Kapitel 7

Besser fahren lernen

7.1 Einleitung

In diesem Kapitel stelle ich eine weitere Anwendung des Vorhersagesystems, nämlich die Optimierung der Fahrtrichtung [Gloye, [2]2004] vor. Ich zeige, wie mit zwei unterschiedlichen Methoden – statisch und dynamisch – der Fehler zwischen gewünschter und tatsächlicher Fahrtrichtung des Roboters minimiert und so den Roboter präziser gesteuert werden kann.

Das Verhaltenssystem sendet dem Roboter den gewünschten Geschwindigkeitsvektor und die gewünschte Drehgeschwindigkeit. Auf dem Roboter ist jedoch nur ein sehr einfaches Modell für die Umsetzung der Geschwindigkeiten in Steuerwerte für die Motoren vorhanden, welches gleichzeitig noch durch das Regelungssystem auf dem Roboter beeinflusst wird (siehe Abschnitt 4.10). Dadurch kommt es zu einem Fehler zwischen gewünschter und umgesetzter Fahrtrichtung. Dieser Fehler tritt besonders stark beim Beschleunigen des Roboters auf.

Das Vorhersagesystem kann dazu benutzt werden, den Fehler zu minimieren, indem die Kommandos für den Roboter in der Form korrigiert werden, dass der Fehler minimal wird (siehe Abbildung 7.1). Eine weitere Verbesserung des Fahrverhaltens kann erreicht werden, indem eine spezielle Vorhersage für genau dieses Problem benutzt wird.

Es ist auch möglich, das Modell des Roboters und den Regelkreis auf dem Roboter selbst so zu verbessern, dass der Roboter das erwartete Verhalten umsetzt. Der hier vorgestellte Ansatz zeigt, dass mit der richtigen Nutzung vorhandener Ressourcen, wie dem Vorhersagesystem, eine Verbesserung des Roboters nicht notwendig ist. Außerdem ist der Ansatz nicht auf Roboter beschränkt, die sich durch Umprogrammierung verbessern lassen. Wenn ein Roboter als „Black-Box“ optimiert werden soll, gibt es keine andere Möglichkeit als die Befehle für dem Roboter an dessen Verhalten anzupassen.

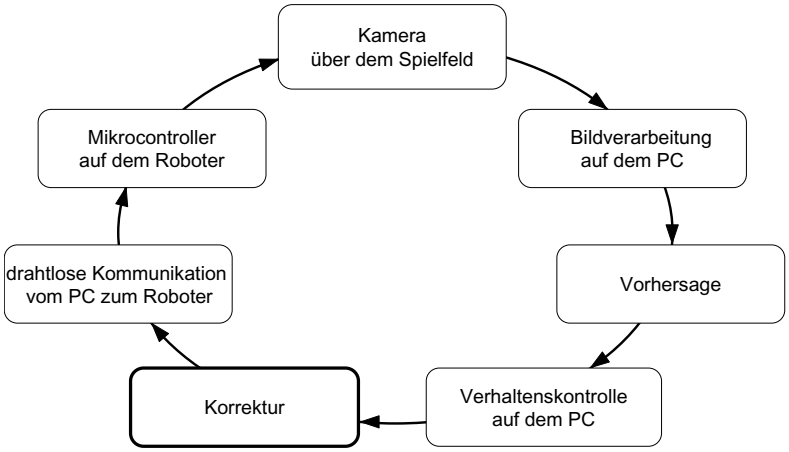

Abbildung 7.1: Systemschleife aus Abbildung 4.1 mit eingefügter Korrektur der Kommandos zur Fahrtrichtungsoptimierung.

Es soll nicht unerwähnt bleiben, dass auch bei scheinbar gleicher Hardware das Verhalten unterschiedlich sein kann. Durch das manuelle Einstellen von Parametern, um diese Unterschiede auszugleichen, kann viel Zeit verloren gehen. Der hier vorgestellte Ansatz macht die manuelle Feinjustierung überflüssig. Durch die Benutzung individueller Vorhersagen der einzelnen Roboter werden die Unterschiede minimiert.

Einen Ansatz auf einer höheren Ebene des Verhaltens zur Korrektur von Hardwareunterschieden wurde von dem Freiburger Mid-Size-Team vorgestellt [Kleiner, 2003]. Die Roboter mit unterschiedlichen Schussstärken lernten individuell den Zeitpunkt, bei dem sie aus dem Dribbeln auf das Tor schießen sollten. Dazu wurden Methoden des Verstärkungslernens benutzt.

Als erstes stelle ich zwei unterschiedliche Methoden zur Korrektur vor, die die Fahrtrichtung mit Hilfe der gelernten Vorhersage korrigieren, die in Kapitel 5 vorgestellt wurde. Beide Ansätze werden miteinander verglichen und als Zwischenergebnis präsentiert. Anschließend stelle ich ein speziell für die Korrektur der Fahrtrichtung konstruiertes Lernverfahren vor, welches das Fahrverhalten weiter verbessert. Die Zusammenfassung am Ende des Kapitels gibt einen Überblick der erzielten Ergebnisse und schlägt weitere mögliche Anwendungen der Verfahren vor.

7.2 Korrektur durch Vorhersage

Für das Vorhersagesystem sind alle Positionen, Orientierungen und Fahrwerte relativ zur aktuellen Position und Orientierung gegeben. Der aktuelle Fahrwert wird jedoch nicht für die Vorhersage verwendet. Der Grund ist, dass die Vorhersage, besser die vorhergesagte Position und Orientierung von dem Verhaltenssystem genutzt wird und das Verhaltenssystem erst anschließend die Fahrwerte berechnet. Außerdem sind die Fahrwerte für die Berechnung der Position und Orientierung im vierten Frame auch nicht notwendig, denn dies ist gerade das Delay, also die Zeit, in der die Fahrwerte noch keinen Einfluss auf das Fahrverhalten haben.

Die Fahrwerte wirken sich erst auf das fünfte Frame, das nächste Frame in dem das Verhalten operiert, aus. Um die Korrektur der Fahrwerte vornehmen zu können, ist eine Modifikation der Vorhersage notwendig, denn der aktuelle Fahrwert soll so gewählt werden, dass er dem Fahrwunsch der Richtung *nach* dem vierten Frame entspricht.

Im Folgenden werden zwei unterschiedliche Methoden vorgestellt, die den Fahrbefehl korrigieren.

7.2.1 Start-Up Korrektur

Die erste Methode folgt der vereinfachten Annahme, dass die Richtung — nicht unbedingt die Geschwindigkeit — des Roboters zwischen dem vierten und fünften Frame hauptsächlich vom aktuell gesendeten Befehl abhängig ist. Die Vergangenheit wird also vernachlässigt. Dadurch ist es möglich, eine statische Tabelle aufzustellen, die zu einem gewünschten Fahrwert die ausgeführte Fahrtrichtung zurückgibt. Die Tabelle wird mit Hilfe der Vorhersage generiert, indem die Positionen und Orientierungen der Vergangenheit auf die Nullposition, also die aktuelle (relative) Position des Roboters, gesetzt werden. Die Fahrwerte der Vergangenheit werden, bis auf die Werte des letzten Frames, ebenfalls auf 0 gesetzt. Der letzte Fahrbefehl ist eine maximale Geschwindigkeit in die Richtung, die für den aktuellen Tabelleneintrag gerade berechnet werden soll. Der Roboter steht also still und bekommt nur einen Befehl, der sich gerade im vierten Frame auf die Bewegung des Roboters auswirken wird. Die relativen vorhergesagten Positionen und Orientierungen sind dann das Ergebnis für die Tabelleneinträge.

In Abbildung 7.2 ist das Ergebnis der Berechnung der Tabellen für drei verschiedene Roboter visualisiert. Die Richtung vom Zentrum des Graphen nach außen gibt die Richtung des Fahrbefehls an. Die Linie von diesem äußeren Punkt ausgehend ist die in der Realität gefahrene Richtung. Die daran anschließende kleine Linie gibt die relative Orientierung nach dem vierten Frame an. Die Graphen geben einen Eindruck, wie stark die gewünschte Richtung — auch bei einem Roboter der nicht manipuliert wurde — von der gefahrenen Richtung abweicht. Bei einem perfekten Roboter müssten die Linien radial nach außen

verlaufen. Wie deutlich zu sehen ist, gehen die Linien bei dem Roboter mit dem um 180^o gedrehten Cover von außen nach innen. Das heißt, dass der Roboter rückwärts fährt, wenn er vorwärts fahren soll.

Abbildung 7.2: Die Korrektur der Roboterbefehle für Roboter, bei denen das Cover gegenüber der Fahrtrichtung verdreht ist. Ein normaler Roboter, ein Roboter, bei dem das Cover um 30^o gedreht ist und ein Roboter, bei dem das Cover um 180^o gedreht ist (von links nach rechts). Wie zu erkennen ist, fährt auch der Roboter mit dem richtig angebrachten Cover schräg an. Die Richtung sollte also korrigiert werden. Der Roboter mit einem 180^o gedrehten Cover würde ohne Korrektur unkontrollierbar sein, denn er würde immer das Gegenteil von dem machen, was von ihm verlangt würde.

Für die spätere Suche nach dem besten Befehl wird die Tabelle umgedreht und der Fahrbefehl genommen, bei dem der Winkel zwischen der ausgeführten Bewegung und dem Richtungswunsch am kleinsten ist.

Nach der hier vorgestellten statischen Methode kommen wir nun zur dynamischen Methode, die auch die Vergangenheit des Roboters berücksichtigt.

7.2.2 Online Korrektur

Die Vorhersage liefert uns direkt keine Information über die Auswirkungen des aktuellen Befehls, da sich der aktuelle Befehl erst zwischen dem vierten und fünften Frame in der Zukunft auswirkt. Wir können jedoch einen virtuellen Zeitschritt weitergehen. Dann wirkt sich der nun ein Schritt in der Vergangenheit liegende Befehl auf das vierte Frame aus. Die uns jetzt interessierende Differenz zwischen dem dritten und dem vierten Frame lässt sich einfach berechnen, da die Vorhersage des dritten Frames gerade die zuletzt vorhergesagte Position und Orientierung ist. Die einzige Schwierigkeit bei dem Blick in die Zukunft ist jedoch, dass die Daten der dann aktuellen Position und Orientierung noch nicht bekannt sind. Um dies zu umgehen, wird die Position und Orientierung des nächsten Frames geschätzt, indem die vorhergesagte relative Position im vierten Frame und der Winkel der vorhergesagten relativen Orientierung geviertelt werden. Damit erhalten wir eine Schätzung der Position und Orientierung im ersten Frame. Wir gehen also davon aus, dass die Bewegung zwischen dem aktuellen und dem vierten Frame nahezu linear ist.

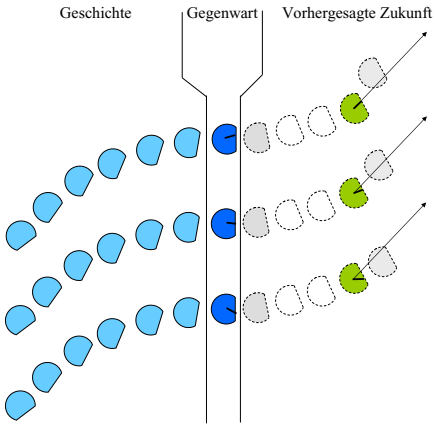

Abbildung 7.3: Die Onlinekorrekturmethode vergleicht die gewünschte Fahrtrichtung (der Pfeil) mit der Richtung der Positionsänderung zwischen dem vierten und dem fünften Frame bei unterschiedlichen Fahrbefehlen (kurzer Strich bei dem aktuellen Roboter und zur Veranschaulichung auch auf dem vorhergesagten Roboter im vierten Frame). Hier sind übereinander drei verschiedene Fahrbefehle und deren Auswirkungen visualisiert. Der Befehl wirkt sich erst im fünften Frame aus, deshalb stimmen die vorhergegangenen Positionen überein. Ohne Korrektur würde der oberste Befehl gewählt werden, denn der Fahrbefehl stimmt mit der gewünschten Richtung überein. Optimal ist jedoch der Befehl in der zweiten Zeile, denn der Pfeil schneidet den Roboter im fünften Frame genau in der Mitte.

Damit ist es möglich, auch das fünfte Frame in der Zukunft vorherzusagen und die Auswirkungen eines aktuellen Befehls zu beobachten. In Abbildung 7.3 wird das Verfahren veranschaulicht. Es wird der Fahrbefehl gewählt, bei dem der kleinste Winkel zwischen der gewünschten Fahrtrichtung und der Position beim fünften Frame, ausgehend vom vierten Frame, besteht.

7.3 Zwischenergebnis

Um die Qualität der Fahrwertoptimierung zu bestimmen, sollten unterschiedlich modifizierte Roboter in der Form eines Sterns über eine Spielfeldhälfte fahren. Das erste Experiment wurde mit einem unmodifizierten Roboter durchgeführt, also einem Roboter wie er im normalen Spiel eingesetzt wird. Wie in Abbildung 7.4 links zu sehen ist, fährt der Roboter seitlich und vorwärts recht gut auf dem Pfad. Bei einem Winkel von 45^o kommt der Roboter jedoch stark

von der Ideallinie ab. Werden die Fahrwerte nun mit der Online Korrektur verändert, so fährt der Roboter schon direkter auf die Zielpunkte zu, wie rechts in Abbildung 7.4 zu sehen ist.

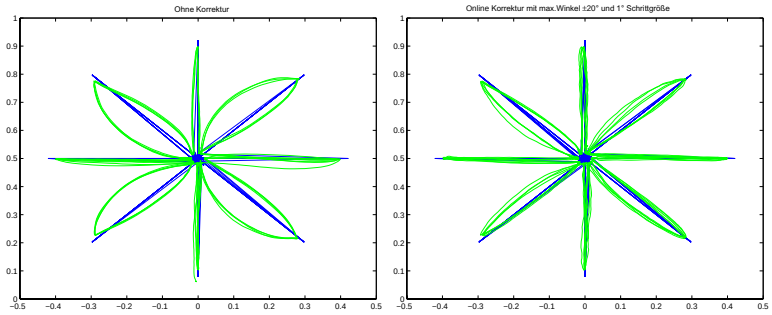

Abbildung 7.4: Die Aufgabe des Roboters ist es Zielpunkte anzufahren, die in Form eines Sterns über dem Spielfeld verteilt sind. Links wird die Aufgabe von einem normalen Roboter ohne Korrektur der Fahrwerte durchgeführt, rechts wird eine Online-Korrektur durchgeführt. Die blaue Linie zeigt den optimalen Pfad, die grüne Linie den Pfad, den der Roboter tatsächlich fährt.

Um den Korrektureffekt noch zu verdeutlichen, wurde das Cover eines Roboters um 30^o so verdreht, dass der Roboter auch bei einer perfekten Fahrweise nun mit einem Winkel von 30^o schief fahren würde. Die Ergebnisse des Experiments sind in Abbildung 7.5 zu sehen: Auf der linken Seite ist die Fahrt ohne Korrektur zu sehen. Der Roboter fährt in großen Schleifen die Zwischenpunkte des Sterns an. Der Regelkreis des Gesamtsystems ist nicht in der Lage, die Richtung des Roboters schnell zu korrigieren. Im mittleren Bild wurde eine Online Korrektur mit einem maximalen Korrekturwinkel von 40^o durchgeführt. Die Korrektur konnte dabei in $0,1^o$ Schritten korrigiert werden. Wie zu sehen ist, liegt der Pfad deutlich näher am Stern. Jedoch gibt es immer noch Winkel, die nicht korrigiert werden. Deshalb wurde der maximal zulässige Korrekturwinkel auf 180^o gesetzt. Der Pfad wird jedoch schlechter, er ist nicht mehr glatt und weicht auch stärker von der Ideallinie ab. Der Roboter „zittert“ geradezu über das Spielfeld. Die Korrekturen werden so stark durchgeführt, dass der Roboter zu oszillieren beginnt. Später in Abschnitt 7.5 werden wir sehen, wie diese Übersteuerung verhindert werden kann.

Als letzter Test wurde das Cover des Roboters um 180^o gedreht. In Abbildung 7.6 ist auf der linken Seite eine Fahrt mit Start-Up Korrektur zu sehen und auf der rechten Seite eine Fahrt mit Online Korrektur. Ohne Korrektur ist der Roboter nicht in der Lage die Zwischenziele zu erreichen, da er grundsätzlich das Gegenteil von dem machen würde, was das Ziel ist. Er würde vom Ziel wegfahren, statt in die ungefähre Richtung des Ziels. Die Start-Up Korrektur lässt den Roboter etwas schlechter fahren als einen unmodifizierten Roboter ohne Korrektur. Er bewältigt aber noch die gestellte Aufgabe. Mit der Online

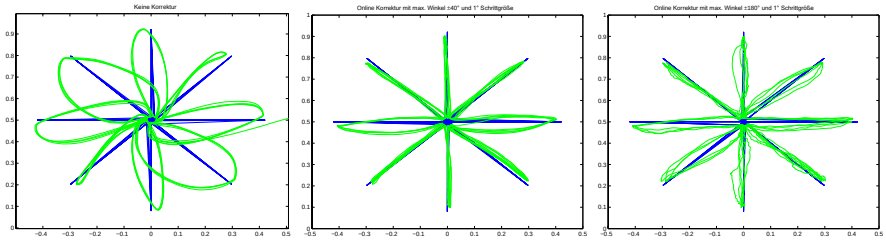

Abbildung 7.5: Die Sternfahrt eines Roboters mit einem um 30^o gedrehten Cover ohne Korrektur (links), mit Online-Korrektur bei einem maximalen Korrekturwinkel von 40^o (mittig) und mit Online-Korrektur bei einem maximalen Korrekturwinkel von 180^o (rechts). Die blaue Linie zeigt den optimalen Pfad, die grüne Linie den Pfad, den der Roboter tatsächlich fährt.

Korrektur erreicht der Roboter zwar auch die Zwischenpunkte, aber deutlich schlechter und durch die stark schwankenden Richtungen auch deutlich langsamer.

Im nächsten Abschnitt wird für die Korrektur eine angepasste Vorhersage verwendet, die in einem Schritt aus den Positionen und Orientierungen der Vergangenheit und der aktuellen Fahrtrichtung die Richtung zwischen dem vierten und fünften Frame berechnet.

7.4 Korrektur mit angepasster Vorhersage

Die Umrechnung der Positionen für verschiedene Zeitpunkte und die Anwendung der Vorhersage mit schon vorhergesagten Positionen und Orientierungen scheint nicht optimal zu sein. Zusätzlich wird angenommen, dass die Bewegung auf kleinen Strecken linear ist. Besser wäre es, eine spezielle Vorhersage zu verwenden, die das Problem direkt behandelt.

Dazu wurde eine lineare Vorhersage trainiert. Als Eingabe werden wieder die letzten sechs Positionen und Orientierungen des Roboters relativ zur aktuellen Position und Orientierung sowie die letzten sechs Fahrbefehle und der aktuelle Fahrbefehl benutzt. Als Ausgabe soll die Richtung und die Distanz zwischen dem vierten und dem fünften Frame in der Zukunft gelernt werden. Durch Variation der aktuellen Fahrwerte kann dann die gewünschte Richtung zwischen dem vierten und dem fünften Frame in der Zukunft weitestgehend bestimmt werden.

Im nächsten Abschnitt wird eine andere Bewertung für die beste Richtungswahl vorgestellt. Die neue Bewertungsfunktion hat gerade bei starken Richtungsänderungen, einen großen Vorteil: Der Roboter oszilliert nicht mehr so stark.

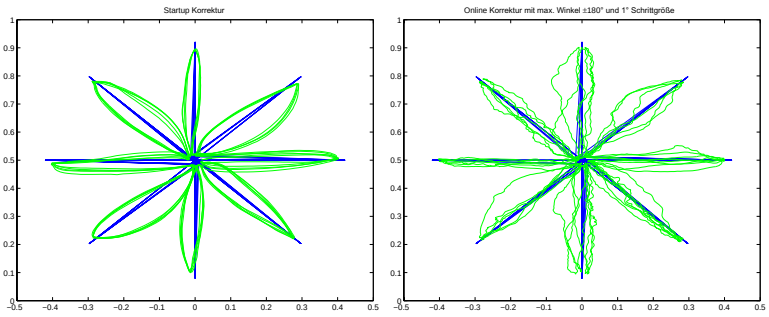

Abbildung 7.6: Die Sternfahrt eines Roboters mit einem um 180^o gedrehten Cover. Ein Fahren ohne Korrektur ist nicht möglich. Links ist der Pfad mit der Start-Up Korrektur zu sehen, rechts mit der Online-Korrektur bei einer maximalen Winkelabweichung von 180^o und einer Schrittweite von 1^o. Die blaue Linie zeigt den optimalen Pfad, die grüne Linie den Pfad, den der Roboter tatsächlich fährt.

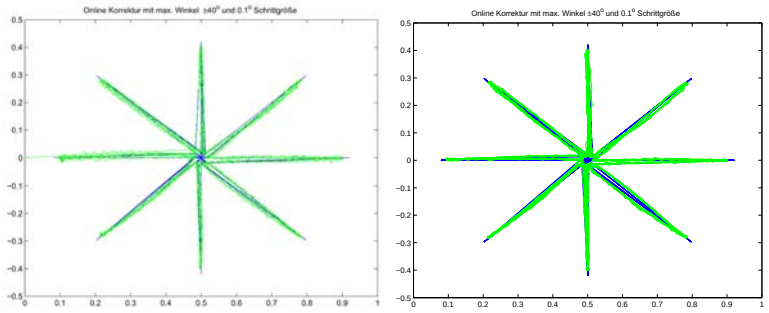

Abbildung 7.7: Die Sternfahrt eines unmodifizierten Roboters mit einer speziellen Vorhersage, welche die Richtung zwischen dem vierten und dem fünften Frame in der Zukunft vorhersagt (links) und zusätzlich zur Auswahl des besten Fahrwerts nicht den Winkel, sondern die Beschleunigung berücksichtigt (rechts). Die blaue Linie zeigt den optimalen Pfad, die grüne Linie den Pfad, den der Roboter tatsächlich fährt.

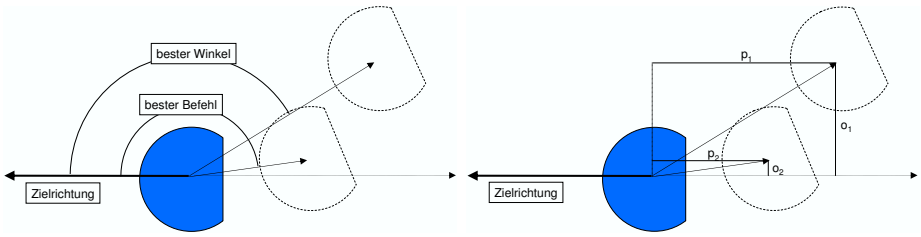

Abbildung 7.8: Beispiel einer starken Richtungsänderung. In der linken Grafik geschieht die Auswahl über die kleinste Winkeldifferenz gegenüber der vorhergesagten und der gewünschten Richtung. Der kleinste Winkel ist nicht die beste Wahl. Der größere Winkel bremst den Roboter stärker in die Richtung ab, in die er fahren soll. Die Richtung mit dem größeren Winkel hat also die bessere Beschleunigung in die gewünschte Richtung. Dies wird bei dem zweiten Verfahren in der rechten Zeichnung berücksichtigt. Die Richtung wird in zwei Komponenten p in o bezüglich des gewünschten Fahrwerts zerlegt. Die bessere Kombination wird, wie im Text erklärt, gewählt.

7.5 Eine bessere Richtung

Bisher wurde die Richtung als optimal bewertet, deren Winkel mit dem Fahrwunsch minimal ist. Bei großen Änderungswünschen gegenüber der aktuellen Fahrtrichtung ist dies jedoch nicht optimal, wie Abbildung 7.8 links verdeutlicht. Der bessere Winkel führt in die falsche Richtung, wenn alle möglichen Winkel recht groß sind. Sind die Winkel recht klein, zum Beispiel unter 40^o, dann kommt es nicht zu diesem Effekt.

Ein Korrekturverfahren, das auch gegenüber großen Richtungsänderungen robust ist, wird in Abbildung 7.8 rechts gezeigt. Die Auswirkung des Fahrbefehls wird hinsichtlich der gewünschten Fahrtrichtung in zwei Komponenten zerlegt. Eine der Wunschfahrtrichtung parallele Richtung p und eine der Wunschrichtung orthogonale Richtung o. Die „beste“ Richtung ergibt sich dann aus dem Maximum einer gewichteten Summe

$$\max_{v_i \in V}(p_i - \lambda|o_i|) \tag{7.1}$$

der beiden Komponenten, wobei V die Menge aller vorhergesagten Richtungen v_i ist, die durch alle zur Vorhersage verwendeten Fahrwerte erzeugt wurden. Das Gewicht λ soll geeignet gewählt werden.

Die hier gezeigten Tests wurden mit $\lambda = 2$ durchgeführt. Tests mit $\lambda = 3$, $\lambda = 4$ und $\lambda = 5$ ergeben vergleichbare Ergebnisse. Für die Auswahl der Fahrwerte wurden jeweils alle Fahrwerte zwischen -40^o und 40^o in einem Abstand von $0,1^o$ gegenüber dem gewünschten Fahrwert getestet. In Abbildung 7.7 ist der

Testlauf eines unmodifizierten Roboters zu sehen. Der Pfad ist glatt, ohne starke Oszillation und trotzdem sehr nah an der Ideallinie.

Der Aufbau einer speziellen Vorhersage, die als Eingabeparameter die Beschleunigung des Roboters berücksichtigt und als Ausgabe direkt den besten Fahrbefehl liefert, würde das Verfahren eleganter aussehen lassen und den Algorithmus zur Befehlsauswahl vereinfachen. Im nächsten Abschnitt wird diese Vorhersage vorgestellt.

7.6 Zusammenfassung

In diesem Kapitel habe ich mich mit dem Lernen des optimalen Fahrbefehls beschäftigt. Ich habe drei Verfahren vorgestellt, die den Fehler zwischen der gewünschten und tatsächlich ausgeführten Richtung minimieren und habe die Ergebnisse miteinander verglichen. Es wurde gezeigt, dass sich die Fahrweise eines Roboters durch „vorausschauende“ Fahrweise deutlich verbessern lässt. Gerade wenn der Roboter eine „Black-Box“ ist und es keine Möglichkeit gibt, die inneren Fahreigenschaften des Roboters zu optimieren, bieten die hier vorgestellten Methoden die Chance, das Verhalten der Roboter maßgeblich zu verbessern.

Der zweite Vorteil der Fahrtoptimierung ist es, durch die hier vorgestellten automatischen Lernmethoden das Verhalten gleicher Robotertypen, die sich auf Grund von Fertigungstoleranzen oder unterschiedlicher Verschleißerscheinungen unterscheiden, anzugleichen. Dadurch ist das zeitaufwändige individuelle Feinjustierung der Roboterparameter nicht mehr notwendig. Dabei ist es gleichzeitig möglich, die Roboter auf höherer Ebene des Teamverhaltens austauschen zu können, ohne die Aufgaben und die notwendigen Befehle für einzelne Roboter anpassen zu müssen.

Eine leichte Verbesserung des Verfahrens wäre es, die Rotation des Roboters, die durch seine Beschleunigung entsteht, auszugleichen. Die beabsichtigte Rotation des Roboters, die durch den Drehbefehl bestimmt wird, ist durch dieses Verfahren leider nicht zu verbessern. Dies liegt an der Art der Befehle. Der Fahrbefehl gibt die Geschwindigkeit und die Richtung an; es wird jedoch nur die Richtung korrigiert. Für die Geschwindigkeit gibt es keinen Vergleichswert mit einer optimalen Geschwindigkeit. Genauso ist es bei der Rotation. Auch hier wird nur ein Wert zwischen -1 und 1 vorgegeben. Der Regelkreis auf dem Roboter versucht, einen Zwischenwert linear bezüglich der maximalen Rotationsgeschwindigkeit nach rechts oder links umzusetzen. Das Ziel des Verhaltens ist auf dieser Ebene, in der die Korrektur eingreift, nicht sichtbar. Um das Verhalten auf dieser Ebene optimieren zu können wären Befehle der Form „fahre 30 cm im Winkel von 45^o und drehe dich um -15^o“ nötig. Diese Umsetzung der Befehle wird jedoch im Verhalten eine Ebene höher durchgeführt. Kapitel 9 zeigt die Optimierung dieser Befehle.

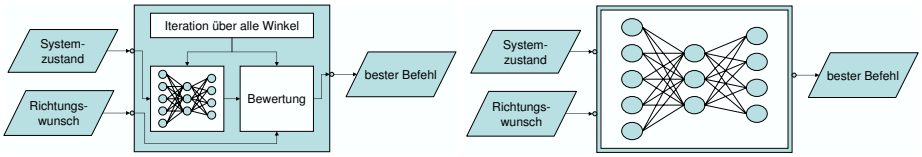

Abbildung 7.9: Links ist das bisherige Verfahren zur Optimierung der Fahrbefehle veranschaulicht. Die Berechnungsfunktion, die in einer Schleife mehrer Fahrbefehle bewertet, könnte durch ein neuronales Netz, wie rechts gezeigt, oder durch einen anderen Funktionsapproximator gelernt werden.

Der optimale Fahrbefehl wurde entweder durch eine Look-Up-Tabelle ausgewählt oder durch mehrfaches, virtuelles Testen mit unterschiedlichen Fahrbefehlen. Gerade die zweite Methode ließe sich optimieren, indem in einem zweiten Schritt des Trainings eine Funktion — zum Beispiel ein neuronales Netz — gelernt werden könnte, die den besten Fahrbefehl liefert (siehe Abbildung 7.9).

Eine andere Anwendungsmöglichkeit ist es, die Korrektur der Fahrwerte eines Roboters während eines längeren Spiels anzupassen, falls sich das Verhalten des Roboters, durch Hardwaredefekte oder leere Batterien, leicht ändert und diese durch den PID Controller nicht mehr ausgeglichen werden können. Bei stärkeren Defekten, bei denen das Verhalten nicht mehr durch Änderungen der Fahrwerte korrigiert werden kann, können andere Verfahren, wie in Kapitel 8 erläutert, angewendet werden.