Technology
Arts Sciences
TH Köln

Bericht zum Turtlebot Projekt Semantische Karte

Ben Konrad Meyer Second Student

Kurs: Cyber Physische Systeme

Lecturer: Professor Dr. Daniel Gaida

February 6, 2025

Abstract

This template can be used for seminar papers. For more tips and tricks regarding the use of figures, tables, quotations, references, footnotes, enumerations, etc. please download the Masterthesis template.

Contents

1 Einführung

1.1 Zielsetzung

1.2 Turtlebot

1.3 Gazebo

2 Navigation

2.1 Genutzte Sensoren

2.2 SLAM Bibliothek

2.2.1 Turtlebot Navigate

2.2.2 Übersetzung zwischen Koordinatensystemen

2.3 Findung des Navigationsziels

2.3.1 Erkundung

2.3.2 Wiederbesuch

2.4 Navigieren zum Ziel

Nachdem ein Ziel gefunden wurde muss noch der Weg zu diesem Ziel gefunden werden. Hierfür haben wir den A* Algorithmus eingesetzt, eine Weiterentwicklung des Dijkstra Algorithmus.

2.4.1 Dijkstra Wegfindung

Der Dijkstra Algorithmus ist ein Algorithmus um den kürzesten Weg zwischen einem Startknoten und allen anderen Knoten zu finden, mit der Option früher abzubrechen, wenn der Weg zu einem bestimmten Endknoten gefunden ist. Dabei ist er ein greedy Algorithmus der schrittweise den kürzesten Weg von dem Startknoten ausgehend bestimmt. Die einzigen Limitierungen an den Graphen, die der Dijkstra Algorithmus stellt, sind, dass der Graph verbunden sein muss (sonst existieren einige Wege nicht) und dass keine Gewichtung einer Verbindung negativ sind. Eine negative Gewichtung kann dazu führen, dass der Algorithmus nicht den optimalen Weg zum Endknoten findet, weil der Algorithmus abbricht, bevor der negative Weg in Betrachtung gezogen wird.
Der erste Anspruch kann in unserem Problem nicht garantiert werden - es ist durchaus möglich, dass Teile des sichtbaren Gebietes nicht erreichbar sind, weil der Weg dahin zu dünn für den Turtlebot ist. Manche Zielpunkte sind also nicht navigierbar, was wir bei der Implementierung beachten werden müssen.
Die zweite Bedingung können wir hingegen garantieren, weil die Gewichtungen unseres Graphen der Entfernung zwischen zwei Knoten entsprechen und damit nicht negativ sein können. Das die Realität des Systems negative Gewichte verbietet ist durchaus üblich, was diese Bedingung des Dijkstra Algorithmus in der Praxis selten relevant macht.
Der Algorithmus selber besteht aus vier Schritten: Initialisierung, Wahl des nächsten Knotens, Verarbeitung des Knotens und Wegrückverfolgung, wobei die mittleren beiden Schritte wiederholt werden, bis der Algorithmus fertig ist.
In der Initialisierung wird die Distanz bis zum Startknoten auf 0 gesetzt und die Entfernung zu allen anderen Knoten auf ∞ gesetzt. Ebenfalls werden alle Knoten als unbesucht markiert.
Dann wird der nächste Knoten gewählt. Dieser ist der Knoten mit der kleinsten Distanz, der noch nicht besucht wurde. Dieser lässt sich trivial in $O(n)$ Laufzeit finden, indem alle Knoten geprüft werden, aber mithilfe von Datenstrukturen wie Priority Queues lässt sich das auf $O(\log n)$ reduzieren.

Als letzter Schritt vor der Wiederholung kommt dann die Verarbeitung des Knotens. Hierbei wird für jeden Nachbarn des ausgesuchten Knotens geprüft, ob die Distanz des ausgesuchten Knotens zusammen mit der Gewichtung der Verbindung zwischen den beiden Knoten kleiner ist als die aktuelle Distanz des Nachbarn. Wenn ja, wird die Distanz des Nachbars dann auf diesen Wert gesetzt und der aktuelle Knoten im Nachbarn als Vorgänger eingetragen. Wenn dieser Schritt abgeschlossen ist, wird der aktuelle Knoten dann noch als besucht markiert.
Nachdem alle Knoten verarbeitet wurden (oder der Endknoten verarbeitet wurde, wenn nur ein Ziel existiert) kann dann eine Route gefunden werden. Dafür wird ein Zielknoten gewählt und von diesem dann der Kette von Vorgängerknoten gefolgt, bis diese den Startknoten erreicht. Diese Kette kann dann invertiert werden, um die Route zum Ziel zu erhalten.

2.4.2 A* Heuristik

Während der Dijkstra Algorithmus den schnellsten Weg zu allen Knoten findet, findet der A* Star Algorithmus nur den schnellsten Weg zu einem bestimmten Zielknoten. Dafür ist der Algorithmus aber performanter, da er Knoten primär in der Richtung des Zielknotens untersucht mithilfe einer Heuristik.

2.4.3 Implementierung

2.4.4 Verwaltung der aktuellen Aufgabe

3 Semantische Karte

3.1 Objekterkennung

3.2 Objekt Positionierung