

# Graph SLAM

**slam\_toolbox**

2026-01-30

Alistair Keiller

# Contents



|     |  |    |
|-----|--|----|
| 1   | Pose graph SLAM .....                    | 4  |
| 1.a | Credit .....                             | 5  |
| 1.b | The pose graph .....                     | 6  |
| 1.c | Create an Edge If... (1) .....           | 7  |
| 1.d | Odometry .....                           | 8  |
| 1.e | Create an Edge If... (2) .....           | 9  |
| 1.f | Lidar .....                              | 10 |
| 1.g | Idea of Pose Graph-based SLAM .....      | 11 |
| 1.h | Idea of Pose Graph-based SLAM pt 2 ..... | 12 |
| 1.i | Graphical Explanation .....              | 13 |
| 1.j | Solving with least squares .....         | 14 |
| 1.k | Least Squares Approach .....             | 15 |
| 1.l | Pose Graph .....                         | 16 |
| 1.m | Minimization .....                       | 17 |
| 1.n | Removing outliers .....                  | 18 |
| 2   | Scan matching .....                      | 19 |

# Contents (ii)



|     |                               |    |
|-----|-------------------------------|----|
| 2.a | What is scan matching .....   | 20 |
| 2.b | Iterative Closest Point ..... | 21 |
| 2.c | Data Association .....        | 22 |
| 2.d | Transformation .....          | 23 |
| 2.e | Iterate .....                 | 24 |
| 3   | All Together .....            | 25 |
| 3.a | Loop Closure .....            | 26 |
| 3.b | Slam Toolbox .....            | 27 |

# **1 Pose graph SLAM**

# Credit



Wolfram Burgard, Giorgio Grisetti, and Cyrill Stachniss: <http://ais.informatik.uni-freiburg.de/teaching/ws11/robotics2/pdfs/ls-slam-tutorial.pdf>



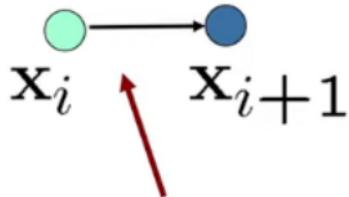
# The pose graph

- It consists of  $n$  nodes  $x = x_{1:n}$
- Each  $x_i$  is a robot pose (at time  $t_i$ )
- We create an edge between nodes  $x_i$  and  $x_j$  if and only if...



## Create an Edge If... (1)

- ...The robot moves from  $x_i$  to  $x_{i+1}$
- Edge corresponds to odometry data

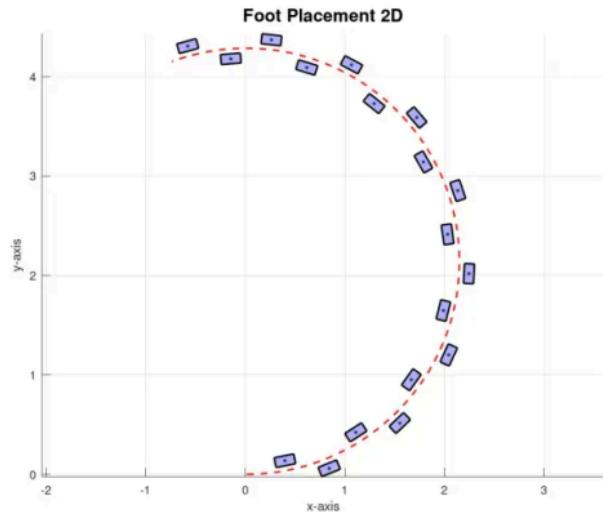


The edge represents the  
**odometry** measurement

# Odometry



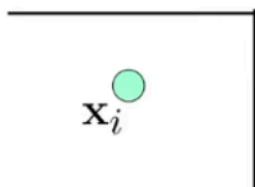
- Constraints connect the poses of the robot while it is moving using odometry
- Constraints are inherently uncertain



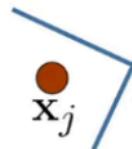


## Create an Edge If... (2)

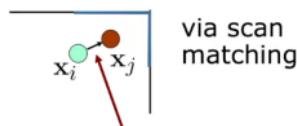
- The robot observes the same part of the environment in both  $x_i$  and  $x_j$ .



Measurement from  $x_i$



Measurement from  $x_j$

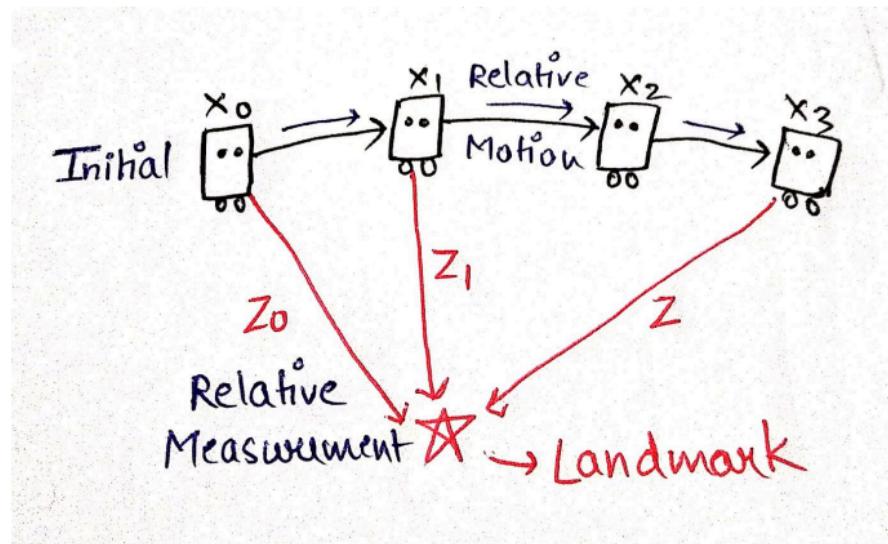


Edge represents the position of  $x_j$  seen  
from  $x_i$  based on the **observation**

# Lidar



- Observing previously seen areas generates new constraints



# Idea of Pose Graph-based SLAM



- **Graph:** represents the problem
- **Node:** corresponds to an estimated pose in the robot at a given time
- **Edge:** an approximate spatial constraints between two nodes.
- **Graph-based SLAM:** Build the graph and find a node configuration that minimizes the error introduced by the constraints

# Idea of Pose Graph-based SLAM pt 2



- The nodes represent the state vector
- Given a state, we can compute what we expect to perceive.
- We have real observations that relate nodes to each other
- **Goal:** Find a configuration of the nodes so that the real and predicted observations are as similar as possible.

# Graphical Explanation



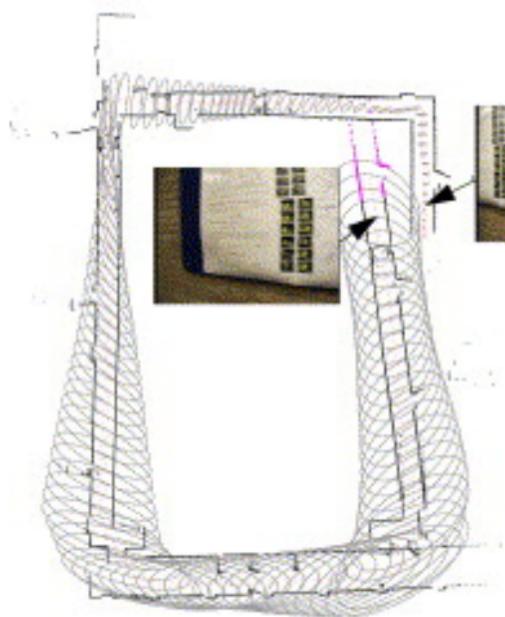
$$\begin{array}{lll} \mathbf{x} & \xrightarrow{\quad} & f_1(\mathbf{x}) = \hat{\mathbf{z}}_1 & z_1 \\ & \xrightarrow{\quad} & f_2(\mathbf{x}) = \hat{\mathbf{z}}_2 & z_2 \\ & \xrightarrow{\quad} & \dots & \\ & \xrightarrow{\quad} & f_n(\mathbf{x}) = \hat{\mathbf{z}}_n & z_n \end{array}$$

state  
(unknown)

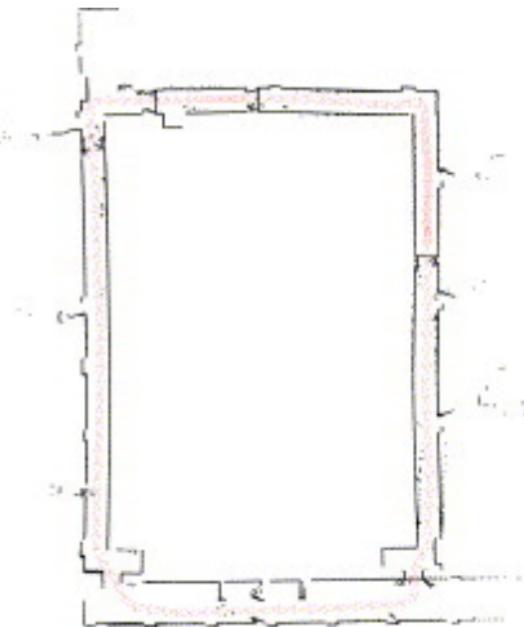
predicted  
measurements

real  
measurements

# Solving with least squares



(a)



(b)



# Least Squares Approach

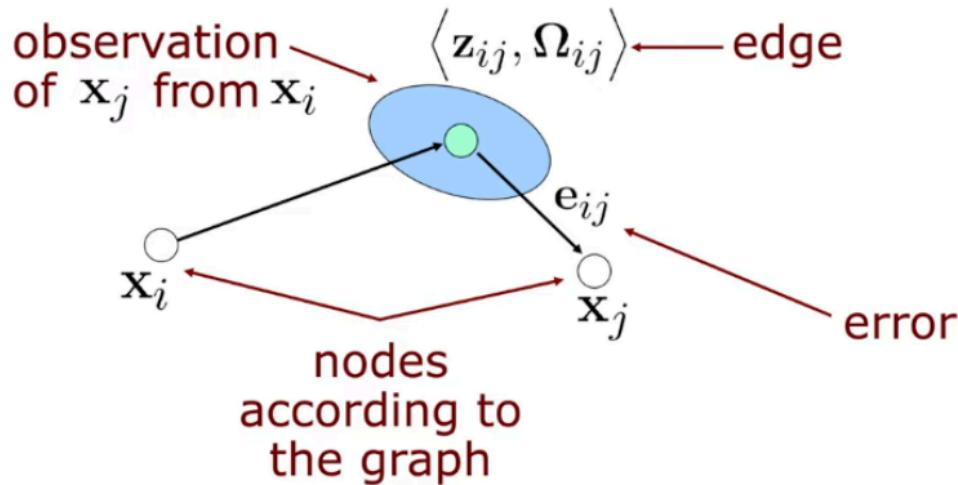
- Least squares error minimization

$$x^* = \underset{x}{\operatorname{argmin}} \sum_{ij} e_{ij}^T \Omega_{ij} e_{ij}$$

- Error function  $e_{ij}$  for an observation

$$e_{ij} = (x_j - x_i) - z_{ij}$$

# Pose Graph

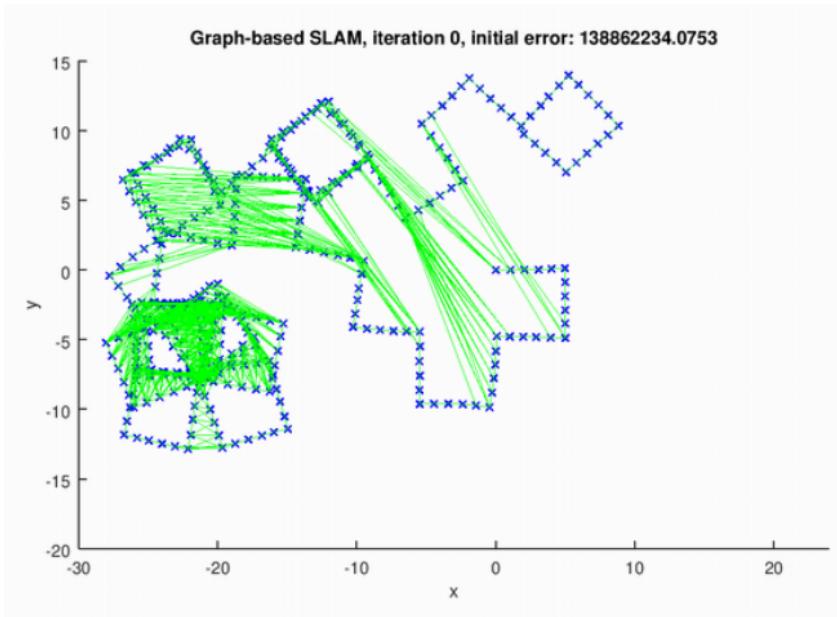


**Goal:**  $\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_{ij} \mathbf{e}_{ij}^\top \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij}$



# Minimization

- Open problem. You can use Gauss-Newton if you were doing this from scratch, but ideally, use a minimization library (slam\_toolbox uses Ceres).

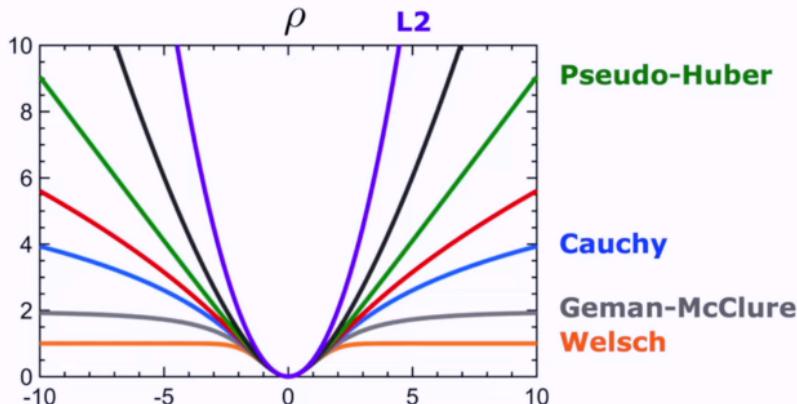


# Removing outliers



## M-Estimators    kernel function as weights for constraints

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i=1}^N \rho(e_i(\mathbf{x}))$$



## 2 Scan matching



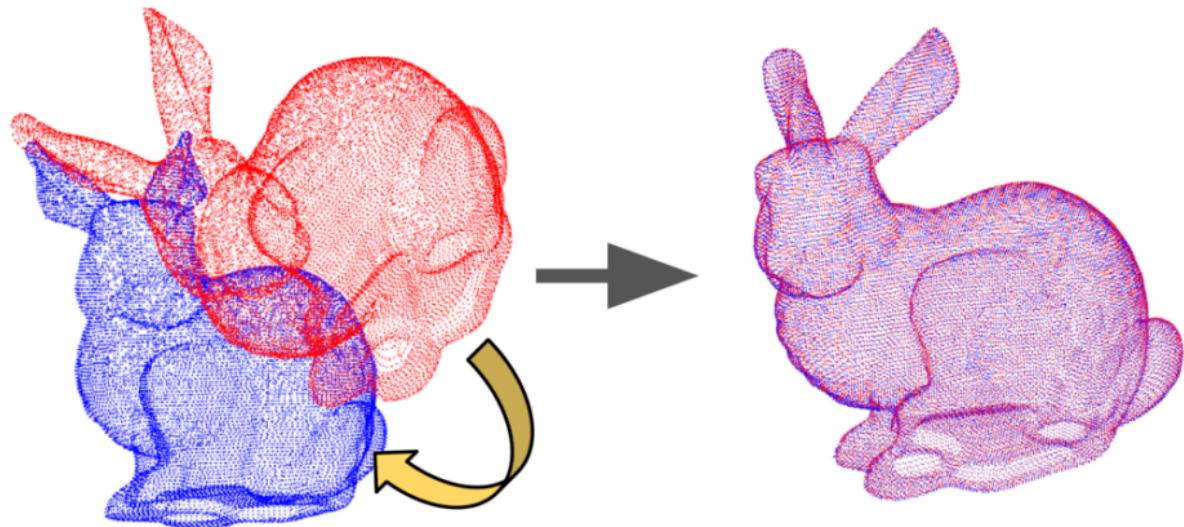
# What is scan matching

- Scan matching creates the lidar edges between  $x_i$  and  $x_j$  by finding lidar observations of the same object.
- There are many different ways to do it:
  - Iterative closest point (ICP)
  - Scan-to-scan
  - Scan-to-map
  - Map-to-map
  - Feature-based
  - RANSAC for outlier rejection
  - Correlative matching



# Iterative Closest Point

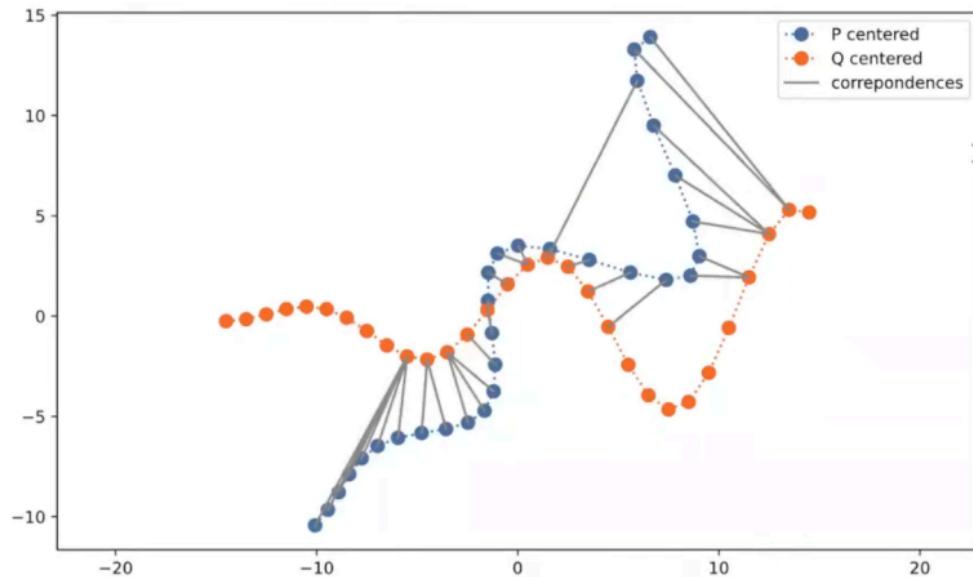
- ICP is a way to match two point clouds.





# Data Association

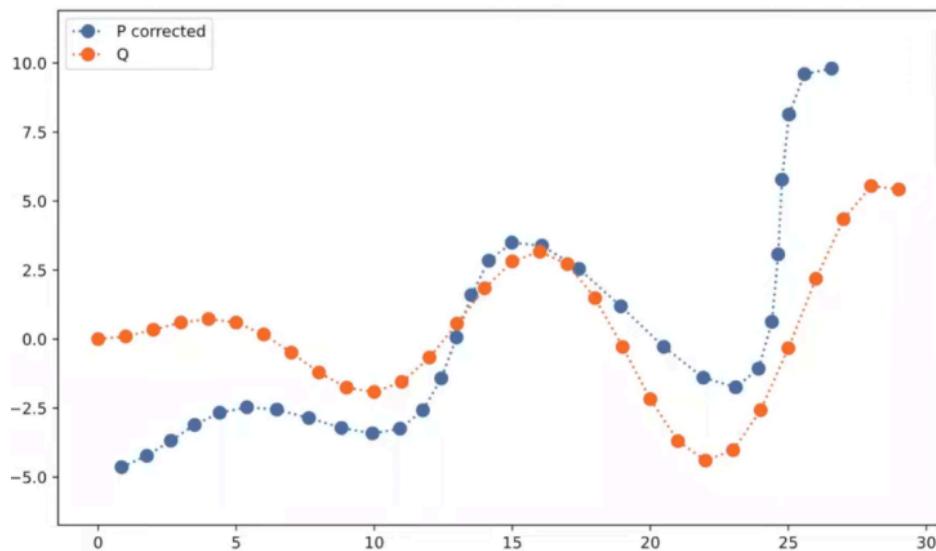
- For each point  $a_i$  on point cloud  $a$ , find the closest point  $b_j$ .





# Transformation

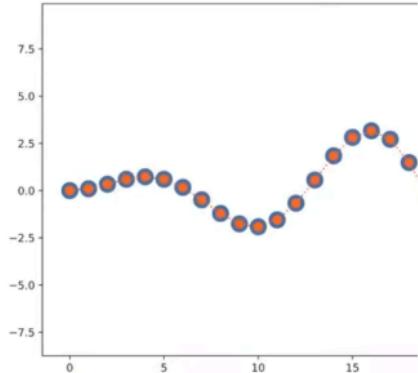
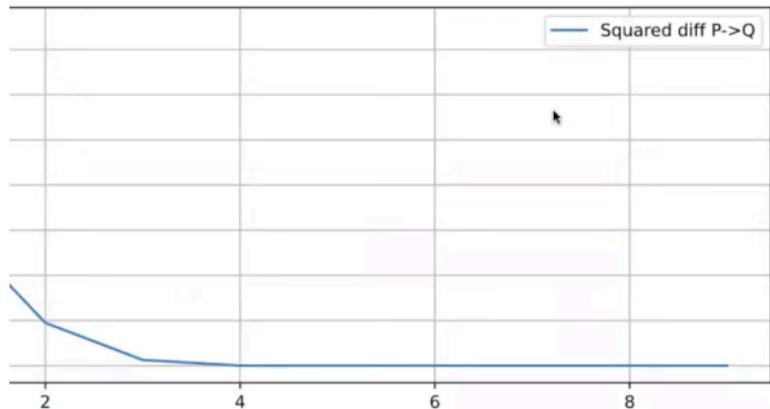
- Find a transformation to align the two point clouds
- First, align the center of mass of both point clouds, and then rotate them using SVD.



# Iterate



- Keep going until they converge!

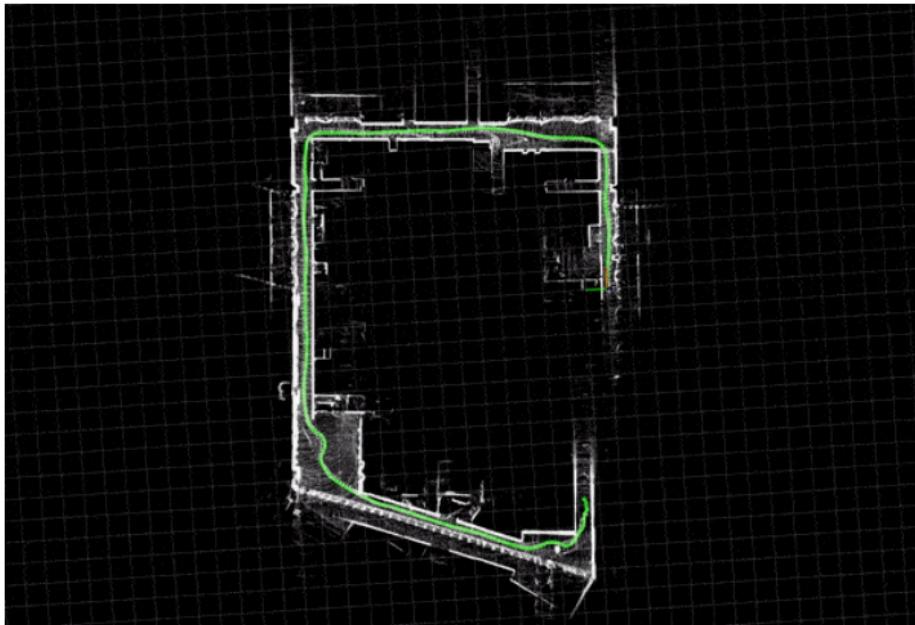


$$\boldsymbol{\xi}^* = \operatorname{argmin}_{\boldsymbol{\xi}} \sum_{i=1}^n [1 - M(\mathbf{S}_i(\boldsymbol{\xi}))]^2$$

# 3 All Together



# Loop Closure



# Slam Toolbox

