Advanced Techniques for Mobile Robotics

Graph-based SLAM using Least Squares

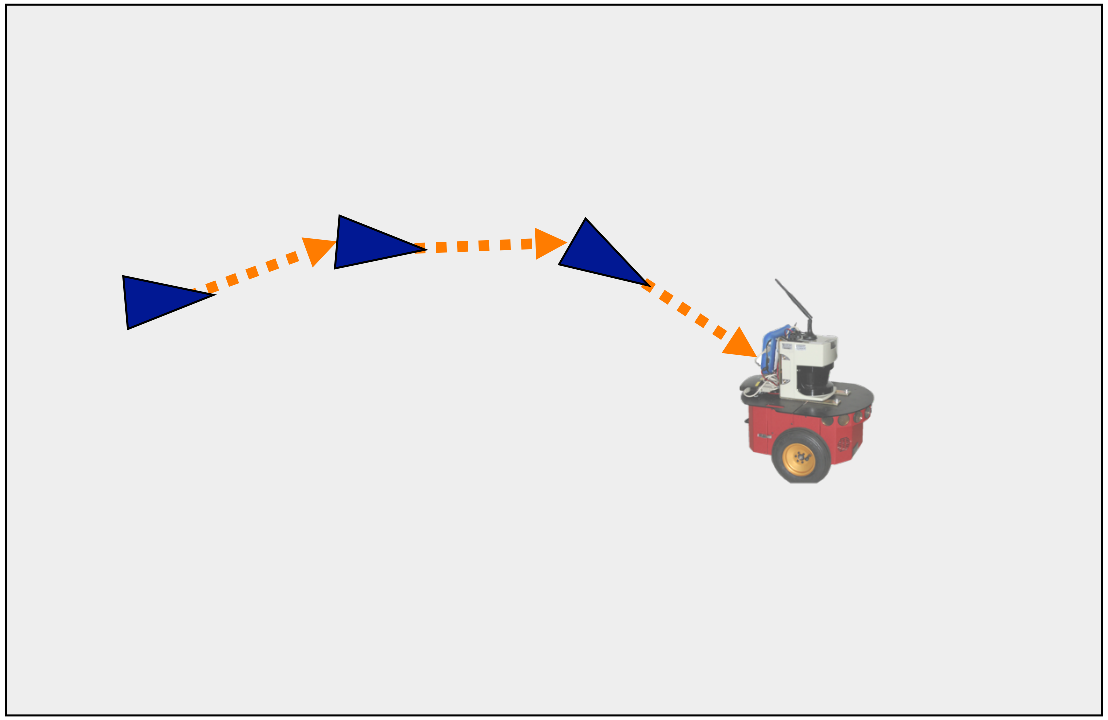
Wolfram Burgard, Cyrill Stachniss,

Kai Arras, Maren Bennewitz

UNI FREIBURG

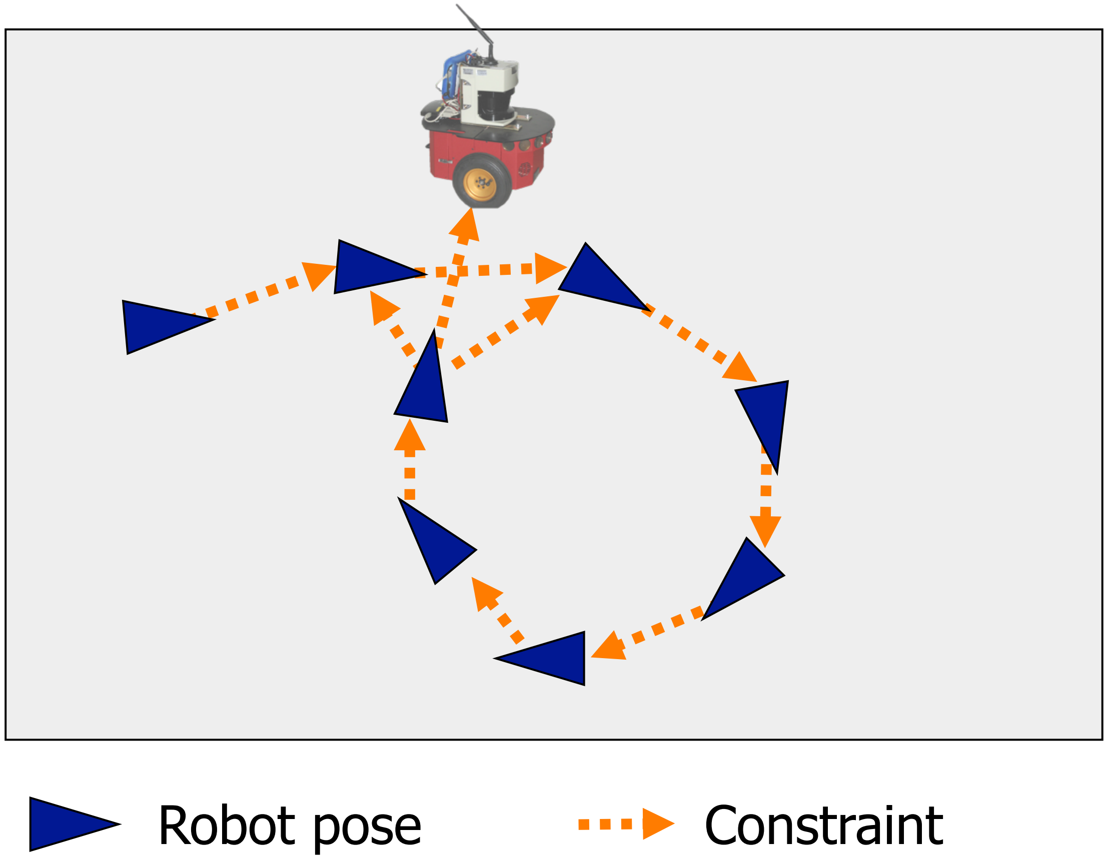
SLAM

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



SLAM

- Observing previously seen areas generates constraints between non-successive poses
- Constraints are inherently uncertain

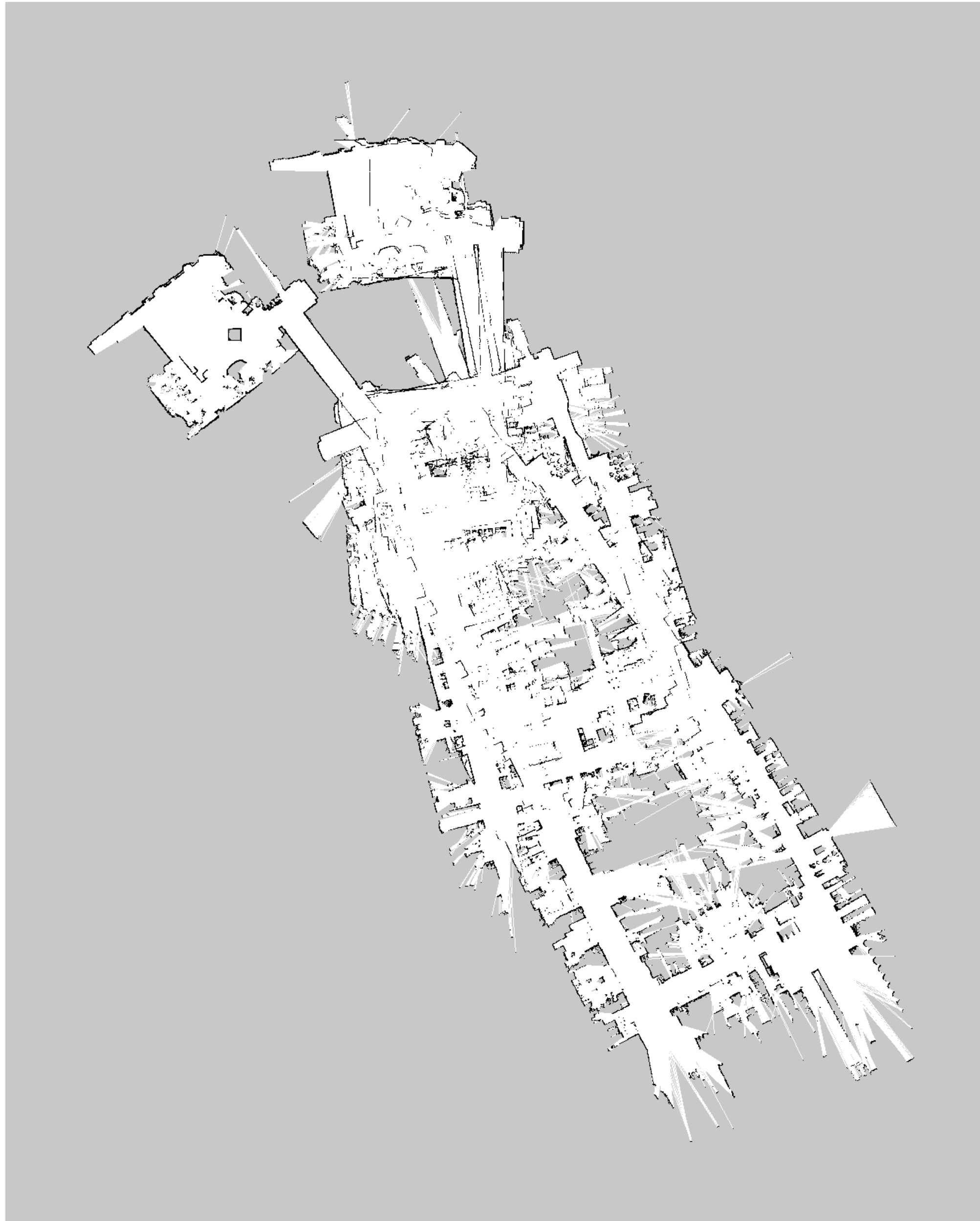


Idea of Graph-Based SLAM

- 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-Based SLAM in a Nutshell

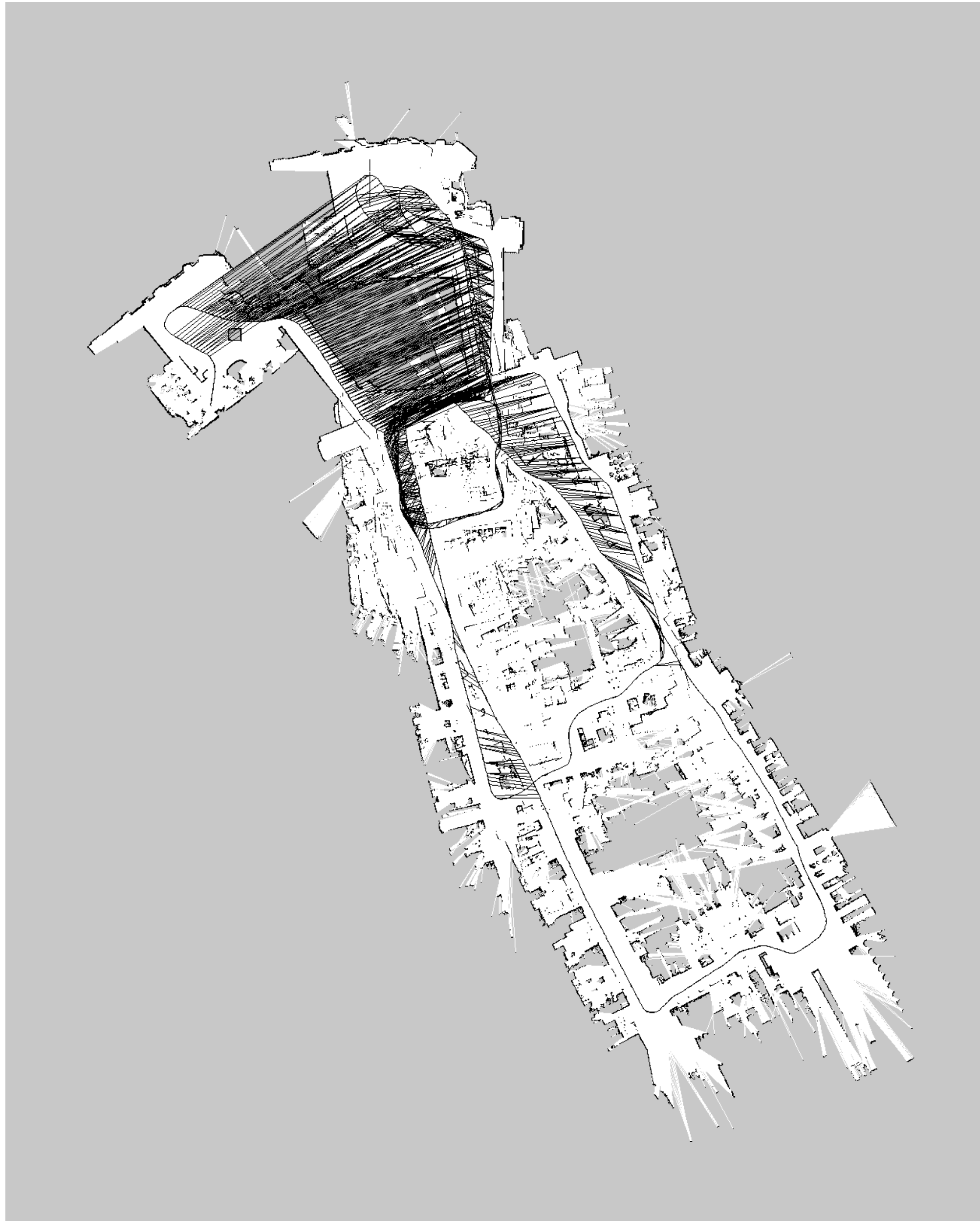
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

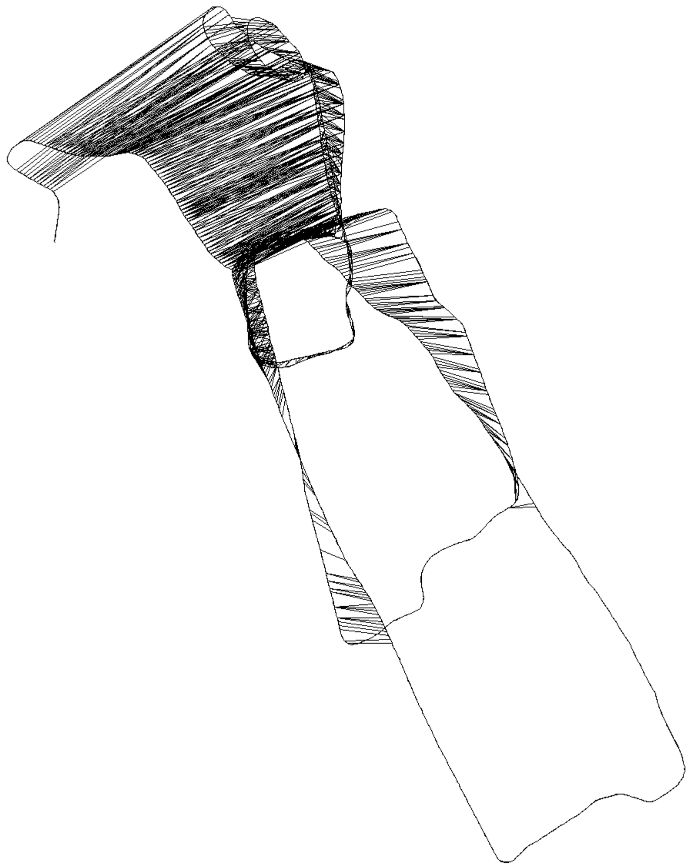
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

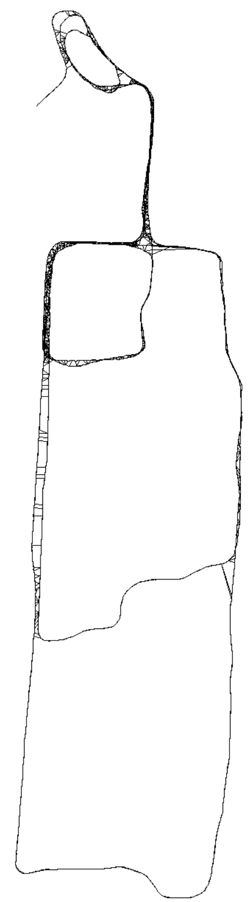
Graph-Based SLAM in a Nutshell

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



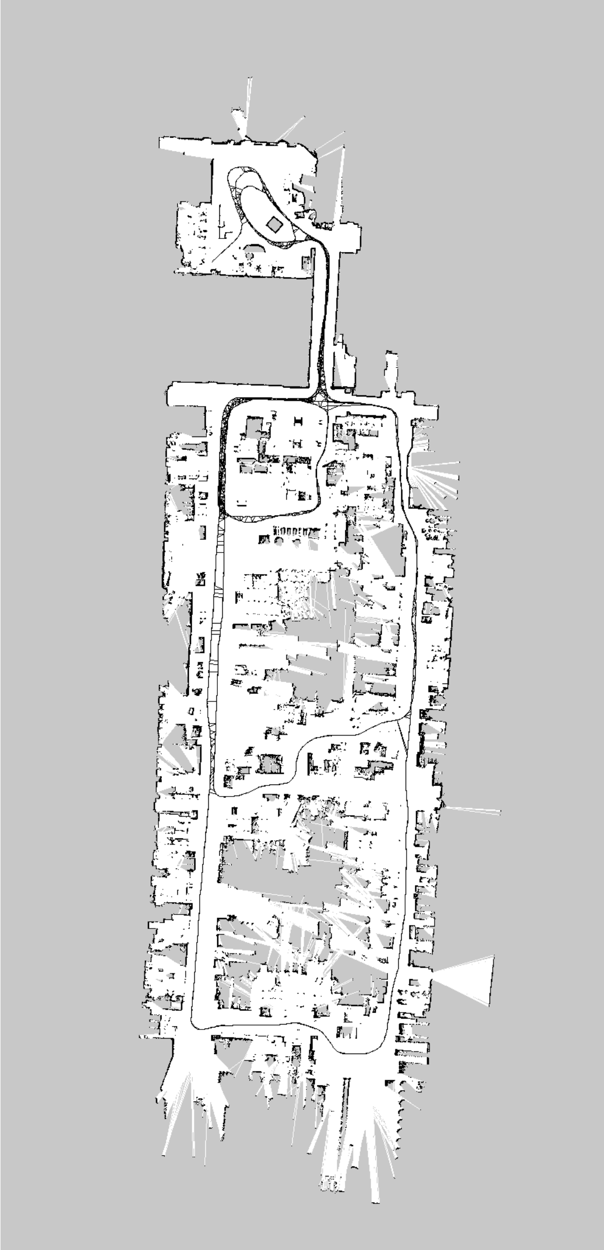
Graph-Based SLAM in a Nutshell

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

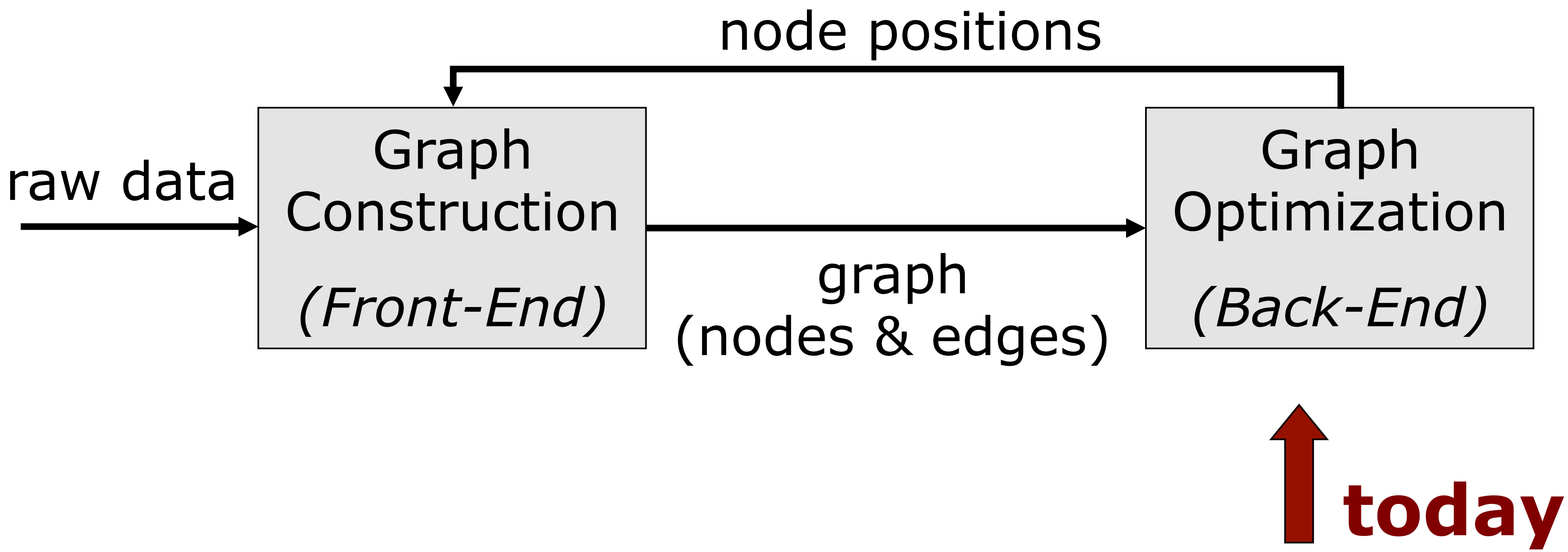


Graph-Based SLAM in a Nutshell

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



The Overall SLAM System



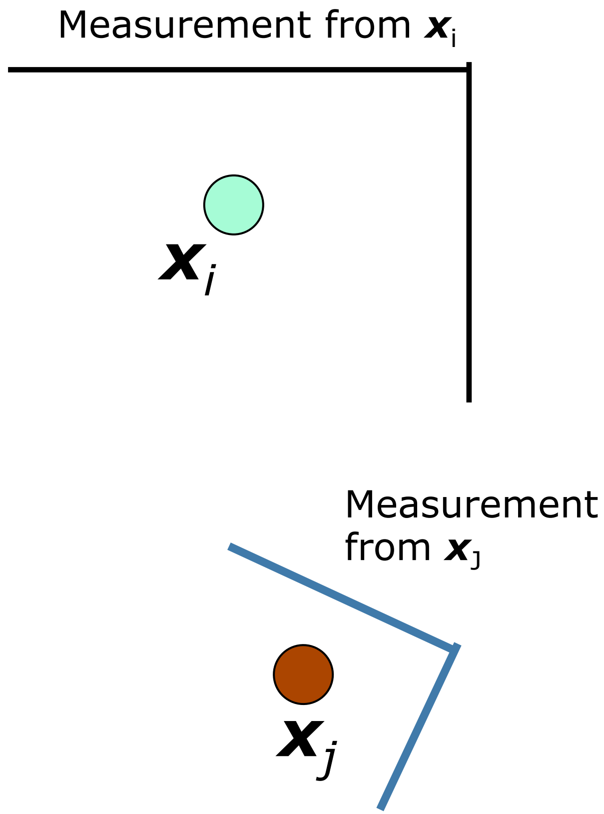
- Interleaving process of front-end and back-end
- A consistent map helps to determine new constraints by reducing the search space
- This lecture focuses only on the optimization part

The Graph

- It consists of $\boldsymbol{n}$ nodes $\mathbf{x}=\mathbf{x}_{1:n}$
- Each node $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time t_i)
- A constraint e_{ij} exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if
  - the robot observed the same part of the environment from $\mathbf{x}_i$ and $\mathbf{x}_j$ and constructs a "virtual measurement" about the position of $\mathbf{x}_j$ seen from or
  - an odometry measurement connects both poses.

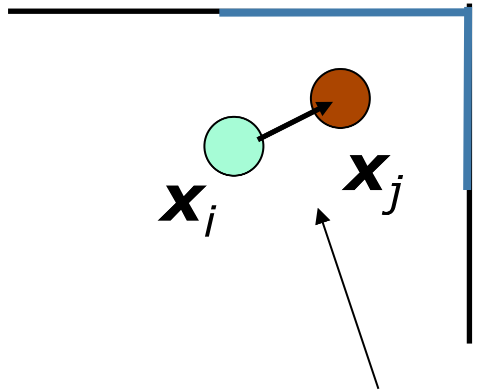
The Graph

- It consists of $\boldsymbol{n}$ nodes $\boldsymbol{x}=\boldsymbol{x}_{1:n}$
- Each node $\boldsymbol{x}_i$ is a 2D or 3D transformation (the pose of the robot at time t_i)
- A constraint e_{ij} exists between the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if
  - the robot observed the same part of the environment from $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and constructs a "virtual measurement" about the position of $\boldsymbol{x}_j$ seen from $\boldsymbol{x}_i$
    or
  - an odometry measurement connects both poses.



The Graph

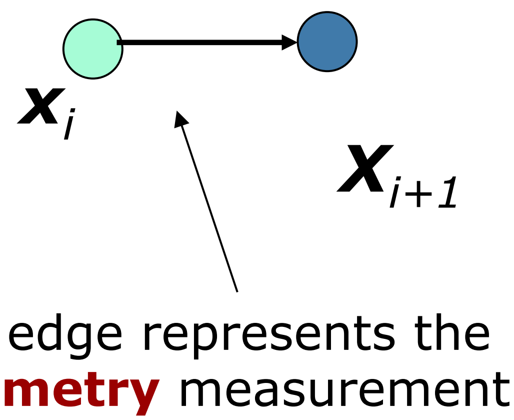
- It consists of ***n*** nodes $\mathbf{x}=\mathbf{x}_{1:n}$
- Each node $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time t_i)
- A constraint e_{ij} exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if
  - the robot observed the same part of the environment from $\mathbf{x}_i$ and $\mathbf{x}_j$ and constructs a "virtual measurement" about the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$ or
  - an odometry measurement connects both poses.



The edge represents the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$, based on the **observations**

The Graph

- It consists of $\boldsymbol{n}$ nodes $\boldsymbol{x}=\boldsymbol{x}_{1:n}$
- Each node $\boldsymbol{x}_i$ is a 2D or 3D transformation (the pose of the robot at time t_i)
- A constraint e_{ij} exists between the nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ if
  - the robot observed the same part of the environment from $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ and constructs a "virtual measurement" about the position of $\boldsymbol{x}_j$ seen from $\boldsymbol{x}_i$ or
  - an odometry measurement connects both poses.



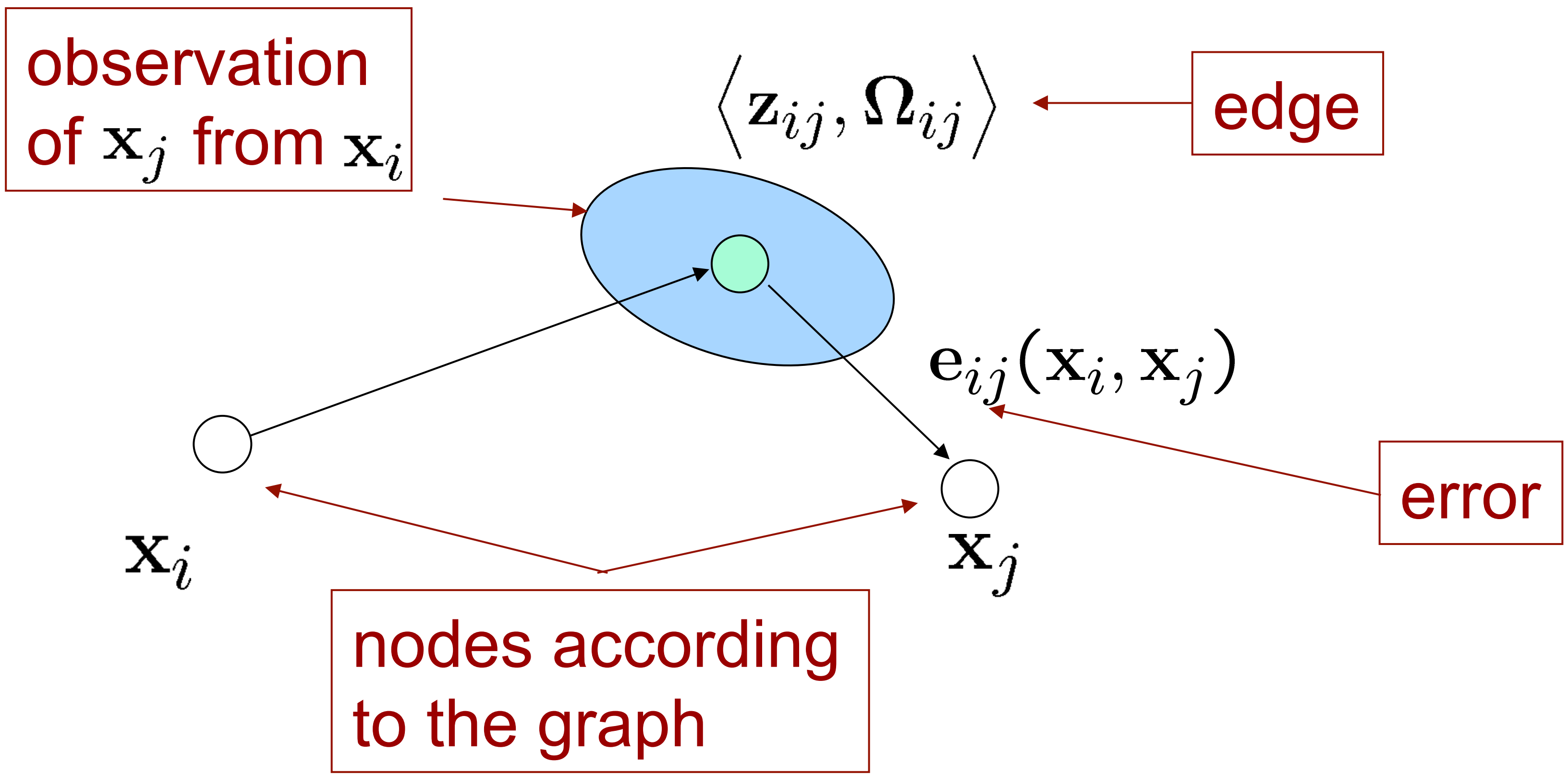
The Edge Information Matrices

- Observations are affected by noise
- We use an information matrix $\boldsymbol{\Omega}_{ij}$ for each edge to encode the uncertainty of the edge
- The "bigger" $\boldsymbol{\Omega}_{ij}$, the more the edge "matters" in the optimization procedure

Questions:

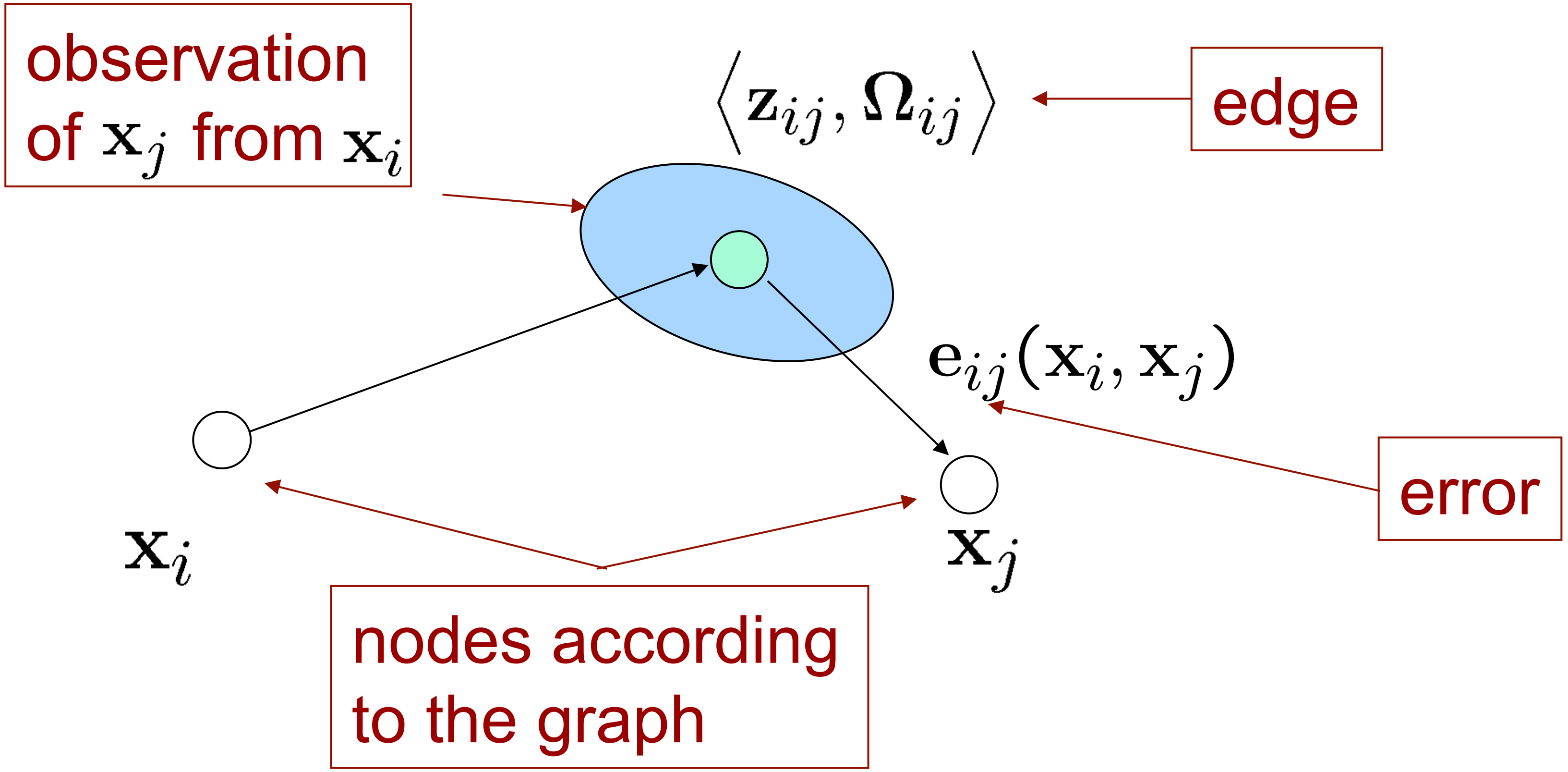
- What do the information matrices look like in case of scan-matching vs. odometry?
- What should these matrices look like in a long, featureless corridor?

Pose Graph



observation of $\mathbf{x}_j$ from $\mathbf{x}_i$
$\langle \mathbf{z}_{ij}, \mathbf{\Omega}_{ij} \rangle$
edge
$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$
error
$\mathbf{x}_i$
$\mathbf{x}_j$
nodes according to the graph

Pose Graph



- **Goal:**

$$\hat{\mathbf{x}} = \operatorname*{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$$

SLAM as a Least Squares Problem

- The error function looks suitable for least squares error minimization

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

SLAM as a Least Squares Problem

- The error function looks suitable for least squares error minimization

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

Questions:

- What is the state vector?

- Specify the error function!

SLAM as a Least Squares Problem

- The error function looks suitable for least squares error minimization

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

Questions:

- What is the state vector?

One block for each node of the graph

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

- Specify the error function!

The Error Function

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

$$\mathrm{e}_{ij}(\mathrm{x}_i, \mathrm{x}_j) = \mathrm{t2v}(\underbrace{\mathrm{Z}_{ij}^{-1}}_{\text{measurement}}\underbrace{(\mathrm{X}_i^{-1}\mathrm{X}_j)}_{\boldsymbol{x}_j \text{ in the reference of } \boldsymbol{x}_i})$$

- The error as a function of all the state ***x*:**

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

- The error function is 0 when

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

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

$$\mathbf{e}_{ij}(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}) = \mathbf{e}_{ij}(\mathbf{x}) + \mathbf{J}_{ij}\boldsymbol{\Delta}\mathbf{x}$$

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

Derivative of the Error Function

- Does one error function $\mathbf{e}_{ij}(\mathbf{x})$ depend on all state variables?

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?

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 $\boldsymbol{x_j}$!

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

Jacobians and Sparsity

- The error function $\mathbf{e}_{ij}$ of one constraint depends only on the two parameter blocks $\mathbf{x}_i$ and $\mathbf{x}_j$

$$\mathrm{e}_{ij}(\mathrm{x}) = \mathrm{e}_{ij}(\mathrm{x}_i, \mathrm{x}_j)$$

- Thus, the Jacobian will be 0 everywhere but in the columns of $\mathbf{x}_i$ and $\mathbf{x}_j$.

$$\mathbf{J}_{ij} = \left(\mathbf{0} \cdots \mathbf{0} \quad \underbrace{\frac{\partial \mathbf{e}(\mathbf{x}_i)}{\partial \mathbf{x}_i}}_{\mathbf{A}_{ij}} \quad \mathbf{0} \cdots \mathbf{0} \quad \underbrace{\frac{\partial \mathbf{e}(\mathbf{x}_j)}{\partial \mathbf{x}_j}}_{\mathbf{B}_{ij}} \quad \mathbf{0} \cdots \mathbf{0} \right)$$

Consequences of the Sparsity

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

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

- The sparse structure of $\boldsymbol{J_{ij}}$ will result in a sparse structure of $\boldsymbol{H}$
- This structure reflects the adjacency matrix of the graph

Illustration of the Structure

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

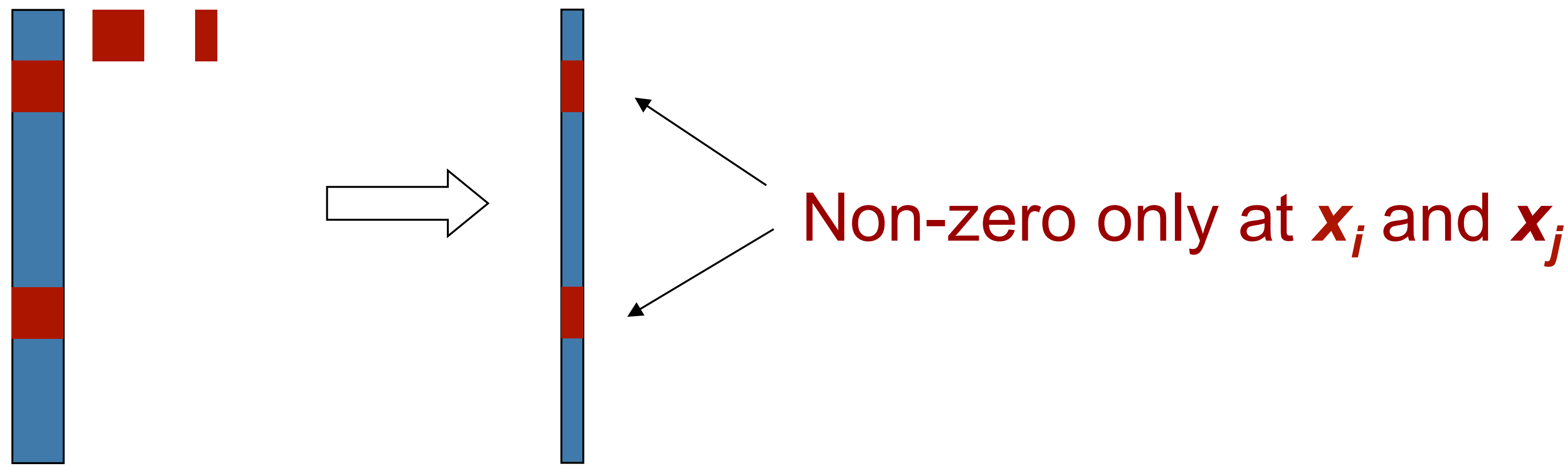
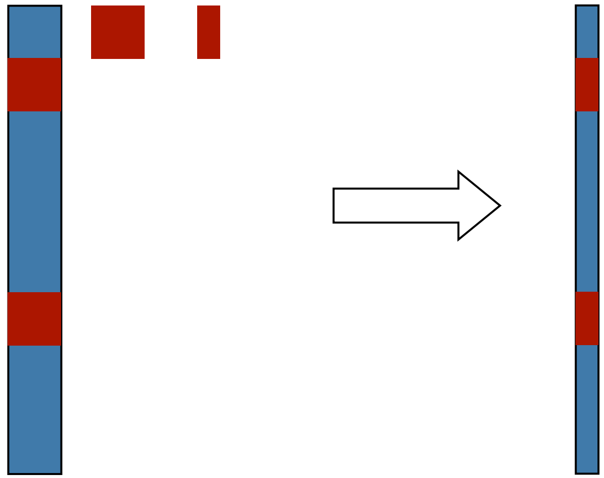


Illustration of the Structure

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



Non-zero only at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

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

Non-zero on the main diagonal at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

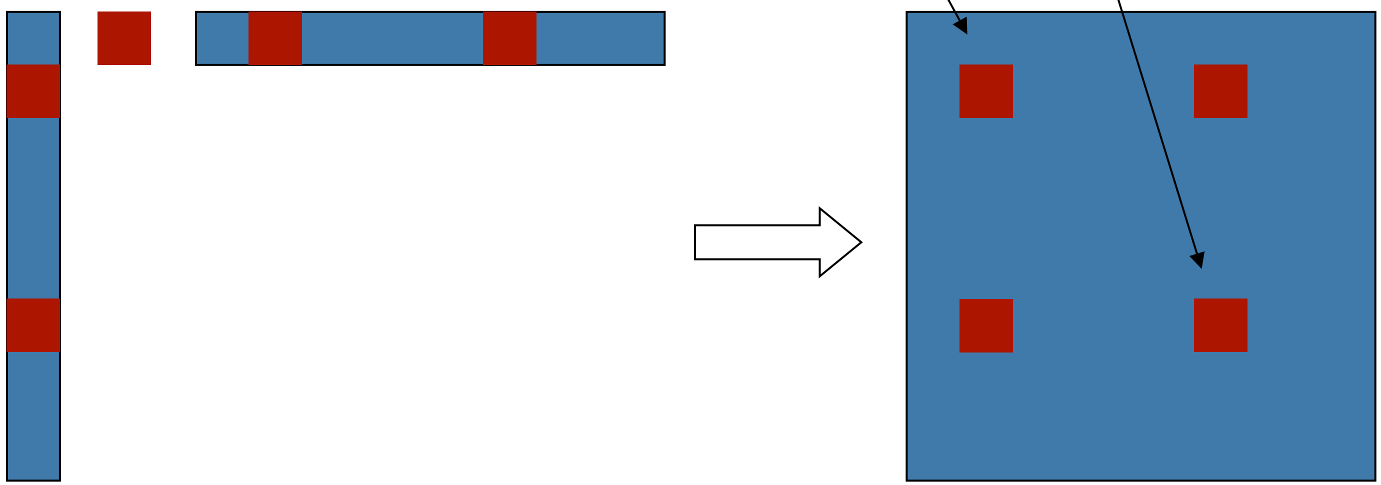
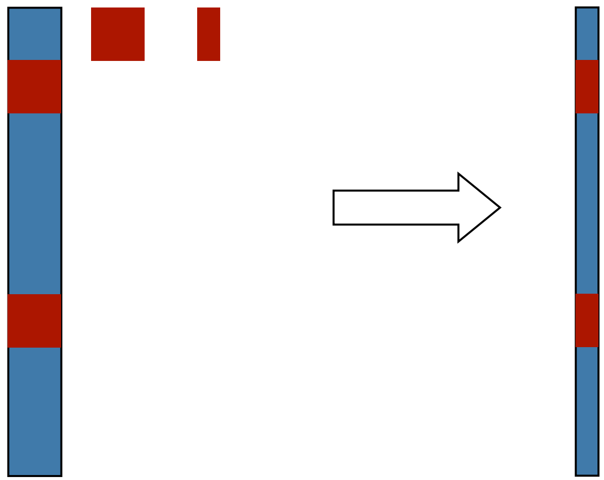


Illustration of the Structure

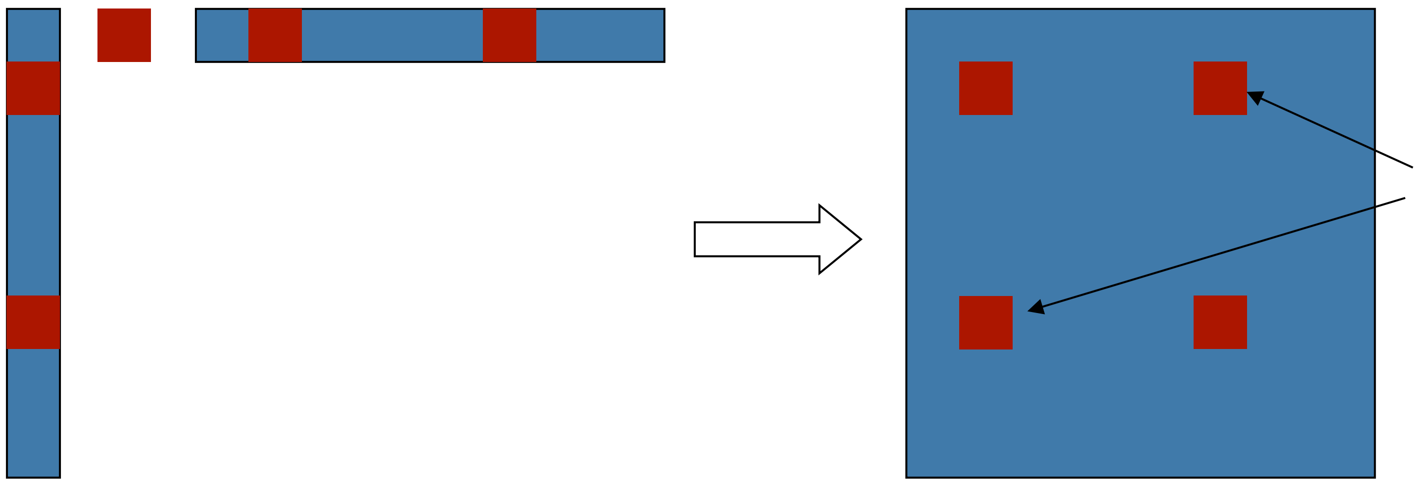
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$$



Non-zero only at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$

$$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}$$

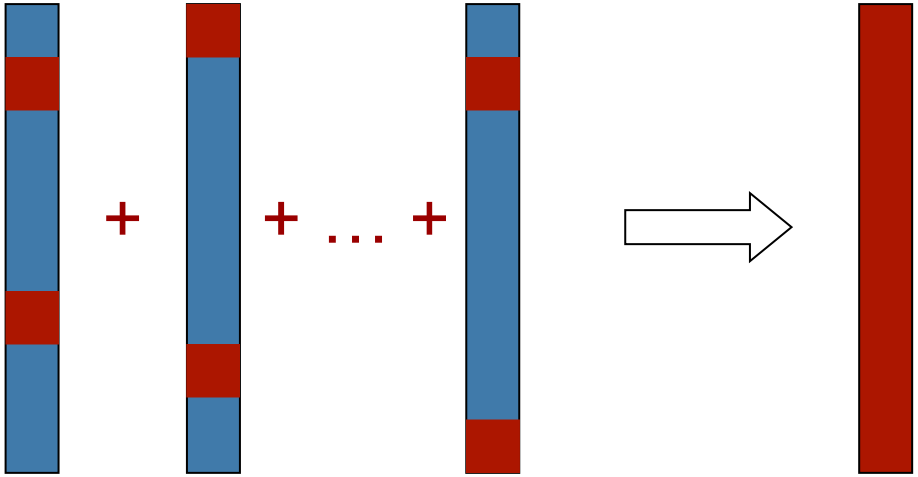
Non-zero on the main diagonal at $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$



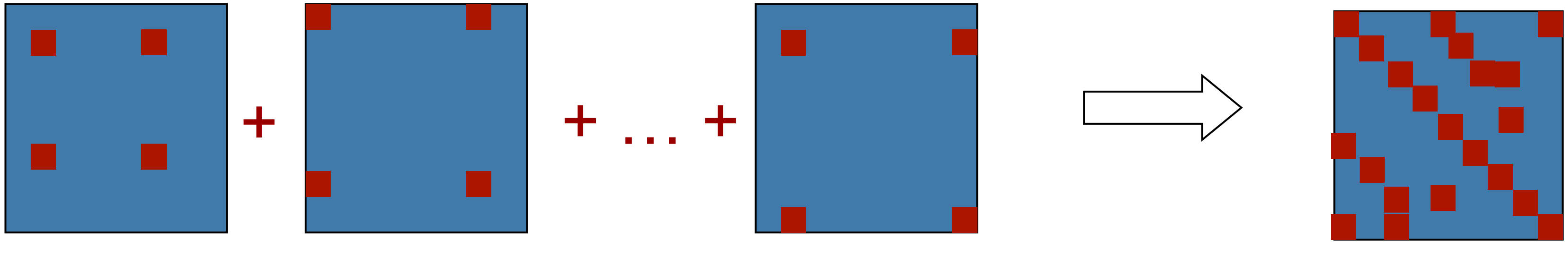
... and at the blocks *ij*,*ji*

Illustration of the Structure

$$\mathbf{b} = \sum_{ij} \mathbf{b}_{ij}$$



$$\mathbf{H} = \sum_{ij} \mathbf{H}_{ij}$$



Consequences of the Sparsity

- An edge of the graph contributes to the linear system via its coefficient vector $\boldsymbol{b_{ij}}$ and its coefficient matrix $\boldsymbol{H_{ij}}$.
- The coefficient vector is:

$$
\begin{aligned}
\mathbf{b}_{ij}^T &= \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij} \\
&= \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \begin{pmatrix} 0 \cdots \mathbf{A}_{ij} \cdots \mathbf{B}_{ij} \cdots 0 \end{pmatrix} \\
&= \begin{pmatrix} 0 \cdots \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{A}_{ij} \cdots \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{B}_{ij} \cdots 0 \end{pmatrix}
\end{aligned}
$$

- It is non-zero only at the indices corresponding to $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$

Consequences of the Sparsity

- The coefficient matrix of an edge is:

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

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

Sparsity Summary

- 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,
  - 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:

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

- System Matrix:

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

- The linear system is a block system with ***n*** blocks, one for each node of the graph.

Building the Linear System

- **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:

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

  - Update the coefficient vector:

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

  - Update the system matrix:

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

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

Algorithm

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

How to Solve the Linear System?

- Linear system $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$
- Can be solved by matrix inversion (in theory)
- In practice:
  - Cholesky factorization
  - QR decomposition
  - Iterative methods such as conjugate gradients (for large systems)
- In Octave, use the backslash operator

```
delta_x = -H\b
```

Example on the Blackboard...

Trivial 1D Example

$$x_1 \xrightarrow{z_{12} = 1} x_2$$

- Two nodes and one observation

$$
\begin{aligned}
\mathbf{x} &= (x_1\, x_2)^T = (0\, 0) \\
\mathbf{z}_{12} &= 1 \\
\mathbf{\Omega} &= 2 \\
\mathbf{e}_{12} &= = z_{12} - (x_2 - x_1) = 1 - (0 - 0) = 1 \\
\mathbf{J}_{12} &= (1 \;\; -1) \\
\mathbf{b}_{12}^T &= \mathbf{e}_{12}^T \mathbf{\Omega}_{12} \mathbf{J}_{12} = (2 \;\; -2) \\
\mathbf{H}_{12} &= \mathbf{J}_{12}^T \mathbf{\Omega} \mathbf{J}_{12} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} \\
\mathbf{\Delta x} &= -\mathbf{H}_{12}^{-1} b_{12}
\end{aligned}
$$

BUT $\det(\mathbf{H}) = 0$ **???**

What Went Wrong?

- The constraint only specifies a **relative constraint** between both nodes
- Any poses for the nodes would be fine as long a their relative coordinates fit
- **One node needs to be fixed**

$$
\begin{aligned}
\mathbf{H} &= \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}} \\
\Delta\mathbf{x} &= -\mathbf{H}^{-1} b_{12} \\
\Delta\mathbf{x} &= (0\,1)^T
\end{aligned}
$$

constraint that sets ***x_1=0***

Exercise

- Consider a 2D graph where each pose ***x_i*** is parameterized as

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

- Consider the error function

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

- Compute the blocks of the Jacobian ***J***

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

- 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

- The back-end part of the SLAM problem can be effectively solved with least squares error minimization
- The ***H*** matrix is typically sparse
- This sparsity allows for efficiently solving the linear system
- One of the state-of-the-art solutions to compute the maximum likelihood estimate