Graph-Based SLAM in 90 Minutes

Cyrill Stachniss

UNIVERSITÄT BONN

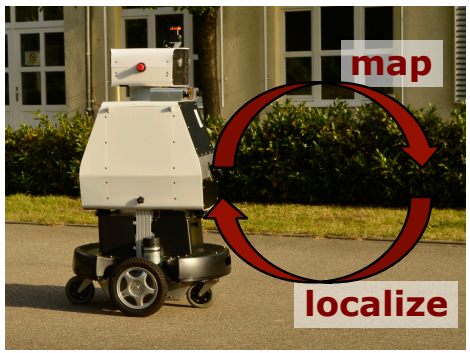
Mobile Robots Need Maps for Autonomous Navigation

SLAM = Build a Map and Localize in that Map

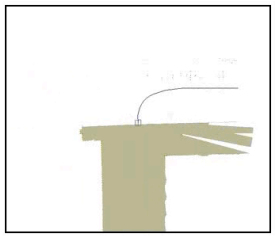
The SLAM Problem

SLAM is a **chicken-or-egg** problem:

- → a map is needed for localization and
- → a pose estimate is needed for mapping



LiDAR-Based 2D SLAM Example



Example 3D SLAM & Semantics

Definition of the SLAM Problem

Given

- The robot's controls

$$u_{1:T} = \{u_1, u_2, u_3, \ldots, u_T\}$$

- Observations

$$z_{1:T} = \{z_1, z_2, z_3, \ldots, z_T\}$$

Wanted

- Map of the environment

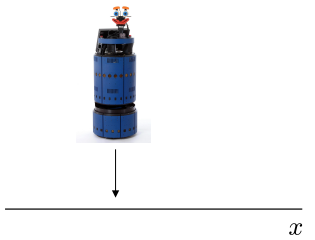
$$m$$

- Path of the robot

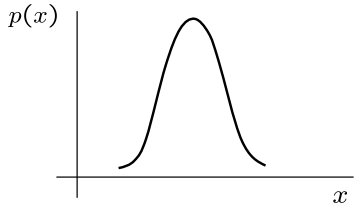
$$x_{0:T} = \{x_0, x_1, x_2, \ldots, x_T\}$$

Probabilistic Approaches

- Uncertainty in the robot's motions and observations
- Use the probability theory to explicitly represent the uncertainty



"The robot is exactly here"



"The robot is somewhere here"

In the Probabilistic World

Estimate the robot's path and the map

$$p(x_{0:T}, m \mid z_{1:T}, u_{1:T})$$

distribution path map given observations controls

Popular approaches for estimation

- Kalman filters
- Particle filters
- Graph-based optimization

Graph-Based SLAM

Graph-Based SLAM

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

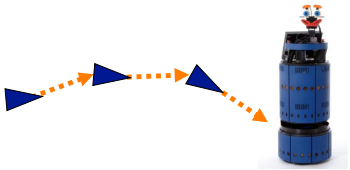
Graph-Based SLAM

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

SLAM = simultaneous localization and mapping

Graph-Based SLAM

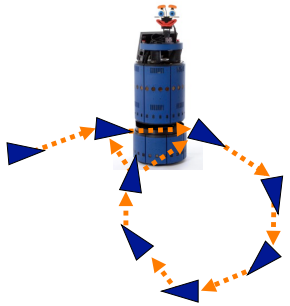
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain



Robot pose Constraint

Graph-Based SLAM

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



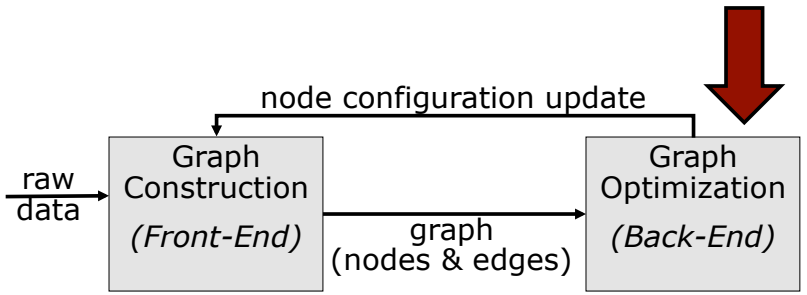
Robot pose Constraint

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

Front-End and Back-End

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



Graph-SLAM and Least Squares

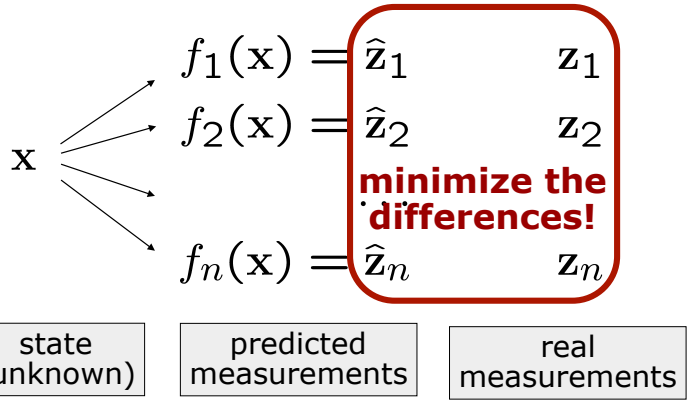
- The nodes represent the **state**
- Given a state, we can compute what we **expect** to perceive
- We have **real observations** relating the nodes with each other

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

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

Graphical Explanation



Error Function

- Error $\mathbf{e}_i$ is typically the **difference** between the **predicted and actual** measurement

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

- We assume that the measurement error has **zero mean** and is **normally distributed**
- Gaussian error with information matrix $\mathbf{\Omega}_i$
- The squared error of a measurement depends only on the state and is a scalar

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

Goal: Find the Minimum

- Find the state $\mathbf{x}^*$ which minimizes the error given all measurements

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}) \quad \leftarrow \text{global error (scalar)}$$

$$= \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i e_i(\mathbf{x}) \quad \leftarrow \text{squared error terms (scalar)}$$

$$= \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \mathbf{e}_i^T(\mathbf{x}) \boldsymbol{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

error terms (vector)

Goal: Find the Minimum

- Find the state $\mathbf{x}^*$ which minimizes the error given all measurements

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_i \mathbf{e}_i^T(\mathbf{x}) \boldsymbol{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

- A general solution is to derive the global error function and find its nulls
- In general complex and no closed form solution

➡ Numerical approaches

Assumption

- A good initial guess is available
- The error functions are "smooth" in the neighborhood of the (hopefully global) minima

- Then, we can solve the problem by iterative local linearizations

Solve Via Iterative Linearizations

- Linearize the error terms around the current solution/initial guess
- Compute the first derivative of the squared error function
- Set it to zero and solve linear system
- Obtain the new state (that is hopefully closer to the minimum)
- Iterate

Linearizing the Error Function

- Approximate the error functions around an initial guess $\mathbf{x}$ via Taylor expansion

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

- Reminder: Jacobian

$$\mathbf{J}_f(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \frac{\partial f_1(x)}{\partial x_2} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \frac{\partial f_2(x)}{\partial x_1} & \frac{\partial f_2(x)}{\partial x_2} & \cdots & \frac{\partial f_2(x)}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m(x)}{\partial x_1} & \frac{\partial f_m(x)}{\partial x_2} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{pmatrix}$$

Squared Error

- With this linearization, we can fix $\mathbf{x}$ and carry out the minimization in the increments $\Delta\mathbf{x}$
- We replace the Taylor expansion in the squared error terms:

$$e_i(\mathbf{x}+\Delta\mathbf{x}) = \dots$$

Squared Error

- With this linearization, we can fix $\mathbf{x}$ and carry out the minimization in the increments $\Delta\mathbf{x}$
- We replace the Taylor expansion in the squared error terms:

$$\begin{aligned} e_i(\mathbf{x}+\Delta\mathbf{x}) &= \mathbf{e}_i^T(\mathbf{x}+\Delta\mathbf{x})\Omega_i\mathbf{e}_i(\mathbf{x}+\Delta\mathbf{x}) \\ &\simeq (\mathbf{e}_i+\mathbf{J}_i\Delta\mathbf{x})^T\Omega_i(\mathbf{e}_i+\mathbf{J}_i\Delta\mathbf{x}) \end{aligned}$$

Squared Error

- With this linearization, we can fix $\mathbf{x}$ and carry out the minimization in the increments $\Delta\mathbf{x}$
- We replace the Taylor expansion in the squared error terms:

$$\begin{aligned} e_i(\mathbf{x}+\Delta\mathbf{x}) &= \mathbf{e}_i^T(\mathbf{x}+\Delta\mathbf{x})\Omega_i\mathbf{e}_i(\mathbf{x}+\Delta\mathbf{x}) \\ &\simeq (\mathbf{e}_i+\mathbf{J}_i\Delta\mathbf{x})^T\Omega_i(\mathbf{e}_i+\mathbf{J}_i\Delta\mathbf{x}) \\ &= \mathbf{e}_i^T\Omega_i\mathbf{e}_i + \\ &\quad \mathbf{e}_i^T\Omega_i\mathbf{J}_i\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{J}_i^T\Omega_i\mathbf{e}_i + \\ &\quad \Delta\mathbf{x}^T\mathbf{J}_i^T\Omega_i\mathbf{J}_i\Delta\mathbf{x} \end{aligned}$$

Squared Error (cont.)

- By grouping similar terms, we obtain:

$$
\begin{aligned}
& e_i(\mathbf{x}+\Delta\mathbf{x}) \\
& \simeq \underbrace{\mathbf{e}_i^T\Omega_i\mathbf{e}_i}_{c_i} + 2\underbrace{\mathbf{e}_i^T\Omega_i\mathbf{J}_i}_{\mathbf{b}_i^T}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\mathbf{J}_i^T\Omega_i\mathbf{J}_i}_{\mathbf{H}_i}\Delta\mathbf{x} \\
& = c_i + 2\mathbf{b}_i^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_i\Delta\mathbf{x}
\end{aligned}
$$

Global Error

- The global error is the sum of the squared errors terms corresponding to the individual measurements

$$
\begin{aligned}
F(\mathbf{x}+\Delta\mathbf{x}) & \simeq \sum_i \left(c_i + \mathbf{b}_i^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_i\Delta\mathbf{x}\right) \\
& = \sum_i c_i + 2\left(\sum_i \mathbf{b}_i^T\right)\Delta\mathbf{x} + \Delta\mathbf{x}^T\left(\sum_i \mathbf{H}_i\right)\Delta\mathbf{x}
\end{aligned}
$$

Global Error

- The global error is the sum of the squared errors terms corresponding to the individual measurements

$$
\begin{aligned}
F(\mathbf{x}+\Delta\mathbf{x}) & \simeq \sum_i \left(c_i + \mathbf{b}_i^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_i\Delta\mathbf{x}\right) \\
& = \underbrace{\sum_i c_i}_{c} + 2\underbrace{\left(\sum_i \mathbf{b}_i^T\right)}_{\mathbf{b}^T}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\left(\sum_i \mathbf{H}_i\right)}_{\mathbf{H}}\Delta\mathbf{x}
\end{aligned}
$$

Global Error

- The global error is the sum of the squared errors terms corresponding to the individual measurements

$$
\begin{aligned}
F(\mathbf{x}+\Delta\mathbf{x}) & \simeq \sum_i \left(c_i + \mathbf{b}_i^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}_i\Delta\mathbf{x}\right) \\
& = \underbrace{\sum_i c_i}_{c} + 2\underbrace{\left(\sum_i \mathbf{b}_i^T\right)}_{\mathbf{b}^T}\Delta\mathbf{x} + \Delta\mathbf{x}^T\underbrace{\left(\sum_i \mathbf{H}_i\right)}_{\mathbf{H}}\Delta\mathbf{x} \\
& = c + 2\mathbf{b}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}
\end{aligned}
$$

$$
\mathbf{b}^T = \sum_i \mathbf{e}_i^T\Omega_i\mathbf{J}_i \qquad \mathbf{H} = \sum_i \mathbf{J}_i^T\Omega\mathbf{J}_i
$$

Quadratic Form

- We can write the global error terms as a quadratic form in $\Delta\mathbf{x}$

$$F(\mathbf{x}+\Delta\mathbf{x}) \simeq c + 2\mathbf{b}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

- Our goal is to minimize this function
- We need to compute the derivative of $F(\mathbf{x}+\Delta\mathbf{x})$ w.r.t. $\Delta\mathbf{x}$

Deriving a Quadratic Form

- Given a quadratic form

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{b}^T\mathbf{x}$$

- its first derivative is

$$\frac{\partial f}{\partial \mathbf{x}} = (\mathbf{H}+\mathbf{H}^T)\mathbf{x} + \mathbf{b}$$

See: The Matrix Cookbook, Section 2.2.4

Quadratic Form

- We can write the linearized global error terms as a quadratic form in $\Delta\mathbf{x}$

$$F(\mathbf{x}+\Delta\mathbf{x}) \simeq c + 2\mathbf{b}^T\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H}\Delta\mathbf{x}$$

- The derivative of $F(\mathbf{x}+\Delta\mathbf{x})$ w.r.t. $\Delta\mathbf{x}$ is then:

$$\frac{\partial F(\mathbf{x}+\Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \simeq 2\mathbf{b} + 2\mathbf{H}\Delta\mathbf{x}$$

Minimizing the Quadratic Form

- Derivative $\frac{\partial F(\mathbf{x}+\Delta\mathbf{x})}{\partial \Delta\mathbf{x}} \simeq 2\mathbf{b} + 2\mathbf{H}\Delta\mathbf{x}$
- Setting it to zero leads to

$$0 = 2\mathbf{b} + 2\mathbf{H}\Delta\mathbf{x}$$

- Which leads to the linear system

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$$

- The solution for the increment $\Delta\mathbf{x}^*$ is

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

Overall Procedure

Iterate the following steps:

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

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

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

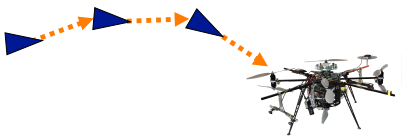
$$\Delta\mathbf{x}^* = -\mathbf{H}^{-1}\mathbf{b}$$

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

Let's use that for SLAM

Pose-Graph-Based SLAM

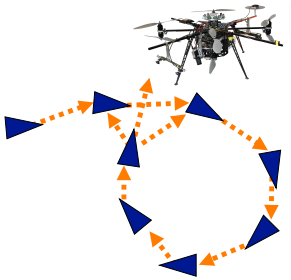
- Constraints connect the poses of the robot while it is moving
- Constraints are inherently uncertain



Robot pose Constraint

Pose-Graph-Based SLAM

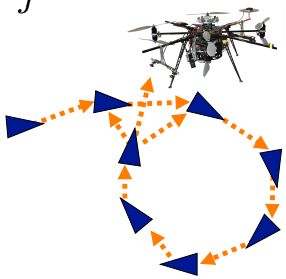
- Observing previously seen areas generates constraints between non-successive poses



Robot pose Constraint

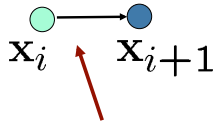
The Pose-Graph

- 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)

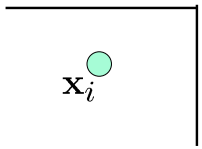
- …the robot moves from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$
- Edge corresponds to odometry



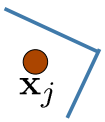
The edge represents the **odometry** measurement

Create an Edge If… (2)

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



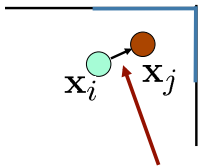
Measurement from $\mathbf{x}_i$



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 does $\mathbf{x}_i$ sees $\mathbf{x}_j$?**
- 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 does $\mathbf{x}_i$ sees $\mathbf{x}_j$?**
- 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**

Homogenous Coordinates

- N-dim space expressed in N+1 dim
- 4 dim. for modeling the 3D space
- To HC: $(x,y,z)^T \rightarrow (x,y,z,1)^T = (a,b,c,d)^T$
- Backwards: $(a,b,c,d)^T \rightarrow \left(\frac{a}{d},\frac{b}{d},\frac{c}{d}\right)^T = (x,y,z)^T$

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{R} = \begin{pmatrix} \mathbf{R}^{3D} & 0 \\ 0 & 1 \end{pmatrix} \qquad \mathbf{X} = \begin{pmatrix} \mathbf{R}^{3D} & \mathbf{t} \\ 0 & 1 \end{pmatrix}$$

translation rotation rigid-body

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”

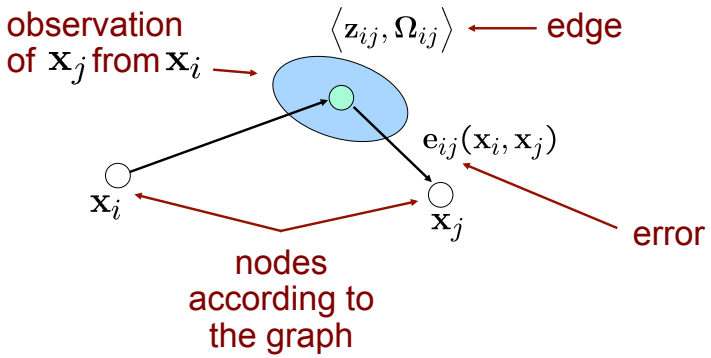
The Edge Information Matrices

- 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

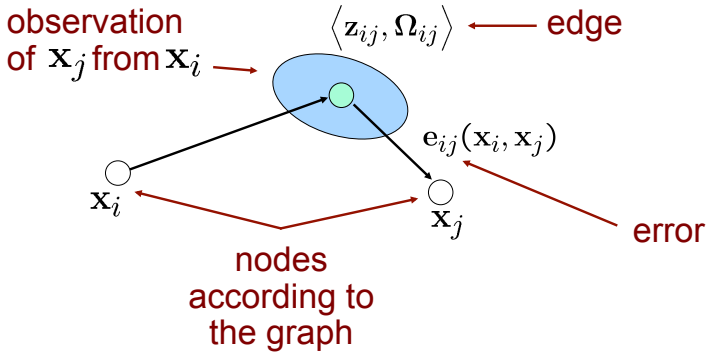
Question

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

Pose-Graph



Pose-Graph



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

The Error Function

- Error function for a single constraint

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

measurement — $\boldsymbol{x}_j$ referenced w.r.t. $\boldsymbol{x}_i$

- 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

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

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

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 $\mathbf{x}_j$

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

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

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 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 $\mathbf{H}$:

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

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

- 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

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

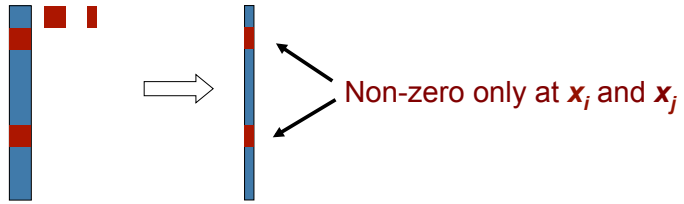
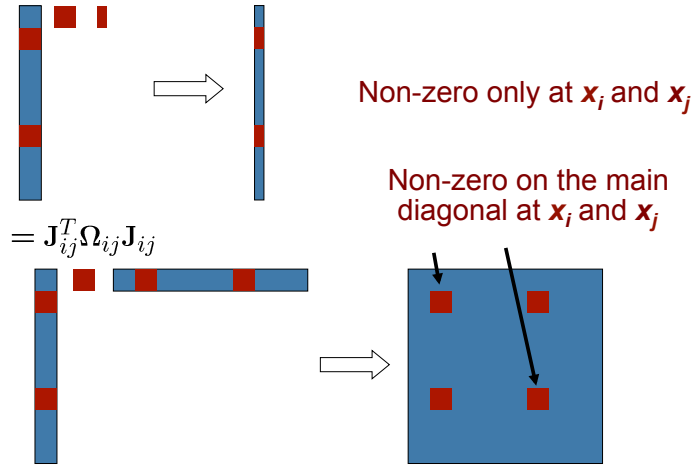


Illustration of the Structure

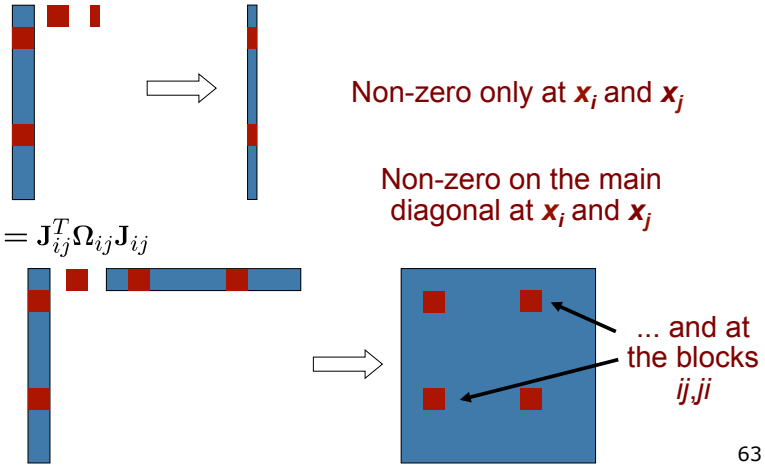
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}$$



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

Illustration of the Structure

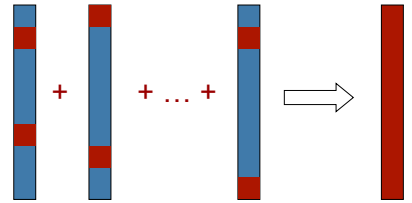
$$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}$$



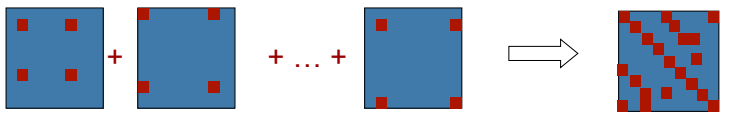
$$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}$$

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:

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

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

Consequences of the Sparsity

- The coefficient matrix of an edge is:

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

- 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

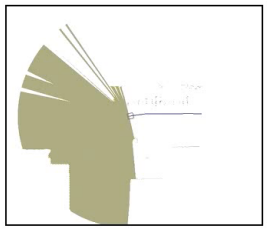
Algorithm

1: **optimize(x):**
2: while (!*converged*)
3: $(\mathbf{H}, \mathbf{b}) = \text{buildLinearSystem}(\mathbf{x})$
4: $\boldsymbol{\Delta}\mathbf{x} = \text{solveSparse}(\mathbf{H}\boldsymbol{\Delta}\mathbf{x} = -\mathbf{b})$
5: $\mathbf{x} = \mathbf{x} + \boldsymbol{\Delta}\mathbf{x}$
6: end
7: *return* $\mathbf{x}$

How to Efficiently Solve the Linear System?

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

Real World Examples



Relation to Probabilistic State Estimation

- So far, we minimized an error function
- How does this relate to state estimation in the probabilistic sense?

General State Estimation

- Bayes rule, independence and Markov assumptions allow us to write

$$p(x_{0:t} \mid z_{1:t}, u_{1:t}) = \eta\, p(x_0) \prod_t [p(x_t \mid x_{t-1}, u_t)\, p(z_t \mid x_t)]$$

Log Likelihood

- Written as the log likelihood, leads to

$$\begin{aligned}&\log p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\ &\quad = \quad \text{const.} + \log p(x_0) \\ &\qquad + \sum_t \left[\log p(x_t \mid x_{t-1}, u_t) + \log p(z_t \mid x_t)\right]\end{aligned}$$

Gaussian Assumption

- Assuming Gaussian distributions

$$\begin{aligned}&\log p(x_{0:t} \mid z_{1:t}, u_{1:t}) \\ &\quad = \quad \text{const.} + \log \underbrace{p(x_0)}_{\mathcal{N}} \\ &\qquad + \sum_t \left[\log \underbrace{p(x_t \mid x_{t-1}, u_t)}_{\mathcal{N}} + \log \underbrace{p(z_t \mid x_t)}_{\mathcal{N}}\right]\end{aligned}$$

Log of a Gaussian

- Log likelihood of a Gaussian

$$\begin{aligned}&\log \mathcal{N}(x, \mu, \Sigma) \\ &\quad = \quad \text{const.} - \frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\end{aligned}$$

Error Function as Exponent

- Log likelihood of a Gaussian

$$\begin{aligned}&\log \mathcal{N}(x, \mu, \Sigma) \\ &\quad = \quad \text{const.} - \frac{1}{2} \underbrace{\underbrace{(x-\mu)^T}_{\mathbf{e}^T(x)} \underbrace{\Sigma^{-1}}_{\Omega} \underbrace{(x-\mu)}_{\mathbf{e}(x)}}_{e(x)}\end{aligned}$$

- is up to a constant equivalent to the error functions used before

Log Likelihood with Error Terms

- Assuming Gaussian distributions

$$\log p(x_{0:t} \mid z_{1:t}, u_{1:t})$$
$$= \text{const.} - \frac{1}{2} e_p(x) - \frac{1}{2} \sum_t [e_{u_t}(x) + e_{z_t}(x)]$$

Maximizing the Log Likelihood

- Assuming Gaussian distributions

$$\log p(x_{0:t} \mid z_{1:t}, u_{1:t})$$
$$= \text{const.} - \frac{1}{2} e_p(x) - \frac{1}{2} \sum_t [e_{u_t}(x) + e_{z_t}(x)]$$

- Maximizing the log likelihood leads to

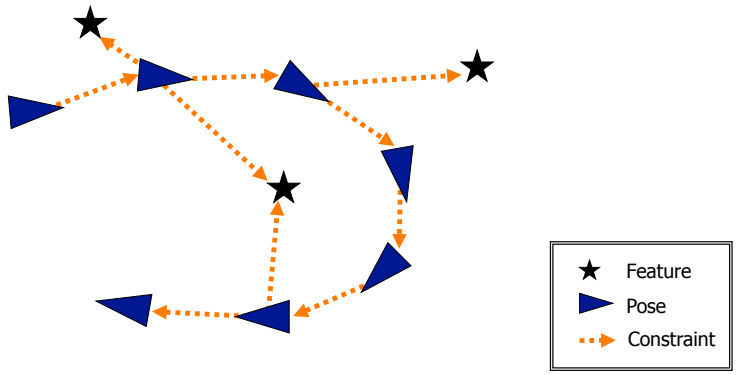
$$\text{argmax} \log p(x_{0:t} \mid z_{1:t}, u_{1:t})$$
$$= \text{argmin}\, e_p(x) + \sum_t [e_{u_t}(x) + e_{z_t}(x)]$$

Minimizing the Squared Error is Equivalent to Maximizing the Log Likelihood of Independent Gaussian Distributions

with individual error terms for the motions, measurements, and prior:

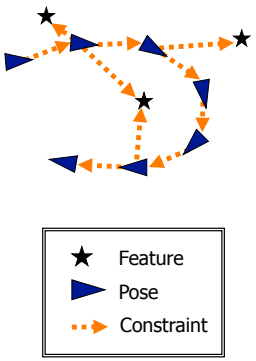
$$\text{argmax} \log p(x_{0:t} \mid z_{1:t}, u_{1:t})$$
$$= \text{argmin}\, e_p(x) + \sum_t [e_{u_t}(x) + e_{z_t}(x)]$$

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)

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

robot landmark

Landmarks Observation

- Expected observation (x-y sensor)

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

robot landmark

- Error function

$$\begin{aligned} \mathbf{e}_{il}(\mathbf{x}_i, \mathbf{x}_l) &= \hat{\mathbf{z}}_{il} - \mathbf{z}_{il} \\ &= \mathbf{X}_i^{-1} \begin{pmatrix} \mathbf{x}_l \\ 1 \end{pmatrix} - \mathbf{z}_{il} \end{aligned}$$

Bearing Only Observations

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

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

robot landmark robot-landmark angle robot orientation

Bearing Only Observations

- Observation function

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

robot, landmark, robot-landmark angle, robot orientation

- Error function

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

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

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

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

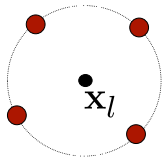
- What is the rank of $\mathbf{H}_{ij}$ for a 2D landmark-pose constraint?
  - The blocks of $\mathbf{J}_{ij}$ are a 2x3 matrices
  - $\mathbf{H}_{ij}$ cannot have more than rank 2

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

- 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

Where is the Robot?

- Robot observes one landmark (x,y)
- Where can the robot be relative to the landmark?

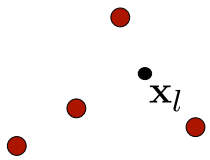


The robot can be somewhere on a circle around the landmark

It is a 1D solution space (constrained by the distance and the robot's orientation)

Where is the Robot?

- Robot observes one landmark (bearing-only)
- Where can the robot be relative to the landmark?



The robot can be anywhere in the x-y plane

It is a 2D solution space (constrained by the robot's orientation)

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 $\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$, we solve

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

What is the effect of that?

Levenberg Marquardt Idea

- Damping factor for $\mathbf{H}$
- $(\mathbf{H} + \lambda\mathbf{I})\Delta\mathbf{x} = -\mathbf{b}$
- The damping factor $\lambda\mathbf{I}$ makes the system positive definite
- Weighted sum of Gauss Newton and Steepest Descent

Simplified Levenberg Marquardt

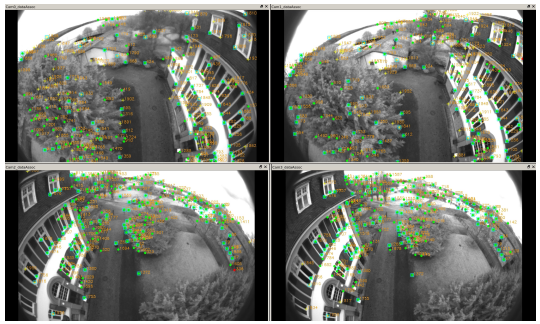
- Damping to regulate the convergence using backup/restore actions

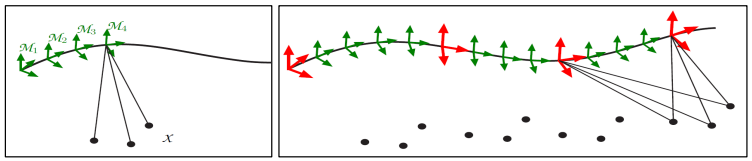
```
x: the initial guess
while (! converged)
  λ = λ_init
  <H,b> = buildLinearSystem(x);
  E = error(x)
  x_old = x;
  Δx  = solveSparse( (H + λ I) Δx = -b);
  x += Δx;
  If (E < error(x)){
    x  = x_old;
    λ *= 2;
} else { λ /= 2; }
```


Bundle Adjustment

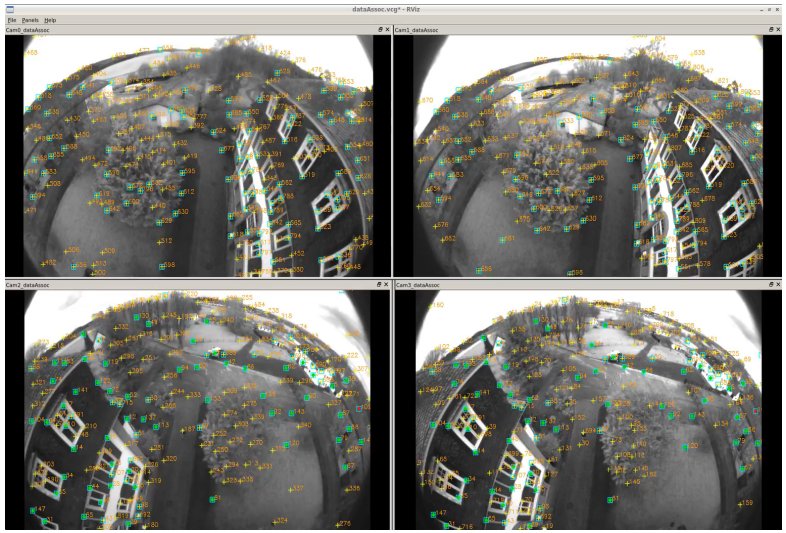
- 3D reconstruction based on images taken at different viewpoints
- Minimizes the reprojection error
- Often uses Levenberg Marquardt
- Developed in photogrammetry during the 1950ies

UAV Example





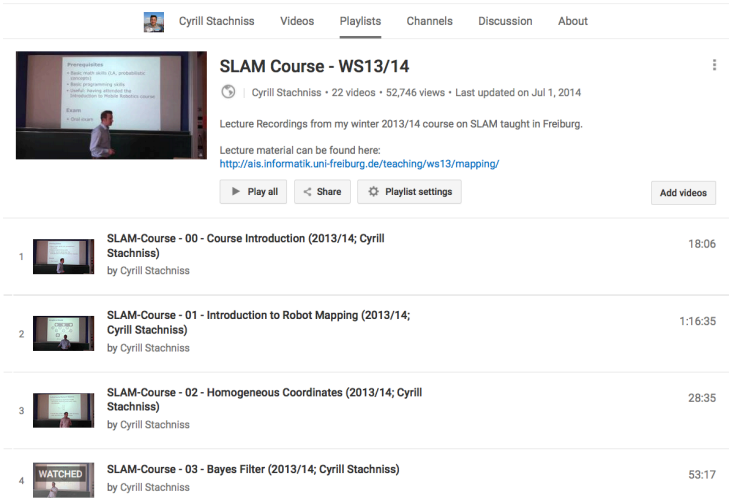
UAV Example



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)

YouTube Lectures



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

Further Reading

Least Squares SLAM

- Grisetti, Kümmerle, Stachniss, Burgard: "A Tutorial on Graph-based SLAM", 2010
- Triggs et al. "Bundle Adjustment — A Modern Synthesis"

Thank you for your attention!

Slide Information

- These slides have been created by Cyrill Stachniss and Giorgio Grisetti evolving from different courses and tutorials we taught over the years between 2010 and 2015.
- 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, 2015
cyrill.stachniss@igg.uni-bonn.de