Automatische Erkennung und Klassifizierung von Nachtigallengesang

(Automatic Classification of Nightingale Songs)

Bachelorarbeit im Fach Informatik

vorgelegt von

Name: Zagler Vorname: Maximilian

Geb. am: 06. März 1983 in: Hamburg

angefertigt am

Institut für Informatik
Freie Universität Berlin

Betreut von
Prof. Dr. Raúl Rojas
Tim Landgraf

Berlin, den 23. Februar 2010

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Datum Unterschrift

Contents

1 Introduction

Nightingales are well known as exceptionally diverse singers. This and the influence of the nightingale's environment on its singing is a matter of ongoing research for biologists.
Due to the complexity and the high amount of different songs a mightingale may produce, a manual classification of these songs is a very time consuming task. To support the work of biologists, in this thesis two approaches for an automatic nightingale song classifier are evaluated. First, an approach based on sequence alignment of spectrograms is described, followed by an approach based on sequence alignment of symbols representing the individual notes of the songs.

2 The Structure of Nightingale Songs

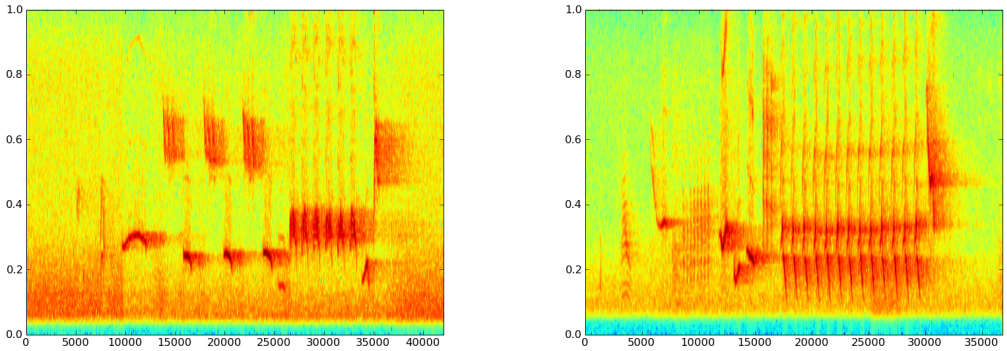
Nightingales are capable of learning up to 260 different song types [1]. and can reproduce these song types with astonishing accuracy. These song types can be categorized into two main classes: whistling and nonwhistling songs. It is assumed that these two groups of song types have different functions. The nonwhistling song is more complex and contains more variety in the selection of its elements [1]. A song can be further divided into syllables and notes. While the nightingale is capable of reproducing a learned song precisely, usually there are some elements in each song may be repeated. This can is shown in fig. 1 and 2.

For this thesis, about 400 recordings of separate songs and a three hour recording containing about 1500 songs were available. The recordings were made in a public park (Treptower Park, Berlin, Germany) and therefore some noise, especially traffic noise is present. The recordings have a sampling rate of 22,050 kHz and a quantization of 16 bit.

3 Sequence Alignment of Spectrograms

The analyis of an audio signal is usually initiated by calculating a spectrogram of the signal, which represents the energy distribution in the frequency domain over time as seen in fig. 1.

To obtain a spectrogram, the audio signal is split into fixed size intervals, so called frames, and on each frame the discrete fourier transform is calculated. Usually these frames are allowed to overlap to a certain amount to improve the resolution in the time domain. A more detailed explanation of this procedure can be found in [2].



(a) (b)

Figure 1: Two examples demonstrating the great diversity of notes within one song.

So, representing the recording of a nightingale song as a spectrogram results in a sequence of frequency spectra for successive periods in time. These frequency spectra give access to information of the audio signal encoded in the frequency domain, e.g. frequency based characteristics or features of the audio signal, such as how the pitch or brightness in a song changes over time.

Using this kind of information, a nightingale song can be expressed as the evolution of such features and the problem of estimating the similarity of two song recordings is reduced to the problem of estimating the similarity of sequences of feature vectors.

3.1 Sequence Alignment Algorithms

Intuitively a similarity between two sequences can be estimated by measuring the similarity independently, feature vector by feature vector, and deriving some overall similarity for the songs, but the problem is obvioulsy complicated by the fact that sequences stemming from natural recordings have a high degree of variation in length. Also, the sequences might not be perfectly aligned to begin with.

Solutions for a very similar problem were devised in research areas concerning automatic speech recognition in the early 1970s, which delt with the task of matching words of different length caused by a different speech rate as explained in [3].

The problem can be approached by stretching and compressing the alignment dynamically by inserting gaps in the sequences at positions where a match is not justified, so that both sequences end up having an equal length (for aligning

purposes that is).

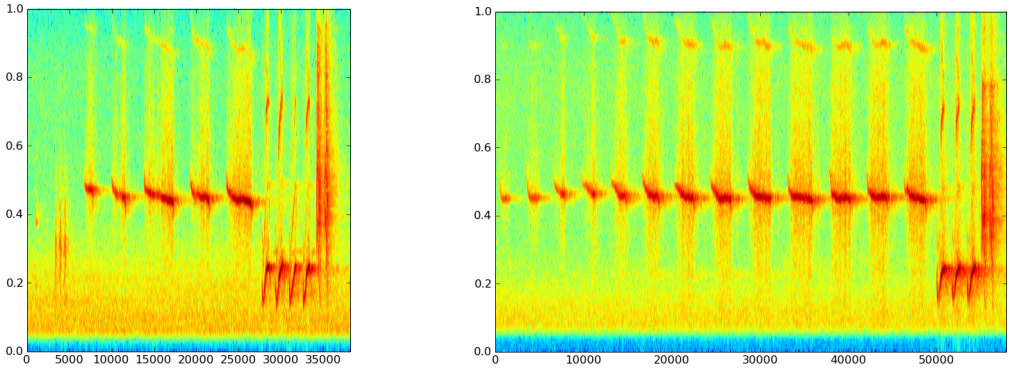
The question of course is where these gaps should be inserted and which frames should be matched to obtain the best overall alignment. This is a typical kind of optimization problem that can be solved using dynamic programming techniques establishing a class of time warping algorithms [3].

as the DTW in speech recognition or the Needleman-Wunsch and Gotoh algorithms in bioinformatics [3] but they all operate on the principle outlined here:

```
def align(seq1, seq2):
    costs = matrix(len(seq1), len(seq2))
    mat[0, 0] = 0
    for i in range(len(seq1)):
        costs[i, 0] = infinity
    for j in range(len(seq2)):
        costs[0, j] = infinity
    for i in range(1, len(seq1)):
        for j in range(1, len(seq2)):
            costs[i, j] =
                min (
                    costs[i - 1, j] + gap_costs,
                    costs[i, j - 1] + gap_costs,
                    costs[i - 1, j - 1] + distance(seq1[i],
                        seq2[j])
                )
    return costs[-1, -1]
```

A cost matrix is constructed row after row where each entry is the sum of the local temporary costs so far and the minimal costs of either inserting a gap in one of the two sequences or matching the current frame. In the end, the last entry in the right matrix corner holds the the overall alignment costs. Backtracking from this entry gives the actual alignment. The alignment cost may be normalized by dividing to the length of the alignment path as suggested in [2]. Deciding on the distance measure and the gap costs is mostly task specific. Such is the value for the matrix initialization.

As hinted earlier, there are different ways to utilize the spectrogram to generate meaningful features. But since it is natural to expect two similar nighingale songs showing a similar progression of their respective energy distributions, the fourier transform on its own performs a fundamental feature generation. Although the



(a) (b)

Figure 2: Two examples demonstrating the great variation in the amount of repetition in songs of the same type.

DFT has a high degree of redundancy, for a start it is a reasonable choice to use the fourier coefficients directly as feature vectors as suggested in [2]. As a distance measure for the feature vectors, simply the euclidean distance between the feature vectors was used.

In this thesis the Gotoh sequence alignment algorithm was used which allows a separate treatment of gap creation and gap extension costs [4]. The gap creation should be more expensive than the extension of a gap. Once a single frame cannot be matched, usually several succeeding frames won't match either because the first missing frame will most likely appear in the context of a missing note or a missing sequence of notes.

3.2 Handling Element Repetitions

A problem that these algorithms do not solve, stems from the particular structure of the nightingale songs. As mentioned before nightingale songs are higly repetitive (see fig. 2), in fact it is one of their most important characteristics, and since the amount of repetition varies for each instance of a song type, similar song types may result in vastly different song lengths.

The sequence alignment algorithm used, produces a gap when one of two similar sequences contains more repetition than the other. In such a case the gap should represent a repetition, but really it represents any kind of signal since the alignment is unaware of the content. The result is that similar songs with a different amount of repetition may result in a higher dissimilarity than two different

songs with a more similar length, but gaps representing repetition should have no cost at all. Adjusting the gap costs at this point doesn't really help the problem, it is necessary to analyze the content of the gap.

One approach would be, to do the alignment as described, but match the resulting gaps with the sub sequence preceeding the gap in a second step. If the gap contains a repetition, this second alignment step would result in lower costs than an alignment with a completely unrelated sequence. This way the overall alignment costs could be increased and hopefully the sum of these two alignment steps would result in a lower overall cost for instances of a similar song type.

Another approach would be, to rely on a separate pre-processing stage that would remove all repetitions from a song. This way, a gap would necessarily appear at dissimilar sequence ranges and the result of the alignment described would work as intended. Such an approach has the advantage that it forms a task that can be clearly separated from the alignment. Detecting repetition in one sequence might be easier than detecting it in two different sequences. Removing the repetitions from the sequence before the alignment would make the alignment more efficient. Also, the database of known songs would shrink dramatically since repetitions would not have to be stored.

Unfortunately, in the scope of this thesis, these approaches could not be pursued any further.

4 Sequence Alignment of Symbolic Representations of Song Elements

Despite the fact that repetitions cause problems during aligning, it appears as if it might be worth persuing song classificaton based on a sequence alignment of spectrograms. On the other hand this approach does not offer much computation and information reuse. Of course, in the end each unknown song must be matched with every song in the database, but the way the song is represented in the database and to the aligner can have a huge impact on the performance of the system as well as the amount of information the system can offer the user. The approach discussed so far uses a database which basically consists of the original recordings and all their inherent redundancy.

First of all most of the fourier coefficients are redundant. A more sophisticated feature extraction stage would extract features such as brightness and pitch of the signal which could dramatically reduce the database size.

Also, so far only two entities were used to describe the singing of the nightingale, songs and DFT-Frames. But the nightingale structures its singing in smaller entities than songs but surely doesn't care about DFT frames, so it is probably not the best entity to quantify a nightingale song. Actually, the structure of a song described earlier was not utilized at all.

A second approach tries to address these problems by converting each song into a sequence of symbols representing the sequence of notes of the nightingale song.

Utilizing the structure of a nightingale song, in a first step a dynamic segmentation of each song in the database is performed, slicing the song at note boundaries into separate segments of different length.

In the next stage, features are extracted from the segments and for each segment a feature vector is constructed. At this point, each song is still represented as a sequence of feature vectors.

Next, all feature vectors are clustered and each cluster is given a label. The clusters represent the alphabet of the nightingale, at least as seen by the segmentation and feature extraction algorithms. Each label is used as a symbol representing all feature vectors belonging to the cluster.

Now each song in the database - still represented as a sequence of feature vectors - can be converted to a sequence of symbols by measuring the distance of each feature vector in the sequence to each cluster and selecting the label of the

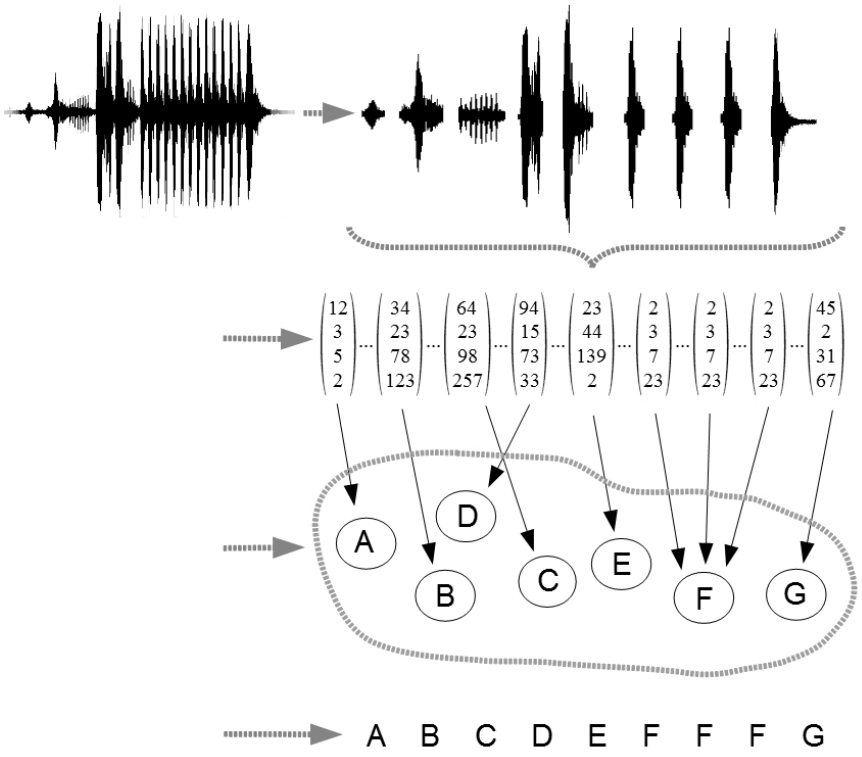


Figure 3: The conversion of an audio signal to a sequence of symbols.

nearest cluster (see fig. 3).

To finalize the database, for quick lookup the dissimilarities between all symbols are computed as the distances between all clusters.

Unknown songs are converted the same way before searching the database. The same sequence alignment technique as before is used, but now symbols instead of feature vectors are used and their distances have been computed beforehand. Also the sequences are a lot shorter due to the dynamic segment length.

4.1 Segmentation Algorithms

Automatically segmenting nightingale songs is by no means a trivial task, but before examining different ideas to approach this problem, there is a basic question that needs to be resolved first: What actually constitutes a segment in a nightingale song - ?

Clearly, it is necessary to decide upon the notion of a segment before a segmentation can be performed. At this point it is helpful to distinguish between a segmentation on a semantic level, respecting the meaning of the signal, and a segmentation that bases its decision on certain properties of the signal only, ignoring the context in which the signal was obtained.

In this thesis an emphasis is put on energy based segmentation algorithms performing a segmentation on a note level. The note onset, the beginning of a note is defined as transition from a period of low energy to a period of high energy, the note offset, accordingly. A segmentation is executed in a low energy valley that may be the result of deliberate pauses between semantically separate elements within a song or stemming from an adjustment of the nightingale's vocal apparatus. It should be noted that the information of deliberate pauses within a song is not preservered during segmentation.

The algorithms used process the energy curve of the signal, computed as the sum over the DFT-coefficients in each frame of the short time fourier transform of the song. The energy curve was smoothed to reduce the influence of noise on the segmentation and normalized.

4.1.1 Segmentation by Peak Finding

An obvious approach in energy based segmentation works like this: An onset is detected, when the energy raises above an appropriate threshold T, once it falls below T again, the corresponding offset is found. The challenge of course, lies in deciding on the threshold T. As discussed, the singing of the nightingale has a very high dynamic range and a second glance at suggests that it will most likely not suffice to use a static global threshold for a song, e.g. based on the song's mean energy.

Estimating a dynamic threshold for segmentation basically means evaluating a local range around a certain point of interest, e.g. a potential note onset or offset. Certainly, it is easier to find note boundaries, once a point inside a note is already known. Since a note is supposed to be a period of high energy, the energy peak in a song can be assumed to be inside a note and therefore a reasonable starting point for segmentation.

This way, segmentation can be reduced to a basic peak finding problem. A similiar approach is used in [5]. The peak value can be used as an indicator for an (initial) local threshold. From the peak value, the algorithm clears points from the energy curve to the left and the right until the note boundaries are found, which the energy curve falls under the local threshold (e.g. a fraction of the peak value) which would indicate a note boundary.

Here is a pseudo code for this algorithm:

```
def peak_find_segmentation( energy ):
    segments = []
    while peak > global_threshold( energy ):
        left = right = peak_index
        while energy[right + 1] > right_threshold( energy[
            peak_index : right] ):
            right += 1
        while energy[left - 1] > left_threshold( energy[left :
            peak_index] ):
            left -= 1
        segments.append( (left, right) )
        remove_segment( energy, (left, right) )
    return segments
```

The implementation mainly differs from [5] in that a global threshold is used to abort segmentation instead of a fixed number of segments, and the overall energy curve is used. Also, the local threshold was computed not only utilizing the peak value but is smoothed by the local mean value.

Despite its simplicity this algorithm performed quite well.

4.1.2 Segmentation with the Slope of the Signal

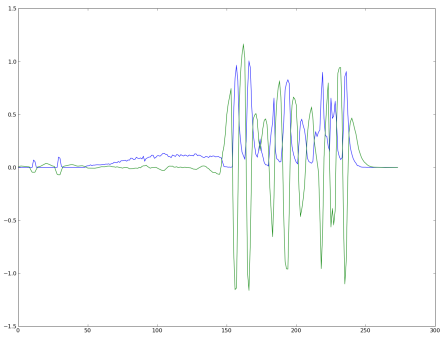
A more sophisticated energy based approach is described in [6]. Here, more emphasis is put on the actual shape of the energy curve.

In each point of the energy curve, the slope is estimated by a linear regression anchored at that point, going through windows to both sides of that point. The windowsize is chosen so that the sum of euclidean distances between successive points does not exceed an experimentally decided value.

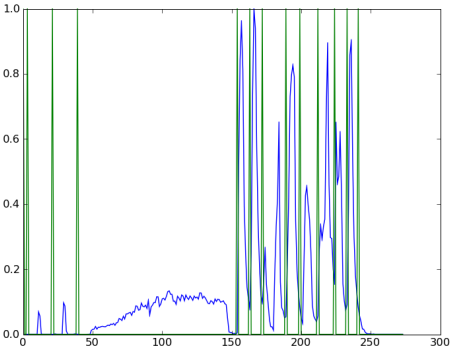
Then the slope curve is thresholded with the original energy curve that may be scaled and shifted by a global threshold. In ranges where the slope curve is above the threshold, the peak value of the slope cuve is used as a segment boundary (see fig. 4).

In addition to that the resulting segments were truncated with the segmentation method described earlier to remove the background signal at the beginning and the end of the segment, even though this may sometimes cut part of the signal.

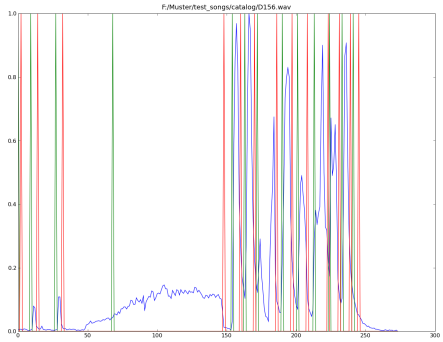
Overall, the second approach performed slightly better but it was difficult to choose the values for window size, scaling of the threshold curve and the offset to receive a uniformly good segmentation. The parameters of the segmentation



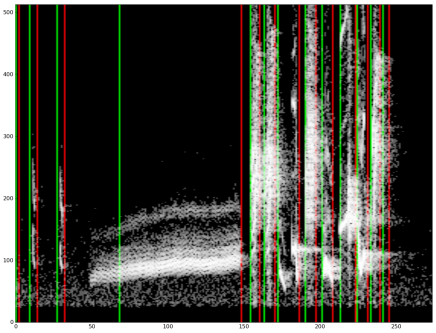
(a) slope (green) and threshold (blue)



(b) segment boundaries (green)



(c) note onset (green) and offset (red)



(d) specttrogram with note boundaries

Figure 4: Different representations of the slope based segmentation algorithm.

algorithms were tuned manually and were gradually improved on a trial and error basis.

It is necessary to understand that the segmentation resulting from both algorithms may differ from a segmentation a person might decide upon, mostly since only a single characteristic of a song was utilized: overall energy. But melodic information is an important feature to differentiate between segments and ignoring it during segmentation can result in missing a note boundary, e.g. a smooth transition from low to high pitch is not necessarily accompanied by a significant change in energy.

Another import criterion a person will surely use during segmentation is the repetition of the song. As can be seen the repetition pattern suggests grouping different elements together to treat them as a single unit even though that wouldn't be a segmentation on a note basis.

4.2 Feature Extraction

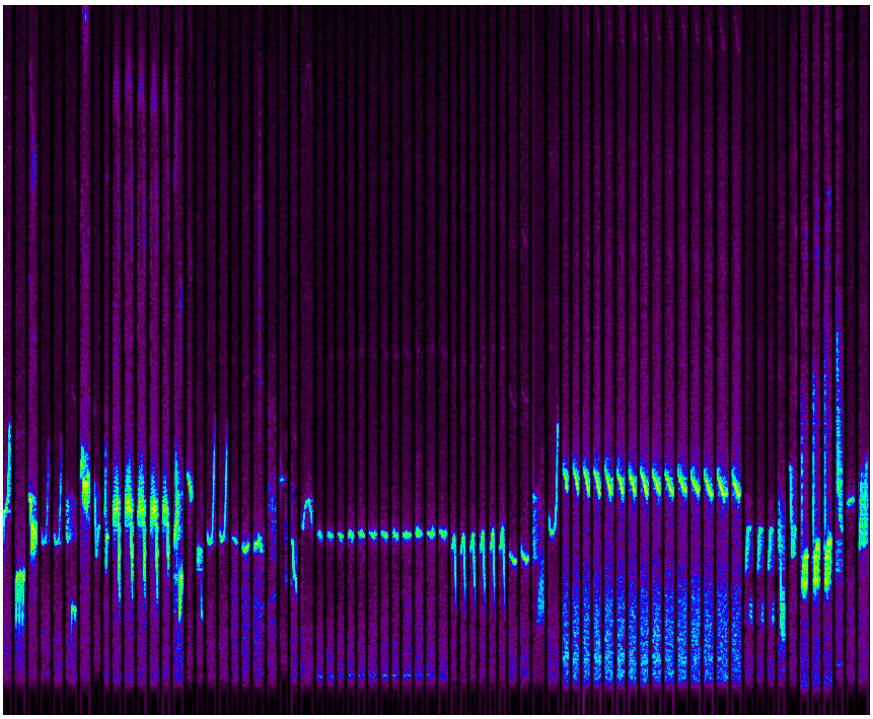
In contrast to the first method, here a more sophisticated feature extraction stage is used. The segments are mostly characterized by the energy distribution in the frequency domain over time. Some basic features derived from this energy distribution are well known for audio analysis:

- the spectral centroid, a measure for the brightness of the signal [2]:
  $C(i) = \frac{\sum_{m=0}^{N-1} m|X_i(m)|}{\sum_{m=0}^{N-1} |X_i(m)|}$
  where $X_i(m)$ is the $m-th$ frequency (DFT-Coefficient) of the $i-thframe$.

- the spectra flux, describing spectrum changes from frame to frame [2]:
  $F(i) = \sum_{m=0}^{N-1}(N_i(m) - N_{i-1}(m))^2$
  where $N_i(m)$ is the $m-th$ normalized magnitude of the $i-thframe$.

Since the segments may have an arbitrary length whereas all feature vectors need to have fixed dimensions, it is necessary to express the evolution of the feature with a fixed number of values. There are different ways to approach this problem:

One solution is to downsample the segment to a fixed number of frames. A static downsampling might cause significant information that appears between samples to be lost. Also the segmentation algorithm must produce perfectly aligned segments for this.



(a)

Figure 5: The content of a single cluster, demonstrating the problems that were encountered during clustering.

A second approach is to fit the curve with a polynom or simply a line. Depending on the framesize of the DFT it might not be possible to obtain useful fitting.

Finally, it is possible to ignore the evolution altogether by averaging all spectra of the segment's spectrogram. Another feature tried was the mean and the variance of fitted gaussians through two or three of the main peaks in such an average spectrogram.

Each of these approaches lose the information on the length of the segment, so it is always necessary to make the length an additional feature.

4.3 Clustering

All feature vectors were whitened as described in [7]. For the clustering, the methods K-Means and Expectation Maximization Clustering were available.

After different combinations of features, downsampling and polynom fitting were evaluated it was found that this approach failed at the clustering stage which was not able to separate the feature space satisfactory. Fig. 5 shows one of the resulting clusters.

It was not successfully determined why the clustering failed. It might have been a problem that all features were uniformly weighted. Also possibly the segments coming from the segmentation stage were too short, so that not enough data points were available, to determine a tendency with a line fitting. In other cases, the extracted features were only able to express some of the segments characteristics like the base frequency but not the more complex overtones responsible for the timbre of the segment.

It might also be a problem that the clustering algorithms used, force a static feature vector on segments of variable length. A different approach might be to perform a sequential clustering based on sequence alignment. The distance measure for the clustering could be the alignment costs instead of an euclidean distance. During clustering, a new cluster would be created if the distance to all other clusters exceeded a threshold. The distance to a cluster might be the mean distance to all members in the cluster or to a prototype. A cluster prototype might be the segment with the least alignment costs all other segments in this cluster. A similar approach is suggested in [8].

5 Conclusion

Two approaches at the classification of nightingale songs were evaluated in this thesis. The first simple approach had the problem that it was computationally expensive and it was found that basic sequence alignment algorithms did not work well with the repetitive structure of the nightingale's singing. Either pre processing of the song, removing the repetitions or by determining a measure to adjust the alignment costs by evaluating the content of the gaps will be necessary. In theory the second approach, a symbol based sequence alignment classification would be more efficient, and, due to dynamic segmentation, might provide more useful information about the structure of the song. Unfortunately this approach was not successfully implemented. Possibly a combined approach of dynamic segmentation and clustering based on sequence alignment of the segments could solve the problems encountered in the symbol based approach so that the feasibility of a such an approach could be determined.

References

[1] Hansjoerg P. Kunc, Valentin Amrhein, and Marc Naguib. Acoustic features of song categories and their possible implications for communication in the common nightingale. *Behaviour*, 142(8):1083 – 1097, 2005.

[2] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition*. Academic Press, 2006.

[3] Toni Giorgino. Computing and visualizing dynamic time warping alignments in r: The dtw package. *Journal of Statistical Software*, 31(7), 2009.

[4] Eugene W. Myers and Webb Miller. Optimal alignments in linear space. *Bioninformatics*, 4(1), 1988.

[5] Aki Härmä. Automatic identification of bird species based on sinusoidal modeling of syllables. Technical report, Laboratory of Acoustics and Audio Signal Processing, University of Helsinki, 2003.

[6] Ping Du and Todd W. Troyer. A segmentation algorithm for zebra finch song at the note level. Technical report, Neuroscience and Cognitive Science Program, Dept. of Psychology, University of Maryland, 2006.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag GmbH, 2006.

[8] Tim Oates, Laura Firoiu, and Paul R. Cohen. Clustering time series with hidden markov models and dynamic time warping. Technical report, Computer Science Department, University of Massachusetts, 1999.