Universität Bielefeld

Cognitronics & Sensor Systems

Projekt:
AMiRo Nachtwächter

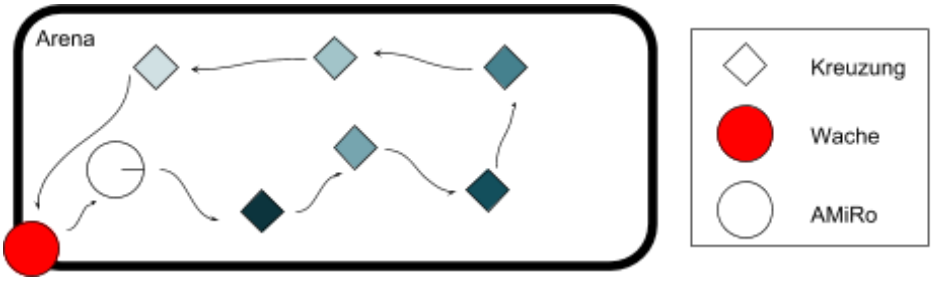
Abgabe: 2018-10-08
Optional: Mündlicher Vortrag
Gruppengröße: 2 Studenten (max. 3 mit Ausnahme)

Stellt euch vor, der AMiRo wäre ein Wachroboter, der einige Straßen patrouillieren muss. Jeden Abend werden ihm gut beleuchtete und breite Pfade zugewiesen, die per Kreuzungspunkte miteinander verbunden sind.

Bevor die Route beginnt soll der kürzeste Pfad ermittelt werden, mit der alle Straßen auf kürzestem Weg abgefahren werden können. Wie schwierig kann das sein? Dieses Problem ist in der Literatur als Nachtwächterproblem bekannt (*watchman route problem*), welches ein paar einfache Definitionen benötigt: Erstens, eine Anzahl von Kreuzungspunkten wird von einer Reihe von Pfaden miteinander verbunden. Zweitens, eine Wächterroute ist ein Satz von Pfaden, dessen gesamter Pfad geschlossen sein muss, d.h. der Anfangs- und der Endpunkt müssen identisch sein (damit der Wachroboter zum Hauptquartier zurückkehrt) und alle Pfadabschnitte müssen besucht werden.

Auf ähnliche Weise soll nun auch der AMiRo unterschiedliche Orte auf möglichst kurzem Weg abfahren. Hierbei müssen folgende Punkte berücksichtigt werden:

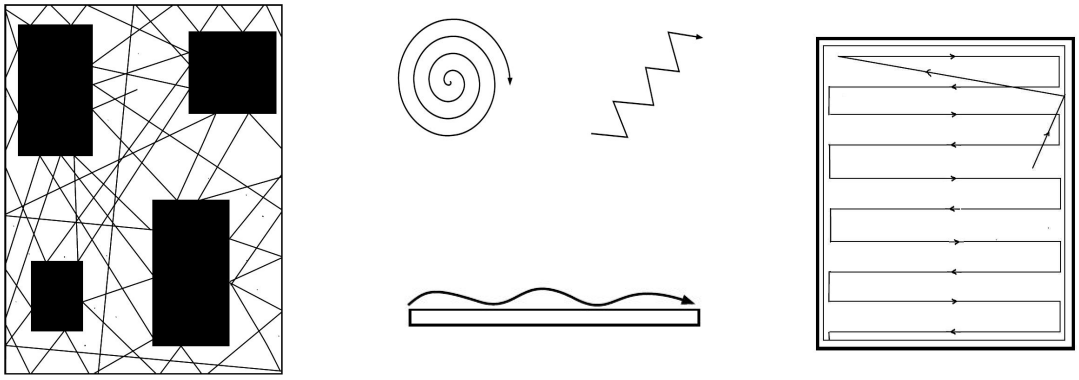
- Der Roboter darf sich frei zwischen verbundenen Kreuzungen bewegen
- Zur Lokalisierung zwischen den Kreuzungspunkten kann nur die Odometrie verwendet werden (diese weist den typischen *Drift-Fehler* auf)
- Die Kreuzungspunkte sind über Bodenmarkierungen unterschiedlicher Graustufen zu klassifizieren (Hinweis: unterschiedliche Reflexionskoeffizienten)
- Kreuzungspunkte dürfen zur Laufzeit entfernt werden, sodass der Roboter entsprechend reagieren muss
- Die Aufgaben werden zunächst in ROS/Gazebo gelöst und dann auf das AMiRo-OS übertragen. Achtet also auf eine möglichst generische Umsetzung.



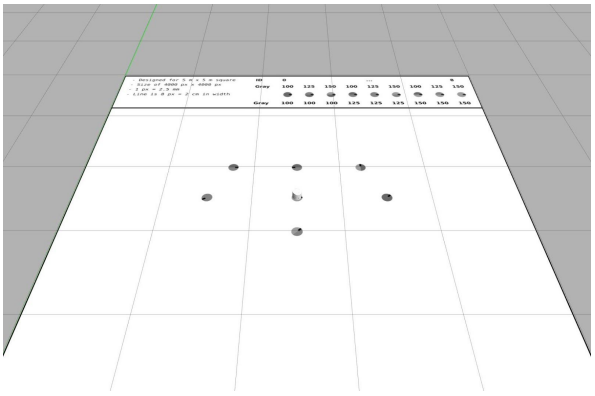
Exemplarische Nachtwächter-Route mit begrenzter Arena

A – Simulation

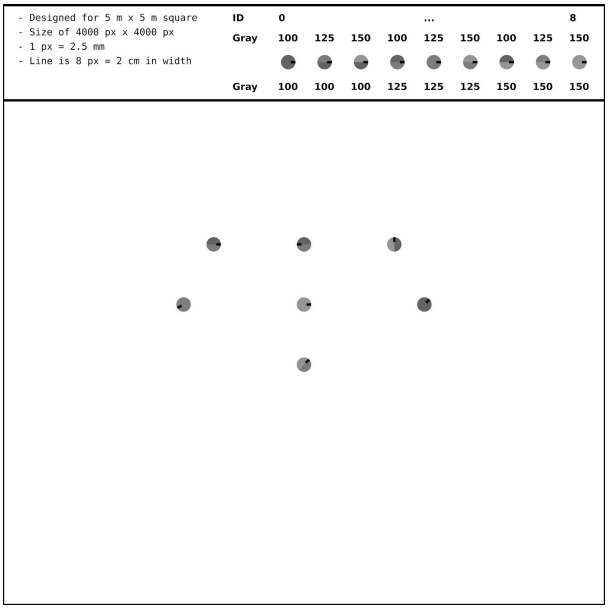
- Nutzen der vorgegebene Simulationsumgebung in Gazebo (**Beachte**: Es gibt mehrere Welten, und zur späteren Beurteilung werden neue herangezogen):
  Repository: *https://github.com/tik0/amiro_robot*
  Launch: *roslaunch amiro_gazebo amiro_watchmen_route_project.launch*
- Überlege dir vorher, welche Architektur die Problematik am besten lösen könnte. **Beachte**: Dokumentation anhand eines Programm-Ablauf-Plans, Zustandsautomaten, oder einer Architekturübersicht müssen vorhanden sein.
- **Aufgabe 1**: Schreibt einen Klassifikator, welcher bis zu **sechs** Kreuzungspunkte anhand der Bodensensoren identifizieren kann.
- **Aufgabe 2**: Realisiert eine Explorationsstrategie zur Kartographierung der Beacons aller Welten in eine YAML-Datei (Format/Welten: siehe vorhandene Dateien aus A3). Es soll die ***World* Odometrie verwendet werden**. Mögliche Explorationsstrategien sind unten grafisch abgebildet. Dokumentiere die benötigte Zeit zur Auffindung aller Beacons unter einer Maximalgeschwindigkeit von v = 0.1 m/s und w = 3.14 rad/s. Dokumentiert den Fehler eurer aufgenommenen Karte als 2-norm für Position und Drehung pro Beacon und gemittelt über alle Beacons gegenüber der *Ground-Truth* Karte. Diskutiere die Exploration unter Verwendung der ***Encoder* basierten Odometrie**.
- **Aufgabe 3**: Realisiert eine *Beacon-Navigation* auf Basis der ***Encoder* basierten Odometrie**, welche den Roboter zwischen Kreuzungspunkten navigieren lässt. Die Karte ist als Graph für jede Simulationsumgebung vollständig bekannt und liegt als YAML Datei vor:
  - Die unterschiedlichen Welten können können in der Datei *amiro_robot/amiro_gazebo/models/watchmen_route_project/materials/watchmen_route.material* (ändere *watchmen_route_0.png* zu einem anderen Bild ab)
  - Die YAML Dateien zu den jeweiligen Welten liegen unter *amiro_robot/amiro_gazebo/YAML/watchmen_route_**
  - Exakte Orte der Kreuzungspunkte sind bekannt und liegen bereits auf dem Parameterserver (Kreuzungspunkte sind base, beacon_0, …, beacon_5). Bsp. zum Laden der x,y,f (Meter,Meter,Grad) Position:
    std::vector<double> base_xyf
    node.getParam("/watchmen_route_0/base/pose2d", base_xyf)
  - **Hinweis**: Für das Anfahren von Kreuzungspunkten dürfen fertige Pakete wie *move_base* verwendet werden
- **Aufgabe 4**: Schreibe eine Pfadplanung, welche einen möglichst kurzen Pfad zum Abfahren aller Kreuzungspunkte herausfindet. Start und Ziel ist die jeweilige Wache.
  - Gib die Laufzeit deines Algorithmus an (O-Notation)
  - Gib die prognostizierte Gesamtstrecke an, welche gefahren wird
- Fahre den Pfad ab und nimm entsprechende Videos zur Dokumentation auf
- **Aufgabe 5**: Der Roboter muss auf Kreuzungspunkte reagieren können, welche nicht wie vorgesehen erscheinen.



Exemplarische Trajektorie eines *Random-Walk*s als Braitenberg-Vehikel (links), Spiral/S/Wall-Following Heuristik (mitte) und Mäander-Strategie (rechts)



Beispiel aus dem Simulator mit der Wache in der Mitte



Karte aus dem Beispiel mit rotations-varianten Markern

Bewertung: Grundsätzlich müssen alle Teilaufgaben in einem einheitlichen Ablauf gelöst werden. Zusätzlich wird die Qualität insbesondere der Pfadplanung und Reaktion auf Sondersituationen bewertet. Insgesamt macht diese Aufgabe 40% der Projektnote aus.

B – Umsetzung auf AMiRo

1. Portierung der Grundfunktionen aus Aufgabenteil A auf den AMiRo (Basisversion)
    - Repräsentation der Arena
    - Navigation mithilfe Encoder basierter Odometrie
    - Detektion und Identifikation der Kreuzungspunkte
    - Hindernisdetektion und -vermeidung
    - Pfadplanung entsprechend Aufgaben A.4 und A.5
2. Optimierung der Implementierung:
    - Verbesserung der Odometrie durch Verwendung von weiteren Sensoren (z.B. Accelerometer, Gyroskop, Magnetometer)
    - Optimierung der Ausführungsgeschwindigkeit ohne dabei Grundfunktionen einzuschränken (z.B. unzuverlässige Hindernisvermeidung oder fehlerhafte Beaconklassifikation)

Bewertung: Grundsätzlich müssen alle Teilaufgaben aus 1. und 2. gelöst werden (Grundanforderung). In die Bewertung fließen weiterhin die Qualität der Lösungen ein. Insgesamt macht diese Aufgabe 40% der Projektnote aus.

C – Abgabe

- Bericht
    - Inhalt
        - kurzer Hintergrund
        - Vorgehen und Methoden
        - Ergebnisse
    - Umfang
        - maximal 10 Inhaltsseiten pro Person
        - zzgl. Front- und Backmatter (Titelseite, Inhaltsverzeichnis, etc.)
        - kein Anhang!
- Demonstrationsvideo des Simultionsablaufes
- Demonstrationsvideo der realen Lösung
- Quellcode zu euren Lösungen

Bewertung: Es werden inhaltliche und sprachliche Qualität des Berichts bewertet. Insgesamt macht diese Aufgabe 20% der Projektnote aus.