Reward Engineering an einer End-to-End Spurhaltung durch Reinforcement Learning

Daniel Leonid Riege

Department Computer Science, HAW Hamburg,
Berliner Tor 7, 20099 Hamburg

Abstract. End-to-End gesteuerte autonome Fahrzeuge durch Reinforcement Learning bieten im Vergleich zu End-to-End Ansätzen durch supervised learning einen Vorteil. Beim Reinforcement Learning macht das Fahrzeug eigene Erfahrungen und wertet diese anhand von Rewards. So kann dieses Fahrzeug auf mehr Situation reagieren, als beim anderen Ansatz, welcher rein auf richtigem Verhalten basiert. Das Reward Engineering ist dabei ein wichtiger Schritt, um das gewünschte Verhalten zu erzielen. In dieser Arbeit werden verschiedene Reward Designs vorgestellt und anhand ausgewählter Metriken und Analysen verglichen und bewertet. Dabei stellte sich heraus, dass ein Reward Design, welches sich mit dem Trainingsforschritt anpasst, die besten Ergebnisse erzielt und einer optimalen Spurhaltung nahe kommt.

Keywords: Reinforcement Learning · Spurhaltung · autonomes Fahren · End-to-End · Maschinelles Lernen · Reward Engineering

1 Motivation

Für eine autonome Spurhaltung eines Fahrzeuges gibt es bereits zahlreiche Ansätze. Darunter auch Verfahren mit Hilfe von neuronalen Netzen. Eine Variante ist dabei das End-to-End gesteuerte Auto. Man übergibt dem neuronalen Netz die komplette Aufgabe, so dass der Input die direkten Sensordaten sind und die Ausgabe die jeweilgen Werte für die Aktorik. Solche Verfahren können einerseits supervised trainiert werden. Dabei lernt das neuronale Netz anhand vom menschlichen Verhalten [1, 3, 12]. Andererseits gibt es aber auch Verfahren, welche Reinforcement Learning nutzen. Dabei lernt das neuronale Netz selber anhand von Rewards, welche von der Umgebung verteilt werden, welche Aktionen sinnvoll sind [6, 13].
Ein solches Reinforcement Learning Verfahren soll in dieser Arbeit umgesetzt werden. Dabei soll allerdings das Reward Engineering untersucht werden, da dies mit komplexeren Systemen zunehmend wichtiger wird [5]. Es sollen verschiedene Reward Designs vorgestellt und miteinander verglichen werden. Um solche Prozesse umzusetzen, wird eine Simulation verwendet. An dieser kann man, im Vergleich zu realen Umgebungen, ein gutes Reward Engineering betreiben. Wie bereits angesprochen nutzen herkömmliche End-to-End Verfahren die direkten Sensordaten. Es soll allerdings keine aufwändige 3D Simulation

genutzt werden, sondern eine einfache 2D. Daher soll der Input kein direkter, sondern ein bearbeiteter Kamera Stream werden. Dieser Kamera Stream soll bereits segmentierte Straßenmarkierungen in Vogelperspektive beinhalten. Das hat zusätzlich den Vorteil, dass man Zwischenergebnisse, in der gesamten Pipeline von Kamera, bis hin zu Segmentierung, bis hin zu Steuerung, kontrollieren kann.

2 Grundlagen

2.1 Deep Deterministic Policy Gradient

Für die Steuerung der Lenkung soll das Deep Deterministic Policy Gradient (DDPG) Verfahren zum Einsatz kommen. Dieser Algorithmus lernt eine Q-Function und eine Policy. DDPG ähnelt dabei dem Q-Learning, jedoch wird beim Q-Learning die optimale Aktion gewählt durch

$$a^*(s) = arg\max_a Q^*(s,a) \tag{1}$$

Das setzt voraus, dass $Q^*(s,a)$ optimal ist und alle actions bekannt sind. Zweiteres ist bei deterministischen Aktionsräumen der Fall. Im Falle der Spurhaltung, welche umgesetzt werden soll, ist der Aktionsraum allerdings kontinuirlich, genauer im Intervall [-1,1]. Daher wird DDPG für kontinuirliche Aktionen genutzt, da die Aktion durch ein zweites neuronales Netz bestimmt wird. Dort entsteht der Unterschied zum Q-Learning und dieser wird Actor-Critic genannt.
Ebenfalls werden in DDPG target networks verwendet. Ein target wird beschrieben durch:

$$r + \gamma \max_{a'} Q_\phi(s', a') \tag{2}$$

Da versucht wird den mean squared Bellman error (MSBE) zu verkleinern, wird die Q-funtion in Richtung dieses targets optimiert. Allerdings hängt der target von ϕ ab, welches eigentlich trainiert wird. Deshalb wird ein target network verwendet, welches eine Kopie des eigentlich Netzes ist, um das Training stabiler zu gestalten. Nach einer gewissen Anzahl von Schritten wird das target network mit dem eigentlich network synchronisiert. [7]

2.2 Action Saturation

Beim Actor-Critic Lernen kann es allerdings zu einer Action Saturation kommen. Das bedeutet, dass es am Anfang des Trainings zu einem explodierendem Gradienten kommen kann, wodurch die Gewichte gerade im letzten Layer ein starkes Update erfahren. So kann es dazu kommen, wenn das Gewicht im letzten Neuron sehr groß ist, dass die gewählte Aktion des Actors, durch Clipping auf den Aktionsraum, immer auf ein Maximum schlägt [16]. Also in diesem Fall immer auf -1 oder 1. Um dies zu Verhindern, kann man die Lernrate der Modelle verkleinern und ein Gradient Clipping einführen [9]. Dieses Gradient Clipping verhindert, dass der Gradient einen bestimmten Wert überschreitet. So kann ein explodierender Gradient gerade am Anfang des Trainings gut verhindert werden. Dieses Verfahren wird in den hier verwendeten Netzen angewendet.

2.3 Experience Replay Buffer

Der Experience Replay Buffer ist ein Speicher, welcher die Erfahrungen des Agenten speichert. Es werden Tuples aus vorherigem Zustand, gewählte Aktion, Reward und aktuellem Zustand gespeichert. Dieser Speicher kann ein sogenanntes catastrophic forgetting oder auch catastrophic inference verhindern. Das bedeutet, dass der Agent nach einer gewissen Zeit vergisst, was er einmal gelernt hatte. Dies ist vor allem ein Problem beim kontinuirlichen Lernen. Indem Erfahrungen aus der Vergangenheit gespeichert werden, gehen diese in späteren Trainingsschritten nicht verloren, da diese weiterhin ins Training einfließen. Ein Problem ist es aber, wenn der Speicher zu klein ist, um genug Erfahrungen zu speichern. Für diese Fälle gibt es verschiedene weitere Methoden, um dem Vergessen entgegenzuwirken [2, 4, 8, 11, 15]. Mit dem verwendetem Versuchsaufbau kann ein Speicher mit 50,000 Tuples verwendet werden, was für die späteren Experimente ausreichend ist. Andere Methoden kommen somit nicht zum Einsatz.

2.4 Exploration Noise

Um auch das Explorieren neuer Zustände für den Agenten zu ermöglichen, wird auf die gewählte Aktion des Actors ein Noise addiert. Dafür wird der Ornstein-Uhlenbeck-Prozess verwendet. Dieser ist definiert durch:

$$X_t = X_{t-1} \cdot \theta \cdot (\mu - X_{t-1})dt + \sigma dW_t \tag{3}$$

W_t ist dabei ein Zufallswert. θ, μ und σ sind dabei Stellgrößen, die man für den Anwendungszweck einstellen kann. Für den verwendeten Aktionsraum [-1,1] wurden gewählt: $\theta = 0.2$, $\mu = 0.0$ und $\sigma = 0.1$. So werden kleine Schwänkungen in der Lenkung erzeugt, die aber zu groß sind, um das Auto aus der Spur zu reißen.

3 Simulation

Anders als herkömmliche End-to-End Ansätze für das autonome Fahren soll keine Frontkamera benutzt werden, welche einen Blick auf die Straße hat, sondern stattdessen soll bereits eine segmentierte Fahrbahnrepräsentation verwendet werden. Das Kamerabild soll also ein Bild in Vogelpersktive sein, welches sich direkt vor dem Fahrzeug befindet und die erkannten Straßenmarkierungen farblich codiert anzeigt. Hintergrund ist, dass man den Teil der Straßenmarkierungenserkennung abgeben kann und so mehr Kontrolle über ein mögliches Verhalten der Fahrzeugsteuerung hat. In der Arbeit "Segmentierung von Straßenmarkierungen durch maschinelles Lernen für die Miniaturautonomie" wird eine mögliche Segmentierung vorgestellt, welche in Vogelperspektive hier als Kamerabild dienen könnte [14]. Ein Beispiel ist in Abbildung 3.1 zu sehen.

Die Anforderung an die Simulation ist somit so speziell, dass für diese Arbeit eine eigene Simulation, mit dem Namen tinycarlo, entwickelt wurde. Diese wird in diesem Abschnitt genauer beschrieben.

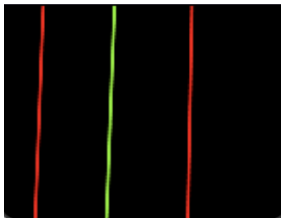


Fig. 3.1: Beispiel eines Kamerabildes in der tinycarlo Simulation. Das Bild befindet sich in Vogelpersktive unmittelbar vor dem Fahrzeug. Eine rote Markierung steht für eine Fahrbahnaußenkante und eine grüne Markierung für eine gestrichelte Linie [14]

3.1 Schnittstelle

Damit diese Simulation auch ausserhalb dieses Projektes Anwendung finden kann, wurde sie als OpenAI Gym Enviroment aufgebaut [10].

Aktorik und Sensorik In dieser Arbeit soll eine Spurhaltung aufgebaut werden, ohne Einfluss auf die Geschwindigkeit des Fahrzeuges. Daher ist der Action Space in dieser Simulation/Enviroment ein Wert im Intervall [-1,1], um die Lenkung zu steuern. Ein Wert von 1 bzw. -1 entspricht hierbei einem vollen Lenkeinschlag von 33 Grad nach rechts bzw. links. Die Geschwindigkeit bleibt konstant. Somit befindet sich das Auto immer in Bewegung.
Der Observation Space, also die Daten, die den State beschreiben, sind wie bereits beschrieben die segmentierten Straßenmarkierungen vor dem Fahrzeug. Die Auflösung ist variabel, allerdings wird ein Standardwert von 160x120 verwendet. Die Bilder sind im RGB Farbraum, da die Markierungen farblich codiert sind. Eine rote Markierung steht für eine Fahrbahnaußenkante und eine grüne Markierung für eine gestrichelte Linie [14].

```
1  import gym
2  import tinycarlo
3  env = gym.make("tinycarlo-v0")
4  observation = env.reset()
5  for _ in range(1000):
6      env.render()
7      action = [1.0]
8      observation, reward, done, info = env.step(action)
9      if done:
10         observation = env.reset()
```

```
11  env.close()
```

Code 3.1: Einfaches Beispiel zur Demonstration der vorgestellten Simulation. Hier werden ingesamt 1000 Simulationsschritte gegangen, in denen jeweils zufällige Akionen aus dem Action Space gewählt werden. Die normalen Gym Methoden wie step(), render(), reset() sind alle implementiert, sodass sich diese Simulation wie alle anderen OpenAI Gyms verwenden lässt.

Einstellungen Um die Simulation für verschiedene Anwendungszwecke zu verwenden, lassen sich viele Parameter einstellen. Zum einen die Größe der Zeitschritte in der Simulation. Dies wird in Frames per Second (FPS) angegeben. Also wie viele Bilder pro Sekunde z.B die Kamera liefern soll. Zum anderen lassen sich auch Fahrzeugparameter einstellen, wie die Spurbreite, der Radstand oder maximaler Lenkeinschlag.

Visualisierung In Code 3.1 wird ein Beispiel zur Verwendung gezeigt. Durch den Aufruf render() werden zwei Fenster erstellt. In einem wird das Kamerabild angezeigt, welches ebenfalls die Observation ist. Ein Beispiel ist in Abbildung 3.1 zu sehen. Das andere Fenster zeigt eine komplette Übersicht der Simulationsumgebung an, wie in Abbildung 3.2 zu sehen.

3.2 Kinematik

Der Experimentaufbau in dieser Arbeit ist simpel gehalten. Der Einfluss von Kräften wird nicht berücksichtigt, da die Geschwindigkeit konstant ist und angenommen wird, dass bei dieser Geschwindigkeit die Dynamik zu vernachlässigen ist. Somit wird lediglich die Kinematik betrachtet. Bei dem simulierten Fahrzeug handelt es sich um ein Einspurmodell. Das heißt, das Fahrzeug besitzt vorne eine lenkbare Achse. Der Bewegungsraum in längs- und querrichtung ist durch diese Lenkung beschränkt. Durch den Lenkwinkel kann bestimmt werden, wie der Kurvenradius zum aktuellen Zeitpunkt ist. Anhand dessen kann man durch lineare Transformationen die neue Position und Ausrichtung des Fahrzeuges bestimmen. Der Nullpunkt des Fahrzeuges ist in dieser Simulation die Mitte der hinteren Achse.
Theoretisch wäre es möglich, dass die Lenkung zum Zeitpunkt t voll links eingeschlagen ist und zum Zeitpunkt $t+1$ voll rechts eingeschlagen. Je nach Größe der Zeitschritte würde dies aber zu einer nicht realen Änderungsrate führen, da Servos auch eine gewisse Zeit brauchen, um den gewünschten Winkel zu erreichen. Daher ist eine weitere Stellgröße der Simulation die maximale Winkeländerung der Lenkung, welche in Grad pro Sekunde angegeben wird. In Abbildung **??** wird gezeigt, wie das Lenkverhalten aussieht.

3.3 Rewardsystem

Damit diese Simulation für Reinforcement Learning verwendet werden kann, braucht es ein Rewardsystem des Enviroments. In dieser Arbeit sollen mögliche

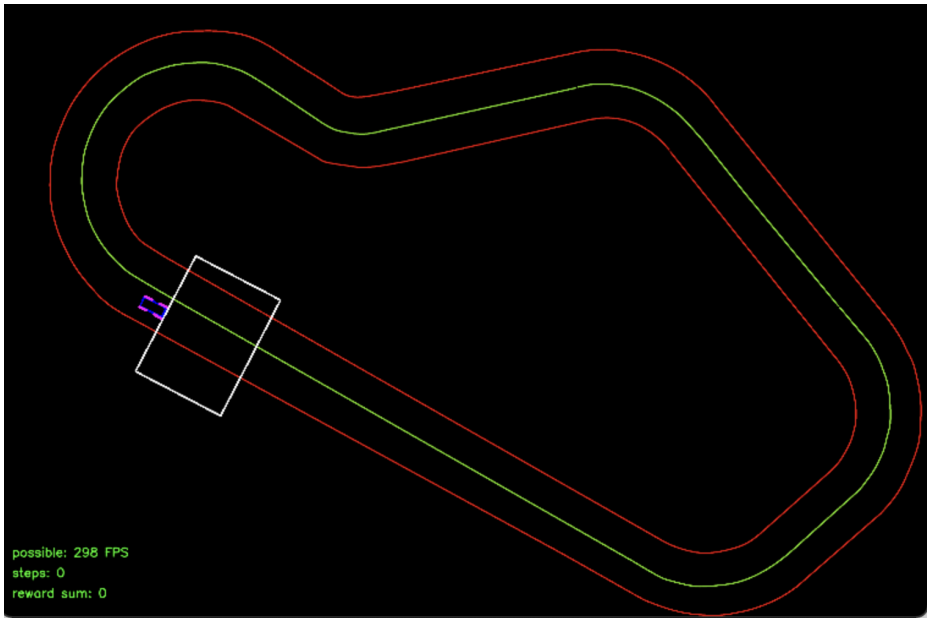


Fig. 3.2: Übersicht über die gesamte Simulationsumgebung. Unten links wird angegeben, wie viele Simulationsschritte dieser Durchlauf vorangeschritten ist oder wie schnell die Simulation rechnet (in FPS angegeben). Es wird der Streckenverlauf und das Fahrzeug angezeigt. In Lila sind die Reifen des Fahrzeuges dargestellt, welche auch den Lenkeinschlag anzeigen. Der weiße Kasten bildet den Rahmen des Kamerabildes, welches in Abbildung 3.1 zu sehen ist.

Reward Designs vorgestellt und getestet werden. Daher sind die Rewards konfigurierbar. Grundsätzlich werden zwei Arten angeboten. Zum einen lässt sich ein Rewardwert einstellen, wenn das Fahrzeug entweder die rote oder grüne Linie überschreitet. Zum anderen gibt es eine Ground Truth Trajektorie, welche sich je nach Fahrtrichtung immer auf der rechten Spur befindet. Diese Trajektorie ist in der jeweiligen Spur in der Mitte. Vom Nullpunkt des Fahrzeuges, Mittelpunkt der Hinterachse, aus, wird der Cross Track Error (CTE) berechnet. Anahand dessen lassen sich verschiedene Rewardkurven übergeben, die je nach Höhe des CTE einen anderen Reward zurückgeben.

4 Reward Design

In den Experimenten sollen verschiedene Reward Designs verwendet und verglichen werden. Im folgenden werden diese vorgestellt.

4.1 Bounding Reward

Bei diesem Reward Design kommt nicht die Ground Truth Trajektorie zum Einsatz, sondern lediglich die die Straßenmarkierungen bzw. deren Übertritte. So gibt es beim Übertritt der roten oder grünen Fahrbahnmarkierung einen Reward von -1, wie in Abbildung 4.1a zu sehen. Dieser gilt für jeden Zeitschritt unabhängig. Befindet sich das Fahrzeug also für 10 Zeitschritte auf der roten Fahrbahnmarkierung, so ist der summierte Reward nach 10 Zeitschritten -10.

4.2 Sparse Reward

Dieser Reward beträgt 1, wenn der CTE kleiner gleich 10 ist. Zum Vergleich, die Spurbreite beträgt ca. 160 Pixel, die gesamte Fahrbahn also 320 Pixel. Somit hat das Fahrzeug einen Raum von ca. 20 Pixeln, um einen Reward von 1 zu bekommen. Ansonsten ist der Reward 0. Anschaulich in Abbildung 4.1b dargestellt.

4.3 Linearer Reward

Beim linearen Reward wird eine lineare Kurve verwendet, um den Reward zu berechnen. An der Spitze, also CTE = 0, beträgt der Reward 1. Von da aus fällt der Reward linear mit ansteigendem CTE an. Berechnet wird der Reward also mit folgender Formel:

$$r = max(m \cdot CTE + 1, 0) \tag{4}$$

m gibt die Steigung an. In diesem Fall wurde eine Steigung von $m = -0.0095$ verwendet. Daraus ergibt sich die Reward Kurve, wie in Abbildung 4.1c zu sehen. Die Idee hinter diesem Reward Shaping ist, dass durch die Ansteigung des Rewards das Fahrzeug in diese Richtung durchs Training gezogen wird.

4.4 Exponentieller Reward

Wie beim linearen Reward, wird auch hier die Reward Kurve abhängig vom CTE geformt. Allerdings statt mit einer linearen Kurve, mit einer exponentell ansteigenden Kurve. Definiert durch folgende Formel:

$$r = b^{CTE} \tag{5}$$

Als Basis wurde $b = 0.93$ gewählt. Die entstehende Kurve ist in Abbildung 4.1d zu erkennen. Auch wird wird versucht das Fahrzeug durchs Training in Richtung der Ground Truth Trajektorie zu drücken, allerdings mit deutlich steilerem Verlauf. So nähert sich diese Kurve einem Sparse Reward.

4.5 Hybrider Reward

Den Linearen und Exponentiellen Reward kann man auch verbinden, in dem man die Reward Kurve über die Zeit verändert. So kann es sinnvoll sein, am Anfang des Trainings einen linearen Reward zu verwenden, damit das Fahrzeug

schnell lernt die Trajektorie zu verwenden. Bei voranschreitendem Training kann dann diese lineare Kurve langsam zu einer exponentiellen umgeformt werden, um genauere Optimierungen vorzunehmen.

$$r = \alpha \cdot b^{CTE} + (1 - \alpha) \cdot max(m \cdot CTE + 1, 0) \tag{6}$$

Für m und b werden die gleichen Werte genommen, wie in den obigen Kurven. α bestimmt also den Anteil der linearen und exponentellen Kurve. In Abbildung 4.1e ist diese Kurve dargestellt.

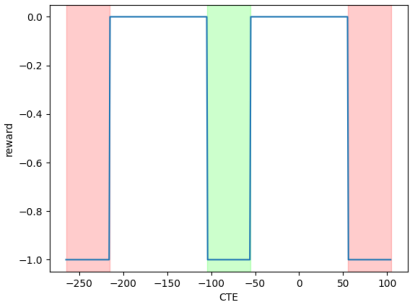
5 Experimente und Ergebnisse

5.1 Training

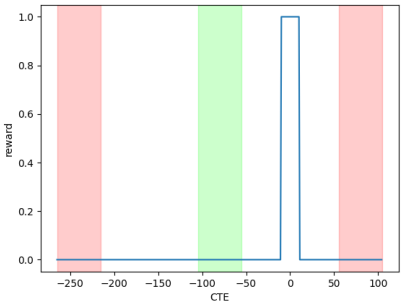
Beim Training startet das Fahrzeug zufällig an einer von 6 möglichen Startpositionen. 3 davon sind im Uhrzeigersinn, 3 gegen den Uhrzeigersinn. Jedes Training findet mit einem der vorgestellten Reward Designs statt und läuft über 300 Episoden. Für die hybride Rewardkurve ist die Schrittweite für alpha auf 1/300 gesetzt, sodass zu Anfang eine rein lineare Kurve verwendet wird und in der letzten Episode eine reine exponentielle. Die maximale Anzahl der Schritte ist auf 1000 begrenzt, was je nach Trajektorie ca 1 1/3 Runden entspricht. Bei einem CTE von 240, was ca. dem Überqueren der ganz linken Fahrbahnaußenkante gleich kommt, wird die aktuelle Episode abgebrochen.
In Abbildung 5.1 ist für jedes Reward Design der Verlauf des summierten Rewards im Durchschnitt zu sehen. Die absoluten Rewardwerte haben im Vergleich allerdings keine Aussagekraft, da jede Kurve anderen Reward verteilt. Somit ist dies kein geeignetes Maß zum Vergleich der Agenten. Diese Verläufe können allerdings ausschluss darüber geben, wie der Trainingsverlauf eines einzelnen Agenten ablief. Beim bounding Reward zum Beispiel kann man erkennen, dass ab ungefähr Episode 50 der Agent stagniert, da sich der Reward weder verbessert, noch verschlechtert. Es lässt also darauf schließen, dass immer der selbe Aktionsablauf in jeder Episode statt findet. Dies kann man auch noch später in Abbildung 5.2 gut erkennen. Beim sparse Reward und linearen Reward lässt sich erkennen, dass sich der Reward im Durchschnitt im Laufe verbessert und kein Abschwächen der Kurve zu erkennen ist. Ein weiteres Training über mehr Episoden könnte bei diesen Kurven daher sinnvoll sein. Generell kann man dennoch sagen, dass ein episodischer Reward weit unter 1000 auf einen Agenten schließen lässt, der die Spur nicht halten konnte und die Episode vorzeitig beendet wurde.

5.2 Evaluation

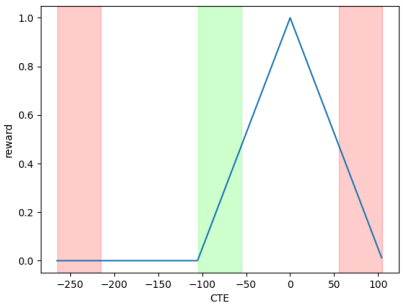
Trajektorienabweichung Wie erwähnt ist der episodische Reward keine gute Metrik, um die Agenten miteinander zu vergleichen. Daher wird als Metrik der durchschnittliche CTE eines Durchlaufes verwendet. Alle Agenten werden in diesem Schritt von der selben Position starten und genau 1000 Schritte



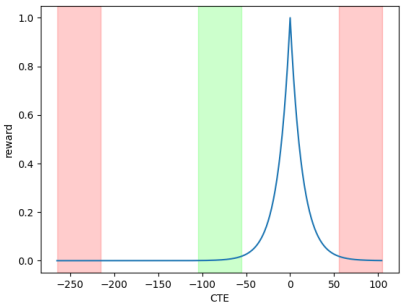
(a) Bounding Reward



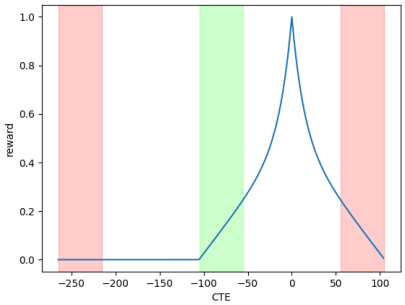
(b) Sparse Reward



(c) Linearer Reward mit $m = -0.0095$

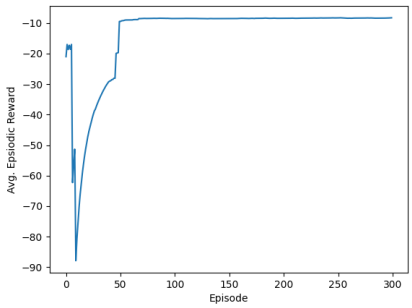


(d) Exponentieller Reward mit $b = 0.93$

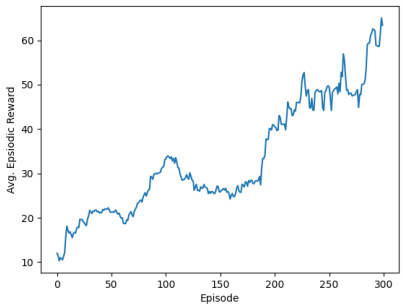


(e) Hybrider Reward mit $\alpha = 0.5$

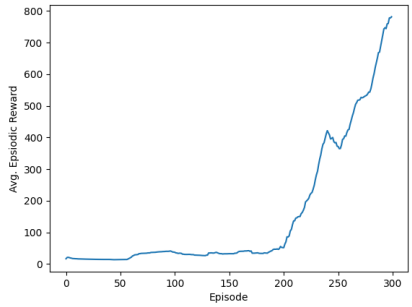
Fig. 4.1: Darstellung der vorgestellten Reward Designs. Der Nullpunkt beschreibt die Position der Ground Truth Trajektorie innerhalb der Fahrbahn. Die roten und grünen Bereiche markieren, innerhalb welchem CTE das Fahrzeug einer dieser Linien berühren würde. Die Breite der Bereiche ist somit von der Fahrzeugbreite abhängig, welche hier 50 beträgt. Anmerkung: Der CTE ist immer ein positiver Wert, hier werden nur zu Darstellungszwecken negative CTE angezeigt, auch wenn der Reward mit dem absoluten Wert berechnet wurde.



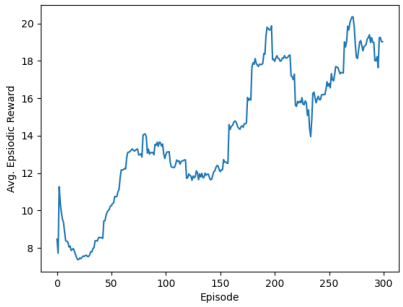
(a) Bounding Reward



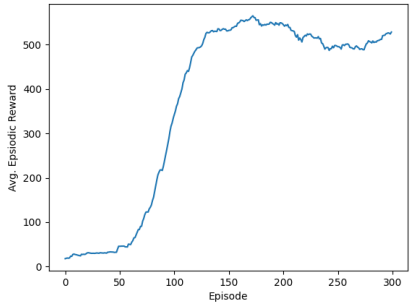
(b) Sparse Reward



(c) Linearer Reward



(d) Exponentieller Reward



(e) Hybrider Reward

Fig. 5.1: Gemittelter episodischer Reward der einzelnen Agenten im Training mit jeweiligem Reward Design. Es wird über 40 Epiosden gemittelt. Der episodische Reward ist der akkumulierte Reward innerhalb einer Episode. Die absoluten episodischen Rewards hängen vom Reward Design ab, da jedes Design anders gutmütig Rewards verteilt. Anmerkung: Für alle Designs, außer beim bounding Reward, gilt, dass bei max 1000 Steps der episodische Reward maximale 1000 sein kann. Bei diesem Reward wäre der Agent nach dem jeweiligem Reward Design optimal.

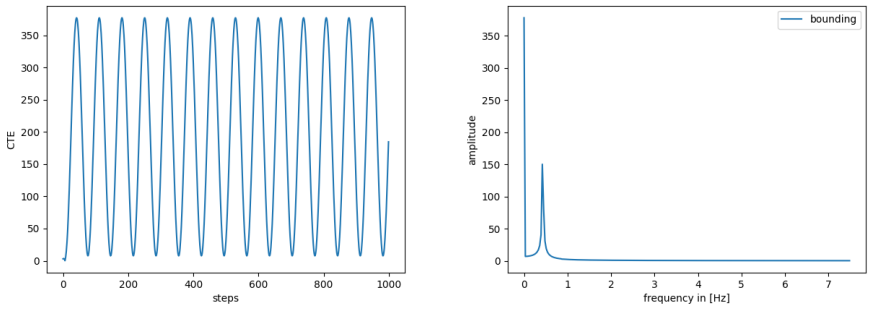
durchführen. Nach diesen 1000 Schritten wird der durchschnittliche CTE ermittelt, welcher als Metrik dient. In Tabelle 5.1 sind die Ergebnisse dieser Evaluierungsrunde angegeben. Die Agenten mit sparse und exponentiellem Reward Shaping verlassen demnach die Fahrbahn komplett. Das beste Ergebnis erzielt der Agent mit dem hybriden Reward Shaping, obwohl der durchschnittliche episodische Reward vom linearen Reward (Abbildung 5.1c) als Wert höher war.

Agent mit Reward Design	avg. CTE
Bounding	188.88
Sparse	3821.06
Linear	12.19
Exponentiell	5425.60
Hybrid	**6.51**

Table 5.1: Durchschnittliche CTEs einer Evaluierungsrunde über 1000 Scrhitte mit jeweiligem Agenten. Ab einem CTE von 240 kann man davon ausgehen, dass das Fahrzeug die Fahrbahn verlassen hat.

Lenkverhalten Als weiteres Maß für einen guten Agenten, kann das Lenkverhalten dienen. Als Beifahrer eines autonomen Autos erhofft man sich, dass keine abrupten Bewegungen stattfinden und das Auto bei gerader Strecke keine Schlangenlinien fährt. Dafür kann man die gewählten Actions des Agenten analysieren, allerdings muss das Fahrzeug auch Kurven fahren. Dadurch wird das Unterscheiden zwischen möglichen Oszillationen im Lenkverhalten und Kurveneinschlag anhand der gewählten Actions schwierig. Eine andere Möglichkeit ist wieder die Verwendung der Ground Truth Trajektorie mit cross track error (CTE). Mit Hilfe der Fast Fourier Transformation (FFT) lassen sich mögliche Schwingungen in den Daten ablesen.
In Abbildung 5.2 lässt sich bereits am CTE Verlauf erkennen, dass es eine Schwingung gibt. Da diese allerdings sehr gleichmäßig ist, kann man anhand dieser aber auch vermuten, dass der Agent immer die selbe Aktion wählt und das Fahrzeug somit im Kreis fährt.

Im vorherigen Vergleich schnitten die Agenten mit linearem und hybriden Reward am besten ab. Daher wird auch an diesen beiden das Lenkverhalten analysiert. Wie in Abbildung 5.3 zu sehen, hat der Agent mit hybridem Reward deutlich geringere Schwingungen. Daraus kann man schließen, dass dieser Agent ein deutlich ruhigeres Fahrverhalten hat. Schaut man sich Abbildung 5.3a an, kann man ebenfalls erkennen, dass die CTEs an den Peaks über 55 sind, was bedeutet, dass an all diesen Punkten das Fahrzeug die Spur verlassen hat. In Abbildung 5.3b ist der höhste CTE bei ca. 35. Somit blieb dieser Agend zusätzlich immer innerhalb der eigenen Spur.



(a) Bounding CTE (b) Bounding FFT

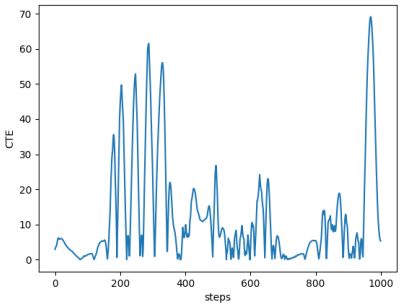
Fig. 5.2: Lenkverhaltensanalyse des Agenten mit bounding reward. Links der CTE über 1000 Schritte hinweg und rechts die Fast Fourier Transformation der CTE Daten. Bei ca. 0.5 Hz lässt sich eine Ausschlag feststellen, welcher der Schwingung links beschreibt.

6 Fazit

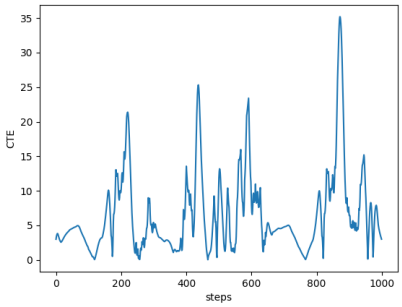
In dieser Arbeit wurden verschiedene Reward Designs vorgestellt und per DDPG angewendet. Die Analyse der episodischen Rewardkurven konnte festgestellt werden, wie das Training der einzelnen Agenten verlief, aber es kann kein verglich zwischen Agenten durchgeführt werden. Dafür wurden zwei Metriken verwendet. Mittels dem durchschnittlichen CTE wurde festgestellt, dass der Agent mit hybridem Reward die geringsten CTE hatte und immer in der Spur geblieben ist. Durch die Fourier Transformation wurde anschließend das Lenkverhalten analysiert, wobei festgestellt wurde dass der gleiche Agent auch die am wenigsten Schwingungen in der Lenkung aufweisen konnte.

6.1 Ausblick

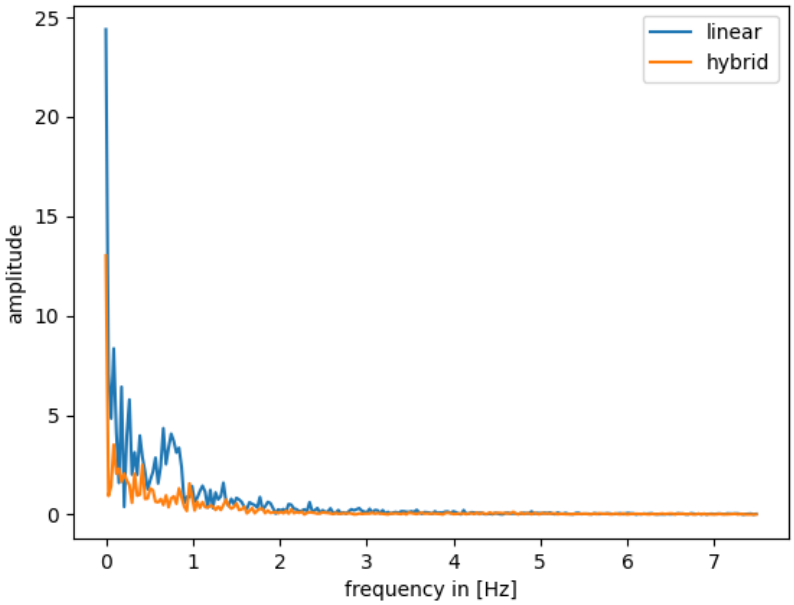
Es konnte eine grobe Einschätzung gegeben werden, welche Reward Designs gute Ergebnisse liefern können. Diese könnte man noch weiter optmieren, in dem zum Beispiel die Steigung der Rewardkurven verändert wird. Ebenfalls kann man das Szenario komplexer gestallten, so dass man nicht nur eine Spurhaltung realisiert, sondern auch die Kontrolle über Geschwindigkeit an den Agenten übergibt. Dafür muss die Simulation aber noch um die Dynamik des Fahrzeuges erweitert werden.



(a) Linear CTE



(b) Hybrid CTE



(c) Linear und Hybrid FFT

Fig. 5.3: Lenkverhaltensanalyse des Agenten mit linearem und hybridem Reward. Beim hybriden ist der Ausschlag bei Frequenzen > 0.0 geringer als beim linearen.

Referenzen

[1] Amini, Alexander ; Rosman, Guy ; Karaman, Sertac ; Rus, Daniela: Variational End-to-End Navigation and Localization. In: *2019 International Conference on Robotics and Automation (ICRA)* (2019), Mai, S. 8958–8964. – URL http://arxiv.org/abs/1811.10119. – Zugriffsdatum: 2022-02-20

[2] Atkinson, Craig ; McCane, Brendan ; Szymanski, Lech ; Robins, Anthony: Pseudo-Rehearsal: Achieving Deep Reinforcement Learning without Catastrophic Forgetting. In: *Neurocomputing* 428 (2021), März, S. 291–307. – URL http://arxiv.org/abs/1812.02464. – Zugriffsdatum: 2022-03-01. – ISSN 09252312

[3] Bojarski, Mariusz ; Yeres, Philip ; Choromanska, Anna ; Choromanski, Krzysztof ; Firner, Bernhard ; Jackel, Lawrence ; Muller, Urs: Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car. In: *arXiv:1704.07911 [cs]* (2017), April. – URL http://arxiv.org/abs/1704.07911. – Zugriffsdatum: 2022-02-20

[4] Cahill, Andy: *Catastrophic Forgetting in Reinforcement-Learning Environments*, University of Otago, Thesis, 2011. – URL https://ourarchive.otago.ac.nz/handle/10523/1765. – Zugriffsdatum: 2022-03-01

[5] Dewey, Dan: Reinforcement Learning and the Reward Engineering Principle. In: *AAAI Spring Symposia*, 2014

[6] Kendall, Alex ; Hawke, Jeffrey ; Janz, David ; Mazur, Przemyslaw ; Reda, Daniele ; Allen, John-Mark ; Lam, Vinh-Dieu ; Bewley, Alex ; Shah, Amar: Learning to Drive in a Day. In: *arXiv:1807.00412 [cs, stat]* (2018), September. – URL http://arxiv.org/abs/1807.00412. – Zugriffsdatum: 2022-01-11

[7] Lillicrap, Timothy P. ; Hunt, Jonathan J. ; Pritzel, Alexander ; Heess, Nicolas ; Erez, Tom ; Tassa, Yuval ; Silver, David ; Wierstra, Daan: Continuous Control with Deep Reinforcement Learning. In: *arXiv:1509.02971 [cs, stat]* (2019), Juli. – URL http://arxiv.org/abs/1509.02971. – Zugriffsdatum: 2022-02-23

[8] Lo, Yat L. ; Ghiassian, Sina: Overcoming Catastrophic Interference in Online Reinforcement Learning with Dynamic Self-Organizing Maps. In: *arXiv:1910.13213 [cs]* (2019), Oktober. – URL http://arxiv.org/abs/1910.13213. – Zugriffsdatum: 2022-03-01

[9] Mai, Vien V. ; Johansson, Mikael: Stability and Convergence of Stochastic Gradient Clipping: Beyond Lipschitz Continuity and Smoothness. In: *Proceedings of the 38th International Conference on Machine Learning*, PMLR, Juli 2021, S. 7325–7335. – URL https://proceedings.mlr.press/v139/mai21a.html. – Zugriffsdatum: 2022-03-04. – ISSN 2640-3498

[10] OpenAI: *Gym: A Toolkit for Developing and Comparing Reinforcement Learning Algorithms*. – URL https://gym.openai.com. – Zugriffsdatum: 2022-03-02

[11] Parisi, German I. ; Kemker, Ronald ; Part, Jose L. ; Kanan, Christopher ; Wermter, Stefan: Continual Lifelong Learning with Neural Networks: A Review. In: *Neural Networks* 113 (2019), Mai, S. 54–71. – URL https://www.sciencedirect.com/science/article/pii/S0893608019300231. – Zugriffsdatum: 2022-03-01. – ISSN 0893-6080

[12] Park, Mingyu ; Kim, Hyeonseok ; Park, Seongkeun: A Convolutional Neural Network-Based End-to-End Self-Driving Using LiDAR and Camera Fusion: Analysis Perspectives in a Real-World Environment. In: *Electronics* 10 (2021), Januar, Nr. 21, S. 2608. – URL https://www.mdpi.com/2079-9292/10/21/2608. – Zugriffsdatum: 2022-02-20. – ISSN 2079-9292

[13] Riedmiller, Martin ; Montemerlo, Mike ; Dahlkamp, Hendrik: Learning to Drive a Real Car in 20 Minutes. In: *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, Oktober 2007, S. 645–650

[14] Riege, Daniel: *Segmentierung von Straßenmarkierungen durch maschinelles Lernen für die Miniaturautonomie*, HAW Hamburg, Bachelorarbeit, Dezember 2021. – URL https://autosys.informatik.haw-hamburg.de/publication/2021danielriege/. – Zugriffsdatum: 2022-03-02

[15] Rostami, Mohammad ; Kolouri, Soheil ; Pilly, Praveen K.: Complementary Learning for Overcoming Catastrophic Forgetting Using Experience Replay. In: *arXiv:1903.04566 [cs, stat]* (2019), Mai. – URL http://arxiv.org/abs/1903.04566. – Zugriffsdatum: 2022-03-01

[16] Wang, Che ; Wu, Yanqiu ; Vuong, Quan ; Ross, Keith: Striving for Simplicity and Performance in Off-Policy DRL: Output Normalization and Non-Uniform Sampling. In: *arXiv:1910.02208 [cs, stat]* (2020), Dezember. – URL http://arxiv.org/abs/1910.02208. – Zugriffsdatum: 2022-03-04