

# Roboternavigation mit Potenzialfeldern

**Intelligente Robotik WS2023/24**  
**Praktische Arbeit**

Carl Schünemann (Mat.Nr. 00107827)

26. Dezember 2023

# Inhaltsverzeichnis

<b>1 Ziel der Implementierung</b>	<b>1</b>
<b>2 Ausführung der Implementierung</b>	<b>2</b>
<b>3 Roboterbewegung im Occupancy Grid</b>	<b>3</b>
<b>4 Konfigurationsraum</b>	<b>4</b>
<b>5 Berechnung der Potenzialfelder</b>	<b>6</b>
5.1 Anziehende & Abstoßende Potenziale . . . . .	6
5.2 Wavefront Potenziale . . . . .	7
<b>6 Roboternavigation im Kraftfeld</b>	<b>9</b>
6.1 Berechnung der Gradienten . . . . .	9
6.1.1 Gradienten an Grenzen und Hindernissen . . . . .	10
6.1.2 Behandlung lokaler Maxima . . . . .	10
6.2 Gradientenabstieg . . . . .	11
<b>7 Diskussion</b>	<b>12</b>
7.1 Lokale Minima und Oszillationen . . . . .	12
7.2 Überdeckungen des Robotermodells mit Hindernissen . . . . .	13
<b>Literatur</b>	<b>14</b>

# 1 Ziel der Implementierung

Gegebene Aufgabenstellung:

Es soll ein Planungssystem zur Mikronavigation implementiert werden. Die Planung erfolgt auf Basis eines statisch gegebenen, einfachen Occupancy Grids mit Hilfe der Potenzialfeldmethode. Der Roboter besitzt eine rechteckige Form variabler Größen. 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
- Robotmodell: - Rechteckiger Roboter mit Variabler Größe (width, length)
- Rotation des Roboters um Änkerpunktum 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 berücksichtigen: 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  $\geq 3.7$  - Python Abhängigkeiten: > matplotlib zur Visualisierung > numpy für Mathematische Berechnungen => Installation aller Abhängigkeiten: pip install -r requirements.txt
- Empfehlung zur Ausführung: Anacoda Environment

### 3 Roboterbewegung im Occupancy Grid

Die physikalische Umgebung des Roboters wird diskretisiert durch das *Occupancy Grid*, ein binärer, zweidimensionaler Raum aus der Vogelperspektive. Implementiert mit der Python Bibliothek *Numpy*, hat das Boolean Array `occupancy_grid[Y] [X]` die Länge `occupancy_grid_length` und Breite `occupancy_grid_width` und den Ursprung links oben. Jede Koordinate ist entweder durch ein Hindernis belegt (`occupancy_grid[Y] [X] == False`) oder frei von Hindernissen (`occupancy_grid[Y] [X] == True`).

Im Occupancy Grid gibt es drei Roboterpositionen: `start_point`, `current_position` und `goal_point`.

```
occupancy_grid =
[[T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T],
 [T, T, T]]
```

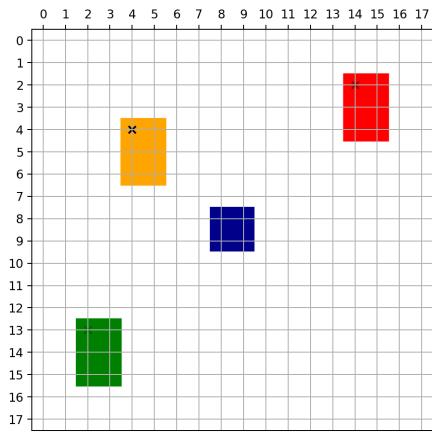


Abbildung 3.1: Ein einfaches Occupancy Grid mit vier Hindernissen, dem `start_point` bei ( $X=2$ ,  $Y=13$ ), der `current_position` bei ( $X=4$ ,  $Y=4$ ) und dem `goal_point` bei ( $X=14$ ,  $Y=2$ )

Die Dimension des Roboters wird durch die Variablen `robot_width` und `robot_length` definiert. Pro Verarbeitungseinheit kann sich der Roboter relativ zum *Ankerpunkt* der aktuellen Position 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 den Ankerpunkt. Bei einer Rotation von  $0^\circ$  liegt dieser Referenzpunkt in der linken oberen Koordinate innerhalb der Roboterdimensionen.

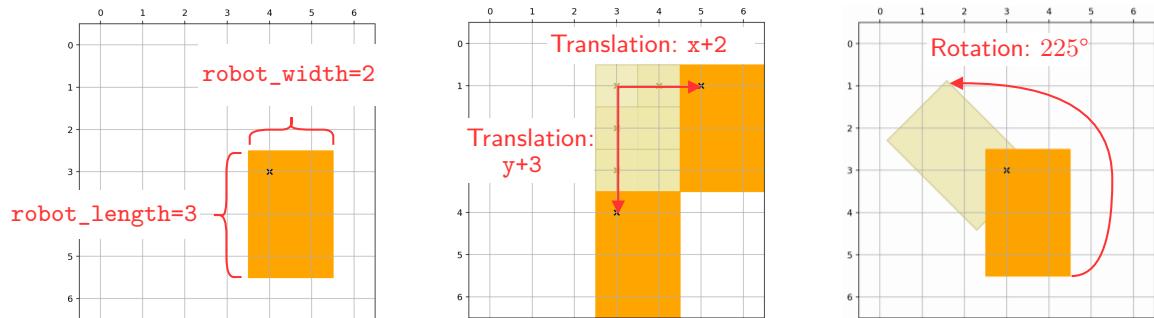


Abbildung 3.2: Bewegung eines Roboters mit variabler Dimension durch Translation und Rotation.

## 4 Konfigurationsraum

Bei einer Roboterlänge und -breite größer als eins können gewisse Punkte im Occupancy Grid nicht erreicht werden, ohne Hindernisse oder Grenzen zu überdecken. Aus diesem Grund wird die Roboterbewegung in eine Punktbewegung (`robot_width = robot_length = 1`) im sogenannten *Konfigurationsraum* transformiert. Dazu wird jedes Hindernis im Occupancy Grid um die Roboterdimensionen erweitert.

Je nach Rotation ändern sich die überdeckten Koordinaten im Occupancy Grid. Um den Verbrauch freier Koordinaten um das Hindernis zu minimieren, schlägt XYZ (TODO) vor, die Roboterrotation in diskrete Zustände zu unterteilen. Die Variable `rotation_step` gibt als Teiler von  $360^\circ$  an, um wie viel Grad sich der Roboter pro Bewegungsschritt drehen kann, wodurch sich `rotations=360/rotation_step` unterschiedliche Rotationszustände ergeben.

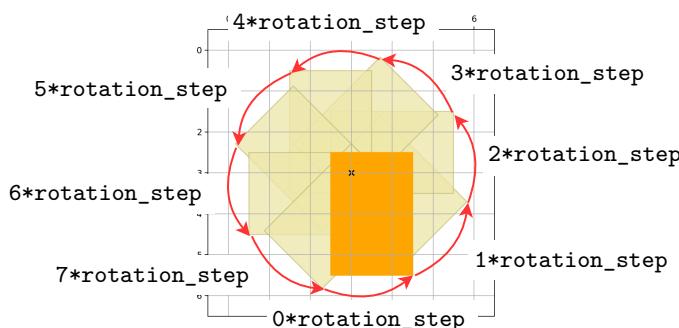


Abbildung 4.1: Für `rotation_step=45` ergeben sich  $(360^\circ \div 45^\circ) = 8$  Rotationszustände.

Für das RobotermodeLL wird eine `robot_length*robot_width` Maske erstellt. Für jeden Rotationszustand wird die Maske mit `sklearn.rotate()` gedreht sowie ein *erweitertes Occupancy Grids* generiert. Dazu wird jedes Hindernis sowie jede Grenzkoordinate des ursprünglichen Occupancy Grids um die rotierte Maske erweitert.

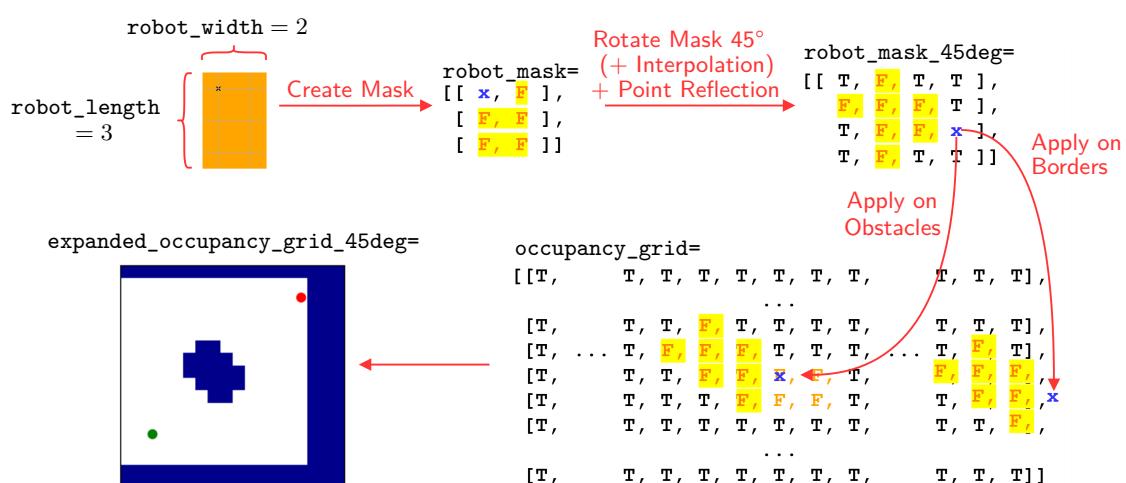


Abbildung 4.2: Erzeugung eines erweiterten Occupancy Grid für eine Rotation um  $45^\circ$

Der Konfigurationsraum `configuration-space[rotation] [y] [x]` mit den Dimensionen fasst die Erweiterten Occupancy Grids in der Dimension `[rotation]` zusammen:

- $(\text{rotation} + 1) \% \text{ rotations} \hat{=} \text{Rotation um rotation\_step gegen den Uhrzeigersinn}$
- $(\text{rotation} - 1) \% \text{ rotations} \hat{=} \text{Rotation um rotation\_step im Uhrzeigersinn}$

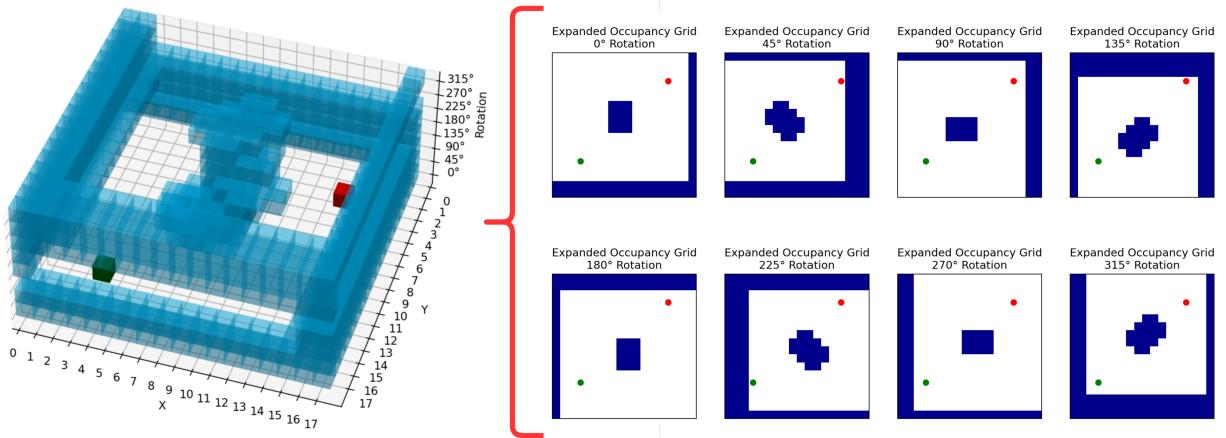


Abbildung 4.3: Die erweiterten Occupancy Grids bilden den Konfigurationsraum

Der Konfigurationsraum `rotations*occupancy_grid_length*occupancy_grid_width` nach XYZ (TODO!) ermöglicht über die Punktbewegung im dreidimensionalen Raum eine kollisionsfreie Translation und Rotation im Occupancy Grid bei minimalem Verbrauch freier Koordinaten. Die Position `x,y` des Roboters sowie dessen Rotationszustand `rotation` im Occupancy Grid wird dabei im Konfigurationsraum über die Dimensionen `rotation, y, x` referenziert.

# 5 Berechnung der Potenzialfelder

Die Grundlage der Pfadplanung für die Roboternavigation zu einem Zielpunkt bildet in dieser Implementierung die *Potenzialfeldmethode*: Der Konfigurationsraum entspricht einem skalaren *Potenzialfeld*, wobei jede Koordinate eine potenzielle Energie  $U(x, y, \text{rotation})$  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. Die Berechnung des Potenzialfelds  $\text{potential}[\text{rotation}][y][x] = U(x, y, \text{rotation})$  erfolgt mit *Potenzialfunktionen*. [4]

## 5.1 Anziehende & Abstoßende Potenziale

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

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

Bei den getesteten Implementierungsszenarien wurden bessere Ergebnisse festgestellt, wenn das anziehenden Potenzial erst normiert und anschließend gewichtet wird. Mit `attraction_weight = 1` hat das anziehende Potenzial einen Wertebereich von  $[0; 1]$ . Mit `attraction_weight < 1` wird die obere Grenze des Wertebereichs verringert, mit `attraction_weight > 1` vergrößert.

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

Die Hindernisse im Konfigurationsraum wirken auf jede Koordinate  $(x, y, \text{rotation})$  mit einem *abstoßendem* (engl. *repulsive*) Potenzial. Dazu zählen auch die Hindernisse aus der benachbarten Rotationsebene  $((\text{rotation}+1) \% \text{rotations})$  sowie  $((\text{rotation}-1) \% \text{rotations})$ . Mit `repulsion_weight = 0` hat das abstoßende Potenzial einen Wertebereich von  $[0; 1]$ . Werte `repulsion_weight > 0` verringern die obere Grenze gegen 0.

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

Yujiang und Huilin definieren das abstoßende Potenzial einer Koordinate als kleinsten Abstand zu allen Hindernissen [4].

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

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].

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

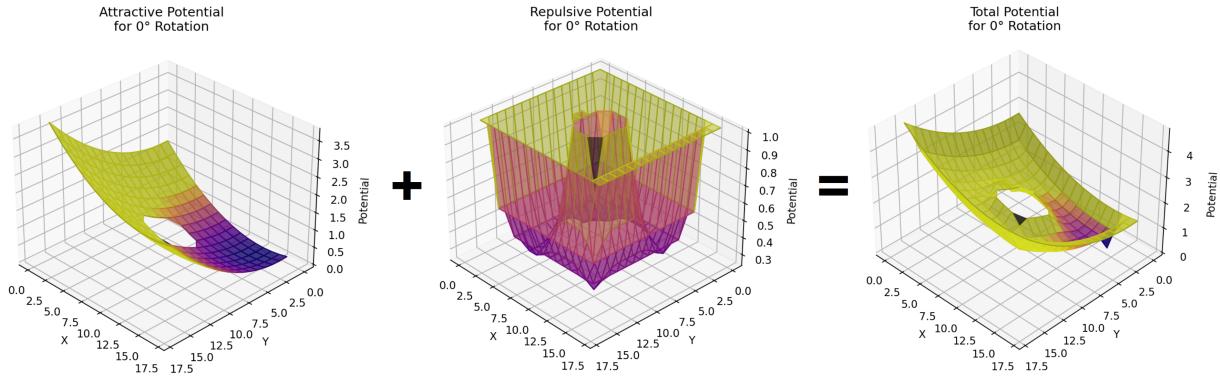


Abbildung 5.1: Die Berechnung des Gestamtpotenzial für die 0°-Rotationsebene des Konfigurationsraums mit `attraction_weight=5` und `repulsion_weight=0`.

## 5.2 Wavefront Potenziale

Choset stellt eine Potenzialfunktion unter Verwendung der Breitensuche ausgehend vom Zielpunkt vor. Beim sogenannten *Wavefront-Algorithmus* entsprechen die Koordinaten des Konfigurationsraums den Knoten der Breitensuche, wobei jeder besuchte Knoten das pro Breitensuchenebene streng monoton steigende Potenzial erhält: [1]

---

### Algorithm 1 Wavefront-Algorithmus

---

```

1: Initialisierung:
2:   Jeder Punkt  $U(\mathbf{x}, \mathbf{y}, \text{rotation}) = 0$ 
3:   Warteschlange  $Q := \{((\mathbf{x}_{Ziel}, \mathbf{y}_{Ziel}, \text{rotation}_{Ziel}), 2)\}$ 
4: while  $Q \neq \emptyset$  do
5:    $((\mathbf{x}, \mathbf{y}, \text{rotation}), \text{potential}) \leftarrow Q$ 
6:    $U(\mathbf{x}, \mathbf{y}, \text{rotation}) = \text{potential}$ 
7:   Nachbarn  $N := \{(\mathbf{x}-1, \mathbf{y}, \text{rotation}), (\mathbf{x}+1, \mathbf{y}, \text{rotation}), \dots, (\mathbf{x}, \mathbf{y}, (\text{rotation}-1) \% \text{rotations})\}$ 
8:   for  $(\mathbf{x}_{Nachbar}, \mathbf{y}_{Nachbar}, \text{rotation}_{Nachbar}) \leftarrow N$  do
9:     if  $0 \leq \mathbf{x}_{Nachbar} < \text{occupancy\_grid\_width}$ 
10:      and  $0 \leq \mathbf{y}_{Nachbar} < \text{occupancy\_grid\_height}$ 
11:      and  $\text{computational\_space}[\text{rotation}_{Nachbar}][\mathbf{y}_{Nachbar}][\mathbf{x}_{Nachbar}] = \text{True}$  then
12:         $((\mathbf{x}_{Nachbar}, \mathbf{y}_{Nachbar}, \text{rotation}_{Nachbar}), \text{potential} + 1) \rightarrow Q$ 
13:      end if
14:    end for
15:  end while

```

---

Somit breitet sich ausgehend vom Zielpunkt als Quelle das monoton steigende Potenzial bildlich als *Wellenfront* im Konfigurationsraum aus. Koordinaten in Hindernisnähe erhalten auf entgegengesetzter Seite der Ausbreitungsrichtung höhere Potenziale.

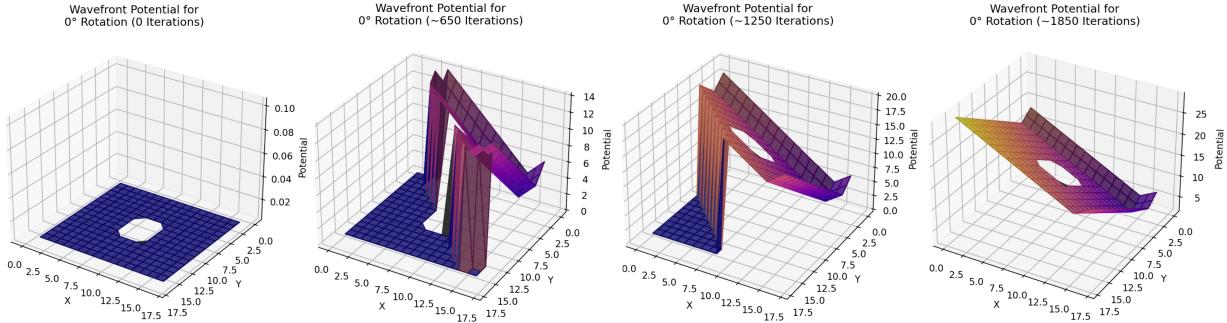


Abbildung 5.2: Die Berechnung des Gestamtpotenzial für die  $0^\circ$ -Rotationsebene des Konfigurationsraums mit `attraction_weight = 5` und `repulsion_weight = 0`.

Unabhängig von der gewählten Potenzialfunktion wird das Potenzialfeld im dreidimensionalen Konfigurationsraum für jede Rotationsebene berechnet. Somit hat das Potenzialfeld die gleichen Dimensionen wie der Konfigurationsraum:

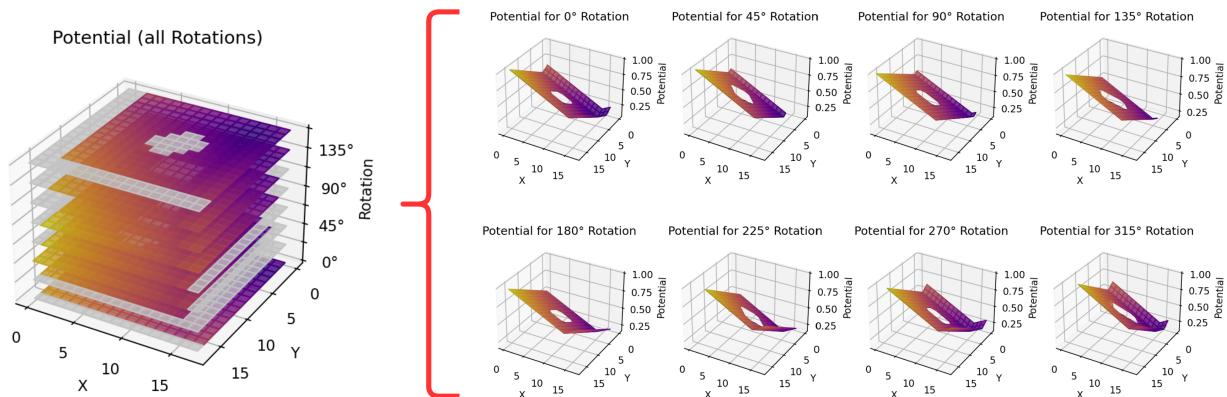


Abbildung 5.3: Das Potenzialfeld wird im dreidimensionalen Konfigurationsraum berechnet.

# 6 Roboternavigation im Kraftfeld

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

## 6.1 Berechnung der Gradienten

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

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

Um die Gradienten zwischen dem letzten Rotationszustand und  $0^\circ$  zu berechnen, wird die erste und letzte Rotationsebene an das jeweils andere Ende der Dimension potential[rotation] kopiert:

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

Die kopierten Rotationsebenen werden nach Berechnung der Kraftfelder entfernt:

```
force_field_rotation, force_field_y, force_field_x = -np.gradients(potential_padded)[1:rotation-1]
```

Somit entspricht  $force\_field\_rotation[rotation] > 0$  einer Kraft in Richtung  $force\_field\_rotation[0]$  und  $force\_field\_rotation[0] < 0$  einer Kraft in Richtung  $force\_field\_rotation[rotation]$ .

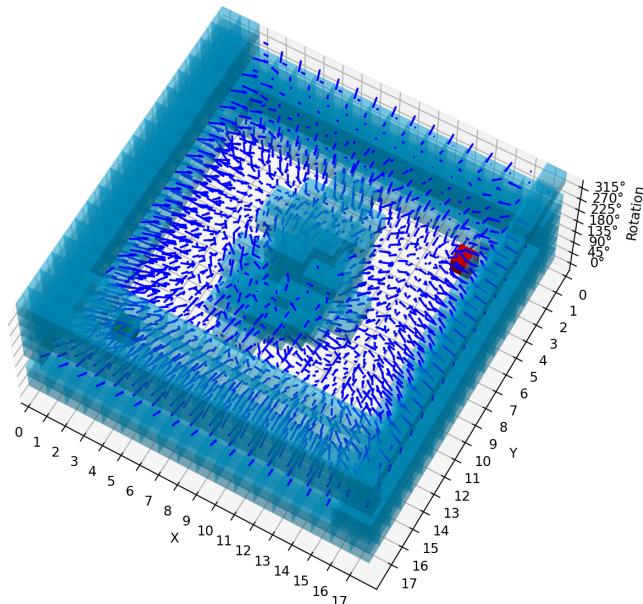


Abbildung 6.1: Darstellung des negativen Gradientenfelds als Kraftpfeile im Konfigurationsraum.

### 6.1.1 Gradienten an Grenzen und Hindernissen

Um die Grenzen des Konfigurationsraums einzuhalten, werden unzulässige Kräfte an den X- und Y-Grenzen auf 0 gesetzt. Durch dieses manuelle Abschneiden der Kräfte können an den Grenzen lokale Minima entstehen.

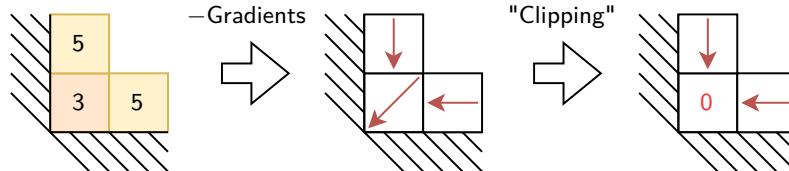


Abbildung 6.2: Das Setzen der Gradienten in Grenznähe auf 0 kann zu lokalen Minima führen.

Im Numpy Potenzialfeld `potential` haben Hindernisse das Potenzial `np.nan`. Somit berechnet `np.gradients()` für jede an ein Hindernis grenzende Koordinate den Gradienten `np.nan`. Um die Gravitationskräfte an Hindernissen analog zu Kräften an Grenzen des Konfigurationsraums zu interpretieren, müssen diese manuell berechnet werden. Sollte die berechnete Kraft einer Dimension in Richtung des Hindernisses zeigen, wird diese ebenfalls auf 0 gesetzt. Nachfolgende Tabelle zeigt dies beispielhaft für Hindernisse im Kraftfeld der X-Achse `force_field_x`:

Koordinate Hindernis	<code>force_field_x[rotation][y][x] = -1*...</code>	= 0, wenn ...
$x + 1$ (rechts)	<code>potential[z, y, x] - potential[z, y, x-1]</code>	$> 0$
$x - 1$ (links)	<code>potential[z, y, x+1] - potential[z, y, x]</code>	$< 0$
$x + 1$ und $x - 1$ (rechts und links)	0	N/A

Wird die Kraft einer Dimension auf 0 gesetzt, wobei die Kräfte der anderen Kraftfelder an dieser Koordinate 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 beiden Nachbarn ein künstliches Hindernis gesetzt und der Gradient neu berechnet.

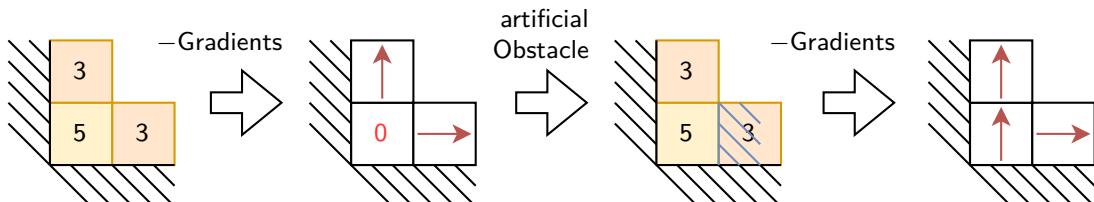


Abbildung 6.3: Lokale Maxima können durch künstliche Hindernisse behoben werden.

## 6.2 Gradientenabstieg

Die berechneten Gradientenfelder ermöglichen die Roboternavigation mit dem *Gradientenabstiegsverfahren*. Pro Verarbeitungsschritt wird die nächste Koordinate basierend auf der betragsmäßig größten Gravitationskraft in  $F_x(x, y, \text{rotation})$ ,  $F_y(x, y, \text{rotation})$  und  $F_{\text{rotation}}(x, y, \text{rotation})$  gewählt, bis eine bereits besuchte Koordinate oder eine Koordinate mit Kräftegleichgewicht  $F_x(x, y, \text{rotation}) = F_y(x, y, \text{rotation}) = F_{\text{rotation}}(x, y, \text{rotation}) = 0$  erreicht wird:

---

### Algorithm 2 Gradientenabstiegsverfahren

---

```

1: Initialisierung:
2:   ( $x_{\text{current}}, y_{\text{current}}, \text{rotation}_{\text{current}}$ ) = start_point
3:   Visited  $V \leftarrow \emptyset$ 

4: while true do
5:   if ( $x_{\text{current}}, y_{\text{current}}, \text{rotation}_{\text{current}}$ ) = goal_point then
6:     return Ziel gefunden
7:   end if
8:   force_field_x/y/rotation[rotationcurrent] [ycurrent] [xcurrent] = 0
9:   (dx, dy, drotation)  $\leftarrow \max(|\text{force\_field}_x/y/\text{rotation}[rotation_{\text{current}}] [y_{\text{current}}] [x_{\text{current}}]|)$ 
10:  if (dx, dy, drotation) = (0, 0, 0) then
11:    return Lokales Minimum oder Plateau
12:  end if
13:  ( $x_{\text{current}}, y_{\text{current}}, \text{rotation}_{\text{current}}$ )  $\leftarrow (x_{\text{current}} + dx, y_{\text{current}} + dy, rotation_{\text{current}} + drotation)$ 
14:  if ( $x_{\text{current}}, y_{\text{current}}, \text{rotation}_{\text{current}}$ )  $\in V$  then
15:    return Lokales Minimum
16:  else
17:    ( $x_{\text{current}}, y_{\text{current}}, \text{rotation}_{\text{current}}$ )  $\rightarrow V$ 
18:  end if
19: end while

```

---

Gemäß der in Kapitel X (TODO!) definierten Roboterbewegungen darf pro Verarbeitungsschritt nur eine Translation um eine Einheit entlang der Dimensionen des Konfigurationsraums durchgeführt werden:  $(dx, dy, drotation) \in \{(1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0), (0, 0, 1), (0, 0, -1)\}$

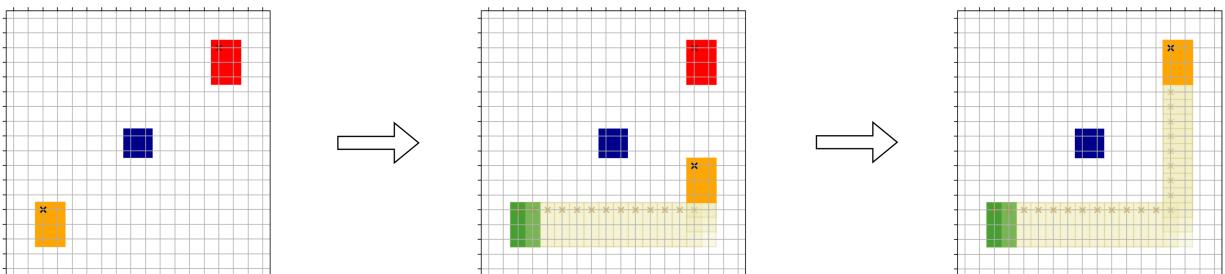


Abbildung 6.4: Roboternavigation über das Gradientenabstiegsverfahren im Kraftfeld des Wavefront-Potenzialfelds.

# 7 Diskussion

## 7.1 Lokale Minima und Oszillationen

Die Potenzialfunktion mit anziehendem und abstoßendem Potenzial hat die Eigenschaft zur Konvergenz des Gradientenabstiegsverfahrens zu einem lokalen Minimum bei konkaven Hindernissen. Davon betroffen sind Occupancy Grids, deren Hindernisse Polygone mit überstumpfen Winkeln ( $180^\circ \leq \text{Winkel} \leq 360^\circ$ ) bilden. Bekannte Beispiele der Literatur sind C- und U-förmige Polygone [3, 4]:

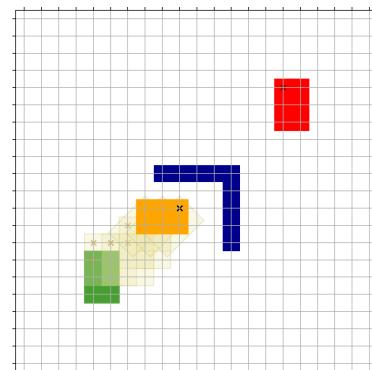


Abbildung 7.1: Für konkave Hindernisse in Potenzialfeldern des anziehenden und abstoßenden Potenzials konvergiert das Gradientenabstiegsverfahren zu einem lokalen Minimum.

Weiterhin sind im Gravitationsfeld von Potenzialfeldern mit anziehendem und abstoßendem Potenzial zwischen Hindernisengstellen Oszillationen möglich. Der Roboter wechselt bei dieser Art des lokalen Minimums zwischen zwei Koordinaten mit entgegengerichteten Gradienten. Insbesondere kleine Occupancy Grids sind von diesem Effekt betroffen, da die X- und Y-Grenzen als Hindernisse interpretiert werden. Abbildung 7.1 zeigt diesen Effekt anhand des Potenzialfelds aus Kapitel X (TODO). Die Erhöhung des Abstandes zwischen den Grenzen des Occupancy-Grid und Hindernissen vermeidet dieses Problem.

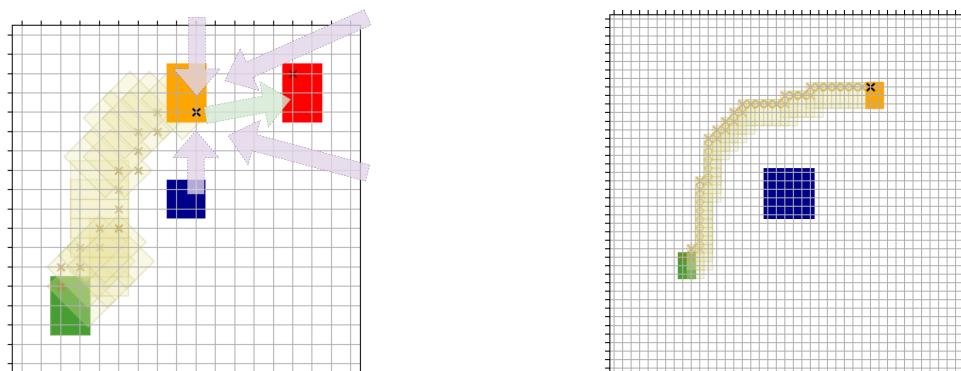


Abbildung 7.2: Bei Engstellen heben sich die abstoßenden Kräfte zwischen Hindernissen auf (links), bei hinreichend Abstand nicht (rechts).

Im Gegensatz dazu verhindert die streng monotone Potenzialänderung des Wavefront-Algorithmus die Konvergenz zu lokalen Minima, sofern sich das Ziel an einem für den Roboter physikalisch erreichbaren Punkt befindet.

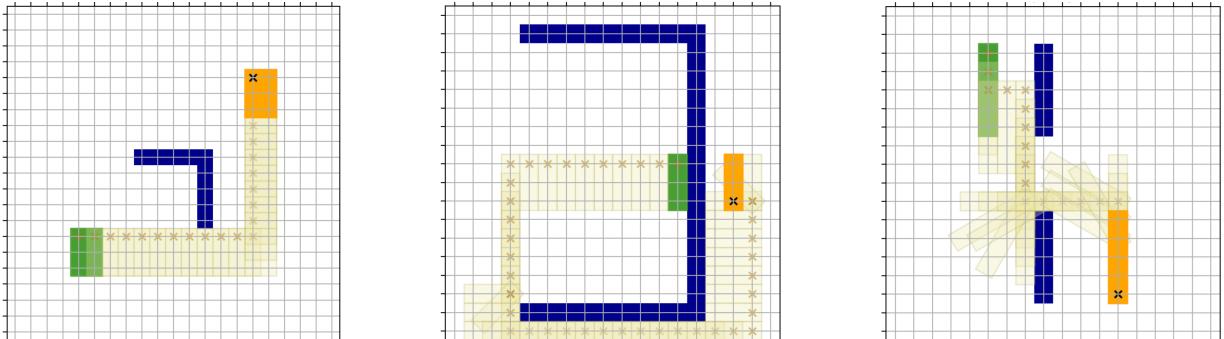


Abbildung 7.3: Das Gradientenabstiegsverfahren in Kraftfeldern der Wavefront-Potenzialfelder konvergiert zum Zielpunkt.

## 7.2 Überdeckungen des RobotermodeLLS mit Hindernissen

Wie in Kapitel 4 beschrieben, wird bei der Berechnung des Konfigurationsraums die Roboterrotation in `rotation_steps` diskretisiert. Dazu wird für das RobotermodeLL eine Array-Maske über die Matrixrotation `sklearn.rotate()` gedreht.

Insbesondere bei kleinen Roboterdimensionen mit kleinen Rotationswinkeln entstehen hier Interpolationsartefakte. Dies zeigt sich bei der Visualisierung der Roboternavigation durch die Überdeckung von Hindernissen mit dem RobotermodeLL sichtbar. In Literatur (Autor Wavefront TODO!) deshalb empfohlen, für kleine Rotationswinkel die Roboterdimensionen sowie das Occupancy Grid zu skalieren. Somit wird die Auflösung der Robotermaske erhöht und die Artefakte der Interpolation verringert.

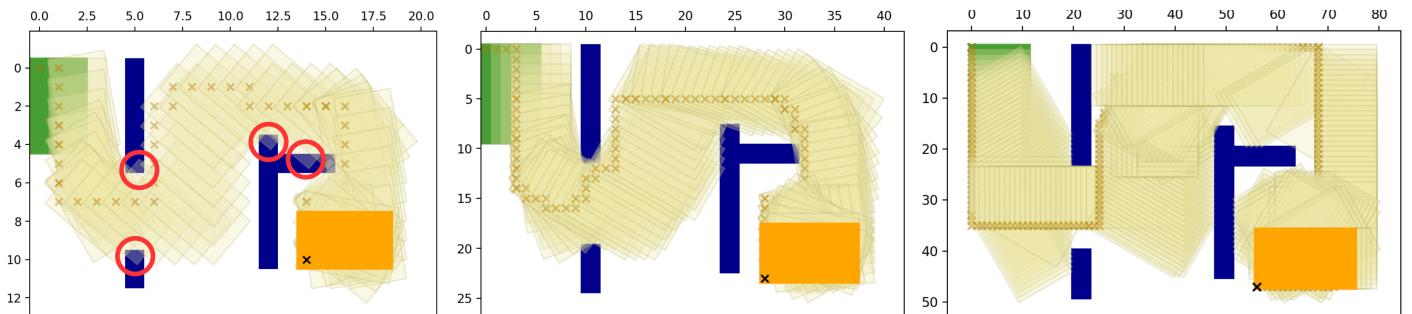


Abbildung 7.4: Die sukzessive Verdopplung der Dimensionen des RobotermodeLLs und Occupancy Grids verringert die Interpolationsartefakte.

# 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](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.