ICRA 2016 Tutorial on SLAM

Graph-Based SLAM and Sparsity

Cyrill Stachniss

igg

universitätbonn

Graph-Based SLAM ??

Graph-Based $\boxed{\text{SLAM}}$??

SLAM = simultaneous localization and mapping

$\boxed{\text{Graph}}$-Based SLAM ??

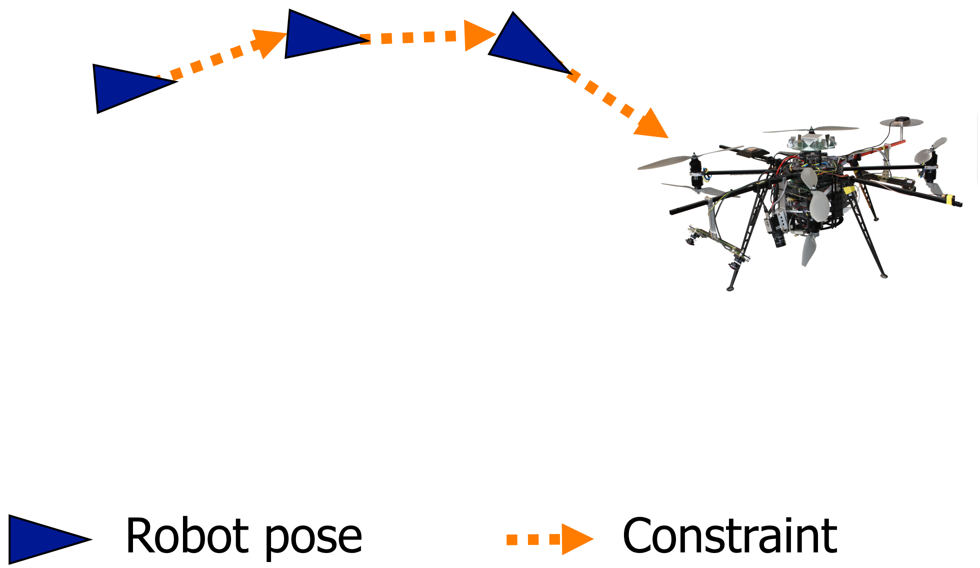
SLAM = simultaneous localization and mapping

graph = representation of a set of objects where pairs of objects are connected by links encoding relations between the objects

What is my goal for today?

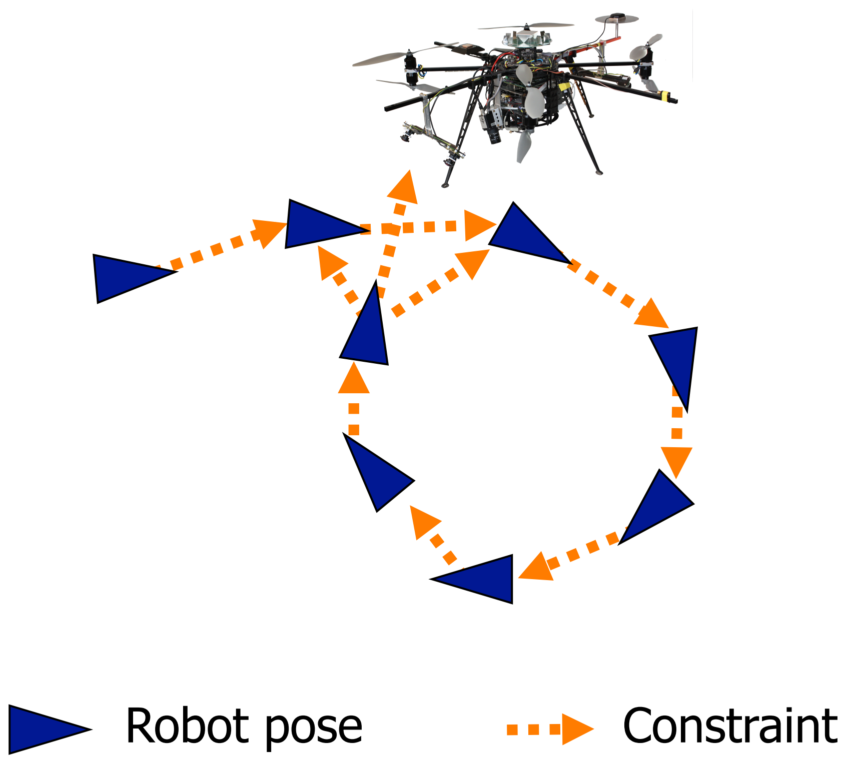
Graph-Based SLAM

- Nodes represent poses or locations
- 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-SLAM and Least Squares

- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive
- We have **real observations** relating the nodes with each other

Graph-SLAM and Least Squares

- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive
- We have **real observations** relating the nodes with each other

Giorgio's lecture

Find a configuration of the nodes so that the real and predicted observations are as similar as possible

Error Function

Giorgio's lecture

$$\mathbf{x}$$

$$h_1(\mathbf{x}) = \hat{\mathbf{z}}_1 \qquad \mathbf{z}_1$$

$$h_2(\mathbf{x}) = \hat{\mathbf{z}}_2 \qquad \mathbf{z}_2$$

minimize the differences!

$$h_n(\mathbf{x}) = \hat{\mathbf{z}}_n \qquad \mathbf{z}_n$$

state (unknown)	predicted measurements	real measurements

$$e_i(\mathbf{x}) = \mathbf{e}_i(\mathbf{x})^T \boldsymbol{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

$$\mathbf{e}_i(\mathbf{x}) = \mathbf{z}_i - h_i(\mathbf{x})$$

Giorgio's lecture

Procedure in Brief

Iterate the following steps:

- Linearize around $\mathbf{x}$ and compute for each measurement

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

- Compute the terms for the linear system $\mathbf{b} = \sum_i \mathbf{J}_i^T \mathbf{\Omega}_i \mathbf{e}_i \quad \mathbf{H} = \sum_i \mathbf{J}_i^T \mathbf{\Omega}_i \mathbf{J}_i$
- Solve the linear system

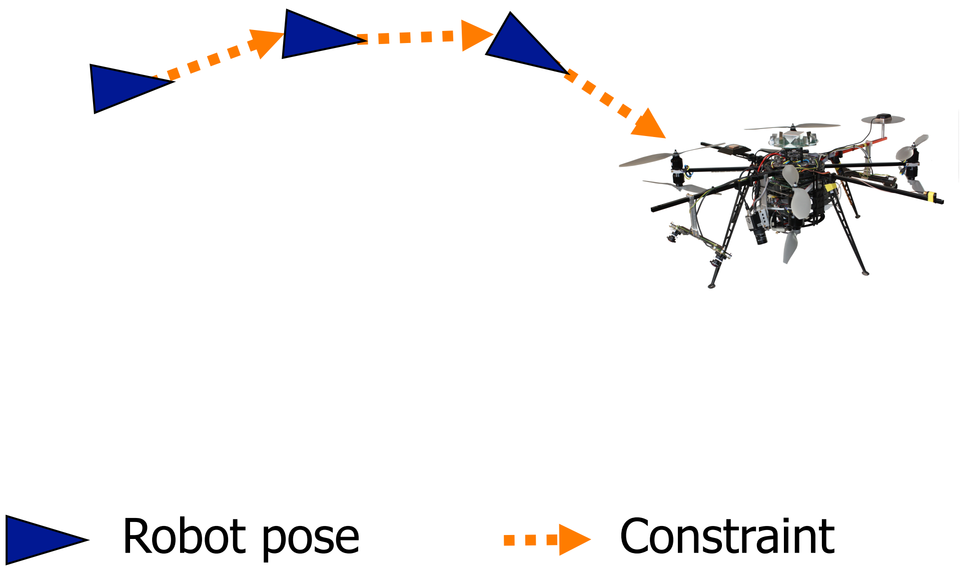
$$\mathbf{\Delta x}^* \;=\; -\mathbf{H}^{-1}\mathbf{b}$$

- Updating state $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{\Delta x}^*$

Let’s use that for SLAM

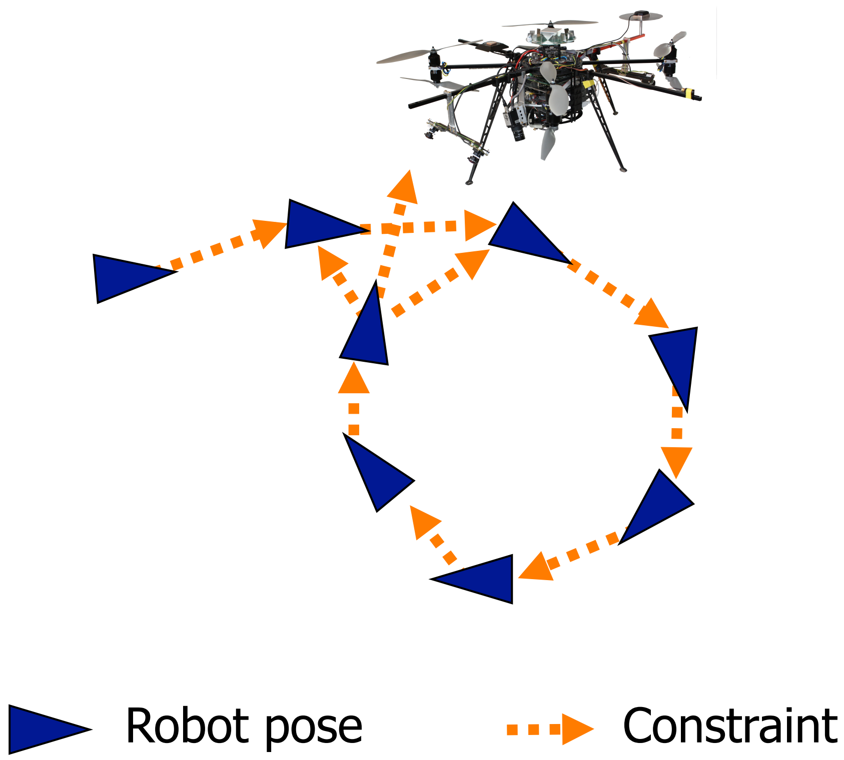
Pose-Graph-Based SLAM

- Nodes represent poses or locations
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain



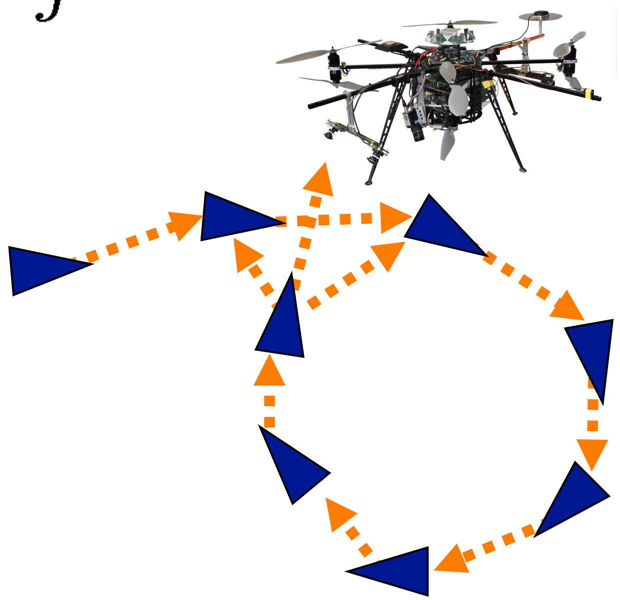
Pose-Graph-Based SLAM

- Observing previously seen areas generates constraints between non-successive poses



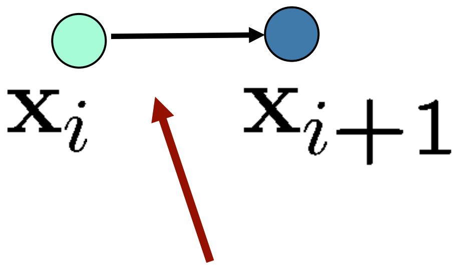
The Pose-Graph

- It consists of n nodes $\mathbf{x} = \mathbf{x}_{1:n}$
- Each $\mathbf{x}_i$ is a 2D or 3D **pose** (position and orientation 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)

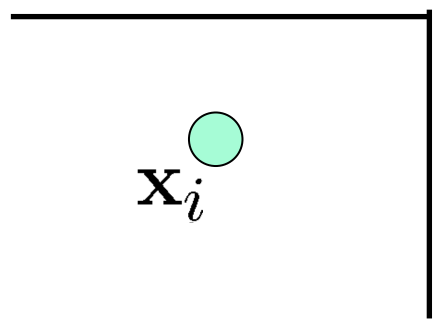
- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



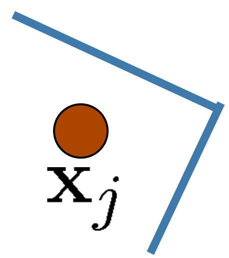
The edge represents the **odometry** measurement

Create an Edge If... (2)

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



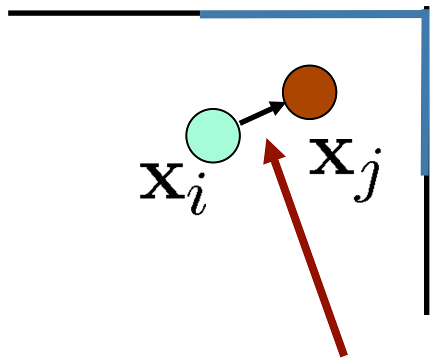
Measurement from $\mathbf{x}_i$



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

- **How to express $\mathbf{x}_j$ relative to $\mathbf{x}_i$?**
- Express this through transformations
- Let $\mathbf{X}_i$ be transformation of the origin into $\mathbf{x}_i$
- Let $\mathbf{X}_i^{-1}$ be the inverse transformation
- We can express relative transformation $\mathbf{X}_i^{-1}\mathbf{X}_j$

Transformations

- **How to express $\mathbf{x}_j$ relative to $\mathbf{x}_i$?**
- Express this through transformations
- Let $\mathbf{X}_i$ be transformation of the origin into $\mathbf{x}_i$
- Let $\mathbf{X}_i^{-1}$ be the inverse transformation
- We can express relative transformation $\mathbf{X}_i^{-1}\mathbf{X}_j$
- Transformations can be expressed using **homogenous coordinates**

Transformations

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

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

- Observation-Based edge

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

describes “how node i sees node j”

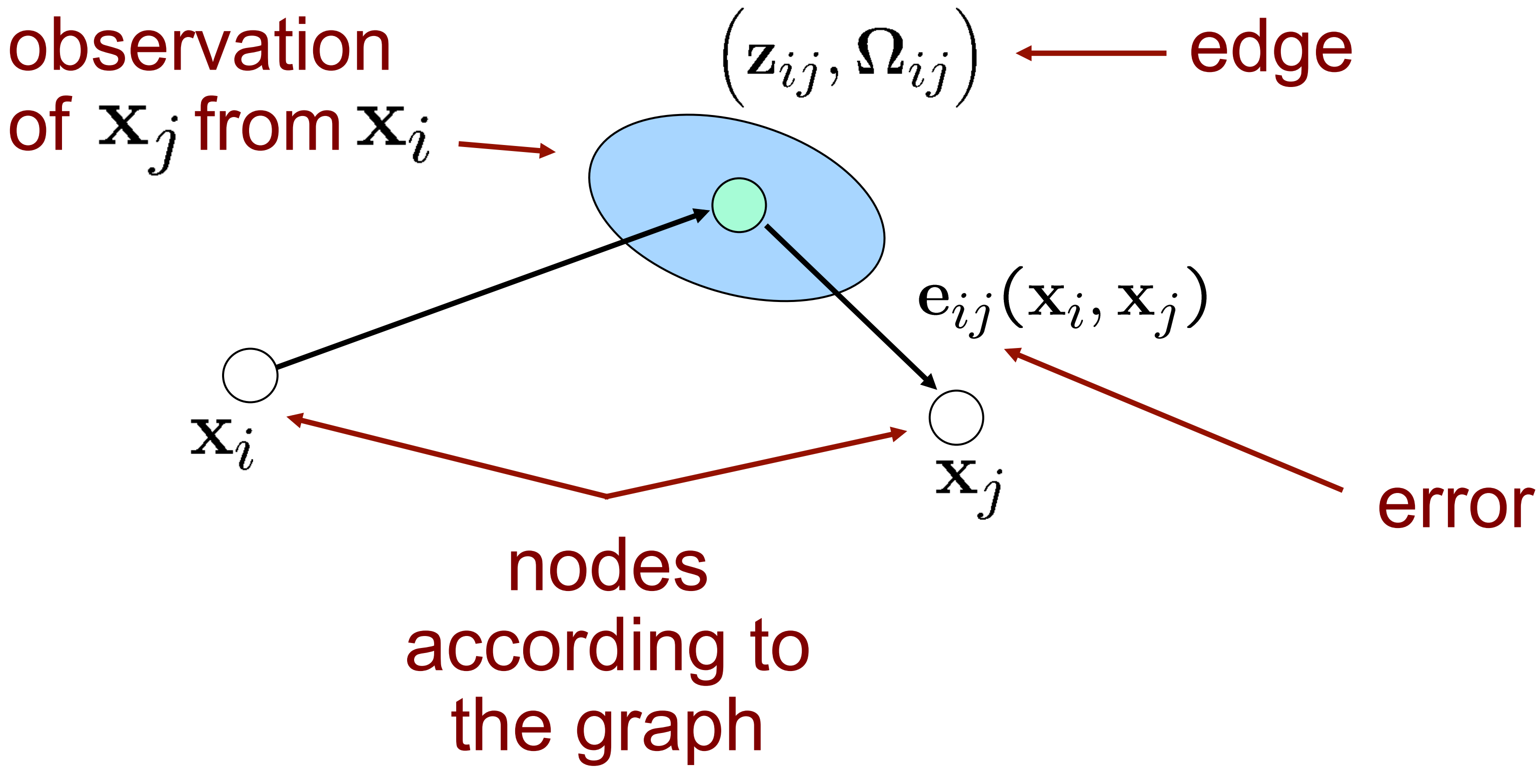
The Edge Information Matrices

- 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

Question

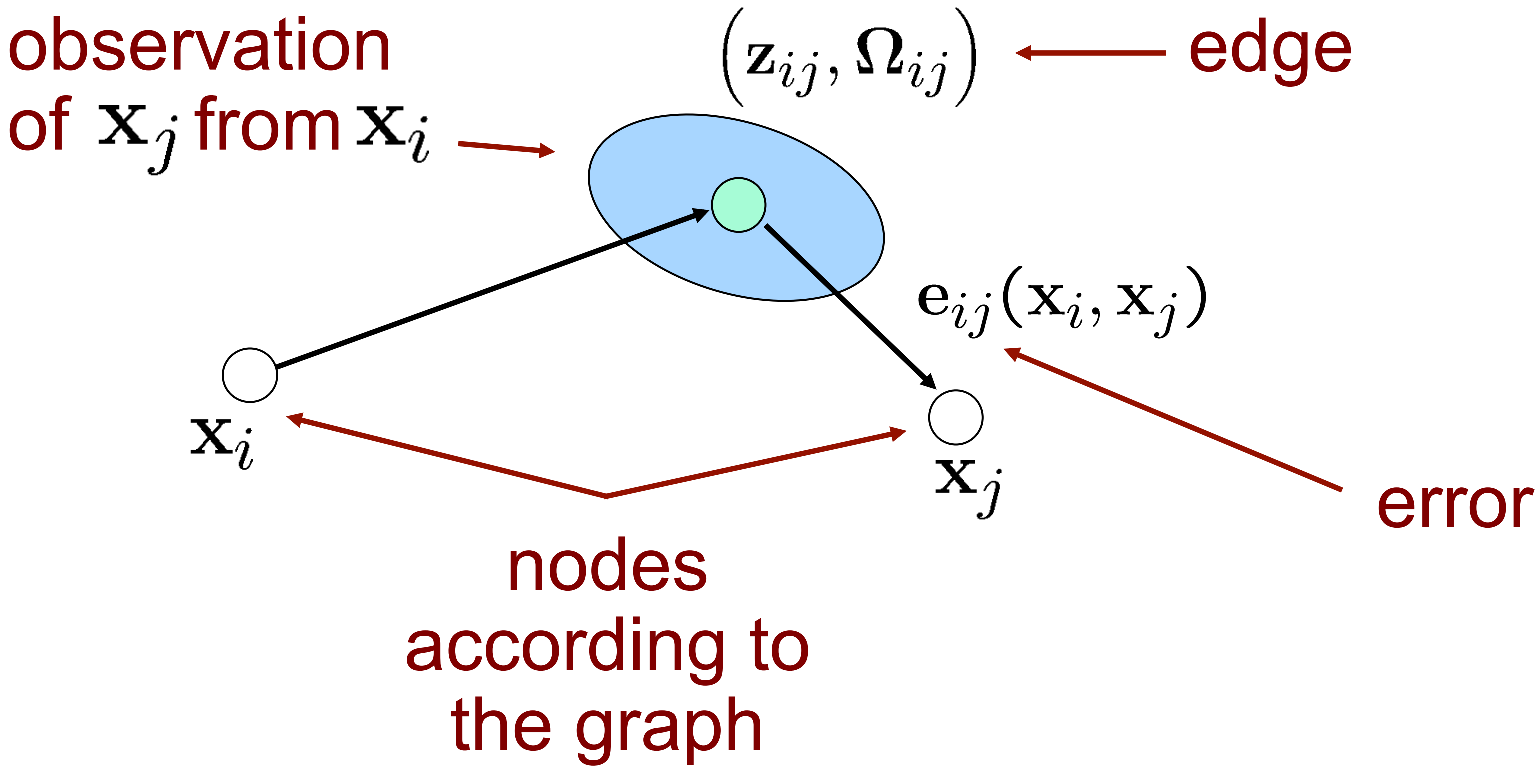
- What should these matrices look like when moving in a long, featureless corridor?

Pose-Graph



observation
of $\mathbf{x}_j$ from $\mathbf{x}_i$
$\left(\mathbf{z}_{ij}, \mathbf{\Omega}_{ij}\right)$
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: $\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{ij} \mathbf{e}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{e}_{ij}$

The Error Function

- Error function for a single constraint

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

- 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

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

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 $\mathbf{x}_j$

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

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 $\mathbf{x}_j$

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

Consequences of the Sparsity

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

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

- 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

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

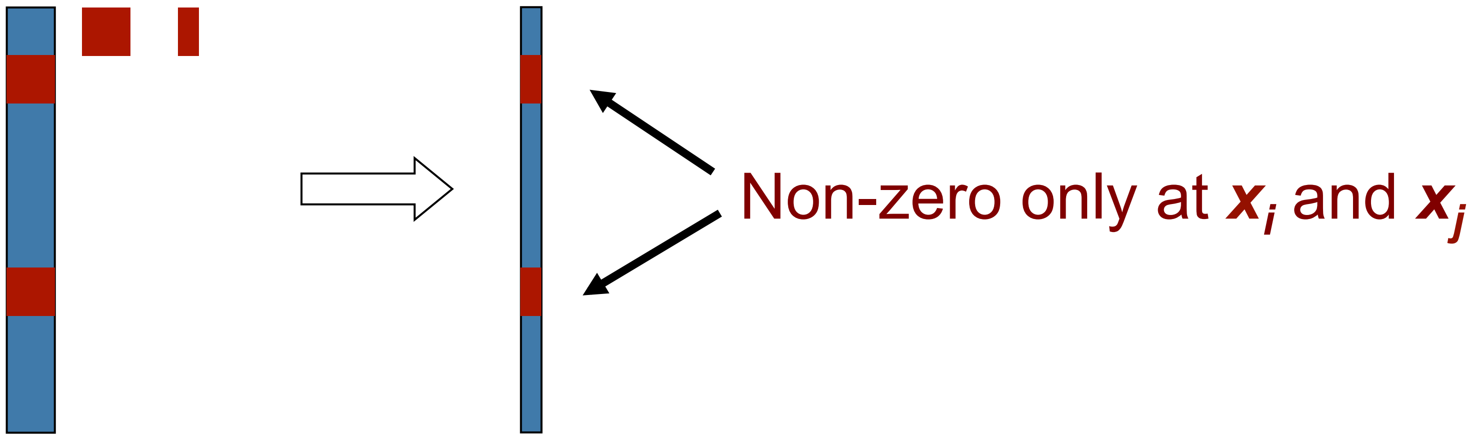
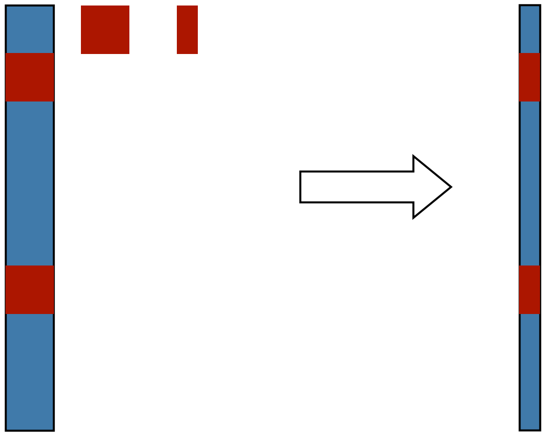


Illustration of the Structure

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



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

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

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

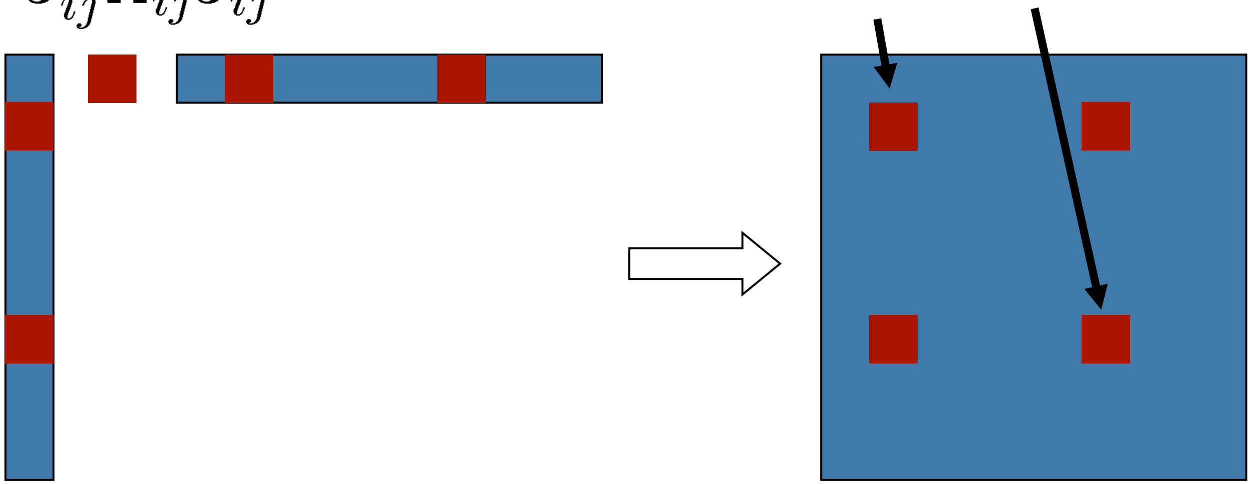
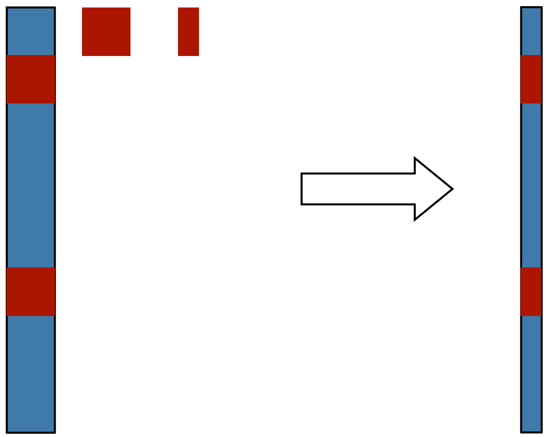


Illustration of the Structure

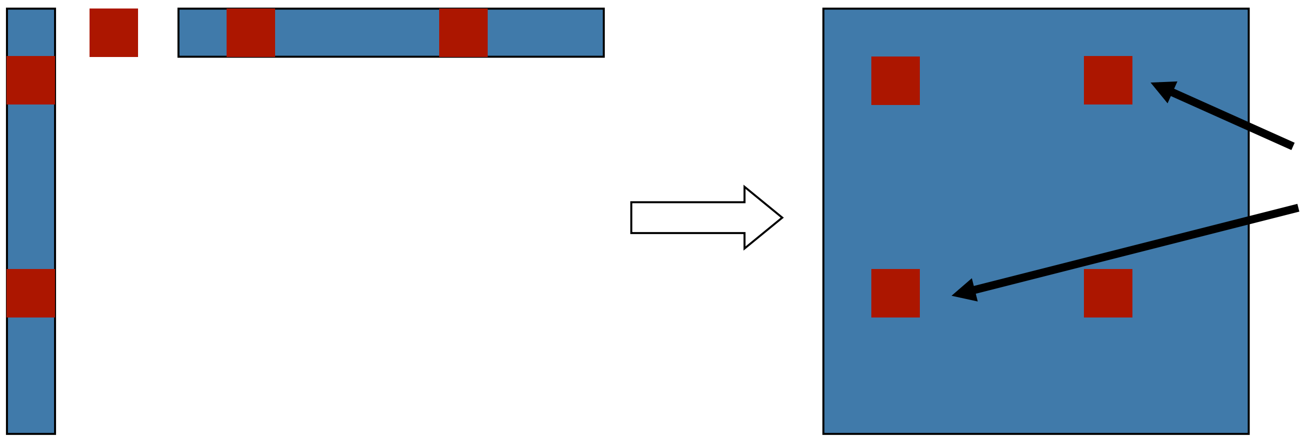
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$$



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

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

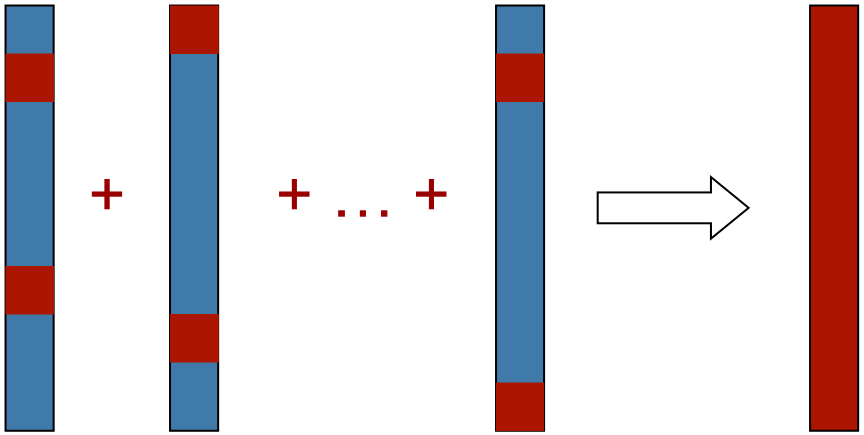
$$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}$$



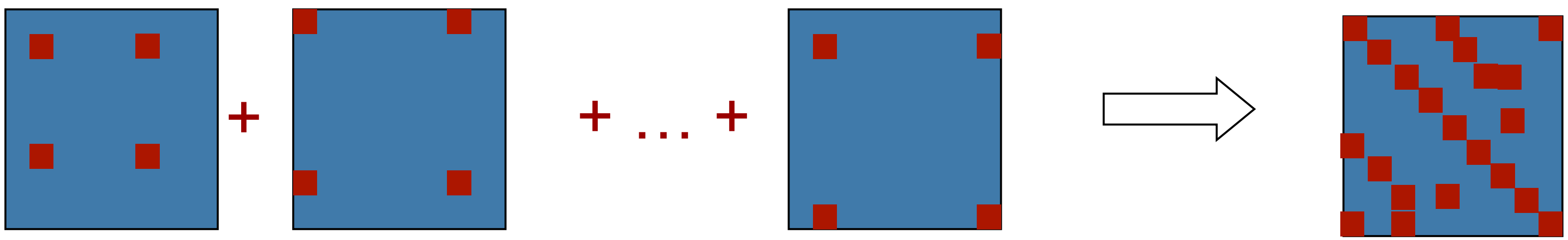
... and at the blocks *ij*,*ji*

Illustration of the Structure

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



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



Sparsity Effect on b

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

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

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

Sparsity Effect on H

- The coefficient matrix of an edge is:

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

- Non-zero only in the blocks relating i,j

Sparsity Summary

- An edge ij contributes only to the
  - ith and the jth 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

All We Need...

- 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:

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

small blocks
(or vectors)
corresponding
to the individual
constraints

... for 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 (!$converged$)

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

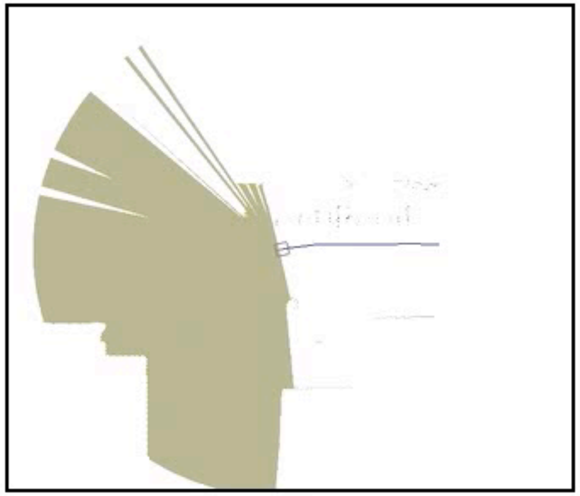
$\boldsymbol{\Delta}\mathbf{x} = \text{solveSparse}(\mathbf{H}\boldsymbol{\Delta}\mathbf{x} = -\mathbf{b})$

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

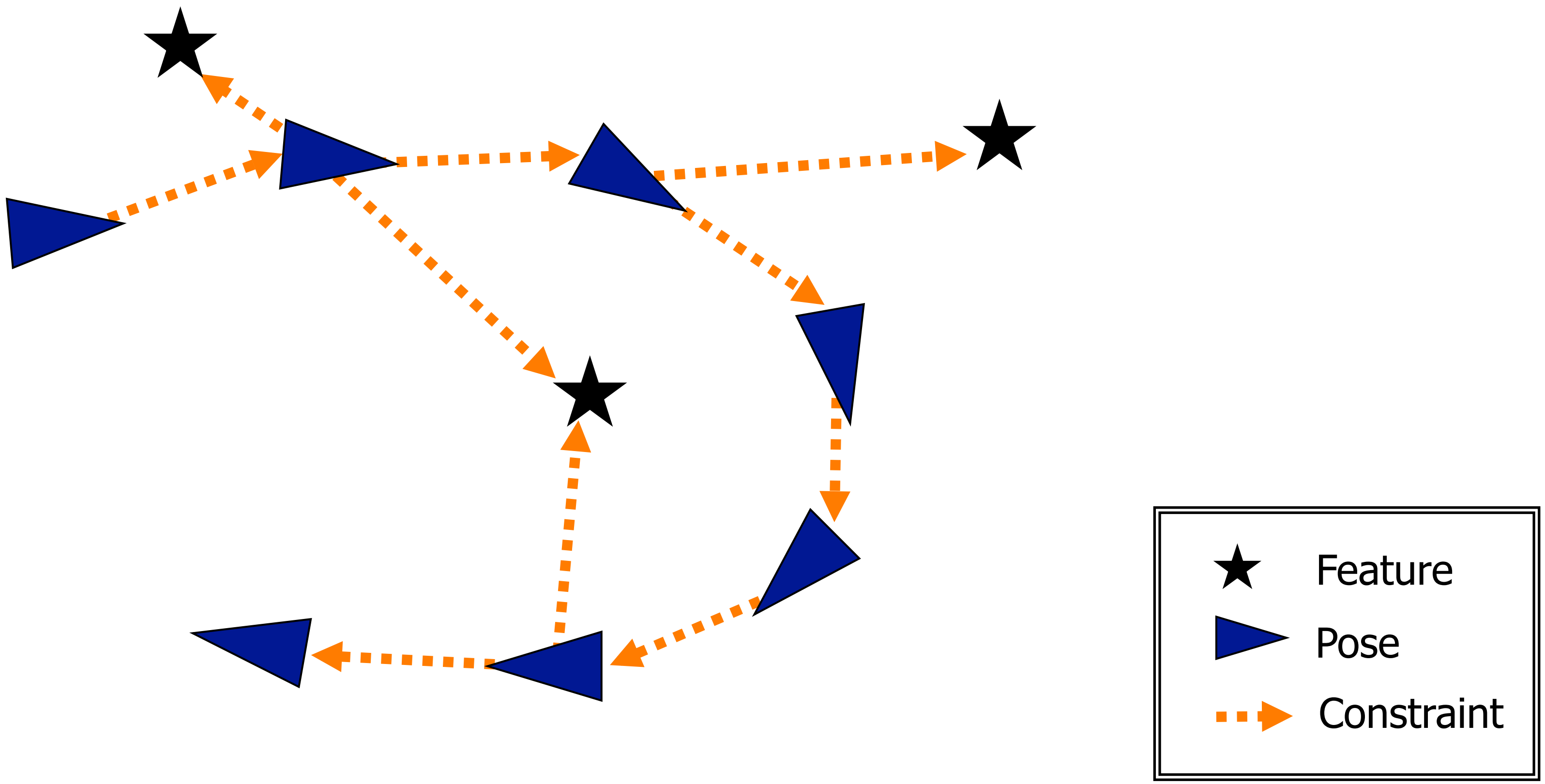
end

return $\mathbf{x}$

Real World Examples

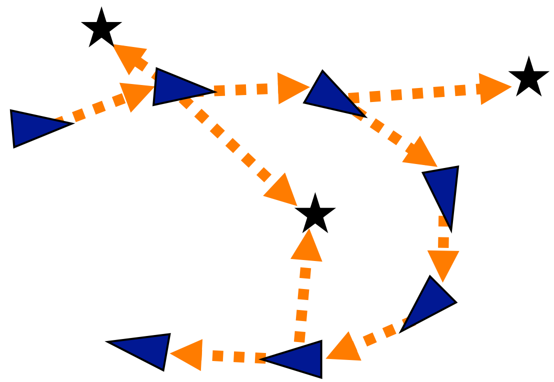


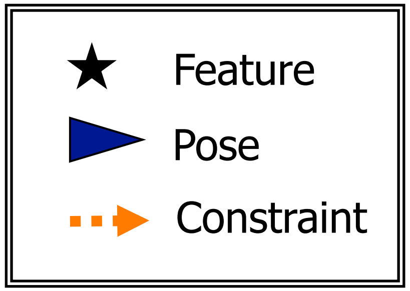
The Graph with Landmarks



The Graph with Landmarks

- **Nodes** can represent:
  - Robot poses
  - Landmark locations
- **Edges** can represent:
  - Landmark observations
  - Odometry measurements
- The minimization optimizes the **landmark locations and robot poses**





Landmarks Observation

- Expected observation (x-y sensor)

$$\hat{\mathbf{z}}_{il}(\underset{\substack{\uparrow\\\text{robot}}}{\mathbf{x}_i}, \underset{\substack{\uparrow\\\text{landmark}}}{\mathbf{x}_l}) = \mathbf{X}_i^{-1} \begin{pmatrix} \mathbf{x}_l \\ 1 \end{pmatrix}$$

Landmarks Observation

- Expected observation (x-y sensor)

$$\underbrace{\hat{\mathbf{z}}_{il}(\mathbf{x}_i}_{\text{robot}}, \underbrace{\mathbf{x}_l}_{\text{landmark}}) = \mathbf{X}_i^{-1} \begin{pmatrix} \mathbf{x}_l \\ 1 \end{pmatrix}$$

- Error function (in Euclidian space)

$$\mathbf{e}_{il}(\mathbf{x}_i, \mathbf{x}_l) \;=\; \hat{\mathbf{z}}_{il} - \mathbf{z}_{il}$$

Bearing Only Observations

- A landmark is still a 2D point
- The robot observe only the bearing towards the landmark
- 1D Observation function

$$\widehat{\mathbf{z}}_{il}(\mathbf{x}_i, \mathbf{x}_l) \quad = \quad \text{atan}\frac{(\mathbf{x}_l - \mathbf{t}_i).y}{(\mathbf{x}_l - \mathbf{t}_i).x} - \theta_i$$

robot landmark robot-landmark angle robot orientation

Bearing Only Observations

- Observation function

$$\hat{\mathbf{z}}_{il}(\mathbf{x}_i, \mathbf{x}_l) \quad = \quad \text{atan}\frac{(\mathbf{x}_l - \mathbf{t}_i).y}{(\mathbf{x}_l - \mathbf{t}_i).x} - \theta_i$$

robot landmark robot-landmark angle robot orientation

- Error function

$$\mathbf{e}_{il}(\mathbf{x}_i, \mathbf{x}_l) \quad = \quad \text{atan}\frac{(\mathbf{x}_l - \mathbf{t}_i).y}{(\mathbf{x}_l - \mathbf{t}_i).x} - \theta_i - \mathbf{z}_{il}$$

The Rank of the Matrix H

- What is the rank of $\mathbf{H}_{ij}$ for a 2D landmark-pose constraint?

The Rank of the Matrix H

- What is the rank of $\mathbf{H}_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $\mathbf{J}_{ij}$ are a 2x3 matrices
  - $\mathbf{H}_{ij}$ cannot have more than rank 2

    $\mathrm{rank}(A^T A) = \mathrm{rank}(A^T) = \mathrm{rank}(A)$

The Rank of the Matrix H

- What is the rank of $\mathbf{H}_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $\mathbf{J}_{ij}$ are a 2x3 matrices
  - $\mathbf{H}_{ij}$ cannot have more than rank 2

  $$\text{rank}(A^T A) = \text{rank}(A^T) = \text{rank}(A)$$

- What is the rank of $\mathbf{H}_{ij}$ for a bearing-only constraint?

The Rank of the Matrix H

- What is the rank of $\mathbf{H}_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $\mathbf{J}_{ij}$ are a 2x3 matrices
  - $\mathbf{H}_{ij}$ cannot have more than rank 2

    $$\text{rank}(A^T A) = \text{rank}(A^T) = \text{rank}(A)$$

- What is the rank of $\mathbf{H}_{ij}$ for a bearing-only constraint?
  - The blocks of $\mathbf{J}_{ij}$ are a 1x3 matrices
  - $\mathbf{H}_{ij}$ has rank 1

Rank

- In landmark-based SLAM, the system can be under-determined
- The rank of $\mathbf{H}$ is **less or equal** to the sum of the ranks of the constraints
- To determine a **unique solution**, the system must have **full rank**

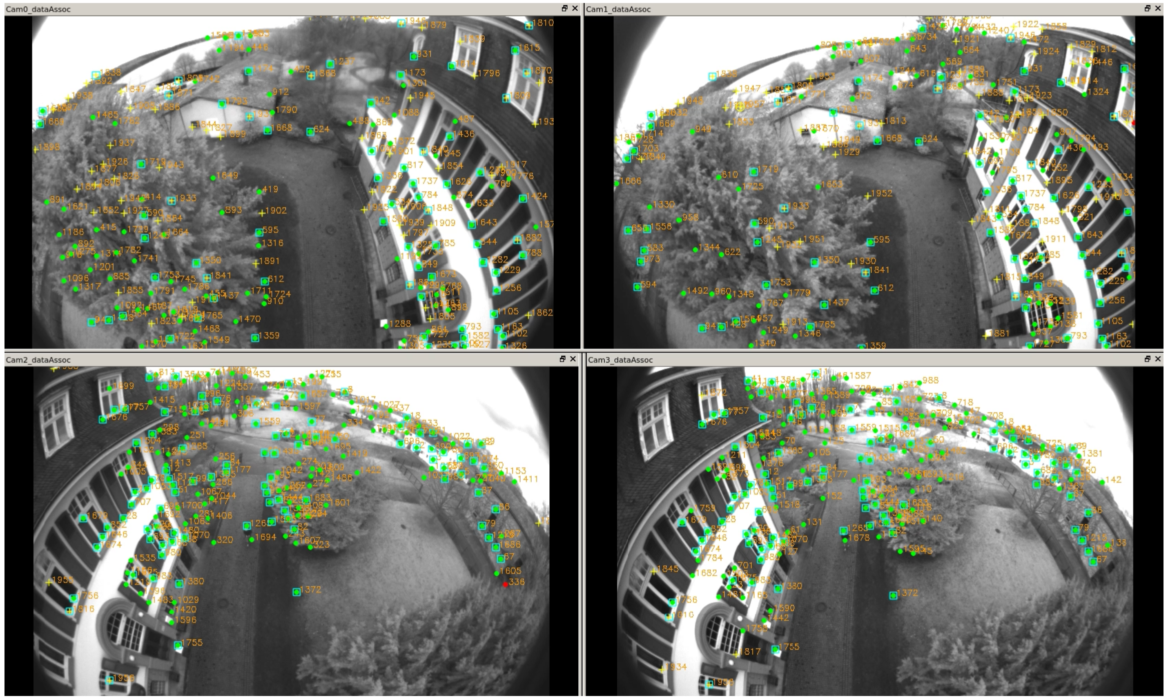
Under-Determined Systems

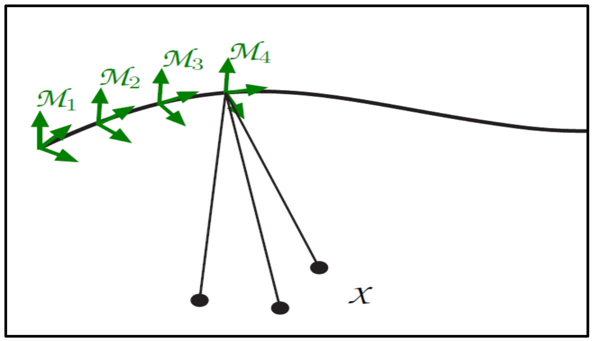
- No guarantee for a full rank system
  - Landmarks may be observed only once
  - Robot might have no odometry
- We can still deal with these situations by adding a "damping" factor to $\mathbf{H}$
- Instead of solving $\mathbf{H}\mathbf{\Delta x} = -\mathbf{b}$, we solve

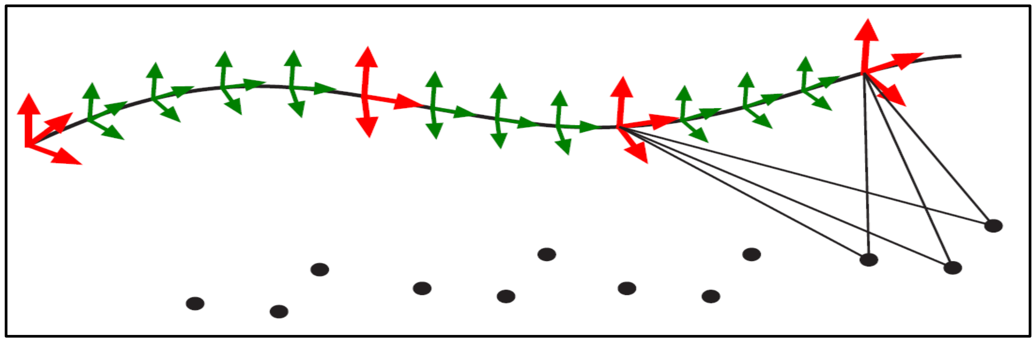
$$(\mathbf{H} + \lambda \mathbf{I})\mathbf{\Delta x} = -\mathbf{b}$$

⇨ **Levenberg Marquardt**

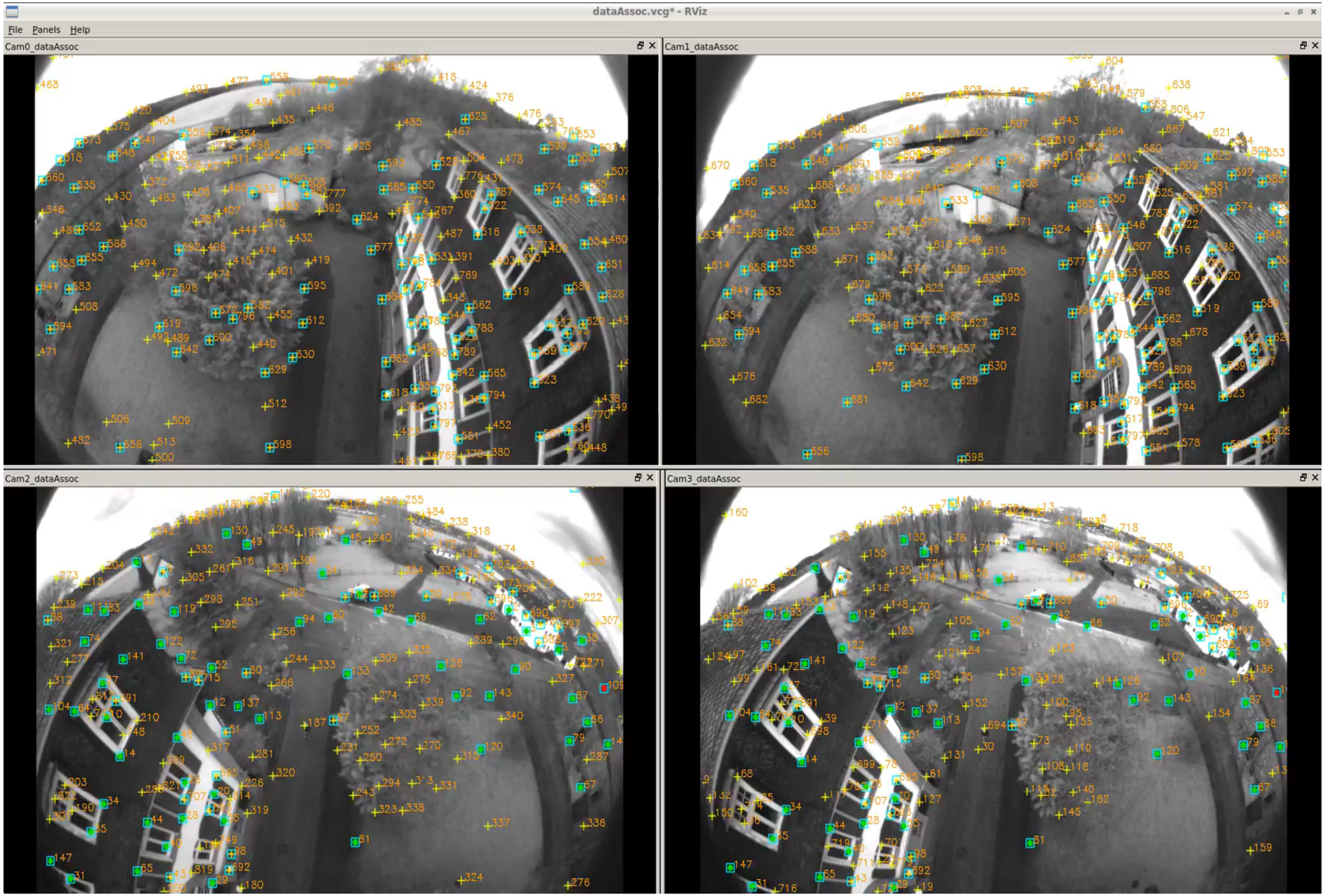
UAV Example


Cam0_dataAssoc
Cam1_dataAssoc
Cam2_dataAssoc
Cam3_dataAssoc


$\mathcal{M}_1$
$\mathcal{M}_2$
$\mathcal{M}_3$
$\mathcal{M}_4$
x



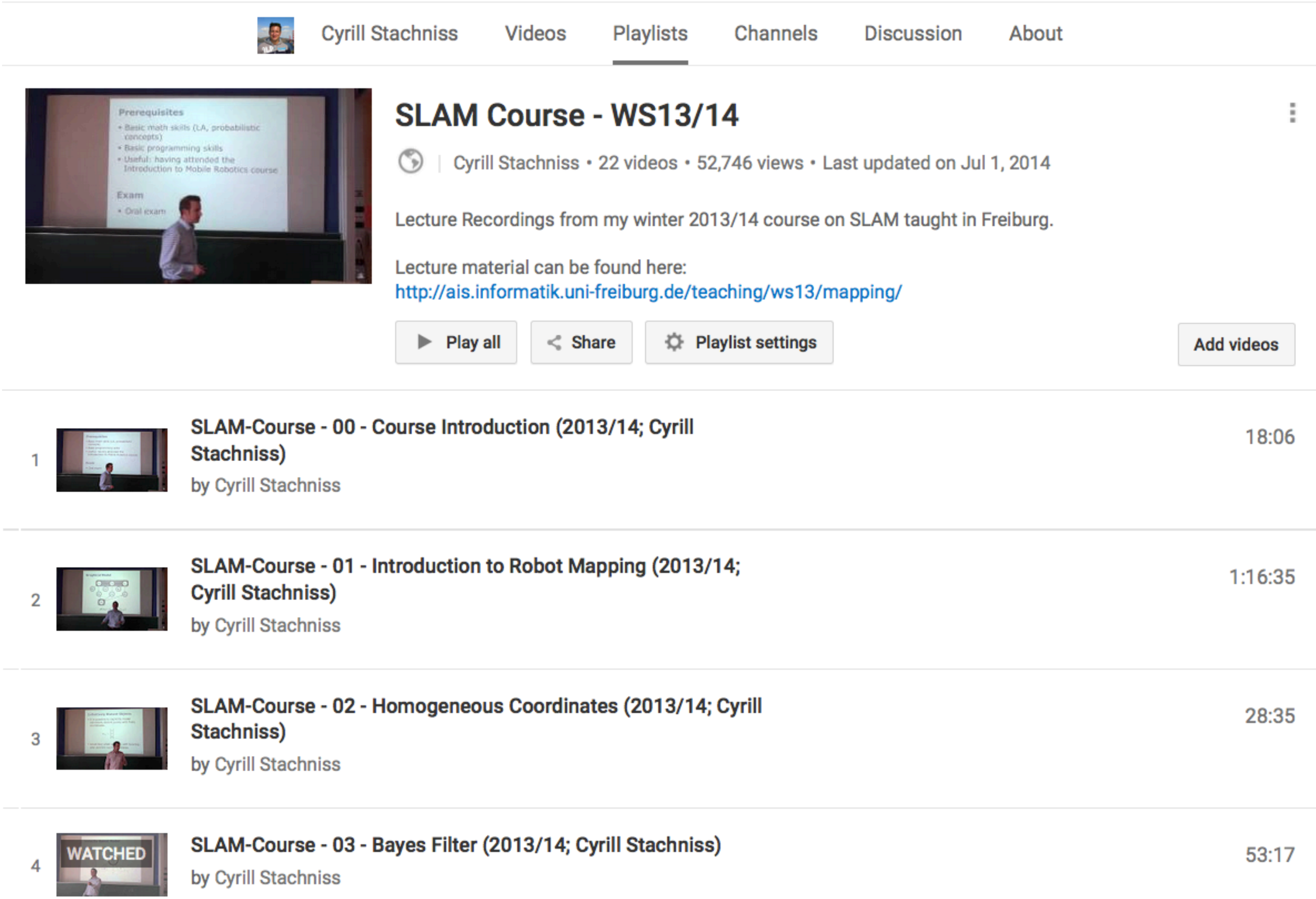
UAV Example


dataAssoc.vcg* - RViz
File Panels Help
Cam0_dataAssoc
Cam1_dataAssoc
Cam2_dataAssoc
Cam3_dataAssoc

Summary

- The back-end part of the SLAM problem can be solved with GN or LM
- The $\mathbf{H}$ matrix is typically sparse
- This sparsity allows for efficiently solving the linear system
- There are several extensions (online, robust methods wrt outliers or initialization, hierarchical approaches, exploiting sparsity, multiple sensors)

YouTube Lectures



https://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_

Thank you for your attention!

Slide Information

- These slides have been created by Cyrill Stachniss, Giorgio Grisetti and Wolfram Burgard evolving from different courses and tutorials that we taught over the years between 2010 and 2016.
- 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 of my lectures on robot mapping are available through YouTube:
http://www.youtube.com/playlist?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_&feature=g-list

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