SOUND IDENTIFICATION OF LOONS

A Design Project Report
Presented to the Engineering Division of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering (Electrical)

by
Sajeev Chandrashekhar Krishnan
Project Advisor: Bruce Robert Land
Degree Date: January 2011

Abstract

Master of Electrical Engineering Program
Cornell University
Design Project Report

Project Title: Sound identification of loons

Author: Sajeev Chandrashekhar Krishnan

Abstract: The focus of this project is sound identification of birds, more specifically loons, through their calls or songs. There are many methods of sound identification that exist today, but most of them are constrained by the fact that they either require individuals to have similar call-types or do not take into account the change in vocal characteristics over time. Also, there is some loss in accuracy due to background noise or varying degrees of recording quality. There was some interesting research published that applied human-speech recognition methods for classification of birds to some success, but even this was limited by its lack of temporal stability. Another interesting characteristic of the loons is that their call (or song) changes with a change in location. The objective is to determine a scheme of sound identification for loon individuals with feature extraction that is stable over time and location, which would greatly increase the ease with which these birds are studied.

Report Approved by

Project Advisor: Bruce Robert Land_____Date: _____

ACKNOWLEDGMENTS

First and foremost I would like to thank Prof. Bruce Land for giving me an opportunity to work on this project. His knowledge and enthusiasm throughout the project gave me the help and confidence necessary to complete this project.

I would like to thank Prof. Charles Walcott, his assistance, guidance and support were instrumental in making this project a success.

I would also like to thank Ms. Laura Hou for her patience and willingness to help me whenever I needed it.

Finally, I would like to thank my Parents, without whom all of this would not have been possible.

EXECUTIVE SUMMARY

INTRODUCTION:

The focus of this project is acoustic identification of the loon. Individual loons have unique yodels, and these yodels show little variation within a specific time frame. It has been observed that loons tend to change their yodels with a change in location. Also, the vocal characteristics of the loon change over time, consequently affecting their yodels; also the loon changes its yodel with a change in location. This results in a number of issues in identification of loons and hence conventional forms of acoustic identification cannot be applied. The objective of this project is to determine a scheme of acoustic identification of loons, which is stable regardless of any changes to their vocal repertoire.

ISSUES TO BE ADDRESSED:

Call independent identification: The loon yodel changes with a change in location, hence a scheme of feature extraction has to be devised that is not dependent on the yodel itself, thus making it stable over various locations.

Temporal stability: The vocal characteristics of the loon, or any animal for that matter, changes with time. There is a reasonable amount of variability in loon yodels due to changes over time and this factor in identification must be appropriately handled.

Background Noise: The scheme to be used for feature extraction, MFCC – Mel-frequency Cepstral Coefficients, is not very robust in the presence of additive noise. Hence, the proposed solution should contain some scheme that would sufficiently mitigate the effect of background noise without a significant loss in the quality of the recording.

APPROACH:

There was a considerable amount of success achieved in applying human speaker recognition schemes for identification of individual animals. MFCC's (Mel- Frequency Cepstral coefficients) are the most popular speech-processing features. They are widely used in speaker recognition systems and are also extensively used in music information retrieval systems.

2. IDENTIFICATION

In 1980, Linde-Buzo-Gray proposed a VQ design algorithm, known as VQ-LBG, based on a training sequence. The initial codevector is obtained by finding the average of the whole cluster; two codevectors are obtained from the initial codevector by splitting the whole cluster into two regions. The iterative algorithm uses these two codevectors as the initial codevector to compute more codevectors, and till we get a codebook with the desired number of codevectors.

In this project the MFCCs derived from the songs of reference subjects and test subjects are vector quantized to obtain satisfactory codebooks. These codebooks are then compared to determine identity of the test subject.

CONCLUSION:

Past research has shown that MFCCs are text/call independent and vector quantization tends to provide satisfactory results even with change in vocal characteristics over time. Although there are certain misclassifications, from the results it is evident that the program performs as originally intended and the program proves to be robust and functional overall

INTRODUCTION

The major issue with physical marking of individual animals is that the procedure is invasive and cumbersome for both the researcher and the animal itself. Sound or Acoustic identification of birds proves to be an effective solution for this particular issue. The focus of this project is acoustic identification of the loon. Individual loons have unique yodels, and these yodels show little variation within a specific time frame. It has been observed that loons tend to change their yodels with a change in location. Call-independent identification is most important for use in species with complex and changing repertoires; the vocal characteristics of the loon changes over time, consequently affecting their yodels, but the variability in the loons' yodels was greater due to a change in location than due to the change in vocal characteristics [Walcott et. al. 2005]. This results in a number of issues in identification of loons and hence conventional forms of acoustic identification, like discriminant function analysis or spectrographic correlation, cannot be applied. The objective of this project is to determine a scheme of acoustic identification of loons, which is stable regardless of any changes to their vocal repertoire.

LOONS

The **loons** (North America) or **divers** (UK/Ireland) are a group of aquatic birds found in many parts of North America and northern Eurasia (Europe, Asia and debatably Africa). All living species of loons are members of one genus (***Gavia***), family (**Gaviidae**) and order (**Gaviiformes**) of their own.

Characteristics

The loons are the size of a large duck or small goose, which they somewhat resemble in shape when swimming. Flying loons resemble a plump goose with a seagull's wings, which seem quite small in proportion to the bulky body. Males are a bit larger on average, but usually this is only conspicuous when directly comparing the two parents. Males and females do not differ in plumage. In winter plumage, they are dark gray above, with some indistinct lighter mottling on the wings, and a white chin, throat and underside. The species can then be distinguished by certain features, such as size and colour of head, neck, back and bill, but often reliable identification of wintering divers by appearance is tough even for experts – particularly as the smaller immature birds look similar to winter-plumage adults, making size also not fully reliable.

Vocalizations

Loons are characterized by their vocalizations, there are these following types;
Hoot (short, single low note) Given by males and females of all ages, beginning when babies are two or three months old. This is a **contact call**, usually given by an individual when it's approaching a group or by individuals within a flock, as during social gatherings "Cluck" similar to hoot given by parents to hatching eggs or when chicks are about to enter the water, serving to encourage the babies.
Tremolo (sounds like the loon is laughing) Given by males and females of all ages, beginning when babies are two or three months old. This is a **distress call**, which suggests that a loon is stressed, often before it escapes. In territorial disputes, the loon who is being chased is the one who most often gives the tremolo call. In defense of nests and chicks, loons give the tremolo call often while in the "penguin posture," an aggressive position, though after giving this call a few times, sometimes the loon flees. When pairs give the tremolo call in a duet, it sometimes is a territorial statement. Often given as a **flight call**, when approaching, leaving, or flying over a lake. The precise meaning of this call is not understood.
Wail (sounds similar to the howl of a wolf or coyote) Given by males and females of all ages, beginning when babies are in their first week. This is a **contact call**; when one loon gives it, the distance between that loon and another loon will soon be smaller. This call helps a young loon to bring its parent(s) closer, and helps an adult to bring its young and/or its mate closer.
Yodel (several upslurred introductory notes, similar to wails, and then loud repeated phrases) Given by males in spring and early summer, especially at night. The yodel is a **territorial call** Its purpose is to alert other males that this loon is defending a territory. Yodels are often answered by other males.

NEED FOR IDENTIFICATION

According to [Walcott et. al, 2006] there are at least four possible reasons why males change their yodels when they switch territories.

1. The change in the male's yodel may be related to a change in the female. During a male take-over, the female remains on the territory or, during a female take-over, the male remains. Female takeovers had no effect on the male's yodel, suggesting that it is the change in territory and not the change in female that triggers the male to change its yodel.

2. A loon that has been forcibly displaced might also change his yodel so that other loons would not recognize his vulnerability. However, if loons attempt to avoid indicating their vulnerability to other male loons by changing their yodels, one would not expect new territorial residents to change their yodels. In fact both displaced males and new territory owners changed their yodels.

3. A male might change his yodel in such a way as to maximize the difference between his yodel and that of his new neighbors. For about half the loons in our study, yodels of resident males did become increasingly different from their neighbors', but for almost as many others the difference decreased.

4. An intruder taking over a previous male's territory might change his yodel to imitate that of the male that he displaced, the limited data (nine cases) suggest exactly the opposite; all intruding loons changed their yodels to sound less like that of the previous territorial male. Thus, the change in yodel does not appear to be a consequence of displacing or being displaced. Rather it appears to be associated with the change in territory itself or the yodel of the previous resident.

Loons exhibit characteristic behavior in the fact that they not only change their yodel when they change territory, but also in that they are aware of the yodel of the resident it replaces in the new lake. This is done so as to make its own yodel as different from the previous resident's yodel as possible. Since loons are threatened in much of the Eastern United States, there is great interest in being able to identify individual loons without the necessity of capturing and banding them. If individual recognition is to be found in any of the loon's vocalizations, the yodel is the most likely. This makes identifying individual loons by their respective calls feasible.

ISSUES

Call independent identification: The loon yodel changes with a change in location, hence a scheme of feature extraction has to be devised that is not dependent on the characteristics of the yodel itself rather on the characteristics of the voice, thus making it stable over various locations.

Temporal stability: The vocal characteristics of the loon, or any animal for that matter, changes with time. There are several factors that cause this change such as sex, age etc. Although the variability in loon yodels due to changes over time are not as great as the variability caused due to a change in location, this remains a factor in identification and must be appropriately handled.

Background Noise: Most of the songs were recorded in the field and invariably has a certain amount of background noise. The scheme to be used for feature extraction, MFCC – Mel-frequency Cepstral Coefficients, is not very robust in the presence of additive noise. Hence, the proposed solution should contain some scheme that would sufficiently mitigate the effect of background noise without a significant loss in the quality of the recording.

It is clear that with only call-dependent identification, acoustic individual identification is limited to species with extensive call sharing and no change in an individual's repertoire over time.

Highly desirable features of an acoustic identification technique are:

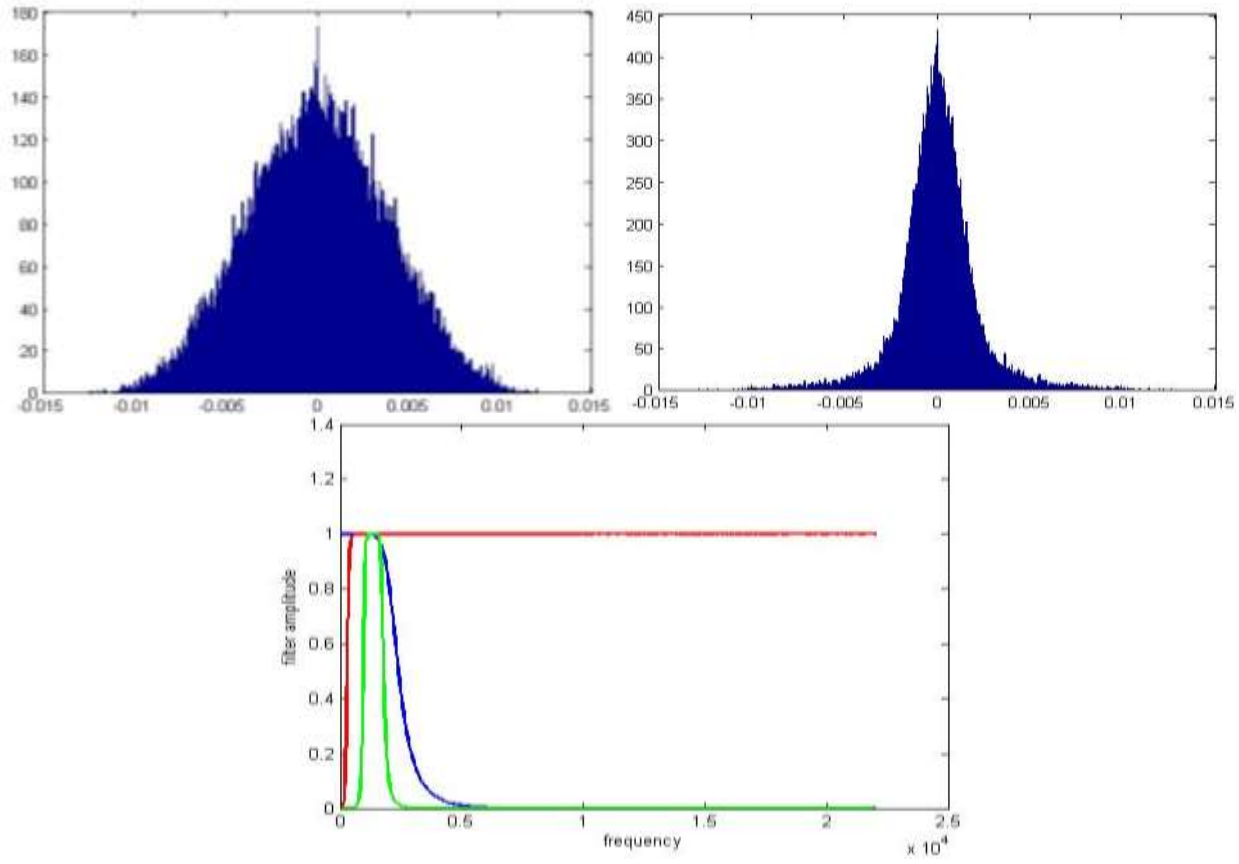
1) The features exhibit little variation over time. This is necessary for studies such as this one, that aims to compare different loons from different lakes over several years.
2) The classifier should be able to clearly distinguish between an existing set of features and an unknown set of features. This is important since loon populations are not closed; new loons arrive in a population by immigrations and births. Also a loon changes its vocalization with a change territory as detailed earlier. The classifier should be able to adequately handle such cases.
3) The features enable identification regardless of the call type produced. This is particularly important in the case of the loon, since it changes its call with a change in territory. Call-dependent identification schemes would fail in such cases. (this assumes that there are no significant physiological changes in the loon itself, that might cause a change in the vocal characteristics of the loon.

Although a direct physiological analogy cannot be drawn between a loon and a human, we can assume that MFCCs, which can suitably distinguish between two humans, can also distinguish between loons. One major advantage of current human speaker recognitions systems is its ability to identify the speaker regardless of the type of sound produced.

APPROACH

Pre-processing

MFCC values are not as robust in the presence of additive noise in the input sample. Since most of the recordings are obtained in the field and not in a closed environment, most samples have intervals of silence or some amount of noise. For MFCCs to function reliably well, it became necessary to pre-process all input samples to remove as much noise/silences as possible without a significant loss in the quality of the recording itself. High frequency (>2Khz) and Low frequency (<200Hz) is removed using a high pass and a low pass filter respectively. The song itself, which lies between 300Hz – 1.5Khz, was isolated using a band pass filter. There is also some noise in the same frequency range as the loon call. With some experimentation, we notice that the histogram of the amplitudes is considerably wider in the presence of noise, i.e. has a greater standard deviation than sections of the recording that have just the song (analogous to image noise). The song is divided into 50ms windows and based on a predefined threshold for standard deviation, a window is either maintained or deleted.



MFCC's

As detailed earlier, we assume that human speech recognition methods would work reasonably well in case of the loon. Hence the following section assumes the input voice signal coming from a human. Before identifying or training using an input sound sample that should be identified by the system, the signal must be processed to extract important characteristics of speech. In human speech, pitch frequency and formants are most important features of the speech signal. The pitch frequency corresponds to the fundamental frequency of vocal cord vibrations. Pitch is a characteristic of the source of excitation. Formants are the resonant frequencies of vocal tract and so they are characteristics of vocal tract.

Working on this model we can infer that a speech signal S(n) is a convolved combination of the excitation signal, e(n), and the impulse response of the vocal tract, θ(n). Only the signal S(n) is available to us, cepstral analysis is used to separate e(n) and θ(n). For feature extraction, we have to calculate the Cepstral coefficients in the mel frequency scale.

Cepstral Analysis

The main idea of Cepstral analysis is the separation of two convolved signals.

The output signal of speech production system S(n), is as follows:

$$s(n) = e(n) * \theta(n)$$

Applying the Fourier transform we get,

$$s(w) = E(w)\theta(w)$$

Taking logs, the following equation is obtained:

$$\log s(w) = \log E(w) + \log\theta(w)$$

$$cs(w) = ce(w) + c\theta(w)$$

Applying the direct cosine transform (DCT), the Cepstral coefficients are obtained.

$$cs(n) = ce(n) + c\theta(n)$$

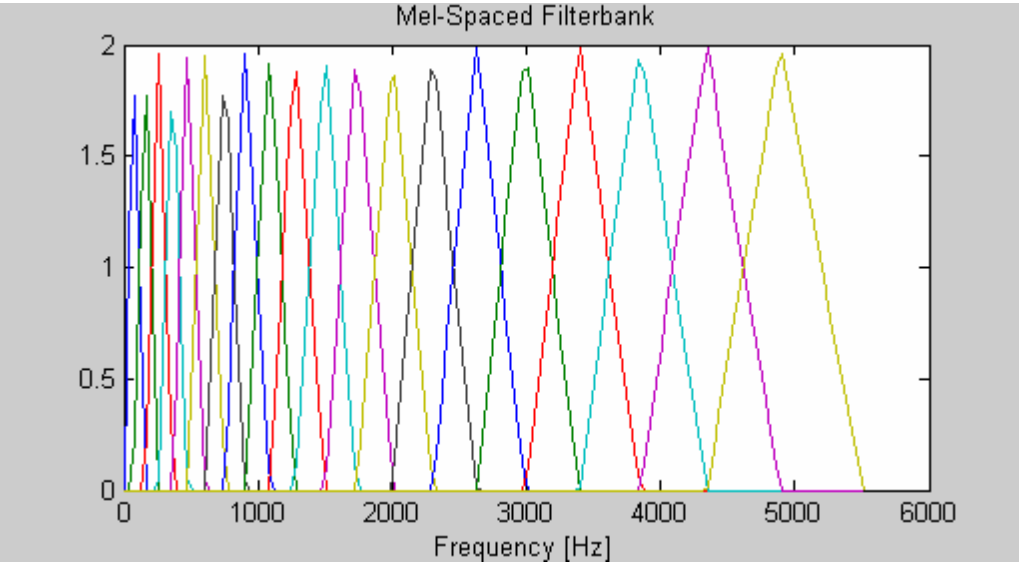
i.e. the Cepstral coefficients are obtained in the form of:

$$cs(n) = f - 1(\log[f(s(n))]$$

Mel-frequency Scaling

Phsyiological studies have proven that the human auditory system does not follow a linear scale, but a scale called the mel-scale. Thus for each tone with an actual frequency, f, measured in Hz, a subjective pitch is mapped on the mel-scale. The mel-frequency scale has linear spacing below 1000 Hz and log spacing above 1000 Hz. The mel-frequency scaling closely mimics the frequency response of the human auditory system and hence can be used to extract phonetically important characteristics of speech. An approach is to use mel-filter banks, each one of which has a triangular bandpass frequency response; the spacing as well as the bandwidth is determind the mel-frequency intervals. The relation between linear frequency and mel frequency is

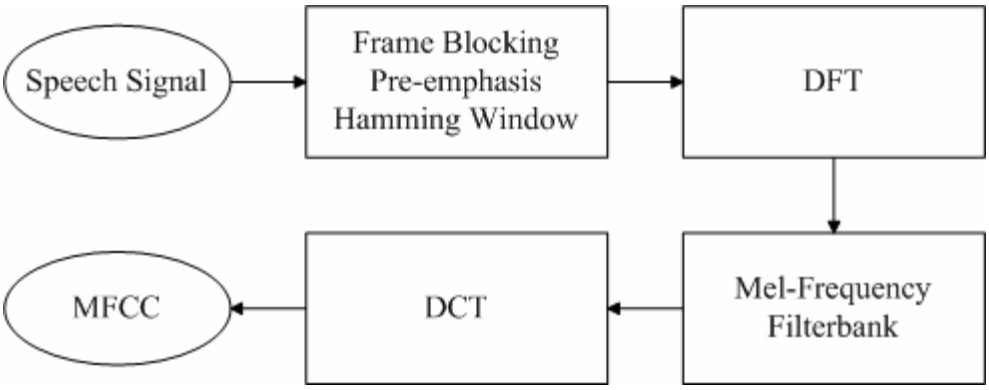
Mel(f)=2595* log 10 (1+ f / 700)



MFCC Computation

The MFCCs are computed as follows

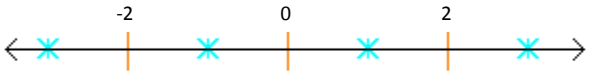
- Apply FFT on a windowed audio signal.
- Using overlapping triangular windows, map the resulting spectrum onto the mel scale.
- Take logs of the powers at each of the Mel frequencies.
- Treat the Mel log powers as a set of signals and take its direct cosine transform.
- The amplitudes of the resulting spectrum are the MFCCs



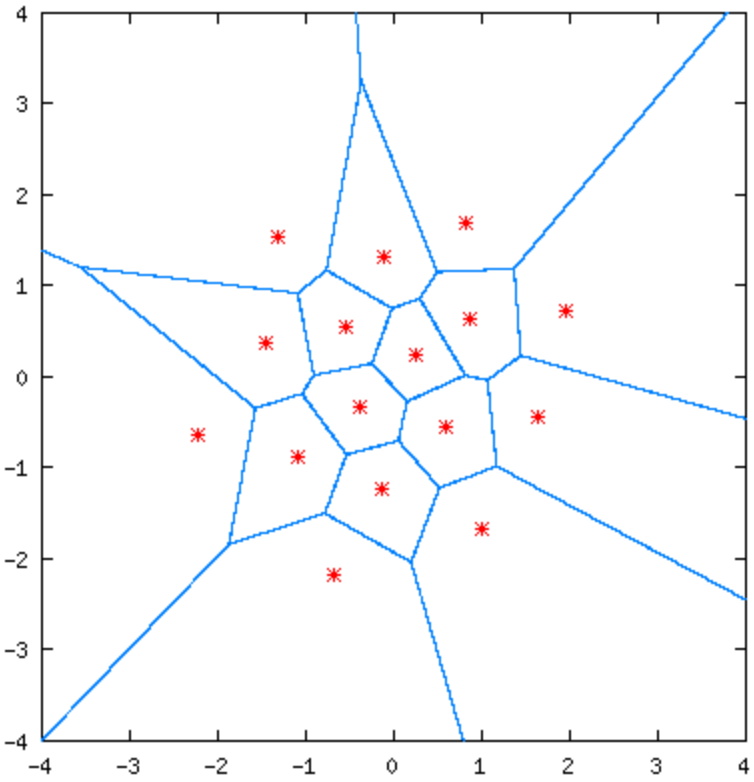
MFCCs provide a good representation of the local spectral properties of the signal. We obtain a significant improvement in performance using triangular filter banks. To improve efficiency further, further compression of data is required, we use Vector Quantization for this purpose.

Vector Quantization

One of the most popular speaker recognition techniques used nowadays is Vector Quantization. A Vector Quantizer (VQ) is essentially an approximator, somewhat similar in nature to "rounding off" to the nearest digit. A simple example could be a number line, the set of numbers between 0 and +2 is approximated by 1 (the centroid), the set of numbers between -2 and 0 is approximated by -1, every number greater than +2 is approximated by +3, every number lesser than -2 is approximated by -3 and so on.



This notion can be extended to 2 dimensions by using centroids for defined regions.



In the above example the red stars are the codevectors (or centroids) and the set of all the codevectors is known as a codebook. The complexity in the design of a VQ increases with the increase in the number of dimensions.

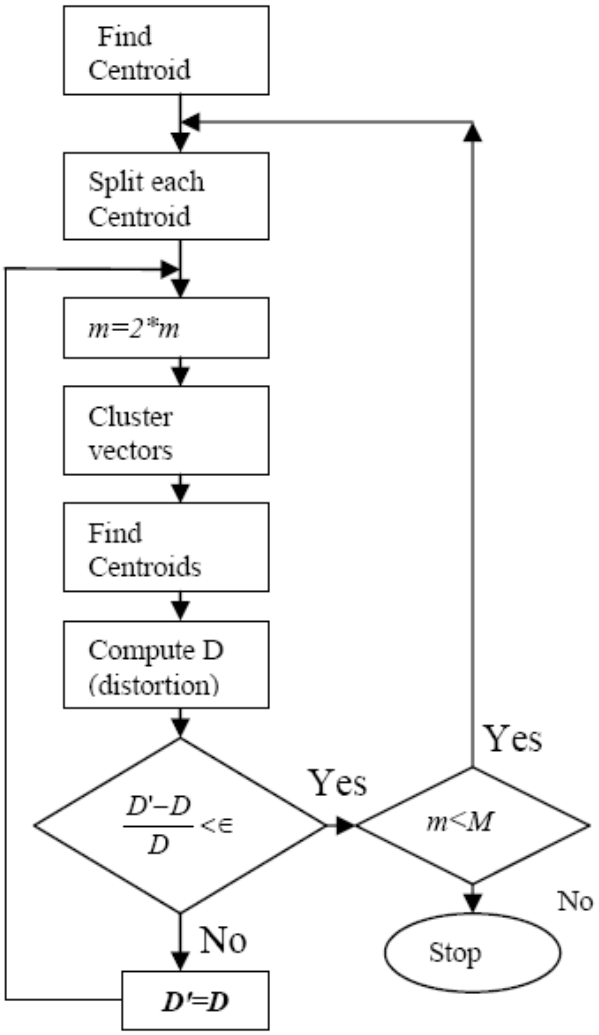
In 1980, Linde-Buzo-Gray proposed a VQ design algorithm, known as VQ-LBG, based on a training sequence. The initial codevector is obtained by finding the average of the whole cluster; two codevectors are obtained from the initial codevector by splitting the whole cluster into two regions. The iterative algorithm uses these two codevectors as the initial codevector to compute more codevectors, and till we get a codebook with the desired number of codevectors.

The algorithm is as follows:

1. Design a 1-vector codebook; this is the centroid of the entire set of training vectors (hence, no iteration is required here).

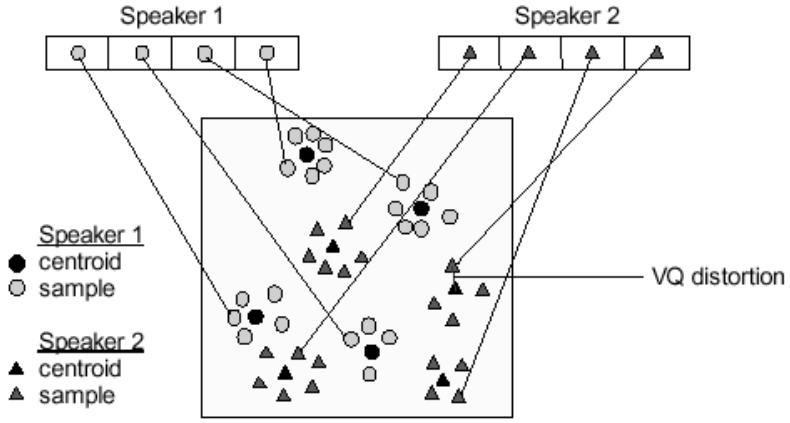
2. Double the size of the codebook by splitting each current codebook yn according to the rule: where n varies from 1 to the current size of the codebook, and e is the splitting parameter. For our system, e = 0.001.(1)

3. Nearest-Neighbor Search: for each training vector, find the centroid in the current codebook that is closest (in terms of similarity measurement), and assign that vector to the corresponding cell (associated with the closest centroid). This is done using the K-means iterative algorithm.
4. Centroid Update: update the centroid in each cell using the centroid of the training vectors assigned to that cell.
5. Iteration 1: repeat steps 3 and 4 until the average distance falls below a preset threshold
6. Iteration 2: repeat steps 2, 3, and 4 until a codebook of size M is reached.

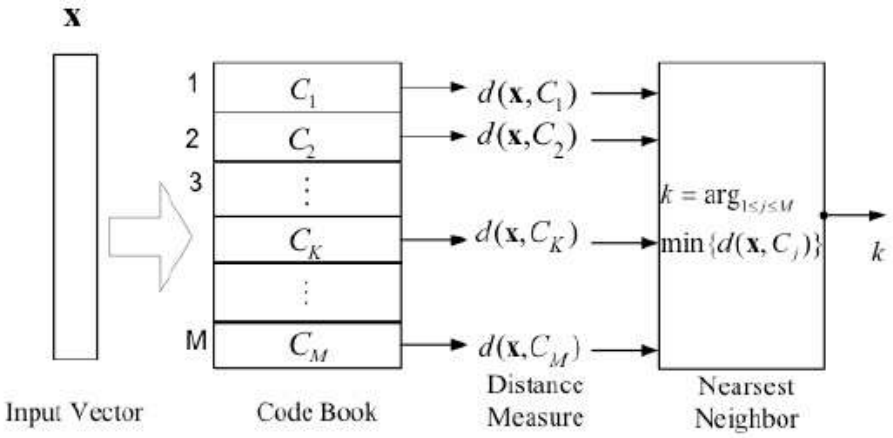


Classification

In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The resultant codewords (centroids) are shown in Figure 4 by circles and triangles at the centers of the corresponding blocks for speaker1 and 2, respectively. The distance from a vector to the closest codeword of a codebook is called a VQ distortion. In the recognition phase, an input utterance of an unknown voice is "vector-quantized" using each trained codebook and the *total VQ distortion* is computed. The speaker corresponding to the VQ codebook with the smallest total distortion is identified.



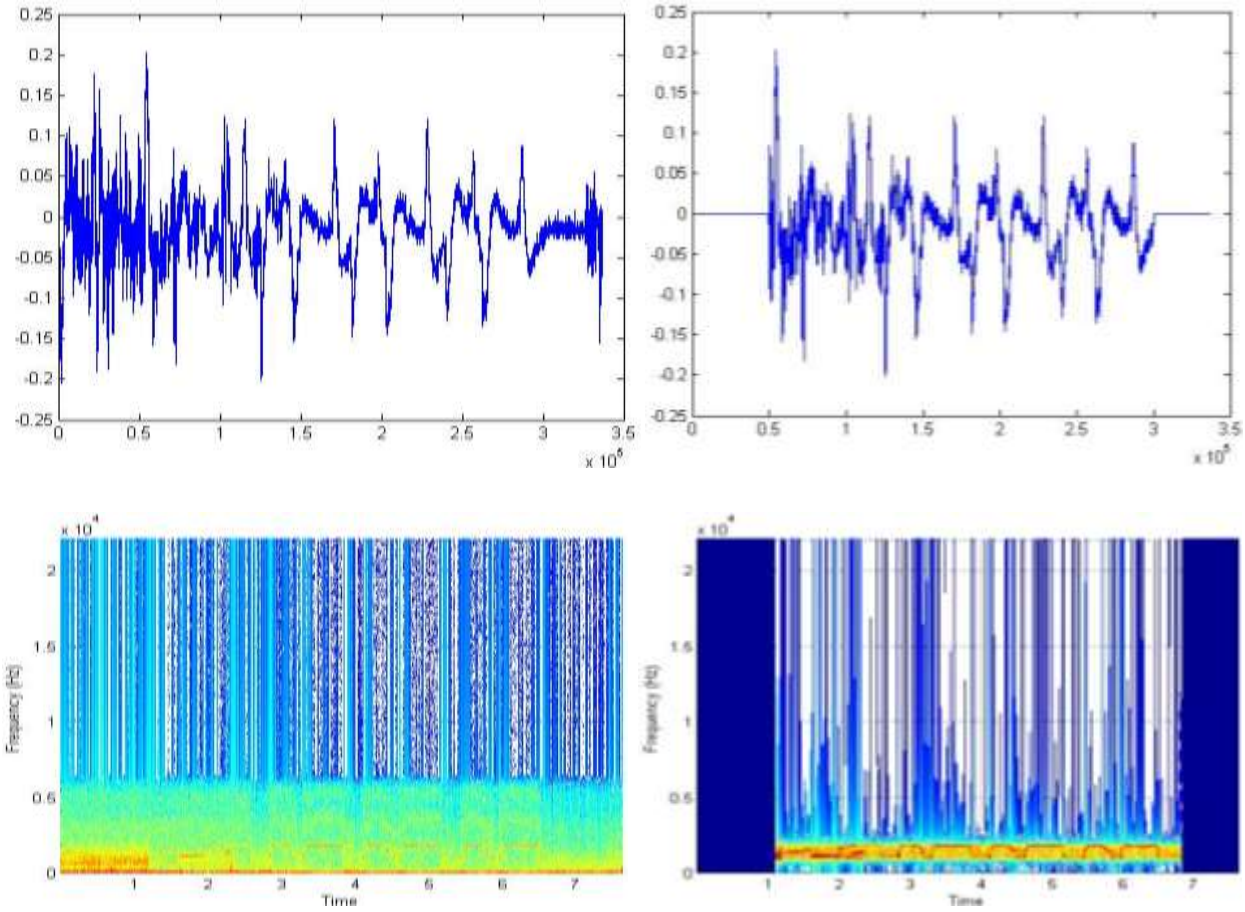
In the recognition phase the features of unknown command are extracted and represented by a sequence of feature vectors {x1… xn}. Each feature vector in the sequence X is compared with all the stored codewords in codebook, and the codeword with the minimum distance from the feature vectors is selected as proposed command For each codebook a distance measure is computed, and the command with the lowest distance is chosen. For this application we chose a codebook with 16 centroids.



RESULTS

Noise Removal

The noise removal succeeded on all preliminary tests, for more detailed results and a comparison without noise removal refer to the next section.



Speech Recognition

Several tests were done, to see if the recognition was consistent with the results published in [1]

Silence Removal testing A	[SR1]
Silence Removal testing B-1	[SR2]
Silence Removal testing B-2	[SR3]
Testing One Loon Against 2 Training Lakes from Same Year	[1L1YAB]
Testing One Loon Against 2+ Training Lakes from Same Year	[1L1YABC]
Testing One Loon Against 2 Training Lakes from Different Years	[1L2YAB]
Testing One Loon Against 2+ Training Lakes from Different Years	[1L2YABC]
Different Days	[DD]
Different Years	[DY]
Different Lakes	[DL]

Control
Purpose:
To see if the program is working, files of loons are matched to themselves. For example, Yodel A in the testing folder is matched to Yodel A in the training folder. For all of the tests here, the same loon, the same lake, and the same year and day are used.

Title: Control - 1

Lakes used:
Test: Shallow 2002
Train: Shallow 2002

Correct proportion:
9/10

Title:
Control - 2

Lakes used:
Test: Langley 2001
Train: Langley 2001

Correct proportion:
10/10

Title: Control – 3

Lakes used:
Test: Hodstradt 2008
Train: Hodstradt 2008

Correct proportion:
10/10

Conclusion:
The program works with a satisfying (29/30) accuracy.

==========

[SR1]
Purpose:
Testing the noise/silence removal, tests are done with manually edited(Audacity) compared with songs edited using the program.

Title: Silence Removal testing - 1

Lakes used:
Test: Langley 2001 – program-edited
Train: Langley 2001 – manually edited

Correct proportion:
5/5

Conclusion:
The silence removal works as expected.

==========

[SR2]
Purpose:
To see if silence removal plays a significant role in identification. Set A and B refer to two different sets of yodels from the same loon on that lake.
Test 2A is to see if they match up to the original song.
Test 2B is to see if both the edited song and the original song match up to another yodel by the same loon.
Test 2C is to see if both the edited song and the original song match up to the correct lake.

Title: Silence Removal testing – 2A

Lakes used:
Test: Set A from Langley 2001 – not edited
Train: Set A from Langley 2001 – program-edited

Correct proportion:
11/13

Title: Silence Removal testing – 2B

Lakes used:
Test: Set B from Langley 2001 – program-edited
Train: Set A from Langley 2001 – not edited
Set A from Langley 2001 – program-edited

Correct proportion:
12/13

Title:
Silence Removal testing – 2C

Lakes used:
Test: Set B from Langley 2001 – program-edited
Lumen 2001 – program-edited
Train: Set A from Langley 2001 – not edited
Set A from Langley 2001 – program-edited

Correct proportion:
13/13

Conclusion:
Although silence removal does not make a significant difference to the program, there are some particularly bad recordings where the silence removal becomes necessary. Also since most recordings have silences in the beginning and the end and are reasonably good quality, silence removal only becomes essential in samples that have silences in between the recordings and eliminate the process of manually removing silences.
==========
[1L1YAB]
Title:
Testing One Loon Against 2 Training Lakes from Same Year
Purpose:
To do a simple test first. A loon from one lake, A, tested using a codebook that contains yodels from loons from lake A (the same loon) and lake B (a different loon).

Title:
Test 1

Lakes Used:
Test: Langley 2001
Train: Langley 2001
Emma 2001

Correct proportion:
10/10

Title:
Test 2

Lakes Used:
Test: Currie 2006
Train: Currie 2006

Lumen 2006

Correct proportion:
6/6

Title:
Test 3

Lakes Used:
Test: Hodstradt 2008
Train: Hodstradt 2008
McGrath 2008

Correct proportion:
6/6

Title:
Test 4

Lakes Used:
Test: Hemlock 1997
Train: Hemlock 1997
Currie 1997

Correct proportion:
5/5

Title:
Test 5

Lakes Used:
Test: Currie 1997
Train: Hemlock 1997
Currie 1997

Correct proportion:
2/2

Conclusion:
Program has 100% accuracy.
==========
Title:
Testing One Loon Against 2+ Training Lakes from Same Year

Purpose:
To see if the program holds up when tested against more than one lake. All the loon yodels used in this test are from the same year.

Title:
Test 1

Lakes Used:
Test: Oscar Jenny 1999
Train: Oscar Jenny 1999
Dorothy 1999
Fawn 1999
Hancock 1999
Muskellunge 1999

Correct proportion:
4/12

Title:
Test 2

Lakes Used:
Test: Carrie 2005
Train: Carrie 2005
Hasbrook 2005
Horsehead 2005
Manson 2005
N. Nikomis

Correct proportion:
4/7

Title:
Test 3

Lakes Used:
Test: Gobler 2007
Train: Gobler 2007
Hanson 2007
Indian 2007
Carrol 2007
O'Day 2007

Correct proportion:
6/6

Title:
Test 4

Lakes Used:
Test: Gross 2004
Train: Gross 2004
    Heiress 2004
    Minocqua 2004
    Oneida 2004
    Swanson 2004

Correct proportion:
4/6

Title:
Test 5

Lakes Used:
Test: Wind Pudding 2004
Train: Wind Pudding 2004
    Townline 2004
    Manson 2004
    Currie 2004
    Hemlock 2004

Correct proportion:
7/10

Conclusion:
In this case the test and training samples were noisy and could not be fixed by the noise removal program. But the accuracy is still reasonable, about 85%
==========
[1L2YAB]
Title:
Testing One Loon Against 2 Training Lakes from Different Years
Purpose:
Simple test like the test between two lakes, but this time the two samples are taken from different years.

Title:
Test 1

Lakes used:
Test: Squash 2002
Train: Squash 2002
Virgin Lake 2005

Correct proportion:
7/7

Title:
Test 2

Lakes used:
Test: Spur 2009
Train: Spur 2009
Brown 2000

Correct proportion:
6/6

Title:
Test 3

Lakes used:
Test: Townline 2006
Train: Townline 2006
North Two 1998

Correct proportion:
6/6

Conclusion:
Again, the program achieves 100% accuracy.
==========
[1L2YABC]
Title:
Testing One Loon Against 2+ Training Lakes from Different Years
Purpose:
A more vigorous version of the previous test with only two lakes.

Title:
Test 1

Lakes used:
Test: Madeline 2004

Maud 2009
Train: Madeline 2004
Maud 2009
Perry 2005
Shallow 2007
Spider 2008
East Horsehead 2006

Correct proportion:
Madeline: 11/14
Maud: 27/27

Title:
Test 2

Lakes used:
Test: Big Bearskin 1997
Shallow 2008
Maud 2009
Train: Big Bearskin 1997
McGrath 2007
Maud 2009
Fox 2001
Shallow 2008

Correct proportion:
Big Bearskin: 7/7
Shallow: 7/7
Maud: 7/7

Title:
Test 3

Lakes used:
Test: Bass 2003
Bear 2008
Flannery 2000
Train: Bass 2003
Bear 2008
Flannery 2000

Correct proportion;
Bass: 5/5
Bear: 6/6
Flannery: 4/4

Conclusion:
The accuracy for this test is great (92/101). This shows that the program can indeed distinguish calls from a large codebook.
==========
[DD]
Title:
Different Days
Purpose:
During the testing, I noticed that the loons with yodels that were from one day tended to be matched with the yodels that were made from the same day, even if the training codebook included yodels from the same year. This test was to see if this was just a fluke.

Title:
Test 1

Lakes used:
Test: Langley 2001
Train: Langley 2001

Correct proportion:
13/13

Title:
Test 2

Lakes used:
Test: Lumen 2001
Train: Lumen 2001

Correct proportion:
13/13

Title:
Test 3

Lakes used:
Test: Pickerel West 2004
Train: Pickerel West 2004

Correct proportion:
7/8

Conclusions:
This is very good accuracy (35/36). More data should probably be taken in the future to see if the pattern holds up. This may suggest that the yodels given by the loons also tell some information that vary with the day.
==========
[DL]
Title:
Different Lakes
Purpose:
This is the main purpose of this analysis, to see if the program functions when tested for yodels that were made before a loon changed lakes and after.

Title:
Test 1

Course of the loon:
South Blue 1999-2002
Bearskin 2003
South Blue 2003

Lakes used:
Test: Bearskin 2003
South Blue 1999
South Blue 2000
Train: East Horsehead 2006
Hodstradt 2007
Madeline 2004
Maud 2009
Shallow 2007
South Blue 2003
South Blue 2002
South Blue 2001
Spider 2008

Correct proportion:
6/10

Title:
Test 2

Course of the loon:
Manson 2006-2007
McGrath 2008
Manson 2008-2009

Lakes used:
Test: Manson 2008
Manson 2009
McGrath 2008
Train: East Horsehead 2006
Hodstradt 2007
Madeline 2004
Maud 2009
Manson 2006
Manson 2007
Shallow 2007
Spider 2008

Correct proportions:
21/29

Conclusion: One lake mismatched every time, McGrath 2008. A test was done again after removing the most commonly lake it was matched with – Hodstradt 2007 – and the results were much more accurate (9/10).

==========

Conclusions/Future Work

Overall the program succeeds at matching the loons correctly, but depends on the choosing the right training set. Some improvements could include using independent component analysis to considerably reduce the influence of noise, One could also use a different scheme for identification, ex: Neural Net/HMM

References

Appleby, R.H.; Steve C. Madge & Mullarney, Killian (1986): Identification of divers in immature and winter plumages. *British Birds* **79**(8): 365-391.

Walcott, C., Mager, J. N. & Walter, P. 2006. Changing territories, changing tunes: male loons, *Gavia immer*, change their vocalizations when they change territories. *Animal Behaviour*, 71, 673-683.

http://en.wikipedia.org/wiki/Loon

T. Linde, A. Buzo, and R. M. Gray. An algorithm for vector quantization. *IEEE Trans. Communications*,28(1):84–95, 1980

Fox, Elizabeth J. *Call-independent Identification in Birds*. Thesis. University of Western Australia, 2008

Deller J.R. Hansen, J.H.L &.Proakis J.G., (1993), Discrete-Time Processing of Speech Signal, New York, Macmillan Publishing Company .

http://www.learner.org/jnorth/tm/loon/Dictionary.html

Appendix A: Code

```
function code = train(traindir)
% ================================================================================
% Input:
% traindir : string name of directory containing all voice sound files
% ================================================================================
% Output:
% code : trained VQ codebooks, code{i} for i-th speaker
% ================================================================================
% Example:
% >> code = build('C:\...\samples\');
% ================================================================================
k = 16; % number of centroids required
D = dir([traindir '//*.wav']);

for i = 1:size(D,1) % train a VQ codebook for each speaker
    x = [traindir D(i).name];
    disp(x);
    [s, fs] = wavread(x);
    Vnorm = melcepst(s, fs);
    v = Vnorm';
    code{i} = vqlbg(v, k); % Train VQ codebook
end
```

```
function c=melcepst(s,fs,w,nc,p,n,inc,fl,fh)
%MELCEPST Calculate the mel cepstrum of a signal C=(S,FS,W,NC,P,N,INC,FL,FH)
%Simple use: c = melcepst(s,fs)
%            calculate mel cepstrum with 12 coefs, 256 sample frames
%            c = melcepst(s,fs,'e0dD')
%            include log energy, 0th cepstral coef, delta and delta-delta
%            coefs
% ================================================================================
% Inputs:
% s   speech signal
% fs  sample rate in Hz (default 11025)
% nc  number of cepstral coefficients excluding 0'th coefficient (default 12)
% n   length of frame (default power of 2 <30 ms))
% p   number of filters in filterbank (default floor(3*log(fs)) )
% inc frame increment (default n/2)
% fl  low end of the lowest filter as a fraction of fs (default = 0)
% fh  high end of highest filter as a fraction of fs (default = 0.5)
% w   any sensible combination of the following:
% 'R' rectangular window in time domain
% 'N' Hamming window in time domain
% 'M' Hamming window in time domain (default)
% 't' triangular shaped filters in mel domain (default)
% 'n' hanning shaped filters in mel domain
% 'm' hamming shaped filters in mel domain
% 'p' filters act in the power domain
% 'a' filters act in the absolute magnitude domain (default)
% '0' include 0'th order cepstral coefficient
% 'e' include log energy
% 'd' include delta coefficients (dc/dt)
% 'D' include delta-delta coefficients (d^2c/dt^2)
```

```
% 'z' highest and lowest filters taper down to zero (default)
% 'y' lowest filter remains at 1 down to 0 frequency and highest filter
%     remains at 1 up to nyquist freqency
% If 'ty' or 'ny' is specified, the total power in the fft is preserved.
% ========================================================================
% Outputs:
% c   mel cepstrum output: one frame per row
% ========================================================================
%
% Copyright (C) Mike Brookes 1997
% Last modified Thu Jun 15 09:14:48 2000
% VOICEBOX is a MATLAB toolbox for speech processing.
% FOR THE REST OF THE FILES IN VOICEBOX GO TO:
% http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
% ========================================================================
% This program is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either version 2 of the License, or
% (at your option) any later version.
%
% This program is distributed in the hope that it will be useful,
% but WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
% GNU General Public License for more details.
%
% You can obtain a copy of the GNU General Public License from
% ftp://prep.ai.mit.edu/pub/gnu/COPYING-2.0 or by writing to
% Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
% ========================================================================
    if nargin<2 fs=11025; end
    if nargin<3 w='M'; end
    if nargin<4 nc=20; end
    if nargin<5 p=floor(3*log(fs)); end
    if nargin<6 n=pow2(floor(log2(0.03*fs))); end
    if nargin<9
        fh=0.5;
        if nargin<8
            fl=0;
            if nargin<7
                inc=floor(n/2);
            end
        end
    end

    if any(w=='R')
        z=enframe(s,n,inc);
    elseif any (w=='N')
        z=enframe(s,hanning(n),inc);
    else
        z=enframe(s,hamming(n),inc);
    end
    f=rfft(z.');
    [m,a,b]=melbankm(p,n,fs,fl,fh,w);
    pw=f(a:b,:).*conj(f(a:b,:));
    pth=max(pw(:))*1E-6;
    if any(w=='p')
        y=log(max(m*pw,pth));
```

```
else
    ath=sqrt(pth);
    y=log(max(m*abs(f(a:b,:)),ath));
end
c=rdct(y).';
nf=size(c,1);
nc=nc+1;
if p>nc
    c(:,nc+1:end)=[];
elseif p<nc
    c=[c zeros(nf,nc-p)];
end
if ~any(w=='0')
    c(:,1)=[];
end
if any(w=='e')
    c=[log(sum(pw)).' c];
end

% calculate derivative
if any(w=='D')
    vf=(4:-1:-4)/60;
    af=(1:-1:-1)/2;
    ww=ones(5,1);
    cx=[c(ww,:); c; c(nf*ww,:)];
    vx=reshape(filter(vf,1,cx(:)),nf+10,nc);
    vx(1:8,:)=[];
    ax=reshape(filter(af,1,vx(:)),nf+2,nc);
    ax(1:2,:)=[];
    vx([1 nf+2],:)=[];
    if any(w=='d')
        c=[c vx ax];
    else
        c=[c ax];
    end
elseif any(w=='d')
    vf=(4:-1:-4)/60;
    ww=ones(4,1);
    cx=[c(ww,:); c; c(nf*ww,:)];
    vx=reshape(filter(vf,1,cx(:)),nf+8,nc);
    vx(1:8,:)=[];
    c=[c vx];
end

if nargout<1
    [nf,nc]=size(c);
    t=((0:nf-1)*inc+(n-1)/2)/fs;
    ci=(1:nc)-any(w=='0')-any(w=='e');
    imh = imagesc(t,ci,c.');
    axis('xy');
    xlabel('Time (s)');
    ylabel('Mel-cepstrum coefficient');
    map = (0:63)'/63;
    colormap([map map map]);
    colorbar;
end
```

```
function r = vqlbg(d,k)
% VQLBG Vector quantization using the Linde-Buzo-Gray algorithm
% ================================================================================
% Inputs: d contains training data vectors (one per column)
% k is number of centroids required
% ================================================================================
% Output: r contains the result VQ codebook
%         (k columns, one for each centroids)
% ================================================================================
% Reference:
% http://www.mathworks.com/matlabcentral/fileexchange/?term=tag%3A%22vqlbg%22
% ================================================================================
    e = .01;
    r = mean(d, 2);
    dpr = 10000;
    for i = 1:log2(k)
        r = [r*(1+e), r*(1-e)];
        while (1 == 1)
            z = disteu(d, r);
            [m,ind] = min(z, [], 2);
            t = 0;
            for j = 1:2^i
                r(:, j) = mean(d(:, find(ind == j)), 2);
                x = disteu(d(:, find(ind == j)), r(:, j));
                for q = 1:length(x)
                    t = t + x(q);
                end
            end
            if (((dpr - t)/t) < e)
                break;
            else
                dpr = t;
            end
        end
    end
```

```
function test(traindir,testdir,code)
% ================================================================================
% Input:
% testdir : string name of directory contains all train sound files
% testdir : string name of directory contains all test sound files
% code : codebooks of all build speakers
% ================================================================================
% Example:
% >> test('C:\...\train\','C:\...\test\', code);
% ================================================================================

Dtest = dir([testdir '//*.wav']);
Dtrain = dir([traindir '//*.wav']);
for k = 1:size(Dtest,1) % read test sound file of each speaker
    xtest = [testdir Dtest(k).name];
    xtrain = [traindir Dtrain(k).name];
    [s, fs] = wavread(xtest);
    Vnorm = melcepst(s, fs); % Compute MFCC's
```

```
    v = Vnorm';
    distmin = inf;
    k1 = 0;
    for l = 1:length(code) % each trained codebook, compute distortion
        d = disteu(v, code{l});
        dist = sum(min(d,[],2)) / size(d,1)
        if dist < distmin
            distmin = dist;
            k1 = l;
        end
    end

    msg = sprintf('%s matches with %s', [Dtest(k).name], [Dtrain(k1).name]);
    disp(msg);
end
```

```
function d = disteu(x, y)
% Calculates Euclidean distances
% ============================================================================
% Input:
% x, y: Two matrices whose columns are vector data.
% ============================================================================
% Output:
% d: Element d(i,j) Euclidean distances between two
% column vectors X(:,i) and Y(:,j)
% ============================================================================
% Note:
% The Euclidean distance is guven by:
% d = sum((x-y).^2).^0.5
% ============================================================================
    [M, N] = size(x);
    [M2, P] = size(y);
    if (M ~= M2)
        error('Matrix dimensions do not match.')
    end
    d = zeros(N, P);
    if (N < P)
        copies = zeros(1,P);
        for n = 1:N
            d(n,:) = sum((x(:, n+copies) - y) .^2, 1);
        end
    else
        copies = zeros(1,N);
        for p = 1:P
            d(:,p) = sum((x - y(:, p+copies)) .^2, 1)';
        end
    end
    d = d.^0.5;
```

```
function songOut = silenceRemover(songIn)
% Removes silences and noise from an audio sample
% ========================================================================
% Input:
% songIn: Unedited audio recording, Windows Waveform Audio(.wav) format
% Mono, (Does not handle Stereo)
% For converting a song from stereo to mono, or to .wav format:
% http://audacity.sourceforge.net/
% ========================================================================
% Output:
% songOut: Edited sample
% ========================================================================
% Example:
% >> x = silenceRemover('C:\.....\loon1.wav');
% ========================================================================
    %The scale has to be adjusted so that filter coefficients are
    % -2.0<coeff<2.0
    %Scaling performed is 2^(-scale)
    scale = 5;
    %For lowpass, set equal to normalized Freq (cutoff/(Fs/2))
    %For bandpass, set equal to normalized Freq vector ([low high]/(Fs/2))
    freq1 = 0.097;
    freq2 = 0.015;
    freq3 = [0.045 0.078];
    order = 5;
    [b1, a1] = butter(order, freq1,'low');
    [b2, a2] = butter(order, freq2, 'high');
    [b3, a3] = butter(order, freq3);
    bs1 = b1 * (2^-scale);
    as1 = -a1 * (2^-scale);
    bs2 = b2 * (2^-scale);
    as2 = -a2 * (2^-scale);
    bs3 = b3 * (2^-scale);
    as3 = -a3 * (2^-scale);
    % Fs = 44100;
    % [fresponse1, ffreq1] = freqz(b1,a1,500);
    % [fresponse2, ffreq2] = freqz(b2,a2,500);
    % [fresponse3, ffreq3] = freqz(b3,a3,500);
    % plot(ffreq1/pi*Fs/2,abs(fresponse1), 'b', 'linewidth',2);
    % xlabel('frequency'); ylabel('filter amplitude');
    % hold on
    % plot(ffreq2/pi*Fs/2,abs(fresponse2), 'r', 'linewidth',2);
    % xlabel('frequency'); ylabel('filter amplitude');
    % plot(ffreq3/pi*Fs/2,abs(fresponse3), 'g', 'linewidth',2);
    % xlabel('frequency'); ylabel('filter amplitude');
    % hold off
    y = wavread(songIn);
    yy = filter(bs1,as1,y);
    yy1 = filter(bs2,as2,yy);
    yy2 = filter(bs3,as3,yy1);
    %figure, plot(y)
    yy3 = yy2;
    step1 = 1;
    stepCount1 = floor(size(yy2,1)/50000);
    finalStep1 = mod(size(yy2,1),50000);
```

```
    for i1 = 1:stepCount1
        finalComp = abs((sum(yy2(step1:i1*50000))));
        if(finalComp < 0.005)
            yy3(step1:i1*50000) = 0;
        else
            yy3(step1:i1*50000) = 1;
        end
        step1 = step1 + 50000;
    end
    finalComp1 =
abs((sum(yy2(stepCount1*50000:stepCount1*50000+finalStep1))));
    if (finalComp1 < 0.005)
        yy3(stepCount1*50000:stepCount1*50000+finalStep1) = 0;
    end
    yy4 = y.*yy3;
    %figure, plot(yy4);
    nonZeroCount = 0;
    newCount = 1;
    for copyCount = 1:size(yy4,1)
        if (yy4(copyCount) ~= 0)
            nonZeroCount = nonZeroCount + 1;
        end
    end
    yy5 = zeros(nonZeroCount,1);
    for copyCount = 1:size(yy4,1)
        if (yy4(copyCount) ~= 0)
            yy5(newCount) = yy4(copyCount);
            newCount = newCount + 1;
        end
    end
    %figure, plot(yy5);
    songOut = yy5;
```

```
function batchSilenceRemover(inputFolderTarget)
% Runs silence remover for all samples in a folder
% ========================================================================
% Input:
% Folder containing all samples that have to be edited
% ========================================================================
% Output:
% Edited samples are stored in the input folder with the prefix 'edit_'
% ========================================================================
% Example:
% >> batchSilenceRemover('C:\....\loonFolder\');
% ========================================================================
targetDir = dir([inputFolderTarget '//*.wav']);
Fs = 44100;
nbits = 32;
for i = 1:size(targetDir,1)
    songIn = [inputFolderTarget '\' targetDir(i).name]
    songOut = silenceRemover(songIn);
    wavwrite(songOut,Fs,nbits,[inputFolderTarget '\edit_' targetDir(i).name])
end
```