

Projektbericht

Thema: Linienfolger mit Q-Learning

Fach: Künstliche Intelligenz für autonome Systeme (KIS)

Informatik Bachelor, Hochschule Mannheim

16.07.2018

Dennis-Immanuel Czogalla

Matrikel Nr.: 1410116

Inhaltsverzeichnis

Einleitung.....	3
Abstract.....	3
Projektbeschreibung.....	3
Exkurs Reinforcement Learning (Q-Learning).....	4
Das Projekt.....	5
Verwendete Hardware.....	5
Verwendete Software.....	5
Simulation.....	6
Umsetzung des Q-Learnings auf dem Roboter.....	7
Ausblick.....	8
Quellen.....	8

Einleitung

Abstract

Ziel dieses Projekts war es auf Basis eines Raspberry Pis einen Roboter zu konstruieren und programmieren, der mit Hilfe von Reinforcement Learning (Q-Learning) lernt einer Linie zu folgen. Zunächst wurde hierfür eine Simulation geschrieben, die die Problemstellung aus der Realen Welt stark vereinfacht repräsentiert. Anhand dieser Simulation konnten mögliche Probleme im Voraus identifiziert werden und der Algorithmus soweit optimiert werden, dass er relativ zügig zu einem guten Ergebnis kommt. Anschließend konnte der Code mit minimalen Änderungen auf den Roboter übertragen und erfolgreich ausgeführt werden.

Projektbeschreibung

Der Roboter besitzt ein simples Kettenlaufwerk, bestehend aus jeweils einem Laufrad und einem Antriebsrad pro Seite. Er verfügt über einen Lichtsensor, der an drei verschiedenen Positionen Werte von seiner Umgebung auslesen kann. Die Werte stellen verschiedene Helligkeitsstufen dar.

Das gesamte Fahrgestell, sowie die vier Räder wurden 3D gedruckt. Angetrieben wird er von einem Arduino Uno, zwei Servo Motoren und sechs AAA Batterien.

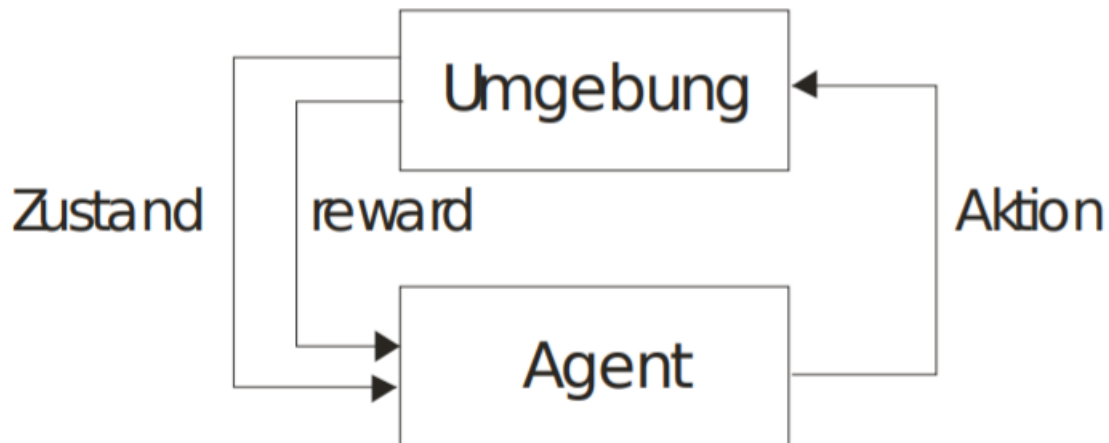
Ziel ist es, dass der Roboter selbständig lernt mit Hilfe des Lichtsensors und dem Einsatz von Q-Learning (eine Variante des Reinforcement Learning) einer schwarzen Linie zu folgen. Hierfür soll der Arduino durch einen Raspberry Pi ersetzt werden. Die Stromversorgung muss dementsprechend angepasst werden und besteht aus einer Powerbank für den Raspberry, sowie vier AAA Batterien für die Servo Motoren.

Der Lichtsensor ist für den Arduino vorgesehen und arbeitet mit 5V. Der Raspberry Pi arbeitet jedoch mit 3.3V und so muss der Pegel mit Hilfe von Widerständen angepasst werden.

Um das Programm auf dem Roboter bequem starten und stoppen zu können, wird ein Taster hinzugefügt, der bei Betätigung einen Befehl an den Raspberry sendet und das Programm startet bzw. stoppt.

Exkurs Reinforcement Learning (Q-Learning)

Reinforcement Learning beschreibt ein Vorgehen, bei dem ein Agent (der Roboter) mit seiner Umgebung interagiert und für jede seiner durchgeführten Aktionen eine entsprechende Belohnung bekommt. Diese Belohnungen werden im Vorfeld festgelegt und können positiv oder negativ sein. Die Umgebung wird als eine Menge von Zuständen dargestellt, über die der Agent iteriert. Für jeden Zustand stehen ihm alle Aktionen zur Auswahl. Gesucht werden dabei die Aktionen, die die Belohnung maximieren. Dieser Vorgang stellt das Lernen des Agenten dar.



Das Q-Learning ist eine Variante des Reinforcement Learnings.

Hierbei kommt eine Bewertungsfunktion zum Einsatz, die eine Aktion in Abhängigkeit mit dem Zustand, in den der Agent durch die Aktion gelangt ist, bewertet und als sogenannten Q-Values abspeichert. Die Bewertungsfunktion sieht folgendermaßen aus [1]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha * [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

a = Aktion

s = State

α = Lernrate (0..1)

r = Reward

γ = Discountfactor (Zustände, die weiter in der Zukunft liegen, werden geringer gewichtet)

Das Projekt

Verwendete Hardware

Raspberry PI 3+

Hamma Joy Powerbank 5200 mAh
(Stromversorgung RPI)

2x Servos AR3603HB

4x AAA Batterien (Stromversorgung Motoren)

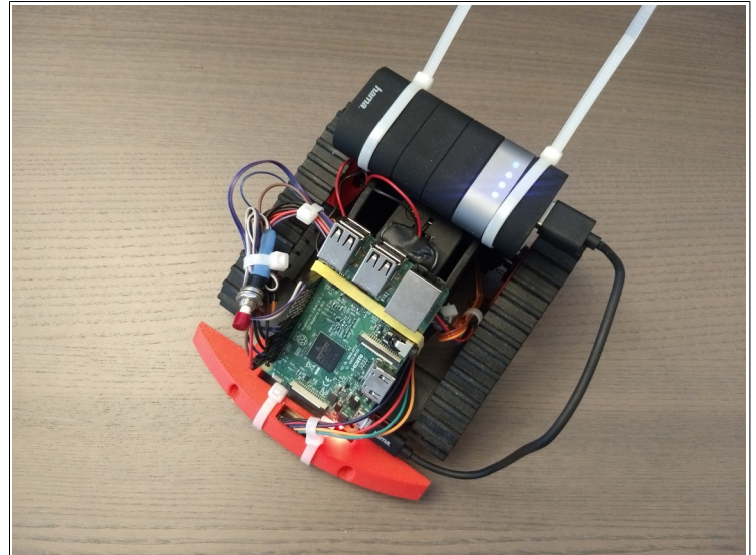
Lichtsensord (Sensorarray)

Fahrgestell 3D gedruckt

Widerstände um den Pegel auf 3.3V zu senken

Kabelbinder

1x Taster



Verwendete Software

Putty

<https://www.putty.org>

Raspbian Stretch Lite

<https://www.raspberrypi.org/downloads/raspbian/>

Pycharm Community Edition

<https://www.jetbrains.com/pycharm/download/#section=windows>

Tkinter Library für die Simulation

Pigpio zum Steuern der Servos

<http://abyz.me.uk/rpi/pigpio/>

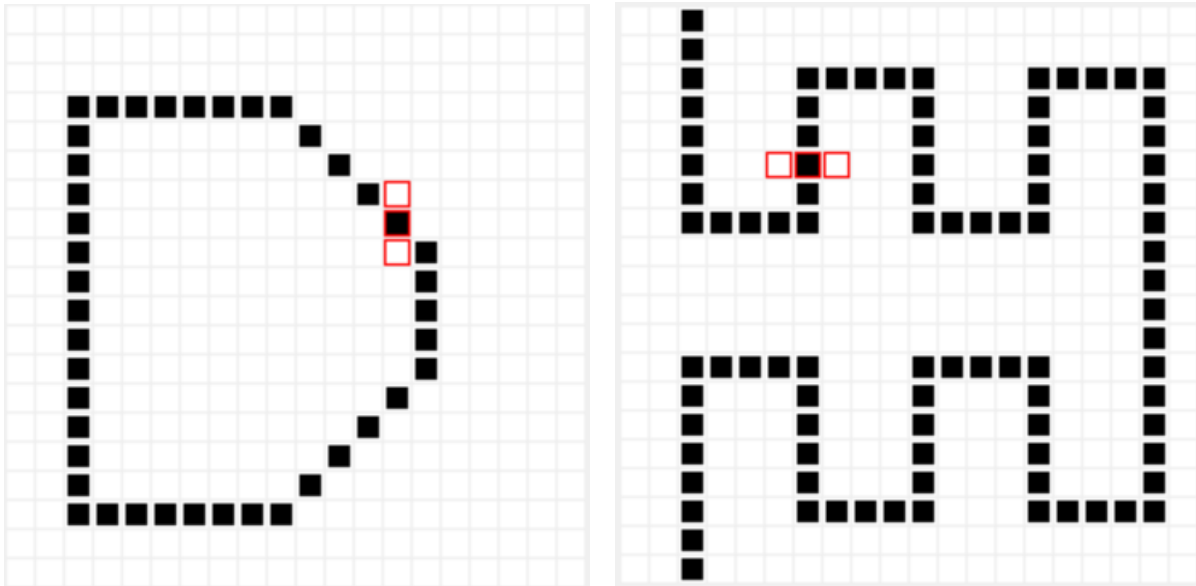
Python 3.x

Simulation

Die Simulation in Python geschrieben, bei der die Problemstellung stark vereinfacht wurde.

Das Spielfeld besteht aus 20*20 Kacheln, die entweder schwarz oder weiß sein können. Es können unterschiedliche Muster dargestellt werden. Hierfür wird mit Hilfe eines Grafikprogramms wie z.B. Photoshop ein Bild generiert (20x20 Pixel), das z.B. einen schwarzen Kreis enthält.

Das Programm liest das Bild ein, wandelt die Informationen in schwarz-weiß um und färbt dementsprechend die richtigen Stellen auf den Spielfeld ein.



Der Roboter bzw. Akteur wird in Form eines roten Rechtecks dargestellt, das drei Felder hintereinander abdeckt. Dies entspricht technisch auch dem Lichtsensor, der nochmals in drei Sensoren aufgeteilt ist, die jeweils einen Helligkeitswert lesen können.

Der Akteur hat drei verschiedene Bewegungsmöglichkeiten: Vorwärts, Rechtsdrehung, Linksdrehung.









Das Spielfeld ist unbegrenzt (ähnlich wie bei dem Spiel Snake), d.h. der Akteur befindet sich nach dem Verlassen von z.B. der linken Spielfeldgrenze, an der rechten Spielfeldgrenze. Gleiches für oben und unten.

Das Reinforcement Learning bzw. Q-Learning in der Simulation sieht folgendermaßen aus:

Der Sensor ist in drei Sektoren unterteilt, die jeweils weiß oder schwarz sein können und verfügt damit über $2^3 = 8$ mögliche Kombinationen.

Multipliziert man die Anzahl aller möglichen Kombinationen mit der Anzahl aller möglichen Bewegungsmöglichkeiten pro Sektor (Vorwärts, Links, Rechts), so erhält man die $2^3 * 3 = 27$ Zustände für das Q-Learning.

Die Bewertung dieser Zustände sieht folgendermaßen aus:

	- 1
	0
	0
	0
	0
	0
	0
	+ 1

D.h. die Kombination Weiß Weiß Weiß erhält für alle Bewegungsmöglichkeiten (Vorne, Links, Rechts) einen negativen Reward von -1. Und für die Kombination Weiß Schwarz Weiß sowie alle Bewegungsmöglichkeiten den positiven Reward +1.

Code: <https://github.com/PhonierDeluxe/RaspberryBot/blob/master/LightSensorRL.py>









Umsetzung des Q-Learnings auf dem Roboter

Zunächst musste herausgefunden werden, wie die Servo Motoren mit Hilfe des Raspberry Pis angesteuert werden. Hierfür wird die Bibliothek Pigpio verwendet, mit der es möglich ist die Geschwindigkeit der Motoren einzustellen. Es wurden die Aktionen festgelegt und anschließend ein erster Test durchgeführt. Der Code musste kaum abgeändert werden.

Nach den ersten Test stellt sich jedoch heraus, dass die drei festgelegten Aktionen nicht gut funktionieren, da das Spielfeld begrenzt ist und der Roboter dazu neigte, es recht zügig zu verlassen.

Es wurden diverse Möglichkeiten durchgetestet wie beispielsweise ein Suchalgorithmus, um wieder zum Spielfeld zurückzukehren, falls nach längerer Zeit kein positiver Reward mehr erfolgte.

Die beste Lösung hierfür war jedoch die Einführung einer zusätzlichen vierten Aktion (Rückwärts), die gleich zwei Verbesserungen mit sich brachte. Zum einen blieb der Roboter eher auf bzw. in der Nähe des Spielfelds und zum anderen war das erlernte Ergebnis deutlich präziser als mit nur drei Aktionsmöglichkeiten. Damit gibt es insgesamt $2^3 \cdot 4 = 32$ Zustände.

	- 5
	0
	+ 10
	0
	+ 10
	0
	0
	+ 50

Es wurden zwei zusätzliche Rewards hinzugefügt und die Bewertung der Rewards wurde geändert. Die zusätzlichen geringer gewichteten Rewards sollen dem Roboter helfen, den Zustand mit dem besten Reward zu erreichen. Dieser ist nicht so leicht zu erreichen, da der Roboter exakt auf der Linie ausgerichtet sein muss. Oft befindet er sich jedoch zu dieser angewinkelt und verpasst so diesen Zustand.

Code: https://github.com/PhonierDeluxe/RaspberryBot/blob/master/LFRL_newest.py

Ausblick

Der Raspberry Pi bietet die Möglichkeit, eine Kamera anzuschließen und anzusteuern. Der Roboter könnte mit Hilfe der Kamera die Linie erkennen und entsprechend Aktionen ausführen. Es würde erneut Reinforcement Learning zum Einsatz kommen. Für die Kamera bzw. das Erkennen der Linien mit Hilfe der Kamera wäre ein Neuronales Netz notwendig. Es gibt bereits Bibliotheken wie OpenCV, mit denen sich Objekte bzw. Linien erkennen lassen können [2]. Denkbar wäre es, die Library für die Erkennung der Linien zu verwenden und in Kombination mit dem Lichtsensor den Roboter mit Q-Learning darauf zu trainieren, einer Linie zu folgen. Der Lichtsensor spielt hierbei nur für das Training eine Rolle. Er dient dazu, die von der Kamera empfangenen Bilder mit einem Reward zu bewerten. Wenn sich der Roboter beispielsweise auf der Linie ausgerichtet ist und der Lichtsensor dies als den Zustand mit dem besten Reward erkennt, dann wird dem aktuellen Bild der Kamera dieser Reward zugewiesen.

Quellen

[1] http://services.informatik.hs-mannheim.de/~fischer/lectures/MLE_Files/MLE.pdf

[2] <https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/>