AGH University of Science and Technology
Faculty of Physics and Computer Science

Engineering thesis

Krzysztof Nowak

Field of study: **Applied Computer Science**

Design and implementation of a massively-parellel algorithm for the retina stimulation data analysis

Thesis supervisor: **Tomasz Szumlak PhD Eng.**

Kraków, February 2020

Uprzedzony(-a) o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2018 r. poz. 1191 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony(-a) o odpowiedzialności dyscyplinarnej na podstawie art. 307 ust. 1 ustawy z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyszym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.) „Student podlega odpowiedzialności dyscyplinarnej za naruszenie przepisów obowiązujących w uczelni oraz za czyn uchybiający godności studenta.", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Jednocześnie Uczelnia informuje, że zgodnie z art. 15a ww. ustawy o prawie autorskim i prawach pokrewnych Uczelni przysługuje pierwszeństwo w opublikowaniu pracy dyplomowej studenta. Jeżeli Uczelnia nie opublikowała pracy dyplomowej w terminie 6 miesięcy od dnia jej obrony, autor może ją opublikować, chyba że praca jest częścią utworu zbiorowego. Ponadto Uczelnia jako podmiot, o którym mowa w art. 7 ust. 1 pkt 1 ustawy z dnia 20 lipca 2018 r. — Prawo o szkolnictwie wyższym i nauce (Dz. U. z 2018 r. poz. 1668 z późn. zm.), może korzystać bez wynagrodzenia i bez konieczności uzyskania zgody autora z utworu stworzonego przez studenta w wyniku wykonywania obowiązków związanych z odbywaniem studiów, udostępniać utwór ministrowi właściwemu do spraw szkolnictwa wyższego i nauki oraz korzystać z utworów znajdujących się w prowadzonych przez niego bazach danych, w celu sprawdzania z wykorzystaniem systemu antyplagiatowego. Minister właściwy do spraw szkolnictwa wyższego i nauki może korzystać z prac dyplomowych znajdujących się w prowadzonych przez niego bazach danych w zakresie niezbędnym do zapewnienia prawidłowego utrzymania i rozwoju tych baz oraz współpracujących z nimi systemów informatycznych.

.....

(czytelny podpis)

Recenzja promotora

Recenzja recenzenta

Abstract

The subject of this thesis is the design and implmentation of an algorithm that can work through large volumes of data, obtained from experiments involving measuring of neuron responses in retinal tissue, which has been artificialy stimulated with and electrical impulse. While still not the best it could be, there have been made significant improvements to the time it takes to perform the calcualtions on the data at hand.

The thesis focuses on the software and hardware aspect of the data analysis, other subjects are used only for a theoretical background introduction to the experiment and can be further studied in the papers, that can be found in the reference section at the end of the document.

Contents

1 Introduction

1.1 General introduction

Due to the tremendous improvement in hardware for spatio-temporal stimulation of nervous tissue the volumes of data obtained from experiments pertaining to that field of study have been growing at an exponential speed and gathering that data these days takes a very short time. And while having such great amounts of data may at the beginning sound great, it is practicly of little value if one cannont extract actual knowledge from it.
Fortunately with the fast growth in the areas of data analysis, statistics, machine learning and many other related subject, a equaly big growth of software and hardware addressed at dealing with those subjects followed. With the availability of those new tools many new oportunities for research in subjects that previously were difficult or impossible due to physichal constrains are now within our grasp.
This thesis describes the design and implementation of an massively parallel algorithm for handling of one of those big data sets.

1.2 Experiment

The data obtained for this thesis comes from an experiment revolving around the electrical stimulation of nervous tissue in the eye with specific electrical patterns and recording of the responses from those cells. The goal of those experiments is the eventual creation of technology that could serve as an replacement for the human eye in situations where the photosenstive cells got damaged and/or destroyed, but the neurons that carry the impulses to the brain are still intact. If a situation like that were to arrise artificial stimulation could solve the problem of the missing layer that translates light into the correct impulses, but first it is necessary to figure out how to do it correctly.
The experiment makes use of modern multielectrode arrays (MEA), which are small surfaces densly populated with electrodes capable of both sending and recording electrical impulses with a high frequency. While this technology has been available for a few decades only recently has there been a big improvement in their quality leading to the beforehand mentioned growth of data obtained from thoses experiments. Which in turn ended up creating a demand for better and faster algorithms to analyse that data.
The full description of the experiment can be found in one of the references below.

1.3 Source code location

The full code developed for this thesis can be found at: `https://github.com/krzynowak/template_matching`

2 Theoretical introduction

2.1 Biology of the experiment

2.1.1 Eye anatomy

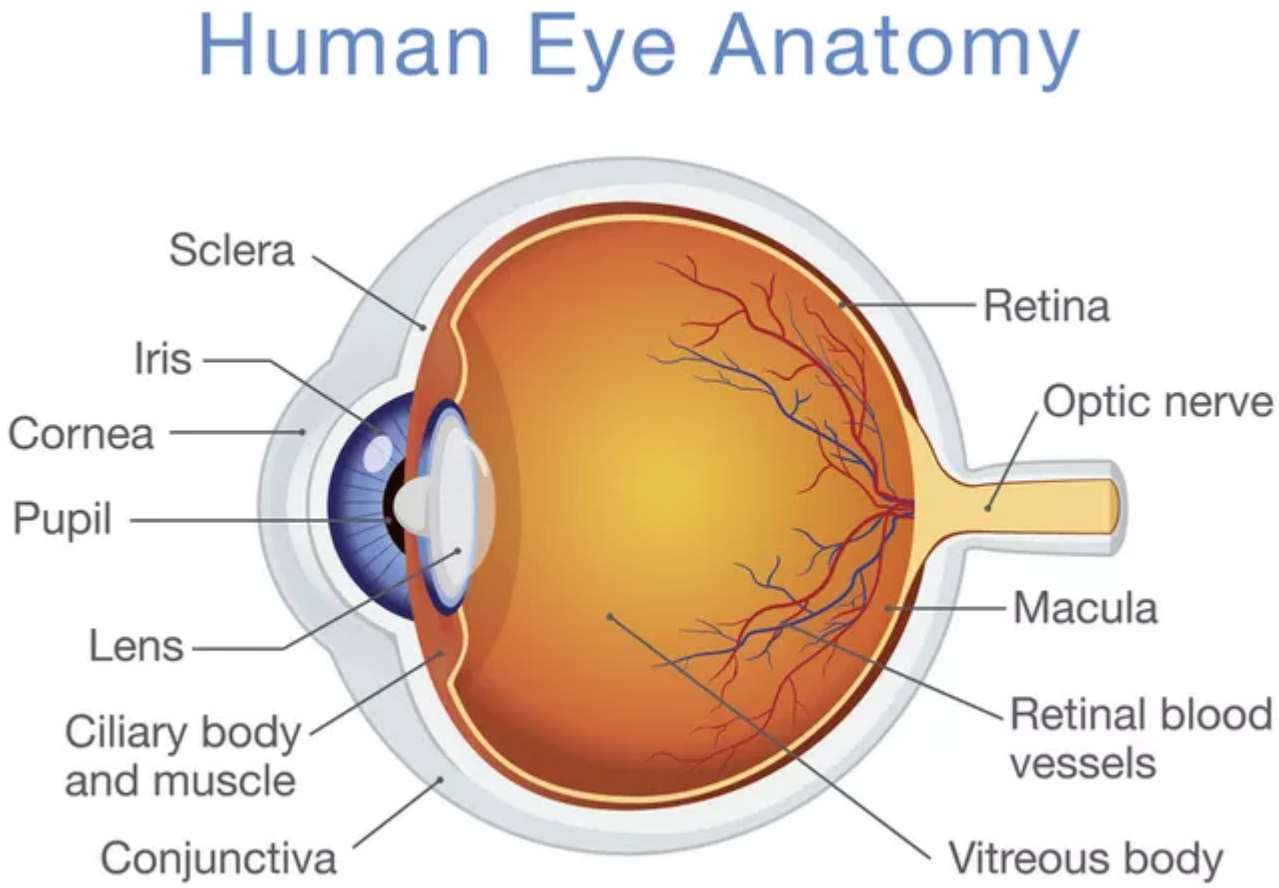

Figure 1: Anatomy of the human eye (source: https://www.thoughtco.com/how-the-human-eye-works-4155646)

The eye is the organ responsible detection of light waves from a certain range of frequencies, its structure is very complicated but for the sake of understanding this paper only a superficial knowledge of this subject is required.
The conversion of light into actual information, which will eventualy translate into „something that we see" starts when light passes through the pupil. Once it gets past there it hits the lense where it gets focused onto the retina. If the light has a certain frequency then the retinal cells responsible for that frequence will activate and send an electro-chemical impulse to the optic nerve which then gets forwarded to the brain where those impulses get translated into what humans consider to be vision.
For the subject at hand we take most interest in the retina as this is the part of the eye easiest to work with.

Ideally this research should conclude with the creation of some sort of device that can record images like a camera and then translate those images into a specific electrical patern that not only generates the same response as an correctly function eye but also doesn't cause any or negligeble amounts of damage to the organ and it's owner.

2.1.2 Retina tissue structure

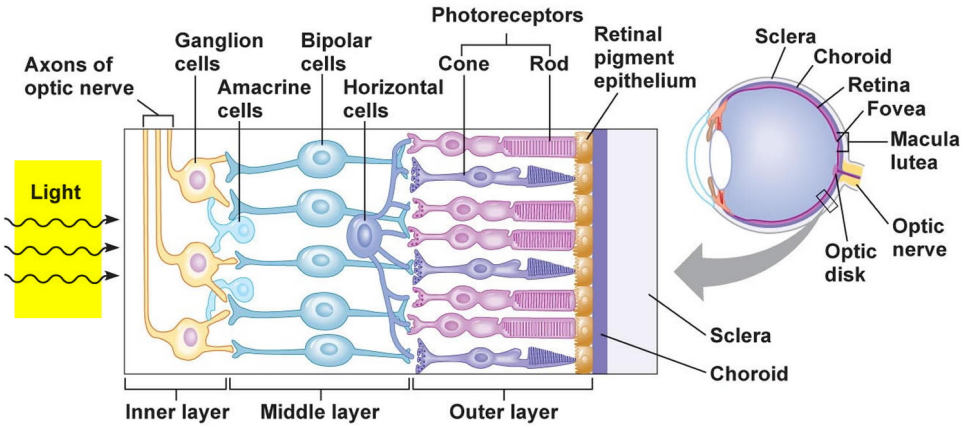

Figure 2: Retina tissue (source: https://healthjade.com/human-eye/)

In the structure of the retina we differentiate between three layers:

- Inner layer - contains axons of the optic nerve and ganglion cells
- Middle layer - contains bipolar, amacrine and horizontal cells
- Outer layer - the photoreceptors (cones and rods)

Our main focus is the inner layer, specifically the ganglion cells as those are the cells that build the most outer layer that can be stimulated using electrical current and measure it's response to it. Becuse of that, the eye tissue used for the experiment require to have their outer layers removed, that way access can be gained to those particular cells that are of value for this experiment.

2.1.3 Neurons

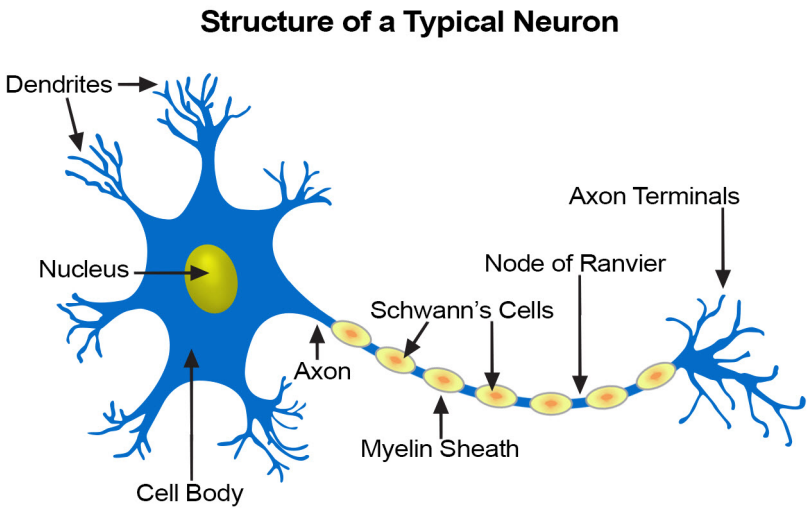

Figure 3: Strucutre of a neuron cell (source: https://training.seer.cancer.gov/anatomy/nervous/tissue.html)

The last biological component that needs to be discusses is the neuron itself. Neurons are electrically excitable cells that operate on an all-or-nothing principle. This means that once a neuron has a charge stored it will fire only when it recieves a signal that is equal to or bigger than his excitation potential, if that happens the neuron will discharge and then start gathering a new charge for the next time it's needed.
Neurons differentiate into three types:

- Sensory neurons - they react to stimulation like pressure, heat, light etc.
- Motor neurons - send signals from the brain or spinal cord and are responsible for the control of many things like muscles and glands.
- Interneurons - those neurons are responsible for the creation of connection and transfering signals.

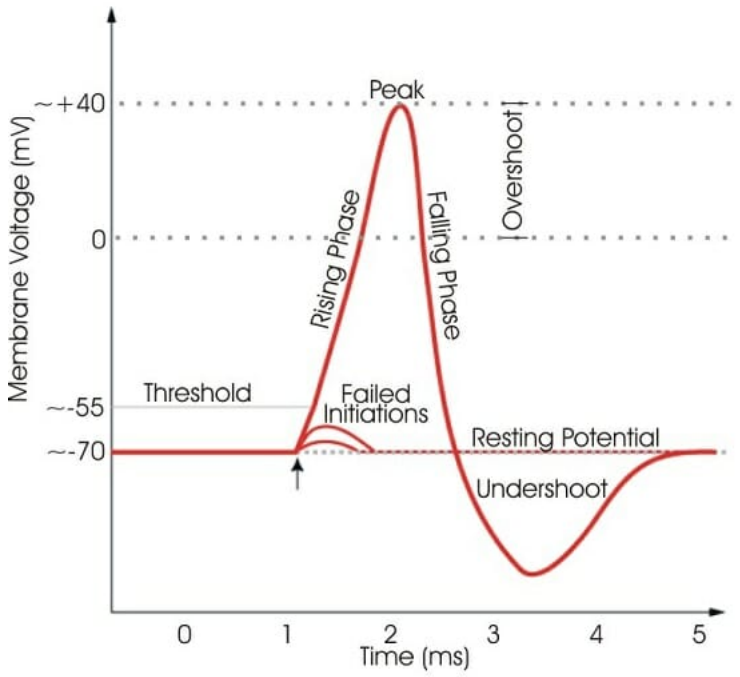

Figure 4: Example activation function for a neuron (source: https://biologydictionary.net/dendrite/)

In the figure above a best case scenario can be seen, which unfortunately isn't as easy to replicate in experiments, because working with neurons on their scale is difficult and attempts to stimulate them always have to deal with possible additional response from other neurons. Even if only one electrode is used to send an impulse, the response will be recorded on multiple electrodes. Those electrodes then need to be analyzed to determine how many neurons fired and which sample is the best one (electrode closest to the activated neuron). Another problem that remains are artifacts, which are leftover charges that end up being noise in the measurements.

In the section discussing the retina tissue special attention has been paid to the ganglion cells, the reason for that is that those cells are a subtype of neuron cells. Which means they can stimulated with a multielectrode array.
Due to them existing in many types of varying sizes, reposnses and connections the experiments focuse only on a small subset of them.

2.2 Experiment hardware

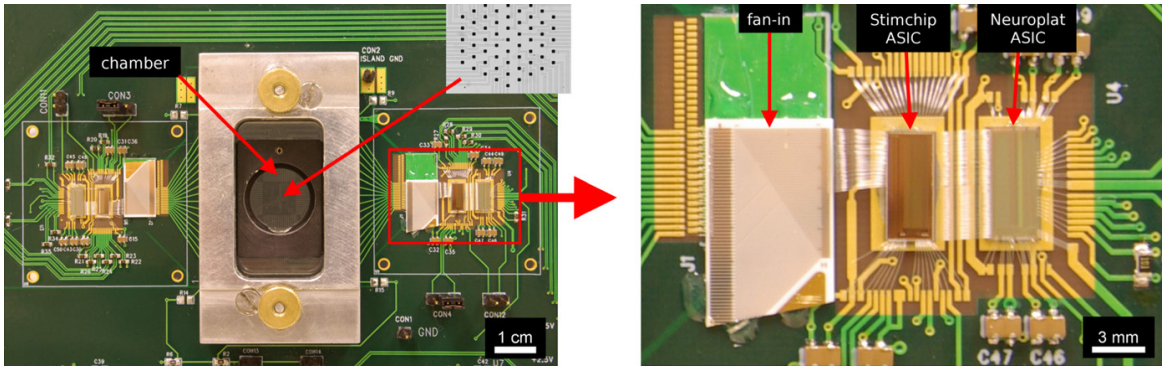

Figure 5: Example setup for 64 electrodes (source: Paweł Hottowy et al 2012 J. Neural Eng. 9 066005)

A example setup can be seen in Figure 5. It consists of a chamber where the electrodes are and where the tissue on which the experiment is being conducted will be placed after it is correctly prepared. On the left and right a stimchip Apliecation-Specific Integrated Circut (ASIC) and a neuroplat ASIC can be found. Those parts are used to program the multielectrode array and to record the results.

2.3 GPGPU

General-Purpose computing on Graphics Processing Units refers to the use of Graphics Processing Units (GPU) in combination with traditional CPUs to obtain very big improvements in performance for calculations that can be done in parallel.
From this point onwards this paper will focus on a specific parallel computing platform which was used in this thesis - Nvidia's Compute Unified Device Architecture (CUDA).

2.3.1 Differences in architecture

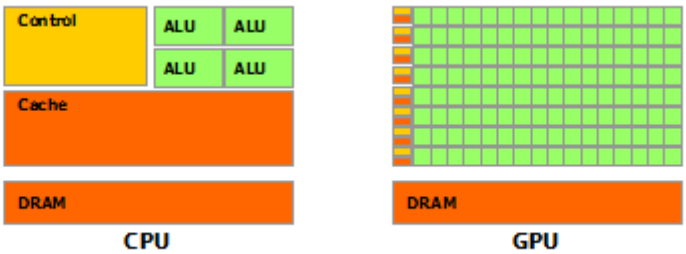

Figure 6: CPU vs GPU (source: CUDA Toolkit v10.2.89 - Programming guide)

While both CPU and GPU are processors, what makes them different is their unique architecture that emerged due to specialisation in fields they have been used in. CPUs were created

to handle logical operations as such they are much better processing serial instructions and in order to do that effectively they rely on a smaller amount of powerfull cores.
GPUs on the other hand emerged to handle graphics, in order to do that efficiently they were optimised to handle simple instructions (weaker cores), but to do a lot of them at the same time(a lot of cores). This specific trait is what makes GPGPU so special as it allows access to the best of both worlds at the same time and gives access to the world of high performance computing.

2.3.2 Programming model

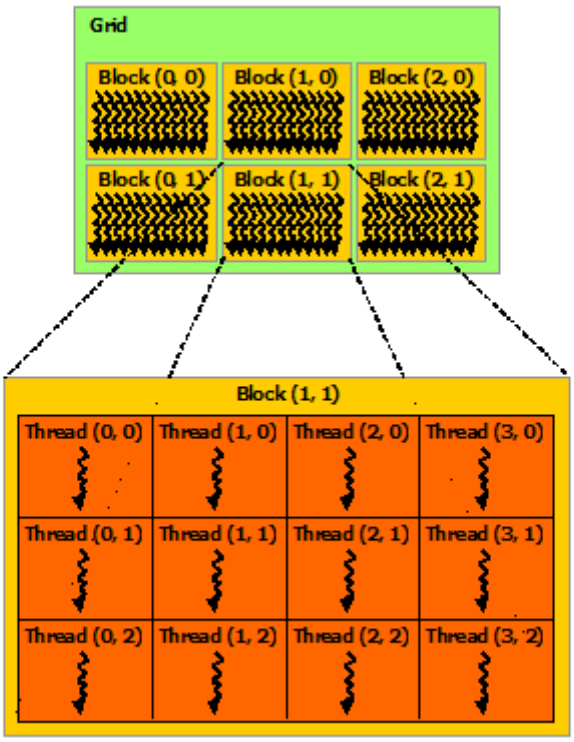

Figure 7: Grid layout(source: CUDA Toolkit v10.2.89 - Programming guide)

In order to be able to execute multiple parallel instances cuda makes use of kernels, grids, blocks and threads. Every kernel called, will need to know how many blocks (grid) to create and how many threads there should be per block. Threads inside of a block can be up to threee dimensions, blocks in grind can have only two dimensions prior to compute capability 2.X and three dimensions for all other versions. This offers a very high-dimensional enviorment to work with.

2.3.3 Hierarchy

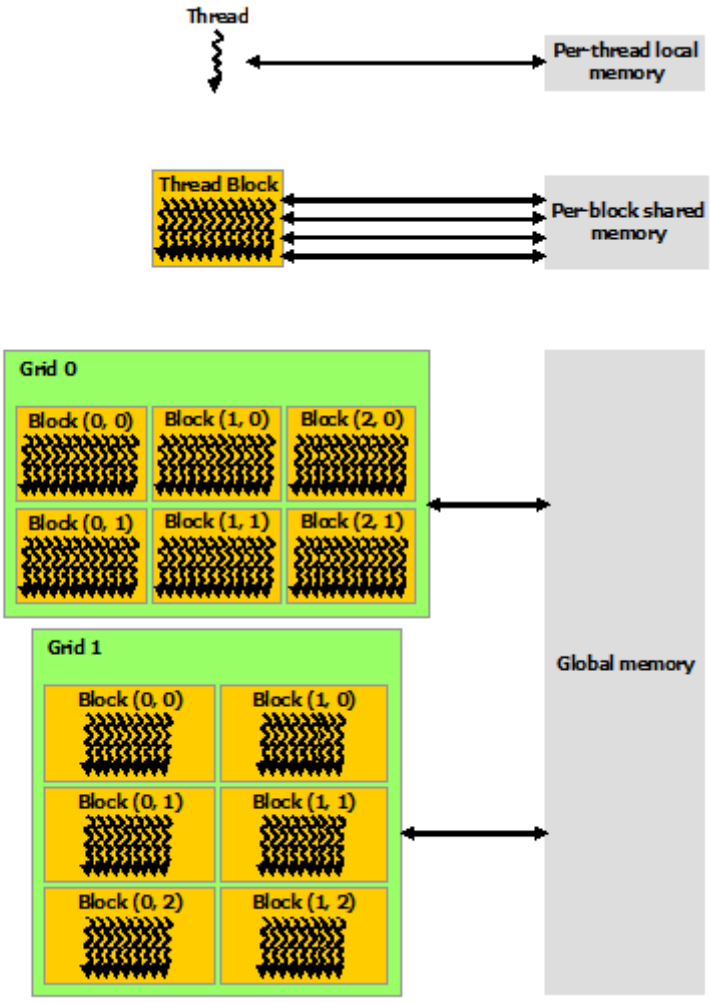

Figure 8: Memory hierarchy(source: CUDA Toolkit v10.2.89 - Programming guide)

Cuda makes use of three types of memory:

- Global memory - accessible for everyone but slow.
- Shared memory - this memory is assigned to specific block, this makes it much faster but access is limited to the threads of only this particular block.
- Local memory - this memory is for personal use of every thread.

The distinction allows to use specific types as they are needed in order to get the best results. The general idea is to use global memory for input and output, shared memory to optimize computation and coalesce results in blocks and local memory for for local variables of each thread.

2.3.4 Host/device paralelization

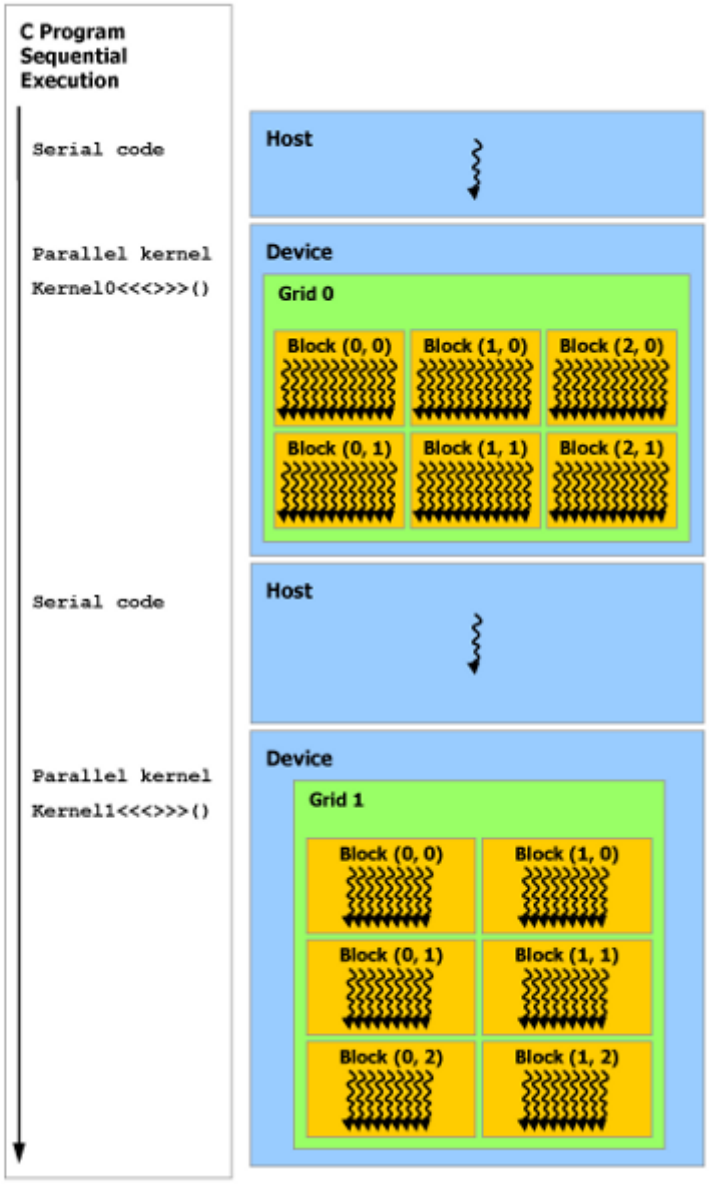

Figure 9: Code execution(source: CUDA Toolkit v10.2.89 - Programming guide)

Another great feature of GPGPU is that not only does the code on the gpu execute parallely for each thread, but due to the CPU and GPU being separate it's possible for them to work at the same time leading to even better optimized code.

3 Software and algorithm

3.1 General algorithm description

The algorithm optimized in this thesis is a classification algorithm. It is responsible for identifying which template fits the best with the signals recorded during experiments.

3.2 Algorithm description

At first it is required to find all templates that match with a specific neuron and electrode. This is executed once for a set of neurons and electrodes. Once that is finished the next step is to read the data recorded from the experiment and subtract the artifacts.
Having obtained both templates and samples it is time to start the the computation.
Each sample file is divided into several sets of samples so it is necessary to iterate over all of them. Now for each of those sample sets one must check all possible combinations without repetitions of templates against the samples.
For every possible combination it is necessary to check the current best fitting option, this is acomplished by calculating the sum of absolute differences between the sample and the template.
Finally the best matching templates (lowest noise value) are returned as the solution for each sample set.

3.3 Initial implementations problems

The initial implementation of the algorithm suffered from many problems that lead to it underperforming when compared to what was needed from it. The biggest problems where multiple big loops nested in each other, resulting in a very long execution time even when dealing with small numbers of neurons and electrodes. Because the dataset itself is very big, even when dealing with small parameter sizes the speed of calculation was at best acceptable. On the other hand if dealing with bigger ones the time would skyrocket to multiple days of calculation. This was caused by one of the nested loops iterated over a set of combinantions without repetitions and since factorials grow extremely fast this led to very lnog times to match all the templates.

3.4 Approach to optimization

The optimization was done with three goals in mind:

- Make loading of data not interfere with the calculation.
- Use CUDA kernels to optimize operations for single batch

- Use cuda streams to have multiple batches run at once

While the original goal was more focused on optimization for a single batch, given the amount of data at hand and having gained some experience working with it, it would seem that the better appproach is to use more kernels but with less optimized algorithms (less focused due to diminishing returns). This approach would guarantee better use with new hardware, and better possibility for adjustments at a later date should the need arise.
This was achieved by using bigger grids instead of more threads. In the long term this should prove to be of use as new cards come out with support for more parallel kernel executions, also the leftover threads that can be used to either handle all possible electrodes at once or instead perform more optimization. It is up to the person responsible for integration into specific experiments to decide which is of more value given the problem at hand.

3.5 Algorithm - optimized

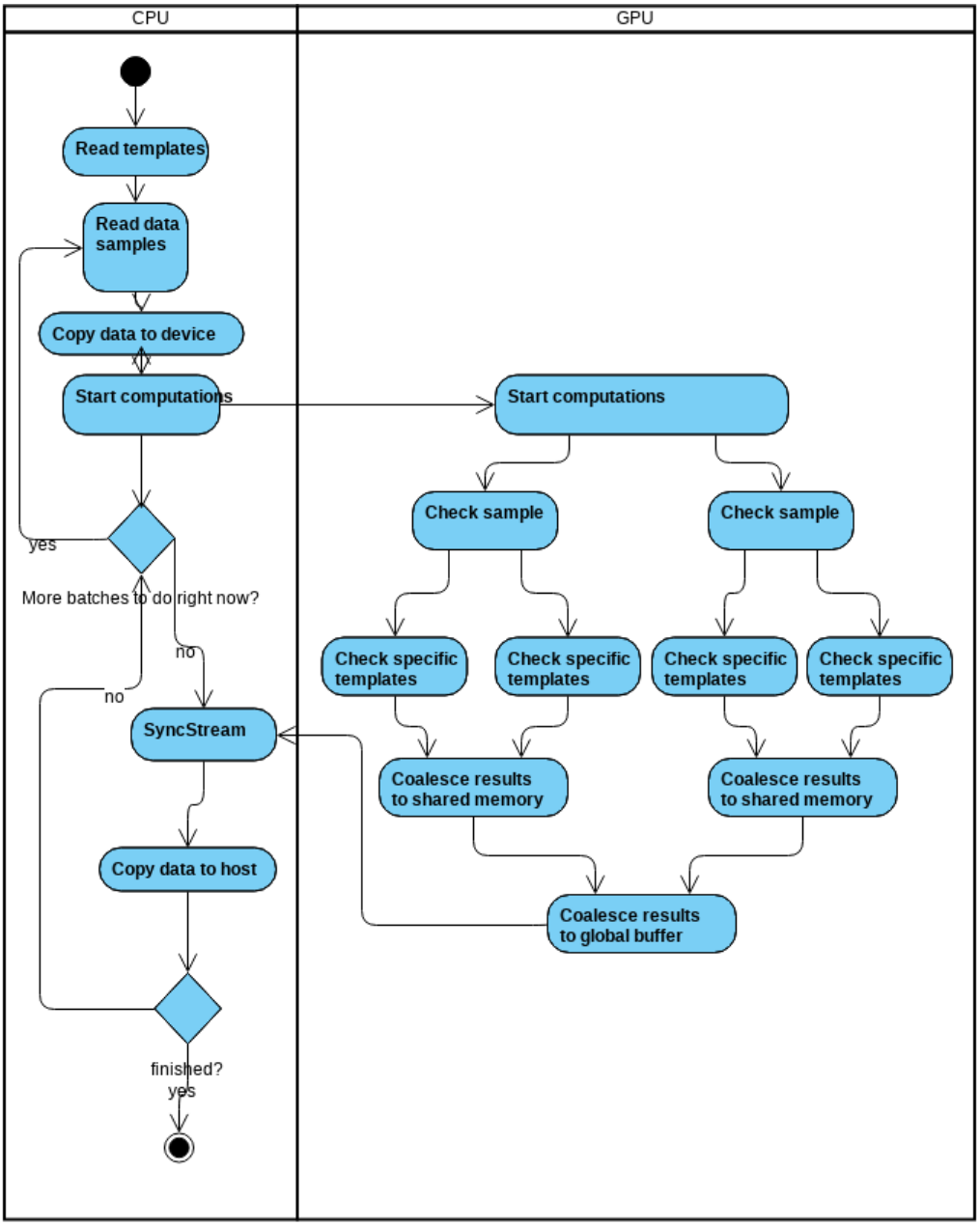

Figure 10: Behavioral diagram

The algorithm is fully implemented in C and CUDA, and was designed in such a manner that it can be adjusted to the needs of a specific experiment and integrated into the data anlysis pipeline in the way it is desired.

Currently the algorithm is using defines and hard coded numbers to set parameters. This aproach allows to perform adjustments to the algorithm parameters from the config file, making it possible to also handle changes to things like the structure of data files with minimal effor. As long as the changes are within the realm of small adjustments that do not require rebuilding signifficant chunks of the algorithms logic, then there shouldn't be any problems with adjustemnts.

Another benefit of this aproach is that many of the data structures can have their size defined at compilation which not only allows to avoid using dynamic memory allocation, but also guarantees that those structures are on the stack which can be accessed faster than the heap.
The algorithm implementation can be divided into three fragments:

- initiation - this part handles the the functionality that has to done only once. It covers the basic initialization of variables, strucutures and is responsible for reading of the ei file which contains the templates and variables that are need for further execution. Allocation of some memory also takes place here.
- The middle layer - this part covers the loop that executes all the functionalitry needed to run a single iteration of the algorithm. This involves reading the sample data, copying of data do device calling the kernel and then retrieving the result.
- End of code - this fragment is responsible for cleanup of all allocated varibles to avoid memory leaks.

The beginning and the end of the code mostly handle simple things that can't be optimized further so there isn't much to go over there. Some simple improvements performed here were a reduction of data reading to only what is needed instead of reading everything and only using what is required. Also copying of the data multiple times has been changed to a single write to memory instead of the several that were there before.

The main improvement in performance is obtained by dealing with the multiple nested loops. In practice this is achieved via the elemination of the two outermost loops and having a single kernel run for each batch of data.
The first loop was the one iterating over all the sample sets, this has been changed to instad call a grid with size equal to the amount of sample sets in the file. The second loop did prove to be a bigger hurdle as it was responsible for generation of combinations, which not only isn't something that's easy to do on CPU but also isn't easy to do in parallel with a GPU.
There always remained the option of generating all possible combinations at start and using them but due to factorials growing extremely fast this would not work for many electrodes, especialy given that the upper limit for them is 513. One more idea was to use necklaces but that also didn't seem to be smart in the long run so it too eneded up being abandoned, due to it actualing being a net loss in performance.

The total possible combinations is decided by the amount of electrodes tested, acceptable activity samples, and the amount of templates to match against, giving a total combination count of:

$$t^n \binom{e}{n}$$

t - number of templates to match against + 1

e - number of electrodes to match

n - number neurons accepted per sample

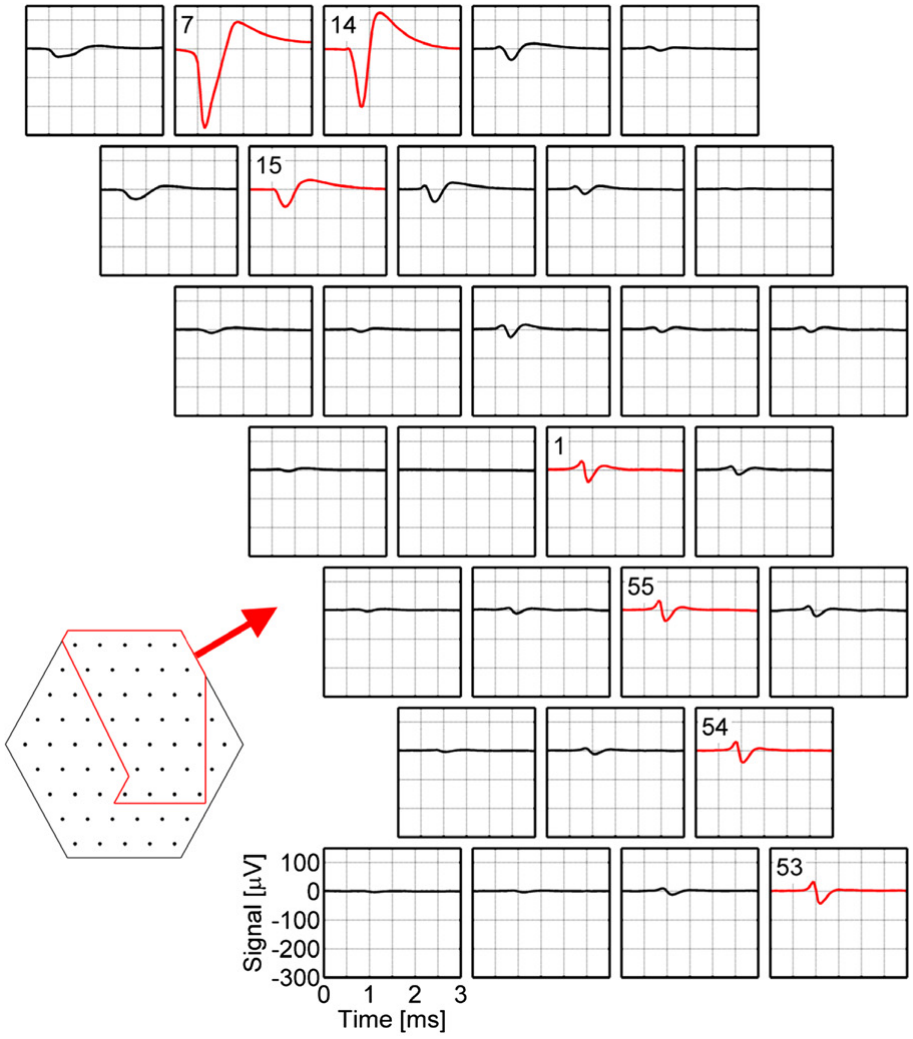

Figure 11: Example of multiple electrodes detecting a single response (source: Paweł Hottowy et al 2012 J. Neural Eng. 9 066005)

Fortunately the number of neurons that requires our attention is only two since the possibility of there being three of them activated by a single impulse is so slim it can be ignored. Both this and the number of electrodes is known at start so it can be used to calculate all position

combinations before code execution. Knowing the positions each stream has to work with all that remains is to assign a thread for each and let them execute.
Final step in calculation is to find the absolute difference between the samples and templates and then return the combination that had lowest noise value.
After this is finished all that that remains to do is wait for all threads to finish, coalesce the results from each block and then write them to the output.

4 Results

4.1 Hardware

Hardware used for time measurements:

- CPU - Ryzen 7 2700
- GPU - GTX 1070ti
- SSD - Samsung evo 860 500GB
- RAM - 16GB 3200MHz

4.2 Tests

Testing for the original implementation has been done only for handling a single data batch. The new implementation was measured for two cases, one where there was only a single data batch and another one that handles 128 batches as this is per hardware documentation four times the number of simultaneous kernel lanuches that is supported.

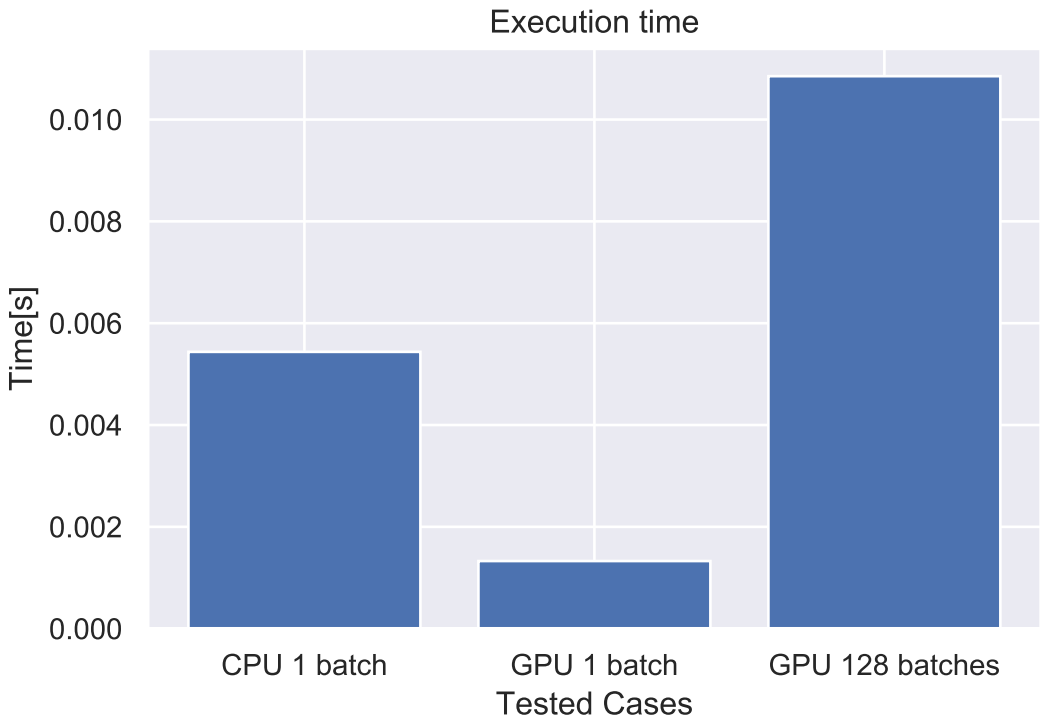

Figure 12: Execution times

4.3 Summary

While the results aren't as spectacular as desired an improvement of 4.09 times for a single batch is acceptable given that the main advantage of this algorithm is the ability to handle multiple batches at once.

On the other hand comparing the reasult from the original implementation to the GPU running an improvement of 64.13 times can be seen, this is much more satisfying and an acceptable end result.

5 Conclusion

In the end obtaining a 100 fold speed increase was not possible. While 64 times is quite satisfactory, there is still room for improvement.
Optimization can still be done in a few places. Especialy the function that generates the combinations of all templates. Unfortunately due to the original assumptions of how to aproach this subject it seemed like a good idea to flatten the arrays with the data, but this eventually ended up being more of a problem than help because the memory used is global and acces isn't sequential but random instead. This leads to this one assignment being probably the biggest sink of time as it is called very frequently and should be addressed.
Aiming for data to be read on the CPU while the GPU is executing it's own code ended up being of little improvement as this takes barely any time to do with use of SSD drives.
The code is written in a way that it's easy to have it be integrated into any data pipeline and should handle new releases of hardware since it will scale with the a growing ammount of kernels that can be exeecuted simulatenously and with faster clock speeds.
One last thing worth mentioning is that gpu used for testing while being on of the better ones, in the grand scheme of things falls short to the newest series which supports four times as many kernel executions as this one and has a faster clock. this means that the 100 time improvement is still achievable with an hardware upgrade.

6 Bibliography

[1] NVIDIA Corporation. Cuda toolkit v10.2.89. `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`.

[2] Paweł Hottowy, Andrzej Skoczeń, Deborah E Gunning, Sergei Kachiguine, Keith Mathieson, Alexander Sher, Piotr Wiącek, Alan M Litke, and Władysław Dąbrowski. Properties and application of a multichannel integrated circuit for low-artifact, patterned electrical stimulation of neural tissue. *Journal of Neural Engineering*, 9(6):066005, nov 2012.

[3] Lauren H. Jepson, Paweł Hottowy, Keith Mathieson, Deborah E. Gunning, Władysław Dąbrowski, Alan M. Litke, and E. J. Chichilnisky. Focal electrical stimulation of major ganglion cell types in the primate retina for the design of visual prostheses. *Journal of Neuroscience*, 33(17):7194–7205, 2013.

[4] Tezeusz Woronko. A massively parallel implementation of the analysis software for data processing coming from retinal stimulation, jan 2018.