Abschlussarbeit am Institut für Informatik der Freien Universität Berlin zur Erlangung des akademischen Grades Bachelor of Science (B.Sc.).

Arbeitsgruppe Intelligent Systems and Robotics

Neurorover: Künstliche Mini-Gehirne für die Kontrolle von Robotern

Oliver Gerhard Hanßen

Matrikelnummer: 4767250

oliver.hanssen@fu-berlin.de

Betreuer: Prof. Dr. Tim Landgraf

Abgabefrist: 29.11.2016

Berlin, 29.11.2016

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Es wurden keine anderen als die angegebenen Quellen und Hinweise verwendet.

Die vorliegende Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Berlin, der 29. November 2016

Oliver Gerhard Hanßen

Zusammenfassung

Die Neurorobotik beschäftigt sich mit der Übertragung eines biologischen Gehirns auf ein kleines künstliches Gehirn. Insektengehirne weisen autonome Verhaltensmuster auf, die überraschend komplex sind und verwenden allgemeine Prinzipien der neuronalen Vorgänge, wie dem Lernen.

Es wurde ein Neuronales Netz entworfen, dass das assoziierte Lernen simuliert und das untersucht, ob der Roboter, der von diesem Netz gesteuert wird, nach der Konditionierung sein Verhalten bei einem neutralen Reiz ändert. Im Rahmen der Bachelorarbeit wurde untersucht, in wie weit das Neuronale Netz einen neutralen und einen bedingten Reiz paaren kann. Es wurde ein Roboter, den NeuroRover, zur Verfügung gestellt, der vom Simulator gesteuert wird. Der Rover wurde mit der HaViMo2.0 Kamera ausgestattet, die Farben erkennt, sodass der Roboter diese Informationen verarbeiten kann. Mit dem Arduino WiFi-Shield ist eine drahtlose Kommunikation zwischen dem NeuroRover und dem Neuronalen Netz garantiert. Die Simulationssoftware wurde um eine Logging-Funkion erweitert, die jede Aktivität aller Neuronen und Synapsen protokolliert. Der neutrale Reiz ist die von der HaViMo2.0 erkannte Farbe, die rot oder blau ist. Beim bedingten Reiz handelt es sich um eine Bestrafung oder eine Belohnung. Nach zahlreichen Versuchsdurchführungen und Auswertungen konnte das Neuronale Netz zwei Reize miteinander paaren. Ein neutraler Reiz sorgte für eine Verhaltensänderung des NeuroRovers. Bei der Konditionierung veränderten sich die Synapsengewichte so derart, dass ein neutraler Reiz das gleiche Verhalten auslöste wie das gleichzeitige Auftreten zweier gepaarter Reize. Die aufgenommenen Daten bestätigten die Mächtigkeit der Simulation des assoziierten Lernens, sowie die Verhaltensänderungen, die in Form von Aktivitäten von bestimmten Neuronen auftraten.

Inhaltsverzeichnis

1 Grundlagen Neuronale Netze **1**

1.1 Biologische Grundlagen 1

1.2 Plastizität und Lernen 2

1.3 Spikende Neuronale Netze 4

1.4 Neurorobotik 5

2 Die Entwicklungsplattform **6**

2.1 IQR - Simulator for large scale neuronal network 6

2.2 Die Arduino-Plattform 8

2.3 HaViMo - Computer Vision Plattforms 10

2.4 Arduino WiFi-Shield 11

2.5 DFRobotShop Rover 13

2.6 Das Einrichten des Neurorovers 14

2.7 Die Kommunikation und Kontrolle zwischen dem Neurorover und dem SSN Simulator 16

3 Durchführung und Ergebnisse **17**

3.1 Das Neuronale Netz: Eligibililty-Neurover 17

3.2 Versuchsaufgaben 19

3.3 Versuchsdurchführung 21

4 Diskussion und Ausblick **30**

5 Literaturverzeichnis **34**

6 Anhang **37**

Abbildungsverzeichnis

1.1 Neuron 1
1.2 Geruchskreislauf 4

2.1 IQR - Graphical User Interface 6
2.2 Beispielprogramm Arduino: Motortest.ino 9
2.3 HaViMo2.0 11
2.4 WiFi-Shield 12
2.5 DFRobotShop Rover 14
2.6 Die Konfiguration des WiFi-Shields 15
2.7 Die Konfiguration des W-Lan Adapters: Die Verbindung wird direkt mit dem WiFi-Shield aufgebaut. Die IP-Adressvergabe ist statisch. 16

3.1 SSN: Eligibililty-Neurover 17
3.2 SSN: Eligibililty-Neurover (Neuronenansicht) 18
3.3 Track 1 21
3.4 Netzwerkaktivität Plot 1 22
3.5 Track 2 23
3.6 Neuronenplot 24
3.7 Netzwerkaktivität Plot 2 24
3.8 Track 3 26
3.9 Neuronenausschnitt 26
3.10 Netzwerkaktivität Plot 3 27
3.11 Track 4 28
3.12 Netzwerkaktivität Plot 4 29

4.1 Ausschnitt neuronLinearThreshold.cpp 32

Tabellenverzeichnis

2.1 Befehlsübersicht Arduiono 9
2.2 Spezifikation Arduino Duemilanove 10
2.3 Befehlsübersicht WiFi-Shield 13

Kapitel 1

1 Grundlagen Neuronale Netze

1.1 Biologische Grundlagen

Die Grundelemente für das Verarbeiten und Speichern von Informationen sind die Neuronen (Strecker 1997).

Ein Neuron ist eine spezialisierte Zelle, die in vier große Bereiche unterteilt werden kann: Zum einen der Zellkörper, auch Soma genannt, der aus einem Zellkern und vielen Organellen besteht (siehe Abb. 1.1). Aus dem Zellkörper entspringen plasmatische Zellstrukturen, die Dendriten genannt werden und als Eingang des Neurons fungieren. Die empfangenen Signale werden an den Zellkörper weitergegeben (Sadava et al. 2011, S. 1253). Je nach Neuronentyp können die Verzweigungen der Dendriten stärker oder weniger stark sein. (Sadava et al. 2011, S. 1253; Strecker 1997; Kriesel 2005).

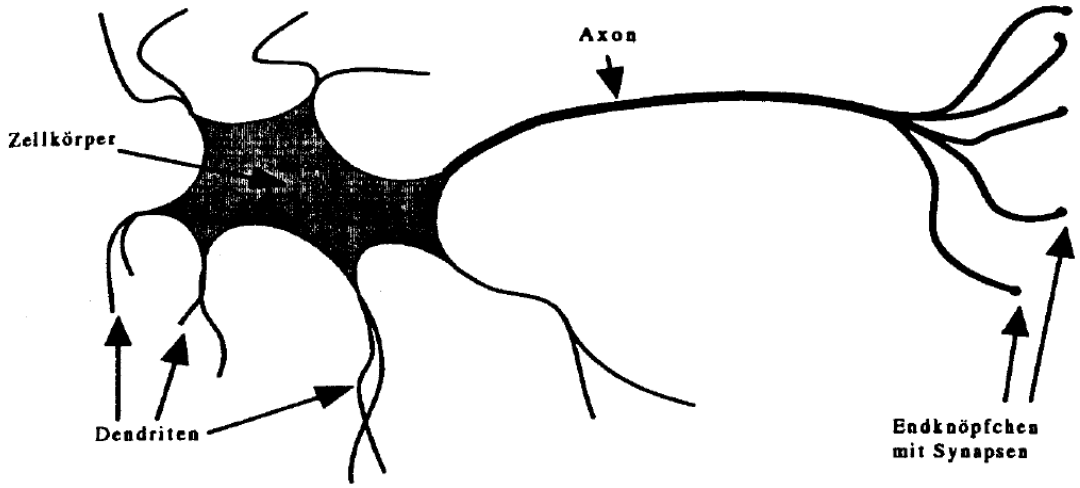


Abb. 1.1: Ein Neuron mit seinen Bestandteilen: Am Dendriten werden die Signale empfangen. Über das Axon werden Spikes bis hin zur Synapse übertragen. (Quelle: Krause 1993, S. 37)

Ein Neuron ist im Durchschnitt mit 14.000, maximal bis zu 200.000, anderen Neuronen verbunden (Strecker 1997). Das Axon leitet die Signale von dem Soma zur anderen Nervenzelle. Das Axon verzweigt sich und endet mit einer Kontaktstelle zur anderen Zelle. Diese Stelle wird Synapse bezeichnet (Savada et al. 2011; Strecker 1997). Von der Präsynapse werden bei einer Reizübertragung Neurotransmitter freigesetzt, die an die Rezeptoren der Postsynapse andocken. Dabei gibt es Neurotransmitter, die eine hemmende und eine erregende Wirkung haben. Falls an einer Postsynapse genug erregende Neurotransmitter angedockt sind, wird ein Aktionspotential ausgelöst (vgl. Sadava et al. 2011, S. 1267ff; Strecker 1997, Kriesel 2005). Ein Aktionspotential, auch Spike genannt, ist eine Änderung der Spannung in einer Nervenzelle, die die elektrische Spannungsdifferenz weiterleitet (vgl. Savada et al. 2011; Strecker 1997, Kriesel 2005).

1.2 Plastizität und Lernen

Die Informationen werden in den Synapsen gelagert. Diese können in Form von Erinnerungen auftreten. Die Erinnerungen bleiben unterschiedlich lang erhalten (vgl. Clopath 2012). Zum Beispiel kann eine Person während der Arbeitszeit vergessen, wo er sein Auto geparkt hat. Wo er sein Auto bei seiner Hochzeit geparkt hat, bleibt deutlich länger in der Erinnerung (vgl. Clopath 2012). Ebenso wurde festgestellt, dass die Gewichte der Synapsen variieren können. Die Veränderung der Stärke von Synapsen wird *Plastizität (englisch: plasticity)* genannt (Clopath 2012, Sadava et al. 2011, S. 1270). Die Plastizität kann in Abhängigkeit der Zeit, in der das Synapsengewicht modifiziert ist, in drei Gruppen klassifiziert werden:

- *short-long-term-plasticity*: dauert wenige Minuten an,
- *early-long-term-plasticity*: dauert wenige Stunden an,
- *late-long-term-plasticity*: dauert um die 10 Stunden an.

Late-long-term-plasticity wird auch *synaptic consolidation genannt* und erlaubt relevante Erinnerungen innerhalb einer Synapse zu festigen (vgl. Clopath 2012). Die synaptische Plastizität wird in der Regel durch die Paarung von prä- und postsynaptische Spikeakti-

vitäten induziert (Clopath 2012).

Late-long-term-plasticity spielt für das *Lernen* eine wichtige Rolle (Savada et. al, 2011; Clopath 2012). Eine Variante des Lernens ist das assoziative Lernen (vgl. Neurosci, 2013). Dabei werden zwei Reize, wobei der eine neutral und der andere entweder eine positive oder negative Auswirkung hat, mit der selben Antwort verknüpft (Savada et al 2011; Spektrum; Izhikevich 2007). Bei dem zweiten Reiz kann es zum Beispiel um eine Belohnung oder eine Bestrafung handeln. Dieser Reiz dient als Verstärkung des neutralen Reizes. Beide Reize müssen zeitlich nah beieinander liegen, wenn das Individuum einen Zusammenhang erkennen soll. Diese Form des Lernen wird auch Konditionierung genannt und ist eine wichtige Grundlage des Gedächtnisses (.[Spektrum]). Es gibt eine Reihe von Studie, die die Konditionierung untersuchten. Ein Beispiel dafür ist die olfaktorische Wahrnehmung eines Duftes in Kombination mit Zuckerwasser bei einer Biene. Die Biene streckt ihren Rüssel raus, wenn die Antenne mit Zuckerwasser in Berührung kommt. Wenn beide Reize kombiniert erscheinen, so streckt die Biene ihren Rüssel, wenn sie den Duft wahrnimmt (vgl. Bitterman et al, 1983).
Das entscheidende Signal für das Verstärken von neutralen Reizen ist Dopamin (Schultz 1998). Dopamin ist ein Neurotransmitter im Nervensystem und reguliert belohnungsbezogende Verhaltensmuster (vgl. Wijekoon & Dudek). Neuronen, die für die Ausschüttung von Dopamin zuständig sind, kommen nicht häufig im Gehirn vor. Dennoch sind ihre Wirkungsbereiche sehr groß (vgl. Wijekoon et. Dudek) Der Pilzkörper (engl: mushroom body), der mehrere hunderttausend Kenyon-Zellen enthält, ist ein Bereich im Gehirn und ist entscheidend für das assoziative Lernen von Gerüchen. Hingegen dazu sind die Neuronen im Antennenlappen, die die Quelle der Kenyon-Zellen sind, weniger vertreten. Die Kenyon-Zellen leiten Informationen an zwei Bereichen, α- und β-Lobus, weiter. Dort befinden sich extrinsische Neuronen. Die Synapsen zwischen den Kenyon-Zellen und den Zellen aus dem β-Lobus sind variabel. Diese Änderungen werden durch die STPD Hebbsche Lernregel hervorgerufen (siehe Abb. 1.2).

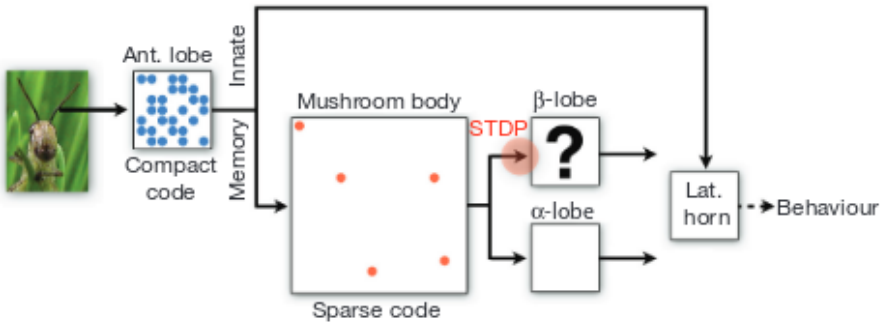


Abb. 1.2: Der Geruchskreislauf eines Schmetterlings: Falls die Gerüche kein instinktbedingtes Verhalten auslöst, werden die Geruchssignale bearbeitet. Abhängig davon, ob Erfahrungen mit dem Geruch gesammelt wurden, werden die Signale an das α- oder β-Lobus weitergeleitet, die dann unterschiedliche Verhaltensmuster auslösen.

1.3 Spikende Neuronale Netze

Um Neuronale Netze mit einem Computer simulieren zu können, ist es wichtig zu wissen, wie Informationen codiert werden, die in Formen von Spikes versendet werden. Nach biologischen Untersuchen können folgende Codierungsformen von Bedeutung sein:

- **Feuerrate:** Die Codierung erfolgt in Form von der Frequenz der Feuerrate. Das heißt, dass der Informationsgehalt mit der Anzahl der Spikes in einer bestimmten Zeit abhängig sein kann.
- **temporale Kodierung:** Der Zeitpunkt, an dem Spikes ausgeführt werden, ist entscheidend für den Informationsgehalt
- **Populationskodierung:** Die Codierung ist von bestimmten Neuronengruppen abhängig.

(vgl. Ruf 1998)

Bisherige Ansätze zur Simulation von künstlichen neuronalen Netzen sind *Artificial Neuroal Networks (ANN)*. In diesen Netzwerken werden die Informationen mit der Feuerrate der Spikes codiert (vgl. Maass 1997). Die Artificial Neural Networks haben den entscheidenden Nachteil, dass aufgrund dieser Codierungsform nicht genügend schnell Informationen verarbeitet werden können, die zum Beispiel bei der Mustererkennung sehr wichtig ist (vgl. Ruf 1998).

Ein weit besserer Ansatz ist das *Spiking Neural Network (SNN)*. Bei diesem Modell ist der Informationsgehalt über die temporale Kodierung abhängig. Spiking Neuroal Netzwerke bieten eine realistische Darstellung von Simulationen, die den realistischen, biologischen Vorgängen der Informationsübertragung und -verarbeitung sehr Nahe ist (vgl. Helgadòttir 2013).

1.4 Neurorobotik

Die Neurorobotik beschäftigt sich mit der biologischen Umsetzung von Modellen im Form von Gehirnen auf ein künstliches Neuronales Netzwerk, das autonome Neuroroboter steuert (Krichmar 2008). Ein Neuroroboter ist ein Gerät, deren Steuerung nach dem Prinzipien eines Nervensystem gerichtet ist und testet Hypothesen aus der Biologie und hilft es die Biologie und das Gehirn besser verstehen zu können (Krichmar 2008).
Dabei soll der Neuroroboter in echten Umweltbedingungen interagieren. Zu seinen weiteren Aufgaben gehört das Übernehmen von Verhaltensmuster. Dabei wird das Verhalten selbst vom neuronalen Netzwerk gesteuert (Krichmar 2008).
Im Rahmen der Bachelorarbeit wurde ein Neuronales Netz entworfen, das das assoziierte Lernen simulieren soll. Dazu wird ein Roboter, den NeuroRover, von diesem Netz gesteuert, der in Abhängigkeit von Erfahrungen bestimmte Verhaltensmuster aufweisen soll, wenn darauf bezogene, neutrale Reize registriert werden.

Kapitel 2

2 Die Entwicklungsplattform

2.1 IQR - Simulator for large scale neuronal network

Die Softwaresimulation für neuronale Netze IQR ist eine open-source Software, die unter der GNU Public Licence lizenziert wurde. Diese Software läuft unter Windows, Mac OS und Linux. Mit der IQR-Software können Neuronale Netze kreiert und simuliert werden. IQR stellt das Neuronale Netz in drei Levels da. Das oberste Level ist das System.

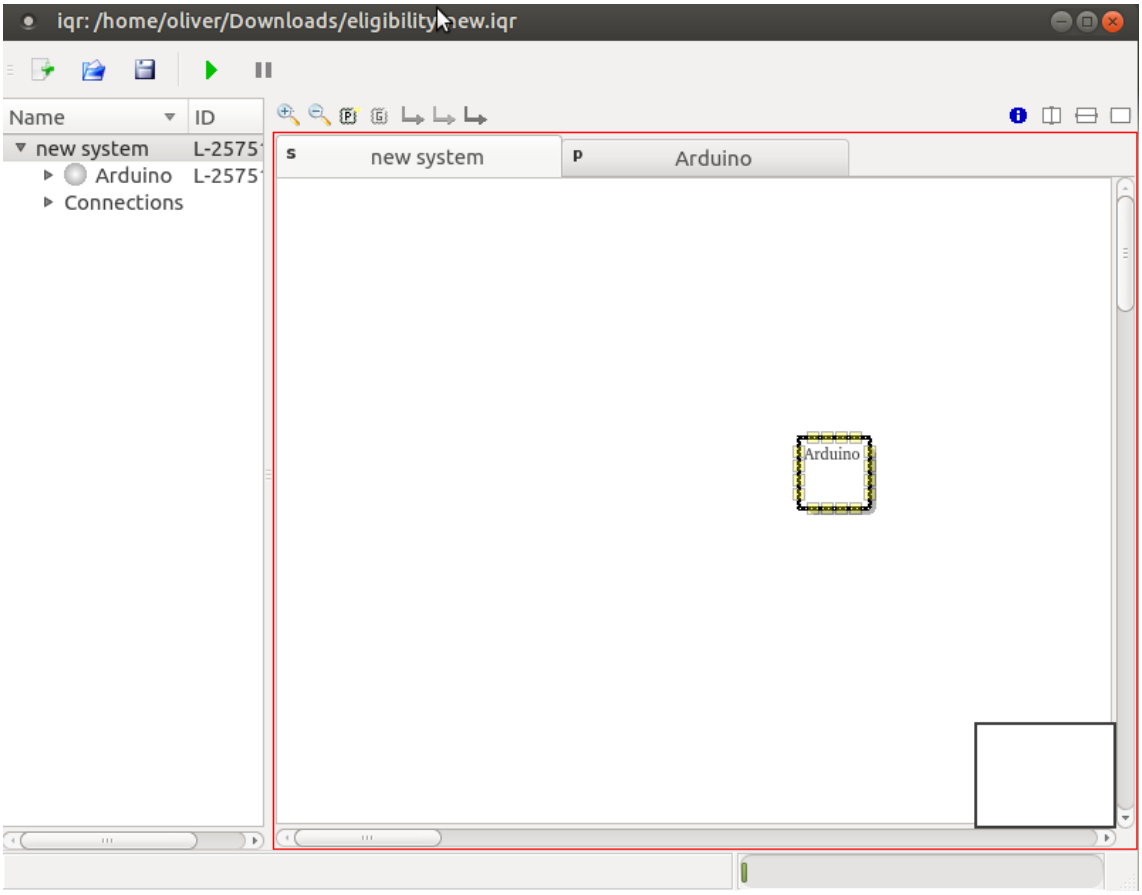


Abb. 2.1: Ein Ausschnitt von der Neuronensimulationssoftware IQR: IQR bietet eine übersichtliche GUI, die das Kreieren von Neuronalen Netzen sehr vereinfacht

In einem System werden die Gruppen erstellt, die Neuronen enthalten. Die Neuronen bzw. Neuronengruppen werden miteinander verbunden. Die Verbindungen simulieren die Synapsen. Da Neuronale Netze sehr groß und komplex sind, ist es möglich, dass man die Übersicht verliert. Daher ist es möglich Prozesse zu exportieren und in andere Systeme zu importieren. IQR bringt eine Reihe von vordefinierten Neuronen und Synapsen. Es können zusätzlich Neuronen und Synapsen selbst erstellt und integriert werden. Zudem kann ein Prozess Module laden. Zum Beispiel ist das Anbinden von einem Roboter möglich, sodass dieser vom Simulator gesteuert werden kann.
Die GUI bietet eine geordnete Übersicht (siehe Abb. 2.1). So bietet die linke Seite des GUIs eine Übersicht aller Neuronengruppen und deren Verbindungen. Auf der rechten Seite können Prozesse erstellt werden. Nach dem Erstellen können Neuronengruppen hinzugefügt werden. Unter den Einstellungen kann bestimmt werden, um welchen Neuronentypen es sich handelt und wie viele Neuronen in einer Gruppe vorhanden sind. Zudem können neuronenspezifische Eigenschaften eingestellt werden. Die Gruppen können sich gegenseitig beeinflussen: Zwischen ihnen gibt es unidirektionale Verbindungen, die folgende Eigenschaften haben:

- exzitatorisch,
- inhibitorsich und,
- modular

Wie Synapsen können die Signale verstärkend oder hemmend auf das Neuron wirken. Während der Simulation können die Spikeaktivitäten der Neuronen grafisch visualisiert werden. Mit dem Time Plot werden die Aktivitäten in Abhängigkeit der Zeit angezeigt. Dabei kann auf Wunsch ausgewählt werden, ob die gesamte Live-Simulation (act) oder bestimmte Inputs, wie exzitatorisch oder inhibitorisch, angezeigt werden sollen. Zum Ansteuern des Roboters wurde ein Modul moduleAnalagBehaviour geschrieben, das in das System geladen wird. Das Modul enthält sieben Gruppen, wobei drei für die Eingabe und vier die Ausgabe zusammengefasst wird. Die Eingabegruppen sind:

- Color,
- Reward und,
- Punishment

Die Color - Gruppe zeigt an, ob eine vordefinierte Farbe wahrgenommen wurde. In Kombination mit der Reward und Punishment Gruppe wird ein Verhaltensmuster ausgelöst, welches die Ausgabegruppe entspricht:

- Nothing: der Roboter tut nichts,
- Explore: der Roboter erkundet,
- Approach: der Roboter geht auf etwas zu und
- Retreat: der Roboter erwidert.

(vgl. Bernardet 2010)

2.2 Die Arduino-Plattform

Das Arduino ist ein Mikrocontrollerboard, das neben dem günstigen Erwerb auch den Vorteil hat, dass die Programmierung auch in C erfolgt. Aufgrund der Popularität gibt es eine große, aktive Community, viele verschiedene Projekte und in- und offizielle Hardware, wie das WiFi Shield, die den Funktionsumfang des Mikrocontrollers erweitern.

Der DFRobotShop Rover basiert auf dem Arduino Duemilanove. Dieser Mikrocontroller ist die neueste Serie der USB Arduino Boards. Sie basiert auf dem ATmega328 und hat 14 digital I/O Pins, 6 analoge Inputs, einen 15 MHz Kristalloszillator, einen USB-Anschluss, einen Power-Eingang und einen Reset-Button.

Die Arduino Duemilanove bietet eine Vielfalt von Kommunikationsmöglichkeiten mit verschiedenen Endgeräten, wie dem Computer, anderen Arduinos oder Mikrocontroller. Zum einen unterstützt der AtMega328 die UART TTL serielle Kommunikation, die über den digitalen Pin 0 (RX) und 1 (TX) verfügbar ist. Mit dem FTDI FT232RL Chip ist die serielle Kommunikation über USB möglich, da die FDTI Treiber einen virtuellen COM Port unterstützt. Zudem ist es mithilfe der SoftwareSerial Bibliothek möglich, über alle digitale Pins vom Arduino Duemilanove seriell zu kommunizieren. Weitere Kommunikationswege, wie I2C und SPI Kommunikation ist dank ATmega328 ebenfalls möglich.

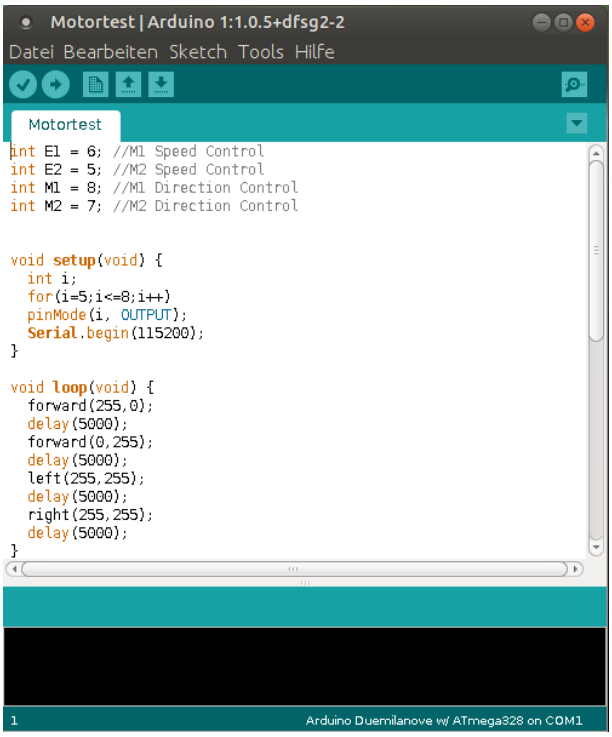


Abb. 2.2: Beispielprogramm für das Arduino: Motortest.ino. In der setup()-Methode wird festgelegt, dass die Pins 5 bis 8 Output-Pins sind. Die Baudrate beträgt 115200. Die loop()-Methode beinhaltet vier Funktionen, die jeweils den Motor ansteuern. Zwischen den Funktionen wird das Sketch für 5000 Millisekunden angehalten.

Für die Arduino-Serie gibt es eine kostenlose Entwicklungsumgebung, die Arduino-GUI genannt wird (siehe Abb. 2.2). Unter dem Menüpunkt Tools>Board kann dann der

Funktion	**Beschreibung**
pinMode(Pin,Modus)	In dieser Funktion kann definiert werden, ob ein digitaler Pin ein Input oder Output ist
digitalWrite(Pin,Wert)	Der digitale Kanal kann auf High oder Low gesetzt werden. Dieser Kanal muss jedoch den pinMode Output haben
digitalRead(Pin)	Am digitalen Kanel werden digitale Signale abgelesen. Der Kanal muss jedoch als pinMode Input definiert sein
delay(Wert)	Der Programmablauf wird auf die angegebene Zeit in Millisekunden verzögert
Serial.begin(Baudrate)	Diese Funktion sorgt für den Start der seriellen Kommunikation zwischen dem PC und dem Arduino-Board. Die Baudrate standatisiert mit 300,1200,...
Serial.println(Daten)	Mit dieser Funktionen werden Daten über die serielle Schnittstelle gesendet

Tab. 2.1: Diese Tabelle beinhaltet die wichtigsten Befehle für das Arduino. (Quelle: [Arduino])

Microcontroller	ATmega168
Operating Voltage	5V
Input Voltage (recommended)	6-20V
Digital I/0 Pins	14
Analog Input Pins	6
DC Current per I/0 Pin	40mA
DC Current for 3.3V Pin	50mA
Flash Memory	16KB (ATmega168) oder 32KB (ATmega328) of which 2KB used by bootloader
SRAM	1KB (ATmega168) or 2KB (ATmega328)
EEPROM	512 bytes (ATmega168) oder 1KB (ATmega328)
Clock Speed	16MHz

Tab. 2.2: Ein tabellarische Zusammenfassung über die Spezifikation der Arduino. (Quelle: [Arduino])

Arduinotyp ausgewählt werden. Für die Arduino Duemilanove wird Arduino Duemilanove ATmega328 selektiert. Der Atmega328 enthält einen Bootloader, sodass das Flashen von neuen Codes keine weitere externe Hardware benötigt wird. Es kann passieren, dass der Bootloader beim Flashen beschädigt wird. Dieses kann mithilfe einer weiteren Arduino-Hardware wieder hergestellt werden. Ein Arduino-Programm wird Sketch genannt. Sie enthält mindestens zwei Funktionen. Zum einen die setup-Methode und zum anderen die loop-Methode.
Die setup-Methode wird nur zum Start der Programmausführung genau einmal aufgerufen. Dort können Pins und Variablen initialisiert werden.
Die loop-Methode, die direkt nach der setup-Methode aufgerufen wird, stellt eine Endlosschleife dar. Dort nach Komplexität eines Sketches können zusätzliche Methoden definiert werden, die den Aufbau des Codes erleichtern können. (vgl. [Arduino])

2.3 HaViMo - Computer Vision Plattforms

HaViMo ist ein Computervision - Modul, welches mit einem CMOS Kamerachip ausgestattet ist (Abb. 2.3). Zudem ist ein Mikrokontroller vorhanden, der die Bildverarbeitung ausführt. Das verarbeitete Bild ist über einen seriellen Port zugänglich. Von der HaViMo gibt es eine Reihe von verschiedenen Ausführungen. Der Roboter wurde mit dem HaViMo 2.0 ausgestattet. Dieses Modul bietet im Gegensatz zu seinen bisherigen

Ausführungen unter anderem eine bessere Bildfrequenz (19 Bilder pro Sekunde).

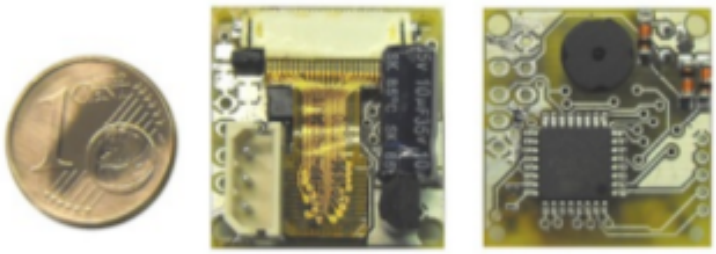


Abb. 2.3: Die HaViMo2.0 Kamera: Sie ist sehr klein und wird vorne am Rover befestigt.

Die integrierte Kamera hat eine Auflösung von 160x120 Pixeln, eine Farbtiefe von 12 Bits und einen vollen Zugriff auf alle Register. Die Werte werden in einem EEPROM (Eletronical Erasable Programmable Read Only Memory) gespeichert. Der Vorteil ist, dass die Kamera nach einem Neustart nicht erneut konfiguriert werden müssen. Falls neue Werte hinzugefügt werden sollen, dann wird der Speicher neu geflasht.
HaViMo 2.0 unterstützt unter anderem die ROBOTIS und RoboBuilder Bridges. Die kommunizieren über halb duplex und voll duplex serial Protokolle. Die Kalibrierung der Kamera kann über jeweils über die beiden Bridges erfolgen, sodass HaViMo 2.0 und der PC kommunizieren können. Allerdings kann auf eine Bridge verzichtet werden: Die Kalibrierung erfolgt direkt vom Computer auf die HaViMo2.0 mithilfe eines USB2Dynamixel.

2.4 Arduino WiFi-Shield

Das Arduino WiFi-Shield ist ein Erweiterungsmodul für das Arduino Board, das eine drahtlose Kommunikation zwischen dem Arduino und dem Zielgerät ermöglicht. Ein großer Vorteil des WiFi-Shields ist, dass es bei Inaktivität im Standby Modus gesetzt wird und somit wertvolle Energie gespart wird.

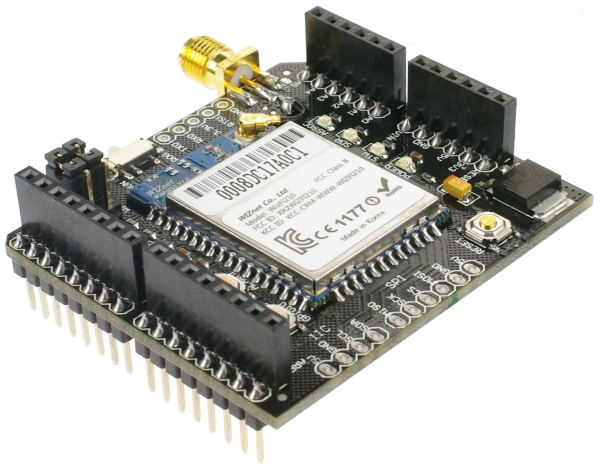


Abb. 2.4: Die Netzwerksschnittstelle: Arduino WiFi-Shield. Diese Erweiterung ermöglicht eine drahtlose Kommunikation mit dem NeuroRover (Quelle: [WiFi Shield])

Das WiFi Modul basiert auf das WizFI210 und unterstützt das Überführen einer TTL seriellen Port Kommunikation zu einer IEEE802.11b/g/n Wireless Kommunikation. Das Arduino WiFi-Shield unterstützt verschiedene Sicherheitsprotokolle, wie WEP, WPA/WPA2-PSK, oder PEAP. Die unterstützten Datenübertragungsraten liegen gemäß dem IEEE802.11b bei 11, 5.5, 2, 1 Mb/s. Das WiFi Shield hat ein Paar aus Jumper und einen Schalter. Die Jumper können folgende Pinbelegungen festlegen:

- USB: WiFi Konfiguration (AT mode),
- WIFI: Arduino kann über WiFi kommunizieren und
- NONE: WiFI Shield ist deaktiviert. Eine Arduino Programmierung ist jetzt möglich!

Mit dem Schalter können folgende Einstellungen getroffen werden:

- RUN: WiFI Konfiguration (AT mode) und
- PROG: Arduino kann über WiFi kommunizieren.

Um das Arduino WiFi Shield konfigurieren zu können, müssen die Jumpers auf USB und der Schalter auf RUN gesetzt werden. Zusätzlich wird ein Terminal, wie Putty, gebraucht, das verschiedene Kommunikationsprotokolle unterstützt und über das Arduinos COM Port mit 115200 Bps verbunden ist. Das WiFI Shield unterstützt den AT-Befehlssatz, dessen wichtige Kommandos in der Tabelle 2.3 aufgelistet sind.

Befehl	Beschreibung
AT:	Betreten des AT Modus
AT+WWPA	Festlegen eines Passwortes
AT+WA	Definiere Routers SSID
AT+NSET	Statische Netzwerkparameter
AT+NDHCP	Aktiviert die DHCP Einstellung
AT+WAUTO	WiFi Parameter werden so gesetzt, dass eine autoconnect stattfinden kann
ATA	Verbinde

Tab. 2.3: Diese Tabelle beinhaltet die wichtigsten Befehle für die Konfiguration des Arduino WiFi-Shields (Quelle: [WiFi Shield])

(vgl. [WiFi Shield])

2.5 DFRobotShop Rover

Der DFRobotShop Rover ist ein vielseitiger, mobiler und panzerähnlicher Roboter, der auf dem Arduino Duemilanove, der den ATMega328 Chip enthält, basiert. Zu den Ausstattungen gehören die Tamiya twin moto gearbox, die separat erhältlich ist, einen dual motor driver, einen Onboard Spannungregulator (mindestens 3.5V und maximal 9V), einen 3.7V LiPo Ladegerät, einen 7.4V LiPo Akku einen Temperatur- und Lichtsensor.

Der dual motor driver ist an den digitalen Pins 5 bis 8, der Lichtsensor an den analogen Pin 0 und Temperatur Sensor am analogen Pin 1 angeschlossen.

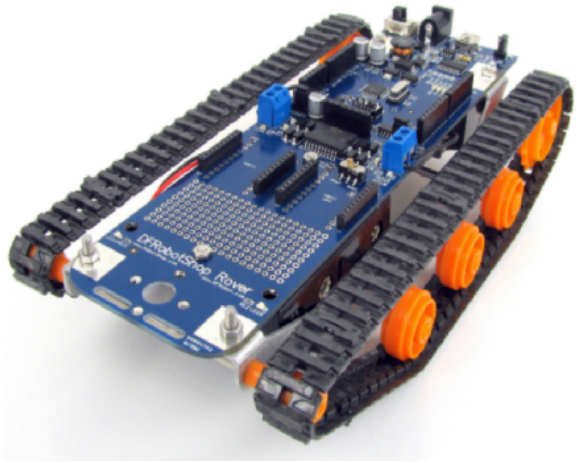


Abb. 2.5: DFRobotShop Rover: Der NeuroRover ohne Arduino-WiFi Shield und HaViMo2.0 Kamera (Quelle: .[RobotShop])

Es gibt verschiedene Möglichkeiten den DFRobotShop mit Strom zu versorgen. Zum einen kann der Roboter über USB mit einem USB-Kabel mit Strom versorgt werden. Das hat den Vorteil, dass eine 3,7V LiPo Batterie aufgeladen wird, wenn sie am weißen mini JST connector angeschlossen ist, aber hat den Nachteil, dass nur der Mikrocontroller mit Energie versorgt wird. Zum anderen kann die Stromzufuhr über den Barrel connector erfolgen. Dabei werden sowohl der Mikrocontroller als auch der Motor mit Strom versorgt. (vgl. [RobotShop])

2.6 Das Einrichten des Neurorovers

Der NeuroRover besteht aus drei Bestandteilen: DFRobotshop Rover, Arduino-WiFI-Shield und HaViMo2.0. Das Arduino-WiFi-Shield auf dem DFRobotshop Rover gesteckt. Die HaViMo2.0 muss am RX und und TX-Pin gesteckt werden. Im simple-command-Arduino Firmware sind die RX und TX-Pins wie folgt definiert:

- RX-Pin = Digitaler Pin 2
- TX-Pin = Digitaler Pin 3

Für die Stromversorgung der HaViMo2.0 werden zwei Kabel in den 5V Pin gesteckt. Über die digitalen Pins 7 und 8 werden die Motoren angesteuert und über die analogen Pins 5 und 6 kann die Geschwindigkeit des Motors festgelegt werden.
Zunächst muss die Kamera kalibriert werden. Dabei muss erst die HaViMo-GUI Firmware auf das Arduino-Board vom DFRobotshop geflasht werden, damit eine Kommunikation zwischen dem PC und der HaViMo2.0 möglich ist. Nach dem Verbinden des Neurorovers mit dem Computer über ein USB-Kabel kann über die HaViMo-GUI der Verbindungstyp und der Port gewählt werden. Die Übertragung erfolgt über den RoboBuilder, sodass der Verbindungstyp *RoboBuilder* ist. Der Port wird automatisch gewählt. Für die Kalibrierung wurden jeweils zwei blaue und zwei rote Blumentöpfe verwendet. Die Farbe jedes einzelnen Topfes wurde in verschiedenen Winkeln erfasst, da der Lichteinfall aufgrund des Standortes der Blumentöpfe unterschiedlich war. Die aufgenommenen Farben können aufgrund des EEPROMs gespeichert werden und eine erneute Kalibrierung ist nach einer unterbrochenen Stromzufuhr nicht nötig.
Damit der Neurorover kabellos vom IQR-Simulator gesteuert werden kann, muss das Arduino WiFi-Shield konfiguriert werden. Dazu wird der DFRobotshop Rover erneut als Bridge benötigt. Die Jumper müssen auf USB und der Schalter des WiFi-Shields auf RUN gesetzt werden. Der DFRobotshop Rover muss mit dem Computer per USB-Kabel verbunden werden. Mithilfe der WizSmart Software kann das WiFi-Shield über Putty die AT-Kommandos empfangen werden. Für die erfolgreiche Konfiguration müssen folgende Kommandos ausgeführt werden:

```
1 at
2 at+ndhcp=0
3 at+nset=192.168.10.100,255.255.255.0,192.168.10.101
4 at+wauto=1,Arduino,,3
5 at+nauto=1,1,,4000
6 at&w0
7 ata
```

Abb. 2.6: Die Konfiguration des WiFi-Shields mit AT-Kommandos. Als Adhoc-Name wird Arduino verwendet. Auf eine Verschlüsselung wird verzichtet.

Nun müssen die Einstellungen für den W-LAN Adapter des Computers vorgenommen werden. Der Konfiguration des Shields zufolge wird eine Adhoc-Verbindung aufgebaut. Dabei werden folgenden Parameter festgelegt:

```
1 SSID: Arduino
2 Mode: Ad-hoc
3 Security: None
4 Statische IP-Vergabe
5 Adresse: 192.168.10.100
6 Netzmaske: 255.255.255.0
7 Gateway:192.168.10.101
```

Abb. 2.7: Die Konfiguration des W-Lan Adapters: Die Verbindung wird direkt mit dem WiFi-Shield aufgebaut. Die IP-Adressvergabe ist statisch.

2.7 Die Kommunikation und Kontrolle zwischen dem Neurorover und dem SSN Simulator

Der DFRoboShop Rover, das WiFi-Shield und die HaViMo2.0 Kamera bilden zusammen den Neurorover. Das WiFi-Shield wird auf dem Rover gestellt und die HaViMo Kamera wird an den digitalen Pins 3 und 4 gesteckt. Der Lichtsensor ist am analogen Pin 0 angeschlossen.

Die HaViMo Kamera nimmt ein Bild auf. Dieses wird verarbeitet und geprüft, ob auf dem Bild eine gespeicherte Farbe erkannt wird. Die Resultate werden an den DFRobotShop Rover gesendet. Auf dem Rover, bzw. auf dem Arduino Duemilanove, läuft die Sketch RobotSimpleStatusAndCommand.

Der Rover liest zuerst die sensorischen Daten aus. Dabei wird jeweils geprüft, ob die Kamera eine bestimmte Farbe erkannt hat. Zudem werden die Licht- und Infrarotsensoren abgelesen, ob diese Reizströme aufgenommen haben. Diese Informationen werden über eine virtuelle serielle Schnittstelle an IQR gesendet. IQR lädt für das Ausführen des SNN das Modul ModuleArduinoAnalogBehaviours, welches Informationen von Neuronengruppen abliest oder hinzufügt. Die aufgenommenen Signale, die Informationen von den Sensoren und der Kamera enthalten, dienen als Input für die Neuronengruppen Colors, Reward, und Punishment. In Abhängikeit der empfangenen Signale werden andere Neurongruppen aktiv. Dazu gehören Nothing, Explore, Approach und Retreat. IQR überprüft welche der vier Neuronengruppe am aktivsten sind und schickt dem Neurorover den zugehörigen Wert der am stärksten aktiven Neuronengruppe.

Der Neurorover liest periodisch die serielle Schnittstelle aus, verarbeitet sie und führt das entsprechende Verhalten aus.

Kapitel 3

3 Durchführung und Ergebnisse

3.1 Das Neuronale Netz: Eligibililty-Neurover

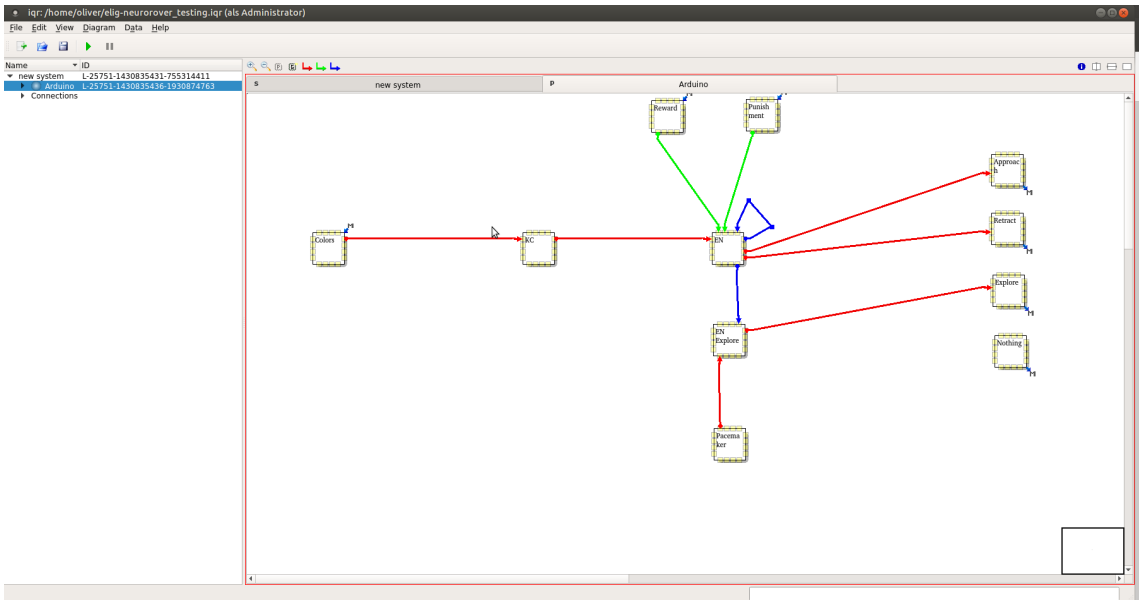


Abb. 3.1: Das Neuronale Netz: Eligibililty-Neurover

Das Neuronale Netzwerk Eligibililty-Neurover besitzt vier Eingangsneuronen. Diese bilden drei Neuronengruppen: Zwei Neuronen für die Color-Gruppe und jeweils ein Neuron für die Punishment- und Rewardgruppe. Zudem existieren vier Ausgangsneuronen, die vier Neuronengruppen bilden: Retreat-, Approach, Expore- und Nothinggruppe (siehe Abbildung 3.1).

Von der Colorgruppe ausgehend existiert eine exzitatorische Synapse (Color $\rightarrow_e$ KC), die mit der KC-Gruppe verbunden ist. Diese Gruppe besteht aus 10x10 Neuronen. Von der KC-Gruppe ausgehend gibt es eine exzitatorische Synapse (KC $\rightarrow_e$ EN), die mit der EN-Gruppe verbunden ist. Von den Punishment- und Rewardneuronengruppen gehen jeweils eine modulare Synapse aus (Reward $\rightarrow_m$ EN) und (Punshment $\rightarrow_m$ EN), die

jeweils ein Neuron, aber nicht das gleiche, der EN-Gruppe treffen. Aus beiden Neuronen gehen jeweils eine exzitatorische Synapse in die Aproach- und Retreatneuronengruppe zu (EN $\rightarrow_e$ Aproach und EN $\rightarrow_e$ Retreat). Außerdem geht eine inhibitorische Synapse von der EN-Gruppe in die EN-Explore-Neuronengruppe hervor (EN $\rightarrow_i$ EN-Explore). Diese Neuronengruppe wird vom Pacemakerneuron verstärkt, das zeitlich zufällige Spikes erzeugt (Pacemaker $\rightarrow_e$ EN-Explore). Die EN-Explore-Neuronengruppe erregt das Explore-Neuron mit einer exzitatorische Synapse (EN-Explore $\rightarrow_e$ Explore).

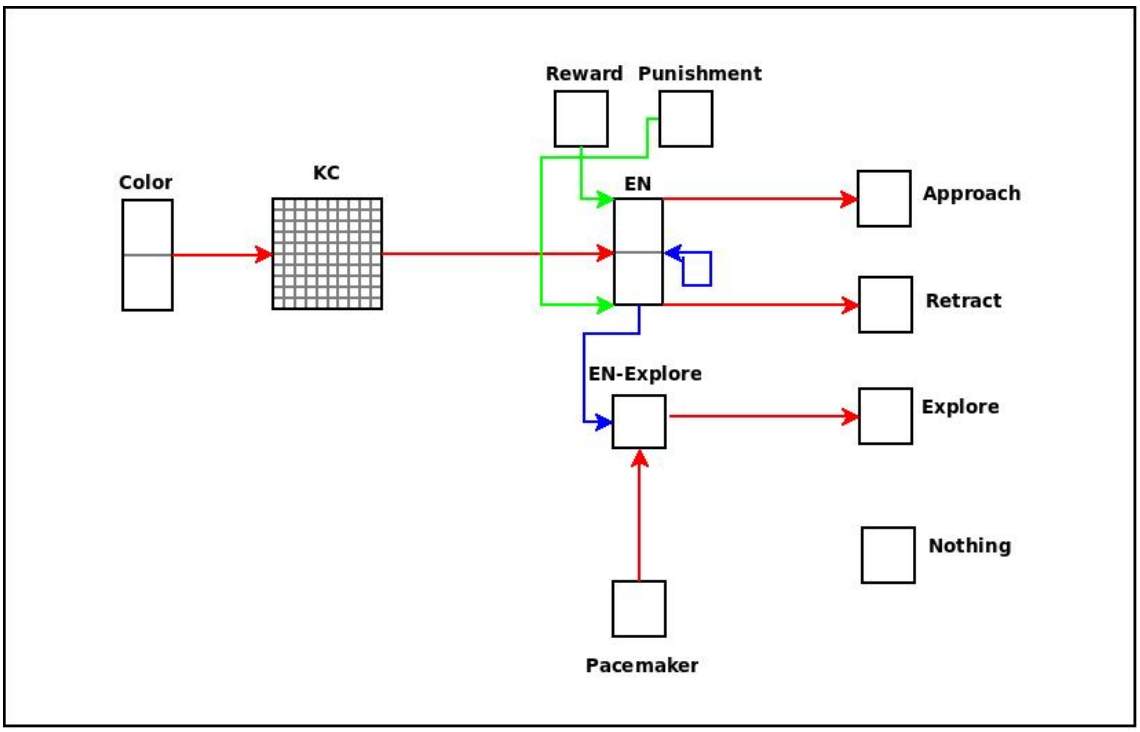


Abb. 3.2: Das Neuronale Netz Eligibililty-Neurover in der Neuronenansicht.

Die Abbildung 3.2 wurde entworfen, damit die Visualisierung nicht nur auf Gruppenebene, sondern auch auf Neuronenebene verstanden wird. Es ist ersichtlich, dass Color $\rightarrow_e$ KC zwei Color-Neuronen und 100 KC-Neuronen verbindet. Diese zwei Neuronen führen Spikes aus, wenn die Farbe blau oder rot gesehen werden, wobei die Farben den Neuronen eindeutig zugeordnet sind. Die Synapsen (Reward $\rightarrow_m$ EN) und (Punishment $\rightarrow_m$ EN) übertragen die Reize, die bestrafend oder belohnend sind. Das Reward-Neuron gibt Spikes an das obere EN-Neuron, das dann das Approach-Neuron erregt. Analog dazu gibt das Punishment Spikes an das untere EN-Neuron, das das Retract-Neuron erregt. Die beiden Synapsen sind modular. Das heißt, dass EN durch diese Eingangsneuronen nicht direkt exzitatorisch erregt wird. Diese Reize sollen lediglich die Gewichte der

(KC $\rightarrow_e$ EN) ändern, sodass in Abhängigkeit der gleichzeitigen Color-Neuronen und Reward- und Punishment-Neuronen das obere oder untere EN-Neuron erregt wird, das letzendlich die Hemmschwelle überschreitet.
In den Versuchen soll untersucht werden, ob die Synapsengewichte sich derart ändern, sodass der Neurorover das Verhalten Approach oder Retract ausführt.

3.2 Versuchsaufgaben

Die Experimente bestehen aus vier größeren Versuchen.

Teil 1:
Ein naiver Roboter wird in einer bestimmten Position der Arena aufgestellt. Dabei ist kein Objekt in Sicht. Die IQR-Simulation soll für 30 Sekunden ausgeführt werden.

Teil 2:
Der Roboter wird auf eine bestimmte Farbe für drei Sekunden belohnt. Anschließend wird die Simulation an der gleichen Stelle durchgeführt. Falls der Roboter die bestimmte Farbe sieht und auf diese zugeht, dann wird die Simulation abgebrochen, ansonsten soll die Simulation für mindestens 30 Sekunden durchgeführt werden.

Teil 3:
Der Roboter wird auf eine bestimmte Farbe für drei Sekunden bestraft. Anschließend wird die Simulation an der gleichen Stelle durchgeführt. Die Simulation wird 30 Sekunden durchgeführt.

Teil 4:
Der Roboter wird auf eine bestimmte Farbe für drei Sekunden belohnt und auf die andere Farbe für drei Sekunden bestraft. Danach wird der Roboter auf die gleiche Stelle gelegt und die Simulation für 30 Sekunden fortgesetzt. Falls der Roboter die Farbe sieht, auf die er belohnt wurde und auf sie zugeht, wird die Simulation beendet.

Die Versuche finden in einer quadratischen Arena statt. In jedem Eckpunkt befindet sich ein Blumentopf, der entweder blau oder rot ist. Die Anordnung der Farbtöpfe ist so gestaltet, dass vom jedem Eckpunkt die benachbarten Blumentöpfe jeweils eine andere Farbe haben. An der Decke wurde eine GoPro Hero Kamera angebracht, die die Bewegungsabläufe des Neurorovers aufnimmt, sodass die Aufnahmen getrackt werden können. Damit die HaViMo 2.0 Kamera jene Farben als solche erkennen kann, muss sie kalibriert werden. Dies kann mithilfe der Software HaViMo-GUI durchgeführt werden. Als Verbindungstyp wird *Robobuilder* verwendet und dazu wird der passende Port gewählt. Anschließend wird auf dem Neurorover die Sketch *Havimo2_5_HavimoGUI.ino* aufgespielt, sodass der Roboter als Bridge funktioniert. Nach dem erfolgreichen Verbindungsaufbau wird die HaViMo 2.0 Kamera auf einen Blumentopf gerichtet und mehrere Bilder aus verschiedenen Positionen und Winkeln aufgenommen. Die GUI bietet mehrere Slots, in denen die markierten Farben gespeichert werden. Da die Blumentöpfe die Farben rot und blau haben, werden Slot 1 und 2 als Speicherquellen verwendet. Für die Farbe rot wird Slot 1 ausgewählt. Auf der GUI erscheint beim geschossenen Foto zwei Bilder. Für das Speichern der roten Farbe wird auf dem linken Bild die roten Bildbereiche angeklickt. Auf dem rechten Bild werden diejenigen Pixel markiert, die die gleiche Farbtiefe haben, wie der Pixel, der angeklickt wurde. Diese Prozedur wird so lange durchgeführt, bis der gesamte Blumentopf markiert ist. Das Fotografieren aus verschiedenen Winkeln ist daher wichtig, weil der Lichteinfall bzw. Schattenfall nicht überall gleich ist und es dadurch zu anderen Farbtiefen kommen kann.
Für die blaue Farbe wird Slot 2 verwendet. Das Kalibrieren der Kamera erfolgt mit allen vier Blumentöpfen.
Damit der Neurorover die seriellen Daten versenden und empfangen kann, muss das Sketch *RobotSimpleStatusAndCommand.ino* geflasht werden. Zudem muss eine serielle Kommunikationsschnittstelle zwischen IQR und dem Neurorover errichtet werden. Als erstes muss der verwendete Laptop mit dem Arduino WiFi-Shield verbunden sein. Anschließend wird socat gestartet, sodass ein virtueller, serieller Port geöffnet wird. Über diesen ist eine Kommunikation zwischen dem Neurorover und IQR möglich.

3.3 Versuchsdurchführung

Versuch 1:

Der Rover wurde in die Mitte der Arena aufgestellt und fuhr für 30 Sekunden rum. Die Bewegung des Rovers war in die Fahrrichtung gerichtet, wobei es häufig kleine Schwankungen nach links oder nach rechts gab. Es war zu beobachten, dass der Roboter sich häufiger nach rechts neigte, sodass ein kreisartiger Verlauf folgte (siehe Abbildung 3.3). Die Bewegungen des Neurorovers waren nicht ganz flüssig, sondern ruckelte häufiger. Während der Laufzeit wurden zwei blaue in ein roter Blumentopf gesichtet. Im neuronalen Netz wurde beobachtet, dass das Ausgabeneuron Explore aktiv war. Hintergrund dafür ist, dass die Pacemaker-Gruppe probabilistisch aktiv ist. Die Ausgabeneuronen Retract und Approach waren nicht aktiv.

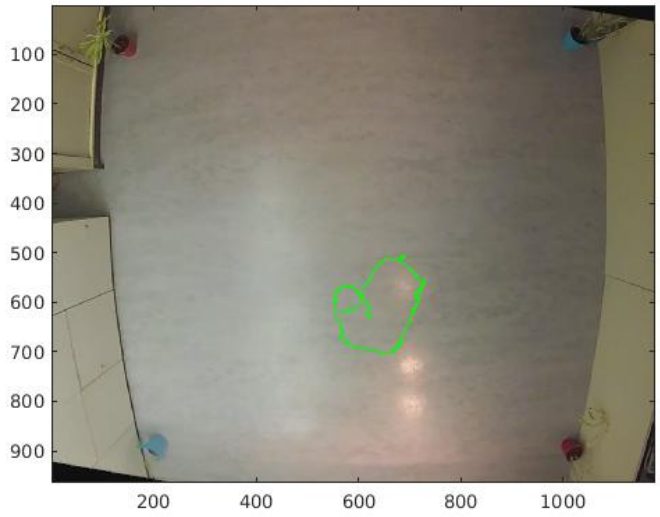


Abb. 3.3: Die getrackte Aufnahme des ersten Versuches: Der NeuroRover fuhr im Erkundungsmodus kreisförmig in der Arena.

In diesem Versuch wurde erwartet, dass der NeuroRover die Arena erkundet. Dabei sollen weder Reward- und Retractneuronengruppen aktiv sein. Das heißt, dass die farblich markierte Blumentöpfe das Verhalten des Rovers nicht beeinflussen sollen. Wie in Abbildung 3.4 zu entnehmen ist, ist keine Änderung der Synapsengewichte der (KC $\rightarrow_e$ EN) Synapse erkennbar. Das Gewicht blieb während der Simulation konstant. Diese Konstanz war zu erwarten, da es keine Kombination aus einer Farberkennung und einem Reiz der Reward- und Retractneuronen gab, sodass das der Rover weder aversiv noch

appetitiv lernen konnte. Das hatte zur Folge, dass auch keine Verhaltensneuronen, bis auf *Explore*, aktiv war, da die EN-Neuronen aufgrund der nicht stärker gewichteten Synapse (KC $\rightarrow_e$ EN) stark genug erregt wurde, sodass der Schwellenwert nicht erreicht wurde und daher auch keine Spikes ausführen konnte.

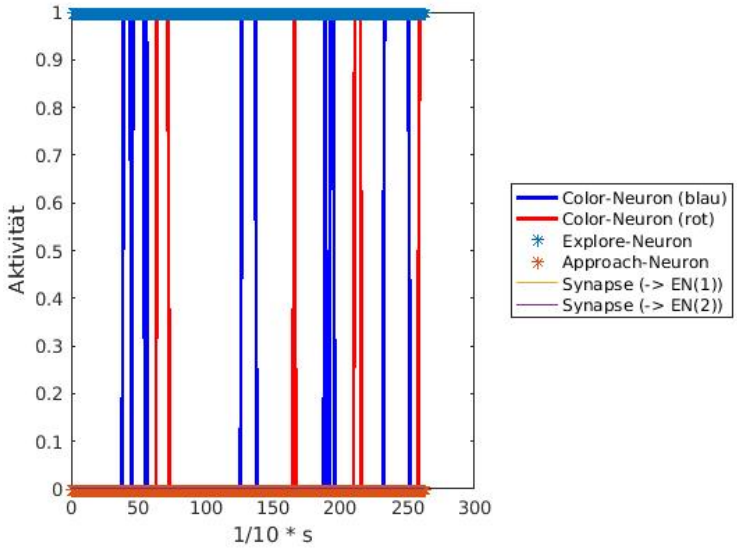


Abb. 3.4: Visualisierte Netzwerkaktivität: Die geloggten Daten wurden geplottet. Der Rover erkundet während der ganzen Simulation die Arena. Das Sehen von Farben wird im Netz wahrgenommen, aber der neutrale Reiz ist nicht stark genug, um das Verhalten zu ändern. Die Synapsengewichten bleiben konstant.

Versuch 2:

Zu Beginn des Versuches wurde der NeuroRover auf einen roten Blumentopf gerichtet. Dabei wurde im Neuronalen Netz das Reward Neuron manuell auf aktiv gesetzt. Der Simulator war für drei Sekunden aktiv, wurde dann gestoppt und anschließend wurde das Reward Neuron auf inaktiv gesetzt. Der Rover wurde in die Mitte der Arena gesetzt und IQR wurde wieder gestartet. Wegen dem Pacemaker-Neuron war das Explore-Ausgabeneuron aktiv gewesen. Daher erkundete der Rover die Arena. Dabei macht er verstärkt eine Drehung nach rechts. Kurzzeitig wurde ein blauer Blumentopf gesehen, was Spikes in der Color-Gruppe auslöste. Allerdings wirkte sich das weder auf das Approach- noch auf das Retract-Neuron aus. Der Rover drehte sich weiterhin tendenziell nach rechts bis ein roter Blumentopf erfasst wurden. Die Color-Neuronengruppe war aktiv. Zudem war das Approach-Ausgabeneuron aktiv. Des Weiteren wurde die EN-Explore-Neuronengruppe gehemmt, was dafür sorgte, dass das Ausgabeneuron Explore nicht mehr aktiv war (siehe Abb. 3.6). Der Neurorover fuhr zum roten Blumentopf

bis er ihn erreichte (siehe Abb. 3.5). In diesem Versuch wurden einige Geschehnisse erwartet:

- Synapsengewichtsänderung,
- das EN-Neuron soll stark genug erregt werden, dass Spikes ausgelöst werden und
- der NeuroRover ändert das Verhalten.

Mit dem Paaren der Neuronenaktivitäten von Color und Reward, ist das Synpasengewicht der Synapse (KC $\rightarrow_e$ EN) gestiegen (siehe Bild). Dieses Phänomen ist nach (Clopath 2012) korrekt, da Endorphine so starke Auswirkungen haben, dass das Synapsengewicht zwischen gepaarten Reize steigt. Das hat zur Folge, dass nun die EN-Neuronengruppe aktiv ist. Wie in der Abbildung 3.7 zu sehen ist, ist nur ein Neuron von zwei aktiv. Der Grund ist, dass das Reward-Neuron mit dem EN(1)-Neuron gepaart ist (siehe Abbildung 3.2) und somit das Synapsengewicht zwischen dem KC- und dem EN(1)-Neuron stärker wurde. Mit dem Erregen und somit das Ausführungen des EN(1)-Neurons, wurde die Approach-Neuronengruppe aktiv und die EN-Explore- Neuronengruppe gehemmt. Das hat zur Folge, dass der NeuroRover sein Verhalten ändert und sich dem roten Blumentopf nähert.

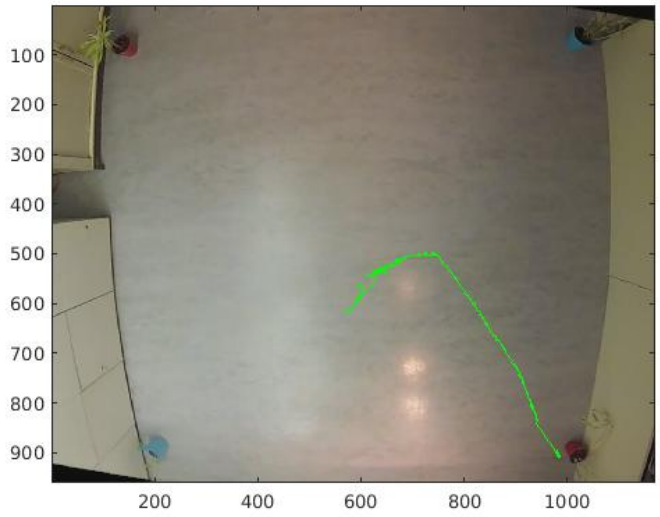


Abb. 3.5: Die getrackte Aufnahme des zweiten Versuches: Der NeuroRover fuhr gradlinig, aber rechtsorientierend bis ein roter Blumentopf gesichtet wurde. Der Rover fuhr anschließend auf ihn zu.

Wie in der Abbildung 3.7 zu sehen ist, wurden auch blaue Blumentöpfe erkannt. Dazu waren keine anderen Neuronen aktiv. Da die blaue Farbe mit keinem anderem Neuron,

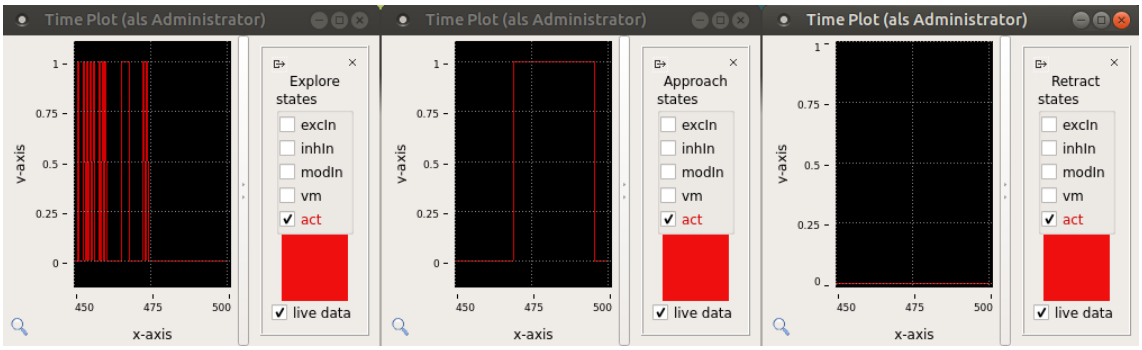


Abb. 3.6: Momentanaufnahme des Versuches: Wie zu sehen ist, wird das Explore-Neuron inaktiv, wenn das Approach-Neuron aktiv ist. Die Retract-Neuronengruppe bleibt ebenfalls inaktiv.

wie Punishment oder Reward, gepaart war, waren die exhabitorische Reize an der EN-Neuronengruppe nicht stark genug gewesen um Spikes auszuführen.

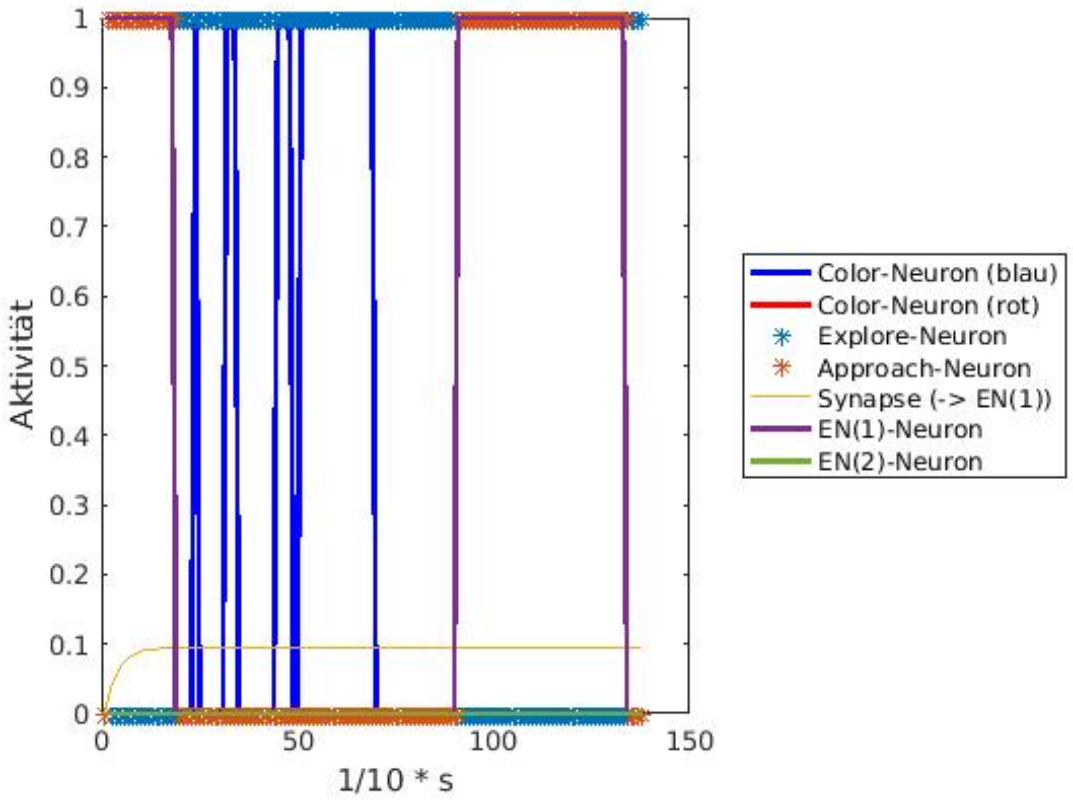


Abb. 3.7: Visualisierte Netzwerkaktivität: Das Color-Neuron (rot) und das Reward-Neuron (nicht zu sehen) sind aktiv. Das hat zu Folge, dass das EN(1)-Neuron und somit das Approach-Neuron aktiv sind. Das Synapsengewicht steigt stark an. Wahrgenommene blaue Farben ändert das Verhaltensmuster nicht. Erst mit der Aktivität des Color-Neuron (rot), wird Explore gehemmt und der Rover geht auf den roten Blumentopf zu.

Versuch 3:

Zu Beginn des Versuches wurde der Neurorover auf einen blauen Blumentopf gerichtet. Dabei wurde im Neuronalen Netz das Punishment Neuron manuell auf aktiv gesetzt. Anschließend war der Simulator für drei Sekunden aktiv, wurde gestoppt und das Punishment Neuron wurde auf inaktiv gesetzt. Der Neurorover wurde in die Startposition gelegt und IQR wurde gestartet. Der Neurorover fuhr zunächst in gerader Richtung, wobei diese etwas nach links geneigt war bis ein roter Blumentopf zu sehen war. Er fuhr sehr kurz auf das Objekt zu, drehte sich aber schnell nach rechts. Bis zu diesem Zeitpunkt waren nur die Color- und Explore-Gruppe aktiv. Der Rover drehte sich solange nach rechts bis ein blauer Blumentopf zu sehen war. Dabei war die Retract-Neuronengruppe aktiv und die En-Explore-Neuronengruppe wurde gehemmt (siehe Abb. 3.9). Der Neurorover machte sofort eine Rückwärtsbewegung, während er um seine eigene Achse drehte bis der blaue Topf nicht mehr zu sehen war. Anschließend war der Rover wieder im Explore-Modus gewesen, sodass die Arena weiter erkundet werden konnte. Sie fuhr zunächst gerade aus, während die Bewegung leicht nach rechts gerichtet war bis der blaue Topf gesehen wurde. Dabei wurde die Bewegung sofort beendet und der Rover wendete sofort rückwärts, rechts geneigt weg. Wieder war die Rectract-Neuronengruppe aktiv. Zum Ende der Versuchsdurchführung befand sich der Rover im Explore-Modus und die Vorwärtsbewegung war linksorientiert (siehe Abb 3.8). Das Sehen des roten Blumentopfes beeinflusste im Neuronalen Netz nur die Color-Gruppe.

Der Versuchsaufbau ähnelte dem 2. Versuch. Der Unterschied ist, dass der NeuroRover auf die blaue Farbe bestraft wird. Mit dem Erkennen der blauen Farbe, wurde korrekterweise die Color-Gruppe (blauer Neuron) aktiv. In Kombination mit der Aktivität vom Punishment-Neuron wurde das EN(2)-Neuron aktiv. Die Begründung dafür ist, dass die Synapse (KC $\rightarrow_e$ EN) stärker gewichtet ist und dass das Punishment-Neuron mit dem EN(2)-Neuron verbunden ist. Anschließend wurde die EN-Neuronengruppe auch dann aktiv, wenn ein blauer Blumentopf gesehen wurde, ohne das Reward- und Punishment-Neuron aktiv war. Mit der Aktivität vom EN(2)-Neuron ist auch die Retract-Neuronengruppe aktiv, denn die beiden Neuronen sind miteinander verbunden (siehe Abbildung 3.2). Das war nach Clopath (2012) und [Spektrum] zu erwarten, weil nach der Kombination von einem neutralen und einem Reiz, der eine negative Auswirkung hat, ein

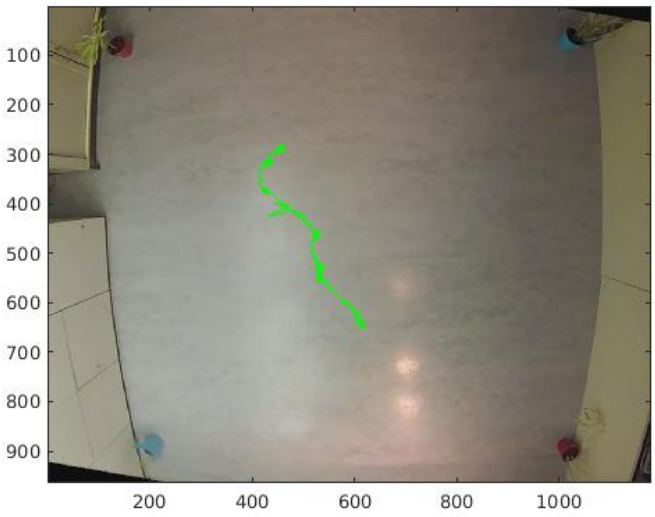


Abb. 3.8: Die getrackte Aufnahme des dritten Versuches: Der NeuroRover fuhr gradlinig, aber linksseitig durch die Arena. Dabei wurde desöfteren nach rechts gedreht. Beim Erkennen eines blauen Blumentopfes, drehte sich der Rover sofort weg.

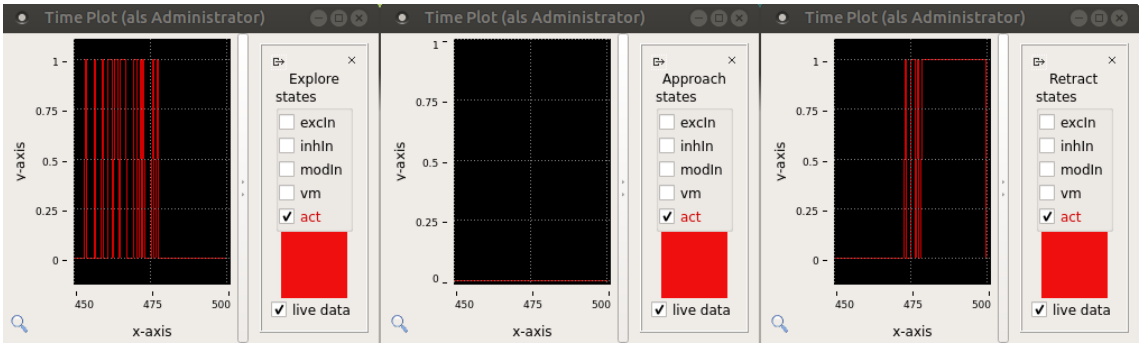


Abb. 3.9: Momentanaufnahme des Versuches: Der Rover erkundete zunächst die Arena bis ein blauer Blumentopf gesichtet wurde. Die Retract-Neuronengruppe war aktiv, während die Explore-Neuronengruppe inaktiv wurde. Das Approach-Neuron löste die ganze Zeit keine Spikes aus

Verhalten auslöst wird, nur noch der neutrale Reiz ausreicht um das gleiche Verhalten hervorzurufen. Mit der Aktivität der EN-Gruppe wird die Explore-EN-Neuronengruppe gehemmt, sodass kein Verhalten außer Retract ausgeführt wird (siehe Abb. 3.9 und 3.10).

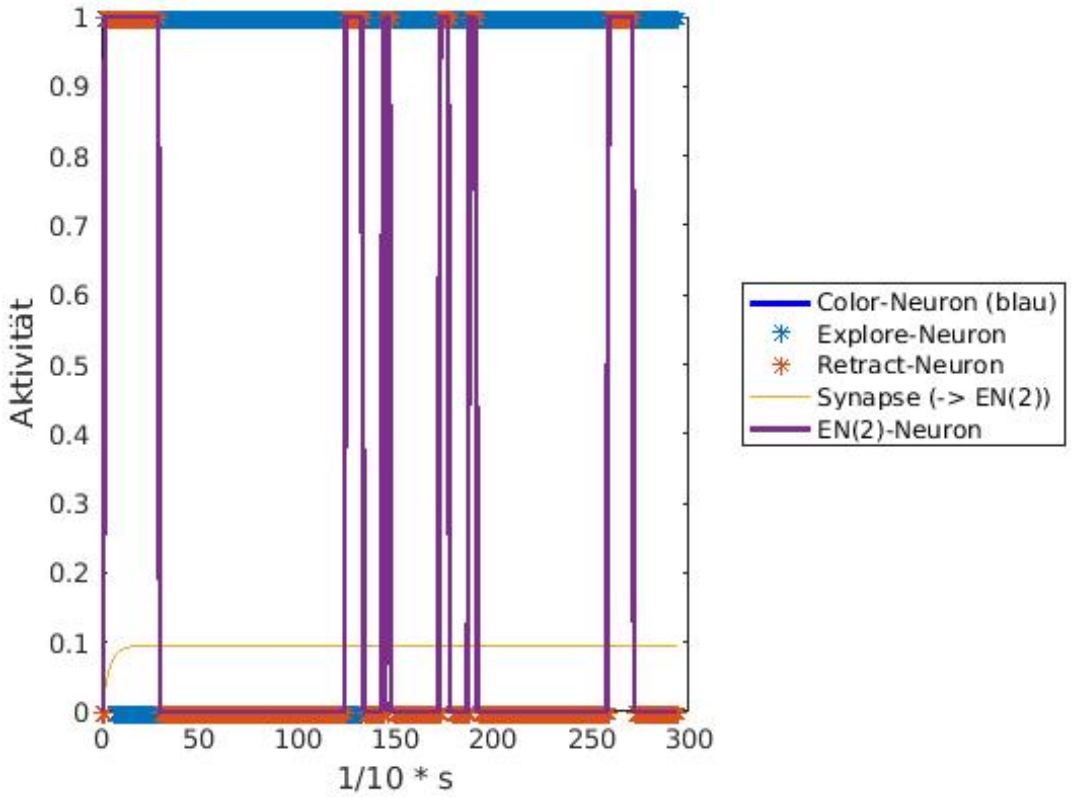


Abb. 3.10: Visualisierte Netzwerkaktivität: Grafische Visualisierung des Loggings: Das blaue Color-Neuron und das Punishment-Neuron (nicht zu sehen) sind gleichzeitig aktiv. Das Synapsengewicht steigt und das EN(2)-Neuron wird aktib und ändert das Verhaltensmuster

Versuch 4:

In diesem Versuch muss der Rover auf beide Farben trainiert werden. Dabei wurde das Reward zunächst aktiv gesetzt und der Neurorover wurde auf den roten Blumentopf gerichtet. Die Simulation wurde für drei Sekunden gestartet, dann gestoppt und das Reward-Neuron auf inaktiv gesetzt. Danach wurde der Rover auf den blauen Blumentopf gerichtet, das Punishment-Neuron auf aktiv gesetzt und anschließend die Simulation für drei Sekunden gestartet. Danach wurde IQR gestoppt, das Punishment-Neuron auf inaktiv gesetzt und der Rover wurde in die Mitte der Arena gestellt. Die Simulation wurde anschließend gestartet. Der Neurorover fuhr geradeaus, während die Neigung rechts gerichtet war. Dies dauerte so lange an, bis ein blauer Topf gesehen wurde. Die Retract-Neuronengruppe war sehr kurz aktiv, denn der Rover drehte sich weiter nach rechts bis das Objekt nicht mehr zu sehen war. Dann drehte sich der Roboter nach links bis der blaue Blumentopf wieder sichtbar war. Die Retract-Neuronengruppe war aktiv

und der Rover macht eine Rückwärtsbewegung, wobei diese nach rechts gerichtet war bis kein Objekt sichtbar war. Anschließend erkundete der Neurorover die Arena in dem er sich rechtsseitig orientiert bewegte. Hin und wieder machte der Rover eine Bewegung nach links oder fuhr in gerader Richtung. Kurz war der blaue Topf sichtbar gewesen, doch der Roboter wandte sich rechtsseitig weg bis ein roter Blumentopf sichtbar war. Die Approach-Neuronen waren aktiv und der Rover fuhr zum roten Topf bis er ihn erreichte (siehe Abb. 3.11).

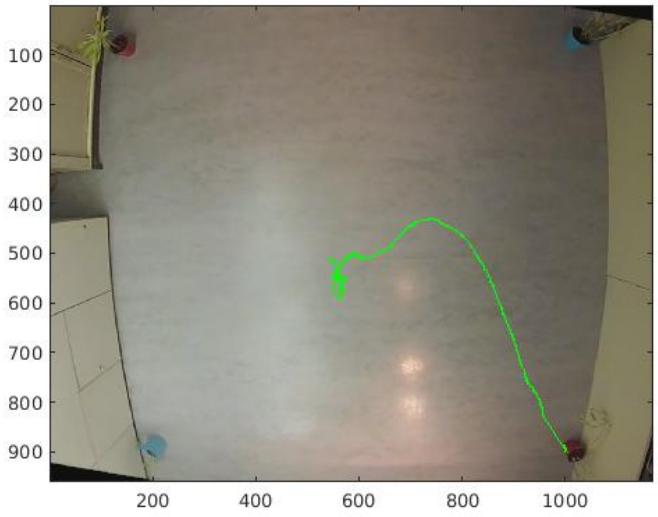
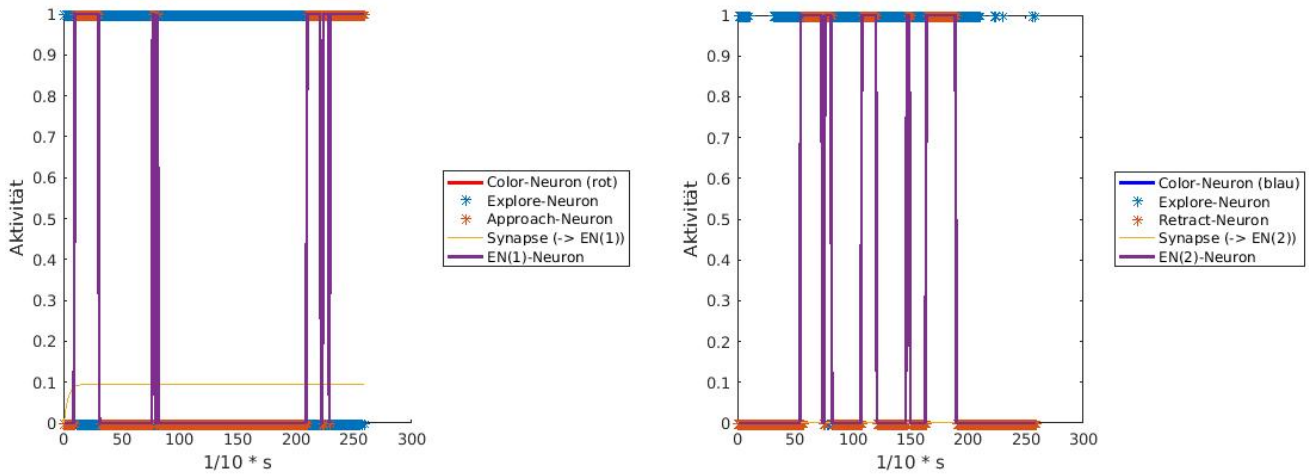


Abb. 3.11: Die getrackte Aufnahme des vierten Versuches: Der NeuroRover fuhr gradlinig, aber rechtsseitig durch die Arena. Als ein blauer Blumentopf gesichtet wurde, drehte sich der Rover weg bis er ein roter Blumentopf gesichtet hat, auf den er schließlich zuging.

Mit der Kombination den Aktivitäten des Reward-Neurons und der Color-Neuronengruppe (das Neuron, das bei gesehener blauer Farbe aktiv wird) wurde das EN(1)-Neuron aktiv und somit auch die Approach-Neuronengruppe (siehe Abb. 3.12a). Ebenfalls wurde mit den Aktivitäten des Punishment-Neurons und der Color-Neuronengruppe (das Neuron, das bei gesehener roter Farbe aktiv wird) das EN(2)-Neuron aktiv und somit auch die Retract-Neuronengruppe (siehe Abb. 3.12b). Außerdem ist dort zu sehen, dass Synapsengewicht, sowohl zum EN(1) und E(2), gestiegen ist. Das erklärt nach Clopath (2012), warum es ausreichte, dass beim Sehen eines blauen Blumentopfes das Retract-Verhalten und beim Sehen eines roten Blumentopfes das Approach-Verhalten ausgelöst wird. Das Neuronale Netz kann somit eindeutig neutrale Reize in Kombination mit einem Reiz, der bestrafend oder belohnend wirkt, unterscheiden, die dann beim

alleinigen Auftreten eines Reizes ein bestimmtes Verhalten auslösen.



(a) Zeitlicher Verlauf des 4. Versuches

(b) Zeitlicher Verlauf des 4. Versuches

Abb. 3.12: Grafische Visualisierung des Loggings: a) Mit dem Paaren der Reize, Color-Neuron (rot) und Reward (nicht zu sehen), stieg das Synapsengewicht und änderte das Verhaltensmuster. Anschließend reichte die alleinige Aktivität des Color-Neurons (rot) aus, das Verhalten zu ändern. Die Verlaufskurve des EN(1)-Neurons "überdeckt"die Verlaufskurve des roten Color-Neurons. Kurzzeitig war das EN(1)-Neuron aktiv, aber zu dieser Zeit drehte sich der Roboter noch, sodass die rote Farbe nicht mehr gesehen wurde. b) Mit dem Paaren der Reize, Color-Neuron (blau) und Punishment (nicht zu sehen), stieg das Synapsengewicht nur sehr leicht und änderte dennoch das Verhaltensmuster. Anschließend reichte die alleinige Aktivität des Color-Neurons (blau) aus, das Verhalten zu ändern. Die Verlaufskurve des EN(2)-Neurons "überdeckt"die Verlaufskurve des blauen Color-Neurons.

Kapitel 4

4 Diskussion und Ausblick

Es wurde ein Neuronales Netz entworfen, dass das Insektengehirn bezüglich des assoziierten Lernens simulieren sollte. Dafür wurde ein Roboter, den NeuroRover, verwenden, der von dem Neuronalen Netz gesteuert wird. Der NeuroRover wurde mit der HaViMo2.0 ausgestattet und konnte somit die Umgebung, mit der Beschränkung, dass nur die Farben der Umgebung erkannt werden, erkunden. Die HaViMo2.0 wurde so kalibriert, dass sie blaue und rote Gegenstände erkennen kann. Das Arduino WiFiShield ermöglichte eine drahtlose Kommunikation zwischen dem Roboter und dem Neuronalen Netz. Nach dem Setup wurde eine quadratische Arena gebaut, die in den Eckpunkten jeweils einen Blumentopf enthielt. Diese Blumentöpfe waren entweder blau oder rot. Der Blumentopf eines Eckpunktes hat eine andere Farbe als seine Nachbarn, sodass diagonal zueinander liegende Blumentöpfe die gleiche Farbe hatten. Es wurden vier Versuche durchgeführt, die sich darin unterschieden, ob der NeuroRover aversive oder appetitive Erfahrungen gegenüber einer bestimmten Farbe gesammelt.

Im ersten Teil hat der Rover zu Beginn keine Erfahrungen mit einer Farbe gesammelt. Dieser fuhr im Erkundungsmodus in der Arena. Das Erkunden beschränkt sich auf zufällige Bewegungen des Roboters, das heißt, dass der Motor nicht deterministisch agierte. Die HaViMo2.0 erkannte die Farben in der Arena. Das Verhalten änderte sich jedoch nicht. Nach Izhikevich (2007) und Savada et al (2007) fehlte ein Reiz, der entweder belohnend oder bestrafend ist, der eine Änderung im Verhaltensmuster hervorruft. Die Synapsengewichte änderten sich ebenfalls nicht. Nach Clopath (2012) kann man dieses Ereignis erklären, in dem man sich den Versuchsaufbau näher betrachtet. Der Rover wurde naiv in die Arena gesetzt. Das heißt, dass er zuvor nicht keine Erfahrungen gesammelt hat, sodass keine Verknüpfung zweier Reize stattfinden konnte.

Im zweiten Teil wurde der NeuroRover zunächst für zwei Sekunden auf die rote Farbe belohnt, in dem die HaViMo2.0 den roten Blumentopf erkannte und währenddessen

das Reward-Neuron aktiv war. Währenddessen änderten sich die Synapsengewichte. Anschließend wurde der Rover in die Arena platziert und startete im Erkundungsmodus. Blaue Blumentöpfe wurden erkannt, aber es zeigten sich keine Verhaltensänderungen. Als ein roter Blumentopf gesichtet wurde, ging der NeuroRover auf den Blumentopf zu. Nach Savada et al (2011) und Izhikevich (2007) kann erklärt werden, dass das Neuronale Netz beide Reize verknüpfen konnte, sodass anschließend ein neutraler Reiz ausreichte, das Verhaltensmuster zu ändern. Die Belohnung die rote Farbe gesehen zu haben, erhöhte das Synapsengewicht nach Clopath (2012) und Savada et al (2011).
Der dritte Teil startete wie der zweite Teil. Der Unterschied war, dass der Rover bestraft wurde, als der blaue Blumentopf gesichtet wurde. Auch hier veränderten sich die Synapsengewichte. Rote Blumentöpfe hatten keine Auswirkungen auf das Verhalten und als blaue Farben gesichtet wurde, drehte sich der Rover sofort weg. Der Sichtkontakt sollte vermieden werden. Die rote Farbe war dem NeuroRover unbekannt. Somit war das Erkennen der roten Farbe ein neutraler Reiz, der nicht stark genug war, das Verhaltensmuster zu ändern nach [Spektrum]. Die blaue Farbe meidete der Rover hingegen. Die Verhaltensmusteränderung ist nach Savada et al (2011) und Izhikevich (2007) zu erklären, dass der bestrafende Reiz mit dem Erkennen der blauen Farbe gepaart wurde. Somit stiegen nach Clopath (2012) die Synapsengewichte, die dafür sorgten, dass mit dem Erkennen der blauen Farbe, als neutraler Reiz, das Verhaltensmuster geändert wurde.
Im vierten Teil wurde der Rover auf die rote Farbe belohnt und auf die blaue Farbe bestraft. Auf jede erkannte Farbe wurde das Verhaltensmuster geändert. Als der blaue Blumentopf erkannt wurde, drehte er sich weg. Beim Sehen eines roten Blumentopfes ging der Roboter auf ihn zu. Die Versuchsdurchführung hat gezeigt, dass der NeuroRover beim Sehen beider Farben jeweils das Verhalten änderte (siehe Abb. 3.11). Nach Savada et al (2011), Izhikevich (2007) und [Spektrum] wurden die neutralen Reize, das Sehen von einer roten oder Blauen, beim Paaren mit einem bestrafenden oder belohnenden Reiz so stark verstärkt, dass das alleinige registrieren vom neutralen Reiz eine Verhaltensänderung auslöste. Dementsprechend schwerer wurden nach Clopath (2012) die Gewichte der Synapsen, die zum EN(1)-Neuron und nach EN(2)-Neuron führen. Überraschend war jedoch, dass die Gewichte der Synapsen, die zu, EN(2) führen, nur sehr leicht stiegen. Dennoch waren die Gewichte schwer genug nach Savada et al

(2011), Izhikevich (2007) und [Spektrum] den neutralem Reiz, das Sehen der blauen Farben, stark genug zu verstärken.

Die Versuche haben gezeigt, dass ein Neuronales Netz entworfen werden kann, dass das Assoziierte Lernen simulieren kann. Dennoch ergab es einige Probleme und Hindernisse bei der Ausführung der Versuche. Zum einen waren die Bewegungsrichtungen des NeuroRovers im Erkundungsmodus nur probabilistisch. Das hatte zur Folge, dass es vorkam, dass der Rover gegen die Wand der Arena fuhr. Ein Neustart des Versuches war die Folge. Man könnte den NeuroRover mit einem Sensor für die Distanzerkennung ausstatten, die nur aktiv ist, wenn der NeuroRover sich im Erkundungsmodus befindet. Somit kann ausgeschlossen werden, dass der Roboter gegen die Wand fährt. Außerdem bereitete die HaViMo2.0 erhebliche Probleme. Diese reagierte sehr empfindlich auf Spannungsdifferenzen, sodass die Kamera abstürzte. Dies war erkennbar, in dem im Neuronalen Netz keine Farben erkannt wurden, obwohl die HaViMo2.0 auf einen Blumentopf gerichtet war. Die Fehlerquellen sollten weitesgehend ausgeschlossen werden. Daher wurde entschieden, dass die Reward- und Punishmentneuronen manuell aktiviert wurden. Leider konnten die Spikeaktivitäten dieser Neuronen nicht geloggt werden, da Spikeaktivitätswerte nur dann auch als aktiv geloggt wird, wenn die exhabitorisch eingehende Signale den Schwellenwert erreichen (siehe Abb. 4.1).

```
1  excitation[0] *= excGain;
2    inhibition[0] *= inhGain;
3
4    // Calculate membrane potential
5    vm[0] *= vmPrs;
6    vm[0] += excitation[0]; // see 1 * excGain;
7    vm[0] -= inhibition[0]; // see 1 * inhGain;
8    .....
9    activity.fillProbabilityMask(probability);
10   activity[0] *= vm[0];
11   activity[0][vm[0] < threshold] = 0.0;
```

Abb. 4.1: Ausschnitt des NeuronLinearThreshold-Neurons: Die eingehenden exhabitorische Signale erhöhen das Membranpotential. Falls dieses Potential groß genug ist und den Schwellenwert erreicht, wird activity nicht auf 0.0 zurückgesetzt. Beim Reward- und Punishment-Neuron wurden die Aktivitäten manuell in der Simulation aktiviert und wurden daher nicht geloggt.

Das wäre der Fall gewesen, wenn Sensoren angebracht wären, die bei Reizen Signale an das Neuronale Netz senden. Der NeuroRover hat einen integrierten Lichtsensor,

aber auf ihm liegt der Arduino WiFi-Shield. Mit dem Anbringen von zwei weiteren Sensoren könnten auch die Reward- und Punishmentneuronen auch korrekt geloggt werden. Ein weiteres Problem ist, dass die Simulation für Neuronale Netzwerke IQR sehr rechenintensiv ist. Die Software wurde auf einem Intel i3-4030U, der sehr schnell überfordert war. Eine Skalierung des Netzwerkes ist auf einem nicht-leistungsstarken Computer nur bedingt möglich.

Nichtsdestotrotz bietet IQR viel Potential für das Erstellen und Simulieren von Neuronalen Netzwerken. Mit dem Feature, dass eigene Synapsen- und Neuronenmodelle entwickelt und integriert werden können, macht es die Software sehr skalierbar. Dass zusätzliche Module geschrieben, integriert und somit Rovers gesteuert werden können, ermöglicht es mehrere biologische Hypothesen zu untersuchen und zu simulieren, damit wir unser Verständnis für das Gehirn verbessern können.

Kapitel 5

5 Literaturverzeichnis

Bernardet, Ulysses ; Verschure, Paul F. M. J.: iqr: A Tool for the Contruction of Multi-level Simulation of Brain and Behaviour, 2010

Bernardet, Ulysses: iqr User-defined Types Documentation, December, 2012

Bitterman, M.E.; Menzel, R.; Fietz, A and Schäfer, S: Classical conditioning of proboscis extension in honeybees (apis mellifera). Journal of comparative psychology, 97(3):107-119, 1983

Cassenaer, Stijin ; Laurent, Gilles: Conditional modulation of spike-timing- dependet plasticity for olfactory learning. Nature 482, 47?52 (02 February 2012)

Clopath, Claudia: Synaptic consolidation: an approach to long-term learning, Cognitive Neurodynamics 6(3):251-7 · June 2012

Helgadóttir, Lovisa I. ; Haenicke, Joachin ; Landgraf, Tim ; Rojas, Raul ; Nawrot, Martin: Conditioned behavior in a robot controlled by a spiking neural network. 6th International IEEE EMBS Conference on Neural Engengeering, San Diego, USA, Nov 5-8, 2013

Izhikevich, Eugine M.: Solving the Distal Reward Problem through Linkage of STDP and Dopamine Signaling. The Neurosciences Institute, 10640 John Jay Hopkins Drive, San Diego, CA 92121, USA, 2007

Jayawan, H. B. W. and Dudek: Analogue CMOS Circuit Implementation of a Dopamine Modulated Synapse (2011)

Krause, C.: Kreditwürdigkeitsprüfung mit Neuronalen Netzen, Düsseldorf: IDW, S. 37,1993

Krichmar, J.L.: Neurorobotics. Scholorpedia. 3(3):1365,2008

Kriesel, D. Ein kleiner Überblick über Neuronale Netze (2005)

Maass, W.: Lower bounds for the computational power of networks with spiking neurons, Neural Computation, 8:1-40, 1996

Neurosci, J.: A Computational Framework for Understanding Decision Making through Integration of Basic Learning Rules. 2013 Mar 27; 33(13): 5686?5697.

Ruf, B.: Computing and Learning with Spiking Neurons - Theory and Simulations, 1998

Sadava, D. ; Hills, D. M. ; Heller, H. C. ; Berenbaum, M. R.: Purves Biologie. Spektrum Verlag, 2011

Schultz, W.: Predictive Reward Signal of Dopamine Neurons. Journal of Neurophysiology Published 1 July 1998 Vol. 80 no. 1, 1-27 DOI, 199

Strecker, S.: Künstliche Neuronale Netze - Aufbau und Funktionsweise, 2010

Wijekoon, J. H. and Dudek, P. Analogue cmos circuit implementation of a dopamine modulated synapse. In Circuits and Systems (ISCAS), 2011 IEEE International Symposium on (pp. 877-880). IEEE. [VLSI implementation of the reward-based STDP rule as in Izhikevich 2007]

.[Arduino], Online unter: https://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

[Stand: 13.11.2016]

.[HaViMo2.0], Online unter. http://www.havisys.com/?page_id=8 [Stand. 11.11.2016]

.[RobotShop]: DFRobotShop Rover - USER GUIDE Rev 2.4.
http://www.robotshop.com: RobotShop

.[Spektrum], Online unter: http://www.spektrum.de/lexikon/neurowissenschaft/assoziatives-lernen/964 [Stand: 12.11.2016]

.[WiFi Shield]: https://www.dfrobot.com/wiki/index.php/WiFi_Shield_V2.1_For_Arduino_(SKU:TEL

Kapitel 6

6 Anhang

Tim Landgraf stellte mir zum Tracken der Aufnahmen sein Programm, das in Matlab geschrieben wurde, zur Verfügung. Sein Tracking-Programm setzte allerdings voraus, dass der NeuroRover zwei Markierungspunkte hat. Ich habe den Roboter lediglich nur mit einem Streifen Papier markiert. Das hatte zu Folge, dass beim Tracken mehrere "Pseudotreffer"gefunden wurden, sodass deren Mittelpunkte errechnet und anschließend skizziert wurden. Der Grund für mehrere Treffer war, dass im Programm ein Intervall vom RGB-Farbraum eingegeben werden sollte, wonach gesucht werden sollte.
Ich habe daher ein eigenes Programm geschrieben, das einige Code-Fragmente von Tims Programm enthält und wie folgt funktioniert:
Es wird zunächst im Bildmittelpunkt geschaut, ob dort die gesuchte Farbe befindet, da der Rover immer in der Mitte der Arena befand zum Anfang eines Versuches. Falls keine passende Farbe gefunden wurde, vergrößerte das Programm iterativ seinen Suchradius. Falls aber die gesuchte Farbe gefunden wurde, wird der Punkt später geplottet und als neuer Startpunkt für das Suchen der Farbe gewählt. Da der Rover nicht sehr schnell fährt, muss die gesuchte Farbe im nächsten Frame unmittelbar in der Nähe sein. Dies wird iterativ durchgeführt bis alle Frames bearbeitet wurden.

```
1 %track.m
2
3 %RGB Intervall
4 channel1Min = 50;
5 channel2Min = 25;
6 channel3Min = 18;
7 channel1Max = 125;
8 channel2Max = 113;
9 channel3Max = 70;
```

```
10
11 videoFile       = '/home/oliver/Rover/Rover4.avi';
12
13 % structure elements for erosion/dilation
14 se1 = strel('square',15);
15 se2 = strel('square',5);
16 se3 = strel('square',3);
17
18 %Startsuchpunkt
19 rover = [600 600];
20
21 v = VideoReader(videoFile);
22 figure
23 list = [];
24 counter = 0;
25 frames = 0;
26 while hasFrame(v)
27     frames = frames + 1;
28     video = readFrame(v);
29     if counter < 1594
30         counter = counter +1;
31     else
32      img = video(:,:,:,1);
33      imgBinary  = ( img(:,:,1) >= channel1Min ) & (img
           (:,:,1)...
34                          <= channel1Max) & ...
35                 (img(:,:,2) >= channel2Min ) & (img
                      (:,:,2)...
36                  <= channel2Max) & ...
37                 (img(:,:,3) >= channel3Min ) & (img
                      (:,:,3)...
38                  <= channel3Max);
```

```
39
40          % morph. operations
41          img3    = imerode(  imgBinary,  se3);
42          img3    = imdilate( img3,       se1);
43          img3    = imerode(  img3,       se2);
44
45              % find connected components
46              CC      = bwconncomp(img3);
47          %find centroids of CCs
48          S       = regionprops(CC,'Centroid');
49
50          % Unter dem gefundenen Centroid muss sich der Rover
                befinden
51          if length(S) > 0
52              %Position des Rovers
53              rover = getPoint(rover,S,img);
54              list = [list;rover];
55              imagesc(img);
56              hold on;
57              plot(rover(1), rover(2), '*');
58              pause(0.01);
59          end
60          %aktualisiere aktuelle Suchposition
61          position = rover;
62          %Debug
63          %image( video);
64          %pause(0.01)
65      end
66  end
67
68  %Lade Hintergrundbild und plotte
69  img = imread('arena.jpg');
```

```
70 imagesc(img);
71 hold on;
72 plot(list(:,1), list(:,2), 'g'); hold on;
```

```
1  function y = getPoint(position,S,img)
2
3  flag = 1;
4  px = position(1);
5  py = position(2);
6  i = 1;
7
8  while flag == 1
9
10     for j = 1:length(S)
11         % Ermittle Punkt
12         sx = round(S(j).Centroid(1),0);
13         sy = round(S(j).Centroid(2),0);
14        if (px == sx && py == sy)
15
16            position = [sx sy];
17            flag = 0;
18            break;
19            %vergrößere Suchradius
20        elseif  (px-i <= sx && sx <= px+i) && (py-i <= sy
              && sy <= py+i)
21            position = [sx sy];
22            flag = 0;
23            break;
24        end
25
26
27     end
28     i = i +1;
```

```
29
30      % Punkt ist zu weit weg => ist wohl nicht der Rover
31      if i > 100
32
33          break;
34      end
35
36      end
37
38
39
40  y = position;
41  end
```