Max Planck Institute for
Dynamics and Self-Organization

Analyse von multivariaten Zeitreihen aus Herzdynamiksimulationen

Internship, März 22, 2021 – Mai 10, 2021

Roland Stenger

1 Einführung

Dieser Bericht beschreibt meine Arbeit im Rahmen des Praktikums in der Biomedical Physics Group am Max-Planck-Institut für Dynamik und Selbstorganisation. Das Praktikum ist Teil eines Projektes in Zusammenarbeit mit Dr. Rupamanjari Majumder, welche für die Simulationen von Herzdynamiken zuständig ist, und Dr. Sebastian Herzog, welcher ebenso wie ich Methoden aus dem Bereich des maschinellen Lernens für die im folgenden beschriebenen Probleme verwendet und die Vorverarbeitung der Simulationsdaten für eine sinnvolle Anwendung der Lernsysteme entwickelt.

Meine Arbeit befasst sich unter anderem mit der Datenaufbereitung welche von der Simulation der elektrischen Aktivität eines Herzens stammt. Dabei sollen die Daten so verarbeitet werden, dass diese als Input eines neuronalen Netzwerkes dienen können um die im folgenden beschriebenen Aufgabenstellungen zu adressieren. Die Entwicklung jener Neuronalen Netzwerke ist ebenfalls Teil meiner Arbeit. Alle hier beschriebenen Methoden wurden von mir in Python implementiert. Die Idee zu dem Algorithmus aus Kapitel 3 stammt von Sebastian Herzog.

Die Simulationen bestehen aus den Zeitreihen der Werte für die Spannung V sowie die Calcium-Ionenkonzentration an allen Vertices des simulierten Herzens. Im Verlauf der Arbeit werden Teile dieser multivariaten Zeitreihe so verarbeitet, dass von dieser Teilmenge der gesamten Simluationsdaten die Werte bisher ungesehener Zeitreihen übriger Vertices vorhergesagt werden. Das kann zum einen so funktionieren, dass ein Teil der Zeitreihen der Vertices an der Oberfläche des simulierten Herzens als Input für ein neuronales Netzwerk dient, worauf dieses die Zeitreihen übriger Oberflächenvertices vorhersagt. Zum anderen ist es auch möglich, dass anhand der Oberflächendynamik (zeitliche Entwicklung der Spannung V oder Ionenkonzentration) der Vertices die Dynamik im Inneren vorhergesagt wird. Diese Möglichkeiten sind zum einen Teil schon ausprobiert, zum anderen noch in Planung. Es stellt sich heraus, dass die Datenaufbereitung einen entscheidenden Einfluss annimmt. Die Simulationsdaten haben raumzeitliche Eigenschaften, weshalb die Position der Vertices eine wichtige Eigenschaft ist. Die neuronalen Netzwerke die in diesem Projekt getestet werden sollen, basieren unter anderem auf einem CNN (Convolutional neural network), welche in seiner geläufigsten Implementation lediglich ein reguläres Grid im N-dimensionalen verarbeiten können. Zunächst muss also das unstrukturierte Gitter, aus welchem die Herzgeometrie besteht so verarbeitet werden, dass daraus ein reguläres Grid wird, welches dabei trotzdem die räumliche Anordnung der Vertices nicht zu sehr verändert.

Im folgenden wird vorgestellt wie die Verarbeitung der Vertices stattfindet. Dabei wird zunächst nur die Oberfläche des Herzens berücksichtigt, welche in ein kartesisches Gitter überführt wird. Dabei wird zeitgleich diskutiert inwiefern diese Umwandlung für das Trainieren eines neuronalen Netzwerkes sinnvoll ist. Zunächst wird kurz erläutert wie die Daten welche Ausgangspunkt für Kapitel 3 sind vorverarbeitet werden.

2 Daten-Vorverarbeitung

Die Rohdaten der Simulationen liegen werden im double-precision floating-point format (float64) gespeichert. Dadurch fallen erhebliche Datenmengen an, allerdings müssen die Werte für V und der Ionenkonzentration eines jeden Vertices für unsere Zwecke vermutlich nicht in der Präzision eines float64-Formates gespeichert werden. Aus diesem Grund werden die Werte jeden Vertices auf Zahlenwerte zwischen -128 und 127 skaliert, was der möglichen Spanne eines Integers mit 8 Bytes entspricht. Dadurch wird der Speicherbedarf um den Faktor 8 reduziert. Die für die Skalierung benötigte Kenntnis über das globale Maximum und Minimum der Werte für V und der Ionenkonzentration werden in einem vorherigen Schritt mithilfe sämtlicher zur Verfügung stehenden Simulationen identifiziert. Bevor die Daten in einem neuronalen Netzwerk verarbeitet werden, werden jene, welche für einen einzelnen Ausführprozess des Netzwerkes benötigt werden, zwischen 0 und 1 skaliert. Damit werden diese in einem float32-Format umgewandelt. Diese Skalierung findet nicht für den gesamten Datensatzes auf einmal statt um den Speicherbedarf zu minimieren.

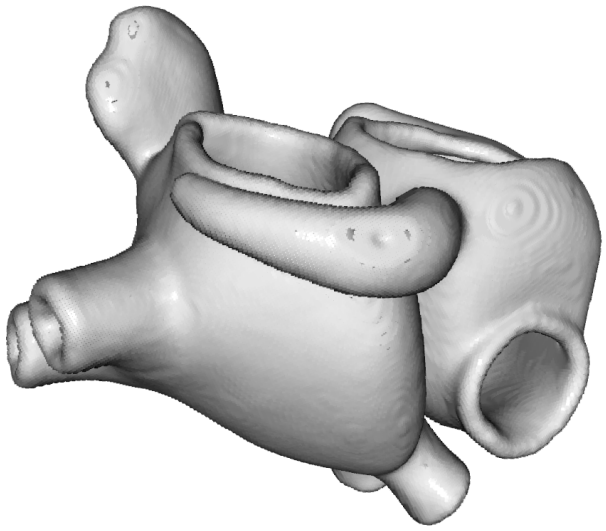

Abbildung 1: *Visualisierung des Herzmodells welches für die Simulation verwendet wird. Die Geometrie setzt sich aus etwa 27Mio Vertices zusammen.*

3 Separation der inneren und äußeren Oberfläche

Das Simulierte Herz hat eine komplexe räumliche Struktur, wie in Abbildung 1 zu sehen ist. Es lässt sich zwischen einer äußeren und inneren Oberfläche unterscheiden. Jeden Oberflächenvertice als Teil der äußeren beziehungsweise inneren Oberfläche zu identifizieren ist in manchen Fällen willkürlich. Dennoch ist eine Projektion auf eine 2-dimensionale Ebene sinnvoller wenn diese Unterscheidung zuvor vorgenommen wurde. Bei einer 2D-Projektion der gesamten Herzoberfläche auf (zum Beispiel) die Oberflächen eines umhüllenden Würfels würden teilweise Vertices nebeneinander liegen, die geometrisch weit voneinander entfernt liegen. Das liegt daran, dass Punkte auf der äußeren beziehungsweise inneren Oberfläche direkt *hintereinander* entlang einer Raumrichtung liegen können. Diese Möglichkeit lässt sich reduzieren wenn zwischen einer inneren und einer äußeren Oberfläche unterschieden wird und die Projektion für beiden Oberflächentypen getrennt ausgeführt wird.

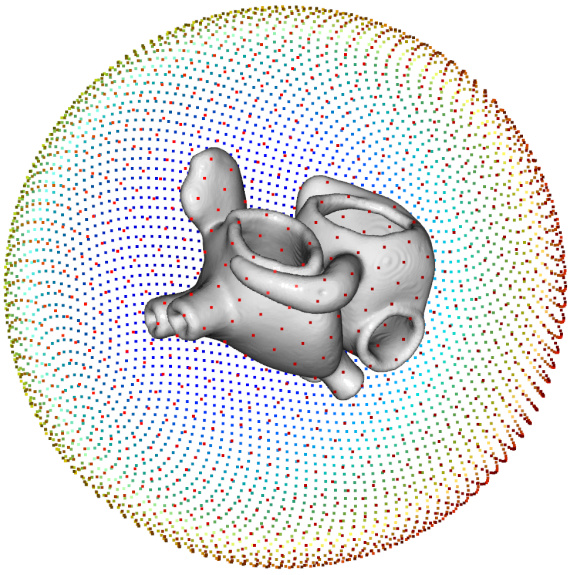

Abbildung 2: *Punkte auf der Oberfläche einer umhüllenden Kugel, die als Ausgangspunkt für Strahlen dienen um die Anzahl der Treffer jeden Vertices mit den Strahlen zu zählen.*

Der Algorithmus zur Separation der inneren und äußeren Oberfläche basiert auf der Annahme, dass die Oberflächenvertices im inneren des Herzens im Schnitt weniger gut sichtbar sind als die Vertices außen, insofern man das Herz von außen und verschiedenen Winkeln betra-

chtet. Gemäß der Annahme steigt mit verringerter Sichtbarkeit die Wahrscheinlichkeit, dass sich ein Vertice auf der inneren Oberfläche befindet. Quantifiziert wird die Eigenschaft der *Sichtbarkeit* durch den folgenden Algorithmus:

Zunächst werden Punkte auf einer umhüllenden Kugel definiert, wie in Abbildung 2 zu sehen ist.

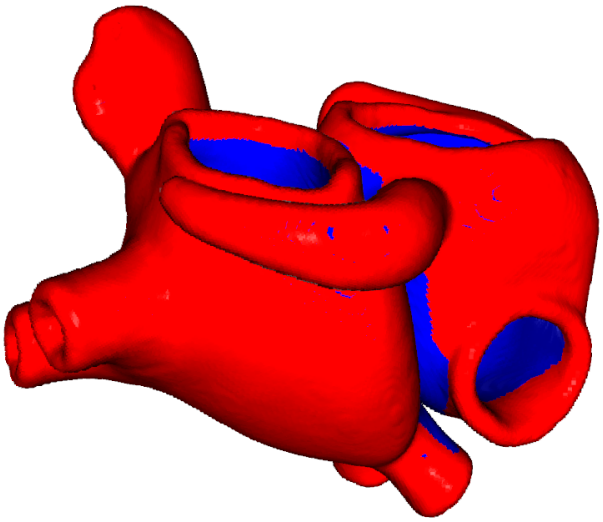

Abbildung 3: *Kolorierte Vertices gemäß ihrer Klassifizierung als Teil der inneren beziehungsweise äußeren Oberfläche.*

Jeder dieser Punkte dient nun als Ausgangspunkt für ein Bündel von Strahlen, welche in alle Raumrichtungen geradlinig verlaufen. Trifft ein Strahl nun ein Vertex auf der Herzoberfläche wird dieses Auftreffen durch einen Zähler aufgezeichnet. Dadurch wird jedem Vertex eine Zahl zugeordnet, welche angibt wie oft dieser von einem Strahl getroffen wurde. Trifft ein Strahl einmal einen Vertex, kann dieser nicht mehr andere Vertices treffen. Der Strahl kann das Herz also nicht *durchdringen*. Je öfter getroffen wurde, desto Wahrscheinlicher ist es, dass sich der Vertex auf der äußeren Oberfläche befindet. Gegenteiliges gilt für die innere Oberfläche. Nach dem Zählen der Anzahl der Treffer für jeden Vertex wird die Herzgeometrie durch Faltung mit mehreren Gauß'schen Kernel weichgezeichnet. Dadurch entsteht ein eindeutigeres Histogramm. Die Anzahl der Treffer nach ebenjenen Weichzeichner kann in einem Histogramm dargestellt werden wie in Abbildung 4. Hier lassen sich deutlich zwei Cluster erkennen. Ein Threshold wird angesetzt und die Vertices der inneren beziehungsweise äußeren Oberfläche zuzuordnen. Das Resultat dieser Klassifizierung ist in Abbildung 3 farblich markiert.

Im folgenden wird die äußere Oberfläche für die weitere Arbeit genutzt. Dabei geht es darum diese sinnvoll auf eine Ebene zu projizieren.

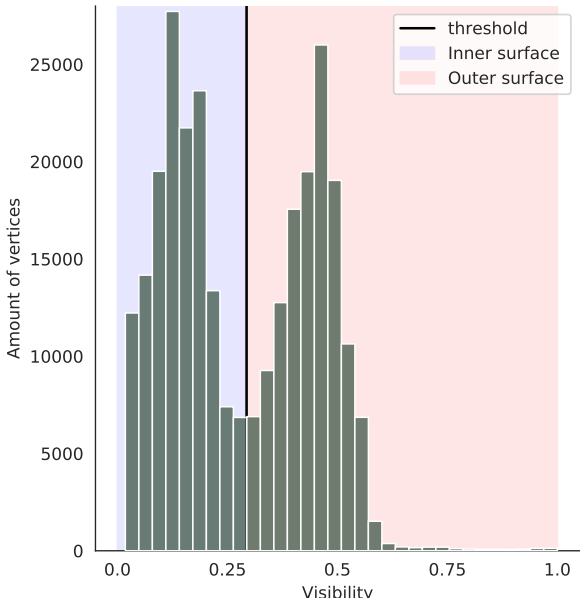

Abbildung 4: *Häufigkeitsverteilung der Sichtbarkeit der Vertices. Der Wert für Visibility ist in Form einer normierten Kennzahl dargestellt. Je häufiger ein Vertex von Strahlen aus der umhüllenden Kugel getroffen werden, desto sichtbarer wird dieser Vertex, wobei die maximale Sichtbarkeit bei 1 liegt.*

4 Ausblick

Schlussendlich soll die Einteilung in innere und äußere Oberfläche dazu dienen, ein neuronales Netzwerk mithilfe einer dieser Oberflächen zu trainieren, welches die raumzeitliche Eigenschaft der Daten effizient verarbeiten kann. Ein Versuch arbeitet mit der Projektion der äußeren Oberfläche auf eine Kugeloberfläche. Der Radius aller Vertices in Polarkoordinaten ist nun für alle derselbe. Die Werte der Latitude und Longitude können nun wie in Abbildung 5 geplottet werden. Dieses unstrukturierte Gitter wird im letzten Verarbeitungsschritt zu einem regulären Gitter interpoliert. Dabei wird die Methode der *nearest-neighbor Interpolation* verwendet. Das interpolierte Gitter ist in Abbildung 6 zu sehen. Dieses kann nun als Ausgangspunkt für ein Convolutional Neural Network dienen. Es war bisher nicht möglich dieses Verfahren für ein Recurrent Neural Network zu verwenden um die zeitliche En-

twicklung der Spannung in der Datenauswertung zu berücksichtigen. Der benötigte Speicherbedarf war Speicherbedarf war zu groß.

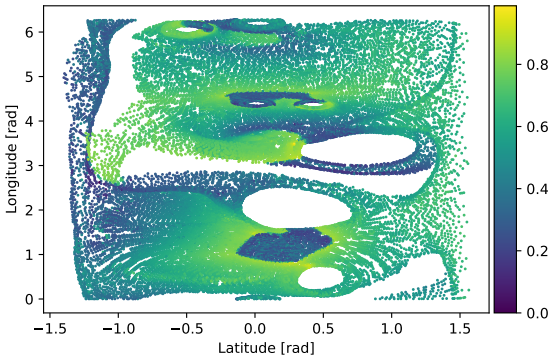

Abbildung 5: *Latitude und Longitude für Vertices auf der projizierten Kugeloberfläche. Die Werte der Spannung V aus der Simulation sind hier zwischen 0 und 1 skaliert und farblich codiert.*

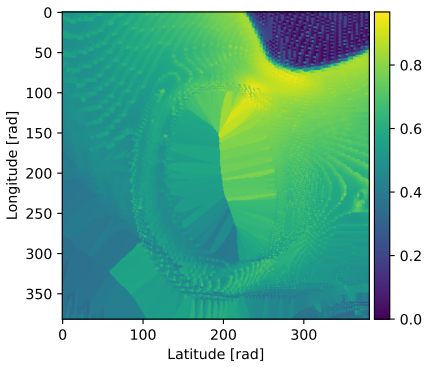

Abbildung 6: *Überführung des irregulären Gitters in ein reguläres. Die Werte der Spannung V aus der Simulation sind hier zwischen 0 und 1 skaliert und farblich codiert.*

Teil einer anderen Idee besteht aus der Projektion der äußeren Oberfläche auf die Seiten eines sechs-seitigen Würfels. Diese Anordnung der Vertices kann nun von einem Convolutional Neural Network verarbeitet werden. Die in der Einführung benannten Aufgabenstellungen sind nun theoretisch durch das Training von neuronalen Netzwerken zu erproben. Praktisch gesehen tut sich hier aber das Problem des Overfitting auf: Um die Vorhersagen von trainierten Netzwerken auf unbekannten Simulationen zu testen bedarf es vermutlich einer Vielzahl an Simulationen mit jeweils individuellen Erregungsmustern. Die große Menge an Daten, die üblicherweise im Bereich des maschinellen Lernens nötig ist, ist aufgrund der Komplexität der Simulation schwer zu liefern. Ein neuronales Netzwerk mit denselben Simulationen zu testen mit denen auch trainiert wurde birgt die Gefahr des Overfitting da die Dynamik in der simulierten Zeitspanne weitestgehend periodisch ist und somit nicht ausgeschlossen werden kann, dass ein Erregungsmuster bestimmter Zeitschritte nicht vollständig unbekannt ist.