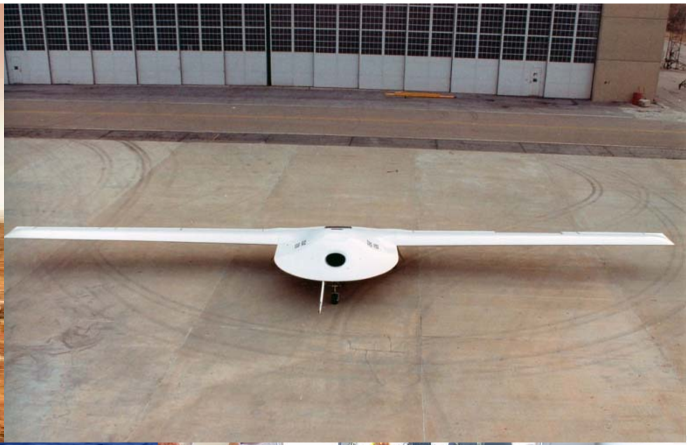
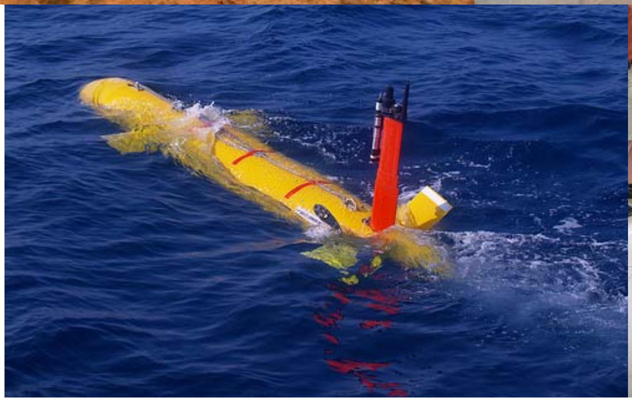
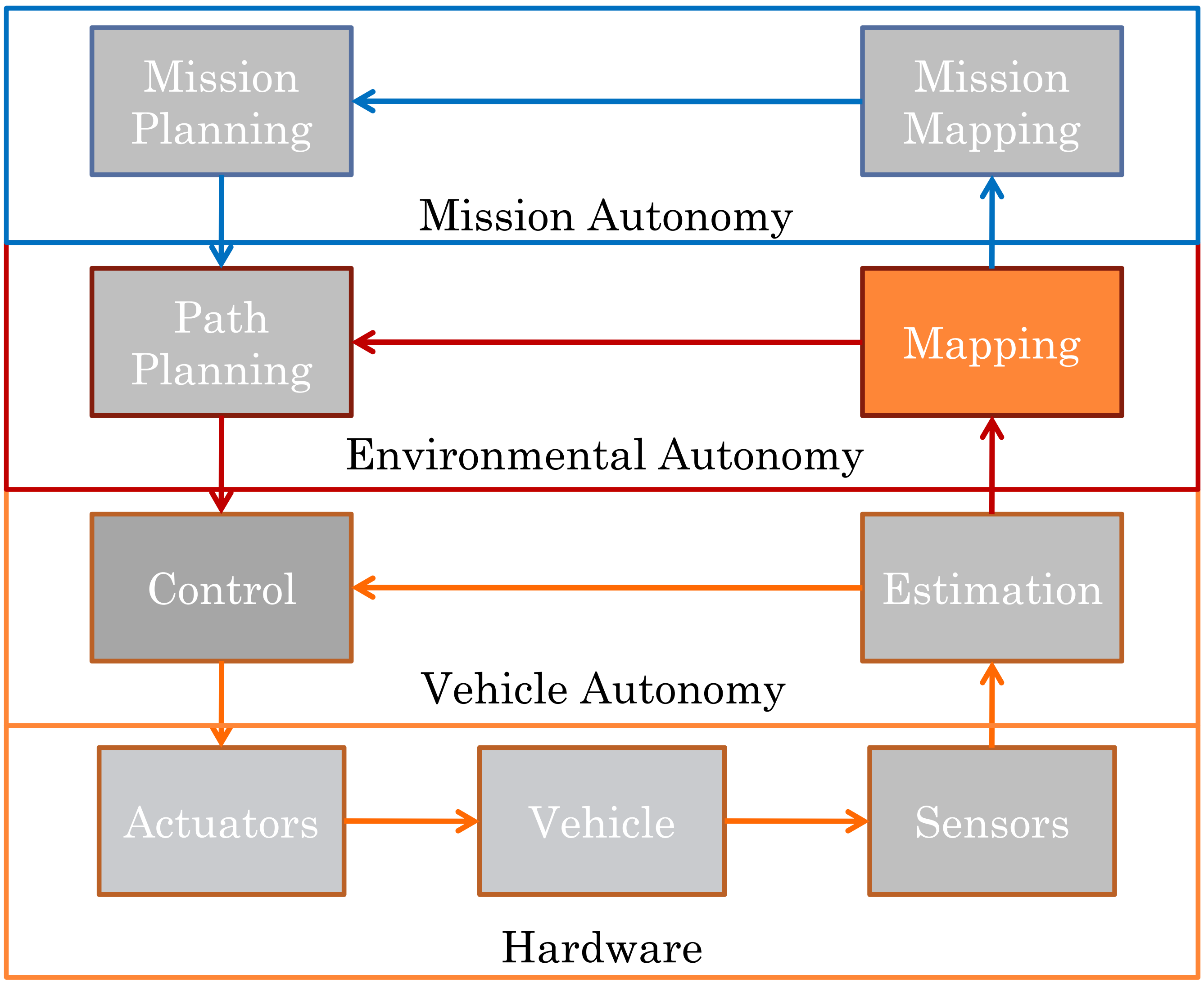


ME 597: Autonomous Mobile Robotics
Section 8 – Mapping III

Prof. Steven Waslander

Components


Mission Planning
Mission Mapping
Mission Autonomy
Path Planning
Mapping
Environmental Autonomy
Control
Estimation
Vehicle Autonomy
Actuators
Vehicle
Sensors
Hardware

Outline

- The GraphSLAM algorithm
  - Derivation of feature based optimization problem
  - Derivation of scan based optimization problem
  - Discussion of solution methods
  - Implementation and Results

Two Main SLAM Approaches

- Online SLAM
  - Filter version of the SLAM problem, maximize

$$p(x_t \mid y_{1:t}, u_{1:t})$$

  - Process new information as it is received
  - Generate current best estimate, rely on Markov assumption and linearity to trust that this is the best you can do, and use the solution in subsequent steps
  - EKF SLAM, FastSLAM Occupancy Grid SLAM, etc.

- Full SLAM
  - Smoothing version of the SLAM problem, maximize

$$p(x_{0:t} \mid y_{1:t}, u_{1:t})$$

  - Store all information as collected, only resolve into poses and map when needed
  - Work on all information, allows for re-linearization during the optimization process
  - Can resolve correspondence as well, allowing for a more robust solution

FULL SLAM PROBLEM

- Full SLAM - Features
  - Simultaneously determine the robot pose history and static feature locations in the environment.

$$x_t^r = \begin{pmatrix} X_t \\ Y_t \\ Z_t \\ \phi_t \\ \theta_t \\ \psi_t \end{pmatrix}, \quad m_i = \begin{pmatrix} m_{i,X} \\ m_{i,Y} \\ m_{i,Z} \end{pmatrix}, \quad m = \begin{pmatrix} m_1 \\ \vdots \\ m_M \end{pmatrix}, \quad x_{0:t} = \begin{pmatrix} x_0^r \\ x_1^r \\ \vdots \\ x_t^r \\ m \end{pmatrix}, \quad x_t = \begin{pmatrix} x_t^r \\ m \end{pmatrix}$$

Robot State at time *t*	i^{th} feature	Feature map	Full state	Full state at time *t*

FULL SLAM PROBLEM

- Available information
  - Inputs and Motion model

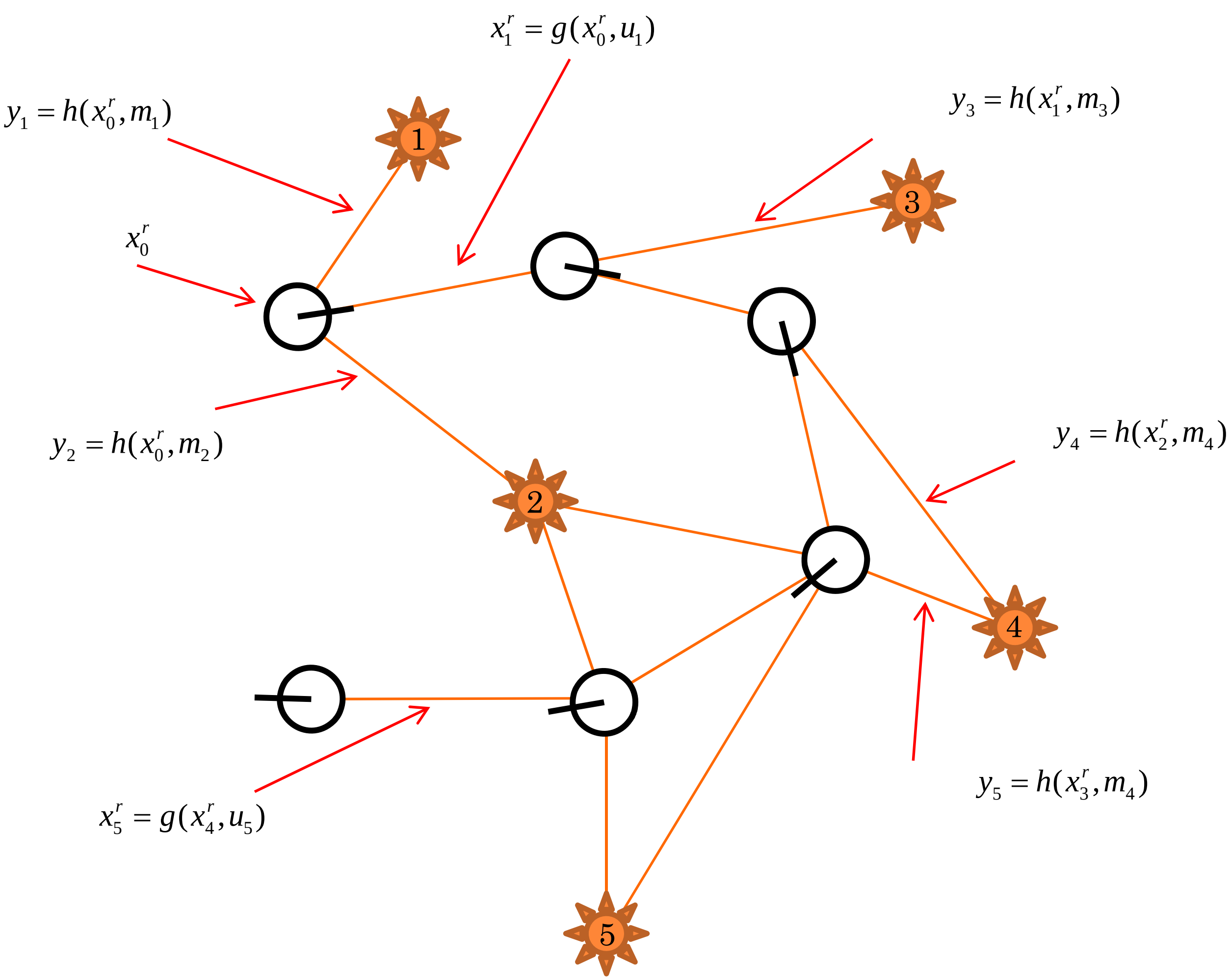
$$u_{0:t}, \quad x_t^r = g(x_{t-1}^r, u_t) + \delta_t$$

  - Measurements and measurement model

$$y_{1:t}, \quad y_t = h(x_t^r, m) + \varepsilon_t$$

  - Again, we'll assume good correspondence information, but this is an important part of GraphSLAM, can include correspondence as part of the optimization

Illustration of the Constraint Graph



GraphSLAM Optimization

- We are interested in finding the maximum likelihood state estimate

$$\max_{x_{0:t}} p(x_{0:t} \mid y_{1:t}, u_{1:t})$$

- Apply Bayes rule to separate out the current measurements

$$\begin{aligned} p(x_{0:t} \mid y_{1:t}, u_{1:t}) &= \eta\, p(y_t \mid x_{0:t}, u_{1:t})\, p(x_{0:t} \mid y_{1:t-1}, u_{1:t}) \\ &= \eta\, p(y_t \mid x_t)\, p(x_{0:t} \mid y_{1:t-1}, u_{1:t}) \end{aligned}$$

GRAPHSLAM OPTIMIZATION

- Next, separate out the motion through factoring of the probabilities of second term, since y_t is not present

$$p(x_{0:t} \mid y_{1:t-1}, u_{1:t})$$
$$= p(x_t^r \mid x_{0:t-1}, u_{1:t})\, p(x_{0:t-1} \mid y_{1:t-1}, u_{1:t})$$
$$= p(x_t^r \mid x_{t-1}, u_t)\, p(x_{0:t-1} \mid y_{1:t-1}, u_{1:t})$$

- These steps we repeat until the beginning of time to get

$$p(x_{0:t} \mid y_{1:t}, u_{1:t}) = \eta\, p(x_0) \prod_{\tau=1}^{t} p(x_\tau^r \mid x_{\tau-1}^r, u_\tau)\, p(y_\tau \mid x_\tau)$$
$$= \eta\, p(x_0) \prod_{\tau=1} \left(p(x_\tau^r \mid x_{\tau-1}^r, u_\tau) \prod_i p(y_\tau^i \mid x_\tau) \right)$$

  - If there is no prior information about the map, use $p(x^r{}_0)$

GraphSLAM Optimization

- We can redefine our optimization problem as

$$\max_{x_{0:t}} p(x_{0:t} \mid y_{1:t}, u_{1:t})$$

⇓

$$\max_{x_{0:t}} \ \eta p(x_0) \prod_{\tau=1} \left(p(x_\tau^r \mid x_{\tau-1}^r, u_\tau) \prod_i p(y_\tau^i \mid x_\tau) \right)$$

⇓

$$\min_{x_{0:t}} \ \text{-ln} \left(\eta p(x_0) \prod_{\tau=1} \left(p(x_\tau^r \mid x_{\tau-1}^r, u_\tau) \prod_i p(y_\tau^i \mid x_\tau) \right) \right)$$

GraphSLAM Optimization

- We can redefine our optimization problem as

$$\min_{x_{0:t}} \text{-ln}\left(\eta p(x_0) \prod_{\tau=1} \left(p(x_\tau^r \mid x_{\tau-1}^r, u_\tau) \prod_i p(y_\tau^i \mid x_\tau) \right) \right)$$

⇓

$$\min_{x_{0:t}} \text{J} = \text{const. - ln}\left(p(x_0) \right)$$

$$- \sum_{\tau=1}^{t} \left(\ln\left(p(x_\tau^r \mid x_{\tau-1}^r, u_t) \right) \right) - \sum_{\tau=1}^{t} \sum_i \ln\left(p(y_t^i \mid x_t) \right)$$

GRAPHSLAM OPTIMIZATION

- The assumption about additive Gaussian noise and disturbances means that the motion and measurement models can be expressed as Gaussian distributions
  - Motion

$$p(x_t^r \mid x_{t-1}^r, u_t) = \eta e^{-\frac{1}{2}\left[x_t^r - g(x_{t-1}^r, u_t)\right]^T R^{-1}\left[x_t^r - g(x_{t-1}^r, u_t)\right]}$$

  - Measurement

$$p(y_t^i \mid x_t) = \eta e^{-\frac{1}{2}\left[y_t^i - h(x_t)\right]^T Q^{-1}\left[y_t^i - h(x_t)\right]}$$

  - Prior $p(x_0) = \eta e^{-\frac{1}{2}\left[x_0 - \mu_0\right]^T \Sigma_0^{-1}\left[x_0 - \mu_0\right]}$

$$\mu_0 = 0, \;\; \Sigma_0 = 0, \;\; \Sigma_0^{-1} = \infty I$$

GraphSLAM Optimization

- The negative log likelihoods therefore all take the *Mahalonobis distance* form

  - Motion

$$-\ln p(x_t^r \mid x_{t-1}^r, u_t) = const. + \left[x_t^r - g(x_{t-1}^r, u_t) \right]^T R^{-1} \left[x_t^r - g(x_{t-1}^r, u_t) \right]$$

  - Measurement

$$-\ln p(y_t^i \mid x_t) = const. + \left[y_t^i - h(x_t) \right]^T Q^{-1} \left[y_t^i - h(x_t) \right]$$

  - Prior

$$-\ln p(x_0) = const. + \left[x_0 - \mu_0 \right]^T \Sigma_0^{-1} \left[x_0 - \mu_0 \right]$$

GraphSLAM Optimization

- The final form of the optimization is now

$$\min_{z_{0:t}} \ J = \text{const.} + \left[x_0 - \mu_0\right]^T \Sigma_0^{-1} \left[x_0 - \mu_0\right]$$
$$+ \sum_{\tau=1}^{t} \left[x_t^r - g(x_{t-1}^r, u_t)\right]^T R^{-1} \left[x_t^r - g(x_{t-1}^r, u_t)\right]$$
$$+ \sum_{\tau=1}^{\tau=1} \sum_{i} \left[y_t^i - h(x_t)\right]^T Q^{-1} \left[y_t^i - h(x_t,)\right]$$

- This is an unconstrained nonlinear optimization problem, which now needs to be solved somehow.
  - There is a lot of structure to the problem, because of the sequential nature of the motion constraints and the measurement of features at only a few instances in time.

Graph constraints

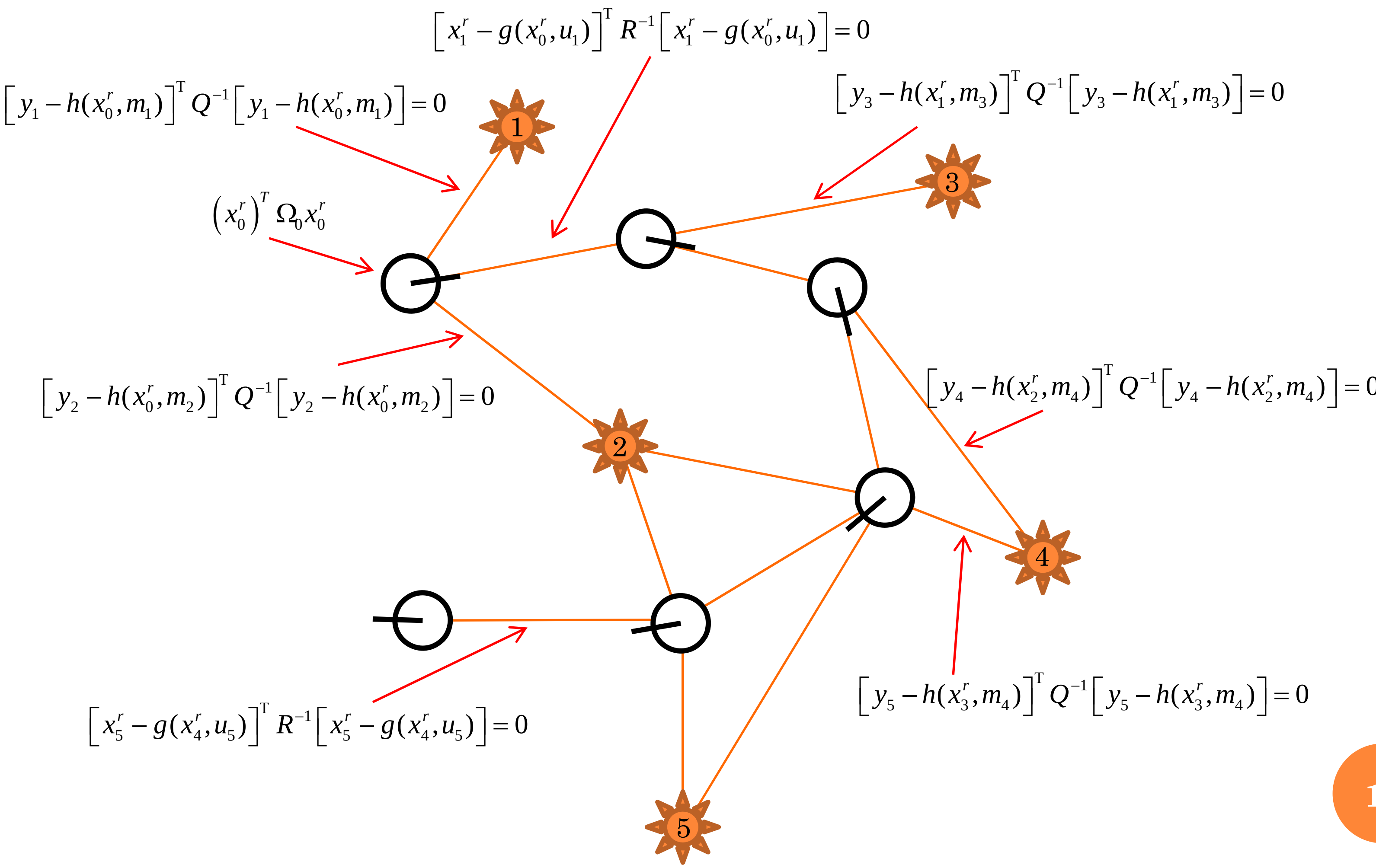
- The constraints on the graph can now be thought of in a least squares sense.
  - Over-determined set of constraints, optimization aims to minimize the total violation of the full set of constraints
  - Can be considered a weighted distance minimization
    - Errors minimized together based on inverse of covariance weighting (information matrix)
    - Motion

$$\left[x_t^r - g(x_{t-1}^r, u_t)\right]^{\mathrm{T}} R^{-1}\left[x_t^r - g(x_{t-1}^r, u_t)\right] = 0$$

    - Measurement

$$\left[y_t - h(x_t, c_t)\right]^{\mathrm{T}} Q^{-1}\left[y_t - h(x_t, c_t)\right] = 0$$

Illustration of the Constraint Graph



GraphSLAM Optimization

- For standard nonlinear optimization packages, you must provide
  - Cost function

$$J = \text{const.} + \left[x_0 - \mu_0\right]^T \Sigma_0^{-1} \left[x_0 - \mu_0\right] + \sum_{\tau=1}^{t} \left[x_t^r - g(x_{t-1}^r, u_t)\right]^T R^{-1} \left[x_t^r - g(x_{t-1}^r, u_t)\right]$$

$$+ \sum_{\tau=1}^{\tau=1} \sum_i \left[y_t^i - h(x_t)\right]^T Q^{-1} \left[y_t^i - h(x_t,)\right]$$

  - Gradient function

$$\frac{\partial J}{\partial x_0} = \left[x_0 - \mu_0\right]^T \Sigma_0^{-1} + \left[x_1^r - g(x_0^r, u_t)\right]^T R^{-1} \frac{-\partial}{\partial x_0}\left[g(x_0^r, u_t)\right]$$

$$\frac{\partial J}{\partial x_t^r} = -\left[x_{t+1}^r - g(x_t^r, u_{t+1})\right]^T R^{-1} \frac{\partial}{\partial x_t^r}\left[g(x_t^r, u_{t+1})\right] + \left[x_t^r - g(x_{t-1}^r, u_t)\right]^T R^{-1}$$

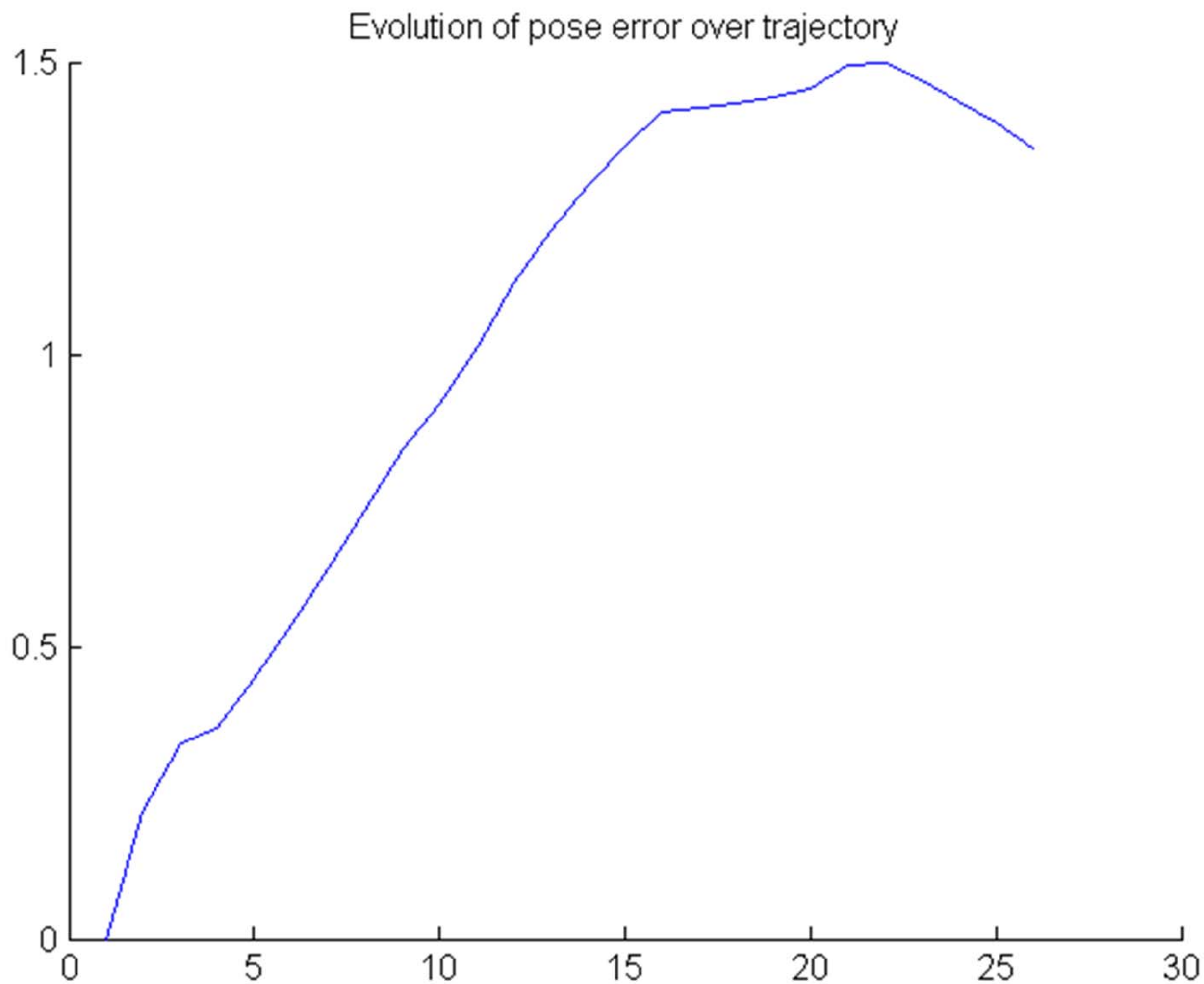
$$- \sum_i \left[y_t^i - h(x_t)\right]^T Q^{-1} \frac{\partial}{\partial x_t^r}\left[h(x_t)\right]$$

$$\frac{\partial J}{\partial x_t^m} = -\sum_i \left[y_t^i - h(x_t)\right]^T Q^{-1} \frac{\partial}{\partial x_t^m}\left[h(x_t)\right]$$

  - Initial Estimate of complete state (from odometry, other sensors)

$$\tilde{x}_{0:t}$$

GraphSLAM Preliminary Results



GraphSLAM

- GraphSLAM by Thrun and Montemerlo [2006]
  - Many interesting customizations to make optimization tractable
    - Linearization of models to form locally quadratic problem
    - Factorization of map into robot poses to reduce graph size
    - Scan points used as features with correspondence updated inside optimization
  - Full details in Chap 11 of Probabilistic Robotics
  - Summarized in extra slides at the end of this presentation
    - Warning: slightly different notation used

Outline

- The GraphSLAM algorithm
  - Derivation of feature based optimization problem
  - Derivation of scan based optimization problem
  - Discussion of solution methods
  - Implementation and Results

GraphSLAM with Scan Registration

- GraphSLAM – Scan Registration
  - No map elements are included in the state vector.
  - Instead, all scans are converted into relative pose measurements through registration

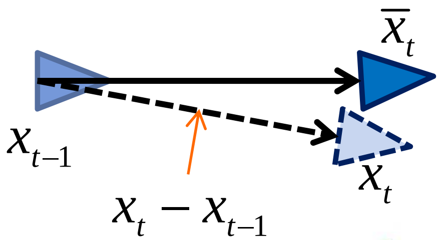
$$x_t^r = \begin{pmatrix} X_t \\ Y_t \\ Z_t \\ \phi_t \\ \theta_t \\ \psi_t \end{pmatrix}, \quad x_{0:t}^r = \begin{pmatrix} x_0^r \\ x_1^r \\ \vdots \\ \vdots \\ x_t^r \end{pmatrix}$$

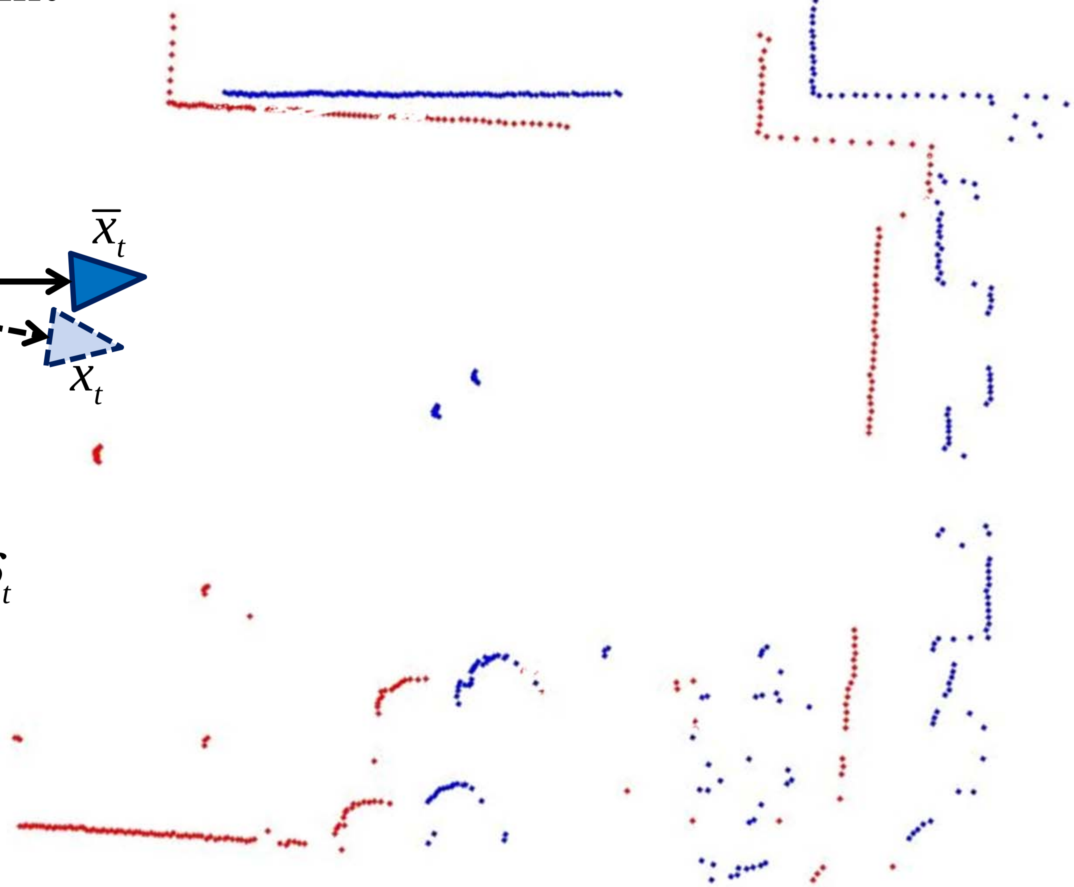
Robot State at time t

Full state

GraphSLAM with Scan Registration

- True, odometry motion and resulting scans, can choose to include motion model constraint

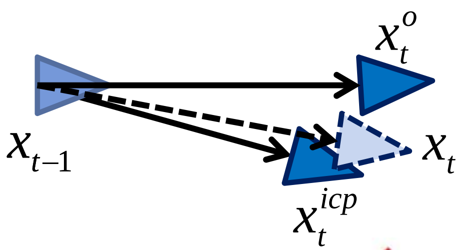


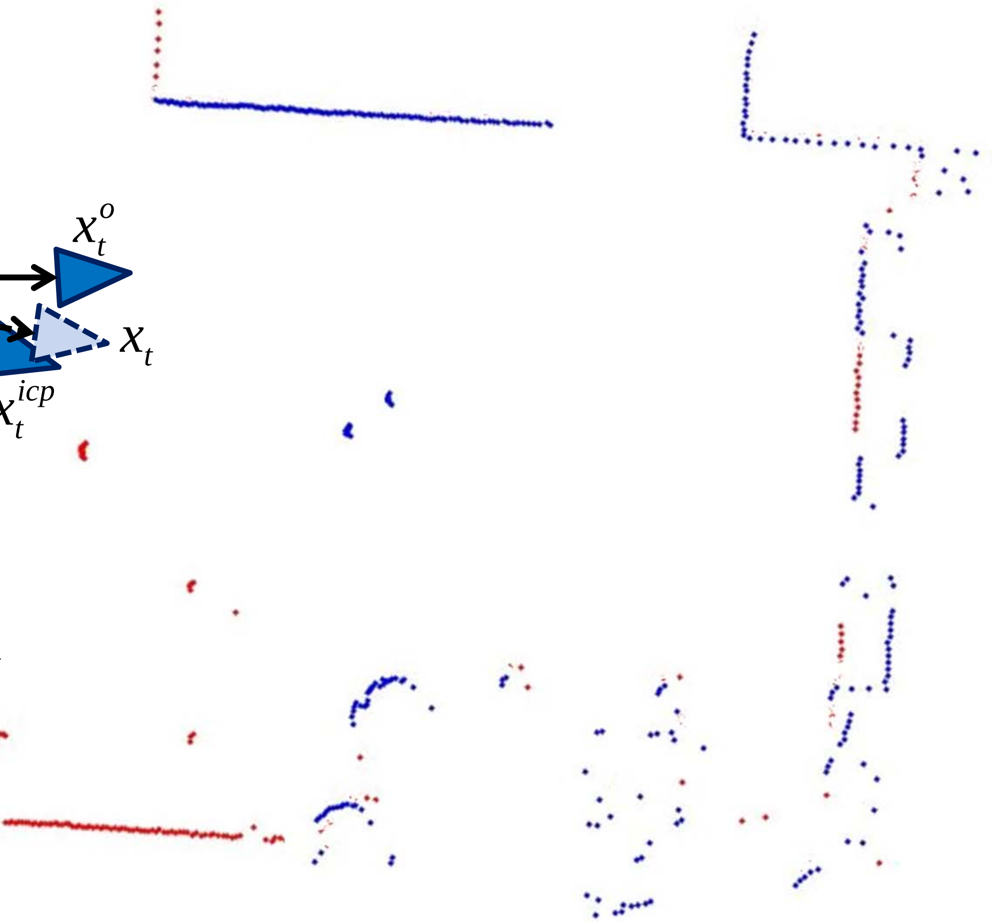


$$x_t = g(x_{t-1}, u_t) + \delta_t$$
$$\delta_t \sim N(0, R_t)$$

GraphSLAM with Scan Registration

- ICP scan match is a measurement between current and previous pose





$$y_t = h(x_t^r, x_{t-1}^r) + \delta_t$$

$$\delta_t \sim N(0, Q_t)$$

GRAPHSLAM WITH SCAN REGISTRATION

- Available information
  - Inputs and Motion model

$$u_{0:t}, \quad x_t^r = g(x_{t-1}^r, u_t) + \delta_t$$

  - Measurements and measurement model

$$y_{1:t}, \qquad y_t = h(x_t^r, x_{t-1}^r) + \varepsilon_t = x_t^r - R_t^* x_{t-1}^r - t_t^*$$

    - Where $y_t = 0$

  - The scan registration process, therefore, changes the measurement model into a motion model
    - Depends on the current and previous robot state only
    - Can choose to include regular motion model too, and will be weighted based on relative uncertainty

GraphSLAM Derivation (Lu/Milios)

- So the resulting negative log likelihood measurement constraint for each ICP match is

$$-\ln p(y_t \mid x_{t-1:t}) = const. + \left[y_t - h(x_{t-1:t}) \right]^T Q_t^{-1} \left[y_t - h(x_{t-1:t}) \right]$$

- In general, if loop closure is detected from scan *i* to scan *j*, we can add a measurement constraint between any two poses

$$-\ln p(y_{i,j} \mid x_{i,j}) = const. + \left[y_{i,j} - h(x_{i,j}) \right]^T Q_{i,j}^{-1} \left[y_{i,j} - h(x_{i,j}) \right]$$

- The full set of constraints collected are once again formed into a large optimization problem

GraphSLAM Derivation (Lu/Milios)

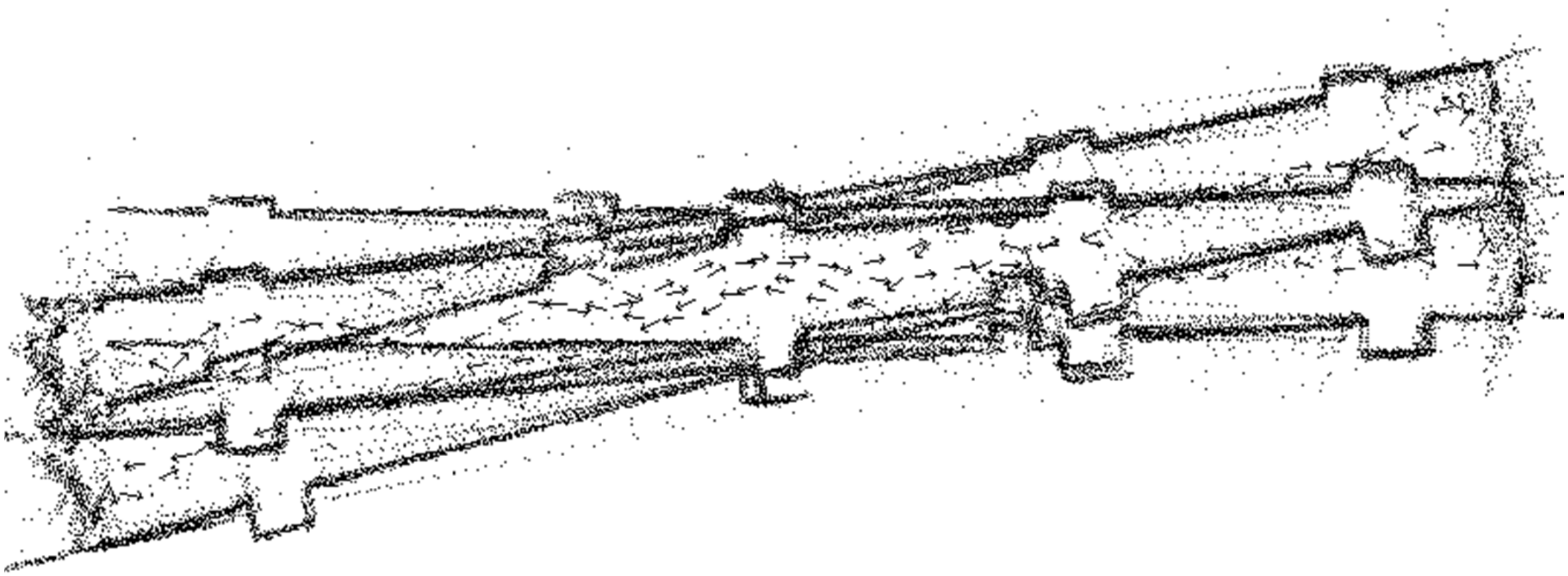
- Again, we take the negative log likelihood version of the cost function

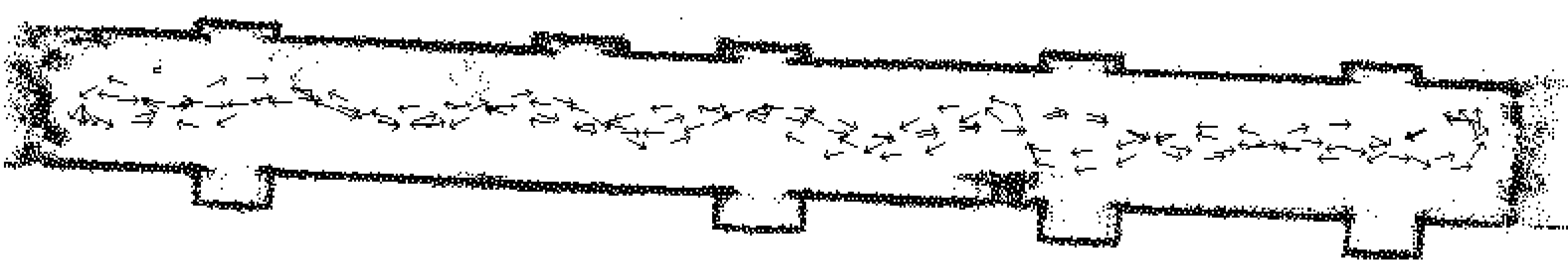
$$\min_{x_{0:t}} \ \mathrm{J} = \text{const.} \ + \sum_{i,j} \left[y_{i,j} - h(x_{i,j}) \right]^T Q_{i,j}^{-1} \left[y_{i,j} - h(x_{i,j}) \right]$$

- Then solve the quadratic program however we'd like
  - Pseudo-inverse
  - Gauss-Newton
  - Levenberg-Marquardt

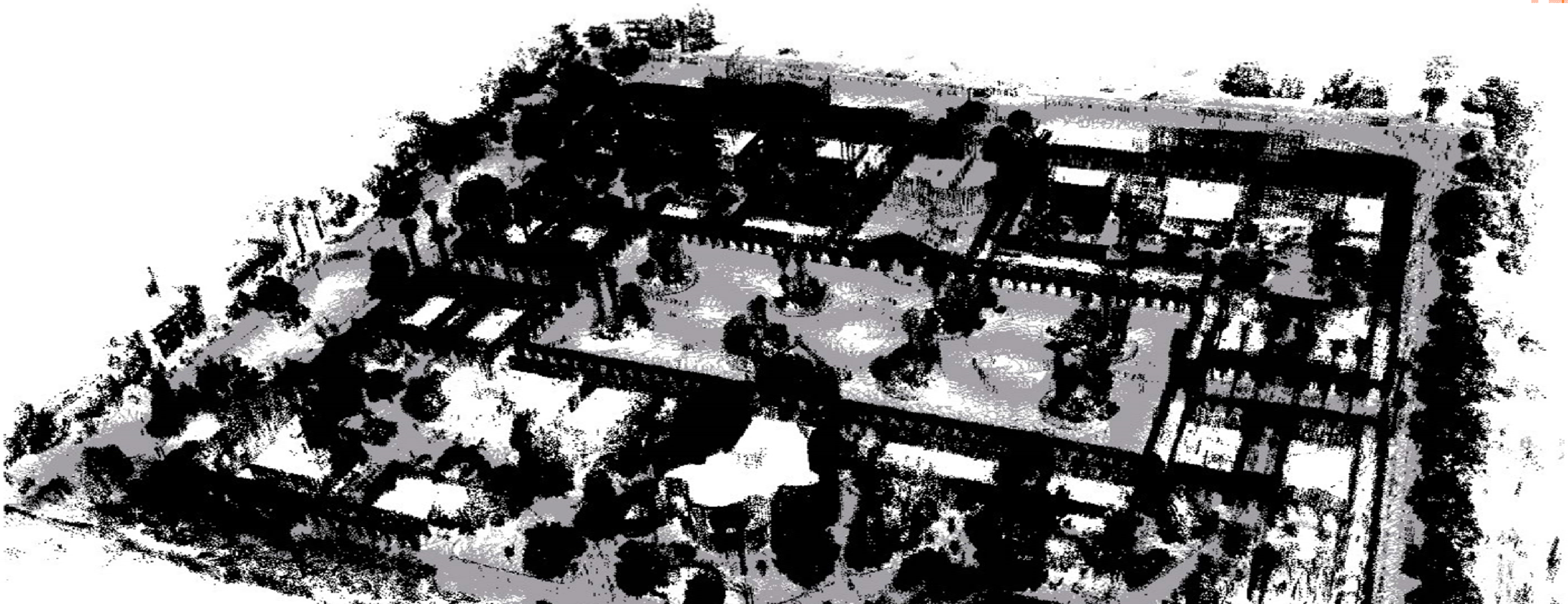
RESULTS FROM LU AND MILIOS

- Odometry only, and with GraphSLAM, using Sick Lidar [Lu, Milios at York in 1997].



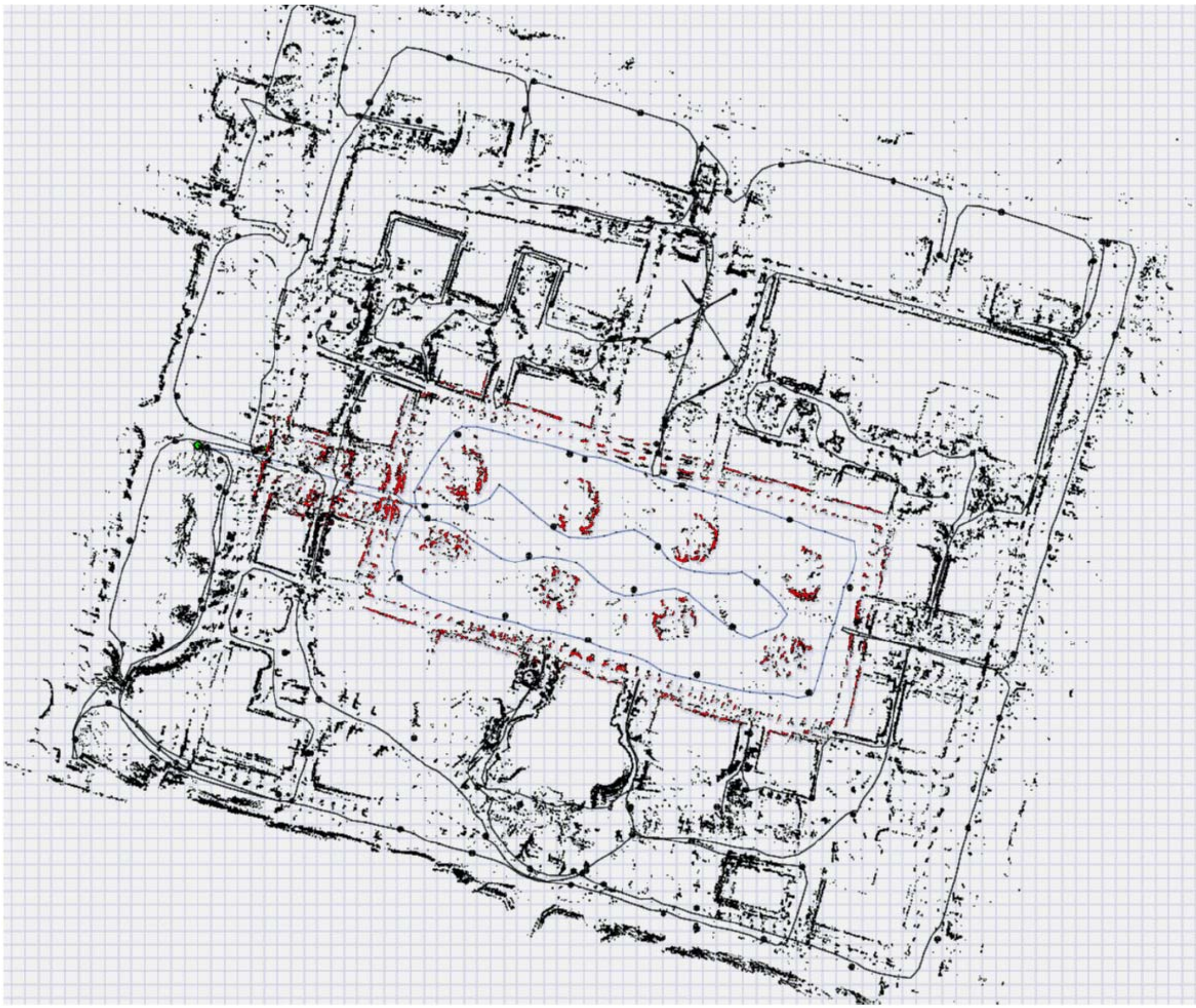


GraphSLAM Results



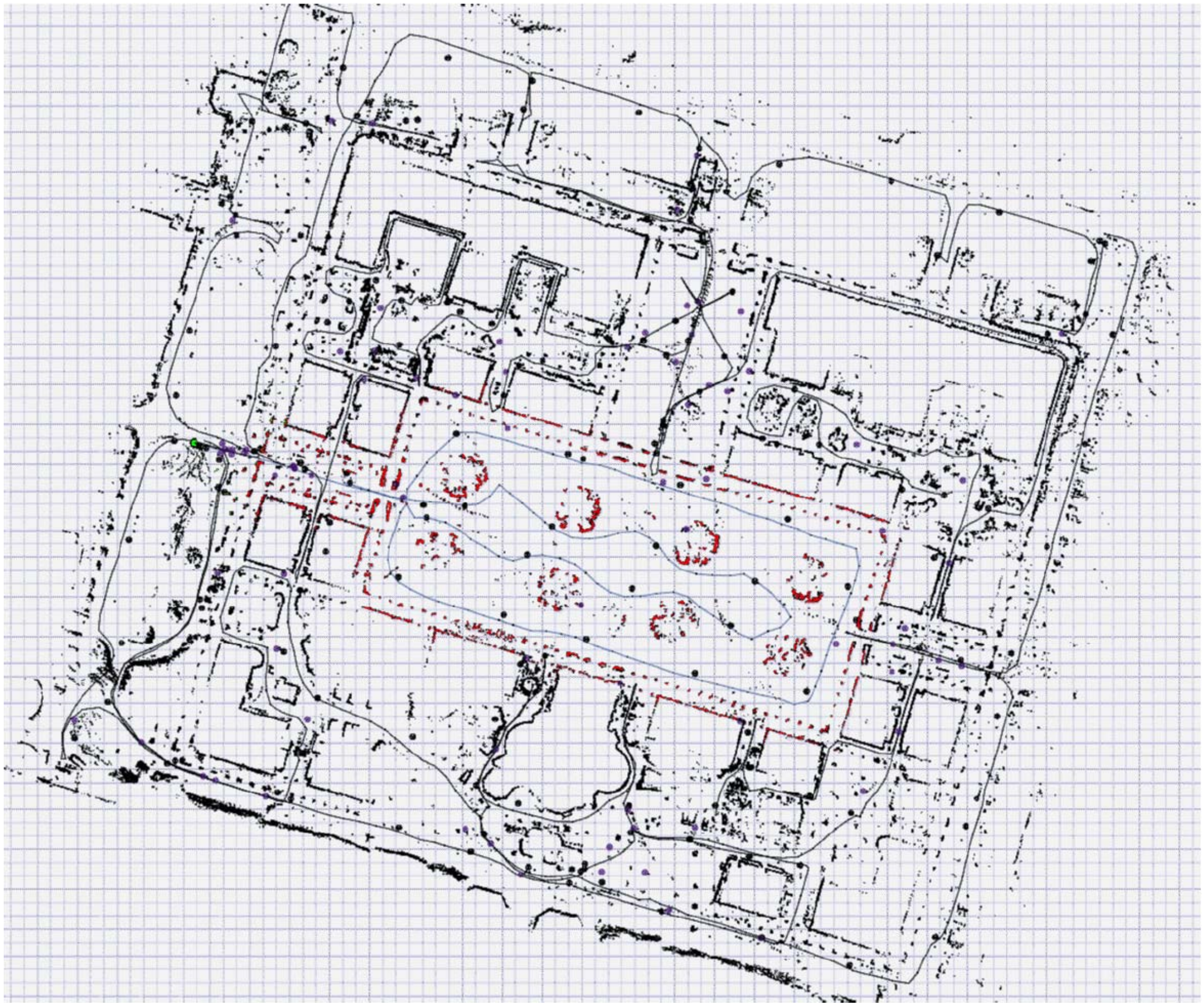
GRAPHSLAM RESULTS

- GPS/Odometry map of Stanford (600m x 600m)



GraphSLAM Results

- Corrected using GraphSLAM

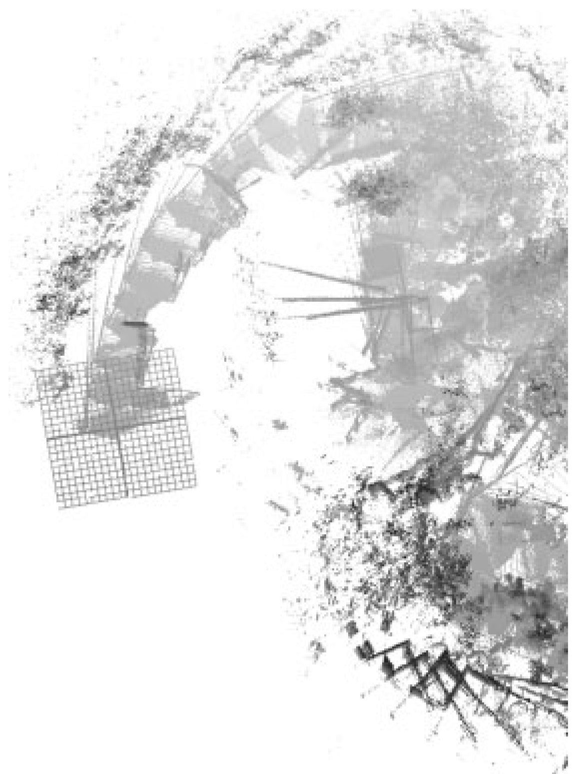


EXTENSIONS

- 3D GraphSLAM [Nuchter 2008]
  - Extension is actually taking the derivatives for linearization and moving it to 3D. Looked a lot at what the best way is for numerical stability of the solution when formulating the problem as a sequence of 3DOF inertial poses.
    - Euler angles
    - Quaternions
    - Helical motion
    - Rotation Linearization
    - ICP related improvements using KD-trees
    - Global Relaxation, a method for revisiting scan matching given the results of GraphSLAM

Results from Nuchter

- Large scale outdoor campus mapping



Odometry

Without loop closure

Results from Nuchter

With loop closure

With global relaxation

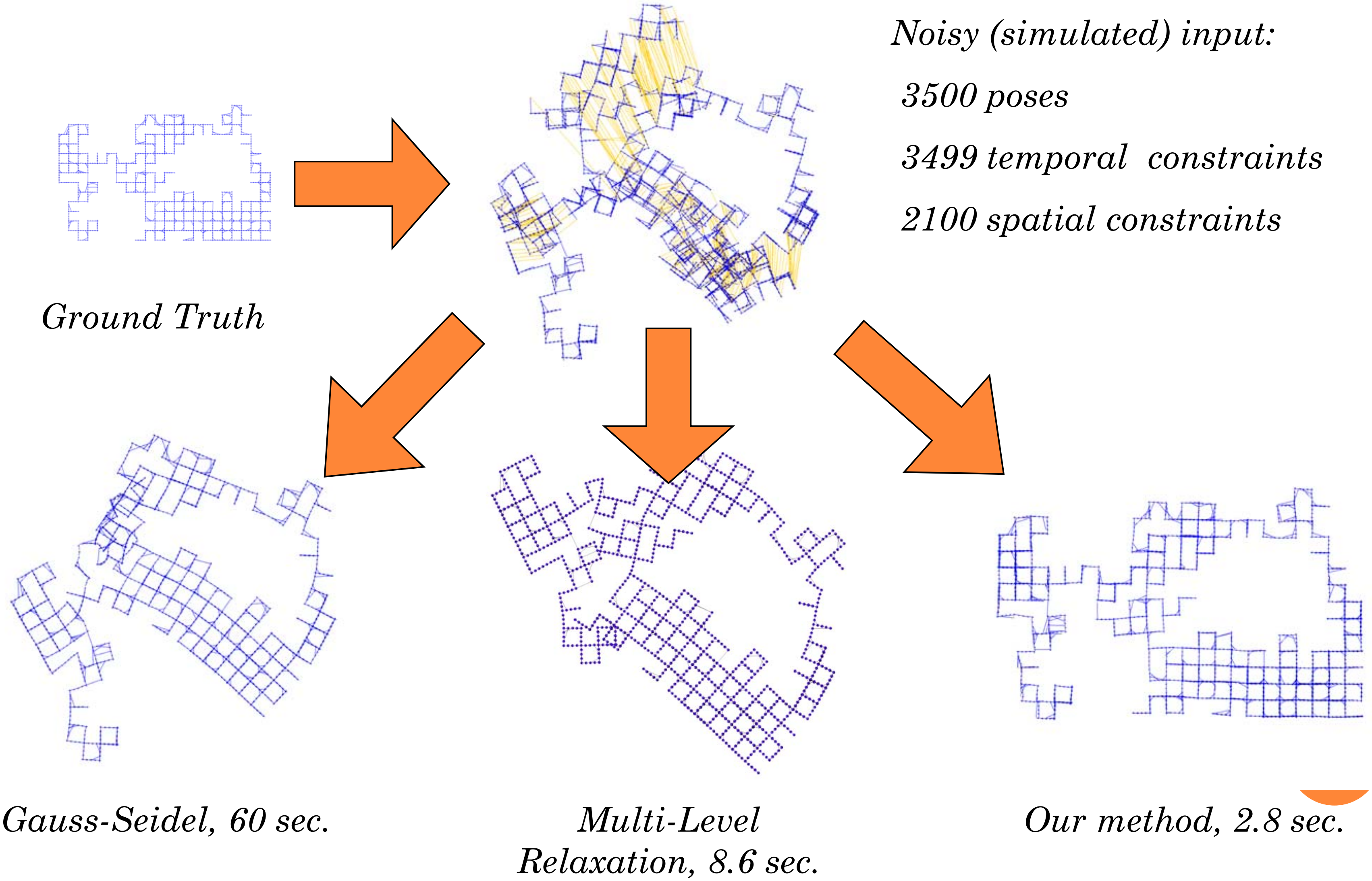
Extensions by Olson

- Re-parametrization [Olson 2009]
  - Most work uses a sequence of transformations between global poses to capture the motion of the robot
  - Olson uses an addition of differences in the pose parameters
  - This is inexact, but much faster

- Stochastic Gradient Descent
  - Do forever:
    - Pick a constraint
    - Descend in direction of constraint's gradient
    - Scale gradient magnitude by alpha/iteration
    - Clamp step size
  - iteration++
  - alpha/iteration→0 as t→∞
  - Robustness to local concavities
    - Hop around the state space, "stick" in the best one
  - Good solution very fast, "perfect" solution only as t→∞

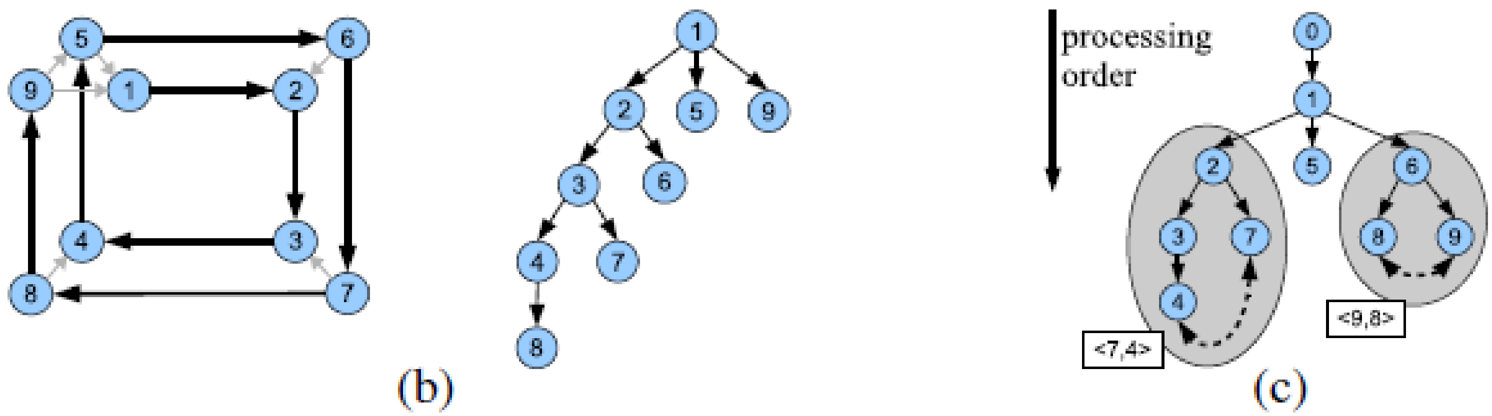
Results from Olson

- Olson



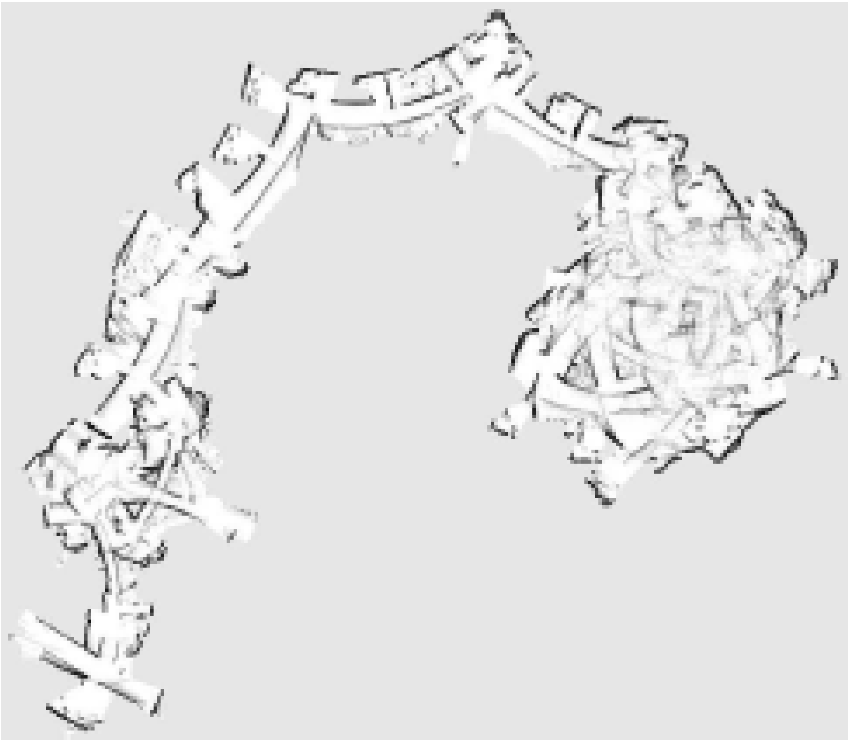
Extensions

- Grisetti further modified the structure of the optimization by reorganizing the nodes of the graph into a tree with extra loop closing links. [Grisetti 2010]
  - A direct extension of Olson's formulation, using the



RESULTS FROM TORO

- 1000 nodes, less than a second to compute.



Results from Toro

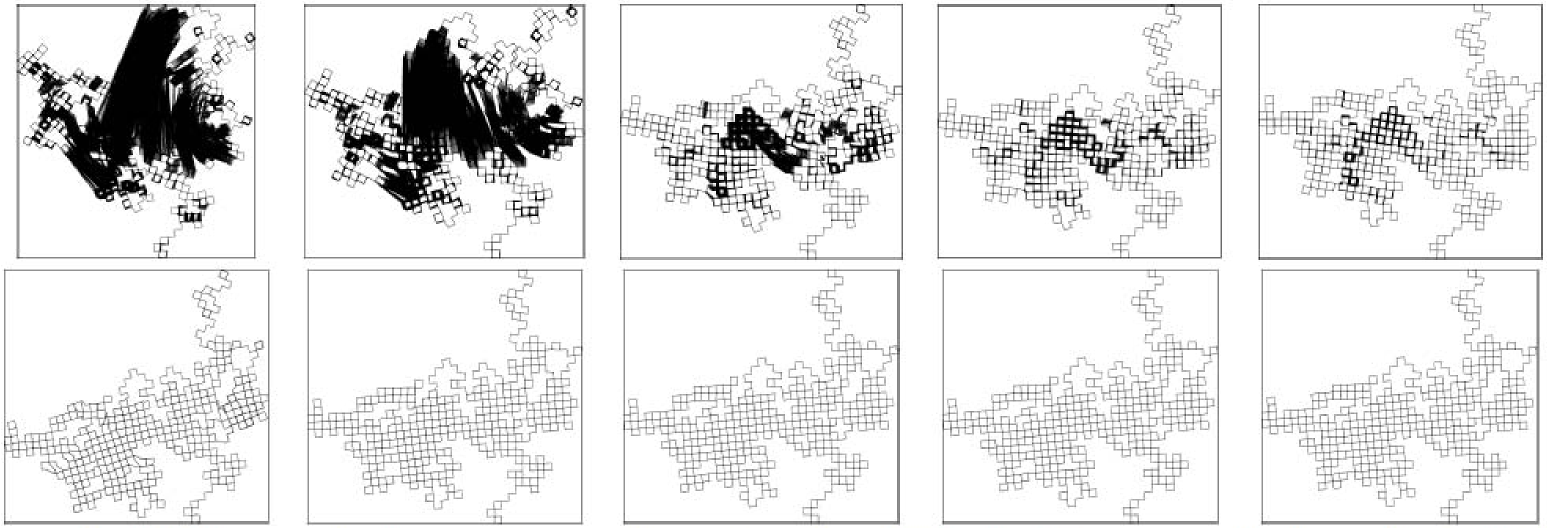
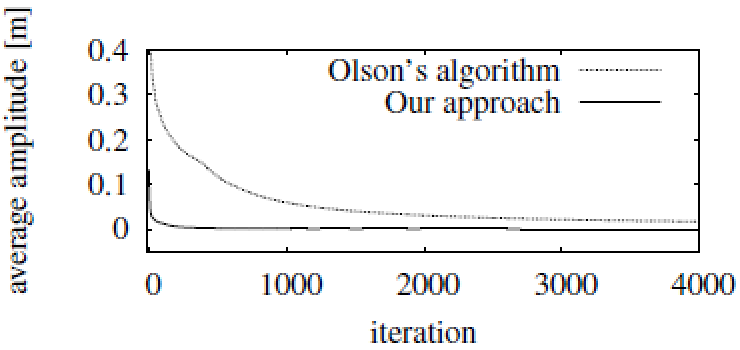
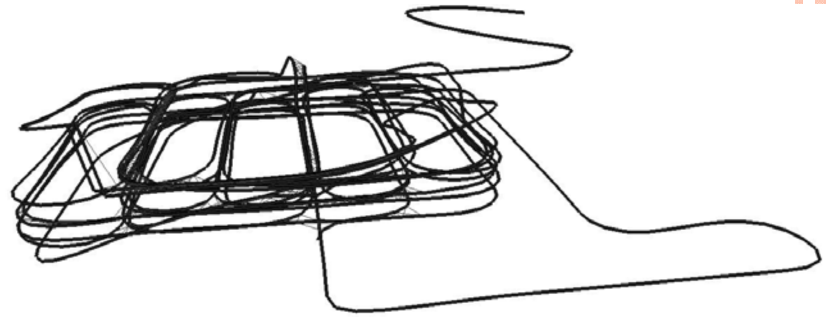


Fig. 4. Results of Olson's algorithm (first row) and our approach (second row) after 1, 10, 50, 100, 300 iterations for a network with 64k constraints. The black areas in the images result from constraints between nodes which are not perfectly corrected after the corresponding iteration (for timings see Figure 6).

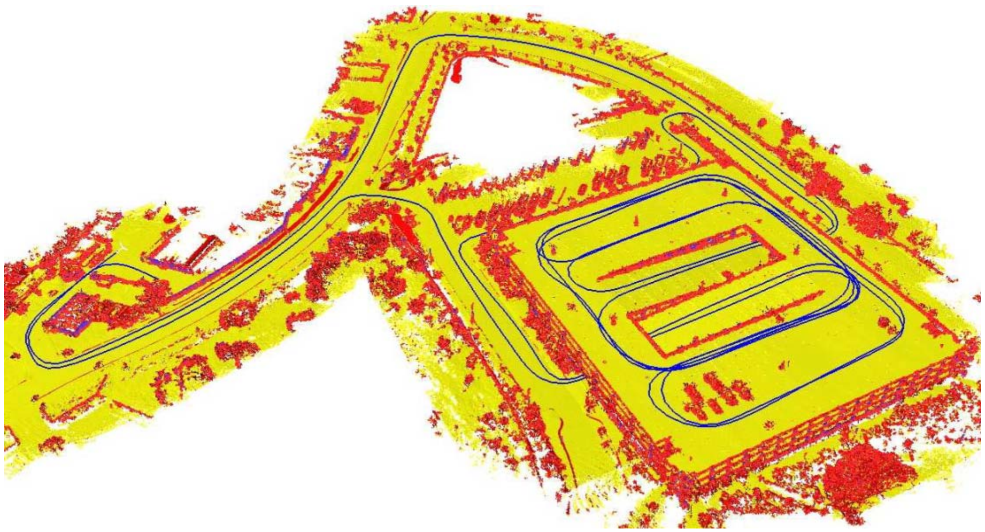


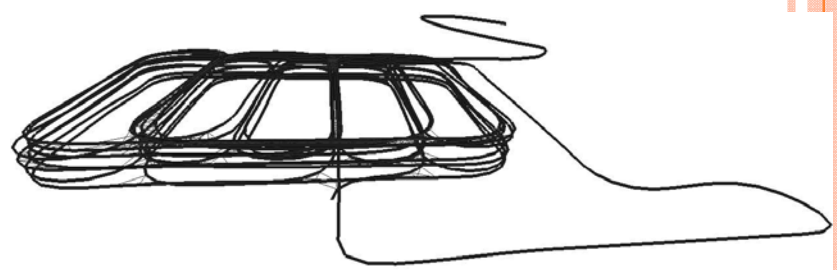
Results from Toro

Original



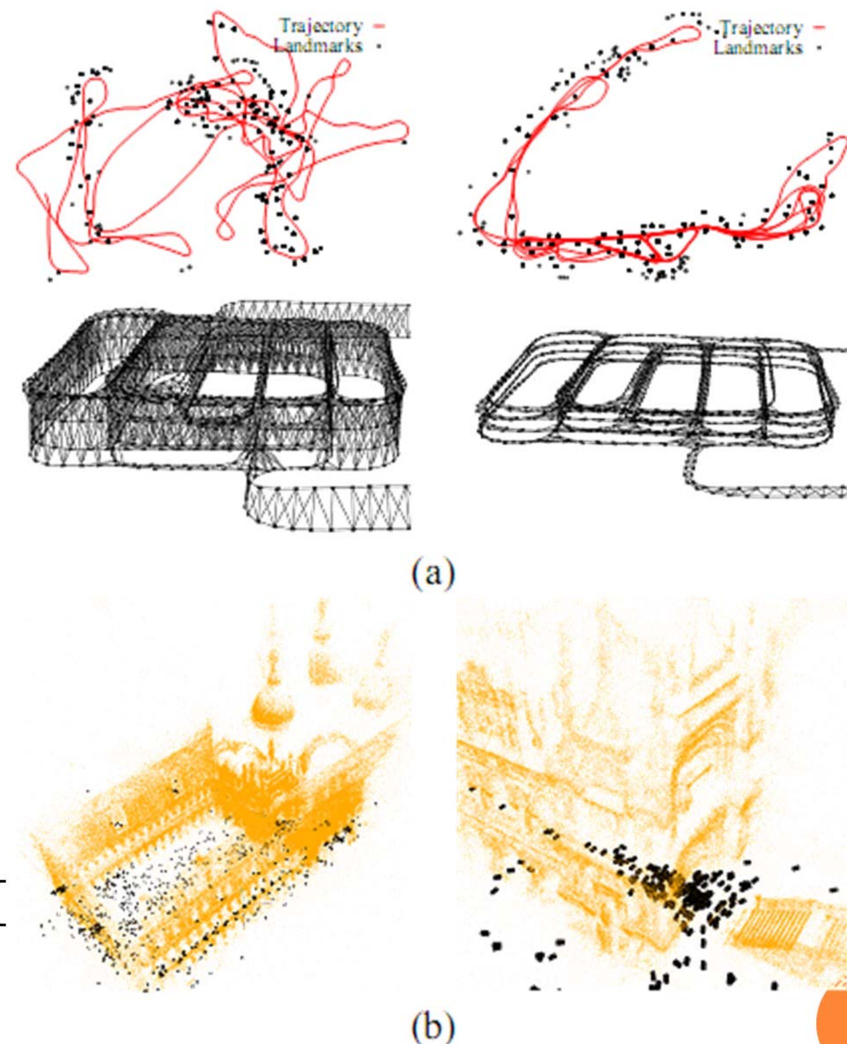
Optimized



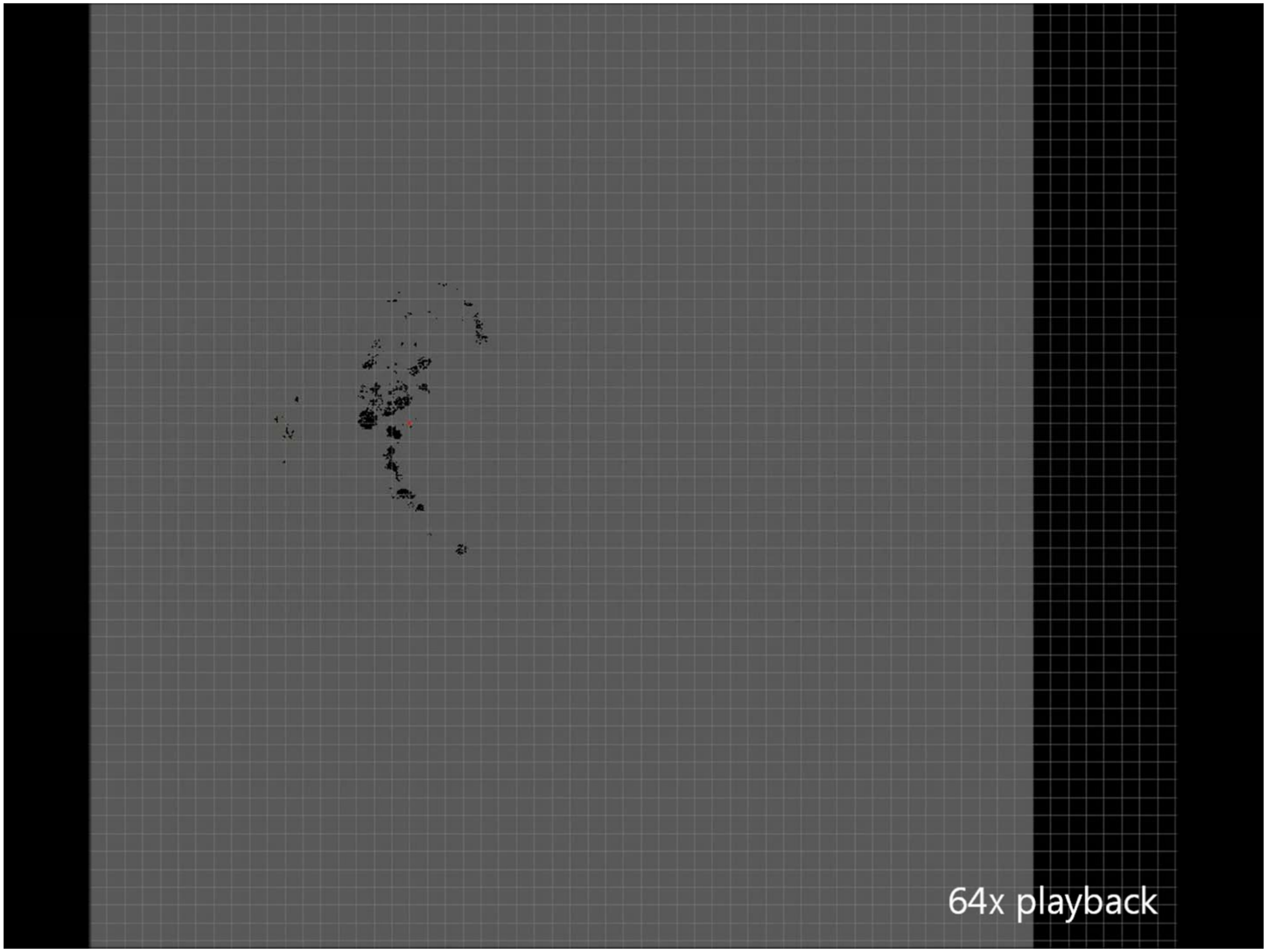


CURRENT STANDARD – G2O

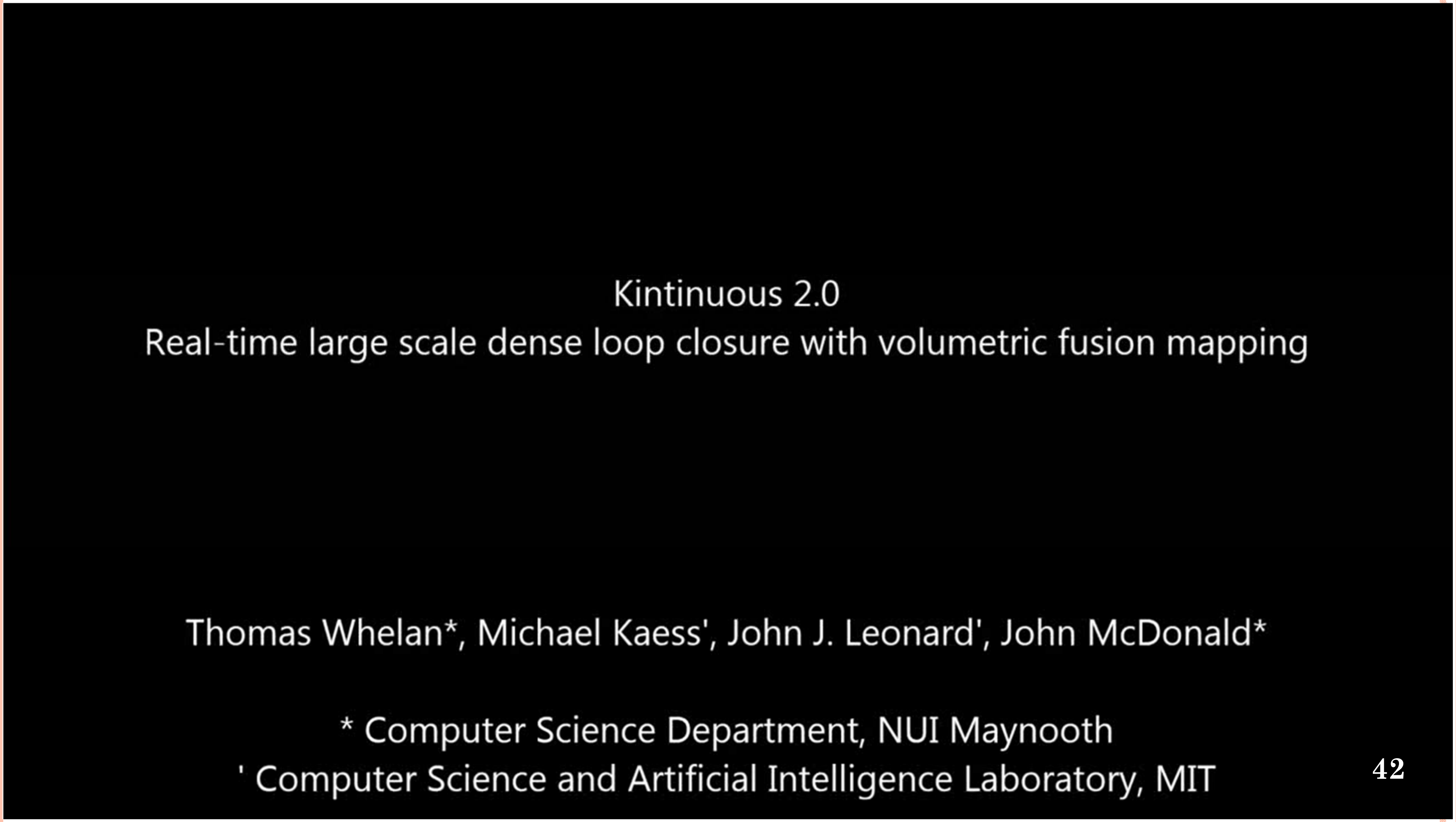
- Fast Backend Solver [Grisetti 2011]
- Takes the best of previous methods
- Works on wide range of problems
- Uses standard linear algebra packages
- Easily extensible, modifiable
- Available on OpenSLAM
- Integrated into ROS



Results from NASA Sample Return



STATE OF THE ART – [KINTINUOUS 2013]

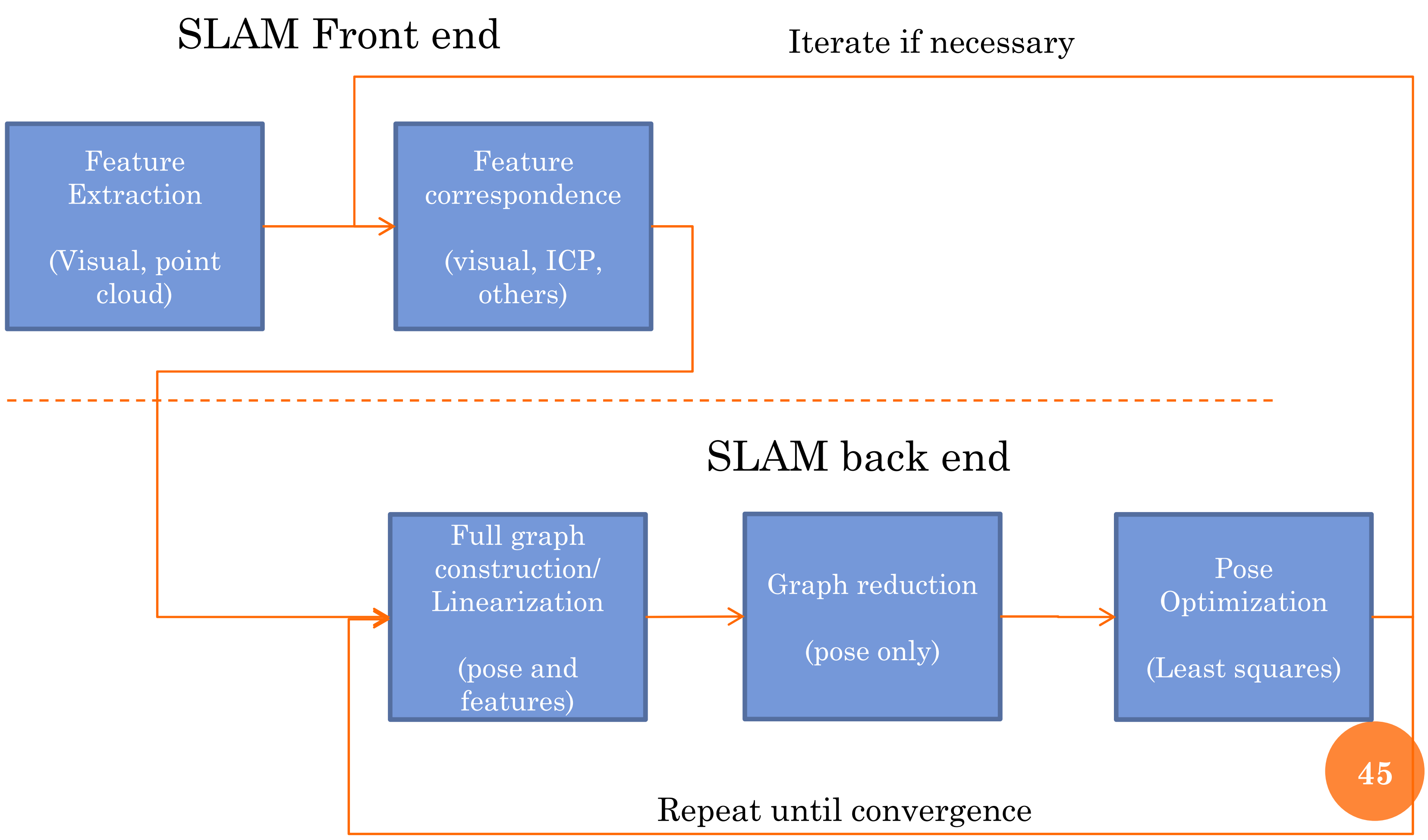


Extra Slides

GraphSLAM Solution Pipeline

- Feature extraction
  - Identify features in images (SIFT, SURF) or use laser scan points as features
- Feature correspondence
  - Standard visual techniques based on descriptors, proximity based such as ICP and many others
- Graph construction
  - Linearize measurement and motion information and populate a sparse matrix with constraint weightings based on covariance inverse (information matrix).
- Graph reduction
  - Reduce graph size by eliminating features, done by converting each feature measurement to an information gain on each pose from which it was measured
- Optimization
  - Any method you wish that can solve a sparse quadratic program (least squares, conjugate gradient, Levenberg-Marquardt)

GRAPHSLAM OFFLINE SOLUTION PIPELINE

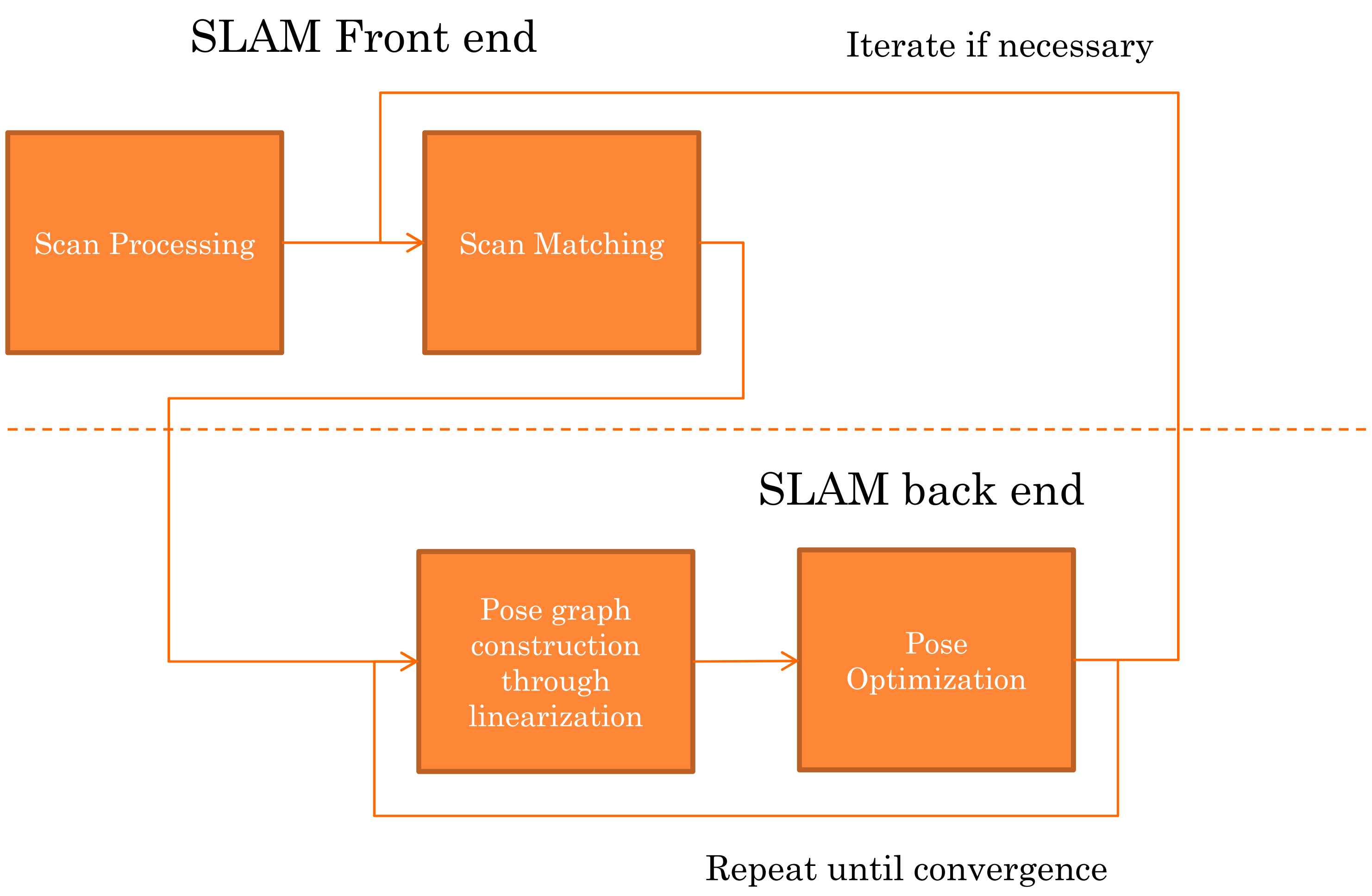


GraphSLAM Derivation (Thrun)

- There are four steps that result in Thrun's version of the GraphSLAM optimization

1. Initialize: find an initial estimate of the trajectory, through odometry, raw ICP, whatever.
2. Linearize: given the current estimate, find Jacobians of measurement and motion information, and construct the full graph.
3. Reduce: eliminate the features from the graph through an explicit step, reducing graph size
4. Solve: solve the quadratically constrained optimization to maximize probability of pose estimate, given measurements, inputs, correspondences.

GraphSLAM Offline Solution Pipeline



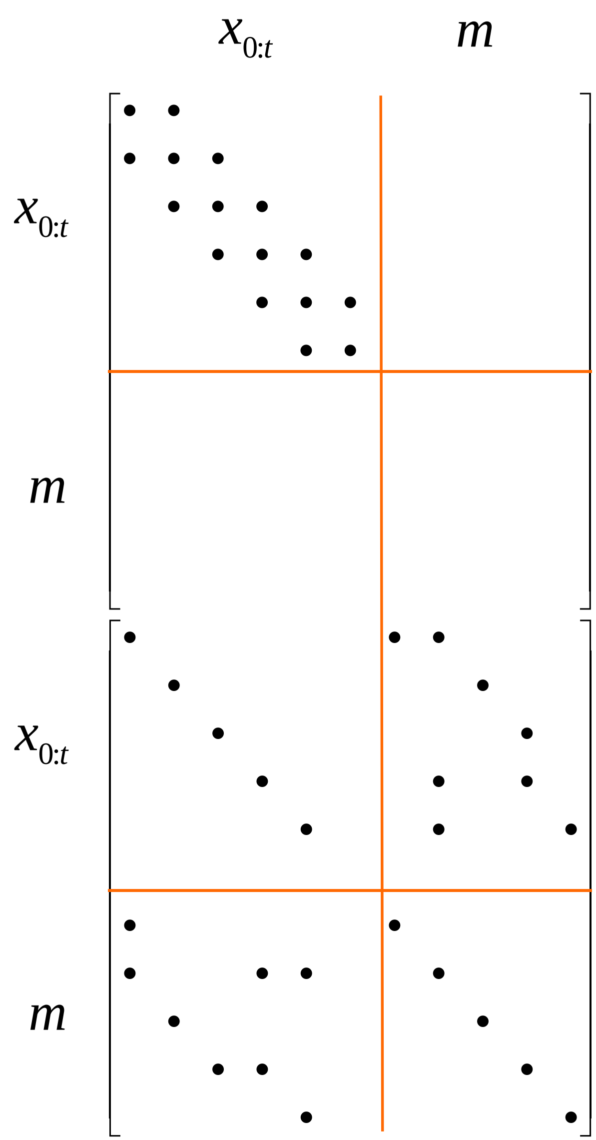
GraphSLAM Derivation (Thrun)

- Derivation proceeds as follows:
  - Derivation of proposed optimization method
    - Define initial solution
    - Linearize measurement and motion models about current estimate of solution
    - Restate quadratic cost that results
    - Figure out how to reduce the graph to eliminate features, using marginals trick (Schur complement)
    - Present simple method for solving for the path and map
    - Repeat as necessary, updating initial solution each time through
    - Can add an outer loop that re-evaluates correspondence too

Population of Sparse Graph Matrix

- Motion constraint information

- Measurement constraint information



LINEARIZATION FOR COST DEFINITION

- A simple first order Taylor series expansion of $g(x,u)$ and $h(x,c)$ will result in a quadratic cost function
  - We can therefore proceed with sequential quadratic programming
- Similar to the EKF, the linearization is as follows

$$g(x_{t-1},u_t) \approx g(\mu_{t-1},u_t) + \frac{\partial}{\partial x_{t-1}} g(x_{t-1},u_t)\Big|_{x_{t-1}=\mu_{t-1}} (x_{t-1}-\mu_{t-1})$$

$$= g(\mu_{t-1},u_t) + G_t \cdot (x_{t-1}-\mu_{t-1})$$

$$h(z_t,c_t^i) \approx h(\mu_t,c_t^i) + \frac{\partial}{\partial z_t} h(z_t,c_t^i)\Big|_{z_t=\mu_t} (z_t-\mu_t)$$

$$= h(\mu_t,c_t^i) + H_t \cdot (z_t-\mu_t)$$

LINEARIZATION FOR MATRIX DEFINITION

- Substituting into the cost function gives

$$J = \text{const.} + \left[x_0 - \mu_0\right]^T \Sigma_0^{-1}\left[x_0 - \mu_0\right]$$
$$+\sum_t [x_t - g(\mu_{t-1}, u_t) - G_t \cdot (x_{t-1} - \mu_{t-1})]^T R^{-1} [x_t - g(\mu_{t-1}, u_t) - G_t \cdot (x_{t-1} - \mu_{t-1})]$$
$$+\sum_t \sum_i \left[y_t^i - h(\mu_t, c_t^i) - H_t \cdot (z_t - \mu_t)\right]^T Q^{-1} \left[y_t^i - h(\mu_t, c_t^i) - H_t \cdot (z_t - \mu_t)\right]$$

  - Which has many constant terms (mean is known) that can be combined into one.

LINEARIZED GRAPHSLAM OPTIMIZATION

- Rearranging, we can write the cost as

$$J = \text{const.} + x_0{}^T \Sigma_0^{-1} x_0 + \sum_t x_{t-1:t}{}^T \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R^{-1} \begin{pmatrix} -G_t^T & 1 \end{pmatrix} x_{t-1:t}$$

$$+ x_{t-1:t}{}^T \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R^{-1} \left[g(\mu_{t-1}, u_t) - G_t \mu_{t-1} \right]$$

$$+ \sum_t \sum_i z_t{}^T H_t^{iT} Q^{-1} H_t^i z_t + z_t{}^T H_t^{iT} Q^{-1} \left[y_t^i - h(\mu_t, c_t^i) + H_t^i \mu_t \right]$$

- Which is of the form, a simple least squares problem.

$$J = const - z_{0:t}^T \Omega z_{0:t} + z_{0:t}^T \xi$$

  - Where the full information matrix and vector are the two coefficients in the cost function

Constructing the Cost Function

- We need to construct both the information vector and matrix
  - All constraints can be added independently in negative log likelihood form, taking care to place the additions in the correct rows and column (per initial diagram)

  - Prior $\Omega = \Omega_0$

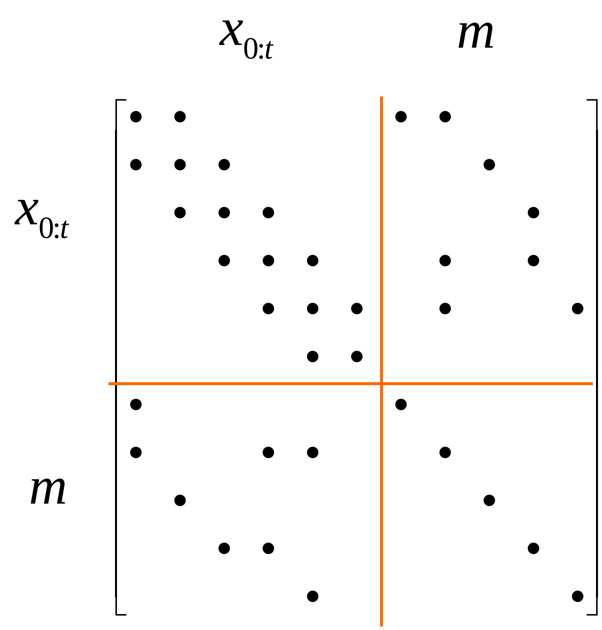
  - Each motion step
$$\Omega = \Omega + \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R^{-1} \begin{pmatrix} -G_t^T & 1 \end{pmatrix}$$
$$\xi = \xi + \begin{pmatrix} -G_t^T \\ 1 \end{pmatrix} R^{-1} \left[g(\mu_{t-1}, u_t) - G_t \mu_{t-1} \right]$$

  - Each measurement
$$\Omega = \Omega + H_t^{iT} Q^{-1} H_t^i$$
$$\xi = \xi + H_t^{iT} Q^{-1} \left[y_t^i - h(\mu_t, c_t^i) + H_t^i \mu_t \right]$$

Population of Sparse Graph Matrix

- So, after linearization, we can form a quadratic cost matrix in all of the decision variables, which looks like

$\Omega =$



Graph Reduction

- The next big step is to eliminate all the features from the cost formulation, to reduce the size of the optimization problem

- The full SLAM posterior can be factored

$$p(z_{0:t} \mid y_{1:t}, u_{1:t}, c_{1:t}) = p(x_{0:t} \mid y_{1:t}, u_{1:t}, c_{1:t})\, p(m \mid x_{0:t}, y_{1:t}, u_{1:t}, c_{1:t})$$

- Where we can find the marginal pose distribution by integrating over map variables

$$p(x_{0:t} \mid y_{1:t}, u_{1:t}, c_{1:t}) = \int p(z_{0:t} \mid y_{1:t}, u_{1:t}, c_{1:t})\, dm$$

Graph Reduction

- To find this marginal probability, we need a famous lemma:
  - Marginals of a multivariate distribution (Schur complement, inversion lemma):

Let the probability distribution $p(x, y)$ over the random variables x, y be a Gaussian represented in the information form:

$$\Omega = \begin{bmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{bmatrix} \text{ and } \xi == \begin{bmatrix} \xi_x \\ \xi_y \end{bmatrix}.$$

If Ω_{yy} is invertible, the marginal $p(x)$ is a Gaussian whose information form is

$$\bar{\Omega}_{xx} = \Omega_{xx} - \Omega_{xy}\Omega_{yy}^{-1}\Omega_{yx} \text{ and } \bar{\xi}_x = \xi_x - \Omega_{xy}\Omega_{yy}^{-1}\xi_y$$

Graph Reduction

- The elimination of features proceeds by using this lemma
  - Applying the marginalization lemma

$$\Omega = \begin{bmatrix} \Omega_{x_{0:t}x_{0:t}} & \Omega_{x_{0:t}m} \\ \Omega_{mx_{0:t}} & \Omega_{mm} \end{bmatrix} \text{ and } \xi == \begin{bmatrix} \xi_{x_{0:t}} \\ \xi m \end{bmatrix}$$

$$\bar{\Omega}_{x_{0:t}x_{0:t}} = \Omega_{x_{0:t}x_{0:t}} - \Omega_{x_{0:t}m}\Omega_{mm}^{-1}\Omega_{mx_{0:t}}$$

$$\bar{\xi}_{x_{0:t}} = \xi_{x_{0:t}} - \Omega_{x_{0:t}m}\Omega_{mm}^{-1}\xi_m$$

$$\bar{\Omega}_{x_{0:t}x_{0:t}} = \Omega_{x_{0:t}x_{0:t}} - \sum_i \Omega_{x_{0:t}m_i}\Omega_{m_i m_i}^{-1}\Omega_{m_i x_{0:t}}$$

$$\bar{\xi}_{x_{0:t}} = \xi_{x_{0:t}} - \sum_i \Omega_{x_{0:t}m_i}\Omega_{m_i m_i}^{-1}\xi_{m_i}$$

The matrix $\Omega_{x_{0:t}m_i}$ is nonzero only for poses in which the feature was measured.

  - But this seems to require an inversion the size of the map features
  - Luckily, each feature is independent of all other features, so it is a block diagonal inversion, that can be done one feature at a time

Solving the reduced optimization

- Finally, we solve the reduced quadratic program by simply inverting the information matrix, and recover the robot poses

$$\bar{\Sigma}_{x_{0:t}x_{0:t}} = \bar{\Omega}^{-1}_{x_{0:t}x_{0:t}}$$
$$\bar{\mu}_{x_{0:t}} = \bar{\Sigma}_{x_{0:t}x_{0:t}} \bar{\xi}_{x_{0:t}}$$

- This inversion would be very fast if every feature was observed at only one time, but is actually a little dense due to loop closures. We have choices, but all must do some serious work to get a solution.
  - Pseudo-inverse
  - Gauss-Newton
  - Levenberg-Marquardt

SOLVING THE REDUCED OPTIMIZATION

- To recover the map (if needed), we want to solve for the conditional map probability

$$p(m \mid x_{0:t}, y_{1:t}, u_{1:t}, c_{1:t})$$

- To find this conditional probability, we need the conditioning lemma

Let the probability distribution $p(x, y)$ over the random variables x, y be a Gaussian represented in the information form:

$$\Omega = \begin{bmatrix} \Omega_{xx} & \Omega_{xy} \\ \Omega_{yx} & \Omega_{yy} \end{bmatrix} \text{ and } \xi == \begin{bmatrix} \xi_x \\ \xi_y \end{bmatrix}.$$

The conditional $p(x \mid y)$ is a Gaussian whose information matrix is Ω_{xx} and whose information vector is $\xi_x - \Omega_{xy} y$

Recovering the map

- Application of the lemma yields

$$\bar{\Sigma}_{mm} = \Omega_{mm}^{-1}$$

$$\bar{\mu}_m = \Sigma_{mm} \left(\xi_m - \Omega_{mx_{0:t}} \bar{\mu}_{x_{0:t}} \right)$$

- The feature locations are finally computed by again applying the fact that each feature is independent, so that we get a simple feature by feature reconstruction
  - Define the set of poses at which feature *i* was observed as $\tau(i)$

$$\Sigma_{m_i m_i} = \Omega_{m_i m_i}^{-1}$$

$$\mu_{m_i} = \Sigma_{m_i m_i} \left(\xi_{m_i} - \Omega_{m_i \tau(i)} \bar{\mu}_{\tau(i)} \right)$$

NUCHTER'S OPENSLAM RESULTS