Entwurf, Simulation, Training & Aufbau eines Roboterarm-Demonstrators mittels Reinforcement Learning

Aaron Kiani, David Retinski, Pascal Graf, Nicolaj C. Stache

Mechatronik und Robotik, Heilbronn University of Applied Sciences

Übersicht

Aufgabe:

Nachbau einer Blockanordnung mit einem Roboterarm ohne explizite Programmierung. Dazu baut ein Anwender einen Turm auf, welcher mittels einer Kamera detektiert wird. Der Roboter baut den Turm in gleicher Anordnung an einer anderen Stelle nach.

Teilprobleme:

1. Erkennung der vorgemachten Blockanordnung
2. Bewegung zur Pick-Position
3. Aufheben des Blocks
4. Bewegung zur Place-Position
5. Rotieren des Blocks
6. Ablegen an richtige Stelle

Kamera-System

Aufbau:

Konstruktion einer Kamerahalterung.
Montage der Kamera in der Realen Umgebung
Kamera Model in die Simulation Umgebung integriert.

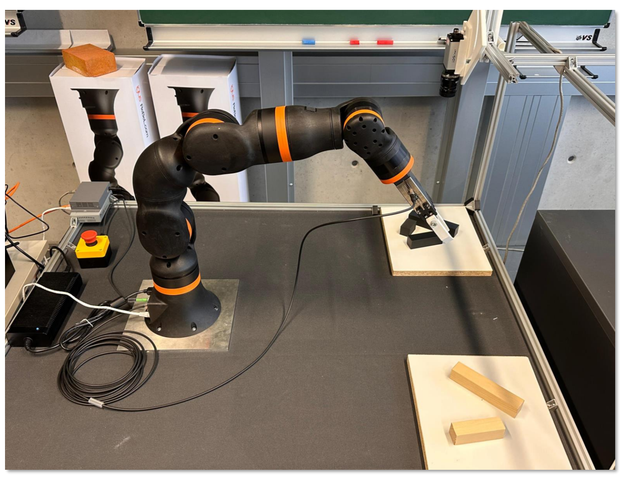
Objektdetektion:

1. Bild Aufnahme
2. Bild Vorverarbeitung
3. Kantendetektion [1]
4. Konturfindung
5. Berechnung der Orientierung & Position
6. Blockposition & Orientierung an den Roboter weiter geben

Roboter und virtueller Zwilling

Versuchsaufbau:

- Montage des 6-DoF Roboters
- Montage der Außenkonstruktion
- Kamerasystem
- 3D-Druck der Parallel-greifer

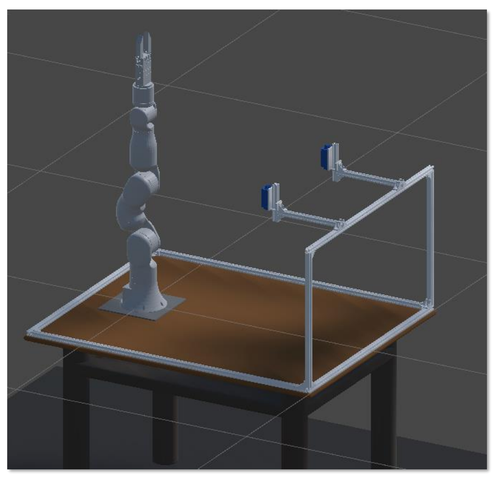


Simulationsumgebung:

- Robotermodell in Format der Physik Engine MuJoCo [2]
- Modellierung der Umgebung in Unity [3]
- RL-Framework ML-Agents [4] in Unity integrieren

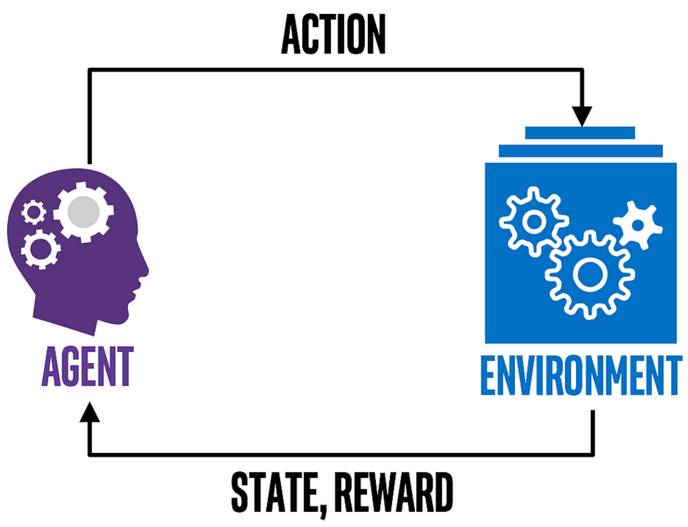
Unity

MuJoCo
Advanced physics simulation



Reinforcement Learning

1. Agent führt Aktion aus z.B. Bewegen des Gelenks
2. Auswirkung der Aktion auf die Umgebung wird bewertet z.B. Distanz des Roboters zum Ziel
3. Geben von Belohnung oder Strafe
4. Update des neuronalen Netzes und damit des Verhaltens des Agenten



Imitation Learning

- Anwender macht Aufgabe vor z.B. durch Steuerung mittels Controller (Inverse Kinematik)
- Agent „imitiert“ die Bewegungen, wodurch das Lernen beschleunigt wird

Herausforderungen und Auswertung

- Physikalisch realistisches Modell des Roboters in Unity (Reibung, Getriebeübersetzungen, Drehmomente)
- Training mittels verschiedener RL-Algorithmen wie Soft-Actor-Critic (SAC) [5] und Proximal Policy Optimization (PPO) [6]
- Unterteilung des Gesamtproblems in Teilschritte für das Training: griffbereit zum Block fahren, Block aufheben, zur Zielposition fahren, Block ablegen
- Erarbeitung und Implementierung von effizienten Reward-Funktionen z.B. abhängig von der Distanz zum Ziel, Ausrichtung des Greifers, Kollisionen, Dauer bis zur Erfüllung der Aufgabe
- Imitation Learning: Intuitive Steuerung eines Zielpunktes und entsprechende Ausrichtung des Roboters mittels Inverser Kinematik für den 6-DoF Roboter in Unity

Literatur

[1] Canny, J., A Computational Approach To Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

[2] MuJoCo — Advanced Physics Simulation. Verfügbar unter: https://mujoco.org/.

[3] Unity. Unity Echtzeit-Entwicklungsplattform | 3D-, 2D-, VR- und AR-Engine. In: Unity, 03. November 2023. Verfügbar unter: https://unity.com/de.

[4] Unity Technologies. Unity ML-Agents Toolkit. [Online]. 26. November 2023. Verfügbar unter: https://unity-technologies.github.io/ml-agents/.

[5] Schulman, John, Wolski, Filip, Dhariwal, Prafulla, Radford, Alec, und Klimov, Oleg. Proximal Policy Optimization Algorithms, 20. Juli 2017. Verfügbar unter: http://arxiv.org/pdf/1707.06347.pdf.

[6] Haarnoja, Tuomas, Zhou, Aurick, Abbeel, Pieter, und Levine, Sergey. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. 04. Januar 2018. Verfügbar unter: http://arxiv.org/pdf/1801.01290.pdf.

akiani@stud.hs-heilbronn.de, dretinski@stud.hs-heilbronn.de, Pascal.Graf@hs-heilbronn.de, Nicolaj.Stache@hs-heilbronn.de