Parallelisierung des signalbasierten CoDi Zellularautomaten zur Simulation von räumlichem neuronalem Wachstum mittels GPGPU

Thomas Gorny & Prof. Dr. Felix Gers

Beuth Hochschule für Technik Berlin, Fachbereich VI - Informatik und Medien

Kurzfassung:

Mit dem CoDi Zellularautomaten wird das Wachstum neuronaler Strukturen auf der Basis aktueller biologische Erkenntnisse realitätsnah simuliert. Dabei werden Growth-Cones, Stellen an den die biologischen Neuron ihr Wachstum fortsetzen, mittels Signalen im Zellularautomaten abgebildet. In einem solchen signalbasierten zellularen Automaten müssen nicht alle Zellen bei jedem Berechnungsschritt aktualisiert werden. Es ist ausreichend nur die Zellen in der Umgebung der Growth-Cones zu betrachten und in die Berechnung einzubeziehen. Daher besteht die Möglichkeit, CoDi als signalbasierten zellularen Automaten mit wesentlich weniger Prozessoren als der Anzahl der Zellen im Zellraum effizient zu parallelisieren. Wir untersuchen, ob CoDi die algorithmischen Anforderung an einem Parallelisierung auf einer Grafikkarten Hardware (General Purpose Computation on Graphics Processing Unit, GPGPU) erfüllt.

Forschungsfrage: Lässt sich der signalbasierte CoDi Zellularautomat auf einer Grafikkarten Hardware mittels GPGPU effizient parallelisieren? Effizient bedeutet dabei, dass die Berechnungszeit nur von der Anzahl der Growth-Cones und nicht von der Anzahl der Zellen im Automatenraum abhängig ist.

1. Einleitung

Mit dem CoDi zellular Automaten [Ger 97, Ger 2016] können neuronale Strukturen innerhalb eines zellularen Raums wachsen und deren Funktionsweise simuliert und analysiert werden. Das Modell geht von initialen Neuronenzellen innerhalb eines diskreten zellularen Raumes aus, von denen Dendriten und Axone als Fortsätze wachsen und sich innerhalb des zellularen Raumes zu einer dichten Netzmorphologie ausbreiten. Die so entstehende Struktur soll den in der Natur vorkommenden neuronalen Netzen möglichst ähnlich sein. Anhand solcher generativ erzeugten neuronalen Strukturen kann anschließend auch das Signalverhalten und der Informationsaustausch zwischen Neuronen innerhalb des Netzes modelliert und untersucht werden. Zur Optimierung der Strukturen in Bezug auf eine gewünschte Funktionalität werden diese entweder mit evolutionären Techniken [Hou 99, Sch 04] oder mit lokalen Lernregeln [Mar 11] optimiert. Sowohl das Wachstumsmodell als auch das Signalmodell sind in CoDi als zellulare Automaten konzipiert. Wir betrachten in im Folgenden ausschließlich die Wachstumsphase von CoDi.

Zellulare Automaten benötigen generell für die Bestimmung des nächsten Zustandes pro Zelle in jedem Verarbeitungsschritt nur die Information über die Zellen in ihrer unmittelbaren Nachbarschaft. Deshalb kann ihre Ausführung auf geeigneter Hardware stark parallelisiert werden. Dies ist entscheidend, um das Modell skalieren zu können und Simulationen in biologisch relevanten Größenordnungen durchzuführen.

Als Hardware zur Simulation bietet sich neben einen Parallelrechner mit mehreren CPUs eine Implementierung auf einer Grafikkarten Hardware an: General Purpose Computation on Graphics Processing Unit (GPGPU). Unter GPGPU versteht man die Auslagerung von Berechnungen auf die GPU, also auf den Grafikprozessor eines Computers, obwohl diese Berechnungen nicht unmittelbar eine visuelle Darstellung auf dem Bildschirm zum Ziel haben. Vielmehr werden ihre Ergebnisse wieder zurück in den Arbeitsspeicher geladen, um dort weiter verarbeitet zu werden. [Rai 09]. Der Vorteil dabei ist, dass die Ausführung parallel auf eine Vielzahl von Rechenkernen der GPU verteilt wird, welche meist auf Berechnungen mit Fließkommazahlen bei einfacher Genauigkeit und großen Matrizen optimiert sind. Die Grafikkarte besitzt dafür auch einen eigenen effizienten Speicher mit schnellen Datenbusraten. Bei geeigneten Algorithmen können dabei sehr große Geschwindigkeitssteigerungen erzielt werden. Entsprechende Karten stehen mit einer Größenordnung von 10.000 Prozessoren kostengünstig zur Verfügung.

Wir untersuchen, ob CoDi als signalbasierter Zellularautomat die algorithmischen Anforderung an eine Parallelisierung erfüllt und erörtern eine praktische GPGPU Umsetzung. Dazu wird zunächst das CoDi Modell mit dem Fokus auf die Wachstumsphase erläutert. Es folgen die Untersuchung zur parallelen Signalverarbeitung und die Beschreibung einer prototypischen Implementierung. Wir schließen mit der Diskussion der Ergebnisse.

2. Die Wachstumsphase im CoDi Modell

CoDi steht als Akronym für „Collect and Distribute“, was sich auf die Signalvorgänge innerhalb eines neuronalen Netzwerks bezieht. Modelliert wird ein neuronales Netz innerhalb eines dreidimensionalen zellularen Raumes. Darin gibt es vier Typen von Zellen: Neuronenkerne, Dendriten, Axone und leere Zellen. Die Simulation wird in zwei Phasen unterteilt, die Wachstumsphase und die Signalphase. In der Wachstumsphase wird ausgehend von den Neuronenkernen zunächst die räumliche Morphologie des Netzes ausgebildet. In der darauf folgenden Signalphase werden innerhalb dieser Struktur dann Signale transportiert, von den Neuronenkernen gesammelt und bei Überschreitung eines Schwellenwertes weitergesendet. Die Axone transportieren dabei neue Signale von den Neuronenkernen weg, die Dendriten sammeln diese Signale von benachbarten Axonen auf und leiten sie ihren zugehörigen Neuronenkernen zu.

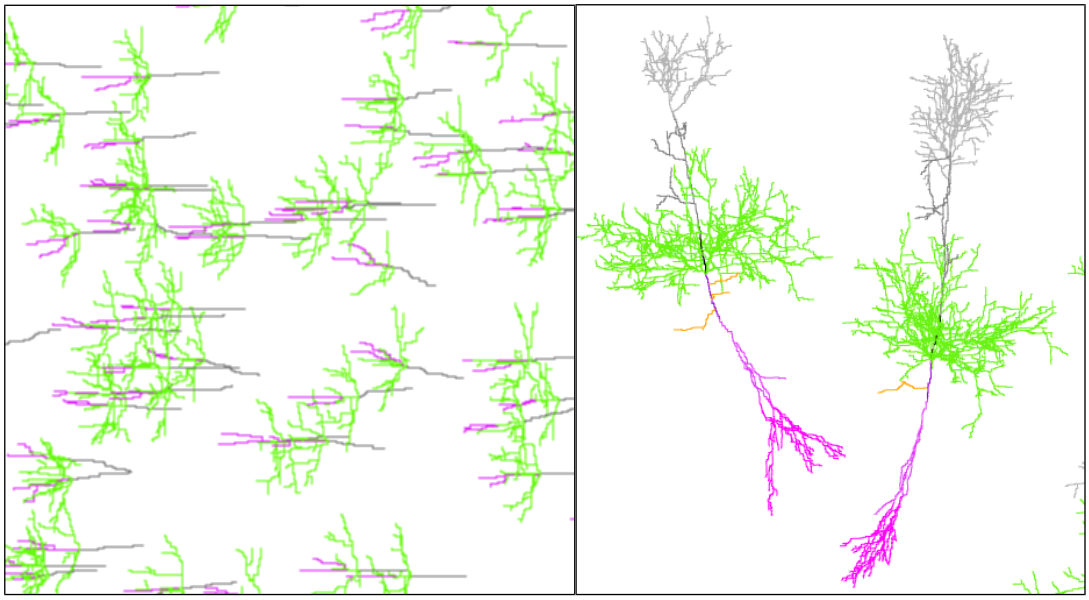

Abbildung1: Links das Wachstum neuronaler Strukturen in CoDi nach 50 Schritten ausgehend von 50 Neuronenkernen; rechts zwei Pyramiden-Neuronen nach Beendigung des Wachstums (Axon in grau, Dendriten in grün und magenta und orange, Neuronenkerne in gelb).

Im CoDi Automaten müssen nur Zellen aktualisiert werden, an denen ein Growth-Cone als Wachstumsimpuls vorliegt, z.B. am Ende eines gerade wachsenden Netzzweiges. Der Großteil des zellularen Raumes bleibt dabei unverändert. Der Anfangszustand des Automaten ist zudem ein nahezu leerer Raum, befüllt nur mit den initialen Neuronenkernen, siehe Abbildung 1 (links). Deshalb wird er auch als dünnbesetzter und signalbasierter Zellularautomat bezeichnet [YiB 96]. Wachstumsereignisse werden als Signale modelliert. Diese Wachstumssignale wandern anhand von probabilistischen Verhaltensmustern durch den Zellenraum, und hinterlassen darin als ihre Spur neue Dendriten- bzw. Axonenzellen. Sie können dabei ihre Richtung variieren, sich verzweigen oder auch absterben.

Die CoDi Modellierung verfolgt das Ziel, eine zu den biologisch nachweisbaren Strukturen möglichst ähnliche Morphologie zu generieren [Vuk 02, Cun 10]. Als Referenz dient dabei die Datenbank NeuroMorpho.Org, welche publizierte dreidimensionale digitale Rekonstruktionen von einzelnen Neuronen sammelt und zur Verfügung stellt [Asc 07]. Aus den Rekonstruktionen von Neuronen bestimmter Zelltypen oder Gehirnregionen lassen sich probabilistische Wachstumsgenome ableiten, durch welche Strukturen im CoDi Automaten erzeugt werden können, die diesen bestimmten Neuronentypen entsprechen, siehe Abbildung 1 rechts.

Die möglichen Zelltypen des Automaten werde dabei auch weiter differenziert. Der CoDi Automat kennt 13 verschiedene Grundtypen, die eine Zelle des ZA-Raumes einnehmen kann. Neben dem Neuronenkern gibt es 3

Typen der Axone: Initial, Stamm und Terminal, 3 Kategorien bei Dendriten: Basal, Oblique und Apical, wobei letztere wiederum aufgeteilt sind in Initial, Trunk und Tuft. Daneben gibt es nun auch Spike und Synapse als eigene Zelltypen. Abbildung 2 zeigt ein Beispiel für das Wachstum neuronaler Strukturen in CoDi nach dem Abschluss der Wachstumsphase.

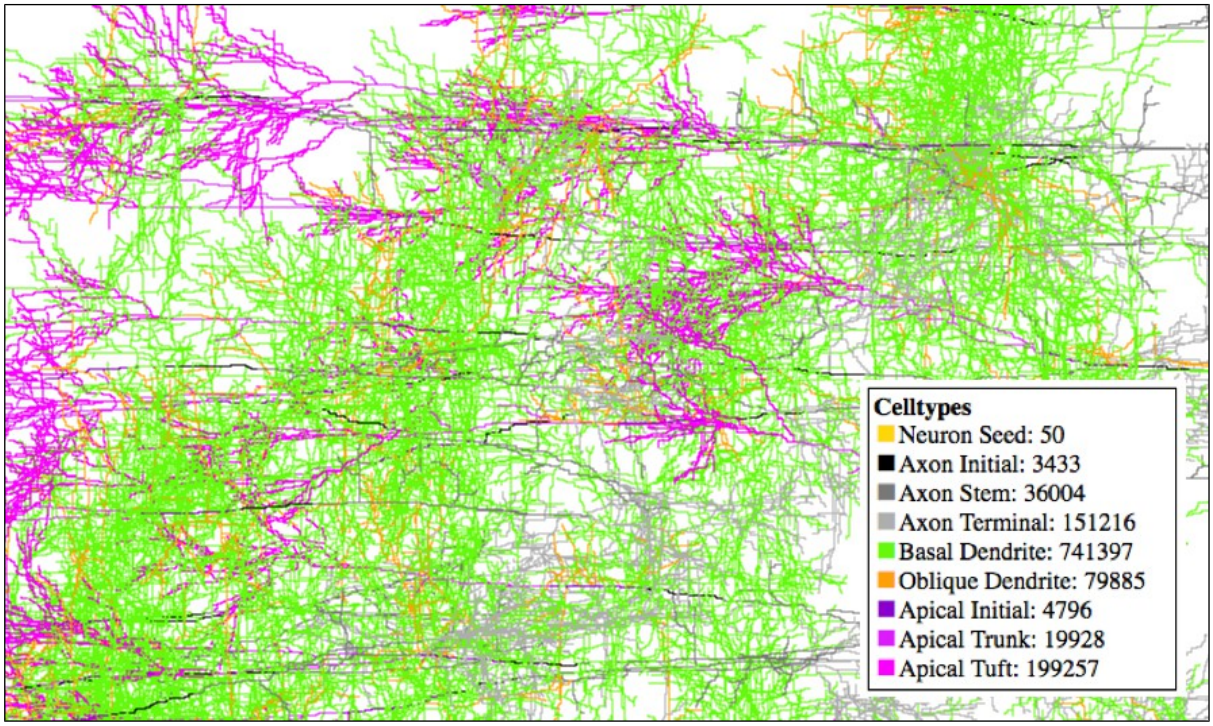

Abbildung 2: Neuronale Strukturen in CoDi nach dem Abschluss der Wachstumsphase. Die Zahlen in der Legende geben an wie viele Zellen des jeweiligen Typs vorhanden sind.

Bezogen auf das Wachstum eines Axons oder eines Dendriten muss die Zelle an der Spitze eines Astes, das Growth-Cone, das Genom kontextabhängig in Verzweigungsanweisungen umsetzen. Diese Zelle enthält ein Wachstumssignal mit dem gesamten Genom und dem Kontext innerhalb der zu erzeugenden Struktur (Haupt- oder Nebenzweig des Neuronenbaumes, Entfernung zum letzten Verzweigungspunkt, etc.) . Um den Speicherbedarf dennoch gering zu halten, beseht das Genom aus wenigen Parametern, die das Wachstum steuern. Die Parameter entsprechen z.B. der Wahrscheinlichkeiten für das Auftreten einer Verzweigung oder dem Ende des Wachstums auf einem Ast. Es handelt sich also um ein probabilistisches, das heißt wahrscheinlichkeitsgesteuertes, Wachstum.

2.1 Wachstumssignale

Das Wachstum des CoDi Automaten basiert auf der Abarbeitung von Signalen, die in Listen variabler Länge gespeichert werden. Dabei ist die Reihenfolge der Verarbeitung der einzelnen Signale beliebig. Zur ihrer Erzeugung werden für jeden initialen Neuronenkern die Position, die Richtung und das Genom übergeben. Aus dieser Information werden jeweils ein Signal für den Kern, ein Signal für ein initiales Axon, ein Signal für einen apicalen Dendriten, und 4 Signale für basale Dendriten erzeugt. Mit diesen Signalen für jedes Neuron startet der Automat dann seine Entwicklung. In jedem Schritt des Zellularautomaten wird darauf hin eine Reihe von möglichen Operationen (stop or transform, turn, branch, move, check collision) geprüft und durchgeführt. Die ersten 3 Operationen (stop or transform, turn, branch) sind abhängig von unterschiedlichen Parametern des Wachstumsgenoms, je nachdem welchen Zellentyp das Signal momentan trägt, bzw. ob es sich auf einem Haupt- oder Nebenzweig des Neuronenbaumes bewegt. Diese Operationen können Teile des Signalzustands verändern, Kopien des Signals erzeugen oder das Signal löschen. Am Ende der Abarbeitung aller Operationen auf allen Signalen steht der neue Zustand des Signalautomaten fest. Alle Operationen bis auf „check collision“ beinhalten ein Zufallsmoment, und können in jedem Schritt unterschiedlich ausfallen. Das modelliert insgesamt ein sehr organisches Wachstum der neuronalen Strukturen. Alle Signale, die nach Ablauf der Operationen ohne Kollision bestehen bleiben, schreiben ihren Zellentyp in den Raum des Zellularautomaten und stehen für den nächsten Bearbeitungsschritt bereit. Somit wird der Raum Schritt für Schritt mit der Morphologie des Neuronennetzes befüllt.

3. Implementierung

Als Schnittstelle zur Grafikhardware nutzen wir WebGL, eine von der Khronos Group als lizenzfreier Standard entwickelte Schnittstelle, um vom Browser aus Computergrafik direkt über die Grafikkarte des Geräts generieren zu lassen. Die Schnittstelle ist mittels programmierbarer Shader ausreichend hardwarenah. Um durch die GPU berechnete Daten auszulesen, werden diese in Texturen gespeichert, also in zweidimensionale Matrizen, in denen jedes Element aus einem Tupel mit 4 Zahlenwerten besteht, welche die Farbkanäle Rot, Grün, Blau und Alpha repräsentieren. Das Datenmodell für Signale und den Zellenraum muss also in diese festgelegte Datenstruktur codiert und daraus wieder decodiert werden. Um zu prüfen, ob an der aktuellen Position eines Signal bereits eine neuronale Zelle im Raum vorhanden ist, muss der aktuell von den Signalen beschreibbare Raum einmal als ganzes als *Kollisions-Textur* im Grafikspeicher vorhanden sein. Signale werden auf eine *Signal-Textur* geschrieben. Deren Größe bestimmt die maximale Anzahl von Signalen, die auf einmal in der GPU-Pipeline verarbeitet werden können.

Die GPU führt mehrere Wachstumsschritte durch, bevor die generierte Netzmorphologie zurückgelesen wird. Da die Signaltexturen nur den aktuellen Zustand der aktiven Signale speichern, und diese sich in jedem Schritt verändern und überschrieben werden, muss die generierte Zellenspur der Signale extra in eine eigene *Ergebnis-Textur* gespeichert werden. Deren Größe berechnet sich aus der Größe der Signaltextur mal die Anzahl der Schritte, die hineingeschrieben werden.

4. Signalverarbeitung

Die Schrittweise Abarbeitung der Signalliste ist an und für sich kein richtiger zellularer Automat, da es keine festgelegten Nachbarschaftsbeziehungen zwischen den einzelnen Signalen gibt. Auch haben die Signale keine feste Position innerhalb des zellularen Raumes und Signale können an beliebiger Stelle der Liste hinzugefügt oder entfernt werden, ohne dass ihre Position innerhalb der Liste einen Einfluss auf das Ergebnis der Berechnung hätte. Jedes Signal ist daher unabhängig von anderen Signalen. Die Signale sind miteinander allein durch eine gemeinsame globale Umgebung verbunden, welche dem gesamten Zellenraum entspricht, und mit der sie interagieren.

Diese Erkenntnis ist für die Implementierung auf der GPU von großer Relevanz. Reguläre zellulare Automaten, in denen jede Zelle eine feste Position und Nachbarschaft besitzt, lassen sich sehr gut auf der Grafikkarte umsetzten, da ihre Struktur sehr den Pixeln eines Monitors ähnlich sind, und der Zellenraum als ganzes gut in einen Framebuffer im Grafikspeicher übersetzt werden kann. Dynamisch wachsende oder schrumpfende Collections, die für die Implementierung der Signalliste nützlich wären, kennt die GPU dagegen nicht, was so eine Umsetzung deutlich erschwert.

Die nötige Größe des Zellenraumes für die CoDi-Simulation, in der tausende von Neuronen dreidimensional wachsen sollen, kommt jedoch schnell an die Grenzen der maximal alloziierbaren Framebuffer-Größe heran. Beispielsweise ist auf einer Hardware mit einer maximale Framebuffer-Größe von 16384 * 16384 Pixeln, was umgerechnet einem 3D-Würfel mit einer Kantenlänge von 645 Einheiten entspricht, ein Wachstum von nur etwa zehn Neuronen möglich.

Deshalb ist die Reduzierung des Wachstumsprozesses mittels dynamischer Signale grundsätzlich sinnvoller, als den Automaten in voller Größe statisch zu implementieren. Die meisten Zellen würden in den einzelnen Schritten ohnehin unverändert blieben, obwohl die GPU dennoch einen Durchlauf über jede Zelle ausführen würde. Es muss daher eine Lösung gefunden werden, welche die dynamische Menge der Signale und ihre Interaktion mit dem Zellenraum effizient auf die Prozessoreinheiten der GPU abbildet.

Eine solche Lösung findet sich in einer verwandten, weit verbreiteten Simulationsform, der Multi-Agenten-Simulation. Agentenbasierte Modellierung, englisch Agent-Based Model (ABM), ist ein etabliertes Verfahren, um computergestützt dynamische Systeme zu untersuchen, wie etwa Insekten-Völker, Schwarm- verhalten, Entstehung von Staus, Wirtschaftssysteme oder soziale Netzwerke. Hier sind Agenten unabhängige Entitäten eines Systems, die nach eigenen Verhaltensregeln innerhalb einer gemeinsamen Umwelt handeln, wobei sie nur eine lokale Sicht auf diese Umwelt haben. Agenten können auch jederzeit in das System hinzukommen oder daraus entfernt werden, etwa durch Reproduktions- und Sterberegeln in Populations-Simulationen. Das ähnelt insgesamt stark den CoDi-Wachstumssignalen.

Da Multi-Agenten-Systeme in der Software-Entwicklung eine lange Tradition haben, gab es auch seit dem Aufkommen des GPGPU Paradigmas Versuche, die Systeme mittels der GPU zu parallelisieren. Die Arbeit von

Lysenko et.al. [Lys 08] beschreibt einen geeigneten Mechanismus zur Verarbeitung der Signale und in Besondere zur Erzeugung neuer Signale auf der GPU.

Bei Verzweigungen stellt sich ein grundsätzliches, durch die spezifische Funktionsweise der GPU vorgegebenes Problem. Verzweigungen bedeuten in der GPU-Implementierung, dass an einer leeren Position innerhalb der Signal-Texturen (leeres Signal) nun ein neues Signal entstehen soll, das eine abgeänderte Kopie eines an einer anderen Texturkoordinate bereits bestehenden, sich verzweigenden, „schwangeren“ Signals ist. Der Shader bearbeitet und schreibt jedoch jeweils nur eine Texturkoordinate auf einmal, und kann nicht in eine zusätzliche Koordinate hineinschreiben sobald eine Verzweigung auftreten soll. Dafür geht der Shader in jedem Durchlauf durch alle Texturfragmente, also sowohl über aktive als auch über leere Signale.

Hier ist also eine Fallunterscheidung im Shader-Code und eine gesonderte Behandlung notwendig für die Fälle, dass an der aktuellen Signalposition ein leeres Signal vorliegt, das neue Daten aufnehmen kann, oder ein aktives schwangeres Signal, welches seine Daten an ein neues Zweigsignal übergeben will. Diese beiden Signale müssen sich gegenseitig finden und interagieren damit eine Verzweigung erfolgreich stattfinden kann. Aktive Signale ohne gesetztes Verzweigungs-Flag werden ohne weiter Bearbeitung durchgereicht.

Da Textur-Zugriffe in den Shadern kostspielig sind und für gute Laufzeitgeschwindigkeiten auf ein Minimum reduziert werden sollten, kann also nicht jedes schwangere Signal die gesamte restliche Signaltextur nach leeren Zellen durchsuchen. Auch muss sichergestellt werden, dass die leeren und schwangeren Signale exakte Paare bilden, zwischen denen die Daten eindeutig ausgetauscht werden, und dabei keine Überlappungen oder Dopplungen bei der parallelen Verarbeitung der Texturen durch die Shader-Einheiten der GPU entstehen. Da Verzweigungen auch theoretisch jederzeit an jeder Position der Signaltextur auftreten können, sind hier sequentielle Sortierversuche sehr umständlich und würden die Vorteile der Parallelisierung gefährden.

Eine einfache, aber nicht deterministische, sondern probabilistische Lösung für die dynamische Generierung neuer Signale wurde vom Lysenko et.al. bei der Simulation von dynamischen Populationsmodellen auf der GPU gefunden [Lys 08]. Dabei wird ein fester Offset festgelegt, und jedes schwangere Signal sucht in der Textur von der eigenen Position aus versetzt um diesen Offset nach einem leeren Gegenstück, und jedes leere Signal sucht versetzt um den negativen Offset, also genau in entgegengesetzter Richtung, nach einem Partner. Durch den festen Offset entstehen dabei bei einem Treffer die gewünschten eindeutigen Paare. Dieses Verfahren wird in Abbildung 3 veranschaulicht, wobei die Signaltextur hier als eindimensionale Liste dargestellt ist.

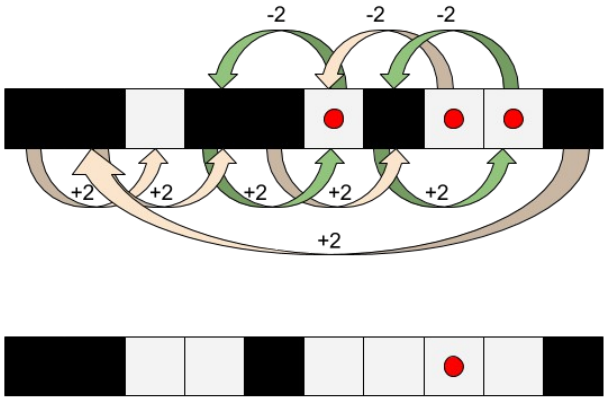

Abbildung 3: Schema zur Feststellung von Verzweigungen innerhalb der Signaltextur. Leere Zellen sich schwarz, aktive Signale sind weiß, schwangere Signale sind mit rotem Punkt markiert. Oben wird die Suche nach Paaren zwischen leeren und schwangeren Signalen mittels festem Offset dargestellt, unten das Ergebnis nach durchgeführter Verzweigung.

Es wird schnell klar, dass dabei nicht sofort alle schwangeren Signale, die sich Verzweigen möchten, eine freie Zelle für das neue Signal finden. Um die Chancen zu erhöhen, kann dieses Verfahren mit einem veränderten Offset wiederholt werden. Die Wahrscheinlichkeit einer Verzweigung hängt direkt von der Anzahl der freien Zellen innerhalb der Signaltextur ab. Lysenko et.al. zeigen, dass durch mehrere Wiederholungen mit unterschiedlichen Offsets diese Wahrscheinlichkeit schnell (exponentiell mit der Anzahl der Wiederholungen) gegen 1 konvertiert. Sind etwa gleich viele leere wie volle Texturzellen vorhanden, liegt die Wahrscheinlichkeit für die Verzweigung aller schwangeren Signale nach 4 Wiederholungen bereits bei ca. 94%.

Diskussion

Die gestellte Forschungsfrage, kann positiv beantwortet werden: Der signalbasierte CoDi Zellularautomat lässt mittels GPGPU effizient parallelisieren. Die Berechnungszeit hängt nur von der Anzahl der Signale, also der Anzahl der Growth-Cones ab.

Wir erwägen im Folgenden die Vor- und Nachteile einer GPU- gegenüber einer CPU-Implementierung. Das Wachstum auf der GPU ist auf den durch die Kollisionstextur bestimmten Zellenraum beschränkt, während die Signale auf einer CPU-Implementierung sich uneingeschränkt ausbreiten könnten. Fast 85% der Ausführungszeit der GPU-Implementierung dauert allein das Rückschreiben und Speichern der Ergebnisse. Die reine Signalverarbeitung auf der GPU nimmt weniger als 3% der gesamten Rechenzeit ein. Das insgesamt kompliziertere GPU-Setup lohnt sich nur, wenn die Übertragung von Daten zwischen GPU und CPU optimiert werden kann.

Literatur

[Asc 07] Ascoli G A, Donohue D E, Halavi M. (2007) NeuroMorpho.Org: a Central Resource for neuronal Morphologies. Journal Neuroscience., 27(35):9247-51

[Cun 10] Cuntz H, Forstner F, Borst A, Häusser M. One Rule to Grow Them All: A General Theory of Neuronal Branching and Its Practical Application, PLoS Computational Biology, 2010

[Ger 97] Gers F. A., De Garis H. Codi-1bit : A Simplified Cellular Automata Based Neuron Model. In Artificial Evolution Conference (AE), Nimes, France, 1997

[Ger 16] Gers. F A. Wachstum von neuronalen Strukturen innerhalb eines zellularen Automaten, Research Day 2016 Stadt der Zukunft, 2016. BWV Berliner Wissenschafts Verlag, 2016.

[Gon 04] González-Burgos G et al.. Synaptic efficacy during repetitive activation of excitatory inputs in primate dorsolateral prefrontal cortex. Cereb Cortex, 14(5):530-42, 2004

[Mar 11] H. Markram, W. Gerstner and P.J. Sjöström, A history of spike-timing-dependent plasticity, *Frontiers in Synaptic Neuroscience*, Vol. 3, Nr. 4, pp. 1-24, 2011

[Hou 99] Hough M, De Garis H, Korkin M, Gers F A, Nawa N E. Spiker : Analog Waveform to Digital Spiketrain Conversion in ATR's Artificial Brain "cam-brain" Broject. In Int. Conf. on Robotics and Artificial Life, Beppu, Japan, 1999

[Rai 09] Raina, R., Madhavan, A., & Ng, A. Y. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning* (pp. 873-880). ACM. 2009

[Sch 04] Schwarzer J. Lernverfahren für evolutionär optimierte Künstliche Neuronale Netze auf der Basis Zellularer Automaten. Logos Verlag Berlin, 2004

[Lys 08] Lysenko M. & D'Souza R. M. A framework for megascale agent based model simulations on graphics processing units. Journal of Artificial Societies and Social Simulation 11(4)10, 2008.

[Vuk 02] Vukšić M et al.. Perinatal growth of prefrontal layer III pyramids in down syndrome. Pediatric Neurology , Vol. 27, Issue 1, p.36 – 38, 2002

[YiB 96] Yi-Bing Lin; Fishwick, P.A., "Asynchronous parallel discrete event simulation," in Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on , vol.26, no.4, pp.397-412, Jul 1996

Abbildungsverzeichnis

Abbildung 1: Beginnendes Wachstum in CoDi und Pyramiden-Neuronen (Quelle: eigene Darstellung)

Abbildung 2: Neuronale Strukturen in CoDi (Quelle: eigene Darstellung)

Abbildung 3: Verzweigungen innerhalb der Signaltextur (Quelle: eigene Darstellung)

Kontakt

Felix Gers, Beuth Hochschule für Technik Berlin, Luxemburger Straße 10, 13353 Berlin

Telefon: (030) 4504 2529, E-Mail**:** gers@beuth-hochschule.de, Internet: http://prof.beuth-hochschule.de/gers/