Bird Song Recognition with Hidden Markov Models

Hugo Jacobus van der Merwe

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering (Electronic Engineering with Computer Science) at the University of Stellenbosch

Supervisor: Ludwig Schwardt

March 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....
H.J. van der Merwe

Date: .....

Abstract

Automatic bird song recognition and transcription is a relatively new field. Reliable automatic recognition systems would be of great benefit to further research in ornithology and conservation, as well as commercially in the very large bird-watching subculture.

This study investigated the use of Hidden Markov Models and duration modelling for bird call recognition. Through use of more accurate duration modelling, very promising results were achieved with feature vectors consisting of only pitch and volume. An accuracy of 51% was achieved for 47 calls from 39 birds, with the models typically trained from only one or two specimens. The ALS pitch tracking algorithm was adapted to bird song to extract the pitch. Bird song synthesis was employed to subjectively evaluate the features.

Compounded Selfloop Duration Modelling was developed as an alternative duration modelling technique. For long durations, this technique can be more computationally efficient than Ferguson stacks.

The application of approximate string matching to bird song was also briefly considered.

Opsomming

Die outomatiese herkenning van voëlsang is 'n relatiewe nuwe veld. Betroubare outomatiese herkenningstelsels sal van groot hulp kan wees in verdere navorsing op die gebiede van voëlkunde en bewaring, asook kommersieel in die baie groot voëlkykersubkultuur.

Hierdie studie ondersoek die gebruik van Verskuilde Markov Modelle en tydsduur modellering vir die herkenning van voëlsang. Ondanks besondere eenvoudige kenmerkvektore wat slegs uit toonhoogte en volume bestaan, is baie belowende resultate bereik deur van meer akkurate tydsduur modellering gebruik te maak. 'n Akkuraatheid van 51% is bereik vir 47 roepe van 39 voëls, wat tipies afgerig is vanaf slegs een of twee individue. Die toonhoogte is bepaal deur 'n ALS toonhoogte-afskatter algoritme wat aangepas is vir voëlsang. Voëlsang sintese is benut om op subjektiewe wyse die kenmerkvektore te evalueer.

"Compounded Selfloop Duration Modelling" (of Saamgestelde Selflus Tydsduur Modellering) is ontwikkel as 'n alternatiewe tydsduur modelleringstegniek. Vir lang tydsduurtes benodig hierdie tegniek minder berekeninge as Ferguson-modelle.

Die toepassing van string vergelykingstegnieke is ook ondersoek.

Acknowledgements

I would like to take this opportunity to thank the following people for their contributions during this journey:

- my study leader, Ludwig Schwardt, who gave me the freedom I needed to find myself during this study,
- Stefan vdW, Steve K and Marianne A for moral support at the times when I needed it most,
- Elsa for lending me an ear, thereby making an invaluable contribution to my shift in consciousness,
- and *everyone* else (and that's practically everyone) that contributed to keeping me sane during an insane time. Yes, you, you know who you are.

Contents

List of Figures

List of Tables

List of Abbreviations

ALS	Adaptive Least Squares
AM	Amplitude Modulation
CSDM	Compounded Selfloops Duration Modelling
DP	Dynamic Programming
DTW	Dynamic Time Warping (an example of a DP algorithm)
FFT	Fast Fourier Transform
FM	Frequency Modulation
HMM	Hidden Markov Model
pdf	Probability Density Function
RMS	Root Mean Squared, $\sqrt{\frac{1}{N}\sum x_i^2}$

Chapter 1

Introduction

This study investigated bird song recognition. In the process, it investigated applications of approximate string matching, HMM duration modelling techniques, pitch tracking algorithms and feature analysis by synthesis.

1.1 Bird Song Overview

While there are dozens of modern bird orders [33], more than half of all bird species are in the order Passeriformes [37], and known as passerines. There are more than 5000 species in this order, one of the most diverse terrestrial vertebrate orders. The order is further subdivided into two suborders, Tyranni and Passeri. There are about 4000 species belonging to the Passeri suborder. These species are called oscines, or songbirds [38].

Ornithologists distinguish between bird "songs" and bird "calls". The distinction is somewhat arbitrary, but "songs" typically refer to longer, more complex vocalisations, usually associated with courtship and mating, in comparison to alarm calls and calls used for keeping members of a flock in contact [6]. This document generally uses the term *bird song* to refer to all bird vocalisations, i.e. both songs and calls.

The organ that birds use to produce vocalisations is a syrinx. It is very different from the mammalian larynx. The larynx is located at the top of the trachea (windpipe), and contains hard membranes known as vocal chords. The vibration of these vocal cords is controlled by muscles and cartilage. A syrinx, on the other hand, is located at the lower end of the trachea, typically deep in the chest cavity, and is surrounded by an air sac. The syrinx becomes a resonating chamber in conjunction with highly elastic membranes. Syringeal muscles control the movement of the syrinx, including the tension in the membranes in similar fashion to the skin of a drum. Together with the tension in

the membranes, birds also alter the air pressure passing from the lungs to the syrinx [34].

In songbirds, it appears that the differences between different species' song result mostly from differences in the learning process, rather than the structure of their vocal apparatus. It appears that the singing ability of songbirds is largely dependent upon the size of the song control regions of their brains. Female Canaries and Zebra Finches have a substantially smaller song control region than their male counterparts, possibly explaining their inability to sing [34]. In many species of songbird, the males are the superior singers.

Songbirds learn their song from one or more adult birds through a gradual process over a period of weeks or months. In many species, there is a "sensitive phase" during which a bird is most sensitive to memorising songs. During or soon after this phase, vocal production begins in the form of *subsong*. Subsong has been compared to babbling in human infants. This vague, jumbled subsong is gradually transformed into a more structured *plastic song*. Plastic song is still quite variable. During this phase, birds start imitating their tutors, though imitations are often incomplete. Much more material is produced during this phase than eventually finds its way into the bird's stable repertoire of *crystallised songs* [7, 22].

While studies of non-passerine birds have been limited, it appears that their calls are typically innate rather than learned. In fact, precocial young (relatively mature and mobile from the moment of hatching) tend to have larger, better-developed repertoires of calls than altricial young (incapable of moving around soon after hatching, requiring care and feeding by parents) [7]. Most passerines, on the other hand, are altricial and learn their song from adults.

Because the songs of songbirds are so pure in frequency, lacking higher harmonics, it was originally thought that all variation in their song quality arises from the syrinx, and that resonances of the vocal tract played no part. This hypothesis contrasts greatly with human speech, where resonances of the vocal tract modulate the sound produced by the vocal chords, and create most of its informational content. The first evidence that showed that this is not the case, was provided by Nowicki in 1987 [23]. Bird song recorded in a helium atmosphere showed higher harmonics, indicating that in a normal atmosphere, the vocal tract does act as an acoustic filter. This filter suppresses the second harmonic, regardless of its absolute frequency, indicating that the filter characteristics are coordinated with the output of the syrinx. The exact mechanism was not certain until more recently.

It was thought that the size of the beak opening, which in adult birds is typically correlated with the frequency of the song, may have the effect of shortening or lengthening, and thereby tuning, the vocal tract. A counter-example

for this hypothesis is found in song sparrows. Juvenile song sparrows initially have prominent higher harmonics in their song. During the late phase of plastic song, the harmonics become less prominent and the song develops its adult-like tonality. This occurs despite the bird not having learned to coordinate its beak movements yet. Riede et al [25] used x-ray cinematography to investigate the vocal tract of birds while singing, concluding that the tuning of the major resonance of the oropharyngeal-esophageal cavity actively tracks the song's fundamental frequency.

Such vocal tract filtering is found mostly in the passerine birds. Non-passerine birds often have harmonics present in their song, with their song timbre often being a defining characteristic. For example, the Hadeda Ibis' call has many significant harmonics, while the Wood Owl's call has about three. Figure 1.1 shows the spectrograms of a passerine (the Marico Sunbird) and a non-passerine (the Hadeda Ibis).

Doupe and Kuhl [5] wrote an in-depth article exploring common themes and mechanisms between bird song and human speech. Some insight into how human speech is learned can be gained by comparatively studying the processes in birds.

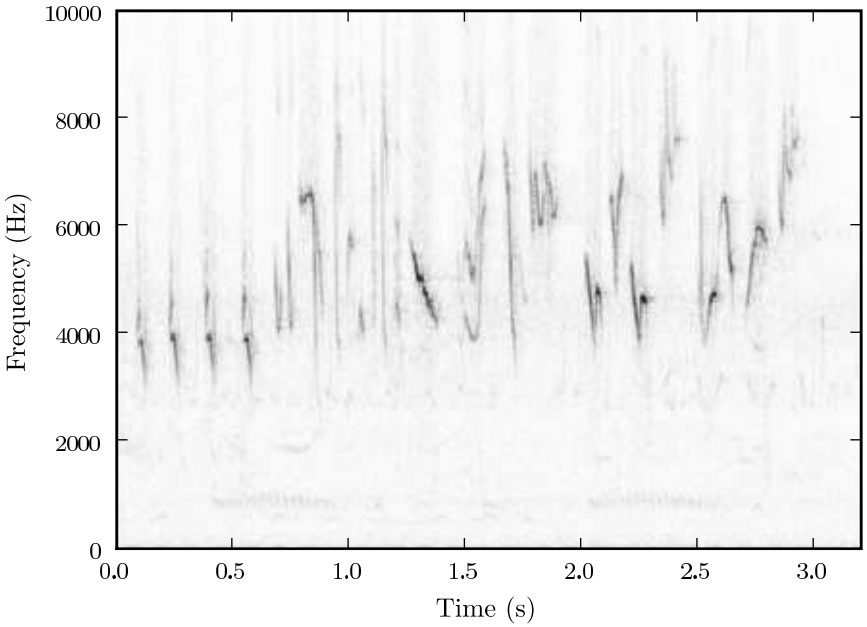
1.2 Bird Song Recognition Overview

The field of human speech recognition is considerably more advanced than bird song recognition. There are numerous speech recognition applications, namely

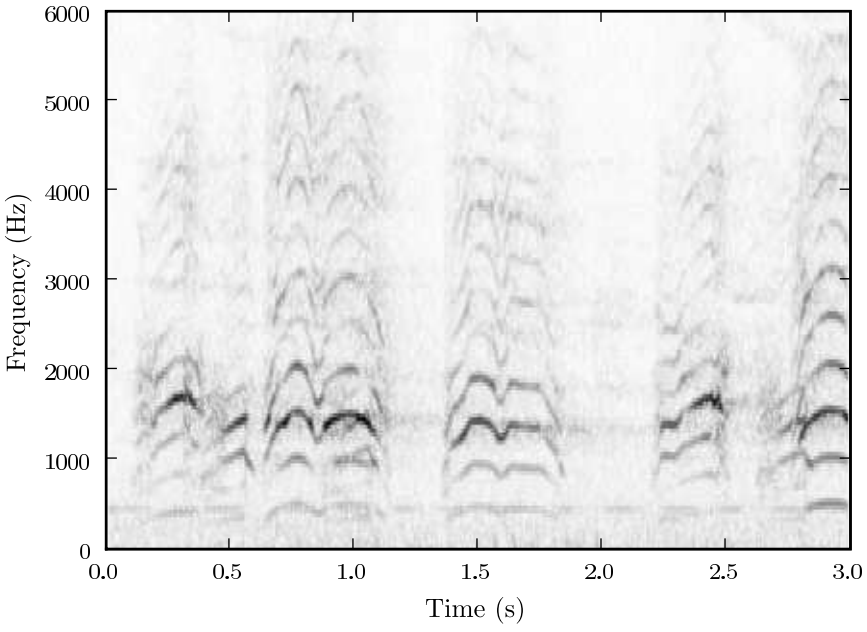
- automatic speech transcription, which is concerned with *what* a person is saying, and transcribes the speech to text,
- speaker recognition, which is concerned with *who* is talking in a recording, regardless of what was said,
- language identification, which is concerned with what *language* is being spoken, regardless of what was said or who was speaking.

Superficially, there are similarities between speech recognition and bird song recognition. There are also numerous applications for bird recognition algorithms.

As pointed out in the previous section, songbirds produce many vocalisations during vocal development. They over-produce vocalisations during the plastic song phase - in the case of song sparrows, sometimes producing as much as five times more song material than is finally used in the bird's adult repertoire [7]. Over the months that it takes for a songbird to develop mature song, the



(a) Marico Sunbird



(b) Hadeda Ibis

Figure 1.1: Spectrograms of a passerine (Marico Sunbird) and a non-passerine (Hadeda Ibis).

number of vocalisations produced makes thorough research prohibitively labour intensive, if some automated analysis tools are not available.

Song development research requires identifying the constituents in the song and transcribing the vocalisations of a specific individual. Models can thus be trained on the same individual that they will be used on. This is similar to speaker-dependent speech transcription.

Research into migration patterns or behaviour amongst adults requires identifying individual birds in the wild, and possibly counting them. An automated system that can recognise particular individuals within a species would allow data collection that is too monotonous or time consuming for a human to do. It could also be of great use for conservation research purposes. Consider parrots in dense tropical forests for example, that are usually heard and not seen. This task is similar to speaker recognition.

Birds are one of the most diverse terrestrial vertebrates with around 10,000 living and recently extinct species [33]. They exhibit diverse behaviour patterns, mating displays, shapes, sizes and colours, with diverse and fascinating songs that have inspired musicians for centuries. Combined with the magic of flight that has inspired engineers to push the limits, it comes with little surprise that bird-watching has developed into a massive sub-culture. The Dutch ethologist and ornithologist (and Nobel prizewinner) Nikolaas Tinbergen considered birdwatching to be an expression of the male hunting instinct [35]. Anthropologist John Liep compared birdwatching to the exchange of Kula valuables in Papua New Guinea — useless items assigned value by human culture, the trade of which serves to enhance social status and prestige [17, 36, 35]. Either way, there is great interest in amateur birdwatching. This provides a market for a commercial bird song identifier, to aid the amateur birdwatcher in identifying a song or call. Such a unit needs to identify a species, rather than an individual, and requires models capable of identifying the characteristics that are unique to a particular species. This task is somewhat similar to language or dialect identification, and is the focus of this study.

1.3 Contributions

As bird song recognition is a relatively new topic, a lot of the work done in this study was innovative. While previously existing methods were investigated, the significant differences between the systems make them hard to compare. Chapter 3 provides an overview of other approaches used, which include DTW and HMMs for transcribing vocalisations of a particular individual [1, 16] and recognition of syllables and syllable pairs based on pitch and volume [12, 13].

In our study, the following contributions were made:

- The ALS pitch detection algorithm [26] was adapted to bird song and demonstrated to be highly successful in most cases.
- The approximate-string-matching approach to tune indexing developed by McNab et al [19] was investigated for potential application to more intricate problems such as bird song recognition.
- Compounded Selfloops Duration Modelling (CSDM) was investigated as a valuable duration modelling technique when Gaussian durations are required. When used with long durations (a few dozen states or more) and moderate variances, CSDM can be much more computationally efficient than Ferguson stacks. CSDM cannot be used with the Viterbi algorithm though (which is commonly used to determine the most likely state sequence or transcription).
- The usefulness of feature analysis by synthesis was demonstrated by synthesising bird calls from the features under investigation.
- The importance of rhythm and duration modelling was demonstrated for signals with significant temporal characteristics, such as bird song. Much of the discriminating information is found in the rhythm of the signal. Some approaches transform these temporal characteristics to a dimension in the feature space [15]. This study demonstrated how the temporal modelling can be successfully performed by HMM topologies. For simple two-dimensional features, namely absolute pitch and volume, modelled by diagonal Gaussian pdfs, a recognition accuracy of about 51% was achieved on an independent evaluation set, for 47 bird calls from 39 birds, typically trained on only one or two individuals.

1.4 Organisation of this document

The next chapter describes the theoretical background used during the rest of this document. This includes a discussion of the analytical signal and the ALS pitch detection algorithm, modelling techniques such as DTW, approximate string matching and HMMs, finishing with a discussion of HMM duration modelling and an in-depth discussion of CSDM.

Chapter 3 provides an overview of most of the approaches to bird song recognition that have been developed recently. As this research field is relatively new, there are relatively few studies available.

The available bird call recordings are discussed in Chapter 4. Basic feature extraction and dataset selection is explained. It also discusses the selection of

birds, with reference to the effects of various features observed through synthesis of these features.

Chapter 5 provides a discussion of the modelling techniques tested in this study, while Chapter 6 provides the experiment details and results.

Concluding remarks and further recommendations are given in Chapter 7.

Chapter 2

Theoretical Background

This chapter describes pitch detection algorithms and pattern recognition techniques that will be used in later chapters.

2.1 The Analytical Signal

The Fourier transform $X(f)$ of a signal $x(t)$ is complex-valued in general. If the signal is real, the Fourier transform exhibits Hermitian symmetry, i.e. $X(-f) = X^*(f)$. This implies that the positive frequency components of the spectrum is sufficient to represent $x(t)$, and that the negative frequency components may be discarded without loss of information.

The analytical signal $x_a(t)$ is a construct which retains all the information in a signal $x(t)$, but discards the negative frequency components. It is defined as $x_a(t) = x(t) + j\hat{x}(t)$ where $\hat{x}(t)$ is known as the Hilbert transform of $x(t)$. The Hilbert transform is in turn obtained by passing $x(t)$ through a linear time-invariant filter with transfer function

$$H(f) = -j\text{sgn}(f) = \begin{cases} -j, & f > 0 \\ 0, & f = 0 \\ j, & f < 0. \end{cases} \tag{2.1}$$

The spectrum of the analytical signal therefore becomes

$$\begin{aligned} X_a(f) &= X(f) + j\hat{X}(f) \\ &= X(f) + jH(f)X(f) \\ &= (1 + \operatorname{sgn}(f))X(f) \\ &= \begin{cases} 2X(f), & f > 0 \\ X(0), & f = 0 \\ 0 & f < 0 \end{cases} \end{aligned} \tag{2.2}$$

which has no negative frequency components. The original signal can be easily reconstructed from the real part of $x_a(t)$, which proves that no information is lost.

The analytical signal $x_a(t)$ is a complex signal, which can be written in polar form, in terms of an amplitude and a phase, as $x_a(t) = A(t)e^{j\phi(t)}$. The two polar components of the signal are then the amplitude envelope $A(t) = |x_a(t)|$ and the instantaneous phase $\phi(t) = \arg(x_a(t))$. The derivative of the phase, $\omega(t) = \frac{d}{dt}\phi(t)$, is called the instantaneous angular frequency of the signal.

If $x(t)$ is an amplitude-modulated signal, $x(t) = x_m(t)\cos(2\pi f_0 t)$, with $x_m(t)$ a real and positive message signal with bandwidth less than f_0, the analytical envelope will be $A(t) = |x_m(t)|$. If $x_m(t) > 0$ for all t, the analytical envelope will simply be the message signal, $A(t)$=$x_m(t)$, and the instantaneous frequency will be constant, $\omega(t) = 2\pi f_0$. (When $x_m(t) < 0$, $\phi(t)$ undergoes a 180° phase shift, making $\omega(t)$ undefined at that point.)

The impulse response of the Hilbert transform is $h(t) = \frac{1}{\pi t}$. As is standard with ideal filters, this filter is non-causal, unstable and has an infinitely long impulse response. The usual trade-offs are made to shorten it, and a time delay is necessary in real-time applications to deal with the non-causality.

The frequency response of the ideal digital Hilbert transformer is specified as

$$H_d(\omega) = \begin{cases} -j, & 0 < \omega \leq \pi \\ j, & -\pi \leq \omega < 0. \end{cases}$$

The corresponding sampled impulse response in the time domain is

$$h(n) = \begin{cases} 0, & \text{for } n \text{ even} \\ \frac{2}{\pi n}, & \text{for } n \text{ odd.} \end{cases}$$

This would be used in a real-time application. In offline applications, the analytical signal is usually calculated via the frequency domain. The Fourier transform is calculated, the negative frequency components are zeroed and the

positive frequencies doubled as per Equation 2.2, and then the inverse Fourier transform is calculated to obtain the analytical signal.

2.2 Adaptive Least Squares Pitch Tracker

Saul et al describe the Adaptive Least Squares algorithm for pitch tracking of voiced speech in [26]. This algorithm splits a signal into numerous bands through the use of bandpass filters, then uses Prony's method to compute the optimal least-squares fit of a sinusoid to each band. Bands are designed such that the fundamental of a harmonic sound is in a band on its own, while for the higher harmonics that are closer together (geometrically speaking), at least two harmonics will fall into each band. A heuristic measure of the sharpness of the least-squares fit is used to select a band, for which the frequency of the optimally fitting sinusoid is taken as the pitch estimate. This is explained in more detail below.

2.2.1 Sinusoid Detection with Prony's method

The formula for a sampled sinusoid is:

$$s(n) = A\sin(\omega n + \theta).$$

Discrete sinusoids obey a simple difference equation: each sample is directly proportional to the mean of its two neighbours [26, Eq. (2)]. The factor of proportionality is $(\cos\omega)^{-1}$; i.e.

$$s(n) = (\cos\omega)^{-1}\left[\frac{s(n-1)+s(n+1)}{2}\right]. \tag{2.3}$$

This is easily shown using the trigonometric identity $\sin(A+B)+\sin(A-B) = 2\sin A\cos B$:

$$\begin{aligned}
(\cos\omega)^{-1}\left[\tfrac{s(n-1)+s(n+1)}{2}\right] &= (\cos\omega)^{-1}\left[\tfrac{A\sin(\omega(n-1)+\theta)+A\sin(\omega(n+1)+\theta)}{2}\right] \\
&= \tfrac{A}{2\cos\omega}\left[\sin\left((\omega n+\theta)-\omega\right)+\sin\left((\omega n+\theta)+\omega\right)\right] \\
&= \tfrac{A}{2\cos\omega}\left[2\sin(\omega n+\theta)\cos\omega\right] \\
&= A\sin(\omega n+\theta) = s(n).
\end{aligned}$$

Note that sampled exponentials also have each sample proportional to the mean

of its neighbours[1], but with a constant of proportionality less than one, whereas in the case of sinusoids, $\left|(\cos\omega)^{-1}\right|$ is always greater than or equal to one (being the inverse of $|\cos\omega| \leq 1$).

Consider a signal $x(n)$. If it is sinusoidal at frequency ω, it will satisfy Equation 2.3. We can thus calculate a sum of squared errors as ,

$$\mathcal{E}(\alpha) = \sum_n \left[x(n) - \alpha\left(\frac{x(n-1)+x(n+1)}{2}\right)\right]^2 \tag{2.4}$$

with $\alpha = (\cos\omega)^{-1}$, which is a measure of how closely $x(n)$ matches a sinusoid of frequency $\omega = \arccos(\alpha^{-1})$. As such, we can determine the frequency of a signal that is approximately sinusoidal, by finding $\alpha = \alpha^*$ that minimises $\mathcal{E}(\alpha)$. This is a least-squares problem, which can be solved by setting the derivative

$$\frac{\partial\mathcal{E}(\alpha)}{\partial\alpha} = 2\sum_n \left[x(n) - \alpha\left(\frac{x(n-1)+x(n+1)}{2}\right)\right]\left(-\frac{x(n-1)+x(n+1)}{2}\right)$$

equal to zero, resulting in the closed-form solution

$$\alpha^* = \frac{2\sum_n x(n)\left[x(n-1)+x(n+1)\right]}{\sum_n \left[x(n-1)+x(n+1)\right]^2}.$$

The sinusoidal frequency is estimated as

$$\omega^* = \arccos(1/\alpha^*) \tag{2.5}$$

on condition that $\alpha^* \geq 1$ as mentioned above, and $\mathcal{E}(\alpha^*) \ll \mathcal{E}(0) = \sum_n (x(n))^2$, to ensure that the mean squared error is small when compared to the mean squared amplitude of the signal.

This method for detecting sinusoids is known as Prony's method and has a number of useful properties. As the method only relies on the unlagged and one-sample-lagged autocorrelations, it can very efficiently return per-sample pitch. The pitch resolution is very good, as the pitch is determined from a least-squares fit rather than selecting the peak of a discrete Fourier transform or autocorrelation.

From the error (Equation 2.4), the ALS pitch tracking algorithm derives a heuristic measure of the quality of the sinusoidal frequency estimate. This

[1] Consider the sampled exponential $x(n) = e^{kn+\theta}$. Let $a = kn + \theta$, then

$$\begin{aligned}\tfrac{x(n-1)+x(n+1)}{2} &= \tfrac{e^{a-k}+e^{a+k}}{2} = \tfrac{e^{-k}+e^{k}}{2}e^{a} \\ &= (\cosh k)e^{a} = (\cosh k)x(n),\end{aligned}$$

thus

$$x(n) = (\cosh k)^{-1}\frac{x(n-1)+x(n+1)}{2}.$$

heuristic measures the sharpness of the least squares fit through use of a second derivative of the error function. It makes use of a normalised error function,

$$\mathcal{N}(\alpha) = \log\left[\frac{\mathcal{E}(\alpha)}{\mathcal{E}(0)}\right],$$

which evaluates the least squares fit on a dimensionless logarithmic scale that does not depend on the signal's amplitude. Furthermore, the second derivative is calculated with respect to $\mu = \log(\omega)$ so that the uncertainty is measured in geometric terms, similar to the musical scale (for example, an octave is two frequencies with a ratio of two, a "perfect fifth" is two frequencies with a ratio of $\frac{3}{2}$). The heuristic uncertainty $\Delta\mu^*$ of the best frequency estimate ω^* is thus calculated at $\mu^* = \log(\omega^*)$ as

$$\Delta\mu^* = \left[\left(\frac{\partial^2\mathcal{N}}{\partial\mu^2}\right)\Big|_{\mu=\mu^*}\right]^{-\frac{1}{2}} = \frac{1}{\omega^*}\left(\frac{\cos^2\omega^*}{\sin\omega^*}\right)\left[\left(\frac{1}{\mathcal{E}}\frac{\partial^2\mathcal{E}}{\partial\alpha^2}\right)\Big|_{\alpha=\alpha^*}\right]^{-\frac{1}{2}} \quad (2.6)$$

with

$$\frac{\partial^2\mathcal{E}}{\partial\alpha^2} = \sum_n \frac{\left(x(n-1)+x(n+1)\right)^2}{2}.$$

A lower uncertainty implies a larger second derivative and therefore a sharper fit.

2.2.2 Finding the pitch of voiced speech

The ALS algorithm detects voiced speech through a number of stages built around the algorithm in the previous section. It assumes the low-frequency spectrum of voiced speech is approximately harmonic, and therefore consists of a sum of sinusoids with frequencies that are integer multiples of a fundamental frequency.

2.2.2.1 Stage 1: Lowpass filtering

When ALS is used for speech, lowpass filtering is used to remove the aperiodic component of voiced fricatives above 1kHz. Following the lowpass filtering, the signal can be downsampled.

2.2.2.2 Stage 2: Pointwise non-linearity

Applying a pointwise non-linearity such as half-wave rectification (clipping negative values to zero) or squaring, concentrates additional energy at the fundamental frequency by crudely emphasising the envelope, which is especially im-

portant if the fundamental is weak or missing. Squaring is easier to understand than half-wave rectification — squaring the sum of two sinusoids concentrates energy at their difference frequency, amongst other things. Consider the second and third harmonics of a signal with a 100 Hz fundamental. The sum of a 200 Hz and 300 Hz sinusoid will have a 100 Hz beating or envelope, which is the fundamental frequency. Half-wave rectification is harder to characterise, but has a similar effect and maintains the dynamic range of the original signal. ALS uses half-wave rectification.

2.2.2.3 Stage 3: Filterbank

The ALS algorithm performs a least-squares fit of a sinusoid. In order to do this, it needs access to an individual sinusoid, rather than the (harmonic) sum of a number of sinusoids.

According to the harmonic assumption, the sinusoids present in the sound are represented by $f_n = nf_1$ where $n \in \mathbb{N}$ and f_1 is the fundamental frequency. The ratio of two successive harmonics f_k and f_{k+1} is thus:

$$\frac{f_{k+1}}{f_k} = \frac{(k+1)f_1}{kf_1} = \frac{k+1}{k}.$$

This ratio is equal to 2 for the first and second harmonics (f_1 and f_2, with $k = 1$), and less than 2 for subsequent harmonics.

We now want a bank of bandpass filters with parameters chosen such that one filter will pass the fundamental frequency and only the fundamental frequency, while any filters that pass a higher harmonic will pass more than one. This should result in one band having a highly sinusoidal output, while the higher bands have non-sinusoidal output due to multiple harmonics being passed.

For the minimum filter bandwidth, consider a bandpass filter passing f_2. In order to have a non-sinusoidal output, it must also pass either f_1 or f_3. This means the ratio of the upper cutoff to the lower cutoff of each bandpass filter must be at least $f_3/f_1 = 3$. If it is any smaller, it will be possible for it to have sinusoidal output by passing only f_2. There must also always be a filter that will pass f_1 but nothing higher, i.e. there must be a bandpass filter that has an upper cut-off that falls between f_1 and f_2. To achieve this, the ratio of one filter's upper cut-off to the previous filter's, should be $f_2/f_1 = 2$ or smaller.

The theoretical bandpass filters specified in [26] have an octave spacing with a width of two octaves, and are given as 25 Hz to 100 Hz, 50 Hz to 200 Hz, 100 Hz to 400 Hz, and 200 Hz to 800 Hz. However, their actual implementation had a larger number of narrower bands. Fei Sha's implementation [8] had bands with the upper frequency three times that of the lower frequency, and spaced at ratios of $\sqrt{2}$ — i.e. 25 Hz to 75 Hz, 36 Hz to 107 Hz, 50 Hz to 150 Hz, 71 Hz

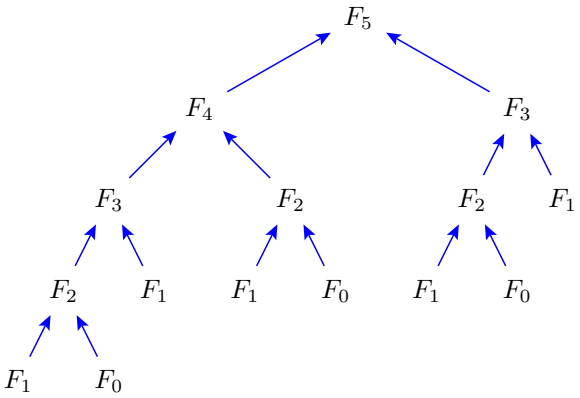


Figure 2.1: Naive approach to calculating F_5.

to 212 Hz, and so on, forming eight bands up to 800 Hz.

2.2.2.4 Stage 4: Sinusoid detection

A sample-by-sample frequency estimate of each output of the filter bank is made according to the algorithm in Section 2.2.1, estimating the frequency over a specific window size. This provides as many estimates as the number of filter banks, of which one must be selected. The criteria used for selection of the best filter bank is the uncertainty measure in Equation 2.6. The authors empirically determined that this is a better criteria than the error function in Equation 2.4. Furthermore, a voiced/unvoiced decision can be made by thresholding the uncertainty function and the energy in the signal.

2.3 Dynamic Programming

Dynamic Programming [3] is a technique for solving larger problems that exhibit the properties of overlapping subproblems and optimal substructure by recursively breaking it down into a number of sub-problems, and solving each sub-problem only once.

Consider for example the Fibonacci sequence, where $F_{n+1} = F_n + F_{n-1}$ with $F_1 = F_0 = 1$. A naive implementation for calculating F_5 using the mathematical formula directly is shown in Figure 2.1. Such a naive implementation results in F_2 being calculated three times and F_1 five times. As F_2 and F_1 never changes, this implementation is clearly suboptimal. In an optimal implementation, each subproblem should only be calculated once, as shown in Figure 2.2.

There are two approaches to avoid duplication of work — the top-down approach, and the bottom-up approach. The top-down approach is often the simplest to implement. It simply requires memoization of the function calls (caching

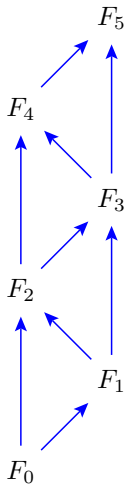


Figure 2.2: Using Dynamic Programming to calculate F_5.

calculated results and simply returning the cached result if it is requested again). In cases where all the sub-problems are not needed, the top-down approach can result in fewer function calls. However, the top-down approach is harder to parallelise and can have higher memory requirements. In the Fibonacci example, a top-down approach requires caching all smaller Fibonacci numbers.

The bottom-up approach calculates all the simplest problems first, then uses those results to calculate increasingly larger problems. This approach is easily parallelised or vectorised because a number of very similar problems are calculated at each step. Furthermore, when the results of the subproblems are no longer needed, their results can be discarded. In the Fibonacci example, we only need to keep the two most recent values to calculate the next.

2.4 Dynamic Time Warping

Dynamic Time Warping (DTW) is a technique for comparing two sequences where some time dilation or compression can occur. Such warping is modelled by insertion (or rather, duplication) and deletion of items in the sequence. DTW is an example of a Dynamic Programming algorithm.

2.4.1 Unconstrained DTW

Consider two sequences to be compared, $\mathbf{a}_{0..N} = a_0, a_1, \ldots, a_N$ and $\mathbf{b}_{0..M} = b_0, b_1, \ldots, b_M$. Each item a_i and b_j in the two sequences represent a feature vector of some sort (for example, spectral coefficients or cepstral coefficients). Let $\mathbf{a}$ be the template and $\mathbf{b}$ a sequence to be matched to it. DTW finds the

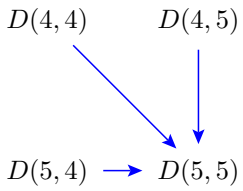


Figure 2.3: One unconstrained DTW step, the calculation of $D(5, 5)$.

best way of pairing elements of **a** and **b** while allowing dilation and compression, minimising some distance measure. Dilation implies that multiple elements of **b** map to a single element of **a**, leading to duplication of template elements in the matched sequence. Compression, on the other hand, maps multiple elements of **a** to a single element of **b**.

Let us define a local distance measure $d(i, j)$ which is the local distance between feature vectors a_i and b_j, and a global distance measure $D(i, j)$ which is the optimal total distance (the sum of local distances) between the optimal pairings of feature vectors in the sequences $\mathbf{a}_{0..i}$ and $\mathbf{b}_{0..j}$. We apply dynamic programming by expressing $D(i, j)$ in terms of $D(i-1, j)$, $D(i, j-1)$ and $D(i-1, j-1)$, as follows:

$$D(i,j) = d(i,j) + \min \begin{cases} D(i-1,j) & \text{(compression, multiple } \mathbf{a} \text{ for every } \mathbf{b}) \\ D(i-1,j-1) & \text{(direct pairing)} \\ D(i,j-1) & \text{(dilation, multiple } \mathbf{b} \text{ for every } \mathbf{a}). \end{cases} \tag{2.7}$$

The boundary conditions, for $i, j > 0$, are

$$\begin{aligned} D(0,0) &= d(0,0) \\ D(i,0) &= \sum_{k=0}^{i} d(k,0) = d(i,0) + D(i-1,0) \\ D(0,j) &= \sum_{k=0}^{j} d(0,k) = d(0,j) + D(0,j-1) \end{aligned}$$

Figure 2.3 represents the three cases of Equation 2.7 graphically.

The algorithm described by the equations above calculate all the global shortest distances up to $D(N, M)$, the shortest summed distance possible between $\mathbf{a}_{0..N}$ and $\mathbf{b}_{0..M}$. All these distances are typically stored in a matrix. This calculation requires three evaluations of the distance function for most items in the matrix, making this an $O(3NM)$ algorithm.

To obtain the actual pairings that provide this minimal score, we could either

save back-pointers in a second matrix, illustrating which of the three paths were used to obtain the score, or we can simply trace backwards through the $D(i, j)$ matrix, starting at $D(N, M)$, re-calculating the distances of the three possible sources to determine which it was. The maximum length of the trace is $N + M$, so the computational complexity of such back-tracking is $O(3(N + M))$. This is significantly less than the calculation of the D matrix in the first place, so the choice between an increase in memory requirements for saving the trace while calculating D and an increase in computational requirements for back-tracking without the back-pointers matrix, is arbitrary.

This solution to the time-warping problem allows unlimited compression and dilation. Since bird song tempo exhibits rather limited variation (typically less than human speech), unlimited compression and dilation may provide too much freedom during matching.

2.4.2 Constrained DTW

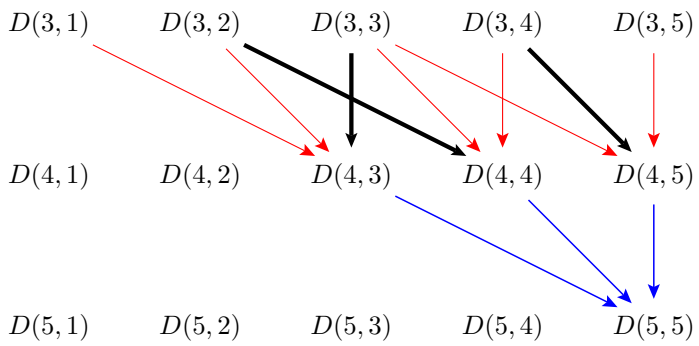
Anderson et al [1] restricts the DTW algorithm to have a maximum compression and dilation of a factor two. Their implementation also skips items in sequence **b** instead of matching multiple items in **b** to an item in **a**. Each item in the template **a** is therefore mapped to only a single item in **b**, while an item in **b** can be mapped to two items in **a**, and items in **b** can be skipped.

Let pairings between **a** and **b** be represented by $(w_i(l), w_j(l))$, where item $w_i(l)$ is the index of the item in **a** that is paired with the $w_j(l)$'th item in **b**, with l indexing the pairings. The recursive dynamic programming formula is

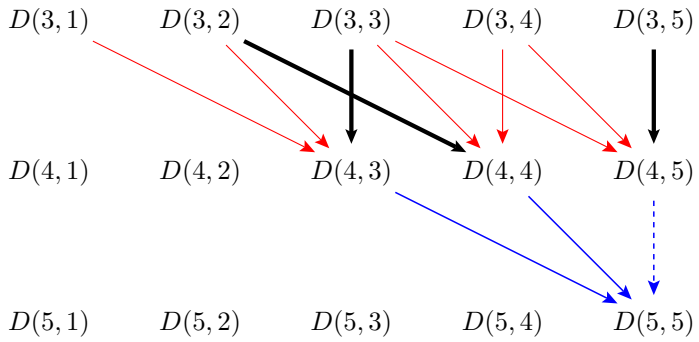
$$D(i,j) = d(i,j) + \min \begin{cases} D(i-1,j) & \text{iff } w_j(l-1) \neq w_j(l-2) \\ D(i-1,j-1) & \\ D(i-1,j-2) & \end{cases}$$

The first case is compression, with two items in **a** matched with one item in **b**. The constraint forbids more than two items in **a** matching an item in **b**, thereby limiting maximum compression to a factor of two. The second case is the normal direct match. The third case actually skips an item in **b**, giving a maximum dilation of factor two, where every item in **a** is matched with every second item in **b**. Figure 2.4 shows this constrained DTW algorithm graphically.

Such constraints to the time warping ensures that the rhythm of the pattern to be recognised plays a more significant role. In the case of unconstrained DTW, unlimited dilation or compression is permissible and not penalised, implying rhythm is irrelevant and only the ordering of the items matters.



(a) Compression constraint not needed.



(b) Compression being constrained.

Figure 2.4: One constrained DTW step with previous step shown. Legal precursors to $D(5,5)$ depends on the route taken to those precursors. The last step in the optimal path to each of the precursors is indicated with a bold arrow. In the second case, the dashed arrow indicates an impermissible route, as it would break the constraint on compression. A route to $D(5,5)$ has to pass through one of $D(3,1)$, $D(3,2)$, $D(3,3)$ or $D(3,4)$. There is no valid route to $D(5,5)$ via $D(3,5)$.

Note also that this constraint means the solution is not necessarily completely optimal, but close enough. In the case of Figure 2.4, the optimal route to $D(4,5)$ is via $D(3,5)$, forcing the route to $D(5,5)$ to go via $D(4,3)$ or $D(4,4)$. However, it is possible that $D(3,4) \rightarrow D(4,5) \rightarrow D(5,5)$ is a better solution. The constrained DTW is therefore not a true globally optimal DP algorithm. For it to be globally optimal, it would need to have a second route to each $D(n,m)$ in cases where the optimal route comes from $D(n-1,m)$, with the second route explicitly coming from either $D(n-1,m-1)$ or $D(n-1,m-2)$, to be used only when calculating the optimal route to $D(n+1,m)$. The sub-optimal constrained solution is good enough though, and the extra algorithmic complexity is not worth it.

2.5 The String-to-String correction problem

Another very similar dynamic programming application is the string-to-string correction problem presented by Wagner and Fischer [32]. This algorithm finds the minimum *edit distance* from one string to another, where an edit is an insertion, deletion or a substitution of a character in the string. While they use the language of *characters* in a *string*, we will continue to refer to *items* in a *sequence.*

An arbitrary cost function is defined, which assigns a cost to any particular edit operation. Deletions are similar to a DTW algorithm skipping items in sequence **a**, and insertions similar to skipping items in **b**. In this sense, it is similar to the implementation of dilation in Section 2.4.2. Unlike the DTW algorithm, however, these skip operations are also assigned a cost.

In each case, $D(i, j)$ can be determined if we already know $D(i-1, j)$, $D(i, j-1)$ and $D(i-1, j-1)$, which are the minimal edit distances between sequences with one less item. Routes to $D(i, j)$ arriving from each of these three cases correspond to the deletion of an item in the first sequence, insertion of an item into the second sequence, or substitution (or direct match) of an item, respectively.

As pointed out by McNab et al [19], this algorithm can be extended to implement fragmentation and consolidation of items. With this extension, two new string edits are included: the *fragmentation* of a single item in the first sequence into many items in the second, and the *consolidation* of a number of items in the first sequence, into a single item in the second. Defining these operations allows assigning a zero cost to a simple fragmentation operation if it has no other effect.

These two operations bring the string-matching approach even closer to the unconstrained DTW. In some ways, the unconstrained DTW algorithm effectively does only fragmentation and consolidation, and no deletions and insertions, except again, the DTW cost function is less flexible. When including fragmentation and consolidation, calculating $D(i, j)$ requires knowing not only the minimal distances of sequences one item shorter, but also of two, three, and more items shorter. For example, $D(0, j-1)$ is needed for the case where b_j is the consolidation of characters $a_0, a_1, a_2, \ldots, a_i$, for calculating the score $D(0, j-1) + d(0..i, j)$, where $d(u..v, w)$ represents the distance, or string-edit-cost, of consolidating sequence $a_u..a_v$ into b_j.

This was also implemented by Mongeau and Sankoff [20].

2.6 Hidden Markov Models

This section is a primer on Hidden Markov Models.

2.6.1 Markov Chains and Processes

Consider a sequence of random variables $X_1, X_2, X_3, ...$ which form a discrete-time stochastic process. Assuming the sequence is causal, the probability distribution of a future state X_{n+1} is $f_{X_{n+1}}(x_{n+1}) = P(X_{n+1} = x_{n+1}|X_n = x_n, ..., X_1 = x_1)$. The probability that the process produces a specific set of values $x_1, x_2, ...x_k$ is given by

$$P\left(\bigcap_{i=1}^{k} X_i = x_i\right) = \prod_{i=1}^{k} P\left(X_i = x_i | X_{i-1} = x_{i-1}, ..., X_1 = x_1\right). \tag{2.8}$$

A Markov chain (or Markov process) is a discrete-time stochastic process that has the Markov property, which for a discrete-time process means that the probability distribution of the next state is dependent only upon the current state, and not any past states. Mathematically, $P(X_{n+1} = x_{n+1}|X_n = x_n, ..., X_1 = x_1) = P(X_{n+1} = x_{n+1}|X_n = x_n)$. Let the set of all x_n be called S. If the Markov chain is time-homogeneous, then $P(X_{n+1} = j|X_n = i) = P(X_n = j|X_{n-1} = i) = a_{ij}$ for $i, j \in S$. $P(X_{n+1} = j|X_n = i)$ is a constant and not a function of n. Due to the Markov property, Equation 2.8 reduces to

$$\begin{aligned} P\left(\bigcap_{i=1}^{k} X_i = x_i\right) &= \prod_{i=1}^{k} P\left(X_i = x_i | X_{i-1} = x_{i-1}\right) \\ &= \prod_{i=1}^{k} a_{x_i x_{i-1}}. \end{aligned}$$

2.6.2 Markov Modelling

It is useful to illustrate Markov Modelling with an example. Suppose, for the purposes of this example, that the weather has one of two states on any given day: rainy or sunny. Also, assume that it is a time-homogeneous process, i.e. there are no seasons. Each day is a sample in a discrete-time stochastic process. If we have complete and infinite historical information on the weather, we can determine the conditional probability density function for tomorrow's weather, given today's weather and all days leading up to today.

This is too much history to be practical, so we make use of some simplifications. Assume either that the daily weather exhibits the Markov property, or that the assumption is an acceptable approximation. This means we assume

	Rainy tomorrow	Sunny tomorrow
Rainy today	0.7	0.3
Sunny today	0.2	0.8

Table 2.1: Probabilities for tomorrow's weather, given today's.

tomorrow's weather is only dependent upon today's weather, and not upon yesterday's.

With these simplifications, we find that the transition probabilities in our stochastic model for tomorrow's weather, based on today's, have only two parameters: the probability that, given it is rainy today, it will be rainy tomorrow, and the probability that, given that it is sunny today, it will be sunny tomorrow. These two probabilities and the probabilities that it will be sunny tomorrow if it is rainy today, or that it will be rainy tomorrow if it is sunny today, must add up to one, respectively, so the latter are not independent parameters. The probability that a system remains in the same state, for a given state, is referred to as that state's loopback probability.

From past observations, we can determine what these parameters are. See Table 2.1 for a contrived example. This example's parameters indicate that tomorrow's weather is more likely to be the same as today's weather, and that it typically remains sunny for longer times than it remains rainy.

Our model also needs to start at some point, so the initial conditions π_s and $\pi_r = 1 - \pi_s$ are the probability of it being sunny or rainy on the first day, respectively. The initial conditions can be chosen to be the same as the conditions for any other day[2]. This is achieved with $\pi_s = 0.6$ and $\pi_r = 0.4$. On average, 6 out of every 10 days are sunny.

The Markov chain can be represented as a graph with nodes representing states and uni-directional edges representing the transition probabilities. Figure 2.5 shows the graph for the Markov chain described by Table 2.1.

Let $P\,(\text{rainy tomorrow}|\text{sunny today})$ be represented by a_{sr} — the probability of transitioning from *sunny* to *rainy*. We can now calculate the probability

[2]It is either rainy or sunny,

$$P\,(\text{rainy}) + P\,(\text{sunny}) \quad = \quad 1.$$

From Table 2.1,

$$\begin{aligned} P\,(\text{tomorrow rainy}|\text{today rainy}) &= 0.7 \\ P\,(\text{tomorrow rainy}|\text{today sunny}) &= 0.2. \end{aligned}$$

Assuming the weather is time-homogeneous,

$$\begin{aligned} P\,(\text{tomorrow rainy}) &= P\,(\text{today rainy}) \\ &= P\,(\text{rainy}) \\ P\,(\text{tomorrow sunny}) &= P\,(\text{today sunny}) \\ &= P\,(\text{sunny})\,. \end{aligned}$$

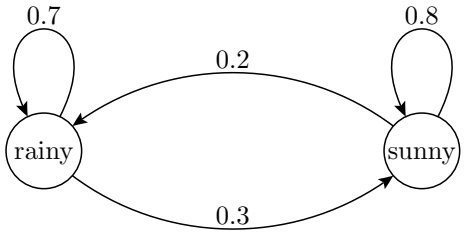


Figure 2.5: Graph of Markov chain in Table 2.1.

of this model generating any given weather sequence. The probability of any arbitrary five days being specifically "sunny, sunny, rainy, rainy, sunny", is the probability of the first day being sunny, multiplied by the probabilities of each of the four following transitions, or

$$P\left(\text{sunny}\right).a_{ss}.a_{sr}.a_{rr}.a_{rs} = 0.6 \times 0.8 \times 0.2 \times 0.7 \times 0.3 = 0.02016.$$

Typically the logarithms of the transition probabilities are used instead, turning the calculation into a sum rather than a product.

Note that this approximately 2% is not the likelihood that the *actual weather* will exhibit this pattern, it is the probability that the *model* generates this weather pattern. How close that likelihood is to the likelihood of the actual weather, depends on how good an approximation the model is for reality.

2.6.3 Hidden Markov Model

If we lack data for last month's weather, but we have data for another variable that is influenced by the weather, such as ice-cream sales, we can make an estimate about probable weather patterns. Suppose we know the transition probabilities in Table 2.1 based on observations made last year, and we also have ice-cream sale statistics with respect to the weather specified in Table 2.2. We can no longer directly observe the states of the Markov process, instead, the state sequence is *hidden* behind a layer of imperfect observations.

Now we can calculate

$$\begin{aligned}
P\left(\text{tomorrow rainy}\right) &= P\left(\text{tomorrow rainy}|\text{today rainy}\right)P\left(\text{today rainy}\right) \\
&\quad + P\left(\text{tomorrow rainy}|\text{today sunny}\right)P\left(\text{today sunny}\right) \\
\Rightarrow P\left(\text{rainy}\right) &= 0.7P\left(\text{rainy}\right) + 0.2P\left(\text{sunny}\right) \\
&= 0.7P\left(\text{rainy}\right) + 0.2\left[1 - P\left(\text{rainy}\right)\right] \\
&= \frac{0.2}{1 - 0.7 + 0.2} \\
&= 0.4 \\
\Rightarrow P\left(\text{sunny}\right) &= 0.6
\end{aligned}$$

	1 ice-cream	2 ice-creams	3 ice-creams
Rainy	0.7	0.2	0.1
Sunny	0.1	0.3	0.6

Table 2.2: Ice-cream sale probabilities depending on weather.

Let o be an observation. It can be a discrete symbol, or it can be an observation vector. To model a hidden Markov process, we associate a probability density function $b_t(o)$ with each state t of the Markov chain, which models the observations. For example, when it is sunny, the probability that three ice-creams were sold is 60%. When it is rainy, the probability of selling three ice-creams is only 10%. Therefore if three ice-creams were sold, we don't know for sure whether it was sunny or rainy, but we know it was *more likely* to be sunny.

Let $O = (o_1, o_2, o_3, o_4, o_5)$ be a sequence of observations, namely the number of ice-creams sold, and $W = (w_1, w_2, w_3, w_4, w_5)$ be the state sequence, namely the weather on each consecutive day. Consider now the weather sequence used in the previous section, "sunny, sunny, rainy, rainy, sunny", represented by $W_a = (s, s, r, r, s)$. The probability of selling three ice-creams every day, given this sequence, is

$$\begin{aligned} P\left(O = (3,3,3,3,3) | W = W_a\right) &= 0.6^3.0.1^2 \\ &= 0.00216. \end{aligned}$$

The most likely sales, given this weather sequence, is $O_\alpha = 3, 3, 1, 1, 3$ with a probability of

$$\begin{aligned} P\left(O = O_\alpha | W = W_a\right) &= 0.6^3.0.7^2 \\ &= 0.10584. \end{aligned}$$

If we don't *know* the weather sequence, however, we have the joint probability of both the observations and the state sequence,

$$\begin{aligned} P\left(O = O_\alpha \cap W = W_a\right) &= P\left(O = O_\alpha | W = W_a\right) P\left(W = W_a\right) \\ &= 0.10584 \times 0.02016 \\ &= 0.0021337 \end{aligned}$$

(to five significant figures). Note that this is the probability for a specific state sequence. The same output could be generated with a state sequence of $W_b =$

(r, s, r, s, s), with the probability

$$\begin{aligned} P\left(O = O_\alpha \cap W = W_b\right) &= P\left(O = O_\alpha | W = W_b\right) P\left(W = W_b\right) \\ &= P\left(O = (3,3,1,1,3) | W = (r,s,r,s,s)\right) P(r) a_{rs} a_{sr} a_{rs} a_{ss} \\ &= (0.1 \times 0.6 \times 0.7 \times 0.1 \times 0.6) \times (0.4 \times 0.3 \times 0.2 \times 0.3 \times 0.8) \\ &= 0.00252 \times 0.00576 \\ &= 1.45152 \times 10^{-5}. \end{aligned}$$

In order to calculate $P\left(O = O_\alpha\right)$, we need to take *all* possible routes into consideration.

$$P\left(O = O_\alpha\right) = \sum_{W_i} P\left(O = O_\alpha \cap W = W_i\right)$$

For a fully-connected Markov chain, where all state transitions are allowed, an N-state system with k observations or steps has N^k possible paths. Calculating each and every possible path's likelihood independently is prohibitively expensive.

An HMM is thus typically represented by three sets of parameters, the initial probabilities π_i, the transition probabilities a_{ij} and the state pdfs $b_i(o)$ which is a function of observation vectors and contains more parameters. There are three operations we would like to perform with HMMs. We need to train the HMM parameters, score a given set of observations to determine how likely an HMM is to generate that particular set of observations, and label a set of observations to determine what the most likely state sequence was if the observations were generated by the HMM in question.

2.6.4 The Forward and Backward algorithms

In order to efficiently determine $P\left(O = O_\alpha\right)$ using dynamic programming, it must be recursively broken down into smaller subproblems. The likelihood of a given set of observations with a specified final state, can be expressed in terms of a sum of the probabilities of each way of getting there, which consists of a subproblem one observation smaller, a transition probability, and the probability of the emitting pdf producing the final observation. Let $\alpha(t+1, q) = P(O_{1..t+1} \cap w_{t+1} = q)$. Due to the output independence assumption,

$$\begin{aligned}\alpha(t+1,q) &= \sum_{p=1}^{N} P\left(O_{1..t} \cap w_t = p\right) a_{pq} b_q\left(o_{t+1}\right) \\ &= \left[\sum_{p=1}^{N} \alpha(t,p) a_{pq}\right] b_q\left(o_{t+1}\right).\end{aligned}$$

The dynamic programming algorithm that makes use of this property in order to calculate $P\left(O = O_\alpha\right)$ is known as the Forward algorithm [24]. The same principle can be implemented in reverse, with $\beta(t-1,p) = P\left(O_{t..T} \cap w_{t-1} = p\right)$ giving

$$\begin{aligned}\beta(t-1,p) &= \sum_{q=1}^{N} a_{pq} b_q\left(o_t\right) P\left(O_{t+1..T} \cap w_t = q\right) \\ &= \sum_{q=1}^{N} a_{pq} b_q\left(o_t\right) \beta\left(t,q\right).\end{aligned}$$

This is known as the Backward algorithm.

2.6.5 The Viterbi Algorithm

The Viterbi algorithm [24] is very similar to the Forward algorithm. It determines not the total probability of a given sequence of observations, but rather the probability of the most likely state sequence for those observations. It does so by finding the maximum probability at each step, instead of the sum of probabilities used in the Forward algorithm. The Viterbi algorithm can build up a matrix of back pointers, a matrix showing which previous state was selected for each observation in order to maximise the score. This matrix allows backtracking to calculate the Viterbi path, the most likely state sequence for the given observations. Because there is no mix of sum and product in the Viterbi algorithm, an implementation can work purely in the log domain.

2.6.6 Training an HMM with Baum-Welch

Finding the transition probabilities in Table 2.1 would be trivial if the state sequences in the training data can be directly observed. In the case of an HMM however, the state sequence is not available.

HMMs can be trained with the Baum-Welch or Forward-Backward algorithm [24], which is an example of a general class of optimisation methods known as Expectation-Maximisation algorithms [21]. The model needs to be initialised after which it can be iteratively improved to a local optimum. How close this is

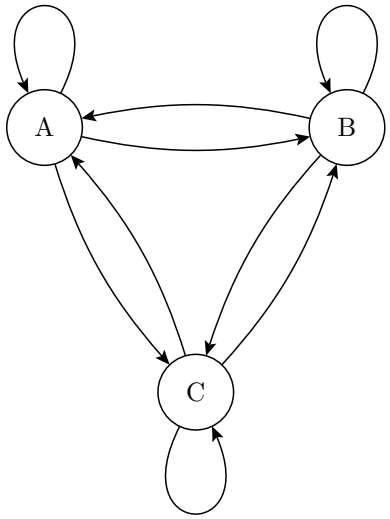


Figure 2.6: A three-state ergodic HMM.

to a global optimum depends on the quality of the initialisation.

On each iteration, the likelihoods of each state occurring at each time t is calculated by using the Forward and Backward algorithms, to find

$$P\left(w_t = p | O = O_\alpha\right) = \frac{P\left(w_t = p \cap O = O_\alpha\right)}{P\left(O = O_\alpha\right)}$$

as well as $P\left(w_t = p \cap w_{t+1} = q | O = O_\alpha\right)$. This is the Expectation step. These likelihoods are then used to re-estimate the transition probabilities a_{ij} as well as the observation pdfs $b_i(o)$ to obtain an improved model. This is the Maximisation step. With enough training data, the Expectation and Maximisation steps can be repeated alternately until the model converges. With insufficient data, specialisation can be a problem, where the model fits the training data increasingly well but no longer testing data. In such cases, it is useful to restrict the number of training iterations. This is not seen during training unless a development set is used to score on each iteration.

2.6.7 HMM Configurations

Fully-connected HMMs are HMMs where every state is connected to every other state, including itself. The weather pattern in Figure 2.5 is an example of a two-state fully-connected HMM. Figure 2.6 shows the structure of a three-state ergodic HMM.

Figure 2.7 shows a *left-to-right* HMM topology. The states can be numbered in such a way that states always transition from lower-numbered states to higher-

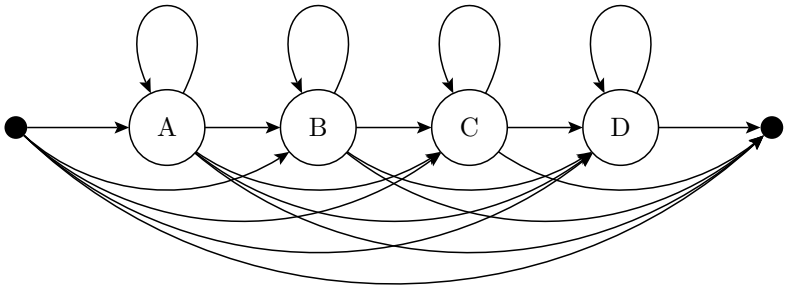


Figure 2.7: A Complete Left-to-Right HMM.

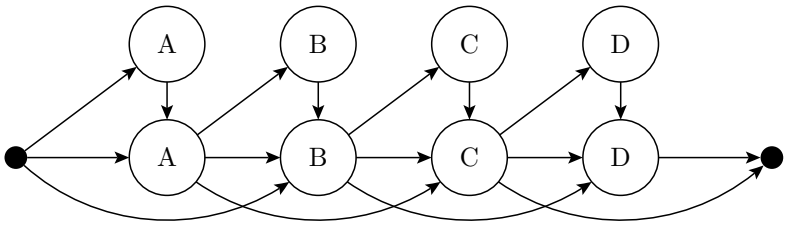


Figure 2.8: An HMM simulating constrained DTW. Vertical stacks of states correspond to the same item in the template, and have the same output pdf.

numbered states. There is a distinct left-to-right flow through the HMM. Models like these are useful for describing signals with time-varying statistics. The small black dots represent *null states* that do not generate an observation vector. Such states have no emitting pdf and are useful for connecting HMMs into larger structures. This particular model has a complete set of *skip links*, meaning any number of states can be skipped. In a sense it has a lot in common with the unconstrained DTW, as it allows infinite compression or dilation.

An HMM can also be constructed that simulates the constrained DTW. Figure 2.8 shows such an HMM. It has eight states, but only four emitting pdfs. Each pair shares a pdf. There are no selfloops (loopback probabilities are zero). With this structure, the HMM can generate at most two observations of each symbol. With the model simulating the DTW template, this implements a "dilation" of factor two. The only skip links in the model skip one state column at a time, thereby limiting the maximum "compression" . Like typical DTW implementations, this model has one pdf for each expected observation, assuming that the compression/dilation ratios range from half to double.

Figure 2.9 shows how we could have half as many pdfs in the model for a particular duration, with the expected duration of each pdf being twice that of Figure 2.8, but compression/dilation allowing for a range of one to four (ranging from half the expected duration to double the expected duration, as before).

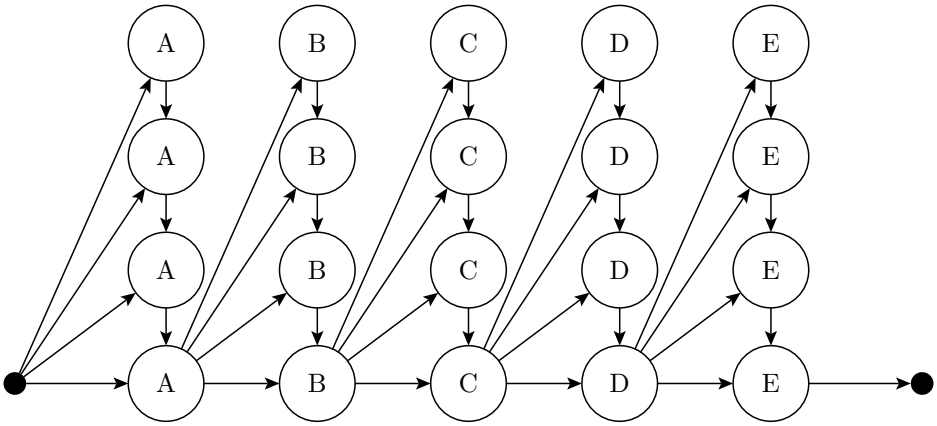


Figure 2.9: Adjusted constrained DTW simulation with Ferguson model.

2.6.8 HMM versus DTW

Section 2.6.7 showed how the more flexible structure of HMMs can implement the same functionality as DTW. Whereas a DTW algorithm uses a template and some distance measure, an HMM uses pdfs that are trained to represent the distribution of the emitted observation vectors. Where various distance measures can be used in DTW, various pdfs can be used in an HMM. Gaussian Mixture Models are popular for their flexibility.

Normal DTW algorithms do not have the equivalent of HMM transition weights, as all permissible transitions effectively have an equal weight. The trainability of the transition weights in an HMM further increases its modelling potential. While constraints can be used in DTW to put limits on compression or dilation, there is still no discrimination with regards to how much compression and dilation occurs within these limits. In the next section, we investigate the duration modelling capabilities that HMM transition weights provide.

This increased flexibility does come at the price of much larger training data requirements.

2.7 Duration Modelling with HMMs

2.7.1 Selfloops and the Left-to-Right model

The simplest and most common method of modelling state durations in HMMs and Markov chains, is via states' loopback probabilities. Consider the simple single-state Markov chain (not counting initial and final null states) in Figure 2.10 , with loopback probability α. Let the length (duration) of the state sequence generated by the model be the random variable L. The probability

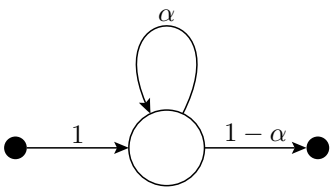


Figure 2.10: Single-state Markov chain with selfloop.

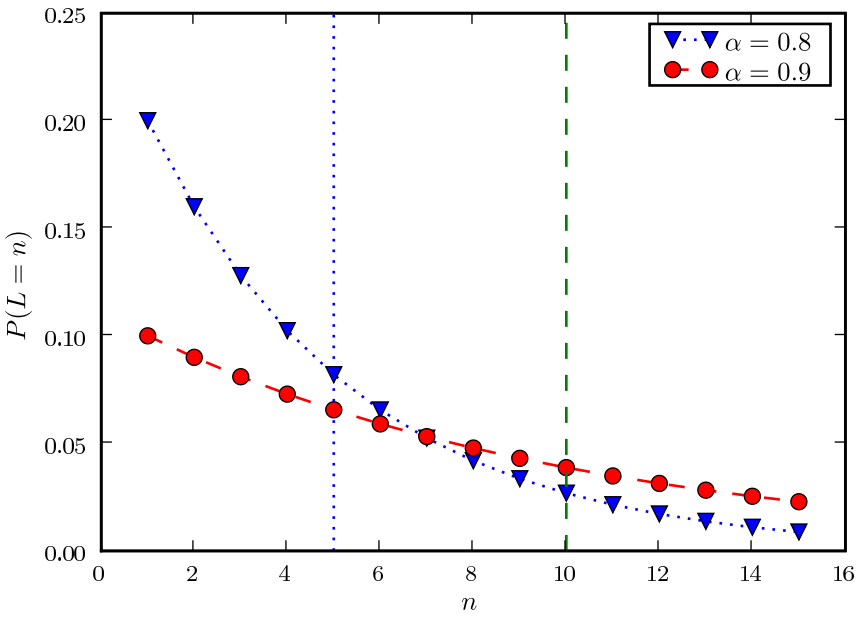


Figure 2.11: Duration distribution of state with selfloop α.

that the length is k steps is

$$P(L = k) = (1 - \alpha)\alpha^{k-1} \tag{2.9}$$

with $k \geq 1$. This is a discrete geometric distribution. The standard form of the geometric pdf is $(1 - p)^{k-1} p$, found by setting $\alpha = 1 - p$. Figure 2.11 shows this distribution for $\alpha = 0.8$ and $\alpha = 0.9$. The expected value of L, the mean state duration, is

$$\mathrm{E}(L) = \overline{L} \quad = \quad \frac{1}{1 - \alpha},$$

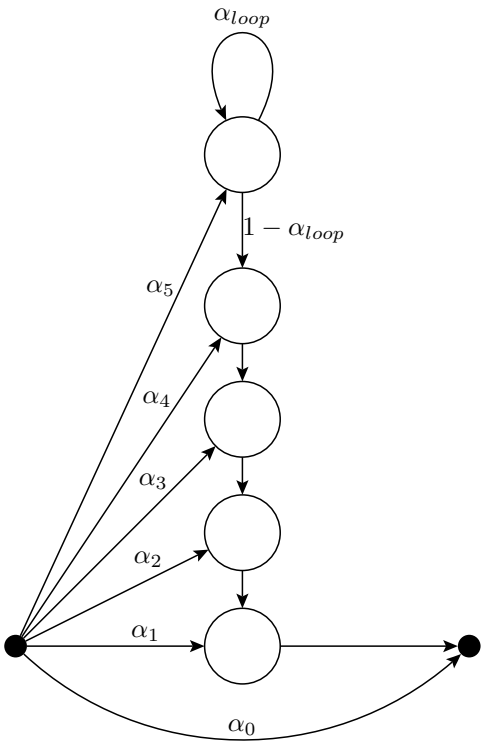


Figure 2.12: A five-state Ferguson stack.

while the geometric distribution variance, $\frac{1-p}{p^2}$, becomes

$$\sigma_L^2 = \frac{\alpha}{(1-\alpha)^2}. \tag{2.10}$$

Solving for α, we can design the selfloop for a specific expected duration as

$$\alpha = \frac{\overline{L} - 1}{\overline{L}}. \tag{2.11}$$

As there is only one parameter, α, the mean and variance cannot be independently specified.

2.7.2 Ferguson Stacks

A well-known method for implementing duration modelling, allowing arbitrary duration pdfs, is the Ferguson stack [9].

An example of a Ferguson stack with five states is shown in Figure 2.12. It has a loop probability of α_{loop} and a skip probability of α_0. All the states share the same output pdf. Any number of states can be used in a Ferguson stack, depending on the duration distribution that needs to be modelled. Note

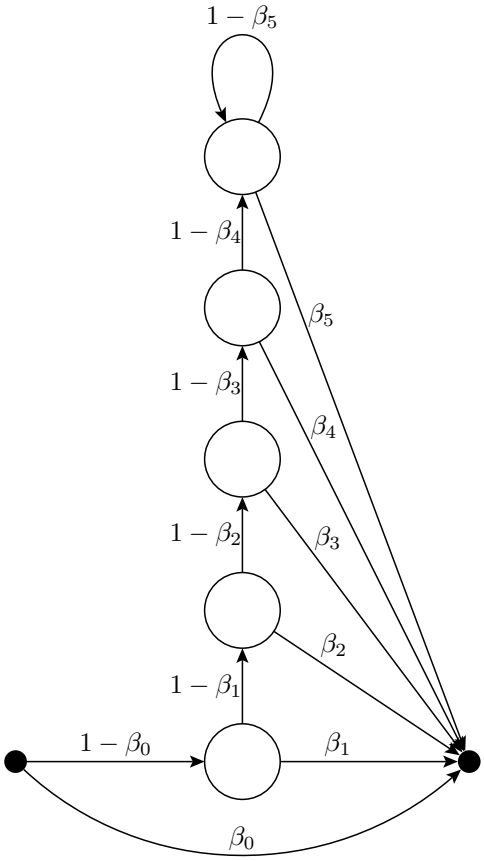


Figure 2.13: Alternative form of a Ferguson stack.

that the HMM configurations shown in Figures 2.8 and 2.9 are in fact simply Ferguson stacks with no selfloops.

Using a stack like this instead of a single state with selfloop allows specifying a relatively arbitrary duration pdf instead of the simple exponential of Section 2.7.1. The probability that the duration will be k when there are n states in the stack, is given by

$$P(L=k) = \begin{cases} \alpha_k, & 0 \leq k < n \\ \alpha_n \left(1-\alpha_{\text{loop}}\right), & k = n \\ \alpha_n \left(1-\alpha_{\text{loop}}\right) \alpha_{\text{loop}}^{k-n}, & k > n. \end{cases} \tag{2.12}$$

Figure 2.13 shows an alternative but functionally equivalent configuration of a Ferguson stack. The β parameters can be calculated from the α parameters in order to achieve functionally the same result.

The weights α_k can be derived from another parametrised distribution, for

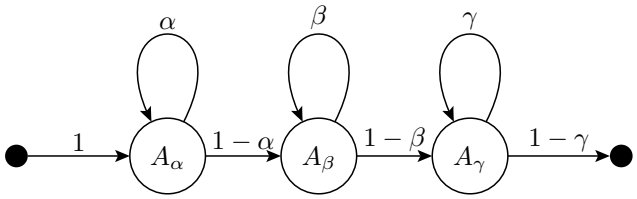


Figure 2.14: Three state left-to-right Markov Chain with no skip links.

example a normal distribution if the duration distribution is required to be Gaussian. This reduces the modelling of the stack's duration to two parameters. The parameter reduction is important if the stack contains dozens of states, as having too many trainable parameters requires prohibitively large quantities of training data.

2.7.3 Compounded Selfloops Duration Modelling

Another way to do duration modelling is to concatenate HMM states with the same (tied) output pdf and selfloop probabilities into a left-to-right model. This will be referred to as a *compounded selfloops HMM*.

2.7.3.1 Properties of Compounded Selflooped states

A left-to-right model with three states and no skip links is shown in Figure 2.14. Let the duration of the three states be represented by the random variables L_α, L_β and L_γ, with variances σ_α^2, σ_β^2 and σ_γ^2. The total duration for this model is $L = L_\alpha + L_\beta + L_\gamma$. Because each duration distribution is independant, the expected duration of the whole model is simply the sum of the durations of each state: $\overline{L} = \overline{L}_\alpha + \overline{L}_\beta + \overline{L}_\gamma$, with variance $\sigma^2 = \sigma_\alpha^2 + \sigma_\beta^2 + \sigma_\gamma^2$.

Suppose the three loopback probabilities are equal, i.e. $\alpha = \beta = \gamma = \frac{\overline{L}/3-1}{\overline{L}/3}$. The duration distribution of the model is then the convolution of three identical discrete geometric distributions. Figure 2.15 compares the duration distribution between this three-state model (Figure 2.14) and a one-state model (Figure 2.10), both with a total expected duration of $\overline{L} = 10$. For the three-state model, each selfloop must contribute a third of the duration, so $\alpha = \beta = \gamma = \frac{10/3-1}{10/3} = 0.7$, while for the one-state model, $\alpha = \frac{10-1}{10} = 0.9$.

From the Central Limit Theorem we know the sum of n independent random variables with identical distributions tends towards a Gaussian with increased n. Figure 2.16 shows the duration distributions for three left-to-right models, each with a total expected duration of $\overline{L} = 50$. These distributions were calculated using convolution. The twenty-state model has a considerably more symmetrical duration distribution than the ten- and five-state models, and is approximately

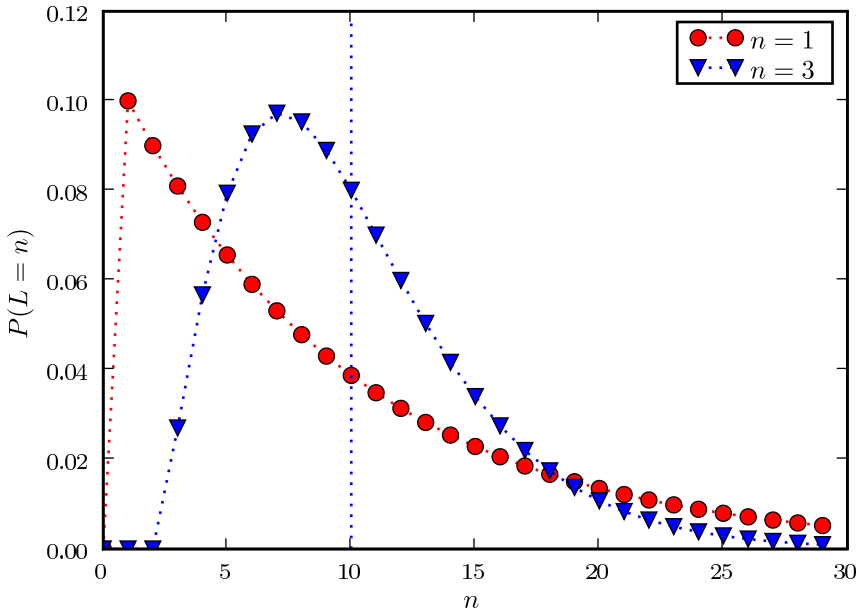


Figure 2.15: Comparison of one- and three-state models with equal $\overline{L} = 10$.

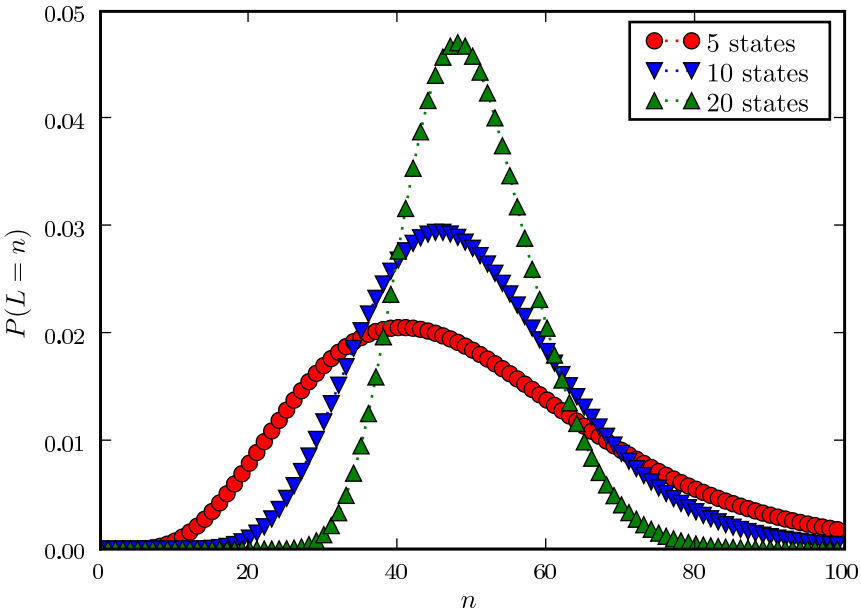


Figure 2.16: CSDM duration distributions for varying number of states. Given a fixed mean, $\overline{L} = 50$, increasing the number of states decreases the variance.

Gaussian in shape. At the same time, the variance decreases with increased n (for a fixed mean $\overline{L}$).

The variance of the duration distribution of the sum of n independent but equal distributions with variance σ_α^2 is $n\sigma_\alpha^2$. From Equation 2.10 we know the variance of the duration of a single self-looped state is $\sigma_\alpha^2 = \frac{\alpha}{(1-\alpha)^2}$, and from Equation 2.11 we know that $\alpha = \frac{\overline{L}_\alpha - 1}{\overline{L}_\alpha}$. We can thus determine the variance in terms of the expected duration, as

$$\begin{aligned} \sigma_\alpha^2 &= \frac{\overline{L}_\alpha - 1}{\overline{L}_\alpha \left(1 - \frac{\overline{L}_\alpha - 1}{\overline{L}_\alpha}\right)^2} \\ &= \left(\overline{L}_\alpha\right)^2 - \overline{L}_\alpha. \end{aligned}$$

When using n states for a total duration of $\overline{L} = n\overline{L}_\alpha$ and $\sigma^2 = n\sigma_\alpha^2$, we see

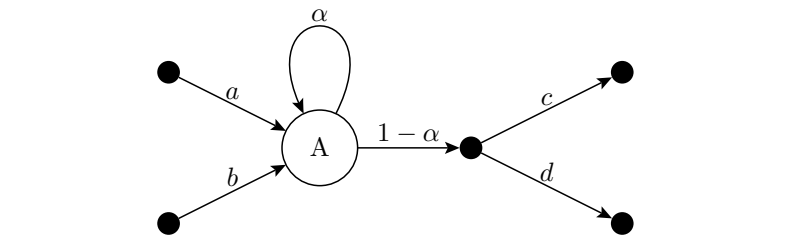
$$\begin{aligned} \sigma^2 &= n\left[\left(\frac{\overline{L}}{n}\right)^2 - \frac{\overline{L}}{n}\right] \\ &= \frac{\overline{L}^2}{n} - \overline{L}. \end{aligned} \tag{2.13}$$

Using Equation 2.13, we can choose n to achieve a particular variance given the expected duration $\overline{L}$. With n and $\overline{L}$ specified, we can then determine what the selfloop probabilities of each of the states must be:

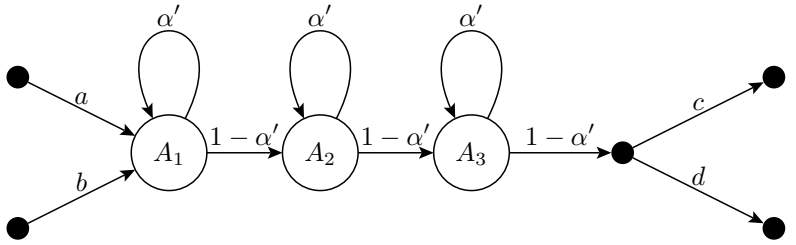
$$\begin{aligned} \alpha &= \frac{\overline{L}_\alpha - 1}{\overline{L}_\alpha} \\ &= \frac{\overline{L}/n - 1}{\overline{L}/n} \\ &= \frac{\overline{L} - n}{\overline{L}}. \end{aligned} \tag{2.14}$$

2.7.3.2 Transforming a simple selfloop into compounded selflooped states

We can also deduce an HMM transformation that can replace a particular self-loop with probability α with n compounded selflooped substates, each with the same loopback probability of α'. This transformation allows independently reducing the variance of the duration of that state pdf. The number of substates needed for a given variance can again be determined by Equation 2.13 (with $\overline{L} = \frac{1}{1-\alpha}$). We do not want to modify the expected duration, so



(a) Original simple selfloop



(b) Three-state compounded selfloops

Figure 2.17: Transformation of selfloop into three compounded selflooped states.

$$\begin{aligned} \overline{L} = \frac{1}{1-\alpha} &= n\left(\frac{1}{1-\alpha'}\right) \\ \Rightarrow 1-\alpha' &= n(1-\alpha) \\ \Rightarrow \alpha' &= 1-n(1-\alpha) \\ &= 1-n+n\alpha. \end{aligned} \tag{2.15}$$

Figure 2.17 shows the transformation of a single selfloop into a three-state compounded selflooped structure. A null state is used to avoid having to adjust the c and d weights.

2.7.3.3 Comparing CSDM and Ferguson stacks

In cases where a state's expected duration $\overline{L}$ needs to be a couple of dozen or more, we can use compounded selfloops duration modelling to achieve Gaussian-like duration distributions with considerably fewer states than a Ferguson stack would need. A Ferguson stack needs more states than the expected duration if it is to approximate a Gaussian distribution. For durations longer than the number of states in the stack, the duration has a discrete geometric distribution (Equation 2.12).

This CSDM method is not useful for modelling short durations though, as the minimum duration is the number of states in the compound. Consider the variance of compounded selfloops with a minimum duration of half the mean

value. In this case, $n = \frac{\overline{L}}{2}$. Equation 2.13 becomes $\sigma^2 = \overline{L}$.

The standard deviation σ for an $n = \frac{\overline{L}}{2}$ CSDM HMM is the square-root of its expected length. For $\overline{L} = 100$, that puts the minimum duration $n = 50$ at five standard deviations ($\sigma = 10$) from the mean. Forcing any duration beyond five standard deviations to zero is often an acceptable approximation. With $\overline{L} = 36$, the minimum duration of eighteen would be at three standard deviations, while for $\overline{L} = 16$ the minimum duration of eight is at two standard deviations. This indicates why CSDM without skip-links is inadequate for modelling states with a normally-distributed duration and a low mean. Achieving a low variance without any skip-links requires too high a minimum duration. This problem is solved by the Extended CSDM structure in the next section.

It is important to realise that CSDM relies on multiple paths through the model. The single most likely path through the model is the shortest path, which visits each state exactly once, which is true for any left-to-right model without skip-links. The only reason a longer duration is more likely, is because there are significantly more paths of that duration, all of which contribute to the total probability of that duration.

Consider sub-figure (b) in Figure 2.17. The most likely sequence of substates is $A_1A_2A_3$ and has length three. A sequence with duration four requires another transition, which reduces the likelihood of that particular sequence being generated, for example $A_1A_2A_2A_3$. However, as all three states have the same output pdf, a particular sequence of four observations could have been generated by $A_1A_1A_2A_3$, $A_1A_2A_2A_3$ or $A_1A_2A_3A_3$. Due to the identical nature of the substates, all three of these state sequences are equally likely, and the likelihood of the model generating a set of four observations is three times the likelihood of any one of these paths.

In the case of five observation vectors, there are six potential sequences: $A_1A_1A_1A_2A_3$, $A_1A_1A_2A_2A_3$, $A_1A_1A_2A_3A_3$, $A_1A_2A_2A_2A_3$, $A_1A_2A_2A_3A_3$ and $A_1A_2A_3A_3A_3$. The formula for the number of potential state sequences for an n-state compound and l observation vectors is $\begin{pmatrix} l-1 \\ n-1 \end{pmatrix}$, as $n-1$ transitions have to occur to get from the first substate to the last, and these transitions have to occur at one of the $l-1$ transitions between the l observations. In the case of a 20-state compound, the number of state sequences by which the model could generate 50 observations, is $\begin{pmatrix} 49 \\ 19 \end{pmatrix}$ which is nearly 2×10^{13}. The likelihood of $L = 50$ is thus nearly 2×10^{13} times the likelihood of any *particular* state sequence.

This kind of duration modelling technique has implications for the Viterbi algorithm. The Viterbi normal algorithm is not aware of the equivalent status

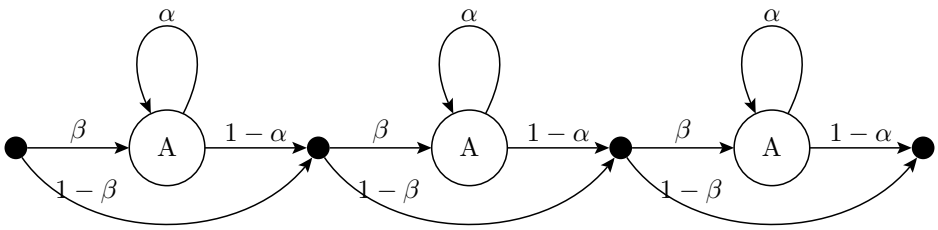


Figure 2.18: Compounded Selflooped states with skip-links.

of substates, so it is not taken into account during the calculation of the Viterbi path. If accurate transcriptions are required or a Viterbi algorithm is needed for another reason, CSDM is not an option and Ferguson stacks would be better. Sometimes Viterbi is used because of its speed advantage: all calculations can be made in the log domain, as it calculates the maximum rather than the sum (Section 2.6.5). The Forward algorithm operates both in the linear and the log domain, which makes it slower than Viterbi algorithm which can be calculated purely in the log domain. In cases where the Viterbi algorithm can be used, Viterbi and Ferguson may be faster than the Forward algorithm and CSDM, depending on the required duration (which impacts the size of the Ferguson stack) and the required variance (which impacts the number of states in the CSDM compound). If only the Forward algorithm is used, CSDM is faster due to fewer states.

The computational complexity of the Viterbi and Forward algorithms is $O((B+K)NT)$ with N the number of unique states in the HMM, K the average number of predecessor states for each, T the number of observations and B the number of operations to compute an observation likelihood [14]. For certain duration distributions, CSDM structures have fewer states than similar Ferguson stacks, and can therefore be faster if both are scored with the Forward algorithm.

2.7.3.4 Extended CSDM

The problems with normal CSDM can be solved by making use of skip-links. Figure 2.18 shows a compounded set of selflooped states with skip-links. It introduces another parameter, β. By adding a skip-link, the model gains the ability to describe a sequence of zero duration. Furthermore, by setting the skip-link probability to $1-\beta = 1-\alpha$, each individual component in the compound still has a discrete geometric distribution. This maintains the properties described in Section 2.7.3.1. Effectively the new duration random variable is $L' = L - 1$. The distribution of Equation 2.16 then becomes zero-based:

$$P(L' = k) = (1 - \alpha)\alpha^k \tag{2.16}$$

for k any non-negative integer. Note, however, that setting $\beta = \alpha$ is not making maximum use of that topology's flexibility. Some smaller value of β will result in a more Gaussian-like duration distribution for each individual substate in the compound, resulting in a more Gaussian-like compounded distribution.

Figure 2.18 is equivalent to a complete left-to-right HMM similar to Figure 2.7 when the null states are removed and the transition probabilities are adjusted accordingly. However, asymptotically, a complete left-to-right HMM with n emitting states has $O(n^2)$ transitions, while the model using null states has $2n+1$ states as every emitting state is followed by a null state, and $4n$ valid transitions, as there are two transitions for each of $2n$ states. The complete left-to-right equivalent of Figure 2.18 is thus more complex. If the model is going to be as restricted as Figure 2.18 requires, Figure 2.18 is more computationally efficient. With the increased freedom of the complete left-to-right model, however, there should be a more optimal set of transition weights that provide a more Gaussian-like shape than the restricted version.

2.7.4 Other techniques

Johnson [14] provides an overview of other HMM duration modelling techniques, including hidden semi-Markov models and variable-transition HMMs, with reference to their capacity and complexity. He suggests that simple adjustments to HMM topologies may be more efficient and desirable than more complex approaches, which require substantial algorithmic development and more computing power.

Another approach to duration modelling is higher-order HMMs, where the future state and conditional pdf depends not only on the current state, but also on the previous state (second order), the previous two states (third order), or more. Such models can be implemented using a first-order equivalent HMM through specific transforms that increase the number of states. Higher-order HMMs are useful when state durations are short. When state durations are typically long, the previous state or two has little relevance, and very high order modelling is required.

Chapter 3

Literature Study

Bird song recognition research is still a relatively new topic, with comparatively little published literature. This chapter provides an overview of some of the methods that have been studied.

3.1 LCSR

Anderson et al [1] used dynamic time warping to analyse and label constituents in continuous recordings of indigo buntings and zebra finches. The system they developed is named Long Continuous Song Recognition, or LCSR. LCSR is based on the one-stage (parallel) DTW algorithm introduced by Vintsyuk [31] and developed by Bridle et al [4], because it is computationally efficient.

A few templates were identified by a human for each syllable type (for example, five), as well as for silence. For noise and hard to classify vocalisations, more templates were used. For example, 42% of the zebra finch songs were too diverse to divide into subclasses, so thirteen exemplar templates were used to represent the diversity more adequately.

From these templates, the log magnitude of the FFT is computed from 500 Hz to 10 kHz. Ignoring frequencies below 500 Hz removes low frequency noise which occurred in the laboratory recording environment, for example fan noise. Comparisons were made using a Euclidean distance measure, giving equal weight to all frequency bins. Time warping is constrained to a maximum factor of two, and is the only mechanism to compensate for the variability in sounds.

DTW is well suited for recognition of vocalisations consisting of highly regular units that may occur in an unknown order. In the case of animal or bird vocalisations, the units are typically well defined, much more so than speech. In identifying syllables in stereotyped songs, it achieved a 97% accuracy, while on the lower amplitude, more variable plastic songs of the indigo bunting, it

achieved an 84% accuracy.

For optimal performance, the selection of templates is important. Because this is done by a human, there is still an element of subjectivity in the system. The performance depends on the degree of variability within classes, and the separability between classes represented by the templates. There is a subtle trade-off between the effectiveness of templates in representing class variability and maintaining separability between them.

3.2 HMMs

In order to improve on these problems, Kogan and Margoliash compared the LCSR system with an HMM-based approach, built with HTK [16]. The comparison was again done on zebra finches and indigo buntings, with songs recorded in the laboratory under a variety of noise conditions. While the DTW approach used by LCSR performed well for ideal recordings, under more challenging noisy conditions the template selection becomes harder. HMMs use statistical representation and estimation of the vocalisations to be recognised. Through training, more information is accumulated by the models, thereby generalising better than template-based techniques. The HTK approach outperformed LCSR in noisy conditions.

Each sound category was modelled by an HMM, where the typical model was a five-state left-to-right model, with three emitting states. (The first and last states are non-emitting, and are used to tie the models together.) Each state had a mixture Gaussian with a diagonal covariance matrix. Experiments used between two and four mixtures. Various parametrisations supported by HTK were investigated and tested with various HTK classifiers.

While the statistical approach of HMMs requires considerably more training examples, only segmentation and labelling of the constituent vocalisations are necessary. In comparison, LCSR requires fewer templates, but needs much greater expert knowledge in template selection, in order to obtain a representative set. For laboratory use Kogan and Margoliash suggest using LCSR for preparing a training set which is then corrected manually. The resulting segmentation can then be used to train the HTK-based recogniser.

A weakness of the HMM approach is misclassification of short-duration vocalisations or song units with more variable structure, such as some calls and syllables of plastic songs.

3.3 Other Approaches

Härmä [12] investigates recognition of birds on individual syllables by sinusoidal modelling. A Short-Time Fourier Transform is used to compute a spectrogram of a song. In one example, a Kaiser window of size 256 was used, and zero-padded to 1024 samples before taking the FFT. Pitch and volume trajectories are picked out of the spectrogram using peak-picking, with syllable boundaries determined according to volume: typically everything within 30 dB of a syllable's maximum volume is considered part of that syllable. Typically, some post-processing is required to remove clearly erroneous syllables. The frequency and amplitude trajectories of templates of each bird to be recognised, are compared with the testing samples through use of a weighted sum of mean differences between the trajectories. To make comparison easier, syllables are centred on maximum amplitude.

This method was used for passerine birds only. Passerine birds usually have a pure sinusoidal timbre. The advantage of this approach is the simplicity of finding single syllables despite multiple birds singing simultaneously, where song overlap can make it extremely difficult to determine which syllables belong to which bird. The advantage of this approach above the LCSR and HTK based systems, is that it is relatively insensitive to environmental noise and colouration, which is often found in field recordings. This system was also developed for use as a simple baseline as well as to investigate what more needs to be done in more complicated systems.

Further work by Härmä and Somervuo made use of syllable pair histograms [30]. Recognition was performed between four species, with access to recordings from 50 individuals in total. Another study classified bird syllables into four classes according to their harmonic structure [13].

In a fourth year project, Joubert [15] investigated various features for bird song recognition, for use with Gaussian Mixture Models and Hidden Markov Models. Twelve birds were investigated at a time.

3.4 Tune Recognition

Some birds are recognised by the tunes they sing, and can be transcribed using normal musical notation. As such, while it does not deal with bird song directly, research into tune retrieval from a database of human folk tunes may be of interest.

McNab et al [18, 19] used approximate string matching to retrieve tunes from a database given acoustic input, for example a human singing. Approximate string matching is another example of a dynamic programming algorithm as

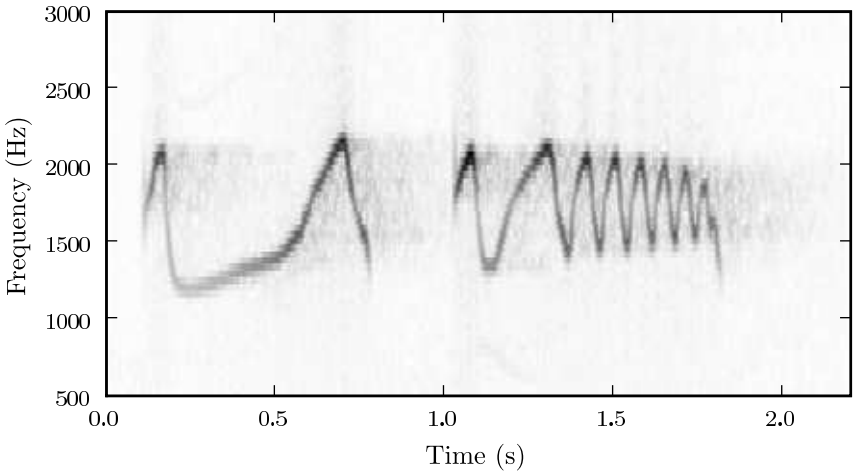


Figure 3.1: Fierynecked Nightjar spectrogram.

is DTW. Query tunes are compared to template tunes, with the comparison returning an edit distance. Edits supported typically include note insertions, note deletions and note substitutions.

Pitch contours can be represented by various mechanisms, the simplest simply being whether the next note is higher (U), lower (D) or repeated (R). Using this notation, *Three Blind Mice* becomes *DDUDDUDRDUDRD. For more discriminatory power, a sequence of intervals can be used. The cost or distance of a string edit operation can be a complex function to penalise dissonant-interval errors more than harmonic substitutions, in order to compensate for typical human singing errors. Rhythm is scored separately in a similar fashion, based on note and silence lengths intead of intervals. The rhythm and interval edit distances are then linearly combined to determine a final score.

As this system is meant for human queries consciously sung, a number of requirements can be made. If notes are sung using "la", "da" or "ta", note segmentation by amplitude is easy: the consonant causes a drop in amplitude at note boundaries. Another alternative is note segmentation directly from the pitch track. Pitch-based segmentation relaxes constraints on the user, but is not suitable for all applications. A slide, or *glissando*, is segmented into a sequence of ascending or descending notes. Human folk tunes are much simpler to compare in this way, as tunes more often consist of simple notes. While some songbirds have tunes that can be similarly transcribed, many birds, especially non-passerines, make extensive use of slides or glissando's. Consider for example the Fierynecked Nightjar, shown in Figure 3.1. Nevertheless, this study briefly explored the string-matching idea, as it might be useful for birds that *can* be transcribed using music notation or tokenised by some other method.

Chapter 4

Exploring Bird Song for Dataset Selection

4.1 Dataset Considerations

4.1.1 Data Scarcity and Domain Knowledge

One of the big challenges during this study was dealing with data scarcity. The two databases that were used were designed for training humans to recognise bird song. For any pattern recognition task, humans can draw on a huge wealth of knowledge and experience, allowing us to learn to recognise a species' song by listening to as little as one example. From experience, humans can often instinctively tell which aspects of the song are important for recognition, and which are not. The databases used had an average of only 30 seconds of audio per bird.

It is a tall order to expect a computer to be able to learn to recognise bird song on as little data as a human, considering the computer lacks the prior knowledge and experience a human draws on. In the process of developing algorithms for pattern recognition, we often draw on our own experience and techniques. For example, speech recognition systems are informed by knowledge of how both the human vocal system and the human auditory system works, following the principle that if humans can recognise speech, there must be something that can be learned from how they manage to do it.

Of course, while humans are not necessarily the ultimate recognisers, and therefore not necessarily the best example to model our artificial recognition systems on, humans evolved their recognition skills over millions of years and are still much better at many recognition tasks than even the best algorithms, especially after they have had sufficient training. It is worthwhile to try to learn

from the solutions evolution provided.

There are two contrasting approaches to pattern recognition systems. One approach is characterised by throwing huge amounts of data at a rather flexible model, with the training process statistically discovering the important discriminating features that humans may be able to identify instinctively. The other approach is to build more human knowledge into a system, resulting in less flexible but more informed models, with fewer parameters, reducing the amount of data necessary to effectively train the models.

Most pattern recognition systems balance these two aspects in various ways. Neural networks may be considered to fall into the first extreme, but even then, human knowledge of the best neural network topologies is often applied. Note also how neural networks were inspired by the functioning of our biological neurons. Manually constructed decision trees or expert systems would be an example of the other extreme, where training data is not required. (Decision trees can also be trained, though.) The significant scarcity of data in this study necessitated the constructing of models informed by as much human knowledge as possible.

4.1.2 Training, Development and Evaluation datasets

We typically desire to build and train models that are able to recognise patterns that have not been seen before. In the case of a bird-song recogniser that identifies species, the aim would be to recognise new birds in the field that have not been heard before. Emulating this in research typically involves splitting data into a training set and an evaluation set. Models trained on the training set are then evaluated on the evaluation set to see how well they perform. Ideally no knowledge of the evaluation data should be allowed to leak into the models.

Typically, models are developed iteratively, with the evaluation results of each iteration used to drive improvements to the models. Every time the results of an evaluation with the evaluation set is used for informing a design or development decision, some knowledge of the evaluation set has effectively snuck into the models, placing the ideal of mimicking the real-world application as closely as possible into jeopardy. By endlessly developing new models based on evaluation results, we would be doing nothing other than higher-level training on our evaluation set.

A common technique to avoid this problem is to use a third independent data set. This *development set* is used for iterative development, with only a few final evaluations and comparisons done on the evaluation set. Using this approach, the development iterations cannot glean information from the evaluation dataset. Again, however, this requires more data. There was insufficient

data for a development set to be possible in this study, so the dangers of certain types of iterative improvements had to be kept in mind. A good way of mitigating that danger is *understanding* what each iteration does, dealing mostly in broader categorical results and limiting the number of iterations.

4.1.3 Gibbon and Gillard databases

The data used in this study was Guy Gibbon's *Southern African Bird Sounds* [10] and Len Gillard's *The Beginner's Guide to Bird Calls of Southern Africa* [11], henceforth referred to as "Gibbon" and "Gillard", respectively. Gibbon's recordings include 888 species on six tapes, in an attempt to be as complete as possible. (At the time of publication in 1991, there were 915 birds on the list of South African bird species.) The recordings were made in the field, not in a laboratory, so they include background noise, overlapping bird song, and various environmental and channel effects. They were digitised by ED Schwardt [28] at a sampling rate of 22.05 kHz with 16 bits per sample (mono). The tapes are accompanied by a booklet describing the recordings and providing some background information. Gillard includes 99 species on a CD (99 being the maximum number of tracks an audio CD can have).

Both databases have on average about 30 seconds of audio per bird. Each of these tracks can consist of more than one vocalisation and attempt to be representative of the most common vocalisations. For the purposes of this study, there were usually between 3 and 12 usable examples of each of the chosen vocalisations in each database. In rare cases there were only two. Gibbon, for instance, has only two examples of the Wood Owl's shorter "wow" call while Gillard has only three. The more intricate characteristic call of the Wood Owl is better represented. Another example is the Greenspotted Dove, which has a call that is longer than 10 seconds, so both Gibbon and Gillard have only two examples of it.

While 30 seconds may be enough for training humans, as discussed in Section 4.1.1, we really would want significantly more data for training flexible statistical models. The booklet accompanying Gibbon suggests scientifically adequate representation of some species could take 5 to 10 minutes of audio. Training statistical models should ideally be given even more. For best results, multiple recordings from many birds in many different settings should be used. In contrast to this ideal, while Gibbon and Gillard may have multiple examples of the same vocalisation, typically all the examples of a specific vocalisation were recorded in the same session, mostly from either a single specimen or else from two birds duetting.

As a minimum requirement, birds chosen must have recordings available

from multiple independent sources in order to be able to select independent training and evaluation datasets. This requirement immediately reduces the viable species to the 99 birds found in Gillard, as all these species are also contained in the Gibbon set. This allows one set to be used as the training set and the other as the evaluation set. With only two independent specimens of most species available, there is not enough data for a development dataset, so great care is required to avoid leaking knowledge of evaluation data into the models.

Another potential pitfall is human knowledge of the testing set. With a lack of training data, more human knowledge and experience must be used to develop better-informed models. The theory is that human knowledge effectively provides more independent training data. However, as soon as the human listens to and informs his or her own knowledge of the subject matter by listening to the evaluation data, the independence of the models and the evaluation data is in danger.

This study lacked both adequate training data and sufficient human expertise with bird song recognition. This has a significant impact on which birds could be selected. By focusing on simpler birds, techniques and solutions are developed that can hopefully be extended to more intricate birds in future work, with more training data and human expertise being available.

4.2 Bird Song Analysis by Synthesis

4.2.1 Synthesis Overview and Rationale

In human speech recognition, most of the information is found in the peaks of the spectral envelope, known as formants. For non-tonal languages, the excitation frequency in the glottis, or pitch, carries very little meaning, and can often be disregarded. Cepstral coefficients are therefore very successful features for speech recognition, as it is able to separate the filtering effect of the vocal tract from the excitation from the glottis.

This contrasts significantly with a majority of bird species' song, where most of the energy is concentrated at a single frequency. Many species, especially passerines (songbirds), have hardly any harmonics in their song. This is because birds typically coordinate their vocal tract with the excitation frequency of the syrinx [25]. Most of birds' song discriminating characteristics are thus found in the tune, consisting of the pitch and the rhythm (how the pitch changes over time). A number of songbirds' songs can be quite accurately transcribed using standard musical notation, indicating that the tune's information lies purely in pitch and rhythm.

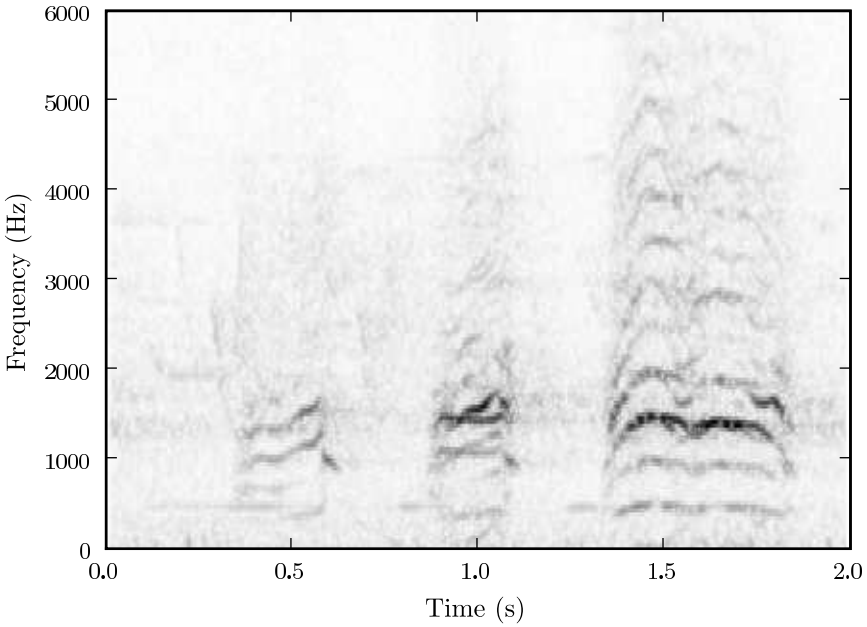


Figure 4.1: A spectrogram of a pair of Hadeda Ibis singing together.

The songs of non-passerine birds typically contain more harmonics. Some have a significant number of harmonics, and are highly characterisable by their *timbre*, for example the Hadeda Ibis, shown in Figure 4.1. Notice the large number of harmonics, as well as the interesting characteristic that the strongest harmonic is not the fundamental but rather the third. Even in most of the non-passerine cases, modulation of the vocal tract still has no significant role in distinguishing the bird from others. Even if a few birds can be recognised via cepstral coefficients and vocal tract modelling, the main discriminative feature for bird song in general still seems to be the tune. To someone that knows bird sounds, a whistled Hadeda does not sound particularly recognisable due to the lack of harmonics. However, when sung with a human voice, it can be recognised easily. While the presence or absence of harmonics plays a role, the more important detail remains the tune.

Because of the properties just described, this study dealt primarily with very simple features: pitch and volume tracks. To evaluate the information found in these tracks, the songs were synthesised from these pitch and volume tracks and compared to the original by listening to them. While the listening tests were not done in a particularly scientific fashion, a number of synthesised calls are practically indistinguishable from the original, while others are highly recognisable even when they are missing harmonics. If a human can still easily recognise the

real bird song from synthesised tracks, they clearly still have enough information for identification purposes.

Spectral techniques are particularly prone to channel effects and background noise. An advantage of a simple system based on pitch and volume is that it is less affected by channel effects and background noise.

4.2.2 Pitch Tracking

A number of pitch tracking algorithms were implemented and investigated for suitability. Accurate quantitative comparisons could not be made, as there are no available transcriptions of the correct pitch. The pitch tracking algorithms were tested using subjective listening tests, which is adequate for selecting an acceptable algorithm, as well as determining whether there is enough information in those features. The algorithms investigated included

- cross-correlation of adjacent windows where the window size is adjusted to achieve a maximum cross-correlation, inspired by Bagshaw's super resolution pitch tracker [2],
- peak picking the periodogram (using no window or the Hamming-window),
- dynamic programming to find a smooth pitch track through the periodogram [29],
- ALS, the Adaptive Least Squares algorithm described in Section 2.2 [26], with modifications for use with bird song.

The ALS algorithm was selected as it performed very well in most cases and has a few additional advantages. As the ALS algorithm can return per-sample pitch, it was easy to experiment with the effects of various decimation rates by calculating and saving per-sample pitch once.

A different choice could also work, although it would have different advantages and disadvantages. The Harmonic Product Spectrum method [27] might for instance be useful for birds where the strongest harmonic is not the fundamental, like the Hadeda Ibis.

4.2.2.1 Adapting ALS to bird song

As ALS (Section 2.2) was originally developed for use with human voices, it requires some adjustments for use with birdsong. Birds have a much larger pitch range than humans. Some passerine birds have fundamental frequencies as high as 8 kHz. Because the data was already lowpass filtered at 11 kHz and sampled at 22 kHz, no further filtering and downsampling was done for

the ALS pitch track. Furthermore, the filter bank was extended from eight to fifteen filters to achieve the necessary range.

Another adjustment was to not make a voiced/unvoiced decision based on energy or uncertainty estimates (the ALS cost function of Equation 2.6). Instead, the ALS cost was always returned together with the best pitch estimate, with any voiced/unvoiced decision left to other code. This enables the use of the cost function in various ways, including as another feature.

See Figure 4.2 for an example of a pitch track overlaid on a spectrogram. Notice that the ALS cost function indicates sinusoidality when the call has greater energy, with a lower cost indicating more sinusoidality.

4.2.3 Frequency and Amplitude

4.2.3.1 The Analytical Signal and ALS

As explained in Section 2.1, the analytical signal contains all the information of a real signal. Furthermore, the analytical signal can be expressed in terms of an instantaneous frequency and an envelope. The instantaneous analytical frequency and amplitude envelope together therefore contains all the information of the original signal. By synthesising the bird song using these two as pitch and volume tracks, the original audio is perfectly reconstructed.

In order to reduce the computational requirements, the features should be simplified. The first simplification replaces the instantaneous frequency with the smoother output of the ALS pitch tracking algorithm. Figure 4.3 compares the ALS pitch track with the instantaneous frequency from the analytical signal. The instantaneous frequency is very unstable during silent periods, but during actual song it correlates well with the output from the ALS algorithm. Figure 4.4 compares ALS and downsampled instantaneous frequency, this time lowpass filtered before the downsampling.

4.2.3.2 Amplitude

To characterise a tune, we also need the volume, or amplitude, of the signal. Two approaches were investigated. The first was the amplitude envelope of the analytical signal (calculated through use of the offline method, see Section 2.1), while the second was the more traditional lowpass filtering of the absolute value of the signal. Song amplitude was normalised by dividing by the maximum according to the call transcriptions.

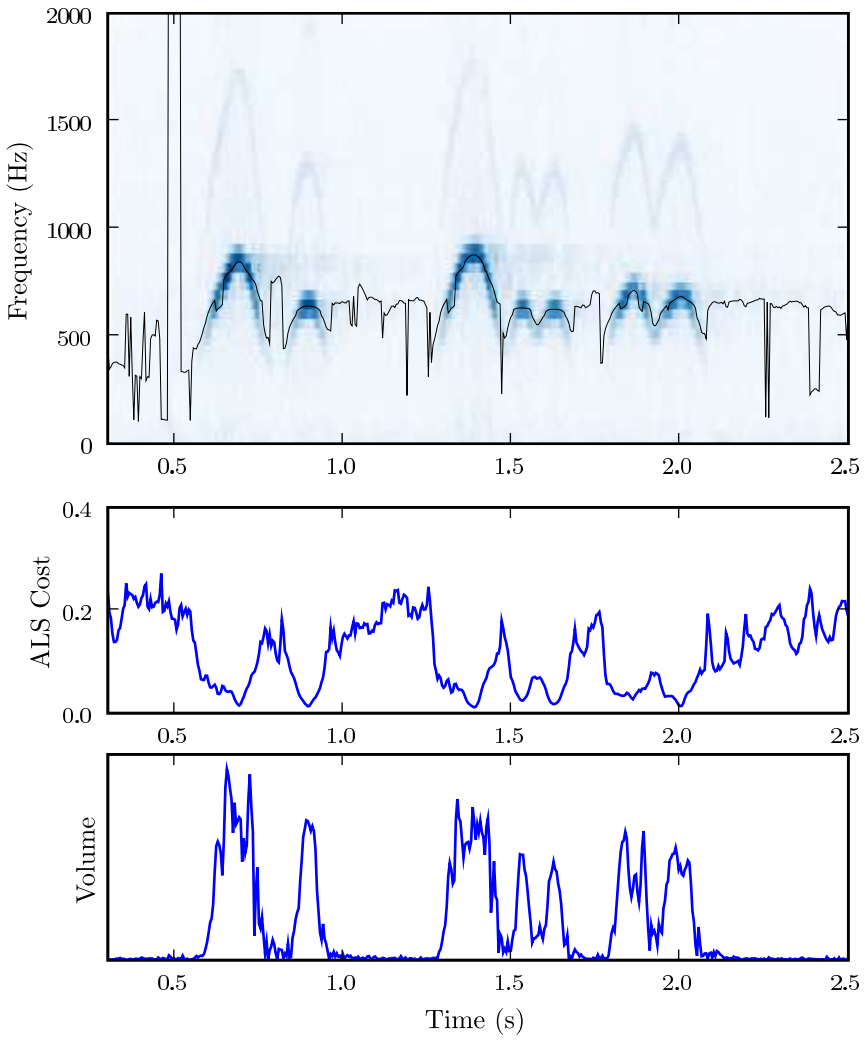
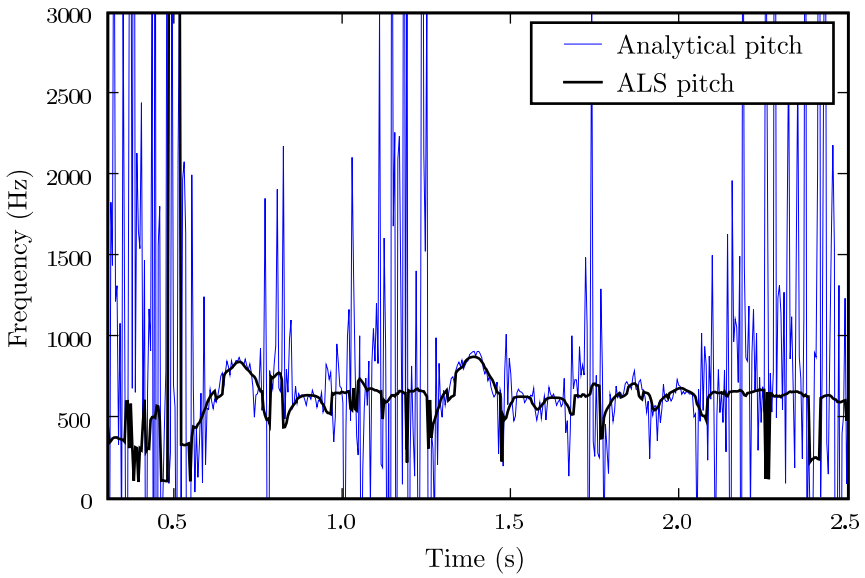
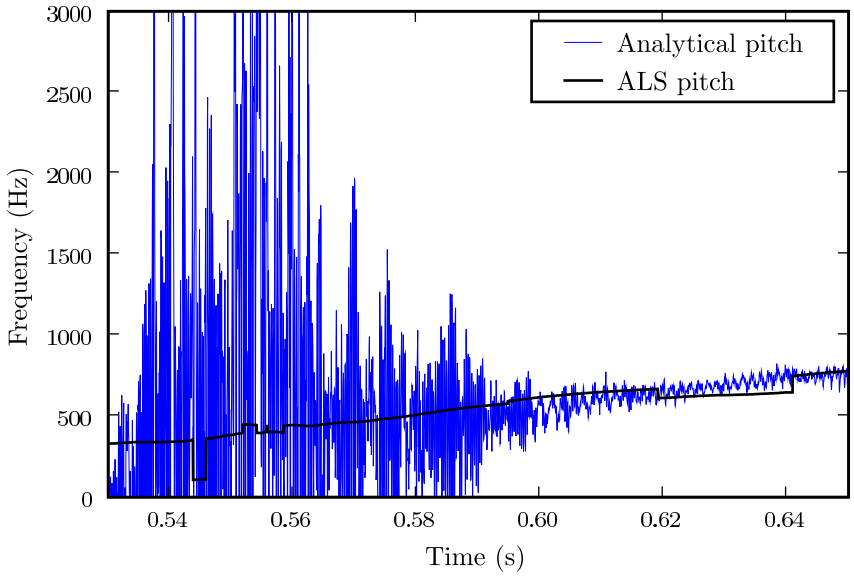


Figure 4.2: ALS pitch track of Wood Owl and silence overlaid on spectrogram. The 22.05 kHz per-sample pitch track was downsampled by a factor of 100 without lowpass filtering. Also shown is the ALS cost function and the analytical envelope, similarly downsampled.



(a) A complete call, downsampled without low-pass



(b) Detail from the beginning of the first note

Figure 4.3: Analytical Pitch versus ALS for the Wood Owl.

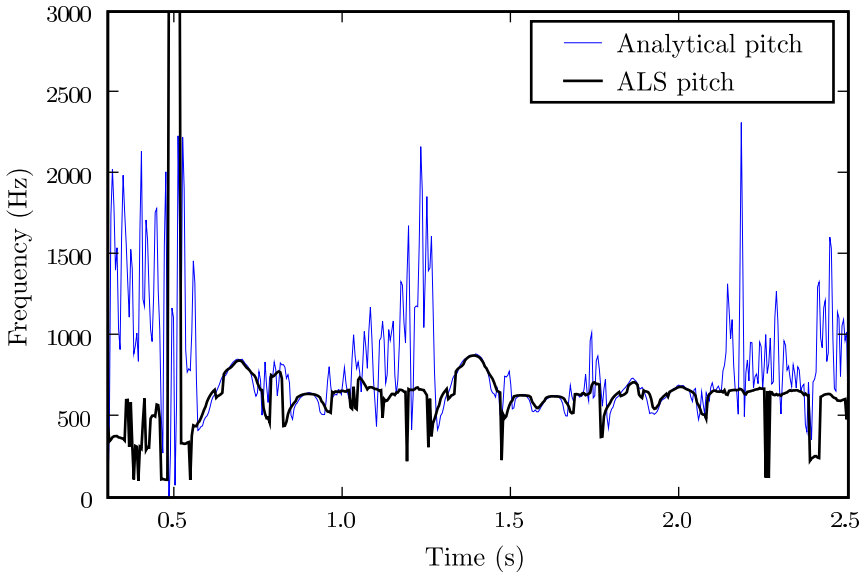


Figure 4.4: Lowpassed Analytical Pitch vs ALS.

4.2.4 Bird Song Synthesis and Comparison

A synthesiser was implemented that synthesises bird calls when given a pitch track and a volume track. These tracks can be at any rate, and are interpolated linearly to the requested synthesiser output sampling rate f_s. Using the interpolated frequency track $f(n)$, a frequency-modulated sinusoid is generated with

$$s(n) = \sin\left(\sum_{i=0}^{n} \frac{2\pi f(i)}{f_s}\right).$$

This sinusoid is then amplitude-modulated by multiplying by the interpolated volume track $v(n)$ to produce the synthesised song,

$$y(n) = v(n)s(n).$$

4.2.4.1 Synthesis from ALS and analytical envelope

The analytical signal can be modelled as a combination AM/FM synthesiser, where the instantaneous frequency specifies the frequency modulation, which is then amplitude-modulated using the envelope. If the frequency is constant, the resulting spectrum is the spectrum of the envelope modulated to the carrier frequency. The instantaneous frequency does not appear to change very rapidly during notes, which makes it similar to an AM system with a slowly varying carrier frequency. The analytical envelope, thus, is the original spectrum,

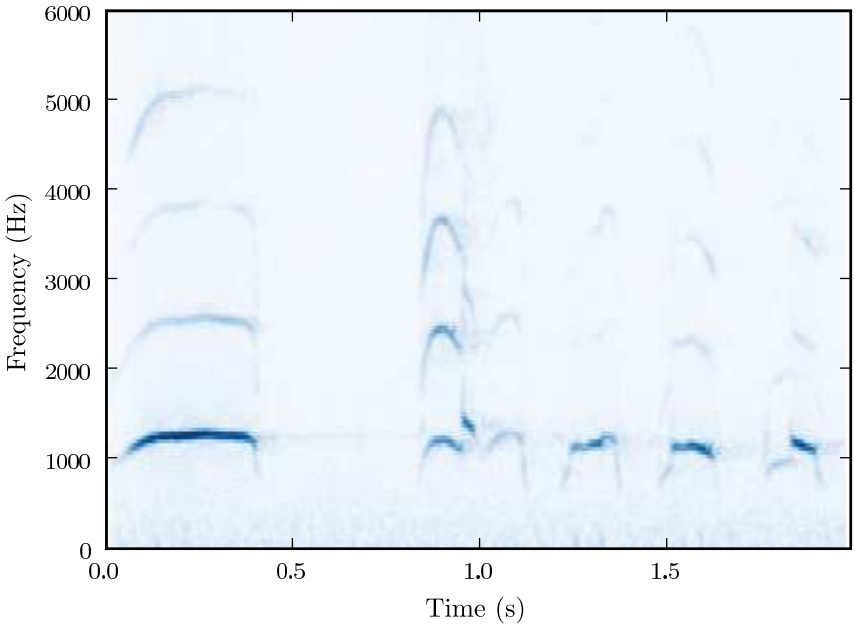


Figure 4.5: Original Spectrogram of African Fish Eagle.

demodulated at the time-varying carrier frequency that is the instantaneous frequency from the analytical envelope. Figure 4.5 shows the original spectrogram of an African Fish Eagle, while Figure 4.6 shows the spectrum of the envelope. Most of the energy is at 0 Hz, but the harmonics are also still present, just mixed down by the same distance as the fundamental. By modulating this envelope back up to the instantaneous frequency, we get the original signal back.

Next, synthesis using ALS and the analytical envelope was investigated. For many birds, the synthesised audio was so close to the original that a casual listening might not even notice the difference. The problematic calls were those from birds whose strongest harmonic is not the fundamental harmonic, and the bird sounds that are not harmonic in the first place (hissing, rasping, etc.) Figure 4.7 shows the spectrum of the African Fish Eagle call when synthesised using the ALS pitch track. The harmonics above the fundamental are contributed by the analytical envelope.

4.2.4.2 Synthesis with simplified amplitude

Features at 22 kHz are too computationally expensive to be of much use, and the data rate must therefore be reduced in some fashion. In the case of the slowly-modulated pitch, simple downsampling is perfectly fine. If the analytical envelope is similarly downsampled without lowpass filtering, aliasing of the

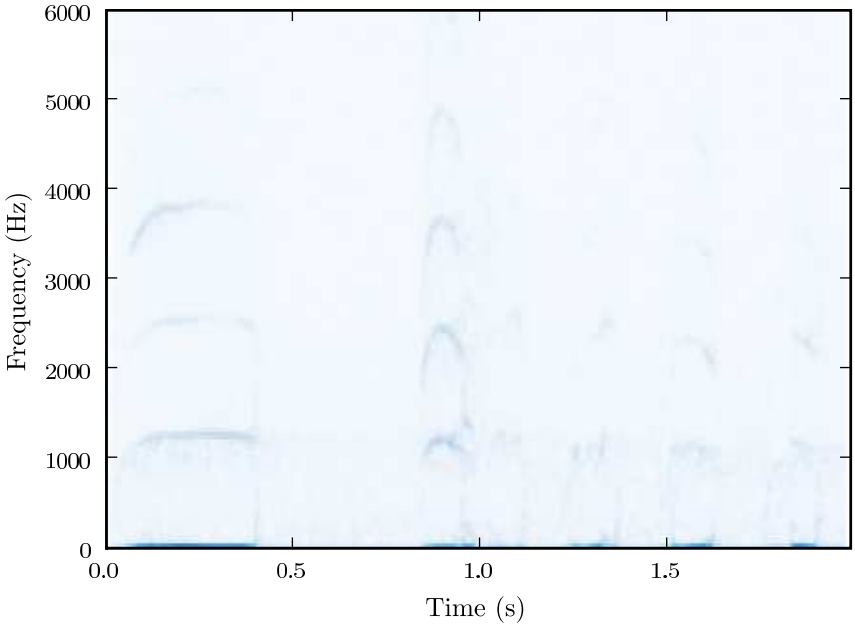


Figure 4.6: Spectrogram of Analytical Envelope of African Fish Eagle.

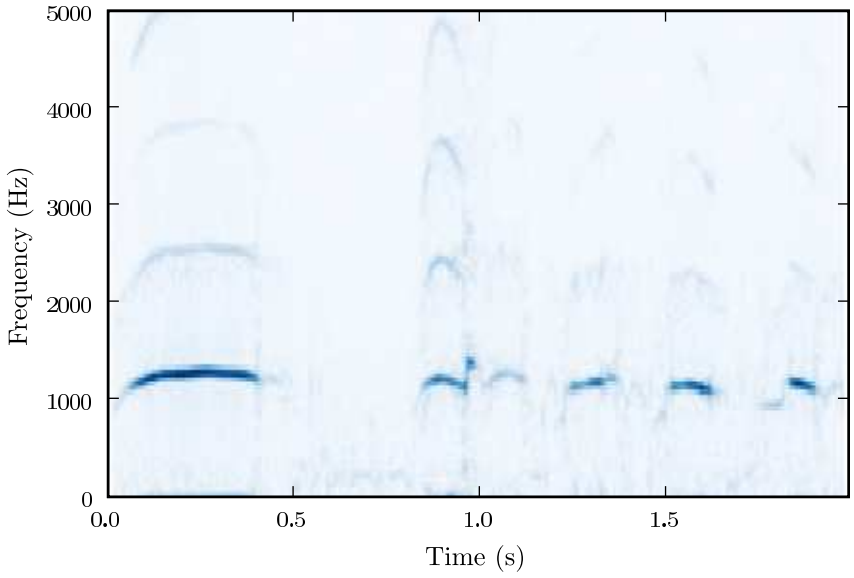


Figure 4.7: Spectrogram of call synthesised from ALS pitch track and analytical envelope (compare to Figure 4.5).

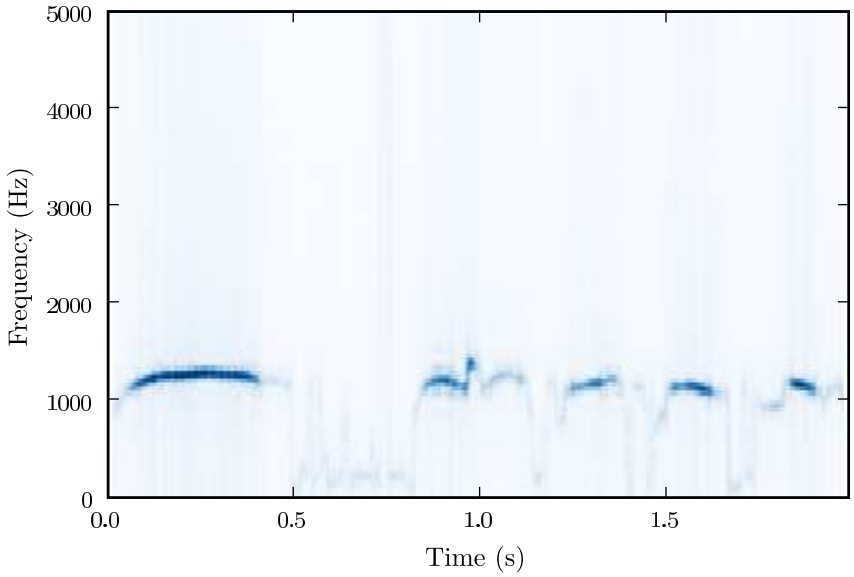


Figure 4.8: Spectrogram of call synthesised from ALS pitch track and analytical envelope downsampled to 220 Hz (without lowpass filtering).

higher frequency harmonics in the analytical envelope (Figure 4.6) occurs. This aliased envelope is then mixed up to the call frequency by the synthesiser.

Figure 4.8 shows the spectrum of the audio when synthesised from the analytical envelope and the ALS pitch track that have both been downsampled by a factor 100. There are small ripples visible on the spectrum as a result of the aliasing.

Aliasing can be demonstrated more clearly by not downsampling as much. Downsampling to 2 kHz and then synthesising produces particularly noticeable aliasing. The spectrum is shown in Figure 4.9.

By lowpass filtering the envelope before decimation, we can remove the aliasing and achieve the spectrum shown in Figure 4.10. This result is very similar to the more regular envelope consisting of the lowpass filtered rectified signal. Because these results are so similar, the latter is not shown.

4.3 Selection of Bird Species for Experiments

As explained in Section 4.1.3, there was a significant shortage of data for adequate scientific representation of the birds' song. In the case of the intelligent passerine singers, multiple tunes may be sung. Humans may be able to recognise song elements and deduce similarities and differences between species using very little data. However, training statistical models for the purpose of recognising

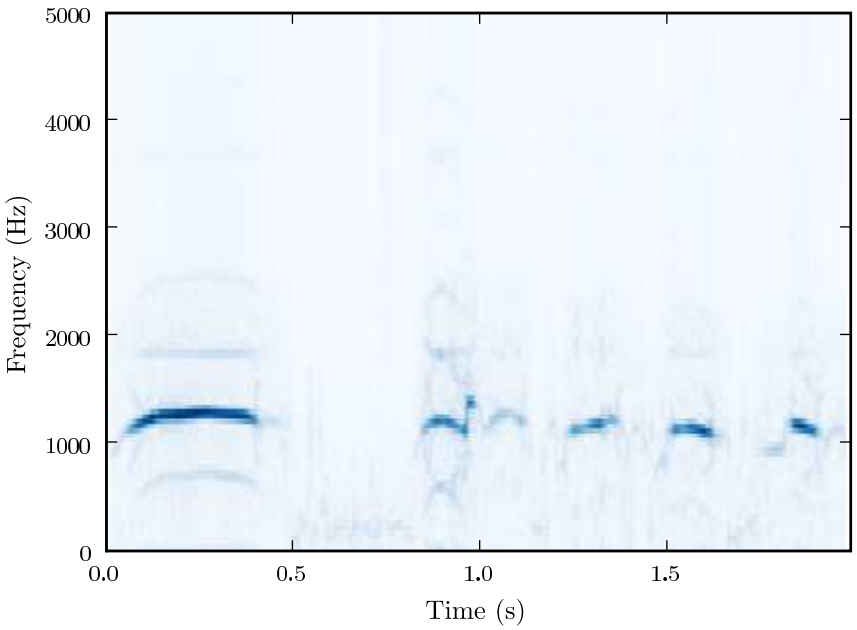


Figure 4.9: Clearly visible aliasing in spectrum at a downsampling factor of 11, effectively downsampled to 2 kHz.

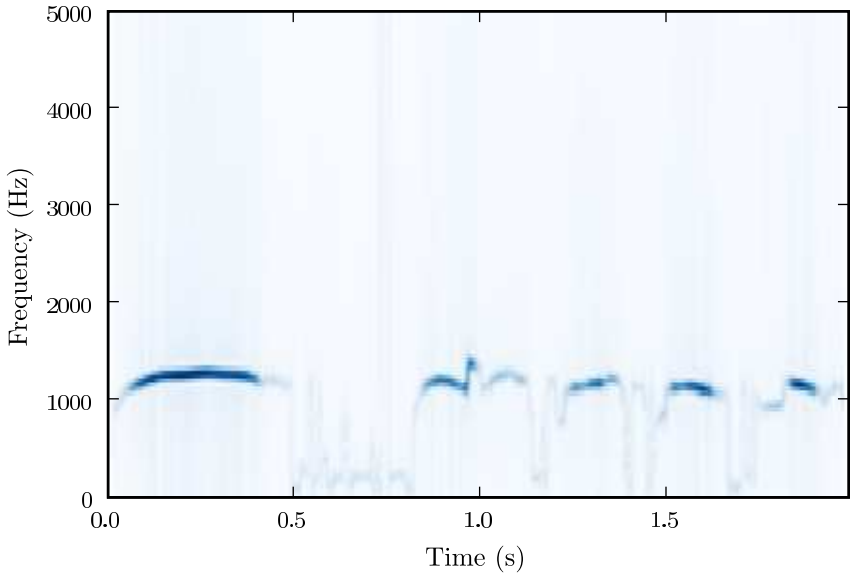


Figure 4.10: Spectrogram of call synthesised from ALS pitch track and lowpassed and downsampled envelope.

such varying songs requires either considerably more data or otherwise expert knowledge. This study focused on the simpler bird songs and calls in order to develop techniques that could be useful in future systems where more data or experience is available. Calls from the following 39 birds were chosen:

African Fish Eagle	Crested Francolin	Swainson's Francolin
Harlequin Quail	Helmeted Guineafowl	Buffspotted Flufftail
Blackbellied Korhaan	Crowned Plover	Greenshank
Spotted Dikkop	Redeyed Dove	Cape Turtle Dove
Laughing Dove	Greenspotted Dove	Knysna Lourie
Purplecrested Lourie	Grey Lourie	Redchested Cuckoo
Black Cuckoo	Emerald Cuckoo	Klaas's Cuckoo
Diederik Cuckoo	Burchell's Coucal	Wood Owl
Marsh Owl	African Scops Owl	Whitefaced Owl
Pearlspotted Owl	Spotted Eagle Owl	Fierynecked Nightjar
Freckled Nightjar	Woodland Kingfisher	Brownhooded Kingfisher
Hoopoe	Redbilled Hornbill	Southern Yellowbilled Hornbill
Pied Barbet	Yellowfronted Tinker Barbet	Greater Honeyguide

4.3.1 Descriptions of some included calls

The *African Fish Eagle* is a very famous call. It is very easily recognised when synthesised with the ALS pitch and analytical envelope, sounding very much like the original. If a lowpassed amplitude feature is used, the timbre contributed by the analytical envelope is lost, but the synthesised call is still very easily recognised. The call usually has one long note followed by three or four shorter notes. For accurate modelling, the last note should be made optional, but that was not done in this study.

The *Crested Francolin* has some interesting texture in its call. Lowpassing and decimation again loses this timbre information, but the bird is still quite recognisable.

Swainson's Francolin also has significant texture. Lowpassed features and decimation throws away a lot of the information that makes it recognisable. It was nevertheless included as a difficult example.

The *Harlequin Quail* has a call consisting of a number of very similar short notes, with the rhythm being the defining characteristic. There are some optional notes which were excluded manually, through the transcriptions.

The *Helmeted Guineafowl* has a repetitive call with two notes and a distinctive rhythm.

The *Buffspotted Flufftail* has one particularly long note which synthesises very well, even lowpassed decimated features.

The *Blackbellied Korhaan* is a tough one. It's call is characterised by a very short popping sound at the end that does not synthesise well. Duration modelling seems very important for this call, as the silence between the first note and the final popping sound seems quite reliable. Even if the synthesised version sounds considerably different from the original, it should still be quite recognisable by the synthesised call's structure and characteristics.

The *Crowned Plover*'s call duration varies a little, sometimes the call is repetitive and sometimes more free-standing. The ALS pitch track is a little inaccurate and the texture information is lost.

The examples of the *Redeyed Dove* and *Cape Turtle Dove* had some background noise. There are also optional notes that were not compensated for.

The *Grey Lourie* (Kwêvoël in Afrikaans) has an unmistakable timbre. ALS struggled significantly to track the pitch correctly, and without the timbre, the bird sounds very different. However, as it still usually has an easily recognised decreasing pitch tendency, it was also included as a difficult example.

4.3.2 Descriptions of some excluded calls

The passerines were excluded due to the variation in their song. To model them statistically would require considerably more training data in order to discover their typical structure.

The *Hadeda Ibis'* call has very many harmonics. The strongest harmonic is also not the fundamental, but rather the third harmonic. In some instances, in the softer initial notes of a call, the fundamental is effectively missing. The fundamental is around 500 Hz, with the third and strongest harmonic at around 1.5 kHz. These birds usually flock together, and their calls frequently overlap in time. The large number of harmonics presents a challenge to pitch tracking algorithms, especially ALS, as does the overlap of the calls of more than one bird.

The recordings of the *Whitefaced Duck* have lots of chatter and many birds overlapping. Although there is a rather characteristic set of three notes that is quite easily identified by a human, extracting the characteristic notes poses a challenge to computer algorithms. Gillard's recording also has significant echo. In the ALS synthesised version, the characteristic three notes can be identified, but they are not clear. Cutting out those three notes with transcriptions would be somewhat unrepresentative, effectively cheating. The chatter is not really identifiable in the synthesised sound. This bird could possibly be modelled with a few HMM states capturing the general chatter, with a left-to-right section

recognising the three notes. The model might want to guarantee there being an instance of those notes, meaning that the left to right part could be made a compulsory centre of the model, with chatter states before and after this centre, and looping back for repetitions.

Egyptian Geese are often found in pairs. The male makes a broadband wheezy hissing sound, while the female's vocalisations are tonal with a rather specific timbre. Songs are often sung in duet, with the two calls overlapping at times. This duet would require a more sophisticated model than is experimented with in this study.

The *Coqui Francolin* poses challenges similar to the Hadeda, in that the strongest harmonic is not the fundamental but rather the third, at approximately 4 kHz. Because of this, the ALS pitch detection has trouble with this one.

The structure of the *Natal Francolin*'s noisy vocalisations are hard to determine in Gibbon and Gillard, having some variety and overlapping often.

The first call in Gibbon for the *Redchested Flufftail* is described as a territorial advertising hoot by the male, heard mainly at night in the breeding season [10]. While it has interesting structure, it differs significantly from the call in Gillard. The second call in Gibbon, an all-year territorial call by both birds, can be recognised in a short part of the Gillard database.

The Gibbon and Gillard recordings for the *Redcrested Korhaan* do not sound the same. More data will be needed to train models for this species.

Chapter 5

Models and Features

5.1 Feature Vectors

In the previous chapter we saw that the pitch and volume tracks probably contain enough information with which to recognise bird song. The actual task of recognition requires modelling these features, for which the first step is parametrising the information in a way that can be modelled.

5.1.1 Absolute and Relative Pitch

Many bird species have a relatively fixed frequency range. For example, the Fierynecked Nightjar typically sings around 1.5 kHz to 2 kHz while the African Fish Eagle typically sings between 1 kHz and 1.5 kHz, although some vocalisations can reach 2 kHz. One of the Harlequin Quail's common vocalisations consists of sounds starting above 2.5 kHz and ending just below 4.5 kHz. The Spotted Eagle Owl typically sings between 300 Hz and 500 Hz. This information is clearly very useful, as it already segments the birds into broad categories. However, there is still some variation within species, which has some significant consequences.

5.1.2 Transposed Tunes and Note Shape Recognition

For best results for songs not built up out of constant-pitch notes, the models must be able to track a pitch trajectory to describe the shape of a note. These trajectories can either be described by dimensions in the feature space, or they can be split up into substates.

Figure 5.1 shows two artificial virtually-identical parabolic pitch tracks, one peaking at 1050 Hz, the other at 1200 Hz. Assume these two notes are sung by two individuals of the same species and are to be recognised by the same

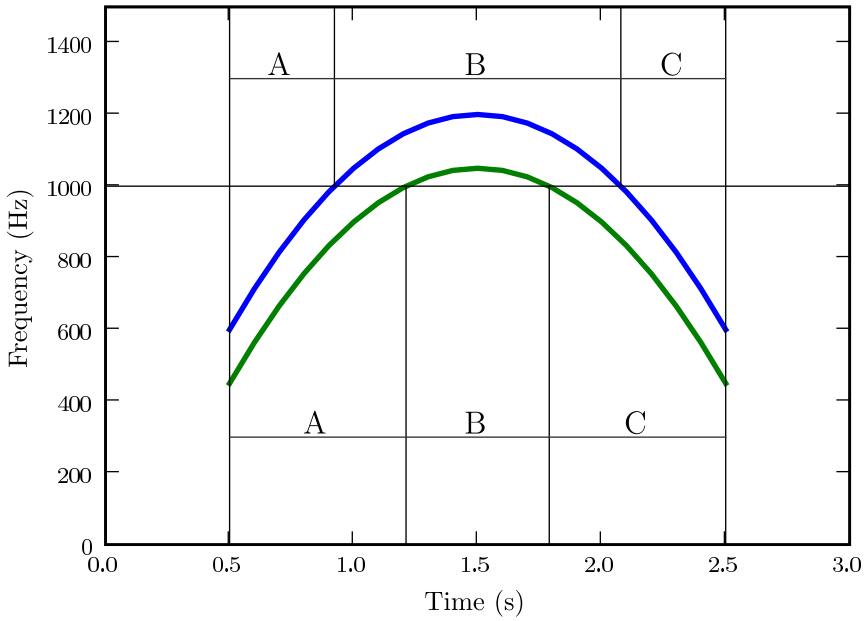


Figure 5.1: The effect of tune transposing on absolute-pitch based features.

model. If the notes were to be modelled by two Gaussian pdfs on absolute pitch only, which discriminate at 1000 Hz, the transcriptions look quite different. If duration modelling is introduced, there would also be some conflict between the duration modelling and the pitch/feature modelling. Low-variance duration modelling could force the transitions, causing higher variance in the Gaussians modelling the pitch. Using absolute pitch in such a simple way is insufficient for accurate note shape modelling if transpositions are involved.

5.1.2.1 Pitch Normalisation

Figure 5.2 shows ALS pitch tracks for two Wood Owl calls from Gibbon, a male and a female. The male sings considerably lower, but apart from that, the tunes are easily recognisable as being the same species. While we could use two different models, one for the male and one for the female, we will model both with the same model as an extreme example of transposition.

In order to compare two pitch tracks at different pitches, we can transpose the two tunes to the same average pitch. This requires calculating the effective mean pitch of each tune. Considering how much the ALS algorithm fluctuates during silence, we need to discount the silent parts somehow. One method would be thresholding to decide where there are notes and where silence, using features such as volume and ALS cost. In order to avoid rigid decisions, we took

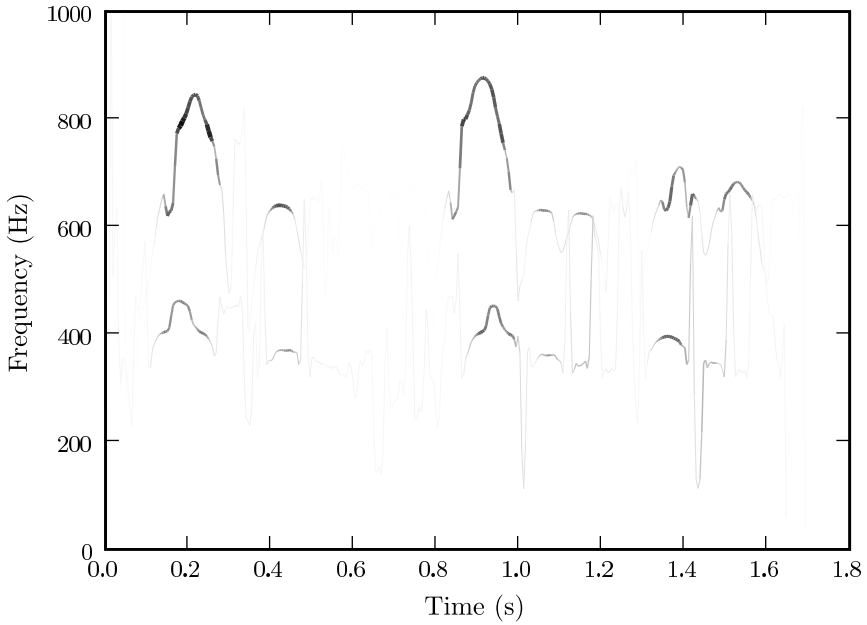


Figure 5.2: Two Wood Owls singing at different pitches. Volume is indicated with grey-scale values as well as line thickness.

a weighted mean instead, weighing the pitch track with the volume. In some sense, weighing by energy (volume squared) may be more correct, but adequate results were obtained by simply using volume.

Pitch is perceived logarithmically. The frequencies of a series of notes separated by equal musical intervals are in fact a geometric series, due to the properties of resonating chambers. The process of transposing a tune up or down thus requires multiplying or dividing by a common factor, rather than adding or subtracting. To simplify the operation, the log domain is used. By subtracting the mean in the log domain, we effectively use a geometric mean. Figure 5.3 shows the effect of normalising the pitch tracks of the same two Wood Owls.

5.1.2.2 Delta Pitch

Another solution to the note shape modelling problem that is also popular in speech is to consider the derivative, the gradient of the pitch track. This feature describes note shape but not note pitch. When dealing with a time-domain signal, it cannot describe the relative intervals between two notes, as there is no stable pitch during silence from which to calculate a delta. In time-domain signals, it can be used in conjunction with absolute pitch or normalised pitch.

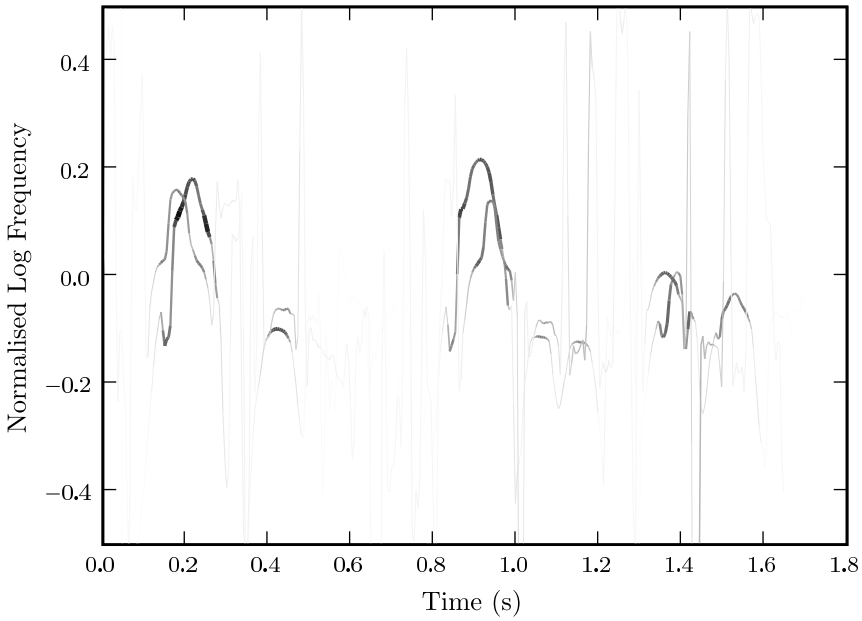


Figure 5.3: Two Wood Owls normalised to the same pitch. Volume is indicated with grey-scale values as well as line thickness.

If we were to work in a *note domain*, where the signal consists of a sequence of notes, taking an intra-note delta becomes trivial. Converting to the note-domain requires note on-set and off-set detection.

5.1.2.3 Parametrised Pitch in the Note Domain

Inter-note deltas, or note intervals, could be useful for describing a tune independently of absolute pitch. The note domain consists of a sequence of notes, detached from the time domain, typically using note duration as one of the feature dimensions. By converting to the note domain, calculating inter-note deltas becomes trivial. Silence can be modelled either as a note with special status or zero volume, or note/silence pairs can be combined into a single observation vector, with one dimension the note duration and another the duration of the silence following the note.

In this study, some note-domain experiments were performed with notes segmented with simple thresholding via volume and ALS cost. Noise on the features could be dealt with using hysteresis. Notes that were separated by short silences were merged, these silences being silences shorter than three milliseconds.

Once note edges have been obtained, note shape can be parametrised in a variety of ways. Notes were parametrised by fitting quadratic polynomials to

the pitch-track. Once again the variability of pitch during low-energy sections is a liability. Again the problem was circumvented without rigid decisions by using a weighted mean, weighing the pitch with the volume.

5.1.3 Volume

Volume was normalised on a call-by-call basis. For each call, the volume was scaled such that the maximum volume was 1.0. In the case of repetitive calls, each whole set of reptitions was normalised as a unit.

5.1.4 Spectral Spread

Another feature investigated for capturing some measure of the timbre, was spectral spread. The spectrogram $X(f)$ was considered a pdf, of which the variance was calculated, thereby effectively proving some measure of the bandwidth of the vocalisation. A passerine with a pure sinusoidal spectrum will thus have a much lower spectral spread than a non-passerine with many harmonics, spreading the energy across the spectrum.

5.2 Comparing Tunes with String-matching

There are a number of birds that sing a relatively simple fixed melody with simple notes. These birds may be recognisable by tune recognition methods similar to those employed by McNab et al [19]. The tune, consisting of volume and pitch tracks, are transformed into the note domain for further recognition by approximate string matching.

5.2.1 Note Segmentation

Converting a tune into a sequence of discrete items, or notes, requires determining note onset and offset events. A simple implementation through the thresholding of pitch and volume was implemented, with notes being taken as areas where both the volume is greater than a certain threshold and the ALS cost (uncertainty) is less than 0.15.

Simple thresholding results in some edge effects which could be removed through use of hysteresis instead of a simple threshold. A simpler method was used though, namely merging notes that are separated by less than three milliseconds of silence. Merging silence sections separated by short notes was also investigated, but it seems unnecessary to do both.

Once the tunes have been segmented into notes, they were parametrised to reduce memory and computational requirements.

5.2.2 Polynomial fitting

One way to parametrise pitch is to make a weighted least squares fit of a parabola onto the pitch track.

With n samples in a note's pitch track, there are n equations of the form $\left(at_i^2 + bt_i + c\right) - p_i = \mathcal{E}$. For the best fit, a, b and c must be chosen to minimises $\mathcal{E}$. Let

$$A = \begin{bmatrix} t_1^2 & t_1 & 1 \\ t_2^2 & t_2 & 1 \\ \vdots & \vdots & \vdots \\ t_n^2 & t_n & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$$

Now the note's pitch track can be parametrised by finding the least squares solution of $A\mathbf{x} = \mathbf{p}$. This is too dependent on the exact choice of note start and finish though, as the ALS pitch track may become unstable near silence. The weighted least squares fit is calculated with volume serving as weighting, through $A'\mathbf{x} = \mathbf{p}'$ and

$$A' = \begin{bmatrix} v_1 t_1^2 & v_1 t_1 & v_1 \\ v_2 t_2^2 & v_2 t_2 & v_2 \\ \vdots & \vdots & \vdots \\ v_n t_n^2 & v_n t_n & v_n \end{bmatrix}, \ \mathbf{p}' = \begin{bmatrix} v_1 p_1 \\ v_2 p_2 \\ \vdots \\ v_n p_n \end{bmatrix}$$

This ensures that the polynomial-fitting places more emphasis where the note has greatest energy, as the volume is the square root of energy, reducing the problem of rigid note boundaries.

The volume of notes were parametrised through a simple root-mean-squared value, $\hat{v} = \sqrt{\overline{v_i^2}}$.

5.2.3 Note Comparison

For an approximate string match using the dynamic programming algorithm from Section 2.5, a distance measure to measure the difference between two notes is necessary.

McNab et al [19] developed a system for folk tune indexing which can be queried by singing a melody. Folk tunes are easily recognisable by humans irrespective of what key[1] they are sung in. As perfect pitch is rare, there is very little guarantee what key a melody will be sung in by a human. Typically a human may transpose a melody in order to make it fit into their vocal range.

[1] In music theory, when a melody is transposed to a different pitch, it is said to be sung in a different key.

As a result, recognising human-sung tunes typically requires tune matching on relative pitch. McNab et al used the intervals between successive notes and performed string matching on these intervals rather than the absolute pitch of the notes themselves.

In order to translate this idea to the domain of bird song, the ways in which bird song is similar and differs must first be determined. Within a species bird tunes are typically much more fixed in pitch, as mentioned previously. Gibbon's example of the Wood Owl includes a pair of birds duetting. The female has her lowest notes at around 650 Hz and her highest notes up to 950 Hz, while the male has it's highest note at 500 Hz and it's lowest below 400 Hz. While different models could be used for the male and the female, this study combined the two into one model as an extreme example of transposition and relative pitch.

Numerous approaches are possible. The mean note could be used to approximate the bird song with a simple sequence of notes similar to that in folk tunes. The string distance can then be calculated based on rhythm and relative pitch (intervals from one note to another) in a very similar fashion to folk tune recognition by McNab et al. The pitch was matched separately from the rhythm, with the tune distance being a linear combination of the distances of the pitch and rhythm (weights determined heuristically).

Another idea was to first find a trace from one sequence to another using rhythm and note shape only and ignoring the absolute pitch. Once this is done, the difference in pitch can be compared between corresponding notes. The better the match, the closer to zero the variance and the mean of the pitch differences between corresponding notes will be. The mean is a measure of how much the tune has been transposed, while the variance is a measure of how well the tune matches in the relative pitch sense.

Since identifying the beginning and the end of a note is not trivial, dealing with note fragmentation and consolidation is important. This requires that the distance function be able to handle comparisons between one note and a set of notes (consolidated set, fragmented set, or empty set in case of deletions or insertions).

Note distance measures experimented with, were

- rhythm comparison, where notes (including "silence notes") were characterised by their length only. Fragmentation and consolidation are easily compared by simply summing the length of the consolidated notes, or the length of the fragments.

- note shape and rhythm with the sum of absolute difference disregarding the actual pitch, and comparing in the log-domain. (This can be determined by subtracting the median difference from one of the two tunes

before finding the sum of the absolute difference.) Pitch differences were normalised and clipped to one octave. Where note/silence or a length mismatch occurred, the mismatched areas were considered to have maximum difference (i.e. the same as one octave). If the notes to be compared had differing lengths, a time-lag was used that minimises the difference measure, with the restriction that the shorter note stays within the boundaries of the longer note (or set of fragments or consolidated notes). The time-lag implementation was prohibitively expensive.

- the Sum of Squared Errors was also considered, disregarding the actual pitch, as this allows finding an analytical representation for the minimum difference

$$\begin{aligned} n_0(t) &= a_0 t^2 + b_0 t + c_0 \\ n_1(t) &= a_1 t^2 + b_1 t + c_1. \end{aligned}$$

We want to calculate, with k chosen to minimise:

$$y = \int_\alpha^\beta \left(n_1(t) - n_0(t) + k\right)^2 \mathrm{d}t$$

Let $At^2 + Bt + C = n_1(t) - n_0(t) + k$. The value of y is minimised with respect to k when $\frac{dy}{dk} = 0$, which reduces to:

$$C = \frac{2A\left(\alpha^3 - \beta^3\right) + 3B\left(\alpha^2 - \beta^2\right)}{6\left(\beta - \alpha\right)}$$

with $A = a_1 - a_0$ and $B = b_1 - b_0$. The integral can then be evaluated at:

$$y = \left[\frac{1}{5}A^2t^5 + \frac{1}{2}ABt^4 + \frac{1}{3}\left(2AC + B^2\right)t^3 + BCt^2 + C^2t\right]_\alpha^\beta$$

This formula can be used to determine the sum of squared error difference in note shape between two notes, starting at α and ending at β. Dealing with fragmentation and consolidation sets in this case requires calculating the distance over the separate polynomial sections independently and summing the result. A time-lag variable can also be introduced. Without fragmentation and consolidation, a closed-form solution to the difference equation can be deduced. With fragmentation and consolidation, it again becomes a non-linear operation, which could be solved by finding

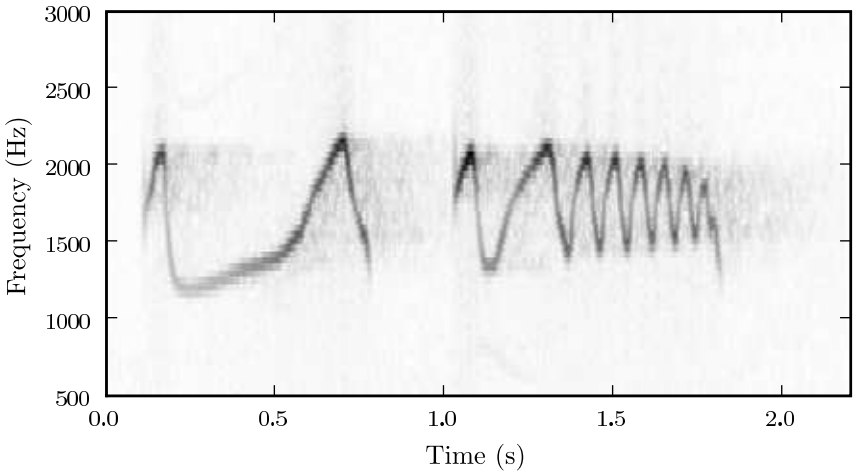


Figure 5.4: Fierynecked Nightjar spectrogram.

the best-fitting polynomial for the fragmented note set as a whole before comparing.

5.2.4 Problems with the string-matching approach

Some optimisation and tree-pruning tricks were implemented to speed up the non-linear sum-of-absolute-difference string matching approach, but it was still prohibitively expensive. The linear versions with closed-form solutions looked promising though, or alternatively a cheaper approximation than brute-force for the ideal time-delay calculation.

This is not the only problem with the string-matching approach though. Hard segmentation of notes remain error prone. Also, the bird songs that can be accurately transcribed as a melody consisting of such simple notes are rather limited. Accurate recognition of a broad range of non-passerine birds requires recognising their timbre as well. Many passerine birds, on the other hand, have a more diverse repertoire and sing varied melodies. Many birds that sing a fixed melody and are characterisable by a pitch trajectory are not necessarily characterisable by a sequence of discrete notes, particularly not parabolically parametrised notes. Consider for example the Fierynecked Nightjar, shown again in Figure 5.4. Dividing this tune into separate notes would either require representing it as two notes with a larger number of parameters describing the shape and oscillating pitch properties, or dividing it up into smaller sections and modelling it as more than a dozen separate fragments.

This solution to the problem still depends on selecting a good template, which is not necessarily possible. It also does not handle time-warping. While

it might work for a couple of very specific birds, a model able to deal with structural variations in the tune is more important, for example missing notes, variable number of repetitions or optional segments. Nevertheless, without enough data, it may be a necessary compromise. More sophisticated statistical modelling techniques can be used if more data can be found.

5.3 Comparing Tunes with Hidden Markov Models

The Hidden Markov Model is a statistical modelling tool used to great effect in speech recognition. As mentioned previously, DTW is still quite popular in the field of bird song recognition. Just as DTW was popular in speech recognition and later supplanted with more sophisticated Hidden Markov Models, HMMs may be very useful in bird song recognition.

5.3.1 Rhythm Modelling Concerns

As explained previously, the rhythm information is of great significance to tune recognition when dealing with features as simple as pitch and volume. Even if more intricate features are used, the information in the rhythm should ideally not be neglected. This means either finding some way to encode the rhythm into the features, or relying on the HMM structure to model the rhythm.

While it is possible to include some rhythm information in the features [15], this study explored the use of HMM duration modelling in order to achieve rhythm recognition. In principle, this is similar to using a constrained DTW algorithm (Section 2.4.2). In comparison to a DTW-like algorithm implemented with observation pdfs, the greatest advantage in using HMMs is significantly more flexibility in defining song structure. HMMs are not limited to left-to-right topologies like a standard DTW algorithm. More flexibility for constraining or guiding rhythm or duration modelling is available, thanks to the transition weights.

5.3.2 Decimation

The primary features the models dealt with were the pitch and volume tracks described in Section 4.2. ALS can provide per-sample pitch which is typically overkill. The data-rate can be significantly reduced through decimation. The pitch is typically not very fast changing, especially not where it matters, so low-pass filtering is not needed prior to downsampling. Furthermore, implementing

a lowpass filter would smooth over the step-wise discontinuities, which may even be disadvantageous.

Some birds have very slow changing sounds and do not require a high temporal resolution. Other birds have quick and short notes. Some of the shortest notes are as short as 5ms. While a higher decimation factor is possible for the low temporal resolution songs, a significantly higher frame rate is needed, i.e. a lower decimation factor, in order to catch the short notes of songs that have a higher temporal resolution.

Following some experimentation, a decimation rate of 110 was chosen for the 22050 Hz features, resulting in an observation frame rate of approximately 200 Hz.

5.3.3 Finding Workable Models

As mentioned in Section 5.3.1, this study primarily employed HMM structure and duration modelling to recognise rhythms.

Traditionally, in speech recognition, the features used are of great importance in modelling the sounds, while the HMM models provide the general structure but not particularly much information. Each state in an HMM models a particular sound produced by the vocal tract, otherwise a transition from one sound to another. The sequence of sounds is important for recognising a word, while the duration is not of very much importance. While some accuracy improvements can be achieved by adding duration modelling, most speech recognition systems use a simple selfloop resulting in the geometric duration distribution, as discussed in Section 2.7.1.

In recognition problems like these, the standard procedure is often to first seek the best features for modelling each sound. This works when each sound is distinct and produced by a specific shape of the vocal tract. Once good state separation is achieved, the features are tested in various models to find what works best. In the case of many birds, however, there is no significant difference between various sounds. This is particularly true for passerines. The only information in such cases is the tune being sung, consisting of pitch, volume and rhythm. It is not possible to isolate different states solely on information such as pitch and volume. Figure 5.1 in Section 5.1.2 illustrates clearly how the relativity of pitch complicates matters. The interaction between pitch and rhythm is fundamental to the recognition of bird song, so the interplay between the model structure and the state pdfs are of great importance. The traditional approach of first choosing the best features and then developing the HMM structures should be reconsidered.

To get around this chicken and egg problem, an iterative approach is used.

As the rhythm modelling is such an integral part, very basic features (pitch and volume) were first picked, and numerous models tested to find a promising rhythm modelling solution. Once some basic promising rhythm modelling has been achieved, alternative feature vectors can be investigated. The model structure can be revisited and the original decision re-evaluated yet again.

5.3.4 Datasets

Recognition was attempted for 47 calls from 39 birds. About half of these calls, or 23, were considered repetitive calls, where a relatively simple unit is repeated an indefinite number of times, while the other half, or 24 calls, were longer, more complicated free-standing calls. Some of the repetitive calls have the simple call repeated continually for more than 20 seconds. For example, Gibbon's recording of the Southern Yellowbilled Hornbill has more than a hundred repetitions of a simple note. The doves (Redeye, Cape Turtle and Laughing Doves) have more complicated calls, repeated less than 10 times, but still clearly repeated a relatively arbitrary number of times. Lastly, Klaas' Cuckoo was also somewhat inaccurately modelled as arbitrarily repetitive, as it was the simplest way to deal with the varying call length (Gibbon's recordings have two repetitions of the note pair, while Gillard's recordings have three). The ideal for this bird is probably to have a model that is aware of the common number of repetitions. For example, the third repetition could be optional, by adding a skip link over that section of the model.

For the non-repetitive calls, there were 121 examples to train with, as well as 121 to classify. For simplicity, repetitive calls were first implemented with models that modelled and recognised a precise number of repetitions. If call pairs were used, there were 354 examples of the 24 calls to train with and 314 examples to classify, while if call triplets were used, there were 226 to train with and 195 to classify.

There are some potential pitfalls in collecting results. When using call triplets, there were two birds for which there were nearly 30 testing samples. One was Burchell's Coucal, which has a couple of seconds' worth of repeated calls containing about two dozen notes with slightly varying tempo and pitch. The other was the Spotted Dikkop. If we score results based simply on the number of correct classifications, any change to the models or features that benefit these birds to the detriment of birds with fewer than 4 examples, will be considered a good result. This bias is even worse when dealing with call pairs, when there are 40 examples of Burchell's Coucal while nine of the 24 repetitive calls have 18 or more examples in the testing set.

This bias towards the repetitive birds is unsatisfactory. To work around this

problem, weighted scores were also calculated, weighing each kind of call equally by dividing the tally of correctly classified examples by the number of examples available.

5.3.5 Transcriptions

Two sets of transcriptions were used. The first, which we will call the *macro transcription*, identifies the calls of each bird. Macro transcriptions were used for both the training and evaluation sets, and for both HMMs and the string-matching approach. Figure 5.5 shows an example of macro transcriptions for the Wood Owl and the Redeyed Dove. Both have two calls that were modelled in this study (the Wood Owl calls that were labelled "half" were ignored). The Redeyed Dove has repetitive calls.

A second set of transcriptions, named *micro transcpritions*, were used for initialisation and structural purposes by some of the HMMs. Figure 5.6 shows these micro transcriptions for the first Wood Owl call and both Redeyed Dove calls. Silence was labelled with the postfix "sil" while notes were labelled by an integer postfix suggesting the number of substates (see Section 5.3.6.2). In repetitive cases like the Redeyed Dove, two repetitions were transcribed by each micro transcription, with the silence that the loop occurs on, labelled with a "sil.loopback" postfix.

5.3.6 Basic Models

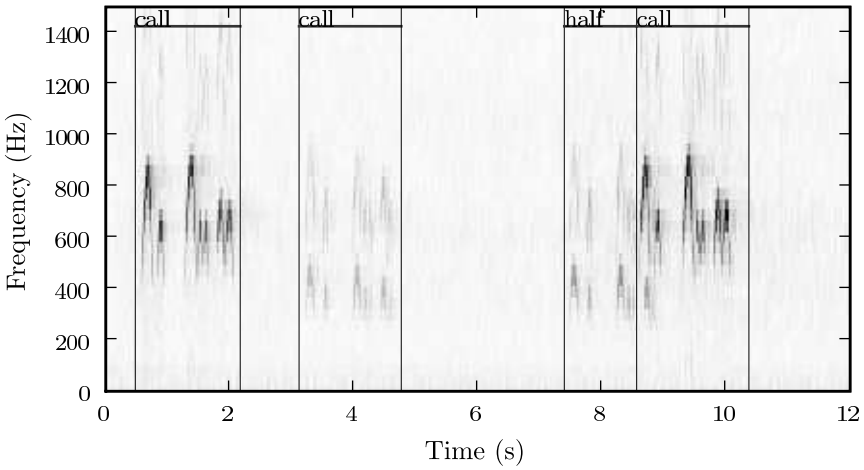
In this section, the models implemented and experimented with are described. Results are given in the next chapter.

5.3.6.1 Left-to-Right models

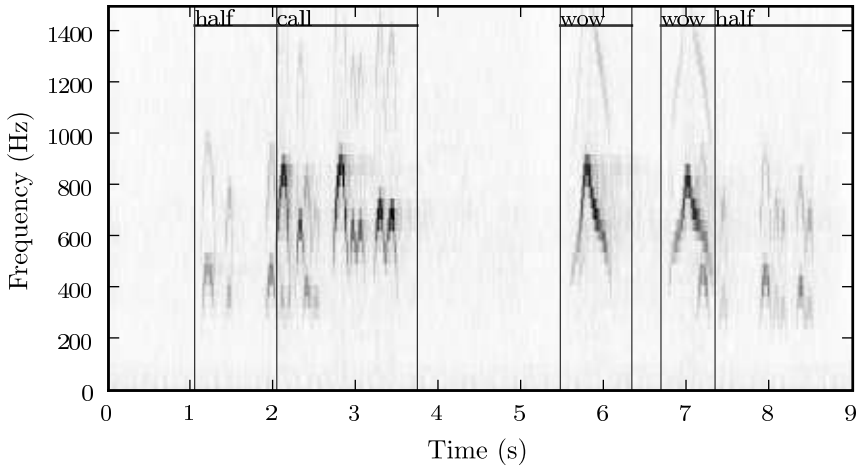
The first model to be implemented was a simple left to right model, as shown in Figure 5.7. Various states-per-second rates were implemented, ranging from ten states per second, to forty. Initialisation took place according to evenly spaced subdivision of the call transcriptions. In the case of looped calls, this subdivision was done over a triplet of calls. A varying number of training iterations were also experimented with.

5.3.6.2 Smarter left-to-right models

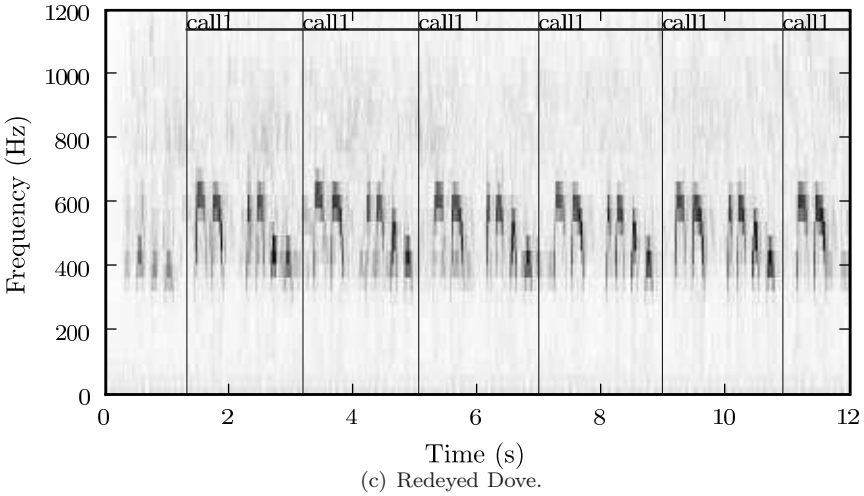
For the next experiment, the micro transcriptions were used to indicate where there is silence and where there are notes, as well as how many states are desired for each note, typically two, three, or four depending on the complexity of the



(a) Wood Owl, part 1.

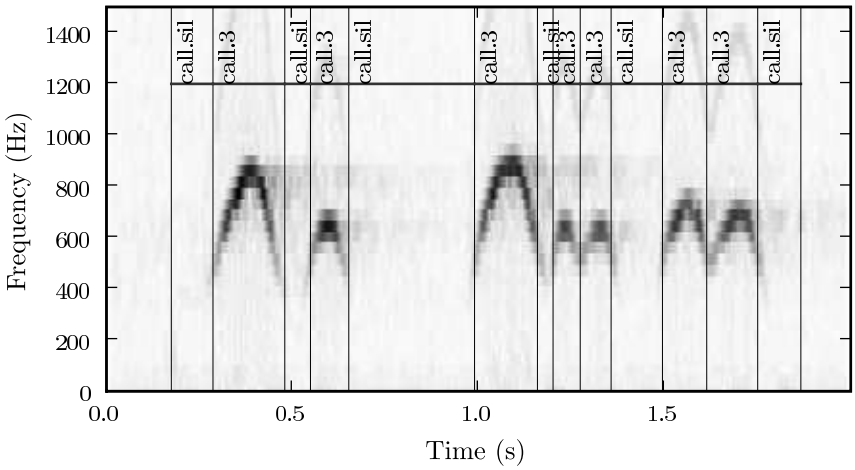


(b) Wood Owl, part 2.

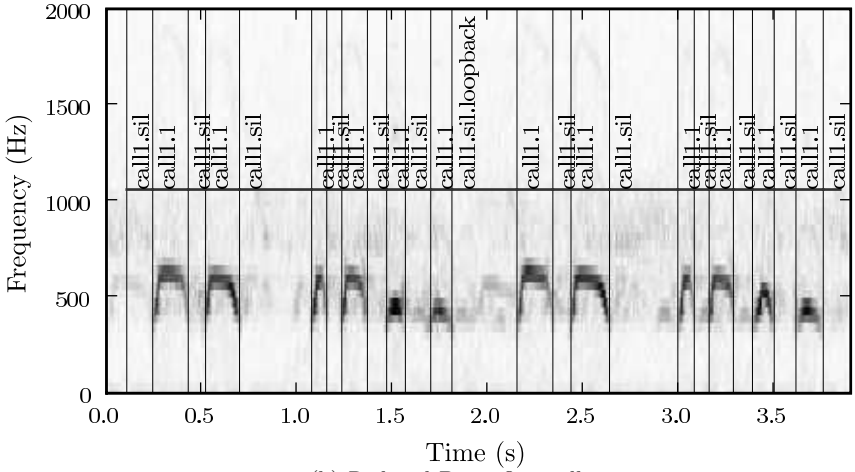


(c) Redeyed Dove.

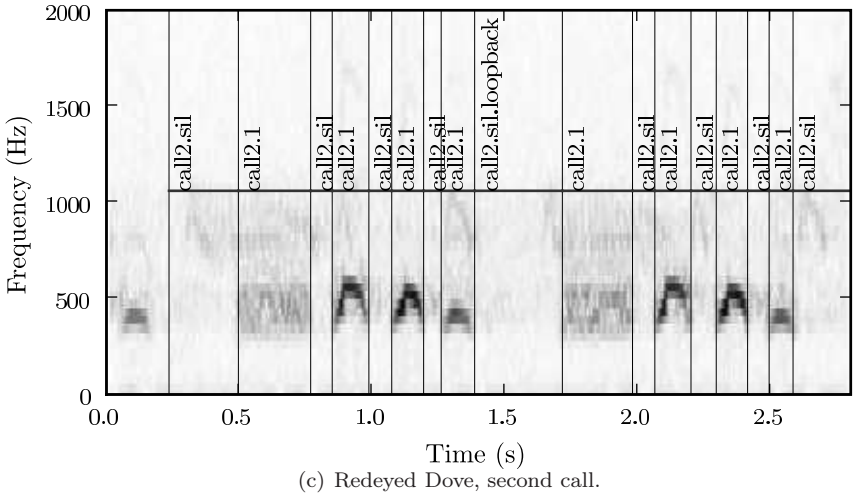
Figure 5.5: Examples of macro transcriptions.



(a) Wood Owl.



(b) Redeyed Dove, first call.



(c) Redeyed Dove, second call.

Figure 5.6: Examples of micro transcriptions.

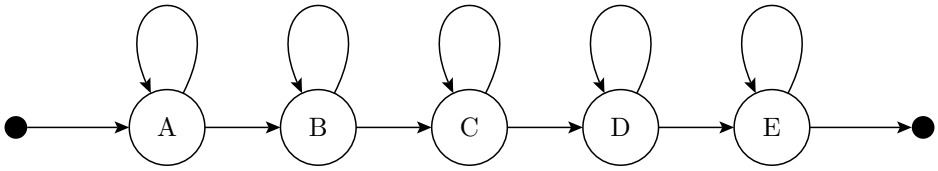


Figure 5.7: Left-to-right HMM.

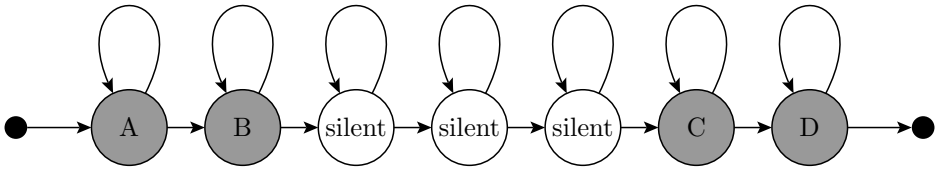


Figure 5.8: A smart left-to-right model

note's pitch track. (See Figure 5.6 — the Redeyed Dove was assigned only one state per note, while the Wood Owl was assigned three.)

To initialise the models, the silence was again divided into states according to a specified number of states per second. The notes, however, were split evenly into the number of states specified in the transcription.

Each bird model had its own silence pdf, shared over all its silence states, which results in Compounded-Selfloop Duration Modelling for the silence between notes. The selfloops could be tied together to ensure they remain equal, but that was unnecessary because the training algorithms are stable — if the selfloops are initialised to be equal, they remain equal for each iteration.

An example of one of these models is shown in Figure 5.8 , with two states per note and a silence modelled by a three-state CSDM configuration. One detail that is not shown due to lack of space, are the silence selfloop states at the beginning and end of the model. These selflooped states are required for the purpose of dealing with imperfect transcriptions.

5.3.6.3 Looped models

The simplest looped model shown in Figure 5.9 has one state for a note and one state for a silence, together with the standard selflooped states to deal with imperfect transcriptions.

5.3.6.4 Split looped model

A split looped model, similar to the looped model of Section 5.3.6.3 but with the looped-note split according to the micro-transcription, is shown in Figure 5.10. In this case, the silence is modelled by a simple selfloop state.

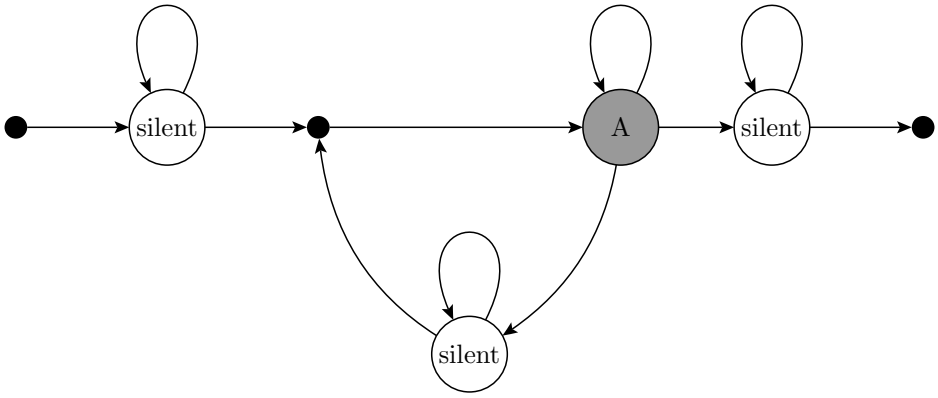


Figure 5.9: Simple looped model

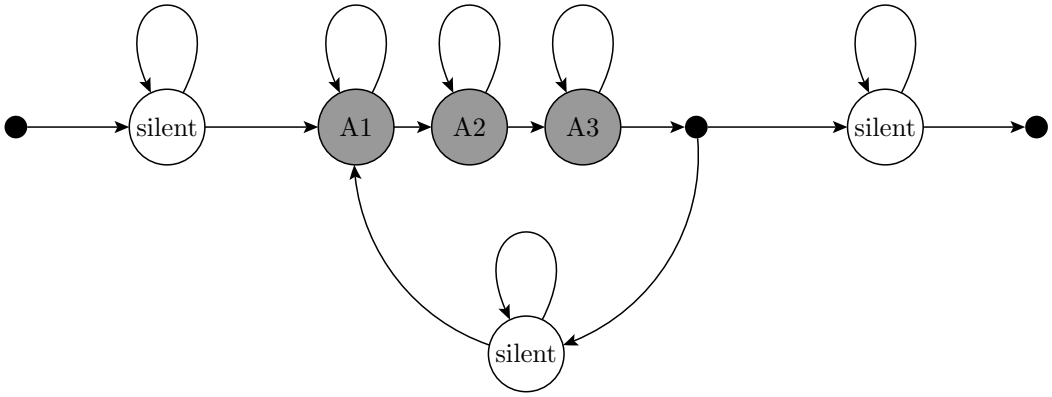


Figure 5.10: Split looped model.

5.3.7 More Sophisticated Model Variations

5.3.7.1 Frozen Models

By freezing duration weights, we can ensure we only train the pdfs without modifying the duration modelling. This is useful if the specified initialisation distribution should be considered authoritative.

5.3.7.2 Duration Modelling Variations

By using Frozen models, we can replace every selfloop with some duration modelling structure with an equivalent mean. Duration modelling structures investigated included Ferguson stacks and CSDM.

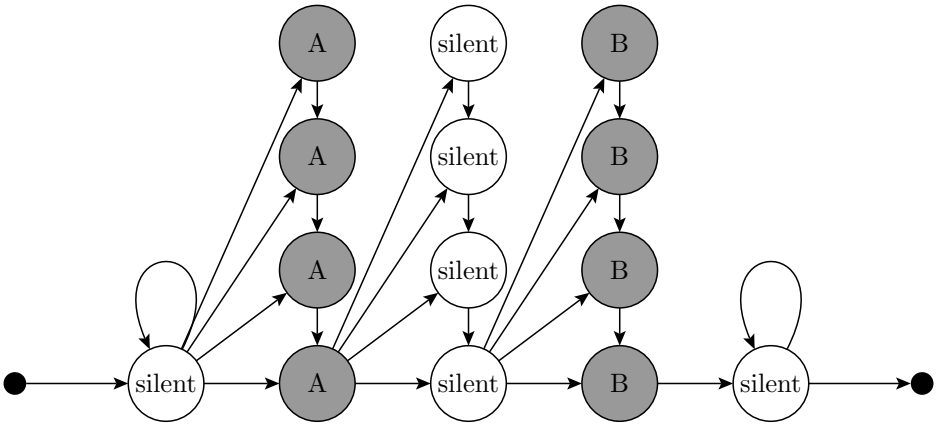


Figure 5.11: Simple Ferguson model for a two-note call.

5.3.7.3 Hierarchically Built Models

Building models hierarchically allows building particular duration distribution structures in the core of the model, still maintaining the first and last selflooped states to deal with imperfect transcriptions.

5.3.7.4 Examples

Figure 5.11 shows a constrained Ferguson-stack based model, with no selfloops. Each column represents one particular pdf. In this case, it is a non-repeating model with two notes. While this example has stacks of size four, they can be any length. With no selfloops, the stacks would have to be large enough to accommodate the longest allowed duration for each state. By setting the first and last states to frozen selflooped states with specified frozen weights, we can ensure that no information can be gleaned from any pattern in the transcription imperfections. This can be important for repeating patterns where the transcriptions are marked to start and finish in more or less the same place every time. A very similar model that can be used for repetitive calls is shown in Figure 5.12.

By taking the micro-transcriptions into consideration, we can model the calls and songs in more detail. Figure 5.13 shows a case where each note is split into two stacks. As all silence can share the same pdf, the inter-note silence can be modelled by a single stack. Again, the stack heights may be considerably higher. Figure 5.14 shows the equivalent structure for repeating calls with three states per note.

Figure 5.15 shows a CSDM model for use with calls with two notes and no repetition. By using more than three states, as shown in this example, the variance of the CSDM structure can be reduced. Figure 5.16 shows a similar

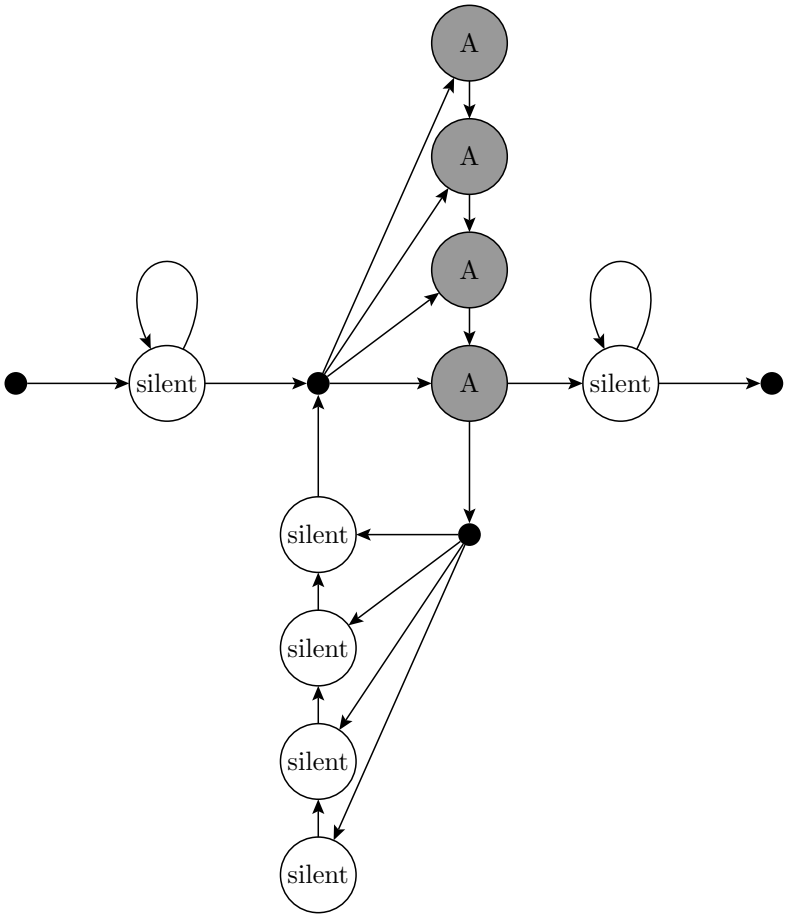


Figure 5.12: Looped Simple Ferguson.

CSDM-based model for repetitive calls.

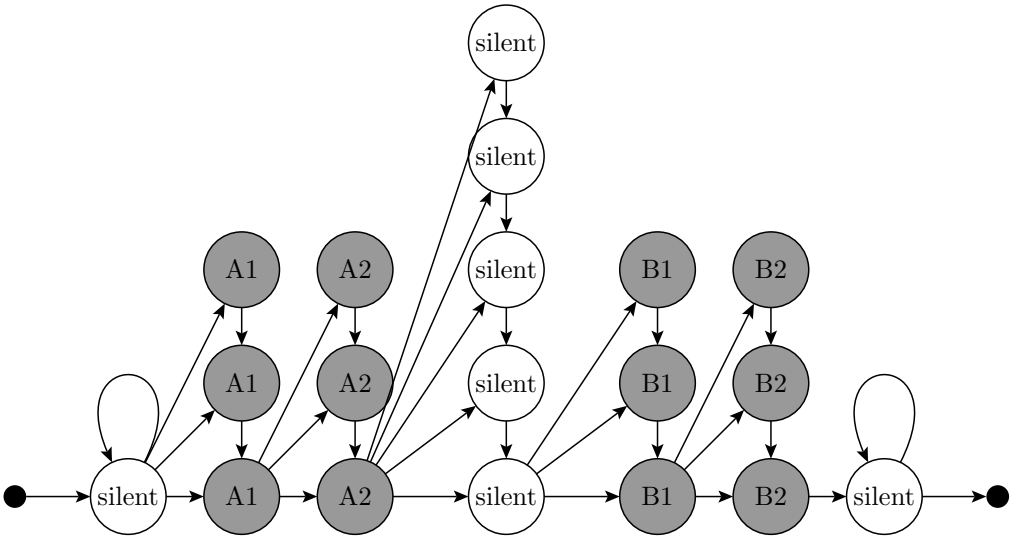


Figure 5.13: Split Ferguson.

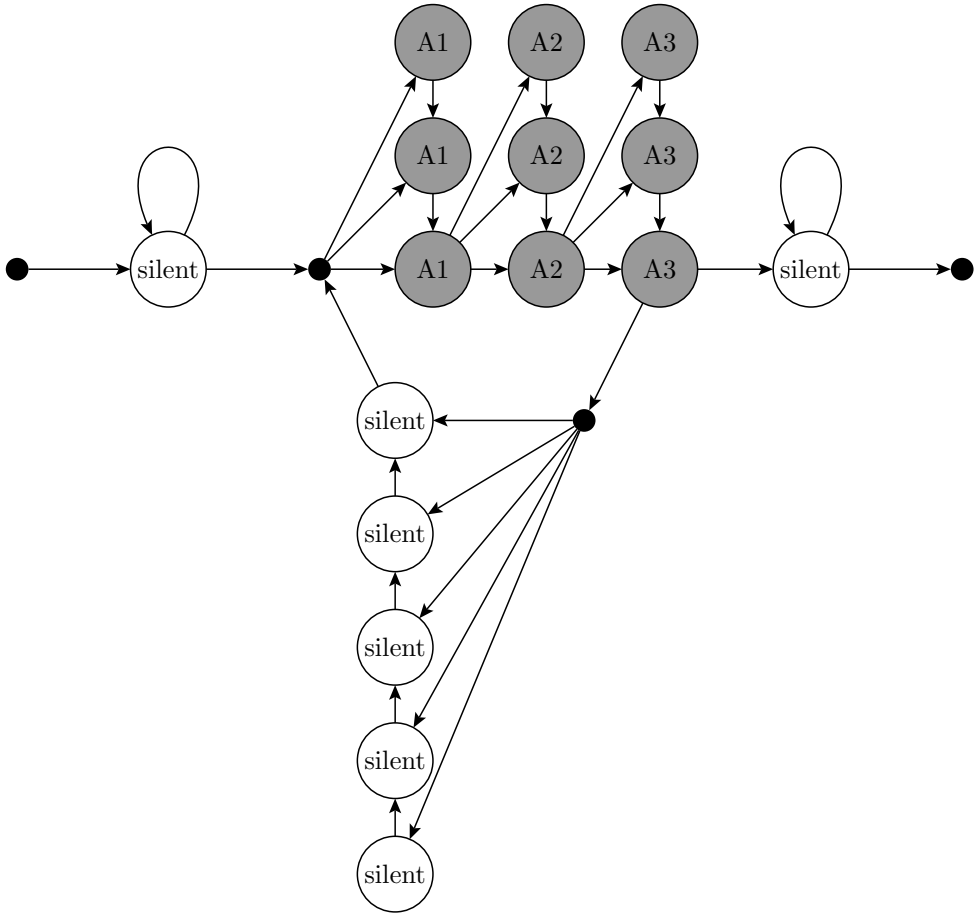


Figure 5.14: Looped Split Ferguson.

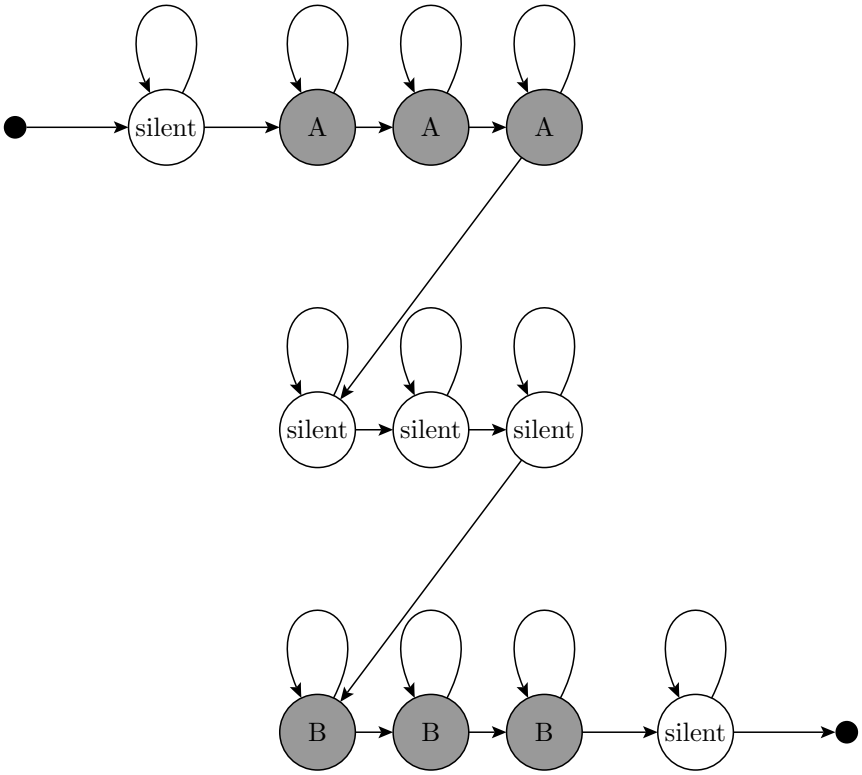


Figure 5.15: CSDM duration modelling.

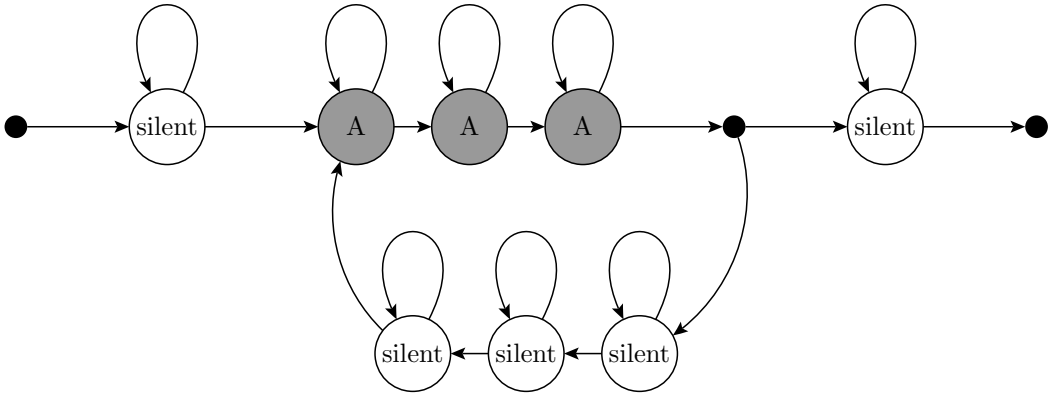


Figure 5.16: Looped CSDM HMM.

Chapter 6

Experimental Results

6.1 Approximate String Matching

While the approximate string matching approach was investigated in some depth, there were a number of problems with it. More effort was spent on the HMM approach. The non-linear approach was prohibitively expensive and the note segmentation was not very robust. Note segmentation also does not work for all birds. The rigid segmentation was not very robust, and spurious notes could have a great effect on trace accuracy.

One experiment calculated a trace disregarding absolute pitch but using note shape (approximated by a quadratic polynomial). Once the trace was available, the difference in the absolute pitch components between each pair of notes or note-sets (in the case of fragmentation or consolidation) was calculated. If we disregard the note shape for a moment, the mean of each difference is a measure for how much one tune must be transposed to get the other. If the tunes match, the differences should all be constant. If the tunes do not match, there will be some variance in the note differences.

This provides us with a two-dimensional feature as comparison between two tunes: the string-to-string edit distance calculated while determining the trace, and the variance of note differences. Figure 6.1 shows a scatter plot for calls from three birds, with each call compared to Gibbon's first example of a Wood Owl call. The data point at (0,0) is the distance and variance from this template call to itself. We see the separation is not particularly good, but there is a basic trend. The correct data points that are separable from the rest are the other samples from the same specimen in Gibbon. The call samples from Gillard lie within a cloud of incorrect matches.

This method was abandoned in favour of the more robust HMM approach.

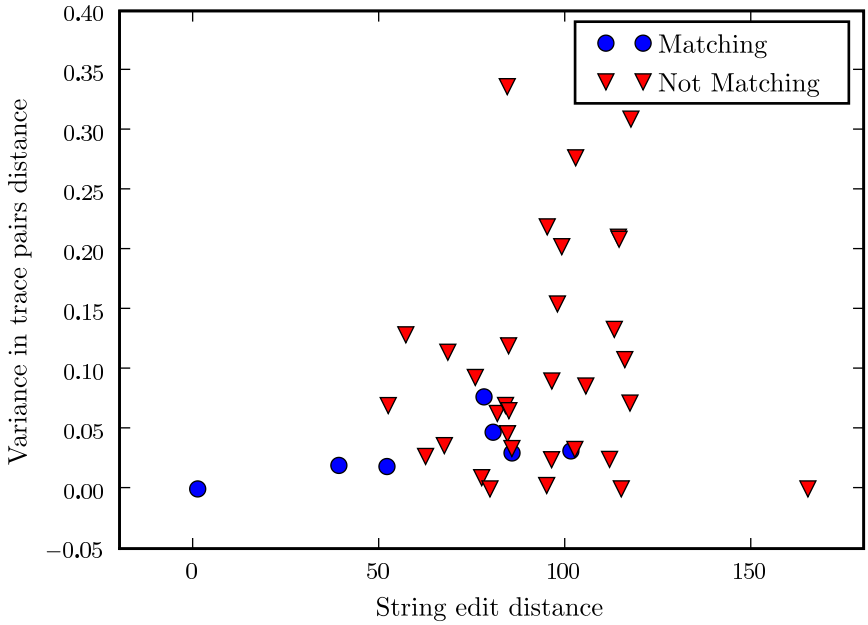


Figure 6.1: Scatter plot for scores for Wood Owl and three other birds. The x-axis shows the edit distance calculated while determining the trace, irrespective of absolute pitch.

6.2 Hidden Markov Models

In this section, all results shown are weighted recognition accuracies (Section 5.3.4) expressed as percentages. With close to 50 calls to recognise, the expected recognition accuracy for a random classifier is under 3%.

While full-covariance Gaussian pdfs were tested for states, they did not seem to be any better than diagonal covariance Gaussian pdfs. Also, when doing higher-dimensional experiments, there was too little training data for full-covariance Gaussians. As such, the remainder of these experiments were all done with diagonal covariance Gaussians.

6.2.1 Datasets Overview

Tables 6.1 and 6.2 show how many examples of each call was used from Gibbon and Gillard. In most cases, Gibbon and Gillard each has only one specimen of each bird species, for each call. For the repetitive calls, classification was typically performed on sets of two or three repetitions. Training of models that can handle repetition occurred over whole sets of repetitions, but classification was still done on only two repetitions in order to keep the results consistent.

Non-repetitive calls		Gibbon	Gillard
Call Code	Bird Species	Calls	Calls
148.call	African Fish Eagle	3	5
201.call2	Harlequin Quail	4	4
218.call	Buffspotted Flufftail	3	4
238.call	Blackbellied Korhaan	3	4
354.call3	Cape Turtle Dove	3	8
358.call	Greenspotted Dove	2	2
373.call	Grey Lourie	4	10
377.call	Redchested Cuckoo	10	8
378.call1	Black Cuckoo	4	4
384.call	Emerald Cuckoo	6	6
386.call	Diederik Cuckoo	6	3
394.call	Wood Owl	4	2
394.wow	Wood Owl	2	3
395.call	Marsh Owl	3	5
396.call	African Scops Owl	7	7
397.call	Whitefaced Owl	4	4
401.call	Spotted Eagle Owl	5	3
405.call	Fierynecked Nightjar	4	8
408.call	Freckled Nightjar	12	6
433.call	Woodland Kingfisher	6	3
435.call	Brownhooded Kingfisher	7	12
451.double	Hoopoe	10	3
451.triple	Hoopoe	9	7

Table 6.1: Non-repetitive calls.

6.2.2 Baseline

For a baseline result, we used the simple left-to-right models shown in Figure 5.7 with diagonal Gaussian pdfs on each state. The pdfs modelled two-dimensional feature vectors consisting of normalised pitch and volume. The models were initialised by dividing transcriptions evenly with a specified number of states per second. In the case of repetitive calls, call triplets were used. Klaas's Cuckoo was excluded as Gibbon's recording only has a call pair with no triplets. The results are shown in Table 6.3, for a variable number of training iterations of the Forward-Backward algorithm. The results are relatively sporadic. With too many states, the model can specialise quite easily, which is why more training iterations are detrimental to classification accuracy.

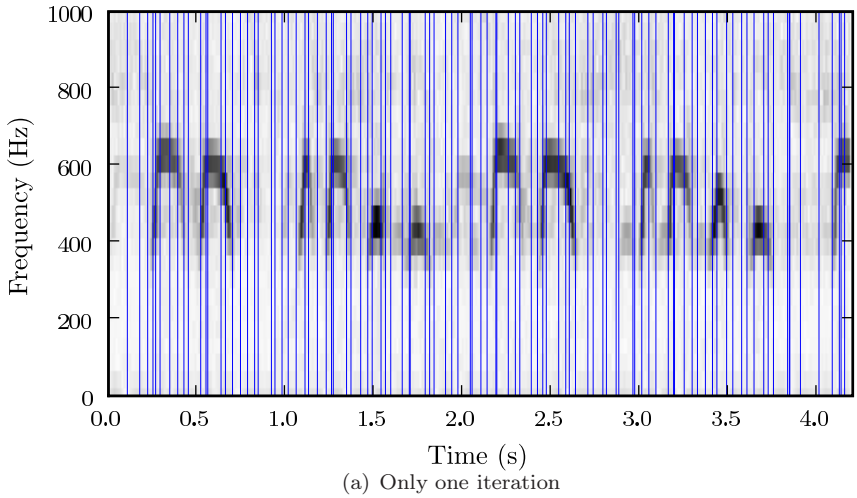
Consider adjacent states i, j and k with equal selfloops $a_{ii} = a_{jj} = a_{kk} = a_L$. We then have the complementary weight $a_{ij} = a_{jk} = 1 - a_L = a_K$. If states i and j have to match n feature vectors, with all other things being equal including the state pdfs, the contribution of the transition probabilities to the total probability will be $a_K^2 a_L^{n-2}$, i.e. the number of observation vectors assigned to each state is

Repetitive Calls		Gibbon		Gillard	
Call Code	Bird Species	Sets	Reps	Sets	Reps
189.call1	Crested Francolin	2	25	3	10
199.call	Swainson's Francolin	2	18	4	27
203.call	Helmeted Guineafowl	1	17	1	5
255.call	Crowned Plover	8	58	3	38
270.call	Greenshank	6	33	12	62
297.call	Spotted Dikkop	1	51	2	83
352.call1	Redeyed Dove	1	8	1	5
352.call2	Redeyed Dove	1	5	1	5
354.call2	Cape Turtle Dove	1	4	1	2
355.call	Laughing Dove	1	9	2	5
370.call	Knysna Lourie	3	45	3	39
371.call	Purplecrested Lourie	2	36	2	38
378.call2	Black Cuckoo	2	17	1	8
385.call	Klaas's Cuckoo	5	10	4	12
391.call	Burchell's Coucal	2	39	4	83
398.call1	Pearlspotted Owl	1	30	2	45
398.call2	Pearlspotted Owl	1	12	2	14
405.call	Fierynecked Nightjar	1	9	1	7
458.call	Redbilled Hornbill	1	74	2	56
459.call	Southern Yellowbilled Hornbill	1	121	1	38
465.call1	Pied Barbet	2	28	2	24
465.call2	Pied Barbet	2	27	1	13
470.call	Yellowfronted Tinker Barbet	1	40	1	29
474.call	Greater Honeyguide	2	17	1	13

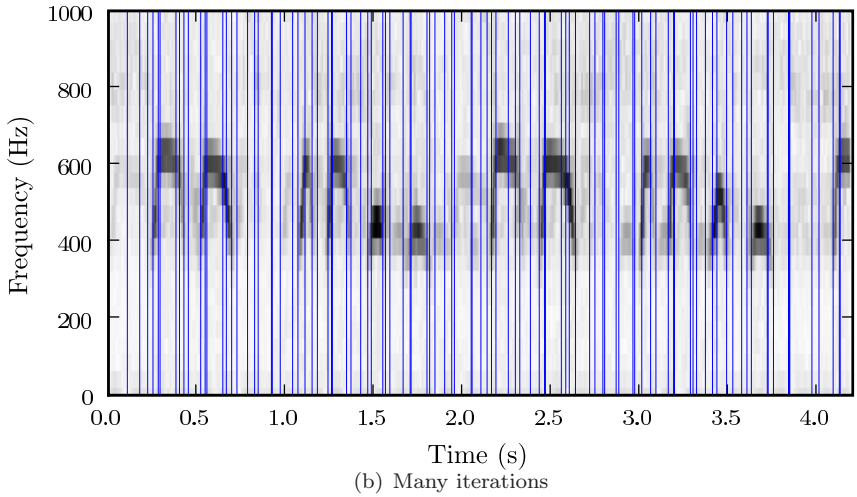
Table 6.2: Repetitive calls. This table shows how many repetitions there were of each repetitive call, and into how many sets these repetitions were grouped.

	Weighted Classification Accuracy (%)		
States per Second	1 iteration	4 iterations	10 iterations
10	28.0	**28.2**	28.1
15	31.9	**33.6**	32.7
20	**28.7**	27.2	27.0
25	**28.2**	24.4	25.8
30	29.3	**30.2**	26.8
35	**30.8**	30.6	28.1
40	**30.4**	26.5	25.7

Table 6.3: Baseline results.



(a) Only one iteration



(b) Many iterations

Figure 6.2: An example demonstrating some specialisation. With only one training iteration, the resulting Viterbi transcription is more *regular*. With many iterations, the excessively simple pdfs on each state allows the model to specialise. See for example the first note. (These spectrograms and transcriptions are of the initialisation sample.)

not relevant. As soon as one selfloop has a higher probability than the other, however, with all other things being equal, the state with the more likely selfloop will swallow all the feature vectors it can, leaving the state with lower selfloop probability with only one feature vector. This unstable characteristic is what causes states to specialise if their pdfs are not different enough, and is the most likely cause of the over-specialisation seen in the results, for example at 40 states per second with ten training iterations.

Figure 6.2 demonstrates the effect of over-training. A simple model can

	Triplets	Doublets		
States per Second	1 iteration	1 iteration	4 iterations	10 iterations
10	28.0	26.7	30.7	31.6
15	31.9	32.6	**33.5**	31.7
20	28.7	32.5	27.2	27.1
25	28.2	31.5	25.8	28.4
30	29.3	30.4	29.7	27.4
35	30.8	28.6	27.7	28.6
40	30.4	28.9	28.1	26.9

Table 6.4: Recognising doublets instead of triplets.

	Classification Accuracy		
Silence States per Second	1 iteration	4 iterations	10 iterations
10	26.7	30.4	29.9
15	28.4	29.9	30.7
20	29.0	30.3	31.4
25	29.7	30.1	30.7
30	29.4	30.2	29.1
35	29.6	30.8	**31.8**
40	30.1	30.5	30.9

Table 6.5: Smart left-to-right model implementing silence duration modelling.

specialise with particularly low variance on some states. This relates to the unconstrained DTW algorithm. By adding some constraints, the result can be more stable and regular. In comparison, DTW is an untrained model. The lack of state distinction is not a problem, as the state pdfs and transition weights are effectively chosen and not trained.

Table 6.4 shows results when repetitive calls are split into doublets instead of triplets. The difference does not seem particularly significant. For further more interesting experiments, only doublets were used.

6.2.3 Split left-to-right models

The next experiment made use of the micro-transcriptions (Section 5.3.6.2) to divide notes into a specified number of sub-notes. Each sub-note was represented by a single selflooped state. The inter-note silence were again modelled by a varying number of states per second. Table 6.5 shows the accuracy generally increases for more silence states per second. This most likely demonstrates how modelling the intra-note silences with increasingly lower-variance Gaussian approximations is beneficial to overall recognition accuracy.

As shown in Table 6.5, increasing the number of silence states per second has a clear increasing effect on the recognition accuracy. Because of the shared

	Unfrozen weights			Frozen weights		
Training Iterations	1	4	10	1	4	10
Looped	16.7	19.1	16.0	15.3	19.8	16.6
Split Looped	19.1	23.3	22.4	19.2	23.7	21.2
Ferguson	21.9	19.6	19.1	20.8	23.0	22.7
Split Ferguson	22.6	26.1	25.4	25.2	**27.8**	**27.5**

Table 6.6: Classification accuracies for looped models.

silence state, all adjacent silence states (with shared pdf) have the same self-loops after training, they form a "compound state", implementing left-to-right duration modelling, with decreasing variance, as shown in Figure 2.16. Clearly implementing duration modelling with more accurately specified durations can increase recognition accuracy.

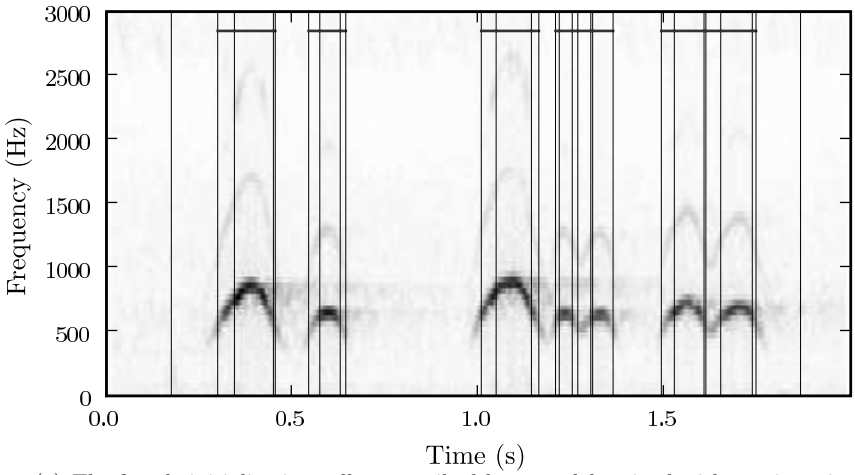
6.2.4 Normalisation and Transposition

Figures 6.3 and 6.4 show the effect of no normalisation and normalisation on attempting to model both the male and female Wood Owl. In both cases, the transcription of the female Wood Owl in the training set seems adequate, but without normalisation, the states misalign on the male Wood Owl, also from the training set.

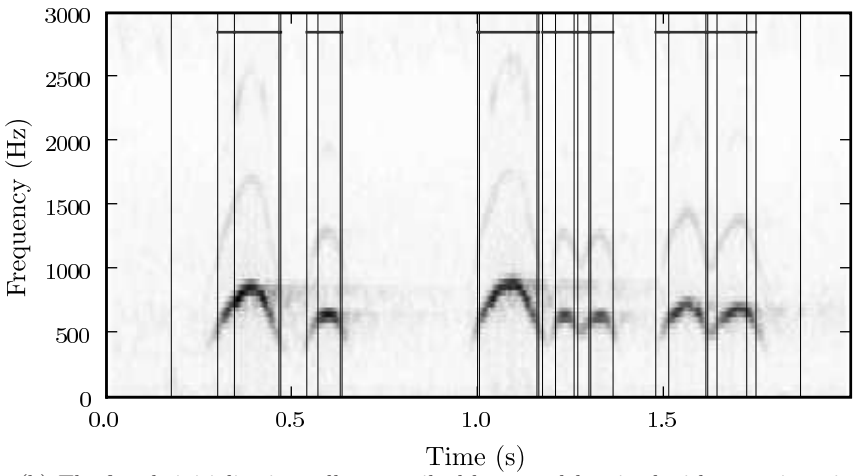
While this is an extreme example and normalisation does not provide a perfect transcription for the male, it demonstrates that normalisation is at least a viable solution to the transposition problem. In conjunction with duration modelling to make the transcriptions more accurate, Figure 6.4 (c) shows a promising result.

6.2.5 More Intricate Models

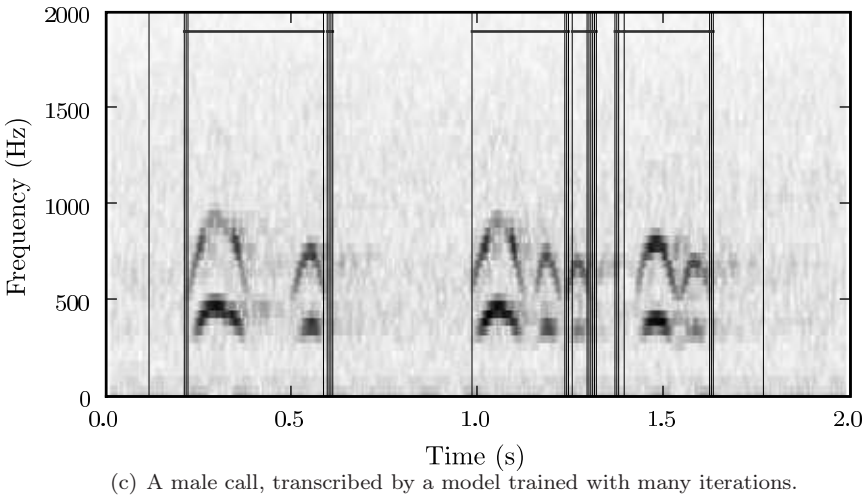
The next four models implemented were more intricate. In each case, the state pdfs were two-dimensional diagonal Gaussians with normalised pitch and volume. The simplest was the looped model illustrated in Figure 5.9, the second was the smarter looped model in Figure 5.10. The next two were a mirror of these two, except the selfloops were replaced by Ferguson stacks. Each stack was the same size, and implemented a discrete (sampled) Gaussian duration pdf. The Gaussian pdfs were initialised to have the same mean as the selfloop it replaces and a standard deviation of 10% of the mean. In each case, every call model had its own independent silence model. The results are shown in Table 6.6. Notice how splitting the notes into sub-notes seems advantageous. In each case, training helped improve results up to a certain point, but started specialising after more iterations. The duration modelling from the Ferguson



(a) The female initialisation call, transcribed by a model trained with one iteration.

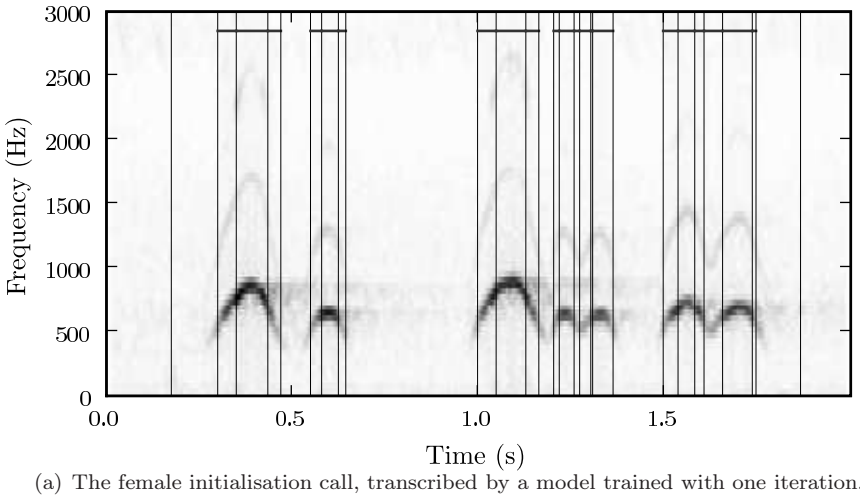


(b) The female initialisation call, transcribed by a model trained with many iterations.

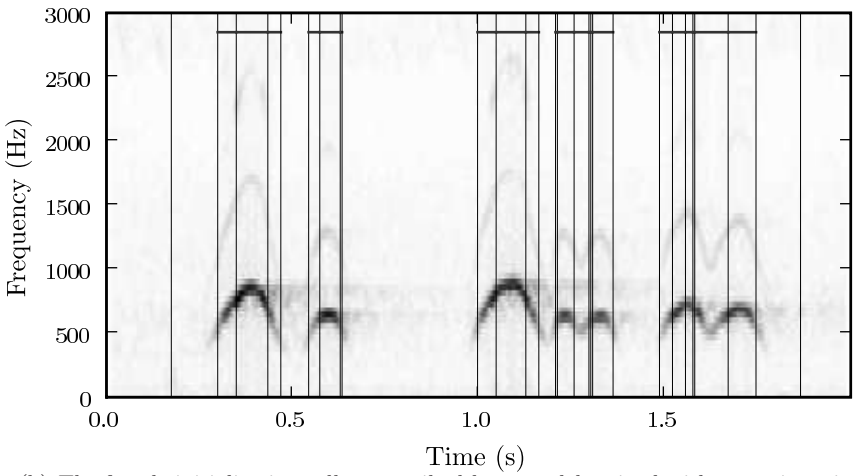


(c) A male call, transcribed by a model trained with many iterations.

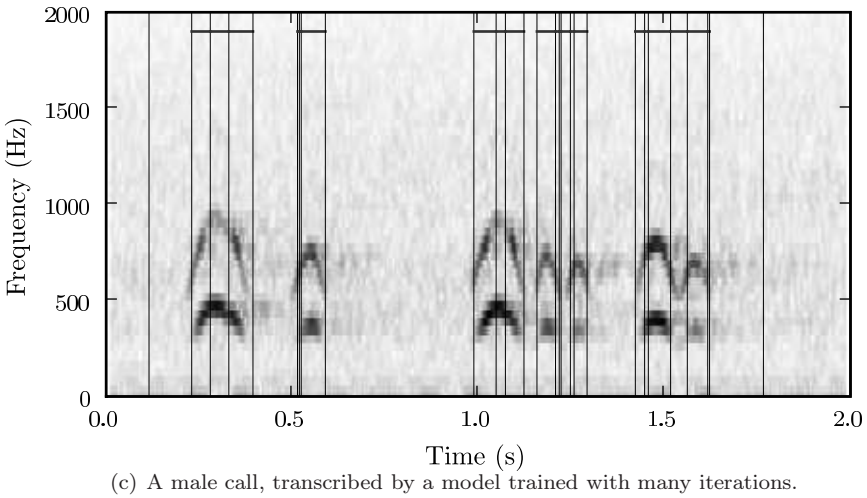
Figure 6.3: The effect of no normalisation in cases of transposition. These Viterbi transcriptions of calls from the Wood Owl training set, with a smarter left-to-right model without normalised pitch, show an extreme example of what happens when transposition is not taken into consideration.



(a) The female initialisation call, transcribed by a model trained with one iteration.



(b) The female initialisation call, transcribed by a model trained with many iterations.



(c) A male call, transcribed by a model trained with many iterations.

Figure 6.4: The effect of normalisation in cases of transposition. These Viterbi transcriptions of the Wood Owl, with a smarter left-to-right model and normalised pitch, shows that normalisation provides a potentially useful solution to the transposition problem.

	Best Unfrozen Thus-far	Enhanced Silence		
Training Iterations	4	1	4	10
Looped	19.1	14.2	12.6	11.8
Split Looped	23.3	16.5	20.8	19.6
Ferguson	19.6	20.0	13.8	12.9
Split Ferguson	**26.1**	23.5	20.7	21.1

Table 6.7: The effect of enhanced silence modelling on classification accuracy.

stacks also helps. This naive implementation had duration modelling on the beginning and ending silence state as well, which means models could potentially recognise calls based on the transcriber's habits in a repetitive call.

Also shown in the same table, is the same models with frozen transition weights. This results in only the pdfs being trained, with the duration modelling determined by the transcriptions. In the greatest majority of the cases, freezing the transition weights helps recognition accuracy. The best solution appears to be frozen models with duration modelling, or in other words, hand-specified duration modelling.

6.2.6 Enhanced Silence

The models in Section 6.2.5 had a separate silence pdf in each call model. The initial and final silence selfloop states were also transformed to a Ferguson stack in the case of duration modelling, which is not ideal. We want all models to be independent of transcribing habits. They should also preferably be independent of background noise or silence.

A more advanced implementation built the models hierarchically, using a silence pdf shared over all models and initial and final silence states similar and frozen between all models. The silence model was trained on all the inter-note silences in all the micro transcriptions, as well as more sections of inter-call silence identified explicitly for this purpose. The effect of these enhancements are shown in Table 6.7. The recognition accuracy decreased, sometimes dramatically. This somewhat disappointing result seems to indicate the initial and final silence model recognised transcription idiosyncrasies or background noise. However, in the enhanced models, the initial and final silence self-loop was frozen on 0.9. Training all the other durations but not the initial and final states could either compress or extend the remaining durations, depending on whether the initial or final silence in the transcriptions is long or short.

	Normalised pitch		Absolute pitch	
ALS cost	Excluded	Included	Excluded	Included
Normal	27.5	29.6	33.0	**39.8**
With delta pitch	26.4	26.8	31.5	34.4
With delta & delta-delta pitch	26.6	28.1	32.2	32.7

Table 6.8: Feature vector investigation.

	Simple	Split
Normal	32.1	33.0
Enhanced, $\sigma = 25\% \times \mu$	33.3	40.6
Enhanced, $\sigma = 15\% \times \mu$	38.5	47.4
Enhanced, $\sigma = 10\% \times \mu$	43.4	**51.1**
Enhanced, $\sigma = 5\% \times \mu$	**45.1**	**51.0**
Enhanced, $\sigma = 2\% \times \mu$	43.4	40.8

Table 6.9: Reducing duration variance.

6.2.7 Feature Investigation

So far, the best result was with split frozen Ferguson models. Looking at only this model structure, we investigated various features, with results shown in Table 6.8. In each case, including ALS cost as a feature aided recognition accuracy. The addition of delta pitch was not useful. Delta pitch should have helped in the absolute pitch case, in order to model note shape. If it helped in the transposition cases, it was detrimental in other cases, as the overall result was negative. A likely reason for this behaviour is the high feature vector rate. The delta feature employed considered five consecutive vectors, which is only 25 ms at the 200 Hz feature vector rate.

In three out of the four delta-pitch cases, it was useful to include delta-delta pitch as well. However, the best solution so far is to use three-dimensional features (pitch, volume and ALS cost).

The spectral spread was also investigated as a feature, adding it onto all six absolute-pitch feature combinations in Table 6.8. In each case, it was detrimental to performance, therefore it was not considered any further.

6.2.8 Frozen Weights and Lower Variance Durations

Using frozen weights, enhanced silence, we switch to using absolute pitch and volume as features. Table 6.9 shows results for various duration modelling variances. The best results are obtained by using a standard deviation between 5% and 10% of the duration means μ. By making the durations more rigid, the state pdfs are less prone to specialise. Making the durations too rigid though (for example, 2%) starts reducing the recognition accuracy again, specialising

ALS Cost	Excluded	Included
Enhanced, $\sigma = 15\% \times \mu$	47.4	46.0
Enhanced, $\sigma = 10\% \times \mu$	**51.1**	48.1
Enhanced, $\sigma = 5\% \times \mu$	**51.0**	49.7
Enhanced, $\sigma = 2\% \times \mu$	40.8	42.9

Table 6.10: The effect of ALS Cost on Frozen Split Ferguson models.

	Normalised pitch		Absolute pitch	
	Simple	Split	Simple	Split
Frozen Ferguson	22.7	27.5	32.1	33.0
CSDM, 10 SPS	17.5	25.3	26.5	32.5
CSDM, 20 SPS	20.8	28.0	28.6	36.7
CSDM, 30 SPS	22.0	29.4	33.0	38.3
CSDM, 40 SPS	25.3	30.4	35.0	40.6
CSDM, 50 SPS	26.2	32.3	37.3	39.5
CSDM, 60 SPS	27.8	33.5	38.5	40.5
CSDM, 70 SPS	29.3	34.0	40.5	41.3
CSDM, 80 SPS	30.3	33.7	41.2	**43.7**
CSDM, 90 SPS	30.1	**34.4**	40.7	42.6
CSDM, 100 SPS	**31.9**	32.6	**42.4**	41.4

Table 6.11: A comparison of Ferguson stacks and CSDM. The Ferguson models used were the four-iteration frozen models, which had the best results so far.

along the time axes instead of in the feature space.

Table 6.10 shows that including ALS cost as an extra feature dimension is not necessarily beneficial. Again, with more training data, these statistics could be different.

6.2.9 Compounded Selfloops Duration Modelling

With CSDM, the variance of each CSDM structure is dependent upon the number of states and the mean duration of the structure. This means in training iterations of a CSDM-based model the mean length and the state pdfs are trained, but the variance is not. This is effectively a compromise between frozen weights and unfrozen weights.

The results for ten training iterations are shown in Table 6.11 for CSDM structures with a specified number of states per second for each transcription unit. Compared to the best models we have so far, specifically four-iteration frozen Fergusons, it appears the CSDM structures are very promising. What we see is that training the mean is advantageous if we keep the variance small enough. In this setup, a lower variance clearly improves the model and more training iterations do not seem disadvantageous.

	Normalised pitch		Absolute pitch	
ALS Cost	Excluded	Included	Excluded	Included
CSDM, 60 SPS, Simple	27.8	24.7	38.5	42.1
CSDM, 80 SPS, Simple	30.3	27.0	41.2	44.2
CSDM, 100 SPS, Simple	31.9	28.9	42.4	44.4
CSDM, 60 SPS, Split	33.5	25.8	40.5	45.1
CSDM, 80 SPS, Split	**33.7**	31.2	43.7	**47.9**
CSDM, 100 SPS, Split	32.6	32.4	41.4	44.3

Table 6.12: CSDM models with ALS Cost.

	Top 1	Top 2	Top 3	Top 4	Top 5
Without ALS cost	56.1	62.3	65.7	71.5	**73.6**
With ALS cost	52.2	61.1	66.2	68.3	71.0

Table 6.13: N-th Best statistics for enhanced frozen split Ferugson with $\sigma = 10\% \times \mu$.

Split models typically perform better than simple models, and using absolute pitch also increases the discriminatory power of the models.

As explained in Section 2.7.3.3, calculating a Viterbi path for the CSDM models will likely not be reliable.

Table 6.12 shows the effect of including ALS cost in these CSDM models. Curiously, it is beneficial to include this feature when dealing with absolute pitch, but not when dealing with normalised pitch.

6.2.10 N-th Best

A classifier used for commercial purposes can display the top few scores to increase the usefulness in cases where the correct match is not the best. Table 6.13 shows such results for the enhanced-silence frozen split Ferugson models with $\sigma = 10\% \times \mu$. The 56.1% is higher than the 51.1% in Table 6.9 because these are unweighted scores (see Section 5.3.4). A higher score indicates that the models with many examples in the evaluation dataset (Gillard) classified better than average. In about two thirds of the cases, the correct call was one of the top three results.

6.2.11 One More Split

The number of sections notes were split into were decided intuitively. One last experiment is to see what the effect is of increasing the number of splits of every note by one. The results are shown in Table 6.14 . These results indicate that the intuitive choice made in the micro transcriptions are relatively good. A general increase of the number of sections over all notes is not beneficial. It

	Split		Split +1	
ALS Cost	Excluded	Included	Excluded	Included
CSDM, 60 SPS	40.5	45.1	40.3	40.7
CSDM, 80 SPS	43.7	**47.9**	39.7	44.9
CSDM, 100 SPS	41.4	44.3	40.3	46.1

Table 6.14: Results of splitting notes into one additional subsection.

is still likely that some individual birds or individual notes do benefit from an additional subsection though.

6.2.12 The Impact of No Development Set

When lacking a development set, many iterations of experiments can result in the leaking of knowledge of the testing data into the models. Through this chapter, we tried to place more emphasis on general trends, rather than simply select the single best-performing set of models. This was done in an attempt to avoid choosing the single experiment that gives the best results for the available testing set.

In studies with more available data, most of these experiments would be run on the development set. The testing set can then remain untouched for the purposes of making a final evaluation of the most promising models, providing more meaningful final results.

Chapter 7

Recommendations and Conclusions

7.1 Recommendations for Further Study

7.1.1 Features

With enough training data to avoid specialisation, more features could be added to model the timbre. This should aid identification of birds such as the Hadeda Ibis or the Coqui Francolin. Alternatives to ALS should also be investigated for dealing with birds with timbres that ALS cannot handle. Also, if a spectrum-based pitch tracking algorithm is used, extracting the energy in the second and third harmonics becomes less expensive.

Using signal energy rather than amplitude may be beneficial. Also, clipping or some other normalisation of the volume might be useful if there is not enough data to train the pdfs thoroughly. It may possibly be useful to score the volume or energy in the log domain, and to test deltas of such features as well.

While absolute pitch is an important characteristic of birds, it is not ideal for birds with more variance in their average pitch level. With enough training data, again, the delta features should aid in modelling the note shapes and avoid the pitfalls associated with dealing only in absolute pitch.

This study did not experiment with different decimation rates. The feature rate of 200 Hz was chosen based on the length of short notes. Using a higher feature rate could be advantageous with adequate duration modelling, but would require more computational power. Lower feature rates would be adequate for birds that do not have short notes. An interesting compromise might be to have different feature rates for different models, depending on which bird is being classified, with the appropriate normalisation.

7.1.2 Models

HMMs have the advantage of flexibility in topology. Using hierarchical HMMs, individual syllables can be modelled and built into a macro structure describing the characteristics of the variations in passerine birds' tunes. These macro structures can be designed by a human with expert knowledge, or trained if significantly more data is available. Improved macro structures would also allow skipping optional notes or sections of calls, for example the fifth note in the African Fish Eagle's call.

This study did not investigate using more subdivisions of notes in the split models. The subdivisions were simply chosen intuitively by a human. It is likely that more subdivisions will be beneficial when used with low variance duration modelling.

A macro structure that is more aware of the number of repetitions could aid in distinguishing birds with few repetitions of their calls, from those with many. In this study, any bird with a clearly repeating structure, even with as few as three repetitions, was modelled as a repetitive bird, with the number of repetitions geometrically distributed.

Developing a spotter which can identify a song within a long continuous recording could make further experiments easier. A diverse set of birds can be concatenated into one long recording and fed to a spotter model recognising a single bird. This would allow the development and testing of a single bird call model individually. In such an application, an equal error rate can be used as a measure of success, similar to the speaker verification application in human speech recognition.

In the case of repetitive calls, recognition occurred on call pairs. The looped models can handle any number of repetitions. By taking a certain amount of time for each call, short repetitive calls will have more data to recognise on, possibly increasing the recognition accuracy. With enough training data, whole sets of repetitions can be recognised.

7.2 Conclusions

7.2.1 CSDM

Compounded Selfloops Duration Modelling shows great promise in modelling normally distributed durations. If the durations are quite long and require more than a couple of dozen states to model with a Ferguson stack, CSDM could be a particularly useful and computationally efficient alternative, as long as a Viterbi path does not have to be calculated.

7.2.2 Pitch Tracking and Feature Synthesis

This study demonstrated that the ALS pitch tracker adapts well to tracking the pitch of most bird songs. While another pitch tracking algorithm may be more successful for some birds with a pronounced timbre that lacks a clear fundamental, the majority of bird song is harmonically simple enough.

This study further demonstrated that signal synthesis from features is a useful technique to gain an understanding of the effects of a particular set of features.

7.2.3 Approximate String Matching

Approximate String Matching, or string-to-string edit distance, can be applied to more intricate problems, but the rigidity of the technique remains a liability.

7.2.4 HMMs and Duration Modelling

Good duration modelling in pattern recognition is of great importance when the signal has significant temporal characteristics. With an adequate HMM topology, transition weights, and duration modelling, significant success can be had with bird song recognition, using only pitch and volume as features. Using low-variance duration modelling also helps avoid model specialisation.

7.2.4.1 Note Shapes and Split Notes

One of the standard ways of modelling feature trajectories is using delta features. In this study, the delta features investigated were not beneficial, most likely due to too high a feature vector rate. By splitting notes into subsections, each subsection could be individually modelled, providing an alternative mechanism for modelling note shapes. If more data is available and the delta is taken over a larger sample of vectors, it might still prove useful.

7.2.4.2 Frozen Weights

Freezing the training weights was beneficial. This indicates that the training of note durations was not advantageous to recognition accuracy. This is very likely due to the extremely limited training data. With sufficient training data, duration training could be beneficial. However, maintaining a low-variance during training could aid in avoiding specialisation, especially when dealing with such simple features.

7.2.4.3 Call Structure

By modelling calls with HMMs, it is quite easy to incorporate human knowledge of bird call structure. While this study only drew on human expertise to suggest the number of splits in each note and where repetition occurs, the inclusion of skip-links for optional notes is very simple. Human knowledge could also be used to specify a higher level structure in a hierarchical HMM. Being able to easily draw on human expert knowledge makes this modelling technique particularly valuable.

7.3 Future Outlook

Bird song recognition research is still in its infancy. With the well-structured calls of some birds, further research in bird song recognition should soon enable research in ornithology, neurobiology (song learning) and conservation that was prohibitively labour intensive until now.

This study also contributed CSDM, which may have applications in other pattern recognition tasks. If use of the Viterbi algorithm is not required, CSDM can be a more efficient alternative to Ferguson stacks for durations with a normal distribution.

List of References

[1] Anderson, S. E., Dave, A. S., and Margoliash, D., "Template-based automatic recognition of birdsong syllables from continuous recordings." *Journal of the Acoustical Society of America*, 1996. 5, 17, 39

[2] Bagshaw, P., Hiller, S., and Jack, M., "Enhanced pitch tracking and the processing of F0 contours for computer." *Proc. Eurospeech*, 1993, Vol. 2, pp. 1003–1006. 48

[3] Bellman, R., *Dynamic Programming*. Princeton Univ Pr, June 1957. 14

[4] Bridle, J., Brown, M., and Chamberlain, R., "An algorithm for connected word recognition." in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7, pp. 899–902, 1982. 39

[5] Doupe, A. J. and Kuhl, P. K., "Birdsong and Human Speech: Common Themes and Mechanisms." *Annual Review of Neuroscience*, March 1999, Vol. 22, pp. 567–631. 3

[6] Ehrlich, P. R., Dobkin, D. S., and Wheye, D., "Bird Voices." 1988. `http://www.stanford.edu/group/stanfordbirds/text/essays/Bird_Voices.html`. 1

[7] Ehrlich, P. R., Dobkin, D. S., and Wheye, D., "Vocal Development." 1988. `http://www.stanford.edu/group/stanfordbirds/text/essays/Vocal_Development.html`. 2, 3

[8] Fei Sha, "Matlab code implementing the als pitch tracking algorithm." `http://www.cs.berkeley.edu/~feisha/codes/track_f0.m`. 13

[9] Ferguson, J. D., "Variable duration models for speech." *Proc. Symposium on the Application of Hidden Markov Models to Text and Speech*, 1980, pp. 143–179. 30

[10] Gibbon, G., "Southern African Bird Sounds." Set of six tapes, 1991. Southern African Birding cc. 45, 59

[11] Gillard, L., "The Beginner's Guide to Bird Calls of Southern Africa." Audio CD, 1998. GBC Sound. 45

[12] Harma, A., "Automatic identification of bird species based on sinusoidal modeling of syllables." in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.*, vol. 5, pp. 545–548, 2003. 5, 41

[13] Harma, A. and Somervuo, P., "Classification of the harmonic structure in bird vocalization." in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.*, vol. 5, pp. 701–704, 2004. 5, 41

[14] Johnson, M. T., "Capacity and complexity of HMM duration modeling techniques." *Signal Processing Letters, IEEE*, 2005, Vol. 12, No. 5, pp. 407–410. 37, 38

[15] Joubert, P. J., "Voëlsang herkenning." 2003. Final-year project report, Electrical & Electronic Engineering Department, Stellenbosch University. 6, 41, 69

[16] Kogan, J. A. and Margoliash, D., "Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden Markov models: A comparative study." *The Journal of the Acoustical Society of America*, 1998, Vol. 103, No. 4, pp. 2185–2196. 5, 40

[17] Liep, J., "Airborne kula: The appropriation of birds by Danish ornithologists." *Anthropology Today*, October 2001, Vol. 17, No. 5, pp. 10–15. 5

[18] McNab, R., Smith, L., and Witten, I., "Signal processing for melody transcription." *Proceedings of the 19th Australasian Computer Science Conference*, 1996. 41

[19] McNab, R., Smith, L., Witten, I., Henderson, C., and Cunningham, S. J., "Towards the digital music library: Tune retrieval from acoustic input." in *Dl '96: proceedings of the first ACM International Conference on Digital Libraries*, (New York, NY, USA), pp. 11–18, ACM Press, 1996. 6, 19, 41, 64, 65

[20] Mongeaul, M. and Sankoff, D., "Comparison of musical sequences." *Computers and the Humanities*, June 1990, Vol. 24, No. 3, pp. 161–175. 19

[21] Moon, T., "The expectation-maximization algorithm." *Signal Processing Magazine, IEEE*, 1996, Vol. 13, pp. 47–60. 25

[22] Nelson, D., "Developmental stages." Accessed November 2007, `http://blb.biosci.ohio-state.edu/Nelson/stages.htm`. 2

[23] Nowicki, S., "Vocal tract resonances in oscine bird sound production: Evidence from birdsongs in a helium atmosphere." *Nature*, 1987, Vol. 325, pp. 53–55. 2

[24] Rabiner, L. and Juang, B., "An introduction to hidden Markov models." *ASSP Magazine, IEEE*, 1986, Vol. 3, pp. 4– 16. 25

[25] Riede, T., Suthers, R. A., Fletcher, N. H., and Blevins, W. E., "Songbirds tune their vocal tract to the fundamental frequency of their song." *Proceedings of the National Academy of Sciences*, April 2006, Vol. 103, pp. 5543–5548. 3, 46

[26] Saul, L. K., Lee, D. D., Isbell, C. L., and Lecun, Y., "Real Time Voice Processing with Audiovisual Feedback: Toward Autonomous Agents with Perfect Pitch." in *Advances in Neural Information Processing Systems 15* (Thrun, S. and Obermayer, K. (Eds)), pp. 1181–1188, Cambridge, MA: MIT Press, 2003. 6, 10, 13, 48

[27] Schroeder, M. R., "Period Histogram and Product Spectrum: New Methods for Fundamental-Frequency Measurement." *The Journal of the Acoustical Society of America*, April 1968, Vol. 43, pp. 829–834. 48

[28] Schwardt, E., "Het Vogelsang." 2002. Final-year project report, Electrical & Electronic Engineering Department, Stellenbosch University. 45

[29] Schwardt, L., "Voice Conversion: An Investigation." March 1998. An Investigation, Master's thesis, Electrical & Electronic Engineering Department, Stellenbosch University. 48

[30] Somervuo, P. and Harma, A., "Bird song recognition based on syllable pair histograms." in *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings.*, vol. 5, pp. 825–828, 2004. 41

[31] Vintsyuk, "Element-wise recognition of continuous speech composed of words from a specified dictionary." *Cybernetics and Systems Analysis*, March 1971, Vol. 7, pp. 361–372. 39

[32] Wagner, R. A. and Fischer, M. J., "The String-to-String Correction Problem." *J. ACM*, January 1974, Vol. 21, No. 1, pp. 168–173. 19

[33] Wikipedia contributors, "Bird." December 2007. `http://en.wikipedia.org/w/index.php?title=Bird&oldid=175937285`. 1, 5

[34] Wikipedia contributors, "Bird song." December 2007. http://en.wikipedia.org/w/index.php?title=Bird_song&oldid=175984398. 2

[35] Wikipedia contributors, "Birdwatching." December 2007. http://en.wikipedia.org/w/index.php?title=Birdwatching&oldid=176017423. 5

[36] Wikipedia contributors, "Kula ring." October 2007. http://en.wikipedia.org/w/index.php?title=Kula_ring&oldid=165373369. 5

[37] Wikipedia contributors, "Passerine." November 2007. http://en.wikipedia.org/w/index.php?title=Passerine&oldid=174466785. 1

[38] Wikipedia contributors, "Songbird." November 2007. http://en.wikipedia.org/w/index.php?title=Songbird&oldid=169218434. 1