Mobile Robotics, Localization: Simultaneous Localization And Mapping (SLAM)

Material based on the book Probabilistic Robotics (Thrun, Burgard, Fox) [PR]; Chapter 10, 11, 13

Part of the material is based on lectures from Cyrill Stachniss and Giorgio Grisetti

Summary

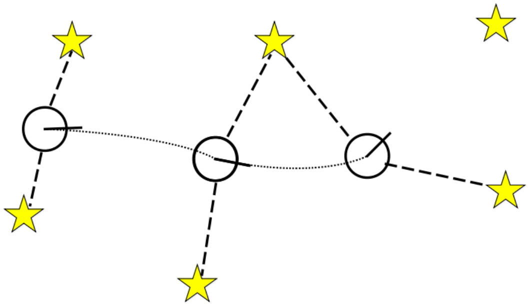
- Introduction to SLAM [Chapter 10.1]
- Grid-based SLAM [Chapter 13.1 – 13.3, partly and paper Grisetti et al., 2007]
- Graph Based Slam [Chapter 11.1 - 11.3]

Intro to SLAM

- Compute the robot's pose and the map of the environment at the same time
- **Localization**: map is given, compute robot pose
  - Markov Localization
  - EKF localization
  - Monte Carlo Localization
- **Mapping**: pose is given, build the map
  - mapping with known poses and occupancy grid
- **SLAM**: **Simultaneous** Localization And Mapping

Localization example

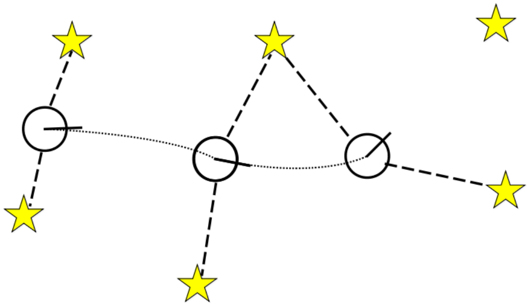
◇ Estimate the robot's poses given landmark



Courtesy Burgard, Fox, Thrun

Mapping example

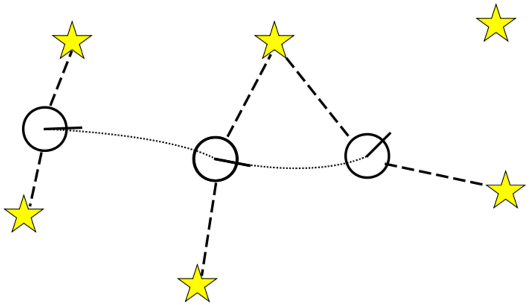
- Estimate the landmarks given the robot's pose



Courtesy Burgard, Fox, Thrun

SLAM example

- Estimate the robot's poses and the landmarks at the same time

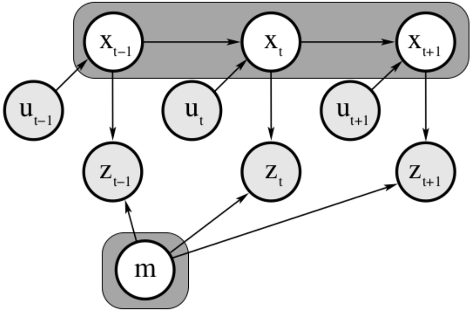


Courtesy Burgard, Fox, Thrun

The full SLAM problem

◇ Compute the posterior over **the whole path of the robot** and the map

$$P(x_{0:T}, m | u_{1:T}, u_{1:T})$$

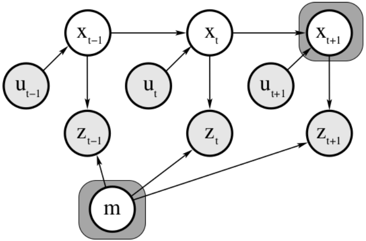


Graphical model for the **full** SLAM Raoproblem, source [PR]

The online SLAM problem

- Compute the posterior over the **current** robot pose and map

$$P(x_{t+1}, m | u_{1:t+1}, u_{1:t+1})$$



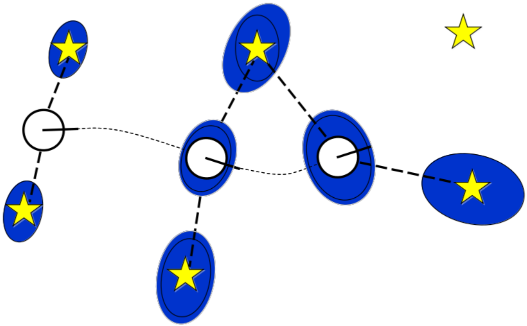
Graphical model for the **online** SLAM problem, source [PR]

$$P(x_{t+1}, m | u_{1:t+1}, u_{1:t+1}) = \int \int \cdots \int p(x_{0:t+1}, m | z_{1:t+1}, u_{1:t+1}) dx_0 dx_1 \cdots dx_t$$

Example of SLAM

Fastslam approach in indoor environment, courtesy Dirk Haehnel

SLAM is a hard problem I



Courtesy Burgard, Fox, Thrun

- Map and pose estimates correlate

SLAM is a hard problem II

Courtesy Burgard, Fox, Thrun

- known vs. unknown correspondences (data association)

Traditional paradigms for SLAM

- Kalman Filter and EKF
- Particle Filters
  - Rao-Blackwellized particle filters
  - FastSlam
- Graph-based
  - Least Squares formulations
  - very popular

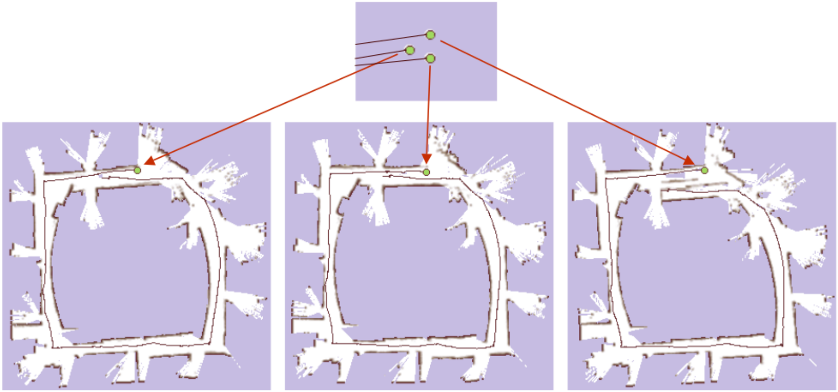
Grid-Based Mapping with Rao-Blackwellized PF, intro

- SLAM for grid maps based on a specific type of PF
- Uses a special type of PF: Rao-Blackwellized

$$p(x_{0:t}, m | z_{1:t}, u_{1:t}) = p(x_{0:t} | z_{1:t}, u_{1:t}) p(m | x_{1:t}, z_{1:t})$$

- decompose joint posterior of map and path in the product of
  - path posterior **given** stream of data
  - map posterior **given** path
- Each particle represent a possible trajectory of the robot
- Each particle maintains its own map
- Each particle updates map as if it was performing "mapping with known poses"

Grid-Based Mapping with Rao-Blackwellized PF, Illustration



Each particle carries its own map, weights are computed based on likelihood of measurements given the particles' own map, source [PR]

Grid-Based Mapping with Rao-Blackwellized PF, Improvement

- ◇ Direct application of previous approach may fall short if we have noisy motion model
- ◇ Need too many samples to make this work in large maps
- ◇ **Idea** improve pose estimate **before** applying the particle.
- ◇ Locally adjust the pose using scan matching
  - align laser scans to improve pose estimate (scan matching)

$$x_t^* = \arg\max_{x_t}\{p(z_t|x_t, m_{t-1})p(x_t|u_{t-1}, x_{t-1}^*)\}$$

- ◇ mathematical issue: we use observation in proposal and corrections
  - use scan matching (and not PF) on chunks of observations
  - correcting with particle filters the map chunks
  - can be seen ad an ad-hoc improved proposal distribution

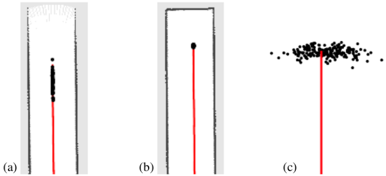
Grid-Based Mapping with Rao-Blackwellized PF, better solution

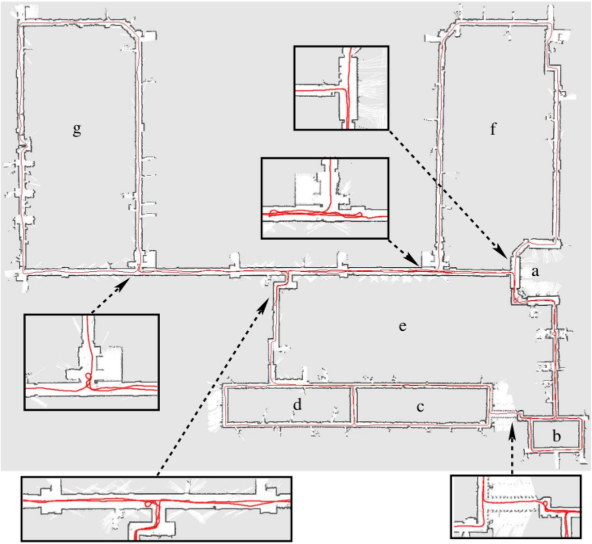
- Compute an improved proposal that considers the most recent observation

$$x_t^{[k]} \sim p(x_t | x_{1:t-1}^{[k]}, u_{1:t}, z_{1:t})$$

- **Key insights**:
  - Proposal distribution considers the accuracy of robot's sensor combining accurate local information from observations (i.e., Laser) and global but inaccurate information from odometry.
  - Adaptive re-sampling technique which maintains a variety of particles (reducing risk of particle depletion)
- **Main benefits**
  - More precise sampling
  - More accurate maps
  - Less particle needed

Grid-Based Mapping with Rao-Blackwellized PF, visualization





(Top) effect of improved proposal distribution.

(Right) MIT-Killian court map, total travelled distance about 2 Km, 80 particles used. Courtesy of Grisetti, Stachniss, Burgard

Graph-Based SLAM

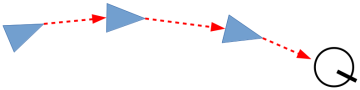
- ◊ Least Squares approach to SLAM using pose graph
- ◊ Very popular, state-of-the-art approach for SLAM
- ◊ Reduce everything to the pose graph
  - marginalize out landmarks and observations
  - sparse graph structure
  - flexible (can handle several types of observation)

Reminder: Least Squares

- Approach to solve **overdetermined** problems
  - more equations than unknowns
- minimizes the **sum of squared errors** in the equations
- standard approach to a large set of problems

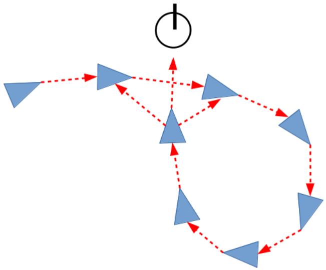
Pose Graph

◇ **Constraints** connect the **poses** of the robot while it is moving
◇ **Poses** are recorded at given intervals and are the nodes of the graphs,
◇ **Constraints** encodes **uncertain** relationships between poses and are the edges of the pose graph



Pose Graph II

◊ Observing previously seen areas generate constraints between **non-successive** poses



Main idea for Graph-Based SLAM I

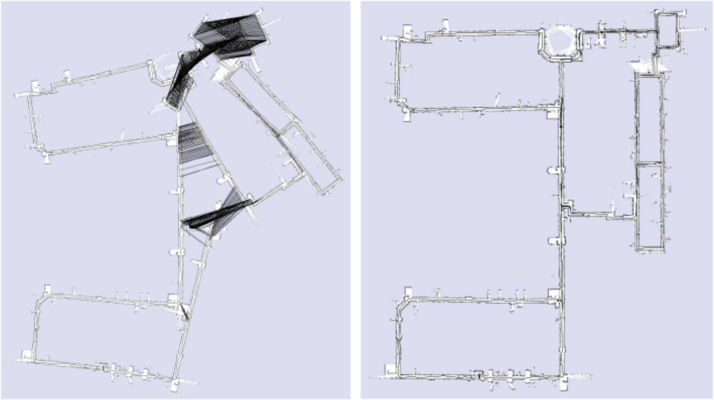
◇ Use a graph to encode the estimation problem

- **Node**: robot poses
- **Edge**: spatial, uncertain constraints between two poses

◇ **Goal**: build the graph and find a node configuration (i.e., pose estimate) that minimizes the error introduced by the constraints

Main idea for Graph-Based SLAM II

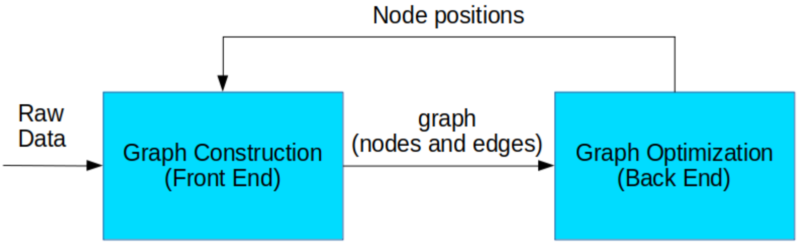
i) build pose graph based on motion and observation model, ii) optimize the pose-graph, iii) render map based on pose estimation



Courtesy of Grisetti, Kümmerle, Stachniss and Burgard

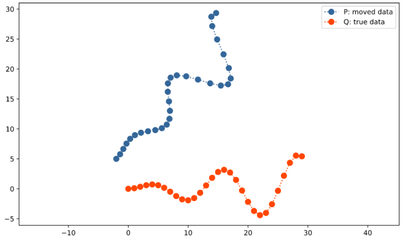
The overall SLAM system

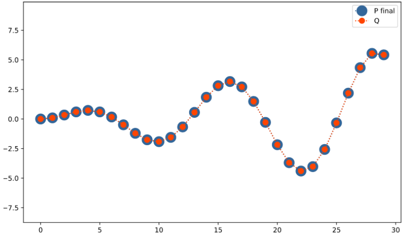
- Interplay of **front-end** and **back-end**
- Map reduces search space to build correct constraints
- We focus on optimization (**back-end**)



Font-end: ICP

- ◇ Iterative Closest Point: align two point clouds and return the transformation (Rotation + Tranlsation)
- ◇ Transformation is used to build the constraints for the pose-graph
- ◇ ICP:
  - Find correspondences (closest point)
  - Given correspondences, compute transformation
  - Repeat

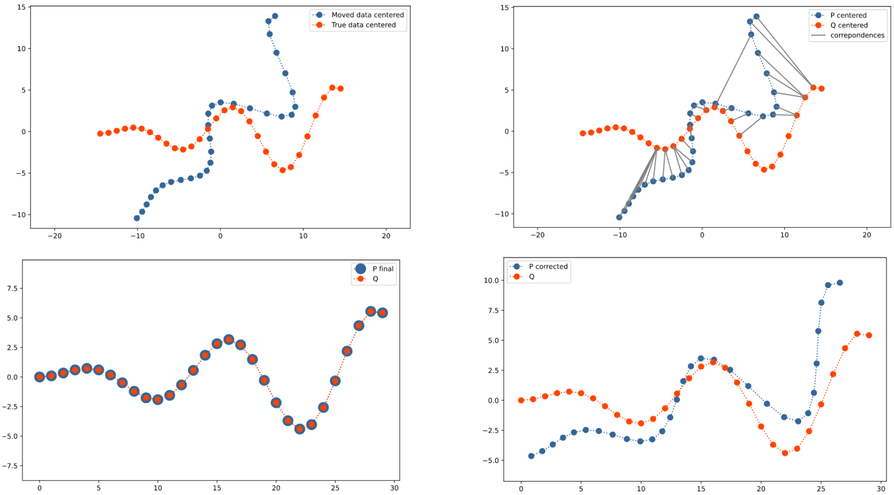




Courtesy of Igor Bogoslakvskyi

Short description of ICP

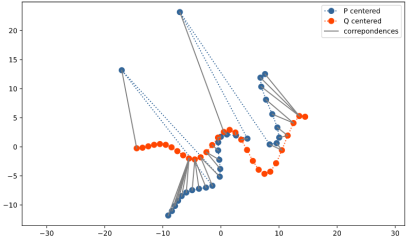
Mobile Robotics, Localization: Simultaneous Localization And Mapping (SLAM)



Courtesy of Igor Bogoslakvskyi

ICP variants

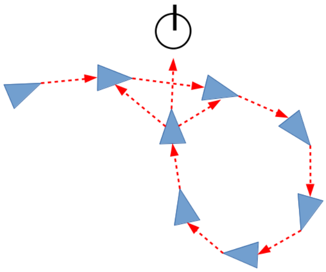
- Use point to plane distance
- Avoid outliers when computing transformation
- For more details: Jupyter notebook by Igor Bogoslakvskyi



Courtesy of Igor Bogoslakvskyi

The Pose Graph

- n nodes $x = x_{1:n}$
  - x_i pose of the robot at time t_i
  - constraint (edge) between x_i and x_j exists if
    - the robot **moves** from x_i to x_{i+1}, odometry measurement (or ICP)
    - the robot **observes** the same part of the environment, **virtual measurement**



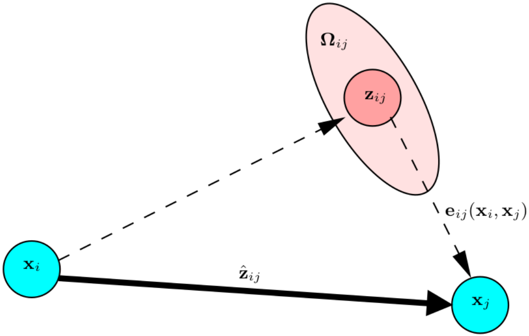
Transformations and Homogeneous Coordinates

- Using homogeneous coordinates we can express rotations and translation with a single matrix
- Odometry edge: $(X_i^{-1}X_{i+1})$
  - relative transformation between x_{i+1} and x_i
  - x_{i+1} as seen from x_i
- Observation edge: $(X_i^{-1}X_j)$
  - relative transformation between x_j and x_i
  - x_j as seen from x_i

The Edge Information Matrices

- Observations are affected by noise
- Information Matrix $\mathbf{\Omega}_{ij}$ for each edge to encode uncertainty
- "Bigger" $\mathbf{\Omega}_{ij}$ matter more in the optimization

Pose Graph Visualization



Courtesy of Grisetti, Kümmerle, Stachniss and Burgard

$$x^* = \arg\min_x \sum_{ij} e_{ij}^T \mathbf{\Omega}_{ij} e_{ij}$$

Least Square SLAM

- Error function is suitable for least square error minimization

$$x^* = \arg\min_x \sum_{ij} e_{ij}^T(x_i, x_j)\boldsymbol{\Omega}_{ij} e_{ij}(x_i, x_j) = \arg\min_x \sum_k e_k^T(x)\boldsymbol{\Omega}_k e_k(x)$$

- State vector is the concatenation of the robot poses: $x^T = (x_1^T, x_2^T, \ldots, x_n^T)$

The error function

- Error function for single constraint: $e_{ij}(x_i, x_j) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j))$

- Error function for whole state vector: $e_{ij}(x) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j))$

- Error is zero is $Z_{ij} = (X_i^{-1}X_j)$

Gauss-Newton for error minimization

1. Define the error function
2. Linearize the error function
3. Compute its derivative
4. Set derivative to zero
5. Solve linear system
6. Iterate until convergence

Linearizing the Error function

- Approximate error function around an initial guess x using taylor expansion

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

where,

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

- Note: $e_{ij}(x)$ does not depend on full state vector but only on x_i and x_j $\Rightarrow$ Jacobian is sparse

$$\frac{\delta e_{ij}(x)}{\delta x} = \left(0 \dots \frac{\delta e_{ij}(x_i)}{\delta x_i} \dots \frac{\delta e_{ij}(x_j)}{\delta x_j} \dots 0\right)$$

$$\mathbf{J}_{ij} = (0 \dots \mathbf{A}_{ij} \dots \mathbf{B}_{ij} \dots 0)$$

Consequences of Sparsity I

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

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

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

- Sparse structure of $\mathbf{J}_{ij}$ results in sparse structure of $\mathbf{H}_{ij}$
- The structure depends on the adjacency matrix of the pose graph

Consequences of Sparsity II

- An edge contributes to the linear system via b_{ij} and $\mathbf{H}_{ij}$

$$b_{ij}^T = e_{ij}^T \mathbf{\Omega}_{ij} \mathbf{J}_{ij} = e_{ij}^T \mathbf{\Omega}_{ij} \left(0 \ldots \mathbf{A}_{ij} \ldots \mathbf{B}_{ij} \ldots 0\right) = \left(0 \ldots e_{ij}^T \mathbf{\Omega}_{ij} \mathbf{A}_{ij} \ldots e_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} \ldots 0\right)$$

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

$$= \begin{pmatrix} \vdots \\ \mathbf{A}_{ij}^T \\ \vdots \\ \mathbf{B}_{ij}^T \\ \vdots \end{pmatrix} \mathbf{\Omega}_{ij} \left(\ldots \mathbf{A}_{ij} \ldots \mathbf{B}_{ij} \ldots\right) = \begin{pmatrix} \ddots & & & & \\ & \mathbf{A}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{A}_{ij} & \ldots & \mathbf{A}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} & \\ & \vdots & \ddots & \vdots & \\ & \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{A}_{ij} & \ldots & \mathbf{B}_{ij}^T \mathbf{\Omega}_{ij} \mathbf{B}_{ij} & \\ & & & & \ddots \end{pmatrix}$$

The linear system

- Vector of state increments

$$\Delta x^T = (\Delta x_1^T \; \Delta x_2^T \; \dots \; \Delta x_n^T)$$

- Vector of coefficients

$$b^T = (\bar{b}_1^T \; \bar{b}_2^T \; \dots \; \bar{b}_n^T)$$

- Normal equation matrix

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

Building the linear system

- For each constraint:
  - compute error: $e_{ij}(x_i, x_j) = t2v(Z_{ij}^{-1}(X_i^{-1}X_j))$
  - compute blocks of Jacobian

$$\mathbf{A}_{ij} = \frac{\delta e_{ij}(x_i, x_j)}{\delta x_i} \qquad \mathbf{B}_{ij} = \frac{\delta e_{ij}(x_i, x_j)}{\delta x_j}$$

  - update the coefficient vector

$$\bar{b}_i + = e_{ij}^T(x_i, x_j)\mathbf{\Omega}_{ij}\mathbf{A}_{ij} \qquad \bar{b}_j + = e_{ij}^T(x_i, x_j)\mathbf{\Omega}_{ij}\mathbf{B}_{ij}$$

  - update the normal equation matrix

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

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

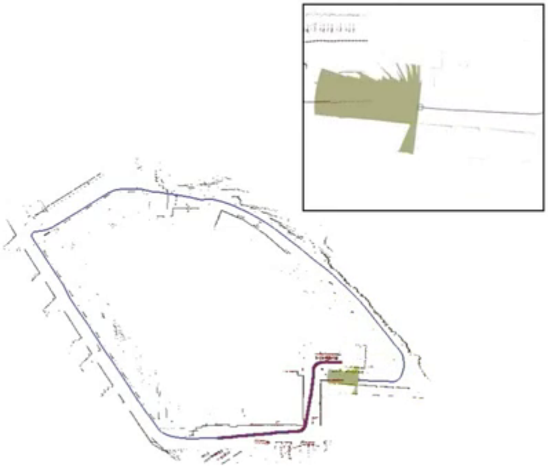
Algorithm

Data: $G(x, E)$
Result: x'
// G is pose graph, x' optimized pose vector
$x' = x$;
while $!converged$ **do**
  $(\mathbf{H}, b) = buildLinearSystem(x')$;
  $\Delta x = solveSparse(\mathbf{H}\Delta x = -b)$;
  $x' = x' + \Delta x$;
end
return x' ;

Role of Prior

- We need to fix the the pose graph to the global frame
- We can do this enforcing a prior knowledge on a node, usually x_0
- Add a constraint for Gaussian estimate about x_0
  - e.g., first pose in the origin, $e(x_0) = t2v(X_0)$

Example of SLAM



Graph-Based SLAM, courtesy Grisetti and Stachniss

Summary

- SLAM: estimate map and robot pose at the same time
  - full SLAM, estimate all poses
  - online SLAM, estimate only last pose
- Can represent SLAM problem using a pose graph
  - node: robot poses
  - edges: constraints between poses (odometry and measurements)
- Graph construction (front-end), typically uses approaches such as ICP
- Graph optimization (back-end), can be efficiently done using Least-Square (Gauss-Newton)
- Efficiency comes from sparsity of matrices (the $\mathbf{H}$ matrix)
- SLAM with pose graphs is one of the state of the art solution for SLAM

References and Further readings

◊ A Tutorial on Graph-Based SLAM, Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss andWolfram Burgard

◊ Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters; Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard IEEE Transactions on Robotics, Volume 23, pages 34-46, 2007