Technische Hochschule Ingolstadt

Roboternavigation mit Potenzialfeldern

Intelligente Robotik WS2023/24
Praktische Arbeit

Carl Schünemann (Mat.Nr. 00107827)

23. Dezember 2023

Inhaltsverzeichnis

1 Ziel der Implementierung **1**

2 Ausführung der Implementierung **2**

3 Roboterbewegung im Occupancy Grid **3**

4 Konfigurationsraum **4**

5 Berechnung der Potenzialfelder **5**
5.1 Anziehende & Abstoßende Potenziale 5
5.2 Wavefront Potenziale 6

6 Roboternavigation im Kraftfeld **7**
6.1 Berechnung der Gradienten 7
6.1.1 Gradienten an Grenzen und Hindernissen 8
6.1.2 Behandlung lokaler Maxima 8
6.2 Gradientenabstieg 8

7 Diskussion **10**
7.1 Vergleich der Potenzialfeldmethoden 10
7.2 Skalierung des Occupancy Grids 10

Literatur **11**

1 Ziel der Implementierung

Gegebene Aufgabenstellung:

Es soll ein Planungssystem zur Mikronavigation implementiert werden. Die Planung erfolgt auf Basis eines statisch gegebenen, einfachen Occupancy Gids mit Hilfe der Potenzialfeldmethode. Der Roboter besitzt eine rechteckige Form variabler Größer. Der gefundene Plan wird visualisiert. Planungsstatistiken werden geführt. Die Leistungsfähigkeit des Planers wird anhand verschiedener Planungsszenarien bewertet.

Umgesetzt durch: - Python Jupyter Notebook - Einzige externe Bibliothek zur Durchführung Mathematischer Operationen: numpy $>$ Fokus der Implementierung - Robotermodell: - Rechteckiger Roboter mit Variabler Größe (width, length) - Rotation des Roboters um Änkerpunktïm und gegen den Uhrzeigersinn in 90° Schritten (0°, 90°, 180°, 270°) $>$ Änkerpunkt- Ecke "links oben"bei 0° Rotation - Konfigurationsraum: - Roboter befindet sich in statischem Occupancy Grid: 2D Array variabler Größe: Wenn sich An Koordinate (X,Y) ein Hindernis befindet $=>$ False, sonst True - Um Roboterrotationen zu berücksichten: Transformation des 2D Occupancy Grid in einen 3D Konfigurationsraum: - Ebene in Z-Richtung entspr. Roboterrotation $=>$ 4 Ebenen - In jeder Ebene wird Occupancy Grid um die Dimensionen des Roboters in der jeweiligen Rotation erweitert - Potenzialfelder: - Roboter wird Startpunkt/-rotation und Zielpunkt/-rotation im Occupancy-Grid gegeben - Berechnung von Potenzialfeldern in Konfigurationsraum 1) Mit Attractive (anziehend zu Zielpunkt) / Repulsive Potenzialen - Roboternavigation: - Berechnung der Potenzialgradienten - Gradientenabstiegsverfahren

2 Ausführung der Implementierung

- Implementierung in Python mit einem Jupyter Notebook. - Quellcode auf GitHub: (QR-Code)

- Bedingungen zur Ausführung: Python $>=$ 3.7 - Python Abhängigkeiten: $>$ matplotlib zur Visualisierung $>$ numpny für Mathematische Berechnungen $=>$ Installation aller Abhängigkeiten: pip install -r requirements.txt

- Empfehlung zur Ausführung: Anacoda Environment

3 Roboterbewegung im Occupancy Grid

Die physikalische Umgebung, in der sich der Roboter bewegen kann, wird diskretisiert durch das *Occupancy Grid*. Dieser binäre, zweidimensionale Raum entspricht der Umgebung aus "Vogelperspektive".

Das numpy Boolean Koordinatensystem `occupancy_grid` mit der Länge `occupancy_grid_length` und Breite `occupancy_grid_width` hat den Ursprung links oben. Bedingt durch Numpy werden Koordinaten mit `occupancy_grid[Y][X]` referenziert. Die Umgebung ist im Occupancy Grid binär, wodruch jede Koordinate (X,Y) durch ein Hindernis belegt ist (`occupancy_grid[Y][X] == False`) oder frei von Hindernissen ist (`occupancy_grid[Y][X] == True`).

*** TODO: Abbildung 2D True/False Array und geplottetes Occupancy Grid ***

Gemäß der gestellten Anforderungen kann die Dimension des Roboters durch die Variablen `robot_width` und `robot_length` definiert werden. Pro Verarbeitungseinheit kann sich der Roboter entweder durch eine Translation oder Rotation im Occupancy Grid bewegen:

- **Translation** nach links (`x-1`), rechts (`x+1`), oben (`y-1`) und unten (`y+1`)
- **Rotation** um einen *Ankerpunkt*. Bei einer Rotation von $0\check{r}$ liegt dieser Referenzpunkt in der linken oberen Ecke des Roboters.

** TODO: Abbildung Robotermodell + Translation/Rotation ***

4 Konfigurationsraum

Bei einer Roboterlänge und -breite größer als eins können Punkte im Occupancy Grid nicht erreicht werden, ohne mit den Roboterdimensionen Hindernisse oder Grenzen zu überdecken. Aus diesem Grund wird die Bewegung des Roboters im Occupancy Grid in eine Punktbewegung im sogenannten *Konfigurationsraum* transformiert. In diesem Raum entsprechen die Roboterdimensionen einem Punkt (Roboterlänge $=$ Roboterbreite $= 1$), was die Überdeckung von Hindernissen und Grenzen verhindert.

Zur Berechnung des Konfigurationsraums wird jedes Hindernis im Occupancy Grid um die Roboterdimensionen erweitert. Aufgrund der variablen Roboterdimension ändern sich die überdeckten Koordinaten im Occupancy Grid je nach aktueller Rotation. Ansätze in der Literatur schlagen deshalb vor, unabhängig von der aktuellen Rotation des Roboters, jedes Hindernis in X- und Y-Richtung um einen Kreis mit maximaler Roboterdimension als Radius zu vergrößern.

3D Konfigurationsraum

XYZ (TODO) schlägt einen restriktiveren Ansatz vor, um möglichst wenig Koordinaten um das Hindernis im Konfigurationsraum zu erweitern. Dazu wird die Roboterrotation in diskrete Schritte unterteilt. Die Variable `rotation_step` gibt als Teiler von 360° an, um wie viel Grad sich der Roboter pro Bewegungsschritt drehen kann. Somit sind `rotations` $= (360° \div$ `rotation_step`$)$ unterschiedliche Ausrichtungen des Roboters um den Ankerpunkt möglich.

Für das Robotermodell wird eine Maske als 2D-Array (`robot_length`$*$`robot_width`) erstellt und für jeden Rotationsschritt mit `sklearn.rotate()`[1] gedreht. Für jede der $(360° \div$ `rotation_step`$)$ möglichen Rotationen wird ein *erweitertes Occupancy Grids* generiert. Dazu werden Hindernisse sowie Grenzen des ursprünglichen Occupancy Grids um die rotierte Maske erweitert. Beispielsweise Rotation um 45° ...

*** TODO: Beispiel für Erweiterung des Roboters um Hindernis ***

Die erweiterten Occupancy Grids werden zum Konfigurationsraum `configuration-space[rotation][y][x]` mit den Dimensionen `rotations` $*$ `occupancy_grid_length` $*$ `occupancy_grid_width` zusammengefasst. Für die Dimension `[rotation]` gilt:

- [(`rotation` $+ 1$) $\mod$ `rotations`] $\widehat{=}$ Rotation um `rotation_step` im Uhrzeigersinn
- [(`rotation` $- 1$) $\mod$ `rotations`] $\widehat{=}$ Rotation um `rotation_step` gegen den Uhrzeigersinn

*** TODO: Abbildung 3D Plot ***

Durch diesen 3D Konfigurationsraum wird somit die kollisionsfreie Translation und Rotation bei minimalem Verbrauch des freien Konfigurationsraums ermöglicht.

[1] Mögliche Artefakte und Interpolation der Matrixrotation müssen manuell korrigiert werden.

5 Berechnung der Potenzialfelder

Gemäß der Aufgabenstellung erfolgt die Pfadplanung der Roboternavigation zu einem Zielpunkt mit der *Potenzialfeldmethode*:

Allgemein betrachtet, entspricht der Konfigurationsraum einem skalaren *Potenzialfeld*, wobei jede Koordinate eine potenzielle Energie $U(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$ besitzt, ausgedrückt durch eine reelle Zahl. Je Höher die potenzielle Energie einer Koordinate, desto weiter ist der Punkt auf dem aktuellen Pfad vom Ziel entfernt. [4] Die Berechnung des Potenzialfelds $\mathtt{potential[rotation][y][x]} = U(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$ erfolgt mit *Potenzialfunktionen*.

5.1 Anziehende & Abstoßende Potenziale

Khatib schlug 1986 vor, das Potenzialfeld in Analogie zu magnetischen Feldern zu berechnen. Der zu erreichende Zielpunkt wirkt mit einem *anziehenden* (engl. *attractive*) Potenzial auf eine Koordinate ($\mathtt{y}$, $\mathtt{x}$) [2]. Das Potenzial ist in allen Rotationsebenen gleich:

$$U_{Attr,Ziel}(\mathtt{x}, \mathtt{y}) = \sqrt{(\mathtt{y} - \mathtt{y}_{Ziel})^2 + (\mathtt{x} - \mathtt{x}_{Ziel})^2}$$

Bei den getesteten Implementierungsszenarien wurden bessere Ergebnisse festgestellt, wenn das anziehenden Potenzial erst normiert und anschließend mit `attraction_weight` gewichtet wird:

$$U_{Attr,Ziel}(\mathtt{x}, \mathtt{y})_{norm} = \frac{U_{Attr,Ziel}(\mathtt{x}, \mathtt{y})}{\max\{U_{Attr,Ziel}\}} * \mathtt{attraction\_weight}$$

Die Hindernisse im Konfigurationsraum wirken auf jede Koordinate (`rotation`, $\mathtt{y}$, $\mathtt{x}$) mit einem *abstoßendem* (engl. *repulsive*) Potenzial. Dazu zählen auch die Hindernisse aus der benachbarten Rotationsebene ($(\mathtt{rotation} + 1) \mod \mathtt{rotations}$) sowie ($(\mathtt{rotation} - 1) \mod \mathtt{rotations}$):

$$U_{Repul,Hindernis}(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = \frac{1}{\mathtt{repulsion\_weight} + \sqrt{(\mathtt{y} - \mathtt{y}_{Hindernis})^2 + (\mathtt{x} - \mathtt{x}_{Hindernis})^2 + (\mathtt{rotation} - \mathtt{rotation}_{Hindernis})^2}}$$

Yujiang und Huilin definieren das abstoßende Potenzial an der Koordinate (`rotation`, $\mathtt{y}$, $\mathtt{x}$) als kleinsten Abstand zu allen Hindernissen [4]:

$$U_{Repul}(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = \min_{\forall\ Hindernis\ \in\ \mathtt{occupancy\_grid}} \{U_{Repul,Hindernis}(\mathtt{x}, \mathtt{y}, \mathtt{rotation})\}$$

*** TODO: Abbildung mit Plot ***

Die gesamte potenzielle Energie einer Koordinate entspricht der Kombination beider Potenziale. In der Literatur werden dazu unterschiedliche Ansätze vorgeschlagen. Beispielsweise wählt XYZ das Maximum beider Potenziale. In dieser Implementierung wurden anziehendes und abstoßendes Potenzial gemäß Khalib addiert [2]:

$$U(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = U_{Attr,Ziel}(\mathtt{x}, \mathtt{y})_{norm} + U_{Repul}(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$$

5.2 Wavefront Potenziale

Choset stellt als Alternative zur Potenzialfunktion basierend auf anziehenden und abstoßenden Potenzialen die Verwendung der Breitensuche ausgehend vom Zielpunkt vor. Beim sogenannten *Wavefront-Algorithmus* entsprechen die Koordinaten des Konfigurationsraums den Knoten, wobei jeder besuchte Knoten das monton steigende Potenzial der jeweiligen Breitensuchenebene erhält [1]:

Algorithm 1 Wavefront-Algorithmus

Initialisierung:
    Jeder Punkt $U(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = 0$
    Warteschlange $Q := \{((\mathtt{x}_{Ziel}, \mathtt{y}_{Ziel}, \mathtt{rotation}_{Ziel}), 2)\}$

while $Q \neq \emptyset$ **do**
    $((\mathtt{x}, \mathtt{y}, \mathtt{rotation}), \mathtt{potential}) \leftarrow Q$
    $U(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = \mathtt{potential}$
    Nachbarn $N := \{(\mathtt{x-1}, \mathtt{y}, \mathtt{rotation}), (\mathtt{x+1}, \mathtt{y}, \mathtt{rotation}), ...(\mathtt{x}, \mathtt{y}, \mathtt{(rotation - 1) \% rotations})\}$
    for $(\mathtt{x}_{Nachbar}, \mathtt{y}_{Nachbar}, \mathtt{rotation}_{Nachbar}) \leftarrow N$ **do**
        if $0 \leq \mathtt{x}_{Nachbar} < \mathtt{occupancy\_grid\_width}$
         and $0 \leq \mathtt{y}_{Nachbar} < \mathtt{occupancy\_grid\_height}$
         and $\mathtt{computational\_space}[\mathtt{rotation}_{Nachbar}][\mathtt{y}_{Nachbar}][\mathtt{x}_{Nachbar}] = \mathtt{True}$ **then**
            $((\mathtt{x}_{Nachbar}, \mathtt{y}_{Nachbar}, \mathtt{rotation}_{Nachbar}), \mathtt{potential + 1}) \rightarrow Q$
        end if
    end for
end while

Somit breitet sich ausgehend vom Zielpunkt als Quelle mit jeder Iteration das monoton steigende Potenzial bildlich als "Wellenfrontïm Konfigurationsraum aus. Koordinaten in Hindernisnähe erhalten auf entgegengesetzter Seite der Ausbreitungsrichtung höhere Potenziale.

*** TODO: Abbildung mit ein paar Iterationen ***

6 Roboternavigation im Kraftfeld

Die Roboternavigation der Potenzialfeldmethode entspricht der Bewegung eines Körpers im *Gravitationsfeld* des Potenzialfelds.

6.1 Berechnung der Gradienten

Das Gravitationsfeld entspricht dem Gradientenfeld des Potenzialfelds: Auf jede Koordinate im Gravitationsfeld wirkt Gravitationskraft zerlegt in die drei kartesische Komponenten des Konfigurationsraums. Jedes dieser drei Kraftfelder $F_{x/y/rotation}(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$ gibt für die jeweilige Dimension des Konfigurationsraums die Größe der Gravitationskraft in der Koordinate (`x`, `y`, `rotation`) an:

$$F_x(\mathtt{x}, \mathtt{y}, \mathtt{rotation}), F_y(\mathtt{x}, \mathtt{y}, \mathtt{rotation}), F_{rotation}(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = -\nabla U(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$$

Um die Gradienten zwischen dem letzten Rotationsschritt und $0°$ zu berechnen, wird die erste und letzte Rotationsebene an das jeweils andere Ende der Dimension `potential[rotation]` kopiert.

```
first_layer = potential[0]
last_layer = potential[rotations - 1]
potential_padded = np.concatenate([last_layer, potential, first_layer], axis=0)
```

Die Gradienten des Potenzialfelds `potential` werden anschließend mit `np.gradients()` berechnet:

```
gradient_rotation, gradient_y, gradient_x = np.gradients(potential_padded)
```

Die Kraftfelder entsprechen somit den Gradienten werden ohne die kopierten Rotationsebenen:

```
force_field_rotation = −gradient_rotation[1:rotation-1]
force_field_x = −gradient_x[1:rotation-1]
force_field_y = −gradient_y[1:rotation-1]
```

Somit entsprechen Gradienten `force_field_rotation[rotation]` > 0 einer Kraft in Richtung `force_field_rotation[0]` und Gradienten `force_field_rotation[0]` < 0 einer Kraft in Richtung `force_field_rotation[rotation]`.

*** Abbildung mit Kraftvektoren in Konfigurationsraum ***

6.1.1 Gradienten an Grenzen und Hindernissen

Für Koordinaten an den X- und Y-Grenzen des Konfigurationsraums werden Kräften in Grenzrichtung in der jeweiligen Dimension auf 0 gesetzt:

- `force_field_x[0]` < 0
- `force_field_x[occupancy_grid_width - 1]` > 0
- `force_field_y[0]` > 0
- `force_field_y[occupancy_grid_length - 1]` > 0

Durch dieses manuelle Äbschneiden"der Kräfte – das Setzen der Kraft auf 0 – können an den Grenzen lokale Minima oder Plateaus entstehen: *** TODO: Abbildung ***

Im Potenzialfeld `potential` haben Hindernisse das Potenzial `np.nan`. Somit berechnet `np.gradients()` für jede an ein Hindernis grenzende Koordinate den Gradienten `np.nan`. Stattdessen sollen Hindernisse bei Berechnung der Gradienten wie Grenzen des Konfigurationsraums interpretiert werden. Ebenso wird hier die Kraft der jeweiligen Dimension auf 0 gesetzt, sollte negative Gradient in Richtung des Hindernisses zeigen. Beispiel für Hindernisse im Kraftfeld der X-Achse:

Kraftfeld Koordinate Hindernis: Gradient: Kraft = 0, wenn Kraft: force_field_x (x+1) => "rechts"pot.[z, y, x] - pot.[z, y, x-1] > 0 force_field_x (x-1) => "links"pot.[z, y, x+1] - pot.[z, y, x] < 0 force_field_x (x+1) und (x-1) 0 N/A => "rechts und links"

Wird für eine Koordinate die Kraft einer Dimension äbgeschnitten", wobei die Kräfte der anderen Kraftfelder ebenfalls 0 sind, entsteht ein lokales Minimum oder Plateau.

6.1.2 Behandlung lokaler Maxima

Im Unterschied zu einem lokalen Minimum oder Plateau ist bei einem lokalen Maximum das Potenzial der Nachbarn streng monoton niedriger als das der aktuellen Koordinate. In diesen Fällen wird zu einem der Nachbarn eine virtuelle Grenze gesetzt und der Gradient wie im vorherigen Abschnitt beschrieben berechnet.

*** Abbildung mit Beispiel ***

6.2 Gradientenabstieg

Die berechneten Gradientenfelder ermöglichen die Roboternavigation durch das *Gradientenabstiegsverfahren*:

Pro Verarbeitungsschritt wird für eine Koordinate $(\mathtt{x}, \mathtt{x}, \mathtt{rotation})$ die nächste Koordinate basierend auf der beträgsmäßig größten Gravitationskraft in $F_x(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$, $F_y(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$ und $F_{rotation}(\mathtt{x}, \mathtt{y}, \mathtt{rotation})$ gewählt, bis in einer Koordinate ein Kräftegleichgewicht $F_x(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = F_y(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = F_{rotation}(\mathtt{x}, \mathtt{y}, \mathtt{rotation}) = 0$ herrscht:

Algorithm 2 Gradientenabstiegsverfahren

Initialisierung:
  $(\mathtt{x}_{\text{current}}, \mathtt{y}_{\text{current}}, \mathtt{rotation}_{\text{current}}) = \mathtt{start\_point}$
  Visited $V \leftarrow \emptyset$
while true **do**
  if $(\mathtt{x}_{\text{current}}, \mathtt{y}_{\text{current}}, \mathtt{rotation}_{\text{current}}) = \mathtt{goal\_point}$ **then**
    return Ziel gefunden
  end if
  $\mathtt{force\_field\_x/y/rotation[rotation}_{\text{current}}\mathtt{][y}_{\text{current}}\mathtt{][x}_{\text{current}}\mathtt{]} = 0$
  $(\mathtt{dx}, \mathtt{dy}, \mathtt{drotation}) \leftarrow \max(|\mathtt{force\_field\_x/y/rotation[rotation}_{\text{current}}\mathtt{][y}_{\text{current}}\mathtt{][x}_{\text{current}}\mathtt{]}|)$
  if $(\mathtt{dx}, \mathtt{dy}, \mathtt{drotation}) = (0, 0, 0)$ **then**
    return Lokales Minimum oder Plateau
  end if
  $(\mathtt{x}_{\text{current}}, \mathtt{y}_{\text{current}}, \mathtt{rotation}_{\text{current}}) \leftarrow (\mathtt{x}_{\text{current}} + \mathtt{dx}, \mathtt{y}_{\text{current}} + \mathtt{dy}, \mathtt{rotation}_{\text{current}} + \mathtt{drotation})$
  if $(\mathtt{x}_{\text{current}}, \mathtt{y}_{\text{current}}, \mathtt{rotation}_{\text{current}}) \in V$ **then**
    return Lokales Minimum
  else
    $(\mathtt{x}_{\text{current}}, \mathtt{y}_{\text{current}}, \mathtt{rotation}_{\text{current}}) \rightarrow V$
  end if
end while

Der Gradientenabstieg muss dabei für die erlaubten Translationen des Konfigurationsraums diskretisiert werden: Gemäß der definierten Roboterbewegungen in Kapitel X (TODO!) darf der Roboter pro Verarbeitungsschritt nur eine Translation um eine Einheit entlang der Dimensionen `[rotation][y][x]` durchführen. Somit gilt:

$$(\mathtt{dx}, \mathtt{dy}, \mathtt{drotation}) \in \{(1,0,0)(-1,0,0), (0,1,0), (0,-1,0), (0,0,1), (0,0,-1)\}$$

7 Diskussion

7.1 Vergleich der Potenzialfeldmethoden

U-shaped obstacle problem [4]

Anziehendes/Abstoßendes Potenzial: - Nachteil : lokale Minima ... - Nicht-Lineare Gradienten

- Vorteil Wavefront Algorithmus: - Garantierte Konvergenz zum Ziel $=>$ Keine Lokalen Minima

"However, the workspace modeled by APF suffers from local minima, especially when mapping convex-shaped obstacles. Path generation by gradient search over an APF containing local minima will result in the robot getting stuck at some intermediate state before reaching the target location."[3]

... Lineare Gradienten

7.2 Skalierung des Occupancy Grids

Literatur

[1] Howie Choset. *Robotic Motion Planning: Potential Functions*. letzter Zugriff am 23.12.2023. CMU School of Computer Science: Robotics Institute 16-735. 2007. URL: https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf.

[2] O. Khatib. „Real-time obstacle avoidance for manipulators and mobile robots“. In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. Bd. 2. 1985, S. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[3] Ayesha Maqbool, Alina Mirza und Farkhanda Afzal. „Modified 2-Way Wavefront (M2W) Algorithm for Efficient Path Planning.“ In: *Int. J. Comput. Intell. Syst.* 14.1 (2021), S. 1066–1077.

[4] Zhang Yujiang und Li Huilin. „Research on mobile robot path planning based on improved artificial potential field“. In: *Mathematical Models in Engineering* 3.2 (2017), S. 135–144.