Visualization of Neurophysiological Dynamic Communities within the Mouse Brain

BY

MANUEL TANZI
B.S, Politecnico di Milano, Milan, Italy, 2015

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2017

Chicago, Illinois

Defense Committee:

Robert V. Kenyon, Chair and Advisor
G. Elisabeta Marai
Pier Luca Lanzi, Politecnico di Milano

"In good information visualization, there are no rules, no guidelines, no templates, no standard technologies, no stylebooks... You must simply do whatever it takes."

Edward Tufte

ACKNOWLEDGMENTS

I would like to thank my advisor, **Professor Robert V. Kenyon**, that always supported me with enthusiasm, providing very precious advice.

I would like to thank the team that worked with me on this project, **Professor Tanya Berger-Wolf**, **Professor Daniel Llano** and **Umberto**.

I would like to thank my Italian advisor, **Professor Pierluca Lanzi**, that is always willing to help his students during their international experience at UIC.

I would like to thank **Professor Elisabeta Marai** to be on my committee and to have been a great advisor for my Italian friends.

I would like to thank **Professor Andy Johnson**, for his amazing visualization class that I had the pleasure to attend.

I would like to thank **Lynn**, for being a 'guardian angel' for the Italian students at UIC.

I would like to thank **my family**, that allowed me to leave home to start this amazing adventure in the U.S., and all of the good friends that I met here, especially **Yui**.

MT

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF ABBREVIATIONS

ROI	Region Of Interest
SD	Standard Deviation

SUMMARY

The thesis is based on a project whose aim is to study, at a very low level, the mice brains, in particular, the auditory corticothalamic system. This research takes advantage of an innovative technology, the calcium imaging technique, that allows the visualization of mice brain neurons where the firing cells are visible with a high resolution. This is a project developed by the University of Illinois at Chicago in collaboration with the University of Illinois at Urbana-Champaign. The project is supervised by the Professor Robert V. Kenyon and the Professor Tanya Berger-Wolf for the UIC, and the Professor Daniel Llano for the UIUC. The dataset used in this project are movies obtained from the Department of Neuroscience of the UIUC. These datasets are passed through a data processing pipeline, that ends with a visualization tool. This tool provided validation and inspection for the community affiliation of the cells. The focus of this thesis is the development of the visualization tool used as part of the project.

CHAPTER 1

INTRODUCTION

1.1 The Importance of Data Visualization

It is very difficult for humans to capture patterns in datasets presented as spreadsheets or reports. When a dataset is composed by few rows, it is possible to compare them and have an idea of what is happening. But, when the amount of data explodes, it becomes very tough to analyze all those numbers, because of the way the human brain processes information. This is why we visualize data.

Data visualization allows visual access to huge amounts of data in easily digestible visuals. It is used in many different fields. For example, a CEO of a company analyzes many charts when he has to make decisions, while a scientist visualizes the results of his experiments to discover new insights. Thus, the visualization becomes the input for a domain expert mind, that can use his experience to produce new knowledge.

But, the benefits of a good visualization are not limited for experts only. Often, a good visualization allows also a nondomain expert person to find patterns between the data, using only his visual capability. This because the human brain is good in recognizing visual patterns.

Moreover, a good visualization can give the ability to verify if the data were processed in a correct and meaningful way. This last case is what we are going to see in this document.

1.2 The Data Pipeline

To analyze our data, we created a data processing pipeline that concludes with our visualization. The input of the pipeline was a set of movies taken on mice brains.

These movies were produced by the department of neuroscience of the University of Illinois at Urbana-Champaign, precisely by the neuroscientist Daniel Llano, who used the calcium imaging technique on mice brains to film the activations of the neurons.

Then, we processed these movies in order to extract the positions and the time courses of the cells that activate during the movie, using a standard deviation based technique.

The time courses of the cells were then used by our data mining team to compute, through some algorithms, the correlations and the communities of the cells. These, together, represent the reconstructed brain network. This construction of the network is based on a relationship measure, in our case the correlation, that determines if each pair of nodes need to be connected by an edge or not, and the weight of this edge. Each node of the network represents a cell.

Then, finally, at the end of the pipeline, the network and the other data were visualized by our visualization tool. This part resulted critically important because it was useful to verify the correctness of the algorithm used in the previous step. Moreover, the neuroscientists can interactively explore the data in different visual representations and have the possibility to control some parameters to modify the network in real-time, in order to extract new insights [1].

To summarize, the steps of the Data Analysis Pipeline are:

1. Data extraction from the movie.

2. Computation of the relationship measure. In our case, the correlation.
3. Creation of the network using the relationship measure.
4. Computation of the communities inside the network.
5. Visualization.

These steps contain a loop. This because, after the visualization step, two outcomes are possible:

1. Some good information is extracted from the final visualized data, increasing our knowledge of the brain function.
2. The created network results to be imprecise if compared with the ground truth data, i.e. the time courses of the cells. So, there is the need to modify the algorithm that computed the network or to choose another relationship measure: return to step 2.

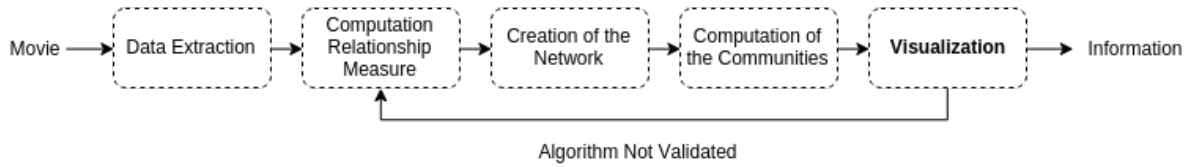

Figure 1: Data processing and visualization pipeline

We will see in the experiments section that the visualization tool helped to notice that the correlation was not a precise factor in building the network since it created many edges between

cells that should not be connected. Based on this finding, the data mining team could start to pursue other directions.

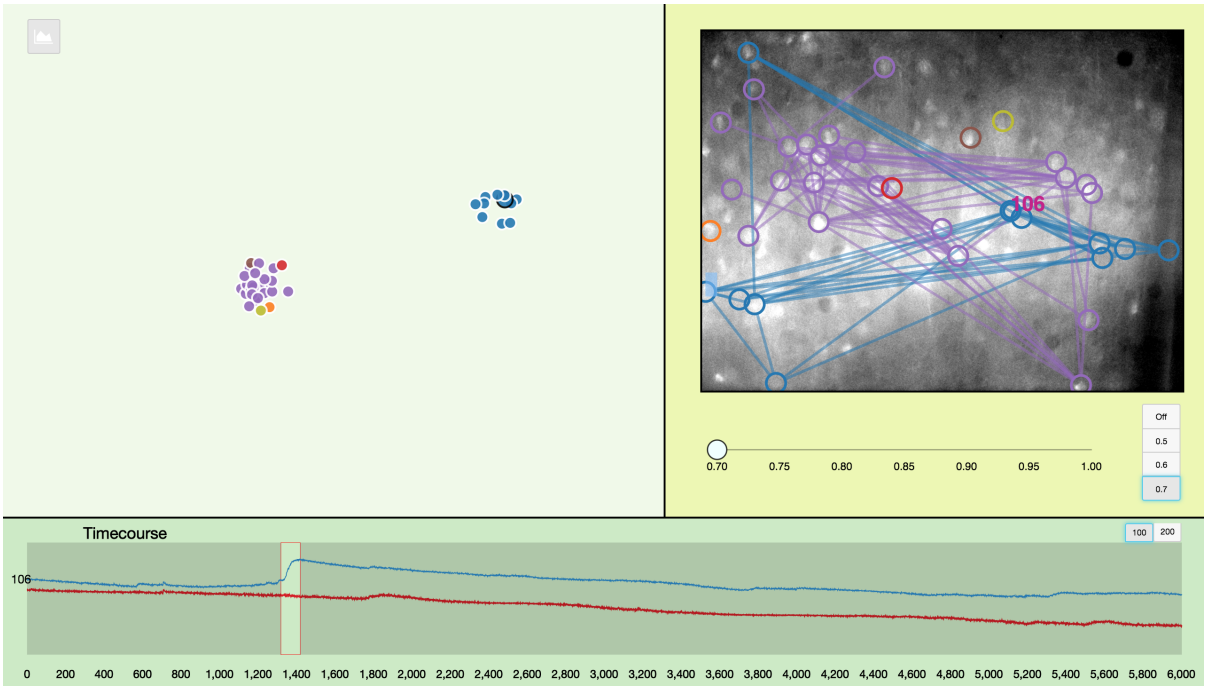

Figure 2: Overview of the visualization tool

CHAPTER 2

CALCIUM IMAGING TECHNIQUE

The calcium imaging is an advanced technique that can be used to record cellular activities inside the brain:

Ca^{2+} imaging techniques enable us to measure the temporal variation in the intracellular Ca^{2+} concentration (Grynkiewicz, Poenie, Tsien, 1985). In the case of nerve cells, the intracellular Ca^{2+} concentration is closely related to the membrane potential of cells because Ca^{2+} is recruited inside through voltage-dependent Ca^{2+} channels whose conductivities depend on the membrane potential. Therefore, the instantaneous elevation of the intracellular Ca^{2+} concentration gives us important information on the time of action potential generation. Many research groups have developed multi-cellular Ca^{2+} imaging systems to record individual cellular activities of a cell assembly in vitro and in vivo. For example, Ikegaya et al. (2004), Ikegaya, Le Bon-Jego, and Yuste (2005) recorded the spike times of hundreds of cortical neurons in in vitro Ca^{2+} imaging and discovered a repeated firing sequence from particular groups of neurons. Dombeck, Harvey, Tian, Looger, and Tank (2010) recorded in vivo hippocampal CA1 neurons of a moving rat and showed the spatial distribution of place cells. [2]

In our case, the calcium imaging technique was used by the Department of Neuroscience at the University of Illinois at Urbana-Champaign, in particular by the Professor Daniel Llano to film the auditory corticothalamic projection of the mouse brain. Then, the movie was sent to us by Professor Llano, and we used it as input for our data processing and visualizing pipeline.

CHAPTER 3

DATA EXTRACTION FROM MOVIE

We received the movie from the neuroscientist as a sequence of thousands of frames in TIF format, like the one showed in Figure 3. The data that we wanted to obtain from these, and then visualize, were the time courses of the cells, i.e. a time series for each cell representing its brightness in the movie over time, and the positions of the cells, i.e. the coordinates of their centroids.

Three steps were necessary to pass from the raw movie to the time courses, plus a step to extract the coordinates of the centroids of the cells.

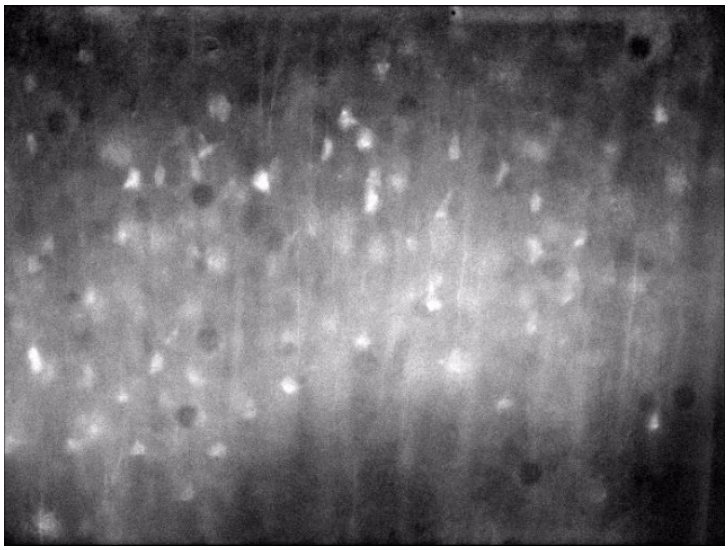

Figure 3: One frame of the movie

3.1 Composed Image Generation

The movie obtained with calcium imaging needed to be composed in a single image that shows all the cells that activate during the movie. This because the obtained image could be passed again to the neuroscientist that, thanks to his knowledge, was able to trace the regions of interest representing the area occupied by each single cell.

Each cell occupies a certain portion of the pixels in each frame. This means that, if a cell is firing in a certain frame of the movie, its pixels will have a higher brightness value with respect to the normal value that they have when the cell is not firing. Using this information, it is possible to observe that the standard deviation of the brightness of each pixel during the entire movie could determine if that pixel belongs to a firing cell or not. This because, as said before, during the activation of a cell the brightness value of its pixels will increase rapidly, causing a higher standard deviation of that pixels with respect to a non-cell pixel. So, we decided to compose the entire set of frames to create a single image where the value of each pixel is equal to the standard deviation of that pixel computed through the entire movie. Since the images were in gray-scale color, we can consider each image as a matrix of $W \times H$ values, where each value represents the brightness of the pixel in the row i and in the column j. Using this notation, formally, we computed the composed image of standard deviations SD in this way:

- Given $F_1, F_2, ...F_n$ matrixes $W \times H$ representing n frames.
- A final matrix SD $W \times H$ representing the composed image.
- $\forall\ i, j,\ SD(i,j) = \sqrt{\frac{\sum_{t=1}^{n}(F_t(i,j) - \overline{F}(i,j))^2}{n}}$ where $\overline{F}(i,j) = \frac{\sum_{t=1}^{n} F_t(i,j)}{n}$

Then, we observed that the image obtained with the standard deviation technique resulted difficult to analyze, since the cells, even if present, appear to be very dark. So, we decided to increase the contrast of the composed image, to make the cells more visible. To do this, we made use of the MATLAB function imadjust. Imadjust work in this way: "scales intensities linearly such that 1% of pixel values saturate at black (0), and 1% of pixel values saturate at white" [3]. The result, showed in Figure 4, was good, but not complete. This because the cells that reached the highest brightness during the movie, resulted in having a very high standard deviation, making less visible those cells that, even if activated during the movie, resulted with a lower standard deviation because they reached lower level of brightness.

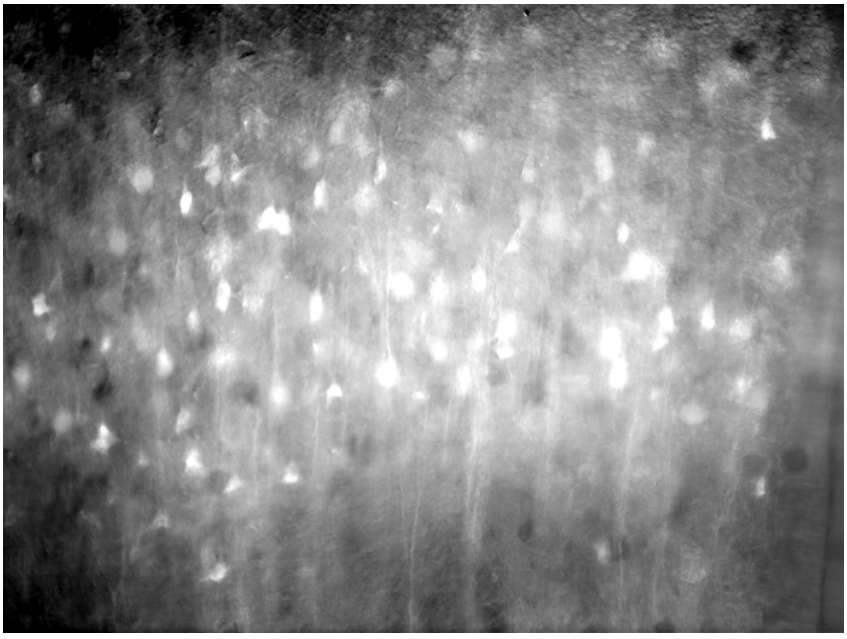

Figure 4: Standard deviation post contrast enhance

So, in order to spot also the other cells, we needed a method to reduce the standard deviation of the brightest cells only. We managed to do this using again the imadjust function. We increased the contrast of each frame before of the composition with the standard deviation technique, instead of increase the contrast of the composed image.

Increasing the contrast in each frame makes very bright those cells that in the previous composed image had the higher standard deviation, while the cells in the dark areas become just more visible. In this way, when we computed the standard deviation using these enhanced contrast frames, we obtained, for the very bright cells, a low standard deviation, while for those cells with an improved visibility we obtained a high standard deviation. This is explainable because a pixel is visible in the composed image if its standard deviation value is high with respect to the standard deviation of all the other pixels in the composed image. In the image obtained using the frames without the contrast enhance (Figure 4), the standard deviation of the pixels in the central area was very high, so the cells that activated in the darker area were not visible. Instead, in this composed image, the standard deviation of the pixels in the central area was very low, because their values remained very high during the entire time course, and so, as showed in Figure 5, the activated cells in the darker areas became visible, cells that we weren't able to see in the previous composed image.

So, the two obtained pictures are complementary, since they shows different cells. We combined them to obtain most of the cells that activate during the movie.

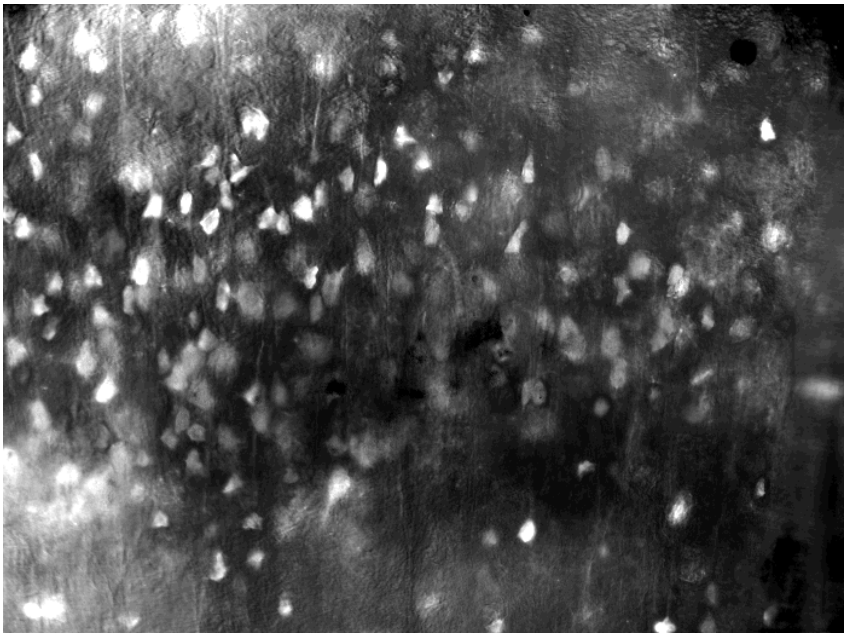

Figure 5: Standard deviation pre contrast enhance

3.2 ROIs and Labeling

After we obtained the two composed images, representing the cells that activated during the movie, we passed them to the neuroscientist, because we needed his knowledge to determine which illuminated areas in the composed image represent a cell and which not. So, he produced a Regions Of Interest binary image, with the same dimensions of the composed images, where each pixel was set to true if it belongs to an area representing a cell, and false if not. The ROIs image is showed in Figure 6.

Figure 6: Manual ROIs

Then, when we received this image from the neuroscientist, we needed to label each ROI. We used a MATLAB function that assigns labels in order from left to right and from top to bottom. The labeling is very important since in the visualization is important to know which cell of the frame we are exploring and because we wanted to associate to each cell a time series, a set of correlations with each other cell, and a community.

3.3 Centroid Coordinates Extraction

For our visualization tool, we needed to know the coordinates of the physical position of each cell in the frames. This because, as we will see, there is a section of the visualization tool

that shows the constructed network projected on the observed frame of the movie. To extract the coordinates of the cells, we computed the centroids of their ROIs.

3.4 Timecourses Extraction

Once that we finally had the physical positions of each cell in the frames, and that we knew their areas, we were ready to generate the time courses of the cells. This is the main dataset of the project since it fed the algorithm that computes the correlations and the communities, i.e., the reconstructed network. We decided to compute them assigning, to each cell, a value of brightness in each frame equal to the average value of its ROI pixels brightness in that frame. We obtained a matrix, in CSV format, where each column represents a cell, identified by its label, and each row represents a frame. So, for example, the value in the position at row 3 and column 7 represents the average of the pixel values in the ROI of the cell with the label number 7 at the frame number 3.

CHAPTER 4

INPUT OF THE VISUALIZATION

After we completed the processing phases explained before, including the computation of the correlations and the communities performed by our data mining team, we ended up with 4 different datasets.

4.1 The Datasets

The first 2 datasets are composed of multiple files, since they change in every timestamp, while the last 2 consist of single files because their data remain constant in each timestamp.

4.1.1 The Dataset of the Correlations

This dataset consists of a series of files, one file for each timestamp, containing the correlations between each cell. The correlation matrix is complete, so, there is a value of correlation for each pair of cells.

The file number represents the first timestamp of the window in which the correlations are computed. So, for example, the file number 40, if we are using a window with a length of 100, will contain the correlation computed using the time courses of the cells from the timestamp 40 to the timestamp 139.

4.1.2 The Dataset of the Communities

This dataset consists of a series of files, one file for each timestamp, containing the corresponding community for each cell. Since these communities are computed using the correlation

files, also for them the file number represents the first timestamp of the window in which the correlations are computed.

4.1.3 The Dataset of the Physical Coordinates

This dataset consists of a single file since it contains the coordinates of the center of each cell, computed using the frame containing the Regions Of Interest. Since the physical positions of the cells remain constants through the movies, there is no need to have different data for each timestamp.

4.1.4 The Dataset of the Time Courses

This dataset consists of a single file containing the time course of each cell. That means that for each cell, there is a list of values, and each value represents the average brightness of the cell pixels during a certain timestamp. So, the file is a table consisting in N rows $\times$ M columns, where N is the total number of timestamps of a time course and M is the number of cells in the dataset.

4.2 Datasets Format

In this section, we explain the syntactical structure of the 4 datasets. This is important because the visual interactive application extracts the data from them, following some precise parsing rules.

4.2.1 Dataset of the Correlations

This dataset consists of N separated files, where N is the total number of frames minus the length of the window used to compute the correlation. Each file is structured in this way:

- There are M rows, each one representing the correlations between two cells. So, if there are K cells, each file will have K^2 rows.
- Each row contains 3 numbers, that can be separated by any number of spaces or by a comma.
- The first two numbers are the labels of two cells and the third number is their correlation.

4.2.2 Dataset of the Communities

Since it is computed using the Dataset of Correlations, also this dataset consists of N separated files, where N is the total number of frames minus the length of the window used to compute the correlations. Each file is structured in this way:

- There are K rows, each one representing the community of a cell. So, if there are K cells, each file will have K rows.
- Each row contains 2 numbers, that can be separated by any number of spaces or by a comma.
- The first number is the label of the cell and the second number represents the community which that cell belongs to.

4.2.3 Dataset of the Physical Coordinates

This dataset consists of a single file. This file is structured in this way:

- There are K rows, each one representing the position of the centroid of a cell in the frames. So, if there are K cells, each file will have K rows.

- Each row contains 3 numbers, that can be separated by any number of spaces or by a comma.
- The first number is the label of the cell, the second number is the x coordinate of the cell centroid and the third number is the y coordinate of the cell centroid.

4.2.4 Dataset of the Time Courses

This dataset consists of a single file. This file is structured in this way:

- There are $N + 1$ rows, each one, except for the first, representing a timestamp of the time courses, or, equivalently, a frame. So, if the movie is composed by N frames, the file will have $N + 1$ rows.
- The first row contains K numbers, that are separated by a comma, each one representing the label of a cell.
- Each successive row contains K numbers, that are separated by a comma since it is a CSV file. Each one of these rows represents a frame.
- Each number in a row is the value of the brightness of the cell with the label equal to the number in the first row of that number column, in the frame with number row - 1.

CHAPTER 5

THE VISUALIZATION TOOL

As we have seen, we had many different data to visualize, ranging in thousands of frames and several correlations. So, instead of creating a series of static visualizations, we decided to build an interactive visualization application, that allows exploring all the data.

If we had just used the data to plot some static graphs, probably we wouldn't have obtained so much. This mainly for two reasons:

- First, it is very hard to know a priori which data are interesting to plot. The data needs to be visually explored in order to discover interesting patterns. If we had known before what is interesting to visualize, probably we would not have had so much need for a visualization.
- Second, we are computer scientists, not experts of the domain of study. The creation of an interactive application moves the important task of choosing which data to visualize from the developer to the user, since the tool let the user to decide which data are visualized and where to perform a deep exploration of them [4].

Moreover, we decided to build this as a web application, accessible by anyone using a modern browser like Chrome. Most of the existing tools for data visualization need to be downloaded and installed. The advantages of having a web application are that it is accessible everywhere you dispose of an Internet connection and that it is simpler. This is an important aspect because

the tool won't be used only by computer scientist, but especially by researchers that need an effective and efficient way to visualize their data. Moreover, if they have, for any reason, to work on another computer, they don't need to install again the application, the tool is always on the web.

Those aspects offer a great user experience for those people that don't want to spend their time in a complex installation of weight software but want to immediately jump to play with the data.

5.1 Layout

We built the visualization tool in such a way that all the useful information are presented to the user in the simplest, but still meaningful, possible way. We had 4 main datasets to visualize, plus the frames of the movie:

- The correlations between the cells.
- The communities of the cells.
- The physical position of the cells.
- The time courses of the cells.

Assigning too many visual attributes to a single graph to encode all the information could have brought to a visualization difficult to understand. This because there are visual attributes that are easier to percept for the human brain, like position and color, and attributes that are more difficult to get, like shapes, in particular, if someone is searching for patterns. Moreover,

sometimes it is useful to visualize the same dataset in different ways since it can bring to different visual information, as we will see.

So, to maintain the application visually approachable, and to analyze the data from different prospectives, we decided to divide the application into 3 main sections (Figure 7).

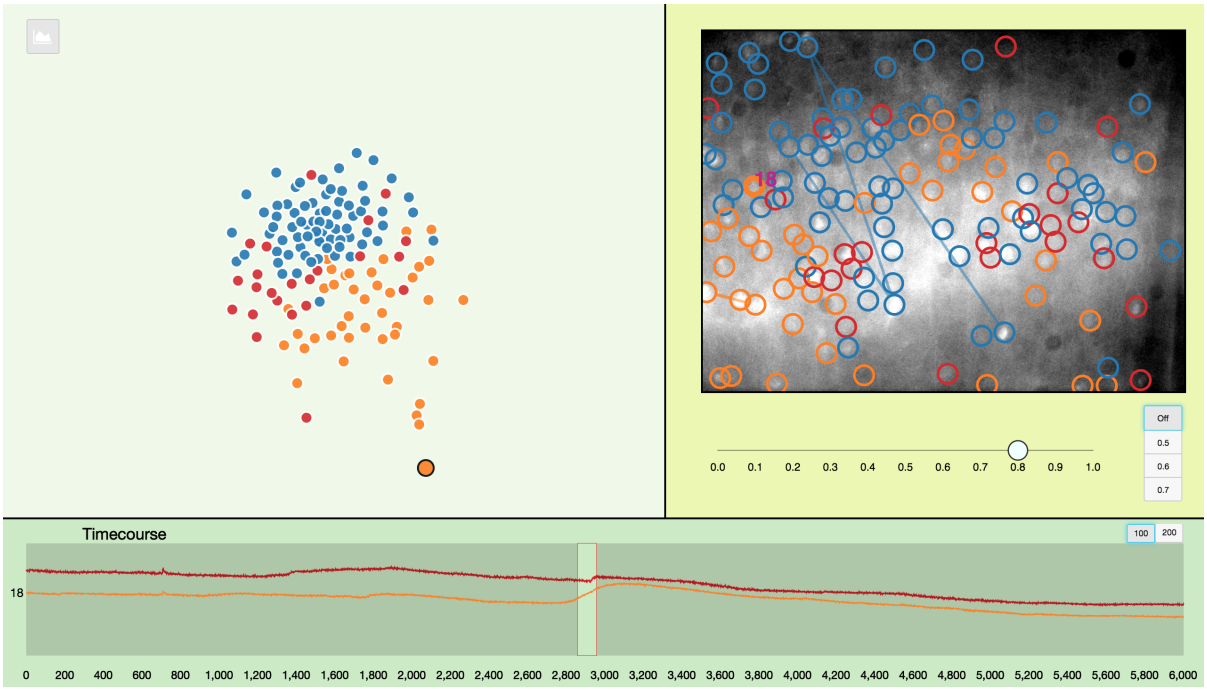

Figure 7: The interactive web application is composed by 3 separate main sections

5.1.1 The Projection of the Network on the Image Frame

On the top right of the web application, the user can find a quadrant containing a network projected over an image. This is the graph representing the physical structure of the network. It is projected on the current frame, the one at the start of the analyzed window.

This graph uses the dataset of the physical positions of the cells to map the nodes of the network on the image frame. The nodes are represented as empty holes so that the real cells in the frame are observable inside the relative nodes. It also uses the dataset of the correlations to visualize the edges between the cells if their respective correlation it's above a certain threshold. This threshold is variable and the user can easily modify it using a slider located under the projection.

The nodes are represented with the color of the community, but, while in the force graph on the left the user can observe the distribution of the communities with respect to the correlations, in this graph he can observe the distribution of the communities with respect to the physical position of the cells. Since the algorithm that computes the communities produces different datasets on different correlation thresholds, the user can use the selector on the right of the slider to choose between different meaningful outputs of communities computed with 3 different threshold cuts or to keep the communities computed without any cut on the network, turning off this functionality.

5.1.2 The Force Graph

On the top left of the web application, the user can observe a graph composed of colored circles that present different distances between them.

This graph is a force graph, a special network based on attraction forces, that uses the dataset of the correlations between the cells to determine the positions of the nodes, and the dataset of the communities of the cells to determine the color of the nodes. Thus, it exposes how the cells group with respect to the correlations and how the communities distribute on the cells. You can also see a button on the top left that can be used to show only the cells that present the faster variation in their time courses during the analyzed window.

5.1.3 The Multi-Series Line Chart of the Time Courses

Finally, on the bottom, the user can find a multi-series line chart. This is the graph representing the time courses of the cells during the movie, i.e. the average brightness of their pixels for each frame. It obviously uses the dataset of the time courses of the brightness of the cells, to draw the relative line chart, and the dataset of the communities, to give the color to the lines. It can be used to observe where the cells fire, and to compare at the same time the time courses of different cells, to analyze their relationship. This graph is also the timestamp or frame selector since it allows the user to click on any point of the time course to move the analyzed window to that point, loading the correlations and communities datasets relative to the newly analyzed window. Finally, since we computed the correlations and the communities either with a window of length 100 and 200 frames, there is a selector on the top right that allows the user to change the length of the analyzed window, loading the relative datasets.

5.1.4 The Data

These 3 sections are separated, but the bounded data are the same, so that, for example, the correlations data visualized on the force graph are the same of the ones that build the physical network, and the communities are the ones computed with that correlations. Thus, the 3 sections evolve together, with respect to a common timestamp controlled by the user. The timestamp determines the start of the window in which the currently visualized correlations and communities are computed. That window is visualized on the multi-series line chart as a red quadrant and adapts if a different window length is selected so that the user can be aware of the portion of the time course used to compute the currently visualized correlations and communities. The active timestamp corresponds to the frame of the movie that is currently visualized under the physical graph so that the active data in the visualization reflects what is physically happening in that moment.

5.2 Projection of the Network

This section visualize a network, that is built in this way:

- The network is an undirected graph $G = (V, E)$, where V are the vertices of the graph and E the edges.
- Each $v \in V$ represents a cell of the analyzed brain section, and the dimension of the set $|V|$ is equal to the number of cells.
- For each pair (x, y), where $x, y \in V$, the unordered pair $\{x, y\}$ belongs to the set of the edges of the graph E if $corr(x, y) > t$, where $corr(x, y)$ is the correlation between the cells corresponding to the vertices x and y and t is a threshold that can be set by the user.

- There is a set of communities $C = \{c_1, c_2, ..., c_n\}$, and for each $v \in V$, $\exists! c_i \in C \mid v \in c_i$. The color of the vertices of the graph is determined by the belonging community.

The representation of the data in such a way that the observer can quickly find patterns, thanks to his visual ability, is the main purpose of information visualization [5], and this network built using a threshold on the correlation to determine the edges gives a rapid understanding on which are the groups of correlated nodes. But for neuroscientists, it is also important to be able to relocate patterns or interesting behavior in the real physical structure of the brain. They could do this looking at labeled data, and then comparing with the previously labeled images but this would not be so practical and it would take a lot of time. So, we decided to create this visualization of the network that is projected on the currently visualized frame. In this way, the user has an immediate idea of what is physically happening in the brain during each timestamp.

This is a similar operation to the one made for fMRI images, where the visualization of the activated areas of the brain is projected on the picture of the physical brain [6]. The main difference is that in fMRI images, each pixel represents several neurons. In our case instead, the resolution is much higher, and one pixel partially represents a cell, since each cell occupies a certain portion of the frame. Moreover, there are pixels that stay in empty spaces, i.e. spaces not containing any cell in the frame, so they do not represent a cell.

So, in order to have the nodes of the network visualized over the physical cells in the frame, the tools set the position of the nodes on the image using the coordinates of the centroids of each cell. An example is showed in Figure 8.

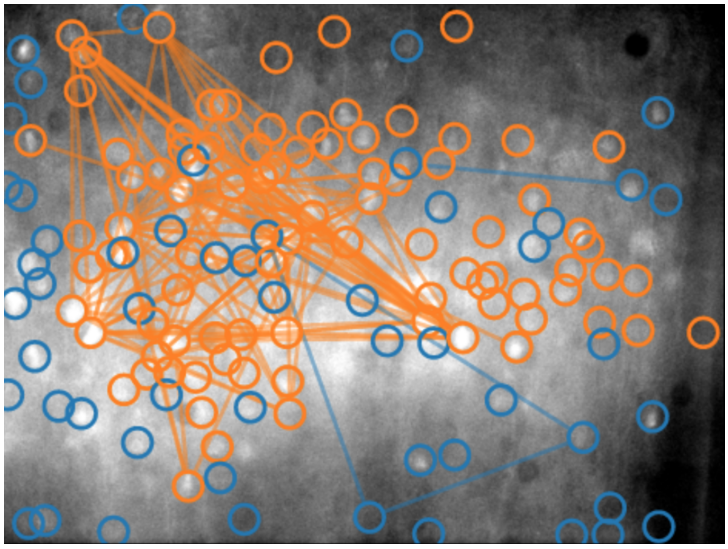

Figure 8: The network projected on the current frame

The nodes are represented with empty circles, as showed in Figure 9, in such a way that the user is able to look at the physical state of the cell, i.e. the level of brightness of its pixels in the frame.

The color of the circle represents the community which the cell belongs to. In this way, it is possible to see the physical distribution of the communities within the brain section.

As explained before, the visible edges depend on the correlations between the two cells of the nodes that each edge connects. A correlation threshold can be set by the user using the slider under the visualized frame. This slider goes from 0 to 1, and when it is moved, it updates the correlation threshold and automatically updates the network, showing in real-time the new one.

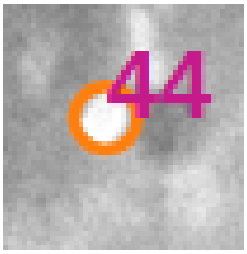

Figure 9: A node represented by a circle on the relative cell. The number is the label of the cell

The color of an edge represents instead the relation between the communities of the two connected nodes: if the communities are the same, the color of the edge will be the color of the two nodes, to represent that the cells belong to the same community. If instead, the two connected nodes belong to different communities, the color of the edge will be black, as showed in Figure 10. This is useful because it could visualize the fact that the algorithm that computes the communities, which is based on the chosen relationship measure, in our case the correlation, could not working in the desired way, if there are too many black edges, since two nodes of different communities should have a weak relationship.

The user has also the possibility to visualize only the nodes connected to a determinate node instead of visualizing the entire network. To do this, he just needs to go with the mouse on a cell, and its relative subnetwork will be visualized, as showed in Figure 11.

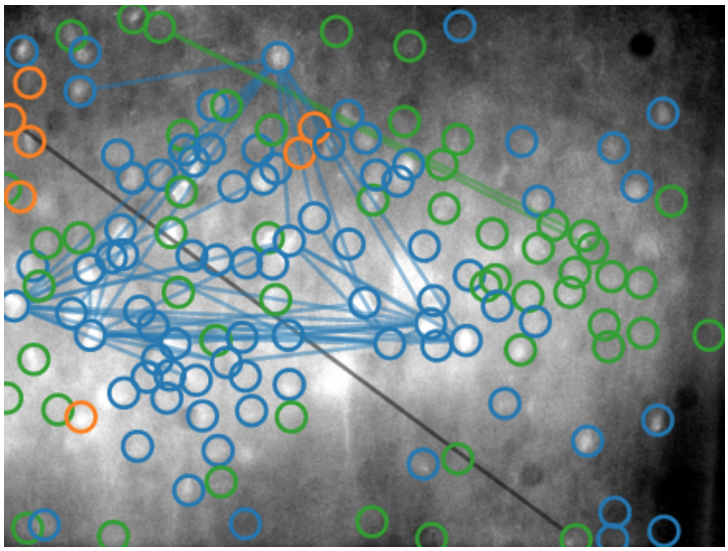

Figure 10: A black edge representing a connection between two cells belonging to different communities

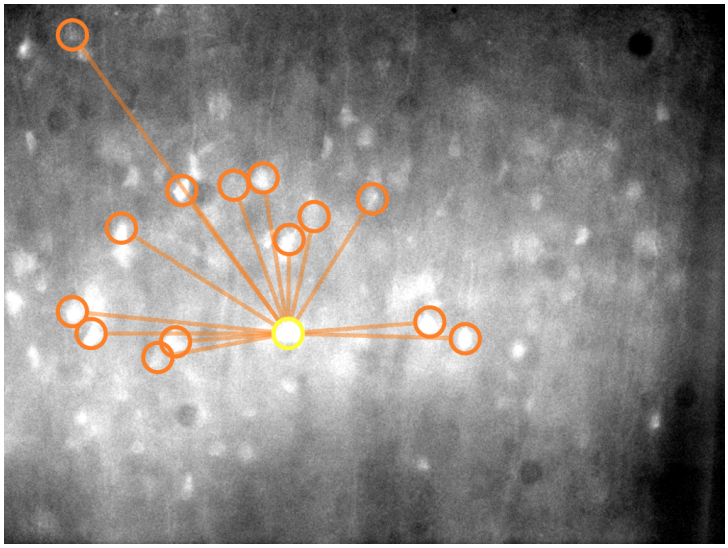

Figure 11: The links of a single cell

5.2.1 Correlation Cut and Communities

As introduced before, the slider can be used to set the correlation threshold and, so, to interactively visualize the network with only edges between nodes with a correlation higher than the threshold. If this helps to have a good visualization of the network connections, it doesn't help so much in terms of communities. In fact, not always the algorithm that generates the communities have the same output with different correlation thresholds, like in our case. The algorithm that we are using makes use of a technique called Louvain Community Detection to generate the communities [7]. Since this method is not only based on the weight of the edges, but also on the presence of the edges between two nodes, it is clear that the communities generated on the entire network will be different from the one generated on a network with some edges cut.

To overcome this problem, since it takes days to compute those communities, we chose 3 meaningful correlation cuts, set at 0.5, 0.6, 0.7, and precomputed the communities for those three different cuts. We also computed the communities obtained without any cut on the network. The user can use the selector on the right of the slider (Figure 12) to choose the value of the correlation cut or to turn this functionality off. If the user selects off, the communities will be the one computed on the entire network without any cut. Instead, if the user selects a cut, the communities generated with that cut will be loaded and the visualization updated.

Indeed, if we load the communities computed on a network with a certain cut, would make no sense to set the correlation threshold of the visualized network under the value of that cut, because all the edges under that value were not considered by the algorithm. So, we made

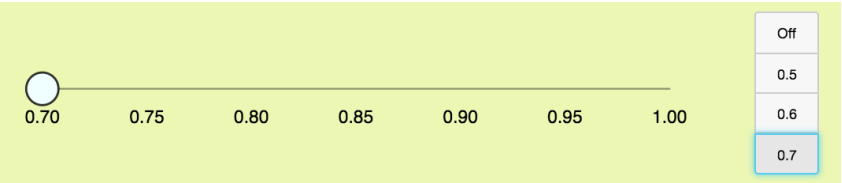

Figure 12: On the left, the correlation slider, that can be used to change the correlation threshold. On the right, the cut selector, that can be used to load the datasets of communities produced on a network having only edges with weights higher than the cut value, or that can be switched off to have the communities computed on the fully connected network

the slider to dynamically change such that its minimum value it is equal to the value of the correlation cut, and it returns to be equal to 0 when this functionality is turned off. For example, setting a cut of 0.7, the slider will mutate to have a range from 0.7 to 1, as shown in Figure 12.

5.3 Force Graph

The network projected on the graph that we have seen before has two big limits:

- It depends on the correlation threshold set by the user.
- Doesn't offer a good visualization of the groups of cells highly correlated between them.

In fact, if we want to identify the groups of most correlated cells using the projected network, to have a global view, we need to go through many plots of the network changing every time the correlation threshold, until we found interesting patterns. It makes really hard to determine at a glance who has a lot of connection and instead who is isolated. With that kind of graph, we have a good idea of the physical distribution of the network and the communities, but a bad representation of the groups of highly correlated cells, as we can see in Figure 13.

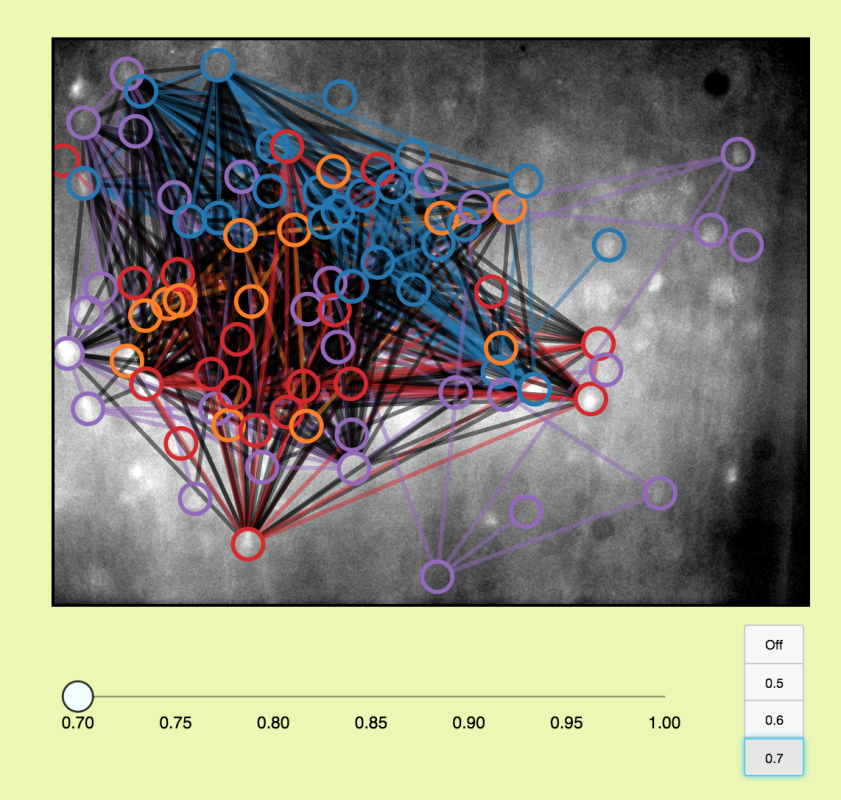

Figure 13: Even with a high correlation threshold, 0.7, the network results to have too many edges, and it's hard to get which are the interesting groups of correlated cells

To overcome these two limits, we created a specialized Force Graph, based on the force-layout of D3.js. A force-layout is a physic based simulation that set forces between nodes based on the length of the edges that connect them and, following these forces, computes a good arrangement for the entire structure. Then there is a gravity that keeps the structure close to the center of the SVG container.

The nodes represent the cells, while the correlation is represented by the intensity of attraction between the nodes. What we wanted to obtain is that cells with higher correlation tend to be more attracted, so that they result closer in the Force Graph, while cells with lower correlation tend to be less attracted, so that they will appear farther in the graph. To enforce this behavior, we compute the length of each edge in a way inversely proportional to the correlation of the cells represented by the two nodes that it connects.

Formally, we set the length of the edges in this way: (1 - corr(A,B)) * K, where A and B are the two cells represented by the nodes that the edges connect, corr(x, y) is a function which output is equal to the correlation of the two parameters x and y, and K is a constant.

Then, those lengths are used by the force-layout to generate the forces that will determine the spatial distribution of the nodes. We can see, in Figure 14, a comparison between the Network representation and the Force Graph. In the first, on the right, is almost impossible to understand the relationships between the cells, but it gives a good visualization of the physical distribution of the communities. The second, on the left, gives instead a good visualization of the group of cells highly correlated, but the information about the physical position of the cells is lost.

There is another way, more traditional, to visualize a matrix of correlations between a set of nodes: the correlation heatmap [8]. This kind of heatmap is able to give some information about the distribution of the correlation values. The main weakness of a correlation heatmap is that it cannot provide meaningful global patterns, since, to visualize the intensity of correlation, it assigns a different color for every couple of objects. So, the groups of cells with the same

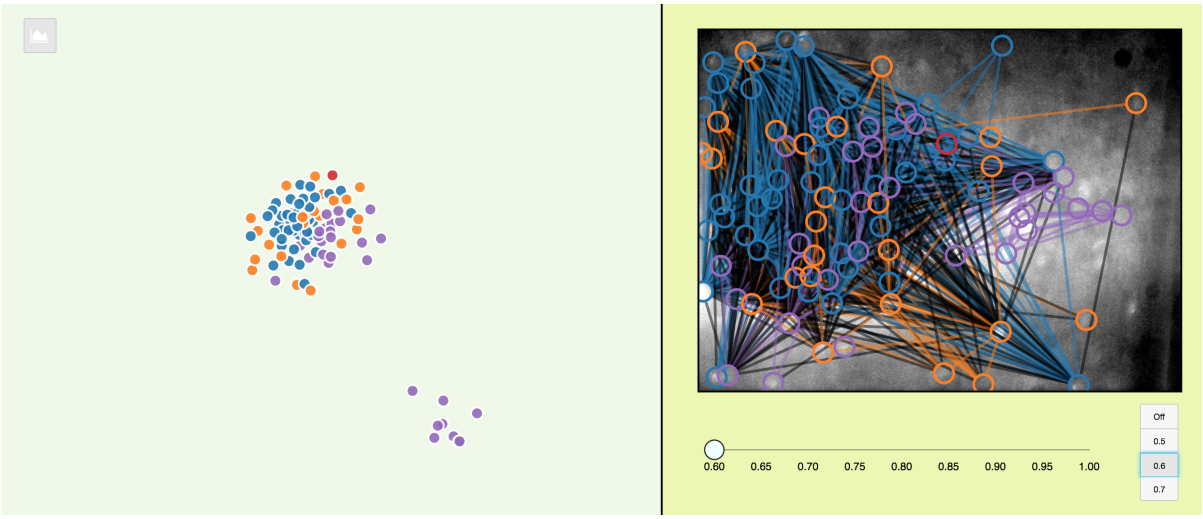

Figure 14: On the left: the force graph of the network. Cells with higher correlations tend to stay closer, while cells with lower correlations between them tend to repulse. So, the interesting group of highly correlated cells results well defined. On the right: the network projected on the frame. It is hard to get the groups of correlated cells, but it gives a view of the physical spatial distribution of the communities

color that your eyes will capture in the matrix will depend on the order of the items on the rows and the columns, and that is totally arbitrary. We will show this problem in the next section, that compares the use of a traditional heatmap of correlations with our Force Graph.

5.3.1 HeatMap VS. Force Graph

In this little experiment, we are going to show why the Force Graph is a better choice to visually represent the distance measure between multiple nodes with respect to an HeatMap. In this case, our distance will be the correlation, so, we compare a Force Graph of correlations with a correlation HeatMap, that are respectively showed in Figure 15 and Figure 16, to determine which one is a better solution to visualize our data.

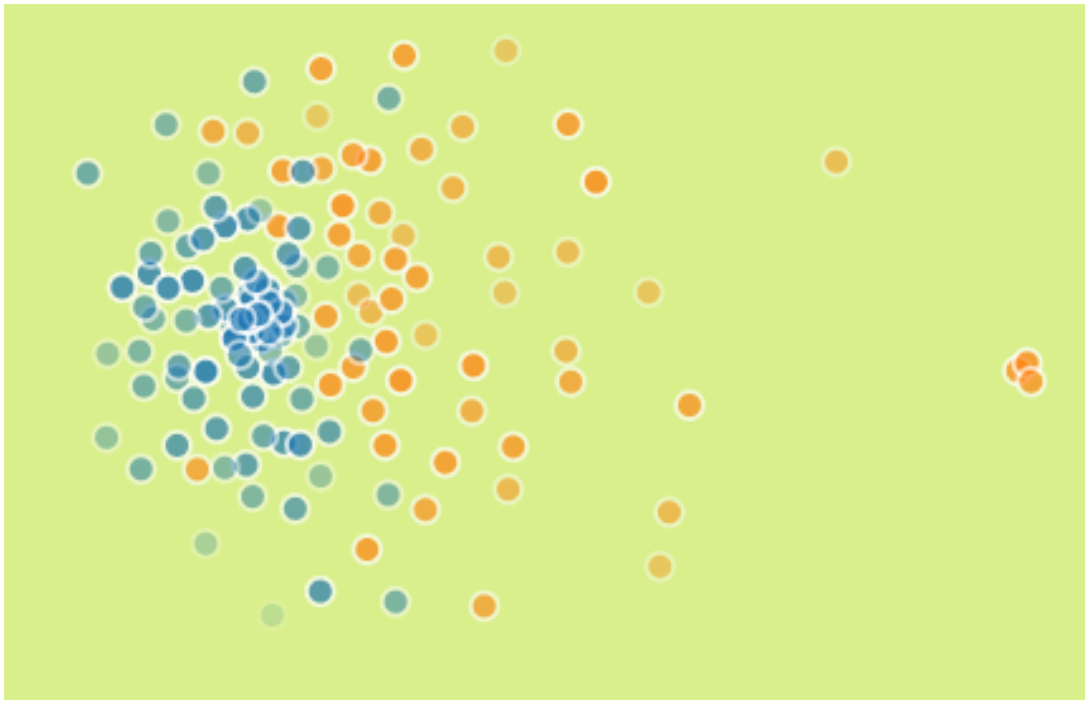

Figure 15: The 3 orange nodes on the right are very close and separated from the rest, since are really correlated between them and low correlated with the others. It was really easy, for our eyes, to percept that group of highly correlated cells

Is well known, since the first studies on visual perceptions made by the Gestalt School of Psychology, in 1912, that humans perceive objects that are close together as a groups [9]. This becomes totally clear when we look at the Force Graph. In particular, if we look at Figure 15, it is possible to see at a glance that the 3 orange nodes on the right are very correlated, while in the correlations HeatMap, it is very hard to notice similar group patterns.

It is true that also objects that share similar attributes, like the color, are perceived as a group, but in the correlation HeatMap, this fact becomes a deceptive attribute. This because

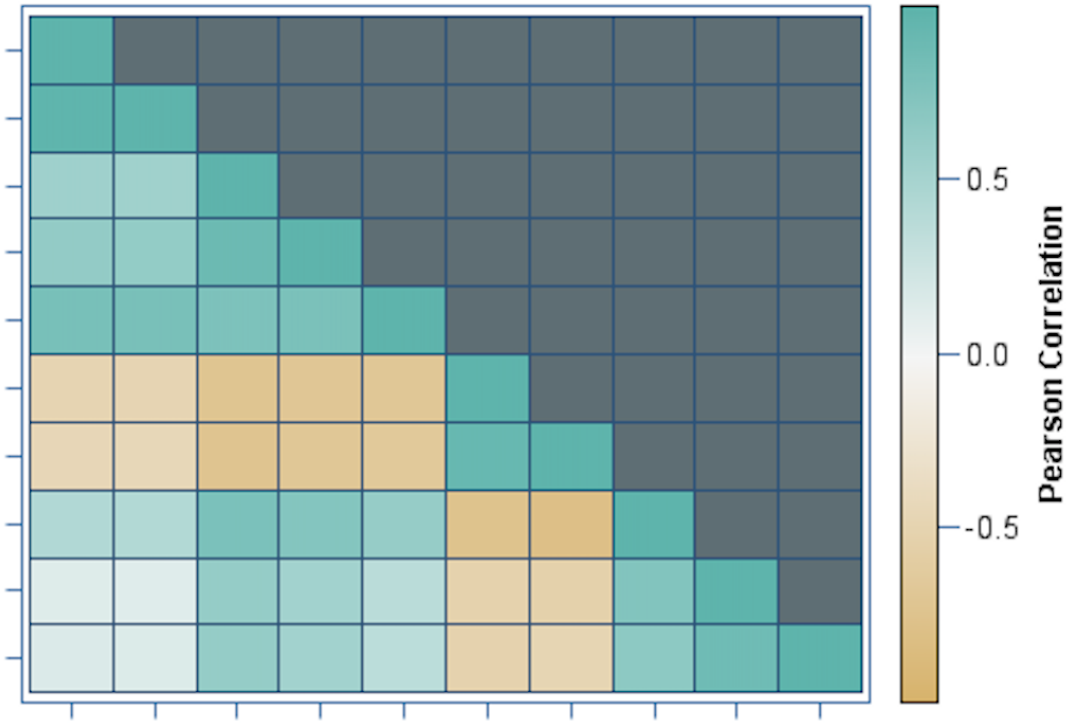

Figure 16: An heatmap of correlations. The distribution of the colors in the matrix space totally depends on the order on which the compared items are presented in the matrix, thus, it is totally arbitrary and provides no useful global information for the observer. The only information that you can get is a pairwise one, i.e., the correlation between each couple of items

the color is determined by the pairwise correlation between the node represented by the row and the node represented by the column. But if nodes A and B have a correlation of 0.5 between them, and also nodes C and D have a correlation of 0.5 between them, the relative squares in the matrix, (A,B) and (B,C), will have the same color, but, in fact, there isn't any relationship between them.

We have another benefit in the force graph with respect to the HeatMap. In Data Visualization, the numbers of good visual attributes that can be used to visualize attributes of the data is limited. In the case of the HeatMap, we are "wasting" the attribute of the position in the space, because, as said before, the position is just used to determine which elements of the rows and the columns correspond to that square, so, it totally depends on the order of the elements, that is arbitrary. So, we have to use the color hue as a visual attribute to represent the correlation values. In the Force Graph this is not necessary, so, it is possible to use the color hue of the nodes to represent another attribute of the data. In our problem, this is fundamental, since we have to visualize also the belonging communities of the cells.

In conclusion, the visualization of correlations through a Force Graph offers a really good global view of the relations between the elements, obtaining an automatic visual clustering of the nodes, through the relative positions, that the common correlation HeatMaps don't offer, and frees the color hue visual attribute, giving the possibility to use it for other purposes, like showing the communities.

5.3.2 Communities

As introduced before, this visualization also shows the community of each cell. The communities are visualized on the Force Graph using the color of the nodes. This technique gives a very good representation of the relationship between the communities and the correlations between elements, since, in the case of strong relations, well defined colored clusters will appear in the visualization.

5.4 Multi-Series Line Chart

This section visualizes a multi-series line chart, that is built in this way:

- There is a set of active lines L, and a dynamic line dl.
- Each $l \in L$ is a line relative to a cell selected by the user in the network or in the force graph, and it represents the entire time course of that cell.
- dl is the dynamic line relative to the last cell that the user hovers with the mouse in the network or in the force graph, and represents the entire time course of that cell.
- Each line is built selecting for each x value, representing a timestamp, an y value that is equal to the value of the brightness of the relative cell during that timestamp.

We can see the section regarding the visualization of the time courses of the cells in Figure 17. This quadrant contains a multiple dynamic line chart, where a line represents the time course of a particular cell.

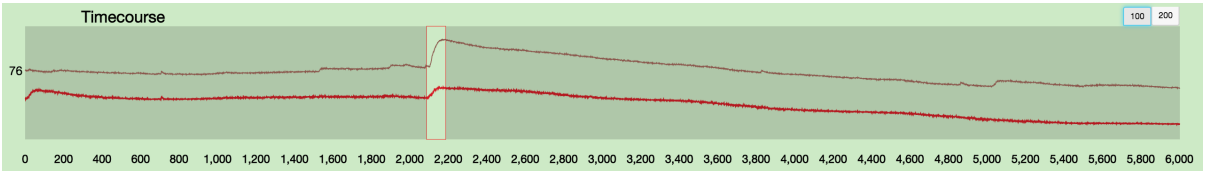

Figure 17: The time courses section of the tool

The red line has a stroke wider than the others line, and represent the navigation line. It is useful to know which is the time course of the cell pointed by the user mouse pointer, and it changes dynamically when the cursor passes from a cell to another.

Then, the user has the possibility to fix other lines. In order to do this, he has to click on a node in the force graph or in the network graph. When a node is clicked, its time course is fixed in the line chart, with the color of its community. This functionality gives the possibility to compare the time courses of different nodes at the same time, as we can see in Figure 18.

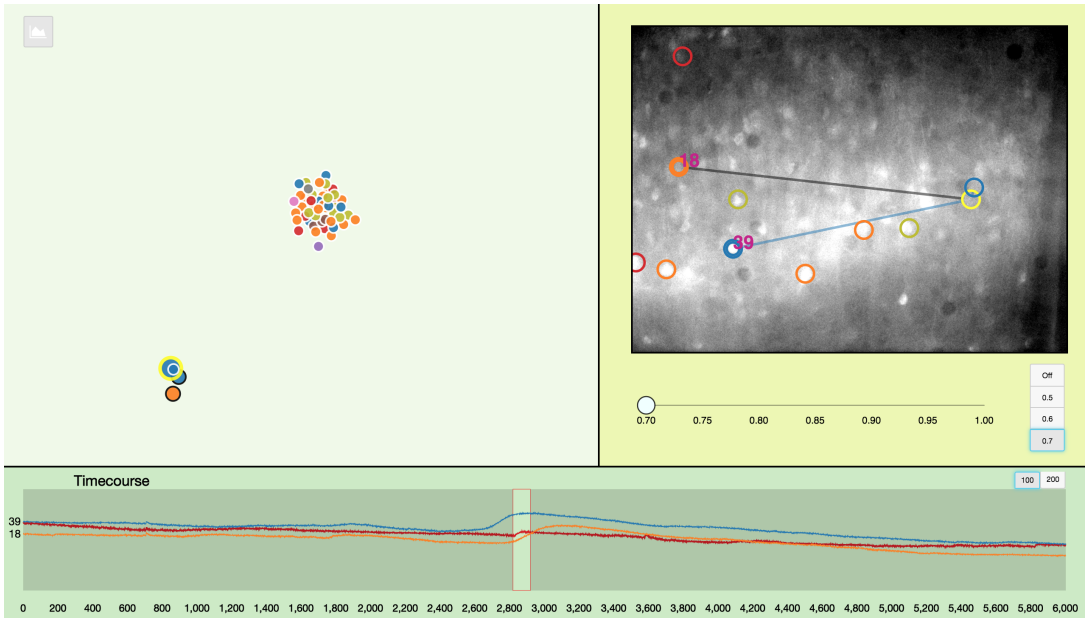

Figure 18: The time courses of the cells number 18 and 39 are set

We decided to use a simple multi-series line chart, instead of a small multiples graph or a horizon graph, since the only section of the time courses that we want to analyze is the one inside the red rectangle, i.e. the current window. It is known that shared-space techniques, like our simple line graph, are usually more efficient to do comparisons over smaller visual windows, while techniques that separate time series in different charts, like small multiples or horizon graphs, are better for comparisons between time series in a larger window [10]. So, for our purpose, a simple line chart resulted in being the best solution. For a better use of the graph, the user should limit to 4-5 fixed lines each time.

5.4.1 Moving the analyzed window

The correlations between cells can vary at any moment of the movie, so, the visualization needed to reflect this behavior. The tool allows the user to navigate through several correlations and communities files representing the state of the network for each analyzed window of the time course. The user just needs to click on any point of the line chart, and the analyzed window will move to that point, dynamically changing the network using the new window files. This gives the user visual hints about the evolution of the network over time, and the possibility to observe a particular window of interest.

5.5 Visualizing the Spikes

The visualization composed of the three different sections resulted in being useful, but still presented a little problem. In fact, what we were really interested in analyzing were the relationships between those cells that fire during a certain time window, i.e. the ones that present a spike in their time courses. To show only those cells using the correlation threshold is

not possible because all the cells that have a slowly decreasing time course during the analyzed window will result as correlated between them as much as the ones that present a spike.

As we can see in Figure 19, at this time, we set a very high correlation threshold, but the visualization still shows many cells that don't present an interesting behavior, like a spike, during the analyzed window.

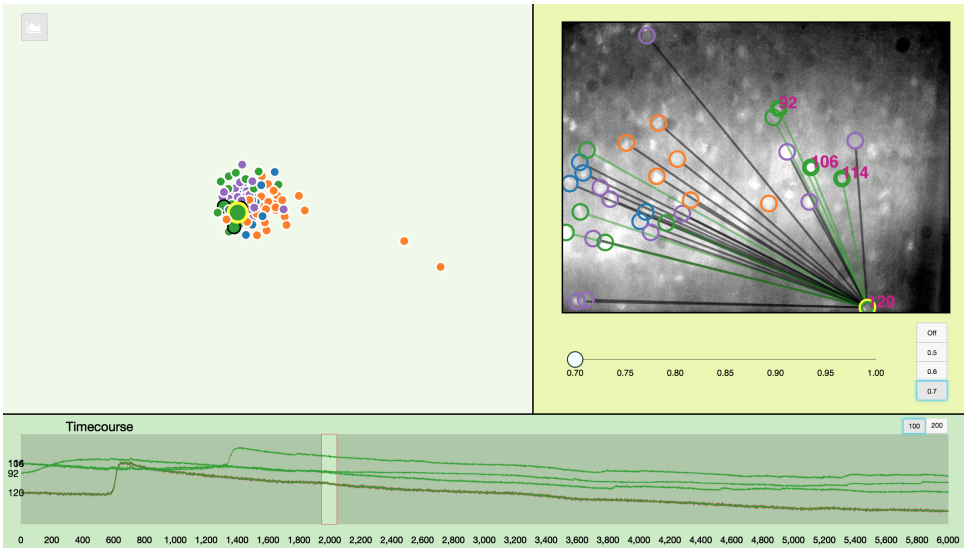

Figure 19: A lot of cells result to be highly correlated, but their behavior is not interesting

To overcome this problem, we developed a special functionality that highlights only those cells that present a very rapid change during the analyzed window. This behavior, most of the time, corresponds to a spike in their time courses. The user can click on the button located at

the top left of the first section to activate this functionality. Once this is activated, all the cells that don't present an enough rapid change during the analyzed window disappear, leaving only the most interesting ones. The user can click again on the spike button to come back to the normal visualization. We can see an example of the network of Figure 19 with this functionality active in Figure 20.

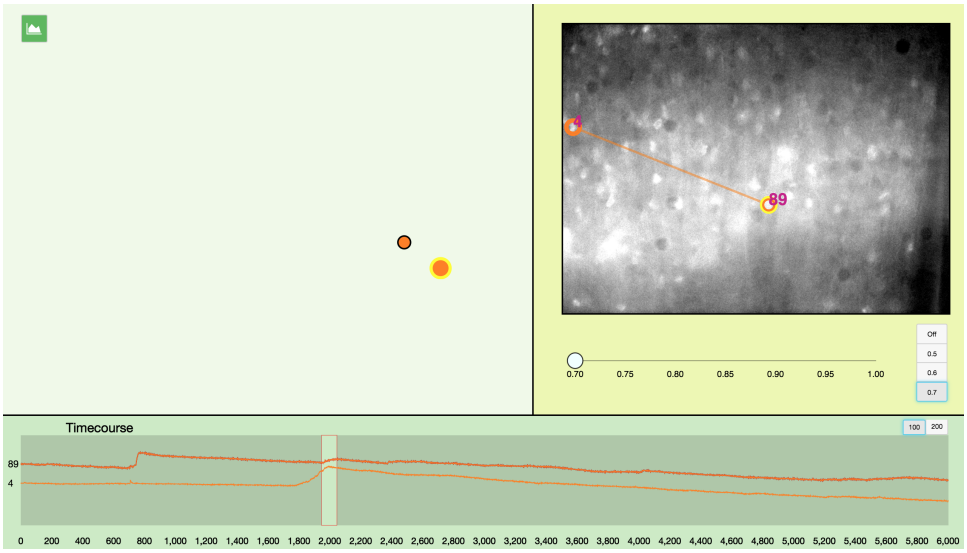

Figure 20: The spike detector is active: only the cells that present a rapid change during the analyzed window are visualized

CHAPTER 6

IMPLEMENTATION

We built the tool as an interactive web application, using a client - server paradigm. This means that the data visualized by the application are stored in a server, and the user can access the web application from any terminal with an Internet connection, using a browser, preferably Chrome. We decided to do this also because of the big amount of data that we wanted to visualize.

The involved technologies are, on the front-end side, HTML, CSS, Javascript and D3.js, the Javascript library for data visualization. The interaction with the server is instead managed through AJAX calls.

6.1 Visualization Elements

The visualization is an interactive web application. It is mainly developed using a famous Javascript library, D3.js [11], that is considered the state of the art for building interactive web applications for data visualization, used for example, by The New York Times to build interactive visualization on its web page. It allows binding elements of the DOM, the Document Object Model of the web page, to the data that you want to visualize. Then, the developer can set these bounded data as parameters for the attributes of the elements, for example for the height of a rectangle.

Since D3.js is a pretty recent library, it is recommended to run the visualization web application on a modern browser, like Chrome, to have better performances.

6.2 Interaction with the Server

When the user opens the web application, the datasets that remain constant with respect to every analyzed window, i.e. the dataset of the coordinates and the dataset of the time courses, are loaded into the user device memory. But the advantage of having a server is that the biggest part of the data, the correlations and the communities of every possible window, are not loaded at the startup. This because they are necessary only when the relative window of the time course is selected by the user to be visualized. So, the only correlations and communities files that are loaded at the startup, are the ones relative to the window starting from the timestamp 0. When a user changes the timestamp, or when he changes the length of the window, the files containing the correlations and the communities relative to the newly analyzed window need to be loaded to the browser, in the user device memory, to be used by the application to update the visualization. These are retrieved using AJAX calls to the Server [12].

AJAX is a web development technique that allows performing asynchronous calls to the server in order to get data. Without AJAX, asynchronous calls wouldn't be possible, and all the data should be loaded at the startup of the application, offering a very bad user experience, since loading all these data on the user machine could take several minutes. Moreover, if all the data are loaded when the web application is open, there is the risk that the memory of the user device has not enough space to save all of them, making the application crash. This can't

happen in the application we built because every time the user loads a new window data, the data relative to the previously analyzed window are removed from memory.

6.3 Flexibility

In our experiments, we used the tool to visualize correlations between cells of the mouse brain, and the relative communities generated by algorithms based on correlations. However, the potential of the tool resides in the fact that the distance measure to determine the network doesn't need to be the correlation, but can be substituted by any other measure. This relationship just needs to be a number between 0 and 1, but it doesn't need to be a correlation. In fact, in our experiments, we discovered that, thanks to the use of this tool, the correlation is not the best factor to build the network, and that it can arise a lot of "false positives".

CHAPTER 7

EXPERIMENTS AND RESULTS

In this section, we illustrate the experiments that we performed using the visualization tool and the relative observations.

7.1 Analyzed Dataset

The datasets used in these experiments are extracted from a 6004 frames movie of a mice brain obtained with the Calcium Imaging technique described before. As explained in the section regarding the extractions of the time courses, each time course represents the average value of the pixels of a cell for every frame, and their length is equal to 6004 frames.

The length of the windows chosen to perform the experiments are 100 frames and 200 frames. This means that, in each window, the visualized correlations and communities are computed using the frame relative to the first timestamp of the window and the successive 99 or 199 frames.

7.2 Validation of the Correlation as a Cells Relationship Measure

The data visualization techniques are not only useful to discover new information or to find some patterns in the data, but can also be used to verify that certain theories or algorithms produce meaningful outputs. Especially in our case, it was very important to have something at the end of the data processing pipeline that allows us to evaluate the correctness of the work made in the previous steps.

We used the tool to verify if the correlation is a good factor in building the network, and if it is computed in the proper way. We found several spots in the data showing that the correlation is not the best factor to generate the network because it produces many "false positives" in terms of edges between nodes.

For example, it is possible to see in Figure 21, that the two the cells 92 and 94 results to be correlated with a correlation higher than 0.7, since the threshold is set to that value and an edge still exists between them. Looking at the time courses of these two cells, we can see that they have nothing in common during the analyzed window, and so, this is a "false positive", since it exists an edge between these two cells in the network, but this edge is totally unmeaningful, since the two time courses have no visible relations.

Playing a bit with the tool it is possible to find many cases where two cells are highly correlated but their two time courses have not so much in common within the analyzed window.

This was a great result in term of the effectiveness of the visualization tool, since it demonstrates that can effectively be used to validate or invalidate the cells relationship measure used to compute the network, in this case, the correlation.

7.3 Calibration of the Window Length

The visualization tool allows also to change the length of the observed window, from 100 to 200. In fact, the algorithm that we use generates different networks depending on the length of the window, since with a different length the correlations change. Thanks to this functionality, we can easily observe that the problem exposed in the previous section, relative to the "false positives" between correlated cells, becomes even worst when we decide to use a larger window.

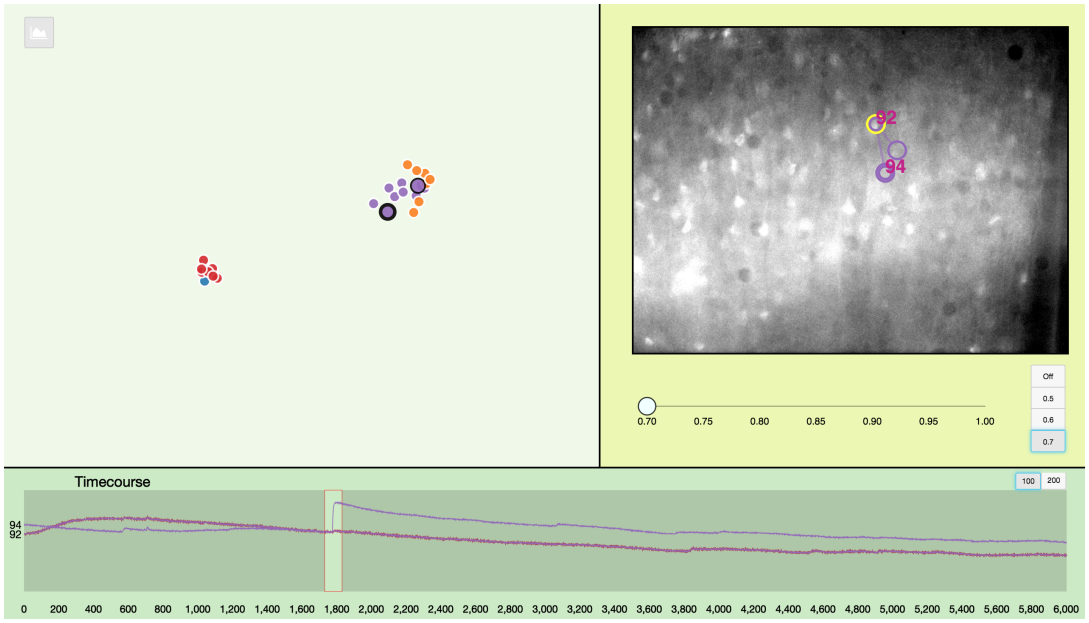

Figure 21: The two cells, with labels 92 and 94, result to have a correlation higher than 0.7, but the sections of their time courses in the considered window have almost nothing in common

For example, if we select a window of length 200 and observe the cells with labels 96 and 104 (Figure 22), we can see that they result to be highly correlated even if their time courses in that window are pretty different. This didn't happen when we observed the same cells in the same window with length 100, as we can see in Figure 23, where the cell labeled 96 disappears since it is not highly correlated with any other cell.

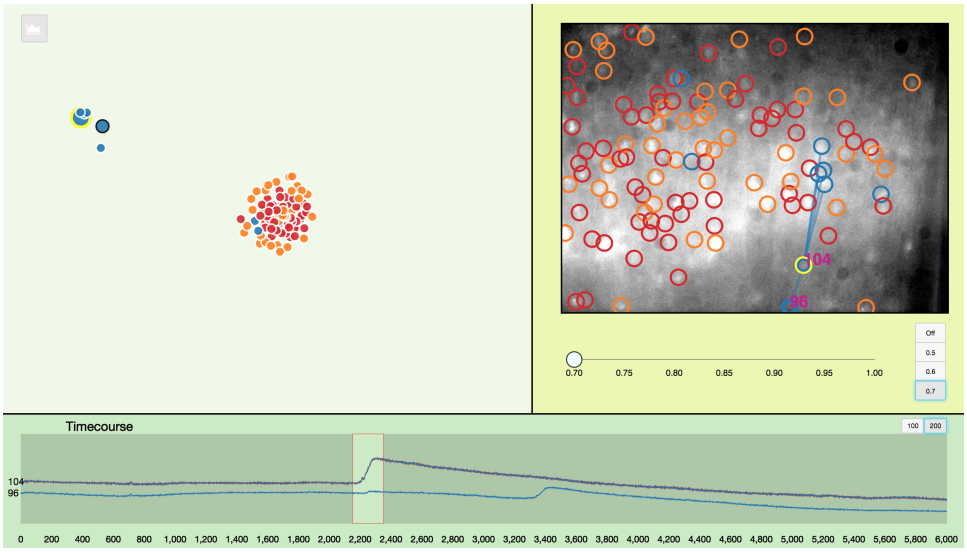

Figure 22: The two cells, with labels 96 and 104, result to have a correlation higher than 0.7, but the sections of their time courses in the considered window have almost nothing in common

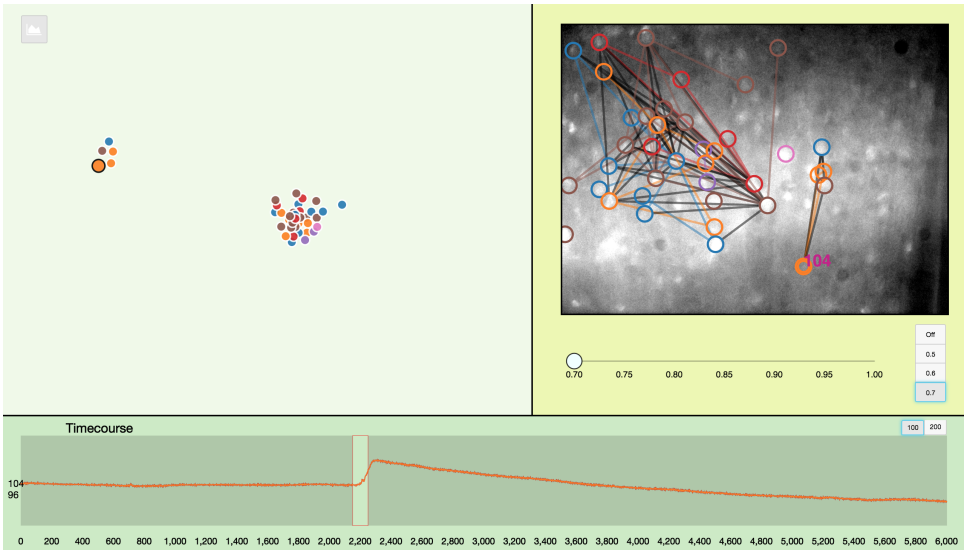

Figure 23: The cell with labels 96, that was highly correlated with the cell 104 using a window of length 200, disappear when the window length is set to 100, since it is not anymore highly correlated with that cell

7.4 Validation of the Generated Communities

The visualization allows the user to validate also the algorithm used to compute the communities, thanks to the black edges feature seen in section 5.2 of this document.

We did this for our dataset, and we discover that the Louvain communities present some undesired behavior in some spots. It is possible to see an example in Figure 24. The correlation threshold is set really high, so, someone could expect that most of the present edges will be between nodes that belong to the same community, because the communities were computed using the correlations. However, in the image (Figure 24) it is possible to see that there are several black edges. This means that a high number of nodes are connected even if they belong to different communities.

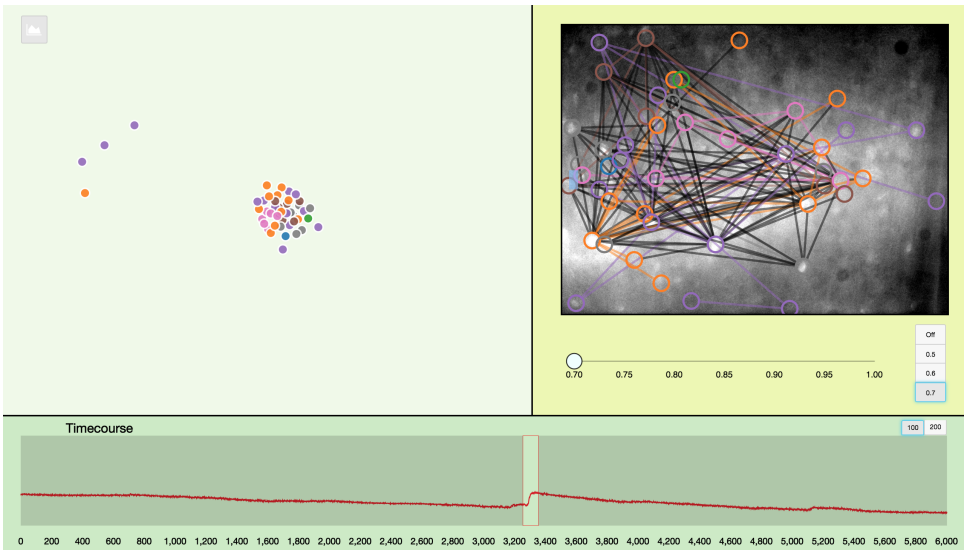

Figure 24: The correlation threshold is set to 0.7, but there are still many edges between cells belonging to different communities

CHAPTER 8

CONCLUSION AND FUTURE WORK

Our experiments demonstrated the advantages of having a visualization tool to explore the data. This can be used to discover new insights or to validate the steps of a data processing pipeline.

In our case, the tool allowed us to discover that the correlation is not a precise factor to compute the network. It also showed that using a too wide window to compute the correlations can bring to less meaningful outputs. Moreover, the spikes highlighting feature resulted in being pretty effective to visualize only the interesting cells in the analyzed window, and something similar could be implemented directly in the previous step of the data processing pipeline, instead of relying only on a correlation cut.

In future, we will get, from the Neuroscience Department of the University of Illinois at Urbana-Champaign, further movies of different sections of the mouse brain. So, it will become interesting to compare the networks computed by our algorithms using different movies. In terms of visualization, this means that the tool could be modified to visualize, at the same time, multiple networks. And, since the number of data will increase, the visualization part of the project will become even more important.

APPENDICES

Appendix A

RUNNING THE APPLICATION

The tool is a web application, so you need to use it in a browser. You don't need to build anything since javascript is compiled on the fly by the browser. In order to run the application, you need to run a local server; for example, if you have python installed on your machine, just write on your terminal 'python -m SimpleHTTPServer 8000'. Then, open a browser, go to the address 'localhost:8000' and navigate to the 'visualization' folder inside the 'Calcium Network Visualization' folder.

If you don't want to run the visualization tool locally on your machine, you can host the visualization tool on your server just uploading the 'Calcium Network Visualization' folder. Then, using a browser, navigate to your server address, and go in the 'visualization' folder.

The visualization will run faster locally (in a case of slow Internet connection) because when you change the window position the new correlations and communities datasets need to be loaded from the server.

The application is already hosted at this address:

`http://compbio.cs.uic.edu/brain/Calcium%20Network%20Visualization/visualization/`

Appendix B

APPLICATION STRUCTURE

The application consists of a series of files (HTML, CSS, js, etc..) contained in a folder called 'Calcium Network Visualization'.

The only files that you need to change, if you want to visualize different data, are the files in the 'data' and in the 'frames' folders.

Inside to the 'data' folder, you will find these files and folders:

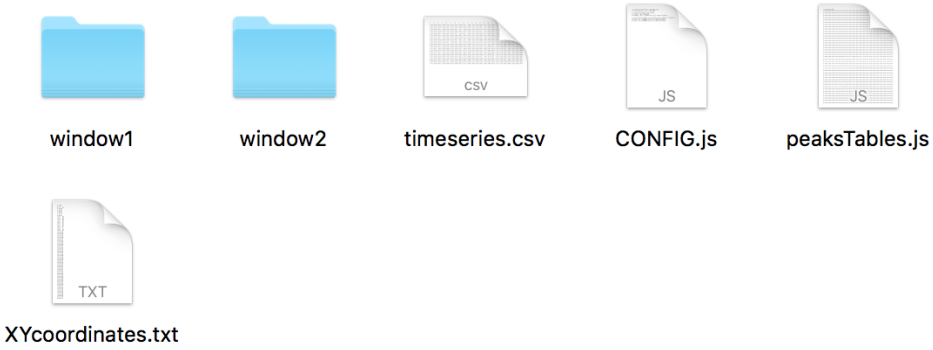

Figure 25: Content of the 'data' folder

In the 'CONFIG.js' file (Figure 26), it is possible to:

- Set the extension of your frames files.

Appendix B (continued)

- Set the window lengths that you used to compute the correlations and communities.
- Set the correlation thresholds relative to the cuts on the network that you used to compute the communities.

```
// Set here the extension of your frames files
var frame_extension = 'png' // NECESSARY

// Set here the length of your windows
var window1 = 100 // NECESSARY
var window2 = 200 // OPTIONAL (set null to remove the button)

// Set here the cut values on your network used to compute the communities
var cut1 = 0.5 // OPTIONAL (set null to remove the button)
var cut2 = 0.6 // OPTIONAL (set null to remove the button)
var cut3 = 0.7 // OPTIONAL (set null to remove the button)
```

Figure 26: The 'CONFIG.js' file

The buttons present in the visualization tool will change accordingly to this file. For example, if in this file you set only two correlation cuts, the application will show only the two buttons relative to these cuts.

The correlations and the communities computed with the windows1 length need to be put in the folder 'windows1', while the ones computed with the window2 length need to be in the 'window2' folder.

Inside to the 'window1' folder, you will find these folders:

- The folder 'correlations' must contain the files relative to the correlations.

Appendix B (continued)

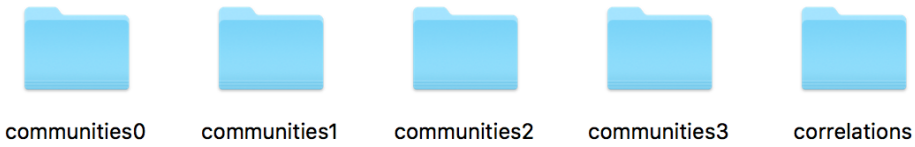

Figure 27: Content of the 'window1' folder

- The folder 'communities0' must contain the files relative to the communities computed on the fully connected network (without cuts).
- The folders 'communities1', 'communities2', 'communities3' can contain the files relative to the communities computed on a network which edges has been cut using the corresponding correlation thresholds set in the 'CONFIG.js' file.
- The folder 'window2' has the exact same structure of the folder 'window1' and can contain the files relative to the communities and correlations computed with the window2, in case you produced them.
- All the files relative to the correlations and the communities need to be named with the number of the starting frame of the window that they represent, as showed in Figure 28.

The file 'timeseries.csv' contains the time series relative to each cell, while the file 'XYcoordinates.txt' contains the coordinates of the centroids of the cells inside the frames.

The formats of all these files are described in section 4.2, and examples of them can be explored at this address:

`http://compbio.cs.uic.edu/brain/Calcium%20Network%20Visualization/data/`

Appendix B (continued)

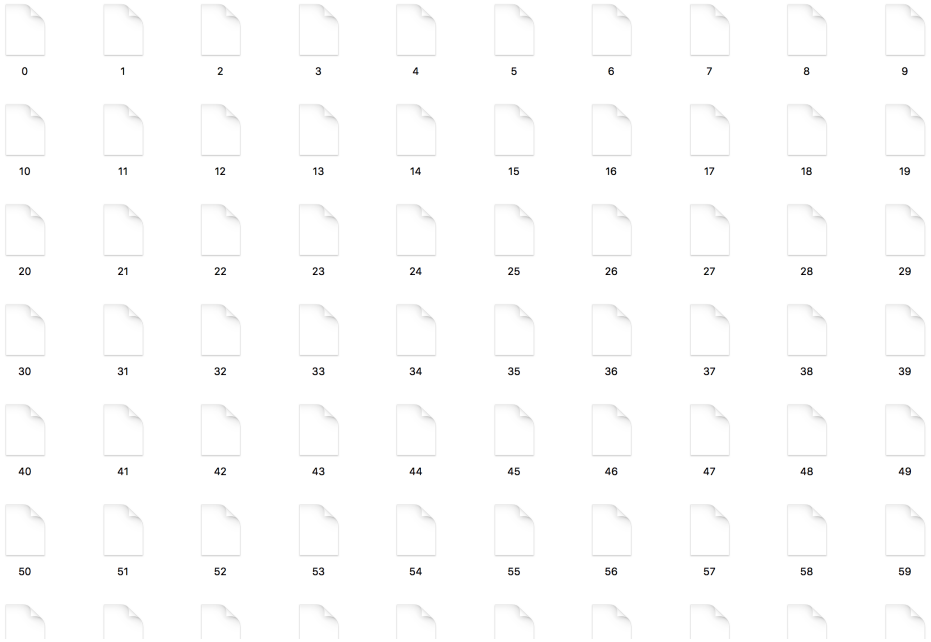

Figure 28: Names of communities and correlations files

Moreover, in the file 'peaksTables.js' it is possible to set the boolean matrices relative to the spikes presents in the time series. In the case that these matrices are not available, set them 'null'.

Finally, the 'frames' folder must contain the frames used to compute all the previous datasets. They must be named in sequence with numbers, starting from 0, and their extension must be the one set in the 'CONFIG.js' file, as shown in Figure 29. An example of the 'frames' folder can be explored at this address:

`http://compbio.cs.uic.edu/brain/Calcium%20Network%20Visualization/frames/`

Appendix B (continued)

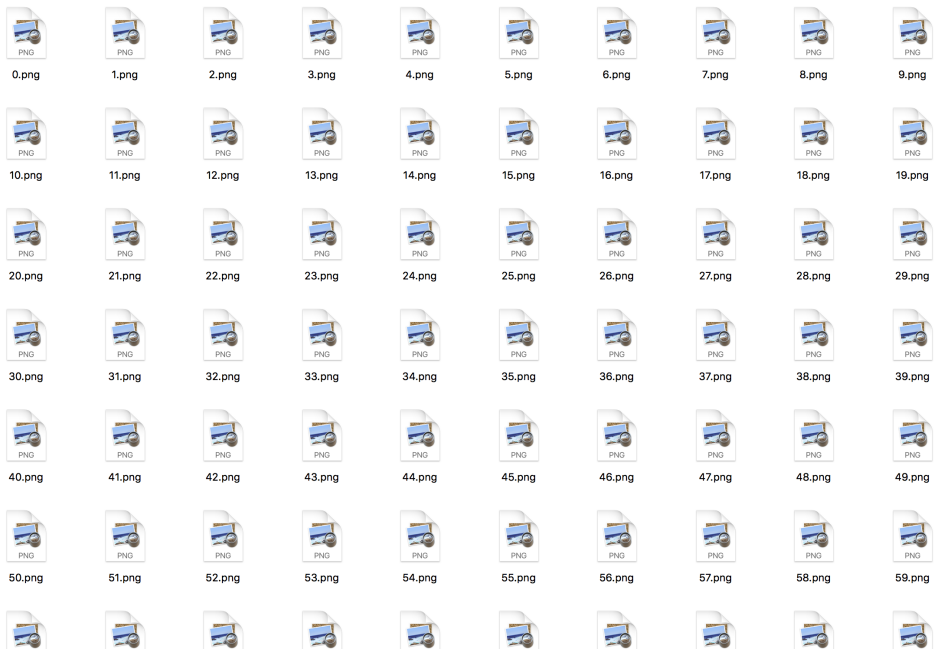

Figure 29: Names of the frames files

CITED LITERATURE

1. Chen, M., Ebert, D., Hagen, H., Laramee, R. S., Van Liere, R., Ma, K.-L., Ribarsky, W., Scheuermann, G., and Silver, D.: Data, information, and knowledge in visualization. Computer Graphics and Applications, IEEE, 29(1):12–19, 2009.

2. Maruyama, R., Maeda, K., Moroda, H., Kato, I., Inoue, M., Miyakawa, H., and Aonishi, T.: Detecting cells using non-negative matrix factorization on calcium imaging data. Neural Networks, 55:11–19, 2014.

3. Hollingsworth, K. P., Bowyer, K. W., and Flynn, P. J.: Image averaging for improved iris recognition. In Advances in Biometrics, pages 1112–1121. Springer, 2009.

4. Yi, J. S., ah Kang, Y., Stasko, J. T., and Jacko, J. A.: Toward a deeper understanding of the role of interaction in information visualization. Visualization and Computer Graphics, IEEE Transactions on, 13(6):1224–1231, 2007.

5. Fayyad, U. M., Wierse, A., and Grinstein, G. G.: Information visualization in data mining and knowledge discovery. Morgan Kaufmann, 2002.

6. Knutson, B., Westdorp, A., Kaiser, E., and Hommer, D.: Fmri visualization of brain activity during a monetary incentive delay task. Neuroimage, 12(1):20–27, 2000.

7. Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment, 2008(10):P10008, 2008.

8. Wilkinson, L. and Friendly, M.: The history of the cluster heat map. The American Statistician, 63(2):179–184, 2009.

9. Rock, I. and Palmer, S.: Gestalt psychology. Sci Am, 263:84–90, 1990.

10. Javed, W., McDonnel, B., and Elmqvist, N.: Graphical perception of multiple time series. Visualization and Computer Graphics, IEEE Transactions on, 16(6):927–934, 2010.

CITED LITERATURE (continued)

11. Bostock, M., Ogievetsky, V., and Heer, J.: D^3 data-driven documents. Visualization and Computer Graphics, IEEE Transactions on, 17(12):2301–2309, 2011.

12. Garrett, J. J. et al.: Ajax: A new approach to web applications. 2005.

VITA

NAME	Manuel Tanzi
EDUCATION	
	Master of Science in Computer Science, University of Illinois at Chicago, USA
	Master Degree in Computer Science and Engineering, Politecnico di Milano, Italy
	Bachelor's Degree in Computer Science and Engineering, Politecnico di Milano, Italy