

**CLASSIFICATION OF WOLF CALL TYPES USING REMOTE SENSOR  
TECHNOLOGY**

---

A Thesis  
Presented to the  
Faculty of  
San Diego State University

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science  
in  
Computer Science

---

by  
Deborah Curless  
Spring 2007

**SAN DIEGO STATE UNIVERSITY**

The Undersigned Faculty Committee Approves the

Thesis of Deborah Curless:

Classification of Wolf Call Types Using Remote Sensor Technology

---

Marie A. Roch, Chair  
Department of Computer Science

---

Joseph Lewis  
Department of Computer Science

---

Roxana N. Smarandache  
Department of Mathematics and Statistics

---

Approval Date

Copyright © 2007

by

Deborah Curless

All Rights Reserved

## **DEDICATION**

To my wonderful parents, for being such good examples.

## **ABSTRACT OF THE THESIS**

Classification of Wolf Call Types Using Remote Sensor  
Technology

by

Deborah Curless

Master of Science in Computer Science  
San Diego State University, 2007

There is an increasing amount of research with the goal of understanding wildlife found in our environment. Observing the behavior of a species, including vocalizations, is fundamental to this goal. Researchers in the biological sciences have traditionally had to gather their observations manually, with a great deal of labor-intensive tasks. It would be beneficial to design and build a system that automatically gathers and analyzes this behavioral data.

This thesis presents such a system. Our system starts with a remote sensor that uses digital signal processing to automate data acquisition. The system then sends the acquired data to a remote lab via a high-speed wireless network for processing. Once the data is in the lab, our system classifies the data using hidden Markov models. The goals of this research are to build this system with the best possible level of performance, and to answer whether a pattern recognition system based on hidden Markov models can classify wolf call types with a reasonable level of success.

## TABLE OF CONTENTS

	PAGE
ABSTRACT .....	v
LIST OF TABLES .....	viii
LIST OF FIGURES .....	ix
ACKNOWLEDGEMENTS .....	x
 CHAPTER	
1 INTRODUCTION .....	1
1.1 Statement of the Problem.....	3
1.2 Purpose of the Study .....	3
1.3 Theoretical Bases and Organization .....	4
1.4 Limitations of the Study.....	5
2 METHODOLOGY .....	6
2.1 Sound and Signal Processing .....	6
2.2 Feature Extraction .....	9
2.3 Bayes Decision Rule .....	11
2.4 Hidden Markov Models .....	12
2.5 System Overview .....	23
2.6 Population or Sample .....	25

2.7 Treatment .....	26
2.7.1 Viper .....	26
2.7.2 Recording.....	27
2.7.3 Endpoint Detection .....	27
2.8 Data Analysis Procedures .....	29
2.8.1 Feature Extraction .....	29
2.8.2 Training.....	29
2.8.3 Classification.....	32
3 RESULTS AND DISCUSSION.....	34
4 SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS .....	36
REFERENCES .....	38

**LIST OF TABLES**

	PAGE
Table 2.1 Wolf Age and Gender Distribution.....	26
Table 2.2 Number of Examples Per Class .....	31
Table 2.3 HMM States Per Class .....	32
Table 3.1 Classification Results of Manually Identified Data.....	34

**LIST OF FIGURES**

	PAGE
Figure 2.1 Wolf howl spectrogram. ....	9
Figure 2.2 Observable Markov model. ....	13
Figure 2.3 Partial path example. ....	16
Figure 2.4 Constrained path probability. ....	20
Figure 2.5 HPWREN network topology. ....	24
Figure 2.6 Distributed component tasks. ....	25
Figure 2.7 Remote sensor at the California Wolf Center.....	27
Figure 2.8 Spectrograms of wolf calls. ....	30

## ACKNOWLEDGEMENTS

I would especially like to thank Dr. Marie Roch for her tireless effort and unending patience. I could not have completed this project without her help. I would like to thank Hans-Werner Braun of the HPWREN project and the National Science Foundation for their support of this project through the NSF grant 0426879.

Thanks to my thesis committee, Dr. Joseph Lewis and Dr. Roxana Smarandache. For their contribution I would like to thank Kim Miller and Melinda Booth at the California Wolf Center, Elizabeth Baker and Dan Moriarty at the University of San Diego, and Melissa Soldevilla at University of California San Diego.

I want to thank my fellow students Rhonda, Jim, and Shyam in Dr. Roch's lab. I have enjoyed the comraderie between the four of us.

And of course I want to thank my dear friends and family who have been supportive and encouraging in more ways than I can describe, and who have waited patiently for me to resurface.

## CHAPTER 1

### INTRODUCTION

The goal of many research studies in the biological sciences is to understand wildlife in our environment. Of these research projects, many are centered around studying the behavior of a particular animal species. Gathering behavioral data, including vocalization data, is fundamental to studying an animal's behavior. The amount of data that these studies require is often substantial. This can be an imposing task when handled manually.

By itself, collecting large amounts of data is not necessarily useful unless its analysis can produce meaningful results. One common analysis problem is distinguishing between classes of data. "Signal detection and classification are necessary to provide useful information about large acoustic datasets which cannot be effectively summarized by human staff due to cost and time constraints." [1]

This presents an opportunity to design and build a system that automates the tasks of collecting and analyzing behavioral data. The fact that the process of collecting the data can involve many repetitive, time-consuming tasks makes it ideally suited for computers. For example, signal detection is a good technique for automating the process of collecting vocalization samples. Pattern recognition methods can be used to solve the classification problem. Some examples of pattern recognition methods that have been used to automate classification are hidden Markov models, neural networks, and support vector machines.

The continuous advancement of hardware technology has resulted in remote sensors that are smaller, cheaper, more powerful, and more robust. Additionally, internet

connectivity in remote areas can provide access to data that would otherwise be difficult to obtain. These innovations have contributed to the feasibility of developing such a system.

Vocal production has been described using a source-filter model [2], where the source is the outflow of air from the lungs through the vocal folds, and the filter is the vocal tract. While it is generally accepted that this model is valid for human vocal production, research supports the idea that the source-filter model can be applied to nonhuman mammal vocalizations as well. According to both Titze [3] and Fitch [4] the physical mechanisms involved in vocal production across many mammal species are very similar. The source-filter model is based on these physical mechanisms, and therefore it is reasonable to extend this model to nonhuman mammals. In their study on vocal tract length and acoustics, Riede and Fitch [5] apply the source-filter model to the domestic dog (*Canis familiaris*). Another study by Fitch [6] on vocal tract and formant frequencies in the rhesus macaques (*Macaca mulatta*) is based on the idea that the principles of the source-filter model and acoustic phonetics apply to nonhuman vocalizations.

Many studies involving animal vocalizations focus on call type classification, species classification, and speaker identification. One study presents a method for identifying dolphin species by applying pattern recognition techniques to recorded vocalizations [7]. Specifically, Gaussian mixture models were used for the classification. Another study investigated whether timber wolves (*Canis lupus*) use variation in vocalizations as an aid in individual recognition [8]. In a study on low-frequency whale sounds, spectrogram correlation was evaluated as a possible classification method [9].

## **1.1 STATEMENT OF THE PROBLEM**

The objective of this research is to build a distributed signal detection and classification system to classify wolf call types from recordings obtained in a remote location. Hidden Markov models are used to perform the classification. The goal is to evaluate whether a pattern recognition system based on hidden Markov models can classify wolf call types with a reasonable level of success.

## **1.2 PURPOSE OF THE STUDY**

This study is done in cooperation with the California Wolf Center in Julian, California. The California Wolf Center provides a very important service. Their stated mission is “to increase awareness and conservation efforts in protecting and understanding the importance of all wildlife and wild lands by focusing on the history, biology and ecology of the North American Gray Wolf through education, exhibition, reproduction of endangered species and studies of captive wolf behavior.” [10] Research of wildlife and how we can better protect it has both immediate and long-lasting benefits to our environment.

This research is significant because it may assist the researchers and caretakers of the wolves at the California Wolf Center. Vocalizations are an important part of many animals' behavior. Being able to determine when or how frequently certain vocalizations occur can help to further understanding, care, and management of the wolves. Experts at the California Wolf Center have described situations where audio monitoring would be useful [11]. For example, sounds of increased aggression from the pack when new pups are present can indicate that the pack is not accepting the litter. Similarly, when wolves are put into a new pen, aggressive sounds can indicate that the animals are not getting along.

As part of the design of this study, we incorporated knowledge from the literature about wolf behavior as well as from local experts at the California Wolf Center. The list of wolf call types that we attempt to classify is derived from *Wolf Ethogram* [12]. An ethogram is a list of behaviors, including vocalizations, of a specific type of animal.

### **1.3 THEORETICAL BASES AND ORGANIZATION**

Research shows that hidden Markov models can be an effective method for classification of structured audio data. While it is commonly applied to human speech recognition applications, it also can be effective in research involving animal vocalizations. In a comparison of bird song classification performance between a dynamic time warping (DTW) technique and system based on hidden Markov models, it was found that the HMM-based system consistently outperformed the DTW-based technique. In particular, the HMM-based system was better able to handle relatively noisy conditions and calls that varied from the stereotypical call types [13]. A study on African elephants used hidden Markov models to investigate whether the vocalizations provide a sufficient basis for call type classification and speaker identification. These classification systems showed reasonably successful performance [14]. In another study, hidden Markov models were used to analyze vocalizations of red deer (*Cervus elaphus*) stags [15]. In this case, the vocalizations were found to have characteristics that could potentially uniquely identify each individual.

This paper will provide an overview of the theoretical and mathematical background for our chosen methods of signal detection and classification. The paper will relate that background to the specific details of our research project. Following will be the results of our research and a discussion of those results.

## 1.4 LIMITATIONS OF THE STUDY

Some limitations of the study exist. One of the system's components is a set of examples of known wolf call types that are obtained by a person listening to the wolf call and deciding how that call should be classified. This process is called manual labeling. One potential problem with manual labeling is that it is very subjective. Another potential issue is that sounds that are faint or disrupted by wind noise may be difficult to classify. The accuracy of the system depends on the consistency of the manual labeling. Therefore, the *Wolf Ethogram* [12] was used as a guide and labels were reviewed to ensure the manual labeling was consistent.

This research only uses data collected from a single location, the California Wolf Center. It is therefore not known how differences in physical location, geography, species, or specific animals would affect the performance of the developed system.

Studies have shown that for optimal performance, hidden Markov models require a substantial amount of training data. While the employed training data set is not minimal, it is possible that the results could have been improved with additional training data.

## **CHAPTER 2**

### **METHODOLOGY**

#### **2.1 SOUND AND SIGNAL PROCESSING**

This section provides an overview of how sound is produced and how it is represented digitally. Sound is caused by a vibrating object that creates a pressure wave. As the object vibrates it causes the surrounding molecules to compress and rarefy. This compression and rarefaction continue outward from the object until the pressure wave reaches your eardrum. Many sounds are caused by a complex combination of vibrations rather than a single vibrating object [16].

A sound pressure wave created by an oscillating source can be represented by a sinusoid, or sine wave. The energy of the pressure wave determines the sine wave amplitude, typically measured in decibels. The inverse wavelength of the pressure wave determines the sine wave frequency. As many sounds are caused by a combination of vibrations, the sound pressure wave can be thought of as being represented by the sum of each of the component sine waves. Another important characteristic of a sinusoidal signal is phase, which is related to timing [17].

Based on this abstract representation of a sound wave, we next describe how sound is represented digitally. The sampling theorem states that an analog signal can be uniquely recovered from the corresponding digital signal as long as the analog signal has no frequencies above the Nyquist limit, which is equal to half the sampling rate. To remove frequencies in the signal above the Nyquist limit, we apply an analog low-pass filter to the signal. The next step is sampling which takes a measurement, or sample, of the analog signal

at regular time intervals. This sample value represents the amplitude of the signal at that particular time. Finally, quantization maps the continuous sample value to a discrete integer. The result of these operations is a discrete signal consisting of a sequence of integers. This sequence is what is typically used to digitally represent sound in an audio file [16].

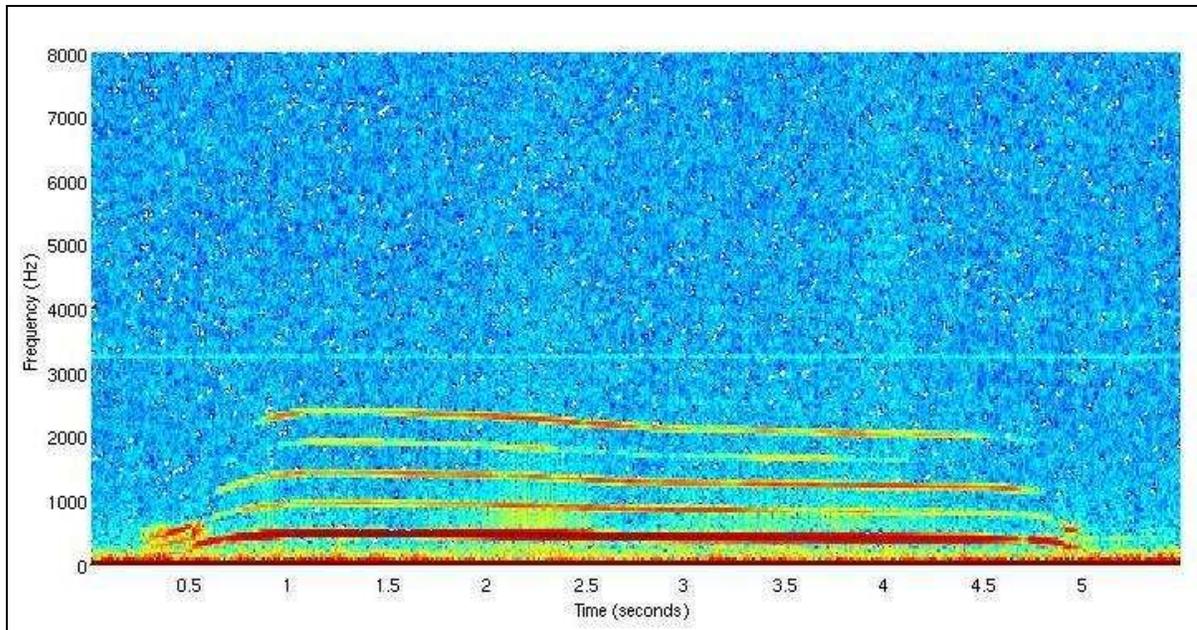
As mentioned earlier, complex sounds can be decomposed into a number of sinusoids, each sinusoid representing a specific amount of energy at a specific frequency. The result of this mapping to a linear combination of sinusoids is referred to as the frequency spectrum. A frequency spectrum can be very useful in that it allows one to examine the amount of energy in a particular frequency range. This can also be referred to as converting the information from the time domain to the frequency domain. There are several discrete-frequency transforms that can convert a discrete time signal into a discrete frequency representation.

The short-time discrete Fourier Transform is one such transform used to decompose a digital signal into its component digital frequency signals. A set of discrete target frequencies from the continuous frequency spectrum is obtained by dividing the frequency spectrum into bins spaced at regular intervals. The number of bins is governed by the length of the sampled audio to be analyzed. More bins result in a higher frequency resolution but with the trade-off that the time domain resolution will be poorer. Conversely, a smaller number of samples in the time domain results in better time resolution but at the expense of frequency resolution. Once we have selected our number of frequency bins and length of time, we can iteratively calculate the amplitude at each of those frequencies. The formal definition of the discrete Fourier transform for a signal  $x_N[n]$  with  $N$  samples is given by the following equation.

$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi n k / N} \quad 0 \leq k \leq N-1 \quad (1)$$

Short-time Fourier analysis sequentially processes small segments of data often referred to as frames. Mathematically this is accomplished by using a window function which is zero everywhere except for the section corresponding to the frame. Applying the window function to the signal gives us each frame. Frame size is typically between 20 and 30 milliseconds [17]. It is common to analyze overlapping frames, which can be done by setting a frame advance. To analyze a signal using short time Fourier analysis, the window function is iteratively applied to the signal, the first frame starting at the beginning of the signal and each successive frame starting at the current frame plus the frame advance.

The exact definition of the Fourier Transform requires knowledge of the signal for infinite time. Although the data in each frame is finite, the Fourier transform treats the data as though it were one period of a continuous periodic signal. The short-time discrete Fourier transform of a signal whose period is not equal to the frame length contains frequency components not present in the original signal. This occurrence is called spectral leakage and is a result of the discontinuity of the signal when the frame is repeated periodically. One way to reduce spectral leakage is to apply a window function such as a Hamming window prior to transforming the signal. Once the frequency spectrum at each frame is calculated, the results can be displayed in a format such as a spectrogram. Spectrograms allow one to visualize the dynamics of energy distribution changes over time. In a spectrogram, time is shown on the horizontal axis and frequency is shown on the vertical axis. Areas of higher energy are shown as darker or are otherwise distinguished by color intensity. Figure 2.1 is an example of a wolf howl spectrogram.



**Figure 2.1 Wolf howl spectrogram.**

## **2.2 FEATURE EXTRACTION**

Feature extraction is a term used to describe how audio data is processed before it is given to the classification component of a pattern recognition system. Feature extraction provides several benefits. Typically the amount of data that pattern recognition applications are required to process is very large. As much as hardware has improved in recent years, these applications still require a great amount of computational power. One benefit of feature extraction is that it reduces the overall amount of data being processed so that the performance in terms of speed is improved. Another benefit of feature extraction is more specific to the goal of pattern recognition applications, which is to be able to distinguish between different classes of data. Ideally, feature extraction enhances the differences between examples of different classes, which will result in better accuracy in classification.

Filterbanks are often used in feature extraction for audio signals. The motivation for using filterbanks is that it exploits mammals' inability to distinguish between closely related frequencies. The benefit of using filterbanks is it further reduces the amount of data being processed, resulting in faster performance. They also remove differences that are not readily distinguishable by mammalian auditory systems. Speech processing applications commonly use Mel filters, which are derived from the way humans perceive sounds with different frequencies [17]. For studies that involve nonhuman vocalizations, more neutral filters such as linearly spaced filters can be used effectively [7]. In feature extraction, the filter is applied to each frame's frequency spectrum.

Recall that we can model mammal call production as a source-filter model, where the source is the outflow of air from the lungs through the vocal folds and the filter is the vocal tract. Given this source-filter model, it would be helpful to separate the source from the filter with the idea that the vocal tract configuration is the primary factor in determining the characteristics of the sound that is produced. Cepstral processing allows us to do this by employing the property that the convolution of two signals is equal to the sum of the signals' cepstrums. In cepstral processing, the "cepstrum" is defined as the inverse Fourier transform of the log magnitude spectrum [18]. Although we could use the inverse Fourier transform to compute the cepstrum, another discrete-frequency transform called the discrete cosine transform (DCT) is commonly used. The following equation shows how to calculate the real cepstrum of a signal using the DCT.

$$C[k] = \sum_{n=0}^{N-1} X[n] \cos(\pi k(n + \frac{1}{2}) / N) \quad 0 \leq k \leq N \quad (2)$$

In human speech processing applications it is common to use only the first 12 coefficients of the cepstrum [17]. We found that by increasing the number of cepstral

coefficients to 18 resulted in better performance. A common technique to improve the error rate is to include the first and second derivatives of the cepstral coefficients. The first 18 coefficients with the first and second derivatives results in a 54 dimensional feature vector.

Another technique that can improve error rate is cepstral mean normalization. The purpose for using cepstral mean normalization is to increase robustness in varying environmental conditions [17]. In cepstral mean normalization, first the cepstrum is calculated from a signal by short time Fourier analysis. Next the mean of the cepstrum vectors is calculated and then subtracted from each vector so that they are normalized.

### 2.3 BAYES DECISION RULE

Our classification system makes use of Bayes decision rule, also known as the maximum *a posteriori* probability (MAP) decision rule. Suppose models  $\Phi_1, \Phi_2, \dots, \Phi_S$  each represent a different class. The problem we want to solve is to choose one class that best represents an observation. If we have no knowledge about the observation, then we may use the prior probability, which is simply to choose the class with the highest probability. The prior probability of a model  $\Phi$  is written  $p(\Phi)$ . The term "prior" is used because it is before we know about the observation. If we then have an observation sequence  $\mathbf{x}$ , we want to choose one of these classes that is mostly likely given that observation. This is the posterior probability of a model  $\Phi$  given  $\mathbf{x}$ , and is written as  $p(\Phi | \mathbf{x})$ . This is difficult to calculate directly, so we can rewrite this using Bayes rule as shown.

$$p(\Phi | \mathbf{x}) = \frac{p(\mathbf{x} | \Phi) p(\Phi)}{p(\mathbf{x})} \quad (3)$$

Note that  $p(\mathbf{x})$  remains constant for all classes, so we rewrite the above equation as  $p(\mathbf{x} | \Phi) p(\Phi)$ . We calculate this for each model  $\Phi_n$  and then choose the model that gives the

maximum probability. Bayes decision rule minimizes the overall risk with respect to a zero-one loss function.

## 2.4 HIDDEN MARKOV MODELS

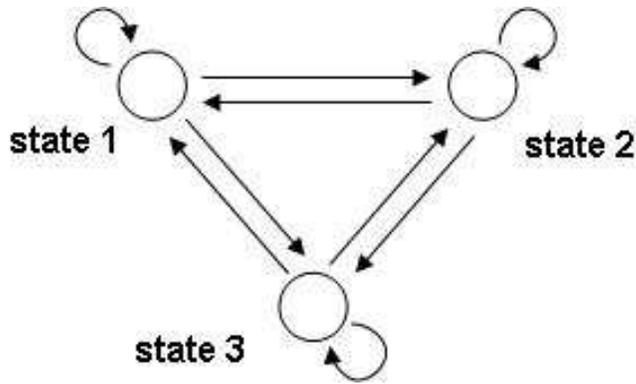
Hidden Markov models are one of several techniques commonly used in pattern recognition applications. While the basic theory of hidden Markov models was published in the late 1960's, interest in HMMs has increased over the last couple of decades. Here we will describe the basic theory of HMMs.

Signal models can be separated into two broad categories: deterministic and stochastic. Deterministic models generally use some known properties of the given signal, while stochastic models, including hidden Markov models, seek to characterize a signal based on some unknown random process or processes [19].

Hidden Markov models are an extension of the Markov chain, so we first give a brief definition of a Markov chain. Consider a chain of random events. These events could be completely independent of each other or have dependencies on other events. In a Markov chain, each event is dependent only on the previous event. In an observable Markov chain, events are associated with states, and each state represents a distribution of possible event outcomes. Thus the observable chain of random events is represented by a state sequence [19]. Figure 2.2 is a diagram illustrating an observable Markov model with 3 states.

Note that observable Markov chains have an output probability distribution for each state, and the state sequence can be observed. However, some processes have a state sequence that cannot be observed. By extending the definition of Markov chains to include chains where the state sequence is unobservable, we can adequately model those processes.

The sequence of state transitions is not known (i.e. "hidden") and also governed by a probability distribution.



**Figure 2.2 Observable Markov model.**

To make the definition more concrete, we will describe the parameters that characterize a hidden Markov model. In our description we will use  $\Phi(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$  to denote a hidden Markov model. Given a number of states  $N$ ,  $\boldsymbol{\pi} = \{ \pi_i \}$  where  $1 \leq i \leq N$  is the probability of starting in state  $i$ ,  $\mathbf{A} = \{ a_{ij} \}$  where  $1 \leq i, j \leq N$  is the probability of a transition from state  $i$  to state  $j$ , and  $\mathbf{B} = \{ b_i(x) \}$  where  $1 \leq i \leq N$  is the probability of seeing observation  $x$  while in state  $i$ . Note that although the initial probabilities  $\boldsymbol{\pi}$  and the state transition probabilities  $\mathbf{A}$  are separate parameters, together they characterize the state sequence. Other required notation includes  $\mathbf{X} = \{ X_1, X_2, \dots, X_T \}$  which represents an observation sequence from time 1 to time  $T$  governed by the parameter  $\mathbf{B}$ , and  $\mathbf{S} = \{ S_1, S_2, \dots, S_N \}$  which represents a state sequence governed by the parameter  $\mathbf{A}$ . The state distributions  $\mathbf{B}$  can either be discrete or continuous, and we will use a Gaussian mixture model (GMM) to represent the distribution of our continuous feature vectors. A Gaussian mixture model is a probability distribution that includes a number of component Gaussian mixtures, each with a

mixture weight. These models are useful for representing arbitrary distributions where a single distribution does not adequately represent the underlying data. A multivariate GMM is defined as:

$$b_j(\mathbf{x}) = \sum_{k=1}^M c_{jk} N(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}) \quad (4)$$

where  $N(\mathbf{x}, \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$  represents a Gaussian density function with mean vector  $\boldsymbol{\mu}_{jk}$  and covariance matrix  $\boldsymbol{\Sigma}_{jk}$ .  $c_{jk}$  is the weight for the  $k^{\text{th}}$  mixture associated with state  $j$ . The sum of the  $M$  mixture weights,  $\sum_{k=1}^M c_{jk}$ , must be 1 to ensure that  $b_j$  is a distribution.

The number of states per model is implicit in the hidden Markov model definition. Rabiner [19] describes two methods commonly used to select the number of states for a given model. One method bases the number of states on the number of distinguishable sounds within the signal, while the second method uses the average length of time of the observation sequences to determine the number of states. When Gaussian mixture models are used for the output distributions, the number of mixtures must be chosen as well.

There are three common problems associated with hidden Markov models. The first problem is to determine the probability of a sequence of observations with respect to a model. To find the probability of the observation sequence over a single state sequence path, we compute the product of the initial state probability, the transition probabilities for the path's state sequence, and the corresponding output probability for each state in the path. The equation for a state sequence probability is:

$$P(\mathbf{S}|\boldsymbol{\Phi}) = \pi_{s1} a_{s1s2} a_{s2s3} \dots a_{s(T-1)sT} \quad (5)$$

Using the same state sequence  $\mathbf{S}$ , the equation for the calculating the observation sequence probability is:

$$P(\mathbf{X}|\mathbf{S},\Phi) = b_{s_1}(X_1) b_{s_2}(X_2) \dots b_{s_T}(X_T) \quad (6)$$

Combining the above equations as a joint probability gives us the probability for a specific path, as shown here.

$$P(\mathbf{X}|\mathbf{S},\Phi) = \pi_{s_1} a_{s_1s_2} a_{s_2s_3} \dots a_{s_{(T-1)T}} b_{s_1}(X_1) b_{s_2}(X_2) \dots b_{s_T}(X_T) \quad (7)$$

As each path is a separate event and the state sequence is unknown, the probability of an observation sequence  $\mathbf{X}$  is the sum of the probabilities of all possible paths through the model. Summing over all state sequences  $\mathbf{S}$  gives us the probability for this observation given the model.

$$P(\mathbf{X}|\Phi) = \sum_{\text{all } \mathbf{S}} \pi_{s_1} a_{s_1s_2} a_{s_2s_3} \dots a_{s_{(T-1)T}} b_{s_1}(X_1) b_{s_2}(X_2) \dots b_{s_T}(X_T) \quad (8)$$

A practical problem arises here in that this computation has exponential complexity because of the number of states and observation sequence length. We can use dynamic programming principles to solve this problem. We will show a simple illustrative example, and then extend that example to our actual solution. Figure 2.3 shows two examples of partial paths through a 3-state hidden Markov model. Keep in mind that many other paths through this model are possible. The illustration shows that for these two particular paths, the computations from time 1 to time 3 are identical. Rather than recomputing these values for each path, we can compute this value once and then save our computation for reuse with other paths. This idea of storing and reusing partial computations is common to dynamic programming and can efficiently solve the exponential complexity problem.

We can extend this idea to store partial results at each time for all possible paths through each state. This method is known as the forward algorithm. Here we introduce a measure called the forward probability denoted  $\alpha_t(i)$ , which is the probability of being in state  $i$  at time  $t$  given the model and observation sequence. The first step of the forward algorithm

is to calculate the forward probability of starting in each state and observing  $X_1$  in that state, as shown in the following equation.

$$\alpha_1(i) = \pi_i b_i(X_1) \quad 1 \leq i \leq N \quad (9)$$

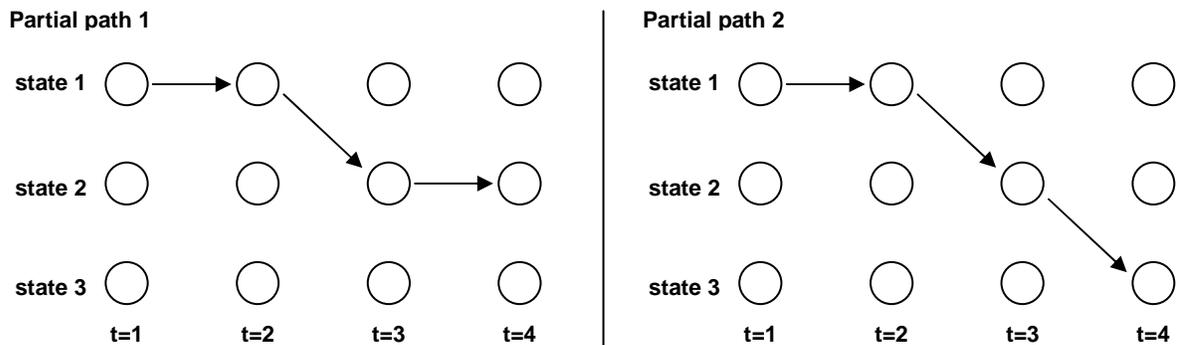
The next step is to iteratively evaluate the forward probability at time  $t = t + 1$  until the final time  $T$  is reached for each state, as shown in this equation.

$$\alpha_t(j) = \left[ \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} \right] b_j(X_t) \quad 2 \leq t \leq T; 1 \leq j \leq N \quad (10)$$

The final step is to sum the forward probabilities of all states in the final time  $T$ , as shown in this equation.

$$P(\mathbf{X}|\Phi) = \sum_{i=1}^N \alpha_T(i) \quad (11)$$

Ultimately, this algorithm sums the probability of the observation sequence over all paths.



**Figure 2.3 Partial path example.**

Now suppose we want to determine which path within the given model is most likely to produce the observation sequence. We can solve this problem by using the Viterbi algorithm which is similar to the forward algorithm except that rather than finding the sum of

all state sequence path probabilities, the goal is to find the single state sequence that yields the highest probability. Here we introduce a probability measure given by the best-path probability  $V_t(i)$  which is the probability of the most likely state sequence at time  $t$  ending in state  $i$ . This measure only gives the probability of this path, not the path itself. To keep track of the path we have a separate variable  $B_t(i)$  that stores the state that maximizes the probability of the path at time  $t$  in state  $i$ . For the first step in the Viterbi algorithm we calculate the best-path probability  $V_1(i)$  at time  $t=1$ :

$$V_1(i) = \pi_i b_i(X_1) \quad 1 \leq i \leq N \quad (12)$$

At time 1 there is no previous state, so  $B_1(i) = 0$ .

The next step in the Viterbi algorithm is to calculate the best-path probability for each observation, and the state at each time that gives the highest probability is stored so that the state sequence can be reconstructed. This is shown in the following equations.

$$V_t(j) = \max_{1 \leq i \leq N} V_{t-1}(i) a_{ij} b_j(X_t) \quad 2 \leq t \leq T; \quad 1 \leq j \leq N \quad (13)$$

$$B_t(j) = \arg \max_{1 \leq i \leq N} V_{t-1}(i) a_{ij} \quad 2 \leq t \leq T; \quad 1 \leq j \leq N \quad (14)$$

Finally, the final best-path probability and state are chosen. The maximum probability is equal to  $\max_{1 \leq i \leq N} [V_T(i)]$ . The final state denoted  $s_T$  is equal to  $\arg \max_{1 \leq i \leq N} [V_T(i)]$ .

To reconstruct the best state sequence we can backtrack through the best-path states that we saved in  $B_t(i)$  as shown here.

$$s_t = B_{t+1}(s_{t+1}) \quad t=T-1, T-2, \dots, 1 \quad (15)$$

These values are used to give the best state sequence  $S = (s_1, s_2, \dots, s_T)$ .

The Viterbi algorithm is used to calculate the probabilities of observation sequences for each model in a pattern recognition application. It is interesting that the probabilities

calculated in the Viterbi algorithm rarely differ significantly from those given by the forward algorithm. This is because of all the possible paths, the ones that are less likely do not contribute much to the overall probability of the observation sequence with respect to a model.

An implementation issue arises with both decoding algorithms. Consider the number of factors in the equations above, and that many of the factors are probabilities which means they are between 0 and 1. These calculations could very quickly result in an arithmetic underflow. For the Viterbi algorithm, we can perform these calculations using log probabilities and replacing the multiplication operations by addition. This is not easily done in the forward algorithm, in which case it is possible to use an algorithm to scale the probabilities such that they remain within the dynamic range of the hardware (see [17] for details).

The third problem is an optimization problem. Given a model and an observation sequence, we may want to adjust the parameters of the model to maximize the probability that the model generated the observation sequence. To solve this problem we can use the Baum-Welch algorithm. This method is sometimes called the forward-backward algorithm and is an instance of the Expectation-Maximization (EM) Algorithm.

The basic idea behind the Baum-Welch algorithm is to iteratively calculate the expected probabilities related to the training data and current model estimates, and then use a maximum likelihood estimator to find better parameters for the given model. With hidden Markov models, the missing information that we want to estimate is the hidden state sequence. When Gaussian mixture models are used to model the output, we also want to estimate the hidden mixture weights. The expectation step calculates estimates for the

hidden information, and then the maximization step determines new parameters based on those estimates. The expectation and maximization steps are repeated until the model parameters converge.

The first step in the Baum-Welch algorithm is to choose estimates for the initial model parameters. For the initial state distribution  $\boldsymbol{\pi}$  and the transition probability distribution  $\mathbf{A}$ , using either random or uniform estimates is almost always sufficient [19]. It is also common to start in state 1, giving an initial state distribution  $\boldsymbol{\pi} = \{\pi_1 = 1, \pi_2 \dots \pi_N = 0\}$  [17]. We will assume this initial state distribution and will discuss the Baum-Welch algorithm excluding  $\boldsymbol{\pi}$  from the re-estimation procedures.

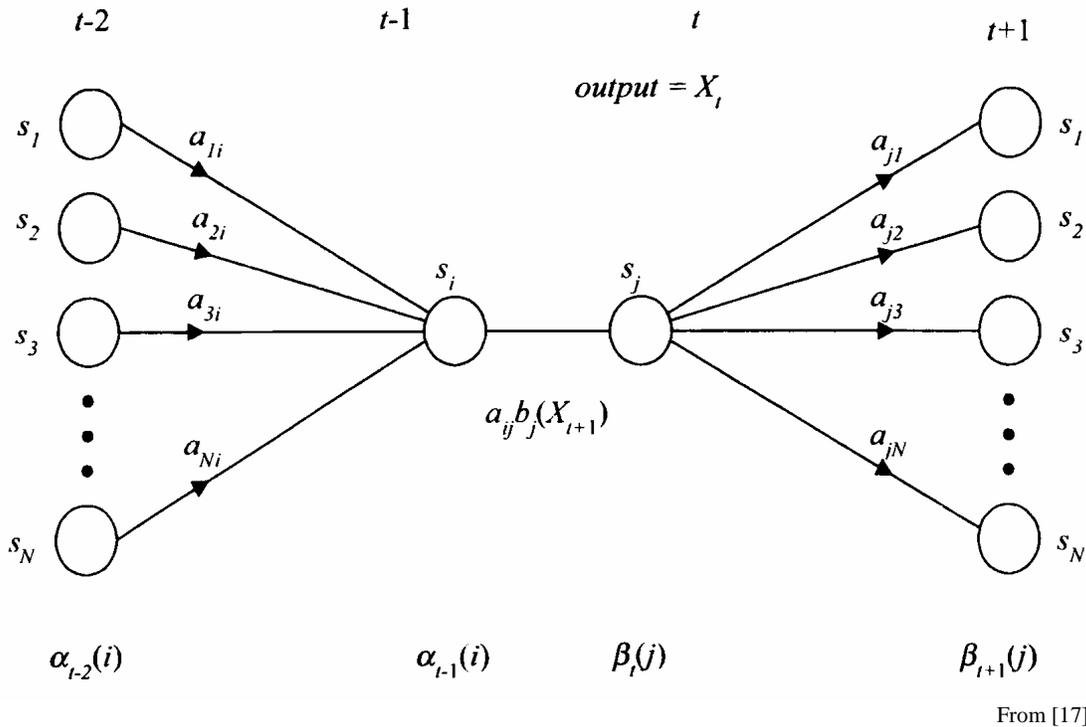
Good initial estimates are essential for the output probability distribution, particularly when the output is a continuous distribution. When Gaussian mixture models are used, one method for selecting initial estimates is to compute the grand mean and variance for all of the observations, and assume a single mixture. These GMM parameters are assigned to each state. After a few iterations of the EM algorithm the GMM parameters converge, at which time the mixtures are split. This process repeats until the number of desired mixtures is reached.

Now that we have initial parameter estimates, we perform the expectation step. We need to define the backward probability measure denoted  $\beta_t(i)$ , which gives the probability of being in state  $i$  at time  $t$  and generating the partial observation sequence from time  $t+1$  to  $T$ . The backward probability calculation is shown in the following equations.

$$\beta_T(i) = 1 / N \quad 1 \leq i \leq N \quad (16)$$

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(X_{t+1}) \beta_{t+1}(j) \quad t=T-1, T-2, \dots, 1; \quad 1 \leq i \leq N \quad (17)$$

For reestimating the transition probabilities, it would be useful to determine the probability of specific state transitions at specific times, given the observation and model. This calculation includes all paths going into a specific state  $i$  at time  $t$ , moving from state  $i$  to state  $j$ , observing  $X_t$ , and then all paths from state  $j$  to the end of the model. We already have the forward probability  $\alpha_{t-1}(i)$  to calculate the probability of all paths going into state  $i$  at time  $t-1$ . The transition probability  $a_{ij}$  gives the probability of the transition from state  $i$  to state  $j$ . The probability of the observation  $X_t$  is given by  $b_j(X_t)$ . The remaining piece is the backward probability  $\beta_t(j)$  which gives the probability of being in state  $j$  at time  $t$  and generating the partial observation sequence from time  $t$  to  $T$ . We are essentially constraining the path probability to a specific state transition at a specific time given the observation sequence and model, as illustrated in figure 2.4.



**Figure 2.4 Constrained path probability.**

This probability measure is denoted  $\gamma_t(i, j)$ , which gives the probability of a state transition from state  $i$  to  $j$  at time  $t$ . The equation for  $\gamma_t(i, j)$  is:

$$\gamma_t(i, j) = \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^N \alpha_T(k)} \quad 1 \leq i, j \leq N \quad (18)$$

As we mentioned, the values for  $\gamma_t(i, j)$  will be used to reestimate the transition probabilities. We will use another calculation,  $\zeta_t(j, k)$ , to reestimate the output probabilities. Note that this calculation applies specifically to multivariate Gaussian mixture density functions. The  $\zeta$  calculation is similar to the constrained path probability  $\gamma$ , but instead of a specific state transition at a specific time we want to find the probability of a specific mixture and state at a specific time. The probability measure  $\zeta_t(j, k)$  represents the probability of the observation in state  $j$  and mixture  $k$  at time  $t$  given the observation and the model. As in the equation for  $\gamma_t(i, j)$ , we use the forward probability  $\alpha_{t-1}(i)$ , the transition probability  $a_{ij}$ , the output probability  $b_j(X_t)$ , and the backward probability  $\beta_t(j)$ . The new component is  $c_{jk}$  which gives the mixture weight for state  $j$  and mixture  $k$ . The following equation gives the formal definition of  $\zeta_t(j, k)$ .

$$\zeta_t(j, k) = \frac{\sum_{i=1}^N \alpha_{t-1}(i) a_{ij} c_{jk} b_{jk}(\mathbf{x}_t) \beta_t(j)}{\sum_{i=1}^N \alpha_T(i)} \quad (19)$$

In the expectation step we calculate the  $\gamma_t(i, j)$ , and  $\zeta_t(j, k)$  which give us new values for the next maximization step.

In the maximization step, we maximize the model parameters **A** and **B** by applying reestimation equations to each parameter separately. First we define the reestimation equation for **A**. For each  $a_{ij}$  we calculate the number of transitions from a state  $i$  to state  $j$ , relative to all transitions from state  $i$ . The re-estimation equation for the state transitions is:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \gamma_t(i, j)}{\sum_{t=1}^T \sum_{k=1}^N \gamma_t(i, k)} \quad (20)$$

The purpose of using this equation is to find the percentage of each specific state transition  $i$  to  $j$  relative to all transitions out of state  $i$ , given our observation and model. This calculation translates to the new estimate for the transition probability.

Next we define the reestimation equations for  $\mathbf{B}$  which include reestimations of the mean, the covariance matrix, and the mixture weights. We have defined  $\zeta_t(j, k)$  as the probability of being in a particular mixture and state at a specific time, given the observation and model. We incorporate this value into the calculations for the new GMM parameters. The mean is calculated by essentially weighting each observation by its contribution to the given state and mixture before finding the observation mean. Similarly, the covariance parameter is calculated by using the weighted covariance for each observation and then finding the overall covariance. Using  $\zeta_t(j, k)$  for time  $t$ , state  $j$ , and mixture  $k$  we have the following reestimation equation for the GMM mean and covariance.

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k) \mathbf{x}_t}{\sum_{t=1}^T \zeta_t(j, k)} \quad (21)$$

$$\hat{\boldsymbol{\Sigma}}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k) (\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_{jk})'}{\sum_{t=1}^T \zeta_t(j, k)} \quad (22)$$

The mixture weights parameter is calculated by basically finding the contribution of the observation within the specific state and mixture to the all mixtures within that state. The reestimation equation for the mixture weights is defined by:

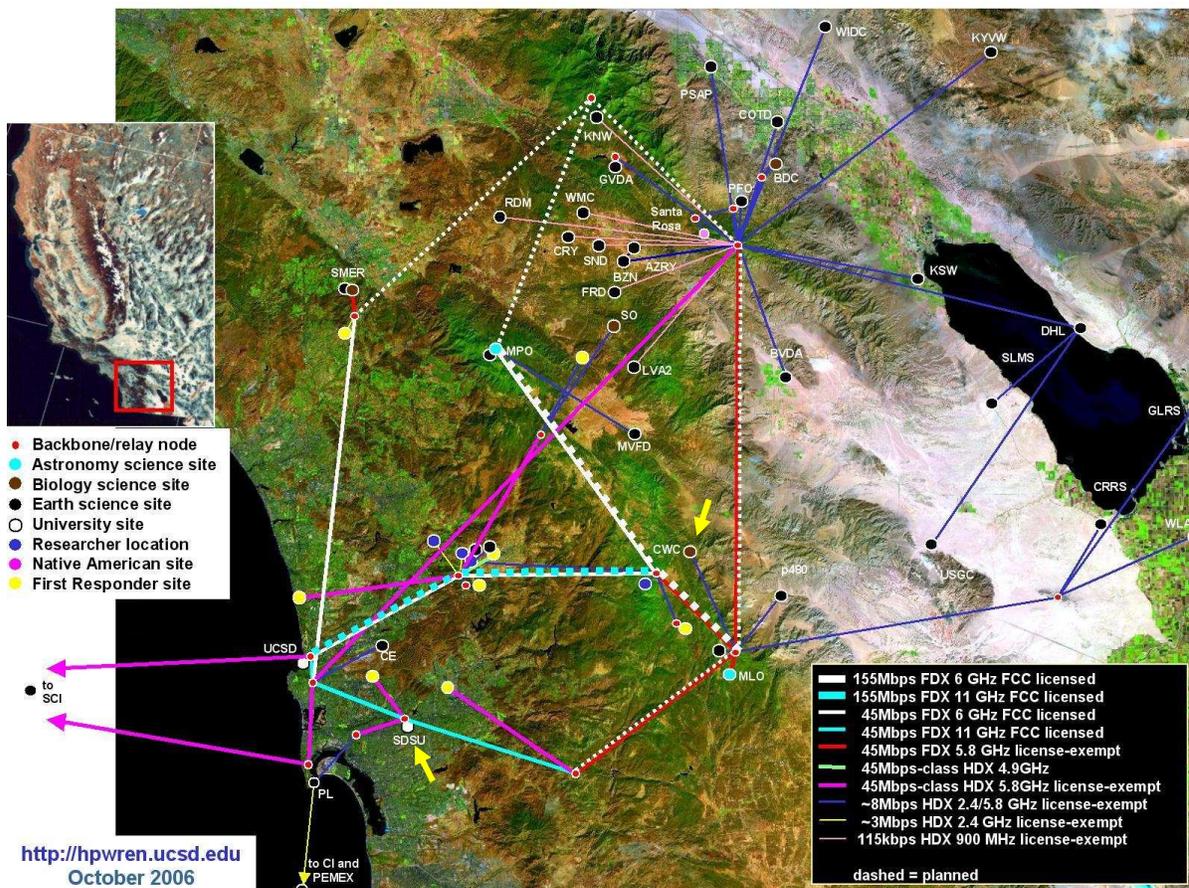
$$\hat{c}_{jk} = \frac{\sum_{t=1}^T \zeta_t(j, k)}{\sum_{t=1}^T \sum_{k=1}^M \zeta_t(j, k)} \quad (23)$$

The reestimation equations give the new model parameters that we will use in the next expectation step. We repeat these steps until the parameters converge. Each iteration of the EM algorithm is guaranteed to produce a model whose probability with respect to the training data is greater than or equal to the previous iteration [19]. Although there are no known proofs of the rate of convergence, convergence is typically fast requiring no more than 5 to 15 iterations.

## 2.5 SYSTEM OVERVIEW

This section gives a brief overview of the system. Details of each component will be addressed later. Our system is a distributed processing system that utilizes the NSF funded High Performance Wireless Research and Education Network (HPWREN), a high-speed wireless network [20]. HPWREN provides wireless internet access to a variety of projects that require network connectivity in remote areas throughout San Diego, Riverside, and Imperial counties. The HPWREN infrastructure provides the network connection between the California Wolf Center and the Speech Processing lab at SDSU. The network topology of HPWREN is shown in figure 2.5.

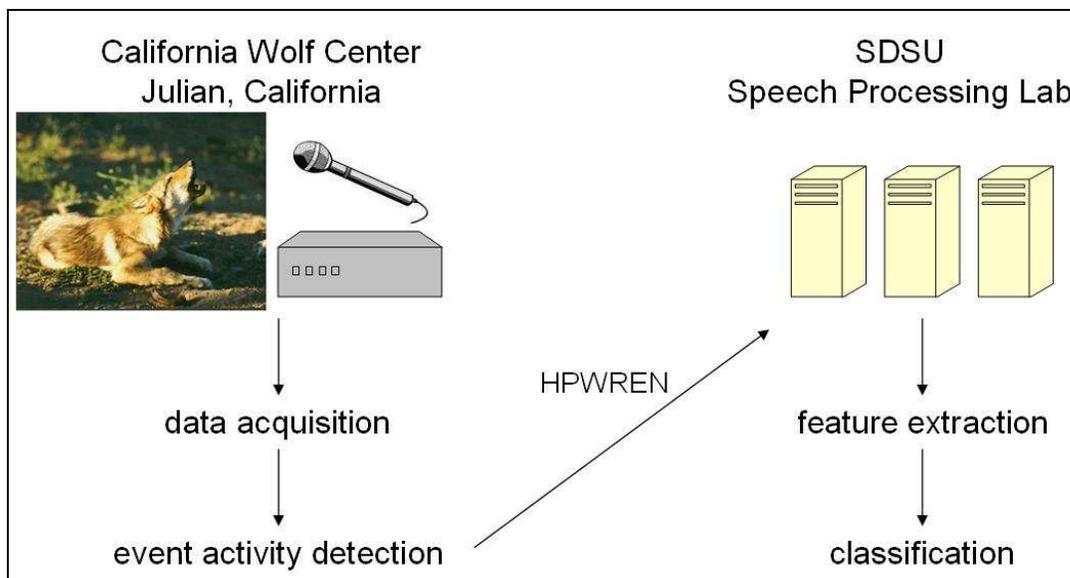
The distributed system can be divided into two main parts: data acquisition and classification. The data acquisition is done at the California Wolf Center using a remote sensor that is a node on HPWREN. This sensor is an embedded system with an attached microphone that is used to record audio data. Rather than transmitting all of the recorded data, an event activity detector identifies audio segments of interest and then transmits only



**Figure 2.5 HPWREN network topology.**

those segments to the SDSU processing lab, thereby conserving network bandwidth.

The classification part of the distributed system is done at the SDSU processing lab. Once the output from the remote sensor is transmitted, the classification component processes the data to prepare it for classification and then classifies the data using pre-trained hidden Markov models. Figure 2.6 is an illustration showing the distributed system and where each processing step occurs.



**Figure 2.6 Distributed component tasks.**

## 2.6 POPULATION OR SAMPLE

The California Wolf Center, located in Julian, California, is an education and conservation center focusing on the North American Gray Wolf. There are two subspecies of the Gray Wolf at the California Wolf Center – the Alaskan Gray Wolf and the Mexican Wolf. Currently, the Wolf Center has 28 total wolves. During most of the recordings there were 29 wolves. Of those, 17 are Alaskan wolves and 11 are Mexican wolves. The Wolf Center has six different enclosures separating the animals into individual packs. Two enclosures are for the Alaskan wolves, and four are for the Mexican wolves. The wolves in an individual enclosure constitute a pack. The four Mexican wolf packs have 4, 15, 2, and 2 wolves. The two Alaskan wolf packs have 15 and 2 wolves. The largest Alaskan wolf pack is most frequently exposed to people through educational programs and tours. The packs are labeled according to their species, A for Alaskan and M for Mexican, and by the enclosure number. Table 2.1 shows the age and gender of the distribution of wolves.

**Table 2.1 Wolf Age and Gender Distribution**

<b>Pack</b>	<b>Age and Gender</b>
A1	5 females, age ranging from 2 years old to 16 years old. 10 males, age ranging from 2 years old to 16 years old
A2	1 female age 11 years old and 1 male age 12 years old
M1	1 female age 12 years old and 1 male age 12 years old
M2	1 female age 5 years old and 1 male age 5 years old
M3	1 male age 12 years old and 2 females age 7 years old
M4	4 females age 3 years old

## 2.7 TREATMENT

### 2.7.1 Viper

The distributed processing system begins with recording at the California Wolf Center. The remote sensor at the California Wolf Center is an Arcom Viper running embedded Linux. It is a low-power single board computer with a 400 MHz ARM-compliant XScale RISC processor. It has 64 MB of memory, 1 GB flash RAM storage, 10/100baseTx Ethernet support, and other peripheral support including on board audio [21]. The sensor has an attached Labtec model Verse 524 Desktop microphone. It is enclosed in a weather-resilient container and is directly connected to the HPWREN network. Figure 2.7 shows photos of the remote sensor. The left picture shows the closed weather-proof enclosure. The right picture shows the remote sensor within the enclosure and a laptop attached temporarily for testing.

At the wolf center, the sensor is centrally located between the enclosures to maximize the number of wolf calls that are recorded. Of course when the sensor is recording, it records any sounds that occur, not just wolf calls. Other incidental sounds include birds and occasional airplanes. The sensor's central location sometimes results in the undesired effect



**Figure 2.7 Remote sensor at the California Wolf Center.**

of recording a lot of wind noise that interferes with the wolf vocalizations. We have a wind shield on the microphone constructed of a wire cage surrounded by fake fur, which helps to reduce the wind noise but does not eliminate it.

### **2.7.2 Recording**

The sensor is configured with an open source software utility called bplay/brec [22] to do the recording. The recording can either be started manually using HPWREN connectivity for testing and development, or it can be configured to run automatically when the classification system is running. We record the audio data at 16000 Hz for the classification system. Some data recorded earlier used as training data was recorded at 44100 Hz.

### **2.7.3 Endpoint Detection**

Audio files can be substantially large, which takes longer both in processing and in network transmission. It would be advantageous to distinguish sounds of interest (ideally, wolf calls) from background noise. We do this with an endpoint detector using signal processing techniques. The endpoint detector identifies segments within the recorded audio

stream where signals are found. Each detected segment is extracted from the original recorded file and transmitted across HPWREN to the Speech Processing lab at SDSU. Detecting the segments of interest and discarding the background noise data significantly reduces the amount of data being transmitted over the network.

Our endpoint detector is adapted from a joint project between Scripps Institution of Oceanography and SDSU. This endpoint detector is rule-based and uses a signal-to-noise ratio to designate where the sounds of interest start and stop. This SNR endpoint detector differs from standard rule based endpoint detectors in that it uses the peak frequency energy rather than the overall energy to determine when the threshold is reached.

The basic algorithm of the endpointer is to process the audio data stream by using short time Fourier analysis. We use 16 millisecond frames with a Hamming window, 1% frame overlap, and a 256 point Fourier transform to optimize for speed. Our signal-to-noise ratio threshold is 16 dB. We limit our analysis to a call bandwidth from 200 to 3500 Hz, which is where most of the wolf calls occur. The noise is calculated by averaging the energy of each frame within the specified call bandwidth over a 30 second moving window.

The endpoint detector moves forward through the audio data, calculating the noise within the call bandwidth. The endpoint detector then calculates the amount of energy within the same bandwidth at each frame and compares that with the noise. If the difference is above the threshold, the endpoint detector designates that as the start of a signal. The endpoint detector continues to move through the audio data, comparing each frame with the noise. When the energy falls below the SNR threshold, that designates the end of the signal. Detections that are shorter than .25 seconds are discarded. Remaining detections are padded by .15 seconds on either end, and then detections that are less than .15 seconds apart are

combined into a single detection. We extract these detected audio segments and automatically transmit them across the network from the remote sensor to the SDSU processing lab using the secure shell (SSH) protocol defined by RFC 4251 [23].

## **2.8 DATA ANALYSIS PROCEDURES**

### **2.8.1 Feature Extraction**

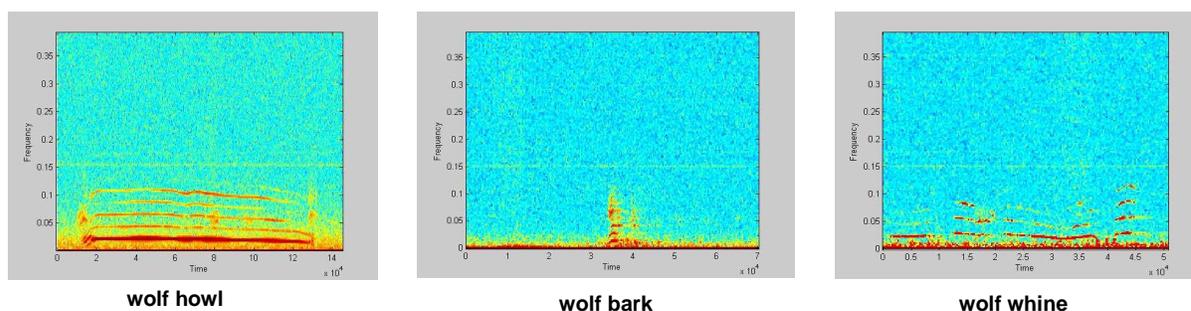
After we have the extracted audio detections at the SDSU processing lab, we perform feature extraction to obtain feature vector data which is the basis for our classification system. As was previously discussed, feature extraction is a method that extracts information to aid in classification while reducing the size of the data used in processing. A software package that we use extensively throughout the feature extraction and classification is HTK (Hidden Markov Model Toolkit) [24], an open source software toolkit that is used to build and manipulate hidden Markov models.

To perform feature extraction we use a 24 millisecond frame, a 10 millisecond advance, and a Hamming window. We apply a linear filter bank before obtaining the cepstrum. We use 18 cepstral coefficients and then include the first and second derivatives to improve classifier performance. We also use cepstral mean normalization to improve performance.

### **2.8.2 Training**

Training data is required to create hidden Markov models. Our system is a supervised learning system, which means that we use known examples of each class to train each of the corresponding models. We based our list of classes on the literature from Schassburger [25] and Goodmann, et al. [12]. For our training data, we collect examples of

each class that we want to be able to identify, and then we manually label each of the examples. Our list of wolf call classes is: howl, duet howl, chorus howl, bark, growl, growl-bark, whine, and yarl. A duet howl is two overlapping howls. A chorus howl is three or more wolves howling at once. A growl-bark is a string of barks in rapid succession that are too close together in time to separate into individual barks. A whine is a repeated sound, relatively brief, and falling in pitch. A yarl is similar to a growl, but have higher frequencies. Figure 2.8 shows some examples of spectrograms of wolf call types.



**Figure 2.8 Spectrograms of wolf calls.**

There are other wolf calls in the literature such as squeal and whimper that we excluded because we do not have sufficient data to model these call types. We also include two classes that are not wolf calls. Our recordings frequently include bird calls, and we grouped those together into a bird class. We also include an unknown class, to group together sounds that we record but do not specifically identify.

The process of manually labeling the data consists of listening to a portion of the recordings and manually labeling each sound accordingly. To label our data, we used an open source audio utility called Wavesurfer [26]. Wavesurfer is used to open audio files and display a graphical representation of the data, such as a spectrogram, while it plays the audio.

It also allows the listener to save start/stop times with a text label such as 'howl' or 'bark' alongside the visual display.

Having a sufficient number of examples of each class is important to having a good pattern recognition system. We have 11 hours of labeled data. Table 2.2 shows the number of training and test examples we used per class.

**Table 2.2 Number of Examples Per Class**

<b>Class</b>	<b>Number of Examples</b>	<b>Training Examples</b>	<b>Test Examples</b>
howl	172	94	78
duet howl	26	11	15
chorus howl	39	22	17
bark	145	88	57
growl	95	46	49
growl-bark	91	39	52
whine	91	88	3
yarl	34	27	7
bird	824	443	381
unknown	172	82	90

After we have our labeled training data, we perform feature extraction on the data to obtain the feature vectors that we will use to train the models. We then apply the Baum-Welch algorithm described earlier to train a hidden Markov model for each class. To do this we use Python programs to interface with low level HTK utilities. When the model has been created and trained, HTK generates a file known as a model definition file that represents the model. This file contains the model parameters such as the transition probabilities and output probabilities that we described earlier in the hidden Markov model overview.

Many human speech recognition systems use HMMs that all have the same number of states. The reason for this is that those systems use subword-level training, and the subword components are selected based on the number of frequency distribution changes. In our system we use models that each represent an entire call. Given that some call types are typically significantly longer than others, we chose to vary the number of states according the class. As initial estimates, we assigned the number of states as a function of call length. We then varied the number states to find reasonable estimates. Table 2.3 lists the number of states per class.

**Table 2.3 HMM States Per Class**

<b>Class</b>	<b>Number of states per class</b>
howl	30
chorus howl	40
bark	10
growl	20
whine	15
yarl	15
bird	10
unknown	10

### **2.8.3 Classification**

Recall that the remote sensor transmits candidate calls in the form of audio segments to the SDSU processing lab. We have a Python script that receives these incoming audio segments and processes them one at a time. The script first performs feature extraction on the audio segment as described earlier, and then the Viterbi likelihood for each class is computed. The call to the low level HTK interface requires parameters such as the list of

classes/models, the HTK-generated model definitions, and a grammar. The grammar specifies sequences of permissible calls. We assume that each segment contains a single one or more calls from the ethogram specified in section 2.5.2. The MAP decision rule is used to decide the class label. HTK generates a recognition file showing how the segment was classified.

## CHAPTER 3

### RESULTS AND DISCUSSION

The classifier was tested on the dataset described in the previous chapter.

Experiments were conducted both using the manually identified segmentations of calls, and a segmentation produced by the automated call detection routines.

The overall accuracy of our classifier on a corpus of 749 manually identified test calls is approximately 75%. Table 3.1 is a confusion matrix showing the accuracy per individual call type. In this table, the column labeled "% correct" is the percentage of the given call type that was correctly labeled. The column labeled "% error" is the percentage of incorrectly classified calls of the given call type relative to the total number of calls.

**Table 3.1 Classification Results of Manually Identified Data**

		classifications										
		bark	bird	growl	growl-bark	howl	chorus howl	duet howl	whine	yarl	% correct	% error
test calls	bark	43	10	0	4	0	0	0	0	0	75.4	1.9
	bird	50	417	0	4	0	0	0	0	0	88.5	7.2
	growl	4	37	0	5	3	0	0	0	0	0.0	6.5
	growl-bark	21	9	0	22	0	0	0	0	0	42.3	4.0
	howl	0	4	0	4	58	0	12	0	0	74.4	2.7
	chorus howl	0	0	0	0	0	12	5	0	0	70.6	0.7
	duet howl	0	0	0	0	5	1	9	0	0	60.0	0.8
	whine	2	1	0	0	0	0	0	0	0	0.0	0.4
	yarl	1	0	0	3	0	0	0	0	3	42.9	0.5

We used 32-mixture GMMs to represent components of each call type. Most calls were represented as a single component with the exception of growl-barks and howl duets,

which consisted of two more consecutive barks or howls respectively. Our HMMs have a number of states as a function of the length of the call (see Table 3).

Wolf vocalizations vary in loudness. For example, it is generally accepted that wolf howls are intended for communication over distances [25, 27] and therefore are required to be louder than other types of vocalizations. Growls on the other hand are intended to communicate with wolves that are in close proximity. The result of this is that a sensor that is well-suited for recording howls may not record other types of vocalizations as well if it is not in close proximity to the animals. The sensor used to collect the corpus is located far from the feeding area where the majority of growls, whines, and yarls are produced. For these call types, collecting more examples with a sensor that is located closer to the wolves may be beneficial. A second sensor has recently been added in proximity to the area in which packs A-1 and M-4 are fed, and we are in the process of collecting additional data for these call types.

A fairly significant number of growl-barks are being misclassified as barks. The structure of a growl-bark is a string of barks so closely timed such that it may be difficult for humans to identify the individual barks within the call. These misclassifications in our system suggest that the structure of the growl-bark components are in fact very similar to an individual bark.

A similar misclassification occurs with howls. The three howl call types: howl, duet howl, and chorus howl, all have similar components. The misclassifications of these call types are most frequently one of the other howl call types.

## **CHAPTER 4**

### **SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS**

We have implemented a distributed pattern recognition system using hidden Markov models. Our system performs classification of calls produced by wolves at a remote conservation, education, and breeding facility. The remote sensor records audio data, processes it with a signal-to-noise ratio endpoint detector, and then automatically transmits the data to the SDSU Speech Processing lab where we perform the classification. Under tested conditions with low wind noise, our system performs with an accuracy rate of 75%.

Our project did not focus on hardware optimization, such as selection of sensors or microphones, which would be an opportunity for research. Recordings taken over a longer period of time could potentially result in a more robust system. Our data was solely from one physical and geographical location.

An ongoing challenge with our project has been handling wind noise that appears in some of the recordings. Based on data obtained from a nearby weather station [27] we observed a positive correlation between a southwest wind direction and the amount of wind noise recorded. Therefore, one possible solution for handling the wind noise would be to automatically retrieve the wind direction and temporarily stop recording. When the wind direction changes then recording could resume.

There are a number of possibilities for future research with regard to our project. While our current system is based on recordings taken primarily from one sensor, the use of

multiple microphones or microphone arrays would provide an opportunity for research in the areas of denoising, and localizing the signal. Pairing the audio data with visual data may provide more complete information about the wolves, which could assist in educational programs or wolf caretaking. Studies could be done involving the seasonal or daily behavioral patterns of wolves. A possible area of research is to investigate how wolves are impacted by the presence of anthropogenic sounds.

In recent years there has been increasing interest in wolves in the wild. Boitani [28] describes the dramatic changes in wolf populations throughout history, including the recent history of the United States. As Europeans settled in North America, wolves were actively pursued with the intent of extermination. By 1930, the wolf population had essentially disappeared from the continental United States. Fortunately, views have begun to change and in the 1970's wolves were given protection under the Endangered Species Act. In 1995 wolves were reintroduced into Yellowstone National Park and central Idaho.

Although wolves are still protected in the United States, some populations' designations have been recently downlisted from endangered to threatened. The presence of wolves remains controversial. Past and present research continues to evaluate the impact of wolves with regard to issues such as livestock depredation, predator-prey relationships and restoration of wolf populations.

Passive acoustic monitoring has the potential to support such research. For example, a study by McDonald and Fox [29] used long-term passive acoustic monitoring to estimate the fin whale population. Applying passive acoustic monitoring to wolf populations in the wild could provide researchers with useful information that could further conservation and education efforts.

## REFERENCES

- [1] L. Girod, and M. A. Roch, "An Overview of the Use of Remote Embedded Sensors for Audio Acquisition and Processing," in Eighth IEEE International Symposium on Multimedia (ISM'06), pp. 567-574, 2006.
- [2] G. Fant, *Acoustic Theory of Speech Production with calculations based on X-ray studies of Russian articulations*, Mouton, 1970.
- [3] I. R. Titze, *Principles of Voice Production*, Prentice Hall, 1994.
- [4] W. Fitch, "The evolution of speech: a comparative review", *Trends in Cognitive Sciences*, vol. 4, pp. 258-267, July 2000.
- [5] T. Riede, and T. Fitch, "Vocal Tract Length and Acoustics of Vocalization in the Domestic Dog (*Canis familiaris*)", *The Journal of Experimental Biology*, vol. 202, issue 20, pp. 2859-2867, 1999.
- [6] W. Fitch, "Vocal tract length and formant frequency dispersion correlate with body size in rhesus macaques", *The Journal of the Acoustical Society of America*, vol. 102, issue 2, pp.1213-1222, August 1997.
- [7] M. A. Roch, M. S. Soldevilla, J. C. Burtenshaw, E. Henderson, and J. A. Hildebrand, "Gaussian mixture model classification of odontocetes in the Southern California Bight and the Gulf of California", *The Journal of the Acoustical Society of America*, vol. 121, issue 3, pp. 1737-1748, March 2007.
- [8] J. A. Goldman, D. P. Phillips, and J. C. Fentress, "An acoustic basis for maternal recognition in timber wolves (*Canis lupus*)?", *The Journal of the Acoustical Society of America*, vol. 97, issue 3, pp. 1970-1973, March 1995.
- [9] D. Mellinger, and C. Clark, "Recognizing transient low-frequency whale sounds by spectrogram correlation", *The Journal of the Acoustical Society of America*, vol. 107, issue 6, pp. 3518-3529, June 2000.
- [10] California Wolf Center. Available from <http://www.californiawolfcenter.org> accessed April 2007.
- [11] K. Miller, California Wolf Center, discussion via e-mail 4/18/07.
- [12] P. A. Goodman, E. Klinghammer, and M. Sloan, *Wolf Ethogram*, North American Wildlife Park Foundation, 1990.
- [13] J. A. Kogan, and D. Margoliash, "Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden Markov models: A

- comparative study", *The Journal of the Acoustical Society of America*, vol. 103, issue 4, pp. 2185-2196, April 1998.
- [14] P. J. Clemins, M. T. Johnson, K. M. Leong, and A. Savage, "Automatic classification and speaker identification of African elephant (*Loxodonta africana*) vocalizations", *The Journal of the Acoustical Society of America*, vol. 117, issue 2, pp. 956-963, February 2005.
- [15] D. Reby, R. Andre-Obrecht, A. Galinier, and J. Farinas, "Cepstral coefficients and hidden Markov models reveal idiosyncratic voice characteristics in red deer (*Cervus elaphus*) stags", *The Journal of the Acoustical Society of America*, vol. 120, issue 6, pp. 4080-4089, December 2006.
- [16] T. Kientzle, *A Programmer's Guide to Sound*, Addison-Wesley, 1997.
- [17] X. Huang, A. Acero, and H.-W. Hon, *Spoken Language Processing*, Prentice Hall, 2001.
- [18] J. Picone, "Signal Modeling Techniques In Speech Recognition", *Proceedings of the IEEE*, vol. 81, issue 9, pp. 1215-1247, September 1993.
- [19] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, issue 2, pp. 257-286, February 1989.
- [20] High Performance Wireless Research and Education Network. Available from <http://hpwren.ucsd.edu> accessed April 2007.
- [21] Arcom Embedded Linux Technical Manual.
- [22] bplay/brec utility. Source code available from <ftp://sunsite.unc.edu/pub/Linux/apps/sound/players/bplay-0.97.tar.gz>.
- [23] RFC 4251. Available from <http://www.ietf.org/rfc/rfc4251.txt>.
- [24] HTK home page. Available from <http://htk.eng.cam.ac.uk>.
- [25] R. M. Schassburger, "Wolf vocalization: An integrated model of structure, motivation and ontogeny," 1987.
- [26] Wavesurfer home page. Available from <http://www.speech.kth.se/wavesurfer>.
- [27] Weather Underground web page. Available from <http://www.wunderground.com/weatherstation/WXDailyHistory.asp?ID=KCAJULIA2>.
- [28] L. D. Mech, and L. Boitani, *Wolves: Behavior, Ecology and Conservation*, The University of Chicago Press, 2003.

- [29] M. McDonald, and C. Fox, "Passive acoustic methods applied to fin whale population density estimation", *The Journal of the Acoustical Society of America*, vol. 105, issue 5, pp. 2643-2651, May 1999.

## ABSTRACT OF THE THESIS

Classification of Wolf Call Types Using Remote Sensor Technology

by

Deborah Curless

Master of Science in Computer Science

San Diego State University, 2007

There is an increasing amount of research with the goal of understanding wildlife found in our environment. Observing the behavior of a species, including vocalizations, is fundamental to this goal. Researchers in the biological sciences have traditionally had to gather their observations manually, with a great deal of labor-intensive tasks. It would be beneficial to design and build a system that automatically gathers and analyzes this behavioral data.

This thesis presents such a system. Our system starts with a remote sensor that uses digital signal processing to automate data acquisition. The system then sends the acquired data to a remote lab via a high-speed wireless network for processing. Once the data is in the lab, our system classifies the data using hidden Markov models. The goals of this research are to build this system with the best possible level of performance, and to answer whether a pattern recognition system based on hidden Markov models can classify wolf call types with a reasonable level of success.