Labyrinthlösung, Erkundung und Mapping durch Jetbot

Junfan Jin, Yihao Wang, Yan Li, Fengyou Wan

***Zusammenfassung*— Die vorliegende Arbeit befasst sich mit der autonomen Navigation und Pfadplanung in mobilen Robotern unter Verwendung von Apriltag-Erkennung und IMU-Daten. Die automatische Kartierung erfolgt durch die Identifizierung von Hindernissen mittels Apriltags, während der A*-Algorithmus zur Pfadoptimierung genutzt wird. Darüber hinaus wird die Implementierung der Software und die eingesetzte Hardware beschrieben, einschließlich der Herausforderungen und Lösungen bei der praktischen Umsetzung.**

***Abstract*— This thesis addresses the topic of autonomous navigation and path planning in mobile robots, with a focus on the use of April tag detection and IMU data. Obstacle identification is achieved through the use of April tags, with the A* algorithm employed for path optimization. Additionally, the software and hardware implementation is described, along with the challenges and solutions encountered during the practical implementation.**

I. Einführung

In dieser Abschnitt handelt es sich um Übersicht des Papiers, 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 diesem 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 diesem 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 des Papiers ist wie folgt: Dieser Teil stellt den Übersicht und die Ziele des Papiers 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

Die Regelung der Motoren des Jetbot Nano basiert auf zwei wesentlichen Prinzipien: der Steuerung der Antriebsgeschwindigkeit mittels PWM-Signalen sowie der präzisen Positionskontrolle durch die Nutzung von Apriltag-Erkennung und IMU-Daten.

1) Steuerung der Antriebsgeschwindigkeit: Der Jetbot Nano ist mit zwei unabhängigen Antriebsrädern ausgestattet, deren Geschwindigkeit über PWM (Pulse Width Modulation) geregelt wird. Die Steuerparameter ?Left? und ?Right? liegen im Bereich von [-1, 1] und werden proportional in entsprechende Antriebsspannungen [0, 6] und PWM-Werte [0, 255] umgewandelt. Die Vorzeichen der Left- und Right-Werte bestimmen die Laufrichtung der Motoren. Bei gleichen Werten für beide Parameter erfolgt eine Vorwärtsbewegung, bei entgegengesetzten Werten eine Rückwärtsbewegung. Durch unterschiedliche Vorzeichen ist zudem eine Drehung möglich. Ein Beispiel für die Wirkung von set_speed (-1.0, 1.0) ist eine Drehung nach links.

2) Präzise Positionskontrolle: Zur Sicherstellung der Bewegungsgenauigkeit werden Apriltag-Erkennung und IMU-Daten verwendet. Bei Vorwärts- oder Rückwärtsbewegung wird durch das motor-control-Modul sichergestellt, dass beide Räder gleich schnell drehen und die maximale Spannung sowie RPM begrenzt werden. Die von den IMU-Sensoren bereitgestellten Geschwindigkeits- und Beschleunigungsinformationen dienen der Verbesserung der Kontrollgenauigkeit, obwohl sie Messfehler wie Weißrauschen und Sensorabweichungen enthalten. Die Kontrollstrategie mit Apriltags ermöglicht die Korrektur der Position des Jetbots während der Fahrt. Ein

Pfadfindungsalgorithmus gewährleistet, dass der Jetbot den Zielort erreicht.

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

Der Jetbot Nano ist ein autonomes Navigationsroboter-Plattform, die auf dem NVIDIA Jetson Nano Entwicklungsboard basiert und speziell für das Lernen und Entwickeln von künstlicher Intelligenz und Robotik-Technologien entwickelt wurde. Das NVIDIA Jetson Nano-Modul stellt das Kernstück des Roboters dar. Es beinhaltet einen Quad-Core ARM Cortex-A57 CPU sowie eine 128-Core Maxwell GPU, welche eine hohe Rechenleistung zur Unterstützung von Deep Learning und Bildverarbeitungsaufgaben gewährleisten. Der Roboter ist mit 4 GB LPDDR4-Speicher und einem MicroSD-Kartensteckplatz ausgestattet, wobei die Benutzer die Speichergröße nach Bedarf auswählen können. Die Kamera verwendet ein CSI-Kameramodul, das für computerbasierte Aufgaben wie Objekterkennung, Verfolgung und Identifikation eingesetzt wird. Das Antriebssystem umfasst zwei unabhängig gesteuerte Gleichstrommotoren, welche Vorwärts-, Rückwärts- und Drehbewegungen ermöglichen.

Der integrierte IMU-Sensor (Inertiale Messeinheit) des Jetbot Nano umfasst einen Beschleunigungsmesser und ein Gyroskop, welche zur Messung von Beschleunigung und Winkelgeschwindigkeit sowie zur Unterstützung der Lage- und Bewegungssteuerung dienen. Das integrierte Energiemanagementsystem gewährleistet eine ausreichende Laufzeit durch einen

Abb. 1: Abbildung Jetbot Nano

Objekt	Daten
Länge (m)	0,134
Breit (m)	0,096
Höhe (m)	0,120
Gewicht (kg)	0,74
Radstand (m)	0,117
Rad-Durchmesser (m)	0,065
Ausladungslänge des Cameras (m)	0,015

TABELLE I: Parameter des Roboter Jetbot

Lithium-Akku sowie einen Schutz der Batterie und anderer Komponenten durch ein Energiemanagementmodul. Der Jetbot Nano verfügt über eine Vielzahl an Kommunikationsschnittstellen, darunter USB-Anschlüsse zum Anschluss externer Geräte sowie GPIO-, I2C- und SPI-Schnittstellen zum Anschluss weiterer Sensoren und Peripheriegeräte zur Erweiterung der Systemfunktionen. In Bezug auf die Software-Unterstützung ist der Jetbot Nano kompatibel mit dem JetPack SDK, welches das Linux-Betriebssystem, CUDA, cuDNN und TensorRT umfasst und eine vollständige Softwareentwicklungsumgebung bereitstellt. Des Weiteren wird ROS (Robot Operating System) unterstützt, welches in der Robotik-Entwicklung eine hohe Verbreitung aufweist und eine Vielzahl von Funktionspaketen und Tools bietet. In der Zusammenfassung lässt sich festhalten, dass der Jetbot Nano, der eine hohe Rechenleistung mit einer Vielzahl von Sensoren verbindet, eine ideale Plattform für das Erlernen und Entwickeln von künstlicher Intelligenz, Computer Vision und Robotik-Technologien darstellt.

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.

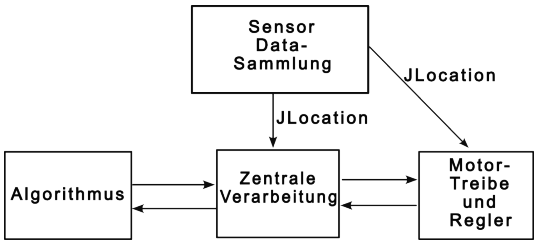

Abb. 2: Abbildung Software-Rahmen

B. Sensor und Data-Sammlung

In diesem Abschnitt erfolgt eine detaillierte Beschreibung der Mechanismen zur Erkennung und Datenverarbeitung der wichtigsten Sensoren des Jetbot Nano. Die Apriltag-Erkennung sowie die IMU-Datenerfassung stellen dabei die wesentlichen Elemente dar. Die präzise Bewegungskontrolle und autonome Navigation des Jetbots werden maßgeblich durch diese beiden Technologien gewährleistet.

1) Apriltag-Erkennungsmechanismus: Der Apriltag ist ein zweidimensionaler Barcode, der für die Computer Vision entwickelt wurde und ähnlich wie ein QR-Code ist, jedoch eine höhere Positionierungsgenauigkeit bietet. Der Erkennungsmechanismus umfasst die folgenden Schritte: Zunächst wird ein Bild der Umgebung mit einer Kamera aufgenommen und das Bild wird durch Graustufen- und Binärisierungsverfahren vorverarbeitet, um den Kontrast des Tags zu erhöhen. Im Anschluss werden mithilfe von Kantendetektionsalgorithmen (beispielsweise Canny-Kantendetektion) alle Kanten im Bild identifiziert. Basierend darauf werden in dem kantenbasierten Bild geschlossene Konturen gesucht und anhand geometrischer Form und Größe Kandidatenregionen gefiltert, die keine Apriltags sein können. Im Anschluss werden die verbleibenden Kandidatenregionen dekodiert und anhand vordefinierter Kodierungsregeln die Tag-ID erkannt. In einem letzten Schritt wird durch die vier Eckpunkte des Tags und unter Zuhilfenahme des PnP-Algorithmus (Perspective-n-Point) die Pose (Position und Orientierung) des Tags relativ zur Kamera berechnet.

2) IMU-Signalerfassungsprinzip: In der Regel ist der Jetbot Nano mit einer IMU (Inertial Measurement Unit) ausgestattet, welche zur Messung der Beschleunigung und Winkelgeschwindigkeit des Fahrzeugs dient. Das Prinzip der Erfassung von Signalen einer Inertial Measurement Unit (IMU) umfasst folgende Aspekte: Eine IMU umfasst in der Regel einen Beschleunigungsmesser sowie ein Gyroskop und in einigen Fällen auch ein Magnetometer. Der Beschleunigungsmesser ist in der Lage, die Beschleunigung entlang der drei Achsen zu messen, während das Gyroskop die Winkelgeschwindigkeit um die drei Achsen bestimmt. Das Magnetometer hingegen ist in der Lage, die Richtung des Erdmagnetfelds zu erfassen, um die Orientierung zu bestimmen. Der Beschleunigungsmesser und das Gyroskop erfassen die Daten mit einer bestimmten Frequenz (z. B. 100 Hz), wodurch Rohdaten der Beschleunigung und Winkelgeschwindigkeit erzeugt werden. Die Umwandlung der analogen in digitale Signale erfolgt durch einen Analog-Digital-Wandler (ADC). Die Rohdaten werden durch Filter, beispielsweise Kalman-Filter oder Tiefpassfilter, verarbeitet, um Rauschen zu entfernen, und mithilfe von Fusionsalgorithmen kombiniert, um stabilere und genauere Lageinformationen zu erhalten. Die verarbeiteten IMU-Daten werden über Kommunikationsschnittstellen wie I2C, SPI oder UART an das Jetson Nano Hauptsteuerungsboard übertragen, wo die Software diese Daten weiterverarbeitet, um sie für Navigations- und Lagekontrollaufgaben zu verwenden. Die Steuerung des Jetbots basiert auf der Verarbeitung der IMU-Daten, welche für die Aufgabenbereiche der Gleichgewichtskontrolle, Routenplanung sowie der Hindernisvermeidung genutzt werden. Des Weiteren besteht die Möglichkeit, die IMU-Daten mit anderen Sensordaten, wie beispielsweise Kameradaten oder Ultraschallsensoren, zu kombinieren, um die Navigation und die Umgebungswahrnehmung zu optimieren. Aufgrund dieser Prinzipien ist der Jetbot Nano in der

Lage, eine präzise Bewegungssteuerung sowie eine autonome Navigation zu realisieren.

C. Motor-Regelung

In zahlreichen Arbeitsbereichen müssen Roboter ihre eigene Geschwindigkeit und exakte Position für die Bewegungssteuerung, Navigation und Interaktion mit der Umgebung ermitteln. Die Trägheitsmesseinheit (IMU) weist aufgrund ihres eigenen akkumulierten Fehlers eine Ungenauigkeit auf, kann jedoch räumliche Informationen über die sechs Freiheitsgrade erfassen. Dies hat den Vorteil, dass sie räumliche Informationen in Echtzeit bereitstellen kann. AprilTags sind für eine hohe Lokalisierungsgenauigkeit ausgelegt und die genaue 3D-Position des AprilTags in Bezug auf die Kamera kann berechnet werden. Aufgrund der langsamen Aktualisierungsrate ist die Echtzeitleistung relativ schlecht. Um eine genaue räumliche Position in Echtzeit zu erhalten, wird ein erweiterter Kalman-Filter implementiert, der visuelle und inertiale Daten zusammenführt und eine genaue Schätzung der Position liefert.

1) Theoretische Grundlagen: Die Homographie-Matrix, die zur Beschreibung der relativen Positionen zweier Ebenen im Raum verwendet wird, ist die theoretische Grundlage der AprilTag Technologie. Die relative Position der AprilTag Ebene und der Kameraebene kann durch die Homographie-Matrix der beiden Ebenen bestimmt werden. Kamera-Parameter-Matrix P enthält die internen Parameter der Kamera, wie Brennweiten (f_x und f_y) sowie die Hauptpunktkoordinaten (u_0 und v_0). Die Form ist wie folgt:

$$P = \begin{bmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Kamera-Parameter-Matrix E enthält die externen Parameter der Kamera, das heißt die Rotationsmatrix (R) und den Translationsvektor (T), die die Position und Orientierung der Kamera im Weltkoordinatensystem beschreiben. Die Form ist wie folgt:

$$E = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

Die Homographie-Matrix H kann somit dargestellt werden als: $H = P \cdot E$

2) Kalibrierung: Kalibrierung der intrinsischen IMU-Parameter und Erstellung einer IMU-Konfigurationsdatei mit Rauschdichte und Driftwerten. Aufzeichnung der Rohdaten der Bildströme und IMU-Messungen in einem ROS-Bag. Alle IMU-Achsen durch Drehungen und Translationen anregen, Bewegungsunschärfe vermeiden. Durchführung einer Batch-Optimierung zur Modellierung der Systempose. Bestimmung der Transformationsmatrizen T_{ci} (IMU zu Kamera) und T_{ic} (Kamera zu IMU). Diese Kalibrierung ermöglicht die präzise Fusion von visuellen und inertialen Daten, was für Anwendungen wie Robotik und erweiterte Realität wichtig ist.

3) Kalmanfilter:

- Koordinatensysteme
  Im Filtersetup werden verschiedene Koordinatenrahmen angenommen:
  W: Inertiales Arbeitsraum-Koordinatensystem
  B: IMU-Koordinatensystem
  V: Kamerakoordinatensystem
  T: AprilTag-Koordinatensystem

- Zustandsvorhersagemodell
  Das Zustandsvorhersagemodell wird verwendet, um den Systemzustand über die Zeit basierend auf den von der IMU bereitgestellten Beschleunigungs- und Winkelgeschwindigkeitsdaten vorherzusagen. Die Zustandsgrößen umfassen:
  r: Position
  v: Geschwindigkeit
  q: Orientierung
  b_f: Beschleunigungs-Bias
  b_w: Winkelgeschwindigkeits-Bias
  r_T und q_T: Position und Orientierung des AprilTags
  r_V und q_V: Extrinsische Kalibrierungsparameter zwischen IMU und Kamera
  Der Zustandsvektor kann als folgt dargestellt werden:

$$x = \left(r, v, q, b_f, b_w, r_T, q_T, r_V, q_v\right)$$

  Die Vorhersagegleichungen des Modells sind:

$$\begin{aligned} \omega_{t+1} &= \omega_t + \Delta t \cdot (\widetilde{\omega_t} - b_\omega - n_\omega) \\ f_{t+1} &= f_t + \Delta t \cdot (\widetilde{f_t} - b_f - n_f) \\ r' &= r + \Delta t \cdot \left(v + 0.5 \cdot g \cdot \Delta t^2\right) \\ v' &= v + \Delta t \cdot (q \cdot f - g) \\ q' &= q \cdot e^{\Delta t \cdot \omega / 2} \end{aligned}$$

  $\widetilde{\omega}$ und $\widetilde{f}$: Messwerte der IMU
  b_w und b_f : Bias-Fehler
  n_ω und n_f: Weißrauschen
  g: Erdbeschleunigung

- Aktualisierungsmodell
  Das Aktualisierungsmodell basiert auf der Verwendung der Position der April-Tag-Eckpunkte im Kamerabild, um den Systemzustand zu korrigieren. Die Berechnung der Pixelkoordinaten der Eckpunkte erfolgt mittels der Kameraprojektionsabbildung π:

$$\begin{aligned} \widetilde{p_i} &= \pi\left(r_{T_i}^C\right) + n_{p,i} \\ y_i &= \widetilde{p_i} - p_i \end{aligned}$$

  $r_{T_i}^C$: Position des AprilTag-Eckpunkts im Kamerakoordinatensystem
  $n_{p,i}$: additive Gaußsche Rauschen
  p_i: im Bild extrahierten Eckpunkte

  Der spezifische Prozess des Aktualisierungsmodells umfasst:

1. Berechnung der 3D-Position der AprilTag-Eckpunkte r_T
2. Transformation der 3D-Position in 2D-Bildkoordinaten $\widetilde{p_i}$ mittels der Kameraprojektionsabbildung π
3. Berechnung des Fehlers y_i zwischen den vorhergesagten Werten $\widetilde{p_i}$ und den tatsächlichen Beobachtungswerten π
4. Anpassung des Zustandsvektors x unter Verwendung des Kalman-Gewinns, um den Vorhersagefehler zu minimieren

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

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

Die beiden obigen Bilder sind das Ergebnis der Berechnung mit der euklidischen Distanz Abb.3 bzw. der Berechnung mit der Manhattan-Distanz Abb.4. 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 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

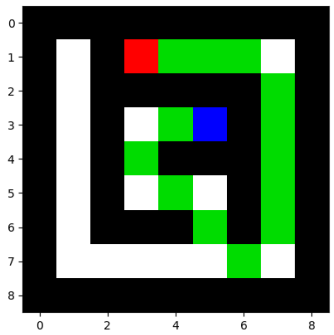

Abb. 3: Abbildung der A*-Algorithmus mit der Euklidischer Abstand

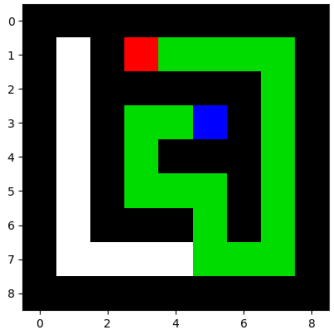

Abb. 4: Abbildung der A*-Algorithmus mit der Manhattan-Metrik

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

TODO

Literaturverzeichnis

[1] R. Tarjan, *DEPTH-FIRST SEARCH AND LINEAR GRAPH ALGORITHMS*, 1971.
[2] M. Li and C. Zhang, "A spatial pose measurement scheme by using imu and apriltag technologies," in *2018 IEEE International Conference on Information and Automation (ICIA)*, 2018, pp. 1048–1052.

Autor: Junfan Jin Biographie Autor Junfan Jin

Autor: Yihao Wang Biographie Autor Yihao Wang

Autor Yan Li Biographie Autor Yan Li

Autor Fengyou Wan Biographie Autor Fengyou Wan