

Robot Mapping

Least Squares Approach to SLAM

Cyrill Stachniss



AiS Autonomous
Intelligent
Systems

Three Main SLAM Paradigms

Kalman
filter

Particle
filter

Graph-
based



**least squares
approach to SLAM**

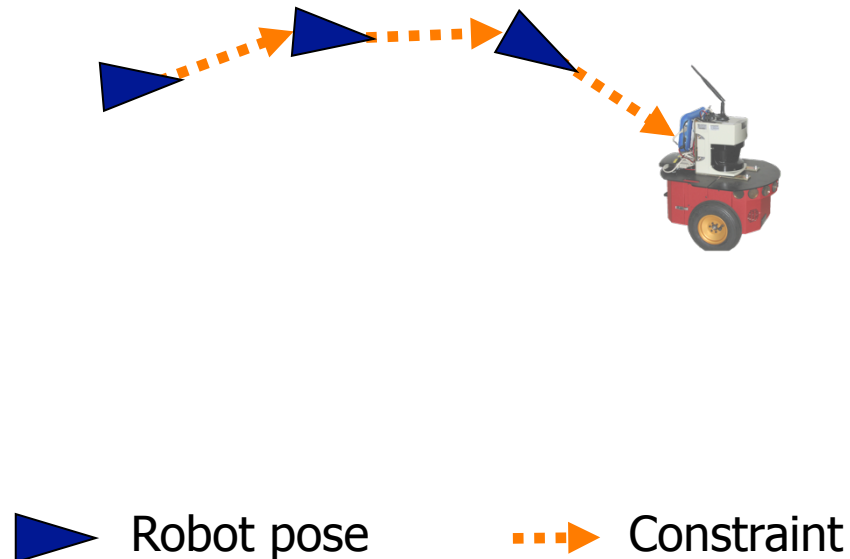
Least Squares in General

- 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

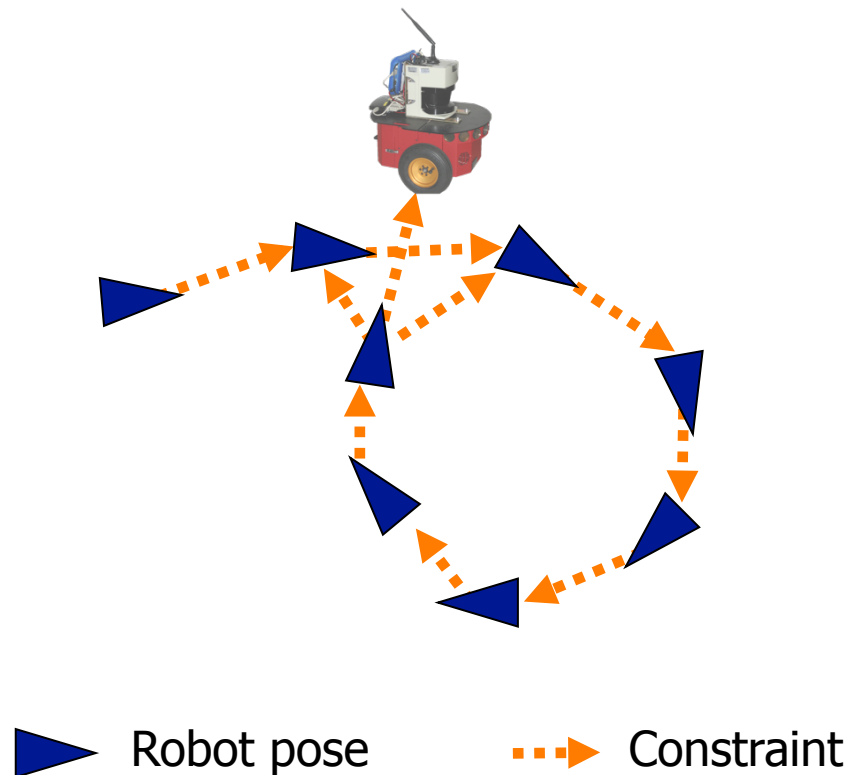
Graph-Based SLAM

- 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

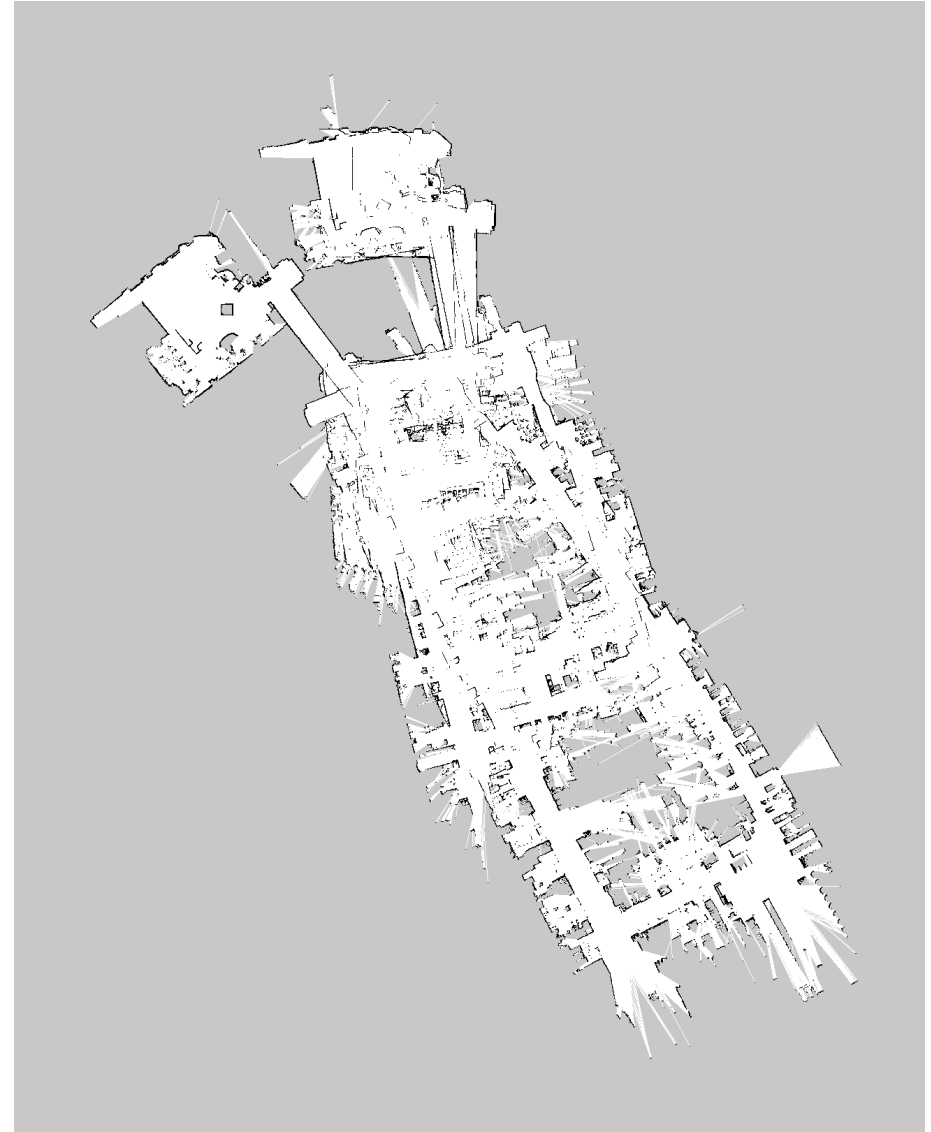


Idea of Graph-Based SLAM

- 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

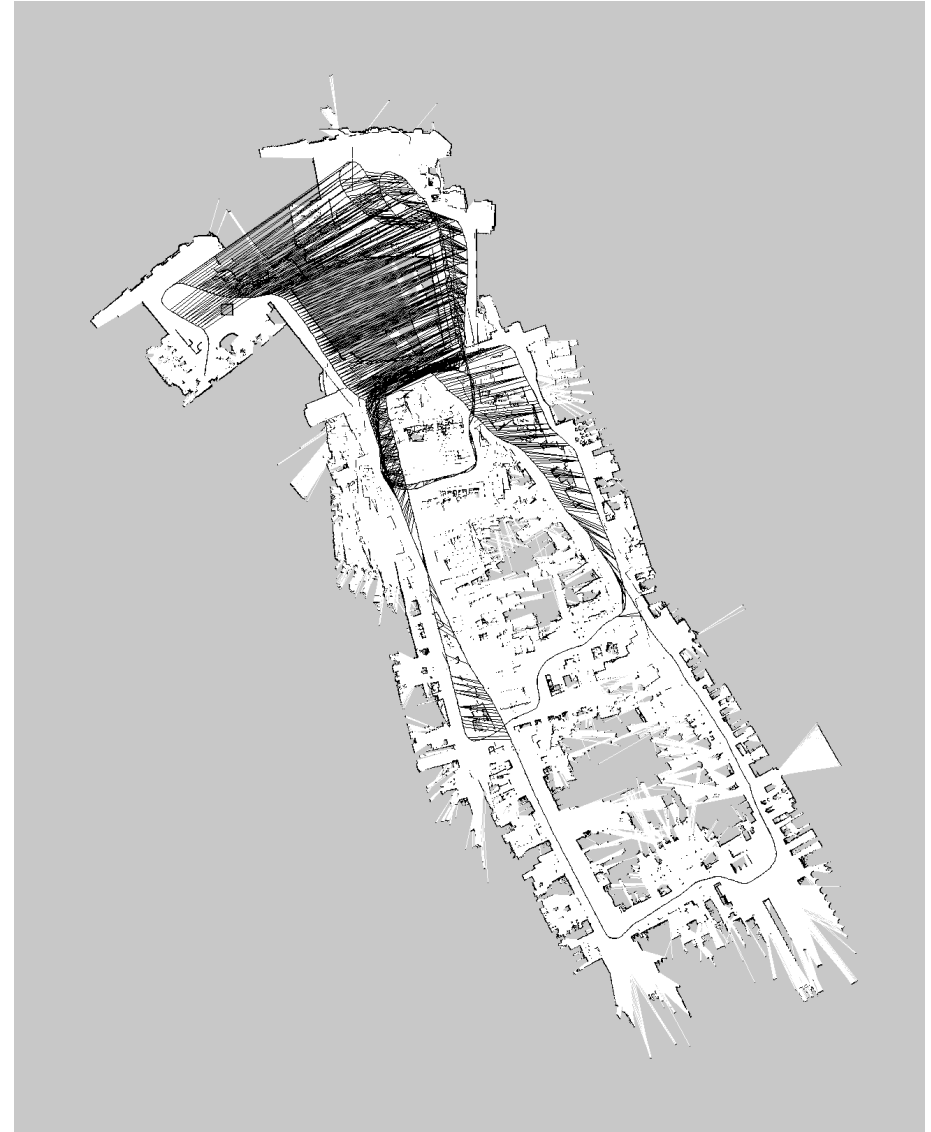
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

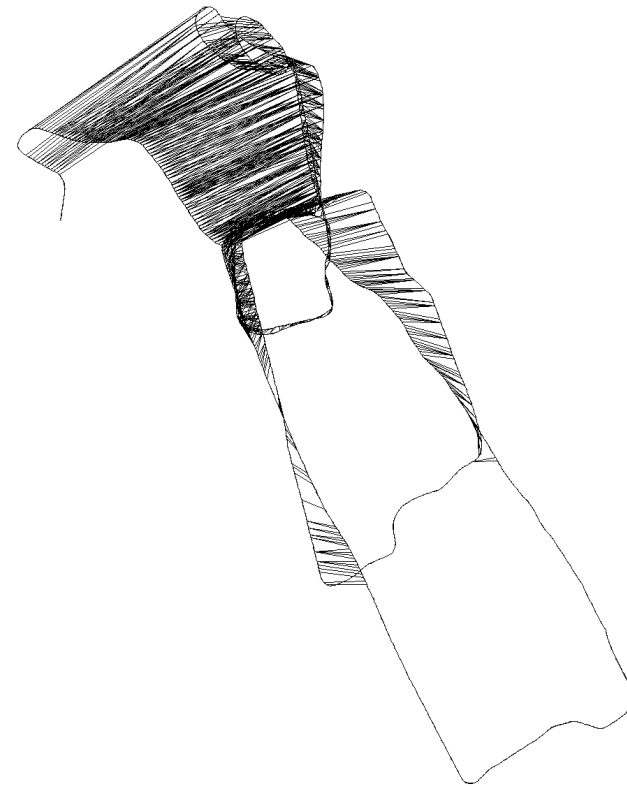
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

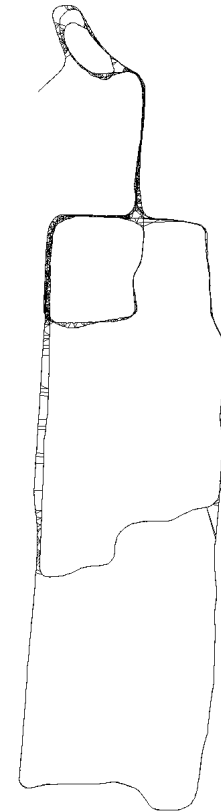
- Once we have the graph, we determine the most likely map by correcting the nodes



Graph-Based SLAM in a Nutshell

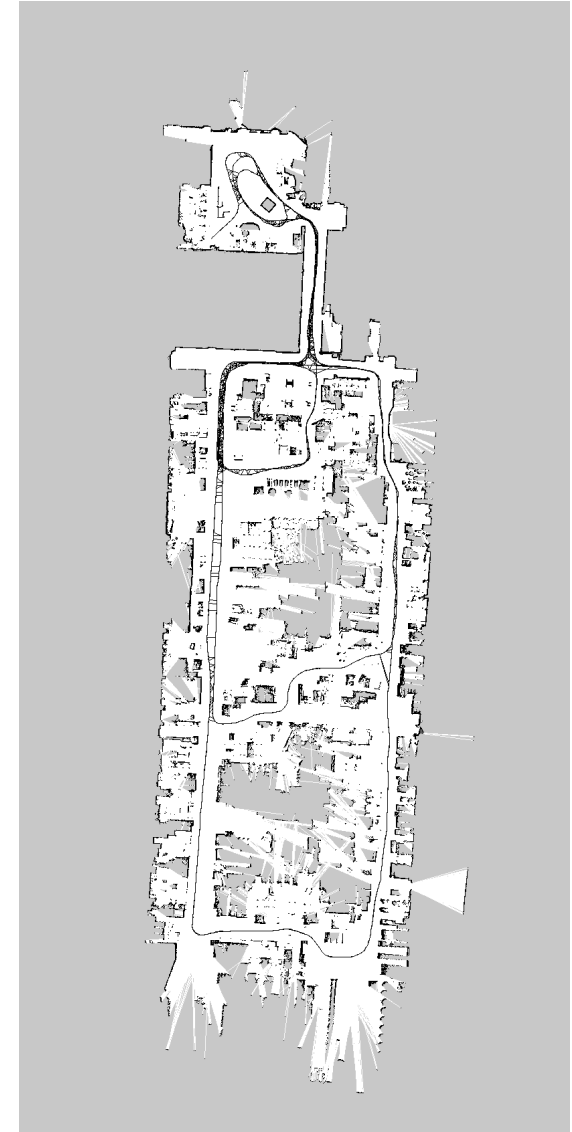
- Once we have the graph, we determine the most likely map by correcting the nodes

... like this



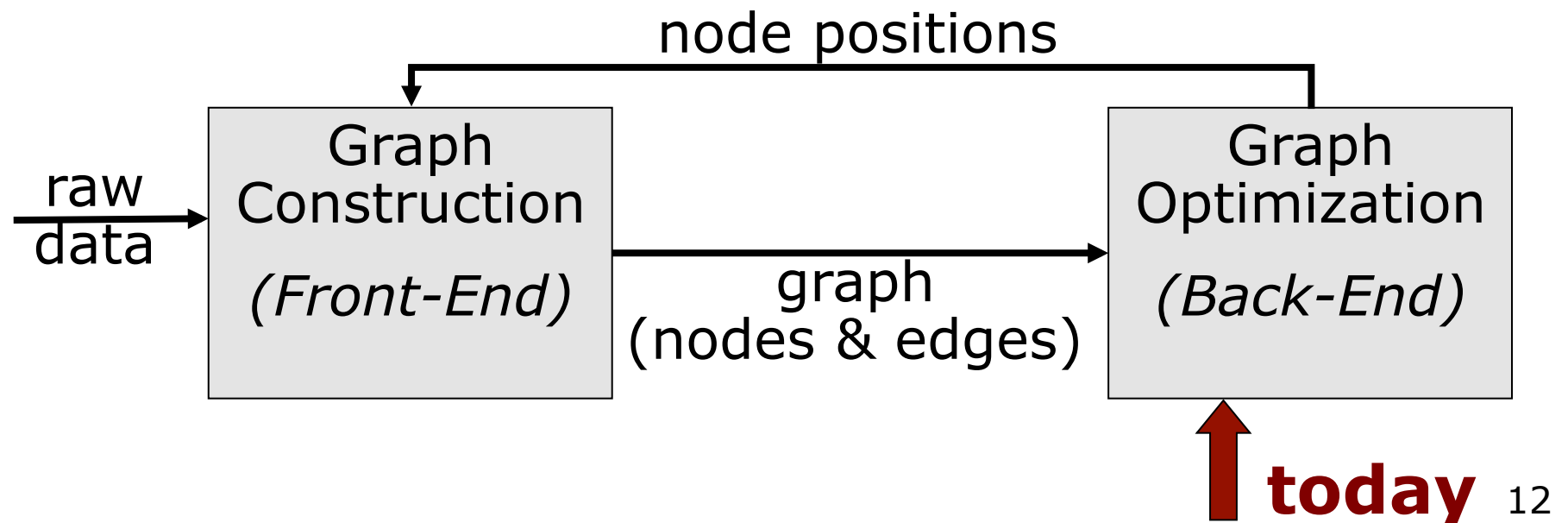
Graph-Based SLAM in a Nutshell

- 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



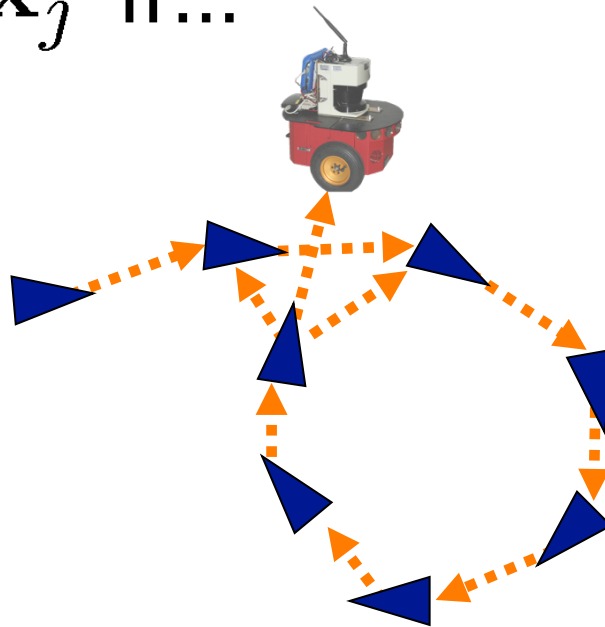
The Overall SLAM System

- Interplay of front-end and back-end
- Map helps to determine constraints by reducing the search space
- Topic today: optimization



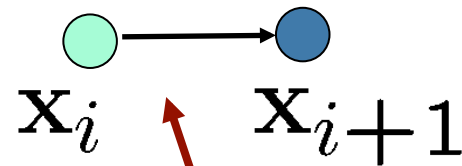
The Graph

- 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)

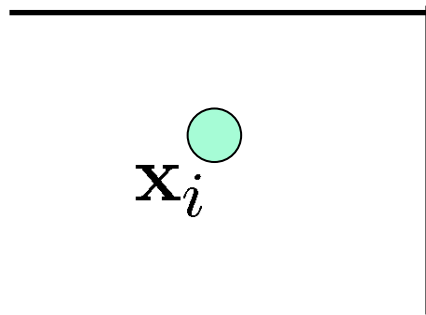
- ...the robot moves from x_i to x_{i+1}
- Edge corresponds to odometry



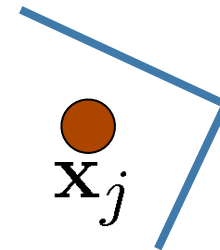
The edge represents the **odometry** measurement

Create an Edge If... (2)

- ...the robot observes the same part of the environment from x_i and from x_j



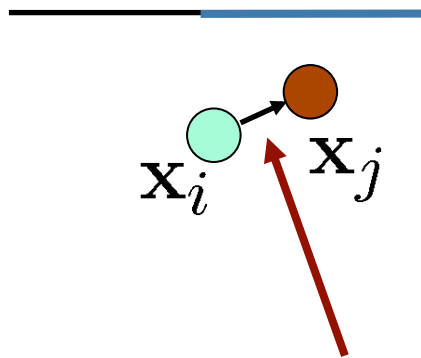
Measurement from x_i



Measurement 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

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

$$(X_i^{-1} X_{i+1})$$

- Observation-Based edge

$$(X_i^{-1} X_j)$$

How node i sees node j

Homogenous Coordinates

- H.C. are a system of coordinates used in projective geometry
- Projective geometry is an alternative algebraic representation of geometric objects and transformations
- Formulas involving H.C. are often simpler than in the Cartesian world
- A single matrix can represent affine transformations and projective transformations

Homogenous Coordinates

- H.C. are a system of coordinates used in projective geometry
- Projective geometry is an alternative algebraic representation of geometric objects and transformations
- Formulas involving H.C. are often simpler than in the Cartesian world
- **A single matrix can represent affine transformations and projective transformations**

Homogenous Coordinates

- N-dim space expressed in N+1 dim
- 4 dim. 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:
- Rotation:

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

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

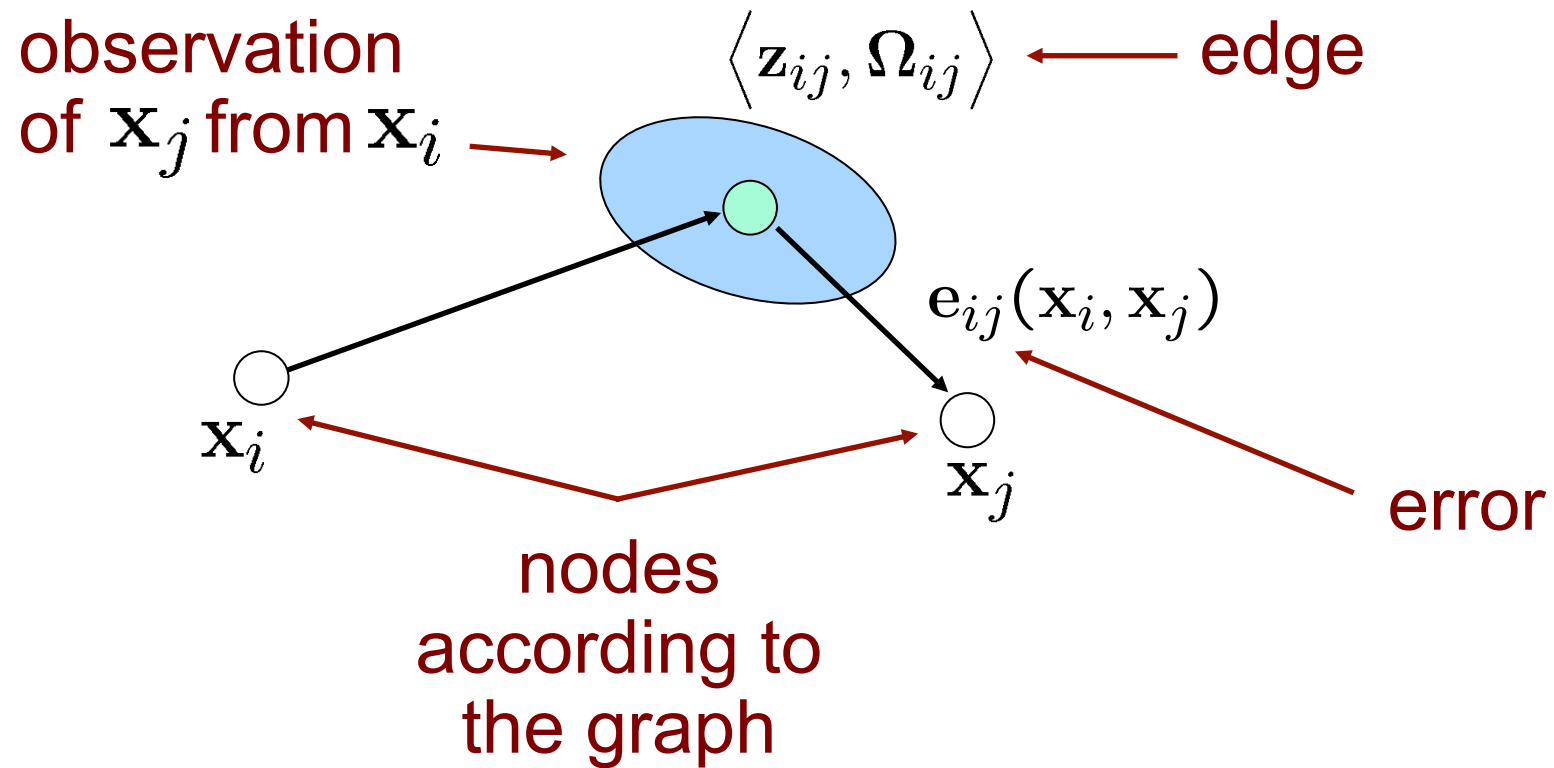
The Edge Information Matrices

- Observations are affected by noise
- Information matrix Ω_{ij} for each edge to encode its uncertainty
- The “bigger” Ω_{ij} , the more the edge “matters” in the optimization

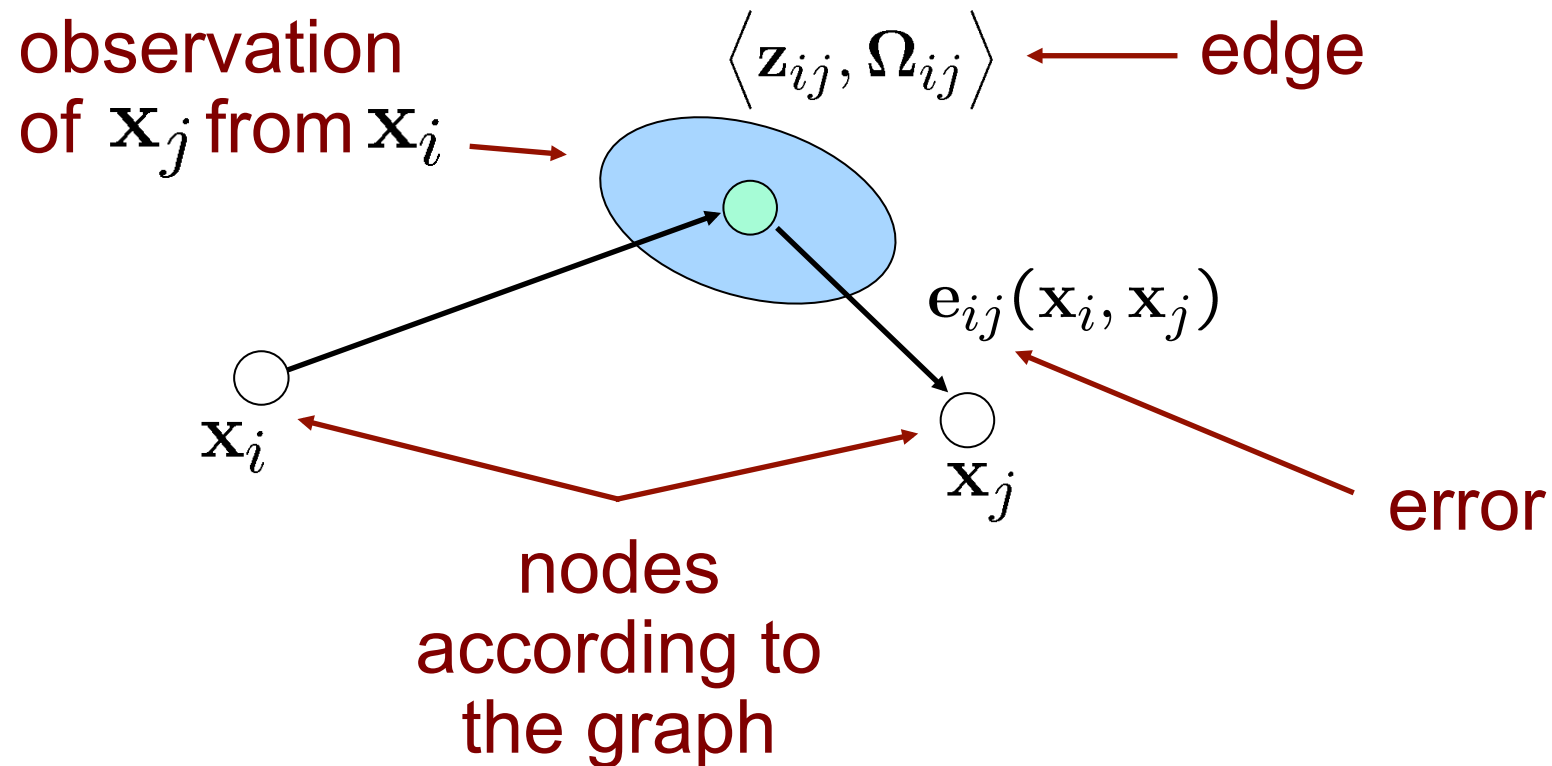
Questions

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

Pose Graph



Pose Graph



▪ **Goal:** $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}$

Least Squares SLAM

- This error function looks suitable for least squares error minimization

$$\begin{aligned}\mathbf{x}^* &= \operatorname{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^T(\mathbf{x}_i, \mathbf{x}_j) \Omega_{ij} \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \\ &= \operatorname{argmin}_{\mathbf{x}} \sum_k \mathbf{e}_k^T(\mathbf{x}) \Omega_k \mathbf{e}_k(\mathbf{x})\end{aligned}$$

Least Squares SLAM

- This error function looks suitable for least squares error minimization

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

Question:

- What is the state vector?

Least Squares SLAM

- This error function looks suitable for least squares error minimization

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

Question:

- What is the state vector?

$$\mathbf{x}^T = \left(\mathbf{x}_1^T \quad \mathbf{x}_2^T \quad \cdots \quad \mathbf{x}_n^T \right)$$

One block for each node of the graph

- Specify the error function!

The Error Function

- Error function for a single constraint

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

measurement

\mathbf{x}_j referenced w.r.t. \mathbf{x}_i

- Error as a function of the whole state vector

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

- 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

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

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

Derivative of the Error Function

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



Derivative of the Error Function

- Does one error term $e_{ij}(\mathbf{x})$ depend on all state variables?
➡ No, only on x_i and x_j

Derivative of the Error Function

- Does one error term $e_{ij}(\mathbf{x})$ 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 $e_{ij}(\mathbf{x})$ 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

$$\frac{\partial e_{ij}(\mathbf{x})}{\partial \mathbf{x}} = \left(0 \dots \frac{\partial e_{ij}(\mathbf{x}_i)}{\partial x_i} \dots \frac{\partial e_{ij}(\mathbf{x}_j)}{\partial x_j} \dots 0 \right)$$

$$\underline{J_{ij}} = \left(0 \dots A_{ij} \dots B_{ij} \dots 0 \right)$$

Jacobians and Sparsity

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

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

- The Jacobian will be zero everywhere except in the columns of \mathbf{x}_i and \mathbf{x}_j

$$\mathbf{J}_{ij} = \begin{pmatrix} \mathbf{0} \dots \mathbf{0} & \underbrace{\frac{\partial e(\mathbf{x}_i)}{\partial \mathbf{x}_i}}_{\mathbf{A}_{ij}} & \mathbf{0} \dots \mathbf{0} & \underbrace{\frac{\partial e(\mathbf{x}_j)}{\partial \mathbf{x}_j}}_{\mathbf{B}_{ij}} & \mathbf{0} \dots \mathbf{0} \end{pmatrix}$$

Consequences of the Sparsity

- We need to compute the coefficient vector \mathbf{b} and matrix \mathbf{H} :

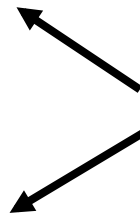
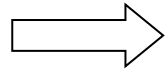
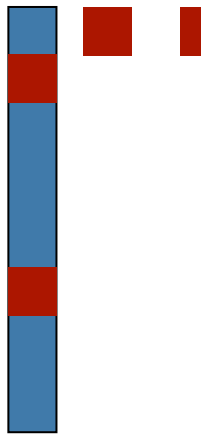
$$\mathbf{b}^T = \sum_{ij} \mathbf{b}_{ij}^T = \sum_{ij} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}$$

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

- 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

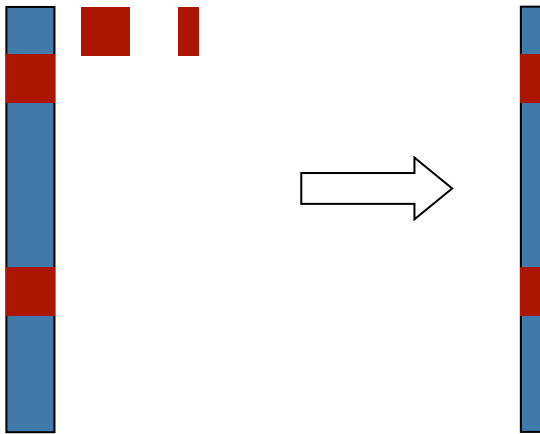
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$$



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

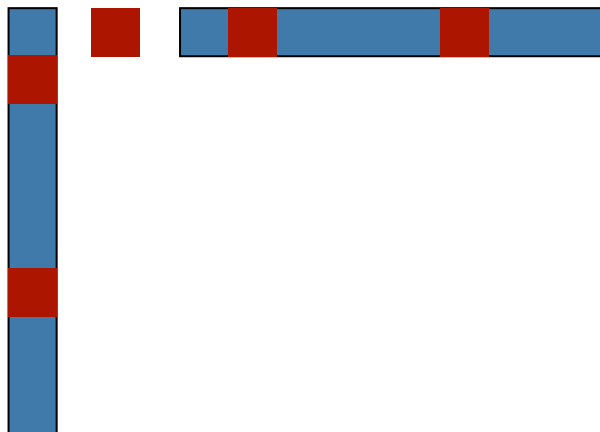
Illustration of the Structure

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



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

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



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

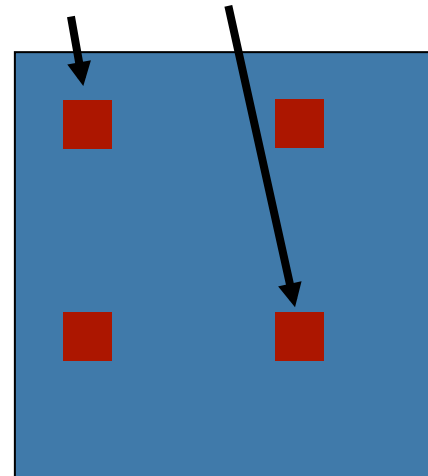
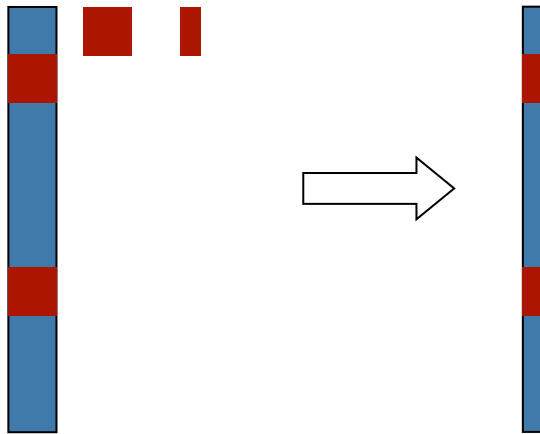


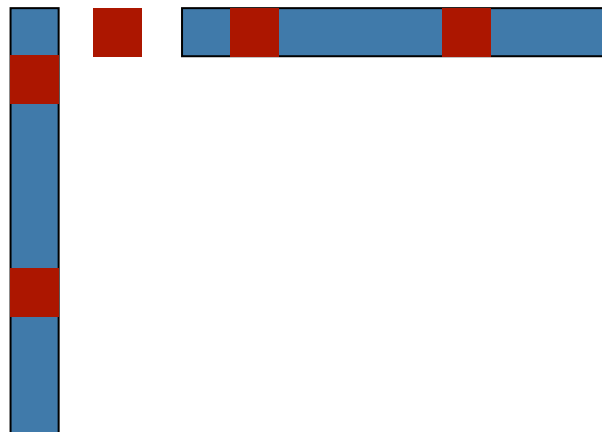
Illustration of the Structure

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



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

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



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

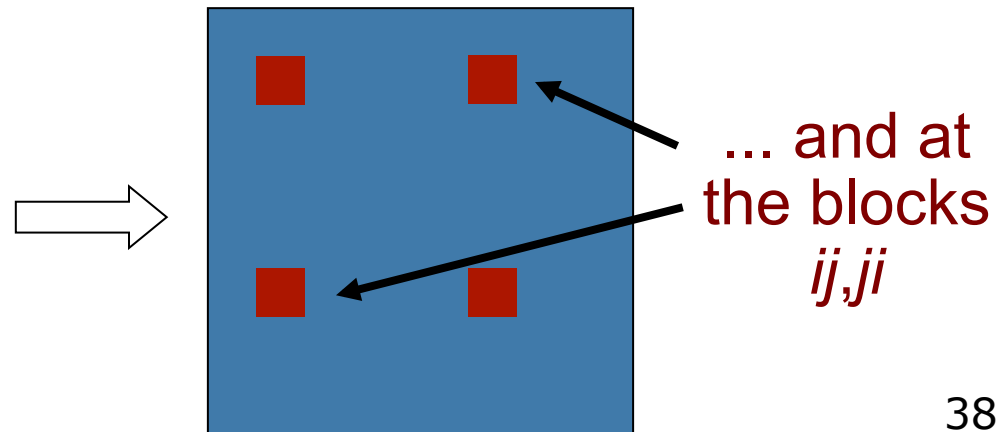
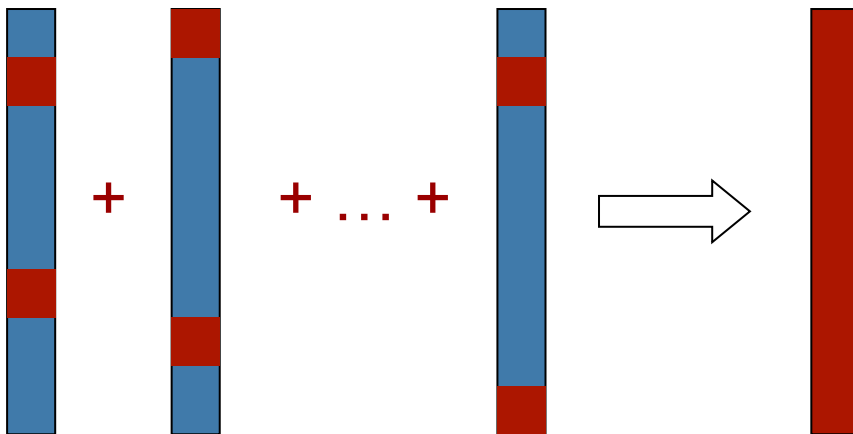
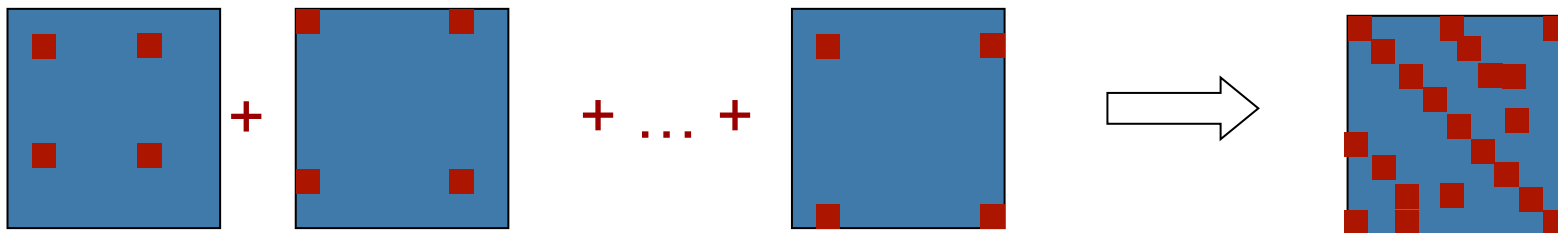


Illustration of the Structure

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



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



Consequences of the Sparsity

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

$$\begin{aligned} b_{ij}^T &= e_{ij}^T \Omega_{ij} J_{ij} \\ &= e_{ij}^T \Omega_{ij} \left(0 \cdots A_{ij} \cdots B_{ij} \cdots 0 \right) \\ &= \left(0 \cdots e_{ij}^T \Omega_{ij} A_{ij} \cdots e_{ij}^T \Omega_{ij} B_{ij} \cdots 0 \right) \end{aligned}$$

- It is non-zero only at the indices corresponding to x_i and x_j

Consequences of the Sparsity

- The coefficient matrix of an edge is:

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

- 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 \mathbf{b}_{ij}
 - to the blocks ii , jj , ij and ji of \mathbf{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

The Linear System

- Vector of the states increments:

$$\Delta \mathbf{x}^T = \left(\Delta \mathbf{x}_1^T \quad \Delta \mathbf{x}_2^T \quad \dots \quad \Delta \mathbf{x}_n^T \right)$$

- Coefficient vector:

$$\mathbf{b}^T = \left(\bar{\mathbf{b}}_1^T \quad \bar{\mathbf{b}}_2^T \quad \dots \quad \bar{\mathbf{b}}_n^T \right)$$

- System matrix:

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

Building the Linear System

For each constraint:

- Compute error $e_{ij} = \text{t2v}(\mathbf{Z}_{ij}^{-1}(\mathbf{X}_i^{-1}\mathbf{X}_j))$
- Compute the blocks of the Jacobian:

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

- Update the coefficient vector:

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

- Update the system matrix:

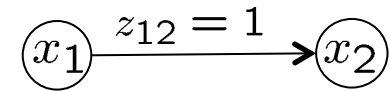
$$\begin{aligned} \bar{\mathbf{H}}^{ii} + &= \mathbf{A}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{A}_{ij} & \bar{\mathbf{H}}^{ij} + &= \mathbf{A}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} \\ \bar{\mathbf{H}}^{ji} + &= \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{A}_{ij} & \bar{\mathbf{H}}^{jj} + &= \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} \end{aligned}$$

Algorithm

```
1:  optimize(x):  
2:      while (!converged)  
3:          (H, b) = buildLinearSystem(x)  
4:          Δx = solveSparse(HΔx = -b)  
5:          x = x + Δx  
6:      end  
7:      return x
```

Example on the Blackboard

Trivial 1D Example



- Two nodes and one observation

$$\mathbf{x} = (x_1 \ x_2)^T = (0 \ 0)$$

$$z_{12} = 1$$

$$\Omega = 2$$

$$e_{12} = z_{12} - (x_2 - x_1) = 1 - (0 - 0) = 1$$

$$\mathbf{J}_{12} = (1 \ -1)$$

$$\mathbf{b}_{12}^T = \mathbf{e}_{12}^T \Omega \mathbf{J}_{12} = (2 \ -2)$$

$$\mathbf{H}_{12} = \mathbf{J}_{12}^T \Omega \mathbf{J}_{12} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix}$$

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

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 as their relative coordinates fit
- **One node needs to be “fixed”**

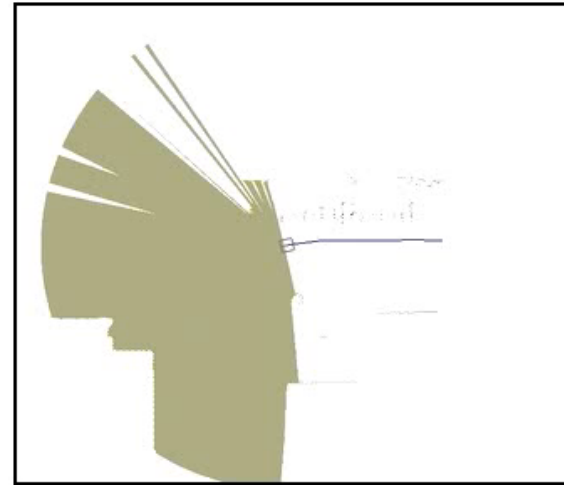
$$\mathbf{H} = \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} + \boxed{\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}} \quad \text{constraint that sets } \mathbf{dx}_1 = \mathbf{0}$$
$$\Delta \mathbf{x} = -\mathbf{H}^{-1} b_{12}$$
$$\Delta \mathbf{x} = (0 \ 1)^T$$

Role of the Prior

- We saw that the matrix \mathbf{H} has not full rank (after adding the constraints)
- The global frame had not been fixed
- Fixing the global reference frame is strongly related to the prior $p(\mathbf{x}_0)$
- A Gaussian estimate about \mathbf{x}_0 results in an additional constraint
- E.g., first pose in the origin:

$$e(\mathbf{x}_0) = \mathbf{t2v}(\mathbf{X}_0)$$

Real World Examples



Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori
- We may want to optimize all others and keep these fixed
- How?

Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori
- We may want to optimize all others and keep these fixed
- How?
- If a variable is not optimized, it should “disappears” from the linear system

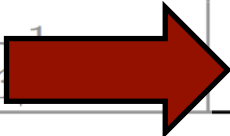
Fixing a Subset of Variables

- Assume that the value of certain variables during the optimization is known a priori
- We may want to optimize all others and keep these fixed
- How?
- If a variable is not optimized, it should “disappears” from the linear system
- Construct the full system
- Suppress the rows and the columns corresponding to the variables to fix

Why Can We Simply Suppress the Rows and Columns of the Corresponding Variables?

$$p(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_\alpha \\ \boldsymbol{\mu}_\beta \end{bmatrix}, \begin{bmatrix} \Sigma_{\alpha\alpha} & \Sigma_{\alpha\beta} \\ \Sigma_{\beta\alpha} & \Sigma_{\beta\beta} \end{bmatrix}\right) = \mathcal{N}^{-1}\left(\begin{bmatrix} \boldsymbol{\eta}_\alpha \\ \boldsymbol{\eta}_\beta \end{bmatrix}, \begin{bmatrix} \Lambda_{\alpha\alpha} & \Lambda_{\alpha\beta} \\ \Lambda_{\beta\alpha} & \Lambda_{\beta\beta} \end{bmatrix}\right)$$

	MARGINALIZATION	CONDITIONING
	$p(\boldsymbol{\alpha}) = \int p(\boldsymbol{\alpha}, \boldsymbol{\beta}) d\boldsymbol{\beta}$	$p(\boldsymbol{\alpha} \boldsymbol{\beta}) = p(\boldsymbol{\alpha}, \boldsymbol{\beta}) / p(\boldsymbol{\beta})$
COV. FORM	$\boldsymbol{\mu} = \boldsymbol{\mu}_\alpha$ $\Sigma = \Sigma_{\alpha\alpha}$	$\boldsymbol{\mu}' = \boldsymbol{\mu}_\alpha + \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_\beta)$ $\Sigma' = \Sigma_{\alpha\alpha} - \Sigma_{\alpha\beta} \Sigma_{\beta\beta}^{-1} \Sigma_{\beta\alpha}$
INFO. FORM	$\boldsymbol{\eta} = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} \boldsymbol{\eta}_\beta$ $\Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta} \Lambda_{\beta\beta}^{-1} \Lambda_{\beta\alpha}$	$\boldsymbol{\eta}' = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta} \boldsymbol{\beta}$ $\Lambda' = \Lambda_{\alpha\alpha}$



Uncertainty

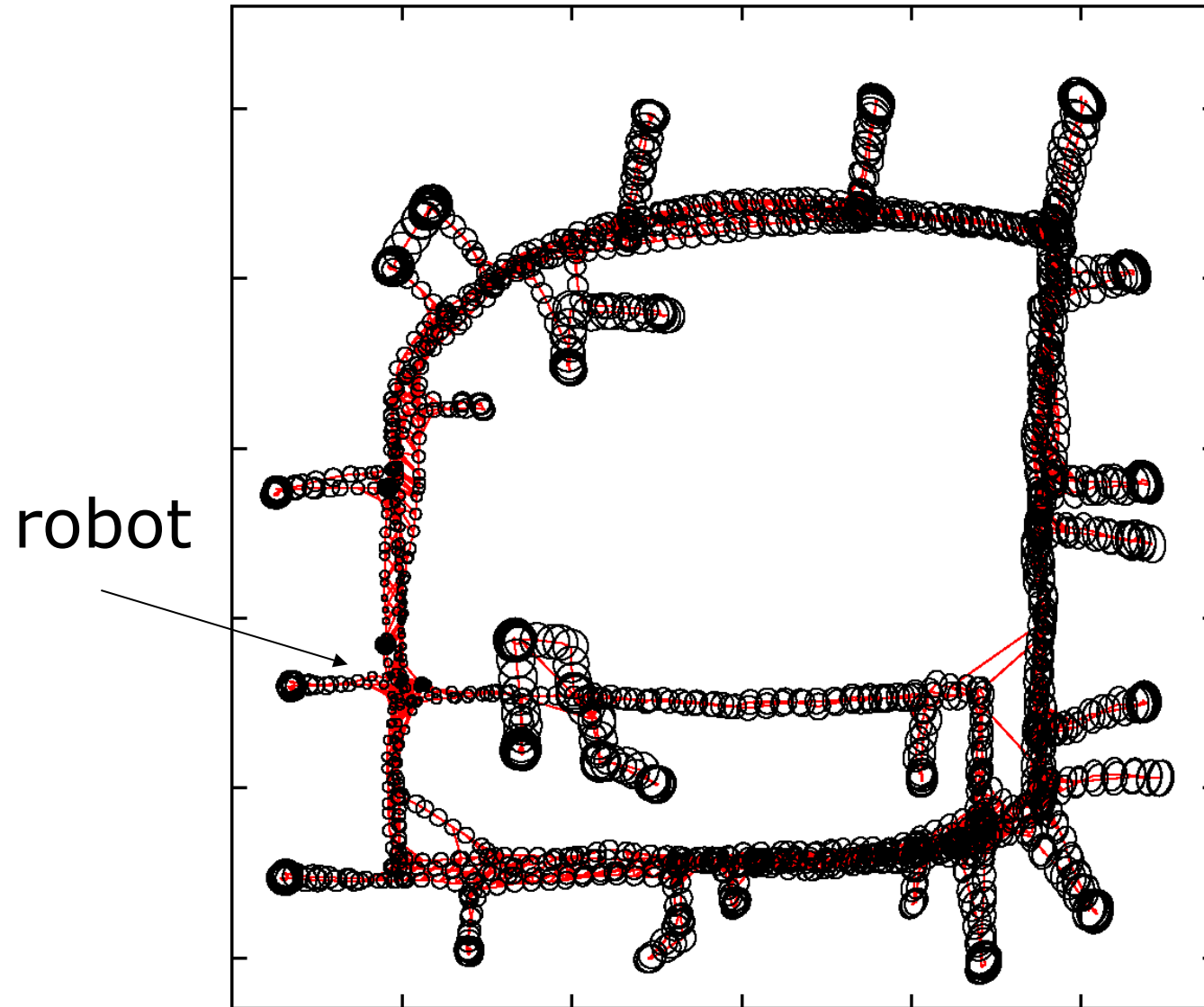
- \mathbf{H} represents the information matrix given the linearization point
- Inverting \mathbf{H} gives the (dense) covariance matrix
- The diagonal blocks of the covariance matrix represent the (absolute) uncertainties of the corresponding variables

Relative Uncertainty

To determine the relative uncertainty between x_i and x_j :

- Construct the full matrix H
- Suppress the rows and the columns of x_i (= do not optimize/fix this variable)
- Compute the block j,j of the inverse
- This block will contain the covariance matrix of x_j w.r.t. x_i , which has been fixed

Example



Conclusions

- The back-end part of the SLAM problem can be effectively solved with Gauss-Newton
- 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

Literature

Least Squares SLAM

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