World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:12, 2007

The Design and Implementation of Classifying Bird Sounds

Haiyi Zhang, Jianli Guo, and Daqian Yang

***Abstract*—**This Classifying Bird Sounds (chip notes) project's purpose is to reduce the unwanted noise from recorded bird sound chip notes, design a scheme to detect differences and similarities between recorded chip notes, and classify bird sound chip notes. The technologies of determining the similarities of sound waves have been used in communication, sound engineering and wireless sound applications for many years. Our research is focused on the similarity of chip notes, which are the sounds from different birds. The program we use is generated by Microsoft C++.

***Keywords*—**Classify Bird Sounds, Noise Filter, High-pass, Low-pass, Band-pass, Band-stop Filter, FIR.

I. Introduction

MILLIONS of songbirds migrate through southern Nova Scotia of Canada each fall. Most fly at night at altitudes between 200 and 600 m. While flying, they frequently make short duration sounds (probably for communication with other birds) that are called chip notes. These chip notes differ among species. Our research attempts to identify the bird based on its chip notes. When the researchers record the chip notes of songbirds flying overhead, it is field work, so there are often unwanted noises on the recording, such as wind noise, dog yelp noise, and mechanical noise from the human environment. In this project, we included a Noise Filter to reduce the unwanted noises from the recording. Noise reduction is the first part of the specification for a program to classify bird sounds (chip notes). The noise filter scheme usually used to reduce the unwanted noise from a standard input audio signal recording (WAV file) incorporates High-pass, Low-pass, Band-pass, Band-stop Filter and FIR (Finite Impulse Response) technology. This Noise Filter will input a WAV format bird sound then, according to parameters chosen by the user, reduce the unwanted noise from the recording of chip notes and output a new, noise-reduced version WAV format bird sound.

The program is written in the C++ language, written and tested on Microsoft Visual C++ 6.0. We constructed a graphical user interface (GUI) window, which is easy for users to operate. After the programming was done, we used Microsoft Visual C++ 6.0 to build a Filter Microsoft Application, Filter.exe file, so we can simply run this noise filter in the Windows operating system without using Microsoft Visual C++ 6.0

The second part of this project is to determine the similarities of the chip notes. Birds make a variety of sounds; in addition to singing, they utter short, sharp notes or "chips." When alarmed, they produce a very different call. Songs are easier to identify than chips or call notes, which can take a long time for a human to learn how to distinguish one from another. Familiarizing ourselves with the songs of at least the common species will make identification that much easier. Then, if we can distinguish among the chip notes, we will be able to identify the birds that voiced them. Scientists have collected thousands of known chip notes, so if we can compare our recordings of unidentified chip notes with those known chip notes, we will be able to identify the various types of birds. [10]

With regard to the signals and systems we employed, relativity is the most important way to find the similarities. In the paper, we will use the knowledge of calculus and linear algebra to compare two waves.

H. Zhang is with the Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, B4P 2R6, Canada (e-mail: Haiyi.Zhang@acadiau.ca).

J. Guo is with the Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, B4P 2R6, Canada (e-mail: 049945g@acadiau.ca).

D. Yang is with the Jodrey School of Computer Science, Acadia University, Wolfville, Nova Scotia, B4P 2R6, Canada (e-mail: 052203y@acadiau.ca).

II. The Design of the Noise Filter Architecture

The Architecture of the Noise Filter (basic flow) is shown as follows:1.Open WAV sound file; 2.Set parameters for the Noise Filter; 3.Reduce the unwanted noise; 4. Save the changes in a new WAV file.

A. WAV File Format

The WAV file format is a subset of Microsoft's RIFF specification for the storage of multimedia files. A RIFF file starts out with a file header followed by a sequence of data chunks, as shown in Figure 1.

| RIFF Chunk | FMT Chunk | DATA Chunk |

Fig 1: WAV File Format

A WAV file is often just a RIFF file with a single "wave" chunk, which consists of two sub-chunks -- an "fmt" chunk specifying the data format and a "data" chunk containing the actual sample data. Call this form the "Canonical form," as shown in Fig 2 [6]:

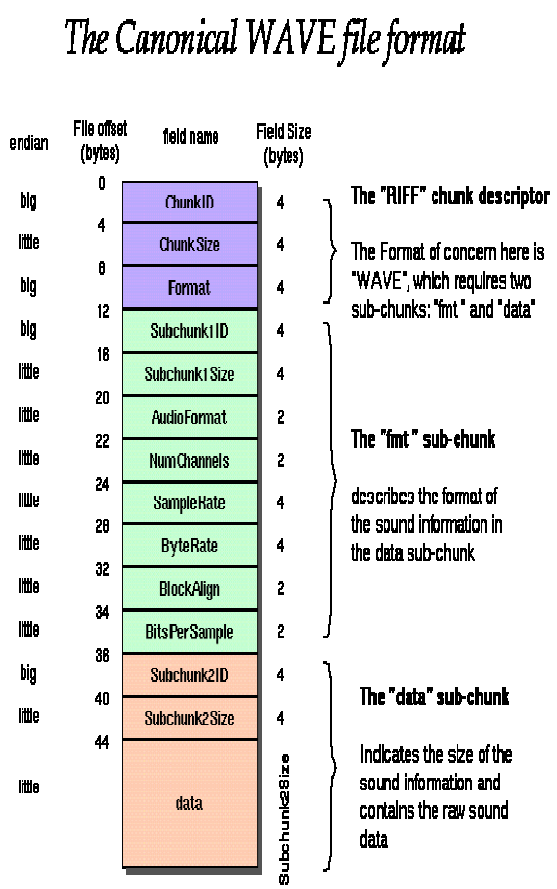


Fig. 2 Canonical WAV File Format

B. Operate WAV Format Files

First, we need to operate the WAV format files. We open the WAV file to get the sound data information according to our purpose, and use related noise reduction algorithms to modify those sound data so we can then save those modified sound data to a new WAV file.

For programming, we can use **CFILE** to open and close the WAV file, or we can use multimedia process functions which come from Microsoft Windows. Those functions all begin with ***mmio***. The basic flow is as follows:

A. Use ***mmioOpen*** function to open a WAV file, and get the ***mmioHandle*** function to handle the file;

*B. According to the data structure of the WAV file format, use **mmioRead** **mmioWrite** and **mmioSeek** functions to implement read, write and seek operations;*

C. Use the ***mmioClose*** function to close a WAV file.

C. FIR Filter

The objective of this filter is to provide digital filtering functions to the system. FIR (Finite Impulse Response) is one of the main types of digital filters. Figure 3 is the sample FIR Filter Window: Dialog.

The upper left panel is called the **Log-Magnitude Spectrum,** which can represent the bird sound frequency curve.

Below the **Log-Magnitude Spectrum**, there are two **scroll bars**, which respond to the cutoff frequencies. The **upper scroll bar** is for setting the lower cutoff frequency, and the **lower scroll bar** is for setting the higher cutoff frequency. The default values of the lower and higher cutoff frequencies are 400 and 2400 Hz, respectively. The user can specify the lower and higher cutoff frequencies by pulling the scroll bars left or right. When the lower and higher cutoff frequency scroll bars are adjusted to the left or right, the lower and higher **Cutoff Frequency panel** will be automatically changed. [1,3,4]

The **Sampling panel** represents the sampling rate. A sampling rate of 22050 samples/s (Hz) is assumed, corresponding to a maximum signal frequency of 11025 Hz.

The **Filter Order panel** represents the filter order. A filter order of 40 is assumed.

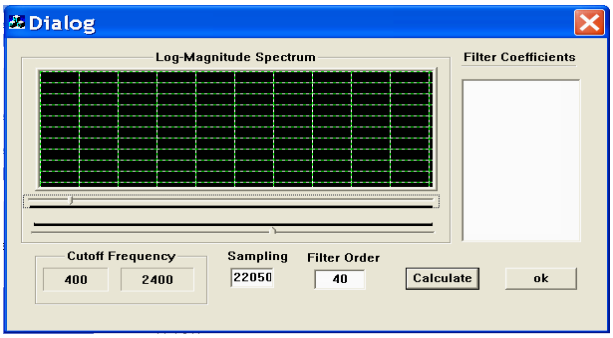


Fig. 3 FIR Filter Window: Dialog

After the lower and higher cutoff frequency, the sampling rate and the filter order have been set, the filter coefficient values can be taken with the "Calculate" button. By clicking the "Calculate" button, the filter coefficient will be displayed in the **Filter Coefficient** list box.

After filter coefficients have been calculated, the user can save the filter coefficients by clicking the "OK" button.

Low-pass and High-pass Filter Configurations

In low-pass and high-pass configurations, the filter order and cutoff frequency parameters specify the filter design. Frequencies are normalized to half the sample frequency. Figure 4 below shows the frequency response of the default order-22 filter with cutoff at 0.4. [8]

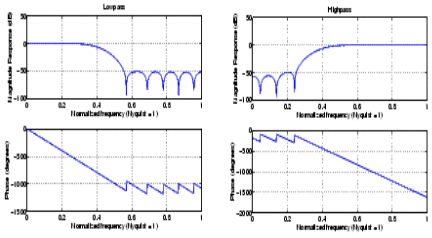


Fig 4 Low-pass and High-pass Configurations

Band-pass and Band-stop Filter Configurations

In band-pass and band-stop configurations, the filter order, lower cutoff frequency and upper cutoff frequency parameters specify the filter design. Frequencies are normalized to half of the sample frequency, and the actual filter order is twice the filter order parameter value. Fig 5 below shows the frequency response of the default order-22 filter with the lower cutoff at 0.4, and upper cutoff at 0.6. [8]

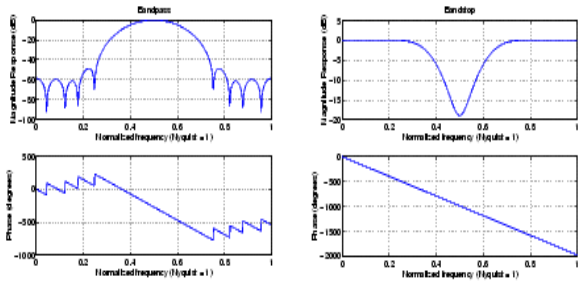


Fig. 5 Band-pass and Band-stop Configurations

III. The Principle of the Noise Filter

Our Noise Filter is built based on Microsoft Foundation Class Library: Filter. By using High-pass, Low-pass, Band-pass, Band-stop Filter and FIR (Finite Impulse Response) technology, our noise reduction feature can reduce background and general broad band noise with minimal reduction in signal quality.

The following documentation comes from Microsoft Foundation Class Library: Filter and that documentation states the purpose of each file of Microsoft Foundation Class Library: Filter.

AppWizard has created this Filter application for you. This application not only demonstrates the basics of using the Microsoft Foundation classes but is also a starting point for writing your application.

This file contains a summary of what you will find in each of the files that make up your Filter application.

These files are used to build a precompiled header (PCH) file named Filter.pch and a precompiled type's file named StdAfx.obj.

Our Noise Filter is based on the technology of High-pass, Low-pass, Band-pass, Band-stop Filter and FIR (Finite Impulse Response). If the noise and signal are separated in the temporal frequency domain, then band-pass filtering can be applied to the traces to remove such noise. A band-pass attenuates very low and very high frequencies, but retains a middle range band of frequencies.

Strong noise might be low-frequency surface waves, 60 Hz electrical noise, wind noise, mechanical noise from the human environment, cows or sharks munching on cables, etc.

Wave filter component values are chosen to efficiently transmit current from input to output in a desired band of frequencies while more or less completely suppressing transmission of current at all other frequencies. Boundary frequencies between the transmission and suppression frequency bands are called cutoff frequencies. [7]

A. Some Definitions [8, 9]

Filter type

Filter type means the type of filter used in the design. In this project, a Low-pass Filter, High-pass Filter, Band-pass Filter, and a Band-stop Filter was used. [8]

Filter order

Filter order means the order of the Filter, usually between 20 and 60. The filter length is one more than this value. For the Band-pass and Band-stop configurations, the order of the final filter is twice this value. The filter order primarily determines the width of the *transition band*: the higher the order, the narrower the transition between the pass-band and stop-band, giving a sharper cutoff in the frequency response. Filter order describes the attenuation rate. Each degree in the order provides a 6 dB/octave (or 20 dB/decade) attenuation and a 180 degree phase shift. [8]

Cutoff frequency

Cutoff frequency means the normalized cutoff frequency for the High-pass and Low-pass filter configurations. A value of 1 specifies half the sample frequency.

Lower cutoff frequency

Lower Cutoff frequency means the lower pass-band or stop-band frequency for the Band-pass and Band-stop filter configurations. A value of 1 specifies half the sample frequency.

Upper cutoff frequency

Upper Cutoff frequency means the upper pass-band or stop-band frequency for the Band-pass and Band-stop filter configurations. A value of 1 specifies half the sample frequency.

Sampling Rate

Sampling (taking the value of a signal at evenly spaced moments in time) is one of the steps in converting an analog signal to a digital signal. The sampling rate is the frequency at which samples are taken. The sampling rate has to be twice or more the rate of the analog frequency that is captured and, the higher the sampling rate, the better quality signal.

Filter Coefficient

Filter coefficient is the set of constants, also called *tap weights*, used to multiply against delayed signal sample values within a digital filter structure. Digital filter design is an exercise in determining the filter coefficients that will yield the desired filter frequency response. For an FIR filter, the filter coefficients are, by definition, the impulse response of the filter.

B. Low-pass Filter

According to the cutoff frequencies, a low-pass filter removes unwanted noise in the high frequencies. It attenuates high frequencies and retains low frequencies unchanged. In this case, high frequencies mean the bird sound frequency is higher than the cutoff frequency.

C. High-pass Filter

According to the cutoff frequencies, a high-pass filter removes unwanted noise in the low frequencies. It attenuates low frequencies and retains high frequencies unchanged. In this case, low frequencies mean the bird sound frequency is lower than the cutoff frequency.

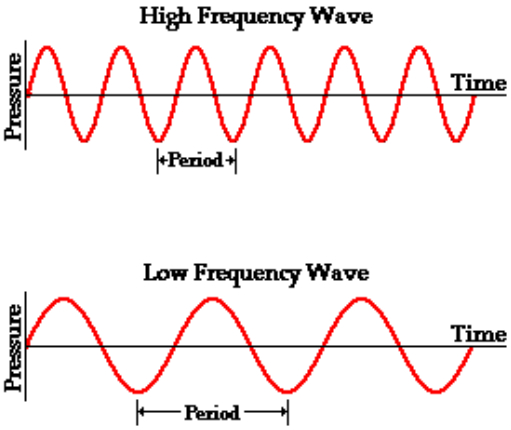


Fig. 6 High and low frequency wave

D. Band-pass Filter

The Band-pass filter removes unwanted noise in a fixed frequency area. It attenuates very low and very high frequencies, but retains a middle-range band of frequencies.

Band-pass filters are a combination of both low-pass and high-pass filters. They attenuate all frequencies smaller than a frequency D0 and higher than a frequency D1, while the frequencies between the two cut-offs remain in the resulting output sound. We obtain the filter function of a band-pass by multiplying the filter functions of a low-pass and of a high-pass in the frequency domain, where the cut-off frequency of the low-pass is higher than that of the high-pass.

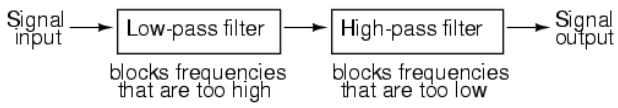


Fig. 7 Bandpass Filter Principle

E. Band-stop Filter

A band-stop filter is the inverse of a band-pass filter. If will stop all frequencies above the lower cut-off frequency and below the upper cut-off frequency.

If you are using *Word,* use either the Microsoft Equation Editor or the *MathType* add-on (http://www.mathtype.com) for equations in your paper (Insert | Object | Create New | Microsoft Equation *or* MathType Equation). "Float over text" should *not* be selected.

IV. Working on the Similarities

First, to determine the similarities between two waves, we must understand relative concepts. Assuming two waves are x(t) and y(t), coefficient is a, and a*y(t) is approaching x(t). We can use the ratio of differences between two waves to tell the similarity. This is similar to the function of latus rectum in mathematics.

The differences can be defined as the area of the integral of x(t) – a*y(t). Coefficient, a, should be chosen to lower the error. Therefore, if x(t)*y(t)=0, the integral will be 0. In other words, two waves are not relative. If x(t)*y(t)=1, the integral will be 1. Two waves are identical.

Algorithms

Before we compare two signals, we have to save them as "WAV" files. Here we are using source and target

```
similarity(CString source, CString target)  //constructor
  {
    sourcefile = source; //source file
    targetfile = target; //target file
  }
```

First, we have to load two files into memory. Here we are using Class of CFile from MFC which is powerful and convenient to use.

```
    char *buf1;
    char *buf2;
    int m_nData1Len,m_nData2Len;

    CFile file;
    file.Open(sourcefile,CFile::modeReadWrite);   //open the
file
    buf1=new char [file.GetLength()];  //initial a buffer
    file.Read(buf1,file.GetLength());  // read the file
    m_nData1Len=file.GetLength();  //find the file length
    file.Close();  // close the file

    file.Open(targetfile,CFile::modeReadWrite);
    buf2=new char [file.GetLength()];
    file.Read(buf2,file.GetLength());
    m_nData2Len=file.GetLength();
      file.Close();
```

The code above is to read two wave files into memory. Buf1 and buf2 are two pointers to wave files, and the data type of them is char. And, m_nData1Len and m_nData2Len are the length of the two files. Once the reading is done, we should close the files. Now, we are ready to calculate the ratio of similarity between the two waves.

```
int N = m_nData1Len > m_nData2Len ? m_nData2Len : m_nData1Len;
```

It is possible that the lengths of the two files are different, so we choose the longer one as the target. The rest of the file is filled by 0. x(t) * y(t) = 0, so it does not make any difference.

```
double A,B,C,Pxy;
A = B = C = Pxy = 0.0f;
```

Here we use floating and double to store the numbers.

```
for(int i=0; i < N; i++) {
    A+=buf1[i]*buf2[i];
    B+=buf1[i]*buf1[i];
    C+=buf2[i]*buf2[i];
  }
  Pxy=A/(sqrt(B*C));
  delete[] buf1;
  delete[] buf2;
  return Pxy;
```

After the calculation, we delete two buffers to release memory. The result has been stored in the value of Pxy.

V. Experiments and Results

A. To Reduce the Noise of a Bird Sound

Now, let's make an example to show how to use this Noise Filter to reduce the noise of a bird sound WAV file, step by step. The objective bird sound file is amre.wav.

1. Double click the Filter.exe file to open the Noise Filter.

2. After double click, we get the "Noise Filter window" (see Fig 8).

Fig. 8 Noise Filter user interface

3. Click the "Get File" button.

4. We get the "Open file window" (see Fig 9).

Fig. 9 Open file window

5. Chose the folder of the bird sound file "amre.wav" then click the "OK" button.

6. After step 5, we come back to the Noise Filter window.

7. Click the "Set Para" button, to set the FIR parameters.

8. After step 7, we get the FIR Filter Window: Dialog (see Fig 10).

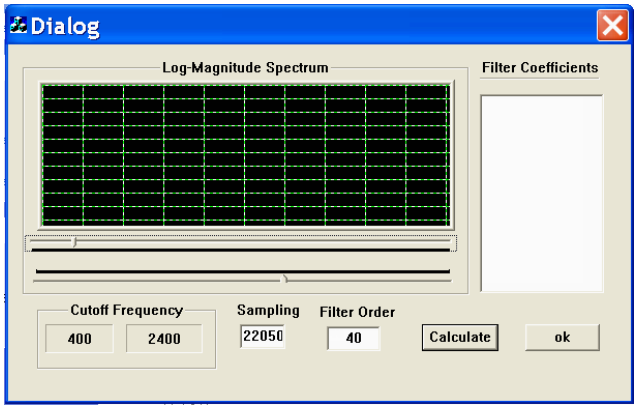


Fig. 10 FIR Filter Window: Dialog

9. Using "Cool Edit Pro" software, we see the frequency of the bird sound in the "amre.wav" file is between 6200 and 7500 Hz (see Figure 11), so we specify the lower cutoff frequency by pulling the lower cutoff frequency scroll bars until we reach 6196.1 Hz and specify the higher cutoff frequency by pulling the higher cutoff frequency scroll bars until we reach 7497.0 Hz. Now, the lower and higher cutoff frequency **cutoff frequency panel** will be automatically changed to 6196.1 and 7497.0 (see Fig 12).

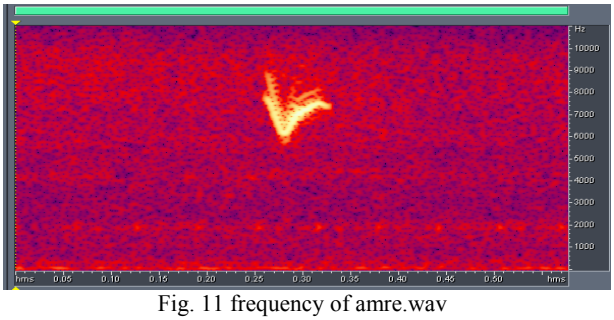


Fig. 11 frequency of amre.wav

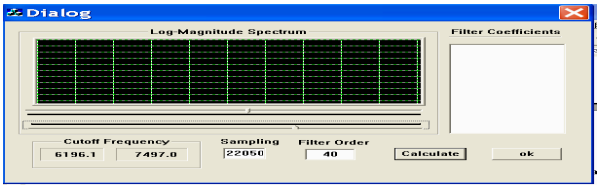


Fig. 12 FIR Filter Window with cutoff frequencies 6196.1 and 7497.0

10. Specify the sampling rate in the **Sampling Panel.** Following the sampling rate definition, the sampling rate is the frequency at which samples are taken. The sampling rate has to be twice or more the rate of the analog frequency that is captured; the higher the sampling rate, the better quality signal. So, the default sampling rate 22050 is good, and we do not have to change.

11. Specify the filter order in the **filter order field.** We still use the default filter order 40.

12. Click the "Calculate" button, then the filter coefficient will be displayed in the **filter coefficient** list box (see Fig 13).

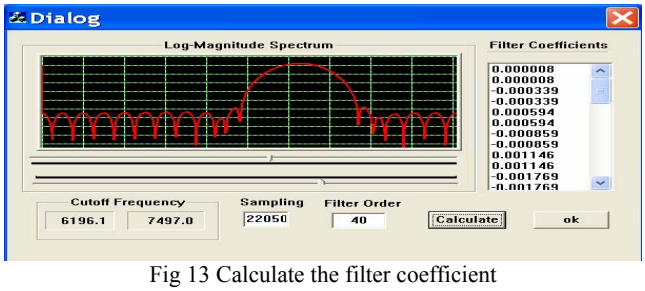


Fig 13 Calculate the filter coefficient

13. After filter coefficients have been calculated, the user can save the filter coefficients by clicking the "OK" button.
14. After step 13, we come back to the Noise Filter window.
15. Click the "Filtering" button to begin reducing the noise of the amre.wav file.
16. Click the "Save" button to save changes in a new, noise-reduced version WAV file. We get the "Save file window" (see Fig 14).

Fig 14 Save file window

17. Chose the folder, then save the new, noise-reduced version WAV bird sound file; give the file name "denoised amre.wav" from the Save File window, then click the "OK" button.

18. Done!

Now we have successfully used this Noise Filter to reduce the noise of a bird sound in the "amre.wav" file, and we have a new, noise-reduced version bird sound file called "denoised amre.wav". At last, let us use "Cool Edit Pro" software to see the frequency of the new, noise-reduced version bird sound file --"denoised amre.wav" (see Fig 15).

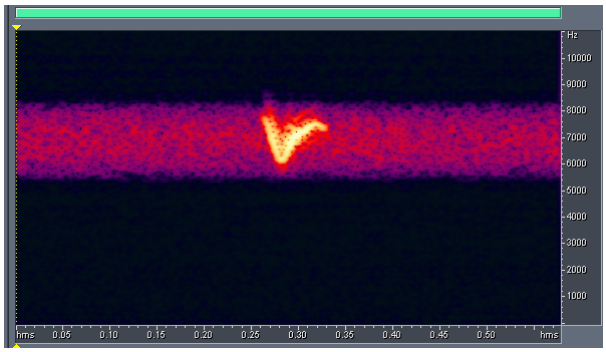
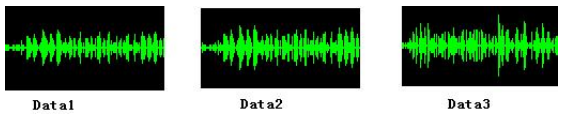


Fig 15 frequency of the noise-reduced version "denoised amre.wav".

From Fig 15, we can see that the noise filter successfully reduced the noise sound which frequency is not in the range of 6196.1 and 7497.0 Hz, and when we listen to the two bird sound files, we can easily notice that the noise-reduced version is much clearer than the original bird sound.

B. Testing Similarity

The following are three waves:



Using our program, the ratio of similarity between Data1 and Data2 is 0.7931, which means they are similar. The ratio of similarity between Data1 and Data3 is -0.0133 and the ratio of Data1 and Data2 is -0.001166, so both two of them are different. And, the results we hear among the three waves are relative to what we observe in the above visual representation.

V. Conclusion

From our experiments, this Noise Filter successfully reduced the noise of a bird sound "amre.wav" file, and delivered a new, noise-reduced version bird sound file "denoised amre.wav". It proves that this Noise Filter, which is based on the High-pass, Low-pass, Band-pass, Band-stop Filter and FIR (Finite Impulse Response) technology, works well. The algorithm we discussed is widely used in current communication technology. Of course, it is possible to make improvements in this algorithm. The program is tested under Windows XP and Microsoft Visual .NET

Acknowledgment

The authors would like to thank Dr. Phil Taylor and Mr. Mike Peckford, Department of Biology at Acadia University for their support work and some useful ideas for this project.

References

[1] Antoniou, A. Digital Filters: Analysis, Design, and Applications. 2nd ed. New York, NY: McGraw-Hill, 1993.
[2] Marler P., *Variations in the song of the chaffinch*, Fringilla coelebs, Ibis, 458-472, 1952.
[3] S. K. Mitra, J. Kaiser, *Handbook for Digital Signal Processing*, John Wiley and Sons, Inc. 1993.
[4] Proakis, J. and D. Manolakis. *Digital Signal Processing.* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[5] Thorpe, W. H., and B. I. Lade. *The songs of some families of the Passeriformes. I. The analysis of bird and their expression in graphic notation.* Ibis, 103a, 231-245, 1961.
[6] "WAVE sound file format" http://ccrma-www.stanford.edu/courses/422/projects/WaveFormat/
[7] "Wave Filters" http://www.smeter.net/filters/wave-filters.php
[8] "Digital FIR Filter Design" http://www.mathworks.com/access/helpdesk/help/toolbox/dspblks/digitalfirfilterdesign.html
[9] "Digital Filter Terminology" http://www.dspguru.com/info/terms/filtterm/index2.htm
[10] Fletcher Wildlife Garden(2003) : Getting Started in Birding http://www.ofnc.ca/fletcher/alphabet/beginbir.php