-

Bearbeitungsbeginn: [01.03.2020]

Vorgelegt am: [31.08.2020]

Thesis

zur Erlangung des Grades

Bachelor of Science

im Studiengang

Medieninformatik an der

Fakultät Digitale Medien

Abd-Rahmen Al-Skaf

Matrikelnummer: 254853

Implementierung eines Unity-ML-Agents zur automatisierten Steuerung eines Endoskops bei der Extraktion von Fremdkörpern. Virtuell simuliert anhand dreidimensionaler Nachbildungen vom menschlichen Körper, eines Endoskops und Fremdkörpern.

Erstbetreuer: *Frau Prof. Dr. Lasowski*

Zweitbetreuer: *Herr Duda*

Abstract

Die Arbeit zeigt die Erstellung einer 3D-Simulation innerhalb der 3D-Echtzeit-Entwicklungsumgebung Unity3D und die Implementierung eines Unity-ML-Agents.
Hier ist das Ziel, den Unity-ML-Agent darauf zu trainieren, dass dieser ein 3D-Endoskop so steuert, dass ein 3D-Fremdkörper gegriffen wird. Hierzu werden 3D-Modelle eines menschlichen Körpers, eines Fremdkörpers und eines Endoskops genutzt, wobei sich der Fremdkörper und das Endoskop innerhalb des menschlichen Magens befinden. Das Endoskop wurde dabei als Unity-ML-Agent implementiert und so um seine Funktionen erweitert. Das Training des Unity-ML-Agents nutzt das Reinforcement Learning (deutsch : bestärkendes Lernen) und zunächst den Proximal Policy Optimization (PPO) Algorithmus‘, welcher später durch das Behavioral Cloning (BC)ergänzt wird. Außerdem werden verschiedenen Änderungen vorgenommen, die das Training beeinflussen, dazu gehören Änderungen an den Aktionen, Observationen, Belohnungen oder der Form des Fremdkörpers.

Aus den gelernten Verhaltensweisen der Unity-ML-Agents lassen sich verschieden Hürden und Konflikte ableiten. Die Hürden bei der Lösung der Aufgabe sind vor allem die Positionsbestimmung des Fremdkörpers und die nötigen Manöver bei der Ausrichtung des Fremdkörpers, vor dem Zugreifen.
Ein Konflikt entsteht zwischen den Belohnungen des Zugreifens und den Belohnungen der vorherigen Zwischenschritte, diese können ausreichen, um weitere Aktionen weniger lohnenswert zu machen, sodass ein Zugreifen nicht notwendig ist.
Die Ergebnisse und das Verhalten des Unity-ML-Agents bei der Lösung der Aufgabe zeigen, dass in zukünftigen Entwicklungen die Manöver und das Zugreifen getrennt von anderen Zwischenschritten trainiert werden sollten. Außerdem müssen die Belohnungen so angepasst werden, dass die Zwischenschritte und das Zugreifen so belohnt werden, dass kein Konflikt zwischen diesen beiden entsteht.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ML……Machine learning
PPO……..…Proximal Policy Optimization
BC……..……Behavioral Cloning
z.B……zum Beispiel
o……..oder
SAC.……..……..Soft-Actor-Critic

Quellcodeverzeichnis

1 Einleitung

Da immer mehr Bereiche der Medizin das Machine Learning (deutsch: maschinelles Lernen) nutzen und deren Vorteile entdecken, könnten eines Tages auch verschiedene Eingriffe am Menschen, von Robotern mit neuronalen Netzen durchgeführt werden. Einer dieser möglichen Eingriffe, ist die Endoskopie des Magens, zur Entfernung von Fremdkörpern. Mit dieser Arbeit soll überprüft werden, wie erfolgreich eine Endoskopie zur Fremdkörperentfernung ist, wenn sie von einem trainierten neuronalen Netz durchgeführt wird und welche Hürden während des Lernprozesses der Fremdkörperentfernung entstehen.

Da keine Tests am Menschen möglich sind und die Entwicklung aller Komponenten sehr aufwendig und kostspielig ist, wird der notwendige Sachverhalt virtuell in Unity3d simuliert. Die Simulation besteht dabei aus dem 3D-Modell eines menschlichen Körpers, einem funktionalen 3D-Nachbau eines Endoskops und einem 3D-Fremdkörper. Hierbei werden das 3D-Endoskop und der 3D-Fremdkörper innerhalb des Magens des 3D-Körpers platziert. Des Weiteren wird das Endoskop als Unity-ML-Agent implementiert und so mit Algorithmen des Reinforcement Learning trainiert und anschließend getestet.

Die genaue Aufgabe des Unity-ML-Agents ist es, den Fremdkörper mit dem Endoskop zu greifen. Dabei soll ein Zielverhalten gelernt werden, bei dem das Endoskop dem Fremdkörper genährt wird, um ihn anschließend zu greifen.

Bevor allerdings auf die Entwicklung eingegangen wird, werden hierauf folgend, zunächst die medizinischen Grundlagen vermittelt, die bei der Entwicklung in Betracht gezogen werden. Dazu gehören grundlegendes Wissen über den Gaster, die Endoskopie oder mögliche Fremdkörper. Anschließend folgen die technischen Methodiken, die im Rahmen der Bachelorarbeit genutzt wurden. Dazu gehören die Entwicklungsumgebung für 3D-Echtzeitanwendungen Unity3D oder auch die Programmiersprache Python, die von der tatsächlichen Realisierung gefolgt werden. Hier wird genauer erläutert wie die Simulation aufgebaut ist und das Training und die Tests durchgeführt werden. Danach folgen die Ergebnisse der Realisierung, welche die Trainingsdokumentation aufzeigen und das Ergebnisverhalten des Unity-ML-Agents beschreiben. Vor dem Schluss, werden anschließend die Simulation und die durchgeführten Versuche evaluiert.

1.1 Ziel der Bachelorarbeit

Das Ziel der Bachelorarbeit ist die Erstellung und Nutzung einer 3D-Simulation innerhalb Unity 3D und die Implementierung eines Unity-ML-Agents. Die Simulation soll hierbei einen menschlichen Körper beinhalten, in dessen Magen sich ein Endoskop und ein nicht-organischer Fremdkörper befinden.
Die Aufgabe des Unity-ML-Agents ist dabei das Endoskop so zu steuern, dass der Fremdkörper von dessen Zange gegriffen wird. Hierzu soll der Unity-ML-Agent ein Zielverhalten erlernen, bei dem er das Endoskop dem Fremdkörper annähert, falls notwendig ein Manöver ausführt, das den Fremdkörper ausrichtet und den Fremdkörper anschließend mit der Zange greift. Die Bachelorarbeit soll ebenfalls Aufschluss darüber liefern welche Hürden bei der Erlernung des Zielverhaltens und Lösung der Aufgabe existieren.

2 Medizinische Grundlagen

Im folgenden Kapitel wird das grundlegende medizinische Wissen vermittelt, das während der Entwicklungen in Betracht gezogen wird. Dazu gehören Informationen über die allgemeine Endoskopie, über den Magen und verschiedene Fremdkörper. Darüber hinaus vermittelt das Kapitel auch Informationen über das Endoskop als Werkzeug.

2.1 Endoskopie

Die Endoskopie ist eine medizinische Technik, die zum Einblick in verborgene Körperhöhlen genutzt wird und gehört heutzutage zur täglichen Routine in verschiedenen medizinischen Fachgebieten, wie zum Beispiel (z.B.) in der Gastroenterologie, Urologie oder der Pneumologie und geht mit ihrer Vielfalt der Möglichkeiten, weit über die visuelle Diagnose hinaus. Die Bezeichnung der endoskopischen Verfahren entstehen durch den Namen des zu untersuchenden Gegenstand in Kombination mit den Suffix „-skopie“, z.B. Gastroskopie. Systeme der Endoskopie setzten sich aus vielfältigen Komponenten zusammen, grundlegend Funktionen sind allerdings eine Ausleuchtung des Behandlungsgegenstand und die Erfassung von Bildern. Des Weiteren werden Instrumente benutzt, die einen invasiven Eingriff möglich machen. [1] Es gibt eine ausgesprochen große Vielfalt an

[1]Vgl. Klaus-Martin Irion / Martin Leonhard: Endoskopie – Geräte, Systeme und Methoden, in: Rüdiger Kramme. (Hrsg.) Medizintechnik, 5.Aufl.,Berlin, Deutschland: Springer Reference Technik, Springer Verlag, 2015, S.2 ff.

verschiedenen Instrumenten, die für unterschiedlichste medizinische Indikationen genutzt werden.[2]

2.2 Magen

Der Magen ist ein Organ des Verdauungssystem und findet sich im linken Oberbauch des menschlichen Körpers und wird in verschiedene Abschnitte eingeteilt. Im Pars cardiaca verläuft der Ösophagus in den Magen. Am höchsten Punkt des Magens befindet sich der Fundus gastricus, der im Stehen mit Luft gefüllt ist. Unter diesen beiden Abschnitten liegt der größte Abschnitt Corpus gastricum. Darunter liegt der Magenausgang Pars pylorica welcher sich ebenfalls ins Abschnitte einteilen lässt, das Antrum pyloricum, der Canalis pyloricus und der Pylorus, gefolgt von der 2-3 mm großen Öffnung Ostium pyloricum.
Die Innenseite des Magens verfügt über verschieden ausgeprägte Falten, welche in Längsrichtung ausgerichtet sind, diese sind im größten Abschnitt Corpus gastricum am höchsten und flachen nach oben in Richtung Curvatura major ab.[3] Die Magenwand besteht ebenfalls aus mehreren Schichten von denen allerdings nur die Tunica mucosa, die Schleimhaut des Magens, aus einer Perspektive innerhalb des Magens sichtbar ist.[4]

2.3 Fremdkörper

Die Extraktion von Fremdkörpern ist in der Endoskopie eine häufig vorkommende Aufgabe, wobei der Vielfalt von Fremdkörpern kaum Grenzen gesetzt sind. Vorwiegend werden die Fremdkörperentfernungen bei psychiatrischen oder kognitiv eingeschränkten Patienten, Gefängnisinsassen oder Kindern durchgeführt. Allerdings braucht es nicht für alle verschluckten Fremdkörper eine endoskopische Entfernung, diese wird erst bei dringlichen oder notfallmäßigen Indikationen notwendig, bei denen die Größe oder Eigenschaften des Objekts eine innere Verletzung auslösen können oder ein Ausscheiden des Objekts sehr unwahrscheinlich ist. Zu diesen notfallmäßig und dringlich zu entfernenden Fremdkörpern gehören im Ganzen scharfe oder spitze Objekte in Speiseröhre und Magen und ebenfalls Objekte, die eine Größe von sechs cm überschreiten. Ebenfalls sind Batterien zu entfernen, da sie durch ihre dünne Außenschicht auslaufen und zu Verätzungen führen können.

[2]Vgl.Irion / Leonhard, 2015, S. 9.
[3]Vgl. Michael Schünke / Erik Schulte / Udo Schumacher: Magen: Lage, Form, Gliederung und Innenansicht, in: Thieme physioLink, [online], https://physiolink.thieme.de/prometheus/236910105/236910105_4_117 [03.08.2020]
[4] Vgl. Michael Schünke / Erik Schulte / Udo Schumacher: Magen: Wandaufbau und Histologie, in: Thieme physioLink, [online], https://physiolink.thieme.de/prometheus/236910105/236910105_4_118 [03.08.2020]

Weitere Beispiele für zu entfernende Fremdkörper sind Essbesteck, Rasierklingen, Zahnbürsten, Magnete oder Knochen.[5]

2.4 Endoskope

Ein flexibles Endoskop besteht im Grunde aus einem Lichtübertragungssystem zur Beleuchtung des Operationsfeldes, welche durch Glasfaserbündelung erfolgt und des Weiteren aus einem System zur Bildübertragung, für welches man kleinste HD-Fernsehkameras verwendet, welche das Bild an einen Monitor übertragen. Sinnvoll ist hier ein kreisförmiges Bild oder eines mit den Formaten 5:3 oder 4:3, mit einem Blickwinkel von 140-170 Grad und einer Tiefenschärfe von 3-100mm. Die flexiblen Endoskope gibt es in verschiedenen Längen und Durchmessern, Routinegastroskope haben allerdings einen Schaftdurchmesser von ca. 10mm. Diese Endoskope werden durch eine Vielzahl von passenden Instrumenten ergänzt, dazu gehören etwa Greifer, Fasszangen, Drähte, Schlingen, Magnete etc. die in einer großen Menge verfügbar sind.[6] Die Spitzen der Geräte verfügen über Bowdenzüge, Mechaniken oder kleine Elektromotoren, mit denen am Handgriff des Geräts, die Endoskopspitze ferngesteuert werden kann. Je nach Größe sind hier zwei oder vier Seiten zur Bewegung, mit bis zu 180 Grad Winkel, möglich (auf-ab / auf-ab-rechts-links).[7]

Zur Extraktion von Fremdkörpern werden flexible Videoendoskopen mit einem Vorausblick und einem Instrument zur Extraktion bevorzugt. Zur Extraktion werden beispielhaft Schlingen, Körbchen und Fasszangen genutzt. Zusätzlich zu dem Videoendoskop, empfiehlt sich in einigen Fällen ein Ösophagustubus, welcher das wiederholte Schlucken und die Entstehung von Verletzungen vermeidet.[8] Eine weitere Schutzmaßnahme bei scharfen Gegenständen sind Overtubes und Schutzkappen, die am Ende des Endoskops angebracht werden. Diese ändern nach Greifen des Fremdkörpers ihre Ausrichtung und werden mit dem Endoskop und dem Objekt herausgezogen.[9]

[5]Vgl. Georg Kähler / Martin Götz / Norbert Senninger (Hrsg.): Therapeutische Endoskopie im Gastrointestinaltrakt, Berlin, Deutschland: Springer, 2016, S.178 ff.
[6]Vgl. Richard Salm / Karl Ernst Grund: Systeme für die Endoskopie, in Rüdiger Kramme (Hrsg.): Medizintechnik3.Aufl, Berlin, Deutschland: Springer, 2007, S. 350 ff.
[7] Vgl. Andreas Neff: Endoskopische Verfahren in der Mund-Kiefer-Gesichtschirurgie, Wiesbaden, Deutschland: Springer, 2015 S.9 f.
[8] Vgl. Gerd Lux: Gastrointerale Fremdkörper,in: Joachim F. Erckenbrecht / Sven Jonas (Hrsg.), Viszeralmedizin, Berlin, Deutschland: Springer, 2015, S.129 f.
[9] Vgl. Peter Collet: Endoskopische Fremdkörperextraktion, in: Georg Kähler / Martin Götz / Norbert Senninger (Hrsg.), Therapeutische Endoskopie im Gastrointestinaltrakt, Berlin, Deutschland: Springer, 2016, S.180 ff.

2.5 Schlussfolge

Die aufgeführten Bestandteile Magen, Endoskop und Fremdkörper sollen nun virtuell simuliert werden. Dazu wird die 3D-Engine Unity3D genutzt, in welche, Modelle eines 3D-Körpers, eines 3D-Endoskops und eines 3D-Fremdkörpers einsetzten werden.

3 Technische Grundlagen

Im folgenden Kapitel sollen die grundlegenden technischen Plattformen erwähnt werden, die bei der Umsetzung der Bachelorarbeit genutzt werden. Dazu gehören die Echtzeit-3D-Engine Unity3D, in welcher die Simulation stattfindet und darüber hinaus die Erweiterung durch Python und das Unity-ML-Agents-Toolkit.

3.1 Unity 3D

Unity ist eine Entwicklungsplattform des gleichnamigen Herstellers für Echtzeit 3D-Inhalte, mit welcher verschiedenste Simulationen entwickelt und getestet werden können. Des Weiteren bietet Unity die Möglichkeit, Toolkits zu integrieren und die Funktionalitäten zu erweitern.[10] Die fundamentalen Funktionen von Unity sind das Rendering von 3D-Szenen, die Verarbeitung von Input, die Berechnungen der Physik und die Kompilierung von C#-Code. [11] Die im Unity-Editor bearbeiteten 3D-Szenen, können anschließend auf verschiedenen Betriebssystemen, wie Windows, MacOS, Android, iOS oder Spielekonsolen veröffentlicht werden.[12] Diese Funktionen für Entwickler, finden auch in verschiedenen Naturwissenschaften Anwendung.[13]

Zur Bearbeitung der Bachelorarbeit wurde Unity 2018.4.22f1 gewählt.[14] Die Gründe hierfür sind, die Erfahrung des Autors mit Unity, der konstante Support der Plattform und die große Nutzerschaft an Entwicklern. 2018.4.22f1 war die Mindestanforderung für darauffolgende Implementationen und die Version, die eine stabile Integration dieser versprach.[15]

[10] Vgl. Unity: Produkte, in: Unity, [online] https://unity.com/de/products [4. 08 2019]

[11] Vgl. Unity: Funktionen in: Unity, [online] https://docs.unity3d.com/Manual/ [4. 08 2019]

[12] Vgl. Unity: Kernplattform, in: Unity, [online] https://unity.com/de/products/core-platform [4. 08 2019]

[13] Vgl. Unity: Lösungen, in: Unity, [online] https://unity.com/de/solutions [4. 08 2019]

[14] Vgl. Unity: Unity 2018.4.22f1, in: Unity Downloads, [online] https://download.unity3d.com/download_unity/3362ffbb7aa1/UnityDownloadAssistant-2018.4.22f1.exe [4. Juli 2019]

[15] Vgl. Unity: Installation Unity-ML-Agents, in: Unity ML-Agents Github, 5.08.2019, [online] https://github.com/Unity-Technologies/mlagents/blob/master/docs/Installation.md [30.07.2020]

3.2 Python

Python ist eine Programmiersprache, welche die objekt-orientierte, die prozedurale und die funktionale Programmierung unterstützt. [16] Der Python-Code wird vom Python 3 Interpreter gelesen und ausgeführt.[17] Zur Erweiterung von Python können verschiedene Bibliotheken genutzt werden, diese beinhalten z.B. mathematische Funktionen oder Funktionen zur Datenanalyse und -visualisierung.[18] So entstehen verschiedene Lösungen und Erfolge für die Wissenschaft, das Ingenieurswesen oder die Wirtschaft.[19] Für die Bachelorarbeit wurde Python 3.7.8 verwendet und zur Erweiterung wurden die beiden grundlegenden Python-Module *ML-Agents 0.18.0* und *Tensorflow 2.2.0* genutzt.

3.2.1 ML Agents - Modul

Das Python-Modul *mlagents* besteht aus anderen, verschiedenen Python-Modulen.[20] Dazu gehören unter anderem tensorflow, numpy oder pillow.[21] So besitzt das Modul verschiedene Algorithmen für das Machine Learning unter Reinforcement Learning oder für die Imitation von Demonstrationen. [22]

3.2.2 Tensorflow - Modul

Tensorflow ist eine Open-Source-Plattform, mit Tools, Bibliotheken und Ressourcen, die für Machine Learning genutzt werden. Die flexible Architektur und die Modellbauart erlauben vielfältige Anwendungsmöglichkeiten[23]

3.3 Unity ML-Agents Toolkit

Das Unity ML-Agents Toolkit ermöglicht die Interaktion mit einer 3D-Szene im Unity-Editor, über eine Python API. Darin beinhaltet ist das ML-Agents SDK mit allen notwendigen Funktionalitäten und C#-Skripten, um eine 3D- Simulation und einen Lernprozess einzurichten. Außerdem im Toolkit enthalten, sind die Reinforcement Learning Algorithmen

[16] Vgl. Python: Python Allgemein, in:Python Dokumentation, [online] https://docs.pyhon.org/3/faq/general.html#what-is-python [5. 08 2019]
[17] Vgl. Python: Python für Anfänger, in: Python Wiki, [online] https://wiki.python.org/moin/BeginnersGuide [5. 08 2019]
[18] Vgl. Python: Python Anwendungen, in Python, [online] https://www.python.org/about/apps/ [5. 08 2019]
[19] Vgl. Python: Python Lösungen und Erfolge,in Python,[online] https://www.python.org/success-stories/ [5. 08 2019]
[20] Vgl. Unity: Installation von Unity-ML-Agents, in: Unity ML-Agents Github, 5. 08 2019, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Installation.md [30.07.2020]
[21] Vgl. Unity: ML-Agents Python Modul, in: Unity ML-Agents Github, 5. 08 2019, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/ml-agents/setup.py [30.07.2020]
[22] Vgl. pypi: ML-Agents Python Modul, in: pypi-mlagents, [online] https://pypi.org/project/mlagents/0.18.0/ [25.08.2020]
[23] Vgl.Tensorflow: Warum Tensorflow, in Tensorflow,[online] https://www.tensorflow.org/ [5. 08 2019]

PPO und *SAC* und die Algorithmen für Imitation Learning, *Behavioral Cloning (BC)* und *Generative Adversarial Imitation Learning*. Des Weiteren bietet das Toolkit Algorithmen zum Selbständigen-Spielen und Erweiterungen für ein Intrinsic Curiosity Module oder auch für Long-Short-Term Cells.

Die wesentlichen Instanzen des ML-Agents SDK sind die Agents, die Sensoren(englisch: sensors) und die Akademie(englisch: Academy). Die Agents sind dabei 3D-Objekte, innerhalb einer 3D-Szene, die Observationen (englisch: *observations)* sammeln, Aktionen (englisch: *actions)* durchführen und dadurch Belohnungen (englisch: *rewards)* erhalten können. Jeder Agent besitzt hier eine policy (deutsch: *Richtlinie),* die einen Verhaltensnamen (englisch: *behavior name*)trägt und auch von mehreren Agents gleichzeitig genutzt werden kann. Dadurch können die während des Trainings gesammelten Erfahrungen, geteilt werden. Die Observationen dienen als Sensoren und können unterschiedliche Datentypen enthalten, wie z.B. Vektoren und deren Länge oder gar gerenderte Aufnahmen der 3D-Szene.

Die Belohnungen bieten dem Agent Signale zum Lernen, welche jederzeit angepasst werden können. Außerdem können Trainings in einen erledigt(englisch: done)–Zustand versetzt werden. Die Akademie existiert nur einmal und verwaltet die Agents und die Anzahl der durchgeführten Schritte (englisch: steps*).* Des Weiteren kann die Akademie, während der Laufzeit, Umgebungsparameter ändern.[24]

4 Realisierung

Im folgenden Kapitel wird die Umsetzung des Projekts dargestellt. Hierzu beschreiben die Texte den Aufbau der Simulation, den Nachbau und die Funktionen des Endoskops als Unity-ML-Agent und zuletzt den Ablauf des Trainings und der anschließenden Tests.

4.1 Simulation

Die Simulation beinhaltet verschiedene Bestandteile, dazu gehören ein 3D-Modell des menschlichen Körpers, ein 3D-Nachbau eines Endoskops und zwei verschiedene 3D-Modelle von Fremdkörpern. Hierbei sind das Endoskop und der Fremdkörper innerhalb des Magens des 3D-Körpers platziert. Die 3D-Objekte werden zudem jeweils mit verschiedenen

[24] Vgl. Unity: Unity: A General Platform for Intelligent Agents, Publikation, San Francisco, USA: Unity Technologies, 2020, S.11 ff.

C#-Skripten ausgestattet. Das Endoskop wird dabei als Unity-ML-Agents implementiert und trägt das wichtigste C#-Skript mit dem Namen EndoscopeAgentScript.cs.

4.1.1 Menschlicher Körper

Zur Simulation des menschlichen Körpers wurde das statische 3D -Modell eines menschlichen Torsos genutzt (Abb. 1 - 3), das den Beschreibungen des Magens in Kapitel 2.2 , möglichst nahe sein sollte. So besitzt das 3D-Modelle im annähernd die Maßstäbe und auch die richtigen Formen und Abschnitte des Magens. Die Vereinfachung der Verdauungsorgane verursacht allerdings auch das Verringern des Detailgrades, so enthält das Modell keine Falten der Magenwand und auch eine vereinfachte Textur.
Die Vereinfachung hat allerdings keinen Einfluss auf die weitere Entwicklung.

Das Modell lag im Dateiformat .fbx vor und enthielt, nach wenigen Änderungen, lediglich die inneren Verdauungsorgane, wie z.B. Speiseröhre, Magen, oder Darm und des Weiteren die Haut des Körpers.

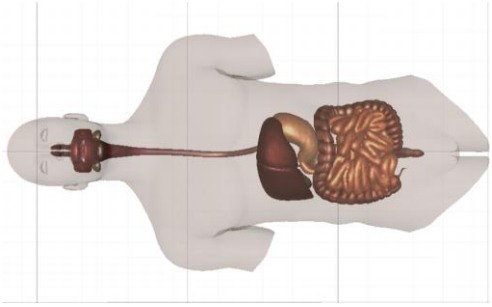


Abb. 1 3D-Modell des Körpers (frontal), Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Abb. 2 3D-Modell des Körpers (seitlich), Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Abb. 3 3D-Modell des Körpers (von oben), Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Um das virtuelle Endoskop innerhalb des Magens zu platzieren und dennoch die Polygone des 3D-Körpers sehen zu können, war es notwendig die Polygone des Torsos, zwecks Normalen-Ausrichtung und Aufzählung der Punkte, umzudrehen.
Dazu wurde das folgende C#-Skript, mit dem Namen FlipFaces.cs (Quellcode 1), erstellt und auf das Körper-Modell angewandt, um die Orientierung der Polygone des 3D-Modelles nach Innen auszurichten. Dies war zum einen notwendig um, auch mit einer Kamera innerhalb des Modells, die Polygone sichtbar zu machen und zum anderen, um im Folgenden von Unitys Kollisionserkennung Gebrauch zu machen, die von der Ausrichtung der Polygone abhängt.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Linq;

public class FlipFaces : MonoBehaviour
{
    void Start()
    {
        //In folgender Zeile wird das Polygon-Netz des Körpers auf die Variable mesh zugewiesen(Objekttyp Mesh)
        Mesh mesh = GetComponent<MeshFilter>().mesh;
        // In der nächsten Zeile werden alle geformten Dreiecke des Polygon-Netzes in ihrer Aufzählung umgedreht, um so ihre Orientierung gegenteilig und nach innen umzukehren.
        mesh.triangles = mesh.triangles.Reverse().ToArray();
    }
```

Quellcode 1 FlipFaces.cs , Quelle: Eigene Programmierung

Anschließend wurde dem 3D-Modell ein Unity-Mesh-Collider hinzugefügt, welcher der Form des Modells entspricht und die Kollisionen zwischen dem 3D-Körper und dem 3D-Endoskop erkennbar macht.
Da das 3D-Modell des Körpers von einem freien Künstler übernommen wurde, gilt es Angaben zum Urheber zu machen, damit eine freie Nutzung des 3D-Modells gestattet ist. Die Urheber- und Rechtsangaben finden sich im Anhang.1

4.1.2 Fremdkörper

Die nicht-organischen Fremdkörper wurden in den Versuchen durch zwei verschiedene 3D-Objekte repräsentiert. Zum einen durch eine graue Münze (Abb. 4-5) und zum anderen durch eine graue Kugel (Abb. 6). Dabei entsprach die Münze dem geometrischen Körper eines flachen Zylinders.

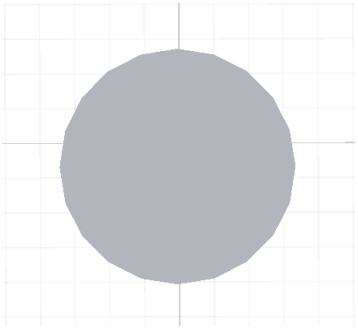


Abb. 4 Fremdkörper Münze (frontal), Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Abb. 5 Fremdkörper Münze (von oben), Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

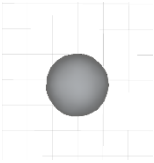


Abb. 6 Fremdkörper Kugel, Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

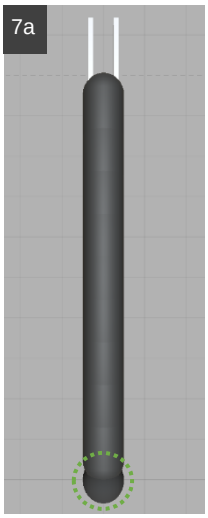
Beide Fremdkörper besaßen zudem einen Unity-Rigidbody und einen passenden Unity-Collider. Der Unity-Rigidbody hat die Funktion die physikalischen Eigenschaften der 3D-Objekte zu simulieren, so können z.B. die Masse oder Reibung der 3D-Objekte festlegen werden. Die beiden 3D-Objekte enthielten ebenfalls einen Unity-Collider. Diese haben beide die Form der Objekte und sollten die Kollisionen zwischen Fremdkörper und Endoskop oder auch Kollisionen zwischen Fremdkörper und dem Körper erkennen.

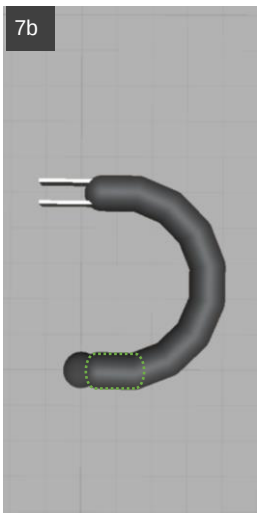
Darüber hinaus wurden die beiden 3D-Fremdkörper mit dem folgenden und selbsterstellten C#-Skript CollisionInformation.cs (Quellcode 2) ausgestattet, indem verschiedene boolesche Werte verändert wurde, je nachdem wo der Fremdkörper die Zangenspitze berührt hat (Innen oder Außen). Diese Informationen über den Kontaktpunkt, wurden dem Agent, über das EndoscopeAgentScript.cs mittgeteilt, indem die booleschen Werte Zange1SideTouched, Zange2SideTouched, Zange1Touched und Zange2Touched verändert wurden. Diese werden im EndoscopeAgentScript.cs initialisiert und können, nach ihrer Veränderung durch das CollisionInformation.cs, vom Agent weiterverarbeitet werden.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CollisionInformation : MonoBehaviour
{
    public GameObject Zange1; // Das sind die beiden Zangenzähne in der 3D-Szene
    public GameObject Zange2;

    //Die Funktion OnCollisionEnter wird aufgerufen sobald es eine Kollision gegeben hat
    private void OnCollisionEnter(Collision collision)
    {
        //Das ist der Normalenvektor derjenigen Fläche, die berührt wurde
        Vector3 normal = collision.contacts[0].normal;

        //Abfrage ob und welcher Zangenzahn berührt wurde
        if (collision.collider.gameObject.name == "ZangeLowPoly1")
        {
            //Falls Zahn1 berührt wurde-> im EndoscopeAgentScript den Boolean Zange1SideTouched auf true
            EndoscopeAgentScript.Zange1SideTouched = true;

            //Abfrage ob die Innenseite des ersten Zangenzahns berührt wurde (Normale der Kollisionsfläche == Normale des
Zangenzahns?)
            if (normal == Zange1.transform.right * -1 )
            {
                //Falls die Innenseite berührt wurde-> im EndoscopeAgentScript den Boolean Zange1Touched auf true setzten
                EndoscopeAgentScript.Zange1Touched = true;
            }
        }
        else if (collision.collider.gameObject.name == "ZangeLowPoly2")
        {
            //Falls Zahn2 berührt wurde-> im EndoscopeAgentScript den Boolean Zange2SideTouched auf true setzen
            EndoscopeAgentScript.Zange2SideTouched = true;

            //Abfrage ob die Innenseite des zweiten Zangenzahns berührt wurde (Normale der Kollisionsfläche == Normale des
Zangenzahns?)
            if (normal == Zange2.transform.right )
            {
                //Falls die Innenseite berührt wurde-> im EndoscopeAgentScript den Boolean Zange2Touched auf true
                EndoscopeAgentScript.Zange2Touched = true;
            }
        }

        //Abfrage ob der Fremdkörper sich in der Mitte der Zange bedindet
        if(collision.collider.gameObject.name == "MitteDerZange")
        {
            //Falls ja, setzte im Agent-Script den Boolean MitteTouched auf true
            EndoscopeAgentScript.MitteTouched = true;
        }
    }

    // Die Funktion OnCollisionExit wird aufgerufen sobald die Kollision beendet wurde
    // Hier machen mir alle obigen Veränderungen der Boolean wieder rückgängig, sobald die Kollisionen beendet werden
    private void OnCollisionExit(Collision collision)
    {
         //Alle Boolschen Werte des ersten Zangenzahns wieder auf false
        if (collision.collider.gameObject.name == "ZangeLowPoly1")
        {
            EndoscopeAgentScript.Zange1Touched = false;
            EndoscopeAgentScript.Zange1SideTouched = false;
```

Quellcode 2 CollisionInformation.cs, Quelle: Eigene Programmierung

4.1.3 Endoskop





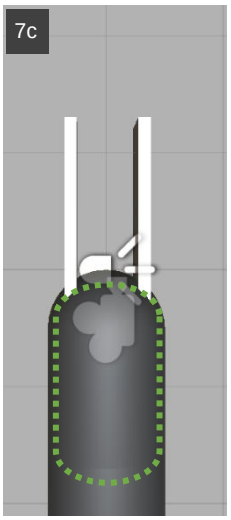
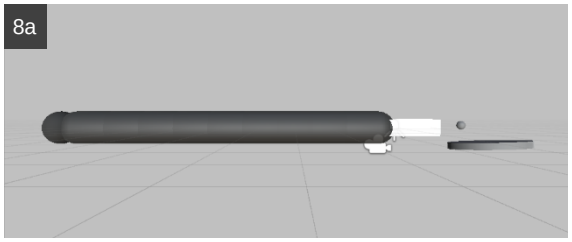


Abb. 7 a-c Aufbau des Endoskops, Quelle: Eigene Darstellung / Unity Bildschirmaufnahme



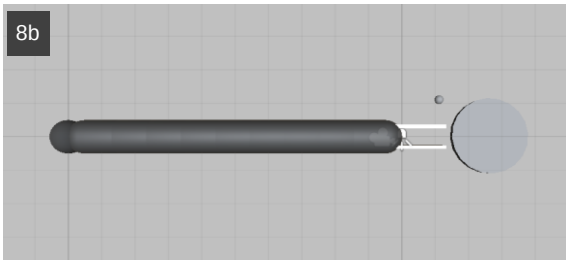
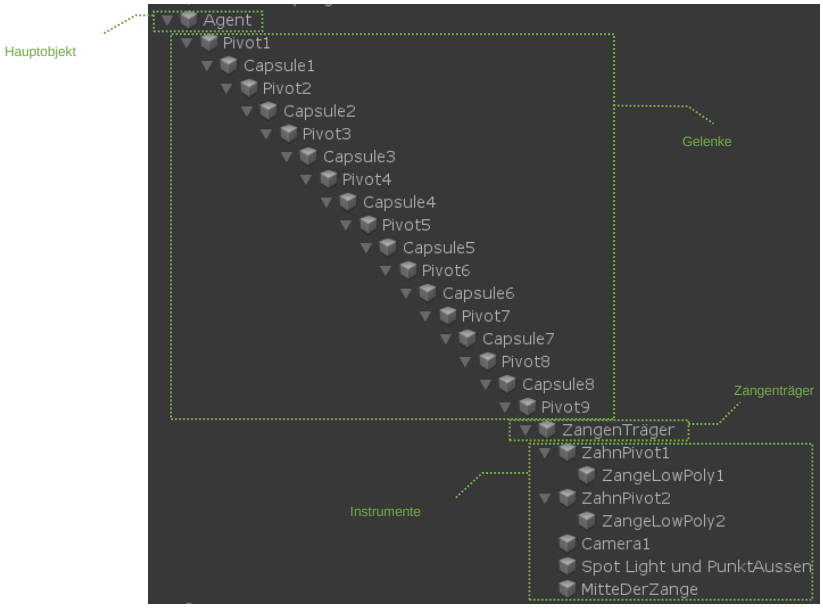


Abb. 8 a-b Größenverhältnisse der 3D-Objekte, Quelle: Eigene Darstellung /Unity Bildschirmaufnahme

Das 3D-Endoskop bestand aus mehreren Unity-GameObjects, die verschiedene Komponenten formten und in ihrer Hierarchie wie in Abb. 9 und Abb. 10 angeordnet sind. Zuerst das Basisobjekt, dann die flexiblen Gelenke und zuletzt der Zangenträger. Diese Objekte sind wie folgt in ihrer Hierarchie angeordnet:



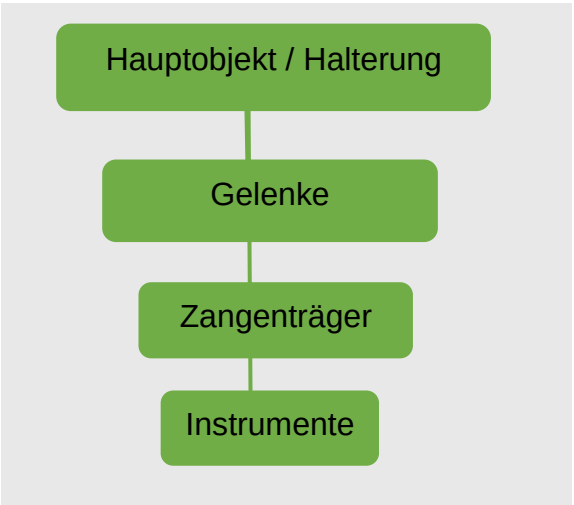


Abb. 9 Vereinfachte Darstellung der Hierarchie, Quelle: Eigene Darstellung

Abb. 10 Hierarchie der 3D-Bestandteile des Endoskops , Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Das Basisobjekt (Abb. 7a) besitzt neben einer grauen Kugel, einen Unity-Rigidbody für die physikalischen Eigenschaften (z.B Masse o. Reibung) und des Weiteren einen Unity-Sphere-Collider für die Kollisionen. Darüber hinaus enthält das Basisobjekt die notwendigen

Komponenten der Unity-ML Agents und das C#-Skript EndoscopeAgentScript.cs, welche das Basisobjekt zum Unity-Agent machten. Die Komponenten und das C#-Skript EndoscopeAgentScript.cs werden nach dem grundlegenden Aufbau des Endoskops genauer erklärt.

Das Gelenk wurde vom Basisobjekt getragen und bestand aus mehreren schwarzen Kapseln, die in einer Linie, durch leere Gelenkobjekte verbunden wurden (Abb.7b). Die Kapseln erbten hierbei der Unity-Rigidbody des Basisobjekts, jedoch erhielten sie einen Unity-Capsule-Colliders, statt eines Unity-Sphere-Colliders.

Der Zangenträger (Abb. 7c), der vom Gelenk getragen wurde, erbte ebenfalls den Unity-Rigidbody des Basisobjekts, erhielt allerdings, wie die Kapseln des Gelenks, einen Unity-Capsule-Collider.

Erweitert wurde der Zangenträger durch die Zangenzähne, eine Kamera und eine Lichtquelle. Die zwei Zangenzähne wurden von zwei Würfeln dargestellt, die in y-Richtung gestreckt und in x- und z-Richtung gestaucht wurden. Diese enthielten neben dem geerbten Unity-Rigidbody auch die passenden Unity-Box-Collider. Neben den Zangen trug der Zangenträger auch eine RGB-Kamera mit einer Auflösung 100x80 und einem Sichtfeld von mindestens 90 Grad. Zuletzt enthielt der Zangenträger eine Spotlight-Lichtquelle mit weißer Farbe und dem Leuchtwinkel des Kamerasichtfelds.

Mit den Abbildungen 8a bis 8b wird zuletzt das Größenverhältnis vom Endoskop zum Fremdkörper verdeutlicht.

4.2 Das Endoskop als Unity-ML-Agent

Um das Endoskop als Unity-ML-Agent zu initialisieren, mussten dem Basisobjekt verschiedene Komponenten hinzugefügt werden, welche die Funktionen der Unity-ML-Agents (Agents) integrieren. Dazu gehörten die Komponenten Behavior Parameter, Decision Requester, Camera Sensor Component, Demonstration Recorder und das selbstgeschriebene C#-Skript EndoscopeAgentScript.cs.

Die Behavior Parameter ermöglichten verschiedene Funktionen, wie z.B. die Zuweisung der späteren Hyperparameter, die Anpassung von Größen und Form der Observations/Actions oder die Funktion, dem Agent ein bereits trainiertes Modell zuzuweisen.

Der Decision Requester forderte den Unity-Agent auf, im angegebenen Intervall eine Entscheidung zu treffen. Diese Anzahl der Entscheidungen konnte dabei von der Anzahl der durchgeführten Schritte abweichen.

Die Camera Sensor Component ermöglichte es, ein Kamerabild der Echtzeit-3D-Szene als Observation des Agents zu nutzen und dieses Bild anzupassen. Verschiedene Variablen sind hier die Auflösung des Bildes, die Farbgebung oder eine mögliche Kompression des Bildes.

Der Demonstration Recorder konnte vor dem Training genutzt werden, um mehrere Trainingsabläufe selbst zu absolvieren und diese auszuzeichnen. Die Aufzeichnung konnten dann in das Training eingebracht werden.

Nachdem die essenziellen Komponenten hinzugefügt wurden um das Basisobjekt als Agent zu initialisieren, wurde der Agent mit dem eigens geschriebenen C#-Skript EndoscopeAgentScript.cs (Quellcode 3) ausgestattet. Dieses Skript enthielt verschiedene Funktionalitäten wie die möglichen Observationen / Aktionen des Agents , die erreichbaren Belohnungen oder die Neupositionieren des Fremdkörpers. Im Folgenden wird das C#-Skript EndoscopeAgentScript.cs im Ganzen dargestellt, wobei das gesamte Verfahren innerhalb des C#-Skripts kommentiert wurde. Die wichtigsten Funktionalitäten werden in anschließenden Texten aufgegriffen und genauer betrachtet.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Sensors;

public class EndoscopeAgentScript : Agent
{

    #region Objekte
    public GameObject OesoUndMagen; // Das sind das Modell des Körpers
    public GameObject Coin;         // Das ist der Fremdkörper
    Rigidbody AgentsRigidbody;      // Das ist das Basisobjekt/ der Agent. Wir brauchen im Code allerdings nur den Physik-Rigidbody.
    public GameObject[] allPivots;  // Das ist ein Array an allen Drehpunkten (Pivots) des Gelenks
    public GameObject CapsuleZangenTraeger;     // Das ist der Träger der Instrumente (Zange, etc.)
    public GameObject aeusesterPunkt;           // Äußester Punkt des Zangenträgers (In diesem Fall die Lichtquelle)

    public GameObject Zahn1;            // Zangenzahn 1 von 2
    public GameObject Zahn2;            // Zangenzahn 2 von 2
    public GameObject ZahnPivot1;       // Gelenkpunkte des Zagenzahns 1 von 2
    public GameObject ZahnPivot2;       // Gelenkpunkte des Zagenzahns 2 von 2
    #endregion

    #region Variablen für Vor-/ Zurückbewegung und Rotation des Basisobjekts / des Agents
    Vector3 beginningPosition;                  //Startposition des Agents
    Quaternion agentStartRotation;              //Startrotation des Agents
    public static float movementSpeed = 200;    //Geschwindigkeit des Vor- und Zurückbewegung des Agents
    public float rotationSpeed = 5.0f;          //Geschwindigkeit der Rotation des Agents
    #endregion

    #region Variablen für das Biegen des Gelenks
    Quaternion pivotRotation;                   //Der Winkel der Beugung
    public float BendingAngleSpeed = 1.0f;      //Die Geschwindigkeit der Beugung / Rotation der Gelenkpunkte
    #endregion

    #region Variablen für die Rotation des Zangenträgers/ Instrumentträgers
    Quaternion CapsuleZangenTraegerStartRotation;   // Der Winkel der Rotation des Zangenträgers
    public float ZangenRotationSpeed = 1;           // Die Geschwindigkeit des Rotation des Zangenträgers
    #endregion

    #region Variablen für das Zugreifen mit der Zange
    Vector3 Zahn1LocalStartPosition;        // Die Startposition des Zangenzahns 1 von 2
    Vector3 Zahn2LocalStartPosition;        // Die Startposition des Zangenzahns 2 von 2
    Quaternion zahn1BeginningRotation;      // Die Startrotation des Zangenzahns 1 von 2
```

```
    Quaternion zahn2BeginningRotation;      // Die Startrotation des Zangenzahns 2 von 2
    public float openingSpeed = 3.0f;       // Die Geschwindigkeit in der die Zange geöffnet oder geschlossen wird
    #endregion

    #region Variablen für den Fremdkörper
    Vector3 coinStartPosition;              // Startposition der Fremdkörpers
    Quaternion coinStartRotation;           // Startrotation des Fremdkörpers (Weltkoordinatensystem)
    Quaternion CoinStartLocalRotation;      // Startrotation des Fremdkörpers (Lokales Koordinatensystem)
    float coinRotationAngle;                // Winkel der Rotation des Fremdkörpers

    public float coinSpawnRadius = 2;       // Radius innerhalb dem der Fremdkörper positioniert werden kann
    public bool randomCoinPosition = false; // Boolen für die Entwicklungen und Code-Testing

    #endregion


  #region Variablen für die Rewards

    //Die folgenden Boolschen Werte vermitteln Informationen über den Kontakt zwischen Zange und Fremdkörper
    public static bool Zange1SideTouched = false;          // Zangenberührung jeder Art für Zangenzahn 1 von 2
    public static bool Zange2SideTouched = false;          // Zangenberührung jeder Art für Zangenzahn 2 von 2
    public static bool Zange1Touched = false;              // Zangenberührung im Inneren des Zangenzahns 1 von 2
    public static bool Zange2Touched = false;              // Zangenberührung im Inneren des Zangenzahns 2 von 2
    public static bool MitteTouched = false;               // Berührung der Zangenmitte zwischen den beiden Zangenzähnen (auch ohne jede Berührung der Zangenzähne)


    float distanceZahn1ToCoin;                          // Abstand zwischen Zangenzahn 1 und dem Fremdkörper
    float distanceZahn2ToCoin;                          // Abstand zwischen Zangenzahn 2 und dem Fremdkörper

    float CoinAbstandNACHaction;                        //Abstand zwischen dem äußesterm Punkt des Zangenträgers und dem Fremdkörper ( VOR der Aktion des Agents)
    float CoinAbstandVORaction;                         //Abstand zwischen dem äußesterm Punkt des Zangenträgers und dem Fremdkörper ( NACH der Aktion des Agents)

    public static float holdingTime = 0;                // Die gemessene Zeit in der die Zange den Fremdkörper hält
    #endregion

    #region Variablen für verschiedene Zähler während des Trainings
    int BeruhrungJederArt = 0;              // Zähler für die Berührungen der Zangenzähne an jeder Stelle
    int BeruhrungDerInnenSeite = 0;         // Zähler für die Berührungen der Innenseite der Zangenzähne
    int AnzahlGegriffen = 0;                // Zähler für das Greifen mit beiden Zangenzähnen
    int AnzahlGehalten = 0;                 // Zähler für das Greifen(1-3 sec.) mit beiden Zangenzähnen.
    int AnzahlInDerMitte = 0;               // Zähler für die Positionierung des Fremdkörpers in der Mitte zwischen beiden Zangenzähnen, auch ohne Berührungen der Zangenzähne
    int AnzahlInDerMitteForCoin = 0;        // Ebenfalls ein Zähler für die Positionierung des Fremdkörpers in der Mitte, allerdings für andere Funktionen

    int stepCounter = 0;                    // Zählt die Aktionen/Schritte die vom Agent durchgefürt wurden
    int VersuchsCounter = 0;                // Zählt die Anzahl der Versuche
    public int ZyklusCount = 50000;         // Anzahl der Schritte in die der CoinSpawnRadius eingeteilt wird (CoindSpawnRadius / ZyklusCount = Schrittgröße )

    #endregion

    //Die Start()-Funktion wird zum Start der Anwendung aufgerufen
    // Hier werden verschiedene, der oben aufgeführten, Variablen ihren zugehörigen Werten in der 3D-Szene zugewiesen
    void Start()
    {
        //Agent: Physik-Rigidbody mit Startposition und -rotation
        beginningPosition = this.transform.position;
        agentStartRotation = this.transform.rotation;
        AgentsRigidbody = GetComponent<Rigidbody>();

        CapsuleZangenTraegerStartRotation = CapsuleZangenTraeger.transform.rotation; //Startposition des Zangenträgers

        //Position und Rotation der beiden Zangenzähne
        Zahn1LocalStartPosition = Zahn1.transform.localPosition;
        Zahn2LocalStartPosition = Zahn2.transform.localPosition;
        zahn1BeginningRotation = ZahnPivot1.transform.localRotation;
        zahn2BeginningRotation = ZahnPivot2.transform.localRotation;

        //Startposition und -rotation des Fremdkörpers
        coinStartPosition = Coin.transform.position;
        coinStartRotation = Coin.transform.rotation;
        CoinStartLocalRotation = Coin.transform.localRotation;
    }

    //Die Heuristic()-Funktion ermöglicht es, Tastatur-Eingaben zu nutzen um diese wie eine Aktion des Agents zu verwenden (Für Testzwecke)
    public override void Heuristic(float[] actionsOut)
    {
        //Agent vor & Zurück mit den Pfeiltasten Hoch und Runter
        actionsOut[0] = Input.GetAxis("Vertical");
        //Agent rotieren mit den Pfeiltasten Links und Rechts
        actionsOut[1] = Input.GetAxis("Horizontal") * -1;
        //Gelenk beugen mit einer Bewegung der Maus auf der vertikalen Achse
        actionsOut[2] = Input.GetAxis("Mouse Y");
        //Zangendrehung mit einer Bewegung der Maus auf horizontaler Ache
        actionsOut[3] = Input.GetAxis("Mouse X");
```

```
    //Zange öffnen und schließen mit der Drehung der Mausrads
    actionsOut[4] = Input.mouseScrollDelta.y;

}

//Die Funktion OnEpisodeBegin wird jedesmal aufgerufen wenn eine neue Trainingseinheit beginnt (der einzelne Versuch)
// Dies kann z.B. passieren wenn der Agent die Aufgabe löst und ein neuer Verusch gestartet wird, oder der Versuch wird abgebrochen und ein neuer gestartet
// In unserer Funktion werden hauptsächlich die Bedingungen zum Start des Versuchs wiederhergestellt
public override void OnEpisodeBegin()
{
    //Position, Rotation und physikalische Gegebenheiten des Agents werden zurückgesetzt (auf ihre Startwerte oder auf 0)
    this.transform.position = beginningPosition;
    this.transform.rotation = agentStartRotation;
    AgentsRigidbody.velocity = new Vector3(0, 0, 0);
    AgentsRigidbody.angularVelocity = new Vector3(0, 0, 0);

    //Alle Gelenkpunkte des Gelenks werden zurückgesetzt
    foreach(GameObject pivot in allPivots)
    {
        pivot.transform.localRotation = Quaternion.identity;

    }

    // Die Rotation des Zangenträgers wird wieder zurückgesetzt
    CapsuleZangenTraeger.transform.rotation = CapsuleZangenTraegerStartRotation;

    // Die Positionen und Rotationen der Zangenzähne wird zurückgesetzt
    Zahn1.transform.localPosition = Zahn1LocalStartPosition;
    Zahn2.transform.localPosition = Zahn2LocalStartPosition;
    ZahnPivot1.transform.localRotation = zahn1BeginningRotation;
    ZahnPivot2.transform.localRotation = zahn2BeginningRotation;

    //Alle Boolschen Werte der Kollisionen werden auf false gesetzt
    Zange1Touched = false;
    Zange2Touched = false;
    Zange1SideTouched = false;
    Zange2SideTouched = false;
    MitteTouched = false;

    //Zu Beginn des Versuchs, die Abfrage ob der Fremkörper zufällig platziert werden soll oder nicht
    if (randomCoinPosition == true)
    {
        SpawnCoinAtRandomPositionInMesh(OesoUndMagen);  //Aufruf der Funktion welche den Fremdkörper zufällig platziert
    }
    else
    {
        Coin.transform.position = coinStartPosition;    // Position des Fremdkörpers bleibt stets die gleiche
    }

   VersuchsCounter++;  //Der Zähler für die Anzahl der Versuche wird erhöhrt
   Debug.Log("Die Anzahl der Versuche: " + VersuchsCounter); //Die Anzahl der Versuche wird ausgegeben

}

// In der Funktion CollectObservations() können wir definieren welche Teile der 3D-Szene, dem Agent als Observationen dienen (z.B. Vektoren oder Winkel)
public override void CollectObservations(VectorSensor sensor)
{
    //Observationen zum Agent selbst
    sensor.AddObservation(beginningPosition - this.transform.position);     // Vektor zwischen Start- und aktueller Position   (Vector3)
    sensor.AddObservation(this.transform.rotation.y);                       // Die Rotation des Agents                         (float)
    sensor.AddObservation(pivotRotation);                                   // Die Winkel der Gelenkbeugung                    (float)
    sensor.AddObservation(CapsuleZangenTraeger.transform.localRotation.y);  // Der Winkel der Rotation des Zangenträgers        (float)
    sensor.AddObservation(Zahn1.transform.localPosition.x);                 // Die lokale Position des ersten Zangenzahn        (float)
    sensor.AddObservation(Zahn2.transform.localPosition.x);                 // Die lokale Position des zweiten Zangenzahn       (float)

    //Observationen zu Berührungen
    sensor.AddObservation(Zange1Touched);      //Berührung der Innenseite von Zangenzahn 1                                  (bool)
    sensor.AddObservation(Zange2Touched);      //Berührung der Innenseite von Zangenzahn 2                                  (bool)
    sensor.AddObservation(MitteTouched);       //Berührung der Mitte zwischen den Zangenzähnen                               (bool)

    //Observationen zur Münze
    sensor.AddObservation(Coin.transform.position - aeusesterPunkt.transform.position);     //Vektor zwischen Fremdkörper und Zangenträger  (Vector3)
    sensor.AddObservation(CoinAbstandNACHaction);                                           //Abstand zwischen Fremdkörper und Zangenmitte  (float)
}

//In der Funktion OnActionReceived() werden die Aktionen des Agents verarbeitet
public override void OnActionReceived(float[] vectorAction)
{
    //Noch bevor die Aktionen durchgeführt werden, wird der Abstand zum Fremdkörper gemessen, um diesen nach der Aktion zu vergleichen
    CoinAbstandVORaction = (aeusesterPunkt.transform.position - Coin.transform.position).magnitude;     //Länge des Vektors zwischen Rand des Zangenträger und Fremdkörper

    //AKTION 1: Vor- und Zurückbewegung des Agents / Endoskops
    float MovementAction = vectorAction[0];            //Die Aktion des Agents wird einem Float zugewiesen
```

```
    AgentsRigidbody.AddForce( MovementAction * AgentsRigidbody.transform.up * movementSpeed);  //Die Aktion wirkt eine Kraft auf den Physik-Rigidbody des Agents und verschiebt ihn nach vorn

    //AKTION 2: Rotation des Agents / Endoskops
    float RotationAction = vectorAction[1];            //Die Aktion des Agents wird einem Float zugewiesen
    transform.Rotate(0, RotationAction * rotationSpeed, 0); //Die Aktion führt zu einer Rotation des Agents

    //AKTION 3: Biegen des Gelenks
    float BendingAction = vectorAction[2];             //Die Aktion des Agents wird einem Float zugewiesen
    //Wir gehen nun alle Gelenkpunkte des Gelenkes durch und wirken die Aktion mit einer Rotation darauf aus
    foreach (GameObject pivot in allPivots)            //Alle Gelenkpunkte des Gelenks werden durchgegangen
    {
        pivotRotation = pivot.transform.localRotation;                    //Zwischenzuweisung des aktuellen Winkels des Gelenkpunkts
        pivotRotation.z += BendingAction * BendingAngleSpeed * Time.deltaTime;  //Die Aktion führt eine Rotation des Gelenkpunktes durch (Z-Achse)
        pivotRotation.z = Mathf.Clamp(pivotRotation.z, -0.2f, 0.2f);          //Die mögliche Rotation der Gelenkpunkte wird zwischen -0.2 und 0.2 gehalten (Radiant)
        pivot.transform.localRotation = pivotRotation;                    //Zuweisung des veränderten Winkels auf die Rotation des Gelenkpunktes
    }

    //AKTION 4: Zangenträger rotieren
    float ZangenRotationAction = vectorAction[3]; //Zuweisung der Aktion des Agents
    CapsuleZangenTraeger.transform.Rotate(0, ZangenRotationAction * -ZangenRotationSpeed, 0, Space.Self); //Die Aktion führt eine Rotation des Zangenträgers aus

    //AKTION 5: Zange schließen und öffnen
    float ZangenOpeningAction = vectorAction[4]; //Aktion des Agents als float
    Vector3 localRight = Zahn1.transform.worldToLocalMatrix.MultiplyVector(Zahn1.transform.right);  //Hilfsvektor der die Richtung der Bewegung der Zangenzähne trägt
    Zahn1.transform.Translate((-localRight * ZangenOpeningAction) * openingSpeed);  //Aktion führt Bewegung in Richtung Hilfsvektor*-1 aus
    Zahn2.transform.Translate((localRight * ZangenOpeningAction) * openingSpeed);   //Aktion führt Bewegung in Richtung Hilfsvektor aus
    //Diese IF-Statements sind Limits für die Zangenzahn-Bewegung
    if (Zahn1.transform.localPosition.x < -0.2f) { Zahn1.transform.localPosition = new Vector3(-0.2f, 1.5f, 0); }
    if(Zahn1.transform.localPosition.x > 0) { Zahn1.transform.localPosition = new Vector3(0, 1.5f, 0); }
    if (Zahn2.transform.localPosition.x > 0.2f) { Zahn2.transform.localPosition = new Vector3(0.2f, 1.5f, 0); }
    if (Zahn2.transform.localPosition.x < 0) { Zahn2.transform.localPosition = new Vector3(0, 1.5f, 0); }


    //Nach allen Aktionen des Agents wird nochmal der Abstand zum Fremdkörper ermittelt
    CoinAbstandNACHaction = (aeusesterPunkt.transform.position - Coin.transform.position).magnitude;

    //Die Anzahl der gezählten Aktionen wird erhöht.
    stepCounter++;

    //Nach allen Aktionen wird die Funktion GiveRewards() aufgerufen, diese verteilt die verschiedene Belohnungen
    GiveRewards();
}

//Die Funktion GiveRewards() wird nach den Aktionen des Agents aufgerufen und verteilt die Belohnungen
void GiveRewards()
{

    //Belohnung/Bestrafung je nach Abstand zum Fremdkörper
    //Abfrage ob der Abstand zum Fremdkörper, nach der Aktion des Agents, kleiner oder größer geworden ist
    if (CoinAbstandNACHaction < CoinAbstandVORaction)
    {
        AddReward(0.001f);  //Bei kleiner werden des Abstands: Belohnung von 0.001
    }
    else if (CoinAbstandNACHaction > CoinAbstandVORaction)
    {
        AddReward(-0.001f);     //Bei größer werden des Abstands: Bestrafung von -0.001
    }


    //Belohnung wenn sich der Fremdkörper zwischen den beiden Zangenzähnen befindet
    if (MitteTouched == true)
    {
        AddReward(0.00125f);         //Belohnung von 0.125
        AnzahlInDerMitte++;         //Zähler für die Anzahl in der Mitte wird erhöht.
        AnzahlInDerMitteForCoin++;  //Zähler für die Anzahl in der Mitte wird erhöht.
    }


    //Belohnunge bei jeder Art von Berührung zwischen Zange und Fremdkörper
    if (Zange1SideTouched == true)
    {
        AddReward(0.0025f);         //Belohnung von 0.25
        BeruhrungJederArt++;        //Zähler für Berührungen jeder Art wird hochgesetzt
    }
    if (Zange2SideTouched == true)
    {
        AddReward(0.0025f);         //Belohnung von 0.25
        BeruhrungJederArt++;        //Zähler für Berührungen jeder Art wird hochgesetzt
    }


    //Belohnugen bei der Berührung der Zangeninnenseite 1 von 2
```

```
        if (Zange1Touched)
        {
            AddReward(0.005f);          //Belohnung von 0.005
            BeruhrungDerInnenSeite++;   //Zähler für Berührungen der Innenseite hochsetzten
        }
        //Belohnugen bei der Berührung der Zangeninnenseite 2 von 2
        if (Zange2Touched)
        {
            AddReward(0.005f);          //Belohnung von 0.005
            BeruhrungDerInnenSeite++;   //Zähler für Berührungen der Innenseite hochsetzten
        }


        //Belohnung bei Griff durch beide Zangenzähne (Ermittelt über die boolschen Werte)
        if (Zange1Touched == true && Zange2Touched == true)
        {
            AddReward(0.0075f);             //Belohnung von 0.0075
            AnzahlGegriffen++;              //Zähler hochsetzten
            holdingTime += Time.deltaTime;      // Die Zeit in der der Fremdkörper gehalten wird, wird gemessen

            //Sobald der Fremdkörper 1 sec gehalten wird
            if (holdingTime > 1f)
            {
                AddReward(1);               //Belohnung von 1
                EndEpisode();               //Erfolgreichen Versuch beenden und ein neuer wird gestartet
                AnzahlGehalten++;               // Zähler hochsetzten
                holdingTime = 0;                // Gemessene Zeit zurücksetzten
            }
        }

        //Wir ermitteln noch mit einem anderen Verfahren ob der Fremdkörper gegriffen wurde: wir messen den Abstand zwischen den beiden Zangenzähnen und der Münze.
        distanceZahn1ToCoin = (Zahn1.transform.position - Coin.transform.position).magnitude;   //Abstand zwischen Zangenzahn 1 und dem Fremdkörper
        distanceZahn2ToCoin = (Zahn2.transform.position - Coin.transform.position).magnitude;   //Abstand zwischen Zangenzahn 2 und dem Fremdkörper

        //Belohnung bei Griff durch beide Zangenzähne (Ermittelt über den Abstand zwischen Fremdkörper und Zangenzähnen)
        //Wenn Abstand zwischen Fremdkörper und den beiden Zanganzähnen kleiner als 0.135 gilt der Fremdkörper als gegriffen
        if (distanceZahn1ToCoin < 0.135f && distanceZahn2ToCoin < 0.135f)
        {
            AddReward(0.0075f);             //Belohnung von 0.0075
            AnzahlGegriffen++;              //Zähler erhöhen
            holdingTime += Time.deltaTime;      //Zeit messen in der gehalten wird

            Zange1Touched = true;   //Der Boolesche Wert wird hier auf True gesetzt da dieser auch als Observation genutzt wird
            Zange2Touched = true;   //Der Boolesche Wert wird hier auf True gesetzt da dieser auch als Observation genutzt wird

            //Sobald der Fremdkörper 1 sec gehalten wird
            if (holdingTime > 1f)
            {
                AddReward(1);               //Belohnung von 1
                EndEpisode();               //Erfolgreicher Versuch wird beendet und ein neuer wird gestartet
                AnzahlGehalten++;               //Zähler hochsetzten
                holdingTime = 0;                //Gemessene Zeit zurücksetzten
            }
        }

        //Zuletzt werden die genutzten Zähler in der Console ausgegeben
        Debug.Log("Berührungen der jeder Art: " + BeruhrungJederArt);
        Debug.Log("Berührungen der einzelnen Innenseite: " + BeruhrungDerInnenSeite);
        Debug.Log("In der Mitte gewesen: " + AnzahlInDerMitte);
        Debug.Log("Gegriffen mit beiden Zähnen: " + AnzahlGegriffen);
        Debug.Log("Gehalten mit beiden Zähnen für 1s: " + AnzahlGehalten);
    }

    //Die Funktion OnCollisionEnter() wird aufgerufen sobald eine Kollision stattfindet
    void OnCollisionEnter(Collision collision)
    {
        //Abfrage ob das Objekt, mit dem kollidiert wurde, der Körper ist
        if(collision.gameObject.name == "ÖsophagusEingang")
        {
            AddReward(-1);  //Falls ja, Bestrafung von -1
            EndEpisode();   //Ende des fehlgeschlagenen Versuchs und Start eines neuen Versuchs
        }
    }

    //Die Funktion SpawnCoinAtRandomPositionInMesh() wird aufgerufen sobald ein neuer Versuch gestartet wird und positioniert den Fremdkörper neu und passt seine Rotation an
    void SpawnCoinAtRandomPositionInMesh(GameObject gameObjectWithMesh)
    {
        //POSITION DES FREMDKÖRPERS
       //Das Prinzip: 1. Es wird ein zufälliger Vektor bestimmt 2. Die Länge des Vektors wird bestimmt 3. Der Fremdkörper wird mit diesem Vektor verschoben 4. Die Rotation wird angepasst
        Vector3 RandomVector = Random.insideUnitSphere.normalized;                  // Zufälliger Vektor wird erstellt
        RandomVector.z = Mathf.Clamp(RandomVector.z, 0, 1);                         // z-wert des Vektors soll positiv (0- 1) bleiben, um nicht in den Agent hinein zu zeigen
```

```
    //Die Länge des Vektor hängt entweder davon ab wie viele Aktioen bereits getätigt wirden ODER wie oft das Ziel erreicht wurde
    //float increasingDistance = (coinSpawnRadius / ZyklusCount) * stepcounter              //(Radius / Teilungen) * Schritt
    float increasingDistance = (coinSpawnRadius / ZyklusCount) * (AnzahlInDerMitteForCoin);     //(Radius / Teilungen) * Erfolge
    Vector3 ThisVector = RandomVector * increasingDistance;                         //Die bestimmte Länge wird auf den Vektor angewnadt
    ThisVector = CoinStartLocalRotation * ThisVector;                               //Der Vektor wird an die Rotation des Fremdkörpers angepasst (Entwicklungszwecke)
    Coin.transform.position = coinStartPosition + ThisVector;                       //Die Position des Fremdkörpers = Startposition + NeuerVektor


    //ROTATION DES FREMDKÖRPERS
    //Das Prinzip: Entweder rein zufällige Rotation ODER eine die von der Anzahl der Aktionen abhängt ODER eine die von der Anzahl der Erfolge abhängt
    Coin.transform.rotation = coinStartRotation;            //Zuvor kurz die Rotation des Fremdkörpers zurücksetzten
    //Coin.transform.rotation = Random.rotation;            //zufällige Rotation
    //coinRotationAngle = ((float)360 / ZyklusCount) * stepcounter; //Abhängig von Anzahl der Aktionen       (360/Teilung) * Aktionen
    coinRotationAngle = ((float)360 / ZyklusCount) * AnzahlInDerMitteForCoin;   //Abhängig von Anzahl der Erfolge      (360/Teilungen) * Erfolge
    Coin.transform.Rotate(coinRotationAngle, coinRotationAngle, coinRotationAngle); //Der neu Bestimmte Rotationwinkel


    //ZURÜCKSETZTEN DES INTERVALLS
    // Falls es mehr Erfolge als Teilungen gab
    if (AnzahlInDerMitteForCoin > ZyklusCount)
    {
        AnzahlInDerMitteForCoin = 0;          //Erfolgszähler wird zurückgesetzt
        coinSpawnRadius += 1;                 //Der mögliche Radius erhöht sich
        ZyklusCount += ZyklusCount;           //Die Teilungen müssen passend erhöht werden
        if (coinSpawnRadius > 5)              //Falls der Radius größer als 5 ist
        {
            coinSpawnRadius = 1;              //Radius wieder verkleinern
            ZyklusCount = 10;                 //Die Teilungen anpassen
        }
    }
  }
}
```

Quellcode 3 EndoscopeAgentScript.cs, Quelle: Eigene Programmierung

4.2.1 Observationen

Die Observationen dienten dem Agent als Beobachtung der Umgebung. Diese konnten verschiedene Datentypen enthalten, wie z.B. boolesche Werte, float Werte, Vektoren aber auch Kamerabilder der 3D-Szene. Innerhalb der Simulation werden, neben dem Kamerabild, verschiedene Observationen unterschieden. Dazu gehören Observationen, die sich auf den Agent beziehen, Observationen die Informationen über den Fremdkörper vermitteln und Observationen, die sich auf die Interaktion mit dem Fremdkörper beziehen.
Während der Entwicklung wurden folgenden Observationen hinzugefügt, die den Agent über seine eigene Ausgangslage informierten:

- Der Vektor zwischen Start- und aktueller Position des Agents (Vector3)
- Die Rotation des Agents (float)
- Der Winkel der Gelenkbiegung (float)
- Der Winkel der Rotation des Zangenträgers (float)
- Der x-Wert der lokalen Position des ersten Zangenzahns (float)
- Der x-Wert der lokalen Position des zweiten Zangenzahns (float)

Diese wurden mit Observationen ergänzt, die Aufschlüsse über die Position des Fremdkörpers geben:

- Der Vektor zwischen Fremdkörper und Zangenträger (Vector3)

- Die Streckenlänge zwischen Fremdkörper und Zangenträger (float)

Des Weiteren erhielt der Agent Observationen, die Informationen über die Interaktion mit dem Fremdkörper enthielten:

- Berührung der Innenseite des Zangenzahn 1 (bool)
- Berührung der Innenseite des Zangenzahn 2 (bool)
- Positionierung des Fremdkörpers zwischen Zangenzähnen (bool)

Die Observations werden in der Funktion CollectObservations() (Quellcode 4) verarbeitet, die sich im C#-Skript EndoscopeAgentScript.cs befindet. Die Funktion wurde kommentiert und folgend dargestellt.

```
    // In der Funktion CollectObservations() können wir definieren welche Teile der 3D-Szene, dem Agent als Observationen dienen (z.B. Vektoren
oder Winkel)
    public override void CollectObservations(VectorSensor sensor)
    {
        //Observationen zum Agent selbst
        sensor.AddObservation(beginningPosition - this.transform.position);  // Vektor zwischen Start- und aktueller Position   (Vector3)
        sensor.AddObservation(this.transform.rotation.y);                    // Die Rotation des Agents                          (float)
        sensor.AddObservation(pivotRotation);                                // Die Winkel der Gelenkbeugung                     (float)
        sensor.AddObservation(CapsuleZangenTraeger.transform.localRotation.y);  // Der Winkel der Rotation des Zangenträgers      (float)
        sensor.AddObservation(Zahn1.transform.localPosition.x);              // Die lokale Position des ersten Zangenzahn        (float)
        sensor.AddObservation(Zahn2.transform.localPosition.x);              // Die lokale Position des zweiten Zangenzahn       (float)

        //Observationen zu Berührungen
        sensor.AddObservation(Zange1Touched);       //Berührung der Innenseite von Zangenzahn 1                                 (bool)
        sensor.AddObservation(Zange2Touched);       //Berührung der Innenseite von Zangenzahn 2                                 (bool)
        sensor.AddObservation(MitteTouched);        //Berührung der Mitte zwischen den Zangenzähnen                             (bool)

        //Observationen zur Münze
        sensor.AddObservation(Coin.transform.position - aeusesterPunkt.transform.position);     //Vektor zwischen Fremdkörper und Zangenträger
(Vector3)
        sensor.AddObservation(CoinAbstandVORaction);                                            //Abstand zwischen Fremdkörper und Zangenmitte
(float)
    }
```

Quellcode 4 Funktion CollectObservations() in EndoscopeAgentScript.cs, Quelle: Eigene Programmierung

4.2.1.1 Convolutional neural network der visuellen Observationen

Neben den oben genannten Observationen erhielt der Agent ebenfalls ein gerendertes Kamerabild der 3D-Szene. Dieses hatte eine Auflösung von 100 x 80 Pixeln und wird als RGB-Bild gerendert. Das Bild wird, über die von Unitys ML-Agents bereitgestellten Convolutional neural networks verarbeitet, die in verschieden Implementierungen vorliegen.[25]

.

[25] Vgl. Unity: Nutzung Convolutional Neural Networks , in: Unity ML-Agents Github,16.08.2020 [online] https://github.com/Unity-Technologies/ml-agents/blob/4261b4bf15e9d394b5d2426bc238d92940ab7e7d/docs/ML-Agents-Overview.md#learning-from-cameras-using-convolutional-neural-networks [18.08.20]

4.2.2 Aktionen

Die Aktionen ermöglichten es dem Agent, innerhalb der 3D-Szene Änderungen vorzunehmen. Nachdem der Agent die Observationen gesammelt hatte, wurde innerhalb der EndoscopeAgentScript.cs, die Funktion OnActionReceived() aufgerufen. In dieser Funktion wurden die Aktionen des Agents weiterarbeitet und auf die 3D-Szene angewandt. Dabei sind die Aktionen, fließende float-Werte zwischen -1 und +1. So konnten die Aktionen des Agents genutzt werden, um dem 3D-Endoskop seine Funktionalitäten zu geben, indem die Aktionen des Agents auf die Komponenten des 3D-Endoskops angewandt wurden. Dadurch konnten die Aktionen des Agents z.B. eine Bewegung oder Rotation des Endoskops durchführen.
So wurden während der Entwicklung folgende Aktionen des Agents verwendet, die das Endoskop um seine Funktionen erweiterten:

- Vor- und Zurückbewegung des Endoskops
- Rotation des Endoskops um die eigene Achse
- Biegung des Gelenks nach oben und unten
- Rotation des Zangenträgers um die eigene Achse
- Öffnen und Schließen der Zange

Die Funktion OnActionReceived() (Quellcode 5) wird im Folgenden dargestellt und enthält die Aktionen des Agents, sowie deren Weiterverarbeitung für die 3D-Szene. Darüber hinaus besitzt die Funktion noch weitere, kleinere Bestandteile, welche im weiteren Verlauf erläutert werden

```
//In der Funktion OnActionReceived() werden die Aktionen des Agents verarbeitet
    public override void OnActionReceived(float[] vectorAction)
    {
        //Noch bevor die Aktionen durchgeführt werden, wird der Abstand zum Fremdkörper gemessen, um diesen nach der Aktion zu vergleichen
        CoinAbstandVORaction = (aeusesterPunkt.transform.position - Coin.transform.position).magnitude;      //Länge des Vektors zwischen Rand des Zangenträger und Fremdkörper

        //------------------------------AKTIONEN-------------------------------------//
        //AKTION 1: Vor- und Zurückbewegung des Agents / Endoskops
        float MovementAction = vectorAction[0];             //Die Aktion des Agents wird einem Float zugewiesen
        AgentsRigidbody.AddForce( MovementAction * AgentsRigidbody.transform.up * movementSpeed);  //Die Aktion wirkt eine Kraft auf den Physik-Rigidbody des Agents und verschiebt ihn nach vorn

        //AKTION 2: Rotation des Agents / Endoskops
        float RotationAction = vectorAction[1];             //Die Aktion des Agents wird einem Float zugewiesen
        transform.Rotate(0, RotationAction * rotationSpeed, 0); //Die Aktion führt zu einer Rotation des Agents

        //AKTION 3: Biegen des Gelenks
        float BendingAction = vectorAction[2];              //Die Aktion des Agents wird einem Float zugewiesen
        //Wir gehen nun alle Gelenkpunkte des Gelenkes durch und wirken die Aktion mit einer Rotation darauf aus
        foreach (GameObject pivot in allPivots)             //Alle Gelenkpunkte des Gelenks werden durchgegangen
        {
            pivotRotation = pivot.transform.localRotation;                      //Zwischenzuweisung des aktuellen Winkels des Gelenkpunkts
            pivotRotation.z += BendingAction * BendingAngleSpeed * Time.deltaTime;  //Die Aktion führt eine Rotation des Gelenkpunktes durch (Z-Achse)
            pivotRotation.z = Mathf.Clamp(pivotRotation.z, -0.2f, 0.2f);            //Die mögliche Rotation der Gelenkpunkte wird zwischen -0.2 und 0.2 gehalten (Radiant)
            pivot.transform.localRotation = pivotRotation;                          //Zuweisung des veränderten Winkels auf die Rotation des Gelenkpunktes
        }

        //AKTION 4: Zangenträger rotieren
        float ZangenRotationAction = vectorAction[3]; //Zuweisung der Aktion des Agents
        CapsuleZangenTraeger.transform.Rotate(0, ZangenRotationAction * -ZangenRotationSpeed, 0, Space.Self); //Die Aktion führt eine Rotation des Zangenträgers aus

        //AKTION 5: Zange schließen und öffnen
        float ZangenOpeningAction = vectorAction[4]; //Aktion des Agents als float
        Vector3 localRight = Zahn1.transform.worldToLocalMatrix.MultiplyVector(Zahn1.transform.right);  //Hilfsvektor der die Richtung der Bewegung der Zangenzähne trägt
        Zahn1.transform.Translate((-localRight * ZangenOpeningAction) * openingSpeed);  //Aktion führt Bewegung in Richtung Hilfsvektor*-1 aus
        Zahn2.transform.Translate((localRight * ZangenOpeningAction) * openingSpeed);   //Aktion führt Bewegung in Richtung Hilfsvektor aus
        //Diese IF-Statements sind Limits für die Zangenzahn-Bewegung
        if (Zahn1.transform.localPosition.x < -0.2f) { Zahn1.transform.localPosition = new Vector3(-0.2f, 1.5f, 0); }
        if(Zahn1.transform.localPosition.x > 0) { Zahn1.transform.localPosition = new Vector3(0, 1.5f, 0); }
        if (Zahn2.transform.localPosition.x > 0.2f) { Zahn2.transform.localPosition = new Vector3(0.2f, 1.5f, 0); }
        if (Zahn2.transform.localPosition.x < 0) { Zahn2.transform.localPosition = new Vector3(0, 1.5f, 0); }

        //------------------------------ENDE DER AKTIONEN-----------------------------------//

        //Nach allen Aktionen des Agents wird nochmal der Abstand zum Fremdkörper ermittelt
        CoinAbstandNACHaction = (aeusesterPunkt.transform.position - Coin.transform.position).magnitude;

        //Die Anzahl der gezählten Aktionen wird erhöht.
        stepCounter++;

        //Nach allen Aktionen wird die Funktion GiveRewards() aufgerufen, diese verteilt die verschiedene Belohnungen
        GiveRewards();
    }
```

Quellcode 5 Funktion OnActionReceived() in EndoscopeAgentScript.cs, Quelle: Eigene Programmierung

.

4.3 Training

4.3.1 Simulationsaufbau während des Trainings

Die Simulation bestand während des Trainings aus einer Kombination aus den zuvor beschriebenen Objekten: 3D- Körper, 3D-Fremdkörper und 3D-Endoskop als Unity-ML-Agent. Hierbei wurden das Endoskop und der Fremdkörper innerhalb des Magens (Abb. 11

& Abb. 12) platziert, wobei sich der Fremdkörper, zum Start des Trainings, direkt vor dem Endoskop befindet.

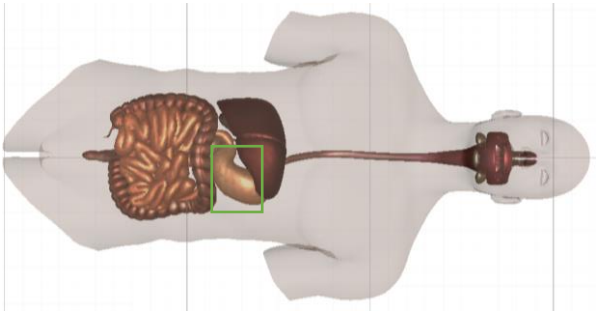


Abb. 11 Ortung des Magens, Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

Abb. 12 Endoskop und Fremdkörper innerhalb des Magens, Quelle: Eigene Darstellung / Unity Bildschirmaufnahme

4.3.2 Aufgabe

Die Aufgabe des Agents war, unter Anwendung der notwendigen Aktionen, das Endoskop so zu steuern, dass es sich dem Fremdkörper nähert und diesen anschließend, mit beiden Zangenzähnen greift. Dies konnte durch verschiedene Kombinationen der Aktionen, erreicht werden. Bei einer Schräglage des Fremdkörpers konnte es zudem erforderlich sein, zuerst ein Manöver auszuführen, das den Fremdkörper so ausrichtete, sodass er anschließend gegriffen werden konnte. Die Aufgabe und der Versuch wurden, vom Agent, erfolgreich beendet, wenn der Fremdkörper, mit den beiden Innenseiten der zwei Zangenzähnen, berührt wurde. Der einzelne Versuch wurde wegen Fehlschlagens abgebrochen, sobald die Magenwand berührt wurde oder die maximale Anzahl an 2500 Aktionen erreicht wurde. Nach einem erfolgreichen oder fehlgeschlagenen Versuch wurde dieser beendet und ein neuer Versuch wurde gestartet, solange bis die zuvor bestimmte maximale Anzahl an

Aktionen erreicht wurde (100.000-5.000.000) und das Training beendet wurde. Der Fremdkörper ändert dabei, bei jedem neuen Versuch, seine Position und Rotation, um so den Schwierigkeitsgrad der Aufgabe und die Varianz zu erhöhen.

4.3.2.1 Positionierung des Fremdkörpers

Zur Positionierung des Fremdkörpers, während des Training und der Tests, wurde eine Methode gewählt, welche die Entfernung zwischen Startposition des Endoskops und Position des Fremdkörpers stetig erhöht und den Fremdkörper dabei in verschiedensten Richtungen positioniert. So konnte sich der Fremdkörper, zum Start des Trainings, direkt vor dem Endoskop befinden, um während des Trainings stetig weiter weg verschoben zu werden.

Zur Steigerung der Entfernung wurde ein Radius bestimmt, innerhalb dessen der Fremdkörper platziert werden konnte (Abb. 13a). Darauffolgend wurde die Schrittgröße berechnet, mit der die Entfernung gesteigert werden sollte, dazu wurde prinzipiell die Berechnung Schrittgröße = Radius / Anzahl der Teilungen durchgeführt (Abb. 13a).

Die Erhöhung der Entfernung konnte entweder von der Anzahl bereits getätigter Aktionen oder von der Anzahl der erfolgreichen Versuche abhängen. So wurde die Entfernung z.B. jedes Mal um eine Schrittgröße erhöht, wenn das Ziel erreicht wurde.

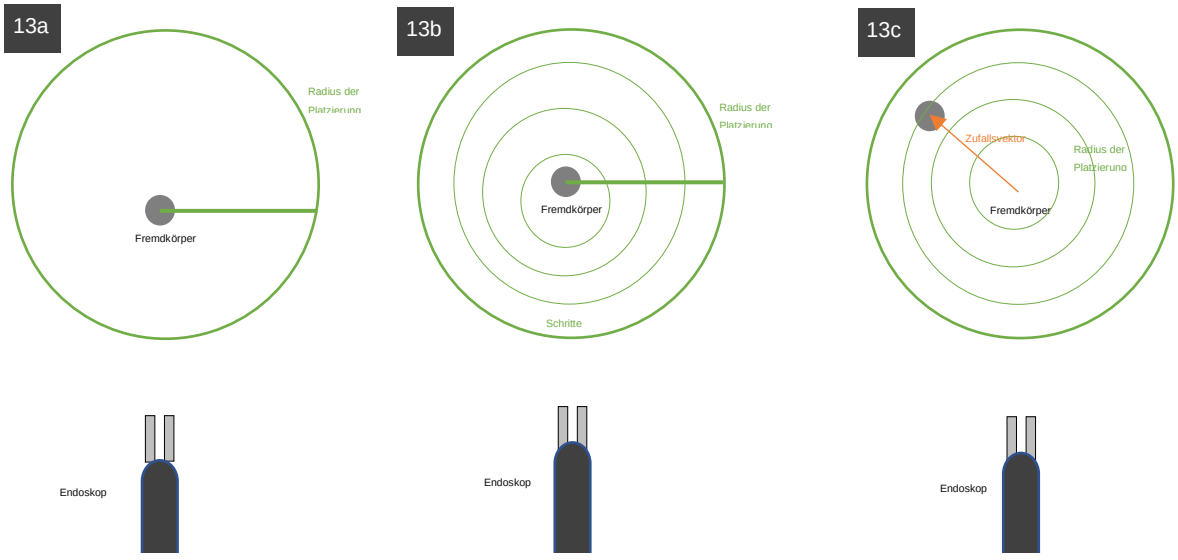


Abb. 13 a-c Positionierung des Fremdkörpers, Quelle: Eigene Darstellung

Nachdem die neue Entfernung ermittelt wurde, musste anschließend die Richtung, in welcher der Fremdkörper liegt, bestimmt werden. Dazu wurde die bereits in Unity implementierte und zufällige Funktion Random. Vector3 genutzt, die einen zufälligen Vektor

zurückgibt. Dieser Zufallsvektor erhielt anschließend die zuvor berechnete Entfernung (Abb. 13c).
Nach dem die Entfernung und die Richtung des Fremdkörpers gesetzt wurden, galt es den Fremdkörper um diesen Vektor zu verschieben (Abb. 13c).
Daraufhin wurde die Rotation des Fremdkörpers angepasst, diese wurde entweder zufällig angepasst oder hing ebenfalls von den Erfolgen des Agents ab.
Zuletzt wurde festgelegt, was passieren soll, falls mehr Erfolge erreicht wurden als Teilungen existieren. In diesem Falle wird der Radius bei gleicher Schrittgröße verdoppelt, um den Bereich, in dem ein Fremdkörper positioniert werden kann, zu vergrößern.
Im Folgenden ist die Funktion SpawnCoinAtRandomPositionInMesh() (Quellcode 6), die sich innerhalb des C#-Skripts EndoscopeAgentScript.cs befindet, dargestellt. Diese Funktion setzt die zuvor beschriebe Positionierung des Fremdkörpers um.

```
//Die Funktion SpawnCoinAtRandomPositionInMesh() wird aufgerufen sobald ein neuer Versuch gestartet wird und positioniert den Fremdkörper neu und passt seine Rotation an
    void SpawnCoinAtRandomPositionInMesh(GameObject gameObjectWithMesh)
    {
        //POSITION DES FREMDKÖRPERS
       //Das Prinzip: 1. Es wird ein zufälliger Vektor bestimmt 2. Die Länge des Vektors wird bestimmt 3. Der Fremdkörper wird mit diesem Vektor verschoben 4. Die Rotation wird angepasst
        Vector3 RandomVector = Random.insideUnitSphere.normalized;                          // Zufälliger Vektor wird erstellt
        RandomVector.z = Mathf.Clamp(RandomVector.z, 0, 1);                                  // z-wert des Vektors soll positiv (0- 1) bleiben, um nicht in den Agent hinein zu zeigen

        //Die Länge des Vektor hängt entweder davon ab wie viele Aktioen bereits getätigt wirden ODER wie oft das Ziel erreicht wurde
        //float increasingDistance = (coinSpawnRadius / ZyklusCount) * stepcounter                    //(Radius / Teilungen) * Schritt
        float increasingDistance = (coinSpawnRadius / ZyklusCount) * (AnzahlInDerMitteForCoin);      //(Radius / Teilungen) * Erfolge
        Vector3 ThisVector = RandomVector * increasingDistance;                                     //Die bestimmte Länge wird auf den Vektor angewnadt
        ThisVector = CoinStartLocalRotation * ThisVector;                                            //Der Vektor wird an die Rotation des Fremdkörpers angepasst (Entwicklungszwecke)
        Coin.transform.position = coinStartPosition + ThisVector;                                   //Die Position des Fremdkörpers = Startposition + NeuerVektor


        //ROTATION DES FREMDKÖRPERS
        //Das Prinzip: Entweder rein zufällige Rotation ODER eine die von der Anzahl der Aktionen abhängt ODER eine die von der Anzahl der Erfolge abhängt
        Coin.transform.rotation = coinStartRotation;                //Zuvor kurz die Rotation des Fremdkörpers zurücksetzten
        //Coin.transform.rotation = Random.rotation;                //zufällige Rotation
        //coinRotationAngle = ((float)360 / ZyklusCount) * stepcounter; //Abhängig von Anzahl der Aktionen         (360/Teilung) * Aktionen
        coinRotationAngle = ((float)360 / ZyklusCount) * AnzahlInDerMitteForCoin;     //Abhängig von Anzahl der Erfolge (360/Teilungen) * Erfolge
        Coin.transform.Rotate(coinRotationAngle, coinRotationAngle, coinRotationAngle); //Der neu Bestimmte Rotationwinkel


        //ZURÜCKSETZTEN DES INTERVALLS
        // Falls es mehr Erfolge als Teilungen gab
        if (AnzahlInDerMitteForCoin > ZyklusCount)
        {
            AnzahlInDerMitteForCoin = 0;            //Erfolgszähler wird zurückgesetzt
            coinSpawnRadius += 1;                   //Der mögliche Radius erhöht sich
            ZyklusCount += ZyklusCount;             //Die Teilungen müssen passend erhöht werden
            if (coinSpawnRadius > 5)                //Falls der Radius größer als 5 ist
            {
                coinSpawnRadius = 1;                //Radius wieder verkleinern
                ZyklusCount = 10;                   //Die Teilungen anpassenn
            }
        }
    }
```

Quellcode 6 Funktion SpawnCoinAtRandomPositionInMesh() in EndoscopeAgentScript.cs, Quelle: Eigene Programmierung

Eine derartige Konzeption der Methodik ermöglicht einen stetig zunehmenden Schwierigkeitsgrad für einen erfolgreichen Abschluss der gestellten Aufgabe. Durch die Regulierung der Schrittgröße konnte beeinflusst werden, wie schnell sich der Fremdkörper entfernt und so die Schwierigkeit, beim nächsten Versuch, erhöht. Außerdem bietet dieser Aufbau den Vorteil einer hohen Varianz, da neben der Entfernung zum Fremdkörper, auch die Richtung und Rotation des Fremdkörpers verändert wird

4.3.3 Zielverhalten zur Lösung der Aufgabe

Zum erfolgreichem Lösen der Aufgabe war es notwendig die verfügbaren Aktionen miteinander zu kombinieren. So sollte sich das Endoskop dem Fremdkörper nähern, indem

die Vor- und Zurückbewegung mit der Rotation und der Biegung der Gelenke kombiniert wurde. Nach der Annäherung gilt es in den meisten Fällen ein Manöver auszuführen, dass den Fremdkörper so ausrichtet, dass dieser anschießend gegriffen werden kann. Diese Manöver sind so vielfältig wie die Positionen und Rotationen des Fremdkörpers und müssen während des Versuchs, vom Agent, improvisiert werden. Dies kann durch viele unterschiedliche Kombination der Aktionen und Berührungen des Fremdkörpers geschehen.

Nach der Ausrichtung des Fremdkörpers, gilt es das Endoskop und den Fremdkörper in eine Lage zu bringen, die es erlaubt dem Fremdkörper zu greifen. Nachdem der Fremdkörper ausgerichtet wurde und das Endoskop in die entsprechende Lage gebracht wurde, muss sich das Endoskop weiter dem Fremdkörper nähern, bis sich zwischen den beiden Zangenzähnen befindet. In dieser Position muss abschließend zugriffen werden, um den Fremdkörper mit beiden Zangenzähnen zu greifen.

Außerdem gilt es für den Agent, den Kontakt und die Kollision mit der Magenwand zu vermeiden.

4.3.4 Algorithmen des Trainings

Ein Teilbereich der künstlichen Intelligenzen ist das Machine Learning , das sich in drei weitere Teilbereiche und Trainingsmethoden untergliedern lässt, das supervised learning (deutsch: überwachtes Lernen), das unsupervised learning (deutsch: unüberwachtes Lernen) und das Reinforcement Learning

Das Reinforcement Learning stützt sich auf das aufeinanderfolgende Treffen von Entscheidungen. Hierbei erhält der Agent verschiedene Sensoren, die ihm Informationen über seine Umgebung vermitteln. Diese Informationen werden verarbeitet und dienen als Entscheidungsgrundlage für die Aktion, die der Agent als nächstes ausführt. So trifft der Agent stetig Entscheidungen auf Grundlage der Umgebung.

Dies hat das Bestreben, *policy* zu erlernen, die als Funktion, von den sensorischen Observationen zu den Aktionen dient. Zuletzt werden die Aktionen des Agents belohnt oder bestraft, um die Richtline dahingehend zu trainieren, die Belohnungen zu maximieren. Diese Belohnungen können genutzt werden, um den Agent darüber aufzuklären wie gut die Aktionen waren.

Das Lernen mittels Reinforcement Learning beinhaltet eine Trainings- und eine Inferenzphase. In der Trainingsphase wird die Policy erlernt, während in der Inferenzphase

die gelernte Strategie genutzt wird ,um in freien Tests, Observationen und Aktionen zu verarbeiten. [26]

Um dem Agent das erwünschte Zielverhalten beizubringen, wurden verschiedene Lernmethoden des Reinforcement Learning, wie den PPO Algorithmus und die Algorithmen des BC.

4.3.4.1 Proximal Policy Optimization

Die Unity-ML-Agents nutzen eine Methode des Reinforcement Learning mit dem Name Proximal Policy Optimization. Bei der Nutzung des PPO Algorithmus, nähert sich das neuronale Netz der optimalen Funktion und Policy an, welche die bestmögliche Aktion für die jeweiligen Observationen bestimmt. Der PPO Algorithmus ist in Tensorflow enthalten und wird in einem eigenen Python-Prozess gestartet. [27] Dieser Algorithmus ist das Standardverfahren für Unity-ML-Agents, da dieser sehr stabil und vielseitig einsetzbar ist.[28] Weitere Vorteile des Algorithmus‘ sind die einfache Implementierung und Anpassung.[29]

Bei der Nutzung des Algorithmus zu Trainingszwecken, können verschiedene Parameter bestimmt werden, die den Aufbau und die Einstellungen des neuronalen Netzes beeinflussen. Dazu gehören z.B. die Anzahl der Netzwerkschichten (englisch: numbers of layers), die Anzahl der Neuronen (englisch: hidden_Units) oder die Lernrate (englisch: learning rate). Im Folgenden sind die Parameter des Trainings, mit dem PPO Algorithmus‘, aufgeführt und in Kürze erläutert:

- trainer_type : Lernmethode, Proximal Policy Optimization oder Soft Actor-Critic (SAC)
- summary_freq : Nach dieser Anzahl an Erfahrungen (Observation, Aktion und Belohnung) werden Statistiken zum Training ausgegeben
- time_horizon : Die Anzahl der Erfahrungen, die gesammelt werden, bevor diese in den Experience Buffer eingetragen werden.
- max_steps : Nach dieser Anzahl an Erfahrungen wird das Training beendet

---

[26] Vgl. Unity: Überblick zum Machine Learning, in Unity-ML-Agents Github,13.06.19,[online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Background-Machine-Learning.md [18.08.2020]

[27] Vgl. Unity: Proximal Policy Optimization, in Unity-ML-Agents Github,23.10.19,[online] https://github.com/Unity-Technologies/ml-agents/blob/3d7c4b8d3c1ad17070308b4e06bb57d4a80f9a0c/docs/Training-PPO.md [18.08.2020]

[28] Vgl. Unity: Unity-ML-Agents Toolkit, in Unity-ML-Agents Github,16.07.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/4261b4bf15e9d394b5d2426bc238d92940ab7e7d/docs/ML-Agents-Overview.md#deep-reinforcement-learning [18.08.2020]

[29] Vgl. OpenAi: Grundlinien Proximal Policy Optimization, in OpenAi,, 20.07.17,[online] https://openai.com/blog/openai-baselines-ppo/ [18.08.2020]

- keep_checkpoints : Die Anzahl an Kontrollpunkten (englisch: checkpoints), die während des Trainings erhalten bleiben
- checkpoint_interval : Nach dieser Anzahl an Erfahrungen wird ein neuer Kontrollpunkt erstellt.
- init_path : Dieser Pfad ermöglicht es, ein bereits trainiertes Model zu laden und weiterzutrainieren
- learning_rate : Die Lernrate für den Gradient Descent Algorithmus und entspricht der Schrittgröße bei jedem Update des Gradient descent.
- batch_size : Die Anzahl der Erfahrungen in jeder Iteration des Gradient Descent.
- buffer_size : Nach dieser Anzahl an Erfahrungen wird die policy aktualisiert.
- learning_rate_schedule : Bestimmt wie die Lernrate während des Trainings verläuft. Zum Beispiel gleichbleibend oder linear fallend.
- num_layers : Die Anzahl an Schichten die das neuronale Netz besitzt.
- normalize : Bestimmt ob die Observationen normalisiert werden.
- vis_encoder_type : Typ der Encoders zur Verarbeitung der visuellen Observationen.
- extrinsic -> strength: Der Faktor mit dem die Belohnungen multipliziert werden.
- extrinsic -> gamme: Abzinsungsfaktor (englisch: discount factor) für zukünftige Belohnungen der Umgebung[30]

4.3.4.2 Behavioral Cloning

Bei komplexen Aufgaben ist es möglich, dem Agent, mittels eigener Demonstrationen, das gewünschte Zielverhalten zu zeigen und dabei die Policy zu trainieren. Zur Ergänzung des PPO Algorithmus‘ kann das von Unity-ML-Agents bereitgestellte Behavioral Cloning genutzt werden. Diese Methode beeinflusst die Policy des Agents dahingehend, bereitgestellte Demonstrationen nachzuahmen.[31]

Auch hierzu können verschiedene Parameter bestimmt werden, die zum Beispiel bestimmen wie stark das Verhalten imitiert werden soll oder wie lange eine Imitation des Verhaltens stattfindet. Die Parameter für das BC, welche die Paramater des PPO Algorithmus erweitern, sind anschließend aufgeführt und in Kürze erläutert:

- demo_path : Der Pfad zur Demonstrations-Dateien

[30] Vgl. Unity: PPO Trainings-Parameter,in Unity-ML-Agents Github,18.08.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Configuration-File.md [18.08.2020]
[31] Vgl. Unity: Unity-ML-Agents Toolkit, in Unity-ML-Agents Github, 16.07.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md#behavioral-cloning-bc [18.08.20]

- strength : Die Lernrate im Verhältnis zur Lernrate des PPO Algorithmus‘. Gibt an wie stark der Einfluss auf die Policy ist.
- steps : Die Anzahl an Erfahrungen in denen das BC aktiv ist.
- batch_size : Die Anzahl der Erfahrungen in jeder Iteration des Gradient Descent.
- num_epoch : Anzahl an Durchläufen des Experience Buffers während des Gradient Descent
- samples_per_update: Die maximal Anzahl an Demonstrationen die während einer Aktualisierung der Imitation genutzt wird.[32]

4.3.4.3 Methode der Nutzung der Lernmethoden

Die beiden Algorithmen und Methoden PPO und BC sollten miteinander kombiniert werden, indem die PPO mit dem BC erweitert wird.

4.3.5 Extrinsische Belohnungen der Umgebung

Während des Trainings sollte der Agent eine Kombination der gegebenen Aktionen ausführen, um so das Ziel der Aufgabe zu erreichen. Um dem Agent das Zielverhalten beizubringen, was zum erfolgreichen Absolvieren der Aufgabe führt, ist es notwendig die richtigen Aktionen zu belohnen und die falschen zu bestrafen. Dies geschieht nachdem der Agent seine Aktionen durchgeführt hat und soll diese Aktionen und den neuen Zustand bewerten. So entstanden während der Entwicklungen verschiedene extrinsische Belohnungen und Bestrafungen, die folgend mir ihrer Begründung beschrieben sind:

- Falls der Abstand zwischen Fremdkörper und Endoskop kleiner wurde
  Grund: Die Bewegung des Endoskops in Richtung des Fremdkörpers sollte so belohnt werden
- Falls der Fremdköper von einer der beiden Zangenzähne berührt wurde
  Grund: Der einzelne Kontakt und auch die Manöver sollten so belohnt werden
- Falls der Fremdkörper sich, in der Mitte, zwischen den beiden Zangenzähnen befindet
  Grund: Der Zwischenzustand, der zum erfolgreichen Zugreifen notwendig ist, wurde somit belohnt
- Falls der Fremdköper von der Innenseite einer der beiden Zangenzähne berührt wurde

[32] Vgl. Unity: BC Trainings-Parameter, in Unity ML-Agents Github,18.08.20, https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Configuration-File.md [18.08.2020]

Grund: Diese Belohnung, sollte die Berührung mit der Innenseite belohnen und hervorheben und das Zugreifen anregen

- Falls der Fremdkörper von den Innenseiten beider Zangenzähne berührt wurde
  Grund: Das endgültige Zugreifen wurde somit belohnt und ein neuer Versuch wurde gestartet.

Zu den Belohnungen lässt sich zusammenfassend sagen, dass diese sowohl die Zwischenschritte als auch das endgültige Ziel belohnen.

Bestrafungen:

- Falls der Abstand zwischen Fremdkörper und Endoskop größer wurde
  Grund: Diese Bestrafung sollte ein zu fernes wegbewegen des Endoskops verhindern
- Wenn die Magenwand vom Zangenträger berührt wurde
  Grund: Dies hatte zum einen den Grund die Magenwand vor Verletzungen zu schützen und des Weiteren die Funktion den Versuch an dieser Stelle zu beenden, um einen neuen zu starten.

Die extrinsischen Belohnungen wurden in der Funktion GiveReward() (Quellcode 7) verteilt, welche sich im C#-Skript EndoscopeAgentScript.cs befindet. Neben den Belohnungen wurden innerhalb der Funktion verschiedene Zähler genutzt, die eine spätere Auswertung der Erfolge möglich machten.

```
    void GiveRewards()
    {

        //Belohnung/Bestrafung je nach Abstand zum Fremdkörper
        //Abfrage ob der Abstand zum Fremdkörper, nach der Aktion des Agents, kleiner oder größer geworden ist
        if (CoinAbstandNACHaction < CoinAbstandVORaction)
        {
            AddReward(0.001f);  //Bei kleiner werden des Abstands: Belohnung von 0.001
        }
        else if (CoinAbstandNACHaction > CoinAbstandVORaction)
        {
            AddReward(-0.001f);     //Bei größer werden des Abstands: Bestrafung von -0.001
        }


        //Belohnung wenn sich der Fremdkörper zwischen den beiden Zangenzähnen befindet
        if (MitteTouched == true)
        {
            AddReward(0.00125f);          //Belohnung von 0.125
            AnzahlInDerMitte++;         //Zähler für die Anzahl in der Mitte wird erhöht.
            AnzahlInDerMitteForCoin++;  //Zähler für die Anzahl in der Mitte wird erhöht.
        }


        //Belohnunge bei jeder Art von Berührung zwischen Zange und Fremdkörper
        if (Zange1SideTouched == true)
        {
            AddReward(0.0025f);           //Belohnung von 0.25
            BeruhrungJederArt++;        //Zähler für Berührungen jeder Art wird hochgesetzt
        }
        if (Zange2SideTouched == true)
        {
            AddReward(0.0025f);           //Belohnung von 0.25
            BeruhrungJederArt++;        //Zähler für Berührungen jeder Art wird hochgesetzt
        }


        //Belohnugen bei der Berührung der Zangeninnenseite 1 von 2
        if (Zange1Touched)
        {
            AddReward(0.005f);          //Belohnung von 0.005
            BeruhrungDerInnenSeite++;   //Zähler für Berührungen der Innenseite hochsetzten
        }
        //Belohnugen bei der Berührung der Zangeninnenseite 2 von 2
        if (Zange2Touched)
        {
            AddReward(0.005f);          //Belohnung von 0.005
            BeruhrungDerInnenSeite++;   //Zähler für Berührungen der Innenseite hochsetzten
        }


        //Belohnung bei Griff (in zwei versch. Ausführungen: 1. mit bools 2. mit Abständen)
        //Belohnung bei Griff durch beide Zangenzähne (Ermittelt über die boolschen Werte)
        if (Zange1Touched == true && Zange2Touched == true)
        {
            AddReward(0.0075f);                 //Belohnung von 0.0075
            AnzahlGegriffen++;                  //Zähler hochsetzten


        }

        //Wir ermitteln noch mit einem anderen Verfahren ob der Fremdkörper gegriffen wurde: wir messen den Abstand zwischen den beiden
Zangenzähnen und der Münze.
        distanceZahn1ToCoin = (Zahn1.transform.position - Coin.transform.position).magnitude;     //Abstand zwischen Zangenzahn 1 und dem
Fremdkörper
        distanceZahn2ToCoin = (Zahn2.transform.position - Coin.transform.position).magnitude;     //Abstand zwischen Zangenzahn 2 und dem
Fremdkörper

        //Belohnung bei Griff durch beide Zangenzähne (Ermittelt über den Abstand zwischen Fremdkörper und Zangenzähnen)
        //Wenn Abstand zwischen Fremdkörper und den beiden Zanganzähnen kleiner als 0.135 gilt der Fremdkörper als gegriffen
        if (distanceZahn1ToCoin < 0.135f && distanceZahn2ToCoin < 0.135f)
        {
            AddReward(0.0075f);                 //Belohnung von 0.0075
            AnzahlGegriffen++;                  //Zähler erhöhen

            Zange1Touched = true;   //Der Boolsche Wert wird hier auf True gesetz da dieser auch als Observation genutzt wird
            Zange2Touched = true;   //Der Boolsche Wert wird hier auf True gesetz da dieser auch als Observation genutzt wird


        }

        //Zuletzt werden die genutzten Zähler in der Console ausgegeben
        Debug.Log("Berührungen jeder Art: " + BeruhrungJederArt);
        Debug.Log("In der Mitte gewesen: " + AnzahlInDerMitte);
        Debug.Log("Berührungen der einzelnen Innenseite: " + BeruhrungDerInnenSeite);
        Debug.Log("Gegriffen mit beiden Zähnen: " + AnzahlGegriffen);
        Debug.Log("Gehalten mit beiden Zähnen für 1s: " + AnzahlGehalten);
    }
```

Quellcode 7 Funktion GiveReward() in EndoscopeAgentScript.cs, Quelle: Eigene Programmierung

4.4 Das Testen

Da die Trainingsumgebung sich zufällig veränderte, konnte diese ebenfalls genutzt werden, um den trainierten Agent zu testen. Ähnlich wie innerhalb des Trainings, wurde auch beim Testen des Agents, der Fremdkörper immer auf einer zufälligen, aber stetig schwerer werdenden Position platziert. Hierbei wurden verschiedene Zähler genutzt, um zu beobachten welche Teilziele der Agent tatsächlich erzielte. Hierzu wurde die Anzahl an Aktionen gezählt, in denen der Agent sich in einem bestimmten Zustand befindet oder eine bestimmte Aktion ausführt. Zu den Zählern gehören:

- Anzahl an Aktionen mit einer Berührung des Fremdkörpers (Berührungen jeder Art
- Anzahl an Aktionen mit einer Berührung des Fremdkörpers, durch die Innenseite einer der beiden Zangenzähne
- Anzahl an Aktionen in der sich der Fremdkörper zwischen den Zangenzähnen befand
- Anzahl an Aktionen in der der Fremdkörper gegriffen wurde

Die Höhe der Zähler wurde hier bei einer Gesamtzahl von 25000 Aktionen gemessen und in den Trainingsdokumentationen dokumentiert.
Die Erkenntnisse der Tests, trugen maßgeblich zur Weiterentwicklung des nächsten Agents bei. Besonders die verwendeten Zähler und die Beobachtungen der Verhaltensweise gaben Aufschluss darüber, welche Anpassungen getroffen werden müssen.

5 Ergebnisse

5.1 Trainingsdokumentation

Die Dokumentationen beinhalten eine chronologische Aufzeichnung von Trainingsversuchen, die den Verlauf und den Fortschritt der Entwicklung beschreiben. Die Dokumentation ist dabei in verschiedene Versuchsarten unterteilt, welche sich durch ihre Lernmethoden und Ziele unterscheiden. Hierbei enthält jeder Trainingsversuch eine tabellarische Darstellung der Ausgangslage. Diese enthält die Hyperparameter des Trainings, die Observationen und Aktionen des Agents, die extrinsischen Belohnungen und gegebenenfalls Änderungen, die an der Simulation selbst vorgenommen wurden.
Anschließend werden die von Tensorboard generierten Ergebnisdaten des Trainings abgebildet, welche Aufschluss über den Trainingsverlauf geben. Diese beinhalten Graphen, welche alle auf ihrer X-Achse die steigende Zahl der Erfahrungen zeigen und auf ihrer Y-Achse verschiedene, zu beobachtende Werte des Trainings darstellen.

Die zu beobachteten Werte, die durch die Graphen veranschaulicht werden, sind:

- Enviroment / Episode Length : Die zeitliche Länge des Versuchs
- Enviroment / Cumulative Reward : Die Höhe der Belohnungen innerhalb eines Versuchs
- Policy / Extrinsic Reward : Die Höhe der Belohnungen die von der Umgebung verteilt wurde. Im Falle eines einzigen Agents, ist dieser Graph mit dem zu vorigen identisch.
- Policy / Value Estimate : Der durchschnittliche Wert aller Zustände des Agents
- Policy / Entropy : Die Entropie ist ein Indikator dafür, wie zufällig die Entscheidungen des Agents getroffen werde.
- Policy / Learning Rate : Darstellung der Lernrate
- Losses / Policy Loss : Die durchschnittliche Höhe der policy-loss-function(Richtlinien-Verlusts-Funktion) Dies ist ein Indikator dafür wie stark die Veränderungen der Richtline sind.
- Losses / Value Loss : The mean loss of the value function update. Dies ist ein Indikator dafür wie sehr der Agent in der Lage ist, den Wert der einzelnen Zustände vorherzusehen.
- Losses/Pretraining: Die durchschnittliche Höhe des Imitations-Verlusts. Dies ist ein Indikator dafür, wie gut die Demonstrationen imitiert wurden.[33]

Nach den Tensorboard-Graphen folgen die Ergebnisse des Tests. Dazu gehören Zähler die verschiedene Zwischenzustände zählten und die beobachteten Verhaltensweises des Agents.

Als letztes folgen in den Versuchen die Erkenntnisse und die daraus resultierenden Änderungen. Diese Änderungen sind ebenfalls in der Ausgangslage des nächsten Versuchs, durch eine dickere Schrift hervorgehoben. Die Änderungen der Hyperparameter finden sich in der Ausganglage des nächsten Versuchs, ebenfalls durch eine dickere Schrift hervorgehoben.

Am Ende der Trainingsdokumentation werden die entstandenen Tensorboard-Graphen überlagert, um einen Vergleich zu ermöglich. Darauf folgt abschließend eine Verlaufstabelle, die eine zusammengefasste Form der Versuche enthält.

[33] Vgl. Unity: Nutzung von Tensorboard, in: Unity-ML-Agents Github,24.07.20,[online] https://github.com/Unity-Technologies/ml-agents/blob/2d0eb6147c031c082522eb683e569dd99b4d65fb/docs/Using-Tensorboard.md [25.08.20]

5.1.1 Training1_PPO_GrabCoin1

5.1.1.1 Ausgangslage

Tabelle 1 Ausgangslage des Training1_PPO_GrabCoin1, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 1024 buffer_size: 10240 learning_rate: 3e-4 beta: 5.0e-3 epsilon: 0.2 lambd: 0.95 num_epoch: 3 learning_rate_schedule: linear network_settings: normalize: false hidden_units: 128 num_layers: 2 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 500000 max_steps: 500000 time_horizon: 64 summary_freq: 50000 threaded: true	- Camera (100x80, greyscale) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als Vec3 Winkel - Biegung als Winkel float - Rotation des Zangenträgers als vec3 - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Kapsel-Spitze zu Münze als float	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	- Wenn der Abstand zur Münze verringert wird (+0.1) - Wenn der Abstand zur Münze größer wird (-0.1) - Wenn Zahn1 jeder Art berührt wird (+0.25) - Wenn Zahn2 jeder Art berührt wird (+0.25) - Wenn beide Zähne Innen die Münze berühren (1 & Neustart) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE)

5.1.1.2 Ergebnisdaten des Trainingsverlaufs

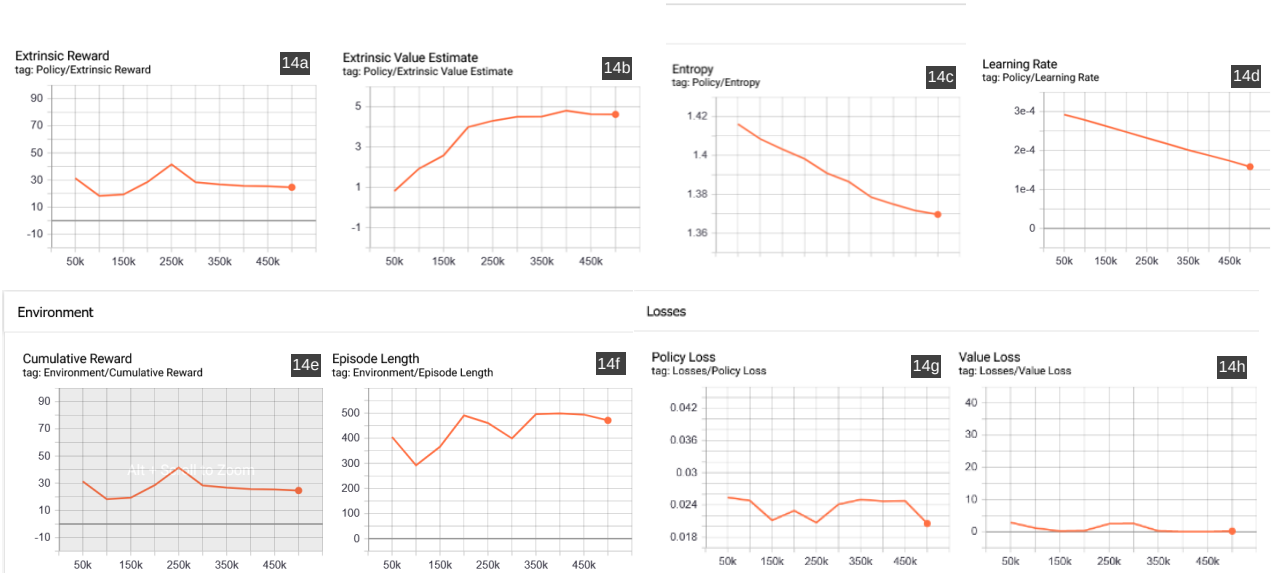


Abb. 14 a-h Trainingsdaten des Training1_PPO_GrabCoin1, Quelle: Tensorboard Bildschirmaufnahme

5.1.1.3 Ergebnisse der Tests

Während des Tests mit einer Dauer von 25000 Aktionen, gelang es dem Agent, während 43 Aktionen den Fremdkörper zu berühren, davon 22-mal mit der Innenseite eines Zangenzahns. Der Fremdkörper befand sich während elf Aktionen zwischen den beiden Zangenzähen, wurde aber kein einiges mal gegriffen.

Der Agent bewegte sich ruckartig und langsam vorwärts, unabhängig davon ob ein Kontakt mit dem Fremdkörper stattgefunden hat oder dieser sogar schon wieder beendet wurde. Andere Aktionen, wie die Rotationen oder die Beugung wurden kaum bis gar nicht benutzt.

5.1.1.4 Erkenntnisse & Änderungen

Um das Vorwärtsstoßen des Agents zu reduzieren, wurden die Belohnungen, die bei der Berührung der Zangenzähne mit dem Fremdkörper verteilt werden, reduziert. Diese wurden immer dann vergeben, wenn bei der Vorwärtsbewegung die Zangenzähne den Fremdkörper berühren, egal in welcher Lage dieser sich befindet. Da diese wegen der zwei Zangenzähne doppelt verteilt werden konnten, wurden sie auf weniger als die Hälfte ihres ursprünglichen Wertes reduziert. Somit entsprechen sie derselben Höhe wie die Belohnungen der Annäherung zum Fremdkörper.

So fallen diese Belohnungen geringer aus, auch um anderen Belohnungen mehr relatives Gewicht zu geben.

Außerdem wurden den Observationen zwei boolesche Werte hinzugefügt, welche dem Agent den Zustand des Griffs vermitteln sollten.

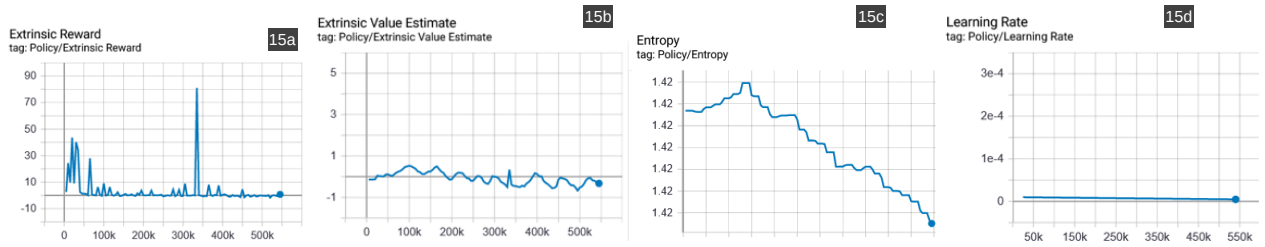
5.1.2 Training2_PPO_GrabCoin2

5.1.2.1 Ausgangslage

Tabelle 2 Ausgangslage des Training2_PPO_GrabCoin2, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: **batch_size: 512** **buffer_size: 20480** **learning_rate: 1e-5** **beta: 1e-2** epsilon: 0.2 lambd: 0.95 num_epoch: 3 learning_rate_schedule: linear network_settings: normalize: false **hidden_units: 256** num_layers: 2 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 **checkpoint_interval: 50000** **max_steps: 1000000** **time_horizon: 2048** **summary_freq: 5000** threaded: true	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Kapsel-Spitze zu Münze **- Zange1 InnenBerührt als Boolean** **- Zange2 InnenBerührt als Boolean**	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	- Wenn der Abstand zur Münze verringert wird (+0.1) - Wenn der Abstand zur Münze größer wird (-0.1) **- Wenn Zahn1 jeder Art berührt wird (+0.1)** **- Wenn Zahn2 jeder Art berührt wird (+0.1)** - Wenn beide Zähne die Münze innen berühren (+1 und Ende) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE)

5.1.2.2 Ergebnisdaten des Trainingsverlaufs



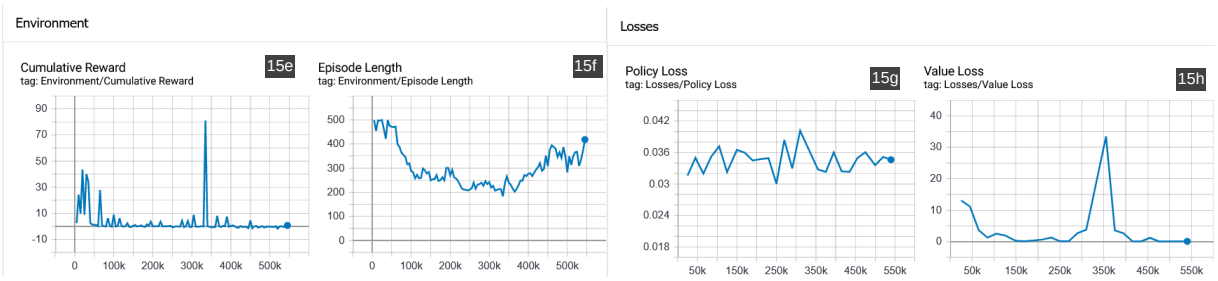


Abb. 15 a-h Trainingsdaten des Training2_PPO_GrabCoin2, Quelle: Tensorboard Bildschirmaufnahme

5.1.2.3 Ergebnisse der Tests

Während des Tests mit 25000 Aktionen, wurde der Fremdkörper während 68 Aktionen berührt, davon 37 mit der Innenseite eines Zangenzahns. Während 17 Aktionen befand sich der Fremdkörper zwischen den Zangenzähnen, dabei wurde der Fremdkörper kein einziges Mal gegriffen.

Innerhalb der beobachteten Testversuche bewegte sich der Agent nur langsam nach vorne. Andere Aktionen wie die Rotationen und Beugung werden genutzt, allerdings sehr ruckartig und schwach.

5.1.2.4 Erkenntnisse & Änderungen

Um dem Fokus auf das Greifen zu verlagern, wurden die Belohnungen, die der Agent bei Annäherungen oder Entfernung zum Fremdkörper erhält, stark heruntergesetzt. Auch die Belohnungen, die bei einer Berührung zwischen Zangenzähnen und dem Fremdkörper verteilt werden, wurden halbiert. Dies sollte verhindern, das alleiniges Berühren des Fremdkörpers zu viele Belohnungen verspricht.
So wurden die Aktionen, die zum Erreichen des Ziels notwendig sind, leicht belohnt um das Zugreifen, mit einer höheren Belohnung, hervorzuheben.

5.1.3 Training3_PPO_GrabCoin3

5.1.3.1 Ausgangslage

Tabelle 3 Ausgangslage des Training3_PPO_GrabCoin3

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 5120 buffer_size: 20480 learning_rate: 1e-5 **beta: 1e-4** epsilon: 0.2 **lambd: 0.90** num_epoch: 3 learning_rate_schedule: linear network_settings: normalize: false **hidden_units: 128** **num_layers: 3** vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 **checkpoint_interval: 50000** max_steps: 500000 time_horizon: 2048 summary_freq: 5000 threaded: true	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	**Wenn der Abstand zur Münze verringert wird (+0.0001)** **- Wenn der Abstand zur Münze größer wird (-0.0001)** **- Wenn Zahn1 jeder art berührt wird (+0.05)** **- Wenn Zahn2 jeder Art berührt wird (+0.05)** - Wenn beide Zähne Innen die Münze berühren (+1 & Ende) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE)

5.1.3.2 Ergebnisdaten des Trainingsverlaufs

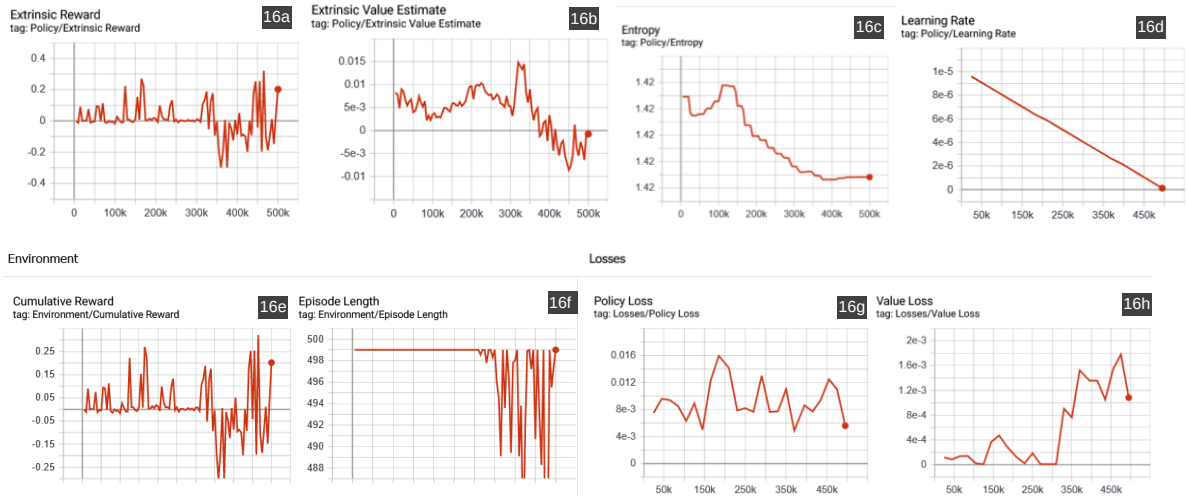


Abb. 16 a-h Trainingsdaten des Training3_PPO_GrabCoin3, Quelle: Tensorboard Bildschirmaufnahme

5.1.3.3 Ergebnisse der Tests

Innerhalb des Test mit 25000 Aktionen, wurde der Fremdkörper nur während 12 Aktionen berührt, davon sechs mit der Innenseite einer der beiden Zangenzähne. Während drei Aktionen befand sich der Fremdkörper in der Mitte, zwischen den beiden Zangenzähnen, hierbei wurde kein einiges mal zugegriffen.

Der Agent zeigte ein sehr passives Verhalten mit sehr wenigen ungezielten und ruckartigen Bewegungen und Rotationen. Das Zugreifen mit den Zangenzähnen oder die Rotation des Zangenträgers wird nicht genutzt.

5.1.3.4 Erkenntnisse und Änderungen

Da dieser Agent nur wenige und passive Bewegungen zeigte, wurden die Belohnungen für die Annäherung an den Fremdkörper, wieder etwas erhöht. Dies sollte den Kontakt mit dem Fremdkörper fördern, um in Zustände zu gelangen die weitere Belohnungen versprechen.

5.1.4 Training4_PPO_GrabCoin4

5.1.4.1 Ausgangslage

Tabelle 4 Ausgangslage des Training4_PPO_GrabCoin4, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: **batch_size: 2048** buffer_size: 20480 learning_rate: 1e-5 beta: 1e-2 #X(5e-3) epsilon: 0.1 **lambd: 0.925** num_epoch: 3 **learning_rate_schedule: constant** network_settings: normalize: false **hidden_units: 256** **num_layers: 3** vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 500000 time_horizon: 2048 summary_freq: 5000 threaded: true	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	**- Wenn der Abstand zur Münze verringert wird (+0.001)** **- Wenn der Abstand zur Münze größer wird (-0.001)** - Wenn Zahn1 berührt wird (+0.05) - Wenn Zahn2 berührt wird (+0.05) - Wenn Münze von beiden Zähnen Innen berührt wurde(+1 & EndE) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE)

5.1.4.2 Ergebnisdaten des Trainingsverlaufs

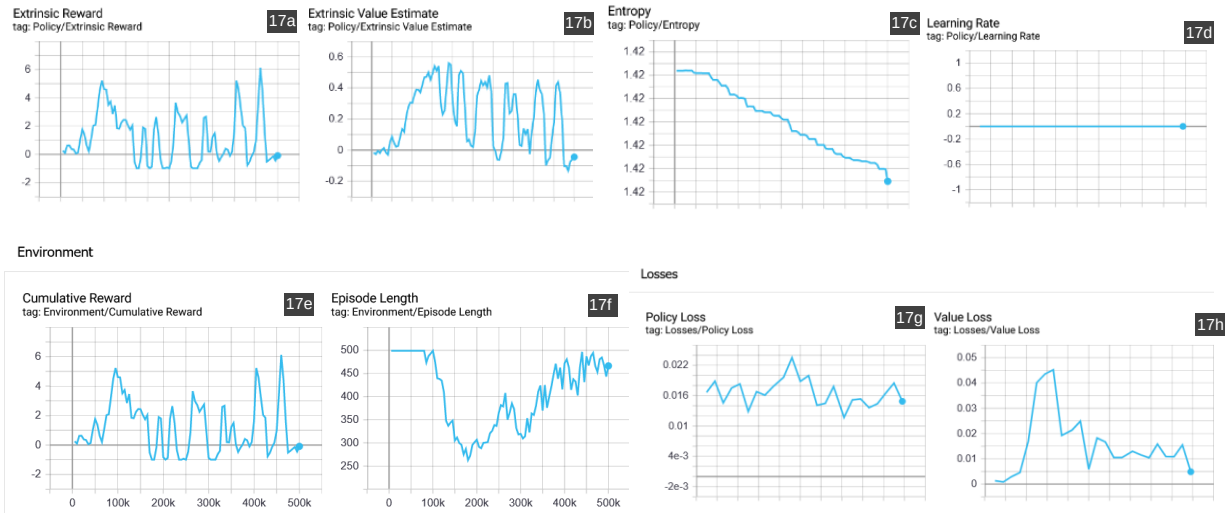


Abb. 17 a-h Trainingsdaten des Training4_PPO_GrabCoin4, Quelle: Tensorboard Bildschirmaufnahme

5.1.4.3 Ergebnisse der Tests

Während der 25000 Aktionen innerhalb des Tests, berührte der Agent den Fremdkörper in 31 Aktionen, davon elf mit der Innenseite eines Zangenzahns. Zwischen den Zangenzähnen befand sich der Fremdkörper während sieben Aktionen, allerdings ohne ein einziges Mal zuzugreifen.

Der Agent bewegt sich sehr langsam nach vorne, dabei wechselt er immer wieder kurz in die Bewegung nach hinten. Die anderen Aktionen werden nur sehr schwach benutzt bis gar nicht benutzt.

5.1.4.4 Erkenntnisse und Änderungen

Bei einem Rückblick fällt auf, dass die Belohnungen nur bei Annäherung oder beim Prozess des Greifens verteilt wurden. Um hier einen Zwischenschritt einzufügen, wurde eine neue Belohnung eingeführt. Diese erhielt der Agent, wenn sich die Münze in der Mitte zwischen den beide Zangenzähnen befand. Dies hat den Zweck die Aufgabe in Teilschritte zu unterteilen, die durch ihre Belohnungen gekennzeichnet sind. So werden die Belohnungen eingeteilt in die Annäherung, den Kontakt, die mittige Platzierung des Fremdkörpers und in das Zugreifen.

5.1.5 Erweiterung durch BC

Die Ergebnisse der ersten Trainingsversuche zeigten, dass das gewünschte Zielverhalten nur schwer allein mittels Training durch PPO erzeugt werden kann. Hierzu schien die Aufgabe zu komplex und der Erhalt wichtigen Belohnungen zu selten. Daher wurde das Trainings mit Unitys-BC ergänzt, wodurch das Zielverhalten und der Verlauf der Aufgabenlösung demonstrieren werden können. Diese Demonstrationen haben anschließend Einfluss auf den Trainingsprozess.

5.1.6 Training5_BC_GrabCoin1

5.1.6.1 Ausgangslage

Tabelle 5 Ausgangslage des Training5_BC_GrabCoin1, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480 learning_rate: 1e-5 **beta: 5e-3** epsilon: 0.1 lambd: 0.95 num_epoch: 3 **learning_rate_schedule: linear** network_settings: normalize: false **hidden_units: 256** **num_layers: 2** vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 500000 time_horizon: 2048 summary_freq: 5000 threaded: true **behavioral_cloning:** **demo_path: Project/Assets/ThesisProject/demos/EndoBC.demo** **steps: 0** **strength: 1** **samples_per_update: 0**	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	**-** Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001) - Wenn Zahn1 im Inneren berührt wird (+0.05) - Wenn Zahn2 im Inneren berührt wird (+0.05) - Wenn Münze von beiden Zähnen Innen berührt wurde(+1 & EndE) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE) **- Wenn Münze zwischen den beiden Zangen ist (+0.05)**

5.1.6.2 Ergebnisdaten des Trainingsverlaufs

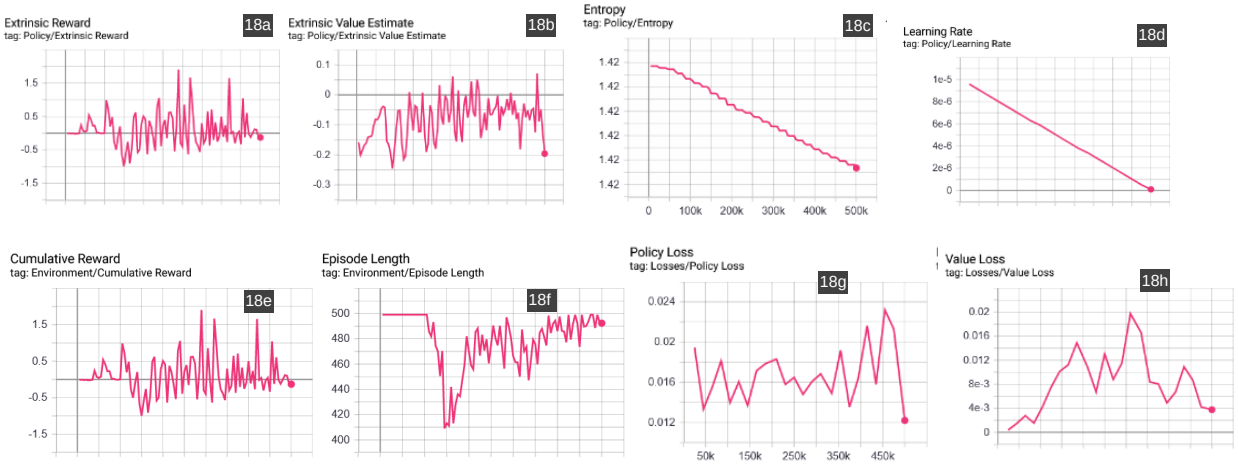


Abb. 18 a-i Trainingsdaten des Training5_BC_GrabCoin1, Quelle: Tensorboard Bildschirmaufnahme

Während des Tests, mit 25000 Versuchen, gelang es dem Agent während 49 Aktionen den Fremdkörper zu berühren, davon waren 15 mit der Innenseite eines Zangenzahns.
Der Fremdkörper befand sich in 13 Aktionen in der Mitte, zwischen den beiden Zangenzähen, doch wurde nicht einmal von diesen gegriffen.

Der Agent bewegte sich in Richtung Fremdkörper, bis dieser, falls die Lage des Fremdkörpers es erlaubte, zwischen den beiden Zangenzähnen liegt. Allerdings ohne anschließendes Greifen. Rotationen und die Biegungen des Gelenks wurden nur minimal eingesetzt.

5.1.6.3 Erkenntnisse und Änderungen

Dass der Agent sich nur wenig weiterbewegte, falls sich der Fremdkörper zwischen den Zangenzähnen befand und dabei auch andere Aktionen nur wenig einsetzte, ließ vermuten, dass die Belohnungen, welche dabei verteilt werden, zu hoch waren. So gab es keinen Grund weitere Belohnungen, wie die des Greifens zu erreichen.
Aus diesem Grund wurde diese Belohnung reduziert.
Des Weiteren wurde den Observationen ein Vektor hinzugefügt. Dieser entsprecht dem Vektor zwischen dem Fremdkörper und der Mitte zwischen den beiden Zangenzähnen. Dieser Vektor gibt Aufschluss über die relative Position des Fremdkörpers zum Agent.

5.1.7 Training6_BC_GrabCoin2

5.1.7.1 Ausgangslage

Tabelle 6 Ausganglage des Training6_BC_GrabCoin2, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float	- vor und zurück bewegen - rotieren - biegen - Zange rotieren	- Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001)

learning_rate: 1e-5 beta: 5e-3 # epsilon: 0.1 lambd: 0.95 num_epoch: 3 **learning_rate_schedule: linear** network_settings: normalize: false hidden_units: 256 num_layers: 2 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 1000000 time_horizon: 2048 summary_freq: 5000 threaded: true behavioral_cloning: demo_path: Project/Assets/ThesisProject/demos/EndoBC.demo **steps: 150000** **strength: 0.5** samples_per_update: 0 num_epoch: 3	- lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean - **Vektor zur Münze eingefügt**	- Zange öffnen und schließen	- Wenn Zahn1 im Inneren berührt wird (+0.05) - Wenn Zahn2 im Inneren berührt wird (+0.05) - Wenn Münze von beiden Zähnen Innen berührt wurde(+1 & EndE) - Bei Kontakt der Zange mit der Magenwand (-1 & EndE) **- Wenn Münze zwischen den beiden Zangen ist (+0.001) reduziert worden**

5.1.7.2 Ergebnisdaten des Trainingsverlaufs

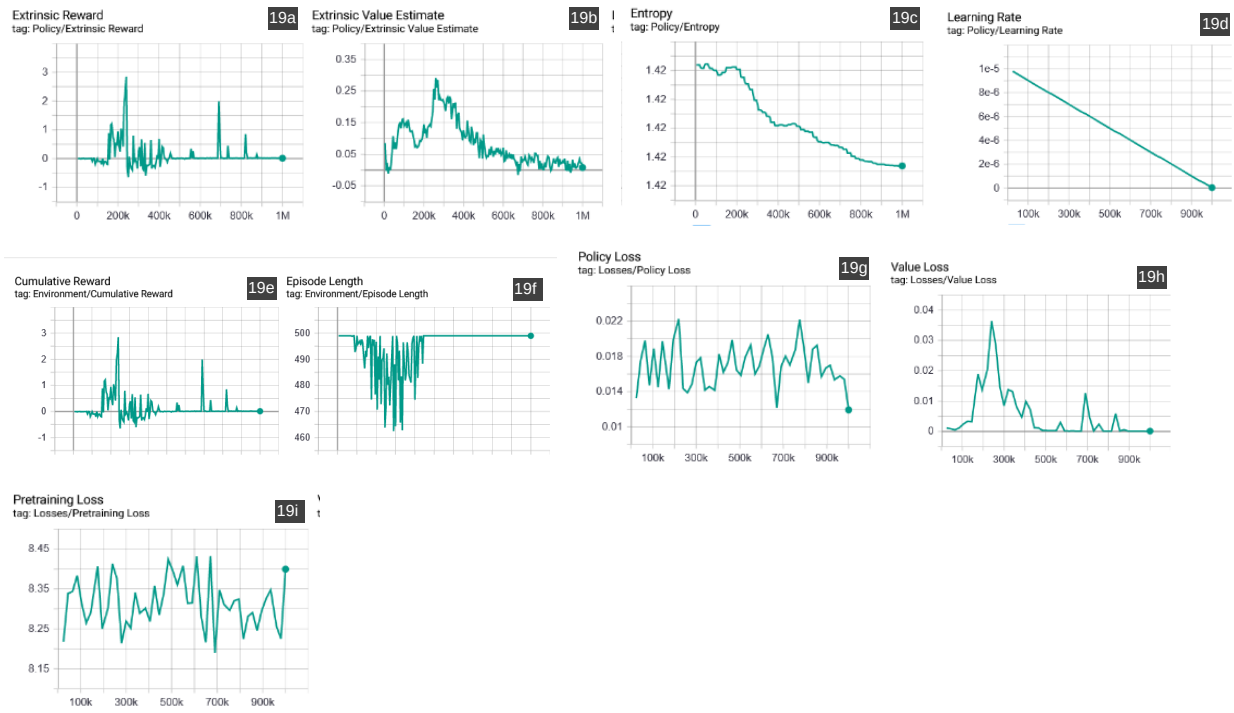


Abb. 19 a-i Trainingsdaten des Training6_BC_GrabCoin2, Quelle: Tensorboard Bildschirmaufnahme

5.1.7.3 Ergebnisse der Tests

Während der 25000 Aktionen innerhalb des Tests, wurde der Fremdkörper in 92 Aktionen berührt, davon 52 mit der Innenseite eines Zangenzahns. In der Mitte, zwischen den beiden Zangenzähnen befand sich der Fremdkörper während 27 Aktionen, allerdings ohne gegriffen zu werden.

Der Agent bewegt sich langsam nach vorne und verwendete auch andere Aktionen wie die Rotationen, das Biegen und das Zugreifen. Die Verwendung dieser war unkoordiniert, aber stärker als in vorherigen Versuchen.

5.1.7.4 Erkenntnisse

Das Hinzufügen der Richtung des Fremdkörpers zu dem Observationen hat dazu beigetragen, dass mehr Berührungen stattgefunden haben. Dieser Vektor bat Informationen für die Annäherung zum Fremdkörper und auch Informationen darüber ob sich der Fremdkörper zwischen den beiden Zangenzähen befand (bei Vektor = 0,0,0). Beide Fälle führten bereits zu Belohnungen, so wurden keine weiteren Aktionen benötigt.

5.1.8 Änderung am Versuchsaufbau

Während der folgenden zwei Trainingsversuchen, wurde das Ziel der Aufgabe dahingehen verändert, dass der Fremdkörper nun nicht mehr gegriffen werden muss, sondern lediglich mittig zwischen den beiden Zangen platziert werden soll, um den Versuch erfolgreich abzuschließen.
Weiter wurden Veränderungen an den Belohnungen und den Observationen vorgenommen. Zu den Änderung der Belohnungen gehört das Beenden des erfolgreichen Versuchs, bei mittiger Platzierung des Fremdkörpers und zu den Änderungen der Observationen gehören das Entfernen des Richtungsvektors zum Fremdkörper und das Hinzufügen eines booleschen Wertes, der auf *true* gesetzt wird, sobald sich der Fremdkörper in der Mitte befindet.

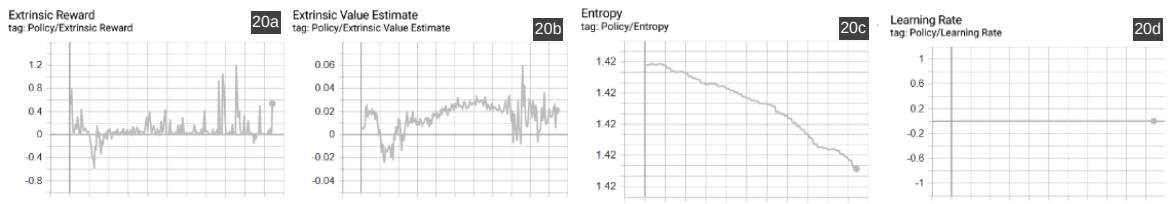
5.1.9 Training7_BC_NoGrabCoin1

5.1.9.1 Ausgangslage

Tabelle 7 Ausganglage des Training7_BC_NoGrabCoin1, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480 learning_rate: 1e-5 beta: 5e-3 epsilon: 0.1 lambd: 0.925 num_epoch: 3 **learning_rate_schedule: constant** network_settings: normalize: false hidden_units: 256 **num_layers: 3** vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 1000000 time_horizon: 2048 summary_freq: 5000 threaded: true behavioral_cloning: demo_path: Project/Assets/ThesisProject/demos/EndoBC.demo **steps: 500000** **strength: 0.5** samples_per_update: 0 num_epoch: 3	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean - MitteTouched als Boolean - Vektor der Richtung gestrichen	- vor und zurück bewegen - rotieren - biegen - Zange rotieren - Zange öffnen und schließen	- Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001) - Wenn Zahn1 im Inneren berührt wird (+0.05) - Wenn Zahn2 im Inneren berührt wird (+0.05) **- Wenn Münze sich in der Mitte zwischen den Zähnen befindet(+1 & EndEpisode())**

5.1.9.2 Ergebnisdaten des Trainingsverlaufs



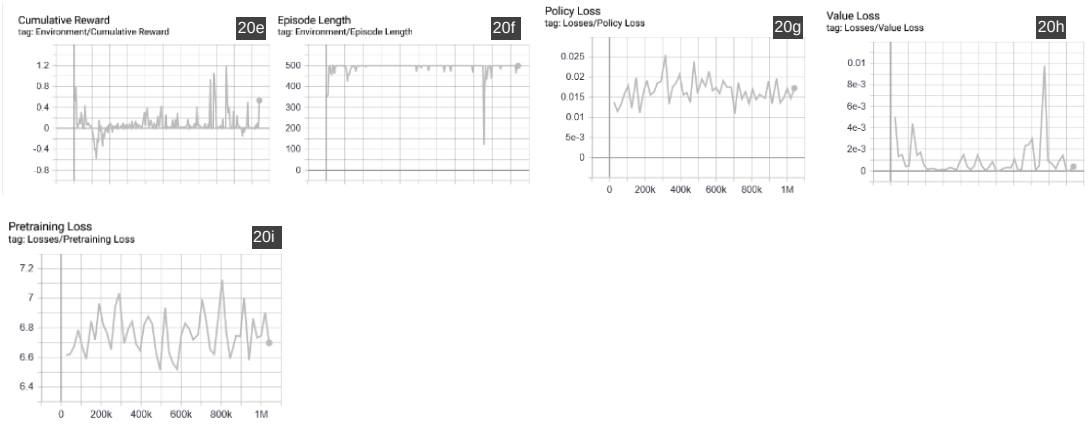


Abb. 20 a-i Trainingsdaten des Training7_BC_NoGrabCoin1, Quelle: Tensorboard Bildschirmaufnahme

In den 25000 Aktionen während des Trainings, wurde während 117 Aktionen der Fremdkörper berührt, davon 72 mit der Innenseite eines Zangenzahns. Das Ziel wurde durch mittiges Platzieren des Fremdkörpers, drei Mal erreicht.

Der Agent bewegt sich nach vorne und nutzt dabei wenige Rotationen und Biegungen, die mit denen er sich dem Fremdkörper ruckartig nähert.

5.1.9.3 Erkenntnisse und Änderungen

Die Änderung der Aufgabenstellung und das neue Ziel, machten es einfacher die Aufgabe zu lösen. Gleichzeitig wurde bei jedem Erfolg, die Aufgabe schwieriger gestaltet.
Die Anzahl an Aktionen in welcher der Fremdkörper mittig platziert war, war zwar niedriger als die der vorherigen Versuche, allerdings wurden die neuen Erfolge unter schwierigeren Bedingungen erzielt, da sich der Schwierigkeitsgrad bei erfolgreichem Lösen der Aufgabe steigt. Zur Erinnerung: Das erfolgreiche Lösen der Aufgabe und die Steigerung des Schwierigkeitsgrads, wären in vorherigen Versuchen beim Greifen des Fremdkörpers erreicht worden.
Die Änderungen waren zum einen die Reduzierung der Belohnungen bei einer Berührung des Fremdkörpers mit der Innenseite eines Zangenzähne. Dies hatte den Zweck die Berührungen der Innenseite in den Hintergrund zu rücken. Des Weiteren wurde dem Agent die Aktionen entfernt, mit welcher sich die Zangenzähne öffnen und schließen. Diese Aktion würde sonst erlernt werden, ohne maßgeblich zum Erfolg beizutragen.

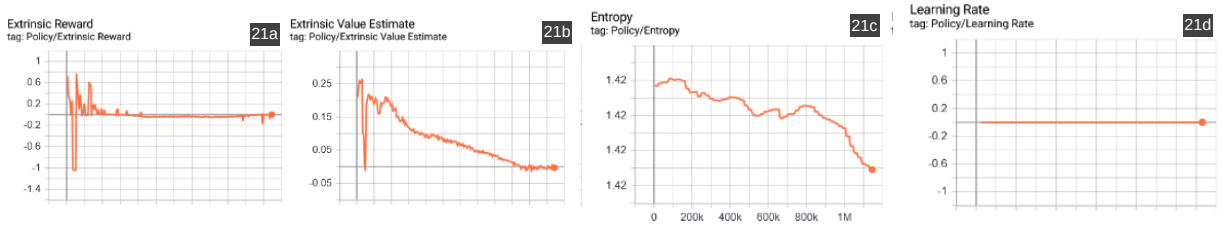
5.1.10 Training8_BC_NoGrabCoin2

5.1.10.1 Ausgangslage

Tabelle 8 Ausganglage des Training8_BC_NoGrabCoin2, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480 learning_rate: 1e-5 beta: 5e-3 epsilon: 0.1 lambd: 0.925 num_epoch: 3 learning_rate_schedule: constant network_settings: normalize: false hidden_units: 256 num_layers: 3 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 1000000 time_horizon: 2048 summary_freq: 5000 threaded: true behavioral_cloning: demo_path: Project/Assets/ThesisProject/demos/EndoBC.demo **steps: 1000000** strength: 0.5 samples_per_update: 0 num_epoch: 3	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean - MitteTouched als Boolean	- vor und zurück bewegen - rotieren - biegen - Zange rotieren **//- Öffnen/Schließen entfernt**	- Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001) **- Wenn Zahn1 im Inneren berührt wird (+0.005)** **- Wenn Zahn2 im Inneren berührt wird (+0.005)** **Um im eher im Bereich der Bewegung zu bleiben** - Wenn Münze sich in der Mitte zwischen den Zähnen befindet(+1 & EndEpisode())

5.1.10.2 Ergebnisdaten des Trainingsverlaufs



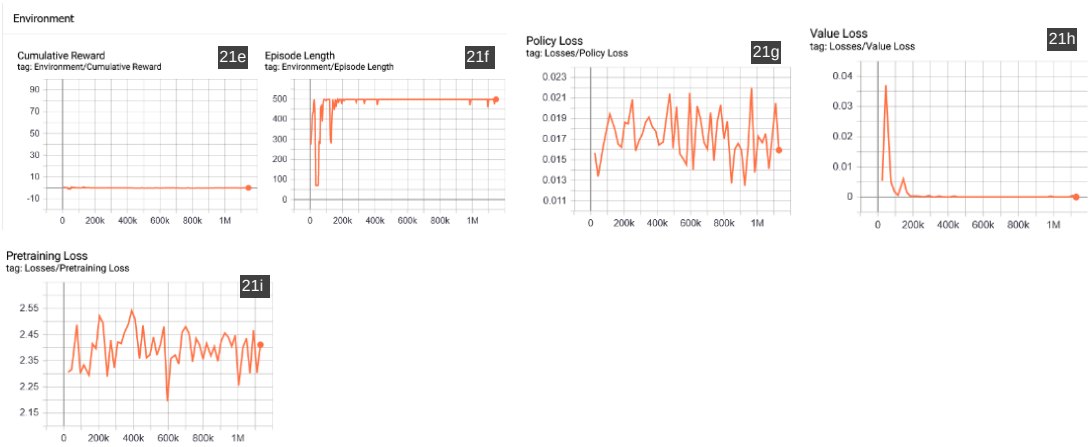


Abb. 21 a-i Trainingsdaten des Training8_BC_NoGrabCoin2, Quelle: Tensorboard Bildschirmaufnahme

5.1.10.3 Ergebnisse der Tests

In den 25000 Aktionen des Trainings, wurde während 131 Aktionen der Fremdkörper berührt, davon 76 mit der Innenseite eines Zangenzahns. Das Ziel wurde durch mittiges Platzieren des Fremdkörpers, vier Mal erreicht.

Der Agent bewegt sich ruckartig vorwärts in Richtung Fremdkörper. Rotationen und Biegungen werden leicht, aber mehr als in vorherigen Versuchen eingesetzt.

5.1.10.4 Erkenntnisse

Das heruntersetzten der Belohnung für die Berührungen der Innenseite und das Entfernen der Aktion des Greifens, haben ein dazu beigetragen die Testergebnisse ein wenig zu verbessern.

5.1.11 Änderung am Versuchsaufbau

Im folgenden Versuch war das Ziel der Aufgabe ebenfalls die mittige Positionierung des Fremdkörpers zwischen den beiden Zangenzähen, allerdings war nun der Fremdkörper keine Münze, sondern eine Kugel.
Eine weitere Änderung ist die Entfernung der Aktion, welche die Rotation des Zangenträgers durchführt. Diese Aktion wird im Falle einer Kugel nicht benötigt. Die letzte Änderung bestand darin, den Richtungsvektor des Fremdkörpers den Observationen hinzuzufügen, um Aufschluss über die Position des Fremdkörpers zu liefern. Auch um nochmal zu überprüfen wie entscheidend diese Informationen sind.

5.1.12 Training9_BC_NoGrabSphere1

5.1.12.1 Ausgangslage

Tabelle 9 Ausganglage des Training9_BC_NoGrabSphere1, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480 learning_rate: 1e-5 beta: 5e-3 epsilon: 0.1 lambd: 0.925 num_epoch: 3 **learning_rate_schedule: constant** network_settings: normalize: false hidden_units: 256 num_layers: 3 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 5000000 time_horizon: 2048 summary_freq: 5000 threaded: true behavioral_cloning: demo_path: Project/Assets/ThesisProject/demos/EndoBC.demo **steps: 0** **strength: 1** samples_per_update: 0 num_epoch: 3	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean - MitteTouched als Boolean - **Vektor der Richtung eingefügt**	- vor und zurück bewegen - rotieren - biegen **- Zange rotieren** **Wurde entfernt**	- Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001) - Wenn Zahn1 im Inneren berührt wird (+0.005) - Wenn Zahn2 im Inneren berührt wird (+0.005) Um im eher im Bereich der Bewegung zu bleiben - Wenn Münze sich in der Mitte zwischen den Zähnen befindet(+1 & EndEpisode())

5.1.12.2 Ergebnisdaten des Trainingsverlaufs

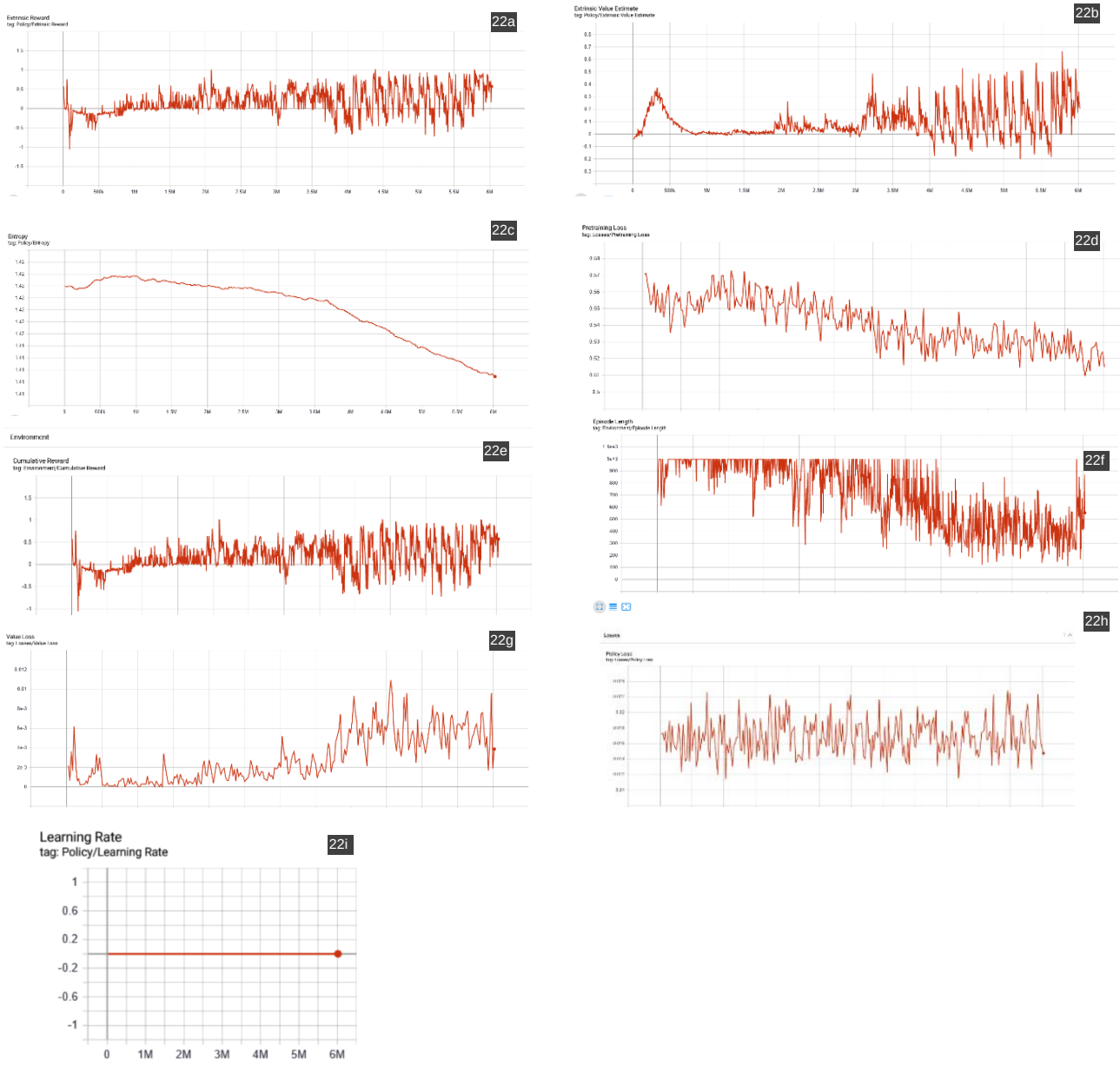


Abb. 22 a-i Trainingsdaten des Training9_BC_NoGrabSphere1, Quelle: Tensorboard Bildschirmaufnahme

5.1.12.3 Ergebnisse der Tests

In den 25000 Aktionen des Trainings, wurde während 143 Aktionen der Fremdkörper berührt, davon 102 mit der Innenseite eines Zangenzahns. Der Fremdkörper befand wurde sieben mal erfolgreich zwischen den Zangenzähnen platziert.

Der Agent bewegt sich nach vorne und bremst ab. Daneben werden die anderen Aktionen mal schwächer und mal stärker genutzt. Auch eine leichte Rotation und Biegung in Richtung des Fremdkörpers ist erkennbar.

5.1.12.4 Erkenntnisse

Das Ersetzen des Fremdkörpers durch eine Kugel vereinfacht die Aufgabe in der Art, dass es nicht mehr notwendig ist, sich der Lage des Objektes anzupassen, um ihn mittig platzieren zu können, da die Kugel von allen Seiten erreichbar ist.

5.1.13 Änderung am Versuchsaufbau

Im folgenden Versuch war das Ziel der Aufgabe erneut das Greifen des Fremdkörpers, jedoch war der Fremdkörper in diesem Fall eine Kugel anstatt einer Münze. Weitere Änderungen sind die Anpassungen der Belohnungen für den veränderten Versuchsaufbau und das hinzufügen der Belohnung, die vergeben wird, sobald der Fremdkörper mittig platziert wurde.

5.1.14 Training10_BC_GrabSphere1

5.1.14.1 Ausgangslage

Tabelle 10 Ausganglage des Training10_BC_GrabSphere1, Quelle: Eigene Darstellung

Hyperparameter	Observationen	Aktionen	Extrinsische Belohnungen
behaviors: ThesisHyper: trainer_type: ppo hyperparameters: batch_size: 2048 buffer_size: 20480 learning_rate: 1e-5 beta: 5e-3 epsilon: 0.1 lambd: 0.925 num_epoch: 3 learning_rate_schedule: constant network_settings: normalize: false hidden_units: 256 num_layers: 3 vis_encode_type: simple reward_signals: extrinsic: gamma: 0.99 strength: 1.0 keep_checkpoints: 5 checkpoint_interval: 50000 max_steps: 500000 time_horizon: 2048 summary_freq: 5000 threaded: true behavioral_cloning: Project/Assets/ThesisProject/demos/EndoBC.demo	- Kamera (100x80, graustufen) - Vector zwischen Startposition und Position als vec3 - eigene Rotation als float des y-wertes - Biegung als Winkel float - lokale Rotation des Zangenträgers als float des y-wertes - Lokale x-Position des ersten Zangenzahns - Lokale x-Position des zweiten Zangenzahns - Abstand zur Münze gemessen von Mitte zu Münze - Zange1 InnenBerührt als Boolean - Zange2 InnenBerührt als Boolean - MitteTouched als Boolean - Vektor der Richtung eingefügt	- vor und zurück bewegen - rotieren - biegen **- Greifen wurde wieder hinzugefügt**	- Wenn der Abstand zur Münze verringert wird (+0.001) - Wenn der Abstand zur Münze größer wird (-0.001) - Wenn Zahn1 im Inneren berührt wird (+0.005) - Wenn Zahn2 im Inneren berührt wird (+0.005) **- Wenn Münze sich in der Mitte zwischen den Zähnen befindet(+0.001)** **- Wenn die Münze gegriffen wurde (+1 & EndEpisode())**

steps: 0 strength: 1 samples_per_update: 0 num_epoch: 3			

5.1.14.2 Ergebnisdaten des Trainingsverlaufs

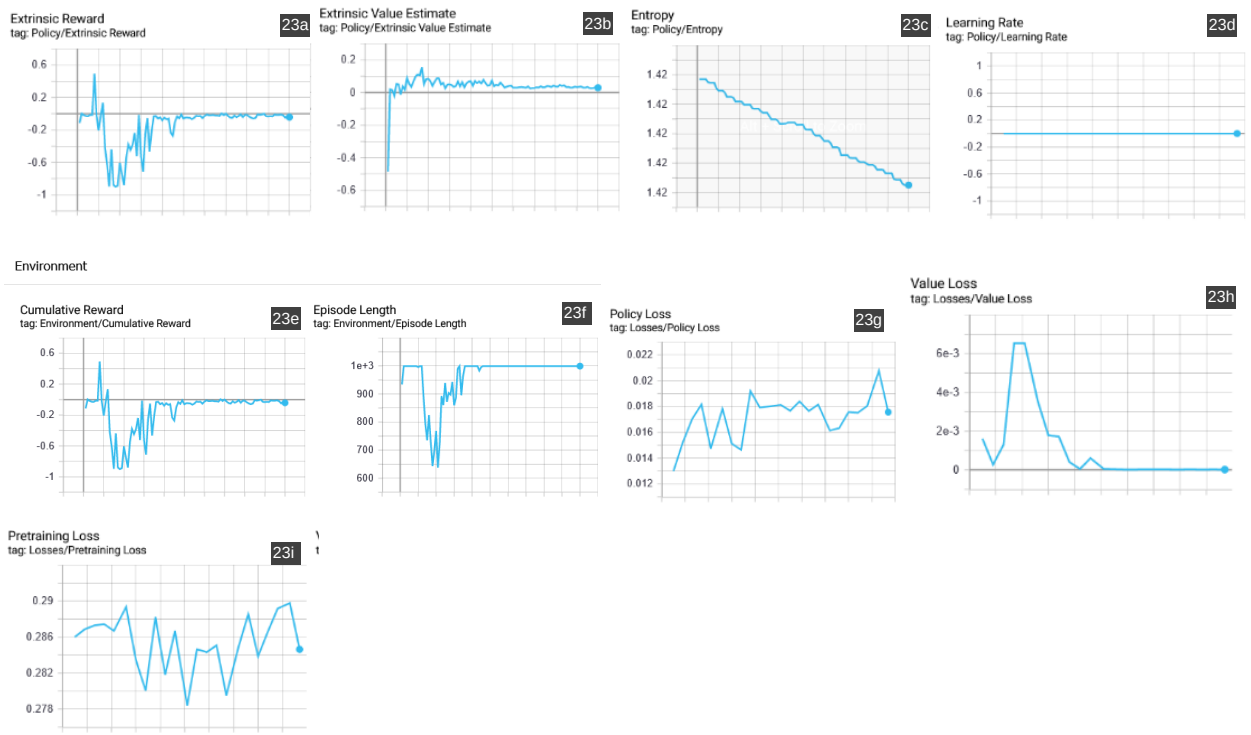


Abb. 23 a-i Trainingsdaten des Training10_BC_GrabSphere1, Quelle: Tensorboard Bildschirmaufnahme

5.1.14.3 Ergebnisse der Tests

In 25000 Aktionen während des Trainings, wurde in 126 Aktionen der Fremdkörper berührt, davon 91 mit der inneren Seite einer der beiden Zangenzähne. Der Fremdkörper war während 56 Aktionen mittig platziert. Es wurde zwei mal zugegriffen.

Der Agent bewegt sich in Richtung Fremdkörper und bremst passend ab während die Stärke der anderen Aktionen schwankt, so wie die Rotation in Richtung des Fremdkörpers.

5.1.14.4 Erkenntnisse

Auch bei einem kugelförmigen Fremdkörper, wurden die Belohnungen des Greifens zu selten verteilt und hatten deshalb keinen signifikanten Einfluss auf den Trainingsprozess.

5.1.15 Überlagerte Graphen

Im Folgenden wurden die Graphen der Trainingsdaten überlagert, um diese im späteren miteinander vergleichen zu können und daraus weitere Erkenntnisse zu gewinnen. Zuerst werden die Überlagerung der Graphen der ersten vier PPO-Trainingsversuche dargestellt. Darauf folgt die Überlagerung der Graphen der BC-Trainingsversuche. Zu Anfang findet sich die Farblegende, welche die Graphen farblich zuordnet

5.1.15.1 PPO-Überlagerung der Graphen

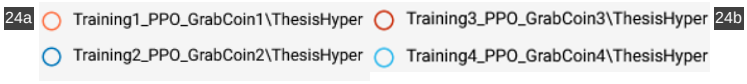


Abb. 24 a-b Farblegende der PPO-Tensorboard-Graphen, Quelle: Tensorboard Bildschirmaufnahme

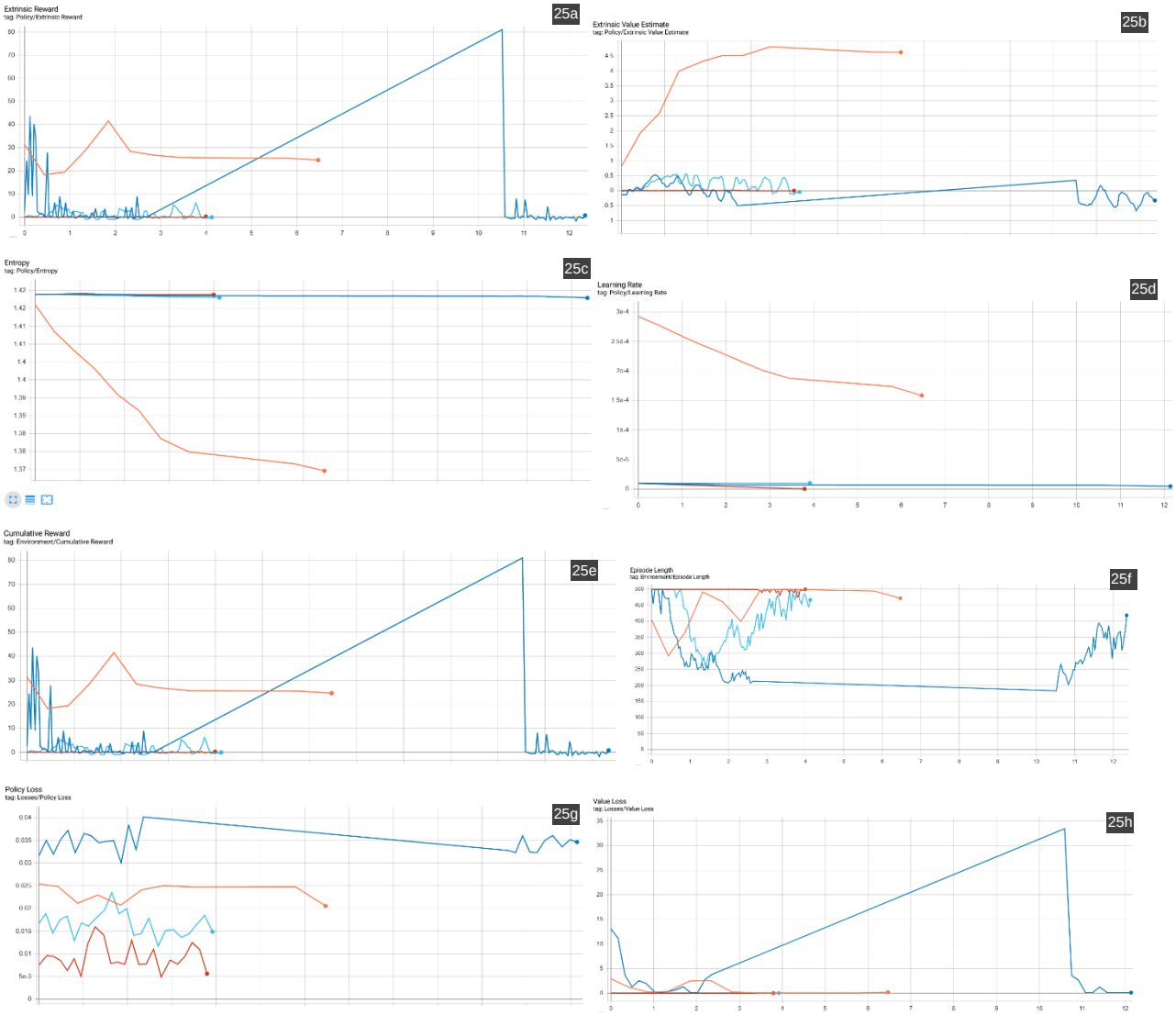


Abb. 25 a-h Überlagerung der PPO-Tensorboard-Graphen, Quelle: Tensorboard Bildschirmaufnahme

5.1.15.2 BC-Überlagerung der Graphen

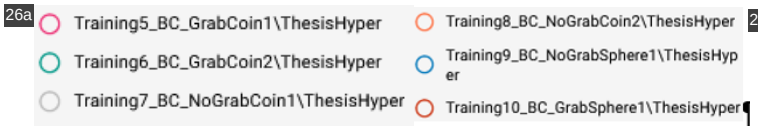


Abb. 26 a-b Farblegende der BC-Tensorboard-Graphen, Quelle: Tensorboard Bildschirmaufnahme

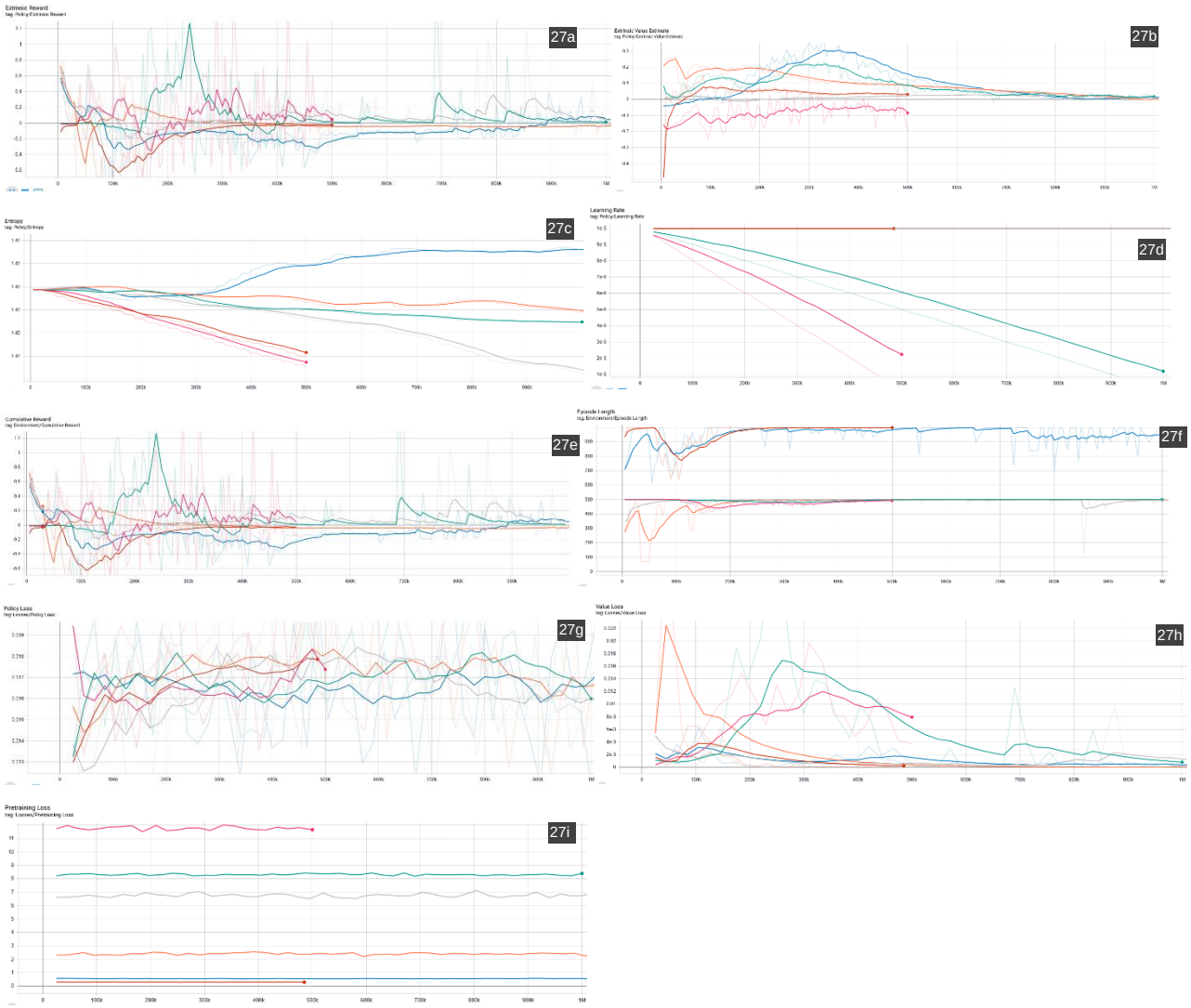


Abb. 27 a-h Überlagerung der BC-Tensorboard Graphen, Quelle: Tensorboard Bildschirmaufnahme

5.1.16 Verlaufstabelle

Tabelle 11 Verlaufstabelle

	Berührungen jeder Art	Innenseite berührt	In der Mitte platziert	Zugegriffen	Verhalten des Agents	Erkenntnisse	Änderungen
Training1_PPO_GrabCoin1	43	22	11	0	-Ruckartig langsam nach vorne -Andere Aktionen kaum benutzt	- Reine Vorwärtsbewegung, mit und ohne Kontakt muss beschränkt werden	Belohnung für Berührung jeder Art runtergesetzt damit stoßen sich nicht lohnt, im Vergleich zu

							Belohnung bei Innenseite
Training2_PPO_GrabCoin2	68	37	17	0	- Langsame Bewegung nach vorne - Andere Aktionen werden auch genutzt, allerdings ruckartig und schwach	- Belohnung der Annäherung und Berührung sind zu präsent	abstand Belohnung und Berührung Belohnung kleiner gemacht um greifen hervorzuheben
Training3_PPO_GrabCoin3	12	6	3	0	-Passive und unkoordinierte Aktionen -Wenige und schwache Bewegungen -Rotation des Zangenträgers und Greifen wird nicht benutzt	Bewegung zu passiv	Belohnung für Bewegung Richtung Münze wieder bisschen höher
Training4_PPO_GrabCoin4	31	11	7	0	-Langsame Bewegung nach vorne -Zwischendurch Ruckler nach hinten -Andere Aktionen schwach bis gar nicht genutzt	- Es braucht einen Zwischenschritt zwischen Bewegung und Greifen	Belohnung wenn Münze in Mitte eingefügt
Training5_BC_GrabCoin1	49	15	13	0	- Bewegung nach vorne - Wenige Rotationen und Biegungen des Gelenks - Ruhend falls Fremdkörper mittig	- Agent ruht falls Fremdkörper mittig, da die Belohnungen für ein mittiges Platzieren zu hoch sind	Belohnung durch Mitte wurde runtergesetzt Vektor zur Münze hinzugefügt
Training6_BC_GrabCoin2	92	52	27	0	- langsame Bewegung nach vorne - Aktionen wie Rotationen,	- Richtungsvektor als Observation hat zu mehr Berührungen beigetragen	Neuer Versuchsaufbau: Fremdkörper nicht greifen, sondern mittig platzieren

					Biegung und Zugreifen genutzt, aber unkoordiniert - Aktionen stärker genutzt als in vorherigen Versuchen		- Richtungsvektor wieder entfernt - Booleschen Wert eingeführt: Fremdkörper mittig platziert?
Training7_BC_NoGrabCoin1	117	72	3	x	- Ruckartige Näherung zum Fremdkörper - Wenige Biegungen und Rotationen in Richtung Fremdkörper	Zwar weniger in der Mitte platziert, dafür aber bei höherem Schwierigkeitsgrad	- Belohnungen für Berührungen runtergesetzt - Die Aktion des Greifens wurde entfernt
Training8_BC_NoGrabCoin2	131	76	4	x	- Ruckartig nach vorne - leichte, Biegungen und Rotationen in Richtung Fremdkörper, aber mehr als in zu vorigen Versuchen	Anpassung der Belohnungen und das Entfernen der Aktionen hab die Ergebnisse ein wenig verbessert In der Mitte platziert, bei höherem Schwierigkeitsgrad	Neuer Versuchsaufbau: Ziel ist die mittige Positionierung eines kugelförmigen Fremdkörpers - Aktion Zangenträger-Rotation wurde entfernt, da nicht notwendig - Richtungsvektor zum Fremdkörper wieder in Observationen
Training9_BC_NoGrabSphere1	143	102	7	x	- Bewegung nach vorne mit passendem abbremsen - Stärke der anderen Aktionen schwankt	Aufgabe leichter, da keine Notwendigkeit die Ausrichtung des Fremdkörpers zu beachten, In der Mitte platziert, bei	Neuer Versuchsaufbau: Ziel ist das Greifen des Fremdkörpers, eine Kugel - Belohnung bei mittiger Platzierung

					- Leichte Rotation und Biegung in Richtung Fremdkörper	höherem Schwierigkeitsgrad	des Fremdkörpers wurde hinzugefügt
Training10_BC_GrabSphere1	126	91	56	2	- Bewegung vorwärts bis Fremdkörper - Leichte Rotationen in Richtung Fremdkörper	Trotz kugelförmigen Fremdkörpers werden die Belohnungen des Greifens zu selten verteilt In der Mitte platziert, bei höherem Schwierigkeitsgrad	

5.2 Ergebnisverhalten des Agents

Die unterschiedlichen Trainingsverläufen, die Zählerstände und das Verhalten der Agents während der Tests, lassen verschiedene Beobachtungen und Rückschlüsse zu, die zu einem Ergebnis interpretiert werden.
Die Ergebnisse beziehen sich auf die Verhaltensweisen des Agents und die festgestellten Hürden beim Lösen der Aufgabe. Sie gliedern sich, wie das gewünschte Zielverhalten in die Annäherung, die Manöver zur Ausrichtung des Fremdkörpers und das Zugreifen.

Die Aktionen Bewegen, Rotieren und Biegen wurden bei der Annäherung in unterschiedlichem Ausmaße genutzt. Die Bewegung des Endoskops nach vorne war in fast allen Versuchen vorhanden, in einigen Fällen, wie im Versuch Training9_BC_NoGrabSphere1, auch mit einen passendem Bremsverhalten.
Der wohl wichtigste Grund für das häufige Auftreten dieser Aktion, ist die meist frontale Positionierung des Fremdkörpers, während des Trainings. Eine einfache Bewegung nach vorne versprach in dem meisten Fällen schnelle Belohnungen durch die ersten Berührungen.
Andere Aktionen der Annäherung wie das Rotieren des Endoskops oder die Biegung der Gelenke, zeigen sich im Großteil der Versuchen, allerdings meist mit einer schwachen und langsamen Nutzung. Erst in Versuchen mit einem höher erreichtem Schwierigkeitsgrad, wie Training7_BC_NoGrabCoin1, Training8_BC_NoGrabCoin2 und

Training9_BC_NoGrabSphere1 zeigen sich stärke Biegungen und Rotationen. Der Grund hierfür ist die Erhöhung des Schwierigkeitsgrads nur bei erfolgreichem Lösen der Aufgabe, da erst hier die Position des Fremdkörpers so verändert wird, dass eine Biegung notwendig ist.

Eine Hürde bei der Annäherung zum Fremdkörper ist die Bestimmung seiner Position. Dies macht ein Vergleich der beiden Versuche Training5_BC_GrabCoin1 und Training5_BC_GrabCoin2 deutlich, wobei in letzterem der Richtungsvektor zum Fremdkörper in den Observationen enthalten ist.

Hier ist erkennbar, dass der Richtungsvektor zu insgesamt mehr Berührungen und Positionierungen in der Mitte beigetragen hat.

Nachdem sich das Endoskop dem Fremdkörper genähert hat, war es meist notwendig ein Manöver auszuführen, das den Fremdkörper so ausrichtet, dass er mittig platziert werden konnte um ihn anschließend zu greifen.

Die erforderlichen Manöver sind eine große Hürde des Trainingsprozesses, zumal sie so vielfältig sein müssen wie die vielen unterschiedlichen Lagen des Fremdkörpers es erfordern. In keinen Versuchen zeigt der Agent ein koordiniertes Verhalten bei der Ausrichtung des Fremdkörpers. In einigen Versuchen gelingt die Ausrichtung der Münze durch das unsystematische Abwechseln von Vorwärts- und Rückwärtsbewegung, wobei die Münze so gestoßen wird, dass sie erreichbar wurde.

Bestätigt wird die Vermutung einer Hürde durch den Vergleich der Versuche mit einer Münze als Fremdkörper (Trainings 1-8) und den Versuchen mit einer Kugel als Fremdkörper(9-10). Hier fällt auf, dass letztere bessere Ergebnisse erzielen und einen höheren Schwierigkeitsgrad erreichen. Der Grund hierfür ist, dass bei einem kugelförmigen Fremdkörper keine Manöver notwendig sind, um diesen passend auszurichten.

Nachdem das Endoskop, dem Fremdkörper genähert wurde und mögliche Manöver absolviert wurde, galt es den Fremdkörper zu greifen.

Hier gab es keinen Versuch, in dem mit den Endoskop-Zangen zugegriffen wurde, mit Ausnahme des letzten Versuchs Training10_BC_GrabSphere1.

Die wenigen male des Zugreifens sind allerdings kein eindeutiges Zeichen für einen Lernprozess. Da keiner der Versuche ein methodisches Zugreifen zeigt, kann auch hier davon ausgegangen werden, dass das Greifen, neben der Bestimmung der Position und den Manövern eine Hürde im Lernprozess darstellt.

Zusammenfassend lässt sich sagen, dass bei der Annäherung alle Aktionen in verschiedenen Ausmaßen genutzt wurden. Besonders die Bewegung nach vorne wurde genutzt während die Rotation des Endoskops erst in späteren Versuchen zu Einsatz kam.
Hier wurde ebenfalls festgestellt, dass die Positionsbestimmung des Fremdkörpers eine Hürde darstellt. Für die vielfältigen Lagen des Fremdkörpers wurden keine koordinierten Manöver erlernt. Ein Vergleich mit den Versuchen mit kugelförmigen Fremdkörpern, hat gezeigt, dass diese Manöver eine große Hürde bei der Lösung der Aufgabe darstellen.
Ein systematisches Greifen zeigte sich in keinem Versuch, unabhängig davon ob der Fremdkörper eine Kugel oder eine Münze war.

6 Evaluierung der Realisierung

6.1 Evaluierung der Simulation

Zur Umsetzung der Simulation wurden die Bestandteile der 3D-Szene in einer abstrakteren und einfacheren Form dargestellt.
Die Nachbildung des menschlichen Körpers, besitzt alle nötigen Organe, allerdings mit niedrigerem Detailgrad, ohne Mageninhalt und ohne Animationen, welche z.B. die Bewegung der Magenwand hätten darstellen können.
Das Endoskop konnte mit einem 3D-Modell und der Erweiterung der Funktionen gut simuliert werden. So konnten hier die grundlegenden Funktionen einen Endoskops übertragen werden. Die Simulation des Fremdkörpers gelang in diesem Fall. Diese konnten dabei auch passend in ihrem Gewicht oder ihrer Reibung beeinflusst werden.
Eine Schwierigkeit bei der Simulierung solcher 3D-Szenen ist die Berechnung der physikalischen Eigenschaften der 3D-Objekte und der Szene. So sind z.B. die Berechnungen der Gravitation oder des Lichts nur so genau wie es die Rechenleistung zulässt. Die Berechnungen beinhalten auch die Kollisionserkennung Unitys, welche auch die Aktionen des Agent beeinflussen Diese Aktionen dürfen nicht schneller sein als Unitys Kollisionserkennung es erkennen könnte.

6.2 Evaluierung der Versuche

Nach den ersten vier Versuchen hatte sich bereits gezeigt, dass das alleinige Nutzen des PPO Algorithmus‘ nicht ausreicht, um das gesamte Zielverhalten zu lernen. Eine Schwierigkeit hierbei ist die komplexe Aufgabe und die Zwischenschritte, die es zu erreichen

gilt. Dabei ist es sehr unwahrscheinlich, dass durch alleiniges Versuchen und Irren des Agents, die gesamte Aufgabe gelöst wird.
Die Erweiterung durch das BC hat zu einer Näherung an das Zielverhalten geführt. Allerdings übertraf die Vielfalt der Lagen des Fremdkörpers bei weiten, diejenige die Teil der Demonstrationen waren.
Während der Anpassungen der Belohnungen entstand ein Konflikt zwischen den Belohnungen der Zwischenschritte und der Belohnung des erfolgreichen Abschließens der Aufgabe. Erstere konnten hierbei schon ausreichen, um letztere nicht mehr lohnenswert zu machen. So reichten z.B. die Belohnungen der Annäherung oder Berührung schon aus, sodass weitere Lernprozesse nicht notwendig waren.
Ein weiterer Konflikt zeigt sich bei der Platzierung des Fremdkörpers und der Erhöhung des Schwierigkeitsgrads, nur bei erfolgreichem Abschließen der Aufgabe. Diese Bedingung sorgt dafür, dass Situationen mit einfachen Zwischenschritte viel öfter vorkommen, als die Situationen die ausgiebigere Aktionen verlangen. So entsteht eine Überanpassung(englisch: Over Fitting) für ein Verhalten, das nur auf einfache Situationen reagieren kann und wenig ausgeprägte Aktionen zeigt.

7 Schluss

7.1 Fazit

Das Ziel ist es eine 3D-Simulation zu erstellen und darin die Unitys-ML-Agents zu nutzen, um einen Agent zu erstellen, der ein funktionales 3D-Endoskop steuert und bei Anwendung dessen ein gewünschtes Zielverhalten lernt, das zur Lösung der gestellten Aufgabe führt. Die Aufgabe des Agents bestand darin, mit der richtigen Kombination an Aktionen, da Endoskop so zu steuern, dass es sich einem simulierten Fremdkörper nähert. Dieser sollte anschließend, nach der Ausrichtung des Fremdkörpers, gegriffen werden.
Diese Aufteilung in Zwischenschritte in Annäherung, Manöver zur Ausrichtung und Zugreifen, erlaubt es auch die Ergebnisse in dieser Gliederung zusammenzufassen.

Die Annäherung des Endoskops an den Fremdkörper geschieht hauptsächlich durch eine Vorwärtsbewegung des Endoskops, in wenigen Fällen auch mit einem vernünftigen Bremsverhalten. In späteren Versuchen zeigt sich auch die Nutzung der Rotation des Endoskops und die Biegung der Gelenke. Die Ausrichtung des Fremdkörpers und das Zugreifen wurden in keinem der Versuche koordiniert und systematisch durchgeführt.

Der Auslöser für die starke Nutzung der Vorwärtsbewegung ist die häufige Platzierung des Fremdkörpers vor dem Endoskop, während des Trainings. Eine Bewegung nach vorne ist hier der schnellste Weg durch die Berührung des Fremdkörpers Belohnungen zu erhalten.
Aus einem Vergleich verschiedener Versuche mit und ohne einen Richtungsvektor zum Fremdkörper, kann ebenfalls geschlossen werden, dass die Bestimmung der Position des Fremdkörpers während der Annäherung, eine Hürde im Lernprozess darstellt. Dies lässt sich durch steigende Zahlen der Berührungen, nach Hinzufügen des Richtungsvektors, bestätigen.
Ein Grund für das schwere Erlernen der Manöver ist die große Anzahl an Positionen und Rotationen in welcher der Fremdkörper sich befinden kann. Das alleinige Nutzen des PPO Algorithmus‘ verspricht nur wenige Erfolge, da es sehr unwahrscheinlich ist, dass der Agent durch lediglich Versuchen und Irren die gesamte Aufgabe löst. Durch die Erweiterung durch BC gelingt es dem Agent sich dem Zielverhalten zu nähern, allerdings können nur begrenzt Demonstrationen eingesetzt werden, die von der Varianz des Trainings übertroffen werden.
Ein Beleg für die festgestellte Hürde ist der Vergleich zwischen Versuchen mit einer Münze oder einer Kugel als Fremdkörper. In letzteren zeigt sich eine Steigerung aller Zähler und auch eine Steigerung des Schwierigkeitsgrads, da die Hürde der Manöver entfernt wurde.
Das schlechte Erlernen des Zugreifens steht mit einem Konflikt der Belohnungen in Zusammenhang. Der Konflikt herrscht hier zwischen den Belohnungen des Zugreifens und den Belohnungen der vorherigen Zwischenschritte. Letztere Belohnungen können bereits ausreichen, um weitere Aktionen weniger lohnenswert zu machen.
Bei der übergeordneten Simulation wurde erfolgreich eine abstraktere Form implementiert, die den Sachverhalt vereinfacht. Diese Abstraktion führen allerdings dazu, dass sich die Simulation, mit jeder Vereinfachung von der Realität entfernt. Daher herrscht hier nur ein eingeschränkter Geltungsbereich, der die Grenzen einer Simulation nicht übersteigt und nicht auf einen reales Endoskop anwendbar ist.
Eine Weiterentwicklung mit einer niedrigeren Abstraktionsebene, einem höheren Detailgrad und einer starken Näherung zur Realität, könnten es allerdings möglich machen, den Agent innerhalb der Simulation auf ein Zielverhalten zu trainieren, um das gelernte Verhalten anschließend in der Realität an einem echten Endoskop anzuwenden.
Bei einer möglichen Weiterentwicklung sollten die Abstraktionsebenen, die Erkenntnisse und die Hürden beachtet werden. So sollte zuerst eine möglichst detailreiche Simulation erstellt werden, welche so wenig Vereinfachungen nutzt wie möglich.

Zum erfolgreichen Erlernen von Manövern wird empfohlen, diese unabhängig von der Annäherung und dem Zugreifen zu trainieren. Hier sollte der Schwierigkeitsgrad langsam gesteigert werden, allerdings nicht so langsam, dass eine Überanpassung für leichtere Szenarien stattfindet.
Für ein erfolgreiches Zugreifen sollten die Belohnungen so angepasst werden, dass auch nach den ersten Belohnungen der Zwischenschritte, eine Belohnung des Zugreifens erstrebenswert bleibt, um weitere Aktionen zu fördern. Auch dieser letzte Schritt kann unabhängig von der Annäherung und den Manövern trainiert werden.

8 Literaturverzeichnis

Andreas Neff: Endoskopische Verfahren in der Mund-Kiefer-Gesichtschirurgie, Wiesbaden, Deutschland: Springer, 2015 S.9 f.

Georg Kähler / Martin Götz / Norbert Senninger (Hrsg.): Therapeutische Endoskopie im Gastrointestinaltrakt, Berlin, Deutschland: Springer, 2016, S.178 ff.

Gerd Lux: Gastrointerale Fremdkörper, in: Joachim F. Erckenbrecht / Sven Jonas (Hrsg.), Viszeralmedizin, Berlin, Deutschland: Springer, 2015, S.129 f.

Klaus-Martin Irion / Martin Leonhard: Endoskopie – Geräte, Systeme und Methoden, in: Rüdiger Kramme. (Hrsg.) Medizintechnik, 5.Aufl.,Berlin, Deutschland: Springer Reference Technik, Springer Verlag, 2015, S.2 ff.

Klaus-Martin Irion / Martin Leonhard: Endoskopie – Geräte, Systeme und Methoden, in: Rüdiger Kramme. (Hrsg.) Medizintechnik, 5.Aufl.,Berlin, Deutschland: Springer Reference Technik, Springer Verlag, 2015, S.9

Michael Schünke / Erik Schulte / Udo Schumacher: Magen: Lage, Form, Gliederung und Innenansicht, in: Thieme physioLink, [online], https://physiolink.thieme.de/prometheus/236910105/236910105_4_117 [03.08.2020]

Michael Schünke / Erik Schulte / Udo Schumacher: Magen: Wandaufbau und Histologie, in: Thieme physioLink, [online], https://physiolink.thieme.de/prometheus/236910105/236910105_4_118 [03.08.2020]

OpenAi: Grundlinien Proximal Policy Optimization, in OpenAi,, 20.07.17,[online] https://openai.com/blog/openai-baselines-ppo/ [18.08.2020]

Peter Collet: Endoskopische Fremdkörperextraktion, in: Georg Kähler / Martin Götz / Norbert Senninger (Hrsg.), Therapeutische Endoskopie im Gastrointestinaltrakt, Berlin, Deutschland: Springer, 2016, S.180 ff.

pypi: ML-Agents Python Modul, in: pypi-mlagents, [online] https://pypi.org/project/mlagents/0.18.0/ [25.08.2020]

Python: Python Allgemein, in: Python Dokumentation, [online] https://docs.pyhon.org/3/faq/general.html#what-is-python [5.08 2019]

Python: Python Anwendungen, in Python, [online] https://www.python.org/about/apps/ [5. 08 2019]

Python: Python für Anfänger, in: Python Wiki, [online] https://wiki.python.org/moin/BeginnersGuide [5. 08 2019]

Python: Python Lösungen und Erfolge, in Python,[online] https://www.python.org/success-stories/ [5. 08 2019]

Richard Salm / Karl Ernst Grund: Systeme für die Endoskopie, in Rüdiger Kramme (Hrsg.): Medizintechnik3.Aufl, Berlin, Deutschland: Springer, 2007, S. 350 ff.

Tensorflow: Warum Tensorflow, in Tensorflow,[online] https://www.tensorflow.org/ [5. August 2019]

Unity, BC Trainings-Parameter, in Unity ML-Agents Github,18.08.20, https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Configuration-File.md [18.08.2020]

Unity: Funktionen in: Unity, [online] https://docs.unity3d.com/Manual/ [4. 08 2019]

Unity: Installation Unity-ML-Agents, in: Unity ML-Agents Github, 5.08.2019, [online] https://github.com/Unity-Technologies/mlagents/blob/master/docs/Installation.md [30.07.2020]

Unity: Kernplattform, in: Unity, [online] https://unity.com/de/products/core-platform [4. 08 2019]

Unity: Lösungen, in: Unity, [online] https://unity.com/de/solutions [4. 08 2019]

Unity: ML-Agents Python Modul, in: Unity ML-Agents Github, 5. August 2019, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/ml-agents/setup.py [30.07.2020]

Unity: Nutzung Convolutional Neural Networks , in: Unity ML-Agents Github,16.08.2020 [online] https://github.com/Unity-Technologies/ml-agents/blob/4261b4bf15e9d394b5d2426bc238d92940ab7e7d/docs/ML-Agents-Overview.md#learning-from-cameras-using-convolutional-neural-networks [18.08.20]

Unity: Nutzung von Tensorboard, in: Unity-ML-Agents Github,24.07.20,[online] https://github.com/Unity-Technologies/ml-agents/blob/2d0eb6147c031c082522eb683e569dd99b4d65fb/docs/Using-Tensorboard.md [25.08.20]

Unity: PPO Trainings-Parameter, in Unity-ML-Agents Github,18.08.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Configuration-File.md [18.08.2020]

Unity: Produkte, in: Unity, [online] https://unity.com/de/products [4. 08 2019]

Unity: Proximal Policy Optimization, in Unity-ML-Agents Github,23.10.19,[online] https://github.com/Unity-Technologies/ml-agents/blob/3d7c4b8d3c1ad17070308b4e06bb57d4a80f9a0c/docs/Training-PPO.md [18.08.2020]

Unity: Überblick zum Machine Learning, in Unity-ML-Agents Github,13.06.19,[online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Background-Machine-Learning.md [18.08.2020]

Unity: Unity: A General Platform for Intelligent Agents, Publikation, San Francisco, USA: Unity Technologies, 2020, S.11 ff.

Unity: Unity 2018.4.22f1, in: Unity Downloads, [online] https://download.unity3d.com/download_unity/3362ffbb7aa1/UnityDownloadAssistant-2018.4.22f1.exe [4. Juli 2019]

Unity: Unity-ML-Agents Toolkit, in Unity-ML-Agents Github,16.07.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/4261b4bf15e9d394b5d2426bc238d92940ab7e7d/docs/ML-Agents-Overview.md#deep-reinforcement-learning [18.08.2020]

Unity: Unity-ML-Agents Toolkit, in Unity-ML-Agents Github, 16.07.20, [online] https://github.com/Unity-Technologies/ml-agents/blob/master/docs/ML-Agents-Overview.md#behavioral-cloning-bc [18.08.20]

9 Anhang

9.1 Anhang 1 : Namensnennung und Lizenz zum 3D-Körper

Quelle des 3D-Modells: https://sketchfab.com/3d-models/digestive-system-2db2c9090e1d4e1891826958398d324f#download (25.Aug.2020)
Titel des 3D-Modells: Digestive-system
Name des Erstellers: Genesis1
Lizenz: CC Attribution

Link zur Lizenz: https://creativecommons.org/licenses/by/4.0/legalcode.de
Link zur Lizenz-Zusammenfassung: https://creativecommons.org/licenses/by/4.0/deed.de#
Änderungen am 3D-Modell: Entfernt wurden die drei Kameras (Camera001, Camera001.Target, VRayDomeCamera001)
Lizenzträger und Urheber sind von jeder Haftung ausgeschlossen.

10 Externer Anhang (USB-Stick)

Externer Anhang 1: Ordner C#-Skripte

Innerhalb des Ordner C#-Skripte finden sich die verwendeten C#-Skripte CollisionInformation.cs, EndoscopeAgentScript.cs und FlipFaces.cs

Externer Anhang 2 : Ordner Hyperparamater

Innerhalb dieses Ordners findet sich die Datei der Hyperparameter mit dem Namen ThesisHyper.yaml

Externer Anhang 3 : Ordner Models

Hierin befinden sich die Modelle der Versuche, innerhalb weiterer Ordner. Diese Ordner haben den selben Namen wie die Trainingsversuche und tragen die Modelle mit dem Dateiformat .nn.

Externer Anhang 4 : Ordner Unity – Projekt

In diesem Ordner befindet sich der Ordner ThesisUnityFinal. Dieser beinhaltet das gesamte Unity-Projekt.

Externer Anhang 5: Ordner Videos

Dieser Ordner beinhaltet veranschaulichende Videos, des Projekts.

11 Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe.

Alle Stellen der Arbeit, die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

_____

Furtwangen 78120, 31.08.2020, Abd-Rahmen Al-Skaf