4 Grundlagen der Gangerzeugung

Die in dieser Arbeit vorgestellte Methode zur Bewegungskoordination beruht auf einer Abwandlung und Rekombination einfacher bekannter Gangmuster, um daraus die notwendigen Bewegungen zur Bewältigung komplexer Situationen zu generieren.

Um diese Strategie anwenden zu können, müssen für den jeweiligen Roboter die Gangmuster für einfache Bewegungen zur Verfügung gestellt werden können. Dies sollte einen möglichst geringen Aufwand an Rechenzeit erfordern, da die Detektion des unbekannten Geländes und die darauf zu bestimmende Reaktion einen maximalen Anteil an der Gesamtrechenkapazität und -zeit zur Verfügung haben sollen.

Die vorgestellte Kontrollstrategie soll dabei möglichst allgemein verwendbar sein. Das heißt, auch Maschinen, deren Kinematik sich von der der hier vorgestellten Roboter unterscheidet, sollen diese Strategie nutzen können. Um die dazu notwendigen Gangmuster für beliebige Gehmaschinen mit mindestens zwei symmetrischen Beinpaaren berechnen zu können, wurde im Fachgebiet Mechanik die Bibliothek WalkingLib entwickelt. Diese Bibliothek bietet die Möglichkeit, kinematisch korrekte Gangmuster für vorgegebene Maschinen zu erstellen. Aufgrund ihrer ausgesprochen generisch angelegten Struktur sind die benötigten Rechenzeiten jedoch relativ hoch. Um einen möglichst großen Zeitanteil eines Schrittzyklus für die Koordination und Adaption der Bewegung zur Verfügung stellen zu können, wurde mit neuronalen *feedforward* Netzen eine Möglichkeit gefunden, eine sowohl schnelle als auch hinreichend genaue Approximation der analytisch erzeugten Gangmuster bereitzustellen. Damit kann der Prozeß der Gangmustererzeugung in zwei Teile zerlegt werden: In die aufwendige Berechnung der Gangmuster und das Training der neuronalen Netze auf der einen Seite sowie die schnelle Approximation dieser Muster mit Hilfe der trainierten neuronalen Netze auf der anderen Seite.

Die Funktionsweisen der WalkingLib und der neuronalen Netze werden in diesem Kapitel erläutert. Außerdem wird der Vergleich der jeweils notwendigen Rechenzeiten und der erzielten Genauigkeit präsentiert.

4.1 Gangerzeugung mit der WalkingLib

Bei vielen existierenden Gehmaschinen sind die verwendeten Gangmuster sehr einfach gehalten und teilweise kinematisch nicht korrekt. Neben den Zwangskräften, die dadurch in den Beinen und im Zentralkörper der Maschinen entstehen können, bedeutet das Fehlen einer analytischen Gangmustererzeugung in der Regel auch ein geringeres Repertoire an Gangarten.

Bei Tarry I wurde bereits in der Arbeit von Amendt [Ame95] Wert auf eine kinematisch korrekte Berechnung harmonisch wirkender Schrittmuster gelegt. Die Form der berechneten Trajektorien orientierte sich dabei am biologischen Vorbild.

Tabelle 6: Gangparameter der WALKINGLIB

Parameter	Beschreibung
v_x	Geschwindigkeit des Zentralkörpers in Längsrichtung
v_y	Geschwindigkeit des Zentralkörpers in Querrichtung
ω_z	Rotationsgeschwindigkeit um die Hochachse der Maschine
h_r	Senkrechter Abstand des Zentralkörperbezugspunktes über der Aufstandsebene
h_l	Maximaler senkrechter Abstand des Fußpunktes über der Aufstandsebene
w_j	Schrittbreite des Beinpaares j
s_j	Verschiebung des Schwingungsmittelpunktes des Beinpaares j
α_x	Neigungswinkel des Zentralkörpers um Längsachse (Rollen)
α_y	Neigungswinkel des Zentralkörpers um die Querachse (Nicken)
Δx	Amplitude der überlagerten Zentralkörperschwingung (Längs)
Δy	Amplitude der überlagerten Zentralkörperschwingung (Quer)
φ_x	Phasenverschiebung der überlagerten Schwingung (Längs)
φ_y	Phasenverschiebung der überlagerten Schwingung (Quer)
T	Zyklusdauer
β_i	Duty-Faktor des Beines i.
ϕ_i	Phase des Beines i

Die dort vorgestellte Berechnungsmethode beschränkte sich jedoch auf den Geradeauslauf und auf Kurven mit großen Radien. Dieser Algorithmus konnte dann auf die beliebige Überlagerung von Bewegungen in die x- und y-Richtung zusammen mit frei wählbaren Kurvenradien erweitert werden. Dadurch war neben dem Seitwärtsgang auch das Drehen auf der Stelle möglich.

Diese Methode der Gangerzeugung blieb aber auf die Kinematik der Gehmaschine TARRY I und einen kleinen Satz von Parametern beschränkt. Zudem war es praktisch nicht möglich, zusätzliche Funktionalität, wie beispielsweise die Ausgabe der Daten in einem anderen Format oder die Optimierung von Gangmustern, zu implementieren. Auch diese Einschränkungen können mit der Bibliothek WALKINGLIB [BFG+98] überwunden werden. Die WALKINGLIB berechnet mit Hilfe der Beschreibung einer Roboterstruktur und der gewünschten Gangparameter die entsprechenden Gangmuster. Die Ergebnisse dieser Berechnung können in verschiedenen Formaten ausgegeben werden, beispielsweise in Form der zugehörigen Gelenkkoordinaten oder als Datensätze zur Berechnung von Animationen.[3]

[3]Die Bibliothek ist auch in der Lage, Gang- oder Strukturparameter einer Maschine hinsichtlich

Die kinematische Beschreibung der Maschine erfolgt mit Hilfe eines C++-Programmes, in dem einfache Elemente wie Gelenke oder Verbindungsstücke auf geeignete Weise parametriert und kombiniert werden. Hiermit ist es möglich, die Geometrie und Kinematik der gewünschten Mechanismen entsprechend zu repräsentieren. Diese Beschreibung dient der WALKINGLIB im weiteren Verlauf als Modell.

Die gewünschte Bewegung wird in Form verschiedener Gangparameter, aufgelistet in Tabelle 6, vorgegeben. Mit ihnen ist es möglich, eine sehr große Palette von Gangarten zu beschreiben und die zugehörigen Gangmuster zu erzeugen.

4.2 Erzeugte Gangmuster

Die Gangparameter können, wie in Tabelle 6 bereits angedeutet, grob in drei Blöcke unterteilt werden. Die Parameter des ersten Blocks ändern Form, Lage und Orientierung der Trajektorien ebenso wie Lage und Orientierung des Zentralkörpers. Die Parameter des zweiten Block wirken sich nur auf die Lage und Orientierung des Zentralkörpers aus. Der dritte Block regelt das Zusammenspiel der einzelnen Beine und beeinflußt dazu die zeitliche Lage und Aufteilung der Trajektorien, verändert diese aber ansonsten nicht.

Je nach Anwendungsfall wird jeweils ein unterschiedlicher Teil der Parameter genutzt. Aus dem zweiten Block dienen die Parameter der überlagerten Schwingungen (Δx, Δy, φ_x und φ_y) hauptsächlich dazu, durch eine Verlagerung des Gesamtschwerpunktes der Maschine die statische Stabilität während des Ganges zu erhöhen. Dies ist besonders für Vierbeiner interessant, da die statische Stabilität dort während der Schwingphasen ein deutliches Problem darstellt, das mit dieser Methode verringert werden kann [BFG+98] [HM00]. Für einen Sechsbeiner sind diese Parameter kaum relevant. Die beiden restlichen Parameter dieses Blockes beeinflussen den Nick- und Rollwinkel des Zentralkörpers. Die Neigung der Maschine wird mit der in dieser Arbeit vorgeschlagenen Koordinationsmethode allerdings mit anderen Mitteln (vgl. Abschnitt 5.2.2) beeinflußt. Aus den genannten Gründen wird der komplette zweite Parameterblock in dieser Arbeit nicht weiter berücksichtigt. Alle folgenden Erläuterungen gehen davon aus, daß die zugehörigen Parameter zu Null gesetzt sind.

Die Trajektorie eines einzelnen Fußes teilt sich auf in eine Stemm- und eine Schwingphase. Deren Gesamtdauer entspricht der Zykluszeit, die auch als Schrittdauer bezeichnet wird. In der Stemmphase stützt das zugehörige Bein den Zentralkörper und sorgt gleichzeitig für die Lokomotion der Maschine. In der Schwingphase wird der Fuß vom Endpunkt der Stemmphase (*Posterior Extreme Position*, PEP) zum Anfangspunkt der folgenden Stemmphase (*Anterior Extreme Position*, AEP) bewegt. Das Verhältnis der Dauer der Stemmphase zur Schrittdauer stellt den Duty-Faktor

der Kippstabilität zu optimieren. Diese spezielle Funktionalität soll hier aber nicht näher betrachtet werden [FBG+00b].

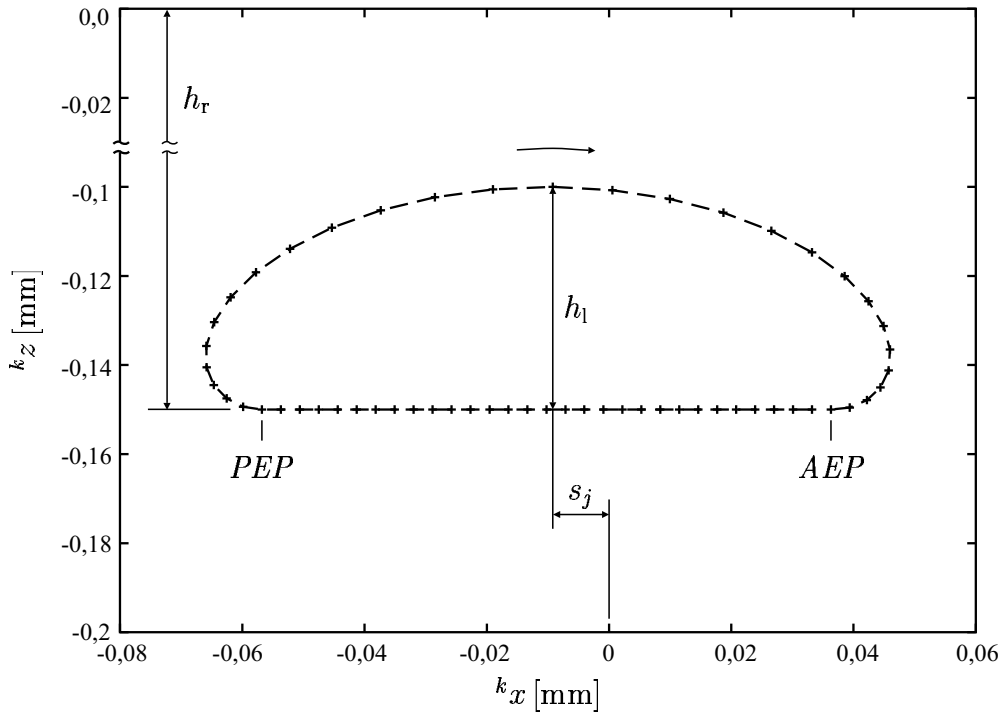


Abbildung 26: Diskretisierte Fußpunkttrajektorie für einen reinen Vorwärtslauf des mittleren linken Beines.

β_i des jeweiligen Beines dar. Die Stemmlinie, die *AEP* und *PEP* verbindet, wird maßgeblich von Parametern v_x, v_y und ω_z bestimmt. Die Körperhöhe h_r und die Schritthöhe h_l sind bei der Berechnung mit der WALKINGLIB für alle Beine gleich.

In der Schwingphase beschreibt der Fuß im Inertialsystem den Weg einer gestauchten Zykloide. Deren Berechnung wird in [BFG+98] beschrieben. Durch die kontinuierliche Bewegung des Zentralkörpers ergibt sich im k-Koordinatensystem des zugehörigen Beines beispielsweise die in Abbildung 26 gezeigte Trajektorie in der x-z-Ebene.

Aus der Kooperation der Beine ergeben sich die unterschiedlichen Gangarten, die durch die Parameter β_i und ϕ_i charakterisiert werden. Die Phasenlage ϕ_i jedes Beines gibt dabei an, an welcher Stelle des Gesamtzyklus die Schwingphase eines Beines beginnt. Die Zyklusdauer bestimmt, in welcher Zeit eine Trajektorie einmal komplett durchlaufen wird und ist für alle Beine gleich.

In der langsamsten Gangart, dem Wellengang, ist zu jedem Zeitpunkt maximal ein Bein in der Schwingphase. Die Maschine hat durch die hohe Zahl der stützenden Beine eine maximale statische Stabilität und Aufnahmemöglichkeit für die größte mögliche Nutzlast. Die Polygone, die durch die Beine in der Stemmphase gebildet werden, sind in Abbildung 27 dargestellt.

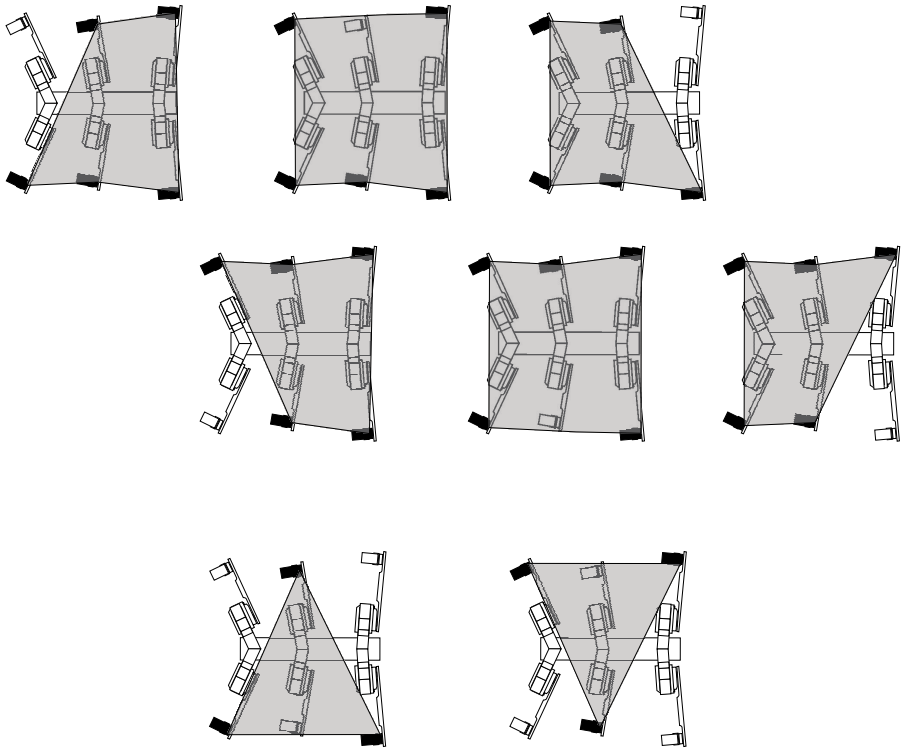


Abbildung 27: Aufstandspolygone (grau) und Beine in der Stemmphase (schwarz) beim Wellengang (a) und Tripod Gang (b)

In der schnellsten Fortbewegungsart, dem Tripod Gang, sind je drei Beine in der Schwing- und drei in der Stemmphase. Das dreieckige Aufstandspolygon wird dabei jeweils von dem statisch stabilen Dreibein, dem sogenannten Tripod, gebildet.

Eine der möglichen Zwischenformen dieser beiden Gangarten ist der Tetrapod Gang. Hier befinden sich jeweils zwei Beine in der Schwingphase, die restlichen vier Beine stützen den Körper. Wie in dem aus der Gangart abgeleiteten Gangdiagramm in Abbildung 28 gesehen werden kann, existieren zwischen dem Wechsel der aufstehenden Beine in der Regel Phasen, in denen alle Beine Bodenkontakt haben.

Neben diesen explizit erwähnten Gangarten können durch die Kombination von Duty-Faktor und Phasenlage beliebige Gangarten erzeugt werden. Somit sind Mischformen der vorgestellten Gangarten realisierbar.

In der im folgenden Kapitel beschriebenen Strategie zur autonomen Bewegungskoordination wird ein Teil der vorgestellten Parameter konstant gehalten. Zur Erzeugung der Trainingsmuster werden lediglich die Geschwindigkeit in x- und y-Richtung, v_{x} und v_{y}, die Rotationgeschwindigkeit ω_{z} um die z-Achse, und die Höhe des Zentralkörpers h_{r} und des Fußes h_{j} variiert. Wegen seiner im Vergleich zu den anderen Gangarten relativ hohen Geschwindigkeit wird nur noch auf den Tripod Gang

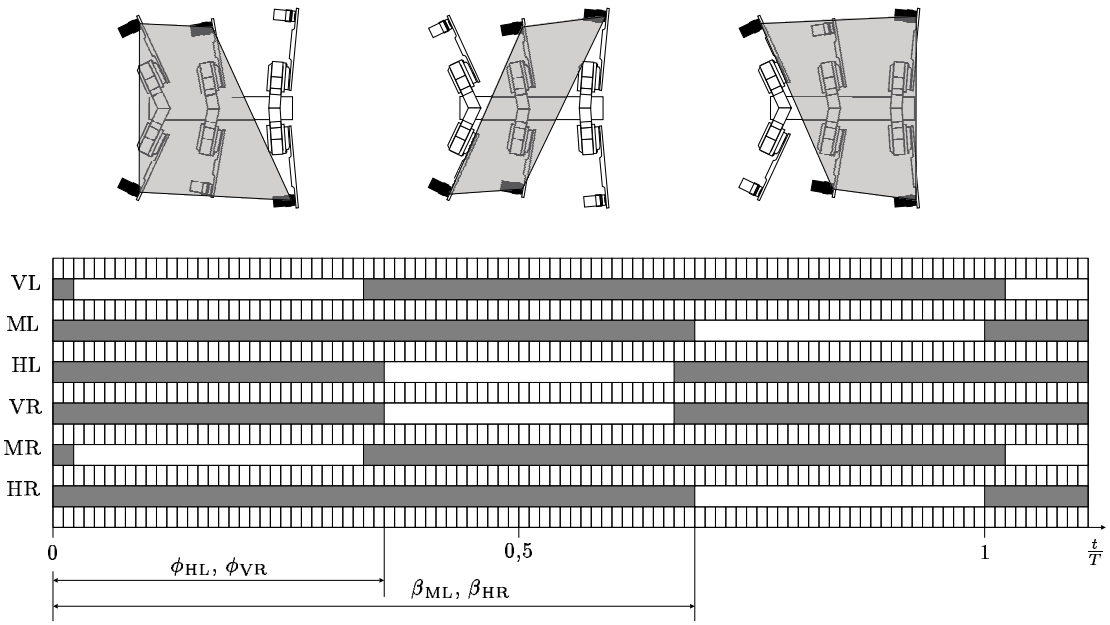


Abbildung 28: Gangdiagramm des Tetrapod Ganges

zurückgegriffen. Die vorgestellten Methoden sind aber auch auf alle andere Gangarten anwendbar.

4.3 Ganggenerierung mit neuronalen Netzen

Die Ganggenerierung mit der WALKINGLIB hat den Nachteil der niedrigen Ausführungsgeschwindigkeit. Um dieses Problem zu umgehen, wird ein Verfahren eingesetzt, für das typische Gangmuster nur vorab analytisch berechnet werden. Diese Muster dienen dann als Trainingsdaten für die Approximation durch neuronale Netzwerke.

Diese Netze haben eine Reihe von günstigen Eigenschaften. Nachdem sie mit einer Auswahl an vorberechneten Ein- und Ausgangswerten auf eine Problemstellung hin trainiert wurden, sind sie in der Lage, auch auf ähnliche, nicht trainierte Eingangsmuster passende Ausgangsdaten zu generieren. Zusammen mit der hohen Ausführungsgeschwindigkeit prädestiniert dieses, als Generalisierungsfähigkeit bezeichnete Verhalten, diese Methode für die gewünschte Nutzung.

Die Verfahren der künstlichen neuronalen Netzwerke entstammen dem Bereich der künstlichen Intelligenz. Sie versuchen, die Informationsverarbeitung von Lebewesen durch eine hohe Anzahl verbundener einfacher Verarbeitungseinheiten, den Neuronen, abstrahiert nachzubilden. Diese Neuronen können mit Hilfe gewichteter und gerichteter Verbindungen untereinander Informationen austauschen.

Jede Zelle kann Signale von anderen Zellen empfangen, weiterverarbeiten und an andere Zellen senden. Es gibt praktisch unbegrenzt viele Möglichkeiten, solche Zellen zu verbinden, um damit Netzwerke aufzubauen. Eine breite Übersicht der verschiedenen Methoden bietet ZELL [Zel94].

In dieser Arbeit sollen nur die sogenannten *feedforward* Netze behandelt werden. Hier werden jeweils mehrere Zellen, auch Units genannt, zu Schichten zusammengefaßt. Es existiert je eine Eingabe- und eine Ausgabeschicht (*input* und *output*). Die Zahl der darin enthaltenen Zellen entspricht der Anzahl der vorhandenen Eingangs- bzw. der gewünschten Ausgangswerte. Zwischen diesen Schichten können sich beliebig viele weitere Schichten befinden. In der Regel sind dies zwei oder mehr. Da diese Schichten nach außen nicht in Erscheinung treten, werden sie auch verdeckte Schichten (*hidden layer*) genannt. In einem *feedforward* Netz laufen die Signale immer nur von der Eingabe- in die Richtung der Ausgabeschicht. Über gewichtete Verbindungen sendet jede Zelle ihr Ausgangssignal zu jeder einzelnen Zelle der nachfolgenden Schicht. Innerhalb einer Schicht existieren keine Verbindungen. Weder werden Schichten übersprungen (*shortcuts*) noch gibt es rückwirkende (*recurrent*) Verbindungen. Ein solches Netz ist in Abbildung 29 dargestellt. Um beispielsweise ein Netz in der dort vorliegenden Konfiguration zu bezeichnen, wird im folgenden die Notation 3-4-4-2 verwendet. Die erste und die letzte Zahl geben die Anzahl der Ein- bzw. Ausgangsneuronen an. Die dazwischenliegenden Zahlen stehen für die Anzahl der Units der jeweiligen verdeckten Schichten.

Die Zellen sind aufgebaut, wie in Abbildung 30 dargestellt. Die Arbeit einer solchen Zelle teilt sich in drei Bereiche, die Propagierung, die Aktivierung und die Ausgabe auf.

Durch die Propagierungsfunktion wird die Summe der gewichteten Signale aller Zellen der Vorgängerschicht ermittelt. Bei n Vorgängerzellen für die Zelle m gilt dann:

$$\text{prop}_m = \sum_{i=1}^{n} o_i w_{im}$$

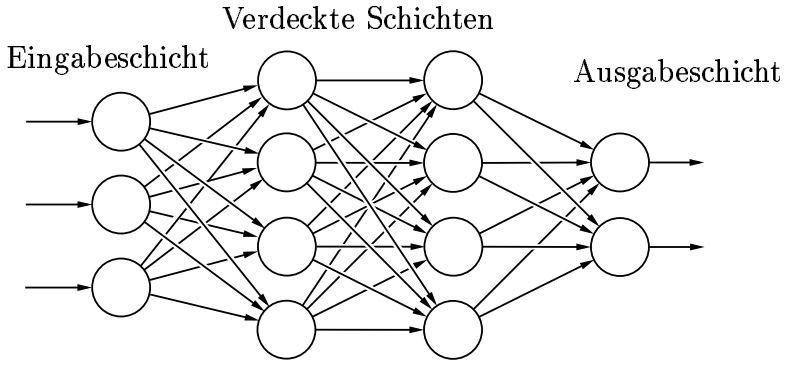


Abbildung 29: Beispiel eines *feedforward* Netzwerkes

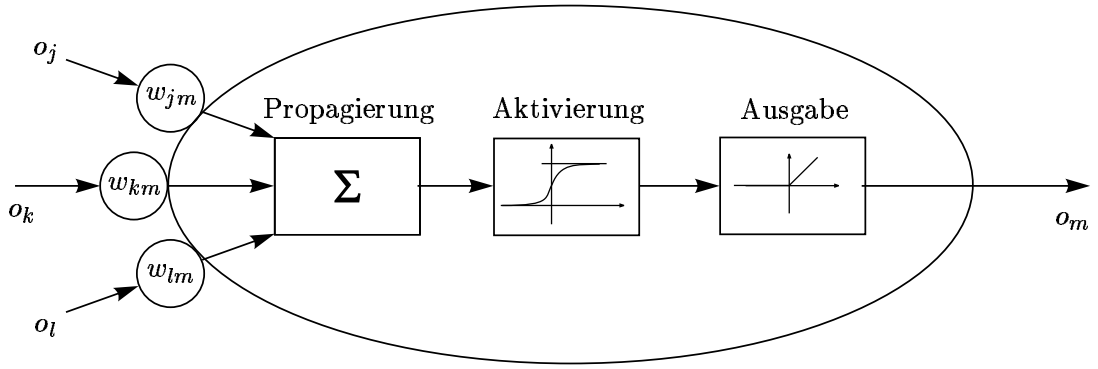


Abbildung 30: Unit eines neuronalen Netzes

Die Aktivierungsfunktion $\text{akt}_m = f(\text{prop}_m)$ legt die Stärke der Zellenaktivierung fest. Eine oft verwendete Funktion ist die sigmoide (s-förmige) logistische Funktion,

$$\text{akt}_m = \frac{1}{1 + e^{-\text{prop}_m}} \quad ,$$

die in den hier verwendeten Netzwerken ebenfalls zum Einsatz kommt.[4]

Als Ausgabefunktion kommen ebenfalls verschiedene Möglichkeiten in Betracht, hier wird allerdings die einfachste Möglichkeit, die Identität

$$\text{aus}_m = \text{akt}_m$$

verwendet.

Das Training der Netzwerke erfolgt mit einem überwachten Verfahren mit Hilfe des sogenannten *back-propagation*-Algorithmus. Das Netz zu trainieren bedeutet, die Gewichte der einzelnen Zellverbindungen so zu adaptieren, daß auf die vorgegebenen Eingangswerte der Trainingsmuster die zugehörigen Ausgangswerte möglichst gut approximiert werden. Zu diesem Zweck werden p Beispiele bereitgestellt, die die zu lösende Aufgabe möglichst gut repräsentieren. Für ein Netz mit μ Eingangs- und ν Ausgangsunits besteht ein Beispiel dabei aus einem Eingangsvektor $\mathbf{i}$, mit $\mathbf{i} \in \mathbb{R}^\mu$ und dem Vektor der zugehörigen Ausgangswerte $\mathbf{t}$, mit $\mathbf{t} \in \mathbb{R}^\nu$.

Die Eingangswerte werden durch das Netz propagiert. Dabei wird durch die Berechnungsvorschriften der einzelnen Zellen und die Gewichte der Verbindungen der Ergebnisvektor $\mathbf{o}$, mit $\mathbf{o} \in \mathbb{R}^\nu$ ermittelt. Die Differenz zwischen dem berechneten und dem erwarteten Ausgangsmuster $(\mathbf{o} - \mathbf{t})$ wird nun genutzt, um die Verbindungsgewichte so zu verändern, daß sich diese Differenz für das gerade präsentierte Muster

[4] Die Form der Aktivierungsfunktion ist hauptsächlich für das Training der neuronalen Netze von Bedeutung.

verringert. Die Gewichtsanpassung findet dabei entgegengesetzt zur üblichen Richtung der Informationsausbreitung statt, woher sich auch der Name *back-propagation* des Verfahrens ableitet. Die Gewichtsanpassung wird mit Hilfe eines generalisierten *delta error* Verfahrens [Köh90] vorgenommen. Hiermit wird bei der Anpassung der Gewichte berücksichtigt, wie stark die entsprechende Verbindung an dem jeweiligen Ergebnis beteiligt war. Je größer der Einfluß einer Verbindung, umso größer die entsprechende Adaption.

Bei der Anpassung der Gewichte handelt es sich um ein gradientenbasiertes minimieren des absoluten quadratischen Fehlers E. Bei einem Netz mit p Trainingspaaren und ν Ausgabeunits berechnet sich dieser mit

$$E = \frac{1}{2} \sum_{i=0}^{p} \sum_{j=0}^{\nu} ({}^{i}o_j - {}^{i}t_j)^2 \quad ,$$

wobei ${}^{i}o_j$ und ${}^{i}t_j$ dem j-ten Element des i-ten Ausgangs- bzw. Ergebnisvektors entspricht.

Wie bei allen gradientenbasierten Verfahren besteht auch hier die Möglichkeit, lediglich ein lokales Minimum in der Fehlerfunktion zu ermitteln. Da die Trainingsmuster in zufälliger Reihenfolge präsentiert werden, kann es sinnvoll sein, den Lernvorgang zu wiederholen, falls das Ergebnis nicht konvergiert. Da die Gewichtsanpassung jeweils nur auf der Basis eines einzelnen Beispiels erfolgt, ist der Verlauf des Fehlers E in der Regel nicht monoton fallend.

Durch die sogenannte Lernrate wird die Stärke aller Gewichtsanpassungen beim Lernen proportional beeinflußt. Nach großen Anpassungsraten zu Beginn des Lernvorgangs, wenn die von den Gewichten gebildete Konnektionsmatrix zufällig besetzt ist, wird durch deren Absenken die Änderungsgeschwindigkeit verringert. Dadurch wird bei fallendem Gesamtfehler die Anpassung der Gewichte schwächer, was Ausreißer abmindert. Zusätzlich kommt ein sogenannter Momentumterm zum Einsatz. Mit diesem Term wird durch die Berücksichtigung der Änderung des vorangegangenen Lernschrittes verhindert, daß eine Gewichtsanpassung die vorherige Modifikation direkt wieder aufhebt.

Das Training läuft so lange, bis entweder eine vorgegebene Fehlergrenze für E unterschritten oder die vorgegebene Anzahl von Trainingsschritten erreicht wurde. Über den Lernfehler kann relativ gut erkannt werden, ob die zu lernenden Werte eventuell nicht eindeutig abgebildet werden oder Unstetigkeiten enthalten. Dies äußert sich in der Regel in einer ungewöhnlich frühen Stagnation des Lernfehlers. Fehler in den Ausgangsdaten lassen sich mit diesem Mittel oft erkennen.

Ob ein Training zum Erfolg geführt hat, läßt sich insbesondere mit dem Vergleich des absoluten Lernfehlers E mit dem Generalisierungsfehler beurteilen. Der Generalisierungsfehler G wird analog zu E berechnet. Zur Ermittlung von G wird auf

einen Satz von q Generalisierungsmustern zurückgegriffen, die in der gleichen Art wie die Trainingsmuster erzeugt werden, aber Muster enthalten, deren Werte „zwischen“ denen der Lernmuster liegen. Die Anzahl der Generalisierungsdaten sollte dabei deutlich größer sein als die der Trainingsdaten.

$$G = \frac{1}{2} \sum_{i=0}^{q} \sum_{j=0}^{\nu} ({}^{i}o_j - {}^{i}t_j)^2 \quad .$$

Da in die beiden Werte die unterschiedliche Anzahl der Muster mit einfließt, bietet es sich an, auf die bezogenen Größen zurückzugreifen, um leichter vergleichen zu können.

$$\overline{E} = \frac{1}{p} E \quad \text{und} \quad \overline{G} = \frac{1}{q} G$$

Mit Hilfe des Generalisierungsfehlers läßt sich prinzipiell beurteilen, wie sich das Generalisierungsverhalten des Netzes im Vergleich zu den gelernten Werten darstellen wird. Lern- und Generalisierungsfehler sollten in den gleichen Größenordnungen liegen. Weichen diese Fehler stark voneinander ab, sollten die Netzparameter oder die zu lernenden Muster überarbeitet werden.

Am Ende des Trainingsvorganges ist das aus den präsentierten Trainingsmustern abgeleiteten „Wissen“ des neuronalen Netzes in den Gewichten der Zellverbindungen repräsentiert.

Als Implementierung künstlicher neuronaler Netze ist in dieser Arbeit die sogenannte NeuroLib eingesetzt worden. Diese in C++ programmierte Bibliothek ist ebenfalls im Fachgebiet Mechanik der Universität Duisburg entstanden und im Rahmen der Arbeit von Kleutges [Kle99] ausführlich vorgestellt worden.

Die steigende Rechenleistung bei PC's wirft nun die Frage auf, ob der Geschwindigkeitsgewinn durch die Verwendung neuronaler Netze immer noch notwendig oder zumindest sinnvoll ist. Eventuell wäre es möglich, die WalkingLib Bibliothek online zu nutzen. In diesem Fall würden die Muster direkt zur Laufzeit den Erfordernissen entsprechend erzeugt. Die vorherige Mustererzeugung und das Training der neuronalen Netze würden entfallen. Dabei soll auch überprüft werden, welche Genauigkeitsverluste durch den Netzeinsatz in Kauf genommen werden müßten. Dazu werden zuerst die Ein- und Ausgangsparameter der Netze näher vorgestellt, um danach die gewünschten Vergleiche anstellen zu können.

4.4 Neuronale Netze zur Erzeugung der Stellwinkel

Die im nächsten Kapitel vorgestellte Methode zur Bewegungskoordination verwendet ein neuronales Netzwerk je Bein zur Berechnung der Konfigurationen. Dadurch

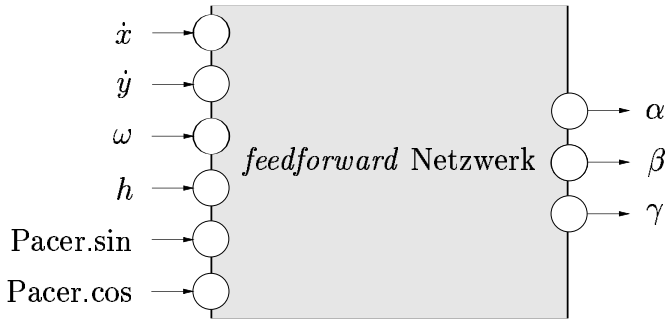


Abbildung 31: Ein- und Ausgabewerte der neuronalen Netze

ergeben sich bereits die Ausgangsgrößen, die den α-, β- und γ-Winkeln des zugehörigen Beines entsprechen. Die Netze werden mit periodischen Zyklen der verschiedenen Gangarten trainiert. Die Zahl der zur Unterscheidung verwendeten Eingangsparameter wird im Gegensatz zur WALKINGLIB deutlich reduziert. Während dort, wie in Tabelle 6 angegeben, sechzehn Parameter zum Einsatz kommen, finden als Eingang für die neuronalen Netzwerke, nur noch sechs Parameter Verwendung. Diese Eingangsgrößen sind neben den Geschwindigkeiten $\dot{x}$ und $\dot{y}$, der Rotationsgeschwindigkeit ω um die z-Achse und der Körperhöhe h_r das charakterisierende Wertepaar eines Schrittmachers. Dieser Schrittmacher (siehe Abschnitt 4.5) dient als Taktgeber, um die aktuelle Position im Schrittzyklus zu bestimmen. Ein entsprechendes Netz ist in Abbildung 31 angedeutet.

Die größte Zahl der Parameter der WALKINGLIB wird nur indirekt verwendet. Dadurch, daß die Bibliothek mit allen Parametern zur Berechnung der Trainingsdaten verwendet wird, fließt aber der komplette Satz in die trainierten Gangmuster mit ein. Werte wie Nick- und Rollwinkel oder die überlagerten Schwingungen kommen dabei allerdings nicht mehr als Veränderliche vor, sondern wirken nur als Konstanten in den Gangmustern.

4.5 Schrittmacher

Bei den trainierten Gangmustern handelt es sich um periodische Daten mit der Zykluszeit T. Um die aktuelle Position in diesem Zyklus als Eingang eines Netzwerkes zu verwenden, wird ein Schrittmacher als Taktgeber eingesetzt. Er ist in Form eines umlaufenden Zeigers implementiert, wie er in Abbildung 32 zu sehen ist. Um eindeutige Eingänge für das neuronale Netz zur Verfügung stellen zu können, wird das Paar der Sinus- und Cosinus-Werte des Zeigerwinkels verwendet.

Die Rotationgeschwindigkeit des Zeigers $\dot{\varphi}(t)$ beeinflußt die Zyklusdauer. Die Geschwindigkeiten, mit denen die neuronalen Netze trainiert werden, beziehen sich

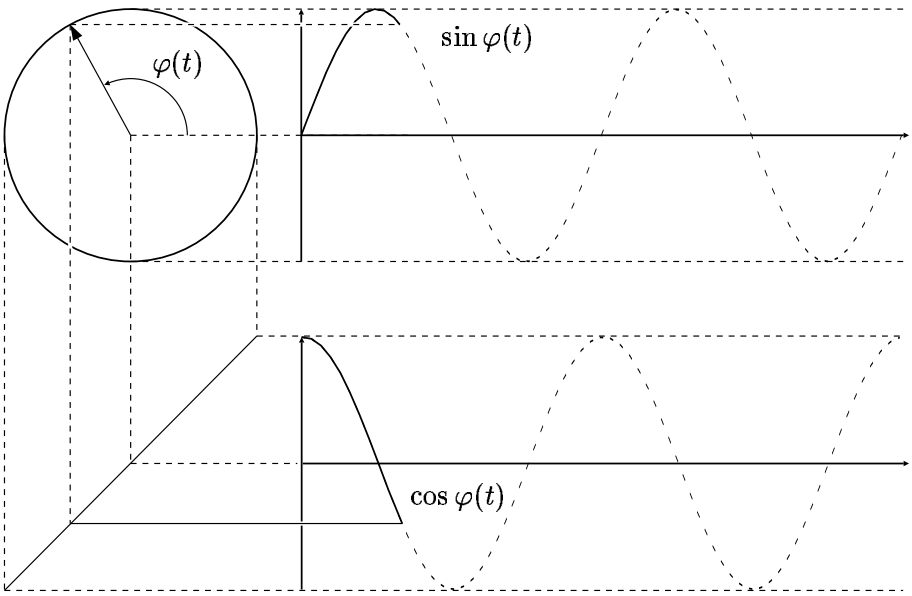


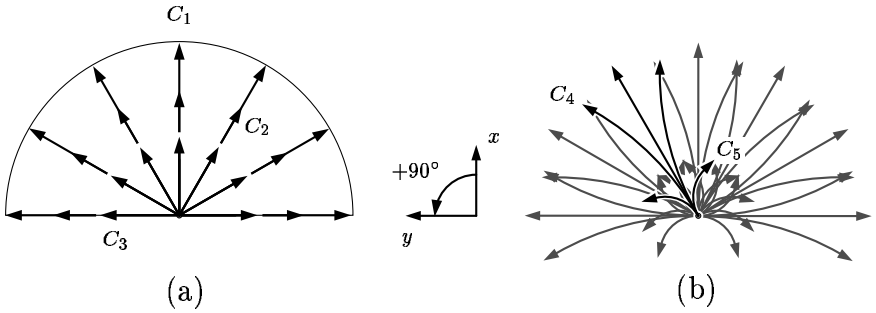
Abbildung 32: Graphische Darstellung eines Schrittmachers und der daraus abgeleiteten Eingangswerte für die neuronalen Netze

dabei auf eine feste Zykluszeit, mit der die Lernmuster erzeugt worden sind. Diese Zykluszeit wird üblicherweise zu $T = 2$ s gesetzt. Eine Umdrehung des Zeigers entspricht einem vollständigen Zyklus dieser Länge. Durch Beschleunigung oder Verlangsamung der Zeigergeschwindigkeit wird die Zykluszeit, und damit die Geschwindigkeit der Maschine, entsprechend skaliert. Die Schrittlängen der Trajektorien bleiben dabei erhalten.

4.6 Lern- und Generalisierungsmuster

Zum Training der Netzwerke werden Lernmuster zur Verfügung gestellt, die die Lösung der Aufgabenstellung repräsentieren. Günstig gewählte Muster resultieren in Netzwerken mit niedrigem Lern- und Generalisierungsfehler, ohne dabei unnötig umfangreich zu sein. Die Lernmuster werden mit der WALKINGLIB vorab berechnet. Die verwendeten Mustersätze entstehen aus den verschiedenen Vorgaben von möglichen Gangrichtungen, überlagerten Rotationen und Körperhöhen der Maschine.

Neben einem skalaren Geschwindigkeitswert, der für alle Mustersätze identisch ist, werden für die Richtungen, die Drehgeschwindigkeit und die Körperhöhe jeweils Minimal- und Maximalwerte sowie die Anzahl der Schritte vorgegeben. In Abbildung 33 sind die Werte, die sich aus den Parametern der Tabelle 7 ergeben angedeutet. Dabei enthält Abbildung 33(a) die sieben Richtungen mit je drei Geschwindigkeiten,



C_i	x	y	ω
1	1,0	0,0	0,0
2	0,6	-0,35	0,0
3	0,0	0,4	0,0
4	0,6	0,86	0,1
5	0,6	0,86	-0,2

Abbildung 33: Grafische Darstellungen der Muster aus Tabelle 7.

während in Abbildung 33(b) die überlagerten Rotationen bei der maximalen Geschwindigkeit angedeutet werden. Die in der Abbildung enthaltene Tabelle gibt die Parameter verschiedener Kurven in diesen Graphiken an. Aus den x- und y-Werten ergeben sich über einen zusätzlichen Faktor die Translationsgeschwindigkeiten der Gehmaschine.

Nach diesem Schema ist ein Lernmustersatz erstellt worden, dessen Parameter der Tabelle 8 entnommen werden können. Die Gesamtanzahl der enthaltenen Muster beläuft sich auf 375.

Um das Generalisierungsverhalten der trainierten Netzwerke beurteilen zu können, ist ein möglichst großer Datensatz erforderlich. Um diesem Anspruch gerecht zu werden, werden diese Muster mit den in Tabelle 9 enthaltenen Parametern gebildet und erzeugen insgesamt 38.400 Varianten.

Für jedes einzelne dieser Muster wird eine zyklische Trajektorie, analog zu der in Abbildung 26 dargestellten, erzeugt. Jede dieser Kurven wird aus 50 diskreten Punkten bzw. den zugehörigen Beinkonfigurationen gebildet. Damit ergeben sich für die Muster insgesamt 18.750, 36.750 bzw. 1.920.000 einzelne Ein- und Ausgangskombinationen.

Tabelle 7: Muster zur grafischen Darstellung in Abbildung 33

Parameter	min.	max.	Schritte
Geschwindigkeit[1]	0,4	1,0	3
Richtung[Grad]	-90	+90	7
OmegaDot[1]	-0,2	+0,2	5
Höhe[m]	0,15	0,15	1
Anzahl der Muster			105

4.7 Training und Genauigkeit der Netzwerke

Eine Eigenschaft der neuronalen Netzwerke ist, daß die gelernten Werte zwar sehr gut angenähert werden können, in der Regel aber nicht genau erreicht werden. Die Güte dieser Näherung ist von verschiedenen Faktoren abhängig. Netzgröße und -struktur bestimmen ebenso wie die günstige Auswahl der Trainingsdaten die Genauigkeit der Ausgabewerte. Besonders interessant sind dabei Reaktionen des Netzwerkes auf nicht gelernte Eingangswerte. Wie bereits erwähnt, kann dieses Verhalten in einer skalaren Form bereits über den bezogenen Lernfehler $\overline{E}$ und den Generalisierungsfehler $\overline{G}$ während des Trainings beobachtet werden.

Bei der Beurteilung der Genauigkeit der neuronalen Netzwerke sollte beachtet werden, daß sie bereits seit mehreren Jahren erfolgreich zur Erzeugung der Gelenkwinkel der Gehmaschine TARRY I eingesetzt worden sind. Dieses Wissen kann insbesondere dazu dienen, die hier ermittelten Fehlergrößen zu beurteilen, wobei auch die in Abschnitt 2.2.2 beschriebene Elastizität berücksichtigt werden sollte. Einige der ermittelten Positionierungsfehler wirken auf den ersten Blick relativ groß. Aber bereits ein Blick auf die zugrundeliegenden Winkelfehler zeigt, daß die Abweichungen in den Gelenken relativ gering sind und damit ein Großteil dieser entstehenden Fehlstellungen von der Elastizität des Mechanismus und dem Gelenkspiel aufgenommen werden kann.

Bisher sind die gelernten Netze und ihre Fehler nach insgesamt vier Kriterien beurteilt worden:

- Größe des absoluten Lernfehlers beim Training des neuronalen Netzes,
- Abweichung der trainierten Winkel von der Referenzvorgabe,
- Augenschein und
- Vergleich der gelaufenen Kurvenradien mit den erwarteten Radien.

Die bisherigen Beurteilungskriterien sollen überprüft und erweitert werden. Dabei soll das Augenmerk auf den beiden ersten, „theoretischen“ Methoden liegen. Die praktischen Methoden beurteilen neben der Güte der Ausgangsdaten zu einem Großteil die Eigenschaften der gesamten Versuchsanordnung. So können zum einen, wie bereits oben erwähnt, Fehler in den Mustern ausgeglichen werden, zum anderen werden durch die Maschine und deren Steuerung neue Fehler eingebracht.

Um zuerst eine günstige Netzwerkgröße zu ermitteln, wurden Netze verschiedener Größe mit den vorhandenen Lernmustern trainiert und gleichzeitig mit den Generalisierungsmustern überprüft. Für die Muster der Tabelle 8 ist der Verlauf der bezogenen Fehler eines 6-15-15-15-3 Netzes in Abbildung 34 aufgetragen. Der Lernfehler $\overline{E}$ verläuft gut, was auf prinzipiell korrekte Daten ohne Sprünge und mit

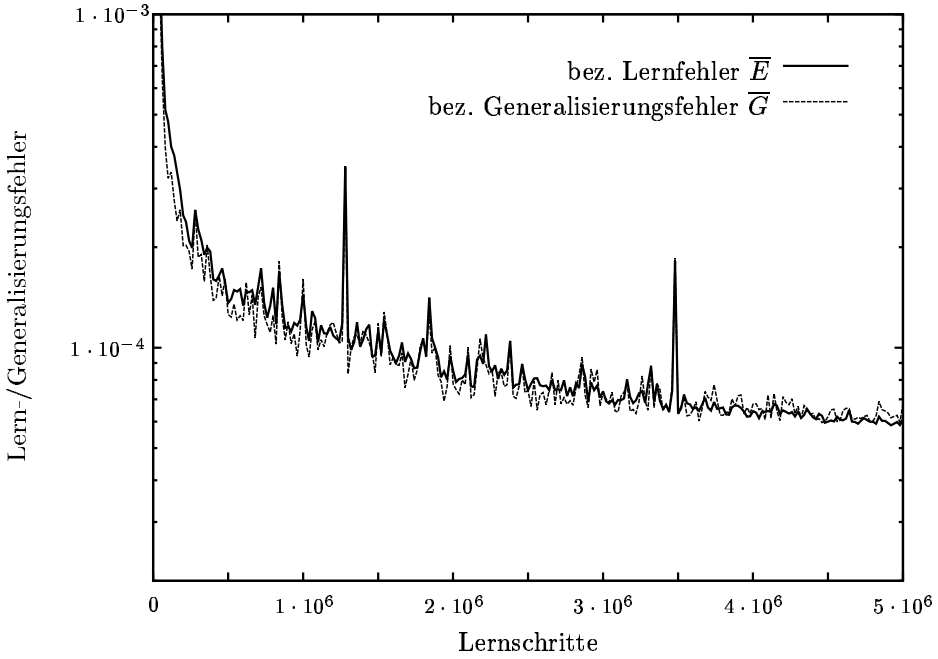


Abbildung 34: Bezogener Lern- und Generalisierungsfehler beim Training mit einem 6-15-15-15-3 Netzwerk

eindeutigen Mustern hinweist. Dazu verläuft der Lernfehler weitgehend parallel zu dem Generalisierungsfehler $\overline{G}$, der mit Hilfe der in Tabelle 9 angegebenen Daten ermittelt wird.

Um nun die Genauigkeit der Netze weiter zu steigern, ist es möglich, die Anzahl der Neuronen des Netzwerkes zu erhöhen. In Abbildung 35 ist der Lernvorgang mit den gleichen Lern- und Generalisierungsdaten für ein 6-30-30-30-3 Netz dargestellt worden. Dabei kann man erkennen, daß sich sowohl der Lern- als auch der Generalisierungsfehler verbessert haben. Allerdings ist gleichzeitig ein Auseinanderdriften

Tabelle 8: Lernmuster

Parameter	min.	max.	Schritte
Geschw.[1]	0,1	1,0	3
Richtg.[°]	-90	+90	5
Omega[1]	-0,2	+0,2	5
Höhe[m]	0,08	0,18	5
Anzahl der Muster			375

Tabelle 9: Generalisierungsmuster

Parameter	min.	max.	Schritte
Geschw.[1]	0,1	1,0	10
Richtg.[°]	-90	+90	20
Omega[1]	-0,2	+0,2	12
Höhe[m]	0,08	0,18	16
Anzahl der Muster			38.400

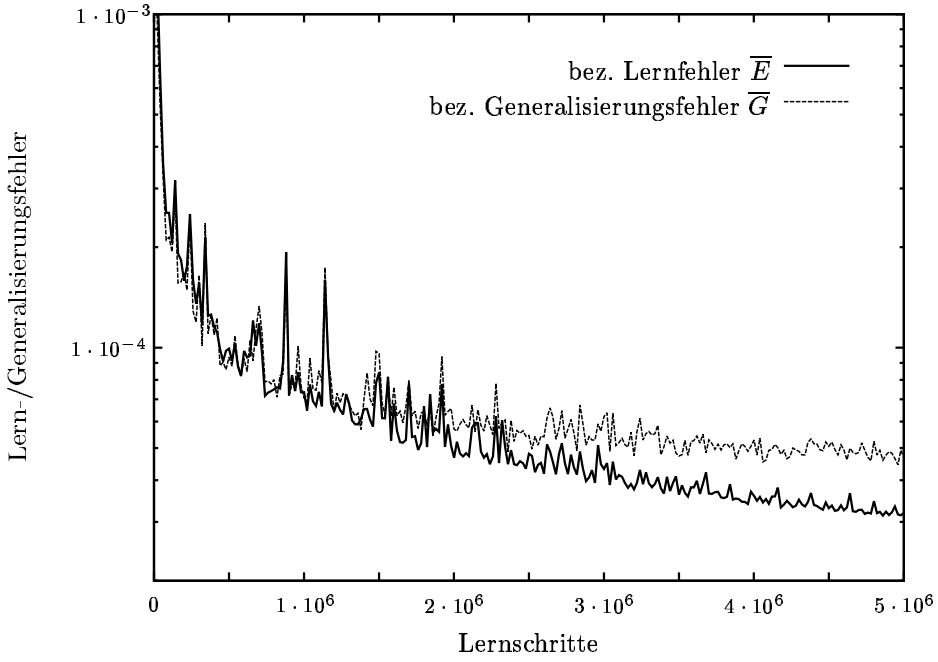


Abbildung 35: Bezogener Lern- und Generalisierungsfehler beim Training mit einem 6-30-30-30-3 Netzwerk

der beiden Kurven zu beobachten. Dies ist ein Effekt, der auch als „Auswendiglernen“ bezeichnet wird und bei dem das Netzwerk mit zunehmender Größe in der Lage ist, die Ein- und Ausgangskombinationen selbst zu lernen, woraufhin sich der Generalisierungsfehler nicht mehr in diesem Maße mitverbessert.

Das Auseinanderlaufen von $\overline{E}$ und $\overline{G}$ setzt bereits bei Netzen mit nur geringfügig größeren verdeckten Schichten ein. Daher wurde für die weiteren Experimente das kleinere der oben beschriebenen neuronalen Netze verwendet, mit dem eine ausgewogene Kombination von Genauigkeit und Rechenzeit erreicht wird.

Zur Untersuchung der Genauigkeit bietet es sich an, zuerst die Gelenkwinkel als gelernte Ausgangswerte einer näheren Betrachtung zu unterziehen. Sucht man in dem Ergebnis des in Abbildung 34 dargestellten Lernvorgangs nach dem Parametersatz, dessen Ausgabe die größte Abweichung $({}^io_j - {}^it_j)^2$ enthält, findet sich eine maximale Abweichung von 5,7° in den trainierten und von 6,7° in den generalisierten Mustern. Die analytischen und approximierten Winkelverläufe für dieses Generalisierungsmuster sind in Abbildung 36 dargestellt. Die maximale Abweichung liegt bei den Winkeln des β-Gelenkes vor.

Die reale Ausgabefunktion wird durch die Vorwärtskinematik der Maschine gebildet. Daher ist das eigentlich gewünschte Ergebnis die korrekte Positionierung des Fußes. Da die Winkel und Winkelfehler je nach Gelenk und Konfiguration unterschiedlich

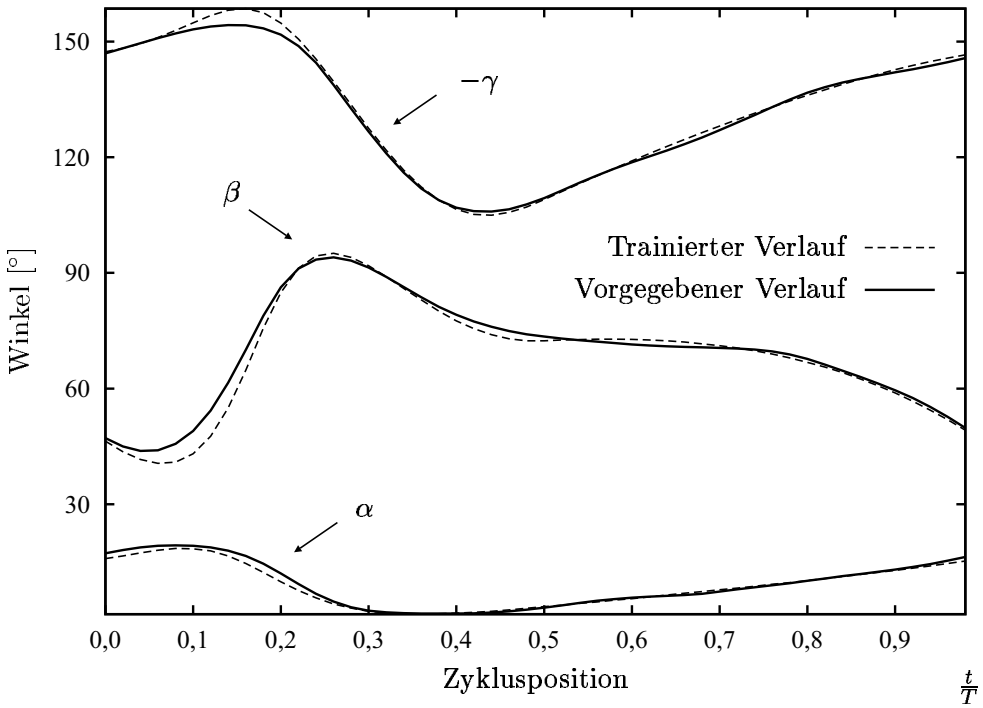


Abbildung 36: Gelernte und vorgegebene Winkelverläufe für das Muster mit dem größten Winkelfehler

in die Position des *Tarsus* eingehen, ist es sinnvoll, statt der Winkel die Soll- und Istwerte der Fußposition zu vergleichen.

Dazu werden für die Lern- und Generalisierungsdaten die kompletten Zyklen der Gelenkwinkel eines Beines sowohl analytisch als auch durch das neuronale Netz berechnet. Über die Vorwärtskinematik, die von der WALKINGLIB mit dem Modell von TARRY II zur Verfügung steht, wird die Differenz der Fußpositionen bestimmt, die sich aus den beiden Konfigurationen ergeben. Für jeden Zyklus wird die maximale Länge dieses Differenzvektors festgehalten. In Abbildung 37 ist die Verteilung dieses Fehlers in aufsteigender Sortierung dargestellt. Als größte Abweichung kann dabei ein Abstand von 0,0136 m festgestellt werden.

Aus der nicht abbgebildeten unsortierten Verteilung kann ein Zusammenhang zwischen den Differenzen und den Eingangsparametern erkannt werden. Durch die Analyse dieses Zusammenhangs ließe sich die Zahl der Lernmuster optimieren, um auch mit wachsender Netzgröße genauere Ergebnisse zu erzielen, bei denen Lern- und Generalisierungsfehler ähnlich verläuft.

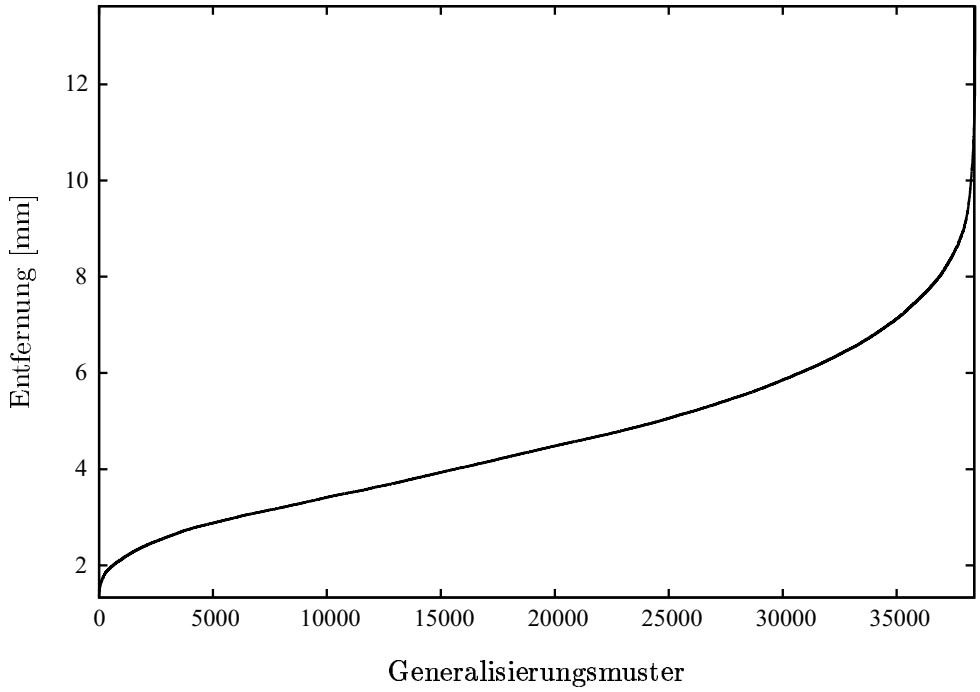


Abbildung 37: Verteilung des maximalen kartesischer Positionierungsfehler eines Beines mit den Mustern der Generalisierungsdaten

4.8 Geschwindigkeitsvergleich

Das Kriterium, das für einen hier verwendeten Gangmustererzeuger erfüllt sein muß, ist seine ausreichende Geschwindigkeit. Um eine flüssige Bewegung zu ermöglichen, darf die Berechnungsdauer einen bestimmten Wert nicht überschreiten. Diese maximale Dauer ist abhängig von der Frequenz der Ansteuerung und ihrem möglichen Anteil am Gesamttakt.

Der bisher vorgegebene Takt liegt bei ca. 50 Hz. Um eine noch flüssigere Bewegung zu ermöglichen, wird für die Zukunft eine Erhöhung dieses Taktes auf ca. 100 Hz angestrebt. Für jeden Takt steht also eine Rechenzeit von max. 20 ms, für die beschleunigte Version bis herunter auf 10 ms, zur Verfügung. Die Gangerzeugung sollte hierbei einen Anteil von weniger als 33 % haben. Damit ergibt sich eine angestrebte Dauer von maximal 3,3 ms.

Bei dem Vergleich muß beachtet werden, daß die Qualität der von der WALKINGLIB berechneten Gangmuster gleichbleibend ist. Die von den neuronalen Netzen ausgegebenen Werte weisen in ihrer Genauigkeit durch ihre Abhängigkeit von der Netzgröße indirekt auch eine Abhängigkeit von der aufgewendeten Rechenzeit auf. Aus diesem Grund werden hier die Rechenzeiten der WALKINGLIB mit den Rechenzeiten und den Genauigkeiten neuronaler Netze verschiedener Größe gegenübergestellt.

Tabelle 10: Vergleich der Rechenzeiten

Methode	Zeit[ms]	Faktor
WALKINGLIB	4,46	1,0
NN 6-09-09-09-3	0,51	8,7
NN 6-15-15-15-3	0,73	6,1
NN 6-22-22-22-3	1,08	4,1
NN 6-30-30-30-3	1,59	2,8

Um den Zeitaufwand zu vergleichen, werden für ein einzelnes Bein die 38.400 Generalisierungsmuster aus Tabelle 9 mit je 50 diskreten Beinkonfigurationen berechnet und die dazu notwendige Zeit t_{ges} gemessen.

Als Vergleichswert ist die durchschnittliche Zeit zur Berechnung von sechs Beinkonfigurationen, also der notwendigen Stelldaten für einen Taktschritt, herangezogen worden[5]. Dabei kann davon ausgegangen werden, daß sowohl bei der WALKINGLIB als auch bei den neuronalen Netzen bei der Berechnung einer einzelnen Konfiguration kaum Abweichungen von der durchschnittlichen Rechenzeit t_i auftreten.

Es gilt $t_i = \frac{t_{ges}}{38.400 \cdot 50}$ oder für alle Beine $t_{\text{Taktschritt}} = \frac{6\, t_{ges}}{38.400 \cdot 50} = \frac{t_{ges}}{320.000}$.

Aus den in Tabelle 10 aufgetragenen Rechenzeiten wird ersichtlich, daß die analytischen Berechnungen in den Bereichen der verwendeten Netzwerkgröße um ein Vielfaches langsamer sind. Da deren Dauer die festgelegte Grenze überschreitet, ist es sinnvoll, die bisher verwendete neuronale Ansteuerung weiter zu nutzen, insbesondere da sich deren Ungenauigkeiten bereits als tragbar erwiesen haben. Aufgrund der ausreichenden Genauigkeit der Approximation sollte der relative Geschwindigkeitsvorteil der neuronalen Netze auch weiter genutzt werden, selbst wenn die Rechenzeit der WALKINGLIB durch eine Verbesserung der Hardware den geforderten Wert unterschreiten könnte.

[5] Alle angegebenen Rechenzeiten sind mit dem TARRY Kontrollrechner ermittelt worden. Dieser Rechner ist ausgestattet mit einem Intel Pentium Pro Prozessor mit 200 MHz und einem Hauptspeicher von 80 MB.