Labyrinthlösung, Erkundung und Mapping durch Jetbot

Junfan Jin, Yihao Wang, Yan Li, Fengyou Wan

***Zusammenfassung*— TODO**

***Abstract*— TODO**

I. Einführung

In dieser Abschnitt handelt es sich um Übersicht der Papier, das Ziel, Struktur.

A. Übersicht

Aufgrund der Entwicklung der Algorithmen von mobilen Roboter leisten Roboter einen steigenden Beitrag zu unserer Industrie und unserem Leben. Dazu werden die autonome mobile Roboter in vielen unterschiedlichen Situation eingesetzt, beispielsweise Hilfe bei der Frachtabfertigung in Lagern sowie Kehrroboter für Haushalt. Wegen der autonomen Eigenschaft können vielen Menschenkraft gespart werden. In der Papier wird es versucht, die automatischen Mapping und Pfadplanung der mobilen Roboter zu erklären und realisieren.

B. Ziel

Für die bekannte Karten gibt es viele Methode, um den Pfad zu planen, wie Depth-First Search (DFS), Breadth-First Search (BFS), Iterative Deepening Search, A* Search usw. In dieser Papier wird es versucht, mit Apriltag die Hindernisse zu identifizieren und DFS zur Mapping zu verwenden. Es wird auch A*-Algorithmus eingesetzt, um den Pfad so zu planen, dass der Roboter von einer beliebigen Position als Anfangspunkt einen bestimmten Endpunkt am kürzesten erreichen kann.

C. Struktur

Die Struktur der Papier ist wie folgt: Dieser Teil stellt den Übersicht und die Ziele der Papier vor. Im zweite Teil handelt sich um einen Überblick über die Motorsteuerungstheorie, die DFS- und A*-Algorithmus. Der Abschnitt 3 beschreibt die Roboter-Hardware, die fürs Experiment verwendet wird. Kapitel 4 erläutert die Realisierung von Software und Algorithmen sowie die Probleme und Lösungen, die wir dabei getroffen haben. Abschließend werden in Kapitel 5 die Zusammenfassung dargelegt.

II. Grundlagen

Um den oben genannten Zweck zu erreichen, ist die folgende theoretische Grundlage erforderlich. In diesem Abschnitt wird es in drei Teile verteilt. Der erste Teil ist die Motorantriebssteuerung, der zweite Teil ist eine kurze Einführung in die DFS-Theorie und der dritte Teil wird kurz beschrieben der A*-Algorithmus.

A. Regelung der Motor

Subsection text.

B. Depth-First Search (DFS)

When selecting an edge to traverse, always choose an edge emanating from the vertex most recently reached which still has unexplored edges. A search which uses this rule is called a depth-first search. [1] DFS könnte in viele Situation eingesetzt werden, z. B. das Durchlaufen und Suchen in Binärbäumen. Das Prinzip der Algorithmus ist einfach und leicht zu implementieren. Aber der Pfad ist möglicherweise nicht der kürzeste. Wenn die Karte sehr groß ist, dann braucht diese Algorithmus mehr Zeit, um eine Lösung zu finden.

C. A* Search

A*-Algorithmus ist eine Erweitungsalgorithmus von Dijkstra-Algorithmus. Es besteht aus zwei Teilen, nämlich heuristische Funktion h(x) und Kost-Funktion g(x). Die Gesamtkosten lautet:

$$f(x) = g(x) + h(x)$$

Bei der heuristischen Funktion geht es hauptsächlich um die Schätzung der Entfernung zwischen der aktuellen Position und der Zielposition. Die Kost-Funktion soll die Entfernungskosten vom Startpunkt zur aktuellen Position berechnen. Für jeden Knoten sollten die Gesamtkosten berechnet werden. Im Vergleich zur DFS-Algorithmus kann A* in einer kürzen Zeit eine optimale Lösung finden. Wenn es eine Lösung gibt, dann kann es durch A* unbedingt herausgefunden werden, wobei A* keine Endlosschleife verursachen wird.

III. Hardware

Subsection text.

IV. Software

In diesem Absatz wird hauptsächlich die funktionale Implementierung der Software erläutert, die hauptsächlich in 5 Teile verteilt ist. Der erste Teil ist der Strukturrahmen des Codes. Der zweite Teil ist die Implementierung der Motorsteuerung, einschließlich PID-Steuerung und JetBot-Lagekorrektur. Der dritte Teil ist die Datenerfassung, hauptsächlich die Erfassung von Sensordaten, einschließlich Klassifizierung, Verpackung und Verarbeitung von Apriltag-Code und IMU-Daten. Der vierte Teil ist die Algorithmusimplementierung der Pfadplanung, die hauptsächlich den A*-Algorithmus im Code implementiert und die Knoten integriert und optimiert. Der

fünfte Teil besteht darin, einen Kartenalgorithmus zu erstellen, hauptsächlich durch das Sammeln, Analysieren und Erstellen einer Karte durch die informationen der Apriltag-Code.

A. Rahmen der Software

Der Software-Rahmen ist in vier Teile verteilt: Sammlungsmodul des Sensorendatens, Modul des Motorantriebs und der Steuerung, Modul der Algorithmus und zentrale Verarbeitungsmodul. Im Datensammlungsmodul werden die Abstands- und Winkelinformationen des Apriltag-Codes sowie die von der IMU gemessenen, physikalischen Informationen als Datenpaket in die Datenklasse JLocation gepackt. Durch Zugriff auf das Datenerfassungsmodul kann die aktuellen Standortinformationen erhalten werden. Dann ruft das zentrale Verarbeitungsmodul die Lagekorrektur im Motorantriebsmodul auf, um die aktuelle Lage und Orientierung des Jetbots zu korrigieren, und ruft dann erneut ein neues Datenpaket vom Datenerfassungsmodul ab und sendet es Nachdem das Algorithmusmodul das Datenpaket empfangen hat, werden die optimierten Pfadknoten an das zentrale Verarbeitungsmodul zurückgegeben. Schließlich sendet das zentrale Verarbeitungsmodul Bewegungsanweisungen an den Motorantrieb. Das Steuermodul erhält Echtzeitinformationen über die Bewegung des Fahrzeugs, indem es kontinuierlich auf das Datenerfassungsmodul zugreift, sodass die Bewegung des Fahrzeugs durch negatives Feedback gesteuert wird. Bei diesem Prozess kann ein Server auf Jetbot aufgebaut werden. Ein Teil wird auf dem Server platziert und dient zum Senden von Anweisungen an das Motorantriebssteuermodul und können auf Benutzerseite nämlich Client verarbeitet werden. Kartenerstellung und Pfadplanungsberechnung. Dadurch kann der Verbrauch von Autoressourcen bis zu einem gewissen Grad reduziert werden.

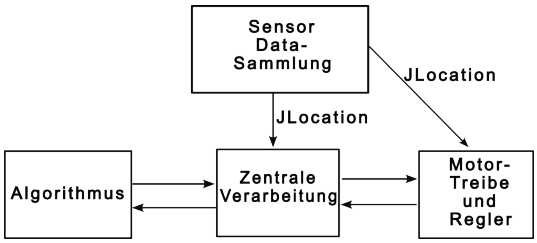

Abb. 1. Abbildung Software-Rahmen

B. Motor-Regelung

Subsection text.

1) Dies ist ein Unter-Unterabschnitt: Subsubsection text.

C. Sensor und Data-Sammlung

Subsection text.

1) Dies ist ein Unter-Unterabschnitt: Subsubsection text.

D. Pfad-Planung

Der Pfadfindungsalgorithmus verwendet den A*-Algorithmus. Zuerst wird die Kartendatei (Yaml-Datei) verarbeitet und die Daten werden in einer Matrix gespeichert, eine davon ist die Objektmatrix die andere ist die abstrakte Matrix. Die Objektmatrix wird die Wand-Objekte erstellen, wobei der Mittelpunktposition jeder Wand basierend auf dem Matrixindexwert speichert werden. Die abstrakte Matrix enthält nur die Zahlen 1 und 0, wobei 1 eine befahrbare Route und 0 eine unpassierbare Route, also eine Hinderniswand, darstellt. Dann werden zwei Klassen erstellt, eine davon ist die Knotenklasse, die zum Aufzeichnen der Standortinformationen des Knotens, der Informationen zum übergeordneten Knoten und der Gesamtkosten im A*-Algorithmus verwendet wird. Die heuristische Funktion hat zwei verschiedene Algorithmen ausprobiert, einer ist euklidische Distanzalgorithmus, einer davon ist der Manhattan-Distanzalgorithmus.Der euklidische Distanzalgorithmus ist der direkte tatsächliche physikalische geradlinige Abstand zwischen zwei Punkten, der mit der Formel unten berechnet werden kann.

$$d_{euklidisch} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Unter diesen ist x_1 die x-Koordinate des Startpunkts, x_2 die x-Koordinate des aktuellen Punkts und in ähnlicher Weise sind y_1 und y_2 die y-Koordinaten des Startpunkts bzw. des aktuellen Punkts. d ist der euklidische Abstand.Die Manhattan-Distanz ist die Summe der Differenzen zwischen den Koordinaten zweier Punkte. Es lautet

$$d_{Manhattan} = (x_2 - x_1) + (y_2 - y_1)$$

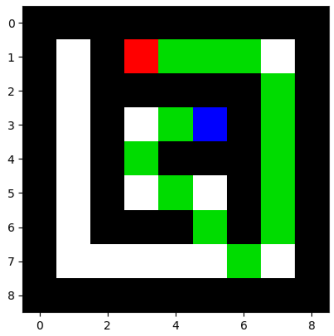

Abb. 2. Abbildung der A*-Algorithmus mit der Euklidischer Abstand

Die beiden obigen Bilder sind das Ergebnis der Berechnung mit der euklidischen Distanz Abb.2 bzw. der Berechnung mit der Manhattan-Distanz Abb.3. Bei Verwendung der euklidischen Distanz wird der aktuelle Knoten zum Erweitern um 8 Zellen verwendet, während die Manhattan-Entfernung zum Erweitern der Knoten in vier Richtungen verwendet wird, nämlich oben, unten, links und rechts. Zuerst werde den Startpunkt als ersten Knoten, dann wird der übergeordneten Knoten an None zugewiesen. Danach wird Knoten nach außen erweitet, dann alle Knoten nach dem Gesamtkostenwert sortiert, wird die Knoten mit dem kleinsten Kostenwert erweitet

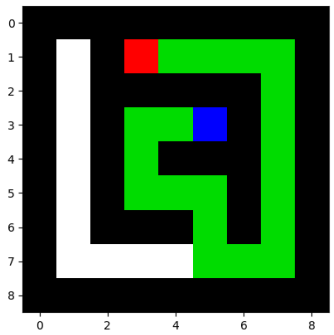

Abb. 3. Abbildung der A*-Algorithmus mit der Manhattan-Metrik

und der Vorgang wird die Sortierung und Erweiterung aller Knoten Knoten wiederholt, bis es schließlich zum Zielort erweitert wird, greift dann auf den übergeordneten Knoten bis zum letzten Knoten zu und sammelt den übergeordneten Knoten, um so Pfadinformationen zu erhalten, die alle Knoten enthalten. Auf dieser Grundlage muss man die Beziehung zwischen Knoten analysieren und Knoten mit derselben X- oder Y-Koordinate integrieren, sodass die Pfadinformationen nur Wendeknoten und Knoten an Gabelungen enthalten, wodurch der Bedarf an zentralen Verarbeitungseinheiten auf die Anzahl der Anweisungen für die Knoten reduziert wird Motorantriebssteuermodul.

E. Mapping und Erkundung

TODO

V. Zusammenfassung

Hier die wichtigsten Ergebnisse der Arbeit in 5-10 Sätzen zusammenfassen. Dies sollte keine Wiederholung des Abstracts oder der Einführung sein. Insbesondere kann hier ein Ausblick auf zukünftige Arbeiten gegeben werden.

Literaturverzeichnis

[1] R. Tarjan, *DEPTH-FIRST SEARCH AND LINEAR GRAPH ALGORITHMS*, 1971.

Autor Junfan Jin Biographie Autor Junfan Jin

Yihao Wang Biographie Autor Yihao Wang

Autor Yan Li Biographie Autor Yan Li

Autor Fengyou Wan Biographie Autor Fengyou Wan