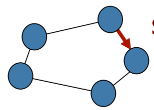
Robot Mapping

TORO – Gradient Descent for SLAM

Cyrill Stachniss

Stochastic Gradient Descent

- Minimize the error individually for each constraint (decomposition of the problem into sub-problems)
- Solve one step of each sub-problem
- Solutions might be contradictory
- The magnitude of the correction decreases with each iteration
- Learning rate to achieve convergence

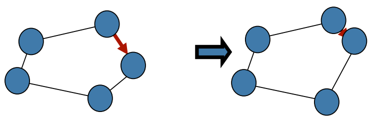


selected constraint

[First used in the SLAM community by Olson et al., '06]

Stochastic Gradient Descent

- Minimize the error individually for each constraint (decomposition of the problem into sub-problems)
- Solve one step of each sub-problem
- Solutions might be contradictory
- The magnitude of the correction decreases with each iteration
- Learning rate to achieve convergence

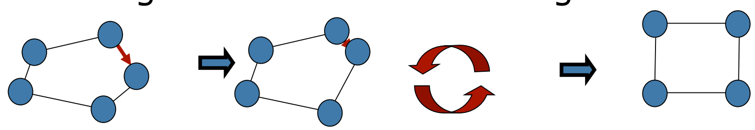


distribute the error over a set of involved nodes

[First used in the SLAM community by Olson et al., '06]

Stochastic Gradient Descent

- Minimize the error individually for each constraint (decomposition of the problem into sub-problems)
- Solve one step of each sub-problem
- Solutions might be contradictory
- The magnitude of the correction decreases with each iteration
- Learning rate to achieve convergence



[First used in the SLAM community by Olson et al., '06]

Preconditioned SGD

- Minimize the error individually for each constraint
- Solve one step of each sub-problem
- A solution is found when an equilibrium is reached
- Update rule for a single constraint:

Previous solution | Hessian | Information matrix

$$\mathbf{x}^{t+1} = \mathbf{x}^{t} + \lambda \mathbf{H}^{-1} \mathbf{J}_{ij}^{T} \boldsymbol{\Omega}_{ij} \mathbf{r}_{ij}$$

Current solution | Learning rate | Jacobian | residual

Node Parameterization

- How to represent the nodes in the graph?
- Impacts which parts need to be updated for a single constraint update
- Transform the problem into a different space so that:
  - the structure of the problem is exploited
  - the calculations become fast and easy

parameters | poses

$$\mathbf{x} = g(\mathbf{p}) \leftrightarrow \mathbf{p} = g^{-1}(\mathbf{x})$$

Mapping function

parameters

$$\mathbf{x}^{*} = \operatorname*{argmin}_{\mathbf{x}} \sum_{i,j} e'_{ij}(\mathbf{x})^{T} \Omega_{ij} e'_{ij}(\mathbf{x})$$

transformed problem

Parameterization of Olson

- Incremental parameterization:

$$x_i = p_i - p_{i-1}$$

parameters | poses

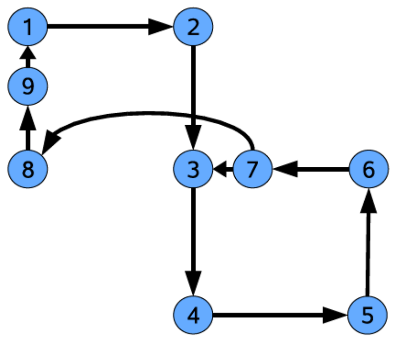
- Directly related to the trajectory
- **Problem:** for optimizing a constraint between the nodes i and k, one needs to updates the nodes i, …, k ignoring the topology of the environment

Alternative Parameterization

- Exploit the topology of the space to compute the parameterization
- Idea: “Loops should be one sub-problem”
- Such a parameterization can be extracted from the graph topology itself

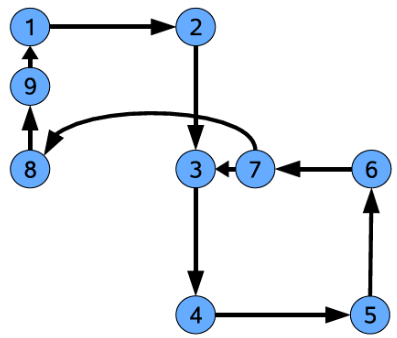
Tree Parameterization

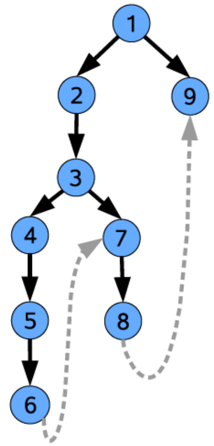
- How should such a problem decomposition look like?



Tree Parameterization

- Use a spanning tree!



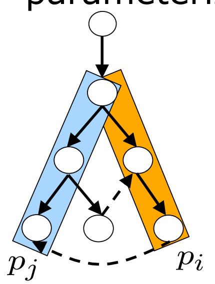


Tree Parameterization

- Construct a spanning tree from the graph
- Mapping between poses and parameters

$$X_i = P_{\text{parent}(i)}^{-1} P_i$$

- Error of a constraint in the new parameterization

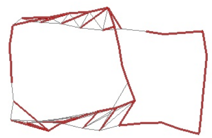


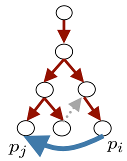
$$E_{ij} = \Delta_{ij}^{-1}\ \text{UpChain}^{-1}\ \text{DownChain}$$

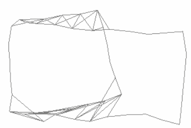
Only variables along the path of a constraint are involved in the update

Stochastic Gradient Descent With The Tree Parameterization

- The tree parameterization leads to several smaller problems which are either:
  - constraints on the tree (“open loop”)
  - constraints not in the tree (“a loop closure”)
- Each SGD equation independently solves one sub-problem at a time
- The solutions are integrated via the learning rate







Computation of the Update Step

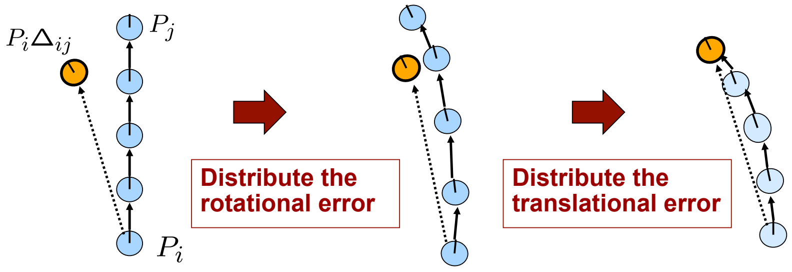
- 3D rotations are non-linear
- Update according to the SGD equation may lead to poor convergence
- SGD update:

$$\mathbf{\Delta x} = \lambda \mathbf{H}^{-1} \mathbf{J}_{ij}^{T} \mathbf{\Omega}_{ij} \mathbf{r}_{ij}$$

- Idea: distribute a fraction of the residual along the parameters so that the error of that constraint is reduced

Computation of the Update Step

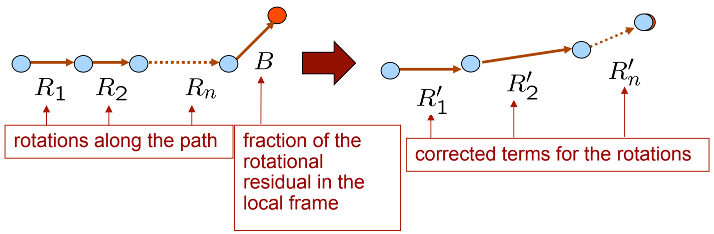
Alternative update in the “spirit” of the SGD: Smoothly deform the path along the constraints so that the error is reduced



Rotational Error

- In 3D, the rotational error cannot be simply added to the parameters because the rotations are not commutative
- Find a set of **incremental** rotations so that the following equality holds:

$$R_1 R_2 \cdots R_n B = R'_1 R'_2 \cdots R'_n$$



Rotational Residual

- Let the first node be the reference frame
- We want a correcting rotation around **a single axis**
- Let A_i be the orientation of the i-th node in the global reference frame

$$A'_n = A_n B$$

Rotational Residual

- Written as a rotation in global frame
  $$A'_n = A_n B = Q A_n$$
- with a decomposition of the rotational residual into a chain of incremental rotations obtained by spherical linear interpolation (slerp)
  $$\begin{aligned} Q &= Q_1 Q_2 \cdots Q_n \\ Q_k &= \text{slerp}(Q, u_{k-1})^T \text{slerp}(Q, u_k) \qquad u \in [0 \ldots \lambda] \end{aligned}$$
- Slerp designed for 3d animations: constant speed motion along a circle

What is the SLERP?

- Spherical LinEar inteRPolation
- Introduced by Ken Shoemake for interpolations in 3D animations
- Constant speed motion along a circle arc with unit radius
- Properties:
  $$\begin{aligned} \mathcal{R}' &:= \mathbf{slerp}(\mathcal{R}, u) \\ \mathbf{axisOf}(\mathcal{R}') &= \mathbf{axisOf}(\mathcal{R}) \\ \mathbf{angleOf}(\mathcal{R}') &= u\ \mathbf{angleOf}(\mathcal{R}) \end{aligned}$$

Rotational Residual

- Given the Q_k, we obtain
  $$A'_k = Q_1 \ldots Q_k A_k = Q_{1:k} A_k$$
- as well as
  $$R'_k = A'^T_{k-1} A'_k$$
- and can then solve:
  $$\begin{aligned} R'_1 &= Q_1 R_1 \\ R'_2 &= (Q_1 R_1)^T Q_{1:2} R_{1:2} = R_1^T Q_1^T Q_1 Q_2 R_1 R_2 \\ &\vdots \\ R'_k &= [(R_{1:k-1})^T Q_k R_{1:k-1}] R_k \end{aligned}$$

Rotational Residual

- Resulting update rule
  $$R'_k = (R_{1:k-1})^T Q_k R_{1:k}$$
- It can be shown that the change in each rotational residual is bounded by
  $$\Delta r'_{k,k-1} \leq |\text{angleOf}(Q_k)|$$
- This bounds a potentially introduced error at node k when correcting a chain of poses including k

How to Determine u_k?

- The u_k describe the distribution of the error

$$Q_k = \text{slerp}(Q, u_{k-1})^T \text{slerp}(Q, u_k) \qquad u \in [0 \ldots \lambda]$$

- Consider the uncertainty of the constraints

$$u_k = \min\left(1, \lambda|\mathcal{P}_{ij}|\right) \left[\sum_{m \in \mathcal{P}_{ij} \wedge m \leq k} d_m^{-1} \right] \left[\sum_{m \in \mathcal{P}_{ij}} d_m^{-1} \right]^{-1}$$

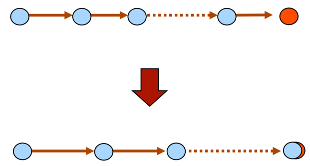
$$d_m = \sum_{\langle l,m \rangle} \min\left[\text{eigen}(\Omega_{lm})\right]$$

all constraints connecting m

- This assumes roughly spherical covariances!

Distributing the Translational Error

- That is trivial
- Just scale the x, y, z movements



Summary of the Algorithm

- Decompose the problem according to the tree parameterization
- Loop:
  - Select a constraint
    - Randomly or sample inverse proportional to the number of nodes involved in the update
  - Compute the nodes involved in update
    - Nodes according to the parameterization tree
  - Reduce the error for this sub-problem
    - Reduce the rotational error (slerp)
    - Reduce the translational error

Complexity

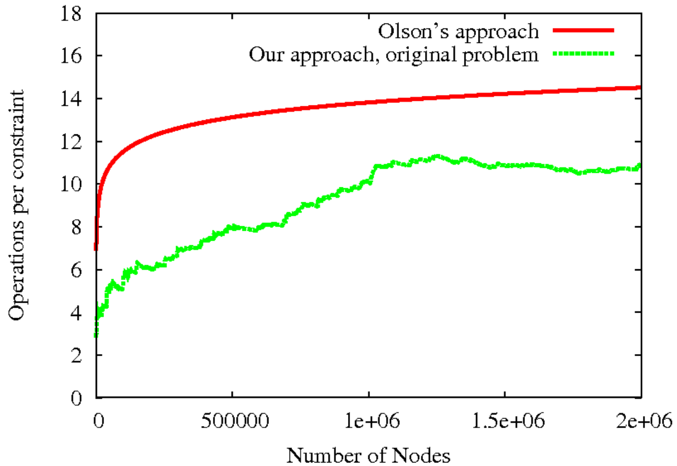
- In each iteration, the approach handles all constraints
- Each constraint optimization requires to update a set of nodes (on average: the average path length according to the tree)

$$\mathcal{O}(Ml)$$

#constraints

avg. path length (parameterization tree)

Cost of a Constraint Update



$\approx \mathcal{O}(M \log(N))$

Node Reduction

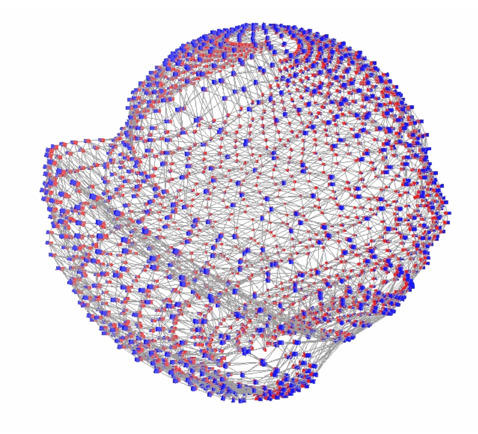
- Complexity grows with the length of the trajectory
- Combine constraints between nodes if the robot is well-localized

$$\Omega_{ij} = \Omega_{ij}^{(1)} + \Omega_{ij}^{(2)}$$

$$z_{ij} = \Omega_{ij}^{-1}\left(\Omega_{ij}^{(1)} z_{ij}^{(1)} + \Omega_{ij}^{(2)} z_{ij}^{(2)}\right)$$

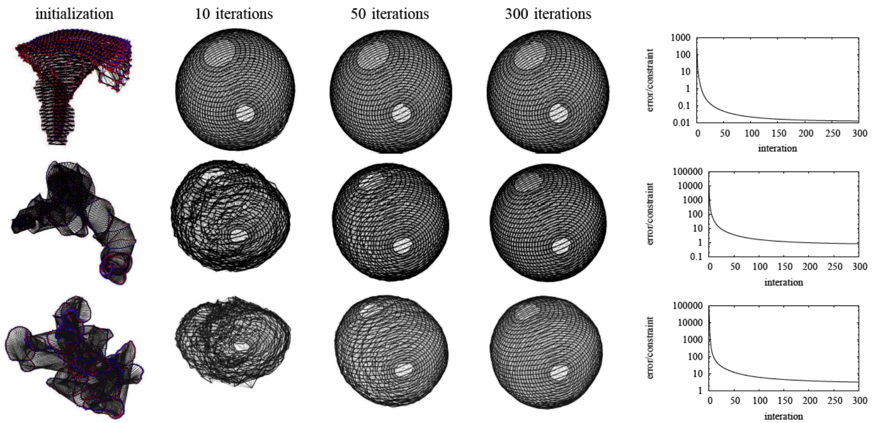
- Similar to adding rigid constraints
- Then, complexity depends on the size of the environment (not trajectory)

Simulated Experiment

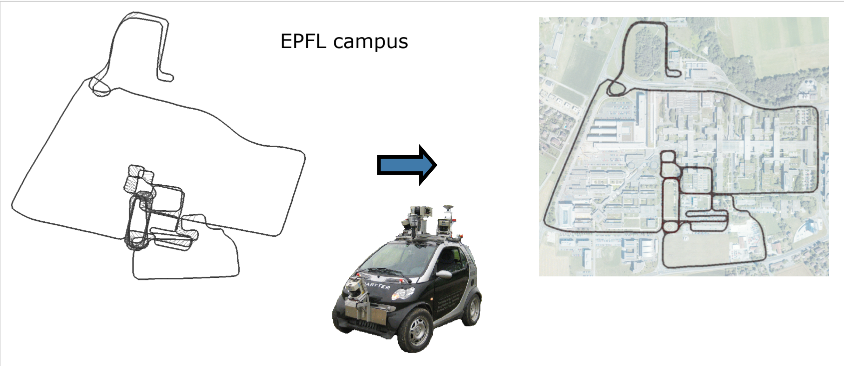


- Highly connected graph
- Poor initial guess
- 2200 nodes
- 8600 constraints

Spheres with Different Noise

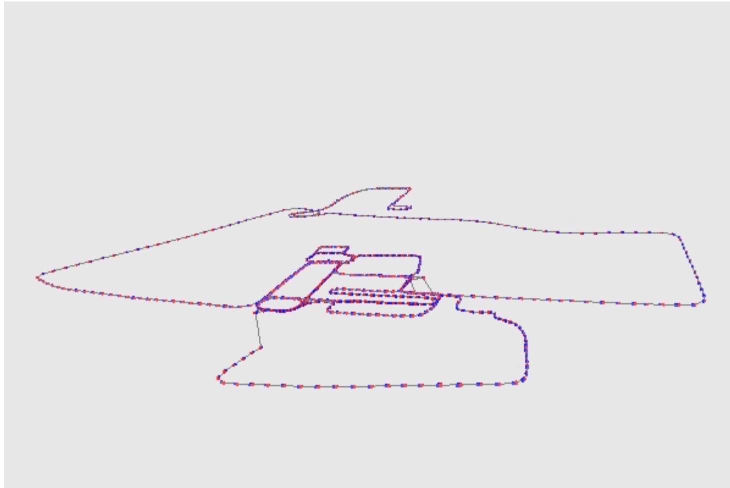


Mapping the EPFL Campus

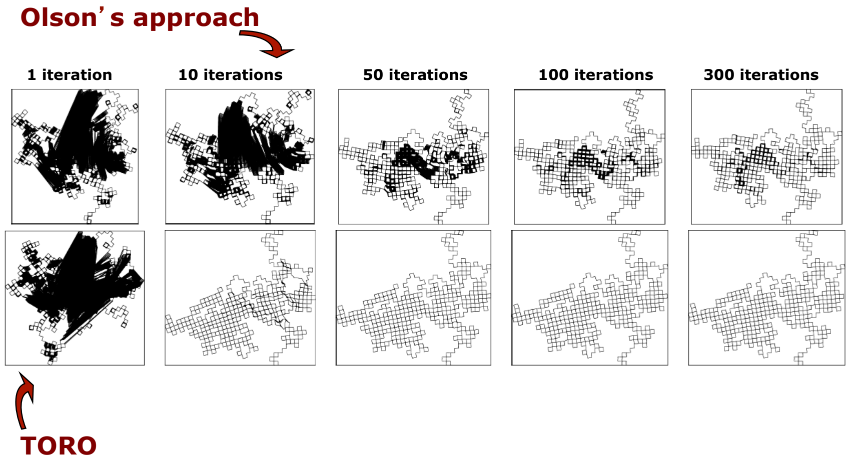


- 10km long trajectory with 3D laser scans

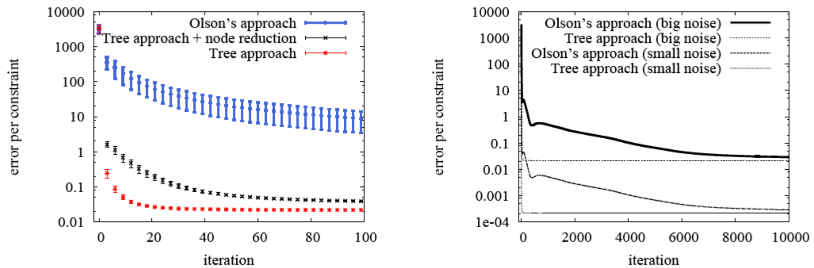
Mapping the EPFL Campus

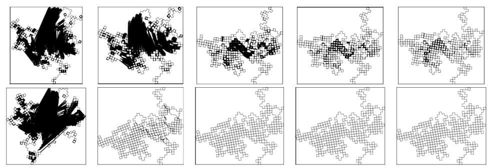


TORO vs. Olson's Approach

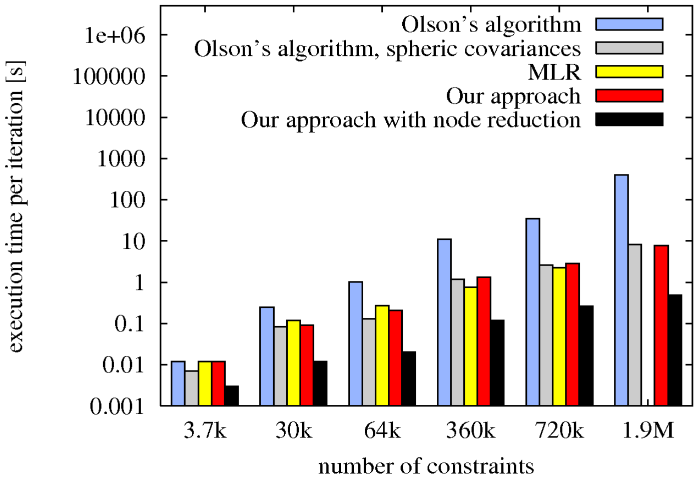


TORO vs. Olson's Approach



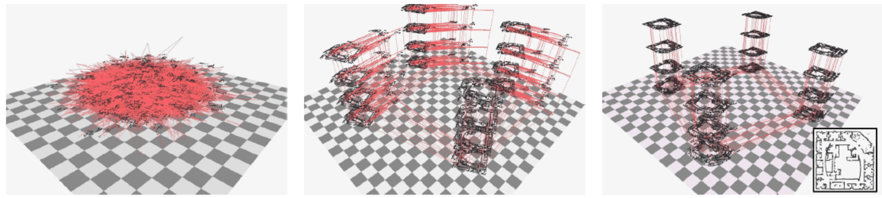


Time Comparison



Robust to the Initial Guess

- Random initial guess
- Intel dataset as the basis for 16 floors distributed over 4 towers



initial configuration | intermediate result | final result (50 iterations)

Drawbacks of TORO

- The slerp-based update rule optimizes rotations and translations separately
- It assume **roughly spherical covariance** ellipses
- Slow convergence speed close to minimum
- No covariance estimates

Conclusions

- TORO - Efficient maximum likelihood estimate for 2D and 3D pose graphs
- Robust to bad initial configurations
- Efficient technique for ML map estimation (or to initialize GN/LM)
- Works in 2D and 3D
- Scales up to millions of constraints
- Available at OpenSLAM.org http://www.openslam.org/toro.html

Literature

SLAM with Stochastic Gradient Descent

- Olson, Leonard, Teller: “Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates”
- Grisetti, Stachniss, Burgard: “Non-linear Constraint Network Optimization for Efficient Map Learning”