

Generalized Iterative Closest Point

Mündliche Prüfung in der Vorlesung Autonome Roboter

bei Prof. Dr.-Ing. Michael Blaich

03.07.2024

Moritz Kaltenstadler, Fabian Klimpel, Johannes Brandenburger

Agenda

1. Einführung
2. Theorie
3. Demo: Eigene Implementierung in Python
4. Implementierung in ROS
5. Experiment
 1. Aufbau
 2. Durchführung
 3. Ergebnisse
 4. ...
6. Fazit

Theorie

- Einzige wirkliche Quelle: „Generalized-ICP“ von Segal, Haehnel & Thrun (2010)
- Algorithmus (ausführlicher als Original):

```
1  $T = T_0$ 
2  $CA = \text{computeCovarianceMatrices}(A)$ 
3 while has not converged do
4   for  $i = 1$  to  $N$  do
5      $m_i = \text{findClosestPointInA}(T * b_i)$ 
6      $c_i = \text{computeCovarianceMatrix}(T * b_i)$ 
7      $w_i = \text{computeWeightMatrix}(c_i, CA_i)$ 
8   end
9    $T = \text{optimizeLossFunction}(T, A, B, W)$ 
10 end
```

Theorie

- „point-to-point“ (Standard-ICP)
- „point-to-plane“
 - vergleicht Punkt mit Ebene durch Normalenvektor
- Generalized-ICP
 - quasi „plane-to-plane“
 - vergleicht die Kovarianzmatrizen der nächsten Punkte → probabilistisch
 - wenn in Ebene → Kovarianzmatrix ist „flach“

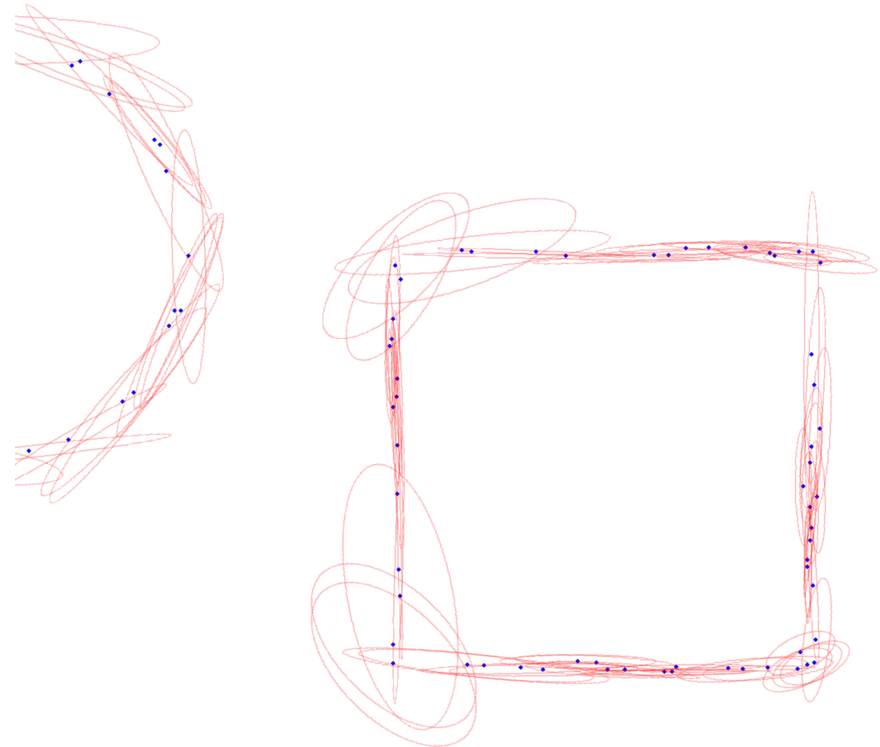


Abbildung 1: Kovarianzmatrizen (eigene Darstellung)

Demo: Eigene Implementierung in Python - Code

- Paper sehr mathematisch
- Algorithmus wurde nie komplett gezeigt
- zwar Implementierungen auf GitHub, aber nicht wirklich lesbar
- daher eigene Implementierung - vor allem für Verständnis
- eigene **2D-GICP-Funktion**
 - Input: Punktwolken A und B , ...
 - Output: Transformationsmatrix T , ...
- Version 1:
 - Visualisierung mit generierten Input-Wolken
 - iterativ durch die Steps klicken
- Version 2:
 - Simulation eines Roboters mit LiDAR-Sensor
 - Live-Berechnung der Transformation + Visualisierung

→ *LIVE DEMO*

Demo: Eigene Implementierung in Python - Ergebnis