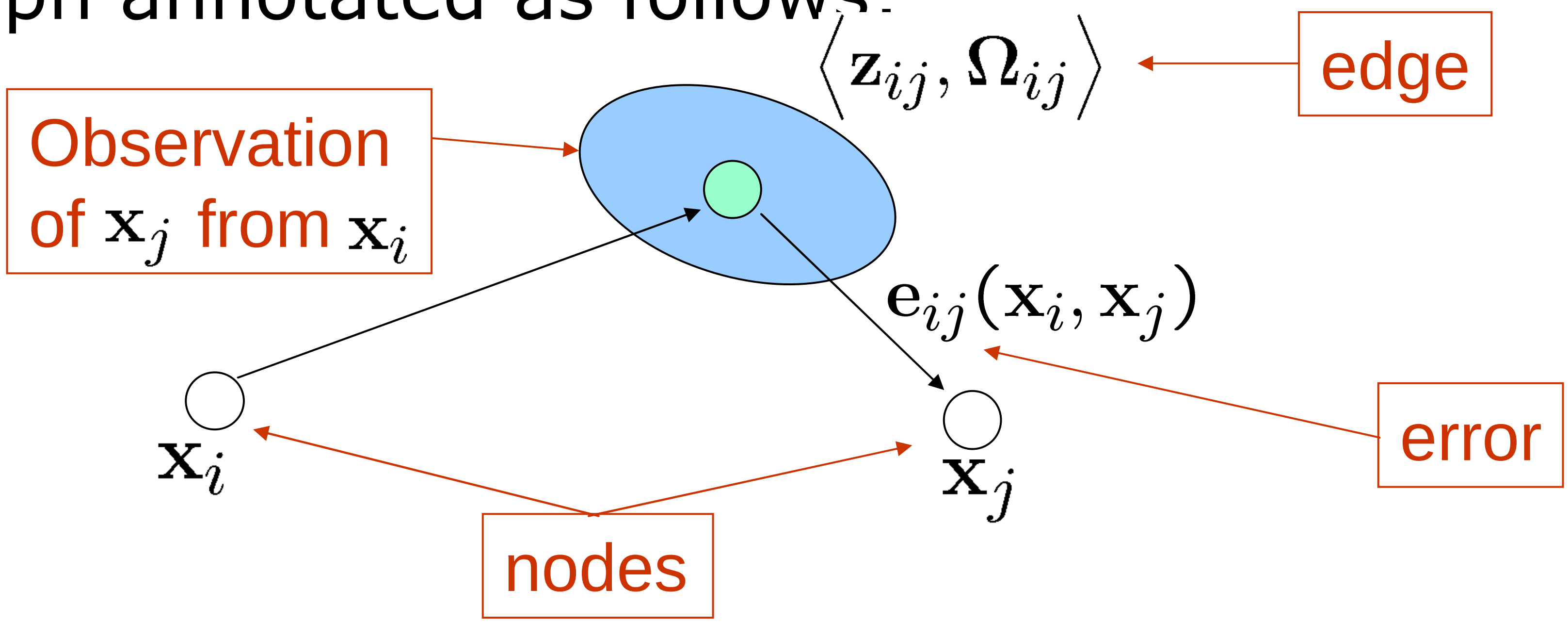
How does the Graph look like?

- To account for the different nature of the observations we add to the edge an information matrix $\boldsymbol{\Omega}_{ij}$ to encode the uncertainty of the edge.
- The “bigger” (in matrix sense) $\boldsymbol{\Omega}_{ij}$ 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:



- **Goal:**
  - Find the assignment of poses to the nodes of the graph which minimizes the negative log likelihood of the observations: $\hat{\mathbf{x}} = \text{argmin} \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}^T$

SLAM as a Least Square Problem

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

$$\begin{aligned}\hat{\mathbf{x}} &= \text{argmin} \sum_{ij} \mathbf{e}_{ij}^T(\mathbf{x}_i, \mathbf{x}_j)\mathbf{\Omega}_{ij}\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \text{argmin} \sum_{k} \mathbf{e}_{k}^T(\mathbf{x})\mathbf{\Omega}_{k}\mathbf{e}_{k}(\mathbf{x})\end{aligned}$$

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

$$\begin{aligned}\hat{\mathbf{x}} &= \operatorname{argmin} \sum_{ij} \mathbf{e}_{ij}^T(\mathbf{x}_i, \mathbf{x}_j)\boldsymbol{\Omega}_{ij}\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \operatorname{argmin} \sum_{k} \mathbf{e}_{k}^T(\mathbf{x})\boldsymbol{\Omega}_{k}\mathbf{e}_{k}(\mathbf{x})\end{aligned}$$

- We can regard each edge as a measurement, and use what we already now.
- Questions:
  - What is the state vector?

One block for each node of the graph

$$\mathbf{x}^T = \begin{pmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T & \cdots & \mathbf{x}_n^T \end{pmatrix}$$

  - What is the error function?

The Error Function

- The generic error function of a constraint characterized by a mean $\boldsymbol{z_{ij}}$ and an information $\boldsymbol{\Omega}_{ij}$ is vector of the same size of a pose $\boldsymbol{x}_i$

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathsf{t2v}(\mathbf{Z}_{ij}^{-1}(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j))$$

measurement

$\boldsymbol{x}_j$ in the reference of $\boldsymbol{x}_i$

- We can write the error as a function of all the state $\boldsymbol{x}$.

$$\mathbf{e}_{ij}(\mathbf{x}) = \mathsf{t2v}(\mathbf{Z}_{ij}^{-1}(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j))$$

- Note that the error function is 0 when

$$\mathbf{Z}_{ij} = (\mathbf{X}_i^{-1} \cdot \mathbf{X}_j)$$

The Derivative of the Error Function

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

The Derivative of the Error Function

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

The Derivative of the Error Function

- Does one error function $\boldsymbol{e_{ij}(x)}$ depend on all state variables?
  - No, only on $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$
- Is there any consequence on the *structure* of the Jacobian?
  - Yes, it will be non-zero only in the rows corresponding to $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$!

$$
\begin{aligned}
\frac{\partial \mathbf{e}_{ij}(\mathbf{x})}{\partial \mathbf{x}} &= \left(\mathbf{0} \cdots \frac{\partial \mathbf{e}_{ij}(\mathbf{x}_i)}{\partial \mathbf{x}_i} \cdots \frac{\partial \mathbf{e}_{ij}(\mathbf{x}_j)}{\partial \mathbf{x}_j} \cdots \mathbf{0} \right) \\
\mathbf{A}_{ij} &= \left(\mathbf{0} \cdots \mathbf{B}_{ij} \cdots \mathbf{C}_{ij} \cdots \mathbf{0} \right)
\end{aligned}
$$

Consequences of the Sparsity

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

$$
\begin{aligned}
\mathbf{b}^T &= \sum_{ij} \mathbf{b}_{ij}^T = \sum_{ij} \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \\
\mathbf{H} &= \sum_{ij} \mathbf{H}_{ij} = \sum_{ij} \mathbf{A}_{ij}^T \boldsymbol{\Omega} \mathbf{A}_{ij}^T
\end{aligned}
$$

- The sparse structure of $\boldsymbol{A_{ij}}$, 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 contribute s to the linear system via its coefficient vector $\boldsymbol{b}_{ij}$ and its coefficient matrix $\boldsymbol{H}_{ij}$.
  - The coefficient vector is:

$$\begin{aligned}
\mathbf{b}_{ij}^T &= \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \\
&= \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \left(\begin{array}{c} 0 \cdots \mathbf{B}_{ij} \cdots \mathbf{C}_{ij} \cdots 0 \end{array} \right) \\
&= \left(\begin{array}{c} 0 \cdots \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} \cdots \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{C}_{ij} \cdots 0 \end{array} \right)
\end{aligned}$$

  - It is non-zero only in correspondence of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

Consequences of the Sparsity (cont.)

- The coefficient matrix of an edge is:

$$
\begin{aligned}
\mathbf{H}_{ij}^T &= \mathbf{A}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \\
&= \begin{pmatrix} \vdots \\ \mathbf{B}_{ij}^T \\ \vdots \\ \mathbf{C}_{ij}^T \\ \vdots \end{pmatrix} \boldsymbol{\Omega}_{ij} \begin{pmatrix} \cdots \mathbf{B}_{ij} \cdots \mathbf{C}_{ij} \cdots \end{pmatrix} \\
&= \begin{pmatrix} & & \\ & \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \mathbf{B}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{C}_{ij} & \\ & \mathbf{C}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} & \mathbf{C}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{C}_{ij} & \\ & & \end{pmatrix}
\end{aligned}
$$

- Is non zero only in the blocks ***i,j.***

Consequences of the Sparsity (cont.)

- An edge between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in the graph contributes only
  - to the $\boldsymbol{i}^{\text{th}}$ and the $\boldsymbol{j}^{\text{th}}$ 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:

$$\Delta \mathbf{x}^T = \begin{pmatrix} \Delta \mathbf{x}_1^T & \Delta \mathbf{x}_2^T & \cdots & \Delta \mathbf{x}_n^T \end{pmatrix}$$

- Coefficient vector:

$$\mathbf{b}^T = \begin{pmatrix} \mathbf{b}_1^T & \mathbf{b}_2^T & \cdots & \mathbf{b}_n^T \end{pmatrix}$$

- System Matrix:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}^{11} & \mathbf{H}^{12} & \cdots & \mathbf{H}^{1n} \\ \mathbf{H}^{21} & \mathbf{H}^{22} & \cdots & \mathbf{H}^{2n} \\ \vdots & \ddots & & \vdots \\ \mathbf{H}^{n1} & \mathbf{H}^{n2} & \cdots & \mathbf{H}^{nn} \end{pmatrix}$$

- 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

$$\mathbf{b} = 0 \qquad \mathbf{H} = 0$$

- For each constraint
  - Compute the error

$$\mathbf{e}_{ij} = \mathrm{t2v}(\mathbf{Z}_{ij}^{-1}(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j))$$

  - Compute the blocks of the Jacobian:

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \qquad \mathbf{C}_{ij} = \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j}$$

  - Update the coefficient vector:

$$\mathbf{b}_i^T += \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij}^T \qquad \mathbf{b}_j^T += \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{C}_{ij}^T$$

  - Update the system matrix:

$$\mathbf{H}^{ii} += \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} \qquad \mathbf{H}^{ij} += \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{C}_{ij}$$

$$\mathbf{H}^{ji} += \mathbf{C}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} \qquad \mathbf{H}^{jj} += \mathbf{C}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{C}_{ij}$$

Algorithm

- ***x***: the initial guess
- While (! converged)
  - ***<H,b>*** = buildLinearSystem(***x***);
  - ***Δx*** = solveSparse(***H Δx = b***);
  - x += ***Δx;***

Exercise(s)

- Consider a 2D graph, where each pose $\boldsymbol{x_i}$ is parameterized as

$$\mathbf{x}_i^T = (x_i \; y_i \; \theta_i)$$

- Consider the error function

$$\mathbf{e}_{ij} = \text{t2v}(\mathbf{Z}_{ij}^{-1}(\mathbf{X}_i^{-1} \cdot \mathbf{X}_j))$$

- Compute the blocks of the jacobian

$$\mathbf{B}_{ij} = \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_i} \qquad \mathbf{C}_{ij} = \frac{\partial \mathbf{e}(\mathbf{x}_i, \mathbf{x}_j)}{\partial \mathbf{x}_j}$$

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

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \;=\; \mathbf{Z}_{ij}^{-1} \begin{pmatrix} -\mathbf{R}_i^T(\mathbf{t}_j - \mathbf{t}_i) \\ \theta_j - \theta_i \end{pmatrix}$$

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 $\boldsymbol{H}$.