ELEC 301 Projects Fall 2008

Collection Editor:
Rice University ELEC 301

ELEC 301 Projects Fall 2008

Collection Editor:
Rice University ELEC 301

Authors:

Shamoor Anis
Ricky Barrera
Bailey Basile
Frank Chen
Yuheng Chen
Yung-Seok Kevin Choi
Christopher Eagleson
Georgios Evangelatos
Michael Foree
Jash GUO
Leslie Goldberg
Jaimeet Gulati
Jose Hernandez
Isaac Hernandez-Fajardo
Joel Khan
Samuel Kim
Ioannis Kougioumtzoglou
Robert LiKamWa
Xin Ming
Christine Moran
Seth O'Brien
Ryan Pei
Jake Poteet
Jeonggoo Song
Jon Stanley
Katherine Threlkeld
Daniel Valvano

Online:
< http://cnx.org/content/col10633/1.1/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: January 22, 2009
PDF generated: February 5, 2011
For copyright and attribution information for the modules contained in this collection, see p. 136.

Table of Contents

Chapter 1

Birdcall Identification

1.1 Birdcall Identification: Introduction[1]

1.1.1 Birdcall Identification Project

1.1.1.1 Motivation

Birdwatchers around the world have struggled with the arduous task of remembering and identifying the many birdcalls native to their area. As an added struggle, those who live in an urban environment are challenged to pull the sound of a birdcall out of the background noise present in the city. Our birdcall identification program attempts to computationally identify birdcalls. With a computational identifier, many problems associated with birdcall identification are mitigated: there is no need to remember birdcalls, the program can algorithmically separate the birdcall from the background, and the sound file can be automatically saved to the computer.

1.1.1.2 Method

Previous attempts to identify bird sounds relied solely on spectrograms. Our process focuses on using a time-domain matched filter and frequency analysis in tandem to achieve accurate results. The matched filter identifies the similarities between two sounds as a function of time. Frequency analysis differs two sounds based on the energy in the frequency spectrum. The matched filter in conjunction with frequency analysis provide accuracy far surpassing that of lone spectrograms.

1.2 Birdcall Identification: Bird Choice[2]

1.2.1 Bird Choice

We elected to use six different birdcalls in our project: the common loon wail; the common loon tremolo; the red-tailed hawk cry; the red-tailed hawk shriek; the bobwhite quail mating-call; and the ferruginous pygmy-owl hoot. The group selected these calls based upon two major criteria. Each call needed to be available from multiple sources, and each call had to be audibly different from the other calls selected.

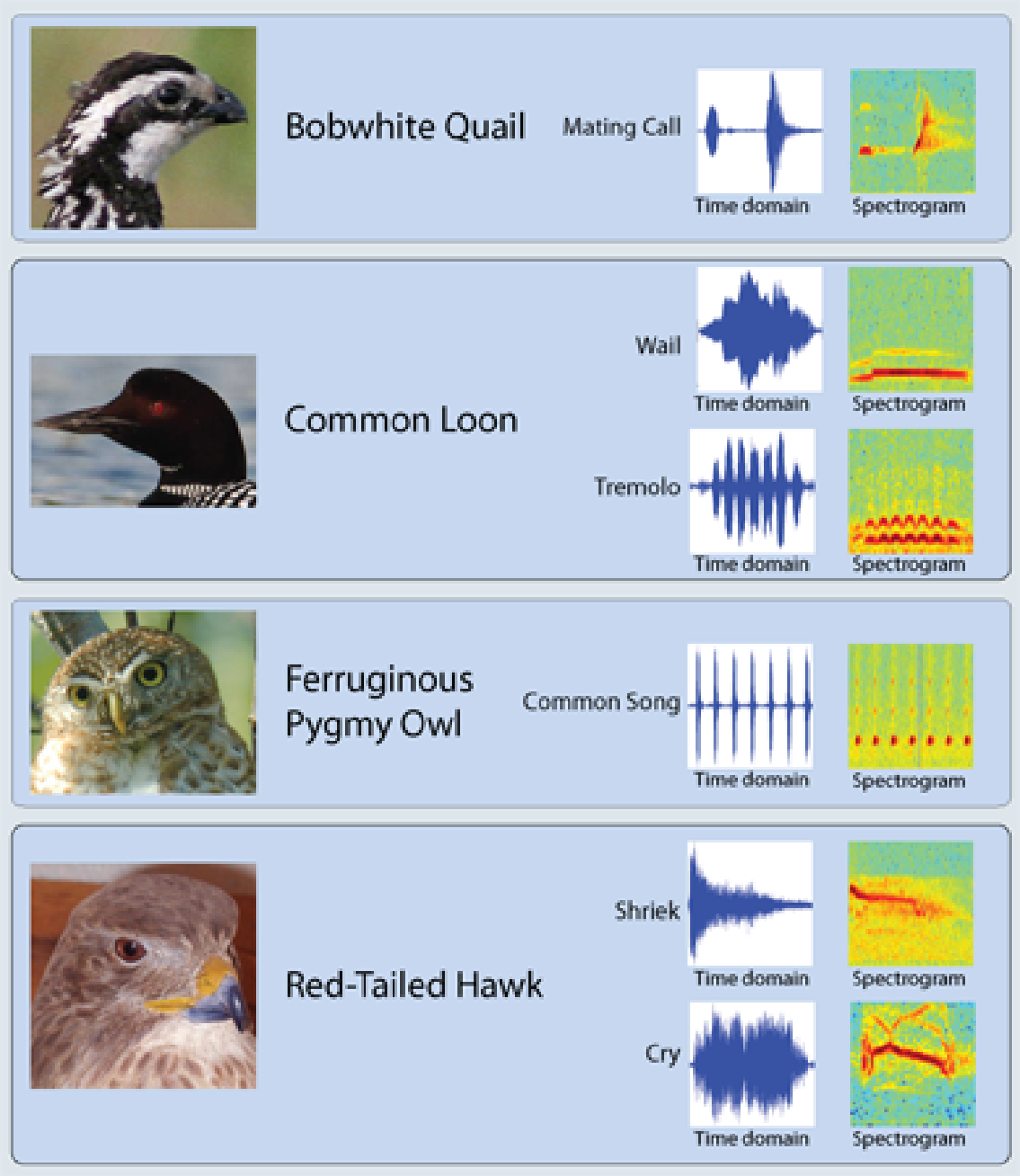
The project group contains no bird experts, so we used only prerecorded birdcall clips as samples. Such audio clips saved us the necessity of making field recordings. Also, the clips' creators, who presumably possess more ornithological expertise than we, had already identified the birds present in each recording. For formatting reasons, we chose only audio clips saved as wav files. However, with this constraint, relatively few bird types suited our needs. Adults of each bird species have up to fourteen or fifteen call types. Of these

[1] This content is available online at <http://cnx.org/content/m18950/1.3/>.
[2] This content is available online at <http://cnx.org/content/m18933/1.6/>.

types, frequently only one or two are available as recordings, and of recorded types, almost none exist in multiple, wav-file examples on the internet, or in libraries. This significantly reduced our pool of candidate birds.

Our second major criterion was that the chosen birdcalls be audibly and spectrographically distinctive. Although spectrographic analysis can easily reveal differences between signals in the frequency domain, human error is less likely when group members can tell the difference between time-domain calls by ear. Having a variety of birdcalls, some similar, some radically different, also produces a more interesting analysis than the use of entirely alike, or entirely dislike calls.

The application of both criteria resulted in the final selection of birds.



1.3 Birdcall Identification: Matched Filter Implementation[3]

1.3.1 Matched Filter Implementation

1.3.1.1 Matched Filter

Matched filters do an excellent job of identifying sound samples, so we decided to apply the method here to identify birdcall audio files. A matched filter searches for a sample clip, the filter, within a longer audio recording. Convolution compares the filter to the longer signal at each possible offset. The greater the maximum amplitude of the convolution result, the stronger the match. By having a different filter for each birdcall, we can search an audio file to identify which birdcall it contains.

The matched filter algorithm is as follows:

1. Reverse the filters in the time-domain.
2. Normalize the energy of each of the filters.
3. Convolve each filter with the input signal and take the maximum amplitude of the resulting convolution signals.
4. The filter that gives us the greatest maximum value indicates which birdcall the signal contains.

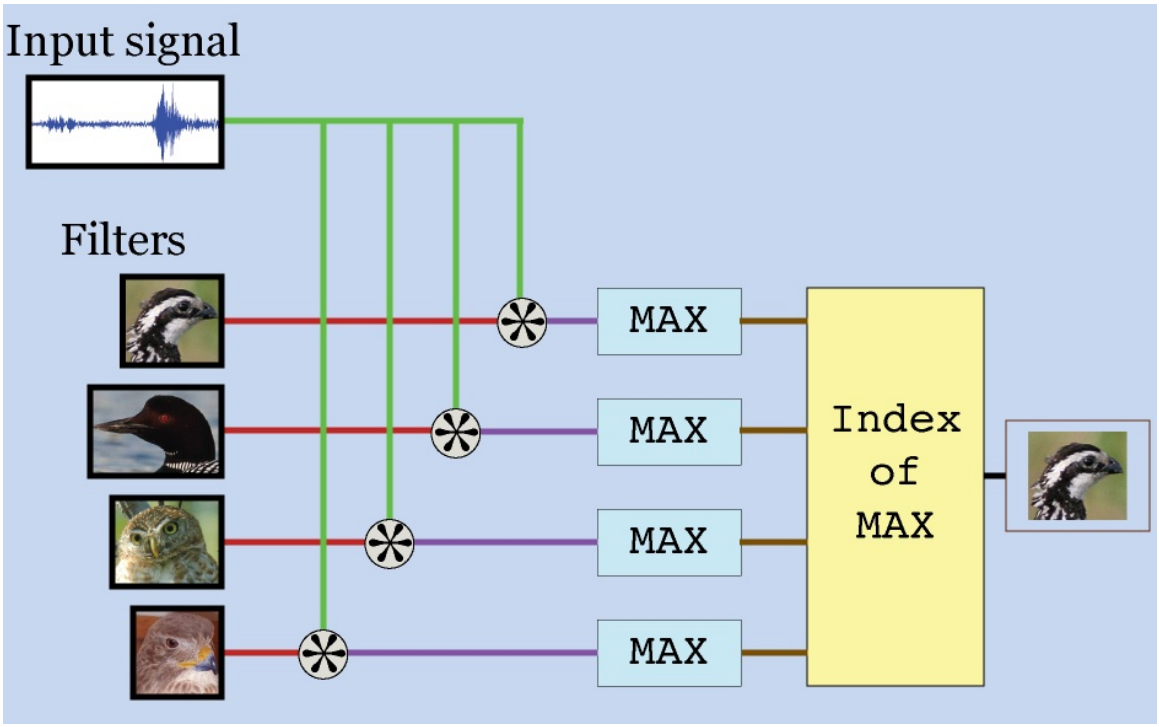


Figure 1.1

[3]This content is available online at <http://cnx.org/content/m18917/1.3/>.

1.3.1.2 Filter Library Creation

Our first step in implementing the matched filter algorithm was to create a library of birdcall filters. To do this, we looked at the spectrograms of a few sample audio files of the same birdcall and selected a portion that looked representative of the call.

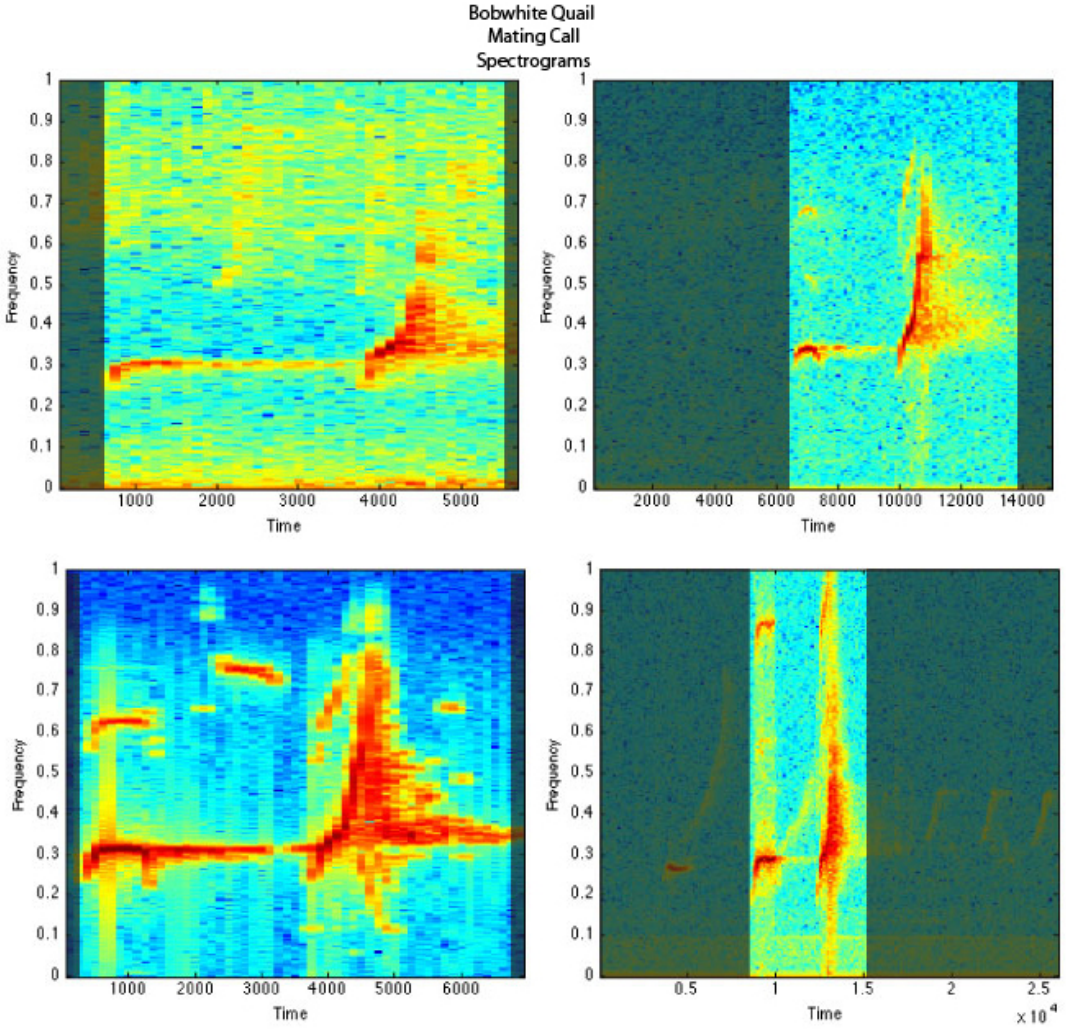


Figure 1.2

For each birdcall, one of the representative audio segments was saved as a filter. Because the first two steps of the above matched filter algorithm affect only the filter library and are independent of the input signal, we reversed the filters and normalized their energy before saving them to wave files.

1.3.1.3 Matlab Implementation

The following MATLAB script performed our matched filter algorithm. When given a wave file as input, it would tell us how well the audio sample matched against each of the 6 birdcall filters.

NOTE: We used circular convolution instead of linear convolution because it computed much faster. MATLAB's built-in cconv function zero-pads the two signals before multiplying their FFTs, generating the convolution result we are looking for.

```
function result = birdcheck(file)
[sig, fs, nbits] = wavread(file);
signal=sig(:,1);
signal=signal/max(abs(signal));
filters{1}=wavread('filters/bob.wav');
filters{2}=wavread('filters/lt.wav');
filters{3}=wavread('filters/lw.wav');
filters{4}=wavread('filters/pygmy.wav');
filters{5}=wavread('filters/red.wav');
filters{6}=wavread('filters/redcry.wav');
for i=1:6
  filter=filters{i};
  result(i) = max(abs(cconv(signal,filter(end:-1:1))));
end
end
```

The script was able to correctly identify several birdcalls. It did fail to correctly identify four cases in two categories:

1. Two of our loon tremolo files registered as pygmy owl common songs.
2. Two of our red-tailed hawk shriek files registered as red-tailed hawk cries.

1.4 Birdcall Identification: Method Refinement[4]

Need for Improvement
The matched filter implementation gave the right result in most cases, but occasionally reported a fairly good correlation between an input signal and a filter that were clearly quite distinct from one another. The matched filter often doesn't recognize a strong difference between two signals if one has power in a particular frequency range that the other lacks. For instance, a loon tremolo has a wide spectrum compared to a ferruginous pygmy owl, but the matched filter often reported a good match because the owl's pitch was in the loon's range.
Frequency-domain Analysis
We hoped to adjust the matched filter results by taking into account the frequency-domain error between the signal and the filter. The convolution step of the matched filter showed not only the strongest match between the signal and the filter, but also the time of that match in the input signal. We windowed out the chunk of the input signal that matched best. We then used the mean-squared error between the magnitudes of that chunk's FFT and the filter's FFT to obtain a number indicating how well the frequency content of the two signals matched.

As an example, a comparison of a hawk shriek versus a filter for a hawk cry shows that their frequency ranges don't match very well:

[4]This content is available online at <http://cnx.org/content/m18919/1.2/>.

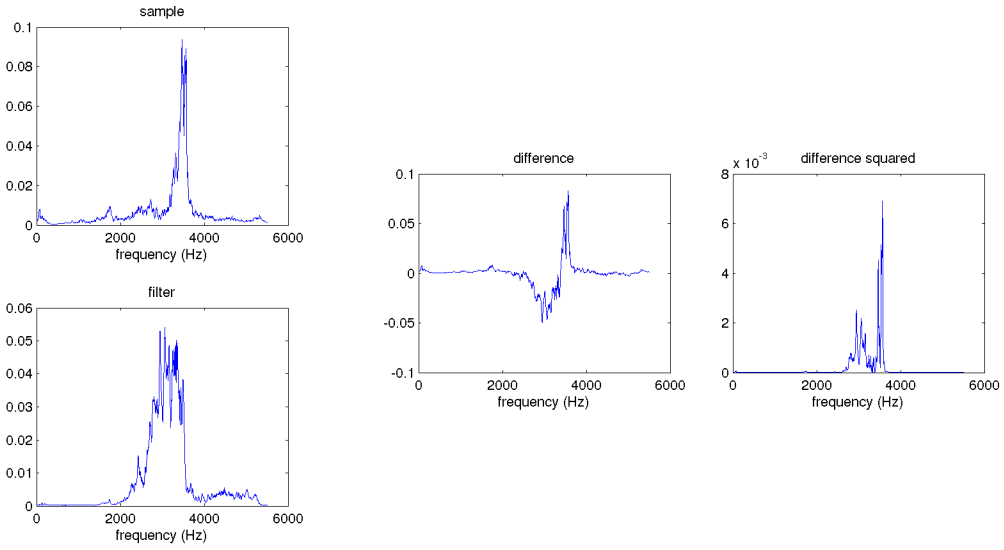


Figure 1.3

In contrast, the same shriek versus a filter for a shriek shows a lower mean-squared error (0.4637 compared to 0.9257, after scaling):

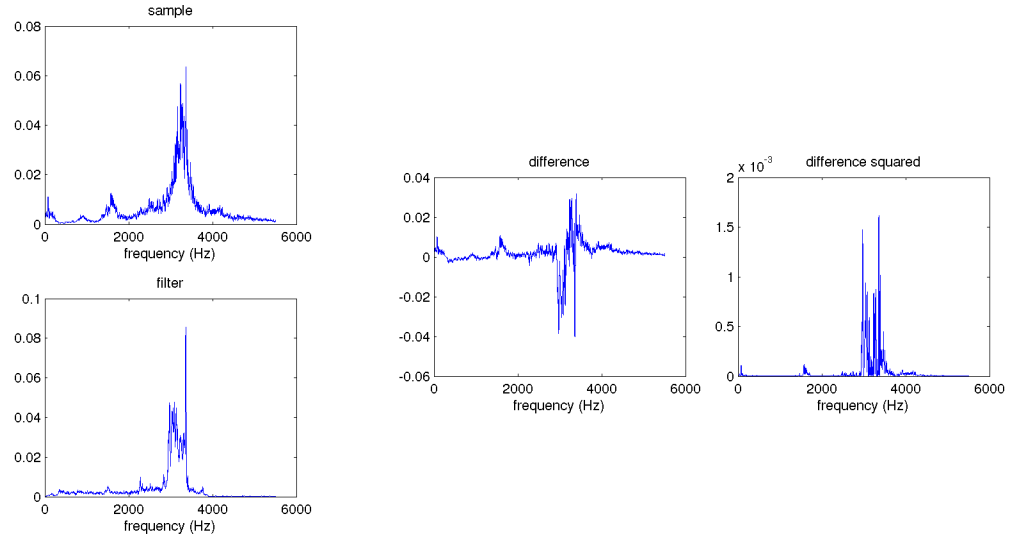


Figure 1.4

Correction Algorithm
The mean-squared FFT error by itself was a good predictor of which filter matched best with an input signal, despite ignoring phase information to focus only on magnitudes. We divided the matched filter output by the mean-squared error to arrive at a final result value.

> NOTE: We added a small number to the error before dividing, to avoid dividing by zero in the case where the signal perfectly matches the filter.

In all cases where the matched filter had selected the correct bird as the strongest match, this adjustment increased the ratio between the strong match and the others. It corrected the errors between different red-tailed hawk vocalizations, but wasn't sufficient to recognize the loon tremolo.

1.5 Birdcall Identification: Matlab GUI[5]

1.5.1 Matlab Bird Call Identification Program

1.5.1.1 Matlab GUI

The group created a MATLAB Graphical User Interface (GUI) to implement the birdcall identification algorithms.

[5] This content is available online at <http://cnx.org/content/m18918/1.2/>.

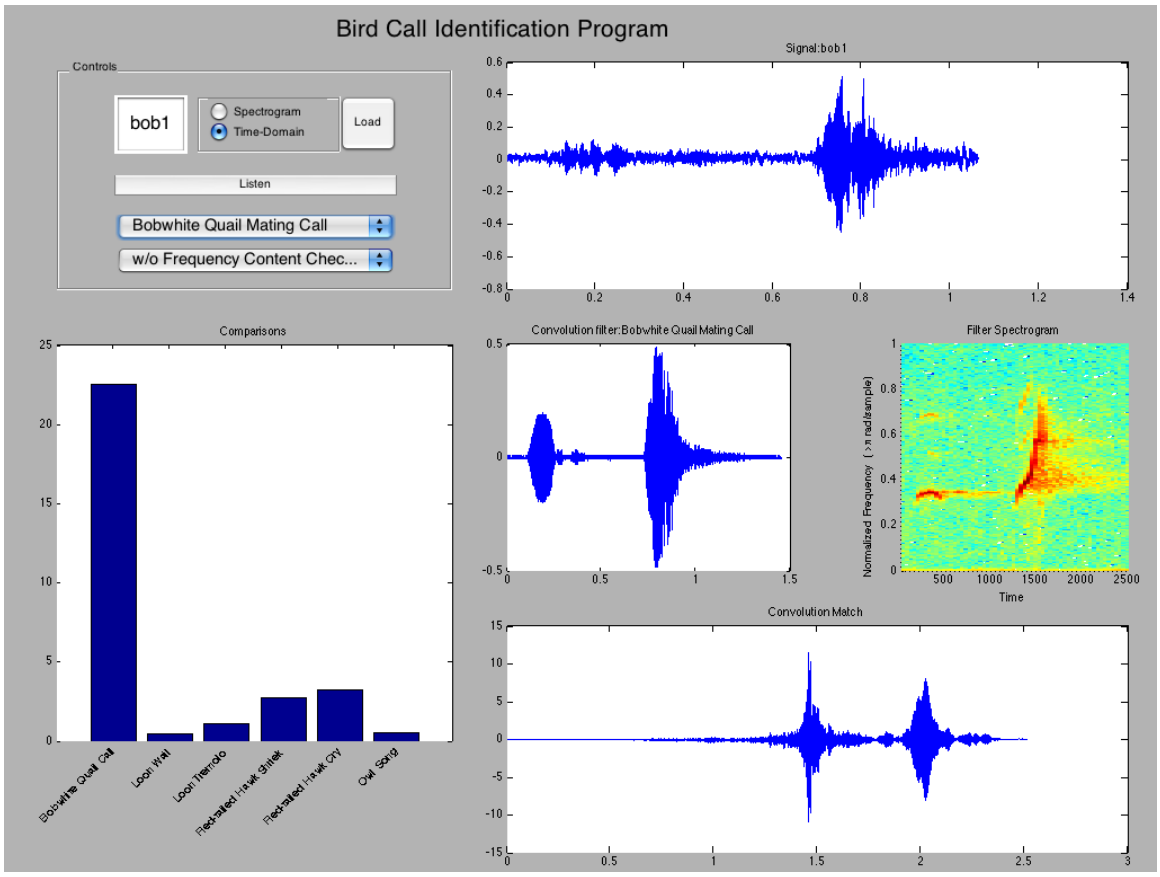


Figure 1.5

Our GUI was able to load in audio files, play them, and display the result of convolving the audio signal with a specific filter. It also produced a bar graph of the maximum of the convolution result for each filter. This would graphically show us not only which birdcall the audio signal contained, but also how confidently the program was reporting one birdcall over another.

The GUI has the ability to choose whether or not to use our frequency content checking algorithm.

1.6 Birdcall Identification: Results[6]

1.6.1 Results

These results show how strongly the different audio samples matched up to the different filters.

The number outside of the parentheses indicates the results with frequency content checking turned on. The number inside the parentheses indicates the results with frequency content checking turned off.

All numbers are relative to the greatest match found for a given sample.

Bobwhite Quail mating call samples against different filters (4 sample audio files):

[6] This content is available online at <http://cnx.org/content/m18952/1.1/>.

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry
1.0000 (1.0000)	0.0195 (0.0492)	0.0076 (0.0200)	0.0092 (0.0241)	0.0661 (0.1210)	0.0648 (0.1444)
1.0000 (1.0000)	0.0002 (0.0050)	0.0001 (0.0019)	0.0002 (0.0044)	0.0050 (0.0651)	0.0055 (0.0681)
1.0000 (1.0000)	0.5229 (0.7098)	0.0312 (0.0555)	0.0194 (0.0343)	0.1808 (0.2484)	0.5020 (0.5506)
1.0000 (1.0000)	0.3218 (0.4168)	0.0252 (0.0372)	0.0735 (0.1055)	0.1417 (0.1777)	0.1406 (0.1810)

Table 1.1

Loon tremolo samples against different filters (5 sample audio files):

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry
0.1415 (0.1839)	0.5374 (0.5355)	0.1321 (0.1692)	1.0000 (1.0000)	0.1288 (0.1597)	0.1314 (0.1729)
0.0687 (0.0858)	1.0000 (1.0000)	0.7520 (0.8008)	0.3405 (0.4000)	0.2189 (0.2581)	0.0478 (0.0604)
0.0145 (0.0432)	0.1217 (0.2971)	0.0138 (0.0415)	1.0000 (1.0000)	0.0227 (0.0651)	0.0058 (0.0176)
0.0004 (0.0077)	1.0000 (1.0000)	0.0079 (0.1194)	0.0045 (0.0758)	0.0011 (0.0197)	0.0002 (0.0032)
0.0142 (0.0417)	1.0000 (1.0000)	0.2819 (0.5282)	0.0375 (0.1053)	0.0363 (0.0966)	0.0305 (0.0884)

Table 1.2

Loon wail samples against different filters (5 sample audio files):

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry
0.0027 (0.0084)	0.2130 (0.3991)	1.0000 (1.0000)	0.0137 (0.0406)	0.0109 (0.0327)	0.0011 (0.0035)
0.0058 (0.0109)	0.3337 (0.4185)	1.0000 (1.0000)	0.0221 (0.0393)	0.0278 (0.0503)	0.0025 (0.0047)
0.0123 (0.0245)	0.1891 (0.2551)	1.0000 (1.0000)	0.1412 (0.2133)	0.0222 (0.0436)	0.0120 (0.0239)
0.0633 (0.1744)	0.1927 (0.3783)	1.0000 (1.0000)	0.0870 (0.2364)	0.1453 (0.3819)	0.0072 (0.0198)
0.0001 (0.0030)	0.0039 (0.0598)	1.0000 (1.0000)	0.0004 (0.0072)	0.0005 (0.0099)	0.0001 (0.0015)

Table 1.3

Pygmy owl common song samples against different filters (3 sample audio files):

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry

continued on next page

0.1991 (0.2862)	0.1992 (0.2607)	0.0325 (0.0482)	1.0000 (1.0000)	0.3698 (0.5210)	0.0497 (0.0689)
0.0002 (0.0044)	0.0023 (0.0390)	0.0003 (0.0051)	1.0000 (1.0000)	0.0004 (0.0080)	0.0001 (0.0014)
0.0056 (0.0432)	0.0340 (0.2394)	0.0054 (0.0451)	1.0000 (1.0000)	0.0086 (0.0654)	0.0071 (0.0560)

Table 1.4

Red-tailed hawk shriek samples against different filters (3 sample audio files):

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry
0.1069 (0.3132)	0.0156 (0.0641)	0.0149 (0.0646)	0.0083 (0.0356)	1.0000 (0.9954)	0.5033 (1.0000)
0.4171 (0.5852)	0.0650 (0.1608)	0.0245 (0.0643)	0.0328 (0.0844)	1.0000 (0.8354)	0.9876 (1.0000)
0.0050 (0.0650)	0.0005 (0.0085)	0.0004 (0.0071)	0.0004 (0.0074)	1.0000 (1.0000)	0.0337 (0.1755)

Table 1.5

Red-tailed cry samples against different filters (2 sample audio files):

Bobwhite Quail Mating Call	Loon Tremolo	Loon Wail	Pygmy Owl Common Song	Red-tailed Hawk Shriek	Red-tailed Hawk Cry
0.0082 (0.0734)	0.0001 (0.0025)	0.0001 (0.0016)	0.0001 (0.0027)	0.0377 (0.2092)	1.0000 (1.0000)
0.0933 (0.3682)	0.0028 (0.0204)	0.0012 (0.0089)	0.0019 (0.0145)	0.4080 (0.5496)	1.0000 (1.0000)

Table 1.6

1.7 Birdcall Identification: Conclusion[7]

1.7.1 Birdcall Identification Project

1.7.1.1 Conclusion

The combination of the matched filter process with the frequency content analysis proved to be a fairly robust tool in identifying birdcalls. The matched filter tells us which birdcall is found in the time-domain, while the frequency analysis confirms the proper frequency content at the site of the match.

Our MATLAB GUI implementation gave a simple interface to identify birdcalls within a wave file. The interface gives a visual confidence reading regarding which birdcalls are within a sound sample. With more time, the program could easily be translated to a web application or mobile interface for use in the field.

1.7.1.2 Future Work

Our implementation was simply a test of concept. If we wished to take the project further, the first step would be to increase the filter library size. If made mobile, we could create a filter library that would mold itself to the species of birds in the specific geographic location, giving our algorithm less to check against.

Though our program can be made faster, the project as a whole was highly successful. Further work needs to be done to ensure the reliability of the program, especially as birdcalls are added. In the end, a venture into a marketing plan could prove to be lucrative.

[7] This content is available online at <http://cnx.org/content/m18954/1.1/>.

1.7.1.3 Acknowledgments :

Special Thanks to:

- Our instructor, Rich Baraniuk
- Our TA, Matthew Moravec
- Our Mentor, Marcos Duarte

Sound files from:

- www.dovehunt.org
- wildspace.ec.gc.ca
- www.michiganloons.org
- www.wildernessbay.com
- www.sylvialake.org
- www.avians.net
- www.birding.com.br
- www.owling.com

Chapter 2

Building an Electrocardiogram (ECG) Diagnostic System

2.1 Introduction: Building an Electrocardiogram (ECG) Based Diagnostic System[1]

2.1.1 Introduction: Building an Electrocardiogram (ECG) Based Diagnostic System

Our goal is to build an Electrocardiogram (ECG) that not only calculates the heart rate automatically, but can also detect other heart abnormalities as well. This requires more advanced analysis of the ECG Signal. There are several steps that need to be accomplished in order to achieve this goal, as outlined in the flowchart below.

Overall Data Flowchart for an Automatic ECG

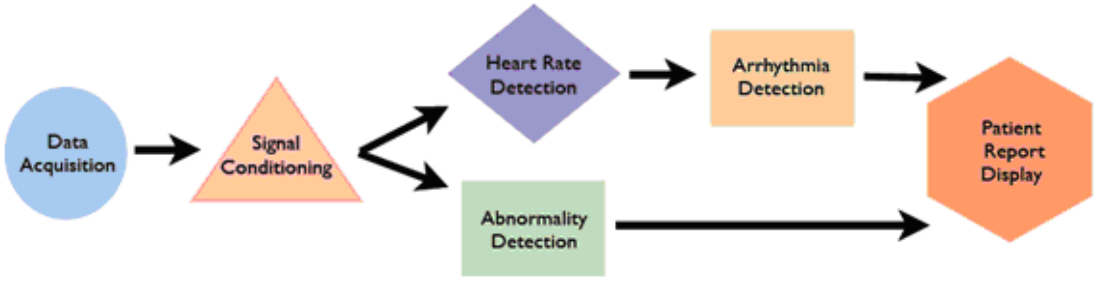


Figure 2.1: Breaks down the different steps which need to be accomplished in order to accomplish ECG signal analysis

Data acquisition and signal conditioning are covered in Collecting and Filtering Live ECG Signal[2] . The remaining phases, which are all related to signal analysis are covered in Algorithms for ECG Signal Analysis[3]

[1] This content is available online at <http://cnx.org/content/m18955/1.2/>.
[2] http://cnx.org/content/m18955/latest/m18956
[3] http://cnx.org/content/m18955/latest/m18957

.

Before we go into more detail about how to build an ECG, it is helpful to understand how the ECG works and how to interpret the data you receive.

2.1.1.1 Physiological Background of the ECG

An ECG is a non-invasive diagnostic device to monitor the condition of the heart through its electrical activity. This signal is acquired through externally located electrodes that adhere to the skin. A simple, clinical lead placement uses three leads: left arm, right arm and left leg (Figure 2). The electrical activity versus time forms an electrograph and can be used to determine and diagnose heart abnormalities and arrhythmias. This principle is based on Einthoven's law.

Electrode Placement for Three-Lead ECG

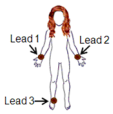


Figure 2.2: Lead 1 is attached to the right wrist, Lead 2 to the left wrist and Lead 3 is attached to the right ankle as a ground electrode.

All recorded electrical activity of the electrocardiogram corresponds to the net electrical current in the heart over time, depolarizing parts of the heart in sequence. The electrical impulse is initiated by the sinoatrial (SA) node. This causes the atria to contract and is evident on the ECG as the **P wave**. Next, there is a delay caused by the conduction of the impulse to the atrioventricular (AV) node such that the physical contraction of the atria have time to complete before the contraction of the ventricles. The **QRS complex** on the ECG is due to the depolarization of the ventricles, and occurs when the ventricles contract. Finally, the **T wave** on the ECG is due to the repolarization of the ventricles (*Pflanzer, 2004*). Therefore, each heartbeat corresponds to a pulse on the ECG beginning with each P wave and the ending with each T wave. The heart rate can be determined by determining the time it takes to complete one beat and is typically reported in beats per minute. Figure 3 demonstrates the characteristic shape of the waveform in a healthy patient.

Typical ECG Signal

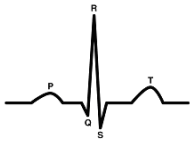


Figure 2.3: An example of the typical shape and location of the various components of an ECG signal.

2.1.1.2 Background Information on Monitored Heart Conditions

To perform automatic detection of an ECG signal, there needs to be something that clearly delineates a certain abnormality from other signals. Therefore, the signal processing selected is to detect ventricular hypertrophy and old myocardial infarctions. Luckily, these types of abnormalities are both very useful for doctors to diagnose a patient and have distinguishable ECG features.

Ventricular hypertrophy is the enlargement of either of the ventricles. Left ventricular hypertrophy is particularly common in athletes as well as an indicator of hypertension. It is also used in the Framingham risk equation to predict future cardiac problems the patient may face (*ECG Abnormalities, 2006*). One of its characteristic ECG patterns is the inverted T wave (Figure 4).

An Example of an ECG Signal with an Inverted T-wave

Figure 2.4

A myocardial infarction (a.k.a. heart attack) is caused by the complete blockage of one of the coronary arteries. The coronary artery is what supplies the heart muscle with blood. A blockage prevents blood from reaching the surrounding muscular tissue resulting in necrosis. This damage is permanent so the resulting ECG characteristic will remain with the patient. Therefore, the doctor can easily tell if a patient has had a heart attack in the past. The ECG of an old myocardial infarction is characterized by a significant Q wave (Figure 5). This means that the Q peak is unusually deep, usually with amplitude of about one-third that of the R peak (*Dubin, 2000*).

An Example of an ECG Signal with a Significant Q-wave

Figure 2.5

By building an ECG that can automatically detect these and potentially other heart abnormalities, it will be easier for doctors to monitor multiple patients. This is especially important in third world countries where there are often far too many patients in one clinic than one doctor or nurse can adequately care for.

2.2 Collecting and Filtering Live ECG Signal[4]

2.2.1 Collecting ECG Signal: Hardware

Using the NI Elvis breadboard and data acquisition system, the ECG signal is collected from the three leads. Lead 1 is connected to the right arm, Lead 2 is connected to the left wrist and Lead 3 is connected to the right ankle (see Introduction: Building an ECG Based Diagnostic System[5] for more information about lead

[4] This content is available online at <http://cnx.org/content/m18956/1.2/>.
[5] http://cnx.org/content/m18956/latest/m18955

placement). The diodes placed between the input leads and the rest of the circuit are to protect the patient from any backflowing current.

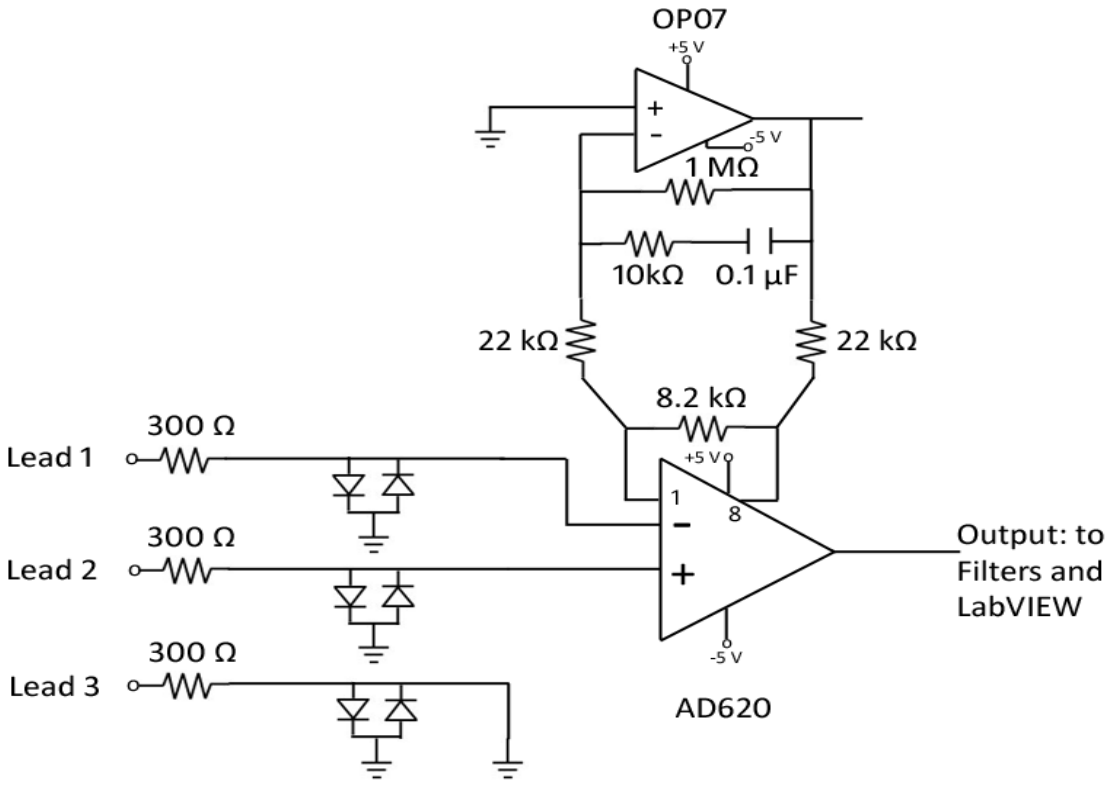


Figure 2.6: ECG signal collection circuit adapted from that of Radio Locman (www.rlocman.ru). The AD620 is used as a differential amplifier with a gain of ∼7 in order to combine the electrical signals from each of the leads into one easily readable signal.

In NI LabVIEW, the Data Acquisition Assistant (DAQ Assistant) is used to collect the signal after preliminary band pass filtering. The data is sampled at a rate of 1 kHz. Our LabVIEW VI is available here[6] .

2.2.2 Signal Conditioning

Signal filtering is necessary to help isolate the frequencies found in the ECG signal from the noise. With a three lead system, the majority of the noise comes from the electrical activity in the muscles on the arm,

[6] http://cnx.org/content/m18956/latest/ECG_LIVE.vi

or electromyography (EMG) noise. EMG signals are present in a wide frequency band which overlaps with the ECG signal in the lower frequencies. Therefore, with this set-up, it is impossible to completely remove EMG noise from the ECG signal. Therefore, it is helpful for the patient to relax and remain still while the data is being collected. In addition, 60 Hz noise is present from power line interference which also must be removed.

2.2.2.1 Analog Band Pass Filter Design

The first stage of filtering is an analog filter built onto the NI Elvis breadboard. It is a bandpass filter with cut-off frequencies of 0.5 and 150 Hz. This will help eliminate the high frequency noise from the muscles before the signal is greatly amplified.

Analog Bandpass Filter Circuit Diagram

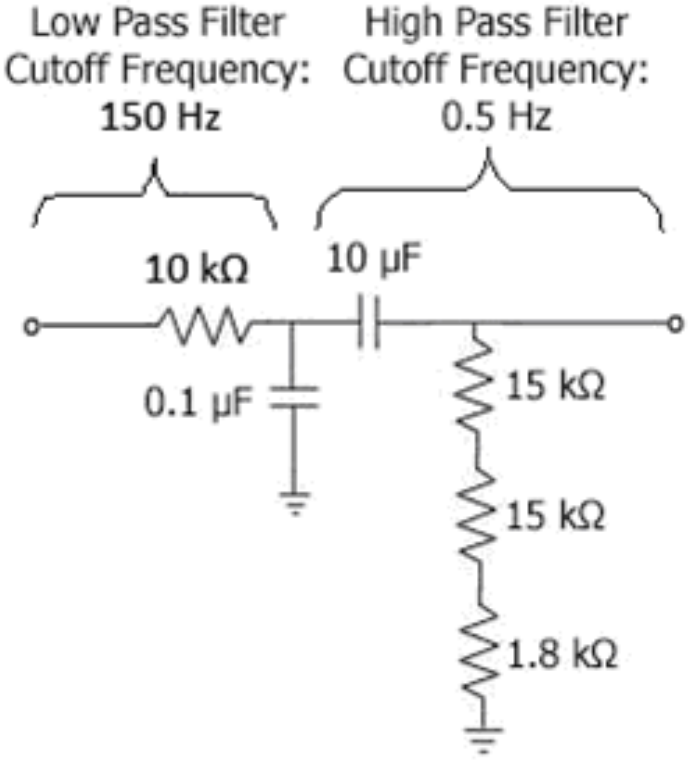


Figure 2.7

Bode Plot – Analog Band Pass Filter

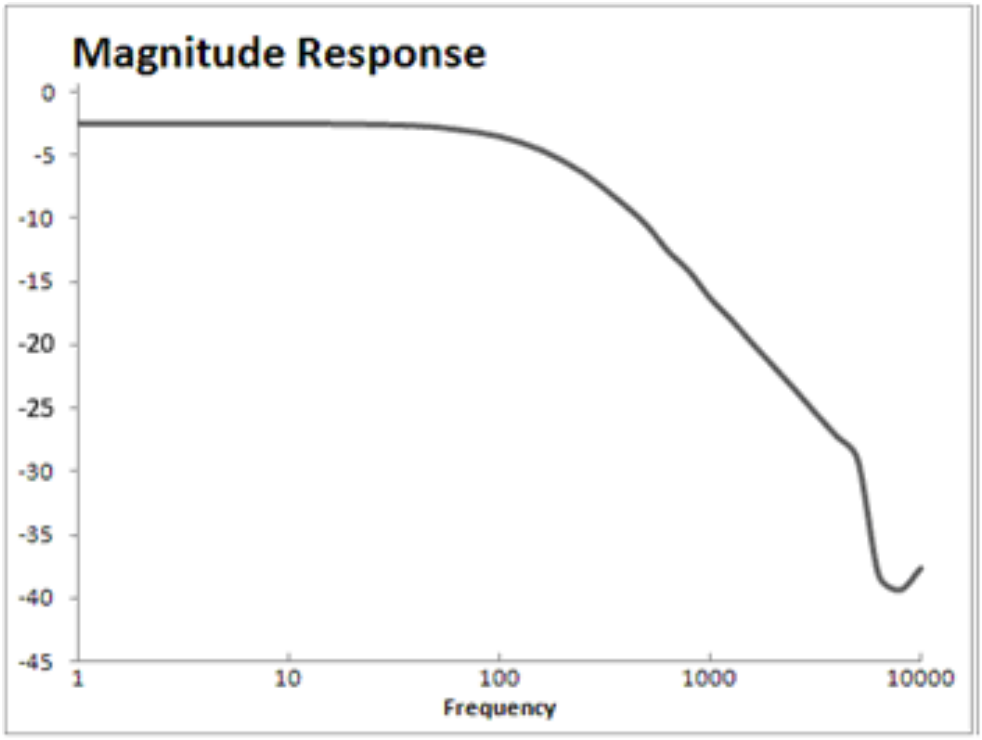


Figure 2.8

2.2.2.2 Digital Filters Using LabVIEW

Once the signal has been acquired by the DAQ Assistant into LabVIEW, it is processed by two additional filters and amplification of 100 times. The first filter is a band stop filter between 55 and 65 Hz to eliminate power line interference. A third order Butterworth (IIR) was used to implement this filter because it is low order and has a good frequency response for this signal.

Magnitude of the Frequency Response – Digital Band Stop Filter

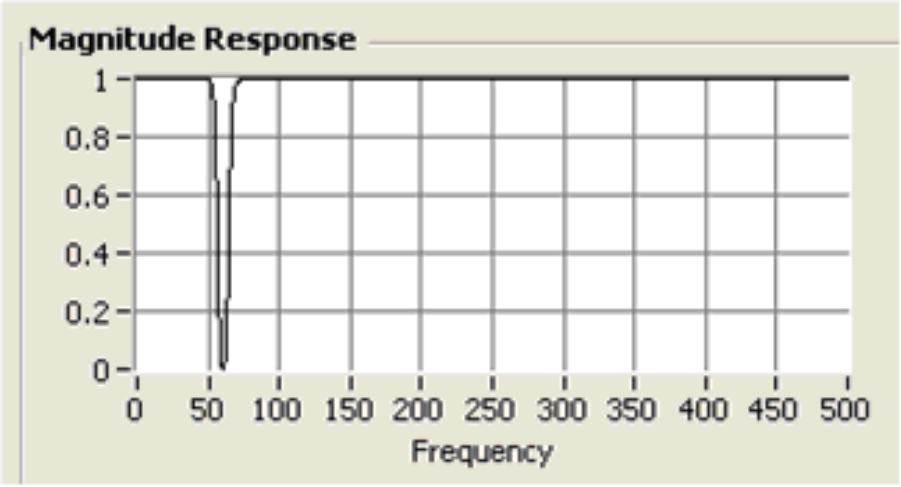


Figure 2.9

The second is a tenth order Butterworth low pass filter. The cutoff frequency of this filter is 80 Hz to further eliminate EMG noise. The ECG signal is located between 0.5 Hz and about 70-80 Hz depending on the individual.

Magnitude of the Frequency Response – Digital Low Pass Filter

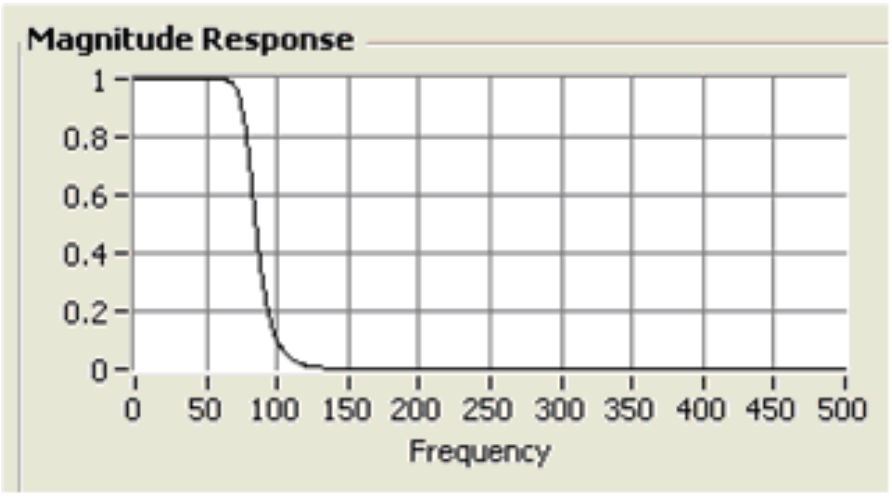


Figure 2.10

The resulting signal is displayed in real time in the graphical user interface, also designed in LabVIEW.

2.3 Algorithms for ECG Signal Analysis[7]

2.3.1 ECG Signal Analysis: Abnormality Detection

The flowchart below breaks down the tasks needed to accomplish signal analysis in greater detail. Both LabVIEW and Matlab were used to accomplish these tasks.

Signal Analysis Flow Chart

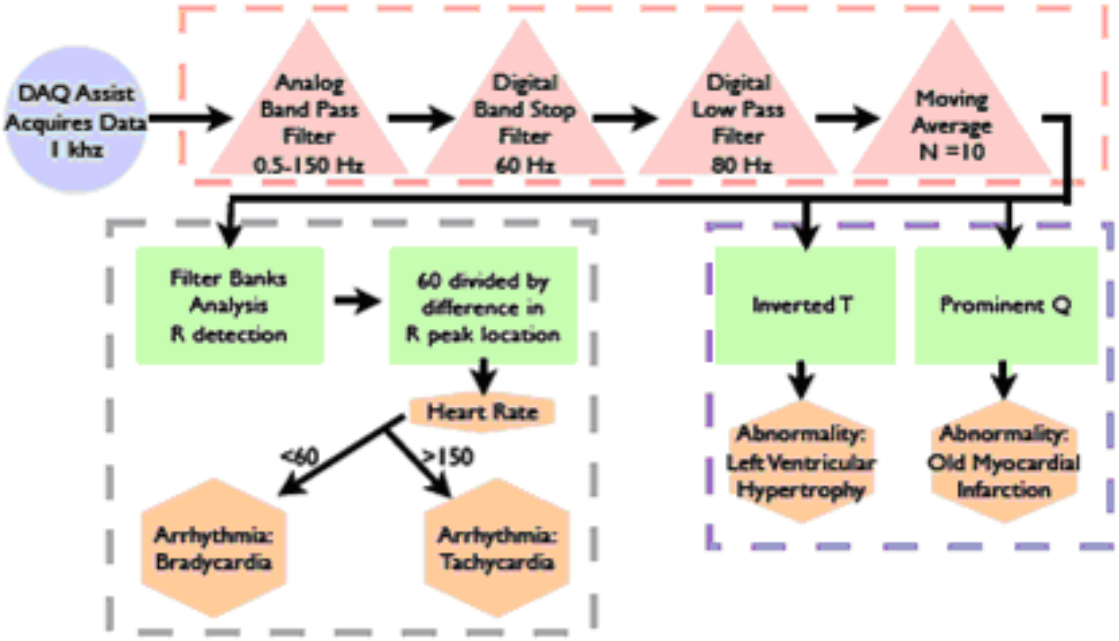


Figure 2.11

2.3.2 Matlab Implementation of Filter Banks Analysis

Many scenarios deal with signals which contain specific energy distributions in the frequency domain. For example, with regard to the ECG, a significant proportion of the energy from the QRS complex extends to a frequency of 40 Hz, and even more if the Q, R, and S waves have very sharp morphologies. The P and T waves, in general, have a significant proportion of their energy only up to 10 Hz. Thus, a strategic approach to detecting heartbeats is to analyze different sub-bands of the ECG, rather than just the output of one filter which maximizes SNR of the QRS.

In this filter bank analysis technique we use 5 sub-bands, each one has bandwidth 5Hz. The ECG signal is processed by those 5 sub-band filters and downsampled. The downsampled signal is

[7] This content is available online at <http://cnx.org/content/m18957/1.1/>.

$$W_l(z) = \frac{1}{M} \sum_{k=0}^{M-1} U_l\left(z^{1/M} W^k\right)$$
$$= \frac{1}{M} \sum_{k=0}^{M-1} H_l\left(z^{1/M} W^k\right) X\left(z^{1/M} W^k\right)$$
$$l = 0, 1, \cdots, M-1.$$

Figure 2.12

A variety of features which are indicative of the QRS complex can be designed by combining sub-bands of interest from l = 0, 1, . . . , M-1. For example a sum-of-absolute values feature P1 can be computed using sub-bands 1, 2, and 3.

$$P_1 = \sum_{i=1}^{3} \left|W_l(z)\right|$$

Figure 2.13

P1 has a value which corresponds to the energy in the frequency band [5.6, 22.5] Hz. Similarly, P2 and P3 can be computed using sub-bands {1, 2, 3, 4}, and {2, 3, 4}, respectively, and these values are proportional to the energy in their respective sub-bands. Heuristic beat detection logic can be used to incorporate some of the above features which are indicative of the QRS complex.

Figure 4 gives an overview of the sequential levels in the beat detection algorithm. The goal of the detection algorithm is to maximize the number of true positives (TP's), while keeping the number of false negatives (FN's) and false positives (FP's) to a minimum. Since it is not possible to arrive at this goal using one simple detector, multiple detectors with complementary FN's and FP's performances are simultaneously operated and the results of each fused together to arrive at an overall decision. The advantage of this strategy is that multiple features which are indicative of the QRS complex can be used to detect beats.

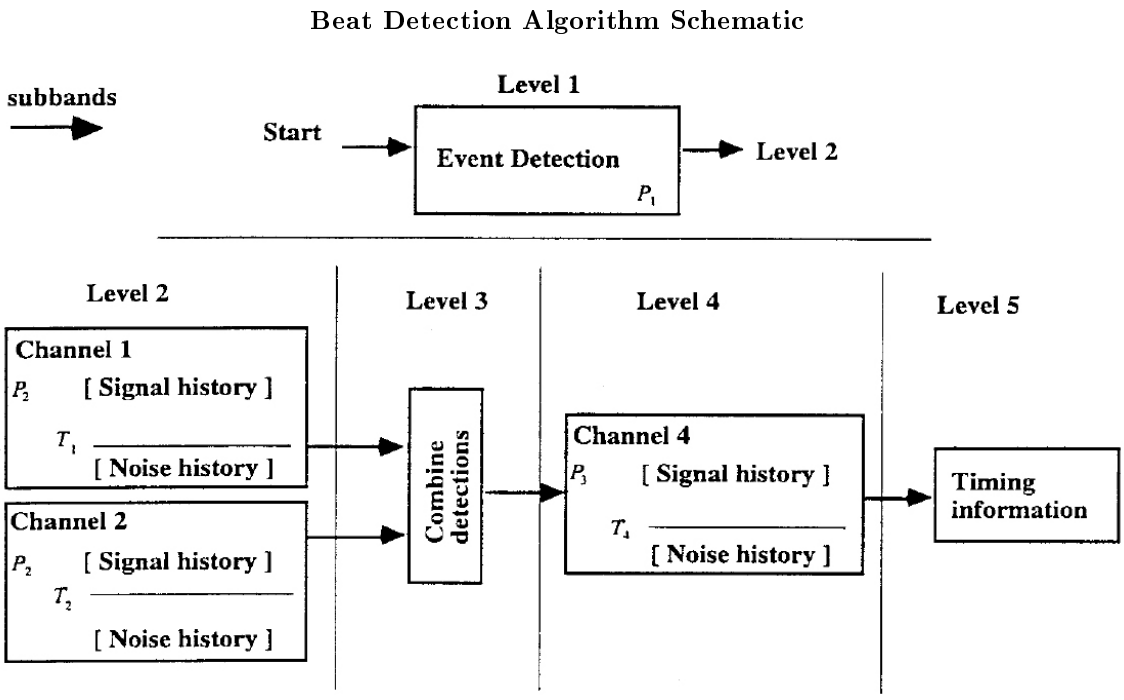


Figure 2.14

A one-channel beat detection block is described in Figure 5.

Schematic of One-Channel Beat Detection Block

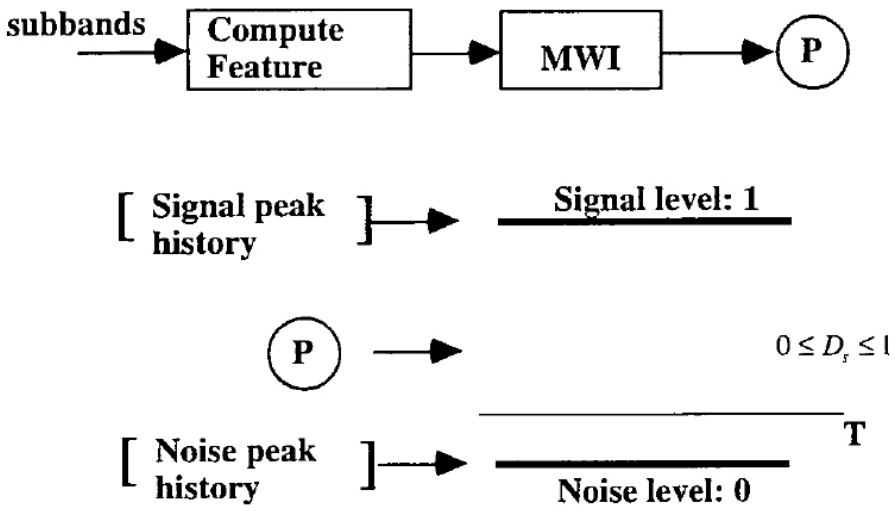


Figure 2.15

The detection strength Ds of an incoming feature (e.g., P1, P2, P3) is determined by comparing with the signal and noise levels (SL and NL, respectively):

$$D_s = \frac{P - N_L}{S_L - N_L}$$

Figure 2.16

If a feature's value is less than NL then Ds is limited at 0, and if it is above SL then DS is limited to one. When a feature has a greater than a specified threshold (preset between zero and one) it is classified as a signal peak and the signal history is updated with the feature's value. If the feature has a smaller than the threshold it is classified as a noise peak and the noise history is updated with the feature's value.

The detail operations of each level can be found in the paper by V. Afonso et al (*ECG beat detection using filter banks, IEEE Trans. Biomed. Eng. 46: 1999*). Also, download and run ECGmain.m[8] to test out filter banks. Supplementary mfiles: nqrsdetect.m[9] , ECGSigProc.m[10] , t.mat[11] , x.mat[12] .

[8] http://cnx.org/content/m18957/latest/ECGmain.m
[9] http://cnx.org/content/m18957/latest/nqrsdetect.m
[10] http://cnx.org/content/m18957/latest/ECGSigProcFB.m
[11] http://cnx.org/content/m18957/latest/t.mat
[12] http://cnx.org/content/m18957/latest/x.mat

2.3.3 LabVIEW Programming for Signal Processing and Interfacing

LabVIEW is used to not only process and acquire the signal, but to also develop an user friendly interface that automatically alerts the user to any abnormalities or arrhythmias detected. This particular ECG is developed to detect left ventricular hypertrophy and old myocardial infarction. It will also automatically display the heart rate and whether or not the patient's heart rate is within a healthy range.

The Graphical User Interface

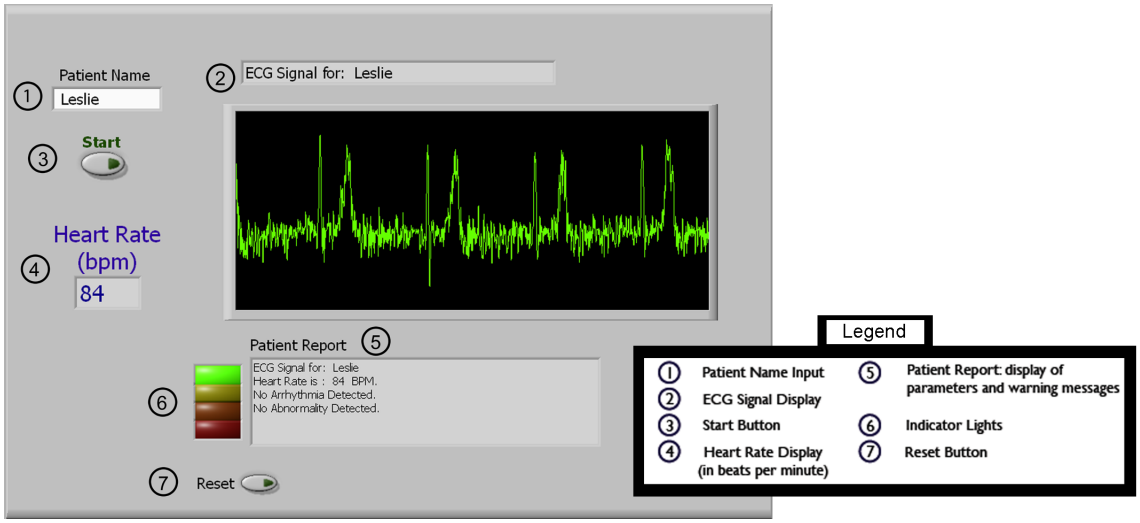


Figure 2.17: This view of the the GUI shows a normal, healthy ECG signal. The indicator lights (6) will change color depending on the diagnosis. Additionally, the green trace turns red if any abnormality or arrhythmia is detected.

The user interface consists of several useful features. First, there is a text input for the patient's name to avoid confusion in the clinic. Second, the ECG chart is automatically updated with the name and live signal data. Third, the user can control the data acquisition periods. The heart rate and any abnormalities are displayed in the patient report. The warning lights to the left of the patient report correspond to the current condition of the patient's ECG. If everything is normal, the green light is on and the signal is green. If an arrhythmia is detected, the yellow light turns on and a warning message will appear in the patient report that indicates the type. Similarly, the orange light turns on if an abnormality is detected and a warning message appears in the report. For either of these, the ECG trace turns red in the chart window. A red light indicates that no recognizable signal is being detected. This can mean that the patient has flat-lined, that the circuit board is not powered or that the leads are not connected properly. As a result, the warning message will ask the user to check the lead and power connections. Finally, the form can be reset between patients. You can download our VI here[13] to test it out.

[13] http://cnx.org/content/m18957/latest/ECG_LIVE.vi

Chapter 3

Discrete Multi-Tone Modulation Using a String Can System

3.1 Introduction[1]

3.1.1 Why Discrete Multi-tone Modulation?

Discrete multi-tone modulation, or DMT, is a modulation method often used for transmitter-receiver systems that have extremely poor channel characteristics. In our case, we are using a string and tin-can telephone; one of those toys that you may know as a childhood novelty where two people can communicate with each other by attaching the ends of a string to the bottoms of two tin cans and having one person speak into one can while the other listens. The string used to communicate between the two cans is a very bad channel for data transmission, which is why you cannot always distinctly understand what someone is trying to say in a tin-can telephone. That's where DMT should come in.

3.1.2 What is DMT?

This modulation scheme uses frequency-division multiplexing of several sub-carriers that each carry a portion of the data that is intended for sending across the channel. The sub-carriers are not far from each other in frequency, but they are all orthogonal so they do not interfere. The sub-carrier itself can be modulated using any desired scheme as would be done for a single-carrier modulation scheme. These sub-carriers are then sent together across the channel.

3.1.3 What are the advantages of DMT for us?

The DMT method is simply slowly modulating narrowband signals as opposed to quickly modulating a wideband signal, meaning channel equalization is simplified. Then if we find out the passband for our string channel, which will probably be quite narrow, we can use DMT to make efficient use of such limited frequency space. DMT is also fairly immune to multipath and attenuation, which will be significant problems for our system as well.

3.2 System Specifications[2]

- Used sewing thread string, considered the lightest material and therefore the most sensitive to vibrations; also tried using copper wire and fishing line

[1] This content is available online at <http://cnx.org/content/m18960/1.1/>.
[2] This content is available online at <http://cnx.org/content/m18961/1.2/>.

- Instead of tin cans we used paper cups, since the paper diaphragm on the cup's bottom is more mobile than that of the tin can, meaning it can better drive the string's vibrations; also considered plastic cups
- Each cup is raised off the ground on a wooden stand several feet tall, to make sure the audio that leaks from the system does not bounce off the floor and hit the system directly again, causing multipath
- The stands also allow us to gauge the length at which we keep the cups separated, as the distance determines the tension in the string in between them, and any change in the string's tension has a tremendous effect on the impulse response, which we cannot afford to change since we are modulating the sub-carriers over very specific frequencies
- Styrofoam is used to separate the cups from the wooden stands, so that the cups do not vibrate the whole stand as much
- The receiver cup with the microphone placed in it is padded with foam as well to keep multipath effects at a minimum
- Speaker used for the transmitter had an output diameter that matched the diameter of the mouth of the cup to minimize leakage of audio from the transmitter

Figure 3.1

3.3 DMT Transmitter[3]

The transmitted signal consists of a sum of sub-carriers with data on each carrier being modulated independently using BPSK

3.3.1 Outline for Transmitting Text Over Channel

1. Convert text to ASCII and ASCII to binary (1 and -1): In order to convert text to ASCII in Matlab, use the int8() function. Once we have the ASCII values, the next step is to convert these values into

[3]This content is available online at <http://cnx.org/content/m18984/1.1/>.

binary. The first step is to convert the int8() output values into values with an output in double format. This can be done by using the double() function in Matlab. Once the values are in double format, use the de2bi() function to convert them to binary values. Now we need to convert all 0s in this binary number into -1s. This allows for easier phase correction at the receiver end.To do this, assign this vector to x and perform the function y = 2.*x-1

2. Transpose and reshape matrix to combine all rows into 1 row vector: Originally, each column will represent a different letter. To get each row to represent a letter, we transpose the matrix. Then we use the reshape() function to convert the matrix into one row vector.
3. Create another symmetric row and take their IFFT: This is a neat trick that can be employed to simplify the transmitter. If we take the IFFT of two symmetric rows instead of one row, the result will be purely real. Therefore, we will only need to transmit and decode one vector instead of two. To produce these vectors, take the original vector and use the shiftlr() function. Use the output of this function as the second row in a 2 row matrix. The first row will be the original row. Take the IFFT of this matrix.
4. Add cyclic prefix of same length as impulse response: The cyclic prefix is described in more detail under Channel Characteristics. For now, create a vector of zeroes that is as long as the non-zero component of the impulse response of the channel and add it to the beginning and end of each symbol (block).
5. Combine rows and modulate to pass band frequency of channel: Make a row vector of the two symmetric rows and cosine/sine modulate them at the pass-band frequency
6. Make .wav file and add impulses before and after signal to synchronize: Use the wavwrite() function in Matlab to convert the row vector into a .wav file. Introduce short bursts before and after the signal to mark the beginning and end of the signal.The block diagram below provides a basic layout of the transmitter employed:

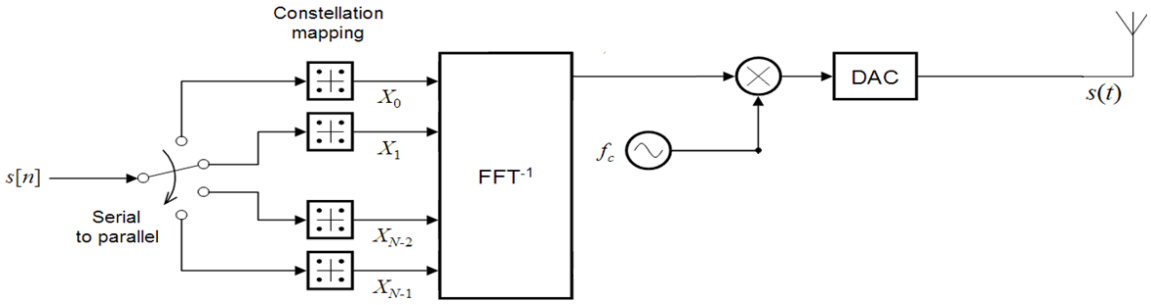


Figure 3.2

3.4 Channel Characteristics[4]

3.4.1 Channel Selection

The channel for our system was chosen after a lot of research into the kind of qualities required by the specific goal of sending data over the channel. We tried various materials such as a fishing line, copper wire and various types of threads, and tested each of them on the system to find out which one of those materials

[4]This content is available online at <http://cnx.org/content/m18971/1.1/>.

would be best to carry longitudinal vibrations (of the transmitted wav-file) successfully and with the least amount of error.

We decided that the material needs to be lightweight and not stretchy – hence sewing thread would work great for our system. The longer the string is, the more lossy it will be – so we moderated the length of the string we used to about 35 inches.

The sewing thread channel has specific parameters that should be taken into consideration when implementing the system. It is important to have an understanding of these specifications to allow for successful data transmission by reducing possible bit errors.

3.4.2 Channel Parameters

The parameters that we took into consideration are:

- Channel Capacity – the capacity of a channel is defined to be the upper bound of the maximum bit rate that can be sent over the channel without introducing errors. After taking the channel capacity into consideration, we realized that it would not be possible to send voice or images over the channel simply because these transmission types would require intensive bit rate, which the sewing thread channel is not capable of. Hence, we resorted to sending text over the channel.
- Multipath Effects – the signal being transmitted might take more than one path to reach the receiver, causing superposition/interference with the original signal on the channel. For example, the transmitted sound might travel through the air and superimpose with the signal on the channel, causing unnecessary complications. To reduce these multipath effects, we used a method called cyclic prefix.(Figure 1) What this does is that it takes a chunk (of the same length as the impulse response of the system) from the end of the bit-stream being sent over the channel, replicates it, and then prefixes the bit-stream with the same data. This essentially makes the channel perform circular convolution rather than the linear convolution it would normally carry out. Hence, this cyclic prefix allows the multipath to settle before the main data arrives at the receiver.

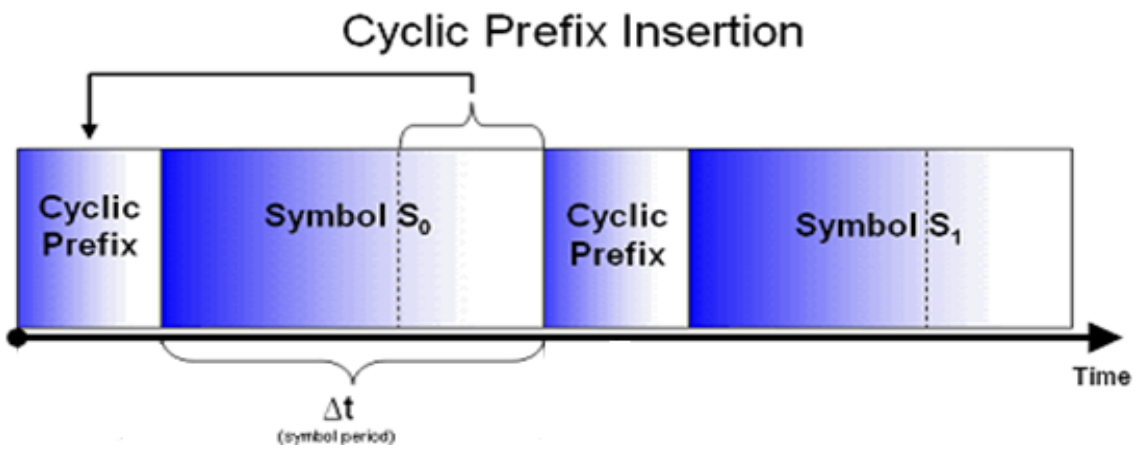


Figure 3.3

- Signal Distortion – this could be caused by noise interfering with the signal being transmitted. This introduction of noise could cause synchronization problems at the receiver end, since the receiver would

not be able to distinguish where it is just receiving noise from the actual signal being transmitted. Therefore, for synchronization, we sent two impulses before and after the actual data being sent over the channel. These impulses help the receiver know when the actual data is being received – starting right after the first impulse and ending right before the second impulse.

- Bandwidth Wastage – not using the entire range of frequencies available on the channel would obviously lead to wastage of its bandwidth, since a lot more data could be sent over the channel by modulating data at different carrier frequencies. This is essentially the idea of Frequency Division Multiplexing, which forms the basis of the DMT modulation depicted in this project. Hence, sending different blocks of data at different available frequency pass-bands would lead an efficient use of the channel bandwidth. In our project, we didn't need to send over multiple frequencies since the amount of data sent could easily be modulated within one pass-band – specifically the one band between 160Hz to about 320Hz.

3.5 DMT Receiver[5]

The received signal is an analog, modulated version of the original signal which has been transmitted by multiple carriers. The receiver is just an inverted version of the transmitter described earlier.

3.5.1 Outline for Receiving Text After Transmission

1. Convert .wav file into a row vector, remove impulses and modulate: Once the signal has been received by the receiver end (see System Specifications on how to do this), the first step is to convert it into a row vector. This can be done by using the wavread() function in Matlab. After this we need to remove the impulses before and after the sent signal. The easiest way to do this is to plot the received signal and then determine a threshold above which only the values for the impulse exist. Truncate the vector to the same length as the original vector without impulses by removing all values greater than the threshold set by analyzing the plot. After retrieving the original data back, cosine/sine modulate again to return to retrieve the original signal.
2. Low pass filter to retrieve original signal and remove some noise Design: LPF filter in Matlab.
3. Remove the cyclic prefix: Remove vector of length equal to the impulse response from the beginning and ending of the current vector. NOTE: These vectors may not be purely zero anymore because of introduction of noise
4. Find one period in the received signal by comparing the complete received signal with the length of the original signal: We need to find a segment of the same length as the original vector that we received after taking the IFFT in the transmitter. This vector should also be approximately periodic.
5. Take the FFT of this signal: This should return approximately the same vector that was the input into the IFFT on the transmitter end.
6. Create thresholds to round values to -1 and 1: This serves two purposes. First, we are now back to the -1/1 values that we had originally when we transmitted our signal. Also, since -1 and 1 represent two very distinct phases, we can assume are values less than 0 to be -1 and all values greater than 0 to be 1. This corrects for the phase errors that may have been introduced by the channel.
7. Repeat steps 4-6 for multiple received periods: This is to minimize the effects that noise may have had on any given block of data and to correct for any phase inversions that may have occurred.
8. Find two rows that are symmetric: Convert all -1 to 0 of first rowDo this by implementing the function (x+1)./2 where x is the original vector
9. Convert values of first row to ASCII and map these values to text: Now we have a binary vector which can be converted back to decimal using the command bi2de() and once the decimal value is retrieved, we can map back to ASCII

[5]This content is available online at <http://cnx.org/content/m18983/1.1/>.

Block Diagram Implementation of Receiver

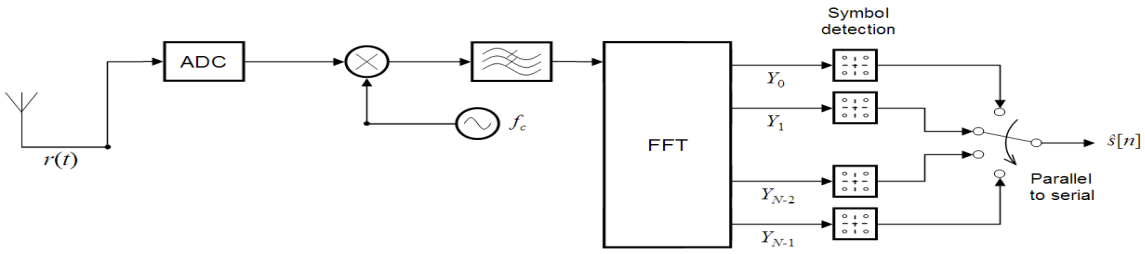


Figure 3.4

3.6 Key Results and Conclusion[6]

3.6.1 Key Results

The first implementation of our system involved sending an impulse over the sewing thread channel. The impulse wave was created by recording a quick flick on the microphone and then using the Sound Recorder on the computer to make a wave file out of it. This wave file can easily be plotted in MATLAB to see how the impulse looks like and if it actually produces a very high amplitude compared to the noise, in as little time as possible.

Once the impulse was received at the receiver cup, the microphone captured this sound and again this was saved as a wave file on the computer. This wave file can easily be read into MATLAB as a vector using the wavread function. The next step is to take the FFT of this impulse response to figure out the transfer function of the channel. The following plot shows you how the transfer function (frequency domain representation of the impulse response) looked like.

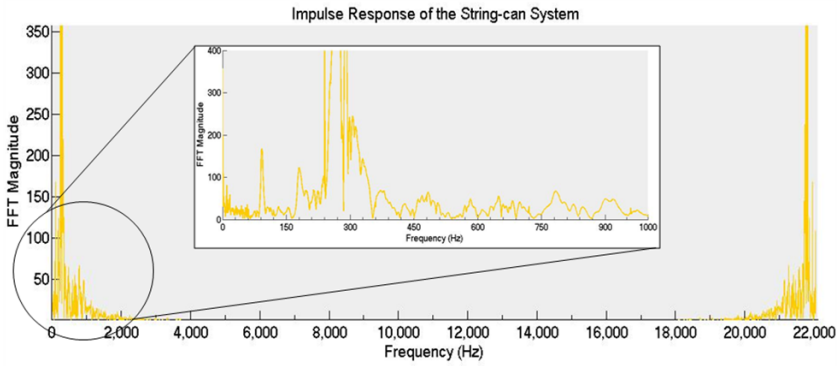


Figure 3.5

[6]This content is available online at <http://cnx.org/content/m18973/1.1/>.

As you can see in the figure above, there is a major pass band at about 250Hz. This is exactly what we chose to send our data stream over the channel. A closer look would reveal that there are actually several other smaller peaks in the transfer function too, which could possibly be used to modulate data at different frequencies, and hence utilize the Frequency Division Multiplexing scheme.

The actual data stream that we sent over the channel involved modulating at 250Hz with a sine wave of an appropriate period. The modulated sine wave is shown below. Each point on the wave represent one data bit that we were sending over the channel (0s, 1s and -1s).

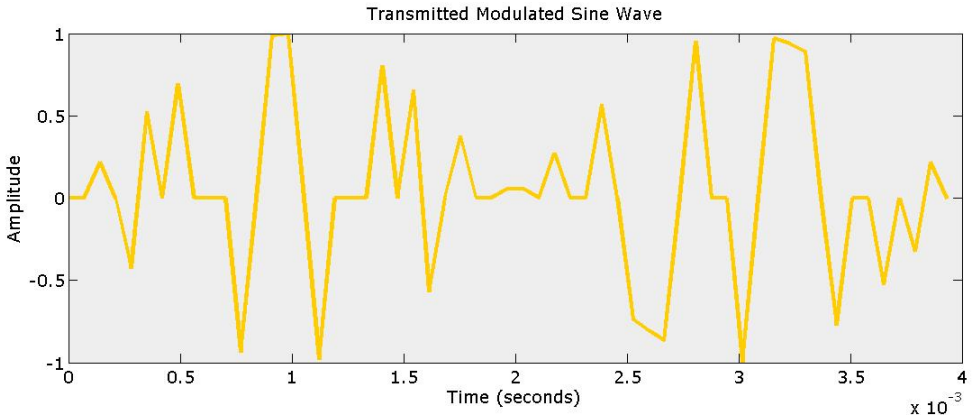


Figure 3.6

Also, to reduce the amount of bit errors at the receiving end, we sent multiple replications of the data stream one after the other. This allowed us to compare multiple strings of data at the receiver and then optimize the choice of the string that best represented the data at the transmitting end.

The spectrogram below will help you visualize the signal we sent over the channel.

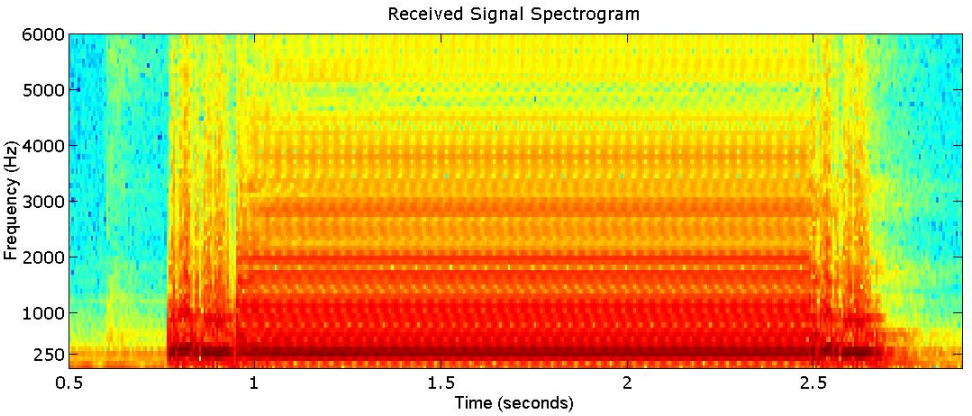


Figure 3.7

The figure above helps us prove and reiterate that the data we sent over the channel was in fact modulated at 250Hz – the horizontal dark red line at 250Hz suggests the major presence of that particular frequency in the transmitted data. Besides that, the multiple vertical sections that you see in the center of the spectrogram represent the data bits that are being sent over the channel at that particular instance. The presence of multiple sections indicates replication of the data bits, as I mentioned before.

Another important feature of this spectrogram is the presence of two impulse signals before and after the data bit stream. It is clearly visible that impulses of the same kind have been sent along with the actual data to help synchronize the receiver with the transmitter (as explained in the Channel Specifications module).

3.6.2 Conclusions

Here are some of the important conclusions and take aways from our project:

- DMT modulation allows us to transmit the word "Hello" over a channel as inconsistent as a string by taking different channel characteristics such as channel capacity, frequency pass bands and noise introduction into account before transmitting data
- Efficient data transmission fully exploits the bandwidth of the channel by using frequency division multiplexing by sending small, parallel blocks of data on orthogonal carriers

- Synchronization is one of the major challenges of using DMT modulation. One method of synchronizing the transmitted and received signals is to send loud bursts before and after the relevant data. This allows us to mark the start and end of the data in the received signal

3.7 The Project Team[7]

Jaimeet Gulati - Electrical and Computer Engineering, Rice University
Ryan Pei - Electrical and Computer Engineering, Rice University
Shamoor Anis - Electrical and Computer Engineering, Rice University

3.8 Acknowledgements and References[8]

3.8.1 Acknowledgements

We would like to thank Dr. Richard Baraniuk and Dr. Don Johnson from the Rice University Department of Electrical and Computer Engineering for their unrelenting support throughout the course of this project.

3.8.2 References

- DIRECTIVE 95/47/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on the use of standards for the transmission of television signals
- ETSI Standard: EN 300 744 V1.5.1 (2004-11).
- http://www.commsdesign.com/article/printableArticle.jhtml?articleID=12805708 Agere gets Sirius about satellite radio design
- http://www.citizens.coop/aboutus/newsreleases/TrulyMobileWireless.pdf
- Nortel 3G World Congress Press Release
- http://en.wikipedia.org/wiki/OFDM

[7] This content is available online at <http://cnx.org/content/m18970/1.1/>.
[8] This content is available online at <http://cnx.org/content/m18982/1.1/>.

Chapter 4

Estimating Target Location Using 2-D Projective Geometry and Background Subtraction

4.1 Introduction to Estimating Target Location Using 2-D Projective Geometry and Background Subtraction[1]

Project Intro-Summary
Our ideal objective of this project was to build a surveillance system that can detect any moving object and locate its exact position. First, We used background subtraction in order to segment out only moving object from the background. Next, we came up with an idea of using projective geometry to map the frame from camera's point of view onto the world plane. When one camera was used to locate an object, the results were sometimes misleading (i.e. could not distinguish height and depth). By using multiple cameras instead, we were able to obtain a more precise location of the object. This project was more focused on detecting the changes in the background and locate its position, rather than finding the object's shape. Thus, each camera had equal weights on providing data. In the next modules, we are going to present how we used background subtraction and 2-D projective geometry in order to build our simple surveillance system.

4.2 Background Subtraction[2]

4.2.1 Background Subtraction

What is Background Subtraction?
Background Subtraction is a process to detect a movement or significant differences inside of the video frame, when compared to a reference, and to remove all the non-significant components (background). Background subtraction is applied in many areas, such as surveillance system (to effectively segment the only moving object).

Steps to implement background subtraction

- Learning Background – we captured ten background frames and calculated the mean(μ) and the standard deviation(σ) with the below equations
- We assumed the value of the background was iid-normal distribution

[1] This content is available online at <http://cnx.org/content/m19011/1.1/>.
[2] This content is available online at <http://cnx.org/content/m19013/1.1/>.

$$\mu = \frac{1}{n}\sum_{i-1}^{n} x_i \quad \sigma = \sqrt{(\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Figure 4.1: calculation of mean and standard deviation. xi is the value at a particular pixel of ith image

Processing Data (real time) – Distinguishing Background and Non-Background Objects

- First, we set the threshold of background as $\mu \pm 2\sigma$
- Each pixel in the background -> N $\sim$ (μ, σ), if that pixel is part of the background, its value will lie within $\mu \pm 2\sigma$ range 95% of the time.
- Any pixels that go beyond the threshold are considered as parts of non-background object.

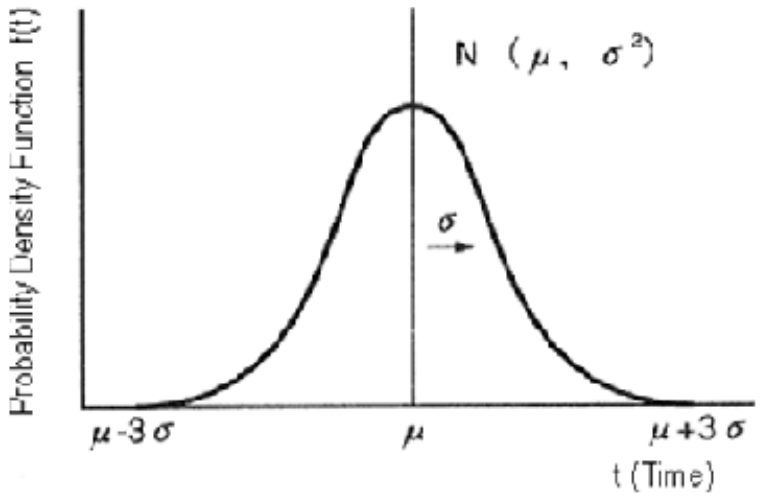


Figure 4.2: Normal distribution of background pixels

The Below figures are the result of implementing background subtraction

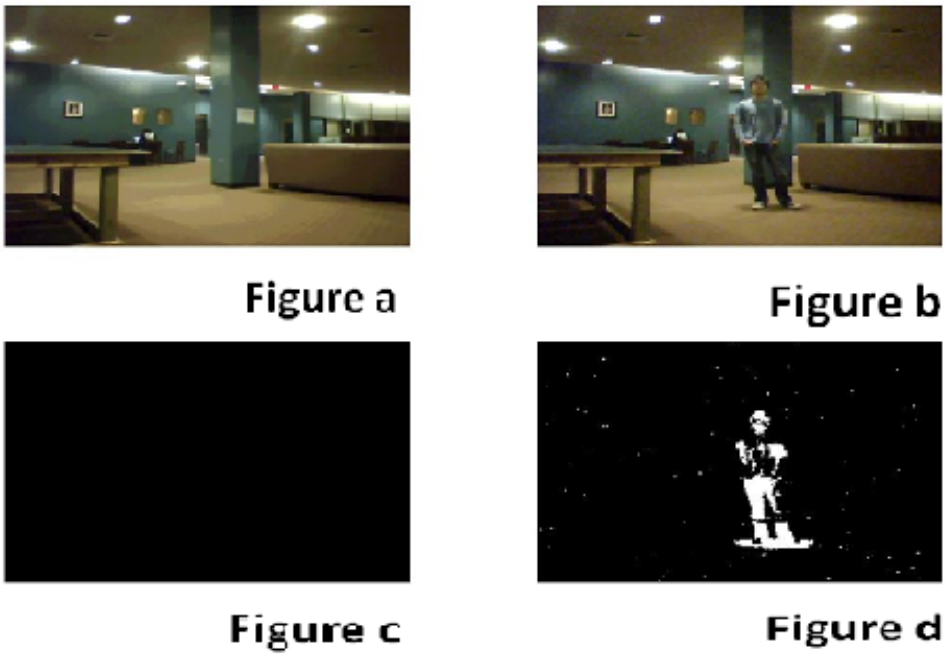


Figure 4.3: Figure a is the background frame we set. When non-background object is introduced (figure b), we can see the object is segmented out (figure d).

4.3 2-D Projective Geometry[3]

4.3.1 2-D Projective Geometry

What is Projective Geometry

Projective Geometry was developed in the 17th century by Girard Desargues, a French mathematician. It is a type of geometry that is based upon the principles of perspective art (approximate representation of an image perceive by the eye on a flat surface). It deals with the properties and invariants of geometric figures under projection-any mapping of points in the plane that preserves straight lines.

Steps to Implement 2-D Projective Geometry

In our case, we performed 2-D Projective Geometry on the image taken by a camera. We defined the location of the object we see in the camera image as the location on "camera plane" and the actual location of the object as the location on "world plane". This is shown in figure 1 below.

[3]This content is available online at <http://cnx.org/content/m19014/1.1/>.

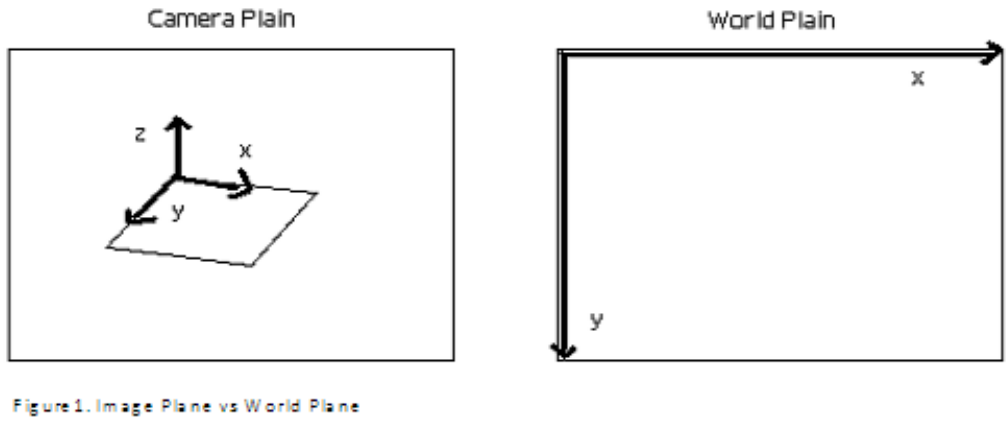


Figure 4.4: Image Plane vs World Plane

The first step to implement 2-D Projective Geometry (i.e. maps camera plane onto world plane) is to calibrate many parameters of our located camera. This step is to find the location of the camera (distance + angles) from our designated world plane. This is essential to perform required calculations in the next step. We calibrated necessay parameters by running a program called "Camera Calibration tool" provided by Caltech. Details about this program can be found on its supporting website:

www.vision.caltech.edu/bouguetj/calib_doc/[4]

Once we run the program to locate the camera's position, we will obtain a result that looks similar to the two figures below:

[4]http://cnx.org/content/m19014/latest/www.vision.caltech.edu/bouguetj/calib_doc/

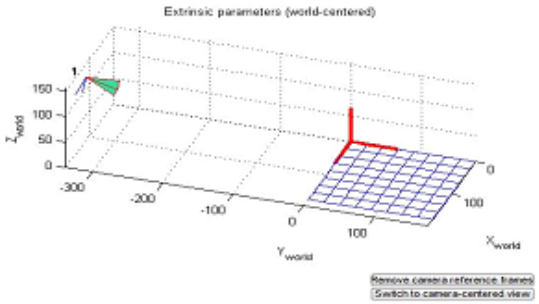


Figure 2. Camera Location - World Centered

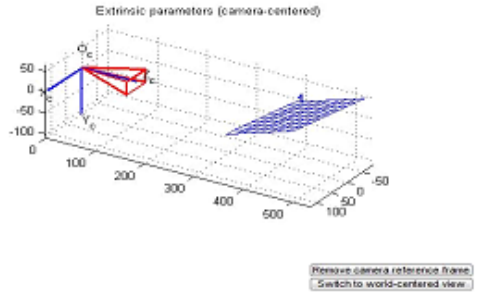


Figure 3. Camera Location – Camera Centered

Figure 4.5

Besides the 2 figures above, we will also obtain all data necessary in the mapping process. These includes:
- Focal length, fc: measurement of how strongly camera converges or diverges light
- Principal Point, cc: the point where camera plane intersects the axis

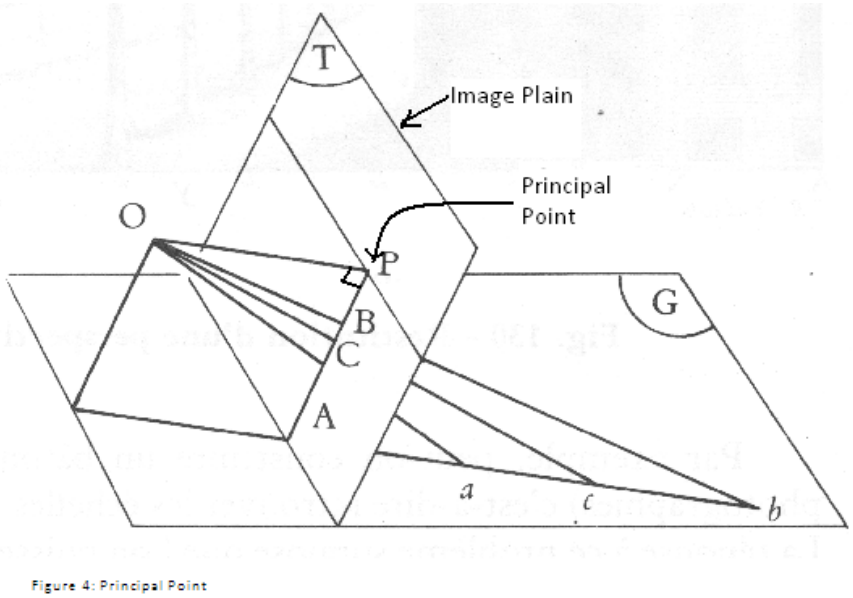


Figure 4: Principal Point

Figure 4.6: Principal Point

- Skew Coefficient, α

- Distortions (image distortion coefficients), kc
- Camera Matrix, KK

$$KK = \begin{bmatrix} fc_x & \alpha * fc_x & cc_x \\ 0 & fc_y & cc_y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 4.7

Coordinate of the origin of the camera reference frame, Tc_1 - Surface normal vector of the plane of camera reference frame, Rc_1

Finally once we obtain all the necessary information, we are now ready to begin the mapping process:The mapping process is as followed (we also obtained these steps from Caltech):

- Let P be a point space of coordinate vector XX = [X;Y;0] in the world plane (point on the right picture of figure 1)
- Let XXc = [Xc;Yc;Zc] be the coordinate vector of P in the camera reference frame. Then we will get the relationship:

$$xx_c = [x_c \quad y_c \quad z_c]' = Rc\_1 * xx + Tc\_1$$

Figure 4.8

- Let xn be the normalized image projection:

$$x_n = \begin{bmatrix} x_c / z_c \\ y_c / z_c \end{bmatrix}$$

Figure 4.9

- Also let the square of the distance:

$$r^2 = x^2 + y^2$$

Figure 4.10

- After including lens distortion, the new normalized point coordinate xd is defined as follows:

$$x_d = \begin{bmatrix} x_d(1) \\ x_d(2) \end{bmatrix} = (1 + kc(1) * r^2 + kc(2) * r^4 + kc(5) * r^6) * x_n + dx$$

Figure 4.11

Where dx is the tangential distortion vector:

$$dx = \begin{bmatrix} 2 * kc(3) * x * y + kc(4) * (r^2 + 2 * x^2) \\ kc(3) * (r^2 + 2 * y^2) + 2 * kc(4) * x * y) \end{bmatrix}$$

Figure 4.12

- Once distortion is applied, the final pixel coordinates x_pixel = [xp;yp] of the projection of P on the image plane is:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = KK * \begin{bmatrix} x_d(1) \\ x_d(2) \\ 1 \end{bmatrix}$$

Figure 4.13

After we follow all the above steps, we will obtain the result as in figure 11. In this case, the process above maps chess board in the photo (left buttom figure) onto chess board on the world plane (right buttom figure). When the image of the car is mapped on to the world plane, it looks deformed. This is because the camera cannot tell the difference between the height and the depth of the object on the image plane. Therefore in this process, everything is assumed to be flat (zero height). This will become a problem later as we would not be able to locate exact position of the object.

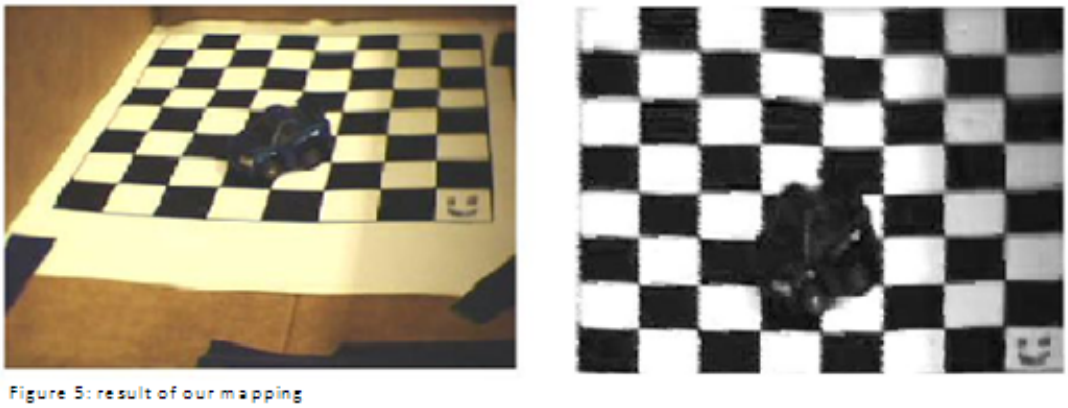


Figure 4.14: Result of our mapping

4.4 Combining Results - Background Subtraction + Projective Geometry[5]

4.4.1 Combining Our Results – Projective Geometry + Background Subtraction

Combining Our Results

By combining the result from our two parts, it is possible to tract the location of the object. As mentioned earlier, we need to use two different camera capturing image from different angle. This is because using one

[5]This content is available online at <http://cnx.org/content/m19012/1.1/>.

camera cannot differentiate depth and height. So by using two cameras, we can superimpose the two images (of world plane) and find an accurate location of the object(See figure 1 for result).

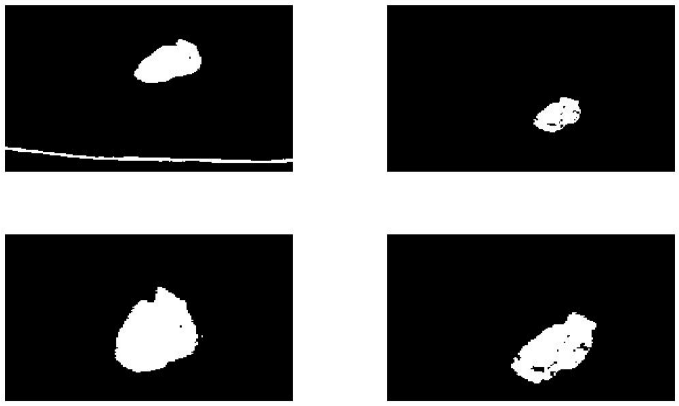


Figure 4.15: Result after we combine the two processes (from 2 different cameras)

4.4.1.1

We represent our result in the form of Weight Particles plots (shown in figure 2). By summing the two plot above (c, d), square this resulting sum and divide it with the sum of all the value in the matrix -> matrix represents the weights of the final particles. If you plot this matrix, you will get the plot (as below) that represents the probability distribution of the target in x-y space.

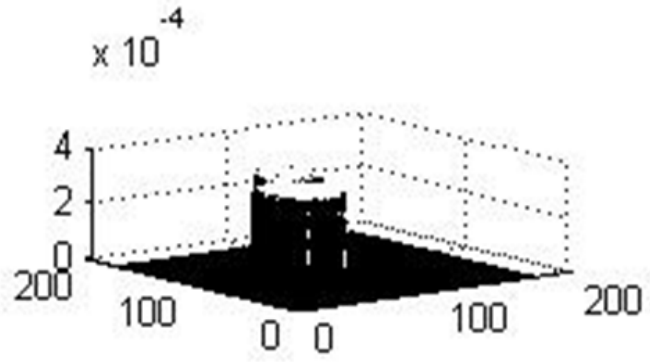


Figure 4.16: Weighted Particles Plots

Chapter 5

Frequency Analysis for Art Forensics

5.1 The Team[1]

5.1.1 The Team

Students

Lucia Sun: ls4@rice.edu[2]

"露露" developed most Matlab codes for the project. Her GPA is greater than 4.0...

[1] This content is available online at <http://cnx.org/content/m18949/1.3/>.

[2] ls4@rice.edu

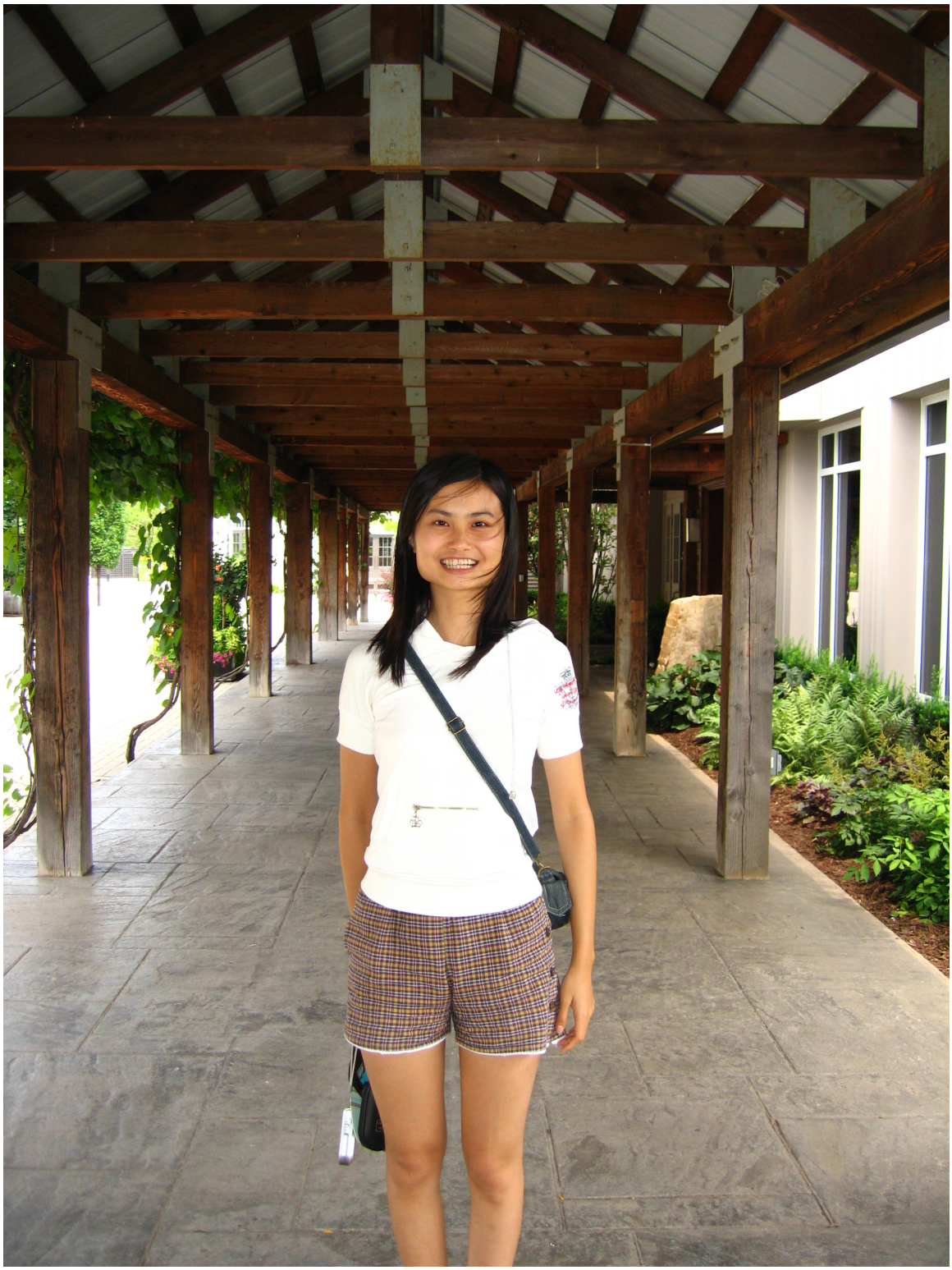


Figure 5.1

Zeting Liu: zbl1@rice.edu[3]

"ミ丁ミ丁" is our poster master. He is an exchange student from a Hong Kong university. Oops, I always forget its name, should be something like HKUST (Hong Kong University of Science and Technology). He is also the CEO of a brilliant website http://www.boliance.com[4] /.

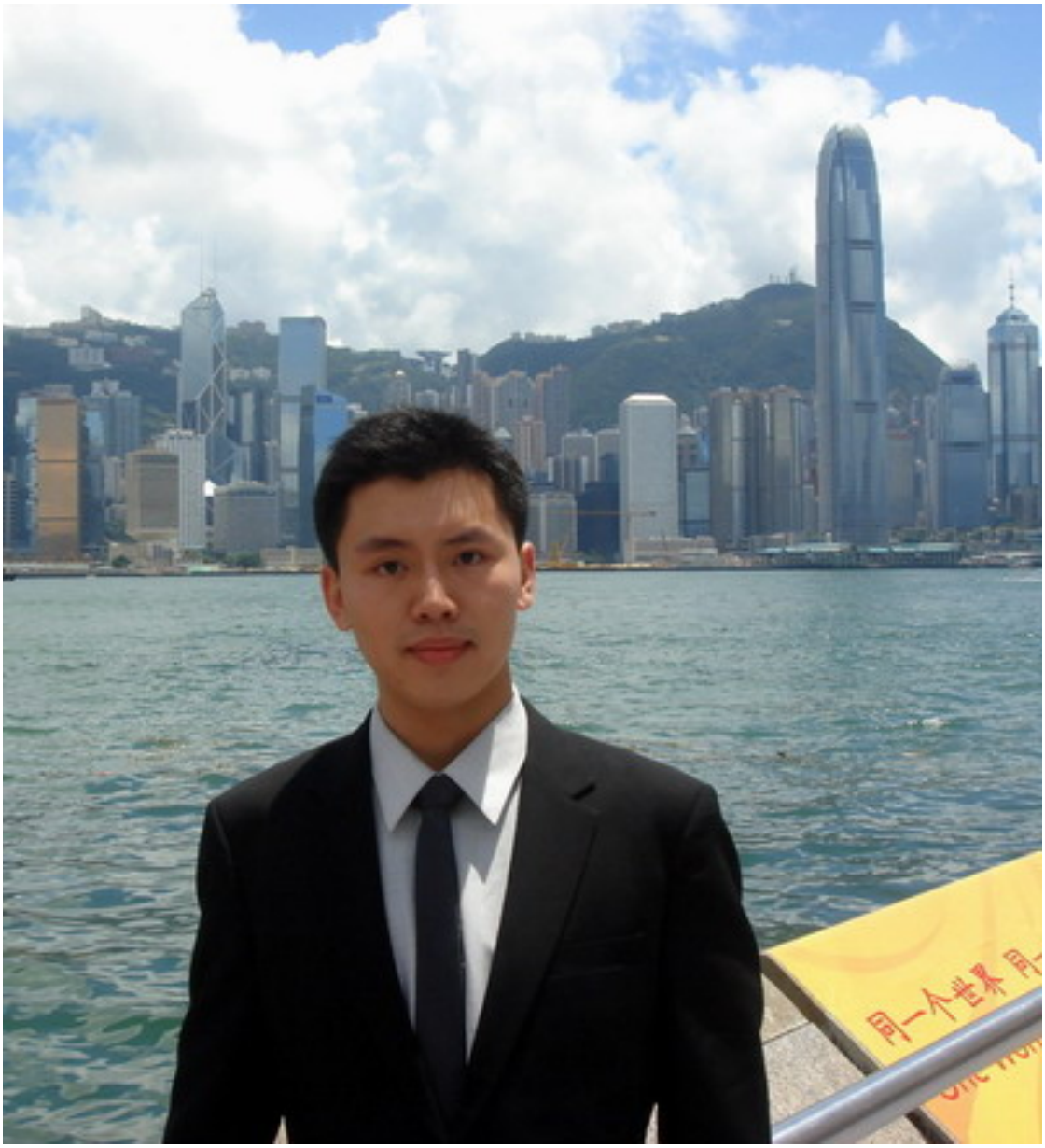


Figure 5.2

[3] zbl1@rice.edu

[4] http://www.boliance.com/

Xiang Guo (Jash): xg1@rice.edu[5]

"享享" Jash did a lot of things including bringing in snacks and treating the team for a buffet dinner. The most important thing he learned in 301 is "Totally!"

Figure 5.3

Mentor

Professor Don H. Johnson: dhj@rice.edu[6]

He is the BIG Guy in DSP. You know what I mean...

[5] xg1@rice.edu
[6] dhj@rice.edu

Figure 5.4

5.2 Introduction[7]

5.2.1 Abstract

A canvas can be characterized by the vertical and horizontal weave densities while the actual painting serves as an additive signal that only distracts from the thread-counting process. The thread counting algorithm and the spectral techniques we employ in our project can analyze weave density for entire paintings with an accuracy comparable to human measurements more efficiently.

5.2.2 Motivation

Van Gogh Museum of Amsterdam has a collection of artist's works and is looking for a more advanced analysis for sequencing paintings in addition to the traditional manual methods. Our whole-painting analysis shows that frequency distributions should match if two paintings are from the same canvas roll. This could better support the forensic evidence quantitatively when comparing two paintings.

5.2.3 Approach

5.2.3.1 Current Approach

Human measurement – weave threads are manually counted and compared. This approach is inefficient and may not be accurate.

5.2.3.2 Our New Approach

Use thread counting algorithm and spectral techniques to analyze weave density for entire paintings. Frequency distributions of weave densities should match if two paintings are from same canvas roll.

[7] This content is available online at <http://cnx.org/content/m18943/1.1/>.

5.3 Background[8]

5.3.1 Warp and Weft

Warp – the vertical threads mounted in a loom. Warps are usually well aligned with a fairly uniform spacing.

Weft – the horizontal threads mounted in a loom. Wefts are usually threaded back and forth through the warp in an interlaced fashion. Therefore the weft shows more variability than the warp.

5.3.2 Vertical vs. Horizontal

An artist may orient the canvas on the stretcher in whatever way once he or she cuts a piece of canvas fro a painting. But we could expect the thread count having the narrower distribution to be the warp direction.

In our model, the vertical threads create oscillations of x-ray intensity in the x direction, which leads to a horizontal frequency component. The similar idea applies to the horizontal threads and the vertical frequency.

5.3.3 Canvas Texture Modeling

Canvas texture can be modeled as a sum of two sinusoids having nearly orthogonal spatial frequencies.

c(x) = p(x) + A/2 * [(2 + ahsin (2πfh*x + θh) + avsin(2πfv*x + θv)]

The quantity c(x) is the canvas x-ray image that depends on the 2D spatial variable x; p(x) represents the artist's contribution (the painting) to the x-ray. The constants A, ah and av determine the average intensity and the amplitudes of the horizontal and vertical weave. fv and fh are the vector frequencies corresponding to the vertical and horizontal thread counts, respectively.

5.3.4 X-Ray Image

Our project uses X-ray images of a painting as the raw data.

The thread used to make a canvas is transparent to x-rays. Fortunately, artists usually prepared their canvases with a white undercoat to smooth the surface. The small variations in undercoat thickness filling the valleys of the canvas weave lead to variations in x-ray opacity that can be measured. The greater the radiographic-absorbing paint thickness along the bean, the greater the opacity, meaning that x-ray image intensity variations correspond to paint thickness variations.

Figure 2 is the X-ray image of Figure 1- van Gogh's **Portrait of an Old Man with a Beard** (F205/JH971).

[8] This content is available online at <http://cnx.org/content/m18945/1.1/>.

Figure 5.5

Figure 5.6

5.3.5 Weave Density

Thread counting algorithms seek the weave density, measured in threads/cm, within a swatch and to study how these counts vary throughout the painting.

5.3.6 Short-Space Spectrum

2-D Fourier transforms of small areas reveal isolated peaks at the proper vertical and horizontal frequencies.

A square section is extracted from the x-ray, the average subtracted and window applies to obtain spectral detail using the 2D Fourier transform.

Figure 3 represents a simple case: a 1" $\times$ 1" swatch from x-ray image and the detailed spectrum computed from a 0.5" $\times$ 0.5" square located in the upper left corner of the swatch. The wedges indicate areas where weave-related spectral peaks are found.

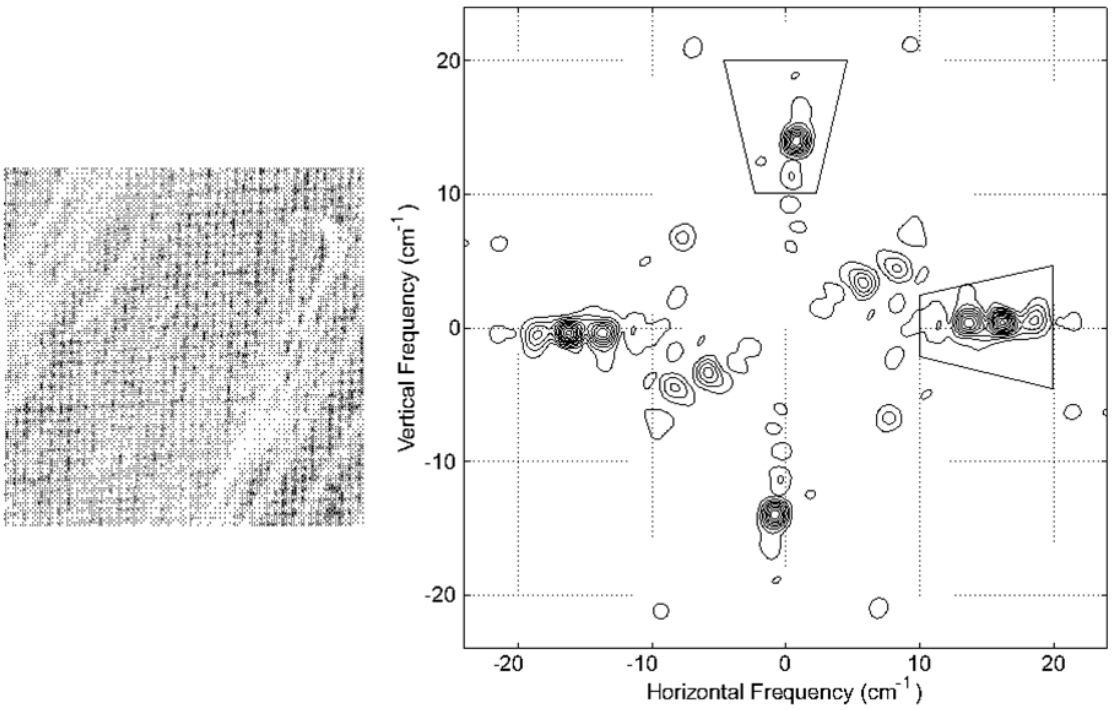


Figure 5.7

5.4 Methods and Results[9]

5.4.1 Whole Painting Analysis and Results

5.4.1.1 Step 1: Obtain the Raw Data

We received x-ray images of paintings from the Van Gogh Museum of Amsterdam. Images usually sampled at 600. Figure 1 shows the x-ray image of Van Gogh's `Backyards of Old Houses in Antwerp in the Snow` (F260) provided by the Van Gogh Museum.

[9] This content is available online at <http://cnx.org/content/m18946/1.1/>.

Figure 5.8

5.4.1.2 Step 2: Short Time Fourier Analysis

We apply shot time Fourier analysis for each 0.5" × 0.5" swatch. We discarded the outrange frequency peaks and set the value as NaN for the corresponding swatches. For multiple peaks in the frequency region of interest, we accepted the peak that is closest to the median value.

5.4.1.3 Step 3: Spectra of Whole-Painting

We sampled the short-space spectrum every 1/4" in both directions (horizontal and vertical) for the whole-painting by choosing swatches overlap each other by half in each direction. Thus the spectra of whole-painting were obtained. And we could determine the warp and weft direction of the canvas according to the spread of measurements. Calculations were made in Matlab and took about three hours to analyze F205 on a laptop computer.

Figure 2 shows the resulting spectra of F205.

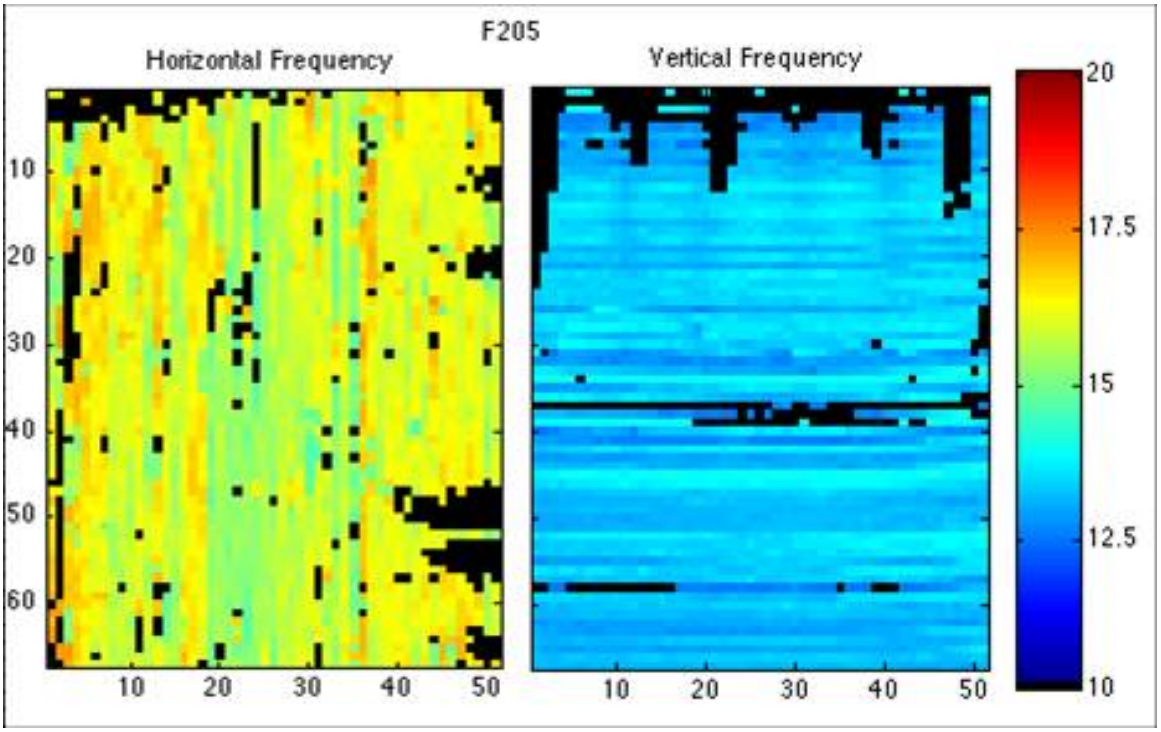


Figure 5.9

5.4.1.4 Step 4: Deviations Matching Analysis

From the spectra of the whole-painting, we obtained thread count deviations spectra by calculating the distributions of frequencies and subtracting the averages.

Figure 3 and 4 show the vertical thread count deviations of F205 and F260 respectively. Horizontal deviations spectra are not shown here.

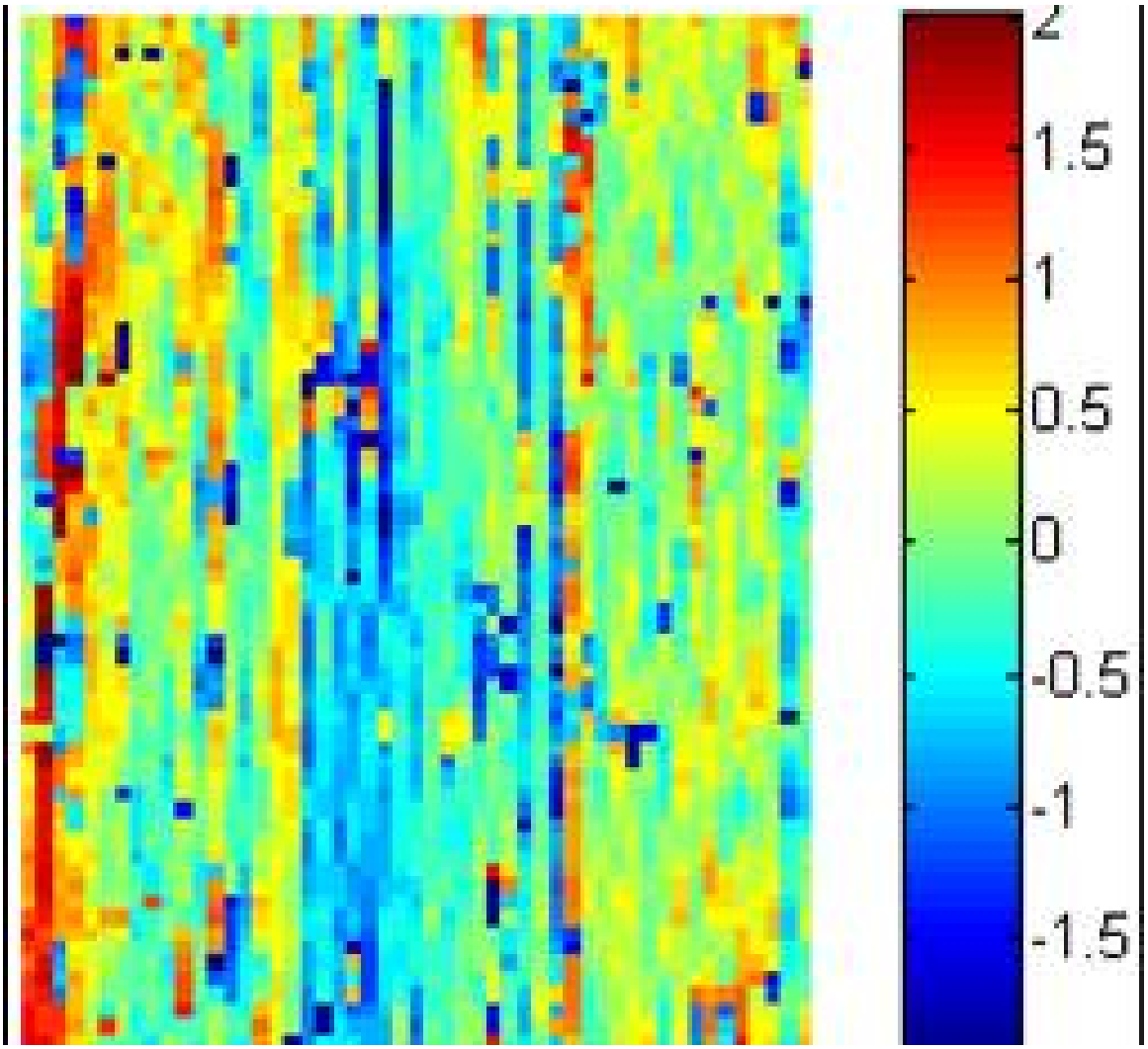


Figure 5.10

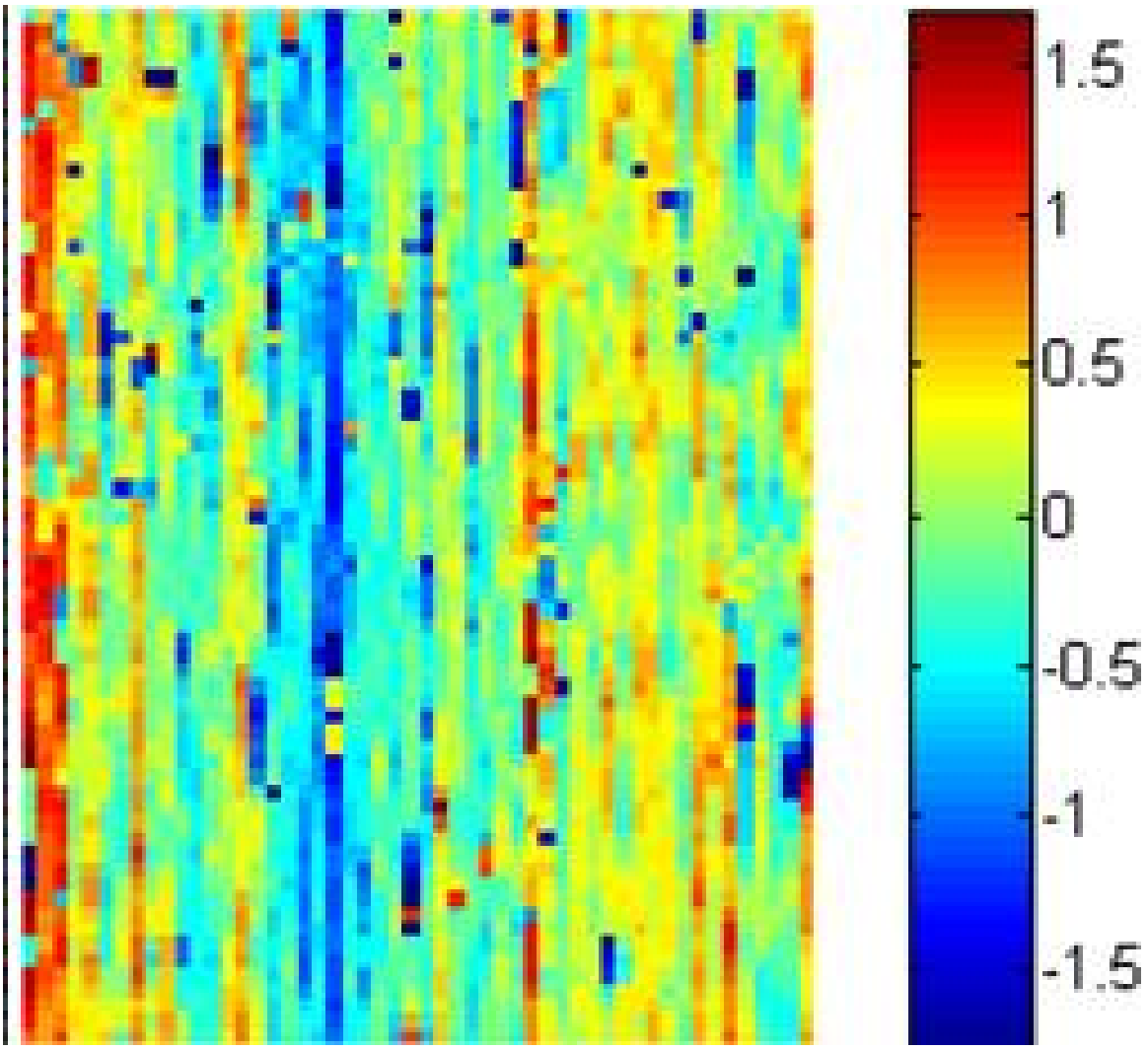


Figure 5.11

Visually, we can see the matching strips between these two painting. But how well do they match?

5.4.1.5 Step 5: 2D to 1D Conversion

We first computed the 1D thread count deviations from the 2D data. The 1D thread count deviations are obtained by summing the column deviations of 2D data while discarding all the NaNs if any.

Figure 5 and 6 are the corresponding 1D plot of F205 and F260.

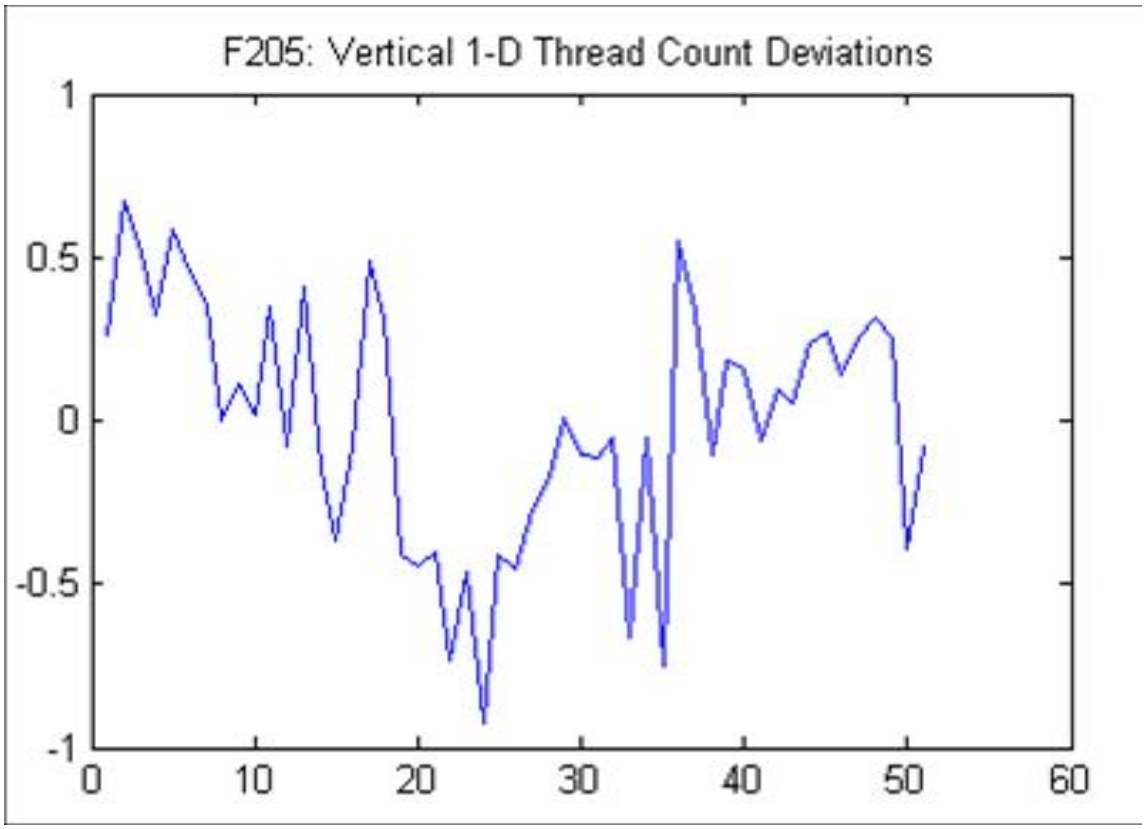

F205: Vertical 1-D Thread Count Deviations
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60

Figure 5.12

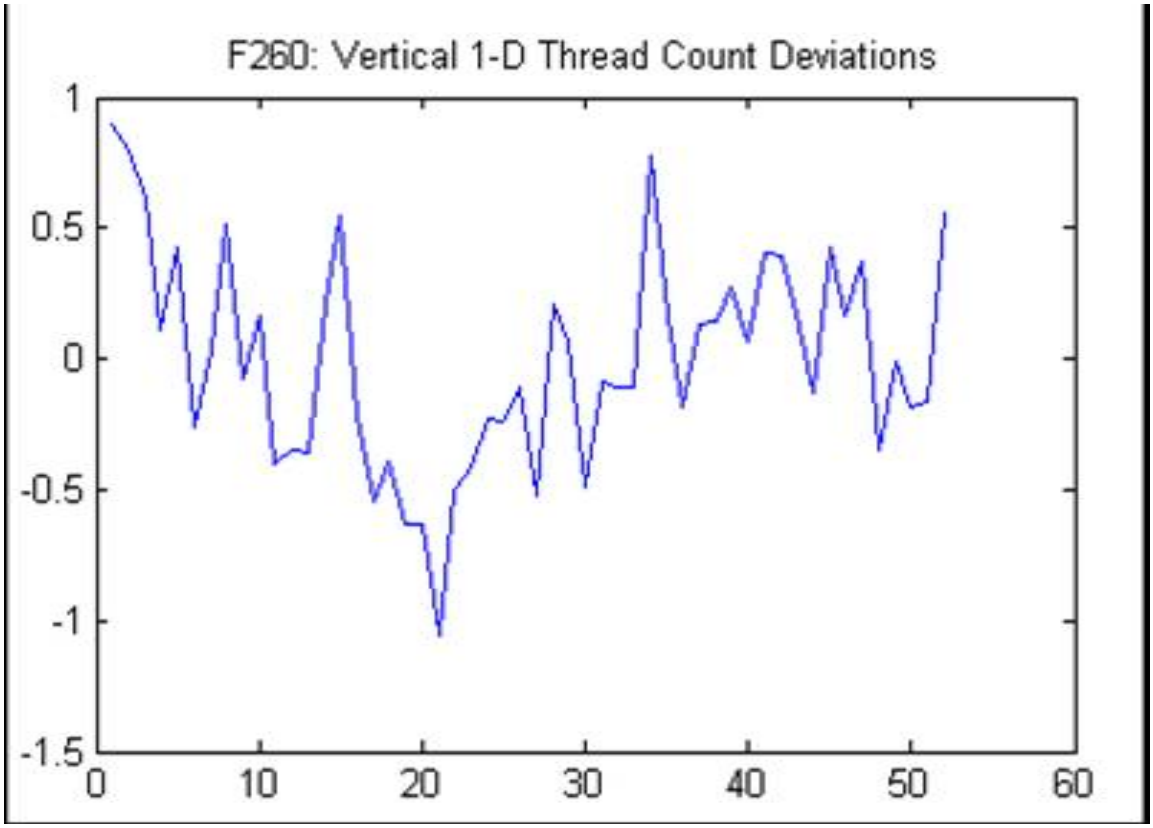


Figure 5.13

We can clearly see the similarities now. But how do they correlate then?

5.4.1.6 Step 6: Correlation Determination

We then computed unbiased correlation coefficient between 1D vertical thread count deviations of F205 and F260 along the x-axis. You can clearly see a peak (0.7479) appears at the 55th alignment as two paintings are mapped to the matching alignment, or visually "best fit" together. The correlation mapping plot is shown in figure 7.

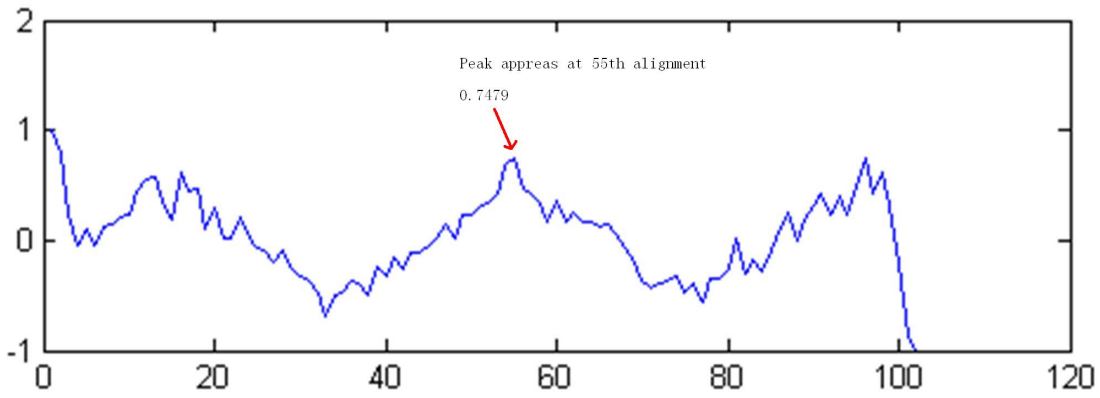


Figure 5.14

The results indicate that these two paintings (F205 and F260) were likely cut from the same canvas roll, sharing the more variable weft direction.

5.4.2 Orientation Issues in Matching

When we computed the correlation for the paintings, we actually did it four times in our Matlab programs. This is because that any two paintings could possibly match in either direction and each painting could be rotated 0, 90 or 180 degree.

In plotting the correlation mapping, we choose the one that gave the best fit among all possible orientations.

5.5 Conclusions and Future Work[10]

5.5.1 Conclusions

Clearly, our correlation analysis result is consistent with the matching of F205 and F260 as we can see from the two spectra. **And in fact, F205 and F260 were painted by Van Gogh in the same month in 1885!**

The spectral techniques of our project offer a more efficient and accurate approach to analyzing and sequencing paintings than manual methods. Whole-painting analysis could provide quantitative support for forensic evidence.

The following figure 1 shows the mapping result between all six paintings we worked on. Some paintings have multiple x-ray images due to its big size. Hot spots represent "good matches". Hot spots along the diagonal are expected as weave density deviations should match in the same painting.

[10] This content is available online at <http://cnx.org/content/m18947/1.1/>.

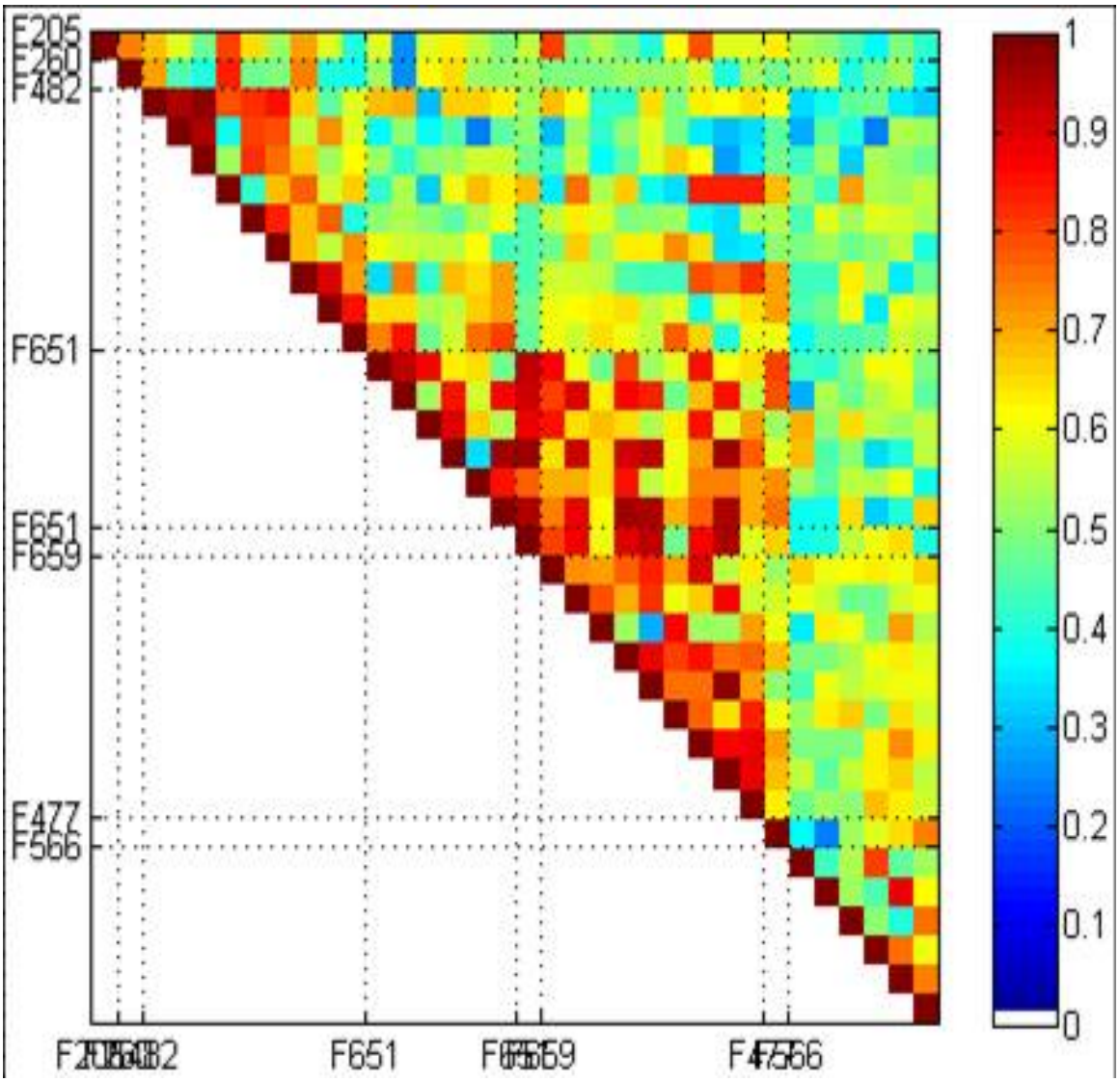


Figure 5.15

Figure 2 shows the same mapping result but with only hot spots left. (With correlation coefficient greater than 0.75)

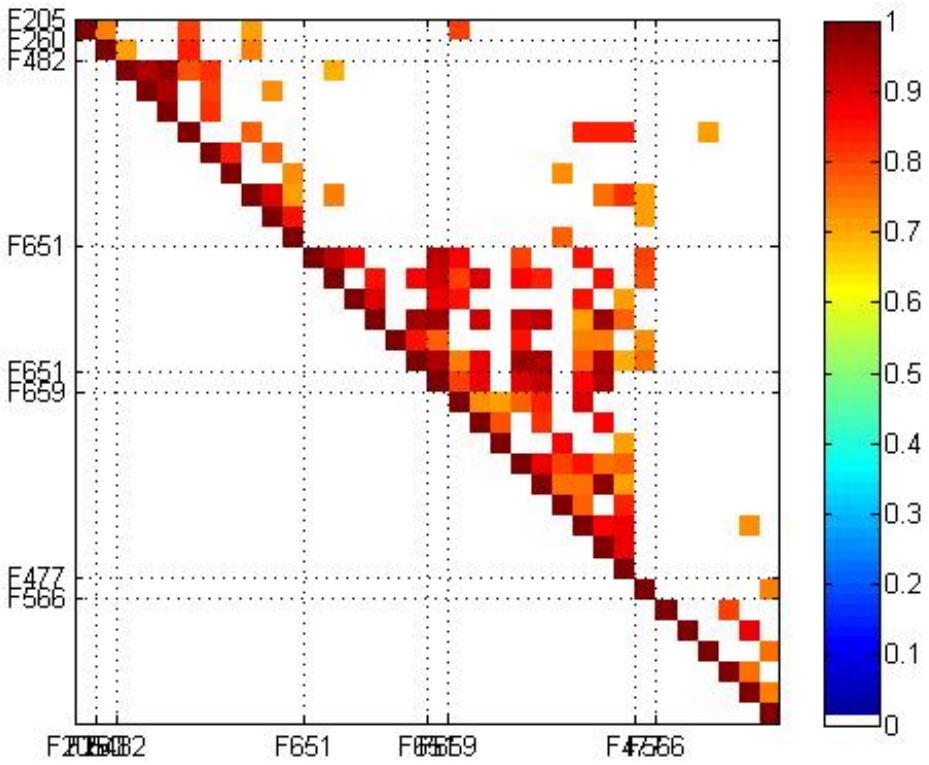


Figure 5.16

5.5.2 Future Work

5.5.2.1 Elimination of NaNs in Thread Counting Densities

NaNs are the result of outrange or multiple frequency peaks in the short time Fourier analysis for sample swatches. The reason for these abnormal frequencies are uncertain yet, might due to the x-ray scanning process, imperfect alignment of threads, etc.

5.5.2.2 Overlap and Critical Values

How big the correlation coefficient should be when we can say that two paintings match? This could vary much according to the size of paintings, alignment directions (warp or weft) and other facts. In addition, we noticed some peaks usually appear at the edge of our correlation plots. This is because small overlap sometimes results in matching accidently. We are working to find a threshold value for overlap and correlation coefficient.

5.5.2.3 Averages vs. Deviations

Another part of our project involves the Thread Count Average match between paintings which could be used collaborate with the result shown here to reach a more convincing analysis for art forensics.

5.5.2.4 Applications in Other Fields

Our research philosophy could extend to many related signal processing areas such as speech analysis, image identification and so on. The methods and techniques we developed could be employed not only for canvas paintings but also similar texture materials.

5.6 References and Acknowledgements[11]

5.6.1 References

D.H. Johnson, C.R. Johnson, Jr., A.G. Klein, W.A. Sethares, H. Lee, E. Hendricks, "A Thread Counting Algorithm for Art Forensics", 2008

5.6.2 Acknowledgements

We wish to thank Dr. Don Johnson of Rice University's ECE Department.

The thread counting algorithm is collaboration between the Thread Count Automation Project directed by Professor Rick Johnson at Cornell University and Van Gogh Museum, Amsterdam.

Images of this collection are courtesy of Van Gogh Museum, Dr. Don H. Johnson.

[11] This content is available online at <http://cnx.org/content/m18948/1.1/>.

Chapter 6

Give Me Your Digits!

6.1 Give Me Your Digits!: Introduction and Background[1]

6.1.1 Give Me Your Digits!: Introduction and Background

6.1.1.1 Introduction

Digital technology can allow us to save labor and time through the automation of menial tasks. Here we are developing a program to transcribe digits using a computer and a camera. Computers are notorious for excelling at repeating similar tasks/ instructions. The following modules will describe how we have implemented a rudimentary system to acquire, process, and identify handwritten digits (0-9).

6.1.1.2 Background

The conversion of handwriting to the digital regime has been implemented industrially by the USPS; more theoretical work has been compiled by academia most notably by Dr. Yann Lecun. We identified and classified handwritten digits through the utilization of the MNIST (modified national institute of standards and technology) database and classifiers – which include feature extractors and identifiers. We also preprocessed input images through the usage of morphological operators. All of the computation was done using Matlab version 7.1. The process we have developed and many like it have its usages in a world which is developing and converting to the digital regime.

6.1.1.3 Process Overview

The systematic process of identifying handwritten digits follows 3 major parts: image acquisition, image processing, and identification.

6.1.1.3.1 Image Acquisition

Firstly in the project an image must be acquired. This part of the project could have been simplified with the usage of a scanning device. Instead we opted for the usage of a digital camera to make our system more flexible. Along the way we found that the images produced by a digital camera required more rigorous image processing steps due to varying input qualities of images.

6.1.1.3.2 Database

The MNIST database: Its extensiveness is described here. We tested our identification algorithms using the database alone.

[1] This content is available online at <http://cnx.org/content/m18959/1.2/>.

6.1.1.3.3 Image Processing

This part of the process deals with converting raw input images into the standard MNIST database format. Most notably here we deal with cleaning up images and use morphological operators to help us separate digits.

6.1.1.3.4 Identification Algorithms

This part of the process deals with extracting features from each individual input digit and comparing them with features of images in the MNIST database. Feature extraction was implemented in the field with the 2D FFT. Identification/ classification or the actual matching algorithms goes into the regime of computer science courses and is slightly out of the scope of this course.

6.2 Give Me Your Digits!: Digit Database[2]

6.2.1 Give Me Your Digits!: Digit Database

The MNIST database
We used a database as a reference for comparing sample inputs. The database is a modified database from the National Institute of Standards and Technology (MNIST) database of handwritten digits. More specifically, we downloaded the database from http://yann.lecun.com/exdb/mnist/ and converted the database into Matlab format using code from http://www.cs.toronto.edu/~hinton/code/converter.m.

The format of the images in the MNIST database is a 20 by 20 grayscale image centered in a 28 by 28 box using its center of mass. The training set of the database is composed of ~60,000 (5,000 images/dig) images compiled from ~250 writers. The test database is disjoint form the training set and is composed of ~10,000 images (1,000 images/dig). This database was designed by computer scientists for the analyzing of classifiers (identification algorithm).

[2]This content is available online at <http://cnx.org/content/m18965/1.1/>.

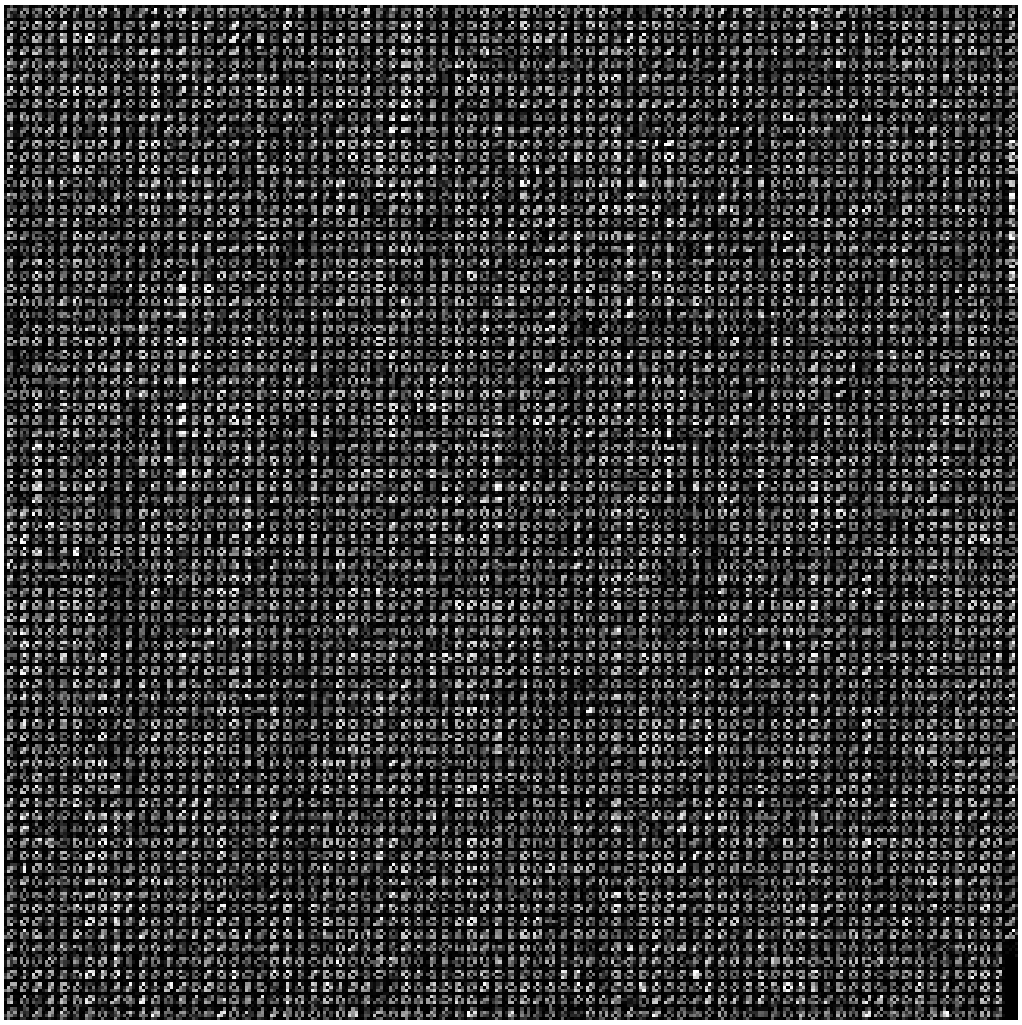


Figure 6.1: Image of the training database for the digit '0'

6.3 Give Me Your Digits!: Image Acquisition[3]

6.3.1 Give Me Your Digits!: Image Acquisition

Camera Setup

A Canon XSI Digital SLR was mounted onto a tripod as shown below. A desk lamp was used for additional lighting. More specifically an 18-55mm zoom lens was used at the 18mm (fully zoomed out), the onboard flash was used on manual settings. Handwriting was standardized to half an inch in height with a pilot ballpoint pen to help simplify the subsequent image processing procedure.

[3]This content is available online at <http://cnx.org/content/m18963/1.1/>.

Figure 6.2: Camera setup

One Time Manual Cropping
The algorithm used requires one line of handwritten digits at a time thus once the image was captured using the camera it was loaded into matlab using the uigetfile –(fancy gui to get filepath and filename) and Imread-(use filepath/filename to load in actual image into matlab workspace) commands. To finish off this step use the imcrop() command to manually crop one line of digits from the image with final product below. Note: this is the only time that there is manually cropping of the image from here on we complete the process algorithmically. Please see improvements on how we could of taken out all manual cropping entirely.

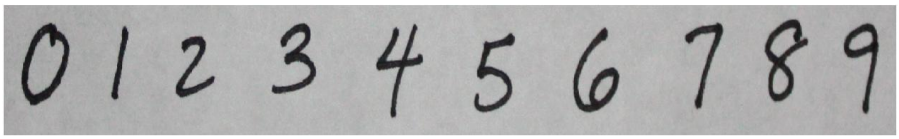


Figure 6.3: Photographed and manually cropped image of digits.

6.4 Give Your Digits!: Image Processing[4]

6.4.1 Give Your Digits!: Image Processing

When an image is acquired, its size and dimensions will vary. Since the database we use is in grayscale and 28x28 pixels large, each input image must be processed so that it also has the same dimensions.

[4]This content is available online at <http://cnx.org/content/m18969/1.1/>.

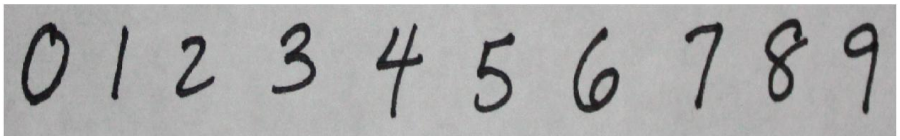


Figure 6.4: Acquired Image

6.4.1.1 Convert to Grayscale

The acquired image is in RGB format, where each pixel in the image consists of 3 components: the intensity of red, green and blue. Grayscale consists of a single intensity value for each pixel. Therefore, to change each input image to grayscale, we will use the Matlab operator, rgb2gray().

6.4.1.2 Edge Detection

The only information of interest in the input image are the intensities of the actual digits, rather than the intensities of the background. Edge detection will remove background noise and filter the image so that only areas of high contrast (which is the shape of the handwritten digits) remain. The figure below shows the result of applying a simple Sobel edge detector using the edge() command with the threshold set to default.

Figure 6.5: Result of edge detection with sobel operator.

6.4.1.3 Dilation: Morphological Operator

A morphological operator such as dilation processes an image so that its shape is changed in some way. Dilation expands all the boundaries of objects in the image. Dilation in this case also serves to connect any disconnected regions of an outline together. Through the imdilate() command, we can specify how to expand the digit outlines by a structuring element. Depending on the shape of the structuring element, the appearance of the image will change. The 'ball' structuring element will be a good choice, since this uses a gradient to make the digits look smoother and rounded out, similar to the way the data is normalized. Depending on how large the digits are written, the size of the structuring element will need adjustment – larger images require a larger structuring element in order to fill more of the outline. In order to get consistent results from dilation, we made sure all writing samples were approximately the same size. The figure below shows the result of dilation.

Figure 6.6: Dilation of Digit Outlines

6.4.1.4 Separating Individual Digits

Now that the only identifiable regions in the image are the digits themselves, the Matlab operator regionprops() with the 'boundingbox' option, we will be able to separate the digits into individual images.

Figure 6.7: Connected regions are identified by 'boundingbox'

6.4.1.5 Resize and Normalize Image

Most input images are much larger than needed and must be rescaled and centered to 28x28 pixels. Imresize() scales the image to 20x20 pixels. Afterwards, zero padding adds zeros to each side of the image, resulting in a 28x28 pixel image. The input image is now standardized and ready to be identified.

6.5 Give Me Your Digits!: Identification Algorithm[5]

6.5.1 Give Me Your Digits!: Identification Algorithm

6.5.1.1 Feature Extraction

In General
Now that the image is in the format of the database (28 x 28 grayscale), a method to match the image to the database must be explored. To achieve a high accuracy rate in matching from the database, a feature extraction method could be employed. Three methods will be explored: pixel matching, Principle Component Analysis (PCA), and two-dimensional Fast Fourier Transform (2D FFT).
Pixel Matching
Pixel matching, the first method, is not to use any feature extraction. It is meant to match the image to the database in the most obvious manner. The identifier literally compares the images pixel by pixel and

[5]This content is available online at <http://cnx.org/content/m18966/1.1/>.

computes the mean squared error (MSE, sums up all the error). In fact, the following two methods employ pixel matching after the transforms.

PCA Principle Component Analysis

Next, the Principle Component Analysis, employs an orthogonal linear transformation that transform the matrix to a new matrix such that the greatest variance of the projection of the data comes to the first coordinate (principle component), and the second highest variance to the second coordinate, and so on. For our purposes of matching the image from the database, the PCA actually performs very poorly. This is due to the fact that because the images so closely resemble each other (same variance). This similar variance is attributed to it being so small (28 x 28) and gray scale. As a result, the PCA transform of the images are almost identical for this specific database. Experimentally, roughly a 30% matching rate was observed, which is barely better than random guessing. PCA would be much better suited for large images with great variances.

2D FFT

Last, our method of choice, is the two-dimensional Fast Fourier Transform. Recall that the FFT is a form of Discrete Fourier Transform. The 2D FFT, for our purpose of image extraction, is the a frequency representation of the images. Note that the magnitude of the FFT must be used to compute the MSE. Also, the Matlab fftshift command was used shift the high frequency to the center.

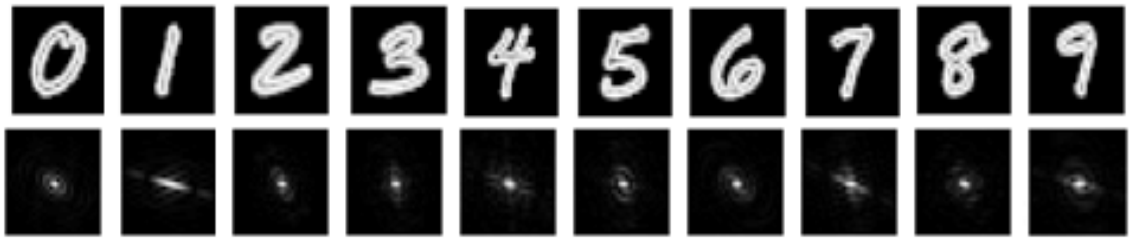


Figure 6.8: The formatted images and their 2D FFT's

6.5.1.2 Identification

In General

Once the input images are transformed, they are now ready to be matched and identified by the images in the database. Three algorithms can be used: a) finding the lowest MSE, b) finding the best averaged MSE, and c) finding the highest frequency digit from the best one hundred MSE.

Absolute Nearest Neighbor

The first algorithm finds the database image that has the lowest MSE from the input image out of all the randomly selected database images and declares that the input image is the digit the matched database image represents. In other words, it is an absolute nearest neighbor approach. The accuracy of this method depends heavily on the number of randomly selected database images since the bigger the database to choose from, the higher the chance that there will be a perfect match.

Averaging Database Per Digit

Another algorithm that could be used is to average all of the MSE's of the database of one digit with a input digit,and declare the digit with the lowest average as the input's digit. This method, in theory, should eliminate the chance picking an outlier.

Majority of Nearest Neighbors

Finally, the last algorithm is a variation of the aforementioned minimum MSE approach. This method

picks the one hundred lowest MSE's observed from the selected database and tallies the frequency of digits represented and declares the highest frequency digit is the input's digit. In other words, it is finding the digit with the most nearest neighbors. In theory, this approach should be more associated with finding common features since higher frequency of certain digit means shared characteristics. A drawback of this method could come from a particularly poor input (say a 3 that looks awfully like an 8), there is a possibly of "tying" since the digits are separated by integers.

6.6 Give Me Your Digits!: Results[6]

Give Me Your Digits!: Results

6.6.1 Using test database as input:

In order to compare and test the accuracy of our classifiers we randomly select 500 test images and test their accuracy using our identification algorithms (classifiers).

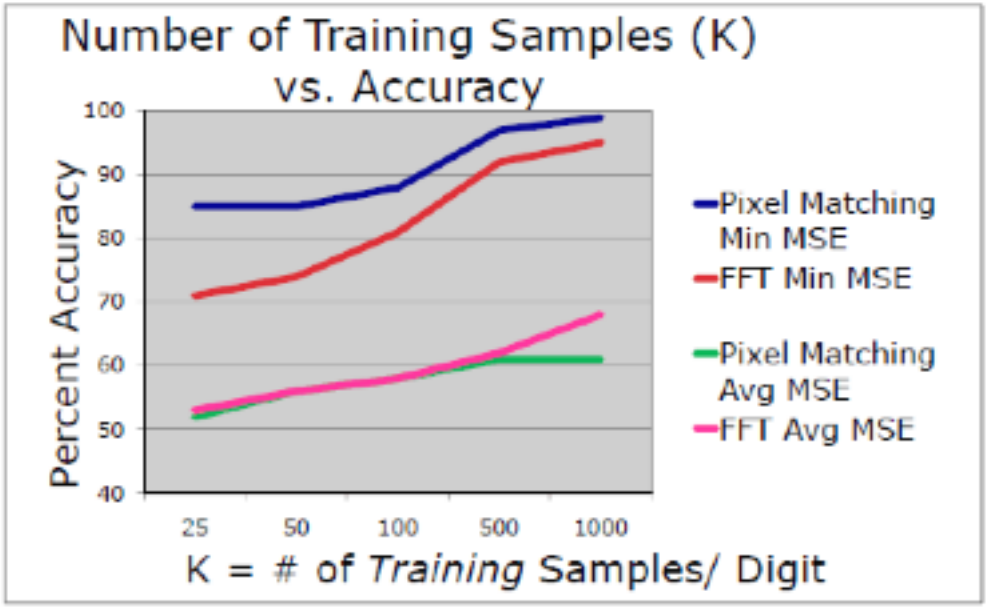


Figure 6.9: Accuracy of Minimum and Average MSE with FFT and without FFT

Reducing Size of Database
In general we vary K the number of training samples we randomly select – this effectively reduces the number of images we compare our input to using classifiers. In general as you increase K (effectively increase the size of the database we are comparing to and thusly our chances for finding a good match) we increase the accuracy of our algorithms.

[6] This content is available online at <http://cnx.org/content/m18968/1.1/>.

Pixel Matching
Pixel matching and minimum MSE give the best results. Pixel matching does so well here because the test images and training images we're created by NIST using the same image processing algorithm and thus match pixel to pixel accurately. It is interesting to note that using the FFT2D does not stray far from the pixel matching result. More extensive testing of different types of nearest neighbors and even neural networks is beyond the scope of this course and enters computer science.
Averaging
Averaging gives a poorer result because the database per digit is so varied: in theory taking a general average of some digits gives extremely similar answers; this could be the reason for such poor comparison; ∼50percent accuracy could signify half of the digits have this poor averaging characteristic. Averaging was originally implemented to rid the chance for out-layers giving erroneous nearest neighbor result, with our random selection of the training database and inputs we should actually be trying to find out-layers (exact) or close to exact matches to our image to get better results.

6.6.2 Using camera photo (image acquisition) as an input:

Due to limitation of resources we were only able to approximate this data by taking 10 digits (0-9) from 10 people and approximating the accuracy.

-FFT2 with minimum mean squared error gave approximately 95% accuracy with adequate spacing between digits – (if not spaced properly bound box will cut off adjacent digits skewing results).

-Pixel matching with minimum mean squared error gave approximately 80% accuracy.

Reasoning behind this is that the sequence and morphological operation parameters we used are not exactly the same as the ones NIST used to develop their database. Thusly the FFT2 will give more of and averaging effect yielding better results than just strait pixel matching alone.

6.7 Give Me Your Digits!: Improvements and Conclusion[7]

6.7.1 Give Me Your Digits!: Improvements and Conclusion

6.7.1.1 Suggested Improvements to the Digit Identifier

The Database
The database utilized in this project was developed solely for the testing classification techniques (analyzing identification algorithms). The size of the database files and vast diversity in the digits was good enough for our purposes. The USPS uses a similar database which can be found online, this database may be more suitable for input from a camera. More importantly we could expand our identification to letters if we expanded our database to include letters; this was out of the scope of our project as it requires extensive optimization of classifiers.
Feature Extraction and Identifiers
Feature extraction is possibly one of the areas for improvement. The FFT2D really does not really vary enough from digit to digit and thus is not a great feature extractor. Identifiers could be an area of extensive improvement, most notably instead of Nearest Neighbor approaches Neural Networks could of been used to train. An large list of types of classifiers has been compiled by Dr. Yann LeCun at http://yann.lecun.com/exdb/mnist/.
Reduce Noise
Noise reduction methods can reduce the chance to detect noise as a digit, the many filters that are in the image processing toolbox of matlab could make our system flexible enough to allow crumpled/ paper, smeared ink ect. A big part of making our project more viable in the real world would be to make the parameters of each morphological operator dynamic with different input images. For example detect pen thickness (how thick the lines that make up the handwritten digit) and increase the size of dilate to fill in

[7] This content is available online at <http://cnx.org/content/m18967/1.1/>.

the larger gap between edges. Edge detection parameter can also be changed to have more precise threshold (lower bounds and upper bounds- see caney edge detection) based on input.

-In more detail one should look into wiener2() command which adaptively filters out noise based on overall variance in the image. This effectively smoothes out edges (especially edges in noise dots) and thusly will reduce detecting edges in noise.

-Another filter that we looked into but did not end up using is medfilt2() which is a median filter which pushes noise out to the boundaries of image and then zero pads the boundaries of the image. Since we have no information at the boundaries of each of our digits this method would have worked with few side effects.

Be Prepared for Angled Imput

Input images may be angled based on both handwriting styles and/or due to tilt of the paper being photographed. Thusly the same identification algorithms used in this project could be calculated simultaneously with rotated (using imrotate()) versions of the input image.

Paragraph Cropping

We have accomplished the algorithm for cropping individual digits all on the same line. The extensive data from regionprops() command can be further analyzed to read in entire paragraphs of handwriting and even understand spacing and indentation.

Scanner

Finally there are more accurate and faster methods of scanning in handwriting using an industrial scanner like the USPS (lazer technology) or even a convention scanner could greatly improve the quality of images.

6.7.1.2 Conclusion:

With the use of image processing tools & identification algorithms, in our case, the FFT2, minimum MSE, one can achieve high levels of accuracy in hand-written digit recognition. As seen from the results above, we see that pixel matching is superior when using the MNIST test images.

Chapter 7

Information Hiding and Watermarking

7.1 Abstract Summary of Information Hiding and Watermarking[1]

7.1.1 Introduction

Objective
The objective of this project is to hide a binary message within a piece of audio without damaging sound quality. The encryption should be resilient against "attacks" from the outside. Specifically, the hidden message should survive the addition of noise, audio recompression, cropping errors, and re-marking an additional message on top of our message.
Background
To store a covert message in a sound file, the bits of the file must be changed at least slightly. These modifications add noise to the listener who wants to enjoy the music in its original form. Therefore the modification is kept as subtle as possible and masked to exploit flaws in the perception of the human ear. Typically, an attentive listener will not perceive a small decrease in audio quality, even with a high quality reproduction of the sound. However, special computer analysis of the modified sound file can reliably reveal the hidden message, provided that the encoder and the decoder must have some pre-arranged "agreement" about where the data may be hidden. The algorithms outlined in this project focus on modifying the original sound as minimally as possible and taking advantage of certain **psycho-acoustical perceptive phenomena** to maintain high sound quality despite the added noise of the encoded message.
Psycho-Acoustical Phenomena
Human auditory perception is limited by several phenomena related to the ear and how the brain perceives sound. These limitations can be exploited to modify audio files such that they sound remarkably close to the original yet contain subtle differences that store a hidden message. First of all, the human ear cannot hear quiet tones masked by loud tones close in pitch. A practical example is that an audience at a choir concert will generally not notice if one individual is slightly out of pitch, unless that individual is exceptionally out of pitch or exceptionally loud. Secondly, the human ear is insensitive to small phase shifts in audio. In practice, the ear is almost completely deaf to time-variations in sound, even when original and phase shifted clips of a sound are sampled back-to-back. Finally, subtle echoes may not significantly change how audio sounds. However, this varies depending on how large the echoes are and the contents of the original sound. Each of the three perceptual limitations listed above inspire the design of an encoding scheme.

[1] This content is available online at <http://cnx.org/content/m18992/1.1/>.

7.2 Encoding Information in Audio for Watermarking[2]

7.2.1 Encoding

Our three encoding algorithms all begin the same way. First we take the user-defined constant for the number of segments and split the input signal in the time domain. Each segment represents a possible bit of hidden data. More segments mean more bits and a higher data rate, but it also means fewer samples from which to decode.

Whenever we wish to encode a one we alter the time chunk according to the algorithm. To encode a zero we leave the segment alone. After altering (or not altering) each segment, we take the Inverse Fourier Transform if needed and recombine them into our marked $\hat{s}(t)$.

7.2.1.1 Frequency-Masking Algorithm (FMA)

In this algorithm we take advantage of frequency masking. Since the human ear cannot hear quieter frequencies next to a louder frequency, we alter these values. To encode a one we find the Fourier Transform for each segment of time and find the max value of this transform. Then we scale the neighboring values on either side by some scalar $\alpha < 1$.

The case where the max value is close to the edge of our segment is slightly more complicated. We chose to alter the values on the non-edge side as normal and to alter as many samples on the edge side as existed.

We cannot encode a bit if the maximum value is zero or if the neighboring frequencies are too small (according to some predefined value).

Testing revealed that $\alpha = .5$ was clearly audible for all of our test files, but $\alpha = .95$ was not enough of a difference for the computer to reliably detect.

Diagram of Frequency Encoding Algorithm

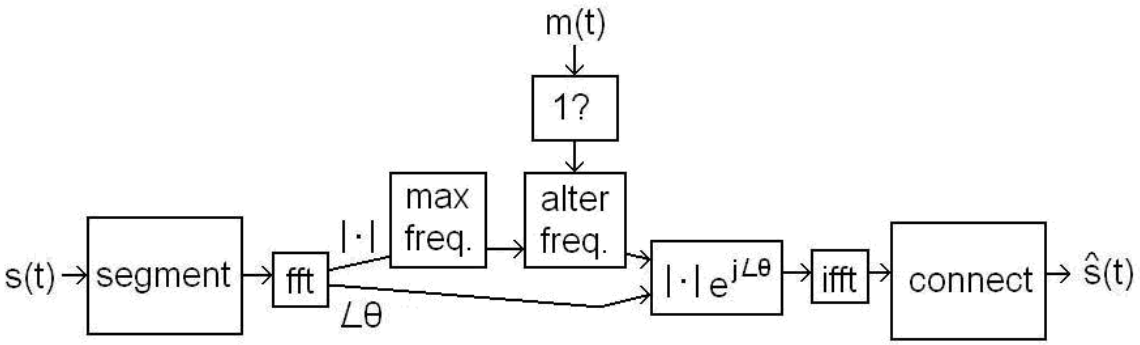


Figure 7.1: Diagram of Frequency Encoding Algorithm

[2]This content is available online at <http://cnx.org/content/m18995/1.6/>.

Example of a change made by the FMA

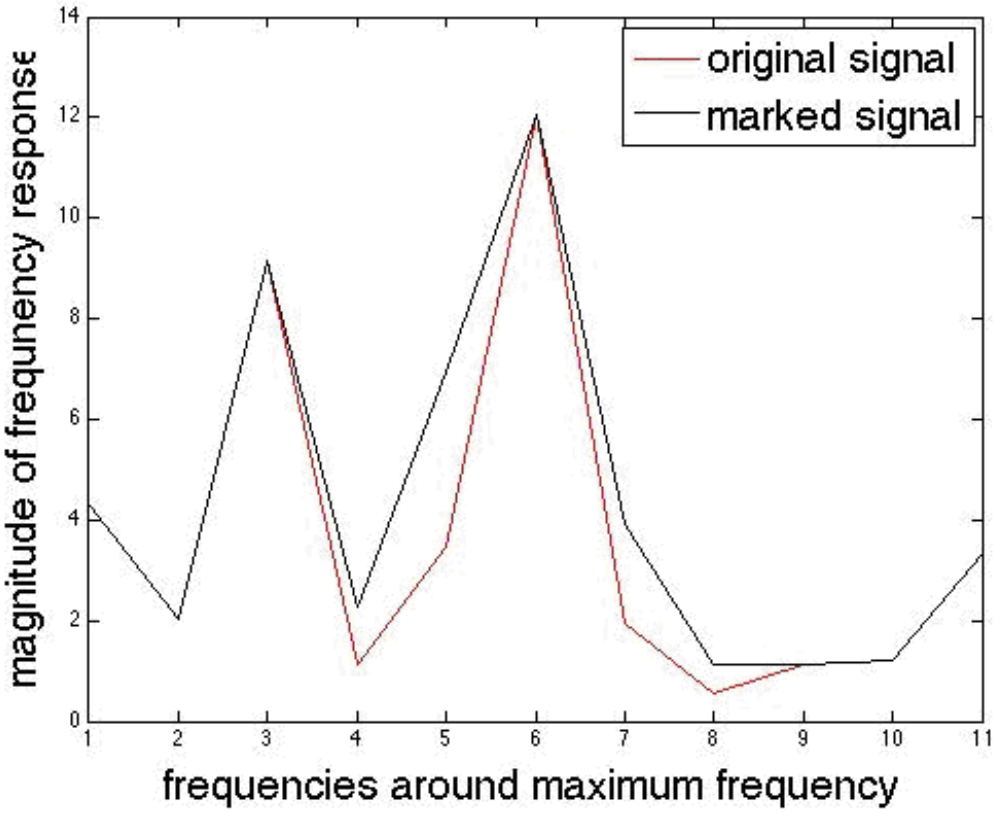


Figure 7.2: Example of a change made by the FMA to encode a 1

7.2.1.2 Phase-Shifting Algorithm (PSA)

In this algorithm we take advantage of the fact that the human ear cannot hear slight variations in phase. To encode a one we find the Fourier Transform of each time segment and slightly alter the phase by a predefined value.

We cannot encode a bit if most of the values are zero.

Testing revealed that a phase shift of $.25\pi$ is audible, while a phase shift of $.001\pi$ became hard for the computer to detect.

Diagram of Phase Encoding Algorithm

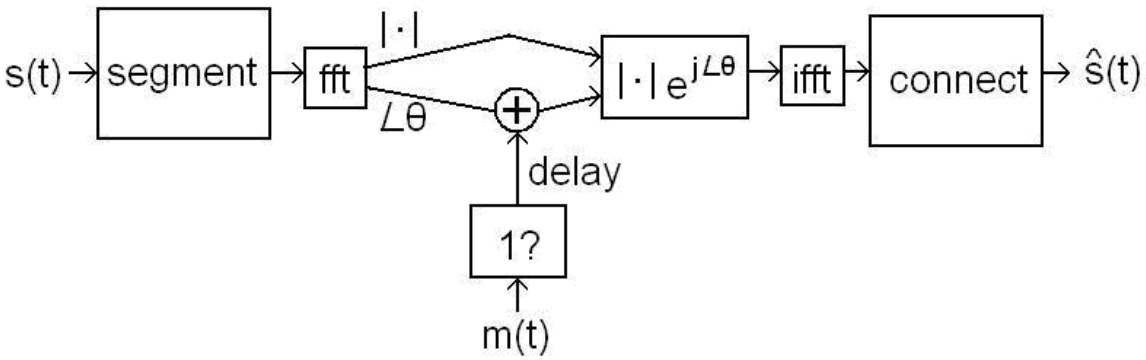


Figure 7.3: Diagram of Phase Encoding Algorithm

Example of a change made by the PSA

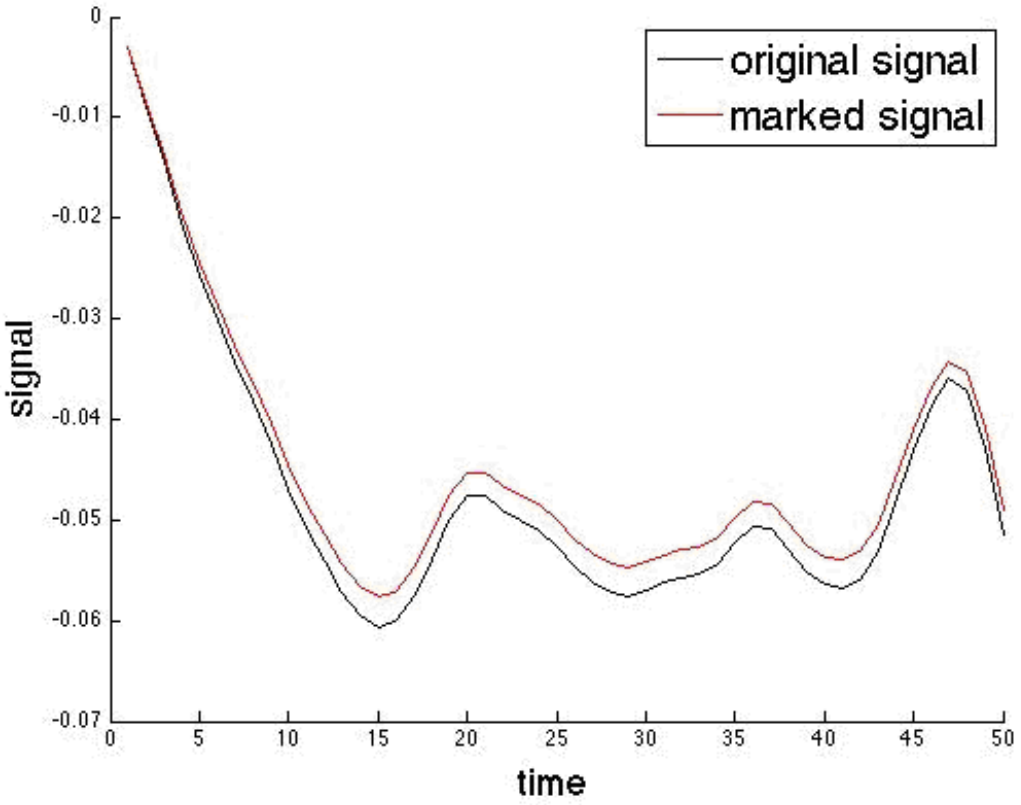


Figure 7.4: Example of a change made by the PSA to encode a 1

7.2.1.3 Echo Algorithm (EA)

Here we use the fact that our test suit is comprised of music and already have some amount of echo. Adding a slight amount more does not make an audible difference. To encode a one we shift the segment by some predefined value, de-amplify it by some scalar $\alpha < 1$, and add this echo back to the original segment.

We cannot encode a bit if most of the values are zero.

Testing revealed that an echo de-amplified to .2 was audible, while a de-amplification by .0001 was not reliably detected. Also a 30 sample delayed echo was inaudible to the human ear.

Diagram of Echo Encoding Algorithm

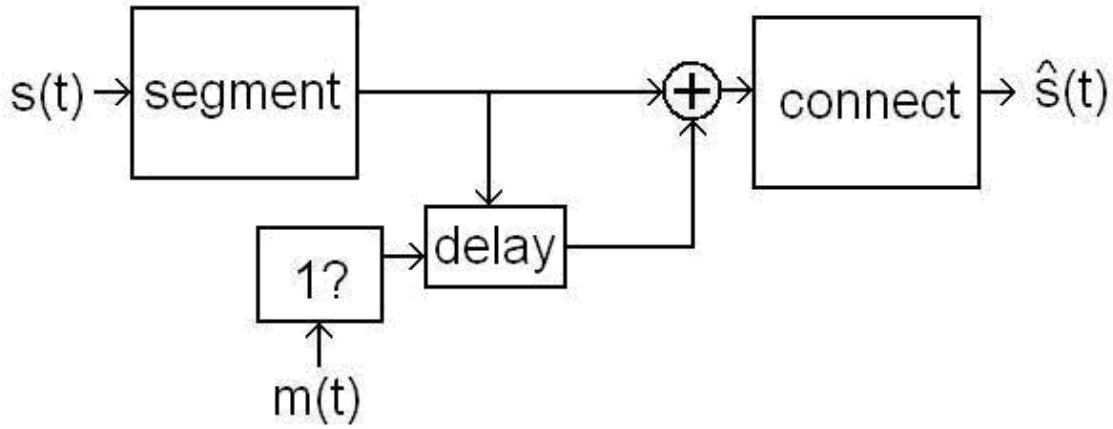


Figure 7.5: Diagram of Echo Encoding Algorithm

Example of a change made by the EA

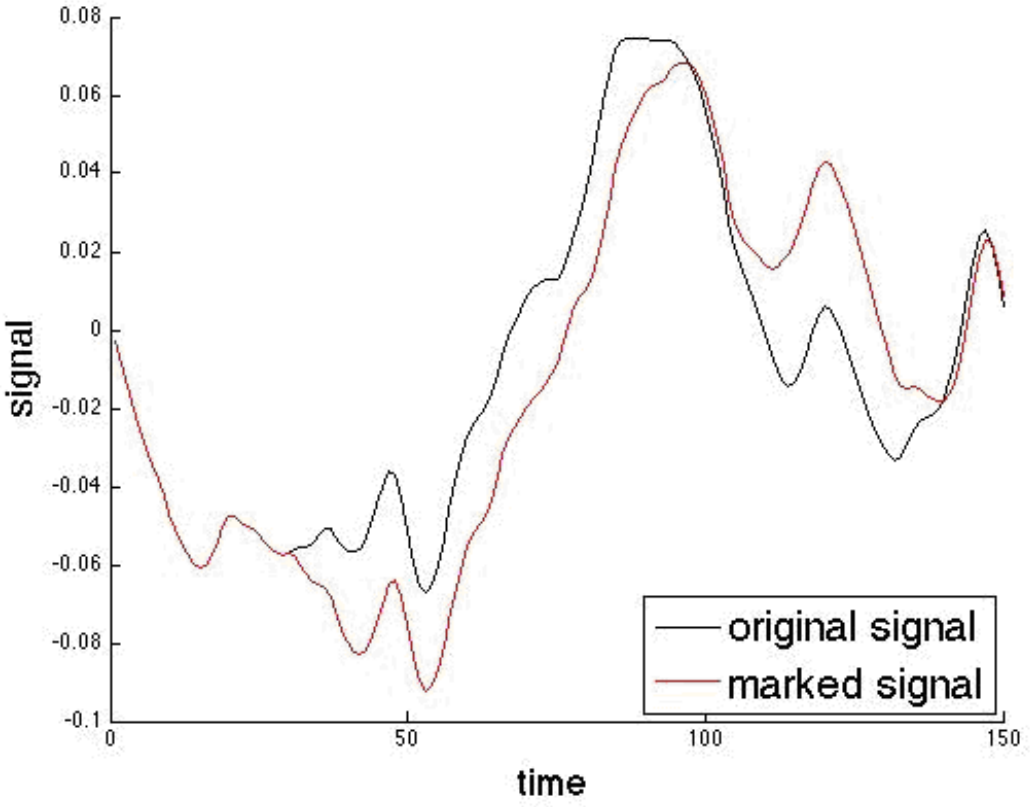


Figure 7.6: Example of a change made by the EA to encode a 1

7.3 Decoding Information Hidden in Audio for Watermarking[3]

To decode, our scheme needs certain information about the original signal. We need to know how many segments were created and the relevant, comparable value(s) for each segment. For simplicity's sake we simply input the original signal, though it would have been possible to not repeat this work.

Our decoding algorithms take the marked signal $\hat{s}(t)$ and segment it in the time domain as in the encoding algorithms. We next compare each segment of $\hat{s}(t)$ to the original $s(t)$ value. If the values are the same it was a zero. If the values have changed it was a one.

To decide whether change has occurred, we subtract the proper values and create a sum of the differences, i.e. a scalar representing the amount of change. We then compare this scalar to some threshold. For marked signals with no noise, this threshold can be 0.

[3]This content is available online at <http://cnx.org/content/m18994/1.2/>.

Diagram of Decoding Algorithm

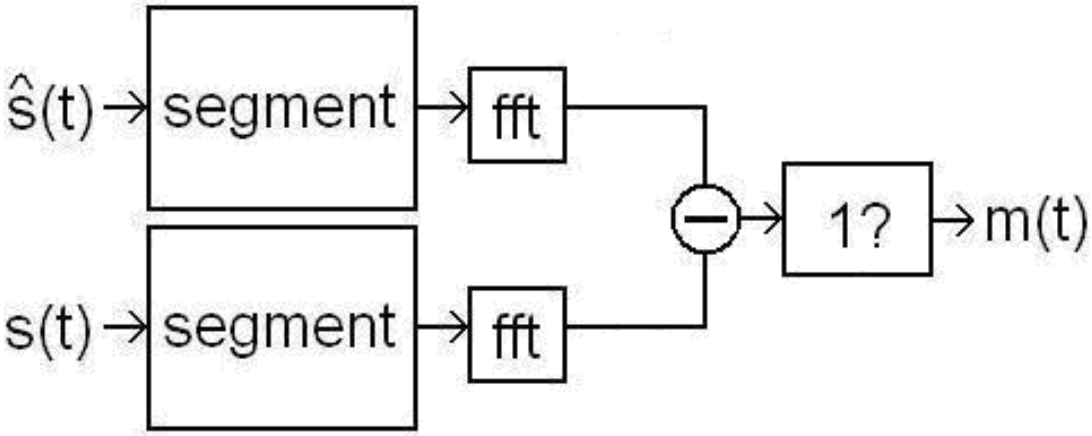


Figure 7.7: Diagram of Decoding Algorithm

7.4 Testing Methods of Information Hiding and Watermarking[4]

7.4.1 Testing

7.4.1.1 Aural Tests

As our primary objective was to make the changes inaudible, we tested all of our algorithms aurally.

We initially tested the algorithms on a 440 Hz tone to ensure that the algorithms were working as expected. (We did not test the FMA on the tone as doing so would have been silly because the tone only has a single frequency with no other frequencies to modify.)

We continued our aural testing with a suite of six songs from different genres: classical, hip-hop, oldies, pop, rock, and techno. We adjusted any thresholds and predefined constants to the point of aural imperceptibility. Working within these limits we were then able to modify these constants to maximize bitrate, accuracy, and noise resilience.

The following figure details the particular songs chosen and their overall frequency spectrums.

[4]This content is available online at <http://cnx.org/content/m18999/1.2/>.

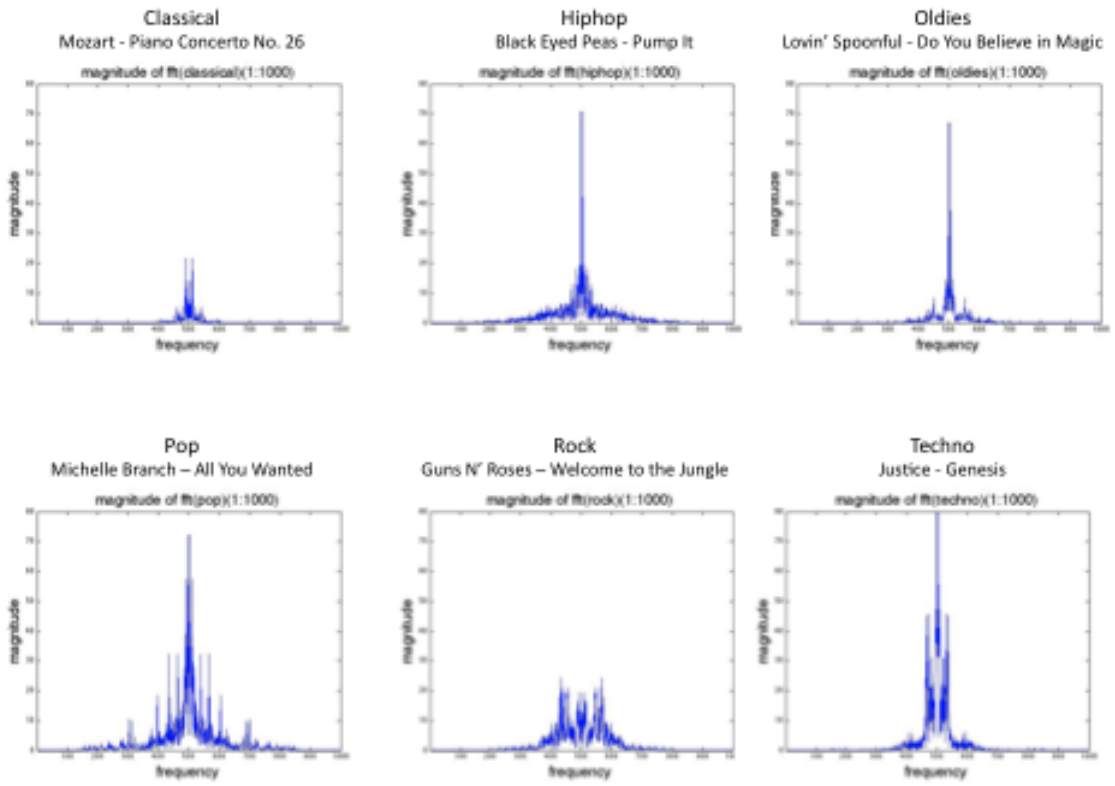


Figure 7.8: Frequency spectrums of test suite songs

7.4.1.2 Bitrates

Our test suite had a CD quality sampling frequency: 44100Hz, which amounts to 220500 samples for a 5 second long clip. Ideally with no noise it would be possible to use a segment length of 2 samples. This setup translates to $220500/2 = 110250$ segments in 5 seconds and $110250/5 = 22050$bits/sec. I.e. at CD quality, we cannot get more than a 22Kbits/sec data rate.

In practice we found that Mat lab was unable to handle this amount of data. We were, however, able to successfully reach 4800 segments, or 46 samples per segment. These values translate to $220500/46 = 4793$ segments in 5 seconds and 958bits/sec. I.e. we reliably demonstrated a 1Kbit/sec data rate.

7.4.1.3 Power Ratios

To measure how much we had changed each of the signals by encoding bits, we took a power ratio of the original signal to the output signal.

Formula for Power Ratio

$$powerratio = \frac{\|s(t)\|^2}{\|\hat{s}(t)\|^2}$$

Figure 7.9: Formula for Power Ratio

We found these ratios for two different input characters: '@' and 'w'. Because '@' is encoded by 100 0000 in ASCII, these power ratios measure the minimum amount of change we make to our signals. Because 'w' is encoded by 111 0111 in ASCII, these power ratios measure the maximum amount of change we make to our signals.

Power Ratios						
	FMA		PSA		EA	
	@	w	@	w	@	w
classical	1.0052	1.0362	1.0056	1.0352	0.9992	0.9955
hip/hop	1.0079	1.0507	1.0068	1.0413	0.997	0.9818
oldies	1.0133	1.0747	1.0069	1.0425	0.9986	0.9897
pop	1.0115	1.0776	1.0063	1.0388	0.9975	0.9842
rock	1.0131	1.0628	1.0072	1.0419	0.9975	0.9888
techno	1.0155	1.0897	1.0077	1.0463	0.9951	0.9723

Table 7.1

Table 1. Power Ratios for each algorithm encoding one 1 per seven bits ("@") and one 0 per seven bits ("w")

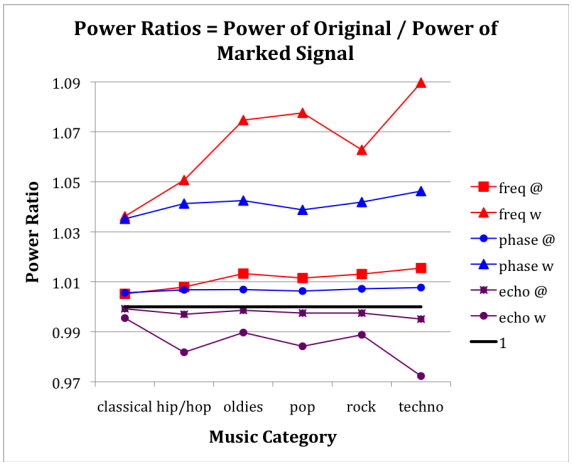


Figure 7.10: Chart of Power Ratios

The most important feature of these results is that all of our power ratios are very close to one, indicating that we have not changed the signal very much.

We also see some variation across the different songs because which values are changed and by how much depends on the song; for example, with the PSA, the delay causes us to drop samples at the end of the segment, and the power in the dropped samples depends on the song.

As expected, for 'w', the power ratio is further from one as more one-bits are encoded. Because adding an echo can be variously constructive or deconstructive, the power ratio does not reflect the number of one-bits as much as FMA and PSA. This fact also explains why the power ratios for the EA are generally lower than those for the FMA and PSA.

Finally for FMA and PSA the power of the marked signal was lower than the power of the original signal. For the FMA, this decline in power was expected because we scaled frequencies down, thus, deceasing the power in the frequency spectrum, which, as Parseval's Theorem tells us, corresponds to decreasing the power of the signal. For the PSA, this decline in power was also expected because the PSA delays the signal in various segments, dropping samples in the marked signal. The EA was the only case in which the marked signal had greater power than the original signal because the echoes in this case were more constructive than destructive.

7.5 Surviving Attacks on Information Hiding and Audio Watermarking[5]

7.5.1 Noise Addition

We added varying amounts of Gaussian noise to our marked test signals and attempted to retrieve the hidden message with varying success.

In order to account for the noise, we adjusted the threshold we used in the decoding algorithm. From Figures 10,12, and 14, it is clear that a threshold can be drawn between the higher observed differences (one-bits) and the lower observed differences. However, in the cases where too much noise was added (see Figures 11,13, and 15), this threshold is not so clearly defined. In fact, for the PSA and EA, any threshold value is difficult to determine whether by calculation or by "eyeing it."

[5]This content is available online at <http://cnx.org/content/m19003/1.1/>.

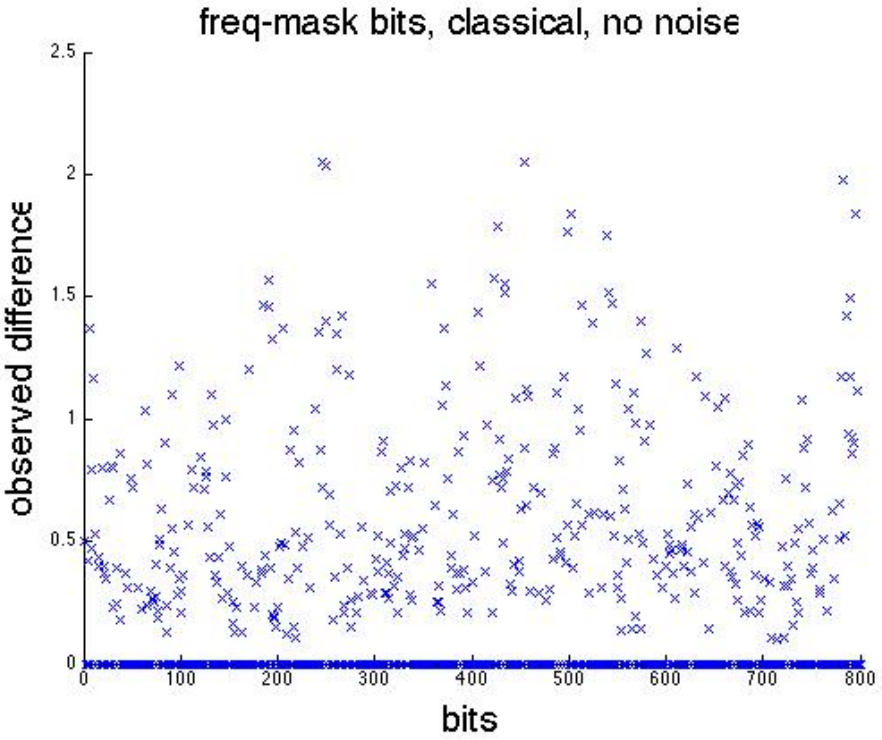


Figure 7.11: Observed differences between frequencies of original signal and frequencies of marked signal for FMA with no added noise

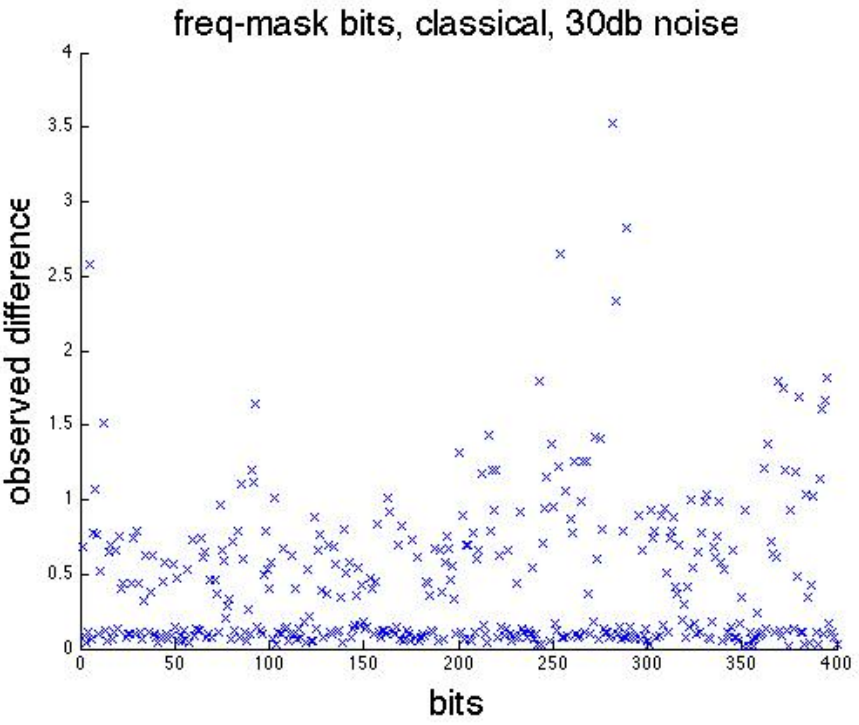


Figure 7.12: Observed differences between frequencies of original signal and frequencies of marked signal for FMA with added noise of SNR 30 dB

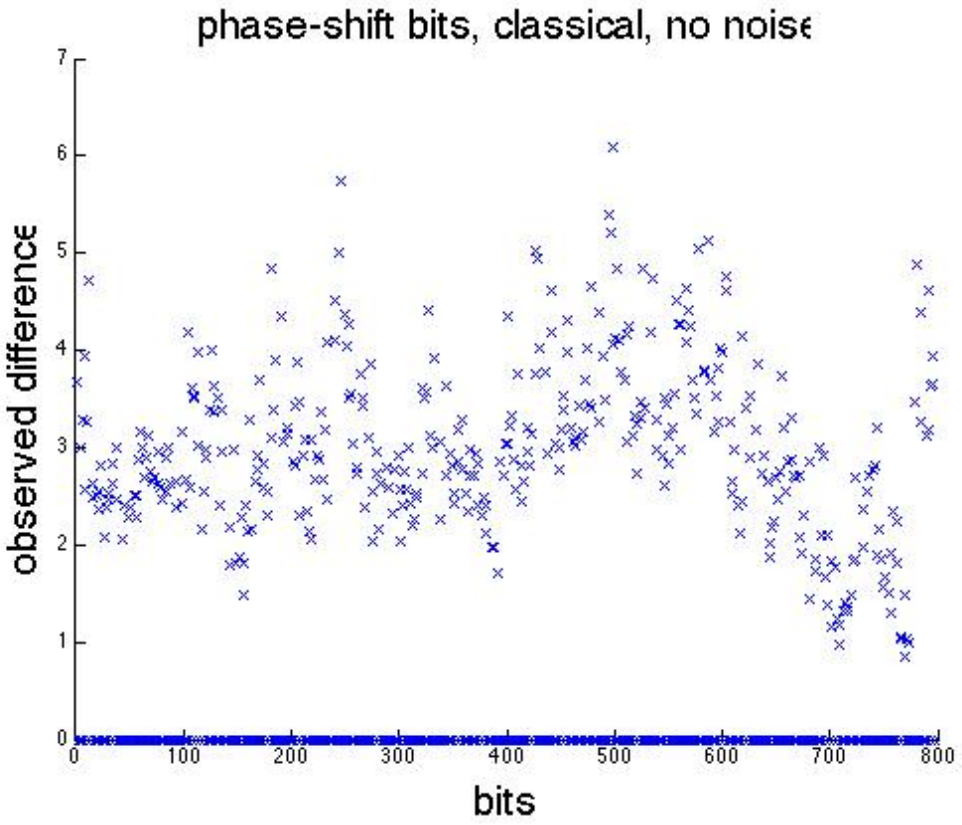


Figure 7.13: Observed differences between frequencies of original signal and frequencies of marked signal for PSA with no added noise

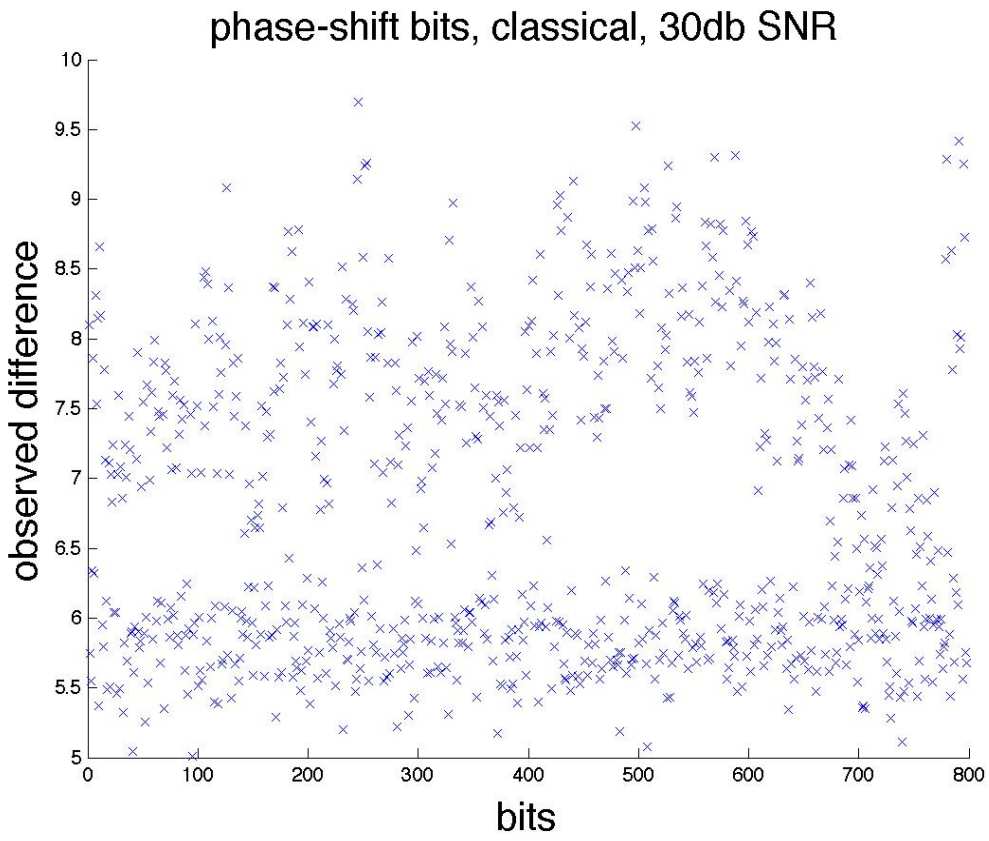


Figure 7.14: Observed differences between frequencies of original signal and frequencies of marked signal for PSA with added noise of SNR 30 dB

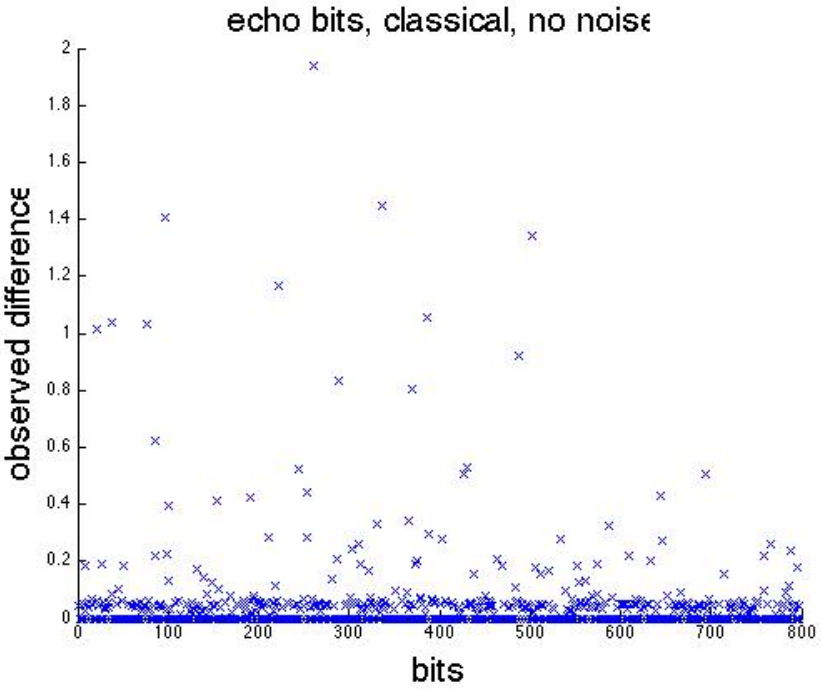


Figure 7.15: Observed differences between frequencies of original signal and frequencies of marked signal for EA with no added noise

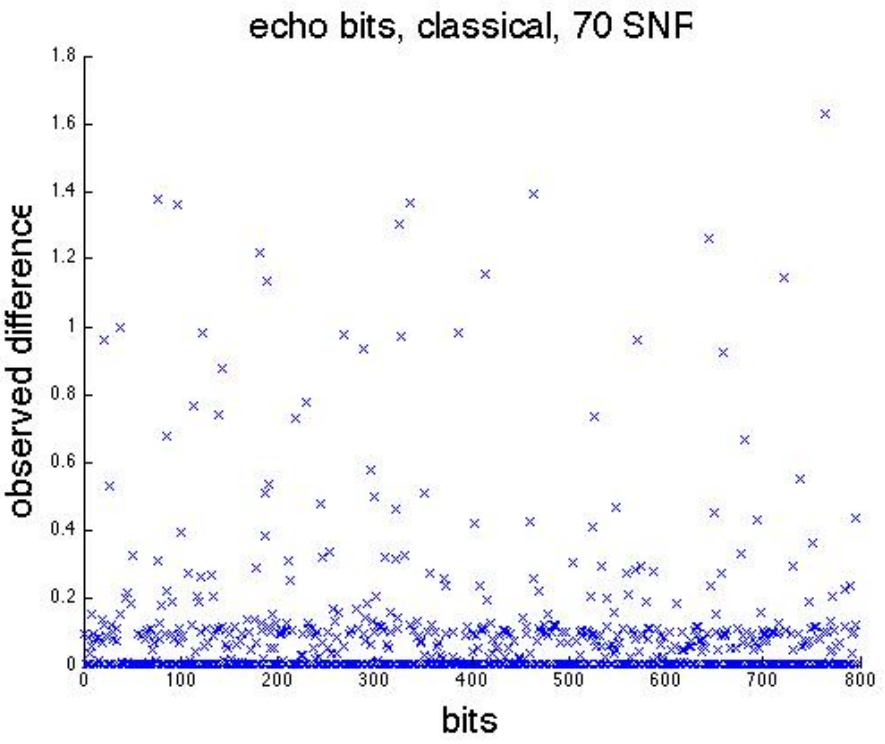


Figure 7.16: Observed differences between frequencies of original signal and frequencies of marked signal for EA with added noise of SNR 70 dB

Based on studying these threshold values, we found maximum noise we could add to the marked signal for each algorithm. The minimum SNR for the FMA, PSA, and EA were 30 dB, 30 dProject! ElB, and 60 dB, respectively. At these SNR values and with an input of "Elec301 Project! ", the algorithms output as follows:

FMA

Elec301 Pro*ect! Elec301 Project! ElB, and 60 dB, respectively. At these SNR values and with an input of "Elec301 Project! ", the algorithms output as follows:

ec301 Project! Elec30ec301 Project! Elec30

PSA

Elec301Elec301Elec301ELec301Elec301EleC#01Elec301Elec300Elec301El ek30qElec301Elec301Elec301Elec301El El'c 1El

EA

Elec381 Pzg*ect!MleC343 RzebesT!eoec#p1 Psozec|#GleC#00 P2ojEct)Mmec301 QrozEkw Umea3p1'PRgbmct!Eleg:0q P2ojea4%D|

The FMA and PSA clearly outperformed the EA in the noise category. In fact, at the point that we begin to miss bits, we can already significantly hear the white noise. Because the FMA only looks at the difference between the frequencies of the original and marked signals for a small segment of the frequency spectrum (in particular around the maximum frequency of the original signal), the noise power included in the difference is much smaller for the FMA than for the PSA and the EA which both calculate the difference over the whole frequency range. The PSA is good because our ears cannot detect significant alterations in

phase; in fact, the phase shift could go up to .1*pi without audible detection. This large phase shift power difference between a one and zero is much more than the power added by the noise.

For all of our algorithms some genres of our test signals performed better in every case. We found that 'pop' and 'techno' both failed noise tests at least 20db SNR higher than any of the others. Examining the magnitude in the frequency domain for both of these signals (see Figure 8) shows us that the 90% power bandwidth is wider than in the others. I.e. there is significant information at higher frequencies, so more significant frequencies are altered by the noise, which lessens the amount of tolerable SNR.

To defend against noise we encode the user-input phrase over and over as many times as will fit. This increases our chances of getting the phrase back since the probability of several bits being wrong is lower than the probability of one bit being judged incorrectly.

Another defense against noise was to raise our various predefined values closer to human-hearable level. For example we can increase the amount of phase shift in the PSA from .01*pi all the way to .1*pi. These increases mean that the value shifting caused by adding noise is not significant when compared to the value shifting created by encoding a one. In general, there is a balance between how much noise a marked signal can take and how audible the mark becomes. This balance is found by toying with the predefined values for each algorithm.

7.5.2 Compression and Decompression

We also tested an attack in which the wave file was compressed using MPEG-1 and AAC compression algorithms. In order to test whether we could still recover our encoded bits in MATLAB (MATLAB can only work with wave audio files), we then decompressed the files. We found that we were completely unable to recover our encoded message and received as output either nothing or complete garbage.

These results were not unexpected as audio compression algorithms take advantage of the same psycho-acoustical phenomena that we used to

7.5.3 Cropping

Our encoding scheme can survive truncation on the back end. We simply lose any bits contained in the deleted data. While we did not implement this process, we could implement a matched filter setup to survive truncation on the front end. We could take the marked audio file and attempt to place it in the unmarked original file using convolution.

7.5.4 Remarking

We tested whether our algorithm could be marked with a second message and still recover either message. We provided our decoder with the original signal and the signal that had been marked two different times. If we had provided the decoder with the once-marked signal and the original signal (cascading the decoder), we could have easily recovered our original signal; however, we felt this solution was trivial and against the point of the attack.

We found, as expected, that when encoding two different messages, we could not correctly recover either message. We, however, found that if two of the same letter were encoded in the same place, we could recover this particular letter. Remarking the signal with the same message does not affect our ability to decode the message, but remarking can affect the quality of the output.

7.6 Uses of Information Hiding and Watermarking[6]

7.6.1 Uses

Secure Storage
Hiding security-sensitive information in music epitomizes the idea of **"security through obscurity"**. Even if an intruder should gain access to the encrypted music files, there may not be any external indication that the files contain encrypted data, rather than simply music. Playing the files as they were intended produces regular, non-suspicious music that is nearly indistinguishable from the unmodified recording. Even if a would-be eavesdropper realizes that the music contains encoded bits, the issue still remains of finding the encoded bits within the signal and deciphering their meaning.
Covert Communication
The same reasoning can be applied to music that is openly broadcast between parties wishing to communicate in secret. An unsuspecting listener just hears music, but the desired audience has the tools to extract the hidden message. This is not just for spies, of course. A system could be designed where a radio station broadcasts music encoded with information about the song that is currently playing. A special receiver interprets the hidden code and offers the listener the option to buy the current song or other songs by that artist. Regardless, listeners with or without the special receiver do not perceive any loss of sound quality compared to a regular radio broadcast.
Copy Control
Individual copies of a piece of music could be labeled with imperceptible watermarks containing serial numbers. The watermarking algorithms designed in this project would have to be modified slightly, but they could be used to verify a signal's compliance with the following rule. A copy is only a legitimate copy if it includes the official watermark and/or a serial number that has already been sold. Furthermore, when each customer purchases a copy of the song, his or her purchase will be assigned a serial number. If multiple copies of a song bearing the same serial number are discovered where they should not be, then it is clear which user is responsible for breaking the rules.

7.7 Future Work in Information Hiding and Watermarking[7]

7.7.1 Future Work

Recover the Encoded Message without Original Signal
All three encoding and decoding schemes require that the modified output signal be compared to the original signal to attempt to recover the encoded message. Obtaining the original signal can be cumbersome in practice and may present logistical problems. Fortunately, this requirement can be lifted with a slight design change.
Detect Whether a Signal has been Watermarked
This project could also be furthered by creating a decoding process which takes in a signal and a message and attempts to discover whether the signal has been marked with that message.
Survive Cropping Attacks
Our algorithms could survive cropping if we set up a matched filter in the decoder. First we would determine where the marked signal is located in the original signal using cross-correlation. Then we could crop the original signal in order to compare it to the marked signal and recreate the message (without, of course, the bits lost in the crop).
Increase Security by Pseudorandom Sequences
By first encoding the message with a pseudorandom sequence, we could increase the security of the message. If the encoder uses the pseudorandom sequence with the message as a seed to select which segments are

[6] This content is available online at <http://cnx.org/content/m19001/1.1/>.
[7] This content is available online at <http://cnx.org/content/m18996/1.1/>.

encoded, the decoder can only find what the original message was if he also has the sequence. Thus, the encoder and the decoder must have some key sharing mechanism.

A second method of increasing the security using the pseudorandom sequences is varying the segment length. In the first step of each encoding process, the original signal is cut up into segments of equal time length. If, instead, the length of each segment is varied according to a pseudo-random sequence known by the transmitter and receiver. Without this sequence key, a potential eavesdropper would have great difficulty finding–let alone interpreting–the changes detected in the modified sound.

Encode on both Audio Channels

All three encoding processes currently only encode on one channel of a stereo audio signal. Both channels may be used to store additional information, at the cost of degrading the sound quality further. The effectiveness of this strategy is limited because the human brain is comparatively good at discerning differences in sound between the two ears.

Use Error Correcting Codes

This project focuses more heavily on the design of the encoding and decoding systems than the contents of the transmitted message. However, system performance in the presence of noise might improve if some form of error correcting code is used. If single-bit errors are evenly distributed throughout a decoded message, error correcting codes will improve accuracy. The trade-off is that fewer unique bits can be encoded, and message length must be reduced.

Extend Findings to Speech Signals

This project focused exclusively on hiding digital data within music files. Perhaps a more practical application is to apply these results to speech signals. Human speech typically covers a smaller frequency range than music and also typically lacks harmonic resonance. It is not clear how well the encoding and decoding schemes will perform when applied to speech.

7.8 Conclusions about Information Hiding and Watermarking[8]

7.8.1 Conclusions

Different genres of music are compared (for example classical, oldies, rock, pop, hip hop, techno) with respect to their data hiding capacity and subjective sound quality when modified by each of the three encoding schemes. At peak, each one encodes over 200 seven-bit characters in a five second audio clip. Matlab becomes unstable when the algorithms are scaled up to encode more bits, although there is no indication that the algorithms would fail if given sufficient computational power. Furthermore, it is nearly impossible to distinguish between the modified and unmodified five-second sound clips, even while listening carefully through high quality headphones.

All of these algorithms can encode about the same number of bits with reasonable quality, although at least one subject complained that the PSA left a slight ringing in the marked audio file. The FMA and PSA both stand up very well to noise, while the EA does poorly. The EA quickly reaches a SNR at which the decoder can no longer tell which values were supposed to be ones and which were zeros. The FMA decoder examines such a small band of frequencies that it is much less affected by broadband Gaussian noise. Since the human ear detects phase shifts poorly we can shift the phase such that the added noise is insignificant. The FMA changes the signal power the most, while the EA changes the power the least.

In conclusion, we were successful in hiding a binary ASCII string in audio files without audible loss of quality, using three different methods: Frequency-Masking, Phase-Shifting, and Echos.

[8] This content is available online at <http://cnx.org/content/m18993/1.1/>.

Chapter 8

Music Synthesizer

8.1 Introduction[1]

8.1.1 Introduction to a Music Synthesizer

8.1.1.1 Our Goal

The aim of the project was to create realistic instrument sounds by means of digital signal processing (DSP). Algorithms and theories already exist for mimicking various instrument families and all revolve around modeling the instrument structure and material as well as how the instrument is played. One of the simpler algorithms is the Karplus-Strong, which produces amazingly realistic guitar sounds. Using this algorithm in combination with attack-delay-sustain-release (ADSR) concepts, we were able to synthesize some decent instrument sounds.

8.1.1.2 Next Step

The first step would be to increase our instrument library to include different instrument families. Karplus-Strong mainly works for string and some percussion instruments, which greatly limited the diversity of our virtual orchestra. This would consist of implementing different algorithms to simulate the various types of instruments. Finally, how does mimicking instruments through DSP constitute music synthesis? The next objective would be to generate random pieces of music. A plausible scheme would be to implement a first or second order Markov model to generate random pieces with musical structure but leave enough room for creativity.

8.2 Karplus-Strong Algorithm[2]

8.2.1 Karplus-Strong Algorithm

8.2.1.1 How It Works

The Karplus-Strong algorithm was developed by Alexander Strong and analyzed by Kevin Karplus as a model of hammered or plucked string instruments. It simulates the sharp impact through a short wideband signal such as a burst of white noise. The signal is fed back through a delay line whose length depends on the frequency of the desired note. The delayed signal is sent through a lowpass filter to attenuate all other frequencies except the frequency of the note and its harmonics.

[1] This content is available online at <http://cnx.org/content/m19005/1.2/>.
[2] This content is available online at <http://cnx.org/content/m19006/1.2/>.

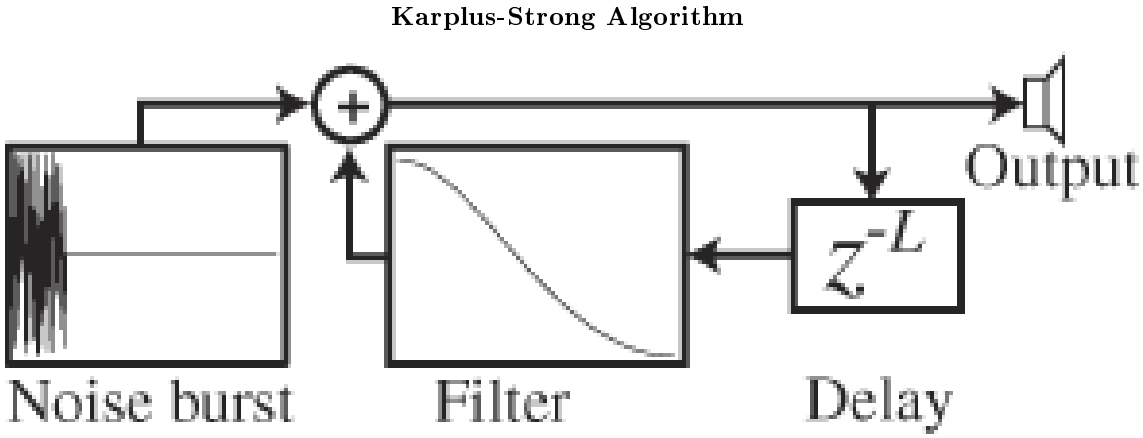


Figure 8.1: A diagram representation of a burst of white noise being delayed, filtered, and combined with the original burst. The output sounds like a realistic guitar string pluck.

8.2.1.2 Concepts

The main concept behind the algorithm is to model the sudden attack of a pluck with white noise containing equal energy in all frequencies. Due to the cavity of the instrument, the instrument material, and various other parameters, only a given frequency and its harmonics will resonate. This is simulated by recursively shaping the output signal. By matching the length of the time delay to correspond with the frequency of the note desired, the output will ultimately sound at the selected frequency given a short period of time. The feedback loop only reinforces the fundamental frequency and its harmonics. This technique is sometimes referred to as a comb filter because of the characteristic shape of the output.

Spectrum of a Plucked String

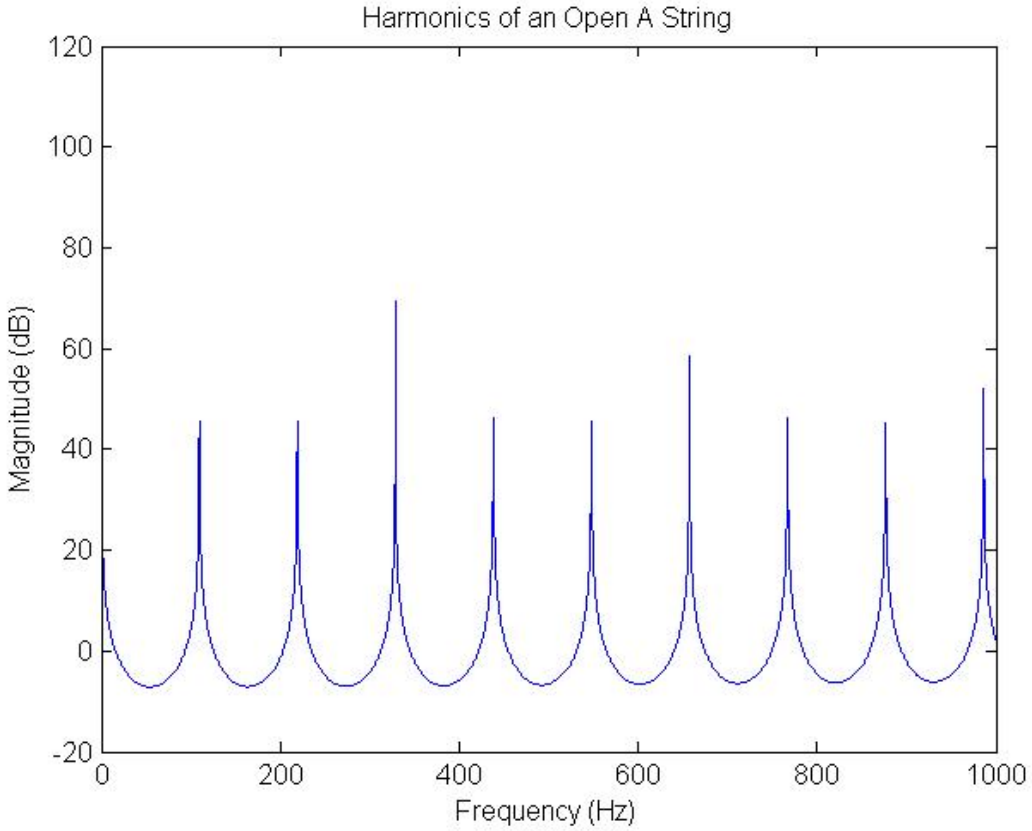


Figure 8.2: The output of the Karplus-Strong algorithm with its characteristic comb shape. Notice the rapid attenuation of all other frequencies not near a harmonic.

8.3 ADSR[3]

8.3.1 ADSR

Attack-Decay-Sustain-Release (ADSR)

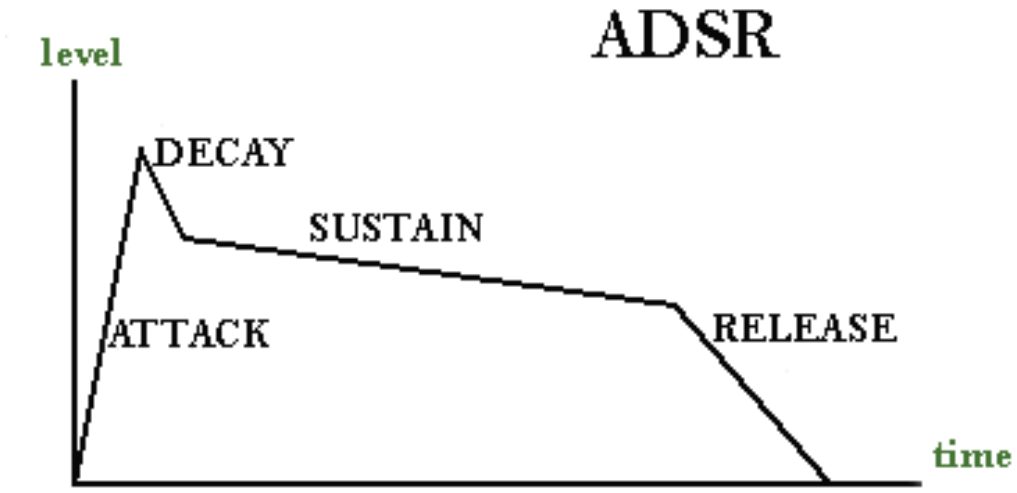


Figure 8.3: The ADSR curve models different instruments by their temporal characteristics. Instruments have varying degrees of the abruptness of the attack, the initial decay in sound, how long the sound resonates for without appreciable attenuation, and how quickly the sound fades away at the end.

8.3.1.1 Introduction

ADSR stands for attack, decay, sustain, and release and is used to model the timbre of an instrument. The timbre or tone quality is determined by various factors such as the way the sound is produced and the material of the instrument. Different families of instruments have their own characteristic ADSR profiles. The attack refers to the phase in which the sound is initiated. This could be the fast attack of a strum on a guitar or the slower one of a pipe organ. The decay phase occurs immediately after the attack impulse and describes how rapidly the sound dies. Some instruments like a drum have extremely fast decay and the sound is virtually nonexistent after the attack. The sustain profile of an instrument refers to how long the sound resonates for when it is played. String instruments such as a violin have an extremely long sustain because the violin's sound box is receiving constant vibrational energy from the bowed string. Finally, the release phase describes how rapidly the sound fades away once the instrument is not being played.

[3]This content is available online at <http://cnx.org/content/m19007/1.2/>.

Guitar ADSR Profile

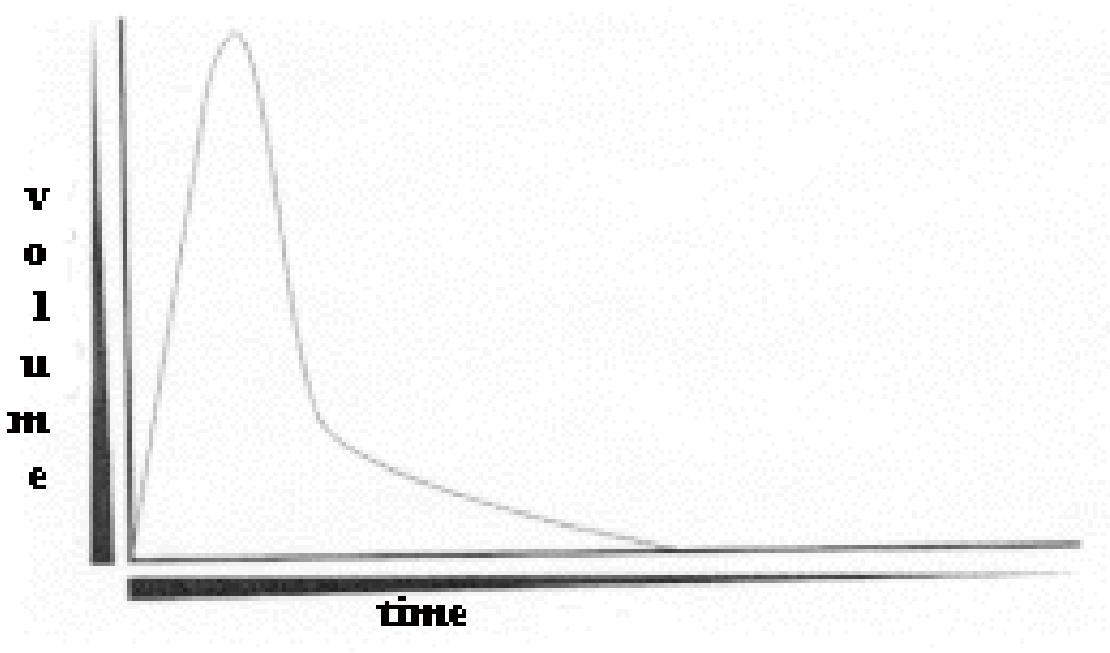


Figure 8.4: The guitar has a fairly abrupt attack due to the method of playing through plucks and strums. The rapid decay is a result of the nonharmonic frequencies fading quickly leaving only the fundamental frequency and its harmonics. The sustain and release phases are merged in this case since the sound just fades away slowly.

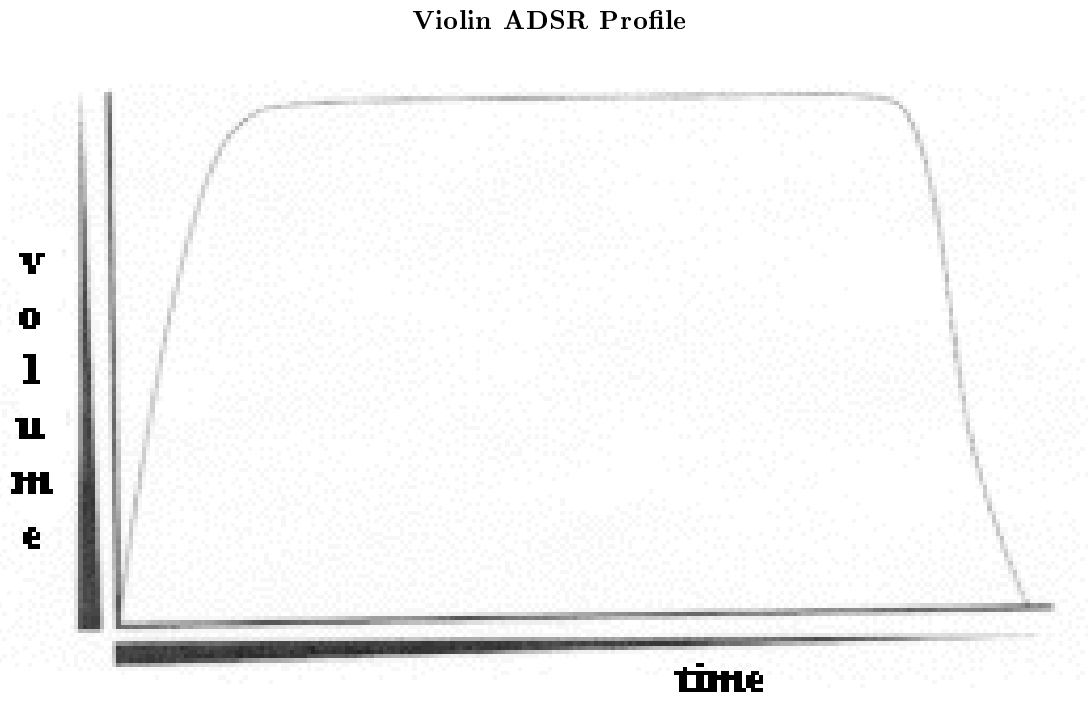


Figure 8.5: The violin has a slightly slower attack than the guitar but the other phases are drastically different. Because a violin is bowed to produce sound, there is virtually no decay and a very long sustain. Constant vibrational energy is delivered to the violin's sound box and the sound only fades once the bowing stops.

8.3.1.2 Implementation

In order to synthesize different sounding instruments, the ADSR envelope could be applied directly to the output of the Karplus-Strong algorithm. Since the algorithm models string and certain percussion instruments, there were limitations on the diversity of instruments that could be synthesized using this technique. After modeling an instrument's temporal characteristics with an ADSR envelope, one could apply it to the output of the Karplus-Strong by point-wise multiplication. By using ADSR, we were able to manipulate the guitar sounding output of the Karplus-Strong algorithm to sound like different instruments such as an organ or a bell.

Applying ADSR to Karplus-Strong

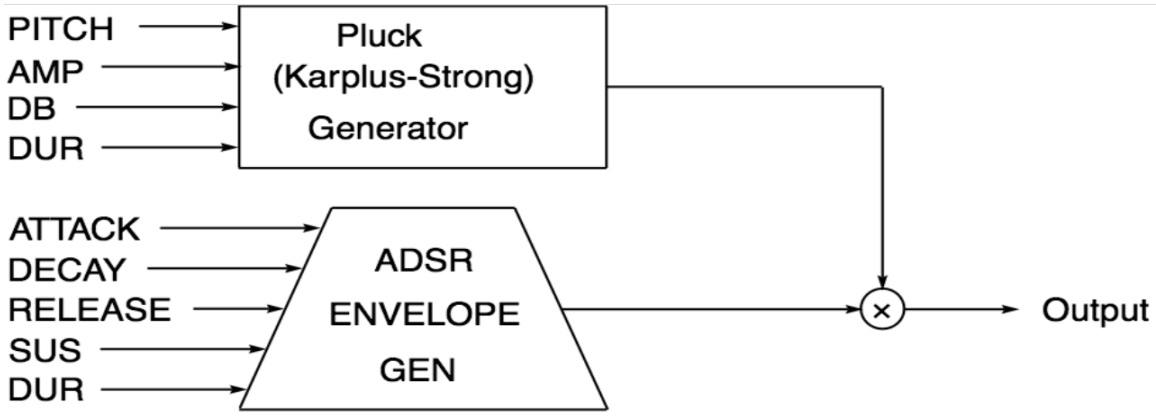


Figure 8.6: Synthesis of different instruments in an instrument family can be achieved through applying ADSR concepts to the Karplus-Strong algorithm. The output of the Karplus-Strong is point-wise multiplied with the modeled ADSR envelope to produce the final output sound.

8.4 Results[4]

8.4.1 Results

8.4.1.1 Karplus-Strong Algorithm

The Karplus-Strong algorithm worked extremely well for producing realistic guitar sounds. We took advantage of this and synthesized chords as well simply by adding the outputs of the individual notes together. To simulate the physical strum of a guitar, we inserted a delay between each of the six strings of the guitar so that they would sound slightly later in time. An electric guitar sound could be easily synthesized by adding some noise to the output of the Karplus-Strong algorithm. The end product was a very pleasing imitation of a guitar.

[4] This content is available online at <http://cnx.org/content/m19008/1.2/>.

Spectrum of a Strummed Chord

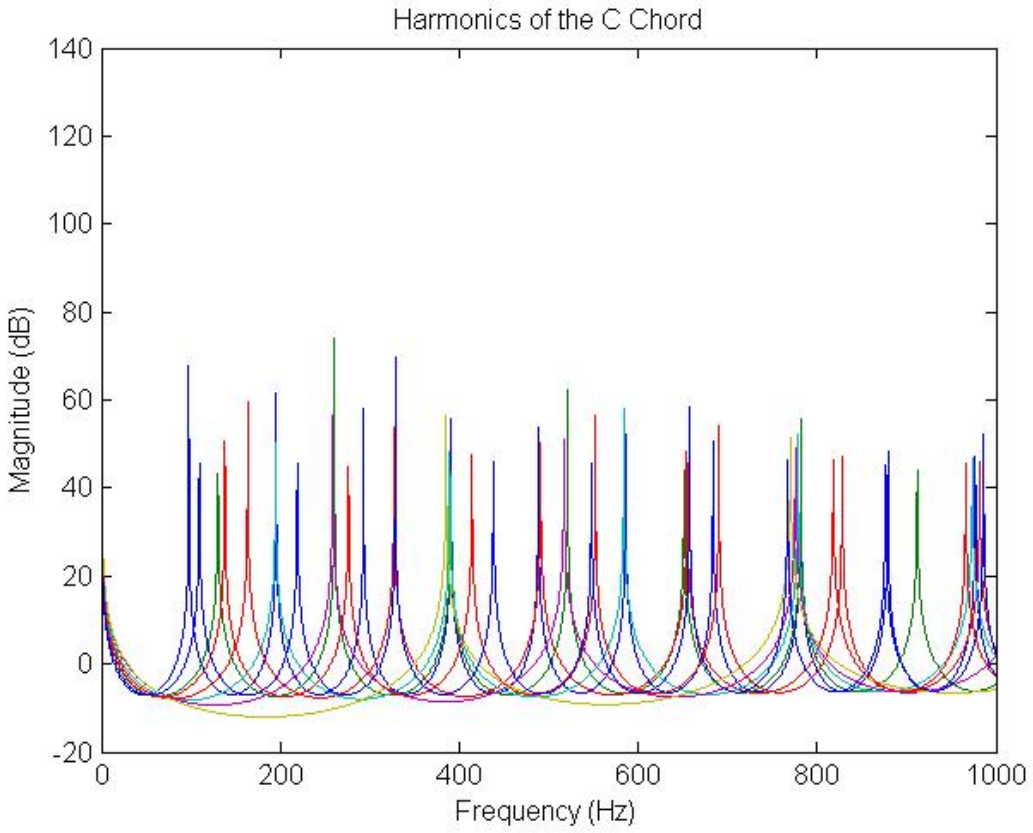


Figure 8.7: The result of combining the individual notes generated by the Karplus-Strong algorithm together to form a chord.

8.4.1.2 ADSR

Applying the ADSR envelope to the output of Karplus-Strong allowed us to alter the guitar sounds into ones that sounded like an organ, bell, and the pluck of a harpsichord piano. We created the ADSR curves through a mixture of research and our own intuition from having heard each instrument.

8.5 Random Music Generator[5]

8.5.1 Random Music Generator

8.5.1.1 Introduction

Our original idea was to be able to generate random music once the instrument sounds were synthesized. Unfortunately, due to time constraints, the best we could do was to have a program randomly pick chords and note durations from an input library. The resulting output was, as expected, random chords and notes that most people would probably not consider as music. How does one teach a program how to write music? One technique is the Markov chain.

8.5.1.2 Markov Chains

Teaching a program music theory so that it can create music would be an extremely tedious task. You would have to teach chord structure, common progressions, the different musical styles, and so on. What if you could give the program examples of pieces you considered to be music and ask it, "write something like that for me." This is essentially how our Markov chain would work. The principle behind Markov chains in music is to generate a probability table to determine what note should come next. By feeding the program an example piece of music, the program can analyze the piece and create a probability table to determine which notes are more likely follow a given note. Below is an example of a first order Markov chain.

Example of First Order Markov Chain Table for Music

	A	A#	B	C	D	E	F	G	G#
A	4/19		3/19		2/19	1/19		6/19	3/19
A#	1								
B	7/15		1/15	4/15		3/15			
C			6/15	3/15	6/15				
D				3/11	3/11	5/11			
E	4/19	1/19		3/19		5/19	4/19	1/19	1/19
F			1/5			1/5		3/5	
G	1/5		1/5	2/5				1/5	
G#			3/4			1/4			

Figure 8.8: The entries in each box indicate the probability or likelihood of the note in the column label to follow the note in the row label. Blank boxes indicate 0 probability, meaning that note will never be the next note given your current note. For example, if the current note is A#, the next note chosen will always be A.

[5]This content is available online at <http://cnx.org/content/m19009/1.2/>.

With the probability table, one can generate random notes that still has some musical structure to it. By constructing a similar table for beats or note durations, one can complete the first order Markov chain for music generation. Increasing to a second order Markov chain simply means constructing a larger probability table where the row headings are now pairs of notes. The program will choose the next note according to the probability table, consequently updating the current note pair to include the newest note. The idea behind determining what order Markov chain to use is a balance between ensuring that the program has enough musical structure and allowing it enough freedom for creativity.

8.6 Extension[6]

8.6.1 Extension

8.6.1.1 What Next?

Having synthesized a few instruments and come up with a method to generate music randomly, the next step would be to include different families of instruments by trying other algorithms to physically model them. This would expand the library of instruments for our synthesizer to make it more interesting. Our long-range goal is to be able to analyze music input such as a voice and generate random musical accompaniment. The inspiration for this project was the research done at Microsoft in developing a program called MySong that automatically chooses chords to accompany a singer. This allows casual singers to create songs complete with accompaniment on their own. The project helped us learn many of the tools we would need to build a program similar to MySong.

Link[7] to Microsoft's MySong page.

[6] This content is available online at <http://cnx.org/content/m19010/1.2/>.
[7] http://research.microsoft.com/en-us/um/people/dan/mysong/

Chapter 9

Selective Listening: Drown Out the Noise

9.1 Selective Listening: Drown Out the Noise- Introduction[1]

9.1.1 Introduction

In common practice, there are few situations in which one can record a single audio source by itself; however, in signal analysis it is often useful to operate on just one component at a time. Consider, for example, what has come to be known as the Cocktail Party problem. A recording of a Cocktail party contains multiple sources, but most recording devices and formats combine all the sources into one signal. Home video makers might want to isolate a voice for a digital scrapbook, or a team of students might want to separate all the speakers in order to automatically generate a transcript of a conversation and properly attribute the lines to individual sources. There are also several situations in which one might run spectral analysis on just one person's voice.

There are a large number of situations in which it is beneficial to isolate individual signals either for individual playback or for individual analysis. We aim to explore the methods of isolating individual sources by developing techniques that use Independent Component Analysis tools in MATLAB.

9.2 Selective Listening: Drown Out the Noise- FICA Past Work[2]

9.2.1 FICA Past Work

The process of singling out individual components is known as Independent Component Analysis (ICA). Research and Development at the Helsinki University of Technology has led to the distribution of a Fast ICA package for MATLAB, which we used for most of our work. A 301 research group from 2007 effectively demonstrated that FastICA neatly separates signals that have been mixed in MATLAB, but failed in situations where mixing occured using microphones that were exposed to multiple sources at once. This latter setup is a scenario that needed to work in order to improve the effectiveness of ourmethod in the field.

[1]This content is available online at <http://cnx.org/content/m18962/1.1/>.
[2]This content is available online at <http://cnx.org/content/m18974/1.1/>.

9.3 Selective Listening: Drown Out the Noise- FICA Assumptions[3]

9.3.1 FICA Assumptions

The fastICA algorithm is founded upon several important assumptions:

1. The sources are all independent random variables: i.e. P(AB)= P(A)P(B)
2. The sum of signals into a microphone is linear.
3. The signal energy is finite
4. The mathematical model is represented as y=Hx where y is the observed mixed signal, H is the square mixing matrix, and x is the source. It is important to note, as will be discussed later, that this assumption states that the mixing matrix is multiplicative with the source.
5. All signals are recorded simultaneously, analogous to the Cocktail Party problem.
6. The number of sources and the number of microphones is equal. The fastICA will find as many independent component sources as there are microphones, which input the mixed signals.

All of these assumptions must be met in order for the fastICA algorithm to successfully isolate the independent sources from one another.

9.4 Selective Listening: Drown Out the Noise- STFICA/Whitening Differences[4]

9.4.1 STFICA/Whitening Differences

In 2007, a 301 Research Group explored how FASTICA works by demonstrating how well it works in its ideal form—when mixing is handled in MATLAB. For use in more general situations, however, Fast ICA must be compatible with every day recording equipment exposed to multiple sources. Mixing in MATLAB creates a simple mixing matrix of the form y=Hx. Given the mixed outputs y, FastICA generates the mixing matrix, H, and its inverse, then multiplies y by the inverse mixing matrix to produce a vector of the original source signals.

Our model accounts for a convolutive mixing matrix that models the effects of the channel and recording equipment. For example, room characteristics can affect echo and dampening, and even the best microphones have an impulse response that does not perfectly model delta pulse. Accordingly, a convolutive model of our system is the most appropriate for our system.

This changes our equations to: y=H*x

where H is a circulant matrix, meaning that each row vector is shifted one to the right relative to the previous row vector. This matrix implements circular convolution.

Sometimes signals that appear to be independent at first glance will, after analysis, have properties of dependence. In order to maximize the effectiveness of component analysis, it makes sense to make signals appear even more independent through a process of decorrelation known as prewhitening, which decorrelates the signals in both space and time.

A signal is white if its values are statistically independent–knowing some values of a signal does not reveal information about other values. Consider white noise, for example. Knowing any value of a white signal gives you no additional information about another.

Because one of the assumptions of FastICA is that the individual components are independent variables, a pre-whitening process applied to the mixed signals improves the signal separation during the ICA process. Originally, the FastICA algorithm included one stage of pre-whitening. This was not sufficient, however, because we needed to add additional stages to achieve better decorrelation with different numbers of sources. The Spatio-Temporal FastICA tools had begun to explore whitening, so we began to look into their work for inspiration.

[3]This content is available online at <http://cnx.org/content/m18975/1.2/>.
[4]This content is available online at <http://cnx.org/content/m18976/1.1/>.

9.5 Selective Listening: Drown Out the Noise- STFICA Implementation[5]

9.5.1 STFICA Implementation

Dr. Scott Douglas, an SMU signals professor, wrote the STFICA code that is well known for its ability to determine demixing matrices that are not simple scalar additions. The key to analyzing audio recordings with the FastICA method is to implement convolutive matrices along with prewhitening—convolution is what occurs between the source and the microphone with the air and the soundwave. The benefit to prewhitening is that it allows us to prepare the recorded signal in such a fashion that they will more accurately fit the Fast ICA model assumptions. A prewhitening stage is already implemented in FastICA as part of the whole package but only in one stage, whereas the STFICA allows one to set as many prewhitening stages as one needs. Therefore, fastICA code was used except for the unmixing matrix algorithm, which we substituted with the one from Douglas's STFICA. This carried the advantage of giving us more control over how many prewhitening stages we were able to implement and creating more elegant demixing matrices. There are some imperfections, however, because we do not understand the correlation between the number of prewhitening stages required to treat the recordings and the complexity of the recordings.

```
    % Signal in this demonstration has a 440Hz tone and noise.
% read the two files

a = wavread('demo3_mix1.wav'); % mixed signal 1
b = wavread('demo3_mix2.wav'); % mixed signal 2

% truncate both signals to equal lengths
if(length(a)>length(b))
    a = a(1:length(b),1);
else
    b = b(1:length(a),1);
end

y = [a';b']; % mixed signal matrix

% perform stfical to demonstrate better success of noise removal
% stfical does not prewhiten the signal beforehand so this must be done

[E, D]=pcamat(y,1,2,'off','off');
[whitesig, whiteningMatrix, dewhiteningMatrix] = whitenv(y,E,D,'off');
[s, W] = stfical(whitesig,1,1);
f2 = dewhiteningMatrix*W*s;
```

This code performs all of the necessary steps to implement the STFICA algorithm with one prewhitening stage.

9.6 Selective Listening: Drown Out the Noise- Results[6]

9.6.1 Results

Note: all supporting code needed for the main Matlab codes to work is attached below

[5] This content is available online at <http://cnx.org/content/m18977/1.1/>.
[6] This content is available online at <http://cnx.org/content/m18978/1.1/>.

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/SupportingCode.zip

9.6.1.1 Ideal Case: MATLAB

Like last year, fastICA works well in MATLAB. For example, mixing and then separating a siren and a voice using MATLAB exclusively works quite well as can be seen in the figure below. fastICA in this very ideal environment was able to separate the two mixed signals into the independent sources of a voice, the lower left spectrogram, and a siren, the lower right spectrogram.

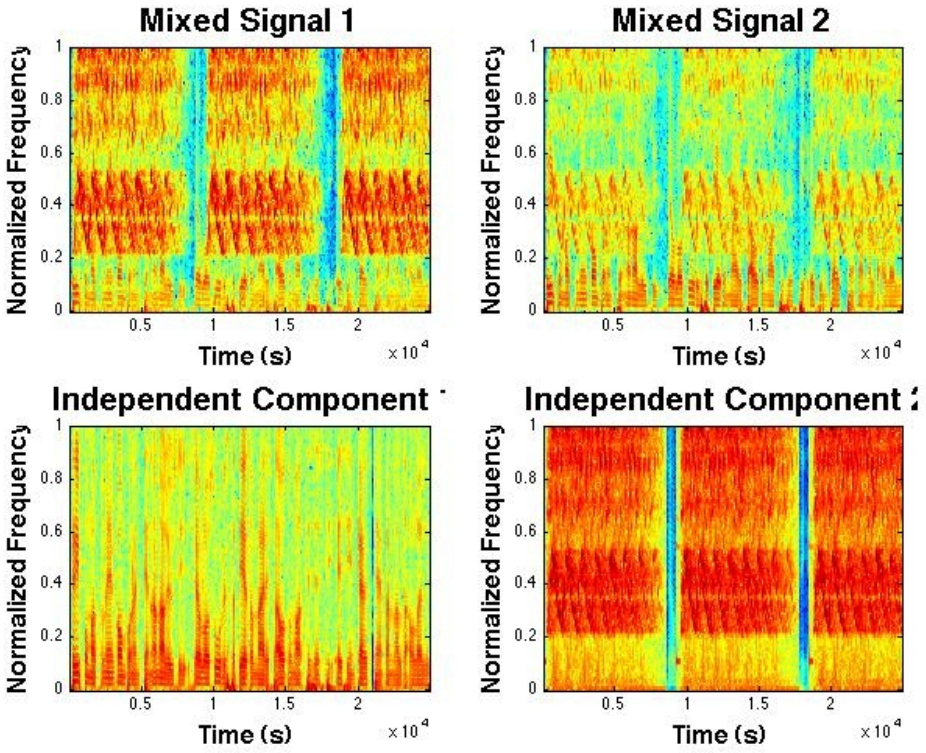


Figure 9.1

For a better grasp of our results, here are the sound files of the mixed signal, isolated siren, and isolated voice, respectively. Also, the MATLAB code used for this trial is also attached after the sound files.

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo1_mix.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo1_sr1.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo1_sr2.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo1.m

9.6.1.2 Real Time Acoustic Mixing Case: fast ICA used

Performing this same feat using an actual microphone, however, fails. For example, we recorded two sources, voice and tone, simultaneously with two microphones which resulted in the spectrograms of the two mixed signals shown below. Once these mixed signals were passed through the fastICA algorithm, the results were terrible source isolation. As you can see by the lower two spectrograms of the figure below, the independent components look almost the same as the mixed signals we started out with.

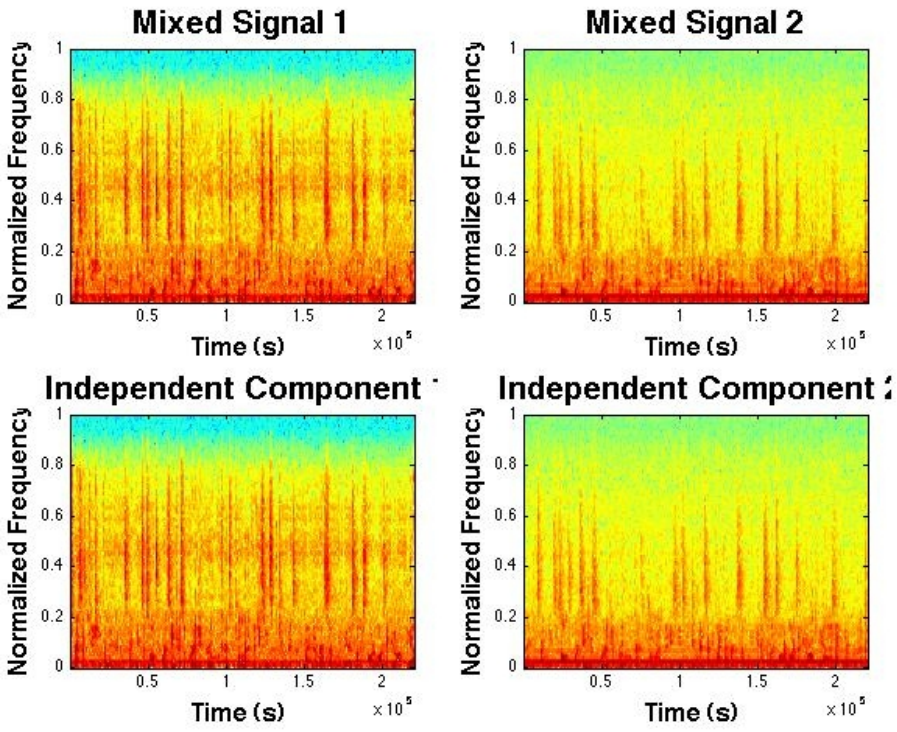


Figure 9.2

Here are the sound files for the two mixed signals and the two "separated sources". The code use to carry out this trial is last.

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo2_mix1.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo2_mix2.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo2_source1.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo2_source2.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo2 code.m

9.6.1.3 Real Time Acoustic Mixing: STFICA or fastICA used

We conjecture two reasons to explain why fastICA is unsuccessful in this scenario.

First, atmospheric and room conditions will change the signal using convolutive operations, rather than the scaling ones that fastICA implements. Second, the characteristic response of the microphones both changes the signals and varies from microphone to microphone, introducing both inaccuracy and imprecision. The original ICA technique, fastICA, does not automatically account for these deviations. Also, although the fastICA does implement a single stage of prewhitening, it may not be enough to alter the input mixed signals so that they look independent of one another in time and space, therefore satisfying the fastICA assumption of independent inputs. So we decided to use the STFICA model in order to account for the convolutive matrix involved and to allow for a user-specifiable number of prewhitening stages.

It was at this time that we experimented with the number of prewhitening stages by setting an iteration level and then watching the output spectrograms for each iteration. Our group could not find a pattern or relation between the iteration number of the prewhitening and the effectiveness of the source isolation, but it was definitely observed that more than stage helps in the source isolation process. Sometimes one iteration would result in some separation, and then the next few iterations did not result in any source separation at all.

Using the STFICA algorithm in some real world cases worked out better than the original fastICA procedure. In one experiment, we produced a pure tone and recoded the source with two microphones. The expected sources that would be isolated were the tone and any ambient noise. The mixed signal of each of the two microphones was passed through the fastICA code and also separately through the STFICA code for comparison. Even with this very simple case, fastICA produced poor results as can be seen in the middle two spectrograms of the output independent components. The spectrograms look almost identical to the original mixed signals that were the inputs. STFICA, on the other hand, separated the pure tone from the

white noise exceptionally well. As can be seen in the last row of the figure, the tone (located on the bottom left) was well isolated from the ambient white noise (spectrogram on the bottom right).

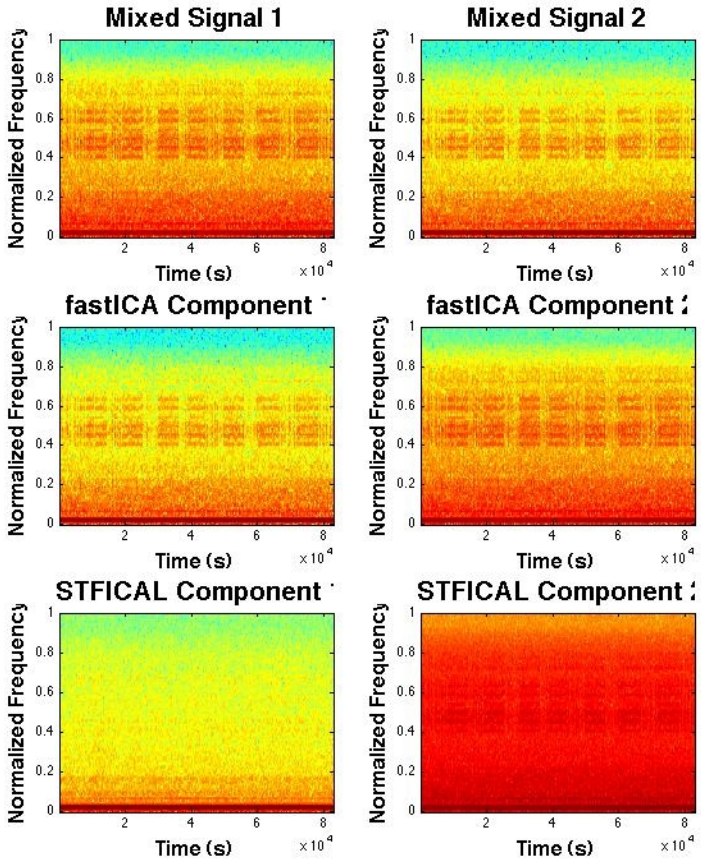


Figure 9.3

Here are the sound files for the two mixed signals, the two "separated signals" produced by fastICA, and the two separated components produced by the STFICA.

This is an unsupported media type. To view, please see
http://cnx.org/content/m18978/latest/demo3_mix1.wav

This is an unsupported media type. To view, please see
http://cnx.org/content/m18978/latest/demo3_mix2.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo3_source1 ICA.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo3_source2 ICA.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo3_source1 STFICA.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo3_source2 STFICA.wav

This is an unsupported media type. To view, please see http://cnx.org/content/m18978/latest/demo3 code.m

In more complicated situations where the sources were multiple human speakers, a human speaker and a tone, or other, we did not achieve the same success. The modified algorithm sometimes made one voice more prominent than the other, but it appeared to be doing filtering in a way that was not achieving the desired result. The success here was not as great as with the simple tone with noise case.

9.7 Selective Listening:Drown Out the Noise - Conclusion[7]

9.7.1 Conclusion

9.7.1.1 Final Thoughts

Our work has shown how important it is to have the right model for the right situation. The fact that the use of convolutive rather than a multiplicative mixing matrix led to success attests to the importance of proper modeling. This was clearly demonstrated by the results of our testing. The Fast ICA model is appropriate for the ideal case of mixing signals in Matlab because all the necessary fast ICA assumptions are met. In particular, the mixing matrix is multiplicative. However, for the case or acoustically mixing signals by recording two sources simutaneously, we saw that fast ICA failed competely to isolate individual sources because a convolutive mixing matrix is involved instead of a multiplicative one. It should be noted that there are a least two stages in which convolution occurs: both in the room and in the air on the way to the microphone, and as the microphone transduces the signal to an electric signal. These processes may be grouped into a single mixing matrix without loss of information because of the transitive property of convolution and because we are not exclusively interested in what the signal of the microphone was, but rather in what the original source sounded like. Using the STFICA algorithm, we were able to at least successfully separate the sources for the tone and noise case, therefore proving that the STFICA model is suited for this secnario. However, we were not successful in more complicated cases such as voice and tone due to the fact that our group did not have enough time to fully undertand and implement the concept of

[7] This content is available online at <http://cnx.org/content/m18979/1.1/>.

variable stages of prewhtening. So there is potential for the STFICA model to be fully applicable to all acoustically mixed sources, but further modifcations as well as a better understand of the model is needed. Another testament to the importance of choosing the correct model for a particular application, is that while it is true that fast ICA failed with acoustically mixed signals, fast ICA works exceptionally well with image applications such as visual noise removal.

9.7.1.2 Steps Toward Improvement

Since our team did not yet achieve exceptional separation of more complicated acoustic signals, such as the voice and tone case, there are some steps to take to improve our results. For example, there must be a refinement of the prewhitening process, which requires additional study on the part of our group members. With this better understanding of the prewhitening concept and process, we expect to gain a better comprehension of the number of required prewhitening stages to make the given mixed signals become more independent of one another in time and space. This should in effect significantly help in the demixing of any complicated mixed signals to its individual sources with as much success as with the tone and noise case. Another method for improvement includes the use of expected behavior in order to better isolate the signal sources. For exmaple, if we know that a man and a woman are speaking, we should be able to tell our algorithm some additional information about the expected spectra of our speakers.

9.8 Selective Listening: Drown Out the Noise- Applications[8]

9.8.1 Applications

Given our results for the STFICA, we are confident that with more knowledge of the prewhitening stage we could effectively separate sources from far more complicated signals such as voice, music, noise, etc. Our original goal was to utilize the FastICA to separate voice from music and improve speech recognition. That is feasible if recording the live audio, but it seems ill-posed if attempted from music files and speakers. There are, however, many other potential avenues such as:

- Effective noise filtering
- Speech recognition improvement
- EKG noise removal
- Surveillance
- Acoustic source separation

For the FastICA, though not effective in acoustic environments, there are applications that it is well-suited for such as:

- Digital image separation
- Digital music separation

9.9 Selective Listening: Drown Out the Noise- Acknowledgments[9]

9.9.1 Acknowledgments

We are grateful for all of the help given to us by our professor Dr. Rich Baraniuk, Dr. Scott Douglas from SMU for doing most of the work by writing the STFICA code for us to implement, Aapo Hyvärinen at Helsinki University of Technology for creating the FastICA algorithm and coding, and Chinmay Hedge for mentoring us on the Fast ICA and some of the more advanced topics of the STFICA.

Group Members:

[8] This content is available online at <http://cnx.org/content/m18980/1.1/>.
[9] This content is available online at <http://cnx.org/content/m18981/1.1/>.

- Jon Stanley
- Jose Hernandez
- Ricardo Frank Barrera
- Jacob Holladay Poteet

Chapter 10

Signal Denoising using Wavelet-based Methods

10.1 Signal Denoising using Wavelet-based Methods[1]

Signal Denoising using Wavelets-based Methods Ioannis Kougioumtzoglou, Isaac Hernandez-Fajardo, Georgios Evangelatos
and Xin Ming.
George R. Brown School of Engineering, Rice University
Houston, TX - USA

10.1.1 Introduction

The basic idea which lies behind wavelets is the representation of an arbitrary function as a combination of simpler functions, generated as scaled and dilated versions of a particular oscillatory "mother" function.

Late Jean Morlet, a geophysical engineer, introduced the term "wavelet" while attempting to analyze signals related to seismic data. The mathematical formulation of the wavelet transform and its inverse was rigorously established by Grossman and Morlet citep([5]). Since then, ideas from diverse scientific fields have resulted in developing wavelets into a powerful analysis tool.

The term **wavelet** is often used to denote a signal located in time with a concentrated amount of energy citep([1]). This "mother" wavelet is used to generate a set of "daughter"functions through the operations of scaling and dilation applied to the mother wavelet. This set forms an orthogonal basis that allows, using inner products, to decompose any given signal much like in the case of Fourier analysis. Wavelets, however, are superior to Fourier analysis for time information is not lost when moving to the frequency domain. This property makes them suitable for applications from diverse fields where the frequency content of a signal as well as the energy's temporal location is valuable.

The wavelets application of interest for this work is their use for data analysis, specifically for signals denoising. Denoising stands for the process of removing noise, i.e unwanted information, present in an unknown signal. The use of wavelets for noise removal was first introduced by Donoho and Johnstone citep([4]). The shrinkage methods for noise removal, first introduced by Donoho citep([3]), have led to a variety of approaches to signal denoising.

This work presents a revision of the wavelets basic theory. Definitions of mother wavelets, wavelets decomposition and its advantage over Fourier analysis are discussed in Section 1. Section 2 presents a summary of basic methods developed for noise removal. Their main features and limitations are presented, and a comparison study taken into account computational efficiency is performed. Section 3 introduces an

[1] This content is available online at <http://cnx.org/content/m18931/1.2/>.

example application for denoising methods. A given function contaminated with Gaussian additive noise is used as testbed for the described methods. Conclusions about the performance of the denoising procedures and the utility of using wavelet decomposition for this type of problem are presented.

10.1.2 Wavelet Decomposition Basics

10.1.2.1 Historical note

The term **wavelet** was first introduced by Jean Morlet while working on the analysis of signals for seismic analysis on oil-related projects. Before Morlet's work remarkable contributions were developed by Haar citep([6]) and Zweig in 1975. After the work of Morlet and Grossmann on the definition of the continous wavelet transform (CWT), several developings have followed. The work of researchers as Stromberg, Daubechies, Mallat and Newland, among others, has pushed forward the theoretical frontiers of wavelets-based orthogonal decomposition and also augmented the scope of possible application fields.

10.1.2.2 Basic concepts

A review of basic concepts in the wavelets framework is presented in next lines. This review is based upon burrus1988.

The term **wavelet** is mostly used for denoting a particular wave whose energy is concentrated in a specific temporal location. A wavelet is therefore a known signal with some peculiar characteristics that allow it to be employed for studying the properties of other signals simultaneously in the frequency and time domains. An typical plot of a wavelet is shown in Figure 10.1

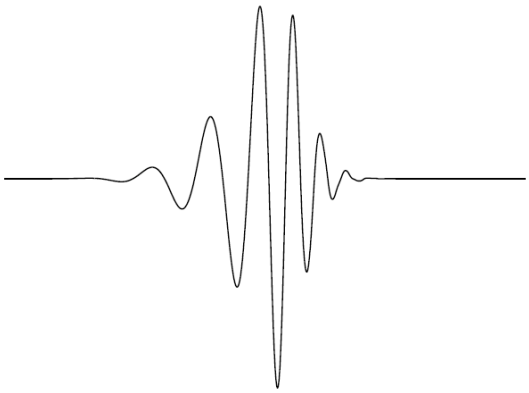


Figure 10.1: Daubechies wavelet $\psi D20$ citep([1])

Based on a particular wavelet, it is possible to define a **wavelet expansion**. A wavelet expansion is the representation of a signal in terms of an orthogonal collection of real-valued functions generated by applying suitable transformations to the original given wavelet. These functions are called "daughter" wavelets while the original wavelet is dubbed "mother" wavelet, acknowledging its function as source of the orthogonal collection. If $f(t)$ is a given signal to be decomposed, then it is possible to represent the signal as:

$$f(t) = \sum_i a_i \psi_i(t) \tag{10.1}$$

In equation (10.1), $\psi_i(t)$ are the orthogonal basis functions, and the coefficients a_i can be found through the inner product of $f(t)$ and the functions $\psi_i(t)$ (Equation (10.2)).

$$a_i =< a_i, \psi_i(t) >= \int f(t)\,\psi_i(t)\,dt \tag{10.2}$$

The previous equations represent the general formulation for orthogonal decomposition, and so they are the same equations discussed in the particular case of Fourier analysis. In the case of wavelets expansion, and consequently with their definition, the wavelets basis functions have two integer indexes, and Equation (10.3) must be rewritten as

$$f(t) = \sum_j \sum_k a_{j,k}\psi_{j,k}(t) \tag{10.3}$$

Equation (10.3) is the **wavelet expansion** of $f(t)$ while the collection of coefficients $a_{j,k}$ is the discrete wavelet transform (DWT) of $f(t)$.

10.1.2.3 Characteristic of wavelet expansions

The properties and hence advantages of a familiy of wavelets depend upon the mother wavelet features. However, a common set of features are shared by the most useful of them citep([1]).

1. A wavelet expansion is formed by a two-dimensional expansion of a signal. It should be noticed that the dimension of the signal itself is not determinant in the wavelet representation.
2. A wavelet expansion provides a dual time-frequency localization of the input signal. This implies that most of the energy of the signal will be captured by a few coefficients.
3. The computational complexity of the discrete wavelet transform is at most $O(nlog(n))$ i.e. as bad as for the discrete Fourier transform (DFT) when calculated using the Fast Fourier Transform (FFT). For some particular types of wavelets, the complexity can be as low as $O(n)$.
4. The basis functions in a wavelet expansion are generated from the mother wavelet by **scaling** and **translation** operations. The indexing in two dimensions is achieved using this expression:

$$\psi_{j,k}(t) = 2^{j/2}\psi\left(2^j t - k\right) \qquad j,k \in \mathbb{Z} \tag{10.4}$$

5. Most wavelets basis functions satisfy **multiresolution** conditions. This property guarantees that if a set of signals can be represented by basis functions generated from a translation $\psi(t-k)$ of the mother wavelet, then a larger set of functions, including the original, can be represented by a new set of basis functions $\psi(2t-k)$. This feature is used in the algorithm of the **fast wavelet transform**, FWT the equivalent of the FFT algorithm for wavelets decomposition.
6. The lower resolution coefficients can be calculated from the higher resolution coefficients using a **filter bank** algorithm. This property contributes to the efficiency of the calculation of the DWT.

10.1.3 Denoising with Wavelets

If $y(t)$ is an empirically recorded signal with and underlying description, $g(t)$, a model for the noise addition process transforming g(t) into y(t) is described by Equation (10.5).

$$y_i = g(t_i) + \sigma\varepsilon_i, \quad i = 1, ..., n \tag{10.5}$$

where ε_i are independent normal random variables $N(0,1)$ and σ represents the intensity of the noise in $y(t)$. Using this model, it follows that the objective of noise removal is, given a finite set of y_i values, reconstruct the original signal g without assuming a particular structure for the signal.

The usual approach to noise removal models noise as a high frequency signal added to an original signal. Fourier transform could be used to track this high frequency, ultimately removing it by adequate filtering. This noise removal strategy is conceptually clear and efficient since depends only on calculating the DFT of a given signal. However, there are some issues that must be taken into account. The most prominent of such issues ocurrs when the original signal has important information associated to the same frequency as the noise. When a frequency domain representation of the signal is obtained, filtering out this frequency will induce noticeable loss of information of the target signal.

In cases as the one described, the wavelets approach is more appropiated due to the fact that the signal will be studied using a "dual" frequency-time representation, which allows separating noise frequencies from valuable signal frequencies. Under this approach, noise will be represented as a consistent high frequency signal in the entire time scope and so its identification will be easier than using Fourier analysis.

Once the noise representation is identified, the removal process starts. It has been proven that a suitable strategy for noise removal consists in making the coefficients associated to the noise frequency equal to zero. This statement represents a global perspective for noise removal, different methods for denoising differ in the way these coefficients are tracked and taken out from the representation. The conceptual details of several of these methods are presented in the next sections. The main reference for the methods discussed here is antoniadis2001

Before attempting to describe the methods is convenient to discuss an alternative definition for wavelet representation used for noise removal. First, the description assumes that the representation is achieved using periodised wavelets bases on $[0,1]$. Also, the basis functions are generated by dilation and translation of a compactely supported scaling function ϕ, also called **father wavelet** and a familiar mother wavelet function, ψ. ψ must be associated with an r-regular multiresolution analysis of $L^2(\mathbb{R})$. An advantage of this approach is that generated wavelets families allow integration of different kinds of smoothness and vanishing moments. This features lead to the fact that many signals in practice can be represented sparsely (with few wavelets coefficients) and uniquely under wavelets decomposition. The decomposition expresion using a father and a mother wavelet is depicted in Equation (10.6).

$$g(t) = \sum_{k=0}^{2^{j_0}-1} \alpha_{j_0 k}\phi_{j_0 k}(t) + \sum_{j=j_0}^{\infty}\sum_{k=0}^{2^j-1} \beta_{jk}\psi_{jk}(t), \quad j_0 \geq 0, \quad t \in [0,1] \tag{10.6}$$

where j_0 is a primary resolution level, and $\alpha_{j_0 k}$ and β_{jk} are calculated as the inner products shown in Equations (10.7) and (10.8)

$$\alpha_{j_0 k} = < g, \phi_{j_0 k} > = \int_0^1 g(t)\,\phi_{j_0 k}(t)\,dt, \quad j_0 \geq 0, \quad k = 0, 1, ..., 2^{j_0} - 1 \tag{10.7}$$

$$\beta_{jk} = < g, \psi_{jk} > = \int_0^1 g(t)\,\psi_{jk}(t)\,dt, \quad j \geq j_0 \geq 0, \quad k = 0, 1, ..., 2^{j} - 1 \tag{10.8}$$

When the discrete wavelet transform is used, the coefficients $c_{j_0 k}$, **discrete scaling coefficients** and d_{jk}, **discrete wavelet coefficients** are used instead of the continous parameters $\alpha_{j_0 k}$ and β_{jk}. The discrete parameters can be approximately calculated by applying a $\sqrt{n}$ factor to the continous coefficients.

Finally, when the DWT is applied to Equation (10.5), these expressions are obtained (Equations (10.9) and (10.10)):

$$\hat{c}_{j_0 k} = c_{j_0 k} + \sigma\varepsilon_{jk}, \quad k = 0, 1, ..., 2^{j_0} - 1 \tag{10.9}$$

$$\hat{d}_{jk} = d_{jk} + \sigma\varepsilon_{jk}, \quad j = j_0, ..., J-1, \quad k = 0, ..., 2^{j} - 1 \tag{10.10}$$

10.1.3.1 Classical approach to wavelet thresholding

The original and simpler way to remove noise from a contaminated signal consists in modifying the wavelets coefficients in a smart way such that the "small" coefficients associated to the noise are basically neglected. The updated coefficients can thus be used to reconstruct the original underlying function free from the effects of noise. It is implicit in the strategy that only a few "large" wavelets coefficients d_{jk} are associated with the original signal, and that their identification and elimination of any other coefficients will allow a perfect reconstruction of the underlying signal g. Several methods use this idea and implement it in different ways. When attempting to descrease the influence of noise wavelets coefficients it is possible to do it in particular ways, also the need of information of the underlying signal leads to different statistical treatments of the available information.

In the linear penalization method every wavelet coefficient is affected by a linear shrinkage particular associated to the resolution level of the coefficient. A mathematical expression for this type of approach using **linear** shrinkage is shown in Equation (10.11).

$$\tilde{d}_{jk} = \frac{\hat{d}_{jk}}{1 + \lambda 2^{2js}} \tag{10.11}$$

In Equation (10.11), parameter s is the **known** smoothness index of the underlying signal g, while parameter λ is a smooting factor whose determination is critical for this type of analysis.

It must be said that linear thresholding is adequate only for spatially homogenous signal with important levels of regularity. When homegeneity and regularity conditions are not met nonlinear wavelet thresholding or shrinkage methods are usually more suitable.

donoho1995 and donoho1995b proposed a nonlinear strategy for thresholding. Under their approach, the thresholding can be done by implementing either a hard or a soft thresholding rule. Their mathematical expressions are shown in Equation (10.12) and Equation (10.13) respectively.

In both methods, the role of the parameter λ as a threshold value is critical as the estimator leading to destruction, reduction, or increase in the value of a wavelet coefficient.

$$\delta_{\lambda}^{H}\left(\hat{d}_{jk}\right) = \{ \begin{array}{ll} 0 & if \quad |\hat{d}_{jk}| \leq \lambda \\ \hat{d}_{jk} & if \quad |\hat{d}_{jk}| > \lambda \end{array} \tag{10.12}$$

$$\delta_{\lambda}^{S}\left(\hat{d}_{jk}\right) = \{ \begin{array}{ll} 0 & if \quad |\hat{d}_{jk}| \leq \lambda \\ \hat{d}_{jk} - \lambda & if \quad \hat{d}_{jk} > \lambda \\ \hat{d}_{jk} + \lambda & if \quad \hat{d}_{jk} < -\lambda \end{array} \tag{10.13}$$

Several authors have discussed the properties and limitations of these two strategies; hard thresholding, due to its induced discontinuity, can be unstable and sensitive even to small changes in the data. On the other hand, soft thresholding can create unnecessary bias when the true coefficients are large. Although more sophisticated methods has been introduced to account for the drawbacks of the described nonlinear strategies, the discussion in this report is limited to the hard and soft approaches.

10.1.3.1.1 Term-by-Term Thresholding

One apparent problem in applying wavelet thresholding methods is the way of selecting an appropriate value for the threshold, λ. There are indeed several approaches for specifying the value of the parameter in question. In a general sense, these strategies can be classified in two groups: **global** thresholds and **level-dependent** thresholds. Global threshold implies the selection of one λ value, applied to all the wavelet coefficients. Level-dependent thresholds implies that a (possibly) different threshold value $lambda_j$ is applied for each resolution level. All the alternatives require an estimate of the noise level σ. The standard deviation of

the data values is clearly not a good estimator, unless the underlying response function g is reasonably flat. donoho1995 considered estimating σ in the wavelet domain by using the expression in Equation (10.14).

$$\hat{\sigma} = \frac{median\left(\left|\hat{d}_{J-1,k}\right|\right)}{0.6745}, \quad k = 0, 1, ..., 2^{J-1} - 1 \tag{10.14}$$

10.1.3.1.2 The minimax threshold

donoho1995 obtained an optimal threshold value λ^M by minimizing the risk involved in estimating a function. The porposed **minimax** threshold depends of the available data and also takes into account the noise level contaminating the signal (Equation (10.15)).

$$\lambda^M = \hat{\sigma}\, \lambda_n^* \tag{10.15}$$

Where, λ_n^* is equal to the value of λ satisfying Equation (10.16)

$$\lambda_n^* = \inf_{\lambda} \sup_{d} \{\frac{R_\lambda(d)}{n^{-1} + R_{oracle}(d)}\} \tag{10.16}$$

In Equation (10.16), $R_\lambda(d)$ is calculated following Equation (10.17).

$$R_\lambda(d) = E\left(\delta_\lambda\, \hat{d}\right)^2 \tag{10.17}$$

while $R_{oracle}(d)$ is an operators called **oracle** used to account for the risk associated to the modification of the value of a given wavelet coeficient. Two of these oracles were introduced by donoho1995: **diagonal linear projection** (DLP), and **diagonal linear shrinker** (DLS). The Equations (10.18) and (10.19) show the expressions for the two oracles.

$$R_{oracle}^{DLP}(d) = min\left(d^2, 1\right) \tag{10.18}$$

$$R_{oracle}^{DLS}(d) = \frac{d^2}{d^2 + 1} \tag{10.19}$$

antoniadis2001 provided values of the minimax threshold for both the hard and soft nonlinear thresholding rules. For the soft rule, 1.669 and 2.226 for n equal to 128 and 1024; for the hard rule, 2.913 and 3.497 again for n equal to 128 and 1024.

10.1.3.1.3 The universal threshold

donoho1995 proposed this threshold as an alternative to the minimax thresholds, applied to all the wavelet coefficients. The universal threshold is defined in Equation (10.20).

$$\lambda^U = \hat{\sigma}\, \sqrt{2logn} \tag{10.20}$$

This threshold is easy to remember and its implementation in software is simpler and the optimization problem implicit in the minimax method are avoided. Also, the universal threshold ensures, with high probability, that every sample in the wavelet transform in which the underlying function is exactly zero will be estimated as zero, although the convergance rate (depending in the size of the sample) is slow.

10.1.3.1.4 The translation invariant method

It has been noted that wavelet thresholding with either minimax thresholds or the universal threshold presents some inconvenient features. In particular, in the vicinity of discontinuities, these wavelet thresholds can exhibit pseudo-Gibbs phenomena. While these phenomena are less pronounced than in the case of Fourier analysis and also are present in a local scale, this situation represents a challenge for the thresholding methods.

coifman1995 proposed the use of the translation invariant wavelet thresholding scheme. The idea is to correct mis-alignments between features in the studied signal and features in the basis used for the decomposition. When the signal contains an important number of discontinuities, the method applies a range of shifts to it, and average the results obtained after such transformations.

If a empirical contaminated signal $y[i]$, $(i = 1, ..., n)$ is provided, the tranlation invariant wavelet thresholding estimator is calculated as (Equation (10.21)):

$$\hat{g}^{TI} = \frac{1}{n} \sum_{k=1}^{n} (WS_k)' \delta_\lambda (WS_k y) \tag{10.21}$$

where δ_λ is either the hard of soft thresholding rule, W is the size n orthogonal matrix associated to the DWT, and S_k is the shift kernel defined as:

$$S_k = \begin{pmatrix} O_{k\times(n-k)} & I_{k\times k} \\ I_{(n-k)\times(n-k)} & O_{(n-k)\times k} \end{pmatrix} \tag{10.22}$$

In Equation (10.22), I is the identity matrix and O stands for a zero matrix with dimensions as indicated in the expression.

10.1.3.1.5 The SureShrink Threshold

donoho1995 proposed a procedure to select a threshold value λ_j for every resolution level j. The method uses Steinâ[U+0080][U+0099]s unbiased risk criterion citep([7]) to get an unbiased estimate of the l^2-risk.

In mathematical terms, given a set $X_1, ..., X_s$ of variables distributed as $N(\mu_i, 1)$ with $i = 1, ..., s$, the problem consists in estimate the vector of means with minimum l^2-risk. It turns out an estimator of μ, can be describes as $\hat{\mu}(X) = X + g(X)$, with g a function from $\mathbb{R}^s$ to $\mathbb{R}^s$ is weakly differentiable. With this information, the risk of the estimation can be described as (Equation (10.23)):

$$E_\mu || \hat{\mu}(X) - \mu||^2 = s + E_\mu \left(||g(X)||^2 + 2\nabla g(X) \right) \tag{10.23}$$

with,

$$\nabla g \equiv \sum_{i=1}^{s} \frac{\partial g_i}{\partial x_i} \tag{10.24}$$

If an operator SURE is defined as (Equation (10.25)):

$$SURE(\lambda; X) = s - 2 \cdot \#\{i : |X_i| \leq \lambda\} + [min(|X_i|, \lambda)]^2 \tag{10.25}$$

where the operator $\#\{A\}$ returns the cardinality of the set A, it is found that SURE is an unbiased estimate of the l^2-risk, i.e,

$$E_\mu || \hat{\mu}(X) - \mu||^2 = SURE(\lambda; X) \tag{10.26}$$

Now, the threshold λ is found by minimizing SURE over the set X of give data. Extending this principle to the entire set of resolution levels, an expression for λ_j^s is found (Equation (10.27)):

$$\lambda_j^s = arg \min_{0 \leq \lambda \leq \lambda^U} SURE\left(\lambda; \frac{\hat{d}_{jk}}{\hat{\sigma}}\right), \quad j = j_0, ..., J-1; \quad k = 0, ..., 2^j - 1 \tag{10.27}$$

where λ^U is the universal threshold, and $\hat{\sigma}$ is the estimator of the noise level (Equation (10.14)).

10.1.3.2 Classical Methods: Block Thresholding

Thresholding approaches resorting to term-by-term modification on the wavelets coefficients attempt to balance variance and bias contribution to the mean squared error in the estimation of the underlying signal g. However, it has been proven that such balance is not optimal. Term-by-term thresholding end sup removing to many terms leading to estimation prone to bias and with a slower convergance rate due to the number of operations involved.

A useful resource to improve the quality of the aforementioned balanced is by using information of the set of data associated to a particular wavelet coefficient. In order to do so, a block strategy for thresholded is proposed. The main idea consists in isolating a block of wavelet coefficients and based upon the information collected about the entire set make a decision about decreasing or even entirely discard the group. This procedure will allow faster manipulation of the information and accelerated convergence rates.

10.1.3.2.1 Overlapping Block Thresholding Estimator

cai2001 considered an overlapping block thresholding estimator by modifying the nonoverlapping block thresholding estimator citep([2]). The effect is that the treatment of empirical wavelet coefficients in the middle of each block depends on the data in the whole block. At each resolution level, this method packs wavelet coefficients $\hat{d}_{jk}$ into nonoverlapping blocks (j_b) of length L_0. Following this, the blocks are extended in each direction an amount $L_1 = max\left(1; \left[\frac{L_0}{2}\right]\right)$, generating overlapping blocks (j_B) of augmented length $L = L_0 + 2L_1$.

If $S^2_{(j_B)}$ is the L^2-energy of the empirical signal in the augmented block j_B, the wavelet coefficients in the blocks j_b will be estimated simultaneously using the expression in Equation (10.28)

$$\overset{\Phi}{d}_{jk}^{(j_b)} = max\left(0, \frac{S^2_{(j_B)} - \lambda L \hat{\sigma}^2}{S^2_{(j_B)}}\right) \hat{d}_{jk} \tag{10.28}$$

Once the estimated wavelet coefficients $\overset{\Phi}{d}_{jk}^{(j_b)}$ have been calculated, an estimation of the underlying signal g can be obtained through using the new wavelet coefficients and the unmodified scaling coefficients $\hat{c}_{j_0k}$ in the IDWT. The results from this method (**NeigBlock**) presented in this document used a value of $L_0 = \left[log\left(\frac{n}{2}\right)\right]$ and a value of $\lambda = 4.50524$ as suggested by cai2001.

10.1.3.3 Bayesian Approach to wavelet shrinkage and thresholding

From Equations (10.9) and (10.10) it can be established that the empirical scaling and wavelet coefficients, conditional on their respective underlying coefficients, are independently distributed, i.e:

$$\hat{c}_{j_0k} / \left(c_{j_0k}, \sigma^2\right) \sim N\left(c_{j_0k}, \sigma^2\right) \tag{10.29}$$

$$\hat{d}_{jk}/\left(c_{jk},\sigma^2\right)\sim N\left(d_{jk},\sigma^2\right) \tag{10.30}$$

The Bayesian approach imposes an **apriori** model for the wavelets coefficients designed to capture the sparseness of the wavelet expansion common to most applications. An usual prior model for each wavelet coefficient $\hat{d}_{jk}$ is a mixture of two distributions, one of them associated to negligable coefficients and the other to significant coefficients. Two types of mixtures have been widely used. One of them employs two normal distributions while the other uses one normal distribution and one point mass at zero.

After mathematical manipulation, it can be shown that an estimator for the underlying signal can be written as (Equation (10.31)):

$$\hat{g}_{BR}\left(t\right)=\sum_{k=0}^{2^{j_0}-1}\frac{\hat{c}_{j_0k}}{\sqrt{n}}\phi_{j_0k}\left(t\right)+\sum_{j=j_0}^{J-1}\sum_{k=0}^{2^j-1}\frac{BR\left(d_{jk}|\left(d_{jk},\sigma^2\right)\right)}{\sqrt{n}}\psi_{jk}\left(t\right) \tag{10.31}$$

i.e. the scaling coefficients are estimated by the empirical scaling coefficients while the wavelet coefficients are estimated by a Bayesian rule (BR), taking into account the obtained empirical wavelet coefficient and the noise level.

10.1.3.3.1 Shrinkage estimates based on deterministic/stochastic decompositions

huang2000 proposed a method that takes into account the value of the prior mean for each wavelet coefficient, by introducing a estimator for the parameter into the general wavelet shrinkage model. These authors assumed thatthe undelying signal is composed of a piecewise deterministic portion with an added zero mean stochastic part.

If $\hat{\mathbf{c}}_{j_0}$ is the vector of empirical scaling coefficients, $\hat{\mathbf{d}}_j$ the vector of empirical wavelet coefficients, $\mathbf{c}_{j_0}$ the vector of underlying scaling coefficients, and $\mathbf{d}_j$ the vector of underlying wavelet coefficients, then the Bayesian model (Equation (10.32)):

$$\omega/\left(\beta,\sigma^2\right)\sim N\left(\beta,\sigma^2 I\right) \tag{10.32}$$

with $\omega=\left(\hat{\mathbf{c}}_{j_0},\hat{\mathbf{d}}_{j_0},...,\hat{\mathbf{d}}_{J-1}^{'}\right)^{'}$ and the underlying signal $\beta=\left(\mathbf{c}_{j_0}^{'},\mathbf{d}_{j_0}^{'},...,\mathbf{d}_{J-1}^{'}\right)^{'}$ is assumed to follow an apriori distribution (Equation (10.33))

$$\beta/\left(\mu,\theta\right)\sim N\left(\mu,\Sigma\left(\theta\right)\right) \tag{10.33}$$

where μ is the deterministic mean structure and $\Sigma\left(\theta\right)$ accounts for the uncertainty and value correlation in the underlying signal. Notice that if η following a distribution $N\left(0,\Sigma\left(\theta\right)\right)$ is defined as the stochastic component representing small variation (high frequency) in the signal, then μ can be interpretated as the stochastic component accounting for the large-scale variation in β. So, it is possible to rewrite β as (Equation (10.34)),

$$\beta=\mu+\eta \tag{10.34}$$

Using this model, a shrinkage rule can be established by calculating the mean of β conditional on σ^2 which is expressed as (Equation (10.35)),

$$E\left(\beta/\left(\omega,\sigma^2\right)\right)=\mu+\frac{\Sigma\left(\theta\right)}{\left(\Sigma\left(\theta\right)+\sigma^2 I\right)}\left(\omega-\mu\right) \tag{10.35}$$

10.1.4 Numerical Simulations

10.1.4.1 Description of the Scheme

In order to assess the efficiency and accuracy of the proposed methods, a number of simulations have been conducted. To this aim, data have been generated according to the following scheme

$$y_i = f(x_i) + \varepsilon_i, \{\varepsilon_i\}\ N\left(0, \sigma^2\right) \tag{10.36}$$

where the data $\{x_i\}$ are considered equally spaced in the interval $[0, 1]$. The signal-to-noise ratio has been taken equal to 3. In these simulations the Symmlet 8 wavelet basis has been used. Given the random nature of $\{\varepsilon_i\}$, 100 realizations of the function $\{y_i\}$ have been produced. This has been done in order to apply the comparison criteria to the ensemble average of the realizations. Since the primary goal of the simulations is the comparison of the different denoising methods, the following criteria are introduced:

1. **Root Mean Squared Error**: The mean squared error defined as (Equation (10.37))

$$\frac{1}{N}\sum_{i=1}^{N}\left(\left(f(x_i) - f_{dn}(x_i)\right)^2 \right. \tag{10.37}$$

   is computed for each realization and averaged over the 100 samples. Then, its square root is taken.
2. **Maximum Deviation**: The average over the 100 samples of $\underset{1<i<N}{max} |f(x_i) - f_{dn}(x_i)|$

Computational efficiency has not been chosen as one of the criteria, since it is greatly depended on the individual programming skills of the individual. Therefore, in order to avoid a non-uniform programming approach which could possibly result in misleading conclusions, time efficiency has not been considered.

The test functions $f(x)$ and the sample sizes N have been chosen as the factors of the comparison studies. To this aim, two samples, one of moderate moderate size ($N = 128$) and another of larger size ($N = 1024$) have been considered.

As far as the test functions are concerned, two smooth signals (Figures Figure 10.2 and Figure 10.3) and two discontinuous ones (Figures Figure 10.4 and Figure 10.5) were taken into account. In Figure 10.2, the function consists of the sum of two sinusoids, whereas in Figure 10.3, a time shifted sine is illustrated. Since the signals are smooth, linear methods are expected to be comparable to the nonlinear ones. On the other hand, nonlinear wavelet estimators are expected to perform better for the functions in (Figure 10.4, Figure 10.5). These highly discontinuous signals have been used as examples in donoho1993

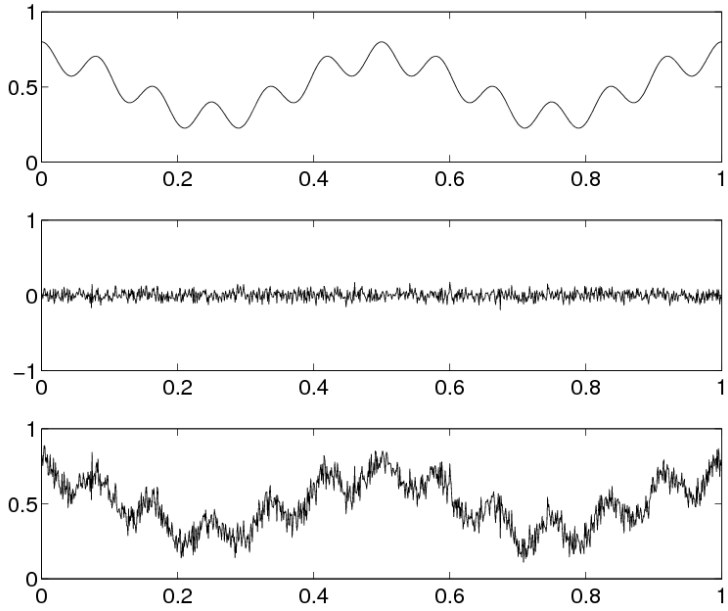


Figure 10.2: Original function with added Gaussian White noise (Wave function)

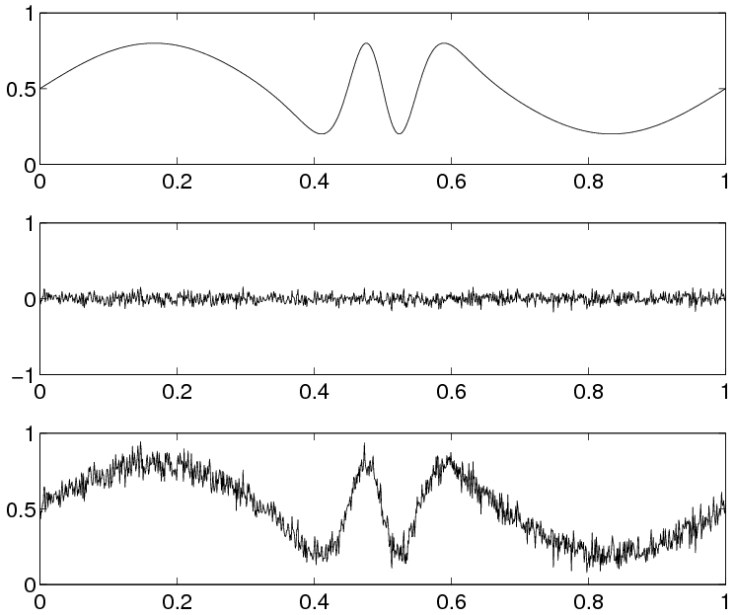


Figure 10.3: Original function with added Gaussian White noise (Time shifted sine function)

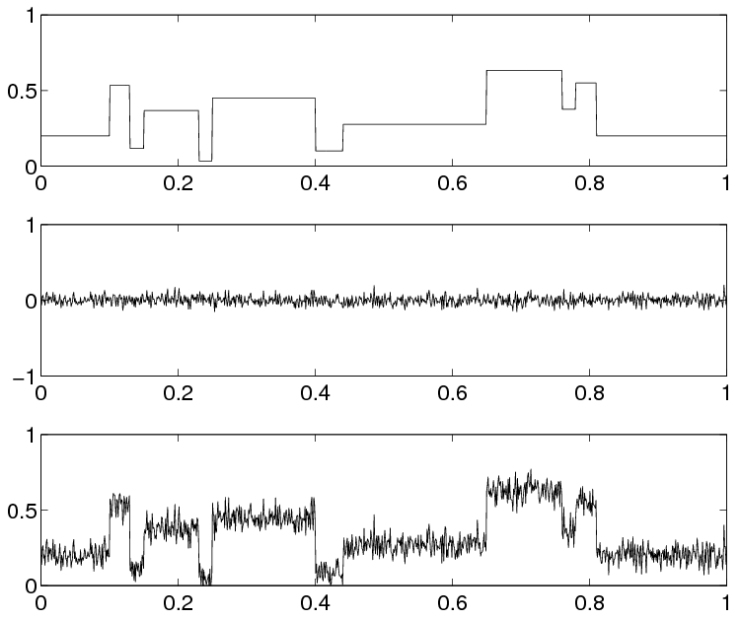


Figure 10.4: Original function with added Gaussian White noise (Blocks function)

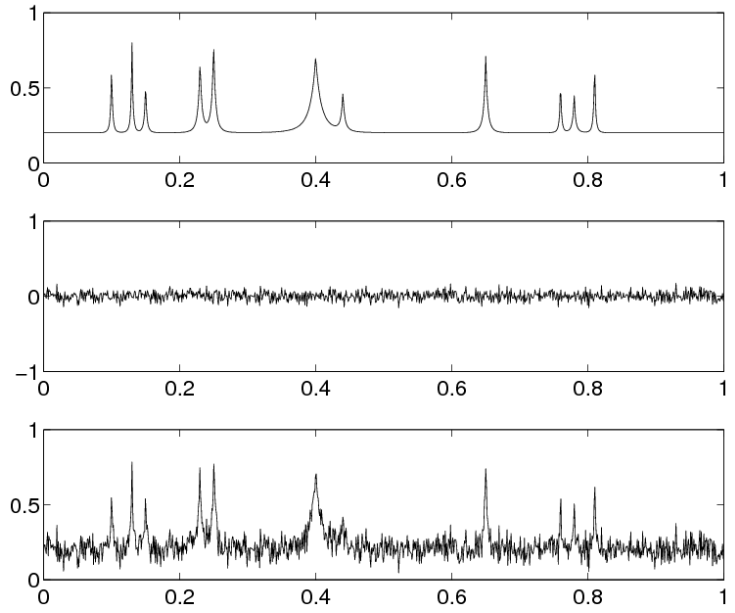


Figure 10.5: Original function with added Gaussian White noise (Bumps function)

10.1.4.2 Results

The following plots, (Figures Figure 10.6 - Figure 10.13), illustrate the denoising performance for the 10 methods used. Each integer corresponds to a particular method as follows

1. VisuShrink-Hard: Universal threshold with hard thresholding rule
2. VisuShrink-Soft: Universal threshold with soft thresholding rule
3. SureShrink: SureShrink threshold
4. Translation-Invariant-Hard: Translation invariant threshold with hard thresholding rule
5. Translation-Invariant-Soft: Translation invariant threshold with soft thresholding rule
6. Minimax-Hard: Minimax threshold with hard thresholding rule
7. Minimax-Soft: Minimax threshold with soft thresholding rule
8. NeighBlock: Overlapping block thresholding (with $L_0 = [logn/2]$, $\lambda = 4.50524$)
9. Linear Penalization: Term-by-term thresholding using linear shrinking
10. Deterministic/Stochastic: Bayesian thresholding method for shrinkage estimates

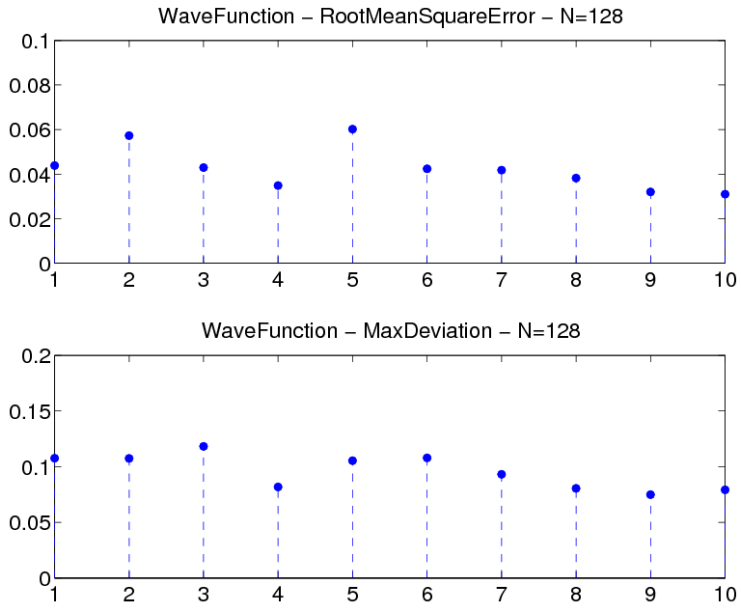


Figure 10.6: Comparison Study using Wave function. N=128

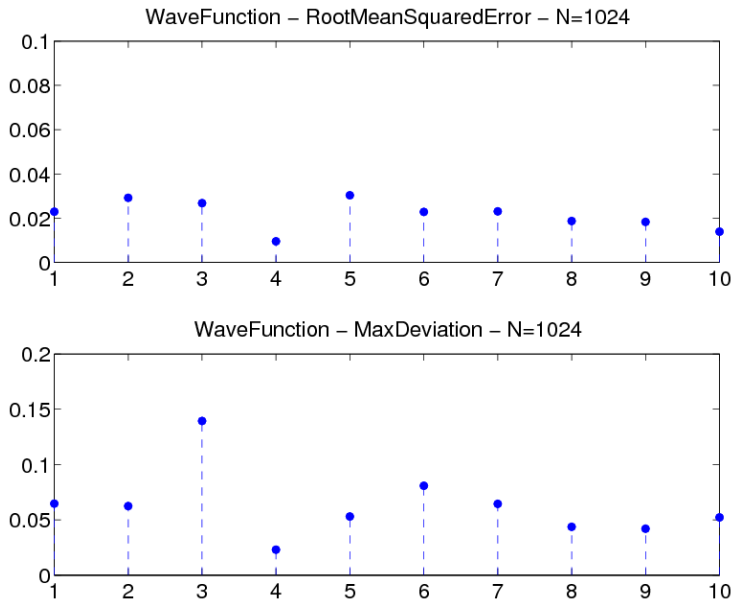


Figure 10.7: Comparison Study using Wave function. N=1024

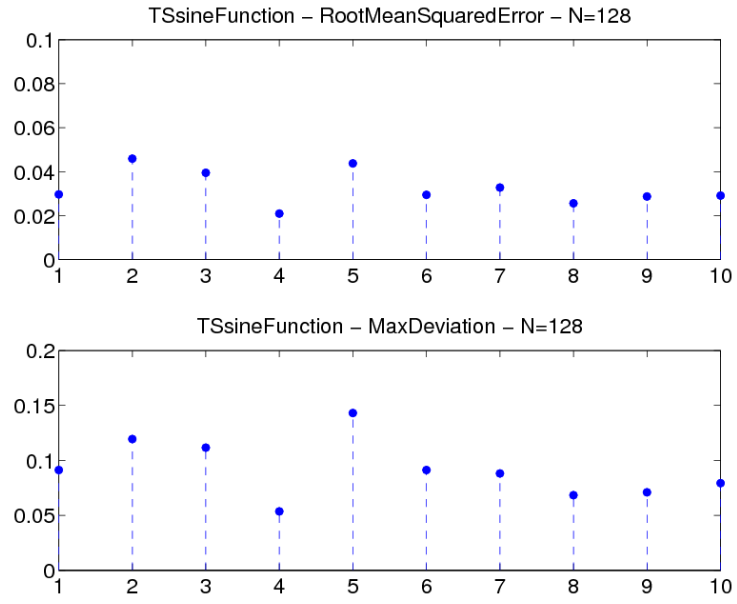


Figure 10.8: Comparison Study using Time-shifted sine function. N=128

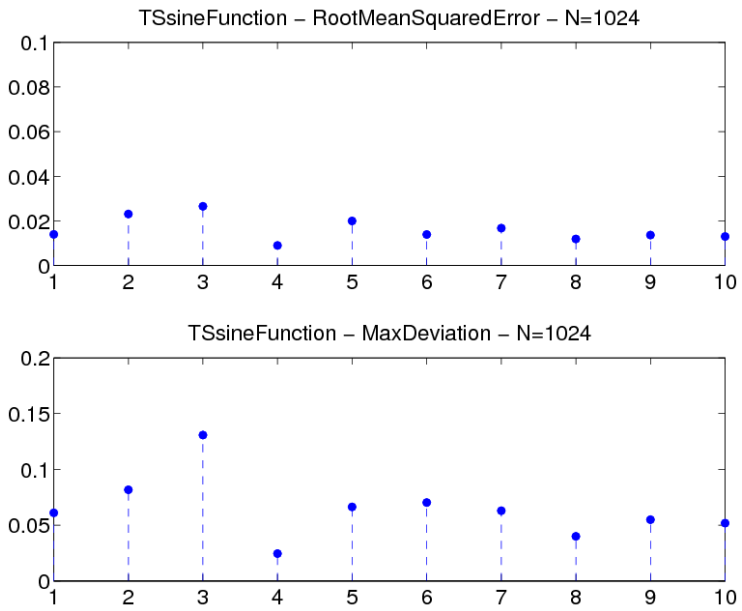


Figure 10.9: Comparison Study using Time-shifted sine function. N=1024

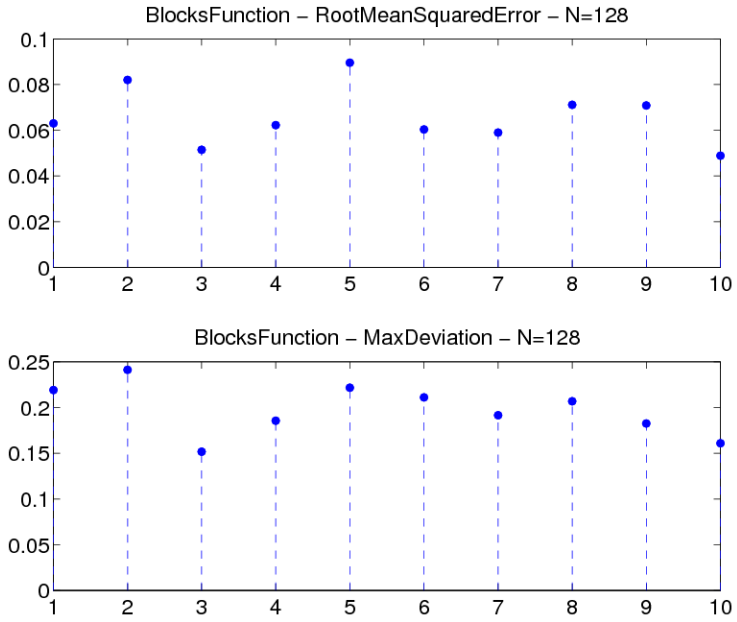


Figure 10.10: Comparison Study using Blocks function. N=128

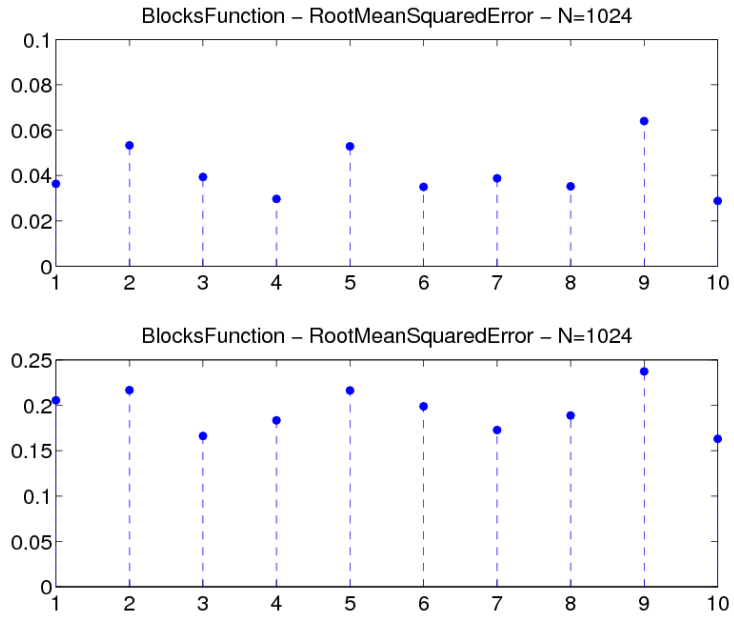


Figure 10.11: Comparison Study using Blocks function. N=1024

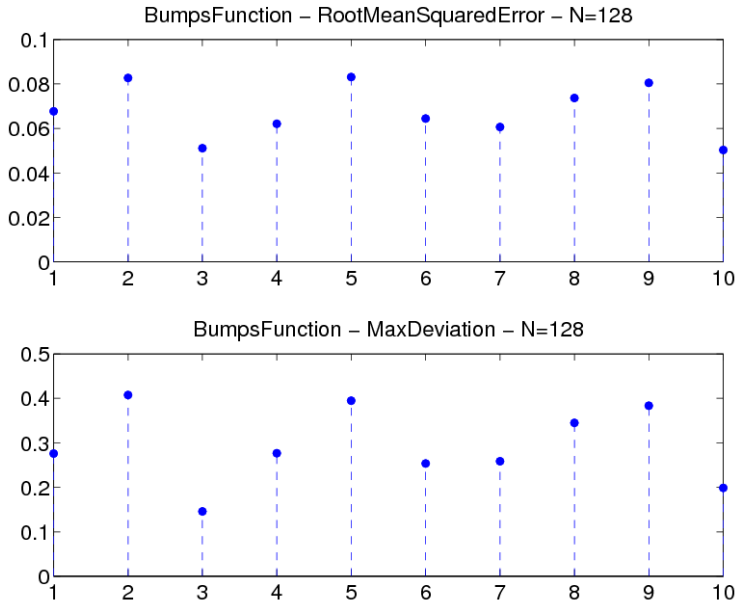


Figure 10.12: Comparison Study using Bumps function. N=128

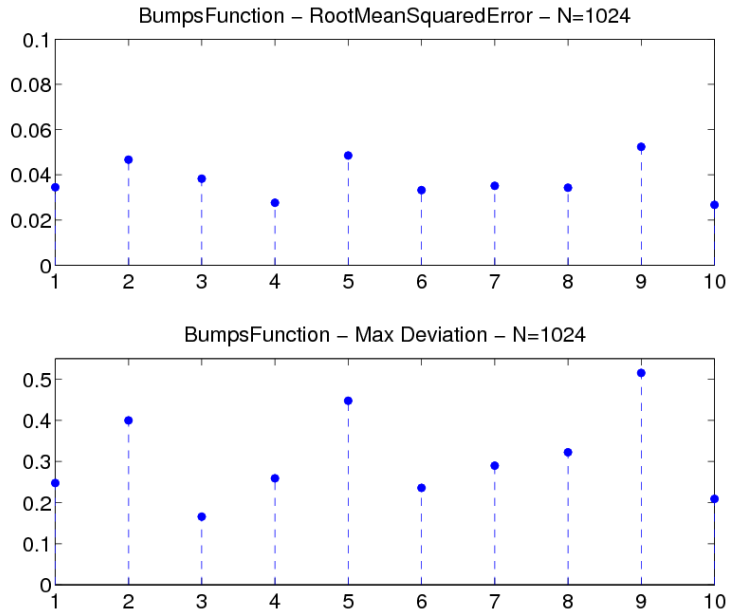


Figure 10.13: Comparison Study using Bumps function. N=1024

10.1.5 Conclusions

A general comment can be made related to the Root Mean Squared Error (RMSE). As expected, the bigger the sample size the lower the value of the RMSE. It is readily seen that this is true for the same test function and denoising procedure.

Focusing on the smooth Wave function, the bayesian method performs well. However, the linear penalization method and the Translation-Invariant-Hard method are very competitive. The performance of the penalization method should not be surprising since the linear estimators are expected to achieve good results in smooth functions such as the Wave signal. Similar remarks can be made about the Time-Shifted Sine signal, a function that shares with the Wave signal the smoothnes feature.

As far as the Bumps function and the Blocks function are concerned, the Bayesian method outperform the classical ones in terms of RMSE. This leads to the conclusion that using Bayesian methods for such type of functions is preferable if computational efficiency is not an issue. In fact, it is well established that non-Bayesian methods uniformly outperform Bayesian methods in terms of CPU time.

Finally, as a general remark, larger values of MaxDeviation occur for functions with many spikes and discontinuities.

10.1.6 Acknowledgments

The authors wish to thank Professor C. Sidney Burrus for his help and guidance through the development of this work.

Bibliography

[1] C.S. Burrus, R.A. Gopinath, and H. Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice-Hall, 1988.

[2] Tony Cai. Adaptive wavelet estimation: a block thresholding and oracle inequality approach. *The Annals of Statistics*, 27:8988211;924, 1999.

[3] D. Donoho. Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and Computational Harmonic Analysis*, 1:1008211;115, 1993.

[4] David L. Donoho and Iain M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90:12008211;1224, 1995.

[5] A. Grossmann and J. Morlet. Decomposition of hardy functions into square integrable wavelets of constant shape. *SIAM Journal on Mathematical Analysis*, 15(4):723–736, 1984.

[6] A. Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69:3318211;371, 1910.

[7] Charles M. Stein. Estimation of the mean of a multivariate normal distribution. *The Annals of Statistics*, 9(6):11358211;1151, 1981.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *ELEC 301 Projects Fall 2008*
Edited by: Rice University ELEC 301
URL: http://cnx.org/content/col10633/1.1/
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Introduction"
By: Michael Foree
URL: http://cnx.org/content/m18950/1.3/
Page: 1
Copyright: Michael Foree
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Bird Choice"
By: Christopher Eagleson
URL: http://cnx.org/content/m18933/1.6/
Pages: 1-2
Copyright: Christopher Eagleson
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Matched Filter Implementation"
By: Robert LiKamWa
URL: http://cnx.org/content/m18917/1.3/
Pages: 3-5
Copyright: Robert LiKamWa
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Method Refinement"
By: Seth O'Brien
URL: http://cnx.org/content/m18919/1.2/
Pages: 5-7
Copyright: Seth O'Brien
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Matlab GUI"
By: Robert LiKamWa
URL: http://cnx.org/content/m18918/1.2/
Pages: 7-8
Copyright: Robert LiKamWa
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Results"
By: Robert LiKamWa
URL: http://cnx.org/content/m18952/1.1/
Pages: 8-10
Copyright: Robert LiKamWa
License: http://creativecommons.org/licenses/by/2.0/

Module: "Birdcall Identification: Conclusion"
By: Michael Foree
URL: http://cnx.org/content/m18954/1.1/
Pages: 10-11
Copyright: Michael Foree
License: http://creativecommons.org/licenses/by/2.0/

Module: "Introduction: Building an Electrocardiogram (ECG) Based Diagnostic System"
By: Christine Moran, Yuheng Chen, Leslie Goldberg
URL: http://cnx.org/content/m18955/1.2/
Pages: 13-15
Copyright: Christine Moran, Yuheng Chen, Leslie Goldberg
License: http://creativecommons.org/licenses/by/2.0/

Module: "Collecting and Filtering Live ECG Signal"
By: Christine Moran, Leslie Goldberg, Yuheng Chen
URL: http://cnx.org/content/m18956/1.2/
Pages: 15-19
Copyright: Christine Moran, Leslie Goldberg, Yuheng Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Algorithms for ECG Signal Analysis"
By: Christine Moran, Yuheng Chen, Leslie Goldberg
URL: http://cnx.org/content/m18957/1.1/
Pages: 20-24
Copyright: Christine Moran, Yuheng Chen, Leslie Goldberg
License: http://creativecommons.org/licenses/by/2.0/

Module: "Introduction"
By: Ryan Pei
URL: http://cnx.org/content/m18960/1.1/
Page: 25
Copyright: Ryan Pei
License: http://creativecommons.org/licenses/by/2.0/

Module: "System Specifications"
By: Ryan Pei
URL: http://cnx.org/content/m18961/1.2/
Pages: 25-27
Copyright: Ryan Pei
License: http://creativecommons.org/licenses/by/2.0/

Module: "DMT Transmitter"
By: Shamoor Anis
URL: http://cnx.org/content/m18984/1.1/
Pages: 27-28
Copyright: Shamoor Anis
License: http://creativecommons.org/licenses/by/2.0/

Module: "Channel Characteristics"
By: Jaimeet Gulati
URL: http://cnx.org/content/m18971/1.1/
Pages: 28-30
Copyright: Jaimeet Gulati
License: http://creativecommons.org/licenses/by/2.0/

Module: "DMT Receiver"
By: Shamoor Anis
URL: http://cnx.org/content/m18983/1.1/
Pages: 30-31
Copyright: Shamoor Anis
License: http://creativecommons.org/licenses/by/2.0/

Module: "Key Results and Conclusion"
By: Jaimeet Gulati
URL: http://cnx.org/content/m18973/1.1/
Pages: 31-33
Copyright: Jaimeet Gulati
License: http://creativecommons.org/licenses/by/2.0/

Module: "The Project Team"
By: Jaimeet Gulati
URL: http://cnx.org/content/m18970/1.1/
Page: 33
Copyright: Jaimeet Gulati
License: http://creativecommons.org/licenses/by/2.0/

Module: "Acknowledgements and References"
By: Shamoor Anis
URL: http://cnx.org/content/m18982/1.1/
Page: 33
Copyright: Shamoor Anis
License: http://creativecommons.org/licenses/by/2.0/

Module: "Introduction to Estimating Target Location Using 2-D Projective Geometry and Background Subtraction"
By: Jeonggoo Song
URL: http://cnx.org/content/m19011/1.1/
Page: 35
Copyright: Jeonggoo Song
License: http://creativecommons.org/licenses/by/2.0/

Module: "Background Subtraction"
By: Jeonggoo Song
URL: http://cnx.org/content/m19013/1.1/
Pages: 35-37
Copyright: Jeonggoo Song
License: http://creativecommons.org/licenses/by/2.0/

Module: "2-D Projective Geometry"
By: Jeonggoo Song
URL: http://cnx.org/content/m19014/1.1/
Pages: 37-42
Copyright: Jeonggoo Song
License: http://creativecommons.org/licenses/by/2.0/

Module: "Combining Results - Background Subtraction + Projective Geometry"
By: Jeonggoo Song
URL: http://cnx.org/content/m19012/1.1/
Pages: 42-43
Copyright: Jeonggoo Song
License: http://creativecommons.org/licenses/by/2.0/

Module: "The Team"
By: Jash GUO
URL: http://cnx.org/content/m18949/1.3/
Pages: 45-49
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "Introduction"
By: Jash GUO
URL: http://cnx.org/content/m18943/1.1/
Page: 49
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "Background"
By: Jash GUO
URL: http://cnx.org/content/m18945/1.1/
Pages: 50-53
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "Methods and Results"
By: Jash GUO
URL: http://cnx.org/content/m18946/1.1/
Pages: 53-60
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "Conclusions and Future Work"
By: Jash GUO
URL: http://cnx.org/content/m18947/1.1/
Pages: 60-63
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "References and Acknowledgements"
By: Jash GUO
URL: http://cnx.org/content/m18948/1.1/
Page: 63
Copyright: Jash GUO
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Introduction and Background"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18959/1.2/
Pages: 65-66
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Digit Database"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18965/1.1/
Pages: 66-67
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Image Acquisition"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18963/1.1/
Pages: 67-68
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Your Digits!: Image Processing"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18969/1.1/
Pages: 68-70
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Identification Algorithm"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18966/1.1/
Pages: 70-72
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Results"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18968/1.1/
Pages: 72-73
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Give Me Your Digits!: Improvements and Conclusion"
By: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
URL: http://cnx.org/content/m18967/1.1/
Pages: 73-74
Copyright: Samuel Kim, Joel Khan, Yung-Seok Kevin Choi
License: http://creativecommons.org/licenses/by/2.0/

Module: "Abstract Summary of Information Hiding and Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18992/1.1/
Page: 75
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Encoding Information in Audio for Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18995/1.6/
Pages: 76-81
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Decoding Information Hidden in Audio for Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18994/1.2/
Pages: 81-82
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Testing Methods of Information Hiding and Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18999/1.2/
Pages: 82-85
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Surviving Attacks on Information Hiding and Audio Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m19003/1.1/
Pages: 85-92
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Uses of Information Hiding and Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m19001/1.1/
Page: 93
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Future Work in Information Hiding and Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18996/1.1/
Pages: 93-94
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Conclusions about Information Hiding and Watermarking"
By: Bailey Basile, Katherine Threlkeld, Daniel Valvano
URL: http://cnx.org/content/m18993/1.1/
Page: 94
Copyright: Bailey Basile, Katherine Threlkeld, Daniel Valvano
License: http://creativecommons.org/licenses/by/2.0/

Module: "Introduction"
By: Frank Chen
URL: http://cnx.org/content/m19005/1.2/
Page: 95
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Karplus-Strong Algorithm"
By: Frank Chen
URL: http://cnx.org/content/m19006/1.2/
Pages: 95-97
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "ADSR"
By: Frank Chen
URL: http://cnx.org/content/m19007/1.2/
Pages: 98-101
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Results"
By: Frank Chen
URL: http://cnx.org/content/m19008/1.2/
Pages: 101-102
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Random Music Generator"
By: Frank Chen
URL: http://cnx.org/content/m19009/1.2/
Pages: 103-104
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Extension"
By: Frank Chen
URL: http://cnx.org/content/m19010/1.2/
Page: 104
Copyright: Frank Chen
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- Introduction"
By: Jake Poteet, Jose Hernandez, Ricky Barrera, Jon Stanley
URL: http://cnx.org/content/m18962/1.1/
Page: 105
Copyright: Jake Poteet, Jose Hernandez, Ricky Barrera, Jon Stanley
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- FICA Past Work"
By: Jake Poteet, Jon Stanley, Ricky Barrera, Jose Hernandez
URL: http://cnx.org/content/m18974/1.1/
Page: 105
Copyright: Jake Poteet, Jon Stanley, Ricky Barrera, Jose Hernandez
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- FICA Assumptions"
By: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
URL: http://cnx.org/content/m18975/1.2/
Page: 106
Copyright: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- STFICA/Whitening Differences"
By: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
URL: http://cnx.org/content/m18976/1.1/
Page: 106
Copyright: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- STFICA Implementation"
By: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
URL: http://cnx.org/content/m18977/1.1/
Page: 107
Copyright: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- Results"
By: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
URL: http://cnx.org/content/m18978/1.1/
Pages: 107-112
Copyright: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening:Drown Out the Noise - Conclusion"
By: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
URL: http://cnx.org/content/m18979/1.1/
Pages: 112-113
Copyright: Jose Hernandez, Jon Stanley, Ricky Barrera, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- Applications"
By: Ricky Barrera, Jose Hernandez, Jon Stanley, Jake Poteet
URL: http://cnx.org/content/m18980/1.1/
Page: 113
Copyright: Ricky Barrera, Jose Hernandez, Jon Stanley, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Selective Listening: Drown Out the Noise- Acknowledgments"
By: Ricky Barrera, Jose Hernandez, Jon Stanley, Jake Poteet
URL: http://cnx.org/content/m18981/1.1/
Pages: 113-114
Copyright: Ricky Barrera, Jose Hernandez, Jon Stanley, Jake Poteet
License: http://creativecommons.org/licenses/by/2.0/

Module: "Signal Denoising using Wavelet-based Methods"
By: Isaac Hernandez-Fajardo, Georgios Evangelatos, Ioannis Kougioumtzoglou, Xin Ming
URL: http://cnx.org/content/m18931/1.2/
Pages: 115-131
Copyright: Isaac Hernandez-Fajardo, Georgios Evangelatos, Ioannis Kougioumtzoglou, Xin Ming
License: http://creativecommons.org/licenses/by/2.0/

About Connexions

Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.