Bis zum High-Score! Wie ein Roboter durch Ausprobieren lernt.

Storyboard

Eine der wichtigsten Aufgaben eines Roboters in der Automatisierung sind sogenannte *Pick and Place* Aufgaben. Dabei soll der Roboter ein Werkstück von einem Startpunkt zu einem Ziel bewegen und dort absetzen. Solche Aufgaben können relativ einfach gelöst werden, indem man dem Roboter die Bewegung mittels *teach-in* beibringt. Beim teach-in wird der Roboter händisch in wichtige Posen, wie z.B. die Aufhebe- und Absetzpose, gebracht und der Roboterzustand gespeichert. Bewegungen zwischen diesen Zuständen werden dann mittels klassischen Methoden der Robotik geplant und ebenfalls abgespeichert. So entsteht ein Bewegungsablauf, welcher abgespielt wird um die Pick and Place Aufgabe durchzuführen. Der große Nachteil dieses Vorgehens ist, dass die Bewegung nicht dynamisch abgeändert werden kann. Kommen Variationen in Aufnahme- und Ablegeposen oder in den auszuweichenden Hindernissen vor, muss das teach-in nochmals durchgeführt werden.

Soll der Roboter nun z.B. ein Werkstück von einer sich immer leicht ändernden Position aufheben, geraten wir an die Grenzen des teach-in. Hier ist es nicht wirtschaftlich für jede mögliche Position des Werkstückes ein neues teach-in durchzuführen. Die klassische Robotik bietet hier eine Lösung. Dabei beschreiben wir den Aufbau des Roboters mathematisch und ermitteln rechnerisch einen Pfad zwischen Start und Ziel. Auf Hindernisse im Arbeitsbereich Rücksicht zu nehmen verkompliziert jedoch so eine Implementierung. Deshalb stellen wir uns die Frage, ob wir mithilfe von AI den Roboter automatisch ein Verhaltensmuster zur Ausführung der Pick and Place Aufgabe lernen lassen können.

Reinforcement Learning

Reinforcement Learning (RL, siehe AIAV Video Reinforcement Learning) beschäftigt sich mit dem Lernen durch intelligentes Ausprobieren. Dabei beschreiben wir das zu lösende Problem als Interaktion zwischen einem Agenten, z.B. einem Roboter, und dessen Umfeld (Environment). Der Agent befindet sich in einem Zustand (State) im Umfeld und führt eine der möglichen Aktionen (Action) aus. Das Ausführen der Aktion verändert den Zustand des Umfeldes und der neue, aus dem vorherigen Zustand und der Aktion resultierende, Zustand (new State) wird zurückgegeben. Zusätzlich zum neuen Zustand gibt das Umfeld auch Feedback darüber, wie erfolgreich die durchgeführte Aktion im Erreichen eines Ziels war. Dieses Feedback kommt in Form einer Zahl, dem Reward. Der Reward funktioniert wie die Punkteanzahl in einem Spiel - je höher desto besser. Ein positiver Reward heißt also, dass die durchgeführte Aktion dem Agenten geholfen hat, sein Ziel zu erreichen. Analog dazu kennzeichnet ein negativer Reward das Scheitern des Agenten und ein Reward von 0 eine neutrale Aktion. Dieser Ablauf, der Markov Decision Process (MDP), ist in Abbildung 1 dargestellt.

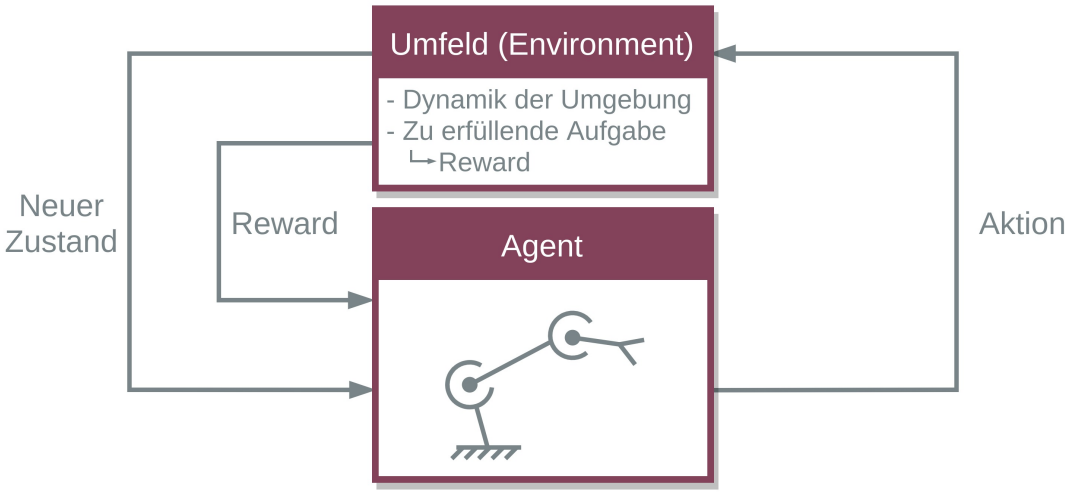

Abbildung 1: Aufgaben werden als Markov Decision Process (MDP) dargestellt, damit sie mittels Reinforcement Learning gelöst werden können. Dabei beschreibt der MDP eine Interaktion zwischen dem Lerner (Agenten) und seinem Umfeld (Environment).

Haben wir eine Aufgabe nun als MDP dargestellt, können wir uns überlegen wie der Agent überhaupt lernen soll. Wärhend viele verschiedene Algorithmen zum Lösen von Reinforcment Learning Problemen eingesetzt werden, haben diese alle das gleiche Ziel: den Reward mit der Zeit zu maximieren. Also soll unser Agent abhängig vom aktuellen Zustand Aktionen wählen, welche den höchsten Reward bringen. Am Anfang des Trainings weiß der Agent allerdings noch nichts über das Umfeld und die zu erfüllende Aufgabe. Deshalb werden zunächst nur zufällige Aktionen durchgeführt, um das Umfeld zu erkunden. Mit der Zeit basieren dann die gewählten Aktionen immer weniger auf Zufall und immer mehr auf der gelernten dynamik der Umgebung.

Praktische Implementierung

Um zu zeigen, wie ein praktisches Problem mittels Reinforcement Learning gelöst werden kann, wurde für diesen Usecase eine Pick and Place Aufgabe realisiert. Dabei werden Werkstücke durch einen Roboter mit drei Freiheitsgraden von einem Tisch aufgehoben. Das Werkstück kann an irgendeiner Position am Tisch, in Reichweite des Roboters, liegen. Zusätzlich dazu, gibt es noch Hindernisse im Arbeitsbereich des Roboters, die wir bei der Pfadplanung berücksichtigen müssen. Um Kollisionen mit Hindernissen zu vermeiden, wollen wir Reinforcement Learning verwenden, um den Pfad zwischen Start- und Zielpose zu planen.

Aber wie formuliert man so ein vorliegendes Problem als Markov Decision Process?

Zunächst legen wir fest, dass der Zustand (State) aus den Stellungen der drei Gelenke des Roboters besteht. Als mögliche Aktionen werden das Vergrößern oder Verkleinern der Stellung jedes Gelenks angenommen. Das heißt, dass der Roboter mit jedem Schritt eine von 6 Aktionen ausführen kann: Gelenk 1 erhöhen, Gelenk 1 vermindern, Gelenk 2 erhöhen, und so weiter. Da Zustand und Aktionen festgelegt sind, fehlt uns noch der Reward. Mit diesem signalisieren wir einerseits Erfolg und Versagen des Agenten, andererseits können wir aber auch, abhängig von der Komplexität der Reward Funktion, die Aufgabe schwerer oder einfacher für den Agenten gestalten.

Der Agent bekommt Reward anhand einer, als Teil der Umgebung definierten, Rewardfunktion. Für die Implementierung des Usecases haben wir uns für eine sogenannte spärliche Rewardfunktion entschieden. Das heißt, dass immer ein Reward von 0 signalisiert wird, außer der Agent hat sein Ziel

erreicht oder ist gescheitert. Erreicht der Roboter also seine Zielpose, bekommt er einen Reward von 1, kollidiert er hingegen mit sich selbst oder Hindernissen, bekommt er einen Reward von -1 und ansonsten einen Reward von 0.

Um nun konkrete Bahnen zu planen, legen wir die Start- und Zielposen fest. Dabei ist die Startpose der anfängliche Zustand des Agenten, wärhend die Zielpose der einzige Zustand ist, der einen positiven Reward gibt. Sind Start und Ziel festgelegt, wenden wir Policy Iteration, ein Verfahren zum Lösen von Reinforcement Learning Problemen, an. Policy Iteration liefert uns als Lösung ein Verhaltensmuster (Policy). Dieses gibt für jeden Zustand an, welche der Aktionen in diesem ausgeführt werden müssen, damit positiver Reward generiert wird. Da positiver Reward nur vom Zielzustand kommt, entsteht durch die Aktionen eine Bahn zum Ziel. Abbildung 2 zeigt die Station sowie einen dieser geplanten Pfade. Dabei entspricht jeder Pfeil einem Eintrag des Verhaltensmusters und zeigt an, in welche Richtung sich der Greifer beim Ausführen der jeweiligen Aktion bewegt.

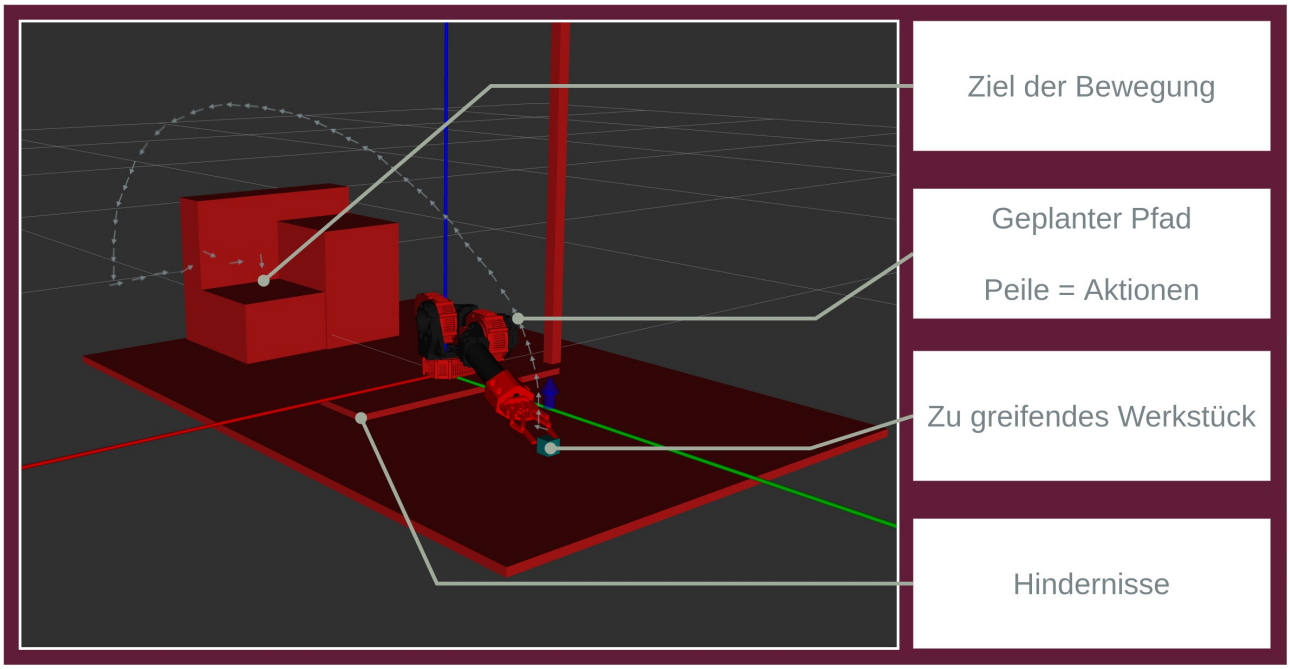

Abbildung 2: Durch Policy Iteration erhalten wir ein Verhaltensmuster, welchem der Roboter folgt um die Zielpose zu erreichen. Hier ist eines dieser Verhaltensmuster dargestellt. Dabei entspricht jeder Pfeil einem Zustand des Roboters, für den dieses Verhaltensmuster eine entsprechende Aktion vorgibt.

Computer Vision

Da wir nun Bewegungen automatisch mittels RL planen können, brauchen wir noch eine Möglichkeit, die Position des abgelegten Werkstücks zu bestimmen. Dafür montieren wir eine Kamera über dem Arbeitsbereich des Roboters und wenden klassische Bildverarbeitung an. Mittels dieser kann die Position des Werkstücks relativ zur Kamera bestimmt werden. Da die Position der Kamera vorab in der Station eingemessen wurde, haben wir einen Bezug zum Weltkoordinatensystem unserer Station und können die Position des Werkstücks durch Koordinatentransformation global nutzen.

Abbildung 3 zeigt die dabei angewendete Bildverarbeitungspipeline: Zunächst lesen wir das Bild ein und bestimmen einen Schwellenwert. Um das Werkstück vom hellen Hintergrund zu trennen, werden alle Bildpunkte über dem Schwellenwert auf weiß und alle Werte unter dem Schwellenwert auf schwarz gesetzt. Diesen Schritt nennt man Quantisierung. Anschließend erkennen wir Kanten im Bild, um die Kontur des Werkstücks zu ermitteln. Um die Kontur wird eine Box gelegt. Die Mitte dieser Box ist dann die Position des Werkstücks im Bild. Da diese Position noch in Pixeln angegeben ist, muss sie in Meter

umgerechnet werden. Diese Umrechnung basiert auf physikalischen Parametern der Kamera, wie die Größe des Sensors und der Brennweite der Linse. Diese Parameter wurden vorab durch Kalibrierung der Kamera ermittelt und abgespeichert.

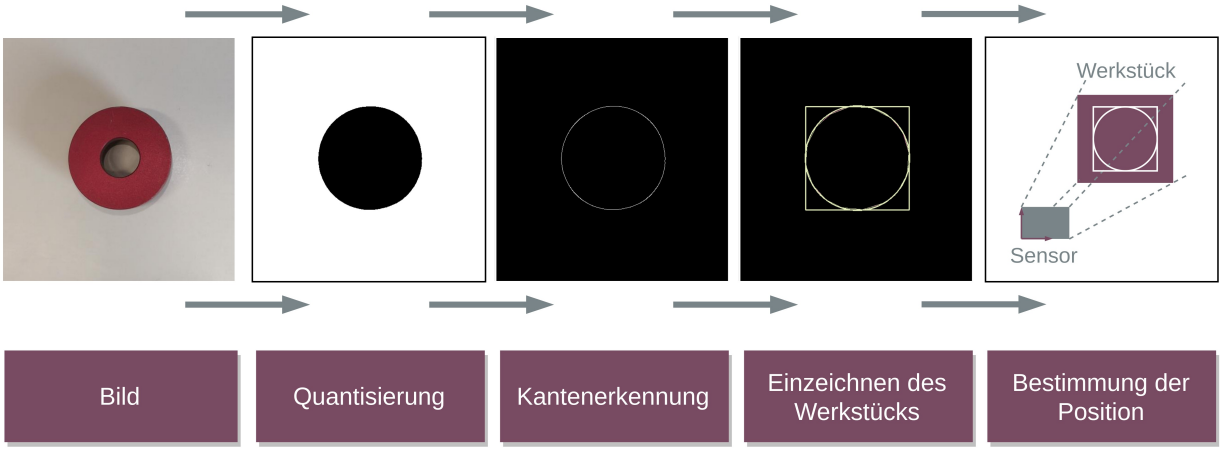

Abbildung 3: Um die Position des Werkstücks zu ermitteln, wenden wir klassische Bildverarbeitung an. Dabei quantisieren wir zunächst das Kamerabild und erkennen die Kanten. Anschließend ziehen wir eine Box rund um das Werkstück und berechnen aus dessen Position im Bild, die Position des Werkstücks in der echten Welt.

Fazit

Reinforcement Learning erlaubt es uns, Planungsprobleme vorab in Simulation zu lösen und die Lösung auf die echte Welt anzuwenden. Neben der Auswahl des Algorithmus, mit dem wir den Agenten implementieren und das Problem lösen, müssen wir uns auch darüber Gedanken machen, wie wir das Problem überhaupt formulieren. Festzulegen, wie Zustand, Aktionen und Reward aussehen, ist ein wesentlicher Schritt im Lösen eines Problems mittels Reinforcement Learning.

Die vorgestellte Lösung erlaubt es uns den Pfad automatisiert zu planen, sodass der Roboter die Pick and Place Aufgabe eigenständig lösen kann. Jedoch hat die Implementierung einige Grenzen. Dadurch, dass wir Policy Iteration zur Lösung des Reinforcement Learning Problems verwenden, muss unser Markov Decision Process komplett beobachtbar sein. Dafür müssen wir in der Simulation zunächst alle möglichen Zustände des Roboters abfahren und speichern. Das resultiert in einer Startzeit des Roboters von mehreren Minuten. Diese Limitierung könnte mit einer anderen Reinforcement Learning Methode, wie z.B. Q-Learning, umgangen werden.

Dadurch, dass zur Erkennung des Werkstücks klassische farbbasierte Bildverarbeitung eingesetzt wird, ist diese aber bei sich ändernden Lichtverhältnissen fehleranfällig. Hier könnte der Einsatz moderner Methoden zur Objekterkennung, wie z.B. ein Convolutional Neural Network (CNN, siehe AIAV Video CNN Detector) die Fehleranfälligkeit der Lösung verbessern.