Inhaltsverzeichnis

Kapitel 1

Stochastische Modelle in der Robotik

Jede elaborierte Methoden erfordert einen Rahmen, in dem sie erarbeitet, formuliert und optimiert werden kann. In technischen Aufgabenstellung erfüllt die Mathematik diese Forderunge, weshalb die Gegebenheiten und Probleme des hiesigen Anwendungsfall zunächst in einem mathematischen Kontext dargestellt werden. Denn nur von dieser Basis ausgehend, besteht die Aussicht auf eine elegante und ansprechende Lösung.

Wie alle Probleme der Robotik, kann auch die autonome Navigation im entferntesten Sinne als Interaktion eines Roboters mit seiner Umwelt aufgefasst werden. Anhand von gesammelten Informationen muss das System eine Entscheidung über seine zukünftigen Aktionen treffen, wobei ein entferntes Ziel ohne ungewollte Kollisionen angesteuert werden soll. Für diese Aufgaben spielen drei Größen eine fundamentale Rolle. Zunächst muss die Position des Roboters beachtet werden, welche in dem Positionsvektor $\vec{\mathbf{x}}(t) \equiv \vec{\mathbf{x}}_t$ erfasst wird. Mithilfe von Sensoren sammelt der Roboter Informationen über seine Umgebung, die in dem Messvektor $\vec{\mathbf{z}}_t$ zusammengefasst werden. Anhand der Mess- und Positionsvektoren wird über die nächste Aktion des Roboters entschieden, die in dem Steuervektor $\vec{\mathbf{u}}_t$ ausgedrückt wird. Die exakte Form der Positions-, Mess- und Steuervektoren hängt von dem gegebenen Anwendungsfall und gewählten Modellformen ab, die im Folgenden näher erläutert werden.

In anderen Fachgebieten, die sich mit der Planung von Steuersignalen bzw. -sequenzen beschäftigen - wie z.B. die Regelungstechnik -, haben sich modellbasierte Methoden bewährt. Zunächst wird auf Basis von physikalischen Gegebenheiten der Zusammenhang zwischen Steuer-, Zustands- und Messvektor hergeleitet, der anschließend genutzt wird, um eine Regelstrategie zu formulieren. Der resultierende Algorithmus berechnet die Stellgröße $\vec{\mathbf{u}}_t$, wofür die aktuellen Mess- und Zustandsvektoren herangezogen werden. Insofern liegt es nahe modellbasierte Ansätze auch bei Problemen der Robotik zu verfolgen. Allerdings kommt dort die ungemeine Komplexität der Problemstellung zu tragen, die sich recht leicht am Beispiel der Navigation illustrieren lässt: Als erstes muss ein Modell für den Einfluss des Stellvektors $\vec{\mathbf{u}}_t$ auf den Positionsvektor $\vec{\mathbf{x}}_t$ erstellt werden. Bei mobilen Roboterplattformen handelt es sich um mechanische Systeme mit mehreren Freiheitsgraden, womit die analytische Modellbildung zwar möglich, jedoch mit einem beachtlichen Aufwand verbunden ist. Spätestens bei der Modelierung der Sensoren werden die Grenzen des Möglichen erreicht: Soll beispielsweise die Position des Roboters mithilfe von Stereokameras erfasst werden kann praktisch kaum ein exaktes, deterministisches Modell für diesen Vorgang erfasst werden, da er von zu vielen unbekannten Einflussfaktoren betroffen ist. Zuletzt kann die Pfadplanung per Definition nicht anhand eines deterministischen Modell erfolgen, da der Roboter dynamischen Hindernissen ausweichen soll, deren Form, Bewegung in der Aufgabenstellung nicht näher spezifiziert werden.

Aus diesen Gründen haben sich in der Robotik stochastische Modellformen etabliert, wobei recht simple Ausgangsmodelle verwendet werden, die um Zufallsvariablen ergänzt werden, um die Ungenauigkeiten und Ungewissheiten des Modells zu repräsentieren. Das Ziel besteht nicht mehr darin konkrete Aussagen über den Verlauf von Zustandsgrößen zu treffen, wie dies z.B. bei einer Zustandsraumdarstellung der Form

$$\vec{\mathbf{x}}(n+1) = \underline{\mathbf{A}} \cdot \vec{\mathbf{x}}(n) + \underline{\mathbf{B}} \cdot \vec{\mathbf{u}}(n) \tag{1.1}$$

erfolgt. Vielmehr soll mithilfe des Modells eine bedingte Wahrscheinlichkeit

$$p(\vec{\mathbf{x}}(n+1) \mid \vec{\mathbf{x}}(n), \vec{\mathbf{u}}(n)) \tag{1.2}$$

berechnet werden. Im Anschluss können die Methoden der Wahrscheinlichkeitstheorie genutzt werden, um Filter- und Planungsalgorithmen zu entwerfen. Um einen ersten Eindruck für die-

se Modellformen zu erhalten, werden im Anschluss rudimentäre Ansätze für ein Bewegungs- und Sensormodell vorgestellt.

1.1 Geschwindigkeitsbasiertes Bewegungsmodell

[1] Als erstes Beispiel wird ein stochastisches Modell für die Roboterbewegung entworfen, wobei anhand des vergangenen Positionsvektor $\vec{\mathbf{x}}_{t-1}$ und des aktuellen Steuervektors $\vec{\mathbf{u}}_t$ die Wahrscheinlichkeitsverteilung des aktuellen Positionsvektors $\vec{\mathbf{x}}_t$

$$p(\vec{\mathbf{x}}_t \mid \vec{\mathbf{u}}_t, \vec{\mathbf{x}}_{t-1}) \tag{1.3}$$

bestimmt werden soll. In diesem Fall wird lediglich eine planare Bewegung betrachtet, das heißt der Roboter bewegt sich in der xy-Ebene. Der Positionsvektor setzt sich somit aus den drei Größen

$$\vec{\mathbf{x}} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{1.4}$$

zusammen, welche die x-/y-Position und Ausrichtung des Roboters wiedergeben. θ gibt dabei den Winkel zwischen der x-Koordinatenachse und der Blickrichtung des Roboters an. Der Steuervektor $\vec{\mathbf{u}}$ gibt die aktuelle Translations- und Rotationsgeschwindigkeit

$$\vec{\mathbf{u}} = \begin{pmatrix} v \\ \omega \end{pmatrix} \tag{1.5}$$

des Roboters an, wobei angenommen wird, dass die beiden Geschwindigkeiten zwischen zwei Abtastpunkten t und $t+1$ konstant sind. v beschreibt die Translationsgeschwindigkeit in Blickrichtung, während ω die Änderung des Blickwinkels θ wiedergibt. Unter der Annahme dass die Geschwindigkeiten $\vec{\mathbf{u}}$ in dem Intervall $]t-1, t]$ zwischen zwei Abtastpunkten konstant bleibt, kann die Bewegung als Rotation um einen konstanten Momentanpol $\vec{\mathbf{c}} = \begin{pmatrix} x_c & y_c \end{pmatrix}^T$ betrachtet werden.

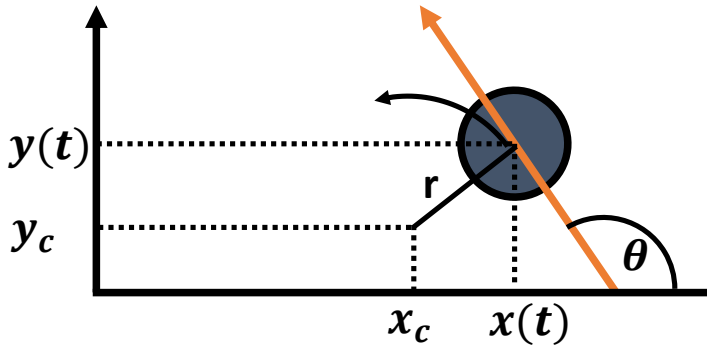

Abbildung 1.1: Darstellung des Momentanpols am Zeitpunkt t [1, S. 126]

Für den Radius gilt

$$r = \left| \frac{v}{\omega} \right|, \tag{1.6}$$

[1] Das Bewegungsmodell und dessen Herleitung stammen aus [1, S. 121 ff]

wobei zu beachten ist, dass der Radius für eine Winkelgeschwindigkeit $\omega = 0$ gegen unendlich konvergiert, was wiederum einer reinen Translation entspricht. Aus dem Positionsvektors des Roboters $\vec{\mathbf{x}}_t$ an dem Zeitpunkt t kann der Momentanpol für die folgende Abtastperiode berechnet werden:

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} x(t) - \frac{v}{\omega} \cdot \sin[\theta] \\ y(t) + \frac{v}{\omega} \cdot \cos[\theta] \end{pmatrix} \quad \leftrightarrow \quad \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{w} \cdot \sin[\theta] \\ y_c - \frac{v}{w} \cdot \cos[\theta] \end{pmatrix}. \tag{1.7}$$

Im nächsten Schritt wird die Bewegung über eine Abtastperiode Δt betrachtet, wodurch der Roboter um die Winkeldifferenz $\Delta t \cdot \omega$ auf dem Kreisbogen wandert. Nach 1.7 folgt für den Positionsvektor am Zeitpunkt $t + \Delta t$

$$\begin{aligned} \begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t \end{pmatrix} &= \begin{pmatrix} x_c + \frac{v}{w} \cdot \sin[\theta(t) + \omega \cdot \Delta t] \\ y_c - \frac{v}{w} \cdot \cos[\theta(t) + \omega \cdot \Delta t] \\ \theta(t) + \omega \cdot \Delta t \end{pmatrix} \\ &= \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} + \begin{pmatrix} -\frac{v}{\omega} \cdot \sin[\theta(t)] + \frac{v}{\omega} \cdot \sin[\theta(t) + \omega \cdot \Delta t] \\ \frac{v}{\omega} \cdot \cos[\theta(t)] - \frac{v}{w} \cdot \cos[\theta(t) + \omega \cdot \Delta t] \\ \omega \cdot \Delta t \end{pmatrix}. \end{aligned} \tag{1.8}$$

Bisher wurden lediglich deterministische Bewegungen betrachtet. Um nun mögliche Fehler des Steuervektors $\vec{\mathbf{u}}$ zu beachten, werden die störbehafteten Geschwindigkeiten

$$\vec{\mathbf{u}} = \begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \begin{pmatrix} v + v_{\text{err}} \\ \omega + \omega_{\text{err}} \end{pmatrix} \tag{1.9}$$

eingeführt. Die Zufallsvariablen v_{err} und ω_{err} dienen der Fehlermodellierung und ihre Wahrscheinlichkeitsverteilungen ε_v und ε_ω werden dem Anwendungsfall nach angepasst. Einsetzen des stochastischen Steuervektors $\vec{\mathbf{u}}$ liefert

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}} \cdot \sin[\theta(t)] + \frac{\hat{v}}{\hat{\omega}} \cdot \sin[\theta(t) + \hat{\omega} \cdot \Delta t] \\ \frac{\hat{v}}{\hat{\omega}} \cdot \cos[\theta(t)] - \frac{\hat{v}}{w} \cdot \cos[\theta(t) + \hat{\omega} \cdot \Delta t] \\ \hat{\omega} \cdot \Delta t \end{pmatrix}. \tag{1.10}$$

In diesem Modell wurden lediglich zwei Zufallsvariablen eingeführt, um die Störung von drei Positionsvariablen zu modellieren. Aus diesem Grund entsteht eine ungewollte stochastische Abhängigkeit zwischen den Elementen des Positionsvektors $\vec{\mathbf{x}}(t + \Delta t)$. Dieses Problem wird behoben, indem eine dritte Zufallsvariable

$$\gamma_{\text{err}} \equiv \hat{\gamma} \tag{1.11}$$

eingeführt wird, die zu einer zusätzlichen Störung der Orientierung $\theta(t + \Delta t)$ in Form von

$$\begin{pmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ \theta(t+\Delta t \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}} \cdot \sin[\theta(t)] + \frac{\hat{v}}{\hat{\omega}} \cdot \sin[\theta(t) + \hat{\omega} \cdot \Delta t] \\ \frac{\hat{v}}{\hat{\omega}} \cdot \cos[\theta(t)] - \frac{\hat{v}}{w} \cdot \cos[\theta(t) + \hat{\omega} \cdot \Delta t] \\ \hat{\omega} \cdot \Delta t + \hat{\gamma} \cdot \Delta t \end{pmatrix} \tag{1.12}$$

führt. Die Aufgabe besteht jetzt darin, ein Bewegungsmodell in Form der bedingten Wahr-

scheinlichkeit

$$p(\vec{\mathbf{x}}_t] \mid \vec{\mathbf{u}}_t, \vec{\mathbf{x}}_{t-1}) \tag{1.13}$$

zu formulieren. Das heißt es soll eine Funktion aufgestellt werden, die anhand der Vektoren $\vec{\mathbf{u}}_t$, $\vec{\mathbf{x}}_t$ und $\vec{\mathbf{x}}_{t-1}$ die Wahrscheinlichkeit dieses Ereignisses berechnet. Dafür wird nach Gleichung (1.12) die Werte der stochastisch unabhängigen Zufallsvariablen v_{err}, ω_{err} und γ_{err} berechnet. Die gesuchte Wahrscheinlichkeit ergibt sich dann aus deren gemeinsamer Verteilung

$$p(\vec{\mathbf{x}}_t, \vec{\mathbf{u}}_t, \vec{\mathbf{x}}_{t-1}) = p(\vec{\mathbf{x}}_t \mid \vec{\mathbf{u}}_t, \vec{\mathbf{x}}_{t-1}) = \varepsilon_v(v_{\text{err}}) \cdot \varepsilon_\omega(\omega_{\text{err}}) \cdot \varepsilon_\gamma(\gamma_{\text{err}})\,. \tag{1.14}$$

Da die direkte Umformung von Gleichung (1.7) zur expliziten Darstellung der gesuchten Fehlergrößen zu einem unhandlichen Ergebnis führt, wird eine indirekte Berechnung über den Momentanpol gewählt. Im ersten Schritt werden die Koordinaten des Momentanpols $\vec{\mathbf{c}}$ berechnet, wofür sich nach [1, S. 130] [2]

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = \begin{pmatrix} \frac{x+\tilde{x}}{2} + \mu(y-\tilde{y}) \\ \frac{y+\tilde{y}}{2} + \mu(\tilde{x}-x) \end{pmatrix} \qquad \mu = \frac{1}{2}\frac{(x-\tilde{x})\cos[\theta] + (y-\tilde{y})\sin[\theta]}{(y-\tilde{y})\cos[\theta] - (x-\tilde{x})\sin[\theta]} \tag{1.15}$$

ergibt. Woraus sich sowohl der Rotationsradius

$$r_{\text{c}} = \sqrt{(x-x_c)^2 + (y-y_c)^2} \tag{1.16}$$

als auch der auf der Kreisbahn zurückgelegte Winkel

$$\Delta\theta = \operatorname{atan}\left[\frac{\tilde{y}-y_c}{\tilde{x}-x_c}\right] - \operatorname{atan}\left[\frac{y-y_c}{x-x_c}\right] \tag{1.17}$$

berechnen lassen. Mithilfe der der Winkeldifferenz lässt sich wiederum auf die zurückgelegte Strecke

$$\Delta s = r_{\text{c}} \cdot \Delta\theta \tag{1.18}$$

schließend, welche zusammen auf den gestörten Stellvektor

$$\hat{\vec{u}} = \begin{pmatrix} \hat{v} \\ \hat{\omega} \end{pmatrix} = \frac{1}{\Delta t} \cdot \begin{pmatrix} \Delta s \\ \Delta\theta \end{pmatrix} \tag{1.19}$$

führen. Zuletzt fehlt der Orientierungsfehler $\hat{\gamma}$, der sich mithilfe von Gleichung (1.12) erschließen lässt.

$$\tilde{\theta} - \theta = \hat{\omega} \cdot \Delta t + \hat{\gamma} \cdot \Delta t \Leftrightarrow \hat{\gamma} = \frac{\tilde{\theta} - \theta}{\Delta t} - \hat{\omega}\,. \tag{1.20}$$

[2] Schreibweise für $\vec{\mathbf{x}}_t = \begin{pmatrix} x & y & \theta \end{pmatrix}^T$, $\vec{\mathbf{x}}_{t+\Delta t} = \begin{pmatrix} \tilde{x} & \tilde{y} & \tilde{\theta} \end{pmatrix}^T$

Kapitel 2

Kartenerstellung

2.1 Occupancy-Grid-Mapping

In dem hiesigen Anwendungsfall sollen metrische Karten verwendet werden, um das Navigationsproblem lösen zu können. Ein passender Kartentyp sind die so genannten Occupancy-Grids, welche die Umgebung als 2- oder 3-dimensionales, diskretes Raster darstellen. Jeder Zelle der Karte wird eine Wahrscheinlichkeit zugeordnet, die wiedergibt, ob die Zelle belegt ist. Durch den stochastischen Charakter des Umgebungsmodells können Ungewissheiten bei der Kartographierung und anschließenden Navigation beachtet werden.

Mathematisch wird die Umgebung als diskrete Menge von binären Zufallsvariablen

$$M = \{m_i\} \tag{2.1}$$

beschrieben, die entweder den Zustand frei oder belegt annehmen können. Die Aufgabe des Kartographierungsalgorithmus besteht darin, eine Karte zu erstellen, wobei es sich um die a posteriori Wahrscheinlichkeit

$$p(M = \text{belegt} \mid \vec{z}_{1:t}, \vec{x}_{1:t})^1 \tag{2.2}$$

handelt. Für jede Zelle soll die Wahrscheinlichkeit, dass diese unter Beachtung aller bisheriger Sensorwerte $\vec{z}_{1:t}$ und aller Zustandswerte $\vec{x}_{1:t}$ belegt ist, ermittelt werden. Um dieses Problem zu lösen, wird als erster Ansatz ein Bayes-Filter für binäre Zufallsvariablen angeführt. Es werden die beiden folgenden Annahmen getroffen: Die Umgebung ist stationär, das heißt keine Objekte werden während der Kartenaufzeichnung verschoben. Somit kann der Algorithmus nicht genutzt werden, um dynamischen Hindernissen auszuweichen. Zweitens wird angenommen, dass die Wahrscheinlichkeiten der einzelnen Zellen unabhängig sind.

$$p(M \mid \vec{z}_{1:t}, \vec{x}_{1:t}) = \prod_i p(m_i \mid \vec{z}_{1:t}, \vec{x}_{1:t})\,. \tag{2.3}$$

Dadurch können die a posteriori Wahrscheinlichkeiten der Zellen separat berechnet werden. Allerdings kann diese Annahmen unter realen Umständen kaum verteidigt werden, da Hindernisse für gewöhnlich größer als eine einzelne Kartenzelle sind.

[1] Im Folgenden wird die Wahrscheinlichkeit für M = belegt bzw. m_i = belegt als $p(M)$ bzw. $p(m_i)$ geschrieben. Die Wahrscheinlichkeiten für freie Zellen werden mit $p(\neg m_i)$ denotiert.

Für die Berechnung der Karte wird das logarithmische Verhältnis

$$l(x) = log\left[\frac{p(m_\mathrm{i})}{1-p(m_\mathrm{i}}\right] = log\left[\frac{p(m_\mathrm{i}}{p(\neg m_\mathrm{i}}\right], \tag{2.4}$$

die bei binären Zufallsvariablen rechentechnische Vorteile mit sich bringen [1, S. 94 f]. Bei der Kartenaufzeichnung soll das logarithmische Verhältnis $l(m_\mathrm{i})_t$ am Zeitpunkt t anhand des vergangen Wertes $l(m_\mathrm{i})_{t-1}$, des Messvektors $\vec{z}_t$ und des Zustandsvektors $\vec{x}_t$ für alle i Zellen der Karte bestimmt werden. Die gesuchte Wahrscheinlichkeit ergibt sich aus

$$p(x \mid =)1 - \frac{1}{1+e^{l(x)_t}}\,. \tag{2.5}$$

Für die Herleitung des Bayes-Filter wird zunächst der Fall betrachtet, dass die Zelle m_i belegt ist. Nach Bayes-Theorem folgt für die a posteriori Wahrscheinlichkeit

$$\begin{aligned} p(m_i \mid \vec{z}_{1:t}) &= p(m_i \mid \vec{z}_t, \vec{z}_{1:t-1}) \\ &= \frac{p(\vec{z}_t \mid m_i, \vec{z}_{1:t-1}) \cdot p(m_i \mid \vec{z}_{1:t-1})}{p(\vec{z}_t \mid \vec{z}_{1:t-1})} \\ &= \frac{p(\vec{z}_t \mid m_i) \cdot p(m_i \mid \vec{z}_{1:t-1})}{p(\vec{z}_t \mid \vec{z}_{1:t-1})}\,. \end{aligned} \tag{2.6}$$

Wie Wahrscheinlichkeit $p(\vec{z}_t \mid m_i)$ heißt Messmodell, da sie angibt, wie wahrscheinlich ein Messwert $\vec{z}_t$ aus einer gegebenen Umgebung m_i folgt. Aus Bayes-Theorem folgt wiederum

$$p(\vec{z}_t \mid m_i) = \frac{p(m_i \mid \vec{z}_t) \cdot p(\vec{z}_t}{p(m_i)} \tag{2.7}$$

und die anschließende Substitution in 2.6 liefert

$$p(m_i \mid \vec{z}_{1:t}) = \frac{p(m_i \mid \vec{z}_t) \cdot p(\vec{z}_t) \cdot p(m_i \mid \vec{z}_{1:t-1})}{p(m_i) \cdot p(\vec{z}_t \mid \vec{z}_{1:t-1})}\,. \tag{2.8}$$

Analog ergibt sich für das komplementäre Ereignis $\neg m_i$

$$p(\neg m_i \mid \vec{z}_{1:t}) = \frac{p(\neg m_i \mid \vec{z}_t) \cdot p(\vec{z}_t) \cdot p(\neg m_i \mid \vec{z}_{1:t-1})}{p(\neg m_i) \cdot p(\vec{z}_t \mid \vec{z}_{1:t-1})}\,. \tag{2.9}$$

Im nächsten Schritt folgt der Quotient der beiden Verteilungen

$$\begin{aligned} \frac{p(m_i \mid \vec{z}_{1:t})}{p(\neg m_i \mid \vec{z}_{1:t})} &= \frac{p(m_i \mid \vec{z}_t) \cdot p(\vec{z}_t) \cdot p(m_i \mid \vec{z}_{1:t-1}) \cdot p(\neg m_i) \cdot p(\vec{z}_t \mid \vec{z}_{1:t-1})}{p(\neg m_i \mid \vec{z}_t) \cdot p(\vec{z}_t) \cdot p(\neg m_i \mid \vec{z}_{1:t-1}) \cdot p(m_i) \cdot p(\vec{z}_t \mid \vec{z}_{1:t-1})} \\ &= \frac{p(m_i \mid \vec{z}_{1:t-1})}{p(\neg m_i \mid \vec{z}_{1:t-1})} \cdot \frac{p(m_i \mid \vec{z}_t)}{p(\neg m_i \mid \vec{z}_{1:t-1})} \cdot \frac{p(\neg m_i)}{p(m_i)}\,, \end{aligned} \tag{2.10}$$

dessen Logarithmus das gesuchte Verhältnis liefert:

$$\begin{aligned} l(m_i)_t &\equiv \log\left[\frac{p(m_i \mid \vec{z}_{1:t})}{p(\neg m_i \mid \vec{z}_{1:t})}\right] \\ &= \underbrace{\log\left[\frac{p(m_i \mid \vec{z}_{1:t-1})}{p(\neg m_i \mid \vec{z}_{1:t-1})}\right]}_{=l(m_i)_{t-1}} - \underbrace{\log\left[\frac{p(\neg m_i)}{p(m_i)}\right]}_{=l(m_i)_{t=0}\equiv l_0} + \log\left[\frac{p(m_i \mid \vec{z}_t)}{p(\neg m_i \mid \vec{z}_{1:t-1})}\right]\,. \end{aligned} \tag{2.11}$$

Somit setzt sich die gesuchte a posteriori Wahrscheinlichkeit aus drei Summanden zusammen. Bei dem Ersten handelt es sich um die Wahrscheinlichkeit $l(m_i)_{t-1}$ am vorherigen Abtastpunkt. Der Zweite gibt das Verhältnis der a priori Wahrscheinlichkeiten zum Zeitpunkt $t = 0$ wider und der letzte Teil ergibt sich aus dem logarithmischen Verhältnis von $p(m_i \mid \vec{z}_t)$, wobei es sich um ein inverses Messmodell handelt. Die Wahrscheinlichkeit $p(m_i \mid \vec{z}_t)$ wird von dem Entwickler vorgegeben und basiert auf der Charakteristik der Sensorik. Als Beispiel dient an dieser Stelle ein rudimentäres Modell für einen Laserscanner. Der Sensor gibt einen Vektor von Distanzmessungen zurück, die jeweils einen Messwinkel φ relativ zu dem Roboter zuzuordnen sind. Somit liegt ein Messbereich in Form eines Kegels mit dem Öffnungswinkel $\Delta\varphi$ vor, der durch die gemessenen Distanzen begrenzt wird. Im inversen Messmodell werden nun

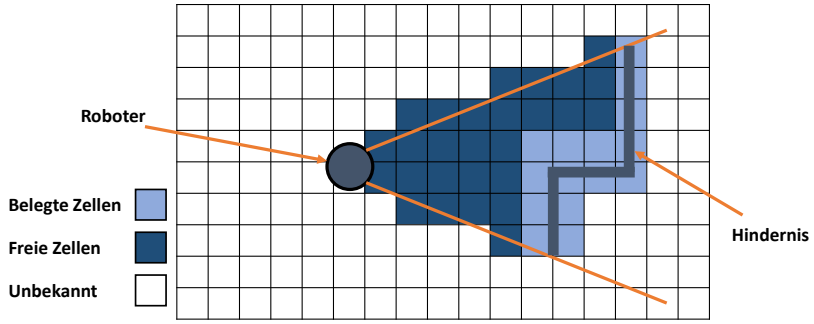

Abbildung 2.1: Schematische Darstellung der Karten und des inversen Messmodells

drei Fälle unterschieden. Liegt eine Zelle außerhalb des Messkegels, kann keine Aussage über den Zustand der Zelle getroffen werden, weshalb der Wert l_0 zurückgegeben wird der in 2.11 eingesetzt zu

$$l(m_i)_t = l(m_i)_{t-1} + l_0 - l_0 = l(m_i)_{t-1} \tag{2.12}$$

führt.In dem Fall, dass die Zelle innerhalb des Messkegels liegt, wird geprüft auf welchem Messstrahl und in welcher Distanz die Zelle sich befindet. Ist der Abstand zwischen der Zelle und der gemessenen Distanz kleiner als ein Toleranzmaß α, wird angenommen, dass die Zelle belegt ist und ein entsprechender Wert l_{occupied} zurückgegeben. Liegt die Zelle mehr als α Einheiten vor der gemessen Distanz, gilt sie als frei und das Modell liefert den Wert l_{free}.

Kapitel 3

ROS-Navigation-Stack

Bei dem ROS-Navigation-Stack handelt es sich um ein umfangreiches Software-Packet, das verschiedenste Funktionalitäten für das autonome Agieren eines Roboters bereitstellt. Von besonderem Interesse ist dabei das Paket *move_base*, welches bereits im Vorgängerprojekt für die Navigation des Roboters verwendet wurde. Funktional basiert das System auf einem zweischichtigen Planungskonzept: Zuerst berechnet ein globaler Planer anhand der vorgegebenen Karte einen optimalen Pfad zwischen Ausgangs- und Zielposition. Im Anschluss wird das Resultat an einen lokalen Planer übergeben, der die Stellgrößen des Roboters berechnet, um die geplante Trajektorie zu realisieren. Dabei werden neben dem optimalen Pfad ein lokaler Ausschnitt der Karte, Sensor- und Lokalisierungsdaten herangezogen, wodurch auch nicht vorhergesehene, dynamische Hindernisse bei der Planung beachtet werden können.

Im Gesamtkonzept spielen der lokale und globale Planer die entscheidenden Rollen, weshalb deren zu Grunde liegenden Algorithmen im Anschluss näher erläutert werden. Bei der Konfiguration ..

3.1 Funktionsprinzip des globalen Planers

Wie bereits erwähnt, wird die Navigationsaufgabe mithilfe zweier sukzessiver Planungsalgorithmen gelöst. Im ersten Schritt berechnet ein globaler Planer einen Weg zwischen der Ausgangs- und Zielposition, wofür die vorgegebene Karte herangezogen wird. Als Planungsalgorithmus werden im Navigation-Stack entweder Dijkstra- oder der A*-Algorithmus verwendet, die an dieser Stelle näher untersucht werden. Prinzipiell besteht die Aufgabe darin, einen Weg und Trajektorie zu ermitteln, die den Ausgangszustand des Roboters mit dem gewünschten Endzustand verbinden. Bei den gesuchten Pfaden handelt es sich um zeit- und ortskontinuierliche Kurven, woraus ein kontinuierlicher Raum von Lösungen resultiert, der nach einer optimalen durchsucht werden muss. Die Kontinuität des Suchraums stellt auf der Planungsseite eine immense Herausforderung dar, da eine unendliche Zahl von Möglichkeiten zur Verfügung stehen. Viel zu schwammige Erklärung, kleinen Einschub über Probleme wie differenzialgleichungen und Gütefunktionale... Um dieser Komplexität Herr zu werden wird der Suchraum diskretisiert und die Planungsaufgabe auf die Bestimmung einer Positionsfolge reduziert.

Allerdings wird die Diskretisierung des Raumes von mehr als einer reinen Vereinfachung des Suchproblems motiviert: Der globale Planer übernimmt lediglich den ersten Schritt bei der Berechnung der letztendlichen Trajektorie. Der lokale Planer berechnet anhand der globalen Sollkurve eine lokale Trajektorie, die in Form von Stellgrößen realisiert wird. Dabei werden neben der globalen Ortskurve auch aktuelle Sensor- und Lokalisierungsinformationen miteinbezogen, welche unter Umständen dazu führen können, dass der Roboter von der global geplanten Kurve abweicht. Insofern ergibt es wenig Sinn, bei der globalen Planung Ressourcen in eine unnötig genaue Trajektorie zu investieren; besonders dann, wenn noch nicht alle nötigen Informationen vorliegen. Für den Anfang genügt ein ungenauer, fehlerbehafteter Pfad, der die Anfangs- und Zielposition verbindet, wofür eine diskrete Darstellung des Suchraums vollkommen genügt. Des Weiteren stimmt die Forderung nach einer diskreten Darstellung des Suchraums mit den üblichen Darstellungsformen von metrischen Karten überein, denn diese repräsentieren die Umgebung in Form von diskreten Zellen.

Im Rahmen dieser Überlegung werden die vereinfachten Bedingungen zunächst in einer mathematischen Darstellung formuliert. Es wird angenommen, dass der Roboter in einer planen Umgebung manövriert, welche mittels einer zweidimensionalen Karte dargestellt werden kann. Hierfür wird eine metrische Karte verwendet, die den Raum in quadratische Zellen fixer Größe unterteilt. Die Positionen von Objekten innerhalb der Karten werden mithilfe des X- und Y-Indexes angegeben, wodurch die Positionsangaben von der Zellengröße entkoppelt werden.

Literaturverzeichnis

[1] Sebastian, Thrun; Wolfram, Burgard; Dieter Fox: „Probabilistic Robotics“, 1. Auflage, Massachusetts Institute of Technology 2006, MIT Press