“Hey! Stop sounding like me!!”

Voice Conversion

18-551, Spring 2004
Group 9, Final Report

Vern Grenade
Jean Keomany
Raphael Ullmann

{vgrenade, jkeomany, rullmann}@andrew.cmu.edu

Table of contents

The voice of mankind has from time to time been controversial while at other times has been a source of inspiration.
--Vern Grenade

1. Introduction

The movie, computing, and toy industries rely heavily on novel ways to draw customers to their business. Innovations in voice technology, particularly in voice conversion, have the potential to lead to the development of unique products and services. Voice conversion could be used to preserve the original actor's voice when dubbing a movie. It could be useful for text-to-speech personification, where the email you receive is read back in the sender's voice. Plus it could be used for speaker normalization for speech recognition system, allowing all input voice to be converted to a voice that the recognition system is designed to handle. In addition, voice conversion done in real-time could be used in karaoke applications, where you can sing and sound like your favorite star. Furthermore, it could be used to create "spy toys" that allow users to mask and record their voice so they can trick their friend's over the phone. In all, a voice conversion system could offer benefits to big businesses in terms of increased profits, and to customers in terms of the availability of innovative and fun applications.

In our project we introduce a beneficial and efficient process of converting the voice properties of a source speaker to that of a target speaker. The essential speech properties considered in this process include the fundamental pitch and the formant frequencies of both speakers. Our goal is to implement a system that is text-independent. Thus, the user can say any sentence and the output will sound as if the target speaker had said it.

2. Our Approach

In our quest to achieve our expectation of a high quality voice conversion system we spent a great deal of time doing research. Initially, research centered on gathering information about speech projects that were previously done in 551. We were able to learn about the widely used techniques of speech synthesis including the use of LPC and PARCOR coefficients. Also, we learned about how theses techniques assumed that speech production was modeled as an excitation signal being feed into a vocal tract filter. In addition, we learned about the difficulties that previous groups faced in terms of finding C-code, getting detailed description of algorithms, finishing all aspect of their project on time, and achieving high quality speech output.

We then sought to find a proposed voice conversion technique that produced high quality speech results and utilized widely used techniques and speech modeling schemes. We found a 1997 publication by L. Arslan and D. Talkin entitled *Voice Conversion by Codebook Mapping of Line Spectral Frequencies and Excitation Spectrum* [1]. In this document they proposed creating codebooks of line spectral frequencies (LSF) and the residual (excitation) signal of source and target speakers that could then be converted from one speaker to the next. We felt that this technique was unique from previous projects but more importantly promised high quality speech results. Finally, after some deliberations we decided to implement this voice conversion system.

3. Prior Projects

Three groups did voice conversion in 18-551 (2 groups in 1999 and 1 in 2001). Their algorithms were basically based on the modification of the LPC coefficients or the PARCOR coefficients and the excitation pitch.

First, they divided the input signal into frames (about 20 ms) and found the LPC or PARCOR coefficients in each frame. They then determined if the current frame was a voiced sound or an unvoiced sound. For voiced sounds, a pitch detection algorithm was used to determine the pitch of the portion of the speech. The voice conversion was then performed based on a Cumulative Density Function (CDF), which is described in [4]. The LPC or PARCOR coefficients were replaced by the corresponding target coefficients. The speech was finally re-synthesized using the new coefficients and an impulse train with the target pitch for voiced sounds or white noise for unvoiced sounds as excitation. Figure 1 illustrates the final stage of their conversion process.

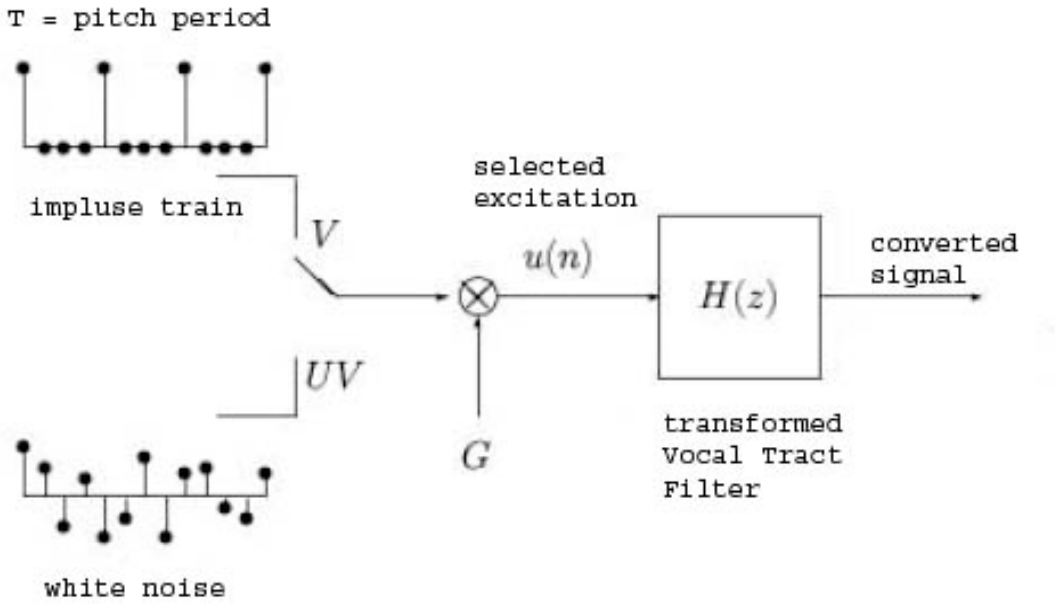

Figure 1: Simplified model for speech conversion

Previous groups reported that the quality of the re-synthesized speech was poor and contained cracks and pops. In addition, the output speech was not always intelligible.

One cause of their results might be the fact that they used LPC coefficients for the voice conversion. Indeed, LPC coefficients are very sensitive. Relatively small changes in the representation of the LPC coefficients result in a “large” change in the pole locations of the vocal tract filter model. This can lead to an unstable filter.

In order to solve this problem we decided to use Line Spectral Frequencies (LSF), which are a different way to represent formant frequencies. LSF are always stable and have advantages such as good interpolation properties and distortion independence properties. More information about LSF are given in [5] and [6].

4. Our Algorithm

As mentioned earlier, our system consists of two parts: A training part and a conversion part. Both parts process speech signals sampled at 16 kHz and 16 bits per sample. Speech is pre-emphasized, decomposed into frames of 256 samples (16 ms), windowed and processed frame by frame. We decided to use 14^{th} order LSF analysis.

4.1 Training Part

In the training part, we build codebooks containing the 14 LSF and the glottal excitation spectrum (256 samples) for each phoneme. There are approximately 40 phonemes in the English language (around 50 with the closures of the phonemes). The idea is that we can map phonemes one to one from a source speaker to a target speaker if we want to convert one voice to another. The training part was implemented in MATLAB.

4.1.1 Overview

In order to extract a given phoneme from speech, we use training sentences whose phonetic transcription is already known. Dynamic Time Warping (DTW) is used to make the phoneme extraction as precise as possible. Average LSF and the excitation spectrum are determined for each phoneme. Ten phonetically rich sentences are used for the training part providing 52 different phonemes. The codebooks are finally written to disk in a format suitable for processing by the EVM.

4.1.2 The TIMIT Database

In [2], Arslan used Sentence Hidden Markov Models (Sentence HMMs) to detect phonemes in a recording in order to build the codebooks. We learned from previous 551 projects that phoneme detection was too complicated, and came up with a different approach: If we had a "reference voice", whose speech had already been phonetically segmented, we could time-align the same speech from any other voice to the reference, and thus obtain a phonetic segmentation of any other voice.

We talked about our idea to Prof. Stern. He thought it was a good idea, and told us about the TIMIT (TI and MIT) database, which contains speech from over 400 different speakers along with the phonetic segmentation. For each speaker, there are ten phonetically diverse sentences available, so almost all phonemes are represented.

4.1.3 Dynamic Time Warping (DTW)

During the training part, the speaker is asked to pronounce the training sentences at the same rate as the reference voice from TIMIT. This is made easier by playing the reference recording simultaneously as the speech is recorded. However, it is nearly impossible for a person to pronounce all the phonemes in a sentence at the same time than the reference. This is why we use Dynamic Time Warping (DTW) to time-align the person's speech to the reference recording.

The basic concept of DTW is to create a grid with the reference and the input speech on the X- and Y-axis respectively. Each cell in this grid contains the "distance" between the two corresponding speech frames. The goal is to find the path from the first to the last frame with the lowest cumulative distance. This path corresponds to the best alignment between reference and input.

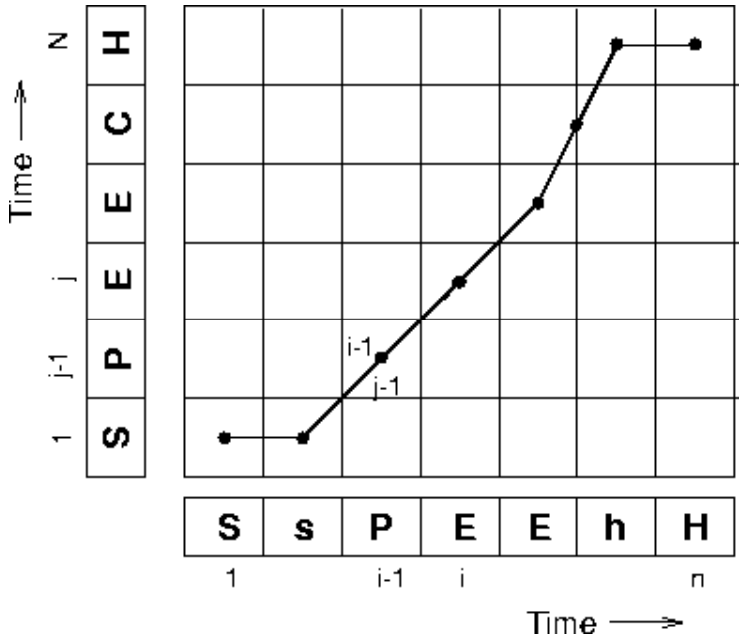

Figure 2: Dynamic Time Warping

There are many variants of DTW, the main difference being how the algorithm deals with frames that have to be combined together or stretched. We used the MATLAB implementation on [7] by Dan Ellis of Columbia University. This algorithm either combines or stretches a series of frames, but does not interpolate between adjacent frames to recover a "missing" frame like the "C" in the above figure.

We tried different methods for computing the distance between frames and for combining or stretching frames. We found that the distance measure proposed in [1] introduced the fewest distortions while still providing a good alignment. To combine frames, e.g. the "S" in the above figure, we simply keep the first frame of the series and discard the other frames. Similarly, to stretch frames, the last frame of the series is repeated until the same length as in the reference is reached.

We tested a more sophisticated approach which takes the average of frames to be combined, but didn't notice any improvement in quality.

4.1.4 Extraction of LSF and Excitation Spectrum

Now that the input speech has been time-aligned to the TIMIT reference, we can use the phonetic segmentation provided on the TIMIT CD to locate the desired phonemes. We wrote a C program to convert the TIMIT phonetic transcription to a list indicating which phoneme occurs during which frames:

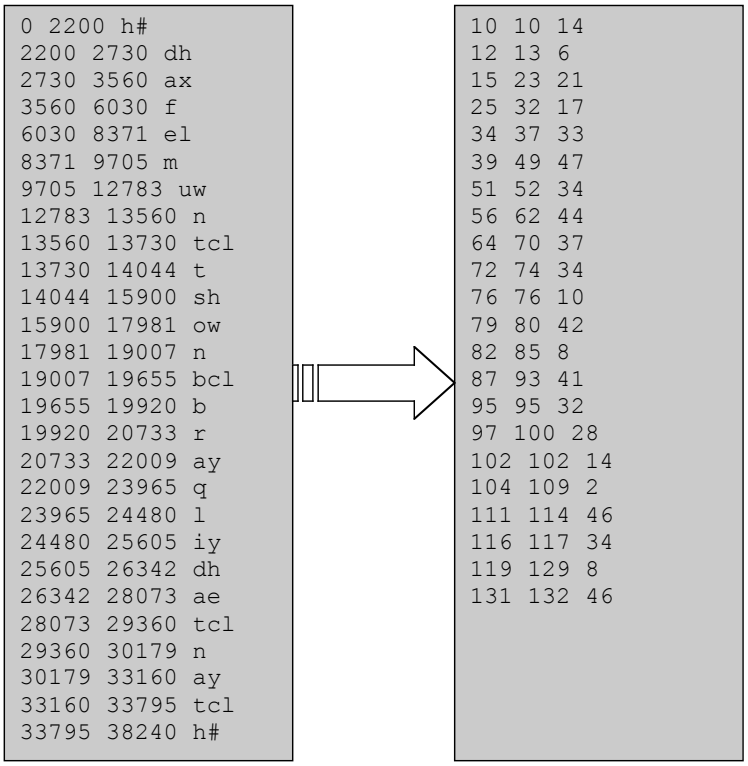

Figure 3: "The full moon shone brightly that night" ("h#" represents silence)

On the left, the TIMIT transcription indicates the first and the last sample in the recording for a given phoneme. On the right, this information has been converted to show the number of the first and the last frame(s) containing phoneme number n, where n is determined with the lookup table below:

1	aa	14	dh	27	ix	40	pcl
2	ae	15	dx	28	iy	41	q
3	ah	16	eh	29	jh	42	r
4	ao	17	el	30	k	43	s
5	aw	18	epi	31	kcl	44	sh
6	ax	19	er	32	l	45	t
7	axr	20	ey	33	m	46	tcl
8	ay	21	f	34	n	47	uw
9	b	22	g	35	ng	48	ux
10	bcl	23	gcl	36	nx	49	v
11	ch	24	hh	37	ow	50	w
12	d	25	hv	38	oy	51	y
13	dcl	26	ih	39	p	52	z

Figure 4: Phonemes in TIMIT and corresponding phoneme number

Some phonemes are left out during the conversion to the frame-list format, because we only consider phonemes with duration of at least one frame (256 samples). Also, most phonemes don’t start or end at multiples of 256; thus we only keep phonemes that last at least during two consecutive multiples of 256. In other words, if a phoneme starts at sample 226 and ends at sample 502, we will not consider it. If it ends at sample 522 (or more than 512), we will keep it. The reason for this is that DTW works on a frame-by-frame basis, so phonemes cannot be reliably extracted unless we use the same frame division as for DTW.

Now that we know the frames of interest, we add the LSF to the corresponding phoneme vector in the LSF codebook, and increment a counter that keeps track of how many LSF sets have been summed together for each phoneme. Once the training part is completed, each vector is divided by the corresponding counter in order to obtain an average LSF set per phoneme.

For the excitation codebook, we wanted to avoid the opening and closing parts of phonemes, during which the glottal excitation can vary greatly. Thus we decided to extract only one frame of excitation signal per phoneme. We first determined manually the longest duration of all 52 phonemes across all ten sentences. We then selected one frame in the middle of the longest duration for each phoneme to extract the excitation.

Subjective listening tests showed that this combination (average of different frames for the LSF, one frame for the excitation) gave the best results in the conversion part.

4.1.5 Summary

The following flowchart shows the implementation of the training part in MATLAB:

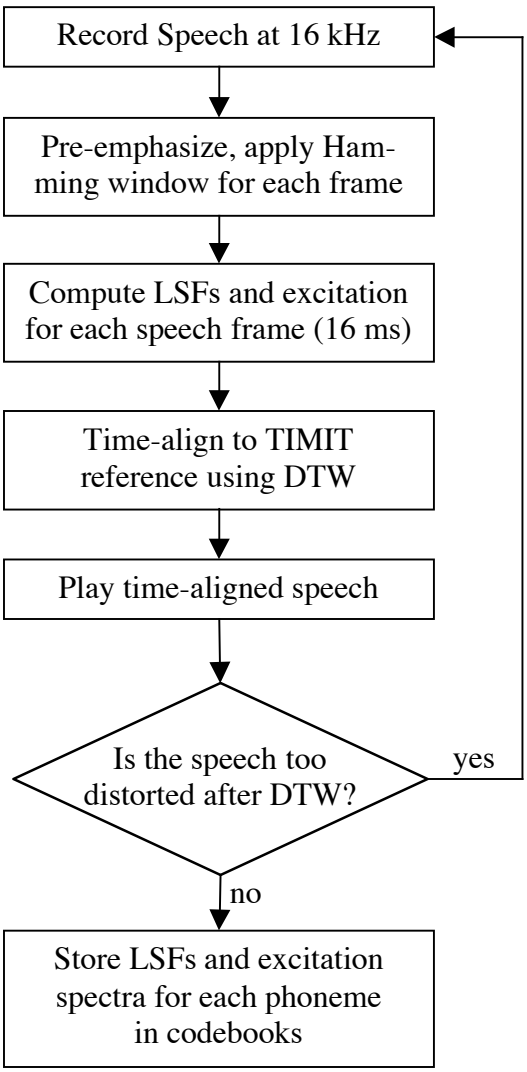

Figure 5: Flowchart of the training part

This is repeated for all ten training sentences. The LSF and excitation spectrum codebooks are then written to disk. The imaginary and complex parts of the excitation spectra are separated by spaces for compliance with the real-complex interleaved format of the TI IFFT function.

4.2 Conversion Part

4.2.1 Overview

In the conversion part, the LSF of the input speech signal are computed frame-by-frame and approximated by a weighted combination of LSF vectors from the source LSF codebook. Using these same weights the target LSF and excitation are estimated. The output speech is obtained by filtering the target excitation with the vocal tract generated with the target LSF.

4.2.2 Pre-emphasis

First of all, the input signal is pre-emphasized with the filter $P(z) = 1 - 0.95z^{-1}$. The goal of the pre-emphasis filter is to amplify the higher frequencies in order to flatten the spectrum. This leads to a better result for the calculation of the LPC coefficients.

4.2.3 Windowing

The output of the pre-emphasis filter is then multiplied by a Hamming window function given by:

$$w(i) = 0.54 - 0.46\cos\left(\frac{2\pi}{n-1}i\right) \qquad \text{for } 0 \le i \le 255$$

The beginning and end of each frame are tapered to zero. This reduces discontinuities (sudden onset or offset) resulting in smoother transitions of the signal from frame to frame.

4.2.4 Input LPC and LSF

Next, 14 LPC coefficients are estimated. The LPC coefficients are obtained by minimizing the sum of the squared components of the error e(n) between the original speech signal s(n) and the estimated speech signal.

$$e(n) = s(n) - \sum_{k=1}^{14} a_k s(n-k)$$

The most efficient algorithm for computing the LPC coefficients is the Levinson-Durbin algorithm. It first performs an autocorrelation of the input signal and then uses a recursion algorithm to inverse a Toeplitz matrix in order to get a unique set of LPC coefficients. This matrix inversion has a complexity of $O(p^2)$ (where p is the predictor order) vs. $O(p^3)$ for standard Gaussian inversion. The LPC coefficients are then converted to line spectral frequencies.

We found C-codes for the Levinson-Durbin algorithm and the conversion from LPC to LSF in [8].

4.2.5 Weights Estimate

The input LSF vector **w** is then approximated by a linear combination of the LSF vectors $\mathbf{S_i}$ of the source codebook. The advantage of using a linear combination is that we can also estimate phonemes that are not in the codebooks. The method used for estimating the weights $\mathbf{v_i}$ for the combination is given in [1]:

"Line spectral frequencies vector **w** is compared with each LSF centroid, $\mathbf{S_i}$, in the source codebook and the distance, $\mathbf{d_i}$, corresponding to each phoneme is calculated. The distance calculation is based on a perceptual criterion where closely spaced line spectral frequencies, which are likely to correspond to formant locations, are assigned higher weights (…).

$$h_k = \frac{1}{\arg\min\left(\left|w_k - w_{k-1}\right|, \left|w_k - w_{k+1}\right|\right)} \quad \text{for k = 1, 2, …, 14}$$

$$d_i = \sum_{k=1}^{14} h_k \left|w_k - S_{ik}\right| \quad \text{for i = 1, 2, …, 52}$$

$$v_i = \frac{e^{-\gamma d_i}}{\sum_{l=1}^{52} e^{-\gamma d_l}} \quad \text{for i = 1, 2, …, 52}$$

(…) where the value of γ for each frame is found by an incremental search with the criterion of minimizing the perceptual weighted distance between the approximated LSF vector and the original LSF vector."

In [2], L. Arslan reported that the value of γ varies from 0.2 to 2. In order to simplify the algorithm, we used a constant value for γ. After several tests, we concluded that γ = 2 gives the best approximation of the input LSF vector.

4.2.6 Target LSF and LPC

Since the codebooks are created in the same conditions and with the same sentences for every speaker, the i^{th} phoneme in the source codebook should correspond exactly to the i^{th} phoneme in the target codebook for the conversion.

Therefore, the weights are applied to the LSF vectors $\mathbf{T_i}$ of the target codebook in order to obtain the corresponding target LSF vector $\mathbf{w^t}$ of the current speech frame.

$$w^t = \sum_{i=1}^{52} v_i T_i$$

The target LSF are then converted to LPC coefficients using the C-code found in [8].

4.2.7 Target Vocal Tract

The vocal tract of the target speaker can now be modeled using the estimated target LPC coefficients. The exact form of the vocal tract used by the author is a mystery. In [1], the vocal tract filter is expressed as

$$V_t = \left| \frac{1}{1 - \sum_{k=1}^{14} a_k e^{-jk\omega}} \right|^{\beta}$$

where β is an undefined variable. In [2], the vocal tract filter is modeled as

$$V_t = \left| \frac{1}{1 - \sum_{k=1}^{14} a_k e^{-jk\omega}} \right|$$

whereas in [3] it is defined as

$$V_t = \left| \frac{1}{1 - \sum_{k=1}^{14} a_k e^{-jk\omega}} \right|^{1/2}.$$

None of these expressions looks like the standard form of the vocal tract that we can find in speech processing books. After many tests, we figured out, helped by Evandro Gouvea, that the human ear is insensitive to the phase of the vocal tract. Consequently, it makes no difference to use the magnitude of the filter or to keep complex results. However, the purpose of the exponent is still unknown. Finally, in order to save memory (array of absolute values instead of complex numbers) and computation time, we adopted the form proposed in [2].

4.2.8 Target excitation

In [1], an overall filter $H(\omega)$, which is a weighted combination of excitation codeword filters, is constructed using the set of weights computed for the LSF:

$$H(\omega) = \sum_{i=1}^{52} v_i \frac{\left|U_i^t(\omega)\right|}{\left|U_i^s(\omega)\right|}$$

where $\left|U_i^t(\omega)\right|$ and $\left|U_i^s(\omega)\right|$ denote the average magnitude spectra of the target and source excitation respectively for the i[th] phoneme. The target excitation spectrum $G_t(\omega)$ is then obtained by applying the filter to the DFT of the input speaker excitation $g_s(n)$:

$$G_t(\omega) = H(\omega) DFT\{g_s(n)\}$$

Following the method described above, we obtained an excitation spectrum containing unexpected peaks caused by some very small values in the denominator of the filter. In order to remove these peaks, we added a constant value of about 0.1 first to every value and then only to the values below a certain threshold. The output became much smoother but still sounded like the source voice. We finally noticed that by adding a constant to every value, the filter was close to 1. Consequently, the output was constructed with the excitation of the input and the target vocal tract.

We then asked Prof. Stern, other CMU faculty members and graduate students to verify the method used to obtain the excitation filter and for any advice that would improve our sound quality. Prof. Stern recommended the use of cepstral coefficients. The cepstral coefficients c(n) of a signal s(n) are defined as:

$$c(n) = ifft(\log|fft(s(n))|)$$

The number of coefficients used for the calculations determines the smoothness of the output. By using only the twelve first coefficients, the excitation filter becomes:

$$H(\omega) = \sum_{i=1}^{52} v_i \exp(fft(c^t_{i,12}(n) - c^s_{i,12}(n)))$$

where $c^t_{i,12}$ and $c^s_{i,12}$ are the twelve first cepstral coefficients of the i^{th} phoneme excitation of the target and the source speaker respectively, padded with zeros. With this filter, the output speech was smoother but still had the same unknown voice no matter the source and target speaker. The pitch was unchanged. Even by increasing the number of cepstral coefficients used, the quality of the output did not improve.

Our TA Efstratios recommended that we normalize the excitation to avoid sudden changes in amplitude from frame to frame. This recommendation led to minimal improvement in sound quality so we decided to modify the way in which we "transformed" the source to the target excitation. We did this by representing the target excitation as a linear combination of the excitation spectra in the target speaker's codebook:

$$G_t(\omega) = \sum_{i=1}^{52} v_i U^t_i(\omega)$$

The quality of the output speech is poor but at least, it sounds like the target speaker.

4.2.9 Combined output

The combination of the target excitation and vocal tract can be done in frequency domain or in time domain.

In frequency domain, we need to compute the vocal tract spectrum from the target LPC coefficients, then multiply the vocal tract with the excitation spectrum and finally take the inverse FFT of the result.

In time domain, we first perform the inverse FFT of the target excitation spectrum. Next, the output signal y(n) is obtained by using the following difference equation:

$$y(n) = g_t(n) + \sum_{k=1}^{14} a_k^t y(n-k)$$

where $g_t(n)$ is the excitation in time domain and a_k^t are the target LPC.

We chose to combine the vocal tract and the excitation in time domain because it requires less operations and the implementation is much easier and faster. The IFFT was performed with an optimized TI radix 4 FFT implementation.

4.2.10 De-emphasis

Pre-emphasis is removed from the speech by applying the inverse pre-emphasis filter:

$$P^{-1} = \frac{1}{1 - 0.95z^{-1}}$$

Finally, by making the amplitude of the output speech equal to that of the input speech, we were able to obtain speech quality that was intelligible and could be recognized as being converted from one speaker to the next.

4.2.11 Summary

The following flowchart summarizes the conversion process:

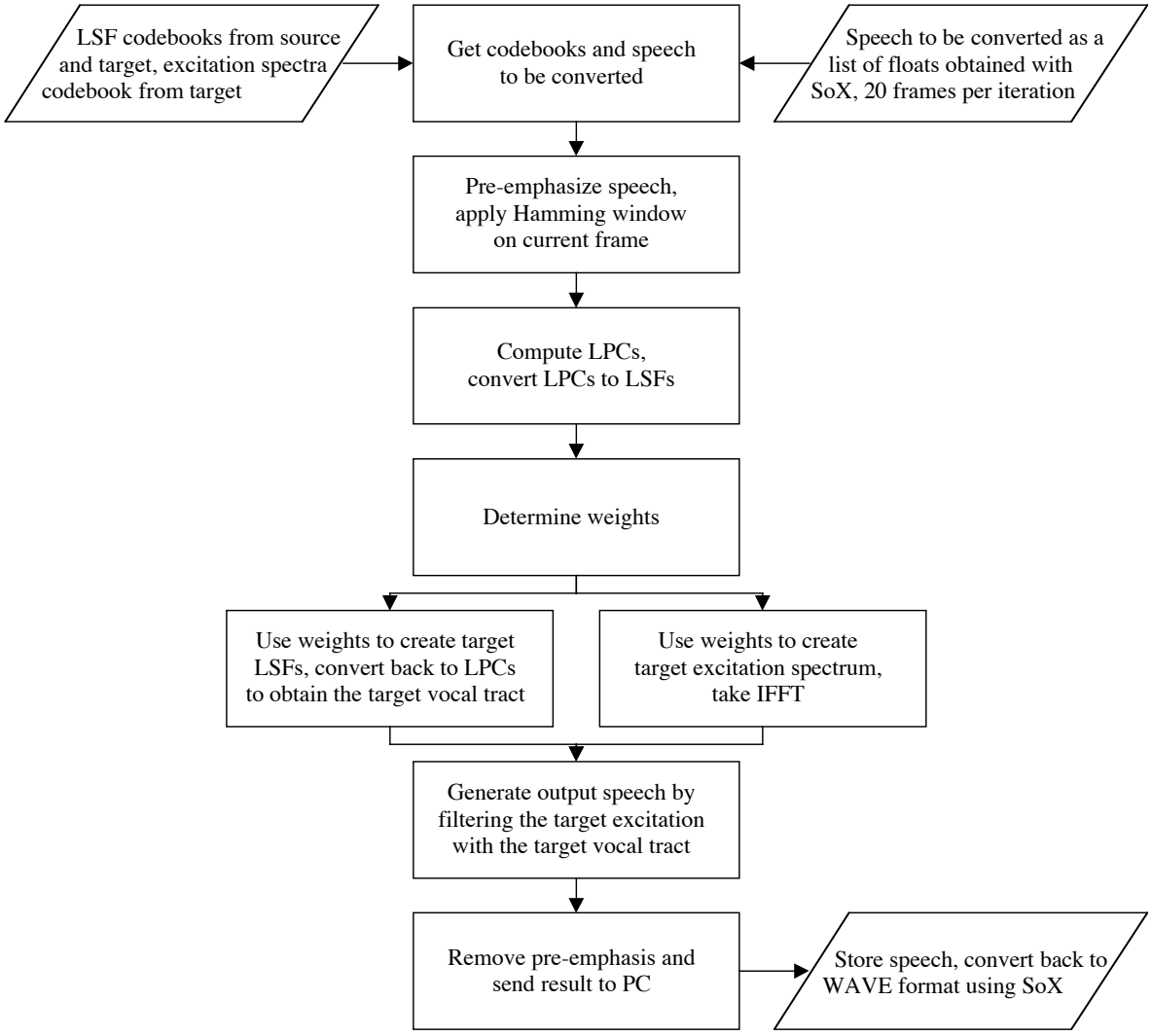

Figure 6: Flowchart of the conversion process

4.3 Attempted Improvements

The output speech from our system had a disappointing sound quality. To correct this we decided to implement the Gradient Descent and Bandwidth Modification functions proposed in [1] into the conversion part. However, there was no noticeable improvement in sound quality so we decided not to include these functions.

On Thursday of the last week we unexpectedly came upon a patent by Arslan and Talkin describing their voice conversion technique in much more detail. In this document we noticed an additional part on prosodic transformation that was not included in the 97 publication we had based our project on. This section was described as being an important part in allowing the target speaker to match the source speaker's pitch, duration, and stress. The algorithm proposed to perform this transformation was called Pitch-Synchronous Overlap-Add (PSOLA). Our intent was to implement it in MATLAB, test our system results, and if it improved our sound quality then implement it in C-code. However, after noticing the complexity of this algorithm, finding no suitable MATLAB or C-code for this method, and due to time constraints we were unable to implement this into our voice conversion system.

We did, however, attempt to implement our project in real-time. Our C-code on the EVM was successfully tailored to allow our conversion process to accept microphone input and send the converted results directly to line out. This attempt failed because our conversion process was not fast enough to work with the interrupts. This was due to the fact that interrupts only worked at level one optimization, which meant that any loop unrolling and/or pipelining would not be done to speed up our conversion process.

5. Data Flow

On the PC we perform the training part of our voice conversion system while the entire conversion part is done on the EVM. The codebooks obtained in the training part are sent to the EVM via an HPI transfer and are stored in the on-chip except for the target excitation codebook, which is kept in the external memory. The speech to be converted is recorded at 16 kHz, saved as a .wav file and converted to a list of floats using SoX (Sound eXchange). This list is divided into chunks of 20 frames, sent to the EVM via an HPI transfer, and stored in the on-chip memory.

On the EVM side the conversion process is performed one frame at a time until the entire chunk sent by the PC is processed. The resulting converted speech chunk is sent back to the PC via an HPI transfer. This process continues until all speech chunks sent by the PC have been converted. The final converted signal is stored on the PC as a list of floats. We then use SoX to change from this raw data format to .wav format, which can be played using Windows Media Player.

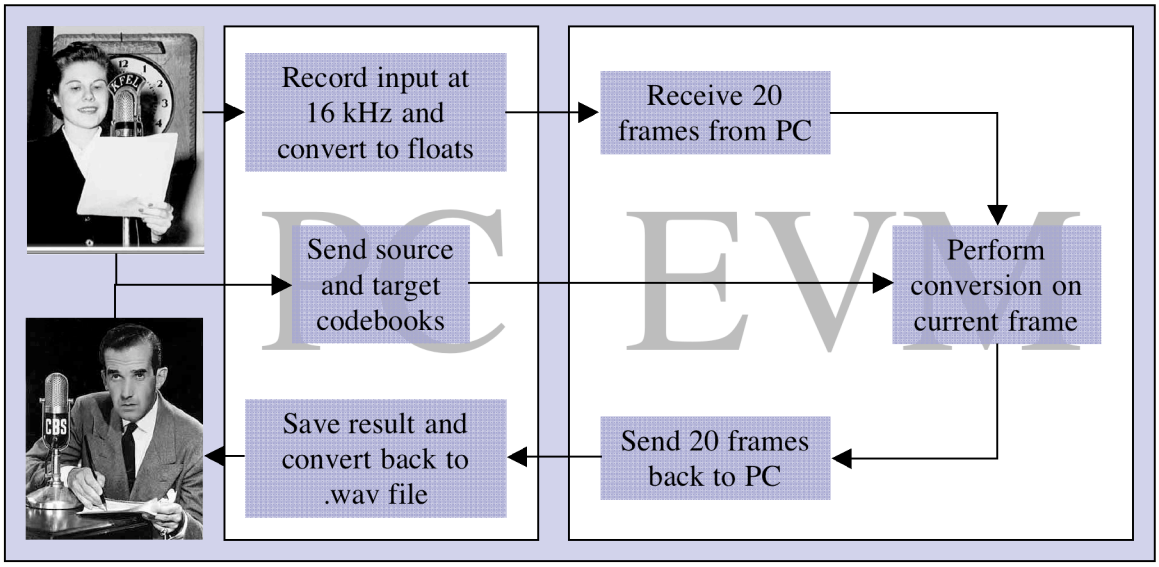

Figure 7: Data Flow between PC and EVM

6. Speed and Memory

The speed of our conversion process is substantially affected by the memory location of the data being used. On-chip memory allows for access time fifteen times faster than that of external memory. Nevertheless, we had to place our excitation codebook on the external memory as a result of its huge size (52 x 256 x sizeof(float)) and access its contents when needed through DMA transfers. The remaining data, including the 20 speech frames being processed, is stored in the on-chip memory (~62k). The size of the C-code is 41k and is also stored on the on-chip memory.

We implemented the gradient descent weight update and the bandwidth modification algorithm. However we noticed that both did not improve the sound quality, and thus decided to leave them out. This allowed us to fit our code in the on-chip memory and to reduce the calculations needed to be performed during the conversion process. We also increased the speed of our conversion process by only using the codebook entries with the highest weights to compute the target LSF and excitation spectrum. This optimization alone considerably improved performance:

Before optimization	After optimization
2,827,772 cycles per speech frame	1,189,671 cycles per speech frame

7. Results

To test our system, we created 8 different codebooks, with 4 male and 4 female voices. We also recorded a different sentence from each speaker that was not represented in the training part. We then converted those sentences from one speaker to another.

While the output was always intelligible and the voice was clearly different, we were disappointed by the sound quality. The output sounded rather robotic and noisy, which was caused by our regenerated excitation signal. If we just change the LSF coefficients, but keep the original excitation signal, the sound quality is as good as the original, but the voice doesn’t change in this case. This is because the pitch is determined by the vocal cords, i.e. the excitation signal. It also shows that the estimation of the target LSF worked successfully.

The poor quality of the excitation signal is caused by the fact that it is created by aligning linear combinations of codebooks excitations one after another. This can lead to sudden differences in amplitude and pitch from frame to frame. The original algorithm used PSOLA (Pitch-Synchronous Overlap-Add) and energy scaling to adjust pitch and amplitude respectively. We found that pitch shifting through PSOLA had been a major part of a 551 project in 1999 [9], and decided not to implement it. However we did adjust the amplitude of the output to match the amplitude of the input, as suggested by Efstratios, which improved the quality significantly.

Another reason might be that the original algorithm by Arslan and Talkin used much more detailed codebooks containing the beginning, middle and closing characteristics of each phoneme. We were not able to implement this entirely because TIMIT only makes this distinction for some phonemes (the “cl” suffix means closure, see appendix).

Subjective listening tests showed that the target speaker could be identified in most cases despite the unpleasant quality, even by persons outside of our group. In some male-to-male conversion cases, the target voice wasn’t identifiable. This may be due to the fact that we not only perceive voice by physical attributes such as pitch and vocal tract, but also by characteristics like emphases and intonations on certain phonemes, which our system cannot change. In general, the smaller the differences between source and target voice, the harder it was to recognize the voice of the output. Cross-gender conversions worked best due to greater differences in pitch between men and women.

During the lab demo, Prof. Casasent completed the training part and created his own codebook. The additional sentence we recorded was “18-551 is a cool class”. We then converted this sentence to a female voice. As expected, the result sounded robotic, but was very easy to understand. The voice sounded the same as previous conversions from other speakers to the same target voice, which had been correctly identified by the target as “That’s me”. This shows that the output voice does not depend on the input.

We also converted a female voice to Prof. Casasent’s voice. Here the source sentence was “551 is a really great class” (note the difference with the previous sentence ;-)). Again, the result was perfectly understandable; the voice was definitely male and could be identified as Prof. Casasent’s voice.

8. Conclusion

This project made us learn a lot about speech modeling, synthesis and modification. We got hands-on experience with standard algorithms such as Linear Predictive Coding and Dynamic Time Warping, and got an insight into current research by talking to faculty members and reading research papers. We gained a solid knowledge of DSP programming and improved our familiarity with MATLAB. Last but not least, we learned to remain cautious when reading research papers…

While our system might not be useable for most of the applications we mentioned, it certainly worked for one application: Having fun!

9. Acknowledgements

We would like to thank the following persons for helping us with our project:

- *Prof. David Casasent* for helping us understand the original algorithm
- *Prof. Richard Stern* for suggesting the use of the TIMIT database and helping us improve our results
- *Prof. Tom Sullivan* for his advice
- *Rohit Patnaik* for helping with EVM related issues
- *Xun Zhang* for coming to the lab almost every night to help everybody
- *Evandro Gouvea* of the ECE department for helping us solve the mystery of the vocal tract model
- *Xiang Li* for indicating us how to read the ADC sound files on the TIMIT CD
- *Afshan, Monisha* and *Janie* from Group 10, *Steve* from Group 11 and *Sky Johnson* for their voices

We would especially like to thank *Efstratios Papadomanolakis* for the time he spent with us to fix the problems we had. We really appreciated his patience, kindness and generosity. He is an exceptional TA!

10. References

[1] "Voice Conversion by Codebook Mapping of Line Spectral Frequencies and Excitation Spectrum", L. Arslan & D. Talkin, in Proc. Eurospeech '97, pp. 1347-1350

[2] "Speaker Transformation Algorithm using Segmental Codebooks (STASC)", L. Arslan, Technical Report, Washington D.C., 1998

[3] "Speaker Transformations using Sentence HMM Based Alignments and Detailed Prosody Modification", L. Arslan, in IEEE Proc. ICASSP, Seattle, May 1998

[4] "Text Independent Voice Transformation", 18-551 Group 18, Spring '99

[5] "Linear Predictive Modeling of Speech - Constraints and Line Spectrum Pair Decomposition", Tom Bäckström, Dissertation of the Helsinki University of Technology, 2004

[6] "The Distance Measure for Line Spectrum Pairs Applied to Speech Recognition", Fa. Zheng, Z. Song, L. Li, W. Yu, Fe. Zheng, W. Wu, in ICSLP '98, pp. 1123-1126

[7] "Dynamic Time Warp (DTW) in Matlab", Dan Ellis, Columbia University, http://labrosa.ee.columbia.edu/matlab/dtw/

[8] SPTK Speech Processing Toolkit, Nagoya Institute of Technology, Japan, http://kt-lab.ics.nitech.ac.jp/~tokuda/SPTK/

[9] "DSP-Enhanced Karaoke", Gavin Haentjens, Zalifah M. Sahir and Marina M. Yusoff, 18-551 Group 1, Spring 1999

Appendix

File: phoncode.doc, updated 10/12/90

This file contains a table of all the phonemic and phonetic symbols used in the TIMIT lexicon and in the phonetic transcriptions. These include the stress markers {1,2} found only in the lexicon and the following symbols which occur only in the transcriptions:

1) the closure intervals of stops which are distinguished from the stop release. The closure symbols for the stops b,d,g,p,t,k are bcl,dcl,gcl,pcl,tck,kcl, respectively. The closure portions of jh and ch, are dcl and tcl.

2) allophones that do not occur in the lexicon. The use of a given allophone may be dependent on the speaker, dialect, speaking rate, and phonemic context, among other factors. Since the use of these allophones is difficult to predict, they have not been used in the phonemic transcriptions in the lexicon.

- flap dx, such as in words "muddy" or "dirty"

- nasal flap nx, as in "winner"

- glottal stop q, which may be an allophone of t, or may mark an initial vowel or a vowel-vowel boundary

- voiced-h hv, a voiced allophone of h, typically found intervocalically

- fronted-u ux, allophone of uw, typically found in alveolar context

- devoiced-schwa ax-h, very short, devoiced vowel, typically occurring for reduced vowels surrounded by voiceless consonants

3) other symbols include two types of silence; pau, marking a pause, and epi, denoting epenthetic silence which is often found between a fricative and a semivowel or nasal, as in "slow", and h#, used to mark the silence and/or non-speech events found at the beginning and end of the signal.

	SYMBOL	EXAMPLE WORD	POSSIBLE PHONETIC TRANSCRIPTION
Stops:			
	b	bee	BCL B iy
	d	day	DCL D ey
	g	gay	GCL G ey
	p	pea	PCL P iy
	t	tea	TCL T iy
	k	key	KCL K iy
	dx	muddy, dirty	m ah DX iy, dcl d er DX iy
	q	bat	bcl b ae Q
Affricates:			
	jh	joke	DCL JH ow kcl k
	ch	choke	TCL CH ow kcl k

Fricatives:

s	sea	S iy
sh	she	SH iy
z	zone	Z ow n
zh	azure	ae ZH er
f	fin	F ih n
th	thin	TH ih n
v	van	V ae n
dh	then	DH e n

Nasals:

m	mom	M aa M
n	noon	N uw N
ng	sing	s ih NG
em	bottom	b aa tcl t EM
en	button	b ah q EN
eng	washington	w aa sh ENG tcl t ax n
nx	winner	w ih NX axr

Semivowels and Glides:

l	lay	L ey
r	ray	R ey
w	way	W ey
y	yacht	Y aa tcl t
hh	hay	HH ey
hv	ahead	ax HV eh dcl d
el	bottle	bcl b aa tcl t EL

Vowels:

iy	beet	bcl b IY tcl t
ih	bit	bcl b IH tcl t
eh	bet	bcl b EH tcl t
ey	bait	bcl b EY tcl t
ae	bat	bcl b AE tcl t
aa	bott	bcl b AA tcl t
aw	bout	bcl b AW tcl t
ay	bite	bcl b AY tcl t
ah	but	bcl b AH tcl t
ao	bought	bcl b AO tcl t
oy	boy	bcl b OY
ow	boat	bcl b OW tcl t
uh	book	bcl b UH kcl k
uw	boot	bcl b UW tcl t
ux	toot	tcl t UX tcl t
er	bird	bcl b ER dcl d
ax	about	AX bcl b aw tcl t
ix	debit	dcl d eh bcl b IX tcl t
axr	butter	bcl b ah dx AXR
ax-h	suspect	s AX-H s pcl p eh kcl k tcl t

Others:

SYMBOL	DESCRIPTION
pau	pause
epi	epenthetic silence
h#	begin/end marker (non-speech events)
1	primary stress marker
2	secondary stress marker