# Robot Mapping

# Least Squares Approach to SLAM

**Cyrill Stachniss**

UNI FREIBURG

AIS Autonomous Intelligent Systems

# Three Main SLAM Paradigms

| Kalman filter | Particle filter | **Graph-based** |
| --- | --- | --- |

**least squares approach to SLAM**

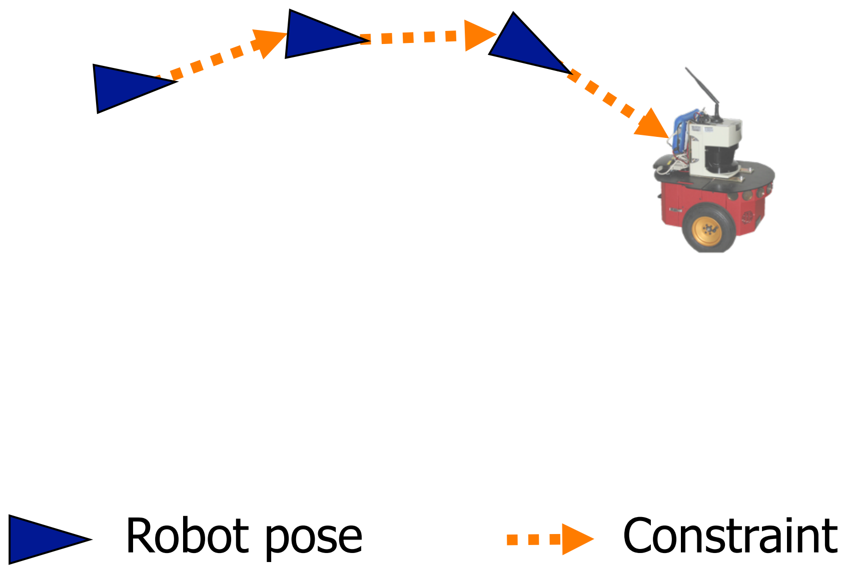
# Least Squares in General

- Approach for computing a solution for an **overdetermined system**
- "More equations than unknowns"
- Minimizes the **sum of the squared errors** in the equations
- Standard approach to a large set of problems

**Today: Application to SLAM**

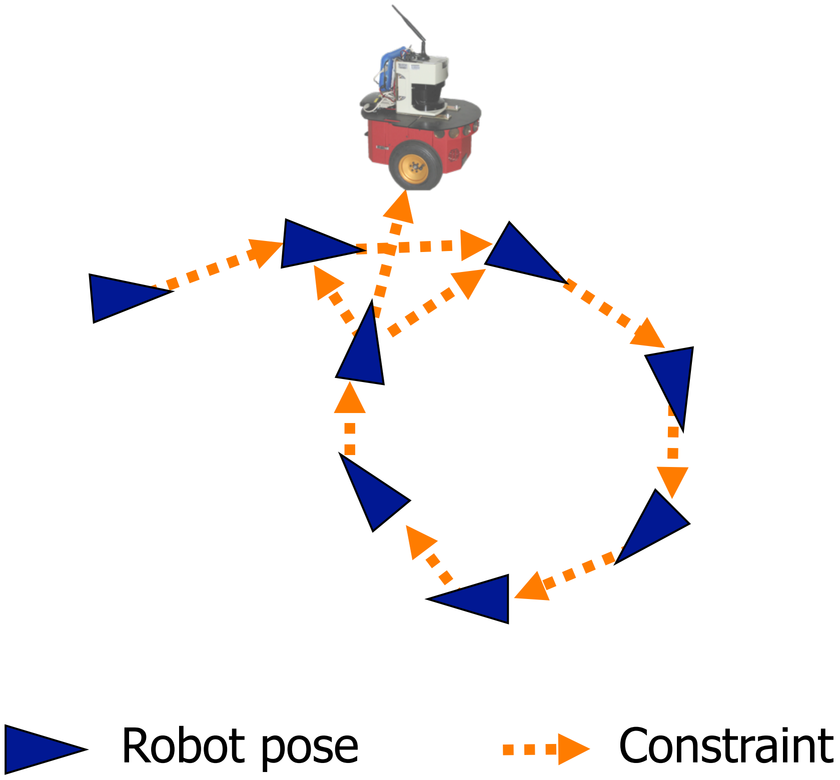
# Graph-Based SLAM

- 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

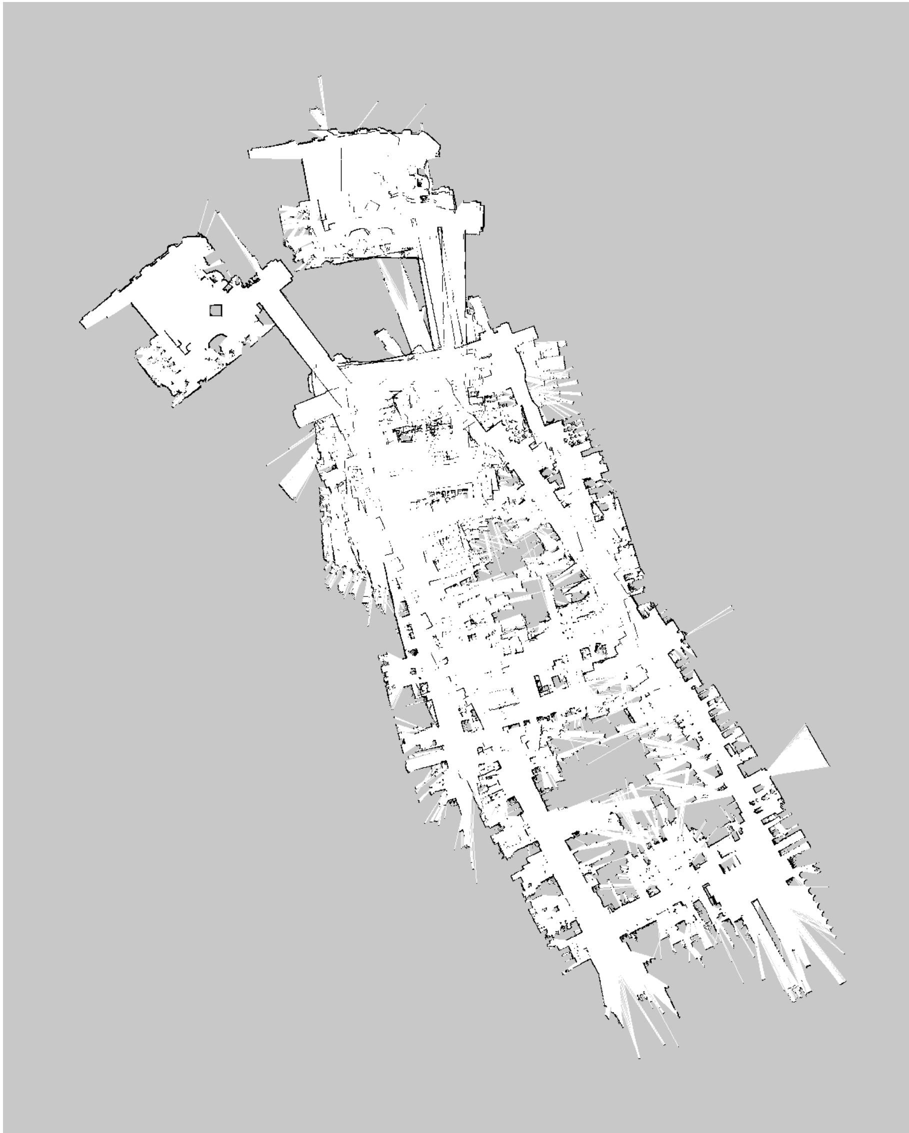


# Idea of Graph-Based SLAM

- 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

# Graph-Based SLAM in a Nutshell

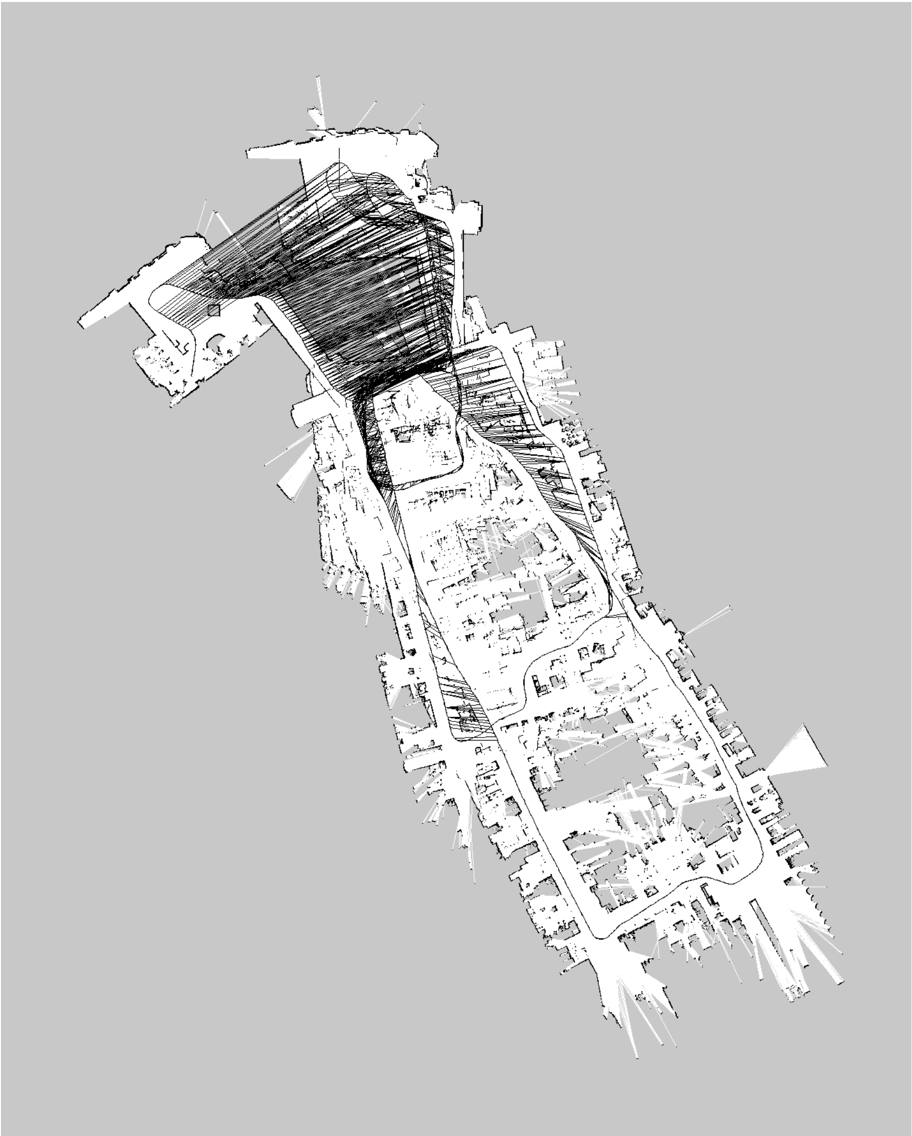
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

# Graph-Based SLAM in a Nutshell

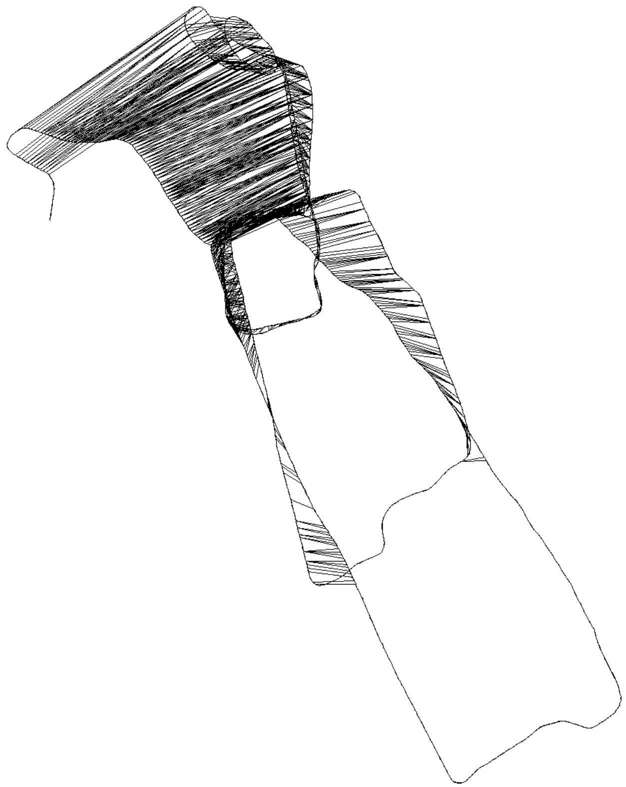
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

# Graph-Based SLAM in a Nutshell

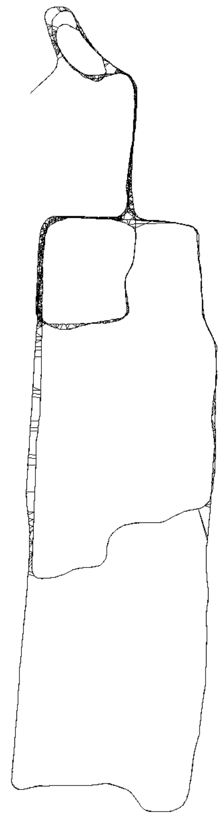
- Once we have the graph, we determine the most likely map by correcting the nodes



# Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes

  … like this

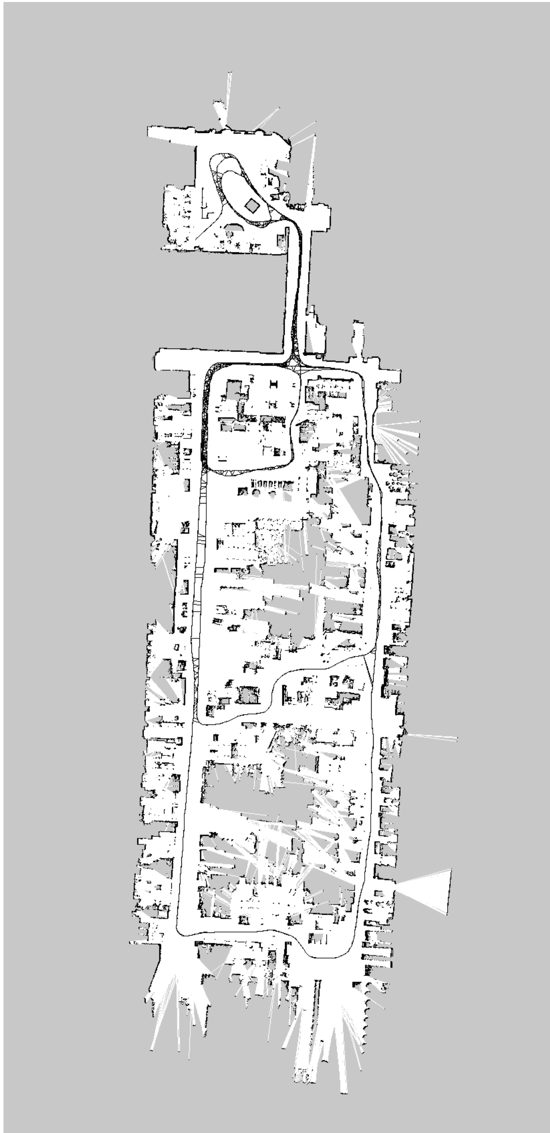


# Graph-Based SLAM in a Nutshell

- Once we have the graph, we determine the most likely map by correcting the nodes

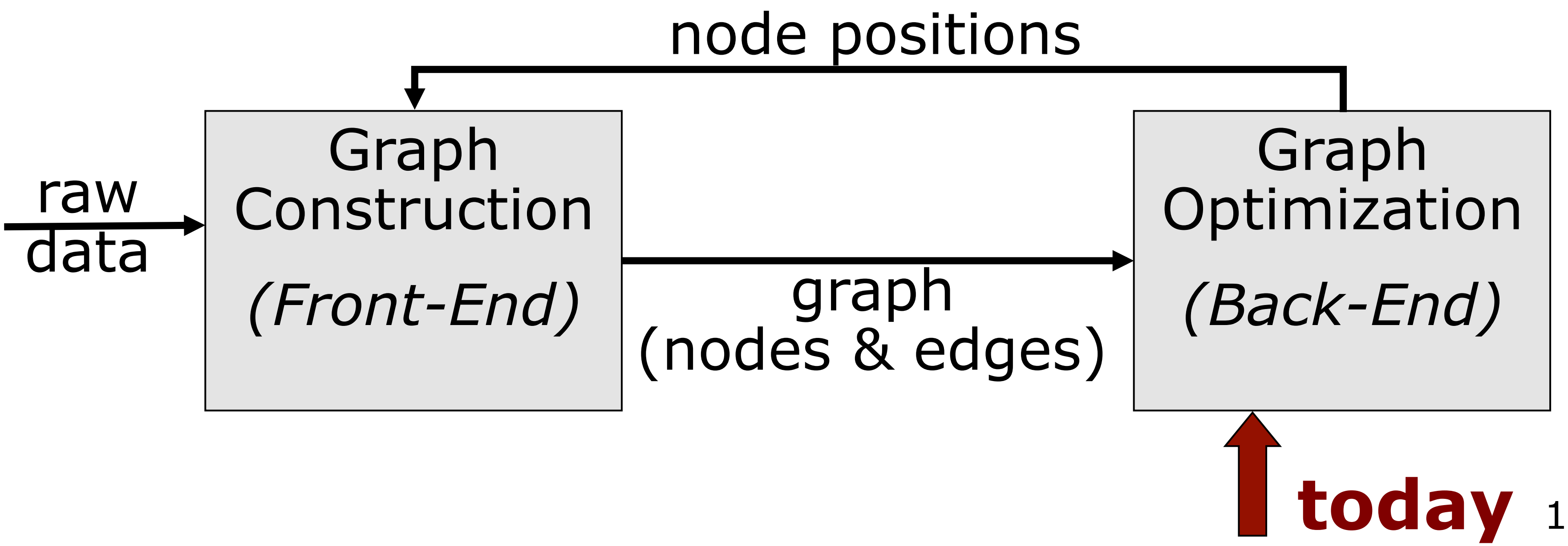
  … like this

- Then, we can render a map based on the known poses



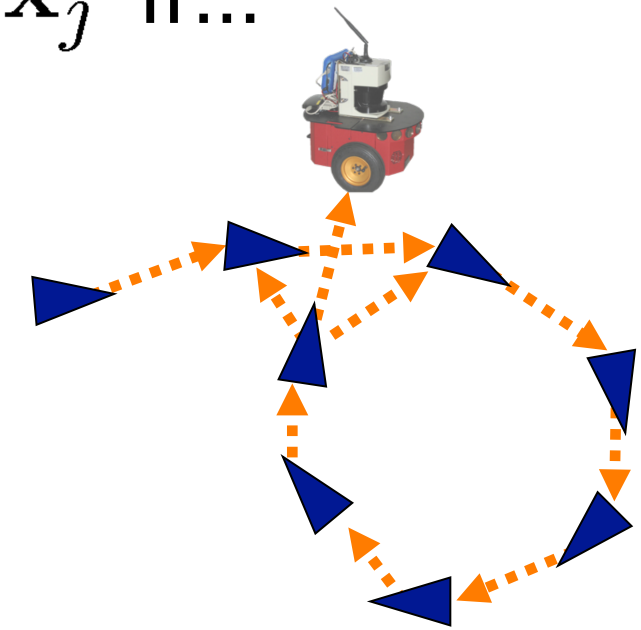
# The Overall SLAM System

- Interplay 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



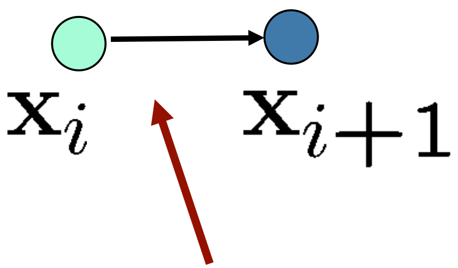
# The Graph

- It consists of n nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each $\mathbf{x}_i$ is a 2D or 3D transformation (the pose of the robot at time $t_i$)
- A constraint/edge exists between the nodes $\mathbf{x}_i$ and $\mathbf{x}_j$ if…



# Create an Edge If… (1)

- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



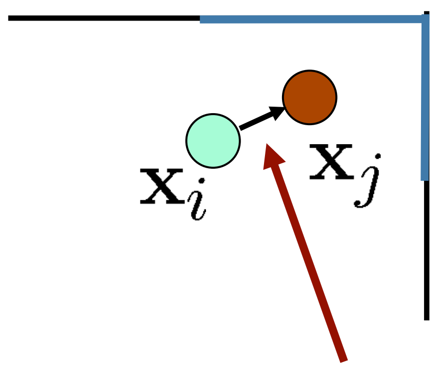
# Create an Edge If... (2)

- ...the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$
- Construct a **virtual measurement** about the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$


Measurement from $\mathbf{x}_i$ Measurement from $\mathbf{x}_j$

# Create an Edge If… (2)

- …the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$
- Construct a **virtual measurement** about the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$




Edge represents the position of $\mathbf{x}_j$ seen from $\mathbf{x}_i$ based on the **observation**

# Transformations

- Transformations can be expressed using **homogenous coordinates**
- Odometry-Based edge

$$(\mathbf{X}_i^{-1}\mathbf{X}_{i+1})$$

- Observation-Based edge

$$(\mathbf{X}_i^{-1}\mathbf{X}_j)$$

↑

How node i sees node j

# Homogenous Coordinates

- N-dim space expressed in N+1 dim
- 4D space for modeling the 3D space
- To HC: $(x, y, z)^T \rightarrow (x, y, z, 1)^T$
- Backwards: $(x, y, z, w)^T \rightarrow (\frac{x}{w}, \frac{y}{w}, \frac{z}{w})^T$
- Vector in HC: $v = (x, y, z, w)^T$
- Translation:

$$T = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Rotation:

$$R = \begin{pmatrix} R^{3D} & 0 \\ 0 & 1 \end{pmatrix}$$

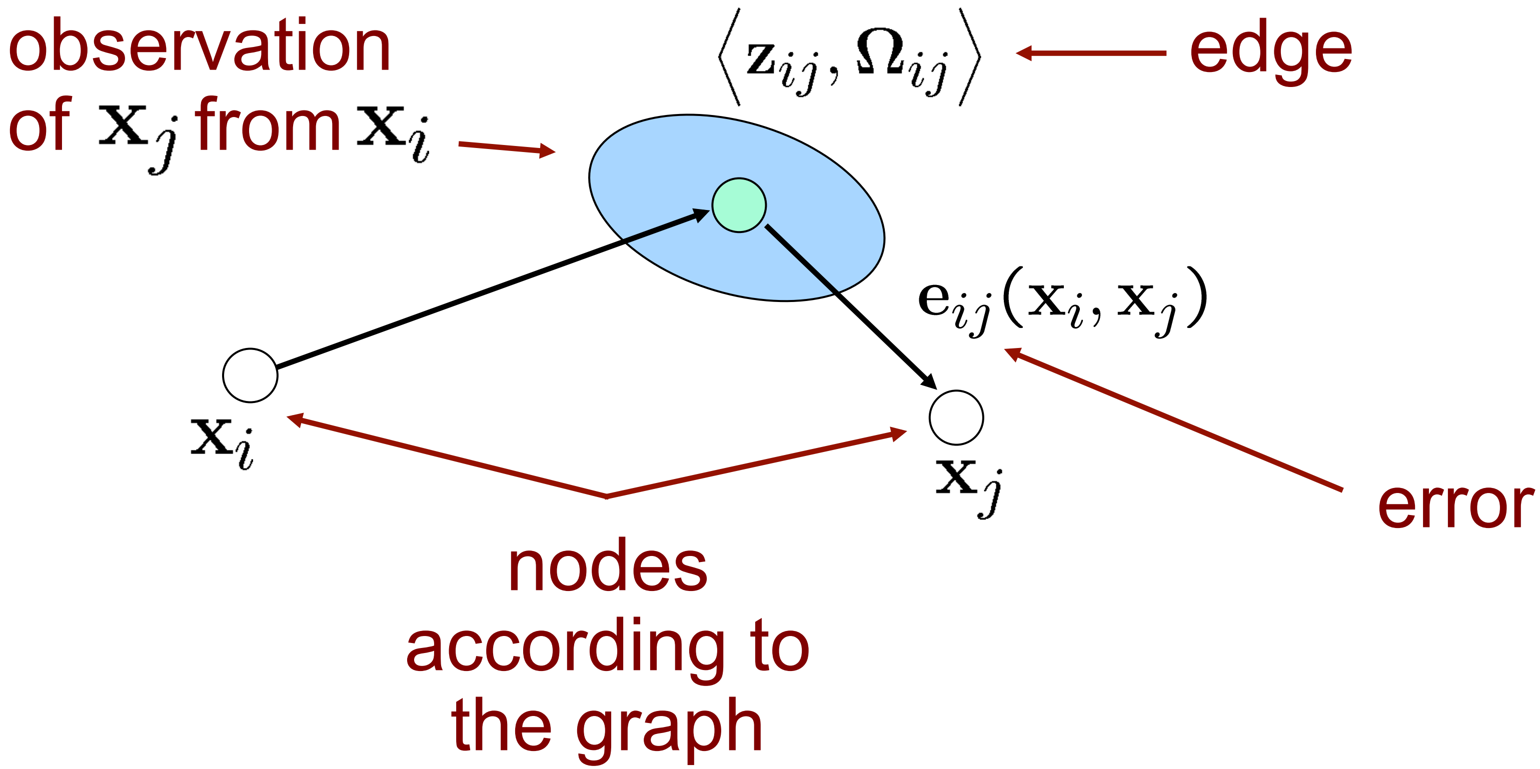
# The Edge Information Matrices

- 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

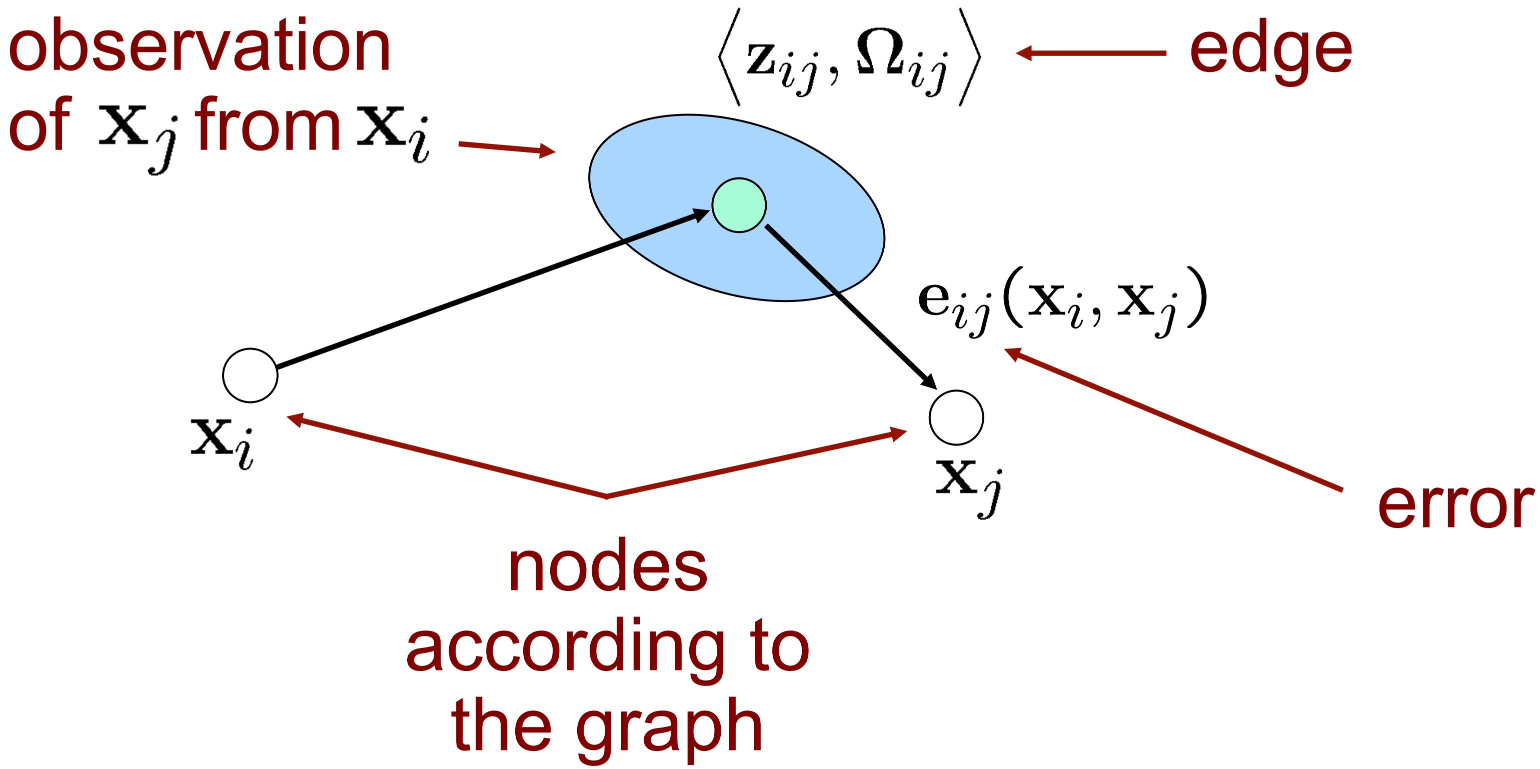
**Questions**

- 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
$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j)$
$\mathbf{x}_i$
$\mathbf{x}_j$
error
nodes
according to
the graph

# Pose Graph




- **Goal:** $\mathbf{x}^* = \underset{\mathbf{x}}{\text{argmin}} \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$

# Least Squares SLAM

- This error function looks suitable for least squares error minimization

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

# Least Squares SLAM

- This error function looks suitable for least squares error minimization

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_k \mathbf{e}_k^T(\mathbf{x}) \boldsymbol{\Omega}_k \mathbf{e}_k(\mathbf{x})$$

**Questions:**

- What is the state vector?

- Specify the error function!

# Least Squares SLAM

- This error function looks suitable for least squares error minimization

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_k \mathbf{e}_k^T(\mathbf{x}) \mathbf{\Omega}_k \mathbf{e}_k(\mathbf{x})$$

**Questions:**

- What is the state vector?

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

One block for each node of the graph

- Specify the error function!

# The Error Function

- Error function for a single constraint

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

measurement

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

- Error as a function of the whole state vector

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

- Error takes a value of zero if

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

# 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 $\mathbf{x}$ via Taylor expansion

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

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

# Derivative of the Error Function

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

# Derivative of the Error Function

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

  ➡ No, only on $\mathbf{x}_i$ and $\mathbf{x}_j$

# Derivative of the Error Function

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

  ➡ No, only on $\mathbf{x}_i$ and $\mathbf{x}_j$

- Is there any consequence on the **structure** of the Jacobian?

# Derivative of the Error Function

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

  ➡ No, only on $\mathbf{x}_i$ and $\mathbf{x}_j$

- Is there any consequence on the **structure** of the Jacobian?

  ➡ Yes, it will be non-zero only in the rows corresponding to $\mathbf{x}_i$ and $\mathbf{x}_j$

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

$$\mathbf{J}_{ij} = \left( \mathbf{0} \cdots \mathbf{A}_{ij} \cdots \mathbf{B}_{ij} \cdots \mathbf{0} \right)$$

# Jacobians and Sparsity

- Error $\mathbf{e}_{ij}(\mathbf{x})$ depends only on the two parameter blocks $\mathbf{x}_i$ and $\mathbf{x}_j$

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

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

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

# Consequences of the Sparsity

- We need to compute the coefficient vectors and the coefficient matrices:

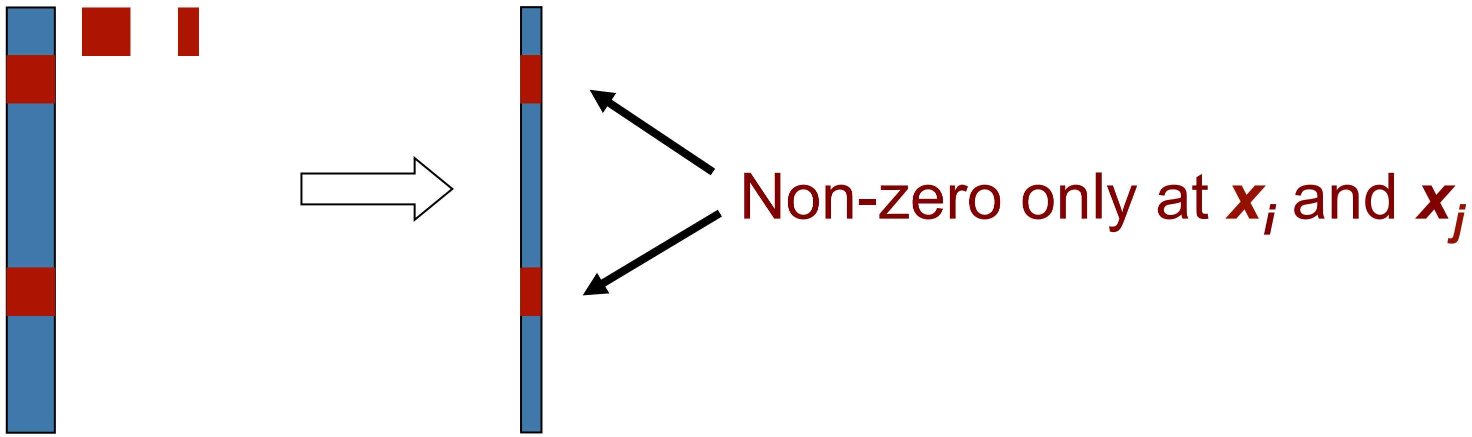
$$\mathbf{b}^T = \sum_{ij} \mathbf{b}_{ij}^T = \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$$

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

- 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

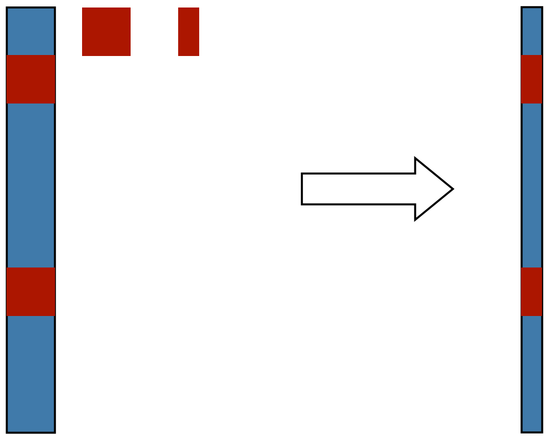
# Illustration of the Structure

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



# Illustration of the Structure

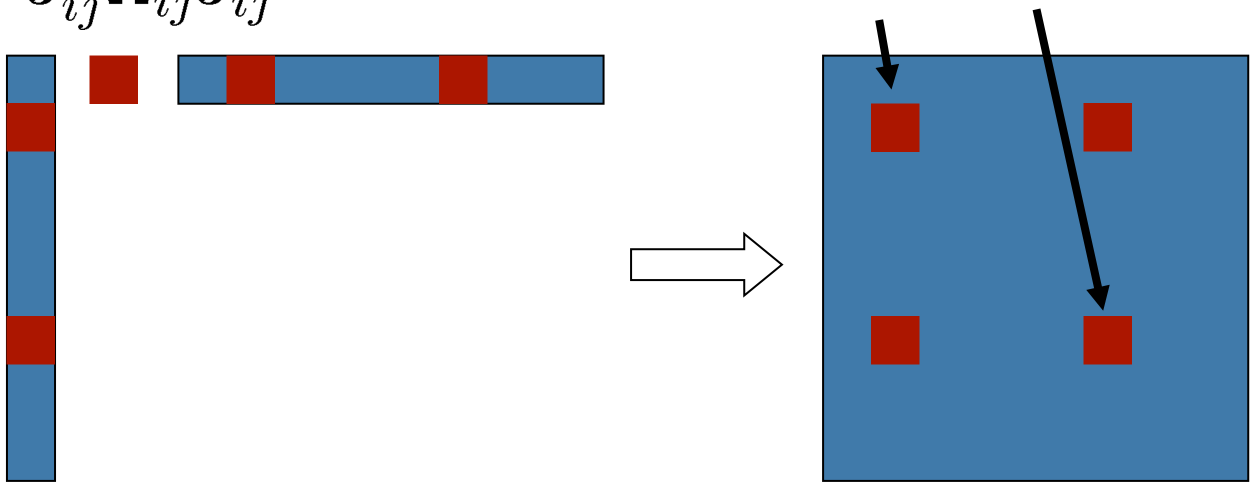
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$$



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

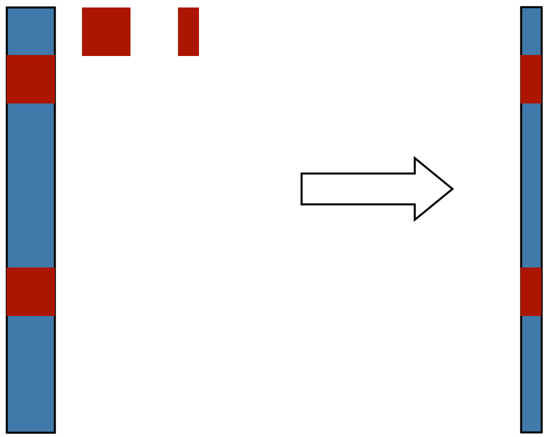
$$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij}$$

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



# Illustration of the Structure

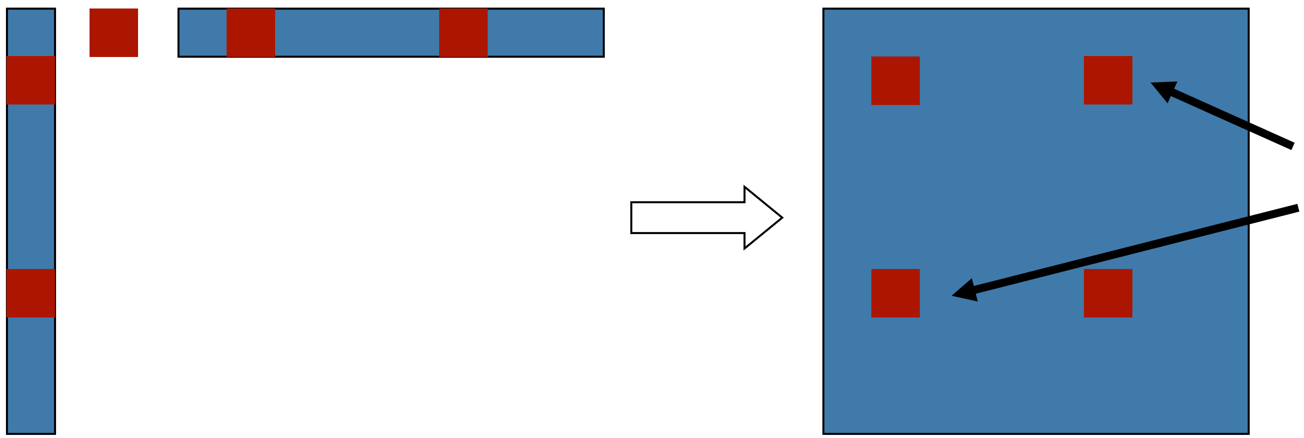
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$$



Non-zero only at $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$

Non-zero on the main diagonal at $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$

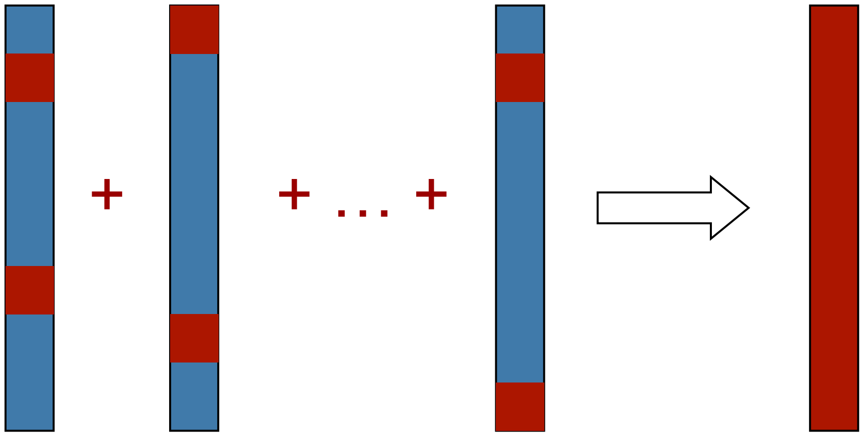
$$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}$$



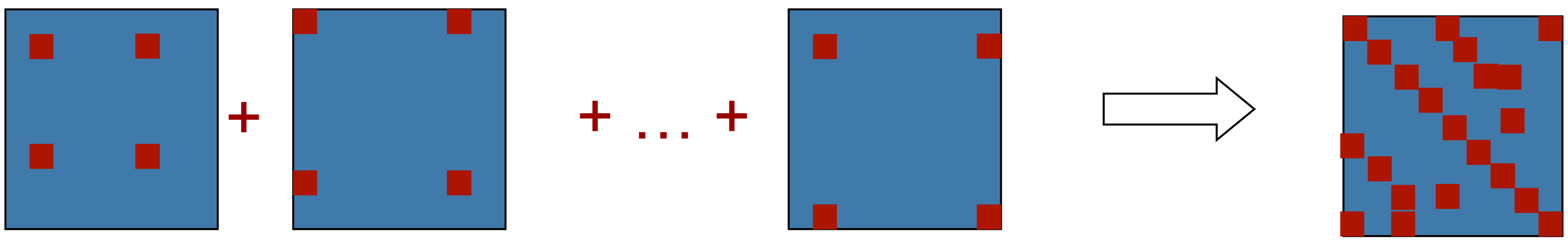
... and at the blocks *ij*,*ji*

# Illustration of the Structure

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



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



# Consequences of the Sparsity

- An edge contributes to the linear system via $\mathbf{b}_{ij}$ and $\mathbf{H}_{ij}$
- The coefficient vector is:

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

- It is non-zero only at the indices corresponding to $\mathbf{x}_i$ and $\mathbf{x}_j$

# 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 ij contributes only to the
  - i$^{th}$ and the j$^{th}$ block of $\mathbf{b}_{ij}$
  - to the blocks ii, jj, ij and ji of $\mathbf{H}_{ij}$
- The resulting system is sparse
- It can be computed by summing up the contribution of each edge
- Efficient solvers can be used
  - Sparse Cholesky decomposition
  - Conjugate gradients
  - … many others

# The Linear System

- Vector of the states increments:

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

- 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

For each constraint:

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

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

- Update the system matrix:

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

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

# Algorithm

**optimize($\mathbf{x}$):**

while (!$converged$)

$(\mathbf{H}, \mathbf{b}) = \text{buildLinearSystem}(\mathbf{x})$

$\boldsymbol{\Delta}\mathbf{x} = \text{solveSparse}(\mathbf{H}\boldsymbol{\Delta}\mathbf{x} = -\mathbf{b})$

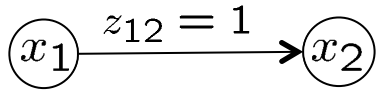
$\mathbf{x} = \mathbf{x} + \boldsymbol{\Delta}\mathbf{x}$

end

*return* $\mathbf{x}$

# Example on the Blackboard

# Trivial 1D Example




- Two nodes and one observation

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

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

# What Went Wrong?

- The constraint 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**

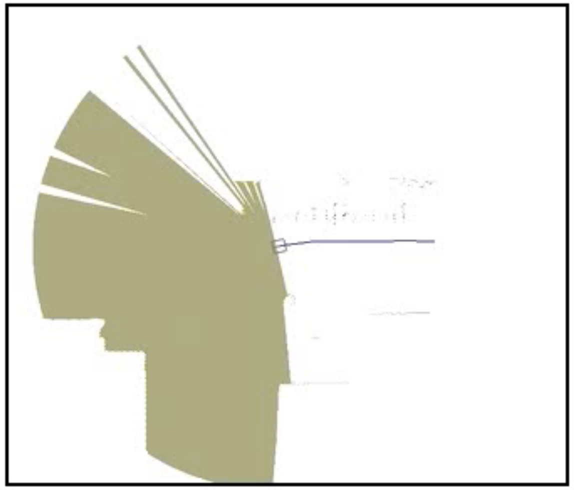
$$\mathbf{H} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}}$$

constraint that sets $\boldsymbol{x_1=0}$

$$\Delta\mathbf{x} = -\mathbf{H}^{-1} b_{12}$$

$$\Delta\mathbf{x} = (0\,1)^T$$

# Real World Examples



# Conclusions

- The back-end part of the SLAM problem can be effectively solved with Gauss-Newton error minimization
- The $\mathbf{H}$ matrix is typically sparse
- This sparsity allows for efficiently solving the linear system
- One of the state-of-the-art solutions for computing maps

# A Note For The Next Exercise

- Consider a 2D graph where each node is parameterized as $\mathbf{x}_i^T = (x_i \; y_i \; \theta_i)$
- Expressed as a transformation $\mathbf{X}_i = \text{v2t}(\mathbf{x}_i)$
- Consider the error function

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

- 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

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

# Literature

## Least Squares SLAM

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