2012

Interfacing To A Synthesized-To-Human Voice Converter By Way Of Web-Based Social Networking

Dwight Hudson
North Carolina Agricultural and Technical State University

INTERFACING TO A SYNTHESIZED-TO-HUMAN VOICE CONVERTER BY WAY OF WEB-BASED SOCIAL NETWORKING

by

Dwight Hudson

A thesis submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

Department: Electrical and Computer Engineering
Major: Electrical Engineering
Major Professor: Dr. Corey A. Graves

North Carolina A&T State University
Greensboro, North Carolina
2012

This is to certify that the Master's Thesis of

Dwight Hudson

has met the thesis requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2012

Approved by:

_____	_____
Dr. Corey A. Graves Major Professor	Dr. Alvernon Walker Committee Member
_____	_____
Dr. Christopher Doss Committee Member	Dr. John C. Kelly Department Chairperson

_____

Dr. Sanjiv Sarin
Associate Vice Chancellor for Research and Graduate Dean

BIOGRAPHICAL SKETCH

Dwight Hudson received his Bachelors of Science in Computer Engineering from North Carolina A&T State University in 2010. During his undergraduate career he developed great interest in micro processing and logic design. During his postgraduate studies he developed an interest for processing voice signals. This interest was paired with his passion for music. Many post-graduate projects included processing or development of audio signals. This played a very large role in the selection of a thesis project. Dwight has had much field experience working for Hewlett-Packard and Intel for multiple summer internships. Dwight is also a member of the following campus organizations Eta Kappa Nu National Electrical Engineering Honors Society, Golden Key International Honors Society, IEEE, NSBE, Kappa Kappa Psi National Honorary Band Fraternity Inc., University Concert Band, University Marching Band Staff. Dwight is from Charleston, SC, where he attended Garrett Academy of Technology and concentrated in Pre-Engineering.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES **PAGE**

LIST OF TABLES

TABLES **PAGE**

ABSTRACT

Hudson, Dwight A. INTERFACING TO A SYNTHESIZED-TO-HUMAN VOICE CONVERTER BY WAY OF WEB-BASED SOCIAL NETWORKING **(Major Professor: Dr. Corey A. Graves),** North Carolina Agricultural and Technical State University

With the exponential growth of social media in today's society there is a growth in the amount of data that is disseminated on these sites. With the amount of communication that takes place on Social Media sites such as Twitter and Facebook, there are not many ways to express ones emotions via sites aside from emoticons and Computer Language (i.e. LOL, SMH etc.). Studies have shown the there is a correlation between what happens on social media and ones emotions [3]. Studies also show that users tend to express their opinions more via Social Media rather than in person [3]. While there are TTS Applications that can be used to convert the text to speech, the speech that is created lacks the emotion that the user would have used if they were to have said it themselves.

The PSEE VOICE (Pervasive System for Educational Enhancement) is designed as add-on system to the UMSEE (Ubiquitous Mobile System for Educational Enhancement) project designed by Dr. Corey Graves and Brandon Judd. By using Twitter, Cinch (Audio Twitter equivalent), and Espeak a TTS Application the PSEE VOICE system can register a user's voice in the database and convert text sent in via Twitter. The purpose of the system is to allow users to send text message that will sound more human like. This thesis addresses the area of pitch and word transition and how it affects the clarity of a speech signal. THE PSEE VOICE uses the Weighted Frequency

Warping and a Gaussian Mixture Model Based Transformation function. The system will be based on training data that is used for registration, the transformation that is used for voice conversion, and the order of the Gaussian Mixture Models that are used for the training. The system proves that having more training sets, using 16^{th} order GMM for training and using the GMM Based Transformation yields the clearest results.

CHAPTER 1

INTRODUCTION

1.1 Motivation

With the continuing expansion of social networks, such as Facebook, Twitter, YouTube, and Foursquare, in modern society, the type of information that is being exchanged is no longer just verbal (auditory) messages. Users are exchanging large amounts of Pictures, Video and Music on a day-to-day basis. According to Doug Beaver [1], in 2008 there were over 10 billion photos uploaded to the Facebook server and expected that 2-3 Terabytes will be uploaded daily. YouTube has over 800,000,000 million viewers every month. Twitter, being the newest of the social networks, went from about 3 million users at the beginning of 2009 to over 40 million users at the end of 2009 [2]. Twitter is a social media site geared towards utilizing messages only as a means for people to express opinions. Taking advantage of this large population one could subliminally educate or miseducate the masses. People between the ages of 14-25 are the primary users of Facebook. This age group also represents the majority of all high school and college students.

The primary method of communication via Twitter is written text. With written text, emotions and other personable traits are absent from the conversation. A study was done in [3] to view how often people actually express emotions on Twitter. The results showed that emotions are often attempted to be expressed via “smiley faces” or other key emotional words, which often have the ability for users to misinterpret ones emotions. Through spoken communication, aside from instances of sarcasm, ones emotions can

usually be identified. While there are many systems that can convert text to synthesize speech, a system capable of accurately converting messages from text form to a target human speech is currently unavailable. With the rise of social media sites such as twitter and further development of "text emotions", all emotions will be able to be expressed via emoticons and other textual devices. A system is needed that will allow users to hear messages, that sound like the sender of the message from Twitter and other social media sites in the form of spoken word.

1.2 Summary of Background

With the evolution of Digital Communication one can easily see how the use of print messages has become a thing of the past. While speech is still commonly used many would rather tweet or use other text-base social media sites to communicate instead of making a voice call via telephone. With this in mind Text-to-Speech and Speech-to-Text Tools have become common throughout modern technology. An example of technology of this type is Apple's Siri Application [4]. This application allows users to input real time speech commands and receive feedback from a synthesized voice known as Siri. With the creation of synthesized speech applications such as Siri and E-speak (Freeware application to convert text to speech) the next step is to create more realistic and human-like audio output [5]. Creating such a tool would require the capturing of intonation, pitch, dialect, tone quality, etc. in order to get the most realistic voice possible.

In the past years some researchers have developed tools that have addressed some of the issues to create voice conversion systems that utilize computational/ visualization Engines, such as MATLAB, to convert voice [6] [7] [8]. Developing systems able to

address these issues may require a variety of techniques such as Weight Frequency Warping, Formant Tracking, Gaussian Mixture Model Based, or Linear Predictive Coding.

1.3 Proposed Solution

Utilizing the UMSEE system that was conceived by Dr. Corey Graves, this system was then modified to become Twitter Based by Brandon Judd M.S. [9] This system utilizes twitter as its primary source for data and captures and analyzes those signals using Octave/MATLAB. Also this system was based on the work of Daniel Valencia M.S. who developed a TTS system that used text messaging as a means for English to Spanish translation [29]. Using this system as a base, the PSSEE VOICE system was developed as an add-on to this system that will capture messages from Twitter. With twitter being one of the top Social Media Sites in society today there is an abundance of data and emotions that can be acquired, having access to twitter is limitless [10]. With this much data one has the ability to capture lots of text that will include emotion. A system that would be able to accurately convert text sent by a user to the voice of the user would be useful.

Finding a MATLAB –based voice conversion system, would be ideal. In nearly all of the research examined on Voice Conversion the mention is made of a "source" and a "target" speaker. The idea is that the source speaker, after conversion, will sound like the target speaker, but saying what the source speaker said [11]. This will require a method for capturing voice files to be used for the source and target speakers. This

project will address Voice Conversion with the source being a synthesized voice that is produced from some Text-to-Speech Application such as Siri, Espeak, or Google Voice.

This project will approach ways to convert the vocal characteristics of a synthesized voice, produced from a TTS system, to a target voice by using social networks to acquire the source and target speaker for training and conversion.

Figure 1.1 Apples Siri Logo

CHAPTER 2

BACKGROUND

2.1 Art of Speaking

Speech is the practice of communicating messages by utilizing a certain set of articulated sounds [12]. These sounds can be manipulated in many ways such as loudness, pitch, intonation, or emphasis. Loudness can be defined as the amount of volume applied to a given phonetic syllable or word. Pitch is the frequency at which a sound is made. Within speech, pitch can also be associated with overtones, which are sounds that are the same just at different octaves. The spontaneity of overtones also aid in making the human voice sound more realistic, in comparison to synthetic speech which has a programmatic way of applying overtones to the voice. Also, intonation plays a major role in how well speech can be interpreted. Intonation is a learned or natural inflection or deflection of the voice to certain words, phrases, or phonemes. Intonation can be broken into two sub categories, which are prosody and articulation. Prosody deals with how long certain sounds are held out for a given word. Articulation deals heavily with the inflection and deflection of the voice. Also known as dialect, articulation is a major factor in the characteristics of speech. Just as in the other characteristics, intonation can be very random an unpredictable. The last characteristic addressed here is emphasis. Emphasis is a combination of intonation and loudness. This characteristic, in the formation of speech, is very similar to loudness and articulation but in terms of speech interpretation it can play a major part. The characteristics discussed: Loudness, Intonation, and Emphasis all deal with what is called the neuromuscular commands [12].

These are controlled by the neuromuscular commands given from the brain to produce the given sound.

The production of speech by humans can be compared to that of a machine. Figure 2.1 from [12] gives a great explanation of how speech generation and speech recognition work. Just as with a computer, human speech starts with a message. The message is then formulated to a language code. Examples of the "language code" include English, Spanish, Chinese, etc. After it is converted to the users language then the neuromuscular system will then communicate with the code that was created and begin to produce the sound waves suitable for recognition from another listener. After the speech has been produced then it is the job of the listener as well as the talker to extract the phonemes, and speech characteristics (Loudness, pitch, intonation, and emphasis) to determine what message was trying to be relayed. The listener will use the Basilar membrane, which is inside of the ear to extract the features. While this process is happening, just as with a computer, features are extracted and saved to a "database" then reassembled for later use. So in the context of speech, the phonemes that can be understood are then processed into a language code and then interpreted as the message.

Speech has other characteristics such as spectral energy, which is classified based on the phonetic sound that is being produced and the amount of intensity associated with it. This spectral energy will be used later for the classifying of spectral changes within Gaussian Mixture Model the authors in [13] gives a great explanation of what a GMM is and how they affect synthesized speech.

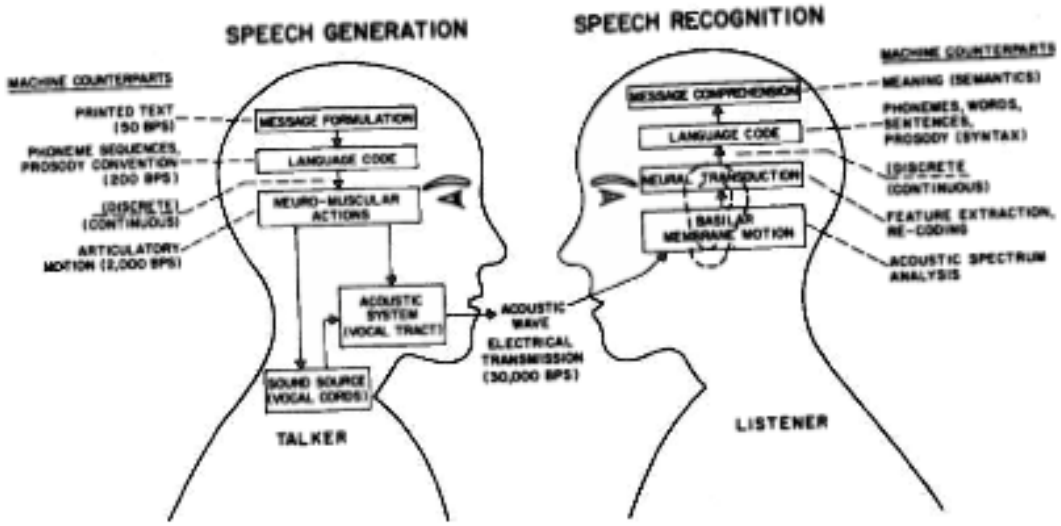

Figure 2.1 Art of Speaking

2.2 Text-To-Speech Technology

Text-to-Speech is the procedure of converting a written message to a verbal message by the use of some synthetic voice creation. Within the past 20 years there have been 3 major methods for synthesizing voice: Residue Excited (RELP), Voice Excited (VELP), and Linear Predictive Coding (LPC). The most commonly used method is LPC due to the advantages discussed in [14]. These methods are used within many applications such as E-Speak, Google Translate, AT&T Natural Voices ®, and Natural Reader. These tools have various methods of creating the databases for the source of the speech sounds. Some TTS applications use a diphone-based method in which there is one sound associated with a certain diphone. Diphones are a pair of phones, the basic unit of phonetics, which deal with a sound a certain sound. With that method, there is no care about the prosody of the sound that is being used. While this method does produce a synthesized voice, the sound produced is inhuman like and sometimes unrecognizable. Another method that is used for voice synthesizing is called unit selection, in which there

are multiple instances of one diphone and the system has the capability to choose an instance of that unit depending on the placement of that sound.

There are also applications such as Apple's Siri in which the synthesize speech has the capability to include articulation specific to the users voice [4]. One of the issues of speech synthesis is that to get the most realistic sound the database that is being used must include a large data set and specification to what dialect is being used. This is important so that the synthesized speech can sound more realistic given a different region and how articulation plays a part.

2.3 Voice Conversion Methods

Voice Conversion can be defined as the method of converting one persons voice to another person's voice. Common terms used throughout voice conversion are source speaker and target speaker. Source speaker being the voice that is going to be converted and target speaker being the voice that the source is being converted to, these two speakers are the foundation of voice conversion.

Many of the voice conversion systems have similar building bocks. Gaussian Mixture Models and Hidden Markov Models are two concepts that are used in the descriptions of voice conversion systems [15]. Hidden Markov Models (HMM) are a statistical method of collecting data from observation vectors that uses probability as the measure to model sequential data. HMMs are created after collecting training data from a target or source speaker. HMMs observe the temporal progression of a sound and separate each accord sound into different states that represent those progressions. Gaussian Mixture Models, unlike the HMM, treats the entire sound as a single class and

classify the data based on the spectral characteristics. While both models are commonly used in speech analysis, HMMs are used more commonly because they can be comprised of GMMs. Each GMM can be viewed as a state within the HMM and classified into different observation vectors and later used for conversion. While this is the basis of voice conversion and how the states are determined, the data captured is what gives us different voice conversion methods.

Voice Conversion based on Formant mapping is one method of voice conversion [16] [17]. This method involves a process called frame alignment in which a training set created by the source speaker and a training set created by the target speaker are aligned. Each training set must contain the same phrases or sentence spoken by the respective speaker. Once the frames are aligned then the formants, the spectral peaks of the sound spectrum of voice, are extracted from each training set and then warped using a method called frequency warping. This process is used to produce a frequency warping function that will be used as a means to convert the source speech to the target speech.

Another voice conversion method is fundamental frequency adjustment. This method takes the fundamental frequency of the source target, captured by a training set and the fundamental frequency of the target speaker and uses the average and variance of the two fundamental frequencies as a means to convert the source to the target. [17]

Spectral Envelope Equalization is voice conversion techniques in which the curve between the average power spectra of the source and target are calculated. This curve is then smoothed and used a filter for voice transformation. [17]

Non-Aligned Weighted frequency warping is another method of voice conversion. This method is very similar to the formant mapping based conversion system. Unlike the formant mapping the training sets that are used during alignment do not have to be the same. During the alignment phase instead of the frames having to be of the exact same length and speech timing, a Maximum likelihood function is created from the Hidden Markov Model that is created. This function will then be used similarly to the "perfectly aligned" training sets for the frequency warping techniques. [18]

2.4 Scientific Computational/Visualization Engines and Voice Conversion

MATLAB is a common computational Engine that is used by engineers for audio, visual, and numerical data. There are many toolboxes that are used to manipulate data. Voice conversion requires the extensive use of calculations, which if done by hand would be nearly impossible. Also the audio tools needed to convert the speech are easily accessible through MATLAB, thus making it a useful tool for voice conversion. There are many voice conversion toolboxes that have been created in MATLAB using many of the different voice conversion methods. These include the Siemens Voice Conversion Toolbox, VOICE BOX and HSM Voice Conversion Toolbox [6][7][8]. These toolkits take the functionality of MATLAB and allows the user to test from simple and very complex voice conversion.

2.4.1 HSM Voice Conversion Toolbox

The HSM Voice Conversion Toolkit uses the Harmonic Stochastic Model for voice conversion. Dr. Daniel Erro of University Politècnica de Catalunya developed this system [19]. This model differs from its predecessors because it pays more attention to

the prosodic details of the signal as well it focus on better way to concatenate the segments that are formed during the analysis and synthesis process. To explain HSM, first consider voice conversion in general, Figure 2.2 shows a diagram of a typical voice conversion system.

For this system, the first step is the analysis and synthesis of the original target and source speaker to assure that they can be used during the alignment phase. Once the source and target have been synthesized they are then ready to create the corpus set, which include the speech parameters: line spectral frequencies, spectral samples, formants, and cepstral coefficient. Those parameters will be used for training. This project will be using the non-parallel corpus and training, which will make use of the nearest neighbor algorithm for alignment. The nearest neighbor algorithm classifies data based on results, in this project, how close the frames are to each other. The content of those frames are not the primary focus but rather the frames themselves. X represents the target speaker and Y represents the source speaker. Two auxiliary vectors are then formed (x' and y') which will be used as the vectors that will be used to test the convergence. The Nearest neighbor algorithm is repeated until the x' and y' converge and the GMM Function or Weighted Frequency Warping Function (WFW) stores the value as F, which will be used as the voice conversion function that will be used to convert the source to the target. The authors in [16] give details about the algorithm.

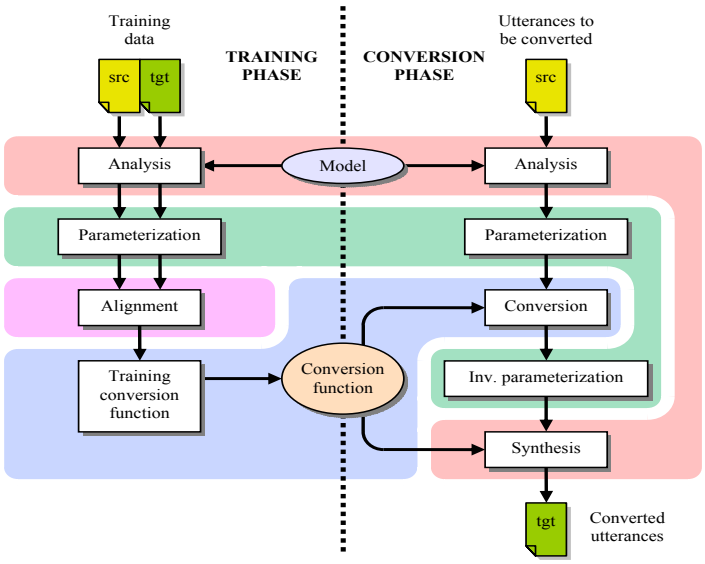

Figure 2.2 Standard Voice Conversion System

After the voice conversion function, F, has been created then it will be stored until the source voice is to be converted again. Once converted the data has to be resynthesized so that it can become a waveform once more. Then a fully converted sound will be available.

2.4.2 Siemens Toolbox

The Siemens Toolbox is a MATLAB based voice conversion system that utilizes the Vocal Length Tract Normalization frequency and time domain transformation or linear transformation [6]. This system is based on pitch tracking which is explained in [20]. This tool allows users to have a fully functional system that allows them to determine the training algorithm that would be used for the wave files that are used.

Unfortunately this system has been discontinued and has been removed by the sponsoring company.

2.4.3 Voicebox

This system as described in [7] has numerous voice conversion methods that could be use. Mike Brooks developed this system for the public to practice voice conversion. Mike Brooks is a professor at Imperial College. While this system has many conversion methods such as, Frequency Warping and LPC filtering, the ease of use of this tool was not as simple. This system was not very user friendly

2.5 Web-based Social Networking: Twitter and Cinch

Web-based Social Networking is the practice in which users of different networks communicate based on a variety of social demographics. This functionality has resulted in the eruption of the Social Media Revolution. Companies such as YouTube, Facebook, Twitter, MySpace, and Google+ have erupted as some of the most prominent members of this revolution in the past few years. While these are the more prominent members of the social media team there are a number of social networks that are growing in popularity.

Twitter with 250,000,000 unique visitors each month is the 2nd most visited social media site in the world [10]. Twitter is a social network unlike its predecessors, Facebook and MySpace, as it has a very different layout in which users are only allowed to send in statuses that must be under 140 characters. With the aid of 3rd party applications such as TwitPic and TwitVid, Twitter can now engage the user in other manners by allowing users to upload videos or pictures. Along with those applications, many websites now have a share button as seen in Figure 2.3, by which users can directly

share links or information. Also other social networks have twitter plugins that allow users to sign into their own sites with their twitter credentials, which can be very helpful as users go from one site to another. Twitter also uses the short-url technology, which allows users to shorten the length of a URL (Uniform Resource Locator)[22]. Twitter uses the "t.co" short code for all URLs that are placed in a message. This is very helpful since there are a limited amount of characters one can use for tweets. This gives the ability to share information across different social networks.

For development purposes, a user may need to expand the short URL that is provided from Twitter. There are several "URL Expansion Sites" that allow users to input the short code and it will provide the original link examples are, expandurl.com, longurl.com, and Google's URL Expander, to name a few. This technology is helpful because some URLS have very pertinent information such as a username or link ID.

Cinch is a social network that allows users to record audio messages onto a "timeline", similar to Twitter, that a user may add to at any time [20]. Cinch allows one to post messages from one's computer, home telephone, or cellular phone. Users have the option to share the message with other social networks such as Twitter or Facebook. Cinch also allows users to sign in with their Twitter or Facebook credentials see Figure 2.3 and 2.4.

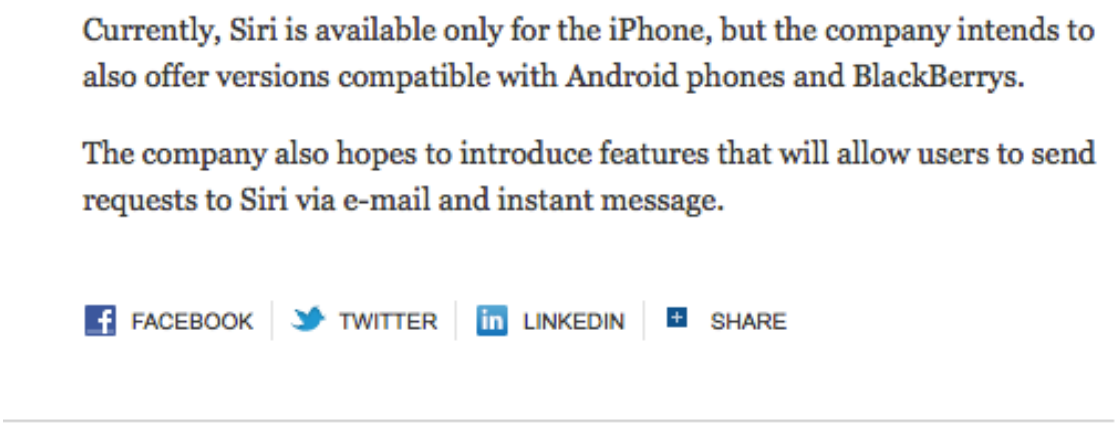

Figure 2.3 Blog with Twitter Share Button

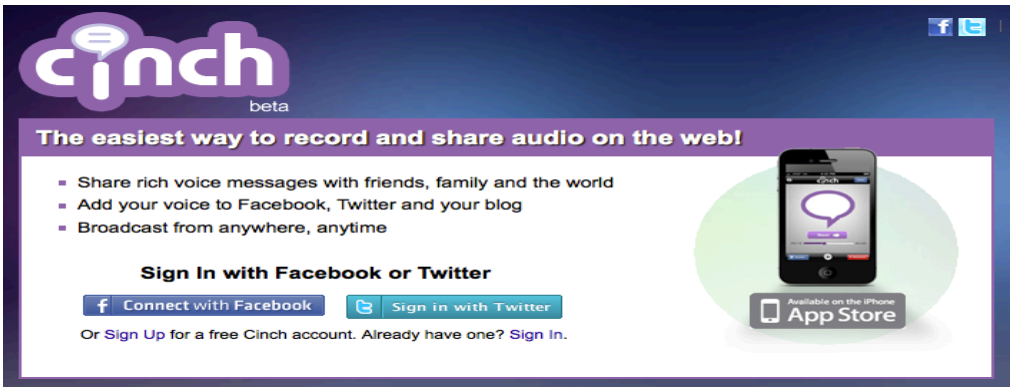

Figure 2.4 Website that uses Twitter credentials to login

CHAPTER 3

PSEE VOICE DEVELOPMENT

3.1 Introduction

The PSEE VOICE system is a C# based application that will be used for the conversion of voice from synthesize speech, produced from text acquired from Twitter, to some target voice captured from a user using Cinch. PSEE VOICE is divided into two major sub functions, Registration and Conversion. Registration encompasses the incorporation of Cinch, Twitter, and MATLAB for the saving and manipulating of the target and source speaker. Conversion involves Twitter, MATLAB, and the "To User's" Cell phone to send a converted file to that user. Figure 3.1 gives a greater explanation of how the system works as a whole. Later on in this chapter the details will be explained.

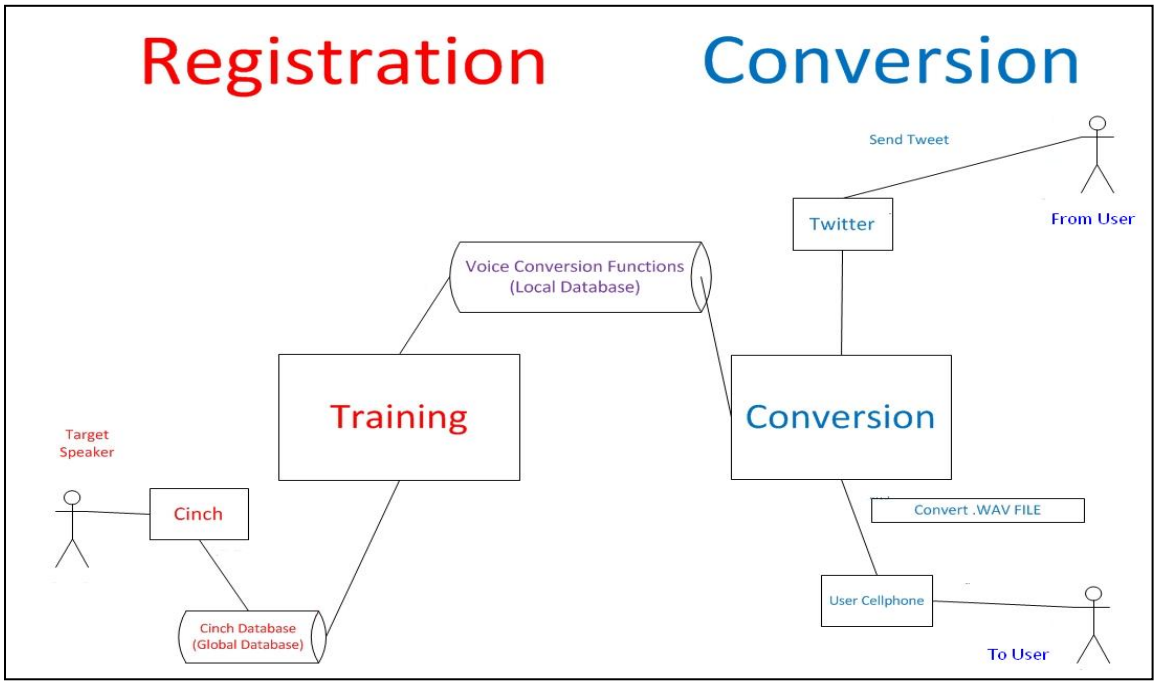

Figure 3.1. PSEE VOICE System Flow Chart

3.2 Voice Conversion Methodology

Dr. Erro of the Polytechnic University of Catalonia has developed the methodologies that will be used for this project. The explanation of the full system can be found in Section 2.4. The following processes explain in greater detail what happens at each step.

Analysis/Synthesis:

The Target/Source (T/S) original training data is passed into the analyze function. The data is then parsed into a series of vectors, which capture the characteristics of the original data. This data is then synthesized and then it is reconstructed into a nearly exact replica of the original data.

Non-Parallel Corpus (Parameterization):

There are three main reasons discussed in [19] for the purpose of parameterization: (1) speaker identification, (2) difficulty in converting high dimensional vectors, and (3) voice conversion parameters have good interpolation properties. The speech parameters that are used in this system are line spectral frequencies, spectral samples, formants, and cepstral coefficient. These properties are then used for training to create a voice conversion function.

Non-Parallel Training:

As described in section 3.1 the nearest neighbor algorithm will be used during the training to train the T/S speakers. The amount of GMMs (Gaussian Mixture Models) can also be specified as an input to the training function. In section 5 we will see how the number of GMMs affect the voice conversion system. This process will yield a Voice

conversion function, which will be used to convert the T/S speakers. This function will be specific to the T/S pair that is used to create it.

GMM/WFW Convert:

The conversion process uses the conversion function that was created in the training process to convert the source speaker to the target speaker. Once that occurs then there has to be an inverse parameterization that takes place to apply the proper vocal characteristics back to the converted voice. After this process has taken place then the signal is then reconstructed to become a wave that can be heard as the final output.

3.3 HSM VC Toolbox and PSEE VOICE

The HSM (Hidden Stochastic Model) Toolbox came with several functions that encompass the methodology described in section 3.2.

3.3.1 Reconstruction

During the Analysis and Synthesis phase there were two functions, HSMAnalyze.m and HSMSynthesize.m. The input parameters to this phase are the original .wav file of the T/S speakers and the output is a .mat file, which contains the parameters of that input file. That file is then used as an input to the HSMSynthesize.m function, which is used to reconstruct a .wav file that will be used for parameterization/training. The project required that the ampreduce.m be applied, which reduce the amplitude of the .wav files that would be used. Figure 3.2 gives an explanation of what takes place during this phase of the process.

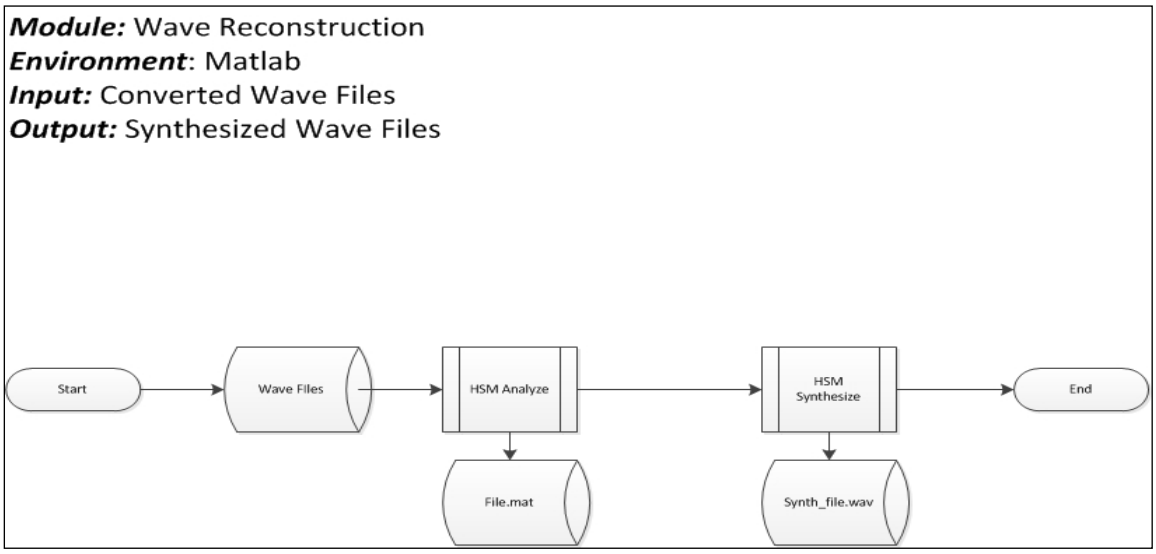

Figure 3.2 Wave Reconstruction

3.3.2 Parameterization

After the Analysis and Synthesis phase the parameterization occurs in the HSMnonpcorpus.m file. For the sake of this project we will be using the nonpcorpus and nonptraining because we are using non-parallel data sets. The inputs for this function are the reconstructed signals of the synthesized signals. The number of data sets can vary. In the experiment section, how the length and number of samples can affect the results will be discussed later. During the parameterization a .mat file containing both the T/S speakers' speech parameters is created. The .mat file that is created is then passed into the HSMnonptraining.m function. This function will create a voice conversion as well an inverse voice conversion function, which will be used during the inverse parameterization of the converted source. After these functions have been created they are stored in a folder and wait to be used for conversion.

3.3.3 Voice Conversion

Once the source speaker has input another file, either the function HSMwfwconvert.m or the function HSMgmmconvert.m is called, depending on the type of warping that is going to be used. Just as the training data, the testing data has to pass through the ampreduce.m function for further analysis. The input file is analyzed, using HSMAnalyze.m and the vocal characteristics of that file are then passed into the respective convert function. After the conversion has taken place then the .mat file is sent to HSMSynthesize.m for reconstruction and the final output is created. Figure 3.3 gives a better description of what is happening during this phase.

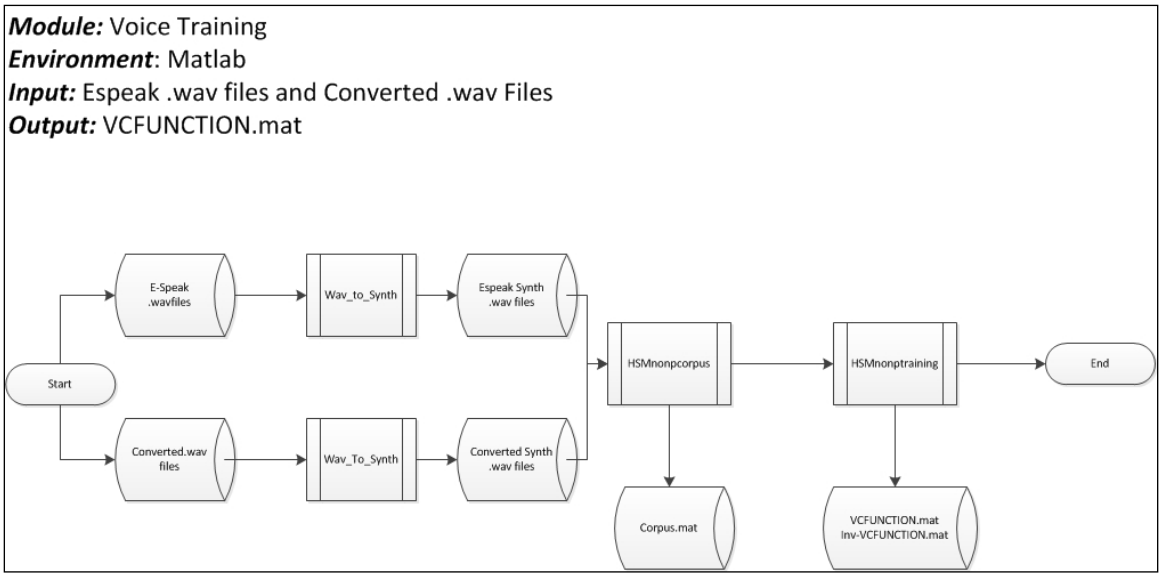

Figure 3.3 Voice Training

In the context of this thesis these are the only functions that were used for the voice conversion process.

3.4 PSEE VOICE System

One great asset to utilizing social networking tools such as twitter is that the sites allow developers to use their functionality for further development. In the scope of the research I will be utilizing the UMSEE model for the interface between Twitter and the C# Application [9]. Along with the UMSEE system this system will make use of the MATLAB Voice Conversion Toolbox described earlier utilizing Espeak and Cinch as a means for source and target speakers. This section will discuss how the full system works including the registration and conversion processes.

3.4.1 Registration

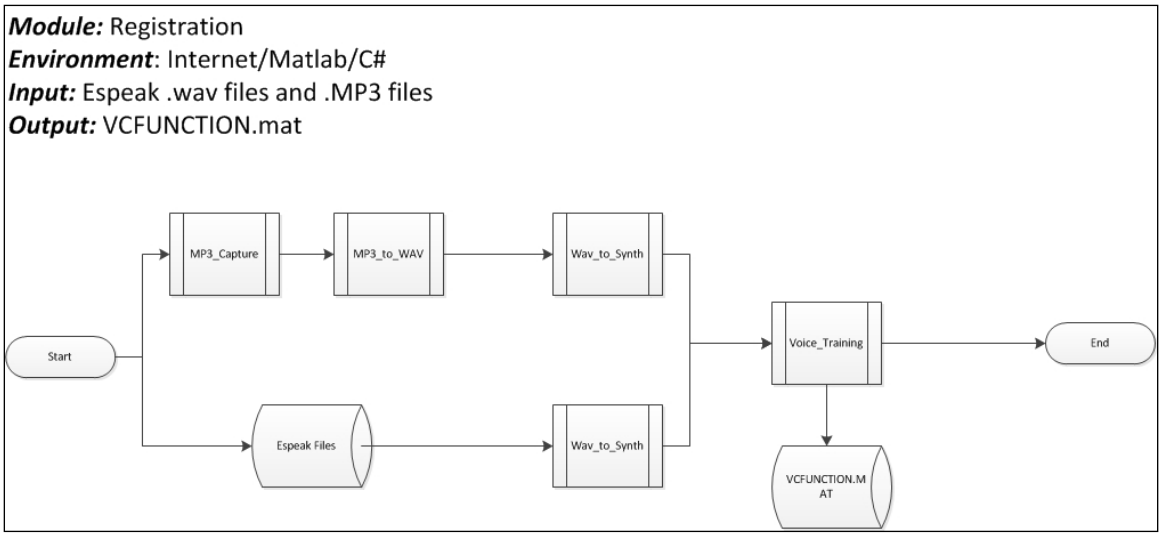

Figure 3.4 Registration

Cinch Record:

Cinch will be used as a mechanism to capture the voice files that will be added to the database of target voices as shown in Figure 3.1. Figure 3.5 explains how the voice collection will work. In utilizing Cinch's recording capability there will be 5 phrases that

will be used as the target voice. Users can record the phrases in via computer, telephone, or cell phone. Once the cinch is recorded there is a short URL that is created which contains the MP3 of the cinch that was recorded. After recording, the cinch must be posted to that users Twitter account. The format for the Tweet is as follows, "@PSEEVOICE ??Phrase1 <icin.ch/#####>". The "cinch.fm" link is only viewed in that form on the twitter timeline.

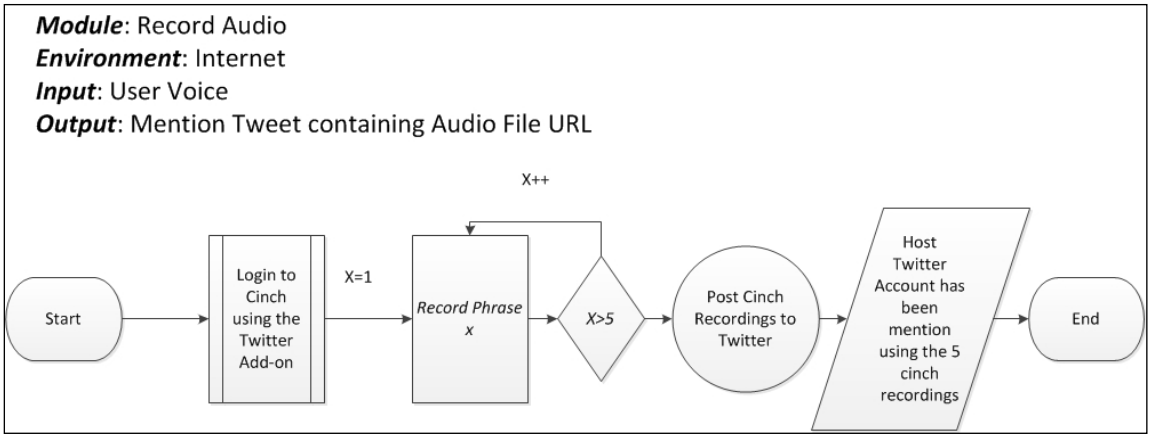

Figure 3.5. Audio Recording from Cinch

MP3 Capture:

After recording the phrases into Cinch and posting them as a mention to the host Twitter account (@PSEEVOICE) the C# Application will use the UMSEE Mention retrieval to capture the 5 most recent mentions that contain "??Phrase" and "t.co". *NOTE: The icin.ch/##### is actually a second generation Short URL so it requires the use of a URL expander, expandurl.com, to get all generations of the link. With each icin.ch short URL there are 3 URLs associated with it: "t.co", "incin.ch", and "cinch.fm". When the C# Application captures the mention it will read the URL in its "t.co" format.*

After deciding if the mention has “t.co” in it that tweet is then split into the various words and the URL is extracted. Through string manipulation the URL containing the MP3 that was recorded is then saved in a directory for later use. Figure 3.6 shows how the entire MP3 Capture works. The data is stored in a folder that has the username (target) of the corresponding twitter/cinch user, as the title.

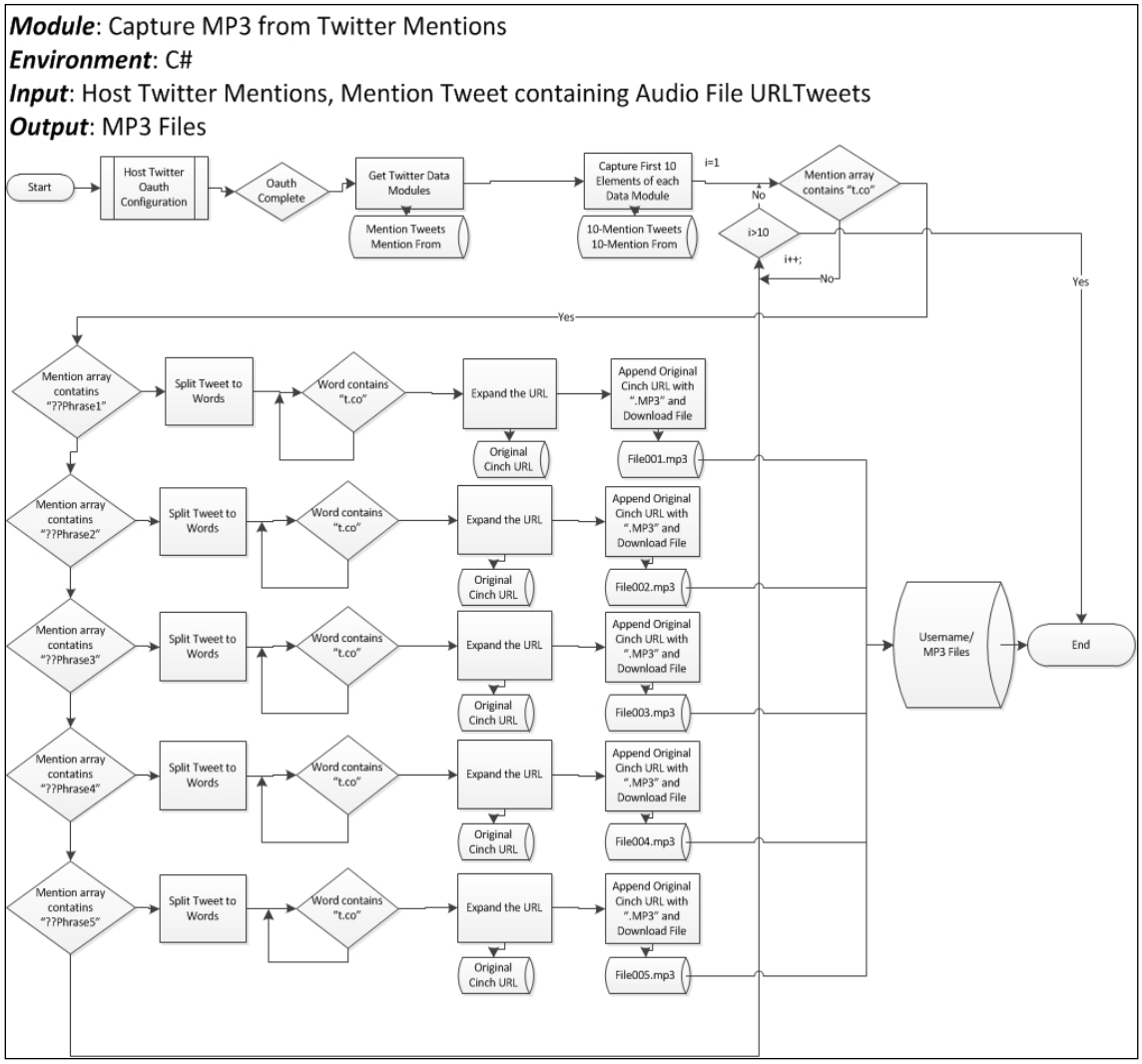

Figure 3.6. MP3 Capture

MP3 Conversion:

This process is to convert the MP3s that have been captured from the Cinch are then converted to .wav files so that they can be used in the MATLAB toolbox. During this process the MP3read.m function, developed by Alfredo Fernandez [28], which takes in an .MP3 file and converts it to a .wav file. After the file has been converted the length of the file is then checked to determine if it is the proper length for testing. If it is not the length that is specified then the zeroappend.m function will add zeros until the signal is the desired length. The converted .wav files are sampled at a rate of 22khz and for the sake of the MATLAB toolbox the signal has to be 8khz. So to create the desired sampling frequency we have to interpolate the .wav file by 12 and then decimate the signal by 33 to get the data's sampling rate to 8khz. After the signal has been resampled then it is pass through a Low-Pass Filter to remove higher frequencies that may have occurred during recording. Then the final signal is rewritten to a .wav file. The system is explained in Figure 3.7.

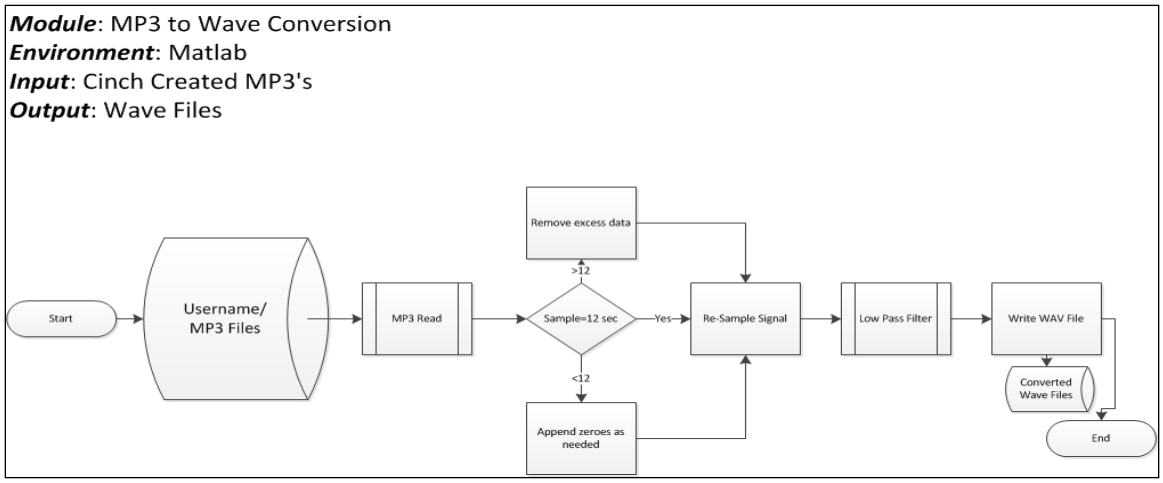

Figure 3.7. MP3 to Wave conversion

Espeak Wave Create:

The TTS system that will be utilized for this thesis is Espeak. "ESpeak is a compact open source software speech synthesizer for English and other languages, for Linux and Windows. http://espeak.sourceforge.net Espeak uses a 'formant synthesis' method. This allows many languages to be provided in a small size. The speech is clear, and can be used at high speeds, but is not as natural or smooth as larger synthesizers which are based on human speech recordings." [5] This synthesizer was chosen because it was the cheapest TTS speech application that was compatible with the system that was created. While the quality of the speech is not as high as some of the more expensive competitors, it does create a synthesized speech that can be used for testing.

Using the Espeak TTS system we use the same phrases that were used by the target voices in Cinch. Espeak creates a wave file that will be put through the reconstruction process that was described in section 3.3. The file is then saved in a database which will be used as the source during the voice conversion. This file will be saved as "TEMP_ORG.wav" in the "from user's" directory.

Cinch Reconstruction and Conversion:

After the Cinch MP3 has been converted then the signal is passed through the parameterization and reconstruction process described in section 3.3. This signal will produce the reconstructed signal that will be used for conversion.

Voice Conversion:

The reconstructed Espeak and cinch voices are then passed into the processes that were described in section 3.3.2 and 3.3.3. The .mat files that are created are then stored for use during the testing phase. There is a registration tutorial shown in Appendix A.

3.4.2 Conversion

Message Retrieval:

At this phase the system will again go to the host twitter account to acquire the textual message from a tweet. The system will parse each tweet looking for "o_o" delimiter to determine if the message is in the proper form. The proper form is as follow "@PSEEVOICE o_o <to user> <message>". After searching for the delimiter it will also save the "To Username", "From Username", and the message. The usernames are then checked to determine if they are in the database, this process will be explained in greater detail in the next sub section. If the *check user* process is validated then the message would be passed into the Espeak wave create function in MATLAB.

Check User:

The "To Username" and "From Username" are used as inputs to see if they are inside of the Userfile.txt, which contain the username, carrier, and telephone number. From that data an EMAILfile.txt is created which creates the email based on telephone number and carrier (i.e. 55555555@pm.sprint.com). This information will be used in the *send message* function. If the username is not in the text file, then the function will return false (invalid) and there is an on screen form that will capture that data and store it in the text file.

Voice Conversion:

After the users have been confirmed then the system uses the "from username" as means to search all directories to get the voice conversion function of that particular user. The files created, from the message, in the Espeak *wave create* function are viewed as the source and the voice conversion functions associated with the "from user" are seen as the target. The two file sets will be used as the input to the MATLAB Voice conversion function. After the file has been converted then the newly signal passes through another low pass filter to remove any other random noises that were created. Once the data has gone through all phases it is stored and saved for use in the message that will be sent.

Send Message:

Once the wave file has been created then the "To Username" is used as an index to search for the email address associated with that username. Once the name has been found the function returns the email address, which is passed in to the send message function. This function sends the message along with the attachment including the wave file.

CHAPTER 4

EXPERIMENTAL SETUP AND RESULTS

4.1 Explanation of Setup

The PSEE VOICE system has several parameters that can be tested. The length of the sample, the order of the GMMs [19], or using a Gaussian Mixture Model based transformation versus using Weighted Frequency Warping transformation are the areas that were tested in this experiment. These are the three areas that will be tested for this study. These tests will help identify the best settings with which to get clear and intelligible conversions.

Since this work only requires conversion from synthetic speech to human voice each test will only require one voice sample. Each sample will undergo subjective testing from a random set of listeners to grade the quality of the voiced based on a 5=(Sounds Human like) to 1=(doesn't sound human like at all).

The data will also be tested objectively to find quantifiable measures of success in voice conversion. When a sound is spoken by the source, after the data has been trained, that does not match what is in the database then a "squeak" will be heard. By capturing the number of squeaks that a sample has this will test the clarity of a particular testing environment. Along with the number of squeaks, the percentage of correctly transformed data will be captured. The length of the squeaks will be parsed from the output and used to determine the percentage of the signal that is correctly trained.

4.1.1 Order of GMM Sample Setup

This experiment will test to see if the order of the GMM has an affect on the quality of the signal. Using the GMM Transformation method the signal will be tested for 4^{th}, 16^{th}, and 32^{nd} order GMM for training. The signals will be divided into five-12 second phrases.

4.1.2 Number of Training Files

This experiment will test to see if the number of training sets will have an affect on the converted voice. Each wave will use the same data set but that data set will be split into different length phrases. (I.e. Five-12 second phrases, Four-15 second phrases, or Two-30s phrases). All other parameters for this test will remain the same.

4.1.3 GMM vs. Weighted Frequency Warping

This experiment will test the type of transformation that is being used for conversion. As discussed in section 2.4.1 there are multiple methods that can be used for converting the trained data. The two options that are available in the HSM Toolbox are Weighted Frequency Warping and GMM based conversion. The length will be 12 seconds and the phrase will remain the same. All recordings will be done through the computer. All other parameters will remain constant except for those specified here.

4.2 Results

In this section we will take a look at the results based on the three experimental setups that were explained in the previous section. The two test that were done on each data included a subjective hearing test by 20 random listeners and an observation of the amount of "squeaks" that are present in a given .wav file plot.

4.2.1 Subjective Hearing Test

All users listened to the phrase that was sent in via E-Speak. The message said, "This is a test for listeners to decide which settings sounds the most like Dwight, now the test is over".

The results of the test show that users found the 16th order GMM was the most human sounding signal. While users were asked to ignore the squeaks that were associated with the signal, the 4th and 32nd order signals did have squeaks which could have taken away from the quality of the speech that they heard. Table 4.1 shows the result of this test. While the 16th order was the most successful in comparison to the other two orders, the average score does reflect that users felt the converted signals sounded slightly more human like than synthesized.

Table 4.1 Order of GMM Mean Opinion Score

GMM Order	Average Score
4th	2.75
16th	3.2
32nd	2.45

The results as it relates to the number of training sets did not vary much in results. Users felt that the signals that were produced here did not sound very human like. While the five-12 second and four-15 second samples yielded the same average score, which was the highest, they were far away from sounding human like. Table 4.2 shows the

results of this test. With an average score of 1.95 users felt that these signals resembled more synthesized speech.

Table 4.2 Number of Samples Mean Opinion Score

Sample-Length	Average Score
Five-12 second	1.95
Four-15 second	1.95
Two-30 second	1.85

These results show that using the Gaussian Mixture Model produces the most human like sounding samples. While the WFW samples did produce a signal that users thought did sounded more human like, comments were made that the GMM transformed samples sounded "smoother". The "smoothness" of the voice equated too more human like transitions from word to word. Table 4.3 shows the results from this test.

Table 4.3 Transformation Method Mean Opinion Score

Transformation Method	Average Score
Weighted Frequency Warping	3.3
Gaussian Mixture Model	3.8

4.2.2 Squeak Observations

The extreme samples that are shown correspond to "squeaks" in the signal. Extreme samples correlate to any sample that is higher than the highest peak of the normal signal. Refer to figure 4.1 for an example of Extreme Samples. Again these

squeaks represent unmatched data. Each test will have a table that shows the number of squeaks that occurred and the percentage of the signal that was unmatched.

Figures 4.1 and 4.3 shows that there were several squeaks recorded in the 4th and 32nd order cases. The 16th order GMM had no squeaks recorded at all, Figure 4.2. The larger number of squeaks recorded in the 32nd order example could be accounted for because the GMMs are looked at as states during the transformation process and the larger number of states require more precise examples of the phoneme that is spoken. The smaller number of GMMs works in the same manner but thus causing for more broad and unspecified phoneme matches.

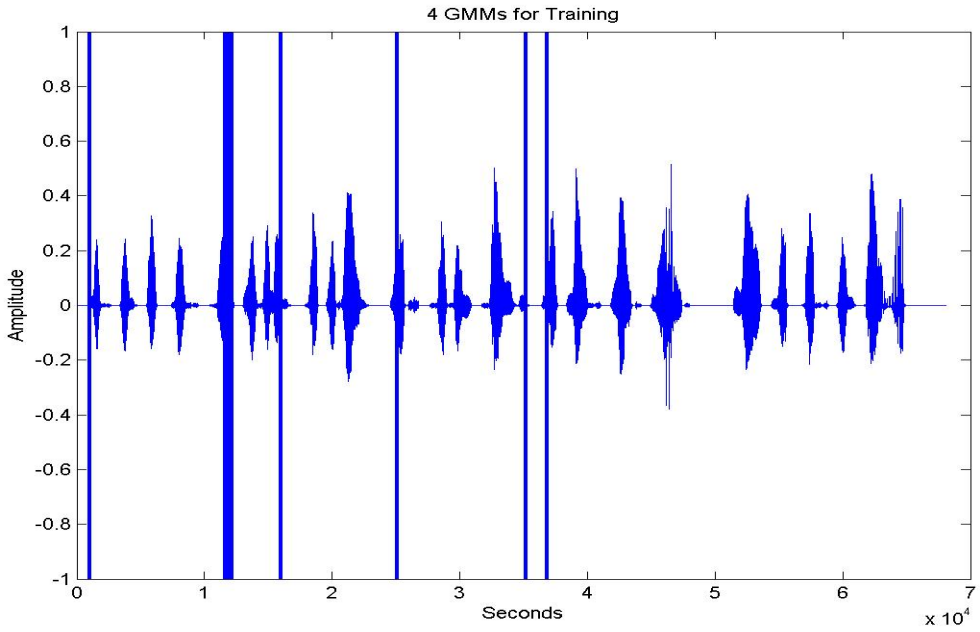

Figure 4.1 4th Order GMM Signal

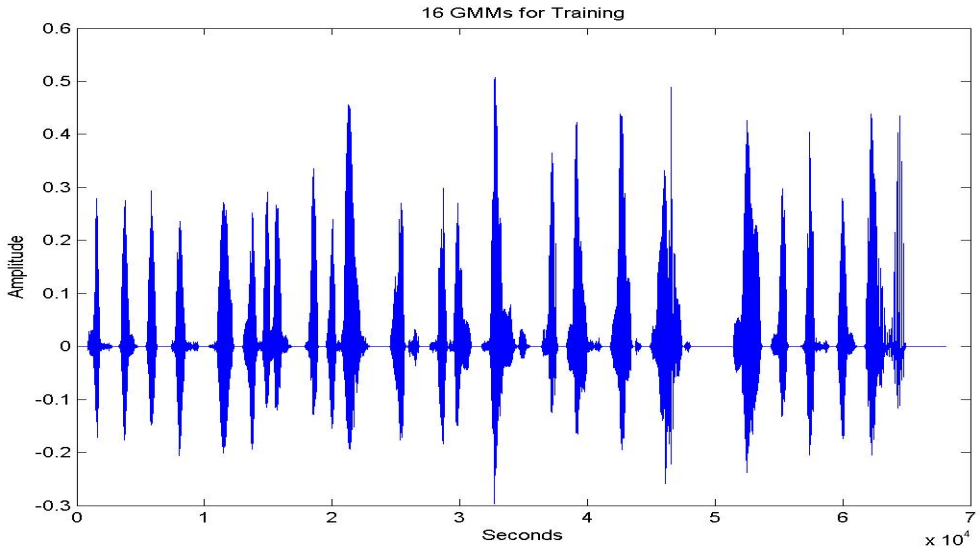

Figure 4.2 16th Order GMM Signal

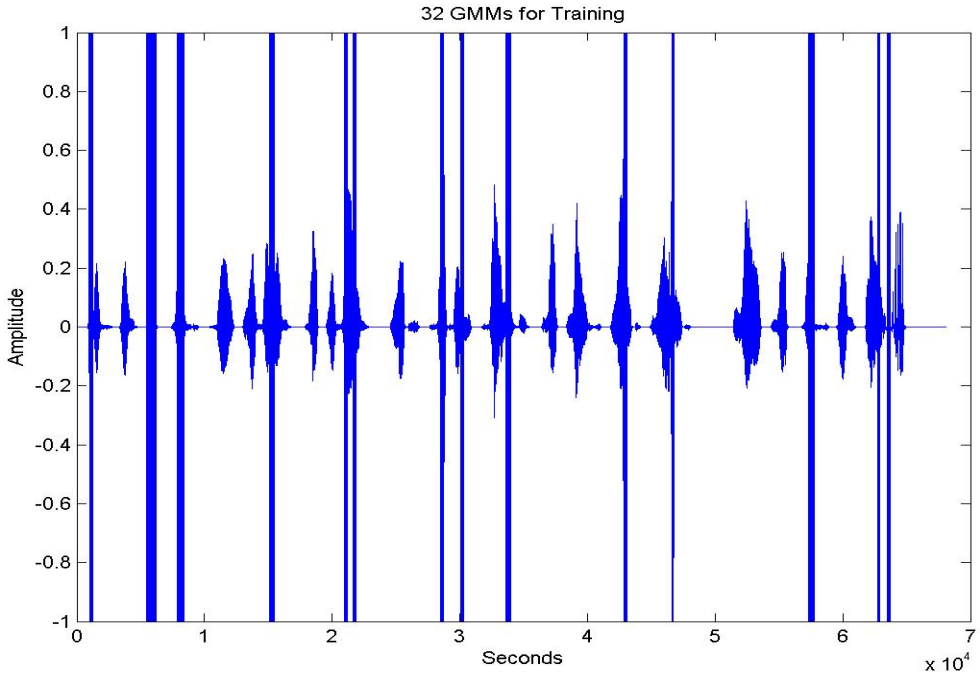

Figure 4.3 32nd Order GMM Signal

The 16th order GMM for this given set of data yielded the clearest results in terms of squeaks. While the pieces that were converted in both the 4th and 32nd order did sound similar to the 16th order GMM conversion, the phonemes that did not match yielded more corrupted results. Table 4 shows the number of squeaks that were found in the data sets. Thus proving that the 16th order GMM was the most efficient for conversion of the signal.

Table 4.4 Order of GMM Squeak Count

GMM Order	Unmatched Data Instances (Squeaks)	Total Unmatched Percentage of the Signal
4th	6	2.8%
16th	0	0.1%
32nd	14	6.6%

The results of this test have shown that the training set with the most number of training signals has yielded the clearest results. Figure 4.4 has shown that using five-12 sec signals yields a system in which all the training data is matched. Figure 4.4 show shows that by decreasing the number training signals by 1 we get a drastic change in the number of unmatched samples. While the message that was used is the same, the more instances of the samples yielded clearer results. Figure 4.6 shows that by using 2 training signals you will yield the largest amount of untrained samples. This could be due to many factors such as the use of the nearest neighbor algorithm and the nun parallel training. With larger sample sets there could be more instances of similar phonemes,

which would provide larger matched samples. Table 5 shows the number of Samples that had unmatched signals

Table 4.5 Number of Samples Squeak Count

Sample-Length	Unmatched Data Instances (Squeaks)	Total Unmatched Percentage of the Signal
Five-12 second	0	0%
Four-15 second	10	5.4%
Two-30 second	11	7.8%

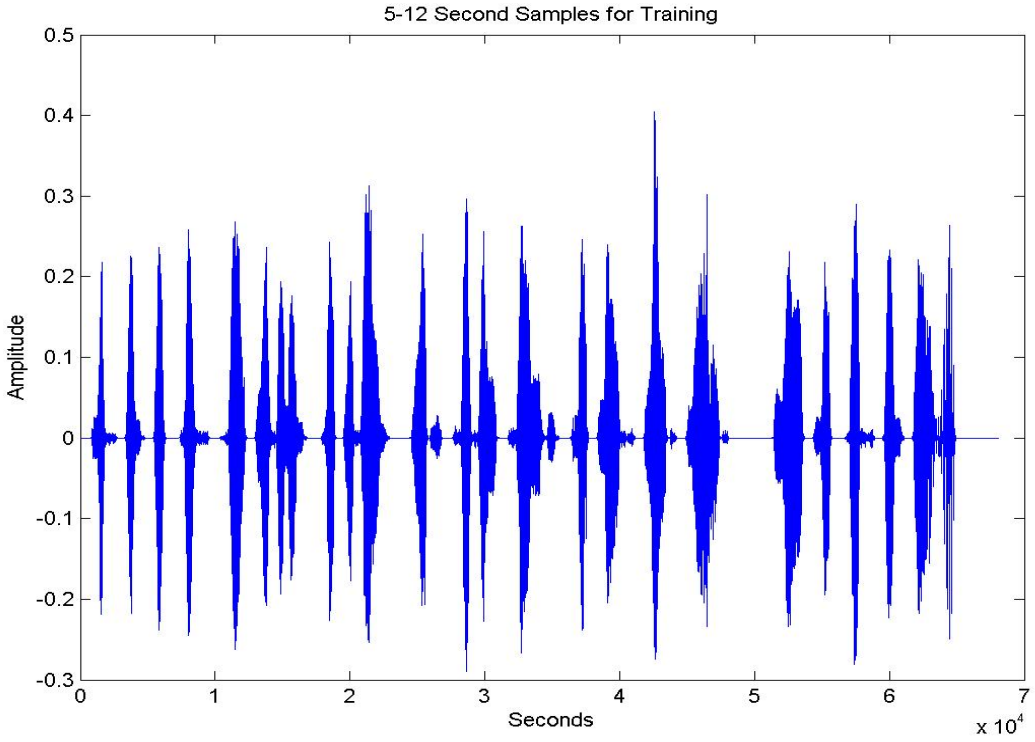

Figure 4.4 Five- 12 Second Samples

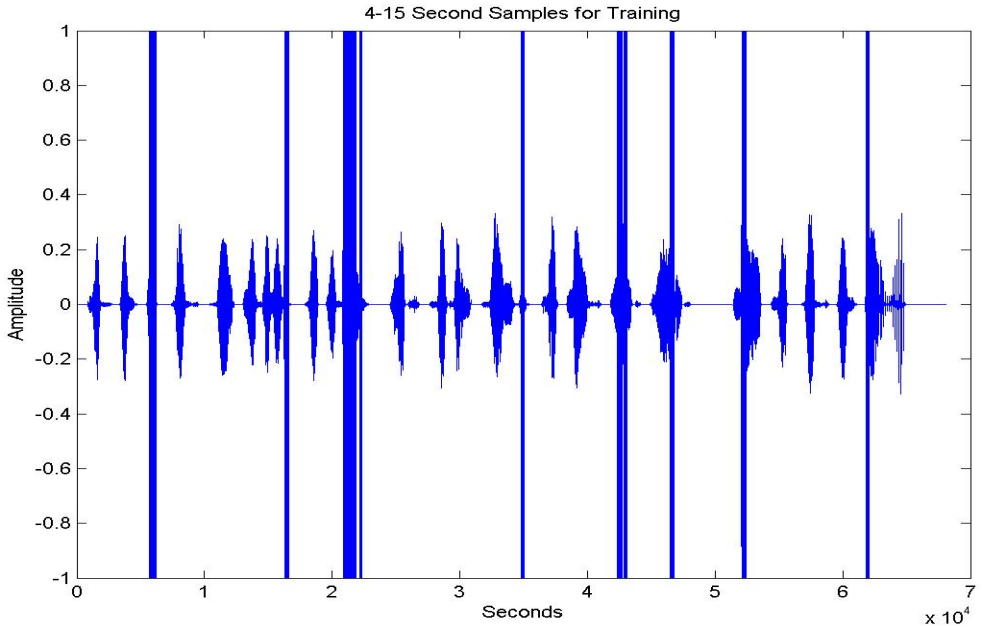

Figure 4.5 Four-15 Second Samples

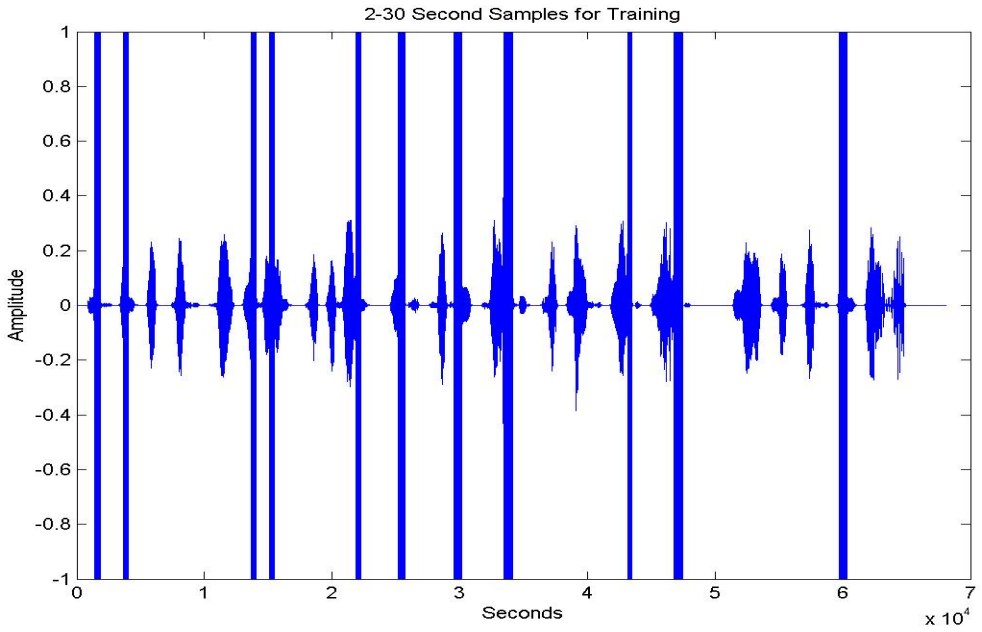

Figure 4.6 Two-30 Second Samples

The results for this test showed that the change in transformation functions did not play a role in how many squeaks were heard. That is predictable, as the transformation function has little to do with the training of the data sets. So this test resulted in perfectly clear results for both instances.

Table 4.6 Transformation Method Squeak Count

Transformation Method	Unmatched Data Instances (Squeaks)	Total Unmatched Percentage of the Signal
WFW	0	0%
GMM	0	0%

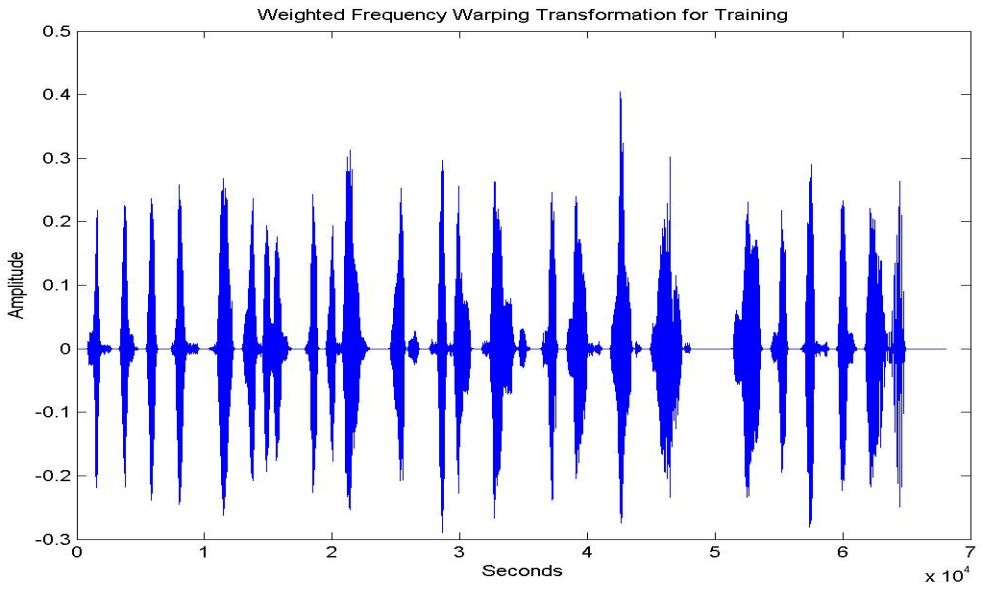

Figure 4.7 Weighted Frequency Warping Function

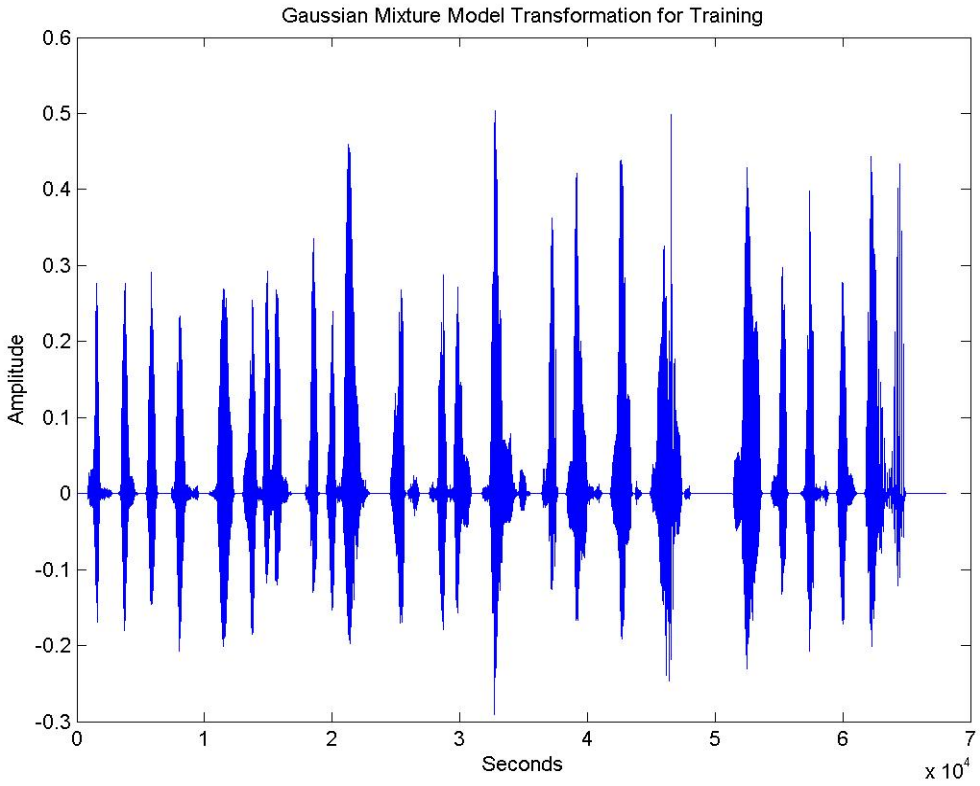

Figure 4.8 Gaussian Mixture Model Based Function

CHAPTER 5

CONCLUSION

5.1 Summary

This thesis has addressed the area of finding a practical method for users to create user specific human sounding synthesized voice from a TTS using a social network as a source for training speech data, as well as text to be converted. With the increase focus on textual messages and less focus on actually spoken messages TTS applications have been created to give users the capability to convert text messages to speech. While these technologies are very helpful for the intended purpose of converting text-to-speech, the emotional component of speech is sacrificed by a synthesized voice that cannot show emotion. This system has aided in the solution to that issue by the use of a voice conversion system that can edit the vocal characteristics of synthesized speech to sound more like human voice.

5.2 Contributions

The contributions of this project to the area of synthesized speech to real speech voice conversion, is that it allows each level of input to be acquired via social media. The system has the capabilities of getting the text that has been extracted from a tweet through twitter. The use of Cinch as a means for voice capture makes further use of social networking. The area of "socially acquired" data as a means for engineering research has not been a heavy focus area for research. By being able to manipulate the vocal characteristics of voice this can be contribute to the ultimate goal, which is a system that will include TTS conversion that includes emotion through textual inferences.

Additionally, the application of social media to the field of Digital Signal Processing has the potential to aid in the development of STEM related interest in younger students. The incorporation of an audio use for the UMSEE model contributes to the research that Dr. Corey Graves at North Carolina A&T State University is doing in an effort to increase STEM interest.

This thesis has also confirmed areas of research as it results to Voice Conversion and given parameter. This system has also tested the voice conversion capabilities of the ESPEAK TTS System. The results shown in chapter 4 gives the explanation of how GMM order, Number of training signals, and the type of voice conversion that is used can affect the quality of converted signal from Espeak.

5.3 Recommendations for Future Direction

There are several areas that could be further developed. First, the investment into a more natural sounding TTS system would yield better results. While the Espeak system did give the results from the standpoint of vocal characteristics, but from the standpoint of intonation and articulation it still maintained a "computerized voice". Secondly the development of other Signal Processing functions with socially acquired data could be used. The development of technologies that could incorporate music would also be area that could be incorporated. For example conversion of songs to another persons voice could be developed. Also a process to make could make this system automated would be very helpful for everyday usage. Developing a system that could handle the large amounts of users would also make this system better. Lastly, investigating use of any Social Media network as means for textual messages, other than Twitter (i.e. the use of

ones Facebook Status as a means for text) would be useful. By doing this the scope of voice conversion can become completely universal.

REFERENCES

[1] D. Beaver. "10 Billion Photos" Facebook. 14 October 2008. Web. 2 November 2011. http://www.facebook.com/note.php?note_id=30695603919

[2] M. Smith, C. Giraud, "Bonding vs. Bridging Social Capital: A Case Study in Twitter", *Social Computing, 2010 IEE Second International Conference*, pp. 385-392, 2010

[3] F. Kivran-Swaine, M. Naaman, "Network Properties and Social Sharing of Emotions in Social Awareness Streams", *Journal of Comp. Mediated Comm. 12, 4 (2007), 1143-1168.*

[4]Unknown, "Learn More About Siri", Web. 1 March 2012. http://www.apple.com/iphone/features/siri-faq.html

[5] D. Jons, "Espeak Text to Speech",22 Feb 2012. Web. 2 March 2012. http://espeak.sourceforge.net/

[6] Suendermann, "*Voice Conversion MATLAB Toolbox*", Seimens Corporate Technology, 2007

[7]M. Brooks, "*Voicebox: Speech Processng Toolbox for MATLAB*" Web. 22 February 2012. http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[8] D. Erro, "*UPC Toolkit for HSM-based Voice Conversion: Small Manual*", Polytechnic University of Catalonia, 2007

[9] B. Judd (2011). "A UBIQUITOUS MOBILE SYSTEM FOR STEM EDUCATION ENCHANCEMNT USING CELLULAR PHONE MESSAGING, SOCIAL NETWORKING TECHNOLOGY AND HIGH LEVEL COMPUTATION/VISUALIZATION" Master Thesis. North Carolina A&T State University: U.S., 2011

[10] eBiz MBA, "Top 15 Most Popular Social Networking Sites" 1 March 2012, Web. 13 March 2012, http://www.ebizmba.com/articles/social-networking-websites

[11] D. Erro, A. Moreno, A. Bonafonte, "Voice Conversion based on Weighted Frequency Warping", *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 5, pp. 922-931, 2010

[12] Rabiner, Lawrence, and Biing-Hwang Juang. Fundamentals of Speech Recognition. Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[13] D. Reynolds, "Gaussian Mixture Models", *IEEE ASSP Magazine pp. 4–29 1984*

[14] M. Rafieee, S. Jafari, H. Ahmadi, M. Jafari "Considerations to Spoken Language Recognition for Text-to-Speech Applications", *IEEE Trans. on Audio, Speech, and Language Processing, vol. 16, no. 7, 2008 PP. 304-309*

[15] J. Brown, P. Smaragdis, "Hidden Markov And Gaussian Mixture Models For Automatic Call Classification" Published Online 2009

[16] K. Liu, J. Zhang, Y. Yan, "High Quality Voice Conversion through Phoneme-based Linear Mapping Functions with STRAIGHT for Mandarin*", Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, 2007

[17] Z. Shuang, R. Bakis, Y. Qin, "Voice Conversiont based on Mapping Formants", *TC-STAR Workshop on Speech-to-Speech Translation,* pp. 219-223, 2006

[18] H. Duxans, D. Erro, J. Perez, F. Diego, A. Bonafonte, .A. Moreno, "Voice Conversion of Non-aligned Data using Unit Selection", *TC-Star Workshop on Speech to Speech Translation, pp 237-242, 2006*

[19] D. Erro, *"Intra-Lingual and Cross Lingual Voice Conversion Using Harmonic Plus Stochastic Models"*, PhD. Dissertation, Polytechnic University of Catalonia,2008

[20]V. Goncharoff and P. Gries, "An Algorithm for Accu- rately Marking Pitch Pulses in Speech Signals," in *Proc. of the SIP*, Las Vegas, USA, 1998.

[21] Cinchcast Inc. "Homepage" Web. 7 December 2011 http://cinch.fm

[22] Unknown, "Manual: Short URL"15 March 2012 Web.14 March 2012,http://www.mediawiki.org/wiki/Manual:Short_URL

[23] H. Barrob´es, "*Voice Conversion Applied to Text-To-Speech Systems*", PhD. Dissertation, Polytechnic UNiverty of Catalonia, 2006

[24] M. Lee, J. Santen, B. Mobius, J. Olive "Format Tracking Using Context-Dependent Phonemic Information", *IEEE Transactions On Speech And Audio Processing, Vol. 13, No. 5, pp. 741-750, 2005*

[25] W. Gao, Q. Cao "Frequency Warping for Speaker Adaption of Text-to-speech Synthesis", *IEEE Transactions Audio, Speech and Language Processing, Vol. 15, No. 8, pp. 2222-2235, 2007*

[26] J. Hao, T. Lee, T. Sejnowski, "Speech Enhancement Using Gaussian Scale Mixture Models", *IEEE Transactions On Audio, Speech, And Language Processing, Vol. 18, No. 6, 2010 1127*

[27] M. Mohamed, P. Gader, "Generalized Hidden Markov Models—Part I: Theoretical Frameworks" *IEEE Transactions On Fuzzy Systems, Vol. 8, No. 1, 2000 pp. 67-81*

[28] Alfredo Fernandez, "MP3WRITE and MP3READ", 16 Febuaray 2006. Web. 24 Feburary 2012, www.mathworks.com/matlabcentral/fileexchange/6152-MP3WRITE and MP3READ

[29] D. Valencia, "*CROSS-LANGUAGE MOBILE COMMUNICATION AND PERVASIVE LANGUAGE LEARNING USING MULTIMEDIA CELLULAR PHONE MESSAGING AND ONLINE TRANSLATION*", " Master Thesis. North Carolina A&T State University: U.S., 2011

APPENDIX A

TWITTER BASED APPLICATION

```
using System;
using System.Collections.Generic;
using System.Configuration;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Web;
using System.Xml;
using System.IO;
using System.Diagnostics;
using System.Runtime.Serialization.Json;
using System.Text.RegularExpressions;
using System.Net.Sockets;
using System.Net.Security;
using System.Net.Mail;
using MLApp;

namespace CodingTheTweet

{

    public partial class MatLabTwitterApp : Form

{

private WebClient client = new WebClient();
string TwitpicUrl = "";
string TwitpicCommand = "";
string LastMention = "";
int exiter = 0;
string cinch = "";
Tweets TweetObj = new Tweets();


public MatLabTwitterApp()

    {

    InitializeComponent();


}//end constructor

private void Form1_Load(object sender, EventArgs e)

    {
// Set up our credentials...
_oAuth.Token = Settings1.Default.token;
_oAuth.TokenSecret = Settings1.Default.secretToken;
_oAuth.ConsumerKey = Settings1.Default.consumerKey;
_oAuth.ConsumerSecret = Settings1.Default.consumerSecret;
_oAuth.PIN = Settings1.Default.pin;


    }
```

```
        public bool IsConfigured

            {

    get

                {

    return !String.IsNullOrEmpty(Settings1.Default.token) &&
      !String.IsNullOrEmpty(Settings1.Default.secretToken) &&
      !String.IsNullOrEmpty(Settings1.Default.consumerKey) &&
      !String.IsNullOrEmpty(Settings1.Default.consumerSecret) &&
      !String.IsNullOrEmpty(Settings1.Default.pin);

                }

            }



        private oAuthTwitter _oAuth = new oAuthTwitter();

        private void TweetCheck_Tick(object sender, EventArgs e)

            {

    TweetCheck.Enabled = true;

            }




//*****************************************************************************************

  //REGISTRATION PROCESS
  private void Voice_Reg_Click(object sender, EventArgs e)

                {

            StreamWriter sw;
            StreamWriter sw2;
            string[] final_message = new string[20];
            string[] final_from = new string[20];
            string[] final_to = new string[20];

            //Create the Database if it doesn't exist
            if (!Directory.Exists("c:\\UMSEE"))
            Directory.CreateDirectory("c:\\UMSEE");
            if (!File.Exists("c:\\UMSEE\\usersfile.txt"))

                    {

    sw = new StreamWriter("c:\\UMSEE\\usersfile.txt");
    sw.Close();
    }//end if
```

```
    if (!File.Exists("c:\\UMSEE\\ProgramInfo"))

                {

    sw2 = new StreamWriter("c:\\UMSEE\\ProgramInfo");
    sw2.WriteLine(LastMention);
    sw2.Close();
    }//end if


         //Read the last mention
         LastMention = File.ReadAllText("c:\\UMSEE\\ProgramInfo").Trim();


  //***************Parse the XML page and display the last 20 tweets**************


         /*initialize a new instance of the XmlDocument class*/
         XmlDocument xDoc = new XmlDocument();
         XmlDocument xDoc2 = new XmlDocument();
         string DirectMessage = "";
         string Mentions = "";
         //load the XML document from the twitter stream
         try

                 {
         DirectMessage = _oAuth.oAuthWebRequest(
         oAuthTwitter.Method.GET,
         "http://api.twitter.com/1/direct_messages.xml",
         "screen_name=pseevoice");
         xDoc.InnerXml = DirectMessage;

                 }

    catch { }
    try

                {

    if (LastMention == "")

                    {
    //check the mentions as well
    Mentions = _oAuth.oAuthWebRequest(
    oAuthTwitter.Method.GET,
    "http://api.twitter.com/1/statuses/mentions.xml",
    "screen_name=pseevoice");
    xDoc2.InnerXml = Mentions;
    }//end if
    else

                    {

    Mentions = _oAuth.oAuthWebRequest(
    oAuthTwitter.Method.GET,
    "http://api.twitter.com/1/statuses/mentions.xml",
    "screen_name=pseevoice&sinceid=" + LastMention);
    xDoc2.InnerXml = Mentions;
    }//end else

                }

    catch { }

  //method GetElementsByTagName() obtains the addresses of a collection of elements that
match the specified name
    XmlNodeList tweets = xDoc.GetElementsByTagName("text");
```

```
      XmlNodeList tweetid = xDoc.GetElementsByTagName("id");
      XmlNodeList screen_name = xDoc.GetElementsByTagName("sender_screen_name");
      //Mentions
      XmlNodeList Mentionid = xDoc2.GetElementsByTagName("id");
      XmlNodeList MentionTweet = xDoc2.GetElementsByTagName("text");
      XmlNodeList Mentionscreen_name = xDoc2.GetElementsByTagName("screen_name");


            //Create an array to convert the XmlNodeList to an array
            string[] tweetarray = new string[tweets.Count];
            string[] tweetidarray = new string[tweetid.Count];
            string[] screen_namearray = new string[screen_name.Count];
            //mentions
            string[] Mentionidarray = new string[Mentionid.Count];
      string[] Mentionscreen_namearray = new string[Mentionscreen_name.Count];
      string[] MentionTweetarray = new string[MentionTweet.Count];
            //Populates the Mention Array that will be used for parsing
            for (int i = 0; i < 10; i++)

                    {

      Mentionidarray[i] = Mentionid[i].InnerText.ToString();
      Mentionscreen_namearray[i] = Mentionscreen_name[i].InnerText.ToString();
      MentionTweetarray[i] = MentionTweet[i].InnerText.ToString();

                    }
            //Remove Duplicate names from the screen name array
            string[] duplicates = RemoveDuplicates(Mentionscreen_namearray);

            //Search mentions for twitpics
            for (int j = 0; j < duplicates.Length - 1; j++)//USer For

                    {

  if (!Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + duplicates[j]))

                        {

      for (int i = 0; i < 5; i++)//All Mentions For

                            {
            if (MentionTweetarray[i].Contains("t.co"))

                                {

      TweetCheck.Enabled = false;  // disable timer
        //Checks to see if message contains Phrase 1
            if (MentionTweetarray[i].Contains("??Phrase1"))

                                    {

      //Splits the entire message up based on ' '
  TwitpicCommand = MentionTweetarray[i].Substring(MentionTweetarray[i].IndexOf("??") +
2);
      string[] words = MentionTweetarray[i].Split(' ');
      foreach (string word in words)

                                        {

      //finds the word that has the t.co link
      if (word.Contains("t.co"))

                                            {

      //Removes the excess spaces from the t.co
      TwitpicUrl = word.Trim();
      string down = " ";
```

```
  //Preparing to expand the ural and get the needed link to download the MP# directly
from the site
      WebClient webClient = new WebClient();
  //expandurl.appspot.com expands the t.co Short Url to all of its previous forms
  byte[] myDataBuffer = webClient.DownloadData("http://expandurl.appspot.com/expand?url="
+ TwitpicUrl);

// Display the downloaded data.
  string download = Encoding.ASCII.GetString(myDataBuffer);
  //String Manipulation to extract the 5 digit code at the end of the cinch.fm link
      int start = download.IndexOf("end_url", 1);
      int end = download.IndexOf("redirects", 1);
  down = download.Substring(start + 13, (end - start - 17));
      string final = down.Replace("\\/", "/");
      string[] userinfo = final.Split('/');
      //Declares which user file the data will be saved in
      string cinchusername = userinfo[0];
      cinch = cinchusername;
      string filename = userinfo[1];
  //appends the extracted code to the link so that it can be downloaded directly
      string finalurl = "http://cinch.fm/" + final;


  if (Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg001.mp3"))
      break;
      else

                                                {

        Directory.CreateDirectory("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3");

                                                }

  //declares which folder the file will be downloaded to
  string destination = "c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg001.mp3";
  //download the file from the web
  webClient.DownloadFile((finalurl + ".mp3"), destination);
           break;

                                            }
           }//end foreach
       }

  else if (MentionTweetarray[i].Contains("??Phrase2"))



//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                                     {



           //Splits the entire message up based on ' '
  TwitpicCommand = MentionTweetarray[i].Substring(MentionTweetarray[i].IndexOf("??") +
2);
      string[] words = MentionTweetarray[i].Split(' ');
      foreach (string word in words)

                                          {

      //finds the word that has the t.co link
      if (word.Contains("t.co"))
```

```
                                                {

      //Removes the excess spaces from the t.co
      TwitpicUrl = word.Trim();
      string down = " ";


  //Preparing to expand the ural and get the needed link to download the MP# directly
from the site
      WebClient webClient = new WebClient();
  //expandurl.appspot.com expands the t.co Short Url to all of its previous forms
  byte[] myDataBuffer = webClient.DownloadData("http://expandurl.appspot.com/expand?url="
+ TwitpicUrl);


          // Display the downloaded data.
  string download = Encoding.ASCII.GetString(myDataBuffer);
  //String Manipulation to extract the 5 digit code at the end of the cinch.fm link
      int start = download.IndexOf("end_url", 1);
      int end = download.IndexOf("redirects", 1);
  down = download.Substring(start + 13, (end - start - 17));
      string final = down.Replace("\\/", "/");
      string[] userinfo = final.Split('/');
      //Declares which user file the data will be saved in
      string cinchusername = userinfo[0];
      cinch = cinchusername;
      string filename = userinfo[1];
  //appends the extracted code to the link so that it can be downloaded directly
      string finalurl = "http://cinch.fm/" + final;


  if (Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg002.mp3"))
      break;
      else

                                                {

       Directory.CreateDirectory("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3");

                                                }

  //declares which folder the file will be downloaded to
  string destination = "c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg002.mp3";
  //download the file from the web
  webClient.DownloadFile((finalurl + ".mp3"), destination);
          break;

                                                }

  }//end foreach

                                        }

  else if (MentionTweetarray[i].Contains("??Phrase3"))



//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                                        {

          //Splits the entire message up based on ' '
  TwitpicCommand = MentionTweetarray[i].Substring(MentionTweetarray[i].IndexOf("??") +
2);
```

```
      string[] words = MentionTweetarray[i].Split(' ');
      foreach (string word in words)

                                        {

      //finds the word that has the t.co link
      if (word.Contains("t.co"))

                                              {

      //Removes the excess spaces from the t.co
      TwitpicUrl = word.Trim();
      string down = " ";


  //Preparing to expand the ural and get the needed link to download the MP# directly
from the site
      WebClient webClient = new WebClient();
  //expandurl.appspot.com expands the t.co Short Url to all of its previous forms
  byte[] myDataBuffer = webClient.DownloadData("http://expandurl.appspot.com/expand?url="
+ TwitpicUrl);


           // Display the downloaded data.
  string download = Encoding.ASCII.GetString(myDataBuffer);
  //String Manipulation to extract the 5 digit code at the end of the cinch.fm link
      int start = download.IndexOf("end_url", 1);
      int end = download.IndexOf("redirects", 1);
  down = download.Substring(start + 13, (end - start - 17));
      string final = down.Replace("\\/", "/");
      string[] userinfo = final.Split('/');
      //Declares which user file the data will be saved in
      string cinchusername = userinfo[0];
      cinch = cinchusername;
      string filename = userinfo[1];
  //appends the extracted code to the link so that it can be downloaded directly
      string finalurl = "http://cinch.fm/" + final;


  if (Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg003.mp3"))
      break;
      else

                                                    {

        Directory.CreateDirectory("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3");

                                                    }

  //declares which folder the file will be downloaded to
  string destination = "c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg003.mp3";
  //download the file from the web
  webClient.DownloadFile((finalurl + ".mp3"), destination);




           break;

                                              }

  }//end foreach
```

```
                                    }
  else if (MentionTweetarray[i].Contains("??Phrase4"))


//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                                    {

      //Splits the entire message up based on ' '
  TwitpicCommand = MentionTweetarray[i].Substring(MentionTweetarray[i].IndexOf("??") +
2);
      string[] words = MentionTweetarray[i].Split(' ');
      foreach (string word in words)

                                        {

      //finds the word that has the t.co link
      if (word.Contains("t.co"))

                                            {

      //Removes the excess spaces from the t.co
      TwitpicUrl = word.Trim();
      string down = " ";


  //Preparing to expand the ural and get the needed link to download the MP# directly
from the site
      WebClient webClient = new WebClient();
  //expandurl.appspot.com expands the t.co Short Url to all of its previous forms
  byte[] myDataBuffer = webClient.DownloadData("http://expandurl.appspot.com/expand?url="
+ TwitpicUrl);


          // Display the downloaded data.
  string download = Encoding.ASCII.GetString(myDataBuffer);
  //String Manipulation to extract the 5 digit code at the end of the cinch.fm link
      int start = download.IndexOf("end_url", 1);
      int end = download.IndexOf("redirects", 1);
  down = download.Substring(start + 13, (end - start - 17));
      string final = down.Replace("\\/", "/");
      string[] userinfo = final.Split('/');
      //Declares which user file the data will be saved in
      string cinchusername = userinfo[0];
      cinch = cinchusername;
      string filename = userinfo[1];
  //appends the extracted code to the link so that it can be downloaded directly
      string finalurl = "http://cinch.fm/" + final;


  if (Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg004.mp3"))
      break;
      else

                                                {

        Directory.CreateDirectory("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3");

                                                }

  //declares which folder the file will be downloaded to
```

```
  string destination = "c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg004.mp3";
  //download the file from the web
  webClient.DownloadFile((finalurl + ".mp3"), destination);



          break;

                                              }

  }//end foreach

                                     }

  else if (MentionTweetarray[i].Contains("??Phrase5"))



//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

                                     {



          //Splits the entire message up based on ' '
  TwitpicCommand = MentionTweetarray[i].Substring(MentionTweetarray[i].IndexOf("??") +
2);
      string[] words = MentionTweetarray[i].Split(' ');
      foreach (string word in words)

                                          {

      //finds the word that has the t.co link
      if (word.Contains("t.co"))

                                              {

      //Removes the excess spaces from the t.co
      TwitpicUrl = word.Trim();
      string down = " ";


  //Preparing to expand the ural and get the needed link to download the MP# directly
from the site
      WebClient webClient = new WebClient();
  //expandurl.appspot.com expands the t.co Short Url to all of its previous forms
  byte[] myDataBuffer = webClient.DownloadData("http://expandurl.appspot.com/expand?url="
+ TwitpicUrl);


          // Display the downloaded data.
  string download = Encoding.ASCII.GetString(myDataBuffer);
  //String Manipulation to extract the 5 digit code at the end of the cinch.fm link
      int start = download.IndexOf("end_url", 1);
      int end = download.IndexOf("redirects", 1);
  down = download.Substring(start + 13, (end - start - 17));
      string final = down.Replace("\\/", "/");
      string[] userinfo = final.Split('/');
      //Declares which user file the data will be saved in
      string cinchusername = userinfo[0];
      cinch = cinchusername;
      string filename = userinfo[1];
  //appends the extracted code to the link so that it can be downloaded directly
      string finalurl = "http://cinch.fm/" + final;
```

```
  if (Directory.Exists("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg005.mp3"))
      break;
      else

                                                {

       Directory.CreateDirectory("c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3");

                                                }

  //declares which folder the file will be downloaded to
  string destination = "c:/Dwight_Research/UPC_HSM_VC 2/" + cinchusername +
"/sourceMP3/fileorg005.mp3";
  //download the file from the web
  webClient.DownloadFile((finalurl + ".mp3"), destination);




          break;

                                          }



          }//end foreach



          }//End t.co Search




          }//Check for 5
          }//end ALL MEntions For
          //Create VCFUNCTION based
          REGISTER(cinch);


          }//Directory Doesnt Exist
          else

                      {

  MessageBox.Show(" " + duplicates[j] + "Voice is already saved in our database", "Voice
Already Registered",
  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);



                      }


          }//end User For
```

```
            return;

            }//End VOICE_REG_CLICK

            //END REGISTRATION PROCESS

//*************************************************************************************
*****************************

//*************************************************************************************
*****************************

  //BEGINCONVERSION PROCESS
  private void button2_Click(object sender, EventArgs e)

                {

      StreamWriter sw;
      StreamWriter sw2;


            string[] final_message = new string[20];
            string[] final_from = new string[20];
            string[] final_to = new string[20];
            string[] tweet_to_B = new string[20];
            string[] tweet_to_B_ScNm = new string[20];
            //Create the Database if it doesn't exist
            if (!Directory.Exists("c:\\UMSEE"))
            Directory.CreateDirectory("c:\\UMSEE");
            if (!File.Exists("c:\\UMSEE\\usersfile.txt"))

                    {

      sw = new StreamWriter("c:\\UMSEE\\usersfile.txt");
      sw.Close();
      }//end if
      if (!File.Exists("c:\\UMSEE\\ProgramInfo"))

                  {

      sw2 = new StreamWriter("c:\\UMSEE\\ProgramInfo");
      sw2.WriteLine(LastMention);
      sw2.Close();
      }//end if


            //Read the last mention
            LastMention = File.ReadAllText("c:\\UMSEE\\ProgramInfo").Trim();


  //***************Parse the XML page and display the last 20 tweets**************


            /*initialize a new instance of the XmlDocument class*/
            XmlDocument xDoc = new XmlDocument();
            XmlDocument xDoc2 = new XmlDocument();
            string DirectMessage = "";
```

```
	string Mentions = "";



	//load the XML document from the twitter stream
	try
		{


	DirectMessage = _oAuth.oAuthWebRequest(
							oAuthTwitter.Method.GET,
						"http://api.twitter.com/1/direct_messages.xml",
"screen_name=pseevoice");


	xDoc.InnerXml = DirectMessage;
		}
catch { }
try
		{
if (LastMention == "")
		{
//check the mentions as well
Mentions = _oAuth.oAuthWebRequest(
							oAuthTwitter.Method.GET,
						"http://api.twitter.com/1/statuses/mentions.xml",
"screen_name=pseevoice");
xDoc2.InnerXml = Mentions;
}//end if
else
		{
Mentions = _oAuth.oAuthWebRequest(
							oAuthTwitter.Method.GET,
						"http://api.twitter.com/1/statuses/mentions.xml",
"screen_name=pseevoice&sinceid=" + LastMention);
xDoc2.InnerXml = Mentions;
}//end else
		}
catch { }
```

```
  //method GetElementsByTagName() obtains the addresses of a collection of elements that
match the specified name
      XmlNodeList tweets = xDoc.GetElementsByTagName("text");
      XmlNodeList tweetid = xDoc.GetElementsByTagName("id");
  XmlNodeList screen_name = xDoc.GetElementsByTagName("sender_screen_name");
      //Mentions
      XmlNodeList Mentionid = xDoc2.GetElementsByTagName("id");
      XmlNodeList MentionTweet = xDoc2.GetElementsByTagName("text");
  XmlNodeList Mentionscreen_name = xDoc2.GetElementsByTagName("screen_name");


           //Create an array to convert the XmlNodeList to an array
           string[] tweetarray = new string[tweets.Count];
           string[] tweetidarray = new string[tweetid.Count];
           string[] screen_namearray = new string[screen_name.Count];
           //mentions
           string[] Mentionidarray = new string[Mentionid.Count];
  string[] Mentionscreen_namearray = new string[Mentionscreen_name.Count];
      string[] MentionTweetarray = new string[MentionTweet.Count];



           //Populate MentionTweet


           for (int i = 0; i < 19; i++)

                   {



  Mentionscreen_namearray[i] = Mentionscreen_name[i].InnerText.ToString();
      MentionTweetarray[i] = MentionTweet[i].InnerText.ToString();

                   }

      //populate DMs



           //Search mentions for twitpics



           int count = 0;


           for (int i = 0; i < 19; i++)

                   {



           if (MentionTweetarray[i].Contains("o_o"))

                       {

      tweet_to_B[count] = MentionTweetarray[i];
      tweet_to_B_ScNm[count] = Mentionscreen_namearray[i];
      count++;

                       }

                   }
```

```
      for (int i = 0; i < count; i++)

                  {

      string[] message = tweet_to_B[i].Split(' ');
      string username = tweet_to_B_ScNm[i];
      string messanger = "";




            for (int j = 3; j < Convert.ToInt16(message.Length); j++)

                        {

      messanger = messanger + " " + message[j];

                        }



            messanger = "\"" + messanger + "\"";
            final_message[i] = messanger;
            final_from[i] = username;
            final_to[i] = message[2];
            bool loop = check_user(final_to[i]);
            while (loop == false)

                        {

  MessageBox.Show("" + final_to[i] + " was not found in our database please fill in the
proper fields and click Submit", "Registration Error",
      MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
      exiter = 1;
      break;


                        }

      if (exiter == 1)

                        {

      break;

                        }



            bool looper = check_user(final_from[i]);
            while (looper == false)

                        {

  MessageBox.Show("" + final_from[i] + " was not found in our database please fill in the
proper fields and click Submit", "Registration Error",
  MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
      exiter = 1;
      break;


                        }
```

```
            if (exiter == 1)

                        {

      break;

                        }



            DialogResult answer;
  answer = MessageBox.Show(final_from[i] + " Would like to send the message " +
final_message[i] + " to " + final_to[i], "Confirmation",
  MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);
      if (answer == DialogResult.Yes)

                        {

      wavcreate(final_from[i], final_message[i]);
  matlab_Command(@" cd ('C:\Dwight_Research\UPC_HSM_VC 2\'); cinch_wfw('" + final_from[i]
+ "') ");
      string final_email = get_email(final_to[i]);
      for (int delay = 0; delay > 1000000000; )

                              {

                              }

      //  SendMessage(final_email, final_message[i], final_from[i]);


            MessageBox.Show("Message Sent", "Confirmation",
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
            //Decides if we want send the next message
  bool keep_going = new_message(final_to[i + 1], final_from[i + 1], final_message[i +
1]);
      if (keep_going == false)

                              {



            break;

                              }

                        }

  } //end for

                }




            //END REGISTRATION PROCESS

//*************************************************************************************
*****************************
```

```
//*************************************************************************************
*****************************

  //ADDITIONAL FUNCTIONS


           //Checks to see if Users wants to convert the next message
           private bool new_message(string to, string from, string message)

               {

      DialogResult result;
  result = MessageBox.Show("Would you like to send the next message", "Send Next
Message",
      MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation);


           if (result == DialogResult.No)

                    {

      return false;

                    }

      else if (result == DialogResult.Yes)

                    {

      return true;

                    }

      return false;

               }







           //Runs the MATLAB code to Register voice
           private void REGISTER(string name)

               {


  string answer = matlab_Command(@" cd ('C:\Dwight_Research\UPC_HSM_VC 2\');
cinch_register('" + name + "') ");




           answer = answer.Replace("\n", "");
           MessageBox.Show("" + name + " Your voice is saved", "Confirmation",
           MessageBoxButtons.OK, MessageBoxIcon.Exclamation);



               }
```

```
          //Creates wavefile from the Twitter
          private void wavcreate(string user, string message)

              {



          //string message is the message that was extracted from Twitter
          //String user
          //DOS Command to save a .wav file
  string text = @"cd C:\Documents and Settings\dahudson\My Documents\Visual Studio
2008\Projects\WindowsFormsApplication2\WindowsFormsApplication2\bin\Debug\eSpeak\command_
line" + Environment.NewLine +


  @"espeak -v en -g 10 -w C:\Dwight_Research\" + user + @"TEMP_ORG.wav " + message;


          //Writes that command to a BATCH file
          System.IO.File.WriteAllText(@"C:\First.bat", text);


          //Runs BATCH file in DOS
          System.Diagnostics.Process.Start(@"C:\First.bat");




              }



          //Checks to see if user is in the USERFILE
          private bool check_user(string user)

              {

      string sample = "";
      int line_count = 0;
      string[,] info = new string[10, 3];
      string telephone = "";
      string carrier = "";
      string username = "";


          string[] email = new string[20];
          // Create an instance of StreamReader to read from a file.
          // The using statement also closes the StreamReader.
  using (StreamReader sr = new StreamReader(@"C:\Dwight_Research\USERFILE.txt"))

                  {

      String line;
      // Read and display lines from the file until the end of
      // the file is reached.
      while ((line = sr.ReadLine()) != null)

                      {

      sample = line;
      string[] all = sample.Split(' ');
      username = all[0];
      carrier = all[1];
      telephone = all[2];
```

```
    info[line_count, 0] = username;
    info[line_count, 1] = carrier;
    info[line_count, 2] = telephone;
email[line_count] = email_create(info[line_count, 2], info[line_count, 1]);
//Writes Emails and USername to file to be used when sending the message


          line_count++;

                      }

                 }

    for (int i = 0; i < line_count; i++)

                 {

    if (user == info[i, 0])

                      {



          return true;

                      }

                 }



          return false;

              }



          //Creates Email address from carrier and number
          public string email_create(string Number, string Carrier)

              {

    if (Carrier.Contains("Sprint"))//if carrier is sprint
    Number = string.Concat(Number, "@pm.sprint.com");
    else if (Carrier.Contains("VERIZON"))//if carrier is sprint
    Number = string.Concat(Number, "@vzwpix.com");
    else if (Carrier.Contains("ATT"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.att.net");
    else if (Carrier.Contains("CRICKET"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.mycricket.com");
    else if (Carrier.Contains("NEXTEL"))//if carrier is sprint
    Number = string.Concat(Number, "@messaging.nextel.com");
    else if (Carrier.Contains("ALLTELL"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.alltel.net");
    else if (Carrier.Contains("Verizon"))//if carrier is sprint
    Number = string.Concat(Number, "@vzwpix.com");
    else if (Carrier.Contains("Alltell"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.alltel.net");
    else if (Carrier.Contains("SPRINT"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.alltel.net");
    else if (Carrier.Contains("Att"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.att.net");
    else if (Carrier.Contains("Cricket"))//if carrier is sprint
    Number = string.Concat(Number, "@mms.mycricket.com");
    else if (Carrier.Contains("Nextel"))//if carrier is sprint
    Number = string.Concat(Number, "@messaging.nextel.com");
    return Number;
```

```
            }



        // Used to run MATLAB Commands
        public String matlab_Command(String cmd)

            {

    MLAppClass ml = new MLAppClass();
    ml.Execute(cmd);
    string ans = "DONE";
    return ans;

            }







        //Sends the Converted Message from the From User to the To User
  public void SendMessage(string SendTo, string MessageToSend, string from)

            {

    try

                {

    //create the mail message
    MailMessage mail = new MailMessage();
    mail.From = new MailAddress("pseevoice@gmail.com");


        mail.To.Add(SendTo);
        mail.Body = MessageToSend;


        System.Net.Mail.Attachment attachment;
  attachment = new System.Net.Mail.Attachment(@"C:/Dwight_Research/UPC_HSM_VC 2/" + from
+ "/FINAL_OUT/output.wav");
    mail.Attachments.Add(attachment);



        SmtpClient smtp = new SmtpClient("smtp.gmail.com");


        smtp.EnableSsl = true;
  smtp.Credentials = new NetworkCredential("pseevoice@gmail.com", "convoice");


        smtp.Send(mail);
        }//end try
        catch (Exception ex)

                {

    MessageBox.Show(ex.ToString(), "SendMessage");
```

```
                }


        }//end SendMessage


        //Captures the email address of a given user from the EMAILFILE.txt
        public string get_email(string username)

            {



        string email = "";


using (StreamReader sr = new StreamReader(@"C:\Dwight_Research\EMAILFILE.txt"))

                {

    String line;
    // Read and display lines from the file until the end of
    // the file is reached.
    while ((line = sr.ReadLine()) != null)

                    {

    string sample = line;
    string[] all = sample.Split(' ');
    username = all[0];
    email = all[1];

                    }

                }

    return email;

            }



        //Remove Duplicates from the list
        public static string[] RemoveDuplicates(string[] s)

            {

    HashSet<string> set = new HashSet<string>(s);
    string[] result = new string[set.Count];
    set.CopyTo(result);
    return result;

            }



        //Used to Input new users into the USERFILE.TXT
        private void button1_Click(object sender, EventArgs e)

            {

    string new_username = textBox1.Text;
    string new_carrier = textBox2.Text;
    string new_telephone = textBox3.Text;
    string text = new_username + " " + new_carrier + " " + new_telephone;
```

```
using (StreamWriter sw = File.AppendText(@"C:\Dwight_Research\USERFILE.txt"))

                {

    sw.WriteLine(text);


                }



         //Modifies the EMAILFILE.TXT with the new users information
         string new_email = email_create(new_telephone, new_carrier);
using (StreamWriter sw = File.AppendText(@"C:\Dwight_Research\EMAILFILE.txt"))

                {

    string email_text = new_username + " " + new_email;
    sw.WriteLine(email_text);


                }

MessageBox.Show("Your Information has been Recorded", "Registration Complete",
MessageBoxButtons.OK, MessageBoxIcon.Information);


         textBox1.Text = "";
         textBox2.Text = "";
         textBox3.Text = "";

             }
         }//end class Form1


         }//end namespace
```

APPENDIX B

PSEE VOICE MATLAB CODE

CINCH REGISTER

```
function [status] = cinch_register (source)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

 if exist(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source))==0



        %Analyzes and Synthesizes WAV files to ensure that they can be used for
        %WFW Voice Conversion
         cinch_synth(source);
 else
        %Makes Directory for source speaker to save acoustic Parameters
        %as well as the corpus and vcfunction needed for the HSMnonpcorpus and
        %HSMnonptraing Functions which will be used for converting to the
        %ESPEAK Voice Files
        %REMOVE  LATER
        cinch_synth(source);
        mkdir(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/CONVERTED/'));
        save(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/CONVERTED/corpus'));
        save(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/CONVERTED/vcfunction'));
        save(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/CONVERTED/CONVSRC'));
         %Does the actual training of the source to speaker

HSMnonpcorpus(strcat('DELL/sourcespk/file'),1:5,strcat(source,'/sourcespk/file'),1:5,3,st
rcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/CONVERTED/corpus'));
        HSMnonptraining(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/CONVERTED/corpus'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/CONVERTED/vcfunction'),4);
status='Voice Saved';
```

```
 end
```

CINCH WFW

```
function [] = cinch_wfw (source)
%Takes the ESPEAK file and converts it to the desired target voice using
%the vcfunction that was created in cinc_register
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




        %Formats the wav file to be used for analysis
        [a b]=wavread(strcat('C:/Dwight_Research/',source,'TEMP_ORG.wav'));
        wavwrite(a,8000,'22khz.wav');
        [SRCFILE2 d]=wavread('22Khz.wav');
        SRCFILE2=ampreduce(SRCFILE2);
        SRCFILE2=interp(SRCFILE2,12);
        SRCFILE2=decimate(SRCFILE2,33);


        mkdir(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/'));
        wavwrite(SRCFILE2,8000,strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP.wav'));
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %Converts the new wav file to the desired target speaker
        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP.mat'));
       %Choose which Transformation you want to use
        %HSMgmmconvert(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/CONVERTED/vcfunction'),strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP_CONV.mat'));
        HSMwfwconvert(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/CONVERTED/vcfunction'),strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/TO_BE_CONV/TEMP_CONV.mat'));
```

```
                HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/TO_BE_CONV/TEMP_CONV.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/TO_BE_CONV/output.wav'));

                data=wavread(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/TO_BE_CONV/output.wav'));

                data=ampreduce(data);



                mkdir(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/FINAL_OUT/'));

                 DataPre=filter([1-.95],[1],data); DataPreInc=DataPre*40;

                 plot(DataPreInc);

        wavwrite(DataPreInc,strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/FINAL_OUT/output.wav'));

        beep_remove(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/FINAL_OUT/output'));



        CINCH SYNTH



        function [] = cinch_synth (source)



        if exist(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source))==0

            display('This source is not in our database. ')



        end

         mkdir(strcat('C:/Dwight_Research/UPC_HSM_VC 2/',source,'/sourcespk/'));

        %Converts the MP3s to WAV files for the

        cinch_conv(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourceMP3/fileorg001.mp3'), strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001org'));

        cinch_conv(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourceMP3/fileorg002.mp3'), strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002org'));

        cinch_conv(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourceMP3/fileorg003.mp3'), strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003org'));

        cinch_conv(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourceMP3/fileorg004.mp3'), strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004org'));
```

```
        cinch_conv(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourceMP3/fileorg005.mp3'), strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005org'));



        %Analyzes and Synthesizes WAV files to ensure that they can be used for

        %WFW Voice Conversion



        % espeak_trans(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001org.wav'));

        % espeak_trans(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002org.wav'));

        % espeak_trans(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003org.wav'));

        % espeak_trans(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004org.wav'));

        % espeak_trans(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005org.wav'));



        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001org.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001.mat'));

        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002org.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002.mat'));

        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003org.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003.mat'));

        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004org.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004.mat'));

        HSManalyze(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005org.wav'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005.mat'));



        HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file001.wav'));

        HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file002.wav'));

        HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file003.wav'));

        HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file004.wav'));
```

```
        HSMsynthesize(strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005.mat'),strcat('C:/Dwight_Research/UPC_HSM_VC
2/',source,'/sourcespk/file005.wav'));


        end
```

CINCH CONV

```
        function [] = cinch_conv (cinchmp3, cinchwav)


        %cinchMp3 is the cinch MP3 file that is being converted to the WAV

        %cinchMp3 is sampled at 22000


        %cinchwav is the location in which you want the wav file to be saved


        %Read in the MP3

        [data fs]=mp3read(cinchmp3);

        if(fs==22050)


        %Reduce the amplitude of the data so that it can be used in the HSM toolbox

        z=zeros(96000-length(data),1);

        data=vertcat(data,z);

        data=ampreduce(data);

        data=ampreduce(data);

        %Interpolate and Decimate to get the data to the 8000 Sample rate that is

        %used for the HSM Toolbox

        %22000*12=264,000 <--Interpolation

        %264000/33=8,000 <--Decimation


        data=interp(data,12);

        data=decimate(data,33);


        %Preemphasis filter to remove all "Pops" from the original WAVE FILE


         DataPre=filter([1-.99],[1],data); DataPreInc=DataPre*30;
```

```
        wavwrite(DataPreInc, strcat(cinchwav,'.wav'));
        else if (fs==11025)
                %Reduce the amplitude of the data so that it can be used in the HSM
toolbox
        z=zeros(96000-length(data),1);
        data=vertcat(data,z);
        data=ampreduce(data);
        data=ampreduce(data);
        data=ampreduce(data);
        %Interpolate and Decimate to get the data to the 8000 Sample rate that is
        %used for the HSM Toolbox
        %11000*8=88,000 <--Interpolation
        %88000/11=8,000 <--Decimation

        data=interp(data,8);
        data=decimate(data,11);

        %Preemphasis filter to remove all "Pops" from the original WAVE FILE

         DataPre=filter([1-.99],[1],data); DataPreInc=DataPre*30;
        wavwrite(DataPreInc, strcat(cinchwav,'.wav'));
            end;
        end;
```

AMPREDUCE

```
        function outputsignal = ampreduce(varargin)
        %ampreduce   Reduces the amplitude of a signal
        %
        %   For vectors, AMPREDUCE(X) is a vector with the same mean but
        %   smaller range than the original vector. The range is reduced by a factor
        %   defined by the user. If the user does not define a factor, the range is
reduced by a factor of 2
        %
```

```
%   Example: If X = [0 2 4]
%
%   then ampreduce(X) is [1 2 3]
%
%   See attached license for terms of use.


%Initial checks
if nargin == 1
    signal = varargin{1};
    reduction = 2;
elseif nargin == 2
    signal = varargin{1};
    reduction = varargin{2};
else
    error('Wrong number of inputs');
end

%Processing...
temp_signal = signal / reduction;
outputsignal = (mean(signal)) - (mean(temp_signal))  + temp_signal;


end
```

ZERO APPEND

```
function [data] = zero_append (data)
z=zeros(480000-length(data),1);
```

APPENDIX C

UPC TOOLBOX MATLAB CODE

HSM ANALYZE

```
function HSManalyze(fileIN,fileOUT)
%_____________________________________________________________________________
_
%
%    UPC Toolkit for HSM-based Voice Conversion
%    Copyright (C) 2007  TALP Research Center
%                        Universitat PolitËcnica de Catalunya (UPC)
%
%    This library is free software; you can redistribute it and/or
%    modify it under the terms of the GNU Lesser General Public
%    License as published by the Free Software Foundation; either
%    version 2.1 of the License, or (at your option) any later version.
%
%    This library is distributed in the hope that it will be useful,
%    but WITHOUT ANY WARRANTY; without even the implied warranty of
%    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
%    Lesser General Public License for more details.
%
%    You should have received a copy of the GNU Lesser General Public
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
%
%    Contact information:
%             AsunciÛn Moreno (asuncion.moreno@upc.edu)
%             TALP Research Center, Universitat PolitËcnica de Catalunya
%             UPC Campus Nord, Building D5
%             08034 - Barcelona, Spain
%_____________________________________________________________________________
_
```

```
%
%
%   HSManalyze(fileIn,fileOut)
%
%   fileIn (string): wav file to be analyzed. Only wav files sampled at
%   16KHz 16bits mono, are accepted.
%   fileOut (string): matlab file containing the HSM parameters of the
%   input signal.
%
%   If the input arguments are not specified, the program will ask for
%   them.
%



% The HSM parameters are stored in a structure called 'picos', whose fields are:
% pm: analysis sample (where the current parameters are measured)
% f0: pitch estimation at pm
% a: amplitudes of the harmonics at f0, 2f0, ... <fmax
% p: phases of the harmonics
% alfa: linear phase term of the harmonics
% e: denominator of the LPC filter (including gain)


% general settings
N=128; % analysis rate: 8ms at 16KHz sampling freq = 128 samples
f0min=60.0; f0max=500.0; % pitch range
fmax=5000.0; % maximum voiced frequency (fixed value)
ordenLPC=14; % order of the filters representing the stochastic component
% go
if nargin==0 [fileIN,pathIN]=uigetfile('*.wav','Select WAV file'); if
(isequal(fileIN,0)|isequal(pathIN,0)) return; end; fileIN=[pathIN fileIN]; end;
if length(fileIN)>4 if fileIN(length(fileIN)-3:length(fileIN))=='.wav'
fileIN=fileIN(1:length(fileIN)-4); end; end;
[x,fs]=wavread(fileIN); x=transpose(x); L=length(x);
if size(x,1)>1 x=x(1,:); disp('   Only the first channel will be processed.');
end;
```

```
        if fs~=8000 disp('   This tool operates only with sounds sampled at 16khz!!');
return; end;

        fmax=min(fmax,0.5*fs);

        pms=1+ceil(1.5*fs/f0min)+N*(0:floor((L-3.0*fs/f0min)/N));

        f0s=f0analysisbyboersma(x,fs,pms,f0min,f0max); % f0 analysis

        picos=harmonicanalysis(x,fs,pms,f0s,fmax); % amplitudes and phases

        picos=stochasticanalysis(x,fs,N,picos,ordenLPC); % noise lpc filter

        picos=polarityanalysis(picos); % waveform peaks are made possitive (necessary for
phase decomposition)

        picos=decomposephase(picos); % decomposition into a linear term and a "filter"
term

        if nargin<2 [fileOUT,pathOUT]=uiputfile('*.mat','Save MAT file',fileIN); if
(isequal(fileOUT,0)|isequal(pathOUT,0)) return; end; fileOUT=[pathOUT fileOUT]; end;

        save(fileOUT,'L','fs','picos');



        % harmonicanalysis

        function picos=harmonicanalysis(x,fs,pms,f0s,fmax)

        picos(1:length(pms))=struct('pm',0,'f0',0.0,'a',[],'alfa',0.0,'p',[],'e',[]);

        for k=1:length(pms)

            picos(k).pm=pms(k); picos(k).f0=f0s(k);

            if f0s(k)>0.0

                Lw=ceil(2.2*fs/min(f0s(k),150.0));

                Lw2=floor(Lw/2); Lw=2*Lw2+1;

                trama2T=x(pms(k)+(-Lw2:Lw2));

                win=transpose(0.54+0.46*cos(pi*(-Lw2:Lw2)/Lw2));

                trama2T=transpose(trama2T).*win;

                K=ceil(fmax/f0s(k))-1;

                h=zeros(Lw,K); h(:,1)=transpose(exp(i*2*pi*f0s(k)*(-Lw2:Lw2)/fs)); for
kk=2:K h(:,kk)=h(:,kk-1).*h(:,1); end; for kk=1:K h(:,kk)=h(:,kk).*win; end; h=[h
conj(h)];

                if 1 coef=(h'*h)\(h'*trama2T); else coef=inv(h'*h)*h'*trama2T; end;

                picos(k).a=transpose(2.0*abs(coef(1:K)));

                       picos(k).p=transpose(angle(coef(1:K)));

            end;

        end;
```

```
% stochanal

function picos=stochasticanalysis(x,fs,N,picos,ordenLPC)

Npm=length(picos);

yd=zeros([1,length(x)]);

for j=1:length(picos(1).a) yd(picos(1).pm-N:picos(1).pm-1)=yd(picos(1).pm-
N:picos(1).pm-1)+(picos(1).a(j)*(0:N-1)/N).*cos(2*pi*j*picos(1).f0*(-N:-
1)/fs+picos(1).p(j)); end;

for k=1:Npm-1

    for j=1:min(length(picos(k).a),length(picos(k+1).a))

        a1=picos(k).a(j); a2=picos(k+1).a(j);


[c0,c1,c2,c3]=bymcaquat(picos(k).p(j),picos(k+1).p(j),j*picos(k).f0,j*picos(k+1).f0,N,fs)
;

        yd(picos(k).pm:picos(k).pm+N-1)=yd(picos(k).pm:picos(k).pm+N-1)+(a1+(a2-
a1)*(0:N-1)/N).*cos(c0+(0:N-1).*(c1+(0:N-1).*(c2+c3*(0:N-1))));

    end;

    for j=length(picos(k+1).a)+1:length(picos(k).a) yd(picos(k).pm:picos(k).pm+N-
1)=yd(picos(k).pm:picos(k).pm+N-1)+(picos(k).a(j)*(N:-
1:1)/N).*cos(2*pi*j*picos(k).f0*(0:N-1)/fs+picos(k).p(j)); end;

    for j=length(picos(k).a)+1:length(picos(k+1).a) yd(picos(k+1).pm-
N:picos(k+1).pm-1)=yd(picos(k+1).pm-N:picos(k+1).pm-1)+(picos(k+1).a(j)*(0:N-
1)/N).*cos(2*pi*j*picos(k+1).f0*(-N:-1)/fs+picos(k+1).p(j)); end;

end;

for j=1:length(picos(Npm).a) yd(picos(Npm).pm:picos(Npm).pm+N-
1)=yd(picos(Npm).pm:picos(Npm).pm+N-1)+(picos(Npm).a(j)*(N:-
1:1)/N).*cos(2*pi*j*picos(Npm).f0*(0:N-1)/fs+picos(Npm).p(j)); end;

ye=x-yd; ye=filter(fir1(48,500/(0.5*fs),'high'),1,ye); ye=[ye(25:length(ye))
zeros(1,24)];

hnnN=sqrt(8/3)*(0.5-0.5*cos(2*pi*(0:N-1)/N));

for k=1:Npm trama=ye(picos(k).pm-floor(N/2):picos(k).pm-floor(N/2)+N-1).*hnnN; if
trama==zeros(size(trama)) picos(k).e=[inf zeros(1,ordenLPC)]; else
R=zeros([1,ordenLPC+1]); for j=0:ordenLPC R(j+1)=sum(trama(j+1:N).*trama(1:N-j)); end;
picos(k).e=sqrt(N)*bylevdurb(R); end; end;



% bymcaquat (mcaulay & quatieri)

function [c0,c1,c2,c3]=bymcaquat(p1,p2,f1,f2,N,fs)

w1=2*pi*f1/fs; w2=2*pi*f2/fs;

M=round((1/(2*pi))*((p1+w1*N-p2)+0.5*N*(w2-w1)));

c0=p1; c1=w1; c2=(3/(N*N))*(p2-p1-w1*N+2*pi*M)-(w2-w1)/N; c3=(-2/(N*N*N))*(p2-p1-
w1*N+2*pi*M)+(w2-w1)/(N*N);



% bylevdurb (levinson-durbin)
```

```
        function ai=bylevdurb(R)

        ordenLPC=length(R)-1; ai=zeros([1,ordenLPC]); e=R(1);

        for u=1:ordenLPC if (u==1) k=R(2)/R(1); ai(1)=k; e=(1-k*k)*e; else k=R(u+1); for
v=1:u-1 k=k-ai(v)*R(u-v+1); end; k=k/e; ai(u)=k; for v=1:u-1 aux(v)=ai(v)-k*ai(u-v); end;
ai(1:u-1)=aux(1:u-1); e=(1-k*k)*e; end; end;

        ai=[1 -ai]/sqrt(abs(R(1)-sum(ai.*R(2:ordenLPC+1))));



        % polarityanalysis

        function [picos,pol]=polarityanalysis(picos)

        pol=0; polE=0.0;

        for k=1:length(picos)

            if picos(k).f0>0.0

                E=sum(picos(k).a.*picos(k).a);

                alfa=2.0*picos(k).p(1)-picos(k).p(2); alfa=alfa-2*pi*floor(alfa/(2*pi));

                dP=min(abs([alfa 2*pi-alfa])); dN=abs(alfa-pi);

                if dN<dP pol=pol-1; polE=polE-E; elseif dP<dN pol=pol+1; polE=polE+E; end;

            end;

        end;

        if pol<0 pol=-1; elseif pol>0 pol=1; elseif polE<0.0 pol=-1; else pol=1; end;

        if pol==-1 for k=1:length(picos) if picos(k).f0>0.0 picos(k).p=picos(k).p+pi; end;
end; end;



        % f0analysisbyboersma (boersma)

        function f0s=f0analysisbyboersma(x,fs,pms,f0min,f0max)

        Ncandidates=6;

        if nargin<5 f0max=500.0; if nargin<4 f0min=60.0; end; end;

        lagmin=ceil(fs/f0max); lagmax=floor(fs/f0min); fact=0.01*fs/(pms(2)-pms(1));

        voith=0.45; silth=0.03; octcost=0.01; vuvcost=0.14*fact; uvvcost=0.14*fact;
uvuvcost=0.0; octjump=0.35*fact;

        L=ceil(3.0*fs/f0min); if rem(L,2)==0 L2=L/2; L=L+1; else L2=(L-1)/2; end;

        Lz=ceil(1.5*L); Lp2=2; while Lp2<Lz Lp2=Lp2*2; end;

        Lxlmax=ceil(0.5*fs/f0min); Ldc=2*Lxlmax;

        w=zeros(1,Lp2); w(1:L)=0.5*(1-cos(2*pi*(0:(L-1))/L));

        rw=real(ifft(abs(fft(w)).^2)); rw=(1/rw(1))*rw(1:L2);

        xgmax=max(abs(x));
```

```
        dat(1:length(pms))=struct('r',[],'f',[],'ac',[],'ant',[]);

        for k=1:length(pms)

            if pms(k)-L2<1 trama=x(1:(pms(k)+L2)); trama=[zeros(1,L-length(trama)) trama];

            elseif pms(k)+L2>length(x) trama=x((pms(k)-L2):length(x)); trama=[trama
zeros(1,L-length(trama))];

            else trama=x(pms(k)+(-L2:L2)); end;

            trama=trama-sum(trama(L2+1+(-Ldc:Ldc)))/(2.0*Ldc+1.0);

            xlmax=max(abs(trama(L2+1+(-Lxlmax:Lxlmax))));

            trama=[trama zeros(1,Lp2-L)].*w;

            ra=real(ifft(abs(fft(trama)).^2)); ra=(1/ra(1))*ra(1:L2);

            rx=ra./rw; rx=max(rx,0.0);

            rk=zeros(1,Ncandidates-1); fk=zeros(1,Ncandidates-1);

            for j=lagmin+1:lagmax+1

                if rx(j)>0.5*voith && rx(j-1)<rx(j) && rx(j+1)<rx(j)

                    tmax=0.5*(rx(j-1)-rx(j+1))/(rx(j-1)-2.0*rx(j)+rx(j+1));

                    rmax=rx(j)+0.5*tmax*(rx(j+1)-rx(j-1)+tmax*(rx(j-1)-2*rx(j)+rx(j+1)));

                    if rmax>1.0 rmax=1.0/rmax; end;

                    tmax=(tmax+j-1)/fs; fmax=1.0/tmax; rmax=rmax-octcost*log2(f0min*tmax);

                    [rmin,jj]=min(rk); if rmax>rmin rk(jj)=rmax; fk(jj)=fmax; end;

                end;

            end;

            ruv=voith+max(0.0,2.0-min(1.0,xlmax/xgmax)/(silth/(1.0+voith)));

            jj=find(fk>0.0); dat(k).r=[rk(jj)+octcost*log2(f0min./fk(jj)) ruv];
dat(k).f=[fk(jj) 0.0];

            dat(k).ac=zeros(size(dat(k).f)); dat(k).ant=zeros(size(dat(k).f));

        end;

        dat(1).ac=-dat(1).r;

        for k=2:length(pms)

            for j=1:length(dat(k).f)

                mincost=inf; jjmincost=-1;

                for jj=1:length(dat(k-1).f)

                    cost=dat(k-1).ac(jj)-dat(k).r(j);

                    if dat(k).f(j)==0.0 && dat(k-1).f(jj)==0.0 cost=cost+uvuvcost;

                    elseif dat(k).f(j)==0.0 cost=cost+vuvcost;

                    elseif dat(k-1).f(jj)==0.0 cost=cost+uvvcost;
```

```
                else cost=cost+octjump*abs(log2(dat(k).f(j)/dat(k-1).f(jj))); end;

                if cost<mincost mincost=cost; jjmincost=jj; end;

            end;

            dat(k).ac(j)=mincost; dat(k).ant(j)=jjmincost;

        end;

    end;

    f0s=zeros(size(pms));

    [mincost,jjmincost]=min(dat(length(dat)).ac);

    f0s(length(f0s))=dat(length(dat)).f(jjmincost);
jjmincost=dat(length(dat)).ant(jjmincost);

    for k=length(pms)-1:-1:1 f0s(k)=dat(k).f(jjmincost);
jjmincost=dat(k).ant(jjmincost); end;



    % decomposephase

    function picos=decomposephase(picos)

    fmaxopt=1000.0;

    for k=1:length(picos)

        if picos(k).f0>0.0

            alfa=linphaseterm(picos(k).a,picos(k).p,fmaxopt);

            picos(k).alfa=-alfa; picos(k).p=picos(k).p+(1:length(picos(k).p))*alfa;

        end;

    end;



    % linphaseterm

    function alfa=linphaseterm(aa,pp,fmaxopt)

    %poffset=-pp(1)+pi; pp=pp+(1:length(pp))*poffset;

    fmax=5000.0; if nargin<3 fmaxopt=1000.0; end;

    K=ceil(length(aa)*fmaxopt/fmax); aa_=aa(1:K); pp_=pp(1:K);

    alfa=sumcos((1:K).*aa_,pp_,'sin'); % x=Sumk{Ak∑cos(k∑alfa+pk)} ,, dx/dalfa=-
Sumk{k∑Ak∑sin(k∑alfa+pk)}=0

    if length(alfa)>1 s=zeros(1,length(alfa)); for k=1:length(alfa)
s(k)=sum(aa_.*cos((1:length(pp_))*alfa(k)+pp_)); end; [smax,imax]=max(s);
alfa=alfa(imax);

    else disp(num2str([aa_.' pp_.'])); error('No hay alfa!!'); end;

    %alfa=alfa+poffset;
```

```
% sumcos

function x=sumcos(aa,pp,modo)

K=length(aa); AcP=aa.*cos(pp); AsP=aa.*sin(pp);

[T,U]=polytseb(K);

if isequal(modo,'cos') Tx=AcP*T; Ux=-AsP*U; elseif isequal(modo,'sin') Ux=AcP*U;
Tx=AsP*T; end;

Px=conv([-1.0 0.0 1.0],conv(Ux,Ux))-[0.0 0.0 conv(Tx,Tx)];

x=roots(Px).'; x=x(find(imag(x)==0.0 & abs(x)<=1.0));

for k=1:length(x) term1=sqrt(1-x(k)*x(k))*polyval(Ux,x(k));
term2=polyval(Tx,x(k)); if abs(term2+term1)<abs(term2-term1) x(k)=acos(x(k)); else x(k)=-
acos(x(k)); end; end;



% polytseb

function [T,U]=polytseb(K)

T=zeros(K+1,K+1); T(1,K+1)=1; T(2,K)=1; U=zeros(K+1,K+1); U(1,K+1)=1; U(2,K)=2;

for k=2:K T(k+1,:)=2*[T(k,2:K+1) 0]-T(k-1,:); U(k+1,:)=[U(k,2:K+1) 0]+T(k+1,:);
end;

T=T(2:K+1,:); U=U(1:K,:);
```

HSM SYNTHESIZE

```
 function HSMsynthesize(fileIN,fileOUT)

%_____________________________________________________________________________
_

%

%    UPC Toolkit for HSM-based Voice Conversion

%    Copyright (C) 2007  TALP Research Center

%                        Universitat PolitËcnica de Catalunya (UPC)

%

%    This library is free software; you can redistribute it and/or

%    modify it under the terms of the GNU Lesser General Public

%    License as published by the Free Software Foundation; either

%    version 2.1 of the License, or (at your option) any later version.

%

%    This library is distributed in the hope that it will be useful,

%    but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
%    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
%    Lesser General Public License for more details.
%
%    You should have received a copy of the GNU Lesser General Public
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
%
%    Contact information:
%             AsunciÛn Moreno (asuncion.moreno@upc.edu)
%             TALP Research Center, Universitat PolitËcnica de Catalunya
%             UPC Campus Nord, Building D5
%             08034 - Barcelona, Spain
%_____________________________________________________________________________
_
%
%
%   HSMsynthesize(fileIn,fileOut)
%
%   fileIn (string): input matlab file to be transformed into sound.
%   fileOut (string): output wav file (16KHz 16bits mono).
%


       if nargin==0 [fileIN,pathIN]=uigetfile('*.mat','Select MAT file'); if
(isequal(fileIN,0)|isequal(pathIN,0)) return; end; fileIN=[pathIN fileIN]; end;

       if length(fileIN)>4 && isequal(fileIN(length(fileIN)-3:length(fileIN)),'.mat')
fileIN=fileIN(1:length(fileIN)-4); end;

       load(fileIN);

       y=synth(L,fs,picos);

       if nargin<2 [fileOUT,pathOUT]=uiputfile('*.wav','Save WAV file',fileIN); if
(isequal(fileOUT,0)|isequal(pathOUT,0)) return; end; fileOUT=[pathOUT fileOUT]; end;

       wavwrite(y,fs,fileOUT);


       % synth

       function y=synth(L,fs,picos)
```

```
        y=zeros([1,L]); Npm=length(picos); ordenLPC=length(picos(1).e)-1;

        for k=1:Npm

            n2=picos(k).pm; if k==1 n3=picos(k+1).pm; n1=max(1,n2-(n3-n2)); elseif k==Npm
n1=picos(k-1).pm; n3=min(L,n2+(n2-n1)); else n1=picos(k-1).pm; n3=picos(k+1).pm; end;

            N12=n2-n1; N23=n3-n2;

            win=[0:1/N12:(N12-1)/N12 1:-1/N23:1/N23];

            trama=zeros(1,n3-n1);

            for j=1:length(picos(k).a) trama=trama+(picos(k).a(j))*cos(2*pi*(-N12:N23-
1)*(j*picos(k).f0)/fs+picos(k).p(j)+j*picos(k).alfa); end;

            trama=trama+filter(1,picos(k).e,randn(1,n3-n1));

            y(n1:n3-1)=y(n1:n3-1)+trama.*win;

        end;

        data=vertcat(data,z);




        HSM Non-Paralell Corpus


        function HSMnonpcorpus(raiz1,inds1,raiz2,inds2,nc,fo)

        %_____________________________________________________________________________
_
        %
        %    UPC Toolkit for HSM-based Voice Conversion
        %    Copyright (C) 2007  TALP Research Center
        %                        Universitat PolitËcnica de Catalunya (UPC)
        %
        %    This library is free software; you can redistribute it and/or
        %    modify it under the terms of the GNU Lesser General Public
        %    License as published by the Free Software Foundation; either
        %    version 2.1 of the License, or (at your option) any later version.
        %
        %    This library is distributed in the hope that it will be useful,
        %    but WITHOUT ANY WARRANTY; without even the implied warranty of
        %    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
        %    Lesser General Public License for more details.
```

```
%
%    You should have received a copy of the GNU Lesser General Public
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
%
%    Contact information:
%             AsunciÛn Moreno (asuncion.moreno@upc.edu)
%             TALP Research Center, Universitat PolitËcnica de Catalunya
%             UPC Campus Nord, Building D5
%             08034 - Barcelona, Spain
%_____________________________________________________________________________
_
%
%
%   HSMnonpcorpus(prefix1,ind1,prefix2,ind2,numdigits,fout)
%
%   Example of usage:
%
%   HSMpcorpus('srcfile',1:2,'tgtfile',6:7,4,'corpus')
%
%   The output training corpus, called 'corpus', is created from the
%   following matlab files: {srcfile0001, srcfile0002}, {tgtfile0006,
%   tgtfile0007}.
%


p1v=[]; p2v=[];
for nf=1:length(inds1)
    n=int2str(inds1(nf)); for k=length(n)+1:nc n=['0' n]; end;
    load([raiz1 n]);
    f0s=zeros(1,length(picos)); for k=1:length(picos) f0s(k)=picos(k).f0; end;
    p1v=[p1v picos(find(f0s>0.0))];
    clear picos fs L f0s;
end;
```

```
for nf=1:length(inds2)
    n=int2str(inds2(nf)); for k=length(n)+1:nc n=['0' n]; end;
    load([raiz2 n]);
    f0s=zeros(1,length(picos)); for k=1:length(picos) f0s(k)=picos(k).f0; end;
    p2v=[p2v picos(find(f0s>0.0))];
    clear picos fs L f0s;
end;
save(fo,'p1v','p2v','inds1','inds2');
```

HSM Non-Parallel Training

```
function HSMnonptraining(corpus,fout,m)
%_____________________________________________________________________________
_
%
%    UPC Toolkit for HSM-based Voice Conversion
%    Copyright (C) 2007  TALP Research Center
%                        Universitat PolitËcnica de Catalunya (UPC)
%
%    This library is free software; you can redistribute it and/or
%    modify it under the terms of the GNU Lesser General Public
%    License as published by the Free Software Foundation; either
%    version 2.1 of the License, or (at your option) any later version.
%
%    This library is distributed in the hope that it will be useful,
%    but WITHOUT ANY WARRANTY; without even the implied warranty of
%    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
%    Lesser General Public License for more details.
%
%    You should have received a copy of the GNU Lesser General Public
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
```

```
        %
        %    Contact information:
        %            AsunciÛn Moreno (asuncion.moreno@upc.edu)
        %            TALP Research Center, Universitat PolitËcnica de Catalunya
        %            UPC Campus Nord, Building D5
        %            08034 - Barcelona, Spain
        %_____________________________________________________________________________
_
        %
        %
        %   HSMnonptraining(corpus,fout,[m])
        %
        %   corpus (string): input non-parallel corpus.
        %   fout (string): matlab file containing the trained voice conversion
        %   function.
        %   m (integer): number of gaussian components of the function (def=8).
        %


        p=14; % order of the LSF vectors

        if nargin<3 || length(m)==0 m=1; m_=8; elseif length(m)==1 m_=m; m=1; else
m_=m(2); m=m(1); end;

        if nargin<1 [corpus,paux]=uigetfile('*.mat','Select training corpus'); if
(isequal(corpus,0)|isequal(paux,0)) return; end; corpus=[paux corpus]; end; if
corpus(length(corpus)-3)=='.' corpus=corpus(1:length(corpus)-4); end; load(corpus);

        Nx=length(p1v); Ny=length(p2v);

        if 1

            lf01=zeros(1,Nx); for k=1:Nx lf01(k)=log10(p1v(k).f0); end;
lf01u=(1/Nx)*sum(lf01); lf01v=sqrt((1/Nx)*sum((lf01-lf01u).^2));

            lf02=zeros(1,Ny); for k=1:Ny lf02(k)=log10(p2v(k).f0); end;
lf02u=(1/Ny)*sum(lf02); lf02v=sqrt((1/Ny)*sum((lf02-lf02u).^2));

            f0f12=[lf01u lf01v lf02u lf02v]; f0f21=[lf02u lf02v lf01u lf01v];

        else f0f12=[]; f0f21=[]; end;

        if 1

            [X_,X]=detalsf(p1v,p);

            [Y_,Y]=detalsf(p2v,p);

            Xe=stochalsf(p1v);

            Ye=stochalsf(p2v);
```

```
        end;

        clear p1v p2v; pack;

        if 1

            Xt=X; sigue=1; iter1=0; itermax1=25; itermax2=25; dtotant=inf; parejaxy=[];
parejayx=[]; ind=[];

            while sigue>0

                iter1=iter1+1; dtot=0;

                parejaxyant=parejaxy; parejayxant=parejayx; indant=ind;

                parejaxy=zeros(1,Nx); for k=1:Nx [parejaxy(k),dpar]=nnfind(Y,Xt(:,k));
dtot=dtot+dpar; end;

                parejayx=zeros(1,Ny); for k=1:Ny [parejayx(k),dpar]=nnfind(Xt,Y(:,k));
dtot=dtot+dpar; end;

                for k=1:length(parejayx) if parejaxy(parejayx(k))==k parejayx(k)=0; end;
end; ind=find(parejayx>0);

                if dtot>=dtotant || iter1>=itermax1 sigue=0; X=X_; Y=Y_; m=m_;
parejaxy=parejaxyant; parejayx=parejayxant; ind=indant; itermax2=inf; clear X_ Y_; pack;
else dtotant=dtot; end;

                Z=[X X(:,parejayx(ind)); Y(:,parejaxy) Y(:,ind)]; N=size(Z,2);

                ugral=(1/N)*sum(Z,2); Egral=zeros(2*p,2*p); for t=1:N v=Z(:,t)-ugral;
Egral=Egral+v*transpose(v); end; Egral=(1/N)*Egral; umb=zeros(1,2*p); for j=1:2*p
umb(j)=0.02*Egral(j,j); end; pert=min(umb); clear ugral Egral umb;

                if m==1

                    ui=(1/N)*sum(Z,2);

                    Ei=zeros(2*p,2*p); for t=1:N Ei=Ei+(Z(:,t)-ui)*transpose(Z(:,t)-ui);
end; Ei=(1/N)*Ei+pert*eye(2*p);

                    thz=struct('a',1.0,'u',ui,'E',Ei);

                else

                    [c,c0]=clustkmeans(Z,m);

                    thz(1:m)=struct('a',0,'u',zeros(2*p,1),'E',zeros(2*p,2*p)); for i=1:m
indi=find(c==i); subZ=Z(:,indi); subN=length(indi); thz(i).a=subN/N; ui=Z(:,c0(i));
thz(i).u=ui; Ei=zeros([2*p,2*p]); for k=1:subN v=subZ(:,k)-ui; Ei=Ei+v*transpose(v); end;
thz(i).E=(1/subN)*Ei+pert*eye(2*p); end;

                    P=zeros(m,N); seguir=1; umb=0.000001; cte=power(2*pi,-p); iter2=1;

                    pxt=zeros(1,N); for i=1:m ai=thz(i).a; ui=thz(i).u; Ei=thz(i).E;
dEi=1/sqrt(det(Ei)); iEi=inv(Ei); for t=1:N P(i,t)=ai*dEi*exp(-0.5*transpose(Z(:,t)-
ui)*iEi*(Z(:,t)-ui)); pxt(t)=pxt(t)+cte*P(i,t); end; end; Lant=sum(log(pxt));
Psum=sum(P,1); for i=1:m P(i,:)=P(i,:)./Psum; end;

                    while seguir==1

                        for i=1:m thz(i).a=(1/N)*sum(P(i,:)); end; % Calcular las nuevas
alfa

                        for i=1:m ui=zeros(2*p,1); for t=1:N ui=ui+P(i,t)*Z(:,t); end;
thz(i).u=ui*(1/(N*thz(i).a)); end; % Calcular las nuevas medias
```

```
                        for i=1:m Ei=zeros(2*p,2*p); for t=1:N Ei=Ei+P(i,t)*(Z(:,t)-
thz(i).u)*transpose(Z(:,t)-thz(i).u); end; thz(i).E=Ei*(1/(N*thz(i).a))+pert*eye(2*p);
end; % Calcular las nuevas covarianzas

                        pxt=zeros(1,N); for i=1:m ai=thz(i).a; ui=thz(i).u; Ei=thz(i).E;
dEi=1/sqrt(det(Ei)); iEi=inv(Ei); for t=1:N P(i,t)=ai*dEi*exp(-0.5*transpose(Z(:,t)-
ui)*iEi*(Z(:,t)-ui)); pxt(t)=pxt(t)+cte*P(i,t); end; end; L=sum(log(pxt)); Psum=sum(P,1);
for i=1:m P(i,:)=P(i,:)./Psum; end; % Calcular la nueva likelihood y las siguientes P's

                        if iter2==itermax2 seguir=0; else iter2=iter2+1; end;

                        if (L-Lant)/Lant<umb seguir=0; else Lant=L; end;

                    end;

                end;

                thxy(1:m)=struct('a',0,'u',[],'E',[],'v',[],'R',[]); thyx=thxy;

                for k=1:m thxy(k).a=thz(k).a; thxy(k).u=thz(k).u(1:p,1);
thxy(k).v=thz(k).u(p+1:2*p,1); thxy(k).E=thz(k).E(1:p,1:p);
thxy(k).R=thz(k).E(p+1:2*p,1:p); thyx(k).a=thz(k).a; thyx(k).u=thz(k).u(p+1:2*p,1);
thyx(k).v=thz(k).u(1:p,1); thyx(k).E=thz(k).E(p+1:2*p,p+1:2*p);
thyx(k).R=thz(k).E(1:p,p+1:2*p); end; clear thz;

                clear P; pack;

                if sigue>0

                    thaux(1:m)=struct('d',0,'I',zeros(p,p)); for j=1:m
thaux(j).d=det(thxy(j).E); thaux(j).I=inv(thxy(j).E); end;

                    Xt=X; for k=1:Nx v=Xt(:,k); subP=zeros(1,m); for j=1:m
subP(j)=thxy(j).a*(1/sqrt(thaux(j).d))*exp(-0.5*transpose(v-thxy(j).u)*thaux(j).I*(v-
thxy(j).u)); end; Psum=sum(subP); subP=(1/Psum)*subP; vt=zeros(size(v)); for j=1:m
vt=vt+subP(j)*(thxy(j).v+thxy(j).R*thaux(j).I*(v-thxy(j).u)); end; Xt(:,k)=vt; end;

                end;

            end;

        end;

        if 1

            thxe=thxy; thye=thyx;

            P=zeros(Nx,m); D=zeros(Nx,m*p);

            for i=1:m ai=thxe(i).a; ui=thxe(i).u; Ei=thxe(i).E; dEi=1/sqrt(det(Ei));
iEi=inv(Ei); iEiT=transpose(iEi); for t=1:Nx P(t,i)=ai*dEi*exp(-0.5*transpose(X(:,t)-
ui)*iEi*(X(:,t)-ui)); D(t,(i-1)*p+(1:p))=P(t,i)*transpose(X(:,t)-ui)*iEiT; end; end; for
t=1:Nx iPsum=1/sum(P(t,:)); P(t,:)=iPsum*P(t,:); D(t,:)=iPsum*D(t,:); end;

            Z=(pinv([P D])*Xe.').'; for i=1:m thxe(i).v=Z(:,i); thxe(i).R=Z(:,m+(i-
1)*p+(1:p)); end;

            P=zeros(Ny,m); D=zeros(Ny,m*p);

            for i=1:m ai=thye(i).a; ui=thye(i).u; Ei=thye(i).E; dEi=1/sqrt(det(Ei));
iEi=inv(Ei); iEiT=transpose(iEi); for t=1:Ny P(t,i)=ai*dEi*exp(-0.5*transpose(Y(:,t)-
ui)*iEi*(Y(:,t)-ui)); D(t,(i-1)*p+(1:p))=P(t,i)*transpose(Y(:,t)-ui)*iEiT; end; end; for
t=1:Ny iPsum=1/sum(P(t,:)); P(t,:)=iPsum*P(t,:); D(t,:)=iPsum*D(t,:); end;

            Z=(pinv([P D])*Ye.').'; for i=1:m thye(i).v=Z(:,i); thye(i).R=Z(:,m+(i-
1)*p+(1:p)); end;
```

```
            clear P D Z Xe Ye; pack;

        end;

        if 1

            [fwxy,fwyx]=automaticth2wfw(thxy);

        end;

        clear X Y; pack;

        if nargin<2 [fout,paux]=uiputfile('*.mat','Save transformation function',['vc-'
corpus]); if (isequal(fout,0)|isequal(paux,0)) return; end; fout=[paux fout]; end;

        th=thxy; th2=thye; fw=fwxy; f0f=f0f12; save(fout,'th','th2','fw','f0f');

        th=thyx; th2=thxe; fw=fwyx; f0f=f0f21; save([fout '-inv'],'th','th2','fw','f0f');



        % clustkmeans

        function [c,c0]=clustkmeans(X,n)

        N=size(X,2); c0=(1:n)*floor(N/n); c=zeros(1,N); Nj=zeros(1,n);

        for k=1:N dmin=Inf; jmin=0; for j=1:n dact=sum((X(:,k)-X(:,c0(j))).^2); if
dact<dmin dmin=dact; jmin=j; end; end; c(k)=jmin; Nj(jmin)=Nj(jmin)+1; end;

        copiac=c; seguir=1; iter=0; maxiters=1000000;

        aux(1:n)=struct('act',0,'ind',[],'sumd',[]);

        while seguir==1

            for k=1:n aux(k).act=1; aux(k).ind=zeros(1,Nj(k)); aux(k).sumd=zeros(1,Nj(k));
end;

            for k=1:N cls=c(k); act=aux(cls).act; aux(cls).ind(act)=k; for j=1:act-1
dact=sum((X(:,k)-X(:,aux(cls).ind(j))).^2); aux(cls).sumd(j)=aux(cls).sumd(j)+dact;
aux(cls).sumd(act)=aux(cls).sumd(act)+dact; end; end;

            for k=1:n dmin=Inf; for j=1:Nj(k) if aux(k).sumd(j)<dmin dmin=aux(k).sumd(j);
c0(k)=aux(k).ind(j); end; end; end;

            Nj=zeros(1,n); for k=1:N dmin=Inf; jmin=0; for j=1:n dact=sum((X(:,k)-
X(:,c0(j))).^2); if dact<dmin dmin=dact; jmin=j; end; end; c(k)=jmin;
Nj(jmin)=Nj(jmin)+1; end;

            if c==copiac seguir=0; else copiac=c; end; iter=iter+1; if iter>=maxiters
seguir=0; end;

        end;



        % detalsf

        function [LSF,CC]=detalsf(picos,p)

        fmax=5000; fs=2*fmax; if nargin<2 p=14; end;

        LSF=zeros(p,length(picos));

        if nargout>1 CC=zeros(p,length(picos)); end;
```

```
        for k=1:length(picos)

            Nk=length(picos(k).a); if Nk==0; continue; end;

            aa=picos(k).a; ff=(1:Nk)*picos(k).f0; PP=aa.*aa; lim=0.0001;

            R=zeros(1,p+1); for j=1:p+1 R(j)=(1/Nk)*sum(PP.*cos(2*pi*(j-1)*ff/fs)); end;

            ai=zeros(1,p); e=R(1); for j=1:p if j==1 K=R(2)/R(1); ai(1)=K; e=(1-K*K)*e;
else K=R(j+1); for u=1:j-1 K=K-ai(u)*R(j-u+1); end; K=K/e; ai(j)=K; for u=1:j-1
aux(u)=ai(u)-K*ai(j-u); end; ai(1:j-1)=aux(1:j-1); e=(1-K*K)*e; end; end;

            ai=[1 -ai]/sqrt(abs(R(1)-sum(ai.*R(2:p+1))));

            %mat(p+1,p+1)=0; for j1=1:p+1 for j2=1:p+1 mat(j1,j2)=R(abs(j1-j2)+1); end;
end; mat=inv(mat);

            %emiw=exp(-i*2*pi*ff/fs); emiwj=ones(p+1,length(ff)); for j=2:p+1
emiwj(j,:)=emiwj(j-1,:).*emiw; end;

            %denom(Nk)=0; h(p+1)=0; eis0=Inf; lap=0; lap2=0; while 1 denom=ai*emiwj;
e=PP.*(denom.*conj(denom)); eis=(1/Nk)*sum(e-log(e)-1); if (eis0>eis) if (eis0-
eis<lim)|(lap==20) break; end; eis0=eis; lap=lap+1; end; lap2=lap2+1; if lap2>50 break;
end; for j=1:p+1 h(j)=(1/Nk)*real(sum(emiwj(j,:)./denom)); end; ai=(mat*h')'; end;
ai=ai/sqrt((1/Nk)*sum(e));

            ai=(1/ai(1))*ai;

            % cepstrales

            if nargout>1 CC(1,k)=-ai(2); for j=2:p CC(j,k)=-ai(j+1)-sum((1-(1:j-
1)/j).*ai(2:j).*CC(j-1:-1:1,k).'); end; end;

            % conversion a LSF

            az1=[ai 0]; az2=az1(length(az1):-1:1); lsf=angle([roots(az1+az2); roots(az1-
az2)]); for j=1:length(lsf)-1 vmin=Inf; imin=-1; for t=j:length(lsf) if lsf(t)<vmin
vmin=lsf(t); imin=t; end; end; aux=lsf(j); lsf(j)=lsf(imin); lsf(imin)=aux; end;

            LSF(:,k)=lsf(length(lsf)-p:length(lsf)-1);

        end;



        % stochalsf

        function LSF=stochalsf(picos)

        p=length(picos(1).e)-1;

        LSF=zeros(p,length(picos));

        for k=1:length(picos)

            ai=picos(k).e; ai=(1/ai(1))*ai; az1=[ai 0]; az2=az1(length(az1):-1:1);
lsf=angle([roots(az1+az2); roots(az1-az2)]); for j=1:length(lsf)-1 vmin=Inf; imin=-1; for
t=j:length(lsf) if lsf(t)<vmin vmin=lsf(t); imin=t; end; end; aux=lsf(j);
lsf(j)=lsf(imin); lsf(imin)=aux; end;

            LSF(:,k)=lsf(length(lsf)-p:length(lsf)-1);

        end;
```

```
% nnfind

function [ind,dout]=nnfind(X,v);

ind=-1; dout=inf;

for k=1:size(X,2)

    dk=sum((X(:,k)-v).^2);

    if dk<dout dout=dk; ind=k; end;

end;


% automaticth2wfw

function [fwxy,fwyx]=automaticth2wfw(th,fs2,fmax)

if nargin<2 fs2=8000.0; end;

if nargin<3 fmax=5000.0; end;

m=length(th); p=length(th(1).u);

polos(1:m)=struct('ai1',[],'ai2',[],'p1',[],'p2',[],'cc1',[],'cc2',[]);

for k=1:m

    lsf=transpose(th(k).u); part1=exp(i*lsf(2:2:p)); part2=exp(i*lsf(1:2:p));
ai=0.5*(poly([1 part1 conj(part1)])+poly([part2 -1 conj(part2)])); ai=ai(1:length(ai)-1);

    cc=zeros(size(lsf)); ai_=-ai(2:p+1); cc(1)=ai_(1); for n=2:p
cc(n)=ai_(n)+sum((1-((1:n-1)/n)).*ai_(1:n-1).*cc(n-1:-1:1)); end;

    R=roots(ai).'; Ra=angle(R); Rm=abs(R); ind=find((Ra>0.0)&(Ra<pi)); Ra=Ra(ind);
Rm=Rm(ind); R=R(ind); [Ra,ind]=sort(Ra); Rm=Rm(ind); R=R(ind);

    if length(R)>=p/2 ind=2:length(R); Ra=Ra(ind); Rm=Rm(ind); R=R(ind); end;
%R=0.98*R; %ai=zeros(1,p+1); ai(1:2*length(R)+1)=poly([R conj(R)]);

    subais=[ones(length(R),1) -2*real(R.') (R.*conj(R)).'];
emiwj=ones(3,length(R)); emiwj(2,:)=exp(-i*Ra); emiwj(3,:)=emiwj(2,:).*emiwj(2,:);
mods=1./abs(subais*emiwj); [modsmax,ind]=max(mods,[],1); ind=find(ind==[1:length(R)]); if
length(ind)<length(R) Ra=Ra(ind); Rm=Rm(ind); R=R(ind); end;

    polos(k).ai1=ai; polos(k).p1=R; polos(k).cc1=cc;

    lsf=transpose(th(k).v); part1=exp(i*lsf(2:2:p)); part2=exp(i*lsf(1:2:p));
ai=0.5*(poly([1 part1 conj(part1)])+poly([part2 -1 conj(part2)])); ai=ai(1:length(ai)-1);

    cc=zeros(size(lsf)); ai_=-ai(2:p+1); cc(1)=ai_(1); for n=2:p
cc(n)=ai_(n)+sum((1-((1:n-1)/n)).*ai_(1:n-1).*cc(n-1:-1:1)); end;

    R=roots(ai).'; Ra=angle(R); Rm=abs(R); ind=find((Ra>0.0)&(Ra<pi)); Ra=Ra(ind);
Rm=Rm(ind); R=R(ind); [Ra,ind]=sort(Ra); Rm=Rm(ind); R=R(ind);

    if length(R)>=p/2 ind=2:length(R); Ra=Ra(ind); Rm=Rm(ind); R=R(ind); end;
%R=0.98*R; %ai=zeros(1,p+1); ai(1:2*length(R)+1)=poly([R conj(R)]);

    subais=[ones(length(R),1) -2*real(R.') (R.*conj(R)).'];
emiwj=ones(3,length(R)); emiwj(2,:)=exp(-i*Ra); emiwj(3,:)=emiwj(2,:).*emiwj(2,:);
mods=1./abs(subais*emiwj); [modsmax,ind]=max(mods,[],1); ind=find(ind==[1:length(R)]); if
length(ind)<length(R) Ra=Ra(ind); Rm=Rm(ind); R=R(ind); end;

    polos(k).ai2=ai; polos(k).p2=R; polos(k).cc2=cc;
```

```
        end;

        mat=zeros(2*p,2*p);

        for k=1:m

            Np1=length(polos(k).p1); Np2=length(polos(k).p2);

            cc1=polos(k).cc1; cc2=polos(k).cc2; ccd1=cc1.*(1:length(cc1));
ccd2=cc2.*(1:length(cc2)); E=sqrt(sum(cc1.*cc1)/sum(ccd1.*ccd1)); ccd1=E*ccd1;
ccd2=E*ccd2;

            p1=angle(polos(k).p1); p2=angle(polos(k).p2);

            gr1opt=[]; gr2opt=[]; dmin=inf;

            for j=min([3 Np1 Np2]):min(Np1,Np2)

                gr1=1:j; gr1max=Np1+1-(j:-1:1);

                while 1

                    gr2=1:j; gr2max=Np2+1-(j:-1:1);

                    while 1

                        wx=[0 p1(gr1) pi]; wy=[0 p2(gr2) pi];

                        demasiadoirregular=0; if length(wx)>=3 signos=sign(wx-wy);
signos(length(signos))=-signos(length(signos)-1); for jj=1:length(signos)-2 if
isequal(signos(jj:jj+2),[1 -1 1]) || isequal(signos(jj:jj+2),[-1 1 -1])
demasiadoirregular=1; break; end; end; end;

                        if demasiadoirregular==0

                            dact1=0.0; dact2=0.0;

                            for jj=1:length(wx)-1

                                A=(wy(jj+1)-wy(jj))/(wx(jj+1)-wx(jj)); B=wy(jj)-A*wx(jj);

                                for j1=1:2*p

                                    for j2=1:j1

                                        if j1<=p if j1==j2
mat(j1,j2)=0.5*cc1(j1)*cc1(j2)*(((1/(j1+j2))*sin((j1+j2)*wx(jj+1))+wx(jj+1))-
((1/(j1+j2))*sin((j1+j2)*wx(jj))+wx(jj))); else
mat(j1,j2)=0.5*cc1(j1)*cc1(j2)*(((1/(j1+j2))*sin((j1+j2)*wx(jj+1))+(1/(j1-j2))*sin((j1-
j2)*wx(jj+1)))-((1/(j1+j2))*sin((j1+j2)*wx(jj))+(1/(j1-j2))*sin((j1-j2)*wx(jj)))); end;

                                        elseif j2>p j1_=j1-p; j2_=j2-p; if j1_==j2_
mat(j1,j2)=0.5*cc2(j1_)*cc2(j2_)*(((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj+1)+B))+wx(jj+
1))-((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj)+B))+wx(jj))); else
mat(j1,j2)=0.5*cc2(j1_)*cc2(j2_)*(((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj+1)+B))+(1/((j
1_-j2_)*A))*sin((j1_-j2_)*(A*wx(jj+1)+B)))-
((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj)+B))+(1/((j1_-j2_)*A))*sin((j1_-
j2_)*(A*wx(jj)+B)))); end;

                                        else j1_=j1-p; mat(j1,j2)=-
0.5*cc1(j1_)*cc2(j2)*(((1/(j1_+A*j2))*sin((j1_+A*j2)*wx(jj+1)+B*j2)+(1/(j1_-
A*j2))*sin((j1_-A*j2)*wx(jj+1)-B*j2))-
((1/(j1_+A*j2))*sin((j1_+A*j2)*wx(jj)+B*j2)+(1/(j1_-A*j2))*sin((j1_-A*j2)*wx(jj)-B*j2)));
end;

                                        if j1~=j2 mat(j2,j1)=mat(j1,j2); end;
```

```
                                    end;

                                end;

                                dact1=dact1+(1+A)*sum(sum(mat));

                                for j1=1:2*p

                                    for j2=1:j1

                                        if j1<=p if j1==j2 mat(j1,j2)=-
0.5*ccd1(j1)*ccd1(j2)*(((1/(j1+j2))*sin((j1+j2)*wx(jj+1))-wx(jj+1))-
((1/(j1+j2))*sin((j1+j2)*wx(jj))-wx(jj))); else mat(j1,j2)=-
0.5*ccd1(j1)*ccd1(j2)*(((1/(j1+j2))*sin((j1+j2)*wx(jj+1))-(1/(j1-j2))*sin((j1-
j2)*wx(jj+1)))-((1/(j1+j2))*sin((j1+j2)*wx(jj))-(1/(j1-j2))*sin((j1-j2)*wx(jj)))); end;

                                        elseif j2>p j1_=j1-p; j2_=j2-p; if j1_==j2_
mat(j1,j2)=-0.5*ccd2(j1_)*ccd2(j2_)*(((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj+1)+B))-
wx(jj+1))-((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj)+B))-wx(jj))); else mat(j1,j2)=-
0.5*ccd2(j1_)*ccd2(j2_)*(((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj+1)+B))-(1/((j1_-
j2_)*A))*sin((j1_-j2_)*(A*wx(jj+1)+B)))-((1/((j1_+j2_)*A))*sin((j1_+j2_)*(A*wx(jj)+B))-
(1/((j1_-j2_)*A))*sin((j1_-j2_)*(A*wx(jj)+B)))); end;

                                        else j1_=j1-p;
mat(j1,j2)=0.5*ccd1(j1_)*ccd2(j2)*(((1/(j1_+A*j2))*sin((j1_+A*j2)*wx(jj+1)+B*j2)-(1/(j1_-
A*j2))*sin((j1_-A*j2)*wx(jj+1)-B*j2))-((1/(j1_+A*j2))*sin((j1_+A*j2)*wx(jj)+B*j2)-
(1/(j1_-A*j2))*sin((j1_-A*j2)*wx(jj)-B*j2))); end;

                                        if j1~=j2 mat(j2,j1)=mat(j1,j2); end;

                                    end;

                                end;

                                dact2=dact2+(1+A)*sum(sum(mat));

                            end;

                            dact=dact1+dact2; if dact<dmin dmin=dact; gr1opt=gr1;
gr2opt=gr2; end;

                        end;

                        if gr2==gr2max break; else for jj=j:-1:1 if gr2(jj)<gr2max(jj)
gr2(jj)=gr2(jj)+1; for jjj=jj+1:j gr2(jjj)=gr2(jjj-1)+1; end; break; end; end; end;

                    end;

                    if gr1==gr1max break; else for jj=j:-1:1 if gr1(jj)<gr1max(jj)
gr1(jj)=gr1(jj)+1; for jjj=jj+1:j gr1(jjj)=gr1(jjj-1)+1; end; break; end; end; end;

                end;

            end;

            polos(k).p1=polos(k).p1(gr1opt); polos(k).p2=polos(k).p2(gr2opt);

        end;

        if m==1 && th(1).a==0 polos(1).p1=[polos(1).p1 -1]; polos(1).p2=[polos(1).p2 -1];
end; % trampilla para inicializaciones

        ff1=0:100:fs2; ff2=zeros(size(ff1));

        fwxy(1:m)=struct('x',[],'y',[]); fwyx=fwxy;
```

```
        for k=1:m

            ff1r=[0.0 (fmax/pi)*angle(polos(k).p1)]; ff2r=[0.0
(fmax/pi)*angle(polos(k).p2)];

            if ff1r(length(ff1r))>ff2r(length(ff2r)) ff1r=[ff1r
fs2*ff1r(length(ff1r))/ff2r(length(ff2r))]; ff2r=[ff2r fs2]; else ff2r=[ff2r
fs2*ff2r(length(ff2r))/ff1r(length(ff1r))]; ff1r=[ff1r fs2]; end;

            jjant=1; for j=1:length(ff1) for jj=jjant:length(ff1r)-1 if
(ff1r(jj)<=ff1(j))&&(ff1r(jj+1)>=ff1(j)) jjant=jj; break; end; end;
ff2(j)=ff2r(jj)+(ff2r(jj+1)-ff2r(jj))*(ff1(j)-ff1r(jj))/(ff1r(jj+1)-ff1r(jj)); end;

            fwxy(k).x=ff1; fwxy(k).y=ff2;

            jjant=1; for j=1:length(ff1) for jj=jjant:length(ff2r)-1 if
(ff2r(jj)<=ff1(j))&&(ff2r(jj+1)>=ff1(j)) jjant=jj; break; end; end;
ff2(j)=ff1r(jj)+(ff1r(jj+1)-ff1r(jj))*(ff1(j)-ff2r(jj))/(ff2r(jj+1)-ff2r(jj)); end;

            fwyx(k).x=ff1; fwyx(k).y=ff2;

        end;
```

HSM GMM Conversion

```
        function HSMgmmconvert(fth,fin,fout)

        %_____________________________________________________________________________
_
        %
        %    UPC Toolkit for HSM-based Voice Conversion
        %    Copyright (C) 2007  TALP Research Center
        %                        Universitat PolitËcnica de Catalunya (UPC)
        %
        %    This library is free software; you can redistribute it and/or
        %    modify it under the terms of the GNU Lesser General Public
        %    License as published by the Free Software Foundation; either
        %    version 2.1 of the License, or (at your option) any later version.
        %
        %    This library is distributed in the hope that it will be useful,
        %    but WITHOUT ANY WARRANTY; without even the implied warranty of
        %    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
        %    Lesser General Public License for more details.
        %
        %    You should have received a copy of the GNU Lesser General Public
```

```
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
%
%    Contact information:
%            AsunciÛn Moreno (asuncion.moreno@upc.edu)
%            TALP Research Center, Universitat PolitËcnica de Catalunya
%            UPC Campus Nord, Building D5
%            08034 - Barcelona, Spain
%_____________________________________________________________________________
_
%
%
%   HSMgmmconvert(VCfunction,fileIn,fileOut)
%
%   VCfunction (string): matlab file containing the voice conversion
%   function.
%   fileIn (string): matlab file containing the HSM parameters of the
%   signal to be converted.
%   fileOut (string): matlab file containing the HSM parameters of the
%   converted signal.
%


       if nargin<1 [fth,pth]=uigetfile('*.mat','Select transformation function'); if
(isequal(fth,0)|isequal(pth,0)) return; end; load([pth fth]); else load(fth); end;

       if nargin<2 [fin,pin]=uigetfile('*.mat','Select MAT file'); if
(isequal(fin,0)|isequal(pin,0)) return; end; fin=[pin fin]; end;

       if fin(length(fin)-3)=='.' fin=fin(1:length(fin)-4); end;

       load(fin);

       fmax=5000;

       u1=f0f(1); v1=f0f(2); u2=f0f(3); v2=f0f(4);

       p=length(th(1).u); m=length(th); mm=length(th2);

       thaux(1:m)=struct('d',0,'I',zeros(p,p)); for j=1:m thaux(j).d=det(th(j).E);
thaux(j).I=inv(th(j).E); end;

       th2aux(1:mm)=struct('d',0,'I',zeros(p,p)); for j=1:mm th2aux(j).d=det(th2(j).E);
th2aux(j).I=inv(th2(j).E); end;
```

```
        for k=1:length(picos)

            if picos(k).f0==0.0 continue; end;

            E=sum((picos(k).a).*(picos(k).a));

            [v,aiv]=aaalsf(picos(k).a,picos(k).f0,p);

            P=zeros(1,m); for j=1:m P(j)=th(j).a*(1/sqrt(thaux(j).d))*exp(-
0.5*transpose(v-th(j).u)*thaux(j).I*(v-th(j).u)); end; Psum=sum(P); P=(1/Psum)*P;

            vt=zeros(size(v)); for j=1:m vt=vt+P(j)*(th(j).v+th(j).R*thaux(j).I*(v-
th(j).u)); end;

            picos(k).f0=power(10,u2+v2*(log10(picos(k).f0)-u1)/v1);

            [aa2,pp2,aivt]=lsfaaapp(picos(k).f0,vt);

            if k>1 if picos(k-1).f0>0.0 picos(k).alfa=picos(k-
1).alfa+pi*(1/fs)*(picos(k).pm-picos(k-1).pm)*(picos(k).f0+picos(k-1).f0); else
picos(k).alfa=0.0; end; end;

            picos(k).p=pp2;

            picos(k).a=sqrt(E/sum(aa2.*aa2))*aa2;

            % componente estocastica

            P=zeros(1,mm); for j=1:mm P(j)=th2(j).a*(1/sqrt(th2aux(j).d))*exp(-
0.5*transpose(vt-th2(j).u)*th2aux(j).I*(vt-th2(j).u)); end; Psum=sum(P); P=(1/Psum)*P;

            vtt=zeros(size(vt)); for j=1:mm
vtt=vtt+P(j)*(th2(j).v+th2(j).R*th2aux(j).I*(vt-th2(j).u)); end; vtt=transpose(vtt);

            % ligera proteccion contra desorden en vtt

            vttaux=[0 vtt pi]; for j=2:length(vttaux)-1 if vttaux(j)<=vttaux(j-1) &
vttaux(j)<vttaux(j+1) & vttaux(j-1)<vttaux(j+1) vtt(j-1)=0.99*vttaux(j-
1)+0.01*vttaux(j+1); end; end;

            part1=exp(i*vtt(2:2:p)); part2=exp(i*vtt(1:2:p)); ai=0.5*(poly([1 part1
conj(part1)])+poly([part2 -1 conj(part2)])); ai=ai(1:length(ai)-1);

            picos(k).e=(picos(k).e(1)/ai(1))*ai;

        end;

        if nargin<3 [fout,pout]=uiputfile('*.mat','Save MAT file',[fin 'vc']); if
(isequal(fout,0)|isequal(pout,0)) return; end; fout=[pout fout]; end;

        save(fout,'L','fs','picos');

        %DPSsynthesize(fout,fout);



        % aaalsf

        function [lsf,aiout]=aaalsf(aa,f0,p)

        fmax=5000; if nargin<4 p=14; end;

        Nk=length(aa); ff=(1:Nk)*f0; PP=aa.*aa; lim=0.0001; fs=2*fmax;

        R=zeros(1,p+1); for j=1:p+1 R(j)=(1/Nk)*sum(PP.*cos(2*pi*(j-1)*ff/fs)); end;
```

```
        ai=zeros(1,p); e=R(1); for j=1:p if j==1 k=R(2)/R(1); ai(1)=k; e=(1-k*k)*e; else
k=R(j+1); for u=1:j-1 k=k-ai(u)*R(j-u+1); end; k=k/e; ai(j)=k; for u=1:j-1 aux(u)=ai(u)-
k*ai(j-u); end; ai(1:j-1)=aux(1:j-1); e=(1-k*k)*e; end; end;

        ai=[1 -ai]/sqrt(abs(R(1)-sum(ai.*R(2:p+1))));

        %mat(p+1,p+1)=0; for j1=1:p+1 for j2=1:p+1 mat(j1,j2)=R(abs(j1-j2)+1); end; end;
mat=inv(mat);

        %emiw=exp(-i*2*pi*ff/fs); emiwj=ones(p+1,length(ff)); for j=2:p+1
emiwj(j,:)=emiwj(j-1,:).*emiw; end;

        %denom(Nk)=0; h(p+1)=0; eis0=Inf; lap=0; lap2=0; while 1 denom=ai*emiwj;
e=PP.*(denom.*conj(denom)); eis=(1/Nk)*sum(e-log(e)-1); if (eis0>eis) if (eis0-
eis<lim)|(lap==20) break; end; eis0=eis; lap=lap+1; end; lap2=lap2+1; if lap2>50 break;
end; for j=1:p+1 h(j)=(1/Nk)*real(sum(emiwj(j,:)./denom)); end; ai=(mat*h')'; end;
ai=ai/sqrt((1/Nk)*sum(e));

        if nargout>=2 aiout=ai; end;

        ai=(1/ai(1))*ai; az1=[ai 0]; az2=az1(length(az1):-1:1);
lsf=angle([transpose(roots(az1+az2)) transpose(roots(az1-az2))]); for j=1:length(lsf)-1
vmin=Inf; imin=-1; for t=j:length(lsf) if lsf(t)<vmin vmin=lsf(t); imin=t; end; end;
aux=lsf(j); lsf(j)=lsf(imin); lsf(imin)=aux; end;

        lsf=transpose(lsf(length(lsf)-p:length(lsf)-1));



        % lsfaaapp

        function [aa,pp,ai]=lsfaaapp(f0,lsf)

        fmax=5000; Nk=ceil(fmax/f0)-1; p=length(lsf); lsf=transpose(lsf);

        part1=exp(i*lsf(2:2:p)); part2=exp(i*lsf(1:2:p)); ai=0.5*(poly([1 part1
conj(part1)])+poly([part2 -1 conj(part2)])); ai=ai(1:length(ai)-1);

        %%%factor=1.01; factorj=factor; for j=2:p+1 ai(j)=ai(j)*factorj;
factorj=factorj*factor; end;

        emw=exp(-i*pi*f0/fmax); emiw=ones(1,Nk); emiw(1)=emw; for j=2:Nk emiw(j)=emiw(j-
1)*emw; end; emiwj=ones(p+1,Nk); for j=2:p+1 emiwj(j,:)=emiwj(j-1,:).*emiw; end;

        Af=ai*emiwj; aa=abs(1./Af); pp=-angle(Af);
```

HSM Weighted Frequency Warping Conversion

```
        function HSMwfwconvert(fth,fin,fout,modo)

        %_____________________________________________________________________________
_
        %
        %    UPC Toolkit for HSM-based Voice Conversion
        %    Copyright (C) 2007  TALP Research Center
        %                        Universitat PolitËcnica de Catalunya (UPC)
        %
```

```
%    This library is free software; you can redistribute it and/or
%    modify it under the terms of the GNU Lesser General Public
%    License as published by the Free Software Foundation; either
%    version 2.1 of the License, or (at your option) any later version.
%
%    This library is distributed in the hope that it will be useful,
%    but WITHOUT ANY WARRANTY; without even the implied warranty of
%    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
%    Lesser General Public License for more details.
%
%    You should have received a copy of the GNU Lesser General Public
%    License along with this library; if not, write to the Free Software
%    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
%
%
%    Contact information:
%            AsunciÛn Moreno (asuncion.moreno@upc.edu)
%            TALP Research Center, Universitat PolitËcnica de Catalunya
%            UPC Campus Nord, Building D5
%            08034 - Barcelona, Spain
%_____________________________________________________________________________
_
%
%
%   HSMwfwconvert(VCfunction,fileIn,fileOut)
%
%   VCfunction (string): matlab file containing the voice conversion
%   function.
%   fileIn (string): matlab file containing the HSM parameters of the
%   signal to be converted.
%   fileOut (string): matlab file containing the HSM parameters of the
%   converted signal.
%
```

```
        if nargin<1 [fth,pth]=uigetfile('*.mat','Select transformation function'); if
(isequal(fth,0)|isequal(pth,0)) return; end; load([pth fth]); else load(fth); end;

        if nargin<2 [fin,pin]=uigetfile('*.mat','Select MAT file'); if
(isequal(fin,0)|isequal(pin,0)) return; end; fin=[pin fin]; end;

        if nargin<4 modo='fwa'; end;

        if fin(length(fin)-3)=='.' fin=fin(1:length(fin)-4); end;

        load(fin);

        fmax=5000.0;

        fasacum=0.0;

        u1=f0f(1); v1=f0f(2); u2=f0f(3); v2=f0f(4);

        f01s=zeros(1,length(picos)); f02s=zeros(size(f01s)); dph=0.0;

        for k=1:length(picos)

                f01s(k)=picos(k).f0;

                if f01s(k)>0.0

                        f02s(k)=power(10.0,u2+v2*(log10(f01s(k))-u1)/v1);

                        if k>1 && f01s(k-1)>0

                                dph=dph+(picos(k).pm-picos(k-1).pm)*pi*((f02s(k)-
f01s(k))+(f02s(k-1)-f01s(k-1)))/fs;

                                picos(k).alfa=picos(k).alfa+dph;

                        else dph=0.0; end;

                else f02s(k)=0.0; dph=0.0; end;

        end;

        clear u1 v1 u2 v2;

        p=length(th(1).u); m=length(th); mm=length(th2);

        for j=1:m th(j).d=det(th(j).E); th(j).I=inv(th(j).E); end;

        for j=1:mm th2(j).d=det(th2(j).E); th2(j).I=inv(th2(j).E); end;

        fx=fw(1).x;

        for k=1:length(picos)

            if picos(k).f0==0.0 continue; end;

            [v,aiv]=aaalsf(picos(k).a,picos(k).f0,p);

            P=zeros(1,m); for j=1:m P(j)=th(j).a*(1/sqrt(th(j).d))*exp(-0.5*transpose(v-
th(j).u)*th(j).I*(v-th(j).u)); end; Psum=sum(P); P=(1/Psum)*P;

            vt=zeros(size(v)); for j=1:m vt=vt+P(j)*(th(j).v+th(j).R*th(j).I*(v-th(j).u));
end;

            aivt=lsfadap(vt);
```

```
            eevt=ones(p+1,ceil(fmax/f02s(k))-1); eevt(2,1)=exp(-i*pi*f02s(k)/fmax); for
jj=2:size(eevt,2) eevt(2,jj)=eevt(2,jj-1)*eevt(2,1); end; for jj=3:p+1
eevt(jj,:)=eevt(jj-1,:).*eevt(2,:); end;

            Afvt=1./(aivt*eevt);

            E1=sum(picos(k).a.*picos(k).a);

            Afvt=Afvt*sqrt(E1/sum(Afvt.*conj(Afvt))); Evt=Afvt.*conj(Afvt);
aavt=sqrt(Evt);

            % parte estocastica

            PP=zeros(1,mm); for j=1:mm PP(j)=th2(j).a*(1/sqrt(th2(j).d))*exp(-
0.5*transpose(vt-th2(j).u)*th2(j).I*(vt-th2(j).u)); end; PPsum=sum(PP); PP=(1/PPsum)*PP;

            vtt=zeros(size(vt)); for j=1:mm vtt=vtt+PP(j)*(th2(j).v+th2(j).R*th2(j).I*(vt-
th2(j).u)); end;

            % ligera proteccion contra desorden en vtt

            vttaux=[0; vtt; pi]; for j=2:length(vttaux)-1 if vttaux(j)<=vttaux(j-1) &
vttaux(j)<vttaux(j+1) & vttaux(j-1)<vttaux(j+1) vtt(j-1)=0.99*vttaux(j-
1)+0.01*vttaux(j+1); end; end;

            aivtt=lsfadap(vtt);

            aivtt=(picos(k).e(1)/aivtt(1))*aivtt;

            % Obtener la trayectoria de warping ponderando con P los warpings individuales
de cada gaussiana

            fy=zeros(size(fw(1).y)); for j=1:m fy=fy+P(j)*fw(j).y; end;

            ff1=[0.0 (1:length(picos(k).a))*f01s(k) fmax];

            aa1=picos(k).a;

            pp1=picos(k).p;

            ap1=aa1.*exp(i*pp1);

            ap1=[aa1(1) ap1 0.0];

            aa1=log([aa1(1) aa1 min(aa1)]);

            aa2=zeros(1,ceil(fmax/f02s(k))-1); pp2=zeros(size(aa2));

            jjant1=1; jjant2=1; scale=0.0;

            for j=1:length(aa2)

                f2j=j*f02s(k); % buscar region de fy donde esta el armonico, y buscar la
antiimagen en fx

                for jj=jjant1:length(fy)-1 if (f2j>=fy(jj))&&(f2j<fy(jj+1)) jjant1=jj;
break; end; end;

                f1j=fx(jj)+(fx(jj+1)-fx(jj))*(f2j-fy(jj))/(fy(jj+1)-fy(jj));

                if f1j<fmax % muestrear envolvente de amplitud y fase

                    for jj=jjant2:length(ff1)-1 if (f1j>=ff1(jj))&&(f1j<ff1(jj+1))
jjant2=jj; break; end; end;

                    aa2(j)=exp(aa1(jj)+(aa1(jj+1)-aa1(jj))*(f1j-ff1(jj))/(ff1(jj+1)-
ff1(jj)));
```

```
                    pp2(j)=angle(ap1(jj)+(ap1(jj+1)-ap1(jj))*(f1j-ff1(jj))/(ff1(jj+1)-
ff1(jj)));

                else % si es mayor que fmax, saco info del filtro convertido gmm

                    if scale==0.0 scale=aa2(j-1)/abs(Afvt(j-1)); end;

                    aa2(j)=scale*abs(Afvt(j)); pp2(j)=angle(Afvt(j));

                end;

            end;

            if isequal(modo,'fwa') % corrijo la energia

                flimsm=1000.0; nf=ceil(flimsm/f02s(k)-1); win=1.0-(f02s(k)/flimsm)*(-
nf:nf); win=(1/sum(win))*win; % ventana triangular con gran anchura en f

                g=aavt./aa2;

                g=log(g); gsm=g; for j=1:length(g) ind=j+(-nf:nf); if j<=nf for
jj=1:length(ind) if ind(jj)<0 ind(jj)=-ind(jj); elseif ind(jj)==0 ind(jj)=1; else break;
end; end; elseif j>length(g)-nf for jj=length(ind):-1:1 if ind(jj)>length(g)+1
ind(jj)=2*length(g)+2-ind(jj); elseif ind(jj)==length(g)+1 ind(jj)=length(g); else break;
end; end; end; gsm(j)=sum(g(ind).*win); end; g=exp(gsm);

                aa2=aa2.*g;

            end;

            aa2=sqrt(E1/sum(aa2.*aa2))*aa2;

            picos(k).f0=f02s(k); picos(k).a=aa2; picos(k).p=pp2; picos(k).e=aivtt;

        end;

        if nargin<3 [fout,pout]=uiputfile('*.mat','Save MAT file',[fin 'vc']); if
(isequal(fout,0)|isequal(pout,0)) return; end; fout=[pout fout]; end;

        save(fout,'L','fs','picos');



        % aaalsf

        function [lsf,aiout]=aaalsf(aa,f0,p)

        fmax=5000; if nargin<4 p=14; end;

        Nk=length(aa); ff=(1:Nk)*f0; PP=aa.*aa; lim=0.0001; fs=2*fmax;

        R=zeros(1,p+1); for j=1:p+1 R(j)=(1/Nk)*sum(PP.*cos(2*pi*(j-1)*ff/fs)); end;

        ai=zeros(1,p); e=R(1); for j=1:p if j==1 k=R(2)/R(1); ai(1)=k; e=(1-k*k)*e; else
k=R(j+1); for u=1:j-1 k=k-ai(u)*R(j-u+1); end; k=k/e; ai(j)=k; for u=1:j-1 aux(u)=ai(u)-
k*ai(j-u); end; ai(1:j-1)=aux(1:j-1); e=(1-k*k)*e; end; end;

        ai=[1 -ai]/sqrt(abs(R(1)-sum(ai.*R(2:p+1))));

        %mat(p+1,p+1)=0; for j1=1:p+1 for j2=1:p+1 mat(j1,j2)=R(abs(j1-j2)+1); end; end;
mat=inv(mat);

        %emiw=exp(-i*2*pi*ff/fs); emiwj=ones(p+1,length(ff)); for j=2:p+1
emiwj(j,:)=emiwj(j-1,:).*emiw; end;

        %denom(Nk)=0; h(p+1)=0; eis0=Inf; lap=0; lap2=0;
```

```
        %while 1 denom=ai*emiwj; e=PP.*(denom.*conj(denom)); eis=(1/Nk)*sum(e-log(e)-1);
if (eis0>eis) if (eis0-eis<lim)|(lap==20) break; end; eis0=eis; lap=lap+1; end;
lap2=lap2+1; if lap2>50 break; end; for j=1:p+1 h(j)=(1/Nk)*real(sum(emiwj(j,:)./denom));
end; ai=(mat*h')'; end;

        %ai=ai/sqrt((1/Nk)*sum(e));

        if nargout>=2 aiout=ai; end;

        ai=(1/ai(1))*ai; az1=[ai 0]; az2=az1(length(az1):-1:1);
lsf=angle([transpose(roots(az1+az2)) transpose(roots(az1-az2))]); for j=1:length(lsf)-1
vmin=Inf; imin=-1; for t=j:length(lsf) if lsf(t)<vmin vmin=lsf(t); imin=t; end; end;
aux=lsf(j); lsf(j)=lsf(imin); lsf(imin)=aux; end;

        lsf=transpose(lsf(length(lsf)-p:length(lsf)-1));



        % lsfadap

        function ai=lsfadap(lsf,factor)

        p=length(lsf);

        lsf=transpose(lsf);

        part1=exp(i*lsf(2:2:p)); part2=exp(i*lsf(1:2:p));

        ai=0.5*(poly([1 part1 conj(part1)])+poly([part2 -1 conj(part2)]));

        ai=ai(1:length(ai)-1);

        if nargin>1 factorj=factor; for j=2:p+1 ai(j)=ai(j)*factorj;
factorj=factorj*factor; end; end;
```