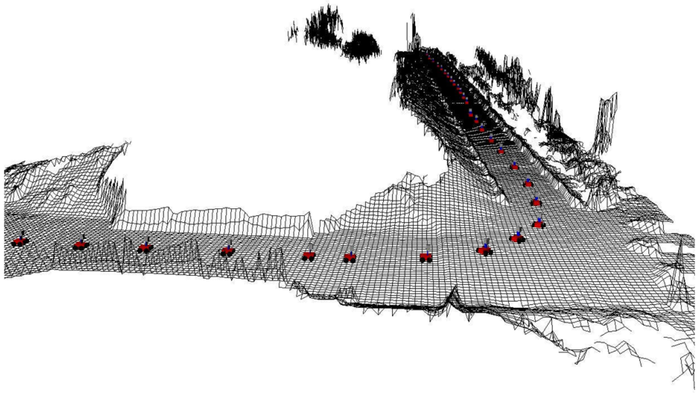
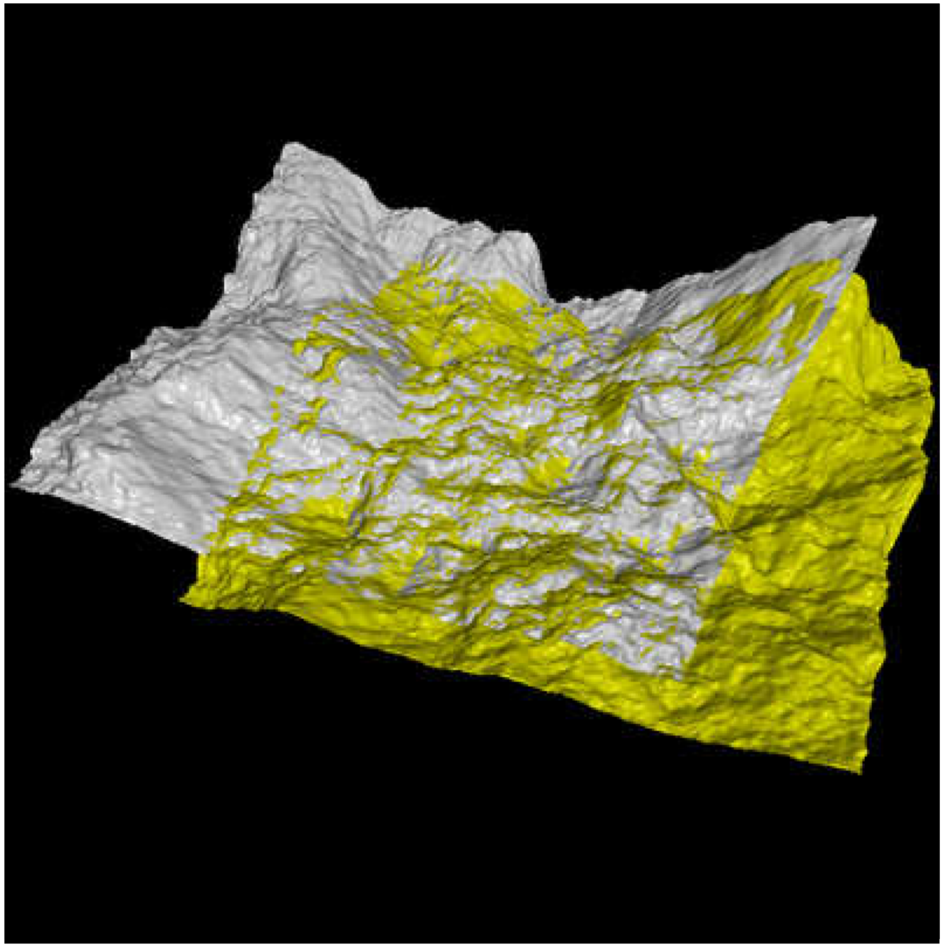
Iterative Closest Point Algorithm

Introduction to the idea behind ICP and implementation of a simple variant using Python programming language

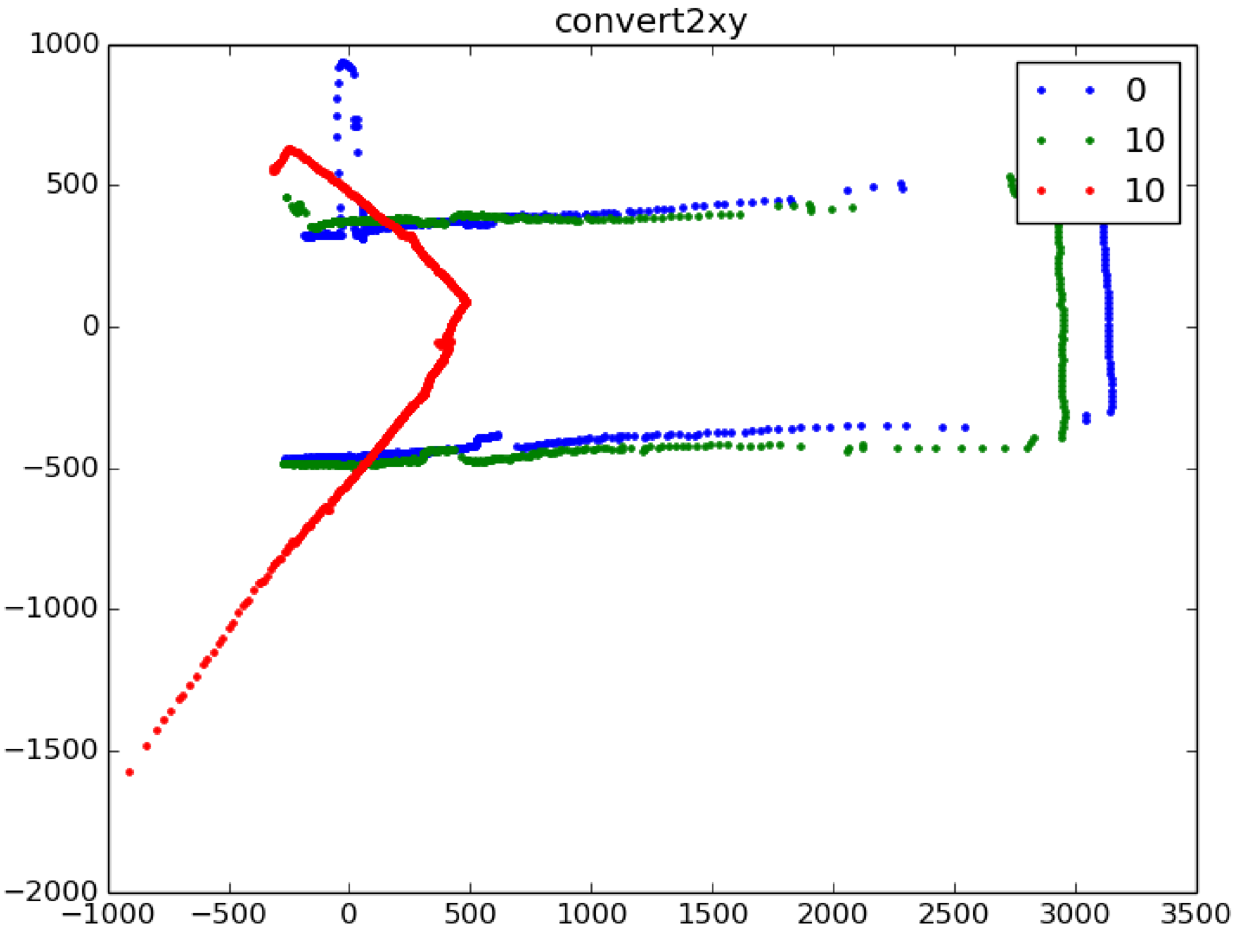
Motivation





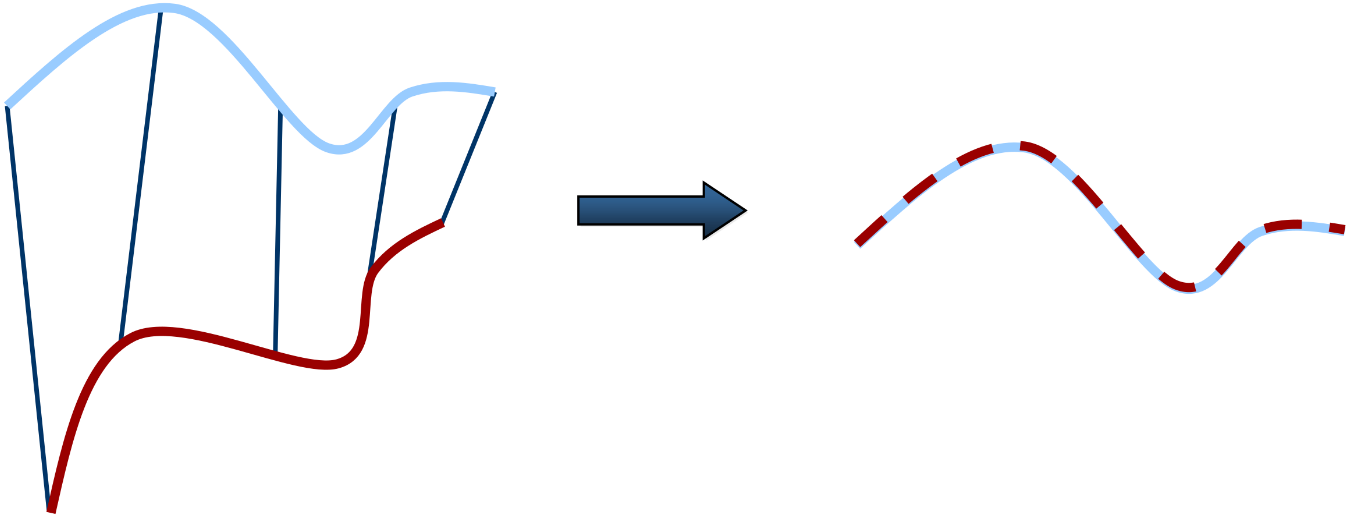
Images from “Introduction to Mobile Robotics. Iterative Closest Point Algorithm”. Wolfram Burgard, et al.

Motivation


convert2xy
1000
500
0
−500
−1000
−1500
−2000
−1000
−500
0
500
1000
1500
2000
2500
3000
3500
● ● 0
● ● 10
● ● 10

Key Idea

- **If the correct correspondences are known, the correct relative rotation/translation can be calculated in closed form.**



Images from “Introduction to Mobile Robotics. Iterative Closest Point Algorithm”. Wolfram Burgard, et al.

The problem

- **Given to corresponding point sets:**

  $P = \{p_i\};\ R = \{r_i\}$

- **Find translation t and rotation matrix R that minimize the following error:**

$$E(R,t) = \frac{1}{N_p} \sum_{i=1}^{N_p} \|p_i - R \cdot r_i - t\|^2$$

Finding roto-translation

- **First we subtract center of mass from each point:**

$$\mu_p = \frac{1}{N_p}\sum_{i=1}^{N_p} p_i \qquad \mu_r = \frac{1}{N_r}\sum_{i=1}^{N_r} r_i$$

$$P' = \{p_i - \mu_p\} = \{p_i'\}$$

$$R' = \{r_i - \mu_r\} = \{r_i'\}$$

Finding roto-translation

- **Let:** $W = \sum_{i=1}^{N_r} p_i \cdot r_i^T$
- **Using SVD we can represent *W* as:**

$$W = U \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} V^T$$

$$U, V \in \mathbb{R}^{3\times 3}$$

Finding roto-translation

- **If rank of *W* equals to *n, the optimal solution of E(R, t) is unique and is given by:***

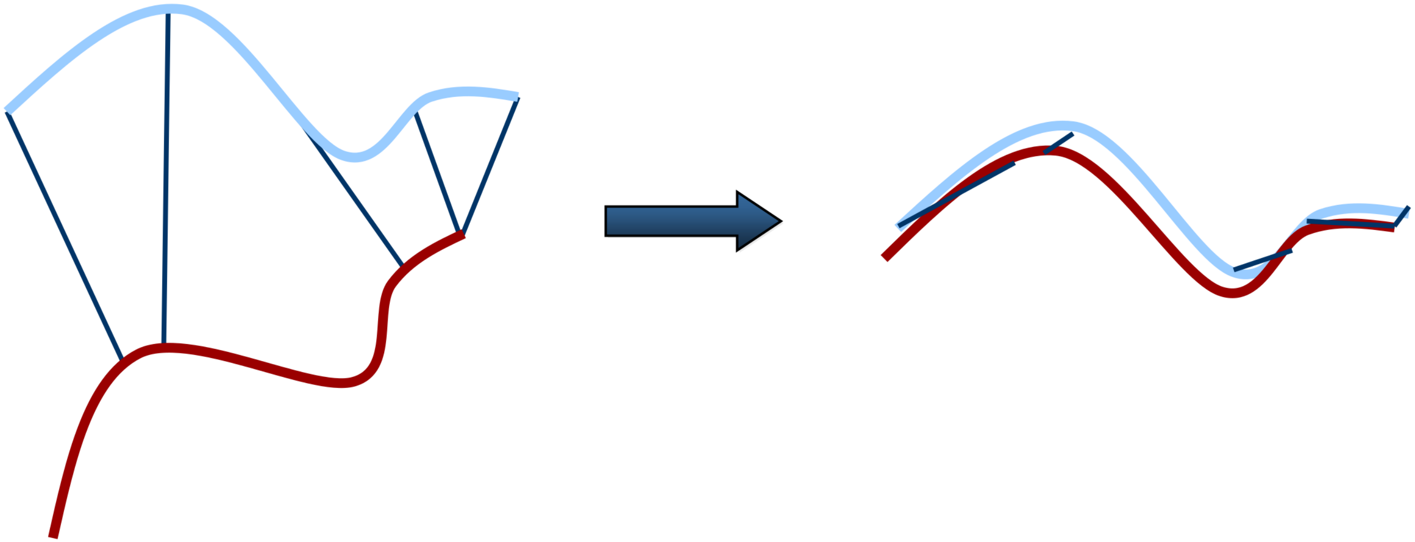
$$R = UV^T \qquad t = \mu_p - R \cdot \mu_p$$

- ***The minimal value of error function is:***

$$E(R,t) = \sum_{i=1}^{N_p} (\|p'_i\|^2 + \|r'_i\|^2) - 2(\sigma_1 + \sigma_2 + \sigma_3)$$

Unknown Correspondences

- **If correct correspondences are not known, it is generally impossible to determine the optimal relative rotation/translation in one step**



Images from “Introduction to Mobile Robotics. Iterative Closest Point Algorithm”. Wolfram Burgard, et al.

Iterative Closest Point

- **Introduced in [Besl & McKay 92]**
- **Algorithm:**
  - Find closest points from both sets and take them as correspondences
  - Find roto-translation for correspondences from the previous step and apply it
  - Check if error change is smaller than threshold, if not go to the step 1

ICP modifications

- Different metrics (point-to-plane, feature based).
- Usage of indexes (e.g. K-d trees)
- Dynamic caching.
- Point sampling and filtering.
- Random restarts.
- Alternative iteration procedures (e.g. AA-ICP).

ICP Summary

- ICP is a powerful algorithm for calculating the displacement between scans.
- The major problem is to determine the correct data associations.
- Given the correct data associations, the transformation can be computed efficiently using SVD.

Any questions?

Task

- Implement simple point-to-point ICP with unknown correspondences (do not forget about threshold)
- Apply k-d tree to speed-up correspondence lookup
- Modify the code so all points will not be duplicated across correspondences (i.e. each point can be part of only one correspondence)