Electrode drift in extracellular recordings

Rareş Tudorache

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2021

Abstract

Advancements in the electrophysiology field from the past years led to in-depth studies about neural activity from thousands of simultaneously recorded neurons. This analysis consists of assigning every spike to the originating neuron, a process called spike sorting. However, this large volume of data comes with an important limitation during the sorting process, called electrode drift, that numerous spike-sorting techniques have tried to overcome. A pipeline, that reliably solves the probe drift, applicable to all type of sorters has been designed to address this issue.

Only requiring one input (2D spike locations), the main functionalities of the pipeline are split into four big steps: clustering, matching, data cleaning and correction. The correction is based on a novel key measurement, called cumulative displacement, which reliably quantifies the electrode drift. This approach has successfully dealt with non-stationary data, removing the drift by 80% and replacing the final manual curation step. The created pipeline is an extension to every spike sorter and can significantly improve the accuracy of every algorithm. However, it is still not fully automatic and requires parameter adjustment in the matching & cleaning steps, depending on the nature of the given dataset. Moreover, the pipeline is affected by noise and therefore a noise aggregation method could notably improve its applicability.

Acknowledgements

I would like to thank Dr Matthias Henning for his guidance and support over the academic year. Without his help and supervision, none of this work and results would have been possible.

Table of Contents

Chapter 1

Introduction

An accurate analysis of neural movement offers a reliable understanding of brain activity and its complex behaviour. Action potentials, also known as spikes, generate rapid reversals in voltage across the plasma membrane of axons, called spike waveforms. An electrode placed sufficiently close to the spiking neuron will record the spike waveform, a process known as extracellular recording (Hennig et al., 2019).

Improvements in CMOS (Complementary metal-oxide-semiconductor) technology led to advanced high-density silicon probes (HD-MEAs), containing numerous recording sites. Each recording channel captures action potentials from many, nearby neurons simultaneously. This growth enabled the possibility of recording thousands of action potentials concurrently, offering a systematic overview of connections between neurons.

However, this advancement comes at a cost. Each electrode, receiving the activity from many neurons, induced the need of identifying every separately detected spike and assigning it to the corresponding neuron, based on the features of the recorded waveforms. This process is called spike sorting. Nevertheless, many spike sorting algorithms can not deal efficiently and reliably with this new type of recordings. This large volume of recorded data comes with important issues such as overlapping spikes, bursting neurons and electrode drift, that numerous algorithms have tried to resolve so far. This paper focuses on the electrode drift problem and provides a reliable and computationally efficient pipeline that each spike sorting algorithm can apply to overcome the electrode drift barrier.

Electrode shift represents the change in the electrode's position relative to the cell body, caused by the neural tissue movement or by the changes in cells' health throughout the experiment. These non-stationarities are common in long *in vivo* experiments and have a significant effect on the spike waveform shapes, making the sorting process challenging. Overlooking this motion would significantly affect the accuracy of every spike sorter (Rey et al., 2015; Calabrese and Paninski, 2011). A few algorithms, described in detail in the background chapter (2), have previously addressed these non-stationarity issues (Jun et al., 2017; Pachitariu et al., 2016; Chung et al., 2017; Yger et al., 2018). Some models are more powerful to some extent than the approach presented here. However, each of these algorithms contains its specific environment and parameters, with no generalised solution to be easily applied to all existing sorters. As a result, this generality and applicability to every spike sorter are what this project's pipeline aims to achieve. The key advantages of the presented approach are its simplicity and computational efficiency, as it only needs spike locations as input. Consequently, this model is not a fully spike sorting solution. Its purpose is to resolve the non-stationary effect that each recording suffers from after the sorting process. Therefore, this pipeline will make the spike sorting step more accurate by dealing with this non-stationary movement.

For the scope of this project, a recording with an imposed drift has been used. This imposed motion, represented as a 'zig-zag' pattern, is a type of drift, manually applied, intended to replicate the brain movement and the effect that it has on the extracellular recording. The main objective is to straighten that 'zig-zag' pattern and have a fully driftless recording after running the pipeline. The input of the created pipeline is represented by the spike locations of a given recording, computed in 2D using the SpikeInterface framework (Buccino et al., 2020). In summary, the solution starts by clustering spikes frame by frame, followed by matching between closest clusters from consecutive time frames. Next, every pair is used to calculate the average displacement (i.e. the shift in spike positions), which will then be used to interpolate values that corrects the movement.

Overall, the main achievements of this project were:

- Designed a complete pipeline to efficiently correct the drift of every sorted dataset.
- Detailed analysis of the imposed drift recording intended to find optimal parameters for clustering and time window size.
- Built all the steps of the designed pipeline, including clustering method, matching algorithm, data cleaning, drift quantification and a baseline correction.
- Implemented an improved correction, while also completed a section analysis of the dataset.
- Evaluated the pipeline performance (on all types of corrections) on two other datasets containing a different type of drift.

Chapter 2

Background

Action potentials are electrical events produced by a temporary shift (from negative to positive) in the neuron's membrane potential caused by ions suddenly flowing in and out of the neuron. Action potentials play an essential role in cell-to-cell communication (Hodgkin and Huxley, 1952) and measuring them is an important step in understanding brain activity. They are also called ”spikes” and can be monitored using extracellular recording techniques that will be explained in the next section.

2.1 Next Generation Electrophysiology

Electrophysiology is the neuroscience branch that analyses the electrical activity of living neurons (Carter and Shieh, 2015). An extracellular recording is a process of monitoring the electrical activity of neural cells without penetrating the neuron's cell membrane. One of the first extracellular recordings was achieved with one single electrode that could access up to 5 neurons (Lefebvre et al., 2016). Starting from the first conventional devices, which had up to tens of recording channels (Quiroga et al., 2004), the rapid improvement of hardware led to a more current version of chips, called microelectrode arrays (MEAs). They usually provide 30 to 160 metal electrodes on a glass substrate, with a spatial resolution of 100-500µm (Berdondini et al., 2005). For the first time, this new technology introduced the concept of simultaneously recording multi-site neural activity (Ballini et al., 2014). However, even if MEAs were a significant improvement in electrophysiology, a larger number of recording sites was needed to deal with complex circuits at a cellular scale.

These limitations can be overcome by the advances in metal-oxide-semiconductor (CMOS) technology. Improvements in electrode material, thickness and implantation methods have driven growth in the number of simultaneously recorded neurons. High-density microelectrode arrays (HD-MEAs) have significantly increased the number of concurrently active recording electrodes that could monitor thousands of neurons simultaneously. The probe relevant to this project, called the Neuropixel probe, provides $\approx$1000 recording sites on a narrow shank offering on-board amplification, digitisation and multiplexing. This type of probe reduces the wire width substantially, achieving 960 recording sites on a 70 x 20 µm cross-section, weighting $\approx$0.3g. The most important design goals of the Neuropixel probe were minimising brain tissue damage (small cross-sectional area) while isolating individual neurons with dense and extensive recording sites (Jun et al., 2017; Steinmetz et al., 2018). This impressive advancement enabled for the first time the examination of neural interactions in large circuits (Hennig et al., 2019; Berdondini et al., 2005). Moreover, because action potentials can be concurrently detected on multiple electrodes, each neuron's spatial information becomes an important feature that can now be registered (Diggelmann et al., 2018).

However, this rapid growth comes with important computation challenges, such as computer runtime and the curse of dimensionality. For instance, modelling pair-wise distance interaction between 100 neurons would require approximately 10,000 parameters estimated from a few hundred trials (Stevenson and Kording, 2011). Moreover, different concurrently active neurons might have overlapping spikes, which can be difficult to resolve simply because there is too much activity. Furthermore, it is expected that the number of simultaneously recorded neurons will continue to double every seven years in the future. This rapid growth over time requires an accurate analysis of interactions between neurons and accurate isolation of each neuron's activity to understand brain activity (Stevenson and Kording, 2011). Therefore, estimating the number of neurons and assigning each detected action potential to one of these neurons is an essential task when dealing with a large volume of recorded data (Diggelmann et al., 2018), (Hurwitz et al., 2019). This challenge becomes a complex assignment problem called spike sorting, described in more detail in the next section.

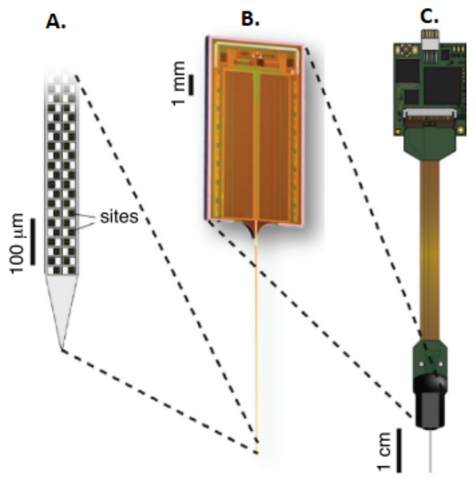


Figure 2.1: The Neuropixel probe (C) containing a CMOS part needed for amplification, digitisation and multiplexing (B). Detailed image of the shank producing amplified and digitised data (A) (Steinmetz et al., 2018).

Additionally, due to brain movements, the spiking unit can move out of the probe's detection range. High-density probes (Figure 2.1) have the potential to solve this problem by offering extended spatial coverage to track drifting units (Jun et al., 2017).

2.2 Spike detection and sorting

A spike sorting pipeline consists of two important steps: detection of action potentials and a method of assigning these action potentials to the corresponding neurons (Hennig et al., 2019; Lewicki, 1998). In the context of HD-MEAs, two main approaches successfully overcame the challenges presented in the previous section. One method uses an amplitude threshold followed by feature extraction combined with clustering, whereas the second method involves building spike templates and assigning them to the most likely template (Rey et al., 2015). We will briefly describe each method in this section.

- *Template detection & clustering*

  This template-based algorithm learns templates of spike shapes (Pachitariu et al., 2016) by applying dimensionality reduction combined with clustering. Each newly detected spike is then classified by matching it with the most likely template (Franke et al., 2012), based on the spatio-temporal event footprints (Figure 3.1). This method is a very suitable approach for recordings with high firing rates, being relatively simple with a low computation cost (Rey et al., 2015). However, recordings can last many hours and therefore this method requires spike shapes and also the location of the template to be constant over time (Muthmann et al., 2015).

- *Threshold detection*

  An alternative approach to the one presented earlier is threshold detection. The most distinguishable feature of the shape of a spiking neuron is the spike's height, also called amplitude. This property is then used to detect the spiking activity by comparing it with a voltage threshold trigger (Lewicki, 1998) (Figure 3.1). The value of the threshold can be set manually, but the alternating noise levels can make the isolation of single-neuron activity a challenging task. If the value is too small, it will lead to false-positive events, whereas a too-large value will not cover low-amplitude spikes. As a result, the threshold is dynamically set for more accurate detection using an estimate based on the percentile of the filtered signal (Rey et al., 2015). Because this threshold-based method requires minimal hardware and software, it became the most common method in spike detection. However, single-neuron activity isolation is not always achieved with this method (Lewicki, 1998), especially in dense arrays, because of overlapping action potentials.

  Following detection, a feature extraction method, such as principal component analysis (PCA), reduces the dimensionality of spike features, such as spike waveform and their estimated locations (Hilgen et al., 2017). Then, a clustering method is used to group spikes and assign them to a specific neuron (Muthmann et al., 2015). Because the number of observed neurons is unknown, non-parametric clustering algorithms are used to overcome this issue. Moreover, during the recording, the electrodes might drift. Drift is the tissue movement bringing sites closer or further away from the spiking source. This fluctuation occurs concurrently across adjacent sites, changing each neuron's relative position to the electrodes (Hennig et al., 2019). This is an important issue that occurs in each spike sorting pipeline, and this paper approaches new ways of solving this movement.

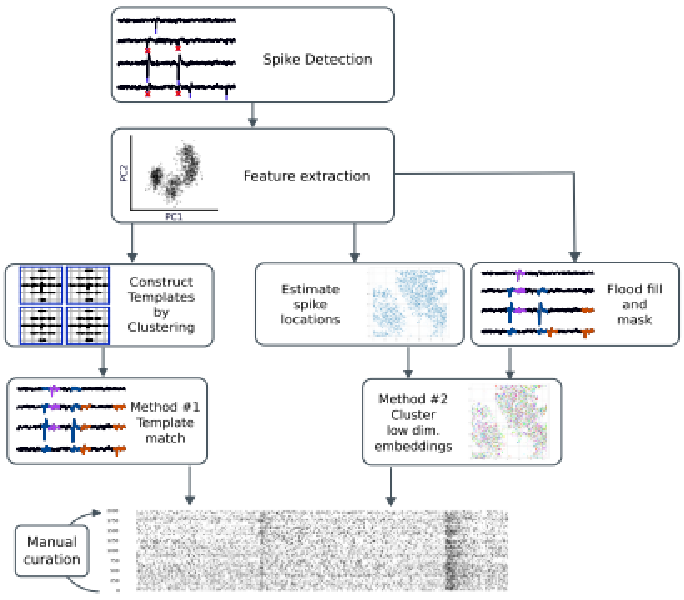


Figure 2.2: Spike sorting pipelines for high-density microelectrode arrays. After detection, either a mask is built to restrict clustering to channels with a detectable signal, or spike locations are estimated and clustered together with waveform features, or neuron templates are constructed based on spatio-temporal event features. The final step usually consists of manual curation, which implies human intervention to correct the sorter (Hennig et al., 2019).

2.3 Challenges in spike sorting

Regardless of these rapid improvements in the spike sorting methods, there is no algorithm to be fully automatic yet. Because of various challenges that occur in a spike sorting pipeline, manual supervision is needed to get improved results (Steinmetz et al., 2018). This section will identify the major challenges, that come with this large volume of data, and analyse how these problems affect the accuracy of the methods described above.

2.3.1 Overlapping spikes

Overlapping spikes are one of the most challenging issues when dealing with single-unit activity (i.e. the activity of each separate neuron). If two nearby neurons fire at sufficiently separated moments in time, their spikes will be correctly identified and classified. However, if two spikes fire simultaneously or with a small delay, they might overlap in time (Rey et al., 2015). Being a very important challenge in spike sorting, various methods have been proposed over the years (Lewicki, 1994; Hulata et al., 2002;

Prentice et al., 2011; Pillow et al., 2013), but without any optimal solution to deal with it.

One simple approach in overcoming this issue, proposed by Lewicki (1998), is to subtract a spike from the waveform once it is classified. However, this method has its drawbacks since it requires a spike shape model, which, if the template is not accurate, will add even more noise to the waveform. A more recent and novel approach was made by Yger et al. (2018). The SpyKING CIRCUS sorter successfully deals with overlapping spikes by a parallelised density-based clustering combined with a template matching approach. This clustering method only needs the centroid of each cluster and not their exact borders.

2.3.2 Bursting neurons

An exciting thing to note about spikes shapes is that they are not stationary, and they change in amplitude and shape with time (Fee et al., 1996). These fast sequences of action potentials, called bursts, may cause different spikes of the same neuron to be assigned to different clusters. Therefore, clusters containing bursting spikes would have to be merged. A good indication of this would be the timing between the sequence of spikes (Rey et al., 2015).

2.3.3 Electrode drift

During the recording, the electrode can drift slowly to a new position because of the neural tissue movement. This motion results in a gradual change in spike amplitudes of the same unit in neighbouring sites (Lewicki, 1998). As a result, clusters might split, appear, or disappear over time, which significantly affects every spike sorter's accuracy (Rey et al., 2015). This is a fundamental issue in spike sorting, and many approaches have been tried over the years, depending on the sorting algorithm used. This work is trying to overcome the drift problem by exploiting the idea of spike localisation.

These previously mentioned challenges lead to a final step in any spike sorting pipeline, called manual curation. This step is a complicated task requiring much human labour with high error rates, greatly limiting HD probes' utility (Jun et al., 2017). In the following sections, previous attempts to create a fully automatic pipeline and solve this movement will be analysed in more detail, followed by a description of this paper's drift correction approach.

2.4 Previous work

The slow movement of the recording electrode, together with background noise, make the data non-stationary, as described by Lewicki (1998). This inevitable motion results in gradual changes in spike shapes over time. As the usage of electrode recordings increased, this non-stationary nature of the data proved to be one of the main barriers to a successful spike sorting algorithm. At that time, this task required much human intervention, and most of the sorting algorithms have not addressed that difficulty yet, assuming data to be fully stationary (Bar-Hillel et al., 2006). Moreover, the overwhelming amount of data coming from larger electrode arrays will make the human intervention a more challenging task, increasing the error rate (Berényi et al., 2014; Chung et al., 2017).

Over the years, there have been several attempts in dealing with lack of stationarity, which had the purpose of automating the spike sorting process. The main approach was to find an optimal method that would combine clusters efficiently, given the gradual spike changes. One of the first efforts was made by Fee et al. (1996), which merges clusters based on the inter-spike interval (ISI) information and density of points on the borders between clusters. Two years later, Snider and Bonds (1998) proved that inter-spike interval information is unreliable criteria for correlating neurons. Next, Shoham et al. (2003) proposed to model clusters using a t-distribution that can deal with changing clusters. However, a significant advancement in coping with drift was brought by Emondi et al. (2004), who suggested a method in which data is divided into small time frames. Each time frame is clustered manually, which will then be automatically matched with clusters from consecutive time frames. This approach was a step forward in spike sorting because, for instance, clusters of similar spike shapes from distant time frames will not necessarily be matched, as was the case with the previous methods (Bar-Hillel et al., 2006).

2.4.1 Bayesian Clustering

None of the previously mentioned approaches has successfully automated the spike sorting process. One of the first approaches in automated pipelines was attempted by Bar-Hillel et al. (2006). At that time, this task still required much human intervention since most of the algorithms assumed data to be entirely stationary. This method is exploiting the clustering process by trying to automate it in a Bayesian framework.

In the first instance, the data is divided into short time frames of fixed duration where for each two consecutive time intervals, stationarity is assumed. Next, there is a search stage in which a set of possible mixture-of-Gaussians solutions is found for each time frame separately by looking for a hidden mixture model that explains the Gaussian components in both time frames. Following the heuristic search, transitions probabilities of consecutive mixtures are computed, which models the data set as a chain of mixture models. The maximum-a-posteriori (MAP) solution of the chain model will then be used to do the final clustering and the matching between clusters in consecutive time frames (Bar-Hillel et al., 2006).

A key thing to note in this algorithm is the computation of the transition probabilities $P(z_t \mid z_t - 1)$, for each time frame t. They are based on the stationarity assumption of the Gaussian components' distribution in the consecutive time frames. However, these assumptions of standard clustering might fail in sparse clusters that represent neurons with low firing rates. As a solution to this, the entire dataset is considered in a single time window and resampled to reduce dense areas' weight (Bar-Hillel et al., 2006). Overcoming this issue will improve the algorithm efficiency bringing it very close to the human expert performance.

By looking at the real-data evaluation, it seems that the model reaches the human performance level. This agreement with human experts was the main metric of evaluation. These human sorters rely on visual representations in two PCA dimensions and therefore, this is also why the algorithm works with two-dimensional spike representation. The chain of Gaussian mixtures model and the insertion of past and future information into the clustering decision at each time window make the method closely follow human intuition. These steps might limit the algorithm's feasibility since it might be difficult to apply all these steps to a more extensive data set efficiently (Bar-Hillel et al., 2006).

In conclusion, the algorithm has successfully automated the sorting mechanism while coping with non-stationary data, but it might be limited in scalability, due to the significant increase in the number of sites.

2.4.2 Kalman filter mixture model

Following the Bayesian clustering method presented earlier, we will also explore the Kalman filter mixture model, designed by Calabrese and Paninski (2011). This system does not represent a spike sorting pipeline; instead, it focuses on a simple and computationally efficient method for the subtask of tracking non-stationary data.

As the previous method, this algorithm also uses the classic mixture-of-Gaussians (MoG) model for spike sorting, being one of the most common probabilistic model underlying the clustering process. Each recorded voltage waveform snippet is a sample from a mixture distribution. Associated with MoG, there is the expectation-maximisation (EM) algorithm for parameter interference in the model. Because the mixture components' identities are 'hidden', standard likelihood optimisation methods may be difficult, and therefore, the EM algorithm is used to estimate the model parameters (Calabrese and Paninski, 2011).

Now, considering the situation where the mean voltage waveforms are non-stationary. Instead of taking each cluster's mean as constant in time, this parameter is now modelled with a random drift. Thus, the given sequence of clusters will correspond to a Kalman filter mixture model. The algorithm consists of the standard MoG E-step used to obtain the probabilistic allocations of each cluster assignment to each sample, given the model parameters from the previous iteration. Next, the M-step updates the mixture parameters followed by a Klaman forward-backward sweep (Calabrese and Paninski, 2011).

In conclusion, this basic model can be perceived as an extension of the usual mixture-of-Gaussian model where the cluster means vary with time. This approach is a very similar one to (Bar-Hillel et al., 2006), which used a more general model allowing both the covariance and the mean of the distribution of spike shapes to change with time. This Kalman filter model proved to be effective and robust when applied to simulated and real data despite its simplicity. Moreover, refractory effects can be incorporated into the model via closely related hidden Markov model techniques, which will make the algorithm even more effective.

2.4.3 MountainSort

Another ambitious spike sorter that aims for complete automation is called MountainSort. This sorter's primary goal is to be applied to different brain regions with a minimum number of user-defined parameters or modelling assumptions (Chung et al., 2017).

MountainSort uses a novel, non-parametric, density-based clustering algorithm called ISO-SPLIT. It operates in a relatively low dimensional feature space corresponding to the first ten principal components, and it makes two main assumptions about cluster distributions. First, each cluster that arises from a density function has only a single region of highest density when projected onto any line. This assumption is called unimodality. Second, in a lower-density neighbourhood, any two distinct clusters may be separated by a hyperplane. As a result, sorting is independently performed on electrode neighbourhoods, and groups are then consolidated across all channels (Chung et al., 2017).

Comparing with other methods, such as KiloSort or SpyKING CIRCUS (Pachitariu et al., 2016; Yger et al., 2018), MountianSort relies on a single, standard set of parameters that eliminates the inter-operator variability. Moreover, MountainSort runs fast on a large dataset due to clustering performed on local electrode neighbourhoods. This way, clustering time scales linearly with the number of channels (Chung et al., 2017).

The unimodality assumption fails when dealing with differences in spike shapes within a sequence as in bursting units or with electrode drift. In the case of bursting neurons, MountainSort will cluster these events in two or more separate groups, labelling the first cluster as the "bursting parent." Then, using this label and timing information, clusters are then merged accordingly. This method occurs in a post-processing step in which bursting pairs are automatically detected. However, this method does not fully solve the drift problem. Even if MountainSort naturally handles some electrode drift level, this challenge remains unresolved. In contrast, current solutions to this challenge, such as KiloSort or SpyKING CIRCUS, rely on model-based frameworks (Chung et al., 2017).

As a result, MountainSort is an automated sorting algorithm that will ease this task substantially. Moreover, each electrode channel neighbourhood is allocated to a logical core, making the algorithm scalable for large electrode arrays. However, electrode drift is still a challenge, and we will explore further how other methods approached it.

2.4.4 JRClust

JRClust is a more recent, scalable and automated solution in spike sorting that successfully handles all of the major challenges, including electrode drift. The new generation of electrophysiology probes has made this process possible, and JRClust exploits the new high-density probes to perform reliably under noise and probe drift.

Jun et al. (2017) introduces a novel feature extraction method based on channel-covariance that can tolerate spike timing variations. Additionally, by using HDMEAs (concurring spikes appearing on closely-spaced sites), this feature extraction method also includes estimated spike locations in the feature set. Spike positions are estimated by averaging the site coordinated weighted by their spike amplitudes (Nadasdy et al., 1998). This new feature will improve the sorter's accuracy, allow faster clustering, and provide means for correcting the drift.

Regarding the clustering stage, conventional clustering algorithms based on fitting a mixture of Gaussian distributions whose parameters are determined by the EM algorithm (Bar-Hillel et al., 2006; Calabrese and Paninski, 2011) are no longer appropriate. In the context of HDMEAs, these clustering methods do not scale well on the increasing number of sites, and the Gaussian distribution assumption does not hold properly for the bursting neurons and electrode drift challenges. As a result, a clustering method based on a fast search on density-peaks (DPCLUS) is used as a solution. It can automatically discover the number of clusters by identifying the local density peaks.

The JRClust approach on drift correction is based on the spike location feature. High-density probes offer additional sites along with their depth, allowing experts to track these moving units. Each event's centroid position is computed from the average of site positions, each site being weighted by its squared spike amplitude (Nadasdy et al., 1998). Next, by comparing spatial relationships between adjacent time bins, the global drift is determined. Each spike position is then corrected by subtracting the probe positions as computed at each time bin to results in constant activity band positions. Probe position is estimated by computing the cross-correlation of the spatial profile to determine the optimal vertical spatial shift between neighbouring time bins. This step is performed before the clustering stage. Moreover, the spiking position allows a more efficient and accurate clustering by restricting its neighbours' search range (Jun et al., 2017).

Regarding run-time, the JRClust pipeline achieves high speed and scalable performance without reducing the analysis's quality. The processing time scales linearly to the recording size for the preprocessing stage, while the rest of the process scales linearly to the number of spikes, without any bottleneck stage. This is a great performance comparing to other methods' computation time that increases exponentially with the number of electrodes (Rossant et al., 2016). The total processing run-time speed of the pipeline is 10x faster than the real-time recording speed from a 120-channel probe using a single workstation with one GPU. As a result, JRClust is a scalable and automated solution in spike sorting that can reliably deal with electrode drift. This advancement is the start of future innovations in algorithmic performance and hardware optimisation (Jun et al., 2017).

To summarise, we can conclude that most of the spike sorting algorithms approach in dealing with high-density probes and electrode drift is to make the clustering process more efficient. To make it scalable for thousands of electrodes, spike sorters parallelised the clustering step by dividing the snippets into groups, while computing spike locations would handle the drift challenge. As a result, the non-stationary data issue is still largely unresolved, and the problem has not been quantified properly so far. From the previous research, we can deduce that spike localisation is reasonably the best approach in overcoming drift. In the next section, we will describe this paper's approach to drift solving, which is also spike positions based.

2.5 Data used in this project

As we can remark from the previous section, every spike sorter suffers from electrode drift. Any sorting algorithm that relies on waveform features, spike locations, or template matching will be affected by this brain movement. Therefore, this task still requires a computationally efficient solution that should be effortlessly scalable among all sorters.

In this project's approach to solving electrode drift, a recording with imposed drift motion has been used. This type of drift is an imposed motion used to replicate the brain movement and manifests itself by co-modulating the same unit's spike amplitudes in neighbouring sites.

2.5.1 Spike detection and localisation

The data used in the created pipeline consists of spike locations, computed in 2D. The detection pipeline, created by Muthmann et al. (2015), builds a spatio-temporal profile of events without the isolation of single units. Starting from the raw voltage data, the online variant of the threshold method designed by Muthmann et al. (2015) was used for detection. For each channel in each time frame t, the algorithm reads the raw signal x_t, together with an online estimate of a baseline b and a variability estimate around the baseline v.

To determine whether an event has happened at time frame t, using the variability estimate v from the previous time frame, the algorithm dynamically updates the baseline value b in step 1, which is then used to compute the threshold for detection in step 2. The variability estimate v is also updated depending on the signal's amplitude variation from the current estimation. As it can be noticed in Figure 2.3, following baseline and threshold steps, there comes the filtering step where only the events exceeding the threshold are accepted as spikes, which are then adjusted for their duration and repolarisation (Muthmann et al., 2015). These two features can be used as additional acceptance criteria.

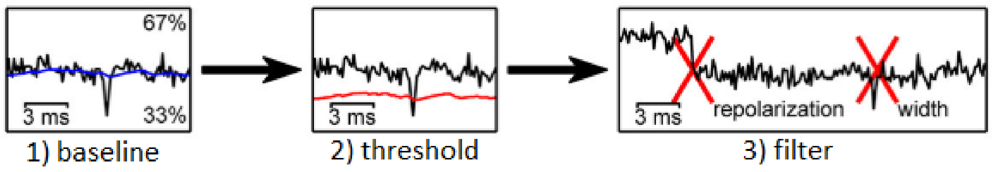


Figure 2.3: The detection algorithm split into three stages: 1) Computation of baseline variable b for every time frame t. 2) Detection of event that exceeds the threshold value computed based on the baseline value b. 3) Adjustment stage, events are filtered based on shape measures to reduce the number of false positive (Muthmann et al., 2015)

.

In addition, to prevent the same neuron's detection multiple times, concurrent events in the adjacent electrodes are assigned to the electrode with the strongest signal (Muthmann et al., 2015). Finally, the source location of the event from the raw signals is estimated. Starting from the assumption of Pettersen and Einevoll (2008), that the action potential varies between $1/r^2$ and $1/r$, given distance r, Muthmann et al. (2015) observed that current spike locations could be estimated using the amplitudes at neighbouring electrodes. Final spike positions are computed as the centre of mass of spike amplitudes from neighbouring channels. Following this detection process, for each recorded event, channel id, spike location and timestamp of the event are stored in a 2D array.

In the implementation of this project, the event locations (x&y coordinates) feature has been used and computed by the method described above. By plotting the 2D spike positions from the original recording with the imposed motion, a 'zig-zag' pattern, shown in Figure 2.4 can be observed. This pattern comes from the manually imposed movement of the probe that gave the recording this specific aspect. The purpose of this project is to correct this imposed movement and to straighten that pattern.

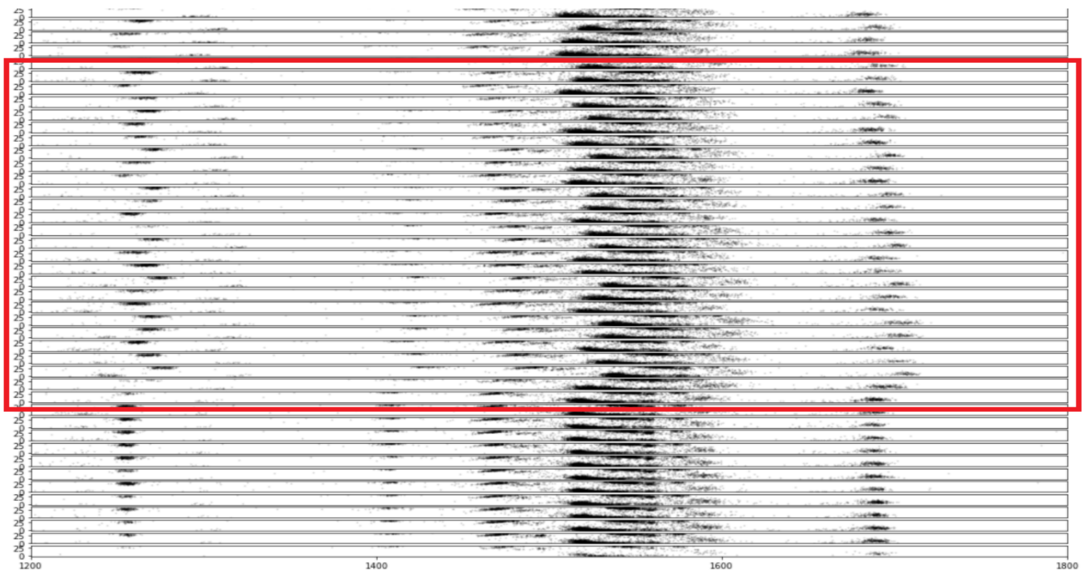


Figure 2.4: Raw data split by time. On the *y-axis*, based on the step value, time is split into equal segments, also called time-frames. Each segment represents a time frame, and the values of the long dimension of spikes are displayed on the *x-axis*. Imposed drift motion is visible in the red rectangle.

2.5.2 Unexplored challenges

There are still unexplored challenges in the electrode drift subproblem. For instance, different parts of the probe might move in different directions (non-rigid drift). Moreover, there might be other possible types of drift that arise when taking positions into account. Therefore, a reliable drift correction algorithm might need to also deal with both horizontal and vertical drift. All of these variations will be explored and analysed by this paper's approach in correcting the drift.

Chapter 3

Methodology

The main goal of this pipeline is to correct the non-stationary aspect of the data that comes with HD-MEAs recordings. In Figure 3.1, an overview of the pipeline designed to resolve this movement is presented step by step. Each part is computationally efficient, easy to implement and to adapt. The only required input is taken by the clustering function and is represented by the spike locations computed in 2D. The next section will dive deeper and will explain in greater detail every part of the process, but also the motivation behind it.

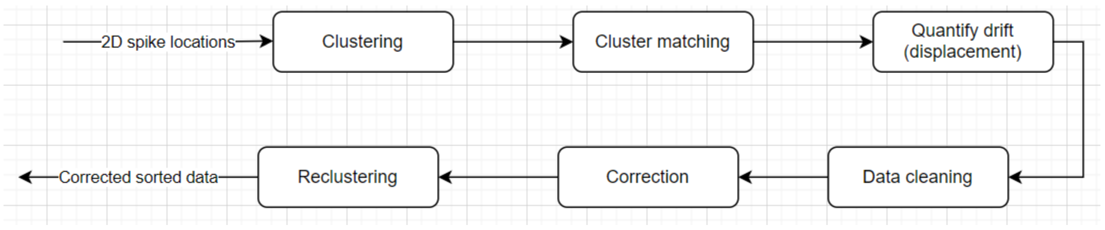


Figure 3.1: Overview of the correction pipeline step by step. 2D spike locations are taken as input, returning the corrected and sorted spikes.

3.1 Clustering

Having all the spike locations detected and computed by the method described in subsection 2.5.1, this project's pipeline starts by clustering the event locations, a process called spike sorting. The clustering phase aims to group spikes accordingly that would characterise a single neuron's activity. To achieve this sorting, the MeanShift clustering algorithm has been used (Comaniciu and Meer, 2002). MeanShift, a so-called mode-seeking algorithm (Cheng, 1995), is a well-known technique that locates a density function maxima, which was initially presented by Fukunaga and Hostetler (1975).

MeanShift approach is an iterative method that *shifts the mean* of a cluster into a higher-density region until convergence. Starting from an initial estimate x, the kernel function of the form $K(x_i - x)$ determines the weight of nearby points for re-estimation of the mean. The only parameter required by the kernel function is the bandwidth, noted as *h*. This parameter determines the size of the cluster and the kernel as well. Each cluster centre is moved to a new position at every iteration until convergence by a mean shift-vector. The kernel is shifted to the new position, which is determined by the centroid of the mean of the cluster's points. The choice of the kernel will determine the method of calculating the mean. Cluster centres within the bandwidth *h* will be merged, and the highest density cluster will be picked. To find the local maxima, MeanShift uses a variation of the gradient ascent technique, knows as *multiple restart gradient descent* (Szeliski, 2010), that deals better with higher dimensions. The algorithm stops when, for the given iteration, the cluster centres do not shift anymore.

In the context of spike sorting, MeanShift clustering proves to be a very efficient approach due to its specific properties that are very suitable in this area. Firstly, unlike K-means for example, MeanShift does not require the number of clusters to be established in advance, comparing to other clustering methods. This feature is an essential characteristic because, in spike sorting, it is challenging to determine the number of neurons before the sorting process. Another crucial aspect is that MeanShift only requires one parameter, the bandwidth, that can be estimated since it represents the cluster's width. Since every spike sorter's goal is to be fully automatic, requiring one single parameter makes MeanShift an excellent choice for the clustering step.

The Neuropixel probe data analysis required splitting it into multiple time frames and applying the MeanShift algorithm on each time interval separately. After rigorous testing, a time step of 20 seconds has been set, which offers a reliable tracking of clusters among all time intervals. A larger time interval would make the imposed drift pattern difficult to observe and track among the time frames. Furthermore, the bandwidth parameter value has been set to 6.5, which is the optimal value of the cluster's radius. A smaller value would generate a much larger number of clusters and also fewer spikes available for clustering, which would make the sorting hard to achieve. Respectively, a higher value would output big clusters that would make the assignment of spikes inaccurate.

- *Baseline clustering*

  As input to the MeanShift algorithm, there is the 2D spike locations array (described in Section 2.5) covering the whole length of the probe, which is divided into multiple time intervals. As a result, the probe data is split into equal time frames, as shown in Figure 3.2. At each iteration, spikes within each time step are clustered using the parameters configuration explained above. Each cluster's position from each frame is stored in an array that will later be used for tracking the movement of the locations. As expected, each time frame contains a similar number of clusters.

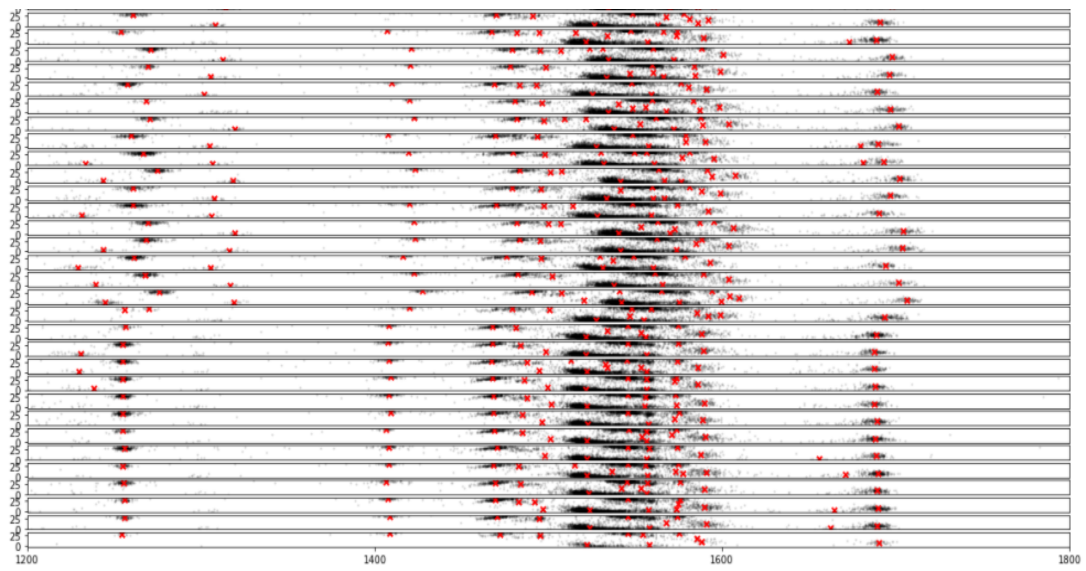


Figure 3.2: Clustering of the raw data frame by frame. The probe's long axis coordinates of spikes are displayed on the x-axis, whereas time (s) is represented on the y column for each time interval. Each cluster within each frame is marked with a red '*x*'.

- *Improved clustering*

  To achieve a more reliable analysis, an extra step is taken during the clustering phase. Some clusters were not a reliable representation of a neuron's activity because of the background noise challenge. Therefore, clusters that contain fewer spikes than a set threshold will be disregarded. As a result, there is a drop of approximately 20% in the number of clusters for each time interval. This way, the remaining groups will represent more accurately the activity of each neuron.

  By comparing Figure 3.2 with Figure 3.3, it can be observed that clusters that do not contain a reliable number of spikes are disregarded. This filtering will improve the accuracy of the matching algorithm that will not consider unreliable groups anymore.

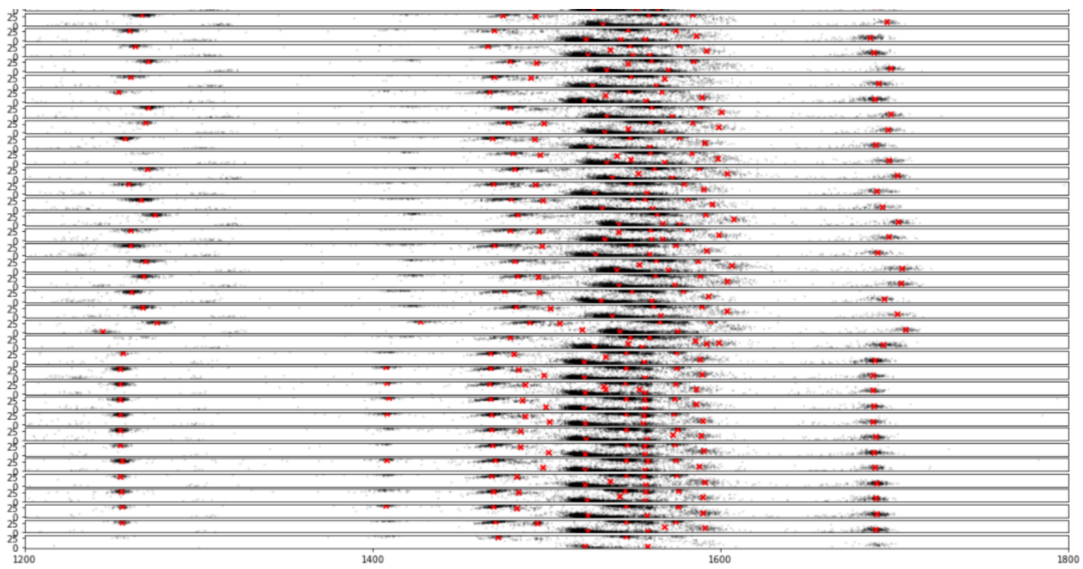


Figure 3.3: Improved clustering of the raw data frame by frame. Spike positions are represented in x&y coordinates and every cluster is marked with a red '*x*'. The probe's long axis coordinates of spikes are displayed on the x-axis, whereas time (s) is represented on the y column for each time interval.

At each time step, the clustering algorithm converges, and all spikes within a cluster indicate each neuron's electrical activity for that specific time interval. Next, each group from each time frame is passed to a matching algorithm to track the neural activity for the whole recording duration. This matching step generates pairs of matched clusters used to calculate their movement.

3.2 Matching algorithm

A matching algorithm has been created in order to track the movement of clusters over time accurately. The idea behind this algorithm is to observe how clusters evolve from frame to frame. Due to the imposed drift movement, some neurons may shift, appear or disappear over time and therefore being able to monitor them is an essential task in correcting this motion. This algorithm aims to match each cluster from a given time frame to the corresponding cluster from the next time interval.

Following the clustering step, every cluster position was stored in a dictionary. Therefore, the matching algorithm constructs a distance matrix for each two consecutive time frames. For instance, if the first frame contains n clusters and the second frame contains m clusters, it will build an n x m matrix. As a result, all pairwise distances between every two clusters from the given time frames will be put into the matrix. Next, the algorithm finds the smallest element of the matrix, the smallest distance, and take that as a pair. Following this, the row and column of the corresponding pair are filled with a very high number, and then the algorithm starts searching for the minimum distance again. The algorithm converges when the minimum distance found is larger than a set threshold, also called ’greedy search’.

As displayed in Figure 3.4, the matching algorithm successfully groups the closest pairs, denoted in red and green. The long euclidean distances represented by the long lines are considered mismatches. Compared to the matching on the improved clustering method, the number of mismatches is reduced, and more reliable pairs are created. In the next sections, we will explore how this improved matching will impact the correction accuracy.

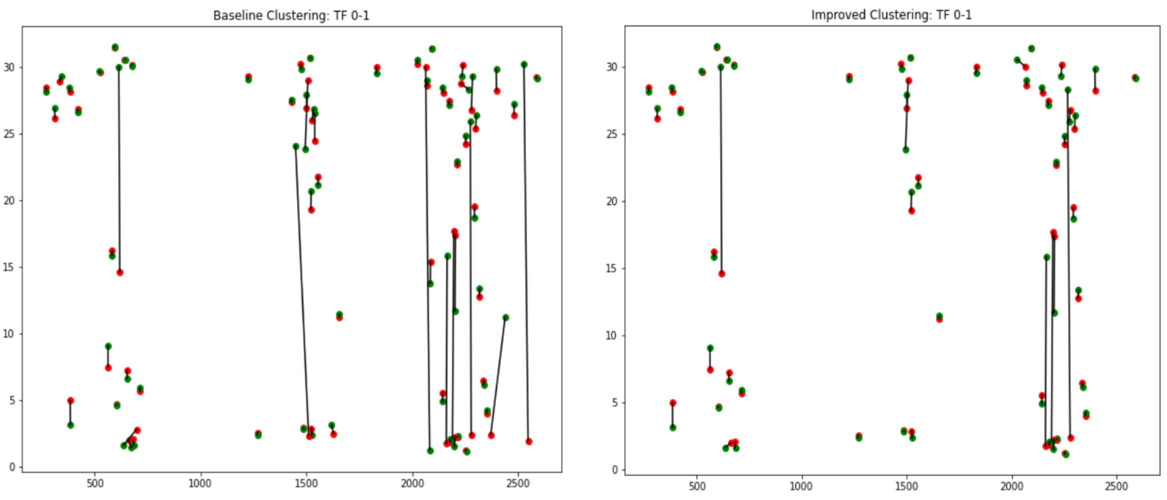


Figure 3.4: Matching pairs of clusters following both baseline and improved clustering on the first two time frames (TF 0&1). On the *x-axis* the clusters’ x-locations are displayed, whereas on the *y-axis* are their y-coordinates. Green dots represent cluster positions from TF 0 in their x&y coordinates, while red dots represent cluster position from TF 1. Every line denotes the euclidean distance between them. A longer line outlines a longer distance.

The algorithm’s output is a dictionary containing all the matched pairs for each two consecutive time frames, which stores the euclidean distance for each pair as a value. The presented solution is a variation of the Hungarian Match algorithm (Kuhn, 1955), which was also taken into consideration as an alternative approach. However, its complexity was relatively high, and the presented algorithm performed well and efficiently on the recorded data since operations over matrices are usually fast. Moreover, for an accurate matching, a threshold was required which makes the method different from the classical Hungarian Match algorithm. Therefore, the described matching solution is also scalable on longer recordings or larger probes.

3.3 Drift profile

The whole matching step’s main purpose was to compute the displacements between the closest pairs for each two given time frames. This computation leads to the next step in our pipeline, which calculates the average displacement of each time interval, which will construct the drift profile.

By analysing the data in Figure 3.5, it can be observed how the displacement differs between the beginning and middle of the recording. Examining the very beginning of the recording (first two time frames), where there is not too much activity, it can be noticed that most of the x-axis displacements are around 0, which denotes that interval 1 did not move much regarding interval 0, as expected. However, when exploring the middle of the recording (time frames 48 & 49), where the imposed drift is applied, it can be remarked that numerous x-axis displacements reach values different than 0. Following this, it shows there is considerable activity, and that time interval 49 shifted significantly from time frame 48.

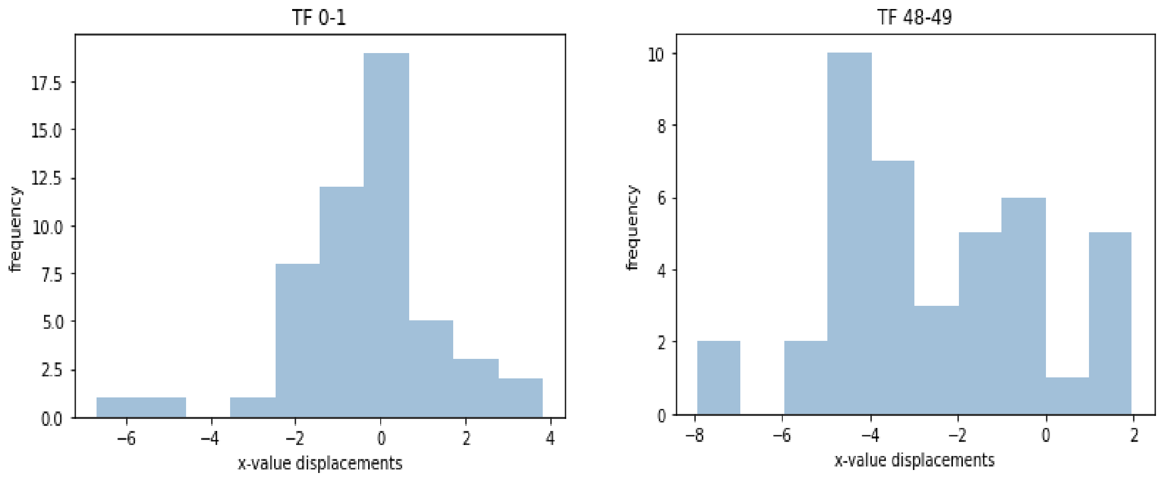


Figure 3.5: Frequency of displacements of cluster pairs at the beginning (TF 0 & 1) and middle of the recording (TF 48 & 49). The differences of x-coordinates of clusters are represented on the *x-axis*, while the frequency of the differences is displayed on the *y-axis*.

3.3.1 Average Displacement

Clusters' movements within a time frame will be averaged to one value representing the overall shift of the entire interval, given the precedent one. This average displacement per time interval is a straightforward metric that characterises and quantifies the drift reliably. Plotting this metric for each time frame over time, as shown in Figure 3.6, creates the drift profile and shows how it evolves from time frame to time frame.

As expected, the 'zig-zag' pattern is predominant, representing the imposed movement during the recording. Having a lower magnitude, the beginning and end of the data are more stable as the observed motion is caused by actual brain movement. However, the middle of the recording has a much higher magnitude, which denotes that the drift was manually imposed during that time interval. Moreover, because spike locations are represented in 2D coordinates, there are two types of drift, vertical and horizontal. During the beginning and end of the recording, where the brain naturally imposes the motion, both drift types are very similar. On the other hand, throughout the imposed motion interval, the probe is moved vertically, which affects the vertical axis, hence the vertical drift, much more than the horizontal one, which explains the large magnitude represented in blue in Figure 3.6.

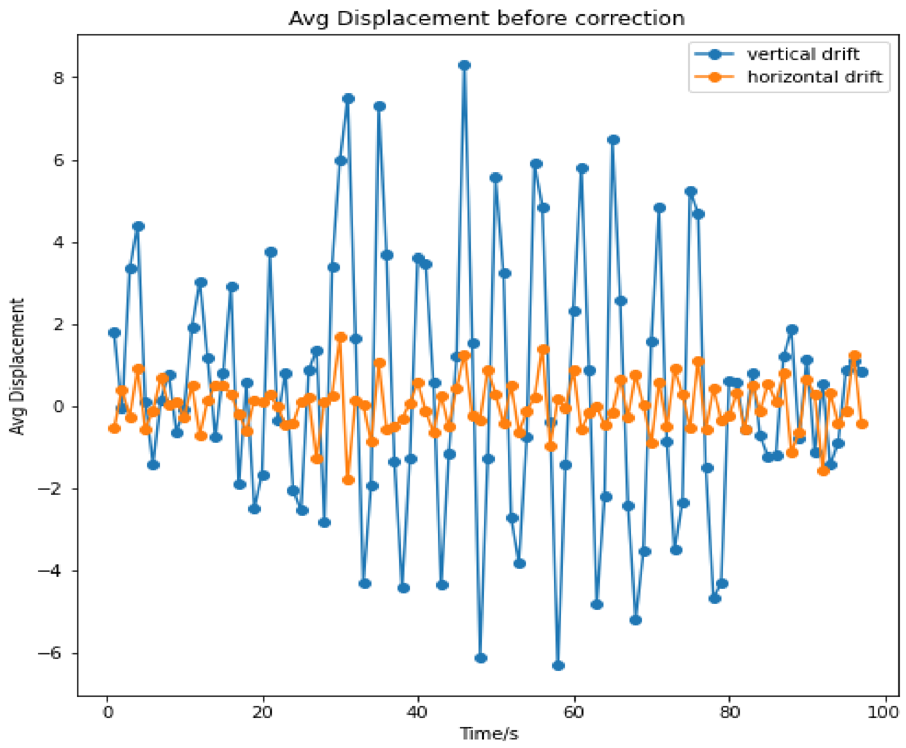


Figure 3.6: Drift profile represented as displacements (µm) over time (s). Vertical drift describes the differences between the x-values, whereas horizontal drift represents the differences between the y-values. Each '-o' mark denotes a time frame.

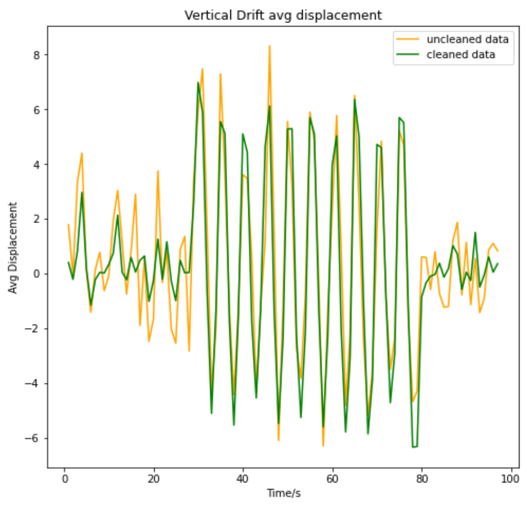
3.3.2 Data cleaning

Even if the improved clustering has substantially reduced the number of mismatched clusters from consecutive time frames, the previously computed average displacement from each time interval is still high since there are still matched pairs with long distances between them. In order to overcome this issue, cleaning the data was necessary.

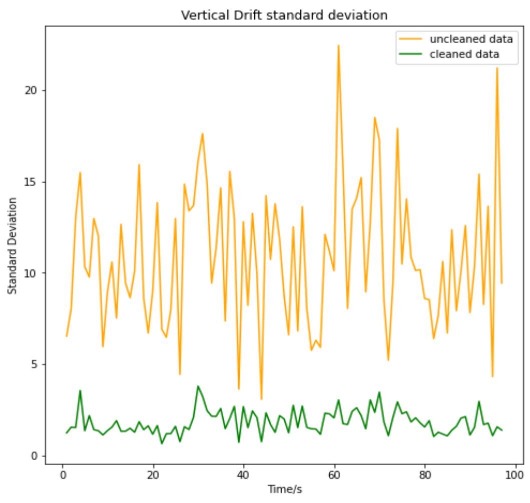
Therefore, a threshold has been set that is used to discard the remaining mismatched pairs. As shown in Equation 3.1, the *threshold* represents an interval. Given time frame t, the interval's limits are computed by subtracting and adding half of the standard deviation to the average displacement for the given time interval t.

$$threshold = (AvgDisp[t,t+1] - \frac{Std[t,t+1]}{2}, AvgDisp[t,t+1] + \frac{Std[t,t+1]}{2}) \quad (3.1)$$

As a result, pairs' distances that are not within the previously computed interval are disregarded, resulting in much cleaner data containing reliable displacements. To emphasise this filtering of the data, a lower magnitude in the average displacement of the cleaned data compared to the uncleaned one can be observed in Figure 3.7 (a). Moreover, the standard deviation value dropped significantly, as can be observed in the standard deviation plot in Figure 3.7 (b). This metric shows that the cleaned dataset's displacements represent real matches of clusters with values in a much smaller range (0, 4) than the uncleaned dataset range (4, 22). Having the data cleaned and preprocessed, the next section will explore different attempts of correcting the drift movement.



(a) Average displacement of the vertical drift displayed as displacements (μm) over time (s).



(b) Standard deviation of the vertical drift displayed as displacements (μm) over time (s).

Figure 3.7: Uncleaned vs cleaned data regarding the average displacement and standard deviation over time.

3.4 Drift correction

The last step in this pipeline is the imposed movement correction. Due to the problem complexity, it was challenging to achieve reliable results. A couple of solutions have been tried, and it will be explored further how the drift profile evolved for each separately implemented solution. To evaluate whether a given approach has worked and how well it performed, the cumulative displacement metric will be used. This metric denotes the cumulative sum of all the average displacements for each time frame. A representation of this measure is shown in Figure 3.8, which presents a comparison between the vertical drift of uncleaned and cleaned data before any correction. As expected, the cleaned data has a lower cumulative displacement.

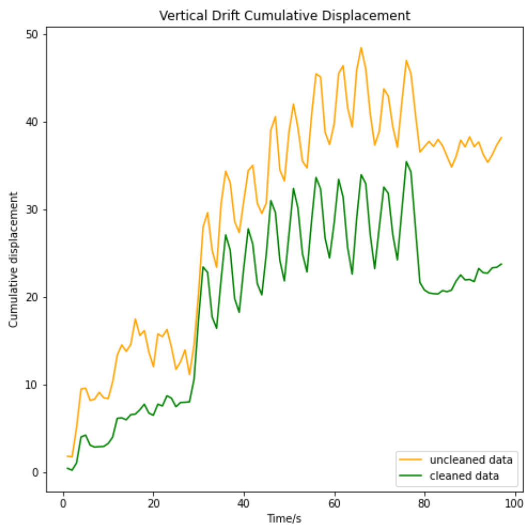


Figure 3.8: Vertical drift profile represented as cumulative displacements (µm) over time (s) for both uncleaned and cleaned data.

Chapter 4

Evaluation

4.1 Correction by average displacement

Given interval *t* and its previous interval *t-1*, the movement of every cluster within frame *t* is averaged. The first straightforward approach is to use this average displacement metric, computed for each time interval, to correct the movement. Frame by frame, the main idea is to move every spike that falls into that window by the same displacement value. This method removes approximately 40% of the drift, performing relatively well as a baseline solution. The correction is shown in Figure 4.1, and it can be noticed how the cumulative displacement metric is reduced.

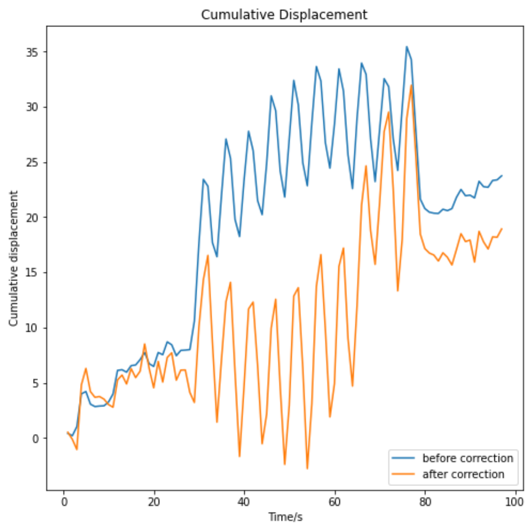


Figure 4.1: Vertical drift profile represented as cumulative displacements (µm) over time (s) before and after correction by average displacement.

However, the reduction is low, and the resulting displacement is still high. Moreover, using the same value for every spike correction within an interval will produce an unsmooth movement. The mean displacement is perfect for the midpoint between that time frame. However, spikes that do not fall at precisely the midpoint within that time interval will be displaced either less or more, depending on their position relative to the midpoint. Therefore, a more rigorous method needs to be applied.

4.2 Correction by interpolation

An improved approach in dealing with drift is to use linear interpolation. Linear interpolation is the process of drawing a straight line through the average displacements that are already computed. Every time point gets a value by interpolating the missing values in between the already known midpoints. Therefore, this method implies that every spike within a time frame will get a displacement value. This computation can be more precise about the direction of the movement, and every spike is displaced by the right amount, which will be slightly different for each time point.

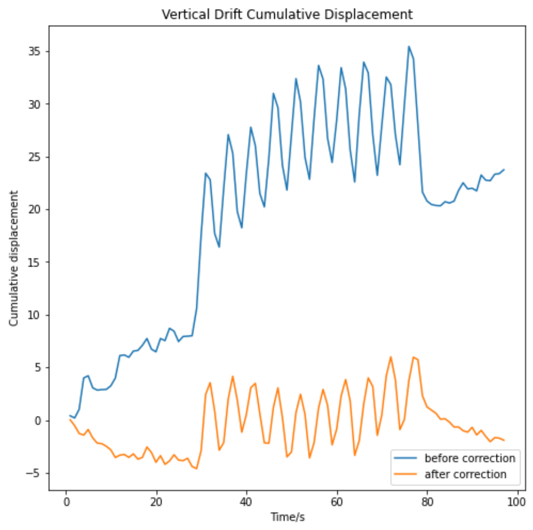


Figure 4.2: Vertical drift profile represented as cumulative displacements (µm) over time (s) before and after correction by interpolation.

As shown in Figure 4.2, the movement is corrected, and the remaining drift has been notably reduced to values close to 0. Moreover, the 'zig-zag' pattern has been significantly diminished, demonstrating the efficiency of the method. However, as it can be noticed, there is still some drift left, and the correction is not perfect. The remaining drift might be caused by the probe's non-rigid displacement. Different parts of the probe might move in different directions and correcting each section separately might perform better than the previous correction. Next, it will be explored how different sections drift and whether this non-rigid movement can be further improved.

4.2.1 Section analysis

To try to further improve the interpolation method described above, a section analysis has been done in order to study the rigidness of the drift. The probe length has been split into three equal sections, and each section is separately inputted into the created pipeline. Therefore, every section will be clustered, matched and corrected (by interpolation) independently. Next, they are put back together, and the remaining drift is compared to the drift that remains when correcting the whole probe directly. As before, the cumulative displacement of each section has been computed and shown in Figure 4.3. It can be observed that section 3 drifts in a different direction than sections 1 & 2, which makes the probe's drift non-rigid.

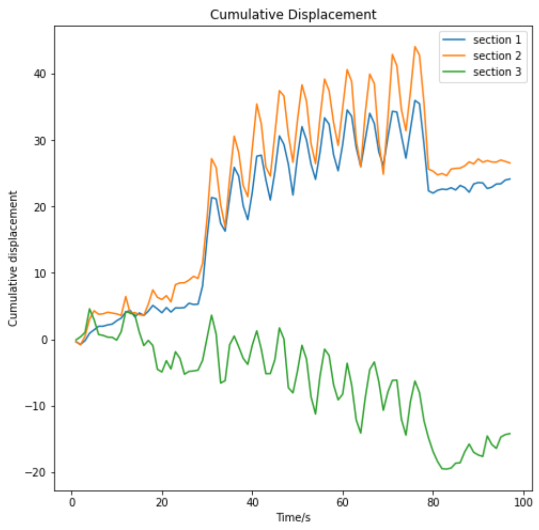


Figure 4.3: Cumulative displacement of vertical drift represented as cumulative displacements (µm) over time (s) for each of the three sections before correction.

The last step of the pipeline is the interpolation correction. Then, all three corrected sections are put back together and re-clustered as a whole. In Figure 4.4, both the whole probe and section corrections are displayed, as long as the cumulative displacement of the probe before any improvement. Correcting each section separately gives an absolute average displacement of 2.4 µm, while the previous correction on the whole probe is approximately 0.5 µm. Therefore, this analysis did not improve the correction. This soft correction might happen because the interpolated values from each section independently are not reliable enough to offer a stronger improvement.

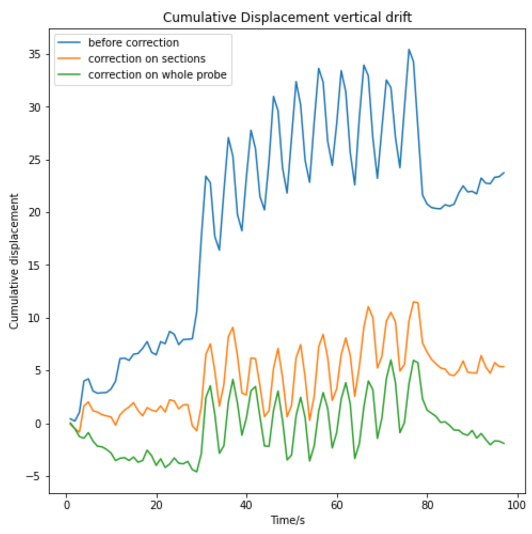
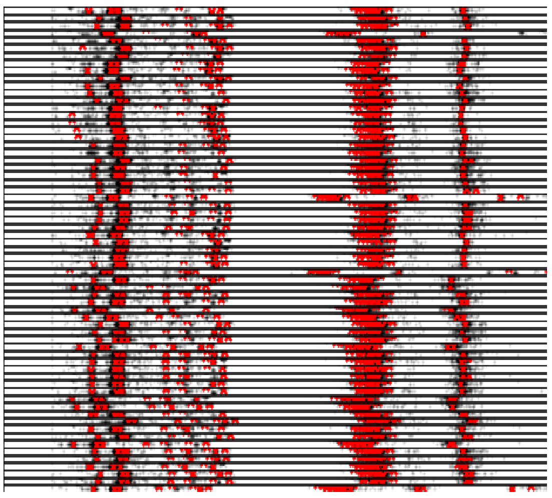


Figure 4.4: Cumulative displacement of vertical drift represented as cumulative displacements (µm) over time (s) for both types of correction comparing to the uncorrected movement.

4.3 Performance on datasets with no imposed drift

To further examine the performance of the developed pipeline, datasets with no imposed drift have been tested. The main approach is to check whether the algorithm can correct a more subtle motion. Once again, the cumulative displacement of the datasets before and after correction is used as an evaluation metric of the algorithm's performance.

Firstly, the raw data of each dataset is presented in Figure 4.5. Even if neither of them contains an imposed drift, a slighter 'zig-zag' pattern can still be noticed for both datasets, representing the natural brain movement visible during the recording. Attempting to correct this tendency will identify both the strengths and limitations of the pipeline.


(a) Allen Mouse dataset

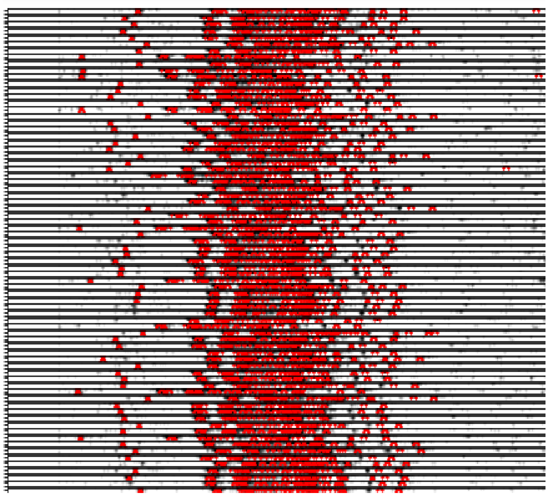
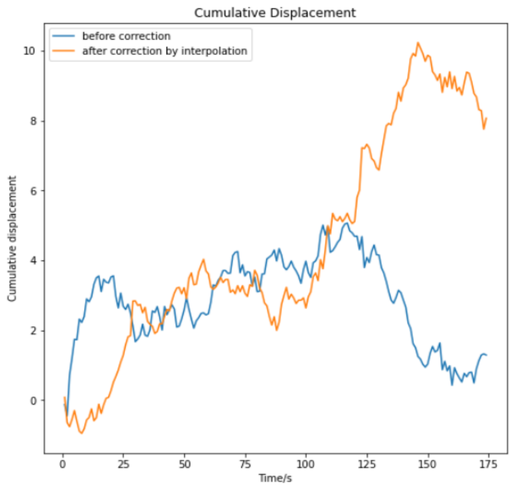
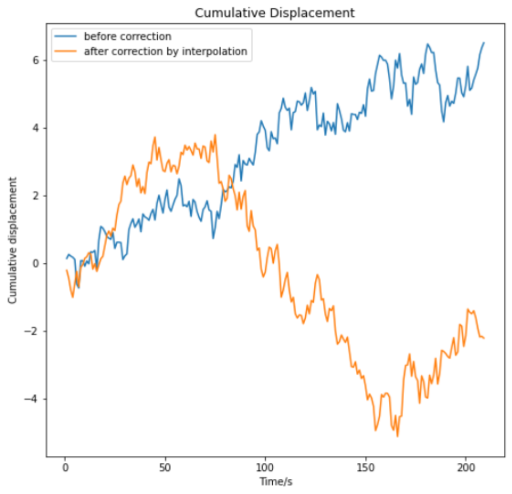

(b) Svoboda dataset

Figure 4.5: Part of the clustered raw data frame by frame for both datasets with no imposed drift. The probe's long axis coordinates of spikes are displayed on the x-axis, whereas time (s) is represented on the y column for each time interval.

The first approach is to correct both probes by interpolation, as this method proved to be the best solution so far. Using the same parameter configuration as for the imposed drift dataset, the interpolation correction performs worse than expected, as shown in Figure 4.6. The obtained result identifies one limitation of the presented method. The initial threshold (3.1) might disregard some large displacements of some time intervals that would be used to generate the interpolated values. Therefore, because the displacements within the threshold are too small, the values used for correction will not be large enough to reliably correct the movement.



(a) Allen Mouse dataset

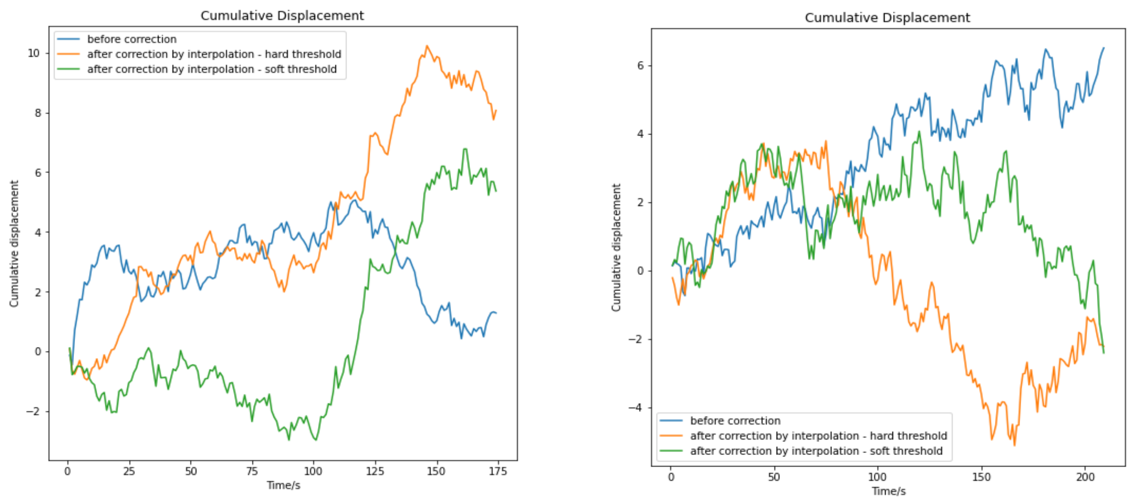


(b) Svoboda dataset

Figure 4.6: Cumulative displacement of the vertical drift displayed as displacements (µm) over time (s) for the whole probe before correction and after correction by interpolation using the initial parameter configuration.

As an attempt to overcome this issue, a softer threshold has been applied to the correction pipeline. However, a too permissive threshold would also take too large displacements into account. The matching algorithm would then create mismatches that will generate large values which would make the correction unreliable. Therefore, the value of the current threshold is a trade-off between these two limitations. This new softer threshold, displayed in Equation 4.1, will cover large enough displacements that will give a better estimate for the interpolated values. This approach is expected to have a better correction than the previous one on the given datasets. The results shown in Figure 4.7 proved how this more flexible filtering improved the correction significantly. The cumulative displacement of the newly corrected probe is closer to 0, offering a more reliable correction than the previous one.

$$threshold = (AvgDisp[t,t+1] - Std[t,t+1], AvgDisp[t,t+1] + Std[t,t+1]) \tag{4.1}$$



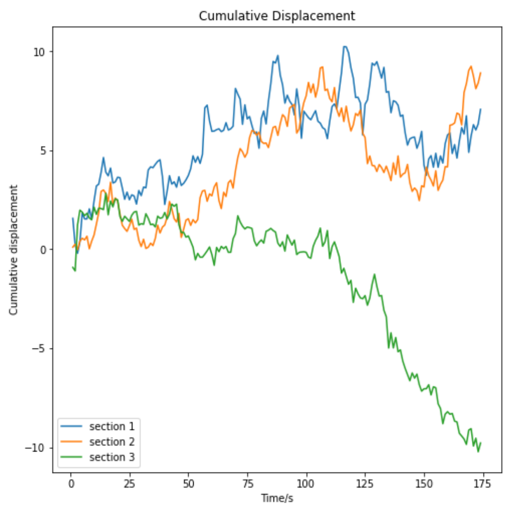
(a) Allen Mouse dataset

(b) Svoboda dataset

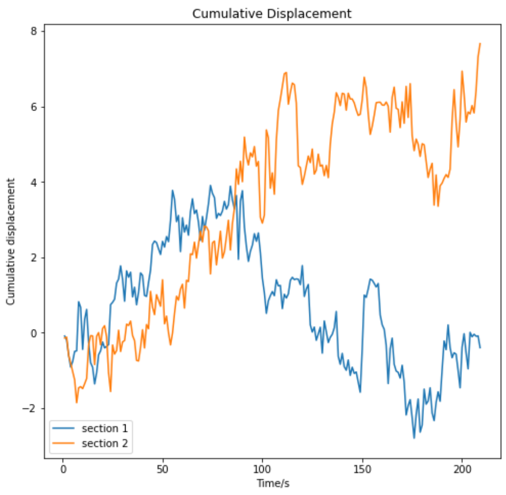
Figure 4.7: Cumulative displacement of the vertical drift displayed as displacements (µm) over time (s) for the whole probe before correction and after correction by interpolation with the new threshold represented by the green line.

4.3.1 Section analysis

Since the correction is still not perfect and it can further be improved, a section analysis is performed in the same manner as before. Given the form of each recording, they were split into three (Figure 4.8 (a)) and two (Figure 4.8 (b)) sections respectively. The sub-figures show that the movement is non-rigid in both cases, meaning that different parts move in different directions. Therefore, because of the non-rigidness, individually correcting each section could offer a better correction than correcting the whole probe.



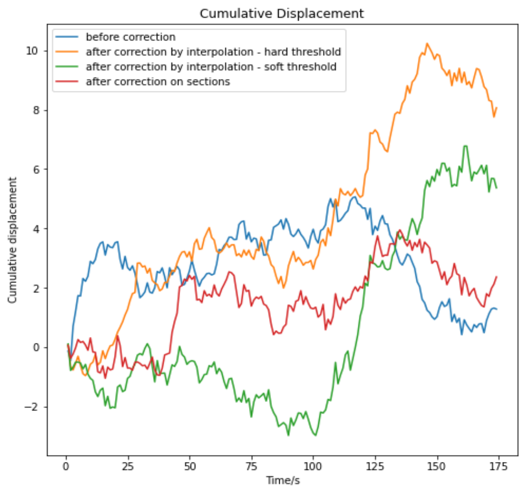
(a) Allen Mouse dataset



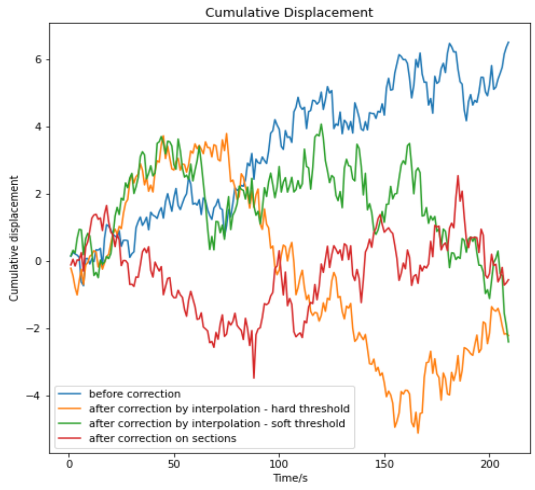
(b) Svoboda dataset

Figure 4.8: Cumulative displacement of the vertical drift displayed as displacements (µm) over time (s) before correction for each section of each dataset.

As it can be observed in Figure 4.9, the section correction performs relatively well for both datasets. Relating to the interpolation correction with a hard threshold, it is significantly better, being close to the correction on the soft threshold. These results indicate that correction on datasets with non-rigid drift might be more reliable with the section approach. Correcting each section individually will offer more stable results in term of displacements.



(a) Allen Mouse dataset



(b) Svoboda dataset

Figure 4.9: Cumulative displacement of the vertical drift displayed as displacements (µm) over time (s) for all types of corrections.

Table 4.1 summarises all type of corrections on all datasets tested on the created pipeline. The presented values represent the absolute average displacements of the vertical drift's long axis. Large values denote a large average displacement during the recording, whereas values closer to 0 indicate a good correction with an almost fully removed ’zig-zag’ pattern.

Absolute Average Vertical Displacement (µm)				
Dataset	**Before correction**	**Corrected by interpolation (hard threshold)**	**Corrected by interpolation (soft threshold)**	**Correction on sections**
Imposed Drift	19.38	0.50	4.67	2.40
Allen Mouse	2.85	4.40	0.61	1.45
Svoboda	3.28	0.37	1.67	0.35

Table 4.1: Correction results represented as absolute average vertical displacements of long axis (µm)

As expected, the initial dataset including the imposed drift contains the largest displacement before any correction. For this specific dataset, the original pipeline with the initial hard threshold performs the best out of all other approaches, correcting the imposed motion almost entirely. The soft threshold takes larger displacements into account, creating mismatches and, therefore performing the worst. Section correction performs relatively well, but not better than the initial approach. The Allen Mouse dataset contains no imposed drift but has some strong and sudden motions as displayed in Figure 4.5 (a). The initial threshold seems to miss those strong movements and therefore, it does not correct the motion reliably. A more flexible threshold captures those sudden movements as well, generating more reliable interpolated values that correct the drift better. Testing the Svoboda dataset, which does not contain any imposed drift (Figure 4.5 (b)), seems to perform similar to the initial dataset. The original threshold corrects the movement relatively well while changing the threshold does not work any better. Finally, the sections correction works best for this type of dataset.

Chapter 5

Discussion

A pipeline for tracking and correcting non-stationary data in extracellular recordings has been successfully introduced. This model is represented as an extension of a spike sorting algorithm. Every spike sorter can efficiently apply the pipeline on the spike locations and improve their sorting accuracy. This advancement will automate the spike sorting process by replacing the manual curation step. Every part of the pipeline was successfully designed to be easily applied to all types of datasets. The imposed drift dataset has been analysed in detail and after few trials, optimal values were set for the clustering phase and the splitting by time. Each step of the pipeline was implemented efficiently as a Python method. The only required input is to the clustering function that needs the 2D spike locations. Moreover, before correction, the pipeline also quantifies the drift (as cumulative displacement), a very important measurement, which has not been done until now. The improved final correction step is also working properly, and it successfully removes the electrode drift movement. As a result, every spike sorter can make use of this pipeline, dramatically improving the sorting accuracy while replacing the human intervention step, which saves considerable time.

Clustering & Matching
The MeanShift clustering algorithm proved to be a very good solution for this correction approach (Comaniciu and Meer, 2002). The main advantage is that it does not require the number of clusters to be established in advance, crucial aspect datasets with millions of data-points. Moreover, only one parameter, the bandwidth, is required, which was relatively easy to optimise. The clustering method takes the spike locations as an argument, which is the only needed input to start the pipeline. After rigorous testing, to further improve the clustering process, only clusters containing more than 30 spikes were considered for matching. This feature is added to the baseline clustering method. The matching algorithm is a straightforward method that pairs the closest clusters from consecutive frames. However, this pairing is not accurate at all times and therefore it comes with some limitations. These limitations will be further discussed in section 5.1.

Data cleaning
The cleaning process is an essential step to the pipeline. It significantly improves the accuracy of the model by filtering the large displacements coming from mismatches

(i.e. errors of the matching algorithm). Depending on the type of dataset used and how large the displacements are, the cleaning process can be adapted by changing the threshold value. Moreover, during the cleaning step, the drift is also quantified in terms of displacements. This measurement is an important feature of the drift used for correction, that no other model uses in their algorithm.

Correction
The correction step was the most difficult part of the pipeline. The baseline correction was relatively easy to implement but yielded unreliable results. Correcting each time frame by the same amount is partially correcting the movement, but can not be considered a reliable solution, offering an unsmooth movement. Therefore, interpolation has been used to reliably correct the drift. By interpolation, each spike within each segment gets its value used for correction. Now, each spike will be corrected by its single value rather than the same average displacement. As expected, the correction is significantly improved and the 'zig-zag' pattern is almost fully resolved.

Evaluation
The implemented pipeline has been tested on two other datasets with no imposed drift. Testing datasets with low displacements show the real performance of the pipeline and its applicability among all types of datasets. On the dataset with a very slight drift (Svoboda), the pipeline performs well, reducing the movement to almost 0. For the other dataset (Allen Mouse), a different threshold had to be used for cleaning in order to reliably remove the drift.

5.1 Limitations

Summarising the evaluation analysis, it can be stated that for some datasets, some parameter optimisation is needed. That is because the matching step, together with the cleaning step, come with some limitations. There is an important trade-off between the pairs created by the matching algorithm and the threshold used to filter these matches. Some pairs of clusters might be too far away from each other (mismatches), representing a big outlier in terms of displacements. As a result, a too flexible threshold would not filter the outliers and that will influence the further correction. Errors accumulate through this procedure of calculating the cumulative displacement. All the errors that are in between the outlier and the end of the recording, are accumulated, getting worse over time if too much noise is present.

On the other hand, a too strict threshold might not include big displacements which are not mismatches and therefore, the interpolated values used for correction would not be strong enough to remove the drift.

5.2 Going further

As shown before, the correction is not perfect. The interpolation method significantly reduces the movement, but there is still some unresolved drift. A significant improvement to the pipeline would be to remove the cumulative displacement effect, described in section 5.1.

One solution would be to aggregate the noise and therefore preventing noise cumulation. To achieve this result, an all to all comparison against all possible pairings in the recording should be made. Next, average out the noise effects since the consistent movement is predictable based on different comparisons.

Bibliography

M. Ballini, J. Müller, P. Livi, Y. Chen, U. Frey, A. Stettler, A. Shadmani, V. Viswam, I. L. Jones, D. Jäckel, M. Radivojevic, M. K. Lewandowska, W. Gong, M. Fiscella, D. J. Bakkum, F. Heer, and A. Hierlemann. A 1024-channel cmos microelectrode array with 26,400 electrodes for recording and stimulation of electrogenic cells in vitro. *IEEE Journal of Solid-State Circuits*, 49(11):2705–2719, 2014. doi: 10.1109/JSSC.2014.2359219.

Aharon Bar-Hillel, Adam Spiro, and Eran Stark. Spike sorting: Bayesian clustering of non-stationary data. *Journal of neuroscience methods*, 157(2):303–316, 2006.

Luca Berdondini, PD Van Der Wal, Olivier Guenat, Nicolaas F de Rooij, Milena Koudelka-Hep, P Seitz, R Kaufmann, P Metzler, N Blanc, and S Rohr. High-density electrode array for imaging in vitro electrophysiological activity. *Biosensors and bioelectronics*, 21(1):167–174, 2005.

Antal Berényi, Zoltán Somogyvári, Anett J Nagy, Lisa Roux, John D Long, Shigeyoshi Fujisawa, Eran Stark, Anthony Leonardo, Timothy D Harris, and György Buzsáki. Large-scale, high-density (up to 512 channels) recording of local circuits in behaving animals. *Journal of neurophysiology*, 111(5):1132–1149, 2014.

Alessio P Buccino, Cole L Hurwitz, Samuel Garcia, Jeremy Magland, Joshua H Siegle, Roger Hurwitz, and Matthias H Hennig. Spikeinterface, a unified framework for spike sorting. *Elife*, 9:e61834, 2020.

Ana Calabrese and Liam Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196(1):159–169, 2011.

Matt Carter and Jennifer C Shieh. *Guide to research techniques in neuroscience*. Academic Press, 2015.

Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

Jason E Chung, Jeremy F Magland, Alex H Barnett, Vanessa M Tolosa, Angela C Tooker, Kye Y Lee, Kedar G Shah, Sarah H Felix, Loren M Frank, and Leslie F Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5): 603–619, 2002.

Roland Diggelmann, Michele Fiscella, Andreas Hierlemann, and Felix Franke. Automatic spike sorting for high-density microelectrode arrays. *Journal of neurophysiology*, 120(6):3155–3171, 2018.

AA Emondi, SP Rebrik, AV Kurgansky, and KD Miller. Tracking neurons recorded from tetrodes across time. *Journal of neuroscience methods*, 135(1-2):95–105, 2004.

Michale S Fee, Partha P Mitra, and DAVID Kleinfeld. Variability of extracellular spike waveforms of cortical neurons. *Journal of neurophysiology*, 76(6):3823–3833, 1996.

Felix Franke, David Jäckel, Jelena Dragas, Jan Müller, Milos Radivojevic, Douglas Bakkum, and Andreas Hierlemann. High-density microelectrode array recordings and real-time spike sorting for closed-loop experiments: an emerging technology to study neural plasticity. *Frontiers in neural circuits*, 6:105, 2012.

Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975.

Matthias H Hennig, Cole Hurwitz, and Martino Sorbaro. Scaling spike detection and sorting for next-generation electrophysiology. In *In Vitro Neuronal Networks*, pages 171–184. Springer, 2019.

Gerrit Hilgen, Martino Sorbaro, Sahar Pirmoradian, Jens-Oliver Muthmann, Ibolya Edit Kepiro, Simona Ullo, Cesar Juarez Ramirez, Albert Puente Encinas, Alessandro Maccione, Luca Berdondini, et al. Unsupervised spike sorting for large-scale, high-density multielectrode arrays. *Cell reports*, 18(10):2521–2532, 2017.

Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

Eyal Hulata, Ronen Segev, and Eshel Ben-Jacob. A method for spike sorting and detection based on wavelet packets and shannon's mutual information. *Journal of neuroscience methods*, 117(1):1–12, 2002.

Cole L Hurwitz, Kai Xu, Akash Srivastava, Alessio P Buccino, and Matthias H Hennig. Scalable spike source localization in extracellular recordings using amortized variational inference. *arXiv preprint arXiv:1905.12375*, 2019.

James J Jun, Catalin Mitelut, Chongxi Lai, Sergey L Gratiy, Costas A Anastassiou, and Timothy D Harris. Real-time spike sorting platform for high-density extracellular probes with ground-truth validation and drift correction. *BioRxiv*, page 101030, 2017.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Baptiste Lefebvre, Pierre Yger, and Olivier Marre. Recent progress in multi-electrode spike sorting methods. *Journal of Physiology-Paris*, 110(4):327–335, 2016.

Michael S Lewicki. Bayesian modeling and classification of neural signals. *Neural computation*, 6(5):1005–1030, 1994.

Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4): R53–R78, 1998.

Jens-Oliver Muthmann, Hayder Amin, Evelyne Sernagor, Alessandro Maccione, Dagmara Panas, Luca Berdondini, Upinder S Bhalla, and Matthias H Hennig. Spike detection for large neural populations using high density multielectrode arrays. *Frontiers in neuroinformatics*, 9:28, 2015.

ZOLTÁN Nadasdy, JOZSEF Csicsvari, MARKKU Penttonen, JAMILLE Hetke, KENSALL Wise, and GYÖRGY Buzsaki. Extracellular recording and analysis of neuronal activity: from single cells to ensembles. *Neuronal Ensembles: Strategies for Recording and Decoding*, pages 17–55, 1998.

Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, et al. Kilosort: realtime spike-sorting for extracellular electrophysiology with hundreds of channels. *BioRxiv*, page 061481, 2016.

Klas H Pettersen and Gaute T Einevoll. Amplitude variability and extracellular low-pass filtering of neuronal spikes. *Biophysical journal*, 94(3):784–802, 2008.

Jonathan W Pillow, Jonathon Shlens, EJ Chichilnisky, and Eero P Simoncelli. A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings. *PloS one*, 8(5):e62123, 2013.

Jason S Prentice, Jan Homann, Kristina D Simmons, Gašper Tkačik, Vijay Balasubramanian, and Philip C Nelson. Fast, scalable, bayesian spike identification for multi-electrode arrays. *PloS one*, 6(7):e19884, 2011.

R Quian Quiroga, Zoltan Nadasdy, and Yoram Ben-Shaul. Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural computation*, 16(8):1661–1687, 2004.

Hernan Gonzalo Rey, Carlos Pedreira, and Rodrigo Quian Quiroga. Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117, 2015.

Cyrille Rossant, Shabnam N Kadir, Dan FM Goodman, John Schulman, Maximilian LD Hunter, Aman B Saleem, Andres Grosmark, Mariano Belluscio, George H Denfield, Alexander S Ecker, et al. Spike sorting for large, dense electrode arrays. *Nature neuroscience*, 19(4):634–641, 2016.

Shy Shoham, Matthew R Fellows, and Richard A Normann. Robust, automatic spike sorting using mixtures of multivariate t-distributions. *Journal of neuroscience methods*, 127(2):111–122, 2003.

RK Snider and AB Bonds. Classification of non-stationary neural signals. *Journal of neuroscience methods*, 84(1-2):155–166, 1998.

Nicholas A Steinmetz, Christof Koch, Kenneth D Harris, and Matteo Carandini. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current opinion in neurobiology*, 50:92–100, 2018.

Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139–142, 2011.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Pierre Yger, Giulia LB Spampinato, Elric Esposito, Baptiste Lefebvre, Stéphane Deny, Christophe Gardella, Marcel Stimberg, Florian Jetter, Guenther Zeck, Serge Picaud, et al. A spike sorting toolbox for up to thousands of electrodes validated with ground truth recordings in vitro and in vivo. *Elife*, 7:e34518, 2018.