Kapitel 5

Lernen der Roboterdynamik

5.1 Einleitung

In diesem Kapitel untersuche ich, wie die Fahrdynamik eines Roboters gelernt werden kann. Dies ist wichtig, denn in der RoboCup Small-Size-Liga erreichen die Roboter eine Geschwindigkeit von bis zu 3m/s. Das Regelungssystem zur Steuerung der Roboter hat jedoch eine Verzögerung von 130ms. Bis ein Roboter auf eine Situation reagieren kann, weicht seine Position bis zu 40cm von der erwarteten Position ab. Eine vorausschauende Fahrweise ist somit wichtig. Diese wird durch eine Vorhersage um die Länge der Verzögerung erreicht.

Ein einfaches Beispiel zeigt, was passiert, wenn keine Vorhersage benutzt wird. Die in Abbildung 5.1 zu sehende Aufgabe ist es, von einem Punkt in einer Ecke des Spielfelds möglichst schnell auf einen Punkt in der anderen Ecke des Spielfelds zu fahren und dort stehen zu bleiben. Die schwarze Linie ist der gefahrene Pfad mit Vorhersage, die graue Linie der gefahrene Pfad ohne Vorhersage. Die Geschwindigkeit des Roboters während der Fahrt ist in Abbildung 5.2 gezeigt. Es ist deutlich zu sehen, dass der Roboter ohne Vorhersage erst über das Ziel hinaus fährt und dann langsam auf die Zielposition zurückfährt. Das Hinausschießen könnte dadurch verhindert werden, dass der Roboter früher anfängt zu bremsen, zum Beispiel wenn die von dem System gesehende Position 1m von der Zielposition entfernt ist. Dies würde aber das ganze System erheblich verlangsamen, was unerwünscht ist.

Die Vorhersage geschieht entweder mit Hilfe eines neuronalen Netzes [Behnke, 2003] oder eines linearen Modells [Gloye, [2]2003]. Die Eingaben und Ausgaben sind für beide Modelle gleich: Die Eingabe besteht aus den Zu-

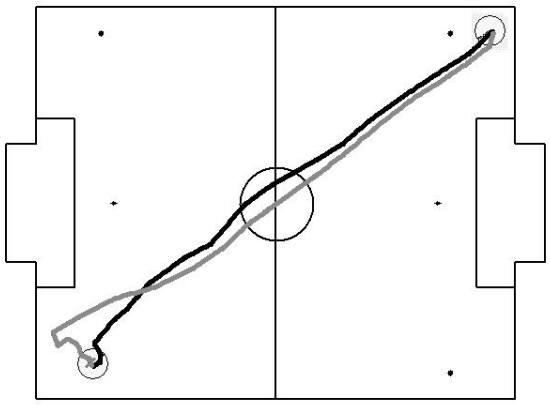

Abbildung 5.1: Ein gefahrener Pfad des Roboters von oben rechts nach unten links mit Vorhersage (schwarze Linie) und ohne Vorhersage (graue Linie).

standsparametern[1] (Positions-, Orientierungs- und Sollgeschwindigkeitsvektor) des Roboters bezüglich der aktuellen Position über die sechs letzten Zeitschritte. Die Ausgabe besteht aus den erwarteten relativen Position und Orientierung nach 130ms bezüglich der aktuellen Position und Orientierung. Die Vorhersage hat einen durchschnittlichen quadratischen Fehler von 1,32cm pro Dimension für die Position und einen durchschnittlichen quadratischen Fehler von $4,5^o$ für die Orientierung.

In diesem Kapitel wird die Verzögerung des Systems analysiert und mit verschiedenen Methoden automatisch und manuell gemessen. Als nächstes werden die implementierten Lernmethoden für die Vorhersage vorgestellt und ihre Ergebnisse miteinander verglichen. Die Vorhersagegenauigkeit und ihre theoretischen und praktischen Grenzen werden analysiert und eventuelle Verbesserungen werden diskutiert. Anschließend werden die Lernverfahren zusammengefasst und ihre Anwendungen bei den FU-Fighters und bei anderen Problemen beschrieben.

5.2 Überall Verzögerung

Bei fast allen natürlichen und technischen Systemen ist die Ursache mit der Wirkung nicht unmittelbar zeitlich verknüpft, es kommt also zu einer Verzögerung, die auch als Totzeit bezeichnet wird. Das erste Konzept der Vorhersage kommt aus dem Bereich der Biologie, als Helmholtz versucht hat, die visuel-

[1] Bei der Vorhersage fremder Roboter entfällt bei der Eingabe die Sollgeschwindigkeit.

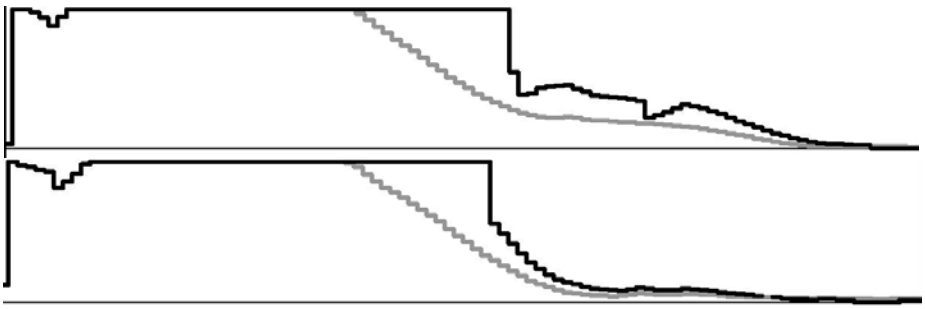

Abbildung 5.2: Die Robotergeschwindigkeit (schwarze Kurve) im Vergleich zum Abstand des Roboters von der Zielposition (graue Kurve). Die obere Grafik zeigt die Situation ohne Vorhersage. Es ist deutlich zu sehen, dass das Ziel langsamer erreicht wird und der Roboter die Geschwindigkeit mehrmals ändern muss. Die untere Grafik zeigt die Situation mit Vorhersage. Hier wird die Zielposition schnell erreicht und die Geschwindigkeit nur leicht korrigiert.

le Lokalisierung beim Menschen zu verstehen [Wolpert, 2001]. Er schlug vor, dass das Gehirn die Blickrichtung anhand der Ausgaben der Motorneuronen an die Augenmuskeln vorhersagt, bevor die Position selbst sensorisch erfasst ist. Es ist noch heute ein aktuelles Forschungsgebiet, wie die Verzögerung zwischen sensorischen Informationen, deren Übertragung zum Gehirn und weiter zu den Aktuatoren in biologischen Systemen kompensiert wird [Kataria, 2002].

Doch auch in technischen Bereichen spielt das Delay eine wichtige Rolle. So zum Beispiel in der Telerobotik. Dort agiert der Operator weit entfernt vom Roboter und erwartet über Kraftrückkopplung eine augenblickliche Reaktion seines Handelns [Hannaford, 1999]. Beispiele der Telerobotik sind durch das Internet gesteuerten Roboter[2], Marsroboter der NASA[3] und Teleoperationen in der Medizin.

Doch auch bei schnellen reaktiven Systemen spielt das Wissen über ein vorhandenes Delay eine entscheidende Rolle für ein stabil zu steuerndes System. So zum Beispiel bei Hubschraubern [Ng, [2]2004, Lozano, 2004] und bei schnellen Roboterarmen [Kataria, 2002]. Selbst bei normalen Steuerungsproblemen, bei denen PID-Controller eingesetzt werden, liegen die Ursachen für die Einstellung der optimalen Parameter in der Trägheit des zu steuernden Systems, was auch als Delay bezeichnet werden kann (siehe Kapitel 9).

In Abschnitt 5.3 wird erläutert, aus welchen Komponenten sich das Delay im FU Fighters System zusammensetzt. Für ein Vorhersagesystem ist es jedoch unerheblich, an welcher Stelle das Delay entsteht. Es kann an einer Stelle zu-

[2]Nur ein Beispiel von vielen ist der Museumsführer im Deutschen Museum Bonn www.cs.uni-bonn.de/~rhino/tourguide

[3]NASA telerobocs program ranier.oact.hq.nasa.gov/telerobotics_page/telerobotics.shtm

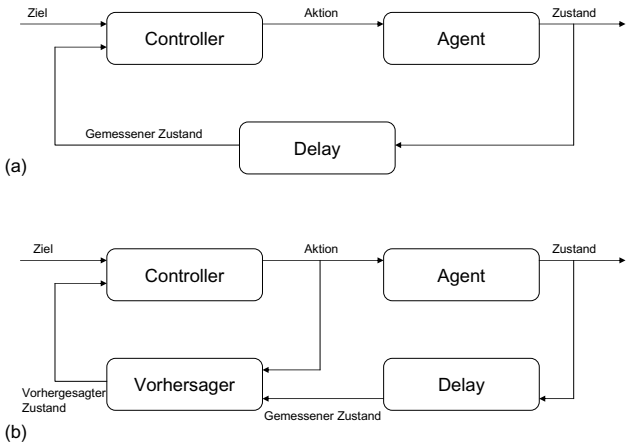

Abbildung 5.3: Regelung mit Delay: (a) Die Totzeit kann als nachgeschaltetes Glied des Agenten aufgefasst werden. (b) Die Vorhersage versucht, den Agenten ohne Verzögerung zu simulieren. Zur Optimierung und Fehlerkorrektur wird der Agent beobachtet. Dadurch kann die Regelung einfach gehalten werden, denn Aktionen wirken sich unmittelbar aus.

sammengefasst werden (siehe Abbildung 5.3(a)). In unserem Fall ist der Regler das Verhalten und die Maschine das System von der Funkübertragung bis zur Bildverarbeitung (siehe Abbildung 5.4).

5.3 Quellen des Delays

Abbildung 5.4 zeigt noch einmal die Kontrollschleife des Systems, deren Komponenten unterschiedlich viel zum Delay beitragen. Im Unterschied zu Abbildung 4.1 auf Seite 46 ist das System jetzt nach Regelung (Kontrolle) und zu regelndem System (Agent) aufgeteilt. Wir gehen auf die einzelnen Komponenten genauer ein:

- Aufnahme des Bildes durch die Kamera. Die Belichtungszeit spielt hier eine entscheidende Rolle. Wenn die Belichtungszeit zu kurz eingestellt ist, dann müssen die Signale zu sehr verstärkt werden, sodass es zu starkem Rauschen kommt, was die Bildauswertung erheblich erschweren würde. Bei einer zu langen Belichtungszeit hat die Kamera nicht genug Zeit zum

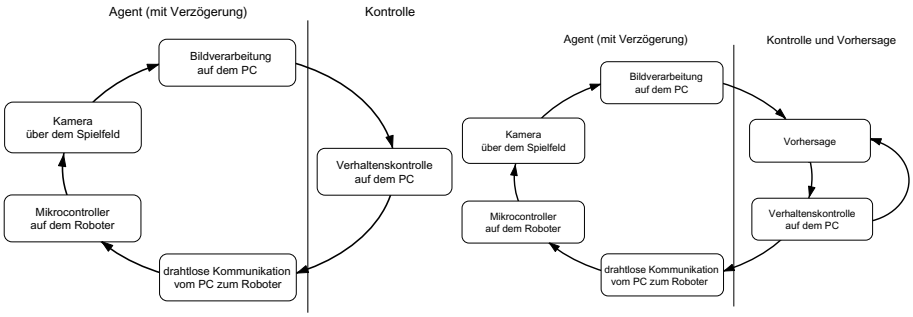

Abbildung 5.4: Die Systemschleife des FU Fighters Systems mit (rechts) und ohne (links) Vorhersagesystem. Die Systemschleife in der Abbildung 4.1 auf Seite 46 wurde hier in Maschine und Controller aufgeteilt, wie dies in der Regelungstechnik üblich ist (siehe Abbildung 5.3 auf Seite 80).

Auslesen der Bilder und die Bildwiederholrate der Kamera würde sinken. Wir verwenden eine Belichtungszeit von etwa 10ms.

- Übertragen des Bildes von der Kamera zum Rechner. Die Bildübertragung zum Rechner erfolgt über eine IEEE 1394 (FireWire) Schnittstelle mit maximal 400Mbps. Die Bilder haben eine Auflösung von 780x582 Punkten. Da die Daten ungefiltert übertragen werden, also nur 8 bit/Pixel, sind für die Übertragung eines Bildes etwa 4Mbit notwendig, wofür 10ms gebraucht werden.
- Bildauswertung und Synchronisation der Kameras. Die Bildauswertung benötigt, wenn keine globale Suche stattfindet, etwa 1ms. Um die beiden Kamerabilder zu synchronisieren, sind im Durchschnitt 3ms notwendig.
- Verhaltenssystem. Benötigt, wenn keine Pfadplanung aktiv ist, auch etwa 1ms. Mit Pfadplanung bis zu 10ms.
- Funkübertragung vom Zentralrechner zu den Robotern. Die Sendemodule sind über eine serielle Schnittstelle angeschlossen. Die Daten brauchen durch den FIFO circa 7ms. Alle Roboter werden nacheinander mit Daten versorgt, eine Schleife über alle Roboter dauert etwa 10ms.
- Dekodierung der Pakete und Umsetzung in Signale für die Motoren auf dem Roboter. Das Programm auf dem Roboter wird alle 8ms ausgeführt.
- Reaktion des physischen Teils des Roboters. Der Roboter und die Motoren haben eine gewisse Trägheit. Eine genaue Messung ist nicht möglich.

Alle bekannten Verzögerungen zusammen ergeben ohne die Roboterhardware 43ms bis 53ms. Im nächsten Abschnitt wird die Methode vorgestellt, um die Gesamtverzögerung, mit der Reaktionszeit des Roboters, zu schätzen.

5.4 Delaymessung

Um das Gesamtdelay zu messen können unterschiedliche Methoden benutzt werden. In dem ersten hier vorgestellten Verfahren werden dem zu testenden Roboter Sollgeschwindigkeiten in Form einer langsamen Sinusschwingung in eine Richtung gesendet. Der Roboter reagiert auf diese Kommandos und fährt dadurch in eine Richtung, beschleunigt langsam und bremst langsam wieder ab, bis sich die Richtung umkehrt. Die Vision nimmt dabei die Positionen der Roboters auf dem Spielfeld auf. Die Richtungsänderungen sind über die Zeit betrachtet sehr deutlich zu erkennen (siehe Abbildung 5.5). Die Differenz zwischen dem Nulldurchgang der Richtungsgeschwindigkeit (Übergang von Vorwärts- in Rückwärts-Fahrt und umgekehrt) und dem lokalen Minimum bzw. Maximum der Position (gemessener Übergang von Vorwärts- in Rückwärts-Fahrt) ist das Delay.

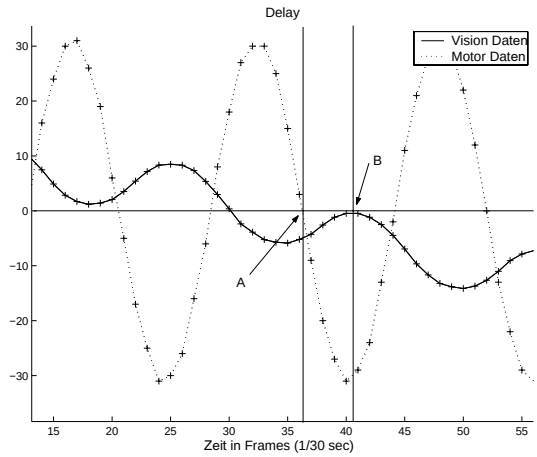

Abbildung 5.5: Zur Delaymessung wird die gesendete Richtungsgeschwindigkeit (gepunktete Linie) mit der gemessenen Position (durchgezogene Linie) in einer Dimension über die Zeit verglichen. Die Differenz zwischen der Richtungsänderung der Sollgeschwindigkeit (Markierung A) und der erkannten Richtungsänderung (Markierung B) ist dann das Delay. Hier liegt das gemessene Delay bei vier Frames, also 132ms.

In Abbildung 5.6 wurde der Roboter angewiesen, mit der Geschwindigkeit einer Sinusfunktion zu drehen. In der linken Grafik wurden die beiden Variablen Drehbefehl und Drehgeschwindigkeit gegeneinander aufgetragen. Rechts wurde ein Versatz von neun Frames berücksichtigt. Bei dieser Verschiebung weisen die Daten die höchste Kovarianz auf. Eine lineare Abhängigkeit ist zu erkennen.

Die gemessene Totzeit bezieht sich auf genau eine Fahrweise, in diesem Fall das langsame Sinusfahren bzw. Sinusdrehen. Oftmals soll der Roboter jedoch

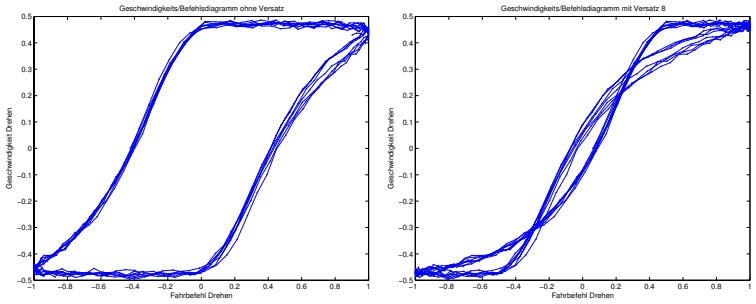

Abbildung 5.6: Der Zusammenhang zwischen Drehbefehl und Drehgeschwindigkeit. Einmal ohne Berücksichtigung des Delays (links) und einmal mit Berücksichtigung des Delays (rechts). Es fällt auf, dass die maximale Drehgeschwindigkeit schon bei dem Steuerwert 0,4 erreicht wird.

in einer Weise fahren, die nicht glatt ist, sondern ruckartige Richtungswechsel enthält. Der manuelle Vergleich der Positionswerte mit den Sollgeschwindigkeiten ist dann aber schwieriger.

Eine andere Möglichkeit zur Bestimmung des Delays ist es, die statistische Abhängigkeit der Fahrwerte in der Vergangenheit zur momentanen Fahrtrichtung zu bestimmen. Wenn die Abhängigkeit in einem bestimmten Bereich signifikant größer ist, kann die zeitliche Differenz als Delay betrachten werden.

Um die Abweichung des Delay bei unterschiedlichen Aufgaben zu zeigen, wurden mehrere Steuersequenzen erzeugt. Erstens wurde der Roboter mit dem Joystick gesteuert. Dabei wurde in verschiedene Richtungen gefahren und während der Fahrt gedreht. Zweitens wurden dem Roboter Fahrbefehle für die Vorwärtsrichtung in Form einer Sinuskurve mit unterschiedlichen Frequenzen geschickt. Drittens bekam der Roboter abwechselnd die Fahrbefehle mit maximaler Geschwindigkeit vorwärts und mit maximaler Geschwindigkeit rückwärts zu fahren. Viertens sollte der Roboter abwechselnd nach links und rechts drehen, wobei die Geschwindigkeit durch eine Sinuskurve vorgegeben wurde. Für alle Aufgaben wurde die Verschiebung der Daten berechnet, bei der die maximale Kovarianz erreicht wird. In Abbildung 5.7 sind die Kovarianzen der Aufgaben für unterschiedliche Translationen gezeigt. Das kleinste Delay tritt mit 8 Frames (154ms) beim Drehen des Roboters auf. Das größte Delay mit 13 Frames (250ms) bei langsamer Vorwärts-Beschleunigung des Roboters.

Das BigRed-Team der Cornell University misst das Gesamtdelay des Systems über die Drehung des Roboters [D'Andrea, 2001]. Dabei wird Frequenz der stabilen Eigenschwingung des Systems gemessen, aus dem dann das Delay hergeleitet werden kann. Für ihr eigenes System haben Sie ein Delay von 120ms gemessen.

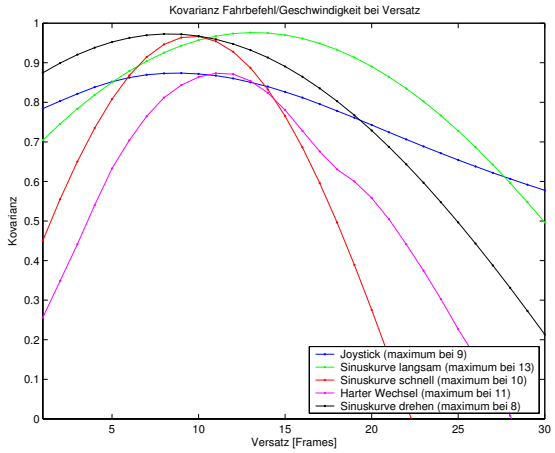

Abbildung 5.7: Kovarianzmessungen bei unterschiedlichen Aufgaben des Roboters. Das Delay schwankt zwischen 8 und 13 Frames. Der verwendete Roboter ist ein anderer als in Abbildung 5.5. Die Bildwiederholrate betrug bei den Versuchen 52Hz.

5.5 Datenaufbereitung

Als Eingabe für die Vorhersage werden die Positionen und Orientierungen der Vision der letzten sechs Frames relativ zur aktuellen Position und Orientierung der Vision sowie die letzten sechs Fahrwerte des Roboters genommen. Die Fahrwerte sind relativ zum Roboter, sie müssen also nicht umgerechnet werden. Statt sechs Frames der Vergangenheit könnten auch mehr oder weniger Frames benutzen werden (siehe Abbildung 5.8). Bei früheren Experimenten hat sich aber gezeigt, dass eine höhere Anzahl von Frames die Qualität der Vorhersage nicht verbessert und eine geringere Anzahl von Frames die Vorhersage verschlechtert.

Die relativen Positionen des Roboters werden in (x, y)-Koordinaten kodiert. Die Orientierungen werden nicht direkt als Winkel ω der Abweichung bezüglich der aktuellen Orientierung genommen, denn dies würde zu einer Unstetigkeit der Eingabe führen. Wenn man den Winkel aus dem Bereich $[-180^o, 180^o]$ wählt, würde beim Übergang von 180^o auf -180^o eine Unstetigkeit in der Eingabe entstehen, die eigentlich nicht vorhanden ist. Die naheliegende Lösung ist es, eine $(\sin\omega, \cos\omega)$-Darstellung, also den Sinus und Kosinus des Winkels als Eingabe zu nehmen.

Als Fahrwerte werden die zwei Komponenten des Geschwindigkeitsrichtungsvektors (δ_x, δ_y) und die Drehgeschwindigkeit δ_ϕ genommen. Pro Frame gibt es also zwei Komponenten für die Position, zwei Komponenten für die Orientierung und drei Komponenten für die Fahrwerte. Die Eingabe hat damit 42

Komponenten, da sechs Frames benutzt werden.

Als Zielwerte sollen die Position und die Orientierung des vierten Frames in der Zukunft berechnet werden, was genau dem gemessenen Delay von 130ms entspricht. Die Orientierung wird bei der Ausgabe genauso kodiert wie bei der Eingabe. Er werden also vier Werte vorhergesagt, zwei für die Position und zwei für die Orientierung.

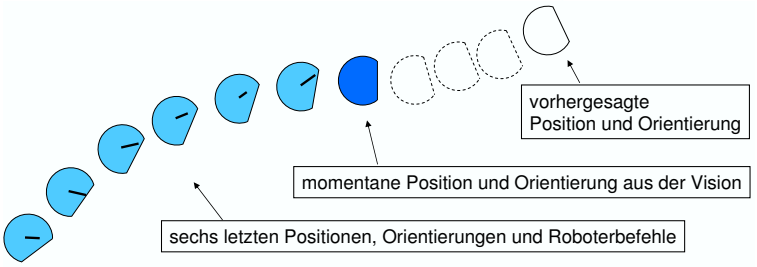

Abbildung 5.8: Die Eingabe- und Ausgabe-Daten zur Vorhersage. Die sechs letzten Positionen und Orientierungen sowie die Fahrwerte der Roboter bezüglich der aktuellen Position und Orientierung des Roboters dienen als Eingabe. Die Aufgabe ist es, die vierte zukünftige Position und Orientierung vorherzusagen. In der Abbildung geben die abgeflachten Seiten die Orientierung des Roboters wieder, die kurzen Linien innerhalb des Roboters visualisieren die Fahrwerte.

5.6 Trainings- und Test-Daten

Für die Trainingsdaten fährt der zu trainierende Roboter verschiedene Pfade ab, um möglichst alle Situationen eines Spiels lernen zu können. Dabei werden die Pfade entweder durch spezielle Verhalten generiert, zum Beispiel soll der Roboter in Form einer Acht über das Spielfeld fahren (siehe Abbildung 5.9), oder der Benutzer gibt manuell generierte Pfade vor. Dies kann dadurch geschehen, dass dem Roboter mit einem Joystick die Richtung vorgegeben wird, in die der Roboter fahren soll, oder indem mit Hilfe der Maus nacheinander Positionen gewählt werden, zu denen der Roboter fahren soll. Die Daten der absoluten Positionen, der absoluten Orientierungen und der relative Fahrbefehle des Roboters werden aufgezeichnet. Später werden die Daten dann in die Trainingsmenge und Testmenge aufgeteilt, indem die Daten jedes Frames abwechselnd der Testmenge und der Trainingsmenge zugeordnet werden.

Ideal wäre es, wenn die Trainingsdaten aus einem regulären Spiel stammen würden. Dabei ergibt sich aber eine Schwierigkeit: Die Roboter werden oft von anderen Robotern blockiert. Diese Situationen müssten herausgefiltert werden, denn die Eingabedaten berücksichtigen diese Situation nicht und ein Lernal-

gorithmus müsste bei gleichen Eingabedaten unterschiedliche Ausgaben lernen. Bisher ist dieser Ansatz noch nicht implementiert worden.

5.7 Vorhersagemethoden

Es gibt viele Möglichkeiten, die Auswirkungen eines Delays zu minimieren. Eine einfache Methode ist es, das System langsamer zu regeln. In unserem Fall hieße das, die Roboter langsam fahren zu lassen. Eine offensichtlich schlechte Idee. Eine weitere Möglichkeit ist es, das Delay nicht zu betrachten. Dann würden die Roboter jedoch sehr ungenau fahren, über die Ziele hinausfahren, den Ball nicht führen oder treffen können, andere Roboter rammen, so dass ein vernünftiges Spiel nicht mehr möglich wäre.

Es gibt viele Ansätze, die Auswirkungen des Delays zu handhaben. Ein klassischer Ansatz ist der Smith-Predictor [Smith, 1959]. Er benutzt ein Modell des zu kontrollierenden Systems ohne Delay. Die virtuellen Sensoren (die Ausgabe des Modells) dienen dann als Eingabe für die Regelung. Da das Modell nicht exakt ist, verwendet das Verfahren auch den Zustand des realen Systems, um die Abweichungen zu korrigieren (siehe Abbildung 5.3(b)). Wenn das Modell schlecht gewählt ist oder das Delay nicht mit dem im Modell angenommenen Delay übereinstimmt, kann dies zu Oszillationen führen. Einige Forscher nehmen an, dass das Kleinhirn als ein Smith-Predictor arbeitet [Miall, 1993].

Ein einfacher Ansatz für eine Vorhersage ist ein Kalman-Filter. Für lineare Probleme ist er ideal geeignet, zum Beispiel für einen frei rollenden Ball [Veloso, 1999, Petrov, 2004]. Wenn sich das System jedoch nicht linear verhält, dann ist ein normaler Kalman-Filter sehr ungenau. Für die Ballvorhersage wird oft auch ein erweiterter Kalman-Filter (Extended Kalman Filter (EKF), Extended Kalman Bucy Filter (EKBF), Improbability Filtering (ImpF)) benutzt, der nichtlineare Eigenschaften, wie zum Beispiel die Ballreibung berücksichtigt [Han, 1997] oder falsche Messungen herausfiltert [Browning, 2002]. Auch dies hilft in vielen Situationen nicht weiter. Ein einfaches Beispiel ist der Ball, der an einem Hindernis abprallt oder der geschossen wird. Ein linearer Filter erzeugt dann selbst ein großes Delay.

Im Small-Size-System der FU-Fighters wird daher ein linearer Kalman-Filter für die Ballvorhersage verwendet, solange sich die Geschwindigkeit des Balls nicht zu stark ändert. Wenn sich die Geschwindigkeit stark ändert, zum Beispiel weil der Ball geschossen wird, werden andere Filter verwendet.

Die Idee, mehrere Filter für die Ballvorhersage zu verwenden wird auch in der AIBO Liga eingesetzt [Kwok, 2004]. Dabei wird der Ball im Zusammenhang mit der Umgebung betrachtet und Wahrscheinlichkeitsmodelle angewendet, wie wahrscheinlich welche Interaktion mit der Umgebung ist. Wenn zum Beispiel ein AIBO in der Nähe des Balls ist, wie hoch ist die Wahrscheinlichkeit, dass der Roboter den Ball in eine bestimmte Richtung schießt. Da die Wahrscheinlich-

keitsverteilungen kontinuierlich sind, werden zur Diskretisierung Partikelfilter eingesetzt. Manche Teams verwenden für die Vorhersage des nichtlinearen Verhaltens von Robotern erweiterte Kalman-Bucy Filter [Browning, 2002]. Dieser Ansatz benötigt aber auch ein sehr gutes Modell des Roboters, um gute Vorhersagen zu machen.

Im nächsten Abschnitt wird vorgestellt, wie auch ohne ein explizites Modell eine gute Vorhersage möglich ist. Das Modell wird gelernt. Die Filterung der Daten fällt dabei gleich mit ab.

5.8 Untersuchte Vorhersageverfahren

Im FU-Fighters-Framework stehen drei Vorhersagemodelle zur Verfügung. Das älteste Modell ist ein dreischichtiges neuronales Netz ohne Rekurrenz, welches mit dem Backpropagation-Algorithmus trainiert wird. Das einfachste Modell, welches für unbekannte gegnerische Roboter benutzt wird ist ein einfaches Geschwindigkeitsmodell. Ein weiteres Modell ist eine Vorhersage mit Hilfe einer linearen Regression. Alle drei Modelle haben die Eigenschaft, dass kein explizites Modell des Roboters vorhanden sein muss. Wenn sich also die Eigenschaften eines Roboters ändern, zum Beispiel durch die Verwendung anderer Motoren oder durch Softwareänderung auf dem Roboter, dann müssen nur neue Daten aufgezeichnet werden und die Vorhersager neu trainiert werden. Es ist keine weitere Programmierung notwendig. Im folgenden soll auf die Vorhersager genauer eingegangen werden.

Für alle vorgestellten Vorhersager sollen die Daten in folgender Form zur Verfügung stehen:

$$\begin{aligned} &(x_1, y_1, \omega_1^x, \omega_1^y, \delta_1^x, \delta_1^y, \delta_1^\phi) \\ &(x_2, y_2, \omega_2^x, \omega_2^y, \delta_2^x, \delta_2^y, \delta_2^\phi) \\ &\qquad\vdots \\ &(x_n, y_n, \omega_n^x, \omega_n^y, \delta_n^x, \delta_n^y, \delta_n^\phi) \end{aligned} \tag{5.1}$$

Dabei ist (x_i, y_i) die Position und (ω_i^x, ω_i^y) die Orientierung des Roboters zum Zeitpunkt i. Die Zeit ist nach dem Takt des Systems diskretisiert, der durch die Bildwiederholrate bestimmt ist. Die Fahrbefehle zum Zeitpunkt i sind $(\delta_n^x, \delta_n^y, \delta_n^\phi)$ für die Vorwärts-, Seitwärts- und Drehgeschwindigkeit.

5.8.1 Einfache Vorhersage

Die einfachste Vorhersage benutzt nur die aktuelle Geschwindigkeit des Roboters, also die Differenz der letzten mit der aktuellen Position bzw. die Differenz

der letzten mit der aktuellen Orientierung. Die Differenzen, bzw. der Drehwinkel werden dann mit der Totzeit multipliziert, um eine zukünftige Position bzw. Orientierung zu erhalten. Formal ergibt sich daraus für die Geschwindigkeit der Position $v_x(t) = x_t - x_{t-1}$ und $v_y(t) = y_t - y_{t-1}$ und für die Geschwindigkeit der Orientierung $v_\omega(t) = \omega_t - \omega_{t-1}$. Exemplarisch erfolgen die nächsten Schritte nur für die x-Koordinate, die Berechnungen für die anderen Dimensionen sind äquivalent. Die vorhergesagte x-Position ist dann

$$\tilde{x}_{t+1} = x_t + v_x(t) = x_t + x_t - x_{t-1} = 2x_t - x_{t-1} \tag{5.2}$$

für den nächsten Schritt und

$$\tilde{x}_{t+4} = x_t + 4v_x(t) = x_t + 4x_t - 4x_{t-1} = 5x_t - 4x_{t-1} \tag{5.3}$$

für das vierte Frame in der Zukunft. Dies ist die einfache Vorhersage, wie sie auch für den Vergleich der Vorhersagemodelle im Abschnitt 5.9 verwendet wurde.

Wir können das Modell jedoch noch verbessern, indem wir die Beschleunigung als weiteres Kriterium für die Vorhersage benutzen. Die Beschleunigung ist die Änderung der Geschwindigkeit, somit

$$\begin{aligned} a_x(t) &= v_x(t) - v_x(t-1) = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2}) && (5.4)\\ &= x_t - 2x_{t-1} + x_{t-2} && (5.5) \end{aligned}$$

und es ergibt sich für die Vorhersage des nächsten Frames

$$\begin{aligned} \tilde{x}_{t+1} &= x_t + v_x(t) + a_x(t) && (5.6)\\ &= x_t + (x_t - x_{t-1}) + (x_t - 2x_{t-1} + x_{t-2}) && (5.7)\\ &= 3x_t - 3x_{t-1} + x_{t-2}. && (5.8) \end{aligned}$$

Dieses lineare Modell, welches drei Werte der Vergangenheit benutzt, um implizit die Beschleunigung mit einzuberechnen, wird häufig als erweiterter Kalman-Filter [Welch, 1995] zur Vorhersage von sich bewegenden Objekten benutzt [Han, 1997, Petrov, 2004, Lindstrot, 2004].

Gegebenenfalls können noch weitere Werte der Vergangenheit für die Vorhersage benutzen werden. Man nennt dieses lineare Modell, welches den zukünftigen Wert x_{t+1} einer Zeitreihe aus Werten der Vergangenheit $(x_t, \ldots, x_{t-p})$ durch eine Linearkombination der Form

$$\tilde{x}_{t+1} = a_1 x_t + a_2 x_{t-1} + \ldots + a_{p+1} x_{t-p} \tag{5.9}$$

vorhersagt „autoregressives Modell“ (AR[4]). Werden andere Eingaben zur Vorhersage benutzt, zum Beispiel die Motorfahrwerte der Seitwärtsrichtung δ_t^x und

[4] **a**uto**r**egressive

Vorwärtsrichtung δ_t^y für eine Geschwindigkeitsvorhersage $v(t)$ durch

$$\begin{aligned}\tilde{v}_t &= b_1\delta_t^x + b_2\delta_{t-1}^x + \ldots + b_{q+1}\delta_{t-q}^x \\ &\quad + b'_1\delta_t^y + b'_2\delta_{t-1}^y + \ldots + b'_{q+1}\delta_{t-q}^y\end{aligned} \tag{5.10}$$

so spricht man von einem „Modell des gleitenden Durchschnitts“ (MA[5]). Werden die beiden Verfahren kombiniert, entsteht ein „autoregressives Modell mit gleitendem Durchschnitt“ (ARMA[6]). Dieses Modell wird für die Vorhersage der FU-Fighters Roboter in der Regel verwendet. Im nächsten Abschnitt wird vorgestellt, wie das Modell genau angewendet wird und wie die Gewichte a_i, b_i berechnet werden.

5.8.2 Lineare Vorhersage

Lineare Vorhersage bedeutet, dass vier unterschiedliche gewichtete Summen aller 42 Eingaben die vier Komponenten der zukünftigen Daten bestmöglich approximiert. Es müssen „nur“ die besten Gewichte gefunden werden.

Wir können davon ausgehen, dass jeder Zustandsparameter des Systems, also jeder der 42 Eingabewerte die vier Dimensionen der Vorhersage beeinflusst. Sei nun $\bar{x}_{t-i} = x_{t-i} - x_t$, die relative x-Position bezüglich des Zeitpunkts t und gelte gleiches für $\bar{y}_{t-i}$, $\bar{\omega}_{t-i}^x$, $\bar{\omega}_{t-i}^y$, $\bar{\delta}_{t-i}^x$, $\bar{\delta}_{t-i}^y$ und $\bar{\delta}_{t-i}^\phi$. Wir können dann die linearen Gleichungen

$$\begin{aligned}\tilde{\bar{x}}_{t+4} &= a_1\bar{x}_{t-1} + \ldots + a_6\bar{x}_{t-6} + a_7\bar{y}_{t-1} + \ldots + a_{12}\bar{y}_{t-6} \\ &\quad + a_{13}\bar{\omega}_{t-1}^x + \ldots + a_{18}\bar{\omega}_{t-6}^x + a_{19}\bar{\omega}_{t-1}^y + \ldots + a_{24}\bar{\omega}_{t-6}^y \\ &\quad + a_{25}\delta_{t-1}^x + \ldots + a_{28}\delta_{t-6}^x + a_{29}\delta_{t-1}^y + \ldots + a_{35}\delta_{t-6}^y \\ &\quad + a_{36}\delta_{t-1}^\phi + \ldots + a_{42}\delta_{t-6}^\phi \\ \tilde{\bar{y}}_{t+4} &= a'_1\bar{x}_{t-1} + \ldots + a'_{42}\delta_{t-6}^\phi \\ \tilde{\bar{\omega}}_{t+4}^x &= a''_1\bar{x}_{t-1} + \ldots + a''_{42}\delta_{t-6}^\phi \\ \tilde{\bar{\omega}}_{t+4}^y &= a'''_1\bar{x}_{t-1} + \ldots + a'''_{42}\delta_{t-6}^\phi\end{aligned} \tag{5.11}$$

aufstellen. Falls eine Eingabe die Ausgabe nicht beeinflussen sollte, dann wäre das Gewicht für diesen Parameter gleich Null. Dieses lineare Gleichungssystem können wir jetzt mit Hilfe der linearen Regression so lösen, dass der quadratische Fehler zwischen den Messungen $(\bar{x}_{t+4}, \ldots, \bar{\omega}_{t+4}^y)$ und der Vorhersage $(\tilde{\bar{x}}_{t+4}, \ldots, \tilde{\bar{\omega}}_{t+4}^y)$ minimal wird.

Sei nun X die Matrix mit den aufgezeichneten Daten, wobei in jeder Zeile ein

[5] **m**oving **a**verage
[6] **a**uto**r**egressive **m**oving **a**verage

Eingabevektor für den Vorhersager steht:

$$X = \begin{pmatrix} \bar{x}_6 & \bar{x}_5 & \dots & \bar{x}_1 & \bar{y}_6 & \dots & \delta_1^\phi \\ \bar{x}_7 & \bar{x}_6 & \dots & \bar{x}_2 & \bar{y}_7 & \dots & \delta_2^\phi \\ \vdots & & & & & & \\ \bar{x}_{t-1} & \bar{x}_{t-2} & \dots & \bar{x}_{t-6} & \bar{y}_{t-1} & \dots & \delta_{t-6}^\phi \\ \vdots & & & & & & \\ \bar{x}_{n-5} & \bar{x}_{n-6} & \dots & \bar{x}_{n-10} & \bar{y}_{n-5} & \dots & \delta_{n-10}^\phi \end{pmatrix}. \tag{5.12}$$

Bei n Datensätzen ergibt sich eine Größe der Matrix von $(n-11) \times 42$, denn es werden sieben Datensätze benötigt um eine vollständige Geschichte zu erhalten und die vier letzten Datensätze können nicht benutzt werden, da uns zu diesem Zeitpunkt kein Zustand zum Vergleich mit der Vorhersage zur Verfügung steht. Gesucht ist nun ein Vektor $\vec{a} = (a_1, \dots, a_{42})^T$ mit der Eigenschaft

$$X\vec{a} = \vec{x}, \tag{5.13}$$

wobei $\vec{x} = (\bar{x}_{11}, \bar{x}_{10}, \dots, \bar{x}_n)^T$ der Vektor der vorherzusagenden Daten ist. Da die Daten mit Rauschen behaftet sind, wird keine Lösung existieren. Wir suchen also besser nach einer Lösung, die eine möglichst gute Näherung ist. Mit anderen Worten, der quadratische Fehler $E = (X\vec{a} - \vec{x})^T(X\vec{a} - \vec{x})$ der vorhergesagten Daten $X\vec{a} = (\tilde{\bar{x}}_1 1, \dots, \tilde{\bar{x}}_n)^T$ gegenüber den gegebenen Daten $\vec{x}$ soll minimiert werden. Diese Eigenschaft liefert gerade die Moore-Penrose-Inverse X^+.

Für die Berechnung der Vektoren $\vec{a}'$, $\vec{a}''$ und $\vec{a}'''$ erfolgt auf die gleiche Art und Weise.

5.8.3 Vorwärtsgerichtetes Netz

Anschaulich handelt es sich bei der linearen Vorhersage um ein zweischichtiges neuronales Netz mit 42 Eingabeneuronen und vier Ausgabeneuronen, wobei die Ausgabeneuronen eine lineare Transferfunktion haben. Da dieses Netz so einfach aufgebaut ist, kann man die besten Gewichte explizit durch vier lineare Regressionen lösen.

Wir können aber auch ein nichtlineares neuronales Netz verwenden, was unser dritter Vorhersager ist. Die Anzahl der Eingabe- und Ausgabeneuronen ist natürlich gleich dem Linearen Modell. Wir verwenden ein dreischichtiges neuronales Netz mit 42 Eingabeneuronen, zehn verdeckten Neuronen und vier Ausgabeneuronen (siehe Abschnitt 5.5). Die verdeckten Neuronen haben eine Sigmoide der Form $s_c(x) = \frac{1}{1+e^{-cx}}$ als Transferfunktion, die Ausgabeneuronen eine lineare Transferfunktion. Das Netz wird mit dem Backpropagation-Algorithmus trainiert [Rojas, 1996].

5.9 Ergebnisse

Die Vorhersage mit Hilfe des neuronalen Netzes und der linearen Vorhersage sind mittlerweile ein fester Bestandteil des FU-Fighters-Frameworks und wurden ausführlich getestet. Für jeden Robotertyp kann individuell ein Vorhersagemodell und können individuelle Parametersätze von der Oberfläche aus gewählt und trainiert werden. Wie im Abschnitt 5.11 zu sehen sein wird, liegt die Genauigkeit der Vorhersagen nahe am Optimum.

Die Testaufgabe zur Qualitätsbestimmung besteht darin, in der Form einer liegenden Acht mit maximaler Geschwindigkeit über das Spielfeld zu fahren. Der Roboter soll sich dabei immer zu einem bestimmten Punkt ausrichten, dort wo der Ball liegt (siehe Abbildung 5.9). Der Ball ist in dieser Darstellung nicht zu sehen, er liegt unten links im Bild. Die hellen dicken Linien geben die Orientierung des Roboters und die dunklen dicken Linien die gewünschte Fahrtrichtung an. Es ist deutlich zu sehen, dass der Fahrwunsch, bedingt durch die Trägheit des Roboters, erheblich vom tatsächlich ausgeführten Pfad abweicht.

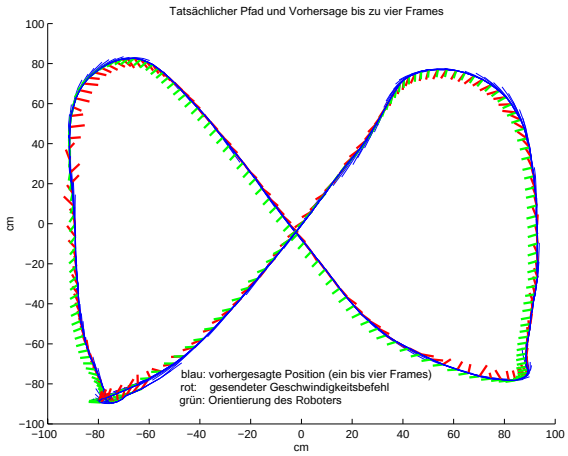

Abbildung 5.9: Die Ausführung der Testaufgabe nach dem Training mit der linearen Vorhersage (siehe Text). Die hellen dicken Linien geben die Orientierung des Roboters an, die dunklen dicken Linien die gewünschte Fahrtrichtung und die dünnen dunklen Linien geben die vorhergesagte Position an.

Die dünnen dunklen Linien geben die vorhergesagte Position an. Sie verbinden die Position des Vision-Systems mit der vorhergesagten Position nach vier Frames. Sie sind zum Glück nicht so gut zu sehen, da sie sehr gut mit dem gefahrenen Pfad übereinstimmen. Die Linien haben eine durchschnittliche Länge von 20cm. In Abbildung 5.10 ist ein vergrößerter Ausschnitt des abgefahrenen Pfades zu sehen.

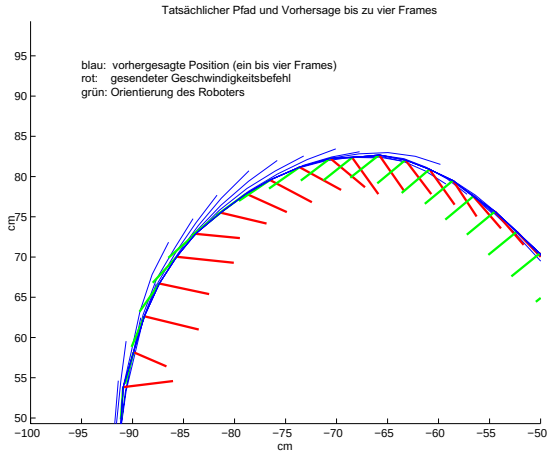

Abbildung 5.10: Ein vergrößerter Ausschnitt von Abbildung 5.9. Die hellen dicken Linien geben die Orientierung des Roboters an, die dunklen dicken Linien die gewünschte Fahrtrichtung und die dünnen dunklen Linien geben die vorhergesagte Position an.

5.9.1 Positionsvorhersage

In Abbildung 5.11 ist zu sehen, dass die Kurve der mit dem neuronalen Netz vorhergesagten Position, hier nur eine Dimension, mit einer in der Zukunft verschobenen Kurve der realen Positionen übereinstimmt. Dies gibt natürlich nur einen ersten Eindruck der Qualität der Vorhersage wieder. Die Kurve der Vorhersage ist etwas glatter als die Kurve der von der Bildverarbeitung gelieferten Positionen, da die Vorhersage naturgemäß das Rauschen der Daten nicht modellieren kann.

In Abbildung 5.13 (links) werden die Fehlerhistogramme der drei Vorhersagemodelle verglichen. Wie deutlich zu erkennen ist, liefert die lineare Vorhersage die genauste Vorhersage. Erwartungsgemäß schneidet das einfache Geschwindigkeitsmodell am schlechtesten ab. Der Fehler der nicht vorhergesagten Position ist der Weg, den ein Roboter im Durchschnitt innerhalb von vier Frames zurücklegt. Die Daten zur Fehlerbestimmung stammen aus einem Versuch, in dem der Roboter mit maximaler Geschwindigkeit über das Feld fährt (siehe Abbildung 5.9). Da ein Roboter mit etwa 2m/s über das Feld fährt, ergibt sich ein Fehler von 27cm. Das ist weit außerhalb des Koordinatensystems in Abbildung 5.13.

Der durchschnittliche Fehler der Positionsvorhersage liegt für die einfache Vorhersage durch die Geschwindigkeit bei 3,48cm, mit dem neuronalen Netz bei 2,65cm und mit der linearen Regression bei 2,13cm.

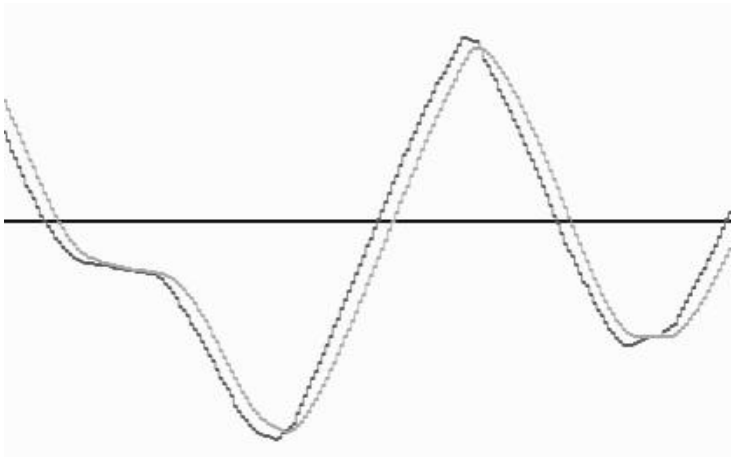

Abbildung 5.11: Beispielsequenz zum Vergleich von Positionsvorhersage (graue Kurve) und der Position der Vision (schwarze Kurve). Es ist nur eine Dimension dargestellt. Die Kurve der Vision entspricht nahezu der Kurve der Vorhersage, wenn man sie um vier Pixel (vier Frames) verschiebt.

5.9.2 Orientierungsvorhersage

Abbildung 5.12 zeigt den Vergleich der um vier Frames vorhergesagten Orientierung mit der von der Vision gemessenen Orientierung. Hier ist noch deutlicher zu erkennen, dass die vorhergesagte Kurve glatter ist als die gemessene. Es fällt auch auf, dass die Messung der Orientierung stark mit Rauschen behaftet ist. Eine Vorhersage hat also den angenehmen Nebeneffekt, dass das Rauschen herausgefiltert wird.

Die rechte Grafik in Abbildung 5.13 gibt zum Vergleich der unterschiedlichen Vorhersagemodelle die Fehlerhistogramme der Orientierung wieder. Hier liefert das nichtlineare neuronale Netz eine bessere Vorhersage als die lineare Vorhersage, im Gegensatz zur Vorhersage der Positionierung. Das einfache physikalische Modell schneidet wieder am schlechtesten ab.

Der durchschnittliche Fehler der Orientierungsvorhersage liegt bei $4,58^o$ für das neuronale Netz, bei $6,47^o$ für die lineare Regression und bei $9,76^o$ für das Geschwindigkeitsmodell.

5.9.3 Vorhersage ohne Fahrwerte

Um zu zeigen, wie wichtig es ist auch die Fahrwerte für die Vorhersage zu berücksichtigen, wurde die lineare Vorhersage sowohl mit als auch ohne Fahrwerte bei

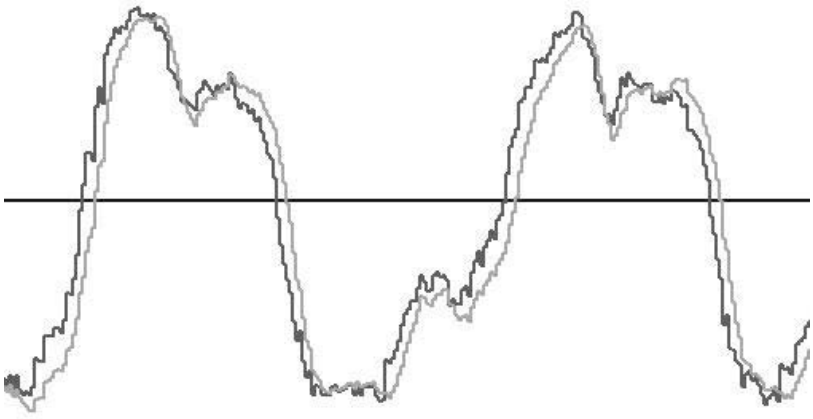

Abbildung 5.12: Vergleich von Orientierungsvorhersage (graue Kurve) und der Orientierungsberechnung der Vision (schwarze Kurve). Auch hier ist die Kurve um vier Frames, die Totzeit, verschoben und geglättet.

sonst gleichen Daten trainiert. Der durchschnittliche Positionierungsfehler[7] für eine Dimension lag bei der Vorhersage mit Fahrwerten bei 1,32cm und bei der Vorhersage ohne Fahrwerten bei 2,43cm. Der Fehler wurde also um circa 50% kleiner.

5.10 Verbesserung der Vorhersage

Ein in der Statistik weit verbreitetes und sehr beliebtes Verfahren zur Vorhersage von Zeitreihen, wie zum Beispiel die Vorhersage der Positionen und Orientierungen, nennt sich „autoregressives integrierendes Modell mit gleitendem Durchschnitt“ (ARIMA[8]) [Shumway, 2000]. Es wird zum Beispiel im Softwarepaket SAS/ETS[9] zur Vorhersage verwendet. ARIMA ist eine Erweiterung des autoregressiven Modells mit gleitendem Durchschnitt (siehe Abschnitt 5.8.2), die auch noch den Fehler der vorausgegangenen Vorhersage für die aktuelle Vorhersage benutzt. Dazu wird die normale lineare Regression als Vorhersage benutzt. Ab dem vierten Frame kann dann der Fehler der berechneten Vorhersage als zusätzlicher Parameter für eine weitere lineare Regression benutzt werden. Zur Qualitätsbestimmung wurden die Positionsvorhersagen mit dieser Methode be-

[7] Bei dem Experiment wurde nicht der quadratische Fehler gemessen, deshalb scheinen die Ergebnisse besser als bei den vorhergehenden Experimenten zu sein.

[8] **a**uto**r**egressive **i**ntegrated **m**oving **a**verage

[9] Die Firma SAS Institute Inc. (www.sas.com) ist Hersteller des gleichnamigen Statistikpakets SAS, für das es viele Zusatzpakete gibt. Zur Vorhersage von wird Zeitreihen das SAS/ETS Paket angeboten.

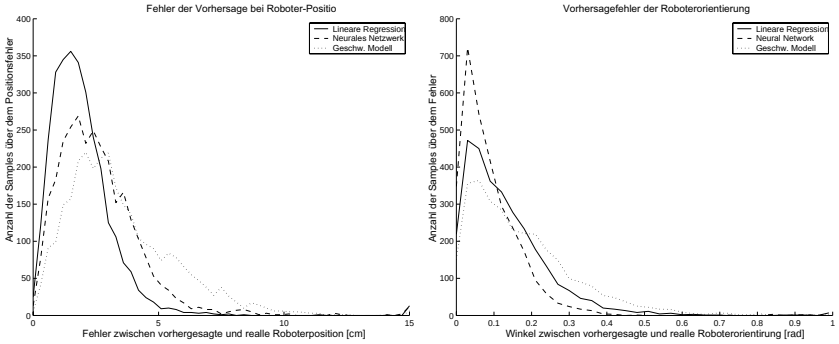

Abbildung 5.13: Vergleich der drei implementierten Vorhersagemodelle. Es ist das Histogramm des Positionierungsfehlers (links) und das Histogramm des Orientierungsfehlers (rechts) dargestellt. Beide Histogramme wurden aus 3000 Testwerten erzeugt.

stimmt. Für die zweite Regression wurde der quadratische Fehler der jeweiligen Dimension als weitere Eingabe benutzt. Wie in den Abbildungen 5.9 und 5.10 zu sehen, ist der Fehler besonders bei harten Richtungsänderungen relativ groß. Da der quadratische Fehler in die Berechnung eingeht, können dadurch nichtlineare Effekte kompensiert werden. Außerdem schaut man indirekt noch mehr in die Vergangenheit, denn der Fehler beruht auf der rekursiven Berechnung älterer Positionen, ohne die Anzahl der Eingabeparameter signifikant zu steigern.

Ein Test mit den gleichen Trainingsdaten wie in Abbildung 5.13 ergibt, dass der durchschnittliche quadratische Fehler der Vorhersage mit dieser Methode auf 1,23cm sinkt. Der durchschnittliche quadratische Fehler der einfachen linearen Regression liegt bei 1,32cm. Das zeigt zwar, dass die Vorhersage besser wird, jedoch nicht sehr stark. Im nächsten Abschnitt wird gezeigt, dass der Vorhersagefehler nicht beliebig minimiert werden kann.

Das Cornell-BigRed-Team verwendete früher eine Art ARIMA, das sich $g - h$ Filter nennt [D'Andrea, 1999]. Es wird jedoch nur die aktuelle Position, die aktuelle Geschwindigkeit und der Fehler der letzten vorhergesagten Position zur aktuellen Position zur Berechnung der Vorhersage der Position sowie der Geschwindigkeit benutzt. Sei $\tilde{v}(t+1)$ die vorhergesagte Geschwindigkeit zum Zeitpunkt t, $\tilde{x}(t+1)$ die vorhergesagte Position und $x(t)$ die Position zum Zeitpunkt t, dann wird die Vorhersage durch

$$\tilde{v}(t+1) = \tilde{v}(t) + h(x(t) - \tilde{x}(t)) \quad (5.14)$$

$$\tilde{x}(t+1) = \tilde{x}(t) + \tilde{v}(t+1) + g(x(t) - \tilde{x}(t)) \quad (5.15)$$

berechnet, wobei h, g Faktoren sind, die geeignet gewählt werden müssen. Über die Genauigkeit ist nichts bekannt. Da die gesendeten Befehle des Roboters nicht

in die Gleichungen eingehen, ist jedoch davon auszugehen, dass die Genauigkeit nicht besonders gut ist.

5.11 Vorhersagegenauigkeit

Die Genauigkeit der Vorhersage hängt von vielen Faktoren ab. Einerseits natürlich von dem verwendeten Vorhersagesystem, aber andererseits auch von der Güte der Eingabedaten. Die Qualität der Eingabedaten liefern eine untere Schranke für die Vorhersage. Sind die Daten schlecht, kann die Vorhersage nichts besser machen. Zwei Sachverhalte könnten die Vorhersage verschlechtern: Positions- und Orientierungs-Fehler im Bildverarbeitungssystem sowie nicht vorhersagbares Verhalten der Roboter. Im Folgenden gehen wir auf diese Sachverhalte genauer ein.

5.11.1 Roboterverhalten

Die Roboter verhalten sich gegenüber der Eingabedaten nicht deterministisch. Die Räder sind nicht perfekt rund, der Roboter „wackelt“ über das Feld. Durch den Teppich kann der Roboter sich an verschiedenen Stellen des Spielfelds unterschiedlich verhalten. Dies wird in den Eingabedaten nicht berücksichtigt. Der Regelkreis auf dem Roboter, gerade der Integrationsteil hat unter Umständen ein längeres Gedächtnis als sechs Frames in der Vergangenheit, die für die Vorhersage verwendet werden. Der Ladestand der Batterien wird nicht berücksichtigt und kann vom Regelkreis nicht egalisiert werden. Diese Unvorhersagbarkeit könnte durch mehrmaliges Wiederholen der gleichen Aufgabe, bzw. einer wiederholt gesendeten Befehlsfolge für den Roboter gemessen werden. Im nächsten Abschnitt werden wir uns mit der Genauigkeit der Vision beschäftigen und den daraus resultierenden Fehler der Vorhersage bestimmen. Wir werden sehen, dass das undeterministische Verhalten des Roboters unbedeutend für den Vorhersagefehler ist.

5.11.2 Gemessene Bildverarbeitungsgenauigkeit

Das Bildverarbeitungssystem gibt nicht die exakten Positionen der Roboter wieder. Die Vision unterliegt sowohl einem globalen, als auch einem lokalen Fehler. Der globale Fehler ist nicht gravierend, da die vorhergesagten Positionen nur lokal berechnet werden. Die Ursache des globalen Fehlers ist vor allem eine ungenaue Transformation der Pixelkoordinaten in Feldkoordinaten, denn diese Abbildung ist nicht trivial (siehe Abschnitt 4.2 in Kapitel 4). Lokale Fehler entstehen vor allem durch das Rauschen der Kamera, durch die diskreten CCD-Punkte, durch die Bewegungsunschärfe der Objekte und durch Helligkeitsschwankungen der Beleuchtung. Durch erhöhen der Belichtungszeit kann man

zwar das Rauschen der Kamera und die Helligkeitsschwankungen (besonders von Leuchtstoffröhren) mindern, dies würde aber die Bewegungsunschärfe vergrößern und außerdem müsste ab einem gewissen Punkt die Bildwiederholrate verringert werden. Wenn ein Roboter oder der Ball sich auf einer festen Position befindet, dann liegt die mittlere Abweichung bei 0,62mm. Erst bei der Bewegung eines Objekts auf dem Spielfeld macht sich ein Fehler bemerkbar. Dadurch, dass auf einem Roboter mehrere Farbpunkte vorhanden sind, ist die Positionsgenauigkeit höher als bei dem Ball, der nur aus einem Farbpunkt besteht. Um die Genauigkeit der Vision zu messen, wurden einige Experimente mit einem rollenden Ball durchgeführt. Auf Experimente mit einem fahrenden Roboter wurde verzichtet, da der Versuchsaufbau ungleich komplexer gewesen wäre. Später wird jedoch gezeigt, wie man indirekt die Genauigkeit der Roboterpositionen messen kann.

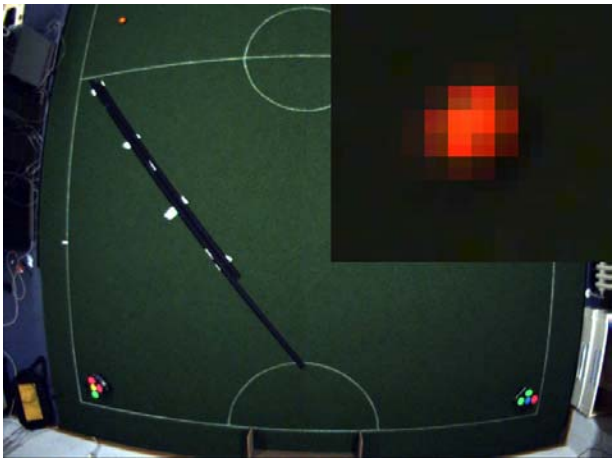

Abbildung 5.14: Der Versuchsaufbau zum Messen der Positionsgenauigkeit des Balls durch die Bildverarbeitung. Die schwarze Schiene links im Bild zwingt den Ball in eine feste Bahn. Rechts oben ist eine 1500% Vergrößerung des Balls zu sehen. Der Ausschnitt besteht aus 22 × 22 Bildpunkten, das gesamte Bild aus 780 × 582 Bildpunkten. Die Auflösung ist durch die Farbmaske noch einmal um ein viertel reduziert.

Die aktuelle Kamera der FU-Fighters hat eine horizontale Auflösung von 780 Bildpunkten. Die Kameras hängen quer über dem Spielfeld, das eine Breite von 400cm hat. Die Kamera muss aber etwas mehr als die Spielfeldbreite im Blickfeld haben, denn durch die Höhe der Roboter erscheinen diese, wenn sie am Rand des Spielfelds stehen, etwas außerhalb. Außerdem ist die Fläche, die ein Bildpunkt auf dem Spielfeld repräsentiert, nicht konstant über das ganze Spielfeld, da durch die Kameraverzerrung die Spielfeldauflösung in der Mitte der Kamera etwas höher ist als an den Rändern. Die genauen Parameter hängen von dem verwendeten Objektiv ab. Die durchschnittliche Seitenlänge eines Pixels ist et-

wa 420cm/780, also 0,5cm. Die Kamera besitzt eine Farbfiltermaske, dadurch werden die Farben für jeden Pixel interpoliert und die Auflösung sinkt noch einmal. Da ein Farbmarker aus mehren Bildpunkten besteht und der Schwerpunkt der zugehörigen Pixel als Position des Markers genommen wird, erhöht sich die Auflösung wieder etwas, was als Superresolution bezeichnet wird.

Um die Genauigkeit der von der Vision gelieferten Ballposition zu messen, wurde eine Schiene auf das Spielfeld gelegt, in welcher der Ball sehr gerade rollen kann (Abbildung 5.14). Um den globalen Fehler abzuschätzen, wurde der Ball zuerst auf der Schiene hin und her gerollt, wobei die Positionen aufgezeichnet wurden. Anschließend wurde eine lineare Regression mit den Daten durchgeführt. Die mittlere Abweichung von der linearen Funktion liegt bei 3,4mm. In dem Fehlerdiagramm von Abbildung 5.15 ist deutlich der globale Fehler der Vision in diesem Teil des Spielfelds zu erkennen.

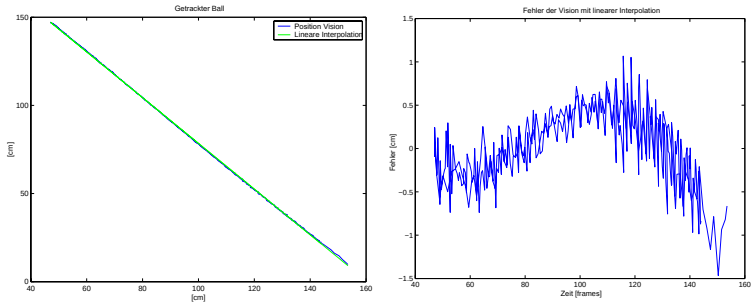

Abbildung 5.15: Globaler Fehler der Positionierungsmessung. Durch die nicht perfekte Entzerrung des Kamerabildes erscheint das Bild einer Geraden als Kurve. Links ist der Pfad mit der linearen Regression zu sehen. Rechts der Positionierungsfehler, der von der globale Position abhängt.

Benutzen wir zur Interpolation ein Polynom höheren Grades, dann verliert der globale Fehler an Bedeutung. In Abbildung 5.16 wird ein Polynom vom Grad 4 benutzt. Der mittlere Fehler liegt bei 1,8mm. Mit einem Polynom 6. Grades wird der Fehler nicht messbar kleiner.

Rollt nun der Ball frei über den Teppich, dann wird der Fehler der Vision durch den auf einer nicht glatten Fläche rollenden Ball beeinflusst. Dieser Gesamtfehler kann dazu benutzt werden, Aussagen über die prinzipielle Vorhersagbarkeit des Balls zu treffen. Zum Beispiel um eine Ballabfangposition zu berechnen oder um die Flugbahn eines hochgeschossenen Balls zu berechnen. In Abbildung 5.17 wurde der Ball quer über eine Spielfeldhälfte geschossen. Durch die Unebenheiten des Untergrundes liegt der mittlere Fehler der Ballposition bezüglich der Interpolation durch eine Gerade bei 2,20cm und bezüglich der Interpolation eines Polynoms sechsten Grades 6 bei 0,35cm. In Abbildung 5.18 ist die gemessene Ballgeschwindigkeit wiedergegeben.

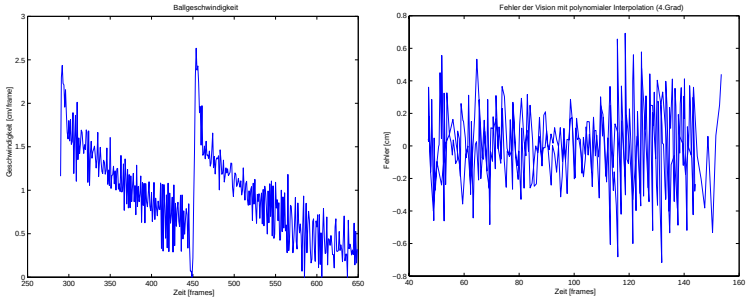

Abbildung 5.16: Lokaler Fehler der Positionierungsmessung. Durch ein Polynom vom Grad 4 wird der globale Fehler herausgerechnet. Es bleibt der lokale Fehler (rechts), der von der Geschwindigkeit des Balls abhängt (links).

5.11.3 Berechnete Bildverarbeitungsgenauigkeit

Wenn wir annehmen, dass ein Roboter auf Grund seiner eigenen Trägheit glatte Bewegungen ausführt, das heißt, das Rauschen der Bewegung sehr viel kleiner ist als das Rauschen der Positionsmessung, dann ist die Differenz zwischen dem glatten Pfad und dem gesehenen Pfad das Rauschen der Positionierungsmessung. Wie im vorherigen Abschnitt gezeigt wurde, kann diese Annahme für den Ball nicht gemacht werden, denn dieser ist sehr leicht und wird durch den Teppichboden stark beeinflusst.

Um einen glatten Pfad aus den aufgezeichneten Daten des Roboters zu berechnen, nehmen wir drei Positionen in der Zukunft und drei Positionen in der Vergangenheit bezüglich seiner aktuellen Position und trainieren eine Vorhersage (ein lineares Modell), welches die aktuelle Position vorhersagt. Unter der Annahme, dass der Roboter innerhalb dieser Distanz glatt fährt und die Vorhersage den verrauschten Pfad glättet, ist die Abweichung der vorhergesagten Position zur gemessenen Position der Fehler der Bildverarbeitung.

Führen wir dieses Experiment durch, dann erhalten wir einen mittleren Fehler von 3mm. Bezogen auf eine Koordinate liegt der mittlere Fehler bei 2,1mm. Dies entspricht etwa dem Fehler von 1,8mm der Vision, der mit dem Ball auf einer Schiene gemessen wurde (siehe Abbildung 5.16).

Um zu zeigen, dass die Vorhersage das Rauschen aus den Daten eliminiert, wird ein künstlich generierter glatter Pfad genommen und gaußsches Rauschen addiert (siehe Abbildung 5.19). Wenn nun wieder die Vorhersage benutzt wird, um eine Position mit Hilfe der drei vorhergehenden Positionen und der drei nachfolgenden Positionen zu schätzen, ergibt sich ein Fehler, der um den Faktor 1,2 höher ist, als das aufmodulierte Rauschen (siehe Abbildung 5.20). Das heißt, der durch die Vorhersage gemessene Fehler von 3mm ist eine obere Schranke für das Rauschen der Bildverarbeitung.

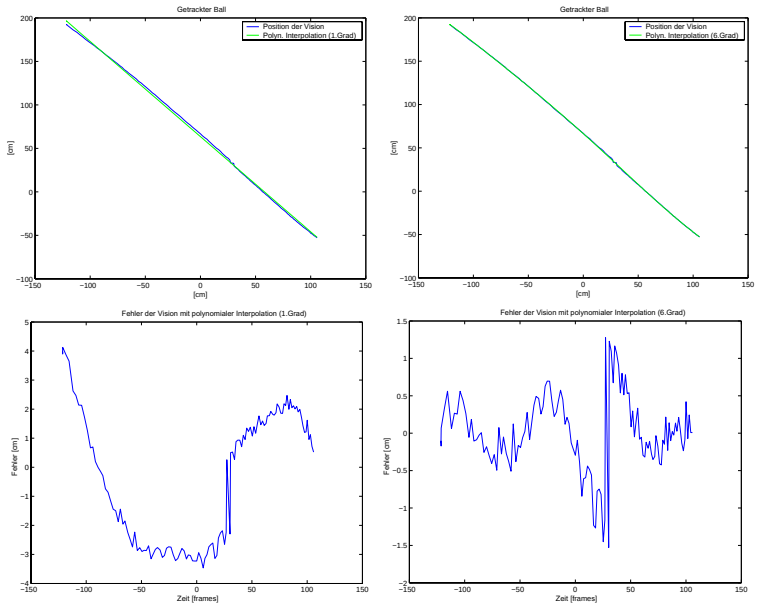

Abbildung 5.17: Die Bahn eines frei rollenden Balls und die Interpolation durch eine Gerade (links oben), ein Polynom 6. Grades (rechts oben) und die Abweichungen von den Interpolierenden (untere Reihe). In Abbildung 5.17 ist die Geschwindigkeit des geschossenen Balls dargestellt.

Wir wandeln nun das Experiment in der Form so ab, dass wir nicht die Position innerhalb eines Zeitfensters betrachten, sondern mit Hilfe eines aktuellen Zeitfensters die Position nach vier Frames in der Zukunft schätzen. Zum Training der Vorhersage wird kein Rauschen addiert. Zu dem künstlich generierten Pfad addieren wir dann ein durchschnittliches Rauschen von 2,1mm für jede Dimension, so wie wir es bei dem realen Pfad geschätzt haben. Mit diesen Daten wird dann die Vorhersagegenauigkeit gemessen. Sie liegt mit diesen Daten bei 3,8mm. Dies ist die beste Vorhersage, die man mit diesen verrauschten Daten erreichen kann. Der Positionierungsfehler für die Vorhersage für den realen Roboter lag bei 1,32cm. Ein sehr guter Wert, wenn man berücksichtigt, dass der künstliche Pfad durch Sinus- und Kosinus-Funktionen generiert wurde, die durch eine lineare Vorhersage sehr gut approximiert werden können.

5.12 Zusammenfassung

In diesem Kapitel habe ich drei Methoden vorgestellt, welche die Roboterdynamik automatisch lernen. Sie überwinden die Totzeit des Systems, indem sie

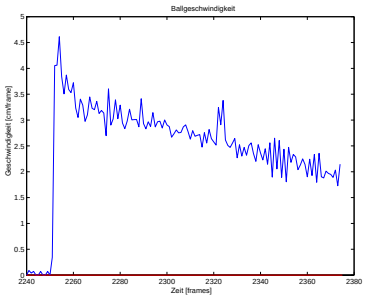

Abbildung 5.18: Die Ballgeschwindigkeit bei dem Experiment von Abbildung 5.17.

die zukünftige Roboter-Positionen und -Orientierungen berechnen. Sie basieren auf verschiedenen Ansätzen, ihre Ergebnisse unterscheiden sich aber nicht schwerwiegend voneinander. Mit Hilfe der gelernten Vorhersage ist es dann möglich, die Roboter viel präziser und schneller zu steuern, ohne über das Ziel hinauszuschießen oder die Bewegung unnötig zu verlangsamen. Ich gehe nun auf einige Probleme sowie eventuelle Verbesserungen ein und beschreibe weitere Anwendungsmöglichkeiten der gelernten Roboterdynamik.

Es ist nicht sehr befriedigend, dass immer eine feste Anzahl von Frames für die Vorhersage benutzt wird. Aus dem durchgeführten Experimenten zur Delaybestimmung ergeben sich — je nach Fahrweise — unterschiedlich viele Frames. Manche Aktionen der Roboter, wie zum Beispiel Schießen, haben ein viel geringeres Delay, da die Trägheit des Roboters nicht überwunden werden muss. Doch wie groß die Verzögerung bei einer derartigen Aktion ist, wurde bisher nicht gemessen. Aber welches Delay ist optimal? Ein zu großes Delay würde die Vorhersage sehr ungenau und den Roboter auch weniger kontrollierbar machen.

Eine Möglichkeit zur optimalen Delaybestimmung ist es, das System mit unterschiedlichsten Annahmen über die Totzeit zu trainieren. Das angenommene Delay mit dem besten Ergebnis wäre dann das optimale Delay. Leider gibt es im Gesamtsystem so viele Feineinstellungen, die das Verhalten zu einem gegebenen Delay beeinflussen, dass es eine kaum zu bewältigende Aufgabe darstellt, auch diese Einstellungen dem aktuell zu testenden Delay anzupassen.

Die Delaymessung könnte auch dazu benutzt werden, die Parameter der Regelkreise des Roboters zu optimieren. Der Roboter soll möglichst schnell seine vorgegebene Geschwindigkeit annehmen. Je schneller dies geschieht, desto geringer ist das gemessene Delay aus den beobachteten Daten. Die Parameter des Roboters könnten somit mit dem Delay als Bewertungsfunktion gelernt werden. Wenn das Delay über die Kovarianz gemessen wird, dann ist deren Wert zusätzlich ein Gütemaß für die Übereinstimmung der vorgegebenen Fahrwerte mit der

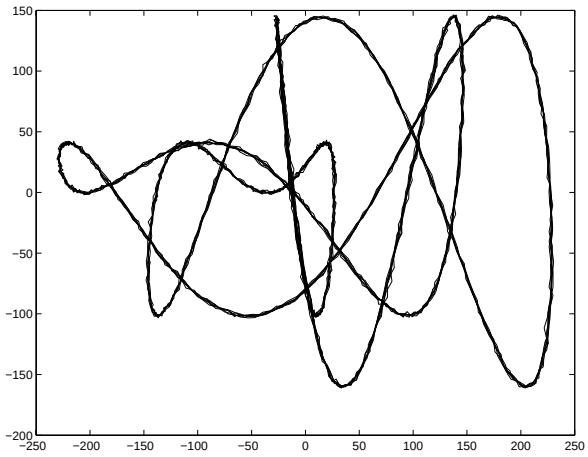

Abbildung 5.19: Ein künstlich generierter Pfad, eine Lissajous-Funktion, mit einem aufmodulierten gaußschen Rauschen.

umgesetzten Geschwindigkeit.

Wünschenswert wäre es auch, Trainingsdaten aus einem Spiel zu generieren, denn dabei führt der Roboter genau die Situationen aus, die auch gelernt werden sollen. Außerdem könnte man dann auch online lernen, also unterschiedliche Roboterzustände, wie zum Beispiel verschlissene Motoren, berücksichtigen.

Das Vorhersagesystem kann auch für andere Aufgaben als die Überbrückung der Totzeit benutzt werden. Im nächsten Kapitel geht es um die Nutzung der gelernten Roboterdynamik als Robotermodell in einem Simulationssystem, in Kapitel 7 wird die Vorhersage zur Korrektur des internen Bewegungsmodells im Roboter selbst benutzt. Im Verhaltenssystem gibt es außerdem ein Modul, welches anhand der Abweichung der Vorhersage von der wirklichen Bewegung des Roboters die Zufriedenheit des Roboters berechnet und bei einer zu großen Abweichung bzw. Unzufriedenheit Alarm auslöst. Dadurch kann der Anwender erkennen, ob ein Hardwaredefekt vorliegt.

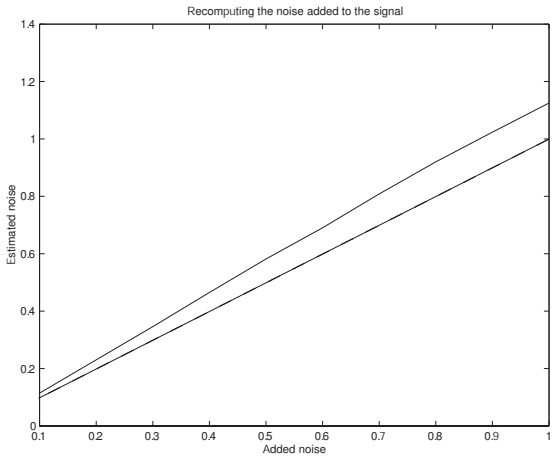

Abbildung 5.20: Wir benutzen den künstlich generierten Pfad und addieren unterschiedlich starkes Rauschen (x-Achse) auf den Pfad. Anschließend wird eine lineare Vorhersage mit den vorherigen vier und den nachfolgenden vier Punkten benutzt, um den aktuellen Punkt und damit das Rauschen des aktuellen Punktes zu schätzen (y-Achse). Wie zu sehen ist, liegt die Schätzung (durchgezogene Linie) immer leicht über dem tatsächlichen Rauschen (gestrichelte Linie).