Photogrammetry & Robotics Lab

Graph-Based SLAM
A Least Squares Approach to SLAM using Pose Graphs

Cyrill Stachniss

Traditional SLAM Paradigms

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

least squares approach to SLAM

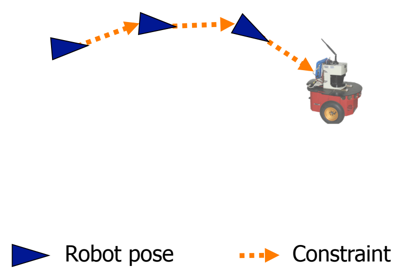
Least Squares in General

- 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

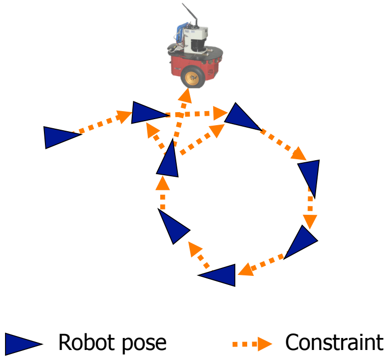
Graph-Based SLAM

- 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

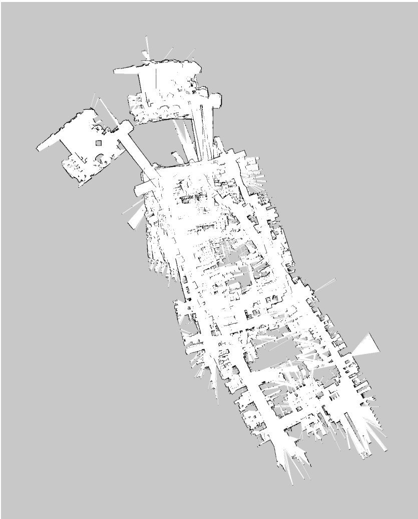


Idea of Graph-Based SLAM

- 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

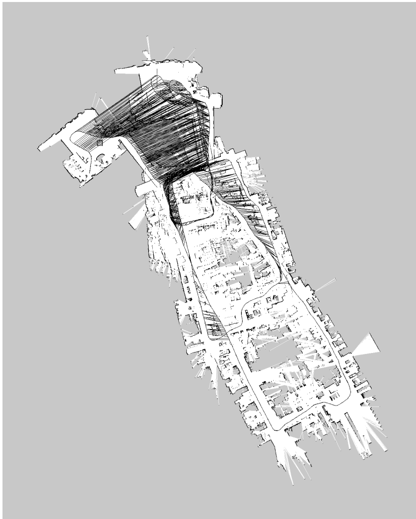
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

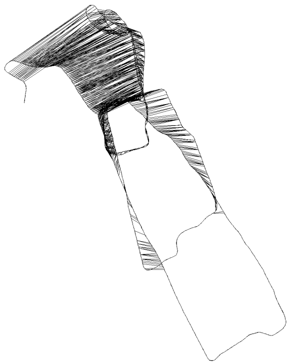
- Every node in the graph corresponds to a robot position and a laser measurement
- An edge between two nodes represents a spatial constraint between the nodes



KUKA Halle 22, courtesy of P. Pfaff

Graph-Based SLAM in a Nutshell

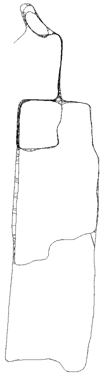
- Once we have the graph, we determine the most likely map by correcting the nodes



Graph-Based SLAM in a Nutshell

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

  … like this

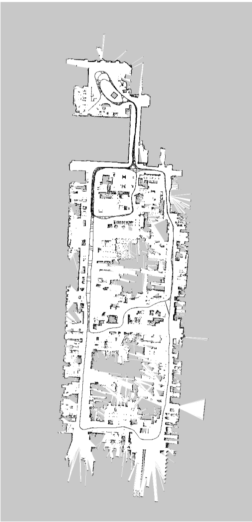


Graph-Based SLAM in a Nutshell

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

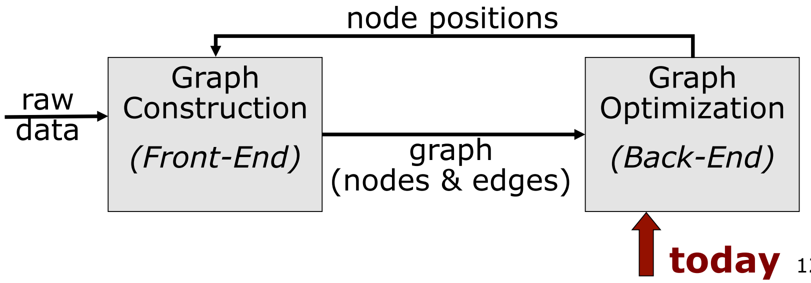
  … like this

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



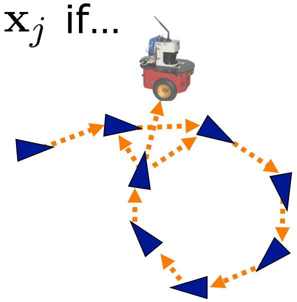
The Overall SLAM System

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



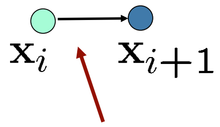
The Graph

- It consists of n nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each $\mathbf{x}_i$ is a 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)

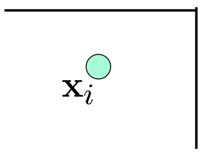
- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



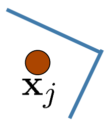
The edge represents the **odometry** measurement

Create an Edge If… (2)

- …the robot observes the same part of the environment from $\mathbf{x}_i$ and from $\mathbf{x}_j$



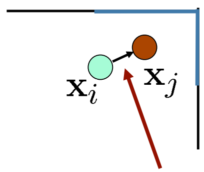
Measurement from $\mathbf{x}_i$



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

- H.C. are a system of coordinates used in projective geometry
- Projective geometry is an alternative representation of geometric objects and transformations
- **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:

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

- Rotation:

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

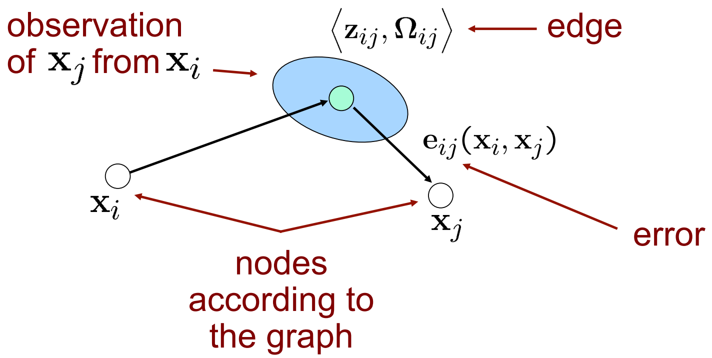
The Edge Information Matrices

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

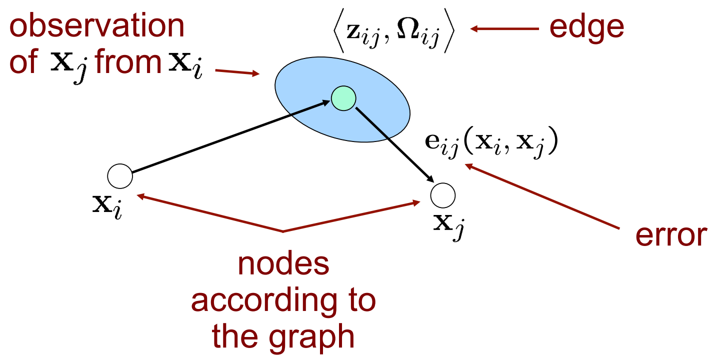
Questions

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

Pose 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

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

Least Squares SLAM

- This error function looks suitable for least squares error minimization

$$\mathbf{x}^* = \underset{\mathbf{x}}{\text{argmin}} \sum_{k} \mathbf{e}_{k}^T(\mathbf{x}) \mathbf{\Omega}_{k} \mathbf{e}_{k}(\mathbf{x})$$

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?

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

One vector for each node of the graph

- Specify the error function!

The Error Function

- Error function for a single constraint

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \text{t2v}(\underbrace{\mathbf{Z}_{ij}^{-1}}_{\text{measurement}}\underbrace{(\mathbf{X}_i^{-1}\mathbf{X}_j)}_{\mathbf{x}_j \text{ referenced w.r.t. } \mathbf{x}_i})$$

- 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

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

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

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 $\mathbf{x}_j$

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

Jacobians and Sparsity

- 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 zero everywhere except 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 vector $\mathbf{b}$ and matrix $\mathbf{H}$:

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

- 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

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

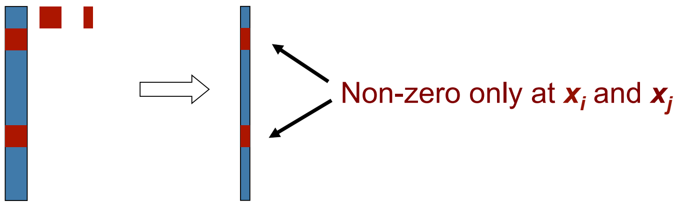
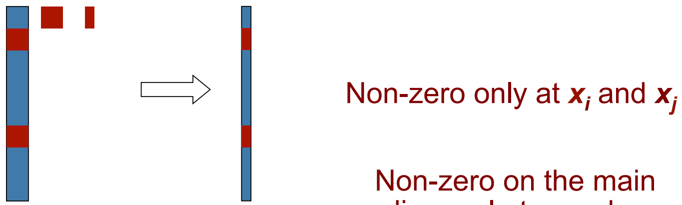


Illustration of the Structure

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



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

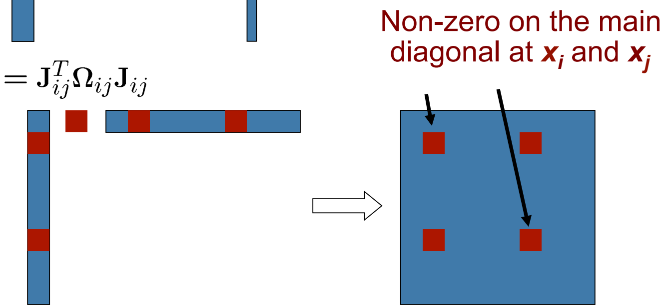
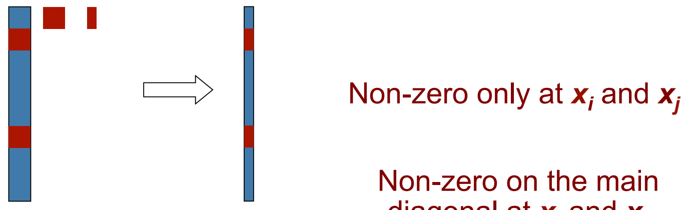


Illustration of the Structure

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



Non-zero on the main diagonal at x_i and x_j

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

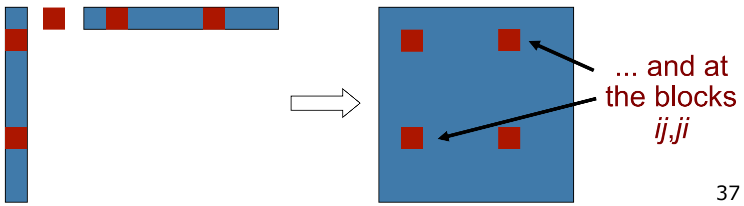
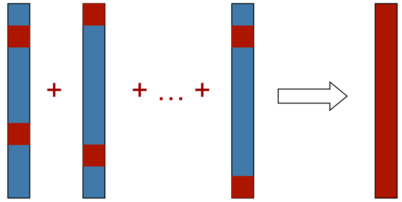
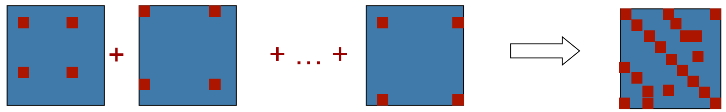


Illustration of the Structure

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



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



Consequences of the Sparsity

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

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

- 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:

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

- 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:

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

- Coefficient vector:

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

- Normal equation matrix:

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

Building the Linear System

For each constraint:

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

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

- Update the normal equation matrix:

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

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

Algorithm

optimize($\mathbf{x}$):

  while (!*converged*)
    $(\mathbf{H}, \mathbf{b}) = \mathrm{buildLinearSystem}(\mathbf{x})$
    $\Delta\mathbf{x} = \mathrm{solveSparse}(\mathbf{H}\Delta\mathbf{x} = -\mathbf{b})$
    $\mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$
  end
  return $\mathbf{x}$

Example on the Blackboard

Trivial 1D Example

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

- Two nodes and one observation

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

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"**

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

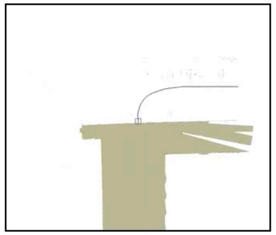
constraint that sets ***dx_1=0***

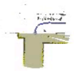
Role of the Prior

- 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:

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

Real World Example





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 disappear 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 disappear 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 $p(\boldsymbol{\alpha}) = \int p(\boldsymbol{\alpha}, \boldsymbol{\beta})\, d\boldsymbol{\beta}$	CONDITIONING $p(\boldsymbol{\alpha} \mid \boldsymbol{\beta}) = p(\boldsymbol{\alpha}, \boldsymbol{\beta}) / p(\boldsymbol{\beta})$
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	$\boldsymbol{\eta} = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\boldsymbol{\eta}_\beta$ $\Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}$	$\boldsymbol{\eta}' = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\boldsymbol{\beta}$ $\Lambda' = \Lambda_{\alpha\alpha}$

Courtesy: R. Eustice

Uncertainty

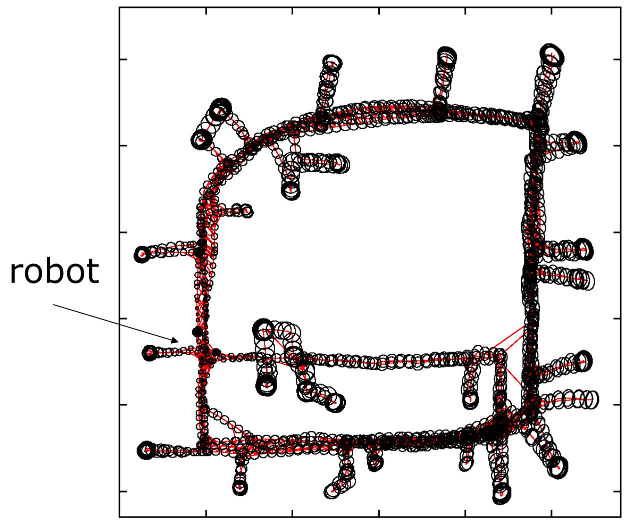
- $\mathbf{H}$ represents the information matrix given the linearization point
- Inverting $\mathbf{H}$ gives the (dense) covariance matrix
- The diagonal blocks of the covariance matrix represent the uncertainties of the corresponding variables

Relative Uncertainty

To determine the relative uncertainty between $\mathbf{x}_i$ and $\mathbf{x}_j$:

- Construct the full matrix $\mathbf{H}$
- Suppress the rows and the columns of $\mathbf{x}_i$ (= do not optimize/fix this variable)
- Compute the block *j,j* of the inverse
- This block will contain the covariance matrix of $\mathbf{x}_j$ w.r.t. $\mathbf{x}_i$, 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

Slide Information

- These slides have been created by Cyrill Stachniss as part of the robot mapping course taught in 2012/13 and 2013/14. I created this set of slides partially extending existing material of Giorgio Grisetti and myself.
- I tried to acknowledge all people that contributed image or video material. In case I missed something, please let me know. If you adapt this course material, please make sure you keep the acknowledgements.
- Feel free to use and change the slides. If you use them, I would appreciate an acknowledgement as well. To satisfy my own curiosity, I appreciate a short email notice in case you use the material in your course.
- My video recordings are available through YouTube: http://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_&feature=g-list

Cyrill Stachniss, 2014
cyrill.stachniss@igg.uni-bonn.de