Generalized Iterative Closest Point

Mündliche Prüfung in der Vorlesung Autonome Roboter

bei Prof. Dr.-Ing. Michael Blaich

03.07.2024

Johannes Brandenburger, Moritz Kaltenstadler, Fabian Klimpel

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

- „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“

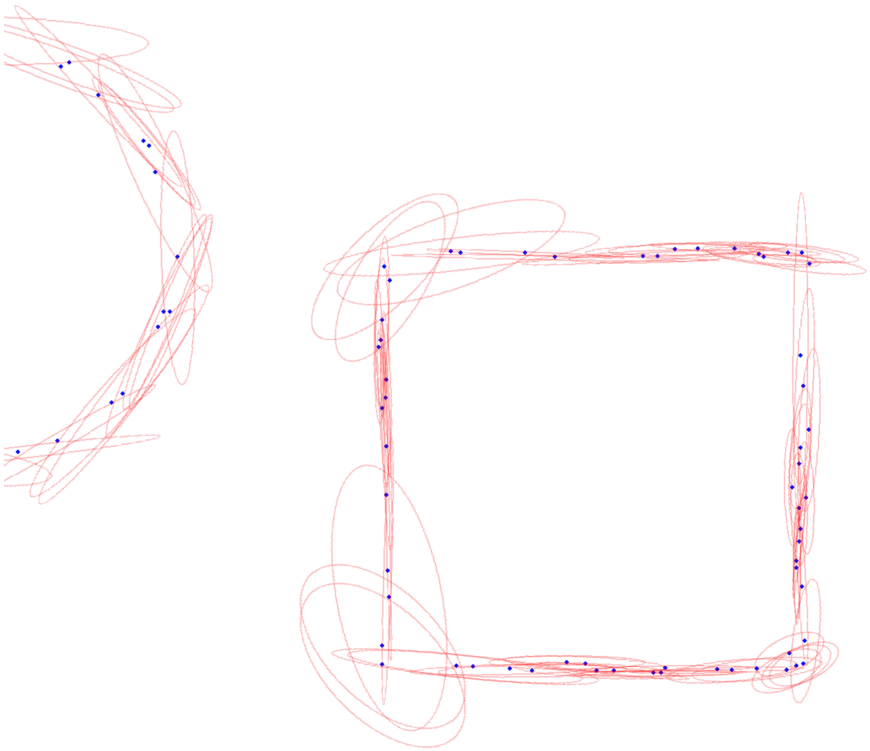

Abbildung 1: Kovarianzmatrizen (eigene Darstellung)

Theorie

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

$T = T_0$
$\mathrm{CA} = \text{computeCovarianceMatrices}(A)$
while has not converged **do**
  for $i = 1$ to N **do**
    $m_i = \text{findClosestPointInA}(T * b_i)$
    $c_i = \text{computeCovarianceMatrix}(T * b_i)$
    $w_i = \text{computeWeightMatrix}(c_i, \mathrm{CA}_i)$
  end
  $T = \text{optimizeLossFunction}(T, A, B, W)$
end

Demo: Eigene Implementierung in Python

- 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

$\rightarrow$ *LIVE DEMO*