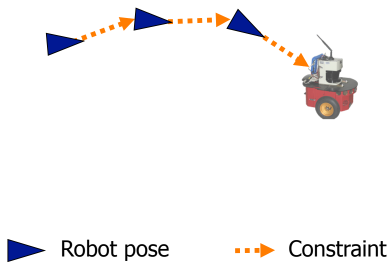
Robot Mapping

Least Squares SLAM Revisited & Hierarchical Approach to Least Squares SLAM

Cyrill Stachniss

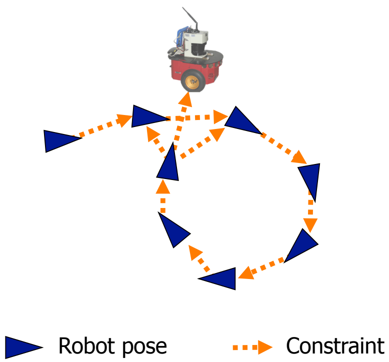
Graph-Based SLAM

- 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

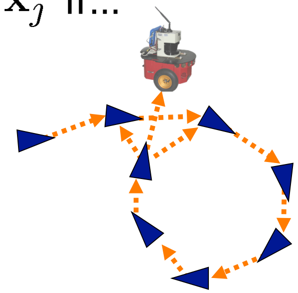


Idea of Graph-Based SLAM

- 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

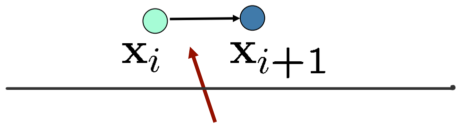
The Graph

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



Create an Edge If… (1)

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



The edge represents the **odometry** measurement

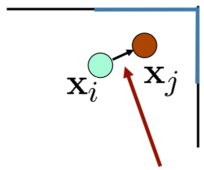
Create an Edge If… (2)

- …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

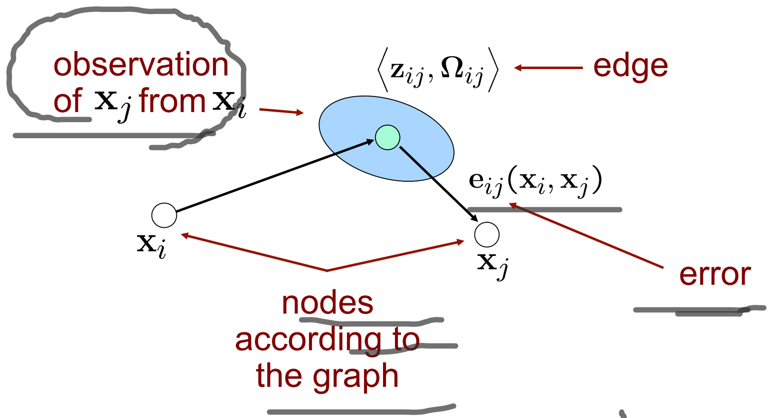
$$(\mathbf{X}_i^{-1}\mathbf{X}_{i+1})$$

- Observation-Based edge

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

How node i sees node j

Pose Graph



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

The Error Function

- Error function for a single constraint

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

measurement

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

- 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

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

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

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 $\mathbf{H}$:

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

- 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

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

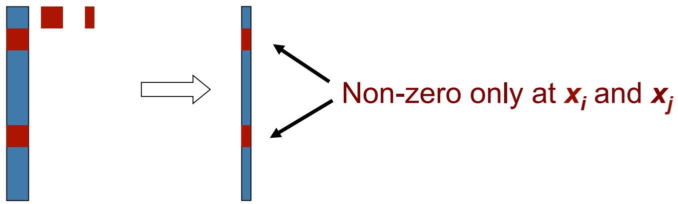
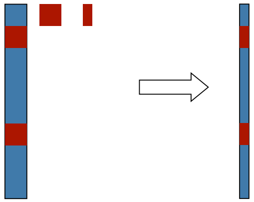


Illustration of the Structure

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



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

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

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

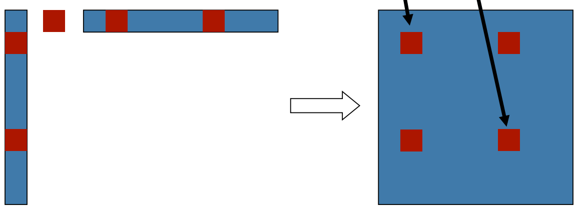
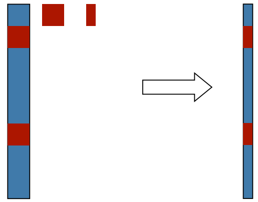


Illustration of the Structure

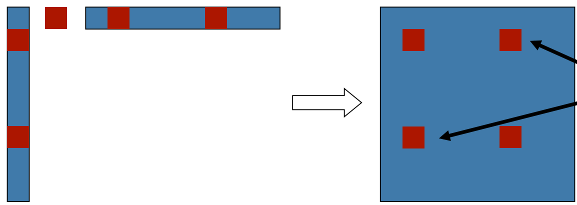
$\mathbf{b}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$



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

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

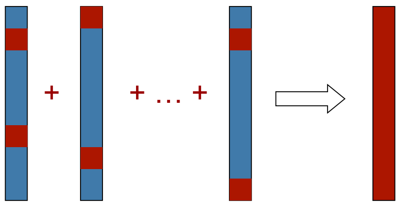
$\mathbf{H}_{ij} = \mathbf{J}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}$



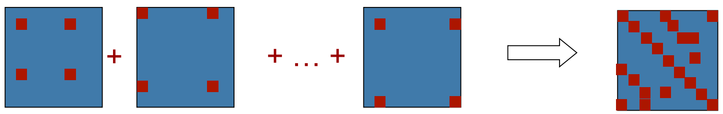
... and at the blocks *ij*,*ji*

Illustration of the Structure

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



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



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:

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

- System matrix:

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

Building the Linear System

For each constraint:

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

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

- Update the coefficient vector:

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

- Update the system matrix:

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

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

Algorithm

optimize($\mathbf{x}$):
  while (!*converged*)
    $(\mathbf{H}, \mathbf{b}) = \text{buildLinearSystem}(\mathbf{x})$
    $\Delta\mathbf{x} = \text{solveSparse}(\mathbf{H}\Delta\mathbf{x} = -\mathbf{b})$
    $\mathbf{x} = \mathbf{x} + \Delta\mathbf{x}$
  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”**

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

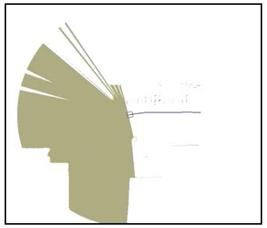
constraint that sets ***dx_1=0***

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 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?

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

	Marginalization	Conditioning
	$p(\boldsymbol{\alpha}) = \int p(\boldsymbol{\alpha},\boldsymbol{\beta})\,d\boldsymbol{\beta}$	$p(\boldsymbol{\alpha} \mid \boldsymbol{\beta}) = p(\boldsymbol{\alpha},\boldsymbol{\beta})/p(\boldsymbol{\beta})$
Cov. Form	$\mu = \mu_\alpha$ $\Sigma = \Sigma_{\alpha\alpha}$	$\mu' = \mu_\alpha + \Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}(\boldsymbol{\beta} - \mu_\beta)$ $\Sigma' = \Sigma_{\alpha\alpha} - \Sigma_{\alpha\beta}\Sigma_{\beta\beta}^{-1}\Sigma_{\beta\alpha}$
Info. Form	$\eta = \eta_\alpha - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\eta_\beta$ $\Lambda = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\beta\alpha}$	$\boldsymbol{\eta}' = \boldsymbol{\eta}_\alpha - \Lambda_{\alpha\beta}\boldsymbol{\beta}$ $\Lambda' = \Lambda_{\alpha\alpha}$

Courtesy: R. Eustice

Uncertainty

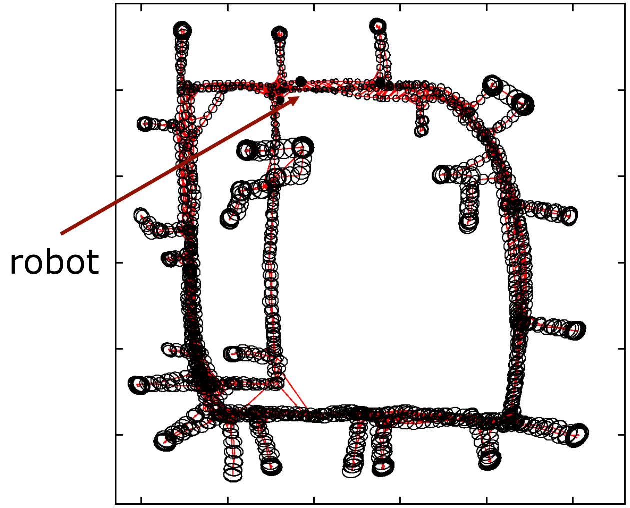
- $\mathbf{H}$ is the information matrix (given the linearization point)
- Inverting $\mathbf{H}$ results in a (dense) covariance matrix
- The diagonal blocks of the covariance matrix represent the (absolute) uncertainties of the corresponding variables

Relative Uncertainty

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

- Construct the matrix $\mathbf{H}$
- Suppress the rows and the columns of $\mathbf{x}_i$ (="fixes" 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

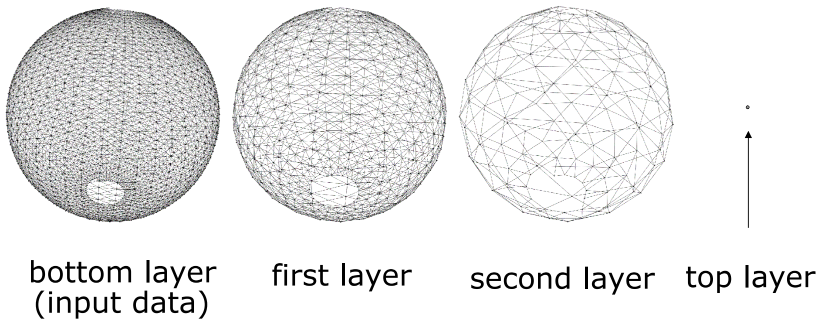


Does all that run online?

Does all that run online?

... it depends on the size of the graph...

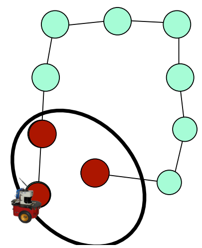
Hierarchical Pose-Graph



"There is no need to optimize the whole graph when a new observation is obtained"

Motivation

- Front-end seeks for loop-closures
- Requires to compare observations to all previously obtained ones
- In practice, limit search to areas in which the robot is likely to be
- This requires to know **in which parts of the graph to search for data associations**

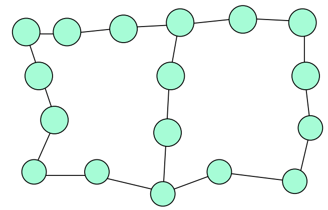


Hierarchical Approach

- **Insight:** to find loop closures, one does not need the perfect global map
- **Idea:** correct only the core structure of the scene, not the overall graph
- The hierarchical pose-graph is a sparse approximation of the original problem
- It exploits the facts that in SLAM
  - Robot moved through the scene and it not "teleported" to locations
  - Sensors have a limited range

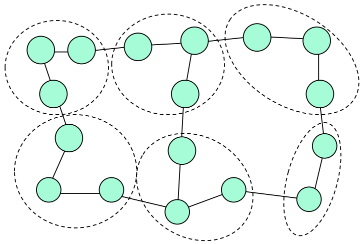
Key Idea of the Hierarchy

- Input is the dense graph



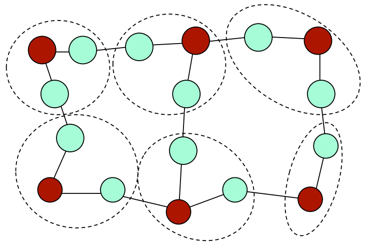
Key Idea of the Hierarchy

- Input is the dense graph
- Group the nodes of the graph based on their local connectivity



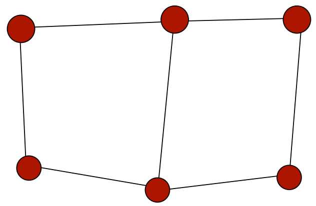
Key Idea of the Hierarchy

- Input is the dense graph
- Group the nodes of the graph based on their local connectivity
- For each group, select one node as a "representative"



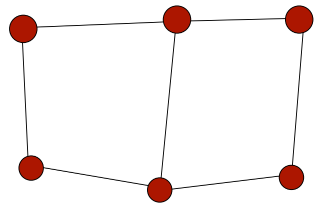
Key Idea of the Hierarchy

- The representatives are the nodes in a new sparsified graph (upper level)



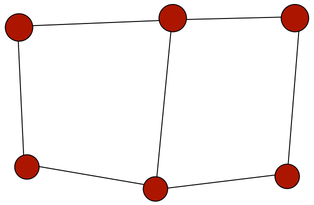
Key Idea of the Hierarchy

- The representatives are the nodes in a new sparsified graph (upper level)
- Edges of the sparse graph are determined by the connectivity of the groups of nodes
- The parameters of the sparse edges are estimated via local optimization



Key Idea of the Hierarchy

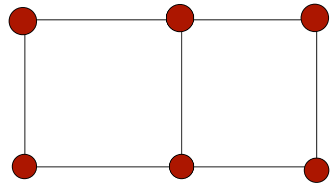
- The representatives are the nodes in a new sparsified graph (upper level)
- Edges of the sparse graph are determined by the connectivity of the groups of nodes
- The parameters of the sparse edges are estimated via local optimization



Process is repeated recursively

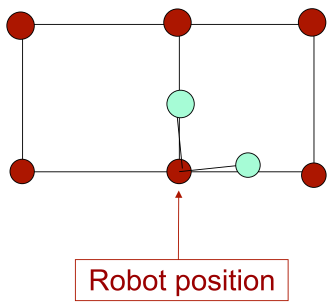
Key Idea of the Hierarchy

- Only the upper level of the hierarchy is optimized completely



Key Idea of the Hierarchy

- Only the upper level of the hierarchy is optimized completely
- The changes are propagated to the bottom levels only close to the current robot position
- Only this part of the graph is relevant for finding constraints

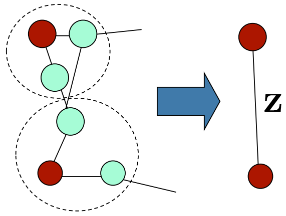


Construction of the Hierarchy

- When and how to generate a new group?
  - A (simple) distance-based decision
  - The first node of a new group is the representative
- When to propagate information downwards?
  - Only when there are inconsistencies
- How to construct an edge in the sparsified graph?
  - Next slides
- How to propagate information downwards?
  - Next slides

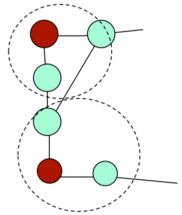
Determining Edge Parameters

- Given two connected groups
- How to compute a virtual observation $\mathbf{z}$ and the information matrix Ω for the new edge?



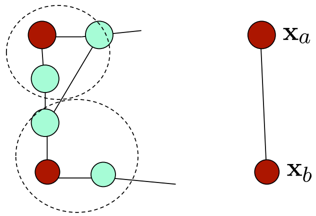
Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest



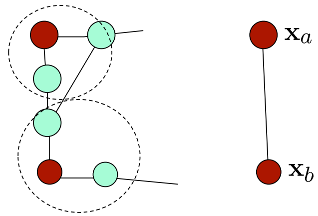
Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest
- The observation is the relative transformation between the two representatives



Determining Edge Parameters

- Optimize the two sub-groups jointly but independently from the rest
- The observation is the relative transformation between the two representatives
- The information matrix is computed from the diagonal block of the matrix ***H***

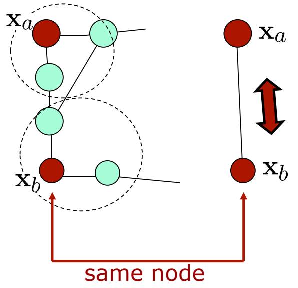


Inverse of the *[b,b]* block of ***H***

$$\Omega_{ab} = (\mathbf{H}^{-1}_{[b,b]})^{-1}$$

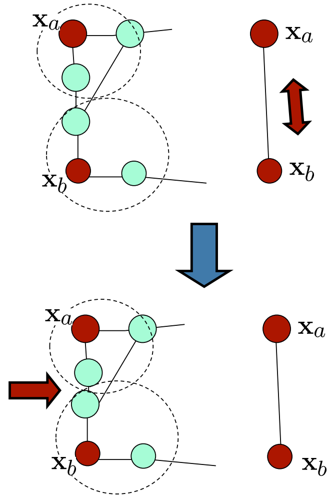
Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level



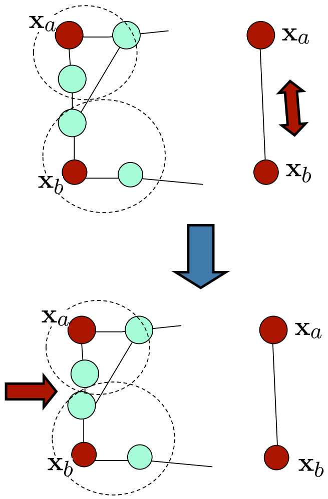
Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level
- Information is propagated downwards by transforming the group at the lower level using a rigid body transformation



Propagating Information Downwards

- All representatives are nodes from the lower (bottom) level
- Information is propagated downwards by transforming the group at the lower level using a rigid body transformation
- Only if the lower level becomes inconsistent, optimize at the lower level



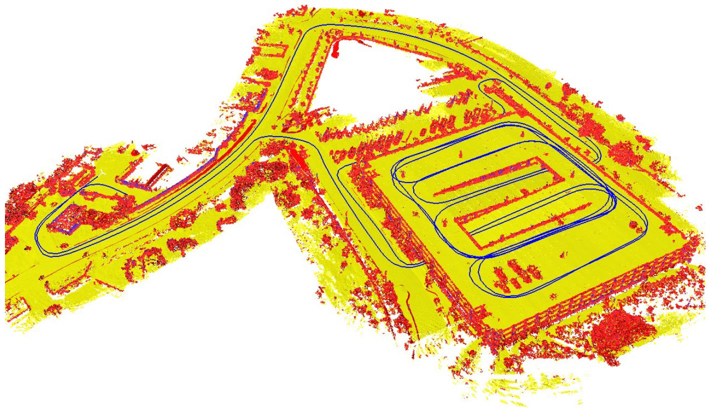
For the Best Possible Map...

- Run the optimization on the lowest level (at the end)
- For offline processing with all constraints, the hierarchy helps convergence faster in case of large errors
- In this case, one pass up the tree (to construct the edges) followed by one pass down the tree is sufficient

Stanford Garage

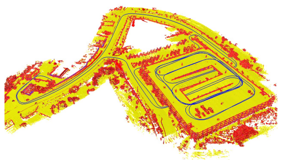
- Parking garage at Stanford University
- Nested loops, trajectory of ~7,000m

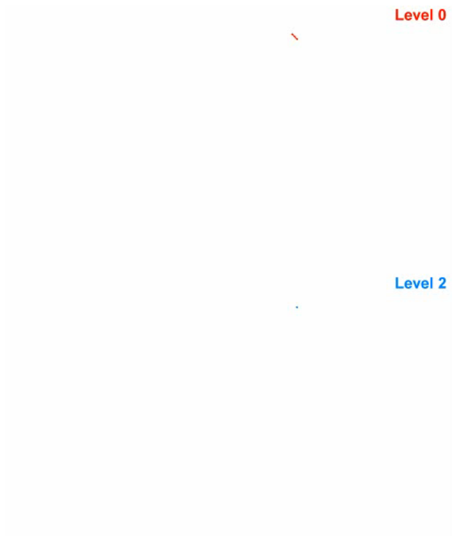
Stanford Garage Result



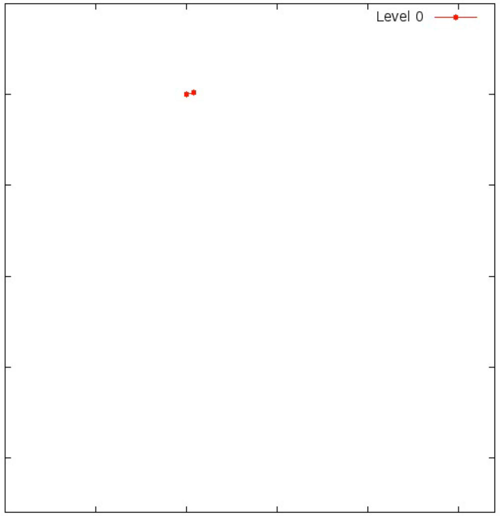
- Parking garage at Stanford University
- Nested loops, trajectory of ~7,000m

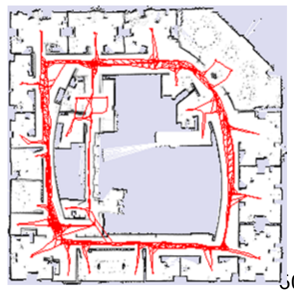
Stanford Garage Video





Intel Research Lab Video





Consistency

- How well does the top level in the hierarchy represent the original input?

Consistency

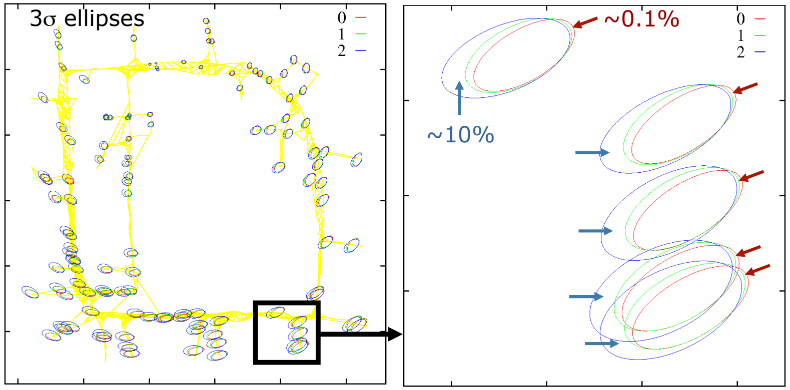
- How well does the top level in the hierarchy represent the original input?
- Probability mass of the marginal distribution in the highest level vs. the one of the true estimate (original problem, lowest level)

	Prob. mass not covered	Prob. mass outside
Intel	0.10%	10.18%
W-10000	2.53%	24.05%
Stanford	0.01%	7.88%
Sphere	2.75%	10.21%

low risk of becoming overly confident

one does not ignore too much information

Consistency



- Red: overly confident (~0.1% prob. mass)
- Blue: under confident (~10% prob. mass)

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
- Hierarchical pose-graph for computing approximate solutions online

Literature

Least Squares SLAM

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

Hierarchical Approach

- Grisetti, Kümmerle, Stachniss, Frese, and Hertzberg: "Hierarchical Optimization on Manifolds for Online 2D and 3D Mapping"
- Code: http://openslam.org/hog-man.html