# Stochastic optimization for Simultaneous Localization and Mapping

Akshay Kumar

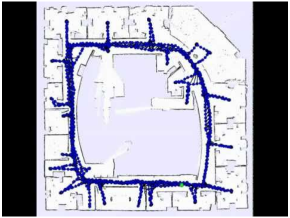
# Table of Contents

# Table of Contents

# Introduction

What we want to do?

- We want to build a 2D/3D map of an environment using a robot.
- We want to localize the position of the robot in the map.



What we need?

- A robot
- Sensors such as LIDAR, camera etc
- A processsing system

Why we want to do?

- Can be used in autonomous navigation and surveillance
- Search and rescue operations

# Table of Contents

# Orthogonal Procrustes Problem

Suppose $A = \{a_i\}$ and $B = \{b_i\}$, $i = 1, .., N$ are two sets of point.

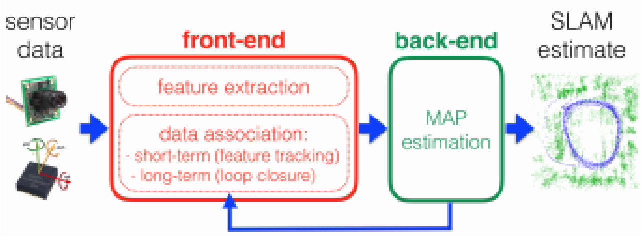
$$R^*, t^* = arg \min_{R,t} \|A - RB - t1\|_F^2$$

$$R^* = UV^T$$

$$t* = \frac{1}{N} \sum_i a_i - Rb_i$$

SLAM consists of two parts - Front-end and Back-end

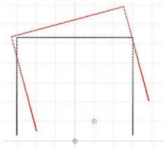
- Front end processes the raw data to generate set of spatial constraint between robot poses.
- Back end finds the optimal robot poses given these spatial constraint.

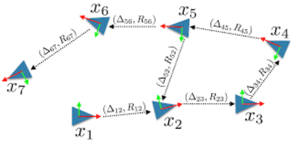


# Front-End

Given set of points,

- $X = (x_1, x_2.....x_n)$
- $Y = (y_1, y_2.....y_n)$

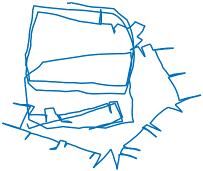




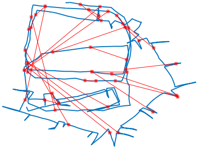

Find R and $\Delta$ which minimizes the error,

$$E(R, \Delta) = \sum_{i=1}^{n} \|y_i - Rx_i - \Delta\|^2$$

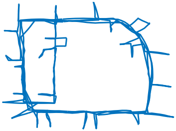
# Why is Back-end necessary?



(a) Odometry



(b) Odometry and Loop closures



(c) After optimization

# Back-end

- Output of front end is network of spatial constraints between poses.
- Back end optimizes on this network to find global poses.
- Let $\mathbf{x_i} = [\mathbf{t_i}, \theta_\mathbf{i}]^\mathbf{T}$ be robot pose in global frame.
- Let $\mathbf{z_{ij}} = [\mathbf{\Delta_{ij}}, \theta_\mathbf{ij}]^\mathbf{T}$ be spatial constraint between $i$th and $j$th pose.
- Measurement model, $f(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{z_{ij}} + \mathbf{e_{ij}}$, $\mathbf{e_{ij}} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Omega_{ij}})$

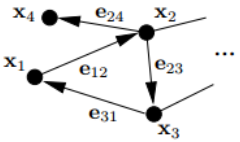
# Error function

- Gaussian error. MAP Estimate,

$$F(\mathbf{x}) = \sum_{i,j} e_{ij}^T \Omega_{ij} e_{ij}$$

$$\mathbf{x}^* = arg \min_{\mathbf{x}} F(\mathbf{x})$$

$$e_{ij} = \begin{pmatrix} R(\theta_i)^T (\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \\ <\theta_j - \theta_i>_{2\pi} - \theta_{ij} \end{pmatrix} \quad (1)$$



# Table of Contents

# Problem Statement

## Objective

Given spatial constraint, optimize and obtain robot poses.

$$\mathbf{x}^* = \mathbf{arg}\min_{\mathbf{x}} \sum_{\mathbf{i,j}} \begin{pmatrix} R(\theta_i)^T(\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \\ <\theta_j - \theta_i>_{2\pi} - \theta_{ij} \end{pmatrix}^{\mathbf{T}} \mathbf{\Omega_{ij}} \begin{pmatrix} R(\theta_i)^T(\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \\ <\theta_j - \theta_i>_{2\pi} - \theta_{ij} \end{pmatrix} \quad (2)$$

# Table of Contents

# Definition

Consider the optimization problem,

$$\min_{x \in \chi} F(x_1, x_2, ...., x_s)$$

where, $F$ is block multi-convex i.e. $F$ is convex function in block $x_i$ if other blocks are fixed. How to solve?

- Assume an initial guess for $x_2, .., x_s$.
- Find the value of $x_1$ which minimizes $F$.
- Update value of $x_1$
- Do this for all $x_i$
- Repeat until convergence.

# Re-formulation of SLAM

Any spatial constraint can be expressed as,

$$e_{ij} = \begin{pmatrix} R(\theta_i)^T(\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \\ <\theta_j - \theta_i>_{2\pi} - \theta_{ij} \end{pmatrix} \tag{3}$$

Since the angular constraint is a major problem. We try to replace it with another distance constraint. Let us say,

$$R(\theta_j)^T(\mathbf{t_i} - \mathbf{t_j}) = \mathbf{\Delta}'_{\mathbf{ij}}$$

Let us use the angular constraint and write $\theta_j = \theta_i + \theta_{ij}$. We get,

$$R(\theta_{ij})^T R(\theta_i)^T(\mathbf{t_i} - \mathbf{t_j}) = \mathbf{\Delta}'_{\mathbf{ij}}$$

Using the distance constraint we have,

$$-R(\theta_{ij})^T \mathbf{\Delta_{ij}} = \mathbf{\Delta}'_{\mathbf{ij}}$$

Hence implicitly using angular constraint we have converted it into distance constraint. Re-formulated error function,

$$e_{ij} = \begin{pmatrix} R(\theta_i)^T(\mathbf{t_j}-\mathbf{t_i})-\mathbf{\Delta_{ij}} \\ R(\theta_j)^T(\mathbf{t_i}-\mathbf{t_j})-\mathbf{\Delta'_{ij}} \end{pmatrix} \tag{4}$$

Optimization problem becomes,

$$\mathbf{x}^* = \mathbf{arg}\min_{\mathbf{x}} \sum_{\mathbf{i,j}} \left\| \mathbf{R}(\theta_\mathbf{i})^\mathbf{T}(\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \right\|^\mathbf{2} + \left\| \mathbf{R}(\theta_\mathbf{j})^\mathbf{T}(\mathbf{t_i} - \mathbf{t_j}) - \mathbf{\Delta'_{ij}} \right\|^\mathbf{2} \tag{5}$$

We will use BCD and iterate over $\theta$ and $\mathbf{t}$. Given $\theta$ finding $\mathbf{t}$ is trivial.

# Estimating Angles

In the optimization problem each $\theta_i$ is separable from each other. We can optimize them individually.

$$\theta_i^* = arg \min_{\theta i} \sum_j \left\| R(\theta_i)^T (\mathbf{t_j} - \mathbf{t_i}) - \mathbf{\Delta_{ij}} \right\|^2$$

$$= \sum_j -2(\mathbf{t_j} - \mathbf{t_i})^{\mathbf{T}} \mathbf{R}(\theta_\mathbf{i}) \mathbf{\Delta_{ij}}$$

$$= \sum_j -2(\mathbf{t_j} - \mathbf{t_i})^{\mathbf{T}} \begin{bmatrix} c_i & -s_i \\ s_i & c_i \end{bmatrix} \begin{bmatrix} \Delta_{xij} \\ \Delta_{yij} \end{bmatrix}$$

$$= \sum_j -2(\mathbf{t_j} - \mathbf{t_i})^{\mathbf{T}} \begin{bmatrix} \Delta_{xij} & -\Delta_{yij} \\ \Delta_{yij} & \Delta_{xij} \end{bmatrix} \begin{bmatrix} c_i \\ s_i \end{bmatrix}$$

$$= {f_i}^T \begin{bmatrix} c_i \\ s_i \end{bmatrix}$$

Hence,

$$\begin{bmatrix} c_i^* \\ s_i^* \end{bmatrix} = -f_i / \|f_i\|$$

# Table of Contents

# Trust Region Method

$$\mathbf{x}^* = \mathbf{arg}\min_{\mathbf{x}} \sum_{\mathbf{i,j}} \left\| \mathbf{R}(\theta_\mathbf{i})^\mathbf{T}(\mathbf{t_j} - \mathbf{t_i}) - \boldsymbol{\Delta}_\mathbf{ij} \right\|^\mathbf{2} \tag{6}$$

$$\mathbf{x}^* = \mathbf{arg}\min_{\mathbf{x}} \sum_{\mathbf{i,j}} \|(\mathbf{t_j} - \mathbf{t_i}) - \mathbf{R}(\theta_\mathbf{i})\boldsymbol{\Delta}_\mathbf{ij}\|^\mathbf{2}$$

$$\mathbf{x}^* = \mathbf{arg}\min_{\mathbf{x}} \sum_{\mathbf{i,j}} \left\| (\mathbf{t_j} - \mathbf{t_i}) - \begin{bmatrix} c_i & -s_i \\ s_i & c_i \end{bmatrix} \begin{bmatrix} \Delta_{xij} \\ \Delta_{yij} \end{bmatrix} \right\|^\mathbf{2}$$

where, $c_i^2 + s_i^2 = 1 \quad \forall i = 1, ..., N$

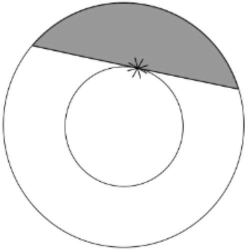
In the above equation, objective is convex while constraint is non-convex.

Hence, iterative convexification of the constraint set.

# Trust Region Method

Let $c_i^t$ and $s_i^t$ be the estimate of $c_i$ and $s_i$ at $t$th iteration.
Then approximate convex set at $(t+1)$th iteration is given by,

$$c_i^2+s_i^2 = 1 \Rightarrow \{c_i^2 + s_i^2 \leqslant 1+\epsilon\} \cap \{2c_ic_i^t + 2s_is_i^t - (c_i^t)^2 - (s_i^t)^2 \geq 1\}$$



# Table of Contents

# Long term SLAM

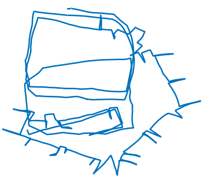
- In applications such as underwater exploration for monitoring robot operates for a long time.
- Requires high computational resources.
- Current research focused on algorithm development for efficient long term SLAM.

# Stochastic optimization for SLAM

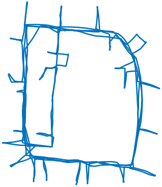
- The graph of N nodes is incrementally processed.
- We optimize first k $<<$ N nodes giving us the pose estimate $\hat{X}_{1:k}$.
- New node (k+1) is added in the graph.
- Choose a $p(< k)$ sized random sub-graph.
- Optimize sub-graph using one iteration of trust region method under $\epsilon$-boundary constraint.
- Repeat until all nodes are covered.

# Table of Contents

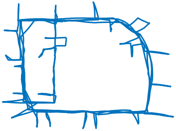
# Intel Dataset



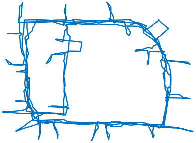
(a) Odometry



(b) g2o

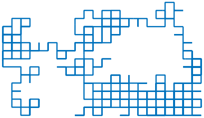


(a) Deterministic

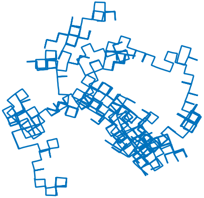


(b) Stochastic

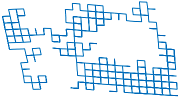
# M3500



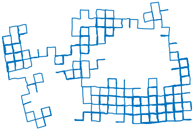
(a) Ground truth



(b) Odometry

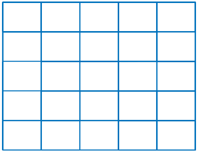


(a) Deterministic

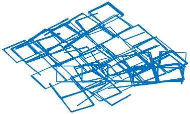


(b) Stochastic

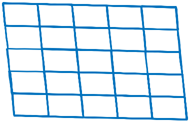
# M3000



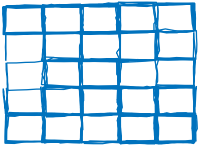
(a) Ground truth



(b) Odometry



(a) Deterministic



(b) Stochastic

Table: M3500 Dataset

| | g2o | Det. | Stoc. |
|---|---|---|---|
| Translation error | $7.06 \cdot 10^{-4}$ | $8.16 \cdot 10^{-4}$ | $4.2 \cdot 10^{-2}$ |
| Rotational error | $2.71 \cdot 10^{-4}$ | $2.73 \cdot 10^{-4}$ | $2.3 \cdot 10^{-3}$ |

Table: M3000 Dataset

| | g2o | Det. | Stoc. |
|---|---|---|---|
| Translation error | $5.77 \cdot 10^{-4}$ | $6.65 \cdot 10^{-4}$ | $2.51 \cdot 10^{-2}$ |
| Rotational error | $1.72 \cdot 10^{-4}$ | $1.96 \cdot 10^{-4}$ | $1.1 \cdot 10^{-3}$ |

# Questions ?

# Thank You