

Generalized Iterative Closest Point

Mündliche Prüfung in der Vorlesung Autonome Roboter

bei Prof. Dr.-Ing. Michael Blaich

15.07.2024

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

- Einzige wirkliche Quelle: „Generalized-ICP“ von Segal, Haehnel & Thrun (2010)
 - Ziel: Iterative-Closest-Point-Algorithmus (ICP) verbessern
 - Standard-ICP & point-to-plane in **generelles Framework** überführen
 - **Probabilistische** Betrachtung
 - Nutzung **Oberflächenstruktur** aus beiden Scans (Kovarianzmatrizen) → **plane-to-plane**

Theorie - Mathematische Grundlagen

Kovarianzmatrix

- beschreibt die Streuung von Zufallsvariablen
- für Punkte in Punktwolken: Verteilung der Punkte in der Umgebung

Maximum Likelihood Estimation (MLE)

- Schätzverfahren für Parameter von Wahrscheinlichkeitsverteilungen
- der Parameter wird ausgewählt, der die beobachteten Daten am wahrscheinlichsten macht
- oft verwendet um: $\arg \max_p \dots / \arg \min_p \dots$ zu finden

Theorie - Standard-ICP

- **Iterative Closest Point** (ICP) ist ein Algorithmus, um die Transformation zwischen zwei Punktwolken zu schätzen
- vergleicht korrespondierende Punkte in beiden Wolken
- minimiert die quadratischen Abstände korrespondierender Punkte

```
1  $T \leftarrow T_0$ 
2 while not converged do
3   for  $i \leftarrow 1$  to  $N$  do
4      $m_i \leftarrow \text{FindClosestPointInA}(T \cdot b_i)$ 
5     if  $\|m_i - b_i\| \leq d_{\max}$  then
6        $w_i \leftarrow 1$ 
7     else
8        $w_i \leftarrow 0$ 
9     end
10  end
11   $T \leftarrow \arg \min_T \left\{ \sum_i w_i (\|T \cdot b_i - m_i\|)^2 \right\}$ 
12 end
```

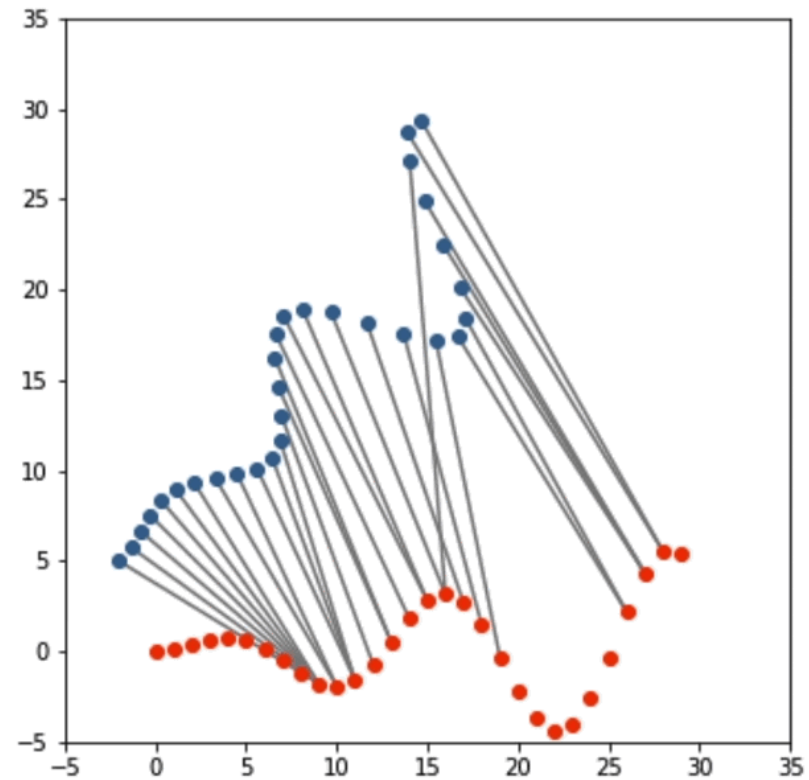


Abbildung 1: Standard-ICP (Igor Bogoslavskyi, 2021)

Theorie - Standard-ICP, point-to-plane, Generalized-ICP

- **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“

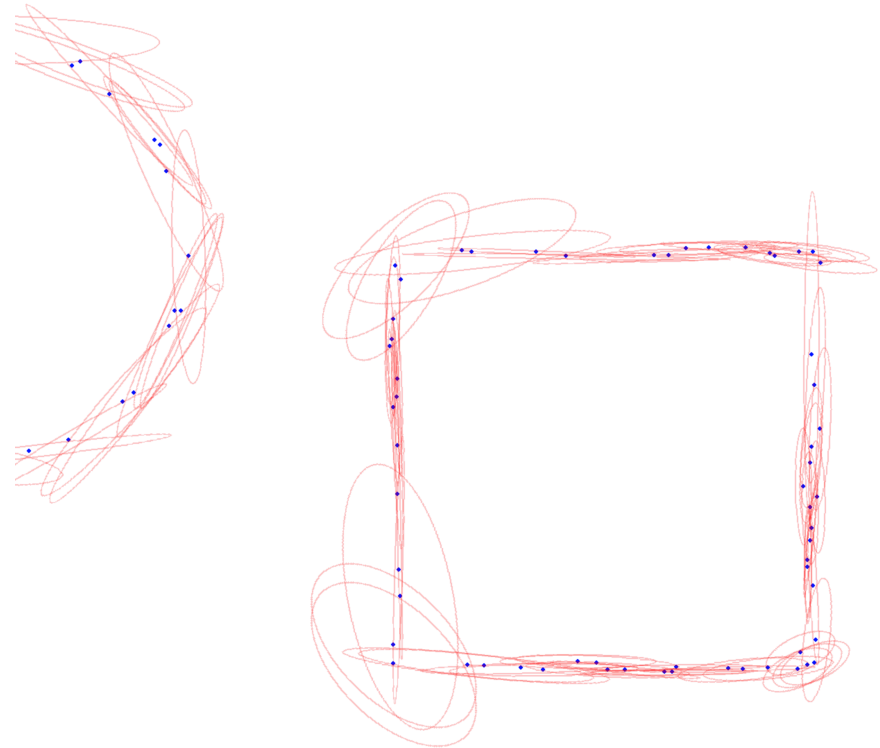


Abbildung 2: Kovarianzmatrizen (eigene Darstellung)

Theorie - GICP-Algorithmus

```
1  $T \leftarrow T_0$ 
2 while not converged do
3   for  $i \leftarrow 1$  to  $N$  do
4      $m_i \leftarrow \text{FindClosestPointInA}(T \cdot b_i)$ 
5      $d_i^{(T)} \leftarrow b_i - T \cdot m_i$            // Residuum / Abstand
6     if  $\| d_i^{(T)} \| \leq d_{\max}$  then
7        $C_i^A \leftarrow \text{computeCovarianceMatrix}(T \cdot b_i)$ 
8        $C_i^B \leftarrow \text{computeCovarianceMatrix}(m_i)$ 
9     else
10       $C_i^A \leftarrow 0$ ;  $C_i^B \leftarrow 0$ 
11    end
12  end
13   $T \leftarrow \arg \min_T \left\{ \sum_i d_i^{(T)^T} (C_i^B + T C_i^A T^T)^{-1} d_i^{(T)} \right\}$  // Maximum Likelihood Estimation
14 end
```

Theorie - GICP-Algorithmus

Variationen für Kovarianzmatrizen

$C_i^A \leftarrow \text{computeCovarianceMatrix}(T \cdot b_i)$

$C_i^B \leftarrow \text{computeCovarianceMatrix}(m_i)$

- für **Standard-ICP** (point-to-point):
 - $C_i^A \leftarrow 0$
 - $C_i^B \leftarrow 1 \rightarrow$ keine Oberflächenstruktur berücksichtigt
- für **point-to-plane**:
 - $C_i^A \leftarrow 0$
 - $C_i^B \leftarrow P_i^{-1} \rightarrow P_i$ ist die Projektionsmatrix auf die Ebene (beinhaltet Normalenvektor)
- für **plane-to-plane** (im Paper vorgeschlagene Methode):
 - `computeCovarianceMatrix` berechnet Kovarianzmatrix unter Betrachtung der nächsten 20 Punkte
 - verwendet **PCA** (Principal Component Analysis/Hauptkomponentenanalyse)

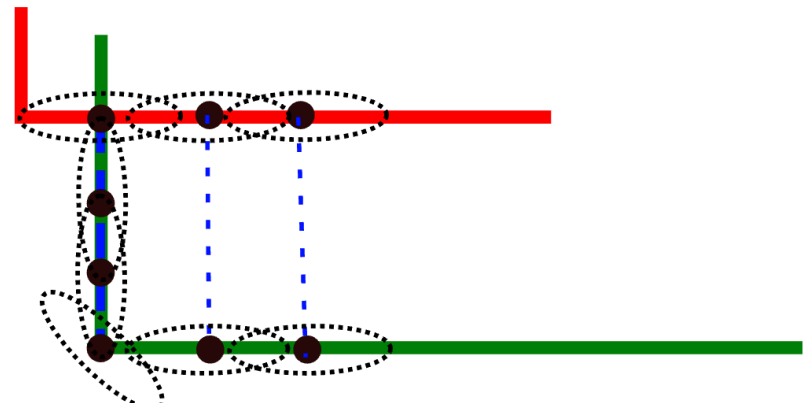


Abbildung 3: Plane-to-plane (Segal et al., 2009)

Theorie - GICP-Algorithmus

Paper Ergebnisse (Segal et al., 2009)

- GICP **genauer** bei simulierten und realen Daten
- immer noch relativ schnell und einfach
- Nutzen von Oberflächenstruktur **minimiert Einfluss von falschen Korrespondenzen**
- Parameter-Wahl für d_{\max} nicht mehr so kritisch → leichter einsetzbar in **unterschiedlichen Szenarien**

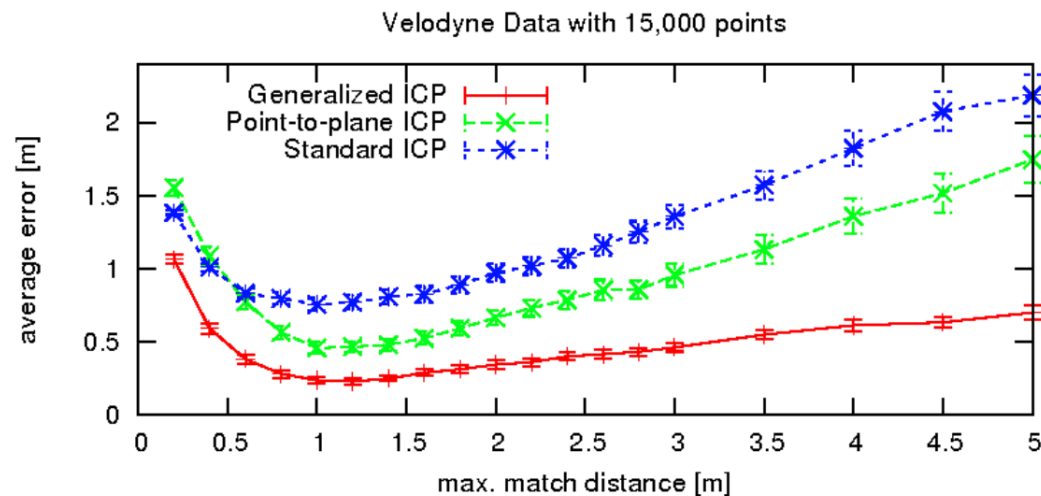


Abbildung 4: Durchschnittsfehler als Funktion von d_{\max} (Segal et al., 2009)

Demo: Eigene Implementierung in Python

- Paper sehr mathematisch
- 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*

→ *CODE OVERVIEW*

(Bild-)Quellen

Igor Bogoslavskyi. (2021). <https://nbviewer.org/github/niosus/notebooks/blob/master/icp.ipynb>

Segal, A. V., Hähnel, D., & Thrun, S. (2009). Generalized-ICP. *Robotics: Science and Systems*. <https://api.semanticscholar.org/CorpusID:231748613>