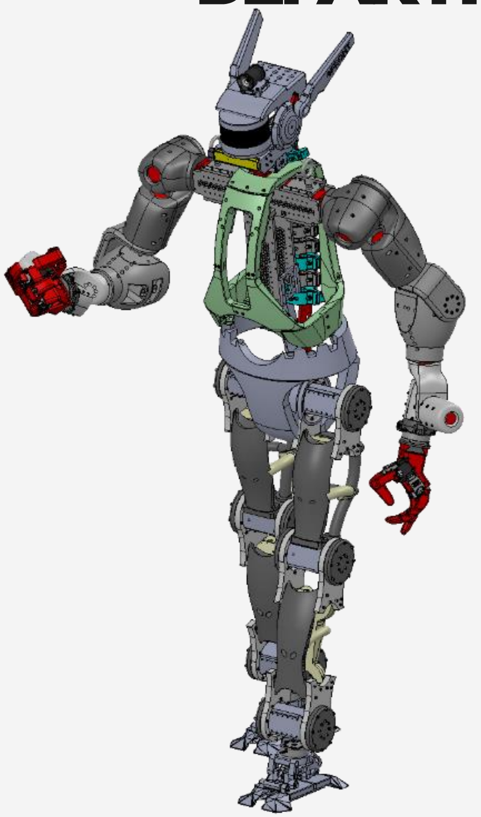
UNIVERSITY OF CALGARY

11: Mapping III

DEPARTMENT OF MECHANICAL ENGINEERING



Mobile Robotics

(ENME 650)

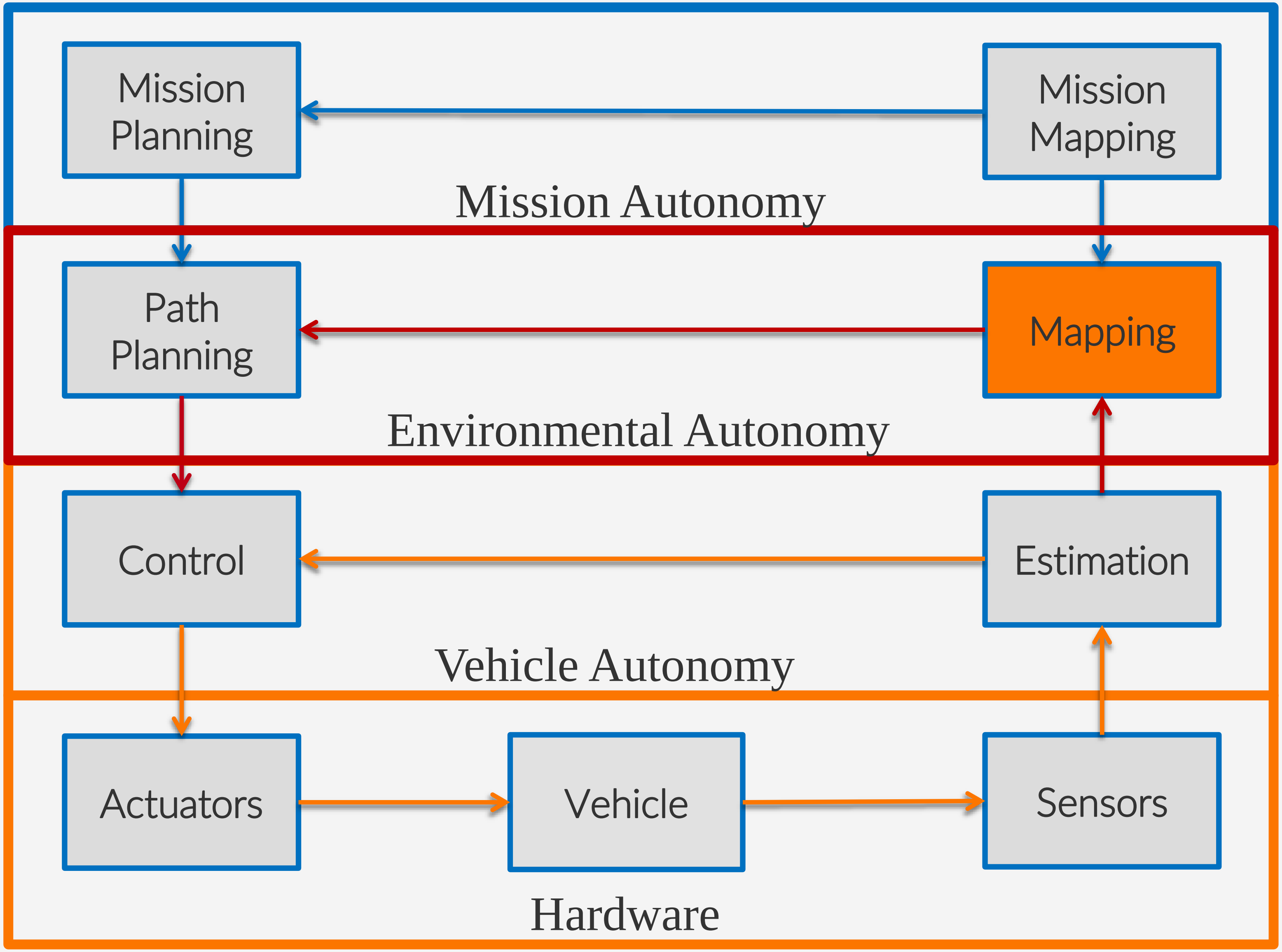
Instructor:

DR. ALEX RAMIREZ-SERRANO

Office: MEB 523
Phone: 220-3632
E-mail: aramirez@ucalgary.ca

CANADA

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

Aim to solve the entire SLAM problem over the entire history of the motion correcting for all past and present states as the robot moves and collects more information!

TWO MAIN SLAM APPROACHES

- Online SLAM (*have discussed this: use new information in estimate*)
  - Filter version of the SLAM problem, maximize

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

  - Process new information as it is received
  - Generate current best estimate, rely on Markov assumption and linearity to trust this is the best you can do, and use the solution in subsequent steps
  - EKF SLAM, FastSLAM, Occupancy Grid SLAM, etc.
- Full SLAM (*use the entire history of the information collected*)
  - Smoothing version of the SLAM problem, maximize

    $$p(x_{0:t} | y_{1:t}, u_{1:T})$$

  - 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

Recalling the SLAM types to know where Graph SLAM fits in.

FULL SLAM PROBLEM

Full SLAM - Features

- Simulation determine the robot pose history and static feature location in the environment.

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

$\uparrow$ *Robot State at time "t"*

$\uparrow$ *i^{th} feature*

$\uparrow$ *Feature map*

$\uparrow$ *Full robot state*

$\uparrow$ *Full state (robot & terrain) at time "t"*

FULL SLAM PROBLEM

- Available information
  - Inputs & Motion model

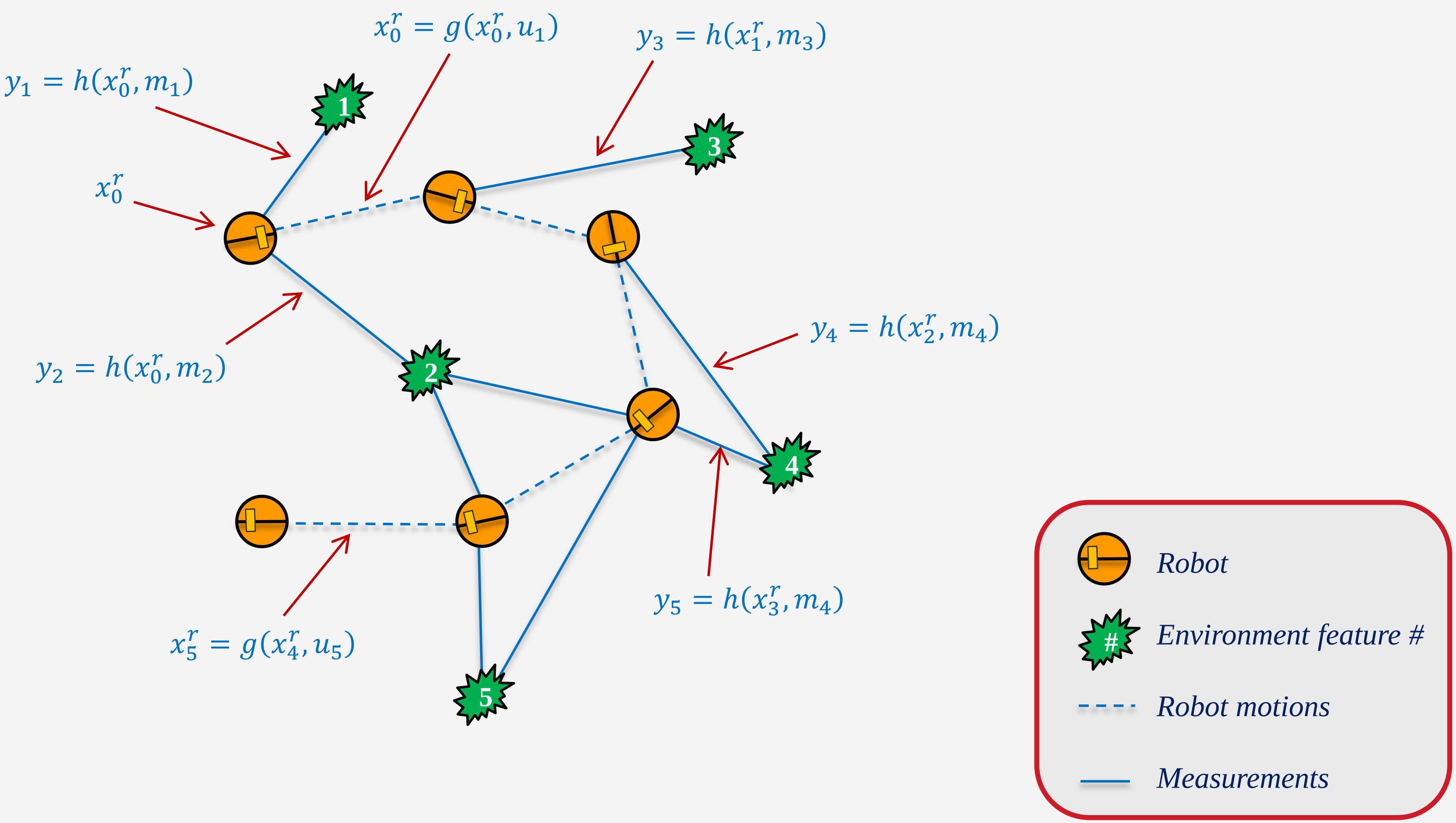
$$u_{0:t}, \qquad x_t^r = g(x_{t-1}^r, u_t) + \delta_t$$

  - Measurements & Measurement model

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

  - 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 estimation

Includes robot & map states, $x_{0:t}$

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

- Apply Bayes rule to separate out the current measurements

$$p(x_{0:t} | y_{1:t}, u_{1:t}) = \eta \, p(y_t | x_{0:t}, u_{1:t}) \, p(x_{0:t} | y_{1:t-1}, u_{1:t})$$

$$= \eta \, p(y_t | x_t) \, p(x_{0:t} | y_{1:t-1}, u_{1:t})$$

GRAPHSLAM OPTIMIZATION

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

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

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

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

  - If there is no prior information about the map, use $p(x_0^r)$

GRAPHSLAM OPTIMIZATION

- We can redefine our optimization problem as:

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

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

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

GRAPHSLAM OPTIMIZATION

- We can redefine our optimization problem as

$$\min -\ln\left(\eta\, p(x_0) \prod_{\tau=1} \left(p(x_\tau^r | x_{\tau-1}^r, u_\tau) \prod_i p(y_\tau^i | x_\tau) \right)\right)$$

$$\Downarrow$$

$$\min_{\mathrm{x}_{0:t}} \mathrm{J} = \mathrm{const.} - \ln\big(p(x_0)\big)$$

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

GRAPHSLAM OPTIMIZATION

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

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

  - Measurement:

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

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

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

GRAPHSLAM OPTIMIZATION

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

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

  - Measurement:

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

  - Prior:

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

GRAPHSLAM OPTIMIZATION

- The final form of optimization is now

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

- 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

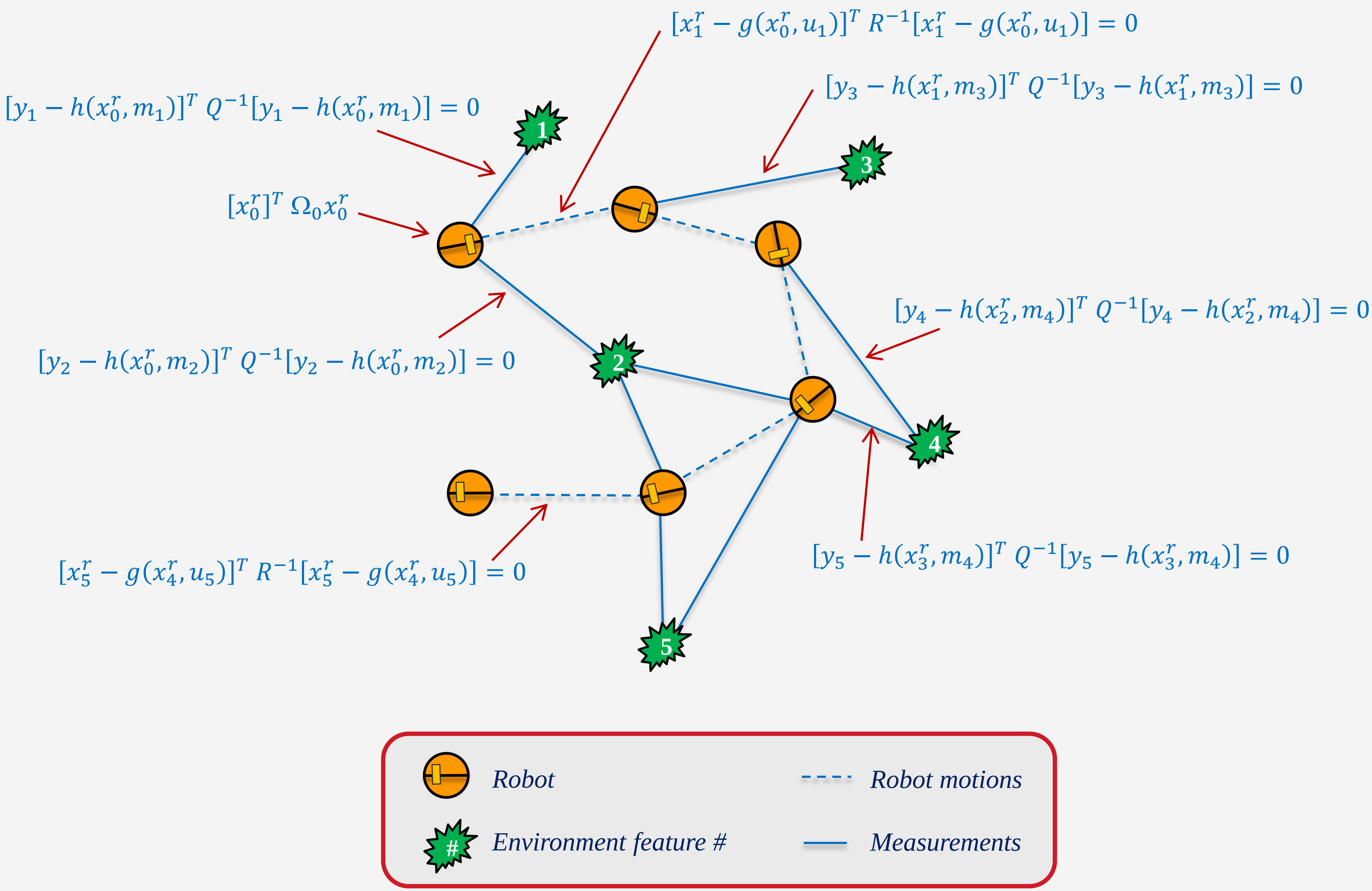
  - Can be considered a weighted distance minimization
    - Errors minimized together based on inverse of covariance weighting (*information matrix*)
    - Motion:

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

    - Measurement:

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

ILLUSTRATION OF THE CONSTRAINT GRAPH



GRAPHSLAM OPTIMIZATION

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

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

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

  - Gradient function

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

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

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

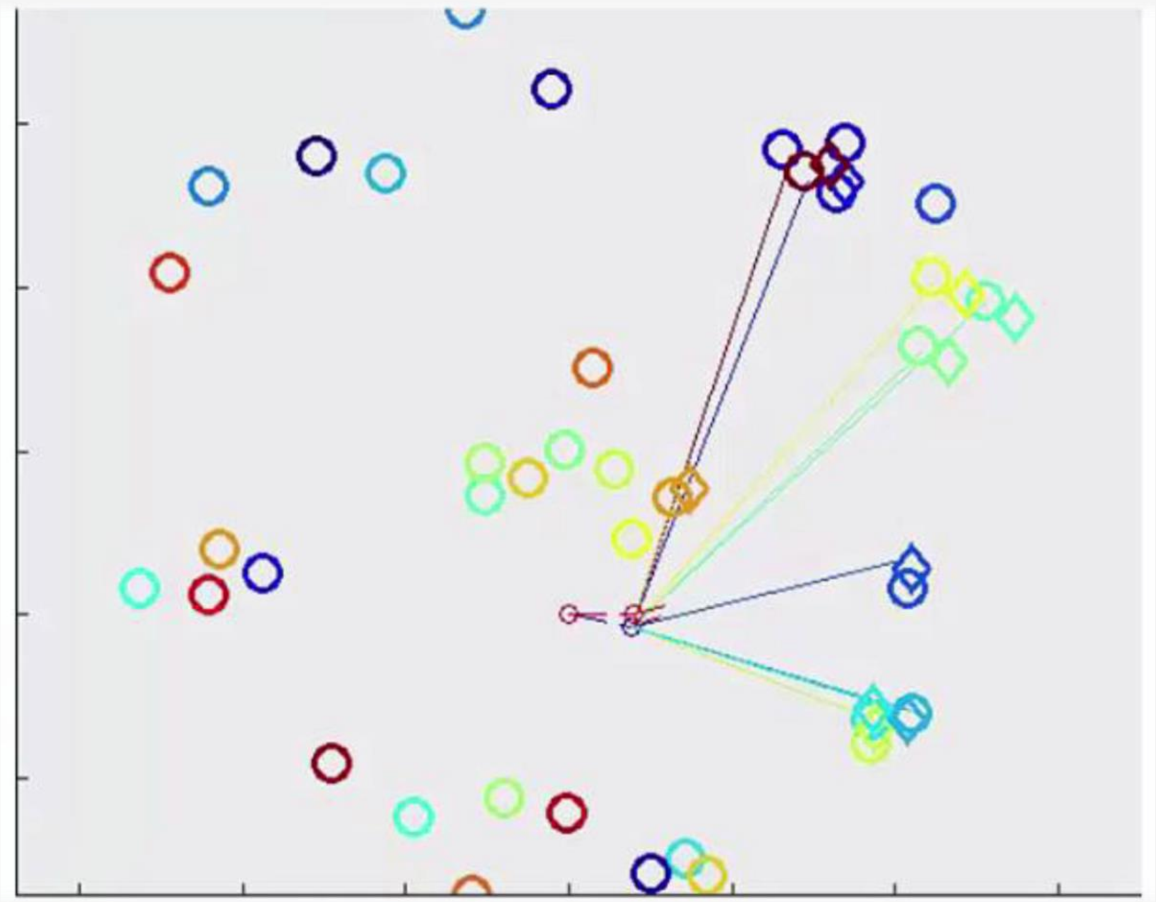
$$\frac{\partial y}{\partial x} = - \sum_i \left[y_t^i - h(x_t)\right]^T\, Q^{-1} \frac{\partial}{\partial x_t^m} [h(x_t)]$$

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

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

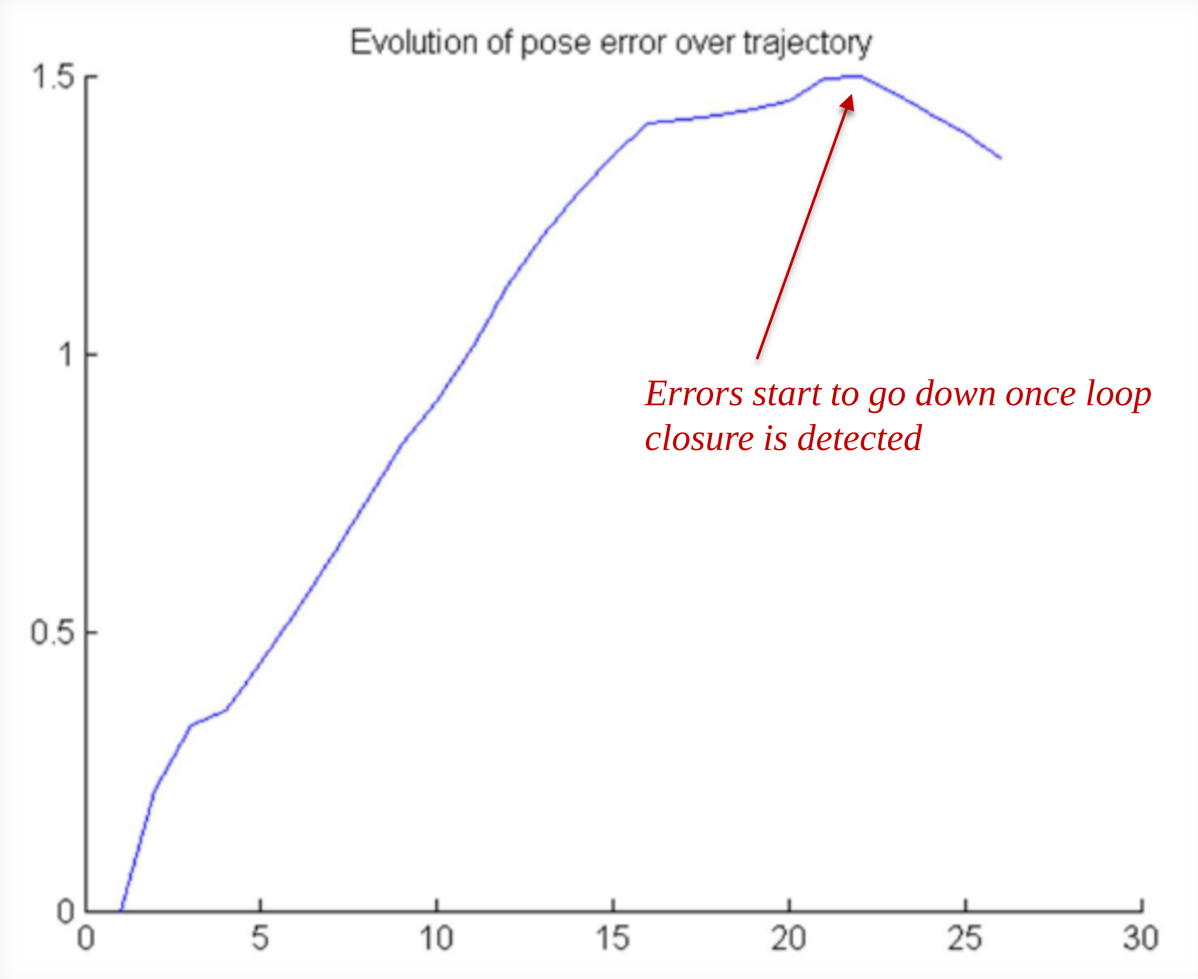
GRAPHSLAM PRELIMINARY RESULTS

- Example (*video*)
  - GraphSLAM: No collision avoidance implemented



Results: Robot going on circles performing GraphSLAM.

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*

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 measurement through registration

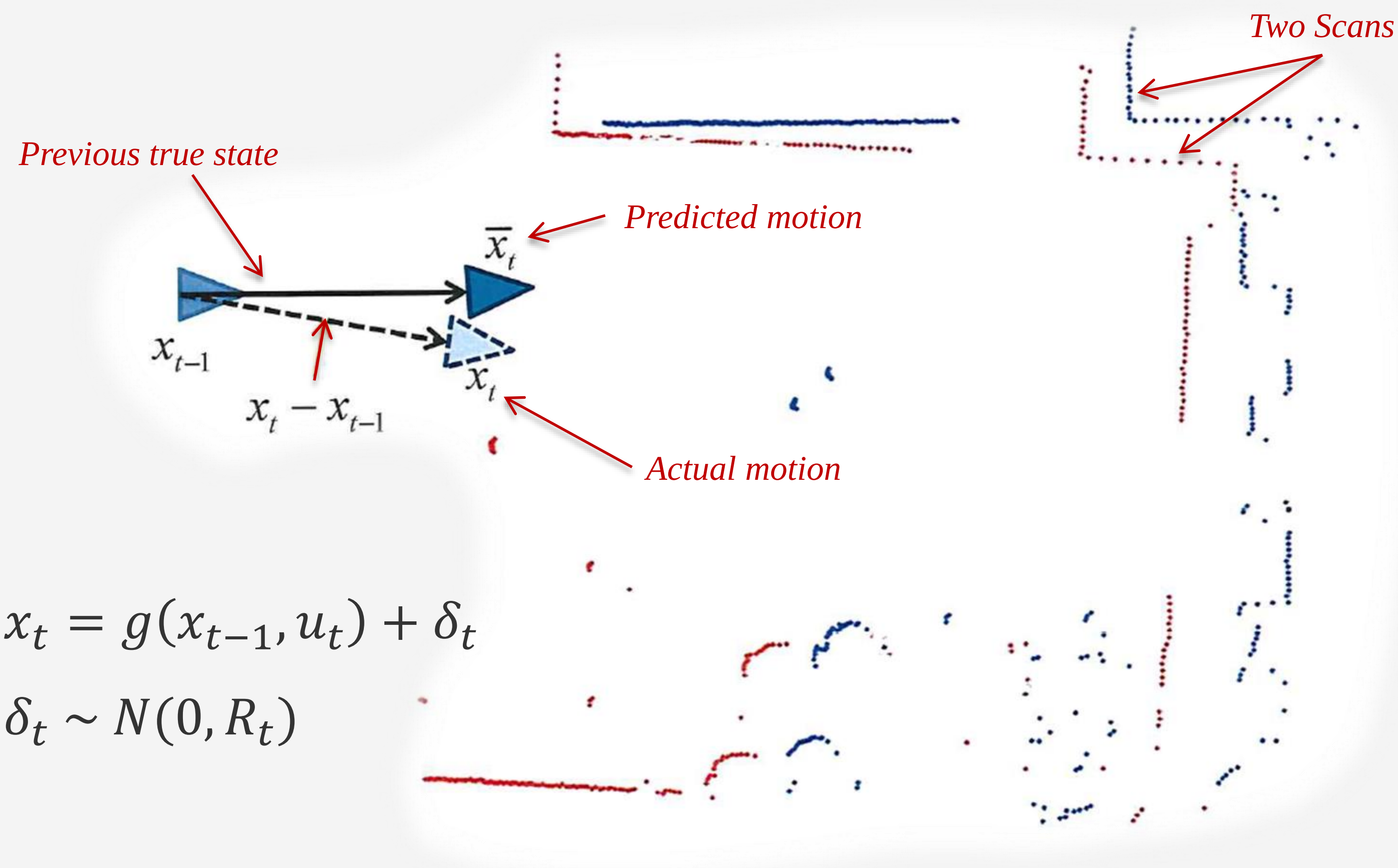
$$x_t^r = \begin{pmatrix} X_t \\ Y_t \\ Z_t \\ \phi_t \\ \theta_t \\ \psi_t \end{pmatrix}, \qquad x_{0:t} = \begin{pmatrix} x_0^r \\ x_1^r \\ \vdots \\ \vdots \\ x_t^r \end{pmatrix}$$

Robot State at time "t"

Full state

GRAPHSLAM WITH SCAN REGISTRATION

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

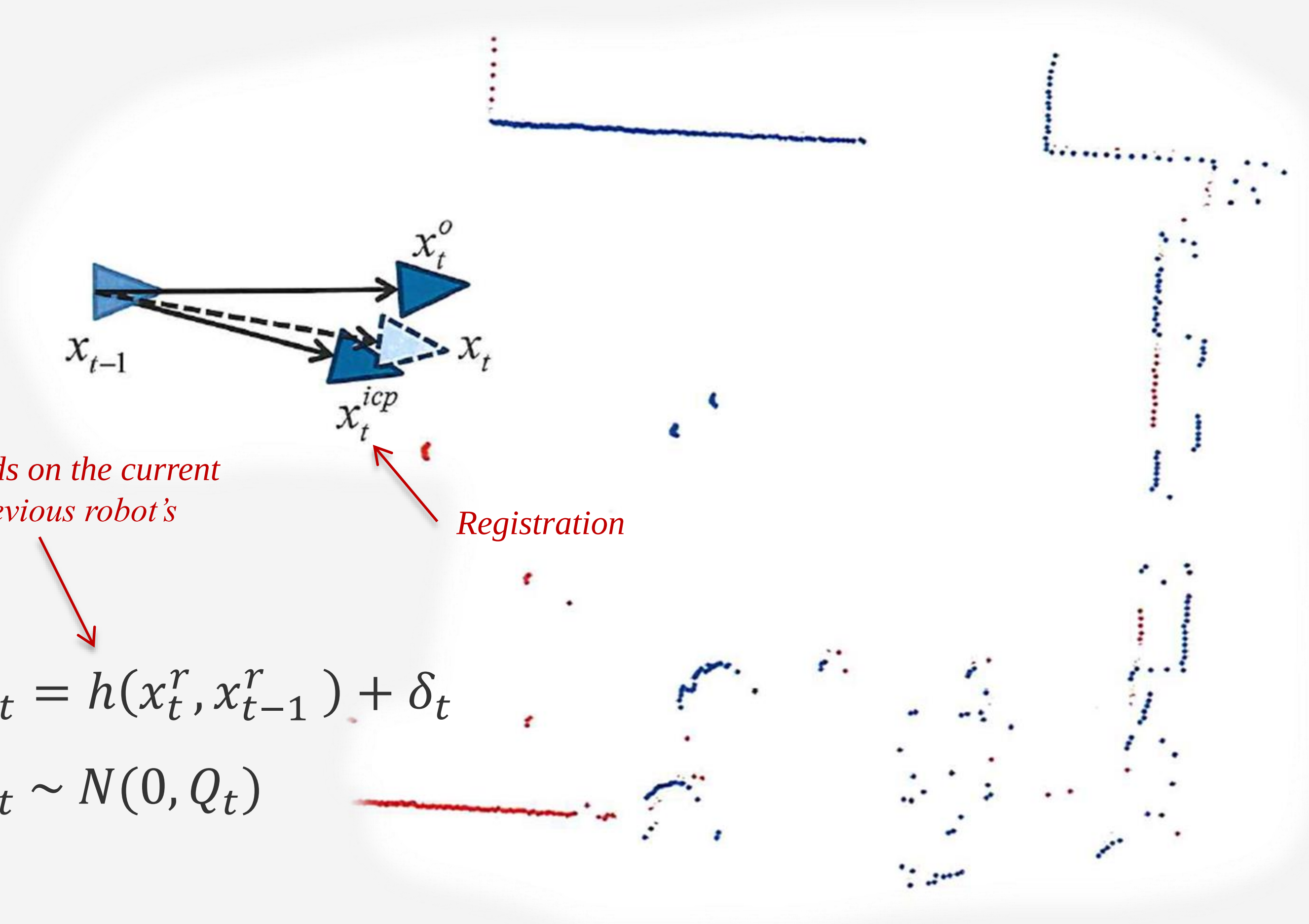


$$x_t = g(x_{t-1}, u_t) + \delta_t$$

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

GRAPHSLAM WITH SCAN REGISTRATION

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



Depends on the current and previous robot's state

Registration

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

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

GRAPHSLAM WITH SCAN REGISTRATION

- Available information
  - Inputs and Motion model

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

  - Measurement and measurement model (*surrogates of the motion model*)

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

    - 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*)

- Thus, the resulting negative log likelihood measurement constraint for each ICP match is:

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

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

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

- 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

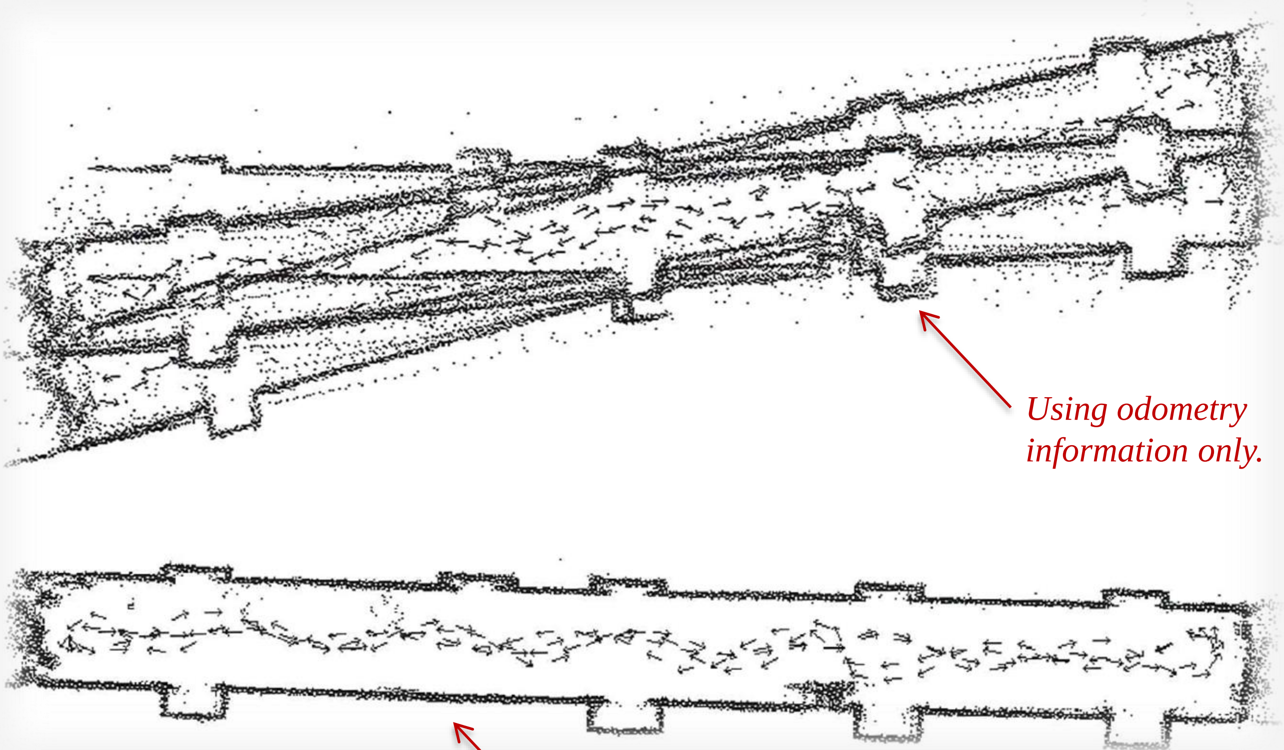
$$\min_{x_{0:t}} J = const. + \sum_{i,j} \left[y_{i,j} - h(x_{i,j})\right]^T Q_{i,j}^{-1} \left[y_{i,j} - h(x_{i,j})\right]$$

Registered pairs of individual scans

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

RESULTS FROM *Lu* AND *Milios*

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

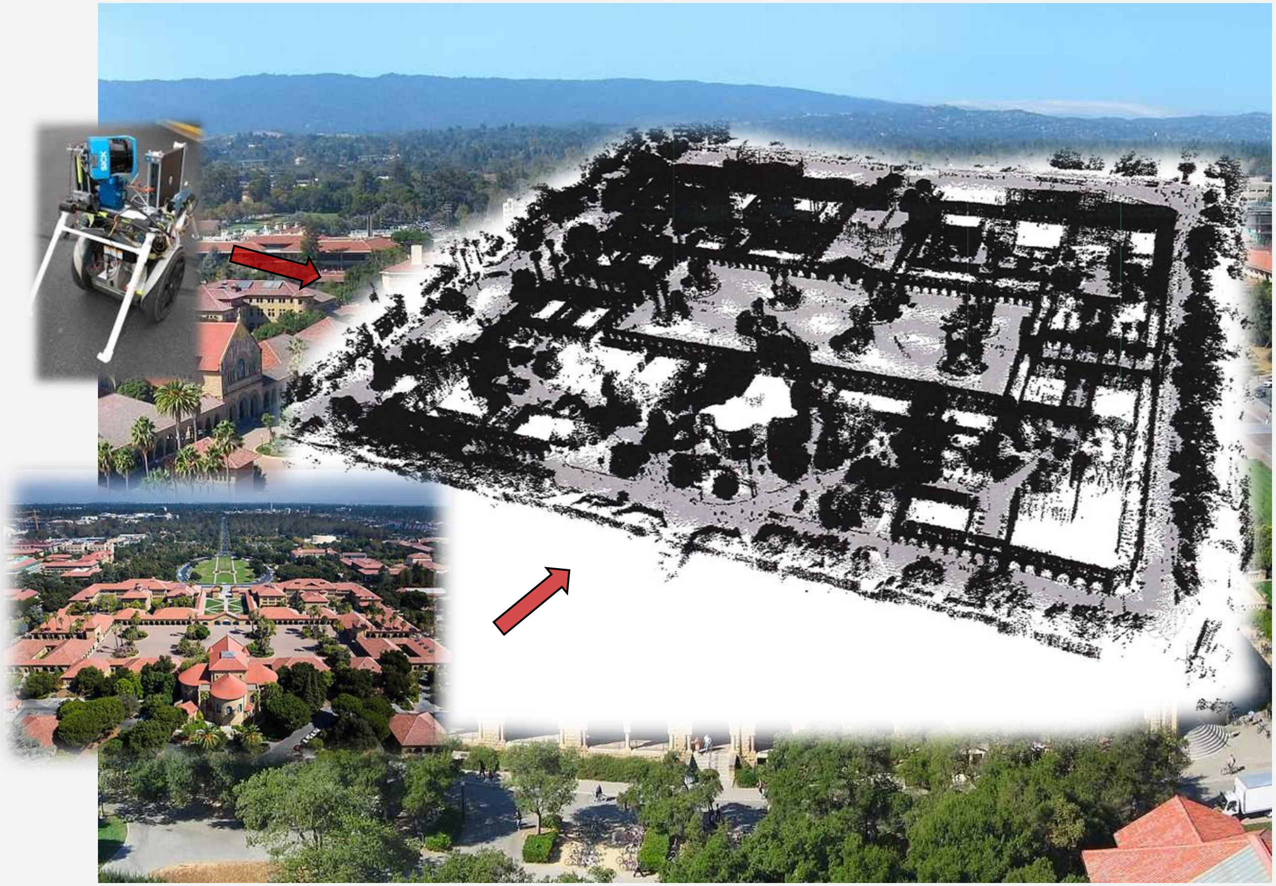


Using odometry information only.

Using every scan recorded and matched with every other scan recorded over the entire motion sequence.

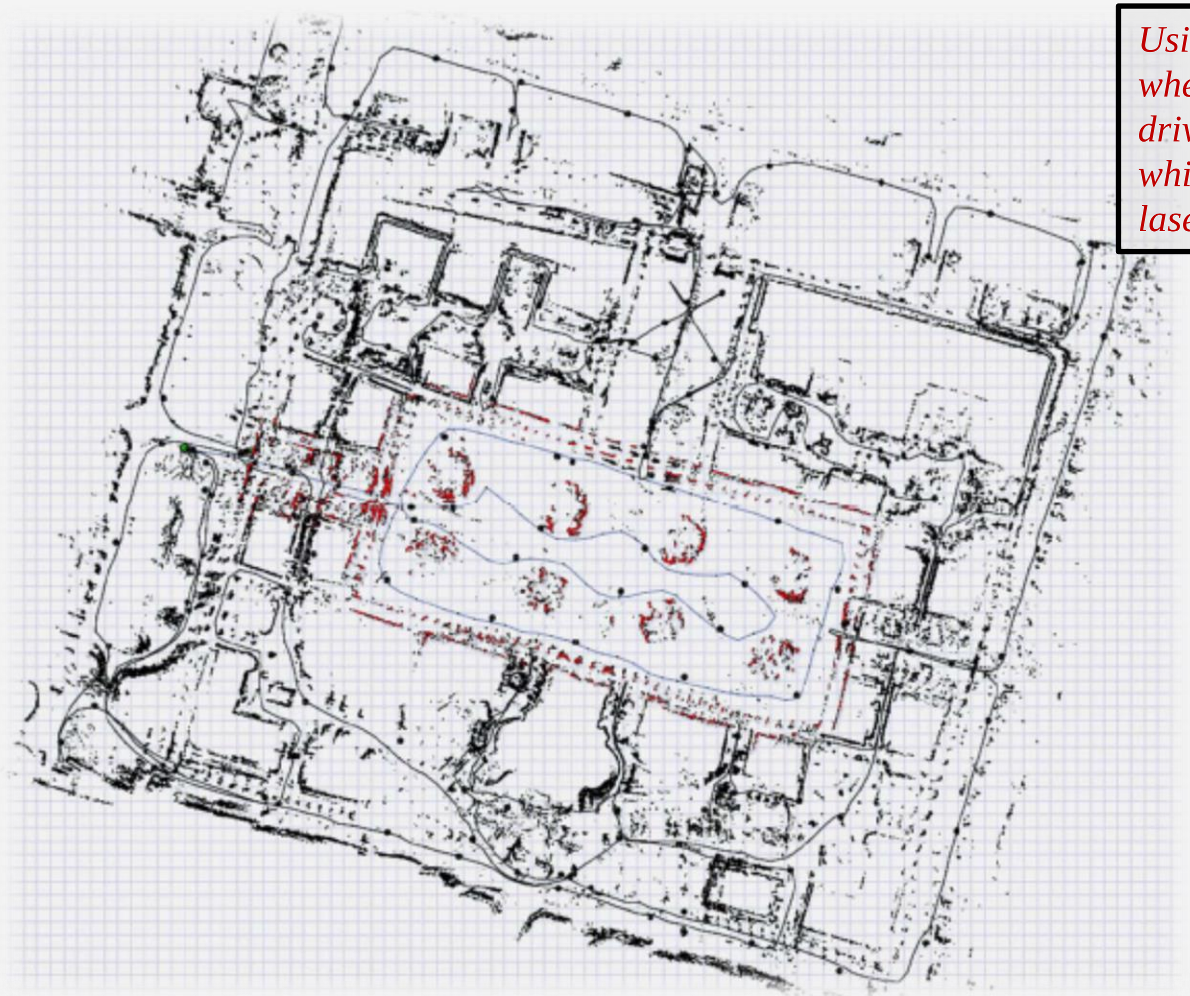
GRAPHSLAM RESULTS

- Example: Same idea applied on the Stanford University campus [*S. Thrun and M. Montemerlo at Stanford in 2006, Intl. J. of Robotic Research, Vol. 25, Issue 5-6, pp. 403-429*]



GRAPHSLAM RESULTS

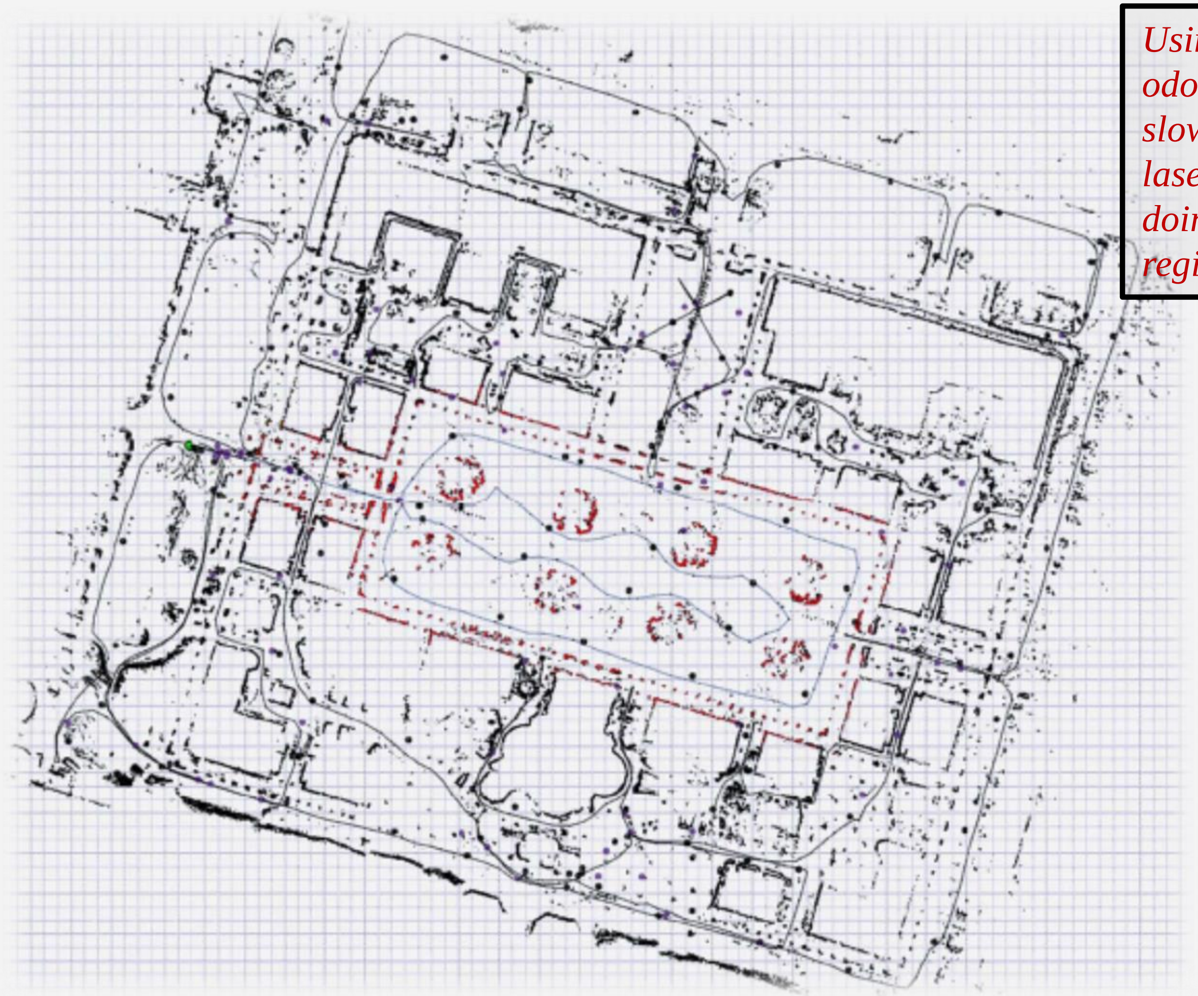
- GPS / Odometry map of Stanford (600m x 600m)



Using GPS and wheel odometry: driving slowly while tilting a 2D laser scanner

GRAPHSLAM RESULTS

- Corrected using GraphSLAM



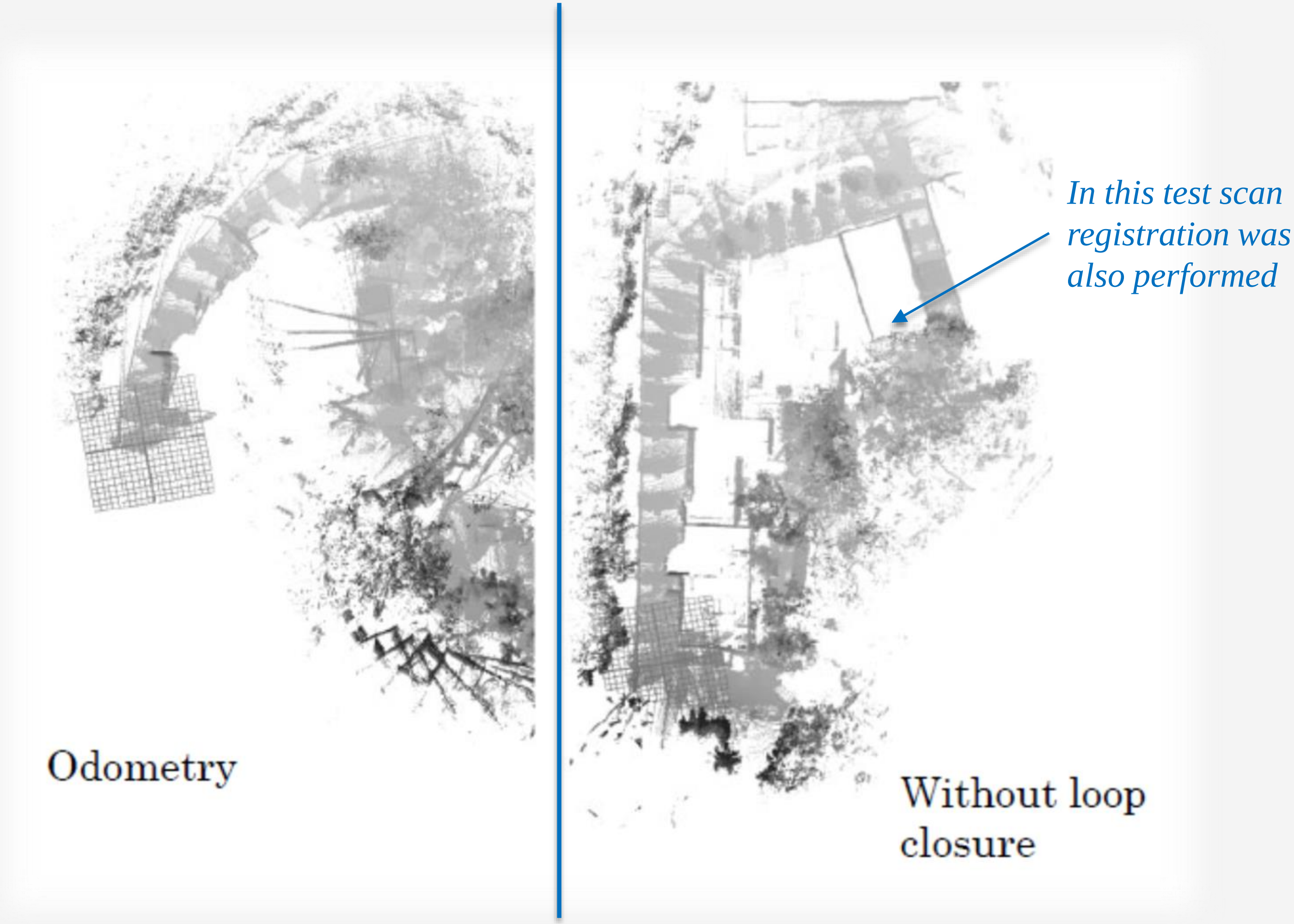
Using same wheel odometry, driving slowly, tiling a 2D laser scanner, and doing scan registration

EXTENSIONS

- 3D GraphSLAM [*Nuchter 2008*]
  - Extension is actually taking the derivation 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*

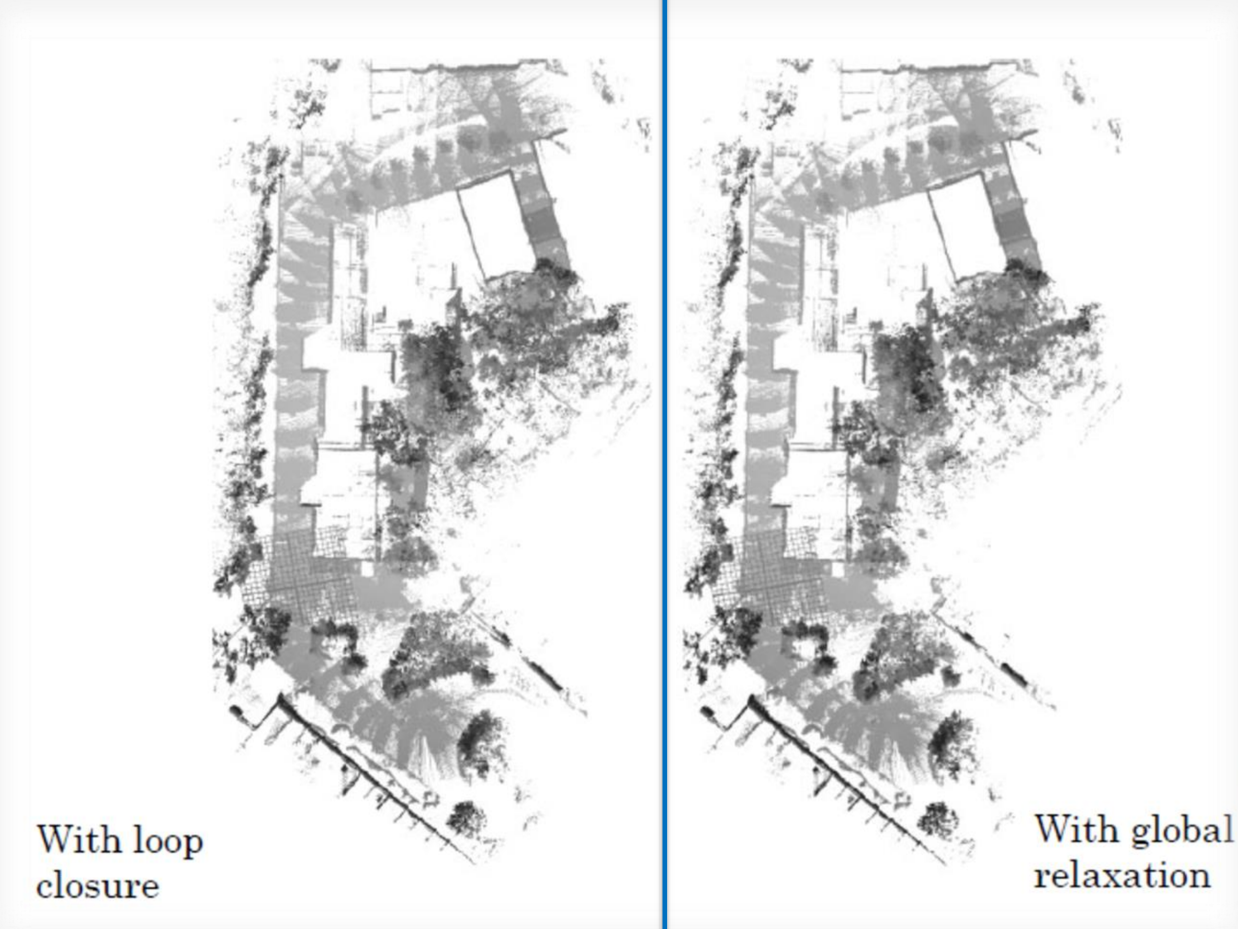
RESULTS FROM NUTCHER

- Large scale outdoor campus mapping without Loop Closure



RESULTS FROM NUTCHER

- Large scale outdoor campus mapping with Loop Closure



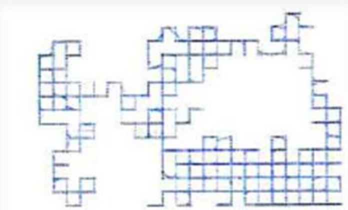
EXTENSIONS BY OLSON

- Re-parameterization [*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 $\rightarrow$ 0 as $\rightarrow \infty$
  - Robustness to local concavities
    - Hop around the state space, "stick" in the best one
  - Good solution very fast "perfect" solution only as t $\rightarrow \infty$

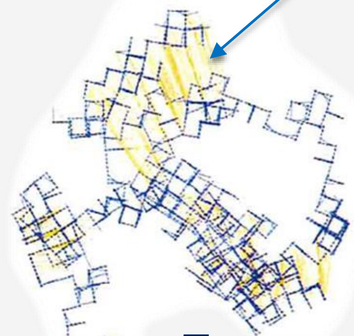
RESULTS FROM OLSON

- *Olson*

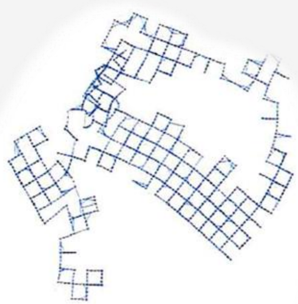
Scan registrations:
odometry version of the map



Ground Truth

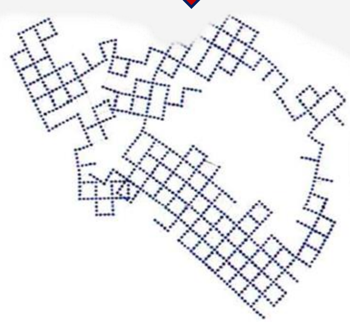


<u>Noisy (simulated) input:</u>
3500 poses
3499 temporal constraints
2100 spatial constraints

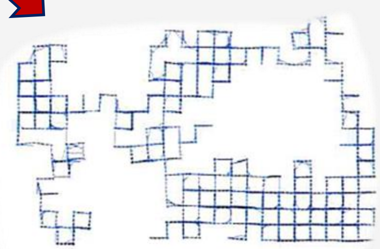


Gauss-Seidel, 60 sec.

Standard optimization method



Multi-Level
Relaxation, 8.6 sec.

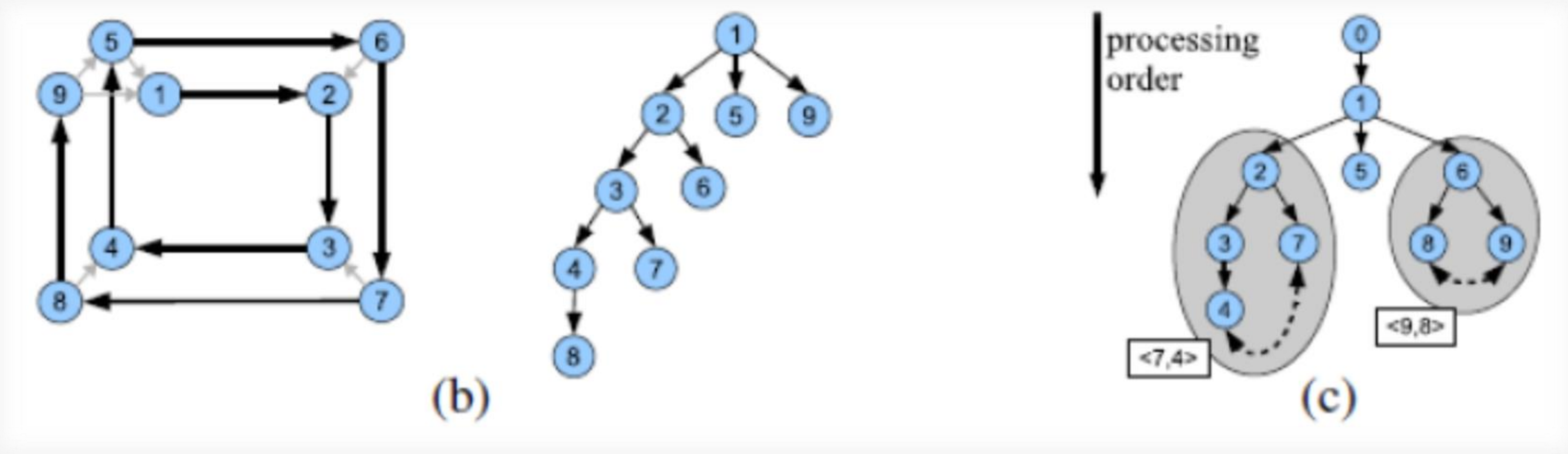


Our method, 2.8 sec.

No stuck in local minima

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



RESULTS FROM *TORO*

- Example: 1000 nodes less than a second to compute

We saw the video on the same environment that took several minutes to generate (see ***Mapping II slides*****,** ***Slide 69*** *– Occupancy Grid SLAM)*

RESULTS FROM *TORO*

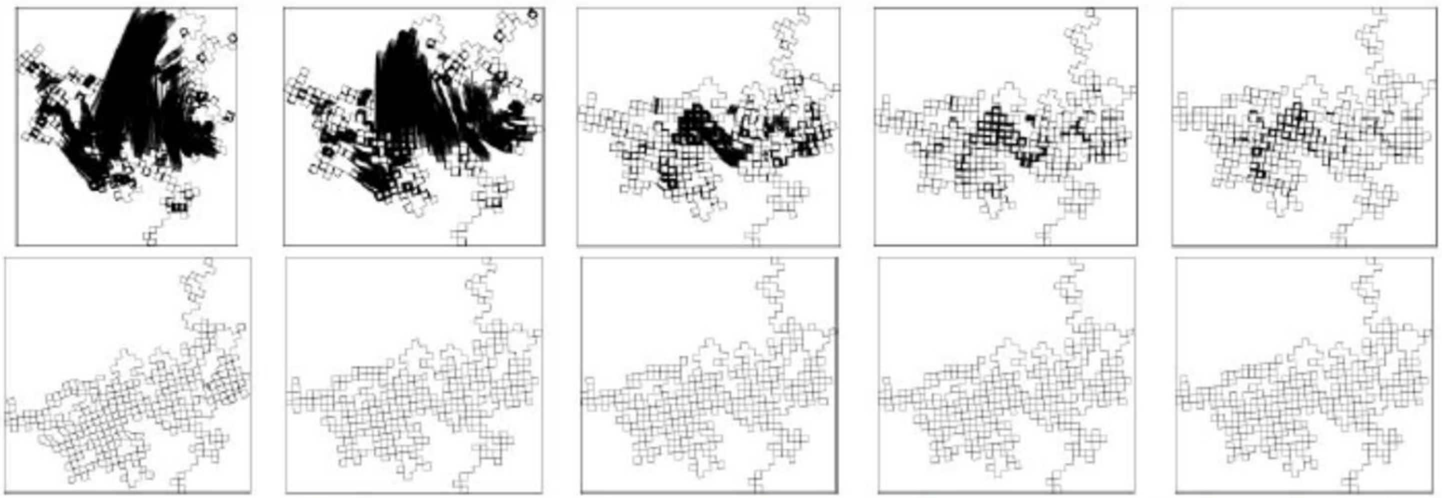
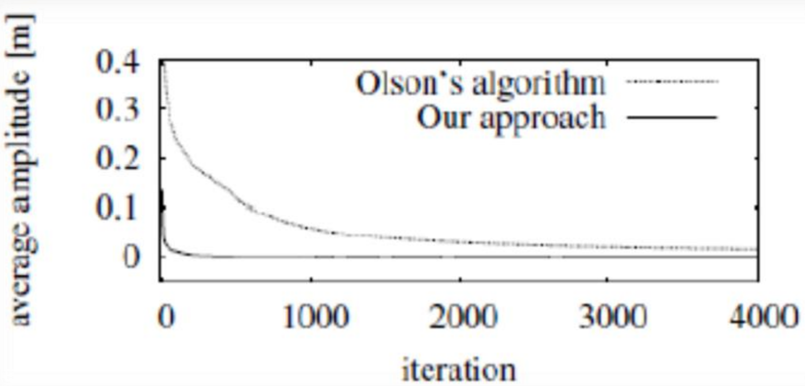
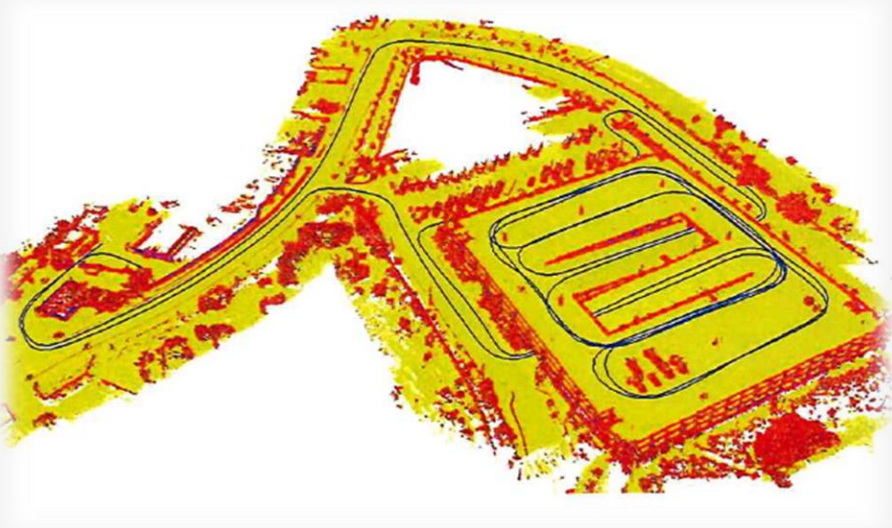


Fig. 4. Results of Olson's algorithm (first row) and our approach (second row) after 1, 10, 50, 100, 300 iterations for a network with 64k constraints. The black areas in the images result from constraints between nodes which are not perfectly corrected after the corresponding iteration (for timings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



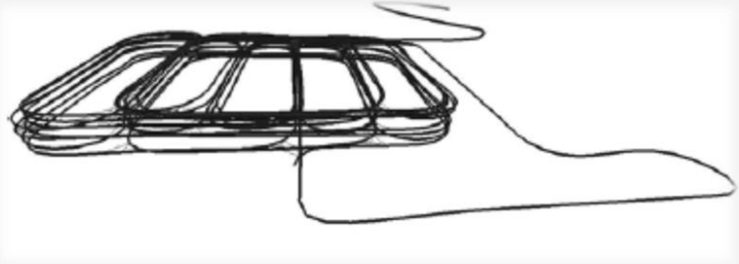
RESULTS FROM *TORO*

Parking Garage



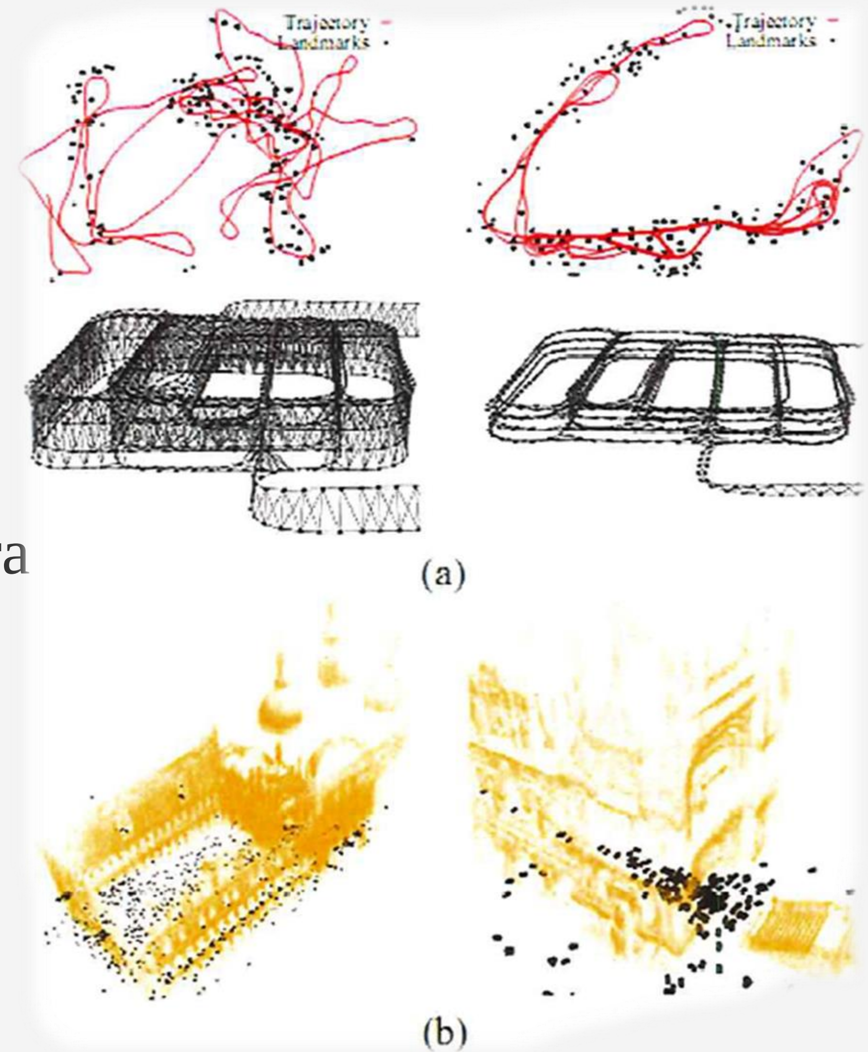
Original

Optimized



CURRENT STANDARD –G2O (*Graphics Based Optimizer – open source*)

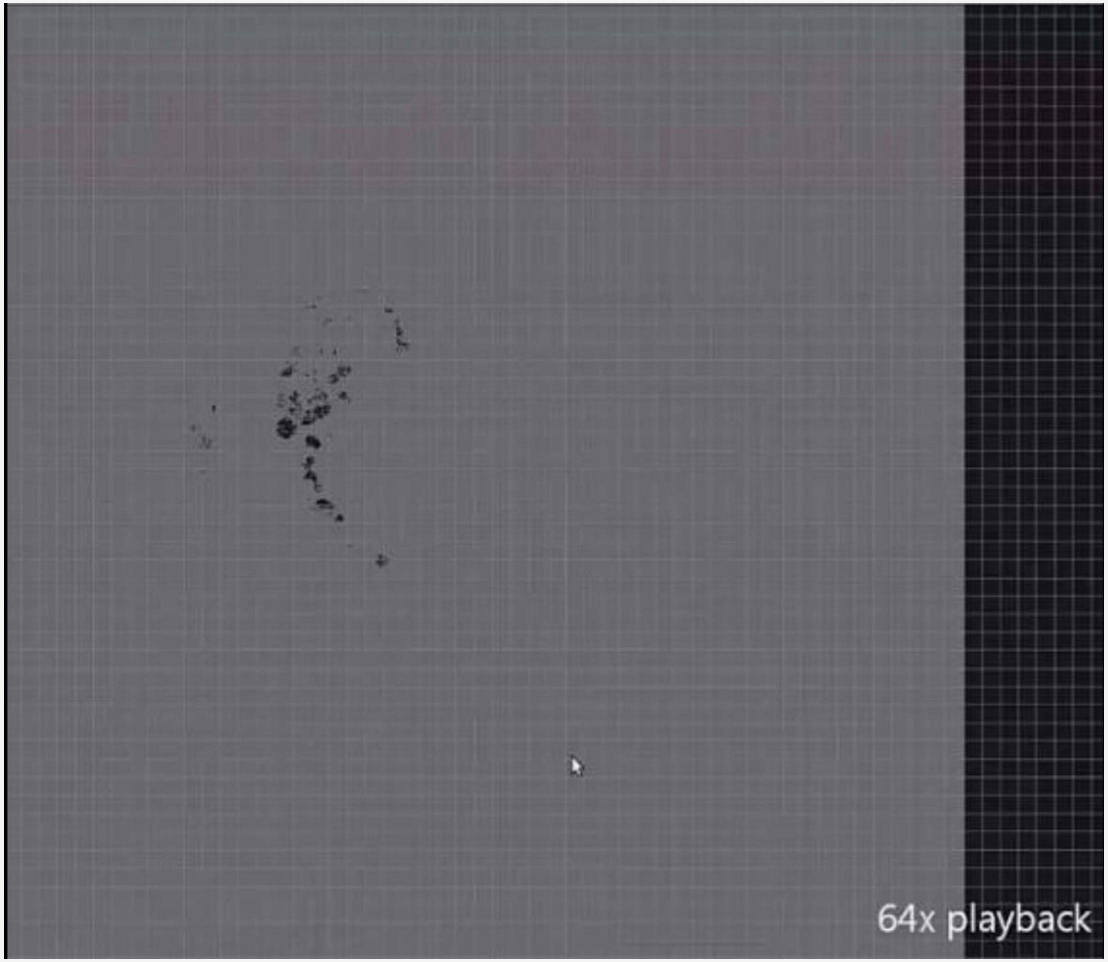
- Fast Backend Solver [*Grisetti, 2011*]
- Takes the best of previous methods
- Works on wide range of problems
- Uses standard linear algebra packages
- Easily extensible, modifiable
- Available on OpenSLAM
- Integrated into ROS



(a)
(b)

RESULTS FROM NASA SAMPLE RETURN

- Example (*video*)
  - NASA Sample return mission



- Individual robot position where a scan was taken
- Scan registration pair between two individual robot scans during its motion though the terrain

Results: Test before the rover was sent to Mars.

STATE OF THE ART [*Oxford Dynamic Robotic Systems Group, 2019*]

- Example (*video*)
  - Real-time large scale dense loop closure with volumetric mapping

Online LiDAR-SLAM for Legged Robots with Robust Registration & Deep-Learned Loop Closure.