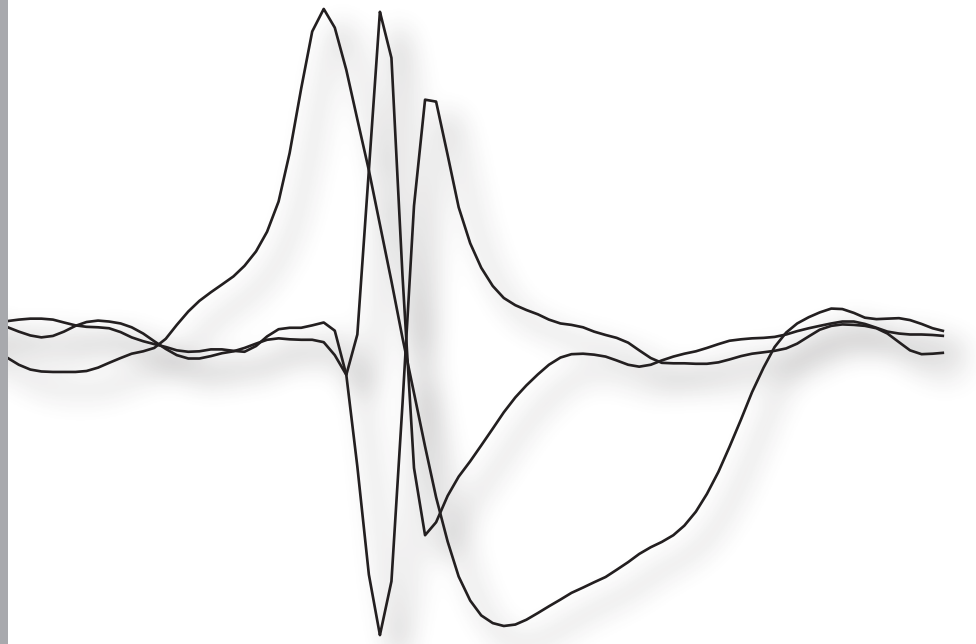
Spike sorting algorithm for intra-cortical recordings using unsupervised Bayesian decomposition



Master Thesis

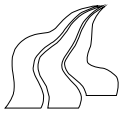
10th semester

Project group: 1085c

Martin Nøhr Nielsen

Department of Health Science and Technology
Aalborg University 2009

The Faculty of Engineering, Science and Medicine
Aalborg University



Title:

Spike sorting algorithm for intra-cortical recordings using unsupervised Bayesian decomposition

Introduction:
Spike sorting is the process of isolating neural signals and assigning each recorded waveform to the neuron of origin. Many spike sorting methods rely on interaction from an expert operator, which can be time consuming, arbitrary, and inaccurate.

Method:
The focus in this study is a novel unsupervised spike sorting algorithm, based on unsupervised Bayesian decomposition (UBD). The Bayesian statistical model and a maximum a posterior (MAP) estimator are originally designed for intra-muscular EMG signals, but are in this present work tuned and used to solve the problem of spike sorting from intra-cortical recordings in a fully automatic way. The UBD method is validated with both simulated and human intra-cortical recordings, and compared with a classical unsupervised spike sorting method (Wave_Clus) for performance evaluation.

Results:
The UBD method showed almost similar performance as Wave_Clus with both simulated and human intra-cortical recordings, and reached an average performance of 80.9 % and 83.2 % with simulated and human signals respectivily. Low performance at approximately 39 % was seen in certain cases with simulated signals with a short refractory period, whearas the UBD method showed high performance in detecting and classifying overlapping spikes, compared to Wave_Clus.

Discussion:
Further development must be done, to increase the performance of the UBD method with intra-cortical recordings. Among these, a re-tuning of the TABU algorithm to enable the detection and classification of more than three spikes per segment, and a multi-channel extension to the UBD method will improve the performance by exploiting the inter-channel inference.

Project period:
10th semester
February 1st 2009 - June 4th 2009

Project group:
1085c

Group member:
Martin Nøhr Nielsen

Supervisor:
Professor Dario Farina

Number of copies: 3

Number of pages in main report: 83

Preface

This project report is the documentation of the work done by Martin Nøhr Nielsen as the 10^{th} semester master thesis at the Department of Health Science and Technology, Aalborg University, Denmark.

The study is written under the area of specialisation of Medical Systems with focus on biomedical signal processing, and the project period was from February 1st - June 4th 2009.

The author would like to thank the following persons, their contributions to this work are very appreciated:

- Ge Di, Institut de Recherche en Communications et Cybernetique de Nantes, for valuable assistance with the spike sorting methods.
- Winnie Jensen, lektor, Department of Health Science and Technology, Aalborg University, for providing data materials.
- Ernest Nlandu Kamavuako, PhD fellow, Department of Health Science and Technology, Aalborg University, for providing data materials.
- Supervisor of this project: Dario Farina, professor, Department of Health Science and Technology, Aalborg University

Aalborg University, June 2009

_____

Martin Nøhr Nielsen

Reading instructions:

This report consists of four major parts, which is intended to be read chronologically. The structure are shown in figure 1.

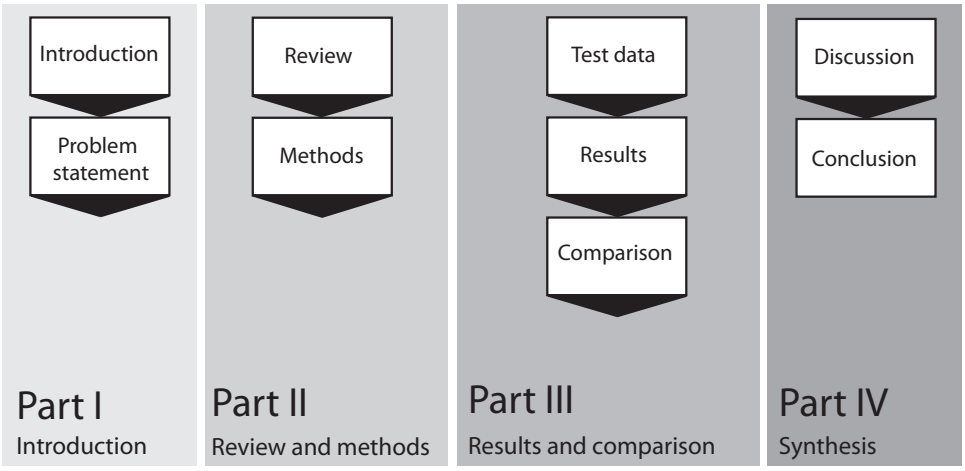


Figure 1: The structure of this report.

Contents

Preface

I

Introduction 1

Recent technological developments have made it possible to simultaneously record the spiking activity of increasingly large populations of cortical neurons using extracellular micro-electrodes. These spike trains may create a foundation for a variety of different applications, that utilise these recordings, including the development of neuroprosthetics that rely on control signals derived from the spike patterns of multiple neurons [Vargas-Irwin & Donoghue 2007], and the analysis of response profiles of single neurons/large ensembles of neurons, with the aim of understanding the principles by which a stimulus such as an odour, an image, or sound is represented within the brain [Horton, Nicol, Kendrick & Feng 2007]. Another example of an application is the use of spike patterns from extracellular recordings as a guide to localizing optimal sites for deep brain stimulation [Aksenova, Chibirova, Dryga, Tetko, Benabid & Villa 2003]. In general, spike trains are central to the analysis of neural data [Wood & Black 2008].
Recordings obtained with extracellular electrodes create a range of signal processing challenges, as each recording may contain signals from several neurons. In many scenarios, the identification of the unique spiking patterns of each single neuron is highly desirable [Vargas-Irwin & Donoghue 2007]. This introduces the time-consuming and non-trivial term "spike sorting", which is the process of isolating neural signals and assigning each recorded waveform to the neuron of origin [Lewicki 1998].

Action potentials recorded from a single neuron tend to have a stereotypical spike shape determined by the cell's structure and biophysical properties, but also by its position relative to the recording electrode. This spike shape is often used to verify that a set of waveforms are attributable to a single neuron, however, also other features, such as the firing history of the cell can introduce variation in waveform shape and amplitude [Vargas-Irwin & Donoghue 2007] [Fee, Mitra & Kleinfeld 1996*b*]. Moreover, waveform variation is also increased by inferring signals such as background activity from other neurons, or from other noise sources such as electrode drift in non-stationary recordings [Fee et al. 1996*b*] [Lewicki 1998] [Aksenova et al. 2003]. Another significant challenge in spike sorting is the complex sums of spike waveforms, due to recordings of multiple neurons on a given electrode. The decomposition of these overlapping spikes into their single-neuron components, generates great computational burden on spike sorting algorithms [Vargas-Irwin & Donoghue 2007]. Very few algorithms have been designed to handle spike overlaps, and ordinary spike sorting algorithms (i.e. based on wavelets [Quiroga, Nadasdy & Ben-Shaul 2004]) can only perform clustering with partially overlapping spikes [Herbst, Gammeter, Ferrero & Hahnloser 2008]. Many spike sorting algorithms dealing with the overlap-problem, suffer from the intractability of exhaustive searching [Herbst et al. 2008]. Other methods tries to overcome this intractability by limiting the number of spikes used to explain an overlap [Atiya 1992] [Lewicki 1994]. Zhang, Wu, Zhou, Liang & Yuan [2004] suggested only to search for overlapping spikes in those cases, where a fit by single spikes fails.

Many spike sorting methods rely in manual sorting by an expert. The usage of an experienced and knowledgeable human operator, who tries to provide a preliminary classification of the waveforms, can be time consuming, arbitrary, and inaccurate [Aksenova et al. 2003] [Bar-Hillel, Spiro & Stark 2006]. The optimal case is a spike sorting algorithm, that is both unsupervised and accurate [Harris, Henze, Csicsvari, Hirase & Buzsáki 2000]. Wood, Fellows, Donoghue & Black [2004] have reported an average of 23 % false positives and 30 % false negatives for manual spike sorting of synthetic signals performed by experts, and Harris et al. [2000] reports similar results. The accuracy of spike sorting critically affects the accuracy of all subsequent analyses [Brown, Kass & Mitra 2004]. Many different algorithms are used for spike sorting, but there is however no consensus as to which are best [Brown et al. 2004]. [Bar-Hillel et al. 2006] presents a fully automatic spike sorting method based on Bayesian clustering, which tries to mimic human experts in the clustering process.

The focus in this study is a novel unsupervised spike sorting algorithm, based on unsupervised Bayesian decomposition (UBD), developed by Ge, Carpentier & Farina [2009]. UBD is originally designed for the decomposition of intra-muscular EMG signals, but is in this project applied to the automatic identification and classification of spikes

from intra-cortical recordings. The UBD method also approaches the overlap-problem in spike sorting with a TABU search implementation. The UBD method is validated with both simulated and human intra-cortical recordings, to examine whether the UBD method can be applied to intra-cortical recordings, and compared with other selected unsupervised spike sorting algorithms for performance evaluation. The aim of the study is summarized in the following problem statement.

Problem statement

2

Based on the different problems related to spike sorting stated in the introduction, the point of departure of this report is defined in the following problem statement.

At what performance level can the UBD method perform spike sorting with simulated and human intra-cortical signals? How is that performance level compared with other classical spike sorting methods?

The aim of this report is to produce a review of spike sorting, including a selection of already developed spike sorting methods, with the purpose of selecting the methods for comparison. Furthermore, it is to tune and validate the UBD method with both simulated and human intra-cortical signals, and make a comparison of spike sorting performance using the UBD method and the selected classical spike sorting methods.

Figure 2.1 summarizes the following aims of the report.

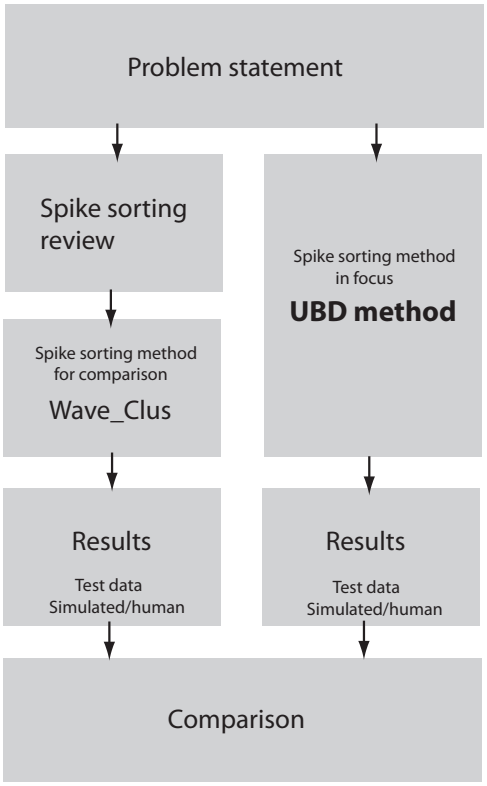


Figure 2.1: Summary of the aims of the report, with focus at the UBD spike sorting method and its performance using intra-cortical recordings.

Review and method

II

Review of spike sorting methods

3

The following chapter describes the principle of spike sorting. The objective is to produce a review of methods used for spike sorting, independent of the specific application in this study (the UBD), with the focus on methods applied to intra-cortical recordings. Moreover, it is to clarify which methods that have been studied previously in the literature. The review should constitute the basis or point of departure of the comparison study with the UBD method, such that a more classical spike sorting method can be identified for the comparison.

The following section will introduce the basic problem in spike sorting, and go through several well known methods and key issues.

3.1 Introduction

Spike sorting can be explained by the grouping of spikes into respective clusters based on their unique shapes. Each single neuron tends to fire spikes of specific shape, which results in clusters corresponding to the activity of each different neuron. The aim of spike sorting is to determinate which spike that corresponds to which of these neurons [Shoham, Fellows & Normann 2003] [Lewicki 1998]. Figure 3.1 illustrates the basic problem in spike sorting, with the extracellular waveform. Parts of neuroscience research focuses on the study of neuron activity recorded extracellular

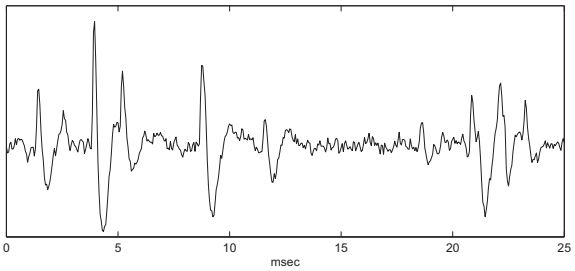


Figure 3.1: This example of an extracellular waveform shows several different action potentials generated by an unknown number of neurons. This illustrates the basic problem in spike sorting. Inspired by [Lewicki 1998]

with different types of electrodes [Brown et al. 2004] [Aksenova et al. 2003] [Vargas-Irwin & Donoghue 2007]. These electrodes record the activity of the most close-by neurons that fires action potentials or so-called spikes. An important property is that each neuron has spikes of characteristic shape. The uniqueness is determined by the morphology of the dendrite trees, and the distance/orientation relative to the recording electrode. [Gold, Henze, Koch & Buzsáki 2006]
Spike sorting has been called a very challenging problem in the literature [Brown et al. 2004]. Complex brain processes are reflected in the activity of large neural populations, therefore the study of single-cells in isolation gives only a limited view of the whole context [Horton et al. 2007].
The overall aim here is to record from a large population of neurons, to ensure this whole picture view. Spike sorting and the development of spike sorting algorithms are an important step towards that aim, so that it can allow the analysis of the activity of close-by neurons from each single recording electrode [Horton et al. 2007]. In figure 3.2 the basic principles in spike sorting are shown. The following will give an introduction to the basic steps in spike sorting.

3.2 Recordings

According to [Lewicki 1998], the first link between neural communication and electrical signals was found by Luige Galvani in 1791. He showed that frog muscles could be stimulated by electricity. In the 1920s it became possible to measure nerve impulses directly with amplified signals from microelectrodes. Usually the measured potentials are

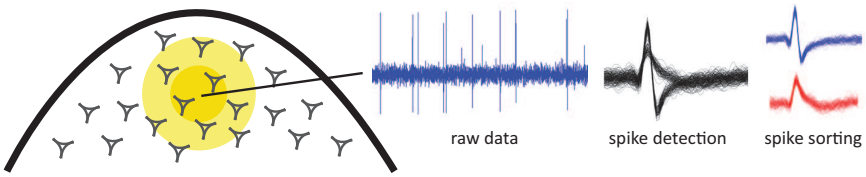


Figure 3.2: The basic principles in spike sorting. Extracellular recordings are normally done with electrodes in the brain. The signals from the electrodes are typically amplified and band-pass filtered, and the neuronal firing appears as spikes on top of background activity.

recorded between ground (a wire under the scalp) and the tip of the microelectrode. The potential changes measured reflects current flow in the extracellular medium, and the largest component of this current is typically generated by the action potential of the neuron [Schmidt 1984]. But one important issue in spike sorting is the noisy components. Lewicki [1998] states that signals that looks like cellular action potentials can be recorded from axonal fiber bundles, and that these signals are smaller than cellular action potentials. Another signal source is the field potential, found in layered structures and results from current flow into a parallel set of dendrites. The field potentials are typically of relatively low bandwidth, so they can be filtered out from the neural action potentials. [Lewicki 1998]

The shape of the recording electrode has effect on the quantity of neurons recorded. Roughly speaking, the larger the tip of the electrode, the greater the number of signals recorded.

3.2.1 Multiple electrodes

To increase the accuracy of spike sorting, and to increase the number of classified single neurons, multiple electrodes can be applied. In many situations, two different neurons generate action potentials having very similar shapes in the recorded waveform, especially in cases where neurons are similar in morphology and have the same distance to the recording electrode. [Lewicki 1998] One approach to solve this problem is to record from multiple electrodes in the same local area. An advantage is that having multiple recordings of the same neuron from different physical locations allows additional information to be considered for accurate spike sorting. This aspect may also reduce the problem of overlapping spikes [Lewicki 1998].
[Gray, Maldonado, Wilson & McNaughton 1995] reported use of tetrodes in cat visual cortex to compare the performance of tetrodes with the best electrode pair and best single electrode, with the best results using tetrodes. Quiroga et al. [2004] has developed a spike sorting software called Wave_Clus, which is capable of performing spike sorting from both single electrodes and polytrodes using both wavelets and principal component analysis (PCA) for feature extraction.

3.2.2 Electrode drift

During the recordings, the electrodes may drift slowly to a new position because of the settlement of neural tissue in response to pressure from the advancement of the electrode. With time, this will result in shape change of the action potentials, due to non-stationarity in the recordings [Vargas-Irwin & Donoghue 2007]. Ideally, this problem should by approached with the same method as with bursting (see section 3.9 page 17), and it must be possible to update the features and templates over time. [Lewicki 1998]

3.3 Basic spike sorting steps

This section will go through some basic spike sorting steps, to introduce the point of departure of the understanding of the fully automatic UBD spike sorting algorithm, considered in this present work. The following are primarily based on [Quiroga et al. 2004] and [Lewicki 1998].

The first and easiest aspect in separating spikes corresponding to different neurons is to use an amplitude discriminator. The use of this classification is usually very fast and relative simple to implement. One problem with this simple approach can be that spikes from different neurons may produce the same peak amplitude, but may differ in shapes.
Many neurons can be successfully characterized by its amplitude as a feature of spike shape, also called the height of the spike. This feature can be measured with a voltage threshold trigger that generates a pulse whenever the measured voltage crosses the threshold. By optimal positioning of the recording electrode, the spikes can be maximally separated from the background activity or noise. [Lewicki 1998]
Figure 3.3 illustrates an example of the quality of spike isolation, with a well isolated neuron and a poorly isolated neuron. In (a), the background spikes or noise have small effect on the quality of the isolation. In (b), two distinct spikes shapes can be seen, and it is not possible to set the threshold so that one spike is isolated.

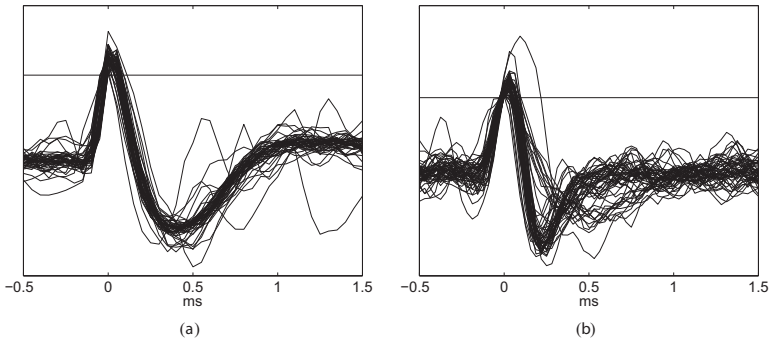


Figure 3.3: (a) An example of a well isolated neuron. Each trace is plotted when the voltage crosses the threshold. (b) An example of a bad isolated neuron, where two distinct spike shapes can be seen. Inspired from [Lewicki 1998]

3.3.1 Detection errors

Often it is not possible to separate the spikes totally from the background noise [Brown et al. 2004]. The voltage threshold determines the trade-off between missed spikes (type II error, false negatives) and false spikes detected (type I error, false positives) [Quiroga et al. 2004]. In figure 3.4, this trade-off is illustrated. The aim is to set the threshold to the desired ratio between type I and II errors. [Lewicki 1998] Spike overlaps can in some cases result in misclassifications. The spike amplitude can vary if there are other firings neurons in the local region. A spike can be missed if the firing of the desired neuron and the background unit lineup.
Another potential detection error is when two background spikes in combination crosses the threshold. [Lewicki 1998]

3.3.2 Other basic spike sorting methods

An example of a relatively straightforward improvement is to use windows discriminators [Lewicki 1998]. The aim here is to assign the spikes that cross one or several windows to the same neuron. Window discriminators are able to be implemented online, but have the disadvantage that it requires manual adjustment of the windows by the expert. This may require readjustment during the sorting process. This fact limits the spike sorting in practice with more than a few channels simultaneously using this method. [Lewicki 1998]
Another drawback with this approach is that spike shapes may overlap and it is very difficult to set up windows that will discriminate correctly in this case. This will introduce a lot of subjectivity in the clustering procedure, which is undesirable. Furthermore it might happen that sparsely firing neurons may be overlooked, especially if the particular input that elicits the firing of the neuron is not present while the windows are adjusted. [Lewicki 1998]

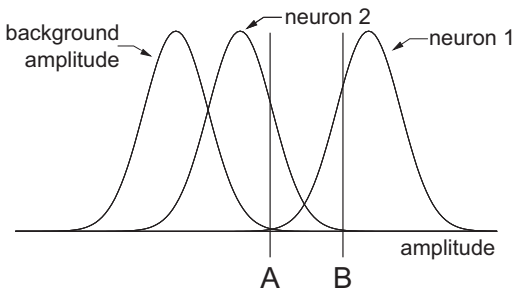


Figure 3.4: Illustrates an example of distributions of amplitudes, both from background noise and from spikes from two neurons. A and B illustrates the trade-off in choosing threshold. A results in many spikes from neuron 1, and B results in many missed spikes. Inspired from [Lewicki 1998]

Another introducing strategy in spike sorting is to select a characteristic spike shape for each cluster and then assign the remaining spikes using template matching. But this method has the same drawbacks as previous mentioned. [Quiroga et al. 2004]

Acquisition systems allow the simultaneous recording of several channels simultaneously [Vargas-Irwin & Donoghue 2007]. The reliability of these data depends on accurately identifying the activity of the individual neurons with spike sorting. When processing large number of channels, supervised methods can be highly time consuming, subjective, and nearly impossible to use in an experiment. It is therefore desirable to develop new methods to deal with recordings from multiple electrodes [Lewicki 1998].

In general spike sorting has four main steps, and each of these steps has great influence on the results. For the same reason, their implementation should be carefully considered.

The following contains selected introductory examples to describe the four major steps in spike sorting, shown in figure 3.5.

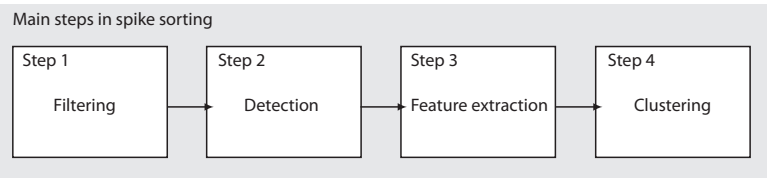


Figure 3.5: Illustrates the four main steps in classical spike sorting methods.

3.3.3 Filtering

The initial step when processing continuously recorded data, that demands spike sorting, is to apply a band pass filter [Quiroga et al. 2004]. This is done to avoid low frequency activity and to better visualize the spikes. The continuously recorded data could e.g. be EMG, intra-cortical recordings (focus in this present work), or intra-fascicular nerve recordings.
In [Quiroga et al. 2004] the continuous data was filtered with a no causal band pass filter between 300 and 3000 Hz. The upper cut-off frequency is to diminish the noisy appearance of the spike shapes. As always with filtering, a

compromise has to be taken. It is desirable to have a narrow filter band to better visualize the spikes, but on the other hand, if the band is too narrow, the filter may hide different features of the spike shapes. An important issue is that the filter should preferably be no causal. Causal filters, which are any recursive or IIR filter, usually produce phase distortions which may really change the spike shapes. [Quiroga et al. 2004] [Quiroga 2009]

3.3.4 Spike detection

Spike detection is the next step in spike sorting after filtering. The following describes general considerations and examples of spike detection from [Quiroga et al. 2004].
In many cases, spikes are detected using an amplitude threshold on the filtered data. Setting the threshold is a compromise between:

- A high threshold may cause spikes to be missed (type 2 error)
- A low threshold may cause false positives due to noise crossing (type 1 error).

In several online spike sorting systems the threshold can be set manually, but in cases with multiple channels, automatic threshold is preferable. In [Pouzat, Mazor & Laurent 2002] the automatic threshold is a multiple of the standard deviation of the signal. This could lead to very high threshold values, especially in cases with high firing rates and large spike amplitudes. Quiroga et al. [2004] proposes an automatic threshold setting, to overcome this limitation, shown in equation 3.1.

$$Th = 5\sigma_n, \text{ where } \sigma = median\left(\frac{|x|}{0.6745}\right) \tag{3.1}$$

where x is the bandpass filtered signal and σ_n is an estimate of the standard deviation of the background noise. Using the median in the estimation of the threshold diminishes the interference of the spikes, because of the assumption that spikes amount to a small fraction of all samples.

When spikes are detected, they have to be stored for clustering, and the first problem is how many data points to store. This depends on the sampling frequency and ideally the whole spike shape should be stored; i.e. about 2 ms of data [Quiroga et al. 2004]. With a sampling frequency of 30 KHz, this corresponds to 60 samples.

3.3.5 Feature extraction

The next step for spike sorting is to extract features of the spike shapes. This step can give a dimensionality reduction, reducing from a space of dimension m to a low dimensional space of fewer features. Ideally it is to extract those features that most optimal separates the different clusters of spikes and remove all the dimensions dominated by noise. This step makes the process more computational efficient and it is mandatory for some clustering algorithms. Furthermore, eliminating inputs contaminated by noise can certainly improve clustering results. [Quiroga et al. 2004]

The first idea for feature extraction could be to take basic characteristics of the spikes, such as their peak, amplitude, width and energy. However, it has been shown that such features are not always optimal for differentiating spike shapes. [Quiroga et al. 2004]
As in figure 3.3, the two spikes have almost the same amplitude, but are different in shape. The aim is here to characterize the shape, and use this information to classify each spike. Figure 3.6 shows an example of classic features to express the difference between the two clusters of spikes, based on minimum/maximum spike amplitude in (a), or spike width/height in (b). (a) shows that the spikes has almost the same maximum amplitudes, but differentiates in two regions with minimum amplitudes. The large cluster near the origin also reflects noise and background spiking neurons. (b) shows clustering with spike height and width, where the two clusters are marked in boxes. [Lewicki 1998]
One of the most used method for feature extraction is principal component analysis (PCA), and to take the first 2 or 3 principal components that contains more than 80% of the energy of the signal [Horton et al. 2007] [Adamos, Kosmidis & Theophilidis 2008] [Quiroga et al. 2004]. However, PCA selects the directions of max variance of the data, which may not be the directions of best separation. In some cases, it may be that the information for separating

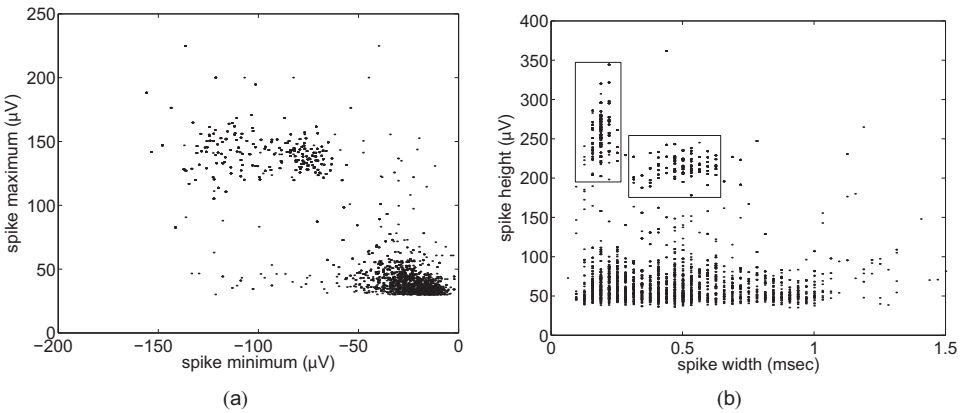


Figure 3.6: Example of different features that expresses spike shape difference. Every dot represents a spike. (a) Maximum versus minimum spike amplitude. (b) Spike width versus spike height. Inspired from [Lewicki 1998]

the clusters is represented in one or a combination of principal components with low eigenvalues. [Quiroga et al. 2004]

It has also been proposed to use wavelets for feature extraction. [Hulata, Segev & Ben Jacob 2002] [Quiroga et al. 2004] The wavelet transform provides a time-frequency decomposition of the signal with optimal resolution in time and frequency domains [Mallat 1989]. One of the advantages of using wavelets for feature extraction is that very localized shape features can be detected because wavelet coefficients are localized in time. [Quiroga et al. 2004]

3.3.6 Clustering

The final step of spike sorting is to group spikes with identical features into clusters, corresponding to the different neurons.
The previous step was to reveal clusters that are relevant to classifying spike shapes. This step, cluster analysis, is the finding of clusters in multidimensional data sets and classifying data based on these clusters.
A basic assumption in clustering is that the data results from several independent classes, each of which can be described by a model. This assumption fits spike sorting, because each action potential arises from a different neuron. The first task in clustering is to describe the cluster location and the variability of the data around that location. [Lewicki 1998]

One very intuitive method is to delimit clusters manually by drawing polygons in 2-dimensional projections of the spike features, also seen in figure 3.6. [Gray et al. 1995] This method can be a very time consuming task and furthermore manual clustering may introduce errors because of limited dimensionality of the cluster cutting space and because of human biases. In many cases clusters overlap and the manual setting of a boundary has the great disadvantage of being very subjective. [Harris et al. 2000]

[Lewicki 1994] have proposed a more refined solution that uses Bayesian classification. This approach assume a Gaussian distribution of the clusters, and is based on the assumption that in any given cluster the spike variability is determined only by additive and Gaussian stationary background noise.
This assumption may be valid in some conditions, but it has been argued by [Fee et al. 1996*b*] that the background noise cannot be represented as a stationary Gaussian random process. There are several aspects that can lead to clusters with non-Gaussian shapes.

1. Electrode drifts during the recordings

2. Variation in the spike shape due to bursting
3. Presence of overlapping spikes
4. Correlations between spikes and local field potentials
5. Non-stationary background noise

[Fee, Mitra & Kleinfeld 1996*a*] has developed a hierarchical clustering algorithm to overcome the assumption of Gaussian clusters, which first sorts the data into an overly large number of clusters and then merges these clusters according to spike shape similarities and statistics.

The use of clustering algorithms based on nearest neighborŠs interactions is another approach to avoid the assumption of Gaussian distributions. Quiroga et al. [2004] uses this principle, which basically group together contiguous set of points given that the local density is larger than a certain value. The method is more specifically called super-paramagnetic clustering (SPC), and has been used for spike sorting.
SPC is a stochastic algorithm with no assumption of any particular distribution of the data. SPC groups the spikes into clusters as a function of a single parameter, which is the temperature. For low temperatures, all the data are grouped into a single cluster and for high temperatures the data are split into many clusters. The optimal case is the middle range of temperatures corresponding to the super-paramagnetic regime, where the data are split into relatively large size clusters. [Quiroga et al. 2004]

Another approach is the nearest-neighbor or k-means clustering. Here the cluster locations are defines as the mean of the data within that cluster. A spike is classified to whichever cluster has the closest mean using Euclidean distance. Hereby a set of implicit decision boundaries are defined, that separates each cluster. Figure 3.7 shows an example of these boundaries for the dataset also shown in section 3.6.1. [Lewicki 1998] These relatively simple approaches are

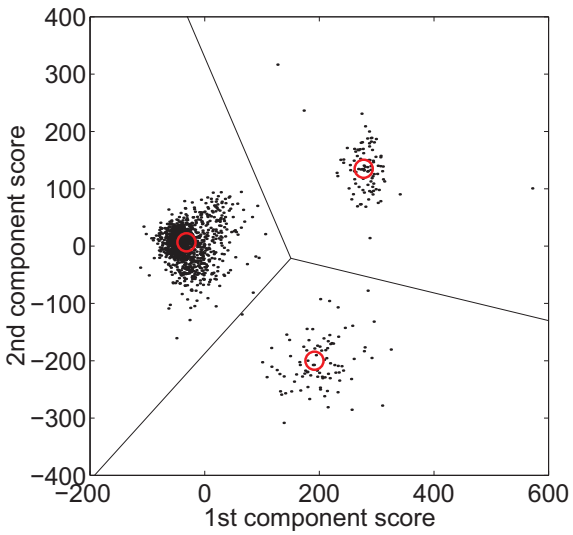


Figure 3.7: Illustrates the decision boundaries for nearest-neighbor clustering. Inspired from [Lewicki 1998]

adequate when the clusters are well separated, but fail when the clusters overlap.

Clustering in higher dimension and template matching are examples of more refined clustering methods. It can be convenient for display purposes only to use 2 features, but it may be desirable to extend the cluster space to higher dimensions. In template matching the waveform are the class means and correspond to the average spike waveform for each class. The idea is to obtain a more accurate classification, by adding more dimensions to the clustering. The aim

is to have clustering procedures where the spike templates are chosen automatically, and in case of Euclidean metric is used to measure the distance to the template, then this corresponds to nearest-neighbor clustering. Template matching with Bayesian integration classifies spikes with the advantages that the classification takes into account the variation around the mean spike shape. [Lewicki 1998]
Figure 3.8 shows an example of three waveforms, or spike templates that defines the cluster means. [Lewicki 1998]

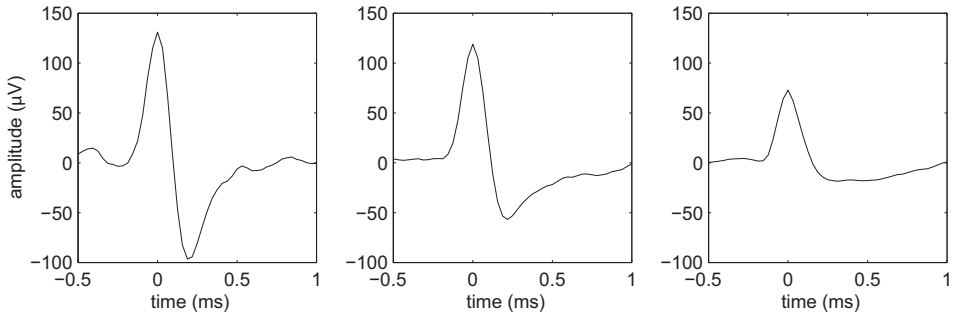


Figure 3.8: An example of three waveforms, or spike templates, that defines the cluster means. Inspired from [Lewicki 1998]

3.4 Overlapping spikes

A lot of the traditional spike sorting algorithms described does not deal with overlapping spikes [Lewicki 1998]. Overlapping spikes occurs if two close-by neurons fire in synchrony or with a small time delay. Overlapping spikes can be i.e. a spike shape generated by the sum of the spikes from two neurons. In a simple case when double peaks appear (small time delay is present), it is relatively easy to identify overlapping spikes. Overlapping spikes can look like the firing of a third neuron when there is no time delay. This is a much more difficult situation to solve. According to [Quiroga et al. 2004], overlapping spikes is one of the most challenging issues in spike sorting.

If two spikes are sufficiently separated in time, it is possible that the traditional spike sorting algorithms can classify the spikes correctly. Serious problems will though appear if two or more spikes fires simultaneously. Spikes sorting approaches with cluster cutting (section 3.3.6) or Bayesian approaches (section 4.0.1) to classification, it may be possible to identify some overlaps as outliers. [Lewicki 1998]
A simple approach to deal with overlaps is to subtract a spike from the waveform after classification. This is done with the aim of improving the classification of subsequent spikes, and requires a template or model of the spike shape. This method though has some drawbacks when spikes come to close together, and it may introduce unwanted noise in the waveform if the spike model is not accurate. Another aspect with subtraction-based approaches is that the spike occurrence time may not be accurately estimated, which also introduces spurious spike shapes, because of artifacts in the residual waveform due to misalignment. [Lewicki 1998]
In Lewicki [1998] different methods are presented to deal with overlapping spikes. Among other an approach based on neural networks, a method that compares the overlap with all possible combinations of two spike models, and a overlap decomposition algorithm using k-dimensional search trees.

3.5 Bursting cells

The definition of a burst is the firing of a fast sequence of spikes by one neuron. Bursts can have a variable number of spikes, and they may appear as concatenated, and in some cases with decreasing amplitude. It is possible to identify bursts visually, and with the help of inter-spike-interval histograms. [Quiroga et al. 2004]

In relation to spike sorting, it is critical that spikes in a burst are not taken as separate clusters, due to different amplitudes of the individual spikes. A lot of the traditional spike sorting methods assumes that the spike shapes are stationary, meaning that their shape do not change with time. Lewicki [1998] states however, that many neurons generate action potentials that can have varying shape. Figure 3.9 shows an example of three cases of bursting neurons. (a) in figure 3.9 shows a scenario where the action potentials becomes progressively smaller. This may result in an elongated cluster in clustering procedures, but for example the technique using multivariate Gaussian clustering can still classify bursts correctly, given that the attenuation is not too large, and that individual spikes can be detected. [Lewicki 1998]
Most traditional method fails though, when several neurons in a local group of neurons burst simultaneously. An example of such a burst are shown in (b) in figure 3.9.

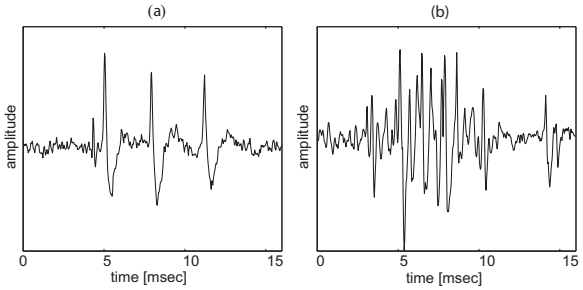


Figure 3.9: (a) Neuron burst where the action potential change shape progressively. (b) Complex burst from many neurons, with no individual visual spikes. Inspired from [Lewicki 1998]

3.6 Methods for spike sorting

This section presents different basic methods used in spike sorting, followed by a list of candidate spike sorting methods for comparison with the method in focus in this present work (the UBD method).

3.6.1 Principal component analysis

The principle behind principal component analysis (PCA) is to compute a sorted set of orthogonal basis vectors that contains information about the directions in the data of largest variation. The input is the spike data from the recorded waveform.
Figure 3.10 shows a sample set of spikes centeret in the spike maximum, which are used in the following description of PCA. [Lewicki 1998] When using PCA for spike sorting, the principal components are scaled and added together to represent the given spike. The principal component vectors are found by computing the eigen-vectors of the covariance matrix of the waveform data.
Another parameter found using PCA is the scale factor for each component, also called the score. The *i*th score is calculated with equation 3.2. [Lewicki 1998]

$$s_i = \sum_t c_i(t)x(t) \tag{3.2}$$

where $x(t)$ is the spike data and $c_i(t)$ is the *i*th principal component. The principal components are ordered in terms of how much variance they describe from the input signal. Increasing the number of components added together, increases the amount of variance accounted for, or in other words, adding additional components yields progressively smaller corrections until the spike is described exactly. In figure 3.11 the first three principal components, computed from the data in figure 3.10, are showed. [Lewicki 1998] Especially the first component has a spike-like shape, and is the direction of largest variation in the data. The second and third component is more and more contamined by noise.
To interpret the results from PCA and determine the number of components used for classification of spikes, it is

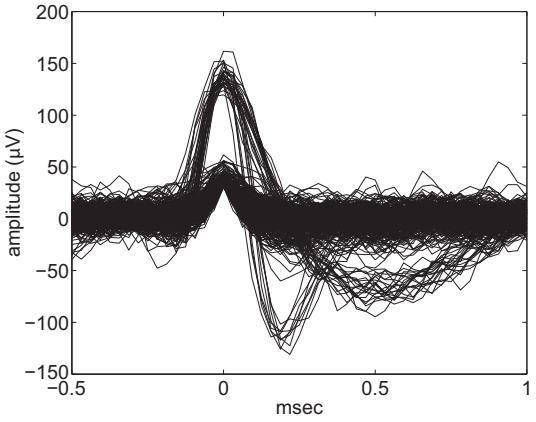


Figure 3.10: Example of a sample set of spikes used in the description of PCA. Inspired from [Lewicki 1998]

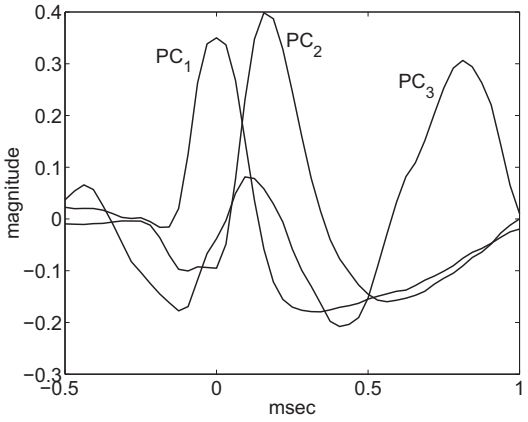


Figure 3.11: The first three principal components, describing the spike data from figure 3.10. [Lewicki 1998]

helpful to plot the standard deviation of the scores in the direction of each component, as shown in figure 3.12. [Lewicki 1998]
As an example, it is seen that the first three components accounts for 76% of the variance in the data. If the background noise level is determined, it is possible to choose the number of components that are significantly above the noise.
If the first two components are chosen, they can serve as features for classifying different spikes. Figure 3.13 shows an example of a scatter plot of the first two components, and a relatively clear separation of the two spike shapes are seen. [Lewicki 1998]

3.6.2 Independent component analysis

A technique that is applicable for multichannel spike sorting is the independent component analysis (ICA). ICA approaches the blind source separation problem, which is suitable in spike sorting. The basic concept is to unmix N independent signals that have been linearly mixed onto N channels with priorily unknown mixing weights. This is illustrated in figure 3.14. The method assumes that the unknown sources are independent, and the signal separation is performed sample by sample, so that no prior information about spike shape are necessary.
There might be different limitations for this method, including the assumption about linearly mixing of the sources, and that the number of channels must equal the number of sources. [Lewicki 1998]

3.6.3 Overview of spike sorting methods

The following tries to give examples of different spike sorting methods, both supervised and unsupervised, to make a selection of a comparable method for the UBD method possible.

Supervised spike sorting methods

A variety of supervised spike sorting methods are described in the literature. In this section, a selection of spike sorting methods is listed.

Delescluse & Pouzat [2005] have developed a spike sorting method using inter-spike intervals information. The algorithm is based on a Markov chain Monte Carlo algorithm, which is capable of estimating and use the firing statistics and spike amplitude dynamics of the neurons. Hulata et al. [2002] presents a spike sorting method based on wavelet packets and ShannonŠs mutual information. The use of the wavelet packets decomposition to analyze neural spikes and extract their main features, is according to Hulata et al. [2002] both efficient in separating spikes from

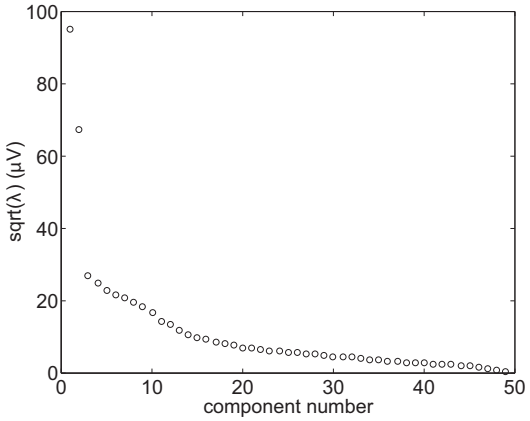


Figure 3.12: The standard deviation of the scores in the direction of each component. Used to determine the number of components used for classification. [Lewicki 1998]

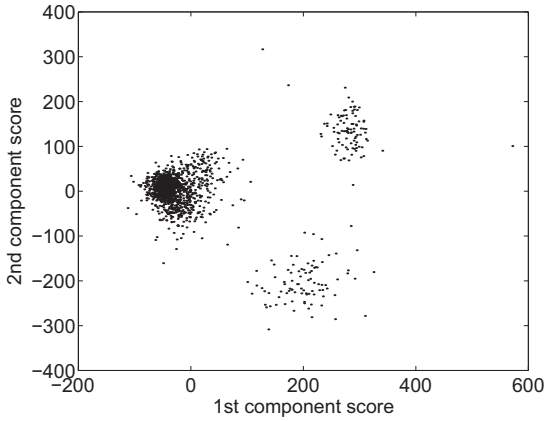


Figure 3.13: A scatter plot of the features using the first two principal components. A relatively clear seperation of the spike shapes are seen. Inspired from [Lewicki 1998]

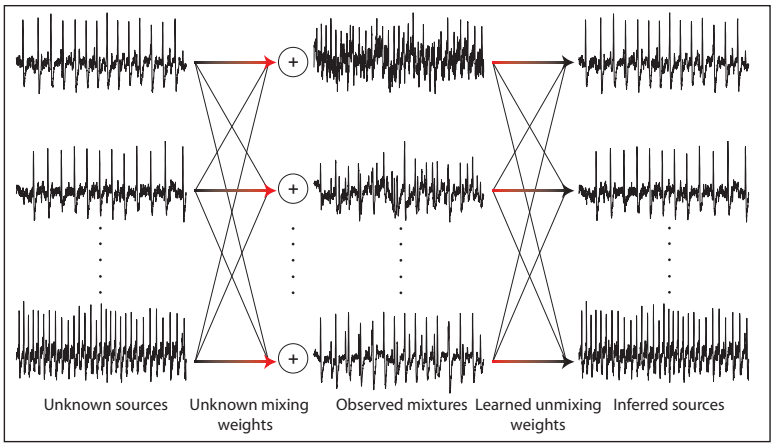


Figure 3.14: Illustration of the basic concept in using ICA for spike sorting. *N* unknown sources are mixed linearly with unknown mixing weights, to form *N* observed mixtures. ICA then finds the unmixing weights to transform the mixtures into independent signals. Inspired from [Lewicki 1998]

noise and sorting overlapping spikes. Song & Wang [2006] presents a spike sorting framework using nonparametric detection and incremental clustering. The method detects spikes based on a nonparametric shape distribution, and spike clustering is performed using second-order statistics covariance matrix. Horton et al. [2007] have developed a spike sorting method based on machine learning algorithms. This method, together with Adamos et al. [2008], uses principal component analysis for feature extraction. Herbst et al. [2008] presents a spike sorting method using hidden Markov models. The method blindly performs source separation in a combination of both spike detection and classification. The first part of the algorithm learns spike templates, firing probabilities, and Gaussian noise parameters. The second part performs spike sorting based on the learned information.

Unsupervised spike sorting methods

The following shows examples of unsupervised spike sorting methods from the literature.

Aksenova et al. [2003] presents an unsupervised method for sorting neuronal waveforms, based on inverse methods of nonlinear oscillation theory. The method is an unsupervised iteration-learning algorithm that estimates the number of classes and their centers according to the distance between spike trajectories in phase space. Vargas-Irwin & Donoghue [2007] presents an automated spike sorting method using density grid contour clustering and subtractive waveform decomposition. The method uses, besides density grid contour clustering, principal component analysis (described in section 3.6.1) and template matching using subtractive waveform decomposition. The spike sorting algorithm deals with the influence of noise, spurious threshold crossing and spike overlapping. Zhang et al. [2004] have developed a spike sorting method based on automatic template reconstruction with a partial solution to the overlapping problem. The method also includes principal component analysis, subtractive clustering techniques, and template matching in the spike sorting process, which also tries to deal with the spike overlapping problem. Quiroga et al. [2004] have developed a spike sorting method named Wave_Clus, based on unsupervised spike detection and sorting using wavelets or principal component analysis for feature extraction. The method uses superparamagnetic clustering for clustering purposes, for automatic classification of spikes without assumptions such as low variance or Gaussian distributions. Wave_Clus deals with partially overlapping spikes, with highly localized features such as wavelet coefficients. Madany, Sharp, Menne, Hofmann & Martinetz [2005] presents an unsupervised spike sorting algorithm using ICA (described in section 3.6.2). Atiya [1992] uses the Isodata clustering algorithm to estimate typical spike shapes in the spike sorting process. To deal with overlapping spikes, the method compares all possible combinations of the templates to find the combination with the highest likelihood, with the drawback of being very computationally expensive. Pouzat et al. [2002] developed a spike sorting procedure for the classification and validation of extracellular data based on a probabilistic model of data generation. The method uses the same spike classification method as Atiya [1992]. Lewicki [1994] applied Bayesian probability theory to define a probabilistic model of the waveform and to quantify the probability of both the form and the number of spike shapes. This method was also designed to deal with overlapping spikes. Bar-Hillel et al. [2006] have developed a spike sorting method, based on Bayesian clustering of non-stationary data. This method is fully automatic, with a clustering procedure in a Bayesian framework, with the source neurons modeled as a non-stationary mixture of Gaussians. Wood & Black [2008] suggests similar approaches. Shoham et al. [2003] suggests handling the non-stationary clusters by modeling clusters using a t-distribution in a automatic spike sorting method. This method, though, does not partition the data into time frames, which may complicate the classification of similar spikes in distant time frames.

3.7 Selection of spike sorting method for comparison

Based on the description of spike sorting methods in section 3.6.3, the method chosen for comparison in this present study is the Wave_Clus method by Quiroga et al. [2004]. This method is, like the UBD method, fully automatic and unsupervised through all phases of the spike sorting process. Furthermore, it has the option of using both wavelets and PCA for feature extraction, and both methods are used in the comparison. A more detailed description of the method can be found in section 6 page 31.

Probabilistic spike sorting

4

The method in focus in this present work, the UBD method, uses a Bayesian framework for conducting spike sorting. This yields a probabilistic foundation in the classification of neuronal waveforms, which substitutes the need of a human operator, and makes the spike sorting algorithm unsupervised, and fully automatic. Only a few already developed spike sorting methods using probabilistic Bayesian approach have been documented in the literature (see section 3.6.3 page 18). Lewicki [1994], Lewicki [1998], and Bar-Hillel et al. [2006] have presented spike sorting methods with a Bayesian probabilistic foundation, and will in the following section give a basic understanding of Bayesian clustering and classification.

4.0.1 Bayesian clustering and classification

Clustering related to spike sorting can also be seen as a model of the statistical distribution of the data. Such a method primarily has the advantage of quantifying the certainty with which spikes are classified. [Lewicki 1998]
One method, or probabilistic approach to clustering is presented by [Lewicki 1998]. It models each cluster with a multivariate Gaussian, which is centered on the cluster. The likelihood of the data given a particular class c_k is given by equation 4.1. [Lewicki 1998]

$$p(x|c_k, \mu_k, \Sigma_k) \tag{4.1}$$

where x is the spike data vector, μ_k is the mean, and Σ_k is the covariance matrix for class c_k. The clustering model assumes that the data are selected independently from the underlying classes. The marginal likelihood, which is not conditioned on the classes, is computed by summing over the likelihood of the K classes, shown in equation 4.2. [Lewicki 1998]

$$p(x|\theta_{1:K}) = \sum_{k=1}^{K} p(x|c_k, \theta_k) p(c_k) \tag{4.2}$$

where $\theta_{1:K}$ defines the parameters for all of the classes, $\theta_{1:K} = \{\mu_1, \Sigma_1, ..., \mu_K, \Sigma_K\}$. $p(c_k)$ is the prior probability of the kth class, with the total probability equals $\Sigma_k p(c_k) = 1$.

Classification is performed by calculating the probability that a recorded sample belongs to each of the classes, which can be described with equation 4.3 using Bayes rule. [Lewicki 1998]

$$p(c_k|x, \theta_{1:K}) = \frac{p(x|c_k, \theta_k) p(c_k)}{\Sigma_k p(x|c_k, \theta_k) p(c_k)} \tag{4.3}$$

In this expression, the Bayesian decision boundaries for the model are defined. To define each cluster boundary, the confidence levels can be computed because each cluster membership is probabilistic. This basis gives a classification with a minimum of misclassifications. [Lewicki 1998]
To optimize the class parameters for each class, the likelihood of the data are maximized in equation 4.4.

$$p(x_{1:N}|\theta_{1:K}) = \prod_{n=1}^{N} p(x_n|c_k, \theta_{1:K}) \tag{4.4}$$

[Lewicki 1998] refers to a free software package, AutoClass, which uses the Bayesian methods described above to determine the means, covariance matrices, and class probabilities.
Figure 4.1 shows an example of a Gaussian model for each cluster, with a three standard deviation error contour. In the Bayesian classification, an advantage is that the Bayesian framework offers a quantification of the certainty of the classification. This is an important property, when decisions about the isolation of spikes in different clusters are to be made. The probability of a spike being classified to a particular cluster, is given by equation 4.3, which generates a probability for each cluster.
To estimate how well a particular class is separated from others, it is possible to consider the distribution of the probabilities for a class. Hereby, the quality of the isolation can be monitored. Figure 4.2 shows a histogram of the

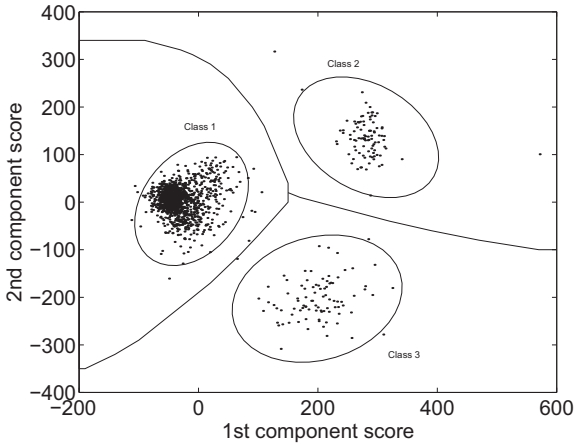


Figure 4.1: An example of Gaussian clustering for spike sorting. The ellipses show the three standard deviation error contours of the clusters, and the lines show the Bayesian decision boundaries that seperates the large clusters. Inspired by [Lewicki 1998]

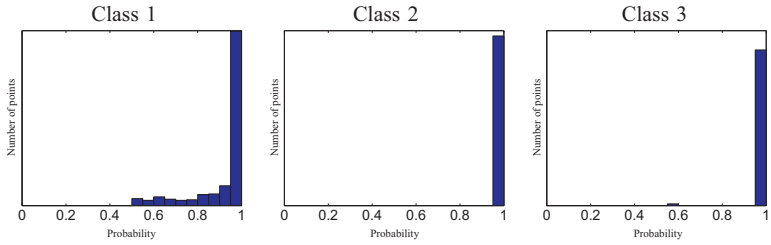


Figure 4.2: A histogram of the distribution of probabilities in the three classes. Class 1 shows diverging certainty, with not all points having a probability equal 1. In class 2 and 3, nearly all points have probabilities equal 1. Inspired by [Lewicki 1998]

distribution of the probabilities in the three classes also shown in figure 4.1. Class 1 shows diverging certainty, with not all points having a probability equal 1. In class 2 and 3, nearly all points have probabilities equal 1, which indicates that all points are assigned to their respective clusters with very high certainty. A decrease in isolation quality can indicate background noise or electrode drift. [Lewicki 1998]

Unsupervised Bayesian decomposition - UBD

5

Spike sorting is usually done with semi-automatic procedures. These procedures often involve interaction from an expert operator, doctor or scientist, and the idea in this present work is that a fully automatic method will be more optimal.

The method in focus in this present work is a spike sorting method based on unsupervised Bayesian decomposition (UBD) using TABU search, developed by Ge et al. [2009]. Originally, this method was developed for the decomposition of multi-unit EMG recordings, but in this present work, the aim is to test the method with simulated intra-cortical recordings.

The problem of spike sorting is solved with a Bayesian statistical model and a maximum a posterior estimator (MAP), which makes the spike sorting method fully automatic. The MAP estimation includes several parameters as prior information integrated in the Bayesian framework, such as physiological constraints (discharge pattern regularity and neuronal refractory period) and the likelihood of the reconstructed signal.
The TABU search is included in the algorithm to deal with overlapping spikes, which is a NP-hard optimization problem. This is done to avoid exhaustive analysis where all possible overlaps are tested.

Both intra-cortical recordings and intramuscular EMG signals are the sum of series of action potentials discharged by the neurons detected by the recordings electrodes. The multi-unit recorded signal can be decomposed or spike sorted into constituent action potentials or spikes, to extract the discharge patterns of the neurons.

This method performs spike sorting like classical approaches when it comes to spike identification from the interference signal with a threshold. One complicated issue in this spike sorting is the separation of spikes that overlap in time. This problem can be characterized as an NP-hard problem, because of the complete search space of overlapped spikes. [Ge et al. 2009]
The following describes an overview of the method based on unsupervised Bayesian decomposition in focus in this report.

5.1 Description of the model

This section describes the theory behind the UBD method. The description should be valid for the neural recordings relevant in this project (simulated and human intra-cortical recordings).

5.1.1 Forward model

The neural signal can be mathematically described with the model shown in equation 5.1. The neural signal contains the contributions from I neurons or sources.[Ge et al. 2009]

$$z = \sum_{i=1}^{I} \mathbf{h}_i * \mathbf{s}_i + \varepsilon \tag{5.1}$$

where z is the recorded neural signal (i.e. intra-cortical recordings) with length N. This signal is modelled as a mixture of convolutions of the impulsive trains $\mathbf{s}_i, i = 1,...,I$ and the linear filters $\mathbf{h}_i, i = 1,...,I$. The impulsive trains can be interpreted as the discharge patterns for the neurons, and the linear filters can be interpreted as the spikes.
The mixture of sources in equation 5.1 result in the following statistical assumptions: [Ge et al. 2009]

1. All impulsive trains (discharge patterns) $\mathbf{s}_i = 1_{\mathbf{x}_i}, i = 1,...,I$ is modeled as an independent process with uniform amplitudes. $\mathbf{x}_i$ is a vector that contains the coordinates of all impulses for each source. $n_i = length(\mathbf{x}_i)$ is the number of discharges.

2. The discharge patterns $(1_{\mathbf{x}_i})_i$ for each neuron are assumed mutually independent.

3. The spike shapes $(\mathbf{h}_i, i = 1, ..., I)$ change slowly during the neural recording (variation in shape occurs not faster than time intervals of seconds).

4. The recorded neural signal z is corrupted by additive white Gaussian noise with unknown variance σ_ε^2.

One vital assumption in this UBD method, is that the data generating process that generates z obeys the Gaussian law shown in equation 5.2, given the source parameters $(\mathbf{x}_i, \mathbf{h}_i)_i, \sigma_\varepsilon^2$. [Ge et al. 2009]

$$P(\mathbf{z}|(\mathbf{x}_i, \mathbf{h}_i)_i, \sigma_\varepsilon^2) = \left(\frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}}\right)^N \exp\left(-\frac{||\mathbf{z} - \Sigma_i 1_{\mathbf{x}_i} * \mathbf{h}_i||^2}{2\sigma_\varepsilon^2}\right) \tag{5.2}$$

The following assumptions apply to the discharge patterns of each neuron: [Ge et al. 2009]

1. The inter-spike interval (ISI) $T_{ij} = \mathbf{x}_{i,j+1} - \mathbf{x}_{i,j}$ between two consecutive spikes for a given source (neuron) $\mathbf{s}_i$ is larger than a threshold value T_R. T_R denotes the neuronal refractory period, which is a physiological constraint on the ISI.

2. The ISI $(T_{ij} - T_R)$ follows a Gaussian shaped distribution $(T_{ij} - T_R) \sim N(m_i, \sigma_i^2)$.

3. The ISI variability is smaller than a threshold $\frac{\sigma_i}{m_i} < \mathrm{Th}_{\sigma_i}$. This constraint expresses the regularity in the discharge patterns and the threshold Th_{σ_i} controls the variability of the ISI. This constraint may be regulated dependent on the type of neural signal.

According to (3) in the above enumeration, the ISI follows a truncated Gaussian distribution expressed in equation 5.3. [Ge et al. 2009]

$$\begin{cases} P(T_{ij}) & = 0 & , T_{ij} < T_R \\ P(T_{ij} - T_R) & \propto g(m_i, \sigma_i^2) & , T_{ij} \geq T_R \end{cases} \tag{5.3}$$

This approximated Gaussian distributed ISI is experimentally observed in the simulated test signals used in this study (see figure 9.3 page 58).
On a discrete grid, the ISI probability described in equation 5.3 is well-defined up to a normalization factor $(\sum_0^\infty \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-(k-m_i)^2/2\sigma_i^2} \to 1$, when $m_i \to \infty$ and $\sigma_i/m_i < \mathrm{Th}_{\sigma_i})$. [Ge et al. 2009]

The probability of $\mathbf{x}_i$ given the parameters for the Gaussian shaped distribution for ISI, also called the source generating process, is shown in equation 5.4. [Ge et al. 2009]

$$\begin{aligned} P(\mathbf{x}_i|m_i, \sigma_i^2) = \frac{1}{4}\mathrm{erfc}\left(\frac{x_{i,1} - T_R - m_i}{\sqrt{2\sigma_i}}\right)\mathrm{erfc}\left(\frac{N - x_{i,n_i} - T_R - m_i}{\sqrt{2\sigma_i}}\right) \\ (2\pi\sigma_i^2)^{-(n_i-1)/2}\exp\left\{-\frac{1}{2\sigma_i^2}\sum_{j=1}^{n_i-1}(\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j} - m_i - T_R)^2\right\} \end{aligned} \tag{5.4}$$

valid for a given configuration of a neuron or source $\mathbf{s}_i = 1_{\mathbf{x}_i}$. The first two terms in equation 5.4 $(\frac{1}{4}\mathrm{erfc}(\bullet)\mathrm{erfc}(\bullet))$ evaluate the first impulse ISI with probability $P(T_{i,1} > x_{i,1})$ and the $(n_i + 1)^{th}$ impulse ISI with probability $P(T_{i,n_i+1} > N - x_{i,n_i})$ respectively.
The expression in equation 5.4 can be simplified to equation 5.5. [Ge et al. 2009]

$$P(\mathbf{x}_i|m_i, \sigma_i^2) \approx (2\pi\sigma_i^2)^{-(n_i-1)/2}\exp\left\{-\frac{1}{2\sigma_i^2}\sum_{j=1}^{n_i-1}(\mathbf{x}_{i,j+1} - \mathbf{x}_{i,j} - m_i - T_R)^2\right\} \tag{5.5}$$

Because of the independence of all discharge patterns $(\mathbf{x}_i)_i$, the prior law for the discharge patterns $P((\mathbf{x}_i)_i)$ can be expressed as in equation 5.6.

$$P((\mathbf{x}_i)_i|(m_i, \sigma_i^2)_i) = \prod_{i=1}^{I} P(\mathbf{x}_i|m_i, \sigma_i^2) \tag{5.6}$$

5.1.2 Posterior probability law

In this spike sorting method, based on a Bayesian estimation framework, a posterior distribution for the unknown variables $\left\{(\mathbf{x}_i, m_i, \phi_i^2, \mathbf{h}_i)_i, \sigma_\varepsilon^2 | \mathbf{z}\right\}$ is established. Equation 5.7 express the probability of all these variables based on the recorded neural signal. [Ge et al. 2009]

$$P((\mathbf{x}_i, m_i, \phi_i^2, \mathbf{h}_i)_i, \sigma_\varepsilon^2 | \mathbf{z}) \propto P(\mathbf{z} | (\mathbf{x}_i, \mathbf{h}_i)_i, \sigma_\varepsilon^2) \prod_{i=1}^{I} (P(\mathbf{x}_i | m_i, \sigma_i^2) P(m_i) P(\sigma_i^2) P(\mathbf{h}_i)) P(\sigma_\varepsilon^2) \tag{5.7}$$

Equation 5.7 is the core of the spike sorting algorithm. The spike sorting task is conducted by maximizing equation 5.7 w.r.t. the discharge patterns $(\mathbf{x}_i)_i$, the statistics of the discharge patterns $(m_i, \sigma_i^2)_i$, the spike shapes $(\mathbf{h}_i)_i$, and the background noise variance σ_ε^2. The two first terms after '$\propto$' in equation 5.7 are computed from equation 5.2 and 5.4. The remaining four terms are expressed as conjugate priors with the non-informative hyper-parameters $(\alpha_i, \beta_i)_{i=1,\ldots,I}, (\alpha_s, \beta_s, \mu_0, \sigma_0^2, \sigma_\mathbf{h}^2)$ in equation 5.8 and 5.9. These probabilities contributed with prior information about discharge rate and spike shape to the spike sorting algorithm. [Ge et al. 2009]

$$\text{Discharge rate:} \qquad P(m_i) \sim N(\mu_0, \sigma_0^2) \qquad P(\sigma_i^2) \sim \text{IG}(\alpha_i, \beta_i) \tag{5.8}$$

$$\text{Spike shape:} \qquad P(\mathbf{h}_i) \sim N(\mathbf{h}_i^{(0)}, \sigma_\mathbf{h}^2) \qquad P(\sigma_\varepsilon^2) \sim \text{IG}(\alpha_s, \beta_s) \tag{5.9}$$

where IG denotes the inverse Gaussian distribution.

5.2 Description of the spike sorting algorithm

Description of the two-phase spike sorting algorithm based on a maximum a posterior estimator (MAP) which is applied to the forward model described in the above section 5.1.

5.2.1 Maximization algorithm

The key equation in the spike sorting algorithm (the joint posterior distribution in equation 5.7) is maximized w.r.t. the unknown parameters shown in equation 5.10.

$$\widehat{\Theta} = \arg\max P(\Theta | \mathbf{z}) \tag{5.10}$$

where $\Theta = \{(\mathbf{x}_i, \mathbf{h}_i, m_i, \sigma_i^2)_i, \sigma_\varepsilon^2\}$.

The main structure of this spike sorting method is divided into a preprocessing phase and a complete spike separation phase. The spike separation phase iteratively maximizes equation 5.7.
In the case where $(\mathbf{x}_i)_i$ are fixed, the remaining parameters $\{(\mathbf{h}_i, m_i, \sigma_i^2)_i, \sigma_\varepsilon^2\}$ can be estimated and leads to a closed form solution. But the most significant problem in spike sorting is exactly the determination of the discharge pattern $(\mathbf{x}_i)_i$, which cannot be solved by exhaustive exploration. The solution is to determine the spike shapes and firing patterns that maximizes the posterior distribution in equation 5.7. The TABU search algorithm is applied to deal with the maximization w.r.t the discharge pattern $(\mathbf{x}_i)_i$. [Ge et al. 2009]

5.2.2 Overview

Figure 5.1 provides an overview of the UBD spike sorting method. The first phase of the two-phase spike sorting method is the preprocessing part. The recorded signal is segmented and representatives of the detected spikes $(\mathbf{h}_i^{(0)})_i$ are extracted to initialize spike shapes. This is done probabilistically as shown in equation 5.9. The aim for this phase in the spike sorting process is to indentify active segments, and at least isolate one spike for each active neuron. The preprocessing phase is implemented as a classical approach for spike detection, including band-pass filtering and amplitude thresholding [Lewicki 1998]. The level of threshold is set proportional to the background noise variance estimate $\widehat{\sigma}_\varepsilon$.

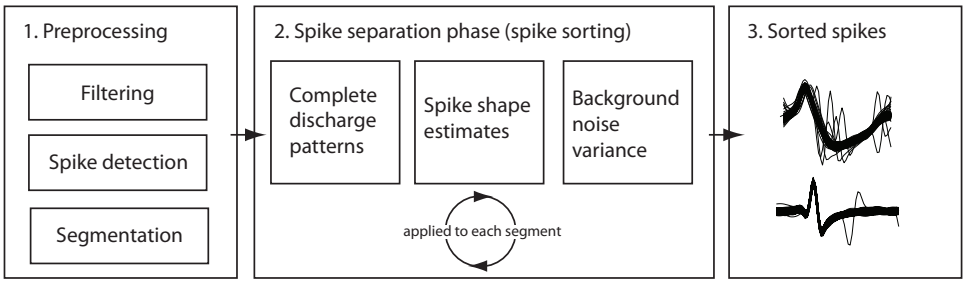


Figure 5.1: Overview of the two-phase UBD spike sorting algorithm.

When an isolated spike or overlapping spikes are detected, an active segment $(\{\text{Seg}_k\})$ is assigned. In the first phase of the preprocessing (segmentation phase), false positives (type I error) are preferred rather than false negatives (type II error). This is because the inclusion of segments Seg_k of pure background noise gives a null solution $\mathbf{x}_i = \phi$, that still belongs to the solution space of the MAP estimation, which is the second phase of the preprocessing. [Ge et al. 2009]

The maximization-decomposition phase is applied after segmentation in the second phase of the spike sorting algorithm. This is done serially to each segment to compute the following:

1. Discharge patterns $(\mathbf{x}_i)_i$ for all neurons
2. Spike shapes estimates $(\mathbf{h}_i)_i$ for all neurons
3. Background noise variance σ_ε^2, with a MAP criterion

One key feature for this spike sorting method is the fully automatic mode of operation. For each segment, the joint posterior distribution in equation 5.7 is maximized over the complete search space of the discharge patterns in the automatic algorithm. Spike overlapping is solved with an algorithm, the TABU search. [Ge et al. 2009]
The TABU search is originally designed for other problems in operational research, but can be adapted to solve the combinatorial problem of spike overlapping.
The MAP optimization steps executed in the second phase can be listed as follows: [Ge et al. 2009]

1. Initialize spike shapes $(\mathbf{h}_i^{(0)})_i$ using the segmentation results from the first phase
2. Initialize discharge rate statistics $(m_i, \sigma_i^2)_i$ by their conjugate prior laws, and the noise parameter σ_ε by the estimate $\widehat{\sigma_\varepsilon}$.
3. Set the discharge patterns $\mathbf{x}_i = \phi(1_{\mathbf{x}_i} = 0)$ for all i
4. Iterate the following:

   a. Optimize the combinatorial problem using TABU search for each segment Seg_k,

   which entails maximization w.r.t. $(\mathbf{x}_i^{(k)})_i$

$$P((\mathbf{x}_i^{(k)})_i | \mathbf{x}_i^{(-k)}, \mathbf{h}_i, m_i, \sigma_i^2)_i, \sigma_\varepsilon^2, \mathbf{z}) \tag{5.11}$$

   b. Compute the closed form solutions for the shape and discharge pattern parameters

   $(\mathbf{h}_i, m_i, \sigma_i^2)_i)_i, \sigma_\varepsilon^2$ until convergence

In equation 5.11, $\mathbf{x}_i^{(k)} = \mathbf{x}_i \cap \text{Seg}_k, i = 1, ..., I$ describe the discharge vector of each neuron within the given segment. The subscript $(-k)$ denotes the variables belonging to segments $\cup_{j \neq k} \text{Seg}_j$.
The maximization algorithm for step 4a using the TABU search algorithm is not further explained in this work, and the maximization algorithm for step 4b is explained in section 5.2.3.

5.2.3 Maximization on continuous parameters

This section describe the maximization of the statistical, continuous parameters $(m_i, \sigma_i^2$. [Ge et al. 2009] The remaining parameters and the maximization on combinatorial sets (TABU search) are not considered in this work.

Maximization w.r.t. m_i

The maximization of m_i is shown in equation 5.12.

$$\begin{aligned} m_i &= \max_{m_i} P(\Theta|\mathbf{z}) = \max_{m_i} P(\mathbf{x}_i|m_i,\sigma_i^2)P(m_i) \\ &= \max_{m_i}\left(\exp\left\{-\frac{1}{2\sigma_i^2}\sum_{j=1}^{n_i-1}(S_{ij}-m_i)^2\right\}P(m_i)\right) \end{aligned} \tag{5.12}$$

where $S_{ij} = \mathbf{x}_{i,j+1} - \mathbf{x}_{i,j} - T_R$ is the set of Gaussian samples (discharge rate minus refractory period). Equation 5.5 describes the Gaussian term $P(\mathbf{x}_i|m_i,\sigma_i^2)$. The prior law $P(m_i)$ and the product in 5.12 is also Gaussian. The maximum is found as in equation 5.13.

$$m_i = \left(\frac{\mu_0}{\sigma_0^2} + \frac{\sum_{j=1}^{n_i-1} S_{ij}}{\sigma_i^2}\right) / \left(\frac{1}{\sigma_0^2} + \frac{n_i-1}{\sigma_i^2}\right) \tag{5.13}$$

Maximization w.r.t. σ_i^2

The maximization of σ_i^2 is shown in equation 5.14.

$$\begin{aligned} \sigma_i^2 &= \max_{\sigma_i^2} P(\Theta|\mathbf{z}) = \max_{\sigma_i^2} P(\mathbf{x}_i|m_i,\sigma_i^2)P(\sigma_i^2) \\ &= \max_{\sigma_i^2}\left(\sigma_i^{-(n_i-1)}\exp\left\{-\frac{1}{2\sigma_i^2}\sum_{j=1}^{n_i-1}(S_{ij}-m_i)^2\right\}P(\sigma_i^2)\right) \end{aligned} \tag{5.14}$$

The maximum is found in equation 5.15.

$$\sigma_i^2 = \left(\beta_i + \sum_{j=1}^{n_i-1}(S_{ij}-m_i)^2/2\right) / \left(\alpha_i + 1 + \frac{n_i-1}{2}\right) \tag{5.15}$$

Because of the regularity constraint for the ISI $\sigma_i/m_i < \mathrm{Th}_{\sigma_i}$, the following is valid for $\sigma_i^2 \leftarrow \min\{\sigma_i^2, (\mathrm{Th}_{\sigma_i} m_i)^2\}$, which can be tuned depending of the type of neural signal relevant for spike sorting.

Spike sorting method chosen for comparison

6

This chapter describes the methods that are chosen for comparison with the method in focus in this present work, based on unsupervised Bayesian decomposition.
For comparison, the method **Wave_Clus** are used, based on the work from Quiroga et al. [2004]. Wave_Clus is an unsupervised spike detection and sorting method, that uses wavelets and superparamagnetic clustering.

6.1 Wave_Clus

The Wave_Clus program is a method for detecting and sorting spikes from multiunit recordings. As a short summary, the method uses the wavelet transform for feature extraction and superparamagnetic clustering (SPC) for automatic spike classification. SPC does the classification without assumptions such as low variance or Gaussian distributions (in contrast to the UBD method in focus in this present work, see chapter 5 page 25).
Throughout this report, Wave_Clus is chosen for comparison with the UBD method, using several simulated data sets with characteristics that closely resemble those of real human intra-cortical recordings, including one human data set. For a description of the test data, see chapter 7.1 page 37. [Quiroga et al. 2004]

The graphical user interface of the Wave_Clus program is shown in figure 6.1. After loading the data, the unsupervised method automatically performs the spike sorting, and plots the different clusters, the distributions of inter-spike intervals for each cluster, the two-dimensional feature projection (the wavelet coefficients), and the cluster size as a function of temperature (see section 6.1.1 below). It is possible for the user to change the cluster size and temperature at any time, after which a new clustering process and plotting are executed.

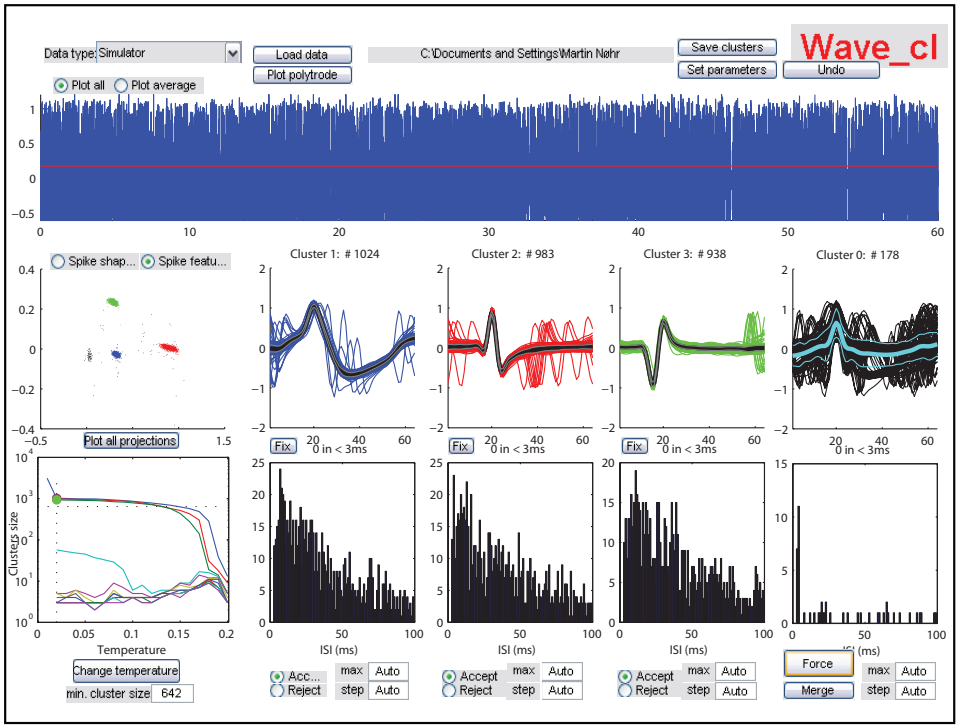


Figure 6.1: Screenshot of the graphical user interface of Wave_Clus. An example of simulated intra-cortical data is loaded, and the spike sorting results are shown. The raw signal, different clusters, the inter-spike interval distributions, and the two-dimensional feature projection are plotted. [Quiroga et al. 2004]

6.1.1 Spike sorting method

The algorithm behind Wave_Clus is a unsupervised clustering algorithm, that uses the wavelet transform for feature extraction. This method gives a time-frequency decomposition if the neural recording with optimal resolution in both time and frequency domains. With the clustering procedure based on superparamagnetic clustering, the Wave_Clus program encompasses three very principle stages of spike sorting, see section 3.3 page 10:

1. Spike detection with thresholding
2. Extraction and selection of spike features using wavelet transform
3. Clustering of the selected spike features

Figure 6.2 summarizes the Wave_Clus spike sorting algorithm.

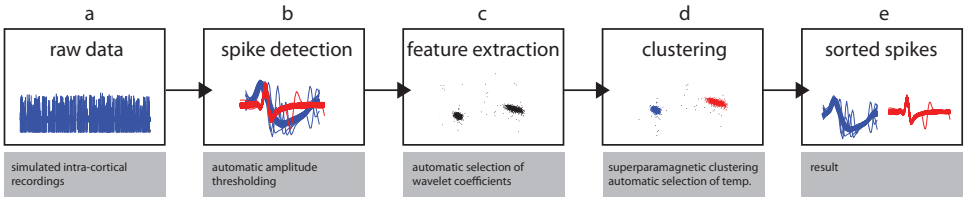


Figure 6.2: Overview of the basic steps in the Wave_Clus spike sorting algorithm. Inspired from [Quiroga et al. 2004]

Spike detection

Wave_Clus performs spike detection by amplitude thresholding after band pass filtering the signal with a four-pole Butterworth filter (300-6000 Hz). The threshold was automatically set in equation 6.1.

$$\text{Thr} = 4\sigma_n \text{ where } \sigma_n = \text{median}\left\{\frac{|\mathbf{x}|}{0.6745}\right\} \tag{6.1}$$

where $\mathbf{x}$ is the filtered signal, and σ_n is an estimate of the standard deviation of the background noise [Quiroga et al. 2004].
One potential problem by taking the standard deviation of the signal (with spikes) could be that a very high threshold was determined, especially in case of high spike firing rates, or high spike amplitudes. The reason of using the median is to diminish the interference of spikes, under the assumption that spikes amount to a small fraction of all samples. [Quiroga et al. 2004]

Feature extraction and selection

The wavelet transform used for feature extraction is a time-frequency representation of the neural recording. The transform has two main features. The first is that it provides an optimal resolution in both time and frequency domains, the second are that it does not require signal stationarity. The wavelet transform is defined in equation 6.2 as the convolution between the signal $x(t)$ and the wavelet functions $\psi_{a,b}(t)$ [Mallat 1989].

$$W_\psi X(a,b) = \langle x(t) | \psi_{a,b}(t) \rangle \tag{6.2}$$

where $\psi_{a,b}(t)$ are dilated and shifted versions of the unique wavelet function $\psi(t)$, which is defined in equation 6.3.

$$\psi_{a,b}(t) = |a|^{-1/2} \psi\left(\frac{t-b}{a}\right) \tag{6.3}$$

where a and b are the scale and translation parameters respectively. The Wave_Clus method uses a four-level decomposition using Haar wavelets, which are rescaled square functions. Haar wavelets allow the discriminative features of

the spikes to be described with a few wavelet coefficients, and without a priori assumptions about the spike shapes. [Quiroga et al. 2004]
The wavelet transform maps the neural recording (represented by the independent variable t) onto a function of two independent variables a, b. Contracted versions of the wavelet function match the high-frequency components, and dilated versions match the low-frequency components. The correlation between the recorded signal and the wavelet functions of different sizes provides details of the signal at several scales. These correlations with the different wavelet functions are arranged in the hierarchical scheme multiresolution decomposition [Mallat 1989].

After spike detection and computation of the wavelet transform, 64 wavelet coefficients are obtained for each spike. The aim is to select a few coefficients that best describe the spike shapes, and is multimodal distributed (more than one spike class). The algorithm selects the 10 best coefficients automatically with a Kolmogorov-Smirnov (KS) test (not described in this work) for normality, without assuming any particular distribution of the data. KS provides an expression for the deviation from normality as a sign of a multimodal distribution. The 10 coefficients with the largest deviation from normality where used as input to the clustering algorithm. [Quiroga et al. 2004]

The Wave_Clus method does not deliberately take the problem of overlapping spikes into consideration. Overlapping spikes may introduce outliers in the distribution of wavelet coefficients that result in high deviation from normality, which result in more clusters. In order to minimize this effect, only coefficients with values within ± 3 standard deviations are considered [Quiroga et al. 2004].

The Wave_Clus method is also capable of using principal component analysis (PCA) for feature extraction. PCA is also explained in section 3.6.1 page 17.

Clustering

Superparamagnetic clustering (SPC) is based on simulated interactions between each data point and its K-nearest neighbors [Blatt, Wiseman & Domany 1996]. The following will not be an exhaustive description of the models behind the superparamagnetic clustering, but will solely go through the important principles relevant to spike sorting. The clustering method is based on a Potts model [Blatt et al. 1996]. The initial step is to represent the m selected features of each spike i by a point $\mathbf{x}_i$ in an m-dimensional phase space. The interaction strength between points $\mathbf{x}_i$ is defined in equation 6.4 [Quiroga et al. 2004].

$$J_{ij} = \begin{cases} \frac{1}{K}\exp\left(-\frac{||\mathbf{x}_i-\mathbf{x}_j||^2}{2a^2}\right) & \text{if } \mathbf{x}_i \text{ is a nearest neighbor of } \mathbf{x}_j \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

where a is the average nearest-neighbors distance, and K is the number of nearest neighbors. The strength of interaction between the nearest-neighbor spikes J_{ij} decreases exponentially with increasing Euclidean distance $d_{ij} = ||\mathbf{x}_i - \mathbf{x}_j||^2$. This can be interpreted as similarity of the selected features, meaning that similar spikes that belongs to the same clusters will have a strong interaction.
The second step in the clustering procedure is to assign an initial random state s from 1 to q to each point $\mathbf{x}_i$. The main idea is to iteratively change the initially configurated state s, for a randomly selected $\mathbf{x}_i$, to a new state s_{new}, randomly chosen between 1 and q. Now, the probability that a the nearest neighbors of x_i will also change their state to s_{new} is given by equation 6.5 [Quiroga et al. 2004].

$$p_{ij} = 1 - \exp\left(-\frac{J_{ij}}{T}\delta_{s_i,s_j}\right) \tag{6.5}$$

where T is the temperature. Only nearest neighbors of $\mathbf{x}_i$ that were in the same previous state s are candidates to change their states to s_{new}. $\mathbf{x}_i$s that change state create a so-called "frontier", and cannot change again during the same iteration. For each point of the frontier, equation 6.5 is computed, to calculate the probability of changing the state to s_{new} for their respective neighbors. The frontier is updated until it does not change any more. This entire procedure is repeated for every point to get representative statistics. The consequence is that points that are close together

(corresponding to the same cluster), will change their state together. This is quantified by measuring the point-point correlation $\langle \delta_{s_i,s_j} \rangle$, and then assign $\mathbf{x}_i, \mathbf{x}_j$ to the same cluster if $\langle \delta_{s_i,s_j} \rangle \geq \theta$, where θ is a given threshold [Quiroga et al. 2004].
Quiroga et al. [2004] uses $q = 20$ states, $K = 11$ nearest neighbors, $N = 500$ iterations, and $\theta = 0.5$.

The clustering results is highly dependent on the temperature T [Blatt et al. 1996]. From equation 6.5 it is seen that a high temperature result in a low probability of changing the state of neighboring points together, and a low temperature corresponds to a higher probability. According to Blatt et al. [1996], at a certain medium range of temperatures between high and low, the system reaches a so-called "superparamagnetic" phase in which neighboring points will change their phase simultaneously. In relation to the spike clustering issue, a low temperature will results in all points being considered as a single cluster, whereas a high temperature will partitioning the data into several clusters with a few members each. Though, the temperatures corresponding to the superparamagnetic phase, only those points that are grouped together will change their state simultaneously. [Quiroga et al. 2004]

To illustrate the clustering procedure, figure 6.3 shows an example of 2400 points distributed in three distinct clusters. (A) in figure 6.3: The challenge in this specific example is that the clusters partially overlap, have large variance, and

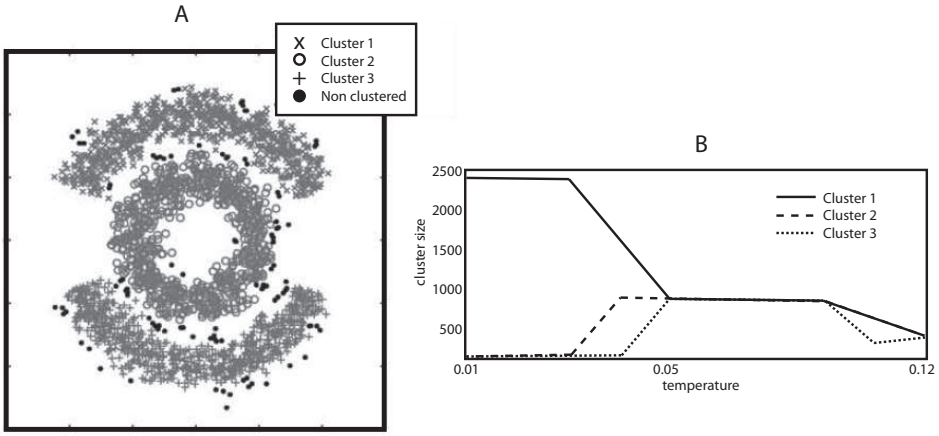


Figure 6.3: Example of superparamagnetic clustering of 2400 points in three clusters. (A) shows the two dimensional data distributed in three clusters. (B) shows the cluster size as a function of temperature. At a temperature of 0.05, the transition to the superparamagnetic phase occurs, and the correct three classes are separated. Inspired by [Quiroga et al. 2004]

their centers fall outside the clusters. Furthermore, the distance between random chosen points of the same cluster, may in some cases be significantly larger than the distance between points from different clusters, which may result in misclassifications from traditionally clustering algorithms. (B) shows the performance of the SPC, and plots the number of points assigned to each cluster as a function of the temperature. At low temperature, all 2400 points are gathered in one single cluster. At temperatures between 0.4 and 0.5 the clusters breaks down into three clusters, in the superparamagnetic transition. The clusters in (A) were performed with a temperature of 0.5.

Results and comparison

III

Test data

7

For testing the method in focus in this report, the UBD method, and the method chosen for comparison, the Wave_Clus method, different test data are used. The UBD method is originally designed for intra-muscular EMG signal decomposition, but is in this present work tested with other types of neural recordings, including simulated intra-cortical recordings and real human intra-cortical recordings. The Wave_Clus method is testes with the exact same data, to make a comparison possible.
Figure 7.1 shows an overview of the three collections of test data.

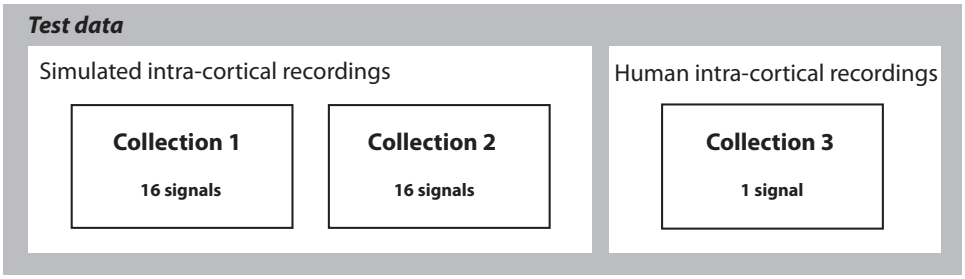


Figure 7.1: Overview of the three collections of test data used.

7.1 Simulated intra-cortical recordings

The testing of the two methods is done with two different sets of simulated intra-cortical recordings. The first collection (collection 1) of simulated intra-cortical signals are especially created for this project, and are described in section 7.1.1. Collection 1 is simulated in this project to able to control the setting of signal parameters (neuronal refraction period, maximal overlap, signal-to-noise ratio, etc.).
The second collection (collection 2) of signals are created by [Quiroga et al. 2004], and a description follows below in section 7.1.2.

7.1.1 Collection 1

The following describes the collection (collection 1) of simulated intra-cortical recordings build for this project.
For testing the two methods in a controllable setting, this signal collection is created with a variety of controlled signal parameters, described in the following.

The simulated signals were build using a database of 12 different spike shapes from actual recordings in the neocortex and basal ganglia [Quiroga et al. 2004], also described for collection 2 in subsection 7.1.2. The 12 spike shapes are subdivided into four sets of three spike shapes, where each set consists of four signals with varying noise level. This results in a total of 16 test signals. Figure 7.2 shows the four sets (12 spikes total) of spike shapes used in the simulations. Every signal was build 60 seconds long, simulated at a sampling rate of 24 kHz, and the generation of

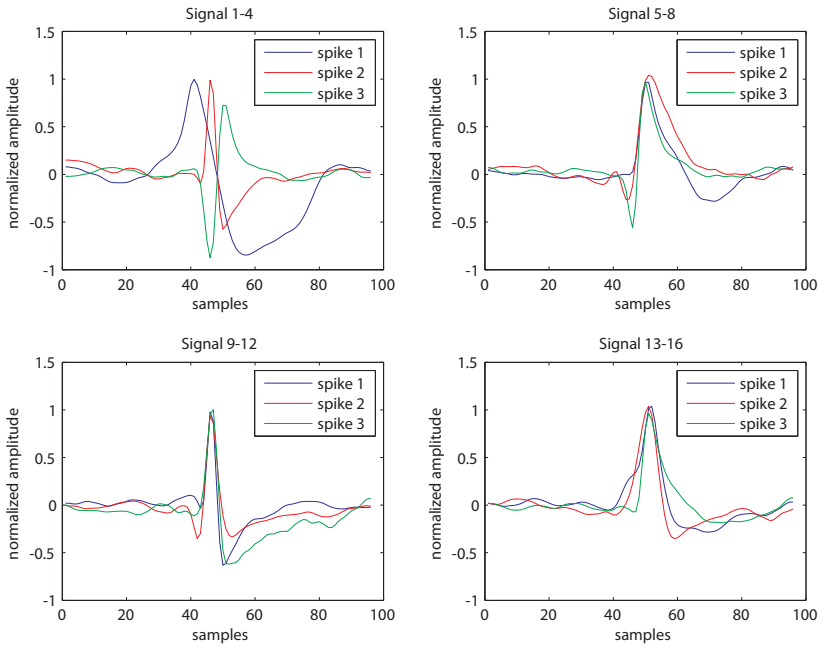


Figure 7.2: Overview of the four sets of spike shapes used in collection 1. Each set is represented by three spike shapes from [Quiroga et al. 2004]. A total of 16 signals are generated.

background noise was made with a white Gaussian noise function in Matlab, to imitate the noise from distant neurons in real recordings. The amplitude level of the background noise was adjusted according to its standard deviation, and was in each set of signals set to [0.05, 0.10, 0.15, 0.20] relative to the amplitude of the spike classes. The four sets of spike classes (train of three distinct spikes with 4 ms length) was superimposed on the noise signals at random times, however with the constraint, that spikes from a given class holds a neuronal refractory period of 10 ms. The peak amplitude of the distinct spikes was normalized to 1, and all simulations have a Gaussian distribution of inter-spike intervals, and a mean firing rate of 15 Hz. Because of the randomized firing of the three spikes, a number of spike overlaps are present in all simulated signals, and total overlap was allowed.
Spike timing and correct spike class identities were saved together with the signal for evaluation.

Signal overview

An overview of collection 1 of simulated signals are shown in table 7.1. An overlapping spike is defined as a spike pair within 64 data points (approx. 2.7 ms).

Collection 1				
Simulated signal	#	Noise level	Number of spikes	Number of overlapping spikes
Set 1	1	0.05	2700	248
	2	0.10	2700	275
	3	0.15	2700	286
	4	0.20	2700	279
Set 2	5	0.05	2700	268
	6	0.10	2700	251
	7	0.15	2700	250
	8	0.20	2700	256
Set 3	9	0.05	2700	281
	10	0.10	2700	300
	11	0.15	2700	291
	12	0.20	2700	265
Set 4	13	0.05	2700	298
	14	0.10	2700	262
	15	0.15	2700	262
	16	0.20	2700	290

Table 7.1: Overview of **collection 1** of simulated intra-cortical signals generated in this project. Four different data sets are provided, with four signals in each with varying noise level. For each 60 seconds signal, the noise level, number of spikes and the number of overlapping spikes are presented.

Figure 7.3 shows four examples of 1 seconds fragments of the simulated signals generated for this project, one fragment for each of the four noise levels, all from the first signal set (signal 1-4).

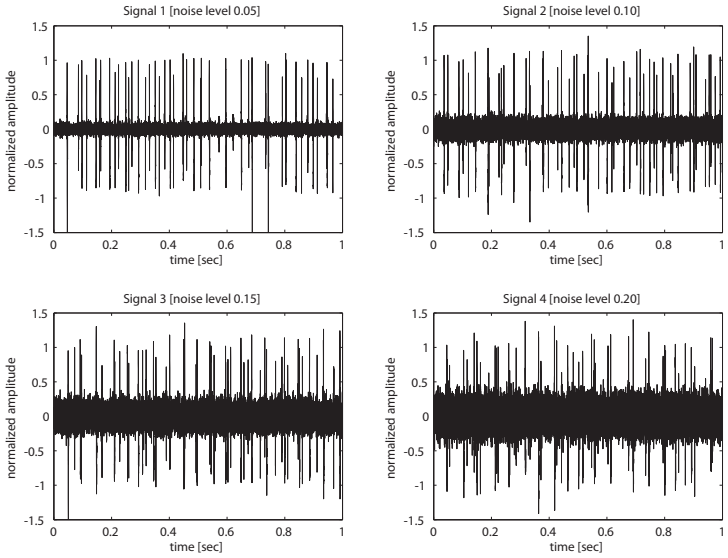


Figure 7.3: Four examples of 1 second fragments of the simulated signals generated for this project, one fragment for each of the four noise levels, all from the first signal set (signal 1-4)

7.1.2 Collection 2

The second collection of test data used is different simulated intra-cortical signals created by [Quiroga et al. 2004]. The simulated signals were formed using a database of 594 different average spike shapes from recordings in the neocortex and basal ganglia. The generation of background noise was created with randomly selected spikes from the database, superimposed at random times and with random amplitudes. The superimposition was conducted for half the times of samples, to imitate the background noise of real recordings, which is generated by the action potentials from distant neurons. Finally, superimposition of a train of three distinct spike shapes was done on the noise signal at random times. These three spike shapes was also selected from the database, and their amplitudes was normalized to a peak value of 1. The amplitude level of the noise was adjusted according to its standard deviation, and was set to [0.05, 0.10, 0.15, 0.20] relative to the amplitude of the spike classes in three signals. An advantage of constructing noise from spikes is that the noise shares a similar power spectrum with the spikes themselves. The aim was to make the procedure of spike sorting more challenging than scenarios with white noise distribution of background activity. Spike timing and correct spike class identities was saved together with the signal for evaluation. [Quiroga et al. 2004]

Initially, the signals was simulated at a sampling rate of 96 kHz, and subsequently down sampled to 24 kHz, and all signals are 60 seconds long. The down sampling was conducted to imitate actual recording conditions in which samples do not necessarily fall on the same features within a spike. The consequence may be that the peak of the signal does not necessarily match with a discrete sample. [Quiroga et al. 2004]

The three clear spikes in all simulations have a Poisson distribution of inter-spike intervals, and a mean firing rate of 20 Hz. Furthermore, a 2 ms refractory period constraint between spikes of the same class was ensured. Because of the randomized firing of the three spikes, a number of spike overlaps are present in all simulated signals. [Quiroga et al. 2004] This complication also improves the imitation of real recordings, and is considered in the results section 9.1 page 55.

Signal overview

An overview of collection 2 of simulated signals are shown in table 7.2. An overlapping spike is defined as a spike pair within 64 data points (approx. 2.7 ms).

Collection 2				
Simulated signal	#	Noise level	Number of spikes	Number of overlapping spikes
1_easy	1	0.05	3514	785
	2	0.10	3522	769
	3	0.15	3477	784
	4	0.20	3474	796
	5	0.25	3298	N/A
	6	0.30	3475	N/A
	7	0.35	3534	N/A
	8	0.40	3386	N/A
2_easy	9	0.05	3410	791
	10	0.10	3520	826
	11	0.15	3411	763
	12	0.20	3526	811
3_diff	13	0.05	3383	767
	14	0.10	3448	810
	15	0.15	3472	812
	16	0.20	3414	790
4_diff	17	0.05	3364	829
	18	0.10	3462	720
	19	0.15	3440	809
	20	0.20	3493	777

Table 7.2: Overview of **collection 2** of simulated intra-cortical signals from [Quiroga et al. 2004]. Four different data sets are provided (two relatively easy to spike sort, two relatively difficult to spike sort). For each 60 seconds signal, the noise level, number of spikes and the number of overlapping spikes are presented.

Figure 7.4 shows an example of a fragment from two different simulated signals in collection 2. In each example, a fragment of the raw signal (B), the three disclosed spike shapes (A), and a small section data with classification results from Wave_Clus (C) is shown. The example with noise level 0.10 is relatively easy to spike sort, because of the noise level and relatively diverse spike shapes. The example with noise level 0.15 is more difficult to spike sort, because of approximately similar peak amplitudes and very identical spike shapes.

7.2 Human intra-cortical recordings

The testing of the two methods are also conducted with real human intra-cortical recordings (collection 3).
Collection 3 consists of a single 30 minutes signal, recorded from the medial temporal lobe of a human subject with a sampling rate of 32.258 kHz. The data are provided from Quiroga et al. [2004] and are described in both Quiroga [2009] and Fried, MacDonald & Wilson [1997].
The subject had pharmacologically intractable epilepsy and was implanted with intracranial electrodes for clinical reasons, in order to identify the seizure focus for potential surgical resection [Fried et al. 1997].
The electrodes were placed based on clinical criteria, and following patients informed consent. The electrodes contains micro wires, were implanted using MRI guidance, and consisted of a flexible polyurethane probe containing 9 40 mm platinum-iridium micro wires protruding approx. 4 mm into the tissue beyond the tip of the probe [Fried et al. 1997].
Throughout the recording session, the subject was presented to pictures of faces and objects in 1 s stimuli, followed by 3-5 s delay before the next stimulus. The recordings were attached to a preamplifier module with a gain of 5000, and a pass band of 0.3 Hz - 6 kHz. [Fried et al. 1997]

Figure 7.5 shows an example of a one second segment of the raw human intra-cortical signal. Figure 7.6 shows an example of a band pass filtered (300-3000 Hz), one second segment of the human intra-cortical signal. Several spikes are clearly seen above background noise. According to Quiroga et al. [2004], the signal contains spikes from three distinct neurons, and the spike shape results from Quiroga [2009] are used as reference in the following tests using this signal. Figure 7.7 shows the average spike shape results for the three neurons, including the inter-spike interval distributions.

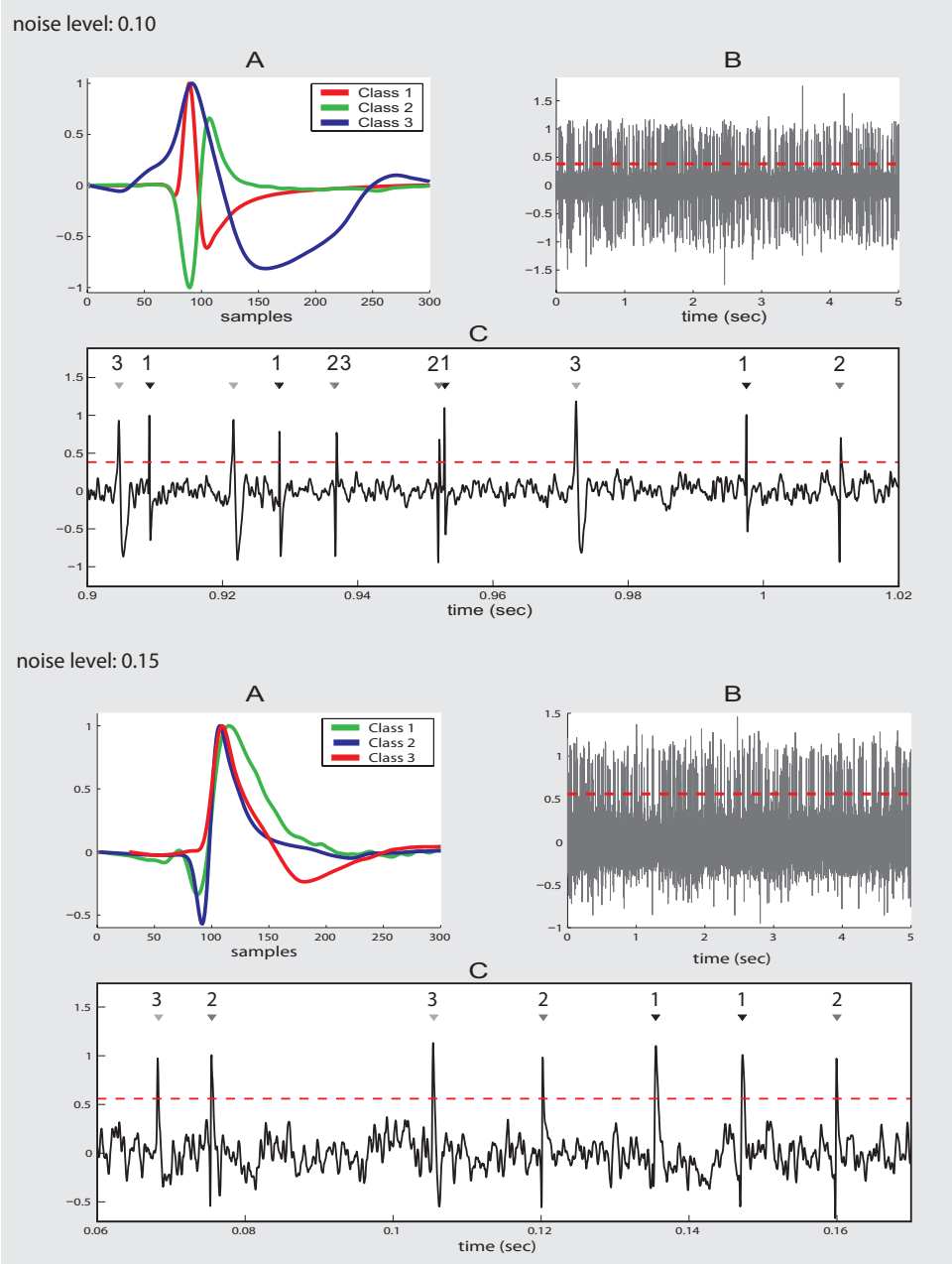


Figure 7.4: Example of a fragment from two different simulated signals from **collection 2**. In each example, a fragment of the raw signal (B), the three disclosed spike shapes (A), and a small section data with classification results from Wave_Clus (C) is shown. Inspired by [Quiroga et al. 2004]

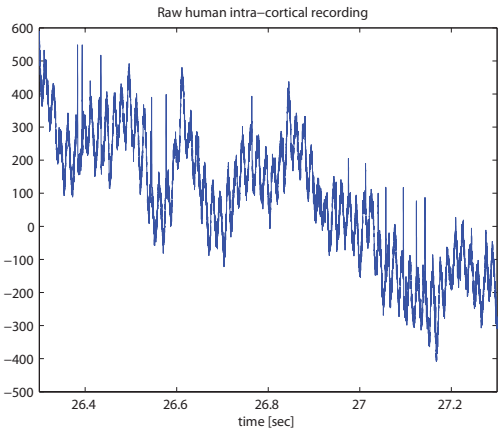


Figure 7.5: Example of a one second segment of the raw human intra-cortical signal.

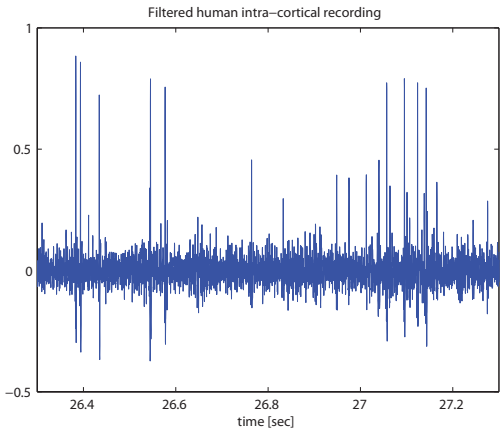


Figure 7.6: Example of a one second segment of the filtered human intra-cortical signal.

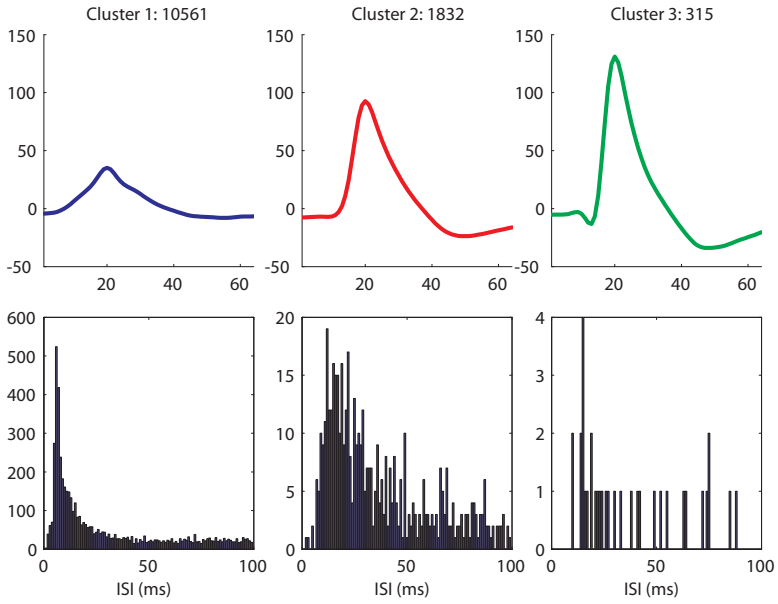


Figure 7.7: The average spike shape results for the three distinct neurons in the human intra-cortical signal, including the inter-spike interval distributions. The number of spikes in each cluster in the 30 minutes signal is given in Quiroga [2009]

Results using UBD

8

This chapter presents the results in the validation of the UBD method. The results from the performance test of the UBD method presented in the following sections are subdivided into spike detection results, and clustering results. Any comparison with results from the Wave_Clus method are first presented in chapter 10, page 63.

The tuning of the UBD method for intra-cortical signals, incorporates the refractory period and the limit of regularity in the inter-spike intervals, described in the method section 5 page 25. Both parameters are tuned for the processing of these results, and the refractory period was set to 10 ms, according to the properties of the simulated signals in collection 1. The limit of regularity, σ_i/m_i, was set to < 0.8, which in practice means that the constraint was very weak for intra-cortical recordings.

8.1 Simulated intra-cortical recordings

8.1.1 Collection 1

These results are produced using simulated data collection 1, described in section 7.1.1 page 38. All results are produced with 60 seconds signals, with a processing time of approximately 3 hours per signal.

Spike detection performance

The detection results from UBD using collection 1 are presented in table 8.1. Both reference data and detection results are presented, and to express detection performance, detection misses and false positives are listed.
To evaluate the overall performance of the method, the number of false positives is not considered in the clustering performance section.

Collection 1 - UBD		Reference		Detection results	
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	Misses total / %	False positives
Set 1	1	0.05	2700 (248)	12 [0.4 %]	53
	2	0.10	2700 (275)	28 [1.0 %]	86
	3	0.15	2700 (286)	309 [11.4 %]	284
	4	0.20	2700 (279)	372 [13.8 %]	531
Set 2	5	0.05	2700 (268)	22 [0.8 %]	41
	6	0.10	2700 (251)	62 [2.3 %]	107
	7	0.15	2700 (250)	254 [9.4 %]	356
	8	0.20	2700 (256)	389 [14.4 %]	402
Set 3	9	0.05	2700 (281)	27 [1.0 %]	31
	10	0.10	2700 (300)	52 [1.9 %]	70
	11	0.15	2700 (291)	245 [9.1 %]	210
	12	0.20	2700 (265)	402 [14.9 %]	387
Set 4	13	0.05	2700 (298)	64 [2.4 %]	87
	14	0.10	2700 (262)	101 [3.7 %]	191
	15	0.15	2700 (262)	251 [9.3 %]	344
	16	0.20	2700 (290)	418 [15.5 %]	577

Table 8.1: Results from spike detections using the UBD method and collection 1. The two uppermost columns separate the reference data and the detection results. In the "misses" column, the total number of misses is listed together with the number of ordinary "single-spike" misses and missed overlapped spikes. In the "False positives" column, the total number of false positives is presented.

A spike identified by the UBD method was considered a correct detection if it was detected within a window of 2 ms centered at the time of the occurrence of the true spike.

According to table 8.1, the amount of detection misses is highly dependent of the noise level in the 16 signals, and is very low (below 4%) in the two cases with low noise, in all four sets. The number of false positives shows the same pattern, with high amount of errors in the high noise signals. By inspecting the results, it is seen that a large fraction of the false positives is "false overlaps" detected by the method.
Figure 8.1 shows an example of a detection miss, where the red circle marks the miss, which is a consequence of spike overlap from spike class 1 and 3.

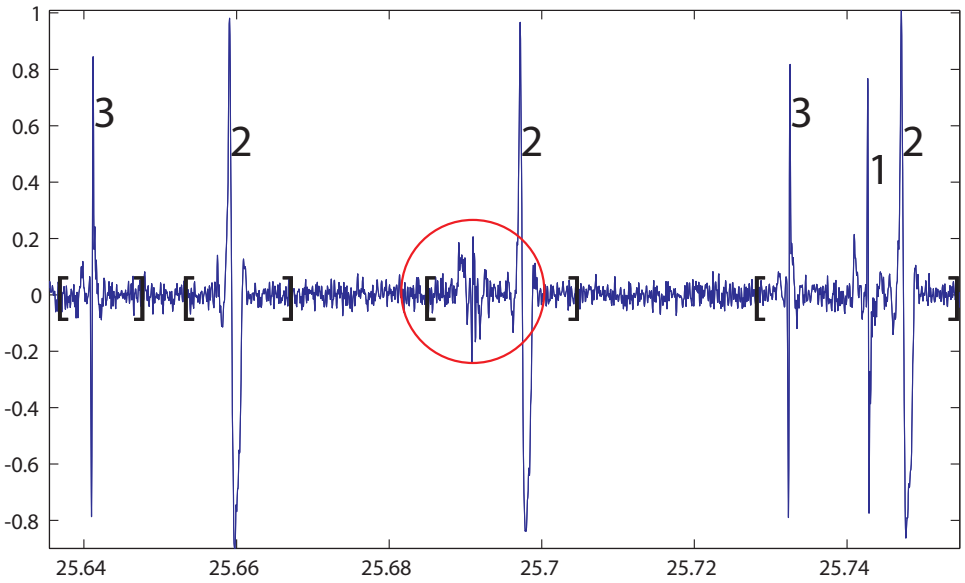


Figure 8.1: Examples of detection misses in signal 1, from collection 1. The miss is marked with the red circle, and is a consequence of spike overlap from spike class 1 and 3.

Clustering performance

The clustering results from UBD using collection 1 are presented in table 8.2.

Collection 1 - UBD		Reference		Detection results	Clustering results (Classification errors)	
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	No. spikes	No. errors / %	Total success
Set 1	1	0.05	2700 (248)	2688	80 [3.0 %]	96.6 %
	2	0.10	2700 (275)	2672	121 [4.5 %]	94.4 %
	3	0.15	2700 (286)	2391	290 [12.1 %]	77.8 %
	4	0.20	2700 (279)	2328	654 [28.1 %]	62.0 %
Set 2	5	0.05	2700 (268)	2678	108 [4.0 %]	95.2 %
	6	0.10	2700 (251)	2638	162 [6.1 %]	91.7 %
	7	0.15	2700 (250)	2446	356 [14.6 %]	73.4 %
	8	0.20	2700 (256)	2311	696 [30.1 %]	59.8 %
Set 3	9	0.05	2700 (281)	2673	75 [2.8 %]	96.2 %
	10	0.10	2700 (300)	2648	202 [7.3 %]	90.6 %
	11	0.15	2700 (291)	2455	265 [10.8 %]	81.1 %
	12	0.20	2700 (265)	2298	627 [27.3 %]	61.9 %
Set 4	13	0.05	2700 (298)	2636	228 [8.7 %]	89.2 %
	14	0.10	2700 (262)	2599	255 [9.8 %]	86.8 %
	15	0.15	2700 (262)	2449	411 [16.8 %]	75.5 %
	16	0.20	2700 (290)	2282	716 [31.4 %]	58.0 %
Average				2512	328 [13.1 %]	80.9 %

Table 8.2: Results from spike clustering using the UBD method and collection 1. Together with the detection results, the number of classification errors and total success are shown. The bottom part of the table presents the average values across all 16 signals.

It is seen that an average of 2512 spikes are detected out of 2700, which corresponds to 93%. The number of classification errors is dependent of the noise level in each signal set, which has an average of 13.1%. Especially signal set 4 has high classification errors, which could be related to more similar and complex spike shapes (see figure 7.2).
The total success is computed with equation 8.1.

$$\text{total success} = \frac{N_{\text{ref}} - N_{\text{miss}} - N_{\text{class}}}{N_{\text{ref}}} * 100 \tag{8.1}$$

where N_{ref} is the number of true spikes, N_{miss} is the number of spike misses, and N_{class} is the number of classification errors.

The UBD method performs above 90% in the two cases with lowest noise in the first three signal sets, and close to 90% in the last set. The average success was 80.9%.
Figure 8.2 shows examples of clustering results in short time intervals, using signal 1 in collection 1. (A) shows an example of five correct classified spikes in three segments (B) shows five correct classified spikes in four segments, with a correct classified overlap between spike 2 and 3 in the first segment. (C) shows five correct classified spikes in four segments, but with a "false positive" classified overlap in the second segment (class 1). Figure 8.3 shows the four sets of spike shapes, both original and the resulting estimated spike shapes by the UBD method, for signal 1,5,9, and 13. Some distortion is seen, especially for signal 9 (C).

8.1.2 Collection 2

These results are produced using collection 2, described in section 7.1.2 page 40. All results are produced with 60 seconds signals, with a processing time of approximately 3 hours per signal.
Collection 2 (used in this section) differs from collection 1 in several signal simulation parameters, among these the

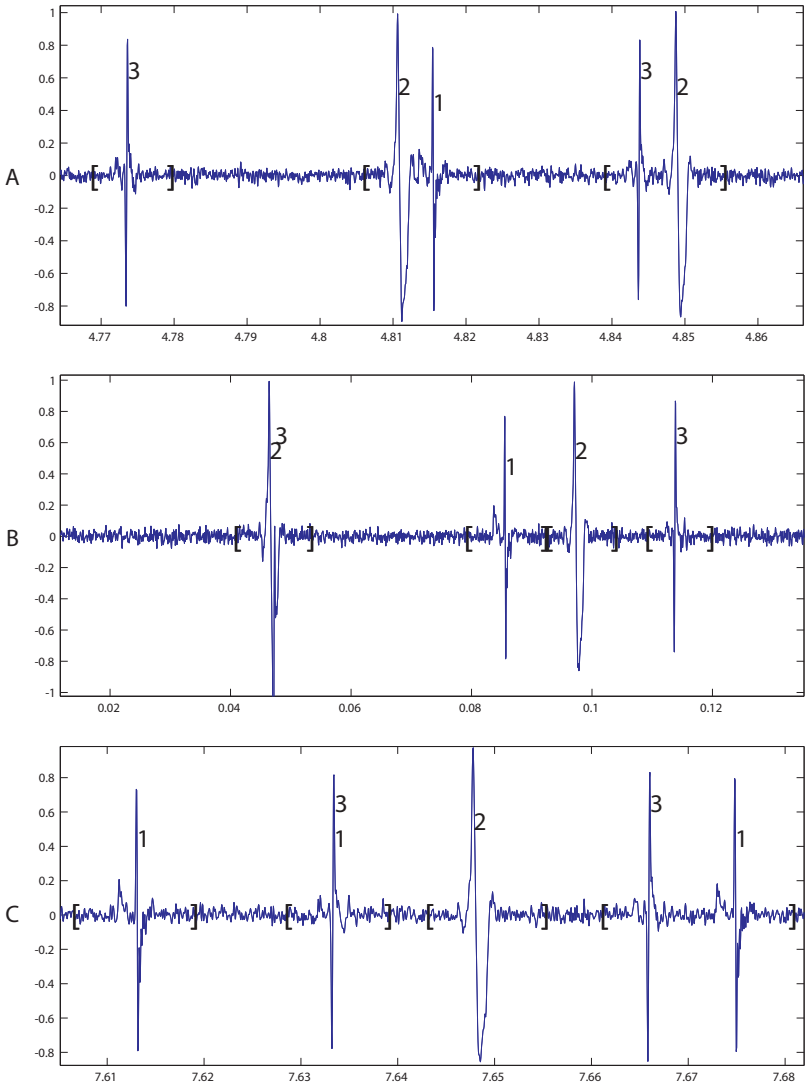


Figure 8.2: Examples of clustering results in short time intervals, using signal 1 in collection 1. (A) shows an example of five correct classified spikes in three segments (B) shows five correct classified spikes in four segments, with a correct classified overlap between spike 2 and 3 in the first segment. (C) shows five correct classified spikes in four segments, but with a "false positive" classified overlap in the second segment (class 1).

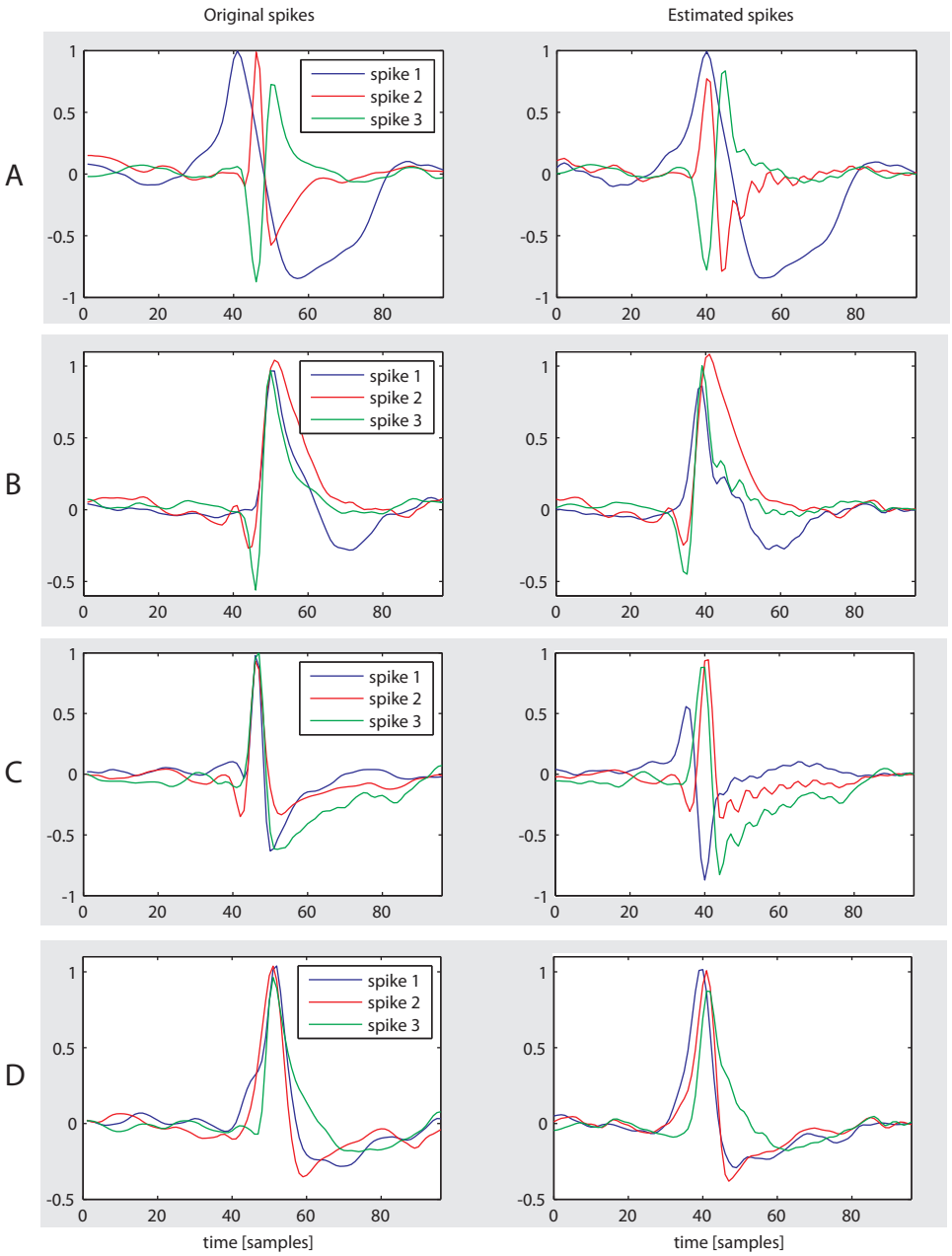


Figure 8.3: Examples of the original spike shapes compared with the resulting estimated spike shapes for signal 1,5,9, and 13 in collection 1.

refractory period, and the noise characteristics. As a consequence, the results may also differ using the two signal collections, which contributes to a thorough test of the two spike sorting methods.

Spike detection performance

The detection results from UBD using collection 2 are presented in table 8.3. Both reference data and detection results are presented, and to express detection performance, detection misses and false positives are listed.
To evaluate the overall performance of the method, the number of false positives is not considered in the clustering performance section.

Collection 2 - UBD		Reference		Detection results	
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	Misses total / %	False positives
1_easy	1	0.05	3514 (785)	1374 [39.1 %]	432
	2	0.10	3522 (769)	1296 [36.8 %]	381
	3	0.15	3477 (784)	1631 [46.9 %]	539
	4	0.20	3474 (796)	2178 [62.7 %]	581
2_easy	5	0.05	3410 (791)	1395 [40.9 %]	233
	6	0.10	3520 (826)	1176 [33.4 %]	634
	7	0.15	3411 (763)	1310 [38.4 %]	672
	8	0.20	3526 (811)	2035 [57.7 %]	599
3_diff	9	0.05	3383 (767)	1397 [41.3 %]	482
	10	0.10	3448 (810)	1490 [43.2 %]	492
	11	0.15	3472 (812)	1739 [50.1 %]	528
	12	0.20	3414 (790)	2124 [62.2 %]	603
4_diff	13	0.05	3364 (829)	1403 [41.7 %]	781
	14	0.10	3462 (720)	1385 [40.0 %]	488
	15	0.15	3440 (809)	1944 [56.5 %]	584
	16	0.20	3493 (777)	2089 [59.8 %]	631

Table 8.3: Results from spike detections using the UBD method and collection 2. The two uppermost columns separate the reference data and the detection results. In the "misses" column, the total number of misses is listed together with the number of ordinary "single-spike" misses and missed overlapped spikes. In the "False positives" column, the total number of false positives is presented.

A spike identified by the UBD method was considered a correct detection if it was detected within a window of 2 ms centered at the time of the occurrence of the true spike.

According to table 8.3, the number of missed spikes in all 16 signals in collection 2 is extremely high, varying from approximately 40 % - 60 % across the signals. The number of misses increases dependent of the noise level, and shows the same pattern as with collection 1. The number of false positives is also relatively high, with an average of 541 spikes.

Figure 8.4 shows an example from signal 1 in collection 2, where a high number of detection misses is present. In both (A) and (B), more than three spikes are present in each segment, which are symptomatic in each of the 16 signals in collection 2. In (A), three spikes are correct detected and classified, but five spikes are missed in the segment. In (B), two spikes are correct detected and classified, but three spikes are missed, and one false positive are present at the last spike in the segment. The refractory period for each single simulated neuron in collection 2 is only 2 ms, which causes more than three spikes in each segment. For further considerations on this topic, see the discussion in chapter 11 page 69.

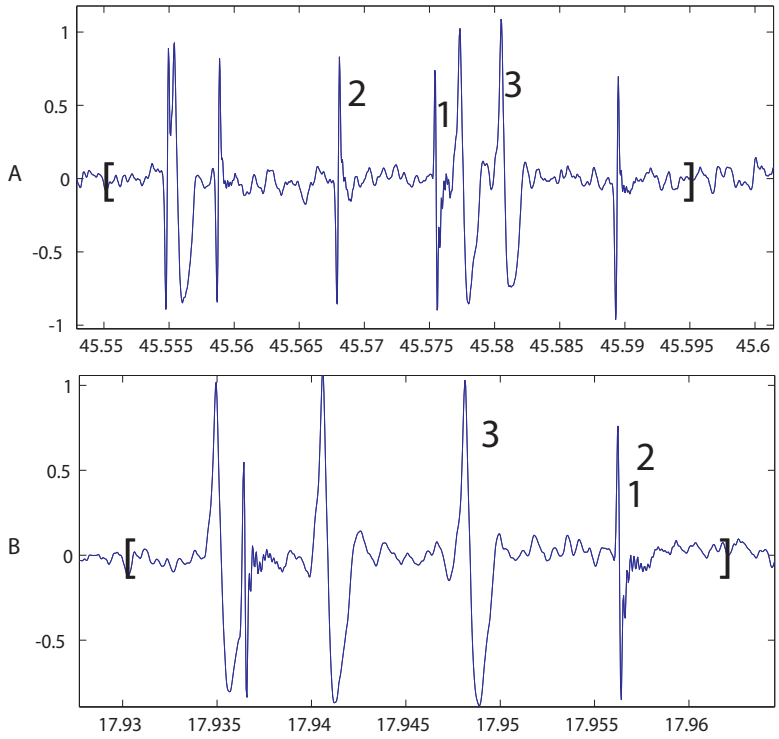


Figure 8.4: Examples from signal 1 in collection 2, where a high number of detection misses are present. In (A), three spikes are correct detected and classified, but five spikes are missed in the segment. In (B), two spikes are correct detected and classified, but three spikes are missed, and one false positive are present at the last spike in the segment.

Clustering performance

The clustering results from UBD using collection 2 are presented in table 8.4.

Collection 2 - UBD		Reference		Detection results	Clustering results (Classification errors)	
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	No. spikes	No. errors / %	Total success
1_easy	1	0.05	3514 (785)	2140	566 [16.1 %]	44.8 %
	2	0.10	3522 (769)	2226	736 [20.9 %]	42.3 %
	3	0.15	3477 (784)	1846	532 [15.3 %]	37.8 %
	4	0.20	3474 (796)	1296	316 [9.1 %]	28.2 %
2_easy	5	0.05	3410 (791)	2015	477 [14.0 %]	45.1 %
	6	0.10	3520 (826)	2344	725 [20.6 %]	46.0 %
	7	0.15	3411 (763)	2101	720 [21.1 %]	40.5 %
	8	0.20	3526 (811)	1491	331 [9.4 %]	32.9 %
3_diff	9	0.05	3383 (767)	1986	423 [12.5 %]	46.2 %
	10	0.10	3448 (810)	1958	369 [10.7 %]	46.1 %
	11	0.15	3472 (812)	1733	417 [12.0 %]	37.9 %
	12	0.20	3414 (790)	1290	256 [7.5 %]	30.4 %
4_diff	13	0.05	3364 (829)	1961	495 [14.7 %]	43.6 %
	14	0.10	3462 (720)	2077	672 [19.4 %]	40.6 %
	15	0.15	3440 (809)	1496	485 [14.1 %]	29.4 %
	16	0.20	3493 (777)	1404	489 [14.0 %]	26.2 %
Average				1835	501 [14.5 %]	38.6 %

Table 8.4: Results from spike clustering using the UBD method and collection 2. Together with the detection results, the number of classification errors and total success are shown. The bottom part of the table presents the average values across all 16 signals.

The average number of correct detected spikes from collection 2 is 1835. The number of classification errors is relatively high, with an average of 14.5 %. The high number of misses and classification errors results in a very low total success, which shows dependence of noise level, and an average of only 38.6 %. The total success is computed with equation 8.1.

8.2 Human intra-cortical recordings

The results in this section are based on the human intra-cortical recordings (collection 3) described in section 7.2 page 41. Because of the absence of knowledge about the "true" firing pattern of each of the distinct neuron in the signal, the results using Wave_Clus are used as reference data in the comparison.

Due to "out-of-memory" issues and a lack of processing time, only 120 seconds of the signal in collection 3 are used in the following results.

Spike detection performance

The results from the UBD method using collection 3 are presented in table 8.5. The neuronal refractory period was set to 2 ms in the detection of spikes from human intra-cortical recordings.

Collection 3 - UBD		Reference	Detection results	
Human signal	#	No. spikes	Misses	False positives
		spikes	total/ %	
Signal 1	1	740	52 [7.0 %]	678

Table 8.5: Results from spike detections and classification using the UBD method with collection 3.

Out of the 740 detected spikes in the reference, 52 (7.0%) spikes was missed by the UBD method. A large number of false positives was observed (678 spikes), but some false positives can be overlapping spikes, which is ignored by Wave_Clus.

Clustering performance

The clustering results from UBD using collection 3 are presented in table 8.6.

Collection 3 - UBD		Reference	Detection results	Clustering results (Classification errors)	
Human signal	#	No. spikes	No. spikes		Total success
		spikes	spikes	No. errors / %	
Signal 1	1	740	688	72 [10.5 %]	83.2 %

Table 8.6: Results from spike clustering using the UBD method and collection 3.

The UBD method detected and classified the same two neurons as in the reference, with 72 classification errors, and a total success of 83.2 % compared to the performance of Wave_Clus.
Figure 8.5 shows examples of clustering results in short time intervals, using collection 3. Both the classification results from UBD (black) and from Wave_Clus (red) are shown. (A) shows an example of four segments, seven spikes detected and classified by the UBD method, including one overlapping spike in the second segment. Wave_Clus agrees, except the detected overlap. The overlapping spike will be seen as a false positive, even though it is correctly detected. (B) shows six segments, nine spikes detected and classified by the UBD method. Compared to the reference from Wave_Clus, the UBD method makes one false positive in the first segment and one missing spike in the fifth segment.

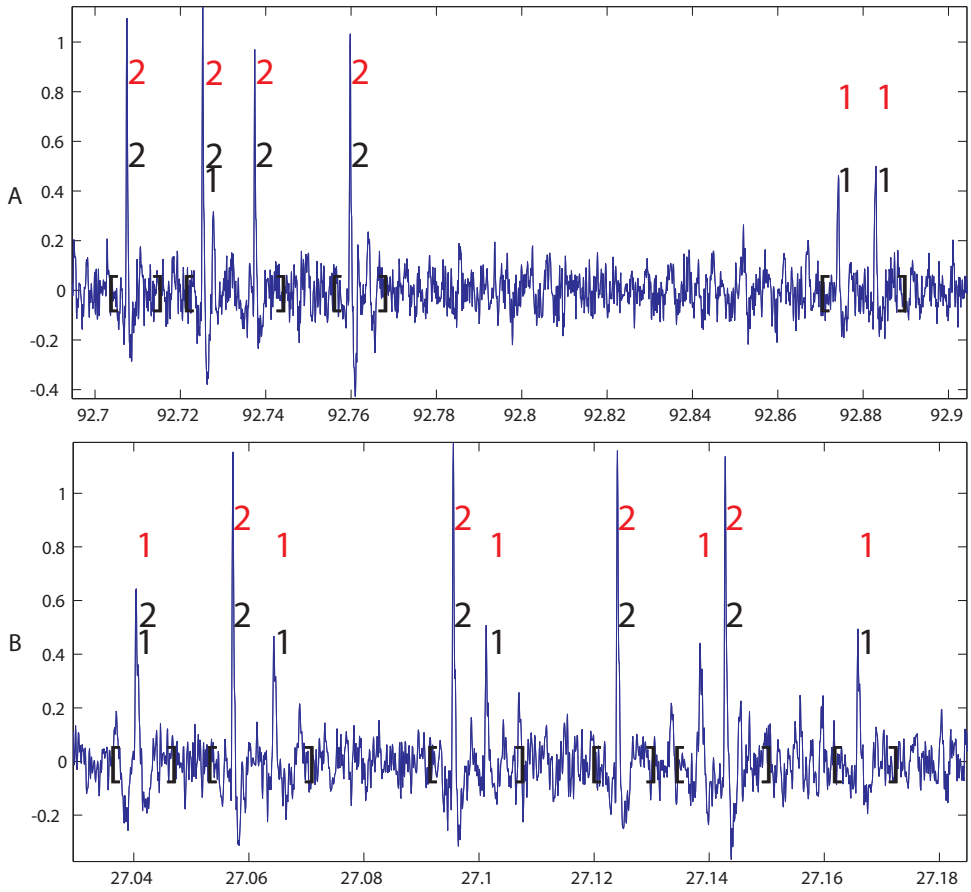


Figure 8.5: Examples of clustering results in short time intervals, using collection 3. (A) shows an example of four segments, seven spikes detected and classified by the UBD method, including one overlapping spike in the second segment. Wave_Clus agrees, except the detected overlap. (B) shows six segments, nine spikes detected and classified by the UBD method. Compared to the reference from Wave_Clus, the UBD method makes one false positive in the first segment, and one missing spike in the fifth segment.

Results using Wave_Clus 9

This chapter presents the results for the Wave_Clus method, in relation to the validation of the UBD method. Throughout the spike sorting process with Wave_Clus, no operator interaction (i.e. changing the temperature for better clustering) was performed, which made the process fully automatic and unsupervised (similar to the UBD method), for comparison reasons.

9.1 Simulated intra-cortical recordings

9.1.1 Collection 1

The following results in this section are based on the simulated intra-cortical recordings (collection 1) described in section 7.1.1, listed in table 7.1. The results are produced using the Wave_Clus method, with 60 seconds signals, and with a processing time of approximately 5 minutes per signal.

Spike detection performance

The detection results from Wave_Clus using collection 1 are presented in table 9.1. Both reference data and detection results are presented, and to express detection performance, detection misses and false positives are listed.

Collection 1 - WC		Reference		Detection results	
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	Misses total / %	False positives
Set 1	1	0.05	2700 (248)	246 [9.1 %]	32
	2	0.10	2700 (275)	278 [10.3 %]	31
	3	0.15	2700 (286)	303 [11.2 %]	3
	4	0.20	2700 (279)	307 [11.4 %]	0
Set 2	5	0.05	2700 (268)	344 [12.7 %]	27
	6	0.10	2700 (251)	253 [9.4 %]	14
	7	0.15	2700 (250)	270 [10.0 %]	11
	8	0.20	2700 (256)	273 [10.1 %]	5
Set 3	9	0.05	2700 (281)	349 [12.9 %]	28
	10	0.10	2700 (300)	299 [11.1 %]	7
	11	0.15	2700 (291)	300 [11.1 %]	9
	12	0.20	2700 (265)	287 [10.6 %]	13
Set 4	13	0.05	2700 (298)	383 [14.2 %]	45
	14	0.10	2700 (262)	266 [9.9 %]	17
	15	0.15	2700 (262)	264 [9.8 %]	5
	16	0.20	2700 (290)	303 [11.2 %]	9

Table 9.1: Results from spike detections using the Wave_Clus method and collection 1. The two uppermost columns separate the reference data and the detection results. In the "misses" column, the total number of misses is listed. In the "False positives" column, the total number of false positives is presented.

A spike identified by the Wave_Clus method was considered a correct detection if it was detected within a window of 2 ms centered at the time of the occurrence of the true spike.

According to table 9.1, the amount of detection misses is almost constant across all signals (not above 14%), independent of noise level and spike shape similarity. The number of false positives is very low in all cases, which expresses that the threshold is not too low.

Clustering performance

The clustering results from Wave_Clus using collection 1 are presented in table 9.2.

Collection 1 - WC		Reference		Detection results	Clustering results (Classification errors)		
Simulated signal	#	Noise level	No. spikes (overlaps)	No. spikes	Wavelets No. errors / %	PCA No. errors / %	Total success Wavelets / PCA
Set 1	1	0.05	2700 (248)	2454	206 [8.4 %]	498 [20.3 %]	83.3 % / 72.4 %
	2	0.10	2700 (275)	2422	166 [6.9 %]	795 [32.8 %]	83.6 % / 60.3 %
	3	0.15	2700 (286)	2397	186 [7.8 %]	548 [22.9 %]	81.9 % / 68.5 %
	4	0.20	2700 (279)	2393	178 [7.4 %]	660 [27.6 %]	82.1 % / 64.2 %
Set 2	5	0.05	2700 (268)	2356	137 [5.8 %]	502 [21.3 %]	82.2 % / 68.7 %
	6	0.10	2700 (251)	2447	185 [7.6 %]	756 [30.9 %]	83.8 % / 62.6 %
	7	0.15	2700 (250)	2430	176 [7.2 %]	1544 [63.5 %]	83.5 % / 32.8 %
	8	0.20	2700 (256)	2427	209 [8.6 %]	1513 [62.3 %]	82.2 % / 33.9 %
Set 3	9	0.05	2700 (281)	2351	80 [3.4 %]	706 [30.0 %]	84.1 % / 60.9 %
	10	0.10	2700 (300)	2401	165 [6.9 %]	954 [39.7 %]	82.9 % / 53.6 %
	11	0.15	2700 (291)	2400	154 [6.4 %]	1253 [52.2 %]	83.2 % / 42.5 %
	12	0.20	2700 (265)	2413	170 [7.0 %]	1749 [72.5 %]	83.1 % / 24.6 %
Set 4	13	0.05	2700 (298)	2317	163 [7.0 %]	385 [16.6 %]	79.8 % / 71.6 %
	14	0.10	2700 (262)	2434	217 [8.9 %]	1036 [42.6 %]	82.1 % / 51.8 %
	15	0.15	2700 (262)	2436	192 [7.9 %]	1500 [61.6 %]	83.1 % / 34.7 %
	16	0.20	2700 (290)	2397	13 [0.5 %]	1702 [71.0 %]	88.3 % / 25.7 %
Average				2405	162 [6.7 %]	1006 [41.8 %]	83.1 % / 51.8 %

Table 9.2: Results from spike clustering using the Wave_Clus method and collection 1, with both wavelets and PCA for feature extraction.

It is seen that an average of 2405 spikes are detected out of 2700, which corresponds to 89%. The number of classification errors using wavelets is almost constant across all signals, and independent of the noise level, with the average of 6.7%. The number of classification errors using PCA is significantly higher, and dependent of the noise level, with very high number of errors in the high noise cases. The average is 41.8%. The total success using wavelets is 83.1%, and 51.8% using PCA. The total success is computed using equation 8.1.

Figure 9.1 shows the results from four (A-D) simulated signals (using noise level 0.1) in collection 1. The first three columns show the three clusters. Spike pairs appearing with a lower time separation than 0.5 ms (overlapping spikes) are not considered by Wave_Clus. The fourth column shows the original spike shapes in each signal for reference. Figure 9.2 shows an example of the temperature setting in the clustering process of four simulated signals (signal 2,6,10,14 in table 9.2) using noise level 0.1 in collection 1. It is seen that the algorithm automatically sets the temperature to a level corresponding the superparamagnetic regime, with clustering of three spike shapes into relatively large clusters. Figure 9.3 shows the distributions of inter-spike intervals in the three clusters for the four simulated signals (signal 2,6,10,14 in table 9.2) using noise level 0.1 in collection 1. The distribution of ISI can be assumed to have a Gaussian shape, which also is an assumption for the UBD method.

9.1.2 Collection 2

The following results in this section are based on the simulated intra-cortical recordings (collection 2) described in section 7.1.2, listed in table 7.2. The results are produced using the Wave_Clus method. Many results are also presented in [Quiroga et al. 2004], and selected are reproduced in this present work, to ensure thorough comparison with the UBD method.
Collection 2 (used in this section) differs from collection 1 in several signal simulation parameters, among these the refractory period, and the noise characteristics. As a consequence, the results may also differ using the two signal collections, which contributes to a thorough test of the two spike sorting methods.

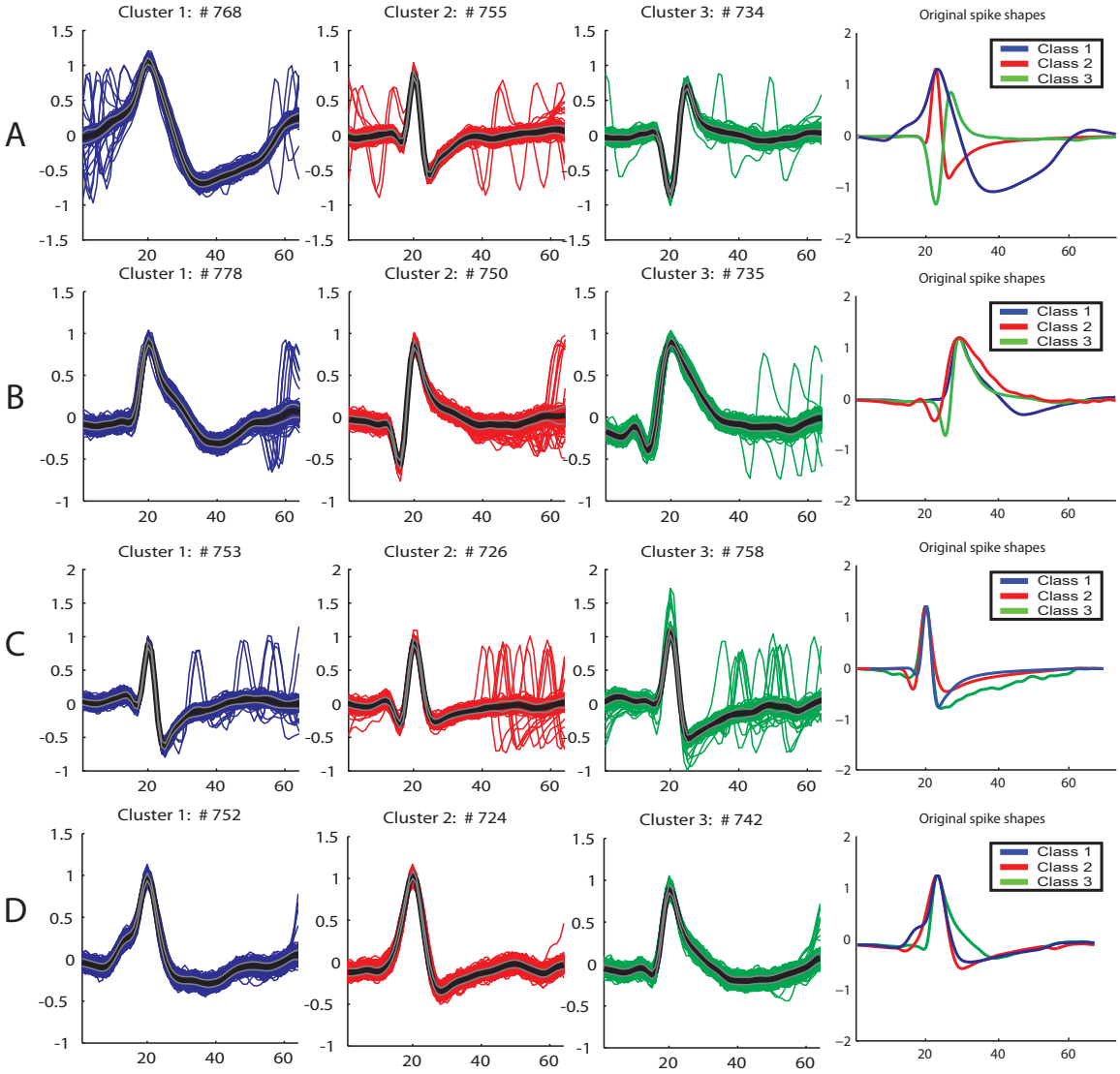


Figure 9.1: Results from four (A-D) simulated signals (using noise level 0.1) in **collection 1**. The first three columns shows the three different clusters, and the fourth column shows the original spike shapes in each signal for reference.

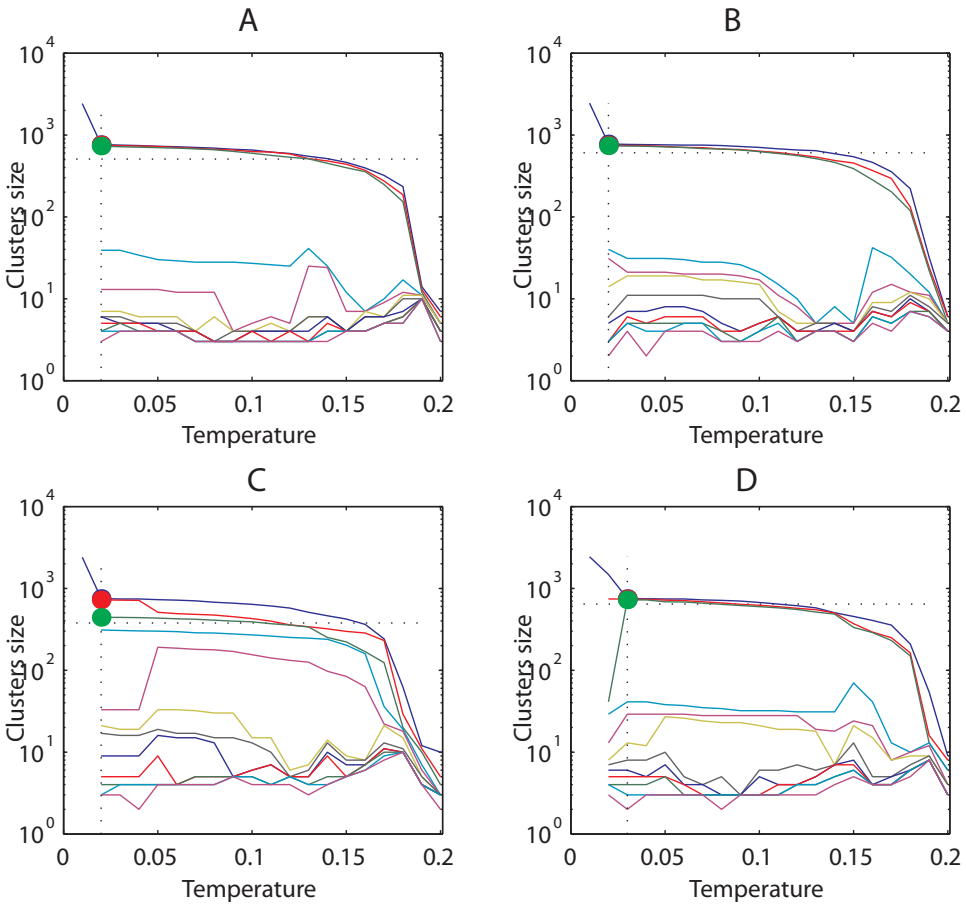


Figure 9.2: Temperature setting of four simulated signals (signal 2,6,10,14 in table 9.2) using noise level 0.1 in collection 1. The algorithm automatically sets the temperature to a level corresponding the superparamagnetic regime, with clustering of three spike shapes into relatively large clusters.

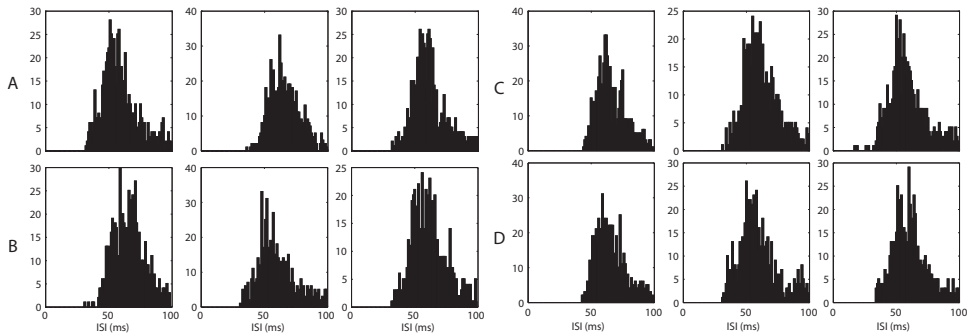


Figure 9.3: The distributions of inter-spike intervals in the three clusters for the four simulated signals (signal 2,6,10,14 in table 9.2) using noise level 0.1 in collection 1.

Spike detection performance

The detection results from Wave_Clus using collection 2 are presented in table 9.3. Both reference data and detection results are presented, and to express detection performance, detection misses and false positives are listed (including the number of single spikes and overlapping spikes).

Collection 2 - WC		Reference		Detection results		
Simulated signal	#	Noise level	No. spikes	Misses		False positives
			spikes (overlaps)	total / %	single / overlap	
1_easy	1	0.05	3514 (785)	210 [6.0 %]	17/193	711
	2	0.10	3522 (769)	179 [5.0 %]	2/177	57
	3	0.15	3477 (784)	360 [10.4 %]	145/215	15
	4	0.20	3474 (796)	989 [28.5 %]	714/275	10
2_easy	9	0.05	3410 (791)	174 [5.1 %]	0/174	0
	10	0.10	3520 (826)	191 [5.4 %]	0/191	2
	11	0.15	3411 (763)	183 [5.3 %]	10/173	1
	12	0.20	3526 (811)	632 [17.9 %]	376/256	5
3_diff	13	0.05	3383 (767)	211 [6.2 %]	1/210	63
	14	0.10	3448 (810)	191 [5.5 %]	0/191	10
	15	0.15	3472 (812)	211 [6.0 %]	8/203	6
	16	0.20	3414 (790)	403 [11.8 %]	184/219	2
4_diff	17	0.05	3364 (829)	182 [5.4 %]	0/182	1
	18	0.10	3462 (720)	152 [4.4 %]	0/152	5
	19	0.15	3440 (809)	189 [5.5 %]	3/186	4
	20	0.20	3493 (777)	490 [14.0 %]	262/228	2

Table 9.3: Results from spike detections using the Wave_Clus method and collection 2. The two uppermost columns separate the reference data and the detection results. In the "misses" column, the total number of misses is listed together with the number of ordinary "single-spike" misses and missed overlapped spikes. In the "False positives" column, the total number of false positives is presented, due to a low threshold.

In general, the detection performance for Wave_Clus are high, and the percentage of misses are low, except the cases (3,4,12,16,20 in table 9.3) with high noise levels. On the other hand, the number of false positives is high in the case (1 in table 9.3) of low noise (low detection threshold). The balance between misses and false positives is a trade-off in threshold level, but a majority of false positives is preferable, because a cluster of "double detections" can be disregarded in the later clustering procedure.

Clustering performance

The clustering results from Wave_Clus using collection 2 are presented in table 9.4.

Collection 2 - WC		Reference		Detection results	Clustering results (Classification errors)		
Simulated signal	#	Noise level	No. spikes spikes (overlaps)	No. spikes	Wavelets No. errors / %	PCA No. errors / %	Total success Wavelets / PCA
1_easy	1	0.05	3514 (785)	3304	2 [0.1 %]	3 [0.1 %]	94.0 % / 93.9 %
	2	0.10	3522 (769)	3343	6 [0.2 %]	21 [0.6 %]	94.7 % / 94.3 %
	3	0.15	3477 (784)	3117	7 [0.2 %]	23 [0.7 %]	89.4 % / 89.0 %
	4	0.20	3474 (796)	2485	16 [0.6 %]	156 [6.3 %]	71.1 % / 67.0 %
2_easy	9	0.05	3410 (791)	3236	5 [0.2 %]	6 [0.2 %]	94.8 % / 94.7 %
	10	0.10	3520 (826)	3329	12 [0.4 %]	845 [25.4 %]	94.2 % / 70.6 %
	11	0.15	3411 (763)	3228	54 [1.7 %]	2078 [64.4 %]	93.1 % / 33.7 %
	12	0.20	3526 (811)	2894	368 [12.7 %]	2149 [74.3 %]	71.6 % / 21.2 %
3_diff	13	0.05	3383 (767)	3172	2 [0.1 %]	10 [0.3 %]	93.7 % / 93.5 %
	14	0.10	3448 (810)	3257	49 [1.5 %]	2137 [65.6 %]	93.0 % / 32.5 %
	15	0.15	3472 (812)	3261	97 [3.0 %]	2098 [64.3 %]	91.1 % / 33.5 %
	16	0.20	3414 (790)	3011	781 [25.9 %]	2055 [68.2 %]	65.3 % / 28.0 %
4_diff	17	0.05	3364 (829)	3182	3 [0.1 %]	1572 [49.4 %]	94.5 % / 47.9 %
	18	0.10	3462 (720)	3310	11 [0.3 %]	1135 [34.3 %]	95.3 % / 62.8 %
	19	0.15	3440 (809)	3251	531 [16.3 %]	2060 [63.4 %]	79.1 % / 34.6 %
	20	0.20	3493 (777)	3003	1754 [58.4 %]	2078 [69.2 %]	35.8 % / 26.5 %
Average				3149	231 [7.3 %]	1152 [36.6 %]	84.4 % / 57.7 %

Table 9.4: Results from spike clustering using the Wave_Clus method and collection 2, with both wavelets and PCA for feature extraction. The classification errors are very high in almost all cases using PCA, and significantly lower using wavelets. The classification error is only high in cases using wavelets with highest noise levels.

The clustering process was performed with both wavelets and PCA for feature extraction, and the classification errors are shown for each simulated signal for wavelets and PCA respectively. The classification errors are very high in almost all cases using PCA, and significantly lower using wavelets. The classification error is only high in cases using wavelets with highest noise levels (12,16,20 in table 9.4).
Figure 9.4 shows the results from the four (A-D) simulated signals (using noise level 0.1) in collection 2. The classification errors is very low in these cases, according to table 9.4. The first three columns show the three clusters. Spike pairs appearing with a lower time separation than 0.5 ms (overlapping spikes) are not considered by Wave_Clus. The fourth column shows the original spike shapes in each signal for reference.

9.2 Human intra-cortical recordings

The results in this section are based on the human intra-cortical recordings (collection 3) described in section 7.2 page 41. Because of the absence of knowledge about the "true" firing pattern of each of the distinct neuron in the signal, these results using Wave_Clus are used as reference data in the comparison with the UBD method.

Due to "out-of-memory" issues and a lack of processing time, only 120 seconds of the signal in collection 3 are used in the following results.

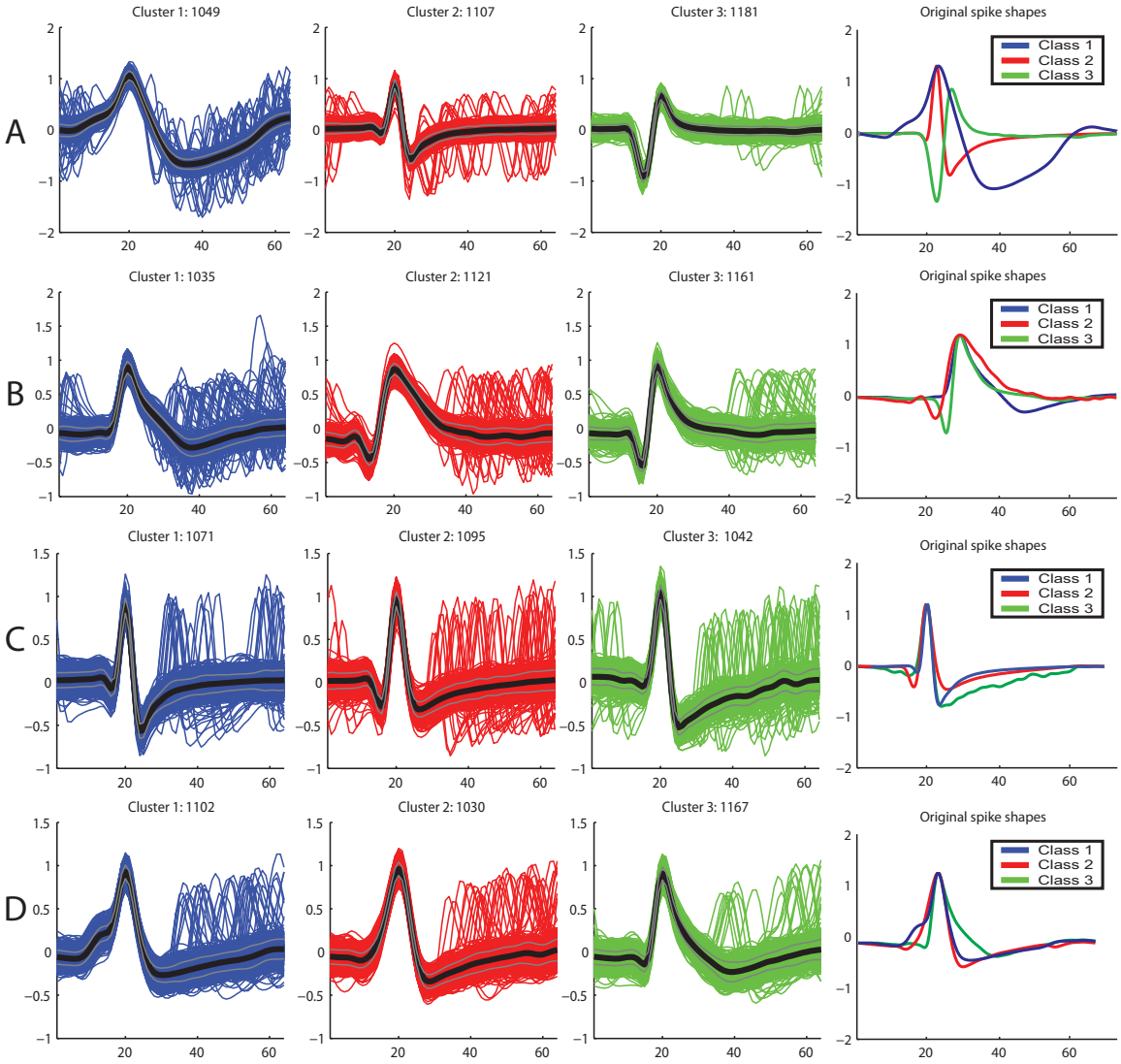


Figure 9.4: Results from the four (A-D) simulated signals (using noise level 0.1) in **collection 2**. The first three columns shows the three different clusters, and the fourth column shows the original spike shapes in each signal for reference.

The results from Wave_Clus (wavelets) using collection 3 are presented in table 9.5.

Collection 3 - WC		Reference		
Human signal	#	No. spikes	Neuron 1	Neuron 2
		spikes	spikes	spikes
Signal 1	1	740	609	131

Table 9.5: Results from spike detections and classification using Wave_Clus (wavelets) with collection 3. The firing patterns for the two neurons are used as reference in the comparison with the UBD method.

Only two spike classes was classified in the first 120 seconds of the 30 minutes signal, which corresponds to the first two clusters in figure 7.7 page 43. Figure 9.5 shows the clustering results from Wave_Clus together with the inter-spike intervals for each cluster.

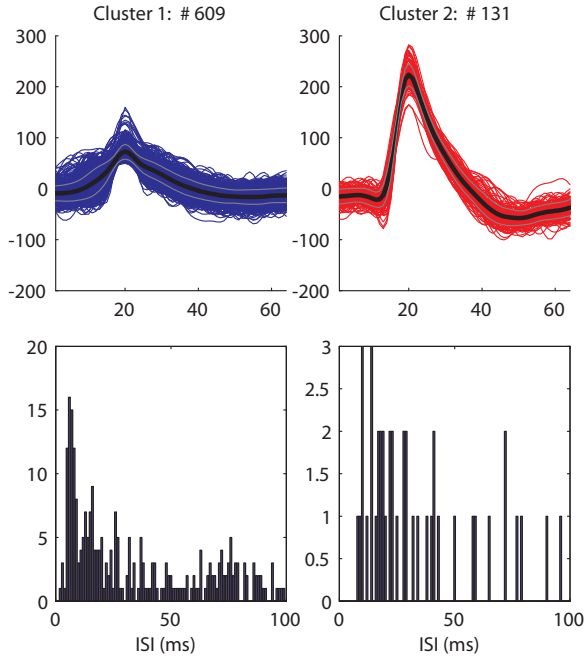


Figure 9.5: The clustering results from Wave_Clus together with the inter-spike intervals for each cluster.

Comparison of UBD and Wave_Clus

10

In this chapter, the results of the UBD method in focus using unsupervised Bayesian decomposition are compared with the Wave_Clus method described in chapter 6.

10.1 Collection 1

Table 10.1 summarizes the results for comparison using collection 1.

Comparison – Collection 1										
#	Missed		False positives		Classification errors			Total succes		
	UBD	WC	UBD	WC	UBD	WC - W	WC - PCA	UBD	WC - W	WC - PCA
1	0.4 %	9.1 %	53	32	3.0 %	8.4 %	20.3 %	96.6 %	83.3 %	72.4 %
2	1.0 %	10.3 %	86	31	4.5 %	6.9 %	32.8 %	94.4 %	83.6 %	60.3 %
3	11.4 %	11.2 %	284	3	12.1 %	7.8 %	22.9 %	77.8 %	81.9 %	68.5 %
4	13.8 %	11.4 %	531	0	28.1 %	7.4 %	27.6 %	62.0 %	82.1 %	64.2 %
5	0.8 %	12.7 %	41	27	4.0 %	5.8 %	21.3 %	95.2 %	82.2 %	68.7 %
6	2.3 %	9.4 %	107	14	6.1 %	7.6 %	30.9 %	91.7 %	83.8 %	62.6 %
7	9.4 %	10.0 %	356	11	14.6 %	7.2 %	63.5 %	73.4 %	83.5 %	32.8 %
8	14.4 %	10.1 %	402	5	30.1 %	8.6 %	62.3 %	59.8 %	82.2 %	33.9 %
9	1.0 %	12.9 %	31	28	2.8 %	3.4 %	30.0 %	96.2 %	84.1 %	60.9 %
10	1.9 %	11.1 %	70	7	7.3 %	6.9 %	39.7 %	90.2 %	82.9 %	53.6 %
11	9.1 %	11.1 %	210	9	10.8 %	6.4 %	52.2 %	81.1 %	83.2 %	42.5 %
12	14.9 %	10.6 %	387	13	27.3 %	7.0 %	72.5 %	61.9 %	83.1 %	24.6 %
13	2.4 %	14.2 %	87	45	8.7 %	7.0 %	16.6 %	89.2 %	79.8 %	71.6 %
14	3.7 %	9.9 %	191	17	9.8 %	8.9 %	42.6 %	86.8 %	82.1 %	51.8 %
15	9.3 %	9.8 %	344	5	16.8 %	7.9 %	61.6 %	75.5 %	83.1 %	34.7 %
16	15.5 %	11.2 %	577	9	31.4 %	0.5 %	71.0 %	58.0 %	88.3 %	25.7 %
μ	**7.0 %**	**10.9 %**	**235**	**16**	**13.1 %**	**6.7 %**	**41.8 %**	**80.9 %**	**83.1 %**	**51.8 %**

Table 10.1: Comparison of results from UBD and Wave_Clus using collection 1.

As seen, the UBD method has very few misses in the low noise cases, with an average of 7.0%. In contrast, Wave_Clus has relatively constant and high number of misses in all cases, with an average of 10.9%. This could be due to an amplitude threshold which is too high, or because of missed overlapping spikes.
The UBD method shows a low number (below 100) of false positives in low noise signals, and significantly more false positives than Wave_Clus in high noise cases. A high portion of the UBD false positives is false overlapping spikes detected and classified.
The UBD method shows a very low number (below 10%) of classification errors in most low noise cases, but a relatively high number of classification errors in high noise cases, with an average of 13.1%. In contrast, Wave_Clus using wavelets performs with a relatively high number of classification errors in all cases, but below the level of the high noise cases using UBD, with an average of 6.7%. Wave_Clus using PCA performs with a very high number of classification errors, especially in the high noise cases, with an average of 41.8%.
The total success was highest for the UBD method (with an average of 80.9%) in the two low noise cases for each signal set, whereas Wave_Clus using wavelets performs best seen from all 16 signals in general, with an average of 83.1%. The performance of Wave_Clus using PCA is significantly lower than the other two methods, with a mean of 51.8%.
An overview of the total success for all three methods are shown in figure 10.1.

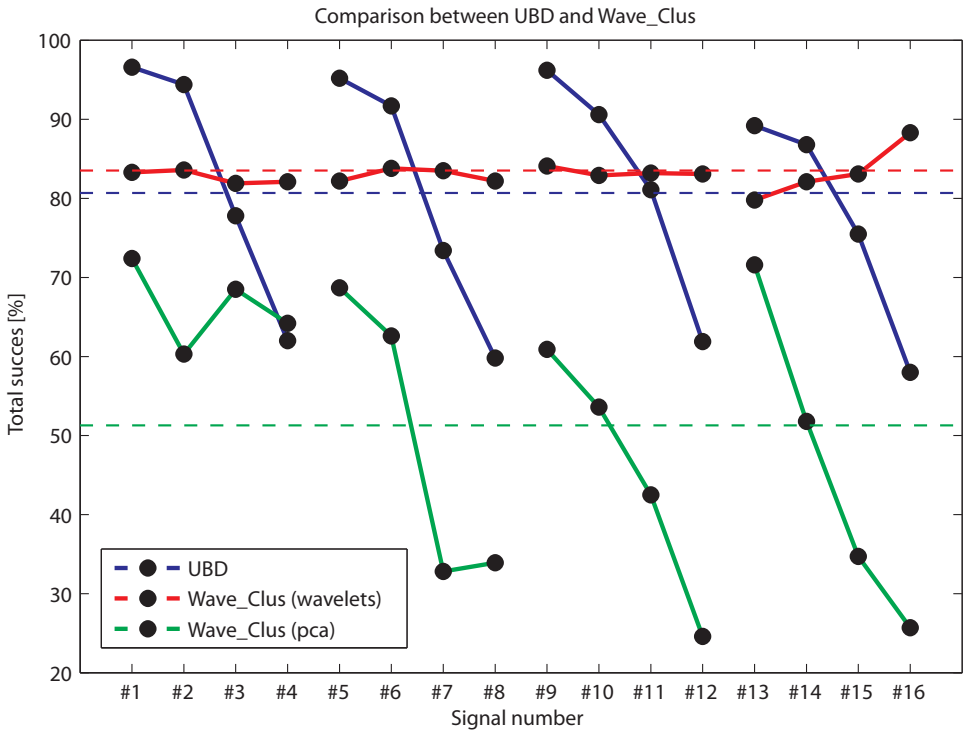


Figure 10.1: Summary of the total succes of each of the three methods, UBD, Wave_Clus (using wavelets and PCA) using collection 1.

10.2 Collection 2

Table 10.2 summarizes the results for comparison using collection 2.

Comparison – Collection 2										
#	Missed		False positives		Classification errors			Total success		
	UBD	WC	UBD	WC	UBD	WC - W	WC - PCA	UBD	WC - W	WC - PCA
1	39.1 %	6.0 %	432	711	16.1 %	0.1 %	0.1 %	44.8 %	94.0 %	93.9 %
2	36.8 %	5.0 %	381	57	20.9 %	0.2 %	0.6 %	42.3 %	94.7 %	94.3 %
3	46.9 %	10.4 %	539	15	15.3 %	0.2 %	0.7 %	37.8 %	89.4 %	89.0 %
4	62.7 %	28.5 %	581	10	9.1 %	0.6 %	6.3 %	28.2 %	71.1 %	67.0 %
5	40.9 %	5.1 %	233	0	14.0 %	0.2 %	0.2 %	45.1 %	94.8 %	94.7 %
6	33.4 %	5.4 %	634	2	20.6 %	0.4 %	25.4 %	46.0 %	94.2 %	70.6 %
7	38.4 %	5.3 %	672	1	21.1 %	1.7 %	64.4 %	40.5 %	93.1 %	33.7 %
8	57.7 %	17.9 %	599	5	9.4 %	12.7 %	74.3 %	32.9 %	71.6 %	21.2 %
9	41.3 %	6.2 %	482	63	12.5 %	0.1 %	0.3 %	46.2 %	93.7 %	93.5 %
10	43.2 %	5.5 %	492	10	10.7 %	1.5 %	65.6 %	46.1 %	93.0 %	32.5 %
11	50.1 %	6.0 %	528	6	12.0 %	3.0 %	64.3 %	37.9 %	91.1 %	33.5 %
12	62.2 %	11.8 %	603	2	7.5 %	25.9 %	68.2 %	30.4 %	65.3 %	28.0 %
13	41.7 %	5.4 %	781	1	14.7 %	0.1 %	49.4 %	43.6 %	94.5 %	47.9 %
14	40.0 %	4.4 %	488	5	19.4 %	0.3 %	34.3 %	40.6 %	95.3 %	62.8 %
15	56.5 %	5.5 %	584	4	14.1 %	16.3 %	63.4 %	29.4 %	79.1 %	34.6 %
16	59.8 %	14.0 %	631	2	14.0 %	58.4 %	69.2 %	26.2 %	35.8 %	26.5 %
μ	**46.9 %**	**8.9 %**	**541**	**56**	**14.5 %**	**7.3 %**	**36.6 %**	**38.6 %**	**84.4 %**	**57.7 %**

Table 10.2: Comparison of results from UBD and Wave_Clus using collection 2.

The UBD method shows a very high number of missed spikes, with an average of 46.9% compared to Wave_Clus with an average of 8.9%. Both methods shows a noise dependence in the number of missed spikes.
The same pattern is seen for the number of false positives in the two methods, except for signal 1. The number of classification errors for the UBD method, is twice as high as with Wave_Clus using wavelets, with an average of 14.5% compared to 7.3%. Wave_Clus using PCA still shows the highest number of classification errors with an average of 36.6%.
The total success was highest for Wave_Clus using wavelets, with an average of 84.4%. The UBD method shows a very low total success using collection 2, with an average of 38.6%.
In general, the UBD method shows significantly lower performance than Wave_Clus using collection 2, which is discussed in chapter 11 page 69.

An overview of the total success for all three methods are shown in figure 10.2.

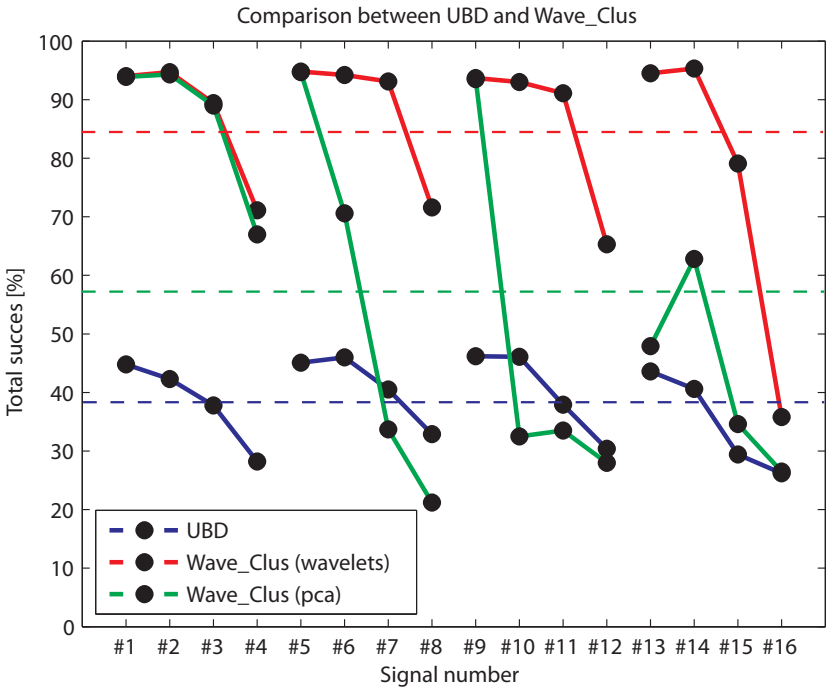


Figure 10.2: Summary of the total succes of each of the three methods, UBD, Wave_Clus (using wavelets and PCA) using collection 2.

10.3 Collection 3

Because the results from Wave_Clus using collection 3 is used as reference data for the UBD method, the comparison is seen together with the results in section 8.2 page 52.

Synthesis

IV

Discussion 11

Spike sorting is an important part of electrophysiological analysis of neural activity at the level of a single neuron. In this present work, the UBD method, which provides fully unsupervised spike sorting of intra-cortical recordings, has been described and validated with both simulated and real human intra-cortical recordings. Furthermore, the performance of the UBD method has been compared with the performance of another unsupervised spike sorting method, Wave_Clus, using classical spike sorting approaches.

Test data

For the validation of the two methods, different intra-cortical test data has been used. To obtain a quantitative measure of the performance of both UBD and Wave_Clus, the testing was initially performed with simulated signals with different noise levels and spike shapes.

In collection 1, designed and simulated especially for this present work, 16 synthetic intra-cortical recordings were generated. The number of distinct spike shapes, noise characteristics, and refractory period was designed to mimic real intra-cortical signals, and fully overlapping spikes was allowed. Further realistic details, such as varying spike amplitude, bursting neurons, more neurons, could be added to the signals in future work, to complicate the spike sorting process. The inter-spike interval distributions in collection 1 had a Gaussian-like shape, which suited the assumptions in the UBD method. It can be discussed whether the Gaussian shape approximation are valid in real intra-cortical recordings.

In collection 2, 16 synthetic intra-cortical recordings designed by Quiroga et al. [2004], was applied. The inter-spike interval distribution of these signals had a Poisson distribution and a short refractory period of 2 ms. This results in more than three spikes per segment for the UBD method, which causes a high number of missed spikes, and classification errors. These aspects of the UBD method must be improved in future work, and a re-tuning of the TABU algorithm is required to allow more than three spikes per segment. These issues are also stated in Ge et al. [2009]. The noise was generated by superposition of a large number of small-amplitude spikes, resembling characteristics of real recordings, which made further spike sorting complications using collection 2.

In collection 3, one human intra-cortical recording was used for testing. This allowed the methods to be validated with a real data set, as a contrast to high amount of synthetic signals. The inter-spike interval distribution for the human intra-cortical signal appeared approximately as a truncated Gaussian distribution, which fitted the assumptions for the UBD method.

The results

The UBD method allowed spike sorting with intra-cortical recordings, because of a parameter tuning of the minimum refractory period and the regularity in the spike discharge patterns. The UBD method performed spike sorting without manual interaction, and reached an accuracy of approximately 80.9 % using collection 1 with different spike shapes and noise levels. That is, however, with a relatively high number of spike misses and false positives in high noise signals, but with a peak performance of 96.6 % low noise signal 1. Wave_Clus showed a slightly better performance with collection 1 using wavelets, with an average of 83.1 %, but with low performance of 51.8 % using PCA.

The UBD method reached an average accuracy of 38.6 % using collection 2, compared with 84.4 % for Wave_Clus using wavelets, and 57.7 % for Wave_Clus using PCA. The very low performance for the UBD method using collection 2, was primarily because of the high number of spike misses, caused by a low refractory period of only 2 ms. The number of classification errors for the UBD method was an average of 14.5 %, compared to Wave_Clus with an average of 7.3 % using wavelets. Furthermore, another reason for the low performance using collection 2 could be because of the way in which the discharge statistics was generated in collection 2. The Poisson distributed inter-spike intervals did not resemble the real situation, nor the discharge statistics for both collection 1 and 3, which both provided high performance using the UBD method.

The UBD method reached an accuracy of 83.2 % using human intra-cortical recordings in collection 3. Wave_Clus provided reference information, which was problematic. It would have been better with a human expert providing the "true" information about spike shape and firing patterns, which would imply that a comparison with Wave_Clus was possible. This important point must be considered in future work, so that the comparison shows which method performs best in real conditions. A very important difference between the UBD method and Wave_Clus using collection 3, was the fact that Wave_Clus ignores the fully overlapping spikes, which caused a lot of missed spikes in the UBD results for collection 3, and subsequent analysis showed that the UBD method made correct detection and classification of the overlapping spikes in several cases using human intra-cortical recordings.
An example of the discharge statistics for human intra-cortical recordings are seen in figure 11.1. It is seen that the discharge statistics for collection 2 is very different from the statistics for collection 1 and 3, which can explain the low performance from the UBD method using collection 2. The difference makes the results from collection 2 not very representative.

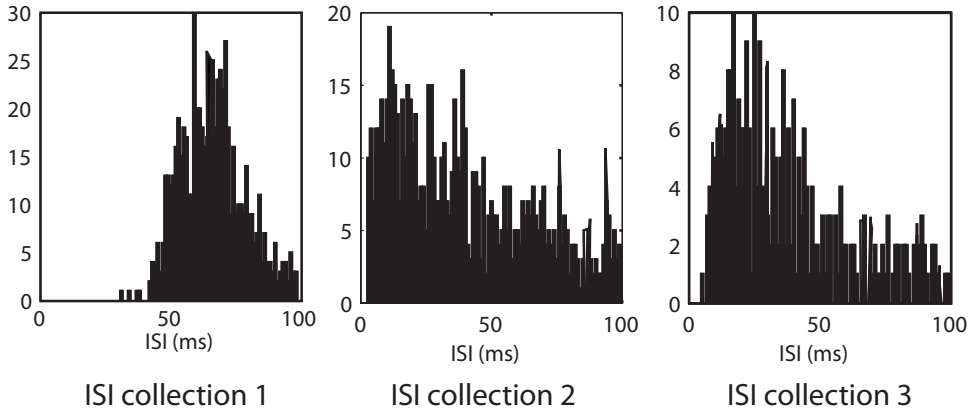


Figure 11.1: Example of inter-spike interval distributions from collection 1,2 and 3.

The comparison between UBD and Wave_Clus

The UBD method, using the MAP estimator, provided an efficient and automatically algorithm to solve the spike sorting problem using intra-cortical recordings. Originally the method was designed with a Bayesian statistical model on the EMG data generation process, but showed high performance using the same probabilistic model with intra-cortical recordings. The model allowed the inclusion of available prior information, including the Gaussian-like distribution of the inter-spike intervals, the refractory period, and the regularity in the spike firing patterns.
The inter-spike intervals for the intra-cortical recordings, showed also Gaussian-like distributions, which justified the use of the model. This was shown for the simulated data (collection 1) in figure 9.3 page 58, and for the human intra-cortical recordings (collection 3) in figure 9.5 page 62, which showed only approximated truncated Gaussian shaped distributions.
The refractory period was tuned for intra-cortical recordings, and was set to 10 ms for collection 1, and to 2 ms for collection 2 and 3. The regularity in the spike firing patterns was, due to the use of intra-cortical recordings, set to 0.8, which made this prior assumption very weak. This weakness may have decreased the spike sorting performance of the UBD method.

In comparison, Wave_Clus does not assume any specific distribution of the data, which makes this approach more robust to alternations in inter-spike intervals. In almost every case, Wave_Clus using PCA performed at a significantly lower level than both the UBD method and Wave_Clus using wavelets. The main drawback with PCA is that eigenvectors accounting for the largest variance of the data are selected, but these directions do not necessarily provide the best separation of the spike classes.

A large difference in the processing time was observed between the UBD method and Wave_Clus. The decomposition of each 60 seconds signal with the UBD method took approximately 3 hours, compared with 5 minutes for Wave_Clus. In contrary, the prolonged processing time for the UBD method was caused by the resolving of superimpositioned spikes, which is a significant advantage of the UBD method compared to Wave_Clus.
Furthermore, the UBD method is based on the modeling of a single-channel intra-cortical recording, and a multichannel extension may improve the spike sorting performance by exploiting the inter-channel interference. In comparison, Wave_Clus has the ability of using polytrodes.

Conclusion 12

Throughout this report, the spike sorting performance level of the UBD method has been investigated with both simulated and human intra-cortical recordings. A literature study in form of a spike sorting review was conducted, and the classical spike sorting method Wave_Clus was chosen for comparison with the UBD method.

The UBD method in focus in the present work, provided spike sorting of intra-cortical recordings in a fully automatic way using a Bayesian framework, without making any assumptions on the particular spike shape of the action potential, and can be applied to intra-cortical recordings as well as intra-muscular EMG signals, because of proper tuning as done in this report.

Throughout the validation and testing of the UBD method, different test data was used. Simulated intra-cortical recordings (collection 1 and collection 2) and human intra-cortical recordings (collection 3) was used as input to the UBD method and to Wave_Clus. Collection 1 was designed and simulated for this present work, collection 2 and 3 was provided from Quiroga et al. [2004].

Using collection 1, the UBD method reached a performance of 80.9 %, compared 83.1 % using Wave_Clus (wavelets) and 51.8 % using Wave_Clus (PCA). Using collection 2, the UBD method only reached a performance of 38.6 %, compared to 84.4 % using Wave_Clus (wavelets) and 57.7 % using Wave_Clus (PCA). A short neuronal refractory period of 2 ms in the simulated signals in collection 2, and a questionable representative discharge statistics in collection 2 causes the detection problems. Using collection 3, the UBD method reached a performance of 83.2 % compared directly to Wave_Clus as reference, but the UBD method resolved a high amount of overlapping spikes compared to Wave_Clus with human data, which was a clear advantage.

In general, the UBD method is capable of performing spike sorting using both simulated and real human intra-cortical recordings, and showed strong capabilities in resolving overlapping spikes, but with some weaknesses as shown. Further development must be done, to increase the performance of the UBD method with intra-cortical recordings. Among these, a re-tuning of the TABU algorithm to enable the detection and classification of more than three spikes per segment. Furthermore, a multi-channel extension to the UBD method will improve the performance by exploiting the inter-channel inference.

Bibliography

Adamos, D. A., Kosmidis, E. K. & Theophilidis, G. [2008], 'Performance evaluation of pca based spike sorting algorithms', *Computer Methods and Programs in Biomedicine* **91**(3), 232–244.

Aksenova, T. I., Chibirova, O. K., Dryga, O. A., Tetko, I. V., Benabid, A.-L. & Villa, A. E. P. [2003], 'An unsupervised automatic method for sorting neuronal spike waveforms in awake and freely moving animals', *Journal of Neuroscience Methods* **30**(2), 178–187.

Atiya, A. [1992], 'Recognition of multiunit neural signals', *IEEE Transactions on Biomedical Engineering* **39**(7), 723–729.

Bar-Hillel, A., Spiro, A. & Stark, E. [2006], 'Spike sorting: Bayesian clustering of non-stationary data', *Journal of Neuroscience Methods* **157**(2), 303–316.

Blatt, M., Wiseman, S. & Domany, E. [1996], 'Superparamagnetic clustering of data', *Physical Review Letters* **76**(18), 3251–3254.

Brown, E., Kass, R. & Mitra, P. [2004], 'Multiple neural spike train data analysis: state-of-the-art and future challenges.', *Nature Neuroscience* **7**(5), 456–461.

Delescluse, M. & Pouzat, C. [2005], 'Efficient spike sorting of multi state neurons using inter spike intervals information', *Journal of Neuroscience Methods* **150**, 16–29.

Fee, M. S., Mitra, P. P. & Kleinfeld, D. [1996*a*], 'Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non gaussian variability', *Journal of Neuroscience Methods* **69**(2), 175–188.

Fee, M. S., Mitra, P. P. & Kleinfeld, D. [1996*b*], 'Variability of extracellular spike waveforms of cortical neurons', *Journal of Neurophysiology* **76**(6), 3823–3833.

Fried, I., MacDonald, K. A. & Wilson, C. L. [1997], 'Single neuron activity in human hippocampus and amygdala during recognition of faces and objects', *Neuron* **18**(5), 753–765.

Ge, D., Carpentier, E. L. & Farina, D. [2009], 'Unsupervised bayesian decomposition of multi-unit emg recordings using tabu search', *IEEE Transactions on Biomedical Engineering* . In Press.

Gold, C., Henze, D., Koch, C. & Buzsáki, G. [2006], 'On the origin of the extracellular action potential waveform: A modeling study', *Journal of Neurophysiology* **95**(5), 3113–3128.

Gray, C. M., Maldonado, P. E., Wilson, M. & McNaughton, B. [1995], 'Tetrodes markedly improve the reliability and yield of multiple single-unit isolation from multi-unit recordings in cat striate cortex', *Journal of Neuroscience Methods* **63**(1-2), 43–54.

Harris, K. D., Henze, D. A., Csicsvari, J., Hirase, H. & Buzsáki, G. [2000], 'Accuracy of tetrode spike separation as determined by simultaneous intracellular and extracellular measurements.', *Journal of Neurophysiology* **84**(1), 401–414.

Herbst, J. A., Gammeter, S., Ferrero, D. & Hahnloser, R. H. [2008], 'Spike sorting with hidden markov models', *Journal of Neuroscience Methods* **174**(1), 126–134.

Horton, P., Nicol, A., Kendrick, K. & Feng, J. [2007], 'Spike sorting based upon machine learning algorithms soma', *Journal of Neuroscience Methods* **160**(1), 52–68.

Hulata, E., Segev, R. & Ben Jacob, E. [2002], 'A method for spike sorting and detection based on wavelet packets and shannons mutual information', *Journal of Neuroscience Methods* **117**(1), 1–12.

Lewicki, M. S. [1994], 'Bayesian modeling and classification of neural signals', *Neural Computation* **6**(5), 1005–1030.

Lewicki, M. S. [1998], 'A review of methods for spike sorting: the detection and classification of neural action potentials', *Network: Computation in Neural Systems* **9**(4), 53–78.

Madany, Sharp, H., Menne, K., Hofmann, U. & Martinetz, T. [2005], 'Unsupervised spike sorting with ica and its evaluation using genesis simulations', *Neurocomputing* **65-66**, 275–282.

Mallat, S. G. [1989], 'A theory for multiresolution signal decomposition: the wavelet representation', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **11**(7), 674–693.

Pouzat, C., Mazor, O. & Laurent, G. [2002], 'Using noise signature to optimize spike-sorting and to assess neuronal classification quality', *Journal of Neuroscience Methods* **122**(1), 43–57.

Quiroga, R. [2009], 'What is the real shape of extracellular spikes?', *Journal of Neuroscience Methods* **177**, 194–198.

Quiroga, R., Nadasdy, Z. & Ben-Shaul, Y. [2004], 'Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering', *Neural Computation* **16**(8), 1661–1687.

Schmidt, E. M. [1984], 'Computer separation of multi-unit neuroelectric data: a review', *Journal of Neuroscience Methods* **12**(2), 95–111.

Shoham, S., Fellows, M. R. & Normann, R. A. [2003], 'Robust, automatic spike sorting using mixtures of multivariate t-distributions', *Journal of Neuroscience Methods* **127**(2), 111–122.

Song, M. & Wang, H. [2006], 'A spike sorting framework using nonparametric detection and incremental clustering', *Neurocomputing* **69**(10-12), 1380–1384.

Vargas-Irwin, C. & Donoghue, J. P. [2007], 'Automated spike sorting using density grid contour clustering and subtractive waveform decomposition', *Journal of Neuroscience Methods* **164**(1), 1–18.

Wood, F. & Black, M. [2008], 'A nonparametric bayesian alternative to spike sorting', *Journal of Neuroscience Methods* **173**(1), 1–12.

Wood, F., Fellows, M., Donoghue, J. P. & Black, M. J. [2004], Automatic spike sorting for neural decoding, *in* 'Proc. the 26th Annual Internaltional Conference of the IEEE EMBS', pp. 4009–4012.

Zhang, P., Wu, J., Zhou, Y., Liang, P. & Yuan, J. [2004], 'Spike sorting based on automatic template reconstruction with a partial solution to the overlapping problem', *Journal of Neuroscience Methods* **135**(1-2), 55–65.