Ausgabe am **21.06.2021**
Abgabe bis zum **06.07.2021 23:59 Uhr**

UNIVERSITÄT OSNABRÜCK

Priv. Doz. Dr. Thomas Wiemann, Alexander Mock

Robotik
Übungsblatt 6

Sommersemester 2021

Aufgabe 6.1 (Monte-Carlo-Lokalisierung) **(6P)**

(a) Beschreiben Sie die Idee der Monte-Carlo-Lokalisierung (MCL).

(b) Wozu dient dabei der *Resampling*-Schritt?

(c) Sind nicht-uniforme Rasterkarten (beispielsweise eine Quadtree-Repräsentation) für die MCL sinnvoll verwendbar? Wenn ja, wie? Wenn nein, warum nicht?

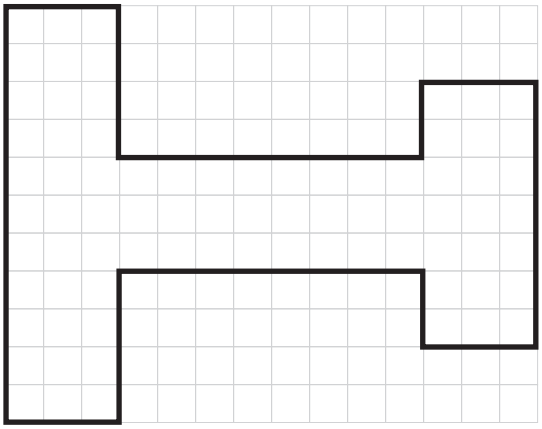
(d) Ist die Zahl der Partikel konstant oder variabel? Begründen Sie Ihre Antwort.

(e) Wäre die MCL auch auf nicht-metrischen (topologischen) Karten einsetzbar? Begründen Sie Ihre Antwort.

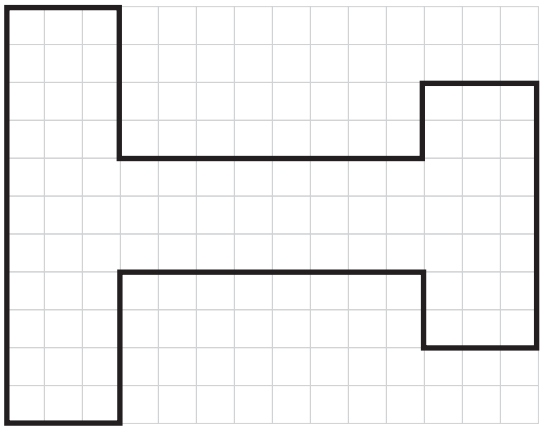
(f) Spielen Sie nun die einzelnen Schritte der Monte-Carlo-Lokalisierung für die folgende Abfolge von Aktionen in der unten gezeigten Umgebung mit den vorgegebenen Messergebnissen durch. Die Karten sind kontinuierlich, die Raster dienen nur als Hilfe für die Roboterbewegung.

Zeichnen Sie für jede Aktion jeweils eine plausible Partikelverteilung ein. Geben Sie zu jeder Zeichnung eine kurze Erklärung.

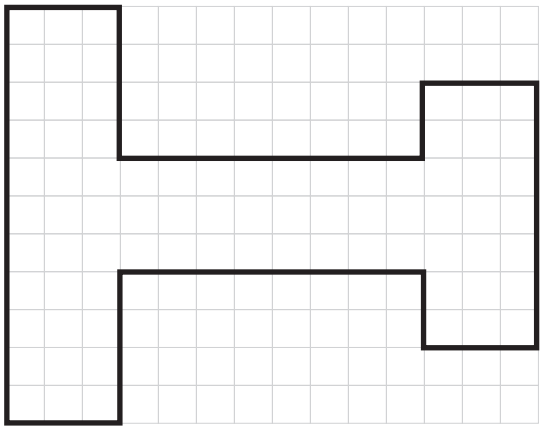
Initialisierung



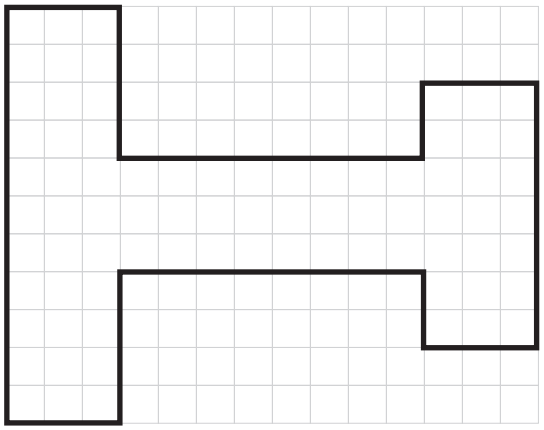
Erste Messung (Messergebnis links)



Bewegung um 2 Einheiten nach vorne



Zweite Messung (Messergebnis links)



Aufgabe 6.2 (Praktisch: Monte-Carlo-Lokalisierung, Partikel Bewegung + Resampling) (5P)

In den nächsten drei Aufgaben werden Sie eine Monte-Carlo Lokalisierung implementieren. In dem ersten Teil beginnen Sie mit der Initialisierung der Partikelmenge, dem *Motion Update* und dem *Resampling*.

(a) Starten Sie die Gazebo-Simulation und den `map_server`. Visualisieren Sie sich danach die geladene Karte in RViz. Klicken Sie dann in RViz auf "2D Pose Estimate". Nun können Sie in RViz eine Pose einzeichnen, die danach auf das Topic `initialpose` gepuplisht wird. Lassen Sie sich diese Pose auf der Kommandozeile ausgeben. Die Pose auf dem Topic `initialpose` können Sie als erste Schätzung der Roboterpose interpretieren. Schreiben Sie eine Node, die eine Pose von diesem Topic liest. Mit dieser Pose als Mittelwert und einer parametrisierbaren Standardabweichung sollen Sie nun eine Menge an Stichproben (Partikel) ziehen. Konvertieren Sie Ihre interne Partikel-Cloud Repräsentation in eine Nachricht vom Typ `geometry_msgs/PoseArray` und publishen Sie diese dann auf ein Topic. Visualisieren Sie sich das Topic dann mit RViz. (2P)

(b) Nehmen Sie nun eine Bewegungsschätzung des Roboters, um das *Motion-Update* auf den Partikeln durchzuführen. Eine mögliche Bewegungsschätzung können Sie aus

- Der Odometrie `odom`
- Ihrem Kalman-Filter des letzen Übungsblattes
- Aus der externen Node `robot_pose_ekf`

erhalten. Visualisieren Sie die Partikel mit RViz, während Sie den Roboter fernsteuern. (2P)

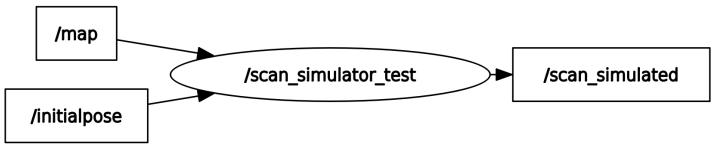
(c) Gehen Sie erstmal davon aus, dass im Sensor-Update alle Partikel die gleichen Gewichte erhalten. Setzen Sie auf dieser Basis den Resampling-Schritt um. (1P)

Hinweis: Diese Aufgabe kann unabhängig zur Aufgabe 3 bearbeitet werden.

Aufgabe 6.3 (Monte-Carlo-Lokalisierung, Simulation) **(4P)**

In dieser Aufgabe werden Sie einen Scan gegeben einer Pose in einer Karte simulieren. Der Nutzer soll eine Pose in der Karte einzeichnen, von dem aus ein LaserScan simuliert werden soll. Der resultierende LaserScan soll danach in RViz im Frame `map` sichtbar sein.

(a) Schreiben Sie dazu Node namens `scan_simulator_test`. Legen Sie die Subscriber und Publisher gemäß der folgenden Grafik an: (1P)



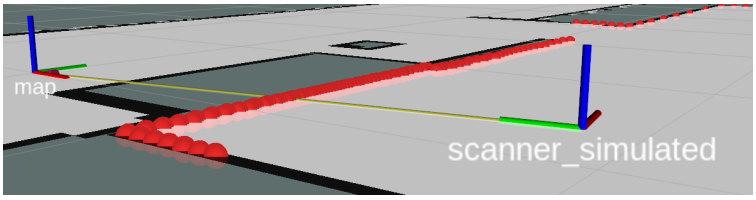
(b) Laden Sie sich das Paket `mcl_helper`[1] in Ihren `src`-Ordner und fügen Sie es als Abhängigkeit in Ihr eigenes Paket hinzu. Danach sollten Sie in Ihrer Node den `ScanSimulator` benutzen können:

```
#include <mcl_helper/scan_simulator.h>
using namespace mcl_helper;

ScanSimulator scan_sim;
...
// Initialize with map
// arguments: map [nav_msgs/OccupancyGrid]
scan_sim.setMap(map);
...
// Simulate a Scan.
// arguments: pose [geometry_msgs/Pose], scan [sensor_msgs/LaserScan]
// input: pose + scan (Meta-Data)
// output: scan (Ranges)
scan_sim.simulateScan(pose, scan);
```

Benutzen Sie nun die Messages der Subscriber, um diesen ScanSimulator zu initialisieren und von einer Pose aus einen Scan zu simulieren. Setzen Sie die Scan-Meta Daten (`angle_increment`, `angle_min`, ...) händisch, indem Sie diese aus dem echten Scan übernehmen. Publishen Sie danach den LaserScan im Frame `scanner_simulated` auf das Topic `scan_simulated`. Wenn Sie den `map_server` gestartet haben, sollte nun Ihre Node reagieren, sobald eine Pose in RViz gezeichnet wird. (2P)

(c) Der aktuelle Zustand erlaubt es noch nicht, sowohl den Scan als auch die Karte gleichzeitig in RViz zu visualisieren. Ermitteln Sie mit der gesetzten Pose eine Transformation zwischen `scanner_simulated` und `map` und broadcasten Sie diese über TF. Überprüfen Sie das Ergebnis über RViz. Dies sollte in etwa so aussehen: (1P)



Hinweis: Diese Aufgabe kann unabhängig zur Aufgabe 2 bearbeitet werden.

[1] `https://gitlab.informatik.uni-osnabrueck.de/robotik_ss21_packages/mcl_helper`

Aufgabe 6.4 (Praktisch: Monte-Carlo-Lokalisierung, Sensor-Update) (5P)

In den vorherigen Aufgaben haben Sie Partikel bewegt (Motion-Update) und Scans in einer gegebenen Rasterkarte simuliert. In dieser Aufgabe werden Sie die Monte-Carlo-Lokalisierung finalisieren, indem Sie das Sensor-Update implementieren.

(a) Ihre Node `mcl` wartet auf eine initiale Poseschätzung von RViz. Sobald eine Pose gesetzt wurde, beginnt die Monte-Carlo-Lokalisierung. Ziehen Sie eine Anzahl an Partikeln um die gesetzte Pose herum. Danach wird auf eine Roboterbewegung gewartet. Sobald der Roboter sich bewegt, führen Sie zunächst ein Motion-Update durch. (1P)

(b) Implementieren Sie danach das Sensor-Update welches die Partikel gewichtet. Simulieren Sie dazu von jedem Partikel aus einen LaserScan in der Karte und vergleichen Sie diesen mit dem echten Scan. Wenn die simulierten Werte denen des echten Scans entsprechen soll die Gewichtung 1 sein. Sind die Werte jedoch sehr verschieden, soll das Gewicht gegen 0 gehen. Als Funktion dazu eignet sich z.B.

$$V = \frac{1}{n} \sum_{i=1}^{n} e^{(\frac{s_i - r_i}{\sigma})^2} \tag{1}$$

wobei s_i eine Distanz des simulierten Scans ist und r_i eine Distanz des echten Scans. Das σ kann frei parametrisiert werden. (2P)

(c) Extrahieren Sie nun aus den gewichteten Partikeln die Roboterpose. Dazu können Sie eine dieser beiden Ansätze nehmen:

- Gewichteter Mittelwert
- Partikel mit bestem Gewicht

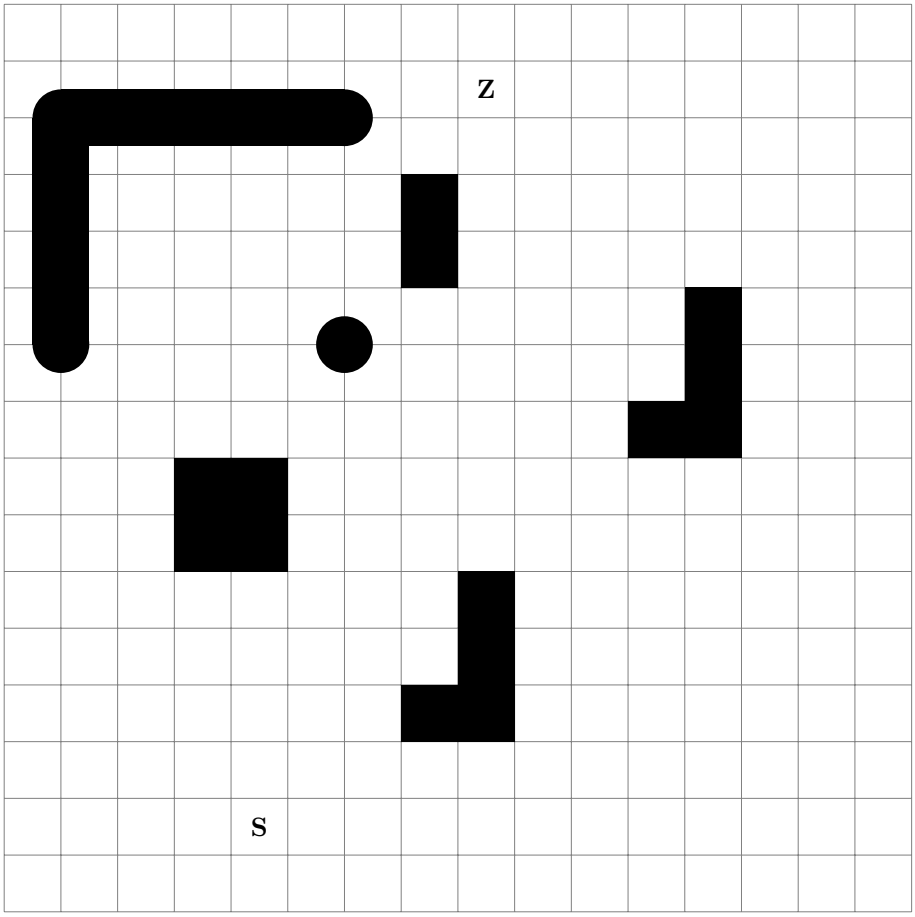
Begründen Sie die zu erwartenden Vor- und Nachteile dieser beiden Varianten. Publishen Sie die resultierende Pose im Karten-Koordinatensystem. (1P)

(d) Publishen Sie basierend auf der resultierenden Pose eine geeignete tf-Transformation zwischen Odometrie- und Karten-Koordinatensystem. Machen Sie sich dazu erst Gedanken welche Transformation die berechnete Lokalisierung eigentlich angibt. Wenn Sie alles richtig gemacht haben, sollten Sie nun in RViz sehen können, wo das Robotermodell in der Karte steht. (1P)

Hinweis: Für diese Aufgabe wird die Simulation und die Partikelbewegung benötigt.

Aufgabe 6.5 (Pfadplanung) **(4P)**

Zur Navigation eines Roboters sei die folgende Rasterkarte zur 2D-Navigation eines Roboters gegeben. Der Roboter sei kreisförmig mit einem Durchmesser von 20 cm und holonom in der Ebene, die räumliche Auflösung der Karte (Raster) sei 10 cm.



(a) Überführen Sie die Karte in einen geeigneten Konfigurationsraum für den gegebenen Roboter.

(b) Transformieren Sie die Karte in eine Quadtree-Darstellung. Die kleinste Auflösung sei durch die Gitterzellen gegeben.

(c) Tragen Sie die Zellgewichte ein, die sich für eine Wavefront-Planung ausgehend vom Ziel ergeben (Moore Nachbarschaft).

Hinweis: In einer Quadtree-Zerlegung hat nicht mehr jede Zelle eine uniforme Zahl von Nachbarzellen wie in einer uniformen Zerlegung!

Berechnen Sie alle kürzesten Pfade vom Start S zum Ziel Z mit Hilfe des Wavefront-Algorithmus ohne Beachtung der Zellgröße und zeichnen Sie die Pfade in die Karte ein.

Bewerten Sie alle Pfade hinsichtlich ihrer Befahrbarkeit.

Hinweis: Sehr ähnliche Pfade können Sie zusammengefasst betrachten.

(d) Nehmen Sie nun an, ihre Umgebung sei zu einem gewissen Teil dynamisch, beispielsweise durch die Anwesenheit von Menschen (siehe Aufgabe 5 (c)). Eine zuvor berechnete Trajektorie kann also unter Umständen nicht vollständig abgefahren werden. Erläutern Sie eine Möglichkeit, dieses Problem zu kompensieren, ohne bei jedes Mal komplett neu planen zu müssen.