Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Ausarbeitung Anwendungen 2

Ivo Nikolov

Steuerungsoptimierung eines autonomen Fahrzeugs
mittels Reinforcement Learning

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Inhaltsverzeichnis

1 Einleitung

Im Forschungsprojekt FAUST aus dem Department für Informatik der Hochschule für Angewandte Wissenschaften Hamburg werden Technologien für Fahrerassistenz- und Autonome Systeme entwickelt und entworfen (FAUST, 2010). Autonomes Fahren wurde auf Autobahnen und Landstraßen seit Anfang der neunziger Jahre in verschiedenen Projekten erforscht. Am Markt verfügbare Fahrerassistenzsysteme greifen bereits heute aktiv in die Fahrzeuglängs- und Querführung ein. Für die Zukunft ist der Automatisierung in der Fahrzeugführung von der reinen Assistenz hin zu automatischen Fahrfunktionen denkbar (Weiser u. a., 2009).

Der Hochschulwettbewerb Carolo-Cup (Carolo-Cup), einer der Schwerpunkte innerhalb des Projektes FAUST bietet Studententeams die Möglichkeit, sich mit der Entwicklung und Umsetzung von autonomen Modellfahrzeugen auseinander zu setzen. Beim Wettbewerb müssen bestimmte Fahraufgaben autonom, möglichst schnell und fehlerfrei bewältigt werden und die erarbeiteten Konzepte in Präsentationen erläutert werden. Laut (Carolo-Cup, 2010) sollen die Fahrzeuge bei der dynamischen Disziplin Rundstrecke autonom drei Minuten lang auf einem unbekannten Rundkurs so weit wie möglich fahren.

Im Carolo-Cup 2010 eingesetzten Verfahren (Nikolov, 2009) lokalisiert kontinuierlich ein Ziel, dass das Fahrzeug verfolgen soll. Um den Lenkwinkel in Bezug auf das Ziel zu bestimmen wird der *Pure Pursuit* Algorithmus (Coulter, 1992) verwendet. *Pure Pursuit* basiert auf die Krümmung des Kreisbogens, der Fahrzeug und Ziel verbindet (vgl. Abbildung 1).

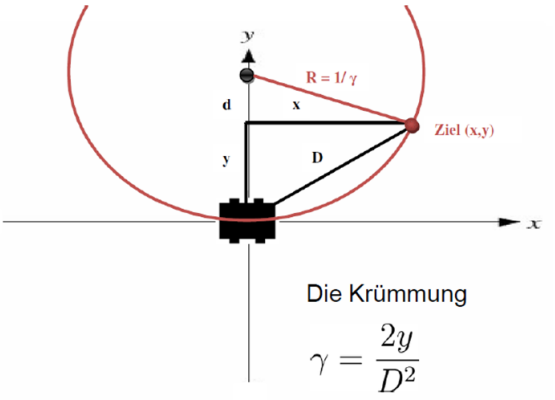

Abbildung 1: Lenkwinkelermittlung mittels *Pure Pursuit*

Das Verfahren berücksichtigt nicht das dynamische Verhalten des Fahrzeugs und schon bei einer Geschwindigkeit von 1,5 m/s wird die Fahrbahnverfolgung unstabil aufgrund Schwingungen im Lenkungssystem. Außerdem liegt das zu verfolgende Ziel immer in einer

konstanten Entfernung vor dem Fahrzeug. Dennoch ist es vorteilhafter die Zielentfernung dynamisch zu berechnen, weil die bestgeeignete Zielentfernung mit dem aktuellen Kurvenradius der Fahrspur gebunden ist.

Um das Verfahren zu optimieren soll ein Regler mittels Bestärkenden Lernens (engl.*Reinforcement Learning*) entworfen werden, der das dynamische Verhalten des Fahrzeugs betrachtet und ein optimiertes Lenkungsverhalten erreicht.

2 Vorstellung vergleichbarer Arbeiten

2.1 *Learning to Drive a Real Car in 20 Minutes*

In (Montemerlo u. a., 2007) wird vorgestellt wie ein Fahrzeug innerhalb von 20 Minuten lenken "lernt". Die autonome Fahrzeugquerführung erfolgt anhand eines Lenkungsreglers, der die Abweichung zur Sollspur minimiert. Das Verfahren basiert auf der *Neural Fitted Q Iteration (NFQ)* (Riedmiller, 2005) und ist komplett datengetrieben d.h. keine Simulation oder Fahrzeugmodell sind erforderlich.

2.1.1 *Q-Learning* und Neuronale Netze

Q-Learning ist eine modellfreie Methode des Bestärkenden Lernens, die iterativ Zustands-Aktionspaare bewertet, um eine optimale Strategie zu erreichen. Nach jeder Systembeobachtung wird der entsprechende Q-Wert wie folgt aktualisiert:

$$Q_{k+1}(s,a) := (1-\alpha)Q(s,a) + \alpha(c(s,a) + \gamma min_b Q_k(s',b)) \quad (1)$$

wobei s die aktuelle Zustand, a die ausgeführte Aktion und s' die Folgezustand repräsentieren. α stellt dabei den Lernfaktor dar und $0 <= \gamma <= 1$ die Skontorate (Sutton und Barto, 1998).

Wenn Zustände und Aktionen diskret sind, werden die Q-Werte tabellarisch verwaltet. Bei kontinuierlichen Zustands- und Aktionsräumen kann eine Diskretisierung angewandt werden. Diskretisierung ist aber für hochdimensionale Probleme nicht geeignet (Ng, 2008). Eine andere Alternative ist das Ganze als Regressionsproblem zu betrachten. Hier geht es um eine Aufgabenstellung des überwachtes Lernens, wobei ein Regressor $Q(s,a|\theta)$ definiert wird, der s und a als Eingabe nutzt und durch einen Vekor von Parametern θ parameterisiert wird, um die Q-Werte zu approximieren (Alpaydin, 2004). Hierzu können z.B. künstliche neuronale Netze eingesetzt werden. Das Problem bei der on-line Implementierung dieses Ansatzes (beim Training werden die Parameter nach jeder Aktion aktualisiert) ist, dass jede einzige Aktualisierung der Parameter die gesamte Approximationsleistung

beeinträchtigen kann. Aus diesem Grund kann es sehr lange dauern bis die Q-Funktion erfolgreich approximiert wird (Riedmiller, 2005).

Es wurde bewiesen, dass unter gewissen Voraussetzungen *Q-Learning* zum Optimum konvergiert. Obwohl dieser Beweis für Konvergenz bei kontinuierlichen Zustände und Aktionen nicht mehr gilt, es existieren viele Anwendungen, die neuronale Q-Funktionen erfolgreich nutzen.

2.1.2 *Neural Fitted Q Iteration*

Im Gegensatz zu der on-line Implementierung, approximiert der NFQ-Algorithmus die Q-Funktion off-line, d.h. um einen Trainingsdatensatz zu bilden werden zuerst Trainingsdaten gesammelt und dann insgesamt erlernt. Weiterer Vorteil generell für off-line Methoden ist, dass auch fortgeschritten Techniken des überwachten Lernens anwendbar sind.

Die Trainingsdaten werden in Tripel der Form (s, a, s') gespeichert. Hier repräsentiert s den ursprünglichen Zustand, a die gewählte Aktion und s' der Folgezustand. Der Trainingsdatensatz wird mit D bezeichnet.

NFQ ist eine Instanz der *Fitted Q Iteration* Familie von Algorithmen (Ernst u. a., 2005), wobei der Regression-Algorithmus durch ein neuronales Netz realisiert wird. Der Algorithmus ist in Abbildung 2 dargestellt. Er besteht aus zwei großen Schritten: Die Generation des Trainingssets P und das Trainieren des neuronalen Netzes mit s (Zustand) und u (Aktion) als Eingabe und die Kosten dieser Transition summiert mit den minimalen geschätzten Kosten für den Folgezustand als Ausgabe.

NFQ_main() {
input: a set of transition samples D; output: Q-value function Q_N
k=0
init_MLP() $\rightarrow Q_0$;
Do {
generate_pattern_set $P = \{(input^l, target^l), l = 1, \ldots, \#D\}$ where:
$input^l = s^l, u^l,$
$target^l = c(s^l, u^l, s'^l) + \gamma\, min_b Q_k(s'^l, b)$
Rprop_training$(P) \rightarrow Q_{k+1}$
k:= k+1
} WHILE $(k < N)$

Abbildung 2: *Neural Fitted Q Iteration*

In der ersten Iteration des NFQ-Algorithmus werden die Kosten $c(s,u,s')$ für die Zustands-Aktionspaare approximiert. Aus diesem Grund benutzt der Regression-Algorithmus die Zustands-Aktionspaare als Eingaben und die beim Training ermittelte Kosten als Ausgaben. In jeder weiteren Iteration bleiben die Eingaben unverändert und die Ausgaben werden in Bezug auf die im vorigen Schritt approximierte Q-Funktion aktualisiert.

Um den Algorithmus zu unterbrechen kann der Anzahl der Iterationen fest definiert werden. Andernfalls iteriert der Algorithmus so lange bis die Q-Funktion sich nicht mehr stark ändert (Ernst u. a., 2005). Dieses Abbruchkriterium ist aber nur dann anwendbar, wenn der Folge Q_N konvergiert.

2.1.3 Der RL-Lenkungsregler

Um autonomes Fahren zu gewährleisten minimiert der RL-Regler die Abweichung zur Sollspur indem er das Lenkungssystem des Fahrzeugs regelt. Der Schwerpunkt ist diese Abweichung oder anders ausgedrückt *cross-track-error (cte)* kleiner als 0.5 m zu halten. Der RL-Lenkungsregler wird mittels der NFQ Methode gelernt. Die Aufgabe des entstehenden Reglers besteht darin, die Kosten zu minimieren. Die Kostenfunktion ist wie folgt definiert:

$$c(s,u) = \begin{cases} 0 & \text{, wenn } |cte| < 0.05m \text{ (Erfolg)} \\ +1 & \text{, wenn } |cte| < 0.5m \\ 0,01 & \text{, sonst} \end{cases} \quad (2)$$

Basiert auf (Hoffmann u. a., 2007) wird der kinematische und dynamische Fahrzeugszustand durch folgende 5 kontinuierliche Variablen beschrieben:

- cte: Abweichung zur Sollspur
- $\dot{cte}$: erste Ableitung der cte
- v: Geschwindigkeit
- $heading\ error$: Differenz zwischen Gierwinkel und Krümmung der Fahrbahn
- $yaw - rate - matching$: Differenz zwischen Beschleunigung des Gierwinkels und Beschleunigung der Fahrbahnkrümmung

Anhand dieser Zustandsinformation kann der gelernte Regler den besten Stellwert für die Lenkung bestimmen, indem er die Kosten für unterschiedliche Stellwerte vergleicht.

Zur Realisierung dieses Ansatzes ist eine Diskretisierung der Aktionen (Lenkungsstellwerte) erforderlich. Das kann aber zu einem instabilen Lenkungsverhalten führen. Um diese Problematik zu lösen kommt ein *dynamic output element* mit einem Integrator (I-DOE) (Riedmiller, 1997) zum Einsatz. Das I-DOE summiert die Regleraktionen kontinuierlich und dessen Zustand lässt sich durch

$$s_{DOE}(t) = s_{DOE}(t-1) + u'(t) \tag{3}$$

ausdrücken, wobei $u'(t)$ die vom Regler ausgewählte Aktion repräsentiert und der I-DOE Ausgang $u(t) = s_{DOE}$ als Stellwert für die Lenkung benutzt wird. Der Ausgang wird gleichzeitig als Eingang für den Regler verwendet, damit Information über den gesamten Systemzustand vorhanden ist. Der daraus resultierenden I-DOE-Regler erreicht ein sehr glattes Verhalten obwohl nur 5 Aktionen angewandt wurden (Montemerlo u. a., 2007).

2.1.4 Ergebnisse

In diesem Kapitel beschriebenen Verfahren wurde auf einem Fahrzeug erfolgreich getestet. Bei einer Geschwindigkeit bis zu 9 m/s wurde schon in 16 Minuten ein RL-Lenkungsergler gelernt, der die Abweichung zur Sollspur kleiner als 0.5 m innerhalb einer vollen Runde der Teststrecke halten könnte (vgl. Abbildung 3).

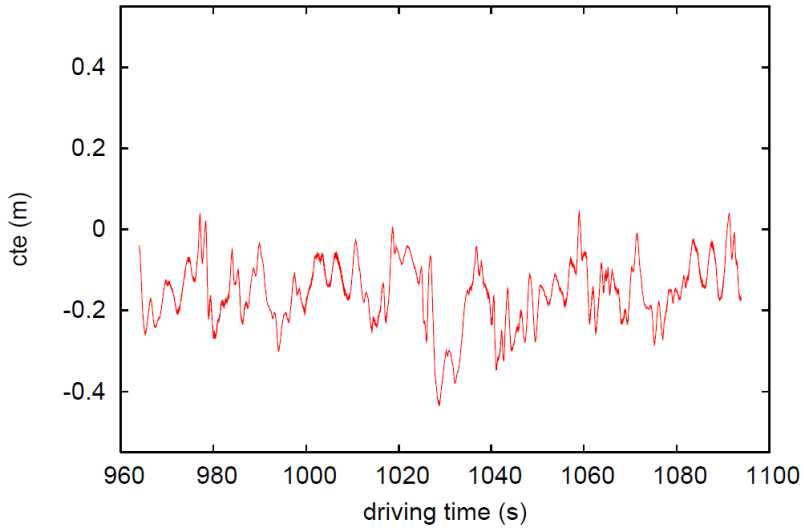

Abbildung 3: *Lenkungsverhalten des Reglers nach 16-Minuten-Training*

2.2 *An Application of RL to Aerobatic Helicopter Flight*

Regelung eines Helikopters wird allgemein als ein sehr anspruchsvolles Thema betrachtet. (Abbeel u. a., 2007) präsentiert die erste erfolgreiche Durchführung von vier autonomen Helikopter-Manövern. Die RL-Aufgabe wird als *Linear Quadratic Regulator (LQR)* modelliert und die optimale Strategie für diesen LQR mittels *Differential Dynamic Programming (DDP)* berechnet.

2.2.1 Helikoptermodell

Zur Verfolgung dieses Ansatzes muss als erstes ein Helikoptermodell gelernt werden. Zu diesem Zweck kommt der Apprenticeship-Algorithmus (Abbeel und Ng, 2005) zum Einsatz. Der Algorithmus besteht aus folgenden 3 Schritten:

1. Flugdaten werden gesammelt indem ein Pilot den Helikopter manuell steuert, um die gewünschte Manöver zu realisieren. Anhand der gesammelten Daten wird ein Helikoptermodell erlernt.

2. Basiert auf dem im 1. gelernten Modell wird ein Regler in einer Simulation entworfen.

3. Falls dieser Regler auch auf dem Helikopter funktioniert wird er weiter verwendet, sonst wird ein verbessertes Modell anhand der gesammelten Daten erlernt und dann wieder zu Schritt 2.

Das gelernte Modell sollte bezüglich Regler-Eingänge und Zustand des Helikopters Beschleunigungen berechnen können (Abbeel u. a., 2006). Diese Beschleunigungen können dann integriert werden, um zukünftige Zustände zu ermitteln.

2.2.2 Reglerentwurf

Die RL-Aufgabe kann mit Hilfe eines Markovschen Entscheidungsprozess (engl. Markov decision prozess, MDP) modelliert werden. Ein MDP ist ein Tupel $(S, A, T, H, s(0), R)$, wobei

- S eine Menge von Zustände,
- A eine Menge von Aktionen,
- T Wahrscheinlichkeit eines Zustandsübergangs $s \to s'$ bei Aktion a,
- H den Horizont oder die Anzahl der zu berücksichtigte Zeitschritte,
- $s(0) \in S$ der Ausgangszustand und
- $R : S \times A \to \Re$ die Belohnungsfunktion ist.

Die Lösung eines MDP ist eine Strategie $\pi = (\mu_0, \mu_1, \cdots, \mu_H)$, ein Tupel von Zuordnungen $S \to A$ für jede $t = 0, \ldots, H$.

LQR ist ein Spezialfall von MDPs, der die optimale Strategie effizient ermittelt. In LQR repräsentiert $S = \Re^n$ die Menge der Zustände und $A = \Re^p$ die Menge der Aktionen. Das dynamische Modell wird wie folgt definiert:

$$s(t+1) = A(t)s(t) + B(t)u(t) + w(t) \tag{4}$$

wobei für alle $t = 0, \ldots, H$: $A(t) \in \Re^{n \times n}$, $B(t) \in \Re^{n \times p}$ und $w(t)$ das Offset ist. Die Belohnung für Ausführung der Aktion a in Zustand s ist gleich

$$-s(t)^T Q(t)s(t) - u(t)^T R(t)u(t). \tag{5}$$

Hier sind $Q(t)$ und $R(t)$ positive Matrizen, die die Belohnungsfunktion parametrisieren. Aus der Definition folgt, dass $s = (0, \ldots, 0_n)$ immer die maximale Belohnung bekommt. Zur Verfolgung einer gewünschten Zustandstrajektorie $s_0^*, \ldots, s_H^*$ lässt sich aber LQR leicht anpassen indem die Zustandsabweichung $e(t) = s(t) - s^*(t)$ statt $s(t)$ verwendet wird (Anderson und Moore, 1989).

Um die optimale Strategie für MDP mit kontinuierlichen Zustandsraum zu ermitteln iteriert *Differential dynamic programming (DDP)* die folgenden 2 Schritte:

1. Mittels der aktuellen Strategie werden eine lineare Approximation der Dynamik und eine quadratische Approximation der Belohnungsfunktion um der Zustandstrajehtorie erstellt.

2. Die optimale Strategie für LQR wird bezüglich der Änderungen in 1 aktualisiert. Zurück zu Schritt 1.

2.2.3 Ergebnisse

Diese RL-Methode wurde erfolgreich für unterschiedliche Helikopter-Manövern angewandt (Abbeel u. a., 2007). Videos von allen Manövern sind unter (stanfordhelicopter) verfügbar.

3 Abgrenzung zu eigenen Arbeitszielen und Methoden

Die im Abschnitt 2.1 beschriebene Methode verwendet keine Simulation oder Fahrzeugmodell um die Q-Funktion zu approximieren. Im Gegenteil dazu setzt der eigene Ansatz *Pure Pursuit* in der Trainingsphase ein. Dadurch wird ein relativ stabiles Lenkungsverhalten gewährleistet d.h. das Fahrzeug bleibt dauerhaft auf der zu verfolgenden Spur. Außerdem minimiert sich gleichzeitig die Trainingszeit, weil der Aktionsraum begrenzt wird. Das beeinflusst auch den resultierenden Regler, da zur Bestimmung des optimalen Stellwerts mittels des trainierten neuronales Netz derselbe Aktionsraum verwendet wird.

Aktuell werden im FAUST Projekt Fahrspur- und Odometrie-basierte Systeme zur Selbstlokalisierung und Kartierung entwickelt (Rull, 2010), (Nowacki, 2010). Dadurch besteht die Möglichkeit ein Regler zur Verfolgung einer gewünschten Zustandstrajektorie $s_0^*, \ldots, s_H^*$ ähnlich wie in Abschnitt 2.2 beschriebene Methode zu entwickeln.

4 Zusammenfassung

In dieser Ausarbeitung wurden zwei Verfahren zum Entwurf eines Reglers mittels *Reinforcement Learning* vorgestellt.

Das erste Verfahren basiert auf dem NFQ Algorithmus und ist komplett datengetrieben d.h. keine Simulation oder Fahrzeugmodell ist erforderlich. NFQ ist eine Instanz der Fitted Q Iteration Familie von Algorithmen, wobei der Regression-Algorithmus durch ein künstliches neuronales Netz realisiert wird (vgl. Abschnitt 2.1).

Um unterschiedliche Helikopter-Manövern autonom zu realisieren, wird ein Regler mittels *Reinforcement Learning* entwickelt. Die RL-Aufgabe wird als *Linear Quadratic Regulator*

(LQR) modelliert und die optimale Strategie für diesen LQR mittels *Differential Dynamic Programming (DDP)* berechnet (vgl. Abschnitt 2.2).

5 Aktueller Stand und Ausblick

Im Projekt 1 (Nikolov, 2010) wurde das Verfahren zur Fahrbahnverfolgung (vgl. Kapitel 1) mittels der *Neural Fitted Q Iteration* Methode optimiert und auf einem Modellfahrzeug des VW Touareg im Maßstab 1:10 getestet. Innerhalb von nur 3 Minuten bei einer Geschwindigkeit zwischen 0.8 m/s und 1.3 m/s wurde ein Lenkungsregler gelernt, der erfolgreich die Abweichung zur Sollspur minimiert (vgl. Abbildung 4). Dieser Regler ermöglicht eine autonome Fahrbahnverfolgung auf der Teststrecke bei einer mittleren Geschwindigkeit von 2,2 m/s.

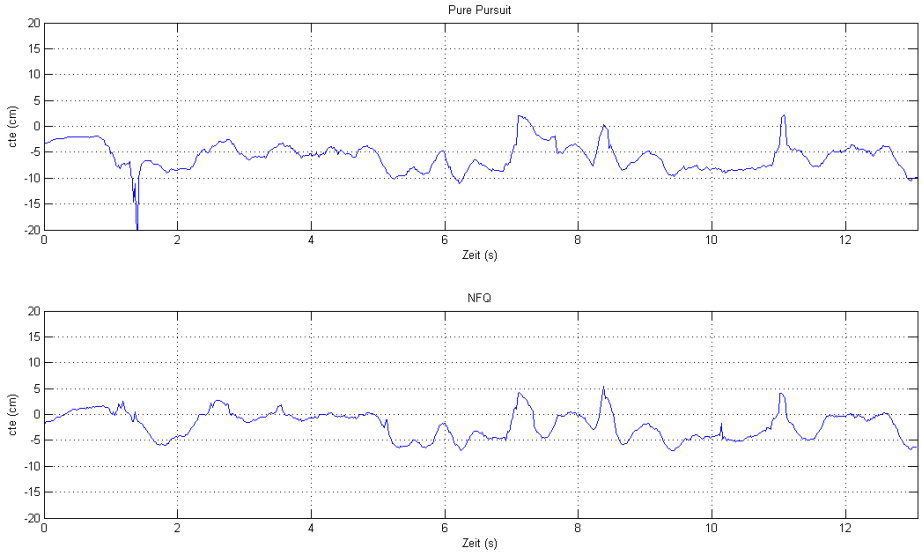

Abbildung 4: *Lenkungsverhalten in einer vollen Runde der Teststrecke*

Um Trainingsdaten für das neuronale Netz zu produzieren, wurde in der Trainingsphase *Pure Pursuit* eingesetzt. Es ist aber vorteilhafter schon beim Training das bereits existierende Wissen über das Lenkungsverhalten zu nutzen, um die Explorationsstrategie zu optimieren. Aus diesem Grund ist es denkbar im Projekt 2 ein System zu entwickeln, das die Präzision der Fahrbahnverfolgung analysiert und die NFQ Methode in der Trainingsphase unterstützt. Das wird das Lernen beschleunigen und zu einer genaueren Approximation der Q-Funktion führen.

Abbildungsverzeichnis

Literatur

[Abbeel u. a. 2006] ABBEEL, P. ; GANAPATHI, Varun ; NG, Andrew Y.: Learning vehicular dynamics with application to modeling helicopters / Computer Science Dept. Stanford University. 2006. – Forschungsbericht

[Abbeel und Ng 2005] ABBEEL, P. ; NG, Andrew Y.: Exploration and apprenticeship learning in reinforcement learning / Computer Science Dept. Stanford University. 2005. – Forschungsbericht

[Abbeel u. a. 2007] ABBEEL, Pieter ; COATES, Adam ; QUIGLEY, Morgan ; NG, Andrew Y.: An Application of Reinforcement Learning to Aerobatic Helicopter Flight / Computer Science Dept. Stanford University. 2007. – Forschungsbericht

[Alpaydin 2004] ALPAYDIN, Ethem: *Introduction to Machine Learning*. The MIT Press Cambridge, Massachusetts London, England, 2004

[Anderson und Moore 1989] ANDERSON, B. ; MOORE, J.: *Optimal Control: Linear Quadratic Methods*. Prentice-Hall, 1989

[Carolo-Cup] CAROLO-CUP: *Homepage des Carolo-Cup Wettbewerbs*. – URL `http://www.carolocup.de`

[Carolo-Cup 2010] CAROLO-CUP: *Carolo-Cup Regelwerk*. 2010. – URL `http://www.carolo-cup.de/uploads/media/Regelwerk2011.pdf`

[Coulter 1992] COULTER, R. C.: Implementation of the Pure Pursuit Tracking Algorithm / Robotics Institute, Carnegie Mellon University. 1992. – Forschungsbericht

[Ernst u. a. 2005] ERNST, D. ; ; WEHENKEL, L. ; GEURTS, P.: *Tree-based batch mode reinforcement learning. Journal of Machine Learning Research*. 2005. – URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.63.7705&rep=rep1&type=pdf`

[FAUST 2010] FAUST: *FAUST Homepage*. 2010. – URL `http://www.informatik.haw-hamburg.de/faust.html`

[Hoffmann u. a. 2007] HOFFMANN, Gabriel M. ; TOMLIN, Claire J. ; MONTEMERLO, Michael ; THRUN, Sebastian: Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing / Stanford University. 2007. – Forschungsbericht

[Montemerlo u. a. 2007] MONTEMERLO, Mike ; DAHLKAMP, Hendrik ; RIEDMILLER, Martin: Learning to Drive a Real Car in 20 Minutes. In: *FBIT conference, Jeju, Korea. Special Track on autonomous robots.*, URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.70.3532&rep=rep1&type=pdf, 2007

[Ng 2008] NG, Andrew: *Machine Learning (Stanford)*. 2008. – URL http://www.youtube.com/watch?v=-ff6l5D8-j8&feature=PlayList&p=A89DCFA6ADACE599&playnext_from=PL&index=18

[Nikolov 2009] NIKOLOV, Ivo: *Verfahren zur Fahrbahnverfolgung eines autonomen Fahrzeugs mittels Pure Pursuit und Follow-the-carrot*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2009

[Nikolov 2010] NIKOLOV, Ivo: *NFQ zur Optimierung eines Lenkungsreglers*, Hochschule für Angewandte Wissenschaften Hamburg, Projekt 1 Ausarbeitung, 2010

[Nowacki 2010] NOWACKI, Filip: *Überwachung und Führung autonomer Modellfahrzeuge basierend auf einer geometrischen Umgebungskartierung*, Hochschule für Angewandte Wissenschaften Hamburg, Projekt 2 Ausarbeitung, 2010

[Riedmiller 1997] RIEDMILLER, Martin: Generating continuous control signals for reinforcement controllers using dynamic output elements. In: *European Symposium on Artificial Neural Networks, Bruges*, URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.9217&rep=rep1&type=pdf, 1997

[Riedmiller 2005] RIEDMILLER, Martin: Neural Fitted Q Iteration - First experiences with a data efficient neural Reinforcement LearningMethod. In: *European Conference on Machine Learning, Porto, Portugal*, URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.72.1193&rep=rep1&type=pdf, 2005

[Rull 2010] RULL, Andrej: *Fahrspur- und Odometrie-basierte Selbstlokalisierung und Kartierung (SLAM)*, Hochschule für Angewandte Wissenschaften Hamburg, Projekt 2 Ausarbeitung, 2010

[stanfordhelicopter] STANFORDHELICOPTER: *Stanford Autonomous Helicopter YouTube Channel*. – URL http://www.youtube.com/stanfordhelicopter

[Sutton und Barto 1998] SUTTON, R. S. ; BARTO, A. G.: *Reinforcement Learning*. MIT Press, Cambridge, MA, 1998

[Weiser u. a. 2009] WEISER, Andreas ; BARTELS, Dr. A. ; STEINMEYER, Simon ; SCHULTZE, Dipl.-Ing K. ; MUSIAL, Dr. M. ; WEISS, Dr. K.: Intelligent Car - Teilautomatisches Fahren auf der Autobahn. In: *AAET*, 2009