An experimental system for computer aided bird call recognition

VOLUME 1 Introduction to Conclusion

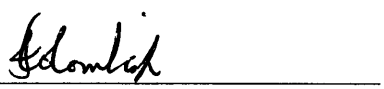
Illan Samson Colombick

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1992

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the degree of Master of Science in Engineering at the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination at any other university.



Illan Samson Colombick

28 day of 08 19 92

Acknowledgements

I would like to thank my supervisor, Dr Alastair Walker, for his guidance and constant encouragement. Mr Len Gillard was most kind and helpful in giving freely of his time, expertise and material. I gratefully acknowledge the financial assistance received from the Foundation for Research Developement, the University of the Witwatersrand and the Adolph Wagner Scholarship. Thanks to my friends and colleagues who advised me and brightened my days. Finally, I am deeply grateful to my parents Selwyn and Jocelyn Colombick, without whose boundless support and understanding this work would never have been completed.

Acknowledgements

I would like to thank my supervisor, Dr Alastair Walker, for his guidance and constant encouragement. Mr Len Gillard was most kind and helpful in giving freely of his time, expertise and material. I gratefully acknowledge the financial assistance received from the Foundation for Research Developement, the University of the Witwatersrand and the Adolph Wagner Scholarship. Thanks to my friends and colleagues who advised me and brightened my days. Finally, I am deeply grateful to my parents Selwyn and Jocelyn Colombick, without whose boundless support and understanding this work would never have been completed.

Contents

VOLUME 1

Table of Figures

1
Introduction

> **Sharp staccato trumpeting, starting off with some single notes, then speeding up into syncopated double notes, *wak, wak, wak, kawak-kawak-kawak,* etc.; clicking *xok*, growling *krrr***
>
> Description of Redbilled Hornbill call, Roberts' Birds of Southern Africa.

Birds are unique in so many ways - masters of the air, free as other wildlife has never been to move where and when the mood takes them. Free, often, from the need to hide to avoid predators, they are the only group of wild animals with which many humans come into regular contact, the only ones able to coexist with us without becoming domesticated, or lurkers in the shadows and recesses of our urban environment.

It is not surprising therefore that birds have come to inspire us, symbolising freedom and recalling the bounty of nature from which we have been separated. Birds are unique in another way; they are the most highly and variably vocal animals besides ourselves. So their special symbolic value has come to be bound up with their calls, and it is the sound of bird song in the early morning which has the most profound effect on our imaginations and on our spirits.

In Southern Africa the presence of birds is perhaps accentuated as in few other places. Due to its variety of habitats and climatic and geographic circumstance, our region is home, or temporary haven, to nearly 900 birds (many more than found in Europe or North America). The comparatively large areas of undeveloped land, the many unpolluted beaches, combined with the existence of some of the most highly developed urban regions on the continent, mean that many Southern Africans have unique opportunities for awareness of birds in their surroundings.

So it is perhaps not strange to find bird calls being studied in an engineering environment, certainly not when it is discovered that a great number of people have a strong interest in bird watching, and in using technology to aid and expand this activity. What is surprising is to find that bird vocalisations, so vital to the experience of bird watching, so important to the birds themselves in their constant need to recognize each other, and so suited to manipulation by computer technology as opposed to traditional media, have not been heretofore incorporated into such technology and made available to the public. The widespread availability of tape recordings of many Southern African birds, due to the admirable efforts of a few dedicated individuals, makes the study of the calls all the more feasible. This project represents a beginning, a first foray into the use of active processing techniques to exploit the potential of bird calls for the benefit of a not insignificant birding public.

1.1 Project overview

The primary aim of this project is to establish an experimental framework for research into automatic bird call recognition. Theoretical or empirical work in this area has not heretofore been published, so that the project is of an exploratory nature. It starts by surveying the available knowledge of bird calls in the context of species identification, and ends not only with an abstract framework for research, but a working tool based on that framework, a prototype of a recognition system. In the process many new or unforeseen issues emerge, many questions are posed and some are answered.

1.1.1 Context of the project

The question of whether automatic bird call recognition is possible arises in the context of a larger research and development initiative - a computerised *bird finder*. This is a device, similar to a hand-held calculator, which will be used to help bird watchers identify species in the field. Its capabilities are to include a textual database of species information, with the ability to rapidly search the database in a way that is tailored to the needs of a bird watcher. It can successively narrow down the possibilities in response to the user's selections of various attributes, finally homing on a short list of likely species. The user may then be presented with detailed information about the short-listed birds.

The bird finder is a model for an entire generation of hand-held computer devices which are set to enter everyday use in wide range of fields, from the household to the specialised engineering work environment. As such it is planned with a view to sophisticated operation. It will eventually incorporate graphical and audio capabilities - useful in many areas and certainly beneficial in that of bird identification. The pictures and sounds of birds together convey an overwhelming proportion

of the information available for identification in the field. The choice of bird identification as the development vehicle for the hand-held device is due to a number of factors, not least the popularity of amateur bird-watching and the exciting possibilities for a device of this kind in that market.

The possibility of audio playback in the device is especially interesting, since it would constitute the first practicable field reference for bird sounds. The sonagrams in conventional (book) field guides, and the recordings on tapes can be useful as confirmation or as a means of consigning selected calls to memory, but do not fulfil the great potential of bird calls as identification parameters in the field. The ability of a computer device to quickly locate and play back a digital recording while the observed bird is still present and calling is a great step in redressing this gap between practice and potential.

An even more desirable advance would be the ability to use the call as the primary key in a search. Thus, instead of narrowing down the list of possible calls to a manageable size, then listening to each call in turn to compare with the observed one, it would be useful to be able to record the calling bird, and have the computer use that recording as a means of narrowing down the list itself. In other words, the computer should recognize the species by its call. The ability to do this would not only make primary use of a heretofore under-utilised but important feature, but would make identification viable under previously difficult conditions. Identification of night birds and birds which remain hidden in dense foliage is one possibility. Another is the differentiation between members of co-occurring, visually similar groups of birds. Many families, such as the Cisticolas and Warblers (*Sylviidae*), and Larks (*Alaudidae*) rely greatly on calls to differentiate between species.

The present project is a first attempt at such a system, undertaken as much to unearth the issues involved as to resolve them.

1.1.2 Research issues

Some of the issues which have guided the progress of the research and influenced its direction are worth mentioning here. Because this is a first exploration of the area, it was not expected that an optimum solution be found. Ideas were generated stimulated by available facts and supported by a minimum level of reasoning. The most promising directions were chosen for implementation, and it was felt that by doing so a base would be established from which more rigourous evaluation of ideas could be performed at a later stage.

In order to ensure some chance of success in the vacuum of previous work, and to make the operation of the system as clear as possible to future experimenters, a major guiding principle has been to match operation to the intuitive behaviour of a human charged with recognizing bird sounds.

Although many numerically-oriented means must exist to approach the problem, it was the stages through which a person's thought processes is estimated to pass that inspired the stages in the computer system. Rather than looking at the bird call as a sequence of arbitrary numbers in which patterns need to be detected by statistical means, the regularities and significant features apparent to a human observer guided the way the data were used in the system.

An important factor borne in mind is that eventually, a system would have to operate under real-world conditions, on hardware that would need to be small and cheap. Speed and memory space requirements would therefore have to be taken into account, if not adhered to in the first prototype. These considerations influenced the choice between alternatives, with simpler options usually being chosen because they generally lead to faster operation. Once a system is in operation, the question of whether its simplicity is leading to inadequate behaviour could be addressed, but at least an idea of the upper bound for speed of the system would result from the experiment.

One of the most important questions to arise in the process of mapping the terrain of bird call recognition was that of the rôle of *syntax*. As is explained in the chapters to come, the bird call appears to have two distinct components which, although partially interdependent, seem to contribute separately to its recognisability. The first is the spectral nature (frequency breakdown) of the utterances present in the call, which determines the sounds' "quality", such as hissing, crying or whistling. The second component is the syntax - the rules or patterns according to which individual utterances are strung together to form the call.

The question is whether both components contribute equally to the distinctive nature of a call. To recognize spectral features, time-consuming sophisticated numerical matching techniques are needed. Syntax, on the other hand, is closely related to symbolic manipulation, and is therefore easily and quickly handled by a computer program. If it were possible to recognize a call based on syntax alone, the result would be a very fast recognition system indeed. Because the syntactic component is however a higher level feature than the spectral component, it is necessary to perform some spectral recognition before the syntax can be dealt with. An overall trend in this project was to make decisions that would lead to the ability to do syntactic recognition with as little spectral processing as possible. In this way, the interesting hypothesis that the syntax can help recognize most calls without too much recourse to spectral features would be testable, and the resulting system would again indicate a speed upper bound rather than being inconclusive regarding speed requirements of a working system.

That the hypothesis mentioned above is not actually proved or disproved as part of this project is due only to limits on the project's scope. The recognition system produced is completely capable of testing the hypothesis, and there is further discussion of the subject in the concluding chapter.

Another issue which is of importance in determining the course of the research is that of bird call quantification. Naturally a computer system has to deal with well-defined quantities. The project therefore has to concern itself with methods of effective quantification of the various features of bird calls. Due to the "humanising" trend discussed earlier, there was some hope that the quantification would be of interest to researchers of bird sounds in general, and not restricted to a convenient numerical representation for a particular recognition scheme. Although not treated as an explicit goal, the result is that the prototype system does embody a means for quantifying and formally describing many key aspects of bird calls, and as such may be a useful basis for others wishing to expand on the methods.

Finally a research issue that was not intentionally addressed, but which has great practical importance, arose of itself during the course of the quantification process mentioned above. It is the issue of compression of bird call data for storage and subsequent playback purposes. A simple but highly effective method is derived and tested for reducing the memory space required to store a bird call, so that it can be reconstructed and played back without undue loss of quality. This is of importance if the highly desirable goal of allowing playback of calls on a hand-held device is to be realised.

1.1.3 Software design issues

The design of the software for the project needed special consideration because of its experimental character. It was not to be an integrated, finished product accessed only by an end user, but a set of tools for on-going experimentation. Its operation had to be easily grasped, because changes would need to be easy to implement, both during the course of the project, and by future researchers.

The structure which was used to achieve the above requirements was one of separately executing modules, arranged in a data chain or pipeline, so that the output of each module is the input to the next one. Since the modules execute independently of one another, the data (input or output) resides on file. This means that all intermediate stages of processing are available for examination at all times. The modules themselves are usually small, each performing a simple, self contained task. They are thus easy to understand and modify. To process a call for recognition, the modules may be run manually one by one, or simple controlling programs may be written to run more than one module in sequence.

The final module in the pipeline is the syntax processing module and is a larger and more complex piece of software than the other pipeline processes. It has therefore been augmented with a debugging facility, which allows the step by step operation of the software to be viewed, leading to greater confidence in its correct operation.

In addition to the primary software being tailored to experimentation, certain additional sub-systems were created to facilitate the management and execution of experiments and the interpretation of results.

Primary amongst these is the Bird Call Descriptor (BCD) system, which is a controlled data sharing mechanism for communication between processes, as well as being a means for managing experiments. Pipeline processes access the BCD in order to get input parameters determining their behaviour. These parameters can be placed in the BCD by the experimenter or by other processes. Access to the various parameters is restricted, the experimenter must specify which processes may see or alter which parameters. The BCD is a structured text file which can be understood by the experimenter. The BCD mechanism allows (almost forces) the experimenter to maintain an up-to-date record of the experiments performed, so output data from any stage of any experiment can be regenerated without the data itself having to be kept on file.

Another essential sub-system is the data viewing program. This allows the output of pipeline stages to be viewed on screen in graphical form, printed, compared and in some cases manipulated. Other sub-systems include utilities for specialised manipulation or output of the stages, and project management utilities.

1.2 Document overview

1.2.1 Finding information of interest

The document consists of four different type of information: Background, descriptive, technical and reflective:

Background information is to be found mostly in chapters 2 and 3. Chapter 2, Background, covers bird calls, including production mechanisms and acoustic characteristics. Human speech is covered next, with comparisons to bird calls, and mention of speech recognition techniques. Then follows a discussion of more general pattern recognition concepts, ending with the subject of Syntactic Pattern Recognition, which is found to be well suited to handle bird calls.

Chapter 3, Computer-based bird call analysis, gives an overview of basic signal processing concepts, concentrating on those principles needed to understand the processing that is required for bird call analysis. It starts with a general explanation of the need for and basics of frequency domain analysis, and leads on to more specific Fourier analysis techniques, concluding with a detailed specification of the kind of Fast Fourier Transform used in the project.

Descriptive information is that which covers the new ideas that make up the Bird Call Recognition (BCR) system developed in this project. The system is described and assumptions and hypotheses introduced and justified. The BCR system is built up in successive stages, and some material in Chapter 2 is descriptive of the first stage of its operation - the spectral analysis. Chapters 4, 5 and 6 contain the bulk of the descriptive information, each chapter concerned with successive stages of BCR system operation.

Chapter 4, Extracting key information from the call, shows how the spectral information required from a bird call can be reduced, gives an algorithm for reducing it, covers limitations and benefits of the algorithm, and finally details a subsidiary use of the algorithm: Producing (playing back) good quality calls from recorded data that has been greatly reduced in size.

Chapter 5, Classifying bird sound primitives, introduces some ideas on what the important features of bird sound might be, from a recognition perspective, and describes how these features are found by algorithmic means. The features are concerned with the tonal quality of the sound and the separation of the call into discrete auditory elements.

Chapter 6, Syntax matching system, describes the part of the BCR system that does the actual recognizing. It starts with a definition of the syntax specification, which is an abstract format for all the calls of a species. Then the algorithm for comparing a syntax specification with a string of elements produced from a call is detailed. The complexities of the algorithm are explored, and the chapter ends by presenting the results of a set of trials, in which the BCR system capability is apparent for the first time.

Technical information is mostly that concerned with details of implementation of the BCR system software. Chapters 3 to 6 have corresponding sections in Appendix A where the software connected with each chapter is described. Appendix B has the source code listings for the software. Other appendices deal with practical steps in running the software, a quick aid to reading the sonagram (a graphical bird call representation) and a reference containing output from all the processed stages of all the calls used in the project.

Reflective material - interpreting results, speculating on alternatives, is mostly to be found in the conclusion, Chapter 7, where the BCR system is evaluated at different levels and ideas for future work are outlined. Some of the descriptive chapters also have a small reflective component.

1.2.2 Note on reading the document

The bulk of this document may give it a somewhat unfriendly appearance. Please note that most of the volume is contained in the extensive appendices, which will be of interest mainly to those wishing to reproduce work described here. However the main section is also large. There are two reasons for its size:

Firstly, the subject matter is of an exploratory nature and traverses territory which is does not necessarily have a well established theoretical basis. The problem is not one which can be clearly and tersely defined in mathematical terms. It cannot be approached by searching for theoretical techniques in the literature, applying the techniques to a comprehensive and well established base of input data, and analysing this application to obtain a conclusion. Rather, a less than formal approach had to be adopted, with techniques being fashioned by common sense and practical need, rather than derived from theory. Ideas are evaluated as much in terms of their value for future research as for their conclusive applicability in the light of current results. In other words, it is often not possible to quickly prove propositions as they have not been formulated within a well-established theoretical context, and wordier and wider ranging justifications have had to take the place of such proofs.

The second factor contributing to the size of the document is the number of different areas which the project encompasses: Signal processing, pattern recognition, ornithology, software development. It was felt that a reader's familiarity with all the terms and issues encountered could not necessarily be assumed. Thus more space has been devoted to background material and hopefully not-too-highly-technical treatments than would otherwise perhaps have been required. At the same time, a detailed and authoritative dissertation in any or all of the areas covered was not feasible and the document can not claim to serve as a reference for this subject matter. One area in which no explanation or background is given is that of computer programming, in particular the use of Pascal, especially Turbo Pascal 6.0, including object-oriented features.

The reader is urged to skip over background material which may be familiar. In particular, chapters 2 and 3 contain such material. Although some of the context of the bird call recognition system is introduced concurrently with this background matter, it is hoped that, through reiteration in other places, the missing areas can be filled in.

To further aid the selective reading of the document, the chapters describing the various practical stages in the construction of the BCR system have been structured in a loosely uniform manner. Background and introductory discussions are followed by descriptions and justifications of methods and algorithms used, together with problems, limitations and successes of these methods. Specific technical details of the workings of the software implementing the algorithms for each

chapter are provided in a separate section in an appendix. These technical discussions are aimed at those wishing to understand fully the software listings provided in the appendices, and may be safely skipped when specific practical details are not of interest.

Finally the concluding chapter contains a substantial summarising element, so that the conclusions presented there may be grasped without too heavy a reliance on the preceding material. It is recommended that this chapter be used as a base from which to proceed to the detailed chapters, if a complete sequential reading of the document is not to be undertaken.

2
Background

This chapter reviews some of the basic facts which have relevance to the problem of automatic bird call recognition. Most important is the survey of available material on the acoustic behaviour of birds. The sources for this area of study number relatively few, indicative of the fact that it is a young field.

Beginning in 1950 with the work of Thorpe [8] and Marler, the field began to grow after publication of Hinde's *Bird Vocalizations* [9] in 1969. This growth led to the compilation of papers in [7], which is the main body of material available. Other references used, Jellis [10] and Catchpole [11], were first published in 1977 and 1979 respectively. A further indication of the late blooming of scientific interest in bird calls is given by Roberts [2], the authoritative South African reference. Only its most recent edition, that of 1985, includes objective bird call information, in the form of sonagrams[1].

In contrast to bird vocalizations, human speech has been studied since before the advent of the computer, and has stimulated and in turn been stimulated by the advances in pattern analysis and recognition. Today a large body of material exists in both these fields, and the depth of coverage in the context of this survey is necessarily limited. An introductory section on speech production serves to highlight the similarities and differences between human and avian sounds. Recognition methods, as applied both to speech and in general, are reviewed in order to ascertain their applicability to bird calls. Although the concepts are covered at a superficial level, they are extremely helpful as guides to those approaches which will be useful and those which will not.

2.1 Bird calls

2.1.1 Sound production mechanisms and their implications

The manner in which sound is produced in birds is important because of the influence this has on the characteristics of the sounds. The fundamental mechanisms of bird sound production are similar

1 Sonagrams are graphical representations of sounds, used widely to record signals whose frequency content varies with time, such as speech and bird calls. Appendix E, Sonagrams, contains a section on how to interpret these graphs.

only in a general sense to those of human speech production. The differences in the mechanisms will signify different solutions to the problem of automatic recognition. Unless otherwise noted, the facts and ideas reviewed have their source in [12].

The sound production organ in birds is the *syrinx*. The syrinx is located at the junction of the bronchi and the trachea, and is thus situated at the base of the trachea. This is in direct contrast to the human larynx, which lies at the top of the tracheal tube. The syrinx owes its sound-producing capability to the *tympaniform membranes* which line the passage through which air can pass (see Figure 2.1) The membranes are surrounded completely by air because of the existence of an *air sac*, a feature of the respiratory system unique to birds.

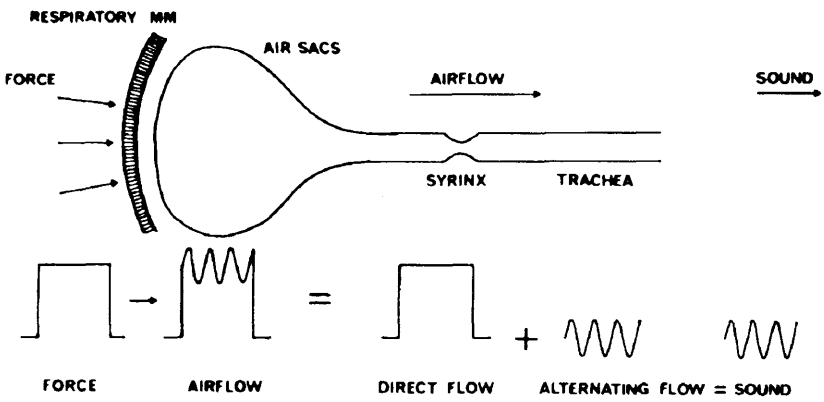


Figure 2.1 Schematic of sound production in the avian respiratory system. The membranes of the syrinx vibrate, inducing modulations in the airflow. (After Brackenbury, [12])

Two mechanisms for sound production have been proposed. The first relies on the interplay between the forces on the membrane to maintain an oscillation of the membranes. The flow of air past the membranes causes a suction force inwards, since there is moving air on the one side of the membrane which has a lower pressure than the still air on the other side. The membranes stretch inwards, which impedes the air flow, reducing the suction force. As the membranes relax outwards, the air flow can increase and suck them in again. Thus the membranes and the amount of air flowing oscillate, producing an audible tone.

The second mechanism attributes the source of the sound to the vortices formed at the membranes due to the constriction in the air passage. The air flow is disturbed by the protruding membranes and becomes turbulent. Vortices are formed and 'shed' from the stream of air at constant rate, and the oscillations may then cause the membranes may to vibrate in sympathy. This is a "whistling"

mechanism which would account for purer tones than the first mechanism. Due to the complex interaction possible between the two, the dominance of one mechanism over the other has not been established.

Referring to Figure 2.1, the sound production apparatus of a bird can be divided into two functional components, a *source* and a *filter*. The source is the vibrating tympaniform membrane. The filter is the combination of the bronchi, trachea and air sac. The source creates the basic sound and the filter modifies that sound, by suppressing some of its frequencies. In this sense the sound production mechanism is similar to that of humans or other mammals (see the later section on speech in this chapter). The human source is the vibrating vocal cord in the larynx, and the filter is the vocal tract - largely the throat and the nasal and oral cavities.

There is an important difference between the two mechanisms, however. A human vocal tract has a highly variable shape, due to the movement of the tongue and lips and the muscles in the throat. A bird's tract is more like a rigid tube - only the length of the trachea, and in some cases the air sacs, can be slightly varied. This means that the *spectral characteristics*, that is, the proportions in which different frequencies combine to make up the full sound, are much less variable in birds than in mammals [13]. The explanation for this is given next, but it is the observation itself which is of significance when considering recognition systems to be used, as will be emphasised in later chapters.

The spectral characteristics of the sounds produced are influenced by three factors. The first is the source behaviour - the vibration of the tympaniform membranes may be in a series of harmonics (i.e. it may not be a pure sinusoidal vibration). In that case, the vibration has a fundamental or base frequency component, and other components are present at twice the fundamental frequency, at three times the fundamental and so on. The second factor is the behaviour of the trachea and bronchi as a filter or resonating tube. The tube will allow certain frequencies (which originate in the source) to pass "through" while suppressing others. The ones which pass are those at the tube's *resonant frequencies*, which are related to each other. For an open tube, the resonances occur at some frequency F, at 2F, 3F and so on. F is not in any way related to the source - it is a characteristic of the tube dimensions. For a tube closed at one end, the resonances are at F, 3F, 5F and so on.

The third factor influencing bird sound characteristics is the direct amplitude and frequency modulation of the source by muscular activity. The syrinx may have up to seven muscles associated with it. These muscles can be used to alter the frequency of the source by varying the tension in the membranes. At the same time, the amplitude of the sound changes since the different membrane configurations mean different amounts of air flowing in the tract. The amplitude can also be made

to change on its own (without an associated frequency change) if a different muscular mechanism is used, in which the external labium (part of the bronchial wall) is intruded into the bronchial passage.

The first two factors on their own give rise to a large number of the sounds typical of birds, those in which a single frequency or a small number of harmonically related frequency components (i.e. fundamental and integral multiples of it) are present. These factors give the sound its particular quality or tone. The third factor is responsible for the change in pitch and volume of the sounds. It is used to form the notes of the "song".

In some cases, however, when the frequency and/or amplitude modulation happens at a rapid rate, it can be perceived as changing the quality of the sound itself. For instance a fast modulation from low to high frequency would be perceived as a single click rather than a rapidly changing whistle. A series of such clicks would be perceived as a buzz or rattle, a low frequency impure tone. Spectrally, one would observe additional frequency components on either side of the main component (the component which is being rapidly varied). These side components would not be related harmonically to the main one (which accounts for their unmusical quality). The highest rate at which a bird can produce this sort of rapid modulation is of the order of 70 Hz. This sort of modulation is limited by muscular and neural capabilities, but higher rates may be possible in complex interactions between resonating portions of the tract and muscular activity.

The important points resulting from the discussion above (and also some of that on speech, which follows) are:

- Except for clicking, buzzing, hissing or other noisy sounds, the spectral content of bird vocalizations is mainly a single frequency or a set of a few harmonically related frequencies.
- The main variation in the bird sound is that of the pitch and/or amplitude; the relationship between frequencies remains more or less fixed.
- By contrast, human speech contains many closely spaced harmonics which are then filtered by the vocal tract resonances, so that a small number of frequencies are dominant in the spectrum, but these are in general not clearly related to each other, and their relationships change during the course of the utterance.

Knowledge of sound production in birds is by no means complete. A large number of questions about the detailed functions of organs are unanswered. A little is known about a small number of species, and many species have never been the subject of research. For instance, the ability of some birds to produce three or four independent "voices" (frequencies) is not well understood. An example of such a bird is the Indian Myna (*Acridotheres tristis*), which uses this capability when imitating human speech ([8], p.117).

2.1.2 Acoustic properties

The acoustic properties of bird calls are important because they help determine the signal processing requirements of a bird call recognition system. This section reports on the properties of those calls which are likely to be useful in identifying species.

The bird sounds of interest to us are those which carry over long distances. Birds produce sounds specifically designed to carry distances of 50 to 200 metres [14]), intended for communication of territorial and species information. They also produce much quieter sounds for communication between close individuals, but these are rarely audible in the field and are not as suitable for species identification.

The evolutionary pressure to broadcast information over long distances has influenced the sounds birds make. The environment acts to attenuate sound in a number of ways. The best way to overcome attenuation with distance, whether by scattering or absorption, due to the atmosphere or vegetation, is to decrease the frequency of the sound [14].

Birds can be roughly divided for vocal behaviour purposes into two groups, the *passerines* and the *non-passerines*. Passerines, meaning "perching birds" are in general small (the largest passerine is the Crow) and consist mainly of the oscines, or true songbirds. Non-passerines are in general larger and less capable of producing varied sounds [10]. Because of size differences, it is not surprising to find that the calls of Non-passerines are in general lower in frequency than those of passerines (larger filtering apparatus means larger resonant wavelengths and more massive moving parts in the source mean slower movements, both of which result in lower frequencies). The passerines which are able to produce low frequency sounds are often those which live under especially arduous conditions of sound attenuation, such as tropical bird species which call from or near the ground. Acoustic propagation near the ground is subject to low pass filtering which renders the frequencies above 1-2 kHz practically useless [14].

A factor other than attenuation influencing the characteristics of long range acoustic communication is the quality degradation which occurs with distance. The combined effect of reverberation, due to multiple reflections and scattering, and random amplitude fluctuations, due to atmospheric turbulence, serves to degrade amplitude modulated sounds. Frequency modulations are far less affected, and in addition the use of different frequencies to encode information improves the signal to noise ratio. Thus sounds using different frequencies, either simultaneously by harmonic and resonance methods, or by varying the frequency with time, are ideal for long range communication, and are prevalent among birds [14].

Some conclusion about the relative importance of bird call properties can be drawn from a study of the auditory capabilities of birds. There is a good correspondence between hearing sensitivity and the spectral characteristics of vocal output for four species reviewed in [1]. The study shows plots of hearing sensitivity which indicate that the attenuation of frequencies above about 8 kHz is so high that it is reasonable to assume that bird calls will not contain significant information in that range. A perusal of sonagram reproductions of a huge variety of bird calls in a comprehensive field guide, Roberts [2], confirms this. Sensitivity decreases below about 2 kHz, but less rapidly than above it. Except for some nocturnal predators such as owls, and Rock Doves which may use low frequency propagational advantages in homing, birds in general have lower sensitivity to all frequencies than man, and a useful definition for low frequency cut-off is about 500 Hz [1].

The frequency discrimination (ability to detect a change in frequency) of avian auditory systems has been found to be about 1-2 percent in the range 1-4 kHz, and sharply degraded thereafter. This means a resolution of 10-20 Hz at 1 kHz and 40-80 Hz at 4 kHz [1]. The figures give an indication of the capabilities required by a signal processing system that deals with bird calls. The resolution here is similar to that of ordinary spectrographic analysis.

The temporal discrimination of sounds by birds is the other important feature in determining signal processing system requirements. Data for birds indicate a duration resolving power of about 2-3 ms in a signal of 20 ms duration. This rivals or surpasses the capability of a spectrograph (device for making sonagrams, which are also called spectrograms) [1].

2.1.3 Species-specific characteristics

It is necessary to determine what sounds and what aspects of bird sounds are useful for identifying the species. These aspects, or parameters as they are often called, might vary between species. But the physical limitations on sound production, propagation and reception, as discussed in the previous sections, make it likely that certain parameters will be commonly used as species identifiers [3].

The function of bird sounds is the subject of much research. It is clear that there are many such functions. The purpose of a particular bird call can often be inferred by observing the circumstances in which it is uttered, the behaviour of the vocalising bird, the reaction of other individuals of like and unlike species, whatever their relation to the source bird, and by laboratory and field experimentation. In this way, sounds uttered at the approach of a predator, which cause a flight reaction in all other birds of similar size to the caller, may be classified as Alarm or Flight calls. Other categories of bird sounds are, briefly:

Anxiety, aggression, contact and other socialising calls, calls associated with food, individual identification, courtship, and territorial calls and songs.

Not all of the above can be used for species identification. Alarm, anxiety and aggression towards predators often result in calls that are rather similar in characteristics amongst many different species. The reasons for this may be to allow different species to cooperate, such as in mobbing an owl, or because the characteristics of the calls have important functions such as high audibility but low direction information content, such as the pure, medium to high frequency alarm of many small species ([11], p.15-6). Contact sounds, for instance communication between parents and young, or between mates or small community groups, may not be required to give location information, and are often too quiet to hear in the field.

It is necessary here to clarify the terms "call" and "song" as applied to bird sounds. Calls are in general short, simple and stereotyped, given by both sexes throughout the year. Songs are generally long and complex, given by males in the breeding season and not generally present amongst non-passerines. Calls are easier to interpret in terms of their function than songs, which are often extremely complex and variable. Songs are not easily correlated with immediate context, often occurring regularly with season, time of day and sexual rhythms. Both calls and songs can contain species and individual information. The songs have the potential to carry more information, but that does not necessarily happen [11]. Note that except for this section, the distinction between call and song is not strictly applied in this document. Both are used to refer to bird vocalizations that may be suitable for use in identifying species.

Logically, the sounds most suited for use as species identifiers are those which are used in such a way by the birds themselves. These are the sounds which, in their function as territorial advertisers and sexual attractors are designed to broadcast species information over long distances [3]. Songs are mostly associated with this type of information. Loud calls are also useful since they are generally stereotyped and certainly must contain species information in those birds which do not sing. The complexity of song can sometimes be a disadvantage when attempting to recognize the singer, since the observer must be familiar with a larger number of song elements and combinations [11],[3].

Becker has identified a general scheme for the encoding of species-specific parameters. Quoting from [3]:

> Characteristics which function as (species-specific) parameters have enough contrast with the noises of the channel to ensure that they can reach the receiver with minimal distortion and loss of information. In most species the parameters are derived from *frequency range, syntax, element structure,* and the *lengths of*

intervals between elements. Amplitude patterns and some aspects of fine structure of the elements (Eg transients) are subject to a variety of disturbances in the channel, and appear not to be used as parameters.

An element is a continuous segment of sound as displayed by a sonagram [10]. By the structure of an element is meant the frequencies present and the manner in which these frequencies vary for the duration of the element. Italics are not in the original.

The characteristics identified above can be used as a first level approach to the problem of bird call recognition.

2.2 Speech

Human speech study is the only field comparable to that of bird vocalizations in which significant attention has been given to automatic recognition. One reason for this attention is clearly the quantum leap in man-machine communications that would be afforded by successful practical speech recognition systems. A second reason is the close correspondence between the ability to understand speech and what we term intelligence. Consider how impressed we are with animals, chiefly birds and large primates, that exhibit the ability, apparent in the former case and real in the latter, to communicate via language. The evolution of human intelligence is closely related to our vocal and linguistic capabilities, and it is the hope of many scientists in the fields of linguistics and artificial intelligence for example, that a successful speech understanding system will bring new insights into human speech and thought processes.

The material which follows serves simply to allow comparison of bird call production mechanisms and spectral characteristics with those of speech, and then to evaluate the applicability of speech recognition methods to bird calls.

2.2.1 Speech Production

The source of voiced sound in speech is the vibration of the vocal cords in the larynx. The cords vibrate in response to the flow of air forced past them by the respiratory muscles. The tension of the cords and the cross-sectional area of obstruction that they present to the airflow are adjusted to control voicing and to vary the pitch of the sound.

Sound issued from the larynx travels through the vocal tract and emerges at the lips. The vocal tract is the air space extending from the larynx in the throat to the lips and nasal chambers. Once again, the larynx is the source of the sound, while the vocal tract is the filter. The source generates a set of harmonics spaced apart by the fundamental frequency. The fundamental is what we perceive

as the pitch of the sound, and it is in general much lower than that of birds (of the order of 100 Hz, whereas birds usually produce sounds of the order of 1000 Hz). Thus the spacing between harmonics is smaller, and there are many more harmonics occurring within a given frequency range in the spectrum. The vocal tract, functioning as an acoustic tube, passes those harmonics that occur at frequencies close to its resonances, and suppresses those which do not. This effect is clearly visible in the sonagrams of Figure 2.2, where the darker bands occur at the resonances (called *formants* in linguistic terminology, and in the figure, centred around 0.8, 2.0 and 2.8 kHz) in the speech signal.

Unvoiced sounds also occur in speech. In this case the sound source is in the random turbulence of air flowing past a constriction in the tract. The spectral shape of this sound source is similar to that of random noise - rather than a set of harmonically related frequencies, a continuous range of frequencies is present, all with similar magnitude. The position of the constriction in the vocal tract determines whether the spectral characteristics will be modified by the tract - the further back in the tract, the more influence it has on the sound.

We vary the effect of the vocal tract by changing its shape, thereby changing the resonant frequencies, which are those frequencies which are emphasised. The tongue, lips, jaw, velum and throat walls are all used to accomplish significant variations in the tract shape.

The information in the speech signal is largely encoded in the variations of the vocal tract resonances. This is in contrast to bird sounds, where the equivalent tract is not significantly variable and information must be encoded chiefly in the source pitch. A clear demonstration that pitch information is of secondary importance in humans is the intelligibility of whispered speech. Whispering is purely unvoiced, meaning there is no periodicity in the source - no fundamental frequency to give the sound a specific pitch. Also there is little inherent shaping (relative difference between the magnitudes of various frequency components) in the noise-type source, making it almost information-free. The spectrum of the source is given shape purely by the filtering action of the vocal tract.

Figure 2.2 shows narrow and wideband sonagrams of a spoken syllable[1]. The narrowband analysis shows the sounds to be composed of many harmonics - the approximately horizontal striations - due to the abrupt impulse-like vibrations of the source (as opposed to smooth sinusoidal vibrations which would result in a single frequency component). The separation between adjacent harmonics is the fundamental frequency, or pitch, of the source. In the wideband analysis, individual harmonics are not resolved, but the impulses of the source can be seen explicitly as the vertical striations

1 The next chapter discusses the sonagram and related subjects, and Appendix E is an informal introduction to sonagrams in bird calls.

separated by more or less silent instants. The separation between the striations is the period of the fundamental vibration. Both sonagrams show the formant contours as the darker, wide bars - emphasised portions of the spectra.

Sources for this subsection: [20]-[24]

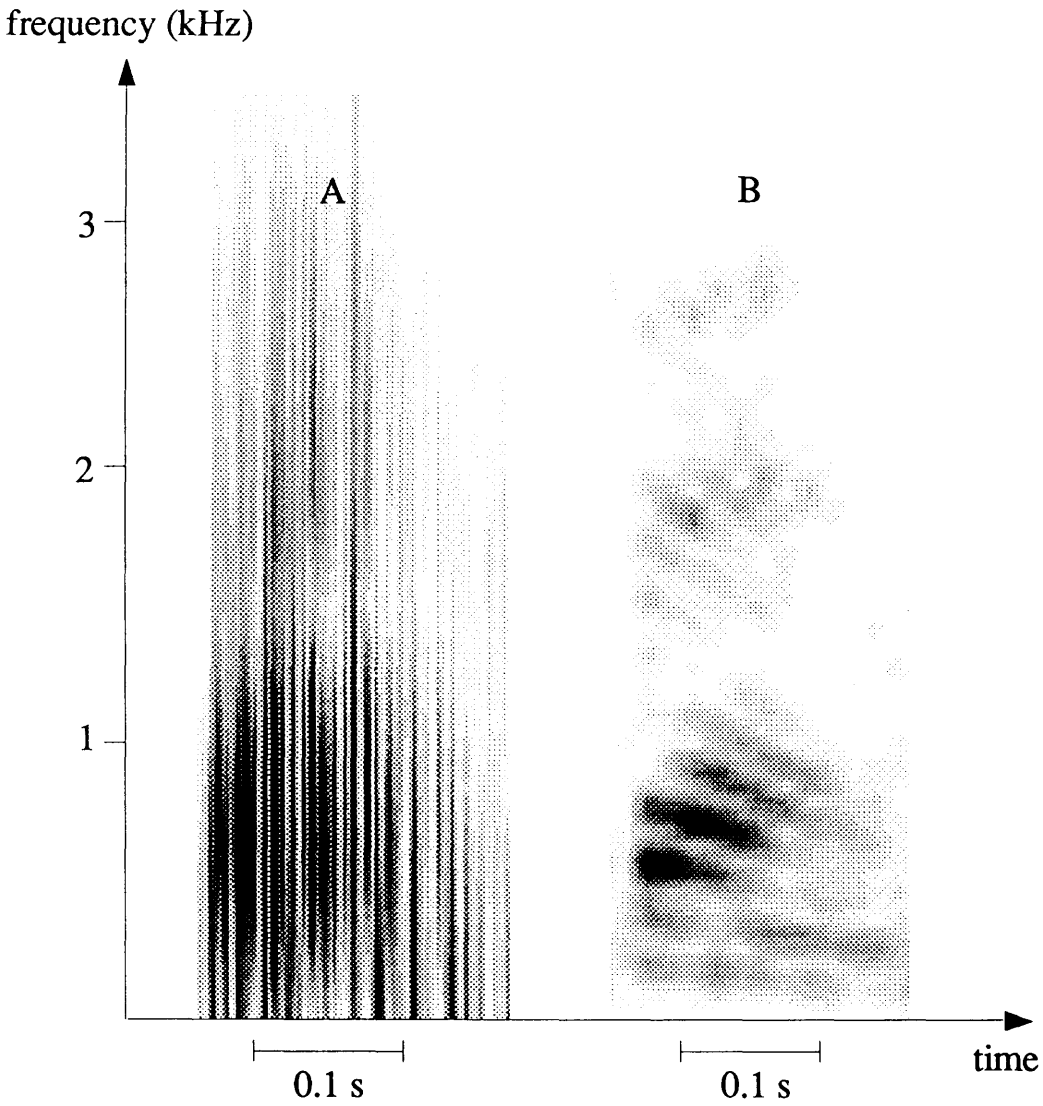


Figure 2.2 Sonagrams of human male voice (latter part of the word "Sang"). **A:** Wideband sonagram, in which the source pulses are visible as thin vertical bands. **B:** Narrowband sonagram, in which the source harmonics are visible as approximately horizontal bands.

2.2.2 Speech Recognition

Practical speech recognition systems are confined largely to the handling of a small vocabulary of isolated words. As the number of speakers and the vocabulary increases, so does the complexity and cost of the system rise, and the performance fall.

When dealing with an isolated word it is possible to form a prototype pattern or template for the word by sampling some of the parameters derived from the speech waveform. The unknown word is then sampled in a similar way and compared with the templates, and a numerical measure of the similarities is obtained. Such systems were typical of the first generation of speech recognition approaches and limited success in the recognition of 10-50 word vocabularies amongst known speakers were obtainable. Today robust, practicable systems for speaker independent small vocabularies are available.

Other systems which attempt to recognize basic phonetic units such as vowels, phonemes or syllables have been the subject of much research. This is because of the hope that large vocabularies would be recognisable by considering any word as a combination of a small set of phonetic units. While this approach has not been fruitless, the realization has come that there is no primitive set of isolated acoustic units corresponding to the linguist's phonemes.

An important feature of speech recognition is the existence of speech segments during which the spectral characteristics are approximately constant. An almost universal approach to speech recognition is the detection and individual analysis of these segments. The segments correspond approximately to the elusive phonetic units, but they are not always well defined and may change depending on the context in which they occur. Nevertheless invaluable information is obtained by characterising the spectral characteristics of each of these segments. Recognition of a word in terms of segments usually requires a time normalization of the input word so that it will "line up" with the time evolution of spectral characteristics of the template. This normalization can be a complicated process - one successful method, known as dynamic programming is outlined in [30].

Relatively few methods ignore the segmentation of speech. One such method is summarised in [6], where zero crossing counts of the time domain speech waveform are used as inputs to a syntactic pattern recognition system (see next section).

In general, the trend in speech recognition is away from brute force numerical representations of speech and towards a data reducing parametrization by feature extraction. The features used are usually some aspect of the formant contours, in other words the way in which the resonances of the tract vary. Whether the shape of the contours themselves are used, or the parameters of a model filter which would give rise to those contours, it is the spectral shaping of the source by the vocal tract that must be recognized in some way.

True speech recognition systems will have to make use of the many levels of knowledge about speech and talking that humans do, including acoustic, phonetic, linguistic, prosodic (accenting of syllables) and grammatical levels. This is because the acoustic speech signal does not code all the information necessary to understand it. Human understanding results from knowing just what words and sentences are plausible in the given context. It is worth noting that all but the most limited of speech recognition systems are highly complex and costly.

References for this subsection are [30]-[35].

2.3 Pattern Recognition

Speech recognition may not be directly applicable to bird calls. It is the nature of recognition systems to try to make use of the unique characteristics of the data they handle. The alternative would be a general "anything"-recognition system, which is one name we could use to describe human intelligence. Given that we can not as yet build a machine with anything like versatile recognition capabilities of a person, it reasonable to hope that by limiting the machine to a small range of input types, it will be simpler and therefore within the reach of current knowledge and technology.

So it becomes necessary to consider pattern recognition in general, in search of concepts that may be more applicable to bird calls than those of speech recognition. A *pattern* can be defined as a set of data which have some sort of organization, or underlying order, as opposed to being completely random. *Pattern recognition* refers to the process of allocating a class, or name, to an input pattern. For example, the set of co-ordinates of points obtained by digitizing an image is a pattern. A device or system which decides that the set of co-ordinates corresponds to a particular class of patterns has "recognized" the pattern. An example of a class of patterns is all those sets of data which come from digitizing a picture of the letter "B", written by many different hands and printed in many different styles. The above definitions of pattern and pattern class correspond with those in [17]

In the context of bird call recognition, a pattern is that set of data obtained by processing an input bird vocalization. A class is the set of patterns derived from the vocalizations of all individuals within a single bird species. Species may have more than one class of vocal patterns in their repertoire.

The process of pattern recognition can be loosely divided into two steps: *Feature extraction* and *classification*. Feature extraction is used here to mean the translation of input data into a set of parameters that are useful to the recognition system. To put it another way, in feature extraction the useful parts of the pattern are kept and the rest discarded. The "features" need not be intuitively

meaningful to human beings, although many methods do use such criteria. As an example of feature extraction, consider a system which "looks" at a page printed in the Latin alphabet and is required to recognize the language in which it is written. Much of the information on the page (ie in the input data) is useless in this task - the size of the text and the number of letters, for instance. Before deciding on the language it should first count how many letters 'A' there are, say, in comparison to letters 'E'. Perhaps it should check whether words such as 'the' or 'of' exist. The ratio of A to E and the presence or absence of the word 'the' are useful features of the input. In contrast, the presence of the word 'no' is less useful since it occurs in many languages - it is not a good feature.

The classification step involves allocation of the feature sets to named classes. In the above example, deciding whether the page belongs to the class "English" or "Italian" perhaps. Approaches to the classification process differ in the types of feature extraction, and the manner in which the features are used.

Uhr [16] distinguishes between three approaches to pattern recognition, the "template" approach, the "N-tuple" approach, and the broader class of miscellaneous approaches that apply an intuitively more satisfying method than the first two. The latter class includes syntactic (or structural) methods, which, being newer, have been less used historically. Thus Fu [6] and Gonzalez [17], who have researched the syntactic method, apply a different division; they distinguish between the syntactic approach and the rest, the classical decision-theoretic approaches. That division is adopted here.

An important aspect of pattern recognition is that of *learning*. By learning is meant the method by which the recognition system obtains the knowledge about what patterns belong to which classes. In a method where the features are numerous and the sample variation great, programming by people is not a viable method for "teaching" the machine. Moreover, the ability of the machine to adapt automatically is often quite desirable. Thus much effort is devoted to learning algorithms, although it is beyond the scope of this survey to discuss them.

2.3.1 Non-syntactic Pattern Recognition

Non-syntactic methods include the simpler and older methods of pattern recognition. Many of them use, in some way or another, template matching. The template is reference set of features representing an ideal member (in some sense) of a class of patterns. Usually the template is a matrix of numerical values. Associated with a template or a set of templates is a characterizer. An object is recognized by having its corresponding features compared to the template by the characterizer. The comparison can be viewed as a superposition of the input object's features over those of the ideal object, to see whether it "fits". The decision as to whether the input fits the template is made by obtaining a numerical measure of the "distance" between features of the object and the template. If the input object is an exact copy of the template the distance is zero.

The main issues in template matching are:

- How to represent the object (feature extraction).
- How to superimpose the object and the template. An example of this problem is the essential equivalence of two graphical images which have been rotated, shifted or enlarged relative to each other. A straight template match between the two would result in gross failure. The problem of superposition may be solved by changing the representation of the object.
- How to compare representations (obtain a distance measure).

The main advantages of template matching are conceptual and theoretical simplicity. All patterns may regarded as points in an N-dimensional space where N is the number of parameters or features used to represent the pattern. The grouping into pattern classes may be visualised as a physical grouping of the points in the space, that is, they are classified by position and distance.

The main disadvantages of template matching are the increases in complexity and computational burden that occur when the template is required to match an input object that is not a rigid, exact copy of the template [16]. As recognition complexity grows, so does the difficulty of teaching the system.

Other methods exist which split an input pattern into many smaller patterns that may then be recognized by template matching (see the N-tuple methods of [16]). This helps increase the flexibility of the system. Other methods attempt to extract features which are more "meaningful" - intuitively plausible [16], because this increases the likelihood that the features chosen are sufficient to recognize the pattern. For example the slope and length of a line may be important, but a feature set that includes every point on the line holds information about slight variations in thickness and direction that may be completely irrelevant, and actually detrimental for classification purposes.

Non syntactic methods which are not closely related to template matching exist, but all have in common the tendency to attempt to find numerical measures of the suitability of an input to fit in a particular class. A well known structure for all pattern recognition programs is that of many independent characterizers operating in parallel on an input pattern that has been preprocessed for feature extraction. The numerical "opinions" of the individual characterizers as to whether the input belongs to "their" class are operated upon to provide a decision about the class of the input.

2.3.2 Syntactic Pattern Recognition

The concepts used in syntactic pattern recognition are presented here in more detail than the non-syntactic methods. Simple string grammars (explained below) appear to be most relevant to

the bird call recognition problem as will be contended in Chapter 6, and the methods are more intuitive and flexible than the non-syntactic methods. Although a formal implementation of syntactic pattern recognition automata was not adopted for the bird call system, the ideas were, and the system could be implemented in a more formal manner if this became desirable. The information presented below was summarised from [17] unless otherwise stated.

Syntactic pattern recognition methods break the pattern into smaller patterns or features and recognize these as primitive elements but, unlike N-tuple or other methods, they use the knowledge about how the primitives are connected in order to recognize the pattern as a whole. Because of the importance of the interconnection of the features, these tend to be of the intuitively meaningful type. An example of a syntactic approach in the recognition of printed text is to first recognize the letters (possibly by a template matching technique) and then to check if the letters spell an existing word. Thus mistakes made in the letter recognition stage can be corrected with some confidence.

Syntactic methods, like others, use a collection of characterizers operating on the input pattern. Each characterizer tests to see whether the pattern belongs to a particular class. The characterizers are derived from formal theoretical descriptions of which patterns belong to which classes. The descriptions are called *grammars* and the characterizer derived from a grammar is called an *automaton*. A grammar is a set of rules for generating the set of all the patterns belonging to a single class. This set of patterns is called a *language*.

The simplest grammars are string grammars. In syntactic methods the primitive elements or features of a pattern are represented by symbols. A string grammar allows interconnection of symbols in only two ways: A symbol may either precede or follow another symbol. Groups of symbols connected in this way are therefore strings. Strings represent patterns.

More complex grammars are born when the ways of interconnection between symbols are extended. Tree grammars are composed of nodes. Each node may point to more than one node lower in the tree. A string is a tree with only one node leading from each preceding node. Web grammars allow more than one connection leading to a node, and plex grammars are the most general cases, allowing more than one mode of attachment to and from each node. Only string grammars will be discussed here, being the most widely used and best understood. It should be noted that some tree grammars can be transformed into string grammars.

The set of symbols allowed in the strings of a language is called the *alphabet*. The symbols consist of two types: *terminals* and *nonterminals*. The nonterminals are similar to variables in that they may be replaced with other symbols or strings during the building of the string. The terminals can never be replaced so that once all nonterminals have been replaced with terminals, the string is "finished". Finished strings are the patterns produced by the grammar. Thus the language of the grammar consists entirely of strings of terminals.

The set of rewriting rules for forming strings are called *productions*. If we let nonterminal symbols be uppercase letters and terminals be lowercase letters, then a production is written as follows:

```
G --> Ha
```

The above production simply means: Any occurrence of the nonterminal G in a string may be replaced by the string Ha. A terminal may not appear on the left hand side of a production.

As an example of a grammar, let us define an alphabet of nonterminals as (S,G,H); an alphabet of terminals as (a,b) and the set of productions as:

```
S --> H    (1)
G --> aH   (2)
G --> a    (3)
H --> bG   (4)
H --> b    (5)
```

S is a starting symbol. The language of strings (class of patterns) that can be formed by this grammar is obtained by starting with the string S and applying in any sequence any appropriate productions to the nonterminals remaining until no nonterminals remain. For example the string **bab** may be generated as follows:

```
        (1)      (4)       (2)        (3)
S  -->  H  -->  bG  -->  baH  -->  bab
```

Where the numbers show which production was applied. The set of strings generated by this grammar is:

```
b,ba,bab,baba,babab,bababa ...
```

Thus we see that the grammar is a formal way of defining the class of all strings which begin with **b** and consist of alternating **b**'s and **a**'s.

The advantage of formalising class definitions in this manner is that the grammars may be mathematically analysed and the theoretical limitations of pattern recognizers based on them can be derived. Another advantage is that the grammars required to classify patterns can be automatically generated from examples. That is, the pattern recognition systems can be programmed to adapt and learn. Learning algorithms are not however trivial to implement, nor are they perfect. Finally, the automaton used to perform recognition based on one of the simpler grammars can be derived from the grammar in a straightforward fashion.

A four-level hierarchy of grammars exists such that the lower grammars in the hierarchy are simpler and more easily analysed than the higher ones. The lowest grammar type is a *regular grammar* and the next lowest is a *context-free grammar*. The two lowest grammars are those most often used in practical pattern recognition systems. Types of grammars differ in the kinds of productions they can contain - the exact difference is not important at this stage. More important are the corresponding types of automata which are derived from these grammars, because they give a good idea of what the structure of a pattern recognizing algorithm should be. The point being made is this:

• Any algorithm which contains the essential features of the automata described below may be assumed with some confidence to be theoretically capable of recognizing the patterns which conform to the most commonly used grammars.

An automaton derived from a regular grammar is called a *finite state automaton*. Such an automaton is well known to electrical engineers in its implementation as a digital circuit. Any digital circuit derived from a state diagram which shows the states and associated outputs, and how transitions occur between states in response to input variable values, is a finite state automaton. In the present context, there is a single input variable which takes on, in sequence, values corresponding to the symbols of a string. It is this string that the automaton is must recognize or not. The next active state of the automaton is determined by the current active state and the current symbol being input. After a state transition occurs, the input variable takes on the value of the next symbol in the string.

The only output of the automaton is "accept" or "reject". When the last symbol of the string has been input, and the last transition between states has occurred, then if the current active state is one of a set of predefined "final" states then the output is "accept" - the input string belongs to the language recognized by the automaton. If the last active state is not a "final" state, the input has not been recognized.

The method by which a finite state automaton is derived from a regular grammar is not described here. It is quite possible to design an automaton and then, if desired, generate the corresponding grammar by applying the rules in reverse. The finite state automaton is a simple and easily implemented machine, but is limited in its application. There is no possibility for recursive definitions of strings to be recognized by finite state automata, because by definition there is a finite number of states, which implies a finite memory for the machine.

A grammar which can handle a type of recursion called *embedding*, in which strings can be embedded in the middle of existing strings rather than being restricted to concatenation at the ends, is the context free grammar. An automaton derived from such a grammar is called a *pushdown automaton*. As the name implies, pushdown automata are implemented using a pushdown stack structure. The operation is similar to a finite state automaton in that the symbols in the input string are applied to the input of the automaton one at a time in sequence. The automaton "guesses" at

A four-level hierarchy of grammars exists such that the lower grammars in the hierarchy are simpler and more easily analysed than the higher ones. The lowest grammar type is a *regular grammar* and the next lowest is a *context-free grammar*. The two lowest grammars are those most often used in practical pattern recognition systems. Types of grammars differ in the kinds of productions they can contain - the exact difference is not important at this stage. More important are the corresponding types of automata which are derived from these grammars, because they give a good idea of what the structure of a pattern recognizing algorithm should be. The point being made is this:

• Any algorithm which contains the essential features of the automata described below may be assumed with some confidence to be theoretically capable of recognizing the patterns which conform to the most commonly used grammars.

An automaton derived from a regular grammar is called a *finite state automaton*. Such an automaton is well known to electrical engineers in its implementation as a digital circuit. Any digital circuit derived from a state diagram which shows the states and associated outputs, and how transitions occur between states in response to input variable values, is a finite state automaton. In the present context, there is a single input variable which takes on, in sequence, values corresponding to the symbols of a string. It is this string that the automaton is must recognize or not. The next active state of the automaton is determined by the current active state and the current symbol being input. After a state transition occurs, the input variable takes on the value of the next symbol in the string.

The only output of the automaton is "accept" or "reject". When the last symbol of the string has been input, and the last transition between states has occurred, then if the current active state is one of a set of predefined "final" states then the output is "accept" - the input string belongs to the language recognized by the automaton. If the last active state is not a "final" state, the input has not been recognized.

The method by which a finite state automaton is derived from a regular grammar is not described here. It is quite possible to design an automaton and then, if desired, generate the corresponding grammar by applying the rules in reverse. The finite state automaton is a simple and easily implemented machine, but is limited in its application. There is no possibility for recursive definitions of strings to be recognized by finite state automata, because by definition there is a finite number of states, which implies a finite memory for the machine.

A grammar which can handle a type of recursion called *embedding*, in which strings can be embedded in the middle of existing strings rather than being restricted to concatenation at the ends, is the context free grammar. An automaton derived from such a grammar is called a *pushdown automaton*. As the name implies, pushdown automata are implemented using a pushdown stack structure. The operation is similar to a finite state automaton in that the symbols in the input string are applied to the input of the automaton one at a time in sequence. The automaton "guesses" at

how the string, as presented on the input, was formed by choosing a production that might have resulted in the current input symbol. The stack is used to remember these choices so that if they turn out to be incorrect, the automaton can return to try a different sequence of productions. If the automaton can reach the last symbol in the string, the latter has been recognized. The automaton chooses the productions either randomly (providing that the present state and input allow), or directed by some heuristic information about which productions are more likely to be used.

Many augmentations to the concepts and methods described here exist. Learning algorithms, specially designed grammars to make learning easier, and statistical methods of aiding the recognition are some examples. The theory becomes more complicated and is beyond the scope of this project.

To conclude, an example of the use of syntactic methods in speech recognition is cited. [18] summarises a paper by DeMori in [19] in which a system for isolated word recognition was investigated. The input word (a single short utterance) was analysed in terms of the number of zero crossings occurring in the waveforms in 20 ms time frames. These numbers were plotted as points in two dimensions. The graphical images resulting were characterised in terms of elemental features such as silences and straight lines. A grammar with an alphabet of 3 terminals and 17 symbols, and with 52 productions was required to recognize the first ten spoken digits "ONE" to "TEN" with high accuracy. Each production had on average 3 decisions (checking whether a certain parameter lay within a certain range) to determine whether it should be applied. The algorithm was efficient enough for real time application.

The above example highlights the immense complexity of the problem of pattern recognition. The recognizer was capable of identifying only ten isolated, well spoken utterances, although increasing its capability was simply a question of adding productions and decision data. The example also highlights the suitability of syntactic methods for representing combinations of graphical elements. This is especially significant because as humans we are comfortable with graphical representations of information, and are most easily able to abstract important features from such representations. Syntactical pattern recognition gives us a way of translating, to some extent, our intuitive pattern recognition abilities into working machines.

3
Computer-based bird call analysis

3.1 Representing bird calls

Before any analysis of the characteristics of bird calls can begin, a means must be found to represent them formally. The primary choice in representing a signal is whether to do so in the *time domain* or in the *frequency domain*, the implications of which are briefly sketched as follows:

3.1.1 Time and frequency domain concepts

Representing the sound in the time domain means recording the moment-to-moment variations in air pressure as the height of successive points on a graph. Any changes in amplitude (scale of the waveform), and gross changes in frequency characteristics (shape of the waveform) would show up at exactly the point in time at which they occurred. The changing nature of the call - the fact that it consists of different sounds at different times, would be well represented.

In the frequency domain the sound is considered to be built up from component sinusoidal waveforms, each with a distinct relative strength (magnitude) and frequency (rate of oscillation). The sinusoidal components are absolutely regular and repetitive, and are considered to have been so and to continue to be so forever. The frequency analysis quantifies the existence of a regularly recurring pattern in the waveform. The frequency domain graph, which shows the magnitude of each component plotted against it's frequency, is the *spectrum*. If the signal is truly regular, i.e. it is periodic, the spectrum shows a finite number of discrete points.

If the signal is aperiodic, an infinite number of points in the spectrum appear, accounting for the fact that a finite number of periodic components always add up to a periodic signal. If the signal is random (white noise) then components at all frequencies, from zero to infinity, will be present, all with the same magnitude. If the signal is not purely random, the components are no longer of equal magnitude (the spectrum is no longer *flat*). Any parts of the signal which form a pattern show up as an increase in the magnitude of some components. The emphasised components will have frequencies related to the interval over which the repetition occurs.

The difference between the two representations thus becomes apparent: The time domain specifies the state of the signal at any given time, so that the moment at which any change occurs is always clear. It says nothing about the long term behaviour of the signal and thus cannot be formally be

used to reveal any sort of pattern which may have emerged. The frequency domain is just the opposite. Patterns of behaviour developing over time show up as emphasised frequencies in the spectrum. But there is no information as to where (when) in the signal the pattern emerges, or whether different patterns existed at different times. The time domain specifies local events and the frequency domain identifies global trends. Just as the time representation is concerned only with single, infinitesimally short moments, so the spectrum examines the signal from the perspective of an infinite interval of time - from infinitely long "ago" until infinitely far in "the future".

3.1.2 Which domain for bird calls?

How do the particular advantages and disadvantages of the two domains apply to the case of bird calls? To answer this, an appraisal of the nature of a bird call, as implied by the discussion in Chapter 2, is necessary. Also, the mechanism of auditory perception is relevant: Hearing can be thought of as "auditory representation of sound" and conversely the representation method which is being sought is a kind of "non-auditory perception of sound". Since the hearing mechanism employed in nature is the legitimate target for the sounds in nature, especially for sounds such as bird calls which are intentionally produced for communication purposes, it is reasonable to assume that it is an efficient and sufficient method of representation. The principles of auditory perception are therefore used as a base from which the computer representation of bird calls is developed, and to introduce the relevant concepts.

The basic sound is that due to the vibratory action of the bird's syrinx. Therefore, the basic nature of the sound is periodic. It has a fundamental frequency, which is perceived as the overall pitch of the sound. A spectral representation would therefore immediately seem viable. The subject of perception itself can be used as a guide to meaningful representation. The inner ear of birds (and humans) is primarily a detector of frequencies. In effect it performs a type of spectral analysis, sending information to the brain specifying which frequencies (components) are present in a sound. Whatever information is contained in the call, it is perceived by the bird as a collection of frequency components, so use of the spectrum could justifiably figure in bird call representation.

A bird call is not, however, a strictly periodic signal. It starts with silence, then a series of different "notes" or elements occur, followed once again by silence. That sequence is not repeated indefinitely. The elements may differ in amplitude, or in pitch, or in the quality of the sound (for example, the whistle of the sparrow as against the honk of a goose). Some bird sounds - hissing and clicking - are more like random noise than periodic tones.

In view of the time-varying aspects of a bird call, should the time domain representation not be considered? Certainly performing a spectral analysis of a bird call in its entirety would not be suitable. The effect of individual, changing elements would be indistinguishable - they would influence the shape of the spectrum but there would be no clue as to their existence as separately detectable units.

On the other hand, looking once again to the perception of the sounds, it is clear that the call is indeed heard as a frequency domain phenomenon. It has a pitch, or a "quality" - an overall continuous sensation is perceived rather than the moment-by-moment variation of air pressure. A closer examination of perception yields a clue as to how the problem of a time-varying frequency source can be solved: The effect of a variation depends on the **rate** of variation. Consider a pure tone whose amplitude is varying sinusoidally with time. From a mathematically ideal point of view the signal can be described equally well in both domains. In the time domain, the description is as above - a sinusoid modulating the amplitude of another sinusoid. In the frequency domain, two new components would appear where before there was one[1]. How does the auditory mechanism of a bird or human choose to perceive this signal?

The answer is straightforward - if the frequency of the modulating wave, that is the rate of amplitude variation, is low then we hear the variation as a time domain effect - the modulated tone grows alternately louder and softer. If the modulating frequency is fast, we hear it as a frequency domain effect - two tones at different frequencies than the original unmodulated tone, combining to form a sound of distinctly different quality. The exact range of frequencies that are considered "low" or "high" will be dealt with, for the case of bird hearing and bird vocalizations, later in this chapter. For now the point is merely that the distinction is important.

So auditory perception is seen to employ a **mixture** of time and frequency domain effects. The advantages of both domains are brought to bear, but it is important to understand the limitations of such an approach, which are really the limitations of representing a real signal using an idealised concept. Thus before describing a practical means to obtain the mixed representation, using digital signal processing, a brief outline of the mechanism of the ear and other similar systems will serve to introduce the factors limiting real-world representations of bird calls.

3.1.3 Real-world spectral analysis - filters, the ear and the sonagraph

Spectral analysis, the process of transforming a time-domain signal into the frequency domain, is performed in nature by the inner ear. Machines have also been constructed which do this job - the

1 Where before there was a single component at F_c, there would be one at $F+F_c$ and one at $F-F_c$, where F_c is the frequency of modulated sinusoid, and F the frequency of the modulating sinusoid.

sonagraph records the spectral information on paper, and is used to represent exactly the type of signals of interest in this case - speech and bird calls. The two work using the same basic principle - that of the *filter bank.*

A filter is a device or system into which a signal is fed, and which outputs that signal after the magnitudes of some of its frequency components have been altered. For example, a transparent red film turns the colour of light passing through it from white to red. The white light, containing a range of frequency components, is the input signal. The filter reduces to zero the high frequency components, that part of the visible spectrum corresponding to green through to violet. The low frequencies, the red part of the range, pass through unattenuated to become the output.

The red filter is a *low-pass* filter because it suppresses all frequencies greater than a certain value. Its complement is the *high-pass* filter, suppressing all frequencies from 0 up to a given value. A combination of the two is a *band-pass* filter, so called because it allows frequencies in a particular range, or band, to pass, suppressing all frequencies below the lower limit of the range and above the higher limit. A general filter suppresses arbitrary frequencies by arbitrary factors, but the simple band-pass is all that is needed for the filter bank.

The filter bank is constructed from a sequence of band-pass filters such that the upper limit of the passband of one filter is the lower limit of the passband of the next. Thus the first filter might allow frequencies in the range 0 to 100 Hz to pass unaltered, sharply attenuating frequencies out of that range. Another filter would pass frequencies in the range 100 Hz to 200 Hz and so on, the last filter passing, say, 900 Hz to 1000 Hz. Such a bank would then be capable of detecting frequencies in the range 0 to 1000 Hz. The signal is applied to the inputs of all the filters simultaneously. The output of the filters is observed. If a particular filter has zero output, this indicates that no frequencies in the range belonging to that filter are present in the input signal. Conversely, the level of activity of a filter indicates the proportion of the input signal that is concentrated in the filter's range[1]

In the case of the sonagraph, the activity signals from the filters are fed to a stylus. The stronger the signal, the darker and thicker the line drawn by the stylus. The line is drawn on a strip of paper wrapped around a rotating drum. As the drum rotates, the signal is fed to the stylus through one of the filters, so the line drawn is a measure of the activity in the signal in a small range of frequencies, versus time. At a time when there is no activity, there will be a break in the line. By the time the drum rotates fully and the stylus reaches the starting point of the line, the signal (which has been

1 The *activity* of the filter is simply the signal composed of the sum of the components in the filter's range. If there is only one component, for example, the output of that filter will be a sinusoidal waveform. In order to assign a quantitative measure to the activity, the signal is integrated or averaged in some way. This means that the output of the filter-integrator is a measure of the power (energy per unit time) present in the signal in the particular frequency range.

recorded) is restarted, and fed back to the stylus through the next filter in the bank. The stylus is displaced so that the line it draws is now parallel and adjacent to the previous line. The new line traces the activity over time of a different range of frequencies. Ultimately, the strip of paper is filled with parallel lines, each one measuring different frequencies. Thus it is possible to see whether or not a particular component was present at any particular time, by looking for the line which contains that component within it's range, and following the line until the point at which the moment of interest is represented. In this way the sonagram is created as a mixed time-frequency domain representation of the sound signal.

The inner ear can not be as closely associated with an actual bank of filters as a sonagraph. Nevertheless the operation is mostly analogous. Different sensors arrayed along the cochlea respond to different frequencies as would different filters. The signal from each sensor, sent separately to the brain, is a measure of the strength or energy of the signal in a particular range of frequencies, the filtering effect taking place due to the complex acoustic behaviour of the inner ear - some frequencies penetrate further into the cochlea, reaching different sensors.

So far the form of a real spectral analysis system has been outlined; now it is necessary to discuss its behaviour.

3.1.4 Frequency and time resolution in real systems

First, there is the question of frequency resolution. The filter bank indicates the presence of a component in a particular range - it does not pinpoint the exact frequency. Clearly if greater accuracy is required, the filters must be made to have narrower passbands, and more of them used to cover a given range of frequencies. There are practical limitations on such a process, however. A filter cannot be manufactured with an absolute cutoff point. That is, it is not possible to build a filter which, while passing unattenuated a particular frequency, completely attenuates the next. Instead a continuous, gradual change must take place, so that if at frequency F_1 the attenuation is zero, and at F_2 it is some finite value A[1], then for frequencies between F_1 and F_2 the attenuation must take on intermediate values, increasing from 0 to A. This limitation means that the ranges of adjacent filters overlap. In a filter bank, it necessary to ensure that where this overlap takes place, the attenuation of one filter is high compared to that of the other, so that any given component chiefly influences one filter only. So when reducing the bandwidth of these filters (in making the ranges smaller), the transition from pass band to stop band must be made more acute, or the relative overlap will increase. Figure 3.1 illustrates this point.

1 For theoretical rather than practical reasons, the attenuation cannot be infinite, as explained further on.

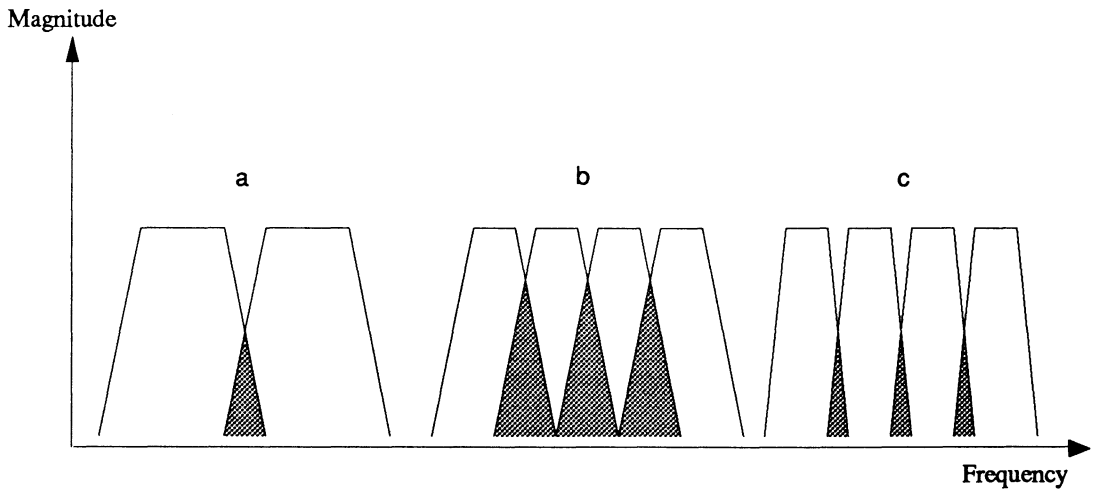


Figure 3.1 Schematic of 3 filter banks. **a:** Two wideband filters are used to cover a particular range of frequencies. **b:** Four narrowband filters with half the bandwidth and spaced twice as close together are used to cover the same size band as in **a**, but because the filters have the same transition characteristics (indicated by the slope) as those in **a**, the relative overlap (grey region) is greater. **c:** The rate of transition (slope) from pass band to stop band has been increased, resulting in the same relative overlap as for **a**.

Thus to obtain a higher frequency resolution, more filters with steeper cutoffs must be used. A steeper cutoff requires a more complex (higher order) filter, so there is a price to pay for increased frequency resolution. But a fundamental theoretical behaviour plays a far more important role in restricting the resolution: The tradeoff between the frequency and time resolutions.

To illustrate the tradeoff, consider a bell's behaviour as a filter. When the bell is struck, it receives a pulse-like input. The ideal mathematical concept of a short pulse is the *impulse* - energy concentrated into a single, infinitesimally short time, in an infinitely intense burst. Such a signal has a flat spectrum - it contains all frequencies, from 0 to infinity, all with the same magnitude. A real pulse must exist for a finite amount of time, and have a finite amplitude. Its spectrum is not flat - the magnitude of the components vary and diminish at higher frequencies. Nevertheless, it contains many frequency components, spread over a range of values that is greater the closer the pulse approximates an ideal impulse.

The bell can be thought of as a very narrow bandpass filter. Most of the components in the input pulse are therefore rejected, and only a small range of frequencies are emitted. The result is a nearly pure sinusoidal output waveform - a single note. The short pulse is converted into a long, ringing sound. It is this lengthening of the pulse that is of interest.

The lengthening effect is dependent on the bandwidth of the filter. A filter with an infinite bandwidth, that is an "all-pass" filter, does nothing to the input - whatever frequencies are fed in are output - so an impulse would remain an impulse at the output. Conversely, a filter with an infinitesimal bandwidth - one which passes a single frequency value only - would emit a pure sinusoid, which continues forever. In between these two ideal extremes are the practical filters; the narrower the bandwidth, the better the filter is able to isolate a particular component and the longer it "rings" in doing so.

The impulse is the time domain counterpart of the frequency domain's *component*. Just as a signal can be thought of as a sum of sinusoids of various frequencies, so is it a sum of impulses occurring at various times. The impulse embodies the essential time domain characteristic - change. Nothing can change faster or more abruptly than an impulse. What this means is that the response of a system to an impulse can be used to deduce it's response to any arbitrary signal. In this case, the deduction is that a filter lengthens the effect of any change. If, for example, the input to a narrow band filter were a sinusoid with a frequency that fell within the passband, the output would be similar to the input. If the input is abruptly changed to another frequency, the filter output continues to "ring" for a while before its output begins to reflect the fact that the input has changed. Exactly like the bell which rang on after the pulse input had ceased.

A band pass filter therefore "blurs" the effects of abrupt changes at its input - it is impossible to pinpoint exactly when a change occurs. Effectively the time resolution of the system is reduced.

How does this impact the filter bank as a frequency analysis tool? Essentially, the effect is responsible for the mixing of domains in the ear to which reference was made at the end of the last section. When a sequence of changes in the input signal occur close together in time, the events blur and are perceived as a continuous sound - a frequency domain effect. When the changes are far apart in comparison to their "ringing" time, the are perceived as discrete events - occurrences in the time domain.

The effects are easily seen in sonagrams of speech, where a distinction is made between wideband and narrowband sonagrams. Referring to Figure 2.2, the wideband sonagram has filters with a short "ringing time" so that the individual pulses of the larynx are seen as vertical lines. As each pulse "strikes" a filter, it activates it (or not) according to the frequencies constituting the pulse. By the time the next pulse arrives, that activity has died down. In the narrowband sonagram, the pulses join in a continuum of activity - but the filters accurately pinpoint components, resulting in thin lines where the wideband graph showed only general smears. It is not that the pulses which "should" be there are omitted - their effect is perceived as the multitude of harmonic components.

3.2 Digital frequency analysis of bird calls

Until now, the discussion has centred around analogue systems and loose descriptions in order to sketch the main features of spectral analysis. Analogue systems could certainly be interfaced to a computer-based recognition system, and such a proposition might have practical advantages, but the spectral analysis can be performed entirely on the digital computer, and this is the approach taken in the present project. The advantage of digital frequency analysis at the experimental stage is its flexibility and power of simulation of a variety of different analogue arrangements.

In the following sections, the implementation and use of a digital frequency analysis system is described in technical detail. Such systems are in common - almost everyday - use in digital signal processing applications, and no particular innovations or deviations from the standard are required in the application to bird calls. In fact, the only decisions specific to the case of bird calls will pertain to the issue of frequency versus time resolution, the same theoretical limitations introduced in the preceding section. For these reasons, the details that follow are of a descriptive rather than didactic nature. Some basic concepts of digital processing are, however, introduced.

3.2.1 The Discrete Fourier Transform

The mathematical tool for performing spectral analysis is the Fourier transform. It transforms a signal, say a function of time $f(t)$, to a spectral density function $F(\omega)$, which specifies the magnitude and phase of the component at each angular frequency ω

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$

Note that frequency *freq* is related to the angular frequency ω as

$$freq = \omega/2\pi$$

The functions $f(t)$ and $F(\omega)$ are in general complex, having both real and imaginary parts. The complex values of the spectral density function specify the sinusoidal components of the spectrum - the magnitude and phase of the component at frequency ω_0 are

$$|F(\omega_0)|$$

$$\text{ang}[F(\omega_0)]$$

respectively. In the present context, $f(t)$ is a real-world signal (a bird call) and thus has the imaginary part is set to zero. This has the mathematical consequence that $F(\omega)$ is symmetrical about the

vertical axis through the origin. Thus the spectrum values for $\omega < 0$ convey no additonal information and require no physical interpretation - they are taken into account only in rigourous mathematical treatments.

The mathematical concept of the Fourier transform handles signals which can be expressed as mathematical functions. Practically, such signals are not of much interest - it is the unpredictably changing signal which conveys information. To represent such a signal we have no choice but to record its behaviour at each moment in time. If we expect the signal to be changing unpredictably literally from one moment to the next, we need to record an infinite number of points during any given interval, because a moment has no length. Practically this is impossible, but fortunately not necessary.

A real signal has a limit as to how fast it changes. This limit can be understood from a frequency domain perspective - we say the signal is *bandlimited.* An ideal bandlimited signal contains no frequency components higher than a cutoff of F_c Hz. In a real signal, there are non-zero components at all frequencies, but after F_c, their magnitudes are so low as to be insignificant. In a bandlimited signal, at no point in the time evolution of the signal can the amplitude change faster than that of a sine wave of frequency F_c. The *sampling theorem* says that such a signal can be reconstructed without loss of information from a sequence of samples of the signal taken every $1/2F_c$ seconds. That is, the sequence of numbers representing the signal is a complete description of the signal if the amplitude values are recorded at the finite rate of $2F_c$ Hz - so for a given interval there will be a finite number of samples fully representing the signal.

If the sample sequence is a full representation of a given continuous signal, then it should be possible to obtain the spectrum of the original signal from that sequence. The Fourier transform for a discrete-time waveform (a sequence of samples) is identical to the continuous form, with the integral replaced by a sum:

$$F(e^{j\omega}) = \sum_{n=-\infty}^{\infty} f(n)e^{-j\omega n}$$

The variable n in the above equation is the sample number. If the samples are taken from a function $f(t)$ then $f(n)$ is short for $f(n\mathrm{T})$, where T is the sampling period (the time between consecutive samples). As the equation shows, the spectrum shape is determined by a sum of complex exponentials of discrete frequencies. The exponential function $e^{j\omega}$ is the complex form of the sinusoidal function and is therefore the general form of a fundamental spectral component. It is periodic of period $2\pi/\omega$, and a sum of such functions with discrete periods has a periodic result. Thus the spectrum $F(\omega)$ resulting from a sequence of samples differs from the continuous case in that the spectral shape is periodically repeated. The period of repetition is the sampling frequency, $f_s = 1/T$. The notation $F(e^{j\omega})$ is used instead of $F(\omega)$ as a reminder that it is a periodic function of ω.

The periodicity of the spectrum justifies the limitation introduced by the sampling theorem. If the spectrum is repeated every f_s Hz, then if there are significant components present beyond $f_s/2$ (since the spectral shape is symmetrical about the 0 Hz axis), then these components will overlap into the adjacent periods. This results in components above $f_s/2$ appearing in the areas representing the lower (below $f_s/2$) frequencies, an effect known as *aliasing*. Another way to explain aliasing is that the samples are taken too slowly to track a rapidly changing signal, in which case it appears to be changing less rapidly.

In the preceding discussion the implications of using discrete samples to represent a time domain signal were outlined. The spectrum was still calculable, and was related to the spectrum of the continuous signal, but had to be bandlimited, and was periodic. The spectrum was, however, still continuous. On a digital computer, the signal in the frequency domain must also be represented by discrete numbers - samples of the spectrum. This has an effect on its interpretation. As mentioned previously, a periodic input signal can be constituted from a finite number of sinusoidal components, and thus has a spectrum in which only certain disjoint points appear. So a periodic signal has a naturally discrete spectrum.

By forcing the digital signal's spectrum to be discrete, the signal itself is forced to be periodic. The sequence of time samples that is submitted to the transform is considered to represent a single cycle in a periodic waveform. Just as the signal had to be bandlimited in the frequency domain in order to allow it to be discrete in the time domain, so must the time signal be limited (i.e. must have a finite duration) so that it can be repeated in a cycle, allowing a discrete frequency domain representation. Because both signals are limited, the Discrete Fourier Transform (DFT) can now be formulated for a time sequence of N samples:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j(2\pi/N)kn}$$

The sample $F(k)$ is unique only for $k = 0...N-1$, after which the $F(k)$ sequence repeats.

Summary: Discrete Fourier Transform

The Discrete Fourier Transform or DFT transforms N equally spaced samples of a waveform to N equally-spaced samples of the waveform's spectrum. Both the time and frequency sequences are interpreted as periods in a periodic sequence. For this reason the original time and frequency waveforms must be limited (finite-duration and bandlimited respectively) if the interpretation of the sequences as samples of those waveforms is to be valid. The two limitations lead to two error effects - the first, aliasing, occurs when the signal is bandlimited to frequency f_c and the sampling rate is less than $2f_c$. The second effect, sometimes called *leakage*, will be discussed in section 3.2.2.

3.2.2 The Short Term Fourier Transform

The DFT embodies a practical method for obtaining the spectrum of a signal. Previously the analogue apparatus for performing this task, the bandpass filter bank, was shown to have inherent limitations in its ability to pinpoint an event in time or a frequency component. It was stated that the tradeoff between time and frequency resolution was a result of a theoretical restriction, so a similar tradeoff should become apparent in the digital frequency analysis. To see this, consider how the DFT can be used to perform a similar job to the ear or sonagraph.

How does the DFT analyse a signal such as speech or a bird call? If the entire signal is sampled and a single transform performed, no information on the nature or order of events occurring within the utterance will be available. The sound starting and stopping, the pitch changing, these and all other events will be transformed to the appropriate frequency domain phenomena, namely the weight of various components. Clearly it necessary to split the utterance into short segments, performing a separate DFT on each segment. Thus if the bird is calling at a certain pitch at time t_1 and a different pitch at time t_2, this fact will be recorded by a change in the position of a frequency component between two spectra, as long as the spectra are DFT's of two separate sequences, the first containing the events around t_1 and the second around t_2.

Thus we find that the DFT's are to be performed on short segments of the signal. The shorter the consecutive segments used, the higher the time resolution captured. When time resolution is increased in the sonagraph, it is at the expense of frequency resolution - how does this tradeoff appear through the DFT mechanism? Consider the mathematical process of extracting a segment illustrated in Figure 3.2.

Figure 3.2a shows a sinusoid and its magnitude spectrum, an impulse representing the concentration of the signal's energy at one frequency value. The negative part of the spectrum is shown for completeness. Performing a Fourier Transform on only a portion the signal is mathematically modelled by multiplication of the signal by a function like the rectangular one shown in Figure 3.2b, before submission to the transform (which integrates over all time). The rectangular function's spectrum is given by the (magnitude of the) *sampling function*, sin(x)/x, as shown. Figure 3.2c shows the product of the sinusoid and rectangular function in the time and frequency domains. The spectrum of the resultant signal is obtained mathematically by convolution of the individual spectra, and when as in this case one of the spectra is an impulse, results in the shifting of the other spectral shape to centre around the frequency of the impulse.

Figure 3.2 thus illustrates that the effect of truncating a sinusoid is to add frequency components spread around the original frequency. This spreading is sometimes called leakage, since the influence of the original component "leaks" over to neighbouring components. The effect of truncation on a more complex signal than a sinusoid can be deduced from the above. In general a

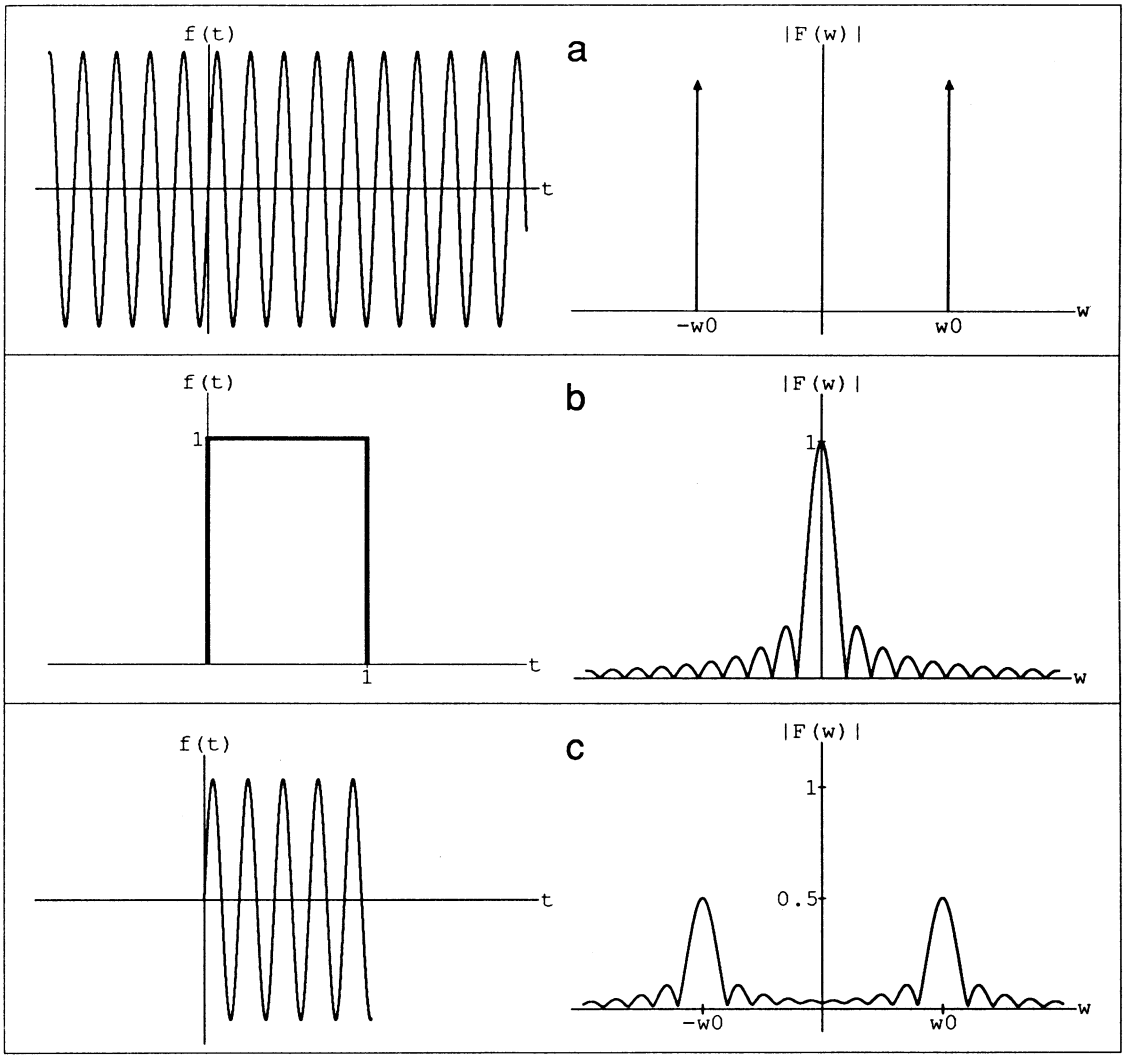


Figure 3.2 Time waveforms (left) and magnitude spectra (right). **a:** Sinusoid f(t) = sin (w_0t). **b:** Rectangular function. **c:** Truncated sinusoid (product of the time waveforms in **a** and **b**).

signal is a sum of sinusoids in the time domain, and a sum of impulses at the frequencies corresponding to those sinusoids in the spectral domain. By truncating the signal, all the sinusoids are truncated, resulting in a sum of sampling function shapes where impulses (single points) used to be. The summation is done in the complex plane and not by summing magnitudes or such like.

So the Fourier transform of a truncated signal does not pinpoint exactly where the frequencies are - neighbouring components leak over to influence the magnitude and phase at each frequency. There is a blurring - a reduction in frequency resolution. The effect is the complement of the

pulse-lengthening effect observed in an analogue bandpass filter. Both have the same mathematical basis, which can be seen in the Fourier transform of the rectangular function, explained in the following paragraphs.

As was pointed out in the section on filters, the behaviour of a filter can be quantified by its response to an impulse input. In the case of an ideal bandpass filter, the output signal spectrum must have the shape of a rectangular function. This is because the impulse input contains all frequencies, and the filter then passes a range of these, attenuating to 0 all the rest. Given the spectrum of the output, its time waveform can be found by applying the inverse Fourier transform - which is mathematically identical to the Fourier transform. Thus the time waveform which gives rise to the rectangular spectrum is found once again to be a sampling function. The infinitely short impulse emerges from the ideal bandpass filter as a longer, sampling-function shaped pulse.

The time-frequency resolution tradeoff is thus expressed in the relationship between the rectangular function and its Fourier transform, the sampling function. Figure 3.3 illustrates this relationship.

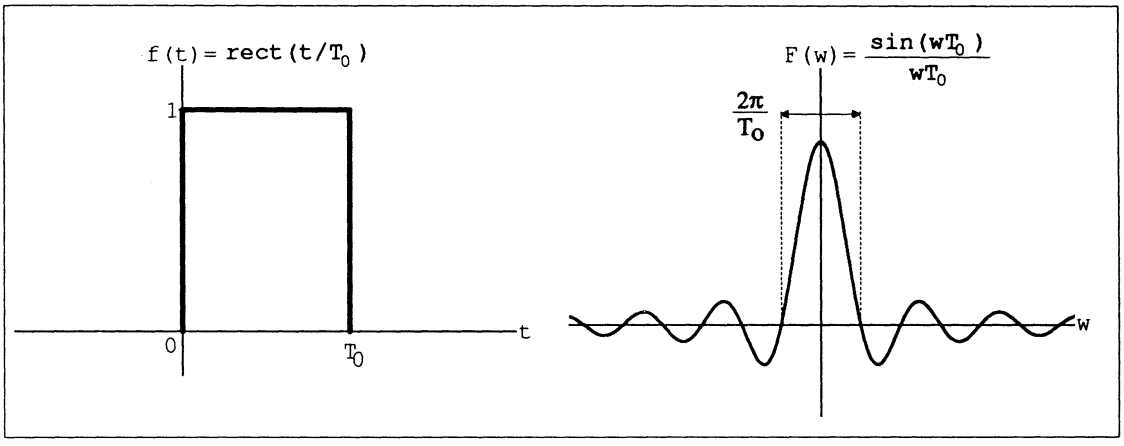


Figure 3.3: The rectangular function (left) and its Fourier transform (or inverse Fourier transform), the sampling function (right).

The most important connection between the two functions is that of the rectangular's width, T and the sampling's width. The sampling function is seen to consist of a main peak or *lobe*, with smaller side lobes placed symmetrically on either side and diminishing in height with distance from the main lobe, but never quite fading to zero. A characteristic width of the sampling function can be chosen as the width of the main lobe, which is $2\pi/T$. The width of the main lobe (and successive lobes) is seen to be inversely proportional to the width of the non-zero part of the rectangular function. This is once again the reason for the narrow-bandpass filter (whose frequency response

is a narrow rectangular shape) producing a long ringing pulse in response to an impulse, and for the spreading of a signal's spectral components in a Fourier transform of a short cut-out (rectangle) of a signal.

The fact that the sampling function extends over the entire time range (does not die down to zero) is also important. It is the reason a filter with a vertical cutoff cannot be a physical reality. When the sampling function is viewed as the output of an ideal filter in response to an impulse at time t=0, it becomes clear that such an output is impossible. The filter starts responding at time $t < 0$; it is said to be *non causal.* A real bandpass filter has a gradually increasing attenuation rather than an abrupt cutoff. The principle of the inverse relationship between filter width and impulse response length remains valid and useful.

Returning to the practical Fourier transform, the DFT, a similar idealization must be dealt with. There can be no doubt that the extraction of a segment from a signal implies the application of an exactly rectangular function. The signal is strictly time-limited. The theory says that its spectrum should therefore be composed of a sum of sampling functions - which, since they do not decay to zero anywhere, are not strictly bandlimited. In contrast, the DFT produces a bandlimited spectrum - non zero only over a finite range. In a practical situation, discreteness in time demands a band-limited signal, discreteness in frequency a time-limited one. Because of the relationship between a the rectangular function (limited) and the sampling function (unlimited), the two conditions are mutually exclusive. So the DFT can only be an approximation to a Fourier transform.

The DFT approximates the Fourier transform in the same way as the real bandpass filter approximates the ideal, and this approximation can be made accurate by ensuring that the signals involved are limited to all intents and purposes - higher frequencies being attenuated to insignificance. But the question of the time-frequency resolution tradeoff cannot be ignored. It is a fundamental characteristic of the analysis system. In the case of digital analysis, however, efforts can be made to improve some aspects of the resolution at the expense of others. The technique is called *windowing*

By applying a function other than the rectangular to extract a segment of signal, the effects of the abrupt cutting off of the signal can be minimised. The windowing functions used all have a smoother transition to 0 than the rectangular function. The Hamming window function, with its Fourier transform, is shown in Figure 3.4 superimposed over the rectangular function of the previous Figure.

Window functions can be used to reduce the magnitude of the side lobes at the expense of a wider main lobe. The spectrum of a signal treated by a windowing function is composed of a sum of the modified sampling functions, one for each discrete component. A wider main lobe will decrease the resolution of two closely adjacent components, but components further away will be influenced

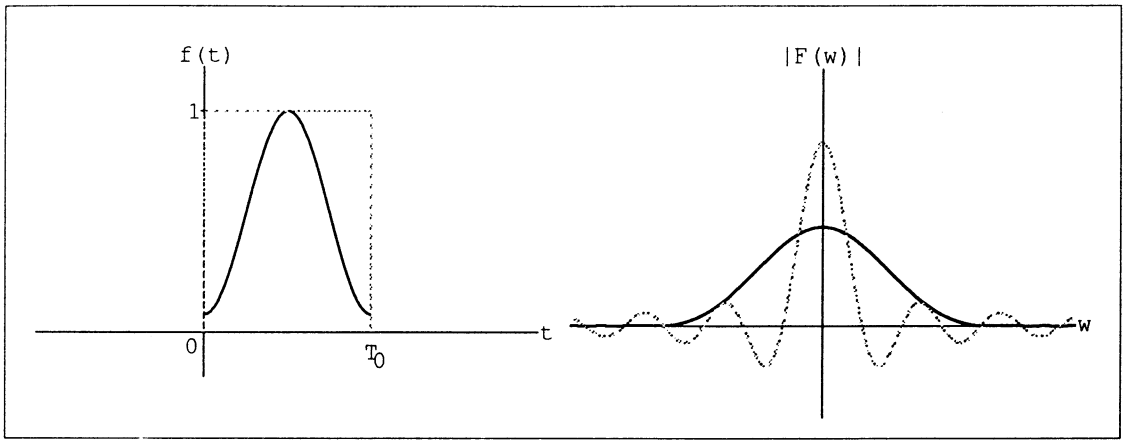


Figure 3.4: Left: A Hamming window, $f(t)=0.54-0.46\cos(2\pi(t-t0)/T_0)$, with the rectangular window of Figure 3.3 overlayed (dashed line). **Right:** Hamming window magnitude spectrum, with the sampling function of Figure 3.3 overlayed. This figure shows how the main lobe of the Hamming window's spectrum is broadened in comparison to that of the rectangular window, while the sidelobes are greatly reduced (they are non-existent at this scale).

to a lesser degree because of the low side lobes. If it is known that all significant components are reasonably far apart from one another, it makes sense to opt for this kind of tradeoff. The components will more truly representative in their magnitude (having had less contributions by the sidelobes of other components), at the expense of the ability to resolve the positions of two adjacent components.

Summary: The Short Term Fourier Transform

The DFT is the discrete equivalent of the bandpass filter as the basic component of a practical frequency analysis system. Like the filter, it is an approximation to the ideal, but a good one under the right conditions. Like the filter, it must be used in a way which trades off frequency and time resolution - the Short Term Fourier Transform. The latter scheme consists of chopping the signal into short time segments, just as the filter bank chops up the signal in the frequency domain. The tradeoff is described in both cases by the relationship between the rectangular and sampling functions, which is essentially an inverse width relationship. It also expresses the implication that a finite (practical) signal in the one domain leads in theory to an infinite signal in the other. Unlike the analogue filter bank, the Short Term transform allows simple manipulation of the nature of the tradeoff with the use of a window function.

3.2.3 The Fast Fourier Transform

The Fast Fourier Transform (FFT) is simply an algorithm for calculating the DFT which has been optimized for operation on a computer. Many variations of the algorithm exist. One of the simplest is the Decimation-in-time Cooley-Tukey radix 2 algorithm, used in this project and thus described here.

The DFT formula of section 3.2.1 is repeated here:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-j(2\pi/N)kn}$$

The algorithm that implements this formula fills a N-element array of complex numbers, **F**, with the results obtained by summing the complex numbers stored in the N-element array **f**, each f(n) being first multiplied by the complex sinusoid $e^{-j2\pi kn/N}$ which equals $\sin(2\pi kn/N) + j\cos(2\pi kn/N)$. There are thus N complex multiplications and additions required for each **k**, or N^2 operations in total (actually there are N-1 additions, but $(N-1)^2 \cong N^2$ for large N). The DFT sum can be split into its even and odd terms as follows (defining $W_N = e^{-j2\pi/N}$):

$$F(k) = \sum_{r=0}^{(N/2)-1} f(2r) W_N^{2rk} + \sum_{r=0}^{(N/2)-1} f(2r+1) W_N^{(2r+1)k}$$

and noting that

$$W_N^2 = e^{-2j(2\pi/N)} = e^{-j2\pi/(N/2)} = W_{N/2}$$

the DFT can be rewritten as the sum of two N/2 point DFTs G(k) and H(k):

$$F(k) = \sum_{r=0}^{(N/2)-1} f(2r) W_{N/2}^{rk} + W_N^k \sum_{r=0}^{(N/2)-1} f(2r+1) W_{N/2}^{rk}$$

$$= G(k) + W_N^k H(k)$$

The number of operations (complex multiplications and additions) that are now required is $2(N/2)^2$ for the two DFTs and N to combine the terms, totalling $N+N^2/2$. For N greater than 2, this is less than the N^2 required for the direct DFT. Thus it is worthwhile to split the two DFTs once again into two smaller DFTs each, continuing until the entire calculation is in terms of two-point DFTs. At that stage, the number of operations has become $N\log_2 N$, which, compared to the original N^2, is the main portion of the savings in computation achieved by the FFT.

A two-point DFT is a pair of equations obtained by substituting the value N = 2, k = 0,1 and n=0,1: F(0) = f(0) + f(1) and F(1) = f(0) + W_2f(1). Equations of this form, ie pairs of linear combination of two variables **x** and **y**, with weighting (**a** and **b**) only on the second variable, can be represented by a "butterfly" flow graph:

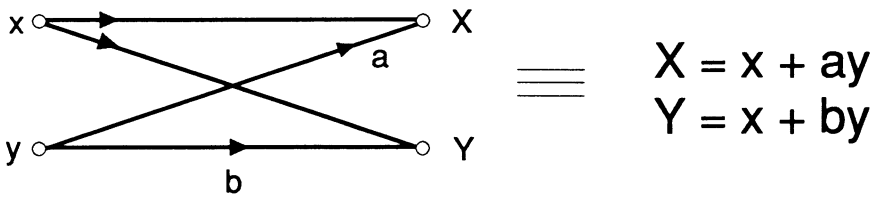


Figure 3.5 below is a butterfly flow graph of an eight-point FFT. The nodes of the graph are storage locations for numerical values. When arranged in the manner shown, it becomes apparent that the results of each butterfly calculation can be stored in the same physical locations as the input values, since the inputs are not used once the butterfly is completed. Thus the FFT's increase in computational efficiency is not at the expense of increased storage requirements. A single N-element complex array is required, storing the f(n) sequence to begin with, and containing the F(n) sequence at the completion of the calculations. In order to take advantage of this storage scheme, it is necessary to store the f(n) sequence in a non-sequential order. The order required is called *bit-reversed* order. It results from repeatedly reordering the sequence by grouping the even- and odd-numbered elements and then joining the two groups, as implied by the decimation-in-time procedure. For example, the splitting process is applied to the sequence of integers from 0 to 7 repeatedly (3 times, since by then only two-point DFTs remain) as follows:

0,1,2,3,4,5,6,7

0,2,4,6,1,3,5,7

0,4,1,5,2,6,3,7

The term *bit-reversed* refers to the fact that element 4 (binary 100, for example, contains the value that was originally stored as element 1 (binary 001), the binary representation of its position having had its bit order reversed.

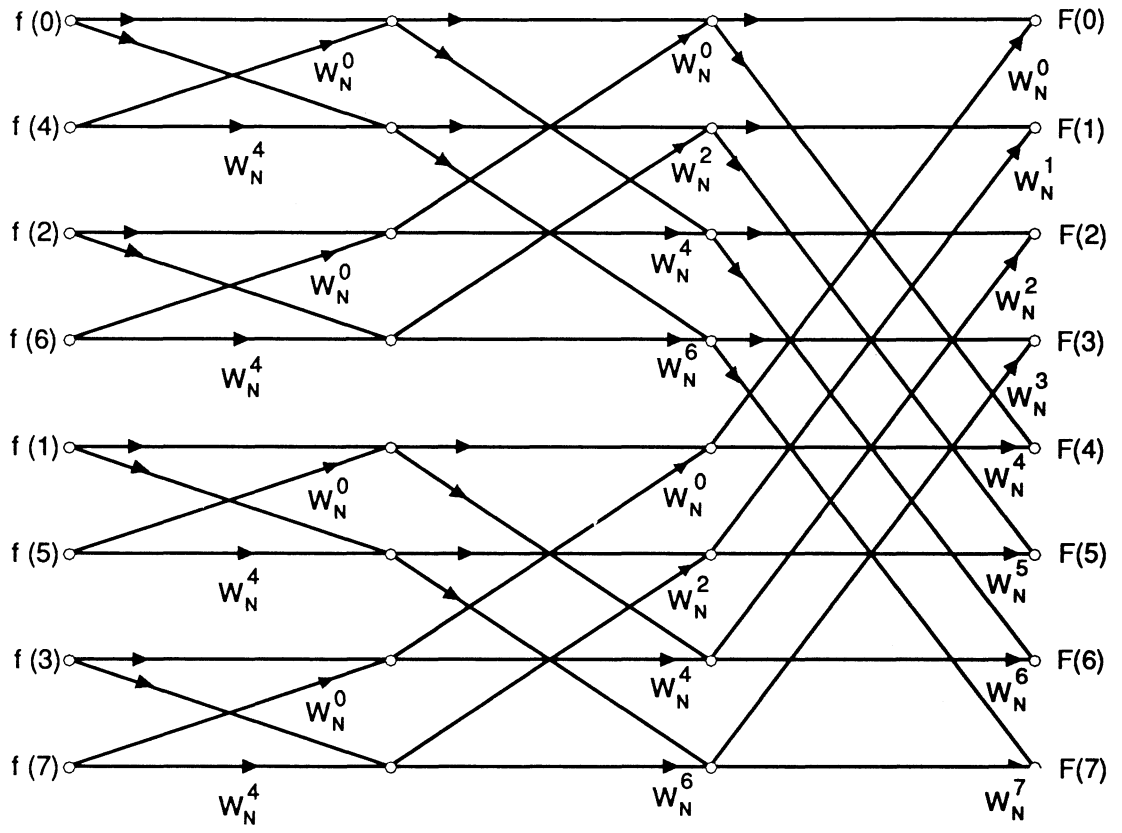


Figure 3.5: Flow graph of complete decimation-in-time decomposition of an eight-point DFT computation, after Oppenheim and Schafer [37].

The number of multiplications in the FFT depicted by figure 3.5 can be halved if it is noticed that all the butterflies represent equation pairs of the form

$$X_{m+1}(p) = X_m(p) + W_N^r X_m(q)$$

$$X_{m+1}(q) = X_m(p) + W_N^{r+N/2} X_m(q)$$

because

$$W_N^{r+N/2} X_m(q) = -W_N^r X_m(q)$$

so that instead of multiplying $X_m(q)$ by two different factors, it is multiplied once and the result reused with its sign negated.

Further savings can be obtained by considering the properties of the computer rather than mathematical behaviour. Special cases often result in reduced computer effort. The cases where W^{nk} = 0 and ±1, if handled separately, eliminate additions or multiplications. If a digital machine is able to perform additions much faster than multiplications, a saving can be made on every complex multiplication (a+jb)(c+jd), since:

$$(a+jb)(c+jd) = (ac-bd)+j(bc+ad) \quad [1]$$

$$= (a-b)d + a(c-d) + j[(a-b)d + b(c+d)] \quad [2]$$

In expansion **1** above, four real multiplications and two real additions are performed. In expansion **2**, the calculation (a-b)d need be performed only once, saved and reused, so that three multiplications are done, and five additions.

Finally, a substantial savings can be obtained by storing the W^r factors in a table rather than invoking trigonometric function evaluations each time a factor is needed. This is especially true when many FFTs are to be performed
one after another, as is the case for a Short Term Fourier Transform of a signal such as a bird call.

All the considerations mentioned in this chapter were used to produce a commercially available FFT routine in the Turbo Numerical Methods Toolbox by Borland International. This is the routine that was used in the SPECTRA process, which is part of the Bird call recognition system software (see below).

3.3 The BCR spectral analysis and display software

The software components relevant to the subject matter of this chapter, namely the digitization of bird calls and their representation in the frequency domain, are as follows; please to refer to Appendix A (software description) and Appendix B (source listings):

SAMPLE Responsible for obtaining the digital samples of the call. Interfaces to the real world, by performing analogue-to-digital conversion, or digital-to-digital conversion of data files obtained independently of the BCR system.

SPECTRA Transforms the sampled data to the frequency domain (performs short-term FFT with windowing).

SONAGRAMConverts the spectral data output by SPECTRA to a form easily displayed graphically.

DRAWER Program for graphical manipulation of the BCR data, including SONAGRAM output.

4

Extracting key information from the call

In the previous chapters, bird calls have been described in terms of their important features, and a framework has been established for representing the calls such that those features will be apparent. The present chapter is the first of two which are concerned with processing the calls in order to find and quantify the features. *Features* are the key descriptive parameters of the data - those parameters which remain constant for calls of a single species, yet differ characteristically from species to species.

It is in terms of features that all but the most primitive of recognition systems work, and a prime example is our own recognition system - the human brain. In listening to a segment of speech, we disregard much of the sound information as being irrelevant. Background sounds and noise, breathing sounds, variations in accent or ill-formed utterances - all may be significant in terms of signal strength, but are effortlessly rejected in our quest to make sense of the signal. Moreover, when we wish to remember, record, repeat or otherwise manipulate the essence of what has been said, it is in terms of the recognised information, not the original signal. We do not attempt to reproduce the exact sound of another's speech - it is enough to repeat (or write down) the words that were spoken. In fact, we do not subjectively perceive the low level details of the signal. We sense the words, and perhaps the intonations, but we are happily oblivious of the frequency components or the amplitude variations. We deal with the abstract speech, not the original raw data.

The same approach will be required on the path to bird call recognition. There is a need to *abstract* from the data, to deal with their essence, in terms of a small number of meaningful parameters rather than a bulk in which meaning is unobtrusively embedded.

What, then, are the features of bird calls that hold the key to their recognition? The BCR system is a means to experiment with possible answers to that question. The choices made for features in later chapters are responsible for the less obvious, more innovative aspects of the system. The present chapter deals with the feature of the call that certainly forms the basis for its recognisablity: The frequency content.

4.1 Significant frequency components in bird calls

While it is clear that the presence and interrelationship of components in the call at each moment in time are what distinguish one call from another, it is not necessarily the case that each and every component plays such a decisive rôle. Possible sources of undesirable or non-significant frequencies include:

- Noise (environmental, or generated in the system hardware)
- Background signals (other bird calls and spurious sounds)
- Components introduced in the signal processing stages

The first two factors listed above are matters of practical importance, which must be taken into account if a recognition system is to work in the field. Noise (such as wind) must be reduced by appropriate methods - shielded microphones for example, and the use of quiet electronic circuitry. Spurious background signals must be avoided in the sample to be recognised, and a practical system might require a means to help the user of the system achieve acceptable input. But for the purposes of an experimental system exploring the issues of recognition, the assumption is made that these factors have been taken care of. More specifically, the noise level in the signal is assumed to be much lower than the signal level (approximately ten times lower, so that an assumed signal-to-noise ratio (SNR) of 10 is used in most of the experiments). The calls are assumed to be uncontaminated by undesirable bird calls or other loud signals.

The third listed factor is one which arises out of the very nature of the transformation to the frequency domain and must therefore be treated as part of the problem of recognition. As explained in Chapter 3, a sinusoidal signal which ideally appears as a single point (or impulse) in the spectrum, becomes a lobe of finite width, surrounded by smaller sidelobes, due to truncation of the sinusoid prior to Fourier transformation.

Thus, as Figure 4.1 illustrates, two pure tones originally emitted by the source (in this case, the bird) would appear to the BCR system merely as the largest components amongst a whole set of newly introduced components at surrounding frequencies.

The extra components carry no useful information about the source signal. The only information of interest (and indeed existing in the source) is the frequency and magnitude of the two source components. The discrete spectrum is a sequence of many frequency and magnitude pairs. The pair designating discrete component closest to the peak of each lobe is the pair which best approximates the source information. The other value pairs are not only redundant but obstructive, in the sense that the recognition system must focus only on the peaks.

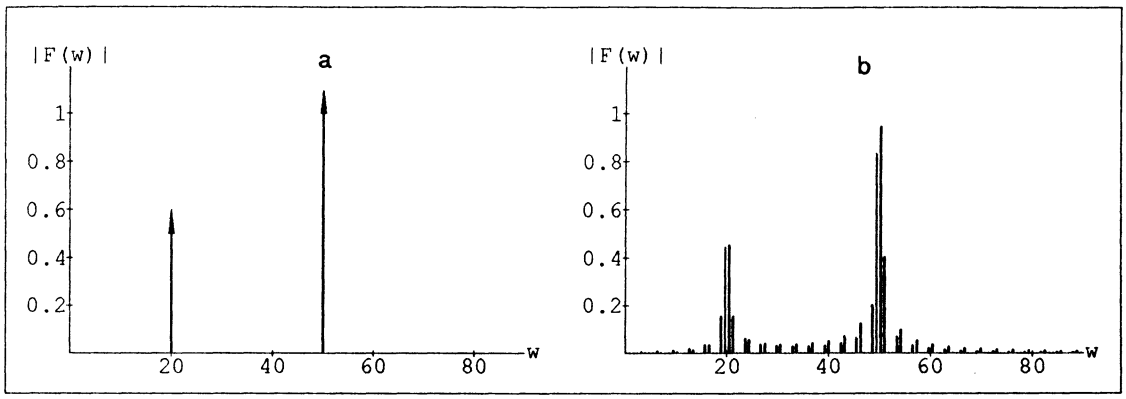


Figure 4.1 a. Ideal spectrum of two pure tones (at frequencies w = 20 rad/s and w=50 rad/s). **b.** Discrete spectrum of a segment of the two-tone signal.

Thus the recognition system's first imperative is to find the position of the peaks in the spectra it obtains. At first this would seem to be a simple task of finding the highest magnitude value in a spectral sequence - except that since more than one peak (i.e more than one component in the source) is in general expected, the task becomes one of finding **local maxima.** The points (*points* is used here interchangeably with *components* in the discrete spectrum) of interest are those with the highest magnitude amongst their neighbours.

If each peak were smooth, detecting the presence of a local maximum would simply be a matter of detecting whether a point's magnitude is greater than that of its immediately adjacent points. Inevitably, however, noise is present in the signal. Even if the level of the noise is much lower than that of the components of interest, as is assumed to be the case, it results in small random fluctuations in the magnitudes at all points in the spectrum. Figure 4.2 illustrates.

Those noise components whose frequencies fall in a part of the spectrum unoccupied by signal components remain small in magnitude. They can be easily ignored with the use of a *threshold.* All points with magnitude below a threshold value simply need not be checked. The value of the threshold to be used is related to the estimate of the prevailing noise level, and the SNR value assumed as explained earlier. If, for example, the background noise magnitude is estimated to be 70 units, and the SNR is assumed to be 10, then it is safe to assume that any component with magnitude of 70, or even up to 0.5xSNRx70 = 350 units arose from noise. A component arising

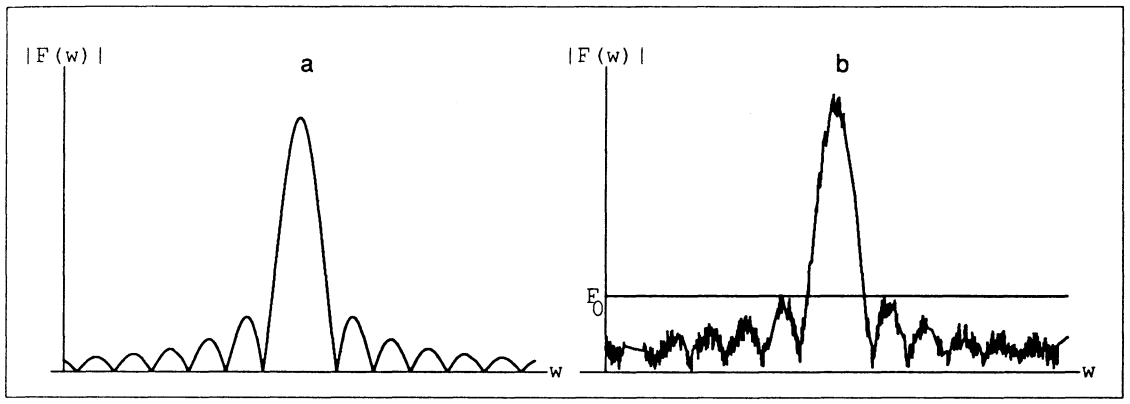


Figure 4.2 a: A smooth magnitude spectrum of an uncontaminated signal **b:** Low-level random noise is added producing many spurious local maxima. Those adding to the signal at magnitudes below a certain threshold (at F_0) are easily rejected.

from the signal (i.e the bird call) is assumed to have a magnitude of about SNRx70 = 700 units. Thus, given a high SNR, the estimate of background level can be conservative, or vice versa, and still allow trouble-free rejection of noise peaks that do not interfere with signal peaks[1].

It is the noise components at frequencies which coincide with components originating in the signal that complicate the process of peak detection. The noise, adding as it does to the "slopes" of the signal peaks, causes small local maxima at magnitudes above the threshold level. An algorithm which examines the spectrum point-by-point will not have a foolproof way of distinguishing between the local maximum that occurs at the top of a peak, and those occurring along its slopes. To do this, we must give the algorithm a more "global" view, in which information from more than just the immediate neighbourhood is used. One way to achieve this would be to smooth the spectrum before looking for peaks. Smoothing involves averaging the neighbouring magnitudes of a point, and assigning the result to that point. Because of the averaging process, random fluctuations caused by noise would tend to cancel each other, with result that their influence is reduced.

The averaging approach has some problems however. It causes some information loss, in effect "blurring" the spectral shape so sharp changes become less prominent. Averaging is useful when the resolution of important features is below that of the averaging window. For example, 7 points

1 The estimate of background noise and the specification of an SNR value are practical steps that must be taken before the peak detection process can begin. They are discussed under BACKGND and PEAK-PICK in Appendix A, which deals with the software for this process.

may be used to calculate an average for the midpoint of the group of 7, in a signal where important variations happen over a sequence of not less than 10 points. All shorter-lived variations might be unacceptably altered by the averaging process. It may be appropriate to use averaging for bird calls, but it would require an analysis to determine likely averaging intervals, based on the characteristics of bird calls and of the expected noise.

The method of peak extraction chosen and described below does not require smoothing before application, but neither does it preclude the use of smoothing. It might return marginally better results should smoothing be used, but is satisfactory on its own. It has the advantage of improving with increased frequency resolution used in the FFT, so that quite accurate results can be obtained at the price of greater processing. Because it is tailored exactly to the bird call data, it has certain other advantages in the context of the BCR system, as will be explained.

4.1.1 Peak extraction algorithm

The peak extraction algorithm used here takes advantage of the knowledge of how the spectrum was produced. Since the length and type of the window used in the short-term Fourier transform is known, the shape of the lobes produced when a pure sine wave is transformed is also known. It is given by the Fourier transform of the window function. The algorithm attempts to locate the original bird call components, by guessing their positions and then removing their contributions to the spectrum. The point with the largest magnitude in the spectrum is used as the guess of a component position. The spectral shape that would occur if that component were indeed present at that position is then subtracted from the magnitude spectrum, leaving a new "largest magnitude" which can then be used for another guess. Figure 4.3 illustrates the peak extraction mechanism. The process continues until at most 5 peaks above the threshold are found. The number 5 is a minimum determined by preliminary consideration of the classification scheme of chapter 5, see section 5.1.1, and could conceiveably change in the wake of further analysis or experimentation. The threshold value is a function of assumed signal-to-noise ratio (SNR) and the measured average background noise level - in the BCR system software it is simply the product of the two.

The above algorithm is implemented by the PEAKPICK process, described in more detail later in Appendix A. An analysis of the applicability of the algorithm occupies the remainder of this section.

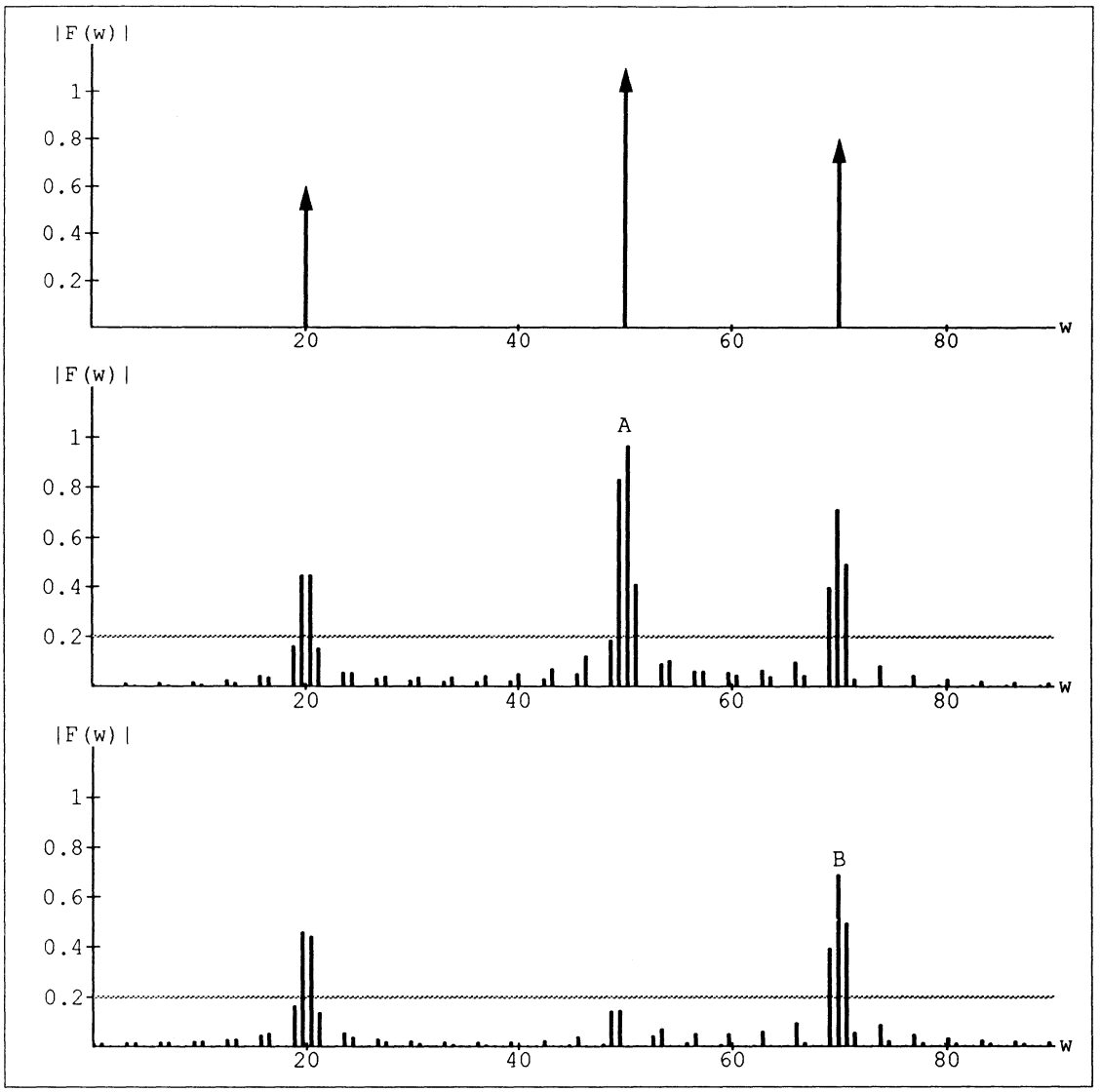


Figure 4.3 Guessing the positions of source components. **Top:** An ideal source spectrum containing three components. **Middle:** The spectrum seen by the BCR system - point A is the largest magnitude, and gives the approximate position of the largest source component. **Bottom:** After subtracting the spectral shape due to a single source component at position A, the largest magnitude in the spectrum is now at point B, which gives the position of the next source component. The horizontal faint line represents a threshold below which peaks are not considered.

4.1.1.1 Interfering peaks

When two components in the source signal are close together, the lobes they produce overlap and add together to give a different shape (see Figure 4.4). It is not the job of the peak extraction algorithm to attempt to resolve such a compound shape into its original component peaks. This resolution problem is the same fundamental limitation on signals whose frequency content is changing that was discussed in chapter 3. Any receiving apparatus, whether machine, human or bird, will have a limit to its ability to separate two close-together tones. In the BCR system, this ability may be traded off with the ability to react to rapid changes in the sound, by changing the frame (window) length parameter in the SPECTRA process.

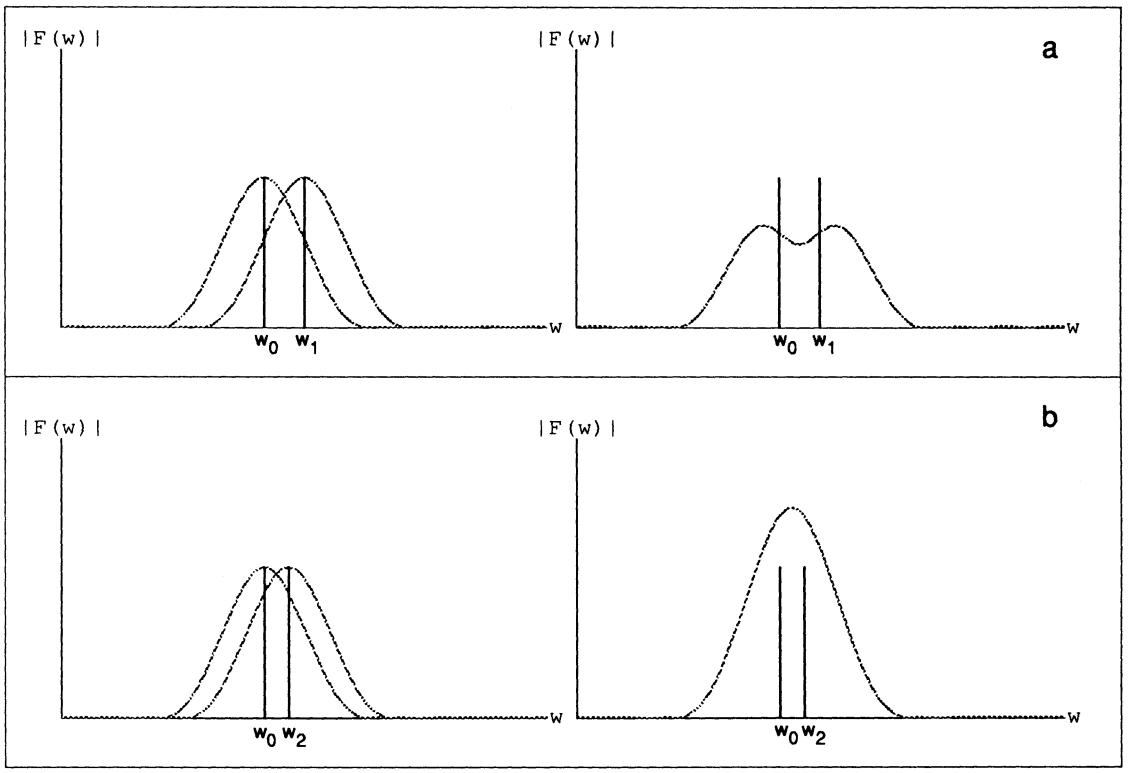


Figure 4.4 Two close-together peaks (left) add in the complex plane to give a new shape (right). **a:** The peaks are far enough apart for separate local maxima to be evident, displaced outwards slightly from the original peak positions. **b:** The peaks are so close that they add to produce a single, larger peak positioned in between them. The vertical lines mark the original peak positions at w_0, w_1 and w_2.

However, as Figure 4.4 shows, the compound shapes are different to those produced by isolated source tones. The danger, therefore, is that in subtracting single-tone shapes, the algorithm will imperfectly remove the influence of the tone pair. The spectrum will be left with meaningless, but possibly obtrusive, "chunks" of leftover lobes. It necessary to calculate when such a problem is likely to occur, so that its seriousness might be gauged.

Ignoring the outward-shifting effect[1] of Figure 4.4a and concentrating on the case of a resultant single lobe, (4.4b), the point at which the individual identity of the peaks begins to be lost is when the midpoint between them has a height equal to that of the peaks themselves. That midpoint is the sum of the contributions of the two lobes, which are equal, so that the individual lobe magnitude at that point is half the peak magnitude. The point in question is shown in Appendix D to be **5.2707/T radians/sec** from the peak position for a Hamming window of length **T**. Converting to Hz, the half-magnitude point is $\mathbf{5.2707/(2\pi T) = 0.839/T}$ **Hz** approximately from the centre. Thus peaks must be at least twice this distance from one another, or 1.678/T Hz, in order not to interfere. For the common case of **T=5 ms,** this means **335 Hz.**

Before concluding on this result it should be noted that the above estimate is rather conservative. Firstly, 5 ms is close to the minimum window length used in bird call analysis. The fastest bird call events do not occur over time scales shorter than about 2 ms (see section 2.1.2). Window length times of 10 ms still yield intelligible sonagrams, and would halve the 335 Hz figure. Secondly, the case analyzed is that of two equal-valued components in phase with each other. If one component is greater in magnitude than the other or if they are not in perfect phase (as would be true for a majority of cases), they would have to be closer together before their positional information was swamped by the resultant new peak. Finally, if a window with a narrower main lobe is used, the critical distance is once again reduced. One of the strong points of the peak subtraction algorithm is its ability to handle the case of significantly high sidelobes (since the main lobe should always be found first and the sidelobes subtracted together with it before being "seen"). Thus it is not unrealistic to consider using a window which allows higher sidelobes in return for a narrower main lobe.

All that remains to be noted is the likelihood of peaks being close enough to interfere. For calls that are periodic in nature, being generated by an oscillating source, the minimum possible distance between spectral peaks equals the fundamental frequency of the oscillation. The lowest frequency in calls of this type is observed to be about 500 Hz, as mentioned in section 2.1.2 in reference to the hearing capabilities of birds. A small number of birds may be able to produce (and recognize)

1 The error evident here is not considered in the BCR system. To do so would require a more sophisticated peak extraction algorithm, one capable of iterating to determine the accuracy of its own guesses. The arguments to follow, which play down the effect of two peaks interfering to produce a single lobe, also apply in most part to separate peak position errors.

calls with lower frequencies, perhaps around 300 Hz. We see therefore that peak interference ought not to be a problem in the vast majority of cases. It may be argued that the group of birds known to produce lower-frequency calls, namely owls, are prime candidates for a call-recognition system due to their nocturnal activity. It would be a simple matter to allow for different analysis parameters (window type and length) for these specialised conditions of observation.

As for the case of aperiodic bird sounds, such as rasping, hissing and clicking - we would expect to find many spectral peaks, often very close together. However, as will be explained in the next chapter (on the classification of bird sounds), it will not be necessary to know the exact position of these peaks. The peak positions in themselves do not contain much, if any, recognisable information. Only their presence and approximate distribution over the spectral range is important.

4.1.1.2 Sampling error

The interference between peaks is an unavoidable effect arising from the nature of real-world frequency analysis. A second possibility for error arises out of the practicalities of digital processing. Because the spectra in the system consist of discrete sample points, the exact position of a peak can not usually be found. For example, if the sampling frequency is 16 kHz, and the FFT is 128 samples in length, each sample lies on a 16000/128 = 125 Hz boundary. If a component in the bird call occurs at 600 Hz, the peak position will lie between the points at 500 Hz and 625 Hz. The latter point will be identified as the maximum (being closer to the true maximum). This error is acceptable, unavoidably part of the discrete analysis. But the peak subtraction algorithm will subtract from surrounding points as if the peak was **centred** on the 625 Hz point, so that the influence of the peak in the spectrum is not correctly eliminated (Figure 4.5).

If the point chosen is far enough from the peak position, the remains of the lobe after subtraction could be large enough in magnitude to cause a second peak to be postulated. The effect is that a pair of points are reported by the algorithm where only one component exists.

A number of ways exist to ensure that this "twin peak" error does not occur. The simplest is to increase the number of sample points in the FFT. This reduces the possible error in peak location because the distance between successive points in the spectrum is reduced. The actual and guessed positions overlap to a greater degree, so the erroneous remains of the lobe are smaller in magnitude and fall below the threshold of peak significance. Figure 4.6 below compares the performance of the algorithm for two different FFT lengths.

calls with lower frequencies, perhaps around 300 Hz. We see therefore that peak interference ought not to be a problem in the vast majority of cases. It may be argued that the group of birds known to produce lower-frequency calls, namely owls, are prime candidates for a call-recognition system due to their nocturnal activity. It would be a simple matter to allow for different analysis parameters (window type and length) for these specialised conditions of observation.

As for the case of aperiodic bird sounds, such as rasping, hissing and clicking - we would expect to find many spectral peaks, often very close together. However, as will be explained in the next chapter (on the classification of bird sounds), it will not be necessary to know the exact position of these peaks. The peak positions in themselves do not contain much, if any, recognisable information. Only their presence and approximate distribution over the spectral range is important.

4.1.1.2 Sampling error

The interference between peaks is an unavoidable effect arising from the nature of real-world frequency analysis. A second possibility for error arises out of the practicalities of digital processing. Because the spectra in the system consist of discrete sample points, the exact position of a peak can not usually be found. For example, if the sampling frequency is 16 kHz, and the FFT is 128 samples in length, each sample lies on a 16000/128 = 125 Hz boundary. If a component in the bird call occurs at 600 Hz, the peak position will lie between the points at 500 Hz and 625 Hz. The latter point will be identified as the maximum (being closer to the true maximum). This error is acceptable, unavoidably part of the discrete analysis. But the peak subtraction algorithm will subtract from surrounding points as if the peak was **centred** on the 625 Hz point, so that the influence of the peak in the spectrum is not correctly eliminated (Figure 4.5).

If the point chosen is far enough from the peak position, the remains of the lobe after subtraction could be large enough in magnitude to cause a second peak to be postulated. The effect is that a pair of points are reported by the algorithm where only one component exists.

A number of ways exist to ensure that this "twin peak" error does not occur. The simplest is to increase the number of sample points in the FFT. This reduces the possible error in peak location because the distance between successive points in the spectrum is reduced. The actual and guessed positions overlap to a greater degree, so the erroneous remains of the lobe are smaller in magnitude and fall below the threshold of peak significance. Figure 4.6 below compares the performance of the algorithm for two different FFT lengths.

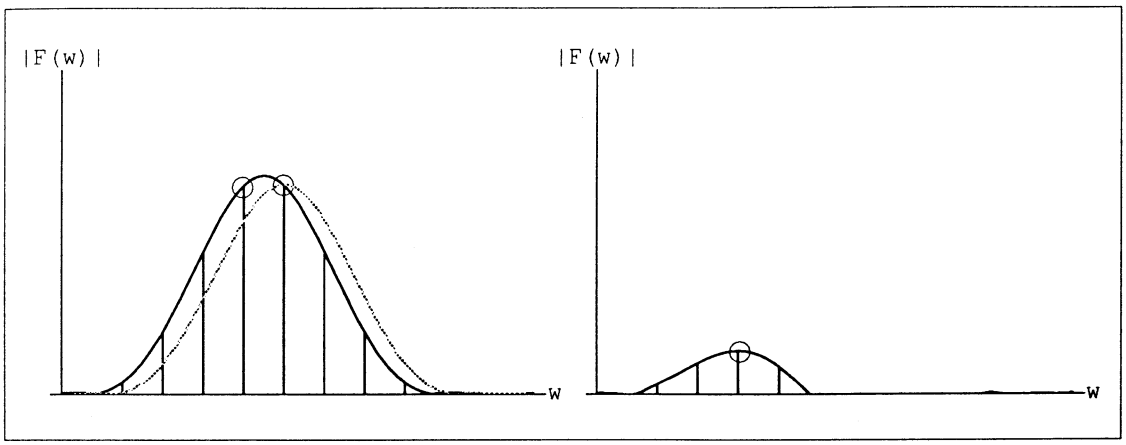


Figure 4.5 Peak subtraction error. **Left:** The bold curve shows the actual lobe position, and the vertical lines mark the sample point positions. The two highest sample points are marked with open circles. The faint curve is the lobe which will be subtracted by the algorithm (i.e, the lobe guessed by using the highest sample point). **Right:** The result of the subtraction (negative point values are forced to 0). Another point (circled) may remain at a high enough magnitude to look like a second peak.

It is important to note that increasing the FFT length does not require an increase in the **window** length (see the discussion of *padding* under SPECTRA, in Appendix A). The increased resolution in the spectrum due to more FFT points simply gives a closer approximation to the true (continuous) Fourier transform, allowing better location of the true peak. Adding window points would change the width of the peaks themselves, and reduce the time resolution of the system.

Increased FFT length is used in the BCR system to avoid twin peak errors, but FFT processing times increase as $N\log_2 N$ where N is the number of FFT points. In a practical system it might be better to try to eliminate the error at the peak extraction stage. One approach would be to detect the occurrence of a twin peak, to estimate the true peak position and to then redo the peak extraction procedure for the spectrum. This is theoretically viable because as discussed above, the minimum distance likely between peaks is of the order of 300 Hz, while the example given above of 125 Hz between points is close to a maximum figure[1]. Thus two peaks as close as 125 Hz are in all likelihood erroneous rather than reflecting the true situation. The estimate of the true peak could be based on

1 The number of points used in the FFT is likely to be greater, rather than less than 128, which provides only 64 useful points. The value of 16 kHz for the sampling frequency is the maximum required, because frequencies higher than 8kHz are not expected in bird calls.

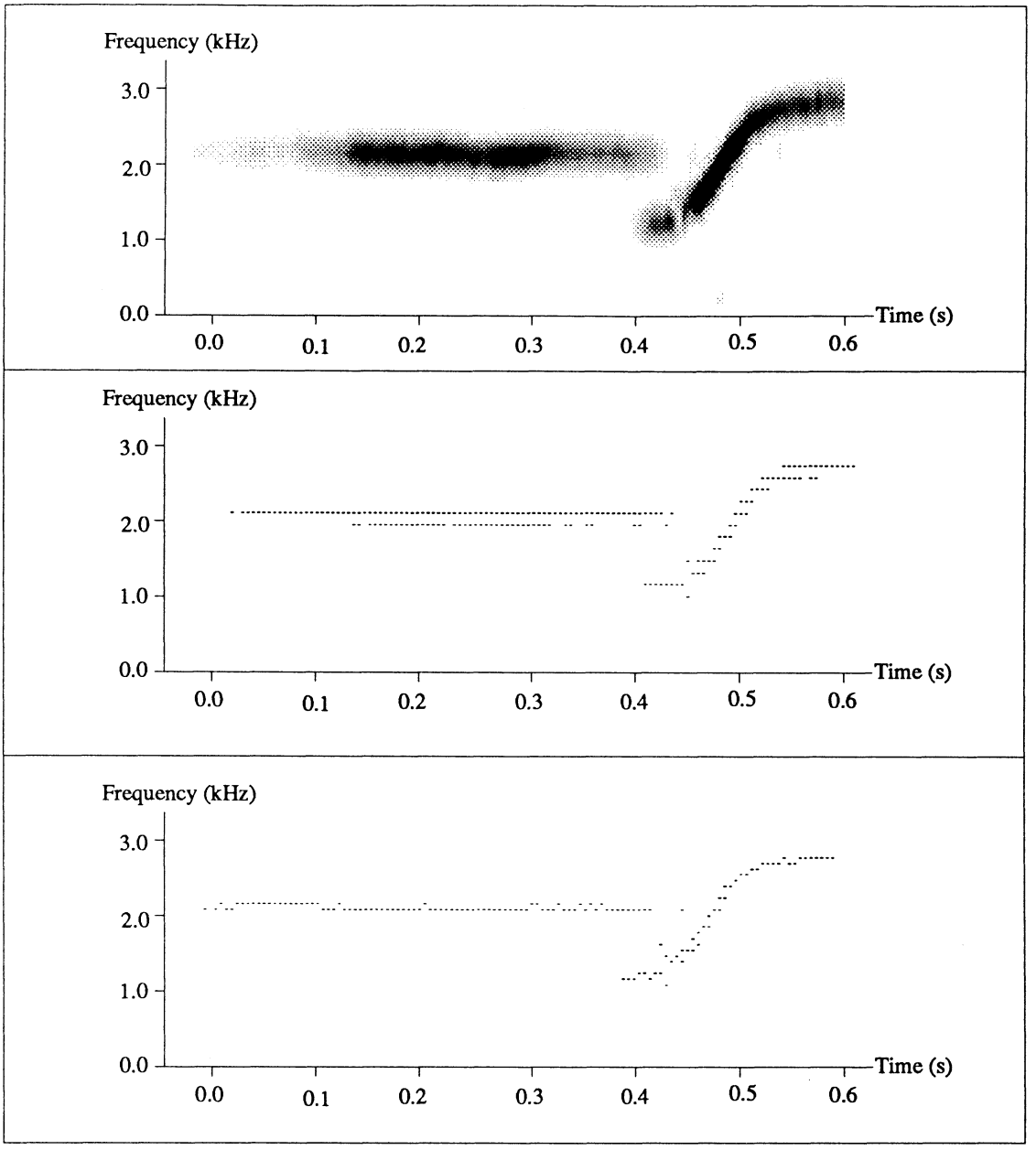


Figure 4.6 "Twin peak" error for a segment of bird call containing a single component. **Top:** Songaram of a portion of the call of the Southern Boubou (*Laniarius ferrugineus*), the first element being a singe tone. **Middle:** Peaks extracted from 128-point spectra - the height of a point indicates the position of a peak - many frames exhibit twin peak error. **Bottom:** Peaks extracted from 256-point spectra contain no errors, having only a single component wherever appropriate.

the relative magnitudes of the two points (before subtraction). Although this method would require repeated extraction on each frame, the operations involved would probably be less, and faster, than the FFT operations.

Another source of information on the existence of error is the occurrence of negative results at the peak subtraction stage. If a lobe is being subtracted at a position where a peak did not actually exist, as occurs after the second peak in twin is found, some of the point-by-point subtractions could yield negative results. This would be an indication that the position is not that of a true peak. The spectrum could then be reprocessed as above.

The ideas outlined here would have to be tested to ensure that they are reliable. Although it is true to say that a peak's position could be guessed more accurately, based on adjacent point magnitudes and the absence of adjacent points in "wrong" positions, it is not easy to quantify just how effective the estimate would be, given that other peaks are present and adding to the relative magnitudes of surrounding points. Increasing FFT points is a simple, reliable and general solution that is adequate for experimental purposes.

4.1.2 Evaluation: Reproducing calls from extracted peaks

The identification of peak positions, while possibly necessary in any general quantitive analysis of bird calls, is a fundamental part of the particular method of describing bird calls used in the BCR system. To evaluate the concept's effectiveness in this capacity, the entire BCR system must be completed and it's performance tested. Rather than wait for that stage before attempting a validation of the peak extraction technique, it was decided to reconstruct a time waveform from the extracted peak data. This waveform could be processed like a bird call to produce a sonagram or audio output, and that compared with the original call.

Reconstruction of the calls is a simple process. Each peak identified in the extraction stage specifies the frequency and amplitude of a sinusoid present for a short interval at a particular moment in time. Sinusoids are generated algorithmically for the required duration, using a sampling period equal to that of the original signal (although a different rate could be used). All the sinusoids present at a particular interval are summed sample by sample. The segments of time waveform thus generated, each one corresponding to a consecutive frame, are joined. The SOUNDGEN utility described in Appendix A performs this task.

Comparing reconstructed calls is a subjective procedure, since no independent method of quantifying bird calls is available. The sonagrams are simply visually compared to determine if they seem to contain the same "recognisable essence". The audible playback is similarly subjectively evaluated for recognisability. Figure 4.7 shows sonagrams of a representative sample of different

types of bird call, with their corresponding peaks as extracted by the PEAKPICK process (described in Appendix A), and the sonagrams formed by first reconstructing the time waveforms from the peak data and then feeding this data to SPECTRA and SONAGRAM. The reconstructed waveforms were also played back through digital-to-analogue conversion hardware and heard by members of the interested public.

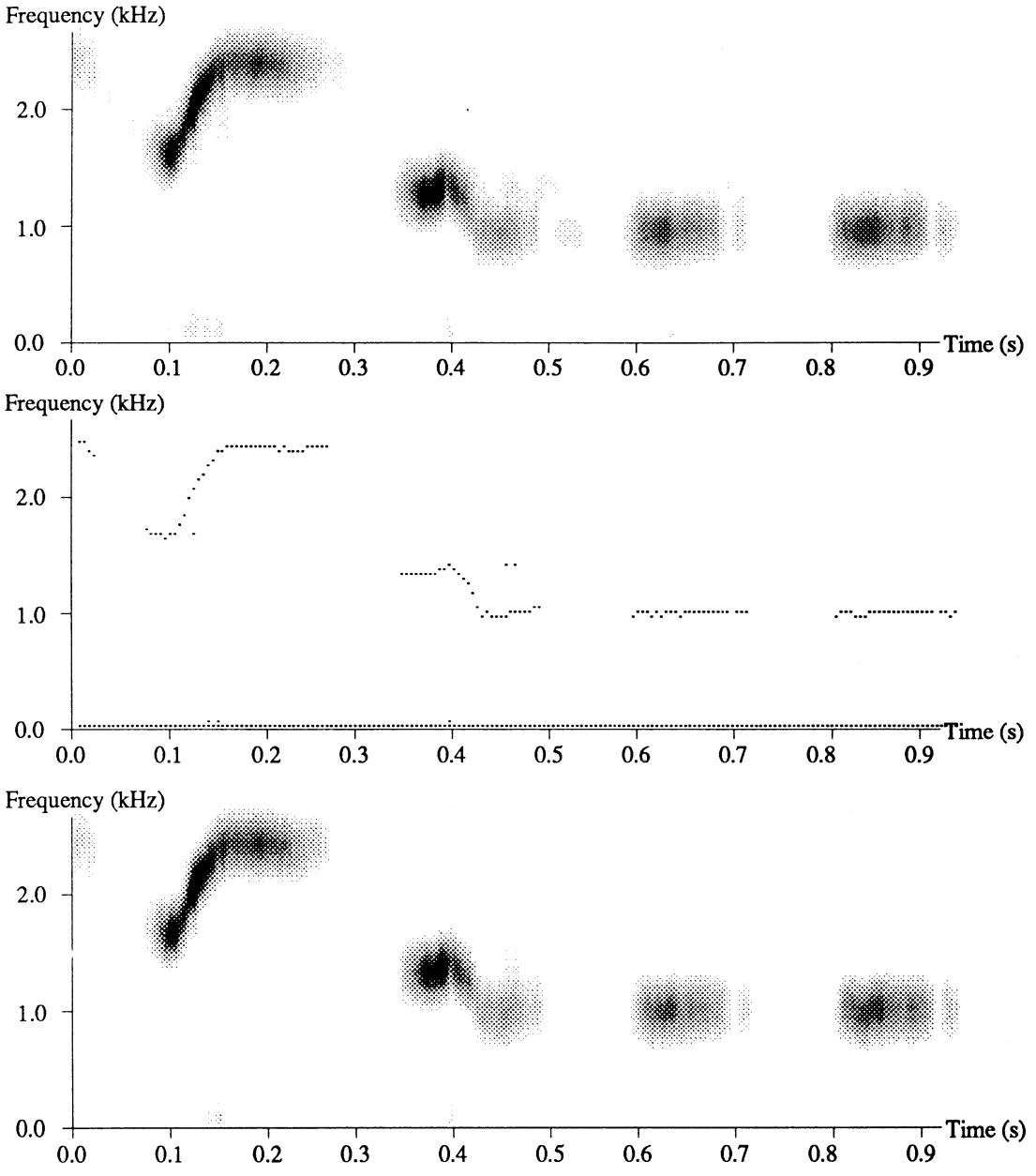


Figure 4.7a Reconstruction comparison for part of the call of the Southern Boubou (*Laniarius ferrugineus*). **Top:** Original call. **Middle:** Peaks extracted from original call. **Bottom:** Reconstruction from peaks.

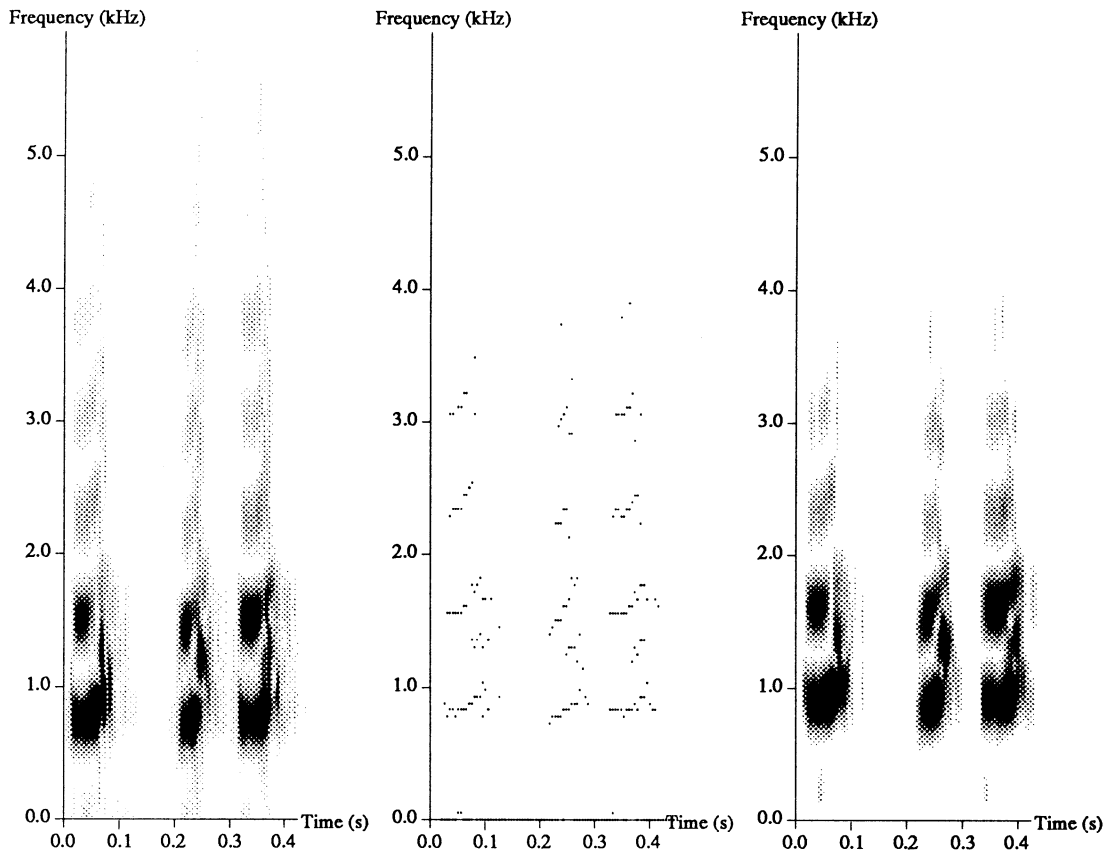


Figure 4.7b Reconstruction comparison for part of the call of the Redbilled Hornbill (*Tockus erythrorhynchus*). **Left:** Original call. **Centre:** Peaks extracted from original call. Note that linked[1] peaks were used alhough the links are not shown here. **Right:** Reconstruction from peaks

The results of the comparison were (subjectively) positive, in that reconstructed calls retained their unique character, and in some cases were almost indistinguishable from the original. As expected, calls with a smaller number of peaks were affected very little, while noisy calls such as that of the

1 See Appendix A, SOUNDGEN and LINKPEAK for a discussion on links.

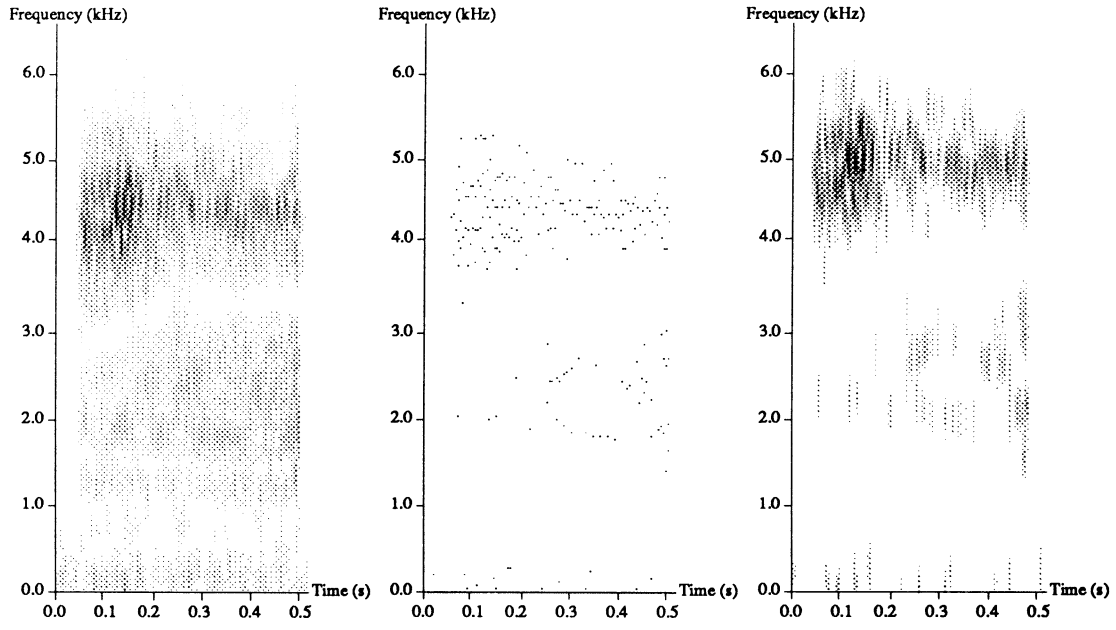


Figure 4.7c Reconstruction comparison for part of the call of the Fiscal Shrike (*Lanius collaris*). **Left:** Original call. **Centre:** Peaks extracted from original call. **Right:** Reconstruction from peaks.

Fiscal Shrike (Figure 4.7c) suffered to a greater degree due to the limiting of the number of peaks. Even noisy calls remained recogniseable, and increasing the peak number for such calls is a viable way to improve the procedure.

The positive results point the way to the possibility of using the peak data as a form of bird call storage, instead of simply storing sampled output directly from the SAMPLE process. The advantage of doing this is the smaller size of the peak data. The disadvantage, besides possible imperfect reproduction, is that the audio signal must be reconstructed as described above each time the call is to be played back.

While the efficient storage and reproduction of bird call's is not a primary goal of this research, it was thought worthwhile to determine the extent of savings that would be possible through the use of peak data. Bird call reproduction on a hand-held device (as outlined in Chapter 1) is a highly desirable feature, technologically far less complex than recognition, yet going a long way towards enabling the increased utilization of bird calls in field identification. In order to be practicable on a hand-held device, the sound data must however occupy a minimum of space. As a rough estimate, the following calculation shows what the digital storage requirements of a full set of South African bird calls might be, without any form of compression.

Sampling rate: Maximum 16 kHz, average is less, assume 10 kHz
Number of bytes per sample: Minimum 1.
Number of seconds of sound per bird: For a realistic impression, a longer segment would be required than shown in the sonagrams in Roberts[2], for example, but possibly less than is recorded by Gillard[39]. If silences are deleted (to be filled in at playback time), 5-10 seconds may be enough. Assume 5.
Total bytes per bird: (10000)(1)(5) = 50000 or 50 kb.
Number of South African birds requiring voice reproduction: Out of a total of approximately 800 Southern African birds, assume that half, or 400, can be better recognized when their call is known.

Total bytes required: (400)(50) = 20 000 kb or 20 Mb.

While the above is perhaps too sketchy a calculation, it serves to illustrate the order of storage space required, which is very large for a hand-held device. Although non-volatile memory technology exists that would be of a small and light enough scale to suffice, it is currently state-of-the-art and expensive. Now consider the storage required by peak data:

Number of peaks per frame: From considerations elaborated in the next chapter, the number of peaks required for recognition purposes to extracted from each spectral frame is no more than 5. For reproduction purposes some calls would benefit from the use of more than 5 peaks, but many (probably a majority) of calls actually have less than 5, so that the average number required is definitely no more than 5.
Number of bytes per peak: Using the same estimate as previously for the amplitude quantization, we need 1 byte for the magnitude of a peak. No tests have been performed to determine the effect of frequency resolution on the reproduction quality, but sonagrams using a an FFT length of 256 seem more than adequate, so that 1 byte for peak position should suffice. Thus a total of 2 bytes per peak is predicted.
Number of frames per second: The quality of the sonagrams and reconstructed audio signal would indicate that a frame length of 5 ms is quite acceptable - even 10 ms would not be out of place for many calls. Assuming 5 ms gives 200 frames per second.
Total bytes per second: (5)(2)(200) = 2000 or 2 kb/s.

Compared with the figure of 10 kb/s estimated for the raw data (see "Sampling rate" and "Number of bytes per sample" above), the peak data figure of 2 kb/s means only 20% of the storage space is required. The parameter (5 peaks per frame) which gives rise to the 2 kb/s figure was used in the reconstruction trials. It is far greater than that required for some calls, and still yielded intelligible results for calls which strictly needed more than 5 peaks per frame.

Having estimated a reduction to 20%, an actual comparison of data file sizes is presented in Figure 4.8. First, the raw sample and peak data files were compared, showing an average size reduction to 30%. However such a comparison does not take into account the reductions possible for the peak data mentioned in the above estimate breakdowns. There, frequency and amplitude values were quantized so that a single byte could be used for each value, and mention was made of the fact that some calls would be able to use less than 5 peaks per frame, or lower sampling frequencies. In contrast, the files used in the BCR system waste space by using 2 bytes per sample (12 bit A/D with 4 bits unused) and 6 bytes (Turbo Pascal reals) per amplitude and frequency value. The number of peaks per frame is fixed at 5 and the sampling frequencies used were all in excess of those required.

In order to estimate what the effects of optimizing the file formats and processing would be, a blanket data compression algorithm was applied to both the sample and peak data. This has the effect of removing redundancies and giving a realistic prediction of the optimizations possible. If the "manual" optimizations do not result in reductions similar to those obtained with compression, the compression algorithm itself could be put to use. The compression software used was PKZIP (v1.1), a public-domain package utilizing a Shannon-Fano compression algorithm. Figure 4.8 shows that when comparing the compressed peak data to the compressed sample data, a very big average reduction to about 6% is evident.

The sample of bird calls used was chosen to be representative of different types of sounds, but is not quantatively representative of the proportions of these sounds found in nature. Therefore, the figure of 6% is not reliable as an indication of the overall savings that might be possible on a hand-held device. The experiment does however provide an indication of the minimum (3%) and maximum (12%) reductions likely. The calls used are identified in the figure by a four letter code, which can be used to find the data for that call in Appendix F. By comparing the sonagrams with that of an untested call and referring to 4.8, a good prediction can be made of the size of the compressed peak data for the untested call.

In order to for a device to take advantage of the savings, the processor must be able to convert the processed stored data into sound within a reasonable amount of time. Since raw data can be fed directly to a D/A convertor, it is the form which can be played back most promptly. In contrast the peak data must be reconstructed, and if some compression is used, must be decompressed. While this is not the place to consider in detail how fast the reconstruction could be done in practice, it is worth noting that a limited processor such as would reside on a hand-held device might take uncomfortably long about it. The possibility of dedicated hardware to perform the reconstruction should be considered, with the emphasis being on low cost and size. Below is an outline of two such hardware configurations that could be used. Their viability is currently under investigation in an independent project.

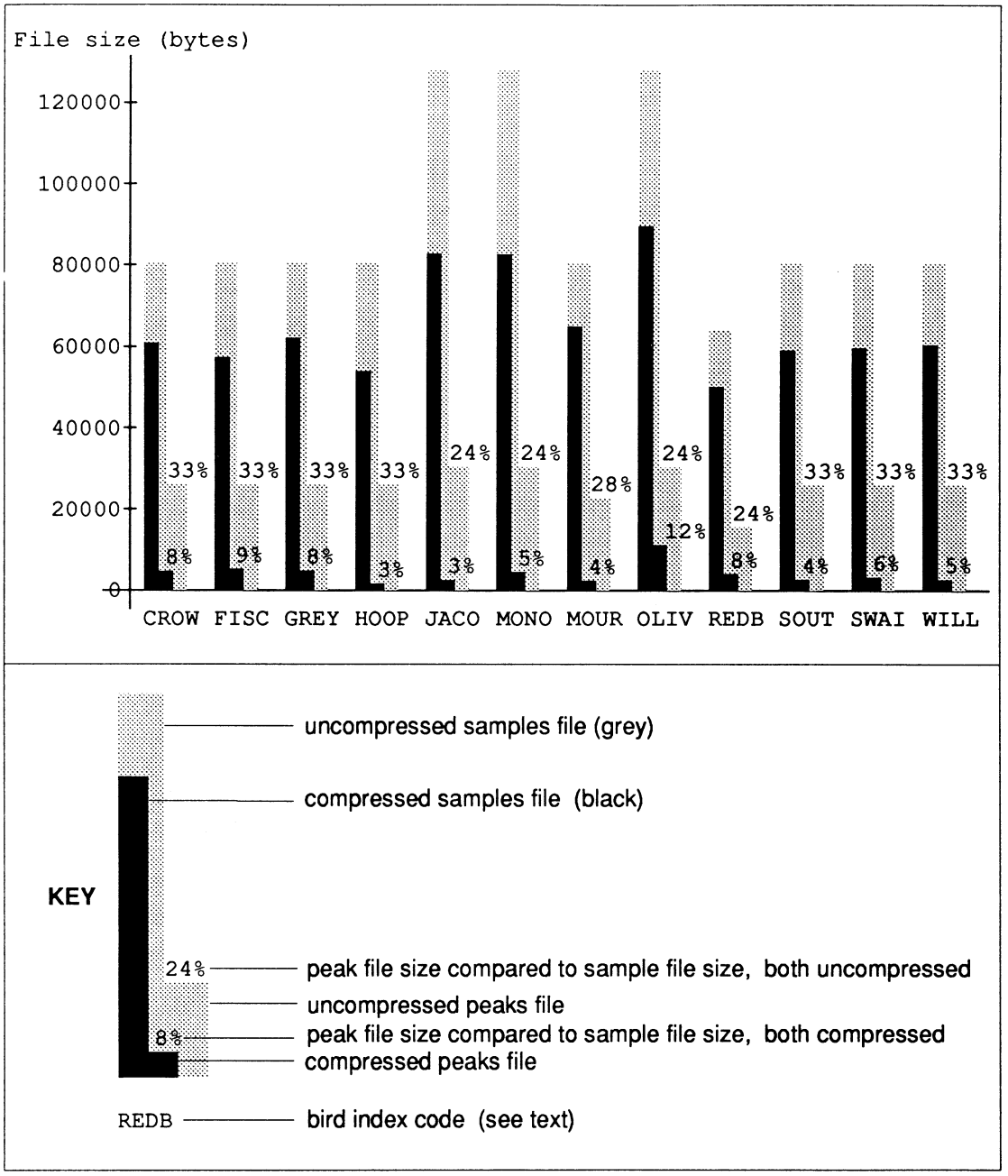


Figure 4.8 Comparison of file sizes of sample and peak frame data. Grey bars represent uncompressed data, black bars represent compressed data. Percentage figure is (peak file size)/(sample file size) x 100.

Discrete analogue reconstruction hardware

The relative simplicity of the reconstruction task means that common building-block components could be used to achieve it. The low complexity level of the components would probably imply low overall cost of this solution. Briefly, each peak would need to be handled in parallel by a sine-generator circuit. The sine-wave would be generated by a voltage controlled oscillator, whose frequency is determined by the peak frequency value. The sine wave would be amplified by a variable-gain amplifier, which would not be required to have particularly stringent operating characteristics. The peak magnitude value would control the gain. The sinusoids originating from each peak would be added together. Some interface circuitry to allow peak data to reach the circuit from the microprocessor sub-system, a simple digital-to-analogue conversion scheme and output gain control would complete the system.

Commercially available integrated circuit

IC units exist which can produce a number of simultaneous tones under the control of a micro-processor. These units are typically used for simple music synthesis in low-cost applications. It is possible that such a device could have, built-in, the ability to produce the required number of simultaneous tones (5), and to vary the frequency and amplitude with the required accuracy. The peak data could then drive such a device with a minimum of preparatory processing.

4.2 The BCR peak extraction software

A number of different software components support the peak-extraction stage of the BCR system processing. The central component is the pipeline extraction process itself, PEAKPICK. The latter requires an estimate of the average background noise level, and this is provided by the BACKGND process. DRAWER has a section which produces a graphic display of peak data, and is also responsible for initiating the BACKGND process. Finally, LINKPEAK and SOUNDGEN are utilities used in reconstructing a bird call from it's peak data. The descriptions of these compenents are to be found in Appendix A and Appendix B.

5
Classifying bird sound primitives

In the previous chapter, it was clear that peaks in the frequency spectrum are the key to deciphering the content of a bird call (as they are to deciphering the content of human speech), and so they were found and made accessible by being separated from the raw signal. Now the time has come to examine this set of peaks to determine what recognizable information they contain. What patterns emerge, and in what ways can the patterns be characteristic of particular bird species?

One possible approach to the problem of pattern detection in data is an empirical one. The data could be analysed statistically, by searching for correlations and similarities. The approach would be valid for a set of as yet meaningless data, of which exhaustive samples were available. However it is an undirected approach which is not guaranteed to succeed. For bird calls, it is inappropriate since large amounts of data are not readily obtainable. Although commercially available tapes have examples of all the calls likely to be encountered in the field, they do not contain a wide variety of samples per species. Such data are not entirely unavailable, but would require extensive effort to collect and process. Given that the alternative of a directed pattern search is applicable, it is much preferred.

The directed approach is based on an underlying knowledge of the abstract characteristics of bird calls. Such knowledge is the result of specific research such as cited in Chapter 2, and of the collective experience of ornithologists as condensed in an authoritative reference such as [2]. What this means is that there exists a basic understanding of the patterns and features that appear in bird calls, and that this understanding need only be related to the data to be used in an automated system. By searching for known patterns in the data we forfeit the chance of discovering new, previously unknown features. That price is seen to be low in this context, since the recognisable features of bird calls would appear to be largely apparent to the knowledgeable observer, and not hidden in some sort of "secret" avian sensory or cognitive function. In return, we benefit from a short and direct path to the extraction of meaning from the data, without which this project would likely not have been undertaken.

By considering the body of knowledge of bird call characteristics, the following conclusions about their form can be drawn:

- A bird call is made up of a time sequence of separate utterances, or elements.
- Elements are differentiated by their acoustic quality, or by their separation in time.

- Both the acoustic quality of an element and the position in which it occurs in the call sequence are characteristic of a particular species, with the importance of one factor or another varying between species and families of birds.

- The acoustic quality of an element is defined by its spectral content, and can be roughly grouped into a small number of classes. These classes correspond to the different modes and mechanisms which the bird uses to produce the sound.

- Elements of the same class can have different acoustic quality, with this difference being defined by the values of parameters, such as fundamental frequency, that are pertinent to that class. Such elements are nevertheless related in overall form.

The last two points are the concern of this chapter - how to differentiate, classify and characterize bird call elements. In the next chapter the factor mentioned in the third point, the order of combination or *syntax* of the elements, will be dealt with.

5.1 Classification of element contents

The contents of an element are defined here as a sequence of peak frames. That is, an element will not be defined at the level of the time signal, even if such definition might lead to a more accurate determination of the element boundaries. It is the peak frames that contain the most meaningful information at this stage, and which the system is committed to using as the means of bird call recognition. It should however be noted that if only the amplitude of the signal is used in determining element boundaries (as is the case here, discussed later) then the time signals can indeed be used to isolate elements before frequency analysis.

The first stage in an element description is therefore the description of the peak frames which it contains. That is the subject of this section. In the next section, **Finding the elements**, the question of characteristics of the element as a whole is addressed.

Figure 5.1 is a sonagram of the typical types of bird vocalizations as identified by Thorpe [8]. Figure 5.2 is a similar collection of sound types analyzed by the BCR system. For each type in the latter figure, the spectrum of a single frame is shown, with the peaks in the corresponding peak frame highlighted. The classification of bird call sounds used in the BCR system is based on observations of these different types.

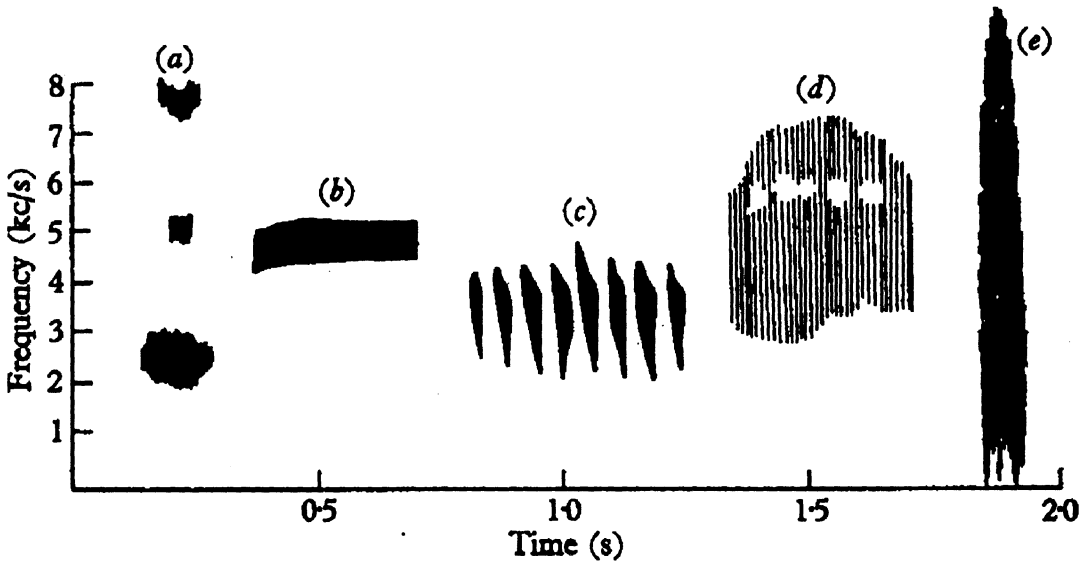


Figure 5.1 A representative series of bird notes showing strikingly different tonal quality. **a.** Nightingale note, very pure, with harmonics. **b.** White-throated Sparrow, clear whistle. **c.** Marsh Warbler, musical trill. **d.** Clay-coloured Sparrow, toneless buzz. **e.** Budgerigar, noisy flight squawk. (After Thorpe [8], p.63).

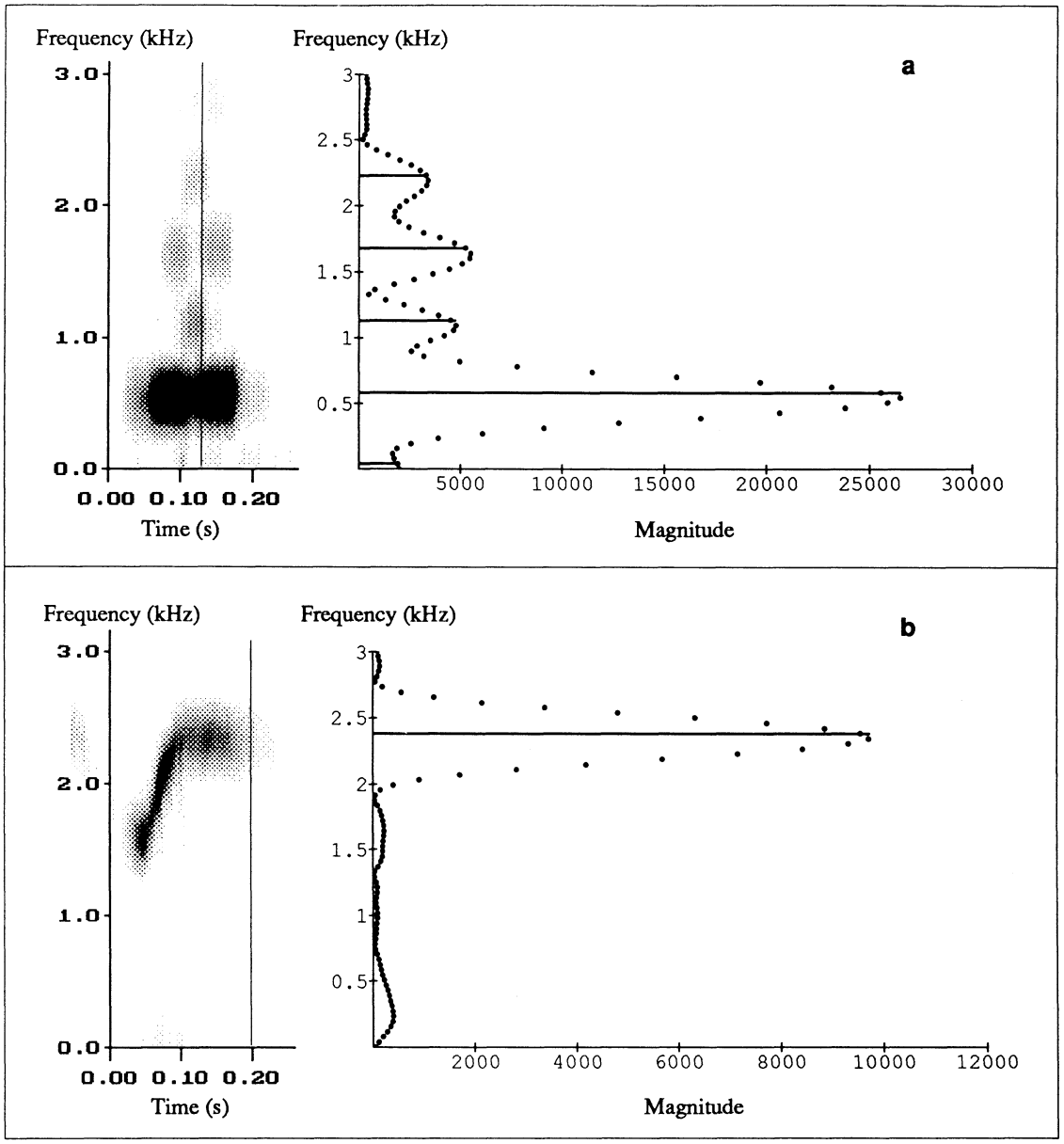


Figure 5.2 (a,b) Typical bird vocalizations as seen by the BCR system - sonagram and spectral frame. The vertical line in the sonagram shows the position to which the spectrum corresponds. The horizontal lines in the spectrum mark the peaks as found by the peak extraction process. **a.** Musical hoot, African Hoopoe (*Upupa epops*). **b.** Short sweeping whistle, Southern Boubou (*Laniarius ferrugineus*).

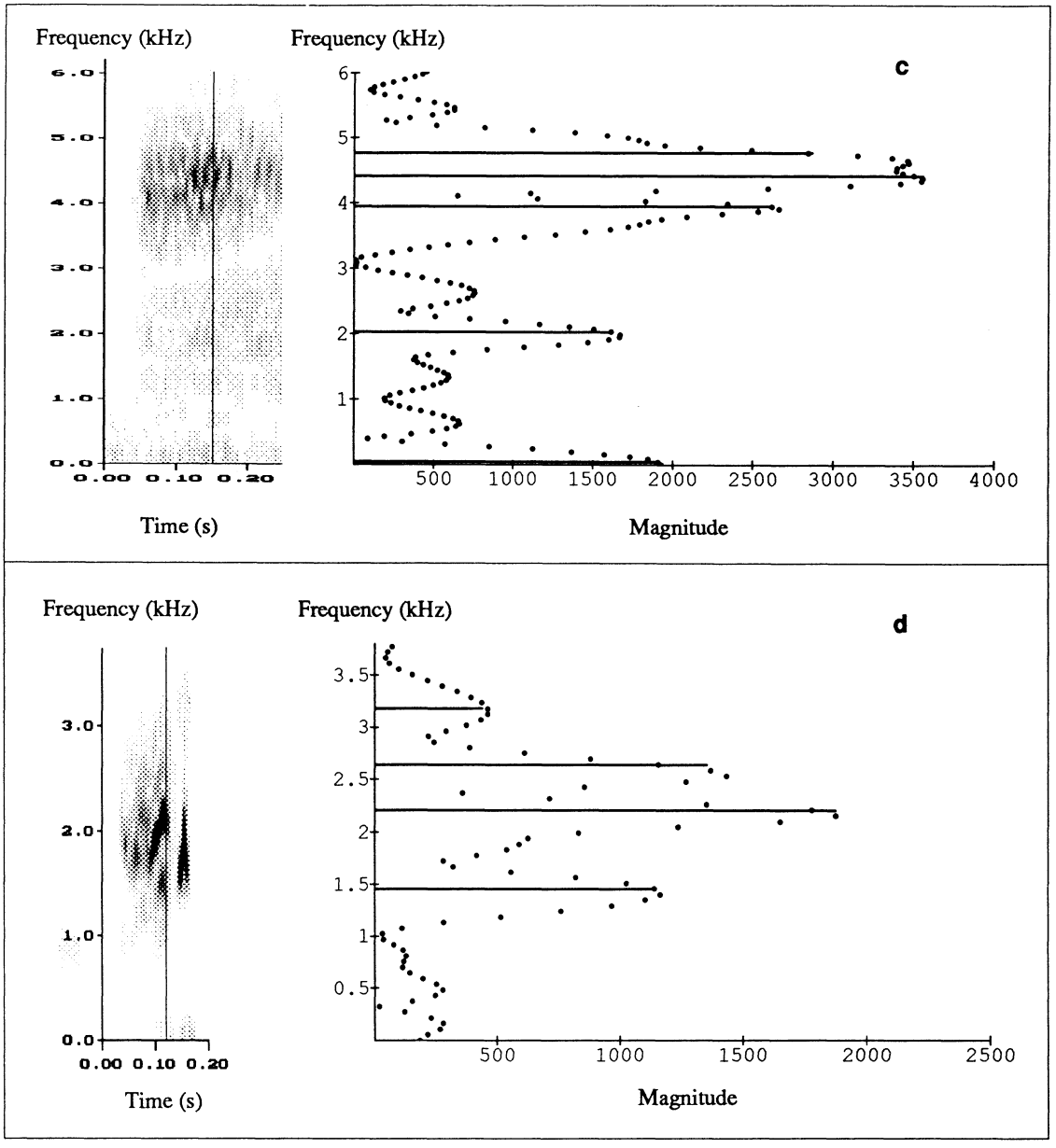


Figure 5.2 (c,d) Typical bird vocalizations as seen by the BCR system - sonagram and spectral frame. The vertical line in the sonagram shows the position to which the spectrum corresponds. The horizontal lines in the spectrum mark the peaks as found by the peak extraction process. **c.** Toneless hissing or grating of Fiscal Shrike (*Lanius collaris*). **d.** Churring or gurgling note of Olivetree Warbler (*Hippolais olivetorum*).

A premise of the BCR system is that the different types of sounds, hissing, clicking and buzzing, vocal, honking and hooting, whistling and trilling, can mostly be identified by looking at a single frame taken from an element in the sound. In the cases where this is not true, it is because the sound is a rapid sequence of many elements, qualitatively different to the sound produced by the same elements in isolation. The latter types of sound are the trilling and buzzing. Slow down a trill and it becomes a series of whistles. A slow buzz is a series of clicks. Since the BCR system will recognize the difference between a buzz and a click when it considers the syntax of the call, it is not important to the sound classification stage that these types of sound are indistinguishable at the frame level.

The following types of spectral frame can therefore be identified. Besides being apparent in the examples of bird sounds above, they are justifiable as different types because they represent the principle types of sound generating mechanism in birds:

Noisy (Broad): These frames come from the hissing, clicking, buzzing rasping and other unmusical or noisy sounds. Their spectra have a large number of frequency components, spread over a broad range, resulting in a plateau-like shape. The production mechanism for this type of sound is probably non-oscillatory - hisses may be made as in human whispers, by air being forced passed a constriction in the vocal tract. Clicks are explosive sudden release of air from behind a constriction. Noise-like sounds may also be formed by the rapid frequency modulation of a oscillating source, such that many new and non-harmonically related components are introduced, combining in a noisy jumble. See Figure 5.2c

Harmonic: In contrast with noisy sounds, the sound containing these frames results from an oscillating source with a significant number of harmonics. The sounds are musical or perhaps vocal in nature, including hoots, honks, cries and barks. The spectrum usually contains a strong peak at the fundamental frequency, and other peaks at integer multiples of the fundamental. Due to filtering in the vocal tract and the various source (oscillator) modes, not all harmonics may be present, but this type of frame is defined by the presence of enough peaks to be able to detect the harmonic relation between them. See Figure 5.2a.

Single: Like the Harmonic frames, these originate in an element produced by an oscillating source. However the harmonics of this oscillation are not significant - either the source is very pure, or the harmonics are filtered out by the vocal tract, the transmission medium or the listener's ears. Such sounds are whistles, "tweets" (whistles with rapidly varying pitch that form a variety of types of sound) and trills made from short whistle segments. They account for the calls of a large proportion of the smaller birds. The spectrum simply has a single peak. See Figure 5.2b.

Multiple: The three basic types above are well defined and cover the basic range of sound quality and sound production mechanism. A smaller number of birds are capable of producing sounds that appear to originate in more than one independent source. That is, the sounds may be oscillatory, but the frequencies produced may not be harmonically related at all. Also, in the transition of the other multi-frequency sounds (Noisy and Harmonic) to silence or to different types of sound, intermediate sounds may be produced that are not strongly characteristic of the basic type. The Multiple frame type is meant to include all sounds that do not fit neatly into the first three, and do not have a well defined sound or production mechanism (the birds that actively produce unrelated frequencies such as the Indian Mynah or some parrots usually make a large variety of sounds, some of which imitate human vocalization). The spectral frames contain more than one peak, and the peaks are not clearly related. See Figure 5.2d.

Silence: This simply refers to those frames which have no significant peaks - originating in (or perhaps defining) the spaces between elements.

The different types, or *classes* of spectral frame have thus been established. Besides that given above, further justification for this classification will have to be empirical. Once a the recognition system based upon this classification scheme has been described, it will be possible to evaluate the scheme's effectiveness in distinguishing between different sounds.

5.1.1 Frame classification scheme

Turning to the question of a computer algorithm that can place a spectral frame in the correct class, it is worth noting that the classification scheme outlined here is one which lends itself to efficient handling by automatic means. As will be seen, there is no need for computation-intensive pattern matching or other artificial-intelligence techniques typically associated with classification. The reason for this is that the classes are broadly defined, differing markedly in overt parameters such as number of peaks. Differences between frames of different classes are not likely be subtle or ambiguous, which translates to a lack of need for sophistication in the classification algorithm. Offsetting this advantage is the lack of ability of the scheme to detect or characterize small differences between frames. Here and in the element characterization described later in this chapter, one of the most important assumptions of the BCR system comes into play: The syntax matching process (see Chapter 6) is assumed to compensate for the weaknesses of the pattern matching processes. This assumption and its implications are discussed in the concluding chapter at greater length.

The first casualty in the search for a non-computation-intensive classification scheme is, as has been implied, the detail-rich **spectral frame.** Only the concise, essential information available in the **peak frame** will be used to detect the frame's class. This decision was not made in isolation,

but as part of the entire BCR system strategy: The peak frames were created with the express purpose of acting as input to a classification algorithm. The number of peaks placed in each frame was preliminarily fixed at the minimum required to be useful in the scheme as outlined above. As will be apparent in the discussion which follows, this number is 5.

The easiest classes to detect are *silent* and *single*. A peak frame with only one peak (frequency magnitude value pair) is always, and immediately, classifiable as single. No significant peaks at all means the frame is silent. In practice it will be found that some frames which strictly contain more than one peak should logically be considered single nevertheless, owing to the dominance of one of the peaks over the others. This notion of the **weight** of a peak will be expanded upon shortly.

If a frame has more than one strong peak, it might belong to any of the remaining classes. The positions of the peaks need to be considered in order to distinguish further between frames of this type. In a *harmonic* frame the frequencies of all the peaks should be integral multiples of some basic (fundamental) frequency value. However more sophistication in determining the relationship between peaks is required than simply searching for a common factor. This is because, due to random fluctuations, distortions in the transmission channel (the acoustic path between sound source and receiver) and possible inaccuracies in the processing of the call (such as quantization of frequency values), a perfect harmonic relationship is not necessarily to be expected. Some leeway must be allowed in the definition of the harmonic class or no frames will qualify.

Given that such a criterion for "harmonic relatedness" is found (as formulated later in this section), it will be possible to find frames which are neither silent, single nor harmonic. The peaks in such frames will have no obvious positional (frequency) relationship. The task of determining which of these frames are *noisy* and which are *multiple* then simply reverts to one of counting peaks. A multiple frame is one which is not noisy or harmonic, which means that it contains a small number of significant, unrelated frequencies. A large number of unrelated frequencies produces a noisy sound, while related frequencies would make it harmonic. Thus it remains to simply determine how many frequencies a non-harmonic frame can have before it is classified noisy. To answer this question the sound-producing mechanism could be considered. It was stated in Chapter 2 that cases of birds producing more than one independent sound source were rare. In such cases the number of independent frequencies would usually be two, but some speech-imitating sounds would require three or even four. Five different frequencies, however, would be highly unlikely and certainly difficult to explain. Thus the presence of 5 peaks would seem to indicate that the frame does not belong to the class of sounds that would be produced by such birds.

Strictly, the criterion for discriminating between noisy and non-noisy frames should be an empirical, possibly subjective one. Calls should be played back and classified by human observers, and the peaks in the calls counted. Otherwise, it might readily be argued that 5 peaks are not enough to induce "noisiness" in a sound. However the number 5 here seems likely to be a useful, if not necessarily rigourous indicator of noisiness. The implication is that a frame with 5 peaks is actually one with "many" peaks, generated by a non oscillatory mechanism. Oscillatory mechanisms generating multiple peaks would usually be harmonic sounds (which possibility should be eliminated first) or sounds in which each peak is generated independently. In the latter case, the frame will contain less than 5 peaks in the vast majority of calls.

The flowchart in Figure 5.3 summarizes the outline of the classification scheme.

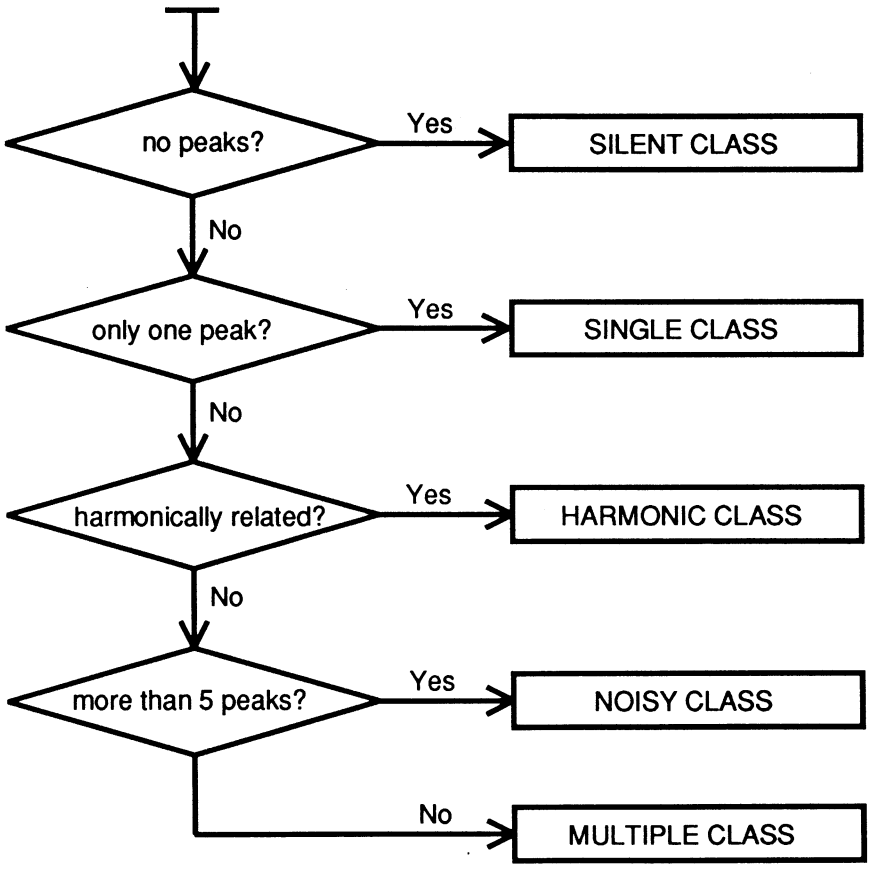


Figure 5.3 Flowchart of peak frame classification scheme. The diamond shapes represent conditional tests and the rectangles, conclusions reached about the frame class.

The scheme is constructed in such a way as to pivot on two decisions: The **single** decision and the **harmonic** decision. Given a procedure for making these two decisions, the other decisions simply become a matter of counting up the non-zero peaks in the frame. Determining the "singleness" or "harmonicness" of a sound is not quite as simple. There is a dependence on the relative significance of peaks in the frame. As mentioned above, a single frame might have multiple peaks, one of which dominates so much over the others that, subjectively, the sound is a single-type sound. Or a number of peaks may be clustered together closely enough to sound like one. There is also the possibility of errors in the frame, such as the "twin peak" error mentioned in the previous chapter. A lenient definition of "singleness" might be able to compensate for such errors. In a harmonic frame, there may be extra, less significant peaks, which do not fit into the overall pattern of harmonic behaviour, yet are not sufficiently influential to warrant the sound being classified noisy or multiple. This is in addition to the possibility of errors mentioned previously.

A continuous scale needs to be defined, against which frames can be measured to determine their singleness. At the one end of the scale would fit a clearly single frame, containing only one peak. At the other end, a clearly non-single frame, containing two or more audible, influential peaks would be placed. Similarly a continuous scale between a harmonic (ordered) and non-harmonic (disordered) frame should be used. A point on each scale can then be designated as the boundary between "singleness" and "non-singleness", or "harmonicness" and "non-harmonicness". Without such continuous scales, only frames belonging at the extremes would be reliably classified, and those not quite fitting the definition either rejected or inappropriately classified.

In fact, it may be true that all frames found for example, subjectively to sound single, do contain only one peak. No study to determine this has been done. Instead it was thought prudent to allow for the possibility that a sound can be "nearly single".

How can a discrete property of a frame be spread over a continuous scale? Surely a frame either has a peak or doesn't, frequencies are either integer multiples of each other or are not? The answer lies in considering those properties of the peak frame that are not characterized by integers, as are the number of peaks and the harmonic numbers. Considering instead the magnitude of the peaks, and the distances (in frequency) between them, it should be possible to define degrees of singleness and harmonicness. Let us see how these parameters may be used to defined the two different scales.

The single/non-single scale: A non-single frame must have at least two peaks, and they must be far enough apart to warrant being considered separate. Note that here the particular means by which the peaks have been extracted (in the peak picking algorithm) is not taken into account. It might be argued that peaks so close as to be unresolvable by the BCR system would never reach the peak frame stage as separate entities, and thus all separate peaks in the peak frame are significantly far apart to be considered separate. This may often be true, but it is important to isolate the behaviour

of the different stages in the BCR algorithm. Peak frames can be generated with different frequency resolutions, and in the future peak picking algorithms might change. The classification stage must as far as possible have its own, independent criteria for making decisions, so that such changes can be made with the minimum of restrictions.

Thus a logical measure of the "singleness" would be a figure that expresses the mean distance between peaks in a frame. A suitable statistical measure is the *standard deviation* of the peak frequencies. A low standard deviation means that most peaks are clustered around the mean position. A high value indicates the presence of a significant number of peaks far from the mean. The ideal, single peak frame would have a standard deviation of 0 and a mean frequency exactly equal to that of the peak.

By using the mean and standard deviation, it is possible not only to extend the concept of the singleness of a peak frame to a continuous range, but also to generalize the concept to include the influence of the peak magnitude, and to be independent of the number of peaks in the frame. If the magnitude of each peak is used as a *weight* for the peak in the calculation of the mean and standard deviation, the desired effect of reducing the influence of lesser peaks is obtained. Thus if a frame has two widely spread peaks, with one significantly lower in magnitude than the other, the mean would be heavily weighted towards the strong peak, and the standard deviation would be much smaller than in an unweighted calculation. This could then lead to the classification of such a peak as single (low standard deviation) and the mean value could be used as a *characteristic pitch value.*

The characteristic value can be used by the BCR to differentiate between frames (and ultimately elements) of the same class - in other words, it allows a more detailed description of an element than a simple class designation. The use of characteristic values and parameters for the various frame types will become clearer in the later sections of this chapter. Note that it is not true in general to say that the perceived pitch of a sound is altered by the presence of other frequencies through a simple frequency-averaging operation, as might be implied by the use of the mean frequency as a characteristic value. The determination of perceived pitch is a complex subject. The mean pitch is not meant to have a perceptual basis, it simply is chosen as useful single value with which to stamp a peak frame and set it apart from other frames.

Having defined the single/non-single scale, it is now easy to see how that definition might be modified to define a harmonic/non-scale. In the single scale, the standard deviation of the peak positions told whether the peaks were all clustered together approximating a single peak. The mean of the peak positions gave that approximate single peak's position. If the same calculation is now applied to the **differences** between neighbouring peaks, a similar measure results for harmonic frames. The standard deviation now indicates how close the differences are to a common value (the mean of the differences). If all differences were identical, the frame is perfectly harmonic,

consisting of peaks at regular intervals. The standard deviation in this case is 0, and the inter-peak distance equals the mean difference, and is the fundamental frequency. For all the above to be true, the distance between the first peak (the fundamental) and a hypothetical peak at 0 must be included in the calculation. If the standard deviation is high, there is no common inter-peak distance, so the frame is likely to be irregular - non-harmonic. Note that for harmonic peaks, the characteristic value used is the mean of the inter-peak differences, not the mean peak value. This is a more suitable choice because it better expresses what is characteristic of that frame, namely the fundamental frequency.

By extension, the magnitudes of the peaks belonging to a particular inter-peak distance can be used to weight that distance in the calculation. Since the peaks on both sides of the inter-peak interval are equally significant, the weight for the interval must be a combination of both peak magnitudes. One possibility is to use the geometric mean of the two magnitudes. The arithmetic mean is unsuitable since if one of the peaks is very small (close to 0), this result in a weight value of half the magnitude of the other peak. That relationship (namely, that if one peak in the pair is small, the interval is "worth" half the magnitude of the other peak) is completely arbitrary. A far better relationship is the multiplicative one of the geometric mean, which says that interval is worth nothing at all if one of the peaks is not present. Other weighting strategies were not considered.

With the scales defined, it now remains to determine a decision threshold for each one. Frames with standard deviation falling below the threshold will be classified as single or harmonic, otherwise they will be checked for another class. To see how a reasonable value can be arrived at for the thresholds, it necessary to think once again of the meaning of the standard deviation value. It has units of frequency in both cases (single and harmonic scales). It is a measure of dispersion of frequency values. It therefore seems acceptable to set the threshold distance for the single scale to one similar to the frequency resolution of the BCR system. More important than the actual value used (for this can be varied and optimized experimentally) is the contention that the same value can be used for both the single and harmonic scales. Although the one measures distance between a mean value and a peak, and the other the deviation of an interval from a mean, the fact is that they both make an assumption about when two frequencies are so close together as to be considered one.

For example, take a frame with a single large peak at 1000 Hz and a smaller second peak at 100 Hz. The weighted mean frequency is close to 1000 Hz. The deviation is a little under 100 Hz. If the standard deviation threshold is 100 Hz, the frame is classified as single. Now consider a frame with many peaks. The average separation between adjacent peaks is 1000 Hz. Two of the weaker peaks are separated by 1090 Hz. The deviation of this interval from the mean is 90 Hz. If in the

single case, a peak closer than 100 Hz is "merged" with the main one, should not interval of between 900 and 1100 Hz be considered essentially the same as one of 1000 Hz? In the BCR system, this is the assumption. A single threshold value is used for both classification scales.

The value used for the standard deviation threshold (or *distance threshold* as called in the BCR software) is usually close to the frequency resolution of the system as expressed by the single component model used in the peak picking algorithm (the width of the ideal single component, as computed from the length of the time window used in the FFT). However, the independence of the distance threshold parameter from the frequency resolution is important as mentioned above. Because the classification scheme takes relative magnitude of peaks into account when looking at the distance between them, the resulting standard deviations can be smaller than the actual distances. Thus a threshold can be usefully set to a value smaller than the possible separation between real peaks. A meaningful lower limit on the threshold value could be dictated by the presence of false peaks such as those introduced by the twin peak error. The small spurious "peaks" close to the real peak that appear due to twin peak error can effectively be ignored if they fall below the distance threshold.

Before leaving the discussion on frame classification, the subject of characteristic values must be summarized: The value used for a single type frame is the weighted mean of the peaks - corresponding approximately (or exactly) to the position of the dominant peak (or only peak). For the harmonic frame, the mean difference between peaks is used, corresponding approximately to the fundamental frequency of the sound. What of the other types? For the multiple class frame, there is no well-defined characteristic that could be parametrized - the weighted average frequency in the frame will do as well as any other value. For the noisy frame, both the mean and the standard deviation are meaningful values, devised precisely for the purpose of describing random distributions of data. Since only one characteristic value has been allowed for in the BCR system, the mean alone is used. The characteristic values will be used to find an overall one for each element, and as such will comprise part of the information available to the syntactic recognition algorithm in matching incoming elements to stored data. In addition to the characteristic frequency, the average magnitude of the peaks in the frame is also preserved for later use - it ultimately gives the syntax matching stage an indication of the loudness of each element.

Figure 5.4 shows how the BCR system classifies the elements of Figure 5.2. The class of a frame is represented by the shape of a dot or line drawn at that frame's position, while the characteristic frequency is represented by the height of the dot or line, as explained in the figure caption. The majority of frames have been classified in accordance with the subjective judgement made by observers. In general, the classification might not be the same as that which an observer would make, but is acceptable as long as it can be expected to be consistent over different samples of the call and of other calls of the species.

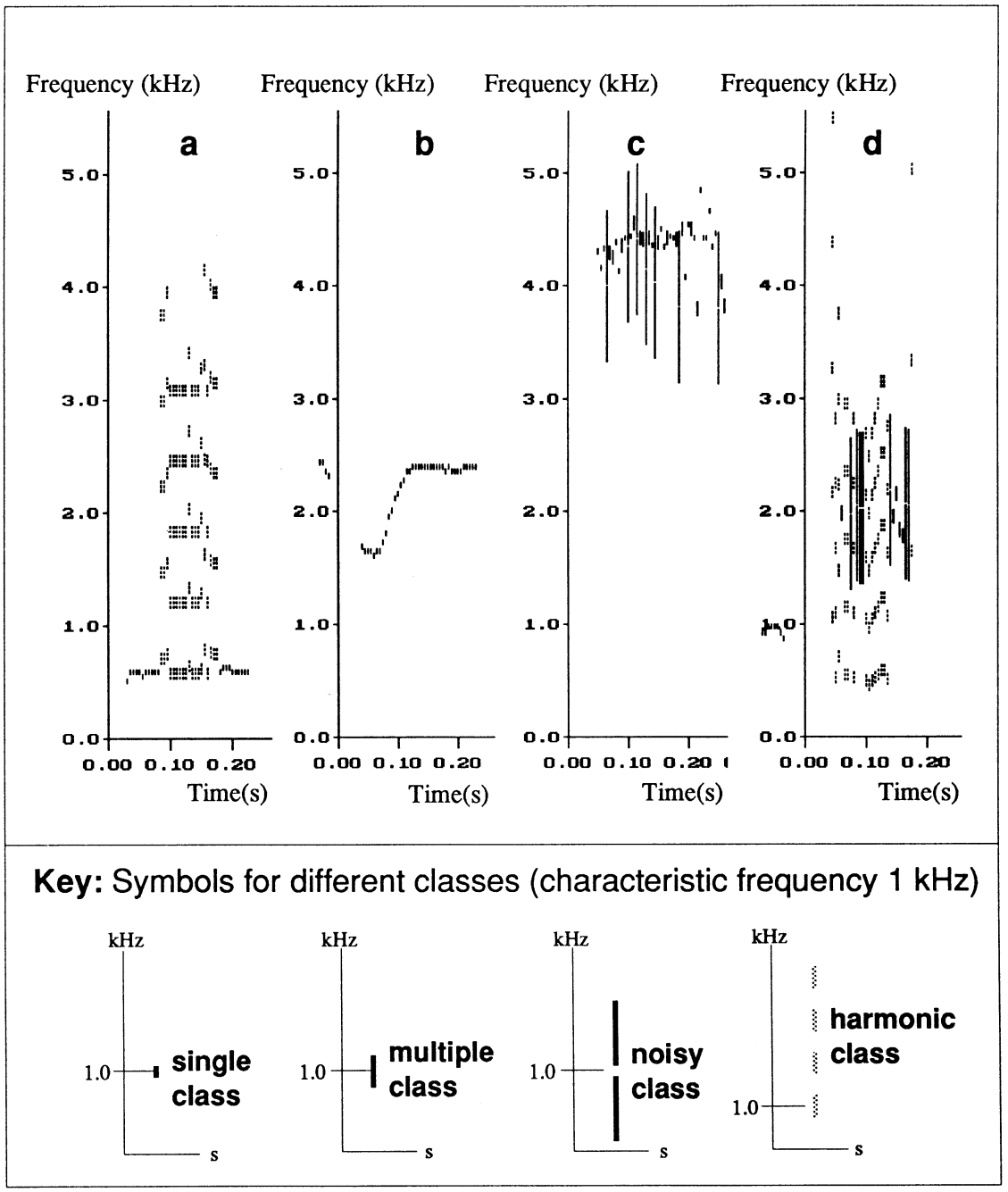


Figure 5.4 Classification of typical bird vocalizations. The labels a,b,c and d refer to the parts of Figure 5.2, which correspond to the portions classified here. In particular, (a) contains Harmonic frames, (b) contains Single frames, (c) contains Broad frames and (d) contains Multiple frames.

5.1.2 Classification software: CLASSIFY

CLASSIFY is the process in the BCR pipeline that accepts the peak frames output from PEAKPICK, classifies and obtains characteristic values for each frame and outputs this data as a file of *classified frames*. See Appendix A and Appendix B for details.

5.2 Finding the elements

Elements in a bird call, as explained, are units of sound distinguishable by their separation from each other in time, and/or by their acoustic character. The BCR system must be able to pinpoint the location of each element, and describe it in terms of its position, duration and acoustic quality. Acoustic quality will be specified using a class name combined with a small number of specific parameter values that give the element an individual flavour.

At each of the stages in the BCR pipeline, a body of data is condensed to produce a smaller body which has greater symbolic meaning when compared with the subjective experience of a person listening to a bird call. Sample data becomes spectral data, with data points corresponding to pitch and texture in the sound rather than rapid pressure variations that are not directly experienced. Spectral data is distilled to peak data, where only the most immediately perceived frequencies are retained. Peak data become classified frames which give a name and a degree to the sound at each moment. Now, in the final stage of abstraction, individual class frames will grouped into a greater entity, the element, thereby once again reducing the amount of data required to describe the call. No longer will the call be represented by numbers valid at discrete, regular moments in time. Instead, the rhythms and punctuations that occur naturally in the sound will emerge.

Obtaining a description in terms of elements involves two issues: Element discrimination, and element characterization. Discrimination is the process of determining the element positions and durations in the call. If period of silence is defined as a silent element, the call can be considered as a string of contiguous elements, and the discrimination process hinges on distinguishing the boundaries between them. Characterization is the determination of attributes for each element, so that it may be described not only as being separate from other elements, but in some way different (or similar). Figure 5.5 shows a bird call with the element boundaries superimposed, and names and example attributes assigned to each element.

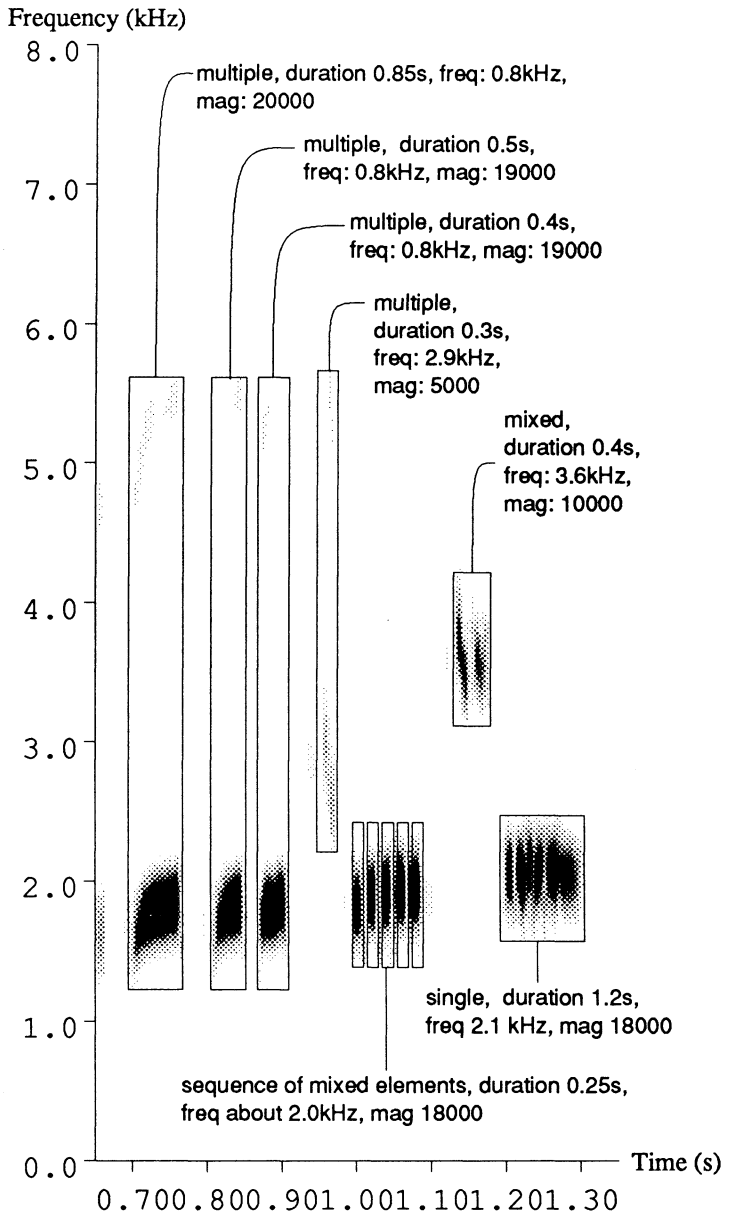


Figure 5.5 Sonagram of the call of the Monotonous Lark (*Mirafra passerina*). Non-silent elements are boxed, defining their boundaries, and the labels are an example of how the elements might be characterized

5.2.1 Element discrimination algorithm

Ideally, the discrimination of elements should be on the basis of their acoustic character. As "experienced" by the BCR system, this character is represented by the information in the classified frames. The discrimination algorithm should search through these frames, grouping together all those (contiguous) ones with similar class frames. In this way, an element in the BCR system would be defined as an interval of sound composed of frames of a single class, or even of a single flavour of frame within a particular class. That definition is the one most likely to preserve the identity of different "syllables" in the sound, given the constraint of using only classified frames as input. In reality such an algorithm is not easily implemented.

The reason for the difficulty is that there is not a clean transition from one element type to another. Frames do not simply switch from one uniform class to another, allowing an algorithm that simply records element boundaries whenever such a class-switch occurs. Instead the boundary between elements is likely to be marked by a number of frame class-switches, a sort of faltering before the new class begins to dominate. In most of the multiple peak frames groups some exception frames may be found. Noisy or harmonic elements may have a few multiple class frames where the number of peaks fell or the relationship between them was not clear. For example, Figure 5.6 shows an element which is easily classifiable as *harmonic* even though not all its constituent frames are classified harmonic the BCR system.

In general, then, an element does not consist solely of frames of a single class. Whether this is due to the nature of the bird call, or simply to shortcomings in the classification system is not important at this stage. It is necessary to cater for such "imperfections" in the elements.

A reasonable goal is to attempt a compromised version of the ideal element discrimination algorithm, in which elements are delineated in such a way as to contain frames of **mostly** a single class. Such an algorithm would have to consider the transitions between classes in the class frame sequence, but not on a frame-by-frame basis. Rather, trends in class frame composition would have to be analysed. In an attempt to view the class frame sequence in terms of such trends, an experimental analysis subroutine was written and incorporated into the DRAWER call-viewing program. The analysis consists of considering a sliding "window" in the class frame stream. The number of frames of a specified class that are to be found in the window is plotted and the window is advanced one frame. The size of the window (number of frames), the class to be counted, and a plotting mode are all selected within the program.

The window of frames shows the effect of considering the class of surrounding frames. Instead of a transition from one class to another causing the BCR system to assume that a new element had been encountered, this conclusion would only be reached if the past N frames were all of a new type, where N is the window size. The DRAWER subroutine makes a separate plot for each N in

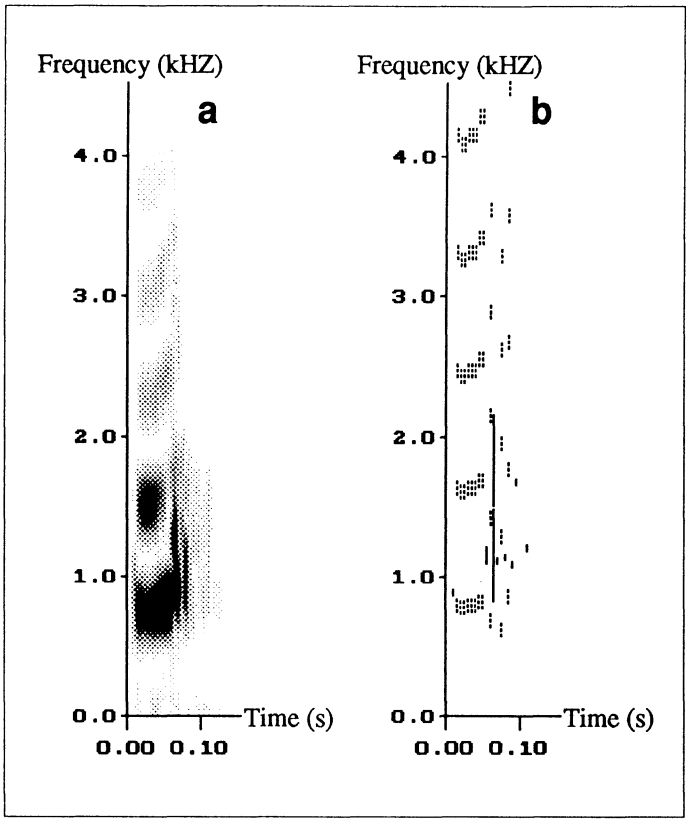


Figure 5.6 a. Sonagram of an element in the call of the Redbilled Hornbill (*Tockus erythrorhynchus*). **b.** Classified frames of the element showing mostly harmonic frames, but with some frames of class single, and one each of classes multiple and noisy (see earlier figure for key to interpreting classified frame symbols).

a specified range. From such plots it was hoped that a reasonable value for N would be found, which would properly trade off the loss of time resolution (only elements longer than the duration implied by N would be detected) with the gain in rejecting spurious short class transitions. Figure 5.7 shows one such set of plots, superimposed on the sonagram of the call.

The peaks indicate positions where the window contains the local maximum number of frames of the chosen class. The positions of the peaks are certainly related to the element boundaries. But a simple algorithm using these contours to determine the boundaries was not evident. The problem appears to be the original assumption that elements can be easily defined based on class content. Even if the fast class transitions near the boundaries of some elements can be smoothed over using

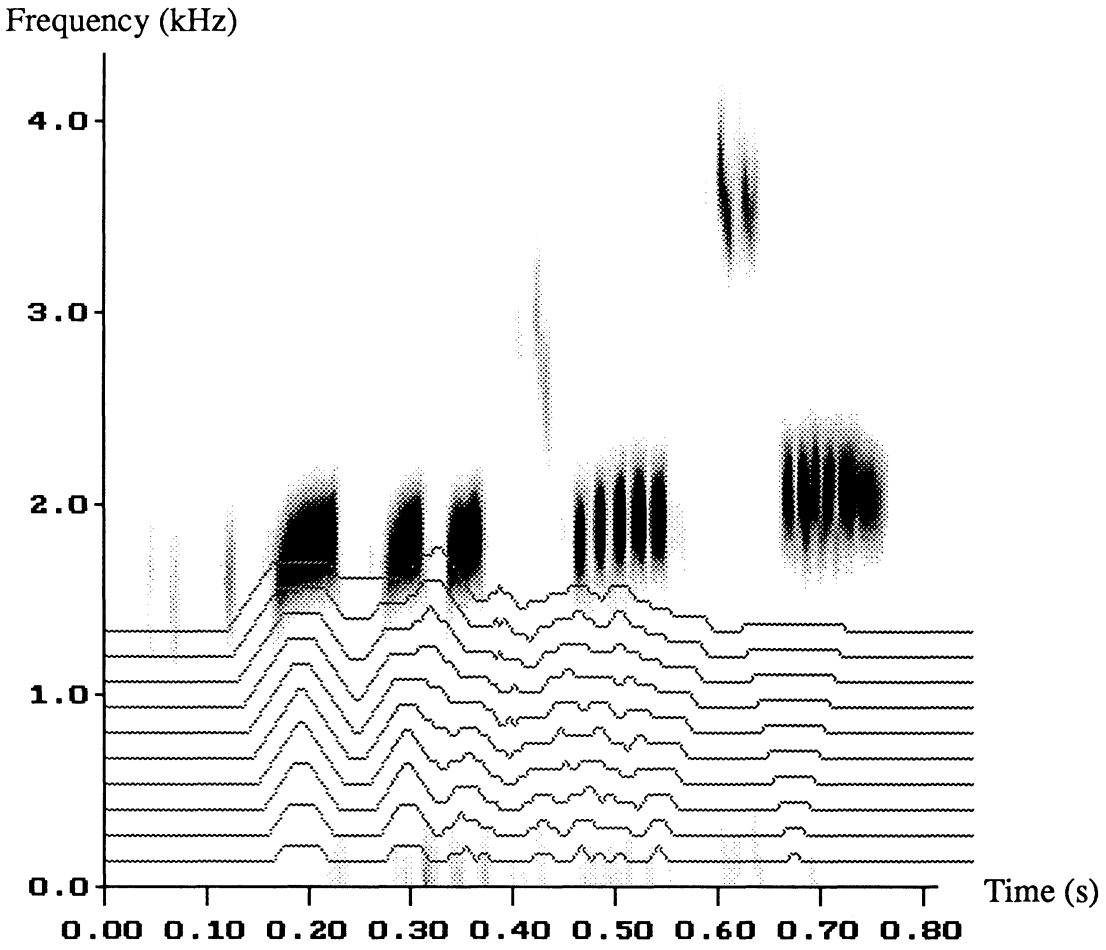


Figure 5.7 A moving window class analysis of the call of the Monotonous Lark (*Mirafra passerina*). Each contour (left-to-right stripe exhibiting peaks and plateaus) corresponds to a different value of window length, ranging from 2 to 20 in steps of 2. In the background is the sonagram plot of the call. By studying the contour peak positions and their relationship to the elements in the sonagram, an algorithm was sought to reliably pinpoint element boundaries.

the windowing idea, the fact remain that sometimes there will be rapid trains of short elements, as in the trills of some birds. The problem is in the nature of the bird call and the inadequacy of the BCR classification system to easily meet the requirements of that definition.

Turning then to other indicators of element boundaries, the characteristic frequency and magnitude of the class frames was considered. The class frame display in DRAWER was augmented to show either the magnitude, the frequency or the product of both, as well as the class (through colour coding). In addition there is a differential plot, where either the difference between the magnitude (or frequency) of adjacent frames, or their ratio, is plotted.

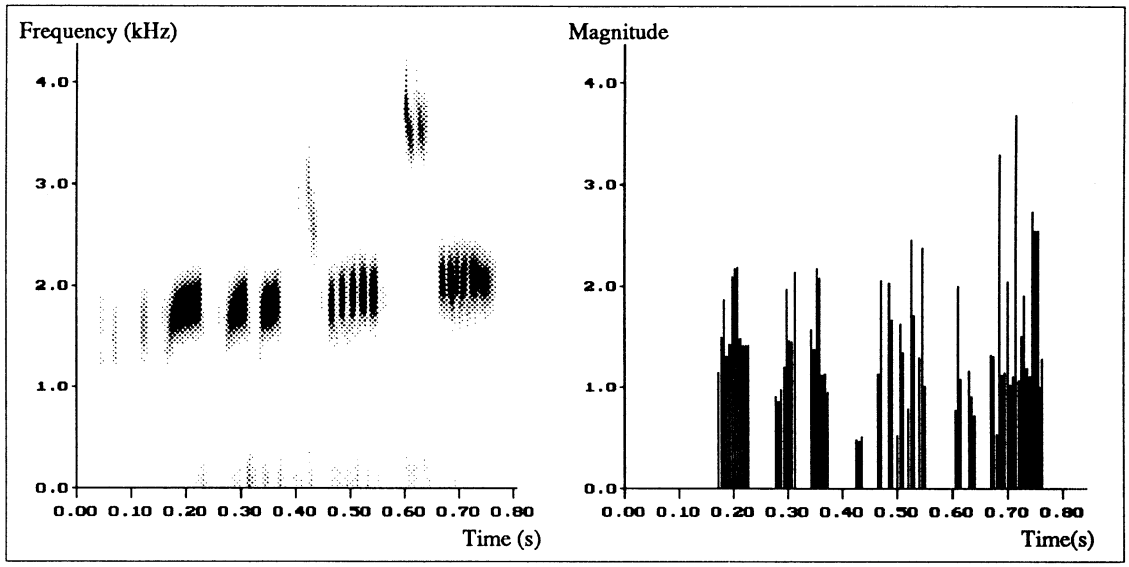


Figure 5.8 Left: Sonagram of the call of Monotonous Lark. **Right:** Characteristic magnitudes of class frames of this call. The periods of silence occur when the magnitude falls below a certain level.

From observing these plots, it was clear that the only useful correlation was between magnitude and element boundary. This is another way of saying that the only simple boundaries to pick out are those between a voiced element and a period of silence. Figure 5.8 shows a plot of characteristic magnitudes compared to the call's sonagram. It was decided to rely on this information only. Referring back to the definition of an element at the start of this section, we are in effect ignoring the possibility that two elements of different acoustic quality but similar overall intensity are joined without an intervening period of silence. Only elements that are separated in time by a silence will be distinguishable by the BCR system.

Although this decision represents a compromise, it has the great attraction of a fast, easily implemented means of element discrimination. Like the decision to use only a few broadly defined classes for the bird sounds, the possibility that the choice will be adequate for the BCR system's purposes is an attractive one. If the BCR system can be made to recognize calls without having to accurately classify frames or discriminate between all possible element combinations, the result

will be a boon for the practical implementation of such a system. It will also show that the ignored information is not essential in species identification. If, on the other hand, the BCR system does not succeed, these compromises should be the first to be reviewed in attempting to improve its performance. It certain that a more sophisticated approach to element discrimination would yield results richer in recognizable information, but it is worth trying to achieve recognition without complex methods.

Having decided to use only the magnitude, the element discrimination algorithm need look no further than the class of each frame. The CLASSIFY algorithm will already have compared the magnitude with a threshold in order to determine whether it is silent or not. This threshold (see **Class_mag_thresh** in the CLASSIFY process) needs to have been chosen to ensure that a "clean" transition between silent and non-silent frame occurs. If it is too low, a few single-frame non-silent "elements" might occur at the edge of each bona-fide element. If too high, elements will be shorter than they should be (the element length is that of the intersection of a horizontal line at the height of the threshold with the essentially hump-like shape of the element magnitude plot, as seen in Figure 5.8).

The fact that the magnitude threshold used in the classification stage influences the way the element discrimination stage performs is perhaps a violation of the principle of independence between stages mentioned earlier in this chapter. Possibly, the element processing stage needs its own threshold parameter. However the classification stage must have such a parameter, and it is hard to see how the two parameters could have different values. While the discrimination is being performed on the basis of a simple magnitude threshold, it seems acceptable to leave the element stage coupled to the classification stage in this way. Should it be found that changing the parameter in order to influence the element processing degrades the performance of the classification stage, a justification for separate parameters will have been found. It seems more likely that the element discrimination might in the future be separated from the classification stage through the use of an entirely different algorithm rather than simply a different parameter value.

In summary, then, the element discrimination is done by placing element boundaries whenever a transition in the frame class between silent and non-silent, or vice versa, is observed.

5.2.2 Element characterization algorithm

Once element boundaries, and therefore element positions and lengths (durations) are known, the element can be analyzed to produce a summarising description. If such a description were to be made in English, perhaps it would be something like "a loud, high-pitched, whistle-like element" or "a quiet, low hoot". The description must be assembled from the information in the constituent

frames of the element. Thus the loudness could be an aggregate of the individual frame's characteristic magnitudes. Likewise, the pitch could come from the characteristic frequencies. Finally the type of sound could be a reflection of the classes of the constituent frames.

The description thus formed is the BCR system's "view" of an element. As will be seen in the next chapter, the systems "sees" calls as strings of these elements, and deals with calls on an element-by-element basis. Each element is compared with element descriptions stored in the system's database. The parameters that are chosen to constitute an element description determine the ability of the BCR system to distinguish between elements. If, for example, there is no loudness parameter, the system will not be able to tell a loud element from a soft one, all other things being equal. If, on the other hand, the separate magnitudes of all the frames in the element are included in the description, the system might become too "fussy": It might reject (not recognize) an element because it's set of magnitudes do not exactly match the sets of any in the database.

The choice of parameters is therefore highly influential in determining the performance of the BCR system. The ideal choice is that set of all parameters which are least variable when originating in different samples of the same element. As illustrated by the example, including parameters which are too variable results in false rejects, while omitting any important unvarying parameters will result in false matches.

Using the variability criterion, loudness is a bad candidate for inclusion in an element description. Certainly, the absolute magnitude of the sound is highly variable between samples of the same call - it depends on the volume of sound emitted by the bird, on the distance of the microphone from the bird, on atmospheric and geographical conditions, and so forth. None of the factors are related to bird species, so should not be allowed influence a recognition system. The relative magnitude of the element, when compared to the average magnitude of the call, might be a better recognition parameter. However an average call magnitude is not easy to obtain, as mentioned earlier in the discussion of peak extraction. There is often no fixed call sequence, but a number of different combinations, so there is no well defined "standard call" over which to perform an average. Such a thing would be possible only with birds with highly stereotyped, simple calls.

So magnitudes in the frames are not to be used as element description parameters. Magnitudes will however be used indirectly in the class parameter set, as will be seen. Frequency is a good recognition parameter, as should be clear by now. The only significant variability in the frequency content of a particular element is likely to come from the bird itself, and since frequency is used as a primary species identification parameter, this variability must be accepted as part of the "signature" of the bird. It makes good sense to include the frequency information available in the element description. The frequency information meant here is the characteristic frequency: Of course one aspect the of the frequency content of a frame is expressed in its class - but this can be

treated as a separate attribute of the sound, just as the type or quality of a sound is separate from the sound's perceived pitch, even though the two both result originally from the frequency content of the sound. Indeed, the characteristic frequency either does correspond to the frame's pitch, or to an analogy of the pitch in the case of sounds (noisy types and some multiple types) which don't have a well defined pitch.

How should the class frames' characteristic frequency information be used in element characterization? Should all the frequencies be available for comparison with database elements? Should a single compound frequency value be derived from all the element's frames? To answer this consider the single-type element in Figure 5.9 - it has a meaningfully varying pattern of characteristic frequencies.

The frequencies in the figure trace a shape (called a *pitch contour*) which is significant in recognizing the element. Many bird calls are composed of element such as these - most of the small oscines ("singers"), which comprise perhaps half of all birds, have single-type sounds whose frequency variation with time gives them their unique character. It is clear that a means of representing and matching the shape of elements such as that in Figure 5.9 is important in a recognition system. To emphasise the point, we can imagine a similar element, which has been reversed in time. Thus instead of starting out high in pitch and "gliding" to a lower pitch, it starts low and ends high. The two elements, the original and the reversed, would give an identical value if their characteristic frequencies were averaged over the element length. Yet they would sound recognizably different.

In spite of the implications of the above illustration, the BCR system in its present form does only use the average of the characteristic frequencies in an element. The reason is similar to the reasons for other compromises made in the system: Encoding the pitch contour of an element would be a more complex task, and not worth doing until it is clear that the BCR system cannot function without the more complex scheme. Unlike with compromises such as broad class definitions and silence-delineated elements, the lack of frequency contour data is probably likely to be a serious deficiency. But it is nevertheless well worthwhile trying to do without it. If it is possible to do so, not only will the system have that much less complexity, but it will bring to light a fact about bird call recognition that would not otherwise be obvious - namely that pitch contour data is not necessarily essential. The project at the current point simply has not reached the stage when addition of pitch contour handling is pertinent. As will be seen when the behaviour of the BCR system is discussed (in the concluding chapter, 7), there is enough functionality to conclude that pitch contours can be left for a later phase of development.

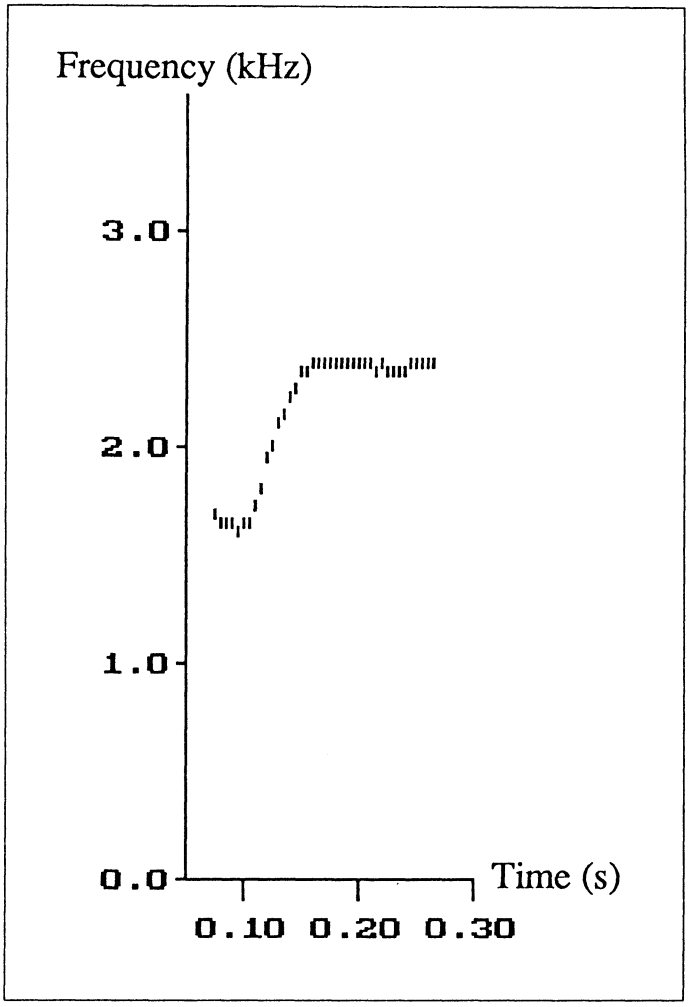


Figure 5.9 Class frame plot of an element from the Southern Boubou, composed of single type frames. The height of each mark is given by the frame's characteristic frequency.

If pitch contours do need to be handled, how should this be done? Essentially, it is necessary to be able to compare a stored contour with an incoming one. Although the next chapter deals with the matching of elements, the hypothetical pitch contour case can be dealt with here.

One possibility is to subtract the contour of the incoming element from each contour in the database of candidate elements. Squaring and summing the differences, a low result would indicate a close match. One problem with this method might be time misalignment. If one element has essentially

the same contour shape as another, but is stretched in time, or does not start at quite the same point in the contour (due, perhaps, to quirks in the element discrimination stage), a mismatch (large sum-of-squares) which is out of proportion to the misalignment could occur.

A more flexible approach might be to use an abstract form of the contour, a linear approximation. The contour data would be approximated by a small set of joined line segments. Then the positions, lengths and slopes of these segments could be compared with those of similar linear contour approximations in the database. Such approximations could be scalable, and allowable ranges set for lengths and slopes, to minimise misalignment problems. The contours stored in such abstracted forms would also take up less space. An algorithm for converting a contour to its linear approximation might once again involve the use of least-squares optimization. A line could be drawn between a point on the contour and another further along. The sum-of-squares of the differences between this line and the contour segment could be calculated and the far point advanced. When the sum reaches a maximum error value, the line thus drawn would constitute a segment and the process would begin again by starting a new line segment at point where the last one ended. Whether this approach would produce consistent similar results (in terms of lengths and slopes) for similar elements remains to be seen.

To summarise the discussion of frequency characterization, it has been decided that a single value will be used, the average of the characteristic frequency values of the elements frames, to differentiate between the pitch quality of different elements. This is in the hope that detailed pitch variations within the element will not prove essential in identifying calls.

Finally the class information must be applied to the element as a whole. As stated, the ideal element contains frames of only one class. The element can be said to belong to this class. In practice, some spurious classes may occur, in the constituent frames, although most elements might still be dominated by one particular class. It is possible, however, that due to limitations of element discrimination algorithms, some "elements" may in fact be composed of more than one acoustically differentiable element, and thus consist of significant proportions of frames of different classes.

Given that it may not be possible to simply specify a single class per element, the question that must be answered is: How much of the class information must be included in the element description? The concept of class was formulated to provide a formal measure of "acoustic quality", pertaining to perception of sounds. It is not meant to describe a parameter of a sound that varies in a meaningful way, as does pitch, for example. And in observing the class content of various calls, this function of the class concept is borne out to a large degree. There is no "class contour", analogous to the pitch contour, expressing a meaningful, ordered arrangement of class frames within an element. Rather a fairly random sprinkling of classes exists, often with one class present in greater numbers than others, and some classes not present at all.

The approached settled upon was to calculate the proportion of each class present in an element. In this way, a characteristic "class signature" or class profile of an element is defined. Rather than specify a class contour, which is not likely to be consistent for all element of the same class, the profile lends itself to being matched against database signatures with enough flexibility to prevent false rejects. The class profile is a set of four numbers (the fifth class, silent, is a special case). Each number is the ratio of the number of frames in the element that belong to a particular class, to the total number of frames in the element. These numbers can be treated by the recognition algorithm as coordinates of a point in a four-dimensional system. To match such a profile with one stored in the database, the distance between the incoming "point" and the stored "point" can be calculated. If the "points" are too far apart, they do not match. The details of the matching algorithm can be found in the software description for the SYNTAX process, in Appendix A.

One important consideration in forming the class profile of an element is the magnitudes of the frames. Often the "contaminant" frames (the ones not belonging to the main or dominant class if there is one) occur near the edge of the element. At the edge, the magnitudes are usually tailing off towards silence. So the contaminants are possibly results of peaks in the frames falling below magnitude thresholds used in the frame classification stage. Their low magnitudes probably indicate low importance. The class magnitudes can be used to weight the profile. This is acoustically correct - the observer hears the high magnitude frames more than the others, and offers a convenient opportunity to play down the influence of frames which perhaps have been misclassified. Thus the weighted sum of frame classes is used in the profile rather than the simplified scheme outlined just previously.

Finally, the element discrimination stage takes upon itself the task of identifying of the dominant class in the element. Although, as will be seen in the next chapter, the element recognition is done on the basis of the profile, it useful for the recognition algorithm to have some preliminary class information, perhaps to narrow down the list of database candidates against which the incoming element is mapped. In this way, the more time-consuming task of profile matching can be left only to those cases where there is some chance of a match. Also, the silent elements have no profile, just a dominant class, and it is convenient to have a separate class parameter associated with the element. The class parameter could also come into it's own should the BCR system evolve to a stage when elements no longer have a mixture of classes. A new class, *mixed* is defined for elements that have no clear dominant class, allowing such indeterminate elements to be distinguished from the more conventional ones, and perhaps opening the way to further processing of these elements.

In summary an element is characterized by an average frequency value, a class occurrence profile, and a dominant class. Figure 5.10 shows an element plot produced by the BCR system.

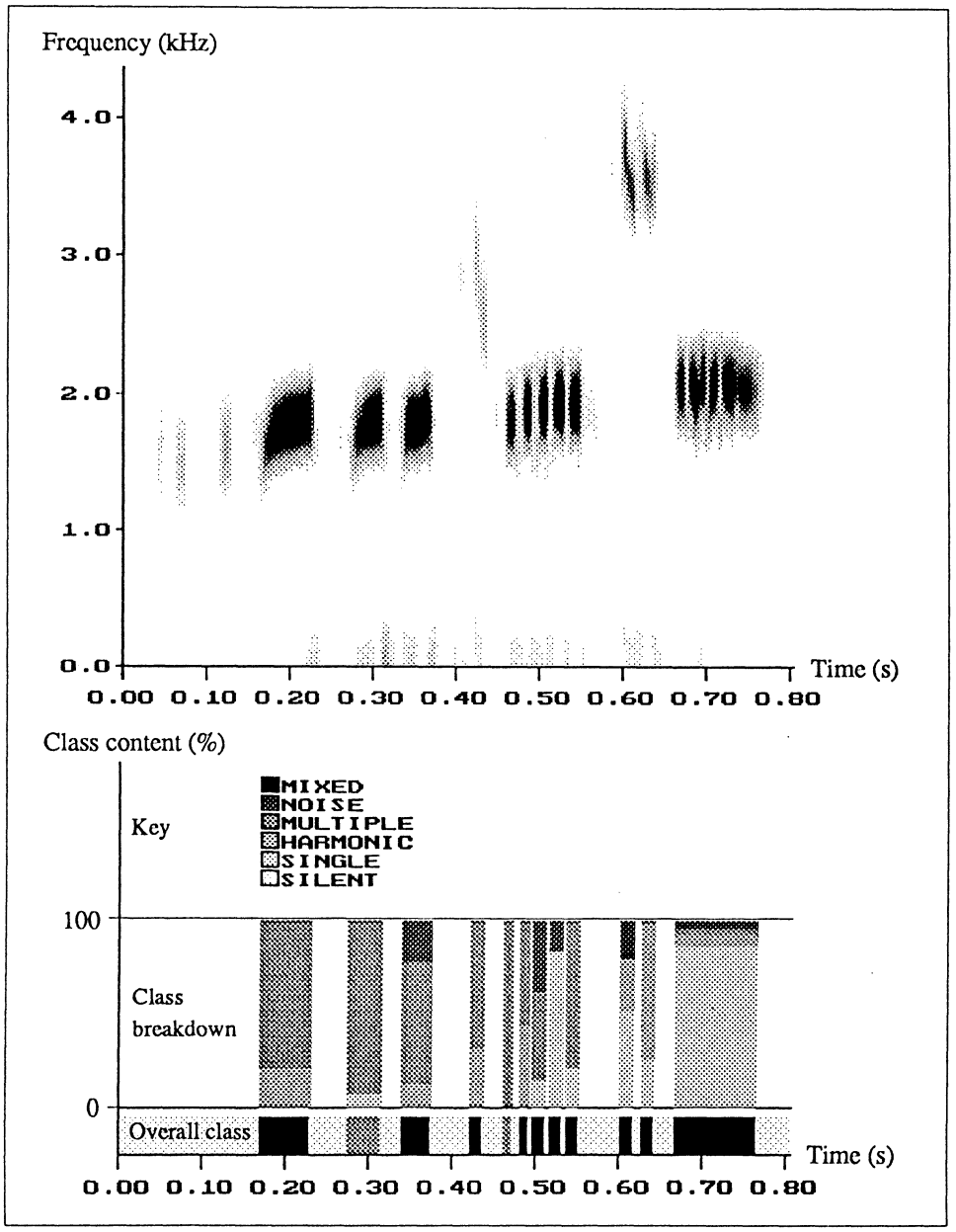


Figure 5.10 Element plot produced by the BCR system. **Top:** Sonagram of part of a call from the Monotonous Lark. **Bottom:** The element plot corresponding to (a). An element is represented as a block, with different coloured sections representing proportions of the different classes comprising the element.

5.2.3 Element software: ELEMENT

ELEMENT is the pipeline process responsible for element discrimination and characterization. Its main input is the stream of classified frames received from CLASSIFY. Its output is no longer a sequence of frames, each having a one-to-one correspondence with one of the input frames. It is a string of Elements - tokens or *symbols* representing contiguous, but not equal-length, intervals of the bird call. Each symbol has a duration, a frequency, a dominant class and a class profile. The symbols are called so in analogy with the primitives used in syntactic pattern recognition (see Chapter 2), which is the basis of the syntax matching algorithm described in the next chapter. More details can be found there. ELEMENT is described in Appendix A and Appendix B.

6
Syntax matching system

The syntax matching stage is the heart of the BCR system. It is the end of the pipeline, the culmination of the processing that is done to the call. In a sense the processing ended with the element discrimination stage described in the previous chapter, and the syntax matching stage is the consumer of the processed call. Its output is no longer another, more abstract incarnation of a bird call, but a decision - a conclusion as to whether the call matches that of a candidate species.

Many of the decisions affecting the behaviour of previous stages were made with the understanding that the syntactic analysis ahead would compensate to some degree for the earlier stages' shortcomings. The premise that a call is recognizable largely on the basis of its syntax, to the extent that only rudimentary pattern analysis and classification need be performed on the sounds, is the central hypothesis around which the BCR system has been constructed. Its ultimate success or failure will have implications primarily for the validity of that hypothesis.

So the syntax matching stage is an important one. On the surface, it is also a simple one: There appear to be few doubts as to exactly what this stage should do. It should accept a call represented by a string of elements, and should determine whether or not this string could have been constructed by a set of 'bird call production rules', using a unique set of rules for every species tested against. It must answer the question: "Could this string be produced by species X?"

The scenario described above, consisting of an input string and a set of production rules, is reminiscent of, and indeed entirely inspired by, the syntactic pattern recognition systems outlined in Chapter 2. Such systems were first used (by linguist Noam Chomsky) in the study of language. The sets of rules are called grammars, since they can encapsulate the grammatical structure of a language. The primitive element in a syntactic recognition system used in linguistic study is the word. Words are indivisible units, and the patterns which the system recognises (as belonging to a language) are strings of words - sentences. Thus the system can be asked, "Is this sentence gramatically correct in language X?" but not, "Do the words in this sentence belong to language X?"

In the bird call context, the primitive units are the utterances we have called bird call elements. The elements as represented by the BCR system are not as abstract as the words in the linguistic system. They are not symbols representing a concept which can also be represented by a sound, as are words. Rather they are essential, distilled versions of the sounds themselves. We cannot

expect to find a single, unique symbol for every bird call element as we can for each spoken word. There would have to be an infinite number of such symbols - corresponding, for example, to the infinite number of allowable durations of an element.

The syntax matching stage of the BCR system can therefore not expect to deal with a finite set of symbols as does a syntactic pattern recogniser. Instead of an exact correspondence of incoming symbols with the symbols used in the syntax rules, the BCR system will have to be content with a "closeness" or near-identity between incoming elements and the entities used to describe elements in the syntax rules. It must compare, or match, the incoming elements with its stored descriptions. The syntax matching stage is therefore actually a two-stage process - the first stage is the matching of incoming elements with element descriptions in the rule set and the second stage is the actual verification of the syntax of the call based on the rules. The first stage, to return to the analogy with a spoken sentence analysis system, is like a word recognition stage that accepts a speaker's utterances and outputs written words. The second stage is like the formal syntactic analysis systems of linguistics.

The syntax matching system differs from a syntactic analysis system, not only in the addition of the element matching stage, but in the way the syntax itself is checked. The possibility that incoming elements do not match perfectly with their stored counterparts means that the syntax checking algorithm diverges somewhat from the formal finite-state or pushdown automata that parse linguistic grammars (see chapter 2). The formal methods allow theoretical analyses of the grammars to be conducted. The BCR system, while doubtless open to theoretical treatment, would not yield to the techniques developed directly for the formal systems reviewed in chapter 2, and no attempt has been made in the present project to develop appropriate analysis techniques. Not only would this be outside the scope of the work required, but the necessity for theoretical analysis of the system's capabilites at this early stage of developement was not felt. The system will be seen to perform the task required of it, and some clearly apparent limitations will be pointed out. A theoretical formulation might be desirable at a future stage, when it important to understand the system's behaviour under all possible conditions. For now, it is sufficient to observe that the behaviour is correct for the conditions found in the sample of calls used as the test base, and those which are forseeable based on the general understanding of bird call characteristics.

6.1 The syntax specification

A syntax specification (*syntax spec* for short) is an abstract description of a bird call which can be compared with the element string formed from an actual call. The recognition system database consists of a collection of syntax specs, one for each species, or perhaps more than one for those

species with distinctly different types of call. Incoming calls are compared in turn to each syntax spec in the database, hoping to find a match. The matching syntax spec identifies the species responsible for the incoming call.

The syntax specification consists of two separate parts: The *element specifications* and the *syntax tree*. The division mirrors the conceptually separate tasks of the syntax matching system, namely the matching of the elements and the checking of the syntax, as outlined above. The element specs define the types of elements which appear in the calls of a particular species. The syntax tree describes how these elements combine to form a call.

There are two types of element spec - silent and non-silent. A non-silent element spec is similar in structure to a non-silent element; it has fields to specify duration, characteristic frequency, and class profile. A silent element spec has only duration. Instead of a single value for each of the above attributes, a range of values is specified into which the fields of a matching element may fall. There is no class field in the element spec, since once the syntax matcher has used the incoming element's class field to determine whether it is silent or not, futher class information comes from the class profile (in a non-silent element match), or is not necessary in a silent element match.

6.1.1 Element specification

Specifying a range for duration and frequency fields is simply a matter of giving a minimum and maximum value for each field. For the class profile, however, minimum and maximum values for each component class percentage are an inefficient way to specify a range. The class profile may be considered a 4-dimensional vector. The axes of the coordinate system in the 4-dimensional space which this vector occupies are the class proportions, the percentages of single, harmonic, mulitple and noisy frames present in the element. It easier to imagine a 2-dimensional vector, however. Figure 6.1 shows the rectangular area defined when separate ranges are specified, through minimum and maximum values, for the two components of a 2-dimensional vector. The implication is that all vectors falling anywhere within the rectangular area are defined as "close enough" to be indistinguishable.

Seen this way, there is little reason to choose a rectangle as the boundary of the "neighbourhood" region. Why not a triangle? In fact a circle is the most egalitarian choice for such a boundary - the points on the boundary are all equidistant from the centre point, which is the nominal value of the vector for which the neighbourhood is defined. On a rectangle, for example, a vector on the corner and one at the centre of a side are both on the verge of non-inclusion in the region - yet the corner vector is further from the nominal value.

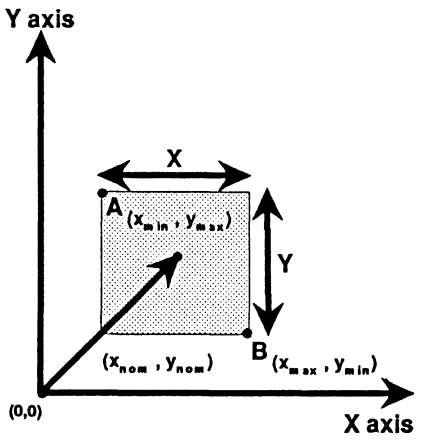


Figure 6.1 A rectangular neighbourhood defined around a nominal vector value at its centre. The rectangle's sides represent the ranges X and Y into which the a vector's x and y components, respectively, can fall to match with the nominal value. The coordinates of the rectangle's corner points, A and B, give the maximum and minimum values for these ranges.

As far as practical efficiency goes, there is an advantage to the circle over the rectangle. The rectangle can be defined by specifying the coordinates of two diagonally opposite corners, a total of four scalar values. A circle needs a centre point - two scalars, and a radius - one scalar. When extended to the higher dimensional cases, the advantage becomes greater. A 4-dimensional hyper-rectangle requires 2x4=8 scalars, while a hyper-circle needs 4+1=5 values.

The element spec thus gives a nominal class profile (centre point) and an *occurrence-distance* (radius) - the furthest vector distance that a matching profile may occur from the nominal point. Below is an example of an element spec, one that would match the "kawak" double-honk of the Redbilled Hornbill. The section 6.2.3 deals in more detail with the matching mechanism. Also shown is an example of a silent element spec.

```
Double-Honk: essential
  frequency 0.7 to 1.5 kHz
  duration 245 to 295 ms
  single 14 harmonic 80 multiple 0 noise 7 %
  occurrence_distance 15 %

Inter-honk-silence:
  silence 35 to 50 ms
```

To enable a computer to read the syntax spec its format must be strictly defined. The BCR system defines the non-silent element spec to consist of 5 lines, the first containing a title terminated by a colon. The title is referenced by the syntax tree part of the spec - see later. The keyword **essential** may follow the title - its meaning will will be explained later. The keyword **frequency** denotes the characteristic frequency range - it is followed by the minimum frequency, then the keyword **to** and then the maximum frequency. Frequency values are understood to be in units of kiloherz, regardless of the text on the rest of the line, which is ignored but is useful for increasing comprehensibility for human readers of the syntax spec. The element duration is denoted by the **duration** keyword on a new line and the range is specified in the same way as the frequency range - the units of time are defined to be milliseconds. The class profile is specified on a separate line, with the proportions (in percent) of each class following the appropriate class name keyword. Finally the occurrence distance has its own keyword on a separate line, and units of percent.

The silent element spec has only a title, followed on the next line by the **silence** keyword and a duration range in milliseconds.

The first part of a syntax spec can now be completely defined: it begins with a title, which is the name of the species or group of calls that the spec describes. The title follows the keyword **BIRD:** on a line by itself - any text before that line is ignored and can be used as commentary space - in fact, all text that is not meaningful to the syntax checking algorithm (i.e. follows a defined keyword and is of a defined format) is ignored.

The next keyword, **ELEMENTS:**, marks the start of the list of element specs. As many different elements as can occur in the call are listed one after the other on the lines following. The element spec list ends with the keyword **SYNTAX:** which marks the start of the syntax tree specification. A full syntax spec for the Redbilled Hornbill might look something like the following:

```
SYNTAX SPECIFICATION (Optional commentary text)

BIRD: Redbilled Hornbill

(Optional commentary text...)

ELEMENTS:

Double-honk: Essential
  frequency 0.7 to 1.5 kHz
  duration 245 to 295 ms
  single 14 harmonic 80 multiple 0 noise 7 %
  occurrence_distance 15 %

Inter-honk-silence:
  silence 35 to 50 ms

General-silence:
  silence 0 to 0 ms

SYNTAX:

First_honk:
  double-honk  First_silence

First_silence:
  inter-honk-silence  Second_honk

Second_honk:
  double-honk  Second_silence

Second_silence:
  inter-honk-silence  Third_honk

Third_honk:
  double-honk  Idle

Idle:
  general-silence  First_honk
```

The element spec named "General-silence:" in the example is a special case of a silent element - it has a duration of exactly 0 with no range variation. This element spec will be interpreted by the BCR system as intended to match any incoming silent element, no matter what the duration. It is a kind of silent element "wildcard" specification. The role it plays is mentioned in the following discussion of the syntax tree.

6.1.2 Syntax tree

The text following the heading "SYNTAX:" is the syntax tree. It describes the syntax of the call by specifying how the elements mentioned in the element specifications fit together. Like the "ELEMENTS:" part, the syntax tree is made up of a list of items, the order of the list being of no importance, and each item beginning with a line containing a title for the item, terminated by a colon.

The items in the list are nodes on a syntax *tree*; they may lead on (point) to other nodes on the tree, and may in turn be pointed to by other nodes. Each node has associated with it one of the element specs from the list in the first part of the syntax spec. The element spec's name is given as the first word on the line following the node title. Following the element name, on the same line, is a list of names of the other nodes in the tree to which the present node points. Thus, in diagram form, the tree part of the example syntax spec presented above would look like Figure 6.2.

When node 1 points to node 2, the implication for the bird call syntax is that the element associated with 2 may legally follow the element associated with 1. The example of the Redbilled Hornbill syntax given above is a relatively simple one: Each node points to only one following node, meaning that there is only one possible sequence of elements. An element matching the spec of "First_honk:" **must** be followed by a silence matching the spec of "First_silence:" and so on. This is in contrast to the tree segment illustrated in Figure 6.3, where an element matching Node_1's spec may be followed by one matching **either** Node_2 **or** Node_3. Moreover, the two alternative nodes correspond not simply to alternative elements in the chain, but, more generally, to alternative branches of the tree that diverge in subsequent syntactic behavior. Thus a hypothetical bird might produce either "tweet-cheep-cheep" or "tweet-cheep-tweet-tweet-tweet" - both alternatives start with "tweet-cheep" but end with a structurally different chain segment.

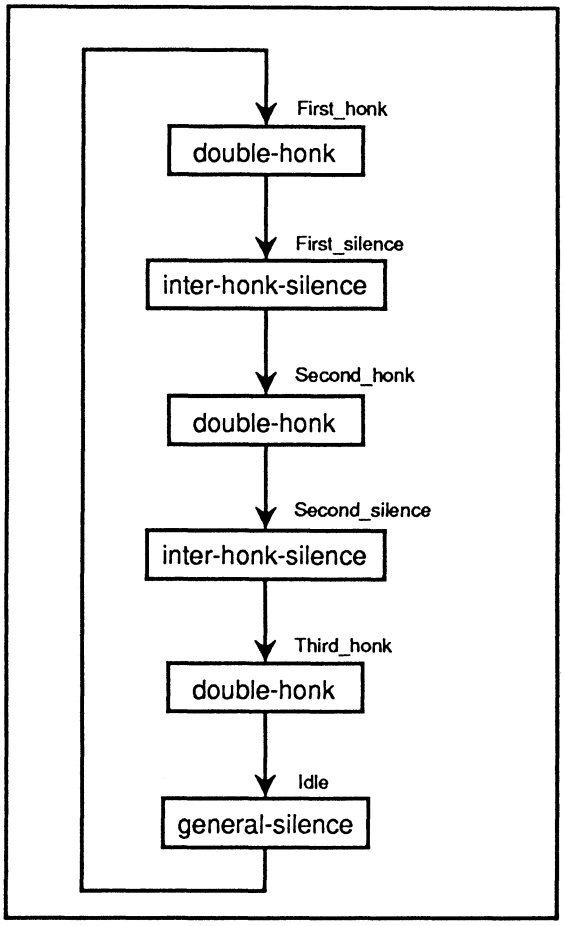


Figure 6.2 Diagram of an example syntax tree for the Redbilled Hornbill. A box represents a node, and the name contained within the box is the element spec associated with a node. An arrow leading from one node to another indicates that the first node points to the second. The name placed immediately above and outside the box is the node title.

The branching in the syntax chain illustrated in Figure 6.3 is specified in syntax spec by placing multiple node names in the "points-to" part of a node declaration:

```
Node_1:
  Element-A  Node_2 Node_3
```

With the introduction of the branching feature, a dual purpose for a node becomes apparent: A node houses an element spec, and it marks a branching point. Examples until now have been of nodes which function in both capacities, or as element spec holders only. There is also a need for nodes which mark branching points without specifying an element. Such a node is specified in the

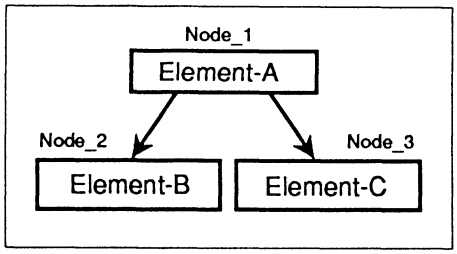


Figure 6.3 Syntax tree nodes can point to more than one subsequent node, indicating that the element to follow the one matching the present node may be one of a number of different alternatives, or branches.

syntax spec by using a dot, ".", in place of the element spec name. It allows the situation illustrated in Figure 6.4, a repeated choice between alternatives, which without the "elementless" node could require a lengthy (or infinitely long) and repetitive notation. An example of the elementless node follows:

```
Node_1:
   .  Node_2 Node_3
```

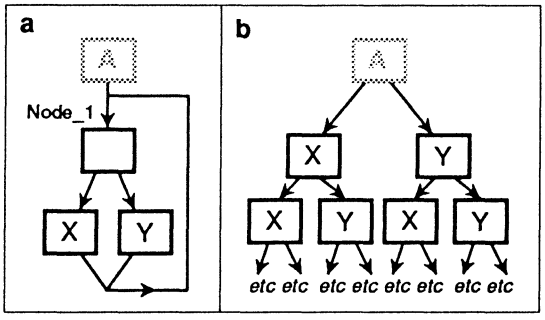


Figure 6.4 a. An elementless node (Node_1) used in a loop allows specification of a chain composed of equally syntactically correct alternatives. **b.** Without the elementless node the situation represented by (a) requires an infinite progression of repeated node specifications.

The Redbilled Hornbill example demonstrates another important feature of the a syntax spec - backward linking, or looping. If the chain of events is followed starting from the "First_honk" node, a linear succession of nodes is encountered in a fixed order: First_honk, First_silence, Second_honk and so on ending in Third_honk and then Idle. The "Idle" node points backwards in

this sequence, to the "First_honk" node. The syntax implied by this spec is thus, if we consider now the actual elements associated with each node, as follows: A Redbilled Hornbill call may consist of any number of repeated sequences, where each sequence consist of three "double_honks" following one after the other and separated by "inter-honk-silences". Between repeated sequences, a silence of any length ("general-silence") may occur[1]. Here is an example of a call that would fit this syntax, using the syllabic approximation found in Roberts [2], "kawak", to represent the double honk sound: "kawak-kawak-kawak, (pause), kawak-kawak-kawak, (pause), kawak-kawak-kawak..."

The loop in the above example is used to express the fact that, after an indefinite pause, the call may be repeated (note the "General_silence" element at the "Idle" node, which is specified as "silence 0 to 0", signifying a silence of any length). Such overall repetitions loops are likely to occur in most, if not all, bird call syntax specifications. The loop can however be used to describe local features of calls, in conjunction with the branching feature of the syntax spec. An example is the long, drawn-out trill of the Crested Barbet and the trills and buzzes of numerous other birds. These types of sound are formed by a string of repeated short elements, and the use of two branches from a node, the one looping back to the same element and the other leading on to a new part of the tree, allows a trill of unlimited length to be specified. See Figure 6.5

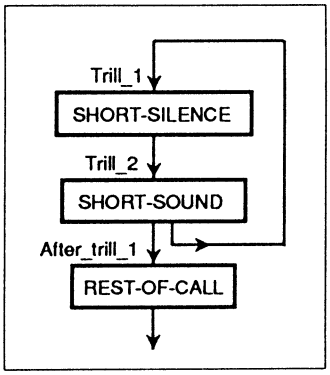


Figure 6.5 Part of a tree showing how a trill or buzz of indefinite length may be composed of a repeated loop containing a short silence and a short non-silent element, and a second branch allowing the loop to terminate.

1 This is not necessarily the correct syntax for the Redbilled Hornbill, it is possible that "double-honks" occur singly or in multiples other than three - the example here is for illustration purposes, although the syntax used is a plausible one.

Clearly this strategy may be used to specify other, more complex, indefinite sequences. What of repeated sequences of limited length? These can be specified explicitly, by using a node in the tree for each repetition of the sound, as for the the three "double-honks" in the Redbilled Hornbill example. For longer sequences, such explicit specification might become tiresome, and a shortcut means of specifiying them in the syntax spec would be useful. A mechanism for doing so has not been implemented in the present BCR system, but is envisaged as taking the form of a loop count associated with a the backward linking branch:

```
Finite_Loop_0:
  Element-A Loop_1
Finite_Loop_1:
  Element-A  After_loop_1 Finite_Loop_0 (5)
```

The hypothetical example is intended to imply that the backward linking branch, "Finite_Loop_0" is applicable for 5 successive times, after which the other branch, which terminates the loop, should be chosen. Additional notation would be needed to specifiy whether the backward loop **must** be chosen while valid, and if not whether the count down from 5 applies only to successive choices of the backward link. Some thought therefore needs to go into the definition of the notation and of the syntax checking algorithm to support the feature. The implementation of this non-essential feature has therefore been deferred until such time as the need for it gains priority.

Repeated sequences in bird calls may be of finite, but indefinite length. That is, they may "legally" range between some minimum and maximum length. Such sequences may be specified as a chain of the minimum length, in the manner of the 3-unit chain of the Redbilled Hornbill, followed by a loop, as in the trill example. That would mean that the sequence must have at least the minimum number of units, and then any number of addional units before terminating. That does not, however, strictly enforce the maximum limit of the sequence length. To do this, the above strategy may be replaced by that illustrated in Figure 6.6 - a chain of the maximum length, with all the nodes from the one marking the minimum length and onwards bearing an "escape" branch which terminates the sequence.

The above example illustrates another "convenience feature" which would be useful in the syntax tree notation, but which is not currently implemented - a sub-sequence specification. It would save labour and enhance readability if the repeatedly used, self contained parts of the tree, when composed of more than one node, were allowed to have a names of their own and thus be specified only once. Instead of having to repeatedly type out identical sequences (in the above case one would have to repeat the two-node sequence "Trill_X" and "Trill_X+1"), a single node could be substituted for each sequence, bearing the name of the newly defined sub-sequence. As in the case of the enumerated loop suggested earlier, the implementation of this feature requires some extra

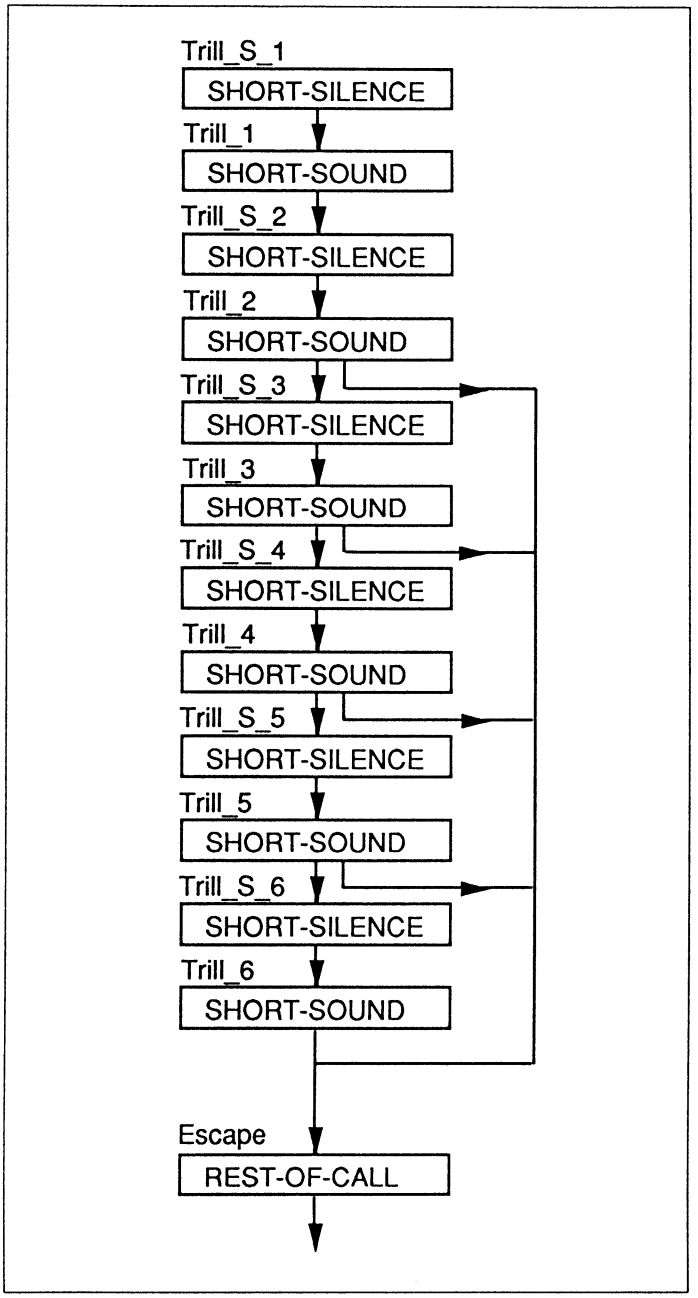


Figure 6.6 Part of a tree showing how a sequence of repeated units, in this case the SHORT-SILENCE/SHORT-SOUND pair, may have both a minimum and a maximum length, in this case 2 and 6 respectively. The "Escape" node marks the end of the repeat sequence and the start of whatever may syntactically follow that sequence.

effort - provision must be made for the sub-sequence's relocatability, for instance, so that it may be followed by different nodes depending on where it is used. The effort has not been justifiable at this stage.

The intention, in bringing these examples of how the syntax tree's features are used, has been not only to clarify the notation but to demonstrate its expressive capability. As stated at the start of this chapter, the fact that a formal grammar consisting of a finite language of symbols and a set of productions (replacement rules) has not been used means that a formal analysis of the syntax spec's expressive power is not readily obtainable. Such an analysis would allow rigourous conclusions as to the types of calls the system could or could not describe. These conclusions would be useful to the degree that bird call's syntactic structural characteristics are understood and available in the literature, which, in general is not the case. Instead it has been necessary to attempt to assess the system's efficacy by empirically considering the information available on a variety of individual calls. The information is mostly limited to short recorded segments and descriptions in field guides - notably Roberts [2].

The assessment was undertaken rather informally, but it is my concluding opinion that the vast majority of calls can be described by the scheme presented here. The fundamental features of branching and repetition are enough to ensure the handling of all the calls considered. It is important to remember that bird calls are limited in their grammatical complexity by the simplicity of their function in nature and by the intelligence of their producers - the birds. The concern for the power of a syntactic system, as expressed by the analytic efforts made in that regard by the pioneers of such systems, was motivated by the considerable complexity of human language. The latter is bound tightly with the issue of human intelligence itself, and suffice to say that a "universal grammar" which can describe the intricacies of any existing human language has, recent claims notwithstanding, yet to be found. Bird calls are vastly simpler, and I do not believe that any major flaws in the current system will surface - although some improvements may well be indicated.

It is informative to note that parallels may be drawn between the features of the syntax tree notation (both existing and suggested) and those of programming languages. Indefinite and definite looping, branching and subroutines can all be compared to similar features in the syntax tree notation. The syntax tree features were provided to cater for bird call characteristics, without reference to programming language features. Yet the final features chosen and envisaged seem to mirror the full range of features required of a generalised programming language, and this fact tends to reinforce the notion that the syntax notation will have sufficeint expressive power.

As a final illustration, a syntax spec for the Southern Boubou (*Lanarius ferrugineus*) is shown. The call of this bird has as high a level of (well-defined) syntactic complexity as has yet been encountered. The reason for this is that the calls are usually duets - an interleaved chain of sounds

produced by a mating pair. Roberts [2] describes a number of different elements produced by the Southern Boubou, and goes on to explain that each bird alternates with one of these elements or sub-sequences, the second bird answering with a *different* such sound, chosen from the same repertoire. Thus the syntax is such that the calls are made of groups of two different sub-sequences, each sub-sequence being chosen from a common pool of possibilites. Figure 6.7 shows a syntax tree that can describe such behaviour. Appendix F contains the full Southern Boubou syntax spec, from which the figure is derived.

The case of the Southern Boubou is also a good illustration of how experts' knowledge of the "rules" of production of a species' call is invaluable in forming the syntax spec. A fully automated learning system, where the computer would "listen" to examples of species' calls and attempt to classify them without expert guidance, would be far more complicated, and likely less successful, than the simple alternative of human-oriented assimilation and formulation of a syntax spec, the route that has been chosen.

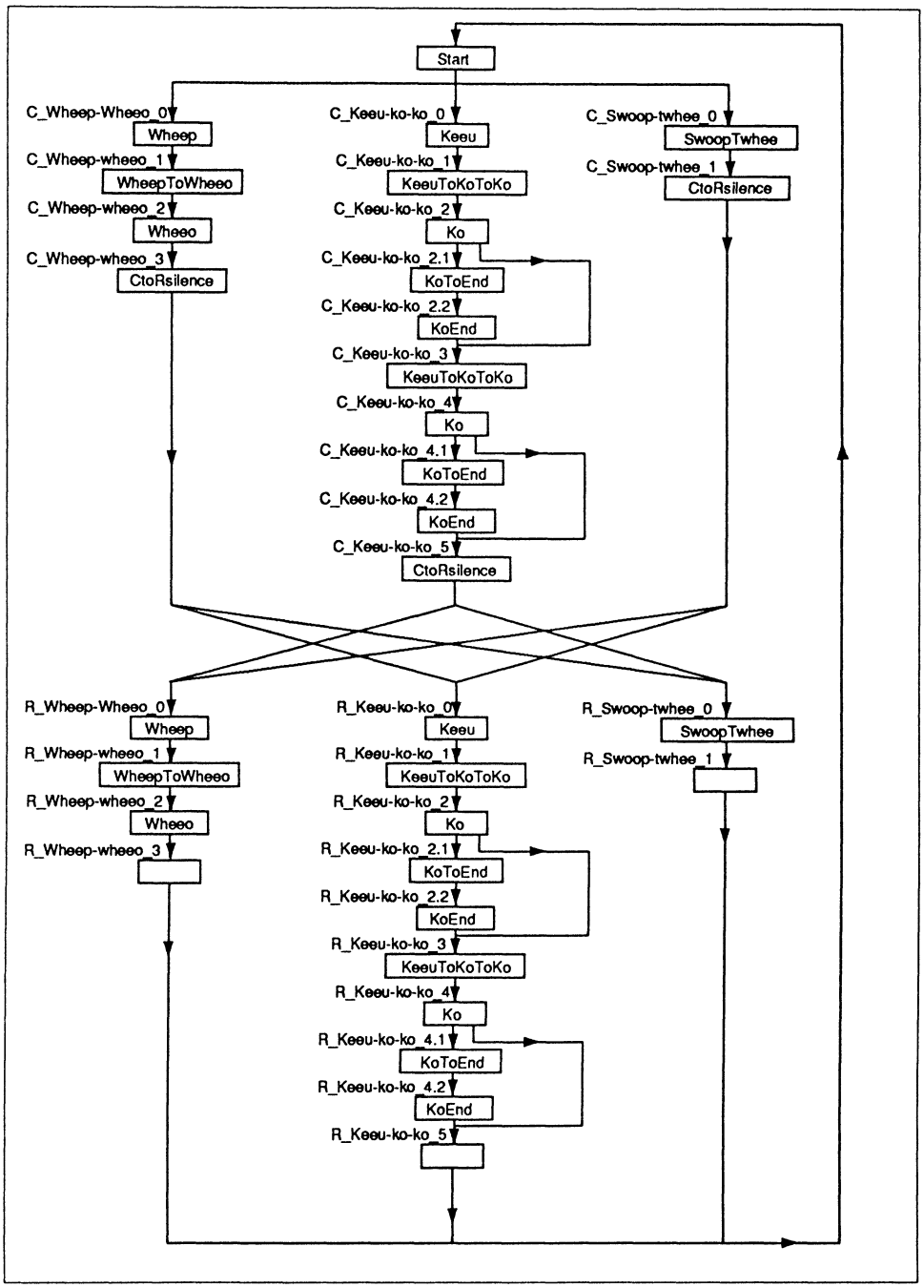


Figure 6.7 Diagrammatic form of the Southern Boubou syntax specification found in Appendix F.

6.2 Evaluating syntax of element strings

The definition of a notation for describing the syntax of a bird call is an important step towards automated processing, but the implementation of the processing is by no means a formality. Many issues remain to be resolved when considering how exactly a call is compared with a syntax spec. Compromises need to be made for practical reasons, ranging from uncertainties and imperfections in the incoming processed bird call, to computing resources available. It is these practicalities, which arise chiefly from the aforementioned differences between the primitive units of the syntax matching system, the elements, and those of formal grammars, the symbols, that give the syntax checking system its unique form. The description of the syntax checking algortihm which follows, however, is best approached from the starting point of the idealised, simplistic mechanism which would implied by the notation were the input an ideal string of symbolic information.

In order to answer the question, "Is this bird call consistent with this syntax specification?", given a particular element string and particular syntax spec against which to compare it, a relatively straightforward set of actions must take place in the syntax checking algorithm. First, a node in the syntax tree which houses an element spec matching the first element in the string is located. The branches leading from that node must then be followed to a set of new nodes whose element specs specify which elements may syntactically follow the current one. Out of this set of nodes, that which matches the next element in the string is chosen. The steps are now repeated, until all the elements in the string have been matched to consecutive nodes in the tree. As long as it always possible to find a node leading from the current node that matches the element following the current element, the element string is syntactically consistent with the syntax spec. In the ideal case, if before the end of the element string is reached it becomes impossible to continue along the syntax tree, then the element string is not syntactically "correct" and the comparison produces a negative result: The call could not have been produced by a bird with the given syntax spec.

The basic element string scanning (*parsing*) mechanism described above is readily illustrated by highlighting the "path" which a string causes the parser to take in the syntax tree: Figure 6.8 shows an input element string composed of ordinary symbols and two syntax trees, one of which will successfully parse the string to completion, thus confirming that the string's syntax is consistent with it. The other tree will not parse the string to completion.

The extra sophistications of the BCR system syntax matching algorithm arise from the deviation of the element string from the ideal, perfectly formed, unambiguous case presented in the above examples. These deviations are now dealt with one by one.

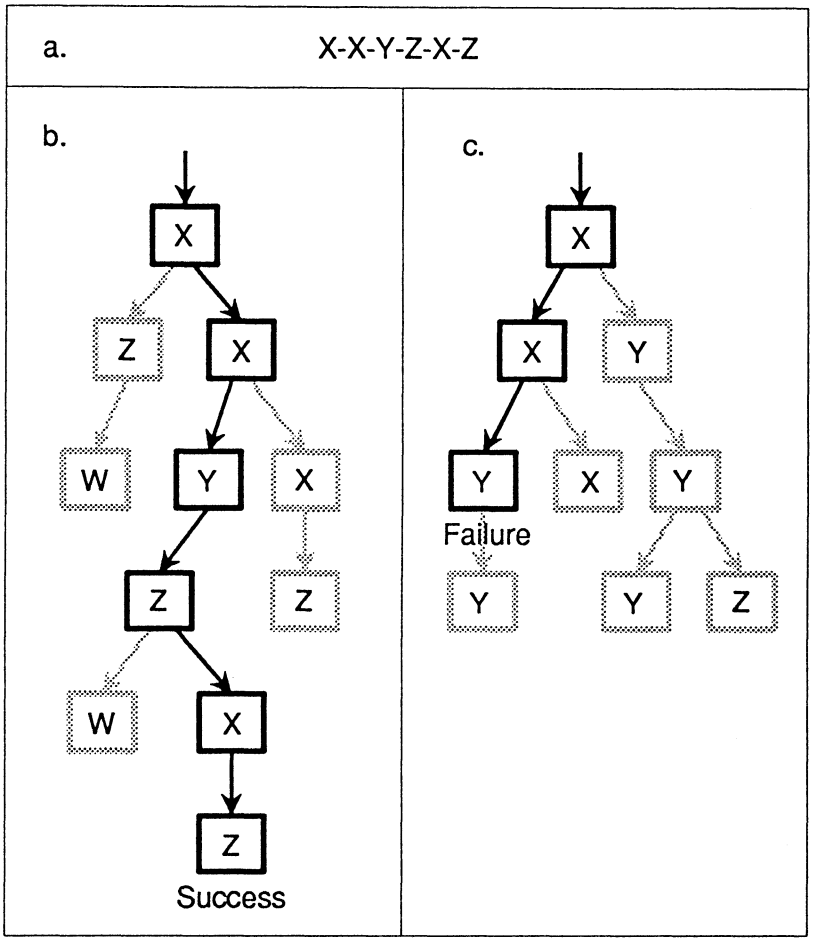


Figure 6.8 An input element string (**a**), when parsed using a syntax tree (**b** or **c**) traces out the path of matching nodes - here highlighted as the bold portion of the diagram. Tree (**b**) parses the string to completion because all the nodes match and occur in the sequence allowed by the tree. Tree (**c**) matches part of the string but then the parse fails when an "ungrammatical" element is encountered.

6.2.1 Entry points

The first practical issue that must be considered is the possibility that the incoming string does not start at a syntactically predictable point. Although the bird itself may be predictable in the manner in which it begins a call, there is no guarantee that the particular sample of the call that has been captured as input to the BCR system starts from that beginning point. This problem might not, on its own, be great enough to justify a special syntax parsing strategy designed to cater for it, as will be presented below. It might be reasonable to demand that a BCR system operator ensure the

quality of the captured input - perhaps by editing a call before presenting it to the system. However, as will be seen later in this chapter, the ability to process a bird call starting at an arbitrary point in the element string is useful and necessary for other aspects of the syntax checking process.

The implications for the syntax matcher of an arbitrary starting point are these: The algorithm must search the syntax tree for a node that matches the starting element in the string - there will be no predetermined starting node. And because there is no reason that more than one node should not match the element, and no way of assigning priority to the matching nodes, the result of the search may be multiple starting nodes. These starting nodes are called *entry points* because they are the places where the syntax tree is "entered" and from which the parsing procedure described previously may commence.

As an example, consider Figure 6.9. It shows a segment of an element string, and a syntax tree which is to be used to parse it, starting at the first element in the string, **X**. Because there are two places where **X** occurs in the tree, these two nodes are the two possible entry points. The figure shows the parsing path possible from each entry point.

Besides requiring a search of all the nodes in the tree before parsing can begin, the added complication of the unpredictable entry point is the question of which of the possible entry points is the "correct" one, when there are more than one. In the example of Figure 6.9 the answer to this question is clear - one of the paths shown does not fully parse the string, while the other (the path marked with the thick solid lines) does. When only one of the entry points leads to a complete parse of the element string, it is easy to say that this entry point is the one "meant" to be used for that string. However, that situation is not always likely to occur. Although two successful paths are not likely, unless the syntax tree has been redundantly specified (i.e. more than one part of the tree describes the same syntax), we will presently see that the possibility of no absolutely successful path being found is quite real. In that case, the simple yes/no decison will have to be replaced by an evaluation of which of the paths is the most successful. This situation is described in the following section, once the reasons for its arising have been discussed.

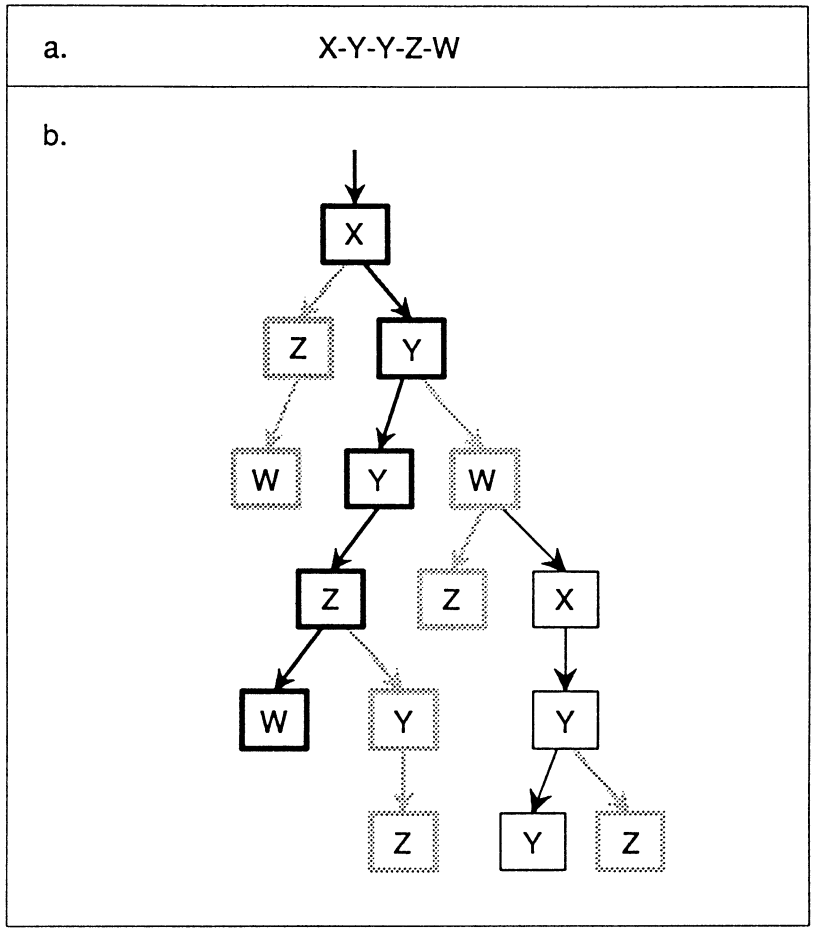


Figure 6.9 An input element string (**a**), may be parsed using a syntax tree (**b**) along a number of different paths, when any node in the tree is legitimate as the start of the parse. The two different paths here correspond to the two entry points that match the first element in string (**a**) and are indicated by the bold lines - one path uses thick lines, the other, thin ones. The light lines constitute the rest of the tree.

6.2.2 Imperfect strings

The most important of the practical difficulties with syntax parsing is the question of the quality of the incoming element string. By its quality is meant its ability to accurately and consistently represent the information in the call, as well as referring to inaccuracies caused by imperfections in the captured sound itself.

As has been shown in previous chapters, the element string is really rather a rough representation of the call. A few broad classes of sound, and a few other crude parameters are used to characterize elements. The detection of the existence (or definition of the boundaries) of an element is done by effectively monitoring the amplitude of the signal - other acoustic characteristics are ignored. These imperfections are not necessarily bad - they mean simpler, faster algorithms, and an easier path to the syntactic analysis stage, thus allowing the possibility that this analysis will compensate for earlier the compromises. But the fact remains that elements in the call are imperfectly represented, and thus the syntax specs cannot be guaranteed to perfectly match the corresponding element strings.

Considering the quality of the captured sound itself, it must be realised that disturbing factors are likely, if not inescapable. These have been mentioned before in the context of the difficulties they cause in the early stages of classification - background noise and interfering bird calls, and other random fluctuations expected from a biological signal. These factors may then combine with the imperfections in the element classification and discrimination process, to produce "mistakes" or inconsistencies.

Of particular importance is the possibility that extra elements get inserted into the stream. They may come from an incidental noise or sound, or a fluctuation in the amplitude whose source is not in the call but in the transmission channel between the bird and the receiving point. The extra elements may be near ends of bona fide elements, where the amplitude tails off, perhaps less than smoothly, past the threshold used to discriminate between sound and silence. Or, as implied by the preceding discussion, the may be legitimate elements that have been badly classified and thus not recognisable via the element specs. Such spurious elements would wreak havoc with a syntax checking algorithm, such as the ideal one described previously, that expected perfectly well-formed strings. As soon as a spurious element is encountered, the string is rejected as "not belonging to this syntax spec", even if the vast majority of the string has been parsed or is parsable.

To cater for these uncertainties, the syntax checking strategy must be made more tolerant of faults and failures. It must no longer simply expect to be able to answer "yes" or "no" to the question of whether a given string is consistent with a given spec. The algorithm must be able to give a probability, assessing the consistency of the string in a graded manner. This will allow relative matching of specs and strings rather than an absolute decision resulting from each spec/string match on its own. The result should be, "String A matches spec B to some degree, while it matches spec C to a greater degree, and thus is probably species C". Thus a scoring mechanism is required, whereby a score is accumulated as the string is parsed. The "better" the parsing goes, the higher the score. When a failure is encountered, the parsing does not stop, but registers the failure and continues. A later section will give details on how the score is obtained, but the implications of continuing a parse after a failure are discussed now.

Let us consider the progress of a typical parse, assuming the spec and the string do match to a large degree. The first element of the string is found to match a particular node in the tree. The next element matches a node leading off from that one, and so on for, say, the first five elements in the string. All is well - the spec is successfully parsing the string. Now the sixth string element is examined - it matches none of the nodes leading from the current node. A failure has occurred. In order to continue parsing, a node in the tree must be found which matches the sixth element - but there is now no information as to where such a node might be located. In effect, the sixth element is like the first of a new string. Like at the start of a string, the algorithm does not "know" where in the tree to start parsing. So a list of entry points like that used at the start must be compiled by searching through the entire tree. See section 6.2.1 for details of this process.

Now the entry points can be used as continuation points for parsing from the sixth element onwards. If the sixth element caused a failure because of a *deletion* (i.e. an element which syntactically should have occurred between the fifth and sixth was omitted), the parse should continue successfully from one of the entry points, while failing soon on the others. In fact the failure is more likely due to an *insertion* (i.e. the sixth is an extra, spurious element) or *mutation* (i.e. the sixth element was a legitimate one that somehow was distorted or not recognised). The consequences of all three errors - deletion, insertion and mutation - are all similar, as will be explained, but deletion will be used as the example here.

The question that arises is how to choose amongst the entry points. As noted in the deletion example, by choosing the right entry point the parsing can continue as it should have, hopefully with no more failures. The other entry points should lead to further failures, because although the sixth element, for which the entry points were found, matches the starting point, the successive elements should not be consistent with the sub-tree emanating from more than one of the entry-point nodes. The problem is that the algorithm is now designed to take failures in its stride. It can no more reject the subsequent failures than it did the first failure. Thus each entry point generated after the failure at the sixth element gives rise in theory to another failure point further down the string. Even the correct entry point could quite conceivably cause a failure, since the basic assumption of the imperfection of the string still holds. So the first failure causes a number of new failures equal, or close to, the number of entry points found. And each of those failures must now have an associated set of entry points generated, resulting in a new set of failures and so on. The single failure can exponentially multiply the work of the algorithm to absurd, time- and memory-consuming lengths.

The explosion of possible paths spawned by a failure must be limited by choosing amongst entry points before continuing to explore further possibilities. An entry point is best if parsing can continue from it to the end of the element string. If no entry points do so, then the one which ultimately leads to the end of the string with the least failures is best. To determine which of the entry points fulfills this criterion absolutely, all points, and all failures resulting from them, and

all successive failures and so on must be tried, until the end is reached, while counting the number of times a failure occurred along a particular parse path. This is the explosive situation depicted above. Rather than follow all possible alternatives to arrive at the absolutely correct result, the algorithm could try guess which entry point is best using a compromise criterion. For instance, the entry point that allows the string to parsed the longest before encountering another failure might reasonably be expected to be the right one. It would seem to allow the end of the string to be reached more quickly, thus reducing the number of failures that can occur (the maximum number of ailures than can occur along a string is equal to the number of elements in the string).

A choice based on the above *segment-length* criterion, where the entry point that leads to the longest parsed string segment is chosen, allows the falure-tolerant algorithm to proceed linearly with the minimum of alternative-checking. At each failure, the entry points are found, and parsing is tried at each of these points. The one which "takes" the algorithm farthest along the element string before failing is accepted as the correct choice. The other alternatives are discarded. Then a new set of possible entry points are generated for the failure at the end of the latest segment, since a segment must end either with another failure or with the end of the string. When a segment ends with the end of the string without failing, the parse is complete.

The above algorithm is the fastest fault-tolerant one, and is used in the BCR system. It is not, however, fool-proof. It is possible for an entry-point to lead to a longer segment than a rival entry point, only to result in a bevy of failures later, while a shorter segment may actaully have been the correct one to choose, leading to less failures in the long run. This possibility of "barking up the wrong tree" is, however, not a particularly likely occurrence. In most cases, one entry point will give a segment reasonable length, while the others fail almost immediately, because the parser finds itself in the wrong part of the tree. It is only in a tree that contains elements with a broad specification that might allow a string to proceed along the "wrong" track for a while before finally going badly awry. Unfortunately, the very uncertainties mentioned as possiblities in the element string at the start of this section mean that a broad specification for the syntax spec is sometimes necessary, so the algorithm's flaw cannot be neglected for all time. Some suggestions for future improvements are given below.

As a compromise between choosing between entry points immediately after the first failure generated by each one occurs, and not choosing until all subsequent alternatives due to all possible failures have been explored, it is possible to follow the alternatives until a fixed number of failures, greater than one, have occurred. Thus after failure number 0, a set of entry points (0a,0b,0c) is generated. Each entry point is tried until another failure occurs, leaving a set of failures, one for each entry point, (0a1, 0b1, 0c1). In the BCR algorithm, the choice between 0a, 0b and 0c is made here. But it could delay the decision for another level: It could generate entry points for each of the level-1 failures. Failure 0a1 would beget (0a1d, 0a1e,0a1f) while the others would be called

(0b1g, 0b1h, 0b1i) and (0c1j,0c1k,0c1l) assuming three entry points for each failure. Now each of the nine entry points could be tried until failures (0a1d2, 0a1e2, 0a1f2), (0b1g2, 0b1h2, 0b1i2) and (0c1j2, 0c1k2, 0c1l2) were reached. The name of the failure point as constructed here indicates its "level" (the number of failures that have occurred along that particular parse path, or eqivalently along the length of the string) as the rightmost digit. The letters embedded in the name show the entry points that were used to reach the failure point.

The algorithm can now see how far down the element string it can go by following each of the paths leading to the found failure points. The longest segment thus obtained gives the entry point, out of the set (a, b, c) that chould be chosen. For example, if failure 0b1h2 is at a point furthest down the string, entry point (b) is chosen. The decision at this stage would be a 2-segment-length algorithm, because two segments (or subsequent failures, or entry points) are generated along each path to a particular failure point. In general, an N-segment-length algorithm is less prone to error, but exponentially more resource-consuming, the greater N is, where the maximum value for N is the number of elements in the string.

A more intelligent choice of entry points might allow the speedy parsing of the 1- or 2-segment-length algorithm while reducing the possibility of "barking up the wrong tree". Instead of choosing only one of the entry points, all points which give rise to segments of reasonable length might be chosen. Thus if some points fail immediately, but two points give segments of length, say, 3 and 5 elements, both the latter entry points should be retained for further processing. It is possible that this strategy would allow the alternatives to be followed all the way until the end of the string is reached without generating an excessive number of alternatives. This possibility would have to be empirically explored as it depends on the bird calls, and the syntax specs.

To end this section the point about similarity between effects of deletion, insertion and mutation must be made. An extra element in the string causes a failure just as the element after a missing one does. A new set of entry points is then sought, to match the element at which failure occurred. The only difference between deletion and insertion is that with the latter, the possibiltiy is much higher that no entry points will be found, or that none of them will lead anywhere. So another failure is likely, after which the following element is just like the first failed element in a deletion - parsing is likely to continue normally from there. A mutated element causing a failure is like an inserted spurious element. It causes a failure, entry points are sought and probably not found, causing a second failure, and then parsing moves on to the following element in the string. Thus the difference between deletion and the other two forms is that the former is likely to cause a single failure, while the latter probably causes two in a row.

6.2.3 Imperfect elements, and scoring mechanisms

The previous section dealt with the syntax-parsing problems caused by less-than-perfectly formed element strings. This ill-formation was caused by extra, missing or modified elements in the string. The reasons these effects occur was explained by the combination of imperfect classification and discrimination of elements and natural randomness in the captured signal. The syntactic discrepancies caused by the imperfections were handled by making the parsing of the tree more tolerant. The present section seeks to show how the matching of the elements to the element specs can also be made more tolerant, and how this leads to a scoring mechanism which can be used to increase the flexibility and tolerance of the recognition process as a whole.

In truth, the element matching process has already been made tolerant by the use of a range of values in the element spec. But this is intended to be tolerance for the natural variation in element characteristics. What is now needed is tolerance in the face of error or lack of information, i.e. the shortcomings of the element classification system. It is necessary to allow the syntax matching process to continue for as long as possible. Rather than let problems with earlier stages hinder the possibly redeeming action of the syntax checking stage, the matching process should record the degree to which elements match, and even if the degree is small, allow the parse to continue. Then, when all candidate syntax specs have been compared to the element string, a delayed decision can be made based on the efficiency of the parse and the degree of matching achieved.

The required "measure of degree of matching" is simply a match *score.* How should a match between element spec and element be scored? One possibility is to measure the degree to which the element corresponds to the nominal element spec values. Thus if the element's class profile is a small distance from the profile in the spec, the score could be large. Similarly if an element frequency is close to the middle of the frequency range in the spec, a large score could be set. Another idea is to allow partial matches. An element whose entire set of parameters (characteristic frequency, duration and class profile) matches those of the spec might be given a large score, while one in which only some of the parameters match might get a lower score but not fail altogether.

In the BCR system, the latter method has been used as a starting place for the scoring mechanism. The former idea, of closeness to nominal parameter values, is not implemented, partly because it would require some study as to what an appropriate weighting scheme for the score should be. For example, how much more important is duration than class profile? Importantly, a closeness score implies something of a partial match scoring effect, since a seperate value must be calculated for each of the elements' parameters before adding, presumably, to a single total. The partial match scoring scheme is currently simple and general - it can be extended to include closeness scoring and other weighting schemes if desired, and explicitly allows the various parameter matches to be handled separately.

The scoring scheme is straightforward. Non-silent and silent elements are treated seperately. If the duration of a non-silent element matches that of a non-silent spec, a score, or *match level*, of 2 is assigned. If, in addition, the frequency matches, a match level of 3 results. If the class profile also matches, the resulting level is 4. If a silent element matches a silent element spec in duration, the score is 4, just like a non-silent element that fully matches a non-silent spec, since duration is all there is to a silent element. If the duration does not match, a non-zero level is still awarded, for reasons to be given presently. The value of this level is 1.

As can be seen, the scheme above prioritizes the match in a rudimentary way. A "full house" of matching parameters gives a higher (which is meant to be interpreted as better) score than a partial match. A fully matching silence is considered to be just as important as a fully matching non-silent element. These considerations are fairly self-evident. However the scheme as implemented in the BCR system has certain restrictions. While a duration match on its own is acceptable (level 2), an isolated frequency match is not. Understandably, a frequency match together with a duration match has a higher score (3) than duration on its own, but if frequency alone were to fit, the element would not be considered to match (match level 0). Similarly, a class profile only contributes to the match level if both duration and frequency are found to fit, otherwise it is disregarded.

The scheme as described here is a reasonable starting point, but is not really based on a meaningful survey of the match situations that are likely to occur. Without a suitable duration and frequency, the class profile is probably not likely to have much value, so the scheme is justified in ignoring it. But whether the frequency can be "right" without the duration matching is not clear, so the prioritization here is somewhat arbitrary. A more general scheme, which should therefore have been implemented in the BCR system if only to preserve complete generality, would be to have a different level for every match combination. Since there are three independent matches, those between duration, frequency and class profile, and each match can be either true (does match) or false (does not match), the possible number of match levels is $2^3=8$.

The eight possible match levels could then be prioritized by a weighting function (or table) so that those combinations that are better indicators of element identity could be contribute proportionally more to the score. The determination of the weighting function would probably best be done empirically, thus requiring substantial analysis of the parsing process for many calls. As a first approximation, allowing the syntax stage to be completed and further issues thus exposed, the simple 4-level prioritization scheme described suffices, and will be the basis for the remainder of the discussion.

Returning to the BCR match scoring scheme, then, the special match level of 1 for silent matches must be explained. A level of 1 is assigned whenever a silent element occurs in the string and matches with the **occurrence**, but not the duration, of a silent spec in the tree. Thus the scheme

gives a score for a simple correlation of **class** between an element and the syntax spec, where silences are concerned. For a non-silent element, it is not enough that its class should match with a node leading from the current one, it must at least match in duration as well, in which case it gets a score of 2. For generality, a class-only match also scoring 1 should perhaps have been allowed for non-silent elements. Incorporated into the general scheme suggested above, this would then mean an additional match parameter, the overall class, which would raise the number of possible match combinations to 16.

As it stands, the level of 1 is reserved for silences because it is considered that the class-only match is not enough in a non-silent situation. Such a feature would lead to, say, an isolated harmonic frame forming an element of 10 ms in duration, being matched with a seconds-long cry in which the majority of frames are harmonic. It was felt that such a comparison should result in a failure rather than simply a low score. On the other hand, a similar mismatch between a short silence and a longer silent spec should be not considered as serious. The reason for this is to be found in a "silence-splitting" phenomenon described in below. For actual examples of this phenomenon please refer to the call tracks in Appendix F.

The phenomenon occurs when a spurious non-silent element, perhaps a single frame, breaks up a silence between two bona fide elements. Let us call the first bona fide element A and the second one B, the spurious one X and the silence S. The syntax spec "expects" a string of elements such as "...ASB..." but receives as input "...AS_1XS_2B...". S_1 and S_2 are the two parts into which S has been split by X. If the syntax checker were to reject S1 when comparing it to the spec for S, assuming that S_1 is much shorter than S and thus does match its duration, then the segment parsed would be "...A", with a failure occurring on S_1. Note also that a second failure might occur if no other silence were found to match S_1 on the subsequent search for entry points. A third failure could occur on X, and again, if X is completely unmatchable elsewhere in the spec, a fourth failure results. Similarly a failure or failure pair is caused by S2, and parsing resumes with the segment "B...". Thus the worst case effect of the spurious element X is to cause 6 failures.

With the class-only feature however, where S matches S_1 and S_2 (albeit with a lower score), the first segment parsed is "...AS_1", X generates one or two failures and S_2 does not generate a failure, because it is parsed as part of the segment "S_2B....". The effect of the class-only feature has been to place the "blame" where it belongs, with the X element, and not to generate failures which seem to be due to silences but which are, in fact, logical consequences of the X failures. The only failures registered (a maximum of 2) are those due directly to X's inappropriateness in the place it was found, and possibly also the fact that it doesn't belong anywhere else in the call either.

The phenomenon could also occur in the opposite way: A spurious silence interrupting a bona fide element, splitting it into two unrecognisable elements. Again, 6 failures are possible in such a situation. Why, then, is a class-only match not justified for non-silent elements? The answer is that this non-silent splitting is a far less likely occurrence than the splitting of a silence. In the interior of a bona fide non-silent element, the amplitude is greatly in excess of the background noise, and is a continuous function of time - not prone to sudden, unplanned drops in level. A bona-fide silence, on the other hand, is always vulnerable to splitting by any incident noise - since it is a "non-element" there is no requirement that the silence be a continuous entity. By including a class-only feature for non-silent elements, we introduce many problematic ambiguities (see later section on ambiguities in the syntax tree) to correct a rare anomaly - an undesirable step.

Scores

The match level scheme was introduced at the start of this section as a means of increasing the system's tolerance to imperfections in elements. It is however also a means for measuring the syntax spec's overall fit with the element string. These two features - tolerance to imperfections and a scoring mechanism - go hand in hand and it is not necessarily true that match levels are more a product of the need for the former than for the latter. The one facilitates the other, and having established the match level scheme in the one context, it is a simple matter to describe the scoring scheme. In addition to the score derived from the match levels, the measure of syntactic performance, derived from the failures described in section 6.2.2, is explained here together with the other scoring mechanisms.

By adding up the levels obtained for each element in the string as it matches one of the element specs in the syntax spec, a total measure of the *appropriateness of the incoming elements* is obtained. The measure has little to do with the the syntax, except that when an element matches more than one of the specs, the specs along the syntactically best path are the ones that actually contribute to the score. Thus if element A matches spec X with a level of 2, and spec Y with level 4, but Y does not appear in the syntax path chosen by the parser, then the contribution of element A to the appropriateness measure will be 2. The appropriateness measure is thus indicative of how well, on average, the elements matched the particular element spec that were required to syntactically parse the string.

A second measure, the *preliminary score*, can be obtained without any syntactic processing at all. It consists of simply comparing all the elements in the string with all the element specs, choosing the best match for each element when more than one is possible. The score for each match is added to a total which is then divided by the number of elements in the string, to give a measure that is independent of string length. This gives an indication similar to the appropriateness measure described previously, except that it is in effect the maximum possible such score, rather than the

particular one achieved by the parse. While the preliminary score is thus less specific in its evaluation of the call, it has the advantage of not requiring a syntactic analysis. It is therefore useful as a quick sifting mechanism when a number of syntax specs are to be checked against a particular input string, as is the intended in the BCR system.

Since the preliminary score is to be used as for fast initial sifting of candidate syntax specs, it is appropriate to include a check for essential elements in this scoring mechanism. Essential elements are those that always appear in the calls of a particular bird, without exception, and are of a sufficiently stereotypical form as to be consistentently recognisable. Such elements, if they exist, allow the BCR system to reject a candidate syntax spec immediately if its essential elements are not found in the input string. Element specs can be specifed as essential by following the element title with the keyword **essential** in the syntax spec. The essential element feature allows those calls that can be recognised chiefly on the basis of sound quality, with less significance in the syntax, to be quickly handled, or their specs to be efficiently by-passed when possible.

The third and final score is the one which indicates the degree of syntactic identity between spec and string - the *failure count*. This is simply the record of all the failures to match an input element with a syntactically expected element spec. A string which is syntactically perfect should be parsed by a the appropriate syntax spec with zero failures. Thus the higher the failure count, the greater the mismatch between string and syntax spec.

The failure count and appropriateness measure complement each other. The latter is a positive score which grows as more of the call is matched to the spec. It does not however matter whether the matches take place in a continuous path along the spec or at isolated points in the spec. The failure count, on the other hand, is a negative score which grows as more of the call is mismatched. It is not influenced by the degree to which the string and spec match, only by the continuity of the parse. Taken together, then, these measures result in a more complete means of gauging the fit between string and spec.

That said, the failure count must be considered the weightier of the two measures. It is possible for a high appropriateness value to be obtained in a comparison between a string from one bird and a syntax spec of a different species. This situation can occur when the two species have similar elements, but use them in a completely different way. It can also occur when the element specs in the syntax spec are so broadly defined as to match a great range of elements. The converse, however, is not possible - a call whose elements do not match many of the element specs in the syntax spec cannot be expected to be parsed without many failures. Thus when a high appropriateness measure is obtained for a string, it should not be considered a valid indicator until it is confirmed that a low failure count was also obtained.

The emphasis on the syntactic component of the score reflects the underlying direction taken in building the BCR system - the reliance on syntax as opposed to pattern matching of sounds. It is also a realistic emphasis to make, when it is considered that the primary use of the BCR system will be in differentiating between species that cannot be reliably identified by other means. Often this means that the species are alike in colouring, habits and habitat - and are thus highly likely to be part of the same family. Their sound producing apparatus is likely therefore to be similar, and the sounds they make will have a common evolutionary source. It is in the higher-level vocalization functions - the call syntax - that they will have had room to vary and diverge most easily. Evidence of this phenomenon - similar birds with similar habitats producing similar types of sound but in different combinations - can be seen in many familes. Some examples are Cuculidae (Cuckoos, Coucals and others), Caprimulgidae (Nightjars) and Sylviidae (Warblers, Cisticolas and others).

Table 6.1 at the end of this section (6.2) is an example of scores obtained by various calls in the BCR system.

6.2.4 Ambiguous branches

A final practical problem which can arise in parsing using a syntax spec is that of ambiguous branches. A branch point, being a node in the syntax tree leading to more than one possible subsequent node, contains an ambiguity when the element being checked matches more than one of branch nodes. Thus if node **A** is followed by nodes **B**, **C**, and **D** in the tree, and element **a** in the input string **...ax...** matches node **A**, then if **x** matches both nodes **B** and **C**, the node A constitutes an ambiguous branch point.

The problem posed by branch ambiguities is the same as the one associated with multiple entry points - the explosion of possible alternative parsing paths. If the decision as to which of the branches is the "correct" one is postponed until the entire string has been processed, an exponential proliferation of alternatives could result. The different paths implied by each ambiguous branch must be remembered until finally the best one can be chosen. This approach puts a great strain on the computing resources of time and storage space. If anything, the potential for proliferation is even greater for branch ambiguities than for multiple entry points, since the latter occurs only on failure, while the former can occur even at "successful" nodes, and within syntax tree loops.

As with the entry point problem, a means to decide between alternatives (in this case, branches), immediately upon encountering them, is needed to cut short explosive growth. The match level mechanism offers a ready-made solution. The branch with the highest match level is certainly the one which, if considering only the immediate step in the parsing path, is the most likely to lead to a good parse. Thus just as the algorithm used for choosing between multiple entry points seeks to find the path with the least failures (by choosing entry points with the longest associated segments),

so does one based on match levels seek the highest-scoring path. As with entry points, the BCR system currently implements a one-level decision for ambiguities. A higher-level decision would mean checking not only the current branches' scores, but those along the resulting paths, although in a compromise situation, not all the way to the ends of these paths.

Unlike multiple entry points, however, ambiguities are not altogether an unavoidable evil. The fact that an element can match more than one element spec is most likely a result of a badly designed spec. If two elements are distinctly recognisable, and cause distinct representations in the BCR system, then distinct non-overlapping element specs can theoretically be defined. If that has not been done, it is likely due to too lax a specification, probably because a small data set did not allow confident setting of the element's parameter range limits. As an example, suppose element **a** is observed to occur with a characteristic frequency between 600 and 630 Hz. An element spec designed to match this element might specify a frequency range of 590 to 640 Hz if the observer were not confident that no elements he recognises as **a** occured with a frequency outside that range. Then an element, **b** with an actual variation between 640 and 660 Hz would match both its own spec and that of **a** to some extent.

The point, then, is that with care and exhaustive data ambiguities can be eliminated. But because such care in the specification cannot be guaranteed by the BCR system (which is not part of the specification process), and especally because a large data sample cannot always be assumed to be available, the BCR system retains its rough mechanism for resolving ambiguities. In most cases the mechanism is likely to make the right decision, because the likelihood of overlapping ranges in all the element spec's parameters is much smaller than that of a single overlapping parameter range, such as the frequency example illustrated above. In most cases, then, a partial overlap results in a partial match, and accordingly a lower match level, while the correct element spec matches more fully. Only when the match level is the same in both cases is a serious ambiguity encountered (in this case, the BCR system arbitrarily chooses one of the branches).

Because the problem of ambiguities is potentially solvable by the specifer of the element specs, the BCR system would do well to report the occurrence of such ambiguities. The experimenter could, in this way, find and concentrate specifically on problem areas, without having to be unnecessarily meticulous about every spec. Such a feedback feature would be a simple matter to implement, requiring no significant changes to the BCR system code. The fact that the BCR system does not currently have this feature is therefore an oversight - one which has prevented an empirical analysis of the influence that ambiguities may have had on the reported results obtained by the system. The BCR system does have extensive feedback features to allow detailed analysis of the progress of a parse. The utility of these features is, however, largely in the area of validation of

the correct functioning of the algorithm, rather than part of the experimental analysis of bird call recognition. The features are therefore described in the technical discussion of the the syntax matching algorithm, in Appendix A.

6.2.5 Syntax match trials

To conclude the description of the principles of syntax matching in the BCR system, the results of a set of preliminary trials are presented in Table 6.1. A small number of representative calls were captured and processed until element strings for each were obtained. The strings were then examined and their contents used as the basis for syntax specs for each bird. The specs are by no means complete in describing the calls of the species. Rather they are intended to match a generalised form of the particular string from which they were taken. The generalizations were made by comparing the captured call to the sonagrams and descriptions of species' voices given in Roberts [2]. Elements in the string which matched those mentioned in Roberts were used to define element specs. Elements which did not appear to be significant according to Roberts were regarded as spurious and were not included in the spec. Any abstract syntactical information provided in the text was also incorporated.

Some generalization (in the form of expanded parameter ranges) was also introduced into the individual element specs to account for variability. Either the variability was apparent in different instances of an element in the captured call, or if multiple instances were not present, it was assumed and arbitrarily estimated. Important elements shown in Roberts but not found in the captured call were included in the spec by roughly estimating their parameters from the Roberts sonagrams.

The resulting syntax specs are useful in validating the operation of the BCR system, although they do not necessarily allow the system to perform at its maximum potential. They are tailored to recognize the particular sample calls on which they are based, but take the form of a spec that could recognize a whole range of calls of the species. Their construction as described here demonstrates the principles by which truly general syntax specs, that could be used in a working recognition system, would be obtained. Essentially it involves the same process of examining particular calls and integrating the parameters thus obtained with overall knowledge of the call available in references on bird vocalizations. The difference between the process used for the trials and that required for "real" specs lies mainly in the number (and variability) of the samples of calls used for each species, and in the detail and specialization of the reference sources on the vocal characteristics of the bird. Some time might also be required to be devoted to inferring new syntactic knowledge about the calls, since in the absence of work on automated bird-call recognition, such knowledge may often be incomplete or not available.

The trials presented here are not conclusive evidence of the success or failure of the BCR system to recognize calls. They prove the practicability of the syntax spec system, showing how such specs can feasibly be assembled and how, even when the assembly is sketchy and less than rigourous, they can be used to distinguish between bird calls with some success. They are a good argument in favour of further effort in assembling specs, conducting trials with more appropriate sets of data (such as groups of species of similar habitat and appearance), and refining the BCR system.

The syntax specs, along with the different processed stages of the calls from which they are derived, are to be found in Appendix F.

Interpreting Table 6.1

The plots in Table 6.1 represent the scores output by the BCR system as density (shades of grey). The score sheets in Appendix F contain the actual score values used to assemble the plots. Each row in a plot records the result of comparing one call segment with all twelve syntax specs. Looking along a row, then, one should expect to find one square in which the score is best (that is, highest (lightest) in the case of plots (a) and (c), and lowest (darkest) for plot (b)). The syntax spec for the column intersecting the row at the "best" square gives the name of the bird that the BCR identifies as matching the call. Both rows and columns are identified by a four-letter bird code which can be used to find the call, spec and recognition scores in Appendix F.

Thus in plot (a), the first row (corresponding to the call of the Monotonous Lark) has a clearly lightest square in the last column, which corresponds to the syntax spec derived from the Monotonous Lark's calls. This indicates that, based on preliminary score alone, the BCR system correctly identifies the Monotonous Lark given a choice of eleven other species.

The plots have calls and specs arranged in the same order. Therefore, by looking at a plot as a whole, rather than at isolated rows, one may obtain an impression of the overall performance of the BCR system (for each type of scoring system). If the best scores all occur along the appropriate diagonal, the system has had a high rate of success in identifying the calls. However, it is important to retain a row-oriented perspective, since the rows represent the way in which the BCR system would be used. Looking along a column tells whether a particular syntax spec can pick "its" call out of a set of different calls, but that has no particular implication for the envisaged call recognition system.

After studying the plots it can be said that the BCR system performs as desired for the rather special conditions under which the trials were performed. The preliminary score, which reflects the best scores obtainable for each element, regardless of number of times, or the order in which, the element actually appears in the call, clearly identifies each call correctly. It's success might have been

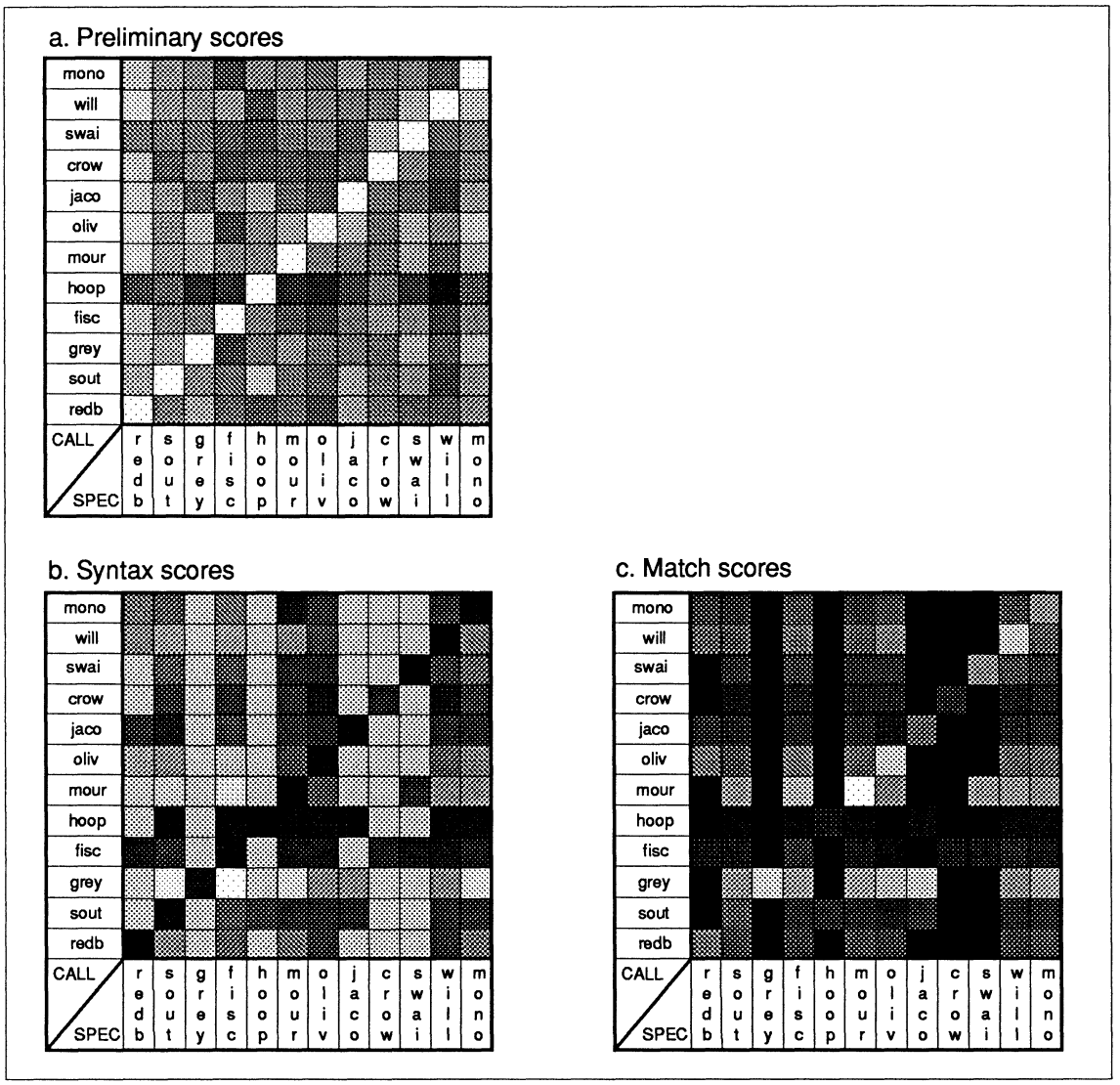


Table 6.1 Scores obtained by a variety of bird calls in BCR system trials, where each call was compared by the system with a number of different syntax specs. See following text for interpretation of the plots. The calls and the specs are those of Appendix F.

unexpected (considering that the emphasis in the BCR system is on the syntax rather than accurate sound classification), but is understandable when one considers that the element specs were guarantied to match the elements - they were derived from the same call as the one which they were later used to evaluate.

The syntax scores (which count failures and therefore are best when lowest) perform almost as well as the preliminary scores, again probably better than in a genuine field situation due to the fact the syntax tree was guarantied to cover a good portion of the input call's behaviour. The match scores do not show as marked an effectiveness at picking out the correct syntax spec, since the diagonal line of squares is not as emphatically evident. This is due to two factors. Firstly, the match score reflects a mixture of the effects recorded by the other two scores. On the one hand it shows how well elements matched their specs on average, and on the other hand it takes into account only those matches that occur along the recognition path. Thus while element A may match element spec B with a score of 4, it might be better from a syntactic viewpoint to pair it with element spec C, which might have a lower score of 2. So Match scores have a tendency to be lower than preliminary scores, but possibly more realistic.

The second factor is the main one contributing the to the fact that the diagonal line is not sharply contrasted with the rest of the plot. It is simply that, unlike the preliminary scores, the match score is not normalised by dividing by the length of the call. This normalization is not required since as mentioned, we are interested in whether a square stands out compared to other squares in the **row**, that is compared to the scores of the same call with different candidates. Indeed, when looking along the rows, many of the correct syntax specs do stand out sufficiently. However there can be little or no comparison along columns, so the overall diagonal effect is not as evident. Examining the score values themselves rather than the plot shades, the match score can be seen to be an effective a means of pin-pointing the correct winning candidate, if even small differences in scores are taken into account.

The effect of preliminary rejection of syntax specs based on the absence of essential elements is reflected in the syntax score and match score plots by an artificial injection of a bad score (since the syntax checking that generates these scores was not performed). The specs that contain "essential" specifications are *redb, grey, hoop, jaco, crow* and *swai*, and the corresponding columns in the plots can be seen to contain mostly uniform bad scores in appropriate places. The correct syntax spec is never rejected by the preliminary mechanism (indeed, the essential elements should always be chosen so that there is no chance of this happening). The plots show that "essential" specifications can significantly reduce the number of full syntax checks that need to be done.

The fact that the syntax score plot does not offer more information than the preliminary score plot might lead one to the conclusion that the entire syntax checking system is unnecessary. That is not the case, however, because as mentioned, the preliminary score will perform as well as it did only under ideal conditions. The expectation is that under practical conditions it is the ability to recognize elements (as reflected in the preliminary and largely in the match score) that will deteriorate, leaving the syntax of the call as a strong clue as to its identity. Also, an important factor to note is that in these trials, the sample calls were chosen mostly to represent the range of sound types

present in bird calls. There is no good example of two calls which have similar elements, arranged in different ways. In such cases, of which a significant number are expected in the field (birds of related species often differentiate between themselves in this way), the syntax score will be indispensable.

To conclude on the effectiveness of the BCR system and the scoring mechanisms, it can be said that by using all three scores (or perhaps only the preliminary and syntax scores since the match score reflects a combination of sorts and is less definitive) a very clear recognition ability is evident. This would imply that, while recognition in a field situation would probably be less efficient, it should certainly be possible, and worth attempting considering the scope for improvement existing in BCR system.

6.3 Syntax matching software: SYNTAX

The software which handles the syntax matching, the "brains" of the BCR system, is the SYNTAX module. It is described in Appendix A and Appendix B

7
Conclusion

In this chapter a summary of the important observations, decisions and results of the preceding chapters is presented, often with additional concluding observations. Following the summary is an evaluation of the successes and failures of the BCR system as it stands. Lastly, implications of the system for future research are discussed, with some speculation and ideas on direction and improvement. The overall goal for this project, namely a framing of the bird-call recognition problem, providing a quantitative basis for further progress, is seen to have been achieved.

7.1 Chapter summaries

The following sections are summmaries of the chapters 2 to 6. The number of the chapter being summarised is given in parentheses in each section heading.

7.1.1 Background (2)

The available information on bird calls is limited, especially in the areas important to this study, such as the classification of the sounds. By implication then, research into methods of classification and quantification of bird calls is a worthwhile endeavour.

While there are similarities in the production mechanisms of sounds in birds and speech in humans, there are also important differences. The rigidity of the bird's syrinx and muscular limitations mean that information in the bird call is less likely to be encoded in the varying spectral shape of the sound (as it is with speech). Instead, variation occurs in basic pitch, or in the temporal arrangement of sound elements. The spectral content is significant, but not highly variable. This makes automatic recognition potentially simpler for bird calls than for speech.

A consideration of the practicalities of communicating recognizable information over distance in the field, and of the sound producing and receiving apparatus in birds, leads to a number of practical starting points for a bird sound processing and recognition system. It should be frequency and syntax oriented, and various parameters required of the system can be estimated, such as the frequency range, frequency resolution and time resolution.

Speech recognition techniques, reviewed because of the possibility of their applicability to bird calls, are probably too specialized and expensive to have direct application. The bird call recognition system should be tailored to the special characteristics of the bird calls, if an efficient and compact

solution is to be achieved. Certain broad principles of speech recognition, such as the use of peaks in the speech spectra, and the division of the speech signal into separately classified units, are sure to be found in a bird call system.

Having discarded the idea of using a speech recognition system directly, the field of general pattern recognition becomes of interest. A brief examination reveals two broad categories of pattern recognition, those of template matching and syntactic methods. The former is a largely numeric approach, while the latter employs symbols to greater degree. It is concluded that the structure of a bird call lends itself to a simple syntactic recognition format. The seeds for the form of the bird call recognition system are sown here: The call is to be translated to a string of symbols, using some partially numerical techniques, and the string is then to be processed by an automaton which applies various rules, leading, in a manner similar to that of a formal syntactic recognition system, to a classification of the call.

The syntactic model, while being inherently suitable for bird calls, has some important features which enhance its applicability. Its symbol-based mechanism is intuitively clearer than the low-level numeric mechanisms of other methods. It therefore allows a certain amount of intervention by a human operator. This is important in designing an experimental system, and also allows the problem of recognition to be tackled without also having to deal with the more difficult problem of learning. At least at the experimental stage, and possibly for a working system, the "knowledge" that is used by the system to recognize calls may be entered by the operator, rather than obtained automatically by the system. In a case, such as with bird call recognition, where such knowledge is more easily available in its abstract form (that of human expertise) than in its raw state (that of comprehensive data samples), the operator-teachable system is highly advantageous.

7.1.2 Computer-based bird call analysis (3)

In manipulating bird calls by computer, the first issue to be settled is that of representing the bird call. The primary representation alternatives are the time domain, with its capability of pin-pointing the exact moment at which a change occurs in the sound, and the frequency domain, which can indicate and quantify the presence of repetitive patterns.

Since a bird call contains elements of both variable and repetitive character, neither representation system will do in its pure form. To see how the advantages of both may be combined (with some attendant disadvantages), some real-world bird-call representation systems are examined. They are the ear and the sonagraph, whose underlying mechanism are those of the filter bank. By monitoring the outputs of a bank of bandpass filters, the presence of different frequency components in a signal can be measured to a certain degree, and should the frequency content of the signal change, this time-variation will naturally also be apparent to some extent at the filter outputs.

The practical and theoretical limitations of a bandpass filter lead to the characteristics which can be expected of our representation system: There will be a blurring and a mixing of information. Rapid changes in the input signal will appear more gradual, just as a sharp rap on a bell produces a drawn-out note. Should we wish to minimise the latter effect, we must trade off frequency resolution - a blurring of close-together components will mean we cannot accurately pinpoint the frequency of a given component. In summary, it will be possible to capture the two opposing elements of the bird call signal in a computer representation, but with some compromise. Slow changes will appear as time variations, while rapid changes will manifest as frequency effects.

Turning to the practicalities of signal processing on digital computer, the Discrete Fourier Transform is introduced as the digital counterpart of the bandpass filter - a basic component of a working frequency analysis system. Like the filter, it is an approximation to the ideal, and must be used in a way that trades off time and frequency resolution. The scheme consists of chopping the signal into short time segments, just as the filter bank chops up the signal in the frequency domain. The tradeoff is described by the relationship between the rectangular and sampling functions - whose essential feature is that of inverse width. The wider the rectangular function (which can be used to characterise the time resolution), the narrower the sampling function (which can characterise the frequency resolution). The use of digital rather than analogue frequency analysis allows far more flexible control of the procedures, useful in an experimental context. One example of this flexibility is the ability to apply a non-rectangular window function, such as the Hamming window, in order to alter the way in which frequency resolution is degraded in a practical system.

The Fast Fourier Transform is digital signal processing in its most practical incarnation - the special considerations leading to a fast, efficient form of the Digital Fourier Transform are detailed here so that the software written to implement the frequency analysis in the BCR system may be understood. That software, described in an appendix, forms the basis for the experimental environment. In addition to the pure signal processing function, it includes user interface aspects that allow the experimenter to visualise and manipulate the data in its various forms along the path to recognition.

7.1.3 Extracting key information from the call (4)

The recognition process relies on the use of essential features extracted from the bird call data - much of the data is spurious or redundant and must be discarded. The first stage of feature extraction is seen to be the identification of the peaks that must have been present in the original signal, but

which have been smothered to some degree in the spectral data available to the BCR system. Sources of distortion of the data include background noise or undesirable signals, but these are not directly catered-for in the BCR system - they are assumed to be reduced to insignificance.

The distorting factor which must be handled by the peak extraction process is the introduction of multiple subsidiary components into the spectrum, an inescapable consequence of the spectral analysis procedure. Each peak in the source signal results in multiple data points in the spectrum. The peak extraction process must deduce the positions of the peaks that result in the arrangement of points present in the data. The simplest way to do this is to detect the local maxima in the spectrum - these maxima give good approximations to the peak positions.

The problem of detecting local peaks is complicated by the likely presence of noise, introducing minor spurious maxima which interfere with the working of any simplistic local maximum extraction scheme. A solution, which applies specifically to the spectra produced in the BCR system, is to predict the effect of the spectrum analysis and attempt to reverse it. This allows the global maximum in the signal to be used, eliminating the problem with local maxima. Although other solutions, such as noise reduction by smoothing, are possible, the global maximum solution is simpler and efficient, and does not preclude the use of smoothing in addition.

The peak extraction scheme does have some limitations - it may report the presence of two peaks where there should be one. This may be caused by peaks which are closer together than the frequency resolution of the analysis, or because of errors in representing a continuous spectrum in a discrete domain. By considering the characteristics of bird calls and of the analysis system likely to be used, the probability of the former phenomenon occurring can be shown to be small. The latter phenomenon can be eliminated very simply, at the cost of increasing the frequency resolution (and thus processing time), or more complicated schemes can be considered.

Having discussed the form of a peak extraction algorithm, a practical evaluation of the algorithm proved useful. The extracted information was shown to embody much of the recognisable essence of the bird call. When signals were resynthesised from the extracted data, they proved to be very similar, and sometimes almost identical, to the original. Besides validating the choice of peaks as good features, and proving the algorithm to be effective, this analysis leads to the possibility of a practical spin-off not directly connected to recognition. Since the peak data occupy much less space than the original signal samples, they could be used to store bird calls for later playback in a device where storage space is limited. Such a device is planned as part of the parent research initiative of the present project.

A conservative estimate showed that to store the required number of call samples would take of the order of 20 Mb of memory, while storing peak data for the same calls would occupy at most about 4 Mb. In practice, these sizes could well be smaller, as indicated by the empirical evaluation that was performed. Assuming problems of decompression and reconstruction time could be solved, the peak extraction method has the potential to greatly improve the capability of a small solid-state playback device.

7.1.4 Classifying bird sound primitives (5)

The peak data is used to place short segments (frames) of the call into one of five classes. The classification is derived from consideration of the sound-production mechanisms in birds as well as observation of typical bird calls. The classes are: Noisy, Harmonic, Single, Multiple and Silent. The concept of division of calls into chains of elements, each element comprising a number of frames whose classes determine the overall class of the element, is developed.

The scheme for classification of frames uses the number of peaks and two measures, those of "singleness" and "harmonicness". A frame is single not only if it contains only one peak - the "singleness" measure takes into account the relative strength and distance from the main peak of subsidiary peaks. A similar consideration is used to broaden the definition of "harmonic". Both measures rely on simple weighted means and standard deviations of peak positions. If the standard deviation of the peak positions (or intervals between peaks) is below a certain threshold value, the frame is classified as single (or harmonic). This threshold is seen to be related to a frequency resolution value - since the singleness and harmonicness measures concern distance between peaks. The same threshold value is applicable to both measures, and is similar to, but independent of, the actual frequency resolution of the system. The reason for the latter is the possibility that factors such as weighting and error can influence the standard deviation calculation and result in smaller values than would be possible taking only resolution into account.

The reason for the use of these more sophisticated measures (rather than simply counting peaks and demanding strict harmonic relationships) is the resultant improvement in the flexibility and independence of the classification algorithm. Whether the BCR system really needs such advanced criteria was not verified.

The minimum number of peaks required per frame in order for the classification scheme to be meaningful was found to be five, and this is the number used throughout the BCR system.

Besides being assigned a class, each frame also has characteristic values associated with it. These values serve to further classify or differentiate the frame, so that it belongs to a general class, but is distinguishable from certain other frames in the same class. The characteristic values are the

average peak magnitude, and a characteristic frequency value. The latter is usually the weighted average frequency (peak position) in the frame, but in the case of the harmonic class, it is the average inter-peak distance, which should be close to the fundamental frequency of the harmonic oscillation. The characteristic values are useful at the element processing stage which follows frame classification.

The scheme is in general successful in classifying frames in the same way as they would informally be classified by a human observer. Frames which would not be readily classified by a human are treated in a perhaps less foreseeable way, but result in a classification which is nevertheless useful in the BCR system, since as the product of an automated scheme it must be consistent.

Once the frames are classified, the way is clear to an element-based, abstract description of the call. The final stage in the call's processing before recognition is attempted is the element discrimination and characterisation stage. Here frames are grouped together to form elements of different lengths, and the classes and characteristic parameters of the frames in each element are used to "name" or characterise that element.

The goal of being able to define the boundaries of an element such that each element contains only one class or sub-class of frame is seen to be unattainable. If we observe what we informally define to be an element, we find that it does not contain a single uniform type of frame, although one type may well dominate. Efforts to arrive at the element boundaries using a less strict definition of class content (i.e. allowing for an element to contain a mixture of frame types) were unsuccessful in providing a decisive algorithm. It was finally clear that the simplest scheme of all, using the magnitude of the frames with a silence threshold, would have to suffice. Although more sophisticated schemes are surely possible, it was worth attempting the simplest (and therefore speediest) one, in case it proved adequate.

Element discrimination thus relies entirely on the amplitude of the signal - wherever this falls below a threshold, an element is deemed to have ended, and a period of silence is recorded. Rather than define a separate threshold value for the discrimination process, the effect of thresholding already performed in the classification process is used. Thus each transition from a non-silent to a silent frame results in an element-silence boundary. In this way the class of each frame is the first parameter to be used in the element processing stage.

The other parameters of the classified frame, their characteristic frequency and magnitude, are used to further characterise the elements. Just as a listener might characterise a sound as "high" or "low", so is the average characteristic frequency of the frames in the element a used a descriptive parameter of the element. The "acoustic quality" of the sound is captured by the use of a class profile for each element - the relative occurrence of each of the four non-silent classes in the

element, weighted by the frames' characteristic magnitudes, is recorded. A simple dominant class value is also calculated - it has some use in the recognition stage and might conceivably be more useful should certain changes in the BCR system take place.

One aspect of element characterisation that should certainly have been implemented was the pitch contour - this is patently useful in recognition. However it would require rather more sophisticated processing to extract (and later to use in recognition) and, like many of the more complicated possibilities raised in the building of the BCR system prototype, it was thought worthwhile omitting it at this stage in case it proved unnecessary. Whether that was the case is discussed later in this chapter.

7.1.5 Syntax matching system (6)

The syntax matching stage is the most important part of the BCR system. Its success would vindicate the many compromises made in earlier stages, where complex processing was avoided with the aim of more quickly reaching the syntax stage. The syntax analysis, it was hoped, would work even on shoddily classified data, thus obviating the need to increase the complexity of the earlier classifying stages.

The syntax matching stage itself does not escape complexity. Its very mission, to behave well in the face of shortcomings in its input, means in part that it must be more sophisticated than it otherwise would have had to be. Some of its complexities are, however, simply consequences of the realities of practical bird call recognition.

To highlight the features and requirements of the syntax matching stage, it is contrasted with the simple, ideal case of a syntactic recognition system such as that employed in human language analysis. Such a system takes a well-defined string of unambiguous symbols as input, and outputs a definitive "yes" or "no" to the question, pertaining to a particular grammar (or set of rules): "Is this input string a legitimate product of the grammar?". In the bird call context, the question would be: "Can this bird call be produced by the rules associated with the bird in question?"

The strict syntactic recognition system can be implemented as a formal finite state machine or a pushdown automaton - straightforward algorithms which handle the well-defined symbols and states found in the language environment. The bird call environment has some similarities but certain differences which demand specialised behaviour from the algorithm to be used for bird call recognition. These differences and their consequences are summarised briefly here. The basic functioning of the syntax matching process is to accept an input string of elements (output of the element processing stage of the BCR system) and compare it with information that has been compiled about a particular species' calls, stored in the form of a "syntax spec".

The strings representing bird calls are made up of elements, which unlike words in a sentence, are chosen from an infinite set of possibilities. The syntax matching stage must therefore compare elements with internally stored prototypes, rather than instantly and absolutely knowing the value and meaning of each word. So the syntax matching process has a preliminary "recognition" stage, in which the elements comprising the string must first be recognised. The prototypes are stored in element specifications (in the syntax spec). The latter specify ranges into which the element parameters, namely frequency, duration and class profile, may fall.

Once the syntax matcher has identified an element in the string as matching one of the element specs, it refers to the syntax tree part of the syntax spec, to find out how (in what context) that element should appear in a bird call. The syntax tree consists of nodes which denote element specs, and from which branches lead to other nodes, thus specifying which elements may syntactically follow on from the node. This simple notation has the power to describe most or all of the forms that bird calls might take, although some additions are possible which would make the notation more efficient.

The syntax matcher must be able to begin a search at any point in the tree - both because the input string (bird call) doesn't start in a predictable way, or because the matcher will often have to restart the matching process (parsing) at points other than the beginning of the call. The consequence of this necessity (which does not arise in strict syntactic recognition systems) is that there are multiple paths in the tree that must be traversed when parsing. If only one path leads to a full parse, it is the correct one, and the string has been recognised.

However, in many cases (again unlike the strict syntactic systems) none of the paths allow the input to be fully parsed. The reason for this is the real-world imperfection of the string - elements may be obscured or corrupted or badly classified. The matcher is now faced with multiple sub-paths in the tree, and must choose one before it can continue parsing. It chooses the longest sub-path. Thus it tries to parse a string with the least number of failures along the way. The number of failures is noted, because the typical use of the BCR system will be to try parse a particular string with a number of different syntax specs, each spec corresponding to a different candidate species. The candidate that produces the lowest failure score is the BCR system's best guess as to the identity of the bird.

The method of choosing the longest sub-path could be extended to a more general one, in which all possible parse paths are tried, and only then choosing the parse which produces the least failures. That however represents a significant processing effort which is deemed unnecessary in this first prototype.

While the possibility of failure points in the parse is catered for, there is an attempt to minimise their occurrence, by making the element matching scheme more tolerant. Thus an element that does not exactly match an element spec does not immediately cause a failure. Instead, a partial match is allowed. Allowing for partial matches also allows multiple matches - the same element may match more than one of the element specs leading on from the current node. As with the failure handling mechanism, the imperfect match handling mechanism results in the possibility of more than one path at many of the parse steps. In this case, the explosion of paths is limited by assigning a score to a match, and chosing the element spec (and its path) with the highest score. As with the failure mechanism, the scores are noted, and accumulated after the parse to give a measure of how well the parse went. This "appropriateness" measure can then be used in conjunction with the failure score to judge the candidates.

Again, the partial match method could be made more general and thus its efficiency increased if necessary. For the prototype it's hoped that the simple match scoring scheme implemented will suffice.

The match score is also used to generate a preliminary measure of the fit between the syntax spec and the element. Without parsing the string, that is without regard to the syntax tree, the element specs are compared in turn to the elements in the string, and the best matching spec's score is noted. The scores for each element are thus added and the result divided by the string length in order to give an average match score for the string's elements. This preliminary score, along with a quick check that the string contains the minimum essential elements required by the syntax spec, allows specs that clearly do not match the call to be skipped, saving parse processing time.

The work of the BCR system implementation culminates at this stage with the recognition trials presented at the end of Chapter 6. Using a small set of calls chosen for their variety, a set of example syntax specs was compiled from the calls' element strings. The syntax specs were artificially generalised to some extent, to simulate variability in the input. Nevertheless the probability is that the specs would perform better than in real-world conditions is quite high, since they cannot fail to match the exact calls from which they are derived.

The trials gave a performance, which, were this a real-world situation, would be completely satisfactory. Using the combined scores, the BCR system located the correct syntax spec for every one of the twelve calls. The implications for a practical system are discussed in a later section of this chapter. As for the implications for the syntax matching stage, it can be seen that the system is able to perform a useful syntactic analysis of the call, even considering the fact that there is room for improvement in some of its mechanisms. One interesting result is the efficiency of the preliminary score - in the trials, it could be used to choose the correct syntax spec all on its own, without resorting to the failure score. On reflection it is clear that this is due largely to the artificial

conditions. In particular the trials do not adequately test the failure score's efficiency, since calls differing only in their syntax were not present in the test. In such cases, it is expected (since the failure score certainly gives correct results on its own terms) that the syntactic analysis capabilities of the system will become more prominent, and the value of the preliminary score will decline somewhat.

7.2 Successes and failures of the BCR system

The success or failure of the BCR system can be evaluated at three different levels: Its performance as a practical system ready for use in the field; its confirmation or rejection of the basic ideas and approach used for bird call recognition, and its capacity as a yardstick, proving ground and framework for future research. The following sections deal with these levels in turn.

7.2.1 Practicality of the BCR system

While a practical, working recognition system was not necessarily expected to result from this project, it was inevitably a working target, directing the research to a large degree. How far from that target is the current BCR system?

7.2.1.1 Particular requirements for practical operation

Looking first at some of the separable, local characteristics required of a practical system, the following areas can be discerned:

Speed. A reasonable demand for the performance of the device is that processing should take of the order of 1 minute from the time of call capture to that of species identification. The shorter the time, the more gratifying the device will be to use. The identification should take place while the bird is still under observation, so that confirmation of the identity is possible.

On the general purpose computer systems used to implement the BCR system, the vast majority (92%) of the processing time is taken up by the first stage, the spectral processing. On such general-purpose hardware, the total processing time for the call is too long to be practical. An IBM PC compatible machine with a 386SX processor running at 16 MHz without math co-processor (80387) requires about 40 minutes to process a 2-second call from sample sequence to element string.

The above does not mean that the BCR system can never be of practical use - it means that special purpose hardware and/or software will be required. A 486-based machine, for example, has specialised hardware in the form of the on-chip floating-point math co-processor. When normalised

to 16 MHz operation, such a machine reduces processing time to 5 minutes for the same 2-second call, and processors are available which run at 33Mhz or more so that 2.5 minutes or less would be required.

Even more specialised hardware, such as a digital signal processing chip, or an analogue filter bank, could be used as a plug-in replacement for the SPECTRA spectral analysis stage. Such hardware would be capable of producing the Spectral Frames in real time, i.e. with no appreciable delay. The speed of the BCR system thus becomes the responsibility of the pipeline processes following SPECTRA.

On the generalised hardware, the second stage, peak extraction, consumes approximately 7% of the total processing time, and the classification and element recognition stages make up the last 1%. On the 386 SX system this translates to about 3.5 minutes, and on the 486 system, less than 10 seconds. Thus with the right hardware, it is indeed possible to obtain practical operation.

We have seen what is required in order to optimize the SPECTRA stage. The only question remaining in the evaluation of the speed capabilities of the BCR system is whether the later stages could be improved upon in the software. It is contended that these stages give an indication of the upper end of the processing speed scale. The stages all do the minimum work required to produce a useful result. The peak picking stage looks for no more than 5 peaks (often it finds less) and requires no smoothing. The classification and element stages likewise produce a result quickly by not attending to finer details.

The time analysis above leaves out the syntax matching stage because it can't be included in the same frame of reference (essentially seconds/frame processed). However, once again we find that the syntax matching stage is pleasingly fast compared with the overall time - the average time for the calls used in the BCR system is approximately 1 second per candidate (for the 12 candidates used in the trials, the syntax matching part of the processing totalled 12 seconds). Compare this with approximately 40 minutes total processing time for each call. The syntax matching stage's speed is due partly to such devices as preliminary score checking, and partly to the inherent speed of symbolic, rather than numeric, processing.

It is important to note that in general the hardware in a hand-held, consumer device is less capable than that available for desk-top systems such as were used in this project. Less space for memory (which can often be traded off with speed), cheaper microprocessors and lower power supply capabilities mean slower speeds. Speed of the software is therefore very important. Because of its minimalist approach, if any system is to be fast enough to be practical, it is the BCR system.

Tolerance. In a practical situation disturbances and variability must be expected and catered for. The BCR system has many mechanisms for handling less than ideal conditions. The peak extraction algorithm is designed to work well in the face of low-level noise, by considering global rather than local maxima. However no measurements of noise levels (which are certainly noticeably present in the sample calls) have been done to quantify the system's performance.

The frame classification algorithm caters for continuous definitions of normally discrete classes - a frame may be "more harmonic than noisy", rather than "harmonic and not noisy". This means that classification is far less rigid and is tolerant to deviations from the ideal input.

The element discrimination algorithm uses only the most blatant of discriminatory features - elements are delineated by the amplitude of the signal, and any further "understanding" of the elements is left to the syntax processing stage, by passing on a low-level 'occurrence profile' of classes rather than simply a definite, discrete class for each element. Again, this makes for better tolerance, since a rigid classification would make syntax matching easier but be prone to more serious errors.

Finally the syntax matching stage goes to great pains to be tolerant of failures and to gather continuous, comparative information about the call rather than adhering to the strict discrete symbols of standard syntactic recognition. A failure during the matching process is counted and the matching resumes. Matches are assigned levels of appropriateness, so that even "bad" matches do not stop the process, rather they contribute correspondingly to a score.

Despite the very lax mechanisms all the way along the recognition pipeline, the net result is a system which largely succeeds in picking out the correct call from a group. Although under extensive testing it may be found that the BCR system is not as successful as under the limited trial conditions, it is clear that the tolerance built into the system has not overly compromised its recognition ability, and should prove a valuable asset in dealing with real-life conditions.

Usability and User Input. A working system must be usable by an ordinary human being. This is largely a matter of interface and is not part of the scope of this project. The BCR system has however purposefully used concepts which are human-oriented in terms of bird call representations and the features of bird calls. This was as much in order to help its designer, and the experimenter, as it was to aid the end-user, but the latter function should not be overlooked.

The various stages of processing closely match the ways which a person might think of a call, and as such they may be presented for user manipulation without the technical jargon and abstract thinking that might be required from a system far removed from the human domain. Spectrally-analysed calls map directly to sonagrams, considered acceptable for public consumption in field guides. Classes map almost always to human perceptions of sounds (whistling, noisy, etc).

Elements are natural and perceptible divisions in the call. Syntax is simply order and context, and since the rules in the syntax spec are formulated by a person in the first place, they are usually guaranteed to be expressible in plain language.

The importance of a user-comprehensible system cannot be overstressed. The bird-watcher remains the most capable recognition system available. The greater the likelihood that the user will be able to contribute to the recognition process, the more realistic is the goal of computer aided bird call recognition. Some speculative examples of the potential of user-guided recognition are as follows:

The user observes the captured call in the form of a sonagram on the device's display, and edits out parts of the call where he/she knows there was interference - a gust of wind or another bird's call. Only the 'clean' part is submitted for recognition.

The user observes that the call is rapidly changing, and alters a parameter to reprocess the call with a higher time resolution. Other parameters that might be altered are frequency resolution (for low-pitch calls) and noise thresholds (for differing prevailing signal-to-noise ratios).

Flexibility and Programmer Input. Flexibility of the system is strictly not directly tied to practicality. It refers to the ability to modify, program and tune the system. The net effect of a flexible system is, however, one of increased practicability, since it allows adaptation to real-life parameters after the strict "design" phase is over.

The BCR system is flexible in that it has many parameters which can be changed with minimum effort and maximum experimenter comprehensibility. The reference here is to the Bird Call Descriptor (BCD) mechanism of human-readable parameter files, and to the very fact that many aspects of algorithm behaviour are controlled through BCD parameters.

Another aspect of flexibility is the modularity of the system, allowing some parts to be changed without necessarily having to change the remaining parts. The BCR system is organised into distinct processing stages, which can be analysed and tested independently to a large degree. The output of each stage is always available and the system contains tools to display and analyse this output.

The most important area of flexibility and programmeability is that of the syntax specification. The syntax spec is designed to be formulated by a person - the expert who programs the system to recognize particular birds. The expert's experience and abstraction capabilities are sure to outperform any automatic learning algorithm. Since no such algorithm is in fact available, the human-training feature of the BCR system means the difference between a working system and no system at all. Because the system is programmeable by a person, it is likely that a working system could be produced, and that it would work well. Once such a system is in place, attention could turn to a learning algorithm that can derive the equivalent of human-produced syntax specs. Of course, the attraction of automatic learning is the reduction in tedium and time required to

produce a working system, and the increase in its comprehensiveness. The point here is that the BCR system is programmeable by either human or machine, and that this is the single most important contributing factor to the opinion that a working system is possible.

7.2.1.2 General assessment of practicability

We return now to the question at the start of this section, on the distance between the present BCR system and a working bird call-recognition device.

The BCR system is a working system. It can be used to attempt to recognize calls. Limited laboratory trials give a high recognition success rate. More realistic samples and conditions will likely reduce this success rate, but it is not expected to fail completely. The difference between the system trials and real-world conditions is one of degree - the real world will show a greater degree of variability. The BCR system is not a brittle one, so less than ideal conditions should result in graceful degradation of performance rather than abrupt breakdown. Thus with some confidence it is possible to say that, as far as recognition ability goes, the BCR system is at least minimally practical.

The questions of cost and speed, cannot be definitively answered here, except to say that powerful or specialised hardware would certainly be required to reduce recognition times from the order of tens of minutes to that of tens of seconds. The software itself is acceptably efficient, and would not need major attention in any time-reducing efforts.

The following sections mention additions, improvements and changes that might be considered in future systems. One of the most important is the addition of pitch contour handling, without which the system must inevitably perform below maximum potential. Others point to weaknesses in the classification and element discrimination stages. The various suggested improvements imply that there is some way to go before a truly high-performance recognition system is obtained. Nevertheless, the current system is workable, and could in fact be well fitted to act as a valuable aid, rather than a single-handed solution, to the problem of bird call recognition.

At the very least, the BCR system could be considered a preliminary recognition processor. The various improvements mentioned later could be incorporated in a second stage to provide more decisive information. Even more likely, the second stage could in fact revert to the user's recognition ability, through the use of recorded bird calls. In that scenario, the user allows the BCR system to narrow down the possible choice of species fitting a given call. Recall that the syntax matching stage is fast compared to the signal processing stage, so that the processing of a hundred or so candidates within a few minutes is quite feasible.

Once a short-list of birds is available, the user would listen to the recorded calls of each, probably recognizing the target bird with greater speed and accuracy in this way than any automated system. The BCR system as it stands has the power to attack the most prominent deficiency in current bird identification systems, the inability to carry out a primary search of a bird database, keyed on the call. Since the ability to store and play back bird calls efficiently is one demonstrably practicable spin-off of this project, it can be argued that in totality, the system is close to providing a workable result. Coupled with the projected short-listing capabilities of the general bird identification device (i.e. using other characteristics of the bird besides call in narrowing the search), the BCR system seems sure to be in a position to make a meaningful contribution.

7.2.2 Basic approach of the BCR system

Given that the BCR system as it stands cannot be termed a fully operational and practical system, it can be argued that the ideas and concepts used to build it are nevertheless valid, and simply in need of further refinement rather than complete revision or rejection. This section attempts to show that that is indeed the case. In the following treatment, the underlying concepts used in the BCR system are identified on a stage-by-stage basis and their validity discussed.

Spectral analysis stage: There can be little doubt that any bird call recognition system needs to analyse the calls in the same way as the hearing apparatus of humans or birds, or the instrument (spectrograph) commonly used for recording calls - on the basis of time-varying frequency content. It has been established that the recognizable information in the call is encoded in the frequencies, and it is clear that they change over time. The particular attributes of the analysis method, such as the frequency/time resolution trade-off, may be varied, but the basis remains valid. The value for this parameter used in the trials (5ms time frame) represents a reasonable compromise between the maximum time resolution of birds' hearing and the typical time frame lengths used in sonagraphic analysis.

Peak extraction stage: Given a spectrum-based analysis, the validity of extracting spectral peaks must be considered. As explained in Chapter 4, there are good reasons to conclude that the shape of many bird call spectra is such that they may be characterised by the positions and magnitudes of a small number of peaks. The peak extraction algorithm of the BCR system uses this assumption and its success (as evidenced by playback of calls reconstructed from the peaks) is an indication of its validity. The algorithm may be open to improvements (its shortcomings are outlined in Chapter 4), but the concept of characterising most bird call spectra by their peaks cannot be faulted. One type of spectrum, the 'noisy' or 'broad' type, could probably be better characterised by a stochastic model - resulting in parameters for the mean, distribution and so on. However the

recognition performance of the BCR system seems to indicate that the need for accurate characterisation of this type of spectrum is not necessarily great, and the existing peak-based system might be adequate.

Frame classification stage: The idea of classifying spectral frames stems from the simple observation that sounds made by birds have different tonal qualities. This observation is evident not only to the informed listener, but in most or all references to bird calls in which the recognizability of the bird's voice is in question. Take, for example, the "harsh, rasping swizzling" of the Masked Weaver or the "highly vocal, ringing" cry of the African Fish Eagle, as described in Roberts [2].

Notwithstanding the existence of distinct tonal quality differences, the classification of these differences is not a cut-and-dried matter. One issue is whether a recognition system requires a classification at all - could it not use the peak data directly in the matching process? The classes themselves must be decided upon and finally the means of classification must be determined.

The first issue, that of the necessity of classification, can only really be resolved by implementing two different systems, one that abstracts classes from the peak data and one that does not, and comparing the performance of the two. For the BCR system the former was chosen because it more explicitly matches what appears to happen when we ourselves recognize a call. The higher abstractions from the data make the system easier to understand for the experimenter and the user, and there are performance benefits implied by using the abstractions rather than the underlying data points in the later matching stage. The availability of the class information also gives the element discrimination stage, which follows classification, the opportunity to make more intelligent decisions, although that advantage was not put to use in the current system.

Assuming the need for classification, the question of what classes to use must be addressed. The BCR system settles on a small, almost self-evident set of widely divergent classes. Because of their distinct, intuitive nature, the validity of these classes is not really questionable. The small number of classes and broadly encompassing definitions mean that they are not as useful as they could be in characterising the call. This sacrifice is made in order to gain the above-mentioned security of validity, and because it greatly simplifies the implementation, and increases the performance, of the classification algorithm.

The method of classification used in the BCR system is probably by no means the most appropriate or accurate. It operates on individual spectral frames in the call, which itself is not necessarily ideal. The BCR system represents the first attempt at formalising and quantifying the concept of a class of bird call sound, and so the algorithm used is not indisputably valid. Suffice to say it has been based on reasonable assumptions about what makes a signal sound "harmonic" or "noisy" or "whistle-like", and for unambiguous examples of these sounds, performs correctly.

Thus we see that although there is theoretical justification for using classes, and a system which uses them appears to work as expected, the concept has not been unequivocally proven superior, and should therefore be given further attention in future development and experimentation. Note however that the net result of the classification stage is that the peak data is used in an intuitive and efficient way, whereas a means of using the peak data directly is far less evident. It would require template matching or other numerical classification methods which are potentially time-consuming and causing the remaining stages to move away from the human-like approach. This in turn would reduce the certainty of the validity of the entire system and require more empirical evaluation.

Element discrimination stage: There can be no doubt that a bird call must be treated as a sequence of elemental sounds. The definition of an element is less obvious, being used in the literature in an intuitively clear but informal way. A reasonable assumption originally made in the BCR system was that an element was a continuous sound of a uniform tonal quality. A break in the sound or a change in the tonal quality would signify the start of a new element. In practice it was found that the detection of tonal quality change based on the classified frames was not decisively achievable by simple means. It was decided to use only the breaks in continuity of the sound as the element discrimination criterion. The effects of tonal quality of the element were then left to the syntax matching algorithm to handle.

Using silence to separate elements is certainly valid and necessary, but it is a first order criterion. The concept of tonal quality discrimination is important and should be further investigated in a future recognition system.

Besides discrimination of element boundaries, this stage is responsible for characterisation of elements, that is, determining the way an element appears to the syntax matching stage. The principle here is to form a balance between including as much information as possible, to enhance the power of the matching process, and as little as possible, to improve the speed of the matching process. Whether the particular compromise used in the BCR system proves to be the best one cannot be determined, but we can consider the validity of principles used in characterisation

The parameters used to characterise an element in the BCR system are its overall class, its duration, its average frequency, and its magnitude-weighted class content profile.

The overall class is a reflection of the idea of an element being defined by its tonal quality - ideally most of the frames in an element are of a uniform class. In the BCR system, a percentage threshold is set so that if an element contains a high enough percentage of a frames of a particular class, that class becomes the overall element class parameter. Since the system does not use class in dis-

criminating elements, however, there is no guarantee that an element will contain a dominant proportion of frames of a particular class. The overall class parameter is therefore chiefly a mechanism placed in the BCR for future use.

Because the dominant class principle is not used, the characteriser includes the tonal quality information by recording a profile, or breakdown, of the percentages of each different class contained in the element. The principle of tonal quality being important in recognition is thus catered for by passing the appropriate information to the syntax stage. As mentioned, future systems might find different ways to use tonal quality information (whether by altering the frame classification stage itself or by changing the way the element characterisation uses the classified frames) but its use in some form is surely essential. The effectiveness of the preliminary score in the BCR system trials shows the usefulness of tonal quality in species identification, and also that the frame-classification/occurrence profile combination is an effective way of incorporating that tonal quality information.

The duration and frequency parameters are self-evident characteristics of elements, serving to differentiate between elements which are otherwise identical in spectral characteristics, as is often needed for elements found in a the call of a single bird or amongst birds of the same family.

One important characteristic that in principle should certainly be a part of a future recognition system, but which is not directly catered for in the BCR system, is the pitch contour, applicable most often to elements of type 'single'. The variation over time of the characteristic frequency (for single frames, the peak frequency) of the frames comprising the element is of great potential use in distinguishing many whistle-like sounds found in bird calls. The non-inclusion of this capability was due to time constraints, not because any difficulties were encountered in implementing it. Since no attempt was made to include the capability, it is however possible that some difficulty exists, and the promising results of the trials suggest that the capability is not essential. Nevertheless it is an obvious feature to include, and any additional aids to recognition ease the burden on other mechanisms and improve the robustness of the system.

Syntax matching stage: The use of syntax itself does not require much in the way of justification, much as the use of spectral information in the first stage did not. Clearly, a large proportion of birds either use the order of utterances to encode a species-specific message, or serendipitously have a stereotyped syntax to their calls. A few sophisticated singers have a complex syntax or a wide variability making the syntactic analysis less useful, but by and large the rest have simpler calls, and these simple syntaxes would be handled in any future system by a process something like the BCR system's syntax matching stage.

Perhaps the less obviously valid assumption made in this project is the one that transfers the *bulk* of responsibility for recognition onto the syntax matching stage. It is this reliance on the syntax of the call that has led to simplistic treatment of such areas as frame classification and element discrimination, and has thus influenced the entire BCR system. Rather than achieve the best conceivable capabilities in the earlier stages, the idea has been to get the call into a form appropriate for syntactic analysis, where, the assumption implied, the earlier fuzziness would be dismissed as the syntax of the call shone through to unequivocally identify the species.

Working under the above hypothesis was a worthwhile endeavour for a number of reasons, which have been mentioned in other places in this document. Relying on earlier stages means more complicated processing at these stages, and requires larger samples of data for teaching the system. The reliance on the syntactic stage greatly reduces the resources required to bring the project to fruition. Naturally, if it turns out that the assumption was false, the extra effort will have to be made, but it makes sense to attempt the shortest path to completion before tackling the longer ones.

Unfortunately this project cannot be said to conclusively prove the validity or lack thereof of the syntax-domination hypothesis. The trials show that a bird call can be recognised without more complicated early stages, and therefore that the syntax matching is a meaningful part of the system. But the performance in the trials was good even without considering the syntax scores, because the correspondence between element specifications and actual input elements was artificially high. It remains to be seen whether the syntax stage has the power to 'rescue' the system performance in the face of more realistic conditions. In other words, it remains to be seen whether, when elements in the bird call begin to match the specs less well, the syntax matching performance deteriorates in step or at a slower rate.

However, the BCR system certainly provides the means to conclude on this issue. The reason this project does not conclude on it is because the appropriate experiments (see section on future work) have not yet been performed. No major additions or changes to the BCR system need to be done in order to perform these experiments - it is simply a question of evaluating its performance using an appropriate sample of bird calls. As such, then, the BCR system and its syntax matching stage constitute a certain amount of unfulfilled potential.

Finally, another principle embraced in the syntax matching stage is that of fault tolerant processing. The syntactic analysis stage differs from a standard syntactic recognition system not least in that it does not endeavour to produce a discrete decision. Rather its output is a graded, relative appraisal of each call it has analysed, and the winner is the best performing call, rather than letting "winner take all" and eliminating all evidence of other calls that performed well. This fault tolerance can certainly be claimed to be a desirable feature of any future recognition system, and is particularly appropriate in the context of the hand-held bird identifier project. The latter is based on providing

lists of birds as possible candidates in response to user input of identification parameters. The BCR system can be made to provide such lists, which can in addition be sorted according to the likelihood that a particular species is the right answer.

To conclude this section, there are many good reasons for the BCR system being constructed the way it is, and although conclusive experiments to prove the concepts have not in general been performed, it seems reasonable to suppose that any future system will look something like the BCR system. Certainly indications are that the system is currently on the right track. Whatever changes are required in the future will likely be local and limited, and often implied by the discussions here, rather than a global revision of the operating principles.

7.2.3 The BCR system as a frame of reference

Should the BCR system ultimately fail to lead to a practical system, and should the concepts upon which it is based be found to be inadequate, it will nevertheless have made a meaningful contribution to any further research on the subject. Even as a total failure, it will have shown which approaches are fruitless, and why. It will have begun the process of filling the vacuum of material that exists on this subject. Later attempts are bound to have greater probability of success, with the hindsight offered by the BCR system.

In fact, as the system cannot decisively be said to have failed at all, it has a greater contribution to make than it would as a "map of where not to go". As detailed in the previous sections, the system has not yet been used to fully test some of the hypotheses upon which it is built, and certainly some aspects of the system are less than ideally efficient, but in general it has much to say about the direction which future work should take.

In any future endeavour, the hard empirical performance of the existing system will be available for comparison. Any idea which does not improve on BCR performance can be rejected immediately. Thus besides being an abstract source of ideas from which to derive new ones, it can be used as practical measuring tool, to quantify behaviour in ways which were not possible before it came into existence. The following outlines some of the particular areas and ways in which the BCR system can be used as a frame of reference.

Recognition performance: Clearly any new system can be compared for recognition ability with the BCR system. If the new system works with similar scoring mechanisms, the scores can be compared, otherwise the final implication of the scores, ultimately which bird is chosen, can be statistically evaluated over a selection of different input calls.

Improvements to the system can be made incrementally, with any new addition to any of the various stages being evaluated in terms of concrete results produced. Such improvements could include the tuning of parameters to various processes, the speeding up or increased sophistication of algorithms, the addition of new sub-systems (such as pitch contour handling), or the complete revision of an approach, for example in the way an element string is syntactically evaluated.

A bird call language: The BCR system defines a number of terms and concepts in the domain of bird call analysis. This allows issues and problems (and possibly their solutions) to be articulated in new ways, perhaps to be expressed where before they were not apparent at all. Such ideas as the number of peaks distinguishable at any time in the frequency content of the call, the class (with well defined ways of placing a sound in a particular class) of a bird call sound, the possible ways of defining and characterising an element, can all be discussed quantitatively and expanded on with specific reference to a particular existing system, namely the BCR system.

Of particular significance is the establishment of a means of specifying the syntax of a call in formal manner. Even if this "language" is not capable of covering the entire spectrum of bird utterances, it is a strong starting point from which to embark on a path of improvement. The language may be used as a new tool for testing hypotheses about the syntax of particular birds' calls, thus allowing new basic research in a little-covered area.

Estimating practical attributes of a device: The BCR system allows firm estimates of the practical requirements for an evisaged recognition or playback device. Memory requirements for processed calls meant for playback, or bird call database size for recognition, are easily and confidently estimable from the current system. Its speed on the generalised computer system where it currently resides can be mapped to apply to other hardware, and its activity can be analysed to show where speed improvements might be possible. As mentioned earlier in this chapter, the BCR system represents something of an upper limit for the speed of this kind of system, hardware improvements notwithstanding. Size and weight are derived from the hardware required to support the desired memory capacity and speed. Cost is derived from these too, as well as the labour required to teach a call to the system, which the BCR system allows one to measure.

In all, the goal of providing an infrastructure for, and first approximation to, a practical bird call recognition system can be said to have been achieved.

7.3 Future

In the preceding sections many references have been made to work to be done in the future, using the BCR system, or on the system itself. Here these projected experiments and modifications are briefly expanded upon, and additional ideas for future research presented.

7.3.1 Concrete suggestions for additions to the system

Pitch Contours: One of the additions to the BCR system most obviously likely to be useful is the ability to handle pitch contours. This possibility was discussed in some detail in Chapter 5. There, it was suggested that the pitch contour facility might turn out to be unnecessary, so that before implementing it, the system should be taken to the stage where calls could be recognised. Having done so, we are now in a position to return to the question of whether to add the facility or not.

In order to answer the question, tests need to be performed on those calls which appear to code information in the pitch variation inside the elements. Calls such as those of many Cuckoos, Tits, Thrushes, Chats and many more. The test would show whether the BCR system handles these calls well - differentiating between similar calls as well as between calls likely to be heard together in the field. If the system is not successful (perhaps even if it is, if truly conclusive evidence is desired), the pitch contour capability should be added, and then tests with and without the facility will prove whether it is advantageous. Please refer to Chapter 5 for ideas on implementation.

User interface: The software for the BCR system is written mostly as small building-block modules with little or no concern for user interface issues. A simple and limited interface program was implemented in the DRAWER software, allowing display and some manipulation of data. A useful addition to the system would be a full-featured graphical interface for managing all aspects of experiments, from set-up, to intermediate display of data, to collation of final results. The current BCR modules could easily be integrated without modification, or with superficial modifications to the various procedures' interface handling (input formats might need to be generalised and text output of prompts and progress indications would no longer be done inside the process module).

The envisaged system would display, and allow the user to manipulate, the information in the Bird Call Descriptor - whose hierarchical arrangement is better suited to a partially graphical form than the current pure textual representation. The user would choose an input call (sample file) and specify stages through which it should be processed, parameters to the process and output file names, and finally run the pipeline all without leaving the environment. The output of any stage could be displayed and compared with other output displays. Previous runs of experiments could be selected and reviewed, and single stages rerun with new parameters. The underlying data formats and means of recording results would largely not need to be changed.

The streamlining of the user actions required to perform experiments would improve the productivity and usefulness of the system. One area where such aids to user input are essential is the formulating of the syntax spec. The current system requires that the operator obtain a text/numeric form of the element string data, and by scanning through this data, and comparing it with graphical

plots, generalise it and formulate the element specs. The process is extremely tedious and once realistic samples involving larger numbers of similar input strings must be analysed for each bird, will tend to be unmanageable.

Instead, what is needed is the ability to graphically manipulate multiple calls, selecting elements in sonagraphic plots and requesting that the computer call up the appropriate data from the corresponding element string. Multiple examples of an element designated as identical by the operator should have their characteristics statistically merged in an automated way, and the results output as an element spec in the normal format. The operator's task would then be to simply designate sets of similar elements, decide on the relative importance of each element and finally assign a symbolic name to the resultant element spec. Once such a spec exists, the program should also have the ability to decide whether a new element belongs to a given spec - in other words, it should have access to the element matching sub-system of the syntax matching stage.

The formulation of the syntax tree part of the spec is less tedious, as it involves simple symbolic manipulation. However the text form of the syntax tree is not as easy to understand as a graphical form, and an interface program should allow graphical creation and modification of the tree. If it is well integrated into the system, it could also provide the capability of playing back calls constructed along a user specified path of the tree - thus allowing the operator to check that it "sounds right".

These aids to specification are essential if the 400 or more birds envisaged for such a system are to be efficiently incorporated. However they require a greater amount of effort than the simple project management interface first suggested. Object orientation and a windowing graphical environment (as well as ability to handle sound input and output) would be well advised, and some BCR modules might need to be converted to one or more objects.

Syntax spec additions: The syntax tree spec could benefit from an expanded definition as outlined in Chapter 6. Labour-saving additions such as loop counters, and relocateable sub-trees would allow the tree to become more understandable, which would reduce the likelihood of human errors in its definition. In order to take advantage of the new facilities, the SYNTAX software module would have to be revised at a fairly low level.

7.3.2 Experiments to be performed using the system

This project has taken the BCR system to a stage where it operates, and shows much potential. However extensive testing remains to be done, to show whether improvements are needed and what those improvements should be. In the process of searching for weaknesses in the system, it is possible that strengths too will be found, and that new conclusions about the recognizable

characteristics of bird calls will be drawn. Thus these tests have both a research and a validation or diagnostic component. The following are some tests that are clearly necessary before any deep understanding of the current system's capabilities is gained:

Exercise the syntax sub-system: The burning question of whether the syntax of a bird call contains enough information to recognize it on the basis of very sketchy spectral analysis must be addressed. The trials of Chapter 6 used calls which were so different that they did not prove whether it was the tonal quality or the syntactic structure that was making them recognisable.

If it is found that the syntax is sufficient even when the sounds are similar, or not easily distinguishable by the part of the system that handles tonal quality, the current system will be a long way toward practical viability, and an interesting characteristic of species-identification calls will have been uncovered. If not, the frame classification and element discrimination stages especially will have to be re-examined and made more powerful. Changes to these stages will have implications for the syntax matching stage and the syntax spec.

The test should consist of running a set of calls with similar sounding notes through the BCR system in the same manner as the trials of Chapter 6; that is, each call should be compared with the syntax specs of all the calls. The set of calls must be chosen so that the preliminary scores are inconclusive, unlike the afore-mentioned trials. If the system then correctly identifies the birds based on the failure scores, the syntax matching stage will have been shown to be indispensable. If not, it will show that without accurate tonal differentiation, the syntax sub-system is crippled in its recognition power.

A second test should consist of calls which, unlike the above tests, are chosen because of the high probability that they would occur together in the field, and belong to birds which might conceivably be mistaken for each other. Once again, an attempt should be made to find a group of such calls which cannot be easily differentiated on tonal quality alone. Such a test would be one of the most stringent possible, and if the BCR system is able to perform acceptably here, the chance that it will be a useful tool in the field is very good.

A final test could use calls that have elements which are difficult to classify and characterise - the sounds might be easy to tell apart for a human, but pose a problem for the BCR system. Such sounds are often those with many spectral components and/or rapid time variations, such as the croak of a Pied Crow. Such calls tend to be "recognised" by many different specs, because the many elements (due to break-up of long elements caused by the rapid and noisy nature of the calls) and the broad definitions in the specs mean the calls easily find a path through the spec. In the trials of Chapter 6, where the Pied Crow (CROW), Fiscal Shrike (FISC) and Jacobin Cuckoo (JACO) are only marginally distinguished from the others by the failure scores (indicating differentiation on the basis of syntax), the preliminary score (differentiation based on spectral char-

acteristics) saves the day. The present test could try to find a set of plausibly co-occurring calls that are problematic in the same way as those mentioned here, and yet sound so similar as to be indistinguishable by the preliminary score mechanism. The hope is that such a set will not be found, indicating that while the weakness may exist in theory in the BCR system, it is not likely to be activated in the field.

Test the limits of the system: The BCR system includes many parameters to the process that can be varied at the will of the experimenter. Some of the parameters have implications for the efficiency of operation of the system, others simply for effectiveness of recognition. Parameters in the first category naturally tend to have good and bad value ranges - those that predictably improve or degrade the system performance, while their effect on recognition capabilities is less predictable. An important parameter of this type is the FFT length, or frequency resolution (coupled also to the time resolution parameters) in the spectral analysis stage. A lower frequency resolution means a shorter and therefore faster FFT (if the increased time resolution is discarded by downsampling) or a cheaper specialised FFT hardware. A series of tests in which the frequency resolution is varied could yield results with useful practical implications, if it is found that recognition ability is not badly affected.

Because of the domination of the processing time by the spectral analysis stage, it is not likely that many other parameters will have as strong effects on performance as the FFT length. But others which have implications for system resources are possibly the number of peaks per frame in the peak extraction stage (not a BCD parameter, but set in the software source code), and thresholds such the SNR in the peak picking process which affect how many peaks have to be considered and therefore the speed of the process.

In the second category of parameters are those such as the distance and magnitude threshold in the frame classification stage, which need to be tuned by repeated testing on typical sets of calls until the optimum values are found.

Use multiple samples: Tests need to be performed on multiple examples of calls from the same species. A syntax spec may be formulated, as in the present project, based on single example of the call and some abstract knowledge, but individual and geographical variability is likely to occur. Confidence in the ability of a spec to recognize any bird of the species can only be gained by empirical confirmation. If the intra-specific variation is found to require the element specs to be very broad, the syntax sub-system might well prove to be of prime importance.

7.3.3 Long term ideas

This section ends with a look further ahead than the well defined work that can be carried out on the BCR system in the immediate future. Some relatively far reaching changes can be speculated upon, ideas which arise out of the experience with the system that has so far been gained.

Learning facility: The ability to teach the BCR system manually (by adhering to a human-like framework for recognition) has been advanced as an important positive feature of the system. That is because the effort required to produce a system that does not require manual intervention is guaranteed to be great, and would have held up the completion of the recognition system unacceptably.

Ideally, however, an automatic or semi-automatic (supervised) learning facility would be desirable, primarily to ease the burden of creating the syntax specs. A successful learning facility would also be modifiable for inclusion in the user system (the pre-taught system) providing the feature of adaptability. The user might then feed back positive identification information, slowly increasing the system's knowledge with use.

It is possible that producing an automatic learning system and feeding it sufficient data would require as much work as assembling all the required syntax specs by hand, and that the quality of the specs would not necessarily be equal to those produced by people. But the possibility of constantly improving and refining the syntax spec database is attractive and might make the effort worthwhile.

How would such a learning system work? Simply put, it would extract probabilities from the input data. Calls would be processed in the normal way, through to the element string stage. The strings would be analysed and elements would be grouped according to closeness or similarity, with thresholds for these attributes being set from outside the system. The grouping of elements would be a process similar to that of element matching in the BCR system's syntax matching stage. The difference would be that no spec exists - it would constantly evolve. Rather than using a threshold set arbitrarily by the supervisor, a more reliable (but less labour-saving) means of specifying whether an element belongs to an established group or a new one would be for the supervisor to give examples of extremes for each group, or to be asked whether a given element belongs to a given group.

Once the elements in a large sample of calls from a particular bird have been grouped, the system would formulate an element spec for each group, and move on to the task of deriving a syntax tree specification.

The most straightforward way for the syntax tree to be determined is for the learning system to gather statistics on which elements tend to follow each other. Ideally, all possible combinations of the given element specs would be found, and those combinations which actually occur then incorporated into a tree specification. However such combinations could be very great in number, especially for long calls. More likely, an analysis would be carried out for shorter subsequences, and the resulting "grammatical" sequences then used in combination to find the overall tree. The algorithm would require many sophistications, such as detecting when a recurring sequence is actually best described by a loop, and there are many pitfalls possible due to input data being less than perfectly well formed.

A more realistic option than a fully automated tree learning system would be an automated tree analysis system. The operator would devise a syntax tree as before, and then allow the system to evaluate it against a large sample of the calls it is supposed to recognize. By accumulating statistics on the number of traversals of each branch, redundant or incorrect branches could be pruned from the tree. It also is possible that the statistics would bring to light segments of strings that were not being properly parsed, so that the operator would know that additions to the syntax tree were required.

The natural result of such an evaluation would be a probability associated with each branch in the tree that would be of use in the recognition process. Rather than counting only failures, and making branch decisions (in the case of multiple alternatives) based on looks ahead or match scores as in the current syntax matching process, these scores and decisions could be based on or include the probabilities. Thus the output of the recognition system would be a true probability that the given call matches the given spec, as opposed to the arbitrary score scales used currently.

In combination with the relatively simpler element spec learning system, the above scheme could provide a simple but useful supervised syntax spec production system. Note however that statistically meaningful numbers of example calls would be required for each bird.

Generalized syntax specification: The syntax spec as defined in this project is designed to be used in a specialised environment - namely the syntax matching process. In Chapter 6 it was noted that various additions to the definition would make the syntax spec similar to a programming language in its flow control capability. The execution of "commands" however was still limited to the simple match operation performed at each step by the syntax matching process.

It is possible to conceive of a more general syntax spec definition, which allows greater choice of what is done at each step. The concept would allow different syntax specs to go about the business of determining whether the input call is "theirs" in different ways. A threshold used in determining

the class content of elements, for instance, might be changed between one spec and the next. The system would then be specifying the following procedure: "To recognize call X, process with threshold A, and to recognize call Y, process with threshold B".

The above scenario requires the entire call to be reprocessed for each candidate, but it could equally break processing up into units of elements. Each segment of the call could be processed only when required by the syntax spec, and in accordance with what it expects to find. The influence of the syntax spec on the processing need not only be at the relatively late level of element discrimination - it could extend all the way to the start of the pipeline, to the spectral analysis stage or even earlier.

In the extreme case, then, the syntax spec could, given a silence-separated string of bird call sample segments, process each segment differently and each different syntax spec could reprocess the call according to its own needs. It would however, be necessary to perform some preliminary uniform processing in the way done by the BCR system, so that decisions on the special needs of each element or each spec will have a basis. For practical reasons, the processing should be such that it does not all have to be redone for each call, since that would make it extremely time-consuming. The later stages, however, as has been shown, are very fast and could well be rerun multiple times.

To clarify the concept, here is an example of the new generalized spec or "recognition program". Assume the call has been passed once through the current BCR system pipeline, resulting in a standard element string. The information in the string is then available to the syntax spec, which reprocesses elements when necessary:

```
BIRD: Rufflethroated Garbler

ELEMENTS:

Garble-1: Essential
 GARBLEPROFILE = (80, 10, 5, 5)

  Peakpick(SNR=30)                ;Use SNR of 30 to find peaks from
                                  ;  Spectral Frames
  Class (Mag_thresh=Call.AvMag    ;In classification stage use magnitude
                                  ;   threshold of a quarter of the call
                                  ;   average magnitude, and a distance
                                  ;   threshold of 0.3 kHz
  Element()                       ;Do a default element discrimination

  Return ((Element.Class=SINGLE) AND (Element.Duration < 100)  AND
          (Element.Freq <= 2.0) OR Match(Element.Profile,GARBLEPROFILE))
End Garble-1
```

```
Bargle-3:
  a = 0;
  For i = 1 to Element.Length do
     if Class.Freqs[i] > Element.Freq then a = a+Class.Freqs[i]
  End For

 Return ((a < 100) AND (a > 73))
End Bargle-3


SYNTAX:

  Node_1: Garble-1
    Node_2
    Node_3

  Node_2: Bargle-3
    Node_1
    Node_3

  Node_3:
    Node_1(Element[Current] = Bargle-1)
    Node_4(
           (Element[Current] = Bargle-1) AND
           (Current > 5)
          )
  Node_4: ENDPARSE
```

The above example is not meant to be a consistent, workable solution but to illustrate some envisaged features of a generalized spec. The element specs could be procedures for identifying an element, having the ability to reprocess a given element from any stage, and with access to the output of each stage so that custom identification indicators can be derived. Thus when it is required to know whether an element matches the Gargle-1 spec, the segment of call corresponding to the element is reprocessed from the peak extraction stage with new parameters. The results of the preliminary processing are also available, so the spec is able to specify a magnitude threshold based on the previously found Call.AvMag (average magnitude for that call).

The element spec returns (in this example) a Boolean value which says whether the input element matches the spec. It calculates this value from the results of the element discrimination process, which are assumed to be available as Element.Class, Element.Duration etc. The second spec, Bargle-3, illustrates the use of a completely customised matching procedure - it sums the char-

acteristic frequency values of the frames produced by the preliminary run of the frame classification process, but only if they are above a certain level. Then it makes the result of the match dependant on this sum falling in a certain range.

In this fashion, elements can be identified in ways that are highly efficient for one type, but not necessarily for all types. The intention is that the syntax spec would have a set of specialised language commands that allowed intervention and customisation of any stage of the pipeline.

The syntax tree part is also shown with some possible extensions - the first two nodes are of the standard format, implying a simple match with a given element spec. The third node specifies one branch for a given match, the other branch requiring both a match and that the current element be more than 5 positions from the start of the string. In general, some of the actions available in an element spec section might be transferred to the syntax tree section or the two sections merged into a more conventional programming-type language.

The generalized syntax spec lends itself to a processing format which would appear to have merit when one considers human's recognition processes and the limitations of the current system: Multiple pass recognition. Rough, quick processing of the call gives the system an idea of what the signal is, eliminating obviously inappropriate candidates and setting the scene for intelligent reprocessing. The system then effectively takes a closer look, applying specialised recognition procedures to particular situations. This is potentially a powerful concept, although it has a number of practical problems. First, a way must be found to reduce the need for constant reprocessing of segments - perhaps by caching processed segments and allowing requests for reprocessing to be filtered by a system which finds a previous result of a similar request. Secondly, more research is required in the formulation of each spec, since many more possibilities exist and are different for each call. The use of a generalized spec would therefore be a major step onwards from the current system.

Parallel processing possibilities: In the event that later pipeline stages become more complex and require more processing resources, multiple processors could be used in parallel to allow real-time performance. The BCR system is easily converted to a parallel environment, because processing is on a frame-by-frame basis and the entire input sequence is not required before a given pipeline process can operate. Not only may all the processes in the pipeline run concurrently, but the call may be split into segments containing multiple frames, and each segment can form a separate, independent pipeline. For example, on system with four processors, one might divide a call into three parts and extract spectral frames concurrently with the FFT stage on three processors, interleaving the results and performing the rest of the pipeline actions in sequence on the fourth.

7.3.4 Concluding remarks

My hope is that this document has illuminated the area of bird call recognition, shown how bird calls may be quantified and defined the shape of a future practical recognition system. Computerised support for the casual and serious bird-watcher is an idea whose time has come, especially where the calls are concerned, and the work described here will hopefully have made a concrete contribution to the realisation of that idea.

Bibliography

[1] Dooling, R.J. Auditory perception in birds, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch. 4.

[2] *Roberts' Birds of Southern Africa*, Trustees of the John Voelcker Fund, 5th ed. 1985.

[3] Becker, P.H. The coding of species-specific characteristics in bird sounds, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch. 7.

[4] Horowitz, S.L. Peak recognition in waveforms, In: Fu, K.S. ed. *Syntactic Pattern Recognition, Applications*, Springer Verlag, 1977, ch. 2.

[5] Moayer, B. And Fu, K.S. Fingerprint classification, In: Fu, K.S. ed. *Syntactic Pattern Recognition, Applications*, Springer Verlag, 1977, ch. 8, pp. 186-7 and 192-3.

[6] Fu, K.S. ed. *Syntactic Pattern Recognition, Applications*, Springer Verlag, 1977.

[7] Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1.

[8] Thorpe, W.H. *Bird Song*, Cambridge University Press, 1961.

[9] Hinde, R.A. ed. *Bird Vocalisations*, Cambridge University Press, 1969.

[10] Jellis, R. *Bird Sounds and Their Meaning*, BBC, 1977.

[11] Catchpole, C.K. *Vocal Communication in Birds, Studies in Biology No. 115*, Edward Arnold, 1979

[12] Brackenbury, J.H. The structural basis of voice production and its relationship to sound characteristics, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch. 2.

[13] Morton, E.S. Grading, discreteness, redundancy, and motivation-structural rules, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch. 6.

[14] Wiley, R.H. And Richards D.G. Adaptions for acoustic communications in birds: sound transmission and signal detection, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch.5.

[15] Thielke, G. Geographic variation in bird vocalizations, In: Hinde, R.A. ed. *Bird Vocalisations*, Cambridge University Press, 1969, ch. 14.

[16] Uhr, L. *Pattern Recognition, Learning and Thought*, Prentice Hall, 1973.

[17] Gonzalez, R.C. and Thomason, M.G. *Syntactic Pattern Recognition, An Introduction*, Addison Wesley, 1978, pp.1-176

[18] Fu, K.S. *Syntactic Methods in Pattern Recognition, Mathematics in Science and Engineering*, Academic Press, 1974, appendix F, pp. 262-8.

[19] DeMori, R. A descriptive technique for automatic speech recognition, *IEEE Transactions on Audio and Electroacoustics*, AU-21, 1972, pp. 89-100.

[20] Oppenheim, A.V. Speech spectrograms using the fast Fourier transform, *IEEE spectrum*, August 1970.

[21] Flanagan, J.L. Voices of men and machines, In: Schafer, R.W. And Markel, J.D. ed. *Speech Analysis*, IEEE Press, 1979, pp. 4-15.

[22] Ladefoged, P. *A Course in Phonetics*, 2nd ed. Harcourt Brace Jovanovich, 1982.

[23] Rabiner L.R. And Schafer, R.W. *Digital Processing of Speech Signals*, Prentice-Hall, 1978.

[24] Schroeder, M.R. Vocoders: analysis and synthesis of speech, In: Schafer, R.W. And Markel, J.D. ed. *Speech Analysis*, IEEE Press, 1979, pp. 352-65.

[25] Fant, G. The acoustics of speech, In: Schafer, R.W. And Markel, J.D. ed. *Speech Analysis*, IEEE Press, 1979, pp. 17.

[26] Holmes, J.N. A survey of methods for digitally encoding speech signals, *Radio and Electronic Engineer*, vol. 52, no. 6, June 1982, pp. 267-76.

[27] Atal, B.S. And Hanauer, S.L. Speech analysis and synthesis by linear prediction of the speech wave, *Journal of Acoustical Society of America*, vol. 50, no. 2, part 2, 1971.

[28] Schafer, R.W. And Rabiner L.R. Parametric representations of speech, In: Reddy D.R. ed. *Speech Recognition*, Academic Press, 1975, pp.136-147.

[29] Hess W. *Pitch Determination of Speech Signals: Algorithms and Devices*, Springer Verlag, 1983.

[30] White, G.M. Speech recognition: an idea whose time is coming, *BYTE*, vol. 9, part 1, Jan 1984.

[31] DeMori, R. Syntactic recognition of speech patterns, In: Fu, K.S. ed. *Syntactic Pattern Recognition, Applications*, Springer Verlag, 1977, ch. 4.

[32] Makhoul, J. Linear prediction in automatic speech recognition, In: Reddy, D.R. ed. *Speech recognition*, Academic Press, 1975, pp. 183-217.

[33] Denes, P.B. Automatic speech recognition: old and new ideas, In: Reddy, D.R. ed. *Speech Recognition*, Academic Press, 1975, pp. 73-91.

[34] Bridle, J.S. Advanced signal processing aspects of automatic speech recognition, In: Greasey, D.J. ed. *Advanced Signal Processing, IEE Telecommunications Series 13*, Peter Peregrinus, 1985, pp. 275-9.

[35] Hyde S.R. Automatic speech recognition: A critical survey and discussion of the literature, In: Dixon, N.R. And Martin, T.B. ed. *Automatic Speech and Speaker Recognition*, IEEE Press, 1979, pp. 16-49.

[36] Wickstrom, C. Factors to consider in recording avian sounds, In: Kroodsma, D.E. And Miller, E.H. ed. *Acoustic Communication in Birds*, Academic Press, 1982, vol. 1, ch. 1.

[37] Oppenheim, A.V. And Schafer, R.W. *Digital Signal Processing*, Prentice-Hall, 1975.

[38] Stremler, F.G. *Introduction to Communication Systems*, Addison-Wesley, 1982.

[39] Gillard L. *Southern African Bird Calls*, cassette tape, Gillard Bird Cassettes, 1983, parts 1, 2 and 3.

An experimental system for computer aided bird call recognition

VOLUME 1 Introduction to Conclusion

Illan Samson Colombick

A dissertation submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1992