Classification of bird syllables in noisy environments using multitapers

Olof Zetterqvist

Abstract

A method for syllable classification of the Great Reed Warbler (Acrocephalus arundinaceus) has been studied and tested. This method uses multitapers in order to calculate the ambiguity domain and extracting features. Inspired by this technique four new methods, that build on extracting features from the ambiguity function using different types of multitaper kernels and noise reduction techniques, have been developed. These methods use different kinds of kernels and multitapers in order to change the focus of the method.

A smaller study is made of how the multitaper windows behave with different kernels and how to solve problems that can occur. The methods are validated in different noise levels on a simulated data set, where especially difficult cases are simulated, and on real datasets with extra noise added. From this study, it is found that kernels which retain the cross-terms and suppress the auto-terms are harder to adjust but may detect smaller differences in the signals. However, these methods lack robustness in noisy environments. On the other hand, methods that focus on the auto-terms are more noise robust but cannot detect smaller differences.

In order to make the methods more robust, different noise reduction techniques are created and validated. these techniques make the methods more robust but will lose accuracy in the classification.

Popular Abstract

Klassificering av fågelsång i brusiga miljöer: ett val mellan noggrannhet och robusthet

I många sammanhang är det intressant att klassificera och analysera fågelsång. I detta projekt studeras olika automatiska klassificeringsmetoder, som lyfter fram olika viktiga aspekter för en klassificering så som robusthet och noggrannhet. Beroende på förutsättningarna vid inspelningen, samt syftet med studien, kommer en lämplig sorteringsmetod behöva väljas.

Att studera fågelsång är av många olika aspekter intressant och används ofta för att förstå sig på fåglars beteende och sociala spel. I analysen studeras det bland annat hur många olika läten de använder de sig av, hur mycket deras sång påverkas av grannfågeln och om det förekommer dialekter mellan fåglar inom samma art. Under dessa analyser utför man ofta en sortering och klassificering av fågelns olika läten för att få en förståelse för hur sången ser ut. Tyvärr finns det idag inga tillräckligt bra automatiserade metoder för att göra en sådan klassificering. Att utföra ett sådant arbete för hand innebär mycket arbetstid och resurser, inte minst på grund av att inspelningar ofta är gjorda i fåglarnas naturliga miljöer, vilket leder till att inspelningarna kommer påverkas av brus av olika slag. Bruset kan till exempel vara i form av störande vind eller porlande läte från vattenfall eller bäck. Att utföra och utvärdera en sådan klassificering är därför svårt. Många experter är ofta inte överens om när två läten bör klassificeras som lika eller olika och hur många olika sorters läten som finns i en sång. Detta innebär att studiens resultat påverkas av noggrannheten i klassificeringen och kan variera mycket beroende på vem som gör analysen. För att underlätta analysen studeras i detta projekt metoder för att göra en automatiserad klassificering av läten i en sång. Detta för att göra processen snabbare och ge ett mer konsekvent resultat. Studien har utgått från en metod publicerad 2016, vilken har lagt grunden för ytterligare fyra metoder som utvecklats under arbetets gång. Dessa nya metoder fokuserar på olika teoretiska principer som ger olika egenskaper vid klassificeringen. Vissa är konstruerade för att kunna ta hand om störningar i inspelningen och bli mer robusta mot olika nivåer av brus. Andra är konstruerade för att upptäcka mindre skillnader i signalerna och kunna få en mer noggrannhet i klassificeringen. Studien visar att dessa nya metoder uppfyller de mål de var konstruerade för men uppvisar vissa nackdelar. Bland annat påverkas noggrannheten i klassificeringen mycket av brusreduceringsmetoderna. Metoderna som fokuserar på att hantera brus lyckas inte skilja mellan mindre skillnader i lätena och kan klassificera två snarlika läten som samma. Dessa metoder beter sig också sämre för inspelningar med låga brusnivåer och kan ge felaktiga resultat. Metoder som fokuserar på noggrannhet däremot, påverkas lätt av brus och kan därför bli svåra att använda i en brusig miljö. Att använda sig av en automatiserad klassificeringsteknik blir därför en avvägning mellan robusthet och noggrannhet och valet av metod kommer att behöva bestämmas beroende på sånginspelningen.

Contents

1 Introduction

Analyzing and classifying different bird songs have many great applications and is an area that is constantly developing. Not only because of the similar behaviour to human languages but also to analyzing bird behaviour. Many birds use their song in order to find a partner and as a competitive tool to advertise themselves to the other sex. Here it is not the size of the repertoire or the complexity of the song that put the bird in advantage but instead the ability to learn new phrases (Kroodsma 2017). Not only do the birds learn new phrases from year to year to expand their repertoire (Wegrzyn 2010) but also, as in the case of humans, have dialects that vary from habitat to habitat (Wayne 2017).

It is often interesting to compare different syllables in the song. In the case of comparing repertoire size, meaning the number of different syllables existing in the song, each syllable in a song is classified and the number of classes is counted. In order to describe and analyze the variations between bird songs you need to have a large data set. The analyzes of these sets are often made by hand which is very time consuming.

A natural approach is to develop methods were the classifications are made automatically. Many different methods have been proposed, all of them have advantages as well as disadvantages. Methods using neural networks have been showing many promising results when it comes to classification. However, these methods require a large amount of data in order to train. Since this is not accessible, neural networks are not an option in this thesis. Other approaches have been to use the features evaluated from the spectrogram, to use dynamic time warping and a random forest distance between spectrogram features (Keen et al. 2014).

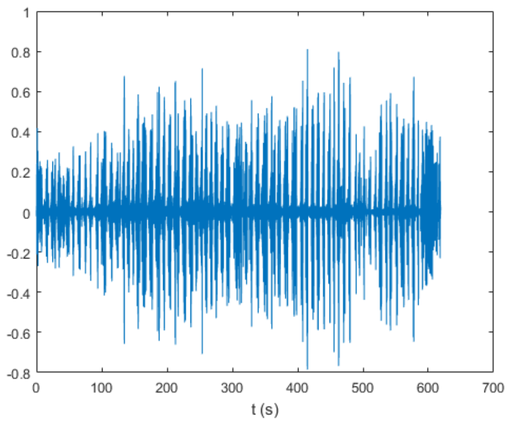
When doing these type of analyzes, experts in biology disagree on how to separate between two similar syllable classes and how large the repertoire size is. Are two syllables the same if they are identical except one of them phrases the ending down and the other phrases the ending up? This makes it hard to do a general automatic method and do an evaluation on real data.

The goal of this thesis is to proceed to work on the method described in Grosse Ruse et al. (2016) and make the method more noise robust. This method makes use of the ambiguity function and the singular value decomposition (SVD) in order to extract features for each syllable. In this thesis, four new methods, which are variations on the original one, are developed with the goal to be accurate in the classification and be more robust to noise. We will see how well these methods are able to separate between signals with very small differences and determine how sensitive they are in their classification. In order to have a method as a reference we compare these to a method based on mel-frequency cepstral coefficients which is the most commonly used in voice recognition today.

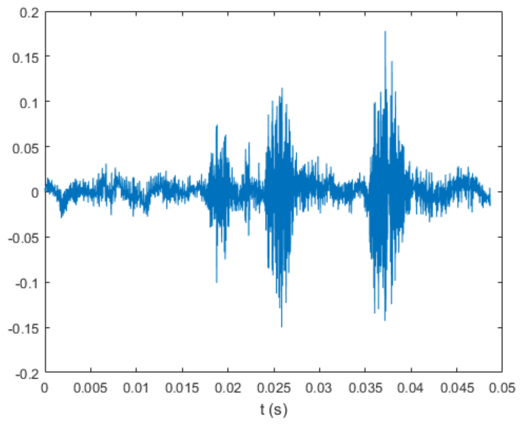
This thesis is organized as follows: In section 2, the used data sets, simulated as well as real, are presented. In section 3, we present the theory and define the spectrogram, Wigner distribution and the ambiguity domain. In section 5, we will introduce six different classification methods based on these domains. In Sections 7-8 the results are presented. We finish the thesis with a discussion and conclusion.

2 Description of the data

In this master thesis, I have studied recordings between year 1989 and 1991 from the Great Reed Warbler (GRW). Since these birds thrive near lakes the recordings are done from a canoe approximately five to six meters away from the bird. In figure 1 a) we can see a song from one of these recordings. When analyzing the bird song we will separate it into different phrases and syllables that will define the song. A phrase is a shorter part of the song that consists of a few sound and are separated by longer silent parts in the song. Each of these phrases consists of a series of syllables which will define the phrase. A typical syllable from the GRW is seen in figure 1 b). The important components in this syllable are the larger peaks that will define the sound of the syllable. The different syllables from the GRW can mainly be separated into two different sounds, whistles and rattles (Wegrzyn 2010). The whistles are often shorter and consist of few components while the rattles are often longer. However, the song consists of many different types of whistle and rattle sounds that we want to be able to separate. In this thesis, I have used the dataset used in Grosse Ruse et al. (2016) where the data already was separated into syllables.



(a) Whole song.



(b) One syllable in the song.

Figure 1: A typical song from the GRW.

2.1 Detection of a syllable in a song

To detect a syllable in a song we use a method described in Grosse Ruse et al. (2016) where the authors used two mean value filters P_{short} and P_{long} where P_{short} is of length corresponding to 90 ms and P_{long} corresponding to 360 ms. The syllable detection is done by solving

$$P_{short}(t) * (x(t)^2) > P_{long}(t) * (x(t)^2) + (1 - \frac{l_{sens}(t)}{100}) max(P_{long}(t) * (x(t)^2)) \tag{1}$$

where x is the signal where we want to detect the syllables, $*$ is the convolution and l_{sens} is a variable which determine the sensitivity of the algorithm. In figure 2, we can see an illustration of the algorithm where we see $x(t)^2$ together with the signal filtered with the corresponding filters. The detection occurs when the red line is higher than the yellow.

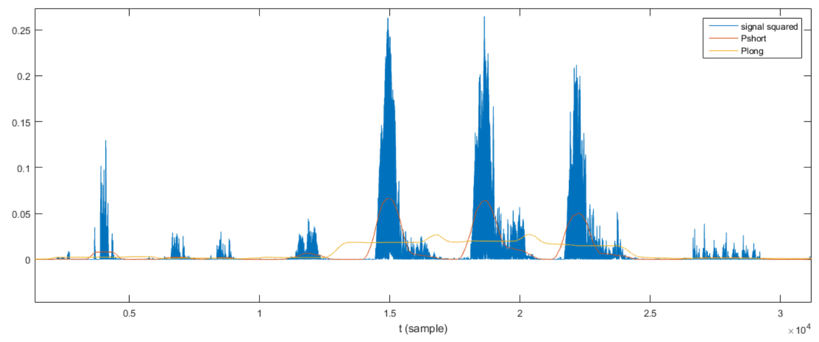


Figure 2: An illustration of the syllable detection algorithm where we see the signal squared (blue line) and its convolution with the short filter (red line) and the long filter (yellow line). The part where the red line is over the yellow will be considered as a syllable.

The choice of l_{sens} could be very hard since it will be a balance between precision and robustness. By increasing l_{sens} the method will be able to detect syllables with a lower amplitude, since the yellow line in figure 2 will be lower, but it will also be more sensitive to noise. This will be a choice that depends on the signal and its noise level.

2.2 Modeling of a syllable

In order to validate different classification methods and see how they will behave for different inputs, we will use simulated syllables that are similar to real data. As seen in figure 1 b), a syllable is often a nonstationary signal meaning that the mean value and standard deviation change over time. It's also noticeable that a syllable consists of a few peaks, in this case three, all containing different frequencies. To simulate this behavior we use simulated signals of the form

$$x(t) = \sum_{k=1}^{N} A_k e^{-\frac{(t-t_k)^2}{\alpha_k}} sin(2\pi f_k t + \theta_k) \tag{2}$$

where N is the number of pulses in the syllable, A_k the amplitude of the pulse k, t_k the location, α_k a variable that defines the width of the pulse, f_k the frequency and θ_k the phase. θ_k is often considered as a random variable that is uniformly distributed between $-\pi$ and π. Since our Gaussian function $e^{-\frac{(t-t_k)^2}{\alpha_k}}$ decreases fast when t goes to $\pm\infty$ the width of a pulse is easily determined with the parameter α_k. When simulating a larger set of syllables of the same type we also shift the peaks location and frequency in each syllable in order to get a more realistic representation of a real syllable set.

3 Theory

In order to analyze a syllable, we mainly use three different concepts: the spectrogram, the Wigner distribution and the ambiguity function. All these have strong connections to each other and will have a big impact in our analyzes. In this section, we will go through these concepts and how they behave in noise. The theory in this section (except section 3.5) could be found in Sandsten (2018).

3.1 The spectrogram

The spectrogram of a signal x is defined as

$$X(t, f) = \left| \int_{-\infty}^{\infty} x(s)h^*(s - t)e^{-i2\pi fs} ds \right|^2 \tag{3}$$

where h is a window function with compact support. Because of this, the spectrogram will be able to tell us how the spectral content in the signal x changes over time since it is basically a Fourier transform of a short part of the signal centered around a timepint t. The length of the window will determine how the spectrogram will behave since it limits the integral to include the part that is local to the time point t. In figure 3, we can see the spectrogram of a Gaussian pulse with a normalized frequency 0.3. As we can see, we get a pulse at the corresponding frequency. The behaviour of the spectrogram will not only depend on the length of the window h but also it shape. The shape of our window will determine what parts and relations in our signal that will be dominant. We will later study how to determine which windows to use for a specific application.

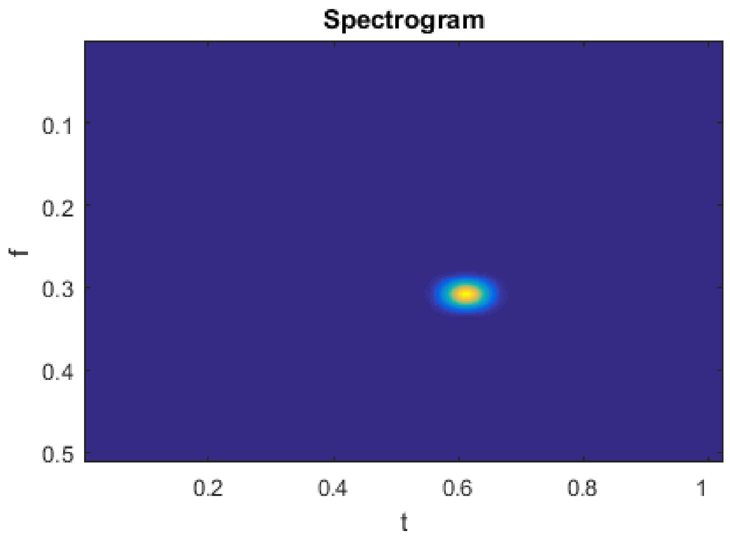


Figure 3: The spectrogram of a Gaussian pulse with a normalized frequency $f_1 = 0.3$, at time point $t_1 = 0.612$ with $\alpha_1 = 30$.

3.2 The Wigner distribution

In order to analyze a signal we could use the Wigner distribution (WD). The Wigner distribution of a continuous time signal is defined as

$$W(t, f) = \int_{-\infty}^{\infty} x(t + \frac{\tau}{2})x^*(t - \frac{\tau}{2})e^{-2i\pi f\tau} d\tau \tag{4}$$

where x is the signal and $*$ is the complex conjugate operator. Here, we can note that this is the same as the Fourier transform of the function

$$r(t, \tau) = x(t + \frac{\tau}{2})x^*(t - \frac{\tau}{2}) \tag{5}$$

also known as the instantaneous autocorrelation function (IAF). Note that due to the symmetry in this expression, a time shift in the signal will cause the same time shift to the instantaneous autocorrelation and therefore also to the Wigner distribution. Notice also that if we study the conjugate of the Wigner distribution and use the variable change $\tau = -\tau$ we can find

$$W_x^*(t,f) = (\int_{-\infty}^{\infty} x(t+\tfrac{\tau}{2})x^*(t-\tfrac{\tau}{2})e^{-2i\pi f\tau}d\tau)^* = \int_{-\infty}^{\infty} x^*(t+\tfrac{\tau}{2})x(t-\tfrac{\tau}{2})e^{2i\pi f\tau}d\tau = \\ \int_{-\infty}^{\infty} x(t+\tfrac{\tau}{2})x^*(t-\tfrac{\tau}{2})e^{-2i\pi f\tau}d\tau = W_x(t,f) \tag{6}$$

which indicates that the Wigner distribution is real valued. In figure 4, we can see the the Wigner distribution of a signal consisting of a single Gaussian pulse with normalized frequency 0.3. By comparing this to the spectrogram of the same pulse in figure 3, its noticeable that the Wigner distribution has a better concentration than the spectrogram which makes the Wigner distribution easier to use when analyzing the signal.

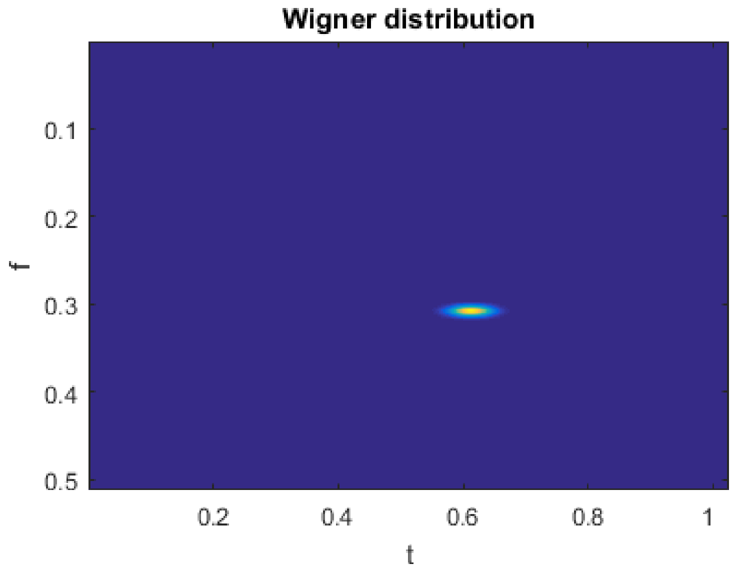


Figure 4: The Wigner distribution of a pulse with a normalized frequency of $f_1 = 0.3$, at time point $t_1 = 0.612$ with $\alpha_1 = 30$.

However, the Wigner distribution has a few disadvantages that spectrogram does not have. We will now study what happens when we use the Wigner distribution on a signal consisting of two components $x(t) = x_1(t) + x_2(t)$. The Wigner distribution becomes

$$W_x(t,f) = \int_{\infty}^{\infty} x(t+\tfrac{\tau}{2})x^*(t-\tfrac{\tau}{2})e^{-2i\pi f\tau}d\tau = \\ \int_{\infty}^{\infty}(x_1(t+\tfrac{\tau}{2}) + x_2(t+\tfrac{\tau}{2}))(x_1^*(t-\tfrac{\tau}{2}) + x_2^*(t-\tfrac{\tau}{2})e^{-2i\pi f\tau}d\tau = \\ \int_{\infty}^{\infty}(x_1(t+\tfrac{\tau}{2})x_1^*(t-\tfrac{\tau}{2}) + x_1(t+\tfrac{\tau}{2})x_2^*(t-\tfrac{\tau}{2}) + x_2(t+\tfrac{\tau}{2})x_1^*(t-\tfrac{\tau}{2}) + x_2(t+\tfrac{\tau}{2})x_2^*(t-\tfrac{\tau}{2}))e^{-2i\pi f\tau}d\tau = \\ W_{x_1} + W_{x_2} + \int_{\infty}^{\infty}(x_1(t+\tfrac{\tau}{2})x_2^*(t-\tfrac{\tau}{2}) + x_2(t+\tfrac{\tau}{2})x_1^*(t-\tfrac{\tau}{2}))e^{-2i\pi f\tau}d\tau \tag{7}$$

As we can see it shows up two terms W_{x_1} and W_{x_2} that are the same as the Wigner distribution of each component. These are called the auto-terms. It also occurs a part that consists of the interaction between the two parts of the signal. This part is called the cross-term and will be located midway between the two auto-terms in the Wigner distribution. In figure 5, we can see the Wigner distribution of a signal consisting of two Gaussian pulses with the normalized frequencies 0.3 and 0.2 at time points 0.7 and 0.1. As expected, it occurs a cross-term between the auto-terms that will interfere with our interpretation of the signal. Notice the oscillation in the cross-term. Since the cross-term does not have the same quadratic behavior as the auto-terms, the cross-term will have a lower amplitude. Cross-terms will occur between all combinations of auto-terms. We will later look at methods filtering out the cross-terms and auto-terms of the signal.

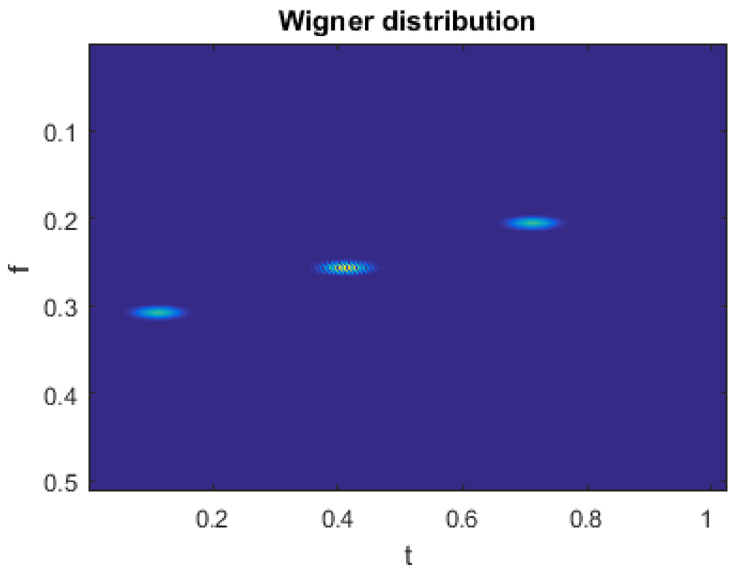


Figure 5: The Wigner distribution of two Gaussian pulses with a normalized frequency of 0.3 and 0.2. As we can see there occur a cross-term between the two pulses.

3.3 The discrete Wigner distribution

When looking at a discrete signal, the Wigner distribution is defined as

$$W(n,l) = 2 \sum_{m=min(n,N-1-n)}^{min(n,N-1-n)} x(n+m)x^*(n-m)e^{-2i\pi m\frac{l}{L}} \tag{8}$$

where N is the number of samples in the signal x and L is the number of frequency points. This definition will have the same properties as the continuous one in terms of cross-terms and time shift. But due to the symmetry of our expression and the discrete Fourier transform it will occur repetition of our signal in the frequency domain and as long we look at a real signal we will also see negative frequencies corresponding to the positive ones. This will lead to aliasing around $f = 0.25$.

To come around this problem, we will transform our signal to be able to remove the negative part of the spectrum of x. This is done with the Hilbert transform

$$z = x + iH(x) \tag{9}$$

where

$$H(x) = \mathbf{F}^{-1}(-isign(f)\mathbf{F}(x)) \tag{10}$$

Here, $\mathbf{F}$ is the Fourier transform and $\mathbf{F}^{-1}$ the inverse Fourier transform. This transformation of the signal x will remove the negative frequencies that will result in aliasing.

3.4 The ambiguity function

The ambiguity function (AMF) is defined as

$$A_x(\nu, \tau) = \int_{-\infty}^{\infty} x(t + \frac{\tau}{2})x^*(t - \frac{\tau}{2})e^{-2i\pi\nu t}dt = \int_{-\infty}^{\infty} r_x(t, \tau)e^{-2i\pi\nu t}dt \tag{11}$$

where $r_x(t, \tau)$ is the IAF function of the signal x. By knowing the Wigner distribution, we can calculate the ambiguity function by

$$A_x(\nu, \tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_x(t, f)e^{-2i\pi t\nu + 2i\pi f\tau}dtdf \tag{12}$$

The ambiguity function has some really nice properties that are useful when analyzing a signal. The first one is that when shifting the signal in time the absolute value of the ambiguity domain will stay the same. This is shown by calculating the ambiguity for $x(t - t_0)$ and use the variable change $t_1 = t - t_0$. Since the integration limits is from $-\infty$ to ∞, the integral will stay the same.

$$|A_{x(t-t_0,f)}(\nu, \tau)| = |\int_{-\infty}^{\infty} r_x(t - t_0, \tau)e^{-2i\pi\nu t}dt| = |\int_{-\infty}^{\infty} r_x(t_1, \tau)e^{-2i\pi\nu t_1}dt_1| = |A_x(t, f)(\nu, \tau)| \tag{13}$$

When shifting the signals frequencies only the argument of the ambiguity will change leaving the amplitude the same. This means that the amplitude of the ambiguity function will stay the same for a signal independently where it is located and in which frequency band its in. The amplitude of the ambiguity function will only change if the relations in the signal changes.

For a signal with a single component, the signal will always be centered at $\nu = 0$, $\tau = 0$. In fact when analyzing a multicomponent signal we will see that the auto-terms will be located at origin while the cross-terms will be located away from the center. For two Gaussian components at time points t_1 and t_2 with frequencies f_1 and f_2, the cross-terms will be located at $\nu_1 = f_2 - f_1$,$\nu_2 = f_1 - f_2$, $\tau_1 = t_2 - t_1$ and $\tau_2 = t_1 - t_2$.

In figure 6, we can see the ambiguity function for a signal with components at time points $t_1 = -100$ and $t_2 = 200$ with frequencies $f_1 = 0.3$ and $f_2 = 0.2$. We get a large component in the middle and two smaller away from the middle. These are corresponding to the cross-terms. These cross-terms will as expected be located at $\nu = f_1 - f_2$ and $\nu = f_2 - f_1$ and at $\tau = t_1 - t_2$ and $\tau = t_2 - t_1$.

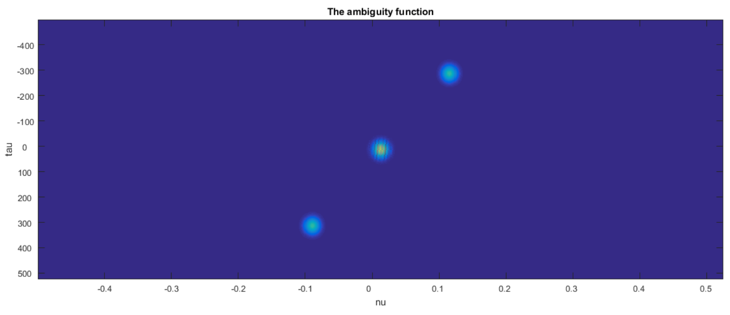


Figure 6: The ambiguity function of a signal with pulses at frequencies ,0.3 and 0.2 and at time points -100 and 200.

3.5 The impact on noise in the Wigner distribution and the ambiguity domain

This section will study how the Wigner distribution and the ambiguity domain will behave with noise. The theory of this subsection could be found in Stankovic (2002).

In order to see how noise impacts the different domains, we model a signal and noise as

$$y(t) = x(t) + e(t) \tag{14}$$

where x is the original signal, e is the noise, which in this case is considered white, and y is the measured signal. In (Stancovic 2002) they discuss this more in detail by first calculating the IAF function

$$\begin{aligned} r_y(t,\tau) &= y(t+\tfrac{\tau}{2})y^*(t-\tfrac{\tau}{2}) = [x(t+\tfrac{\tau}{2}) + e(t+\tfrac{\tau}{2})][x^*(t-\tfrac{\tau}{2}) + e^*(t-\tfrac{\tau}{2})] \\ &= r_x(t,\tau) + r_e(t,\tau) + x(t+\tfrac{\tau}{2})e^*(t-\tfrac{\tau}{2}) + x^*(t-\tfrac{\tau}{2})e(t+\tfrac{\tau}{2}) \end{aligned} \tag{15}$$

Here we see that the IAF is a sum of the IAF of x, e and the cross-term between them. Since x is a deterministic signal, the variance of $r_y(t,\tau)$ will consist of two parts, i.e.,

$$V[r_y(t,\tau)] = \sigma_{ee}^2 + \sigma_{xe}^2 \tag{16}$$

where σ_{ee}^2 is the variance caused by the noise and σ_{xe}^2 is the variance caused by the interaction between the signal and the noise. This can be interpreted as that the variance increases where the IAF of the original signal is large.

Since the noise and the signal are uncorrelated, the expected value will be

$$E[r_{\hat{X}}(t,\tau)] = r_X(t,\tau) + r_e(t,\tau) \tag{17}$$

For white noise $r_e(t,\tau)$ becomes $\sigma_e^2\delta(\tau)$. This means that the expected value of the IAF of the total signal will be as the IAF for the original signal except for $\tau = 0$.

If we now look at the Wigner distribution, see equation (4), the expected value is

$$
\begin{aligned}
E[\int_{-\infty}^{\infty} y(t+\tfrac{\tau}{2})y^*(t-\tfrac{\tau}{2})e^{-2i\pi f\tau}d\tau] &= \int_{-\infty}^{\infty} E[y(t+\tfrac{\tau}{2})y^*(t-\tfrac{\tau}{2})]e^{-2i\pi f\tau}d\tau \\
&= \int_{-\infty}^{\infty} r_x(t,\tau)e^{-2i\pi f\tau} + \sigma_e^2\delta(\tau)d\tau = W_x(t,f) + \sigma_e^2
\end{aligned}
\tag{18}
$$

Here, we see that the noise will be spread out equally everywhere in the domain. In practice it is not possible to store signals of infinite length which will make our calculations hard at the edges. To come around this, we zeropad, extending the signal with zeros, causing the IAF to exhibit a decreasing behaviour at the edges.

Since the variance of the IAF will be larger where the original signal has its components, the same behaviour will occur in the Wigner distribution.

We now look at the ambiguity domain. The expected value of A_y will be

$$
\begin{aligned}
E[\int_{-\infty}^{\infty} y(t+\tfrac{\tau}{2})y^*(t-\tfrac{\tau}{2})e^{-2i\pi\nu t}dt] &= \int_{-\infty}^{\infty} E[y(t+\tfrac{\tau}{2})y^*(t-\tfrac{\tau}{2})]e^{-2i\pi\nu t}dt \\
&= \int_{-\infty}^{\infty}(r_x(t,\tau) + \sigma_e^2\delta(\tau))e^{-2i\pi\nu t}dt = A_x(\nu,\tau) + \sigma_e^2\delta(\nu)\delta(\tau)
\end{aligned}
\tag{19}
$$

Here, we see that the main part of the noise will be located at the origin in the ambiguity domain.

In figure 7, we can compare the ambiguity domain and the Wigner distribution of a signal with and without noise. As expected there is a large peak added in the ambiguity domain at origin due to noise. We can also see some disturbances in the rest of the domain but not as large as at the origin. Notice also that although we have a noise with a quite high variance it is still quite clear where the peaks from the original signal are. When we do the same comparison with the Wigner distribution we see that the noise is located everywhere and that we still can see the components of the signal quite well.

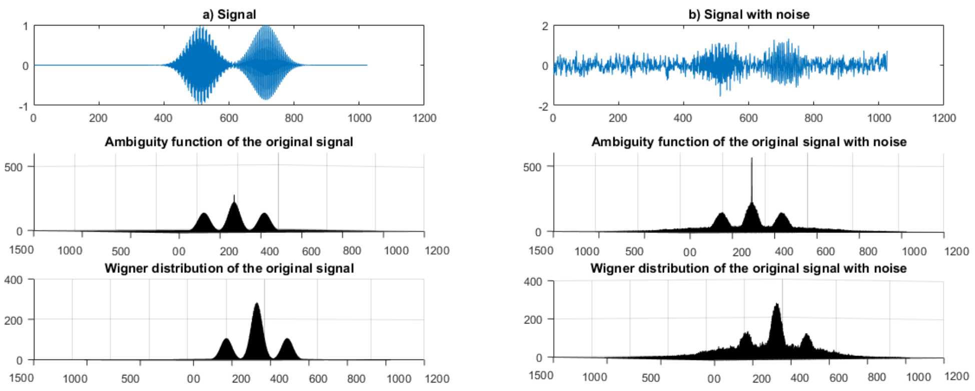


Figure 7: An illustration of how the noise in a signal will be located in the Wigner distribution and in the ambiguity domain. This is calculated on a signal where $t_1 = 512$, $t_2 = 712$, $f_1 = 0.3$, $f_2 = 0.2$ and $\alpha_1 = \alpha_2 = 40$. In the figures to the right we see the same signal but with noise added.

3.6 The filtered ambiguity function and Wigner distribution

We have seen that the cross-terms will be allocated away from the auto-terms in the ambiguity domain and we know that the auto-terms will be located at the origin. This makes us believe that we could also construct some sort of filter to separate out the different parts of the ambiguity function and hopefully get rid of noise.

The filtered ambiguity function (FAMF) is defined as

$$A_x^Q(\nu,\tau) = Q(\nu,\tau)A_x(\nu,\tau) \tag{20}$$

where Q is a kernel function of ν and τ that suppresses selected parts of the ambiguity function. The most common approach is to construct a kernel that suppresses the cross-terms and leaves the auto-terms untouched. This could also be done directly in the Wigner distribution. By calculating the Wigner distribution from the filtered ambiguity function we could see how this is done. The filtered Wigner distribution (FWD) will be

$$W_x^Q(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} A_x(\nu,\tau)Q(\nu,\tau)e^{2i\pi\nu t-2i\pi\tau f}d\nu d\tau = W_x(t,f) **q(t,f) \tag{21}$$

where

$$q(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} Q(\nu,\tau)e^{2i\pi\nu t-2i\pi\tau f}d\nu d\tau \tag{22}$$

and $**$ is a double convolution in the two variables. This mean that one can always translate the filter between the ambiguity, Wigner distribution and autocorrelation domains without losing any information.

3.7 Connection between the filtered Wigner distribution, the spectrogram and the multitaper method

In the definition of the filtered Wigner distribution Eq (21) we can see a clear quadratic behavior. This suggest the possibility of a connection between the Wigner distribution and the spectrogram and that this connection will tell us something about how to determine the windows in the spectrogram. The filtered Wigner distribution can be expressed as

$$W_x^Q(t,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(u+\frac{\tau}{2})x^*(u-\frac{\tau}{2})\rho(t-u,\tau)e^{-2i\pi f\tau}dud\tau \tag{23}$$

where ρ is the filter in the instantaneous auto-correlation domain. By using the variable change $u = \frac{t_1+t_2}{2}$ and $\tau = t_1 - t_2$, one obtains

$$\begin{gathered} W_x^Q(t_1+t_2,f) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1)x^*(t_2)\rho(t-\tfrac{t_1+t_2}{2},t_1-t_2)e^{-2i\pi f(t_1-t_2)}dt_1dt_2 = \\ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1)x^*(t_2)\rho^{rot}(t-t_1,t_2)e^{-2i\pi f(t_1-t_2)}dt_1dt_2 \end{gathered} \tag{24}$$

.

In the last step, we have introduced a new kernel

$$\rho^{rot}(t_1,t_2) = \rho(\frac{t_1+t_2}{2},t_1-t_2) \tag{25}$$

.

This corresponds to a scaling and a rotation of our original kernel ρ. If we now compare this to the spectrogram

$$S_x(t,f) = |\int_{-\infty}^{\infty} x(t_1)h^*(t_1-t)e^{-i2\pi f t_1}dt_1|^2 =$$
$$(\int_{-\infty}^{\infty} x(t_1)h^*(t_1-t)e^{-i2\pi f t_1}dt_1)(\int_{-\infty}^{\infty} x(t_2)h^*(t_2-t)e^{-i2\pi f t_2}dt_2)^* = \tag{26}$$
$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1)x^*(t_2)h^*(t_1-t)h(t_2-t)e^{i2\pi f t_2}e^{-i2\pi f t_1}dt_1dt_2$$

we can see that the expression for the spectrogram and the filtered Wigner distribution is similar and will become identical if we find a window that fulfills

$$h^*(t_1)h(t_2) = \rho^{rot}(t_1,t_2) \tag{27}$$

For a kernel ρ^{rot} that's symmetric, i.e., $\rho^{rot}(t_1,t_2) = \rho^{rot}(t_2,t_1)^*$, the window h can be calculated by solving the eigenvalue problem $\int_{-\infty}^{\infty} \rho^{rot}(t_1,t_2)u(t_1)dt_1 = \lambda u(t_2)$. Since ρ^{rot} is assumed to be symmetric, the vector set u_k will fulfill

$$\sum_{k=1}^{\infty} \lambda_k u_k^*(t_1)u_k(t_2) = \rho^{rot}(t_1,t_2) \tag{28}$$

where λ_k is the eigenvalues of ρ^{rot} and u_k the eigenfunctions. We can then rewrite the Wigner distribution as

$$W_x^Q(t,f) = \sum_{k=1}^{\infty} \lambda_K \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x(t_1)x^*(t_2)u_k^*(t_1-t)u_k(t_2-t)e^{2i\pi f t_2}e^{-2i\pi f t_1}dt_1dt_2$$
$$= \sum_{k=1}^{\infty} \lambda_k |\int_{-\infty}^{\infty} x(t_1)u_k(t_1-t)e^{-2i\pi f t_1}dt_1|^2$$

(29)

This means that the filtered Wigner distribution can be calculated by a summation over the spectrograms with different windows and weights. This method is often called the multitaper method where the windows u_k is called the multitapers (MT). In theory, this means that to achieve the correct Wigner kernel we need an infinite number of windows in our spectrograms which in practise is not possible. Instead we use a finite number of windows and select a number that will correspond to a good approximation of our desired kernel. The number of windows will depend on the shape of the kernel and the length of the windows.

In comparison to calculating the ambiguity domain and apply a desired kernel, the multitaper method will not get the same resolution and effect. Anyhow the multitaper method has another big advantage and that is speed. In order to calculate the Wigner distribution we first need to calculate the IAF which could be quite expensive and then use the FFT in order to get the Wigner distribution. The multitaper method is often very fast since its based on the spectrogram which is very fast. Later on we will study how the number of windows and the length of them will affect our final kernel.

4 The classification problem

In order to classify the syllables we calculate features for each syllable and use these features to compare and compute some sort of distance between each syllable. In this section, we will go through calculation of features, distance measures between features, classification and determining how many classes we could find.

4.1 Motivation and calculation of features

When comparing two different syllables to each other its important that the algorithm is robust to different aspects. The first one is that even if we have some small shifts of the signal in the time and frequency direction we still want to classify the syllable as the same. Since the algorithm described in chapter 2.1 sometimes will detect a syllable with some time shifts, especially in a noisy environment, its important that the methods do not depend on these irregularities. The same holds for shifts in their frequencies. Different birds might not be able to have the same range in their tone register and might sing the same thing but in a different key. In order to detect this it's important that our features do not depend on these shifts.

As we have seen before the amplitude of the ambiguity domain is invariant to time and frequency shifts. This domain is also quite noise robust since the largest part is the noise will be concentrated at the origin. If we compare this to the spectrogram or the Wigner distribution they do not fulfill either of these conditions. This makes us believe that the ambiguity domain is a good representation of our syllables.

In order to be able to compare different signals with as little impact from noise as possible we use the singular value decomposition (SVD) to calculate our features. From the singular value decomposition of a matrix X we get the matrices U, S and V so $X = USV^T$ where U and V is unitary matrices and S is a diagonal matrix with positive elements sorted in decreasing order. The U matrix in this decomposition will form an unitary basis spanning the columns of X sorted after impact on X and the V matrix will form a unitary basis spanning the rows of X. Since these vectors is sorted by impact on X the vectors with lots of information will be placed in the first columns of U and V. The noise which is assumed to be white does not have any structure and will be spread out on all vectors. By only using the first n vectors from U and V as features we will hopefully get features that is quite noise robust and invariant to time and frequency shifts. We can also conclude that by not including the S matrix in our features our method will be amplitude invariant.

In figure 8, we can see the ambiguity domain of a signal with three components at frequencies 0.2, 0.2 and 0.1 at time points 300, 0 and -100 and the first three vectors from the U and V matrices. Notice that when varying τ the ambiguity spectrum has three positions where there are peaks corresponding to the three peaks represented in the U vectors. When varying ν, we instead have five positions where we find peaks. This we see in the vectors from the V matrix. We can also see that the peaks will not be represented in one vector but spread out to many different ones. This will lead to a consideration of how many vectors to use as features since we want a good representation over the ambiguity domain without including too much noise.

When studying very noisy signals the order of the vectors in the U and V matrix can sometime vary if the singular values are close to each other. This could have a large impact on our methods since we do not include all vectors and need to find the "same" vectors in all signals in order to compare them. In our application, these shifts tend not to happen for the first vector but is more common for the other ones, giving us no reason to worry if we only use one vector as a feature.

4.2 Comparing two sets of features

When comparing the features from two different syllables we need a measure of the distance between the features, a so called metric. In Grosse Ruse et al. (2016), the authors use only one feature vector from the U and V matrix and compare two different sets of features using the metric

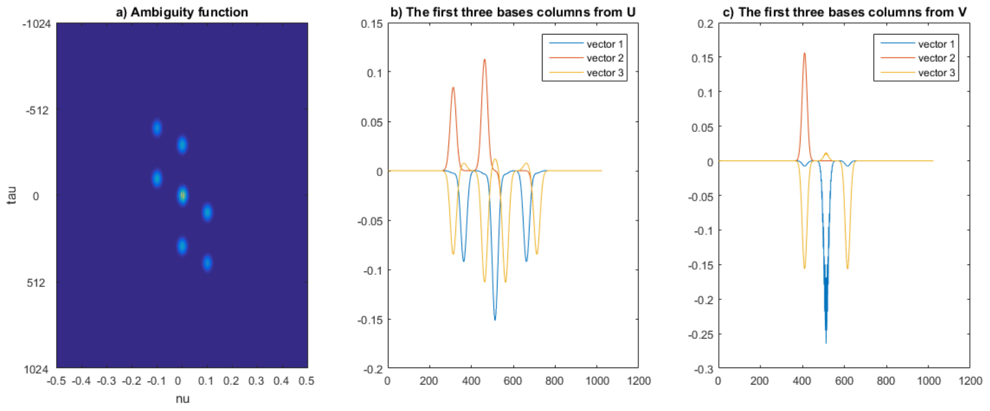


Figure 8: a) The ambiguity domain of a signal with three components at frequencies $0,2$, $0,2$ and $0,1$ at time points 300, 0 and -100. (b) The first three columns from the U matrix. (c) The first three columns from the V matrix.

$$d(u_1, u_2, v_1, v_2) = 1 - min(|u_1|^T|u_2|, |v_1|^T|v_2|) \tag{30}$$

where $|\cdot|$ is the elementwise absolute value, u_1 and v_1 is the first vectors from the $U_1 = U$ and $V_1 = V$ matrices for syllable one and u_2 and v_2 is the first vectors from the $U_2 = U$ and $V_2 = V$ matrices for syllable two. We call this metric the cosine metric. Since U and V from the SVD are unitary matrices, $d(u_1, u_2, v_1, v_2)$ will always be between zero and one. Since the u vector only tell about relationships in the τ direction in the ambiguity domain, this will not change if something will change in the ν direction. That's why its important to include both u and v. To use more than one feature vector from U and V, we propose a similar way to measure the distance

$$d(U_{1n}, U_{2n}, V_{1n}, V_{2n}) = 1 - min\left(\frac{2||U_{1n}^T U_{2n}||}{||U_{1n}||^2 + ||U_{2n}||^2}, \frac{2||V_{1n}^T V_{2n}||}{||V_{1n}||^2 + ||V_{2n}||^2}\right) \tag{31}$$

where $||\cdot||$ is the spectral matrix norm, U_{1n}, U_{2n}, V_{1n} and V_{2n} are matrices with the first n columns from the original matrices U_1, U_2, V_1 and V_2. Since we work with unitary matrices, meant that $||U_{1n}|| = 1$ and $||V_{1n}|| = 1$, the metric can be rewritten as

$$d(U_{1n}, U_{2n}, V_{1n}, V_{2n}) = 1 - min(||U_{1n}^T U_{2n}||, ||V_{1n}^T V_{2n}||) \tag{32}$$

4.3 The classification

The following subsection can be read about in Everitt B. et al. (2011). In order to classify our different signals into clusters we make use of the linkage method included in the statistics and machine learning toolbox in Matlab which uses average linked clustering. This is a recursive method who starts with each element as its own cluster and then recursively merges the two clusters with shortest distance together into one. The distance between two clusters is measured as the average distance between all combinations of elements in the two sets. In figure 9, we can see a dendrogram with over 300 elements. The dendrogram shows us how each element has been merged together into different classes. At the bottom of the tree, we see each element as its own cluster and the further up we go in the tree the elements get merged together constructing a cluster. The number of clusters produced from our method will depend on where we "cut" the tree. In this case, cutting the tree at 0.025 will cut through four branches giving us four clusters. The same reasoning will give us two clusters if we cut the tree at 0.04.

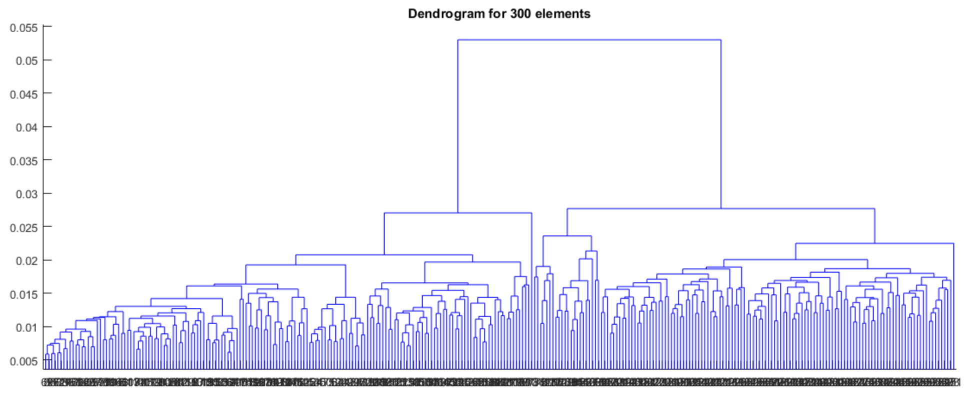


Figure 9: Dendrogram over a set of 300 elements. The dendrogram shows us how the elements has been merged together to construct the different clusters. In order to get a classification we cut the dendrogram at a given height. The clusters is then given by the subtrees created by the cut.

The compactness of a tree, meaning how separated each merge point is between each other, tells us a lot of how well our method is working and how hard it is to classify into different classes. In the case of figure 9 we can see that to construct two clusters we can cut the tree within a large interval and still get the same result. These two clusters is then quite well separated. If we instead construct four clusters the cut interval is significantly smaller meaning that the clusters are not as well separated. This will have a large impact on determine the number of clusters in our dataset.

4.4 Silhouette values and determining the number of clusters

In order to determining the number of clusters present in a dataset we make use of something called the silhouette value. The silhouette value is calculated for each element i and is defined as

$$S(i) = \frac{\min_k(d_{other\ cluster}(i,k)) - d_{own\ cluster}(i)}{\max(\min_k[d_{other\ cluster}(i,k)]\ ,\ d_{own\ cluster})} \tag{33}$$

where $d_{other\ cluster}(i,k)$ is the average distance between element i and element k in another cluster and $d_{own\ cluster}(i)$ is the average distance between element i and the other elements in the same cluster. The silhouette value is always between -1 and 1 and is a measure of how well the element fits in its own cluster. A silhouette value close to one will mean than the element is correctly classified and a value close to minus one means that the element is classified into the wrong class.

We can now get a measure of how the classification performs by calculating the average of each elements silhouette value. If a classification problem should be treated as a successful classification, each element should have a high silhouette value leading to a high average. By varying the number of clusters, meaning cutting the dendrogram tree at different levels, we could see how the score for each classification behaves. The number of clusters in the dataset is set to the number giving the best score.

Note that this method will encounter some problems when trying to check the silhouette score for a large number of clusters. When we use the same number of clusters as elements the distance between an element and its own class will be set to zero giving its silhouette value to one. This means that the total score for that classification also will be set to one. In order to get around this problem we make the assumption that the number of clusters is significantly smaller than the number of syllables.

5 A study of some special kernels and six different methods for feature calculation

In this section, we will go through some interesting kernels and their behaviour. Inspired from this study, six different methods for feature extraction and noise reduction are designed. The first five methods are all built on calculating the filtered ambiguity function using multitapers with different windows and then extracting features using the SVD.

5.1 Some special and interesting kernels and their corresponding windows

One common approach when analyzing a signal is to try get rid of the cross-terms and only look at the auto-terms. This could be done with a filter of the form

$$\Theta(\nu, \tau) = e^{-\frac{(\nu^2+\tau^2)}{\alpha}} \tag{34}$$

where α is a scaling parameter that determines how wide the kernel will be. In the top image in figure 10, we can see the Wigner distribution of a signal with frequencies 0.1 , 0.2 and 0.4 at time points 321, 821 and 521. Notice the cross-terms located between every combination of the auto-terms. In the lower image we have calculated the corresponding windows of length 256 and weights to the kernel above and used the first eight windows to calculate the Wigner distribution with the multitapers method. As expected, the cross-terms reduced and only the auto-terms remain. However, the auto-terms are smoothed. This can be explained by the numbers of windows. Since we use a window length of 256 samples we need the same amount of windows in order to get a correct representation of our kernel. Instead we only used eight windows which will correspond to an approximation of our desired kernel.

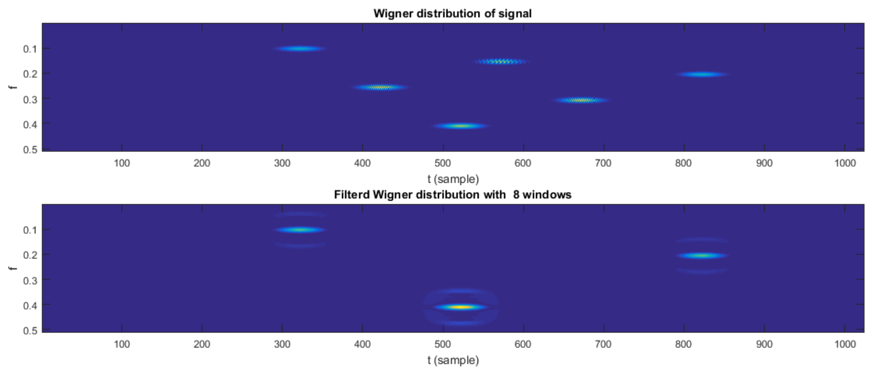
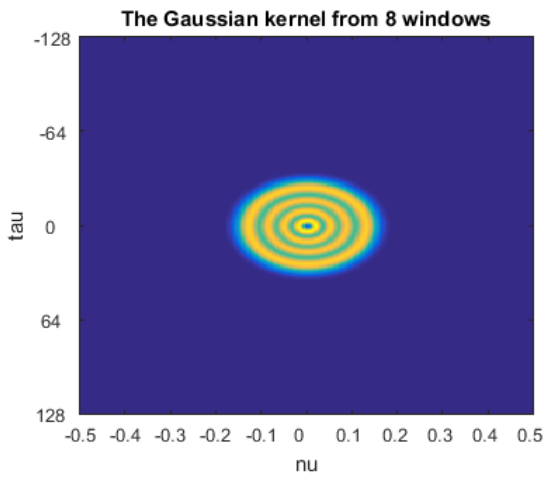
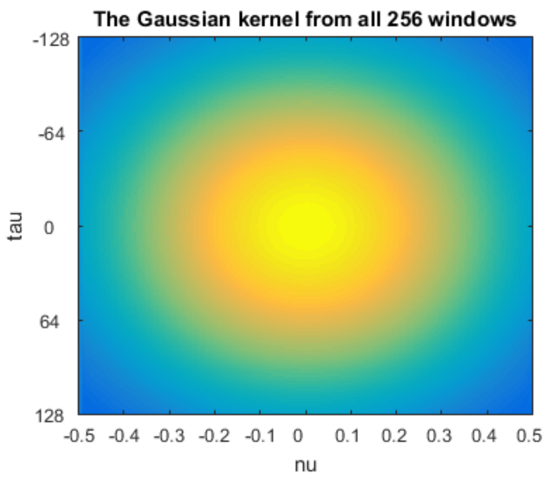


Figure 10: In the upper image we see the Wigner distribution of a simulated signal with three Gaussian components with frequencies 0.1, 0.2 and 0.4 at time points 321, 821 and 521, respectively. In the lower image, we see the same signal but filtered with 8 windows of length 256 samples corresponding to a Gaussian kernel in the ambiguity domain.

In figure 11, we can see the corresponding kernel for eight windows and for all 256. As expected, the kernels are very different from each other. By using only eight windows, the kernel will get smaller and we can also see some oscillation in the kernel. This gives us the smoothing we saw earlier. However, the kernel will still remove cross-terms since the kernel is only located at the origin.



(a)



(b)

Figure 11: a) The Gaussian kernel with $\alpha = 0.16$ when represented with the first eight windows from the multitaper method. b) The original Gaussian kernel with $\alpha = 0.16$.

In figure 12, we see the first four windows from the Gaussian kernel. They seem to be close to sinusoidal functions of different order.

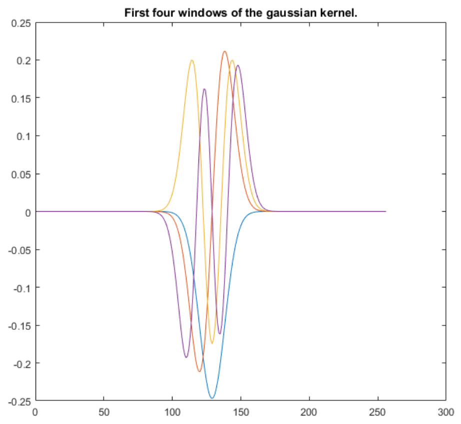
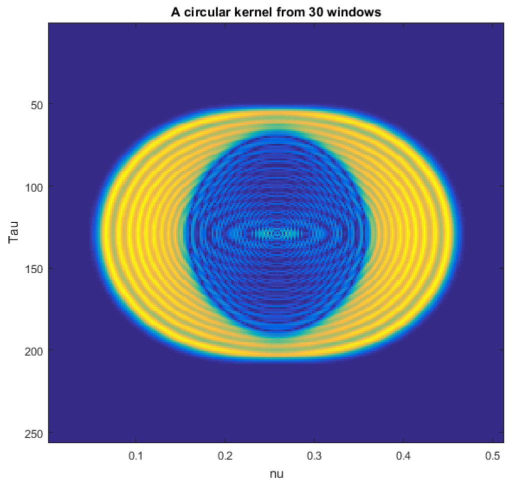


Figure 12: The first four windows from the Gaussian kernel.

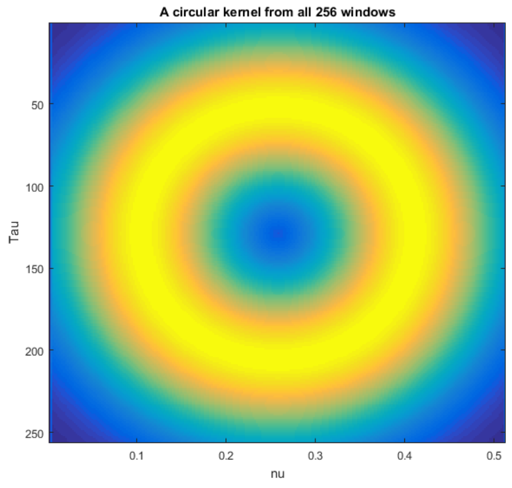
If we now instead look at a kernel that suppresses the auto-terms and retain the cross-terms it could look like

$$\Theta(\nu, \tau) = e^{-\frac{(\sqrt{\nu^2+\tau^2}-R)}{\alpha}} \tag{35}$$

which will reach its maximum at a distance R away from the origin and is approximately zero at the origin. This kernel will suppress the auto-terms and keep the cross-terms at the distance R. In figure 13, we can see how this kernel performs when we represent it with 30 or all 256 windows. Here we see that when scaling down the kernel to 30 windows we will get a quite different result. We find large oscillations in the "hole" in the middle which will interfere with our auto-terms and not suppress them as we expect. We can also see that the width of the circle changes, giving a better resolution for the ν variable. This means that in order to get a good approximation of our kernel and get rid of the auto-terms, we need a lot of windows which will make it a consideration if multitapers is worth the effort.



(a)



(b)

Figure 13: a) The representation of the circular kernel when using the first 30 multitaper windows when $\alpha = 0.16$ and $R = 0.6$. b) The representation of the circular kernel when using all 256 multitaper windows when $\alpha = 0.16$ and $R = 0.6$.

In figure 14, we can see the first four windows of the circular kernel. Unlike the windows for the Gaussian windows, these are not centered around the middle but consists of two sinusoidal functions. However, this is since the windows want to measure the interplay between signal components. This way when the windows is centred between two auto-terms the sinusoids will align with those terms and detect the crossterm. Windows of this kernel tend to come in pairs. In figure 14, the red and the blue windows are the same on the left part and shifted to the right. The same behaviour is for the yellow and the purple windows.

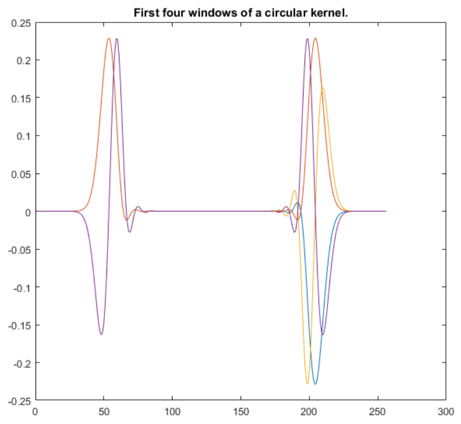


Figure 14: The first four windows of a circular kernel.

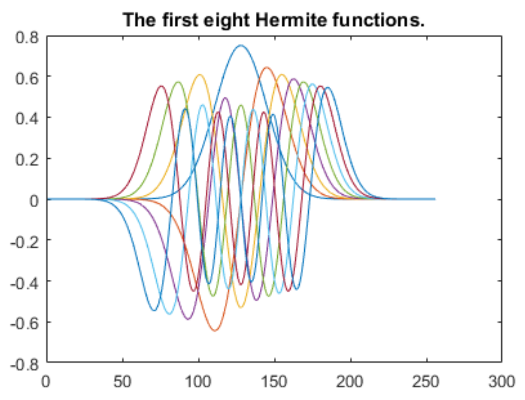
Notice that when using a circular kernel its important to have a suitable radius. Since the length of the sinusoids in the windows is determined by the radius and that the windows only detect cross-terms between auto-terms with a corresponding distance between each other.

5.2 The Hermite kernel method

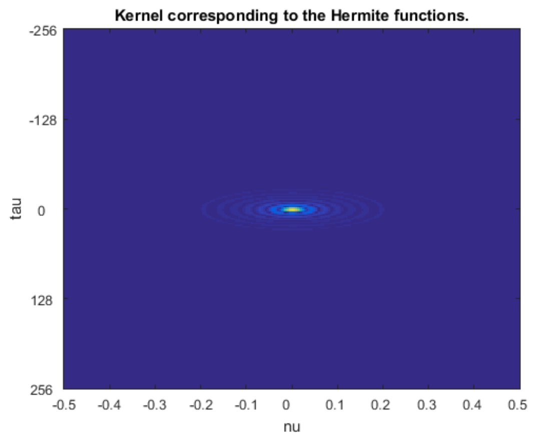
The first method we call the Hermite kernel method (HKM) and is the one described and used in Grosse Ruse et al. (2016). It makes use of the Hermite functions defined as

$$h_n(x) = \frac{(-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n}(e^{-x^2})}{2^n n! \sqrt{\pi}} \tag{36}$$

and uses them as windows in the MT method with weights one. In figure 15, we can see the first eight Hermite functions and the corresponding kernel these windows produce. As seen this kernel is centered at the origin meaning that it focus on the auto-terms. Earlier we saw that white noise in a signal will be spread out over the AMF and give us a large peak at the origin. The goal with this kernel is to remove the noise away from the origin and keep as much from the auto-terms as possible. The size of the kernel is determined by the number of windows used. By increasing the number of windows, the resulting kernel gets more narrow and smaller. In this thesis, the number of windows for the HKM is set to eight.



(a)



(b)

Figure 15: (a) The first eight Hermite functions. (b) The corresponding kernel for the first eight Hermite functions with weights one.

After calculated the FAMF we use the SVD of the absolute value in order to get the U and V matrices. The first column of these matrices are used as features.

5.3 The truncated Wigner method

The idea behind the truncated Wigner method (TWM) is as for the HKM to use the Hermite functions to filter out noise from the signal and get a FAMF to extract features. The difference is an extra step in the process that hopefully removes more noise. Similar to the HKM, we use the first eight Hermite functions as windows for the multitaper method and calculate the WD. We calculate the truncated Wigner distribution and uses it to calculate the FAMF. The truncated Wigner distribution W_T is defined as

$$W_T(t,f) = \begin{cases} W(t,f) & if \quad W(t,f) \geq \alpha \\ 0 & if \quad W(t,f) < \alpha \end{cases} \quad (37)$$

where W is the WD, and α is a precalculated constant. The parameter α is set to be so that approximately 95 percent of WD:s amplitude is below α. By removing noise in the Wigner domain its easier to remove noise that otherwise will be located near the origin in the ambiguity domain.

When calculating the features we use the double Fourier transform of W_T to calculate the AMF. We calculate the SVD and use the first vectors from the U and V matrices as features for the signal.

5.4 The principal component method

The principal component method (PCM) uses the same idea as the TWM and calculates the WD using the MT method with the Hermite windows and try to reduce noise in the Wigner domain. This is done with the SVD. Since the U matrix from the SVD spans the columns in WD we can reduce noise by calculating the U matrix and then project our WD on to the first d columns in U. Since the noise will typically be spread out on all columns of U the projection on a subset of U will hopefully contain less noise. The projected Wigner distribution is calculated as

$$W_p = U_d(U_d^T U_d)^{-1} U_d^T W = U_d U_d^T W \tag{38}$$

where U_d is a matrix with the first d columns from the U matrix. In this thesis $d = 5$. In figure 16 a) we can see the FWD of a signal at frequencies 0.3 and 0.4 at time points 0.3 and 0.7 when using the first eight Hermite functions as windows. In figure 16 b) we see the same FWD after we have used the noise reduction method. We see here that the FWD becomes more smooth and that the signal components are more square than the original one. Since we have reduced the rank of our domain to five that is a quite reasonable result.

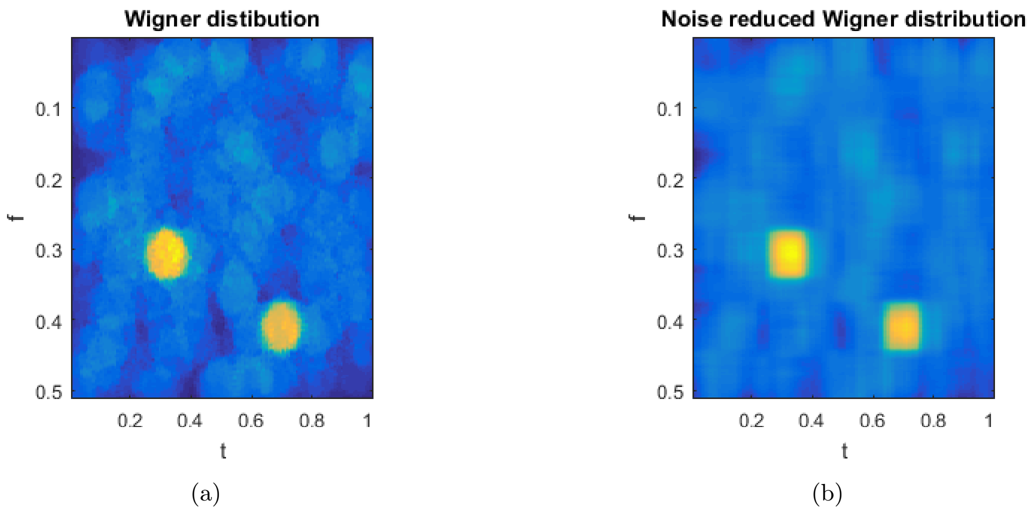


Figure 16: (a) The FWD of a signal with two simulated components at frequencies 0.3 and 0.4 at time points 0.3 and 0.7. b) The result after using the noise reduction in PCM using the first 5 columns in the U matrix.

As in the TWM we use the noise reduced Wigner distribution to calculate the FAMF and then the first vectors from the U and V matrices of the SVD as features.

5.5 The circular kernel method

The idea behind the circular kernel method (CKM) is inspired from Sandsten (2017) and is not to study the auto-terms of the signal but instead calculate features based on the cross-terms. In order to do so we need a kernel that removes the auto-terms and keep the cross-terms in the AMD. This is done with a kernel defined as

$$\Theta(\nu, \theta) = e^{-\frac{(\sqrt{\nu^2+\theta^2}-R)}{\alpha}} \tag{39}$$

where R is a constant defining the radius of our kernel and α the width. In this thesis, R is set to 0.45 and α to 1. As discussed before the multitaper method may have some problems with representing a circular kernel with a reasonable number of windows without including the auto-terms located at the origin. This problem can be solve in two ways. We either include enough windows so that the kernel is represented in such a good way that the corresponding kernel will be zero at the origin. We can also include only a few windows giving us less disturbances at the origin but also a worse representation of the kernel. Since we do not want a method that is too slow it could be convenient to use as few windows as possible. In figure 17, we can see how the kernel will be represented with the first 20 windows. This representation is not near to represent a circular kernel but since we have assumed that our signals do not consist of two frequencies at the same time point, we can also assume that the cross-terms will not be located at the line $\nu = 0$. There is there for no need to represent a kernel covering that area.

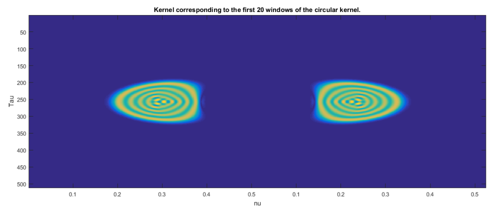
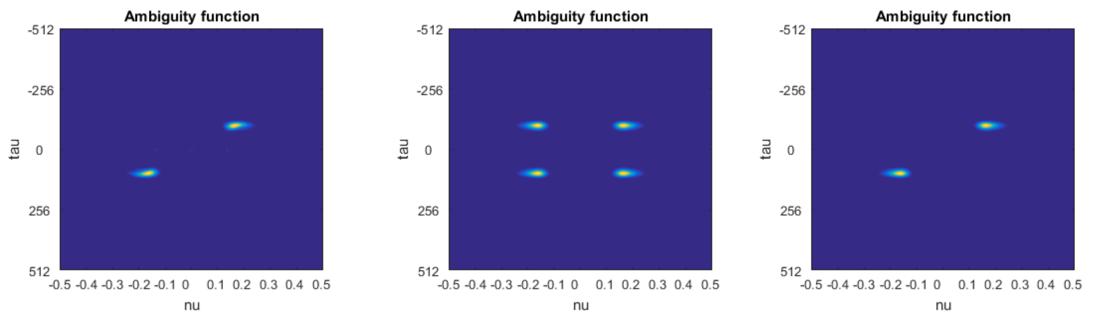


Figure 17: The circular kernel with $R = 0.45$, $\alpha = 1$ represented with the first 20 multitapers.

When calculating the feature vectors from the FAMF it is important to be careful. Since our FAMF consist of at least two separated components it will be hard to represent this with only one vector from the U and V matrices. In figure 18 we can see the FAMF of a simulated signal with components at time points 0 and 150 with frequencies 0.1 and 0.2 and how the representation of this FAMF will look for different number of feature vectors. When using only one feature vector it is not clear due to symmetry where the components is located. But by including more vectors there is enough information in order to separate the two cases.



(a) (b) (c)

Figure 18: a) The filtered ambiguity function of a simulated signal at sample components 0 and 150 with frequencies 0.1 and 0.2 using a circular kernel with 20 windows. b) The representation of the signal with the first vector from the U and the V matrix from the SVD. c) The representation of the signal with the first two vector from the U and the V matrix from the SVD.

When using the CKM it is important to have a few things in mind. Since the location of the cross-terms depend on the structure of the signal and can vary much it is hard to find a kernel that covers the location of all cross-terms. A syllable with three components will for example have six cross-terms which two corresponds to a difference between component one and two. Two cross-terms corresponding to component one and three and two cross-terms corresponding to component two and tree. This leads to a lot to take into account when finding the right parameters for the kernel. When dealing with a syllable where the components have a large time interval between each other the radius R of the kernel must be set accordingly in order to include the cross-terms matching that time difference. Since the spread of the cross-terms of a syllable can be quite large, it is desirable to make the kernels width larger with a higher α. A larger kernel will however require more multitapers in order to get a good representation of the kernel. Since more windows will make the method slower it is not practical to try to include all the cross-terms for the syllable. When R and α are set to $R = 0.45$ and $\alpha = 1$ the kernel will cover the cross-terms corresponding to a time difference of around 220 samples. This will hopefully give us the cross-terms corresponding to two syllable parts located next to each other. Cross-terms between other syllable parts are not included with this kernel.

5.5.1 Some special problems with the circular kernel method and how they are solved

As discussed before, it can occur some problems when including more than one feature vector from U and V matrices when studying noisy signals due to some shifts in the vector order. In order to solve this problem and be able to compare the correct vectors to each other, we need to find which vectors that match. When comparing two signals, we pick out the first N vectors from the first signal. Lets call these vectors u_{11},... , u_{1N}, v_{11},... ,v_{1N-1} and v_{1N}. We then find the matching vectors from signal two by solving

$$[k_1, ..., k_N] = \underset{k_i, k_i \neq k_j \forall i,j}{\operatorname{argmin}} [N - \sum_{i=1}^{N} \frac{|{u_1}_i^T u_{2k_i}| + |{v_1}_i^T v_{2k_i}|}{2}] \tag{40}$$

where u_{2k} and v_{2k} is the k:th feature vector from the U and V matrices from the second signal and k_n is the index of the vector from signal two that best correspond to the vector n from signal one. With this method, we will find the best match for our N first features from our first signal. N is often set to be 2 or 3.

Another problem occurs when calculating the U and V matrices with the SVD. Then the columns in the U and V matrices are only defined up to sign. This means that when comparing two similar signals their feature vectors could have changed sign. When this happens it occurs in both the U and the V matrices. As long as the change occurs in both the U and V they will still represent the same AMF. However, if a sign change only occurs in either U or V the features will no longer represent the same signal. This is something that the cosine metric can not separate since it do not care about sign changes. Instead, we use a metric defined as

$$\begin{split} d(U_{1n}, V_{1n}, U_{2n}, V_{2n}) = 1 - \min[\tfrac{\sum_{i=1}^{N} |{u_1}_i^T u_{2k_i}|}{N}, \tfrac{\sum_{i_1}^{N} |{v_1}_i^T v_{2k_i}|}{N}, \\ \tfrac{\sum_{i=1}^{N} |{u_1}_i^T u_{2k_i} {v_1}_i^T v_{2k_i}|}{N}] \end{split} \tag{41}$$

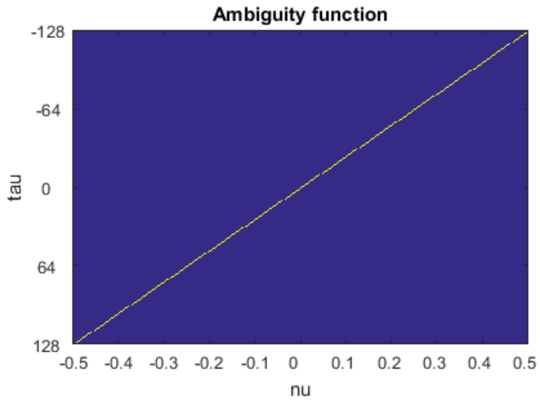
where k_i is the optimized index calculated to find the optimum pair. This function will solve the problem since a sign change in both u_{1i} and v_{1i} will not change the function value. However, if the sign change occurs in only u_{1i} or v_{1i} the two first components of the function will stay the same but the last component will change significantly.

5.6 The line kernel method

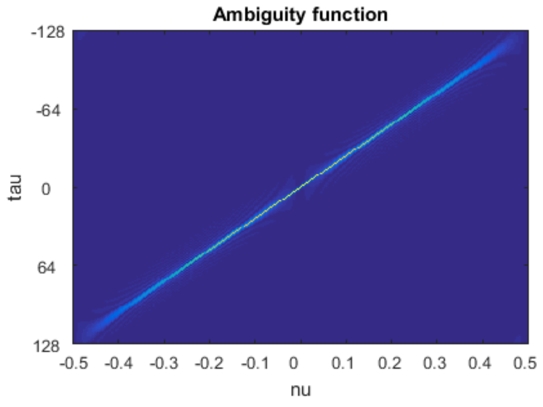
The idea behind the line kernel method (LKM) is to get a kernel that only includes a very small part of the auto-terms located at the origin. This is done with the kernel

$$\Theta(\nu, \theta) = \delta(\nu + \theta) \tag{42}$$

where δ is the dirac delta function. With this kernel we will include a small part of the auto-term since the kernel is one on the line $\nu + \theta = 0$ and zero otherwise. A big advantage of using a kernel forming a line this way is that it is easy to represent with few multitaper windows. In figure 19. we can see a comparison between the original kernel and the one represented with the first multitaper window. As we can see they are similar which indicates that we only need the first window in our method. Notice that since our kernel is not represented perfectly the line is wider, which will make it include more of the signal.



(a)



(b)

Figure 19: a) The line kernel. b) The line kernel represented with one multitaper window.

As in the HKM we calculate the filtered ambiguity function but now with our new window and then use the SVD to calculate our features for our signal. Notice that with this kernel it is possible that we will include cross-terms lying on the diagonal. But since the auto-term always is larger than the cross-terms the first vector from the U and V matrices will correspond to the auto-terms. So by using the first two vector as features the feature will not represent any cross-terms that happened to be included.

5.7 Mel-frequency cepstral coefficients method

The Mel-frequency cepstral coefficients method (MFCC), which is a method that often is used in speech recognition, is not based on the ambiguity domain and features from the SVD. MFCC is instead based on cepstral coefficients of a signal. The cepstral coefficients are briefly calculated the following way. First, the windowed spectrogram of the signal is calculated. The result is then passed to a mel-spaced frequenzy bank. The first half of the features are then calculated with the cosine transform. The second half is then the delta coefficients of the cosine transformed data (Jancovic P. 2011).

The implementation of this method is the same implementation as Grosse Ruse et al. (2016). This method is included in this project as a reference since it is very popular in similar applications.

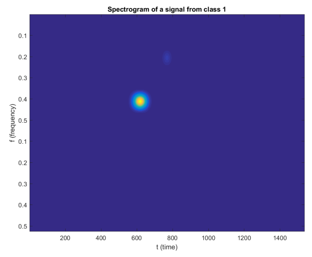
6 Methods for evaluation

In order to validate and see how well the methods work as classifiers and how well they take care of noise we will look at some different factors. This is done applying all methods for classification of a simulated dataset containing five different classes and with 60 signals in each class. In figure 20, we can see the spectrogram of a typical signal from each class in our simulated dataset. By including class one and two, we will simulate how well the methods are able to separate different phrasings. Class three is quite similar to class two but contains a few more small components. Class four and five are supposed to be quite different from the other classes but contain some similarities. We can for example find the same frequency leaps in class three and class five that's similar even if the signal is quite different in many other ways.

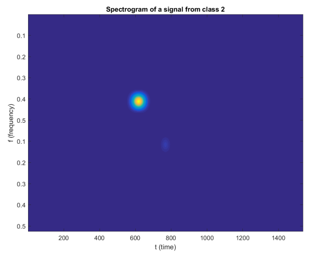
In order to get a more realistic representation of a real bird recording, we shift each signal randomly in time and frequency so that no signal is identical to another. The same is done with the time variable. Each signal is produced according to the form

$$x(t) = \sum_{k=1}^{N} A_k e^{-\frac{(t-(t_k-\Delta t_k))^2}{\alpha_k}} sin((2\pi f_k - \Delta f_k)t + \theta_k) \tag{43}$$

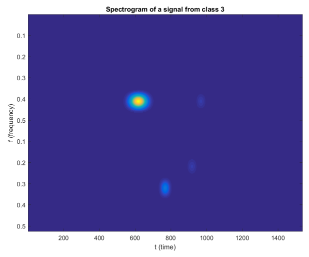
which is a extension of equation (2). The new variables Δt_k and Δf_k are uniformly distributed random variables that corresponds to the shifts.



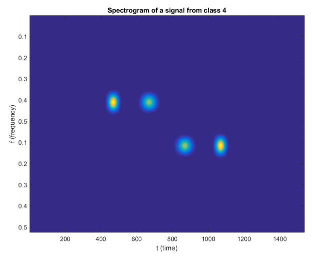
(a)



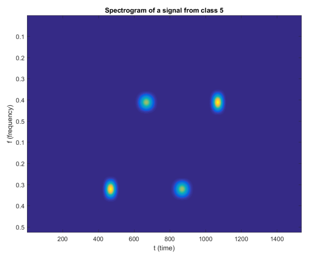
(b)



(c)



(d)



(e)

Figure 20: Typical spectrograms of the classes in the simulated data set. Each f_k, t_k, A_k and α_k can be seen in table 1. Class 1 can be seen in image a), class 2 in b), class 3 in c), class 4 in d) and class 5 in e).

$Class$	f_k	t_k	A_k	α_k
1	0.2, 0.1	618, 768	4, 1	30, 20
2	0.2, 0.4	618, 768	4, 1	30, 20
3	0.2, 0.4, 0.35, 0.2	618, 768, 918, 968	4, 2, 1, 1	40, 20, 20, 20
4	0.2, 0.2, 0.3, 0.3	468, 668, 868, 1068	4, 3, 3, 4	20, 30, 30, 20
5	0.4, 0.2, 0.4, 0.2	468, 668, 868, 1068	4, 3, 3, 4	20, 30, 30, 20

Table 1: Table over how each class in the simulated data set is constructed.

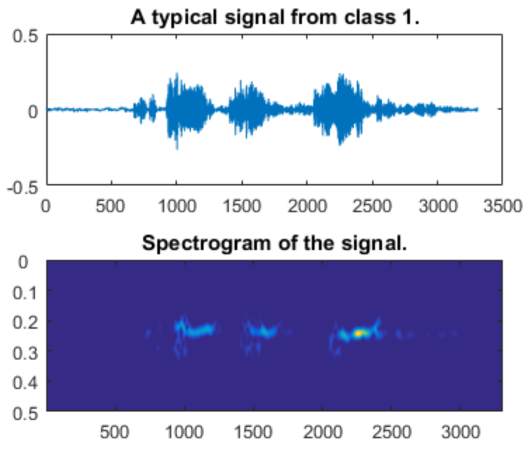
By varying the signal to noise ratio (SNR) on the dataset, we are able to see how well our methods handle noise without breaking apart and how well they can classify the signals into their corresponding class. The SNR is defined as

$$SNR_X = 10\log(\frac{\sum_{t=0}^{N-1} X(t)^2}{N\sigma_e^2}) \tag{44}$$

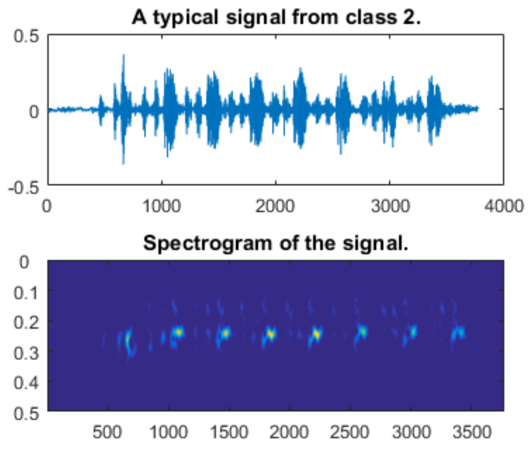
where N is the number of samples in the signal and σ_e^2 is the variance of the noise. The SNR is a measure of how much noise we can find in the signal. When studying real bird recordings the main noise source is often caused by wind which can be assumed to behave as white noise. Because of this we will limit our investigations to see how our methods work when handling white noise. The results is demonstrated in two ways. The first one is to study the dendrogram each method produces and see how they change with SNR. By study how the branches in the dendrogram tree changes with SNR we will get a feeling of how much the methods are affected by noise and how likely the method will classify into a fix number of classes.

Another way to illustrate our results for our methods is to let them classify our simulated data set and see which classes that are confused with each other for different SNRs. This is interesting in two cases. The first one is when we tell the methods the correct number of classes in the set. The second one is when the methods themselves determine the number of classes with help of the silhouette values.

In order to see how well the methods work on real recordings from the GRW, we validate them on two different datasets. In order to get a ground truth the signals are first classified by hand. The first dataset contains two classes that are very different from each other. In figure 21, we can see one signal from each class. The first class is a so-called whistle sound and contains only a few peaks. The second class is a so-called rattle sound and contains a lot more components than the first class does.



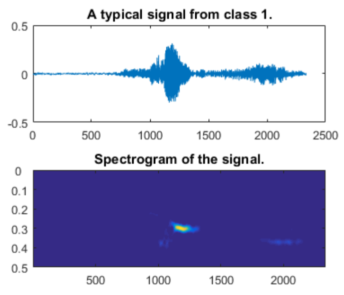
(a)



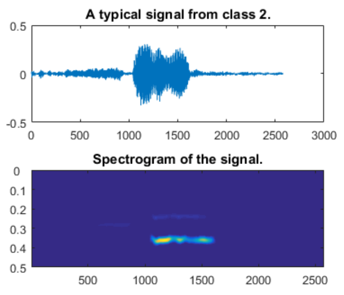
(b)

Figure 21: A typical syllable from each class in the first real dataset. In the top left image we see a plot of a typical signal from the first class and to the bottom right the see the corresponding spectrogram. The images to the right show the corresponding images for a typical signal from class two.

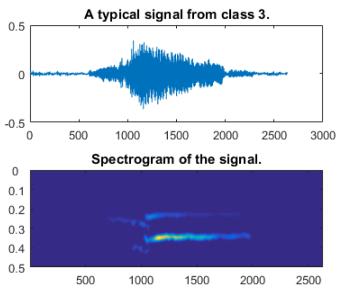
The second real dataset contains of three classes seen in figure 22. These are all rattle sounds and consist of quite a few components.



(a)



(b)



(c)

Figure 22: A typical syllable from each class in the second real dataset.

In both datasets, we will vary the SNR level between -10 and 10 and see how each method is able to classify these signals into their corresponding class. Since in practice we do not know how many classes there is in a recording we let the methods determine the number of classes themselves with the silhouette values. In this way, we get a feeling of how trustworthy the methods are.

When determine the SNR of a signal from real recordings, the original signal is treated as noise free when no extra noise is added. This is of course not a realistic assumption since there will always be noise in the recording from wind and other sources. These noise levels are however quite low and hard to remove so for simplicity we do not take them into account.

7 Results from simulated data

In figure 23, we see how the branches of the dendrograms changes with SNR. We have picked the branches that separates our dataset into 3, 4, 5 and 6 classes.

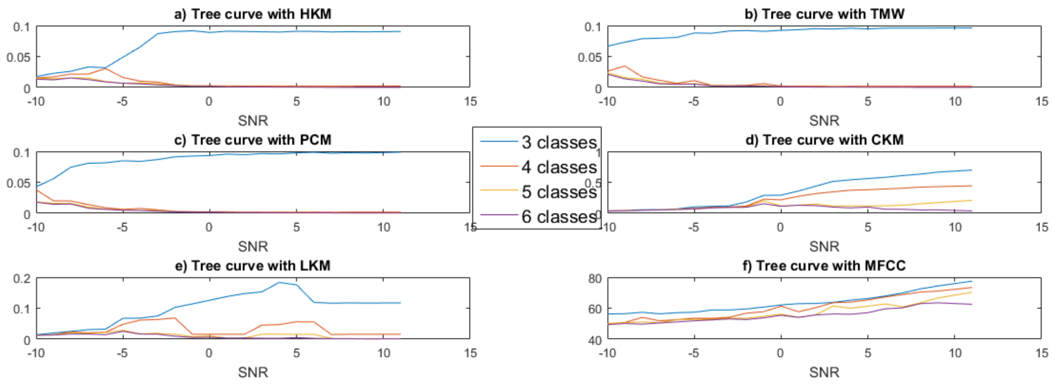


Figure 23: Plot of how the height of the branches in the dendrogram changes with SNR. Each curve shows where to cut the dendrogram in order to get a fix number of classes in different SNR. This in done with 3, 4, 5 and 6 classes. In a) we see the results for the HKM; b), TWM; c), PCM; d), CKM with two feature vectors from the U and V matrices; e), LKM; f), MFCC.

In figure 24, we can see how each signal in our simulated dataset has been classified in different SNR levels. Along the x axes, we have all our 300 signals and along the y axes we see the SNR level.

The colour of each pixel shows what class the corresponding signal has been classified into. In order to easier validate the methods the signals are sorted after the correct classification, meaning that all signals that should be classified as equal is located next to each other.

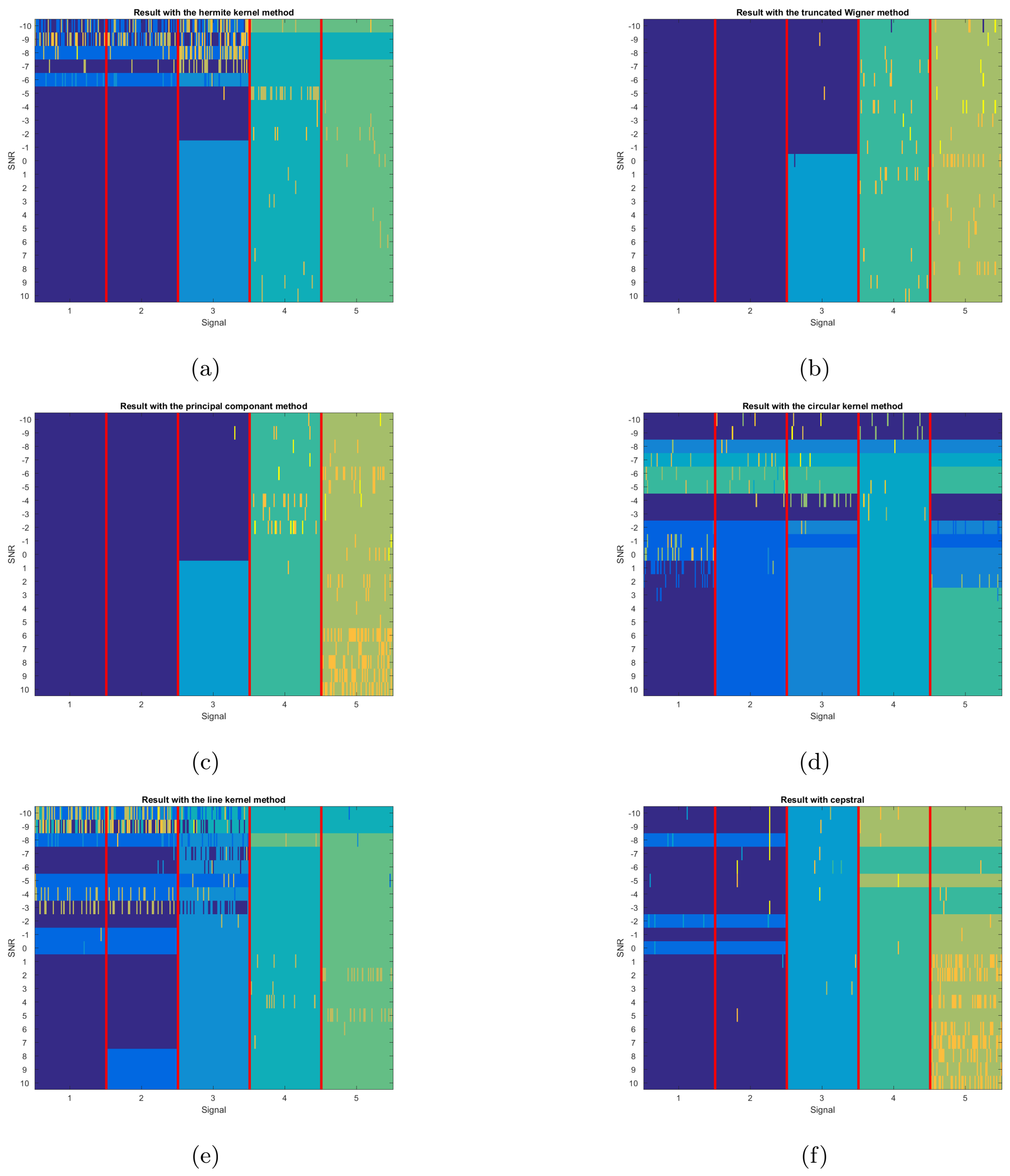
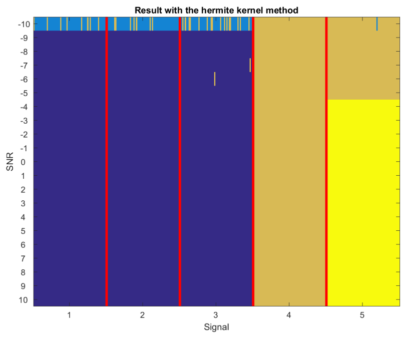
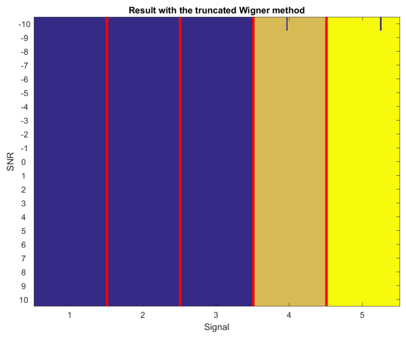


Figure 24: An image of how each signal has been classified at different SNR. Each pixel in the image corresponds to what class that signal has been classified into. For each SNR level the methods have classified the signals into the correct number of classes (five). The red lines show where the signals are supposed to change class; a), HKM; b), TWM; c), PCM; d), CKM with two feature vectors from the U and V matrices; e), LKM; f), MFCC.

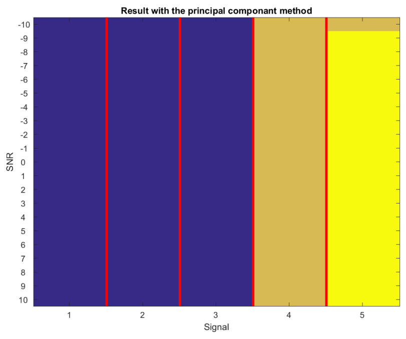
In figure 25, we see the same 300 signals for different SNR. However, now the number of classes is determined with the silhouette value method.



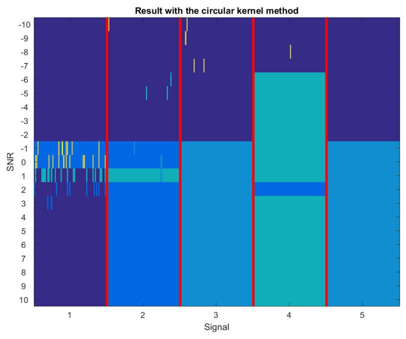
(a)



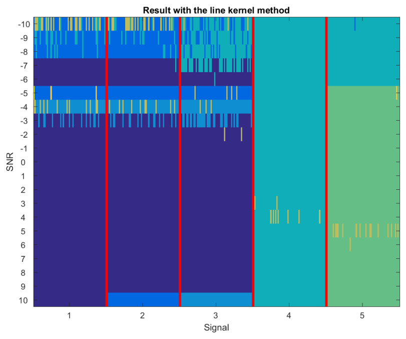
(b)



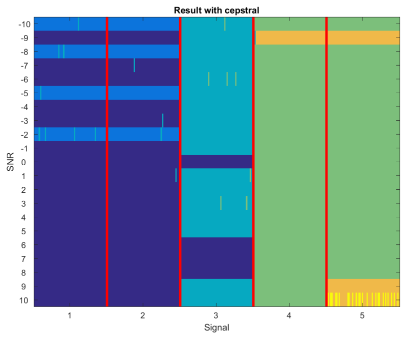
(c)



(d)



(e)

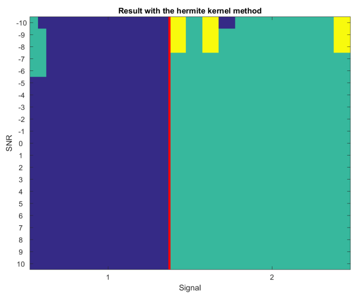


(f)

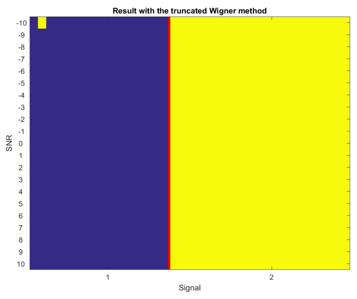
Figure 25: An image of how each signal has been classified at different SNR. Each pixel in the image corresponds to what class that signal has been classified into. For each SNR level the number of classes have been determined with silhouette values based on features from each method. The red lines show where the signals are supposed to change class; a), HKM; b), TWM; c), PCM; d), CKM with two feature vectors from the U and V matrices; e), LKM; f), MFCC.

8 Results from real data

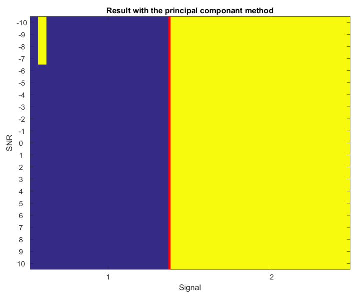
In figure 26, we see how each signal from the first real dataset has been classified in different SNR levels. On the x axes, we have all 39 signals from our set and on the y axes we can see how it has been classified. The number of classes is determined using the silhouette values.



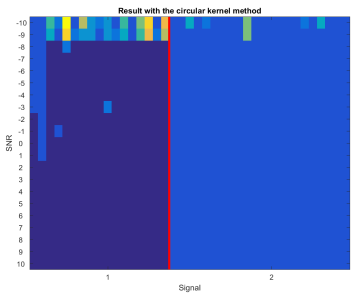
(a)



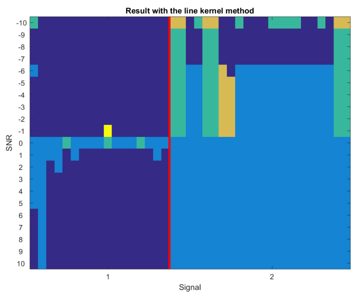
(b)



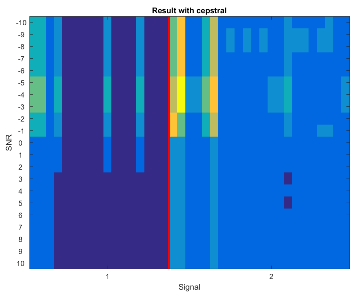
(c)



(d)



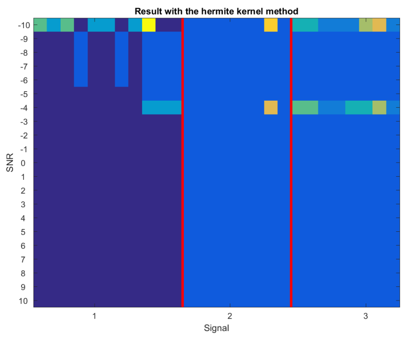
(e)



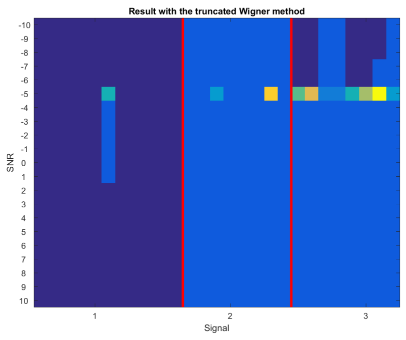
(f)

Figure 26: Figures of how each signal from the first real dataset has been classified with the different methods in different SNR.The color of the pixel correspond to what class the signal has been classified into. The number of classes is determined using the silhouette values and the red line shows where the signals is supposed to change class; a), HKM; b), TWM; c), PCM; d), CKM with three feature vectors from the U and V matrices; e), LKM; f), MFCC.

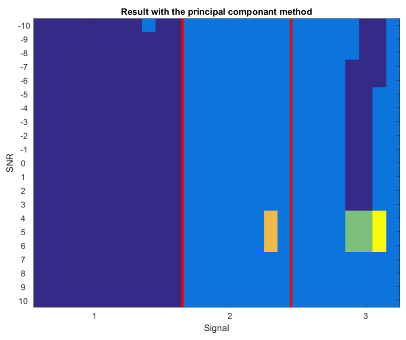
In figure 27, we see how each of the 27 signal from the second real dataset has been classified with the different methods when the number of classes is determined with the silhouette values.



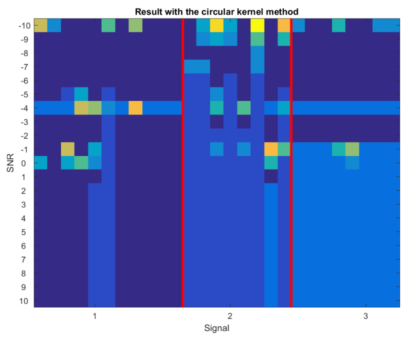
(a)



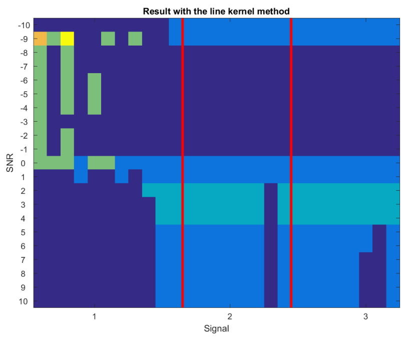
(b)



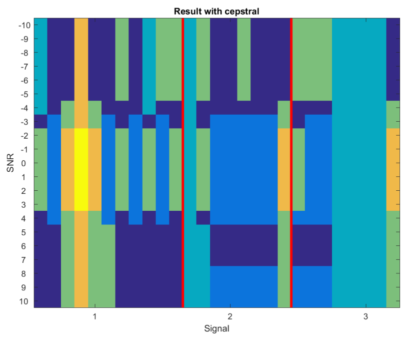
(c)



(d)



(e)



(f)

Figure 27: Figures of how each signal from the second real dataset has been classified with the different methods in different SNR.The color of the pixel correspond to what class the signal has been classified into. The number of classes is determined using the silhouette values and the red line shows where the signals is supposed to change class; a), HKM; b), TWM; c), PCM; d), CKM with three feature vectors from the U and V matrices; e), LKM; f), MFCC.

9 Discussion

9.1 Results from simulated data

The HKM is a method that is user friendly and give robust results. In figure 23 a), we see that the top branches of the dendrogram tree do not change considerably for a SNR higher than -2 dB. During this interval, we can also see that there is a large distance between the branches separating the set into three respectively four clusters, in comparison to the distance between the other branches. This indicates that it is likely that the method would classify into three classes. This is confirmed by figure 25 a) where the HKM has classified each signal and determined the number of classes with the silhouette values. The HKM classifies the data set into three classes when the SNR is higher than -5 dB. This indicates, as we later on will see for some other methods, that the distance between the branches in the dendrogram has a high impact on the number of classes found by the method. Also shown in figure 25 a) the HKM classifies classes one, two and three as the same. However, class four and five are well separated from the other classes when the SNR is higher than -5. Recall that the structure in classes one, two and three are very similar, namely a large component with a high amplitude followed by one or several smaller components at different frequencies. The HKM finds these very similar to each other which makes it harder for the method to separate the first three classes. Classes four and five both consist of relative large components and are then easier to separate. If we force the method to classify into five classes, seen in figure 24 a), we can see that it still can not separate classes one and two but are able to classify class three as a separate class. The extra components in class three makes it easier to separate them from the others.

The PCM method is very similar to the HKM except for an extra noise reduction step in the process. We will also see that the results from the two methods is very simular. In figure 23 c) we see as, for the HKM, that the distance between branches three and four is quite large, much larger than the distances between the other branches. This is true even for lower SNR indicating that the method is less affected by noise. In figure 25 c), we have a very similar result as for the HKM but more consistent in different SNR. However, this comes to a price when we force the method to classify into five classes. In figure 24 c), we see that the PCM, as the HKM, is able to distinguish class three from class one and two. However, there is more misclassifications in class five for higher SNR and less misclassifications on lower SNR. It seems that when there is lower noise levels the PCM will destroy more of the original signal and remove important information. However, on lower SNR it produces a lot more robust result since the noise reduction technique is focused on the noise. We see this in class one, two and three where the HKM changes its classifications considerably while the PCM do not. The sensitivity of the PCM is determined by the number of columns from the U matrix we use in the projection, that is how many dimensions we project on. The lower number of dimensions the WD is projected on, the more noise, as well as information from the original signal, will disappear. When using this method on a signal with little noise its therefore convenient to increase the number of dimensions. This will of course be a consideration between noise robustness and misclassification.

A method similar to the PCM is the TWM. In figure 23 b), we see that the dendrogram behaves very similar to the HKM's dendrogram. The branches giving four, five, and six classes are very close to each other while the branch giving three classes is much larger. Notice that the branch curve for three classes increases slower than for the HKM and PCM indicating that this method is less affected by noise. The gap between branches three and four indicates that the method will classify the set into three classes which also is confirmed in figure 25 b). Here, we see that the TWM performs very similar to the PCM. When we force the method to classify into five classes the result is not very different, see figure 24 b). In comparison to the PCM, the TWM has less misclassifications for all SNR levels. If we compare to the HKM, the TWM performs better in lower SNR, but worse in higher SNR where it is more misclassifications. The reason for this is that even when there is almost no noise in the signal, these methods will still try to reduce noise. However, when there is no noise to remove the TWM will rather remove important information from the signal. Since the signals are not identical within a class, they will be destroyed in different ways which lead to different representations. On the other hand, in lower SNR the TWM will rather remove noise and not the original signal.

There are further disadvantages with the TWM method. The first one is that when truncating a function in the Wigner domain, the probability of losing important components from the original signal is quite high. If we study the different classes we use in figure 20 we see that class one, two and three all consist of one large component and several smaller ones. When truncating the FWD, those smaller components could disappear and leave the large one as the only remaining. This will make it even harder to separate between classes like one and two where the only difference consists of a small component. These components will probably will be lost in the process. Another factor that could lead to degraded results come from the fact that when truncating the FWD, we introduce discontinuities in the function. These discontinuities will introduce new frequencies that will be shown in the FAMD and interfere with the features. The PCM on the other hand uses the vectors from the U matrix which will be continuous, resulting in that the filtered result also will be continuous.

The last method with focus on the auto-terms is the LKM. One big advantage with this method is that its kernel is easily represented with few windows. This makes the method fast. The LKM also performs very similar to the HKM in higher SNR. In figure 25 e) we can see that with a $SNR = 10$ dB the method is able to separate each class. This is the only method able to do so when determining the number of classes with the silhouette values. On the other hand, in lower SNR the LKM encounter problems with classes one, two and three.

The method that performs quite differently from the other is the CKM. In figure 23 d), we can see that the dendrogram for the CKM is more spread between branches in higher SNR. This indicates that it is easier to find more classes and separate between them in the data set. For lower SNR, on the other hand, the branches come closer to each other indicating that the method will have troubles to differ the classes. This is confirmed in figure 25 d). When determining the number of classes by silhouette values for high values on SNR, the CKM is able to separate into more classes than the other methods. The method separates between class one, two and three very well as long the SNR is higher than 2 dB. As the other methods, it can also separate between four and five. However, unlike the other methods the CKM fails to separate between class three and five. This is explained, considering what the CKM actually does. Unlike the other methods, this method focuses on the cross-terms which are determined be the differences between each component in the signal. As seen in figure 20, both signals in class three and five contain very similar frequency jumps. Both classes have two components with a frequency difference of size 0.2 and with a similar time difference. This will result in similar cross-terms for the two classes, making them hard to separate.

Since the circular kernel has a fix radius corresponding to cross-terms between components located nearby each other, lots of information will be lost in the analysis. For example, in the case of class five this kernel will include cross-terms between component two and three but not between component two and four. If we compare this to the other methods, none of them have the same problem. Class two and four both contain a frequency difference of size 0.1 but since class four also have jump of size 0 this is not a problem.

For the CKM it is harder to get an optimal result due to the number of parameters to determine. The risk is that we end up with a kernel that needs to many windows in order to be represented in a correct way. As discussed before, it is crucial not to include the auto-terms which is forcing us to use many windows if we want the whole kernels area. In our case, we have chosen only to use a small part of the original kernel as seen in figure 17. This will restrict us to only compare components located with a fix time interval between each other.

When exposed to signals with low SNR the performance of the CKM becomes worse. At a SNR of 0 dB, the method break down completely and at a $SNR = -2$ dB it can not separate anything. Since the cross-terms come with a lower amplitude than the auto-terms they are more affected by white noise and will easily be overshadowed.

The MFCC method does not perform as well as the other methods on our simulated dataset. As we see in figure 23, the dendrogram is quite compact for all SNR with a trend of being lower with a lower SNR. These higher gradients in the branch curves indicate that the method is not very noise robust. When the number of classes is set with the silhouette values it can anyhow separate between some important classes. Class three is very well separated from the other classes. As many other methods it can not separate between class one and two but unfortunately it has also problems with separating between class four and five which almost every other method could do.

9.2 Results from real data.

If we study the first real data set, we can notice that the two different type of signals are very different from each other. The first signal consists of relative few components in comparison to the second one, see figure 21. The first one is a whistle sound while the second is a rattle sound which makes this somewhat a minimum criteria for that the methods should be able to separate. In figure 26 we can see how each method has classified the data set for different SNR. Here, the HKM, TWM, PCM, CKM and the LKM classify the set very well for high SNR where almost none have any misclassifications. The MFCC method on the other hand classifies the set into to many classes and have more misclassifications. For lower SNR, the PCM and the TWM are unchanged and perform better than the HKM. This indicates that the noise reduction techniques works well. The CKM has more misclassifications but is not much affected by the SNR.

The results are quite different when the methods classify the signals from data set number two. This data set contains signals with only rattle sounds, which makes them harder to separate, see figure 22. The results from this is seen in figure 27. For high values on SNR, the CKM is the only method that are able to find three different classes in the set and separates them quite well. The HKM, TWM and PCM can only separate the first class from the others. Notice in figure 22 that the differences between class two and three are the length of the last component and a small change in the first component. The CKM specializes at the relationship between components with a fixed time interval between each other; in this case a distance of approximately 230 samples in the signal. This means that the CKM will detect cross-terms between component one and two in class three. In class two on the other hand is the two components to far away from each other. When extracting features from the auto-terms, as in the case of the HKM, TWM and the PCM, these differences will still be present in the ambiguity domain but not give as large impact as for the CKM. The reason is that the auto-terms in the ambiguity domain will be located at the same place and not move as in the case of the cross-terms.

The LKM performs very poorly on this dataset. It can almost separate class one from the others for high SNR but give more misclassifications than HKM, TWM and PCM. For lower SNR it breaks down completely. The MFCC method can not separate between the classes at all.

10 Conclusions

When comparing different methods we can conclude that the performance of a method depends on the signal and SNR. Methods such as the TWM and the PCM, which focus on using the Hermite windows and reducing noise in the Wigner domain, improve the robustness of classification. However, these methods needs to be selected with care since they behave worse for low levels of noise. These methods along with the HKM work well when used in a dataset that have clear separate classes, like in the case of the first real dataset. However, when looking on a dataset where the different classes are more similar to each other, these methods lack accuracy. This is illustrated in the case of the second real dataset and between class one and two of the simulated dataset. Here, a method like the

CKM perform better since small variations between signals lead to bigger differences in the ambiguity domain. The CKM is the only method that can separate all sets fairly good. This comes to the cost of robustness since this method can not handle very noisy environments. This method also has problems when it comes to implementation. Since the location of the cross-terms could vary between each signal it is important to have a kernel covering all possible locations. This on the other hand is not a simple task since a large kernel will make the method computationally slower.

The LKM gives quite promising results when used on the simulated data where it finds all five classes for high SNR. On real data on the other hand this method degrades and is worse than the HKM. In comparison to the other ambiguity based methods the LKM is not good enough in order to be a reliable method. The other methods are better in almost every other way.

We can conclude that when these methods classify datasets it is important to check the results by hand. The main reason for this is that many methods have problems to determine how many classes present in the dataset. We see this clearly in the results from the simulated dataset where many methods perform better when we force them to classify into the correct number of classes. For example, the CKM classifies this dataset perfectly for higher SNR when we force the method to classify into the correct number of classes.

11 Further investigations and improvements

From this thesis, there are many things that would be interesting to further investigate and that I think will improve the methods. When it comes to CKM, one of its disadvantages is that the parameters is very dependent of the signal and can be hard to determine. There is however a possibility to make an adaptive method that estimates the optimal parameters in order to include the cross-terms. This can for example be done with a method that find each peak in the syllable, in a similar way to the method described in chapter 2.1. From this we can calculate for which τ there will be a cross-term located. The same can be done with ν if we estimate the frequencies of each peak. After estimating where each cross-term will be located it's possible to estimate R and α in order to cover these areas. This method has been tested on simulated data and was able to estimate the location of the cross-terms quite well for higher SNR. However, I was never able to get these methods robust enough to be used in practice. Problems also occur when determining the number of multitaper windows to use in order to get a good representation of the kernel. However, the solution of this problem is outside of this project.

The results for the CKM also depend very much on how many and what feature vectors we use when representing our signals. My way of solving this is to assume that when comparing two signals the two most important vectors are the first two in the U and V matrices. We then find the corresponding ones from the other signal. There is however no guarantee that the first two vectors will be the most representative, especially not for low SNR where the vectors have a tendency to shift places. A method that find the optimum vectors from each signal could improve this method even more.

12 References

Everitt B., Landau S., Leese M. & Stahl D. (2011). *Cluster Analysis.* Wiley.

Grosse Ruse M., Hasselquist D., Hansson B., Tarka M. & Sandsten M. (2016). Automated analysis of song in complex birdsong. *Animal Behaviour. Nr. 112, p 39-51.*

Jancovic P. & Köküer M. (2011). Automatic Detection and Recognition of Tonal Bird Sounds in Noisy Environments. *EURASIP Journal on Advances in Signal Processing.* Vol. 2011, Article ID 982936, 10 pages.

Keen S., Ross J., Griffiths E., Lanzone M. & Farnswoth A. (2014). A comparison of similarity-based approaches in the classification of flight calls of four species of North American wood-warblers (Parulidae). *Ecological Informatics.* Vol. 21, p 25-33.

Kroodsma D. (2017). Birdsong performance studies: a contrary view. *Animal Behaviour.* Nr 125, 16 pages. MS. number: AF-16-00107R.

Sandsten M. & Brynolfsson J. (2017). Classification of Bird Song Syllables Using Wigner-Ville Ambiguity Function Cross-Terms. *Proceedings of the 25th European Signal Processing Conference (EUSIPCO)*, p 1739-1743.

Sandsten M. (2018). *Time-Frequency Analysis of Time-Varying Signals and Non-Stationary Processes. An introduction.* Centre for Mathematical Sciences, Lund University.

Stankovic L. (2002). Analysis of Noise in Time-Frequenzy Distributions. *IEEE signal processing letters*, vol 9, nr 9, p 286-289.

Janes S., Ryker L. & Ryan R. (2017). Geographic variation in type 1 dialects of Hermite Warblers: does fragmented habitat promote variation in song? *J Ornithol.* Vol. 158, p. 421-430.

Wegrzyn E., Leniowski K. (2010). Syllable sharing and changes in syllable repertoire size and composition within and between years in the great reed warbler, Acrocephalus arundinaceus. *J Ornithol.* Vol. 151, p 255-267.