Evolution und Robotik

Teilnehmer:

David Schwalb	Herder-Oberschule
Elmo Feiten	Herder-Oberschule
Johannes Dyck	Heinrich-Hertz-Oberschule
Konrad Seifert	Heinrich-Hertz-Oberschule
Elias Pipping	Herder-Oberschule
Hans Dettmar	Herder-Oberschule
Benjamin Schneider	Herder-Oberschule

Gruppenleiter:

Manfred Hild	Humboldt-Universität zu Berlin

Die Evolutionsmechanismen der Natur (natürliche Auslese, Kreuzung und Mutation) lassen sich auch in der Robotik verwenden und können mobile autonome Roboter mit interessanten Verhaltensweisen hervorbringen.

Die Kursteilnehmer beschäftigten sich mit der Funktionsweise von künstlicher Evolution und der Dynamik rekurrenter, neuronaler Netze – das heißt deren Attraktoren und Bifurkationen. In einer Simulationsumgebung evolvierten sie Roboter mit verschiedenen Verhaltensweisen (z.B. Hindernisvermeidung und Tropismen) und analysierten die Netze einzelner Individuen.

Zum Schluss des Kurses übertrugen sie die Netze auf einen realen Roboter und verglichen jeweils dessen Verhaltensweise mit ihren theoretischen Überlegungen.

Evolution und Robotik

1 Einführung

Ursprünglich wurden Roboter designt, um Arbeiten zu verrichten, die Menschen nicht übernehmen können oder um menschliche Arbeitskräfte zu ersetzen. Sie sind meist hoch spezialisiert und besitzen besondere Eigenschaften, wie sehr viel Kraft oder eine große Genauigkeit. Dafür sind sie jedoch sehr unflexibel und können nur zu dem ihnen zugedachten Zweck eingesetzt werden.

Im Gegensatz dazu sind die im weiteren Verlauf betrachteten Roboter autonom, d.h. sie sind selbstständig und besitzen eine eigene Energiequelle, Sensorik, Motorik und sind intelligent. Aufgrund technischer Einschränkungen verfügen solche Maschinen jedoch nur über die weniger komplexen Aspekte der Intelligenz, sind aber dennoch innerhalb gewisser Grenzen in der Lage, einzuschätzen, Probleme zu lösen, zu reagieren, zu planen und zu urteilen. Höhere Formen der Intelligenz, wie der „gesunde Menschenverstand“, Kreativität und ästhetisches Empfinden bleiben ihnen jedoch bislang verschlossen.

Es gibt zwei grundlegende Ansätze, intelligentes Verhalten zu simulieren. Im reaktiven Modell besitzt der Roboter keine Orientierung und kennt seine Umgebung nicht. Er reagiert spontan auf den aktuellen Input und betreibt keine langfristige Planung. Deliberativ aufgebaute Roboter besitzen ein Weltmodell, planen voraus, benötigen dafür aber viel Zeit und sind im Vergleich zum reaktiven Modell träge. Beide Ansätze haben ihre Vor- und Nachteile, somit ist ein Kompromiss zwischen beiden die beste Lösung. Da es sehr schwer

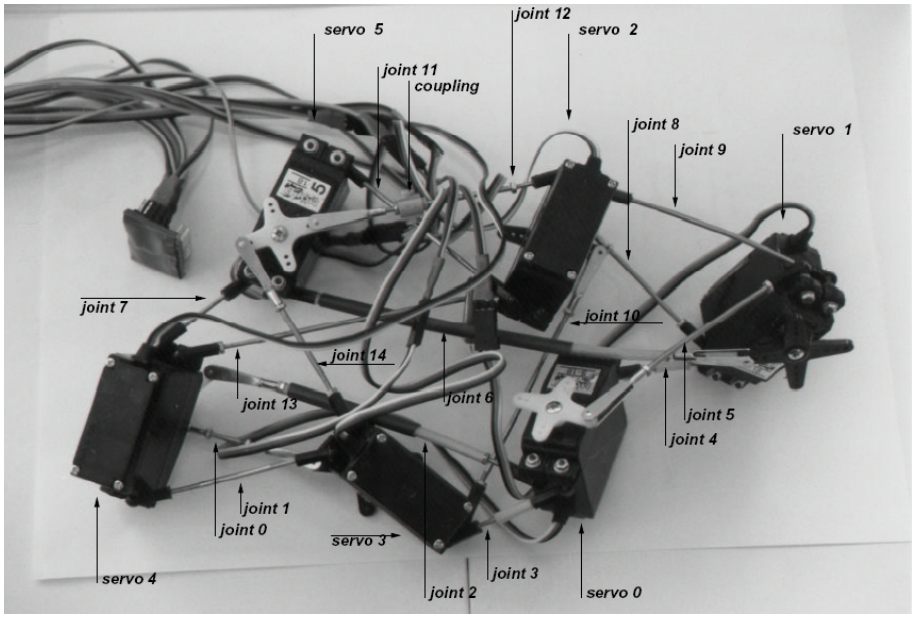

Abbildung 1.1: Hier sind sechs miteinander verbunden Servomotoren zu sehen, welche sich allein durch die Motorenbewegungen fortbewegen sollen. Durch die künstliche Evolution lernt dieses Gebilde bereits nach kurzer Zeit zu „laufen“.

ist, das Verhalten solcher intelligenten, autonomen Roboter explizit zu programmieren,

orientiert man sich an der Natur und macht sich das Prinzip der Evolution zu Nutze. Es gibt beispielsweise Aufgaben, die so komplex sind, dass es nahezu unmöglich ist, sie durch gezielte Überlegungen programmiertechnisch zu lösen (siehe Abbildung 1.1).

2 Künstliche Evolution

Um solche Roboter herzustellen, bedient man sich einer Simulation des natürlichen Evolutionsprozesses, basierend auf Selektion, Mutation und Rekombination. Hierbei steht das Verhalten der Roboter für den Phänotypen in der Biologie und der Vektor, welcher das Verhalten definiert, findet seine Entsprechung in dem Genotypen. Konkret wird jedem Roboter ein sogenannter „Fitness-Wert“ zugeordnet, der seine Fähigkeit widerspiegelt, seine Aufgabe effektiv zu erledigen. Anhand dieses Wertes wird bestimmt, welche Genotypen in die nächste Generation übernommen werden - diejenigen mit den niedrigsten Fitness-Werten fallen der Selektion zum Opfer.

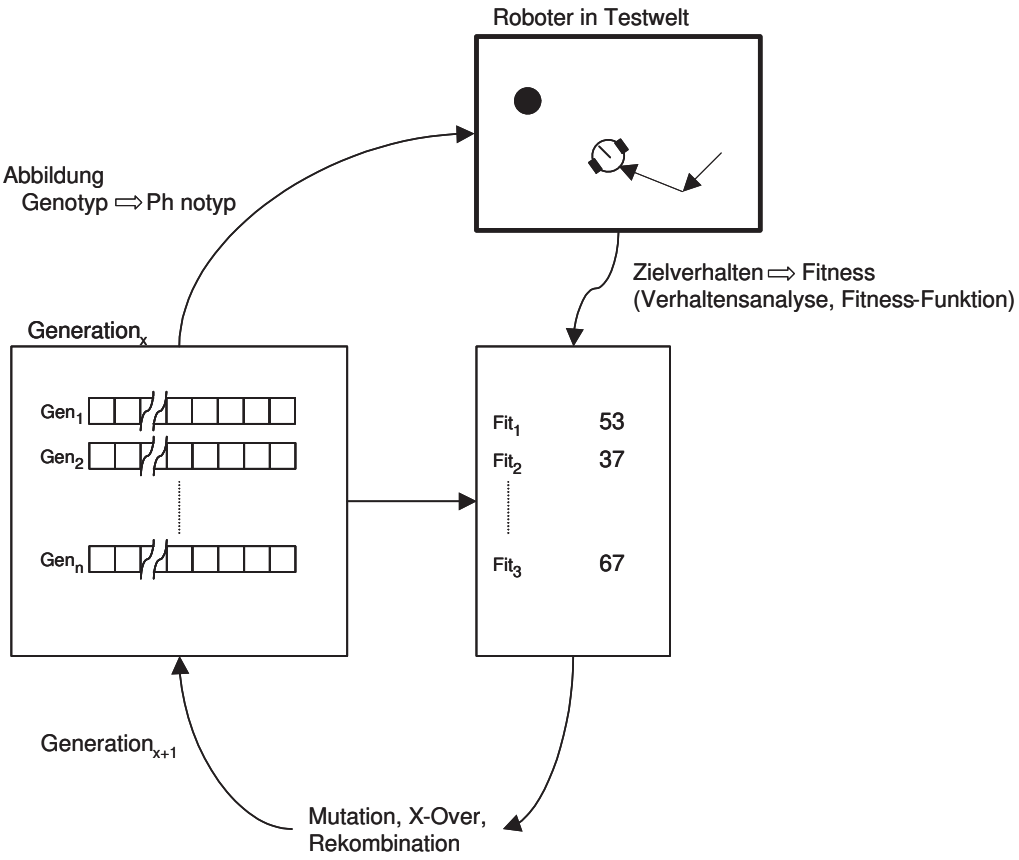

Abbildung 2.2: Künstliche Evolution

Rekombination bedeutet, dass zwei Genotypen an derselben Stelle zertrennt und die hinteren Teile vertauscht werden. Durch Mutation werden schließlich einige der Genotypen zufällig an einer Stelle verändert, bevor die neue Generation von Robotern erschaffen wird.

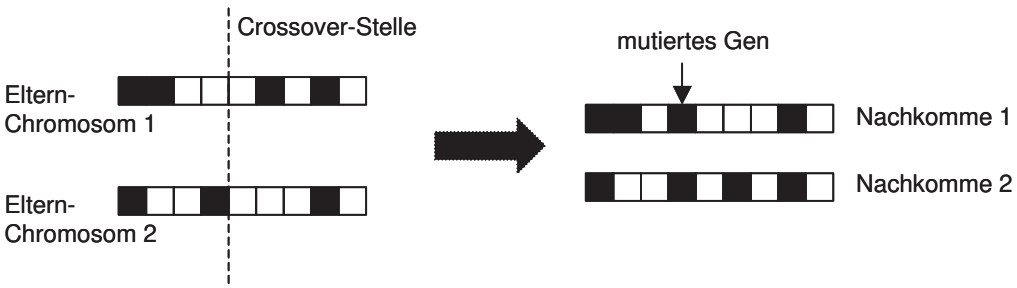

Abbildung 2.3: Rekombination und Mutation

3 Neuronale Netze

Ein neuronales Netz (wie beispielsweise auch das menschliche Gehirn) besteht aus mehreren Neuronen. Ein Neuron wiederum fungiert als signalverarbeitendes Element, das von außen oder von anderen Neuronen Signale empfängt und sie verarbeitet bzw. weiterleitet.

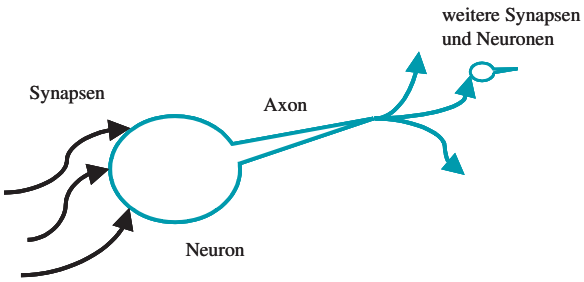

Abbildung 3.4: Schematische Darstellung eines Neurons

Die Funktionsweise ist wie folgt vorstellbar: Ein Neuron wird von einer oder mehreren Synapsen (die wiederum von anderen Neuronen kommen) umlagert. Durch diese Synapsen können elektrische Impulse an das Neuron weitergeleitet werden, die erregend oder hemmend wirken. Wird das Neuron ausreichend erregt, so „feuert“ es, das heißt, es sendet seinerseits einen elektrischen Impuls, der durch sein Axon geht und sich dann auf die Synapsen verteilt, die den Imupls wiederum an weitere Neuronen weiterleiten können.

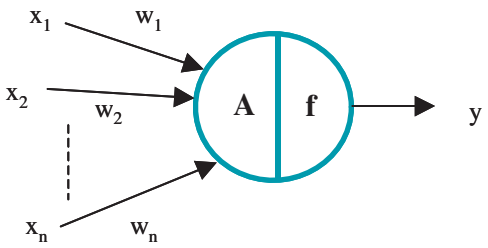

Abbildung 3.5: Mathematisches Modell eines Neurons

Um sich das Prinzip eines neuronalen Netzes für die künstliche Evolution zu Nutze zu machen, benutzt man ein abstraktes Modell-Neuron. Das Bild zeigt dieses Modell-Neuron, das n x-Werte als Input bekommt $(x_1, x_2, \ldots, x_n)$, $x_n \in (-1, +1)$. Jeder dieser Werte verfügt über eine Gewichtung $(w_1, w_2, \ldots, w_n)$. Hierbei benötigt man lediglich Werte in kleineren Bereichen, so also zum Beispiel $w_i \in [-10, 10]$. Aus diesen Werten berechnet sich die sogenannte Aktivität **A** des Neurons, nämlich durch:

$$A := \sum_{i=1}^{n} x_i w_i \tag{1}$$

Die Ermittlung des Output-Wertes, der weitergegeben wird, findet durch eine geeignete Funktion f statt, also: $y := f(A)$.

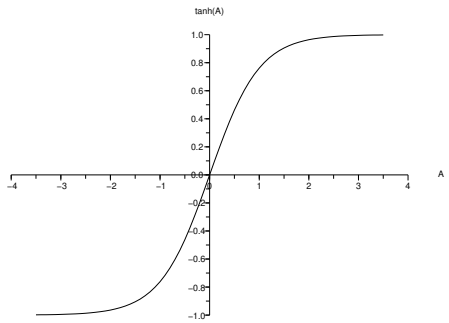

Abbildung 3.6: Graph des Tangens Hyperbolicus

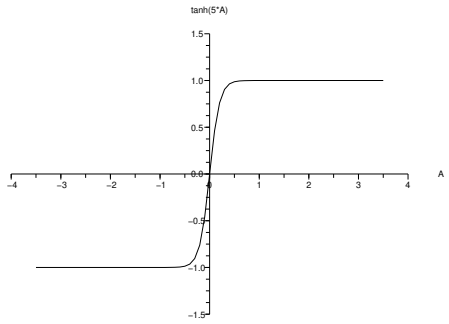

Abbildung 3.7: Graph des 5-fach in A-Achse gestauchten Tangens Hyperbolicus

Hierfür eignet sich besonders die oben abgebildete Funktion Tangens Hyperbolicus (tanh), da sie sehr vielseitig einsetzbar ist. Durch Änderung der Gewichtungen $w_1, w_2, \ldots, w_n$ kann die Funktion stark verändert werden. Wählt man ein hohes w, wird die Funktion soweit gestaucht, dass sie im Extremfall zu einer Sprungfunktion wird, die der Signum-Funktion ähnelt. Wählt man das w dagegen sehr klein, so entsteht im Prinzip eine Null-Funktion. Zusätzlich hat die tanh-Funktion in einer sehr kleinen x-Umgebung um 0 einen nahezu linearen Verlauf.

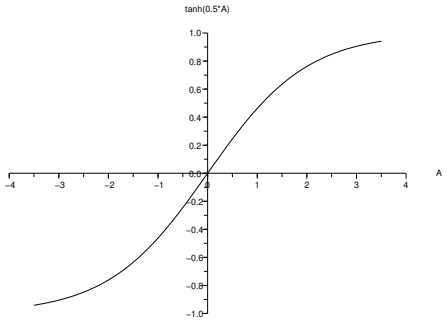

Abbildung 3.8: Graph des 2-fach in A-Achse gestreckten Tangens Hyperbolicus

Somit erhält man ein Neuronenmodell, das wie folgt aussieht:

$$t \in \mathbb{N} \qquad \text{die Zeit} \tag{2}$$

$$x_i(t) \in (-1;1) \qquad \text{Signal des Neurons } x_i \text{ zum Zeitpunkt } t \tag{3}$$

$$w_{ij} \in \mathbb{R} \qquad \text{Verbindungsgewicht von Neuron j zu Neuron i} \tag{4}$$

Es gibt Input-Neuronen (Sensorsignale), darunter optional ein Bias-Neuron (mit konstantem Wert 1) und Output-Neuronen (Motorsignale), sowie Hidden-Neuronen (alle anderen).

$$n \qquad \text{Anzahl aller Neuronen} \tag{5}$$

$$m \qquad m < n \text{ Anzahl der Input-Neuronen (inkl. Bias)} \tag{6}$$

$$x_i(t+1) := \tanh\left(\sum_{j=1}^{n} w_{ij} x_j(t)\right), \qquad i = (m+1), \ldots, n. \tag{7}$$

Zwar besteht das menschliche Gehirn, wie bereits erwähnt, aus mehreren hundert Millionen Neuronen, jedoch kann man schon mit ein bis zwei Neuronen viel anfangen.

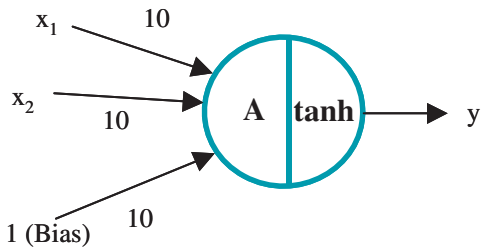

Abbildung 3.9: Neuronale Realisierung der boolschen „Oder“-Funktion

Wählt man (wie im obigen Bild) beispielsweise die Gewichtungen der zwei Input-Werte sehr hoch, so wird die tanh-Funktion derart gestaucht, dass sie im Prinzip zu einer Sprungfunktion wird. Zusätzlich existiert in diesem speziellen Anwendungsfall noch ein sogenanntes

Bias-Neuron, das unabhängig von äußeren Werten stets den Wert 1 an das Neuron weiterliefert und über eine Gewichtung von 10 verfügt. Es ergibt sich:

$$A = 10x_1 + 10x_2 + 10 \quad (8)$$
$$y = f(A) = \tanh(10x_1 + 10x_2 + 10) \quad (9)$$

Stellt man sich nun vor, dass die Input-Werte nur -1 oder +1 annehmen können, ergeben sich vier Möglichkeiten:

x_1	x_2	y
-1	-1	$\tanh(-10-10+10) = \tanh(-10) = -1$
-1	+1	$\tanh(-10+10+10) = \tanh(10) = +1$
+1	-1	$\tanh(10-10+10) = \tanh(10) = +1$
+1	+1	$\tanh(10+10+10) = \tanh(30) = +1$

Tabelle 1: Wertetabelle: logisches „Oder“

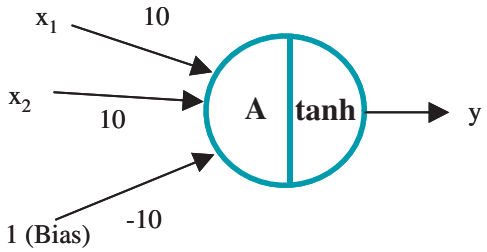

Abbildung 3.10: Neuronale Realisierung der boolschen „Und“-Funktion

Ebenso ist selbstverständlich auch eine Schaltung für das logische „Und“ möglich. Dafür wird lediglich die Gewichtung des Bias verändert und zwar zu -10. Es ergibt sich folgendes:

$$A = 10x_1 + 10x_2 - 10 \quad (10)$$
$$y = f(A) = \tanh(10x_1 + 10x_2 - 10) \quad (11)$$

x_1	x_2	y
-1	-1	$\tanh(-10-10-10) = \tanh(-30) = -1$
-1	+1	$\tanh(-10+10-10) = \tanh(-10) = -1$
+1	-1	$\tanh(10-10-10) = \tanh(-10) = -1$
+1	+1	$\tanh(10+10-10) = \tanh(10) = +1$

Tabelle 2: Wertetabelle: logisches „Und“

Um diese einfachen logischen Schaltungen darzustellen, benötigt man also tatsächlich nur ein Neuron. Die Darstellung des Xor (exklusives Oder) ist zwar ein wenig aufwändiger,

aber genauso möglich. Außerdem ist durch die Kombination zweier Neuronen noch der Aufbau von Hochpassfiltern, Tiefpassfiltern, Verstärkern, An-/Ausschaltern und weiteren Funktionen möglich.

Trotzdem stellt sich natürlich die Frage, welchen konkreten Vorteil neuronale Netze bieten. Ebenso wie die künstliche Evolution ist auch das Prinzip eines neuronalen Netzes der Natur entnommen, da es offensichtlich funktioniert. Außerdem ist ein neuronales Netz sehr einfach zu evolvieren, da nur wenige Parameter existieren, nämlich die Gewichtungen $w_1, w_2, \ldots, w_n$. Tatsächlich enthalten die Genotypen als Informationen lediglich die Gewichtungen der einzelnen Verbindungen.

4 Die Fitnessfunktion

Wenn man eine bestimmte Anzahl von Individuen künstlich evolviert, so bedient man sich einer sogenannten Fitnessfunktion, welche die Leistung der einzelnen Individuen bestimmt und somit hilft, die besten bzw. leistungsfähigsten einer Generation zu selektieren und in die nächste zu übertragen. Das Resultat einer künstlichen Evolution hängt sehr stark von der Form dieser Fitnessfunktion ab, welche sich mit Hilfe der Abbildung 4.11 beschreiben lässt. Die Kategorien lassen sich hierbei wie folgt gegeneinander abgrenzen:

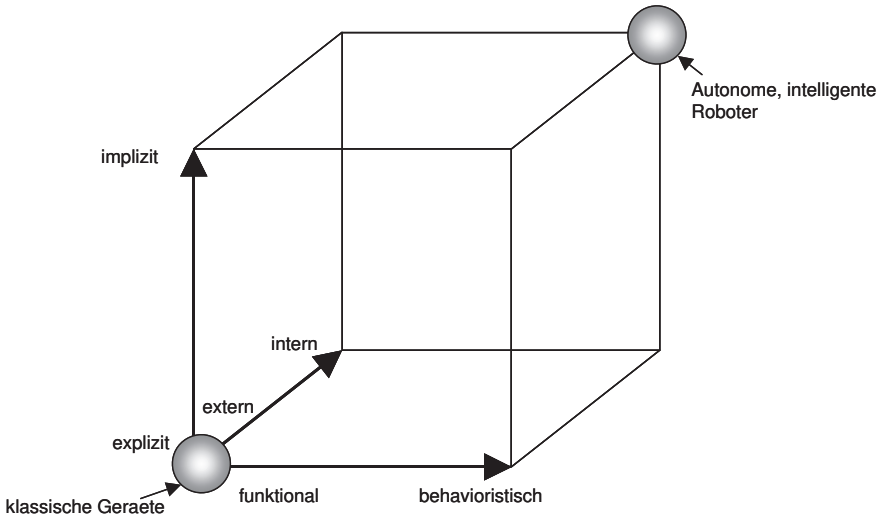

Abbildung 4.11: Implementationsmöglichkeiten der Fitnessfunktion

- Intern: die Fitnesswerte werden direkt aus den neuronalen Signalen berechnet.
- Extern: die Fitnesswerte werden mit Hilfe der Beurteilung eines externen Beobachters erstellt.
- Behavioristisch: allein das Resultat ist für den Fitnesswert ausschlaggebend (wenn die Aufgabe also lautet, von Punkt A nach Punkt B zu gelangen, so ist die Art und Weise der Fortbewegung nicht von Interesse; ausschlaggebend ist die benötigte Zeit).

- Funktional: die konkrete Funktionsweise wird bewertet; z.B. wird vorgeschrieben, mit welchen Bewegungen der Roboter von A nach B gelangen soll.
- Implizit: Anzahl von Variablen, Konstanten und Randbedingungen ist gering (in unserem Beispiel wäre nur die Tatsache, dass der Roboter geradeaus fährt von Bedeutung, der Abstand eines Sensors von der Wand z.B. nicht).
- Explizit: Anzahl von Variablen, Konstanten und Randbedingungen ist hoch.

Beispiel einer Fitnessfunktion

Eine Fitnessfunktion könnte man wie folgt realisieren: Die Länge des Zeitraums, in dem der Roboter geradeaus fährt, minus die Länge des Zeitraums, in dem er Kurven fährt (wobei die Länge der Kurve ebenfalls von Bedeutung ist), plus die Entfernung des Roboters von der vor ihm liegenden Wand. Je länger der Roboter also geradeaus fährt, desto höher ist der Fitnesswert des entsprechenden Individuums. Wenn wir den Abstand der Sensoren von der Wand mit Hilfe von neuen Variablen, die wir I_1 und I_2 nennen, angeben, so könnte die Fitnessfunktion auch folgendermaßen lauten:

$$f(x) = \frac{O_1 + O_2}{2} - \frac{|O_1 - O_2|}{2} - \frac{I_1 + I_2 + 2}{4}, \tag{12}$$

wobei

$$O_1 := \text{Output, der den linken Motor ansteuert,} \tag{13}$$
$$O_2 := \text{Output, der den rechten Motor ansteuert.} \tag{14}$$

Hierbei sind $I_1, I_2 \in [-1, +1]$, wobei -1 bedeutet, dass der Roboter einen großen Abstand von der vor ihm liegenden Wand hat und $+1$, dass er die vor ihm liegende Wand berührt. Diese Fitnessfunktion ist ein gutes Beispiel für eine explizite, interne und funktionale Funktion.

5 Praktisches Beispiel

Als praktische Aufgabe wollten wir einen Roboter evolvieren, welcher sich mit drei IR-Abstands-Sensoren in einer realen Welt zurecht findet. Mittels zweier Motoren, die von zwei Output-Neuronen angesteuert werden, kann er sich frei in seiner Welt bewegen und Hindernissen ausweichen. Das Experiment umfasste die folgenden Teilschritte:

1. In einem Seminarraum wurde die Welt aus drei Tischen und einem Teil der Wand aufgebaut.
2. Die virtuelle Welt wurde als Weltpolygonzug per x-y-Koordinaten in den PC eingegeben. Hierzu wurde der World-Simulator benutzt.
3. Im Robot-Simulator wurden die Maße des Roboters als x-y-Koordinaten eingegeben, wobei der Ursprung dem Drehpunkt entsprach.

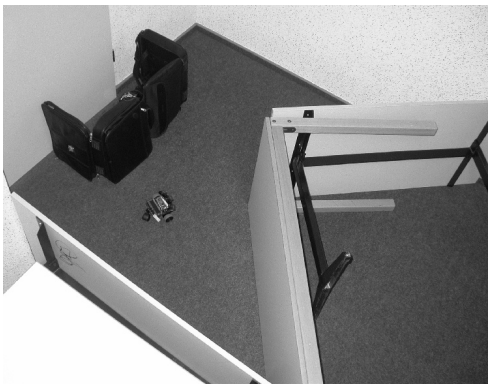

Abbildung 5.12: Aufgebaute Welt.

4. Durch die Verknüpfung der Programme World-Simulator, Robot-Simulator und Populations-Manager konnte die künstliche Evolution mit dem virtuellen Roboter in der virtuellen Welt durchgeführt werden.

5. Das evolvierte neuronale Netz wurde auf den realen Roboter übertragen.

6. Der Roboter wurde nun in die Testumgebung gesetzt und gestartet.

Das Resultat des Experiments: Der Roboter hat in der realen Welt exploratives Verhalten gezeigt, das heißt, er fuhr in der Welt einen großen Bereich ab. Dabei stieß er nicht an Hindernisse und fand erfolgreich den Weg aus spitzen Ecken und Sackgassen.

Wie sich zeigte, konnte der Roboter Tischbeinen nicht ausweichen, da derart dünne Gegenstände nicht in der Testwelt vorhanden waren, in welcher er evolviert wurde.

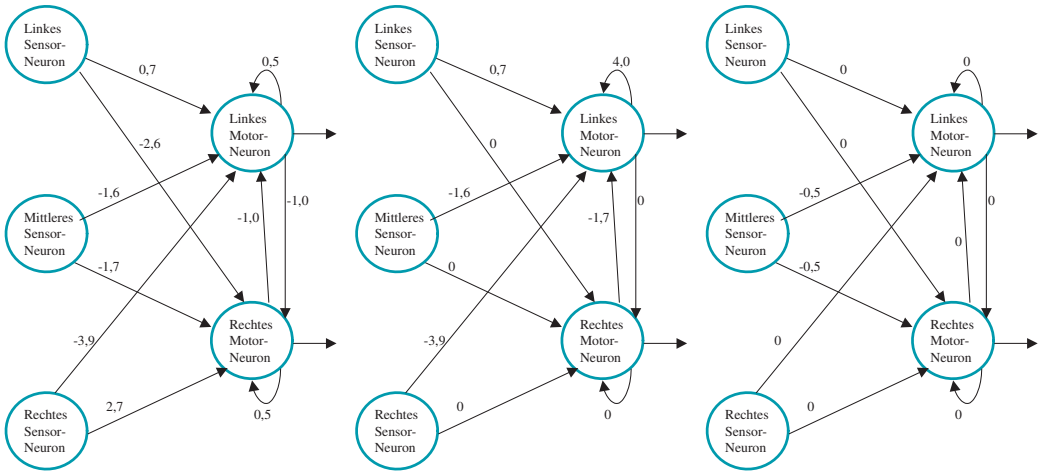

Abbildung 5.13: Modifizierte Kopien des aus der Evolution entnommenen Netzes

Um die Auswirkungen eines veränderten Genotyps auf das Verhalten des Roboters zu untersuchen, werden zum Schluss verschiedene neuronale Netze auf dem Roboter ausprobiert, die durch gezielte Modifikation aus dem ursprünglichen Netz abgeleitet wurden. In Abbildung 5.13 sind drei dieser Netze zu sehen.

Verhalten des Roboters mit dem ganz links abgebildeten Netz: Das Verhalten des Roboters veränderte sich dahingehend, dass er bei frontalem Wandkontakt nach links anstatt nach rechts auswich. Er konnte sich aus einer spitzwinkligen Ecke nicht mehr befreien.

Verhalten des Roboters mit dem in der Mitte abgebildeten Netz: Der Roboter rotierte um sein rechtes Rad gegen den Uhrzeigersinn. Sobald der rechte Sensor dabei ein Hinderniss erfasste, wechselte der Roboter die Drehrichtung.

Verhalten des Roboters mit dem ganz rechts abgebildeten Netz: War kein Hindernis in Sicht, fuhr der Roboter geradeaus. War ein Gegenstand weniger als 10cm von ihm entfernt, dann blieb er stehen. Bei weiterer Annäherung wich er entsprechend zurück - entfernte sich das Objekt, so verfolgte er es.

Insgesamt konnte gezeigt werden, dass die künstliche Evolution neuronaler Netze geeignet ist, einen mobilen Roboter mit einer gewünschten Verhaltensweise auszustatten – sowohl in der Simulation, wie auch in der realen Welt.

Abbildung 5.14: Die Kursteilnehmer mit ihrem Roboter