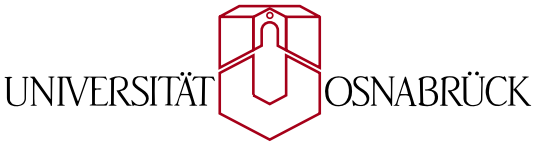
Ausgabe am **26.04.2021**
Abgabe bis zum **09.05.2021 23:59 Uhr**



Priv. Doz. Dr. Thomas Wiemann, Alexander Mock

Robotik
Übungsblatt 2

Sommersemester 2021

Aufgabe 2.1 (Filterung von Laserscans) **(4P)**

(a) Schreiben Sie eine Node, die sich zu einem Laserscan-`Topic` verbindet, die ankommenden Daten mit einem Medianfilter glättet und die neuen Werte wieder in das ROS-System publisht. Erstellen Sie dazu zunächst im Paket `blatt02_gruppeX` eine Node namens `median_node`. Die Anzahl der zur Berechnung herangezogenen Werte sollte über einen privaten ROS-Parameter `filter_size` variabel einstellbar sein. Informationen über ROS-Parameter finden Sie unter `http://wiki.ros.org/Parameter%20Server`. Wie man solche ROS-Parameter dann in Ihrem C++-Code benutzt, wird hier beschrieben: `http://wiki.ros.org/roscpp/Overview/Parameter%20Server`. Sind die Parameter implementiert, kann man diese beim Start der Node anpassen, zum Beispiel:

```
$ rosrun blatt02 median_node _filter_size:=20
```

Hierbei signalisiert der _ vor dem Parameter `filter_size` der Node, dass es sich um einen privaten Parameter nur für sie selbst handelt. Beachten Sie die Verwendung von ":=" anstelle von "="! Probieren Sie verschiedene Filtergrößen aus. Was fällt Ihnen auf?

Im StudIP finden Sie eine `laser.bag` mit Aufnahmen eines statischen `kurt`-Roboters mit der Sie Ihre Node testen können. Visualisieren Sie den Originalscan sowie Ihren gefilterten Scan in `RViz` und variieren Sie die Filterbreite. (2P)

Hinweis: Als Fixed-Frame der `laser.bag` verwenden Sie `/laser`.

(b) Berechnen Sie das Ergebnis der Reduktionsfilterung mit $\Delta = 5$ auf die folgende Messreihe schriftlich.

4 4 3 2 2 9 10 8 5 8 9 42 2 2 5 7 40 6 3 3

(1P)

(c) Vergleichen Sie den Reduktionsfilter mit dem Medianfilter. Welche qualitativen Unterschiede bestehen zwischen diesen beiden? (1P)

Aufgabe 2.2 (Konvertierung von Laserscans) **(6P)**

In der Praxis werden statt der Rohdaten des Laserscans meist Punktwolken zur Repräsentation der Daten eingesetzt. In dieser Aufgabe lernen Sie, wie Sie einen oder mehrere Laserscans zu einer solchen Punktwolke konvertieren können.

(a) Konvertieren Sie zunächst einen Laserscan in eine Punktwolke. Erstellen Sie dazu in Ihrem ROS-Paket eine neue Node mit dem Namen `scan_to_cloud` in der Sie einen Laserscan in eine solche Punktwolke umwandeln. Die Formel zur Umrechnung von Laser-Abstandsmessungen zu kartesischen Koordinaten finden Sie in den Vorlesungsfolien.

Informieren Sie sich dazu mit Hilfe des `rosmsg` Befehls und des Internets über die Definitionen der jeweiligen Messages, die in dieser Konvertierung verwendet werden:

- `LaserScan` (Paket: `sensor_msgs`)
- `PointCloud2` (Paket: `sensor_msgs`)

Die Winkel eines `LaserScan`s können Sie mit Hilfe der Felder `angle_min` und `angle_increment` ausrechnen. Eine mögliche Fehlerquelle ist in den Kommentaren einer der drei Message-Definitionen versteckt. Achten Sie deshalb auf eine geeignete Fehlerbehandlung. Visualisieren Sie anschließend die resultierende Punktwolke in `RViz`. (3P)

(b) Schreiben Sie nun Ihre erste Launch-File. Platzieren Sie dazu in den Ordner `blatt02_gruppeX/launch` eine Datei namens `scan_merger.launch`. In einer Launch-File kann man definieren, welche ROS-Nodes mit welchen Parametern gebündelt gestartet werden sollen. Eine solche Beschreibung geschieht über XML. Füllen Sie Ihre Launch-File mit Inhalt. Fügen Sie dazu Ihre Node zweimal zu der Launch-File hinzu. Dabei soll jede Ihrer Node-Instanzen einen Scan aus der Bagfile `laser.bag` in eine Punktwolke konvertieren und auf ein seperates Topic publishen. (1P)

(c) Konkatenieren Sie nun die zwei Punktwolken zweier Topics zu einer gemeinsamen Punktwolke. Nennen Sie diese Node `merge_clouds`. Um sich simultan auf zwei Topics zu subscriben, müssen zwei Nachrichten mit einem ähnlichen Zeitstempel an einen Callback geschickt werden. In ROS kann man dafür z.B. den `ApproximateTimeSynchronizer` benutzen. Sobald Ihnen die beiden Punktwolken in Ihrem Callback zur Verfügung stehen, müssen Sie diese beiden zu einer Punktwolke zusammenfassen. Dazu muss zunächst eine der beiden Punktwolken in das jeweilige andere Koordinatensystem transformiert werden. Konkatenieren Sie dann die beiden Punktwolken. Publishen Sie die resultierende Punktwolke auf ein Topic Ihrer Wahl und visualisieren Sie diese mit `RViz`.

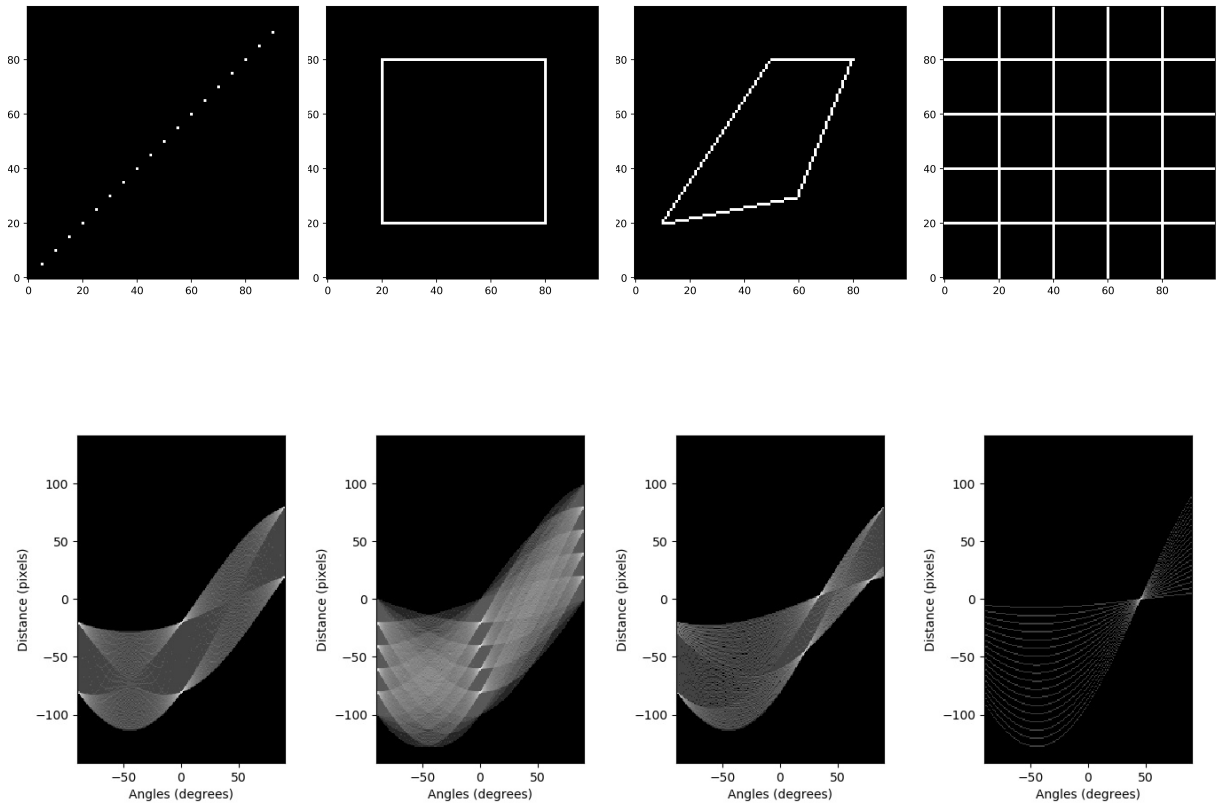
Tipp: Für eine Transformation einer Punktwolke hilft die Funktion `tf2::doTransform`. Die benötigten Includes sind:

```
#include <tf2_sensor_msgs/tf2_sensor_msgs.h>
#include <tf2/transform_datatypes.h>
```

(2P)

Aufgabe 2.3 (Theorieaufgabe: Hough-Transformation) **(2P)**

In der Vorlesung haben Sie die Hough-Transformation zur Linienerkennung kennengelernt. Die Bilder der ersten Reihe stellen Sensordaten (z.B. von einem Laserscanner) dar. In der zweiten Zeile sehen Sie die Hough-Räume, die aus diesen Sensordaten entstanden sind. Weisen Sie die Hough-Räume den jeweiligen Sensordaten zu.



Aufgabe 2.4 (Hough-Transformation) **(5P)**

In dieser Aufgabe werden Sie auf Ihrer Pointcloud vom aktuellen Übungsblatt eine Linienerkennung mit der Hough-Transformation durchführen. Ihre Node soll sich zu einem `Pointcloud`-Topic verbinden, anschließend alle Linien in einer Pointcloud erkennen und diese als `visualization_msgs/Marker` publishen um sie in `Rviz` darstellen zu können.

(a) Implementieren Sie ein geeignetes Voting-Array für die Parameter r und φ und detektieren Sie anschließend die Einträge mit den meisten Treffern. Probieren Sie verschiedene Diskretisierungen von θ (ca. zwischen 1° und 5° Grad) und r (ca. 10 cm). Achtung, r hat einen Wertebereich zwischen -10 m und 10 m. (3P)

(b) Wie ändert sich das Ergebnis, wenn Sie statt den ungefilterten Scandaten Ihren Medianfilter dazwischenschalten? Wäre der Reduktionsfilter eine angemessene Filtermethode? (1P)

(c) Visualisieren Sie neben den gefundenen Linien zusätzlich das Voting-Array als eine `sensor_msgs/Image`, so dass auch dieses direkt über Rviz angezeigt werden kann. Tipp: Das Encoding müssen Sie auf "mono8" stellen, womit sie 8-Bit Graustufen darstellen können. ***Hinweis:*** Da sie 8-Bit pro Pixel zur Verfügung haben, müssen Sie die Daten in Ihrem Voting-Array normalisieren. (1P)

(d) Verwenden Sie diesmal für die dynamische Parametrisierung `dynamic_reconfigure`. Tutorials und Dokumentation finden Sie unter: `http://wiki.ros.org/dynamic_reconfigure`. (1P)

Zum Testen können Sie auch die Bagfiles des vorherigen Blättes verwenden.

Eine Anleitung zum Umgang mit den Markern finden Sie hier: `http://www.ros.org/wiki/rviz/Tutorials/Markers%3A%20Points%20and%20Lines`.

Aufgabe 2.5 (Online-Linienerkennung) **(3P)**

(a) Implementieren Sie zusätzlich das lokale Finden von Linien gemäß den Vorlesungsfolien. Achten Sie dabei darauf, auch hier auf einer Pointcloud zu arbeiten, da die Berechung des euklidischen Abstandes in kartesischen Koordinaten einfacher ist. Benutzen Sie erneut einen `Marker` um die gefundenen Linien in `RViz` anzeigen zu können. (2P)

(b) Vergleichen Sie die Qualität der beiden von Ihnen implementierten Verfahren zum Linienfinden: Welches Verfahren liefert auf dem zur Verfügung stehenden Datensatz die besseren Ergebnisse? Welches ist rechenintensiver? Wie gut sind beide Verfahren ohne Medianfilter? (1P)

Aufgabe 2.6 (Datei-Upload)

Das Uploadlimit pro Datei pro Student liegt bei StudIP bei 10MB. Ihre reinen ROS-Pakete bestehend aus Launch-Files, Konfigurationsdateien, Source-Code, `CMakeLists.txt` und `package.xml` wird in der Summe dieses Upload-Limit vermutlich nicht überscheiten. Laden Sie deshalb das ROS-Paket ohne zusätzlichen Inhalt als Zip-Datei `blattX_gruppeY.zip` hoch, damit die Verzeichnisstruktur für uns erhalten bleibt. Alle anderen Abgaben in Form von Bildern oder PDFs laden Sie **einzeln** in einen seperaten Ordner hoch. Die Benennung der zusätzlichen Dateien sollte uns klar machen, zu welcher Aufgabe diese gehören. Ein Beispiel für Gruppe 1 und Übungsblatt 2:

```
Dateiordner der Gruppe: Gruppe_1
    blatt02_gruppe01.zip
    blatt02_gruppe01_zusatz/
        aufgabe1.pdf
        aufgabe2.png
```