Generalized Iterative Closest Point

Mündliche Prüfung in der Vorlesung Autonome Roboter

bei Prof. Dr.-Ing. Michael Blaich

Hochschule Konstanz
Fakultät Informatik

16.07.2024

Johannes Brandenburger, Moritz Kaltenstadler, Fabian Klimpel

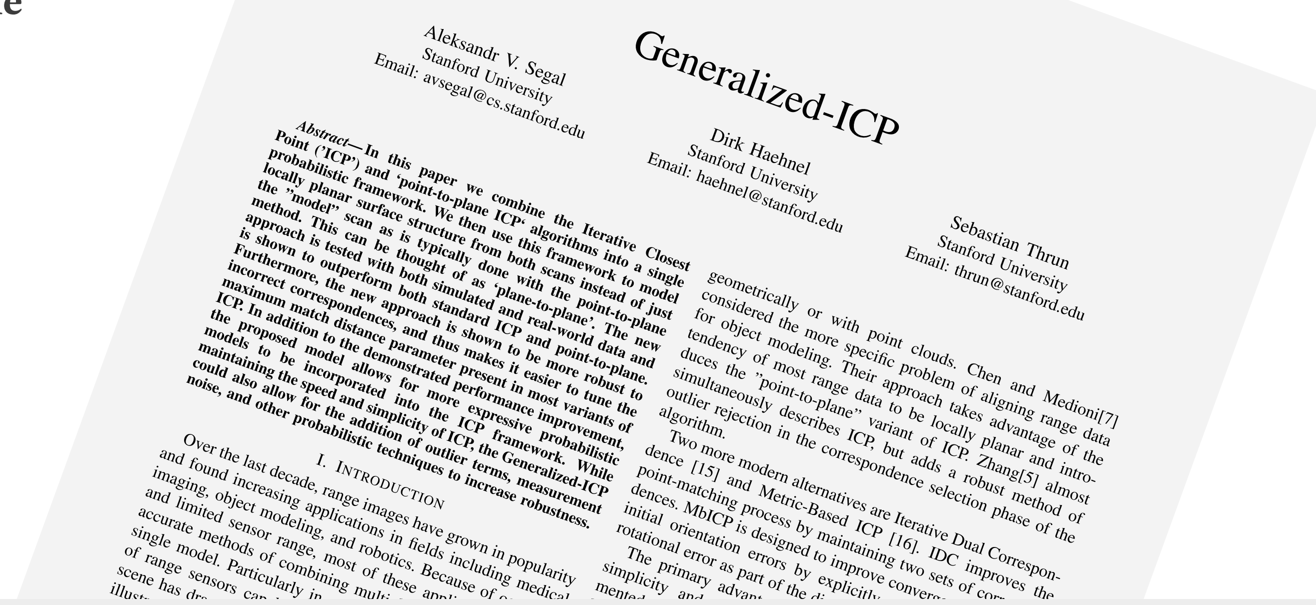
Agenda

1. Einführung
2. Theorie
   1. Mathematische Grundlagen
   2. Standard-ICP
   3. Point-to-Plane-ICP
   4. Generalized-ICP
   5. Ergebnisse von Segal et al.
3. Demo: Eigene Implementierung in Python
4. Implementierung in ROS - Versuch
   1. Versuchsaufbau
   2. Parameterisierung
   3. Implementierung ICP & GICP
   4. Problem
   5. Zeitmessung
   6. Maps
5. Auswertung
6. Fazit
7. (Bild-)Quellen

Einführung

- ICP: **Iterative Closest Point**
  - Scan-Matching-Algorithmus
  - Schätzung der Transformation zwischen zwei Punktwolken
  - Anwendung in der Lokalisierung mit z.B. LiDAR-Sensoren

- GICP: **Generalized-ICP**
  - veröffentlicht von Segal, Haehnel & Thrun (2009)
  - Stanford University
  - Ziel: ICP-Algorithmus verbessern und verallgemeinern
  - Standard-ICP & point-to-plane in **generelles Framework** überführen
  - **probabilistische** Betrachtung
  - Nutzung **Oberflächenstruktur** aus beiden Scans (Kovarianzmatrizen) ⟶ **plane-to-plane**


Generalized-ICP

Aleksandr V. Segal
Stanford University
Email: avsegal@cs.stanford.edu

Dirk Haehnel
Stanford University
Email: haehnel@stanford.edu

Sebastian Thrun
Stanford University
Email: thrun@stanford.edu

Abstract—In this paper we combine the Iterative Closest Point ('ICP') and 'point-to-plane ICP' algorithms into a single probabilistic framework. We then use this framework to model locally planar surface structure from both scans instead of just the "model" scan as is typically done with the point-to-plane method. This can be thought of as 'plane-to-plane'. The new approach is tested with both simulated and real-world data and is shown to outperform both standard ICP and point-to-plane. Furthermore, the new approach is shown to be more robust to incorrect correspondences, and thus makes it easier to tune the maximum match distance parameter present in most variants of ICP. In addition to the demonstrated performance improvement, the proposed model allows for more expressive probabilistic models to be incorporated into the ICP framework. While maintaining the speed and simplicity of ICP, the Generalized-ICP could also allow for the addition of outlier terms, measurement noise, and other probabilistic techniques to increase robustness.

I. Introduction

Over the last decade, range images have grown in popularity and found increasing applications in fields including medical imaging, object modeling, and robotics. Because of ...

geometrically or with point clouds. Chen and Medioni[7] considered the more specific problem of aligning range data for object modeling. Their approach takes advantage of the tendency of most range data to be locally planar and introduces the "point-to-plane" variant of ICP. Zhang[5] almost simultaneously describes ICP, but adds a robust method of outlier rejection in the correspondence selection phase of the algorithm.

Two more modern alternatives are Iterative Dual Correspondence [15] and Metric-Based ICP [16]. IDC improves the point-matching process by maintaining two sets of ...

Speaker Notes

- ICP Verfahren schon in Vorlesung
- grober Überblick
- ICP steht für Iterative Closest Point
- Scan-Matching-Algorithmus
- Schätzung der Transformation zwischen zwei Punktwolken
- Anwendung in der Lokalisierung mit z.B. LiDAR-Sensoren
- wurde 2009 von Segal, Haehnel & Thrun verbessert
- an Stanford University Verfahren entwickelt
- Generalized-ICP
- Ziel: ICP-Algorithmus verbessern und verallgemeinern
  - Probleme Standard-ICP:
    - nicht sehr robust
    - sehr empfindlich gegenüber der Parameterwahl
    - daher schlecht in mehreren Szenarien einsetzbar
  - Standard-ICP & point-to-plane in generelles Framework überführen
  - 2 große Änderungen:
    - Wahrscheinlichkeitstheorie
    - Nutzung planarer Strukturen
      - Punktwolke sind nicht random im Raum verteilt
      - haben Struktur zb von Wänden
      - GICP nutzt diese Struktur in Form von Kovarianzmatrizen
        - Überleitung Kovarianzmatrizen -> Mathematische Grundlagen

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: $\arg\max_p \ldots$ / $\arg\min_p \ldots$ zu finden

Speaker Notes

- Mathematische Grundlagen sind lediglich für ein gemeinsames Verständnis
- sodass auch Leute, die nicht die Vorlesung besucht haben, den Vortrag verstehen könnten
- Kovarianzmatrix beschreibt die Streuung von Zufallsvariablen
- in unserem Kontext: Verteilung von Punkte in der Umgebung
- wo unsere Punkte mit welcher Wahrscheinlichkeit liegen
- Maximum Likelihood Estimation brauchen wir für die Schätzung der Transformation bei ICP und GICP
- Schätzverfahren für Parameter von Wahrscheinlichkeitsverteilungen
- versucht quasi die Parameter zu finden, die eine gegebene Wahrscheinlichkeitsverteilung am besten beschreiben

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

$T \leftarrow T_0$
while not converged **do**
  for $i \leftarrow 1$ **to** N **do**
    $m_i \leftarrow \texttt{FindClosestPointInA}(T \cdot b_i)$
    if $\| m_i - b_i \| \leq d_{\max}$ **then**
      $w_i \leftarrow 1$
    else
      $w_i \leftarrow 0$
    end
  end
  $T \leftarrow \arg\min_T \left\{ \sum_i w_i (\| T \cdot b_i - m_i \|)^2 \right\}$
end

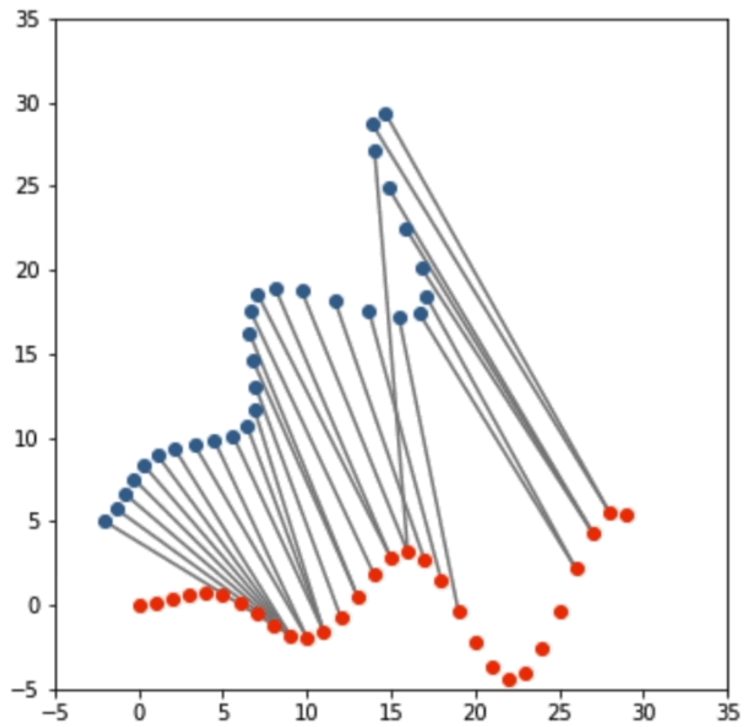


Abbildung 1: Standard-ICP (Igor Bogoslavskyi, 2021)

Speaker Notes

- wie schon gesagt: Standard ICP lässt uns Transformation zwischen zwei Punktwolken schätzen
- dabei ist er sehr einfach und schnell
- vergleicht korrespondierende Punkte in beiden Wolken
- minimiert die quadratischen Abstände korrespondierender Punkte
- schätzt so die Transformation
- in Pseudocode dargestellt
- Startwert für Transformation T_0
  - kann bereits eine sinnvolle Schätzung sein (z.b. Odometrie)
- Loop bis der Algorithmus konvergiert (daher auch „Iterative“)
- für jeden Punkt in der Quellwolke b_i wird der nächste Punkt in der Zielwolke A gesucht
- dann wird geschaut ob der Abstand kleiner als das Threshold $d_{\max}$ ist
  - Parameter steuert also, welche Punkte berücksichtigt werden und welche nicht
  - ist je nach Anwendungsszenario unterschiedlich
    - wenn Roboter schneller fährt, dann muss $d_{\max}$ größer sein
    - schwierig einzustellen
- Punkt wird gewichtet, oder nicht
- am Ende jedes Durchlaufs wird die Transformation berechnet
  - durch Veränderung der Transformationsparameter
  - so dass die quadratischen Abstände minimiert werden

Theorie - Point-to-Plane-ICP

- **Point to Plane ICP** ist eine Erweiterung des ICP Algorithmus
- vergleicht korrespondierende Punkte in einer Wolke zu Ebenen in der anderen
- Ebenen wird durch Punkt und Normalenvektor definiert

$$T \leftarrow \arg\min_T \left\{ \sum_i \left((T \cdot b_i - m_i) \cdot \boldsymbol{n_i} \right)^2 \right\}$$

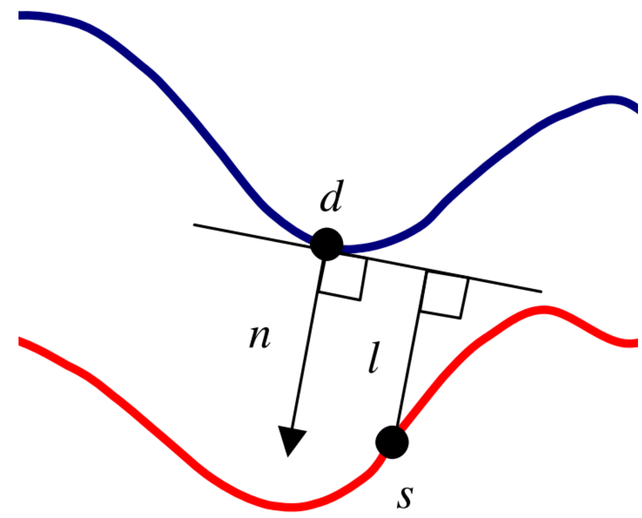


Abbildung 2: Point-to-Plane-ICP (Kok-Lim Low, 2004)

Speaker Notes

- Point-to-Plane-ICP ist eine Erweiterung des Standard-ICP
- Standard-ICP betrachtet keine Oberflächenstruktur
  - bei Laserscan zum Beispiel:
    - wenn 2 mal eine Wand gescant
    - Punkte zwar durch die Abtastrate unterschiedlichen Stellen im Raum
    - kann sein, dass die Bewegung zur Wand nicht wirklich groß war
- um dies entgegenzuwirken, vergleicht Point-to-Plane-ICP Punkte in einer Wolke zu Ebenen in der anderen
- Ebenen wird durch Punkt und Normalenvektor definiert
- Optimierungsfunktion in der letzten Zeile des Algorithmus:
  - minimiert quadratische Abstände zwischen Punkt und Ebene

Theorie - Standard-ICP, Point-to-Plane, Generalized-ICP

- **Point-to-Point**
  - Standard-ICP
  - vergleicht Punkt mit Punkt
- **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“

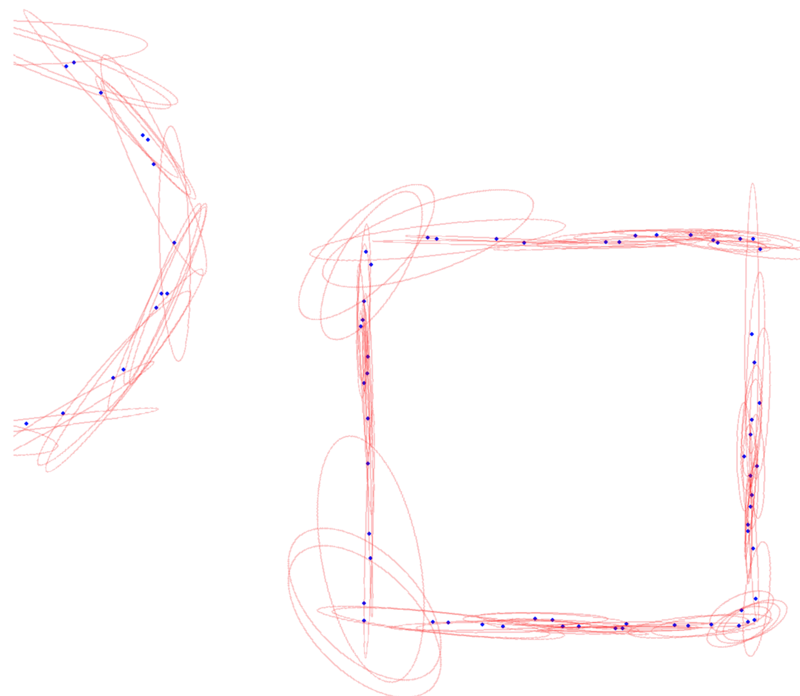


Abbildung 3: Kovarianzmatrizen (eigene Darstellung)

Speaker Notes

- hier nochmal ein Überblick über die verschiedenen ICP-Verfahren
- Point-to-Point: Standard-ICP
  - vergleicht Punkt mit Punkt
  - einfach und schnell
  - aber nicht robust sehr empfindlich gegenüber Parameterwahl
- Point-to-Plane:
  - vergleicht Punkt mit Ebene durch Normalenvektor
  - besser als Standard-ICP
  - nutzt Oberflächenstruktur einer Punktwolke
  - aber eben nur von einer
- Generalized-ICP:
  - quasi „Plane-to-Plane“
  - vergleicht die Kovarianzmatrizen der nächsten Punkte ⟶ probabilistisch
  - wenn in Ebene ⟶ Kovarianzmatrix ist „flach“
    - sieht man in Bild rechts
  - nutzt Oberflächenstruktur beider Punktwolken
  - welche Ergebnisse dies hat, dazu später mehr

Theorie - GICP-Algorithmus

$T \leftarrow T_0$
while not converged **do**
  for $i \leftarrow 1$ **to** N **do**
    $m_i \leftarrow$ `FindClosestPointInA`$(T \cdot b_i)$
    $d_i^{(T)} \leftarrow b_i - T \cdot m_i$ // Residuum / Abstand
    if $\| d_i^{(T)} \| \leq d_{\max}$ **then**
      $C_i^A \leftarrow$ `computeCovarianceMatrix`$(T \cdot b_i)$
      $C_i^B \leftarrow$ `computeCovarianceMatrix`(m_i)
    else
      $C_i^A \leftarrow 0; \quad C_i^B \leftarrow 0$
    end
  end
  $T \leftarrow \arg\min_T \left\{ \sum_i {d_i^{(T)}}^T \left(C_i^B + T C_i^A T^T \right)^{-1} d_i^{(T)} \right\}$ // Maximum Likelihood Estimation
end

Speaker Notes

- im Paper wurde Algorithmus nie zusammenhängend dargestellt
  - deshalb hier nochmals selber zusammengebaut
- Anfang des Algorithmus gleich wie bei Standard-ICP
- statt Gewichtung mit 0 oder 1 werden Kovarianzmatrizen verwendet
  - wie genau diese berechnet bzw gewählt werden, dazu mehr auf der nächsten Folie
- Minimierungsfunktion am Ende des Schleifendurchlaufs anders
  - nutzt Gewichtungsmatrix, die aus den Kovarianzmatrizen berechnet wird
    - mittlere Teil der Minimierungsfunktion
- im Paper wird für die Optimierung der Transformation also für arg min Maximum Likelihood Estimation verwendet
  - wählt Transformationsmatrix T so dass die Verteilung am wahrscheinlichsten ist

```
T ← T_0
while not converged do
  for i ← 1 to N do
    m_i ← FindClosestPointInA(T · b_i)
    d_i^(T) ← b_i − T · m_i                     // Residuum / Abstand
    if ‖ d_i^(T) ‖ ≤ d_max then
      C_i^A ← computeCovarianceMatrix(T · b_i)
      C_i^B ← computeCovarianceMatrix(m_i)
    else
      C_i^A ← 0;   C_i^B ← 0
    end
  end
  T ← argmin_T { Σ_i d_i^(T)^T (C_i^B + T C_i^A T^T)^(−1) d_i^(T) }   // Maximum Likelihood Estimation
end
```

Theorie - GICP-Algorithmus

Variationen für Kovarianzmatrizen

$C_i^A \leftarrow \texttt{computeCovarianceMatrix}(T \cdot b_i)$
$C_i^B \leftarrow \texttt{computeCovarianceMatrix}(m_i)$

- für **Standard-ICP** (Point-to-Point):
  - $C_i^A \leftarrow 0$
  - $C_i^B \leftarrow 1$ $\longrightarrow$ keine Oberflächenstruktur berücksichtigt (einfache Gewichtung)
- für **Point-to-Plane**:
  - $C_i^A \leftarrow 0$
  - $C_i^B \leftarrow P_i^{-1}$ $\longrightarrow$ 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)

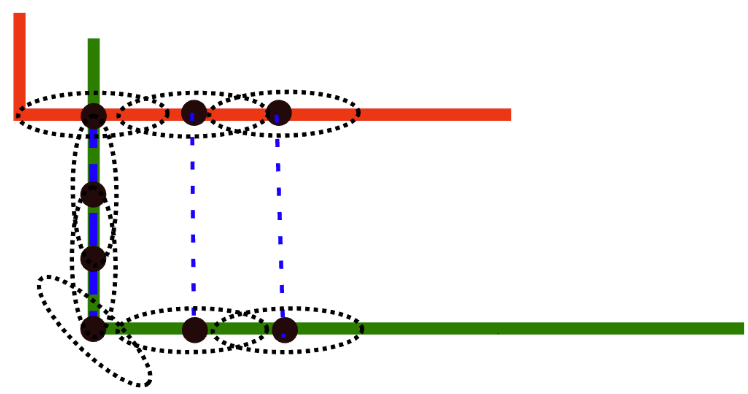


Abbildung 4: Plane-to-Plane (Segal et al., 2009)

Speaker Notes

- Wahl der Kovarianzmatrizen sind also entscheidend für den GICP-Algorithmus
- auch der Grund, warum „Generalized“ ICP
  - denn es lässt sich auch Standard-ICP und Point-to-Plane-ICP dadurch abdecken
- für Standard-ICP werden Kovarianzmatrizen einfach auf 0 bzw 1 gesetzt
  - dadurch werden die Punkte einfach gewichtet und es wird keine Oberflächenstruktur berücksichtigt
- für Point-to-Plane-ICP werden die Source-Kovarianzmatrizen auf Projektionsmatrizen gesetzt
  - diese beinhalten den Normalenvektor der Ebene
  - Oberflächenstruktur der einen Wolke berücksichtigt
- aber richtig gut erst bei dem im Paper vorgeschlagenen Verfahren
  - quasi „Plane to Plane“
- hier werden wirklich Kovarianzmatrizen ausgerechnet
  - 20 umliegende Punkte werden betrachtet
  - Verteilung mit Hauptkomponentenanalyse bestimmt
- allerdings auch etwas mehr Rechenaufwand bei jeder Iteration
- Berechnen Kovarianzmatrizen geschiet bei beiden Wolken -> Berücksichtigung beider Oberflächenstrukturen
- Bild rechts zeigt Kovarianzmatrizen für paar Punkte
  - man sieht:
    - Ausrichtung/Wölbung stimmt mit Ebene überein
    - Kovarianzmatrix ist „flach“

Theorie - GICP-Algorithmus

Ergebnisse von Segal et al.

- 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**

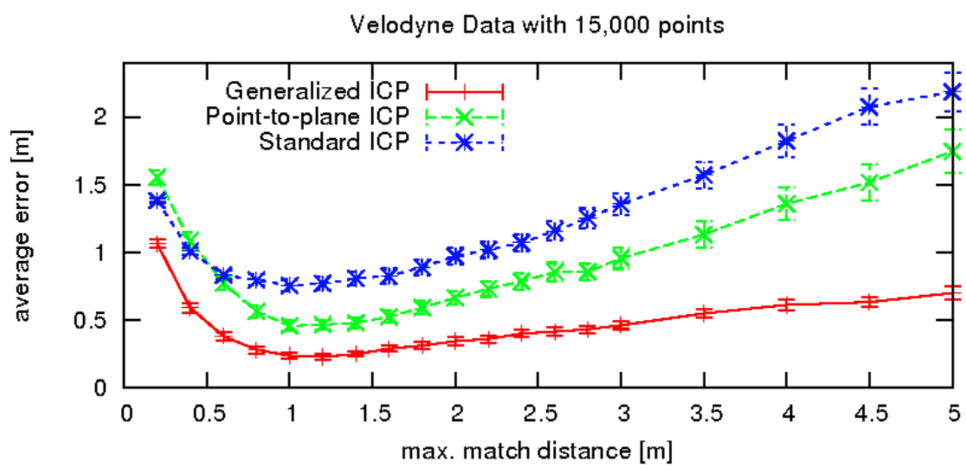


Abbildung 5: Durchschnittsfehler als Funktion von $d_{\max}$ (Segal et al., 2009)

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*

Implementierung in ROS - Versuch

Versuchsaufbau

- Vorbedingungen:
  - Bag File
  - Skript für Nodes
  - Yaml files als configuration

- Durchführung:
  - fünf Durchgänge mit Standardparameterisierung (ICP & GICP)
  - drei unterschiedliche Maps
  - unterschiedliche Parameterisierung

- Auswertung:
  - Bag Files mit Topics
  - Python-Skript zur Auswertung
  - Bokeh für Visualisierung

Implementierung in ROS - Versuch

Parameterisierung

```
//name des odom topics
this->declare_parameter("odom_topic", "");
//name des icp topics
this->declare_parameter("gicp_result_topic", "");
//name des zeitmessung topics
this->declare_parameter("alignment_time_topic", "");
//parameter ob gicp oder icp verwendet wird
this->declare_parameter("gicp", false);
//ob manuelle transformation gepublished wird
this->declare_parameter("publish_tf", false);

//icp parameter
this->declare_parameter("max_correspondence_distance", 0.0);
this->declare_parameter("maximum_iterations", 0);
this->declare_parameter("transformation_epsilon", 0.0);
this->declare_parameter("euclidean_fitness_epsilon", 0.0);
```

Implementierung in ROS - Versuch

Parameterisierung über YAML file

```yaml
gicp_lio:
  ros__parameters:
    gicp: True
    publish_tf: False
    alignment_time_topic:
"galignment_time"
    odom_topic: "glidar_odom"
    gicp_result_topic: "glidar_odom_eval"
    max_correspondence_distance: 0.2
    maximum_iterations: 100
    transformation_epsilon: 0.000000001
    euclidean_fitness_epsilon: 0.00001
```

```yaml
gicp_lio:
  ros__parameters:
    gicp: False
    publish_tf: False
    alignment_time_topic:
"alignment_time"
    odom_topic: "lidar_odom"
    gicp_result_topic: "lidar_odom_eval"
    max_correspondence_distance: 0.2
    maximum_iterations: 100
    transformation_epsilon: 0.000000001
    euclidean_fitness_epsilon: 0.00001
```

Implementierung in ROS - Versuch

Implementierung ICP & GICP

- ICP:

```
pcl::IterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> icp;
icp.setInputSource(src);
icp.setInputTarget(tgt);
pcl::PointCloud<pcl::PointXYZ>::Ptr output(new
pcl::PointCloud<pcl::PointXYZ>);
icp.align(*output);
```

- GICP:

```
pcl::GeneralizedIterativeClosestPoint<pcl::PointXYZ, pcl::PointXYZ> gicp;
gicp.setInputSource(src);
gicp.setInputTarget(tgt);
pcl::PointCloud<pcl::PointXYZ>::Ptr output(new
pcl::PointCloud<pcl::PointXYZ>);
gicp.align(*output);
```

Implementierung in ROS

Problem

```
// reduce tick speed in topic_callback
tick++;
if (tick % 3 != 0) {
  return;
}
```

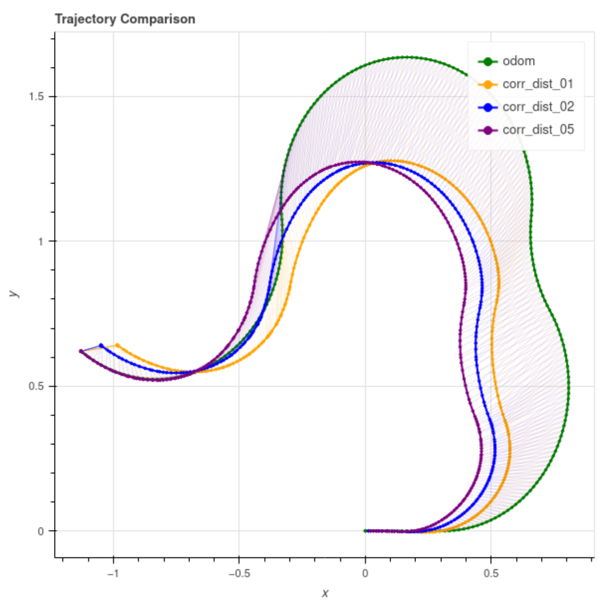


Abbildung 6: Trajectory plot with higher tick speed

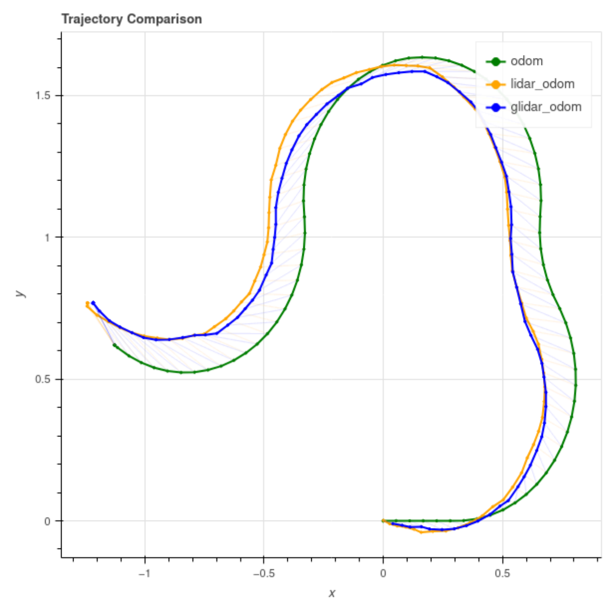


Abbildung 7: Trajectory plot with lower tick speed

Implementierung in ROS - Versuch

Zeitmessung

```
auto start = std::chrono::high_resolution_clock::now();
icp.align(*output);
transformation = icp.getFinalTransformation();
auto finish = std::chrono::high_resolution_clock::now();

std::chrono::duration<double> elapsed = finish - start;
std_msgs::msg::Float64 time_msg;
time_msg.data = elapsed.count();
time_publisher_->publish(time_msg);
```

Implementierung in ROS - Versuch

Turtlebot3 World

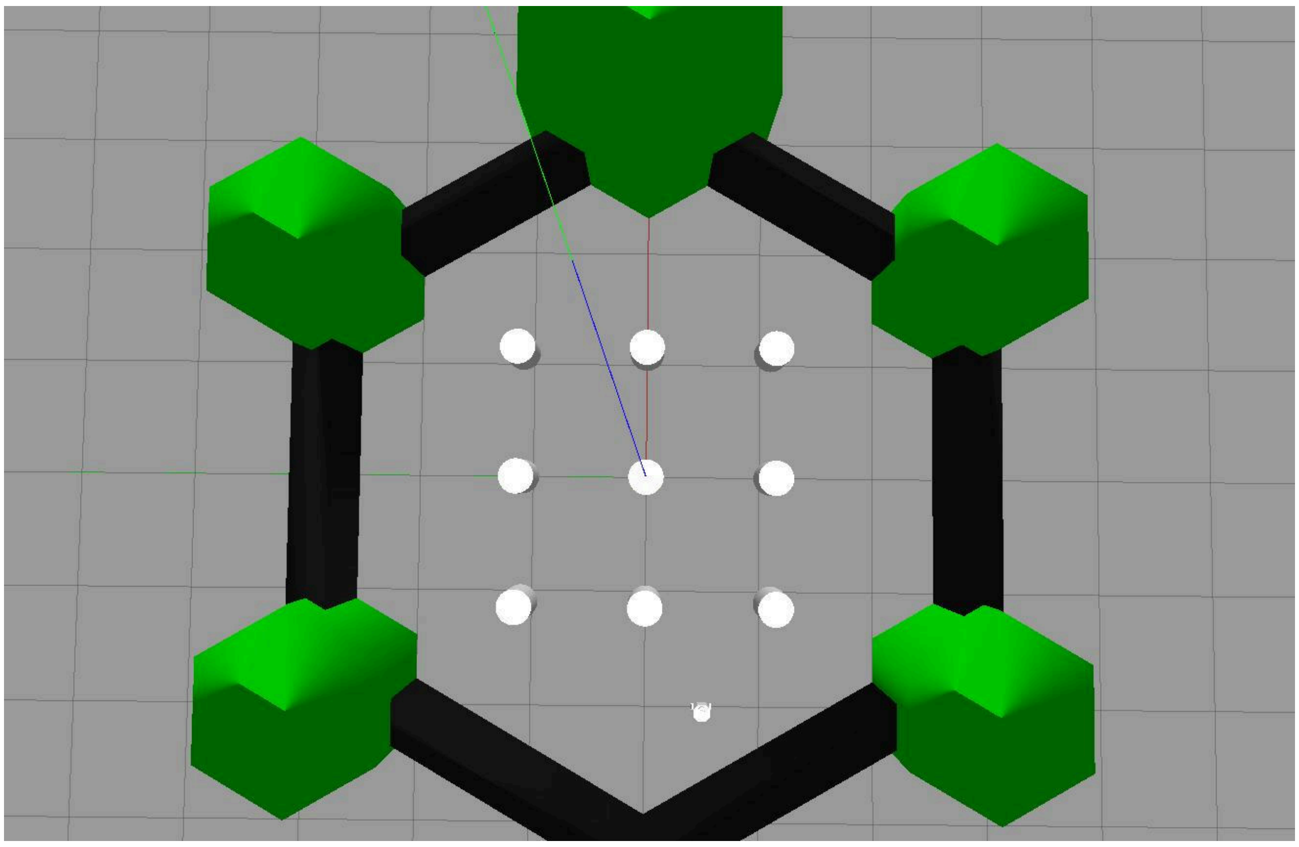


Abbildung 8: Screenshot Gazebo

Implementierung in ROS - Versuch

Turtlebot3 ICP World

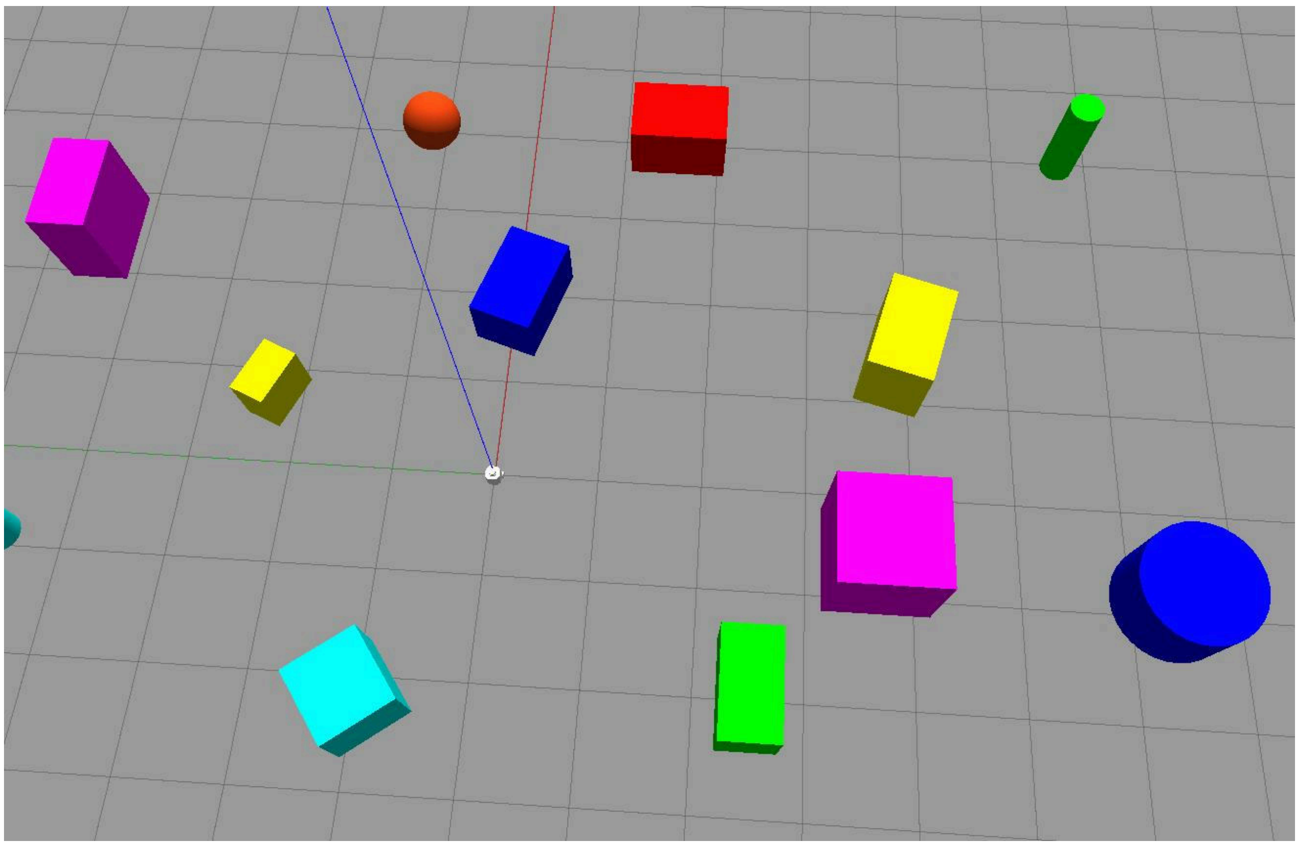


Abbildung 9: Screenshot Gazebo

Implementierung in ROS - Versuch

Turtlebot3 World

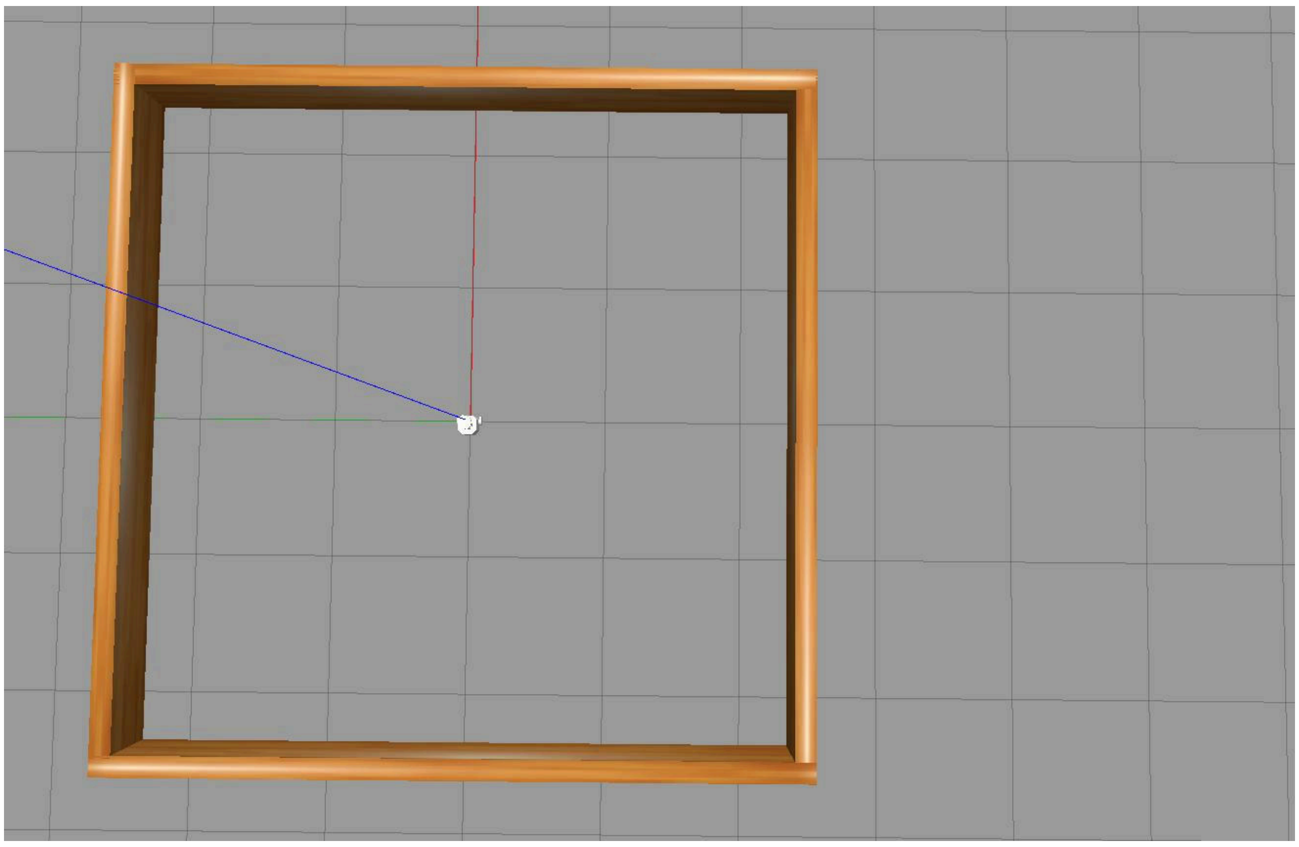


Abbildung 10: Screenshot Gazebo

Auswertung

- Link zu Bokeh Files

Fazit

(Bild-)Quellen

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

Kok-Lim Low. (2004). *Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration*. https://www.comp.nus.edu.sg/~lowkl/publications/lowk_point-to-plane_icp_techrep.pdf

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