iqr

Eine Präsentation von Laura Jozefini.
Thema: Neuronale Netze (2010)

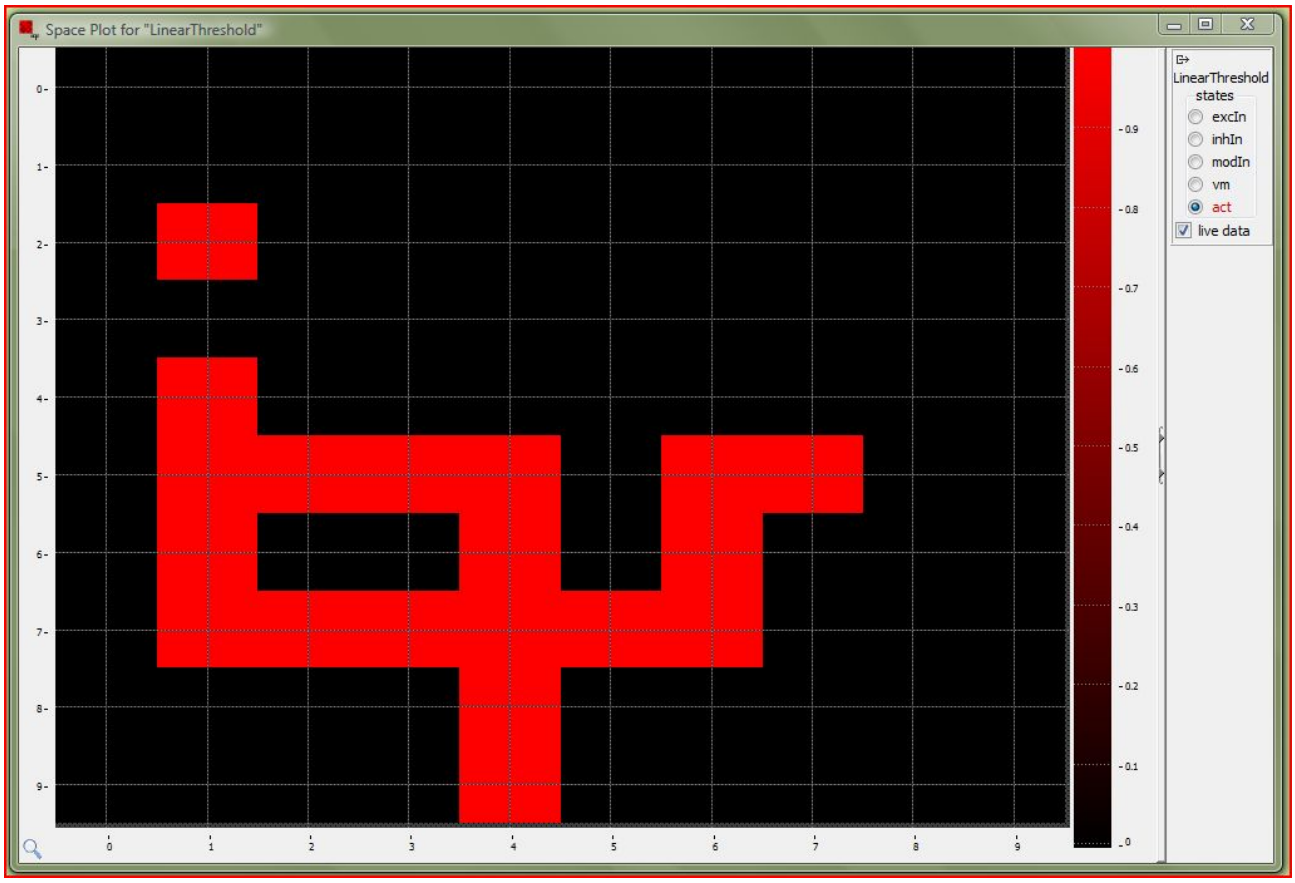


Bild 0a

Allgemeines

Die iqr-Software ist eine open-source Software (GNU Public Licence) zum kreieren und simulieren von Neuronalen Netzen im großen Stil.
Ihre Vorteile sind das graphische Interface, das sehr leicht zu bedienen ist und somit die Erstellung der Netzwerke erleichtert und den Prozess übersichtlich hält. Die Software gibt einem die Möglichkeit einer graphischen on-line Kontrolle sowie die Gelegenheit, Parameter während der laufenden Simulation zu verändern. Auch kann sich der Benutzer jeder Zeit eine on-line Visualisierung und Analyse der eingegebenen Daten zeigen zu lassen.
Ein weiterer Vorzug des Programms ist die Gegebenheit, Roboter und auch Kameras an die Software anzuschließen und in die Netze einzubinden. Es sind sogar schon einige vordefinierte Modelle dafür vorhanden.
Auch gibt iqr einem die Möglichkeit, selbst eigene Neuronen und Synapsen für das eigene Neuronale Netz zu erstellen.
Natürlich gibt es jedoch auch Nachteile:
Zum einen ist die Darstellung der Modelle nicht allzu detailliert und auf das Nötigste beschränkt.
Es kommt darauf an, dass man die Verbindungen zwischen den Neuronen erkennen kann, nicht darauf, dass man sich durch die schöne Gestalten ablenken lässt. Außerdem laufen in iqr die Simulationen zyklisch ab. Das bedeutet, das alle Elemente pseudo-parallel geupdated werden und nicht darauf geachtet wird, ob sie überhaupt aktuell aktiv sind.

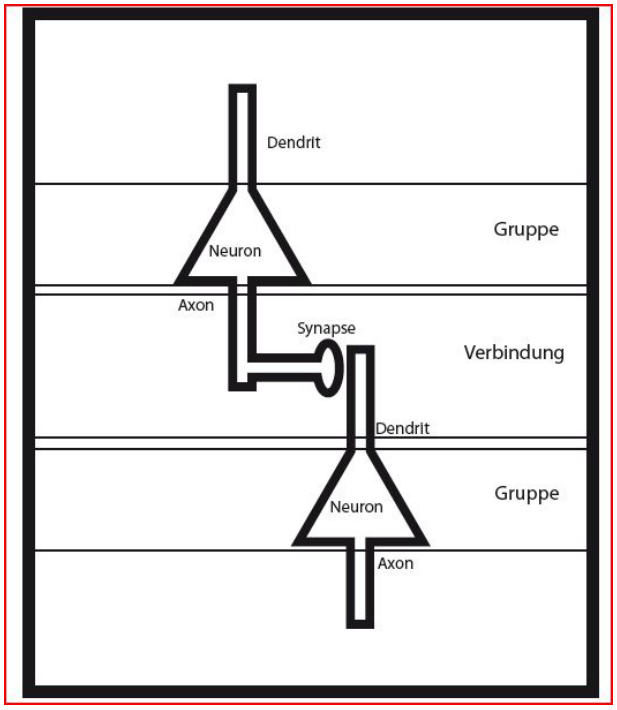


Bild 0b

Im Großen und Ganzen liegt iqr irgendwo zwischen den high- und den low-level Simulatoren von Neuronalen Netzen.
Durch die einfach, übersichtliche graphische Beschaffenheit kann man schnell komplexe Systeme aufbauen. Andererseits ist die Konstruktion auf biologisch realistische Modelle beschränkt.
Für mich persönliche ist das jedoch eher ein Vorteil als Nachteil, da ich die Systeme verständlicher finde, wenn sie auf realer Biologie basieren.

Ein weiterer, wie ich finde, zu erwähnender Nachteil der iqr-Software ist das unzureichende Angebot an Nachschlagemöglichkeiten, was das Verständnis betrifft.
Es wird nur ein Basic- und ein Advanced-Tutorial von den Entwicklern angeboten, das zwar zeigt, wie man die Software bedient, allerdings keine Zusammenhänge erklärt. Selbst wenn man sich durch beide durchgearbeitet hat, weiß man immer noch nicht wirklich, warum man Neuronen nun so und so eingestellt und verbunden hat.
Ein gravierender Nachteile, denke ich.

Modelle

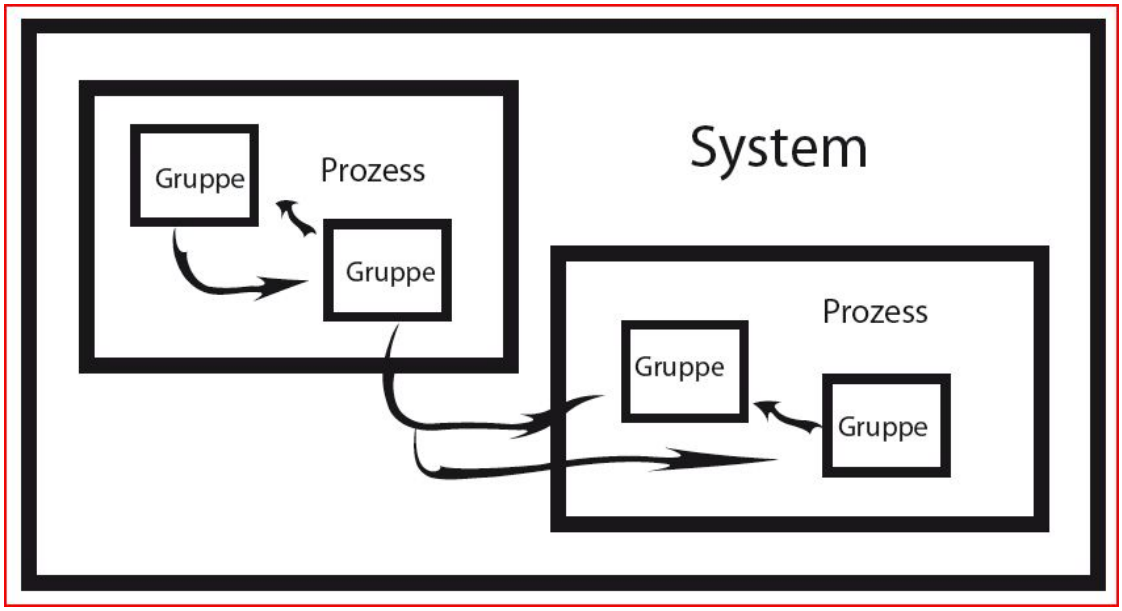


Bild 1a

Das Neuronale Netzwerk erstellt man in iqr auf drei verschiedenen Levels.
Das oberste Level, das Top-Level ist das System, in dem man neue Prozesse und Verbindungen anlegt. Unter den Prozessen liegen dann die Gruppen,
die wiederum die Neuronen enthalten, die man verbinden und verändern kann.

Da es bei Neuronalen Netzen meist so ist, das sie aus großen Modellen mit tausenden kleinen, verzweigten Subsystemen bestehen bei denen man leicht die Übersicht verliert, ist es in iqr dem Benutzer möglich Prozesse zu exportieren und in andere Systeme wieder zu importieren.
Sie sind dann miteinander verbunden, existieren aber in unterschiedlichen Dateien.
An die Systeme kann man Kameras, Khepera, e-Puck Roboter, Lego Mindstorm und die serial VISCA pan-tilt - eine pan-tilt für Kameras- anschließen.

Benutzung

Der Start

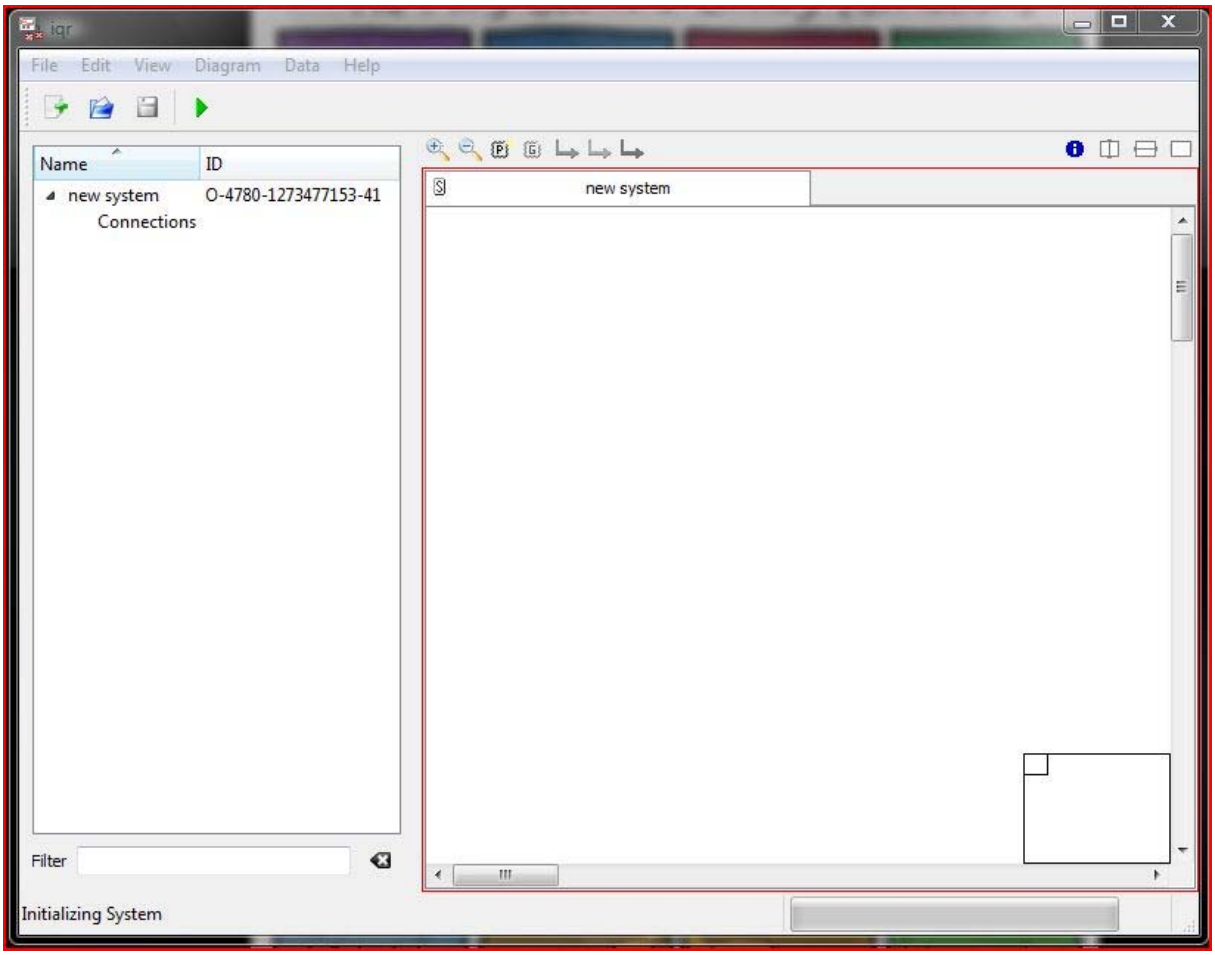
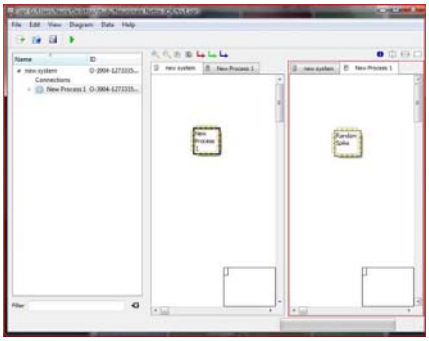


Bild 1b

Das ist der Startbildschirm iqr's (1b) und damit das erste, was man sieht, wenn man das Programm öffnet. Auf der linken Seite steht das System, in dem man sich gerade befindet. Rechts findet man das System selbst –das hier im Augenblick noch leer ist. Dort kann man Prozesse und deren Verbindungen (exitatorisch, inhibitorisch, modular) anlegen.



Hier links wurde genau das getan, und man hat, zur besseren Übersicht, eine vertikale Teilung des Fensters benutzt. Eine horizontale Teilung wäre auch möglich. So kann man gleichzeitig den derzeitigen Prozess mit seinen aktuellen Gruppen betrachten und auch zwischen verschiedenen Gruppen und Prozessen hin- und herschalten, was das einbauen von Verbindungen sehr erleichtert.

Bild 1c

Eigenschaften

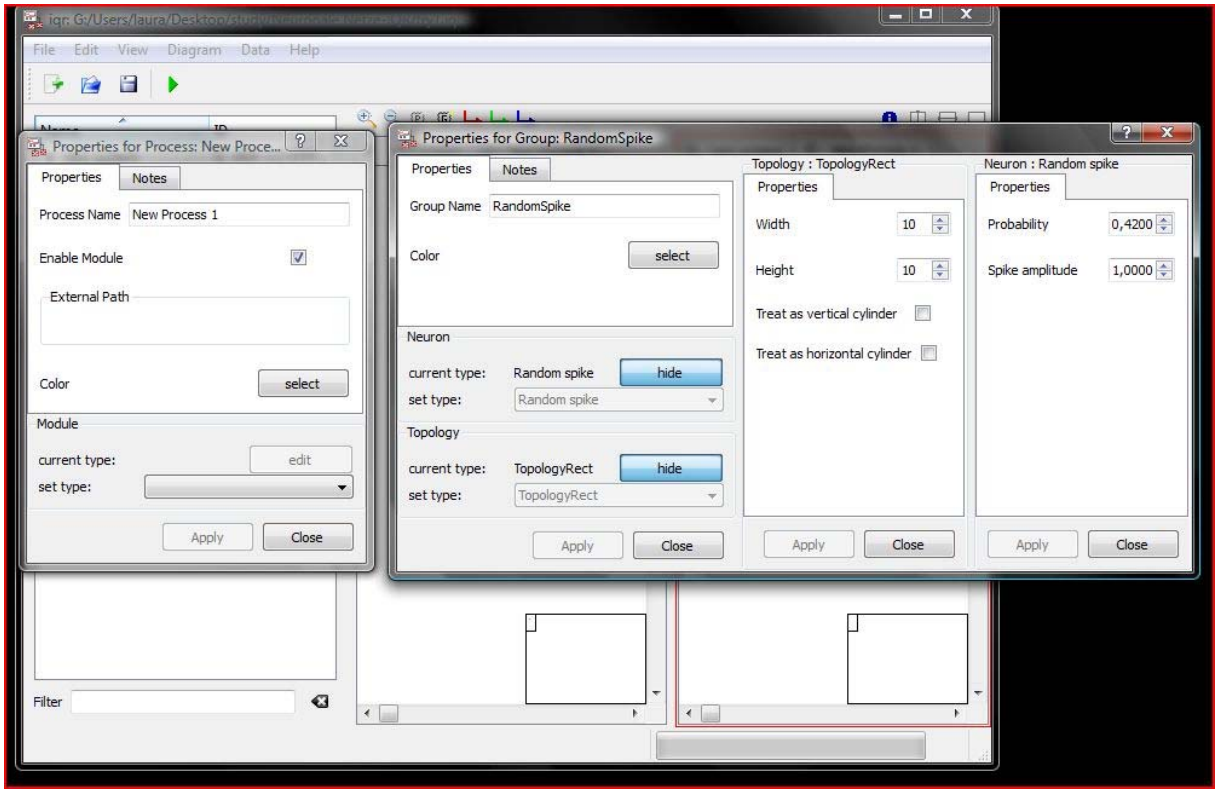


Bild 2

Bild 2 zeigt die veränderbaren Eigenschaften von Prozessen (links) und Gruppen (rechts).
Bei einem Prozess kann man z.B. den Namen und die Farbe ändern, mit dem er angezeigt wird, damit man die Prozesse leichter auseinanderhalten kann. Außerdem kann man das Module ändern. Das sind die, zuvor schon erwähnten, vorgefertigten Modelle, mit denen man dem Prozesse sagt, mit welchen angeschlossenen Gerät er sich verbinden soll.
Anschließend kann man die dafür zutreffenden Eigenschaften verändern.
Die in iqr vorhandenen Modelle sind das Epuck Module, Joystick Module, der OpenCV mpeg Video Player und das Video Module.

Bei den Gruppen kann man, wie auch bei den Prozessen, Name und Farbe ändern. Zusätzlich sucht man aber auch die Art des Neurons und die Topologie aus.
Mit Topologie ist die Anordnung und Dichtung der Neuronen gemeint, die hier im Default (“TopolgyRect“) einer Gitterstruktur entspricht, mit jeweils einem Neuron pro Feld. Die einzige Änderungsmöglichkeit ist die zu “TopolgySparse“, denn dort kann man selbst entscheiden, welche Felder besetzt und welche unbesetzt sind.
Die Anordnung fängt mit 1,1 oben links an. Die y-Achse geht runter, und die x-Achse nach rechts.

Bei den Neuronentypen kann man zwischen “Integrate & Fire“,“Linear Threshold“,“Pyramidal with Apical Shunt“ ,“ Random Spike“ und “Sigmoid“ entscheiden.
Die Eigenschaften der Neuronen kann man dann in Bezug auf Input, Membran, sowie Clip und Spike Potential verändern. Nicht jeder Neuronentyp hat jedoch die gleichen Eigenschaften und damit Veränderungsmöglichkeiten. Von den vorgegebenen Neuronentypen beginnt man, soweit ich weiß, immer damit, sich eine RandomSpike-Gruppe von Neuronen anzulegen. Dieser Typ ist der einzige, der von sich aus feuert.
Alle anderen Gruppen benötigen einen Input, den dann die RandomSpike-Gruppe liefert.

Plots

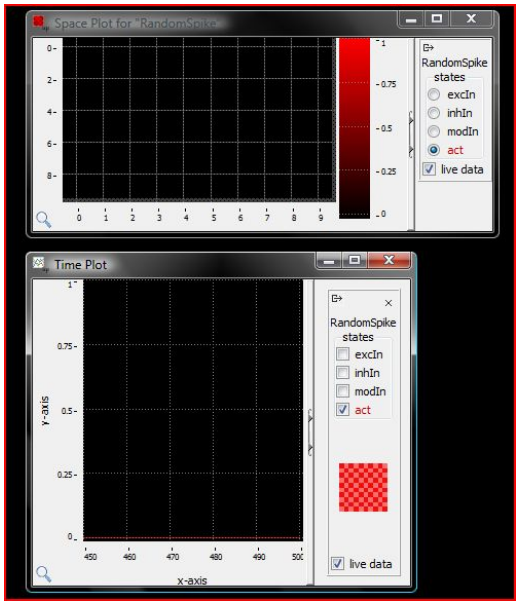


Bild 3a

In Bild 3a sind der Space Plot (oben) und Time Plot (unten) der RandomSpike-Neuronen dargestellt (hier: inaktiv).
Dies sind die wichtigsten Plots während einer Simulation.
Wenn man die Simulation gestartet hat, kann man hier das Verhalten der Neuronen beobachten.
Die Plots zeigen das Spikeverhalten der Neuronen im Bezug zur Struktur und zur Zeit an. Im schwarzen Plotarea werden die Spikes angezeigt, im grauen State Panel recht daneben kann man sich aussuchen, was gezeigt werden soll.

Man kann sich also aussuchen, ob man die gesamte Live-Simulation (“act“) oder nur die exitatorischen/ inhibitorischen/ modularen Inputs betrachten und zeigen lassen will.
Zudem kann man sich zusätzlich auch nur bestimmte Bereiche anzeigen lassen. Dazu wählt man den Bereich im Plotarea aus und zieht ihn auf das State Panel. Man erhält dann ein zweites Plotarea (nur bei Space Plot) indem der ausgewählt Bereich und dessen Verhalten gezeigt wird.

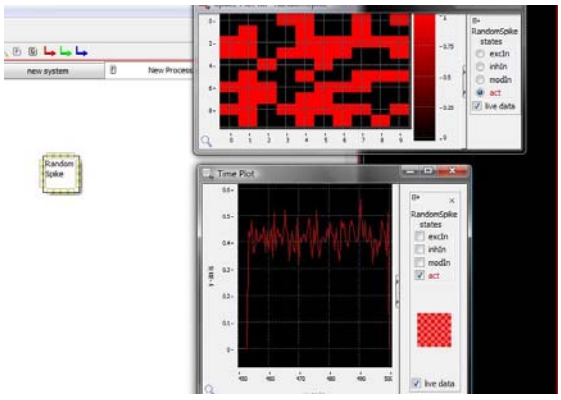


Bild 3b

Bild 3b zeigt die die Simulation der aktivierten RandomSpike-Neuronen im Space Plot (oben) und Time Plot (unten).

Neben den Space und Time Plots gibt es auch noch einen dritten Plot, den “Connection Plot“. Dieser visualisiert die statischen und dynamischen Eigenschaften einer ausgewählten Verbindung.

Verbindungen

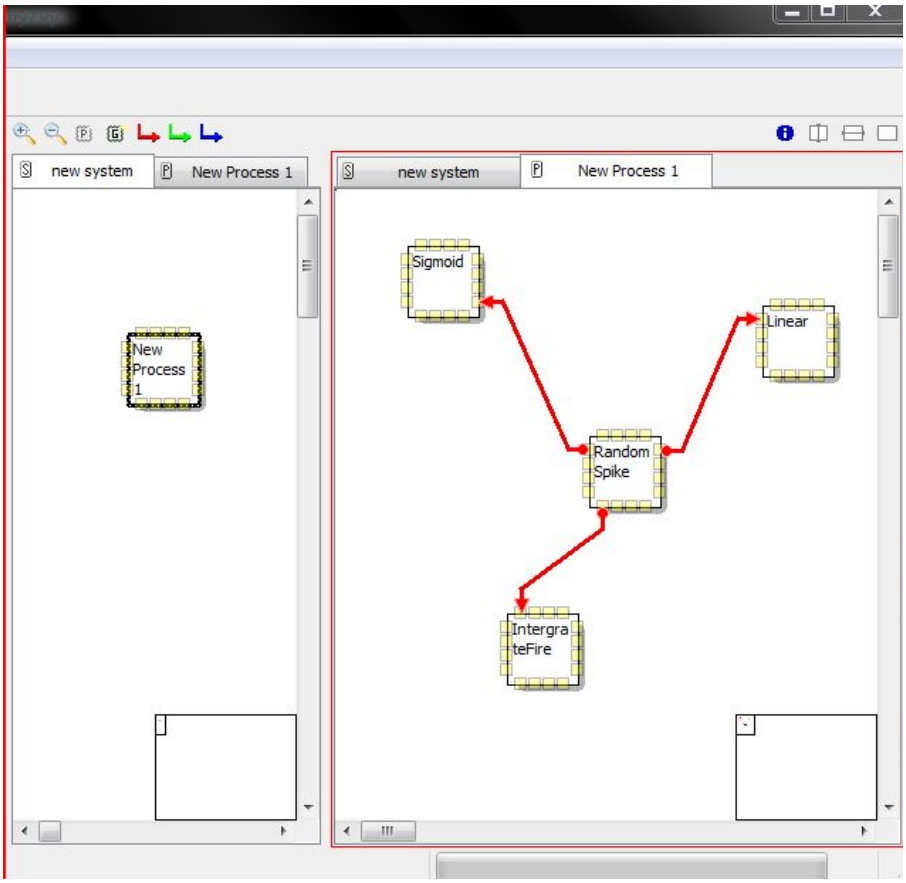


Bild 4a

Einen Prozesse mit vier verbundenen Gruppen sieht man in Bild 4a.
Hier ist eine RandomSpike-Neuronengruppe durch exitatorischen Verbindungen mit einer Sigmoid-, einer LinearThreshold- und einer Integrate&Fire-Gruppe verbunden und gibt ihnen einen Input.

In Bild 4b sieht man noch einmal die benutzen Gruppen und ihre verschiedenen Eigenschaften.

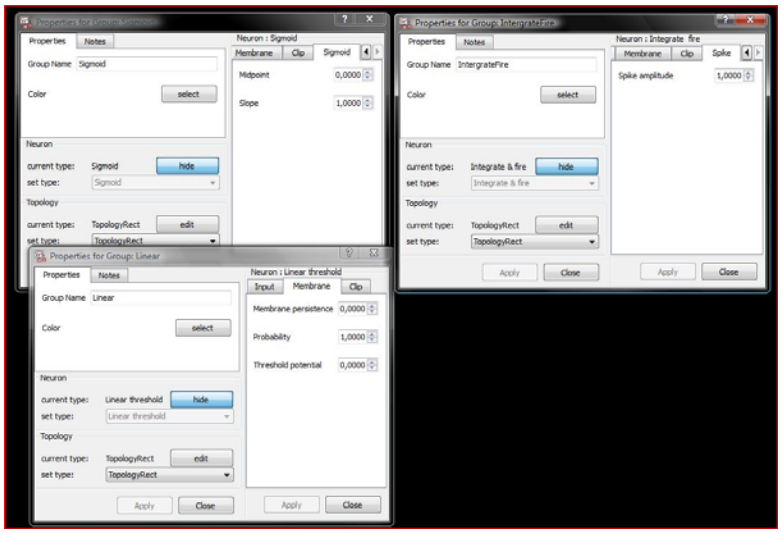


Bild 4b

Die Neuronengruppen unterscheiden sich nicht nur in der Auswahl ihrer Eigenschaften sondern in ihren verschiedenen Formeln, die für jeden Anfänger
-was Neuronale Netze und iqr betrifft-
unglaublich schwer verständlich und kaum nachvollziehbar sind.
Man findet sie im Appendix des iqr- UserManual.

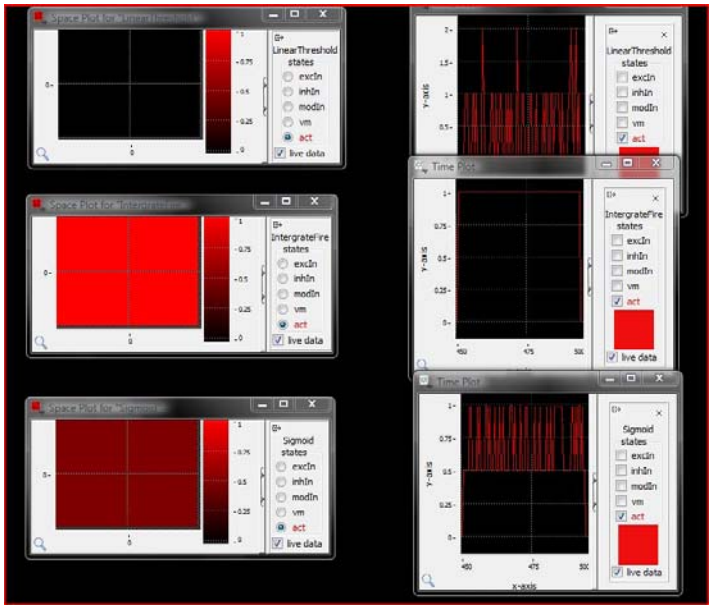
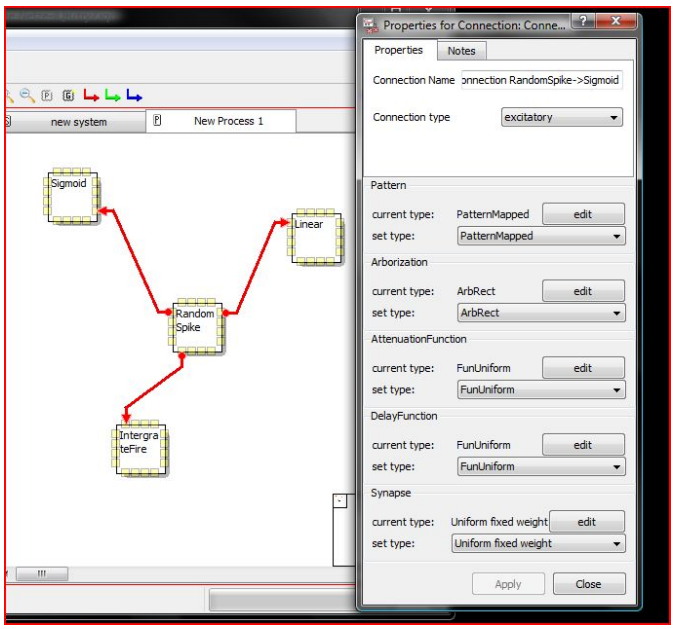


Bild 4c zeigt noch einmal die Unterschiede in den Space und Time Plots der drei,
mit der RandomSpike-Gruppe verbundenen, Neuronengruppen. Ganz oben ist dabei die LinearThreshold-, in der Mitte die Integrate&Fire- und ganz unten die Sigmoid-Gruppe.
Alle sind in aktiv.

Bild 4c



Im Bild links (4d) sind die veränderbaren Eigenschaften einer Verbindung gezeigt.
Der Verbindungsname ganz oben zeigt an, welche Neuronen verbunden sind und in welche Richtung die Verbindung verläuft. Außerdem kann man auch im Nachhinein, also nachdem die Verbindung gesetzt wurde, den Verbindungstyp ändern und aus der exitatorischen Verbindung eine inhibitorische oder modulare machen.

Bild 4d

Das Pattern bestimmt die Gruppe und Anzahl von verbundenen Neuronen im Gitter der pre- und postsynaptischen Membran.
Zur Auswahl der Neuronen stehen einem drei Möglichkeiten zur Verfügung: “Foreach“ bestimmt, dass alle Zellen senden und empfangen, bei “Mapped“ muss man die Projektionspunkte selbst definieren und bei “Tuples“ wird eine individuelle Projektion von einer Zelle auf eine andere Zelle besagt.

Bei der Arborization, der Verzweigung der Neuronen, bestimmt man die Neuronen, die ein Signal senden und empfangen können. Die Arborization befindet sich an jedem Punkt der Gitterstruktur und wird vom Pattern bestimmt, da dieses die zur Verwendung möglichen Neuronen angibt.
Zudem legt man hier fest, ob man ein “RF“ (receptive field) oder ein “PF“ (projective field) verwenden möchte. Ersteres funktioniert nach dem Prinzip many-to-one, es werden also mehrere Neuronen auf eine gemapped, letzteres genau umgekehrt. Das Prinzip ist one-to-many und ein einziges Neuron wird auf mehrere gemapped. So wird entschieden, ob die Arborization auf die pre- oder postsynaptische Membran angewandt wird.

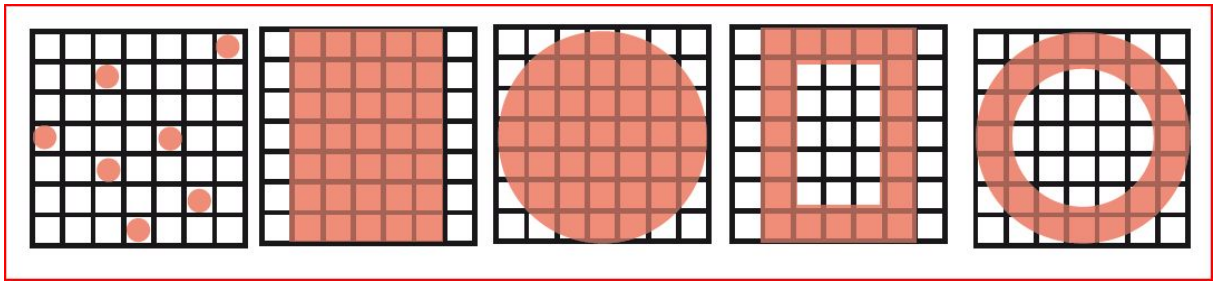


Bild 4e

Die in iqr möglichen Arborizations sind (4e, von links nach rechts):
“ArbRandom“, “ArbRect“, “ArbEllipse“, “ArbRectWindow“, “ArbEllipseWindow“ und “ArbAll“ (hier: nicht dargestellt).

Mit der Änderung der Attenuation ändert man die Verarbeitung der Signalabschwächung für alle Synapsen in der Verzweigung (Arborization). Dabei gilt: Je größer die Attenuation, desto stärker ist das Signal abgeschwächt.

Die Delay-Funktion ist nicht unbedingt ungleich der Attenuation.
Hier wird sich auf die Distanz zwischen Sende- und Empfang-Neuron bezogen und die Verzögerung zwischen ihnen wird verarbeitet.
Der entscheidende Unterschied zwischen Delay-Funktion und Attenuation ist jedoch, dass die Werte ersterer kontinuierlich und stetig gleich bleiben.
Die Werte der Attenuation sind jedoch unstetig und diskret.

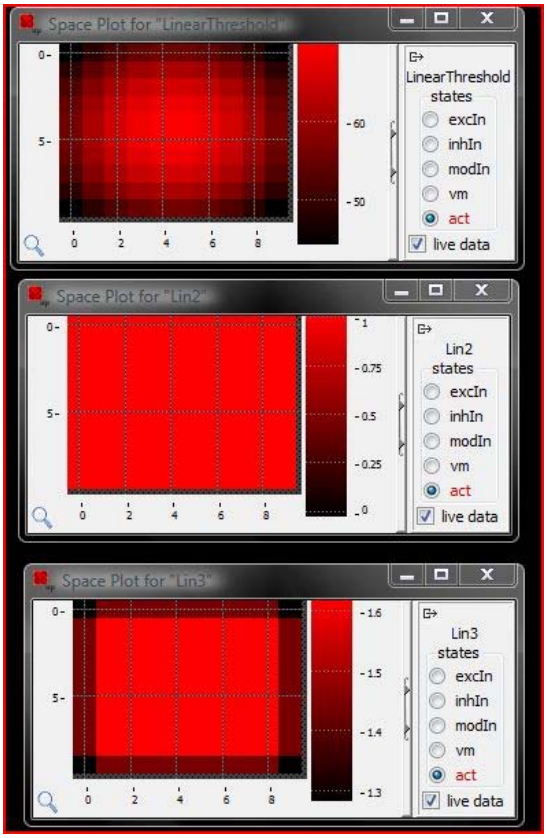


Bild 4f

Im Bild links (4f) wurden die Verbindungen zu drei LinearThreshold-Gruppen so verändert, dass man in der Spikeaktivität Muster erkennen kann.
Bei allen ist das Pattern (mapped-all), sowie die Arborization (ArbAll, PF) und der Synapsentyp (uniform-fixed-weight) gleich. Verändert wurde hier nur die Attenuation.
Das oberste kreisförmige Muster entsteht durch eine lineare Attenuation, das blockförmige mittige durch eine block Attenuation und das unterste Muster durch eine gaussische Attenuation.

Advanced Beispiel: Video Module

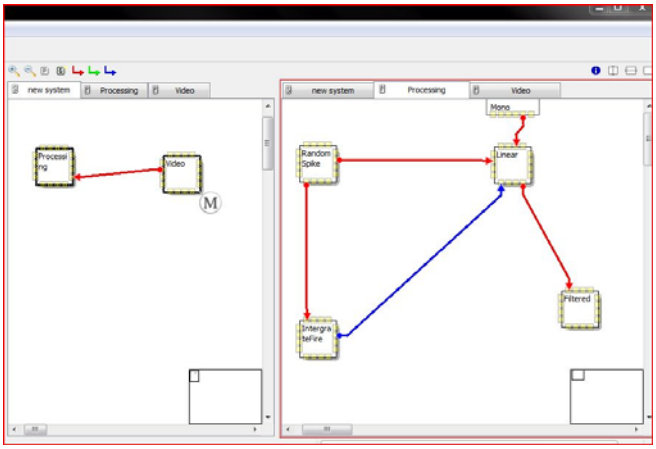


Bild 5a

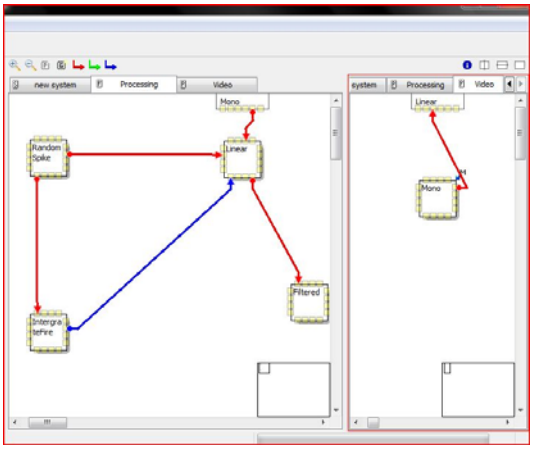


Bild 5b

In den Bildern 5a und 5b sieht man wie an einen unbestimmt Prozesse ein zweiter Prozess angeschlossen wird, der ein Video Module enthält.

Man möchte hier also als Input ein Video von einer angeschlossenen Kamera (z.B. einer Webcam) verwenden.
Die Mono-Gruppe (5b, rechts) des Video Modules ist eine LinearThreshold-Gruppe mit der man die Größe des angezeigten Videos bestimmt. Man sollte das Video jedoch nicht zu groß wiedergeben, da das natürlich viel Kapazität verbraucht.
Mehr als den Video-Prozess mit seiner Mono-Gruppe braucht man eigentlich nicht, um nur das Bild wiederzugeben.
In dem Falle dieses Systems wird jedoch das Video Module an einen anderen Prozess (hier: Processing) angeschlossen, um das Bild noch weiter zu bearbeiten.
Dafür werden die beiden Prozesse verbunden, indem man die Mono-Gruppe an die linearen Neuronen Processings anschließt und ihnen die gleichen Dimensionen gibt, ansonsten wird das Bild stark verpixelt.
An die Linear-Gruppe, die schon von der RandomSpike- und der Integrate&Fire-Gruppe beeinflusst wird, hängt man nun noch eine lineare Gruppe (hier: Filtered).
Diese erhält die schon bearbeiteten Signale der vorhergehenden Gruppen.
Um das Bild zu bearbeiten und z.B. einen Blur-Effekt oder einen EdgeExtraction-Effekt zu erhalten muss man nun nur noch mit der exitatorischen Verbindung von Linear zu Filtered rumprobieren. Möchte man aber z.B. nur noch die Lichtquellen im Bild wiedergeben, spielt man nicht mit der Verbindung sondern mit den Gruppeneigenschaften von Filtered.

Referenzen

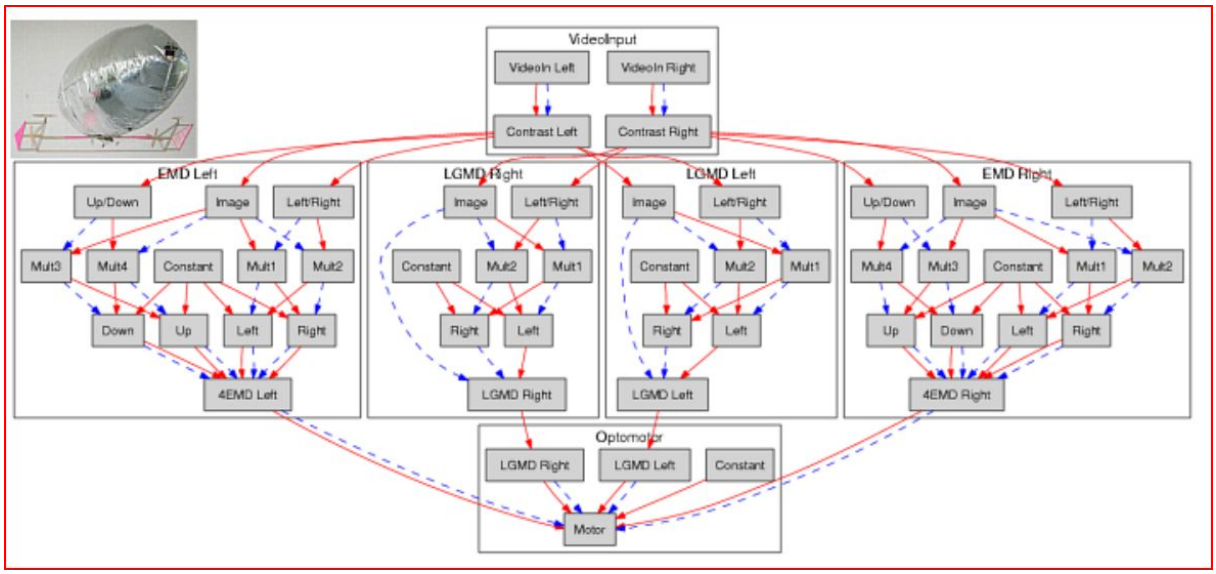


Bild 6

Iqr wurde schon in verschiedenen Projekten für Neuronale Netze verwendet.

Das bekannteste Projekt ist wahrscheinlich das “Artifical Moth Project“, kurz “AMOTH“. In diesem wurde ein fliegender Roboter mit Hilfe von iqr kontrolliert (Bild 6). Man verwendete einen elementary motion detector (EM) der Fliege für die Stabilisation des Roboters und einen locust lobula giant movement detector (LGMD) um Kollisionen zu verhindern.

Ein weiteres Projekt, bei dem iqr verwendet wurde, ist das "Distributiv Adaptiv Control Project" ("DAC"). Darin ging es um das Erlernen von Farbmustern und das Erinnern an diese. Ein Roboter sollte sich bestimmte Farbmuster bzw. Farbfelder einprägen und diese erlernen, während er ein Gebiet erkundete. Das spätere Erinnern an dieser sollte ihm dabei helfen leichter zu seinem Ziel zu navigieren.

Iqr wurde auch im Projekt "Ada: The Intelligent Space" verwendet. Hier war iqr die Basis für die Integration von mehreren Sensoren und Effektoren.
Die Weiterentwicklung dieses Projekt war die "eXperience Induction Machine". Dort baute man einen immersiven Raum ausgestattet mit Sensoren und Effektoren, um Experimente mit einer vermischten Realität (mixed-reality) zu betreiben.

Ein anderes, unglaubliches interessantes Projekt ist das " ReNaChip"-Projekt. Dort verwendet man iqr für Modelle des Kleinhirns. Das Ziel dieses Projekt ist es ein biohybrides Modell zu entwickeln, das die Wiedergewinnung erlernter Inhalte und Verhaltensweisen, die im Alter verloren gehen, demonstriert.

Iqr wird jedoch auch für kleinere Projekte verwendet, was die CSIM Studenten (Interdisciplinary Master in Cognitive Systems and Interacvitve Media) des Kurses "Systemdesign, Integration und Kontrolle" bei der Präsentation ihrer Projekte dieses Jahr (2009-2010) beweisen. Die Kontrolle ihrer e-Pucks baut so gut wie nur auf iqr auf.

Quellen

Für die Basis meiner Präsentation habe ich "iqr: A tool for the construction of multi-level simulations of brain and behaviour" von Ulysses Bernardet und Paul Verschure verwendet und die Informationen auf iqr-Webseite.
Die Informationen für die iqr-Projekte habe ich von der jeweils verlinkten Seiten.